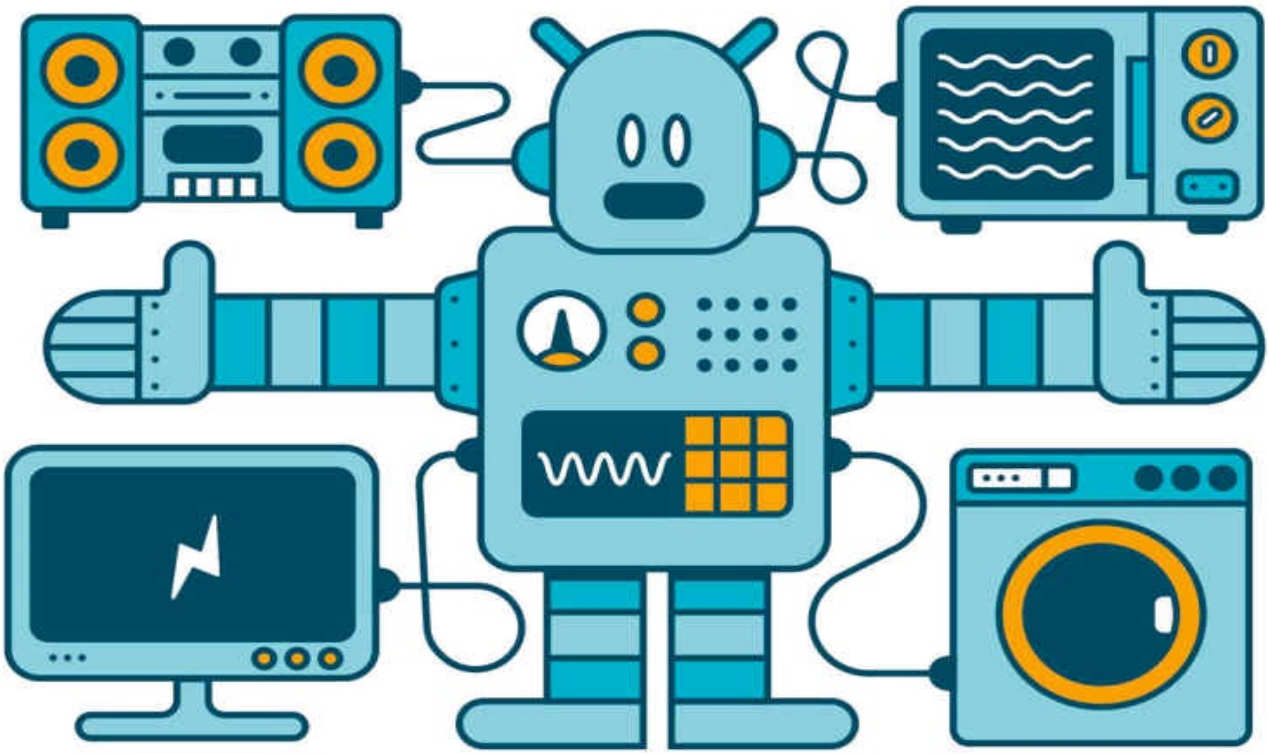
PIER CALDERAN

Il manuale del Maker domestico



Progetti di domotica DIY con Arduino, Raspberry Pi e Windows 10 IoT

APOGEO

Il manuale del Maker domestico
Progetti di domotica DIY con Arduino, Raspberry Pi e Windows
10 IoT

Pier Calderan

APOGEO

ISBN edizione cartacea: 9788850334100

~

Rimani aggiornato iscrivendoti alla nostra newsletter

Introduzione

Archimede, Erone, Leonardo, Galileo, Volta, Tesla, Edison, Meucci, Marconi… quanti sono i maker che hanno fatto grande la storia della scienza e della tecnica? E quanti ce ne saranno? Be', almeno uno in più, dopo la lettura di questo manuale.

Grazie a questo libro, gli amici e i parenti che verranno a casa vostra si sentiranno un po' come sul ponte di comando di Star Trek.

I vostri ospiti dovranno pronunciare una parola per entrare in una stanza segreta. Potranno ascoltare musica in ogni stanza, selezionandola dal tablet o dallo smartphone. Senza il riconoscimento dell'impronta digitale o di un badge non potranno entrare in casa. Andando verso lo specchio vedranno riflesse anche le previsioni meteo e l'orario. Salendo sulla bilancia sentiranno una voce che dirà loro il peso. Non ci sarà più la preoccupazione di annaffiare le piante quando si va in vacanza.

Tutto questo non è fantascienza. È possibile realizzando qualche progetto fatto con pochi componenti elettronici e un po' di passione.

Il maker che sa già armeggiare con le piattaforme hardware e software proposte in questo libro potrà saltare direttamente ai capitoli della seconda parte, dedicata ai progetti. Il neofita con poca o nessuna esperienza, che vuole comunque diventare un maker di tutto rispetto, è invitato ad "assorbire" tutto il libro fin dal primo capitolo. A tutti, auguriamo buona lettura!

Struttura del libro

Parte I

- Il Capitolo 1, *Elettronica e meccanica per maker*, è dedicato alle nozioni di base dell'elettronica, con riferimenti ai componenti attivi e passivi, a cosa serve nel laboratorio del maker e agli attuatori meccanici usare per alcuni progetti hardware.
- Il Capitolo 2, *Schede hardware*, è una panoramica delle schede più usate nel mondo dei maker, con particolare riferimento ad Arduino e Raspberry Pi.
- Capitolo 3, *Ambienti di programmazione*. Per una maggior comprensione dei listati usati nei progetti del libro e per poterli scrivere con maggior facilità, questo capitolo illustra gli ambienti di programmazione che fanno parte del bagaglio di conoscenze del maker: dall'IDE di Arduino a Python per Raspberry Pi, fino a Visual Studio 2017 e altro ancora.
- Il Capitolo 4, *Piattaforme IoT e Cloud*, è dedicato alle piattaforme IoT e cloud che vengono messe a disposizione dei maker per connettere i dispositivi elettronici al Web.

Parte II

- Il progetto *Monitoraggio meteo* del Capitolo 5 serve a monitorare alcune condizioni meteo, ovvero temperatura, umidità e luce, usando Arduino e il modulo Wi-Fi ESP8266.
- Con il Capitolo 6,*Irrigazione intelligente*, si può avere cura del proprio giardino anche quando si va in vacanza, usando Arduino e

qualche sensore.

- Il progetto *Serratura con impronta digitale* del Capitolo 7 impedisce l'ingresso alle persone non autorizzate tramite Arduino e un sensore di impronta digitale.
- Il progetto *Sistema di allarme* del Capitolo 8 è un semplice circuito che rileva la presenza di un corpo umano e attiva una sirena o l'invio di email e di SMS, tramite Arduino e un servizio cloud IoT.
- Con il Capitolo 9, *Controllo RFID* si può costruire un semplice sistema di controllo degli accessi, usando Arduino e un modulo RFID.
- Il progetto *Apertura cancello da smartphone* del Capitolo 10 impiega un modem Bluetooth e Arduino per poter aprire un cancello a distanza usando il proprio smartphone.
- Con il Capitolo 11 arriva un po' di relax con il progetto *Musica in casa* che sfrutta le potenzialità multimediali di Raspberry Pi.
- Il progetto *Videosorveglianza* del Capitolo 12 impiega una scheda Raspberry Pi e la sua videocamera ad alta risoluzione per intercettare il movimento di qualsiasi persona in una stanza e per inviare un messaggio email di allarme.
- Con il Capitolo 13, *Apriti Sesamo*, si può comandare l'apertura di una porta o controllare qualsiasi altro dispositivo usando Arduino e un modulo di riconoscimento vocale.
- Nel Capitolo 14, grazie alle potenzialità di Raspberry Pi, il progetto *Specchio magico* trasforma lo schermo di un monitor o di un apparecchio TV in uno specchio in cui, oltre alla propria immagine riflessa, si possono vedere sovrapposte le notizie del giorno, l'orario e qualsiasi informazioni dal Web.
- Chiude la carrellata di progetti il Capitolo 15, *Bilancia intelligente*, che oltre a mostrare il peso sullo smartphone, possiede una voce che ci dice se siamo in sovrappeso.

File degli esempi

I file di esempio e i diagrammi e le immagini a colori del testo sono disponibili sul sito dell'autore all'indirizzo http://www.pierduino.com.

Parte I

Strumenti e ambiente di lavoro

In questa parte:

Elettronica e meccanica per maker

In questo libro si affronterà la costruzione di circuiti elettronici che utilizzano componenti sia analogici sia digitali. Per avere un quadro completo della situazione, sarà pertanto necessaria al maker la conoscenza del concetto di “analogico” e di “digitale”.

Prendendo come esempio una diffusa metafora, un segnale analogico può essere paragonato al profilo di uno scivolo o di una retta inclinata, mentre un segnale digitale può essere paragonato a quello di una scala o di una linea a gradini (Figura 1.1a).

Si intuisce immediatamente che la linea continua rappresenta un insieme di valori infiniti, nel senso di “non definibili” con precisione. La linea a gradini rappresenta invece un insieme di valori finiti, ovvero “distinguibili” come numero di gradini.

In altre parole, per descrivere una posizione sulla rampa la si può indicare solo in modo grossolano, usando un’analogia del tipo “all’inizio, a metà, a tre quarti e così via”, mentre nella scala a gradini si può indicare la posizione con precisione, specificando il numero esatto corrispondente del gradino.

La differenza fra andamento “analogico” e andamento “numerico” sta proprio nel senso di continuità dalla rampa, contro il senso di precisione offerto dalla scala a gradini.

I valori continui vengono pertanto definiti “analogici”, mentre i valori numerici vengono definiti “digitali”, dal latino *digitus*, che significa dito della mano, con cui si contano i numeri.

Nel mondo digitale, spesso si usa il termine “discreto”, inteso come participio passato del verbo *discernere*, cioè distinguere. Questo ci porta a enunciare un concetto fondamentale alla base della tecnologia analogica e di quella digitale:

- analogico è sinonimo di continuo, non analizzabile con precisione;
- digitale è sinonimo di discreto, analizzabile con precisione.

Nell’elettronica analogica vengono usati circuiti, per lo più basati su transistor o su circuiti integrati analogici, che trattano segnali in cui i valori di tensione variano in continuazione entro un intervallo di ampiezza minimo e massimo.

Per esempio, un tipico circuito analogico è quello di un amplificatore usato nel settore audio: il suono in uscita dagli altoparlanti viene prodotto da una tensione applicata ai capi dell’altoparlante, il quale si sposta in “analogia” al segnale elettrico applicato al suo ingresso, per esempio quello amplificato di un microfono o di un’altra sorgente sonora. Quello che succede è che il debole segnale del microfono (o dell’altra sorgente sonora) attraversa uno stadio di amplificazione che produce una tensione variabile sui terminali dell’altoparlante. Questo potrà muovere l’aria circostante e rendere udibili i suoni.

Nell’elettronica digitale vengono impiegati invece circuiti integrati, microprocessori e altri dispositivi “logici”, in grado di trattare segnali digitali basati su due soli stati di tensione, 0 V e 5 V (Figura 1.1b).

Rimanendo nell’ambito audio, un tipico circuito digitale chiamato ADC (*Analog to Digital Converter* = convertitore analogico digitale) è in grado di trasformare in numeri, ovvero “digitalizzare” il segnale analogico proveniente da un microfono (o altra sorgente) in una serie di impulsi di tensione di 0 e 5 volt (Figura 1.1c). Questi impulsi digitali possono venire facilmente memorizzati su un supporto come un disco rigido, un chip di memoria, un CD, un DVD, una chiavetta USB e così via.

Per facilitare la comprensione dei segnali digitali vengono usati gli stati logici “basso” e “alto”, corrispondenti rispettivamente a “bit 0” e “bit 1”. Ricordiamo che *bit* è la contrazione dei termini *binary digit*, ovvero numero binario. In una conversione analogico-digitale si otterrà un segnale digitalizzato come quello illustrato nella Figura 1.1d. Aumentando la risoluzione in bit si potrà ottenere una forma d’onda digitalizzata più fedele all’originale (Figura 1.1e). Spesso viene usato come sinonimo di digitalizzazione il termine “campionamento”, inteso come conversione dei dati analogici in “campioni” digitali.

Nel caso di un campionamento audio, i dati memorizzati su un supporto digitale (per esempio, un CD) non possono venire riprodotti direttamente da un altoparlante. Questi dati vanno “letti” da un dispositivo digitale (lettore CD) e convertiti in segnali analogici. Il circuito digitale responsabile di questo si chiama DAC (*Digital to Analog Converter*) ed è contenuto nel dispositivo di lettura del CD. Il DAC è in grado di ricostruire il segnale analogico fornendo a un amplificatore una serie di tensioni analogiche variabili, che potranno poi essere riprodotte da un altoparlante.

Per fare un esempio, il segnale audio analogico, normalmente percepito in un intervallo di frequenza compreso fra 20 Hz e 20 kHz deve essere digitalizzato (campionato) a una frequenza di almeno 44,1 kHz e 16 bit di risoluzione per poter essere percepito con una buona qualità. I normali CD audio sono prodotti in base a questo standard. Più sono elevate la frequenza di campionamento e la risoluzione in bit, più il segnale riconvertito in analogico sarà simile a quello originale.

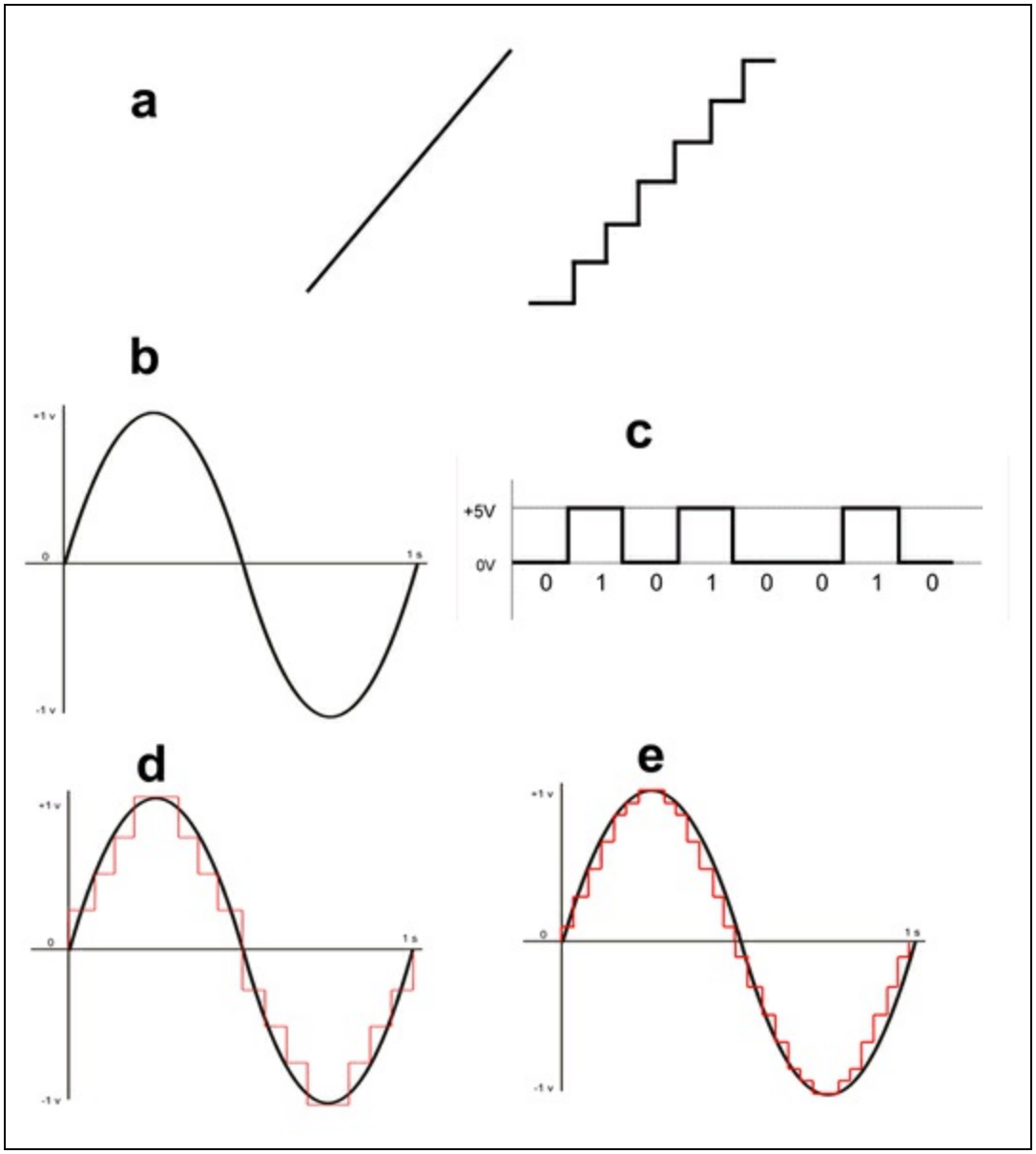


Figura 1.1 Lo scivolo e la scala a gradini per rappresentare un andamento analogico e digitale (a). Segnale con il tipico andamento analogico (b). Segnale digitale costituito da una serie di bit (c). Segnale analogico digitalizzato a una risoluzione di 16 bit (d). Segnale analogico digitalizzato a una risoluzione di 24 bit (e).

Bisogna dire che, quasi sempre, elettronica digitale e analogica convivono. Anche nel più sofisticato dispositivo digitale ci sarà sempre la necessità di comunicare dati ai sensi umani. Per cui, da una parte ci sarà bisogno di sensori in grado di catturare luce, movimento, suono,

magnetismo e così via. Questi dati analogici verranno campionati e trattati in digitale e, in molte occasioni, riconvertiti in analogico per poter essere percepiti dai nostri sensi.

In questo libro verranno trattati progetti costituiti da circuiti analogici, circuiti digitali e circuiti ibridi. Si imparerà a costruire dei semplici circuiti analogici che potranno far funzionare circuiti digitali e circuiti digitali in grado di convertire i dati digitali in segnali analogici.

Elettronica di base

La corrente elettrica è un flusso di “elettroni”, cioè di quelle particelle che costituiscono la parte esterna di tutti gli atomi della materia di cui è composto l’universo.

L’atomo è costituito da un nucleo di protoni dotati di carica positiva e di neutroni, privi di carica. Attorno al nucleo ruotano alla velocità della luce gli elettroni che, essendo dotati di carica negativa (-), devono sfuggire all’attrazione della carica positiva (+) dei protoni del nucleo (Figura 1.2a). I neutroni non hanno nessun ruolo in questa azione di attrazione.

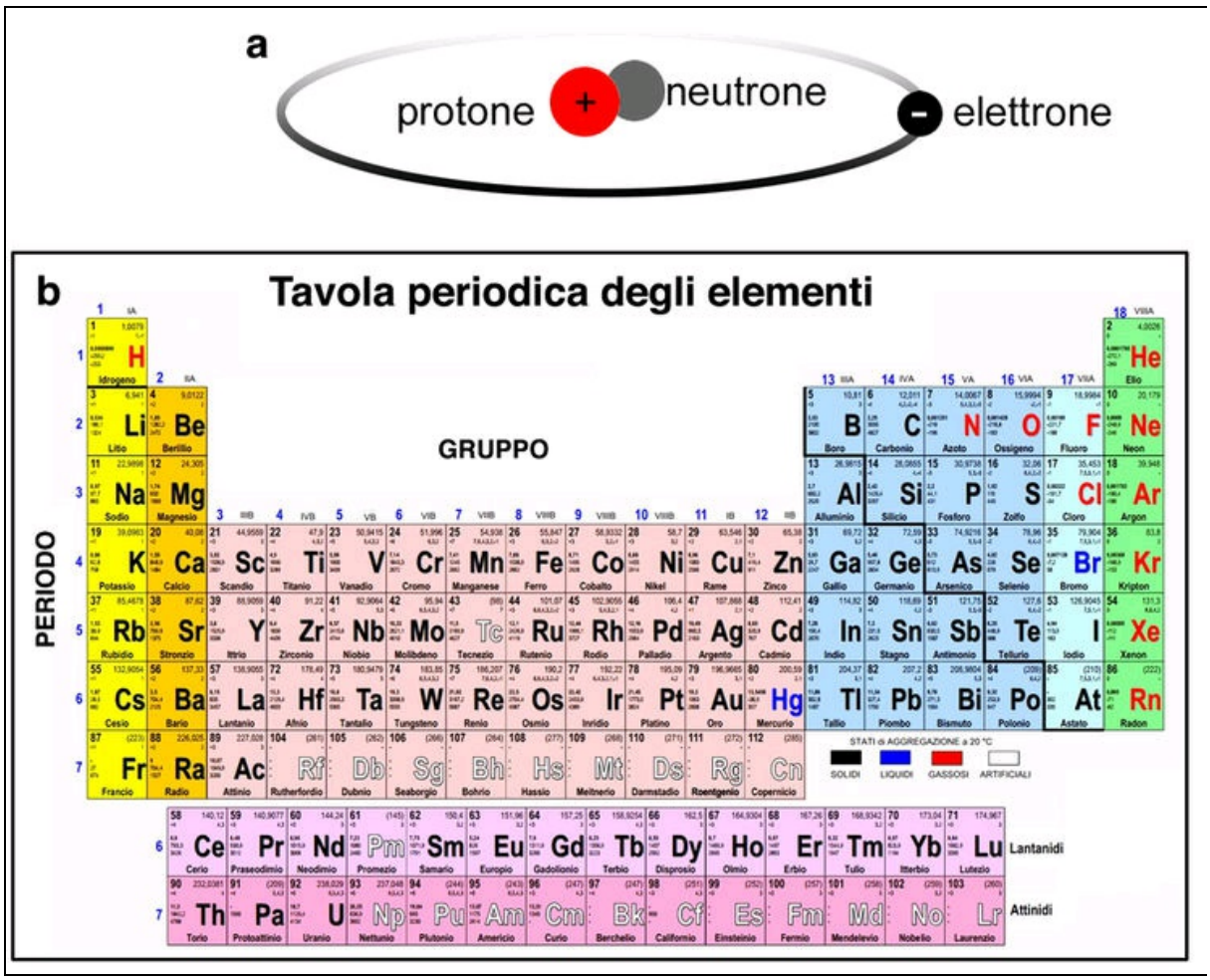


Figura 1.2 L’atomo è formato da protoni (+), neutroni (carica nulla) ed elettroni (-) in perfetto equilibrio energetico.

Ogni atomo, a seconda dell’elemento cui appartiene, ha un numero ben definito di protoni, neutroni ed elettroni. In condizioni normali questo numero è sempre uguale e l’atomo si trova quindi in equilibrio perfetto, con la carica positiva e quella negativa che si controbilanciano a livello energetico. Ecco, per esempio il “numero atomico” di alcuni elementi.

- *Idrogeno*: 1 protone, 1 neutrone, 1 elettrone.
- *Elio*: 2 protoni, 2 neutroni, 2 elettroni.
- *Rame*: 29 protoni, 29 neutroni, 29 elettroni.
- *Argento*: 47 protoni, 47 neutroni, 47 elettroni.
- *Oro*: 79 protoni, 79 neutroni, 79 elettroni.

Tutti gli elementi chimici sono catalogati nella cosiddetta tavola periodica (Figura 1.2b), nella quale sono disposti in base al loro numero atomico, per gruppo (colonna) e periodo (riga) di appartenenza.

Alcuni elementi sono portati, naturalmente, ad acquisire o perdere elettroni. Per esempio, se si strofina una bacchetta di plastica su un indumento di lana, si crea uno squilibrio di elettroni nella bacchetta di plastica per cui questa è in grado di attrarre piccoli pezzi di carta o capelli. Gli antichi greci si accorsero che questo accadeva facilmente con l’ambra, che chiamavano *electron*. Questo fenomeno venne chiamato triboelettrico (*tribos* = strofinio). Pertanto, dall’antico termine greco dell’ambra deriva il nostro termine “elettronica”.

Nella pratica, in tutti i circuiti elettrici, la perdita di elettroni di un elemento provoca uno squilibrio di cariche che deve sempre essere compensato. Un atomo con elettroni in eccesso viene chiamato *ione negativo* o *anione*. Un atomo con protoni in eccesso, ovvero che ha perso elettroni, viene chiamato *ione positivo* o *catione*.

I terminali di un generatore di corrente elettrica, per esempio quelli di una pila, si chiamano elettrodi (dal greco *odos*=strada, quindi “strada degli elettroni”).

Per cui si avrà:

- *anodo*, ovvero l’elettrodo che attira gli anioni (elettrodo positivo);
- *catodo*, ovvero l’elettrodo che attira i cationi (elettrodo negativo).

Di solito si è portati a pensare che il flusso della corrente elettrica scorra dal polo positivo verso il polo negativo, in virtù di una convenzione ormai diffusa che definisce il verso della corrente convenzionale come la direzione del flusso di carica positiva. Tale convenzione si deve a Benjamin Franklin. Al contrario, il flusso della corrente elettrica scorre sempre dal polo negativo verso il polo positivo. Questo perché sono sempre gli ioni positivi che attirano gli elettroni per ristabilire l’equilibrio di energia dell’atomo. Nelle applicazioni pratiche, il verso della corrente è importante per il corretto funzionamento dei circuiti elettronici, mentre ha un’importanza minore nei circuiti elettrici.

Tensione elettrica

Per spiegare cosa sia la tensione elettrica, partiamo da un semplice assunto:

- le cariche elettriche di uguale polarità si respingono reciprocamente;
- le cariche elettriche di polarità inversa si attraggono reciprocamente.

Essendo elettricità e magnetismo strettamente legati assieme, possiamo fare un semplice esperimento con due calamite. Se proviamo ad avvicinare il polo positivo di una calamità al polo positivo dell’altra, avvertiremo una certa resistenza e non riusciremo a unire i due poli. Al

contrario, il polo positivo di una calamità attirerà il polo negativo dell'altra, unendosi a questa con una certa energia.

La calamita è in grado di attirare anche gli oggetti metallici perché gli atomi di questi ultimi sono in equilibrio atomico e gli elettroni più esterni vengono attratti dai protoni in eccesso della calamità, considerato che sono i protoni (le cariche positive) che attraggono gli elettroni (cariche negative).

Osservando i terminali di una comune pila (Figura 1.3), noteremo sempre un polo contrassegnato con il segno positivo (+) e un polo contrassegnato con il segno negativo (-). Significa che, in una situazione normale, la pila possiede un eccesso di ioni positivi sul polo positivo e, viceversa, un eccesso di ioni negativi sul polo negativo.

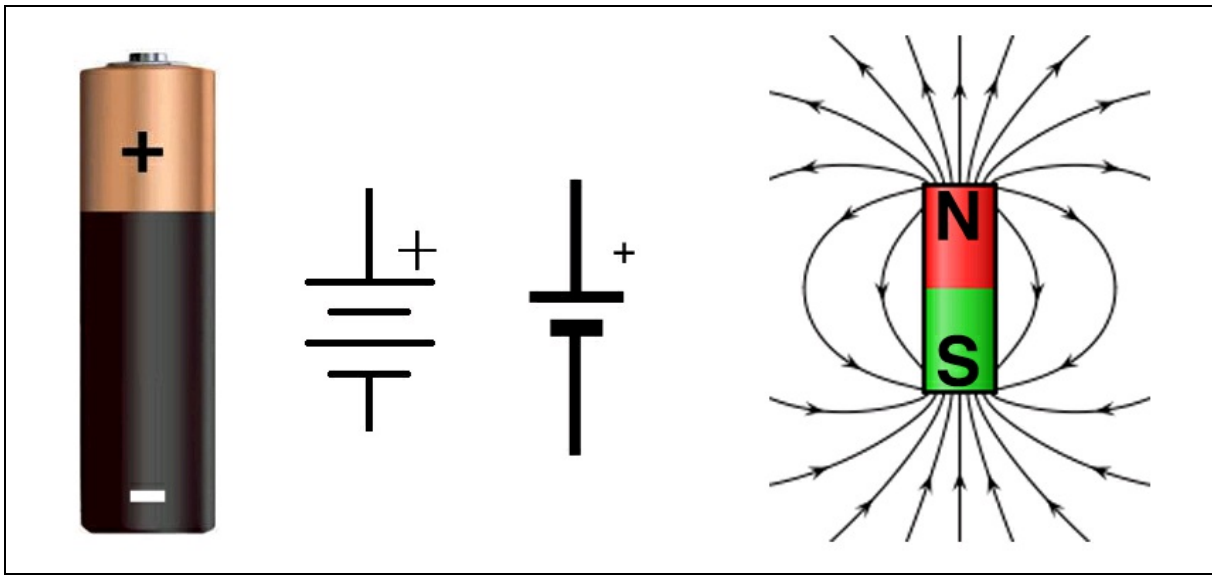


Figura 1.3 Una comune pila da 1,5 volt con le indicazioni del polo positivo e del polo negativo. A destra, i simboli elettrici più comunemente usati negli schemi elettrici e il paragone con una calamita.

Lo squilibrio di cariche positive e cariche negative di una pila o di un generatore di corrente elettrica viene chiamato "differenza di potenziale". Fra due poli con differente potenziale elettrico si genera pertanto la cosiddetta "tensione" elettrica.

L’unità di misura della tensione è il *volt* (simbolo V), in onore di Alessandro Volta (Como, 1745 - Camnago Volta, 1827), l’inventore della prima pila costruita su base elettrochimica.

Se colleghiamo i due terminali della pila con un filo di materiale conduttore, per esempio di rame, gli elettroni del polo negativo verranno attratti dal polo positivo. Questo movimento di elettroni genera la cosiddetta corrente elettrica e, come vedremo, anche un campo magnetico.

La corrente elettrica cesserà quando verrà ristabilito negli atomi l’equilibrio energetico tra i protoni e gli elettroni degli atomi che compongono gli elementi chimici usati per la pila. In altre parole, la pila si “scarica” con il passaggio di cariche negative verso il polo positivo. Il flusso di elettroni può essere quindi sfruttato per produrre la cosiddetta “energia elettrica”.

Per esempio, per produrre calore si fanno passare gli elettroni attraverso un materiale resistivo che oppone “resistenza” al flusso di elettroni.

Si può produrre luce se gli elettroni attraversano un sottile filamento, causando un attrito così forte da far diventare incandescente il filamento stesso (lampadina a incandescenza). Con l’energia degli elettroni si possono realizzare “elettrocalamite” quando gli elettroni vengono fatti passare attraverso un filo avvolto attorno a un pezzo di ferro o in una spirale in aria.

Si possono costruire circuiti elettronici in grado di “guidare” gli elettroni in modo che interagiscano con i componenti passivi e i componenti attivi per un’infinità di applicazioni: dalla semplice accensione di diodi LED al controllo digitale di circuiti integrati, di microprocessori e via dicendo.

Una pila (chiamata anche accumulatore) possiede un elettrodo positivo e un elettrodo negativo, perché al suo interno gli elementi

chimici presentano uno squilibrio di elettroni dovuto alla diversa ionizzazione.

Le pile possono essere di vario tipo e offrire tensioni diverse per poterle sfruttare opportunamente come sorgenti di alimentazione in vari dispositivi elettronici.

Il *volt* è l'unità di misura che misura lo squilibrio fra le cariche positive e negative, ovvero la differenza di potenziale elettrico. In altre parole, il volt misura la "forza" dei singoli elettroni che potenzialmente vengono attratti quando il circuito viene chiuso. Una pila da 1,5 volt ha un potenziale basso e può alimentare un circuito il cui assorbimento di energia è esiguo. Una pila da 9 volt ha un potenziale sei volte maggiore rispetto a una pila da 1,5 volt e può alimentare un circuito molto più esigente, in termini di energia.

Per comodità, la differenza di potenziale elettrico viene espressa anche con i multipli e sottomultipli del volt:

MV	Megavolt (10^{6})	milione di volt
kV	kilovolt (10^{3})	mille volt
mV	millivolt (10^{-3})	millesimo di volt
µV	microvolt (10^{-6})	milionesimo di volt

Corrente elettrica

Quando vengono messi in movimento in un circuito chiuso, gli elettroni che vanno dal polo negativo al polo positivo producono la cosiddetta *corrente elettrica*.

L'unità di misura della corrente elettrica è l'*ampere* (simbolo A), in onore di André-Marie Ampère (Poleymieux-au-Mont-d'Or, 1775 - Marsiglia, 1836), fisico francese che dedicò la sua vita allo studio dei fenomeni elettrici e fisici.

Giusto per dare un'idea del significato di corrente elettrica, 1 ampere corrisponde a circa $6{,}28^{-18}$ (6 280 000 000 000 000 000) elettroni che

scorrono dal polo negativo verso il polo positivo nel tempo di 1 secondo, ovvero 1 *coulomb* al secondo:

$$1\ A = 1\ C/1\ s$$

NOTA

Il coulomb è l'unità di misura della carica elettrica, così denominata in onore di Charles Augustin de Coulomb (Angoulême, 1736 - Parigi, 1806), scienziato francese che per primo studiò il moto delle cariche elettriche. Definito con il simbolo C nel sistema internazionale, un coulomb è pari alla quantità di carica elettrica trasportata in 1 secondo dalla corrente di 1 ampere: 1 C = 1 A x s.

Contrariamente a una cattiva abitudine, bisogna sempre prestare molta attenzione a maneggiare un circuito in cui scorre corrente elettrica. L'intensità della corrente elettrica non dipende dal valore di tensione. È possibile prelevare 1 A di corrente sia da una tensione di 1 volt sia dalla tensione di rete di 230 volt.

In modo analogo, un circuito che eroga una tensione di milioni di volt potrebbe produrre una corrente irrisoria di qualche milionesimo di ampere ed essere quindi del tutto innocua. Per esempio, la leggera scossa (pur sempre fastidiosa, ma non letale) che si percepisce sulle dita quando si apre la portiera dell'auto, è prodotta dalla scarica di corrente elettrostatica prodotta dallo sfregamento di tessuti plastici (effetto triboelettrico). La scintilla che talvolta si vede in questi casi viene prodotta dall'attrito degli elettroni che superano l'isolamento del materiale dielettrico (l'aria) per ristabilire l'equilibrio ionico "disturbato" dallo strofinamento. Si tratta di un piccolissimo fulmine che scarica a terra migliaia di volt con una corrente infinitesimale. Al contrario, i fulmini di un temporale scaricano a terra milioni di volt con correnti dell'ordine delle decine di migliaia di ampere.

La scarica elettrostatica quando si maneggiano circuiti elettronici può essere letale per i circuiti elettronici, specie quelli con schede di memoria e processori. Anche se la corrente è debole, le alte tensioni possono danneggiare irreparabilmente le loro delicate strutture interne.

Per evitare danni, è consigliabile “scaricarsi” a terra toccando un termosifone o un rubinetto metallico prima di maneggiare circuiti integrati, microchip o transistor.

Nell’elettronica digitale vengono usate tensioni basse e amperaggi molto limitati, per cui si usano solo i sottomultipli dell’ampere. I più usati sono i seguenti:

mA	milliampere (10^{-3})	millesimo di ampere
µA	microampere (10^{-6})	milionesimo di ampere

Nelle formule, la corrente elettrica viene indicata con la lettera I (lettera i maiuscola), corrispondente alla misura della *intensità* di corrente.

Corrente continua e corrente alternata

La corrente elettrica può essere di due tipi:

- corrente continua o diretta (pile, celle solari, alimentatori DC);
- corrente alternata (alternatori, trasformatori, alimentatori AC).

Questo significa che la tensione ai capi del generatore può produrre un flusso di elettroni continuo o alternato.

Gli elettrodi di una pila hanno sempre un conduttore con polarità negativa e uno con polarità positiva, quindi gli elettroni scorrono sempre in un’unica direzione, producendo una *corrente continua* (o corrente diretta) costante nel tempo.

La corrente continua si indica con la sigla CC (corrente continua) o DC (in inglese, *Direct Current*). La Figura 1.4a è una rappresentazione del flusso di corrente continua nel tempo. L’asse y rappresenta la tensione V. L’asse x rappresenta il tempo t. Nell’esempio la tensione è di 9 volt.

Ai capi di un cavo inserito in una presa della rete domestica o di un trasformatore, la corrente cambia di direzione in modo alternato, cioè

ognuno dei due poli cambia alternativamente la polarità, producendo così una *corrente alternata*.

La corrente alternata si indica con la sigla CA (corrente alternata) o AC (in inglese *Alternate Current*). L'inversione della polarità in un generatore di corrente alternata avviene alla frequenza di 50 hertz (simbolo Hz), ovvero di 50 periodi al secondo, seguendo un andamento sinusoidale per non causare bruschi cambiamenti di polarità. La Figura 1.4b è una rappresentazione del flusso di corrente alternata nel tempo. L'asse y rappresenta la tensione V. L'asse x rappresenta il tempo t. Nell'esempio, la frequenza è di 1 Hz (un ciclo al secondo). La frequenza di rete è di 50 Hz.

Potenza elettrica

La potenza elettrica è data dal prodotto di tensione e corrente. Conoscendo il valore di tensione di un qualsiasi generatore (pila, trasformatore, rete elettrica e così via) e la corrente prelevata per alimentare un circuito, è possibile calcolare il valore della "potenza" assorbita dal circuito, che in genere si traduce in "consumo" di corrente.

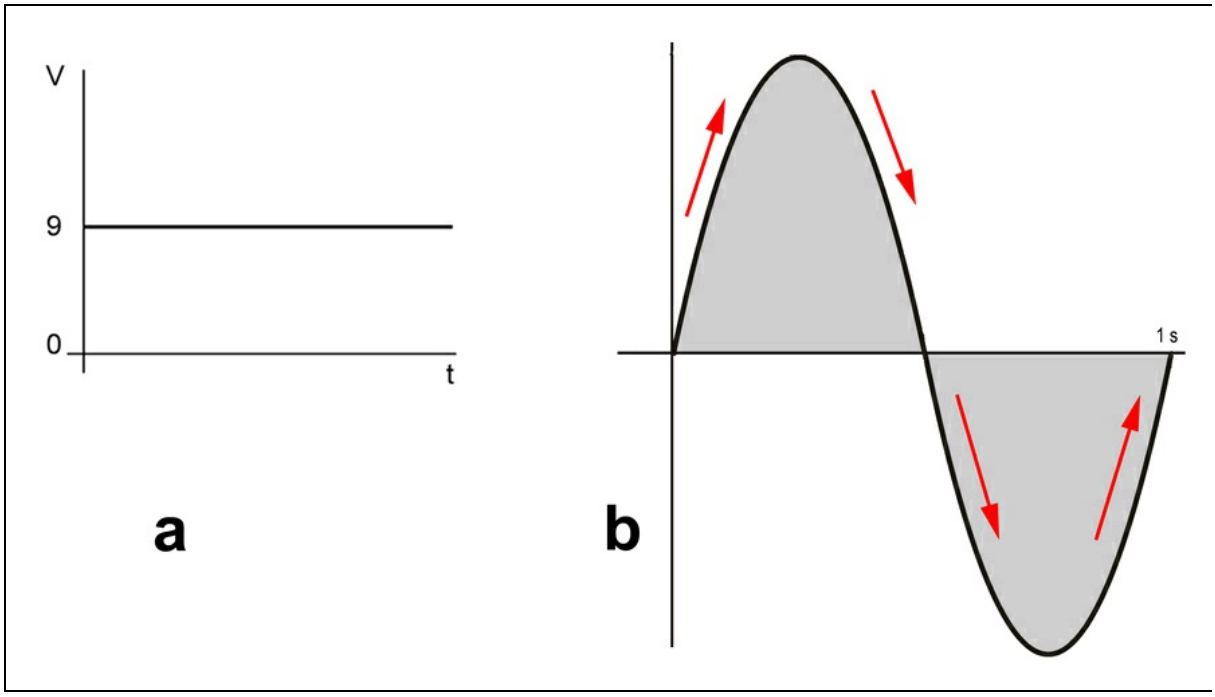


Figura 1.4 Corrente continua (a). Corrente alternata (b).

L’unità di misura della potenza elettrica è il *watt* (simbolo W), in onore di James Watt (Greenock, 1736 - Handsworth, 1819), ingegnere britannico che studiò le leggi fisiche e meccaniche derivate dalle macchine a vapore.

La formula per esprimere la potenza in watt è molto semplice:

watt = volt x ampere

Per esempio, un motore elettrico alimentato a 9 volt con una corrente circolante di 0,5 ampere assorbe una potenza elettrica di:

9 × 0,5 = 4,5 watt

Conoscendo la potenza assorbita in watt e la corrente circolante è possibile ricavare il valore della tensione di alimentazione tramite la formula inversa:

volt = watt / ampere

Conoscendo la potenza assorbita in watt e la tensione applicata è possibile ricavare il valore della corrente necessaria tramite la formula inversa:

ampere = watt / volt

Si noti che si usa il concetto di potenza anche per esprimere l'erogazione di energia elettrica. Un circuito di alimentazione può erogare un certo numero di watt, ma per farlo deve comunque consumare energia e, quindi, assorbire potenza dalla rete. Molta energia viene purtroppo persa nel processo di trasformazione e dispersa sotto forma di calore.

In elettrotecnica vengono usati i multipli del watt, mentre in elettronica è più comune usare il watt e i suoi sottomultipli. I più usati sono i seguenti:

MW	Megawatt (10^6)	milione di watt
kW	kilowatt (10^3)	mille watt
mW	milliwatt (10^{-3})	millesimo di watt
µW	microwatt (10^{-6})	milionesimo di watt

Resistenza elettrica

Normalmente la corrente elettrica è in grado di interessare tutti i corpi fisici, siano essi solidi, liquidi o gassosi. Ma non tutti i materiali sono buoni conduttori di elettricità. Esistono in natura elementi chimici che possiedono molti elettroni liberi. Per esempio, oro, argento, rame, alluminio, ferro e stagno sono ottimi conduttori elettrici.

Altri composti, come ceramica, vetro o plastica contengono pochi elettroni liberi e quindi non riescono a far scorrere gli elettroni quasi in nessun modo. Per questo motivo, questi materiali sono chiamati isolanti. Anche la carta o il legno sono considerati ottimi isolanti elettrici.

Fra la categoria di materiali isolanti e quella di conduttori esiste anche una categoria di materiali intermedi che sono considerati scarsi conduttori o scarsi isolanti, come, per esempio il nichelcromo, la costantana e la grafite. Tutti questi materiali che offrono una scarsa conduttività e per questo oppongono una certa "resistenza" al flusso di

elettroni, vengono utilizzati in elettronica per la costruzione di resistori, potenziometri e trimmer. Si tratta di componenti elettronici passivi che possono introdurre nel circuito una resistenza fissa al passaggio di elettroni (nel caso del resistore fisso) oppure regolarne il flusso in modo variabile in base a condizioni esterne (potenziometri, sensori ottici, sensori termici e così via).

L'unità di misura della resistenza elettrica si chiama *ohm* (simbolo Ω) indicata con la lettera greca omega maiuscola, in onore di Georg Simon Ohm (Erlangen, 1789 - Monaco di Baviera, 1854), un fisico tedesco che enunciò l'omonima legge.

Per comodità, in ambito elettronico vengono usati spesso i multipli seguenti:

KΩ	Kilo ohm (10^3)	1 000 ohm
MΩ	Mega ohm (10^6)	1 000 000 ohm

Legge di Ohm

La legge di Ohm esprime il rapporto lineare tra la tensione e la corrente circolanti in un circuito. In pratica, è il rapporto tra la differenza di potenziale *V* (tensione in volt), posta ai capi di un conduttore elettrico, e la corrente elettrica *I* (corrente in ampere) che lo attraversa:

R (valore in ohm) = V (tensione in volt) / I (intensità di corrente in ampere)

Dalla suddetta formula è facile ricavare che la resistenza di 1 ohm è pari alla corrente di 1 ampere con una caduta di tensione di 1 volt ai capi del conduttore.

Per i più curiosi, 1 ohm corrisponde alla resistenza che gli elettroni incontrano passando attraverso una colonna di mercurio lunga 1 063 millimetri, del peso di 14,4521 grammi, a una temperatura di 0 gradi.

Nella Figura 1.5 è illustrato il tipico triangolo per ricordare facilmente la legge di Ohm. Basta mettere la V della parola *VIR* (in latino, uomo) al

vertice del triangolo. Da qui è facile stabilire le relazioni fra volt, intensità di corrente e resistenza.

1. Nascondendo la V, rimane il prodotto I × R.
2. Nascondendo la I, rimane il rapporto V / R.
3. Nascondendo la R, rimane il rapporto V / I.

I componenti elettronici passivi che oppongono resistenza al passaggio di elettroni si chiamano *resistori* o *resistenze*.

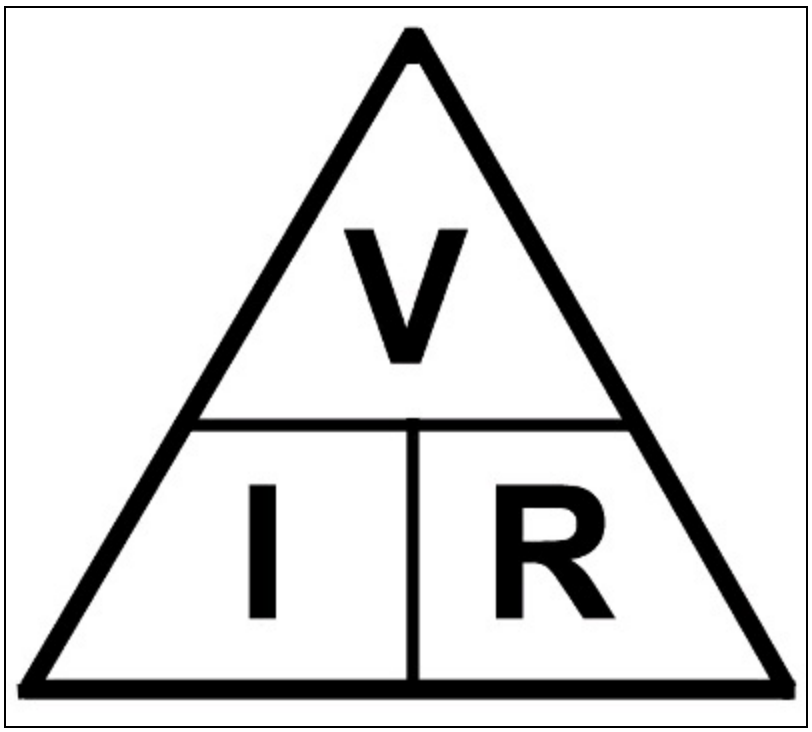


Figura 1.5 Triangolo VIR per ricordare la legge di Ohm.

Capacità elettrica

La capacità elettrica (o più raramente, capacitanza) è la predisposizione di un corpo ad aumentare il proprio potenziale elettrico quando viene sottoposto a una tensione elettrica. In altre parole, un corpo in grado di accumulare una carica elettrica viene chiamato *condensatore*. Fisicamente, un condensatore è composto da due elementi

metallici separati da un materiale isolante come plastica e derivati (poliestere, polistirene, polipropilene, Mylar), carta, ceramica, ossido di tantalio, mica o anche semplicemente aria.

Il comportamento di un condensatore può venire paragonato anche al già citato effetto triboelettrico, ovvero lo strofinamento di un oggetto isolante su un corpo conduttore (per esempio, una bacchetta di plastica su un indumento).

Collegando i terminali di un condensatore a una pila che fornisce una tensione continua, gli elettroni si dispongono sul polo negativo, tentando di raggiungere il polo positivo. Essendo il polo positivo isolato, gli elettroni non lo potranno mai raggiungere e rimarranno condensati sul terminale negativo (Figura 1.6).

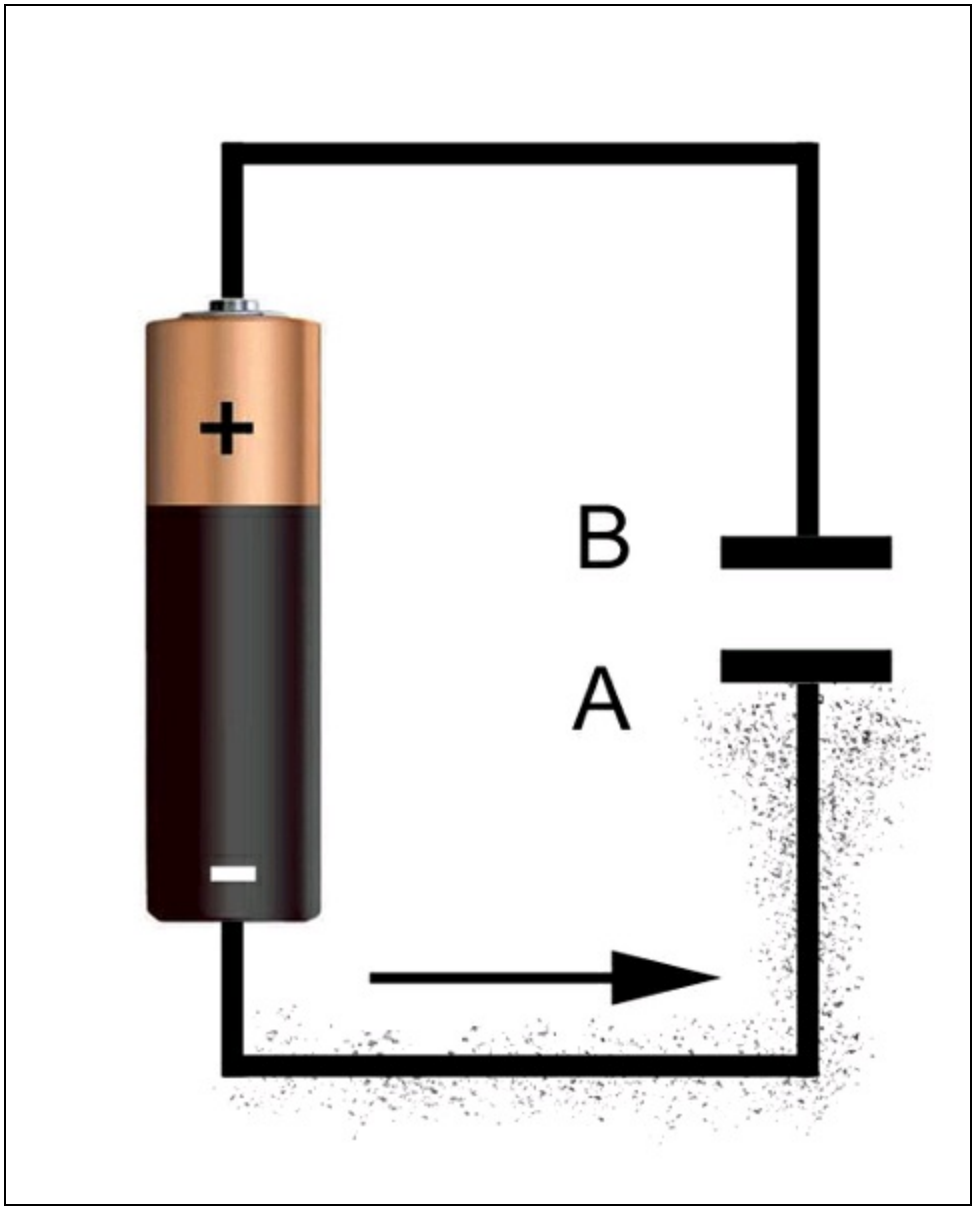


Figura 1.6 Applicando sui terminali di un condensatore una tensione continua, gli elettroni si accumulano sul terminale A, ma non potranno raggiungere il terminale B perché risulta isolato.

Togliendo l'alimentazione, i terminali del condensatore rimangono carichi, cioè su un terminale ci sarà un eccesso di elettroni, i quali rimarranno "condensati" sul terminale fino a quando il terminale non verrà messo in contatto con il terminale positivo (cortocircuito).

Se invece viene applicata una tensione alternata, si otterrà un passaggio “alternato” di elettroni fra i due terminali del condensatore, con la differenza che gli elettroni incontreranno una certa resistenza proporzionale alla capacità stessa del condensatore e alla frequenza della tensione alternata applicata sui terminali (Figura 1.7).

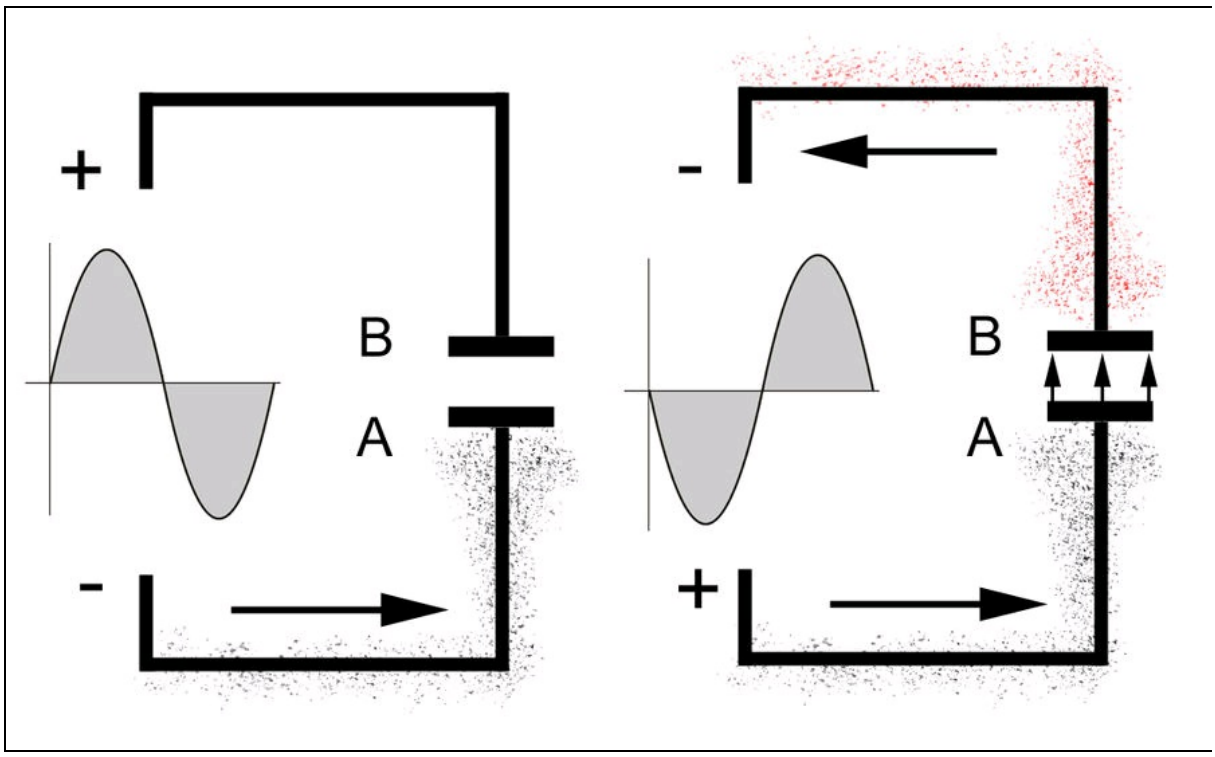


Figura 1.7 Quando la tensione alternata inverte la sua polarità, gli elettroni accumulati sul terminale A si riversano sul terminale positivo e il terminale opposto B si carica di elettroni. Il ciclo si ripete quando la corrente inverte nuovamente la polarità.

La capacità elettrica di un condensatore si misura in *farad* (simbolo F), in onore di Michael Faraday (Newington Butts, 1791 - Hampton Court, 1867), un chimico e fisico britannico che fece numerose scoperte nel campo dell’elettricità, fra cui la capacità.

Il farad può essere espresso come il rapporto di quantità elettrica e tensione:

$$F = Q / V$$

dove F = farad, Q è la quantità di carica e V è la tensione.

Essendo 1 farad un valore teorico elevato, di solito vengono usati i suoi sottomultipli:

μF	microfarad (10^{-6})	milionesimo di farad
nF	nanofarad (10^{-9})	miliardesimo di farad
pF	picofarad (10^{-12})	millesimo di miliardesimo di farad

I componenti passivi in grado di condensare cariche elettriche vengono chiamati condensatori o anche capacitori.

Induttanza elettrica

Viene chiamata *induttanza* la proprietà dei dispostivi elettrici di indurre una forza elettromagnetica o elettromotrice. Tali componenti si chiamano induttori.

In altre parole, quando una corrente elettrica scorre in un circuito elettrico produce un campo magnetico nello spazio circostante (Figura 1.8).

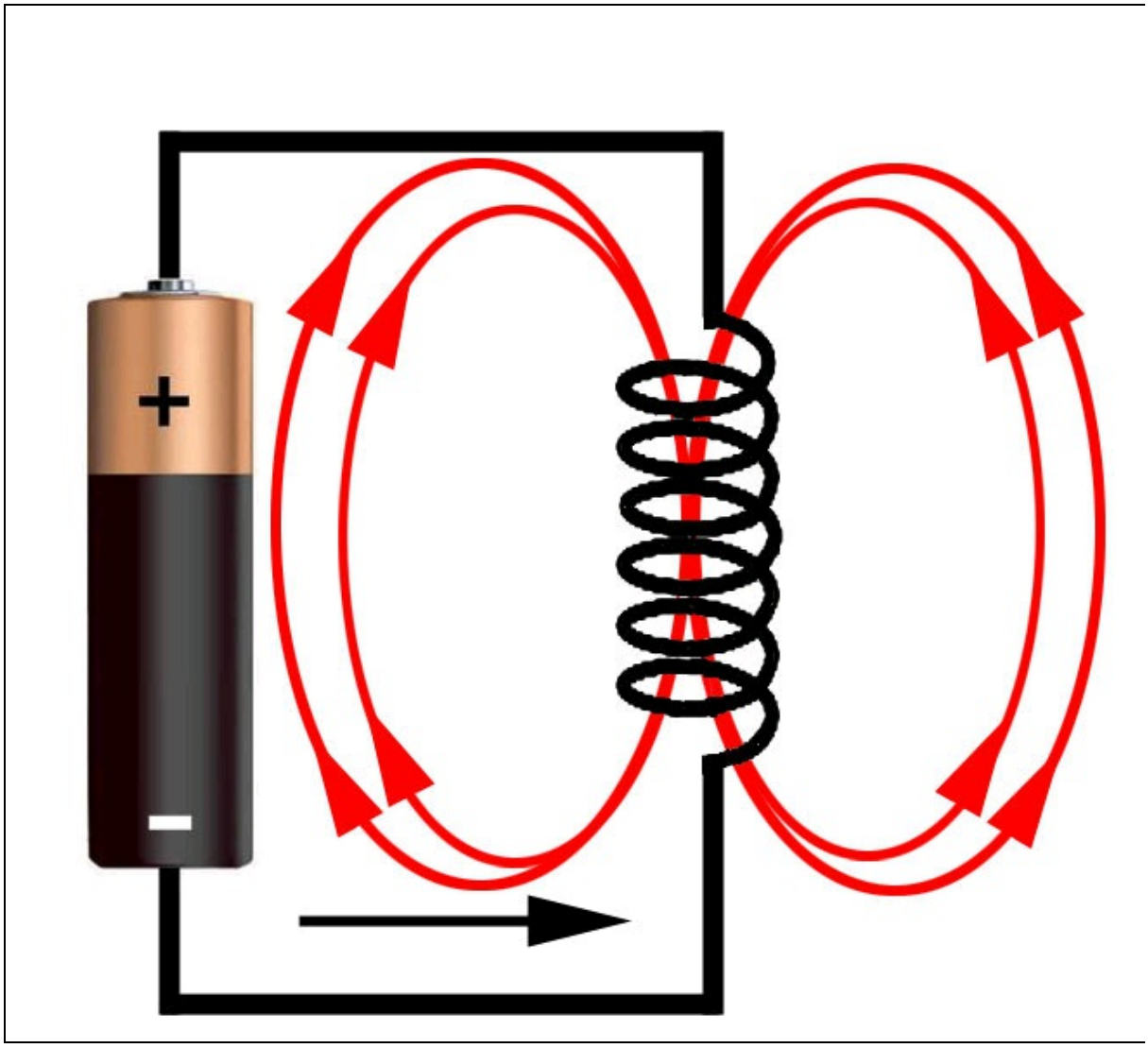


Figura 1.8 Il campo magnetico prodotto da un induttore percorso da una corrente elettrica.

L’unità di misura dell’induttanza è detta *henry* (simbolo H), in onore di Joseph Henry, (Albany, 1797 - Washington, 1878), un fisico statunitense che studiò le correnti indotte realizzando elettromagneti, in seguito impiegati nel primo telegrafo a filo.

Per definizione, in un induttore di 1 henry, una corrente di 1 ampere al secondo genera una f.e.m. (forza elettromotrice) di 1 volt, oppure la resistenza di 1 ohm al secondo:

$$1\ H = 1\ Wb / 1\ A$$

dove H = henry, Wb = weber, A = ampere.

Da cui, effettuando le sostituzioni, si ottiene l’uguaglianza:

$$1\ H = 1\ V \times s / 1\ A = 1\ ohm \times s$$

dove H = henry, V = volt, A = ampere, s = secondo.

NOTA

Il weber (simbolo Wb) è l’unità di misura del flusso magnetico, in onore del fisico tedesco Wilhelm Eduard Weber (Wittenberg, 24 ottobre 1804 - Gottinga, 23 giugno 1891). Un weber è pari al flusso magnetico attraverso una spira che produce una forza elettromotrice di 1 volt per secondo su 1 ampere di corrente.

L’induzione elettromagnetica viene sfruttata in vari dispositivi e componenti passivi come trasformatori, bobine, relè, antenne, motori elettrici, elettrocalamite e così via.

Componenti passivi

In tutti i circuiti elettronici sono sempre presenti i cosiddetti *componenti passivi*, ovvero quei componenti che possono creare una resistenza al passaggio di elettroni (resistori) o accumulare cariche di elettroni (condensatori) o diffondere elettroni nell'aria e nei corpi circostanti (induttori). Il tutto accade in modo passivo, senza alcun rilascio di energia. Al contrario, assorbono sempre un po' di energia per svolgere il proprio compito.

L'introduzione di componenti passivi in un qualsiasi circuito elettronico provoca una perdita di energia, che in parte viene trasformata in calore e in parte assorbita dal circuito stesso. Per questo motivo è necessario calcolare adeguatamente i parametri per il loro utilizzo ottimale e scegliere i componenti più idonei al circuito.

I componenti passivi più comunemente usati in elettronica sono i seguenti:

- resistore;
- condensatore;
- induttore;
- trasformatore;
- relè.

Resistore

Il resistore, chiamato in modo improprio anche "resistenza", è sicuramente il componente elettronico più usato in assoluto e, soprattutto in fase di progettazione, è di fondamentale importanza conoscerne il valore e le caratteristiche. Innanzitutto, distinguiamo quattro tipi fondamentali di resistori:

- resistori fissi;
- resistori variabili;
- fotoresistori;
- termistori.

Nella Figura 1.9 sono illustrati i simboli per i vari tipi di resistori usati comunemente negli schemi elettrici.

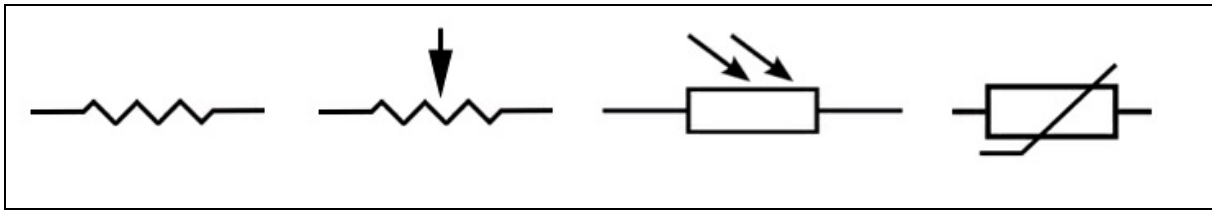


Figura 1.9 I simboli comunemente usati per il resistore fisso, il resistore variabile, il fotoresistore e il termistore.

Resistore fisso

Il resistore fisso più comunemente usato è di forma cilindrica con due terminali ai lati chiamati reofori. Il materiale di cui è composto il corpo è solitamente un impasto di ossidi metallici o di carbone.

Il valore resistivo in Ω è riportato sul corpo tramite una serie di anelli colorati (Figura 1.10a) che vanno letti in base a un codice colori (vedi più avanti). Normalmente, nei circuiti elettronici di bassa potenza, ovvero a bassa tensione e con esigue correnti circolanti, si impiegano resistori da 1/8 di watt, 1/4 di watt o 1/2 watt.

Nei circuiti elettronici di potenza, in cui circolano forti correnti, si impiegano resistori a filo di nichelcromo con potenza dissipabile superiore ai 2 watt. In questi resistori il valore resistivo non viene indicato tramite anelli colorati, ma scritto per esteso (Figura 1.10b).

Per circuiti miniaturizzati si usano resistori SMD (*Surface Mounting Device*), ovvero resistori grandi quanto la capocchia di uno spillo o anche meno, con montaggio e saldatura sulla superficie del circuito

stampato (Figura 1.10c) tramite macchinari automatici appositi. Il valore resistivo viene stampigliato sul corpo del resistore in base a una codifica personalizzata a seconda del costruttore.

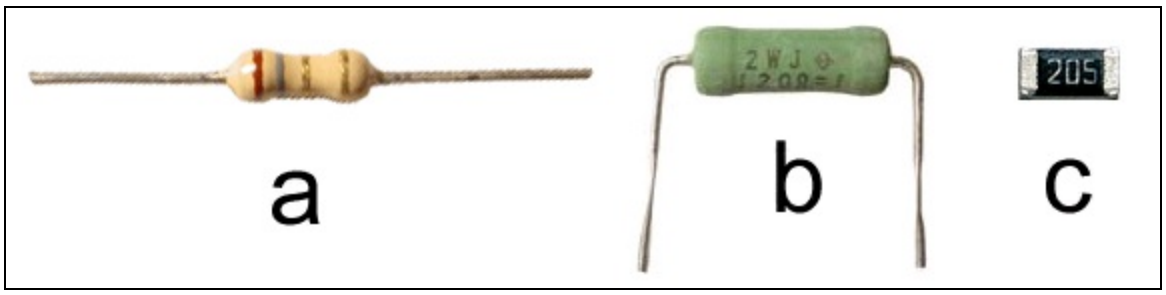


Figura 1.10 A sinistra, un comune resistore fisso con il valore in Ω riportato secondo un codice colori (a). Al centro, un resistore di potenza con il valore stampigliato per esteso (b). A destra, un resistore SMD ingrandito (c).

NOTA

Nei progetti di questo libro vengono usati resistori standard da 1/4 di watt con quattro anelli per la lettura del codice colori.

I resistori più comunemente usati riportano un codice colori a quattro anelli per identificare il valore in Ω resistivo. Il codice colori è stato pensato in modo estremamente semplice per essere facilmente ricordato. I resistori di precisione riportano cinque o anche sei anelli (non verranno trattati in questa sede).

Per una corretta lettura del codice colori, la posizione del primo anello è sempre opposta all'anello della tolleranza, che di solito è di color argento o oro.

La logica del codice colori degli anelli impone sempre che il primo e il secondo anello riportino il valore numerico e che il terzo anello sia il moltiplicatore. Il quarto anello (se presente) riporta sempre la tolleranza. Per tolleranza si intende la percentuale di fluttuazione del valore resistivo nominale. La Tabella 1.1 e la Figura 1.11 mostrano la logica del codice colori delle resistenze a quattro anelli.

Tabella 1.1 Logica del codice colori delle resistenze a quattro anelli.

Colore	1° anello	2° anello	3° anello	4° anello

	1ª cifra	2ª cifra	Moltiplicatore	Tolleranza
Argento	-	-	× 0,01 (10^{-2})	± 10%
Oro	-	-	× 0,1 (10^{-1})	± 5%
Nero	0	0	× 1 (10^{0})	-
Marrone	1	1	× 10 (10^{1})	± 1%
Rosso	2	2	× 100 (10^{2})	± 2%
Arancio	3	3	× 1 000 (10^{3})	-
Giallo	4	4	× 10 000 (10^{4})	-
Verde	5	5	× 100 000 (10^{5})	± 0,5%
Blu	6	6	× 1 000 000 (10^{6})	± 0,25%
Viola	7	7	× 10 000 000 (10^{7})	± 0,1%
Grigio	8	8	× 100 000 000 (10^{8})	± 0,05%
Bianco	9	9	× 1 000 000 000 (10^{9})	-

Esempi

Codice colori di un resistore da 470 Ω con tolleranza ± 5% (Figura 1.11a):

- 1º anello = giallo (1ª cifra) = 4.
- 2º anello = viola (2 ª cifra) = 7.
- 3º anello = marrone (moltiplicatore × 10) = 470.
- 4º anello = oro (tolleranza) = ± 5%.

Codice colori di un resistore da 22 000 Ω con tolleranza ± 10% (Figura 1.11b):

- 1º anello = rosso (1 ª cifra) = 2.
- 2º anello = rosso (2 ª cifra) = 2.
- 3º anello = arancio (moltiplicatore × 1 000) = 22 000.
- 4º anello = argento (tolleranza) = ± 10%.

Codice colori di un resistore da 820 KΩ con tolleranza ± 5% (Figura 1.11c):

- 1º anello = grigio (1 ª cifra) = 8.

- 2º anello = rosso (2 ª cifra) = 2.
- 3º anello = giallo (moltiplicatore × 10 000) = 820 000.
- 4º anello = oro (tolleranza) = ± 5%.

NOTA

Se non è presente il quarto anello, la tolleranza è del 20% rispetto al valore nominale e quindi il resistore è di scarsa qualità. Se il quarto anello è di color argento, la tolleranza è del 10% e il resistore è di media qualità. Se il quarto anello è di color oro, la tolleranza è del 5% e il resistore è di buona qualità (il caso più comune). Nei resistori di altissima precisione la tolleranza può arrivare a 0,05%.

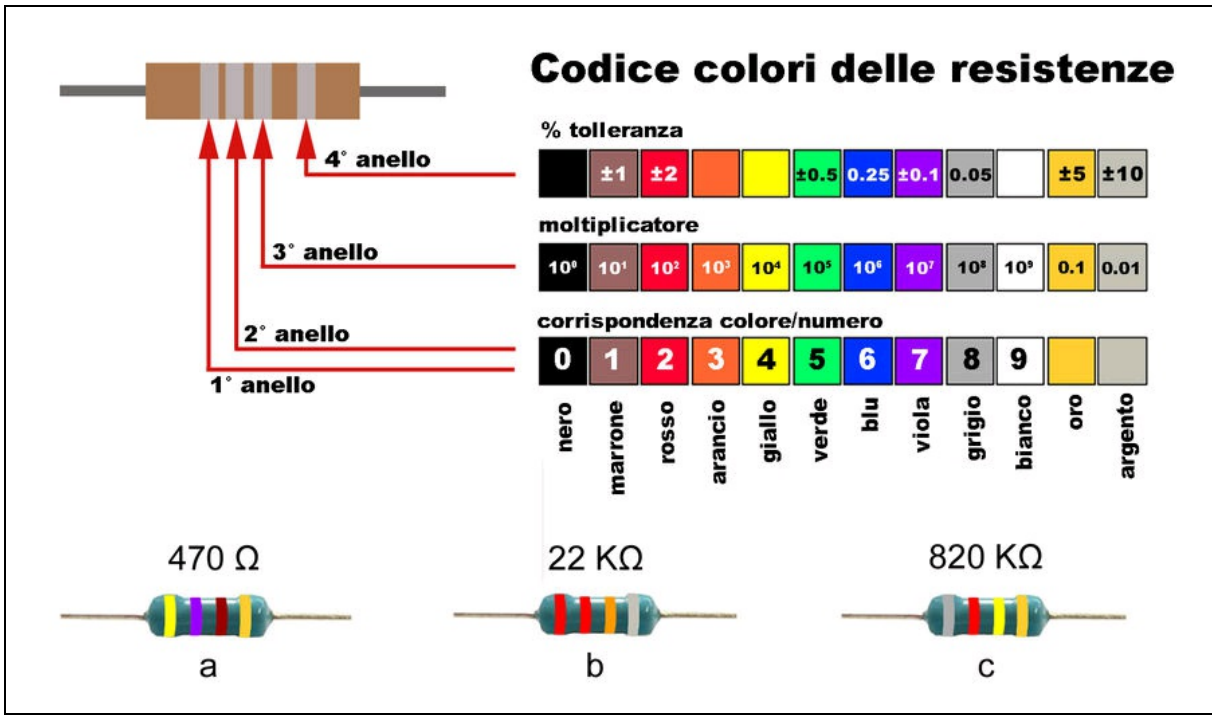


Figura 1.11 Il codice colori di tre resistori fissi.

Resistori commerciali

Un fattore importante da tenere in considerazione è che normalmente i resistori hanno valori fissi, standard, ovvero non esistono resistori di qualsiasi valore.

I valori resistivi standard che si trovano normalmente in commercio e soprattutto nei negozi per hobbisti sono i seguenti:

- 10
- 12
- 15
- 18
- 22
- 27
- 33
- 39
- 47
- 56
- 68
- 82

Ovviamente, esistono i relativi valori multipli, cioè 100, 120, 150, ..., 1 000, 1 200, 1 500, ..., 10 000, 12 000, 15 000 e così via. Per cui, se fosse necessario utilizzare un resistore con un valore fuori standard, di solito se ne cerca uno con un valore vicino (se il circuito non è critico) oppure, se il valore resistivo nel circuito è critico, si possono usare due o più resistori collegati in serie o in parallelo, a seconda delle necessità.

Si fa notare che il resistore non un è componente polarizzato e può essere montato in un circuito senza preoccuparsi del verso di inserimento.

Collegamento di resistori in serie

Con il collegamento di due o più resistori in serie si ottiene un resistore equivalente alla somma dei valori resistivi dei singoli resistori.

Per esempio, per ottenere il valore fuori standard di 386 Ω, basta collegare in serie un resistore standard da 330 Ω e uno da 56 Ω (Figura

1.12a). Per ottenere un valore fuori standard di 1 400 Ω, a seconda di cosa si ha disposizione in laboratorio, si possono mettere in serie tre resistori da 470 Ω (totale 1 410 Ω) oppure uno da 1 KΩ, uno da 220 Ω e uno da 180 Ω (totale 1 400 Ω). Spesso, sfruttando la tolleranza dei resistori si arriva a ottenere un valore pressoché perfetto.

Collegamento di resistori in parallelo

Se si collegano due o più resistori in parallelo, il calcolo per il valore risultante è leggermente più complicato (Figura 1.12b).

Per due soli resistori si applica la seguente formula:

$$R = (R1 \times R2) / (R1 + R2)$$

Nel caso si colleghino un resistore da 220 Ω (R1) e un resistore da 330 Ω (R2):

$$R = (220 \times 330) / (220 + 330) = 72\,600 / 550 = 132\ \Omega$$

Se i resistori sono di valore uguale, con il collegamento in parallelo si ottiene sempre un resistore pari alla metà del valore di ciascun resistore. Supponendo di collegare due resistori da 220 Ω in parallelo, si otterrà un resistore da 110 Ω:

$$R = (220 \times 220) / (220 + 220) = 48\,400 / 440 = 110\ \Omega$$

Per il collegamento in parallelo di *n* resistori, invece, la formula da applicare è più complessa (Figura 1.12c):

$$1/R = 1/R1 + 1/R2 + 1/R3... + 1/Rn$$

Supponendo di voler collegare in parallelo un resistore da 220 Ω (R1), un resistore da 100 Ω (R2) e un resistore da 330 Ω (R3), si otterrà un resistore equivalente con il valore seguente:

$$1/R = 1/220 + 1/100 + 1/330 = 56{,}9\ \Omega\ (circa)$$

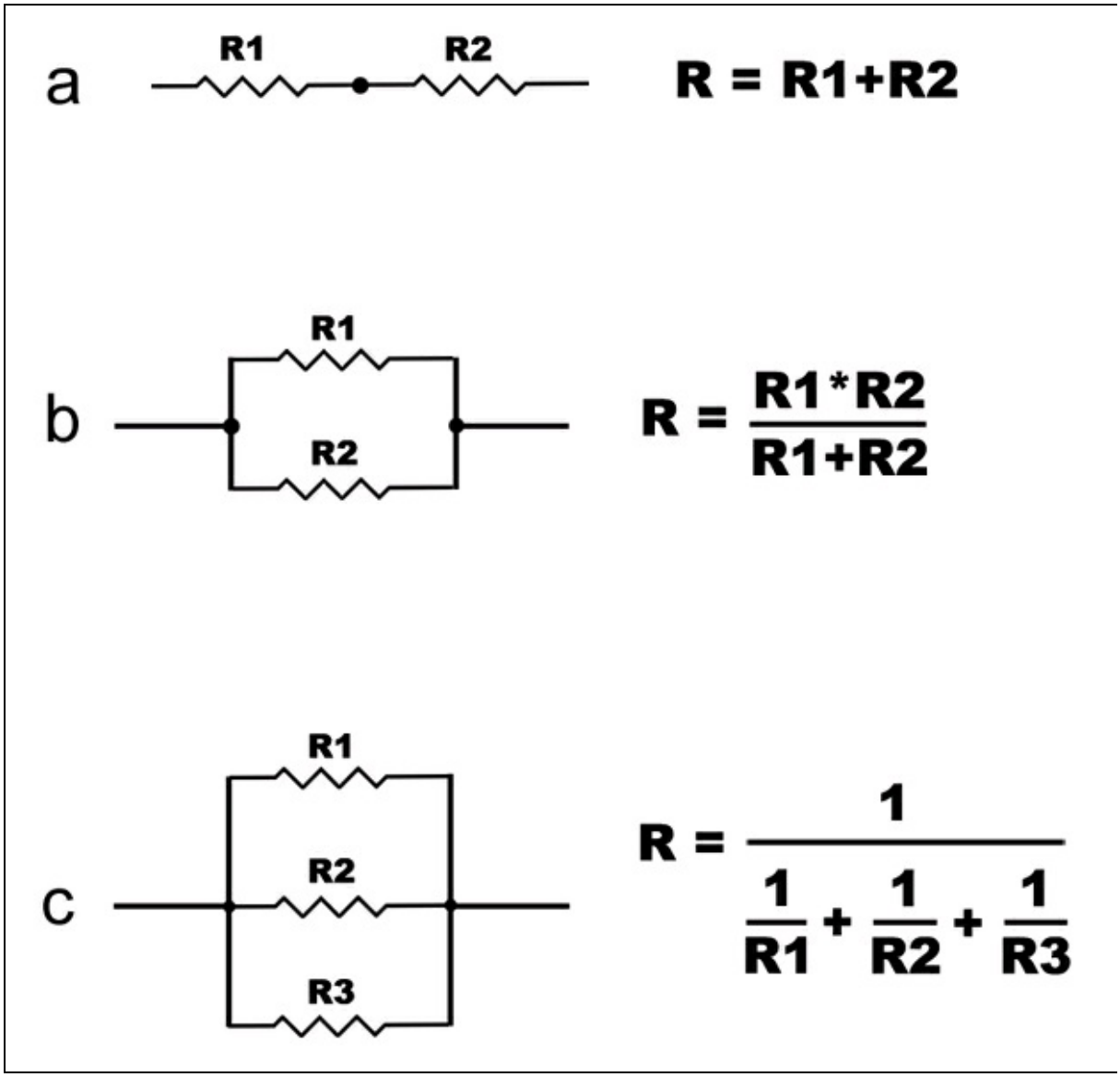


Figura 1.12 Collegamento di resistori in serie e in parallelo.

Esperimento con un resistore

Facendo riferimento alla legge di Ohm, si possono ricavare tutti i valori di corrente, resistenza e tensione, a seconda delle necessità circuitali.

Per esempio, se un resistore oppone una resistenza di 10 Ω in un circuito alimentato da una tensione di 5 Volt, la corrente circolante sarà:

I = V / R

I = 5 V / 10 Ω = 0,5 A

Se nello stesso circuito inseriamo un resistore di 100 Ω avremo:

I = V / R

I = 5 V / 100 Ω = 0,05 A

Applicando la stessa formula, è possibile calcolare il valore della resistenza da utilizzare sapendo di dover limitare la corrente a un certo valore in un circuito di cui si conosce la tensione applicata.

Per esempio, supponiamo di dover alimentare un diodo LED utilizzando una pila da 9 V (Figura 1.13). Sapendo che il diodo LED sopporta una tensione massima di circa 2,5 volt e una corrente di circa 30 milliampere (0,03 A), dovremo utilizzare un resistore adeguato per non bruciare il diodo, applicando la seguente formula:

R (resistenza) = V (tensione) / I (corrente)

R = (9 - 2,5) / 0,03 = 6,5 / 0,03 = 216,6 Ω (valore standard da usare 220 Ω)

Dando un senso più chiaro alla formula, si deve sottrarre la tensione massima da fornire al diodo (2,5 volt) alla tensione di alimentazione (pila da 9 volt). Si divide il risultato ottenuto (6,5 volt) per l'intensità di corrente sopportata (0,03 ampere) e si ottiene il valore in ohm del resistore da utilizzare (6,5 / 0,03 = 216,6 Ω). Volendo alimentare il diodo con una tensione di 24 volt, applicando la stessa formula si ottiene:

R = (24 - 2,5) / 0,03 = 21,5 / 0,03 = 716,6 Ω (valore standard da usare 680 Ω)

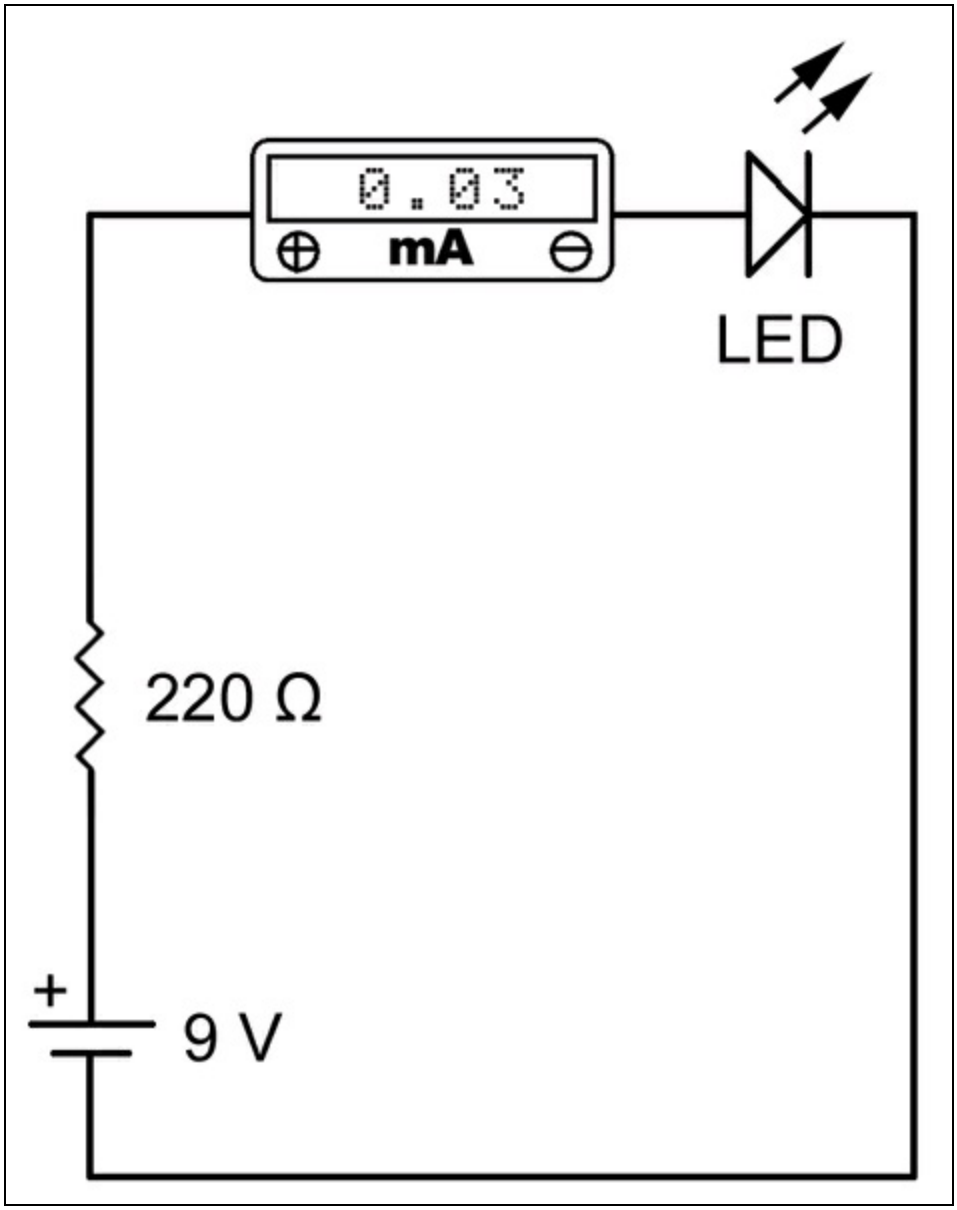


Figura 1.13 Un resistore da 220 Ω serve a limitare la corrente a 0,03 mA che alimenta il LED in un circuito alimentato a 9 volt.

Resistore variabile

I resistori variabili possono essere paragonati a partitori di tensione, ovvero a due resistori posti in serie con presa di uscita centrale (Figura 1.14a).

I resistori variabili si distinguono principalmente in:

- potenziometri;
- trimmer.

La differenza sostanziale fra potenziometri e trimmer sta nella loro forma e nel loro utilizzo nei circuiti (Figura 1.14b).

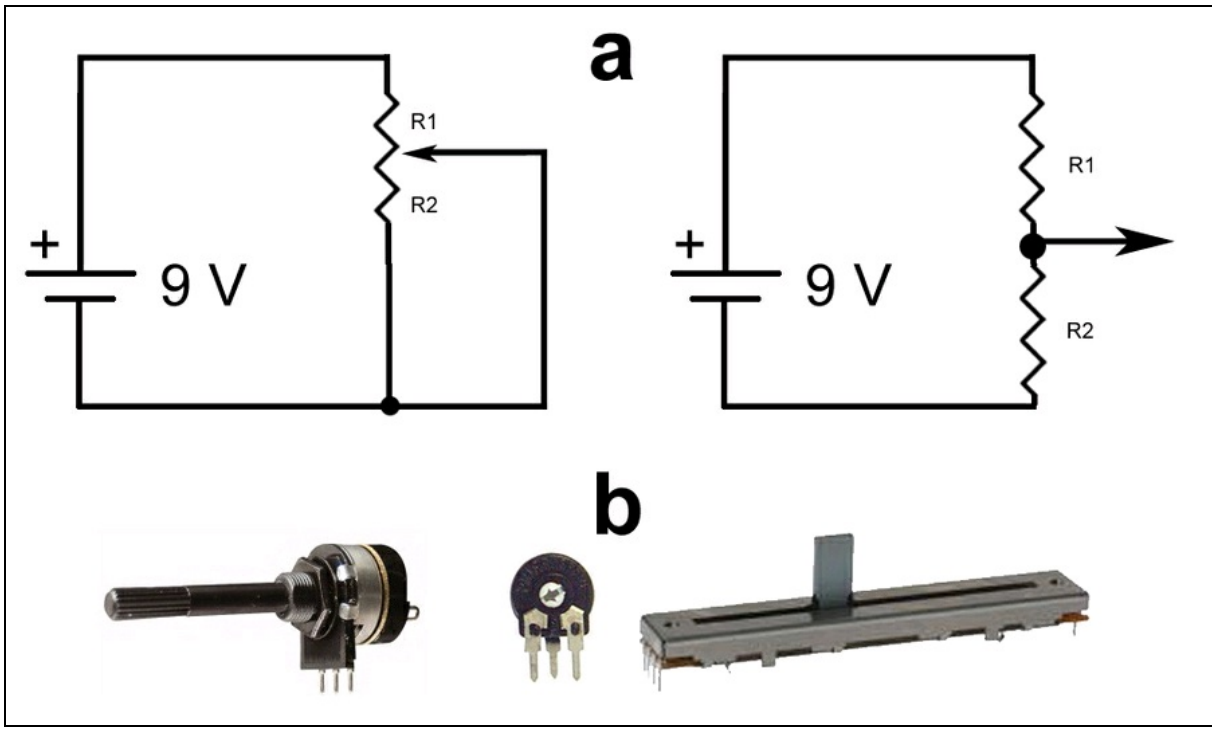


Figura 1.14 Partitore di tensione equivalente a un resistore variabile (a). Da sinistra, un potenziometro, un trimmer e un potenziometro a slitta (b).

Il potenziometro può essere di tipo rotativo, con un alberino di rotazione collegato a un contatto mobile su un materiale resistivo (grafite), oppure di forma allungata con una slitta scorrevole (in inglese, *slider*).

Un potenziometro può servire a variare in qualsiasi momento il livello di uscita di un circuito: per esempio, serve a variare il volume nei circuiti di amplificazione, il livello dei toni alti e bassi in un circuito di equalizzazione o la luminosità di LED e display o a controllare la velocità di un motore elettrico.

È abbastanza comune l’uso di potenziometri rotativi dotati anche di interruttore per lo spegnimento dell’alimentazione. Nel caso di controllo simultaneo di più circuiti identici, esistono potenziometri multipli con alberino coassiale collegato a più elementi resistivi, come nel caso di amplificatori o equalizzatori stereo.

Il trimmer è dotato di una vite di regolazione che fa le veci dell’alberino di rotazione. Il trimmer viene usato solitamente per tarare circuiti e, una volta trovato il valore resistivo adeguato, la vite di regolazione viene bloccata con una goccia di colla. Esistono anche trimmer con alberino asportabile.

Una particolare proprietà da tenere in considerazione in fase di progettazione, sia dei potenziometri che dei trimmer, è il tipo di variazione resistiva, che può essere di due tipi:

- lineare;
- logaritmica.

Il potenziometro o trimmer può essere di tipo lineare o logaritmico per poter variare i valori resistivi “laterali” in modo lineare o logaritmico rispetto alla presa centrale. La variazione logaritmica viene impiegata principalmente per regolazioni del volume audio negli amplificatori, per compensare l’andamento logaritmico tipico dell’ascolto umano.

In circuiti di precisione esistono anche potenziometri rotativi di tipo multigiro, in cui l’alberino può compiere fino a 20 giri prima di raggiungere il fine corsa.

Fotoresistore

Un’importante categoria di resistori usati in elettronica è quella dei cosiddetti *fotoresistori* o *fotocellule*, il cui valore resistivo cambia in funzione dell’intensità della luce (Figura 1.15). Vengono talvolta indicati con la sigla LDR (*Light-Dependent Resistor*).

In pratica, quando non vengono colpiti dalla luce, i fotoresistori presentano valori resistivi elevati (circa 2 MΩ), mentre, se colpiti con una luce abbastanza forte la loro resistenza può scendere a valori bassissimi, anche inferiori ai 100 Ω.

I fotoresistori sono costituiti da materiali semiconduttori drogati come solfuro di cadmio (CdS), solfuro di piombo (PbS), selenio (Se) e altri composti fotoconduttivi. A causa dell'effetto inerziale della reazione elettrochimica, la curva luce/resistenza, pur essendo abbastanza lineare, potrebbe risultare un po' troppo lenta. Per questo motivo, l'uso dei fotoresistori è limitato alle applicazioni in cui non serve un controllo immediato del dispositivo collegato, come interruttori crepuscolari, giocattoli, chiusura delle porte degli ascensori, passaggio di persone e così via.

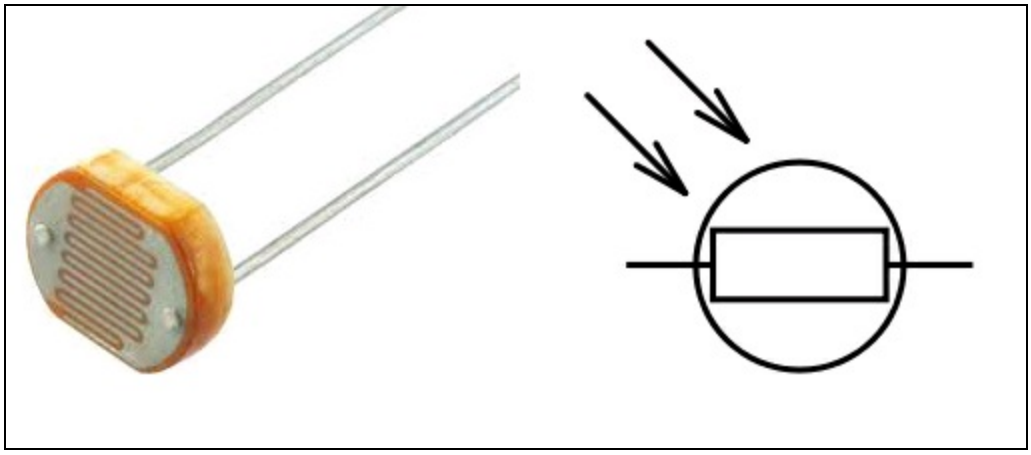


Figura 1.15 Un tipo di fotoresistore e il suo simbolo elettrico.

Termistore

In elettronica sono chiamati *termistori* o anche *termoresistori* quei componenti passivi il cui valore resistivo cambia in funzione del calore (Figura 1.16).

Esistono due tipi di termistori.

- NTC (*Negative Temperature Coefficient*), componente a coefficiente negativo di temperatura. I termistori NTC diminuiscono la resistenza interna con l'aumento della temperatura.
- PTC (*Positive Temperature Coefficient*), componente a coefficiente positivo di temperatura. I termistori PTC aumentano la resistenza interna con l'aumento della temperatura.

Per la natura stessa della loro composizione non sono trasduttori veloci e la variazione della resistenza interna non è repentina, né la loro curva di risposta alla temperatura è lineare. Possono venire impiegati per il controllo automatico della temperatura, come termostati e applicazioni simili. Sia i modelli PTC che NTC possono essere sensibili a temperature che vanno dai -100 °C fino a 300 °C.

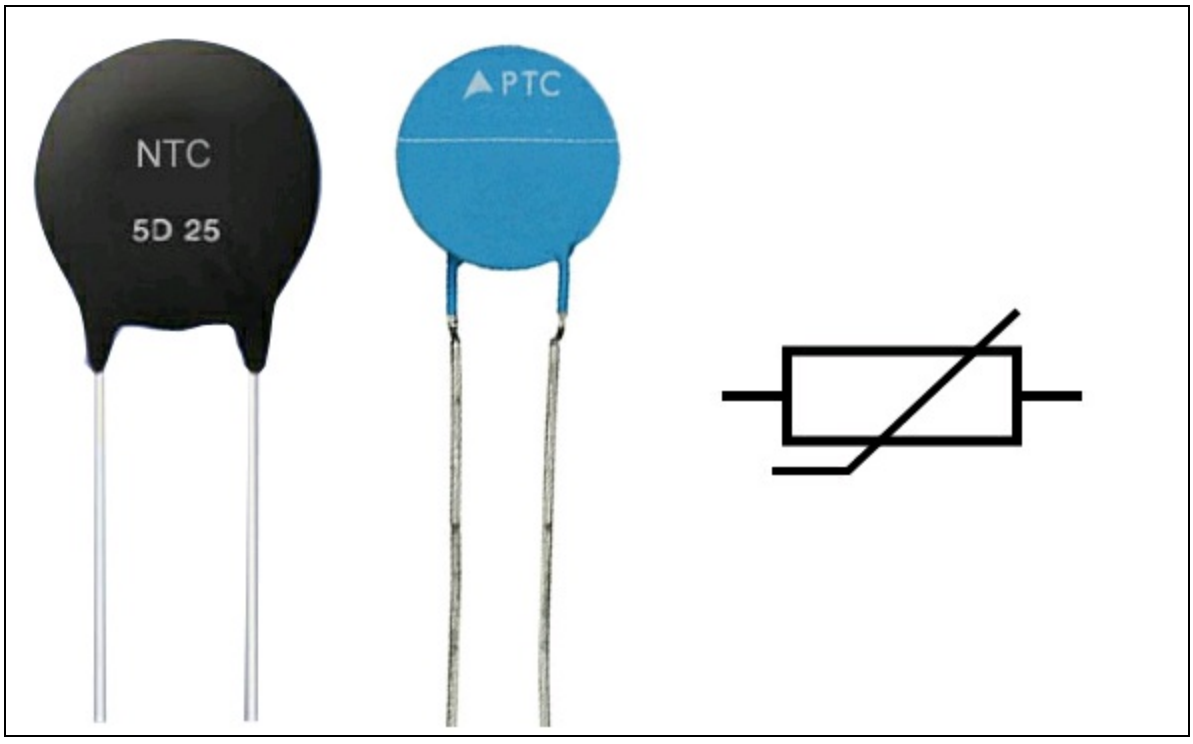


Figura 1.16 Da sinistra, un termistore NTC, un termistore PTC e il simbolo elettrico.

Condensatore

Il condensatore (raramente detto “capacitore”) è il componente passivo usato nei circuiti per mantenere la carica elettrica e rilasciarla in maniera controllata, spesso in abbinamento a resistori, transistor o circuiti integrati.

Nei comuni circuiti elettronici, il condensatore lascia passare le correnti variabili (per esempio, segnali audio) e blocca le correnti continue. Per esempio, con i condensatori si possono facilmente filtrare frequenze o far lavorare i filtri a determinate frequenze o livellare tensioni a valori definiti nei circuiti di alimentazione e così via.

Distinguiamo tre tipi fondamentali di condensatori:

- condensatori normali;
- condensatori polarizzati (elettrolitici e al tantalio);
- condensatori variabili.

Nella Figura 1.17 sono illustrati vari tipi di condensatori normali, elettrolitici e variabili e i simboli più comuni per i vari tipi di condensatori usati negli schemi elettrici.

Il condensatore normale è costituito da due “armature” separate da un materiale isolante, chiamato *dielettrico*, che può essere di ceramica, plastica, carta, aria e altro ancora.

Si fa notare che il condensatore normale non un è componente polarizzato e può essere montato in un circuito senza preoccuparsi del verso di inserimento.

Il condensatore elettrolitico ha l’isolamento costituito da uno strato di ossido metallico sulla superficie di un’armatura immersa in una soluzione chimica (soluzione elettrolitica).

I condensatori elettrolitici hanno solitamente capacità elevate e sono componenti polarizzati, cioè devono essere inseriti nel circuito rispettando la polarità positiva e negativa della tensione applicata. In caso di montaggio errato l’isolamento può cedere e causare l’esplosione dell’elettrolita. Di solito vengono utilizzati in circuiti alimentati in

corrente continua e il dielettrico è costituito da uno strato di ossido di alluminio.

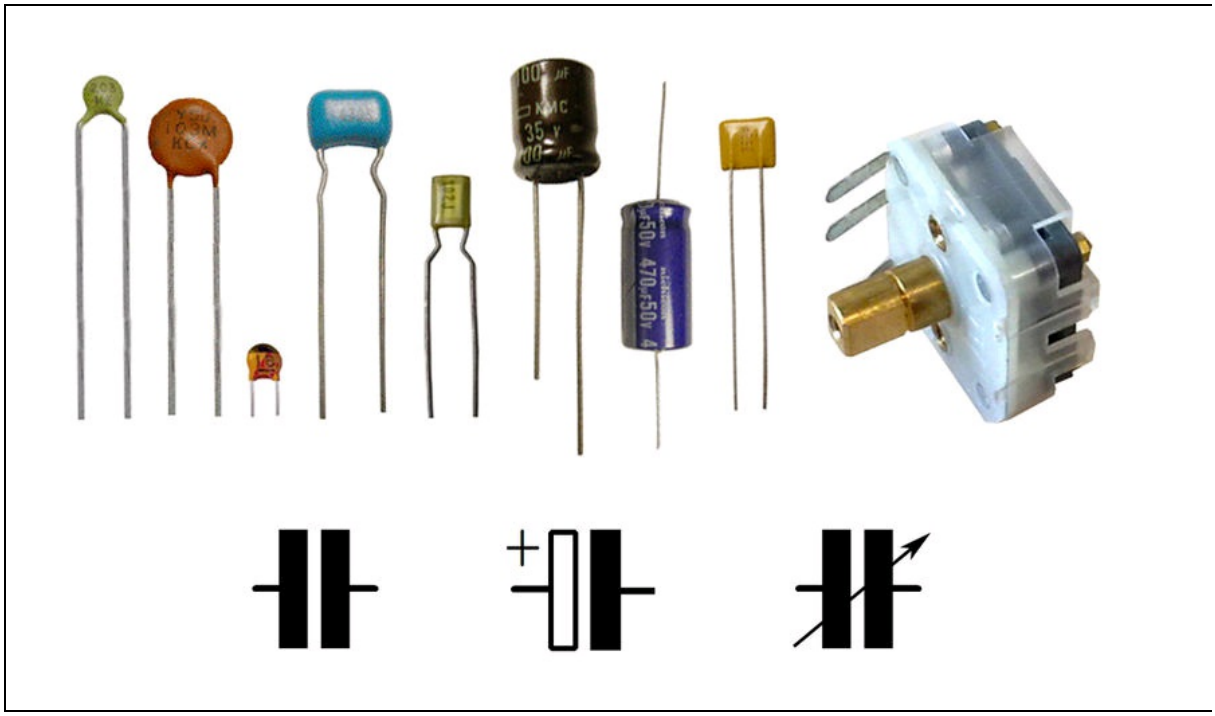


Figura 1.17 Vari tipi di condensatori normali, elettrolitici e, a destra, uno variabile ad aria (a). I simboli comunemente usati per il condensatore normale, il condensatore elettrolitico o al tantalio e il condensatore variabile (b).

I condensatori elettrolitici al tantalio, rispetto ai condensatori a ossido di alluminio, hanno una capacità più accurata e bassa impedenza alle basse frequenze. Possono esplodere violentemente se vengono alimentati con polarità invertita.

Per circuiti miniaturizzati si usano condensatori elettrolitici SMD (*Surface Mounting Device*).

I condensatori variabili hanno la capacità che può essere variata manualmente, per esempio, per sintonizzare le stazioni radiofoniche o per tarare circuiti RF. In pratica, si tratta di un condensatore che varia la frequenza di risonanza di un circuito RLC (resistore, induttore, condensatore). I normali condensatori variabili di sintonia utilizzano l’aria come dielettrico. I condensatori variabili montati direttamente sul

circuito stampato e dotati di vite di regolazione sono chiamati *compensatori*. Sono simili ai trimmer potenziometrici e servono principalmente per la calibrazione di circuiti di risonanza o di filtro. Al termine della calibrazione vengono fissati con una goccia di colla o di silicone.

Come per i resistori, sono disponibili solo condensatori con valori capacitivi standard. I valori più comuni sono i seguenti (con i relativi multipli e sottomultipli):

- 1
- 10
- 22
- 33
- 47
- 56
- 68
- 82

Per l’identificazione del valore capacitivo esistono vari codici numerici stampigliati sul corpo dei condensatori.

Contrariamente ai resistori, a volte può essere disagevole leggere i valori sul corpo dei condensatori a causa delle loro ridotte dimensioni e perché i costruttori adottano sistemi di codifica diversi (codice americano, codice europeo, codice asiatico).

Ecco alcuni esempi.

Sigla stampigliata sul corpo	Valore in pF
22	22 pF
270, 271, n27	270 pF
47n, 473	47 000 pF
100n,104	100 000 pF
392, 3n9, .0039, n0039	3 900 pF
683, 68n, .068, n068	68 000 pF

474, 470n, .47, u47	470 000 pF

La lettera M, K o J, se presente, indica la tolleranza:

- M inferiore al 20%;
- K inferiore al 10%;
- J inferiore al 5%.

Accanto alla lettera, ma non sempre, viene indicata anche la tensione di lavoro (VL), che invece risulta di vitale importanza per evitare di “bruciare” il condensatore.

Le tensioni di lavoro dei condensatori comunemente usati nei circuiti elettronici di bassa potenza vanno da 12 VL a 100 VL.

Solitamente le tensioni di lavoro sono proporzionali alla grandezza fisica del condensatore. Basta avere il buon senso di non usare condensatori fisicamente piccoli in circuiti con tensioni elevate e tenere a mente la regola “condensatore piccolo = bassa tensione di lavoro” e viceversa.

Nei condensatori elettrolitici a ossido di alluminio, la lettura del valore capacitivo e della tensione di lavoro è più facile perché solitamente vengono riportati per esteso in modo molto chiaro sia il valore della capacità in μF sia la tensione di lavoro. Viene riportato anche il simbolo - (meno) in corrispondenza del reoforo negativo che va collegato alla massa del circuito, ovvero al polo negativo di alimentazione. Nei condensatori elettrolitici con montaggio verticale, il reoforo positivo è più lungo e in quelli a montaggio orizzontale il reoforo positivo viene indicato con una strozzatura sul corpo del condensatore.

Alcuni condensatori elettrolitici al tantalio riportano il valore capacitivo seguendo il codice colori dei resistori a quattro anelli e l’eventuale presenza di una quinta banda colorata indica la tensione di lavoro.

Collegamento di condensatori in parallelo

Per ottenere un valore capacitivo fuori standard è possibile collegare due o più condensatori in parallelo. Con il collegamento in parallelo si ottiene un condensatore equivalente alla somma dei valori capacitivi dei singoli condensatori.

Per esempio, collegando in parallelo un condensatore di 220 pF e uno di 470 pF si ottiene un condensatore equivalente di 690 pF (Figura 1.18a).

Collegamento di condensatori in serie

Per ottenere un valore capacitivo fuori standard è possibile collegare due o più condensatori in serie (Figura 1.18b). Il calcolo di due condensatori in serie è simile a quello di due resistori in parallelo:

$$C = (C1 \times C2) / (C1 + C2)$$

In pratica, nel caso si colleghino in serie un condensatore da 220 pF (C1) e un condensatore da 330 pF (C2):

$$C = (220 \times 330) / (220 + 330) = 132 \text{ pF}$$

Nel caso si colleghino in serie due condensatori identici, il valore capacitivo risultante è la metà del valore dei condensatori. Supponendo di collegare due condensatori da 220 pF in serie, si otterrà un condensatore di 110 pF:

$$C = (220 \times 220) / (220 + 220) = 48\,400 / 440 = 110 \text{ pF}$$

Per il collegamento in serie di *n* condensatori, invece, la formula da applicare è più complessa (Figura 1.18c):

$$1/C = 1/C1 + 1/C2 + 1/C3... + 1/Cn$$

Supponendo di voler collegare in serie un condensatore da 220 pF (C1), un condensatore da 100 pF (C2) e un condensatore da 330 pF (C3), si otterrà un condensatore equivalente con il valore seguente:

$$1/C = 1/220 + 1/100 + 1/330 = 56{,}9 \text{ pF (circa)}$$

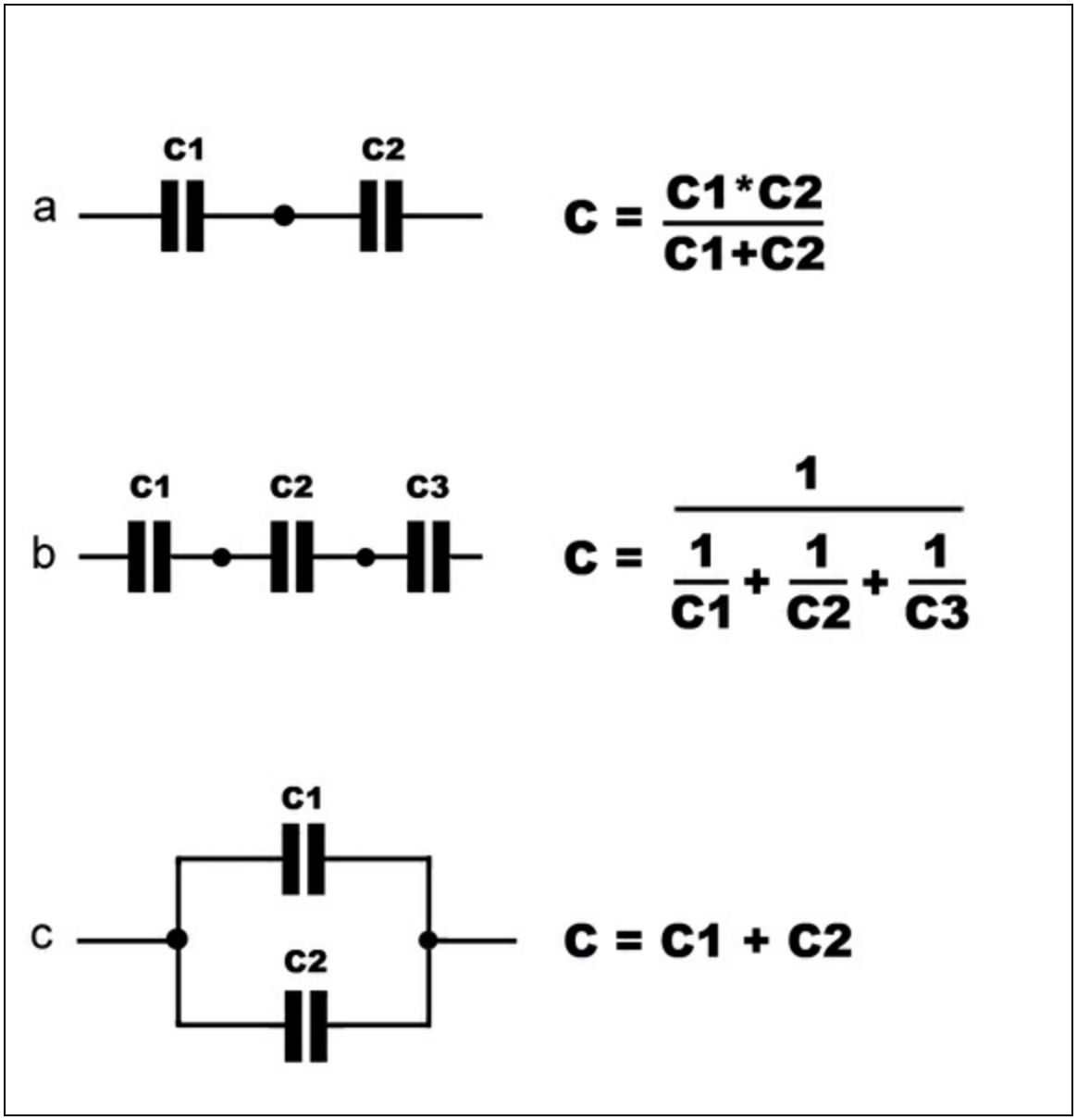


Figura 1.18 Collegamento di condensatori in parallelo e in serie.

Esperimento con un condensatore

Per capire meglio il funzionamento dei condensatori ecco un piccolo esperimento, illustrato schematicamente nella Figura 1.19.

1. Prendere un condensatore elettrolitico di qualsiasi capacità, ma meglio se di 100 µF o 1 000 µF o anche più grosso, con tensione di

lavoro 12 o 25 V.

2. Collegare per qualche secondo a un pila da 9 V i conduttori del condensatore, rispettando la polarità, cioè collegando il reoforo più lungo (+) al polo positivo della pila e quello più corto o quello contrassegnato con il segno (-) al polo negativo della pila.
3. Staccare la pila dal condensatore e collegare i due conduttori del condensatore elettrolitico a un diodo LED, rispettando la polarità, cioè collegando il reoforo lungo al reoforo lungo del LED.
4. Si vedrà lampeggiare per un istante il diodo LED.
5. Mettendo in serie, su uno dei due conduttori del condensatore, un resistore di circa 1 000 Ω, si può vedere il LED spegnersi lentamente.

Questo esperimento dimostra come il condensatore agisca da accumulatore di carica elettrica, in modo simile a una pila ricaricabile. Quando il condensatore viene collegato a un circuito chiuso, ovvero a un carico (nel nostro caso il diodo LED), il condensatore si scarica, rilasciando l'energia accumulata. Mettendo in serie un resistore, la scarica del condensatore elettrolitico avviene più lentamente, attraverso la resistenza che limita il passaggio di corrente. Più è alto il valore del resistore, più la scarica avviene lentamente.

In questo esperimento si possono usare anche condensatori normali (non polarizzati), magari più di uno collegati in parallelo per ottenere, se possibile, qualche decina di µF, visto che difficilmente si trovano valori elevati nei condensatori normali. La misurazione del rilascio della carica deve però essere eseguita tramite un voltmetro impostato su mV (millivolt) e non un diodo LED, essendo la tensione rilasciata insufficiente per la sua accensione (ovvero, è al di sotto del valore di soglia del diodo).

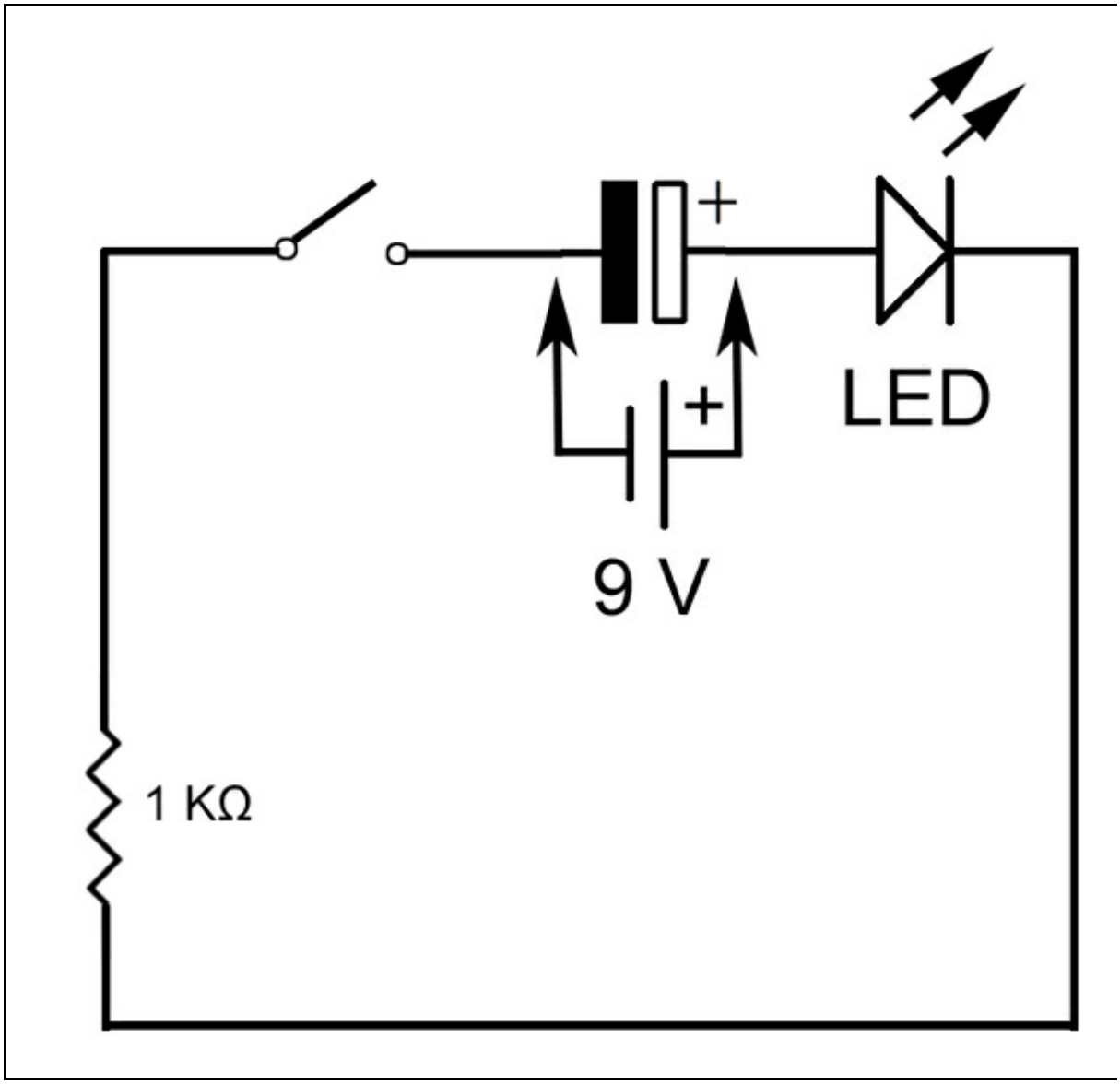


Figura 1.19 Carica e scarica del condensatore. Collegare la pila da 9 V ai capi del condensatore e poi, dopo aver staccato la pila, chiudere il circuito tramite l'interruttore.

Induttore

L'induttore è un componente passivo composto, di solito, da un avvolgimento di filo di rame smaltato, con spire avvolte in aria o attorno a un supporto di ferrite, che serve a generare un campo magnetico sfruttando il passaggio della corrente elettrica.

Il principio di induzione elettromagnetica viene impiegato nei dispositivi elettrici ed elettronici, come trasformatori, motori elettrici e applicazioni in radiofrequenza (antenne).

Nella Figura 1.20 sono illustrati sono illustrati vari tipi di induttori e i simboli più usati negli schemi elettrici.

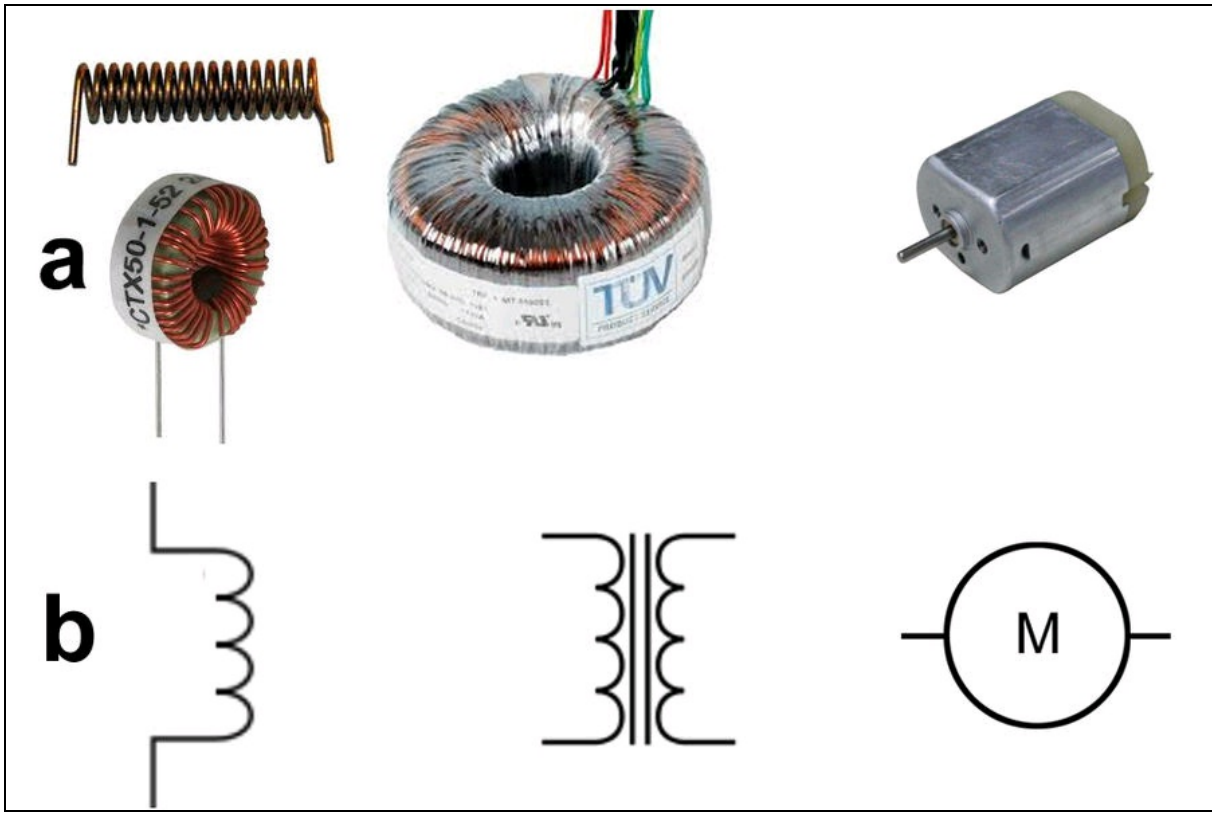


Figura 1.20 Vari tipi di induttori: bobina in aria, trasformatore toroidale, motore elettrico (a). Simboli comunemente usati per gli induttori. Da sinistra: bobina, trasformatore, motore elettrico (b).

Un tipico induttore è costituito da un avvolgimento di filo di rame smaltato (chiamato solenoide) e può avere due capi solamente o avere diversi punti di presa intermedi.

L’unità di misura degli induttori è l’henry (simbolo H), ma è molto più facile trovare induttori con valore sottomultiplo in µH (microhenry), ovvero milionesimi di henry.

Per aumentare l’induttanza del solenoide si avvolge il filo di rame smaltato su un nucleo di ferrite a elevata permeabilità magnetica.

A volte anche un semplice filo con poche spire o una pista di un circuito stampato possono venire usati come induttori.

Esperimento con un induttore

Per capire meglio il principio di induzione elettromagnetica ecco un piccolo esperimento, illustrato schematicamente nella Figura 1.21.

1. Prendere uno spezzone di filo di rame nudo non smaltato.
2. Prendere un piccolo magnete (meglio se cilindrico) e attaccarlo alla parte inferiore (polo negativo) di una pila da 1,5 V.
3. Creare un arco a forma di cuore, in modo che la parte appuntita del cuore tocchi il polo positivo della pila da 1,5 V e le due estremità tocchino il magnete, da una parte e dall’altra.

Si vedrà il “cuore” ruotare attorno alla pila da 1,5 V. Questo dimostra che un solenoide percorso da corrente elettrica genera un campo magnetico, pertanto tenterà di sfuggire al campo magnetico opposto creato dal magnete. È da notare che la pila e il filo diventeranno molto caldi e la pila si scaricherà velocemente.

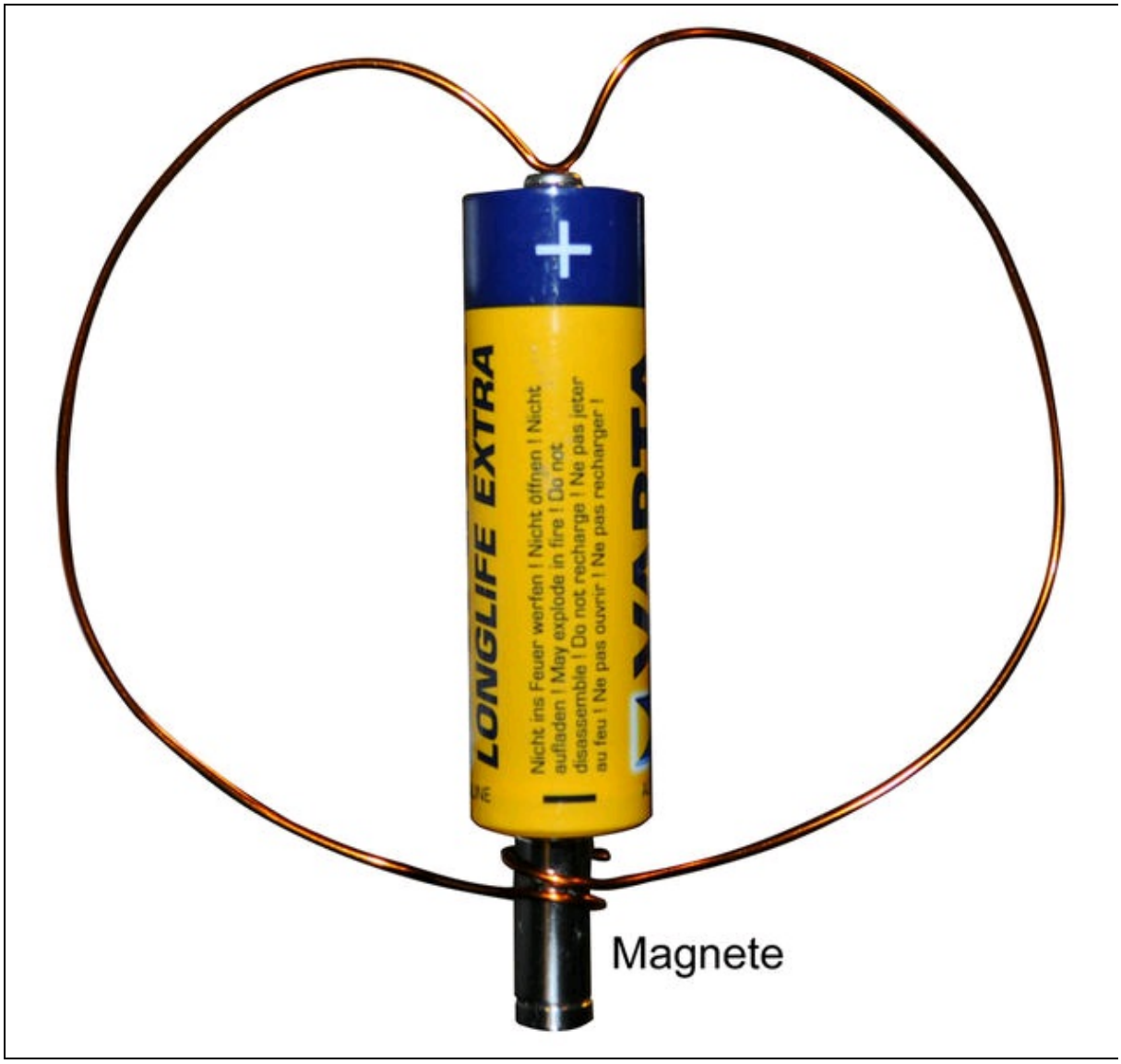


Figura 1.21 Principio di induzione elettromagnetica. La bobina di rame poggia sul polo positivo della pila e sulla calamita cilindrica posta sotto e collegata al polo negativo della pila. La bobina genera così un campo magnetico. Il campo magnetico della bobina tenterà di sfuggire al campo magnetico della calamita ruotando attorno alla pila.

Trasformatore

Il trasformatore viene definito come una “macchina elettrica statica” in grado di variare i parametri di tensione e corrente in ingresso rispetto a quelli in uscita.

Per funzionare, il trasformatore sfrutta l’induzione elettromagnetica, ma è in grado di operare solo in corrente alternata o segnali elettrici con flusso variabile.

Di solito, il trasformatore viene usato per abbassare il valore della tensione di rete da 220 V a una tensione più bassa, solitamente 12 V, 9 V o 5 V per poter alimentare circuiti elettronici e dispositivi di bassa potenza.

Il classico trasformatore è composto da due avvolgimenti di filo di rame smaltato su supporti lamellari di ferrite. Per convenzione, l’avvolgimento *primario* è quello che viene collegato alla rete domestica a 220 V e l’avvolgimento *secondario* è quello che viene collegato all’utilizzatore.

Di solito un trasformatore per usi domestici ha l’avvolgimento secondario dotato di più punti di uscita per prelevare tensioni diverse. Per esempio, è abbastanza comune trovare trasformatori con avvolgimento primario 220 V e secondario con uscita a 6 V, 9 V, 12 V e 15 V.

Nella Figura 1.22 sono illustrati i simboli di vari tipi di trasformatore e un tipico trasformatore per hobbisti.

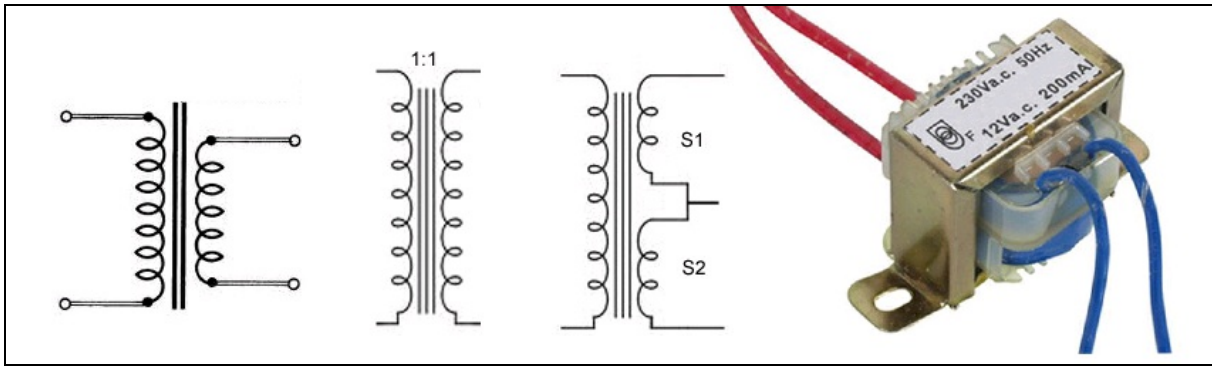


Figura 1.22 Vari simboli di trasformatori. A destra un tipico trasformatore 220-12 V, usato per circuiti elettronici di bassa potenza.

Relè

Il relè (dal francese *relais*) è un dispositivo elettromeccanico a rilascio che sfrutta l'induzione elettromagnetica come fonte per il controllo meccanico di chiusura o apertura di contatti collegati a circuiti elettrici.

In pratica, un relè è dotato di un elettromagnete al quale viene applicata una tensione continua per attirare o rilasciare una barretta metallica collegata a una serie di contatti che aprono o chiudono circuiti elettrici di diversa natura (Figura 1.23).

Tramite un semplice circuito elettronico dotato un di relè con solenoide a 5 o 12 V è possibile accendere e spegnere un intero impianto luci di un palazzo (luci scale o installazioni simili).

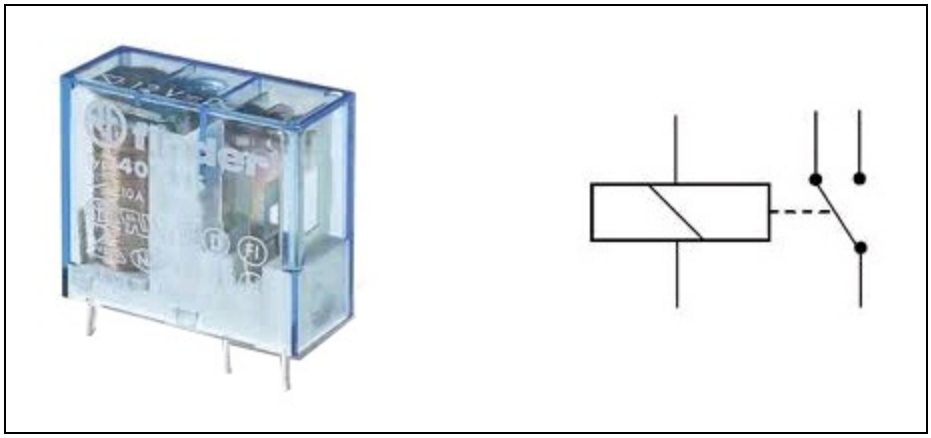


Figura 1.23 Un tipico relè Finder e relativo simbolo elettrico.

Esistono in commercio vari tipi di relè che si differenziano per il numero e il tipo di contatti, oltre che per tensione di alimentazione e carico applicabile.

Le caratteristiche elettriche più comuni sono le seguenti:

- relè a eccitazione comune, con un contatto collegato direttamente alla bobina e alimentazione diretta dalla tensione di rete (230 volt);
- relè a eccitazione separata, con alimentazione della bobina a bassa tensione (solitamente 12 o 24 volt) e contatti di potenza separati;

- isolamento tra i circuiti;
- potenza nominale che il relè è in grado di commutare.

Oltre ai relè con elettromagnete ci sono anche relè per usi specifici:

- relè *dry-reed* con contatti in una ampolla di vetro sottovuoto, usato nei sistemi antifurto;
- relè a stato solido, che non hanno contatti meccanici e sono costituiti da due circuiti elettronici separati da un fotoaccoppiatore. Vengono usati per lavorare in corrente alternata.

Componenti vari

Non potendo esaurire il vasto elenco di componenti passivi, diamo un breve sguardo a una serie di componenti utilizzati in alcuni progetti del libro.

Oscillatore al quarzo

Un oscillatore con cristallo di quarzo (abbreviato anche come XTAL) è un circuito elettronico in miniatura che sfrutta le proprietà piezoelettriche e la risonanza meccanica del cristallo per ottenere un segnale oscillante a una frequenza molto precisa. Di solito, il piccolo frammento di quarzo è inserito in un contenitore metallico sottovuoto.

Inserendo un quarzo in un circuito oscillante, la frequenza di oscillazione rimane estremamente stabile, perché entra in risonanza solo a una determinata frequenza. Normalmente si usano i quarzi per mantenere in sincrono orologi digitali o per stabilizzare il clock nei circuiti integrati e nelle CPU, come pure per stabilizzare la frequenza di trasmissione dei segnali RF.

Gli oscillatori al quarzo possono avere una frequenza di risonanza di pochi kHz fino a qualche MHz. Nei circuiti digitali, in abbinamento a CPU o circuiti integrati, la frequenza di risonanza del quarzo può servire

come clock di riferimento o per impostare la velocità di ricezione e trasmissione seriale dei dati (frequenza di clock).

Per regolare la frequenza di risonanza del cristallo, si possono mettere in serie carichi capacitivi (condensatori) o carichi induttivi (bobine). Mettendo in serie un induttore si abbassa la frequenza, mettendo in serie un condensatore la si alza.

Nella Figura 1.24 è illustrato il simbolo tipico di un oscillatore al quarzo e un comune quarzo con il valore di 16 MHz stampigliato sull'involucro.

Si fa notare che il quarzo non un è componente polarizzato e può essere montato in un circuito senza preoccuparsi del verso di inserimento.

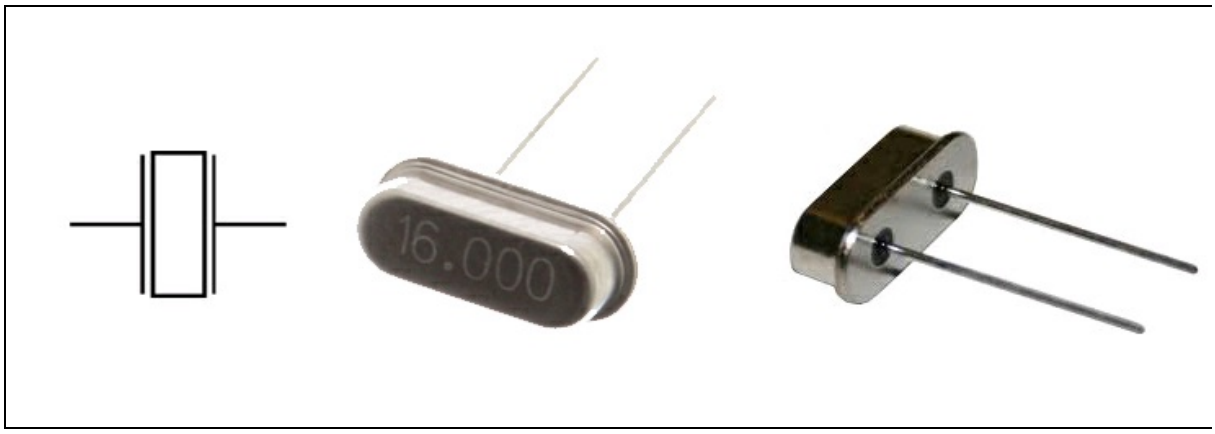


Figura 1.24 Simbolo di un oscillatore al quarzo. A destra un quarzo a 16 MHz con contenitore ribassato.

Altoparlante

L'altoparlante è un trasduttore elettroacustico in grado di trasformare l'energia elettrica in energia meccanica. Grazie a un sistema basato sull'induzione elettromagnetica, un altoparlante riesce a muovere l'aria circostante, creando una serie di pressioni e depressioni delle molecole d'aria sull'orecchio, che le percepisce come suoni.

In altre parole, applicando un segnale elettrico a frequenza variabile ai capi della bobina di un altoparlante, questo muove un cono di cartone o altro materiale facendo vibrare l'aria antistante alla stessa frequenza del segnale elettrico immesso ai capi del solenoide (Figura 1.25). È da notare che l'indicazione di polarità positiva e negativa (massa) serve solo per la fase del segnale in uscita. Il segnale applicato a un altoparlante è di tipo variabile e non deve mai essere in corrente continua.

La caratteristica di un altoparlante di riprodurre le frequenze acustiche più o meno fedelmente si chiama "risposta in frequenza". In linea teorica, un altoparlante è in grado di riprodurre le frequenze sonore della gamma udibile che va da 20 Hz a 20 kHz.

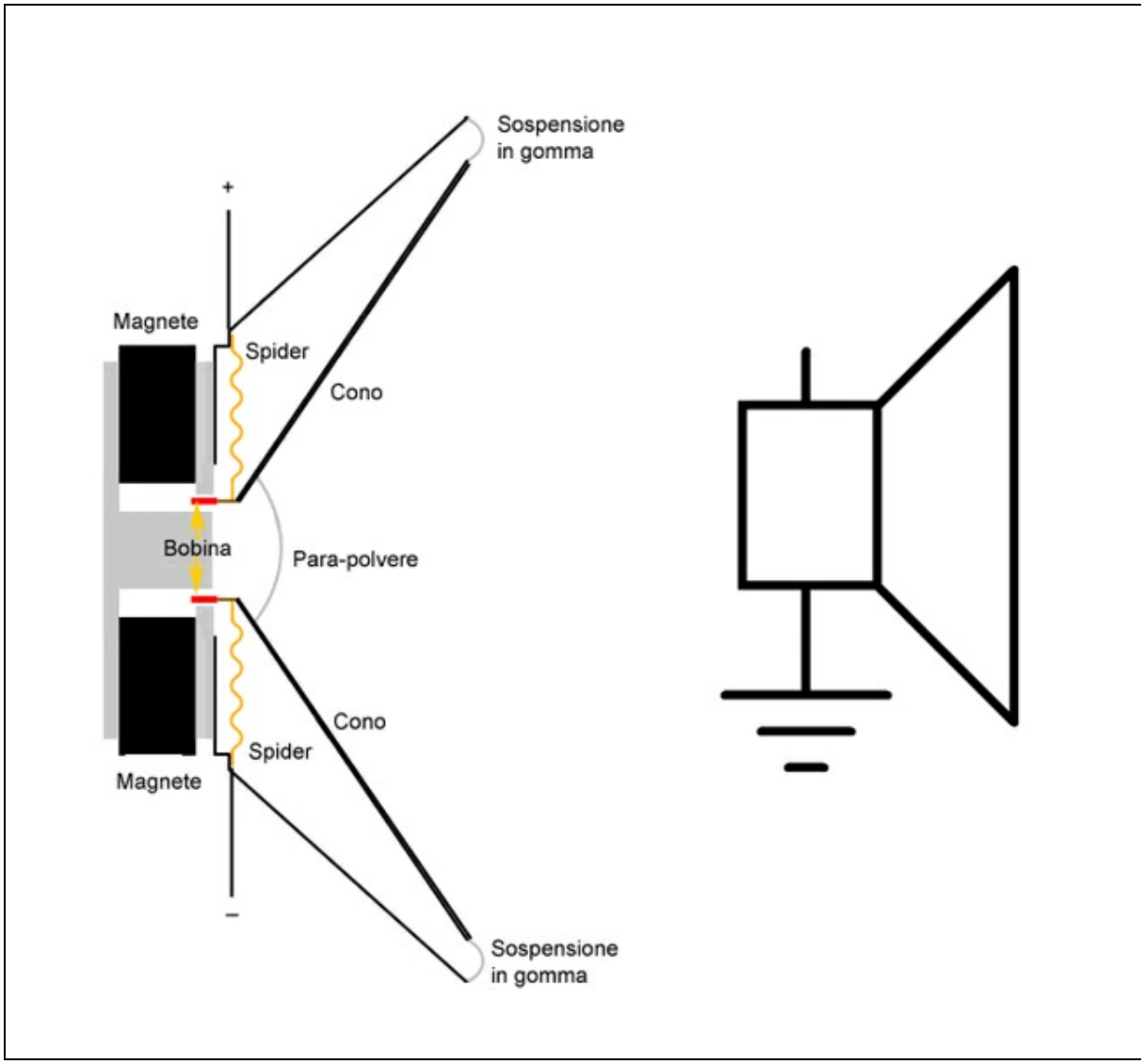


Figura 1.25 Principio di funzionamento di un altoparlante e simbolo elettrico usato negli schemi.

Oltre alla risposta in frequenza, gli altoparlanti sono caratterizzati da una serie di parametri importanti, uno fra tutti la potenza dissipata in watt. Normalmente, per poter trasformare la corrente elettrica in energia acustica, l'altoparlante, ovvero il suo sistema bobina-magnete-cono, deve avere dimensioni adeguate.

È da notare che la potenza dissipata in watt non ha nulla a che vedere con l'efficienza di un altoparlante, che invece viene misurata in dB SPL

(*decibel Sound Pressure Level*). Un altoparlante più efficiente significa che offre una pressione (dB SPL) maggiore di un altoparlante di pari potenza.

Per produrre basse frequenze serve un cono più grande e quindi un magnete più grosso per poterlo muovere, come nel caso dei woofer e sub-woofer. Per produrre frequenze acute è sufficiente un cono piccolo, per cui i tweeter possono avere dimensioni ridottissime.

Esistono vari tipi di altoparlante a seconda del tipo di utilizzo e della frequenza acustica da riprodurre.

- Super-tweeter, altoparlante specializzato nella riproduzione di frequenze altissime (oltre 18 kHz).
- Tweeter, altoparlante specializzato nella riproduzione di frequenze acute (da 15 kHz a 18 kHz).
- Mid-range, altoparlante specializzato nella riproduzione di frequenze medie (da 5 kHz a 12 kHz).
- Woofer, altoparlante specializzato nella riproduzione di frequenze basse (120 Hz a 3 kHz).
- Sub-woofer, altoparlante specializzato nella riproduzione di frequenze sub-basse (sotto i 120 Hz).
- Full-range, altoparlante a gamma intera specializzato nella riproduzione di tutte le frequenze udibili (da 20 Hz a 20 kHz, in teoria).

Gli altoparlanti adottano come unità misura l'impedenza (simbolo Z), ovvero la resistenza elettrica alla frequenza tipica di 1 kHz. Normalmente l'impedenza nominale di un altoparlante è di 4 Ω oppure di 8 Ω, ma questi valori variano in base alla frequenza del segnale.

Come per le resistenze, si possono mettere in serie e parallelo più altoparlanti per aumentare l'efficienza acustica, stando però attenti a non scendere o salire troppo con l'impedenza nominale. A parità di potenza, sotto i 2 Ω un sistema di amplificazione inizia a scaldare troppo i circuiti

di uscita dello stadio finale. Significa che l’uscita dell’amplificatore è troppo vicina al cortocircuito. Sopra i 16 Ω si inizia a perdere in efficienza. Significa che serve più potenza elettrica per ottenere lo stesso livello di pressione sonora.

È abbastanza comune utilizzare due altoparlanti da 8 Ω in parallelo per ottenere un sistema a 4 Ω, in base alla formula (R1 × R2) / (R1 + R2).

Microfono

Se l’altoparlante può essere considerato la “bocca” di un sistema audio, il microfono è l’“orecchio”. Con un comportamento fisico simile all’altoparlante, il microfono è un trasduttore elettroacustico in grado di trasformare i suoni in energia elettrica. La massa d’aria che viene spostata davanti a un microfono può muovere un sistema bobina-magnete nei cosiddetti microfoni dinamici. Esistono altri tipi di microfoni, che però non verranno trattati in questa sede:

- microfono a nastro;
- microfono a condensatore;
- microfono piezoelettrico;
- microfono a carbone.

Nel microfono dinamico, la capsula è composta da una bobina sospesa in un campo magnetico, solidale con il diaframma. Quando le onde sonore mettono in vibrazione il diaframma, la bobina immersa nel campo magnetico inizia a vibrare alla stessa frequenza dell’onda sonora e genera un analogo segnale elettrico in uscita. Per questo motivo il microfono dinamico viene talvolta chiamato “a bobina mobile” (Figura 1.26).

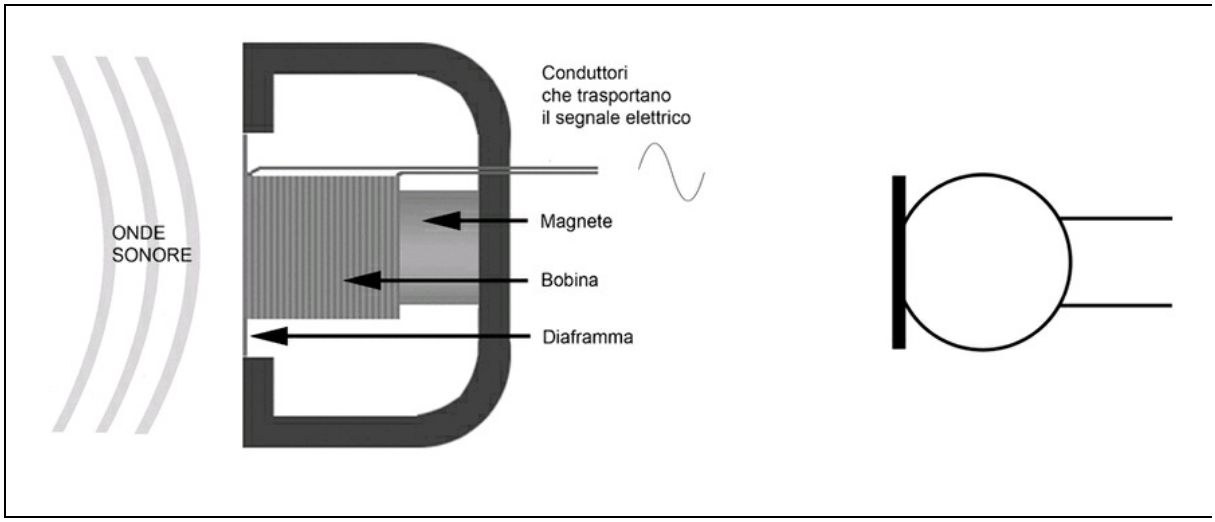


Figura 1.26 Principio di funzionamento di un microfono dinamico e simbolo elettrico usato negli schemi.

Non essendo questa la sede per un esame approfondito delle caratteristiche dei microfoni, diremo solo che un buon microfono dovrebbe avere una buona risposta in frequenza per poter riprendere i suoni fedelmente.

Ecco un elenco di sorgenti sonore e la relativa risposta in frequenza di un microfono adeguato alla loro ripresa, rispettandone i canoni di fedeltà.

- Buona parte degli strumenti musicali: da 80 Hz a 15 kHz.
- Strumenti con frequenze basse: da 40 Hz a 9 kHz.
- Ottoni: da 80 Hz a 12 kHz.
- Pianoforte: da 40 Hz a 12 kHz.
- Piatti: da 15 o 20 kHz.
- Percussioni: da 300 Hz a 15 kHz.
- Orchestra sinfonica o banda: da 40 Hz a 15 kHz.
- Voce umana: da 80 Hz a 12 kHz.

Un altro fattore da tenere in forte considerazione nella scelta di un microfono è il livello di pressione sonora (SPL). Il suono meno intenso è

a 0 dB SPL. La conversazione normale alla distanza di 30 cm è a circa 70 dB SPL. Un suono forte (vicino alla soglia del dolore) è superiore a 120 dB SPL.

Se un microfono ha un SPL massimo di 125 dB SPL, significa che inizia a distorcere in modo percepibile quando il suono prodotto raggiunge e supera i 125 dB SPL. Un microfono con SPL massimo di 120 dB è considerato buono, con SPL massimo di 135 dB è ottimo, con SPL massimo di 150 dB è eccellente.

I microfoni dinamici, generalmente non generano distorsioni del segnale elettrico neppure in presenza di suoni molto forti, a differenza di altri microfoni a condensatore o a nastro.

Anche con segnali molto forti, all'uscita del microfono la corrente del segnale elettrico è comunque irrisoria (pochi millivolt) e va pertanto amplificata per poter essere gestita correttamente da un dispositivo di acquisizione. Come per gli altoparlanti, anche per i microfoni si adotta come unità di misura l'impedenza (simbolo Z), ovvero la resistenza effettiva in uscita alla frequenza di 1 kHz. Nel collegamento a uno stadio di amplificazione va sempre tenuta in considerazione l'impedenza di un microfono per ottenere la migliore efficienza del segnale su tutta la linea.

- L'impedenza di un microfono tra 150 e 600 Ω è considerata bassa (valore standard per quasi tutti i microfoni).
- L'impedenza di un microfono tra 1 000 e 4 000 Ω è considerata media.
- L'impedenza di un microfono sopra i 10 kΩ è considerata alta (per esempio, il pick-up di una chitarra).

Per esempio, se si collega un microfono a bassa impedenza a uno stadio di amplificazione con ingresso ad alta impedenza, si ha una perdita di segnale, per cui si rende necessario un adattatore di impedenza, ovvero un trasformatore in salita con rapporto 1:15 o 1:30.

Componenti attivi

I componenti elettronici che vengono definiti “attivi” sono in grado di influenzare “attivamente” il flusso della corrente elettrica e quindi il comportamento degli altri componenti. Tutti i componenti attivi trattati in questa sede si basano sulla tecnologia a semiconduttore.

Diodo

Il diodo è un componente attivo composto da due terminali collegati a una giunzione PN (positivo/negativo) di silicio sottoposto a drogaggio, ovvero in grado di lasciar passare le cariche elettriche solo positive o solo negative, in base alla direzione della corrente. Pertanto si tratta di un componente polarizzato che va montato in un circuito rispettando le sue caratteristiche di semiconduttore. I suoi elettrodi vengono contraddistinti in un circuito elettrico come anodo e catodo.

La lettera K indica il catodo (-) e la lettera A indica l’anodo (+). Una fascetta sul corpo, una tacca o il terminale più corto indicano il reoforo corrispondente al catodo.

Nella Figura 1.27a viene mostrata la predisposizione del diodo a bloccare la corrente positiva che scorre verso il catodo (giunzione N) e di lasciarla invece passare quando proviene dall’anodo (giunzione P).

Nella Figura 1.27b sono illustrati alcuni tipi di diodo e i simboli usati negli schemi elettrici.

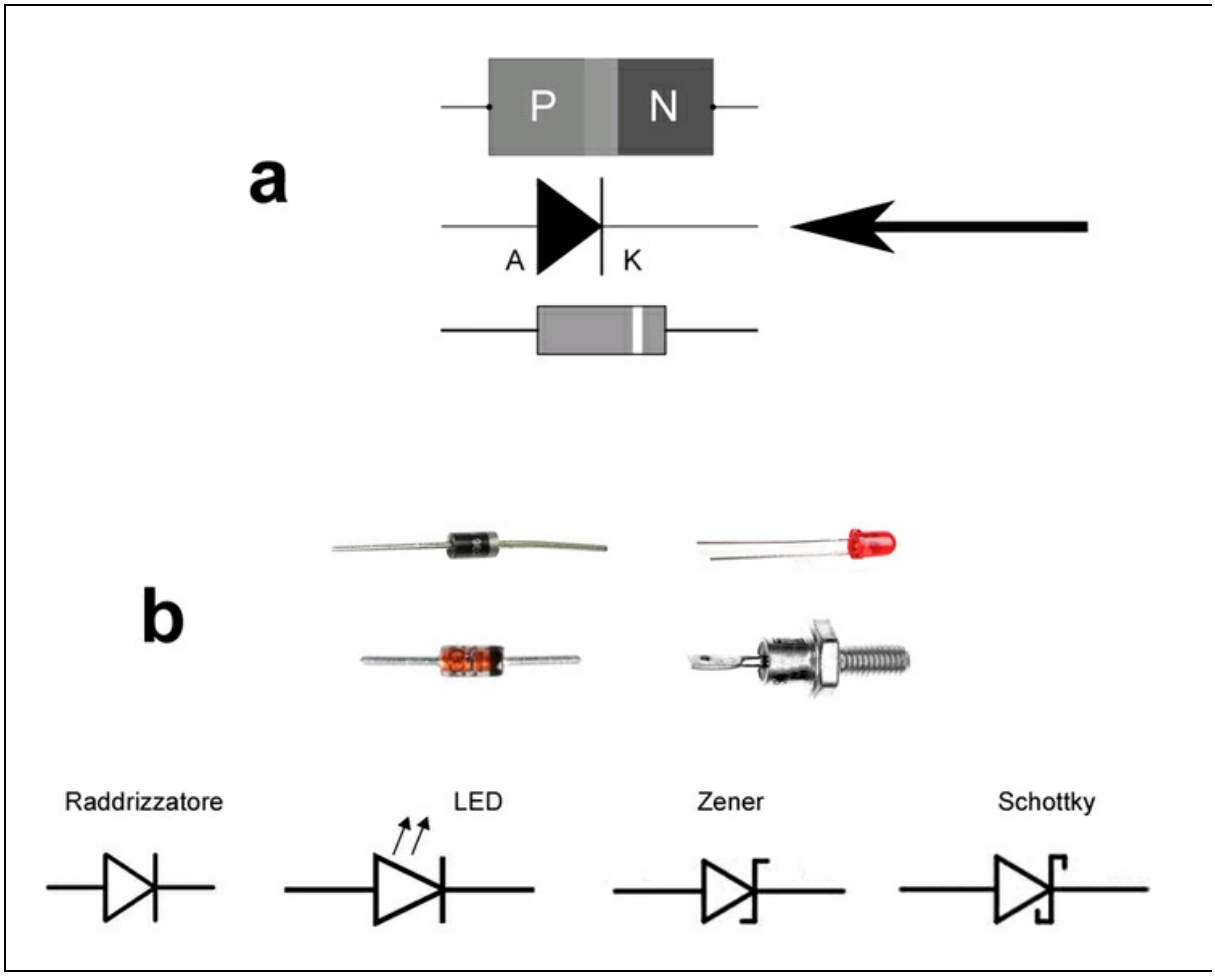


Figura 1.27 Funzionamento del diodo (a). Alcuni tipi di diodo e simboli elettrici corrispondenti (b). Da sinistra, diodo raddrizzatore normale, diodo LED, diodo Zener, diodo Schottky.

In elettronica esistono molti tipo di diodi, per cui vale la pena esaminarne i tipi più comunemente usati. È da notare che l'eventuale fascetta sul corpo del diodo indica sempre il catodo.

Diodo a giunzione

Il più comune diodo è di tipo a giunzione (PN). Come già detto, quando un diodo a giunzione viene attraversato da una corrente continua si comporta come un resistore, il cui valore resistivo varia in base alle caratteristiche del costruttore. La cosiddetta tensione di soglia di un

normale diodo a giunzione si aggira intorno ai 0,6 V. Se viene percorso dalla corrente in senso opposto, si comporta invece come un scudo impenetrabile.

Se il diodo viene attraversato da una corrente alternata, dell'intera onda viene fatta passare solo la semionda positiva e viene bloccata la semionda negativa. Disponendo un ponte di diodi in modo opportuno, ovvero un ponte di Graetz (Figura 1.28a), si può "raddrizzare" la corrente alternata e produrre un flusso di corrente continua.

Per esempio, utilizzando quattro comuni diodi a giunzione della serie 1N400*n* (dove *n* = 1, 2, 3, 4, 5, 6 e 7 a seconda della tensione e della potenza dissipata dal diodo) si può facilmente creare un ponte raddrizzatore e ottenere una corrente continua "pulsante" da un trasformatore in corrente alternata.

La tensione in uscita è pulsante perché è formata dalla serie di semionde positive (il tipico effetto *ripple*). Si può tentare di livellare la tensione ponendo in parallelo all'uscita del ponte un condensatore elettrolitico di circa 47 o 100 µF. La carica e scarica del condensatore permette di ottenere un buon livellamento delle semionde pulsanti.

Esistono in commercio ponti raddrizzatori di diverse potenze pronti all'uso, che talvolta fanno risparmiare un po' di tempo (Figura 1.28b). Basta collegare all'ingresso il generatore di corrente alternata, per esempio un trasformatore, per avere in uscita la corrente continua. Essendo comunque una corrente continua pulsante è necessario usare anche in questo caso un condensatore elettrolitico di livellamento, messo in parallelo all'uscita.

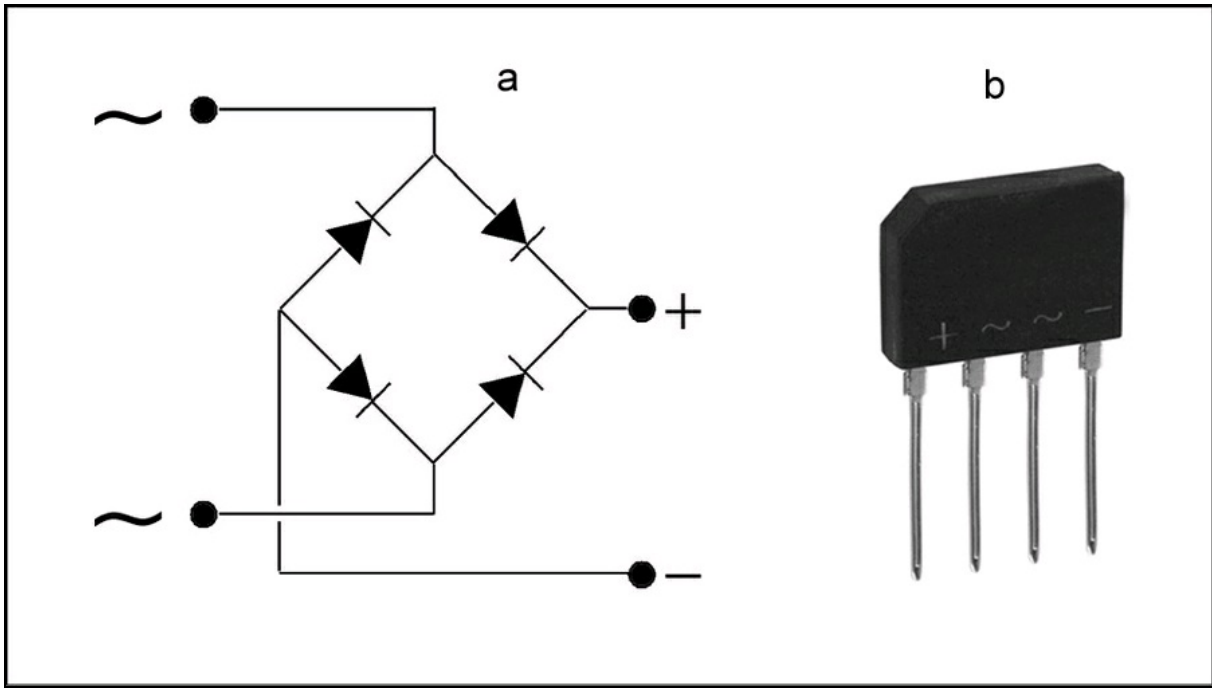


Figura 1.28 Schema di un ponte raddrizzatore di Graetz con 4 diodi (a). A destra, un ponte raddrizzatore pronto all'uso (b).

LED

Il LED (dall'inglese *Light Emitting Diode*) è sicuramente il diodo che si "vede" di più, anche perché è onnipresente in tutti gli apparecchi elettronici. Oltre che per gli scopi tradizionali di monitoraggio dell'accensione di qualsiasi dispositivo e di altre funzioni di controllo e misurazione, i LED hanno ormai invaso il mercato dell'illuminazione domestica e pubblica, visto che anche i semafori e le segnalazioni stradali sono sempre più spesso illuminati a LED. Senza parlare del settore automobilistico. Nella Figura 1.29 sono illustrati vari tipi di LED colorati.

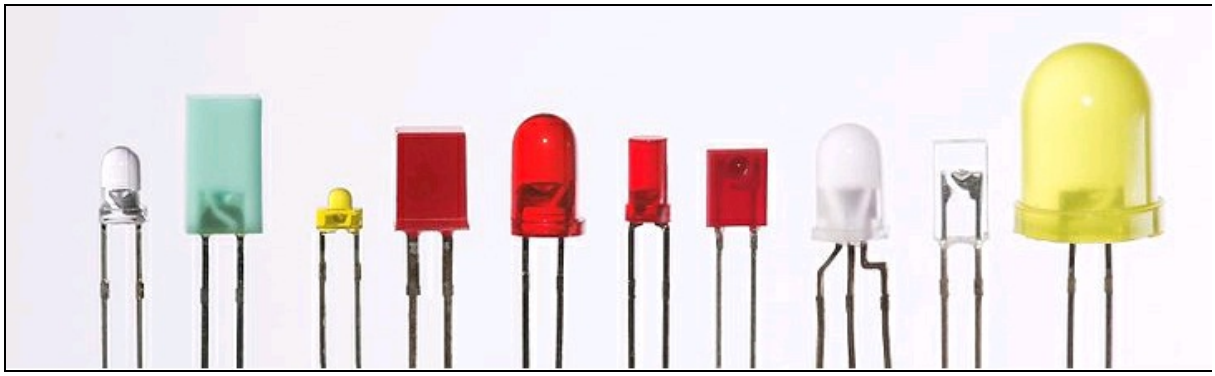


Figura 1.29 Vari tipi di LED colorati.

In base alla composizione chimica della giunzione è possibile trovare in commercio varie colorazioni del diodo LED. Nella Tabella 1.2, la prima colonna indica il colore, la seconda la composizione chimica e la terza la lunghezza d'onda della luce emessa.

Tabella 1.2 Alcune colorazioni dei diodi LED.

Colore	Composizione	Lunghezza d'onda
Infrarosso	Arseniuro di gallio e alluminio	Superiore a 780 nm
Rosso	Arseniuro di gallio e alluminio	620-780 nm
Arancione	Fosfuro arseniuro di gallio	590-620 nm
Giallo	Fosfuro arseniuro di gallio	570-590 nm
Verde	Nitruro di gallio e indio	495-570 nm
Blu	Seleniuro di zinco	450-495 nm
Viola	Nitruro di gallio e indio	380-450 nm
Ultravioletto	Diamante	Inferiore a 450 nm
Bianco	LED blu + fosfori gialli	LED blu + fosfori gialli

I LED, come tutti i diodi, hanno un terminale positivo (anodo) e uno negativo (catodo) e devono essere polarizzati in modo diretto per accenderli.

In pratica, è necessario collegare l'anodo al polo positivo dell'alimentazione e il catodo va collegato al polo negativo. Nei comuni LED cilindrici da 5 mm di diametro, il catodo è riconoscibile da una tacca sul contenitore plastico e l'anodo è sempre il terminale più lungo.

Nel caso ci si trovi di fronte un LED usato con i terminali tagliati e senza tacca di riferimento sul contenitore, si può tentare di riconoscerne la polarità guardandolo in controluce: l'anodo è sempre sottile e a forma appuntita, mentre il catodo è più grande e somiglia a una piccola bandiera (Figura 1.30a). Disponendo di un multimetro, ovviamente, la cosa diventa ancora più semplice: impostando il multimetro su Ω × 1, basta collegare il puntale rosso (+) e il puntale nero (-) ai terminali del LED. Se il LED si accende debolmente è polarizzato correttamente.

Per utilizzare correttamente un LED senza correre il rischio di bruciare la delicata giunzione, è necessario alimentarlo con valori di tensione e corrente adeguati. La caduta di tensione ai capi di un LED può variare da 1 volt a 1,6 volt e l'assorbimento è di circa 20-30 mA.

Anche la luminosità e la densità del colore dipendono dalla tensione applicata, per cui è bene alimentare il LED con una corretta tensione piuttosto che esagerare correndo il rischio di bruciarlo o, al contrario, sottoalimentarlo ottenendo una luminosità scadente.

Nella quasi totalità dei casi bisogna inserire un resistore in serie; per il calcolo del valore resistivo si ricorre alla consueta formula derivata dalla legge di Ohm:

$$R = V / I$$

Conoscendo la tensione di alimentazione del circuito, la tensione di soglia del LED e la corrente che deve circolare nel LED, è sufficiente sostituire i valori nella formula.

Supponendo di alimentare il circuito con una pila a 9 V, sapendo che il LED ha una tensione di soglia di 1,5 V e una corrente di 0,03 A (30 mA), avremo:

$$R = (9 - 1{,}5) / 0{,}03 = 7{,}5 / 0{,}03 = 250\ \Omega$$

Quindi, mettendo in serie un resistore di circa 250 Ω, ovvero compreso fra i valori standard da 220 o 270 Ω, il LED verrà alimentato in maniera adeguata (Figura 1.30b). Va da sé che aumentando o

diminuendo la tensione di alimentazione, il valore del resistore dovrà cambiare di conseguenza. Se si alimentasse il LED con una pila da 1,5 V potrebbe non essere necessario mettere un resistore in serie oppure il LED potrebbe non accendersi.

I LED monocromatici sono disponibili anche in strisce di LED per insegne luminose o per l'illuminazione casalinga o industriale (Figura 1.30c). Di solito le strisce di LED sono dotate di un controller che alimenta la striscia a 12 V.

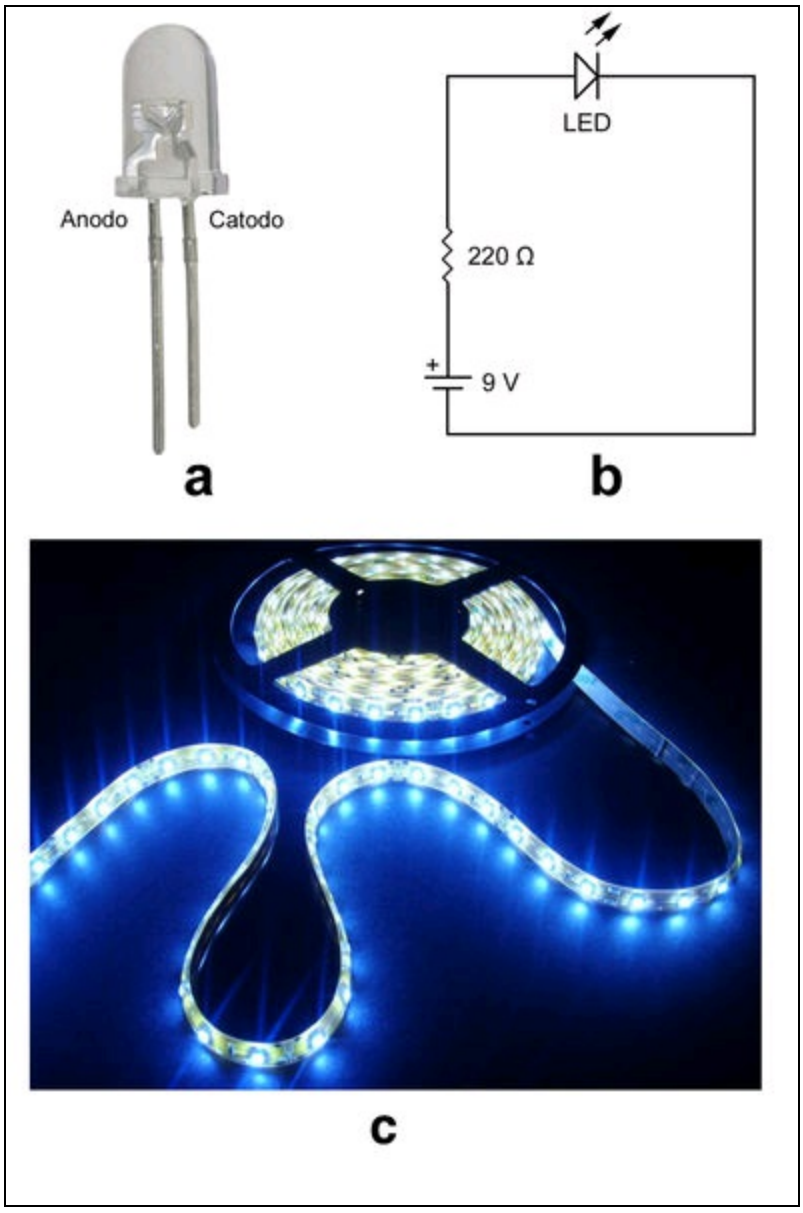


Figura 1.30 Un LED nel suo tipico contenitore plastico con anodo e catodo facilmente distinguibili in controluce (a). Tipico circuito di alimentazione di un LED (b). Una striscia di LED (c).

Diodo LASER

I diodi LASER sono particolari LED che emettono luce stimolata LASER, come dice la sigla *Light Amplification by Stimulated Emission of Radiation*.

La maggior parte dei diodi laser è realizzata in arseniuro di gallio e alluminio ed emette luce LASER quando la corrente di soglia supera i 20 o 30 mA.

La radiazione controllata di luce coerente ne permette l'uso in vari campi dell'elettronica di consumo e nell'elettromedicale. I diodi LASER vengono impiegati largamente per la scrittura/lettura di CD e supporti digitali affini, in macchine CNC per il taglio laser, così come nei mouse ottici, nei lettori di codici a barre, nella trasmissione dati tramite fibre ottiche, nei puntatori, nei segnalatori puntiformi e così via (Figura 1.31).

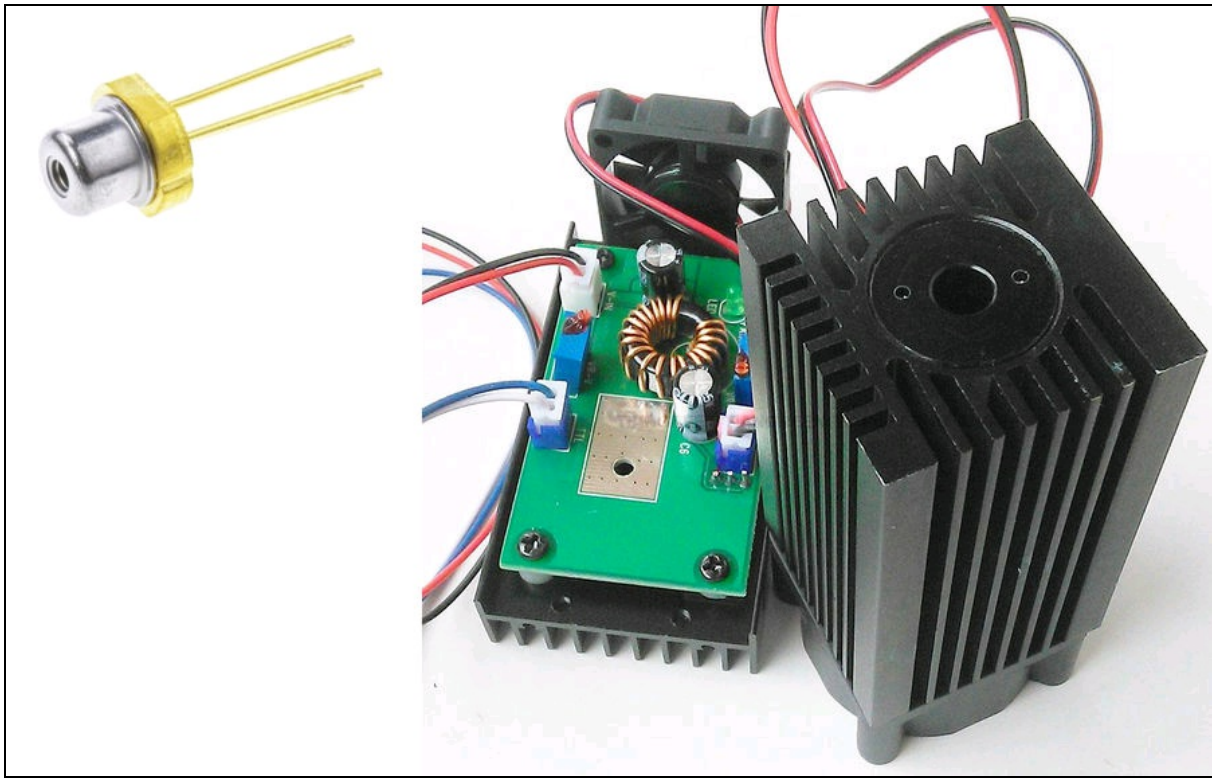


Figura 1.31 A sinistra, un diodo laser e a destra un driver per il taglio laser da montare su una macchina CNC fai da te.

LED RGB

Si tratta di un particolare LED composto da tre giunzioni con un terminale comune (catodo o anodo) in grado di emettere luce rossa (Red), verde (Green) e blu (Blue) (Figura 1.32a).

In pratica, con la combinazione dei tre colori è possibile ottenere ben 16 777 216 variazioni di colore, in un circuito di controllo a 24 bit (3 × 8 bit).

Essendo 256 i valori possibili per ogni byte di colore, il risultato è il prodotto di 256 × 256 × 256 = 16 777 216. Sono le stesse combinazioni colore dei display LCD a colori o degli schermi TV a LED o dei sensori CCD.

Per l’alimentazione di un LED RGB valgono le stesse considerazioni fatte per i LED singoli.

Anche i LED RGB sono disponibili in strisce (b).

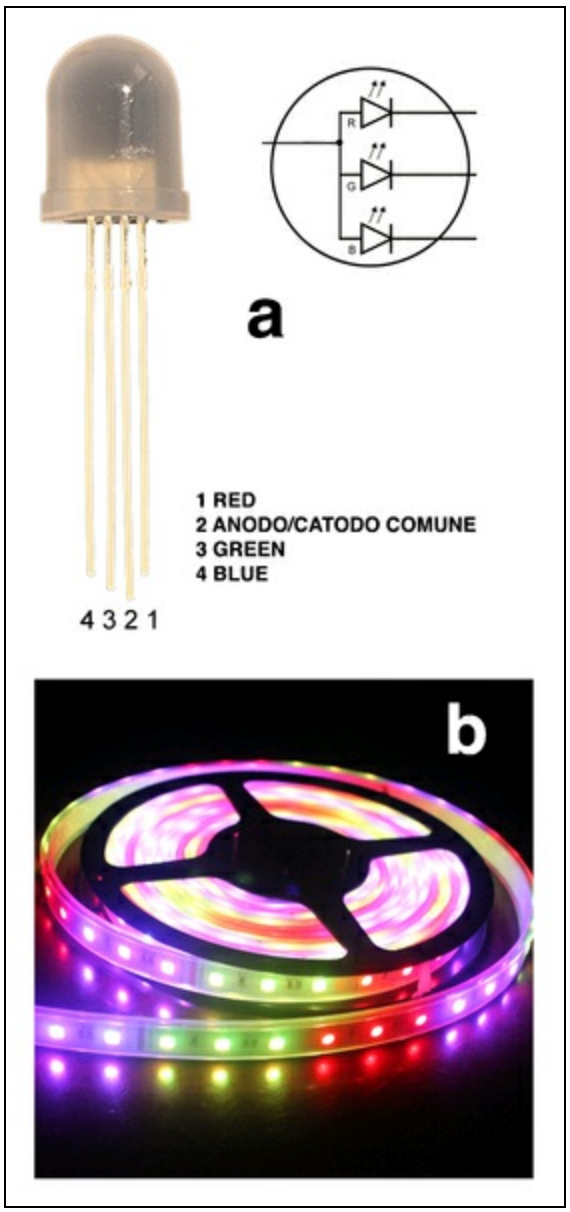


Figura 1.32 Un tipico LED RGB e il suo simbolo elettrico (a). Una striscia di LED RGB (b).

Matrice di LED

Un altro sistema di visualizzazione molto diffuso che sfrutta i LED, sia monocromatici sia RGB, è la cosiddetta matrice di LED. Si tratta di una serie di LED collegati in senso orizzontale e verticale a formare una matrice a catodo o anodo comune. Fornendo in modo opportuno i dati di ON/OFF alle righe e alle colonne della matrice si può costruire un display personalizzato nel quale visualizzare qualsiasi combinazione di punti accesi e spenti.

Esistono prodotti già pronti all'uso come il LED Matrix Tri Color, distribuito da Sparkfun e visibile nella Figura 1.33.

Questa grande matrice ha 64 LED rossi, 64 LED verdi e 64 LED blu integrati in un unico alloggiamento a catodo comune facilmente collegabile alle porte di un controller.

Si possono collegare i moduli fra di loro per creare matrici di LED più grandi in base alle esigenze. Di solito, si assemblano più matrici di LED per visualizzare scritte scorrevoli, disegni o, negli schermi più grandi (video wall), anche foto e filmati.

La Figura 1.33 illustra una matrice di LED RGB spenta e accesa e lo schema di collegamento interno dei LED.

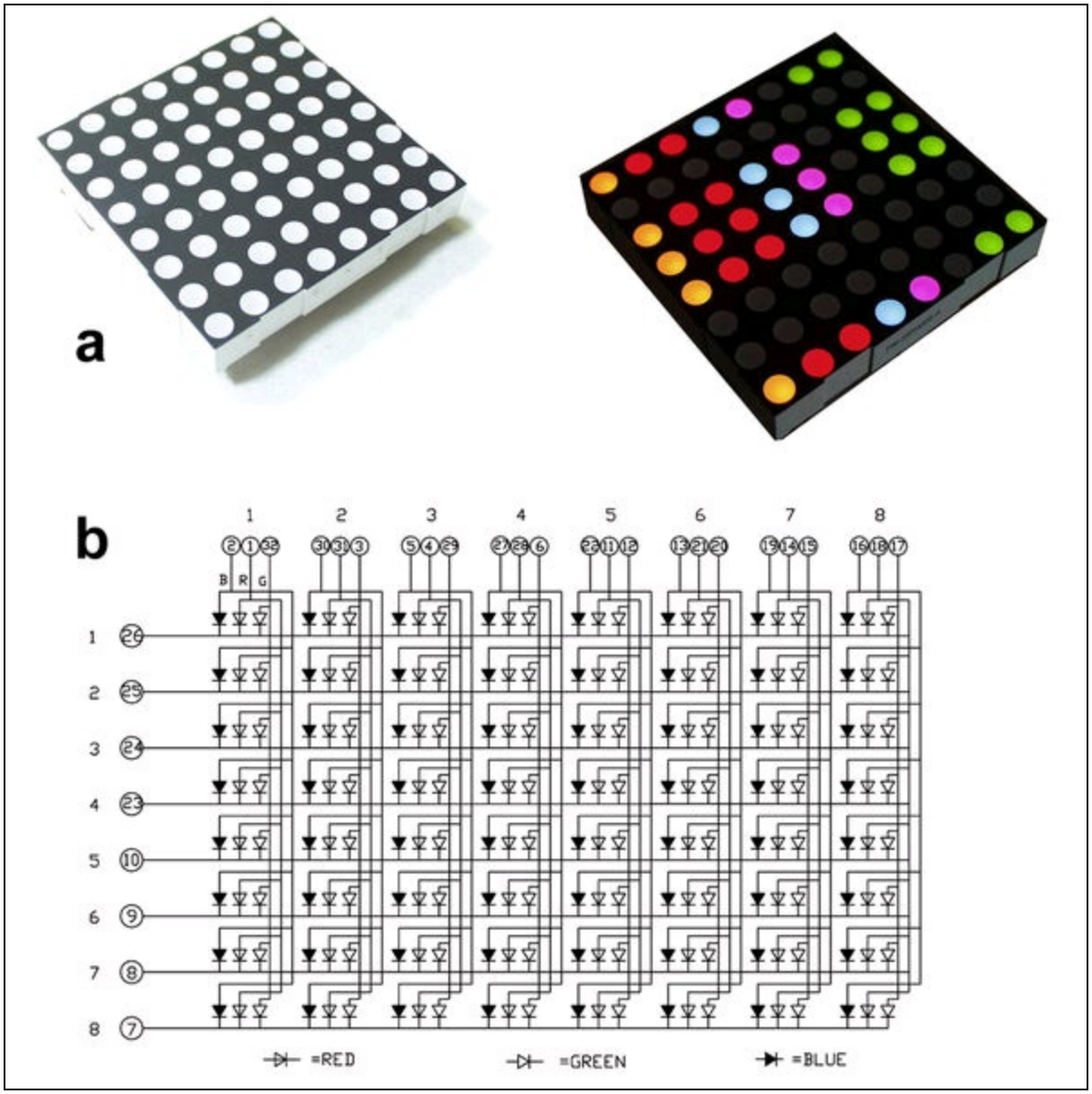


Figura 1.33 LED Matrix Tri Color, distribuito da Sparkfun (a). Schema di collegamento interno della matrice dei LED RGB (b).

Transistor

Il transistor è un componente attivo a semiconduttore, in genere composto di silicio, munito di tre piedini, detti reofori, disponibile in un vasto numero di modelli, dimensioni e forme (Figura 1.34).

Il suo nome deriva dalla contrazione dei termini *Transfer* e *Resistor*, ossia “trasferimento di resistenza” e, in pratica, quello che accade al suo interno è proprio questo: la resistenza ai capi dei terminali di uscita può essere fatta variare in base alla tensione e corrente in ingresso. Pertanto il transistor può essere usato principalmente come amplificatore del segnale posto al suo ingresso oppure come interruttore elettronico.

Nei circuiti di amplificazione, la tensione e la corrente in uscita di un transistor possono essere anche molto elevate rispetto alla tensione e alla corrente immesse in entrata. Nei cosiddetti circuiti *switching* i transistor si comportano come perfetti interruttori elettronici.

Per queste e altre particolari caratteristiche, i transistor sono impiegati in tutti i settori dell’elettronica analogica e digitale. La loro miniaturizzazione, a livello microscopico, ha consentito la costruzione di circuiti integrati sempre più piccoli, fino alla progettazione attuale di CPU (*Central Processing Unit*) con miliardi di transistor in un singolo *chip*.

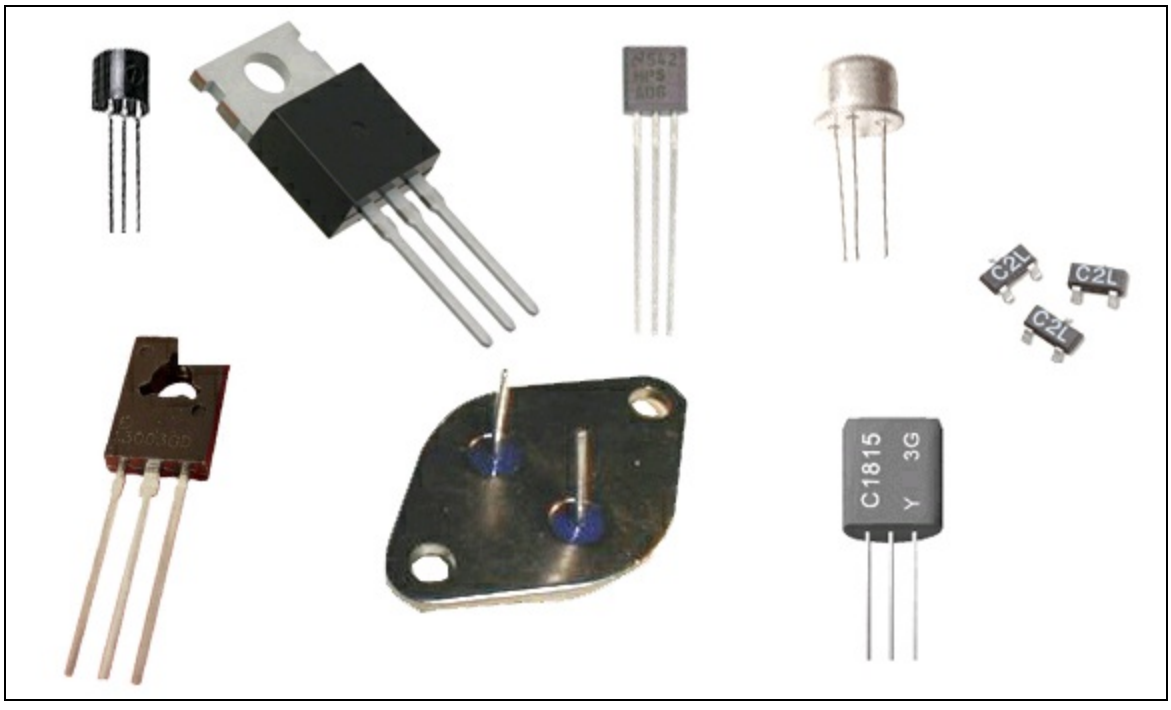


Figura 1.34 Vari tipi di contenitori per transistor.

Il funzionamento del transistor è basato sul principio della giunzione PN vista in precedenza per il diodo. Per cui non è errato immaginare il transistor come due diodi accoppiati. Le caratteristiche funzionali del transistor come amplificatore o come interruttore derivano dalla costruzione interna delle barrette di silicio drogato in modo che le giunzioni possono condurre cariche elettriche positive o negative, a seconda del tipo di drogaggio P o N. In base alla direzione della corrente, il transistor può trovarsi in un stato di conduzione o di interdizione, cioè può far passare la corrente o bloccarla (vedi l'esperimento più avanti).

I tipi più comuni di transistor usati nei circuiti elettronici sono quelli a giunzione NPN o PNP:

- il transistor NPN è formato da una barretta di silicio negativa, da una positiva e da un'altra negativa;
- il transistor PNP è formato da una barretta di silicio positiva, da una negativa e da un'altra positiva.

In pratica, essendo costituito dall'unione di due giunzioni PN (diodo), il transistor di questo tipo viene chiamato BJT (*Bipolar Junction Transistor*), ovvero a giunzione bipolare

I tre terminali del transistor BJT prendono il nome di:

- base (b);
- collettore (c);
- emettitore o anche emittore (e).

Nella Figura 1.35 sono illustrati i simboli elettrici dei transistor e la schematizzazione del loro funzionamento.

È da notare che la disposizione dei tre terminali non è sempre la stessa e varia da un tipo di transistor all'altro, a seconda dell'involucro TO (*Transistor Outline*).

Nei transistor più comunemente usati, una tacca sull'involucro (TO-39) evidenzia la posizione dell'emittore.

In altri casi, è la diversa conformazione dell'involucro che permette di conoscere la disposizione dei terminali. Molto spesso è necessario ricorrere al datasheet del transistor per essere sicuri della disposizione dei terminali. Un posizionamento errato nel circuito causa, il più delle volte, la rottura del transistor. Per nostra fortuna, oggi esistono ottimi motori di ricerca su Internet per ottenere gratuitamente le informazioni su praticamente tutti i transistor e circuiti integrati attualmente in commercio. Uno fra i tanti è www.datasheetcatalog.net.

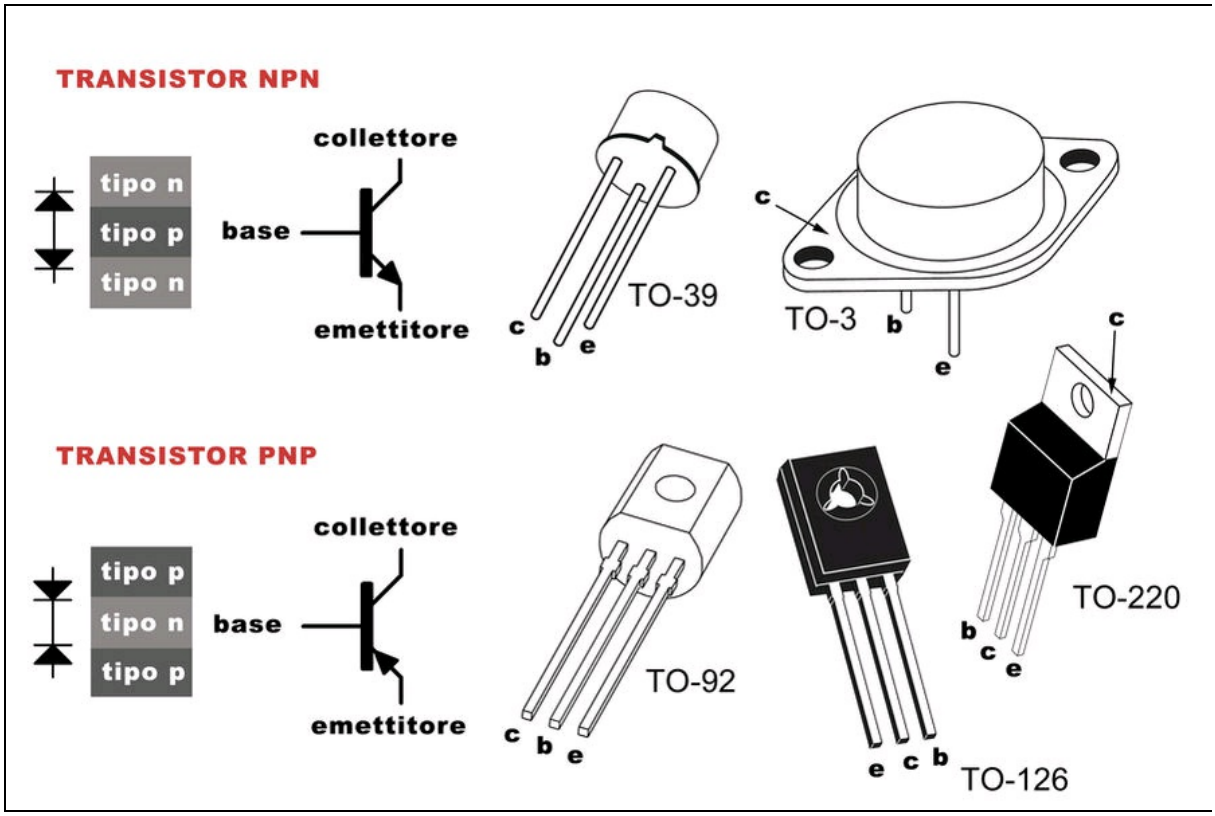


Figura 1.35 Simboli elettrici dei transistor NPN e PNP di tipo BJT e schema di funzionamento del transistor. A destra vari modelli di involucro e posizionamento dei terminali b, c, e.

Un esperimento con il transistor

Per capire meglio il concetto di interdizione e conduzione di un transistor NPN o PNP, questo esperimento illustra il comportamento di un “interruttore controllabile” in corrente. Si tratta di un esperimento che tutti possono fare senza correre rischi di scosse elettriche o altro.

Il circuito è formato da un comune transistor NPN di qualsiasi tipo, per esempio un BC547, da un LED e da una pila a 9 volt. Eseguire i collegamenti come illustrato nella Figura 1.36.

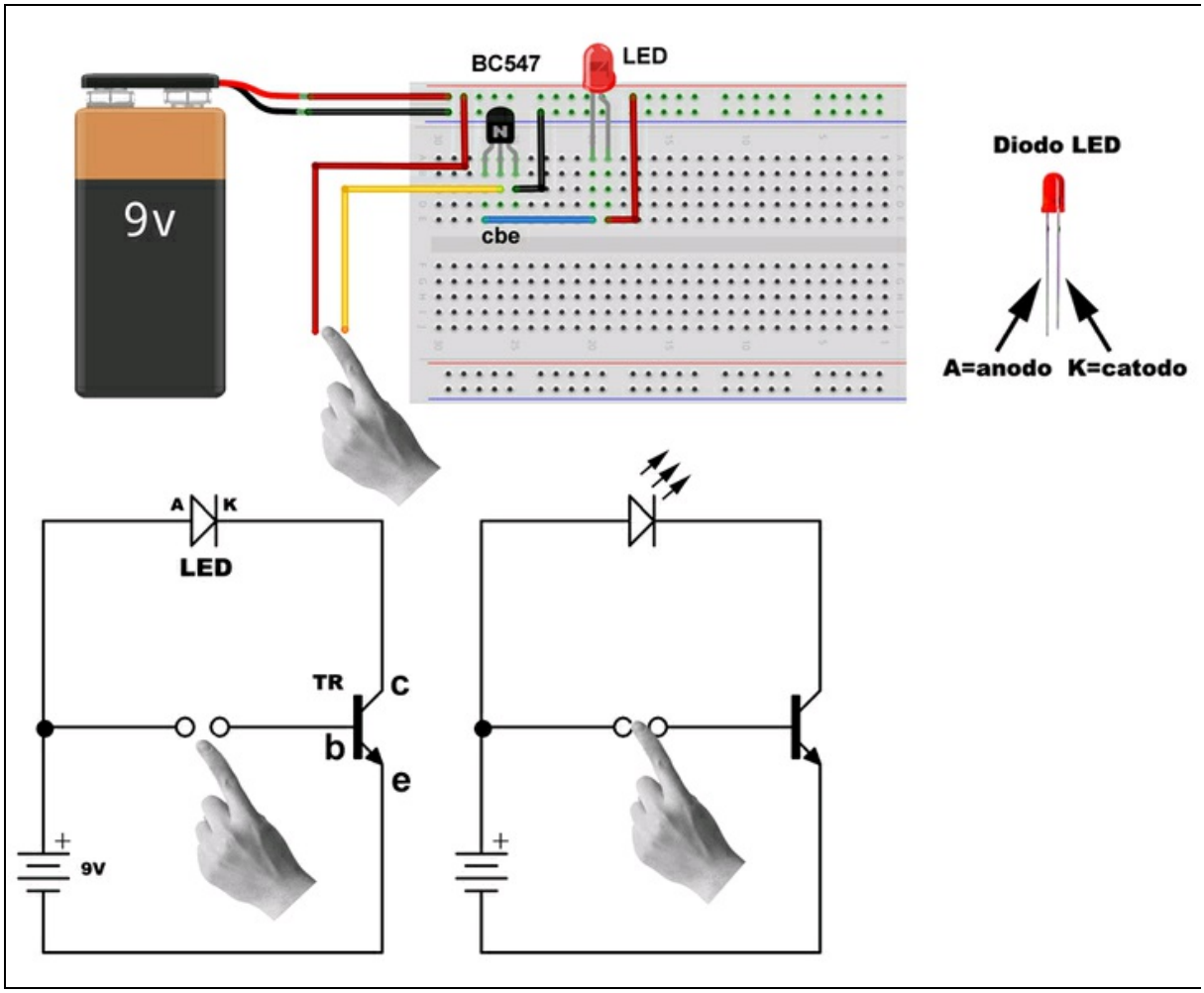


Figura 1.36 Un transistor usato come interruttore controllabile. Appoggiando un dito per chiudere il circuito, si polarizza la base del transistor, che entra in conduzione permettendo l’accensione del LED. Premendo con più forza, il LED si illumina di più.

1. Collegare il LED (attenzione alla polarità anodo-catodo) al collettore del transistor.
2. Alimentare il circuito portando la tensione negativa all'emettitore e la tensione positiva all'anodo del LED.
3. Il LED rimarrà spento, perché non c'è alcuna corrente circolante fra collettore ed emettitore.
4. Portiamo una piccola corrente alla base del transistor, ponendo semplicemente un dito fra il polo positivo della pila e la base del transistor, il LED si accenderà debolmente.
5. Premendo più energicamente il dito sui contatti, si otterrà una maggiore luminosità del LED. Questo perché, in pratica, si diminuisce il valore resistivo e quindi anche la debole corrente della pila "attraverserà" il dito più facilmente, permettendo l'accensione del LED.

Quello che succede in pratica è che il transistor passa da uno stato di interdizione a uno stato di conduzione, grazie alla chiusura del circuito per mezzo del dito, permettendo il passaggio di corrente fra collettore ed emettitore.

Ovviamente, al posto del dito possiamo inserire un potenziometro e quindi misurare la corrente assorbita da circuito. Tramite un multimetro digitale impostato su 200 mA fondo scala, si può osservare il cambiamento dello stato del transistor in modo più analitico (Figura 1.37).

Da questo esperimento di intuisce la necessità di far lavorare un transistor nel modo migliore possibile. Questo si traduce nella cosiddetta "polarizzazione" di un transistor.

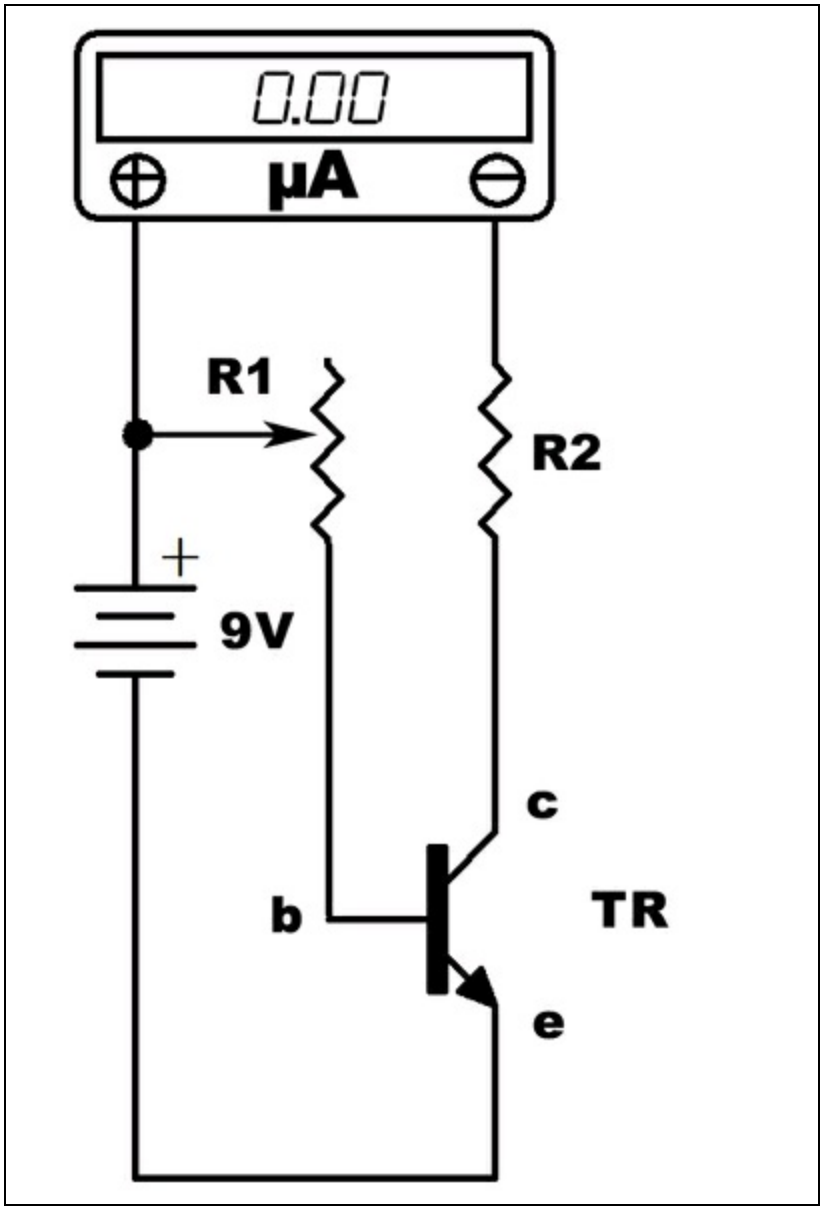


Figura 1.37 Un transistor come un interruttore controllabile. TR è un transistor BC547. Il potenziometro R1 è di 1 kΩ (ma può avere un qualsiasi altro valore), il resistore R2 è di 1 kΩ 1/4 di watt. L'alimentazione è una pila da 9V. Il multimetro digitale è impostato su 200 mA fondo scala.

Polarizzazione

La polarizzazione di un transistor è di fondamentale importanza, perché solo con una corretta polarizzazione si può determinare il

cosiddetto “punto di lavoro” ottimale del transistor.

La miriade di transistor in commercio serve a far capire che ogni transistor ha caratteristiche ben precise e può funzionare solo se inserito in un circuito ben progettato. Per esempio, se un segnale deve essere amplificato in maniera adeguata, bisogna trovare il punto di lavoro ottimale del transistor utilizzato a quello scopo.

Per capire come impiegare correttamente un transistor in un circuito non basta una trattazione teorica, per cui abbiamo preferito realizzare un semplice circuito sperimentale per fissare alcuni concetti di base che serviranno in futuro, per circuiti più complessi.

Se nell’esperimento precedente abbiamo visto come il transistor passa dallo stato di interdizione a quello di conduzione, semplicemente ponendo un dito (o un potenziometro) fra il polo positivo della pila e la base del transistor, nella Figura 1.38 viene illustrata come la conduttività elettrica tra collettore ed emettitore del transistor non può verificarsi finché non viene applicata alla base una tensione specifica, calcolando la cosiddetta “resistenza di polarizzazione”.

Nel semplice circuito sono stati utilizzati i seguenti componenti, facilmente reperibili:

- TR = transistor 2N2222 (BC107, BC 237, 2N1711…);
- R1 = resistenza 100 Ω 1/4 di Watt;
- pila da 9 V;
- multimetro digitale impostato su 200-20 mA fondo scala.

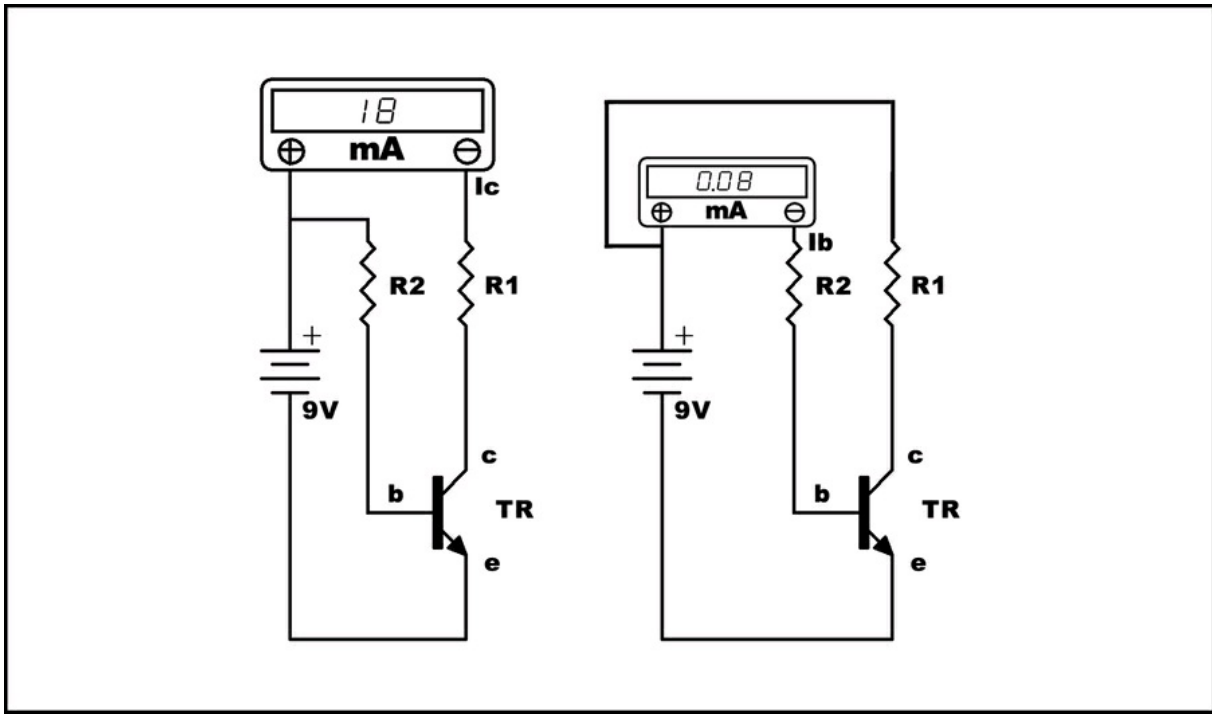


Figura 1.38 Polarizzazione di un transistor. Il transistor TR è un 2N2222. La resistenza R1 è di 100 Ω 1/4 di watt. La resistenza R2 è di 100 KΩ 1/4 watt. L'alimentazione è una pila da 9 V. Il multimetro digitale è impostato su 200 mA fondo scala.

Innanzitutto, il piedino corrispondente alla base del transistor deve rimanere libero. Una volta applicata la tensione di 9 volt della pila, si vedrà che il transistor, nella giunzione collettore-emettitore, non condurrà nulla (o quasi) e il display del multimetro digitale resterà a 0,00 mA.

Quando la base del transistor viene "polarizzata" tramite il resistore R2 da 100 KΩ, il multimetro misurerà una corrente di collettore (Ic) di circa 18 mA.

Collegando il multimetro fra base e polo positivo della pila, si misurerà una corrente di base (Ib) di circa 0,08 mA.

Anche senza misurarla direttamente, è possibile arrivare allo stesso valore di corrente di base ricorrendo alla legge di Ohm e utilizzando i valori di tensione e resistenza in gioco:

9 V (tensione del circuito) - 0,7 (caduta di tensione base-emettitore) = 8,3 V resistenza di collettore = 100 KΩ

Ib (corrente alla base) = 8,3 : 100 000 = 0,000083 Ampere (0,083 mA)

Risulta chiaro che, polarizzando opportunamente la base con un resistore che porta una debole corrente dalla stessa linea di alimentazione collegata al collettore (nel nostro caso il polo positivo della pila), si ha un aumento del passaggio di corrente fra collettore ed emettitore.

È come se la corrente venisse “amplificata” e la base fungesse da “controllo” di flusso fra collettore ed emettitore. Una specie di “rubinetto” elettronico.

Coefficiente di amplificazione

Per far lavorare un transistor al massimo delle sue possibilità bisogna inserirlo in un circuito elettrico in cui i valori delle resistenze di polarizzazione assumono un ruolo determinante.

La trattazione di questo argomento prevede la conoscenza di molti altri parametri del transistor che normalmente si trovano nei datasheet (Vcb, Vce, Veb, Ic, hFE, Ptot e così via) e sarebbe troppo lungo parlarne in questa sede.

Diciamo solo che il rapporto che intercorre fra la corrente di collettore (Ic) e quella di base (Ib) si traduce normalmente nel “coefficiente di amplificazione” o “beta” del transistor (contrassegnato dalla lettera greca β).

In pratica, è un numero che indica quanto è il “guadagno” del transistor.

Nel nostro caso, sperimentato nell’esempio precedente, il coefficiente beta è:

18 mA (Ic) : 0,08 mA (Ib) = 200

È ovvio che il guadagno di un transistor può cambiare sensibilmente a seconda delle caratteristiche “fisiche” del componente e, di solito, s’intende come valore medio. Diciamo che in uno stesso modello di transistor, ci possono essere variazioni abbastanza consistenti, per cui si può considerare per un transistor 2N2222 un beta medio che va da 100 a 300 (come indicato nel suo datasheet) per cui, facendolo lavorare con un guadagno di 200, possiamo stare tranquilli che non si surriscalderà troppo. Se il transistor dovesse scaldarsi troppo, è possibile raffreddarlo con appositi dissipatori di calore.

In progettazioni elettroniche molto sofisticate, per le quali servono coefficienti ben precisi, la ricerca dei componenti è determinante ai fini della qualità del prodotto da costruire. Si arriva a scartare decine e decine di transistor provandoli uno per uno per determinare con sicurezza il coefficiente beta “giusto”.

Transistor MOSFET

La trattazione profonda dei MOSFET (dall’inglese *Metal-Oxide-Semiconductor Field-Effect Transistor*) richiederebbe molto spazio. Per dovere di cronaca, diamo uno sguardo ai MOSFET, ovvero ai transistor a effetto di campo basati su semiconduttore a ossido di metallo, largamente usati in elettronica sia analogica che digitale.

A differenza del transistor BJT, il MOSFET è composto da uno strato di silicio drogato, dal quale vengono derivati tre terminali:

- Gate (G);
- Source (S);
- Drain (D).

In pratica, applicando una tensione al Gate è possibile controllare il passaggio di corrente tra il terminale Source e il terminale Drain. Da qui

si intuisce che il suo comportamento è più simile a un interruttore polarizzato.

Nella Figura 1.39 sono illustrati i vari simboli elettrici e la schematizzazione del suo funzionamento.

A seconda del drogaggio di tipo N o di tipo P, il transistor può essere nMOS o pMOS, ovvero MOSFET con Gate negativo e MOSFET con Gate positivo.

Quando il MOSFET viene polarizzato negativamente o positivamente, a seconda del canale, viene creato nel substrato un canale n o p, in base al tipo di substrato p o n corrispondente.

Per le loro caratteristiche costruttive innovative, i MOSFET hanno praticamente soppiantato i transistor BJT nello sviluppo delle tecnologie digitali e la logica digitale CMOS ha di fatto permesso la progettazione di computer sempre più performanti.

Nei circuiti analogici i MOSFET vengono usati nel settore consumer come amplificatori audio di potenza, inverter e alimentatori switching e nel settore industriale come interruttori o resistori di precisione.

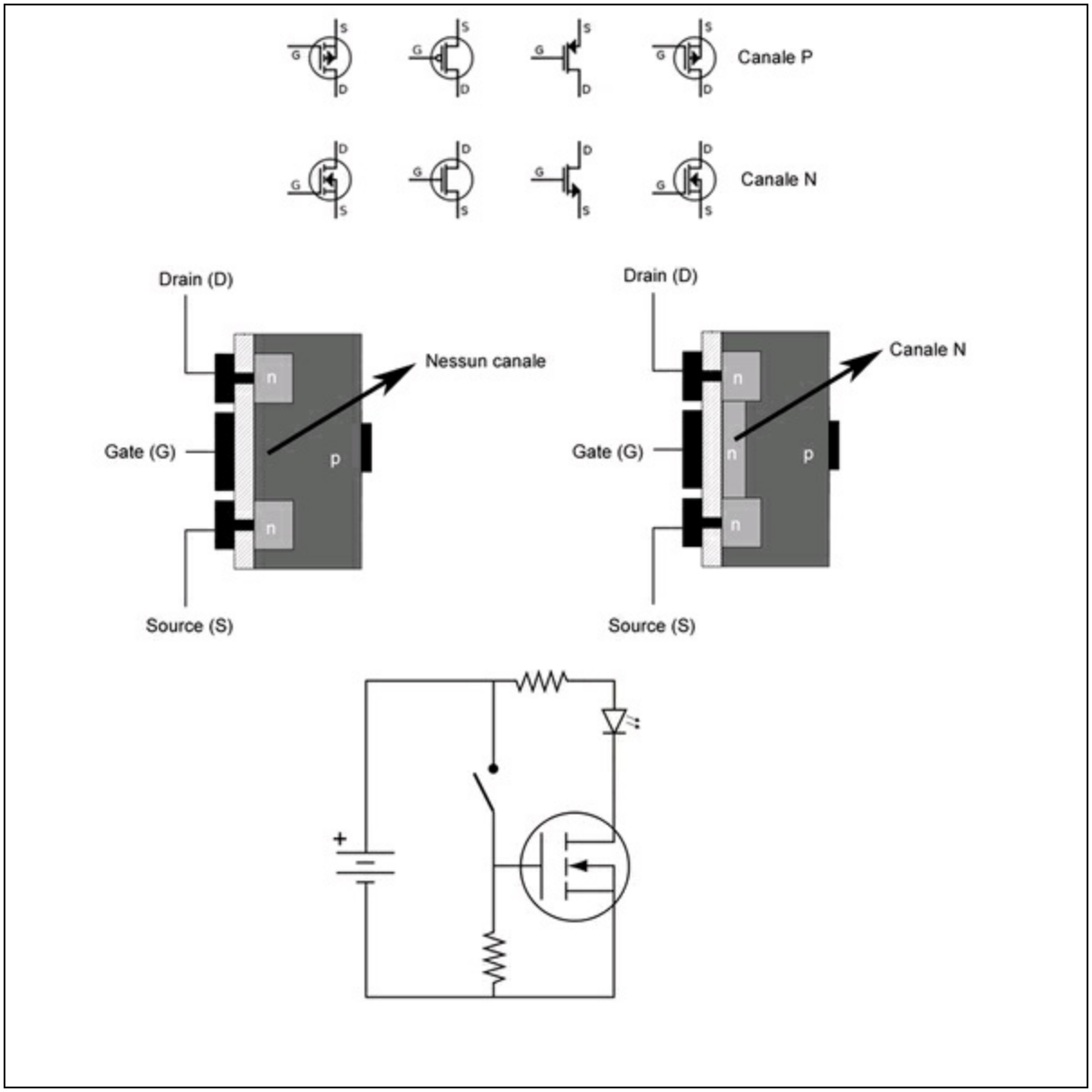


Figura 1.39 Simboli dei MOSFET e funzionamento schematizzato. In basso, un piccolo circuito di prova.

Circuito integrato

Il circuito integrato, siglato comunemente con IC (dal termine inglese *Integrated Circuit*), o CI negli schemi italiani, è un circuito elettronico miniaturizzato composto principalmente da un certo numero di

transistor, diodi e anche componenti passivi come resistori e condensatori ottenuti dalla lavorazione di un *chip* di silicio, ovvero di una “fettina” di materiale semiconduttore. Tant’è che i circuiti integrati vengono spesso chiamati amichevolmente *chip*. Il livello di integrazione può limitarsi a pochi transistor o arrivare a milioni di transistor:

- SSI (*Small-Scale Integration*) fino a 10 transistor;
- MSI (*Medium-Scale Integration*) fino a 100 transistor;
- LSI (*Large-Scale Integration*) fino a 10 000 transistor;
- VLSI (*Very-Large Scale Integration*) fino a 100 000 transistor;
- ULSI (*Ultra-Large Scale Integration*) fino a 10 000 000 transistor;

Per le CPU multicore più potenti si arriva all’integrazione di oltre 2 miliardi di transistor.

I circuiti integrati si dividono in diverse famiglie a seconda del loro scopo e della tecnologia utilizzata, analogica o digitale.

I seguenti tipi di IC sono impiegati in circuiti analogici:

- amplificatori operazionali;
- amplificatori audio;
- amplificatori di alta frequenza;
- regolatori di tensione;
- sintonizzatori e mixer di alta frequenza;
- rivelatori e stadi di media frequenza;
- oscillatori di bassa frequenza.

In circuiti digitali sono impiegati i seguenti tipi di IC:

- porte logiche;
- contatori;
- convertitori;
- LED driver;
- driver per motori;
- divisori di frequenza;

- logiche programmabili;
- unità aritmetico logiche;
- microprocessori;
- memorie;
- fotoaccoppiatori.

Il tipico contenitore dei circuiti integrati è di tipo DIP (*Dual In-line Package*), ovvero con due file parallele da 3 fino a 64 piedini e oltre.

Per i circuiti integrati miniaturizzati o SMD (*Surface Mounting Device*) l'involucro può essere di tipo SO (*Small Outline*) o quadrato di tipo QFP (*Quad Flat Package*), specie per CPU e microcontroller, con piedinature che possono arrivare alle centinaia di pin.

Nei progetti di questo libro vengono usati alcuni circuiti integrati di tipo DIP, le cui caratteristiche verranno illustrate di volta in volta.

Per comodità di saldatura o per facilitare la sostituzione in caso di rottura, i circuiti integrati vengono montati su zoccoli precedentemente saldati alla basetta del circuito stampato.

Nella Figura 1.40 sono illustrati i simboli elettrici di alcuni IC:

- un fotoaccoppiatore 6N137;
- un regolatore di tensione 7805;
- un quad channel driver L293;
- un processore ATMega 328.

Nella stessa figura sono visibili anche gli zoccoli DIL (*Dual In Line*) con diverse piedinature. Notare lo zoccolo DIL con 16 pin a tulipano anziché di tipo lamellare. Molti preferiscono montare gli zoccoli a tulipano, perché i contatti con i piedini del circuito integrato sono più sicuri.

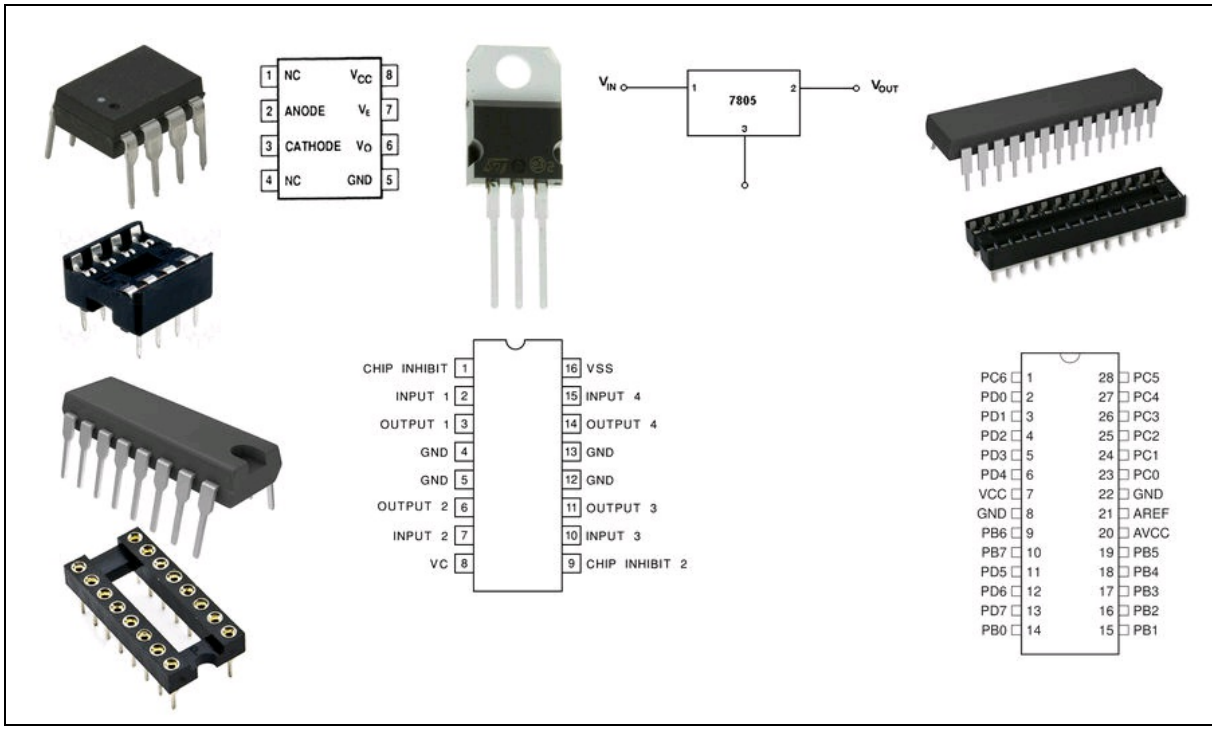


Figura 1.40 Alcuni circuiti integrati e rispettivi simboli. Da sinistra, un fotoaccoppiatore 6N137, un regolatore di tensione 7805, un quad channel driver L293, un processore ATMega 328.

Microprocessore

Come abbiamo già visto, l'ultra integrazione di transistor su larga scala (ULSI) in un solo chip di silicio permette la progettazione di computer dalle prestazioni impensabili fino a qualche anno fa.

In questo libro si farà largo uso del microprocessore ATMega 328: nato una decina di fa, è disponibile con contenitore DIP e quindi particolarmente facile da usare, montato regolarmente su schede tipo Arduino.

Si tratta di un processore AVR a 8 bit della Atmel, ma le sue prestazioni sono sicuramente superiori alle prime CPU degli anni Sessanta, grazie alla sua architettura RISC (*Reduced Instruction Set Computer*).

Micro-storia

Un orgoglio tutto italiano è l'invenzione della innovativa tecnologia *Silicon Gate Technology*, sviluppata nel 1968 da Federico Faggin (Vicenza, 1941) presso l'azienda americana Fairchild. Faggin riuscì a integrare la prima CPU in un solo chip di silicio.

Da allora fu possibile parlare di LSI e, dagli anni Ottanta in poi, i microprocessori sono diventati parte della nostra esistenza, entrando nelle case sotto forma di personal computer e poi di telefoni, di giochi e di robot.

Il primo microprocessore su singolo chip fu un Intel 4004, chiamato così perché corredato da altri circuiti integrati (4001, 40020 e 4003) per la gestione della memoria e dei dati. Possedeva 2 300 transistor, dati e registri a 4 bit, ma con istruzioni anche a 8 e 16 bit. L'indirizzamento della memoria RAM era di soli 640 byte.

Subito dopo il 4004 nacque il microprocessore 8008, il primo "8 bit" al mondo, che venne sviluppato diventando nel 1974 il celeberrimo Intel 8080, padre di una famiglia di nobili discendenti e competitor come lo Z80 della ZiLOG (fondata da Federico Faggin) e il Motorola 6800.

Bisogna arrivare verso la fine degli anni Settanta per vedere consolidata la tecnologia a 16 bit con processori del calibro dell'Intel 8086, il primo con l'architettura X86, arrivata al suo culmine con la famiglia Pentium in tempi molto recenti.

Il primo processore a 32 bit nel mercato dei personal computer fu il Motorola 68000 e i suoi derivati, che negli anni Ottanta fecero la fortuna dei personal computer Atari, Macintosh e Commodore. La risposta a 32 bit di Intel fu la CPU 80386 e successive (Figura 1.41).

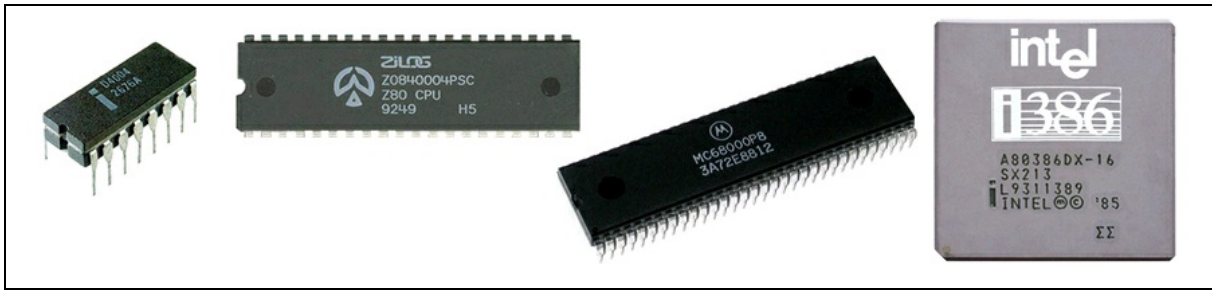


Figura 1.41 Alcuni storici microprocessori. Da sinistra: Intel 4004, il primo microprocessore su singolo chip della storia, ZiLOG Z80, Motorola 68000, Intel 80386.

Negli anni Novanta, la tecnologia a 64 bit consolidò con Intel e AMD nella lotta della velocità di clock a frequenze sempre più elevate. Ma seguendo la logica del raddoppio del numero di transistor integrabili nello stesso chip per raddoppiare le prestazioni (la nota legge di Moore) si è arrivati al numero massimo di transistor integrabili, per cui oggi si preferisce raddoppiare il numero di *core* piuttosto che diminuire sempre di più lo spazio nanometrico dei transistor nella CPU, incorrendo nel cosiddetto “effetto tunnel” ovvero nella interferenza a livello atomico fra elementi fisicamente troppo piccoli e troppo affollati. Una soluzione per diminuire anche l’eccessiva produzione di calore che le CPU a un solo core producono è l’utilizzo di più core, ovvero di più microprocessori su un singolo chip.

Dall’architettura *dual core* del 2005 si è passati dopo pochi anni al *multi core*, per arrivare oggi a CPU con ben 16 core, paventando già nel prossimo futuro 32 core e oltre.

Architettura

In generale, i circuiti integrati, per quanto complessi possano essere, sono disponibili per un ben preciso numero di funzioni. Per esempio, un Hex-Inverter (famiglia 7404) ha sei porte logiche per invertire l’ingresso logico (inversione da A a NOT A), un integrato ACIA (famiglia 6850) può convertire dati digitali seriali in paralleli e così via.

A differenza dei circuiti integrati, una CPU può essere dotata di un suo sistema operativo interno, ed elaborare programmi che possono venire scritti nella sua memoria interna (EPROM) o in memorie esterne (RAM).

Senza entrare nei dettagli di una trattazione specifica, diremo semplicemente che una CPU risponde a una determinata architettura. Per esempio, l'architettura Von Neumann, che prevede dati e istruzioni nella stessa memoria del processore, o l'architettura Harvard, che prevede dati e istruzioni in due memorie separate.

Semplificando molto, all'interno di una CPU si possono trovare i seguenti elementi:

- CU (*Control Unit*) che controlla la memoria e le istruzioni;
- FPU (*Floating Point Unit*) per eseguire calcoli in virgola mobile;
- ALU (*Arithmetic Logic Unit*) per il calcolo delle operazioni matematiche e logiche;
- MMU (*Memory Management Unit*) per la gestione degli indirizzi di memoria;
- Registri (*Register*), due o più locazioni di memoria per il puntamento all'istruzione successiva (IP) e il flag di stato come Overflow, Zero, Interrupt e così via.

Istruzioni

Per eseguire i calcoli e le istruzioni, una CPU deve essere sincronizzata al sistema, per cui ha bisogno di un orologio interno (CLOCK), di linee di Interrupt Request (IRQ) per richieste hardware al sistema e disporre di una linea di ripristino in caso di blocco del sistema o di errore (RESET). La velocità del clock stabilisce il numero di operazioni che il sistema può elaborare. Nelle prime CPU il clock poteva

arrivare a qualche kHz, mentre oggi si arriva tranquillamente oltre i 3 GHz, ovvero con un clock un milione di volte più veloce.

Con il termine “istruzione” si intende il “compito” che una CPU deve eseguire in sincrono con il resto del sistema. Per esempio, se si programma all’interno della memoria l’istruzione `PRINT "A"`, la CPU deve elaborare l’istruzione interpretando il comando `PRINT` in una serie di dati in linguaggio macchina e spedirli alla periferica a cui è destinata l’istruzione, in questo caso il monitor o la stampante eventualmente collegata, e visualizzare a video il carattere “A” o su stamparlo su carta.

L’insieme di istruzioni che una CPU può capire sono codici binari, ovvero stringhe di impulsi elettrici di 0 o 5 volt corrispondenti alle cifre binarie 0 e 1. Quindi, quando si scrive un programma in qualsiasi linguaggio a basso livello o ad alto livello (Assembly, C, C++, Pascal, Basic e così via) le “parole” scritte come istruzioni in linguaggio macchina vengono prelevate dalla memoria e inviate alle unità di elaborazione della CPU, che interpreta il linguaggio macchina producendo l’azione contenuta nell’istruzione. Per esempio, una tipica istruzione in linguaggio macchina potrebbe essere “00000100000011”, ovvero una serie di stati logici. La CPU legge gli stati logici inviati da un periferica o memorizzati in un memoria di massa come stati elettrici. La CPU compie l’istruzione, seguendo l’impostazione della sua architettura e alla velocità imposta dal suo clock (tempo di elaborazione).

Se una CPU dei primordi, con un clock di pochi kHz, era in grado di eseguire qualche decina di istruzioni al secondo, si è ben presto passati a CPU in grado di compiere qualche centinaia di MIPS (*Million Instructions Per Second*). Oggi si parla di CPU da 20 peta FLOPS (*FLoating point Operations Per Second*) ovvero $20 \times 10^{15} = 20$ milioni di miliardi di operazioni in virgola mobile al secondo.

RISC

Come già accennato, l'architettura RISC sta per *Reduced Instruction Set Computer* che, già dal nome stesso, è orientata allo sviluppo di programmi semplici. La semplicità permette ai microprocessori di eseguire il set di istruzioni più velocemente rispetto a un'architettura CISC (*Complex Instruction Set Computer*), che invece si basa su un set di istruzioni complesse.

La Figura 1.42 illustra alcuni processori RISC molto diffusi:

- AVR (Atmel);
- ARM (Raspberry PI);
- PIC (PICAXE);
- SPARC (Sun Microsystems);
- MIPS (Toshiba).

Una menzione particolare merita il processore ARM, usato nel progetto Raspberry PI Foundation, una fondazione benefica per promuovere lo studio dell'informatica e di argomenti correlati.

Montato su una motherboard grande quanto un biglietto da visita (85 × 53 mm circa), un processore ARM di questo tipo è in grado di gestire un sistema operativo basato su kernel Linux Debian o RISC OS.

In pratica, Raspberry PI è un computer completo, dotato di interfaccia USB, interfaccia video e audio HDMI, 256 o 512 MB di memoria, porta Ethernet e porta per scheda SD.

Presentato nel 2011 come il più piccolo computer del mondo, è stato immesso nel mercato a febbraio 2012 a soli 35 dollari.

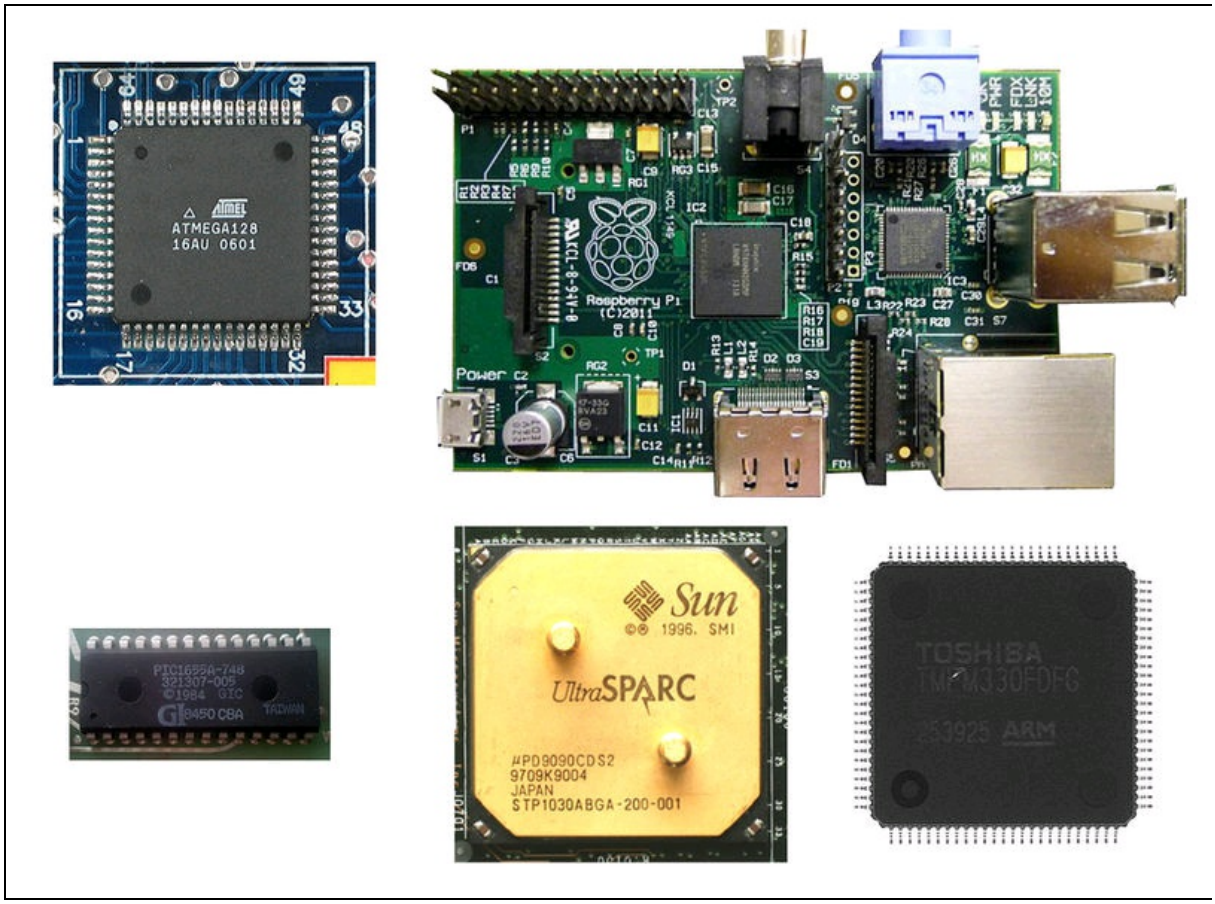


Figura 1.42 Alcuni processori RISC: Atmel, Raspberry PI, PICAXE, Sun Microsystems, Toshiba.

Motori elettrici

I motori elettrici di piccola e media potenza possono essere facilmente impiegati nella costruzione di aeromodelli, droni, piccoli rover e bracci robotici.

I motori elettrici possono essere principalmente di tre tipi:

- motori DC;
- servomotori;
- motori passo-passo (o stepper).

Motore DC

Il tipico motore DC funziona solamente in corrente continua. È composto principalmente da due parti:

- un avvolgimento di spire di filo di rame smaltato, chiamato *rotore*;
- un magnete che crea un campo magnetico fisso, chiamato *statore*.

Il rotore, posto sotto tensione, crea un campo elettromagnetico al passaggio di corrente. Essendo il rotore immerso in un campo magnetico, quando viene applicata una tensione inizia a girare per induzione elettromagnetica. Nella sua forma più semplice, il campo elettromagnetico del rotore tende a sfuggire dal campo magnetico dello statore, ovvero il polo positivo del campo elettromagnetico del rotore viene attratto dal polo negativo dello statore, producendo una rotazione continua.

Nei motori elettrici dotati di più poli (cioè con più avvolgimenti), sono presenti due piccole spazzole che forniscono la corrente necessaria agli avvolgimenti del rotore in modo sequenziale durante la rotazione (Figura 1.43a). In questo modo, avviene uno scambio di polarità del campo elettromagnetico. Il campo magnetico dello statore e quello del

rotore non sono mai allineati perfettamente, rendendo così la rotazione più fluida.

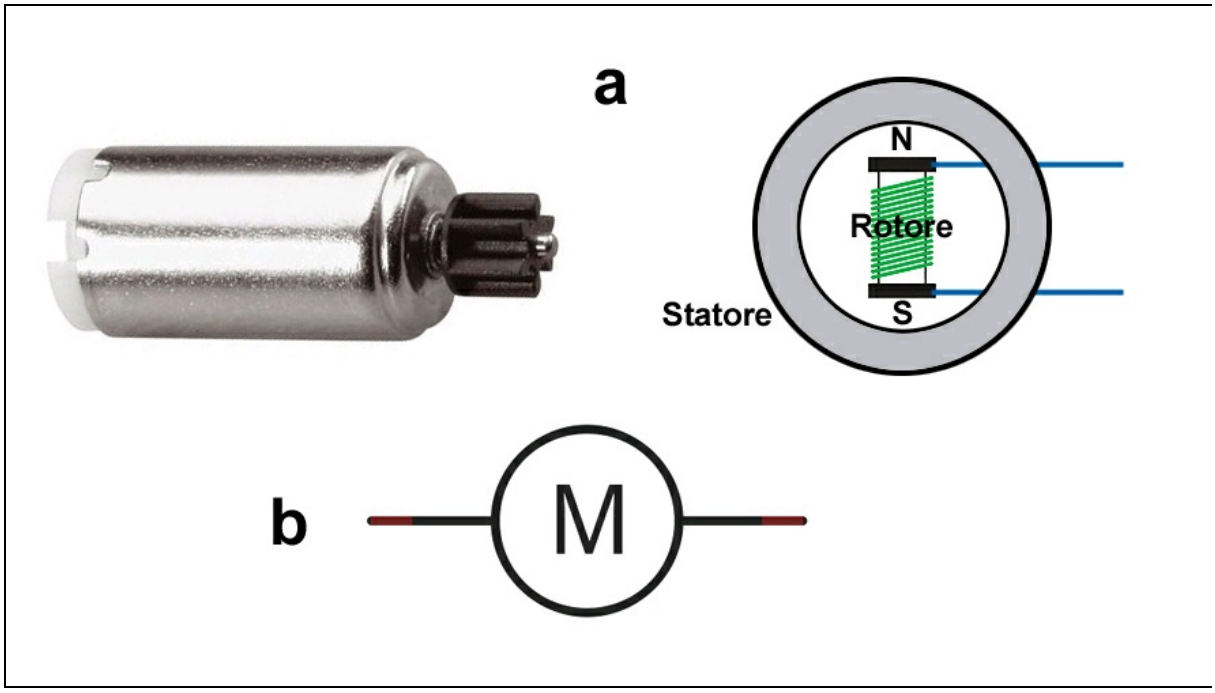


Figura 1.43 Motore elettrico DC dotato di spazzole e principio di funzionamento (a). Simbolo elettrico usato negli schemi elettrici (b).

Motore DC senza spazzole

Il motore DC senza spazzole (in inglese, *brushless*) è un motore in corrente continua composto da magneti permanenti e da avvolgimenti alimentati sequenzialmente da un circuito elettronico e non da un sistema fisso di alimentazione, come nel caso delle spazzole striscianti sul rotore.

Il vantaggio è una minore resistenza meccanica, assenza di scintille provenienti dai contatti striscianti, una maggiore durata e una maggiore precisione nel posizionamento meccanico (Figura 1.44).

I magneti permanenti sono posizionati sul rotore, consentendo un controllo preciso sia della velocità sia dell’accelerazione. Sono motori

ideali per il controllo della rotazione nei lettori CD/DVD e simili, ma si sono rivelati ottimi anche nell'aeromodellismo e nell'automodellismo, quando serve più potenza e stabilità. Sono infatti usati nella costruzioni di droni.

Il motore brushless necessita di un controller elettronico per funzionare e per questo motivo risulta essere più costoso del motore con spazzole.

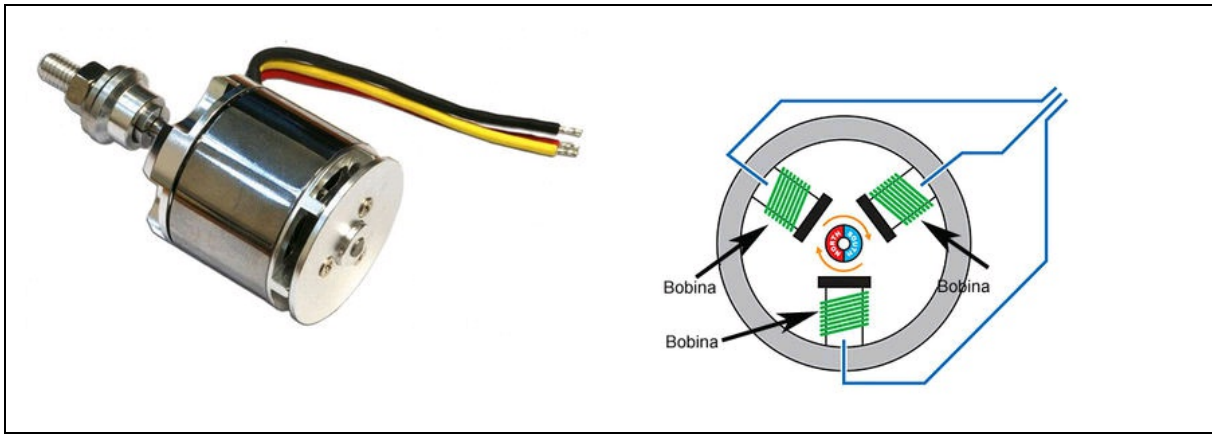


Figura 1.44 Motore elettrico DC senza spazzole e principio di funzionamento.

Motore passo-passo

Il motore passo-passo (in inglese *step o stepper motor*) è un motore brushless sincrono in corrente continua, la cui posizione del rotore può essere controllata a passi (Figura 1.45).

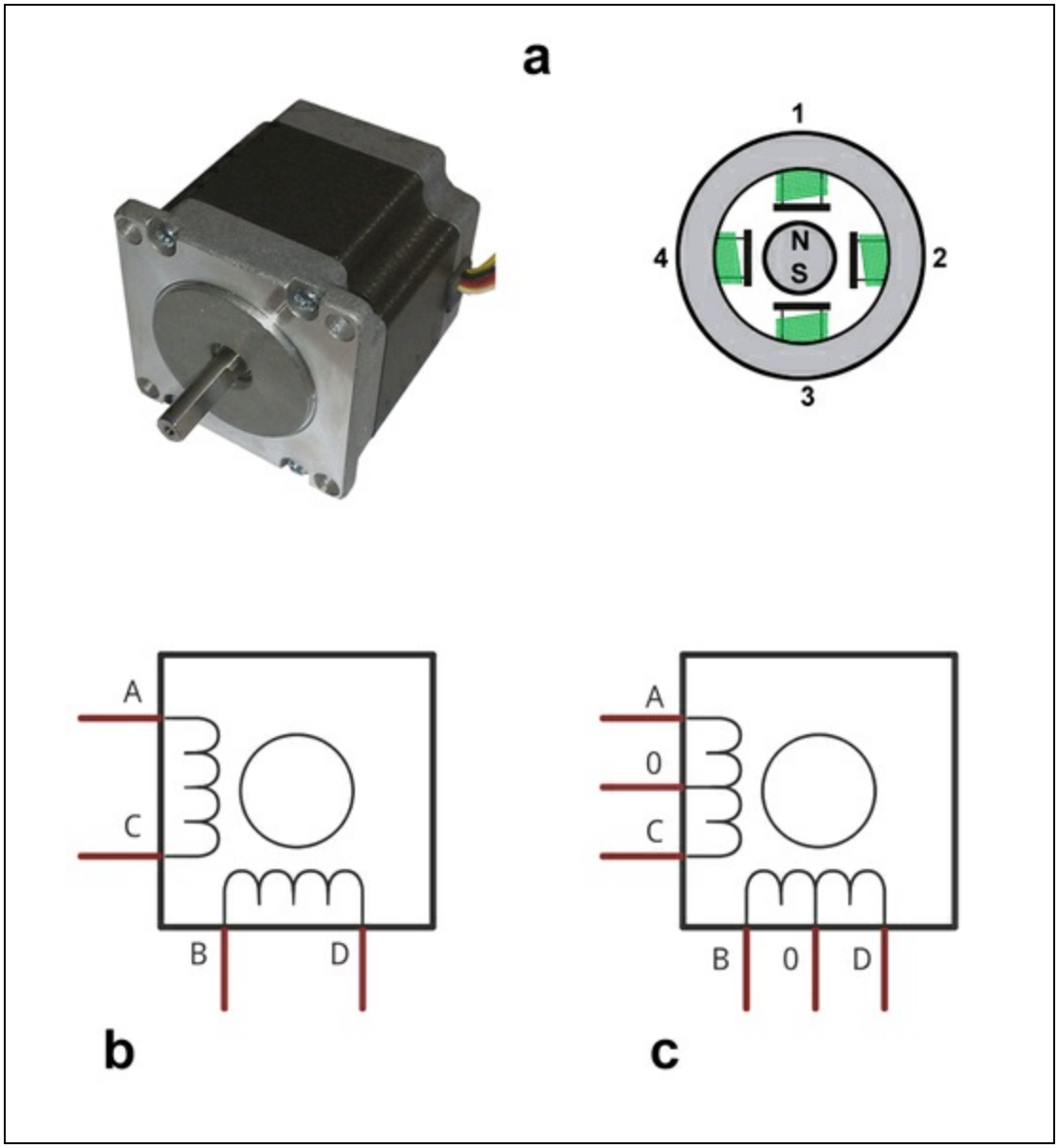


Figura 1.45 Motore elettrico passo-passo tipo NEMA 17 e principio di funzionamento (a). Simbolo elettrico di un motore passo-passo bipolare (b) e di uno unipolare (c).

A differenza del motore tradizionale, il motore passo-passo è utilizzabile nelle applicazioni in cui serve controllare la rotazione o fermare l'albero in una determinata posizione. Pertanto sono ideali nella robotica, per costruire bracci o articolazioni con controllo da microprocessore, e nella costruzione di stampanti 3D.

Per le applicazioni robotiche e per il controllo preciso del movimento, come nelle stampanti 3D, è consigliabile usare un motore passo-passo per sfruttare la coppia meccanica e per far compiere all'albero rotazioni angolari molto precise. Altro fattore importante è che i motori passo-passo hanno un momento inerziale molto basso e sono molto stabili quando si blocca il rotore in una posizione.

I motori passo-passo possono essere principalmente di due tipi.

- *Bipolari*: sono dotati di due avvolgimenti separati.
- *Unipolari*: sono dotati di due avvolgimenti con un polo comune.

I bipolari si distinguono dagli unipolari perché hanno quattro fili corrispondenti ai due avvolgimenti separati. Gli unipolari hanno cinque o sei fili a seconda che le coppie centrali siano in comune con un solo filo o con due fili.

I motori passo-passo sono facilmente reperibili in commercio oppure smontando vecchie stampanti 2D e scanner. Per il riconoscimento degli avvolgimenti, basta usare un multimetro impostato su 200 Ω fondo scala e trovare l'avvolgimento, provando i collegamenti uno alla volta.

In commercio esistono moltissimi tipi di motori passo-passo, le cui caratteristiche principali possono essere simili alle seguenti.

- Coppia statica bipolare: 0,28 Nm (la coppia è espressa in Newton-metro).
- Corrente di fase bipolare: 0,7 Arms (corrente in ampere RMS).
- Inerzia del rotore: 34g cm^2 (espressa in grammi su cm quadrato).
- Lunghezza: 34 mm (misura fisica del motore).
- Peso: 240 g (peso del motore).
- Flangia NEMA 17: 42,3 × 42,3 mm (indica le dimensioni della flangia del motore).
- Numero di fili: 4.

- Angolo di passo intero: 1,8º (lo spostamento che il rotore compie quando esegue un passo intero).

Servomotore

I servomotori sono particolari motori passo-passo dotati di ingranaggi e servomeccanismi controllati da un microprocessore interno. Sono studiati per gli utilizzi di controllo più disparati.

Esistono centinaia di tipi di servomotori, per cui ci limiteremo a descriverne uno molto adattabile a progetti di vario tipo. Si tratta del servomotore HS-5035HD, dalle dimensioni davvero contenute e leggerissimo, costruito da Hitec Robotics (Figura 1.46).

Riportiamo di seguito le caratteristiche principali.

- Servo ultra nano leggerissimo, solo 4,5 grammi.
- Digitale programmabile.
- 0,8 kg cm di coppia a 4,8 V.
- 0,10 sec/60º di velocità.
- Scatola ingranaggi in KARBONITE ultra resistente.
- Potenziometro LONG LIFE.
- Dimensioni 18,6 × 7,6 × 15,5 mm.

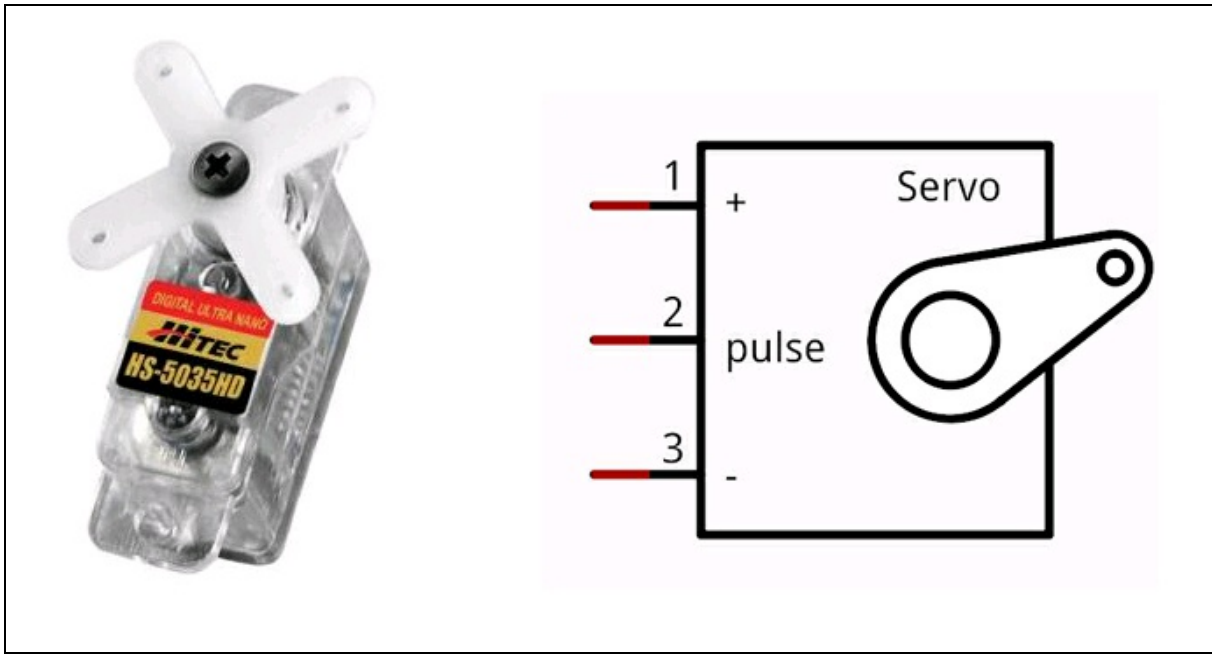


Figura 1.46 Servomotore HS-5035HD costruito da Hitec Robotics e il simbolo usato negli schemi elettrici.

Funzioni programmabili:

- protezione dal sovraccarico;
- senso di rotazione;
- velocità di rotazione;
- fine corsa;
- dentraggio;
- failsafe on/off;
- posizione failsafe.

Esistono servomotori normali, solitamente con rotazione da 0^o a 180^o, oppure servomotori *full rotation*, ovvero con rotazione continua a 360^o, molto più simili in questo ai motori passo-passo.

Un tipico utilizzo dei servomotori è il controllo *pan&tilt* di una webcam in un progetto di videosorveglianza.

Sensori

Da quando esiste l’elettronica, l’uomo ha sempre cercato di fornire i suoi robot di organi sensoriali, facendoli parlare e interagire agli stimoli di luce, suono, calore e movimento, per renderli un po’ più umani.

Lo scopo dei progetti di questo libro è quello di dotare alcuni circuiti di sensori per l’allerta o il monitoraggio, così come il controllo degli accessi e così via.

Qui di seguito diamo uno sguardo ad alcune tipologie di sensori che possono venire facilmente utilizzati nei progetti di questo libro.

- Sensore ottico.
- Sensore acustico.
- Sensore di movimento.
- Sensore di temperatura.
- Sensore magnetico.
- Sensore GPS.
- Sensore tattile (impronta digitale).
- Sensore RFID/NFC.

Sensore ottico

In questa famiglia di sensori vengono annoverati tutti i fotorivelatori o dispositivi optoelettronici sensibili alle emissioni di luce. Questi sensori consentono il controllo di circuiti elettrici remoti tramite il cambiamento del flusso di luce diretta o indiretta. I più comuni fotorivelatori sono i seguenti:

- fotocellula o fotoresistore;
- fotodiodo e modulo IR;
- fototransistor.

Fotocellula

Il componente passivo più usato per il controllo a distanza per l’apertura di porte e cancelli è sicuramente la fotocellula o fotoresistore, di cui abbiamo già parlato all’inizio del capitolo. La sua particolare lentezza di risposta agli stimoli luminosi ne fa un componente ideale per progetti in cui non serve una repentina reazione alla luce.

Sfruttando questa particolarità, è possibile controllare gradualmente la tensione in uscita per graduare la velocità di un motore elettrico o per diminuire e aumentare l’intensità di un singolo LED o anche di un intero sistema di illuminazione.

Il fotoresistore trova spazio anche in applicazioni musicali, come regolatore dell’intensità e dell’altezza di un suono, rilevando la posizione delle mani in aria, un po’ come faceva Leon Thermen e il suo Theremin agli inizi del secolo scorso.

Con la stessa facilità si possono progettare dei circuiti con fotocellula per il conteggio di persone o di oggetti oppure per l’automazione di porte e finestre, così come l’inseguimento solare. I fotoresistori vengono impiegati infatti in circuiti crepuscolari per accendere o spegnere impianti di illuminazione pubblica e privata, in base alla quantità di luce solare.

Nella Figura 1.47 è illustrato un semplice circuito di controllo di un relè tramite un fotoresistore. I componenti del circuito sono i seguenti:

- transistor NPN (tipo BC108, 2N2222 o simile a bassa potenza);
- diodo 1N4001 o equivalente;
- fotocellula da 1 MΩ non illuminata;
- potenziometro 47 KΩ;
- resistore 2,2 KΩ;
- relè con bobina a 9 volt in corrente continua con contatto normalmente aperto.

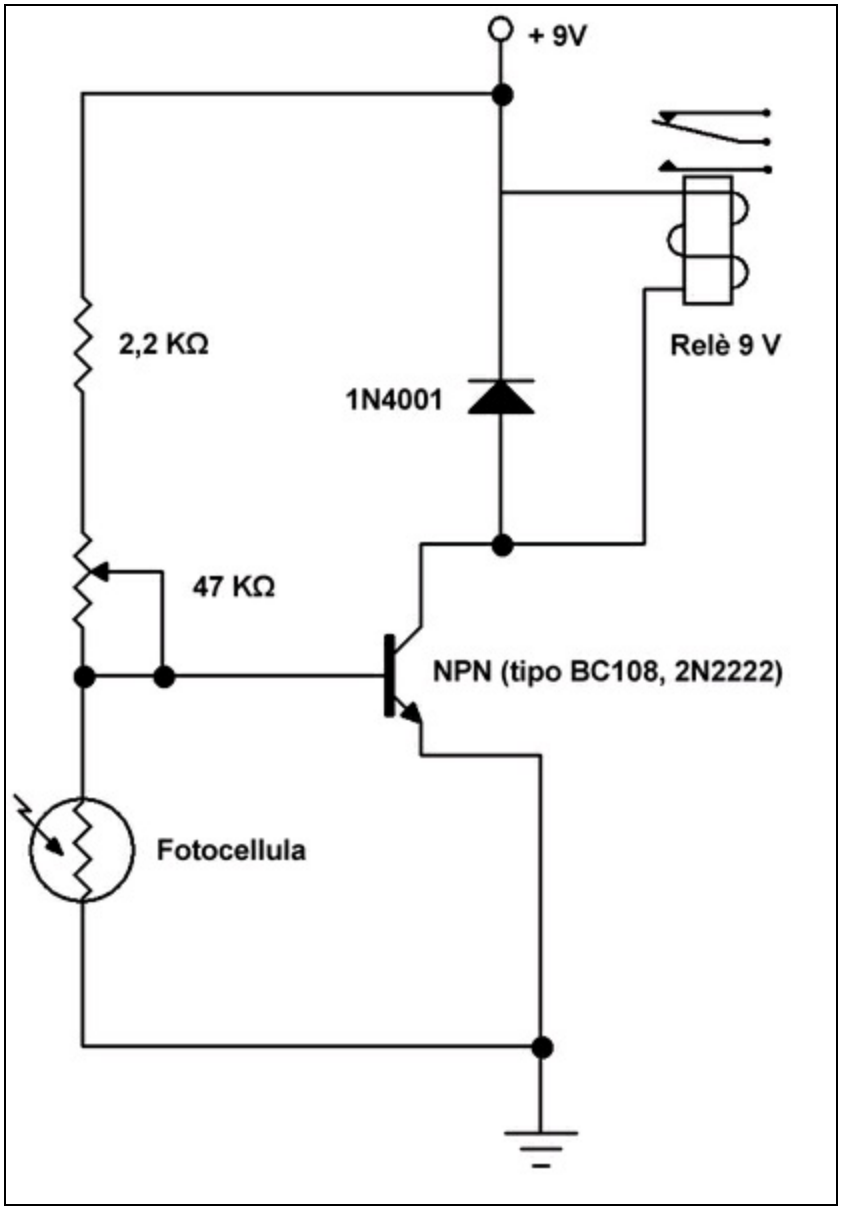


Figura 1.47 Circuito di controllo di un relè con un fotoresistore.

Fotodiodo e modulo IR

Il diodo a semiconduttore è costituito da una giunzione di silicio drogato PN e, in taluni casi, è in grado di emettere luce (LED).

Oltre a poter emettere luce, l'altra peculiarità di questa giunzione è di essere sensibile alla luce. Perciò, illuminando la giunzione con un fascio

luminoso, si ottiene una tensione ai terminali del diodo. Per esempio, se si prova a collegare un diodo tipo 1N4148 o similare (con involucro in vetro trasparente) e lo illumina, si misurerà sul multimetro una debole tensione di qualche millivolt (effetto Mims). È la prova di come i fotoni si possono trasformare in elettroni.

Il fotodiodo è un particolare diodo in cui viene esaltata questa sensibilità alla luce, anche grazie a una lente che fa confluire in modo perpendicolare la luce incidente.

Quando un fotodiodo è polarizzato direttamente, ovvero la corrente positiva scorre fra anodo e catodo, si comporta come un normale diodo. Se il fotodiodo viene polarizzato in maniera inversa, cioè con la tensione più alta sul catodo (giunzione N), il diodo non conduce, se non riceve luce. Per cui, il fotodiodo *illuminato* oppone una resistenza bassissima al passaggio di corrente (pochi Ω); *senza illuminazione* oppone una resistenza elevatissima (qualche MΩ).

In pratica, i fotodiodi vanno inseriti nel circuito con polarizzazione inversa comportandosi così come un fotoresistore, la cui resistenza dipende dalla quantità di luce ricevuta, ma con risposta ai cambiamenti di luce molto più veloci.

Al contrario del fotodiodo ricevitore, un fotodiodo trasmettitore si comporta come un LED tarato sulle frequenze dell'infrarosso.

Nella Figura 1.48a sono illustrati due fotodiodi solitamente impiegati, rispettivamente, per la ricezione e la trasmissione IR (*Infrared Radiation*).

Esistono moduli IR con tre piedini (per esempio quelli della famiglia TSOP17XX) per il controllo di apparecchi tramite telecomando. In questi ricevitori, assieme al fotodiodo, sono integrati anche i circuiti di controllo della frequenza, un filtro passa-banda e un demodulatore.

Nella Figura 1.48b è illustrato un circuito standard di controllo che un utilizza un fotodiodo ricevitore a infrarossi modello TSOP1740.

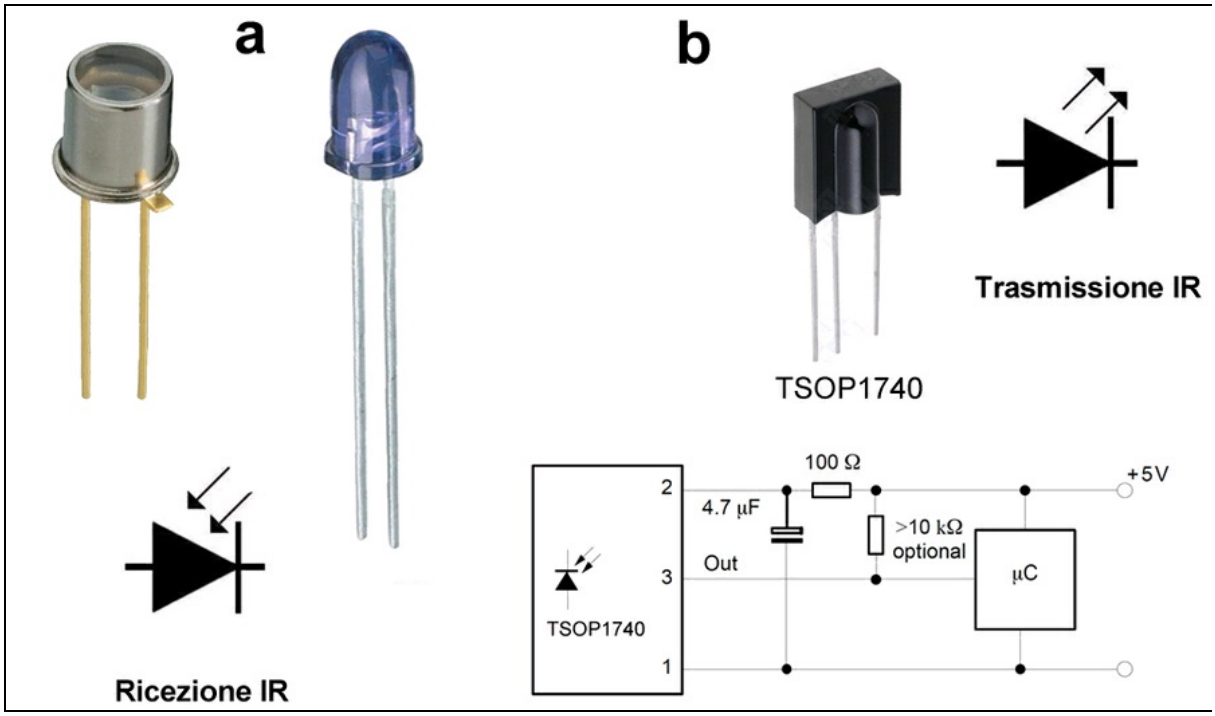


Figura 1.48 Fotodiodi solitamente impiegati per la ricezione e la trasmissione IR e rispettivi simboli usati negli schemi elettrici (a). Un modulo TSOP1740 e un semplice circuito di controllo tratto dal datasheet ufficiale del produttore.

Fototransistor

In modo simile al fotodiodo, il fototransistor è un transistor BJT a due giunzioni, con contenitore dotato di lente che fa confluire la luce sulla giunzione collettore-base.

In pratica, il principio di funzionamento è quello di un transistor normale, solo che la corrente sulla base (che modula la corrente tra collettore ed emettitore) viene determinata dalla quantità del flusso luminoso incidente.

Il fototransistor, rispetto al fotodiodo, è più sensibile alle radiazioni luminose e può avere una corrente di uscita molto alta (dell'ordine dei mA), ma per contro ha una risposta più lenta.

Nella Figura 1.49 è illustrato un semplice circuito di controllo con un fototransistor modello BPW77.

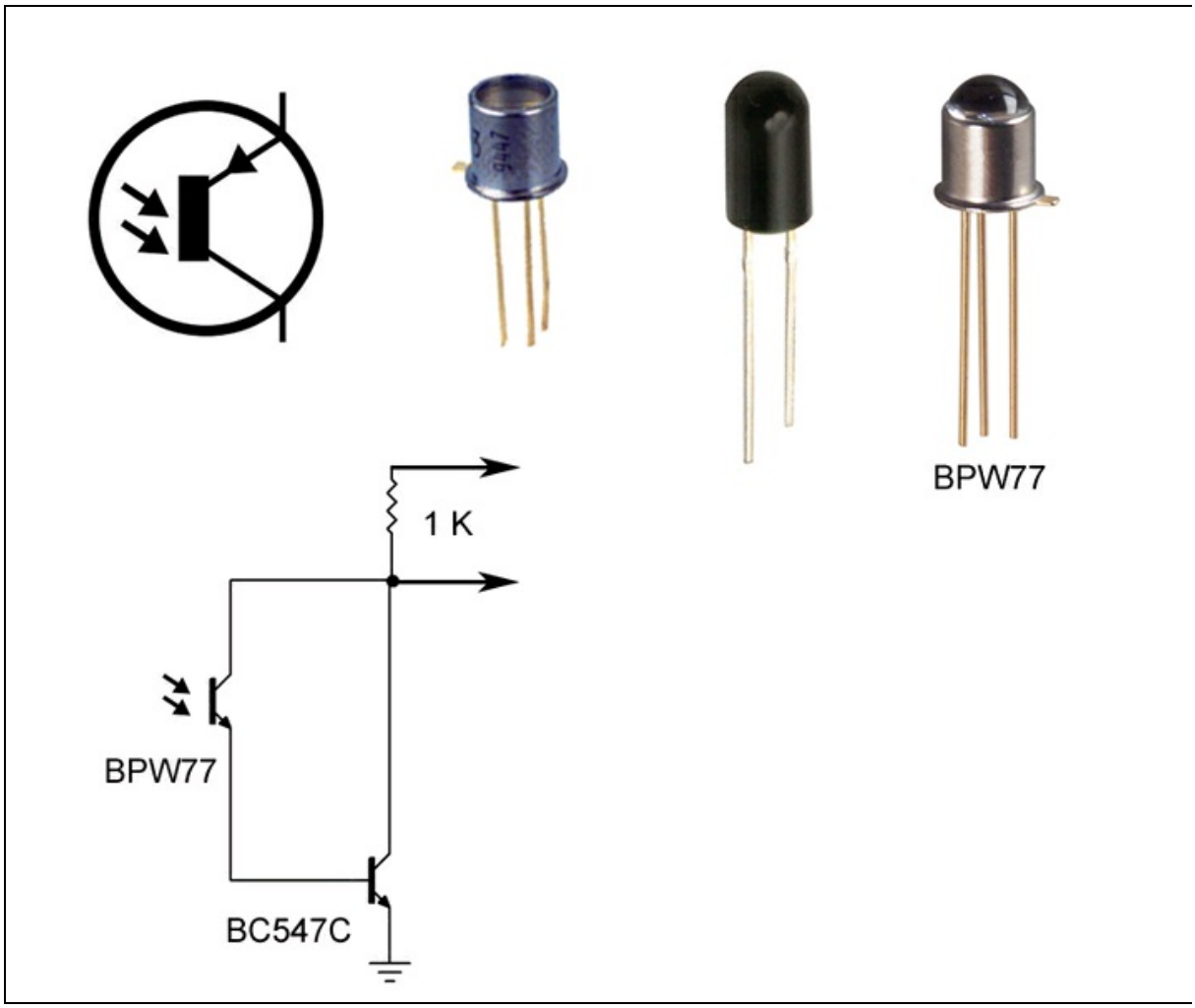


Figura 1.49 Uno dei simboli elettrici usati per identificare un fototransistor e alcuni tipi \di fototransistor. Sotto, un semplice circuito di controllo con BPW77.

Modulo PIR (infrarosso passivo)

Quando si vuole monitorare l'ambiente tramite il rilevamento di emissioni sulla gamma dell'infrarosso prodotto da una persona o da un corpo caldo, è necessario usare sensori specificamente calibrati. Di tanti prodotti disponibili in commercio, di solito viene usato un sensore a infrarossi passivo (PIR, acronimo di *Passive InfraRed*). Si tratta di un sensore elettronico che misura i raggi infrarossi (IR) irradiati dagli

oggetti nel suo campo di vista e viene usato come rilevatore di presenza umana o di movimento.

Il termine “passivo” si riferisce al fatto che i PIR non emettono energia, ma lavorano esclusivamente rilevando l’energia sprigionata dai corpi caldi.

Di solito il modulo PIR è dotato di due trimmer per la regolazione della sensibilità (distanza) e del ritardo di intervento. Il fotodiodo è ricoperto da una calotta di plastica, che agisce come lente di Fresnel per distribuire la sensibilità del fotodiodo su 180 gradi sferici. Quando il sensore rileva un corpo caldo, il modulo invia un impulso che può venire rilevato da un controller.

Figura 1.50 Sensore PIR.

Sensore acustico

I sensori acustici sono trasduttori elettro-meccanici in grado di convertire segnali acustici in segnali elettrici variabili. Un po' come i microfoni, sono sensibili alle variazioni di pressione sonora, ma senza avere la qualità dei microfoni progettati per la ripresa audio. D'altra parte, anche un qualsiasi microfono può essere considerato un sensore

acustico e potrebbe essere utilizzato per la funzione di rilevamento delle onde sonore.

Sensore piezoelettrico

Dal nome stesso si intuisce che il sensore piezoelettrico basa il suo funzionamento sul principio fisico della piezoelettricità. Il termine greco *piezein* significa *premere*, per cui si dicono piezoelettrici quei materiali che producono una tensione se sottoposti a una pressione. È il caso di alcuni cristalli che quando vengono deformati meccanicamente producono una debolissima tensione.

Di solito i sensori piezoelettrici sono molto resistenti alle pressioni. Tant'è che vengono chiamati anche sensori di pressione e vengono posti direttamente sul corpo dell'oggetto da monitorare. Anche a forti pressioni il segnale in uscita è dell'ordine di qualche millivolt (Figura 1.51) e va opportunamente amplificato.

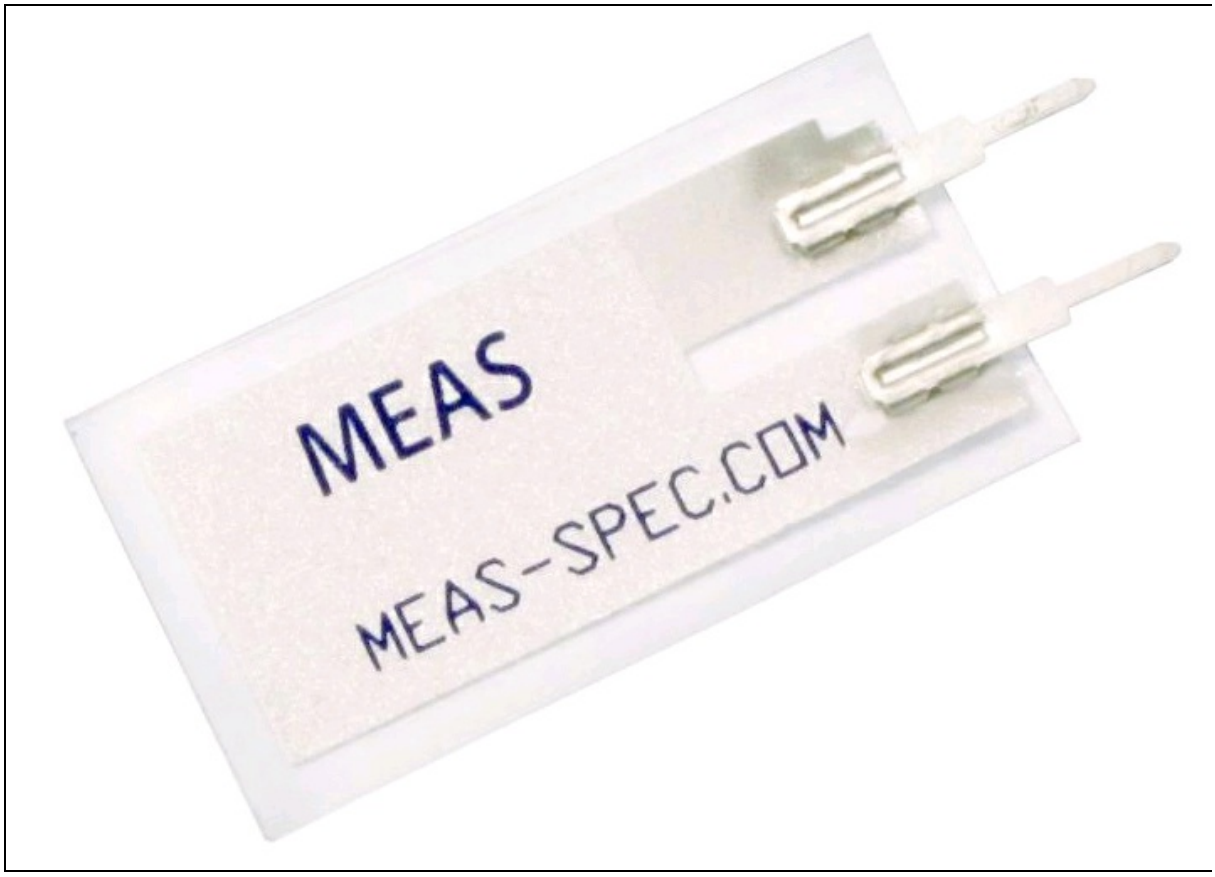


Figura 1.51 Un tipico sensore piezoelettrico MEAS.

Sensore a ultrasuoni

I sensori a ultrasuoni sono sensori acustici che funzionano sullo stesso principio dell'ecoscandaglio o sonar e sono in grado di rilevare un impulso sonoro ultrasonico di ritorno dopo che questo è stato riflesso da un oggetto remoto. I sensori ultrasonici lavorano su frequenze audio da 200 a 400 kHz e normalmente riescono a rilevare un oggetto in movimento o un ostacolo distante all'incirca 10 metri.

Di tutti i numerosi modelli di sensori ultrasonici di solito si usano il modello economico HC-SR04 o il più sofisticato URM37, prodotto da DFrobot (www.dfrobot.com) (Figura 1.52).

Il sensore a ultrasuoni è montato su un circuito stampato dotato di una striscia di pin che si collega facilmente alle porte di un controller, per esempio Arduino o Raspberry Pi. Per questi tipi di controller esistono librerie pronte all'uso, che ne facilitano enormemente l'uso.

Figura 1.52 Il sensore a ultrasuoni URM37.

Sensore di movimento

Anche se i sensori a infrarossi e a ultrasuoni possono considerarsi in qualche modo sensori di movimento, vogliamo solo aggiungere tre tipologie diffuse come gli accelerometri, i giroscopi e i sensori di tilt.

Accelerometro

In tempi recenti questo tipo di sensore ha avuto un enorme successo nel settore ludico e dell'intrattenimento. Ormai presente anche negli smartphone e nei tablet, l'accelerometro è un dispositivo in grado di rilevare l'accelerazione a cui è sottoposto. Viene usato moltissimo anche nei droni.

Il principio di funzionamento si basa sulla rilevazione dell'inerzia della massa alla quale viene applicata un'accelerazione. La massa è sospesa a un elemento elastico, mentre un sensore ne rileva gli spostamenti. La massa sottoposta a un'accelerazione si sposta dalla posizione di riposo in modo proporzionale all'accelerazione e il sensore trasforma lo spostamento in un segnale elettrico. Il sensore può essere di tipo piezoelettrico, induttivo, capacitivo, meccanico e un misto di questo tecnologie.

Il segnale elettrico analogico proveniente dal sensore viene campionato e misurato da un software, che così può interpretare l'informazione elettrica dell'accelerazione per mettere in atto le corrispondenti azioni programmate. In altre parole, le accelerazioni possono essere usare per i più svariati usi: per il gioco, lo sport, la musica e via dicendo.

Fra i numerosi prodotti in commercio, abbiamo individuato un accelerometro a tre assi ADXL335, prodotto da Analog Devices, già montato su un modulo breakout facile da reperire e dal costo accessibile.

Il dispositivo ADXL335 è un sensore analogico di tipo meccanico, in grado di misurare l'accelerazione su tre assi con rilevamento della gravità fino a +/- 3g. La scheda viene fornita completamente assemblata e testata con componenti esterni installati da Sparkfun (Figura 1.53). I condensatori da 0,1 µF inclusi impostano la larghezza di banda di ciascun asse a 50 Hz.

Si può collegare facilmente a tre ingressi analogici di un controller come Arduino o Raspberry Pi e gestire l'accelerometro tramite una libreria pronta all'uso.

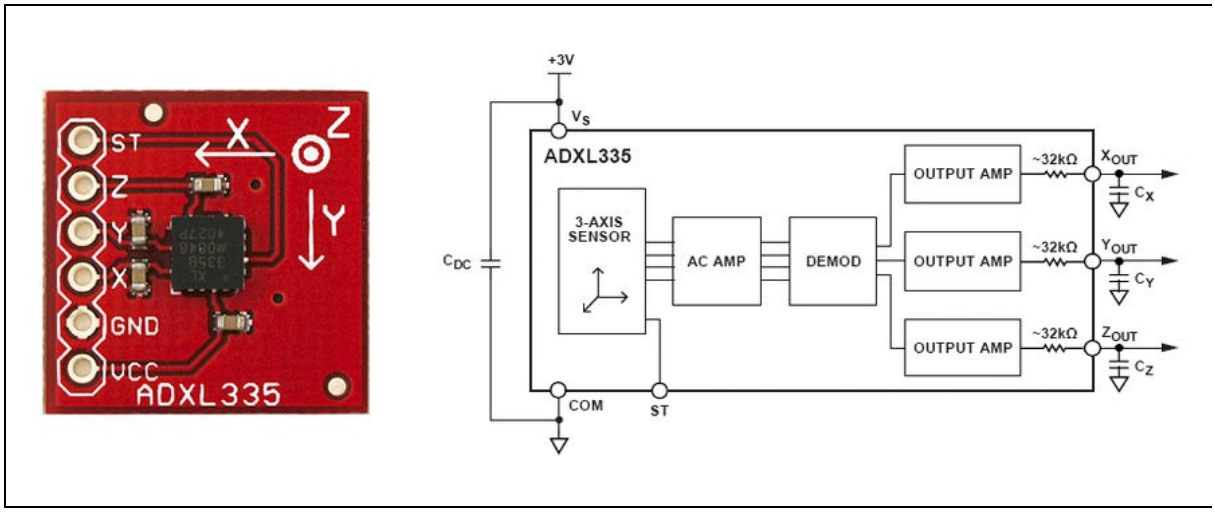


Figura 1.53 Accelerometro a tre assi ADXL335 della Analog Devices e lo schema di esempio.

Giroscopio

Uno dei più diffusi ed economici giroscopi è il sensore MPU-6050 della InvenSense, che contiene un accelerometro MEMS e un giroscopio MEMS in un singolo chip (Figura 1.54). È molto preciso, in quanto contiene un hardware a conversione digitale 16 bit analogico per ciascun canale. Pertanto cattura allo stesso tempo i canali x, y e z. Il sensore utilizza la il bus I^2C per interfacciarsi facilmente con Arduino o Raspberry Pi. Il giroscopio è ormai parte integrante della circuitazione di bordo di tutti i droni.

NOTA

La sigla MEMS sta per Micro Electro-Mechanical Systems, una nanotecnologia di ultra-miniaturizzazione a livello microscopico di circuiti elettro-meccanici.

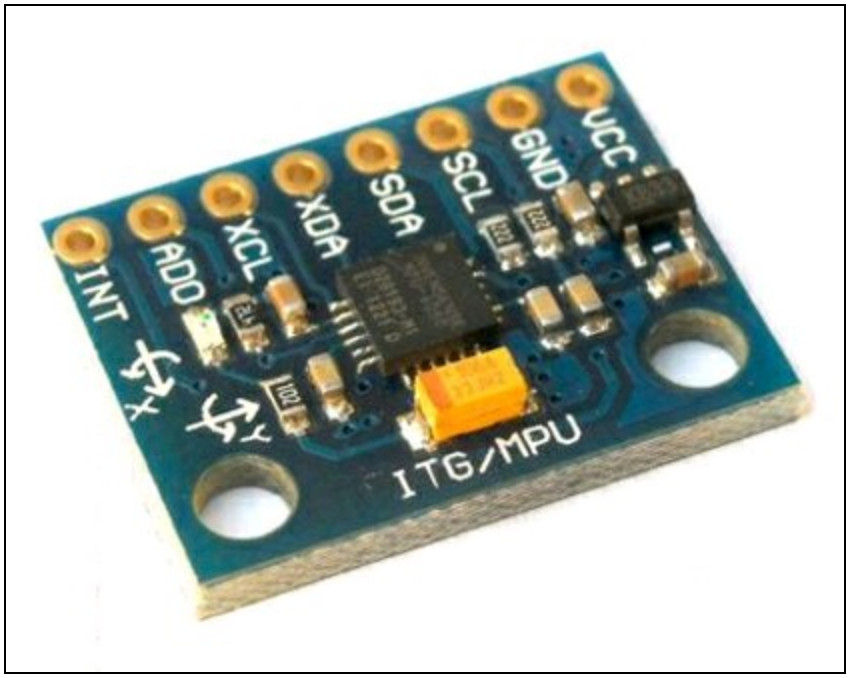


Figura 1.54 Giroscopio basato su MPU-6050.

Sensore di tilt

Molto più semplice da usare è il sensore di *tilt* (inclinazione), il quale riesce a percepire un'inclinazione di 180^{o} sull'asse orizzontale. Il sensore è disponibile sia come componente singolo che su modulo, come quello proposto da Tinkerkit presso lo store del sito Arduino (Figura 1.55).

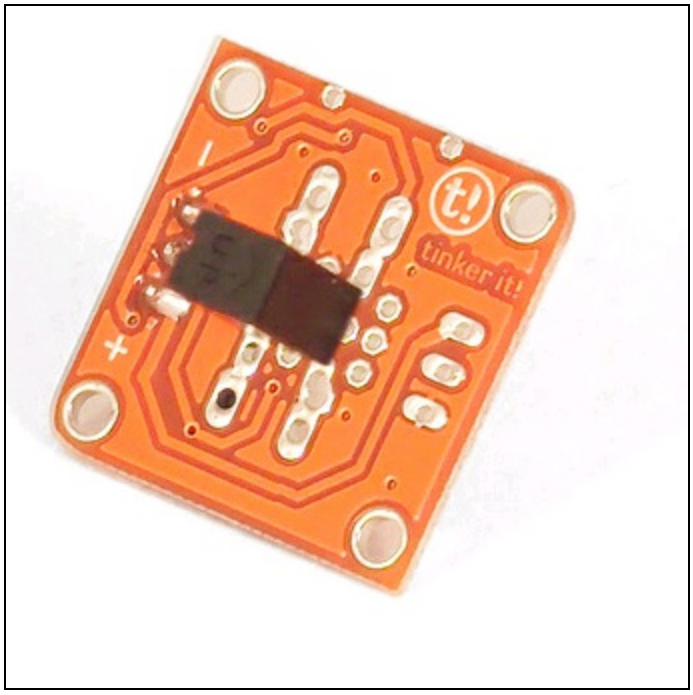


Figura 1.55 Modulo Tilt Sensor prodotto da Tinkerkit.

Il sensore dispone all'interno di due contatti che vengono chiusi da una piccola sfera di metallo quando viene piegato su un lato o capovolto. Quando il sensore si trova nella posizione eretta la sfera chiude i due contatti del circuito. Quando il sensore è inclinato di 90°, la sfera apre il circuito. Nella posizione eretta il modulo restituisce 5 V mentre restituisce 0 V se è stato capovolto.

Sensore di temperatura

Oltre ai termistori, esistono anche dei sensori di temperatura specificamente tarati su livelli di tensioni precisi.

Per esempio, il diffusissimo LM35 è un sensore di temperatura dotato di tre piedini e disponibile in due package TO92 e TO220 (Figura 1.56a).

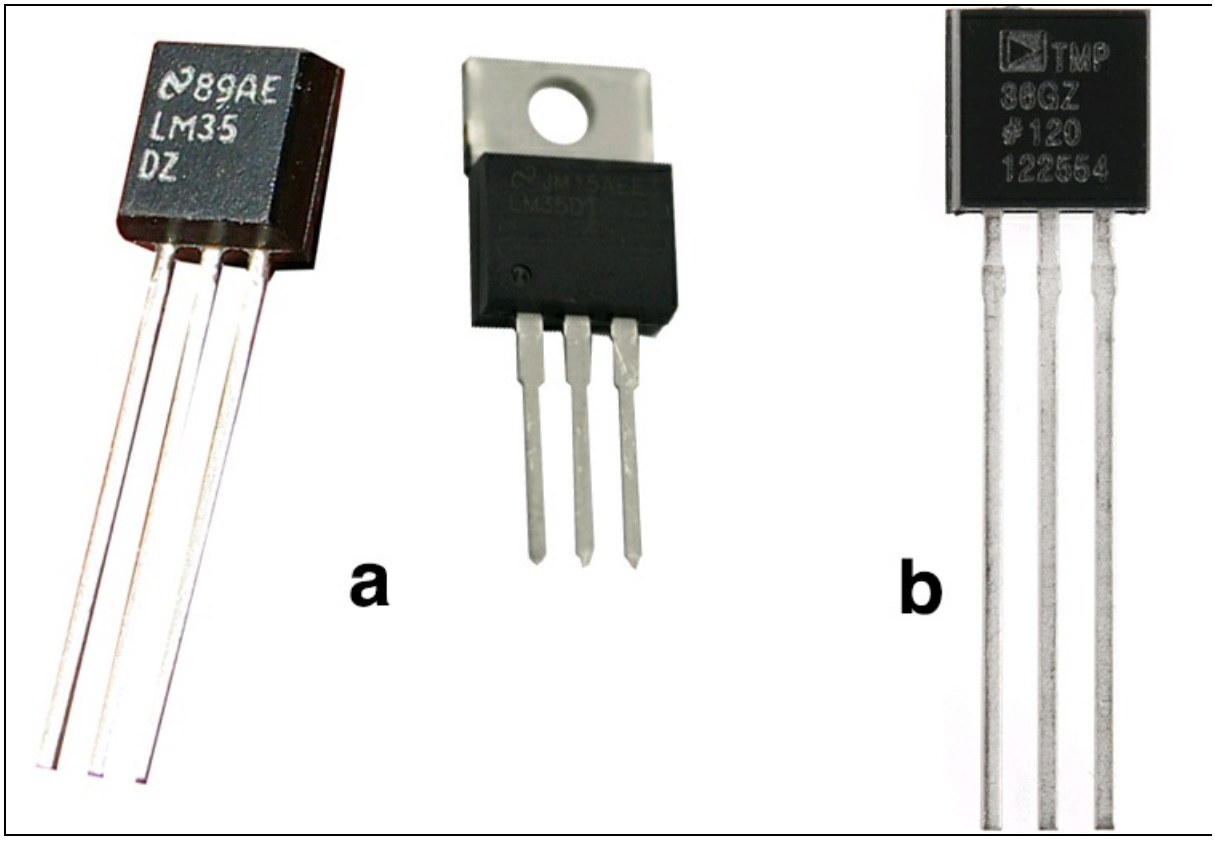


Figura 1.56 Sensore di temperatura LM35 (a) e TMP36 (b).

I tre terminali sono uno per l'alimentazione, uno per la massa e uno per l'uscita della tensione. L'uscita della tensione analogica è proporzionale alla temperatura rilevata, che è pari a 10 mV per ogni grado centigrado. Il sensore è già calibrato in gradi Celsius e le sue caratteristiche principali sono le seguenti.

- Precisione: ±0.5°C.
- Sensibilità: 10 mV/°C.
- Temperatura massima: 150 °C.
- Temperatura minima: -55 °C.
- Tensione tipica di funzionamento: 4-30 V.
- Tipo di uscita: analogica.

Può essere collegato direttamente alla porta analogica di un microprocessore. In alternativa, si può usare il TMP36, un sensore di temperatura analogico simile, fornito con package TO92 (Figura 1.56b). Fornisce un'uscita analogica lineare direttamente proporzionale alla temperatura espressa in °C. Non è richiesta la calibrazione per avere l'accuratezza tipica di ±1°C a 25°C e ±2°C sulla scala di -40 °C +125°C. È molto semplice da utilizzare come l'LM35.

Sensore magnetico

I sensori magnetici si basano essenzialmente sull'effetto Hall, dal nome del fisico Edwin Hall che lo scoprì nel 1879. Si tratta di un fenomeno elettromagnetico in base al quale si crea una differenza di potenziale (potenziale di Hall) sulle facce opposte di due conduttori sotto tensione, quando viene avvicinato perpendicolarmente un flusso magnetico.

Un sensore a effetto Hall fa sì che la tensione restituita dipenda dai campi magnetici da lui rilevati. Può essere utilizzato per rilevare la distanza da un magnete vicino o per rilevare i campi magnetici prodotti da un filo o una bobina quando passa una corrente.

Nella Figura 1.57 si può vedere un sensore di Hall proposto da Tinkerkit presso lo store del sito di arduino.cc. Può essere collegato direttamente a una porta digitale di Arduino o di Raspberry Pi.

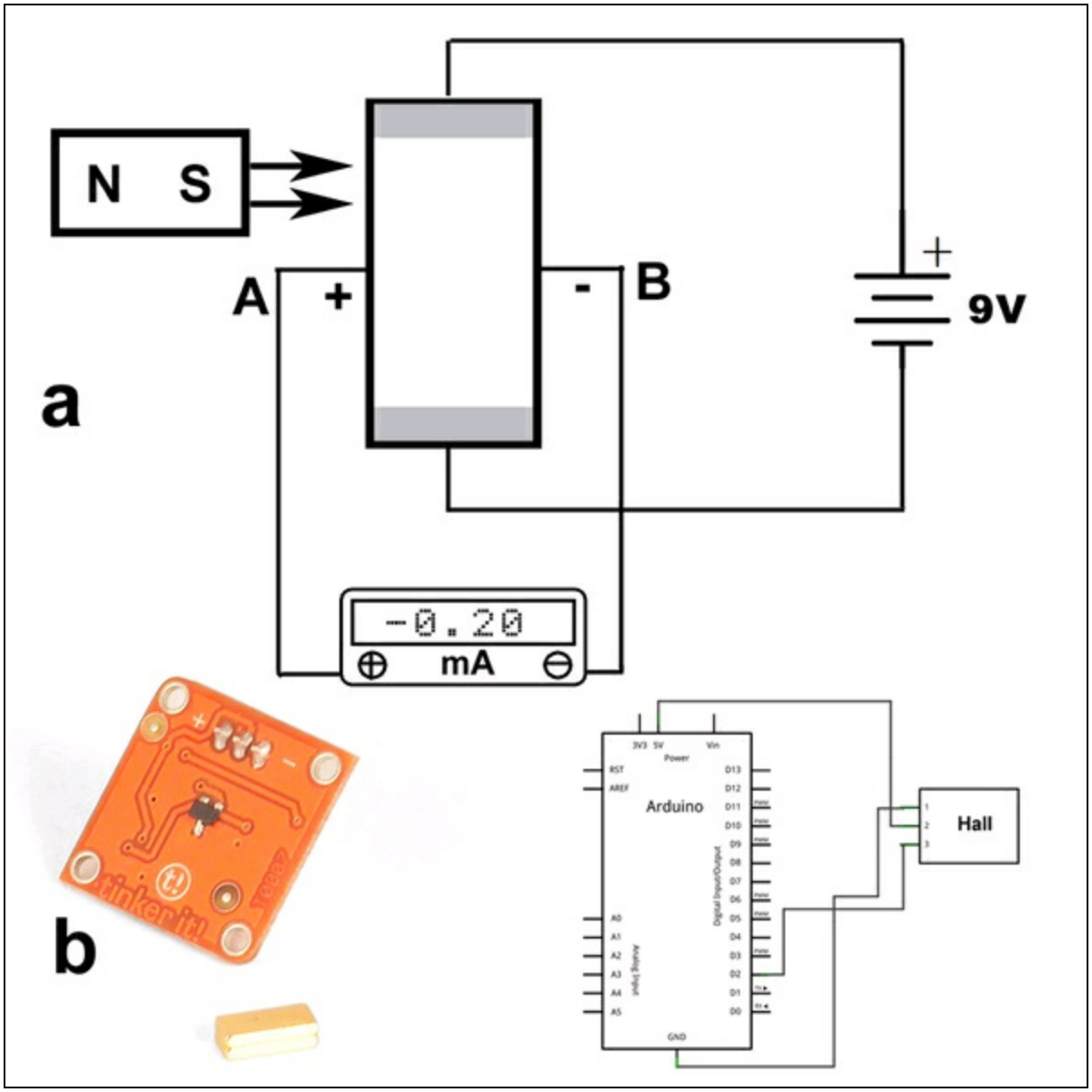


Figura 1.57 Modulo Hall Sensor prodotto da Tinkerkit e schema di collegamento.

Questo modulo restituisce 5 volt quando un campo magnetico (per esempio un corpo umano) si trova nei pressi del sensore e 0 quando non vi è nulla nelle vicinanze. Se connesso a un ingresso digitale, i valori variano tra 0 (nessuna presenza) e 1023 (presenza).

Sulla superficie del sensore vi sono alcuni contatti elettrici esposti; per questo motivo, al fine di evitare un corto circuito, è bene evitare di toccare la scheda con oggetti metallici.

Sensore di rilevamento geografico (GPS)

Una scheda GPS potrebbe essere considerata un sensore di rilevamento della posizione geografica, dato che GPS sta per *Global Positioning System*, ovvero sistema di posizionamento globale.

Il sistema GPS è basato su un vasto numero di satelliti in orbita geostazionaria, per cui la rilevazione della posizione relativa rispetto a una serie di satelliti fornisce la posizione e l'orario in ogni luogo del nostro pianeta. Viene pertanto usato per la navigazione e la geolocalizzazione in apparecchi di consumo come, per esempio, navigatori satellitari, smartphone e droni.

Volendo dotare il proprio progetto robotico di capacità di geolocalizzazione, sono disponibili in commercio alcune schede pronte all'uso, da installare facilmente in Arduino o Raspberry Pi.

Nella Figura 1.58 sono illustrati il GPS shield & data-logger prodotto da Adafruit (https://www.adafruit.com/product/1272) e il GPRS+GPS Quadband Module per Arduino/Raspberry Pi (SIM908) prodotto da Cooking Hacks (https://www.cooking-hacks.com/gprs-gps-quadband-shield-for-raspberry-pi-sim908).

Questi shield vanno montati a sandwich su una scheda Arduino o Raspberry Pi e necessitano di un programma per la lettura dei dati provenienti dal GPS. I dati in arrivo possono venire memorizzati su una card SD. Nei siti dei produttori si trovano molte risorse gratuite per sviluppare il software di geolocalizzazione. Con il modulo GPRS SIM908 è possibile trasformare Arduino o Raspberry Pi in un telefono cellulare per chiamate vocali e SMS oppure per il collegamento dati su Internet. Il modulo SIM908 si programma con normali comandi AT, come un comune modem analogico.

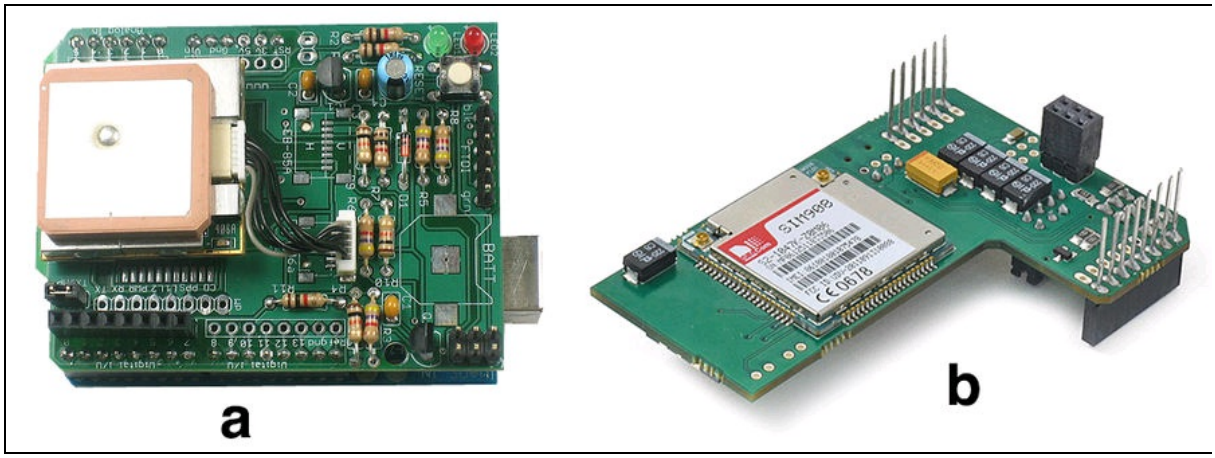


Figura 1.58 Il GPS shield & data-logger prodotto dalla Adafruit (a). Il GPRS+GPS Quadband Module for Arduino/Raspberry Pi (SIM908) di Cooking Hacks (b).

Sensore tattile

In uno dei nostri progetti abbiamo usato un sensore tattile che rileva l'impronta digitale. Fra i diversi modelli in commercio, la scelta è ricaduta sul Fingerprint Scanner TTL GT-511C3 di SparkFun, facilmente reperibile online e compatibile con Arduino.

Il sensore permette di leggere e memorizzare fino a 200 impronte digitali. Riconosce e confronta un'impronta letta con quelle memorizzate nel database interno. Arduino (o altri controller) deve solo inviare tramite la porta UART i comandi al sensore e attendere la risposta. La pagina di SparkFun dedidata al prodotto (`https://www.sparkfun.com/products/11792`) spiega esaurientemente il protocollo di comunicazione, ma per sviluppare l'interfaccia di comunicazione con il sensore, si può scaricare la libreria per Arduino già pronta all'uso.

Sensore RFID/NFC

In uno dei nostri progetti abbiamo usato un sensore RFID/NFC per il controllo accessi/presenza. Il modulo più diffuso in assoluto, disponibile

online anche in un kit molto economico, è il lettore RFID RC522. Si tratta di un dispositivo che consente di leggere i tag RFID/NFC molto facilmente grazie a un libreria per Arduino pronta all'uso.

L'acronimo RFID sta per *Radio Frequency IDentification*, che in italiano suona come "identificazione a radiofrequenza". È una tecnologia per l'identificazione e la memorizzazione di informazioni di qualsiasi tipo su un supporto chiamato *tag* o *transponder*.

Figura 1.59 Sensore di impronte digitali.

L’acronimo NFC sta per *Near Field Communication*, che in italiano potrebbe essere tradotto come “comunicazione in prossimità”. Similmente a RFID, è una tecnologia che fornisce una connettività bidirezionale a corto raggio, fino a un massimo di 10 cm, tramite l’uso di tag.

In pratica, i tag sono etichette elettroniche su cui vengono memorizzate le informazioni dell’utente, oltre all’ID del tag, ovvero il numero univoco di identificazione dell’etichetta.

Il dispositivo di lettura/scrittura dei tag RFID/NFC è in grado di leggere e/o scrivere le informazioni contenute nei tag che sta interrogando.

Figura 1.60 Il kit del sensore RFID/NFC RC522.

La frequenza di trasmissione di solito è di 13,56 MHz e la velocità di trasmissione dei dati arriva a 424 kbit/s.

Il principio di funzionamento è molto semplice. Il tag contiene un microchip con un identificativo univoco, una memoria di circa 1Kb e un’antenna. Di solito il tag è passivo, nel senso che è privo di alimentazione elettrica, e viene alimentato al passaggio del lettore RFID che emette un segnale radio a frequenza che attiva il microchip del tag.

Una volta alimentato, il tag può rispondere al lettore, ritrasmettendogli un segnale contenente le informazioni memorizzate nel chip. Allo stesso modo, il lettore può diventare anche scrittore e scrivere i dati nell’area di memoria libera del tag.

Display

Una volta che i nostri progetti sono dotati di sensi, possono venire arricchiti di display per il monitoraggio delle funzioni o la visualizzazione dei dati in ingresso e uscita.

Di tutti i dispositivi disponibili, ci soffermiamo sui quattro tipi di display più comunemente usati e facilmente reperibili:

- LCD 16 × 2;
- LCD grafico 128 × 64 pixel;
- display a sette segmenti;
- display OLED;
- touchscreen.

LCD 16 x 2

LCD è l'acronimo di *Liquid Crystal Display*, ovvero schermo a cristalli liquidi. Questo tipo di schermo è diffusissimo ovunque sotto forma di monitor per computer o apparecchi TV. Per alcuni dei nostri progetti abbiamo pensato a un display LCD molto diffuso e dal costo veramente irrisorio. Lo schermo che tratteremo è un diffusissimo modello con driver Hitachi HD44780, in grado di visualizzare 16 caratteri su 2 righe, ma soprattutto è compatibile con Arduino e Raspberry Pi. Una volta collegato a uno di questi controller, è possibile alimentare senza problemi questo display direttamente dall'alimentazione della scheda e regolare il contrasto tramite un potenziometro.

Di solito i caratteri sono bianchi su fondo blu o neri su fondo verde, a seconda del produttore. Si tratta di un LCD facilmente reperibile,

economico e semplicissimo da implementare e da programmare (Figura 1.61).

Il display dispone di un set interno di 256 caratteri ASCII esteso, con un'area di memoria per la creazione di 8 caratteri personalizzati.

Per i dettagli sull'uso del display LCD, si vedano i progetti della seconda parte del libro.

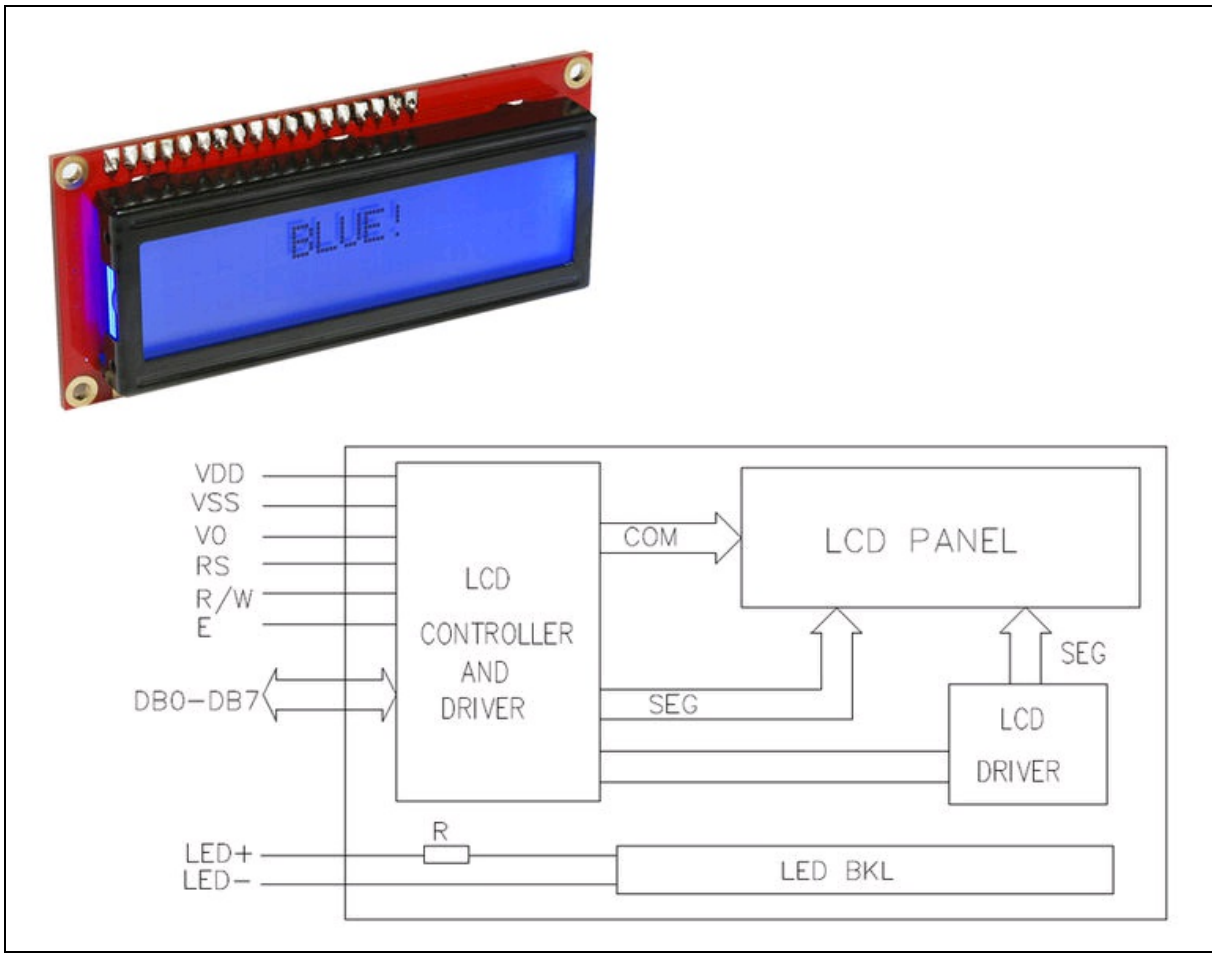


Figura 1.61 Display LCD con driver Hitachi HD44780 e schema a blocchi.

LCD grafico 128 x 64 pixel

Una visualizzazione senza dubbio più professionale può venire offerta da un display grafico dotato di una risoluzione di 128 × 64 pixel. Un display con queste caratteristiche molto diffuso ed economico è

sicuramente quello prodotto dall'azienda Crystal Clear Technology, più precisamente il modello della serie G64128X17 (Figura 1.62), ma ne esistono di molte altre marche. L'importante che sia compatibile con la libreria u8glib per Arduino (https://github.com/olikraus/U8glib_Arduino).

Oltre a poter gestire i dati di testo, il display grafico può visualizzare disegni, font, immagini e forme geometriche per creare interfacce o visualizzare forme d'onda per strumenti di misurazione.

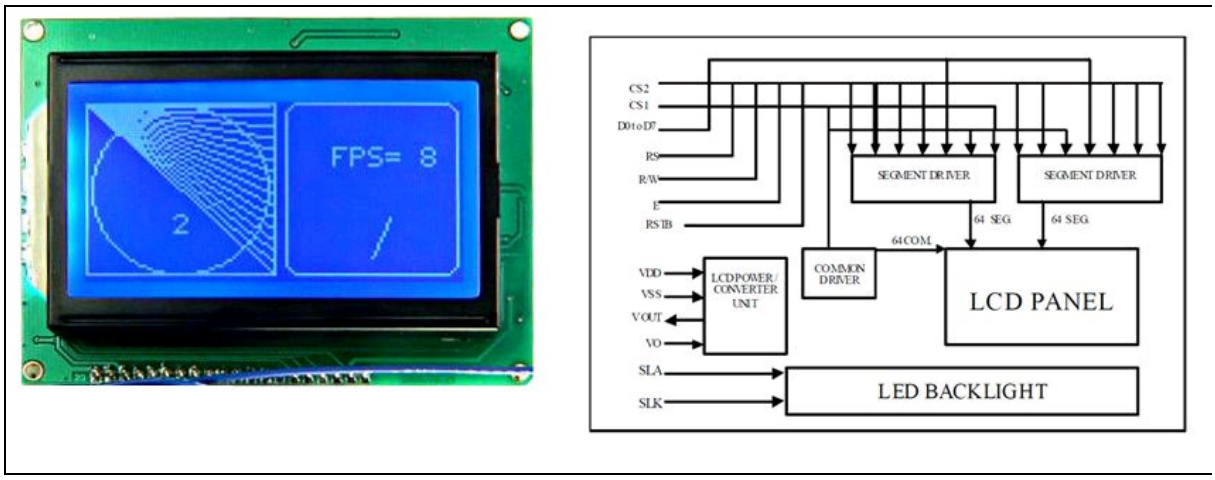


Figura 1.62 Display LCD grafico modello G64128X17 e schema a blocchi.

Display a LED sette segmenti

Si tratta di un particolare circuito a LED usato come display alfanumerico per il monitoraggio di parametri o funzioni di dispositivi elettronici (Figura 1.63a).

In pratica, il circuito per il controllo dei 7 segmenti LED è pronto all'uso ed è facilmente implementabile con l'utilizzo di un circuito integrato dotato di porte logiche come il 7447 o 7448 o, come vedremo, direttamente dalle porte digitali di un processore.

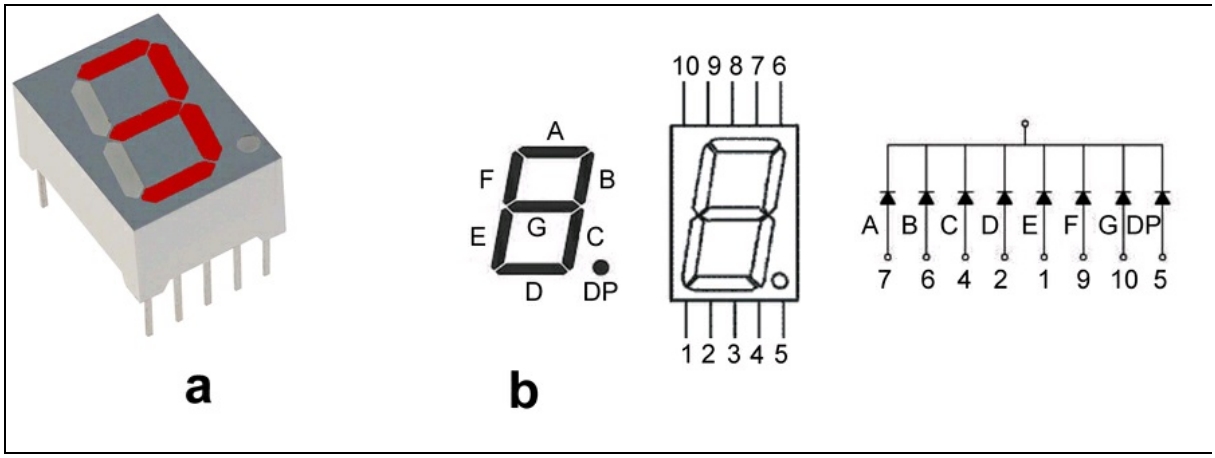


Figura 1.63 Display LED a 7 segmenti (a). Piedinatura tipica dei display FND serie 500.

Controllo display 7 segmenti

Fra i moltissimi tipi di display a 7 segmenti, uno dei più comunemente usati è siglato FND serie 500. Quello impiegato per gli esempi è a catodo comune, ma ce ne sono ad anodo comune, di diversi colori, grandezze. La logica di funzionamento è comunque sempre la stessa. Cosa si possa fare con un display a 7 segmenti dipende solo dalla fantasia. Per esempio, può essere usato per il conteggio di oggetti, come display di parametri, come timer, come orologio o anche solo per dare un tocco estetico d'altri tempi.

La grande visibilità a distanza in ambienti anche con forte illuminazione lo rendono talvolta insostituibile.

- *Display a 7 segmenti a catodo comune* - Tutti i catodi dei LED sono collegati a un pin che deve essere collegato alla massa del circuito. In un circuito digitale i segmenti si accendono quando il loro anodo viene collegato a un tensione positiva, corrispondente al dato 1 logico (HIGH).
- *Display a 7 segmenti ad anodo comune* - Tutti gli anodi dei LED sono collegati a un pin che deve essere collegato al positivo del

circuito. In un circuito digitale i segmenti si accendono quando il loro catodo viene collegato a massa, corrispondente al dato 0 logico (LOW).

- *Logica di funzionamento* - I display a 7 segmenti più comuni hanno 10 pin e la piedinatura è illustrata nella Figura 1.63b, nella quale vengono riportati anche i riferimenti ai segmenti e al punto decimale (*Decimal Point*).

Ci sono circuiti di decodifica per controllare l'accensione programmata dei segmenti per poter visualizzare numeri da 0 a 9 più il punto decimale e anche alcune lettere dell'alfabeto. Per esempio, si possono visualizzare in modo riconoscibile le lettere maiuscole A, C, E, F, H, J, L, P, U e le minuscole b, c, d, e, g, h, i, o, r, t e u. Ovviamente si possono accendere i segmenti in qualsiasi altro per visualizzare i simboli più strani.

Display OLED

Oltre ai già citati display a cristalli liquidi, si stanno sempre più affermando i cosiddetti display OLED, sigla che sta per *Organic Light Emitting Diode*, ovvero diodo organico a emissione di luce (Figura 1.64).

Figura 1.64 Un display OLED da 0,96 pollici.

A differenza dei display a cristalli liquidi, questa tecnologia permette di realizzare display a colori con la capacità di emettere luce propria. In pratica, gli OLED non richiedono componenti aggiuntivi per essere illuminati mentre i display a cristalli liquidi vengono solitamente retroilluminati da una fonte di luce esterna. Gli OLED producono luce propria e questo permette di realizzare display molto sottili e addirittura pieghevoli.

Grazie alla natura monopolare degli strati di materiale organico, i display OLED conducono corrente solo in una direzione, comportandosi quindi in modo analogo a un diodo.

Un’altra caratteristica molto importante è di essere molto dettagliati e precisi nella visualizzazione di caratteri e immagini. Anche con un piccolo display OLED da 0,96 pollici si può avere una risoluzione di 128×64 pixel e un angolo di visione maggiore di 160 gradi.

Il collegamento al controller sfrutta il protocollo I^2C, con solo due pin di I/O; le librerie di Arduino permettono di utilizzare questi display con estrema facilità.

Touchscreen

Il touchscreen che presentiamo qui è lo schermo touchscreen da 7 pollici esclusivamente creato per Raspberry Pi prodotto dal sito ufficiale (Figura 1.65).

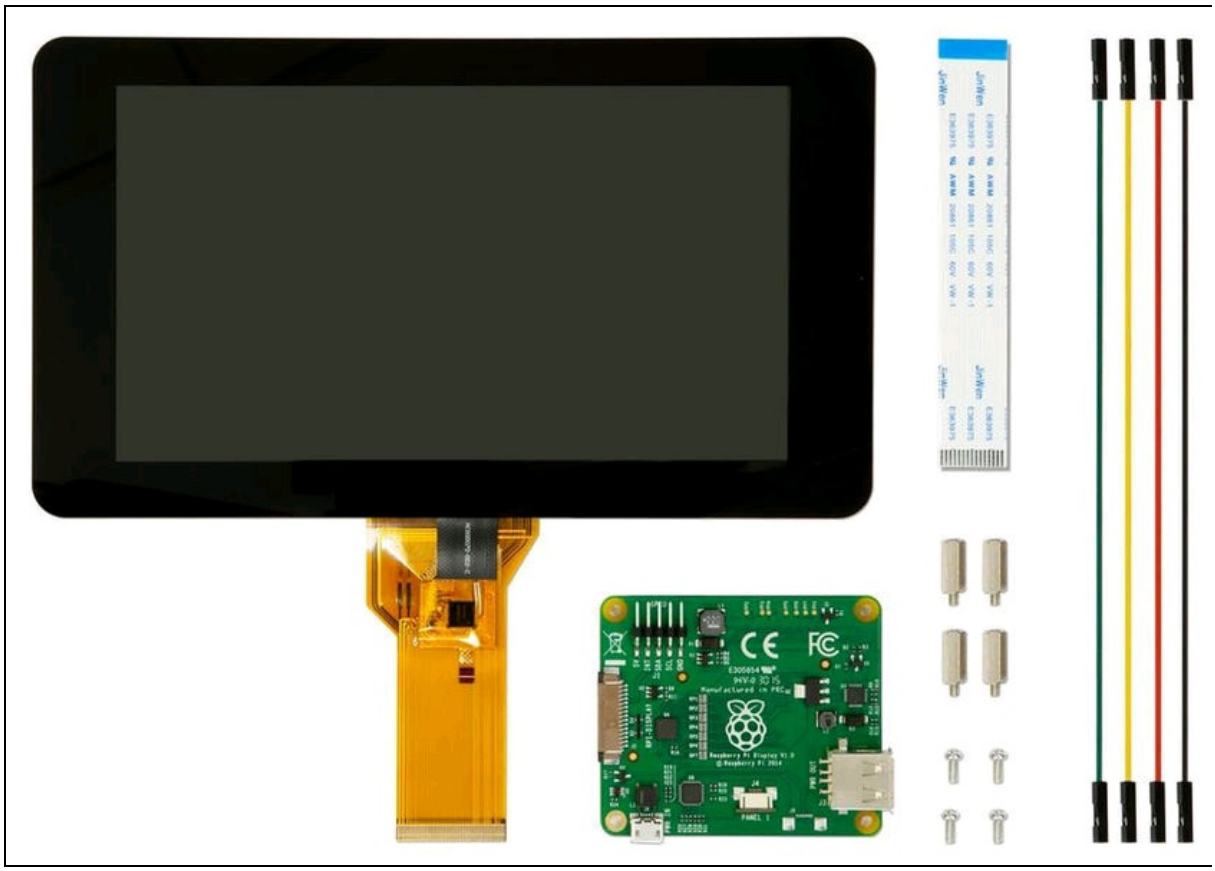


Figura 1.65 Official Raspberry Pi 7” Touchscreen Display.

Il display offre funzioni dedicate alla visualizzazione e al controllo di un Raspberry Pi con una risoluzione di 800×480 pixel e si collega tramite una scheda/adattatore che gestisce sia l'alimentazione sia il segnale ad alta definizione.

Sono necessarie solo due connessioni: l'alimentazione a 5 V dalla porta GPIO del Raspberry Pi e un cavo a nastro che si collega alla porta DSI presente su tutti i modelli di Raspberry Pi. I driver software touchscreen con supporto per il tocco a 10 dita e una tastiera virtuale su schermo sono integrati nell'ultimo OS Raspbian e anche nel più recente Windows 10 IoT Core, per una completa funzionalità anche senza una tastiera fisica o un mouse.

Il kit *Official Raspberry Pi 7" Touchscreen Display*, completo di distanziali, viti, scheda/adattatore, cavetti e connettore a nastro DSI, è disponibile nello store ufficiale di Raspberry Pi, all'indirizzo `http://thepihut.com`. Opzionalmente si può acquistare anche una cornice colorata a scelta. Le istruzioni di montaggio dettagliate sono disponibili nella stessa pagina di acquisto.

Attuatori

Sarebbe sicuramente molto utile fare una digressione sugli elementi di meccanica applicata e la meccanica di base. È un mondo affascinante almeno quanto l'elettronica. Per gli scopi di questo libro ci soffermeremo solamente sull'utilizzo di alcuni attuatori facilmente abbinabili ai nostri progetti elettronici.

NOTA
I concetti base di meccanica classica sono elencati nell'Appendice.

L'attuatore viene definito come un meccanismo attraverso il quale si agisce sull'ambiente fisico. In altre parole, l'attuatore è un meccanismo che mette qualcosa in azione automaticamente. Fra i diversi tipi di attuatori, ecco quelli che possono fare al nostro caso:

- attuatore elettrico;
- attuatore meccanico;
- attuatore idraulico.

Attuatore elettrico

Un attuatore elettrico è alimentato da un motore che converte l'energia elettrica in coppia meccanica. Un esempio tipico è quello del relè, di cui abbiamo parlato in questo capitolo. In alcuni progetti verrà usato un comodo modulo relè multiplo, compatibile con Arduino e Raspberry Pi (Figura 1.66).

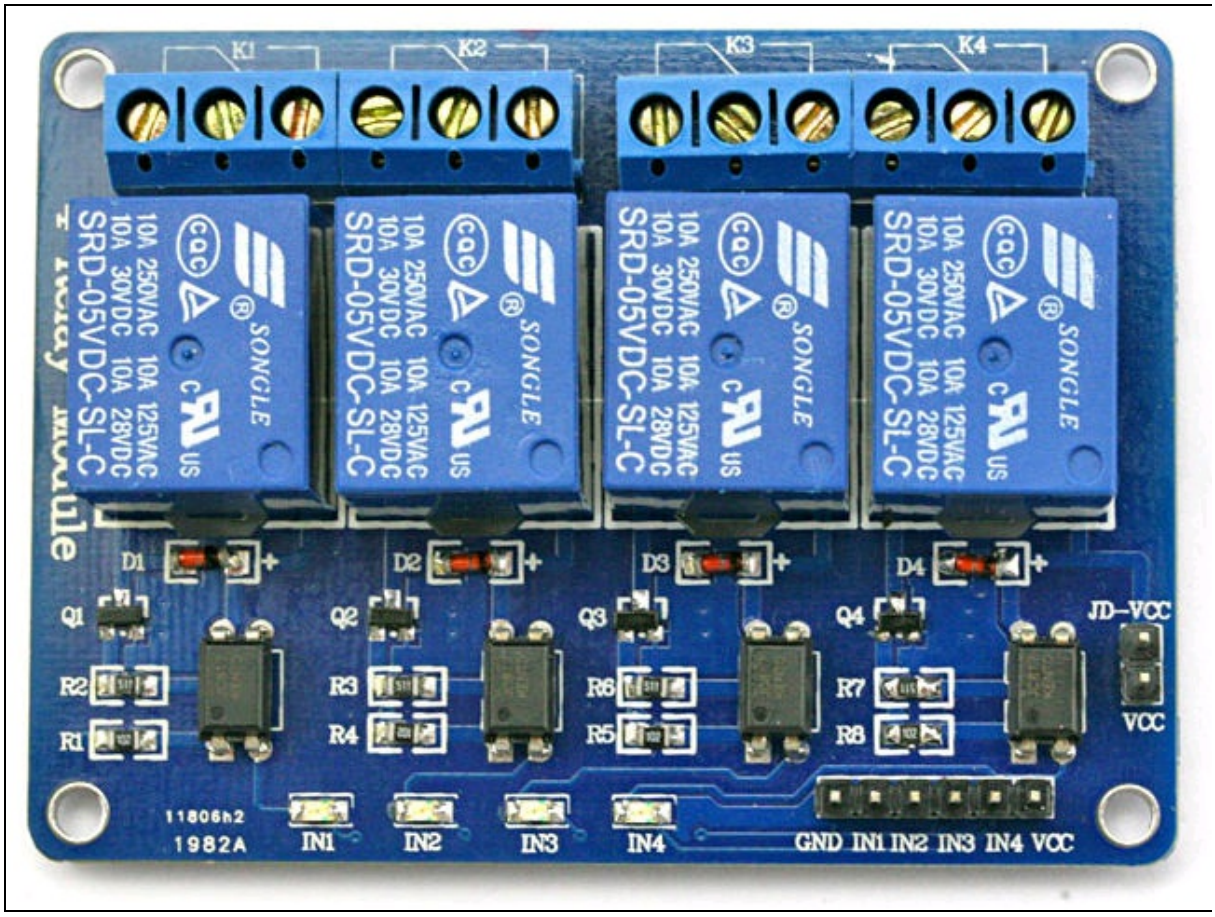


Figura 1.66 Modulo a 4 relè per Arduino e Raspberry Pi.

Attuatore meccanico

Un attuatore meccanico converte un tipo di movimento, per esempio rotatorio, in un altro tipo di movimento, per esempio lineare. Il funzionamento degli attuatori meccanici si basa su combinazioni di componenti strutturali, quali ingranaggi, rotaie, pulegge e catene.

Per aprire cancelli e porte o per spostare l’angolazione della TV e così via, sono utili gli attuatori meccanici illustrati nella Figura 1.67.

Figura 1.67 Attuatori meccanici.

Attuatore idraulico

Un attuatore idraulico è costituito da un motore e un cilindro che utilizza la potenza idraulica dei fluidi per facilitare il funzionamento meccanico. Poiché i liquidi sono quasi impossibili da comprimere, un attuatore idraulico può esercitare una grande forza.

Il cilindro idraulico è costituito da un tubo cilindrico cavo, lungo il quale può scorrere un pistone. Quando su un lato del pistone viene applicata la pressione del fluido, il pistone può muoversi in una sola direzione. Una molla viene spesso utilizzata per garantire il ritorno del pistone.

Per spostare un cancello o altri oggetti molto pesanti può essere utile un attuatore idraulico come quello illustrato nella Figura 1.68.

Figura 1.68 Attuatore idraulico.

Il laboratorio del maker

Questo libro è dedicato ai maker o a coloro che lo vogliono diventare. In ogni caso, per realizzare i progetti del libro, serve un minimo di attrezzattura e una certa quantità di componenti essenziali.

In tempi recenti abbiamo assistito a un fenomeno di crescita del fai da te elettronico e lo si vede dalla proliferazione dei negozi online. Al contrario, sono sempre meno diffusi i negozi fisici. Solo nei grossi centri urbani è possibile trovare il negozio ben fornito di componenti elettronici e aggiornato con le ultime schede Arduino, Raspberry Pi e simili.

La cosa più semplice è affidarsi a Internet, stando però attenti all'ubicazione dei fornitori. Se sono oltreoceano, si rischia di pagare un'enormità per il trasporto e per i dazi doganali.

Fortunatamente sono presenti negozi online italiani che importano grandi quantitativi dai giganti dell'elettronica per maker, per cui, anche pagando leggermente di più rispetto al prezzo in dollari, si può risparmiare sul trasporto e sul dazio. Un'alternativa che sta diventando sempre più conveniente, ma anche più divertente, sono le fiere di elettronica. Ultimamente si stanno diffondendo un po' dappertutto e offrono sempre più spesso materiale elettronico di qualità. Oltre ai soliti banchetti con pacchi di componentistica surplus a prezzi irrisori, si vedono anche le schede Arduino, Raspberry Pi, vari shield e kit per il principiante. Per chi inizia l'organizzazione del proprio laboratorio da maker è l'occasione per trovare i materiali di base. Ecco un elenco di cosa potrebbe essere utile:

- scheda Arduino;
- scheda Raspberry Pi;
- resistori di vari valori;

- condensatori di vari valori;
- diodi di vari tipi;
- LED di vari colori;
- transistor vari (scelti in base ai progetti);
- circuiti integrati (scelti in base ai progetti);
- display LCD;
- breadboard;
- cavetti per breadboard MM e MF;
- pulsanti mini da PCB;
- interruttori mini da PCB;
- cassettiere per componenti elettronici.

Strumentazione essenziale

Oltre a questa dotazione di base, e limitandoci allo stretto necessario, sarebbero utili i seguenti strumenti:

- multimetro;
- alimentatore da banco;
- stazione saldante;
- terza mano.

Multimetro

Il multimetro (Figura 1.69a), chiamato anche *tester*, è indubbiamente un apparecchio utile in qualsiasi laboratorio di elettronica. Esistono multimetri di diverse tipologie e specializzati per misurazioni elettriche particolari. È possibile acquistare per pochi euro un multimetro digitale di discreta qualità, ma è consigliabile non risparmiare troppo.

Un multimetro deve dare, possibilmente, misurazioni con un’approssimazione vicina allo zero, anche se nel campo dell’hobbistica non viene richiesta una tale precisione.

In ogni caso, il multimetro deve essere in grado di svolgere le seguenti misurazioni:

- resistenza elettrica in ohm;
- tensione elettrica continua DC in volt;
- tensione elettrica alternata AC in volt;
- corrente elettrica continua DC in ampere;
- corrente elettrica alternata AC in ampere.

Nei multimetri un po’ più sofisticati, si possono trovare anche le seguenti caratteristiche:

- misurazioni di capacità e induttanze;
- prova diodi e transistor;
- misurazioni di correnti fino 10 A e oltre.

Oggi sono sempre più diffusi i multimetri digitali, ma esistono ancora i multimetri analogici con amperometro ad ago. Per un certo tipo di misurazioni elettriche è sicuramente più comodo osservare l’andamento “analogico” del movimento dell’ago per rilevare variazioni continue di un segnale, altrimenti difficili da interpretare tramite il display numerico di un multimetro digitale.

Alimentatore da banco

Per alimentare i circuiti con assorbimenti maggiori e tensioni diverse da 3,3 o 5 volt della porta GPIO di Raspberry Pi o della porta di alimentazione di Arduino, sarebbe utile un alimentatore da banco regolabile 0-30 volt, 5 ampere, con display digitale (Figura 1.69b).

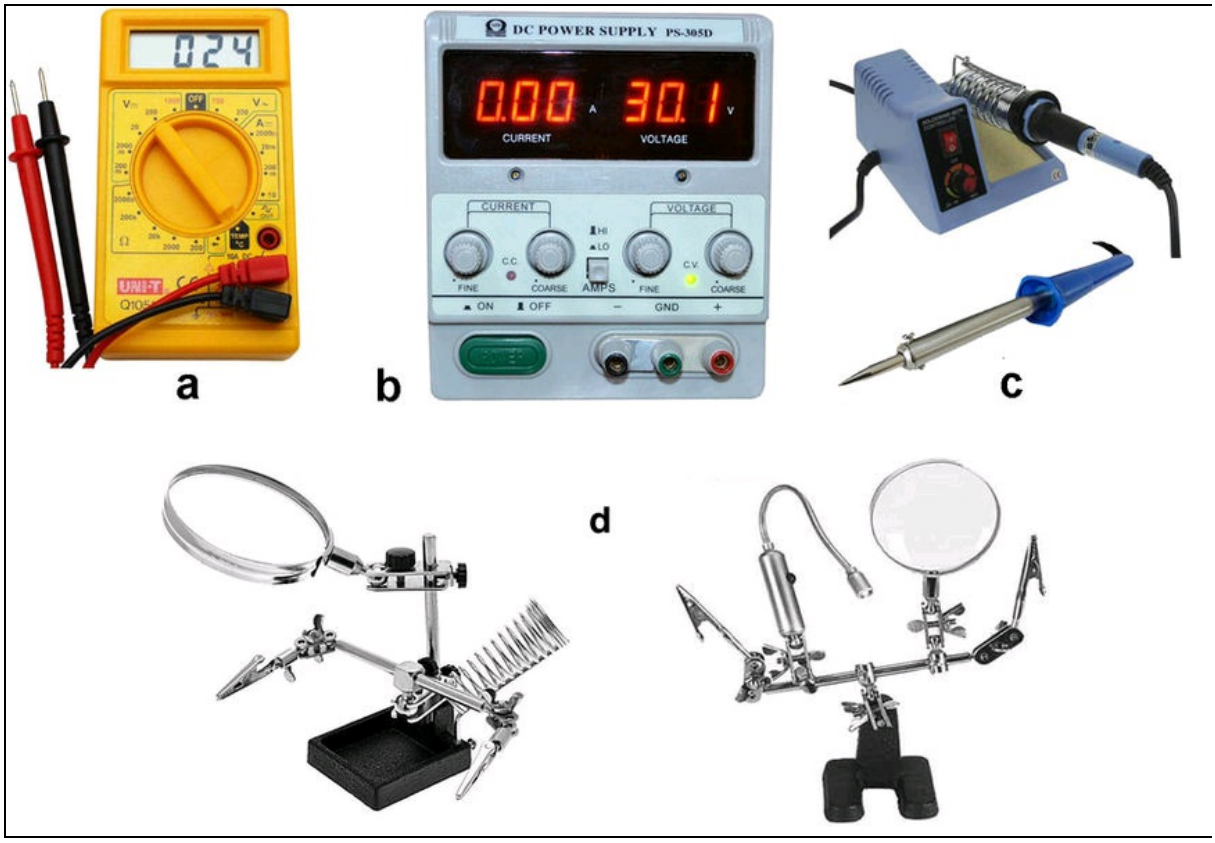


Figura 1.69 Multimetro digitale (a). Alimentatore da banco (b). Stazione saldante e saldatore a stilo (c). Terza mano (d).

Stazione saldante

Infine, per passare a circuiti più stabili di una breadboard, cioè una millefori o un circuito stampato, è necessaria una stazione saldante con temperatura regolabile o un saldatore a stilo da 30 watt (Figura 1.69c).

La terza mano

Soprattutto se si lavora da soli, l'operazione di saldatura diventa problematica quando è necessario tenere ferma la basetta o quando si devono saldare cavi, prese, spine e quant'altro. Il sistema di aiuto più diffuso viene chiamato amichevolmente "terza mano".

La Figura 1.69d illustra un paio di modelli disponibili in commercio. Di solito, una terza mano è dotata di lente di ingrandimento, di coccodrilli orientabili, di vaschetta porta-spugnetta e di un supporto porta-saldatore. Alcuni modelli sono dotati anche di illuminazione sotto la lente o di faretto orientabile a LED.

Arduino Starter Kit

Esistono in commercio molti Starter Kit per hobbisti e maker. È innegabile il fascino che esercitano queste confezioni o le cosiddette valigette di elettronica su chi, come chi scrive, è appassionato di elettronica.

Il consiglio è quello di leggere sempre il contenuto per evitare di riempire il laboratorio di cose inutili o scarsamente riutilizzabili. E non è necessario sborsare cifre assurde.

Ci sono molti siti che vendono l'Arduino Starter Kit. Qui facciamo riferimento a quello venduto dal sito ufficiale di Arduino (http://arduino.cc).

L'Arduino Starter Kit (Figura 1.70) comprende tutto il materiale per decine di progetti. Al di là della bellezza intrinseca del kit, oltre ai progetti fatti con Arduino si potranno riutilizzare molti dei componenti elettronici del kit per ulteriori progetti. Ecco il contenuto del kit:

- 1× scheda Arduino UNO rev.3;
- 1× cavo USB;
- 1× breadboard;
- 1× basetta di supporto in legno;
- 1× adattatore per batteria 9V;
- 70× cavetti plastificati rigidi;
- 2× cavetti plastificati flessibili;
- 6× fotoresistori;

- 3× potenziometri da 10 kΩ;
- 10× tasti a pressione;
- 1× sensore di temperatura;
- 1× sensore di tilt;
- 1× LCD alfanumerico (16×2 caratteri);
- 1× LED (bianco);
- 1× LED (RGB);
- 8× LED (rosso);
- 8× LED (verde);
- 8× LED (giallo);
- 3× LED (blu);
- 1× motore CC 6/9V;
- 1× servomotore;
- 1× cicalino piezoelettrico;
- 1× ponte H L293D;
- 2× fotoaccoppiatore 4N35;
- 5× transistor BC547;
- 2× transistor mosfet IRF520;
- 5× condensatori 100 nF;
- 3× condensatori 100 µF;
- 5× condensatori 100 pF;
- 5× diodi 1N4007;
- 3× gelatine trasparenti (rosso, verde, blu);
- 1× strip di connettori maschio (40×1);
- 20× resistori 220 Ω;
- 5× resistori 560 Ω;
- 5× resistori 1 kΩ;
- 5× resistori 4,7 kΩ;
- 10× resistori 10 kΩ;
- 5× resistori 1 MΩ;

- 5× resistori 10 MΩ;
- 1× libro di progetti.

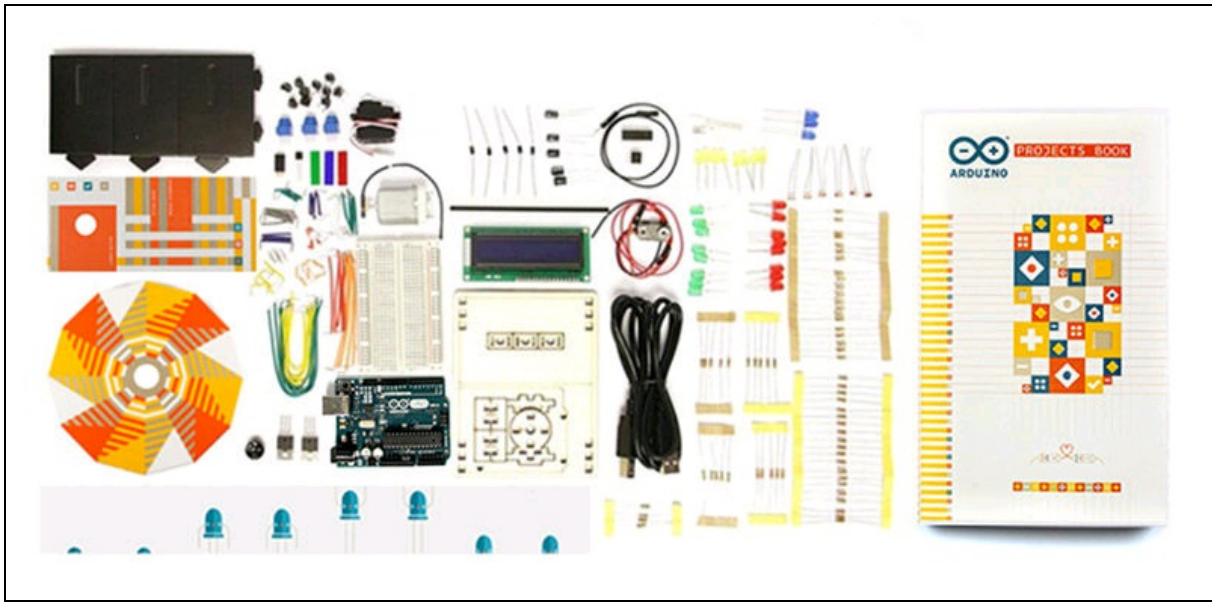


Figura 1.70 Arduino Starter Kit.

Come si può vedere, il kit è davvero ricco e permette di creare molti progetti elettronici proposti nella confezione e anche in questo libro.

Si consiglia di separare tutti componenti in apposite cassettiere con l'etichetta del contenuto: resistenze, diodi, transistor, motori, integrati e così via.

Sulla falsariga del sito Arduino sono nati molti altri distributori di hardware elettronico che offrono kit simili al precedente, come per esempio *Starter Pack for Arduino* del sito Adafruit (`www.adafruit.com`) oppure *SparkFun Inventor's Kit*, distribuito da Robot-Italy (`www.robot-italy.com`).

Microsoft IoT Pack for Raspberry Pi 3

Con l'introduzione di Windows 10 IoT Core, Microsoft ha stretto rapporti con il colosso dell'elettronica Adafruit. Da questo connubio è

nato *Microsoft IoT Pack for Raspberry Pi 3*, un kit molto ricco basato su Raspberry Pi 3, che può essere acquistato con o senza la scheda. È uno dei modi per iniziare a utilizzare Windows 10 Iot Core e Raspberry Pi 3 come dispositivo IoT.

A partire da dicembre 2016 questo pacchetto viene fornito con la versione aggiornata di Windows 10 IoT per Raspberry Pi 3, ma ci potranno essere aggiornamenti in futuro.

La pagina di riferimento è: https://www.adafruit.com/product/2733.

La confezione contiene (Figura 1.71):

- 1× Raspberry Pi 3;
- 1× Adafruit Raspberry Pi B+ Case;
- 1× breadboard;
- 20× cavetti MM;
- 20× cavetti MF;
- 1× modulo Wi-Fi Raspberry Pi;
- 1× alimentatore switching 5 V 2A;
- 1× sensore di temperatura e di pressione Adafruit BMP280 assemblato;
- 1× MCP3008 ADC 8 canali a 10 bit con interfaccia SPI;
- 1× cavo Ethernet;
- 1× micro-SD 8GB con Windows 10 IoT Core precaricato;
- 1× fotoresistori;
- 2× potenziometr1;
- 5× resistori 10 KΩ;
- 5× 560 Ω;
- 2× LED blu;
- 1× condensatore elettrolitico 1.0 µF;
- 2× LED rosso;
- 2× LED verde;
- 3× mini pulsanti.

Con questo kit si possono condurre molti esperimenti proposti nel portale IoT di Microsoft e anche in questo libro.

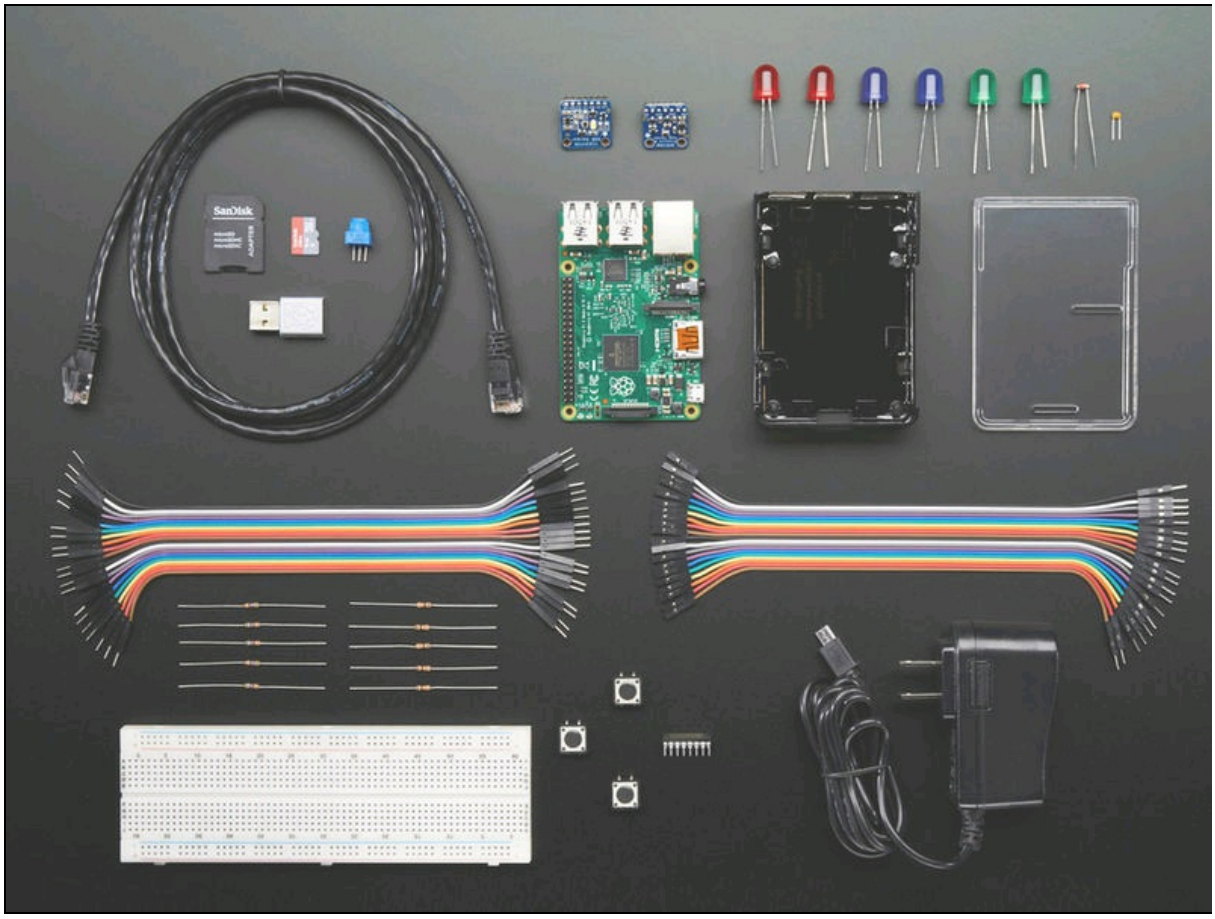


Figura 1.71 Microsoft IoT Pack for Raspberry Pi 3.

Breadboard

Inutile dire che se non ci fosse, bisognerebbe inventarla. La basetta sperimentale, comunemente chiamata *breadboard*, ha le dimensioni standard di circa 165 × 55 mm, ma esistono modelli di dimensioni pari alla metà o al doppio delle dimensioni standard e per alcuni kit di sperimentazione particolari si possono trovare anche di dimensioni mini e tiny.

La logica di funzionamento è sempre la stessa: i collegamenti non richiedono saldature e basta inserire negli appositi fori spezzoni di cavo rigido e/o cavetti appositi per breadboard per poter collegare i vari componenti elettronici in modo veloce e sicuro.

In pratica, l'elemento base è composto da una matrice di fori a passo standard (2,54 mm) collegati internamente secondo uno schema predefinito in senso verticale e orizzontale. Nella Figura 1.72a si può notare la foto di una breadboard "reale" di dimensioni standard, mentre la Figura 1.72b è uno screenshot di breadboard "virtuale" di Fritzing (Capitolo 3).

Usare una breadboard è semplicissimo e prestando un po' di attenzione può essere riutilizzata indefinitamente. Attenzione a non forzare l'inserimento nei fori con cavi troppo grossi (oltre 1 mm) oppure applicare correnti troppo elevate (oltre 1 ampere). Ovviamente, i progetti su breadboard non potranno mai essere considerati definitivi, soprattutto perché è facile che qualche cavo di collegamento si sfili durante gli spostamenti. Il sistema è sicuramente vantaggioso, perché permette di poter verificare e modificare velocemente qualsiasi collegamento senza inutili perdite di tempo e, se tutto funziona su breadboard, si potrà passare a qualcosa di più stabile, come una millefori o un circuito stampato (PCB).

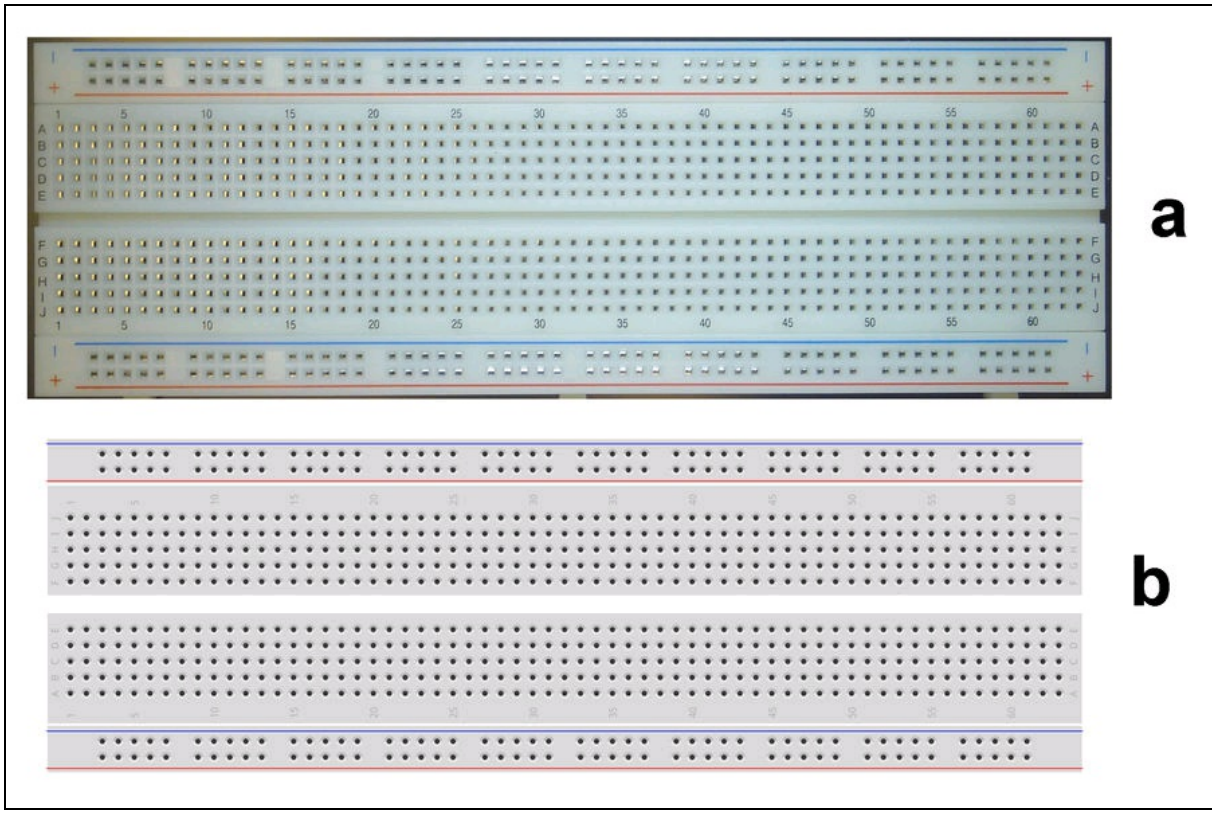


Figura 1.72 Breadboard reale e breadboard virtuale (a). Breadboard di tipo semplice di diverse dimensioni e senza contrassegni dell'alimentazione (b).

Una volta fatta un po' di pratica con la realizzazione di alcuni progetti, si potrà tentare di assemblare i prototipi direttamente su breadboard, ma per le prime volte si consiglia vivamente l'uso di Fritzing, di cui si parla nel Capitolo 3.

Alimentazione e uso della breadboard

Sulla parte esterna all'area centrale corrono due file parallele alle quali applicare l'alimentazione positiva (+) e la massa del circuito (-). Di solito si collega l'alimentazione positiva di 3,3 volt e 5 volt di Raspberry Pi o di Arduino. La massa del circuito è sempre contrassegnata con GND (*ground*). Tutti i pin GND sono collegati fra di loro.

Tenendo conto delle due sezioni di una breadboard di dimensioni standard (full+) si ha la possibilità di due alimentazioni indipendenti. Ci

sono breadboard in commercio con quattro sezioni di alimentazione con le due file separate a metà

Attenzione! Le file di alimentazione sono sempre collegate in senso longitudinale e sono solitamente contrassegnate dal colore rosso per l'alimentazione positiva e dal colore blu per il collegamento a massa.

Millefori

Un prototipo di un progetto non può certamente stare su una breadboard all'infinito. Il passo successivo è solitamente un circuito più stabile su una basetta millefori o, meglio, su circuito stampato, chiamato amichevolmente "basetta". In entrambe le soluzioni si saldano i componenti in modo definitivo.

La soluzione su millefori è la più sbrigativa. Si tratta di un supporto in vetronite o bachelite su cui sono depositate le piazzole in rame con un foro per la saldatura dei componenti passanti. Si fa notare che "millefori" è un nomignolo riferito al design della basetta e non alla reale quantità di fori. Sono disponibili in commercio decine di tipi di basette millefori di varie grandezze e con diverse tipologie di collegamento (Figura 1.73a).

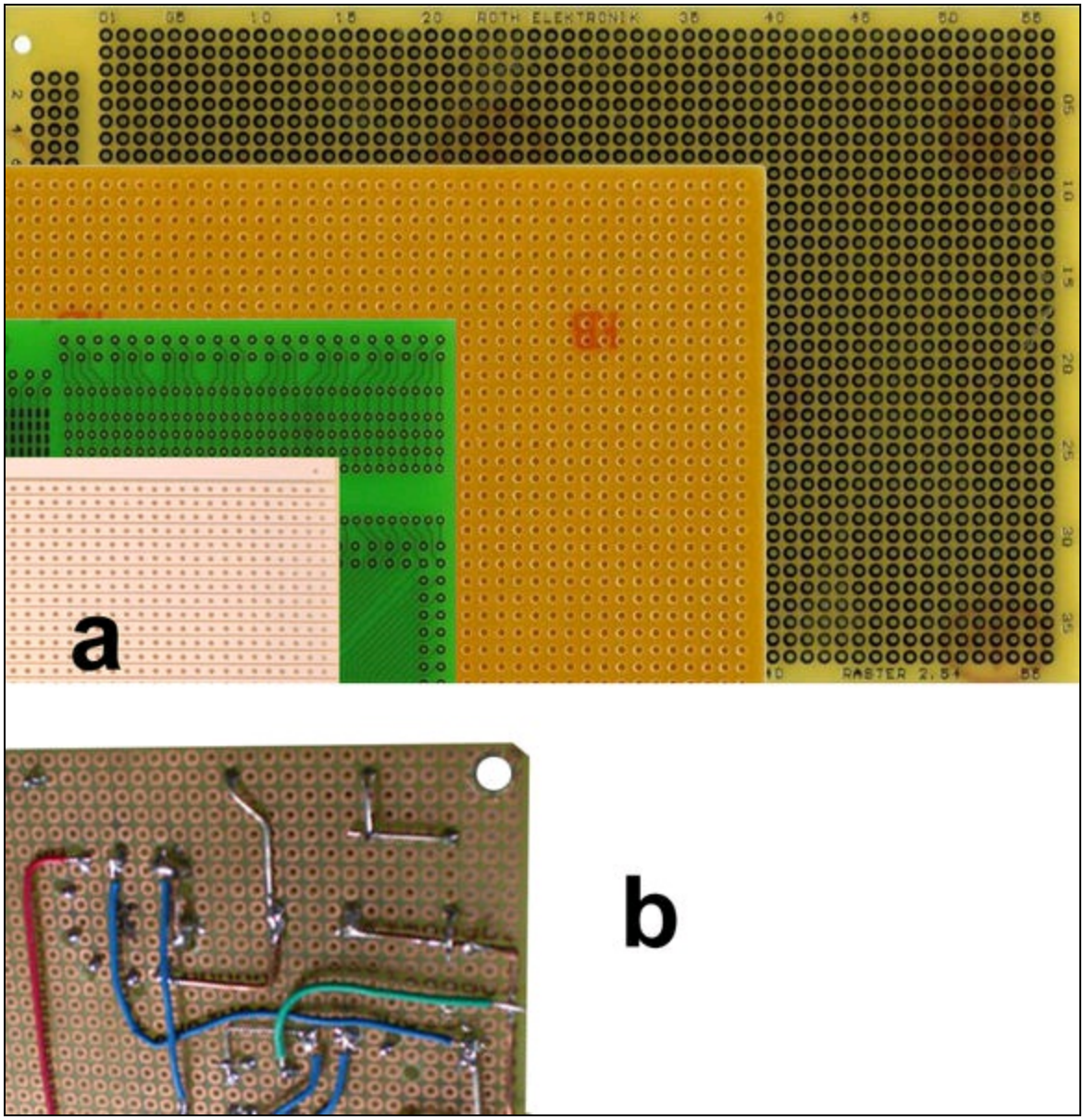


Figura 1.73 Vari tipi di basette millefori (a). Saldatura su millefori (b).

Per gli usi più comuni si utilizza la basetta standard con piazzole circolari e passo standard di 2,54 mm, che è lo stesso passo utilizzato per gli zoccoli dei circuiti integrati, delle strisce di pin, dei pulsantini e molti altri componenti elettronici standard.

Contrariamente al montaggio su breadboard, la soluzione millefori è molto più vicina a un progetto su PCB, perché la posizione dei

componenti e il collegamento dei terminali sulla parte ramata risultano molto simili a quelli della basetta PCB.

Prima di effettuare le saldature definitive, si possono progettare i circuiti su millefori anche con Fritzing, per essere più sicuri.

Sono disponibili in commercio basette millefori di varie misure, a singolo e a doppio lato, ma la misura più comune è il classico rettangolo in formato Eurocard 160 × 100 mm con spessore di 1,6 mm, spessore del rame di 35 µm e piazzole su un singolo lato.

Alcuni modelli più costosi hanno le piazzole prestagnate. Quelle a doppio lato hanno anche i fori metallizzati. Per applicazioni particolari esistono millefori di tipo “pad”, con piazzole collegate a gruppi, o “stripboard”, con piazzole collegate in strisce parallele. Queste ultime sono utili se si fa largo uso di circuiti integrati.

Le piste nella parte ramata possono venire unite saldando spezzoni di cavo rigido. Di solito si sfrutta la lunghezza dei reofori dei componenti passivi, saldandoli fra loro (Figura 1.73b).

Circuito stampato o PCB

La soluzione su circuito stampato o PCB (*Printed Circuit Board*) è sicuramente più professionale ed è alla portata di tutti.

Si può mandare un file di produzione a un servizio di stampa PCB anche tramite lo stesso software di progettazione Fritzing (si veda il Capitolo 3). Per cui è possibile ordinare da casa la stampa PCB del proprio prototipo o produrre un file professionale da mandare a un qualsiasi servizio di stampa PCB professionale.

La Figura 1.74 mostra una basetta professionale prodotta dal servizio di stampa Fritzing. Si notino la verniciatura bianca del Solder Mask, l’accuratezza della serigrafia e la prestagnatura dei fori.

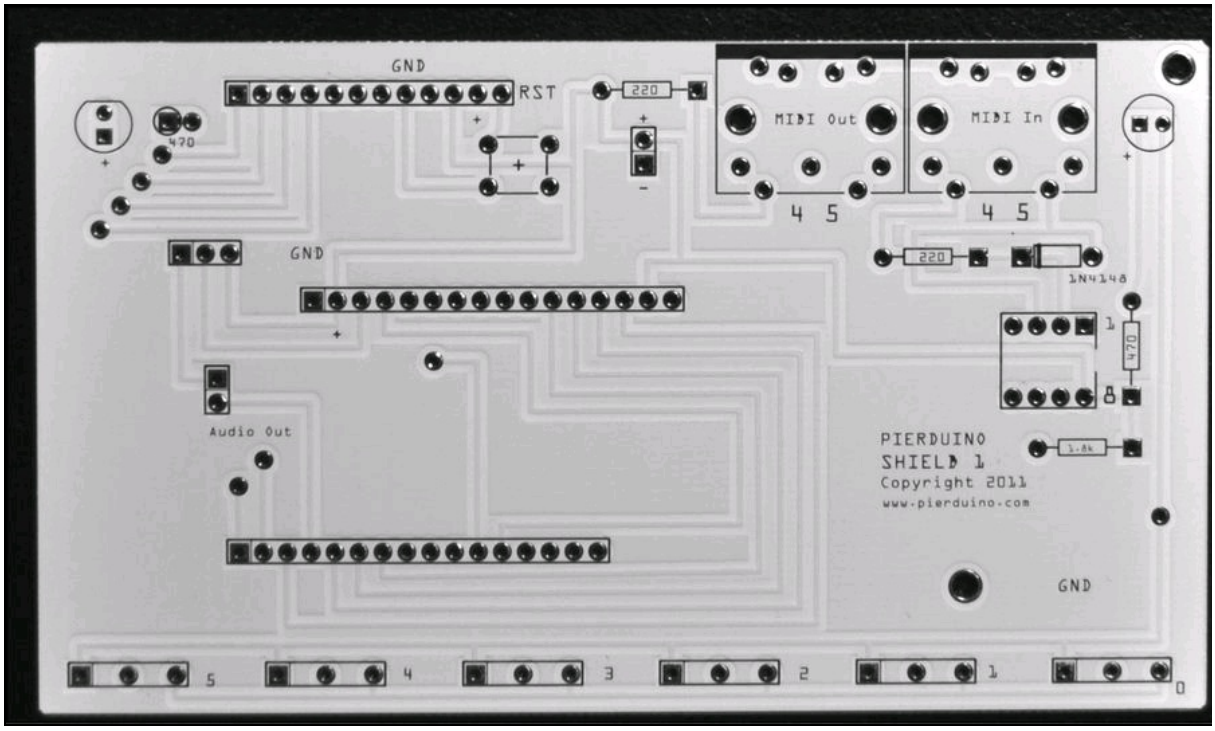


Figura 1.74 PCB a doppio lato dal servizio di stampa Fritzing.

Fotoincisione

Il passaggio da breadboard a millefori o circuito stampato è abbastanza semplice. Per chi invece si vuole cimentare nell'arte di fare tutto da soli, in stile *old-school*, esiste anche la possibilità di fabbricare in casa il proprio circuito stampato.

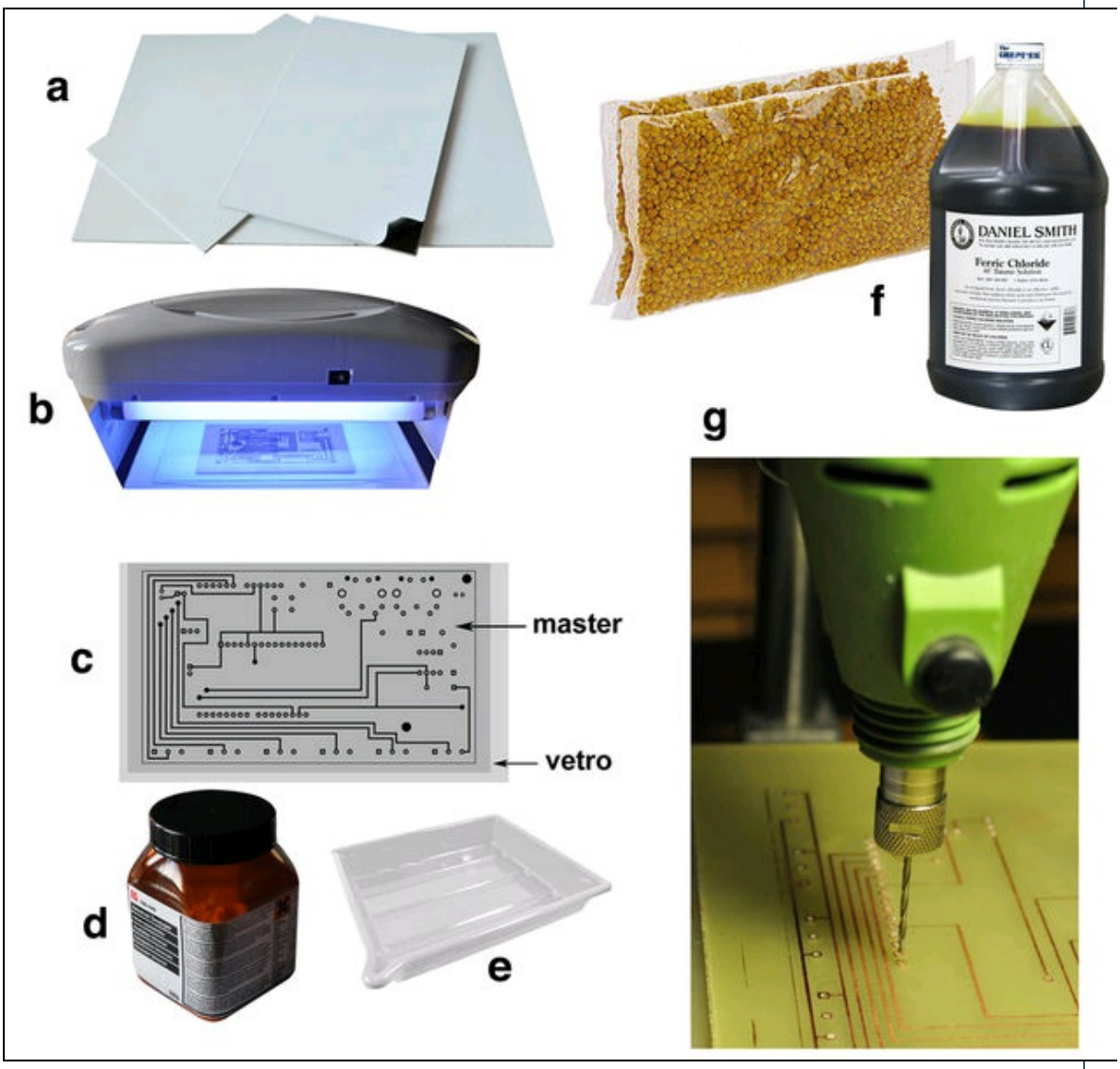
Il maker che "non vuole chiedere niente a nessuno" può pensare all'uso di basette presensibilizzate per la fotoincisione in casa. Chiaramente, il luogo per operare non può essere la cucina o il salotto. Per lo sviluppo fotografico e l'incisione chimica bisogna disporre di un ambiente adatto, come un angolo del garage o dello scantinato.

Le basette presensibilizzate sono disponibili in vari formati e sono solitamente offerte a prezzi accessibili. Non si consiglia l'applicazione manuale di Photoresist in bomboletta spray, da spruzzare su basetta ramata per rendere la superficie fotosensibile. Questo processo è complicato e non offre quasi mai il risultato sperato.

Le basette presensibilizzate (solitamente di tipo positivo) sono disponibili nel formato standard 160 x 100 mm, ma ce ne sono anche di misure maggiori (a). Esistono anche a doppia faccia presensibilizzata, per cui è possibile produrre circuiti stampati a doppio lato.

L'emulsione fotosensibile della basetta va esposta fotograficamente utilizzando una fonte omogenea di raggi ultravioletti. Il tempo di esposizione ai raggi ultravioletti può

variare in base alle caratteristiche fornite dal produttore, ovvero alle caratteristiche dell'emulsione, all'intensità e alla distanza della sorgente luminosa.



Normalmente il tempo di esposizione di aggira intorno ai 10-15 minuti se si usa una lampada UV o un bromografo. Esistono bromografi per l'hobbistica a prezzi accessibili e si trovano in rete molti progetti di bromografi fai-da-te. Per esperienza diretta di chi scrive, è sufficiente "hackerare" l'impianto di illuminazione di una comunissima zanzariera per costruire un perfetto bromografo dotato di due lampade UV (b). Per ottenere una buona maschera, altrimenti detta "master" di stampa, basta stampare il circuito tramite una stampante laser (consigliata) su un foglio di acetato trasparente e fissarlo poi alla basetta presensibilizzata (c). La procedura di esposizione è semplice: basta fissare con del nastro adesivo il master su un vetro e appoggiare il tutto sul lato sensibile della basetta. Una volta esposto

per il tempo ottimale il lato sensibile mascherato, si potrà togliere il master e procedere allo sviluppo dell'emulsione, esattamente come avviene in fotografia.
Dopo lo sviluppo dell'emulsione, e lavata la basetta da ogni residuo chimico, si potrà procedere all'incisione per corrodere le parti in rame che sono state esposte alla luce UV, lasciando intatte le piste in rame protette dal master. Per il processo di sviluppo si può usare una soluzione fatta in casa a base di idrossido di sodio (la comune soda caustica) oppure usare una soluzione commerciale (d) senza idrossido di sodio, molto meno caustica e molto più efficace. In ogni caso sono consigliati guanti in latice. È sufficiente versare un cucchiaino di granuli (circa 20 grammi) in una vaschetta (e) contenente mezzo litro di acqua tiepida (50° C) per ottenere una soluzione sufficiente a sviluppare in pochi secondi una basetta di 100 x 160 mm. Muovere di tanto in tanto la basetta nella vaschetta e, quando l'immagine sviluppata del circuito appare ben definita, si può rimuovere e sciacquare la basetta.
Una volta sviluppata, la basetta va sottoposta al processo di incisione, che avviene in una soluzione di cloruro ferrico versato in una vaschetta diversa da quella usata per lo sviluppo. Sono disponibili confezioni in granuli da sciogliere in acqua al 50%, oppure in bottiglie di soluzione liquida pronta all'uso (f). Essendo un prodotto altamente corrosivo e nocivo per l'ambiente, si consiglia di effettuare l'operazione nel garage o in un ambiente adatto. In alternativa, si può usare anche una soluzione di persolfato di sodio, disponibile in polvere solubile. Questo prodotto è però molto più pericoloso del cloruro ferrico e se ne consiglia l'uso solo dopo aver fatto un po' di esperienza.
È necessario comunque vestire guanti in gomma e occhiali protettivi, nonché indumenti da lavoro. Se oggetti metallici vengono a contatto con il cloruro ferrico, vanno lavati immediatamente con abbondante acqua. Evitare comunque il contatto diretto con la pelle o schizzi su qualsiasi oggetto. Un litro di soluzione di cloruro ferrico è sufficiente a corrodere la superficie di rame pari a un metro quadrato di basette, per cui, una volta usato, il prodotto può venire riutilizzato. Il tempo di corrosione del rame da parte del cloruro ferrico dipende da diversi fattori, primo fra tutti la temperatura ambiente. Per accelerare il processo di corrosione si può riscaldare la soluzione tramite apposite resistenze elettriche o agitare la vaschetta con moto ondulatorio a intervalli regolari. Usare sempre una pinza in plastica o anche le mani, purché siano protette da guanti in gomma.
Terminata la corrosione del rame, è necessario far gocciolare la basetta il più possibile e quindi lavare con abbondante acqua. Con una soluzione fresca di cloruro ferrico, il tempo per la corrosione si aggira intorno ai 45-60 minuti. Accertarsi di corrodere il più possibile le parti esposte, facendo attenzione però che la soluzione non intacchi anche le piste protette.
Prima di procedere alla foratura della basetta bisogna asportare l'emulsione dalle piste in rame con un batuffolo di cotone imbevuto con qualche goccia di acetone

(quello normalmente usato per togliere lo smalto alle unghie).
Il trapano deve essere possibilmente del tipo a colonna a velocità variabile (g). La velocità di rotazione consigliata è di 3 500 - 4 000 giri al minuto. Il diametro delle punte da utilizzare varia solitamente da 0,8 a 1 mm fino a 1,5 mm o anche 2 mm per i piedini dei componenti come potenziometri e morsetti. Per i fori delle viti di fissaggio o dei distanziatori di solito si usa la punta da 3 mm.

Attrezzatura essenziale

Per adeguare il laboratorio di maker a tutte le situazioni di lavoro, ecco un possibile elenco di attrezzatura essenziale.

- Cassettiere per componenti elettronici.
- Set di cacciaviti a stella e a taglio.
- Forbici da elettricista.
- Tronchese.
- Pinza a becco.
- Pinza normale.
- Crimpatrice.
- Trapano a colonna.
- Seghetto alternativo o piccolo banco sega.
- Piccola levigatrice da banco o orbitale.
- Set di chiavi inglesi.
- Set di chiavi a brugola.
- Minuteria varia (viti, dadi ecc.).

Stampante 3D

Dulcis in fundo, per un maker totalmente autonomo potrebbe essere utile in laboratorio anche una stampante 3D, da usare per la prototipazione rapida di oggetti in plastica. Avere a disposizione nel proprio laboratorio una stampante 3D che possa stampare un contenitore, una cornice, dei bracci e così via, è un’opzione di cui

difficilmente si può fare a meno. È sufficiente anche una piccola stampante con area di stampa 20×20 cm di qualsiasi tipo. La Figura 1.75 illustra una buona stampante cartesiana prodotta da Wanhao e dal costo accessibile. Ecco le sue caratteristiche principali.

- Interfaccia: SD o cavo USB.
- Dimensione massima di stampa: 200×200×185 mm.
- Risoluzione di posizionamento sul piano XY: 0,009 mm.
- Velocità massima assi XY: 180 000 mm/min.
- Risoluzione di posizionamento asse Z: 0,0025 mm.
- Velocità massima asse Z: 1000 mm/min.
- Ugello fornito: 0,4 mm.
- Piano riscaldato gestito elettronicamente da programma.
- Temperatura massima piano di stampa: 115°C.
- Temperatura massima ugello: 260°C.
- File formato STL per il software di slicing Wanhao-Cura.
- File G-Code per la stampante.
- Sistemi operativi supportati: Windows, Mac OSX, Linux.
- Certificazione CE.

Figura 1.75 La stampante 3D Wanhao Duplicator 6.

Schede hardware

La rivoluzione IoT iniziata qualche anno fa ha stimolato la produzione industriale di mini computer, microcontroller e relativi accessori. Non potendo esaurire nel poco spazio disponibile l’enorme quantità di prodotti disponibili in commercio, ci soffermeremo solo su quelli ritenuti più importanti. Senza nulla togliere a tutti gli altri, ne diamo solo una breve descrizione nel paragrafo “Altre schede”.

Arduino

Dal 2005, anno della sua nascita, a oggi, Arduino si è imposto a livello mondiale come una vera e propria piattaforma di sviluppo hardware e software per la robotica, la meccatronica, la domotica e moltissime applicazioni IoT. Non solo, il suo successo planetario ha indotto la diffusione di moltissimi prodotti di terze parti compatibili con la piattaforma. Tanto che la stessa Microsoft nel 2015 ha costituito una partnership con Arduino per creare una versione di Windows 10 che potesse essere impiegata anche in dispositivi smart, combinando l'hardware di Arduino con le risorse software di Windows 10, diventando così il primo sistema operativo "Arduino Certified".

Arduino UNO

Dei molti prodotti Arduino, la scheda *Arduino UNO* è senza dubbio quella di riferimento, oltre a essere tuttora quella più venduta e la più clonata al mondo.

Si tratta di una scheda basata sul microcontrollore Atmel ATmega328P (ora Microchip).

Dispone di 14 pin digitali di ingresso/uscita, di cui 6 possono essere utilizzati come uscite PWM, 6 ingressi analogici, un quarzo a 16 MHz, una connessione USB, una presa di alimentazione, un header ICSP e un pulsante di reset.

Le due file di pin hanno una spaziatura standard di 2,54 millimetri.

Per il suo funzionamento, basta semplicemente collegarla a un computer con un cavo USB. Una volta programmata, la scheda può essere alimentata anche con un adattatore o una pila a 9 volt. La Figura 2.1 illustra la scheda Arduino UNO. Ecco le sue caratteristiche principali.

- Microcontroller: ATmega328P.
- Tensione di funzionamento: 5 V.
- Tensione in ingresso (consigliato): 7-12 V.
- Pin digitali I/O: 14 (di cui 6 forniscono PWM).
- Pin digitali PWM: 6.
- Pin di ingresso analogico: 6 (tramite convertitore analogico/digitale ADC a 10 bit)..
- Interfaccia seriale UART.
- Interfaccia I^2C.
- Interfaccia SPI.
- Corrente DC per i pin I/O: 20 mA.
- Corrente DC per il pin 3,3 V: 50 mA.
- Memoria Flash: 32 KB (ATmega328P) di cui 0,5 KB utilizzato dal bootloader.
- SRAM: 2 KB (ATmega328P).
- EEPROM: 1 KB (ATmega328P).
- Frequenza di clock: 16 MHz
- Lunghezza × larghezza: 68,6 × 53,4 mm

La scheda hardware viene programmata tramite un'interfaccia integrata di sviluppo, altrimenti detta IDE (*Integrated Development Environment*) che viene costantemente aggiornata e distribuita gratuitamente dal sito ufficiale: `http://arduino.cc`. Per i dettagli sull'IDE di Arduino si veda il Capitolo 3.

Pin analogici e digitali

La scheda Arduino UNO può gestire segnali in ingresso e in uscita che possono essere analogici o digitali.

Un segnale analogico può assumere un qualsiasi valore, all'interno dell'intervallo di tensione 0-5 volt. I pin di ingresso analogico della

scheda sono 6 (numerati da A0 ad A5), con una risoluzione di 10 bit (2^{10}), quindi il segnale campionato in ogni ingresso va da 0 a 1023.

Un segnale digitale può assumere due soli stati (High e Low ovvero 1 e 0), corrispondenti ai due livelli di tensione convenzionali, cioè 5 volt e 0 volt. I pin di ingresso e di uscita digitale della scheda sono 14 (numerati da 0 a 13), con una risoluzione di 8 bit (2^{10}).

I 6 pin digitali , contrassegnati da una tilde (~), possono essere usati in uscita come PWM (*Pulse Width Modulation*), per cui si ha la possibilità di scrivere su ogni pin un byte da 0 a 255.

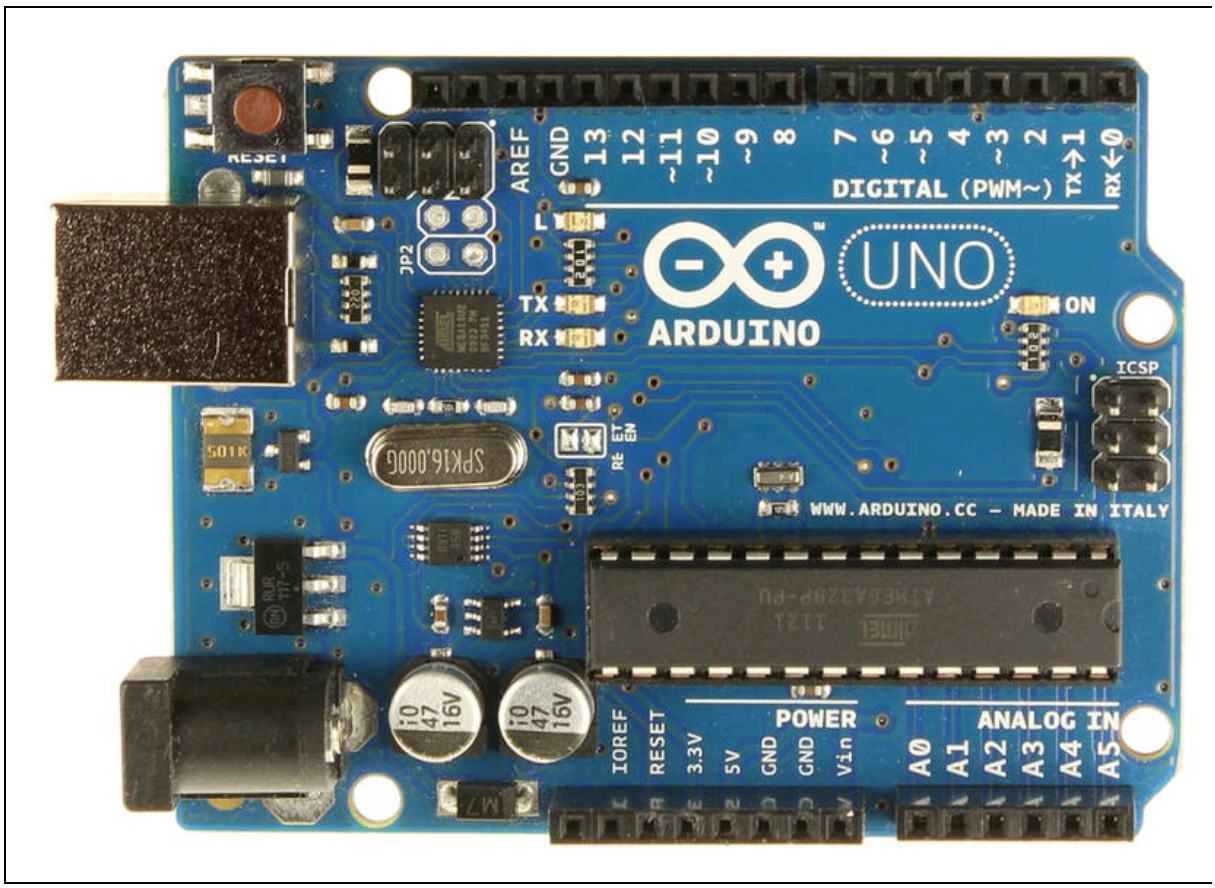


Figura 2.1 La scheda Arduino UNO.

Perché Arduino possa riconoscere se sul pin di riferimento è presente un segnale di ingresso o uscita, i pin digitali devono essere preventivamente impostati in modalità INPUT o OUTPUT.

Un’esemplificazione della lettura dei segnali analogici e della lettura/scrittura dei segnali digitali è illustrata nella Figura 2.2a. Il funzionamento della tecnologia PWM è illustrato nella Figura 2.2b.

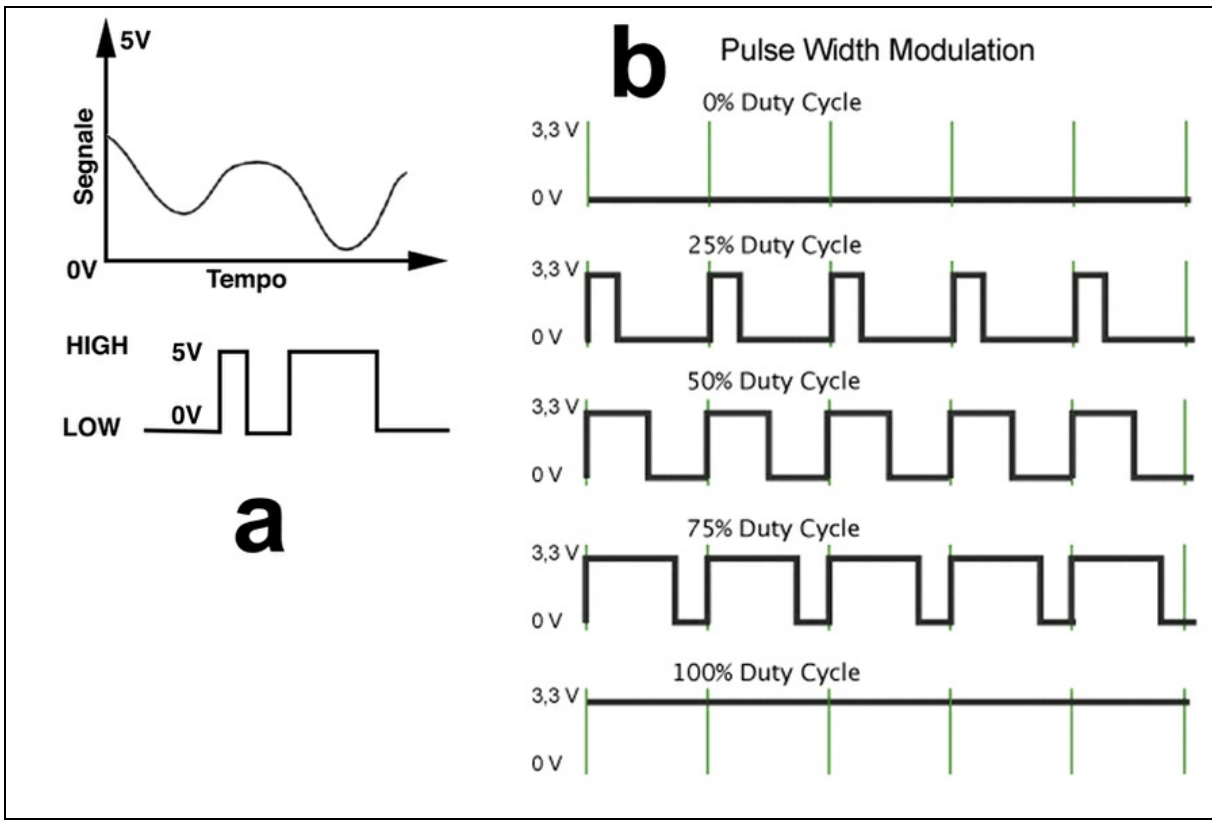


Figura 2.2 Lettura dei segnali analogici e della lettura/scrittura dei segnali digitali.

Bus di comunicazione

Arduino UNO è dotato di bus di comunicazione molto importanti, disponibili sui pin digitali e/o analogici.

- I^2C: pin digitali 16 (SDA) e 17 (SCL) oppure pin analogici A4 e A5.
- UART: pin digitali 0 (RX) e 1 (TX).
- SPI: pin digitali 13 (SCK), 12 (MISO), 11 (MOSI), 10 (reset).

Per la spiegazione di questi bus si veda più avanti.

Raspberry Pi

Dopo il successo planetario del primo modello di Raspberry Pi, presentato il 29 febbraio 2012 come il computer più piccolo ed economico al mondo, si sono susseguiti vari modelli di Raspberry Pi, che hanno avuto lo stesso successo se non di più. Nel momento in cui scriviamo, i modelli disponibili e, secondo le informazioni del sito ufficiale, ancora in produzione sono i seguenti.

- Raspberry Pi 3 Model B.
- Raspberry Pi ZERO W.
- Raspberry Pi ZERO.
- Raspberry Pi 2 Model B.
- Raspberry Pi 1 Model B+.
- Raspberry Pi 1 Model A+.

Il recente modello Raspberry Pi 3 è ancora più potente del modello 2, che già era sei volte più potente del primo modello. È in grado di accogliere nella sua SD card più sistemi operativi contemporaneamente, grazie alla distribuzione NOOBS (*New Out Of the Box Software*) ed è dotato di connettività Wi-Fi e Bluetooth 4.1 e BLE. Questo è stato possibile cambiando il processore e aumentando la RAM. Ha sostituito il modello Raspberry Pi 2 Model B nel mese di febbraio 2016.

Ecco le sue caratteristiche principali:

- CPU 1.2GHz 64 bit quad-core ARMv8 CPU;
- 802.11n Wireless LAN;
- Bluetooth 4.1;
- Bluetooth Low Energy (BLE);
- 1 GB di RAM;
- 4 porte USB;
- porta GPIO a 40 pin;

- porta full HDMI;
- porta Ethernet;
- jack audio da 3,5 mm con video composito combinato;
- interfaccia per fotocamera (CSI);
- interfaccia display (DSI);
- slot per microSD (ora a inserimento senza molla);
- core grafico VideoCore IV 3D.

Grazie al processore ARMv8, è possibile eseguire l'intera gamma di distribuzioni ARM GNU/Linux, tra cui Ubuntu Mate e Snappy Core, così come il kernel di Microsoft Windows 10 IoT Core, OSMC, Librelec, Pinet e Risc OS.

La scheda Raspberry Pi 3 ha un fattore di forma identico a quello del precedente Pi 2 ed è totalmente compatibile anche con Raspberry Pi prima versione (Figura 2.3).

Per news, blog, download e risorse online, il sito ufficiale è: https://www.raspberrypi.org. Per approfondire l'uso di Raspberry Pi, si consiglia la lettura del libro Raspberry Pi (seconda edizione), Edizioni Apogeo 2015.

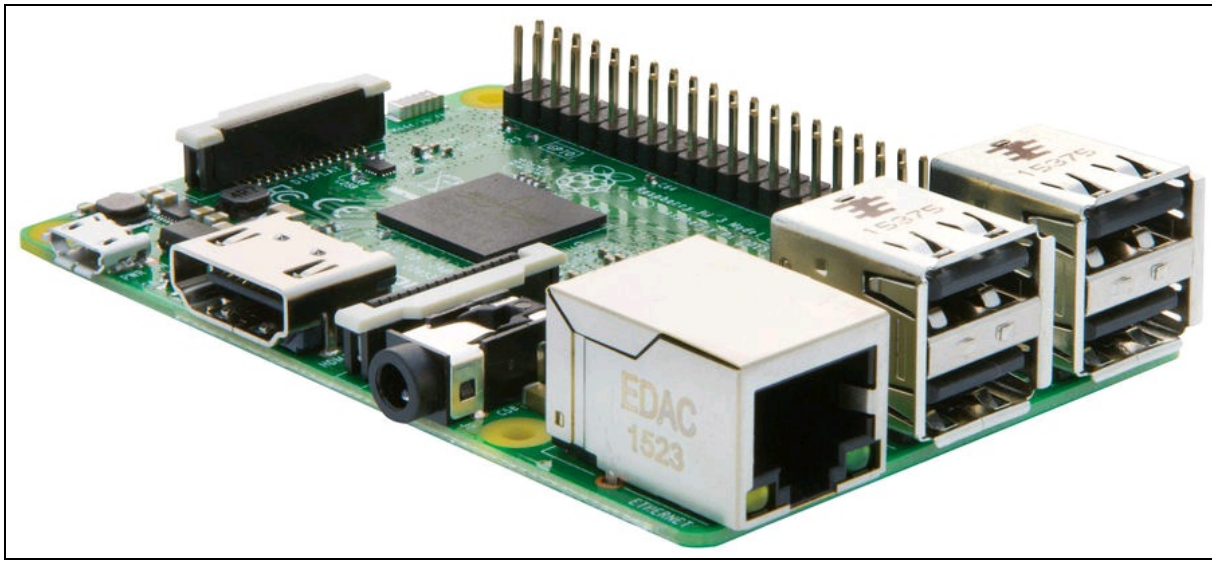


Figura 2.3 La scheda Raspberry Pi 3.

La porta GPIO

L’elemento che rende davvero interessante l’uso di un computer come Raspberry Pi è la sua porta GPIO. L’interfaccia *General Purpose Input Output*, altrimenti nota come porta GPIO, può avere una o più connessioni digitali di ingresso e uscita. Gli ingressi (*input*) leggono i segnali digitali dall’esterno mentre le uscite (*output*) possono controllare o mandare segnali digitali a dispositivi esterni.

Fin dal suo esordio, Raspberry Pi ha avuto diverse versioni della porta GPIO. Nelle schede ormai fuori produzione della prima versione delle schede Raspberry Pi la porta GPIO era dotata di 26 pin.

L’attuale produzione dei modelli Raspberry Pi 3, Raspberry Pi 2, Model B+ e Model A+ e Raspberry Pi Zero/W prevede una porta GPIO con 40 pin (Figura 2.4), compatibile nei primi 26 pin con la piedinatura della “vecchia” porta Rev. 2.

Per dovere di cronaca, dobbiamo anche dire che la vecchia porta ha subito due revisioni (Rev. 1 e Rev. 2) con qualche differenza nella piedinatura, per cui alcuni progetti elettronici potrebbero non essere perfettamente compatibili con le schede obsolete.

NOTA

Nel libro si farà riferimento esclusivamente alla porta GPIO a 40 pin della scheda Raspberry Pi 2/3 e ZERO/W.

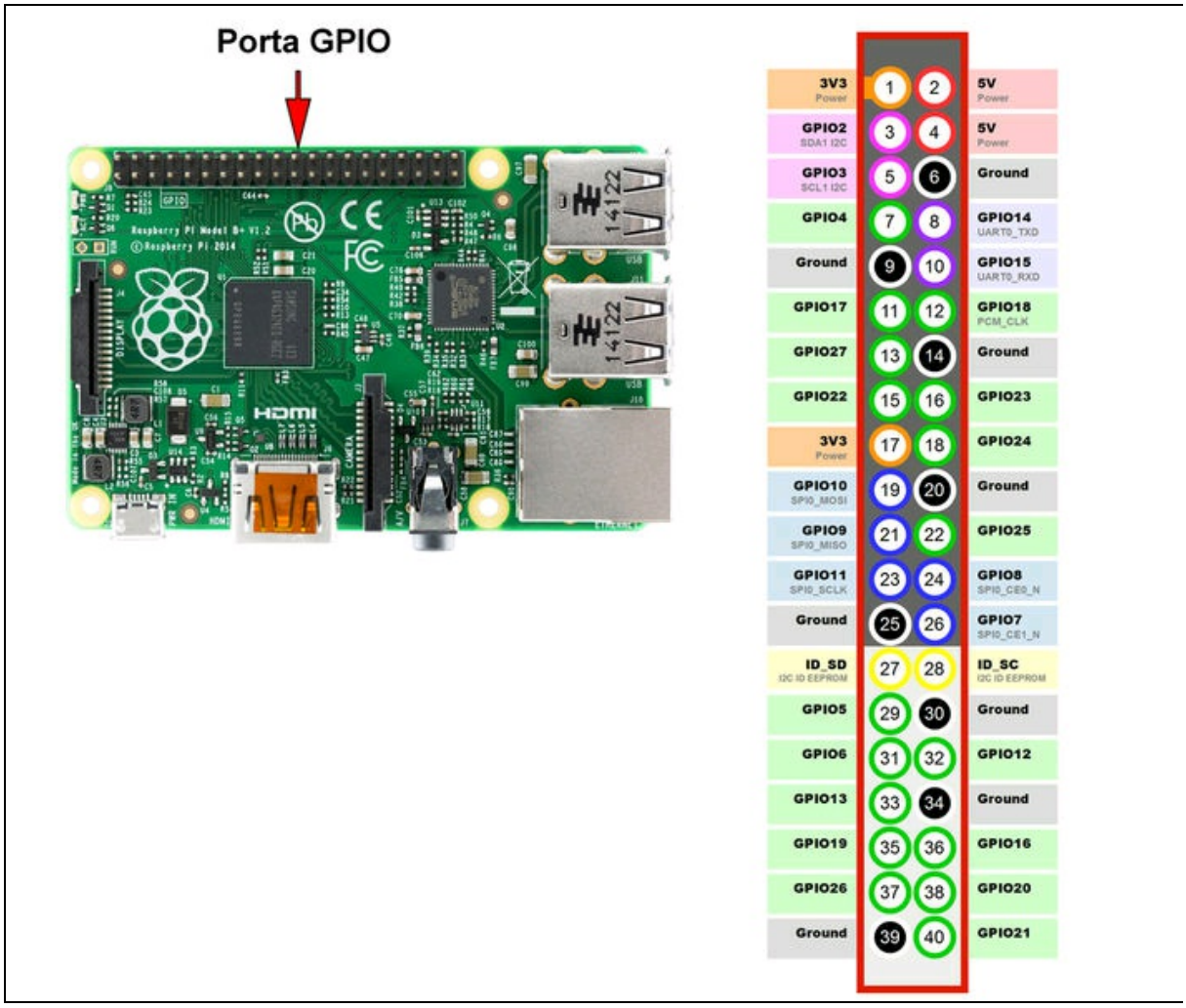


Figura 2.4 Piedinatura della porta GPIO.

Piedinatura della porta GPIO

Guardando la scheda dal lato componenti con la serigrafia del logo dritta, la porta GPIO si trova in alto a sinistra. È dotata di 40 piedini maschi su due file di 20, con spaziatura standard di 2,54 millimetri.

Tale distanza è molto utile, perché permette di collegare facilmente cavi flat dotati di connettori femmina standard, come quelli usati normalmente nei computer, o di basette millefori a passo standard.

Risulta facile anche la connessione di cavi femmina per breadboard che possono venire infilati senza problemi di spaziatura.

È estremamente importante conoscere la piedinatura della porta per effettuare i collegamenti corretti con cavi flat, basette o cavi per breadboard o con circuiti di terze parti.

È da notare che la disposizione dei piedini è numerata su due file alternate e non su una fila alla volta. Si consiglia di studiare bene la disposizione delle porte GPIO rispetto alla numerazione dei piedini fisici, per non confondere il numero di porta GPIO con il numero del piedino. Questo perché si troveranno spesso i riferimenti al numero del pin della porta GPIO e non al numero di piedino.

Attenzione! Come per tutti i circuiti elettronici, anche per la porta GPIO è necessario prestare un minimo di attenzione. Ecco alcune semplici raccomandazioni di buon senso:

- non toccare con le dita i piedini della porta;
- non applicare tensioni superiori a 3,3 volt ai piedini;
- non cortocircuitare i piedini, specie quelli dell'alimentazione e accertarsi di conoscere bene la piedinatura della breadboard prima di collegare i cavetti;
- usare solo cavetti con attacco femmina;
- non saldare fili o componenti ai piedini.

Livello logico

La porta GPIO fornisce un'alimentazione di 5 V sui piedini 2 e 4, ma questo è solo un collegamento diretto proveniente dalla presa micro-USB.

Bisogna sempre tenere presente che i circuiti interni delle schede Raspberry Pi si basano su livelli logici TTL a 3,3 V e che tutti i componenti interni lavorano con alimentazione 3,3 V.

Quando si progetta un circuito da collegare alla porta GPIO, è necessario usare componenti compatibili con il livello logico TTL di 3,3

V.

Si dicono livelli logici i numeri binari 0 e 1 che corrispondono in elettronica ai livelli di tensione usati nei circuiti digitali, ovvero ai livelli di tensione di 0 volt e 5 volt.

Esiste una certa tolleranza nei valori di tensione, per cui un livello di tensione compreso fra 0 e 2 volt rappresenta il livello logico 0, mentre una tensione tra 3 e 5 volt rappresenta il livello logico 1. Per questo motivo, il livello di 3,3 volt della porta GPIO può essere compatibile con un circuito slave a livello logico 5 volt, per esempio, una scheda o uno shield Arduino.

Tecnologia	Tensione livello 0	Tensione livello 1
CMOS	0~2 V	2~18 V
TTL	0~0,8 V	2~5,25 V max

ATTENZIONE

Se si collega una tensione di 5 V a un piedino qualsiasi della porta GPIO si danneggia irrimediabilmente la scheda. Si consiglia di fare molta attenzione quando si usa il piedino 2 o il piedino 4 (5 V) per alimentare circuiti esterni quando questi vengono collegati direttamente ai piedini della porta GPIO con livello di tensione TTL a 3,3 V.

Bus della porta GPIO

Oltre alla gestione delle funzioni normali GPIO di ingresso e uscita digitali, alcuni piedini sono dedicati a particolari bus:

- bus seriale UART;
- bus seriale I^2C;
- bus seriale SPI.

La Figura 2.3 illustra la configurazione dei piedini della porta GPIO dedicati ai suddetti bus. La scritta GEN accanto alle porte GPIO significa *General Purpose* (uso generico). La Tabella 2.1 riassume il nome e il significato di tutte le porte.

Si tenga presente che le porte GPIO dedicate alle interfacce non possono essere usate per scopi generici.

Si invita a fare riferimento alla Figura 2.5 e alla Tabella 2.1 durante la progettazione dei circuiti elettronici. A questo proposito si consiglia vivamente di fare una stampa della piedinatura di base e quella dedicata ai bus e di tenerle sempre a portata di mano.

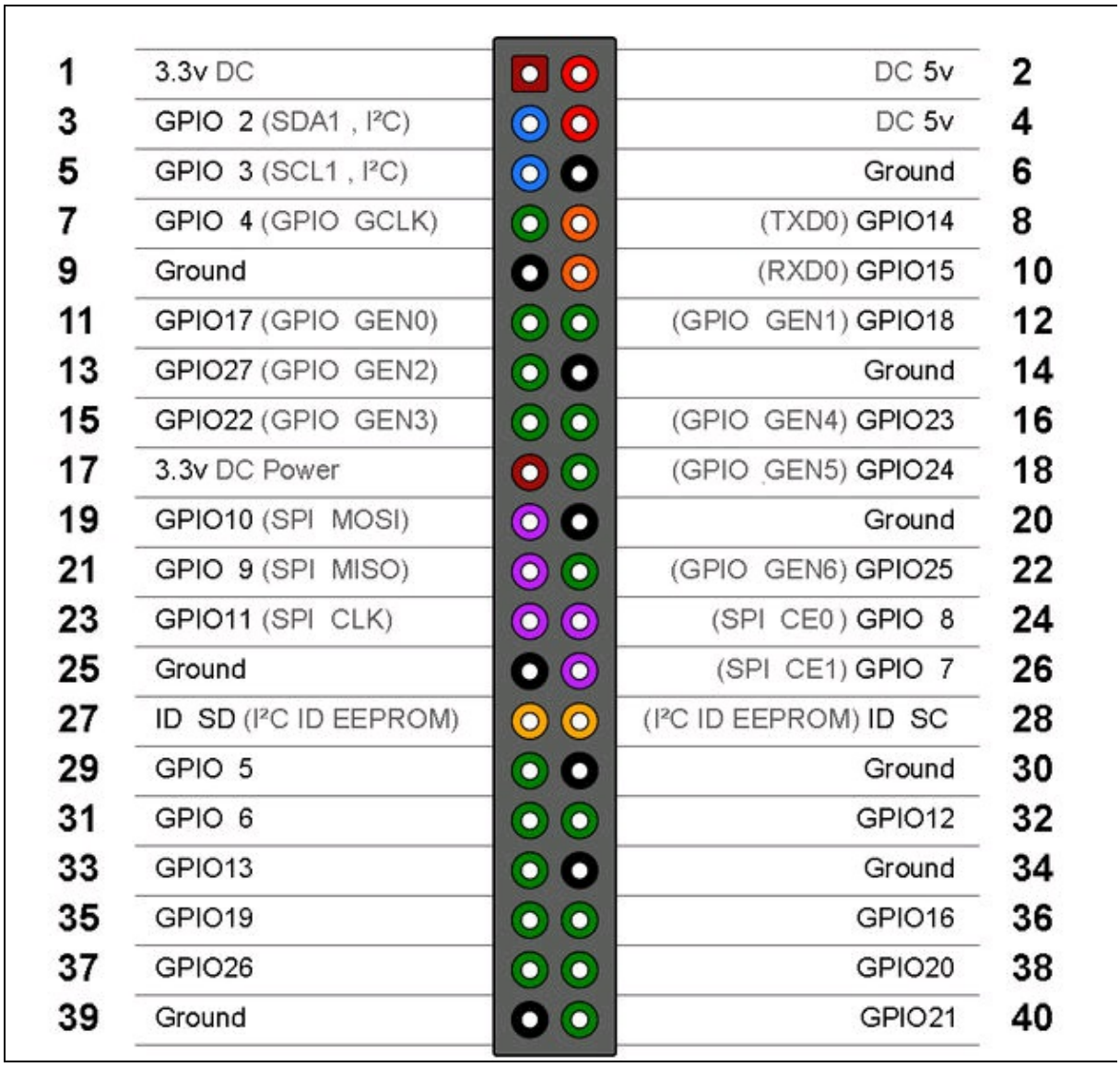


Figura 2.5 Configurazione dei piedini dei bus della porta GPIO.

Tabella 2.1 Piedinatura della porta GPIO.

Piedino	Nome della porta	Descrizione

1	+3,3 V	Alimentazione
2	+5 V	Alimentazione
3	GPIO 2	SDA1 I²C
4	+5 V	Alimentazione
5	GPIO 3	SCL1 I²C
6	GND	Massa
7	GPIO 4	GCLK
8	GPIO 14	TX
9	GND	Massa
10	GPIO 15	RX
11	GPIO 17	GEN 0
12	GPIO 18	GEN 1
13	GPIO 27	GEN 2
14	GND	Massa
15	GPIO 22	GEN 3
16	GPIO 23	GEN 4
17	+3,3 V	Alimentazione
18	GPIO 24	GEN 5
19	GPIO 10	SPI MOSI
20	GND	Massa
21	GPIO 9	SPI MISO
22	GPIO 25	GEN 6
23	GPIO 11	SPI CLK
24	GPIO 8	SPI CE0
25	GND	Massa
26	GPIO 7	SPI CE1
27	ID SD	I²C ID EEPROM
28	ID SC	I²C ID EEPROM
29	GPIO 5	
30	GND	Massa
31	GPIO 6	
32	GPIO 12	
33	GPIO 13	

34	GND	Massa
35	GPIO 19	
36	GPIO 16	
37	GPIO 26	
38	GPIO 20	
39	GND	Massa
40	GPIO 21	

Bus UART

Il bus UART (*Universal Asynchronous Receiver/Transmitter*) fornisce una semplice interfaccia seriale. Il collegamento avviene tramite due piedini di ricezione e trasmissione dei dati seriali.

- Trasmissione dati (TX): piedino 8.
- Ricezione dati (RX): piedino 10.

Benché il bus UART possa gestire qualsiasi velocità, si suggerisce di usare valori standard:

- 9600
- 14400
- 19200
- 28800
- 38400
- 56000
- 57600
- 115200

Bus I²C

La sigla si legge “i due ci” o “i quadro c” e sta per *Inter-Integrated Circuit*. Si tratta di un bus sincrono, progettato per fornire

comunicazioni veloci tra più circuiti integrati (IC).

Il bus I²C è composto da un circuito *master* e da un circuito *slave*. Normalmente viene stabilita la connessione tramite una linea di dati e una di clock:

- SDA (*Serial DAta line*) per i dati;
- SCL (*Serial CLock line*) per il clock.

Il bus I²C è accessibile tramite i piedini 3 e 5:

- il piedino 3 fornisce la linea seriale di dati (SDA);
- il piedino 5 fornisce il clock seriale del segnale (SCL).

È possibile collegare in cascata fino a 127 dispositivi I²C, collegandoli semplicemente in parallelo a un dispositivo master (Figura 2.6). Di solito, la gestione degli ID dei dispositivi viene effettuata via software.

Per l'accesso al bus I²C, sono disponibili circuiti esterni appositamente progettati per la trasmissione e la ricezione dei dati, compreso lo stesso Arduino.

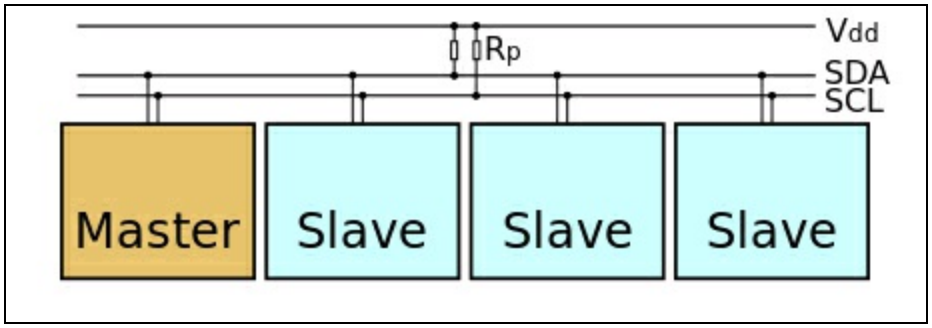


Figura 2.6 Collegamento in cascata di dispositivi I²C.

Bus SPI

Il bus SPI (*Serial Peripheral Interface*) è un bus seriale sincrono progettato principalmente per la programmazione ISP (*In-System Programming*) di microprocessori e altri dispositivi. I microprocessori

che adottano la tecnologia ISP possono venire programmati mentre sono già installati in un sistema. In questo modo, il bus SPI, essendo dotato di quattro collegamenti, consente di comunicare con più di un dispositivo di destinazione, in base a una logica master-slave (Figura 2.7).
Raspberry Pi è dotato di due interfacce SPI. Si noti che viene usata la numerazione fisica dei piedini e non la corrispondenza dei numeri della porta GPIO.

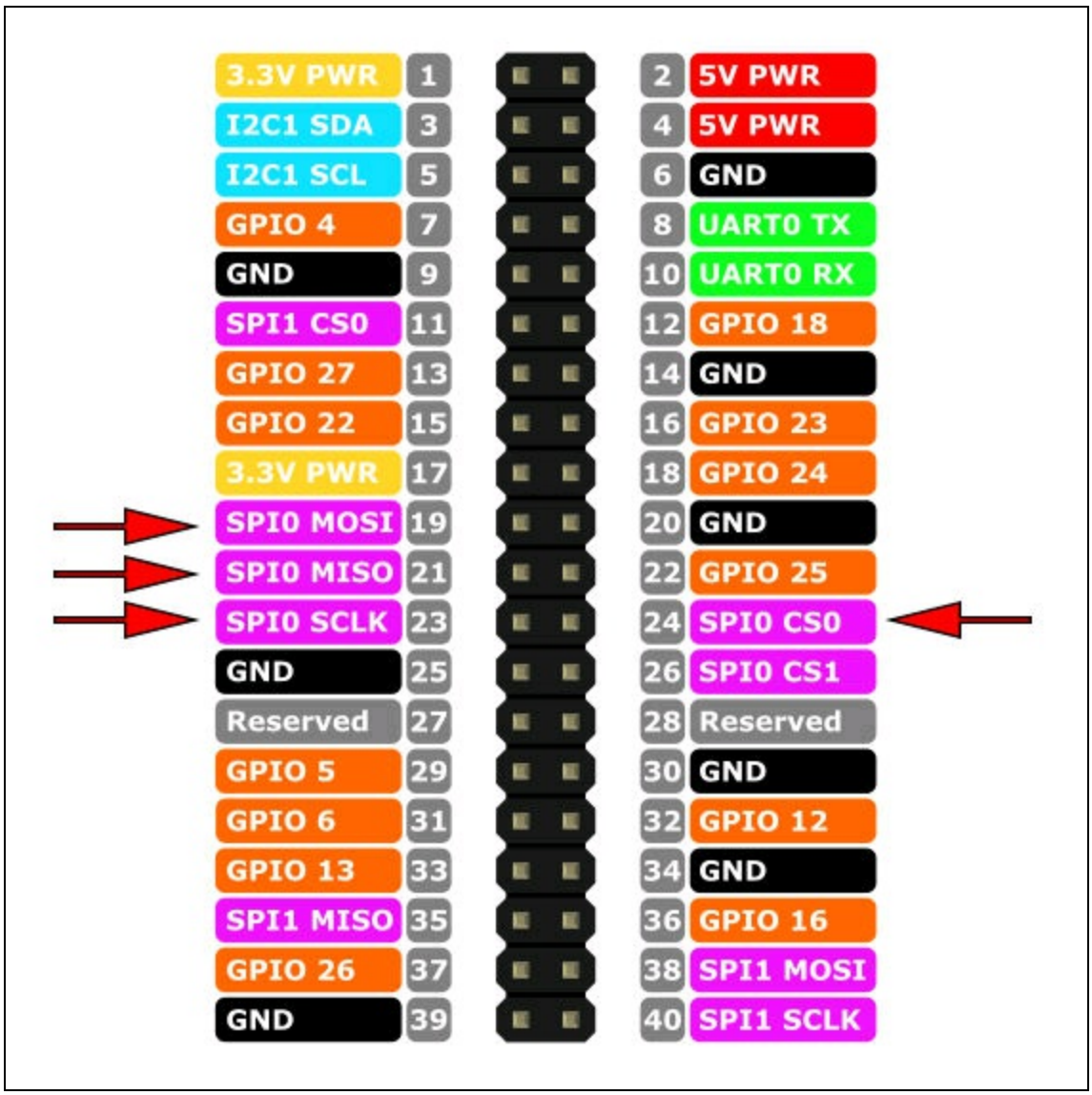


Figura 2.7 Collegamento in cascata di dispositivi SPI.

Il primo bus SPI è disponibile sui seguenti piedini.

- SPI0 MOSI (*Master Output Slave Input*): piedino 19.
- SPI0 MISO (*Master Input Slave Output*): piedino 21.
- SPI0 SLCK (*Serial Clock*): piedino 23.
- SPI0 CS0 (*Chip Select 0*): piedino 24.

Per maggiore chiarezza:

- il piedino 19 fornisce l'uscita SPI Master (MOSI);
- il piedino 21 fornisce l'ingresso SPI Master (MISO);
- il piedino 23 fornisce il clock per sincronizzare la comunicazione;
- il piedino 24 serve alla selezione (*chip select*) del dispositivo *Slave*.

Il secondo bus SPI è disponibile sui seguenti piedini.

- SPI1 MOSI (*Master Output Slave Input*): piedino 38.
- SPI1MISO (*Master Input Slave Output*): piedino 35.
- SPI1SLCK (*Serial Clock*): piedino 40.
- SPI1CS1 (*Chip Select 1*): piedino 26.

Per maggiore chiarezza:

- il piedino 38 fornisce l'uscita SPI Master (MOSI);
- il piedino 35 fornisce l'ingresso SPI Master (MISO);
- il piedino 40 fornisce il clock per sincronizzare la comunicazione;
- il piedino 26 serve alla selezione (*chip select*) del dispositivo *Slave*.

Nella Figura 2.8 sono indicati i piedini usati per il primo bus SPI che normalmente viene usato nei progetti. Il secondo bus SPI non viene usato.

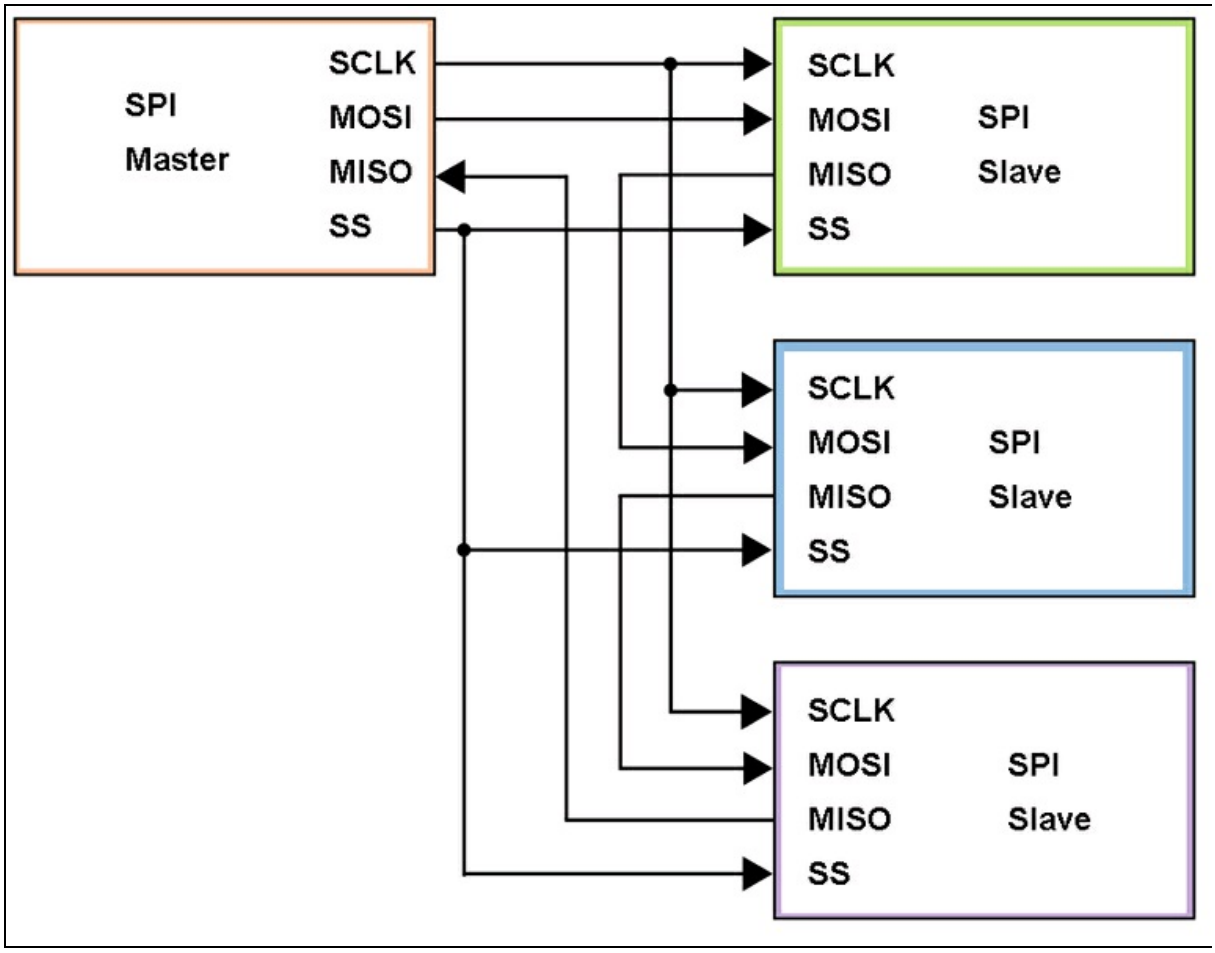


Figura 2.8 Il bus SPI usato nei progetti.

Uso della porta GPIO con Windows 10 IoT Core

Alla porta GPIO e alle sue interfacce (I²C, SPI e UART) si possono connettere componenti elettronici attivi e passivi, circuiti integrati, sensori, display, pulsanti, così come altre schede o altri computer.

Nei progetti di questo libro, verrà usata la libreria GPIO appositamente progettata da Microsoft, che viene installata con il pacchetto Windows 10 IoT Core per Raspberry Pi. Si veda il Capitolo 3 per i dettagli.

Python

Raspberry Pi è nato principalmente con lo scopo di essere programmato con Python, come ricorda la parola *Pi* nel nome. Per i dettagli di programmazione con Python si veda il Capitolo 3.

Modulo ESP8266

Il modulo ESP8266 è un chip Wi-Fi a basso costo con stack completo TCP/IP e capacità MCU (cioè di microcontrollore) prodotto da Espressif Systems di Shanghai (https://espressif.com).

Il modulo Wi-Fi è un microcontrollore da 80 MHz che fornisce accesso a una rete Wi-Fi e dispone di front-end con funzioni (cioè sia di client sia di access point) e supporto DNS. Il chip è venuto all'attenzione dei produttori occidentali nell'agosto 2014 con il modello ESP-01, realizzato da un produttore di terze parti, AI-Thinker (Figura 2.9a).

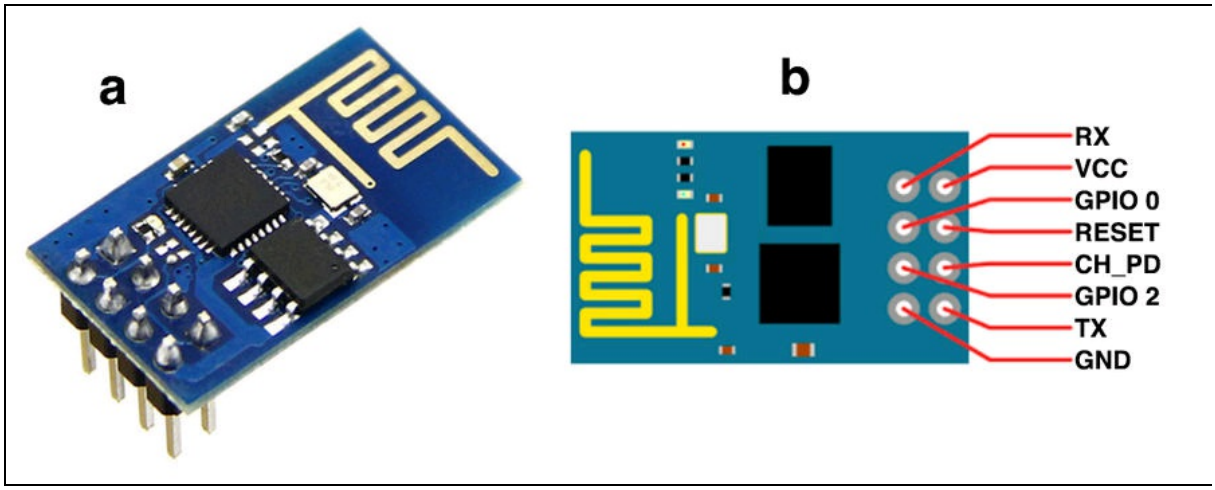


Figura 2.9 Modulo Wi-Fi ESP8266-01 (a) e piedinatura (b).

Questo piccolo modulo consente a microcontrollori esterni (per esempio, Arduino o Raspberry Pi) di connettersi a una rete Wi-Fi e di eseguire semplici connessioni TCP/IP utilizzando comandi AT/Hayes.

Il modulo funziona anche come unità autonoma per la connessione Wi-Fi, per cui non ha bisogno di microcontroller esterni. Dato che il primo modello ESP-01 è dotato solo di un pin GPIO per la comunicazione con sensori e/o attuatori, diventa quasi obbligatorio l'uso di un controller esterno per aumentare le porte digitali e analogiche.

Ecco la piedinatura standard del modello ESP8266-01 (abbreviato come ESP-01).

Pin	Nome	Descrizione
1	GND	Ground
2	U0TXD	UART TX
3	GPIO2	Possiede pull-up interno
4	CHIP_EN	Chip Enable, HIGH = attivo
5	GPIO0	Possiede pull-up interno
6	EXT_RSTB	Reset esterno, LOW= attivo
7	U0RXD	UART RX, possiede pull-up interno
8	VDD	Alimentazione +3,3 V

Stranamente, il modulo è nato un po' in sordina e senza documentazione in inglese. Per fortuna, il prezzo molto basso e il fatto di avere pochissimi componenti esterni hanno spinto molti hacker a esplorare il modulo, il chip e il firmware, oltre a tradurre la documentazione cinese.

Oggi esiste una vasta documentazione online presso il repository https://github.com/esp8266 e presso la community molto attiva presso: http://www.esp8266.com.

Caratteristiche principali.

- Wi-Fi 802.11 b/g/n.
- Velocità di trasmissione seriale UART: 115200 bps.
- Protocollo integrato protocollo TCP/IP.
- Alimentazione: 3,3.
- Tolleranza della tensione di I/O: 3,6 V Max.
- Corrente di funzionamento normale: ~ 70mA
- Picco di corrente di funzionamento: ~ 300mA.
- Corrente di dispersione corrente: <10µA.
- Sensibilità: + 19,5dBm in modalità 802.11b.
- Memoria flash: 1MB (8Mbit).

- Modalità di protezione Wi-Fi: WPA, WPA2.
- Dimensioni del modulo: 24,75 × 14,5 mm.

Altri modelli di ESP8266

Dal suo esordio a oggi sono stati prodotti altri modelli di ESP8266 con caratteristiche migliorate e, salvo modelli particolari, con un numero maggiore di pin (alimentazione, UART e GPIO). La Figura 2.10 ne illustra alcuni esempi. A tutt'oggi, i modelli di ESP8266 disponibili in commercio sono i seguenti (fonte http://www.esp8266.com).

Modello	Pin
ESP-01	8
ESP-02	8
ESP-03	14
ESP-04	14
ESP-05	5
ESP-06	12
ESP-07	16
ESP-07S	16
ESP-08	16
ESP-09	12
ESP-10	5
ESP-11	8
ESP-12	16
ESP-12-F	22
ESP-12-E	22
ESP-12S	16
ESP-13	18
ESP-14	22

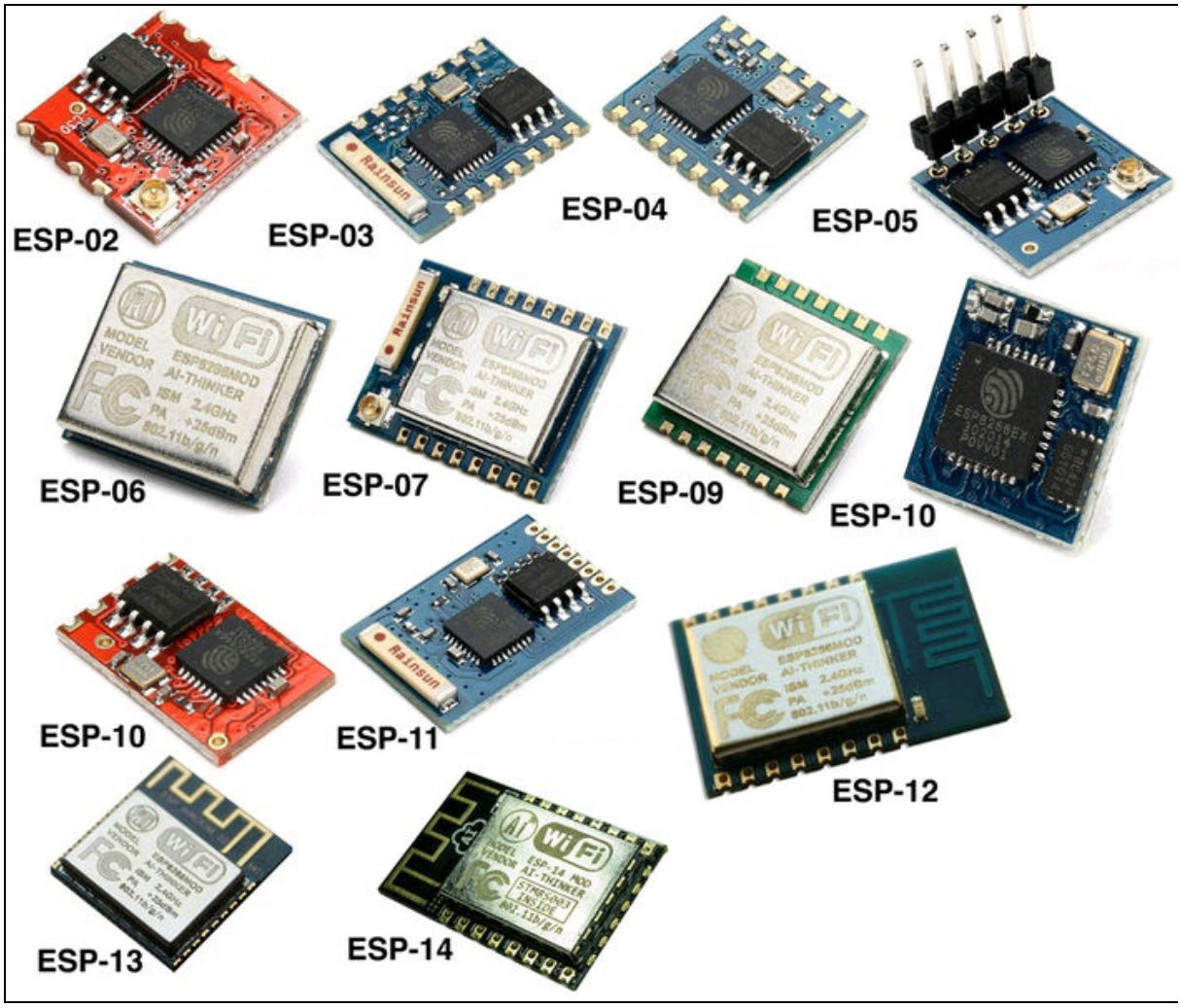


Figura 2.10 I diversi modelli di ESP8266.

Altri prodotti con ESP8266

Molti produttori si sono dati da fare per integrare i vari moduli ESP8266 nei loro prodotti e, addirittura, sono nati nuovi fabbricanti di schede con ESP8266.

Ecco alcuni esempi (in ordine alfabetico e non di importanza):

- Adafruit Huzzah ESP8266;
- Arduino UNO Wi-Fi;
- ESPert ESPresso Lite;

- ESPert ESPresso Lite V2.0;
- ESPino;
- In-Circuit ESP-ADC;
- Knewron Technologies smartWIFI;
- NodeMCU DEVKIT;
- Olimex MOD-WIFI-ESP8266;
- Olimex MOD-WIFI-ESP8266-DEV;
- SparkFun ESP8266 Thing;
- SweetPea ESP-210;
- Watterott ESP-WROOM02;
- WeMos D1;
- WeMos D1 Mini;
- WeMos D1 Pro Mini;
- WeMos D1 R2.

Vediamo in breve le caratteristiche principali di alcune di queste schede.

Adafruit Huzzah ESP8266

La Figura 2.11a mostra il modulo Huzzah ESP8266 proposto dal colosso dell'elettronica per maker Adafruit. Viene venduto in kit con la scheda breakout con ESP-12 e i pin saldabili a parte (https://www.adafruit.com/product/2471). La scheda è pre-programmata con NodeMCU (interprete Lua).

Arduino UNO WiFi

Anche Arduino non ha potuto resistere al fascino del modulo ESP8266. Lo ha montato nel suo prodotto di punta Arduino UNO (Figura 2.11b).

La scheda Arduino UNO WiFi è a tutti gli effetti una normale scheda Arduino UNO con modulo Wi-Fi integrato ESP8266 che, oltre alla

connettività Wi-Fi, permette anche la programmazione OTA (*Over The Air*), sia per il trasferimento degli sketch di Arduino sia per il firmware Wi-Fi per il modulo ESP8266.

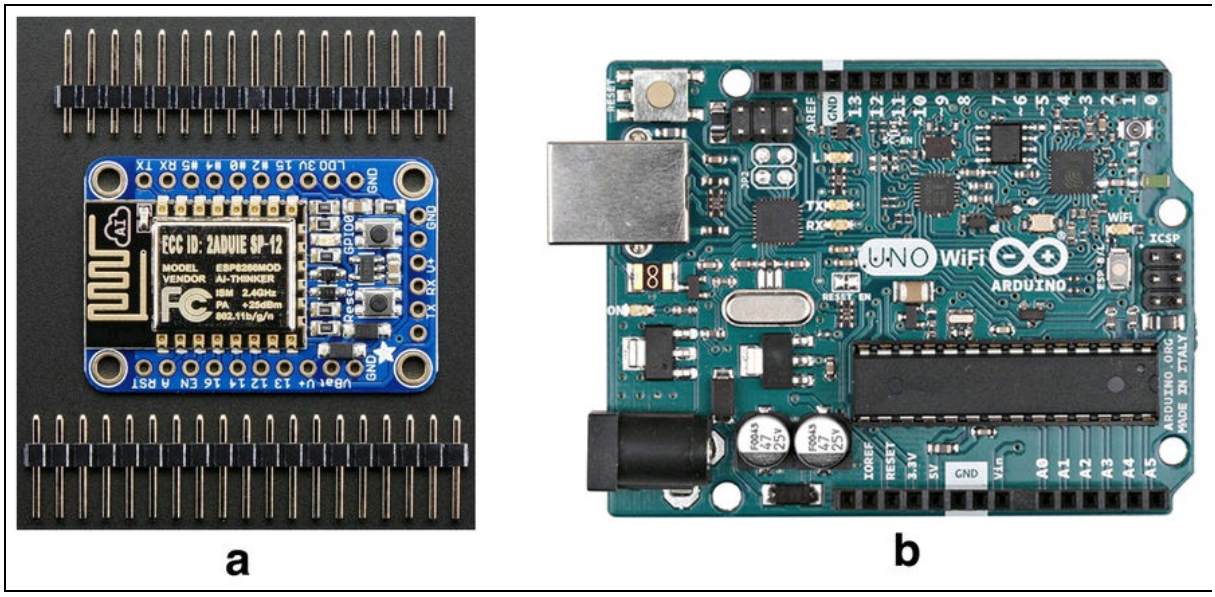


Figura 2.11 Adafruit Huzzah ESP8266 (a). Arduino UNO WiFi (b).

NodeMCU DEVKIT

NodeMCU è una piattaforma open source sviluppata specificatamente per il mondo IoT (http://www.nodemcu.com). Ha sviluppato un firmware che funziona sul modulo ESP8266 e una scheda hardware basata sul modulo ESP-12 (Figura 2.12a). La scheda NodeMCU è programmabile in Lua. È disponibile su molti store online, per esempio, Amazon.

Per la programmazione della scheda ESP8266 e dei prodotti di terze parti dotati di ESP8266, si può usare l'IDE di Arduino. Per i dettagli si veda il Capitolo 3.

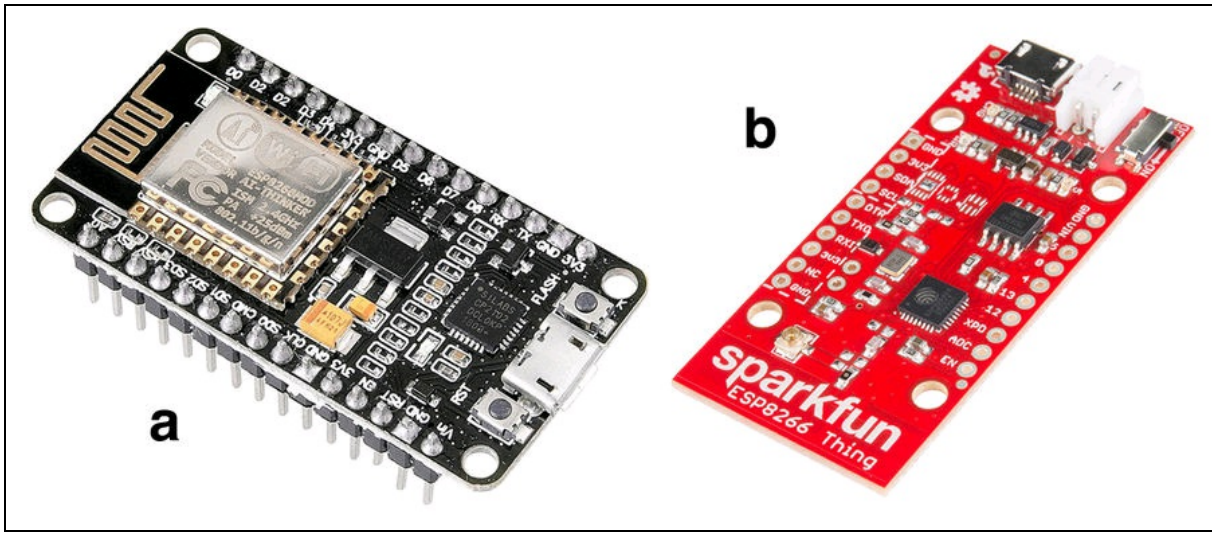


Figura 2.12 NodeMCU DEVKIT (a). SparkFun ESP8266 Thing(b).

SparkFun ESP8266 Thing

SparkFun ESP8266 Thing è una scheda breakout di sviluppo per ESP8266 (https://www.sparkfun.com/products/13231). Una caratteristica particolare è l'aggiunta di un caricatore e una presa LiPo, per poter utilizzare il prodotto in applicazioni remote (Figura 2.12b).

WeMos

I prodotti WeMos (https://www.wemos.cc) meritano un paragrafo dedicato. L'azienda cinese è nata qualche anno fa proponendo una scheda in tutto e per tutto simile ad Arduino UNO, con integrato un modulo ESP8266. In seguito, è stata commercializzata la scheda WeMos D1 R2, come seconda revisione. Oggi le schede disponibili sono molto più piccole e, pur mantenendo la totale compatibilità con Arduino, sono studiate per ospitare degli shield aggiuntivi.

La Figura 2.13a illustra il modello WeMos D1 mini pro, di cui descriviamo le caratteristiche principali.

- 11 pin di I/O digitali.

- Tecnologie: interrupt/PWM/I^2C/OneWire.
- 1 ingresso analogico (ingresso max 3,2 V).
- Flash: 16M byte (128M bit).
- Connettore per l’antenna esterna.
- Antenna in ceramica integrata.
- Nuovo circuito integrato CP2104 USB-TO-UART.
- Stessa dimensione del D1 mini, ma più leggero.

Gli shield WeMos disponibili sono i seguenti:

- WS2812B RGB Shield;
- Relay Shield;
- ProtoBoard;
- OLED Shield;
- Motor Shield;
- DHT Pro Shield;
- SHT30 Shield;
- DC Power Shield;
- Micro SD Card Shield.

Fra tutti, vogliamo evidenziare il piccolissimo OLED shield (Figura 2.13b) che permette una risoluzione di ben 64×48 pixel su una diagonale di soli 0,66 pollici. L’interfaccia di comunicazione è I^2C.

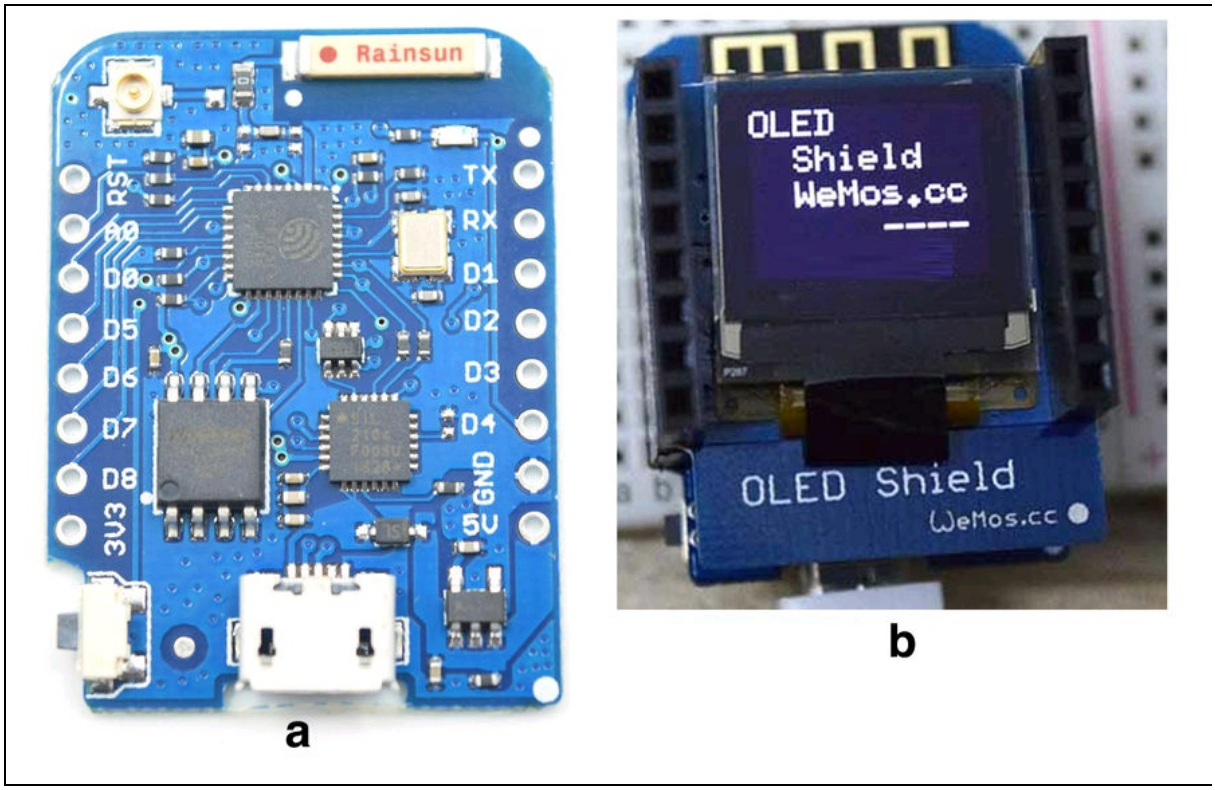


Figura 2.13 WeMos D1 mini pro (a). WeMos OLED shield (b).

ESP32

Dopo l'enorme successo di ESP8266, nel 2016 Espressif ha voluto superarsi con il modello ESP32 (Figura 2.14a). Sulla scia del suo predecessore, la scheda ESP32 è un sistema a basso costo con Wi-Fi e dual-mode Bluetooth integrati e molte interfacce di comunicazione.

Caratteristiche dell'ESP32.

- CPU: Tensilica Xtensa Dual-Core a 32 bit LX6 microprocessore, che opera a 160 o 240 MHz e che esegue fino a 600 DMIPS.
- Memoria: 520 KB SRAM.
- Connettività wireless: Wi-Fi: 802.11 b/g/n/e/i.
- Bluetooth: v4.2 BR/EDR e BLE.
- Interfacce periferiche: SAR-ADC a 12 bit fino a 18 canali.

- 2× DAC a 8 bit.
- 10× sensori di tocco.
- Sensore di temperatura.
- 4× SPI.
- 2× I²S.
- 2× I²C.
- 3× UART.
- Host SD/SDIO/MMC.
- Slave (SDIO/SPI).
- Interfaccia MAC Ethernet con supporto dedicato DMA e IEEE 1588.
- CAN bus 2.0.
- IR (TX/RX).
- Motore PWM.
- LED PWM fino a 16 canali.
- Sensore effetto Hall.
- Preamplificatore analogico a basso consumo energetico.
- Sicurezza: protezione standard IEEE 802.11 tutte supportate, tra cui WFA, WPA/WPA2 e WAPI.
- Avvio sicuro.
- Crittografia Flash.
- Accelerazione hardware crittografica: AES, SHA-2, RSA, crittografia ellittica della curva (ECC), generatore di numeri casuali (RNG).

Maggiori informazioni si possono trovare nel sito del produttore ufficiale, `https://espressif.com`, e nel sito della community dedicato alla scheda: `http://esp32.net`.

Come per il modello ESP8266, anche la scheda ESP32 viene ormai ospitata in molti prodotti di terze parti, come parte integrante per un

sistema Wi-Fi e Bluetooth oppure come scheda di sviluppo per applicazioni IoT più evolute.

Ecco alcuni esempi fra i più interessanti (in ordine alfabetico):

- Fluo Technology Fluo WiFi (Figura 2.14b);
- Protocentral KALAM32-DEV ESP32 (Figura 2.14c);
- NodeMCU-32S;
- Seeed Studio ESP-32S;
- SparkFun ESP32 Thing.

La scheda ESP32 è programmabile con Arduino IDE, Lua e altri ambienti di sviluppo.

Figura 2.14 La scheda ESP32 versione NoeMCU (a). Fluo WiFi (b). KALAM32-DEV ESP32 (c).

Intel

Dal 2013 il colosso mondiale Intel si è dedicato alla produzione di varie schede dedicate all'IoT e, grazie alla partnership di Arduino, è riuscita a fornire anche il supporto per alcuni prodotti davvero molto interessanti.

Le principali schede di Intel dedicate al mondo IoT sono:

- Galileo Gen 2;
- Edison;
- Curie;
- Joule;
- MinnowBoard Turbot.

Intel Galileo Gen 2

La scheda Intel Galileo Gen 2 (di seconda generazione) fornisce una scheda a singolo controller, basata sul Quark SoC X1000, un processore Intel Pentium a 32 bit (Figura 2.15).

La pagina ufficiale della scheda Galileo è: https://software.intel.com/en-us/iot/hardware/galileo.

La scheda Galileo Gen 2 offre un'offerta completa per un'ampia gamma di applicazioni. Essendo "Arduino-Certified", la scheda è stata progettata per essere compatibile sia dal punto di vista hardware che software con una vasta gamma di shield Arduino UNO R3. La scheda Gen 2 fornisce anche uno sviluppo più semplice e più economico rispetto ai processori Intel Atom e Intel Core.

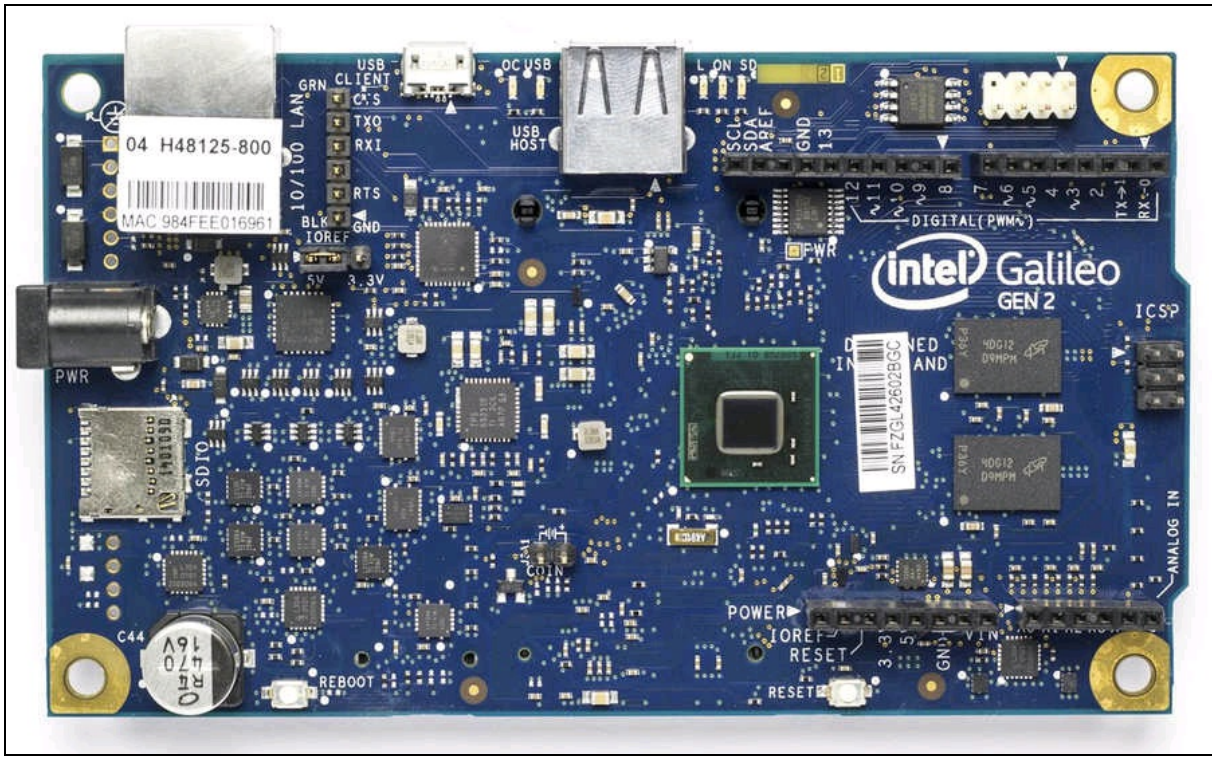


Figura 2.15 La scheda Galileo Gen 2.

Caratteristiche principali.

- Connettori USB.
- Presa RJ45 (Ethernet).
- Presa di alimentazione.
- Piedinatura compatibile con Arduino:
  - 20× I/O digitali;
  - 6× ingressi analogici;
  - 6× PWM con risoluzione a 12 bit;
  - 1× SPI;
  - 2× UART (1 condivisa con la UART della console);
  - 1× I^2C.
- ICSP a 6 poli.
- JTAG a 10 pin per il debug.
- RJ45 Ethernet, Power over Ethernet.

- Host USB 2.0 (tipo A standard).
- Client USB 2.0 (micro-USB Tipo B).
- 1× Mini slot PCI Express.
- Alimentazione con gamma aumentata (da 7 a 15 V).
- Supporta Power over Ethernet (richiede l'installazione del modulo PoE).
- Piedini per RTC.
- Tasti reset dello sketch e dello shield.
- Reset per ripristinare l'Intel Quark SoC X1000.

Di default è installata sul Galileo Gen 2 la distribuzione Linux Yocto 1.4 Poky. È possibile accedere a diverse funzioni Linux con la chiamata `system()`.

Per la programmazione della scheda Galileo si può usare anche l'IDE di Arduino.

Intel Edison

Dal punto di vista storico, Edison è il naturale successore di Galileo e la scheda Edison è il follow-up della precedente scheda, di cui ha in comune molte caratteristiche. Il pacchetto Edison viene consegnato in kit, composto dai seguenti elementi (Figura 2.16):

- Edison Compute Module;
- Arduino Expansion board;
- Kit di montaggio (viti e distanziali).

La novità sta nelle ridotte dimensioni del modulo Edison, che può essere pre-programmato sulla *Arduino Expansion board* e poi usato come dispositivo IoT autonomo.

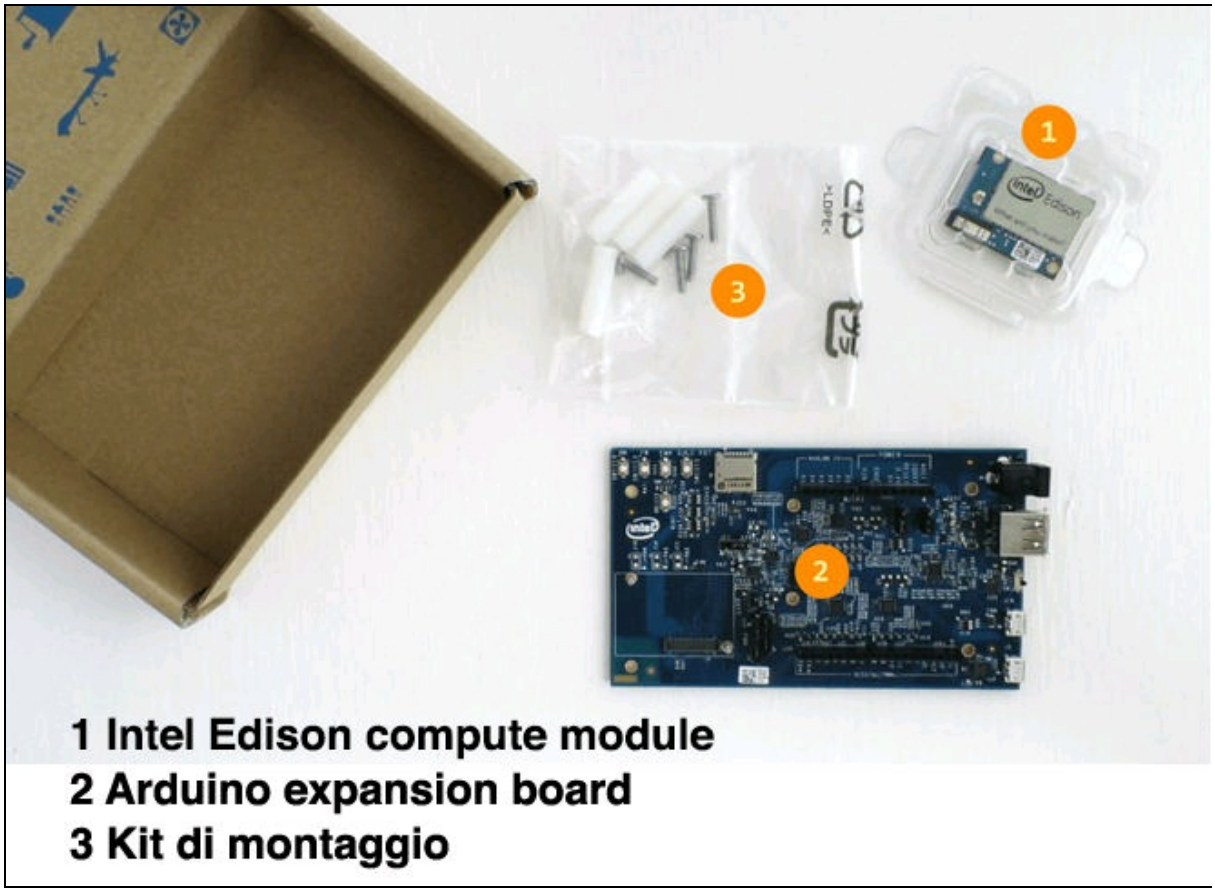


Figura 2.16 Il kit della scheda Intel Edison.

Caratteristiche principali.

- Supporto per Arduino sketch, Linux, Wi-Fi e Bluetooth.
- I/O della scheda: compatibili con Arduino UNO (tranne 4 PWM invece di 6 PWM).
- 20× pin digitali di ingresso/uscita, inclusi 4 pin come uscite PWM.
- 6× ingressi analogici.
- 1× UART (RX/TX).
- 1× I^2C.
- 1× ICSP (SPI).
- Connettore micro-USB del dispositivo.
- Micro-USB (collegato a UART).
- Connettore scheda SD.

- Presa di alimentazione (ingresso da 7 a 15 VDC).
- CPU Intel Atom 500 MHz a 32 bit.
- Microcontrollore Intel Quark a 100 MHz.
- RAM 1 GB LPDDR3 memoria POP.
- Memoria Flash 4 GB eMMC (v4.51 spec).
- WiFi Broadcom 802.11 a/b/g/n doppia banda (2,4 e 5 GHz).
- Antenna a bordo.
- Bluetooth 4.0.

Il sistema operativo della CPU è Yocto Linux v1.6. Gli ambienti di sviluppo sono i seguenti.

- Arduino IDE.
- Eclipse: C, C ++ e Python.
- Intel XDK: Node.JS e HTML5.
- MCU OS: RTOS.

La pagina ufficiale della scheda Edison è: https://software.intel.com/en-us/get-started-edison-windows.

Il modulo Edison è programmabile utilizzando Arduino IDE, C/C++, JavaScript, Node.js e Python.

Intel Curie

La risposta tecnologica della partnership con Arduino è il minuscolo modulo Intel Curie, le cui dimensioni sono illustrate nella Figura 2.17a, prodotto appositamente per essere alloggiato nella scheda Arduino 101 (Figura 2.17b).

Arduino 101 è una scheda pensata per l'educazione e l'apprendimento. Combina le prestazioni avanzate del modulo Intel Curie con la semplicità di Arduino, a livello di principiante. Infatti,

mantiene lo stesso fattore di forma di Arduino UNO, con l'aggiunta delle funzionalità Bluetooth LE e un accelerometro/giroscopio a sei assi.

Il modulo contiene due core, un Quark x86 e un core con architettura ARC a 32 bit, entrambi a 32 MHz. La toolchain Intel compila gli sketch Arduino in modo ottimale su entrambi i core per eseguire anche le applicazioni più impegnative.

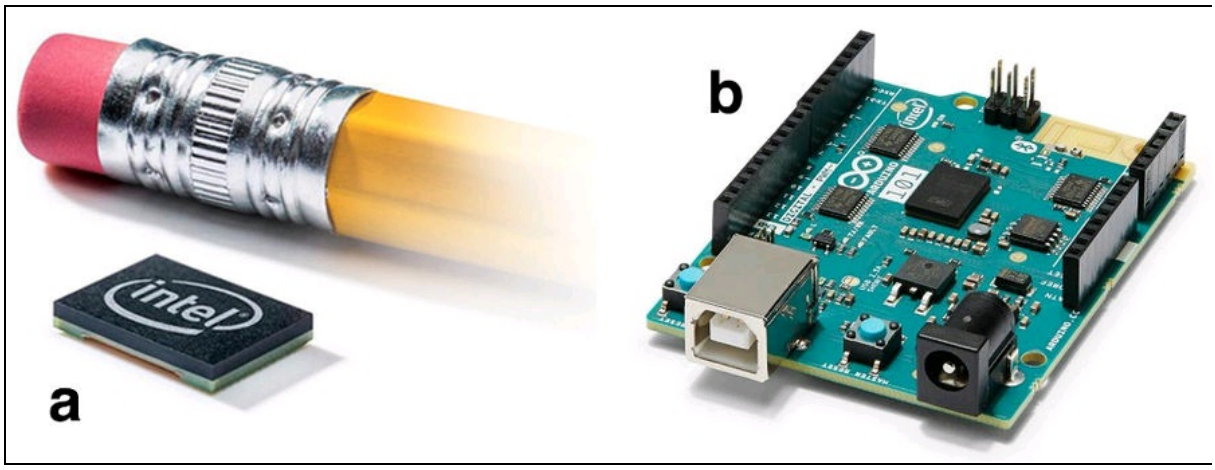


Figura 2.17 Il modulo Intel Curie (a). La scheda Arduino 101 (b).

Caratteristiche principali.

- Microcontroller: Intel Curie.
- Livello TTL: 3,3 V (I/O tolleranti 5 V).
- Tensione di ingresso (consigliata): 7-12 V.
- Pin I/O digitali: 14 (di cui 4 con PWM).
- Pins I/O digitali PWM: 4.
- Pin di ingresso analogici: 6.
- Corrente continua per pin I/O: 20 mA.
- Memoria Flash: 196 kB.
- SRAM: 24 kB.
- Velocità di clock: 32 MHz.
- LED_BUILTIN: 13.
- Bluetooth LE.

- Accelerometro/giroscopio a sei assi.

La pagina di riferimento per Arduino 101 è disponibile nel sito ufficiale Arduino: https://www.arduino.cc/en/Main/ArduinoBoard101.

L'ambiente di programmazione è l'IDE di Arduino.

Intel Joule

La scheda più recente di Intel si chiama Joule, rimanendo in tema con i nomi dei prodotti dedicati ai personaggi più famosi della fisica.

La piattaforma Intel Joule è un sistema su modulo (SoM) ed è disponibile in più configurazioni che condividono lo stesso ingombro e lo stesso collegamento al connettore dell'interfaccia (Figura 2.18). Ciò consente una progettazione semplificata del prodotto, fornendo più livelli di potenza di calcolo, prestazioni grafiche e opzioni di memoria in un'unica piattaforma.

Esistono due configurazioni del modulo, descritti nella seguente tabella.

Modulo	Clock CPU	Clock grafico	Memoria e stoccaggio
Intel Joule 550×	1,5 GHz	300 GHz	3 GB RAM 8 GB eMMC Flash
Intel Joule 570×	1,7 GHZ Turbo Boost fino 2,4 GHz	450 MHz Base 650 MHz Turbo	4 GB RAM 16 GB eMMC Flash

Il modulo 550× eccelle per applicazioni IoT e, in generale, per attività di raccolta di dati, processi di controllo o l'esecuzione di un'interfaccia utente grafica.

Il modulo 570× fornisce più potenza e più memoria per l'archiviazione di dati in un ambiente desktop Linux, la transcodifica video 4k e il supporto per le tecnologie Intel RealSense.

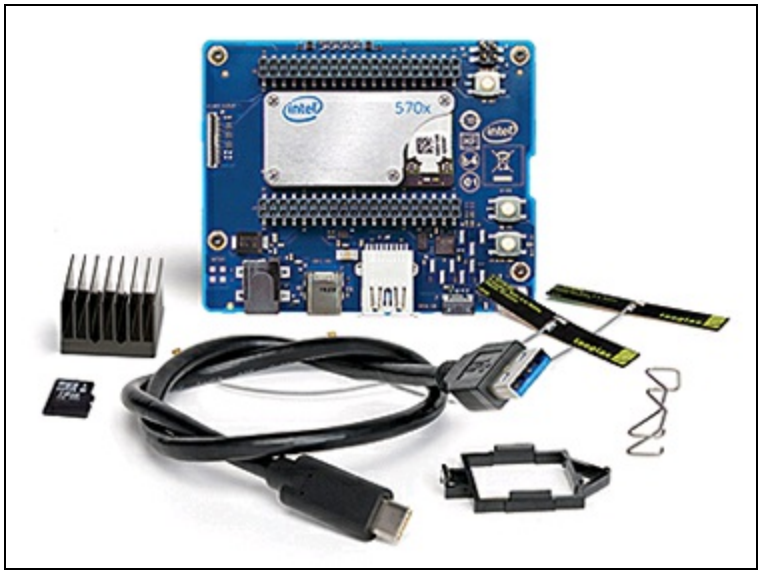


Figura 2.18 La piattaforma Intel Joule.

Caratteristiche principali.

- I^2C: sono disponibili cinque porte dal modulo. Alcune sono su pin multiuso e l'utilizzo effettivo può variare tra piattaforme e configurazioni OS.
- I^2S: viene fornita una porta dual channel per un'interfaccia audio (codec). La piattaforma di sviluppo supporta l'audio solo tramite USB o uscita HDMI.
- GPIO: 8 linee sono dotate di immagine BIOS predefinita, altre interfacce possono essere riconfigurate come GPIO fino a 48 totali.
- Sono disponibili 3 porte UART: sulla scheda di espansione, la UART2 viene indirizzata all'interfaccia di debug seriale.
- È disponibile un'interfaccia seriale SDIO.
- PWM: sono disponibili quattro linee PWM dedicate.
- Wi-Fi 802.11a/b/n con modalità multipla e peer to peer.
- Firewall integrato abilitato tramite libreria IPTABLE in BSP.
- Modalità Soft Access Point e servizi Client Scan.
- Modalità volo attivata selezionando tutti i servizi radio.
- Supporto Streaming e Miracast, quando configurati.

- Server e client WPS e protezione WPA2.
- Il sistema operativo di riferimento per IoT incorpora il layer di servizio BlueZ per la funzionalità Bluetooth.
- Scansione e associazione dei dispositivi fino a 15 modalità Bluetooth.
- Modalità volo attivata selezionando tutti i servizi radio.
- BLE console seriale per il debug remoto.
- Crittografia hardware per standard WPA2 e AES-CCMP.

La pagina di riferimento per Intel Joule è https://software.intel.com/en-us/intel-joule-getting-started. La piattaforma di sviluppo di Intel Joule è disponibile nel portale http://www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html.

Particle

Nata nel 2012 come Spark, l’azienda americana ha cambiato nome dopo un paio d’anni, preferendone uno meno “scintillante”, ma più facilmente identificabile.

Particle è oggi una realtà molto affermata nel settore IoT ed è arrivata a offrire una serie di prodotti che vanno dalle ottime schede hardware Photon ed Electron, alle soluzioni Internet come Internet Button e Asset Tracker, fino a Raspberry Pi + Particle Cloud.

Parlando di hardware, vediamo in breve i prodotti di punta di Particle:

- Photon;
- Electron.

Photon

Basata su architettura WICED di Cypress, la serie Photon Series combina un potente microcontrollore STM32 ARM Cortex M3 e un chip Wi-Fi Cypress.

Sono disponibili tre formati di Photon, per soddisfare le diverse esigenze personali o aziendali. Il modulo Photon su scheda breakout è perfetto per progetti e prototipi, mentre i moduli P0 e P1 sono pronti per la produzione orientata al mercato.

Caratteristiche principali.

- Wi-Fi: chip Cypress BCM43362.
- Single band 2.4 GHz 802.11b/g/n.
- Velocità di trasmissione dati fino a 65 Mbit/s.
- Alimentazione a basso consumo, modalità standby e stop.
- Modalità di protezione WiFi: aperta, WEP, WAPI, WPA e WPA2-PSK.

- Impostazione Soft AP.
- STM32F205 120 MHz ARM Cortex M3.
- 1 MB Flash.
- 128 KB di RAM.
- 18 GPIO per segnali misti e periferiche avanzate.

La pagina di riferimento di Photon è https://www.particle.io/products/hardware/photon-wifi-dev-kit. La programmazione di Photon si basa su Web IDE.

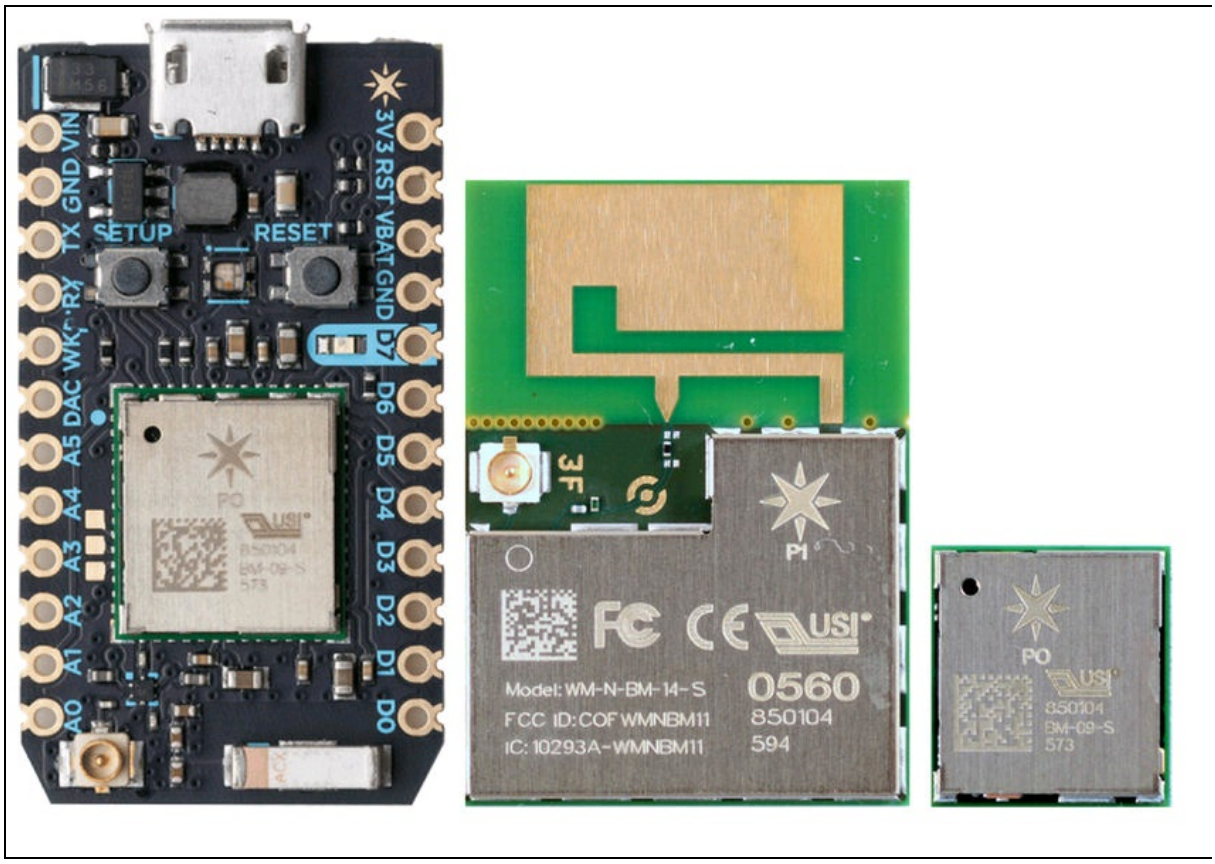


Figura 2.19 Il modulo Particle Photon in tre formati.

Electron

Il modulo Electron rende facile collegare il prodotto a una rete cellulare. È certificato FCC/CE/IC/PTCRB e quindi conforme con

qualsiasi norma per la rete cellulare. La serie Electron dispone di scheda SIM, permettendo la connettività alla rete globale e ai servizi cloud. Completamente integrato e attivato, il modulo Electron è pre-certificato e costruito in modo efficiente per ridurre i costi della trasmissione dati.

Caratteristiche principali.

- U-Blox SARA U260/U270 (3G) o G350 (2G).
- SIM Telefonica globale.
- Piano di dati Particle IoT: i piani variano in base all'uso e alla localizzazione.
- Microcontroller STM32F205 ARM Cortex M3.
- 1 MB Flash.
- 128 K RAM.
- 30 GPIO di ingresso misto con periferiche avanzate.
- 36 piedini totali: 28 GPIO (D0-D13, A0-A13), più TX/RX, 2 GND, VIN, VBAT, WKP, 3 V3, RST.
- Circuito di gestione dell'alimentazione incluso a bordo.
- FCC, IC e CE pre-certificati.

La pagina di riferimento di Electron è https://www.particle.io/products/hardware/electron-cellular-dev-kit. Per la programmazione di Electron è disponibile un toolkit completo di applicazioni Cloud.

Particle Cloud è una suite di strumenti che consente di gestire tutte le SIM da un unico luogo, configurando facilmente le applicazioni 2G e 3G sul cloud.

Si può scrivere e gestire il firmware con un insieme di diverse opzioni dell'ambiente di sviluppo.

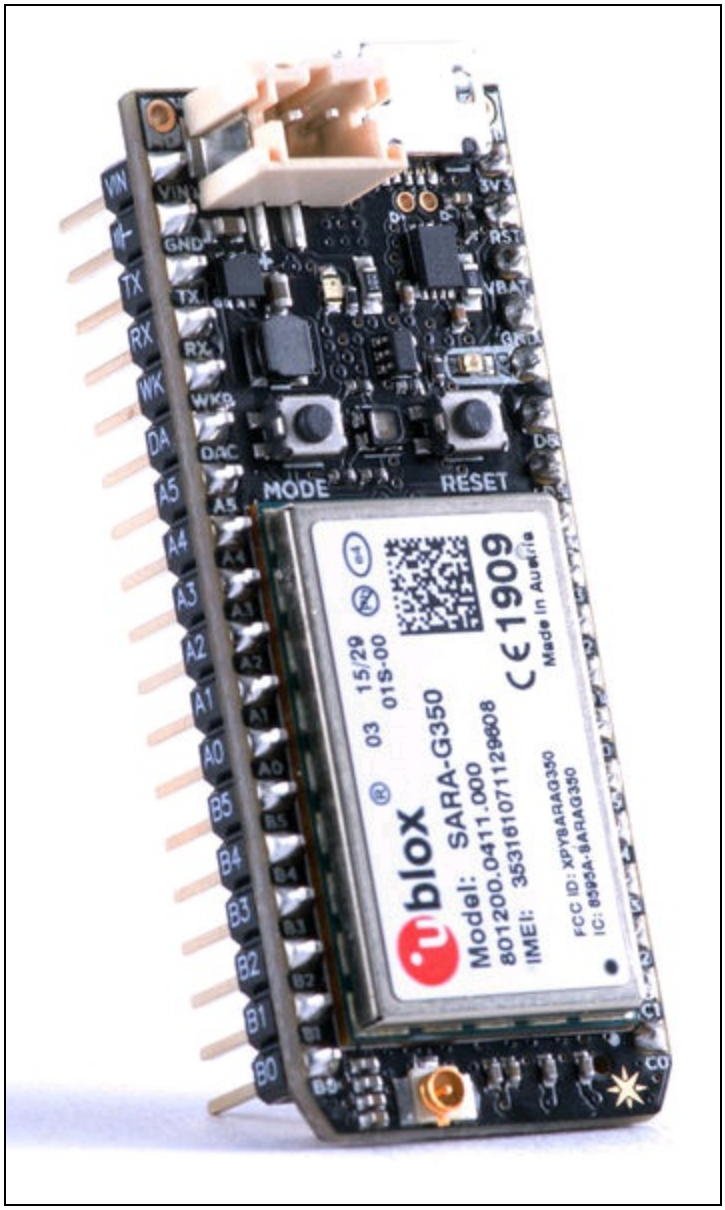


Figura 2.20 Il modulo Particle Electron.

Altre schede

Come già detto, sarebbe impossibile esaurire in queste un argomento così vasto. Diamo pertanto una veloce occhiata ad alcuni marchi altrettanto importanti di quelli trattati in queste pagine (in ordine alfabetico, non di importanza):

- Cypress;
- Samsung;
- Siemens;
- STMicroelectronics;
- Texas Instruments.

Cypress

L'azienda americana Cypress si è distinta negli ultimi tempi per alcuni prodotti dedicati al mondo IoT. Una soluzione fra tutte è il kit CY8CKIT-042-BLE-A Bluetooth Low Energy 4.2 Compliant Pioneer Kit (Figura 2.21).

Il kit consente di valutare e sviluppare soluzioni conformi a Bluetooth 4.2 utilizzando i dispositivi PSoC 4 BLE e PRoC BLE.

Pagina di riferimento: http://www.cypress.com/ble. La scheda si programma tramite PSoC Creator IDE, scaricabile dalla pagina di riferimento.

Figura 2.21 CY8CKIT-042-BLE-A Bluetooth Low Energy 4.2 Compliant Pioneer Kit.

Samsung

Dal 2016 anche Samsung ha esordito nel settore IoT con una linea di prodotti che ha battezzato come Artik.

Artik comprende una serie di moduli hardware, un cloud e un ecosistema per partner industriali.

I moduli hardware sono suddivisi in quattro famiglie: Artik 0, Artik 5, Artik 7 e Artik 10.

- *ARTIK 020* è un modulo Bluetooth destinato ad applicazioni Bluetooth Low Energy (BLE) in cui affidabilità RF, consumo ridotto e sviluppo di applicazioni industriali sono requisiti fondamentali.
- *ARTIK 030* è un modulo integrato pre-certificato per soluzioni di rete wireless che utilizza protocolli ZigBee o Thread. Combina un

SoC wireless multiprotocollo con un design RF e stack software wireless in un ambiente di sviluppo industriale.

- *ARTIK 520* offre dispositivi con una grande potenza di elaborazione e capacità di archiviazione per le sue dimensioni. Le opzioni wireless includono Wi-Fi, Bluetooth 4.1, Bluetooth Smart, ZigBee e Thread.
- *ARTIK 530* prende la potenza di ARTIK 520 e aggiunge energia riducendo i costi. Dispone di un processore Quad core per l'elaborazione dati locale e di un motore multimediale per l'elaborazione audio e video. I moduli supportano anche la maggior parte degli standard wireless come 802.11, Bluetooth, ZigBee e Thread.
- *ARTIK 710* offre 8 core per servire efficacemente come un gateway per un grande edificio o una fabbrica e gestire anche analisi locali per migliorare la latenza e la capacità di risposta. Fornisce inoltre soluzioni di connettività multiple IoT e dispone di un processore multimediale di fascia alta per gestire l'elaborazione video e audio.
- *ARTIK 1020* è ideale per applicazioni con elevati requisiti di calcolo locale, come il controllo robotizzato basato su modelli, realtà virtuale e l'elaborazione delle immagini. Ha DRAM e memoria flash integrata, interfacce per fotocamera e display, oltre a un completo complemento set di I/O digitali e analogici.

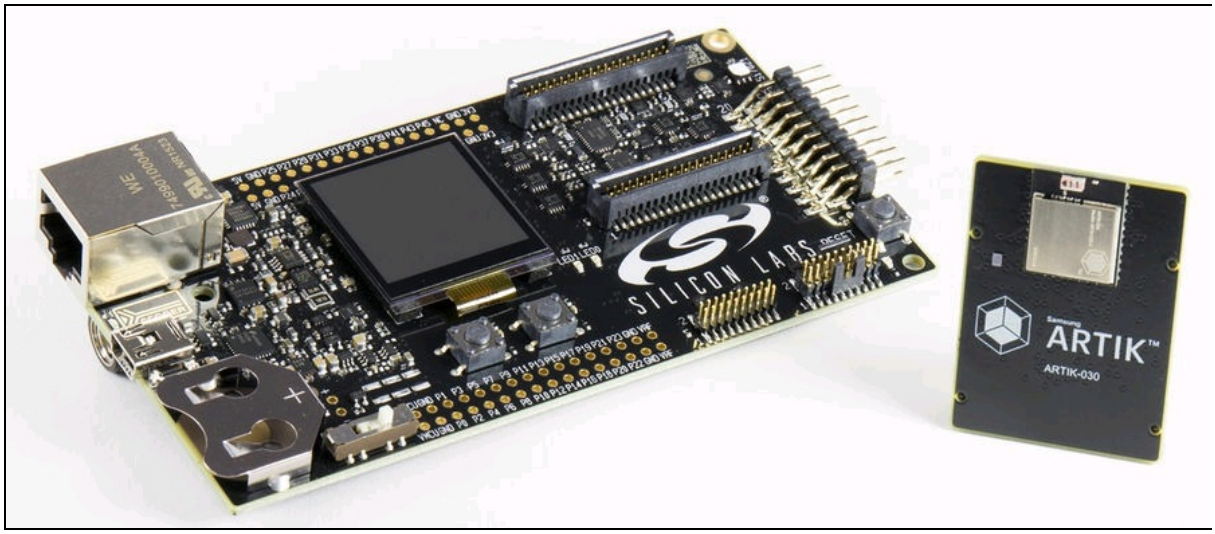


Figura 2.22 Il modulo Samsung Artik 030 e la scheda di sviluppo.

Pagina di riferimento: https://www.artik.io. Le schede delle famiglie Artik 5, 7 e 10 possono essere programmate tramite Eclipse e Arduino IDE.

Siemens

Non ha certo bisogno di presentazioni questo colosso dell'industria. Grazie alla partnership con Arduino e RS Components nel 2016, la conglomerata tedesca ha sfoderato un prodotto entry-level e a basso costo per studenti e maker che vogliono cimentarsi nella programmazione di oggetti IoT certificati per l'industria.

Il prodotto di cui vogliamo parlare si chiama SIMATIC IOT2020 (Figura 2.23). Presentata al mercato mondiale da Massimo Banzi (Arduino) in persona, questa scheda combina in un confezione di sapore industriale tutte le qualità di un prodotto di elevate prestazioni con la facilità di programmazione e di utilizzo di una scheda Arduino UNO.

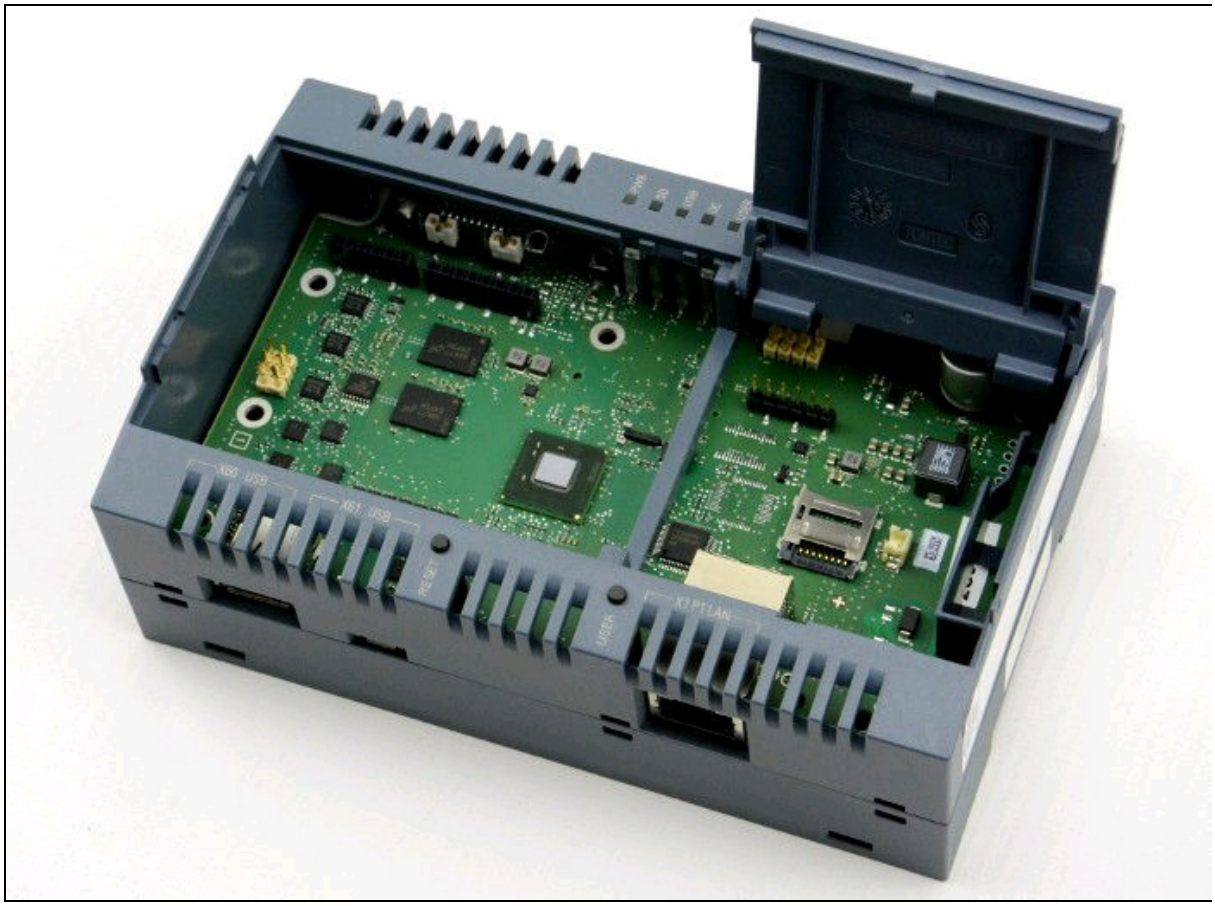


Figura 2.23 La scheda Siemens SIMATIC IOT2020.

Caratteristiche principali.

- Processore: Intel Quark ×1000.
- Memoria di massa su microSD fino a 32 GB.
- Memoria RAM/Flash/SRAM: 512 MB/8 MB /256 kB.
- Interfacce: interfacce seriali USB controller + USB dispositivo.
- Interfacce di programmazione: FTDI (console di sistema).
- TTL-232R-3 V3 (accessibile internamente).
- Interfacce Ethernet 10/100 Ethernet RJ45.
- Interfacce grafiche su slot di espansione.
- Espansione Arduino/Pinout.
- Espansioni mPCIe.
- Compatibile con Arduino UNO R3.

- Sistema operativo su SD: Linux Yocto V2.1 (Krogoth).

Pagina di riferimento:

https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/HomeSCE/SupportoDidattico/Par

La scheda SIMATIC IOT2020 è programmabile tramite l'IDE di Arduino, Eclipse (C/C++), Node-Red.

STMicroelectronics

STMicroelectronics (o come viene comunemente chiamata, ST) è un'azienda franco-italiana con sede a Ginevra. Forse non tutti sanno che l'azienda inizialmente fu fondata da Roberto Olivetti, figlio del più famoso Adriano Olivetti.

Oggi ST è un colosso mondiale dell'elettronica e si distingue soprattutto per soluzioni embedded destinate perlopiù a terze parti.

Da qualche anno è divenuta famosa anche nel mondo dei maker grazie a prodotti della serie Nucleo. Delle decine di modelli disponibili, diamo un'occhiata al modello STM32 Nucleo-64 con processore STM32F030R8 (Figura 2.24).

Caratteristiche principali.

- Microcontrollore STM32 con package QFP64.
- Connettività compatibile con Arduino UNO R3.
- Estensione morpho ST per un accesso completo a tutti gli I/O STM32.
- Debugger/programmatore ST-LINK/V2-1 a bordo con connettore SWD.
- Alimentazione flessibile: USB o fonte esterna (3,3 V, 5 V, 7-12 V).
- Tre LED: USB (LD1), LED utente (LD2), LED di alimentazione (LD3).
- Due pulsanti: USER e RESET.

- Capacità di re-enumerazione USB. Tre diverse interfacce supportate su USB: porta COM virtuale, memoria di massa, porta di debug.

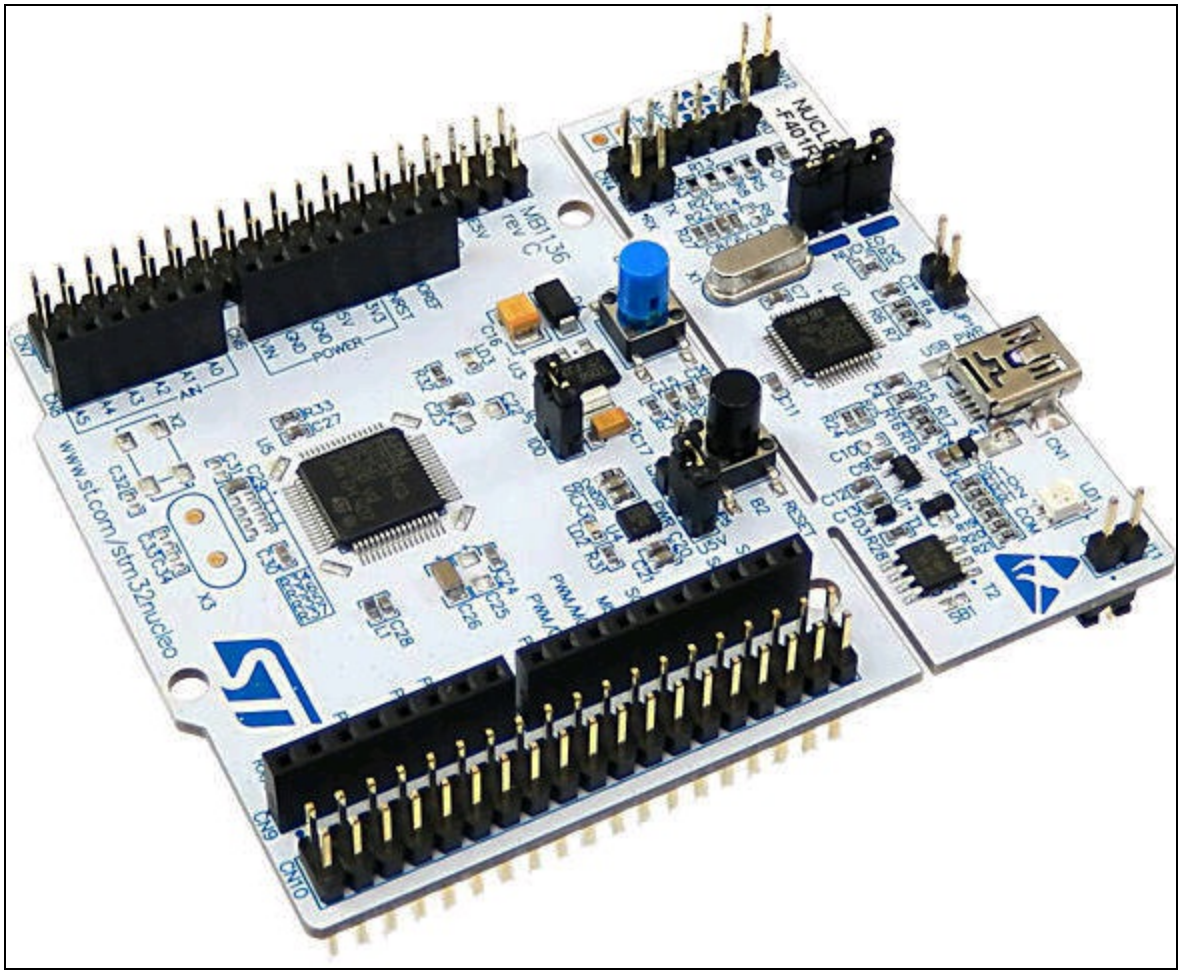


Figura 2.24 La scheda STM32 Nucleo-64 con processore STM32F030R8 (NUCLEO-F030R8).

Pagina di riferimento:
http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo/nucleo-f030r8.html.

La scheda supporta una vasta gamma di ambienti di sviluppo integrato (IDE), tra cui IAR, ARM Keil, IDE su GCC e Arduino (vedi http://www.stm32duino.com) e ARM mbed (vedi http://mbed.org).

Texas Instruments

Texas Instruments, meglio conosciuta come TI, è un’azienda americana nata negli anni Trenta e divenuta famosa per la produzione e la vendita di componenti elettronici e di chip e, in seguito, per la produzione di calcolatrici e di computer.

TI, preceduta da Intel e da Samsung, risulta essere il terzo produttore mondiale di dispositivi elettronici sparsi in tutto il mondo.

Da qualche anno TI si è distinta per alcuni prodotti destinati al settore maker e IoT, creando un marchio apposito e un sito dedicato. Parliamo delle schede BeagleBone e BeagleBoard. I modelli più recenti sono:

- BeagleBone Black Wireless;
- BeagleBone Blue;
- BeagleBone Green Wireless (by Seeedstudio).

Caratteristiche principali di BeagleBone Black Wireless (Figura 2.25a).

- Processore: Octavo Systems OSD3358 1GHz ARM® Cortex-A8.
- RAM: 512 MB DDR3.
- 4GB di memoria flash eMMC a bordo.
- Acceleratore grafico 3D.
- Acceleratore NEON in virgola mobile.
- 2× microcontrollori PRU a 32 bit.
- Connettività: USB per alimentazione e comunicazione.
- Host USB.
- Wi-Fi 802.11 b/g/n e Bluetooth 4.1 e BLE.
- HDMI.
- 2× header a 46 pin.
- Debian con Cloud9 IDE su libreria Node.js con BoneScript.
- Supporto di terze parti per Android e Ubuntu e altro.

Caratteristiche principali di BeagleBone Blue (Figura 2.25b).

- Processore: Octavo Systems OSD3358 1GHz ARM® Cortex-A8.
- RAM: 512 MB DDR3.
- Gestione integrata dell'alimentazione.
- 2× microcontrollori PRU a 32 bit.
- ARM Cortex-M3.
- Memoria a 4GB di memoria eMMC a bordo.
- Batteria: supporto LiPo a 2 celle con bilanciamento, ingresso 9-18 V.
- Wireless: 802.11b/g/n.
- Bluetooth 4.1 e BLE.
- Comando motore: 8 servo out 6 V, 4 motori DC fuori, 4 encoder.
- Sensori: 9 assi IMU, barometro.
- Connettività: client USB HighSpeed e Host.
- Interfaccia utente: 11 LED programmabili dall'utente, 2 pulsanti programmabili dall'utente.
- Facili interfacce di collegamento per l'aggiunta di sensori aggiuntivi quali: GPS, radio DSM2, UART, SPI, I^2C, 1,8 V analog, GPIO 3,3 V.
- Debian, ROS, ArduPilot e altri.
- Programmazione grafica con LabVIEW, Cloud9 IDE su Node.js.

BeagleBone Green Wireless (Figura 2.25c). Questo modello è uno sforzo congiunto di BeagleBoard.org e Seeed Studio. È basato sul disegno hardware open source di BeagleBone Black e sviluppato in una versione differenziata.

SeeedStudio BeagleBone Green Wireless ha incluso un'interfaccia WiFi/Bluetooth flessibile ad alte prestazioni e due connettori Grove, rendendo più semplice la connessione alla vasta famiglia di sensori Grove.

I connettori HDMI e Ethernet sono stati rimossi per fare spazio a queste funzionalità wireless e al connettore Grove. Per il resto, le caratteristiche principali sono identiche ai due modelli precedenti.

Sito di riferimento: http://beagleboard.org. Le schede BeagleBone sono programmabili in ambiente Linux o Windows tramite Eclipse e Linaro-gcc toolchain.

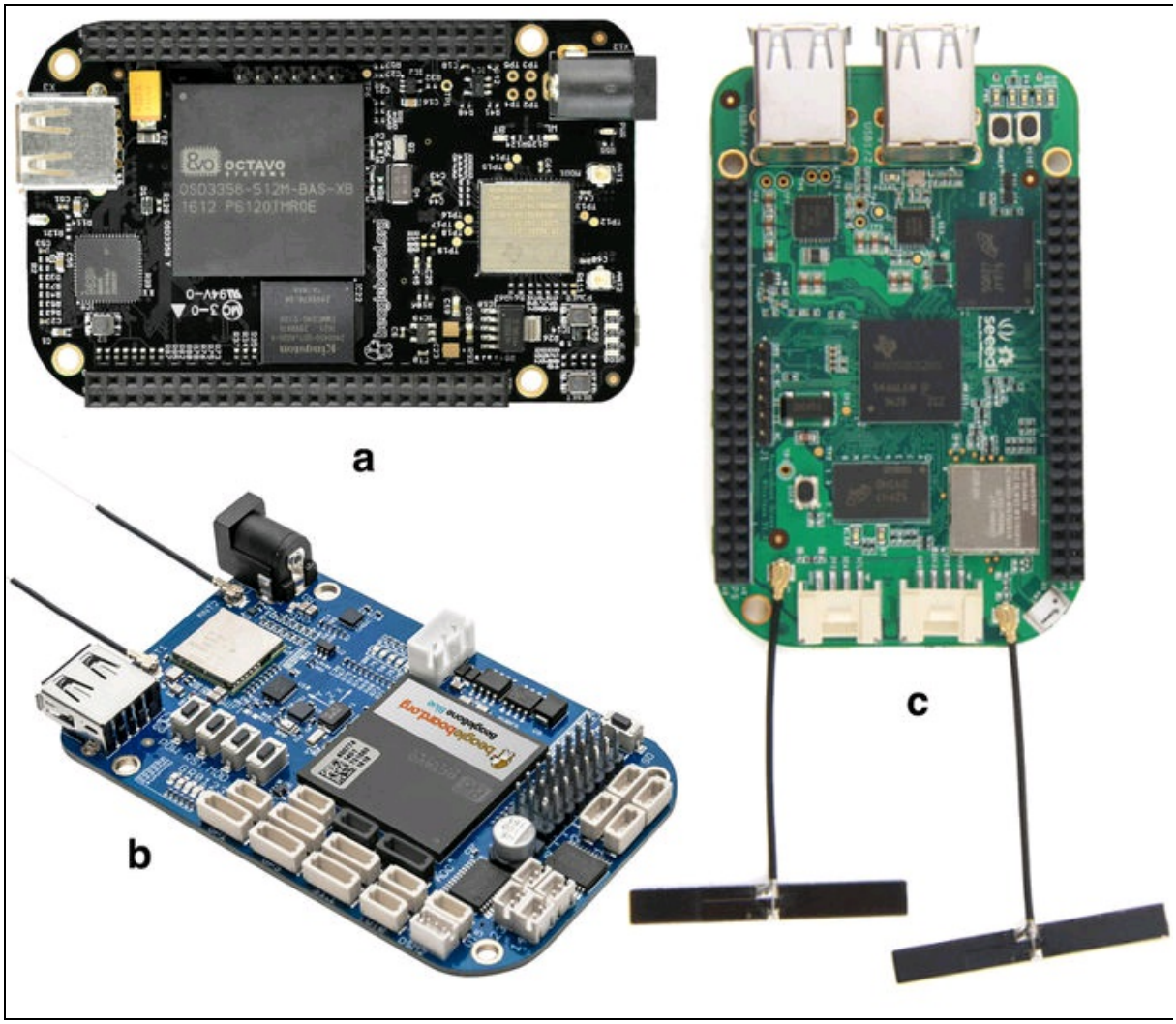


Figura 2.25 Le schede BeagleBone Black Wireless (a), Blue (b) e Green Wireless (c).

Microchip Atmel

Anche se questo paragrafo non parla di una scheda specifica, vogliamo ugualmente sottolineare l'importanza di questi due giganti dell'elettronica.

Nel 2016 il colosso dell'elettronica Microchip Technology ha acquisito l'altro colosso Atmel per 3,6 miliardi di dollari. Entrambe le aziende sono americane e sono state fondate negli anni Ottanta.

Atmel ha il merito di aver inventato chip importanti come l'ATmega328, che ha dato origine al primo Arduino. Microchip ha il merito di aver inventato il PIC, il predecessore di Arduino. Microchip ha acquisito tutto il catalogo di Atmel e oggi possiamo ancora usufruire dei chip denominati con i nomi di Atmel. Sito ufficiale:

http://www.microchip.com.

Oltre al già citato ATmega328 (montato sugli attuali Arduino UNO), quello che ci piace ricordare e porre all'attenzione dei maker è la famiglia di chip ATtiny, in particolare i tre piccoli ma potenti *ATtiny25*, *ATtiny45* e *ATtiny85* (Figura 2.26).

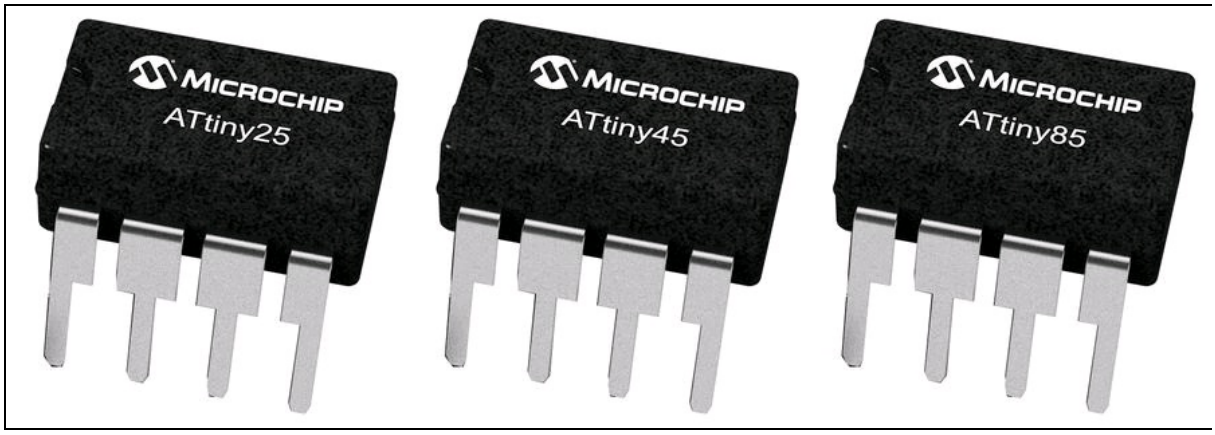


Figura 2.26 ATtiny25, ATtiny45 e ATtiny85.

ATtiny25, 45 e 85

Per la serie “piccolo è bello”, questi tre chip sono microprocessori a 8 piedini DIP (*Dual In-line Package*).

Nel loro nome c’è tutto: tiny (si pronuncia “taini”) in inglese significa minuscolo e i processori di tutta la famiglia ATtiny lo sono per davvero. Piccoli fuori ma grandi al loro interno, possono ospitare interi sketch fatti con l’IDE di Arduino ed essere collegati con il mondo esterno con un’infinità di componenti attivi e passivi. Si può davvero pensare ad applicazioni hardware miniaturizzate sfruttando un processore di pochi millimetri e senza impiegare ingombranti schede.

Dal sito Microchip ecco le caratteristiche dei tre chip che, a parte la memoria Flash e la memoria EEPROM, sono identiche.

Caratteristiche	ATtiny25	ATtiny45	ATtiny85
Memoria Flash (Kbytes)	2 Kbyte	4 Kbyte	8 Kbyte
Numero di pin	8	8	8
Frequenza massima di funzionamento	20 MHz	20 MHz	20 MHz
EEPROM (byte)	128	256	512
Numero di canali Touch	3	3	3
Numero massimo di pin I/O	6	6	6
Interrupt esterni	6	6	6
Interfaccia SPI	Sì	Sì	Sì
Interfaccia I^2C	Sì	Sì	Sì
Intervallo di tensione operativa (volt)	1,8-5,5	1,8-5,5	1,8-5,5
Capture/Compare/PWM Peripherals	5PWM	5PWM	5PWM

Le specifiche tecniche comuni ai tre microprocessori sono:

- 6× ingressi/uscite (GPIO);
- 32× registri General Purpose;
- 1× 8 bit Timer/Counter;
- 1× 8 bit High Speed Timer/Counter;
- Universal Serial Interface;

- interrupt esterni e interni;
- ADC a 4 canali 10 bit;
- Watchdog Timer programmabile con oscillatore interno;
- tre modalità per risparmio energia;
- Analog Comparator;
- ADC Noise Reduction;
- On-chip ISP Flash;
- interfaccia seriale SPI;
- programmabile in C, Macro Assembler, Arduino IDE e altri kit di sviluppo.

Come si vede dalle specifiche, i tre microprocessori sono identici (eccetto la memoria Flash e EEPROM) anche nella piedinatura (Figura 2.27), per cui si possono progettare circuiti stampati o esperimenti su breadboard identiche. La possibilità di alimentare i chip anche con basse tensioni consente di costruire dispositivi portatili alimentati con una pila a bottone da 3 volt.

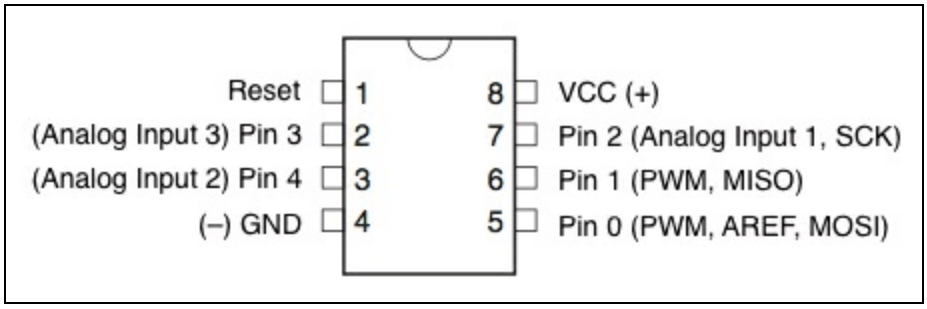


Figura 2.27 Piedinatura di ATtiny25/45/85.

Per la programmazione di un chip ATtiny tramite l’IDE di Arduino, si veda il Capitolo 3.

Ambienti di programmazione

La vita di un maker può essere allietata dalla conoscenza degli elementi di base della programmazione. Quello che fa di un maker qualcosa di simile a un hacker esperto è la capacità di trovare soluzioni ed escamotage in qualsiasi situazione.

È fuori di dubbio che online si trovi tanta “pappa pronta”, ma è altrettanto vero che bisogna conoscere gli strumenti per poter sfruttare tale “pappa” al fine di realizzare o elaborare i progetti proposti da altri.

In altre parole, è indispensabile che il maker sappia usare, almeno in parte, i principali software che sono efficaci a tale scopo e che servono ad accrescere l’esperienza.

ESEMPI DI CODICE

Tutti gli sketch di Arduino, gli script di Python, i progetti con Visual Studio, i layout di Fritzing usati in questo Capitolo sono disponibili nelle risorse del libro presso il sito dell’autore all’indirizzo `http://www.pierduino.com`.

Fritzing

Fritzing è un software di progettazione elettronica divenuto famoso per la sua facilità di utilizzo. La maggior parte degli schemi usati nel Web e tutti quelli in questo libro sono fatti con Fritzing, per cui sarà molto facile seguire il montaggio dei componenti sia dei progetti illustrati qui sia dei progetti che ognuno potrà inventare per proprio conto, una volta che avrà imparato a usare questo strumento insostituibile.

Fritzing è nato come software libero per aiutare gli sviluppatori dilettanti, ma anche i professionisti, in progetti orientati principalmente ad Arduino, ma con l'avvento di Raspberry Pi e altre nuove schede, la sua libreria si è arricchita a dismisura e viene aggiornata costantemente.

La caratteristica principale che lo distingue da altri software di progettazione elettronica è l'interfaccia innovativa che permette la prototipazione su *breadboard* (altrimenti detta basetta sperimentale), su schema elettrico e su PCB (*Printed Circuit Board*) ovvero su circuito stampato.

Il software è stato sviluppato e viene mantenuto da un gruppo di volontari dell'Interaction Design Lab di Potsdam (Germania), molto attivo e al quale va sicuramente tutto il nostro sostegno. Nella pagina del download gratuito si può provvedere anche a una donazione. Il sito ufficiale è `http://fritzing.org`.

Fritzing supporta le seguenti piattaforme:

- Windows a 32 e 64 bit;
- OS X (dalla versione 10.7 in poi);
- Linux (64 e 32 bit);
- codice sorgente per sviluppatori.

Una volta aperto il programma, si può impostare la lingua italiana dal menu *Edit* > *Preferences*, anche se non tutte le voci dei menu sono localizzate. All'avvio di Fritzing appare una pagina *Welcome* con news dal blog del sito, il menu dei file recenti, la finestra dei suggerimenti del giorno e il link per lo Shop Fab di Fritzing. Si può passare alla fase di progettazione facendo clic sulla scheda *Breadboard*. La vista Breadboard appare simile a quella illustrata nella Figura 3.1, in cui è visibile una breadboard vuota.

NOTA

Fritzing non è un emulatore di circuiti, ovvero non simula il funzionamento del circuito elettronico costruito su breadboard o su schema elettrico. Per testare i progetti, è necessario prima montare i componenti su breadboard ed effettuare i collegamenti alle varie schede. Per un emulatore su breadboard (a pagamento) consultare il sito commerciale Virtual breadboard (http://www.virtualbreadboard.com).

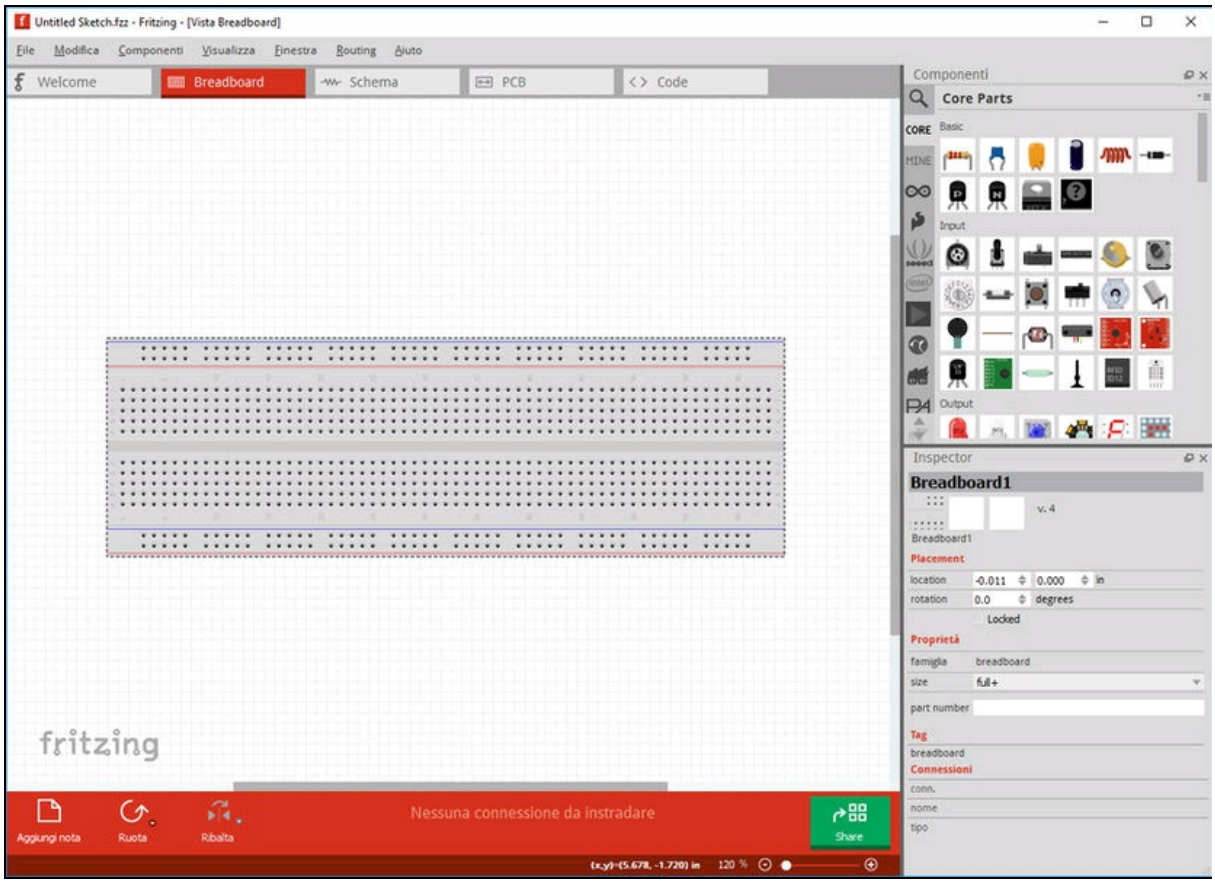


Figura 3.1 Finestra principale di Fritzing (versione 0.9.3) con la vista Breadboard.

Interfaccia

Premesso che il software è troppo esteso per una trattazione approfondita, daremo le informazioni necessarie alla realizzazione dei progetti del libro.

Prima di spiegare i controlli e i menu dell'interfaccia è utile imparare a muoversi senza fatica nella finestra principale.

- Lo zoom può essere effettuato comodamente tramite la rotella del mouse, oltre che con il controllo specifico in basso a destra nella finestra.
- Lo spostamento nella finestra può essere effettuato premendo la barra spaziatrice e trascinando il mouse (il cursore diventa la classica "manina"). La stessa operazione può venire effettuata con la rotella del mouse più il tasto Ctrl.

All'avvio, Fritzing si apre con la pagina *Welcome* (Ctrl+1), da cui è possibile aprire i file recenti, leggere le news dal blog del sito, leggere i suggerimenti del giorno o effettuare ordini nel Fritzing Fab. Facendo clic sulle schede della finestra principale si può accedere alle quattro sezioni principali:

- vista *Breadboard* (accessibile anche con Ctrl+2);
- vista *Schema* (accessibile anche con Ctrl+3);
- vista *PCB* (accessibile anche con Ctrl+4);
- vista *Code* (accessibile anche con Ctrl+5).

Vista Breadboard

È la vista ideale per coloro che vogliono sperimentare velocemente un prototipo su breadboard. La breadboard visualizzata ha un aspetto

realistico e riporta fedelmente la stessa disposizione standard dei collegamenti centrali e dell’alimentazione sui due lati.

Anche i componenti elettronici hanno un aspetto realistico e vanno “infilati” nei fori esattamente come si farebbe con una vera breadboard.

ATTENZIONE

È molto importante verificare che tutti i collegamenti sullo schermo siano fedelmente riportati nella breadboard reale. Accertarsi sempre che non ci siano corti circuiti o alimentazioni errate. Si possono provocare danni irreparabili alla scheda collegata. Si consiglia di perdere un po’ di tempo nel controllo piuttosto di rischiare di buttare via tutto.

Vista Schema

Per coloro che sono abituati a progettare su schema elettrico, la vista *Schema* permette di collegare i componenti attraverso i simboli elettrici (Figura 3.2). Per il neofita è un ambiente abbastanza complicato, per cui si consiglia la progettazione su breadboard, anche perché il circuito creato su schema elettrico non viene riportato nella breadboard, essendo la simbologia molto diversa. Si consiglia di partire a progettare dalla vista *Breadboard* e di adattare manualmente lo schema elettrico in un secondo tempo.

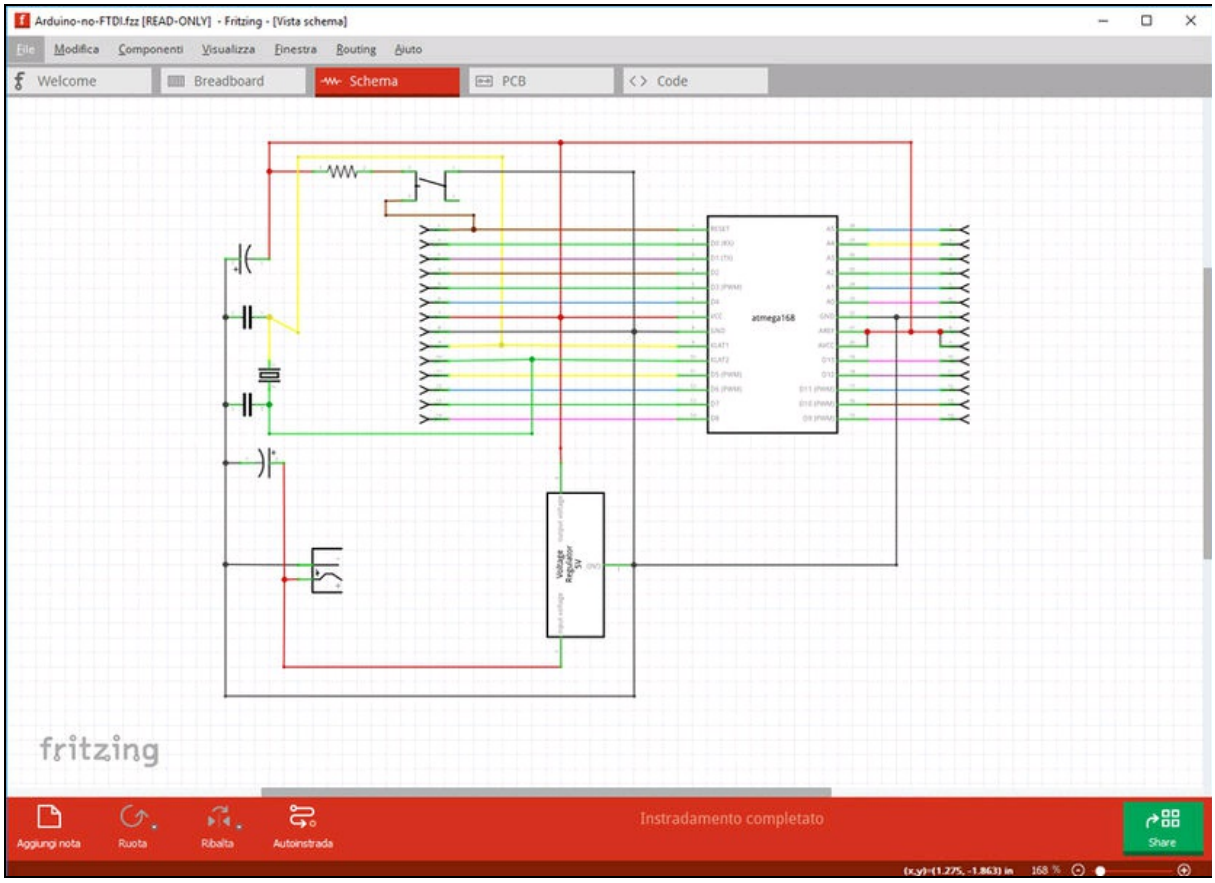


Figura 3.2 Vista Schema.

Vista PCB

È l'ambiente ideale per creare un circuito stampato perfetto (Figura 3.3). Dopo aver sperimentato con la breadboard è auspicabile dare al progetto un aspetto più professionale, e soprattutto più solido.

Dalla vista *PCB* si può creare il file di produzione per la stampa di una vera basetta da mandare al servizio di stampa online Fritzing Fab o a un altro stampatore locale. In alternativa, per chi ha tempo e voglia, è possibile stampare il file PCB su un foglio di acetato e creare la basetta in casa. Il risultato è sicuramente un po' meno professionale, ma più "cool".

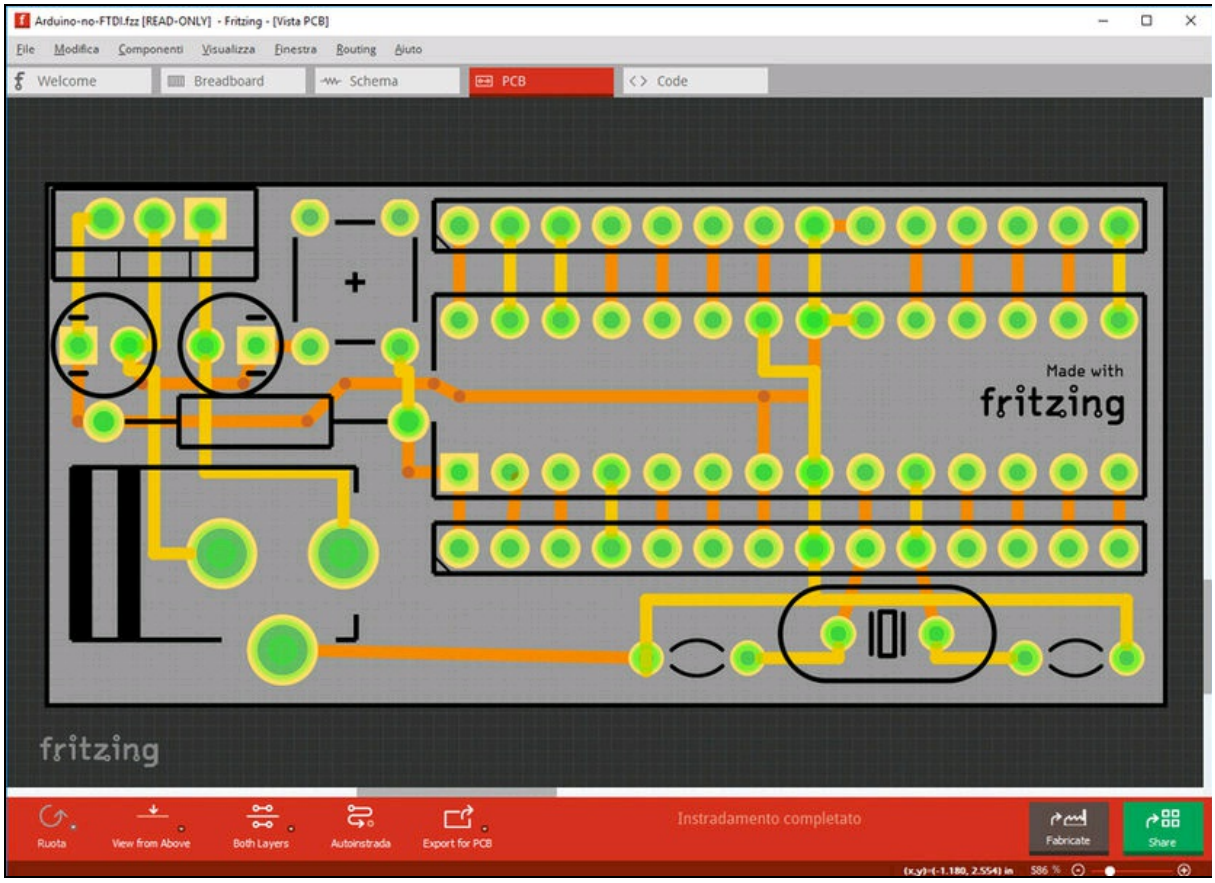


Figura 3.3 Vista PCB.

Vista Code

In pratica, è una specie di IDE per Arduino. Qui si possono importare gli sketch di Arduino, compilarli e caricarli nella scheda esattamente come fa l'IDE di Arduino. Serve solamente per avere sott'occhio il codice quando si progetta un circuito, ma non ha nessuna funzione di controllo della scheda o di simulazione. Probabilmente, si tratta di funzionalità che gli sviluppatori di Fritzing hanno in mente di implementare nelle versioni future.

Pannello Componenti

Nella parte destra della finestra principale è presente il pannello *Componenti* (Figura 3.4). Qui sono presenti diverse librerie di componenti elettronici di tutti i tipi. A ogni aggiornamento di Fritzing il pannello componenti si arricchisce sempre di nuove categorie. Se un determinato componente dovesse mancare, è comunque possibile modificarne uno o crearlo da zero attraverso l'editor interno, molto sofisticato. I componenti vanno posizionati nelle varie viste con il semplice trascinamento del mouse. A seconda della vista utilizzata, ogni componente ha un aspetto diverso e, ovviamente, non possono essere usati componenti incompatibili con quel tipo di vista.

Ogni libreria è suddivisa in categorie. Nella libreria *Core*, quella con i componenti più usati, sono disponibili le seguenti categorie.

- *Basic*: resistori, condensatori, induttori, diodi, transistor, FET e mystery part, un componente di cui non è specificato l'uso ma che si può modificare.
- *Input*: potenziometri/trimmer, compensatori, encoder, pulsanti, interruttori, fotocellule, sensori di vario tipo, accelerometri, antenne e altro ancora.
- *Output*: LED, display a 7 segmenti, schermi LCD, altoparlanti piezo e normali, microfoni, motori DC, motori servo, motori stepper, solenoidi, relè e altro ancora.
- *Textile:* componenti elettrici per l'uso dell'elettronica nel tessile.

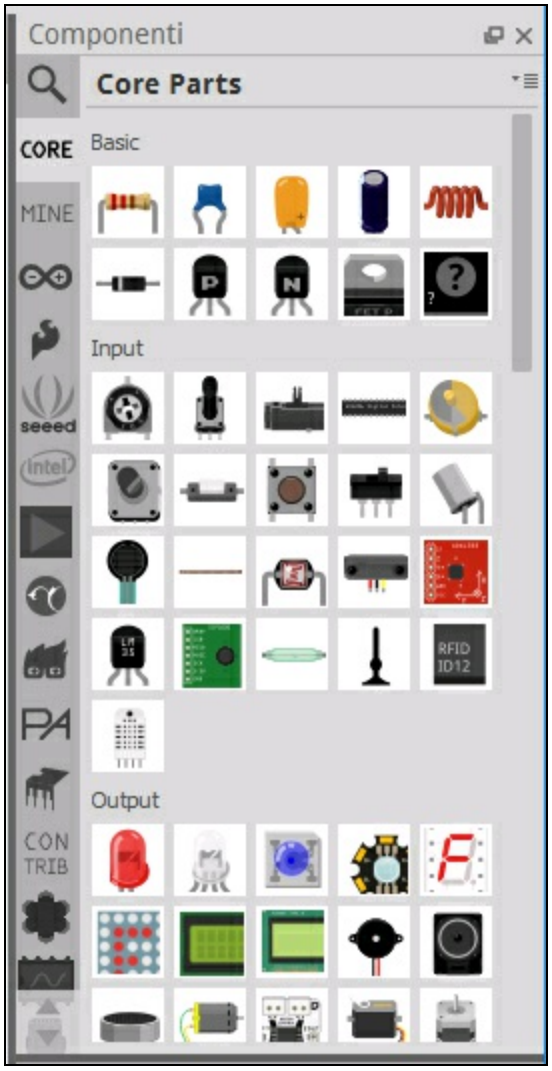


Figura 3.4 Pannello Componenti.

- *ICs*: circuiti integrati più utilizzati e quarzi.
- *Power*: batterie, regolatori di tensione e altro ancora.
- *Connection*: pin header di vario tipo, terminali, prese USB, shield per Arduino, Xbee, Motor Driver e altro ancora.
- *Microcontroller*: Arduino, Raspberry Pi, Netduino, Picaxe, Parallax e altro ancora.
- *Computer*: Raspberry Pi 2/3/Zero, Yun, Galileo, Edison, Linino, Udoo, 96 Boards.

- *Vista Breadboard*: elementi aggiuntivi per la breadboard (millefori, ponticelli).
- *Connection*: connettori di tutti i tipi.
- *Vista Schema*: elementi aggiuntivi per lo schema elettrico (simboli elettrici, note del progettista).
- *Vista PCB*: elementi aggiuntivi per il circuito stampato (jumper, vie, fori, piazzole ramate e altro).
- *Tools*: blocco note e righello con misure in centimetri e pollici.

Ogni versione aggiornata di Fritzing o il sistema di aggiornamento automatico delle parti e dei componenti aggiunge sempre nuovi componenti e farne un elenco completo è impossibile. Le librerie sono spesso aggiornate in tempo reale tramite la connessione Internet al sito. Fra le più interessanti, citiamo:

- Arduino;
- Raspberry Pi;
- Intel;
- Analog Devices;
- Seed Studio;
- Atlas Scientific;
- Parallax;
- Picaxe;
- Snootlab;
- Lilypad;
- Sparkfun (con una serie di altre librerie aggiuntive);
- Dagos;
- WeMos.

Inoltre, la sezione CONTRIB mette a disposizione i contributi degli utenti e viene sempre aggiornata. Chiunque voglia condividere il proprio

progetto fatto con Fritzing, lo può mettere a disposizione della community, registrandosi gratuitamente al sito `fritzing.org`.

Nel pannello *Componenti* si può creare anche una propria libreria personalizzata.

Infine, un comodo motore di ricerca consente di cercare i componenti per gestire meglio gli elementi visualizzati nel pannello.

Pannello Adafruit

Si consiglia di scaricare e di aggiungere a Fritzing il pannello dei componenti di Adafruit, scaricabile dal sito `https://github.com/adafruit/Fritzing-Library`. Una volta decompresso il file ZIP, è possibile installare il pannello aprendo il file `AdaFruit.fzbz` e, volendo, anche tutti gli altri. Si fa notare la presenza nella stessa cartella del file `CircuitPlayground.fzbz`, che installa alcuni componenti "sorridenti", adatti alla programmazione di circuiti didattici per bambini.

Inspector

Selezionando un elemento dal pannello *Componenti* e quando l'elemento è stato posizionato nella vista, è possibile visualizzarne e/o modificarne le proprietà. Per esempio, selezionando un resistore standard e posizionandolo nella breadboard se ne possono modificare le proprietà, come illustrato nella Figura 3.5.

- *Package*: più THT (terminali passanti) o SMD (*Surface Mount Device*).
- *Tolerance*: tolleranza da +/- 0.05% a 20%.
- *Bands*: 4 o 5 anelli.
- *Resistenza*: i valori standard più usati.
- *Power*: qualsiasi potenza.

- *Pin spacing*: spaziatura da 100 mil (in piedi) a 800 mil (non ha senso nella vista *Breadboard*, ma solo nella vista *PCB*).
- *Part number*: testo libero per inserire il numero della parte.

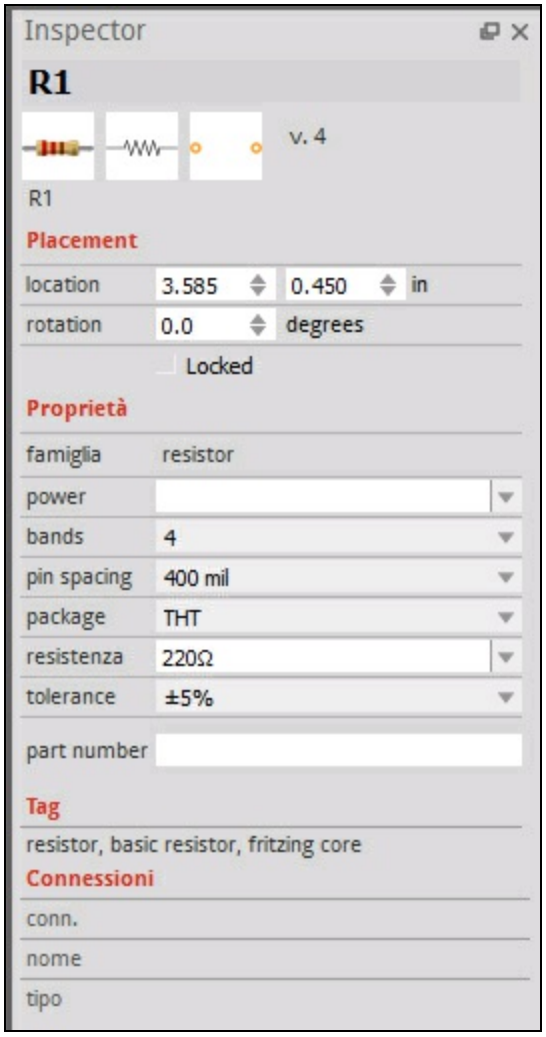


Figura 3.5 Pannello Inspector relativo a un resistore.

Come funziona?

Per chi non avesse mai usato una breadboard, quella di Fritzing si comporta esattamente come nella realtà, per cui è un ottimo sistema per imparare.

I fori delle linee di alimentazione di colore rosso (positivo) e blu (massa) sono comunicanti in senso orizzontale e di solito servono per “alimentare” la breadboard.

I fori delle due sezioni della breadboard comunicano in senso verticale, per cui bisogna creare dei “ponti” fra i componenti solo in senso verticale.

La breadboard è composta da due metà, una superiore e una inferiore, che non comunicano fra loro.

In Fritzing, facendo clic su un foro, viene colorato in giallo tutto il percorso di comunicazione con gli altri fori collegati (Figura 3.6a).

Quando i componenti e i fili sono infilati correttamente nei fori della breadboard, il percorso di collegamento viene evidenziato in verde.

È da notare che i fili si possono piegare, allungare e accorciare a piacere con il semplice uso del mouse. Se nelle preferenze viene impostata l’opzione *Curvy lines and legs*, si possono creare collegamenti con fili “morbidi”.

Puntando il mouse in un punto qualsiasi del filo, si può creare automaticamente un punto per la piegatura (rigida o morbida).

Per togliere una piega, basta fare doppio clic su di essa. Questa operazione è possibile su tutte le viste.

Il filo ha un suo menu contestuale (Figura 3.6b) tramite il quale è possibile cambiare colore (solo vista breadboard e schema), aggiungere punti di piegatura, portare sotto o sopra e così via.

Creare un circuito stampato

Quando si passa dalla vista *Breadboard* alla vista *PCB*, di solito il layout è disordinato. Se non si vuole rifare tutto da zero, ci sono alcune cose che si possono fare.

Innanzitutto, è necessario spostare manualmente all’interno della basetta i componenti e i collegamenti eventualmente rimasti all’esterno.

Basta selezionare gli elementi con il mouse e portarli all'interno del rettangolo corrispondente alle dimensioni della basetta.

Dopodiché, a seconda della complessità del circuito, è consigliabile impostare uno o due layer (strati). Per fare questo, è sufficiente andare alla voce *layers* dall'Inspector e impostare il valore su *one layer (single sided)* o su *two layers (double sided)*. Questa operazione è molto importante ai fini della disposizione delle piste ramate, dei componenti e delle serigrafie.

A questo punto, si può tentare un posizionamento automatico delle piste facendo clic sul pulsante *Autoinstrada*. L'instradamento automatico posiziona le piste in modo ottimale, ma non definitivo. Bisogna agire con il mouse per spostare le piste e piegarle in modo tale che siano sempre a distanza di sicurezza le une dalle altre.

Alla fine del lavoro, è possibile riempire di rame gli spazi vuoti in modo automatico tramite il comando *Copper Fill* del menu *Routing*. Questa operazione è consigliata, perché la ramatura funge da schermo per eventuali interferenze radio. In fase di stampa, quest'area verrà ricoperta dal Solder Mask, ovvero la maschera di saldatura.

Dallo stesso menu *Routing*, è consigliabile effettuare sempre un controllo delle piste tramite il comando *Design Rules Check* (Ctrl+Shift+D). Se le piste sono troppo vicine o se ci sono degli errori di sovrapposizione, apparirà un avviso e bisognerà correggere manualmente gli errori segnalati in rosso e ripetere il controllo finché appare l'avviso che tutto è OK.

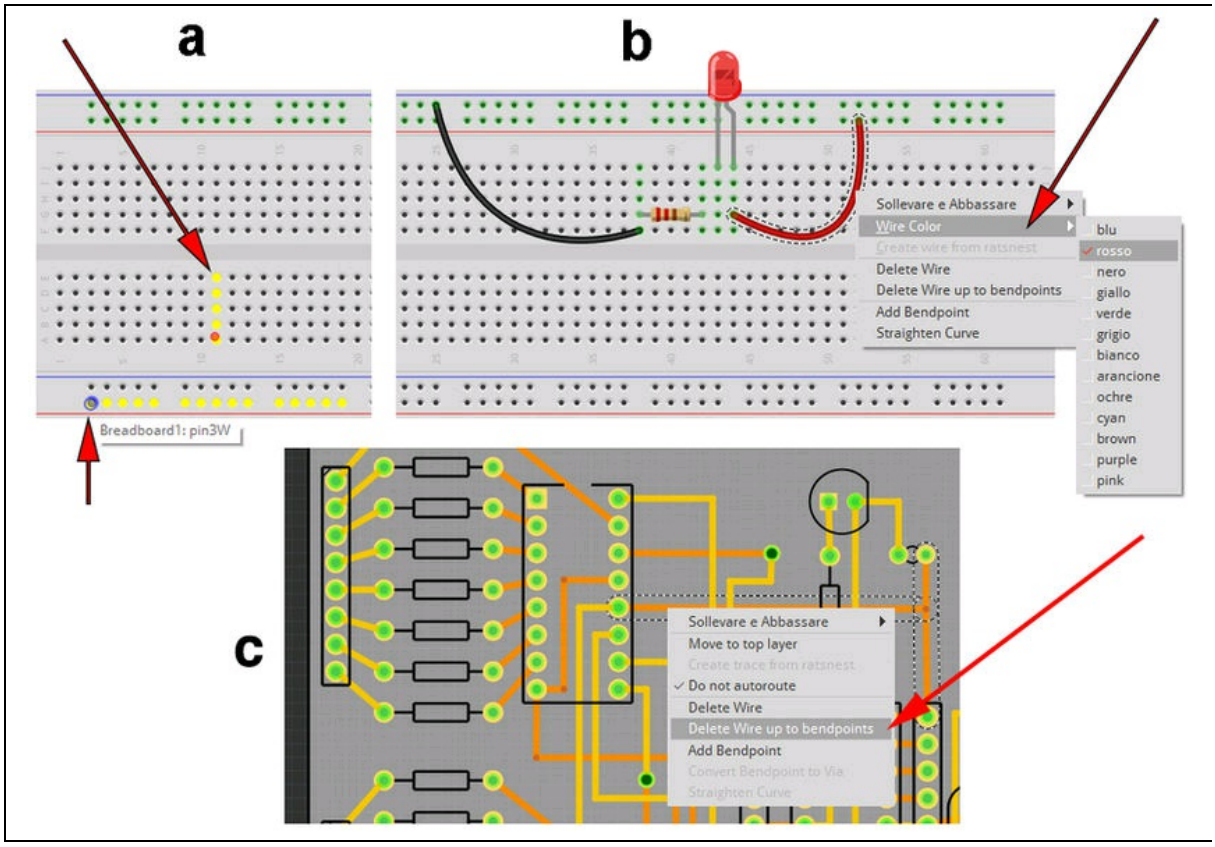


Figura 3.6 I percorsi evidenziati con il mouse (a). Il menu contestuale di un filo nella vista Breadboard (b). Il menu contestuale di una pista nella vista PCB (c).

Nella vista PCB, il menu contestuale può servire a portare la pista sul layer superiore (*move to top layer*) o sul layer inferiore (*move to bottom layer*), quando si progettano basette a doppio lato. Dallo stesso menu, È comoda anche la funzione che permette anche di eliminare un collegamento fino al primo gomito (*delete wire up to bendpoint*), per evitare di dover ricollegare tutta la pista da capo (Figura 3.6c).

Si fa notare che, nella vista PCB, quando risulta impossibile creare collegamenti non sovrapposti per l'affollamento delle piste, è possibile sfruttare i *via*, ovvero dei fori metallizzati che uniscono il livello superiore con quello inferiore.

Anche ogni componente ha un suo menu contestuale, tramite il quale è possibile ruotarlo (45, 90 e 180 gradi, in senso orario e antiorario),

portarlo in primo piano o in fondo, bloccarlo, eliminarlo. Da questo menu è possibile anche togliere i collegamenti *ratsnest* (nido di ratto) che appaiono come linee tratteggiate quando si eliminano componenti ritenuti essenziali dal software, ma che in realtà non servono più e rimangono visibili come “sporcizia”. Se invece si vuole ripristinare il collegamento perduto, lo si può ricreare con un doppio clic su un ratsnest.

Esempi pratici

Per fare pratica con Fritzing è consigliabile aprire alcuni dei numerosi esempi pratici dal menu *File > Apri esempio*, riguardanti la breadboard e il relativo schema/PCB, suddivisi per tipologia. Si fa notare che tutti i progetti sono perfetti nelle tre viste, per cui spesso è meglio partire da uno di questi esempi, se si vuole risparmiare tempo con le operazioni di instradamento.

- *Arduino*: progetti con Arduino completi di riferimenti agli sketch online.
- *Fritzing Starter Kit*: progetti con Arduino e Fritzing Starter Kit, un kit per principianti messo a disposizione dal sito `fritzing.org`.
- *Fritzing Creator Kit*: progetti con Arduino e Fritzing Super Upgrade Kit, un kit avanzato messo a disposizione dal sito `fritzing.org`.
- *Fritzing Workshop*: progetti proposti per un laboratorio.
- *All*: lista completa di tutti i progetti.

Raspberry Pi: un semplice progetto per controllare un LED

La Figura 3.7 illustra un esempio di collegamento di un LED alle porte GPIO di Raspberry Pi. Il progetto utilizza solo un LED e un resistore da 220 Ω (codice colore: rosso, rosso, marrone).

I componenti vanno semplicemente trascinati dalla finestra *Componenti* alla breadboard. I collegamenti vanno fatti trascinando il mouse da un punto all'altro della breadboard e verso i pin della porta GPIO.

- Dal pannello *Componenti* > *Computer*, selezionare l'icona con il lampone (Raspberry Pi) e trascinarla nella finestra del progetto. Nell'Inspector si può impostare anche il tipo di scheda. Nel nostro caso, Raspberry Pi 3.
- Il filo nero va collegato a GND (Ground) della porta GPIO e al terminale negativo (catodo) del LED.
- Il filo rosso va collegato al piedino della porta GPIO 17 e a un terminale del resistore.
- L'altro terminale del resistore va collegato al terminale positivo (anodo) del LED.

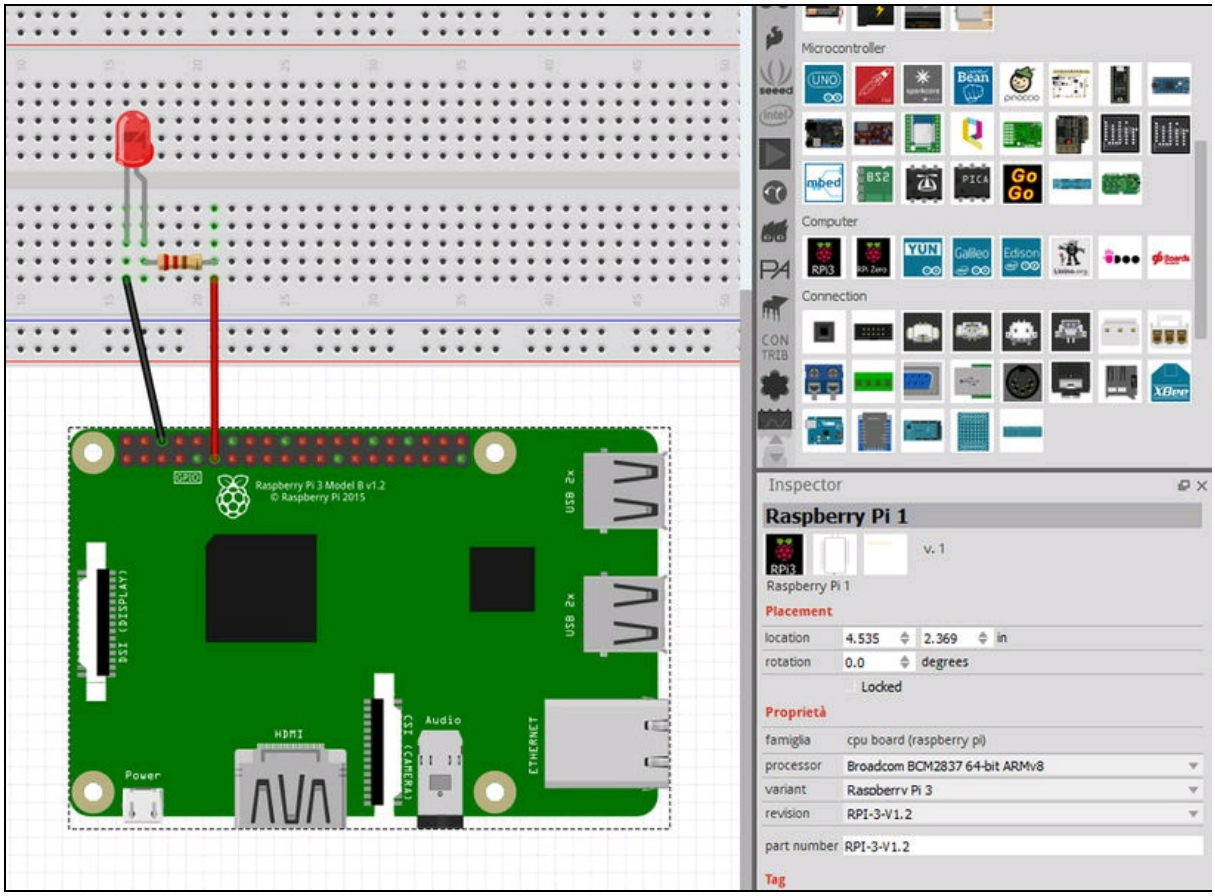


Figura 3.7 Il progetto LED nella vista Breadboard di Fritzing.

Per completezza di informazioni riportiamo anche la vista schema elettrico (Figura 3.8a) e la vista PCB (Figura 3.8b).

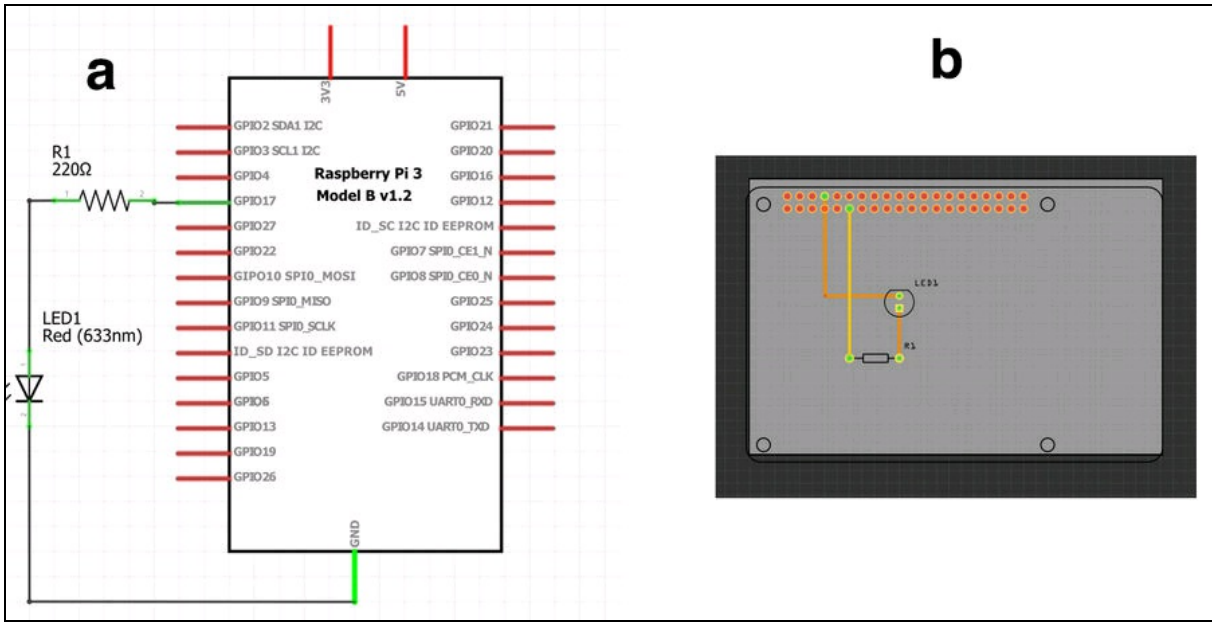


Figura 3.8 Vista Schema (a) e vista PCB (b) del progetto LED con Raspberry Pi.

Stampa PCB

Fritzing Fab (*Powered by AISLER*) è un servizio online di stampa PCB fornito dal sito `fritzing.org` e, dal 2017, potenziato da AISLER, un servizio di produzione avanzato, sempre in Germania, con qualità ancora più elevata, ma a costi più accessibili.

Una volta terminato il progetto, è sufficiente effettuare l'upload del file in formato Fritzing e ordinare la quantità desiderata di circuiti stampati.

Il nuovo servizio AISLER ha aggiunto la possibilità di visualizzare un'anteprima online della propria basetta, mostrando i vari layer separatamente, le serigrafie, le forature e così via.

È possibile effettuare ordini anche di un solo circuito stampato, ma è consigliabile aumentare il numero ad almeno cinque o dieci unità per risparmiare sui costi.

Il pagamento avviene online con consegna a domicilio entro un paio di settimane. La qualità del prodotto giustifica il costo decisamente più alto rispetto ad altri stampatori di PCB: ne vale veramente la pena. Il prodotto è molto curato e viene controllato prima di essere stampato. La serigrafia standard è nera su Solder Mask verde.

Esportazione del file di produzione

Come in un vero programma professionale, Fritzing offre varie opzioni per l'esportazione del file di produzione.

Oltre alla voce di menu *Order PCB* c'è il pulsante *Fabricate* che porta al sito per il calcolo dei costi di produzione e invio dell'ordine online al Fritzing Fab.

In alternativa, facendo clic sul pulsante *Export for PCB* o dal menu *File* > *Esporta*, è possibile accedere a una serie di opzioni per la produzione in proprio.

- *Etchable (PDF)*: documento PDF per l'incisione.
- *Etchable (SVG)*: documento SVG per l'incisione.
- *Etchable mirrored (SVG)*: documento SVG speculare per l'incisione.
- *Extended Gerber (RS-274X)*: documento Gerber (per servizi di stampa PCB professionali).

Si può esportare il progetto anche in vari formati immagine, come lista componenti e XML Netlist. Esportando come documento *Etchable (PDF)* si può stampare il circuito su un foglio di acetato con una stampante laser e, una volta impressionato e sviluppato il circuito su una basetta ramata presensibilizzata come spiegato nel Capitolo 1, si potrà inciderla in un bagno di acido o cloruro ferrico.

NOTA

Per mandare a un servizio di stampa il file di produzione, nella quasi totalità dei casi, è necessario esportarlo nel formato Extended Gerber (RS-274X). Il file

Gerber è stato introdotto da Gerber Systems Corp. per la produzione industriale di PCB su macchine CNC. Lo standard industriale prevede la presenza di un numero qualsiasi di file inerenti le varie fasi di lavorazione: incisione, maschere di saldatura, fresatura, foratura, serigrafia e così via. I file sono in formato testo, pertanto il primo controllo visivo può essere effettuato aprendoli con un Gerber Viewer, per esempio l'ottimo Online Gerber Viewer (http://www.gerber-viewer.com) che consente di controllare gratuitamente tutti i file di produzione.

IDE di Arduino

Anche se non è questa la sede per approfondire i temi sulla programmazione di Arduino, diamo un'occhiata all'ambiente di sviluppo integrato, IDE (*Integrated Development Environment*), di Arduino. La sua comprensione servirà a realizzare più facilmente i programmi relativi ad alcuni progetti di questo libro.

NOTA

Anche se l'IDE di Arduino può servire a una miriade di altre schede anche non-Arduino, in questo paragrafo si farà riferimento solo alla scheda Arduino UNO.

Installazione

Si fa notare che da gennaio 2017, la bega legale fra i detentori dei diritti del marchio Arduino è terminata, per cui i due siti, benché propongano tuttora prodotti diversificati, hanno unificato l'uso dell'IDE di Arduino.

La versione di riferimento usata in questo libro è la 1.8.2 (maggio 2017) e si può scaricare liberamente dalla sezione *Software* dai siti ufficiali `arduino.cc` o `arduino.org`.

Sono disponibili le versioni per sistemi operativi Windows 32/64 bit, Mac OS X 10.7 o superiore, Linux 32/64 bit e Linux ARM, come Source Code per sviluppatori e come Arduino Web Editor, ovvero come IDE utilizzabile online.

Per gli utenti Linux o ARM, per esempio, (Raspberry Pi) è possibile installare l'IDE anche tramite il comando `apt-get install arduino` da terminale oppure tramite Synaptic e gestori simili.

Qualsiasi piattaforma e sistema operativo si usi, una volta lanciato il file eseguibile `arduino`, si aprirà l'interfaccia principale (Figura 3.9). Si tratta di una finestra che riporta il nome dello *sketch*, così viene

chiamato il programma in ambiente Arduino, con due funzioni obbligatorie:

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

All’interno della funzione `setup()` si può scrivere il codice che viene eseguito una sola volta all’avvio dello sketch, ovvero ogni volta che la scheda viene accesa.

All’interno della funzione `loop()` si può scrivere il codice che viene eseguito ciclicamente fintantoché la scheda rimane accesa.

Collegando per la prima volta una scheda Arduino a una porta USB del computer, viene installato automaticamente il driver FTDI, che crea una porta seriale virtuale COM (nei sistemi Windows) o tty (nei sistemi Mac OS o Linux). In questo modo il driver virtuale consente la comunicazione dei dati seriali fra l’IDE e la scheda Arduino collegata via USB.

La prima cosa da fare è controllare che nel menu *Strumenti* > *Porta* sia presente una porta seriale COM*x* oppure tty*x*, dove *x* è un numero arbitrario della porta, dato dal sistema. Controllare anche la selezione del tipo di scheda collegata, dal menu *Strumenti* > *Scheda*. Nel caso si usi una scheda Arduino UNO, è necessario selezionarla nella lista.

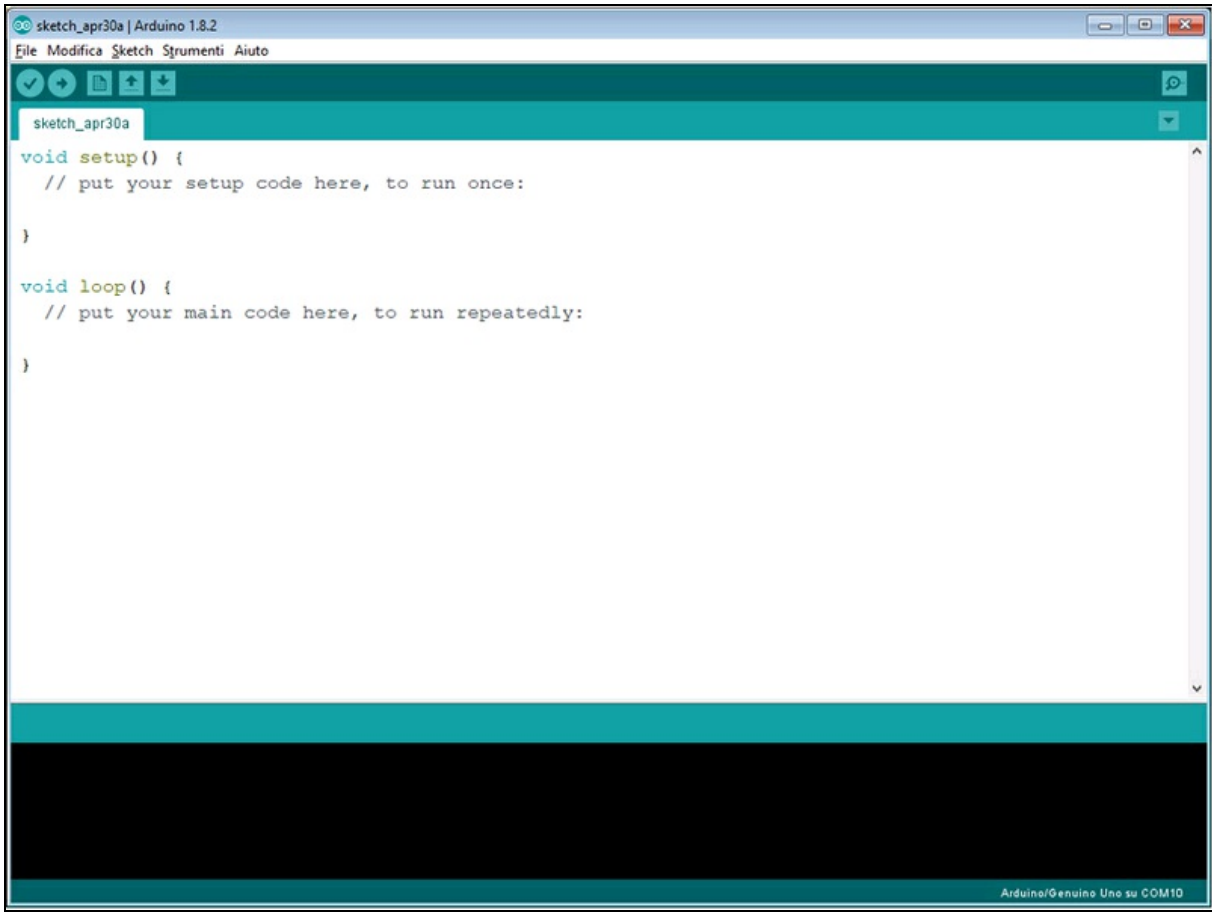


Figura 3.9 IDE di Arduino.

Esempi

Per provare che tutto funzioni, dal menu *File* si può accedere alla voce *Esempi* per aprire un esempio dai molti disponibili. Il primo esempio che di solito si carica è il tipico “Hello World” dell’elettronica, ovvero *Blink*, accessibile dal menu *File* > *Esempi* > *1.Basics* > *Blink*.

Si tratta di un semplice programma che fa lampeggiare il LED interno della scheda o, volendo, un LED esterno collegato al pin 13 della scheda.

Una volta caricato Blink, si aprirà una nuova finestra simile a quella di Figura 3.10, con tutto il listato del programma commentato riga per

riga.

A beneficio dei principianti e di chi non è avvezzo al linguaggio di programmazione, diamo una breve spiegazione del listato.

Per prima cosa, seguendo fedelmente la sintassi del linguaggio C++, le indicazioni di commento alle righe di codice sono di due tipi.

- Viene usata la doppia barra `//` per commentare una riga isolata. Il testo apparirà in grigio dopo il segno `//`.
- Viene usata la barra più l'asterisco `/*` all'inizio di un blocco di testo e l'asterisco più la barra `*/` alla fine del blocco di testo. Tutto il testo all'interno di `/*` e `*/` verrà visualizzato in grigio.
- Le righe di commento non vengono compilate e non influiscono minimamente nelle dimensioni del programma.

Altre regole di base della sintassi:

- le istruzioni devono terminare sempre con `;` (punto e virgola);
- ogni funzione deve contenere il codice all'interno di una parentesi graffa aperta `{` all'inizio e una parentesi graffa chiusa `}` alla fine;
- si possono usare spazi, tabulazioni e indentazioni di qualsiasi tipo per la formattazione del testo; l'interfaccia intercetta automaticamente la posizione del testo nella finestra, la posizione dei punti e virgola di fine istruzione e delle parentesi graffe.

Listato originale dello sketch Blink, con i commenti tradotti in italiano:

```
// La funzione setup viene eseguita una volta sola quando si preme reset o si
alimenta la scheda
void setup() {
    // Inizializza il pin digitale LED_BUILTIN come uscita (corrispondente al pin
13 di Arduino UNO
    pinMode(LED_BUILTIN, OUTPUT);
}
// La funzione loop viene eseguita ciclicamente
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);       // Accende il LED (HIGH = livello di
tensione 5 volt)
    delay(1000);              // Attesa di un secondo
```

```
    digitalWrite(LED_BUILTIN, LOW); // Spegne il LED (LOW = 0 volt)
    delay(1000);            // Attesa di un secondo
}
```

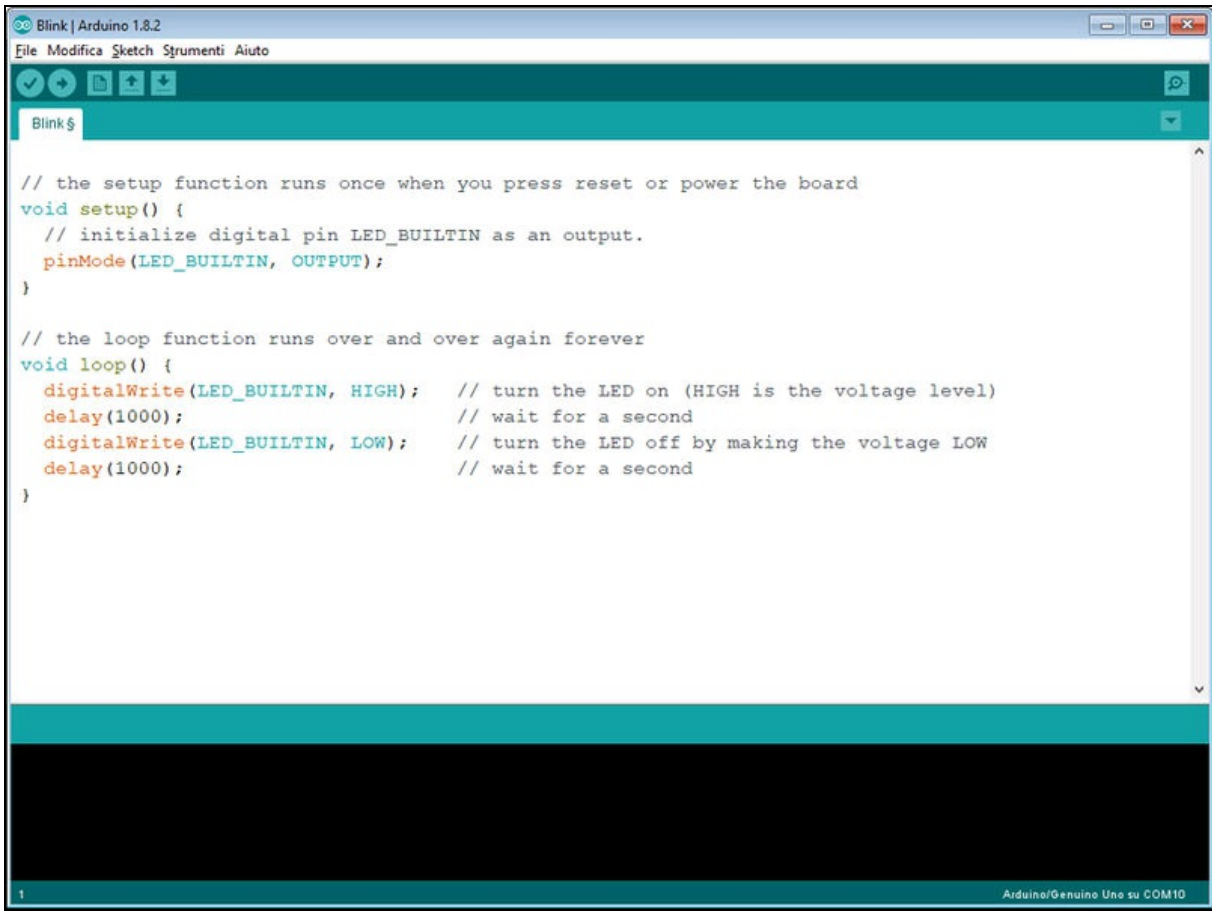


Figura 3.10 L'esempio Blink caricato nell'IDE.

Rispetto al C++, la sintassi di programmazione nell'ambiente Arduino è stata molto semplificata. Anche se è possibile sviluppare programmi usando il linguaggio C++, la maggior parte delle funzioni è stata "tradotta" con termini più facili da ricordare.

Analizzando il breve codice di esempio dello sketch Blink si possono capire molte cose del linguaggio utilizzato.

- La funzione `pinMode(LED_BUILTIN, OUTPUT)` all'interno della funzione `setup()` imposta la modalità del pin interno 13 come `OUTPUT` (tutto

maiuscolo). In altre parole, la porta 13 di Arduino UNO viene impostata per la trasmissione di dati digitali in uscita;

- La funzione `digitalWrite(LED_BUILTIN, OUTPUT, HIGH)` non fa altro che eseguire un comando simile a “scrivi sulla porta digitale 13 un valore HIGH”. Scrivendo un valore `HIGH` (tutto maiuscolo), si ottiene una tensione di 5 volt, ovvero la porta viene alzata allo stato logico 1. Tant’è che si può scrivere l’istruzione anche in questo modo: `digitalWrite(13, 1);`.
- La funzione `delay(1000)` introduce un ritardo di 1000 millisecondi, cioè di un secondo. Volendo modificare il ritardo è possibile scrivere un qualsiasi valore di millisecondi;
- La funzione `digitalWrite(LED_BUILTIN, OUTPUT, LOW)` esegue il comando “scrivi sulla porta digitale 13 un valore LOW”. Scrivendo un valore `LOW` (tutto maiuscolo), si ottiene una tensione di 0 volt, ovvero la porta viene abbassata allo stato logico 0. Anche in questo caso, si può scrivere `digitalWrite(13, 0)`.
- Di nuovo la funzione `delay(1000)` introduce un ritardo di 1 secondo e il ciclo ricomincia.

Da queste poche righe di codice si intuisce la semplicità con cui è stata pensata la programmazione di una scheda Arduino o di una scheda compatibile.

Verifica e compilazione

Ogni sketch deve essere compilato prima di venire caricato nella memoria del processore della scheda. Nella barra degli strumenti, il pulsante *Verifica* (l’icona con il segno di spunta) consente di verificare l’esattezza della sintassi prima della compilazione, mentre il pulsante

Carica (l’icona con la freccia) esegue la compilazione e il caricamento dello sketch nella memoria del processore.

Facendo clic sul pulsante *Verifica*, il listato viene compilato e, se non ci sono errori, nella barra della finestra di output dopo qualche secondo appare la scritta *Compilazione terminata*. Nella finestra di output viene riportata anche l’indicazione della dimensione di memoria occupata dallo sketch e della memoria disponibile (massimo 32 KB).

In caso di errore di compilazione, verranno segnalati in questa stessa finestra tutti gli errori di sintassi e i riferimenti alle righe di codice che li contengono. Se non si riesce a ottenere la compilazione del codice, è inutile procedere alla fase successiva del caricamento in memoria, in quanto il processore non può caricare un programma contenente anche un solo errore, perché ne causerebbe il crash. Si consiglia sempre di verificare la sintassi prima del caricamento in memoria.

Caricamento in memoria

Una volta compilato con successo il codice, è possibile caricare il programma nella memoria flash del processore, ovvero nella sua area di memoria riscrivibile.

Facendo clic sul pulsante Carica, il codice verrà compilato nuovamente e caricato in memoria del processore. Nel frattempo una barra di progresso visualizzerà lo stato della scrittura. Al termine del caricamento, se non si verificano problemi, apparirà la scritta *Caricamento terminato*. Se per qualche motivo non imputabile a errori di compilazione il caricamento non dovesse andare a buon fine, nella stessa finestra apparirà l’indicazione del problema. Spesso succede che la porta virtuale non sia selezionata o che non sia stata selezionata la scheda corretta. È sufficiente verificare questi problemi dal menu *Strumenti* e riprovare il caricamento. Se i problemi persistono, verificare l’installazione del driver seriale.

Una volta caricato in memoria lo sketch *Blink*, si vedrà il LED incorporato lampeggiare a intervalli di un secondo. Volendo, si può inserire un LED al pin 13 (anodo) e al pin GND (catodo), come illustrato nella Figura 3.11.

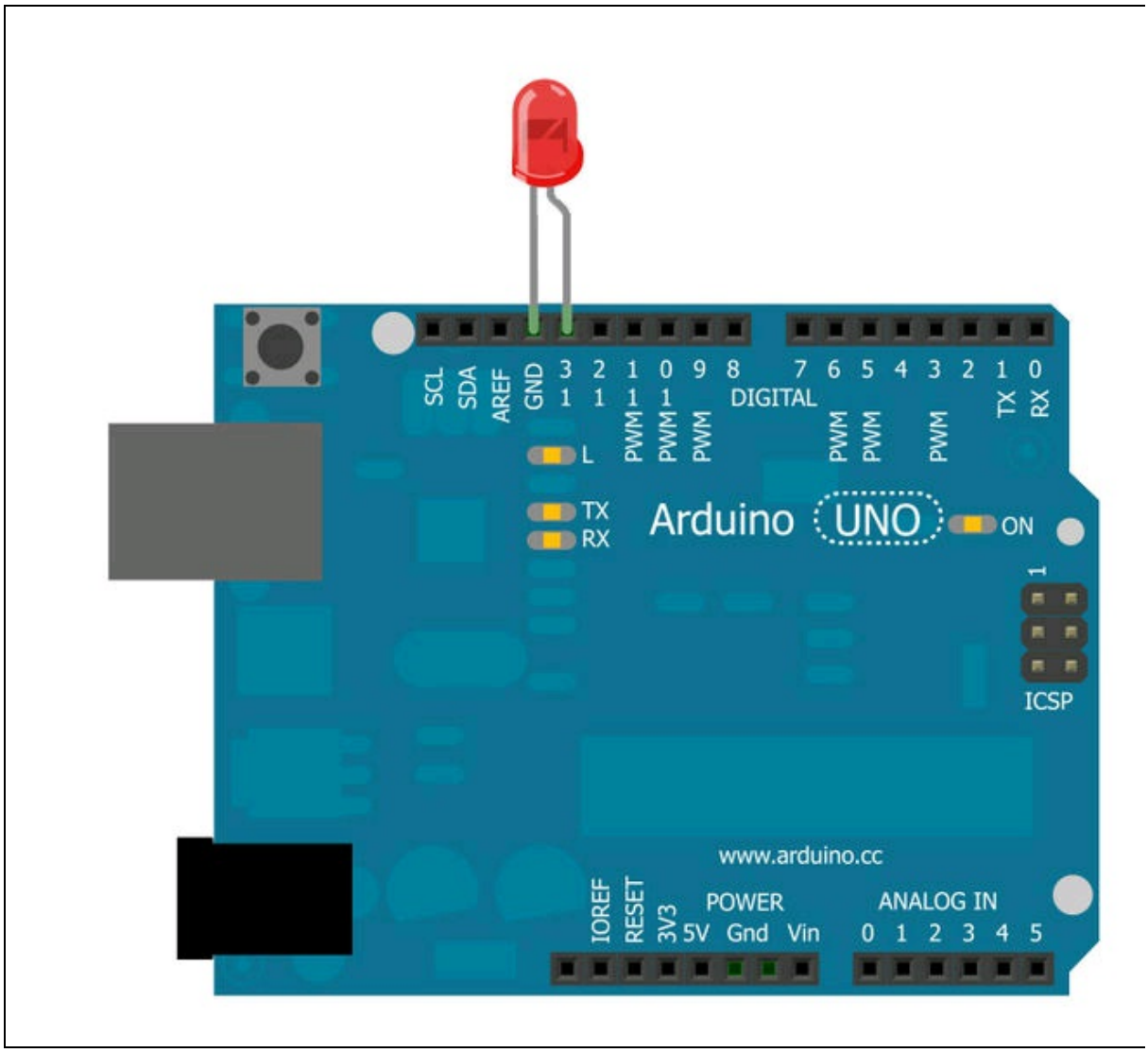


Figura 3.11 Un LED collegato al pin 13 e al pin GND della scheda Arduino UNO.

Con riferimento alle caratteristiche della scheda Arduino UNO, i 14 pin digitali (numerati da 0 a 13) possono venire usati come ingressi e uscite di*gitali* (`digitalWrite`, 1-0).

I pin digitali 3, 5, 6, 9, 10 e 11 sono anche PWM, cioè permettono il controllo della cosiddetta *Pulse Width Modulation* (modulazione della larghezza di impulso) per cui si possono scrivere valori digitali impulsivi tramite un byte da 0 a 255. In questo caso si può scrivere l'istruzione `analogWrite`, 0-255 in modo da simulare un comportamento quasi analogico in uscita.

I 6 pin analogici (numerati da A0 ad A5) possono venire usati come ingressi analogici con l'istruzione `analogRead`, A0-A5. Con una risoluzione di 10 bit si otterranno letture digitali da 0 a 1023. Questi ingressi si possono impostare anche come uscite digitali, esattamente con gli altri pin digitali.

Monitor seriale

L'IDE di Arduino non consente un debug vero e proprio della scheda collegata. È comunque possibile visualizzare quello che succede all'interno della scheda sfruttando il monitor seriale. La finestra del monitor seriale è in pratica una console di comunicazione seriale RX/TX con l'interfaccia UART della scheda e permette la lettura o scrittura diretta di dati. È accessibile dal menu *Strumenti* oppure facendo clic sull'icona della lente sulla barra in alto a destra. È altresì possibile aprirla con la scorciatoia da tastiera CTRL+MAIUSC+M.

Nella finestra del monitor seriale si possono stampare in tempo reale tutte le informazioni di debug, inserendo in qualsiasi punto dello sketch l'istruzione `Serial.print()` oppure `Serial.println()` per stampare anche un "a capo".

Per attivare il monitor seriale all'interno di uno sketch è necessaria l'istruzione `Serial.begin(x)` nella funzione `setup()` dello sketch, dove *x* è la velocita di trasmissione seriale in baud, solitamente `9600`.

Per esempio, modificando l'esempio Blink:

```
void setup() {
    pinMode(LED_BUILTIN,OUTPUT);
    Serial.begin(9600);   // inizializzazione della porta seriale a 9600 baud
}
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("LED ACCESO");   // stampa LED ACCESO
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    Serial.println("LED SPENTO");   // stampa LED ACCESO
    delay(1000);
}
```

Aprendo il monitor seriale si vedranno le stringhe `LED ACCESO` e `LED SPENTO` in corrispondenza dell'accensione e lo spegnimento del LED (Figura 3.12). Se si cambia la velocità di comunicazione nel `setup()`, ricordarsi di cambiarla anche nel monitor seriale aprendo il menu che si trova in basso a destra della finestra.

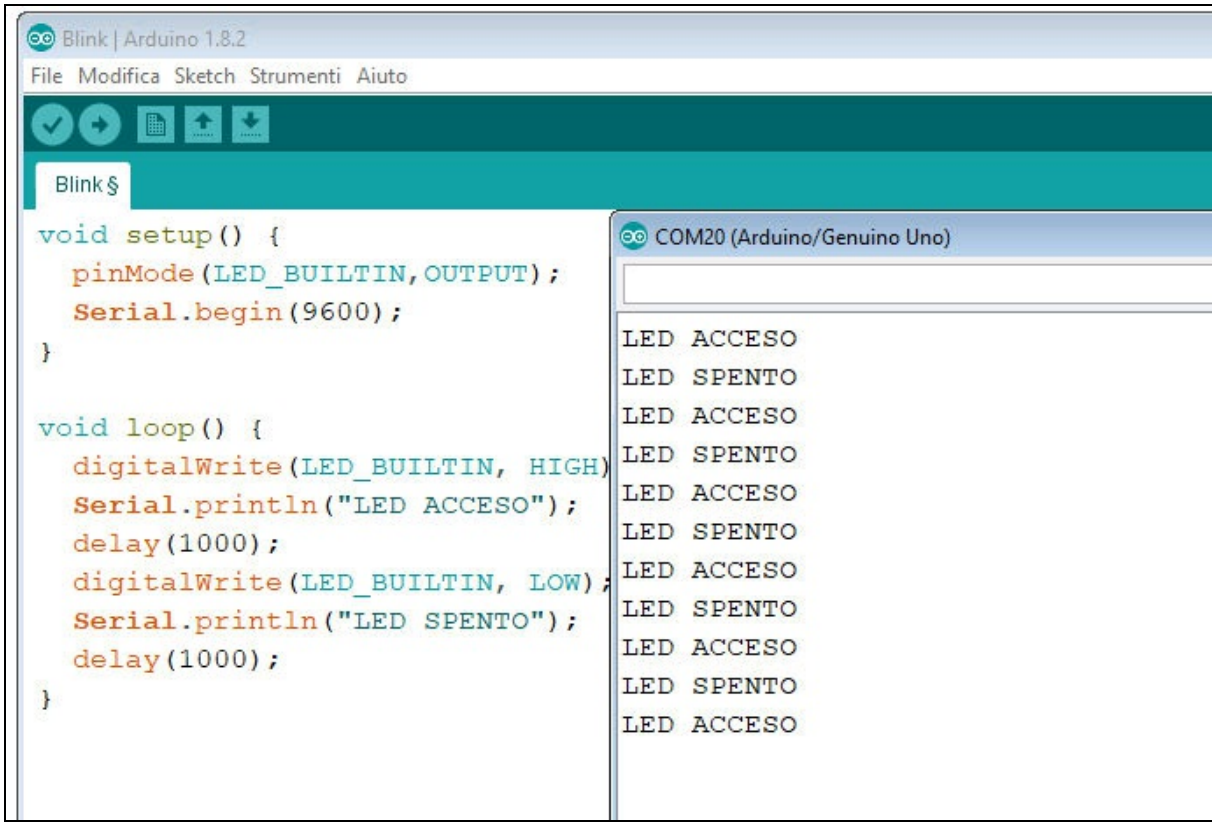


Figura 3.12 La finestra del monitor seriale.

Shield

Probabilmente, l’idea vincente che ha decretato il successo di Arduino fin dalla nascita è stata la possibilità di “impilare” schede aggiuntive, chiamate *shield*, sopra una scheda Arduino in corrispondenza delle strisce di pin, che poi vengono replicate nella scheda shield per essere ulteriormente usate.

Esiste una vasta produzione di *shield* originali e di terze parti, che aumentano notevolmente le prestazioni di base di una scheda Arduino.

Ecco un breve elenco di shield disponibili in commercio e utili per progetti IoT.

- *Arduino WiFi Shield 101*: shield Wi-Fi per la connettività Internet (Figura 3.13a).
- *Arduino Ethernet Shield*: shield Ethernet per la connettività Internet (Figura 3.13b).
- *Arduino GSM Shield*: shield per la connettività cellulare (GSM) e Internet (GPRS) (Figura 3.13c).
- *Arduino Bluetooth Shield*: shield per la connettività Bluetooth (Figura 3.13d).

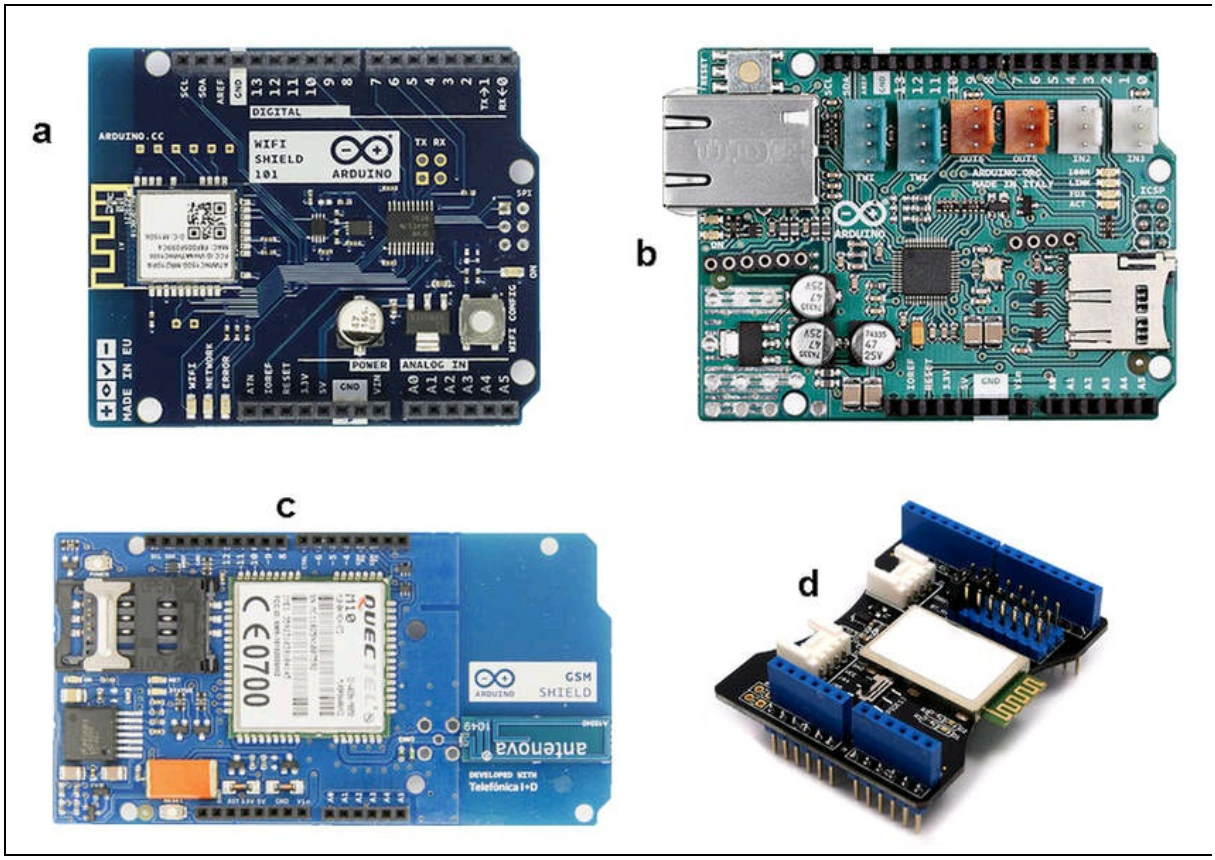


Figura 3.13 Arduino WiFi Shield 101 (a). Arduino Ethernet Shield (b). Arduino GSM Shield (c). Arduino Bluetooth Shield (d).

Gestore schede

Da alcune versioni dell'IDE di Arduino ha assunto un'importanza determinante per l'utilizzo di nuove schede Arduino e, soprattutto, di schede di terze parti.

Per essere più chiari, è possibile usare l'IDE di Arduino per programmare il modulo ESP8266 oppure la scheda Siemens SIMATIC IO2020, così come un chip Atmel Attiny e così via.

L'operazione di aggiornamento del gestore schede avviene tramite una connessione Internet. Dal menu *Strumenti* > *Scheda* > *Gestore schede* si apre una finestra simile a quella illustrata nella Figura 3.14.

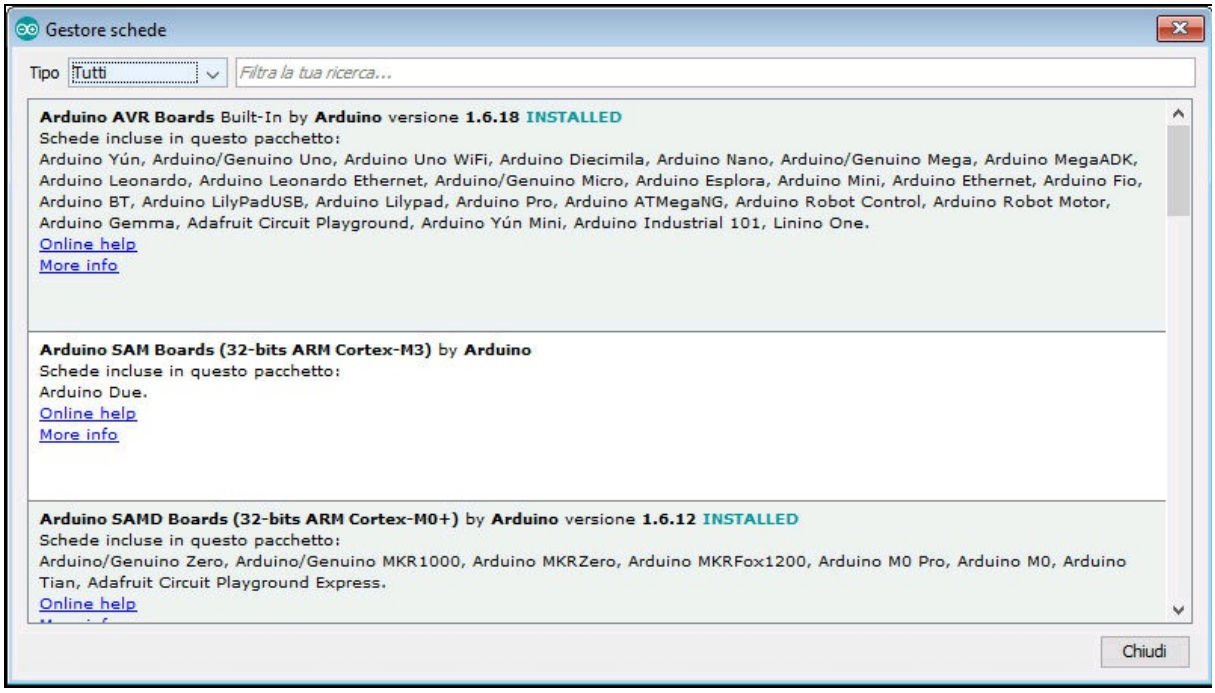


Figura 3.14 Gestore schede dell'IDE di Arduino.

In questa finestra è possibile applicare un filtro per la ricerca delle schede da installare o aggiornare. Aprendo il menu a tendina *Tipo*, si potrà scegliere:

- *Tutti*;
- *Aggiornabile*;
- *Arduino*;
- *Fornita da terze parti*;
- *Arduino Certified*;
- *Partner*;
- *Arduino @Heart*.

Una volta scelta la tipologia di schede da aggiungere, basta fare clic sul pulsante *Installa* e la scheda o il gruppo di schede della stessa famiglia verrà aggiunto automaticamente all'IDE. Le nuove schede aggiunte saranno disponibili nel menu *Strumenti*. Inoltre, per ogni

gruppo di nuove schede verranno aggiunti anche gli esempi di utilizzo nel menu *Esempi*.

URL aggiuntivo per il gestore schede

Per alcuni produttori di schede di terze parti, può accadere che venga reso disponibile un URL da aggiungere al gestore schede tramite la finestra *Impostazioni* dell'IDE. È il caso del modulo ESP8266 e ATtiny, cui dedichiamo i paragrafi seguenti.

Installazione della scheda ESP8266

Per installare e quindi poter usare l'IDE di Arduino per programmare un qualsiasi modulo ESP8266, è disponibile il seguente URL sul sito della community dedicata:

`http://arduino.esp8266.com/stable/package_esp8266com_index.json`.

Una volta aperta la finestra *Impostazioni* dal menu *File* dell'IDE, basterà incollare l'URL nello spazio *URL aggiuntivi per il Gestore schede*, come indicato dalla freccia della Figura 3.15a.

Dopodiché, basta aprire il Gestore schede e cercare il pannello di installazione della scheda ESP8266, come indicato dalla freccia della Figura 3.15b. Come si può vedere, le schede aggiunte sono tutte quelle supportate dalla community esp8266 (si veda il Capitolo 2).

Alla fine dell'installazione apparirà la scritta `INSTALLED` e tutti i modelli delle schede ESP8266 saranno disponibili nel menu *Strumenti* > *Scheda*.

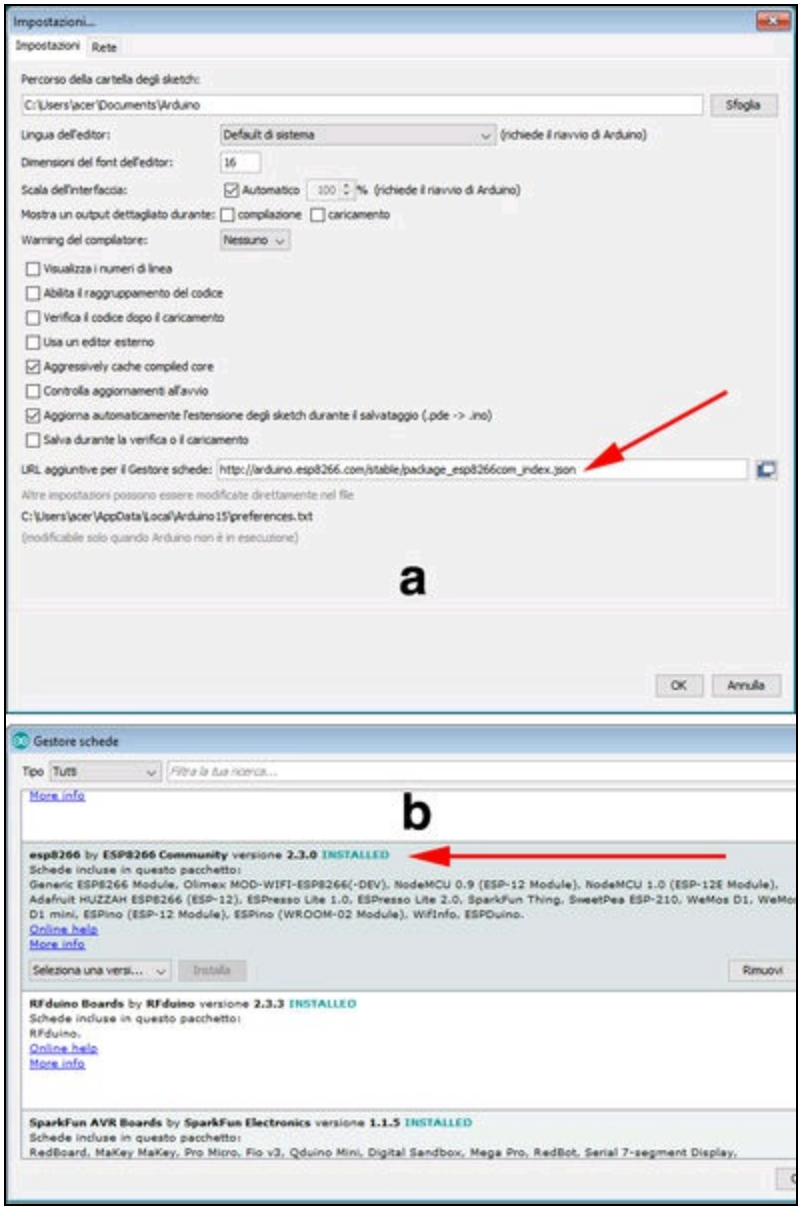


Figura 3.15 La finestra Impostazioni (a). Il pannello di installazione dei moduli ESP8266 (b).

Compilare e caricare uno sketch in un modulo ESP8266-01

Per la programmazione del firmware del modulo ESP8266-01 si sfrutta la sua porta UART. Si consiglia di collegare la porta UART del modulo ESP8266-01 a un adattatore USB FTDI regolato con tensione di 3,3 volt, disponibile anche su Amazon. Ricordiamo che il modulo

ESP8266 deve assolutamente essere alimentato a 3,3 volt, per cui un'alimentazione a 5 volt (per esempio, quella di Arduino) potrebbe bruciare velocemente il chip. Anche gli ingressi sono TTL 3.3 V e una ricezione di dati TTL 5 V non potrebbe funzionare o danneggiare il modulo (per esempio, il pin TX di Arduino).

La Figura 3.16a illustra un adattatore USB FTDI regolabile a 3,3 o 5 volt. Basta spostare il jumper sulla posizione di 3.3 V e si otterrà un'alimentazione e un livello TTL corretti per il modulo ESP8266.

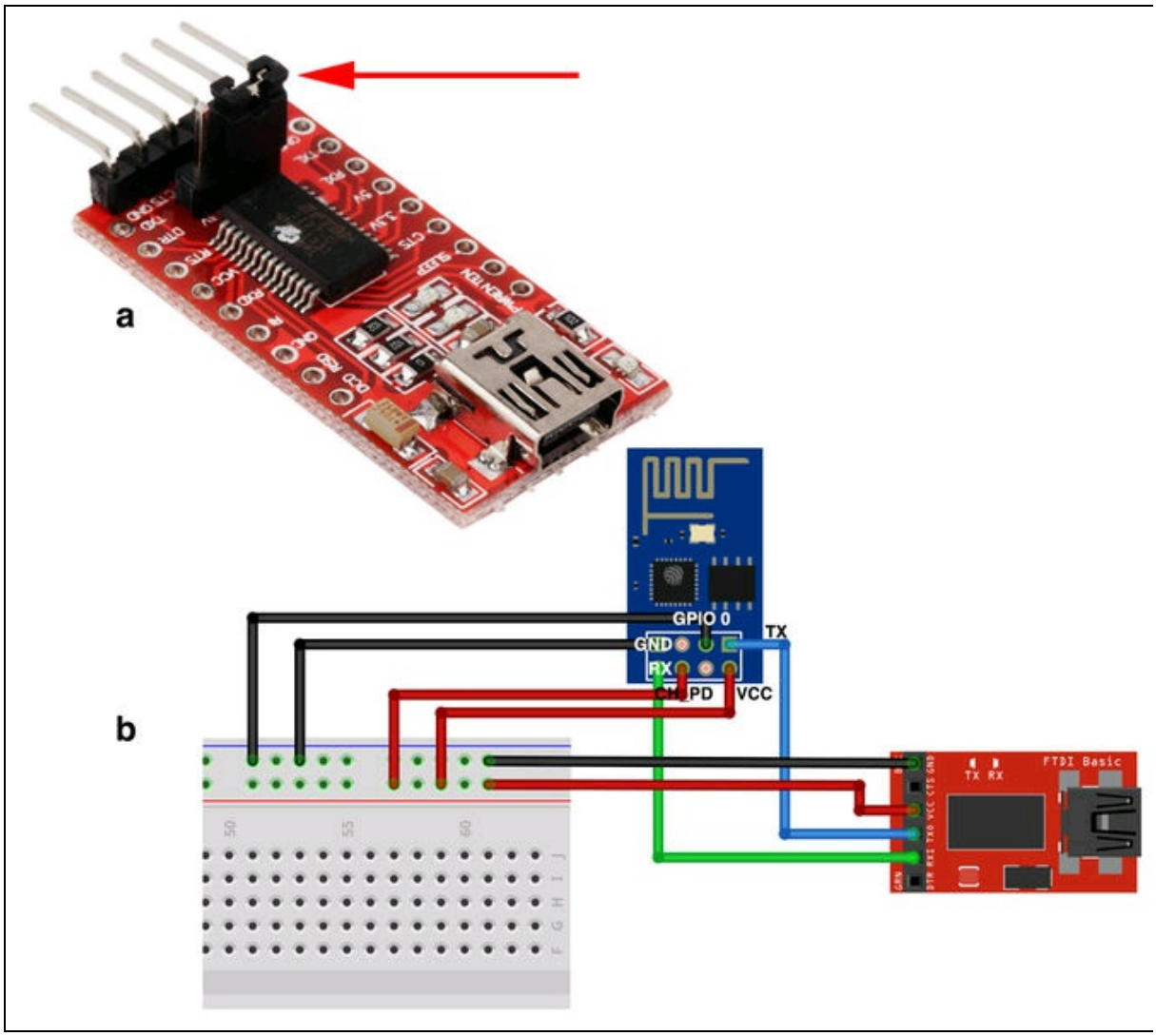


Figura 3.16 Adattatore USB FTDI regolabile a 3,3/5 volt (a). Collegamento dell'adattatore al modulo ESP8266-01 (b).

In alternativa si potrebbe usare anche una scheda Arduino UNO sfruttando il pin di alimentazione a 3,3 V. Ma sarebbe necessario anche un partitore di tensione o un *level shifter* per ridurre il livello TTL del pin TX. Tutta questa fatica viene eliminata con l'acquisto di un adattatore USB FTDI.

Il collegamento FTDI/ESP8266 è illustrato nella Figura 3.16b. La breadboard di appoggio serve solo per raddoppiare le alimentazioni ai pin VCC e CH_PD e per le masse che devono portate ai pin GND e GPIO_0.

I collegamenti da effettuare sono i seguenti:

Pin ESP8266	Pin FTDI
VCC	VCC
GND	GND
GPIO_0	GND
CH_PD	VCC
TX	RX
RX	TX

Si fa notare che il pin GPIO_0 a massa serve per impostare il modulo in modalità di programmazione UART. Dopo la programmazione questo pin deve essere scollegato dalla massa per impostare il modulo in modalità Flash, ovvero in modalità di esecuzione del programma in memoria.

Il pin CH_PD deve essere sempre alto, cioè alimentato a 3,3 volt per attivare il modulo.

Dopo la programmazione, la porta UART (RX/TX) può anche venire scollegata se non la si usa per altri scopi (si veda più avanti).

A questo punto, dal menu *Strumenti* dell'IDE di Arduino, bisogna selezionare la voce *Generic ESP8266 Module*, per caricare le librerie corrispondenti dal gestore schede, come indicato nella Figura 3.17. Quindi, sempre dal menu *Strumenti*, bisogna selezionare la porta di

comunicazione installata dall'adattatore USB FTDI (per esempio, *COMx* o *ttyx* dove *x* è un numero assegnato dal driver).

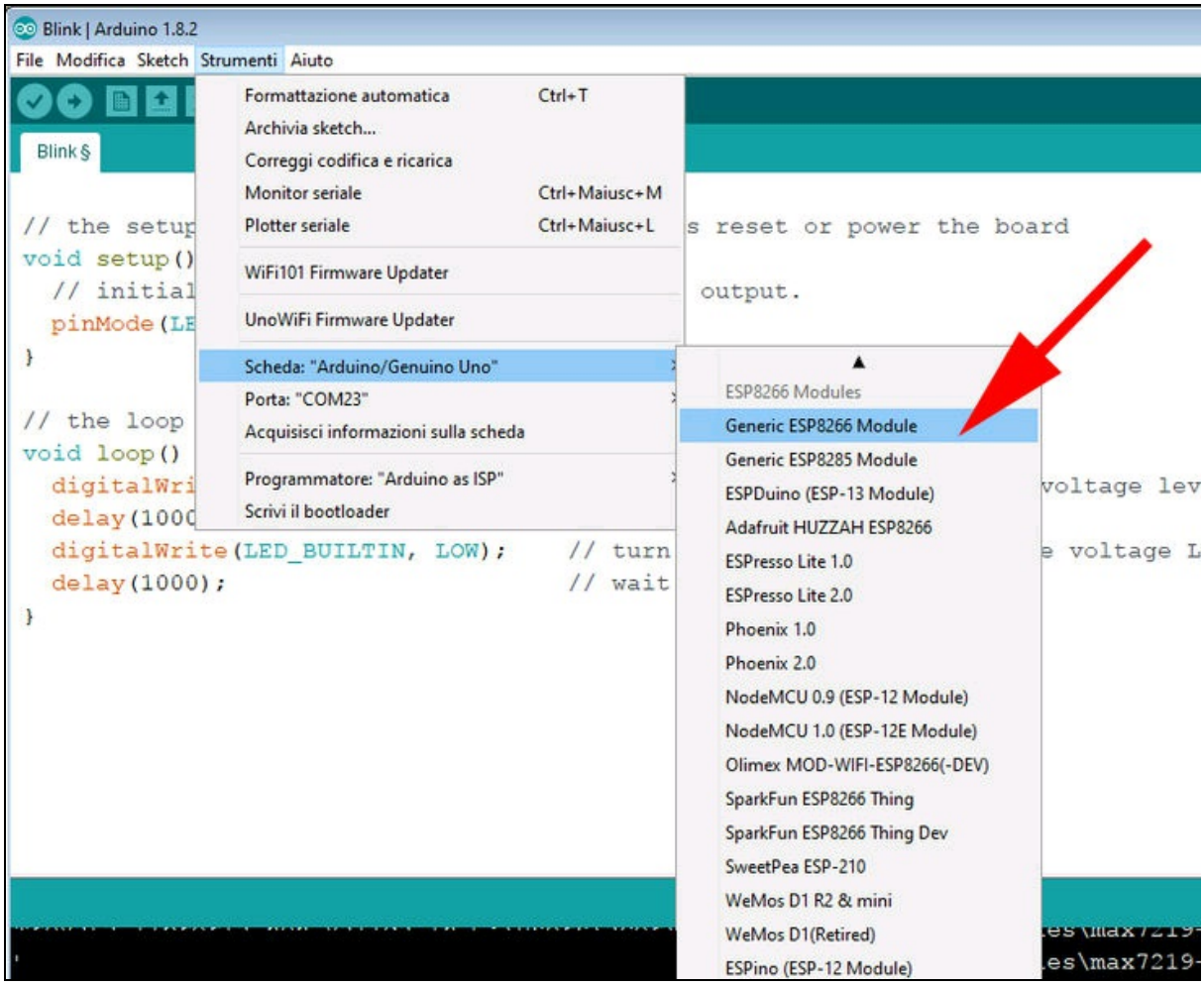


Figura 3.17 La selezione del modulo Generic ESP8266 Module dal menu Strumenti.

Per provare che tutto funzioni, basta caricare l'esempio Blink e modificare le tre righe dello sketch mettendo il valore `2` al posto di `LED_BUILTIN` come indicato dalle tre frecce in alto della Figura 3.18.

Dopo la compilazione, si vedrà nella finestra di output la progressione del caricamento nella memoria Flash del modulo con una serie di punti come indicato dalla freccia in basso della Figura 3.18.

Per vedere in funzione il led lampeggiante, basta collegare al pin GPIO_2 l'anodo di un LED con un resistore da 220 Ω in serie e il suo

catodo a massa.

Scollegare il cavo USB per togliere l'alimentazione al modulo ESP8266 e scollegare il pin GPIO_0 dalla massa della breadboard. Riattivare l'alimentazione collegando nuovamente il cavo USB. In questo modo il modulo ESP8266 farà il boot del programma in memoria e inizierà a far lampeggiare il LED collegato.

Se questo piccolo esperimento funziona, funzioneranno sicuramente anche tutti gli altri. E siccome abbiamo a che fare con un modulo Wi-Fi, vale la pena provare un esempio che si chiama *WiFiWebServer*, raggiungibile da *File > Esempi > Esempi per Generic ESP8266 Module > WiFi > WiFiWebServer*.

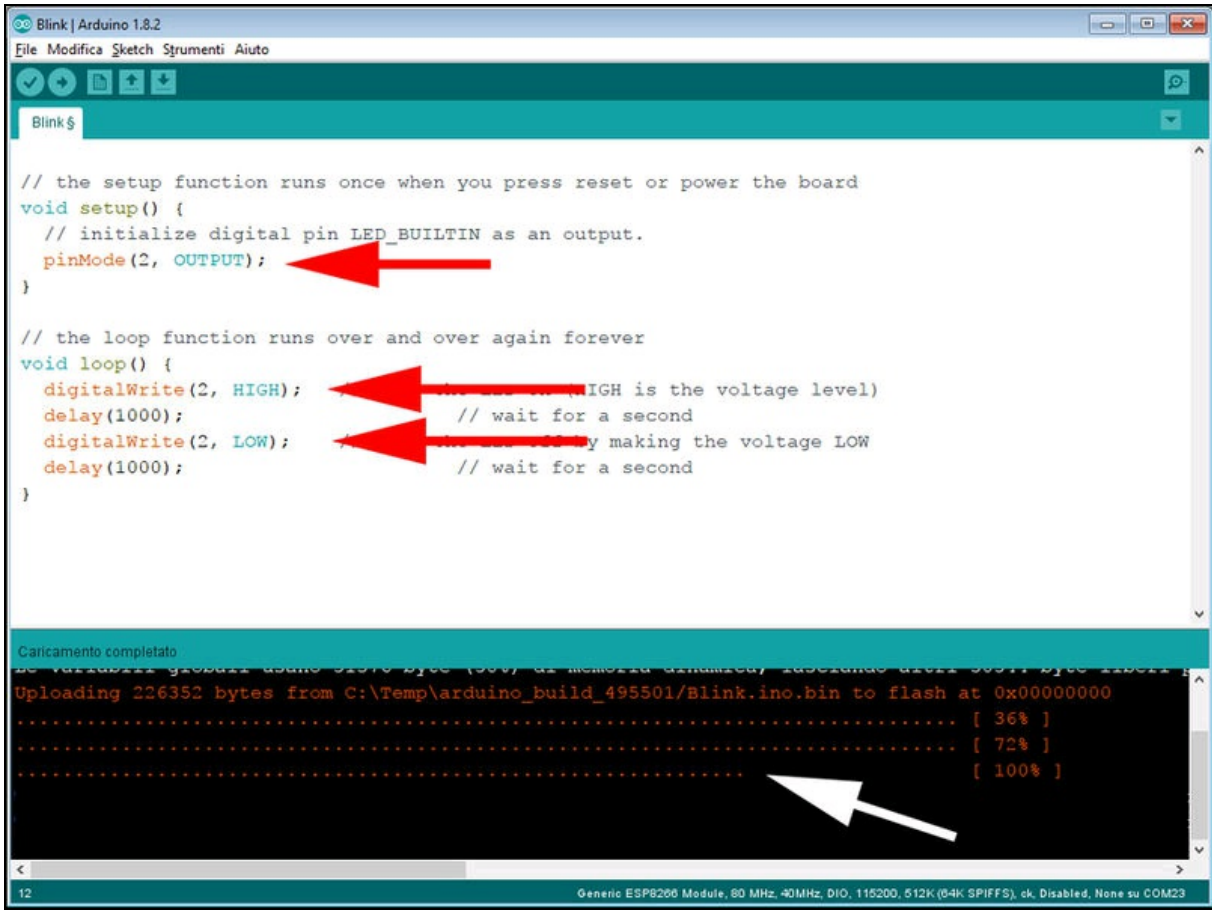


Figura 3.18 Esempio Blink modificato e caricamento in memoria del modulo ESP8266.

WiFiWebServer con ESP8266

Mantenendo il LED collegato al pin GPIO 2, questo semplice esempio, illustra come impostare un semplice server HTTP per accendere e spegnere il LED dal Web.

Il server imposta un pin GPIO in base alla richiesta del client:

- `http://server_ip/gpio/0` imposterà il GPIO_2 basso;
- `http://server_ip/gpio/1` imposterà il GPIO_2 alto.

`server_ip` è l'indirizzo IP del modulo ESP8266, che verrà stampato sul monitor seriale quando il modulo è collegato alla rete e al quale ci si dovrà connettere.

Una volta aperto lo sketch nell'IDE, le uniche righe da cambiare sono:

```
const char* ssid = "your-ssid";
const char* password = "your-password";
```

Al posto di *your-ssid* basta mettere l'SSID della rete Wi-Fi e al posto di *your-password*, la password di accesso alla rete.

Una volta caricato lo sketch nel modulo ESP8266, seguendo la procedura spiegata prima e ricordandosi di collegare prima il pin GPIO_0 a massa, si potrà effettuare una connessione Wi-Fi alla rete di casa.

Dopo il boot del modulo, basta aprire il monitor seriale per visualizzare la connessione avvenuta, l'avvio del server e l'indirizzo IP assegnato dal router DHCP, come illustrato dalla freccia della Figura 3.19a (si noti anche la freccia che indica la velocità del monitor seriale impostata a 115200 baud). Nell'esempio, l'indirizzo al quale connettersi è 192.168.1.2.

Aprendo un browser qualsiasi si potrà digitare l'url seguente:

- `192.168.1.2/gpio/1` per accendere il LED collegato al pin GPIO_2.
- `192.168.1.2/gpio/0` per spegnere il LED collegato al pin GPIO_2.

Come illustrato nella Figura 3.19b, il server risponderà inviando la scritta *GPIO is now HIGH*, quando il LED è acceso e la scritta *GPIO is now LOW*, quando il LED è spento (Figura 3.19c).

La parte dello sketch che fa tutto questo è la seguente:

```
void loop() {
    int val;
    if (req.indexOf("/gpio/0") != -1)
    {
        val = 0;
        digitalWrite(2,LOW);
    }
    else if (req.indexOf("/gpio/1") != -1)
    {
        val = 1;
        digitalWrite(2,HIGH);
    }
    client.flush();
    String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nGPIO is now ";
    s += (val)?"high":"low";
    s += "</html>\n";
    client.print(s);
    delay(1);
}
```

Come si può vedere, se il server riceve la richiesta dal browser, ovvero l'URL contenente la stringa `/gpio/1`, accende il LED con l'istruzione `digitalWrite(2, HIGH)`.

Se il server riceve l'URL contenente la stringa `/gpio/0`, spegne il LED con l'istruzione `digitalWrite(2, LOW)`. Contemporaneamente la variabile `val` è uguale a `1` o a `0`, per cui la stringa che viene visualizzata nel browser cambia di conseguenza con l'istruzione `s += (val)?"high":"low"`.

Si fa notare che la stringa "`s`", contiene l'intestazione http e il codice HTML per il client. Volendo cambiare qualcosa, si può scrivere tutto il codice HTML che si desidera, per esempio usando dei tag HTML per creare titoli, caratteri in grassetto, in corsivo e così via, per esempio:

```
<h1>Web Server</h1><b>GPIO is now </b>.
```

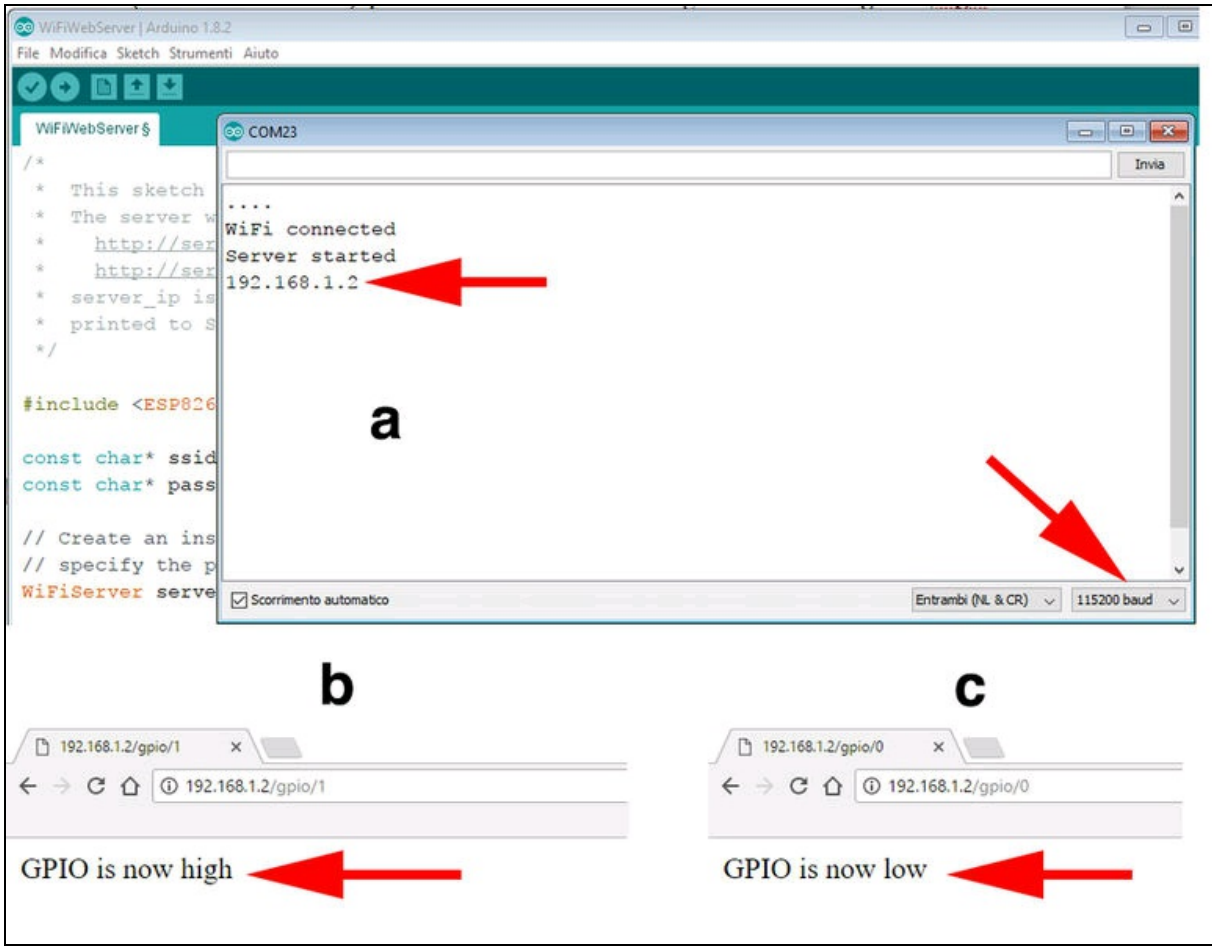


Figura 3.19 Monitor seriale con l'avvenuta connessione alla rete e l'IP ricevuto (a). La risposta del server quando il LED è acceso (b). La risposta del server quando il LED è spento (c).

Programmare i chip ATtiny

Innanzitutto, bisogna aggiungere i chip ATtiny al gestore schede. Per fare questo, occorre usare il seguente URL aggiuntivo, da scrivere nelle impostazioni dell'IDE di Arduino, come già spiegato in precedenza per le schede ESP8266. Attenzione! Se è già stato aggiunto l'URL per i moduli ESP8266, bisogna separare i due URL con una virgola.

L'URL per l'aggiunta di ATtiny al gestore schede è:

```
https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-
```

`manager/package_damellis_attiny_index.json`.

Procedendo come spiegato nel paragrafo precedente, dal menu *Strumenti* > *Scheda* si potranno selezionare i seguenti chip ATtiny:

- *ATtiny25/45/85*.
- *ATtiny24/44/84*.

I chip ATtiny24/44/84 sono microprocessori molto simili ai chip ATtiny25/45/85. Le uniche differenze sono il package DIP a 14 pin e 12 I/O anziché 6.

Arduino as ISP

Ecco cosa bisogna fare per poter procedere al trasferimento di uno sketch di Arduino alla memoria di un ATtiny25/45/85.

Con l'aiuto di una breadboard per i collegamenti, basta usare una scheda Arduino come programmatore ISP (*In-System Programming*), ovvero usare l'interfaccia SPI di Arduino per programmare un chip ATtiny. L'operazione è davvero semplice.

Collegare Arduino UNO e ATtiny25/45/85

Facendo attenzione a posizionare il chip a cavallo delle due sezioni della breadboard e posizionando la tacca di riferimento a sinistra, i collegamenti fisici da Arduino a un chip ATtiny sono i seguenti (Figura 3.20a):

ATtiny	Arduino	Nome del pin SPI
7	13	SCK
6	12	MISO
5	11	MOSI
1	10	RESET
8	5 V	
4	GND	

Collegare i pin di alimentazione 5 V e GND di Arduino direttamente ai pin 8 (VCC) e 4 (GND) di ATtiny25/45/85.

Caricare uno sketch nella memoria di un chip ATtiny

Ecco l’elenco di tutte le operazioni da effettuare.

1. Dal menu *Strumenti* impostare *Programmatore > Arduino as ISP* e *Scheda: Arduino UNO*. Ricordarsi di selezionare anche la porta seriale.
2. Dal menu *File > Esempi* aprire lo sketch *ArduinoISP* e caricarlo nella scheda Arduino. Questo renderà possibile usare Arduino come programmatore ISP.
3. Dal menu *Strumenti > Board* selezionare il chip *ATtiny* e i relativi parametri. Ecco un esempio.
   - *Scheda*: *ATtiny25/45/85*.
   - *Processore*: *ATtiny45*.
   - *Internal*: *8 MHz*.
4. Dal menu *Strumenti* selezionare la voce *Scrivi il bootloader*. Questo servirà a rendere il chip ATtiny compatibile con il firmware di Arduino.

Se tutto è stato collegato correttamente, dopo pochi secondi apparirà la scritta “Scrittura bootloader completata”. In caso contrario, bisognerà ricontrollare tutti i collegamenti.

A questo punto, è possibile caricare uno sketch di esempio per provare che tutto funzioni. Si consiglia di caricare lo sketch *Blink* dal menu *Esempi*.

Nel listato dello sketch cambiare `LED_BUILTIN` con `0` (vedi il listato seguente).

```
void setup() {
    pinMode(0, OUTPUT);
}
void loop() {
    digitalWrite(0, HIGH);
```

```
    delay(1000);
    digitalWrite(0, LOW);
    delay(1000);
}
```

Per caricare lo sketch nella memoria del chip ATtiny bisogna usare la funzione *Sketch > Carica attraverso un programmatore*. Lo sketch verrà compilato e caricato usando la scheda Arduino come programmatore ISP.

Dopo il caricamento dello sketch *Blink* nella memoria di ATtiny, si può collegare il catodo di un LED a GND e l'anodo (con un resistore da 220 Ω in serie) al pin 0 del chip ATtiny, come illustrato nella Figura 3.20b. Il LED lampeggerà esattamente come farebbe con una scheda Arduino. Attenzione! Se si vuole caricare un altro sketch, bisogna togliere il LED, perché sta impegnando la porta MOSI e non sarebbe possibile usarla per il caricamento via SPI.

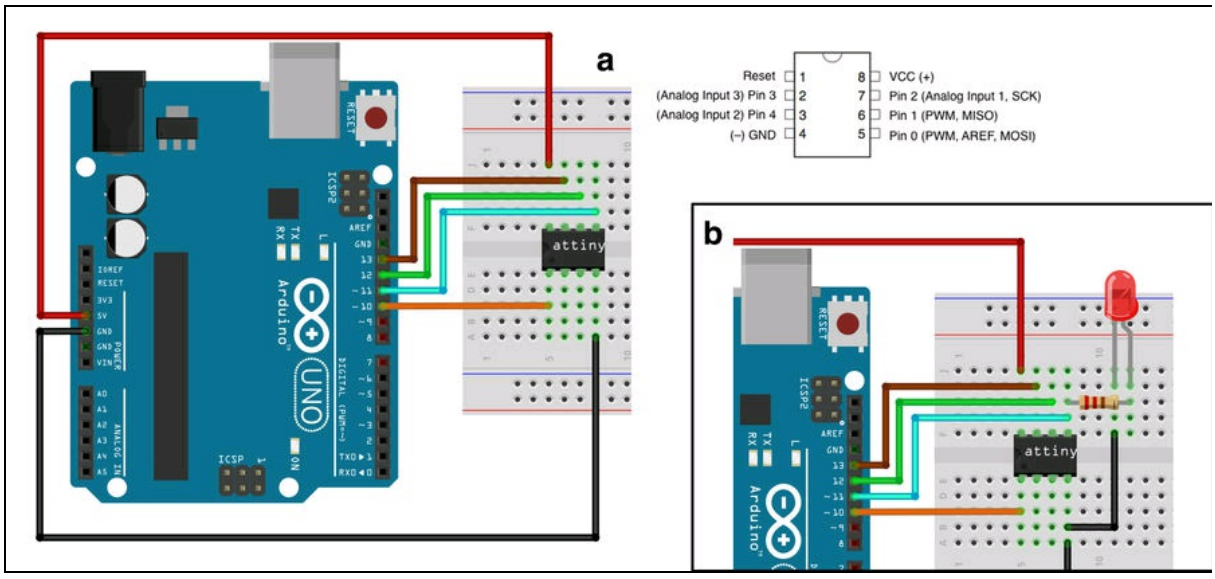


Figura 3.20 Collegamento ISP di ATtiny alla scheda Arduino (a). Collegamento del LED al pin 0 di ATtiny (b).

Libreria SoftwareSerial

È importante sottolineare la compatibilità di ATtiny con la libreria SoftwareSerial, inclusa nativamente nell'IDE di Arduino. Questa

utilissima libreria consente di gestire la comunicazione seriale UART creando una porta RX e una porta TX usando due porte digitali qualsiasi. Nel nostro caso abbiamo scelto le porte 2 e 3.

Per poter funzionare, la libreria ha bisogno che il clock sia impostato a 8 MHz, per cui è necessario scrivere il bootloader con il parametro `Clock` impostato a 8 MHz.

Caricando lo sketch di esempio *SoftwareSerialExample*, bisognerà impostare i due pin digitali di ATtiny come RX e TX.

Per esempio, l'esempio può essere modificato in questo modo:

```
#include <SoftwareSerial.h>
    SoftwareSerial mySerial(2, 3); // RX, TX.
```

Per non avere problemi di memoria con la libreria SoftwareSerial si consiglia di utilizzare un ATtiny45 o 85 perché la libreria è "famelica".

Per sfruttare al massimo la comunicazione seriale, resa possibile alla libreria SoftwareSerial, si può collegare un sensore a un ingresso analogico di ATtiny e scrivere i dati del sensore sul pin di trasmissione seriale (TX).

Sarà quindi possibile collegare a questo pin TX, il pin RX di un modem Bluetooth, Wi-Fi, Xbee, RF24, LoRa o qualsiasi altro dispositivo dotato di porta seriale. In questo modo, si potranno mandare i dati del sensore a una qualsiasi applicazione remota, per esempio una pagina web.

La Figura 3.21 illustra un esempio di collegamento di un modulo ESP8266 e di un sensore di temperatura a un chip ATtiny.

Avendo usato una pila LiPo da 3,7V non è necessaria nessuna regolazione. Usando invece una pila a 9V come sorgente di alimentazione, si dovrà usare un regolatore di tensione a 3,3 V (LM1086 o similare) per fornire l'alimentazione corretta sia al modulo ESP8266 che al chip ATtiny.

Il sensore di temperatura usato nel circuito è il diffusissimo LM35 (si veda il Capitolo 1).

Ecco un semplice listato di esempio per la trasmissione seriale dei dati di temperatura al modulo Wi-Fi.

Si fa notare che la porta analogica del sensore è *Analog Input 1*, corrispondente al piedino fisico 7 di ATtiny.

Ecco il listato commentato:

```
#include <SoftwareSerial.h>
const byte analogInPin = A1; // pin di lettura del sensore
SoftwareSerial mySerial(2, 3); // pin RX, TX
void setup() {
    mySerial.begin(115200); //inizializzazione della porta seriale a 115200 baud
    pinMode(analogInPin, INPUT); //impostazione del pin come ingresso
}
void loop() {
    int val = analogRead(analogInPin); //lettura della porta analogica
    float temp = (val * 3.7)/10; //variabile per la temperatura
    mySerial.print(char(temp));
        //manda alla seriale il valore di temperatura convertito in char
    delay(500); //ritardo di 500 millisecondi
}
```

Attenzione! La velocità di trasmissione dei dati è stata impostata a 115200 nel `setup()` con l'istruzione `mySerial.begin(115200)`. Questa è la velocità standard della porta UART del modulo ESP8266.

Il codice da sviluppare è molto semplice, basta considerare che:

- il sensore LM35 fornisce 10 millivolt per ogni grado Celsius;
- il valore della lettura analogica e la relativa conversione digitale è compresa fra 0 e 1024;
- moltiplicando il valore di lettura per il valore minimo di tensione (1024/tensione di alimentazione, nel nostro caso 3,7 volt) si ottiene il valore di tensione effettiva all'ingresso;
- dividendo il risultato per 10, si otterranno i gradi centigradi (arrotondati al valore intero).

Basterà quindi scrivere questo valore sulla porta TX con l'istruzione `mySerial.print(char(temp))`. Questo valore verrà letto dalla porta RX del modulo ESP8266, come spiegato nel paragrafo seguente.

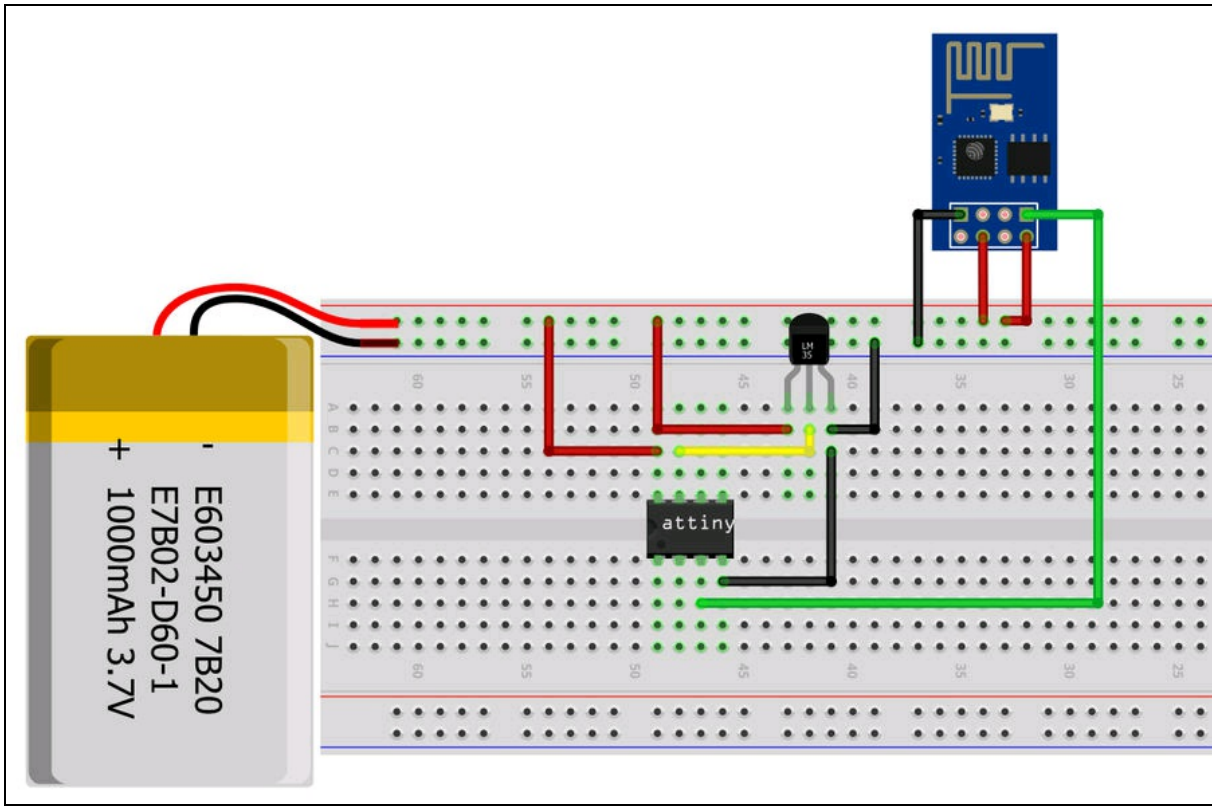


Figura 3.21 Layout su breadboard del collegamento di un sensore di temperatura e di un modulo ESP8266 a un ATtiny45/85.

Visualizzazione della temperatura sul Web

A questo punto, per leggere dalla porta seriale del modulo, non resta che modificare l'esempio *WiFiWebServer* visto prima, aggiungendo qualche riga di codice.

Nel loop basta aggiungere il seguente codice:

```
void loop() {
    WiFiClient client = server.available();
    int inByte; // variabile per il byte da leggere
    if (Serial.available()>0) // se la seriale è disponibile
    {
        inByte = Serial.read(); // viene letto un byte dalla seriale
        if (inByte) // se il byte esiste
    {
    s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nTemperature is now ";
    // stringa per l'header HHTP e codice HTML
    s += (inByte); // si aggiunge il valore letto dalla seriale
    s += " °C</html>\n"; // si aggiunge il tag di fine documento HTML
```

```
    }
}
```

Dopo aver caricato nella memoria del modulo ESP8266 il codice così modificato e aver collegato la porta RX del modulo ESP8266 al pin TX di ATtiny, si potrà digitare l'indirizzo IP come fatto nel precedente esempio e si vedrà apparire sul browser la scritta: "Temperature is now 26 ºC", come illustrato nella Figura 3.22. Toccando con un dito il sensore di temperatura ed eseguendo il refresh della pagina del browser si potrà vedere la temperatura che aumenta.

Figura 3.22 Pagina web con la visualizzazione della temperatura.

Python

Python è un linguaggio di programmazione ad alto livello, disponibile per tutte le piattaforme Windows, Mac e Linux, e certificato open source da OSI (*Open Source Initiative*). Fu rilasciato pubblicamente per la prima volta nel 1991 dal suo creatore Guido van Rossum, grande appassionato della celebre commedia inglese *Monty Python's Flying Circus* del gruppo comico Monty Python.

Lo sviluppo di Python è in continuo fermento, grazie all'enorme comunità internazionale di sviluppatori, gestita dall'organizzazione no-profit Python Software Foundation.

Ritenuto un ambiente di programmazione semplice da usare, Python è stato abbracciato da Raspberry Pi come nucleo di programmazione principale, come ricorda la parola "Pi" nel nome.

Python è un linguaggio pseudocompilato: un interprete analizza il codice sorgente, formato da semplici file di testo che vengono salvati con estensione `.py`.

Come vedremo, esiste anche una fase di compilazione, come avviene per esempio nel linguaggio C, per cui è possibile generare un file eseguibile partendo dal file sorgente.

Essendo pseudo-interpretato, Python è un linguaggio portabile su qualsiasi piattaforma e può essere interpretato su gran parte delle piattaforme Apple (Mac), Microsoft (Windows) e GNU/Linux. È sufficiente che nel computer in uso sia installata la versione corretta dell'interprete.

Python è un software libero e completamente gratuito e pertanto può essere liberamente modificato e ridistribuito, secondo le regole di una licenza open source. Questo ha fatto sì che molte applicazioni anche di un certo rilievo siano programmate in Python (YouTube, per esempio).

Si consiglia di visitare il sito in italiano (http://www.python.it), nel quale sono disponibili guide e tutorial nella nostra lingua, download, aggiornamenti del software per tutte le piattaforme e moltissime altre risorse gratuite.

Coloro che non hanno confidenza con la programmazione in generale e con Python in particolare, prima di passare al prossimo paragrafo, sono fortemente invitati a dare uno sguardo al più semplice degli esempi, riportato qui di seguito, e alle procedure per rendere eseguibile il file sorgente.

Avviare Python

Dal menu *Programming* di Raspbian, si può scegliere indifferentemente la versione Python 2.7.9 o la versione Python 3.4.2 (alla data di maggio 2017), facendo clic su *Python 2 (IDLE)* o *Python 3 (IDLE)*.

La ragione per cui ci sono due IDLE è che ci sono due versioni di Python installate su Raspbian: Python 3 è la più recente, ma non è pienamente compatibile con la versione 2. Per cui bisogna prestare attenzione ad aprire il programma IDLE corretto in base alla versione del file.

Come illustrato nella Figura 3.23, la finestra *IDLE* è una semplice shell di testo in cui si possono digitare direttamente i comandi.

Un progetto Python non è altro che un file di testo contenente le istruzioni da compilare per il processore, tant'è che un file di Python, spesso chiamato script, può essere creato o modificato usando un qualsiasi editor di testo come, per esempio, nano o Leafpad.

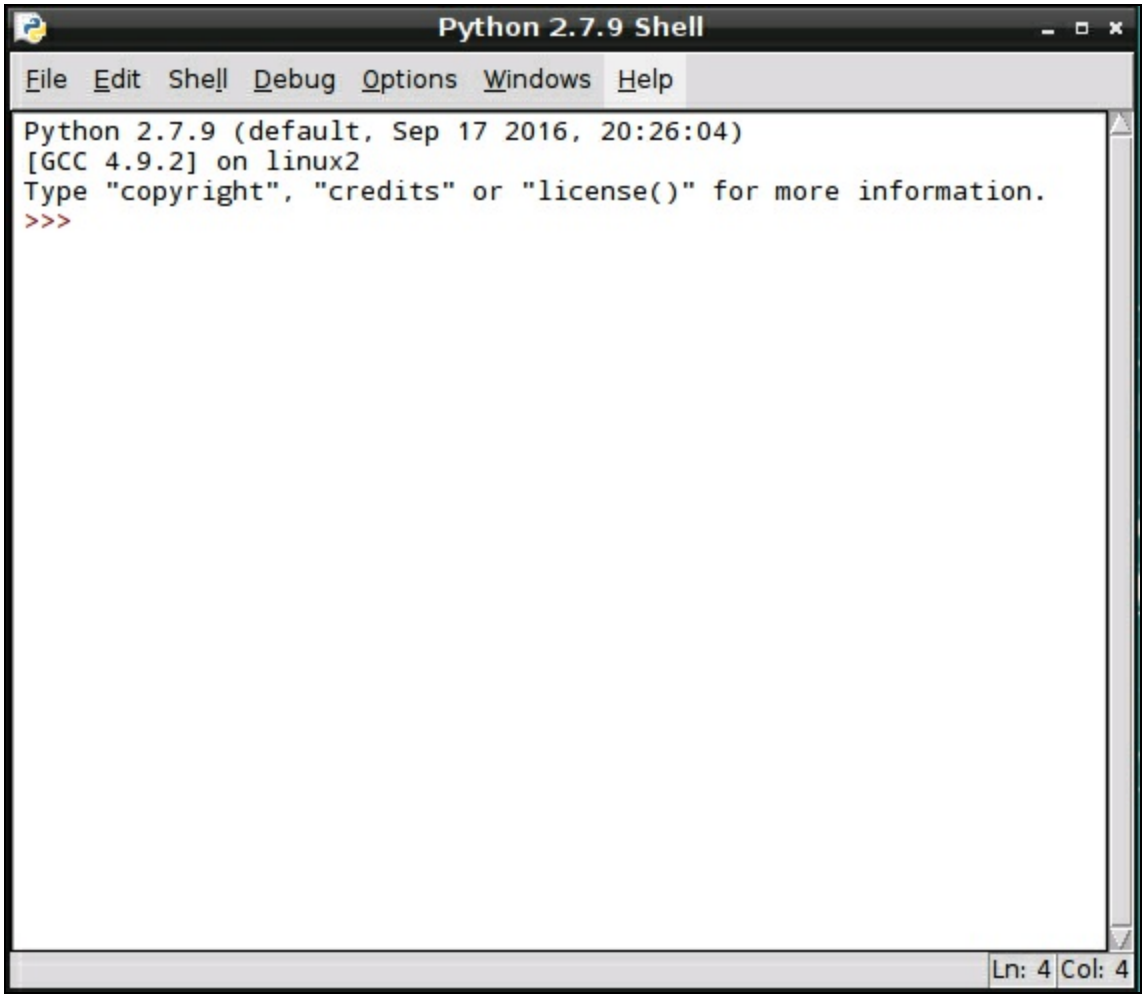


Figura 3.23 La finestra IDLE di Python 2.

È bene iniziare tutti gli script di Python con la prima riga conosciuta come *hash bang* (o *shabang*, *shebang*, *sha-bang*), che prende il nome dal simbolo `#` seguito dal carattere `!` all'inizio della riga. La combinazione di simboli `#!` corrisponde a un comando Unix che consente di eseguire le istruzioni dell'interprete corrispondente. Per esempio: `#!/usr/bin/python` esegue le istruzioni usando Python.

Questa riga indica al sistema operativo dove deve cercare i file di Python. Anche se questa istruzione non è necessaria per i programmi eseguiti all'interno della finestra *IDLE*, il comando chiamerà esplicitamente il terminale, necessario per i programmi che vengono

eseguiti direttamente dal nome del file del programma (con estensione `.py`), quando questo file viene salvato.

Per garantire l'esecuzione del programma, indipendentemente dalla piattaforma, la prima riga del programma dovrebbe essere la seguente:

```
#!/usr/bin/env python.
```

Questa riga indica al sistema operativo di cercare la variabile d'ambiente `$PATH`, ovvero la posizione in cui viene eseguito Python, che è anche dove Linux memorizza la posizione dei file che possono essere eseguiti come programmi. In questo modo si dovrebbe rendere compatibile il file su qualsiasi distribuzione Linux utilizzata su Raspberry Pi.

La variabile `$PATH` contiene una lista di directory in cui sono archiviati i file eseguibili, e viene utilizzata per trovare i programmi quando si digita il loro nome nella finestra del terminale.

Per stampare a video un messaggio, si utilizza il comando `print` di Python, come in quasi tutti gli ambienti di programmazione. Questo comando invia il testo a un dispositivo di output definito per default come la finestra della console o del terminale in cui il programma è in esecuzione.

Esempio Hello World

Per stampare una scritta sul terminale, serve solo un'istruzione. La sintassi per Python 3 è la seguente:

```
print ('Hello world')
```

oppure

```
print ("Hello world")
```

La stringa con apici semplici o doppi va sempre fra parentesi, altrimenti apparirà un errore di sintassi.

La sintassi per Python 2 è la seguente:

```
print "Hello world"
```

oppure

```
print 'Hello world'
```

La stringa con apici semplici o doppi può non essere messa fra parentesi.

La Figura 3.24a mostra il semplice programma di stampa a video di “Hello world” eseguito dalla riga di comando nella shell di Python 3. Notare la segnalazione dell’errore di sintassi se non si mettono le parentesi.

La Figura 3.24b mostra lo stesso programma eseguito dalla riga di comando nella shell di Python 2.

A questo punto è possibile scrivere il programma completo aprendo una nuova finestra dal menu *File > New Window* (oppure Ctrl+N).

Eseguire lo script

Una volta salvato il file dal menu *File > Save* (oppure Ctrl+S) con il nome `hello_world.py` (o qualsiasi altro nome) si può eseguire il programma con il comando *Run* dal menu omonimo, oppure premendo il tasto F5. Se non si salva il file, apparirà un avviso e la finestra di dialogo per il salvataggio (Figura 3.24c).

Una volta salvato il file nel percorso locale `/home/pi`, si vedrà la finestra di output (Figura 3.24d).

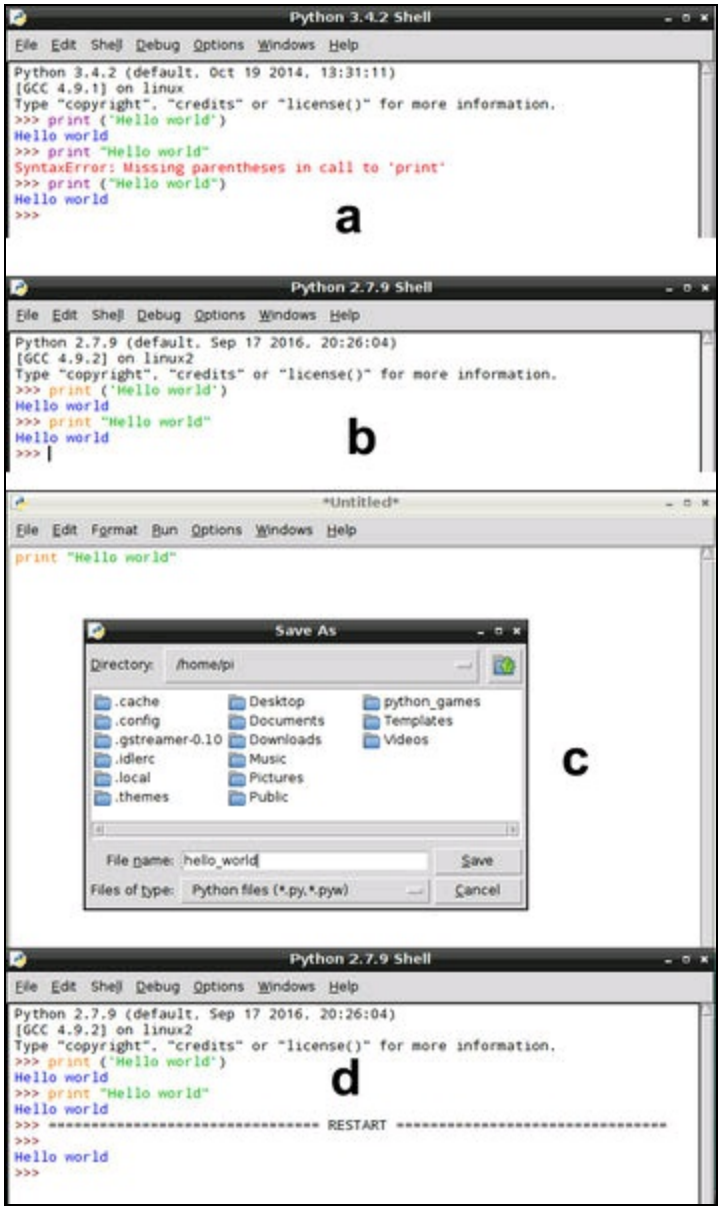


Figura 3.24 La sintassi del comando print in Python 3 (a). La sintassi del comando print in Python 2 (b). La finestra di salvataggio del file (c). L'output del programma dopo il comando Run e il salvataggio del file (d).

Una volta chiuso l'editor di Python, si può eseguire lo script dal terminale digitando i seguenti comandi:

```
cd /home/pi
python hello_world.py
```

L'output del programma sul terminale sarà simile a quello rappresentato nella Figura 3.25.

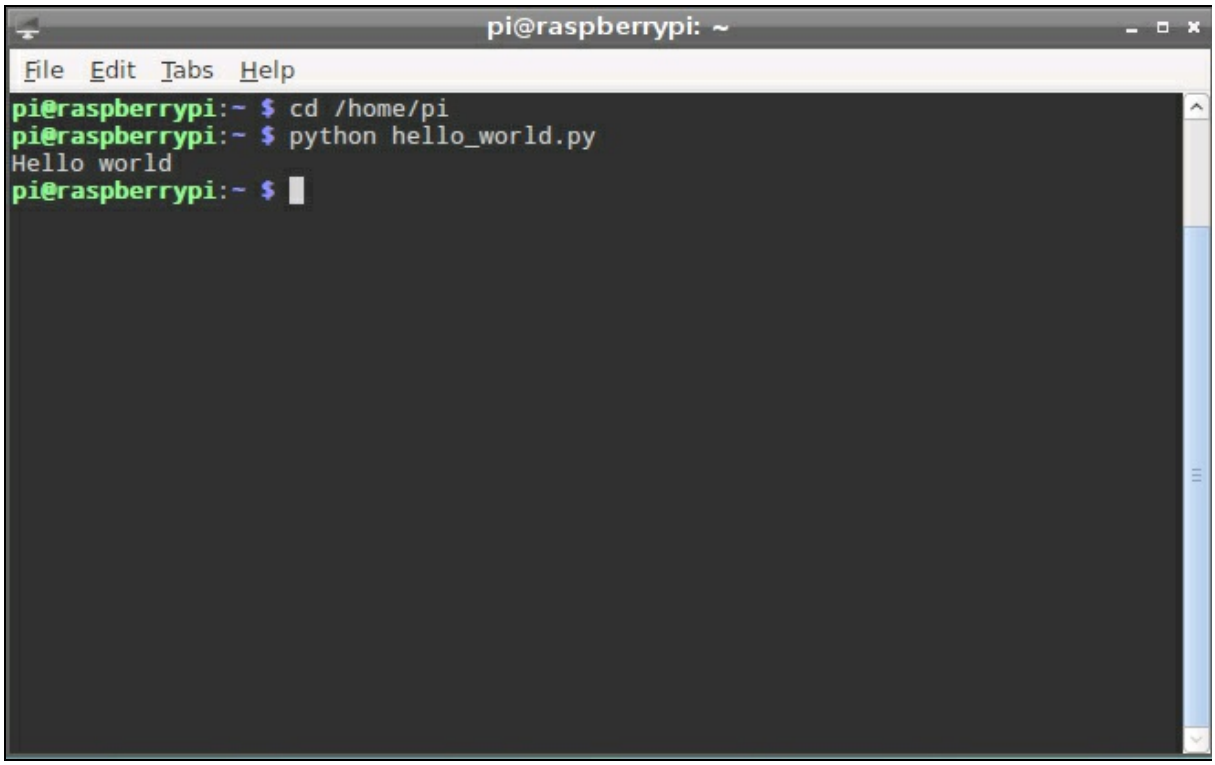


Figura 3.25 Cambio della directory ed esecuzione di hello_world.py dal terminale con l'output della scritta Hello world.

Rendere eseguibili gli script di Python

Abbiamo appena visto che per eseguire uno script di Python si possono usare la shell di Python o una shell del terminale. In quest'ultimo caso, si dà il comando a Python per eseguire in background il file con l'estensione `.py`.

Con l'istruzione *hash bang* posta all'inizio del file, è possibile eseguire il file anche senza dover chiamare prima Python. Con questo metodo si potranno lanciare i propri script dal terminale digitando solo il nome del file.

Per fare questo, bisogna dire al kernel del sistema operativo che il file creato con Python deve essere contraddistinto come eseguibile, pertanto occorre assegnare un attributo che indichi che il file è un programma eseguibile. Normalmente questo attributo non viene impostato

automaticamente, per proteggere il sistema da malware o virus scaricati da Internet.

Per rendere eseguibile il file `hello_world.py` creato in precedenza, basta usare il comando `chmod`. Il comando è l'abbreviazione di *change mode* e serve per cambiare la modalità del file. È il tipico comando usato nei sistemi operativi Unix e Unix-like per modificare i permessi di file e directory. Ecco il comando da digitare in una shell del terminale:

```
sudo chmod +x hello_world.py
```

dove `+x` è l'attributo che permette al programma di essere eseguito.

A questo punto, per eseguire il programma, basta digitare dal terminale il seguente comando:

```
./hello_world.py
```

Il programma `hello_world.py` viene eseguito senza digitare il comando completo `python hello_world.py` che, in pratica, informa Python di aprire il file.

Il programma può però essere eseguito solo se è stato salvato nel percorso locale dell'utente, ovvero `/home/pi/hello_world.py`.

Per rendere il file accessibile come per qualsiasi altro comando dal terminale, il file deve essere copiato nella cartella `/usr/local/bin`.

Da una shell di LXTerminal digitare il seguente comando:

```
sudo cp hello_world.py /usr/local/bin/
```

dove `cp` sta per *copy*.

È necessario copiare il file come *superuser* (`sudo`) perché gli utenti senza privilegi di *root* non possono scrivere nella directory `/usr/local/bin`.

Una volta copiato il file `hello_world.py` nella cartella `/usr/local/bin`, ovvero nel percorso della variabile d'ambiente `$PATH`, il file può essere eseguito da qualsiasi directory semplicemente digitandone il nome.

Per verificare il funzionamento, basta cambiare directory entrando in una directory diversa con il comando `cd` (ovvero *change directory*) ed

eseguire il programma digitando solamente il nome del file:

```
hello_world.py
```

Ora manca solo un tocco professionale. Se si elimina l'estensione `.py` dal file, i programmi in Python sembreranno simili ai programmi di sistema o di utilità.

È possibile rinominare il file per rimuovere l'estensione come segue. Dal terminale digitare la seguente riga di comando:

```
sudo mv /usr/local/bin/hello_world.py /usr/local/bin/helloworld
```

dove `mv` sta per *move*, cioè sposta.

Ovviamente, si può dare un qualsiasi altro nome al file e, una volta rinominato il file, il programma potrà essere eseguito da ogni altra posizione semplicemente digitando da terminale `helloworld` senza estensione (o qualsiasi altro nome).

Editor del codice

Quando si apre Python 2 o Python 3 dal menu, viene visualizzata la finestra della shell che, in pratica, non è altro che la versione grafica della shell del terminale. Per aprire l'editor del codice bisogna aprire una nuova finestra dal menu *File* > *New Window* o usare i tasti Ctrl+N.

Nella finestra dell'editor si possono scrivere le istruzioni e tutto il codice che serve per la programmazione, rispettando due regole fondamentali per la scrittura del codice:

- lettere maiuscole e minuscole;
- indentazione del codice.

Lettere maiuscole e minuscole

Tutte le istruzioni devono rispettare la sintassi in termini di lettere maiuscole o minuscole. Per esempio:

- `print` (non `Print`);
- `while` (non `While`);
- `if` (non `If`);
- `elif` (non `Elif`);
- `True` (non `true`);
- `False` (non `false`).

C’è di buono che quando una parola riservata viene scritta correttamente, si colora di arancio (swe è una parola chiave) o in violetto (se è una direttiva).

Tutte le variabili dichiarate in maiuscolo e minuscolo devono essere sempre riportate allo stesso modo nel listato. Per esempio, se si dichiara una variabile `pulsante` non può essere richiamata come `Pulsante`, `PULSANTE` o qualsiasi altra combinazione di lettere maiuscole/minuscole, perché non verrebbe riconosciuta.

Indentazione del codice

L’indentazione del codice è il sistema con cui Python raggruppa e mette in ordine le istruzioni. Altri ambienti di sviluppo software, come il C, sono dotati di interpreti intelligenti che riescono a intercettare le righe di codice a prescindere dalla loro posizione nel listato. Per esempio, nel linguaggio C (e derivati) è sufficiente apporre un punto e virgola (`;`) alla fine dell’istruzione per far capire al compilatore che l’istruzione finisce. Non importa quanti spazi o tabulazioni si inseriscono fra una parola chiave, una variabile o un valore.

Con Python, invece, si devono usare le tabulazioni per ordinare le righe all’interno di un’istruzione. Ogni istruzione o funzione deve terminare con due punti (`:`) e l’argomento deve andare a capo con indentazione automatica oppure rimanere sulla stessa riga.

Questa particolarità di Python può sembrare una limitazione. In realtà, si può considerare positivo il fatto che costringa il programmatore a mettere tutto in un ordine logico.

Esempio (Figura 3.26):

```
while True:
    x = float (input ('Digita un numero: ') )
    if x < 0:
        print ('Il numero è minore di zero')
    elif x == 0:
        print ('Il numero è uguale a zero')
    elif x > 0:
        print ('Il numero è maggiore di zero')
```

Dopo i due punti (:) si può continuare a scrivere il codice invece di andare a capo. Per esempio, il listato precedente può essere scritto anche così:

```
while True:
    x = float ( input ('Digita un numero: ') )
    if x < 0: print ('Il numero è minore di zero')
    elif x == 0: print ('Il numero è uguale a zero')
    elif x > 0: print ('Il numero è maggiore di zero')
```

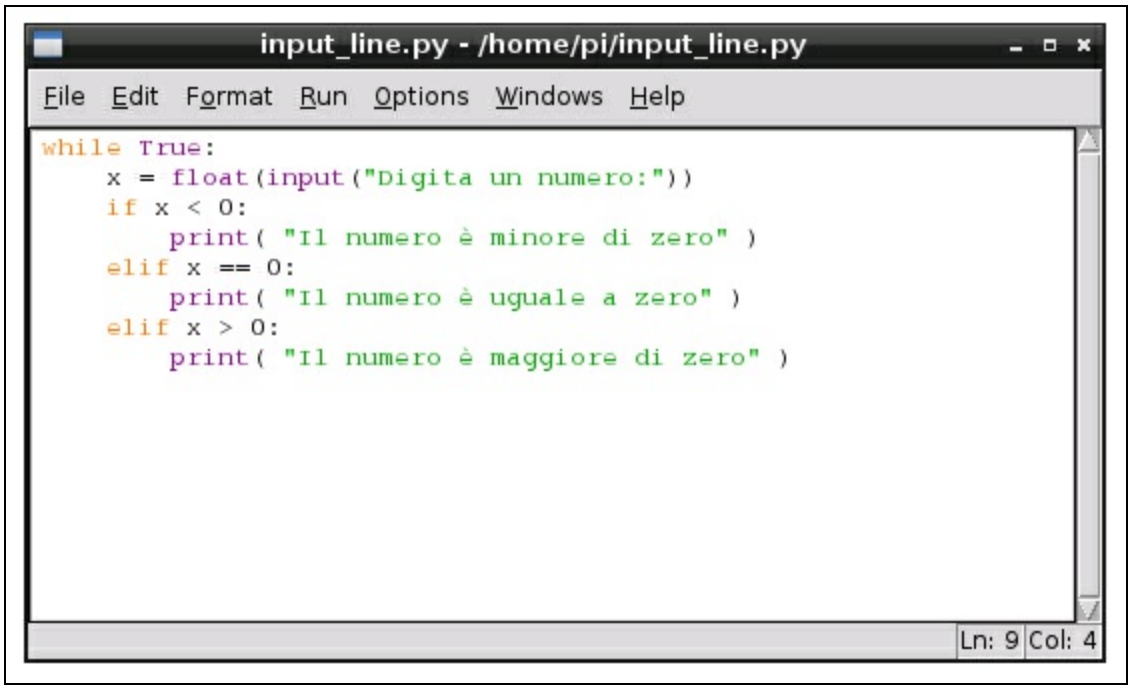


Figura 3.26 La finestra dell'editor di Python con un semplice listato (a).

Mandando a capo a fine riga, si noterà che l'editor indenta il codice in modo automatico, ovvero immette automaticamente delle tabulazioni sulla riga successiva. Nella shell, invece, vengono usati i tre punti (...).

Si fa notare che l'indentazione dopo l'iterazione `while True:` è obbligatoria.

Per scrivere sulla stessa riga più istruzioni, queste devono essere separate da un punto e virgola (;).

Per esempio:

```
elif x == 0:
    print ('Il numero è zero');print ('riprova')
```

NOTA

La condizione `elif` è simile a `else if` di altri linguaggi.

Per avviare l'esecuzione dello script basta premere il tasto F5. Se non è stato salvato, prima di avviare lo script, viene chiesto di salvare il listato con un nome.

Python e l'hardware di Raspberry Pi

La prima cosa che viene in mente quando si parla di controllo dell'hardware di Raspberry Pi (leggi porta GPIO) è quella di accendere e spegnere un LED. Può sembrare banale ma, anche in questo caso, si tratta di una sorta di "Hello World" dell'elettronica.

Per schede come Arduino, come abbiamo visto, si carica l'esempio *Blink* per vedere se tutto funziona. In Raspberry Pi, non essendoci un LED nella scheda che possa avere la stessa funzione di una scheda Arduino, è necessario costruire un piccolo circuito esterno. La Figura 3.27 illustra il semplice collegamento. I componenti necessari sono i seguenti:

- un diodo LED;
- un resistore da 220 Ω;

- cavetti maschio-femmina per breadboard;
- una breadboard.

Collegare il catodo del LED a una massa e l’anodo con il resistore da 220 Ω in serie al pin GPIO7 di Raspberry Pi. Diamo un’occhiata al codice.

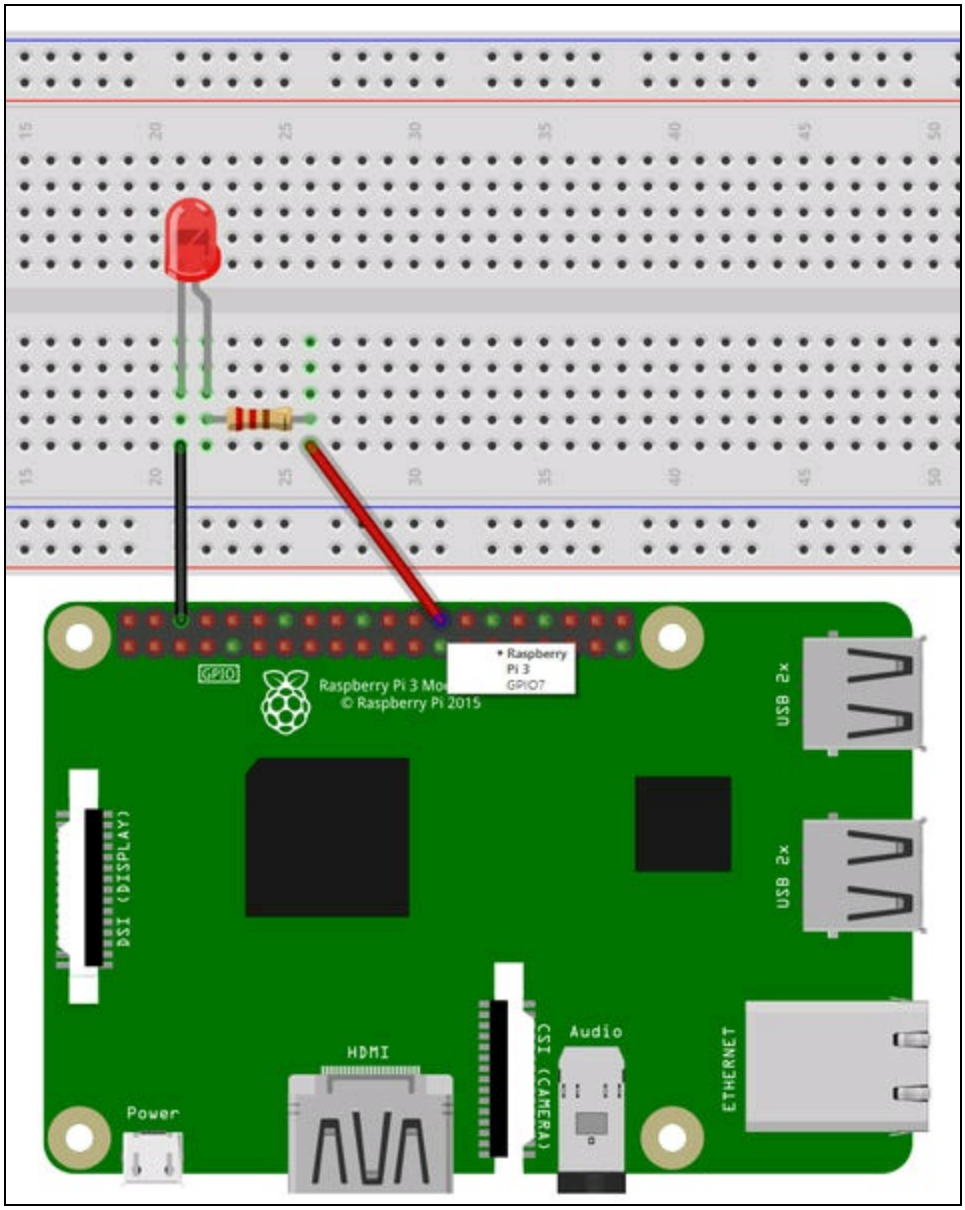


Figura 3.27 Il circuito per collegare un LED a Raspberry Pi.

Python Blink

File di esempio nelle risorse del libro: `Python_Blink.py`.

Per creare un programma simile al già citato *Blink* per Arduino, ovvero uno script in grado di accendere e spegnere un LED a intervalli di tempo, si può aprire una finestra dell'editor Python e scrivere il codice seguente, rispettando l'indentazione imposta da Python.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.OUT)
GPIO.setwarnings(False)
while True:
    GPIO.output(7, True)
    time.sleep(1)
    GPIO.output(7, False)
    time.sleep(1)
```

Osservando il listato, si noterà l'uso della libreria `RPi.GPIO`, appositamente creata per gestire la porta GPIO di Raspberry Pi.

- L'istruzione `import RPi.GPIO as GPIO` serve a importare la libreria GPIO nello spazio utente GPIO.
- Con l'istruzione `import time` si importa semplicemente una libreria specifica di Python per gestire la temporizzazione del LED.
- Con l'istruzione `GPIO.setmode(GPIO.BCM)` si imposta la modalità GPIO per il processore in uso, ovvero per il Broadcom BCM2835/2836/2837, a seconda del modello di Raspberry Pi.
- Con `GPIO.setup(7, GPIO.OUT)` si imposta la direzione della porta GPIO7, che in questo caso è `OUT` per poter mandare i dati in uscita. L'istruzione è simile a `pinMode(x, OUTPUT)` di Arduino.
- La funzione `GPIO.setwarnings(False)` è opzionale, ma consigliata per disabilitare gli avvertimenti di eventuali errori GPIO.
- L'iterazione `while True` crea un ciclo infinito, simile alla funzione `void loop()` di Arduino.

- Il comando `GPIO.output(7, True)` imposta il livello logico della porta a `1`. Il LED si accende, in modo simile a `digitalWrite(led, HIGH)` di Arduino.
- La funzione `time.sleep(1)` crea un ritardo di un secondo, in modo simile a `delay(1000)` di Arduino.
- Il comando `GPIO.output(7, 0)` imposta il livello logico della porta a `0`, in modo simile a `digitalWrite(led, LOW)` di Arduino.
- Di nuovo, la funzione `time.sleep(1)` crea un altro ritardo di un secondo, in modo simile a `delay(1000)` di Arduino.

> **NOTA**
> Al posto dei termini `True` e `False` si possono usare indifferentemente i valori `1` e `0`, ottenendo lo stesso risultato in uscita.

Una volta salvato il file, il programma potrà venire eseguito all'interno di Python oppure dal terminale con il comando `python my_blink.py`.

Il LED si accenderà e si spegnerà a intervalli di un secondo. Per aumentare o ridurre il tempo di accensione/spegnimento del LED, basta modificare `time.sleep(1)` con un valore a piacere.

I valori della libreria `time` sono espressi in secondi, per cui `time.sleep(0.1)` imposta la temporizzazione a 1/10 di secondo e `time.sleep(30)` imposta trenta secondi. Per terminare il programma premere Ctrl+C.

Python LED Button

File di esempio nelle risorse del libro: `Python_LED_Button.py`.

Una volta capito come controllare l'uscita della porta GPIO, si può modificare leggermente il circuito precedente e fare in modo di

accendere e spegnare il LED tramite un pulsante, andando a controllare lo stato dell'ingresso a cui viene collegato.

Il pulsante da usare è di tipo sempre aperto a due vie con quattro piedini.

Come si può vedere dalla Figura 3.28, il piccolo pulsante è dotato di quattro terminali collegati a coppie. Le coppie dei terminali 1-2 e 3-4 sono sempre collegate. Quando si preme il pulsante si cortocircuitano le due coppie di terminali 1-3 e 2-4. Quando si rilascia il pulsante il circuito torna aperto.

I quattro terminali sono utili per gestire meglio il collegamento nella breadboard con resistori *pull-up* o *pull-down*.

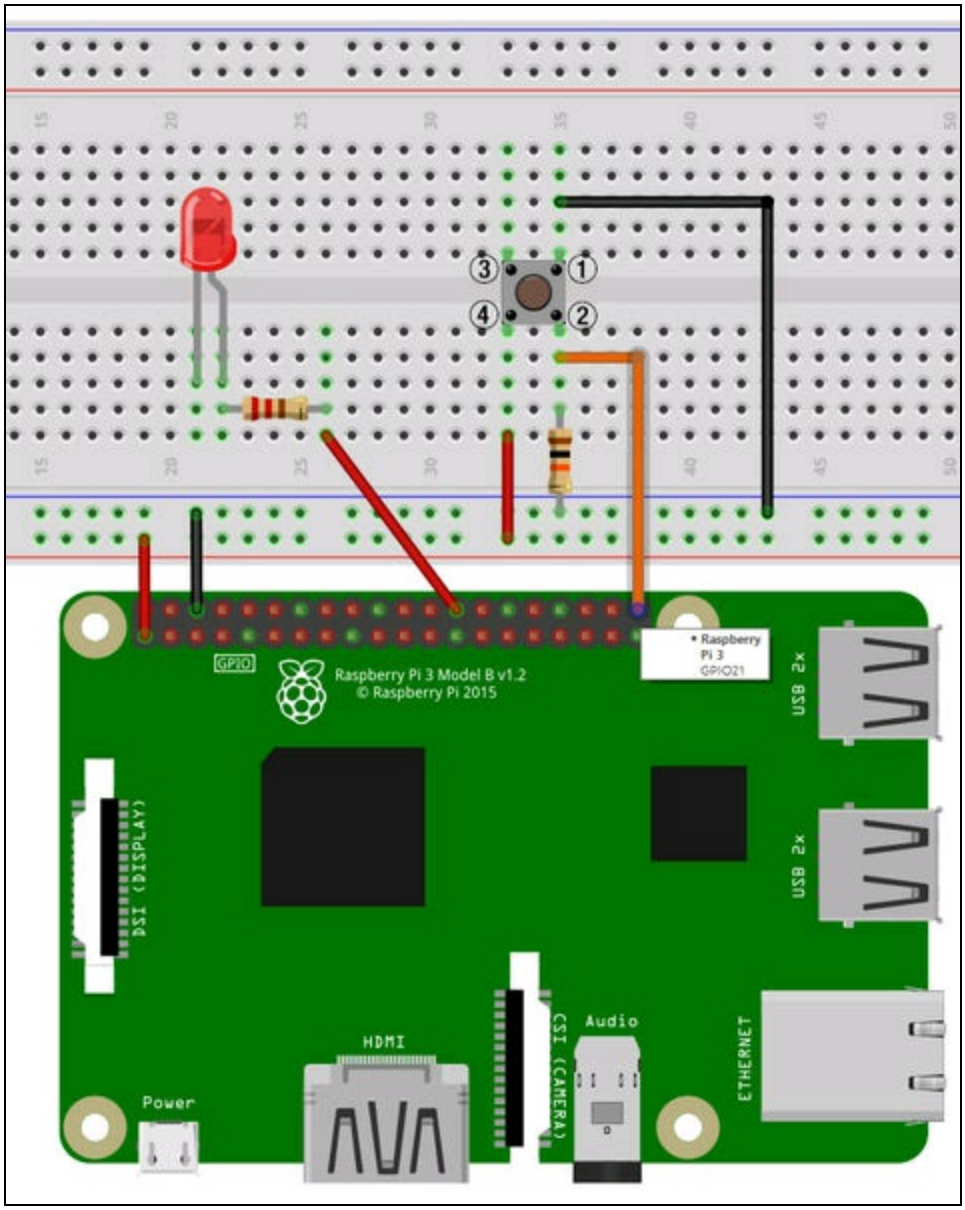


Figura 3.28 Circuito di un LED con un pulsante di accensione/spegnimento.

NOTA

I resistori pull-up e pull-down servono nei collegamenti di circuiti logici per fornire una tensione di riferimento quando si devono confrontare differenti livelli di tensione in ingresso. Di solito, è sufficiente un valore resistivo di 10 kΩ per fare in modo che il resistore “tiri” la tensione verso il polo positivo (*pull-up*) o verso il polo negativo (*pull-down*). Se l’ingresso logico su cui si opera la lettura della tensione in ingresso non ha un livello di riferimento, l’ingresso potrebbe rilevare che il pulsante è stato premuto ma anche il contrario, dando risultati imprevedibili.

Osservando la Figura 3.28, seguire la procedura presentata di seguito.

- Alimentare la breadboard dai pin GND (massa) e 3.3 V della porta GPIO (è importante non sbagliare l'alimentazione).
- Collegare il LED con il catodo a massa e l'anodo al resistore da 220 Ω.
- Collegare l'altro capo del resistore da 220 Ω al pin GPIO 7.
- Collegare il piedino 4 del pulsante al positivo della breadboard.
- Collegare un capo del resistore pull-down da 10 kΩ al piedino 2 del pulsante.
- Collegare l'altro capo del resistore da 10 kΩ a massa.
- Collegare il piedino 2 del pulsante al pin GPIO 21.
- Collegare il piedino 1 del pulsante a massa.

Non ci resta che scrivere il codice dello script in una nuova finestra dell'editor.

Python LED Button: codice

Abbiamo chiamato così il progetto per l'accensione e lo spegnimento del LED tramite un pulsante. Ecco il semplice listato:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(21, GPIO.IN)
GPIO.output(7, False)
while True:
    input_value=GPIO.input(21)
    if input_value == False:
        GPIO.output(7, False)
        print ('Pulsante non premuto ')
        time.sleep(0.5)
    if input_value == True:
        GPIO.output(7, True)
        print ('Pulsante premuto. LED acceso.')
```

Il codice è in parte simile al precedente. Le istruzioni che cambiano o sono state aggiunte sono le seguenti.

- Con l’istruzione `GPIO.setup(12, GPIO.IN)` si imposta la direzione dei dati in ingresso dal pin GPIO 21.
- Con l’istruzione `GPIO.output(7, False)` si imposta il livello logico 0 sul pin GPIO 7, per iniziare con il LED spento.
- L’iterazione `While True` crea il ciclo infinito, mentre la variabile `input_value=GPIO.input(12)` legge il valore in ingresso al pin GPIO 21.
- La condizione `if input_value == False` controlla il valore in ingresso della variabile `input_value`; se il valore è 0, allora...
- ... `GPIO.output(7, False)`, cioè il valore sul pin GPIO 7 al quale è collegato il LED è 0, il LED rimane spento.
- Contemporaneamente, la funzione `print (''Pulsante non premuto. LED spento ')` stampa lo stato del pulsante.
- La condizione `if input_value == True` controlla il valore in ingresso della variabile `input_value`; se il valore è `1`, allora...
- ... `GPIO.output(7, True)` cioè il valore sul pin GPIO7 al quale è collegato il LED è `1`, e il LED si accende.
- Allo stesso tempo, la funzione `print` stampa la scritta *Pulsante premuto. LED acceso* fintantoché il pulsante rimane premuto.

È stato inserito un valore `time` di 0,5 secondi, ma si può eliminare qualora non fosse più necessario stampare le scritte a video, ottenendo così una riposta rapida del pulsante. Per terminare il programma premere Ctrl+C.

Windows 10 IoT Core

Di tutti i prodotti disponibili nel settore dei maker, Microsoft ha concentrato la sua attenzione su due prodotti che sono diventati sinonimi di IoT: Raspberry Pi e Arduino. Ma la notizia che rende felici i maker di tutto mondo è che Microsoft regala (finalmente!) praticamente tutto: da Windows 10 a Visual Studio 2017, da Windows 10 IoT Core alle librerie Arduino, al portale dedicato all'IoT.

Per iniziare l'avventura con Windows 10 IoT, si parte dal seguente indirizzo: `https://dev.windows.com/it-it/iot`.

Facendo clic sul pulsante *Get started* si entra nel portale vero e proprio di Windows 10 IoT. È una pagina che vale la pena di memorizzare nei segnalibri, perché è da qui che si inizia, ed è qui che, in molte occasioni, si vorrà tornare (Figura 3.29).

NOTA

Per poter sfruttare le risorse gratuite di Windows 10 IoT è necessario aver installato il sistema operativo Windows 10 nel proprio PC.

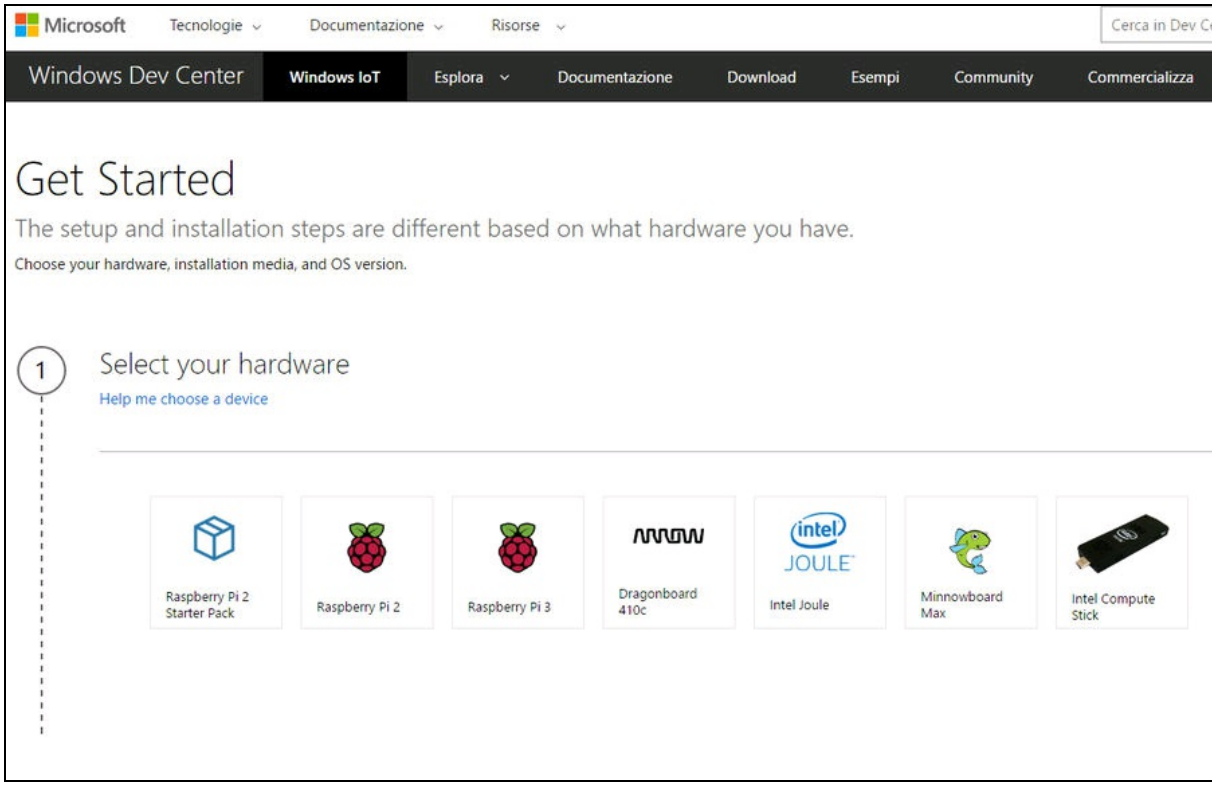


Figura 3.29 La pagina Get Started del portale di Windows 10 IoT Core.

La prima pagina invita a selezionare il proprio hardware: *Select your hardware* propone la scelta dell'hardware su cui installare Windows 10 IoT Core. Al momento (maggio 2017) le schede disponibili sono le seguenti:

- Raspberry Pi 2;
- Raspberry Pi 3;
- Arrow Dragonboard 410C;
- Intel Joule;
- Intel Compute Stick;
- Minnowboard Max (by Intel).

Oltre a queste schede è disponibile il supporto per Arduino.

Dato che le schede Arrow e Intel sono praticamente sconosciute nel nostro Paese, la scelta ricadrà inevitabilmente sulle schede Raspberry Pi.

Si fa notare che il supporto per il modello 2 e 3 di Raspberry Pi è lo stesso.

Una volta scelta la scheda, si potrà selezionare il tipo di installazione, ovvero se si vuole optare per una card SD vuota oppure una con NOOBS (*New Out Of Box Software*), l'immagine multi OS di Raspberry Pi.

Si consiglia di installare Windows 10 IoT Core su una card vuota e di selezionare il sistema operativo Windows 10 IoT Core. Il sistema operativo Windows 10 IoT Core Insider Preview prevede l'installazione di Windows 10 per sviluppatori.

Con il pulsante *Next* si passa alla fase successiva, ovvero l'installazione nella card di Windows 10 IoT Core.

Installazione di Windows 10 IoT Core

In questa pagina viene proposta una procedura di installazione tramite *Dashboard*. Viene chiesto di verificare che il sistema operativo in esecuzione sia aggiornato alla build 14393 o superiore. Si può trovare il numero di build corrente facendo clic sul pulsante *Start*, digitando `winver` e premendo il tasto Invio.

Facendo clic sul pulsante *Download Dashboard* verrà scaricato il file `setup.exe`, che permette di aprire il pannello *IoT Dashboard* (Figura 3.30).

Si consiglia di inserire una card microSD di almeno 8 GB nello slot per card del PC o in un lettore di card.

Con l'opzione *Configura un nuovo dispositivo* si aprirà una finestra in cui sono disponibili i dispositivi IoT supportati:

- *Raspberry Pi 2 & 3*;
- *MinnowBoard MAX*;
- *Dragonboard 410c*;
- *Personalizzato*.

Con l’opzione *Personalizzato* è possibile creare una card da un’immagine scaricata in precedenza.

Selezionando dal menu a discesa una scheda, verrà scelto automaticamente il file appropriato di installazione di Windows 10 IoT Core. Nell’esempio è stato scelto il dispositivo *Raspberry Pi 2 & 3* e il relativo file *Windows 10 IoT (15063)*. Ovviamente il numero di build sarà diverso nelle versioni future.

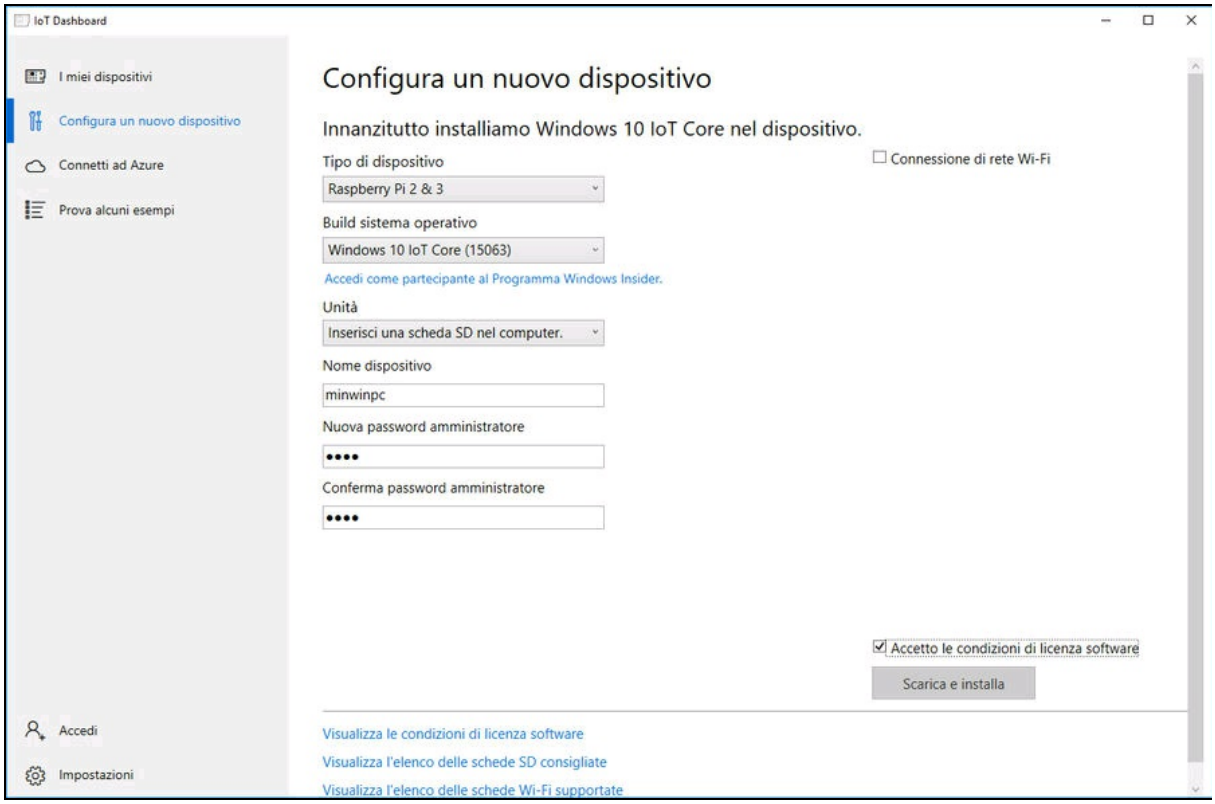


Figura 3.30 Finestra principale IoT Dashboard.

Nella sezione *Unità* viene visualizzata la card microSD inserita nel PC.

Nella casella di testo *Nome del dispositivo* appare il nome predefinito `minwinpc` del dispositivo, che si può cambiare a piacere.

Nelle caselle *Nuova password amministratore* e *Conferma password amministratore* bisogna inserire una password qualsiasi per poter accedere in seguito al portale del dispositivo (si veda più avanti).

Dopo aver messo il segno di spunta su *Accetto le condizioni di licenza software*, facendo clic sul pulsante *Scarica e installa* inizia il download del file immagine e la successiva installazione dell'immagine nella card.

Collegamento alla scheda Raspberry Pi

Facendo riferimento alla Figura 3.31, collegare le porte della scheda Raspberry Pi in questo modo.

- Inserire nello slot la scheda microSD preparata tramite *Dashboard*.
- Collegare alla porta Ethernet della scheda un cavo al router della rete locale (Raspberry Pi 2) oppure direttamente al PC, ma in questo caso non si avrà la connettività Internet.
- In alternativa, è possibile collegare un dongle Wi-Fi a una porta USB della scheda Raspberry Pi 2. In questo caso, bisognerà effettuare la connessione Wi-Fi da Raspberry Pi 2.
- Con Raspberry Pi 3 ci si può collegare alla rete Wi-Fi senza utilizzo di dongle USB.
- Collegare un monitor o una TV con porta HDMI alla porta HDMI della scheda.
- Collegare una tastiera e un mouse alle porte USB.
- Collegare l'alimentatore alla presa DC In della scheda. Dato che la presa di alimentazione della scheda è di tipo micro-USB, si consiglia vivamente di lasciare la spina dell'alimentatore sempre collegata e di spegnere e accendere la scheda tramite una ciabatta commutata. In questo modo si evita di danneggiare la delicata presa della scheda.

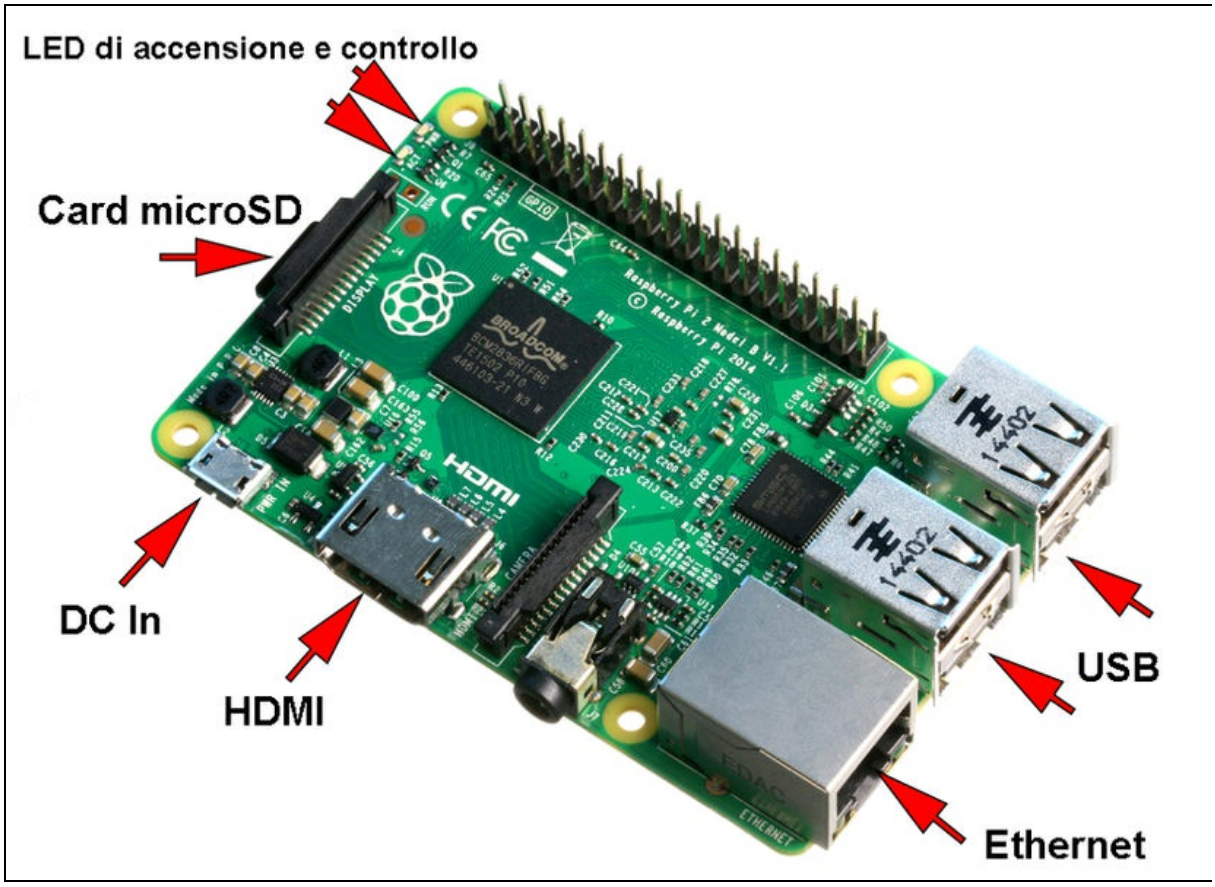


Figura 3.31 Le connessioni della scheda Raspberry Pi 2/3.

Al primo avvio di Windows 10 IoT Core verrà chiesto di selezionare la lingua e la tastiera italiana. Al termine del setup si vedrà apparire una schermata simile alla Figura 3.32. Qui sono disponibili le seguenti informazioni.

- *Nome dispositivo*: `minwinpc`.
- *Rete*: `Ethernet`.
- *Indirizzo IP*: `192.168.1.xxx` (dove *xxx* è un numero fornito dal router).
- *Versione SO*: `10.0.15063.0`.
- *Dispositivi connessi*: `USB Optical Mouse`, `USB Keyboard`.
- *Informazioni rete*: `IPv6`, `IPv4`, `Stato`.

Il nome di default del dispositivo è `minwinpc`, se non è stato modificato nel pannello *Dashboard*.

La rete è Ethernet o Wi-Fi e l'indirizzo IP viene dato dal router di casa e di solito è del tipo `192.168.1.`*xxx*, dove *xxx* è un numero assegnato dal server DHCP. Se si collega la scheda direttamente alla porta Ethernet del PC senza passare da un router, l'indirizzo IP sarà di tipo `169.254.1.xxx`, dove *xxx* è un numero assegnato dalla scheda del proprio PC.

La versione del sistema operativo `10.0.15063.0` (maggio 2017) deve essere la stessa del sistema Windows 10 correntemente in esecuzione sul PC.

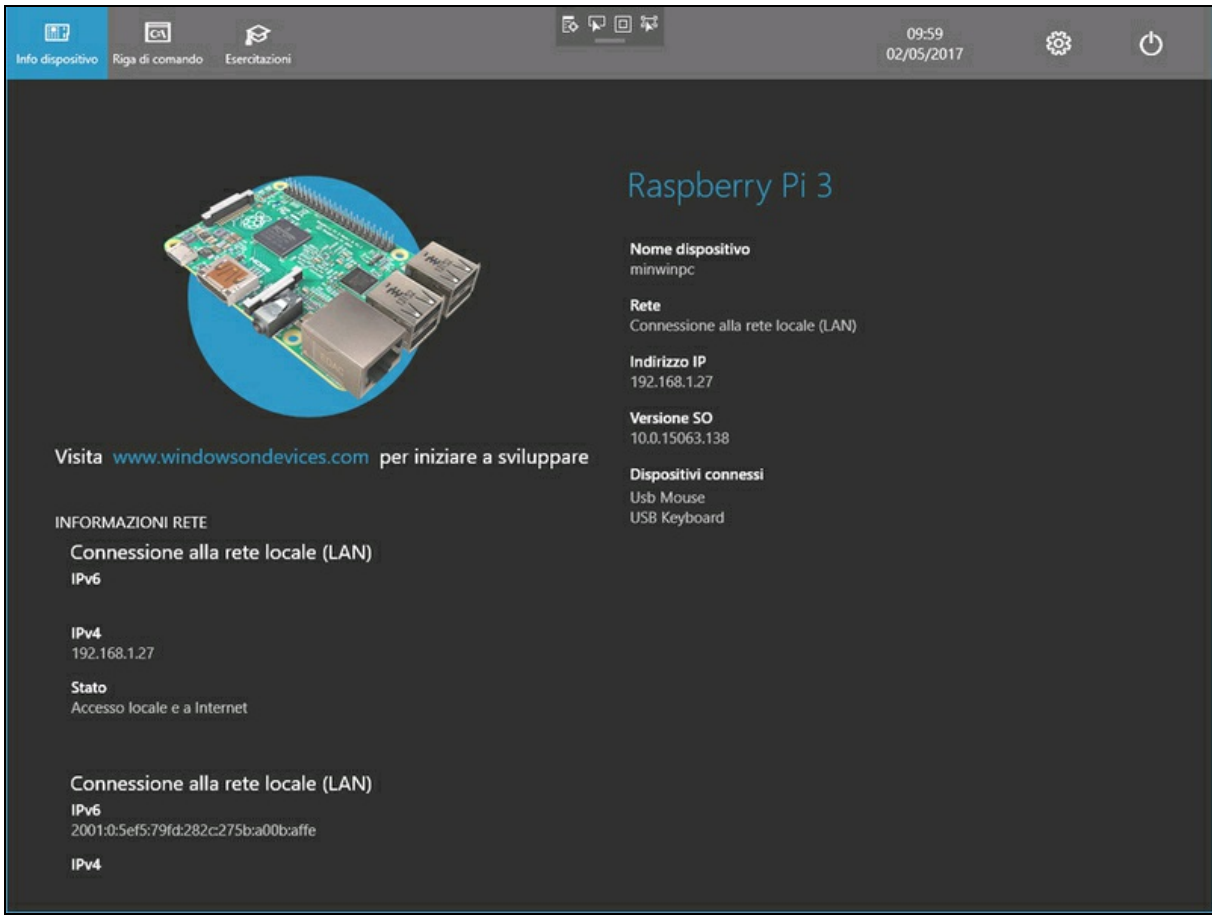


Figura 3.32 La schermata principale di Windows 10 IoT Core per Raspberry Pi 3.

I dispositivi connessi sono la tastiera e il mouse e qualsiasi altro dispositivo riconosciuto. Per una lista di periferiche compatibili si veda la pagina all'indirizzo http://ms-iot.github.io/content/en-US/win10/SupportedInterfaces.html.

Connessione a Raspberry Pi da Dashboard

Dopo l'avvio di Windows 10 IoT Core è possibile effettuare la connessione al dispositivo tramite *Dashboard*. Facendo clic sul pulsante *I miei dispositivi* apparirà una finestra simile alla Figura 3.33. Qui vengono visualizzate le informazioni del dispositivo collegato:

- il nome di default;
- il tipo;
- la versione del sistema operativo;
- l'indirizzo IP.

Con il tasto destro del mouse sul nome del dispositivo si può aprire il menu contestuale e selezionare la voce *Apri in Device Portal*, come indicato dalla freccia.

Si aprirà la home page del dispositivo nel browser di sistema. All'apertura viene visualizzata una finestra in cui immettere il nome utente (Administrator) e la password scelta al momento della creazione della card.

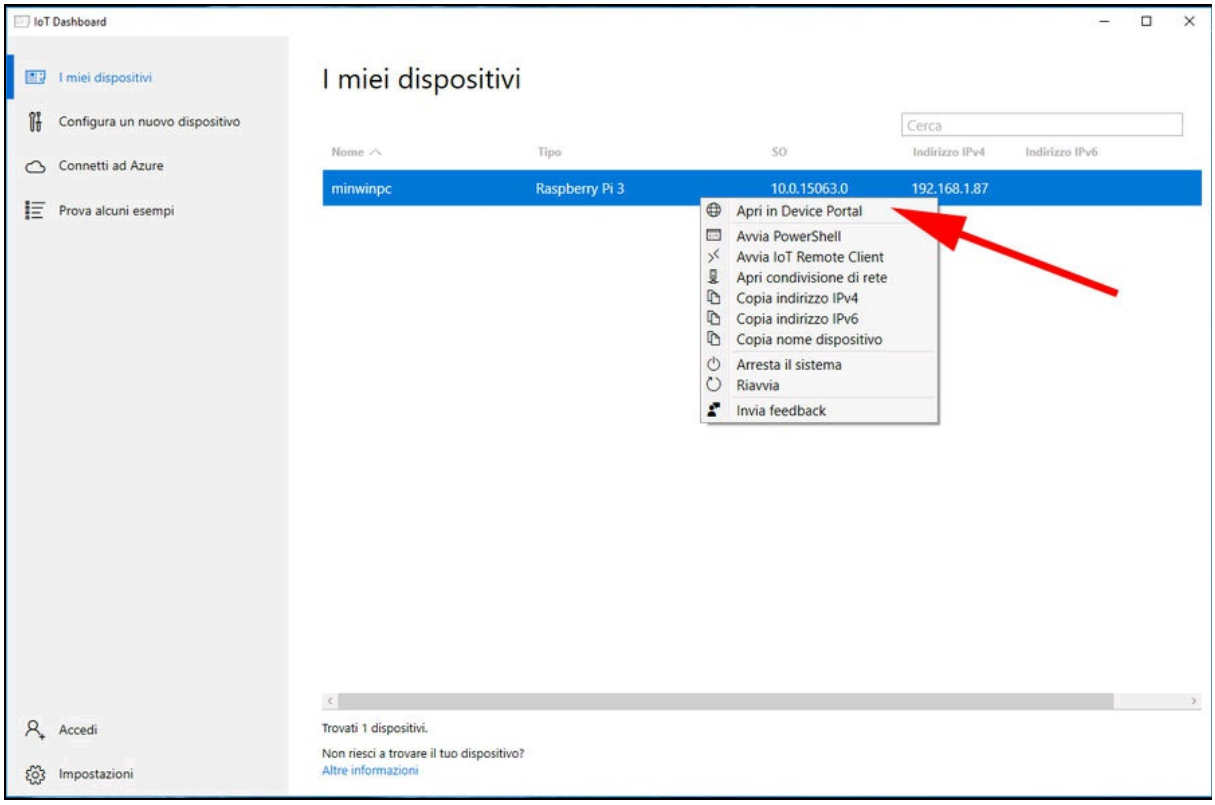


Figura 3.33 Finestra “I miei dispositivi” del pannello Dashboard.

La home page del portale del dispositivo è utile per poter monitorare e cambiare molte cose. Sulla barra laterale sinistra della home page è presente un menu per accedere a varie funzioni del dispositivo.

- *Apps*: visualizza le applicazioni installate nel dispositivo e consente di installarne delle nuove. È possibile avviare un’applicazione o rimuoverla e impostare l’applicazione di avvio (si veda più avanti).
- *Processes*: visualizza i processi in esecuzione nel dispositivo. È possibile anche eseguire un comando personalizzato.
- *Debug*: visualizza le informazioni relative al debug.
- *Devices*: visualizza lo stato dei dispositivi installati e del *Default Controller Driver*, che dovrà essere cambiato a seconda del tipo di applicazioni con le librerie Wiring di Arduino.

- *Connectivity*: visualizza le informazioni e permette di cambiare le connessioni Bluetooth, Wi-Fi, Ethernet e così via.
- *Windows Update*: pagina per il controllo degli aggiornamenti Microsoft.
- *Remote*: Windows IoT Remote Server consente agli utenti di connettersi da un altro computer a questo dispositivo tramite una connessione di rete, utilizzando un'applicazione client. Per accedere al dispositivo IoT Core Windows, installare l'applicazione client remoto da `https://www.microsoft.com/store/apps/9nblggh5mnxz`.
- *Scratch*: non ancora implementato (maggio 2017).
- *Tutorial*: link al portale Microsoft.

Visual Studio Community 2017

Dopo aver collegato Raspberry Pi al PC e aver controllato la connessione al dispositivo remoto, il passo successivo è quello di installare la versione gratuita di Visual Studio 2017.

Visual Studio 2017 (disponibile in versione definitiva dal marzo 2017) è un potente ambiente di sviluppo per la creazione di applicazioni universali per Windows (UWP), Android, iOS e dispositivi IoT, nella fattispecie, Raspberry Pi e Arduino.

La versione gratuita di Visual Studio 2017 si chiama *Community* ed è completa di tutte le caratteristiche per lo sviluppo di applicazioni "non-aziendali". Volendo si può estendere la licenza acquistando la versione Professional per singoli sviluppatori e piccoli team o la versione Enterprise aziendale con funzionalità avanzate per team che lavorano a grandi progetti DevOps.

I linguaggi di programmazione inclusi in ogni versione di Visual Studio 2017 sono:

- Visual C#;

- Visual Basic;
- C++;
- ASP.net;
- Office;
- Python;
- Node.js;
- HTML/JavaScript;
- Azure;
- Xamarin;
- Unity.

Si fa notare che la versione 2017 ha aggiunto gli ambienti di programmazione Xamarin, per lo sviluppo di applicazioni Android e iOS, e Unity, per lo sviluppo di giochi.

Dopo aver scaricato e lanciato l'installer, si può selezionare l'installazione standard di circa 15 GB o l'installazione personalizzata dei molti pacchetti disponibili. Se si installano tutti i pacchetti, si arriva a 70 GB di installazione.

La versione gratuita di Visual Studio 2017 è una grossa opportunità sia per lo sviluppatore esperto sia per il neofita. Soprattutto a chi non ha esperienza di programmazione a basso livello (come C++, Python e Node.js), l'ambiente "visuale" di Visual C#, Visual Basic o Visual C++ sarà di grande aiuto per la creazione di interfacce grafiche per l'utente che interagiscono con l'hardware di Raspberry Pi e Arduino.

La novità introdotta con la versione 2015 di Visual Studio è quella di poter creare applicazioni universali, altrimenti dette UWP (*Universal Windows Platform*) che possono essere compilate indifferentemente per la piattaforma locale (×86 e ×64) oppure, nel nostro caso, per il dispositivo remoto ARM (Raspberry Pi).

Tutte le informazioni e il download di Visual Studio 2017 sono disponibili presso il sito Microsoft: https://www.visualstudio.com/it/vs.

ATTENZIONE

Dopo l'installazione, prima di avviare Visual Studio 2017, è necessario abilitare la modalità sviluppatore nel PC per poter aprire un progetto UWP. Se si apre un progetto UWP e la modalità sviluppatore non è abilitata, viene visualizzato automaticamente il pannello *Impostazioni*. Per usare le funzionalità per sviluppatori:

- aprire il pannello *Impostazioni*;
- fare clic su *Aggiornamento e sicurezza*;
- scegliere l'opzione *Per sviluppatori*;
- scegliere *Modalità sviluppatore*.

Hello Blinky

Dopo aver installato Visual Studio 2017, il passo successivo è quello di provare qualche esempio del portale, per vedere se tutto funziona. Oltre a quelli online, tutti gli esempi sono scaricabili dal repository https://github.com/ms-iot/samples. Questo repository viene aggiornato costantemente, per cui vale la pena visitarlo di tanto in tanto o quando si installa un aggiornamento di Windows 10.

Il primo esempio proposto nel portale si chiama "Hello blinky", l'equivalente di un Hello world dell'elettronica che fa lampeggiare un LED collegato a un pin della porta GPIO di Raspberry Pi.

Il codice è disponibile per C# e C++. La descrizione del circuito e il codice commentato per C# è disponibile nel portale online all'indirizzo http://ms-iot.github.io/content/en-US/win10/samples/Blinky.htm.

Il circuito prevede l'uso di una breadboard, un LED e una resistenza da 220 Ω. Il catodo del LED va collegato alla porta GPIO 5 mentre l'anodo va collegato con la resistenza in serie all'alimentazione positiva di 3,3 volt, come illustrato nella Figura 3.34. Si noti che il collegamento del LED è diverso dal solito. Questa configurazione è comunemente

nota come *Active Low*, ovvero il LED si accende quando il catodo viene portato a 0.

Per l'esecuzione e/o la modifica dell'esempio Hello Blinky si veda il prossimo paragrafo.

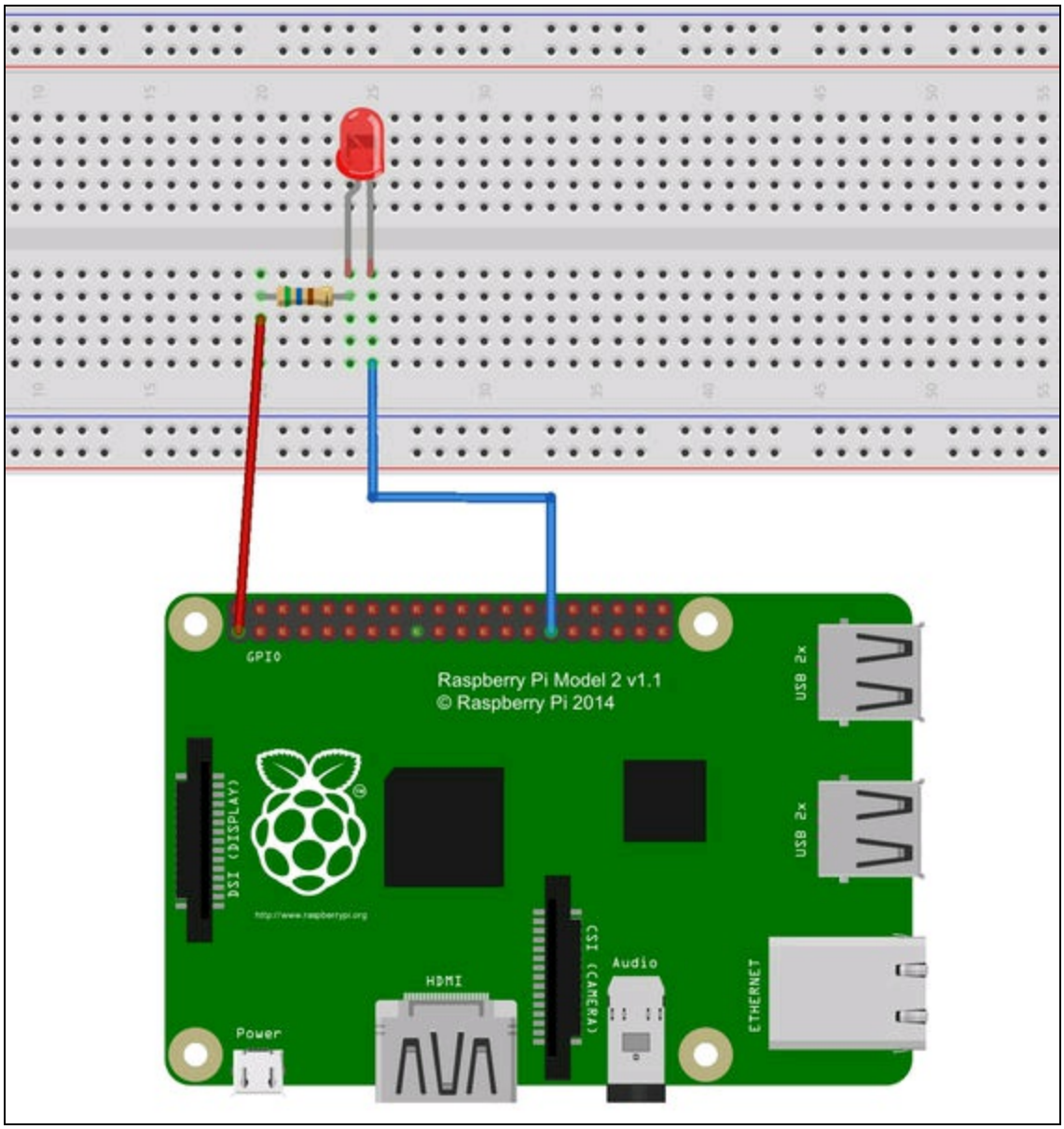


Figura 3.34 Il cablaggio del LED alla scheda Raspberry Pi 2/3.

NOTA

Di tutti gli ambienti di programmazione disponibili in Visual Studio 2017, verrà trattato solamente il C#.

Visual C#

Il linguaggio C# (il simbolo diesis, #, si pronuncia *sharp*) è un linguaggio di programmazione orientato agli oggetti sviluppato da Microsoft per il framework .NET.

La sintassi del C# prende spunto da Delphi, C++ e Visual Basic. Per gli strumenti di programmazione visuale si appoggia sul linguaggio XAML (si pronuncia "*csamol*").

XAML, è l'abbreviazione di *eXtensible Application Markup Language* ed è un linguaggio di markup basato sulla sintassi XML, orientato però alla creazione di interfacce grafiche.

Per lo scopo di questo libro, non verrà spiegata la sintassi del linguaggio, ma l'uso del C# limitatamente all'apertura e la distribuzione dei progetti di esempio. Vengono date anche alcune indicazioni su come creare applicazioni UWP o come modificare il codice degli esempi.

Un esempio

Dal menu *File* > *Nuovo* > *Progetto* aprire la finestra dei template, cioè dei modelli disponibili con i quali iniziare un progetto. Nella finestra *Nuovo progetto* sono visibili le indicazioni che servono a creare un'applicazione universale:

- il framework .NET più recente nel menu a discesa posto in alto;
- il modello *Visual C#* > *Windows* > *Universale*:
- la voce *App vuota (Windows Universale)*;
- *Nome*.

Nella casella *Nome* si consiglia di digitarne uno che si possa riconoscere. Per questo esempio, digitare "Test". Il nome di default per l'applicazione è "App", più un numero progressivo che si incrementa ogni volta che se ne crea una. La cartella di default in cui vengono salvati i progetti è `C:\Users\...\Documents\Visual Studio 2017\Projects`.

Si consiglia di lasciare la spunta nell'opzione *Crea directory per la soluzione*, in modo che venga creata anche una cartella contenete tutti file del progetto.

Facendo clic su *OK*, la finestra si chiude, e appare un messaggio con la scelta della versione minima e della destinazione dell'applicazione UWP, come illustrato nella Figura 3.36a. In pratica, con Visual Studio 2017 si può scegliere la versione corrente della versione UWP e quella minima di una versione precedente. Nell'esempio è stata scelta l'opzione Windows 10 Anniversary Edition (10.0: build 14393). Quindi apparirà una barra di progresso *Creazione del progetto 'Test' in corso*.

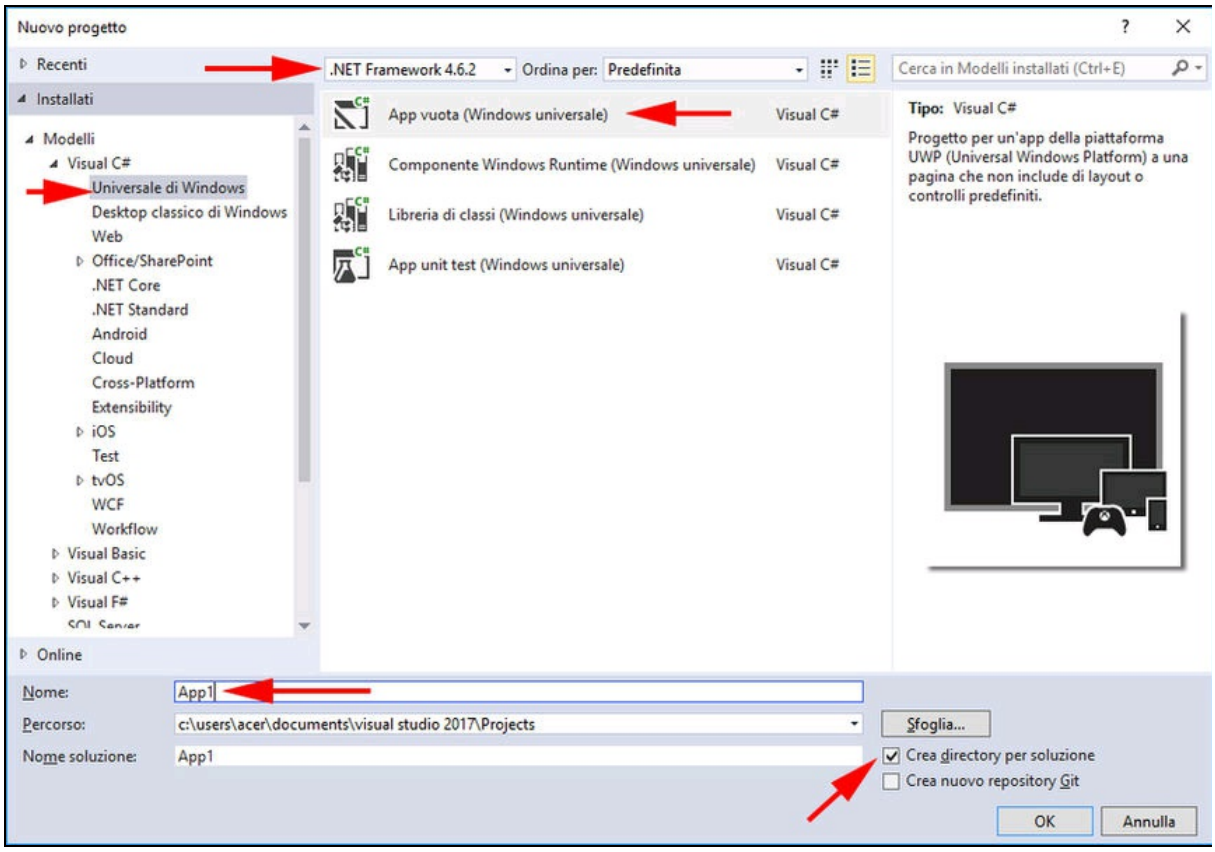


Figura 3.35 Finestra Nuovo Progetto.

Esplora soluzioni

Al termine della creazione del progetto, apparirà nella finestra *Esplora soluzioni* una serie di file e cartelle del progetto chiamato “Test” (Figura 3.36b).

La soluzione viene denominata “Test” come il progetto, ma si può salvare la soluzione con un nome differente. Una soluzione, infatti, può contenere diversi progetti, che possono venire importati con il comando *Aggiungi* dal menu contestuale (Figura 3.36c). Nel caso di più progetti, il progetto designato come predefinito viene visualizzato in grassetto.

Nella finestra *Esplora soluzioni* appaiono i seguenti file (e cartelle) di default.

- `Properties`: questa cartella contiene il file in C# `AssemblyInfo.cs` con le informazioni Assembly e il file XML `Default.rd.xml`, che contiene le direttive runtime utilizzate dalla piattaforma .NET nativa.
- `Riferimenti`: qui vengono elencati i riferimenti caricati di default al momento della creazione del progetto, più quelli eventualmente aggiunti in seguito.
- `Assets`: questa cartella contiene i file grafici in formato png di diverse dimensioni, per adattare lo splashscreeen all’avvio dell’applicazione e altri loghi (si veda più avanti come modificarlo).
- `App.xaml`: è il file dell’applicazione contenente le informazioni XAML dell’interfaccia grafica dell’applicazione.

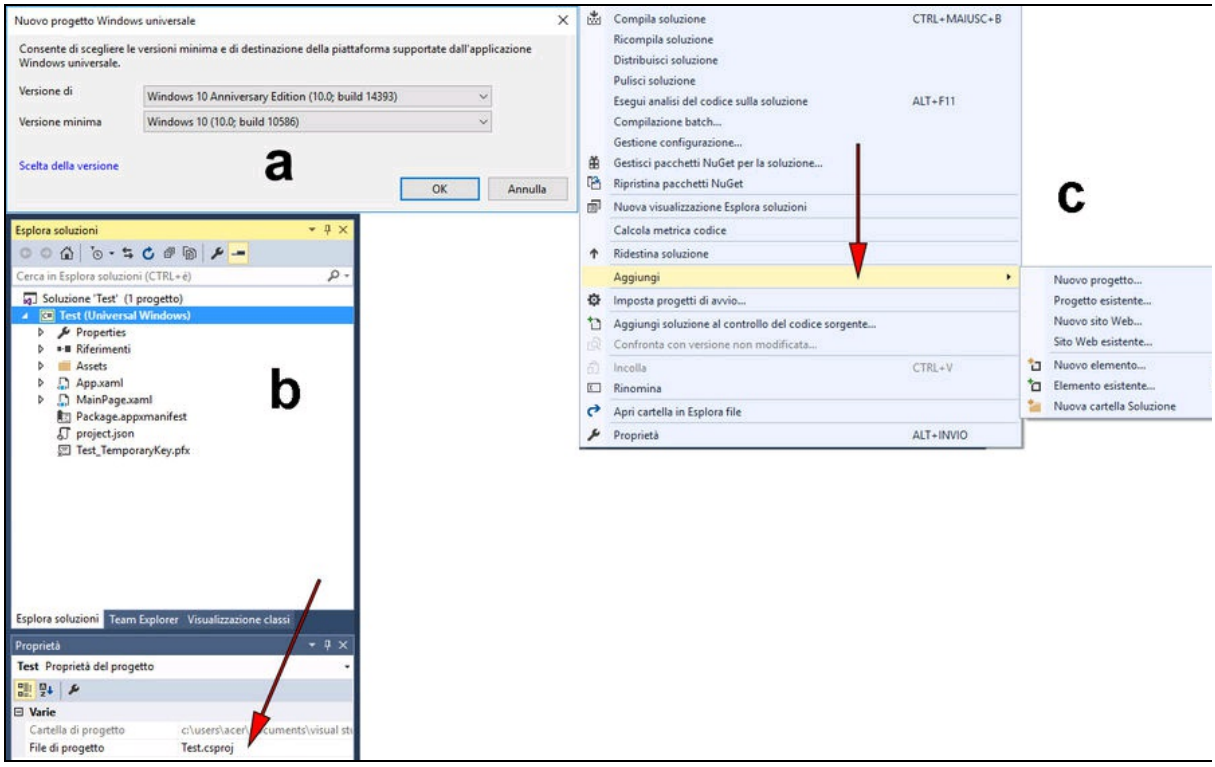


Figura 3.36 Barra di progresso durante la creazione del progetto (a). Il pannello “Esplora soluzioni” (b). Il menu contestuale della soluzione (c).

- `App.xaml.cs`: è il file in C# dell’applicazione XAML con le informazioni per inizializzare i componenti dell’applicazione.
- `MainPage.xaml`: è il file XAML dell’applicazione, ovvero l’interfaccia utente (UI) in cui inserire gli oggetti (pulsanti, cursori, caselle di testo e altro) per il controllo del programma, delle periferiche, della porta GPIO e così via.
- `MainPage.xaml.cs`: è il file C# dell’applicazione, in cui scrivere il codice.
- `Package.appxmanifest`: è il file delle proprietà del pacchetto di distribuzione in cui decidere il nome visualizzato, il punto di ingresso, la lingua predefinita, la rotazione della finestra e altre impostazioni (si veda più avanti).

- `Project.json`: è il file JSON creato automaticamente in cui vengono definite le dipendenze, il framework e i tipi di runtime.

Ogni volta che si seleziona una cartella o un file, nel pannello *Proprietà* appaiono le proprietà corrispondenti.

La pagina `App.xaml.cs` non va mai modificata, se non in casi eccezionali. Serve a inizializzare l'oggetto `Application singleton`. Come dice il commento "Si tratta della prima riga del codice creato e, come tale, corrisponde all'equivalente logico di main() o WinMain()".

La pagina che ci interessa aprire è `MainPage.xaml.cs`, contenuta nella cartella `MainPage.xaml` (Figura 3.37), ovvero la pagina principale dell'applicazione. Questa pagina contiene il codice C# che bisognerà scrivere o modificare.

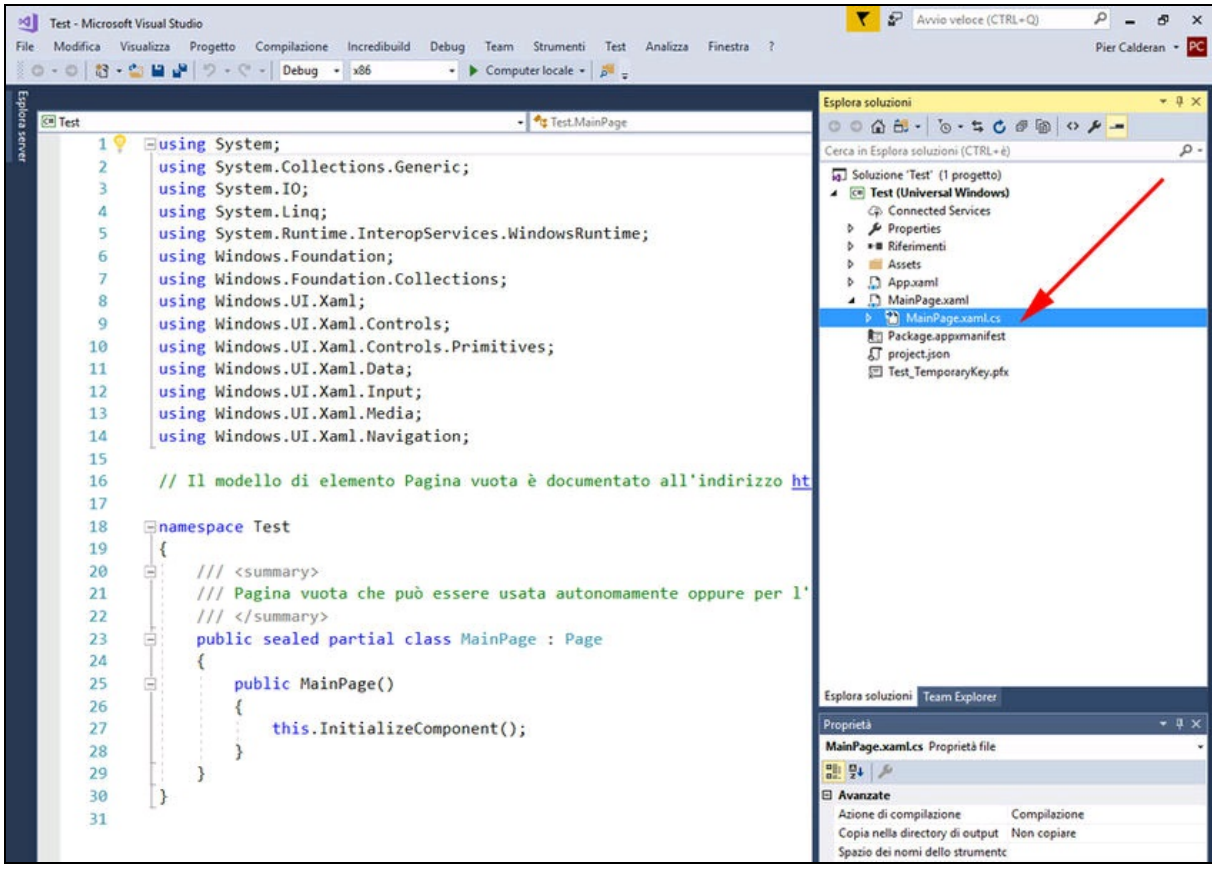


Figura 3.37 La pagina di codice MainPage.xaml.cs.

Il codice creato di default è simile al seguente listato:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
// Il modello di elemento Pagina vuota è documentato all'indirizzo
// https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x410
namespace Test
{
    /// <summary>
    /// Pagina vuota che può essere utilizzata autonomamente oppure esplorata
all'interno di un frame.
    /// </summary>
 public sealed partial class MainPage : Page
 {
        public MainPage()
        {
            this.InitializeComponent();
        }
    }
}
```

La direttiva `using` serve a fornire i riferimenti specifici al programma. Si fa notare che molte righe `using` all'inizio del listato appaiono in grigio, perché non sono necessarie. Spesso succede che vengano aggiunte funzionalità superflue che possono venire ignorate. Per toglierle basta solo posizionare il mouse su una riga in grigio e apparirà l'icona di una lampadina che apre un menu da cui selezionare l'opzione *Rimuovi istruzioni using non necessarie*, come illustrato nella Figura 3.38.

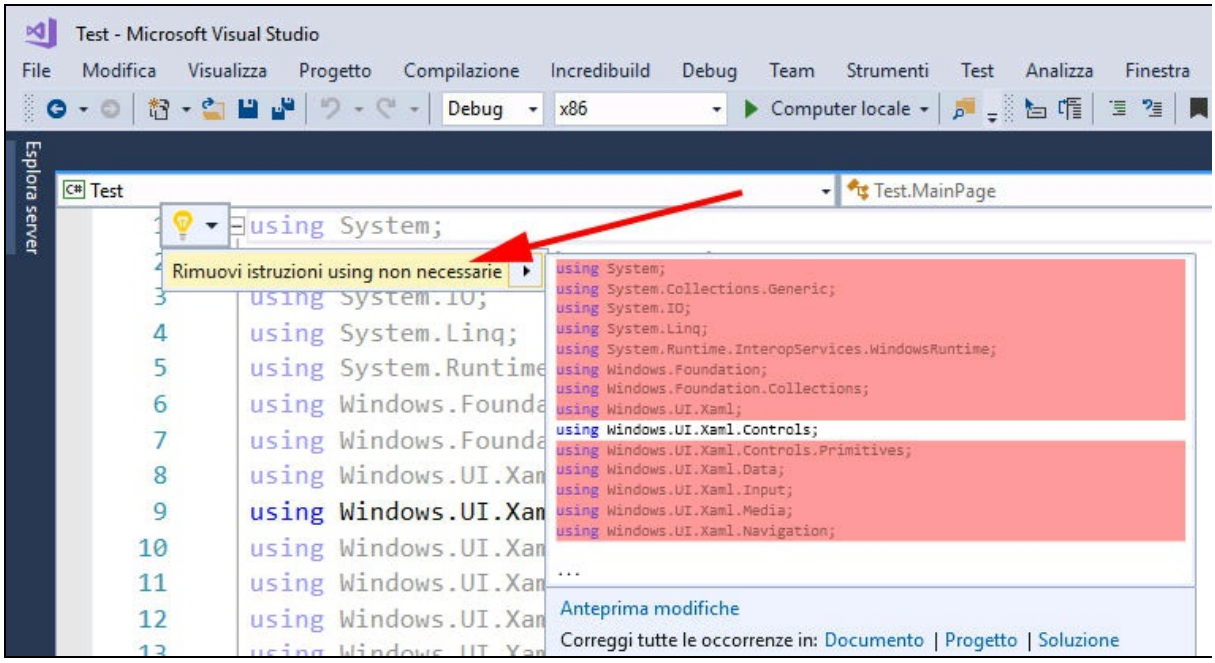


Figura 3.38 Opzione Rimuovi istruzioni using non necessarie.

Sotto le istruzioni `using` inizia il `namespace`, cioè lo spazio dei nomi, con il nome del progetto creato. Nel nostro caso il `namespace` è `Test`, cioè i nome del progetto. Attenzione: una volta creato, non si può cambiare il `namespace`.

Il commento `summary` (sommario) dice `"Pagina vuota che può essere utilizzata autonomamente oppure esplorata all'interno di un frame."`. Questo significa si possono scrivere all'interno del `namespace` tutte le funzioni desiderate. La funzione principale dell'applicazione è:

```
public sealed partial class MainPage : Page
```

Al suo interno vanno inserite tutte le funzioni utente.

La parola chiave `sealed` significa semplicemente che la classe non può essere ereditata. Non ha effetto sul modo in cui il codice della classe viene strutturato. La parola chiave `partial` consente semplicemente a una classe di essere suddivisa in diversi file. Nella programmazione orientata

agli oggetti, una classe è una struttura per creare oggetti. In pratica, un "oggetto" è l'istanza di una classe.

La prima funzione creata automaticamente all'interno della classe `MainPage` è `public MainPage()`.

L'unica istruzione di default al suo interno è `this.InitializeComponent()`, che serve unicamente a inizializzare i componenti dell'applicazione, contenuti in `App.xaml` e `App.xaml.cs`.

Compilazione e distribuzione

Dalla barra degli strumenti di Visual Studio selezionare la piattaforma ×86 o ×64 (a seconda del sistema operativo installato o del PC per cui si vuole compilare la soluzione) e fare clic su *Computer locale* per compilare l'applicazione per il computer locale (Figura 3.39a). Si può avviare la compilazione e la distribuzione anche con il tasto F5.

Per compilare la stessa applicazione per Raspberry Pi, scegliere la piattaforma ARM dal menu a discesa, illustrato nella Figura 3.39b. Una volta selezionata la piattaforma ARM, si deve selezionare *Computer remoto* dal menu a discesa. Si aprirà una finestra di dialogo per la scelta del dispositivo remoto collegato, il cui nome dovrebbe apparire sotto la voce *Rilevamento automatico* (Figura 3.39c).

Se appare il nome del dispositivo `minwinpc`, lo si può selezionare con il tasto *Seleziona*. Nel caso non apparisse il nome del dispositivo si può digitare nella casella *Configurazione manuale* il nome del dispositivo `minwinpc` o l'indirizzo IP, per esempio, `192.168.1.xxx`, dove `xxx` è il numero assegnato dal router. Controllare che sia selezionata l'opzione *Universale (protocollo non crittografato)* nel menu *Modalità di autenticazione* e fare clic sul pulsante *Seleziona*. Ricordiamo che l'indirizzo IP da inserire è quello che appare nella schermata principale di Raspberry Pi o nel pannello Dashboard.

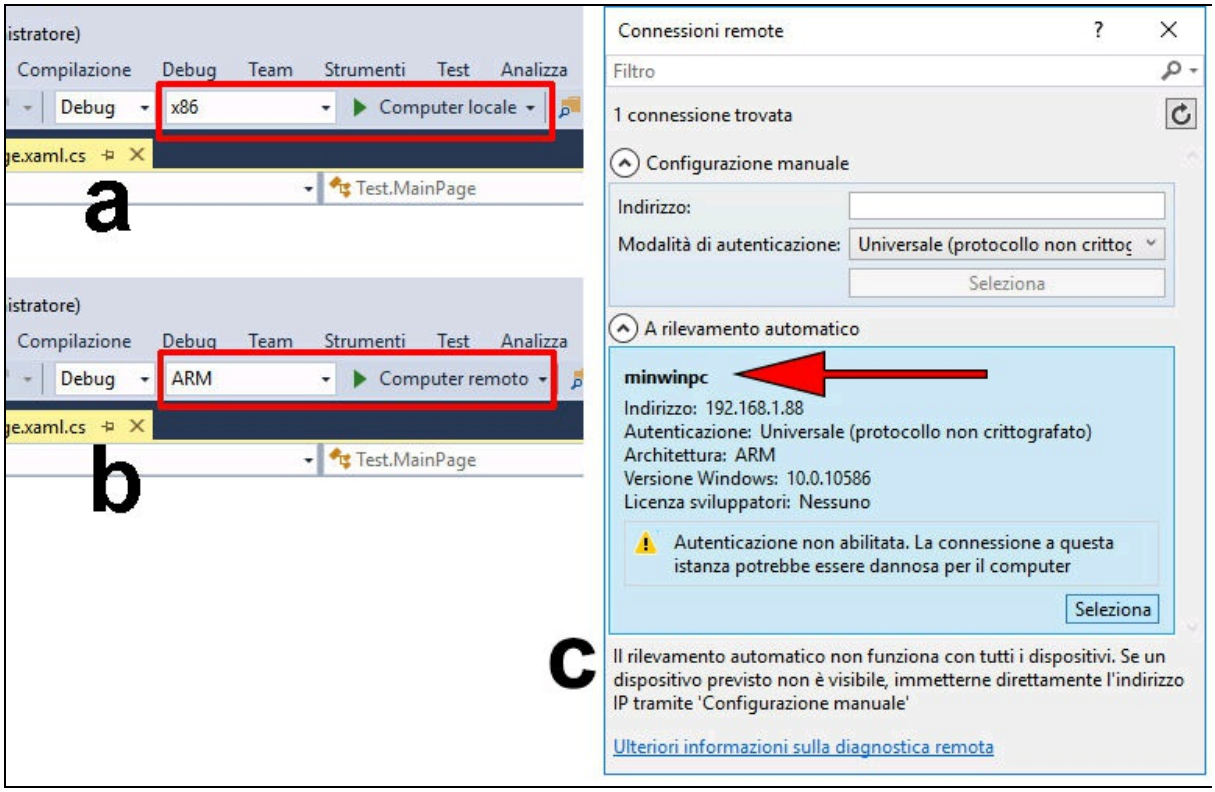


Figura 3.39 Il menu per la scelta della piattaforma ARM (a). La finestra per il rilevamento del dispositivo remoto (b). Finestra delle connessioni remote con Raspberry Pi (c).

Senza aggiungere nulla al codice di default, l'applicazione può essere compilata e distribuita per il computer locale o per il computer remoto, dipende dalla piattaforma scelta nel menu omonimo.

Facendo clic direttamente sulla casella *Computer remoto* o premendo il tasto F5, inizierà la compilazione per piattaforma ARM e la distribuzione dell'applicazione nella card di Raspberry Pi.

Si noti che la configurazione della compilazione può essere impostata su *Debug* o su *Release*, se non si vogliono includere le informazioni di debug.

Nella console di output appariranno le informazioni relative alla compilazione e alla distribuzione (Figura 3.40a) e subito dopo apparirà la schermata dell'applicazione (Figura 3.40b). Essendo un'applicazione vuota, la finestra sarà vuota. In alto a sinistra appaiono due numeri,

perché la configurazione per la compilazione è impostata come *Debug*. Se si compila con la configurazione *Release*, i due numeri non appariranno.

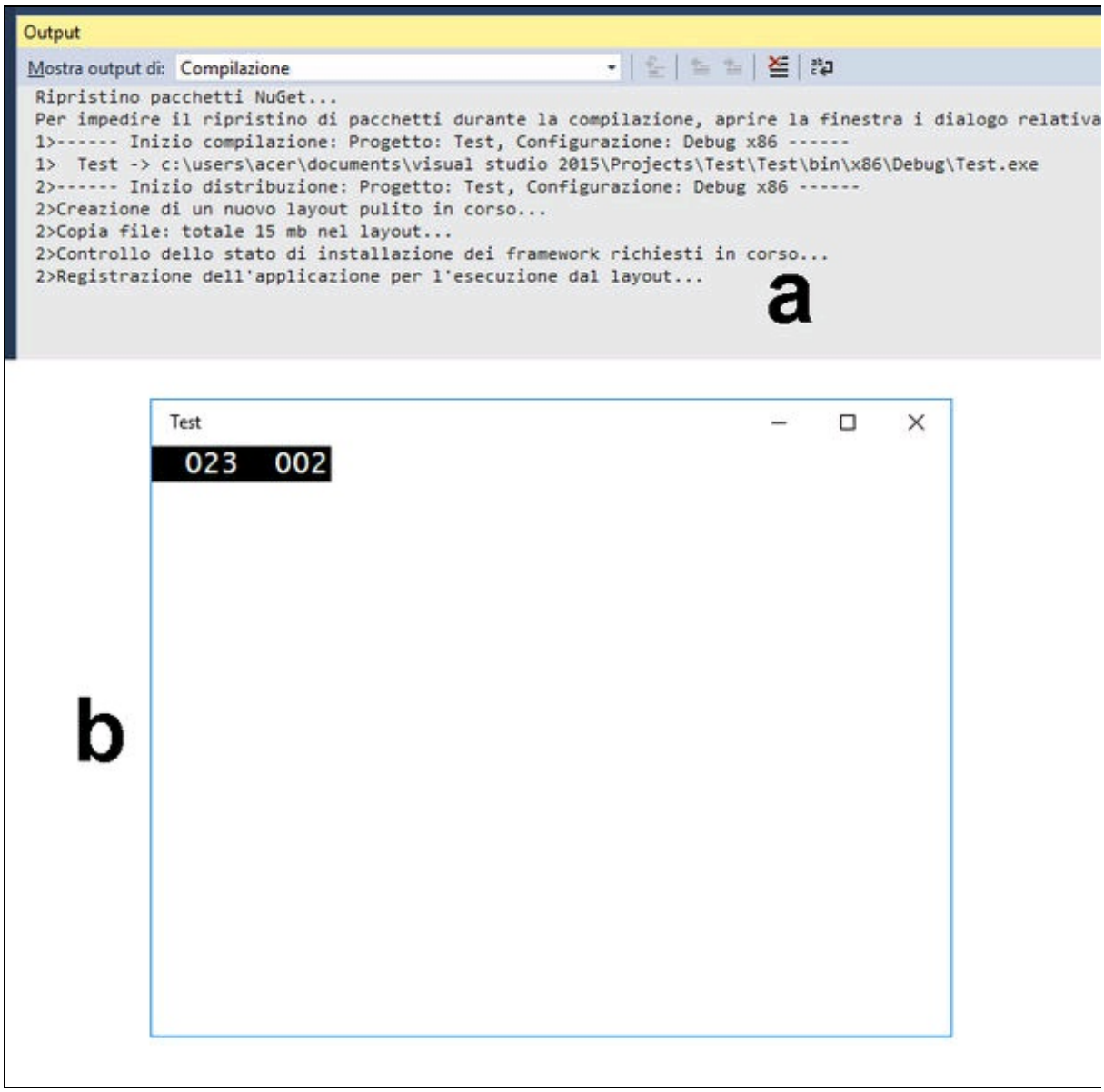


Figura 3.40 La console di output (a). La schermata dell’applicazione (b).

Nella console di output appare anche l’avviso *Ripristino pacchetti NuGet*. Per impedire il ripristino di pacchetti durante la compilazione, aprire la finestra di dialogo relativa alle opzioni di Visual Studio, fare

clic sul nodo *Gestione pacchetti* e deselezionare l'opzione *Consenti a NuGet di scaricare i pacchetti mancanti durante la compilazione*.

La gestione dei pacchetti *NuGet* è automatica di default, ma si può impedire che intervenga, aprendo il pannello *Strumenti > Gestione pacchetti NuGet > Impostazione di Gestione pacchetti* (Figura 3.41).

NuGet è un gestore open source di pacchetti per la piattaforma di sviluppo Microsoft. È distribuito come un'estensione di Visual Studio e, a partire da Visual Studio 2012, è preinstallato di default. *NuGet* può essere gestito anche da riga di comando e automatizzato tramite script. Supporta più linguaggi di programmazione, tra cui i pacchetti del framework Microsoft .NET. Per ulteriori informazioni visitare il sito: https://www.nuget.org.

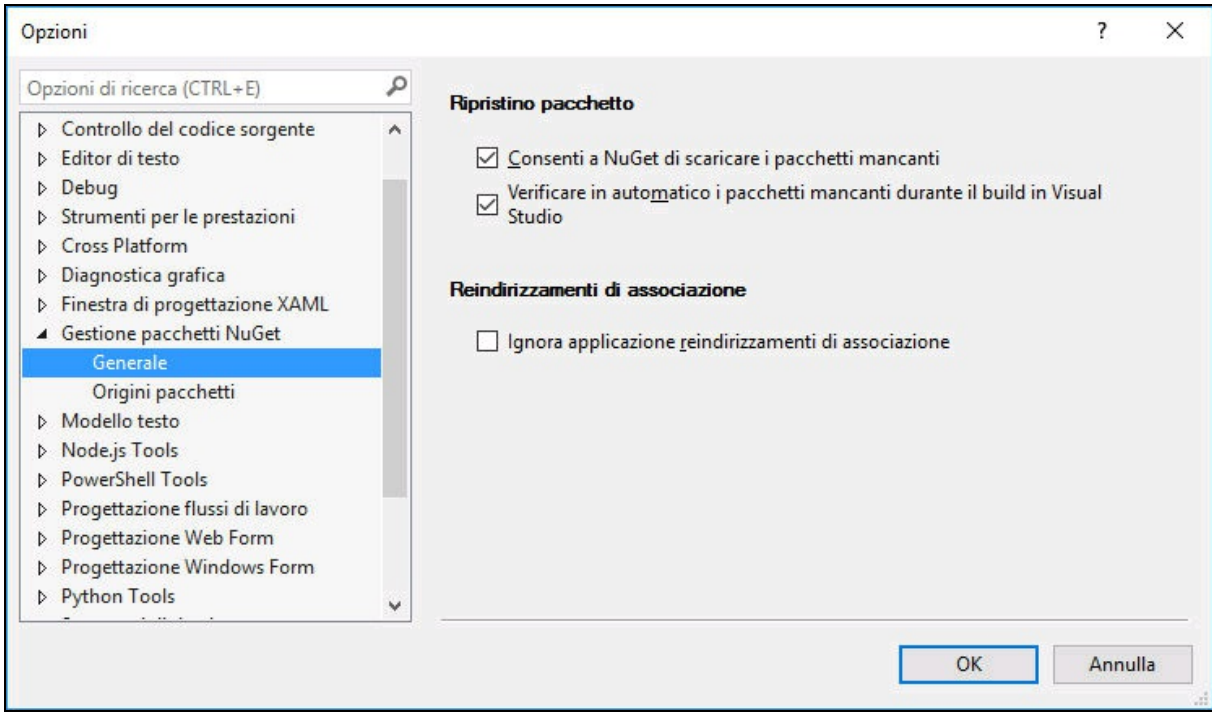


Figura 3.41 La finestra Gestione pacchetti NuGet.

Assets

La cartella *Assets* contiene i file grafici con diverse risoluzioni per adattare lo splashscreeen all'avvio dell'applicazione e altri loghi (Figura 3.42a). I file grafici sono i seguenti:

- `LockScreenLogo.scale-200.png`;
- `SplashScreen.scale-200.png`;
- `Square150x150Logo.scale-200.png`;
- `Square44x44Logo.scale-200.png`;
- `Square44x44Logo.targetsize-24_altform-unplated.png`;
- `StoreLogo.png`;
- `Wide310x150Logo.scale-200.png`.

Sono tutte immagini con un rettangolo barrato, che appare ogni volta che si avvia l'applicazione (Figura 3.42b).

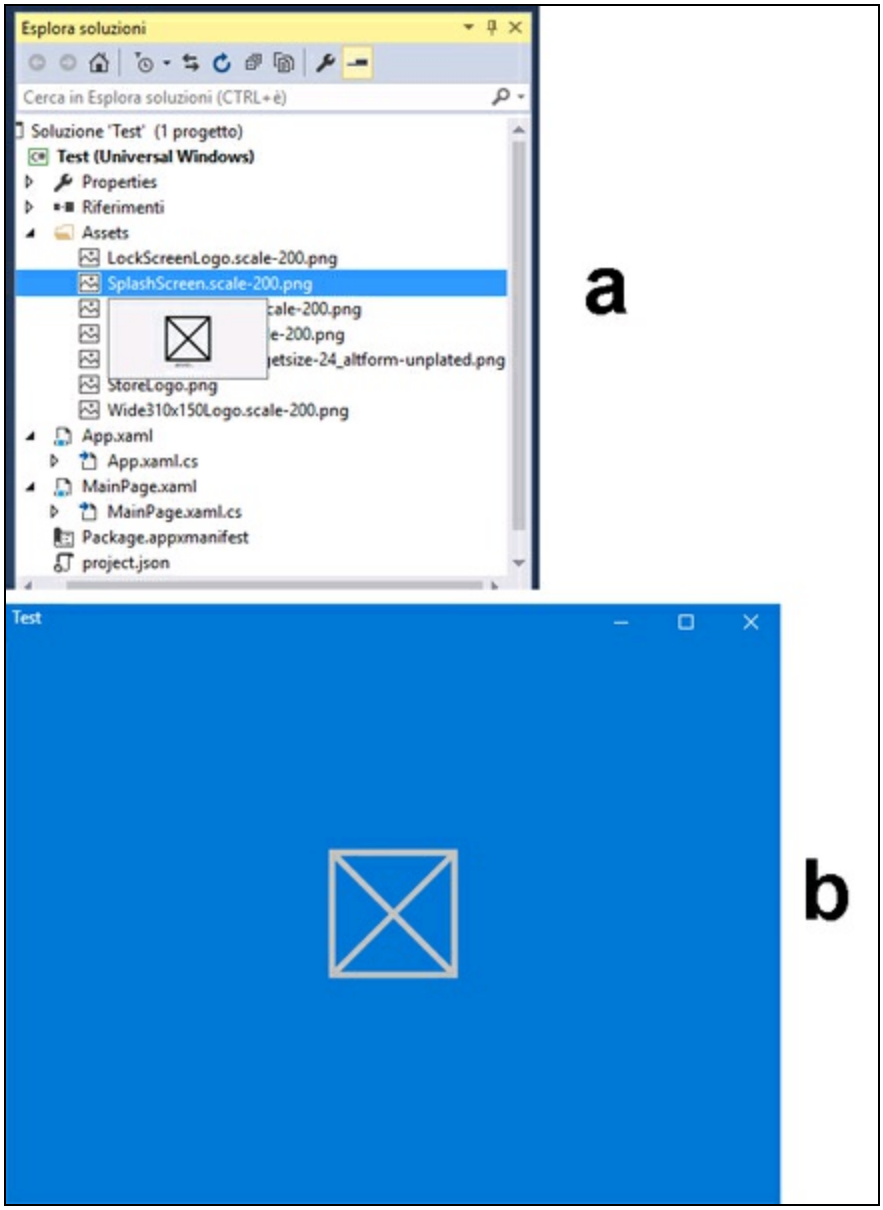


Figura 3.42 La cartella Assets (a). Lo splashscreen d'avvio (b).

Modificare lo SplashScreen

Facendo clic sul file `SplashScreen.scale-200.png` si apre una finestra di modifica simile alla Figura 3.43. Si tratta di un ambiente di editing completo, con tutti gli strumenti per la modifica dell'immagine. Si possono aggiungere testi, scegliere font, tagliare, ruotare, spruzzare o

riempire con il secchiello le forme scegliendo i colori da una tavolozza e molto altro. In alternativa, si può aprire la cartella fisica nel percorso del progetto e modificare l'immagine con un editor DTP come Photoshop o simili.

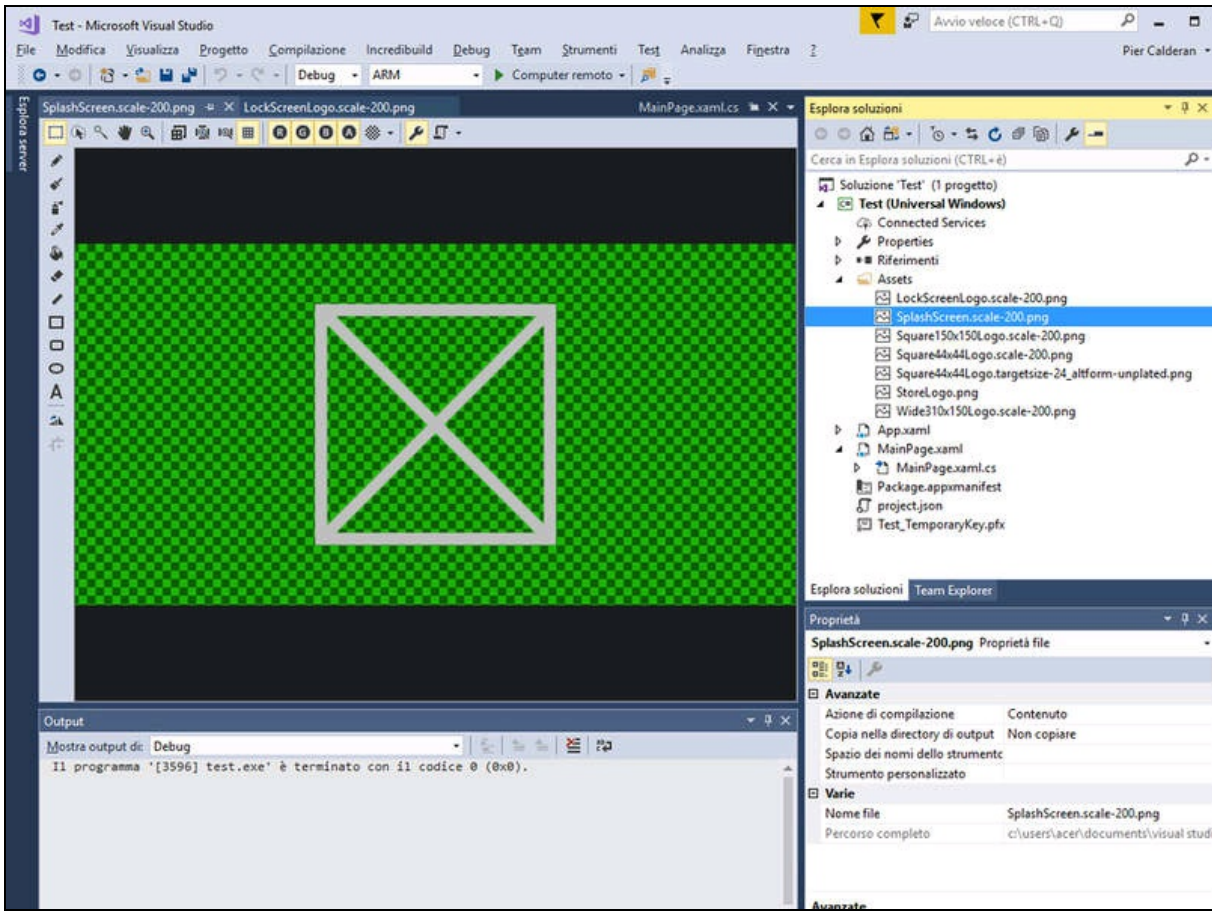


Figura 3.43 La finestra dell'editor di immagini di Visual Studio 2017.

Per aprire la cartella direttamente da Visual Studio, fare clic con il tasto destro del mouse sulla cartella *Assets* e scegliere dal menu contestuale l'opzione *Apri cartella in Esplora File*. Si può effettuare questa operazione anche per la cartella del progetto o della soluzione.

Per impostare lo splashscreen di avvio, con un doppio clic aprire dal pannello *Esplora soluzioni* il file `Package.appxmanifest`. Selezionare la scheda *Asset visivi* e la voce *Schermata iniziale* dalle opzioni sulla

sinistra, come illustrato nella Figura 3.44a. Quindi, selezionare l'immagine desiderata per lo splashscreen di avvio.

Dopo un nostro piccolo "hacking", lavorando direttamente sul file `SplashScreen.scale-200.png`, il risultato è visibile nella Figura 3.44b.

Inoltre, sotto ogni immagine è disponibile un pulsante "…" che apre una finestra di dialogo in cui selezionare un file esterno da importare. Si raccomanda di prestare attenzione alla risoluzione in pixel suggerita, per non creare problemi di taglio dell'immagine.

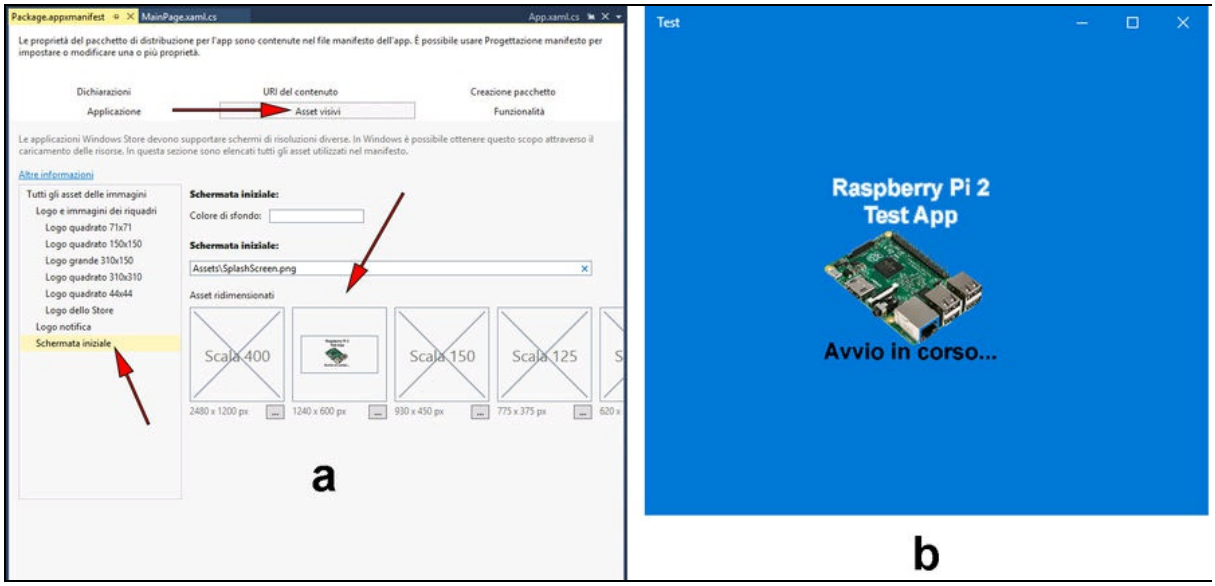


Figura 3.44 La finestra di Package.appxmanifest (a). Lo splashscreen modificato (b).

Nome dell'applicazione visualizzato

Rimanendo nella finestra *Package.appxmanifest*, selezionare la scheda *Applicazione* (Figura 3.45a). Qui si possono impostare diverse opzioni. Ecco le più importanti.

- *Nome visualizzato*: specifica il nome dell'applicazione.
- *Punto di ingresso*: di solito si lascia inalterato, cioè quello dell'applicazione principale.
- *Lingua predefinita*: dovrebbe essere *it-IT*, per l'italiano.

- *Descrizione*: casella di testo per descrivere l’applicazione.
- *Rotazioni supportate*: si può scegliere l’orientamento della schermata dell’applicazione: orizzontale, verticale, orizzontale capovolto, verticale capovolto. Di solito si lascia inalterato su orizzontale.

Facendo clic sulla scheda *Creazione pacchetto* si apre una finestra simile alla Figura 3.45b. Anche qui si possono impostare o visualizzare diverse opzioni. Ecco le più importanti.

- *Nome del pacchetto*: specifica il nome univoco che identifica il pacchetto nel sistema. Qui di solito appare un nome indecifrabile, fatto di cifre e lettere. Per identificare il pacchetto è meglio inserire un nome riconoscibile, per esempio *TestApp*. Il nome non deve contenere spazi, simboli o caratteri di punteggiatura.
- *Nome visualizzato*: è quello scelto nella scheda precedente.
- *Nome visualizzato dell’editore*: è quello del computer cui è stato fornito il certificato di firma digitale.
- *Nome della famiglia di pacchetti*: è il nome del pacchetto più il numero di versione e l’eventuale piattaforma, seguito da un numero univoco di riconoscimento.

Una volta distribuito il pacchetto al computer remoto, il nome del pacchetto viene facilmente individuato nel menu delle applicazione installate, dalla pagina *Apps* nella home page del dispositivo.

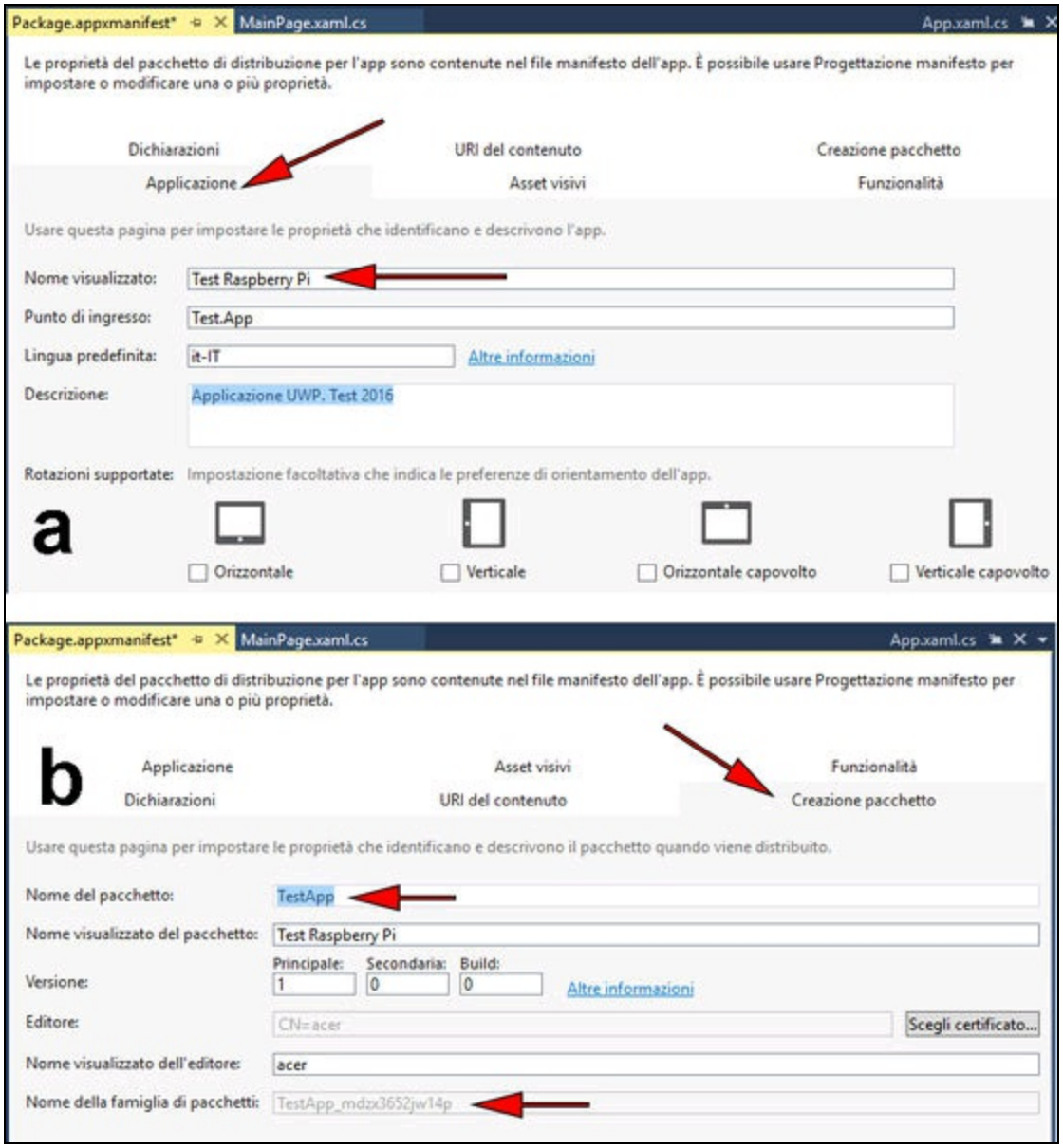


Figura 3.45 La finestra Applicazione di Package.appxmanifest (a). La finestra Creazione del pacchetto di Package.appxmanifest (b).

Lanciare l’applicazione locale senza Visual Studio

Quando un’applicazione viene distribuita al computer locale, la si può lanciare senza aprire il progetto in Visual Studio. Basta fare clic sul menu *Start* di Windows 10 e cercare l’applicazione con l’opzione *Tutte le app*.

Apparirà l'elenco con tutte le applicazioni e, nel nostro caso, si vedrà apparire l'applicazione *Test Raspberry Pi*, come illustrato nella Figura 3.46a. L'icona dell'applicazione è un rettangolo barrato e se si vuole far apparire un'icona personalizzata, basta modificare l'immagine `Square44x44Logo.scale-200.png` che si trova nella cartella `Assets`.

Nel caso si volesse distribuire l'applicazione nello Store Microsoft, si può cambiare l'icona modificando l'immagine `StoreLogo.png`. Ovviamente, in questo caso bisogna possedere un account di sviluppatore e, nel caso si volesse vendere o regalare l'applicazione, saranno necessarie le autorizzazioni di Microsoft. Tutte le informazioni a riguardo sono disponibili all'indirizzo `https://dev.windows.com/it-it/publish`.

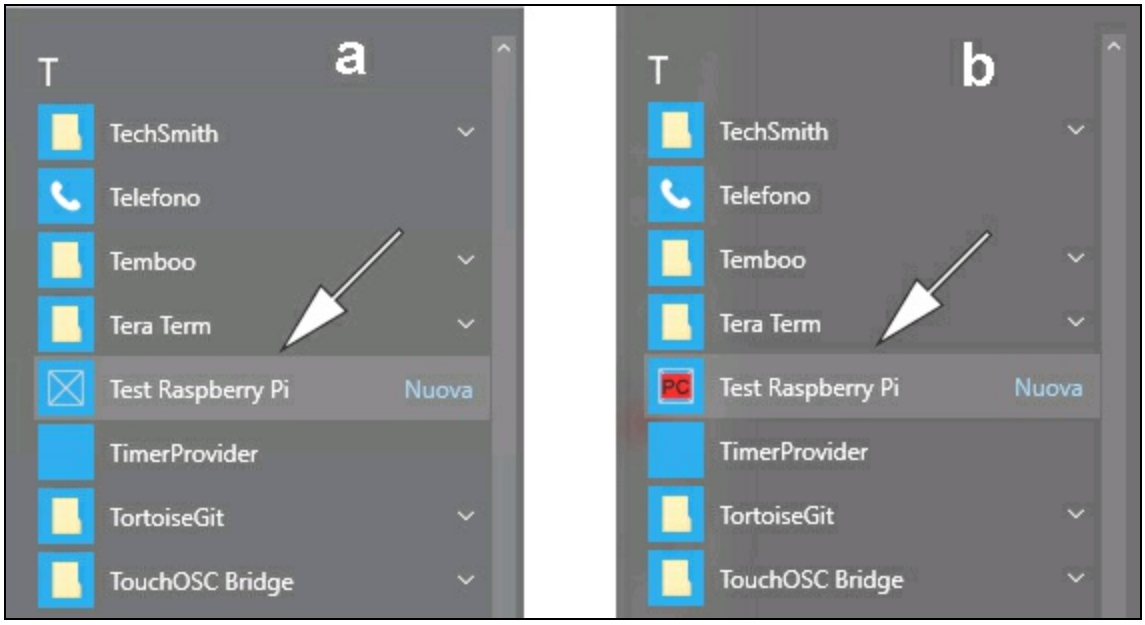


Figura 3.46 Il menu Start con l'applicazione Test Raspberry Pi (a). L'icona modificata dell'applicazione (b).

Modificare l'interfaccia grafica

Come già detto, la caratteristica più interessante di C# è l'implementazione al suo interno del linguaggio XAML (*eXtensible*

Application Markup Language).

Per far apparire l'interfaccia grafica dell'applicazione, fare doppio clic sul file `MainPage.xaml` nel pannello *Esplora soluzioni*. La finestra di progettazione impiega un po' di tempo prima di aprirsi, perché il motore grafico mette in moto parecchie risorse del sistema, anche della scheda video.

Nel caso di un'applicazione vuota apparirà un rettangolo bianco simile a quello rappresentato nella Figura 3.47. In caso di errore di apertura della finestra di progettazione, si può provare a ricaricarla. Si può aprire la finestra di progettazione anche dal menu contestuale del mouse, facendo clic destro sul nome del file.

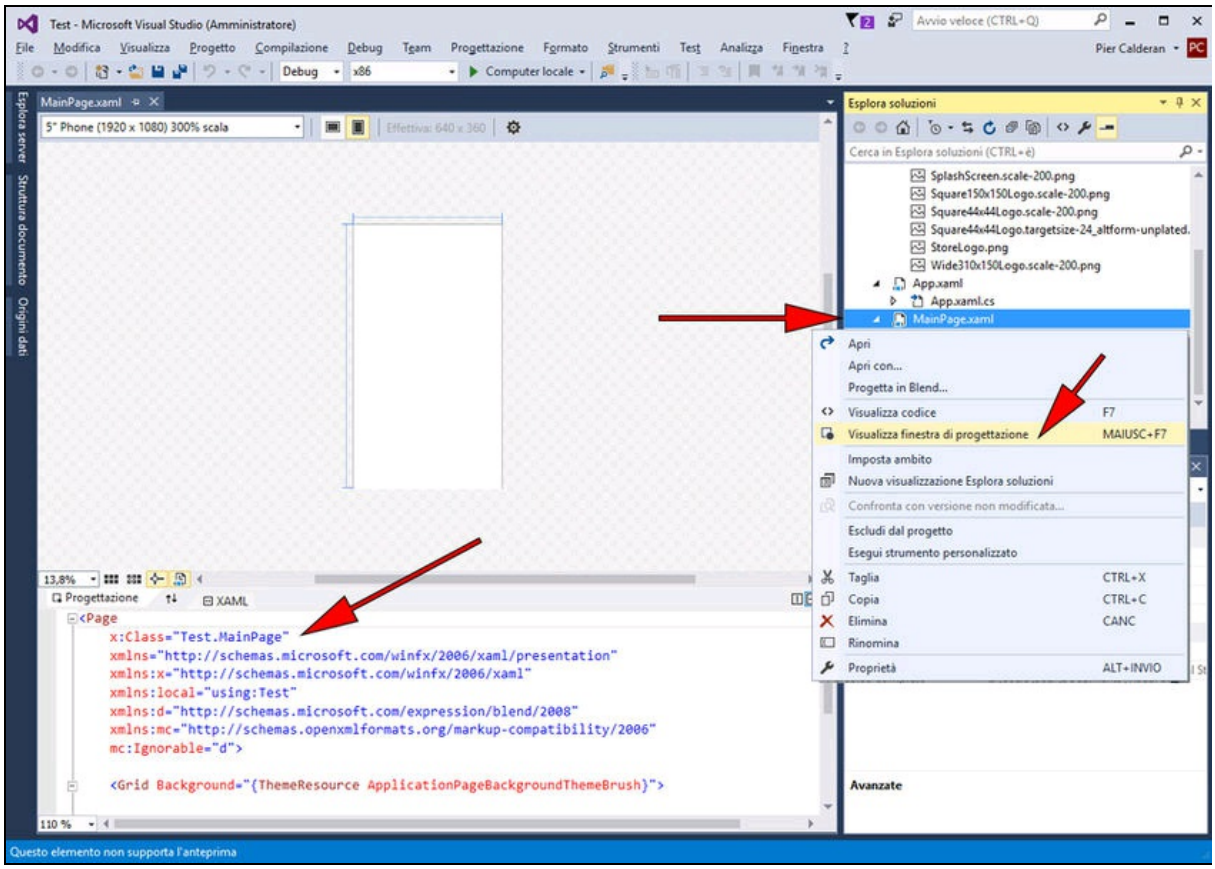


Figura 3.47 La finestra di progettazione MainPage.xaml.

Il codice XAML di default che appare sotto la finestra di progettazione è il seguente:

```
<Page
 x:Class="Test.MainPage"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:local="using:Test"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 mc:Ignorable="d">
 <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
 </Grid>
    </Page>
```

Per iniziare a modificare l'interfaccia utente o UI (*User Interface*) bisogna far apparire la *Casella degli strumenti* dal menu *Visualizza*.

Strumenti della finestra progettazione

Nella parte in basso a sinistra della finestra di progettazione ci sono alcune icone che servono ad adattare il rettangolo bianco dell'interfaccia allo schermo e a ottimizzare l'area di lavoro (Figura 3.48a).

- Nella prima casella si imposta lo zoom, che facilita il posizionamento degli oggetti. Lo zoom va da *Adatta tutto*, *Adatta selezione* fino a un ingrandimento dell'800%.

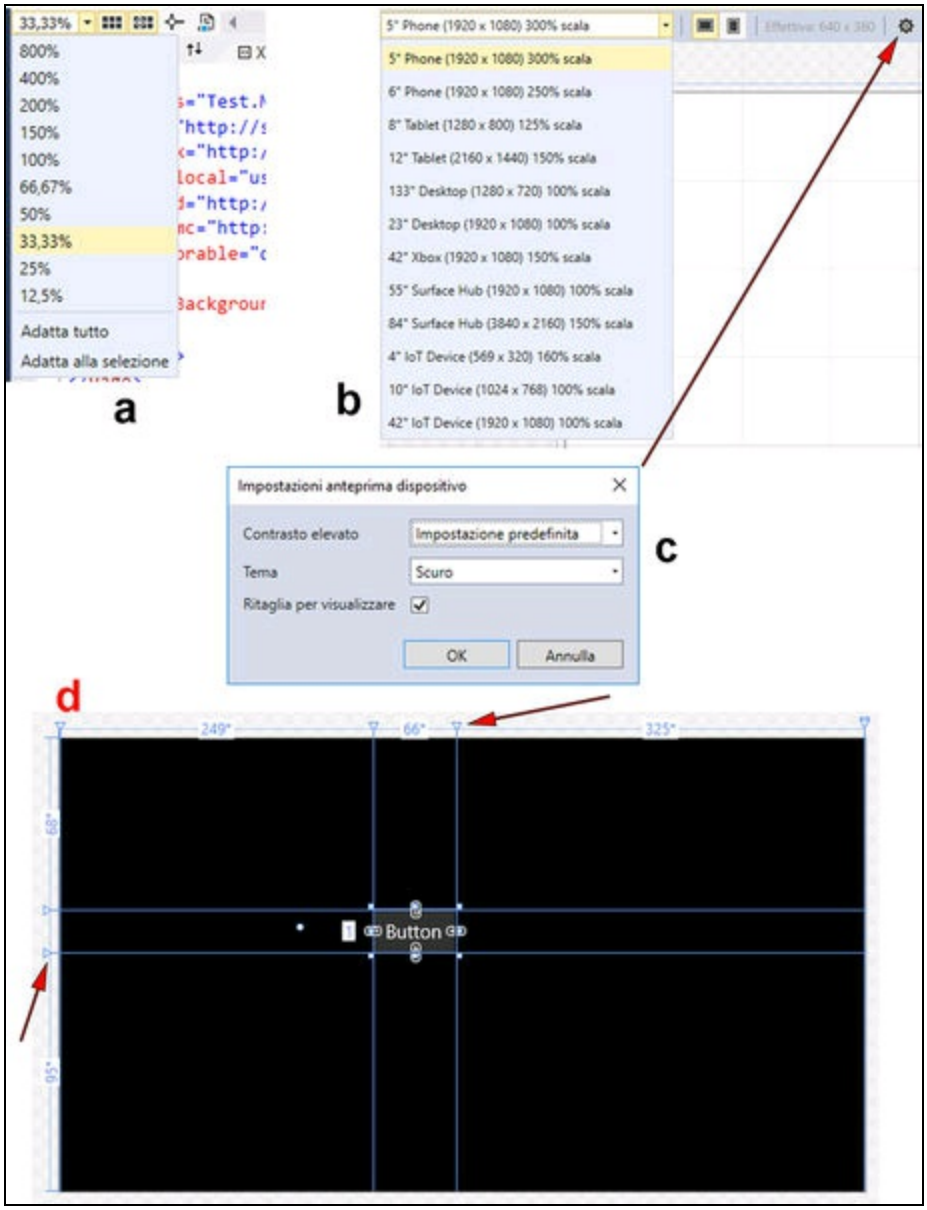


Figura 3.48 Finestra di progettazione: icone di controllo in basso (a), in alto (b) e finestra Impostazioni anteprima del dispositivo (c). Le guide verticali e orizzontali (d).

- Seguono nell’ordine le seguenti icone:
  - *Mostra/Nascondi la griglia*;
  - *Attiva/Disattiva l’allineamento alla griglia*;
  - *Attiva/Disattiva l’allineamento alle guide*;
  - *Abilita/Disabilita il codice del progetto*.

In alto nella finestra di progettazione ci sono altri controlli e icone (Figura 3.48b).

- Nella prima casella si imposta la dimensione dell'interfaccia relativamente alla risoluzione dello schermo del dispositivo. Per esempio: 5" Phone (1920 × 1080) 300% scala, 133" Desktop (1280 × 720) 100% scala, 10" IoT Device (1024 × 768) 100% scala e così via.
- L'icona subito a destra imposta l'interfaccia per l'orientamento orizzontale e quella successiva l'orientamento verticale.
- L'ultima icona di un ingranaggio apre una finestra di dialogo in cui impostare l'anteprima del dispositivo (Figura 3.48c): *Contrasto* (*Contrasto elevato*, *Nero*, *Bianco*, *Contrasto elevato 1*, *Contrasto elevato 2*), *Tema* (*Chiaro*, *Scuro*, *Contrasto elevato*) e l'opzione *Ritaglia per visualizzare*. Si fa notare che queste regolazioni sono valide per la finestra di progettazione e non per l'interfaccia grafica.

Per il cambiare il colore di fondo dell'applicazione, per esempio, su *Nero*, è necessario modificare il codice XAML o le proprietà dello sfondo.

Una volta selezionato il rettangolo dell'interfaccia, dal pannello *Proprietà* > *Aspetto*, selezionare il menu *RequestedTheme* e impostarlo su *Light* (bianco) o *Dark* (nero). Nel codice XAML verrà aggiunta la modifica nella riga seguente:

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
RequestedTheme="Dark">
```

Per inserire le guide in modo da facilitare l'allineamento degli oggetti, basta fare clic con il mouse nella parte superiore e sulla parte sinistra del rettangolo di lavoro, come indicato nella Figura 3.48d.

Lanciare l’applicazione remota senza Visual Studio

Volendo avviare un’applicazione già distribuita nel computer remoto senza utilizzare Visual Studio, basta aprire il pannello *Dashboard* e aprire l’home page del dispositivo con *Apri in Device Portal* come abbiamo già visto in precedenza. Un metodo ancora più rapido è digitare direttamente l’indirizzo IP nella barra degli indirizzi del browser per aprire la home page del dispositivo, per esempio, `192.168.1.xxx:8080`. 8080 è la porta HHTP secondaria che viene usata per evitare che il browser cerchi un indirizzo in Rete. Se si salva l’indirizzo nei segnalibri e si memorizza la password, si potrà accedere sempre in questo modo alla home page del dispositivo.

Dalla home page del dispositivo selezionare dal menu laterale la voce *Apps* e quindi *Apps Manager*. Si aprirà la pagina *Apps* (Figura 3.49a). Qui vengono elencate tutte le applicazioni residenti e quelle distribuite da Visual Studio.

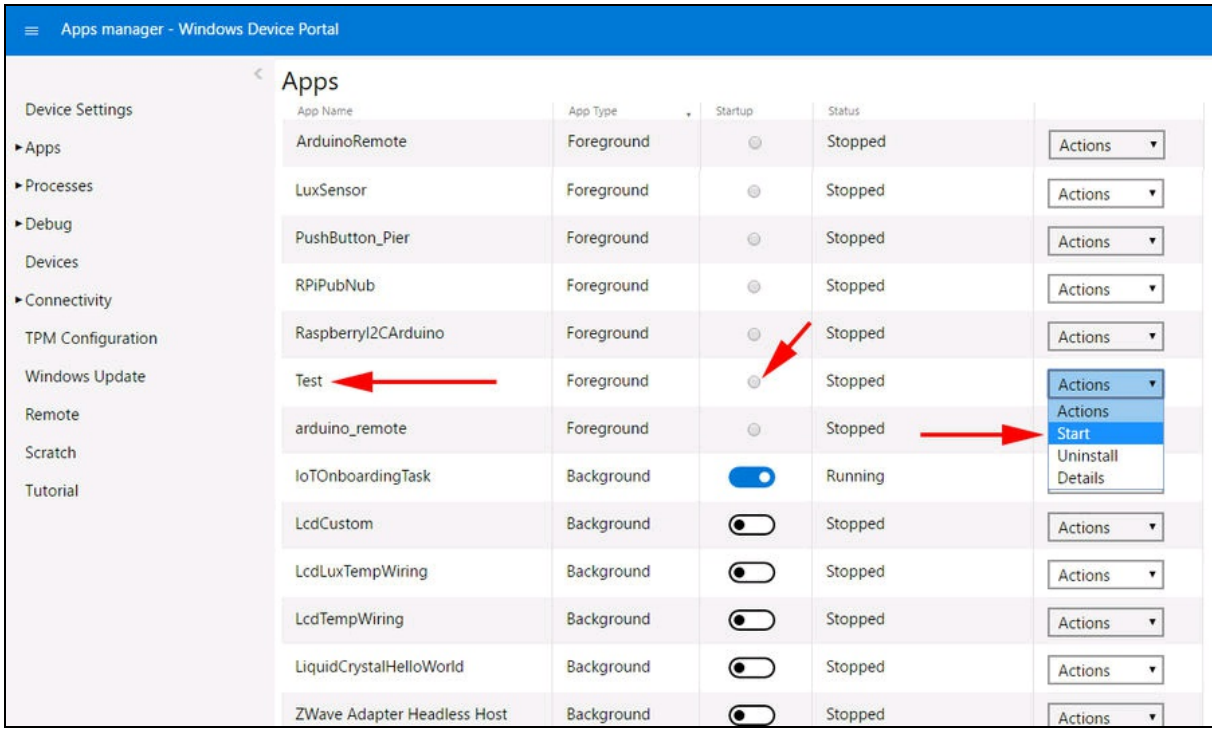


Figura 3.49 La pagina Apps della home page del dispositivo.

Selezionare l'applicazione distribuita, per esempio *Test* (come indicato dalla freccia a sinistra) e avviarla con *Start* dal menu a discesa *Actions* (come indicato dalla freccia a destra). L'avvio sarà estremamente veloce. Dallo stesso menu si può fermare l'applicazione con la voce *Stop*. Anche l'arresto sarà estremamente veloce.

Con la voce *Unistall* si può disinstallare l'applicazione selezionata, mentre la voce *Details* permette di visualizzare i dettagli dell'applicazione.

Nel caso non apparisse nel menu il nome dell'applicazione desiderata o se si volesse cambiarlo, è possibile assegnare il nome da visualizzare nel suddetto file `Package.appxmanifest`. Facendo doppio clic su questo file nel pannello *Esplora soluzioni* si aprirà una finestra in cui immettere un nome senza spazi nella casella *Nome visualizzato*. Questo sarà il nome visualizzato nel menu delle applicazioni quando vengono distribuite.

Si può far diventare l’applicazione selezionata quella di avvio. Nella colonna *Startup* (freccia al centro) selezionare quella che si vuole avviare al boot del sistema.

La console di output

Un’ultima cosa prima di abbandonare l’ambiente di lavoro C#. Per effettuare il debug dell’applicazione in esecuzione è molto utile stampare i dati restituiti dalle funzioni nella console di output. Supponendo di voler visualizzare il valore di una variabile `dato`, si può attivare la stampa nella console di output con la seguente istruzione:

```
Debug.WriteLine(dato);
```

Siccome lo strumento di diagnostica non è implementato di default, bisogna aggiungere all’inizio del listato la seguente direttiva:

```
using System.Diagnostics;
```

Inserendo nei punti cruciali l’istruzione `Debug.WriteLine` si può eseguire un debug delle funzioni in tempo reale.

Piattaforme IoT e Cloud

Cloud computing

La metafora del *cloud computing*, cioè della “nuvola di computer”, è ispirata all’idea di un insieme di tecnologie IT che coinvolgono processi di archiviazione, di elaborazione dati e di applicazioni condivise nella rete globale. Viene usato moltissimo a livello globale, ma una cosa è certa: il cloud computing è in progressiva espansione e svolgerà sempre di più un ruolo decisivo nello sviluppo delle reti a livello aziendale.

Per far comprendere meglio il concetto di cloud computing, il NIST (*National Institute of Standards and Technology*), un istituto governativo americano che definisce gli standard e le tecnologie, nel 2011 ha pubblicato un documento con la definizione ufficiale di *cloud computing*. Il documento è scaricabile liberamente all’indirizzo http://www.nist.gov/itl/cloud.

Servizi del cloud computing

Dal documento del NIST si possono leggere le definizioni dei tipi di servizi del cloud computing.

- *SaaS* (*Software as a Service*): software erogati come servizi. Per esempio, modelli di SaaS sono Microsoft Office 365, che consente di usare online software come Word o Excel, e Gmail, per i servizi di posta elettronica.

- *PaaS* (*Platform as a Service*): piattaforme come servizio, ovvero piattaforme per lo sviluppo e la distribuzione di applicazioni web. Per esempio, modelli di PaaS sono Microsoft Azure, con cui è possibile distribuire applicazioni scritte con la piattaforma .Net, e Google Cloud Platform, che permette di scrivere applicazioni in alcuni linguaggi di programmazione popolari, come Python, Java, PHP e Go, e associare le applicazioni con Compute Engine per integrare altre tecnologie, come Node.js, C, Scala, Hadoop, MongoDB, Redis e altre.
- *IaaS* (*Infrastructure as a Service*): infrastrutture come servizi. La funzionalità fornita all'utente è quella di elaborazione, storage e altre risorse di calcolo fondamentali, in cui l'utente è in grado di distribuire ed eseguire software arbitrario, come sistemi operativi e applicazioni. Il consumatore non può gestire o controllare l'infrastruttura Cloud, ma ha il controllo sui sistemi operativi, lo storage e sulle applicazioni distribuite. Il vantaggio di questa soluzione è la capacità di controllo consentita all'utente. Una volta scelto l'hardware più opportuno, è possibile sfruttare da remoto e da computer poco potenti o da cellulari, la potenza presente su altri computer ed eseguire elaborazioni complesse che possono richiedere molto tempo.

I modelli di distribuzione del cloud computing

Il NIST dà anche precise indicazioni riguardo a come deve essere organizzato un modello di cloud computing. Le possibilità sono quattro.

- *Private Cloud*: i servizi di cloud computing vengono erogati da aziende. L'infrastruttura può essere gestita da un provider o da terze parti o dalla combinazione di questi soggetti.

- *Community Cloud*: i servizi di cloud computing vengono erogati per l’uso esclusivo da una specifica comunità di consumatori da organizzazioni che hanno condiviso gli scopi. L’infrastruttura può essere gestita da una o più organizzazioni della comunità, da terze parti o da una combinazione di questi soggetti.
- *Public Cloud*: i servizi di cloud computing vengono erogati per l’uso gratuito pubblico. L’infrastruttura può essere gestita da un organizzazione commerciale, accademica o governativa o da una combinazione di questi soggetti.
- *Hybrid Cloud*: l’infrastruttura Cloud è una composizione di due o più infrastrutture cloud distinte (private, comunità o pubbliche) che rimangono soggetti unici, ma sono legati dalla tecnologia standardizzata o proprietaria che permette la portabilità dei dati e delle applicazioni.

Con queste definizioni il concetto di cloud computing è sicuramente più chiaro e ci fa capire che si tratta di un’architettura informatica che coinvolge tutti i settori dell’Information Technology.

La Figura 4.1 illustra in modo schematico l’interazione esistente fra software come servizi (SaaS), piattaforme come servizi (PaaS), infrastrutture come servizi (IaaS) e il mondo esterno costituito dai client. I client collegati alla nuvola sono dispositivi IoT, computer desktop, computer laptop, server, smartphone e tablet. In pratica, tutto.

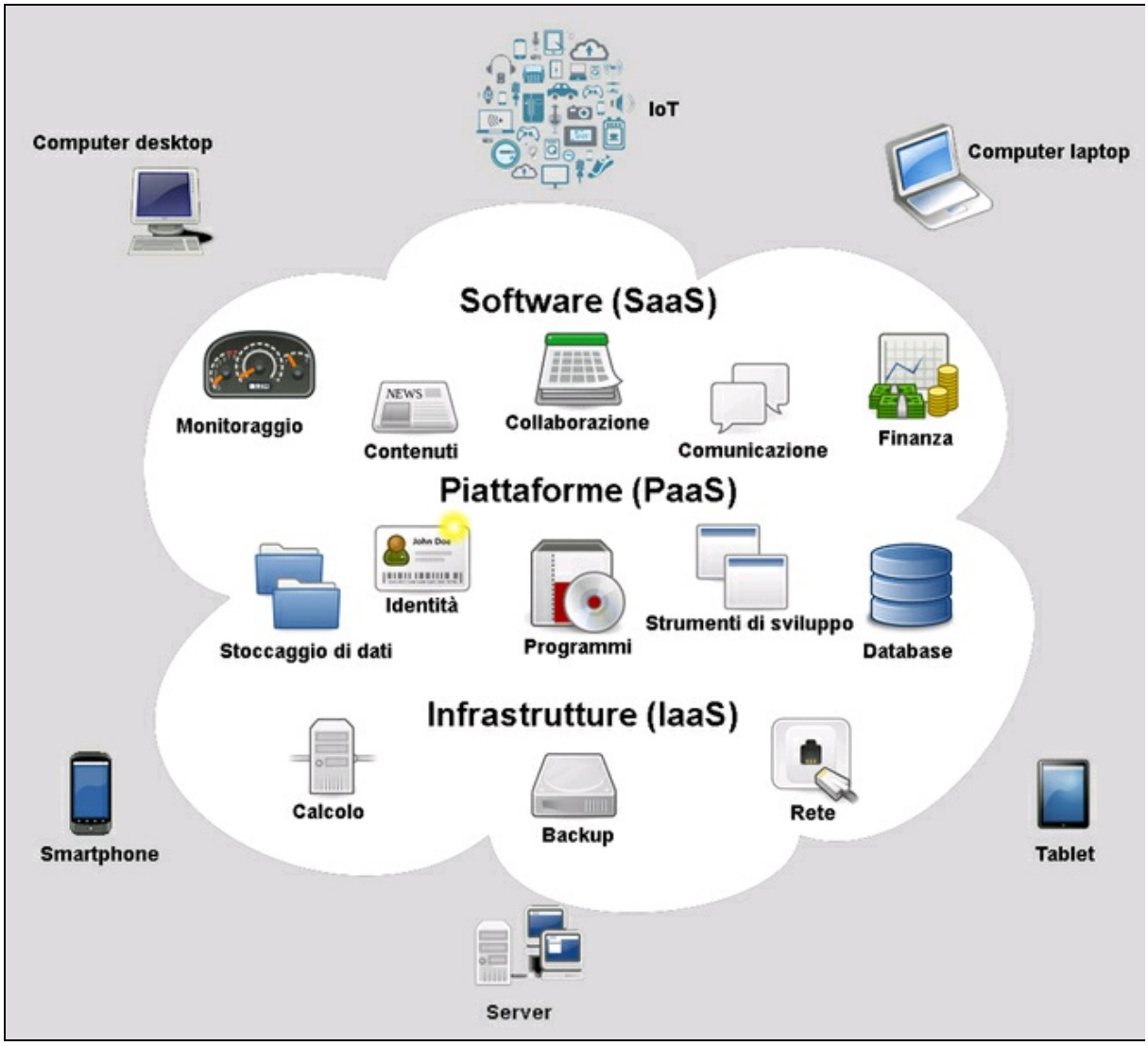


Figura 4.1 Cloud computing: interazione fra software (SaaS), piattaforme (PaaS) e infrastrutture (IaaS) con il mondo esterno costituito dai client: IoT, computer desktop, dispositivi remoti (Raspberry Pi, Arduino, ecc.), computer laptop, server, smartphone e tablet.

Vendor di soluzioni Cloud storage e VPS

Una delle definizioni di cloud computing si basa sul concetto di *on-demand self-service*, che consente all'utente di scegliere il server di storage e di rete in modo automatico, senza richiedere l'interazione umana del fornitore dei servizi. Negli ultimi tempi sono nati molti

servizi per il cosiddetto cloud storage e VPS (*Virtual Private Server*) ovvero server privato virtuale.

È il modello di cloud computing più semplice e bastano pochi minuti per creare da soli un server privato funzionante, senza richiedere l'intervento del provider. Una volta attivato il server, si può scegliere di utilizzare lo spazio come cloud storage e/o per programmare applicazioni web o creare un proprio sito.

È ormai prassi comune offrire pannelli di configurazione per server privati scalabili a piacere. Un esempio fra i tanti è illustrato nella Figura 4.2. La sezione Cloud server del sito *1&1* (`https://www.1and1.it`) propone un pannello di configurazione in cui selezionare le opzioni della tariffazione (mensile o oraria), numero di CPU, quantità di RAM, dimensione del disco, sistema operativo (Linux o Windows) e, opzionalmente, il nome del dominio da associare al server.

Questo significa che, invece di acquistare un dominio con uno spazio web fisso e prestazioni standard, con una soluzione di cloud storage è possibile personalizzare il proprio spazio e decidere quando e se usarlo, pagando in base alla scelta delle prestazioni desiderate e del tempo di utilizzo.

Ma come è possibile personalizzare un server scegliendo le prestazioni di CPU, RAM, disco e sistema operativo? Semplicemente grazie a sofisticati software di virtualizzazione che possono costruire in pochi minuti macchine virtuali su misura per ogni esigenza. I più importanti software di virtualizzazione usati dai provider di cloud computing sono *VMware*, *Microsoft Virtual PC*, *Oracle VM VirtualBox* e *Xen*.

Figura 4.2 Home page Cloud Server di 1&1.

Vendor di servizi di cloud computing

Fra le molte aziende internazionali che offrono soluzioni complete di cloud computing ne citiamo quattro. Fra parentesi i tipi di servizi offerti:

- Amazon (IaaS);
- Google (PaaS);
- Microsoft (IaaS, PaaS e SaaS);
- VMWare (IaaS).

IoT, Internet delle Cose

Rimanendo in tema di cloud computing, il mondo IoT potrebbe esser definito un modello di tipo *TaaS*, ovvero *Thing as a Service*. Un termine non ufficiale, ma che rende l'idea delle "cose come servizio".

Internet of Things, ovvero Internet delle cose, è un concetto nato per rappresentare una serie di dispositivi elettronici che possono connettersi alla Rete globale o alle reti domestiche, condividendo i dati di rilevamento provenienti da sensori, data center o server remoti.

L'Internet delle cose è un mondo di dispositivi che monitorano l'ambiente, accedono a Internet e alla rete locale (LAN o WLAN), memorizzano o recuperano i dati e interagiscono con gli utenti.

Tutti i dispositivi IoT sfruttano la potenza di mini-computer o microcontrollori collegati in Wi-Fi e sono dotati di sensori di ogni tipo. La loro proliferazione ha indotto all'uso del protocollo IPv6 a 128 bit per sopperire al problema della mancanza di indirizzi IPv4 e le stime degli osservatori del mercato su Internet dicono che presto saremo invasi da dispositivi "intelligenti" in grado di coprire qualsiasi ambito di utilizzo. Eccone alcuni esempi (in ordine alfabetico).

- Controllo del traffico.
- Controllo di spazi e degli accessi pubblici.
- Controllo domotico.
- Gestione degli inventari.
- Gestione dei rifiuti.
- Gestione dell'illuminazione pubblica.
- Gestione delle serre.
- Gestione delle stalle.
- Gestione intelligenti parcheggi.
- Mappatura del rumore del traffico.

- Monitoraggio dell’energia.
- Monitoraggio dell’energia domestica.
- Monitoraggio della qualità dell’aria.
- Monitoraggio della temperatura.
- Monitoraggio delle presenze.
- Monitoraggio di posizione e avvisi.
- Monitoraggio e controllo dell’inquinamento dell’aria.
- Monitoraggio presenza sostanze pericolose.
- Monitoraggio qualità dell’acqua (fiumi o mare).
- Monitoraggio sanitario a distanza.
- Rilevamento di incidenti.
- Sistemi di allarme (catastrofi naturali).
- Sistemi di allarme pubblico e privato.

NOTA

Talvolta i dispositivi IoT vengono associati alla tecnologia M2M, acronimo di *Machine-to-Machine*. Il collegamento di dispositivi M2M consente di gestire i dati di input e output dell’hardware per funzioni di monitoraggio e controllo remoto tramite applicazioni web.

Amazon

L’offerta di Amazon è di tipo IaaS. Si chiama *Amazon Web Services*, abbreviata in *AWS*. È disponibile all’indirizzo `http://aws.amazon.com/it` (Figura 4.3).

Si tratta di una serie di servizi che permettono di costruire un’infrastruttura gratuita per 12 mesi. Quando scadono i 12 mesi di utilizzo gratuito o se l’utilizzo delle applicazioni supera quello previsto dal piano gratuito, se l’utente desidera mantenere attivo il servizio pagherà solo in base alle tariffe standard di utilizzo effettivo. Alcune offerte incluse nel piano gratuito non scadono automaticamente al

termine dei 12 mesi del piano gratuito, bensì rimangono disponibili sia per i nuovi clienti sia per i vecchi clienti AWS, in modo indefinito.

Attualmente Amazon mette a disposizione decine di servizi di cloud computing, fra cui *AWS IoT*. Il kit Device SDK di AWS IoT consente di connettere in modo semplice e veloce i dispositivi hardware ad AWS IoT. Il kit include librerie open source, una guida per sviluppatori corredata da esempi e una guida per il porting, indispensabili per creare prodotti e soluzioni IoT innovative su qualsiasi piattaforma hardware. Ecco un elenco di SDK per AWS IoT:

- C per sistemi *embedded*;
- JavaScript;
- Arduino Yún;
- Java;
- Python;
- iOS;
- Android.

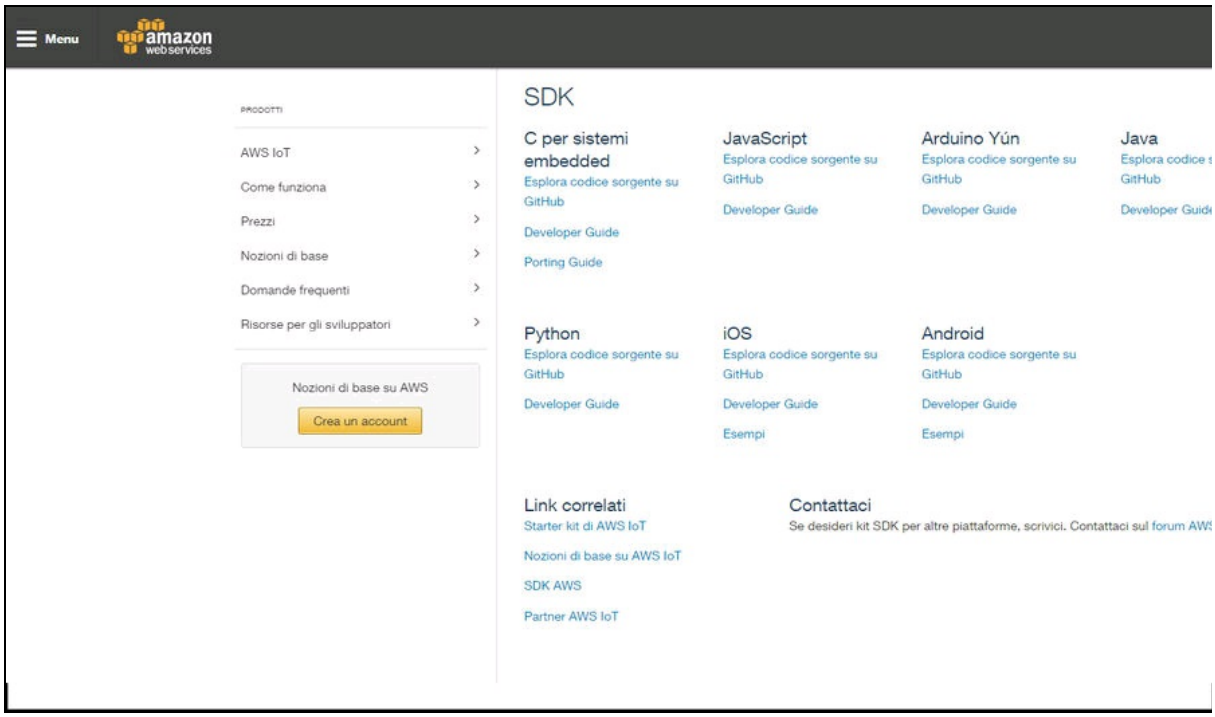


Figura 4.3 La home page di AWS IoT.

Google

Google offre servizi di piattaforma (PaaS) sotto forma di *Google Cloud Platform* (Figura 4.4). I servizi sono disponibili all'indirizzo `https://cloud.google.com`.

L'uso di Google Cloud Platform è gratuito per 60 giorni con 300 dollari di credito da spendere su tutti i prodotti della piattaforma. Durante la prova gratuita ci sono alcune limitazioni del prodotto. Per esempio, Compute Engine è limitato a otto core simultanei. Per la prova gratuita non si paga nulla, ma quando si conclude, l'account viene messo in attesa; con la possibilità di passare a un account a pagamento. È necessario pagare entro 30 giorni dalla data di conclusione della prova gratuita.

È impressionante il numero di applicazioni di Google Platform a cui si può accedere con il proprio account. La piattaforma è indubbiamente rivolta agli sviluppatori, ma offre servizi di cloud computing anche per chi vuole solo un servizio di networking e di storage per i propri documenti.

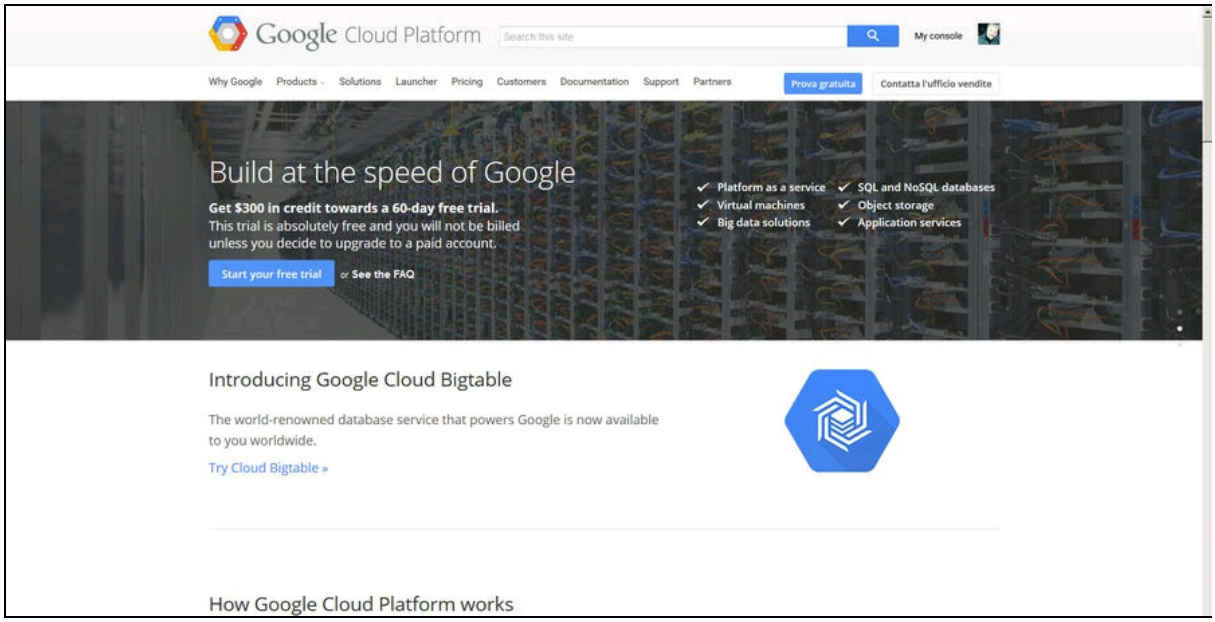


Figura 4.4 La home page di Google Cloud Platform.

VMWare

VMWare è diventata famosa per i suoi software di virtualizzazione usati universalmente da un gran numero di provider di servizi VPS. Da qualche tempo VMWare ha aggiunto servizi di cloud computing che sono disponibili all'indirizzo `https://www.vmware.com/cloud-services` (Figura 4.5).

I servizi offerti sono di cloud privato e cloud ibrido.

- *vSphere*: l'architettura è un cloud privato per ottenere applicazioni e servizi a elevata disponibilità con un data center standardizzato e consolidato.
- *vCloud*: è un servizio di cloud ibrido, che consente di estendere in modo semplice e sicuro il data center e le applicazioni al cloud.

Dal sito si possono valutare online prodotti come vSphere e vCloud tramite una registrazione gratuita.

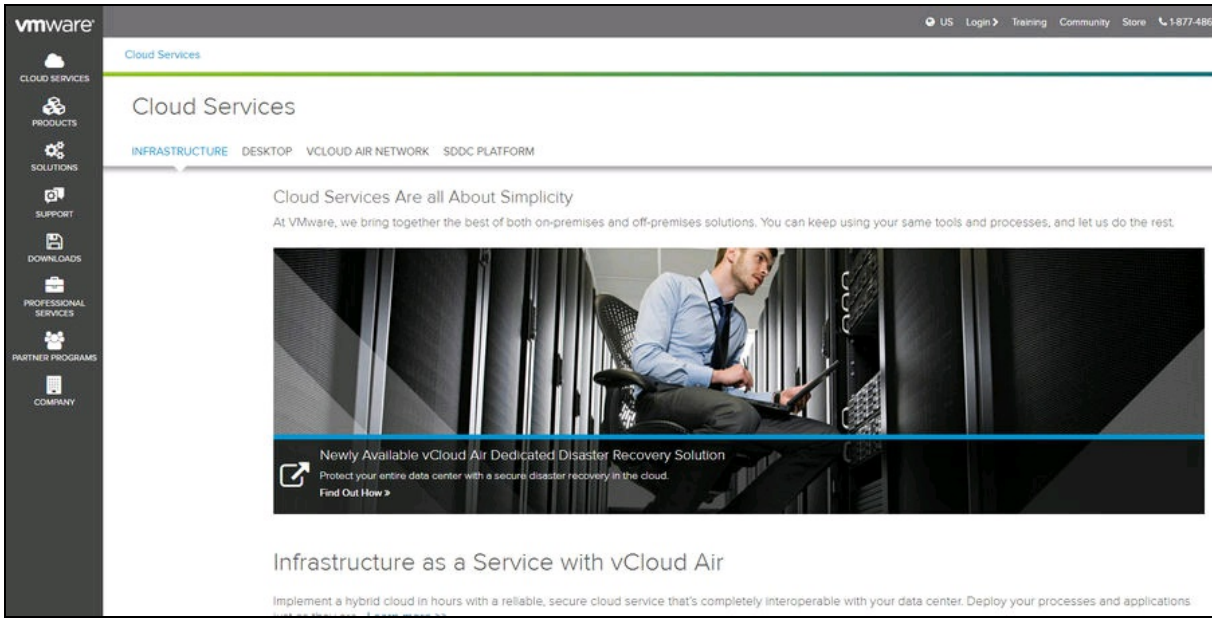


Figura 4.5 La home page di VMWare Cloud services.

Microsoft Azure

Il cloud di Microsoft è rivolto alle imprese, ma anche ai privati e agli sviluppatori IT che vogliono sfruttare la sua enorme capacità di data streaming, di archiviazione, di macchine virtuali altro ancora. Con Azure si possono creare e gestire siti web, app aziendali per dispositivi mobili, data storage e data streaming per dispositivi IoT (per esempio, Raspberry Pi).

La piattaforma Azure è sia IAAS (*Infrastructure as a Service*), sia PAAS (*Platform as a Service*), sia SAAS (*Storage as a Service*) ed è diventata in pochi anni un leader di mercato. Tutte le informazioni e i riferimenti sono disponibili nel sito ufficiale in italiano: `https://azure.microsoft.com/it-it` (Figura 4.6).

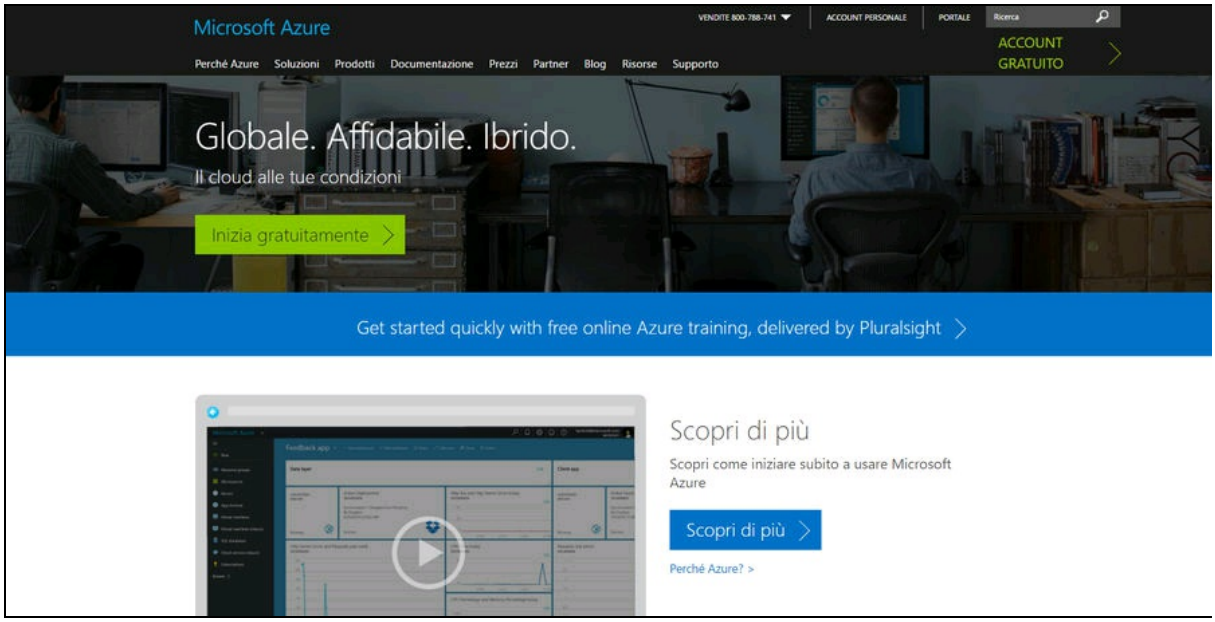


Figura 4.6 La home page di Microsoft Azure.

Versione di prova gratuita

C’è la possibilità di provare la piattaforma a costo zero per un mese. Con l’iscrizione tramite autenticazione della propria carta di credito, si può accedere a tutti i servizi senza limitazioni. L’offerta prevede un credito di 170 euro per l’accesso completo senza addebito, se dopo un mese non si completa il contratto di sottoscrizione. I dettagli per la sottoscrizione sono disponibili al momento dell’accettazione e conviene leggere attentamente tutte le condizioni.

Per esempio, con la versione di valutazione gratuita, si ricevono 170 euro in crediti Azure ed è possibile usare i crediti per qualsiasi servizio, in base alle esigenze, tra cui macchine virtuali, siti web, servizi cloud, servizi mobili, archiviazione, database, servizi multimediali e molto altro.

È disponibile un “calcolatore prezzi” per stimare la quantità di servizi di cui si può usufruire con 170 euro in crediti. Ecco alcuni scenari possibili che rientrano nella cifra:

- eseguire due istanze di macchine virtuali piccole per tutto il mese;
- archiviare 800 GB di dati nel servizio di archiviazione;
- sviluppare e testare un’applicazione web mediante servizi cloud, con tre ruoli web e due ruoli di lavoro su istanze medie, per 10 ore al giorno e 5 giorni alla settimana;
- eseguire due database SQL S2 per un mese intero.

Per i nostri scopi, abbiamo approfittato dell’account gratuito con i crediti di Azure per eseguire un’applicazione web e per effettuare l’archiviazione di dati. Per i dettagli si veda il progetto Photostorage.

IoT nel cloud

Nonostante l'Internet delle cose sia un concetto relativamente nuovo, ci sono già molte piattaforme open source e a pagamento che consentono la gestione remota su web dei dispositivi IoT, con la visualizzazione dei dati dei sensori in tempo reale e la condivisione dei dati nel Cloud. Le piattaforme normalmente offrono profili personali per un uso gratuito, ma esistono anche piattaforme professionali a pagamento e con hardware dedicato. Per citarne solo alcune, ecco una lista delle piattaforme Cloud per dispositivi IoT più diffuse.

- *Temboo*: è una piattaforma Cloud con generazione di codice con oltre 2000 processi API per qualsiasi ambiente di sviluppo. Sito ufficiale: `https://temboo.com`.
- *ThingSpeak* (si veda il paragrafo dedicato, qui sotto).
- *PubNub*: uno dei più diffusi servizi web per dispositivi IoT (`https://www.pubnub.com`). PubNub mette a disposizione una rete per il flusso di dati in tempo reale per sviluppatori. PubNub offre molti servizi e la sezione IoT è davvero molto ricca di esempi e template per oltre 70 piattaforme, Raspberry Pi e Arduino compresi.
- PubNub offre un uso gratuito del suo cloud per la connessione di 100 dispositivi al giorno, un milione di messaggi e 30 giorni di prova per gli add-on. Per le altre proposte commerciali si veda la sezione Pricing del sito.
- *Nimbits*: piattaforma per la connessione di persone, sensori e software per il Cloud. Il server Nimbits è disponibile per Linux. È in grado di registrare ed elaborare dati di tempo e dati geografici e di eseguire formule definite in base a tali informazioni. Le formule possono essere calcoli, statistiche, avvisi email, messaggi XMPP,

notifiche push e molto altro ancora. Sito ufficiale: http://www.nimbits.com.

- *Cosm*: è un insieme di applicazioni e protocolli aperti, progettati per consentire ai computer di tutto il mondo di lavorare insieme per il calcolo distribuito, GRID, SensorNet o applicazioni cloud. Le applicazioni e i progetti possono essere scientifici, educativi o commerciali. Ora è confluito nella piattaforma Xively.
- *Xively*: (precedentemente noto come Cosm e Pachube) è costruito su piattaforma cloud Gravity di LogMeIn, che gestisce oltre 255 milioni di dispositivi, utenti e clienti in 7 data center in tutto il mondo. Sito ufficiale: https://www.xively.com.
- *Carriots*: Carriots è un modello PaaS progettato per IoT e M2M. Raccoglie e memorizza qualsiasi tipo di dati dai dispositivi e consente di creare applicazioni con un potente motore di SDK. Per l'invio di dati al cloud di Carriots si possono programmare applicazioni con messaggi in cURL, hURL, Poster, MQTT oppure tramite l'hardware Arduino Yun o Arduino UNO dotato di Ethernet Shield. Sito ufficiale: https://www.carriots.com.
- *IFFT*: IFTTT (*If This Then That*, cioè "se fa questo allora fa quello) è un servizio web che permette di creare semplici sequenze di comandi attivati da altri servizi web come Gmail, Facebook, Instagram e Pinterest. Sito ufficiale: https://ifttt.com.
- *Twilio*: Twilio è una piattaforma di comunicazione cloud modello PaaS che consente agli sviluppatori di programmare e ricevere chiamate telefoniche e inviare e ricevere messaggi di testo utilizzando le API di servizi web. Sito ufficiale: https://www.twilio.com.
- *Bluemix*: Bluemix by IBM permette di sviluppare, eseguire, rilasciare e gestire applicazioni nell'ambiente cloud. Supporta vari linguaggi di programmazione (Java, Node.js, Go, PHP, Python,

Ruby on Rails) e offre dei servizi pronti all'uso per la gestione dei database, della reportistica, dell'IoT, delle applicazioni mobile e altro ancora. Sito ufficiale: https://www.ibm.com/cloud-computing/bluemix/it/internet-of-things.

ThingSpeak

Secondo i suoi sviluppatori, ThingSpeak è allo stesso tempo un'applicazione e un'interfaccia API open source per IoT, studiata per archiviare e recuperare dati da cose che utilizzano il protocollo HTTP tramite Internet o tramite una rete locale.

ThingSpeak consente di creare applicazioni di *sensor logging* (registrazione dati dei sensori), *location tracking* (tracciamento della posizione) e un *things social network* (rete sociale di cose) con l'aggiornamento del loro stato. ThingSpeak offre il supporto software di calcolo numerico Matlab da MathWorks.

La piattaforma ThingSpeak è stata recensita favorevolmente da siti specializzati per maker, come *Instructables* (http://www.instructables.com), *CodeProject* (http://www.codeproject.com) e *Channel 9* (http://channel9.msdn.com).

Sito ufficiale: https://thingspeak.com.

Creare un dispositivo IoT

Il titolo di questo paragrafo potrebbe spaventare, ma per far capire meglio come funziona il mondo IoT bastano 15 minuti, una scheda Arduino, uno shield Ethernet, un sensore di temperatura e un servizio Cloud.

L'obiettivo è quello di creare un semplice dispositivo domotico che misuri la temperatura ambientale e che si connetta in tempo reale a

Internet. I dati di temperatura possono essere visualizzati su un grafico da chiunque o elaborati da altri utenti.

Con piccole modifiche, lo stesso progetto potrebbe essere impiegato per decine di altri usi, come, per esempio, mandare allarmi a dispositivi Wi-Fi, inviare email o SMS oppure monitorare l'ambiente.

Circuito elettronico

Iniziamo dal collegamento della scheda Arduino UNO a uno shield Ethernet, che aggiunge la connettività di rete. Il sensore utilizzato è un comunissimo LM35. In alternativa, si può usare un qualsiasi sensore di temperatura attivo (per esempio DHT11 o DHT22) o anche un termistore passivo.

Una volta impilato lo shield Ethernet su Arduino UNO, effettuare i collegamenti del sensore di temperatura. La Figura 4.7a illustra lo shield Ethernet montato su Arduino UNO e il sensore LM35 con contenitore TO220 collegato direttamente all'alimentazione e all'ingresso A0 di Arduino. La Figura 4.7b illustra il collegamento fisico dei componenti e il progetto fatto con Fritzing con il contenitore TO92 dell'LM35 (stare attenti alla piedinatura diversa).

I collegamenti da effettuare sono i seguenti:

- piedino dell'alimentazione positiva del sensore LM35 al piedino 5 volt della scheda Arduino;
- piedino di uscita del sensore LM35 al piedino A0 (porta analogica) della scheda Arduino;
- piedino di massa del sensore LM35 al piedino GND della scheda Arduino.

A questo punto, basta collegare un cavo Ethernet al router di casa e la scheda Arduino a una presa USB del computer. Lo sketch da caricare viene spiegato nel prossimo paragrafo.

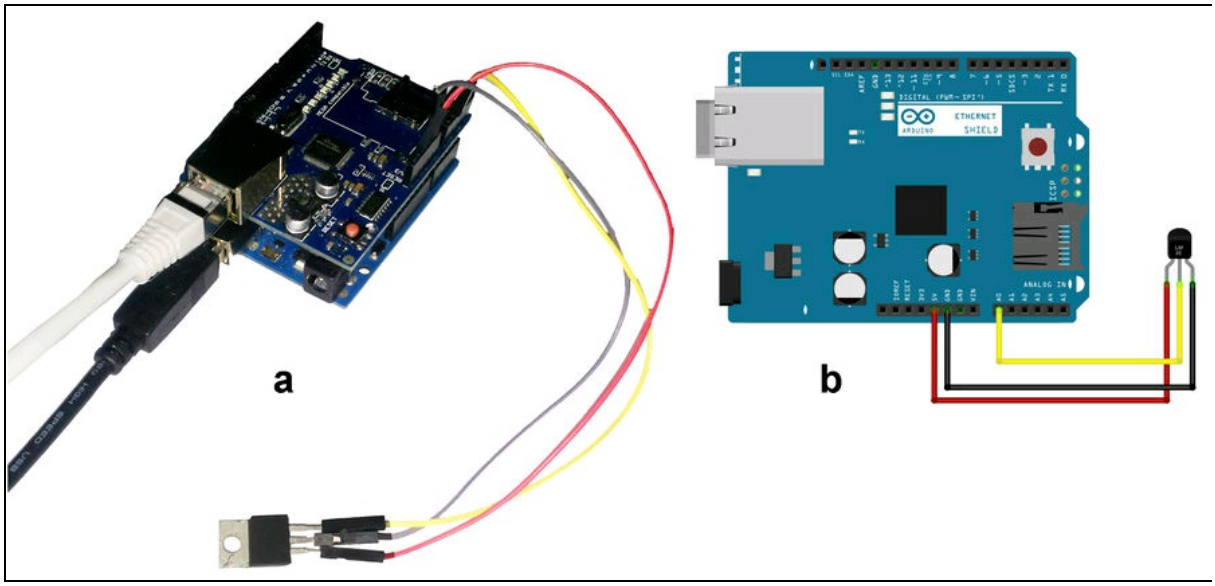


Figura 4.7 Il collegamento fisico (a). Il collegamento fatto con Fritzing (b).

Caricare lo sketch in Arduino UNO

Lo sketch originale è disponibile dal sito ThingSpeak. L'indirizzo per il download diretto del file originale è https://github.com/iobridge/ThingSpeak-Arduino-Examples/blob/master/Ethernet/Arduino_to_ThingSpeak.ino. Si consiglia però di scaricare lo sketch dalla pagina delle risorse del libro, perché lo sketch originale è stato modificato opportunamente per la lettura della temperatura in gradi Celsius e in gradi Fahrenheit e commentato in italiano. Basta solo seguire le poche istruzioni che seguono per apportare le opportune modifiche.

Facendo riferimento alla freccia di Figura 4.8, dopo la riga di codice relativa al DNS di ThingSpeak c'è una riga da modificare:

```
char thingSpeakAddress[] = "api.thingspeak.com"; // DNS di ThingSpeak
String writeAPIKey = "3WKARTBC5QDI09RW"; //chiave API di ThingSpeak
```

Il valore in grassetto è quello da modificare. Si tratta della chiave API che viene fornita dal sito ThingSpeak dall'applicazione creata online.

Sul prossimo paragrafo viene spiegato come recuperare questa chiave API.

Una volta modificato lo sketch, basta caricarlo nella scheda Arduino. Al termine del caricamento, tutto è pronto per mandare i dati all'app sul Web tramite lo shield Ethernet.

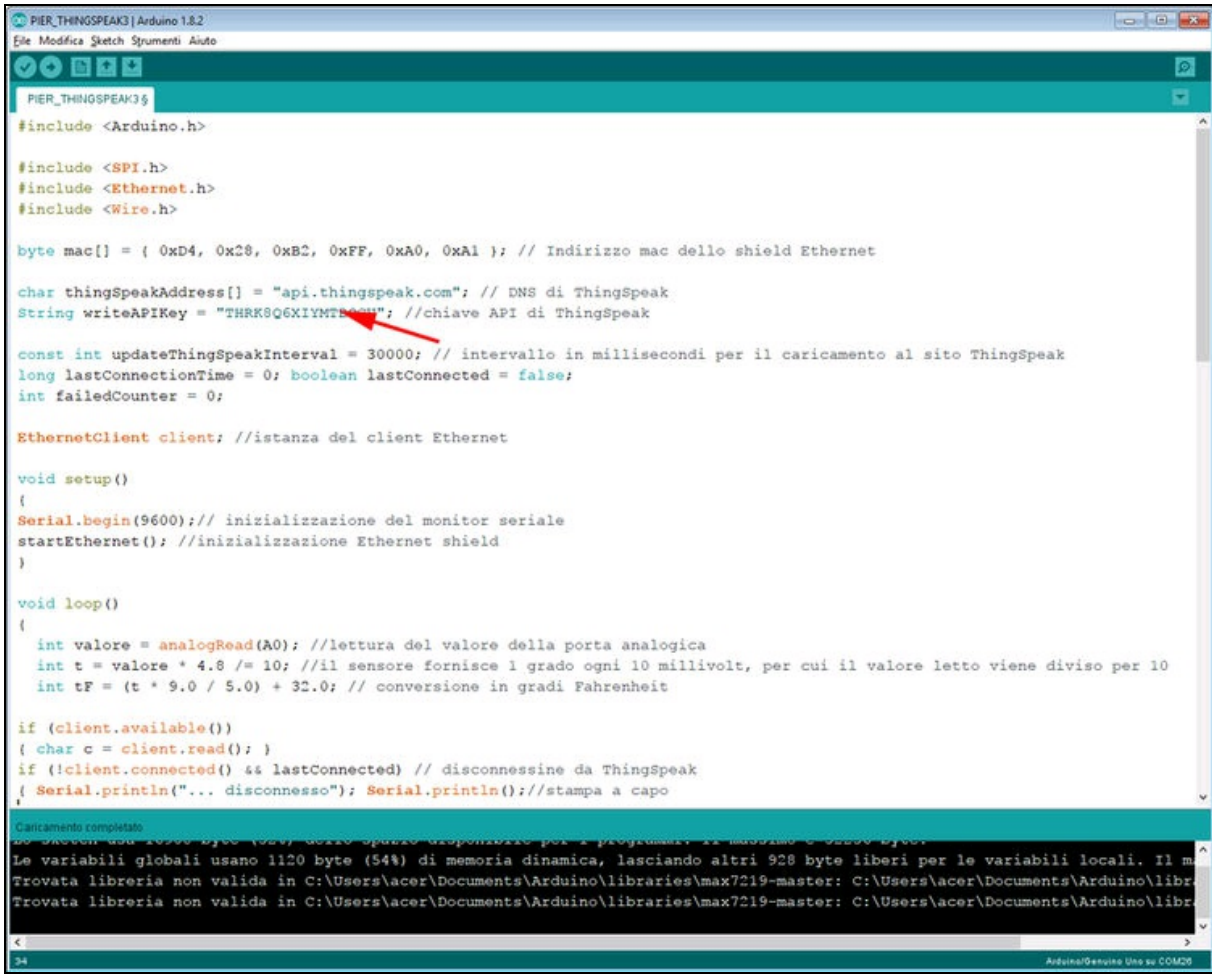


Figura 4.8 La schermata dello sketch ThingSpeak nell'IDE di Arduino.

Ma come funziona?

Senza entrare nel dettaglio, possiamo accennare a qualche riga dello sketch per spiegare come una scheda Ethernet possa comunicare i dati del sensore di temperatura.

Innanzitutto, la porta di ingresso analogico A0 riceve i dati del sensore di temperatura LM35. Il sensore viene alimentato direttamente dalla tensione di 5 volt di Arduino. Dato che il valore di riferimento della tensione è 5 volt, il segnale in uscita del sensore sarà una tensione variabile da 0 a 5 volt.

Il campionamento a 10 bit della porta analogica consente di ottenere una risoluzione di circa 4,8 mV (cioè 5 volt diviso 1024).

Sapendo che il sensore di temperatura fornisce 10 mV ogni grado centigrado, si dovrà dividere la tensione d'uscita del sensore per 10. La formula nello sketch è la seguente:

```
int valore = analogRead(A0);  // lettura del valore della porta analogica
int t = valore * 4.8 /= 10; // il valore letto viene diviso per 10
int tF = (t * 9.0 / 5.0) + 32.0; // conversione in gradi Fahrenheit
```

Per la conversione in gradi Fahrenheit, l'operazione è:

```
int tF = (t * 9.0 / 5.0) + 32.0; // conversione in gradi Fahrenheit
```

Una volta ottenuti i due valori basta convertirli in stringa nella funzione seguente:

```
updateThingSpeak("1="+String(t)+"&2="+String(tF));
```

Una volta ottenuta la stringa per l'update dei due valori, basta mandarli via HTTP al server ThingSpeak con le seguenti righe di codice:

```
client.print("POST /update HTTP/1.1\n");
client.print("Host: api.thingspeak.com\n");
client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
client.print("Content-Type: application/x-www-form-urlencoded\n");
client.print("Content-Length: ");
client.print(tsData.length());
client.print("\n\n");
client.print(tsData);
```

Il server di ThingSpeak riceverà i dati e li elaborerà, mettendoli in un grafico, come spiegato nel prossimo paragrafo.

Creare un'app ThingSpeak

Dalla homepage basta fare clic su *Get Started Now* per iniziare subito il lavoro. Dopo aver creato un account gratuito, si può effettuare il login ed entrare subito nel vivo. Per iniziare un progetto si deve selezionare dalla barra dei menu la voce *Canali > I miei canali*. Si aprirà una pagina con un tasto *New Channel* che porta a una pagina *Channel Settings*, simile alla Figura 4.9.

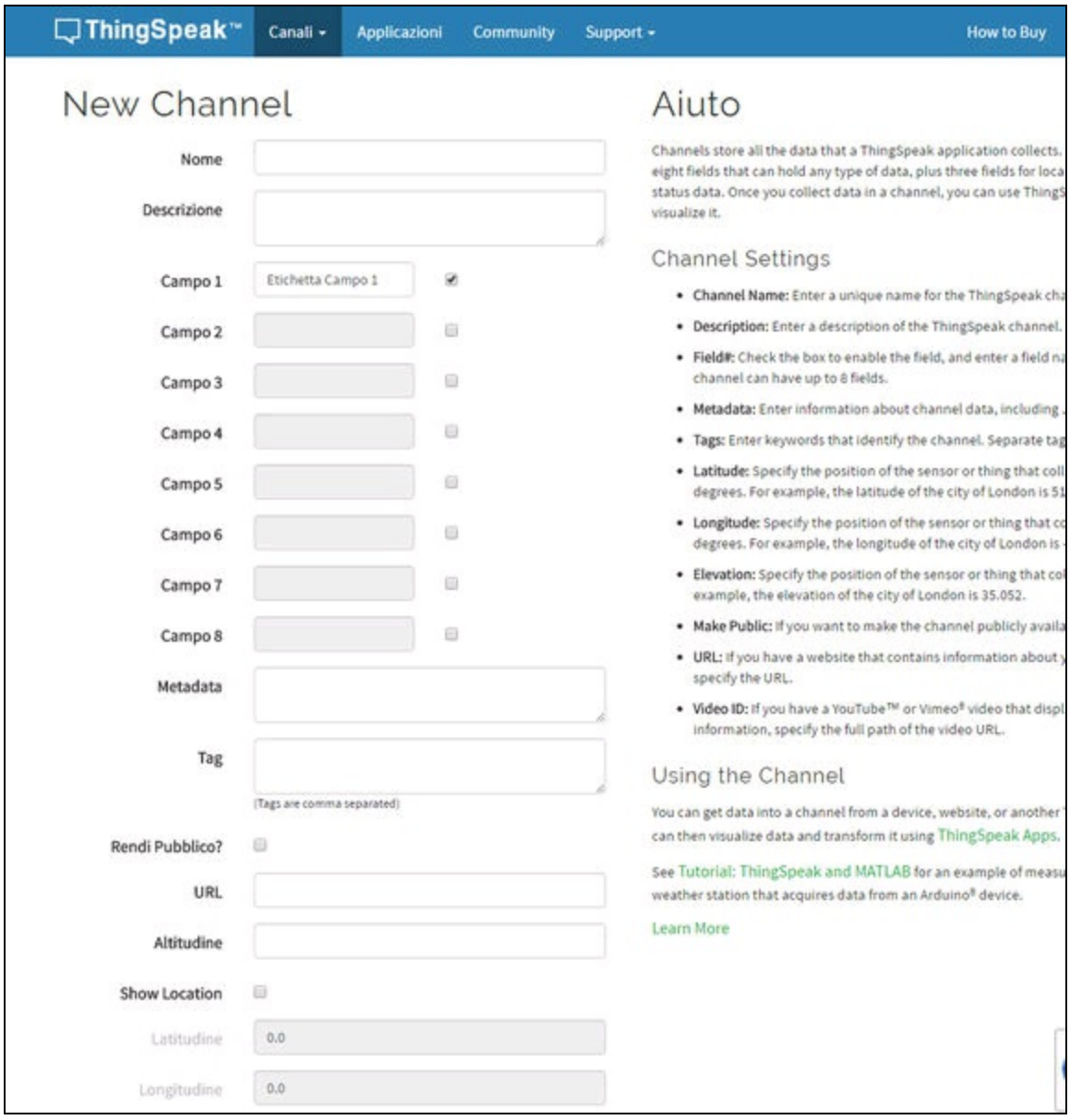


Figura 4.9 La pagina Channel Settings.

Il canale usato per questo esempio è stato pensato per visualizzare un grafico di temperatura in gradi Celsius e in gradi Fahrenheit più un termometro ad ago. Nell'esempio, i dati di temperatura provengono dalla scheda Arduino collegata in rete a un router di Cerete, in provincia di Bergamo (la residenza di chi scrive). Ovviamente basterà inserire i dati della propria location per ottenere lo stesso risultato.

I dati di esempio sono i seguenti.

- *ID Canale*: `36729` (il numero viene assegnato automaticamente e non può essere modificato).
- *Nome*: `Temperatura a Cerete`.
- *Descrizione*: `Rilevamento temperatura locale a Cerete (Bergamo)`.
- *Metadata*: `temperatura`, `Arduino`, `Cerete`, `Bergamo`.
- *Tag*: `Cerete`, `Bergamo`.
- *Latitudine*: `45.86299`.
- *Longitudine*: `9.98912`.
- *Altitudine*: `612 m s.l.m.`.
- *Rendi Pubblico?*: vistare la casella per rendere pubblico il canale.
- *URL*: `http://it.wikipedia.org/wiki/Cerete`.
- *ID Video*: nessuno (si può inserire un link di un video su YouTube o Vimeo).
- *Campo 1*: `Gradi Celsius`.
- *Campo 2*: `Gradi Fahrenheit`.

Gli altri campi possono essere lasciati vuoti. Facendo clic sul pulsante *Save Channel*, il canale viene salvato ed è subito visualizzabile come canale pubblico.

Accanto alla scheda *Channel Settings* c'è la scheda *Chiavi API* che apre la pagina (Figura 4.10) in cui generare le chiavi API di scrittura e di lettura, ovvero per scrivere o leggere i dati sul canale.

Facendo clic sul pulsante *Genera Nuova Chiave di Scrittura* viene generata la chiave API da inserire nello sketch Arduino (si veda il paragrafo precedente). Le chiavi API di lettura possono essere usate per permettere ad altre persone di visualizzare i dati e i grafici del canale.

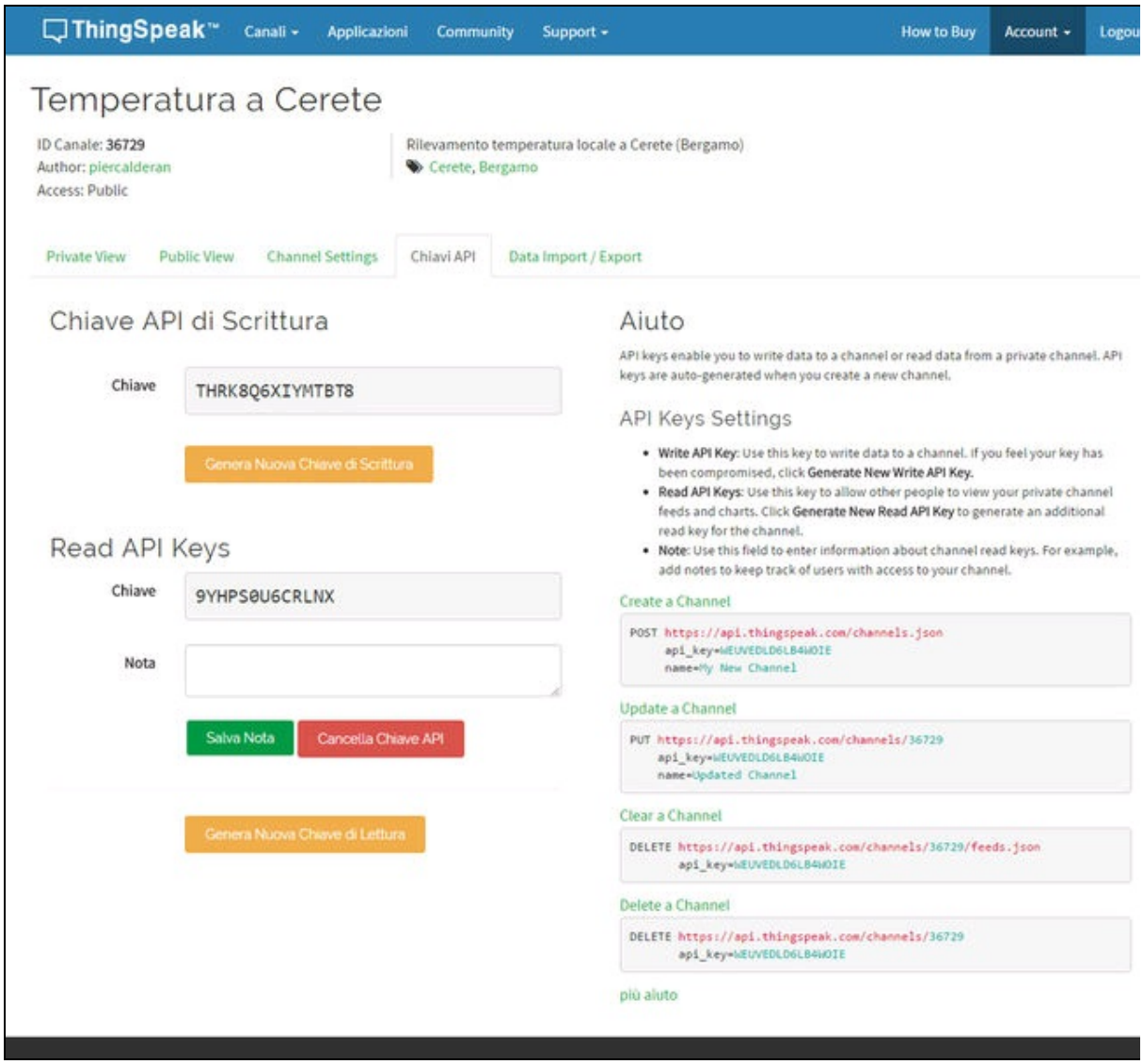


Figura 4.10 La scheda Chiavi API.

Facendo clic sul pulsante *Public View* si può aprire la pagina di visualizzazione dei grafici di temperatura e di Google Maps relativa alla location Cerete, come illustrato nella Figura 4.11.

I riquadri nella *Public View* sono mobili e riposizionabili a piacere. Si possono anche chiudere o minimizzare. Questa pagina può essere visibile pubblicamente all'indirizzo `https://thingspeak.com/channels/36729`, dove l'ultimo numero è l'ID del canale.

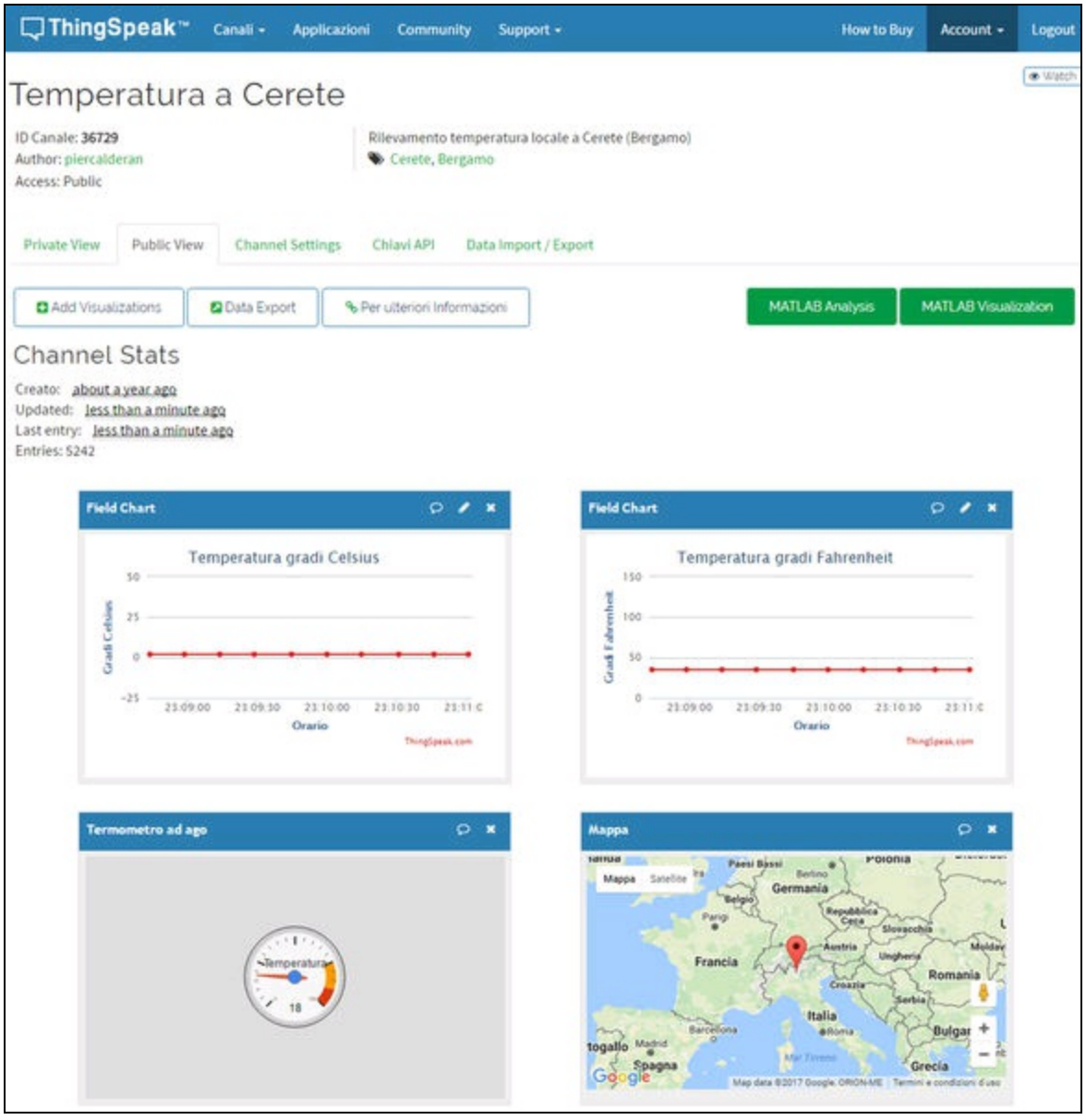


Figura 4.11 La pagina pubblica del canale ThingSpeak.

Nella pagina, si può notare la presenza del termometro ad ago. Per farlo apparire basta fare clic sulla voce *Applicazioni* > *Plugin* della barra

del menu per aprire la pagina in cui aggiungere un nuovo plugin tramite il tasto *Nuovo* (Figura 4.12a). I plugin disponibili sono: *Google Gauge* (manometro ad ago), *Chart With Multiple Series* (grafico con più linee) e *Default* (plugin configurabile dall'utente). Una volta scelto il plugin si può modificare il codice del plugin per personalizzare i parametri HTML, CSS e JavaScript (Figura 4.12b).

I nomi che appaiono sui grafici e i parametri possono venire modificati facendo clic sullo strumento matita presente nella barra del grafico (Figura 4.12c). Qui si possono scegliere i parametri per visualizzare il grafico con una linea, uno spline, una barra verticale o una barra orizzontale e altri parametri per il colore della linea, il colore dello sfondo, il numero di risultati e altre impostazioni per i valori minimi e massimi.

Nella pagina *Public View* sono disponibili le seguenti schede.

- *Add Visualizations*: riapre le finestre e i plugin eventualmente chiusi in precedenza.
- *Per ulteriori informazioni*: apre la pagina di Wikipedia relativa alla location.
- *Developer Info*: consente di scaricare i dati del sensore in formato JSON, XML e CSV.

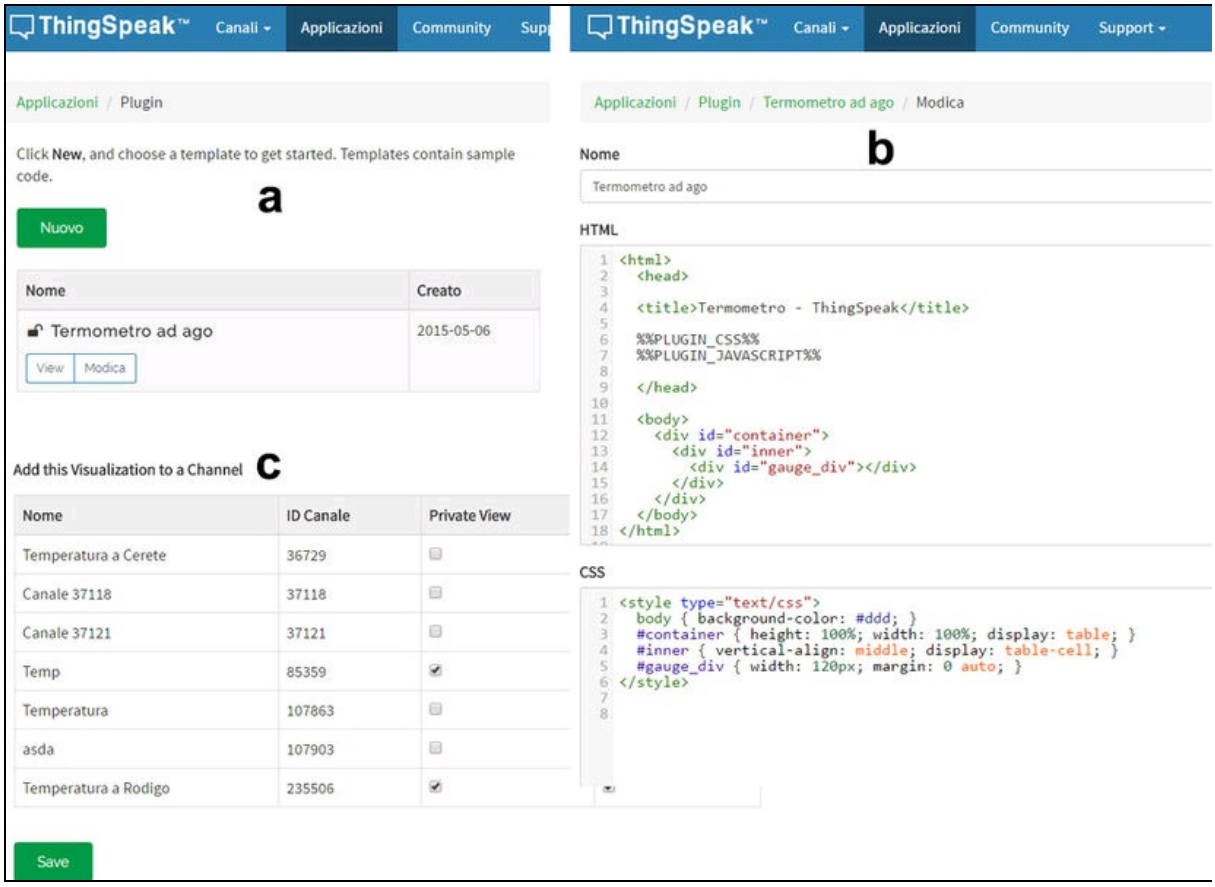


Figura 4.12 La finestra “Plugin” (a). Modifica del codice HTML, CSS e JavaScript del plugin (b). La finestra di modifica dei nomi e dei parametri del grafico (c).

Altre applicazioni

ThingSpeak mette a disposizione altre applicazioni, oltre che per Arduino anche per i seguenti prodotti.

- *ioBridge*: è un produttore di una scheda hardware di controllo e un fornitore di servizi Cloud.
- *Twilio*: è un servizio cloud di tipo IaaS.
- *Prowl*: è un client Growl per iOS, che esegue notifiche PUSH da un computer Mac o Windows a iPhone e iPad.

Tra le applicazioni disponibili ricordiamo le seguenti.

- *ThingTweet*: applicazione che collega l’account Twitter a ThingSpeak e invia messaggi su Twitter utilizzando semplici API.
- *ThingHTTP*: applicazione per creare POST o GET personalizzati ad altri servizi web.
- *TweetControl*: applicazione per ascoltare i comandi da Twitter per eseguire un’azione.
- *React*: applicazione che esegue azioni quando sono soddisfatte le condizioni provenienti dai dati nei canali.
- *TalkBack*: consente ai dispositivi di eseguire comandi accodati.
- *TimeControl*: esegue automaticamente le richieste ThingHTTP o ThingTweet in un momento predeterminato.

La community hackster.io

Ecco un sito, anzi, un portale da salvare nei segnalibri: https://www.hackster.io (Figura 4.13). Il portale è nato nel 2013 ed è diventato in pochi anni il sito di riferimento per la community dei maker e degli hacker. La mission del sito è quella di aiutare chiunque a imparare a progettare, creare e programmare hardware collegato a Internet. La rete della community conta circa 200 000 fra ingegneri, creatori e hacker, 90 partner tecnologici e 100 ambasciatori di Hackster Live, un portale dedicato all'organizzazione di eventi e meetup per maker, a cui tutti possono partecipare.

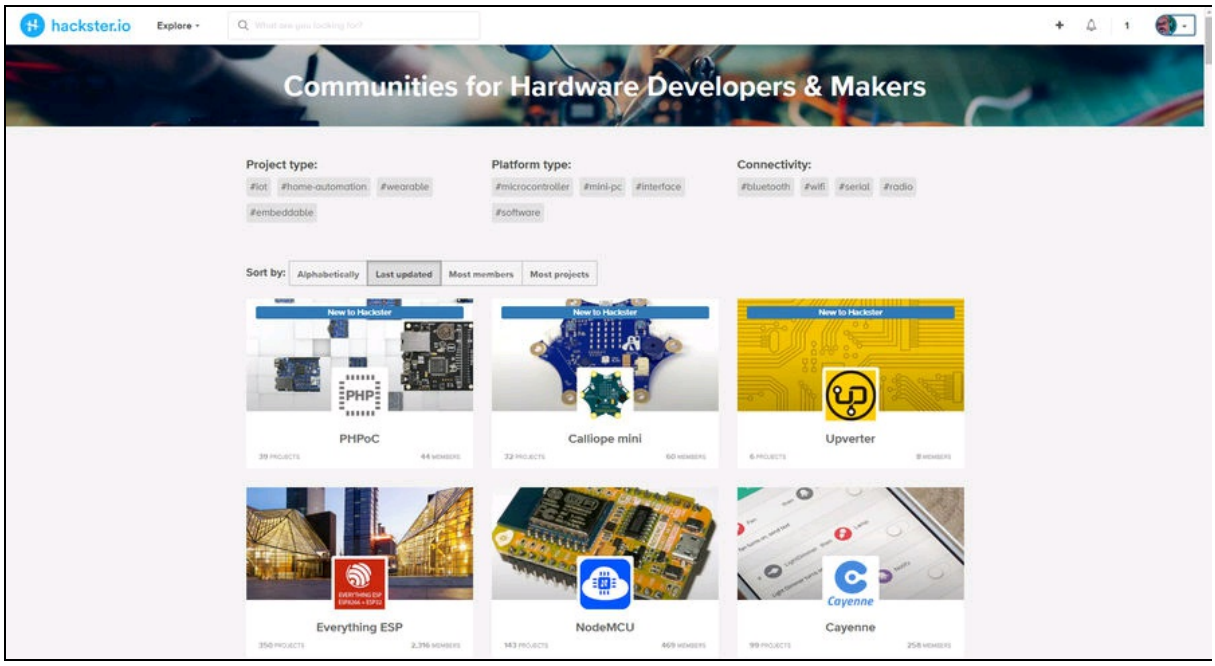


Figura 4.13 Il portale hackster.io.

Le piattaforme per maker sono ormai… tutte! Solo per citarne alcune fra le più popolari:

- Adafruit;
- Arduino;

- Raspberry Pi;
- Intel Edison;
- Particle;
- Microsoft Azure;
- Amazon Alexa;
- Blinky Lights;
- Cayenne;
- Everything ESP;
- Itead;
- NodeMCU;
- Pycom;
- SeeedStudio;
- SparkFun;
- Walabot.

Fra migliaia di progetti è difficile districarsi, ma per fortuna un ottimo motore di ricerca permette di trovare qualsiasi cosa. E la cosa bella è che tutti i progetti sono di libero utilizzo con licenza GPL.

Normalmente i progetti sono completi di schemi di collegamento e file sorgente. Può capitare che qualche progetto non funzioni subito, ma di solito lo si può sempre sistemare… da buon hacker!

Far parte della community hackster.io

Per un appassionato non c'è posto migliore. Oltre ai progetti, il portale si fa spesso promotore di concorsi con in palio premi succulenti. I concorsi si trovano alla pagina *Contests*. C'è anche un'area *Live* che informa sugli eventi organizzati dai maker in tutto il mondo, di cui si può diventare ambasciatore o sponsor.

- Gli eventi Live organizzati dagli ambasciatori sono sostenuti con hardware, software e materiale didattico fornito dagli sponsor.

Inoltre viene fornita la formazione necessaria per costruire una comunità accogliente in modo che gli eventi abbiano successo.

- Gli sponsor Live offrono alla comunità gli strumenti hardware e gli strumenti di formazione. Lo sponsor fornisce tutto quel che serve all’ambasciatore per organizzare meeting e workshop e mettere i prodotti direttamente nelle mani dei maker.

Se si vuole ricevere l’aggiornamento dei nuovi progetti aggiunti al portale, ci si può iscrivere alla newsletter. I progetti vengono aggiornati e mandati via email quotidianamente. Buon lavoro!

Parte II

Progetti

In questa parte:

Monitoraggio meteo

Descrizione

Questo semplice progetto serve a monitorare alcune condizioni meteo, ovvero temperatura, umidità e luce. Si potrebbero monitorare anche altri valori, quali pressione atmosferica, altitudine e velocità del vento, ma abbiamo voluto semplificare al massimo la costruzione della stazione meteo. Alla fine del capitolo ci sono comunque le indicazioni su come implementare facilmente anche un barometro e un anemometro.

Per un monitoraggio meteo sicuro è preferibile collocare la stazione all'esterno, ma senza esporla alla pioggia, a meno che non si voglia inserire la stazione in un contenitore impermeabile. Si può scegliere una posizione riparata, per esempio sotto una pensilina.

Essendo la stazione dotata di connessione Wi-Fi, i dati meteo possono essere inviati in casa e visualizzati sul browser di un computer o su uno smartphone.

CODICE DI ESEMPIO

Tutti i file del progetto "Monitoraggio meteo" sono disponibili nelle risorse del libro presso il sito dell'autore all'indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Arduino UNO;
- 1× modulo ESP8266-01;
- 1× sensore di temperatura e umidità DHT11 (o DHT22);
- 1× sensore di luce (fotoresistore);
- 1× resistore da 10 KΩ.

Sensore di temperatura e umidità DHT

La famiglia di sensori DHT usa la cosiddetta tecnologia OneWire, ovvero “un filo”. Serve infatti un solo collegamento digitale per rilevare contemporaneamente i dati di temperatura e di umidità dell’ambiente.

Il sensore utilizza una tecnica digitale esclusiva, basata su un processore interno a 8 bit. Il sensore è compensato in temperatura e tarato in un’apposita camera di calibrazione che determina in modo preciso un coefficiente che viene poi salvato all’interno della memoria del processore.

Ecco le sue caratteristiche principali.

- Alimentazione: da 3 a 5,5 V.
- Segnale di uscita: digitale del segnale tramite un unico filo (OneWire).
- Elemento sensibile: resistenza in polimero.
- Campo di misura umidità: 20-90% (umidità relativa).
- Temperatura: da 0 a 50 gradi, calibrazione in Celsius.
- Precisione umidità: +/- 4% RH (max +/-5% di umidità relativa).
- Precisione temperatura: +/-2 °Celsius.
- Risoluzione umidità: 1% di umidità relativa.

- Risoluzione temperatura: 0,1 °Celsius.
- Tempo di rilevazione: 2 secondi.

I componenti interni e la piedinatura del sensore DHT11 sono illustrati nella Figura 5.1. Si fa notare che il piedino 3 non è collegato.

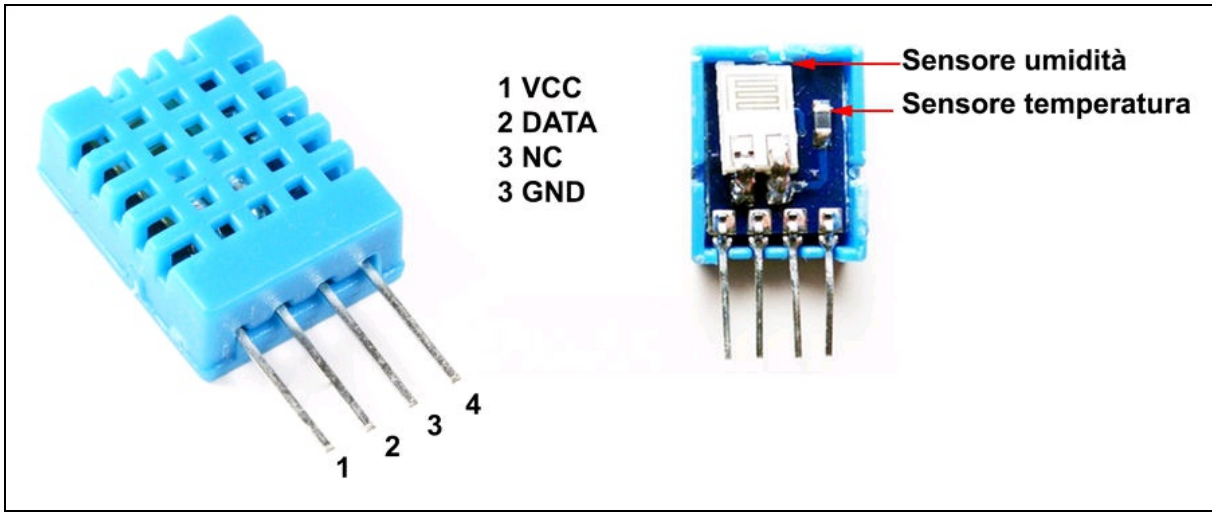


Figura 5.1 Piedinatura del sensore DHT11 e i suoi componenti interni.

Il circuito

Il prototipo del circuito è fatto su breadboard; per il progetto definitivo si consiglia di usare una millefori o, meglio, un circuito stampato, per una soluzione più stabile e duratura.

La Figura 5.2 illustra i collegamenti dei sensori DHT11 e LDR alla scheda Arduino UNO e al modulo Wi-Fi ESP8266-01.

Fare attenzione alla doppia alimentazione di 5 volt per il fotoresistore e di 3,3 volt per il modulo ESP8266 (bisogna alimentare anche il pin CH_PD).

Per la trasmissione seriale dei dati, il pin TX di Arduino va collegato al pin RX di ESP8266.

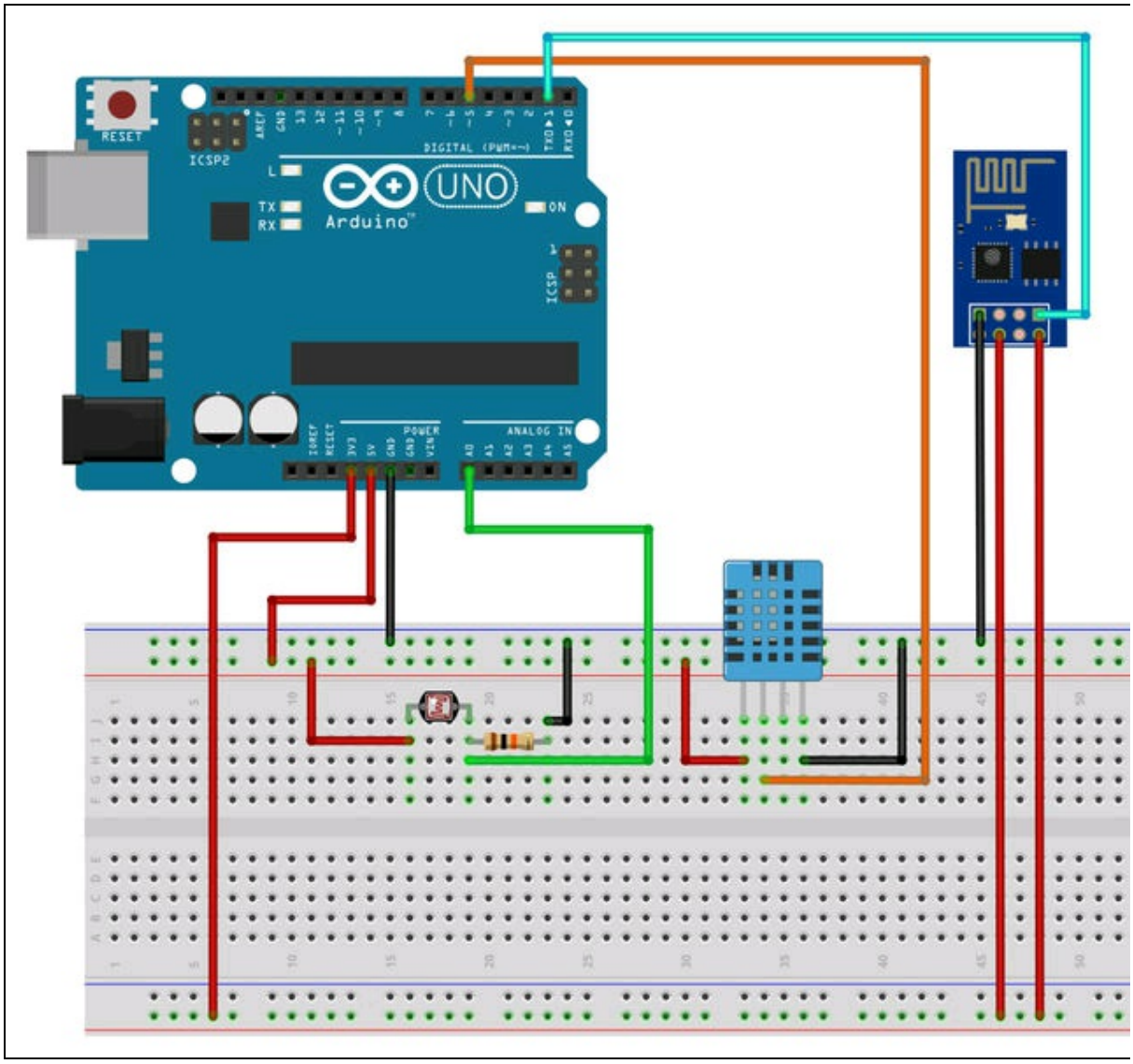


Figura 5.2 Schema di collegamento su breadboard.

Alimentazione

Per l’alimentazione del circuito, ovvero della scheda Arduino e il modulo ESP8266, si dovrà impiegare una pila da 9 volt usando la presa incorporata, oppure una batteria LiPo ricaricabile, usando i pin di alimentazione +5 V. L’uscita a 3,3 volt per l’ESP8266 è comunque garantita.

Contenitore

Per l’alloggiamento del circuito si può pensare a una scatola in plastica pronta all’uso oppure a stampare uno dei tanti contenitori di cui si trovano online i file STL per la stampa 3D.

Per esempio, su Thingiverse, si possono trovare contenitori per Arduino di tutti i tipi. Uno abbastanza capiente per contenere anche uno shield o per altri circuiti, come una pila e un modulo ESP8266-01, è disponibile all’indirizzo `http://www.thingiverse.com/thing:206362` (Figura 5.3). Si tenga presente che l’involucro non è impermeabile e non va esposto alla pioggia.

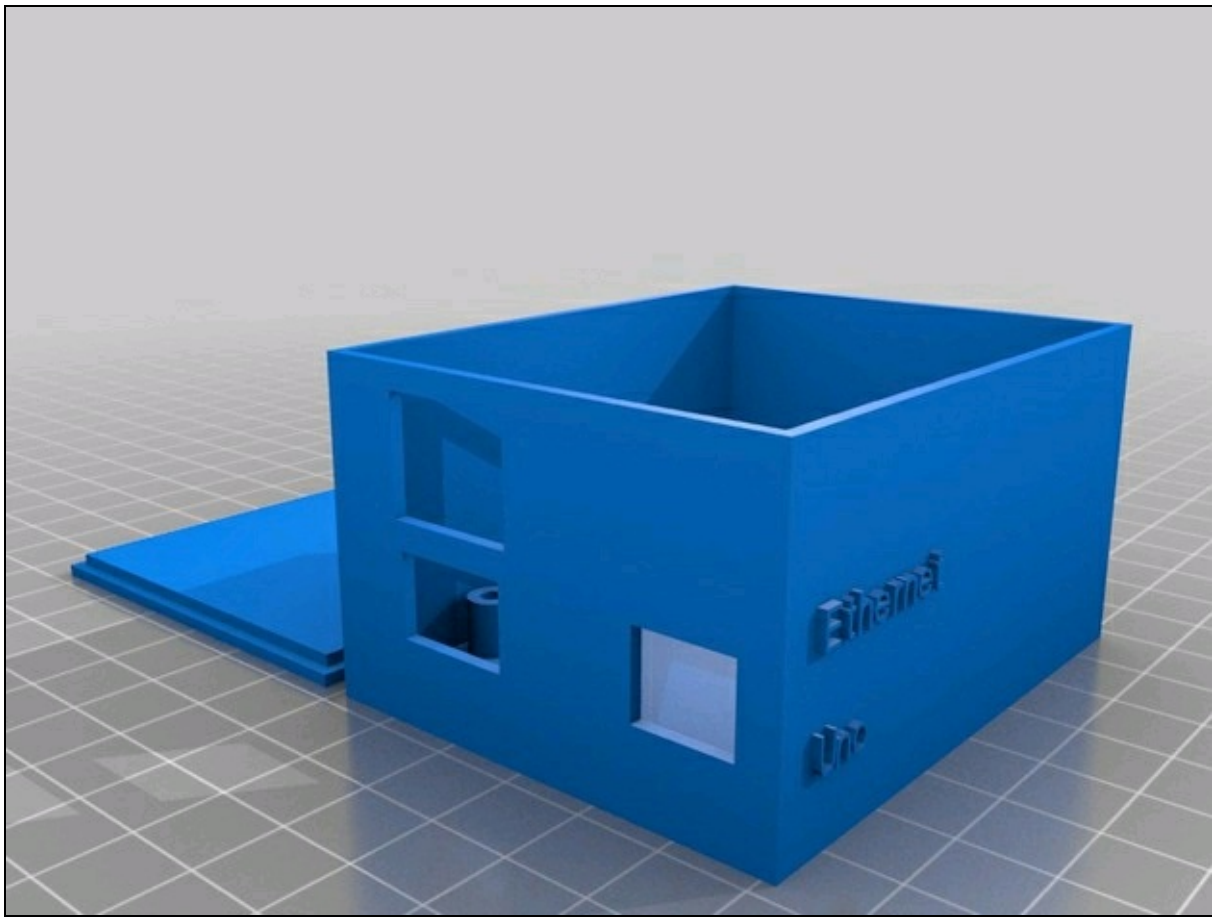


Figura 5.3 La scatola per Arduino e uno shield da stampare in 3D.

Il codice

Per la lettura dei dati dei sensori occorre caricare uno sketch su Arduino. Per la trasmissione dei dati sul Web uno sketch per ESP8266.

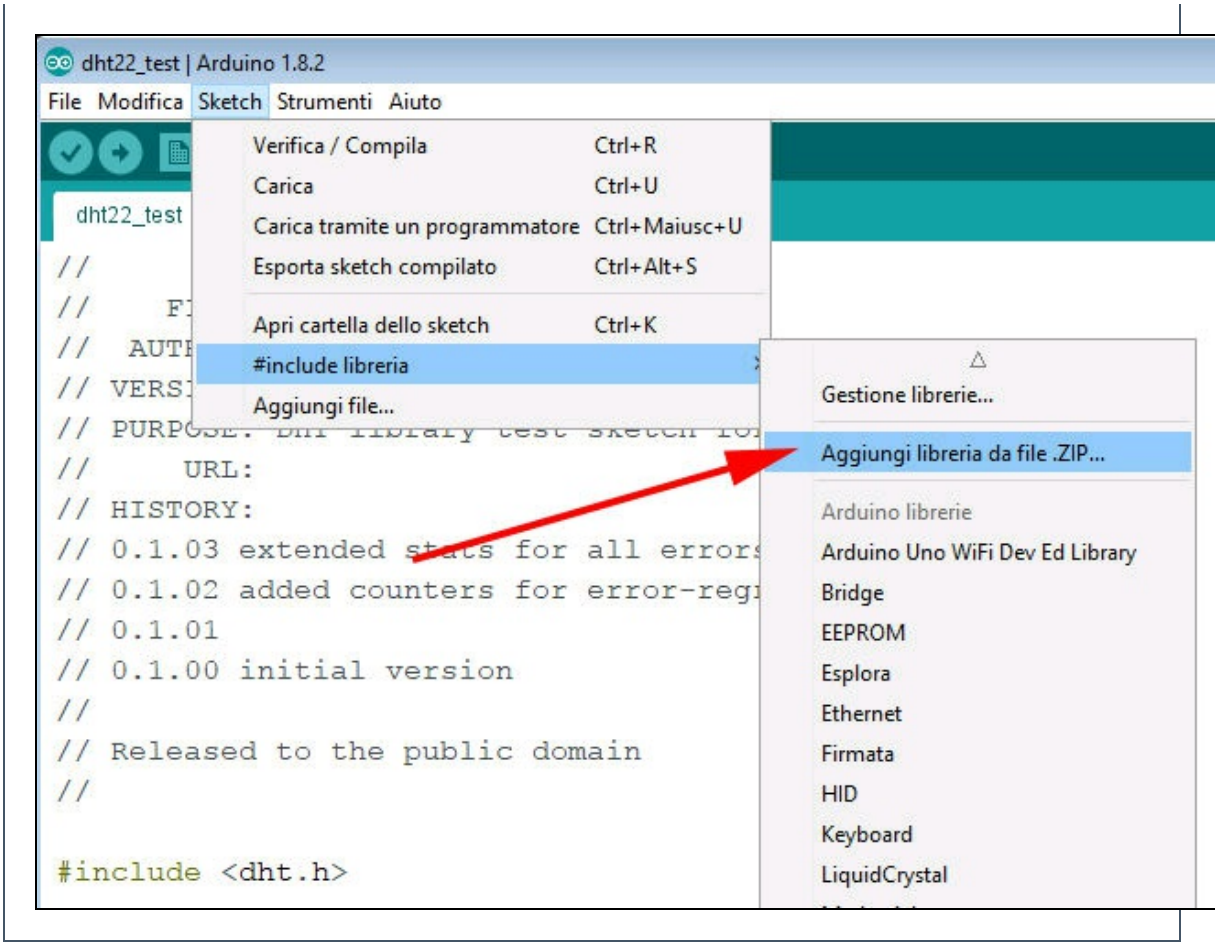
Codice per Arduino

Lo sketch è molto semplice, ma prevede l'installazione della libreria per il sensore DHT. Le informazioni di questa libreria e il download sono disponibili all'indirizzo `http://playground.arduino.cc/Main/DHTLib`. Si può scaricare la libreria anche dalle risorse del libro assieme agli sketch di questo progetto.

Come aggiungere una libreria all'IDE di Arduino

1. Selezionare il menu *Sketch* > *#include libreria* > *Aggiungi libreria da file zip*.
2. Individuare il file zip scaricato della libreria.
3. Fare clic su Apri.
4. La libreria viene aggiunta automaticamente nella cartella libraries in `C:\...\documenti\Arduino`.
5. Gli esempi della libreria installata sono disponibili dal menu File > Esempi.

Se la libreria non è in una cartella compressa, trascinare la cartella della libreria nella suddetta cartella libraries oppure nella cartella libraries del programma Arduino (per esempio: `C:\Program Files (x86)\Arduino\libraries`). In questo caso bisogna riavviare l'IDE di Arduino.



Il listato per Arduino prevede l'inclusione della libreria DHT, che svolge tutto il lavoro di conversione dei dati OneWire sul pin digitale 5, sia per l'umidità relativa sia per la temperatura. Per la lettura dei lux, basta leggere l'ingresso analogico A0 collegato al partitore di tensione costituito da LDR e resistore *pull-down*.

È importante collegare il fotoresistore (come qualsiasi altro sensore analogico passivo) con un resistore di *pull-down* da 10 kΩ per creare il corretto riferimento a massa in rapporto alla tensione di 5 volt all'altro capo del fotoresistore. In pratica, è come creare un potenziometro con una resistenza fissa di 10 kΩ e una che cambia in relazione al flusso luminoso.

Il terminale in comune fra il fotoresistore e il resistore da 10 KΩ corrisponde al "terminale centrale" del potenziometro. La resistenza di

un comune fotoresistore può variare da pochi Ω a qualche MΩ e questa grande variabilità va tenuta in considerazione nel calcolo dei lux.

Chiaramente, prima di diventare lux, i dati prelevati all'ingresso analogico, che sono una serie di numeri decimali da 0 a 1023, devono essere convertiti in qualcosa che abbia senso. Per fare questo, bisogna innanzitutto considerare la relazione che lega la resistenza del fotoresistore all'intensità luminosa. La Figura 5.4 illustra la relazione inversamente proporzionale di resistenza e luminosità.

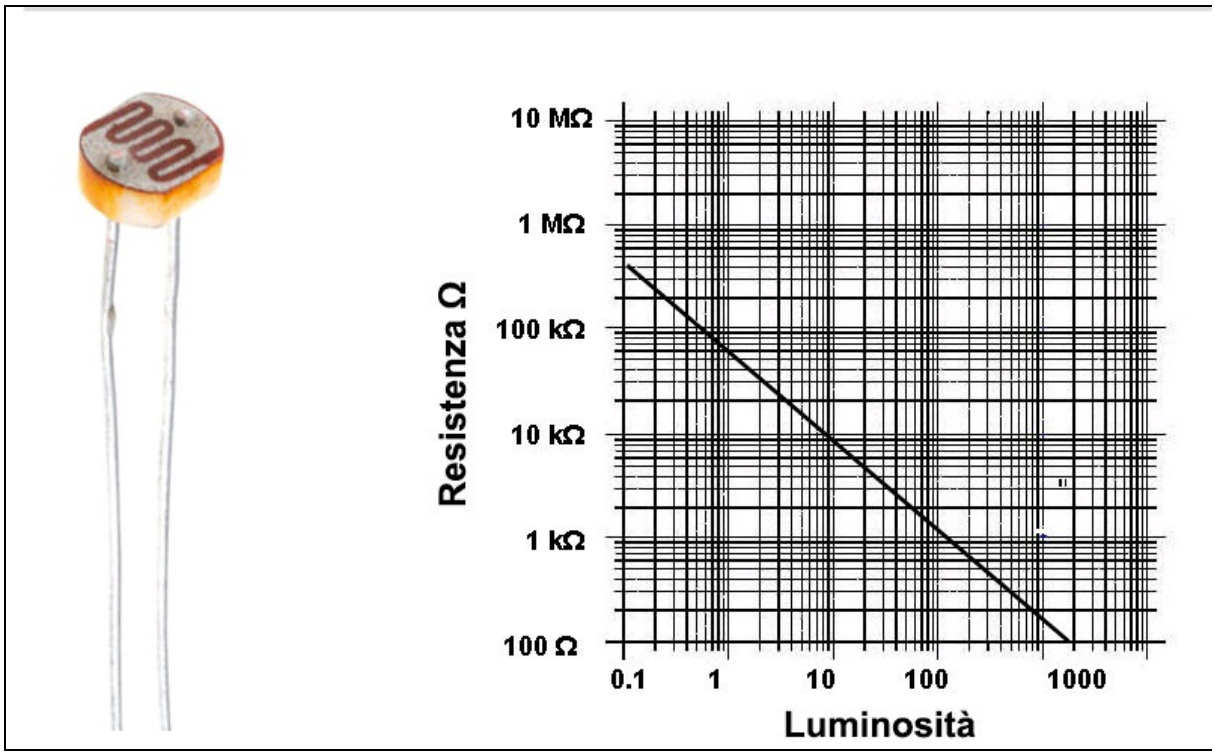


Figura 5.4 LDR (a sinistra) e relazione resistenza/luminosità (a destra).

La pendenza della retta ci fa capire la proporzione inversa fra resistenza e quantità di luce: più luce = minor resistenza e viceversa. Il grafico ci fa capire anche che la scala è logaritmica. Questo perché è basata sulla percezione dell'occhio umano, che non è lineare. Un po' come l'udito e questo complicherà un po' i calcoli.

Lux

Nel Sistema Internazionale il lux è l'unità di misura per l'illuminamento. Un lux è pari a un lumen su un metro quadrato. È l'unità di misura relativa alla luce percepita dall'occhio umano, attraverso una curva di sensibilità soggettiva. In base a uno standard condiviso si possono misurare i seguenti valori di lux in base a tipiche condizioni ambientali:

Luce del sole	32 000 - 100 000 lux
Riflettori	1 000-2000 lux
Ufficio luminoso	400-500 lux
Luce riflessa luna piena;	1 lux

Misurare l'intensità luminosa

Senza entrare nel dettaglio scientifico, si possono identificare questi valori approssimativi di resistenza in relazione al grafico della pendenza (ottenibile dal datasheet del fotoresistore). Questo valore viene chiamato *Gamma Value* e di solito è pari a 0,7 ~ 0,9.

Riportiamo qui di seguito alcuni valori standard di lux e di resistenza, riferiti a diverse condizioni ambientali.

Condizioni di luce	lux	Resistenza del fotoresistore Ω
Luce scarsa	1	100 kΩ
Stanza illuminata	10	70 kΩ
Illuminazione diretta	100	1,5 kΩ
Luce diurna	1000	100 Ω

Pertanto, la relazione che lega resistenza, lux e pendenza (Gamma Value) è data dalla seguente formula logaritmica:

$$log(lux) = -pendenza * log(res)$$

$$log(lux) / log(res) = -pendenza$$

$$-pendenza = logres(lux)$$

Si noti che la pendenza è negativa (rappresentata nel piano cartesiano come una funzione y = -x). Dal grafico si deduce quindi che la resistenza a 1 lux è di circa 70 kΩ.

Per cui, ponendo:

$$pendenza = 0,7 \ (Gamma \ Value)$$

$$-0,7 = lux^{res}$$

… e semplificando:

$$1 = lux^{res/-0.7}$$

… avremo:

$$lux = res^{1/-0.7}$$

Il valore `res` della resistenza nel nostro circuito è dato dal rapporto fra la resistenza risultante dalla lettura della tensione d'ingresso e la resistenza di riferimento che a 1 lux è di 70 kΩ. Pertanto la formula finale sarà:

$$lux = (res\_lettura/res\_rif)^{1/-0.7}$$

Bisognerà tradurre questa formula in codice per Arduino.

Codice commentato

All'inizio dello sketch si deve importare la libreria e istanziare l'oggetto `dht` per il modulo DHT11 collegato al pin digitale 5.

```
#include "DHT.h"
    DHT dht(5, DHT11);
```

Quindi, si definiscono tutte le variabili per il luxmetro.

```
int sensore = A0; // variabile di lettura dell'ingresso analogico
float valR;               // lettura resistenza
float Vout;               // uscita partitore
float Vin = 5.0;          // tensione di riferimento
float R_nota = 10000.0;   // resistenza pulldown (10 KΩ)
float ldr;                // valore risultante del fotoresistore
float ldr_1 = 70000.0;    // valore illuminamento unitario ldr
float gamma  = 0.6;       // Valore gamma della fotoresistore
double lux;               // lux risultanti
```

La funzione luxmetro legge i dati dalla porta analogica A0 e li converte in lux, rispettando esattamente la formula suesposta.

```
void luxmetro() { // funzione luxmetro
    valR = 0;
    for (int i = 0; i < 5; i++)   { // ciclo di 5 letture per affinare il valore
        int valoreldr = analogRead(sensore);
        delay (10);
        valR = valR + valoreldr;
    }
    valR = valR/5; // media aritmetica delle 5 letture
    Vout = (Vin/1024.0 * valR);             // valore in volt in uscita dal
```

```
partitore
    ldr = ((R_nota * Vin/Vout )- R_nota);    // calcolo della resistenza (ohmetro)
    lux = pow((ldr/ldr_1), (1.0/-gamma));    // calcolo dei lux con elevamento a
potenza di ldr alla 1/-0.6
}
```

Nel `setup()` si inizializza la porta seriale a 115200 baud per la trasmissione seriale a ESP8266 e il sensore DHT.

```
void setup() {
    Serial.begin(115200);  // 115200 baud per la trasmissione seriale a ESP8266
    dht.begin();           // inizializzazione del sensore DHT
}
```

Nel `loop()` viene chiamata la funzione `luxmetro()` e quindi si esegue la lettura del sensore DHT. Si noti l'uso della funzione C++ `isnan` (*is not a number*) che evita la lettura di dati non numerici.

I dati di lux, umidità e temperatura vengono inviati alla porta seriale con l'istruzione `Serial.print`, che provvede a mandare anche un punto e virgola (`;`) come elemento di separazione dei dati. Un ritardo alla fine della funzione provvede all'invio dei dati ogni secondo, ma si può anche dilatare questo tempo.

```
void loop() {
    luxmetro(); // chiamata alla funzione luxmetro
    float h = dht.readHumidity();      //lettura umidità
    float t = dht.readTemperature();   //lettura temperatura
    if (isnan(h) || isnan(t)) // is not a number (se non è un numero…)
    {
        return; // ritorna
    }
    Serial.print(String(lux));Serial.print(";");   // invio lux  più separatore
    Serial.print(String(h));Serial.print(";");     // invio umidità più separatore
    Serial.print(String(t));Serial.print(";\n");   // invio temperatura più
separatore e un a capo
    delay(1000);
}
```

Codice per ESP8266

Lo sketch per il modulo ESP8266-01 è un web server con funzioni avanzate. Il suo compito è quello di ricevere i dati dai tre sensori di Arduino e di renderli disponibili online, in base alla richiesta proveniente da qualsiasi browser collegato allo stesso IP.

La visualizzazione sul Web poteva essere limitata al solo testo, ma ci siamo spinti un po' più in là e abbiamo pensato a una visualizzazione grafica in stile ThingSpeak. La Figura 5.5 illustra quello che vogliamo visualizzare online.

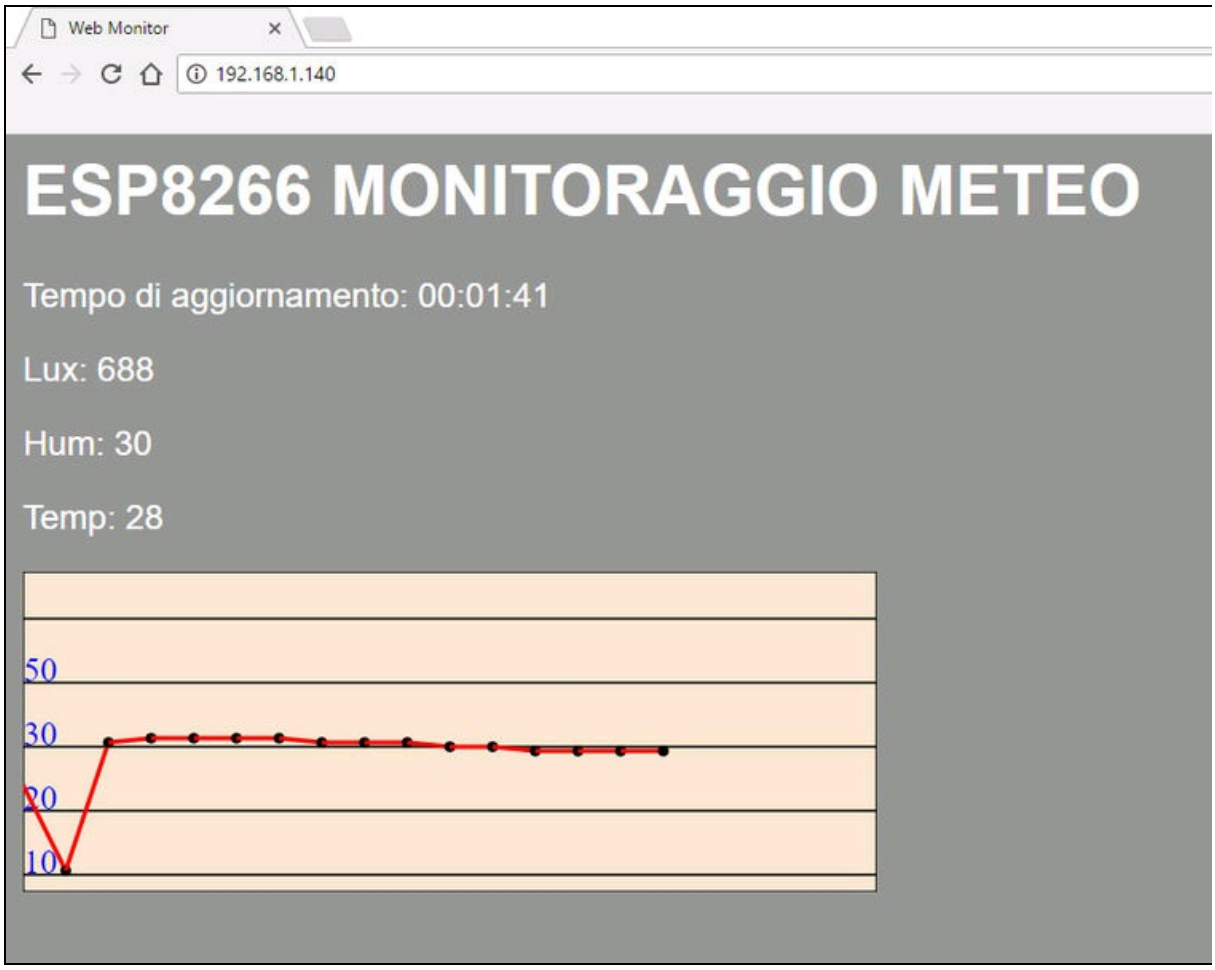


Figura 5.5 Visualizzazione grafica sul Web dei dati di monitoraggio meteo.

Il grafico viene prodotto grazie alla possibilità dei browser di gestire il formato SVG (*Scalable Vector Graphics*) che definisce grafici basati su vettori in formato XML. Un ottimo tutorial per imparare a usare tutti gli elementi grafici SVG è disponibile presso *w3schools* all'indirizzo `https://www.w3schools.com/graphics/svg_intro.asp`.

Il listato non è eccessivamente complesso, ma per chi non ha esperienza con l'uso del protocollo HTTP e della programmazione

HTML, diamo qui di seguito alcune indicazioni per poter modificare il codice a piacere.

Codice commentato

All'inizio vanno incluse le librerie per gestire Wi-Fi e web server.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
```

Quindi si devono digitare SSID e password per l'accesso alla rete di casa nei campi `mySSID` e `myPassword`. Si deve istanziare il server sulla porta 80, la porta standard per la navigazione Internet.

```
const char* ssid = "mySSID"; // ssid Wi-Fi
const char* password = "myPassword"; // password
ESP8266WebServer server (80);
```

Quindi si definiscono le variabili da usare per la lettura dei dati in arrivo dalla porta seriale.

```
String data;
String dati[3]={"","",""};
int lx=0;
int hum=0;
int tem=0;
int letture=0;
```

La funzione *HomePage()* contiene tutto il codice per la creazione dei testi HTML e dell'immagine SVG:

- titolo della pagina;
- stile per il colore di fondo (grigio), il font da usare (Arial) e il colore dei font (bianco);
- titolo 1;
- tempo di aggiornamento;
- dati dei sensori;
- immagine SVG.

Nell'intestazione HTTP, viene usata l'istruzione `refresh` per aggiornare la home page ogni 15 secondi, ma si potrà cambiare questo valore a piacere.

Si noti l’uso del carattere escape “\” per poter scrivere i tag HTML nell’IDE e la funzione C++ `snprintf` per poter stampare nel buffer `temp` tutto il contenuto con una sola istruzione. La sintassi è: `int snprintf ( char * s, size_t n, const char * format, ... )`.

Da notare anche l’uso delle variabili `sec`, `min` e `hr` per visualizzare il tempo di aggiornamento. Quest’ultimo non è basato sull’orario in tempo reale, perché Arduino non possiede un RTC (*Real Time Clock*) interno. Per l’acquisizione dell’orario in tempo reale si veda il prossimo paragrafo.

```
void HomePage() {
String data;                // variabile per la lettura della stringa di dati
String dati[3]={"","",""};  // buffer per i tre dati
int lx=0;                   // variabile per i lux
int hum=0;                  // variabile per l'umidità
int tem=0;          // variabile per la temperatura
int letture=0;      // variabile per le letture
byte riga[20];      // variabile per il numero di letture
snprintf (temp, 400,
    "<html>\<head>\<meta http-equiv='refresh' content='15'/>\
    <title>Web Monitor</title>\
    <style>\body { background-color: #969696; font-family: Arial; Color: #ffffff;
}\</style>\
    </head>\<body>\
    <h1>ESP8266 MONITORAGGIO METEO</h1>\
    <p>Tempo di aggiornamento: %02d:%02d:%02d</p>\
    <p>Lux: %02d</p>\
    <p>Hum: %02d</p>\
    <p>Temp: %02d</p>\
    <img src=\"/grafico.svg\" />\
</body>\</html>", hr, min % 60, sec % 60, lx, hum, tem);
```

Alla fine della funzione, si manda al client, con l’istruzione `server.send`, il comando `HTTP 200 OK`, il tipo di contenuto `text/html` e tutto il contenuto del buffer `temp`, ovvero il contenuto della home page. Pensiamo che sia davvero cool!

```
server.send (200, "text/html", temp);
```

Nel `setup()` vengono inizializzate la porta seriale a 115200 baud e la connnessione Wi-Fi. Quindi il ciclo `while` attende la connessione che, se avviene con successo, stamperà sul monitor seriale il nome SSID e l’indirizzo IP al quale ci si dovrà connettere con il browser.

```
void setup (void) {
    Serial.begin (115200);
    WiFi.begin (ssid, password);
        Serial.println ("");
```

Attesa per la connessione (stampa una serie di punti nel monitor seriale).

```
    while (WiFi.status() != WL_CONNECTED) {
        delay (500);
        Serial.print (".");
    }
    Serial.println ("");
    Serial.print ("Connessione a: ");
    Serial.println (ssid);                      // stampa il nome SSID
    Serial.print ("DHCP IP: ");
    Serial.println (WiFi.localIP());            // stampa l'indirizzo IP al quale
connettersi
    server.on ("/", HomePage);                  // attiva la Home Page
    server.on ("/grafico.svg", grafico);        // attiva il grafico SVG
    server.on ("/inline", []()
    {
        server.send (200, "text/plain", "OK");  // intestazione HTTP
    });
    server.begin(); // avvia il server
    Serial.println ("HTTP server avviato");
}
```

Nel `loop()`, con l’istruzione `Serial.readString()`, si legge la stringa inviata da Arduino alla porta seriale. La stringa viene messa in un array di caratteri per poterla suddividere con la funzione C++ `strtok`, che consente di dividere una stringa in base a un carattere di separazione, nel nostro caso un punto e virgola. Dopodiché, si mettono i dati nelle variabili `lx`, `hum` e `tem`. Questi valori verranno visualizzati nella home page come testo e nel grafico.

```
void loop (void) {
    if (Serial.available())
    {
        data = Serial.readString();
        char charBuf[20];
        data.toCharArray(charBuf, 20);
        char * pch;
        pch = strtok (charBuf,";");
        int count = 0;
        while (pch != NULL)
        {
            dati[count]= pch;
            pch = strtok (NULL, ";");
            lx = dati[0].toInt();
            hum = dati[1].toInt();
            tem = dati[2].toInt();
            count++;
```

```
        }
    }
    server.handleClient();
}
```

La funzione `grafico()` serve a creare il grafico SVG per visualizzare i dati di temperatura. Questi dati vengono letti 20 volte e, usando la sintassi SVG, vengono disegnati come linee e punti che si susseguono a intervalli regolari.

La stringa `out` raccoglie e somma tutti gli oggetti SVG che vengono mandati al client attraverso la solita istruzione `server.send`.

Dopo 20 letture il grafico ricomincia da capo. La Figura 5.5 (vista prima) illustra l'andamento della temperatura da 0 a 31 gradi. Questi valori sono stati ottenuti sconnettendo il sensore e riscaldandolo poi con la mano. Nel caso questa calibrazione vada rivista, basterà modificare qualche parametro nel codice SVG.

```
void grafico()
{
    letture++;
    if (letture == 20) letture = 0;
    riga[letture]=tem;
    String out = "";
    char temp[200];
    out += "<svg xmlns=\"http://www.w3.org/2000/svg\" version=\"1.1\"
        width=\"400\" height=\"150\">\n";
    out += "<rect width=\"400\" height=\"150\" fill=\"rgb(250, 230, 210)\"
        stroke-width=\"1\" stroke=\"rgb(0, 0, 0)\" />\n";
    out += "<line x1=\"0\" y1=\"22\" x2=\"400\" y2=\"22\"
        style=\"stroke:rgb(0,0,0);stroke-width:1\" />";
    out += "<line x1=\"0\" y1=\"52\" x2=\"400\" y2=\"52\"
        style=\"stroke:rgb(0,0,0);stroke-width:1\" />";
    out += "<line x1=\"0\" y1=\"82\" x2=\"400\" y2=\"82\"
        style=\"stroke:rgb(0,0,0);stroke-width:1\" />";
    out += "<line x1=\"0\" y1=\"112\" x2=\"400\" y2=\"112\"
        style=\"stroke:rgb(0,0,0);stroke-width:1\" />";
    out += "<line x1=\"0\" y1=\"142\" x2=\"400\" y2=\"142\"
        style=\"stroke:rgb(0,0,0);stroke-width:1\" />";
    out += "<text x=\"0\" y=\"51\" fill=\"blue\">50</text> />";
    out += "<text x=\"0\" y=\"81\" fill=\"blue\">30</text> />";
    out += "<text x=\"0\" y=\"111\" fill=\"blue\">20</text> />";
    out += "<text x=\"0\" y=\"141\" fill=\"blue\">10</text> />";
    out += "<g stroke=\"black\">\n";
    int y = 20 * 2;
    int inc=0;
    for (int x = 0; x < (letture*20); x+=20)
    {
        inc = x/20;
        int y2 = riga[inc] * 2;
        sprintf(temp, "<line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\"
            stroke=\"red\" stroke-width=\"2\" />\n", x, 140 - y, x + 20, 140 -
```

```
y2);
        out += temp;
        sprintf(temp, "<circle cx=\"%d\" cy=\"%d\" r=\"%d\"
            fill=\"black\"/>\n", x+20, 140 - y2, 2);
        out += temp;
        y=y2;
    }
    out += "</g>\n</svg>\n";
    server.send (200, "image/svg+xml", out);
}
```

NOTA

Per la prima visualizzazione dell'IP a cui ci si deve collegare si deve utilizzare il monitor seriale. Se in futuro l'IP Wi-Fi dovesse cambiare, basterà aprire la pagina di gestione del router di casa (di solito 192.168.1.1) e vedere quale IP è stato assegnato dal server DHCP alla connessione Wi-Fi del modulo ESP8266.

Aggiunta di un server NTP

Per ottenere l'orario preciso da Internet ci si può connettere a un server NTP (*Network Time Protocol*). È un protocollo per sincronizzare gli orologi dei computer connessi a Internet. L'NTP è un protocollo client-server, per cui il client manda una richiesta e ricevere dal server un pacchetto di dati UDP. Questo pacchetto andrà decodificato per estrarre il tempo UTC. La Figura 5.6 illustra il tempo di aggiornamento reale inviato da un server NTP.

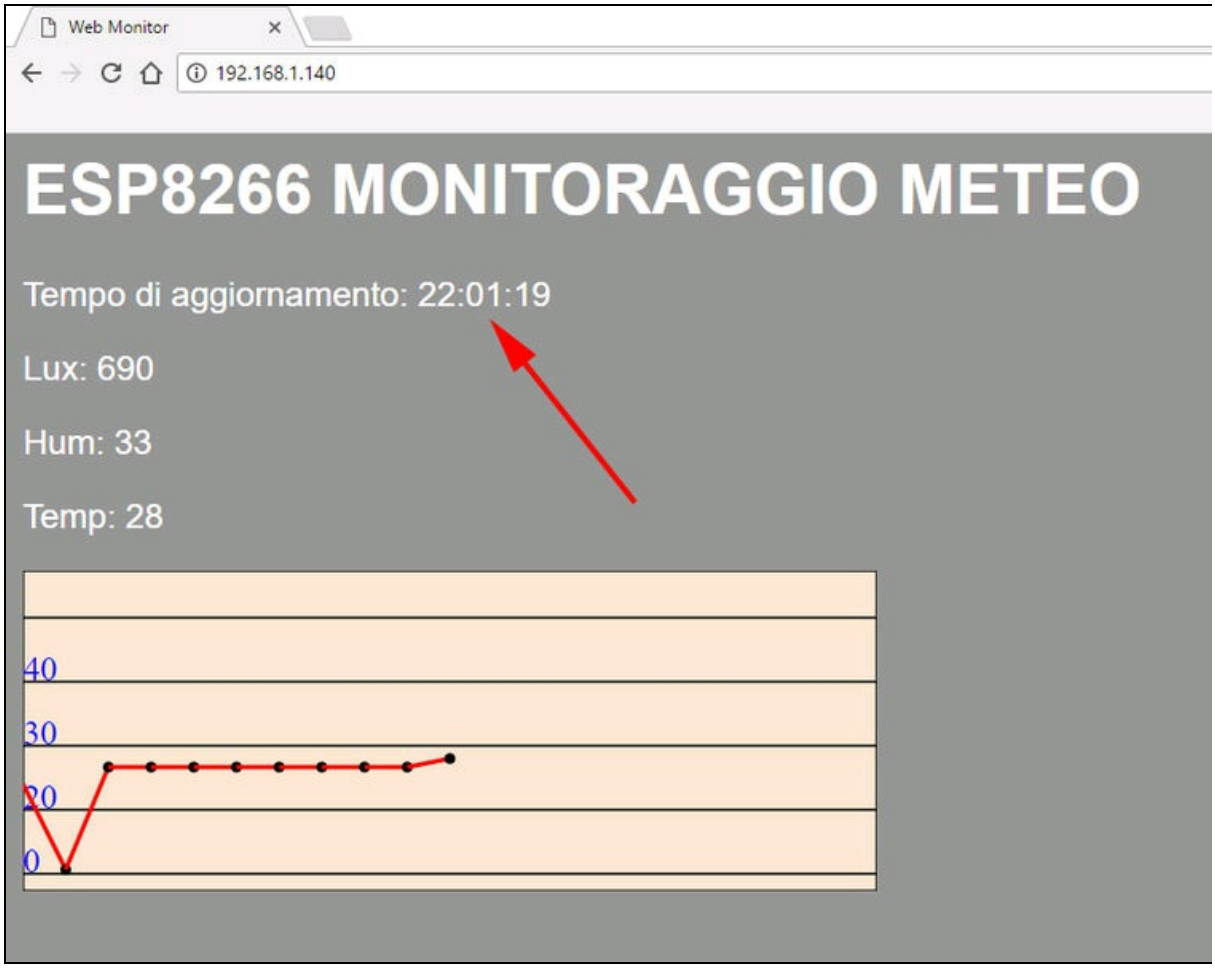


Figura 5.6 Monitoraggio meteo con orario preciso da un server NTP.

Il codice

L’esempio completo con l’orario NTP perfettamente funzionante è disponibile nelle risorse del libro. Qui di seguito vediamo solo uno stralcio delle funzioni NTP che abbiamo aggiunto.

All’inizio viene inclusa la libreria WiFiUdp per ESP8266 per creare l’oggetto `udp`.

```
#include <WiFiUdp.h>
    WiFiUDP udp;
```

La funzione `ntp()` compie la “magia”. Una volta mandata la richiesta a un server NTP con la funzione `sendNTPpacket(timeServerIP)`, si otterrà un

pacchetto UDP di risposta. Per informazioni approfondite sul protocollo NTP, si veda Wikipedia o un'altra fonte. Qui diremo solo che il tempo dei diversi server NTP è sincronizzato, a seconda dei "layer", con un orologio atomico, con un segnale GPS o con un orologio radiocontrollato. Per convenzione si fa risalire l'inizio del tempo alla mezzanotte del 1º gennaio 1970. Questo tempo viene chiamato `epoch` nei sistemi Unix e Unix-like. Una volta ottenuto l'offset da `epoch`, con dei semplici calcoli aritmetici si estrae l'orario in ore, minuti e secondi.

```
void ntp()
{
    WiFi.hostByName(ntpServerName, timeServerIP);
    sendNTPpacket(timeServerIP);
    delay(1000);

    int cb = udp.parsePacket();
    if (!cb) { }
    else {
        udp.read(packetBuffer, NTP_PACKET_SIZE);
        unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
        unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
        unsigned long secsSince1900 = highWord << 16 | lowWord;
        const unsigned long seventyYears = 2208988800UL;
        unsigned long epoch = secsSince1900 - seventyYears;
        ora = (epoch % 86400L) / 3600;   // valore delle ore
        minuti=(epoch % 3600) / 60;      // valore dei minuti
        secondi=epoch % 60;              // valore dei secondi
    }
}
```

A questo punto, basta chiamare la funzione `ntp()` all'interno della funzione `HomePage()` e si sostituiscono i valori ottenuti dal server NTP alle variabili `sec`, `min` e `hr`. Si fa notare che per l'orario UTC in Italia bisogna aggiungere due ore.

```
ntp();
int sec = secondi;
int min = minuti;
int hr = ora+2; // UTC+2 in Italia
```

ESP8266 con ThingSpeak

Un'alternativa più semplice al progetto precedente può essere basata solo sul modulo ESP8266-01, sfruttando la sua unica porta GPIO2. In

questo modo si possono leggere tranquillamente i dati del sensore DHT11 di temperatura e umidità, senza usare il fotoresistore. Inoltre, si potrebbe usare ThingSpeak per la visualizzazione dei dati sul Web. Volendo usare più sensori, bisognerà usare un modello di ESP8266 dotato di più pin GPIO.

La Figura 5.7 illustra il semplice collegamento di un sensore DHT11 al modulo ESP8266-01. Per collegare il modulo al circuito sono disponibili in commercio anche degli adattatori per ESP8266-01 (nel riquadro della figura).

Il codice

Il codice per ESP8266 e ThingSpeak è disponibile nelle risorse del libro, quindi diamo solo alcune indicazioni di massima.

All'inizio dello sketch vanno incluse le librerie DHT e ESP8266WiFi. Quindi bisogna inserire SSID e password e chiave API del canale ThingSpeak, ottenuta come spiegato nel Capitolo 4.

```
#include <ESP8266WiFi.h>
#include <DHT.h>
const char* ssid = "mySSID";           // ssid Wi-Fi
const char* password = "myPassword ";  // password
String apiKey = "PQABFCXLB0AS8KW";     // chiave API ThingSpeak
```

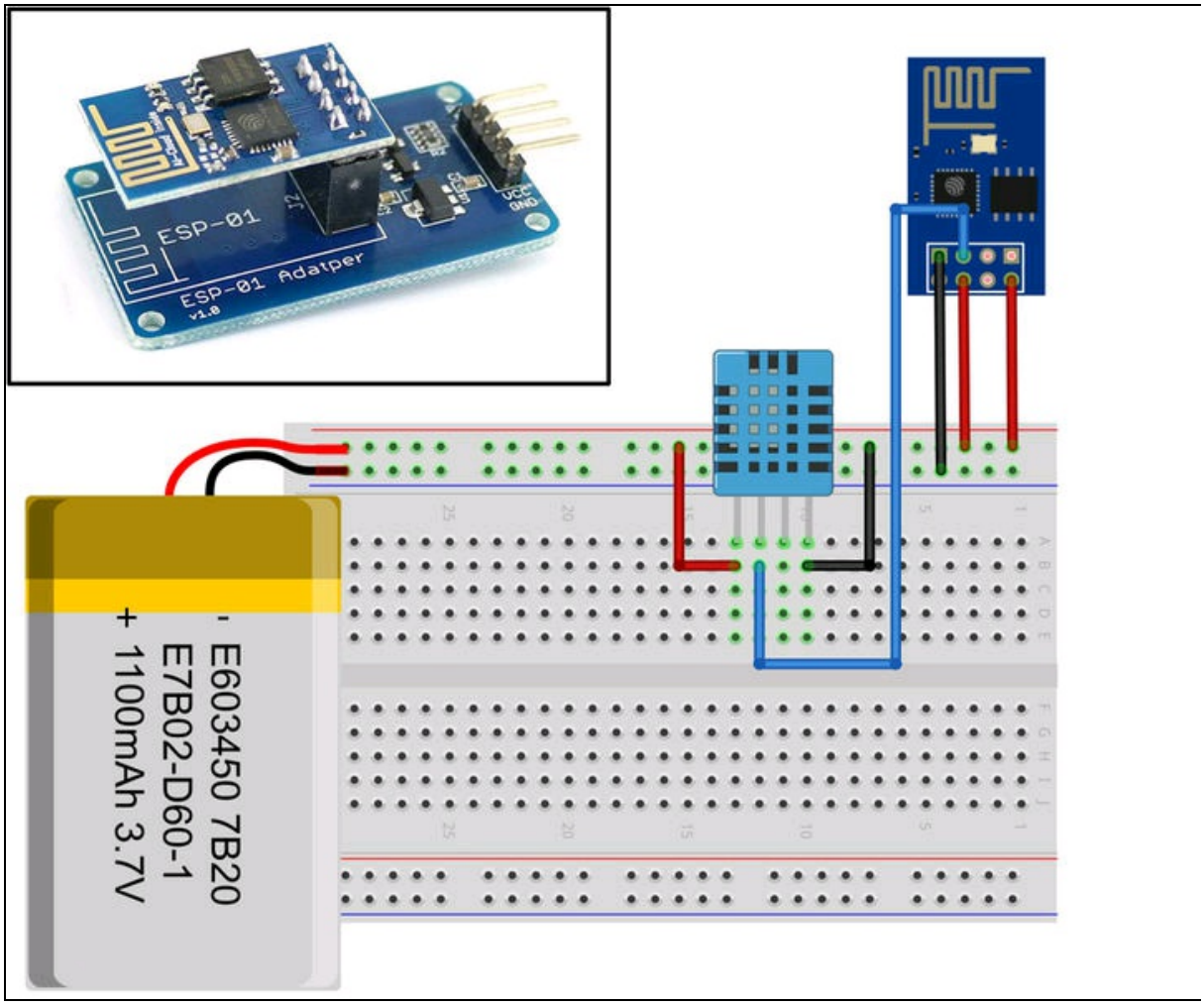


Figura 5.7 Collegamento di un sensore DHT11 al modulo ESP8266-01

Una volta ottenuta la connessione a ThingSpeak, nel `loop()` avviene la lettura del sensore e l'invio dei dati.

```
void loop() {
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    if (isnan(h) || isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
}
```

I dati vanno inviati a `field1` e a `field2`, corrispondenti a temperatura e umidità nel canale ThingSpeak. Per comodità si consiglia di scrivere tutto in una sola stringa `postStr`.

```
    if (client.connect(server,80)) {
        String postStr = apiKey;
        postStr +="&field1=";
        postStr += String(t);
        postStr +="&field2=";
        postStr += String(h);
        postStr += "\r\n\r\n";
```

Dopo la consueta intestazione HTTP, si deve inviare la suddetta stringa `postStr` e chiudere la connessione. Un ritardo programmato di 20 secondi chiude il listato.

```
        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(postStr.length());
        client.print("\n\n");
        client.print(postStr);
    }
    client.stop();
    delay(20000); // il tempo di aggiornamento a ThingSpeak
}
```

Su ThingSpeak, basterà creare un canale con due campi (`field1` e `field2`) chiamati *Temperatura* e *Umidità*. Il risultato è visibile nella Figura 5.8.

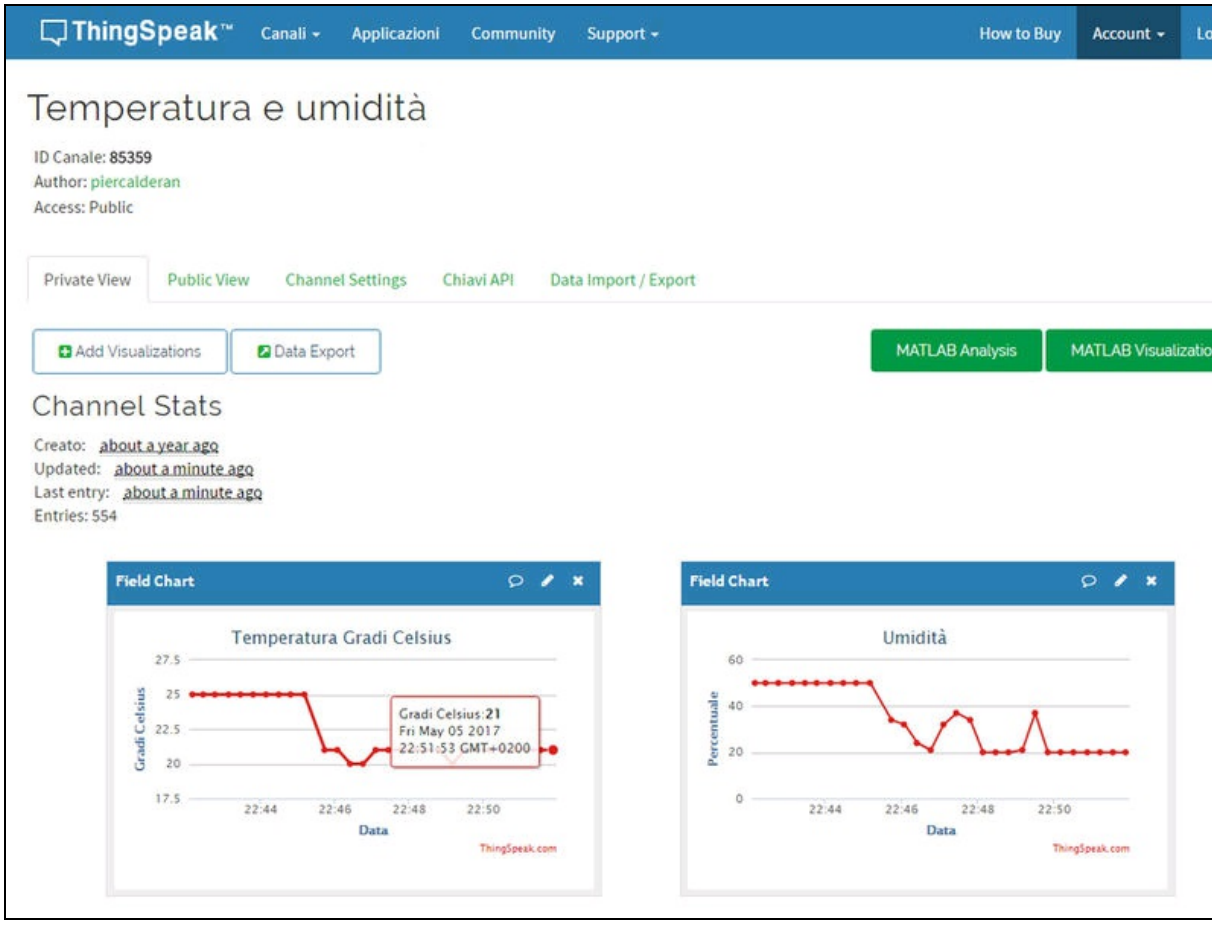


Figura 5.8 La pagina del canale “Temperatura e umidità” su ThingSpeak.

Aggiungere un barometro

Per aggiungere un sensore di pressione, altrimenti conosciuto come barometro, si può optare per il modulo BMP180 (Figura 5.9a), dotato anche di sensore di temperatura, oppure per il BME280 (Figura 5.9b), dotato di sensore di temperatura e anche umidità. Entrambi sono disponibili da vari produttori su Internet.

Caratteristiche principali del modulo BMP180.

- Interfaccia digitale a due fili I²C.
- Ampia gamma di pressione barometrica.
- Tensione di alimentazione da 1,8 V a 3,6 V.

- Include sensore di temperatura.

Caratteristiche principali del modulo BME280.

- Interfaccia di comunicazione I^2C e SPI.
- Intervallo temperatura: da -40C a 85C.
- Intervallo umidità: 0 - 100% RH.
- Intervallo pressione: da 30 000Pa a 110 000Pa.
- Intervallo altitudine: da 0 a 9200 m.

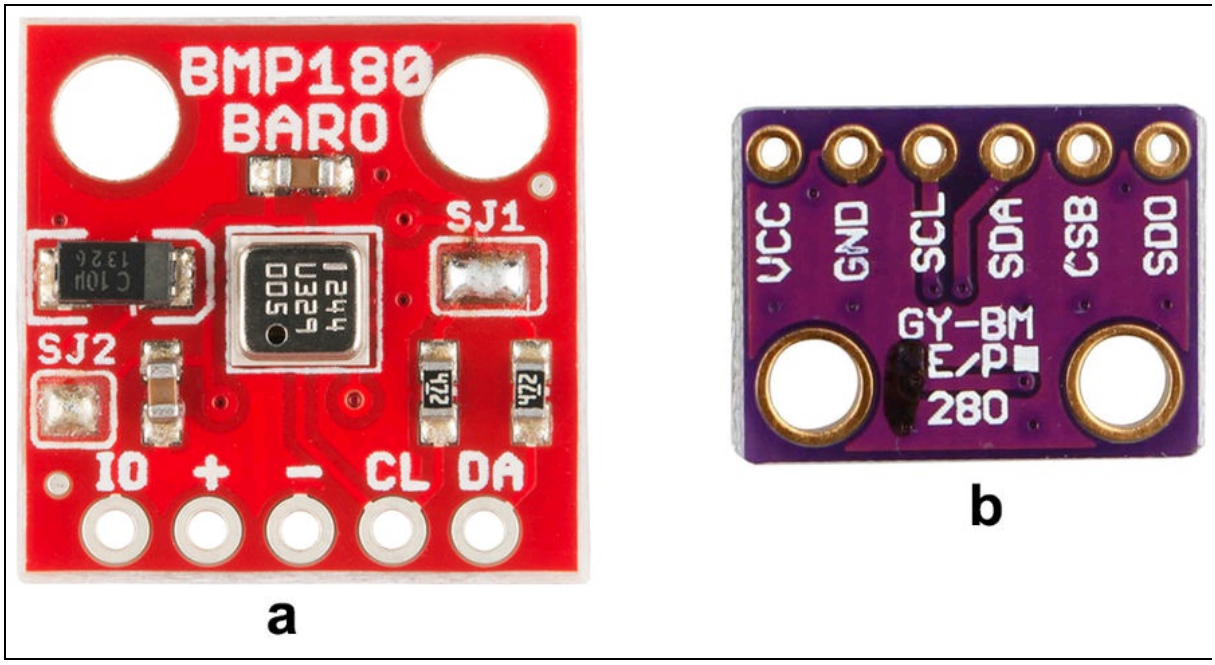


Figura 5.9 Sensore BMP180 (a). Sensore BME280 (b).

Il circuito

Il collegamento con Arduino è davvero molto semplice. Basta collegare i due fili dell'interfaccia I^2C di uno dei due moduli ai pin I^2C di Arduino. Ricordiamo che l'indicazione SDA e SCL è sul retro della scheda Arduino

La Figura 5.10 illustra il collegamento con il modulo BMP180, ma se si vuole usare il modulo BME280, basta collegare la porta I^2C nello

stesso modo.

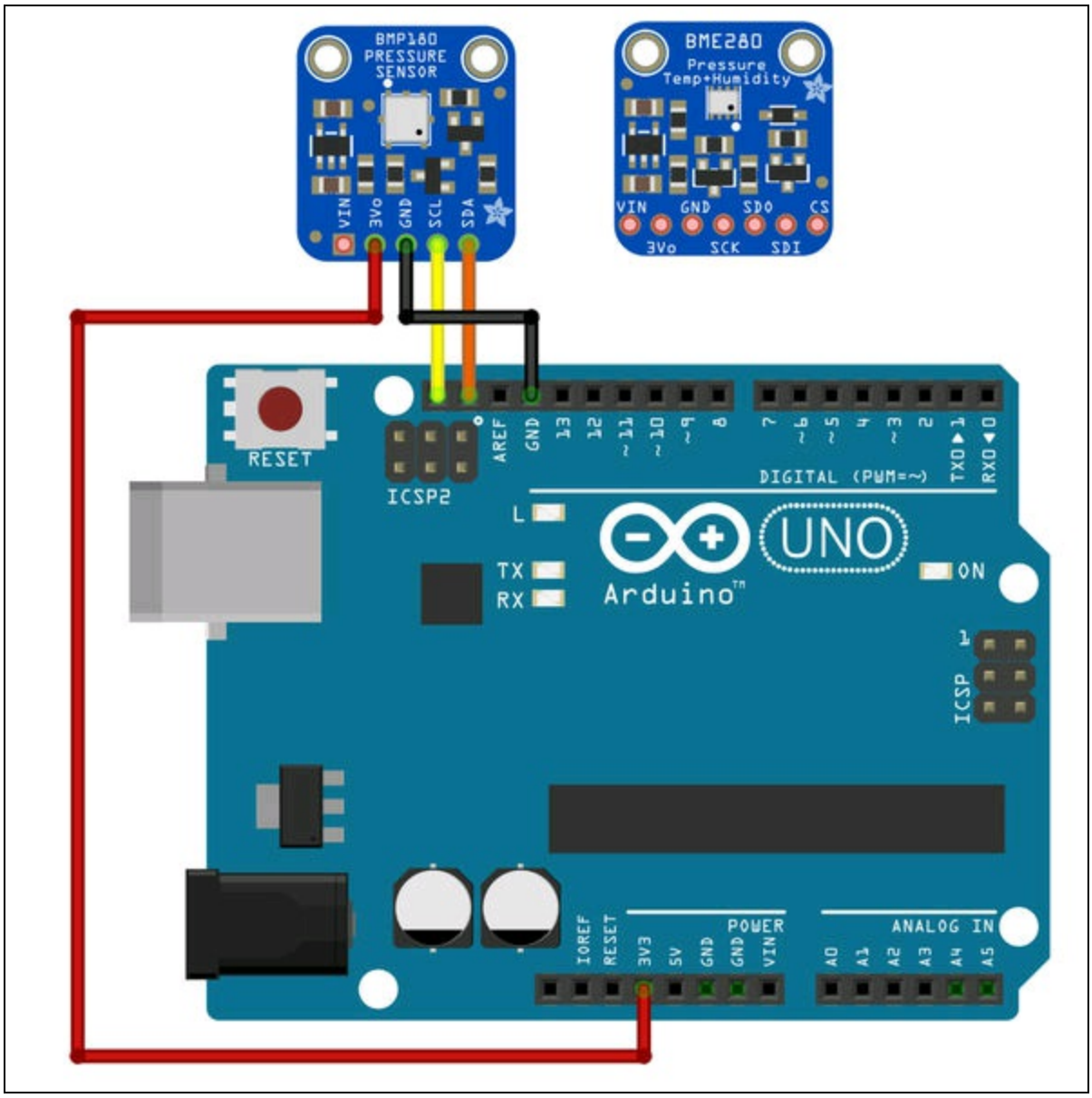


Figura 5.10 Collegamento del modulo BMP180 alla scheda Arduino.

Il codice

Entrambi i moduli di pressione hanno bisogno di librerie esterne, che vanno aggiunte all'IDE di Arduino come già spiegato. Le librerie per BMP180 e BME280 sono incluse nelle risorse del libro assieme agli

sketch di esempio, che sono completi del codice per la trasmissione seriale dei dati. Gli indirizzi da cui scaricare le librerie dal repository di SparkFun sono https://github.com/sparkfun/BMP180_Breakout_Arduino_Library e https://github.com/sparkfun/SparkFun_BME280_Arduino_Library

Aggiungere un anemometro

Per misurare la velocità del vento, lo strumento che serve è un anemometro. Ce ne sono di diversi tipi in commercio, più o meno costosi. Sarebbe bello (e forse più da maker) costruirne uno, stampando le pale in 3D e collegando il perno a un encoder alla sua base, per misurare la tensione e quindi la velocità del vento.

Siccome lo spazio è tiranno, spieghiamo soltanto come collegare un anemometro standard per Arduino, come quello fabbricato da Adafruit e disponibile all'indirizzo https://www.adafruit.com/product/1733 (Figura 5.11).

Figura 5.11 Anemometro di Adafruit.

Dato che l’alimentazione di riferimento dell’anemometro varia da 7 a 24 volt, basta collegare una pila (ne abbiamo usata una da 9 V) e misurare la tensione collegando l’uscita dell’anemometro a un ingresso analogico di Arduino.

La tensione d’uscita può variare da 0,4 V (0 m, assenza di vento) fino a 2,0 V (velocità del vento di 32,4 m/s), ovvero, secondo la scala di Beaufort (da Francis Beaufort, inventore della scala), si possono misurare le “forze” del vento da 1 (calma) a 12 (uragano). Come al solito, le tensioni all’ingresso analogico saranno solo dei numeri da 0 a 1023, per cui si dovranno convertire in valori anemometrici. Volendo visualizzare i valori della scala e una descrizione, si può fare riferimento alla seguente tabella.

Numero di Beaufort	Velocità del vento m/s	Descrizione
0	0	Calma
1	0,3 ÷ 1,5	Bava di vento
2	1,6 ÷ 3,4	Brezza leggera
3	3,5 ÷ 5,4	Brezza tesa
4	5,5 ÷ 7,9	Vento moderato
5	8,0 ÷ 10,7	Vento teso
6	10,8 ÷ 13,8	Vento fresco
7	13,9 ÷ 17,1	Vento forte
8	17,2 ÷ 20,7	Burrasca
9	20,8 ÷ 24,4	Burrasca forte
10	24,5 ÷ 28,4	Tempesta
11	28,5 ÷ 32,6	Tempesta violenta
12-17	>32,7	Uragano

Lo sketch è disponibile nelle risorse del libro, completo del codice per la trasmissione seriale dei dati.

Aggiungere un anemoscopio

Per completare la stazione meteo, oltre all'anemometro, si potrebbe pensare anche a un anemoscopio, per visualizzare la direzione del vento (Figura 5.12a). In questo caso, basta misurare la tensione analogica prodotta dalla torsione sull'asse dell'anemoscopio e da qui estrarre i valori di direzione Nord, Sud, Ovest, Est e tutte le direzioni intermedie. Si potrebbero anche visualizzare i nomi dei venti, come vengono chiamati nella classica rosa dei venti (Figura 5.12b).

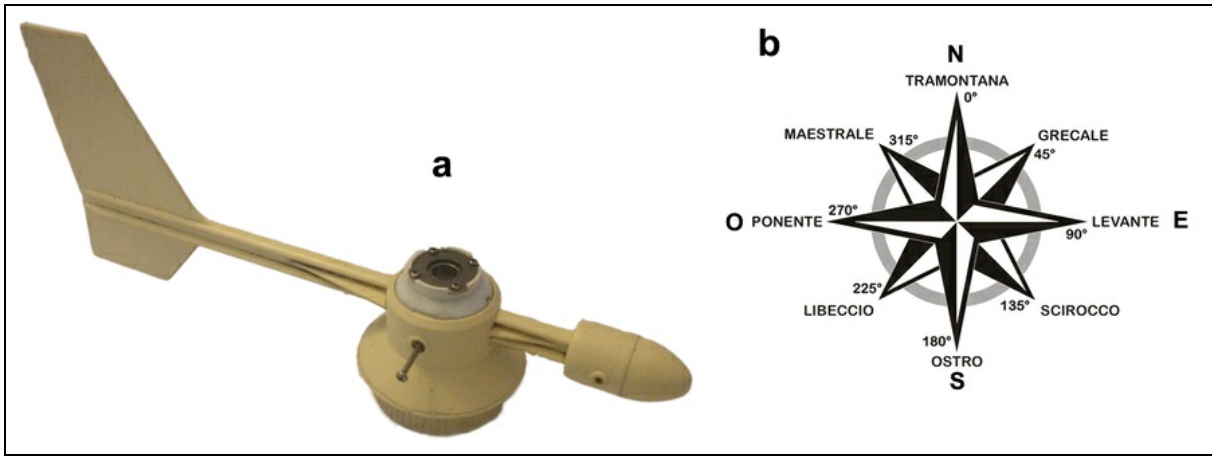


Figura 5.12 Esempio di anemoscopio (a). La rosa dei venti (b).

Irrigazione intelligente

Descrizione

Avere cura del proprio giardino talvolta potrebbe rivelarsi un’impresa difficoltosa. L’idea del giardino intelligente si basa sulla necessità di irrigare le care pianticelle di casa quando si è assenti, specialmente quando non si ha un parente o un vicino che se ne possa occupare mentre siamo in vacanza.

Questo semplice progetto prevede l’uso di una scheda Arduino che controlla un sistema di irrigazione che può essere posto all’interno o all’esterno, in base alle esigenze. Il progetto è pensato per essere posizionato in casa, anche se i sensori possono esser usati all’esterno senza problemi. Se si pensa di usare il circuito elettronico all’esterno, bisogna pensare a un contenitore impermeabile.

CODICE DI ESEMPIO

Tutti i file del progetto “Irrigazione intelligente” sono disponibili nelle risorse del libro presso il sito dell’autore all’indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Arduino UNO;
- 1× modulo relè;
- 1× sensore di umidità del terreno;
- 1× elettrovalvola idraulica;
- 1× alimentatore 12 VDC.

Sensore di umidità del terreno

A differenza del sensore di umidità relativa che abbiamo visto nel progetto precedente, il sensore di umidità del terreno permette di misurare la conducibilità elettrica del terreno, che può cambiare in base alla quantità dei sali minerali disciolti dall'acqua. Il sensore permette di sapere quanto secco o umido è il terriccio e quindi quanta acqua bisogna dare alla pianta.

Nella Figura 6.1 è illustrato un tipico sensore di umidità del terreno, venduto di solito con scheda breakout. Se il modello è passivo senza breakout funziona allo stesso modo.

Una volta alimentato il sensore con una tensione di 5 volt fornita dalla stessa scheda Arduino, è possibile rilevare la tensione di uscita del sensore stesso, che sarà più o meno alta in base alla conducibilità del terreno. In pratica, il sensore si comporta come una resistenza elettrica e quindi avremo questa situazione all'ingresso analogico:

- Valori bassi per terriccio con molta acqua;
- Valori elevati per terriccio secco.

I valori letti saranno, come al solito, una serie di byte a 10 bit, ovvero da 0 a 1023. Con una semplice funzione si potranno stabilire le

condizioni per controllare l’apertura e la chiusura del relè e quindi l’elettrovalvola collegata al circuito idraulico.

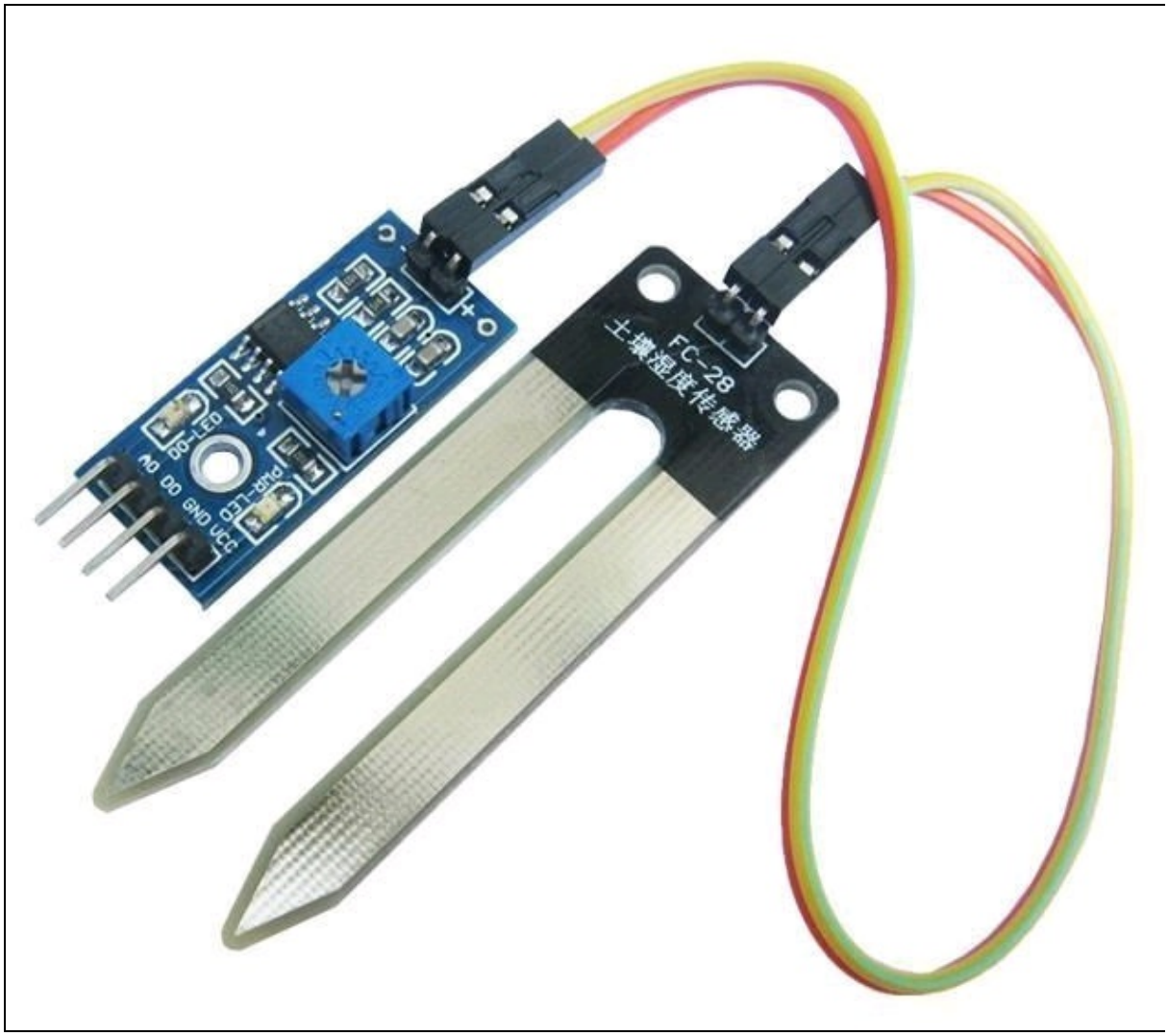


Figura 6.1 Sensore di umidità del terreno.

Elettrovalvola idraulica

In commercio ci sono elettrovalvole di tutti i tipi. Per il nostro progetto abbiamo identificato il prodotto raffigurato nella Figura 6.2, disponibile presso Futura Shop: http://www.futurashop.it/elettrovalvola-12-vdc-7300-elettrovalvola. Si tratta di un’elettrovalvola normalmente chiusa

con corpo in materiale plastico dotata di bobina a 12 V e attacchi filettati standard da 3/4”, adatta per l’interruzione del flusso d’acqua dell’impianto domestico.

La pressione minima di funzionamento è di 3 PSI (pari a circa 20 kPa) con la quale è garantito un flusso pari a circa 3 litri al minuto. L’assorbimento con la bobina in funzione è di circa 300 mA. Non ha un solenoide a gravità, per cui bisogna disporre di una pressione d’acqua sufficiente per aprire completamente la valvola.

A causa della tensione di alimentazione e dell’assorbimento, l’elettrovalvola non può essere controllata direttamente da Arduino, per cui si dovrà disporre di un modulo relè e di un alimentatore 12 VDC per l’elettrovalvola.

Se l’elettrovalvola va posta in giardino, bisogna pensare a un alloggiamento impermeabile per proteggere i suoi contatti. L’alimentatore va messo in casa, collegato a una presa a muro.

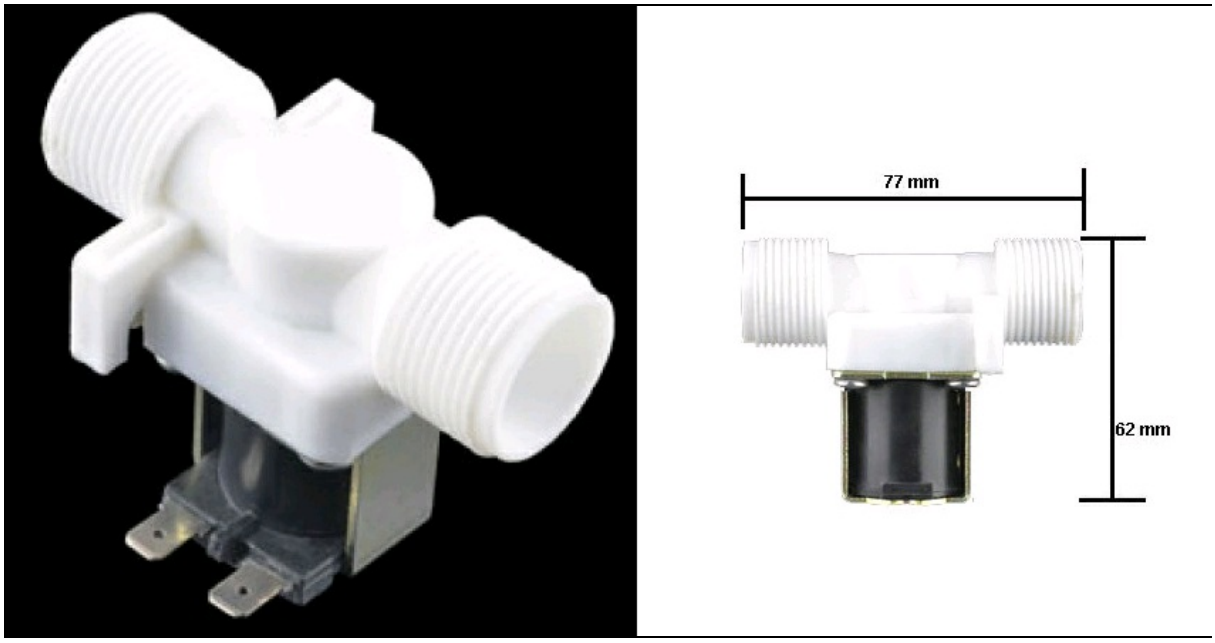


Figura 6.2 Elettrovalvola idraulica.

Modulo relè

Sono disponibili vari moduli relè appositamente studiati per Arduino. Possono essere dotati di 1, 2, 4, 8 o anche 16 relè. La Figura 6.3 illustra un modulo con un solo relè e un modulo con quattro relè.

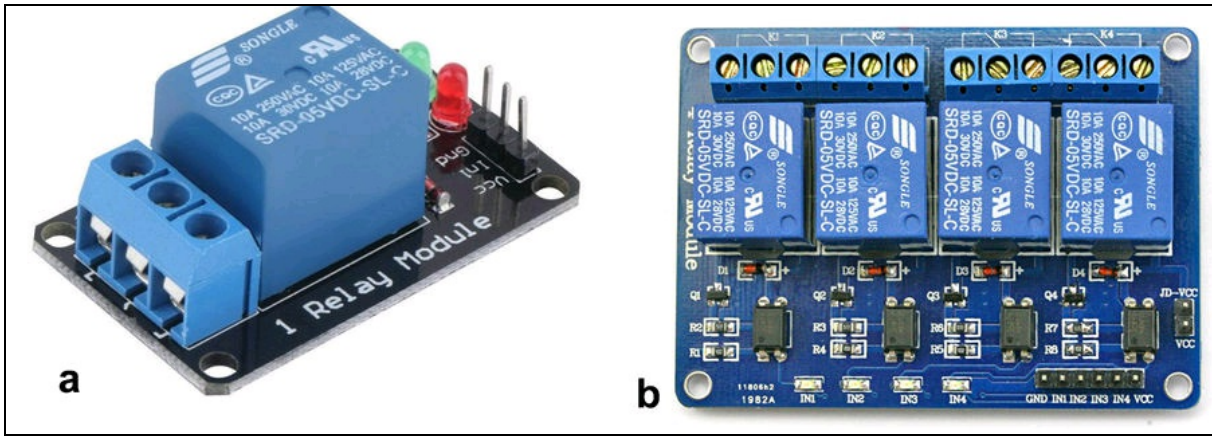


Figura 6.3 Modulo con un relè (a). Modulo con quattro relè (b).

Per semplificare il più possibile il nostro progetto, abbiamo previsto un solo relè per l'irrigazione di una stessa tipologia di piante. Nel caso servisse irrigare più tipologie di piante, basta replicare il numero di relè, di elettrovalvole e di sensori di umidità del terreno.

Il circuito elettrico

Per praticità, il prototipo di questo circuito usa una breadboard, ma per il progetto definitivo si possono collegare i pochi componenti con dei cavetti volanti e dei mammut. Se si usano moduli con più relè o più sensori, si dovrà optare per una millefori o un circuito stampato.

La Figura 6.4 illustra i collegamenti del sensore di umidità del terreno e del modulo relè alla scheda Arduino. Il relè è collegato al pin digitale 2. Il modulo del sensore di umidità del terreno è collegato all'ingresso analogico A0.

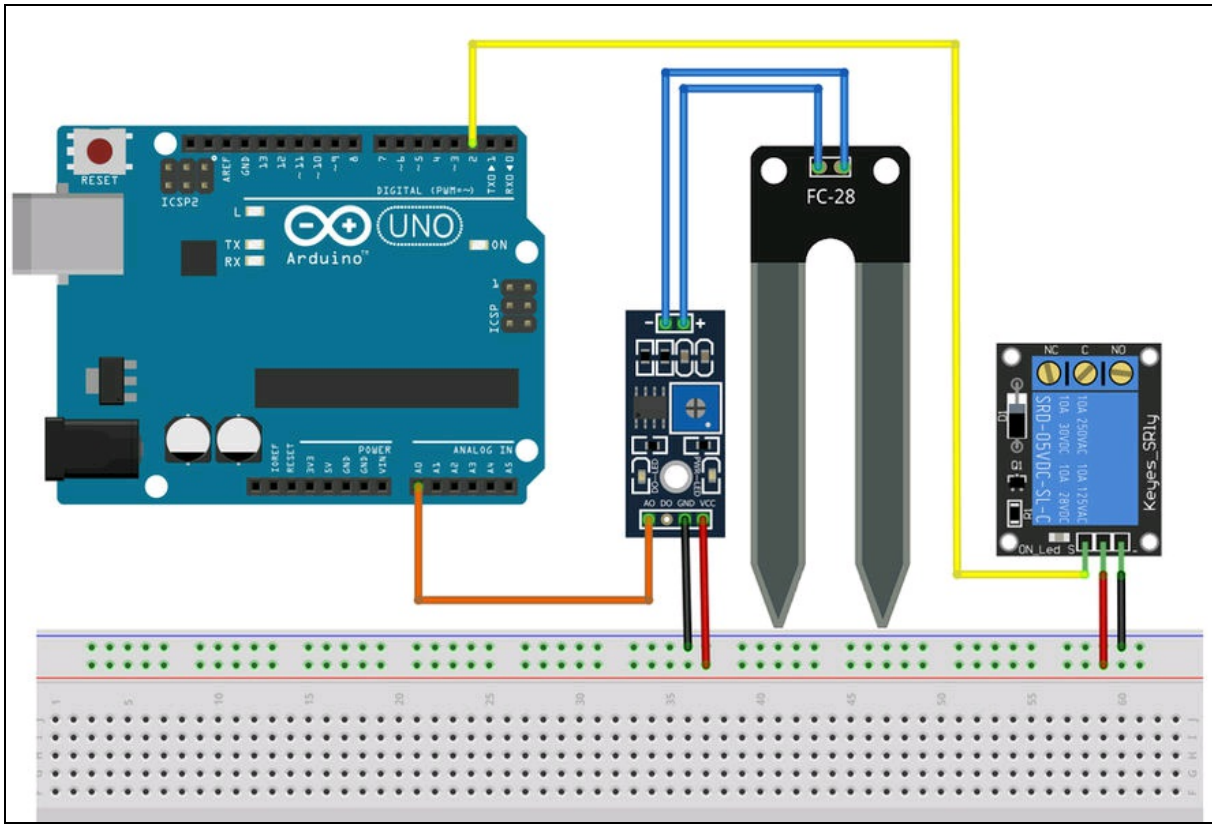


Figura 6.4 Schema di collegamento su breadboard.

Il circuito idraulico

Per il circuito idraulico da collegare all'elettrovalvola si possono usare dei tubi in PVC standard da ¾ di pollice, di quelli normalmente usati per l'impiantistica di casa fai da te. Il tubo va collegato a un rubinetto di casa o una valvola a saracinesca da giardino, che dovranno essere aperti solo al termine del montaggio dell'impianto. In base alla sensibilità della elettrovalvola si potrà dosare la potenza del flusso di entrata per avere un flusso di uscita né troppo violento né troppo scarso.

Sul mercato esistono tubi morbidi per impianti di irrigazione fai da te con tutti gli accessori come raccordi, boccole, guarnizioni, teflon e così via.

Per l'irrigazione si può usare un tubo forato opportunamente per annaffiare uno o più vasi. Ovviamente, il tubo andrà chiuso con un tappo nella parte terminale.

La Figura 6.5 illustra come collegare l'uscita del relè all'alimentatore 12 VDC e all'elettrovalvola. L'interruzione dell'alimentazione va fatta solo su un cavo, collegandolo alla presa NO (*Normally Open*) e C (*Common*) del relè. L'altro cavo di alimentazione va direttamente all'elettrovalvola.

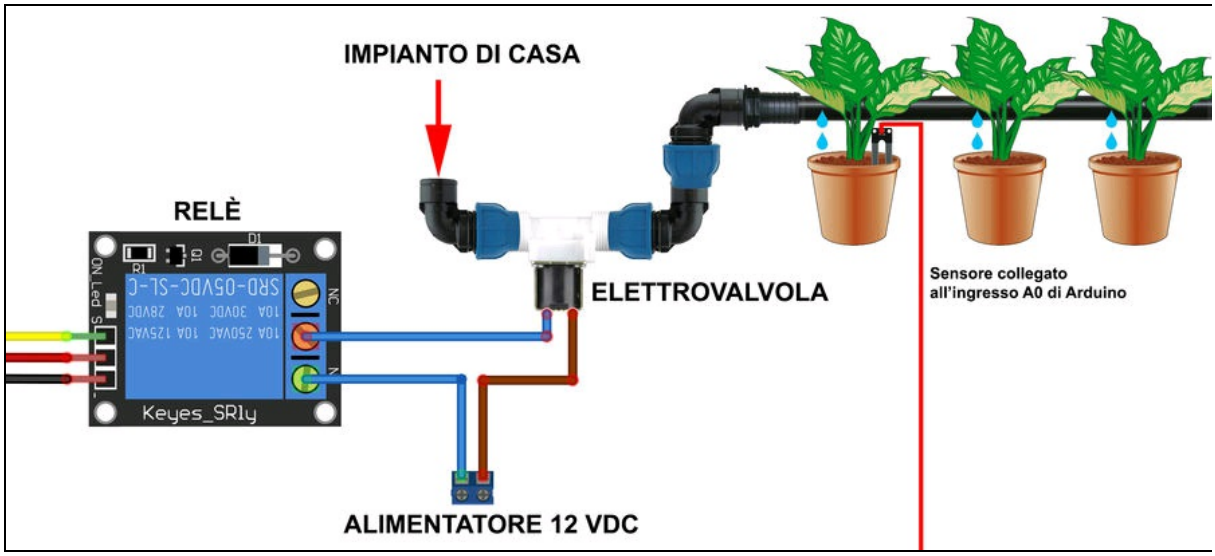


Figura 6.5 Schema di collegamento idraulico.

ATTENZIONE

Onde evitare allagamenti e problemi di altro tipo, se non si è portati all'impiantistica di questo tipo, chiedere aiuto a un idraulico.

Il codice

Lo sketch per Arduino prevede il rilevamento della tensione analogica all'ingresso A0, fornita dal sensore di umidità, e il controllo del pin digitale 2 di uscita del relè. Nel caso si volessero usare più sensori e più relè, è sufficiente adattare lo sketch al numero di relé desiderato.

Innanzitutto va operata una calibrazione del sensore di umidità per ottenere i migliori risultati. Questa va eseguita in modo totalmente empirico. Si consiglia di introdurre completamente il sensore in un vaso di terriccio asciutto, dello stesso tipo che si intende irrigare e osservare il valore rilevato nel monitor seriale con un piccolo sketch di test (disponibile nelle risorse del libro). Lo sketch rileva il dato all'ingresso analogico A0 e stampa il valore corrispondente nel monitor seriale.

```
int sensorPin = A0;
int sensorValue = 0;
void setup() {
    Serial.begin(9600);
}
void loop() {
    sensorValue = analogRead(sensorPin);
    Serial.print("Valore: ");
    Serial.println(sensorValue);
    delay(1000);
}
```

Con il terriccio asciutto, annotare il valore iniziale stampato nel monitor seriale. Quindi, versare un po' d'acqua per rendere il terriccio leggermente umido e annotare il valore letto nel monitor seriale. Infine, versare ancora dell'acqua per rendere il terriccio ben irrigato e annotare il valore. I valori annotati saranno quelli di riferimento per una corretta irrigazione.

La Figura 6.6 illustra le tre fasi con una piantina appena germogliata di Pothos, una pianta da interno che bisogna irrigare almeno una volta alla settimana. Anche se è abbastanza resistente, è meglio non farla soffrire quando si va in vacanza per una settimana o due.

- Con il terriccio asciutto (Figura 6.6a) il monitor seriale mostra un valore del sensore intorno a 460-470.
- Con il terriccio inumidito con mezzo bicchiere d’acqua (Figura 6.6b) il monitor seriale mostra un valore del sensore intorno a 180-200.
- Con un altro mezzo bicchiere d’acqua il terriccio è decisamente ricco d’acqua (Figura 6.6c) e il monitor seriale mostra un valore del sensore intorno a 70-80.

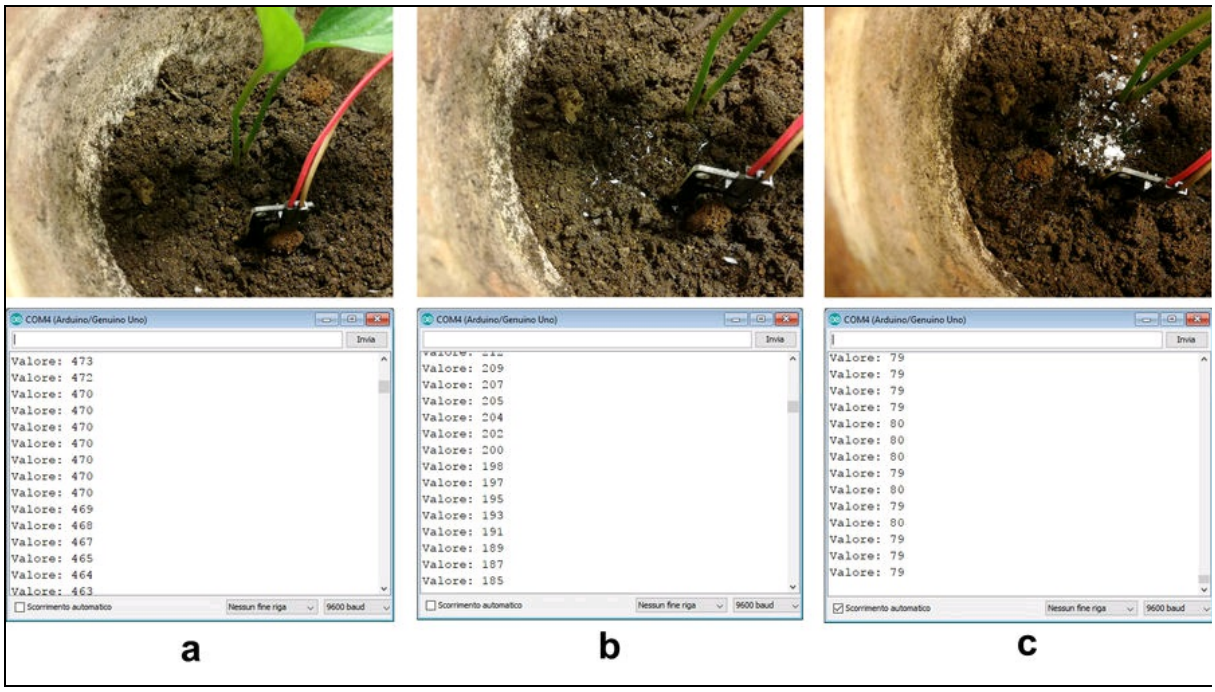


Figura 6.6 Le condizioni di irrigazione con il sensore di umidità nel terriccio asciutto (a), umido (9b) e ricco d’acqua (c).

Questo ci fa capire quando e quanto dobbiamo irrigare il vaso, per cui bisogna modificare lo sketch in modo che attivi il relè per il tempo necessario ad aprire l’elettrovalvola.

Anche qui si dovrà effettuare un test empirico. Si consiglia di svolgere il test con un tubo forato in un lavandino e non direttamente sulle piante.

A seconda delle situazioni si dovrà attivare il relè per ottenere un flusso d’acqua proporzionale al fabbisogno della pianta o di tutto il giardino.

Codice commentato

Allo sketch precedente, oltre alla variabile *relay* per controllare il relè collegato al pin digitale 2, è stata aggiunta la variabile booleana *flag* per controllare l’azione di apertura e chiusura del relè.

```
int relay = 2;
int sensorPin = A0;
boolean flag = false; // flag per l'azione
void setup() {
    Serial.begin(9600);
    pinMode(relay, OUTPUT);
}
```

Nella funzione `loop()` ci sono le istruzioni `if` per il controllo del relè, impostate in base alle letture effettuate nel test precedente.

Ovviamente, si dovrà scegliere una sola condizione di irrigazione e mettere in commento le altre. Per esempio, se si vuole irrigare la pianta quando è il terriccio è molto asciutto, si dovrà togliere il commento dalla sezione corrispondente, ovvero quando il valore del sensore è fra 500 e 800.

```
void loop()
{
    int sensorValue = analogRead(sensorPin);
    Serial.print("Valore: ");
    Serial.println(sensorValue);
    if (sensorValue > 900)  // terriccio secco
    {
        //if (flag==false)
        //{
            //  digitalWrite(relay, HIGH);
            //  delay (5000);  // apre l'elettrovalvola per 5000 ms
            //  digitalWrite(relay, LOW);
            //  flag=true;
        //}
    }
    else if (sensorValue >= 500 && sensorValue < 800)  // terriccio molto asciutto
    {
        if (flag==false)
        {
            digitalWrite(relay, HIGH);
            delay (3000);  // apre l'elettrovalvola per 3000 ms
            digitalWrite(relay, LOW);
```

```
            flag=true;
        }
    }
    else if (sensorValue >= 400  && sensorValue < 500)  // terriccio asciutto
    {
        //if (flag==false)
        //{
            //  digitalWrite(relay, HIGH);
            //  delay (1500);  // apre l'elettrovalvola per 1500 ms
            //  digitalWrite(relay, LOW);
            //  flag=true;
        //}
     }
    else if (sensorValue >= 200 && sensorValue < 400)  // terriccio umido
    {
        //if (flag==false)
        //{
            //  digitalWrite(relay, HIGH);
            //  delay (500);  // apre l'elettrovalvola per 500 ms
            //  digitalWrite(relay, LOW);
            //  flag=true;
        //}
    }
    delay(1000);
}
```

Come funziona

All’inizio del codice, la variabile booleana `flag` viene posta a `false`, per cui, con l’istruzione `digitalWrite(relay, HIGH)`, si attiverà il relè per 3000 millisecondi, che a sua volta attiverà l’elettrovalvola per 3 secondi di irrigazione. Contemporaneamente, la variabile `flag` viene posta a `true`, per cui il relè non si attiverà più fino a quando non si farà ripartire Arduino. In questo modo la pianta verrà irrigata a sufficienza per il ritorno dalle vacanze.

Se invece di riavviare manualmente Arduino, si vuole aggiungere un timer che riporti la variabile `flag` a `false` e permettere di riattivare l’irrigazione automatica dopo un intervallo di tempo predeterminato, bisogna dotare Arduino di un clock in tempo reale oppure di una connessione Internet a un server NTP.

Questa opzione può essere facilmente implementata usando un modulo Wi-Fi ESP8266 qualsiasi o usando direttamente la scheda

Arduino UNO Wi-Fi. Si veda il progetto “Monitoraggio meteo” per implementare un server NTP nello sketch.

Serratura con impronta digitale

Descrizione

Ci sono serrature e serrature. Quando si vuole ottenere il massimo della sicurezza, probabilmente una serratura che si apre con l'impronta digitale è la soluzione migliore e anche un po' "cool".

Questo progetto si basa su un sensore molto diffuso nell'ambiente Arduino, tanto che esiste un libreria dedicata per facilitarne l'uso. Oltre al sensore per impronte digitali servirà collegare un modulo relè per poter far "scattare" un serratura alimentata dalla rete elettrica. Ovviamente servirà anche una serratura, se non è già montata sulla porta. Infine, un display LCD potrà fare le veci del monitor seriale, dato che il sistema di apertura dovrà essere gestito autonomamente e montato all'esterno della porta.

CODICE DI ESEMPIO

Tutti i file del progetto "Serratura con impronta digitale" sono disponibili nelle risorse del libro presso il sito dell'autore all'indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Arduino UNO;
- 1× sensore Fingerprint Scanner TTL GT-511C3;
- 1× display LCD 16x2;
- 1× trimmer da 10 KΩ;
- 1× resistore 150 Ω;
- 1× resistore 330 Ω;
- 1× resistore 220 Ω;
- 1× modulo relè;
- 1× serratura elettrica;
- 1× LED.

Sensore Fingerprint Scanner TTL GT-511C3

Questo sensore permette di memorizzare fino a 200 impronte digitali. Consente il riconoscimento e il confronto fra un'impronta letta e quella memorizzata nel suo database interno. La comunicazione con Arduino avviene tramite UART e tutti i comandi sono facilitati grazie all'uso di una libreria dedicata.

Caratteristiche principali.

- Identificazione delle impronte digitali ad alta velocità e precisione.
- Algoritmo usato: SmackFinger 3.0.
- Possibilità di scaricare le immagini delle impronte digitali dal dispositivo.
- Legge e scrive i modelli delle impronte digitali nel database.
- Protocollo UART (default 9600 baud).

- Capacità di verifica 1:1 e identificazione 1:N.
- Alimentazione da 3,3 a 6 V.
- LED azzurro commutabile On/Off via software per la retroilluminazione della superfice sensibile.
- Il sensore è dotato di due alette laterali per il fissaggio a vite.

Al momento dell'acquisto del sensore, bisogna accertarsi che venga venduto con il cavo di collegamento. Se non è incluso, bisogna ordinarlo a parte. Il connettore è di tipo JST SH a 4 fili.

Collegamento del sensore Fingerprint Scanner

La Figura 7.1 illustra il sensore, il suo connettore JST SH a 4 fili e la piedinatura. L'alimentazione è compresa fra 3,3 e 6 V quindi può essere collegato direttamente al pin 5 V di Arduino. I pin TX e RX vanno collegati a due pin digitali di Arduino e non alla porta seriale TX/RX. Questo perché la libreria del sensore si appoggia alla libreria SoftwareSerial per facilitare il debug sul monitor seriale.

La libreria per il sensore Fingerprint Scanner TTL GT-511C3 è disponibile come risorsa del libro, ma è scaricabile direttamente dal repository di SparkFun: `https://github.com/sparkfun/Fingerprint_Scanner-TTL`. Per l'installazione della libreria nell'IDE di Arduino, si veda il box *Come aggiungere una libreria all'IDE di Arduino* del progetto "Monitoraggio meteo".

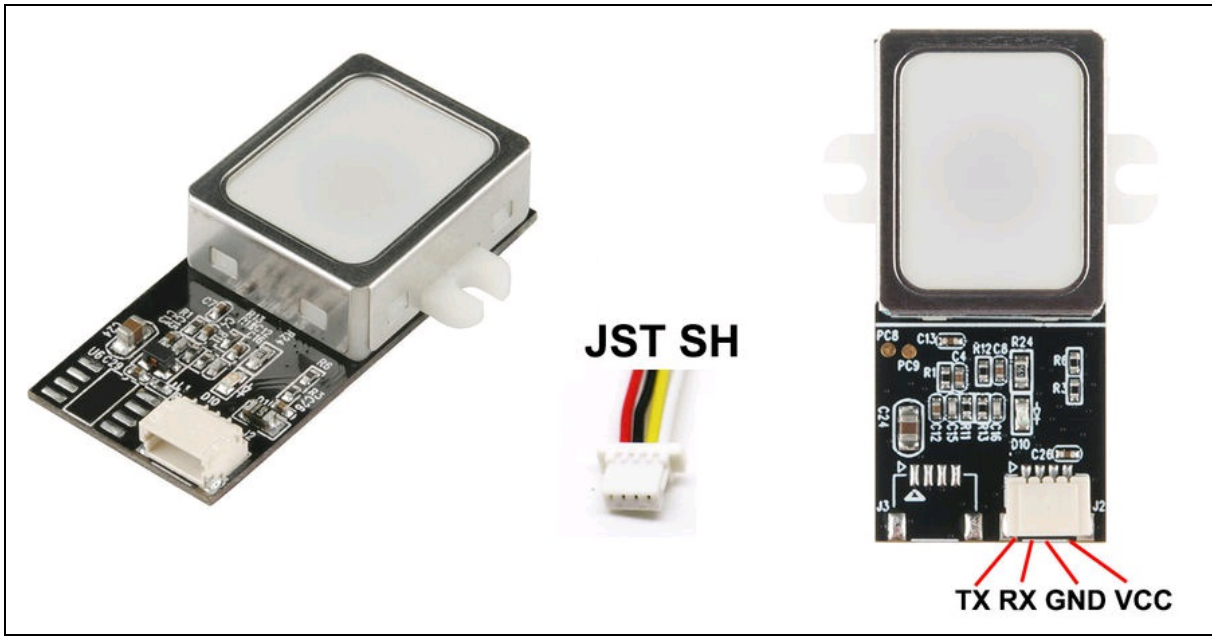


Figura 7.1 Il sensore Fingerprint Scanner TTL GT-511C3, il connettore JST SH e la piedinatura.

Il circuito elettrico

Il progetto su breadboard è visibile nella Figura 7.2. Come si può vedere, viene impiegato anche un display LCD 16×2 caratteri per poter utilizzare il dispositivo autonomamente. Si consiglia di assemblare il progetto definitivo su una millefori o un circuito stampato.

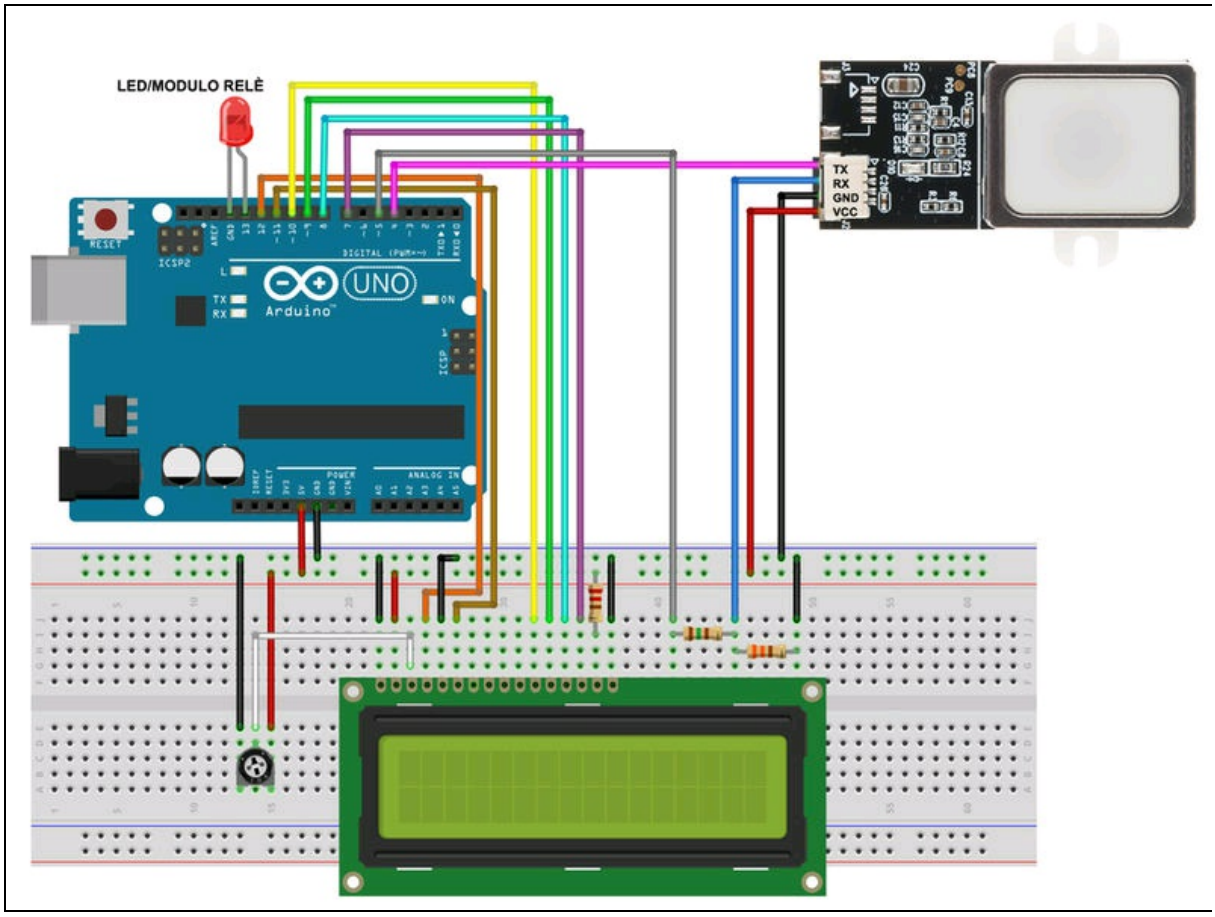


Figura 7.2 Schema di collegamento su breadboard.

Il collegamento del display è leggermente differente dal solito, per permettere il collegamento del sensore alle porte 4 e 5 di Arduino. Il trimmer serve unicamente per regolare il contrasto del display. Un resistore da 220 Ω collegato all'anodo della retroilluminazione del display limita l'assorbimento di corrente.

Il LED collegato al pin digitale 13 serve per il test, ma potrà essere sostituito da un modulo relè che controllerà l'apertura della serratura elettrica.

Il partitore di tensione creato dai due resistori da 150 e 330 Ω serve a regolare la tensione d'uscita del pin TX di Arduino. Siccome il livello TTL di Arduino è 5 volt, si rischia di bruciare la porta TTL a 3,3 volt del sensore. Nei commenti contenuti negli esempi della libreria del sensore viene suggerito l'uso di un partitore con 560 e 1000 Ω, ma abbiamo verificato che non funziona, perché la caduta di tensione è eccessiva. L'uscita del partitore dovrebbe aggirarsi sui 3,3 volt. Si consiglia di effettuare una misurazione con il multimetro. Sembra che questo valore sia abbastanza critico, per cui vale la pena di trovare la coppia di resistori adeguata, misurando i valori effettivi con un ohmetro. Per il calcolo del partitore di tensione, si veda il box dedicato.

Calcolo del partitore di tensione

Un partitore di tensione resistivo è un circuito costituito da due resistori in serie, ai capi dei quali viene applicata una tensione (Vin). La tensione verrà ripartita nei due rami, in base alla legge di Ohm. Di solito serve per abbassare la tensione di 5 V a 3,3 V per adeguare la compatibilità TTL di due circuiti. La relazione che lega le resistenze e la tensione di un partitore di tensione è la seguente:

$$Vout = (R2 / R1 + R2) \times Vin.$$

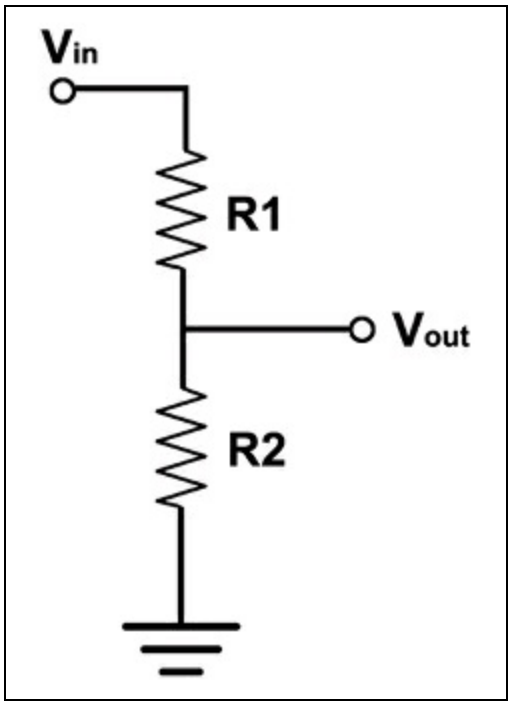
Esempio:

R1 = 150 Ω

R2 = 330 Ω

Vin = 5 V

Vout = (330 / 150 + 330) x 5 = 3,3÷3,4 volt (circa, a seconda della tolleranza dei resistori).



Contenitore

Per il contenitore da fissare alla porta che possa contenere il display e la scheda Arduino si può fare una ricerca su `thingiverse.com` o altri siti simili per trovare un file STL di stampa 3D. La soluzione ideale sarebbe quella di progettare il contenitore anche per fissare il sensore e di stamparlo in 3D (Figura 7.3).

Figura 7.3 Il contenitore per il display e il sensore.

Serratura elettrica

Se non è già installata sulla porta, bisogna acquistarla presso un negozio specializzato e se non si è portati al fai da te, è meglio farsi aiutare da un fabbro.

A seconda del livello di sicurezza desiderato, si può scegliere una serratura con uno o più chiavistelli, più o meno profondi (Figura 7.4a). Non è importante il tipo di alimentazione, ovvero se è a 12 volt DC o 230 volt dalla rete elettrica. La serratura viene attivata da un relè collegato ad Arduino, come illustrato nella Figura 7.4b. Il modulo relè viene alimentato a 5 V da Arduino.

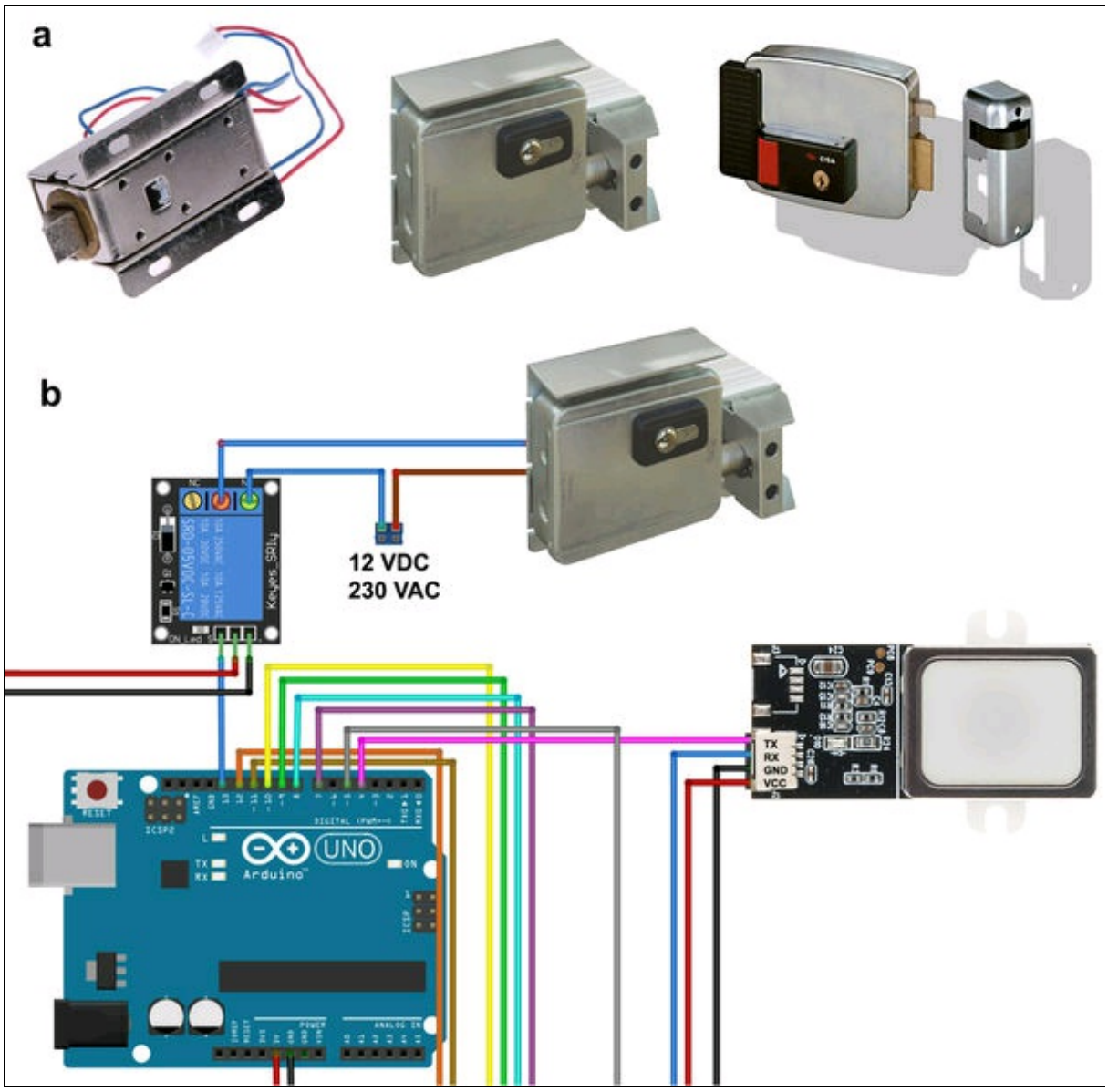


Figura 7.4 Vari tipi di serrature (a). Schema di collegamento del modulo relè alla serratura (b).

Il codice

Una volta installata la libreria FPS_GT511C3 per il sensore, si possono aprire i suoi sketch di esempio. Questi esempi possono essere impiegati per la lettura e la memorizzazione delle impronte digitali e per la loro successiva verifica.

Esempio 1

Per vedere se tutto funziona, il primo esempio da usare si chiama *FPD_Enroll*, che serve al cosiddetto *enrollment*, traducibile nel nostro caso come “registrazione” delle impronte digitali nella memoria del sensore.

All’inizio dello sketch vengono incluse le librerie `FPS_GT511C3` e `SoftwareSerial`. L’autore della libreria ha preferito usare SoftwareSerial per creare una porta UART autonoma usando i pin digitali 4 e 5 di Arduino e lasciare libera la porta UART standard per il monitor seriale. Con l’istruzione `FPS_GT511C3 fps(4, 5)` viene creato l’oggetto `fps` che comunicherà con la porta UART del sensore sui pin RX e TX. Ovviamente si potrebbero usare due pin qualsiasi, ma dovendo usare anche un display, abbiamo preferito spostare i pin di collegamento di quest’ultimo (si veda più avanti).

```
#include "FPS_GT511C3.h"
#include "SoftwareSerial.h"
    FPS_GT511C3 fps(4, 5);
```

Nella funzione `setup()` viene inizializzato il monitor seriale per la stampa delle istruzioni. Il sensore viene inizializzato con l’istruzione `fps.Open()`.

Con l’istruzione `fps.SetLED` si può accendere (con `true`) o spegnere (con `false`) il LED della retroilluminazione del sensore. Infine, viene chiamata

la funzione `Enroll()`.

```
void setup()
{
    Serial.begin(9600);
    delay(100);
    fps.Open();
    fps.SetLED(true);
    Enroll();
}
```

La funzione `Enroll()` controlla il sensore e mostra sul monitor i vari passaggi per la memorizzazione delle impronte digitali. Qui di seguito commentiamo solo le parti essenziali del codice che stampa le istruzioni del processo di enrollment sul monitor seriale.

```
void Enroll()
{
        …
```

Una volta aperto il monitor seriale, la prima istruzione è “Press finger to Enroll #” e il numero è `0`, visto che è la prima volta che si usa il sensore. Significa “premere un dito per la registrazione numero 0”. Di solito si preme l’indice, ma va bene qualsiasi altro dito.

```
Serial.print("Press finger to Enroll #");
Serial.println(enrollid);
while(fps.IsPressFinger() == false) delay(100);
bool bret = fps.CaptureFinger(true);
int iret = 0;
```

Se l’impronta viene letta correttamente, apparirà “Remove finger” (rimuovere il dito) e l’impronta verrà memorizzata con l’ID 0.

```
if (bret != false)
{
    Serial.println("Remove finger");
    fps.Enroll1();
    while(fps.IsPressFinger() == true) delay(100);
```

Quindi apparirà “Press same finger again” (premere ancora lo stesso dito) che serve a confrontare l’impronta acquisita.

```
Serial.println("Press same finger again");
while(fps.IsPressFinger() == false) delay(100);
bret = fps.CaptureFinger(true);
```

Ancora una volta apparirà “Remove finger” (rimuovere il dito).

```
if (bret != false)
{
    Serial.println("Remove finger");
    fps.Enroll2();
```

```
    while(fps.IsPressFinger() == false) delay(100);
    bret = fps.CaptureFinger(true);
```

A questo punto apparirà “Enrolling Successfull” (registrazione riuscita).

```
Serial.println("Enrolling Successfull");
```

Altrimenti apparirà “Enrolling Failed with error code”, con il numero dell’errore.

```
Serial.print("Enrolling Failed with error code:");
Serial.println(iret);
```

Riavviando Arduino con il tasto di reset, si può procedere alla registrazione di altre impronte, che verranno numerate progressivamente. La Figura 7.5a illustra il processo di registrazione dell’impronta con ID 0 e dell’impronta con ID 1 (Figura 7.5b).

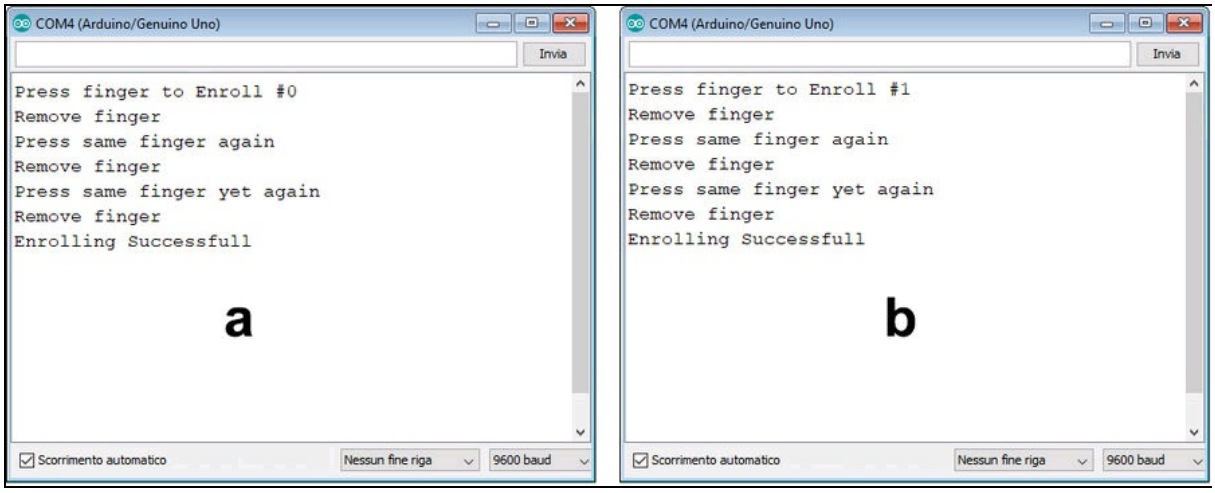


Figura 7.5 Processo di registrazione dell’impronta 0 (a) e dell’impronta 1 (b).

Esempio 2

Il secondo esempio della libreria si chiama *FPS IDFinger* e spiega come riconoscere le impronte memorizzate. Lo sketch rimane identico al precedente fino alla funzione `loop()` che contiene il codice per l’identificazione delle impronte. Aprendo il monitor seriale si vedrà la stampa della frase “Please press finger” (premere un dito prego).

Appoggiando sul sensore il primo dito memorizzato, apparirà “Verified ID”: e il numero ID corrispondente al dito. Se il dito non è stato memorizzato, apparirà “Finger not found” (dito non trovato).

```
void loop() {
    if (fps.IsPressFinger()) {
        fps.CaptureFinger(false);
        int id = fps.Identify1_N();
        if (id <200) {
            Serial.print("Verified ID:");
            Serial.println(id);
        }
        else {
            Serial.println("Finger not found");
        }
    }
    else {
        Serial.println("Please press finger");
    }
    delay(100);
}
```

La Figura 7.6a illustra il processo di identificazione del primo dito avvenuta con successo, mentre la Figura 7.6b mostra il dito non riconosciuto.

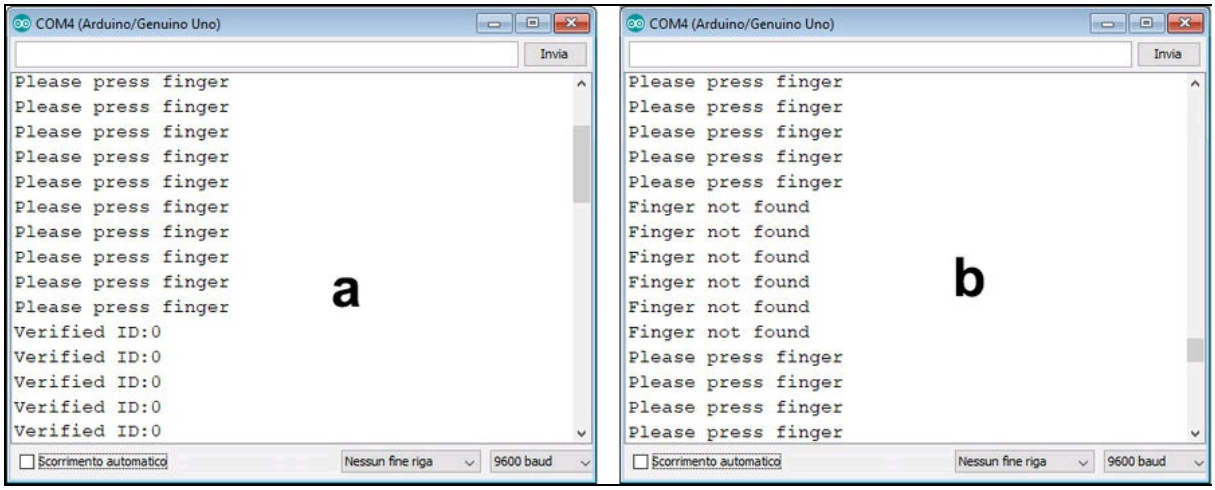


Figura 7.6 Processo di identificazione del primo dito (a). Dito non riconosciuto (b).

Uso del display

L’implementazione di un display rende sicuramente tutto più comodo. Il codice che abbiamo prodotto per questo progetto non è molto

differente dagli esempi. È stato aggiunto il codice per la gestione del display LCD 16×2 caratteri, di cui abbiamo parlato nel Capitolo 1. La libreria per la gestione del display è già presente nell'IDE di Arduino come libreria standard, per cui basta solo aggiungere le istruzioni di stampa sul display al posto di quelle per la stampa sul monitor seriale.

Codice commentato

Essendo il listato del nostro progetto `IMPRONTA_DIGITALE` identico all'esempio `FPD_Enroll`, verranno evidenziate solo le parti inerenti all'uso del display.

All'inizio dello sketch, oltre alle librerie `FPS_GT511C3` e `SoftwareSerial`, viene inclusa anche la libreria `LiquidCrystal`. Con questa libreria viene creata l'istanza `lcd(12,11,10,9,8,7)` che si riferisce al collegamento fisico dei piedini del display ai pin digitali di Arduino. Come già detto, il collegamento che di solito viene usato dalla libreria LiquidCrystal è stato modificato per lasciare posto ai pin 4 e 5 usati dalla libreria del sensore. Quindi, chi vuole provare gli esempi della libreria LiquidCrystal tenga in considerazione questo collegamento diverso.

```
#include "FPS_GT511C3.h"
#include "SoftwareSerial.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
    FPS_GT511C3 fps(4, 5);
```

Nella funzione `setup()` viene inizializzato il display su 16 caratteri e 2 righe con l'istruzione `lcd.begin(16, 2)`. Quindi viene posizionato il cursore del display sul primo carattere e sulla prima riga con l'istruzione `lcd.setCursor(0, 0)`. Infine, con l'istruzione `lcd.print("FingerPrint Scan")` viene stampata sul display la scritta "FingerPrint Scan", che scomparirà dopo 2 secondi. Come nell'esempio originale, viene chiamata la funzione `Enroll()`.

```
void setup()
{
```

```
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);
    lcd.print("FingerPrint Scan");
    delay(100);
    fps.Open();
    fps.SetLED(true);
    delay(2000);
    Enroll();
}
```

Dato che la funzione `Enroll()` rimane identica, verranno evidenziate solo le parti di stampa sul display. Si fa notare che viene usata l'istruzione `lcd.clear()` per pulire il display all'inizio di ogni operazione. Per comodità le frasi sono state tradotte in italiano. Quando la frase supera i 16 caratteri viene divisa su due righe. In questo caso si usa l'istruzione `lcd.setCursor(0, 1)` per iniziare dal primo carattere della seconda riga.

```
void Enroll()
{
        …
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato "Premere il dito N".

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Premere dito N");
lcd.setCursor(0, 1);
lcd.print(enrollid);
    ….
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato "Rimuovi il dito".

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Rimuovi il dito");
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato "Premi lo stesso", quindi viene posizionato il cursore sul primo carattere della seconda riga e stampato "dito".

```
lcd.clear();
lcd.setCursor(0, 0);
```

```
lcd.print("Premi lo stesso ");
lcd.setCursor(0, 1);
lcd.print("dito");
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato “Rimuovi il dito”.

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Rimuovi il dito");
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato “Premi lo stesso”, quindi viene posizionato il cursore sul primo carattere della seconda riga e stampato “dito ancora”.

```
lcd.print("Premi lo stesso");
lcd.setCursor(0, 1);
lcd.print("dito ancora");
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato “Rimuovi il dito”.

```
lcd.clear();
lcd.print("Rimuovi il dito");
```

Con le seguenti istruzioni viene pulito lo schermo, posizionato il cursore sul primo carattere della prima riga e stampato “Registrazione”, quindi viene posizionato il cursore sul primo carattere della seconda riga e stampato “riuscita!”.

```
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Registrazione");
lcd.setCursor(0,1);
lcd.print("riuscita! ");
```

Se qualcosa non va, viene stampato il numero di errore.

```
lcd.clear();
lcd.print("Errore n. ");
lcd.print(iret);
```

Per la registrazione di più impronte digitali basta riavviare Arduino con il tasto di reset e iniziare da capo la procedura. Nella Figura 7.7 trovate una serie di immagini reali del display in funzione.

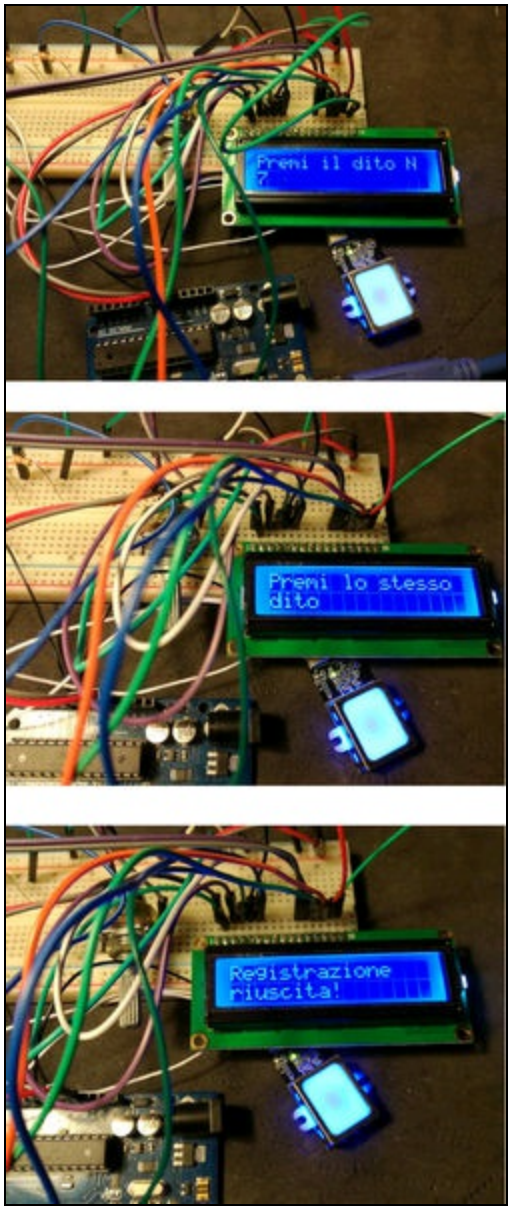


Figura 7.7 Alcuni passaggi del processo di registrazione delle impronte digitali.

Uso del relè

Il nostro progetto `IMPRONTA_DIGITALE_LOCK` ricalca in gran parte l'esempio `FPS IDFinger` della libreria. È stata aggiunta la gestione del display e la sezione di accensione del LED e/o l'attivazione del modulo relè.

Codice commentato

All’inizio dello sketch l’unica aggiunta riguarda la variabile `led` impostata sul pin digitale 13, che può essere usato anche per il modulo relè.

```
#include "FPS_GT511C3.h"
#include "SoftwareSerial.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
FPS_GT511C3 fps(4, 5);
byte led = 13;
```

Nella funzione `setup()` viene inizializzato il display su 16 caratteri e due 2 righe con l’istruzione `lcd.begin(16, 2)`. Quindi viene posizionato il cursore del display sul primo carattere e sulla prima riga con l’istruzione `lcd.setCursor(0, 0)`. Infine, con l’istruzione `lcd.print("FingerPrint Lock")` viene stampata sul display la scritta “FingerPrint Lock” che rimarrà sempre visibile.

```
void setup()
{
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);
    lcd.print("FingerPrint Lock");
    pinMode(led, OUTPUT);
    delay(100);
    fps.Open();
    fps.SetLED(1);
}
```

Nella funzione `loop()` viene effettuata la lettura del sensore con l’istruzione `fps.IsPressFinger()` e tentata l’identificazione con l’istruzione `fps.Identify1_N()`.

```
void loop()
{
    if (fps.IsPressFinger())
    {
        fps.CaptureFinger(false);
            int id = fps.Identify1_N();
```

Quindi se il numero di ID rilevato è inferiore a 200, viene stampata sulla seconda riga del display la scritta “Verificato:” e il numero di ID.

```
if (id < 200)
{
    lcd.setCursor(0,1);
```

```
    lcd.print("Verificato:   ");
    lcd.print(id);
```

Contemporaneamente viene attivato il LED per un tempo di 2 secondi. Si fa notare che il `delay` può essere modificato a seconda delle necessità. Alcune serrature scattano con un impulso di un secondo, altre con un tempo superiore. Sul display appare la scritta “Aperto OK!”.

```
    digitalWrite(led,1);
    delay(2000);
    lcd.setCursor(0,1);
    lcd.print("Aperto OK!        ");
    digitalWrite(led,0);
    delay(2000);
}
```

Se il riconoscimento del dito fallisce o se si usa un dito non memorizzato, appare la scritta “Dito non trovato”.

```
    else
    {
        lcd.setCursor(0,1);
        lcd.print("Dito non trovato");
    }
}
```

La scritta predefinita è “Premere un dito”.

```
lcd.setCursor(0,1);
lcd.print("Premere un dito ");
```

La Figura 7.8 illustra i passaggi del processo di identificazione e di apertura della porta.

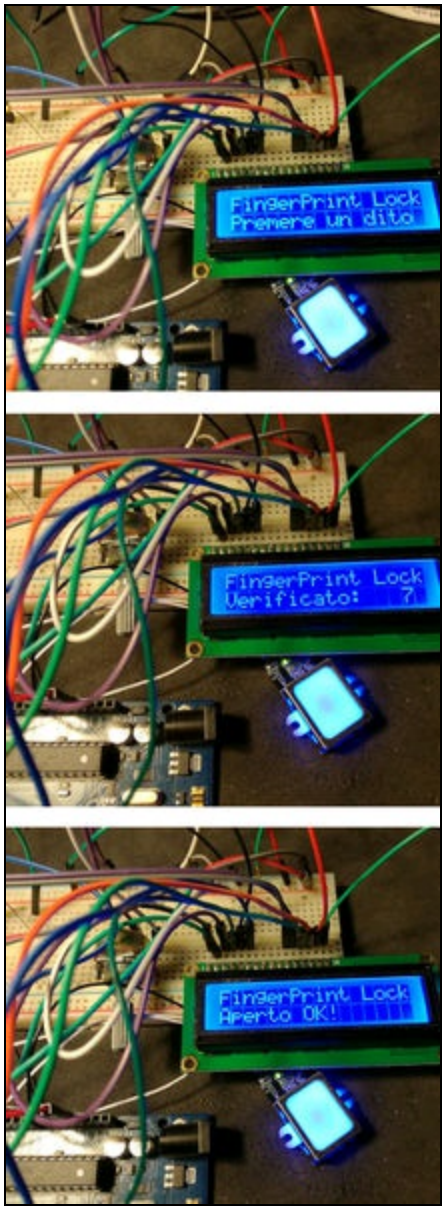


Figura 7.8 I passaggi del processo di identificazione e apertura della porta.

Sistema di allarme

Descrizione

Tutti i corpi caldi emettono energia sotto forma di radiazioni luminose. La maggior parte di queste radiazioni risulta invisibile all'occhio umano, poiché ha una frequenza inferiore a quella della luce visibile nella gamma del rosso: vengono chiamate *radiazioni infrarosse*. Queste radiazioni possono però essere rilevate tramite specifici sensori elettronici.

Il nostro progetto consente di monitorare i locali della casa attraverso l'uso di sensori a infrarossi sensibili alla presenza umana. Quindi ci sembrano ideali per essere impiegati in un sistema di allarme. È stato previsto l'uso di una sirena come sistema acustico di allarme, ma anche una connessione remota per l'invio di un'email a un indirizzo predefinito o di un SMS a un numero di telefono.

CODICE DI ESEMPIO

Tutti i file del progetto "Sistema di allarme" sono disponibili nelle risorse del libro presso il sito dell'autore all'indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Arduino UNO;
- 1× sensore PIR;
- 1× buzzer o sirena;
- 1× LED (opzionale);
- 1× ESP8266 (opzionale).

Sensore PIR

Un sensore PIR (*Passive InfraRed*) è un sensore a infrarossi passivo che rileva i raggi infrarossi irradiati dai corpi caldi. Sono molto usati come rilevatori di movimento di persone. Secondo le specifiche del datasheet, il sensore PIR è in grado di rilevare un oggetto in movimento in una stanza entro un raggio di 7 metri e con un angolo di incidenza di 110°. Per alimentarlo è sufficiente collegarlo al pin 5 V di Arduino. I tipici sensori PIR in commercio sono dotati di una cupola in plastica che funziona da lente per aumentare la copertura ottica.

La Figura 8.1 illustra un sensore PIR e la sua piedinatura. Di solito è dotato anche di due trimmer per la regolazione della sensibilità (da 3 a 7 metri) e del ritardo (da 5 a 200 secondi).

Il jumper L-H (LOW-HIGH) può impostare il trigger (L) o (H), ovvero l’attivazione del segnale singola oppure ripetuta. Il pin contrassegnato come OUT è l’uscita del sensore che va collegata a un ingresso digitale di Arduino.

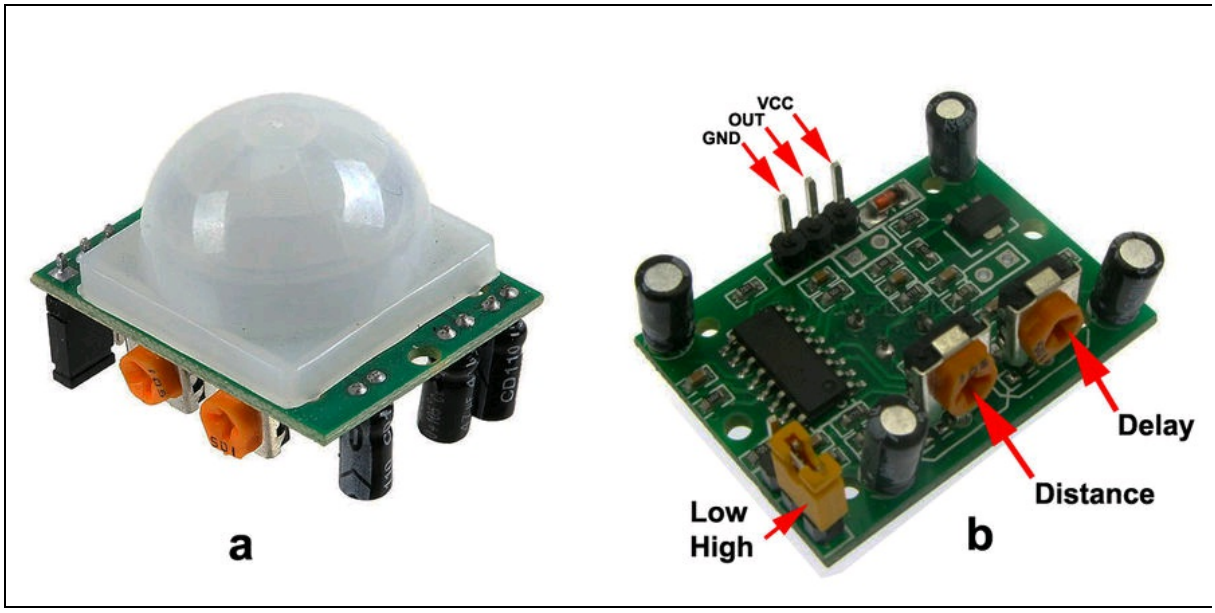


Figura 8.1 Un sensore PIR e la sua piedinatura.

Buzzer o sirena

Per il test viene usato un normale buzzer piezoelettrico, ma per il progetto definitivo si può pensare a una potente sirena esterna da alimentare separatamente e attivabile con un relè. La Figura 8.2 illustra un buzzer per Arduino e una sirena di potenza a 12 volt. Per alimentare la sirena si può utilizzare un alimentatore oppure una batteria al piombo da 12 volt.

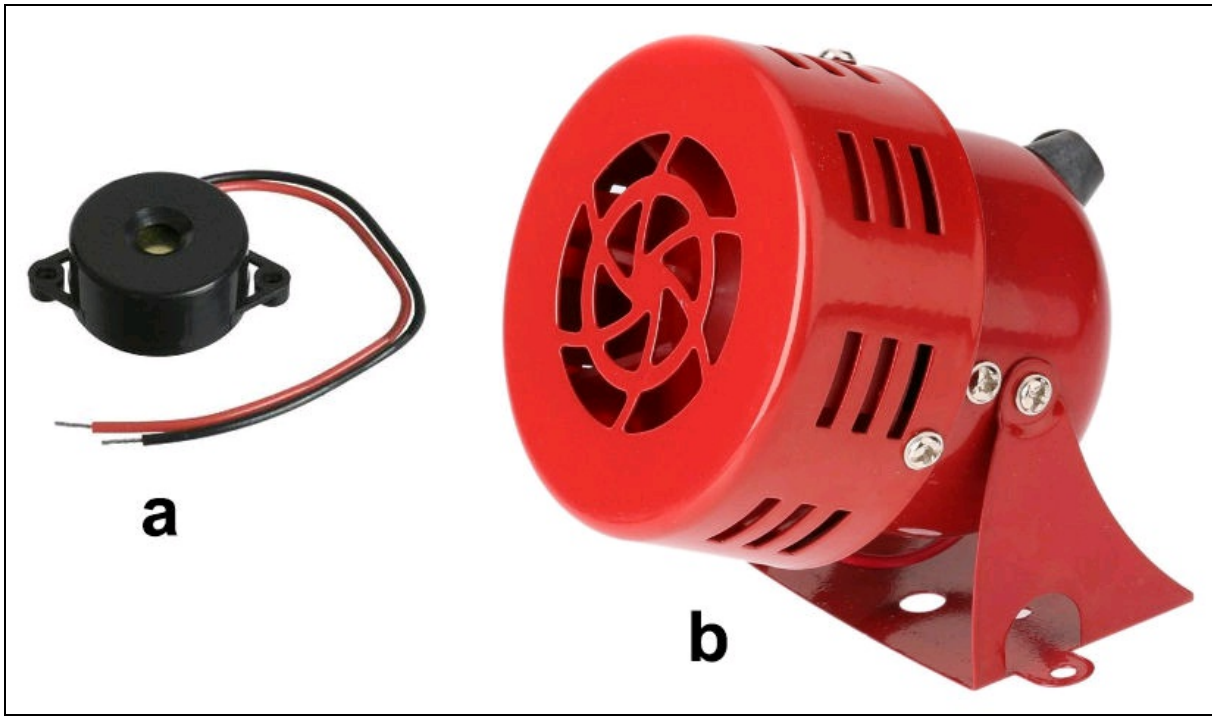


Figura 8.2 Buzzer per Arduino (a). Una sirena di potenza a 12 volt (b).

Il circuito elettrico

La Figura 8.3a illustra lo schema generale, che prevede l’uso di un solo sensore PIR per il controllo a infrarossi di una stanza della casa. Se dovessero servire più sensori, basta collegarli a più porte digitali di Arduino.

Per il test viene usato un buzzer, ma per il progetto definitivo si può pensare a una sirena collegata a un’alimentazione esterna e un modulo relè. Un LED opzionale può essere connesso direttamente al pin 13 di Arduino per monitorare quando l’allarme viene attivato. Questo pin può essere usato per attivare un modulo relè cui collegare una sirena esterna (Figura 8.3b).

Per il prototipo di questo circuito viene usata una breadboard, ma per il progetto definitivo si possono collegare i pochi componenti con cavetti volanti e dei mammut. Se però si usano più sensori PIR, si consiglia una millefori o un circuito stampato.

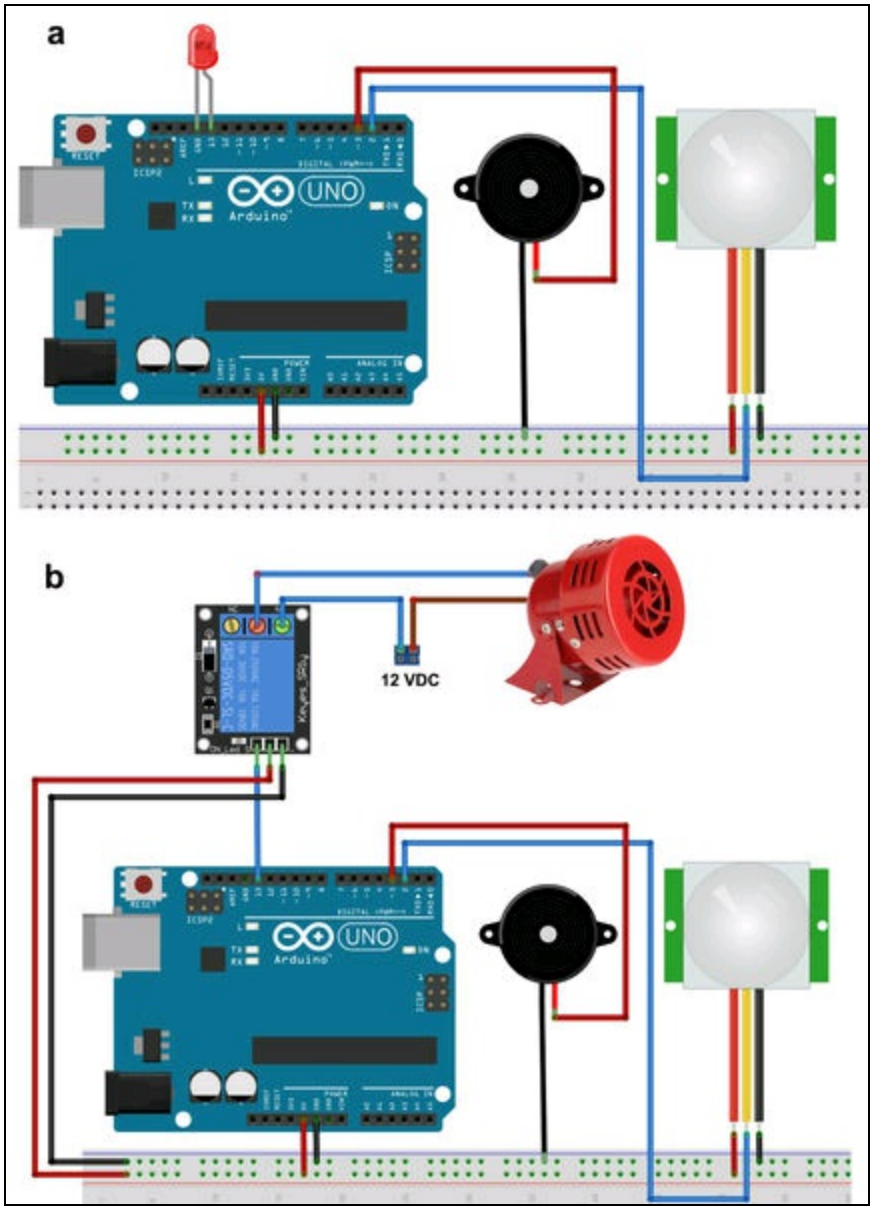


Figura 8.3 Schema di collegamento su breadboard (a). Collegamento a un relè e a una sirena esterna (b).

Il codice

Lo sketch prevede il monitoraggio dell'ingresso digitale 2, collegato al sensore PIR. Quando il sensore rileva la presenza di un corpo caldo, attiva il pin digitale 13 collegato al LED oppure al relè, che a sua volta attiva la sirena esterna. Contemporaneamente viene emesso un suono dal buzzer collegato al pin digitale 3.

All'inizio vengono dichiarate le variabili per i pin digitali e una variabile booleana `flag` per controllare lo stato della trasmissione seriale.

```
byte buzzer = 3;
byte signal = 2;
byte ledRed = 13;
bool flag = false;
```

Nella funzione `setup()` vengono inizializzati i pin digitali del LED e del buzzer e la porta seriale a 115200 per la comunicazione verso un modulo ESP8266, opzionale.

```
void setup()
{
    pinMode(buzzer,OUTPUT);   // imposta il pin come uscita
    pinMode(ledRed,OUTPUT);   // imposta il pin come uscita
    Serial.begin(115200);     // per la comunicazione seriale
}
```

Nella funzione `loop()` viene monitorato lo stato del sensore PIR collegato al pin digitale 2. Quindi viene chiamata la funzione `alarm`.

```
void loop()
{
    bool pirSignal = digitalRead(signal); // monitoraggio dell'ingresso PIR
    alarm(pirSignal); // attiva/disattiva l'allarme
    delay(100);
}
```

La funzione `alarm` riceve lo stato del sensore PIR, che può essere alto o basso, ovvero vero o falso, per cui, se la variabile booleana `state` è vera, attiverà il buzzer e il LED/relè/sirena. Quando la variabile `state` torna falsa, spegnerà il buzzer e il LED/relè/sirena. Si fa notare che per

emettere un suono con Arduino si usa l’istruzione `tone`. Per fermare il suono si usa l’istruzione `noTone`.

Contemporaneamente viene inviato un messaggio “allarme” alla porta seriale e viene posto il `flag` a `true` per impedire invii multipli.

Quando il sensore PIR non rileva più la presenza di un corpo caldo, il sistema buzzer/LED/relè/sirena cessa e il `flag` viene posto di nuovo a `false`.

```
void alarm(bool state)
{
    if (state)
    {
        tone(buzzer, 300);            // produce un suono a 300 Hz
        digitalWrite(ledRed, state);  // accende il LED/relè/sirena
        if (flag==false)
        {
            Serial.print("allarme");  // invia alla porta seriale "allarme"
            flag=true;                // il flag è true
        }
    }
    else
    {
        noTone(buzzer);               // ferma il suono
        digitalWrite(ledRed, state);  // spegne il LED/relè/sirena
        flag = false;                 // il flag è false
    }
}
```

Test e regolazione dell’allarme

Una volta collegati i componenti, per provare se tutto funziona basta posizionare il sensore PIR in modo che non riceva nessun movimento di oggetti caldi o persone. Quindi, mettere una mano o muoversi nel raggio di 110^{o} davanti al sensore. In base alle regolazioni della sensibilità (3-7 metri) e del ritardo (5-200 secondi), si vedrà il LED accendersi e si sentirà il buzzer suonare. Aprendo il monitor seriale si vedrà la scritta “allarme”. Basta togliere la mano o spostarsi dietro al sensore per spegnere il LED e il buzzer.

Regolare la sensibilità e il ritardo come necessario. Per esempio potrebbe essere utile regolare il ritardo di intervento del sensore PIR a qualche decina di secondi, in modo da poter uscire dalla stanza quando il

sistema d’allarme viene messo in funzione. La sensibilità può essere regolata in base alle dimensioni della stanza da monitorare.

Email o SMS in caso di allarme

Ricevere un’email o un SMS di allerta in caso di intrusione è quanto mai destabilizzante. Cosa fare? Si ritorna in fretta e furia dalle vacanze? Si chiamano i carabinieri? Si chiamano i vicini? E se l’allarme si è attivato da solo? A ogni modo, per chi vuole essere avvisato in caso di attivazione dell’allarme, ecco un esempio per ESP8266, in grado di spedire un’email a un indirizzo predefinito o un SMS a un numero di telefono.

Invio email con ESP8266

È la soluzione più semplice. Basta collegare il pin TX di Arduino al pin RX di un modulo ESP8266. Per lo schema di collegamento hardware si veda il progetto “Monitoraggio meteo”.

Codice commentato

Lo sketch `ESP8266_EMAIL_SENDER` è disponibile nelle risorse del libro, per cui limitiamo la spiegazione alle parti più importanti e soprattutto al codice da modificare. Si fa notare che lo sketch include la libreria `Gsender`, che viene caricata localmente. All’inizio dello sketch bisogna includere la libreria `Gsender`, pensata per effettuare l’invio tramite il server `smtp.gmail.com`, ma che si potrà modificare per spedire email con qualsiasi server SMTP. Quindi bisogna inserire SSID e password della rete di casa. La variabile `led` imposta il LED sulla porta GPIO2 di ESP8266. Il collegamento del LED è opzionale.

```
#include <ESP8266WiFi.h>
#include "Gsender.h"
#pragma region Globals
const char* ssid = "SSID";
const char* password = "password";
int led = 2;
```

Nel `loop()` viene effettuato il monitoraggio della porta seriale: se arriva la stringa `"allarme"` inviata da Arduino, viene chiamata la funzione `sendemail()`.

```
void loop()
{
    if (Serial.available())
    {
        String data = Serial.readString();
        Serial.println(data);
        if (data = "allarme")
        {
            sendemail();
        }
    }
}
```

La funzione `sendemail()` fa la magia. Innanzitutto, accende il led su GPIO2 per segnalare che la funzione è stata chiamata correttamente, quindi imposta il puntatore alla classe `Gsender`. La stringa `subject` invia il soggetto dell'email, nel nostro caso `"ALLARME!"`. Nel corpo del messaggio si può scrivere un qualsiasi testo, per esempio `"ALLARME ATTIVATO!"`.

Se l'invio all'indirizzo predefinito ha successo, verrà stampato sul monitor seriale `"Messaggio inviato!"`, come illustrato nella Figura 8.4a, e il LED si spegne.

Aprendo il client di posta si vedrà il messaggio arrivato con il soggetto `ALLARME!` e il testo `ALLARME ATTIVATO!`, come illustrato nella Figura 8.4b.

Attenzione! ricordarsi di inserire nella riga `if(gsender->Subject(subject)->Send(xxx@email.com...` l'indirizzo del destinatario. Nell'esempio è `info@pierduino.com`.

```
void sendemail()
{
    digitalWrite(led,1);
    Gsender *gsender = Gsender::Instance();    // puntatore alla classe
    String subject = "ALLARME!";
    if(gsender->Subject(subject)->Send("info@pierduino.com", "ALLARME ATTIVATO!"))
    {
        Serial.println("Messaggio inviato!");
        digitalWrite(led,0);
    } else {
        Serial.print("Errore invio: ");
```

```
        Serial.println(gsender->getError());
    }
}
```

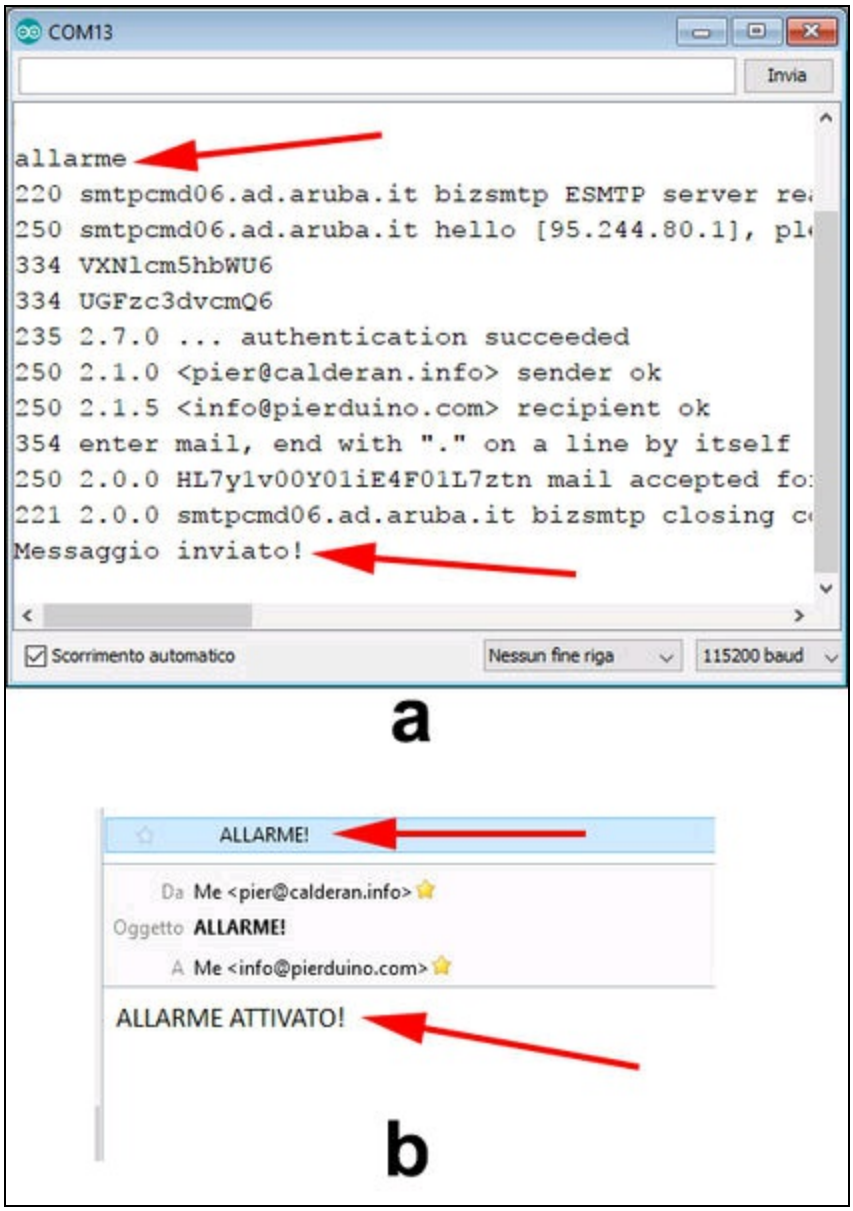


Figura 8.4 Il monitor seriale con la ricezione del messaggio allarme e l'invio del messaggio di posta all'indirizzo designato (a). Il messaggio di ALLARME! ricevuto (b).

Modificare il codice

Per inviare correttamente un'email usando il server che viene usato di solito con il client di posta, è necessario modificare il codice del file

`Gsender.h`. Questo file viene caricato insieme allo sketch. Le uniche righe da modificare sono le seguenti:

```
const int SMTP_PORT = 465;
const char* SMTP_SERVER = "smtps.aruba.it"; // server di posta
const char* EMAILBASE64_LOGIN = "cFlckBjYWxkhbi5pbmZv"; // codifica base64 del
login
const char* EMAILBASE64_PASSWORD = "bmFadfayZG8="; // codifica base64 della
password
    const char* FROM = "pier@calderan.info"; // indirizzo del mittente
```

La variabile `SMTP_PORT` imposta la porta del server di posta, di solito 465. La variabile `SMTP_SERVER` imposta il nome del server di posta. Si può usare il server di posta normalmente usato per mandare le email dal proprio client. Nell'esempio è `smtps.aruba.it`. La variabile `EMAILBASE64_LOGIN` imposta il parametro `base64` dell'account di posta (si veda il prossimo paragrafo). Allo stesso modo, la variabile `EMAILBASE64_PASSWORD` imposta il parametro `base64` della password di posta (si veda il prossimo paragrafo). La variabile `FROM` imposta l'indirizzo predefinito del mittente. Nell'esempio è `pier@calderan.info`.

Una volta impostati questi parametri, si può salvare il file e tornare allo sketch.

Codifica base64

Base64 è un sistema di codifica che usa 64 simboli e viene usato principalmente per codificare i dati binari nelle email. Per codificare l'indirizzo del mittente e la password recarsi sul sito `https://www.base64encode.org`.

Come indicato nella Figura 8.5, basta incollare l'indirizzo del mittente e fare clic sul pulsante *ENCODE*. Copiare il testo codificato e incollarlo nel suddetto parametro `EMAILBASE64_LOGIN`. Ripetere l'operazione anche per la password di posta e incollare il testo codificato nel parametro `EMAILBASE64_PASSWORD`.

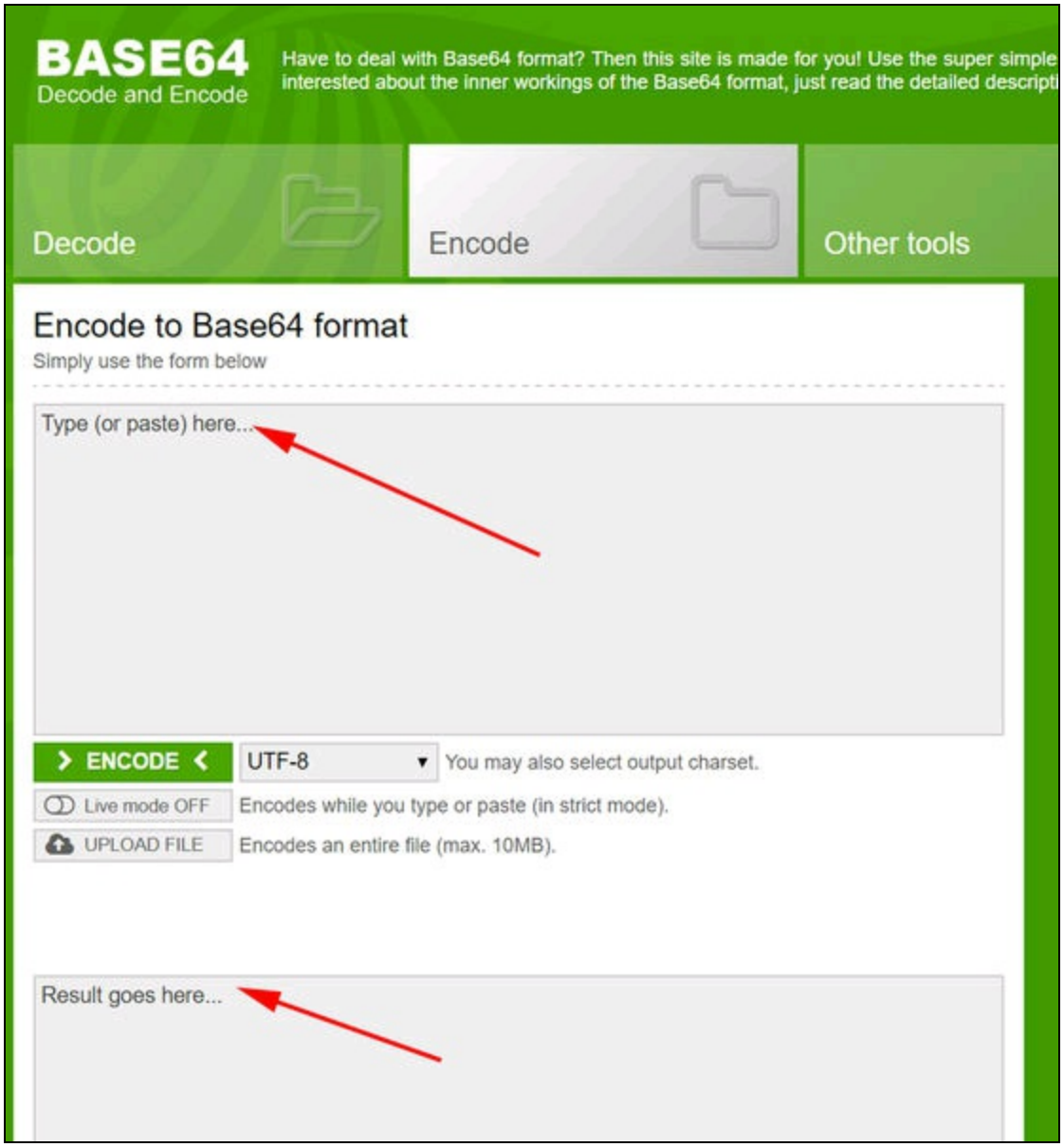


Figura 8.5 La home page del sito base64.org.

Invio SMS con ESP8266

Se per qualche motivo si preferisce ricevere un SMS invece di un'email, oppure si vuole mandare il messaggio di allarme al telefono di un vicino di casa, ecco un progetto adatto. A questo scopo si può utilizzare una soluzione proposta da Twilio.

Innanzitutto, bisogna creare un account gratuito presso Twilio (https://www.twilio.com) di cui abbiamo parlato nel Capitolo 4. Twilio è una piattaforma per ricevere chiamate telefoniche e inviare e ricevere messaggi di testo.

Una volta creato un account, si ottengono un numero di autorizzazione (SID) e un token di autorizzazione (AUTH TOKEN). Questi dovranno essere usati nello sketch di esempio per la connessione remota. Dalla barra laterale della home page di Twilio si può accedere alla console e quindi alla sezione *Products & Services* (Figura 8.6).

Da qui si può accedere alla sezione *Phone Numbers* per ottenere un numero di telefono. Seguire la semplice procedura online per attivare il numero ed effettuare il test di invio di un SMS. Il costo è zero, perché l'account gratuito prevede un credito di 26 USD. Attenzione! Se si esaurisce il credito bisognerà acquistare il numero e pagare le chiamate e/o gli invii di SMS.

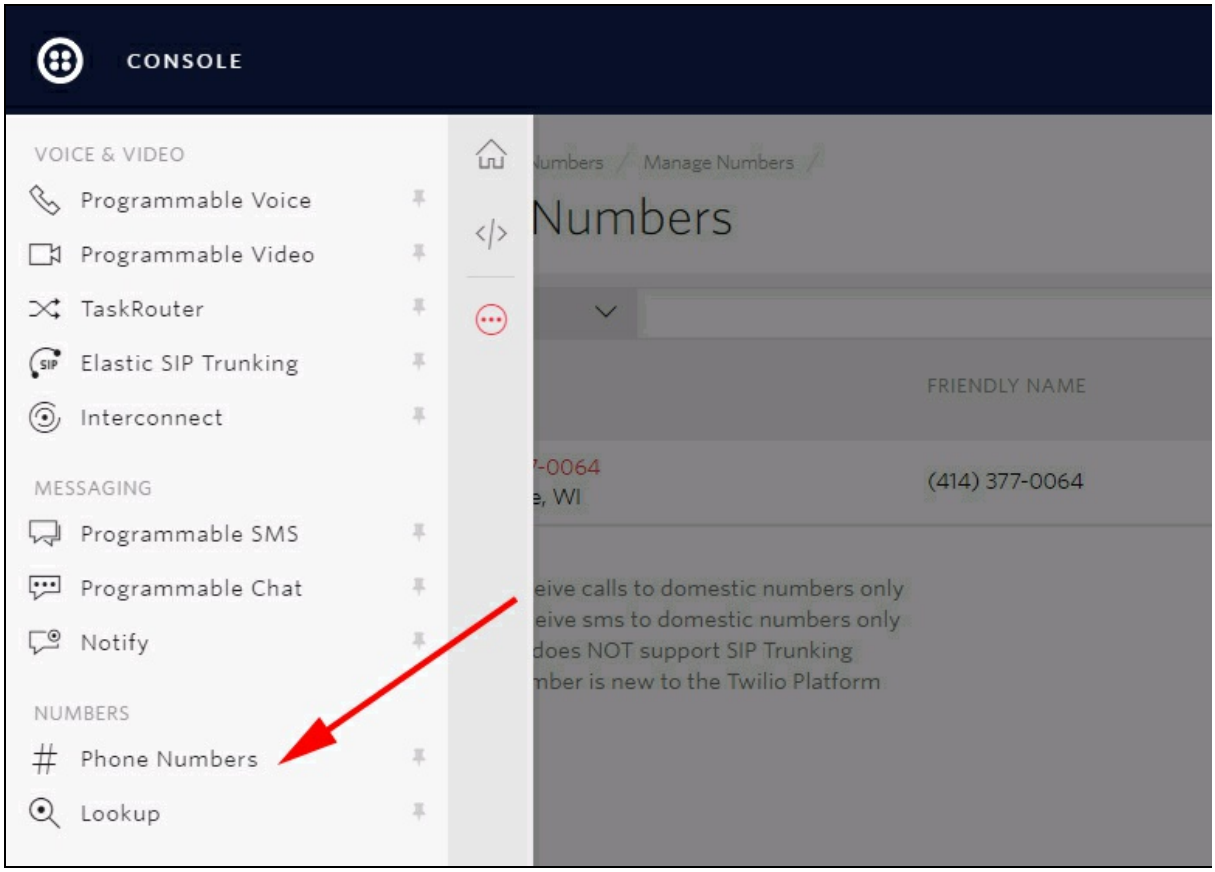


Figura 8.6 La sezione Products & Services della console Twilio.

Ottenuto il numero di telefono (Figura 8.7), lo si potrà usare per l'invio di SMS tramite una connessione remota.

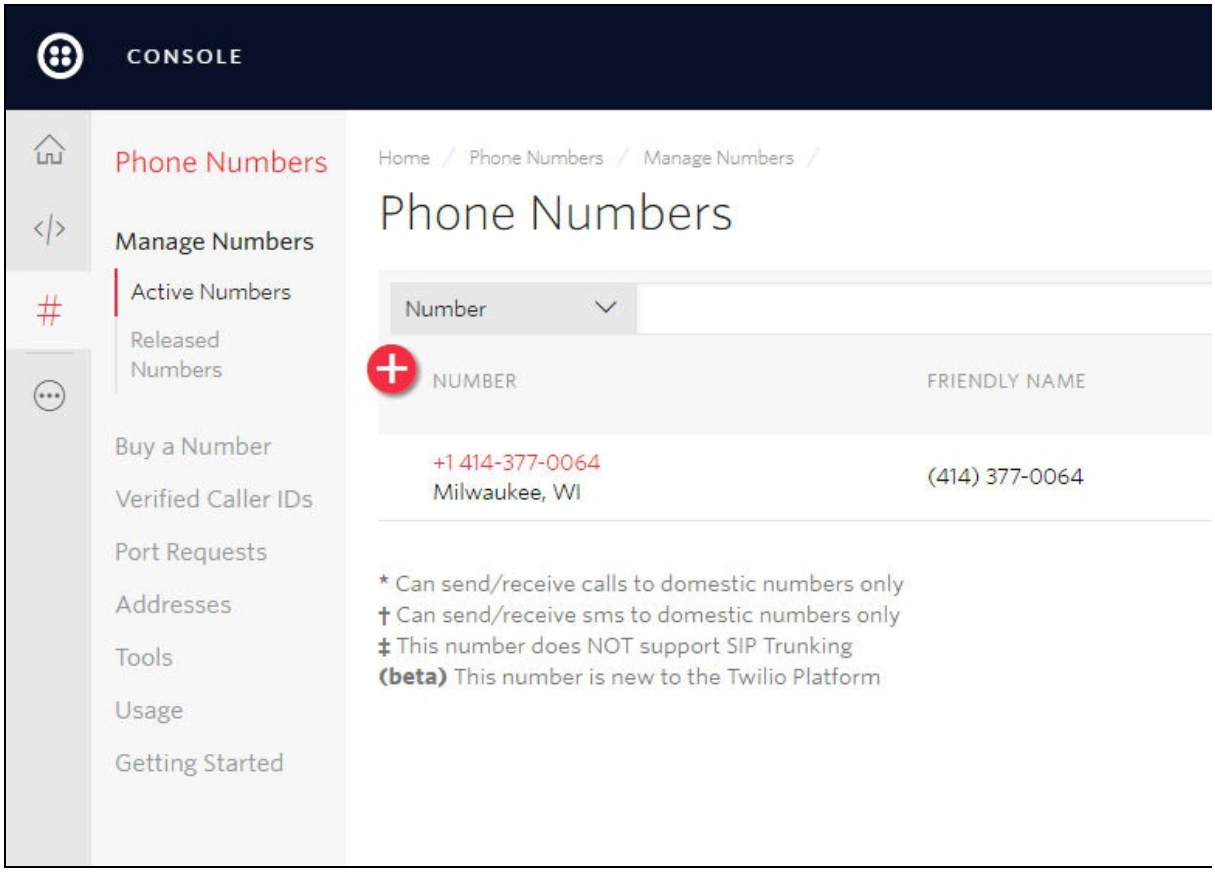


Figura 8.7 La sezione Phone Numbers della console Twilio.

Usare Twilio con ESP8266

Recandosi all'indirizzo seguente, si possono leggere le istruzioni per eseguire l'invio di SMS con ESP8266: `https://www.twilio.com/docs/guides/send-sms-and-mms-messages-esp8266-cpp`. Dalla seguente pagina è possibile scaricare il file di esempio dal repository Twilio: `https://github.com/TwilioDevEd/twilio_esp8266_arduino_example`. Ovviamente, abbiamo usato questo esempio per integrarlo nel nostro sketch.

Il codice

Nel codice di esempio sono poche le cose da cambiare. Come al solito, bisogna inserire SSID e password della rete di casa, quindi il `SID` e l'`AUTH TOKEN` dell'account Twilio e infine il numero cui mandare l'SMS e il numero di telefono assegnato da Twilio.

```
const char* ssid = "Your SSID";               // SSID
const char* password = "Network Password";    // Password
const char* account_sid = "ACXXXXXXXXXXXXX";  // SID dell'account Twilio
const char* auth_token = "Your AUTH TOKEN";   // AUTH TOKEN dell'account Twilio
String to_number    = "+18005551212";         // il numero a cui mandare l'SMS
String from_number = "+18005551212";          // il numero assegnato da Twilio
```

Come nell'esempio precedente, nel `loop()` è stato aggiunto il monitoraggio della porta seriale che, nel caso venga ricevuta la stringa `"allarme"` da Arduino, effettua la chiamata alla funzione

`twilio_server.handleClient()`.

```
void loop()
{
    if (Serial.available())
    {
        String data = Serial.readString();
        Serial.println(data);
        if (data = "allarme")
        {
            twilio_server.handleClient();;
        }
    }
}
```

Controllo RFID

Descrizione

La tecnologia RFID (*Radio Frequency IDentification*) e la sua derivata NFC (*Near Field Communication*) hanno letteralmente cambiato il nostro modo di vivere, invadendo il settore pubblico e privato: dal Telepass ai parcheggi, dal pagamento con smartphone al rilevamento delle presenze nelle aziende, dall'apertura di porte d'albergo fino al controllo anti-taccheggio nei negozi.

Questa tecnologia per l'identificazione automatica a radiofrequenza si è diffusa anche nel mondo dei maker. Tant'è che si trovano diversi dispositivi per sviluppare applicazioni con Arduino, Raspberry Pi e affini. La tecnologia si basa sulla capacità di memorizzare dati numerici o di testo su particolari etichette elettroniche, comunemente chiamate *transponder* o *tag*.

A seconda della tecnologia passiva o attiva, tutto si svolge attraverso l'interscambio di informazioni fra il richiedente, ovvero il dispositivo *reader* (che agisce anche come *writer*) e il risponditore, ovvero il *trasponder*. In pratica, sfruttando la radiofrequenza, si possono scrivere informazioni di vario tipo sui tag che poi vengono lette senza contatto. Se i tag sono di tipo attivo, cioè alimentati da una pila interna (per esempio il Telepass) inviano i dati su richiesta del reader che li riceve quando il tag si avvicina. In altre parole, i dispositivi RFID possono essere considerati sistemi di lettura e scrittura senza fili.

Lo standard dei tag passivi RFID/NFC prevede una frequenza di funzionamento a 13,56 MHz e 125 KHz e una distanza massima di circa

5-6 cm. I tag attivi, a seconda dei modelli e della gamma di frequenza utilizzata (fino a 5,8 GHz), possono raggiungere distanze di qualche decina di metri. Per i tag attivi il sistema di lettura/scrittura è completamente diverso.

Fra i diversi prodotti in commercio, ci siamo orientati sul modulo RFID tipo RC522, per il quale esiste una libreria per Arduino che ne facilita lo sviluppo e di cui esistono kit completi molto economici.

Applicazioni

Questo progetto può servire a molteplici usi. Per esempio, si potrebbe controllare l'apertura di una porta, l'accensione delle luci di una stanza o l'accensione di qualsiasi altro apparecchio in casa. In un ambiente lavorativo, si può pensare al controllo degli accessi o delle presenze. Lasciamo spazio alla fantasia.

CODICE DI ESEMPIO

Tutti i file del progetto "Controllo RFID" sono disponibili nelle risorse del libro presso il sito dell'autore all'indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× Arduino UNO;
- 1× modulo RFID tipo RC522;
- 2× tag RFID (almeno);
- 1× pulsante;
- 1× LED;
- 2× resistenza da 220 Ω;
- 1× LED RGB (consigliato a catodo comune);
- 1× trimmer o potenziometro da 10 KΩ;
- 1× display LCD 16 caratteri × 2 righe.

Modulo RC522

Questo modulo reader/writer funziona a 13,56 MHz ed è basato sul chip MFRC522. È progettato per la lettura/scrittura di tag standard MIFARE. Può essere acquistato come modulo singolo, ma si consiglia vivamente l'acquisto del diffusissimo kit RFID RC522 composto da:

- 1× modulo reader/writer RC522;
- 1× card S50 standard;
- 1× card S50 a forma di portachiavi;
- 1× connettore 8 pin a 90^{o} (da saldare);
- 1× connettore 8 pin dritto (da saldare).

La Figura 9.1a illustra il contenuto del kit. La Figura 9.1b illustra la piedinatura e i bus di comunicazione disponibili. La piedinatura del lettore MFRC-RC522 offre tre tipi di connettività:

- interfaccia UART;

- interfaccia I^2C;
- interfaccia SPI;
- il pin IRQ non viene usato.

Con la libreria per Arduino verrà usata l’interfaccia SPI, cioè i pin contrassegnati come:

- SDA;
- SCK;
- MOSI;
- MISO.

L’alimentazione deve essere di 3,3 volt, per cui si dovrà collegare il modulo al pin 3.3 V di Arduino.

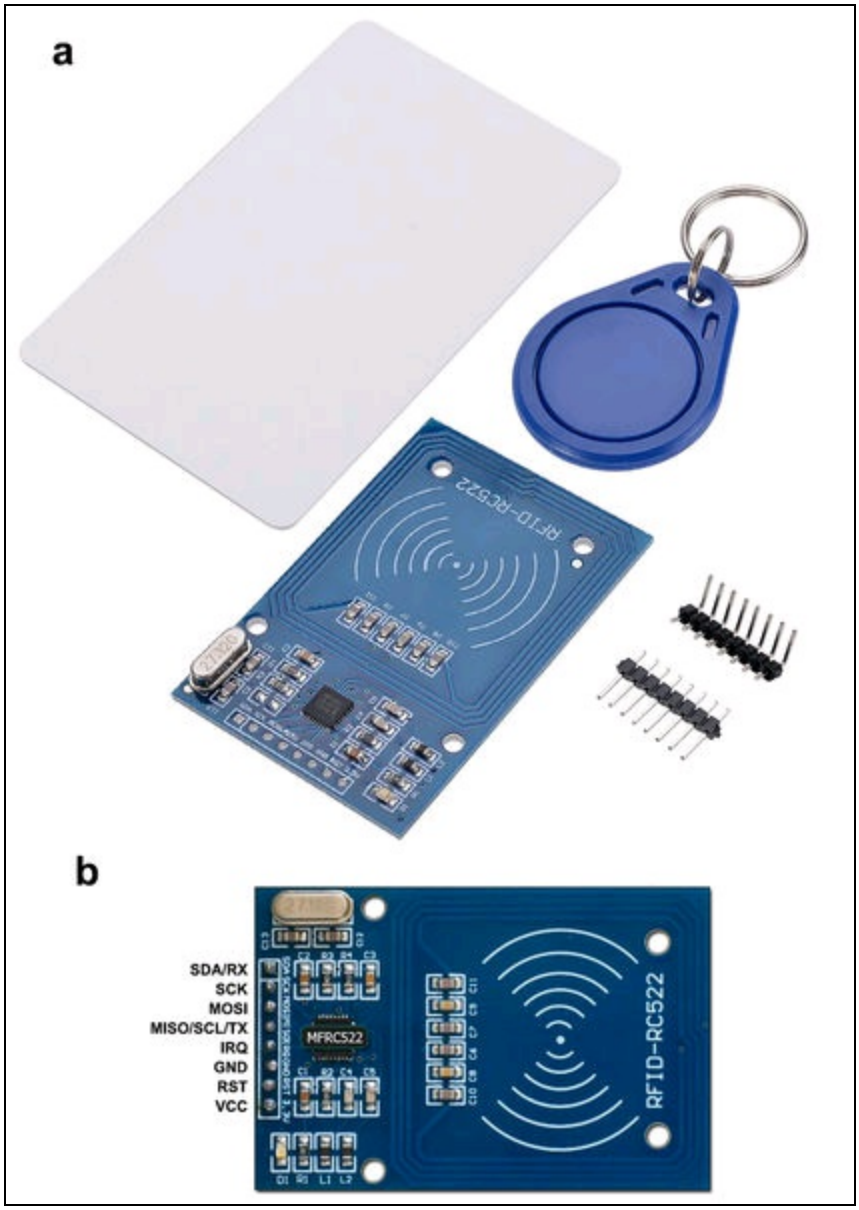


Figura 9.1 Il kit RFID RC522 (a). Piedinatura e bus del modulo RC522.

Standard MIFARE

MIFARE è il marchio di NXP Semiconductors che produce una serie di chip utilizzati ampiamente in smartcard senza contatto e schede di prossimità RFID/NFC.

Le card MIFARE rispondono alla normativa ISO 14443. Utilizzano una memoria interna (da 1 a 4 KB) e, a seconda della destinazione d'uso, possono essere inserite in oggetti di qualsiasi forma e dimensione.

Nella figura sono illustrati alcuni tipi di tag disponibili in commercio, che possono inseriti in oggetti come portachiavi, anelli, braccialetti per bambini, etichette adesive

e così via.

Il circuito

Il prototipo su breadboard è illustrato nella Figura 9.2, ma quello definitivo va montato su millefori o circuito stampato (consigliato).

Il circuito prevede l'uso di un display per ottenere un dispositivo usabile in autonomia per avere la visualizzazione di tutte le funzioni del reader.

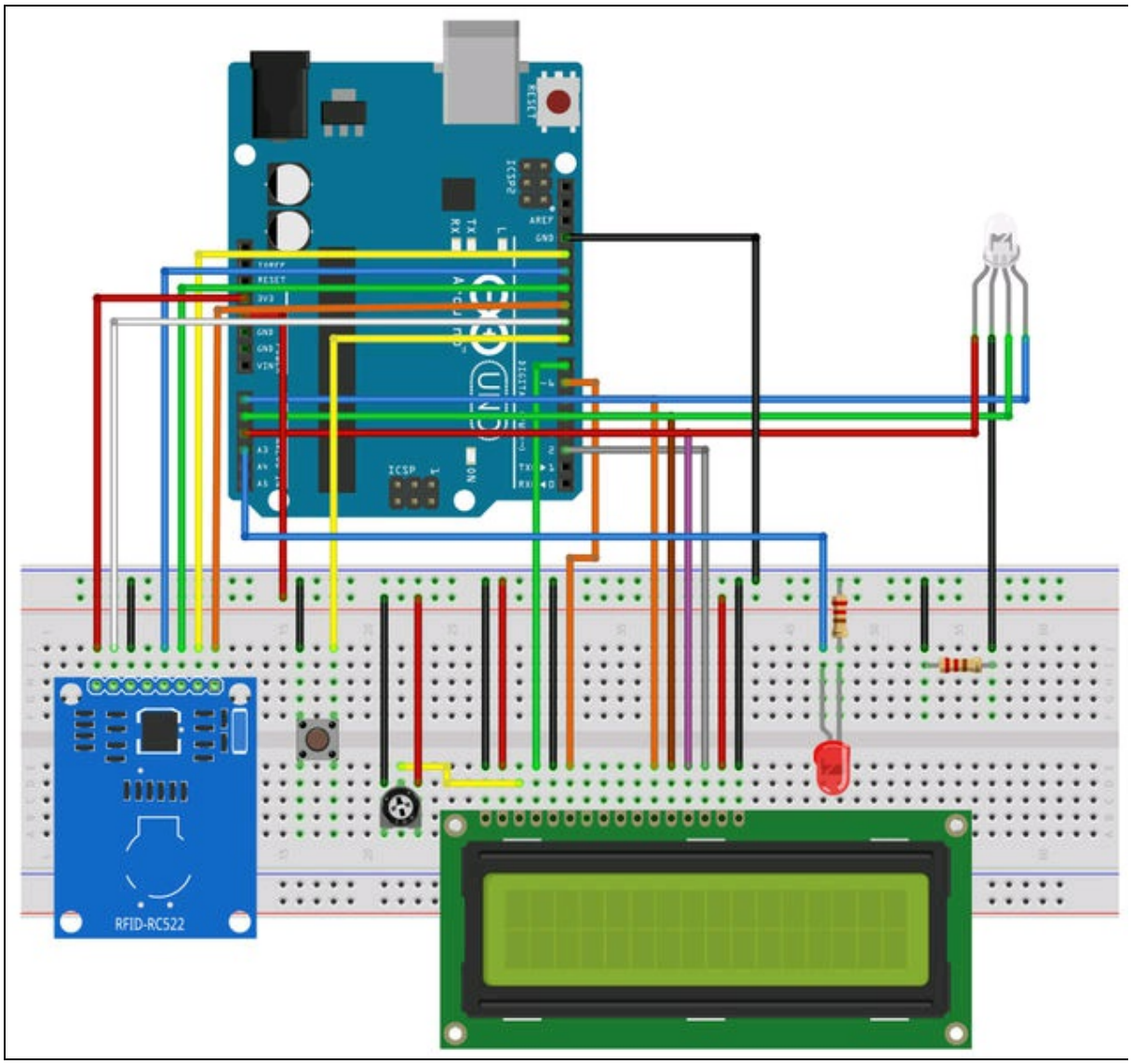


Figura 9.2 Schema di collegamento su breadboard.

Collegamento del display

Il display segue un collegamento diverso da quello previsto dalla sua libreria di riferimento. Questo per lasciar posto ai collegamenti del lettore RFID, che necessita dei pin della porta SPI di Arduino, cioè 10, 11, 12 e 13. Il trimmer serve per il contrasto del display e va collegato alla linea 5 V e GND della breadboard e al pin 3 del display.

Tabella 9.1 Schema di collegamento del display alla scheda Arduino e alla breadboard.

Pin del display	Pin di Arduino/breadboard
1	GND
2	5 V
3	Piedino centrale del trimmer
4	7
5	GND
6	6
7-10	Non usati
11	5
12	4
13	3
14	2
15	5 V
16	GND

Collegamento del modulo RC522

Dopo aver deciso in che posizione mettere il lettore RC522, bisogna saldare uno dei connettori a 8 pin contenuti nel kit. Uno è a 90° e uno è dritto, per cui il reader può stare appoggiato o stare in piedi. Inserire il modulo nella breadboard in senso orizzontale o verticale a seconda di come è stata saldata la striscia di pin.

Tabella 9.2 Schema di collegamento del modulo RC522 alla scheda Arduino e alla breadboard.

Pin RC522	Pin di Arduino/breadboard
3.3	3.3
RST	9
GND	GND
MISO	12
MOSI	11
SCK	13
SDA	10

Collegamento degli altri componenti

Posizionare il pulsante a cavallo della breadboard e collegarlo al pin 8 di Arduino. Questo pulsante è collegato direttamente al pin digitale senza resistore di pull-down, perché verrà attivato come ingresso `INPUT_PULLUP` dallo sketch.

A questo punto non sono disponibili altri pin digitali, per cui si useranno come uscite digitali i pin analogici.

1. Inserire un LED nella breadboard collegando il catodo a massa con una resistenza da 220 Ω in serie.
2. Collegare l'anodo del LED all'ingresso analogico A3 che verrà usato come output.
3. Inserire un LED RGB per visualizzare otticamente le varie funzioni del lettore RFID.
   a. Collegare il terminale R (corrispondente al rosso) all'ingresso analogico A2.
   b. Collegare il terminale G (corrispondente al verde) all'ingresso analogico A1.
   c. Collegare il terminale B (corrispondente al blu) all'ingresso analogico A0.
   d. Collegare il catodo comune a massa, mettendo in serie un resistenza da 220 Ω (nel caso in cui il LED RGB sia ad anodo

comune, collegarlo all’alimentazione positiva +5 V). Nel nostro caso il LED RGB è a catodo comune.

Il codice

La libreria usata per il modulo RC522 si chiama MFRC522 ed è disponibile sia nelle risorse del libro sia all'indirizzo `https://github.com/miguelbalboa/rfid`. La libreria legge e scrive diversi tipi di schede basati sullo standard MIFARE. Per l'installazione della libreria nell'IDE di Arduino, si veda il box *Come aggiungere una libreria all'IDE di Arduino* del progetto "Monitoraggio meteo".

Dal menu della libreria si possono provare molti esempi per testare il modulo RFID, ma abbiamo preferito scrivere uno sketch completo di gestione del display e soprattutto in grado di gestire un controllo degli accessi a livello professionale.

Codice commentato

Lo sketch che abbiamo creato per questo progetto si chiama `CONTROLLO_RFID`. In questo sketch viene fatto uso della EEPROM (*Electrically Erasable Programmable Read-Only Memory*) di Arduino attraverso l'inclusione della libreria omonima. Nella memoria EEPROM si possono leggere e scrivere fino a 1KB di dati (1024 byte). Questi dati restano in memoria anche quando si spegne la scheda e non vengono più cancellati fino a che non lo si desidera. Sfruttando questa caratteristica, si potranno scrivere i dati identificativi di una card, che fungerà da Master, e i dati identificativi degli altri tag, che fungeranno da Slave.

Siccome nel kit è inclusa una card RFID S50 a forma di carta di credito, verrà usata come Master Card. Il tag a forma di portachiavi del kit, invece, verrà usato come Slave.

All'inizio dello sketch si includono le librerie `EEPROM`, `SPI`, `MFRC522` e `LiquidCrystal`. Quindi viene creato l'oggetto `lcd(7,6,5,4,3,2)` con la

piedinatura che rispecchia quella dello schema elettrico.

```
#include <EEPROM.h>         // Libreria EEPROM per leggere e scrivere fino a 1KB
(1024 byte)
#include <SPI.h>            // Libreria SPI
#include <MFRC522.h>  // Libreria Mifare RC522
#include <LiquidCrystal.h>  // Libreria LiquidCrystal
    LiquidCrystal lcd(7,6,5,4,3,2);
```

Poi vengono dichiarate tutte le variabili usate nel programma. Si fa notare che il codice prevede si l'uso di un LED RGB a catodo comune o ad anodo comune. Si ricorda che le variabili per i LED usano gli ingressi analogici A0, A1 e A2 come uscite digitali.

```
#define LED_ON HIGH  // invertire in base all'anodo o catodo comune LED RGB
#define LED_OFF LOW  // invertire in base all'anodo o catodo comune LED RGB
#define blueLed A0   // uscita Analog A0 al pin B del LED RGB
#define greenLed A1  // uscita Analog A1 al pin G del LED RGB
#define redLed A2    // uscita Analog A2 al pin R del LED RGB
```

Questo pin analogico di uscita può essere usato per un LED o un modulo relè.

```
#define relay A3                // pin del led/relay
#define pulsanteCanc 8          // pin per cancellare la EEPROM
boolean match = false;          // variabile di confronto
boolean modoProgramma = false;  // variabile modalità programma
int successRead;                // variabile successRead
byte storedCard[4];             // salva la lettura ID dalla EEPROM
byte readCard[4];               // salva la scansione ID dal modulo RFID
byte MasterCard[4];             // salva il Master ID nella EEPROM
#define SS_PIN 10               // pin ss
#define RST_PIN 9               // pin rst
```

Con questa istruzione si crea l'oggetto `RFID`, istanziandolo sui pin SS e RST.

```
MFRC522 RFID(SS_PIN, RST_PIN); // Crea l'istanza RFID.
```

Nella funzione `setup()` vengono inizializzati tutti i pin digitali, compresi i pin analogici come uscite digitali.

```
void setup()
{
    pinMode(redLed, OUTPUT);
    pinMode(greenLed, OUTPUT);
    pinMode(blueLed, OUTPUT);
    pinMode(pulsanteCanc, INPUT_PULLUP);
    pinMode(relay, OUTPUT);
    digitalWrite(relay, LOW);
    digitalWrite(redLed, LED_OFF);
    digitalWrite(greenLed, LED_OFF);
    digitalWrite(blueLed, LED_OFF);
```

Si inizializza il display, il monitor seriale, l’interfaccia SPI e il modulo RC522.

```
lcd.begin(16,2);     // inizializza il display
lcd.print("RFID ACCESS 1.0");
Serial.begin(9600);  // inizializza il monitor seriale
SPI.begin();         // inizializza il bus SPI
RFID.PCD_Init();     // inizializza il modulo RC522
```

ATTENZIONE

Lo sketch di questo progetto è molto esteso e abbastanza complesso. Per motivi di spazio vengono riportate solo le operazione e le indicazioni più importanti. Si prega di fare riferimento al codice dello sketch nelle risorse del libro.

Procedura per la scrittura dei dati nella EEPROM

Per poter gestire la Master Card come card di amministrazione del sistema e gestire il riconoscimento dei vari tag Slave, è necessario scrivere i rispettivi ID nella EEPROM. In questo modo potranno essere riconosciuti nelle letture successive, anche dopo lo spegnimento della scheda Arduino.

All’avvio dello sketch, sul monitor seriale viene stampato `"Controllo accessi"` e con la funzione `showrederdetails` vengono mostrati i dettagli del lettore.

```
Serial.println(F("Controllo accessi"));
ShowReaderDetails(); // stampa i dettagli del lettore RC522 sul monitor seriale
```

A questo punto, se viene tenuto premuto il pulsante sul pin 8 per 5 secondi, si procede alla cancellazione della EEPROM. Il LED rosso si accende.

Come illustrato nella Figura 9.3a, sul monitor seriale appaiono in sequenza le seguenti indicazioni:

- `"Pulsante di cancellazione premuto";`
- `"Attendi 5 secondi per annullare o tieni premuto per cancellare";`
- `"La memoria verrà cancellata e non potrà essere ripristinata".`

Inizierà l’attesa di 5 secondi. Se il pulsante viene tenuto ancora premuto, inizia la cancellazione della EEPROM e verrà visualizzato `"Inizio cancellazione EEPROM"` (Figura 9.3b).

Il ciclo `for` scorre tutti i 1024 indirizzi della EEPROM. Se l’indirizzo EEPROM è 0 non fa nulla e va all’indirizzo successivo, altrimenti scrive 0. Segue `"EEPROM cancellata con successo"`. Inizierà un lampeggio del LED RGB rosso a intervalli di 100 ms. Se la cancellazione viene annullata, il LED rosso si spegne e sul monitor seriale appare `"Cancellazione annullata"`.

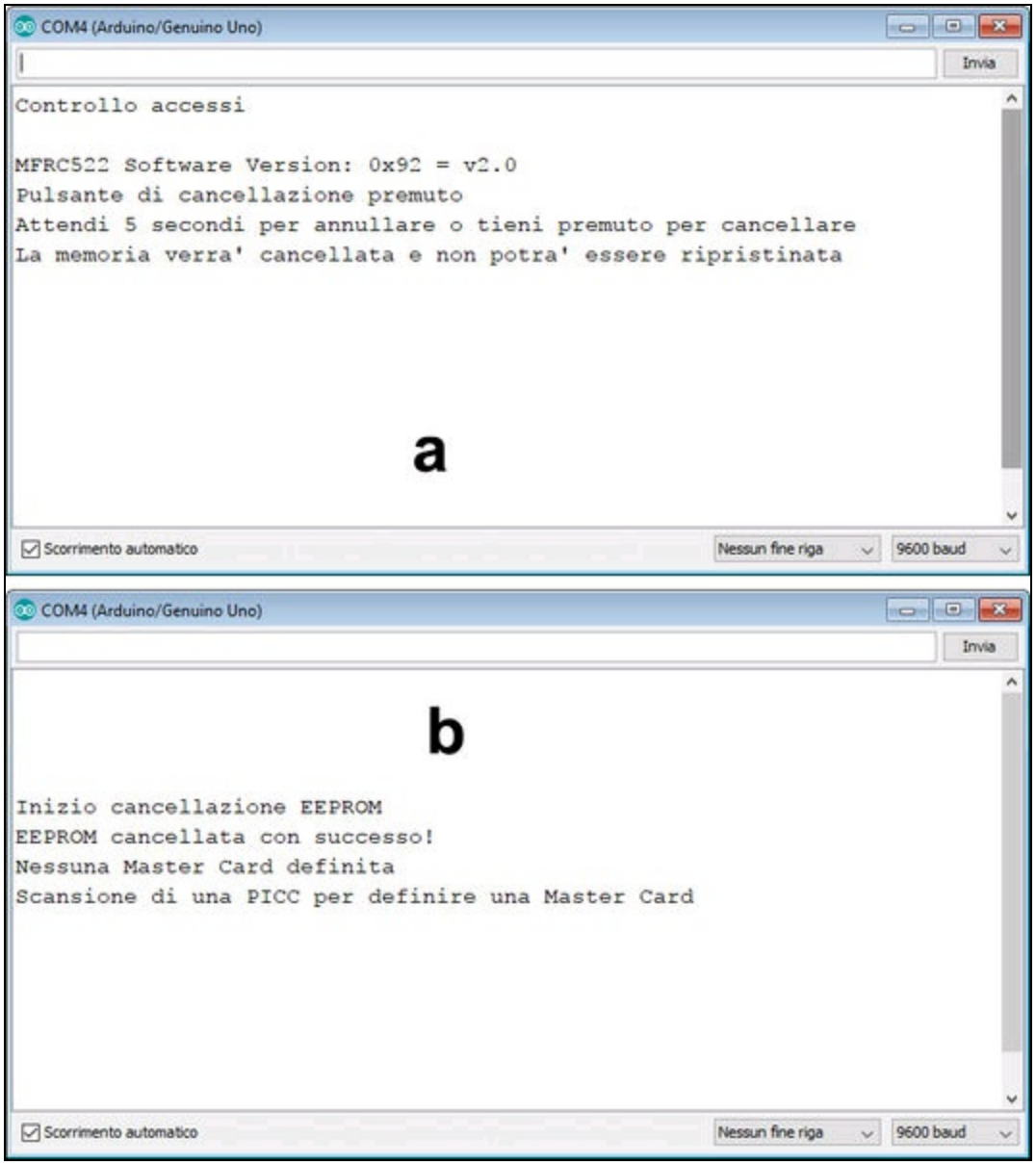


Figura 9.3 La sequenza di avvisi sul monitor seriale prima della cancellazione della EEPROM (a). La sequenza di avvisi sul monitor seriale durante la cancellazione della EEPROM (b).

Definizione della Master Card

Quando il lettore RC522 viene usato per la prima volta o, dopo la cancellazione della EEPROM, in qualsiasi momento, il programma controlla se è stata definita una Master Card. È un passaggio importante

per definire (o ridefinire) la Master Card che funge da carta di amministrazione degli accessi e, in modalità Programma, gestisce la memorizzazione e l'eliminazione di ogni tag Slave.

Con la funzione `EEPROM.read(1)` viene letto il primo indirizzo. L'indirizzo 1 della card dovrebbe contenere un "numero magico", ovvero una specie di chiave di accesso di amministratore del sistema. Per questo esempio è stato scelto il numero `143`, ma si può scegliere un numero qualsiasi.

Se l'istruzione `EEPROM.read(1)` legge un primo indirizzo che non corrisponde al numero magico `143`, verrà stampato sul monitor seriale `"Nessuna Master Card definita"` e `"Scansione di una PICC per definire una Master Card"`.

Nel ciclo `do … while` l'istruzione `successRead=getID()` imposta la variabile `successRead` a `1` quando legge dal lettore, altrimenti è `0`. Contemporaneamente lampeggia il LED blu per una volta. Il programma non va oltre se non ottiene una lettura.

Il ciclo `for` scrive un UID PICC nella EEPROM, a partire dall'indirizzo 3 e con l'istruzione `EEPROM.write(2+ j, readCard[j])`; poi lo scrive nella EEPROM definita come Master Card all'indirizzo 1 e nel monitor seriale appare `"Master Card definita"`.

Il ciclo `for` legge l'UID della Master Card dalla EEPROM e lo scrive nella variabile `MasterCard`. Sul monitor seriale e nell'LCD viene stampato `"Tutto pronto!"` e `"In attesa di una PICC da sottoporre a scansione"` (Figura 9.4).

La funzione `cicloLed()` accende ciclicamente i LED RGB.

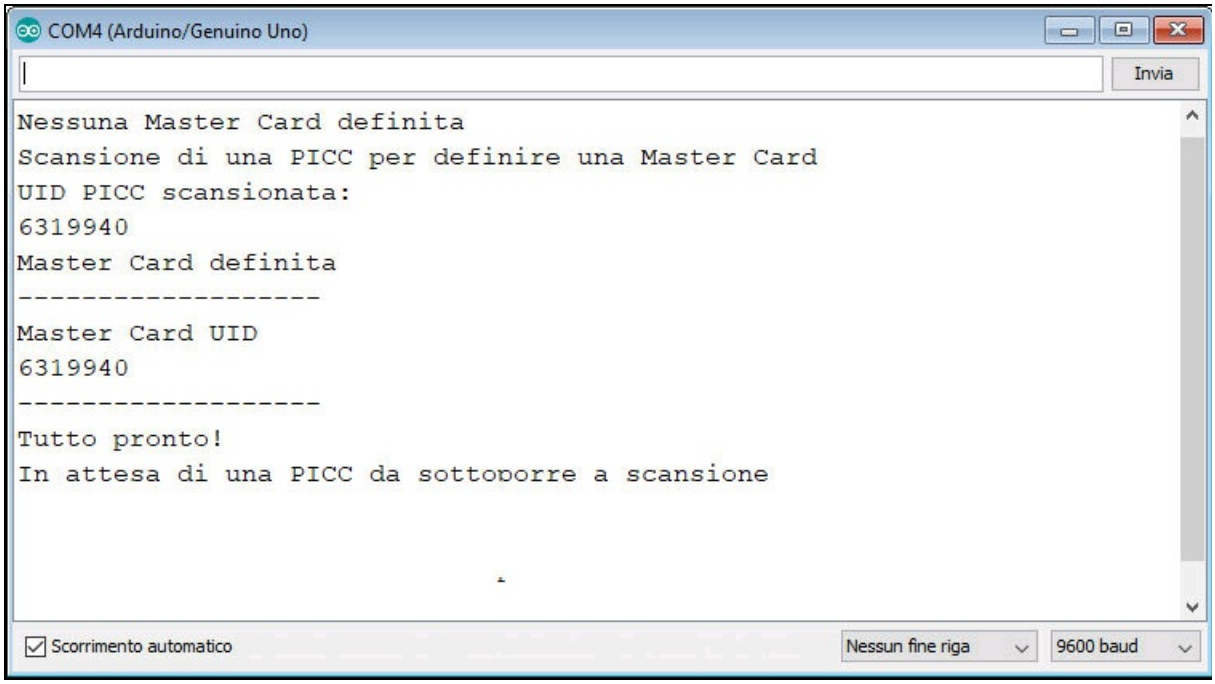


Figura 9.4 Definizione di una Master Card.

Modalità Programma

Per entrare in modalità Programma, basta avvicinare la Master Card al lettore. Nel `loop()`, se è attiva la modalità Programma, la funzione resta in attesa di leggere una nuova card. I LED vengono accesi ciclicamente. Altrimenti si entra in modalità Normale con il LED blu acceso e gli altri spenti. Con `while not successRead`, il programma non va oltre se non ottiene una lettura.

Nel monitor seriale appare (Figura 9.5a):

- `"Master Card - Modo Programma attivato";`
- `"Sono presenti 0 record nella EEPROM";`
- `"Scansione di una card per aggiungerla o rimuoverla".`

Se `modoProgramma` è attivo e si avvicina ancora la Master Card, si esce dalla modalità Programma e apparirà `"Uscita dalla modalità Program Mode".`

Quindi apparirà `"Tutto pronto!"` per indicare che il lettore rimane in attesa di leggere un tag Slave (Figura 9.5b).

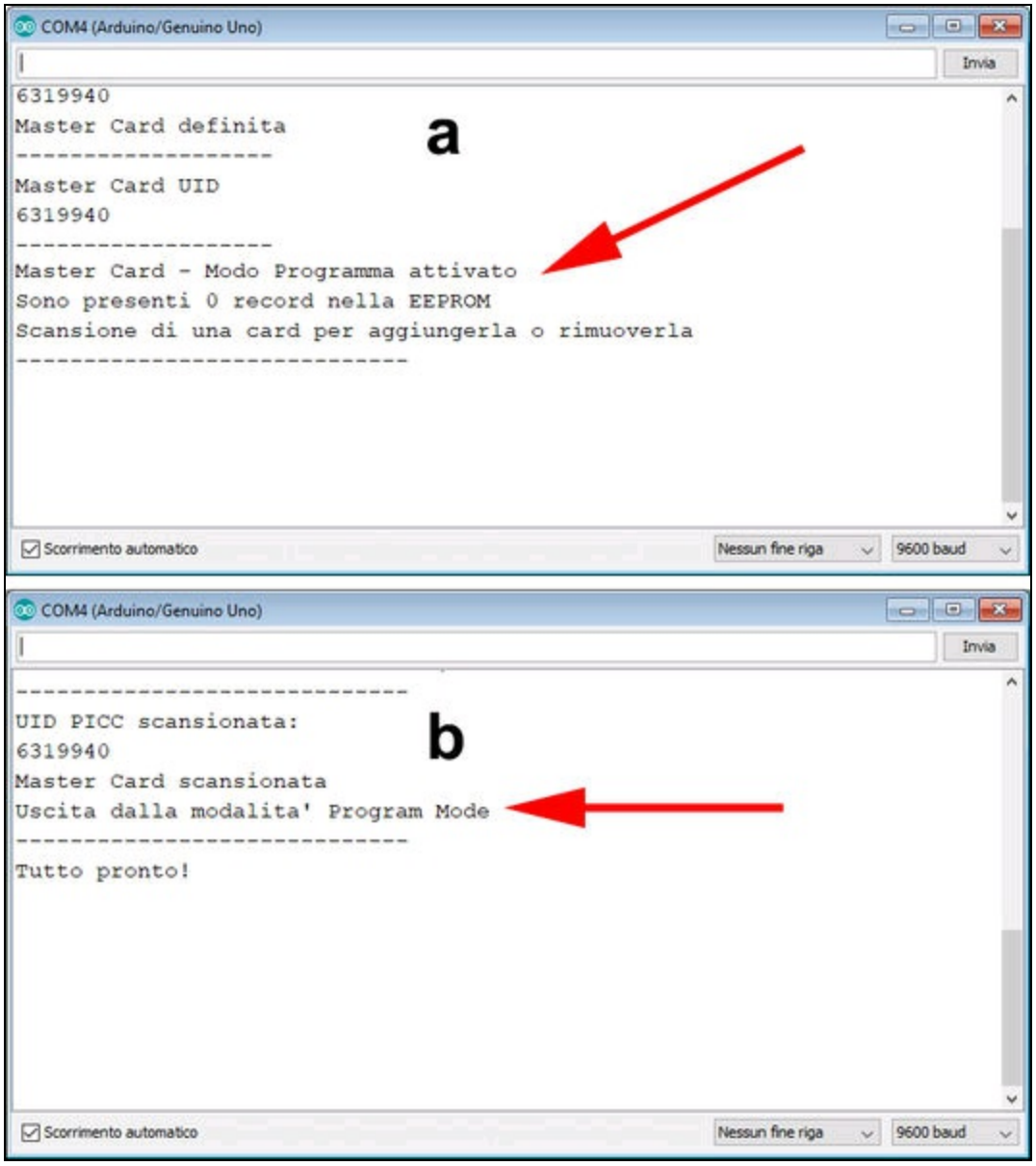


Figura 9.5 Modalità Programma (a). Uscita dalla modalità Programma (b).

Memorizzazione di un tag Slave

Una volta entrati in modalità Programma avvicinando la Master Card, si può aggiungere un tag Slave (il tag a forma di portachiavi contenuto

nel kit). Se il tag scansionato non è noto, viene aggiunto in memoria e apparirà `"Non conosco questa PICC, la aggiungo"` e `"Record aggiunto con successo alla EEPROM"` (freccia in alto di Figura 9.6).

Per terminare la modalità Programma, avvicinare la Master card e sul monitor apparirà (freccia in basso di Figura 9.6):

- `"Master card scansionata"`;
- `"Uscita dalla modalità Program Mode"`;
- `"Tutto pronto!"`.

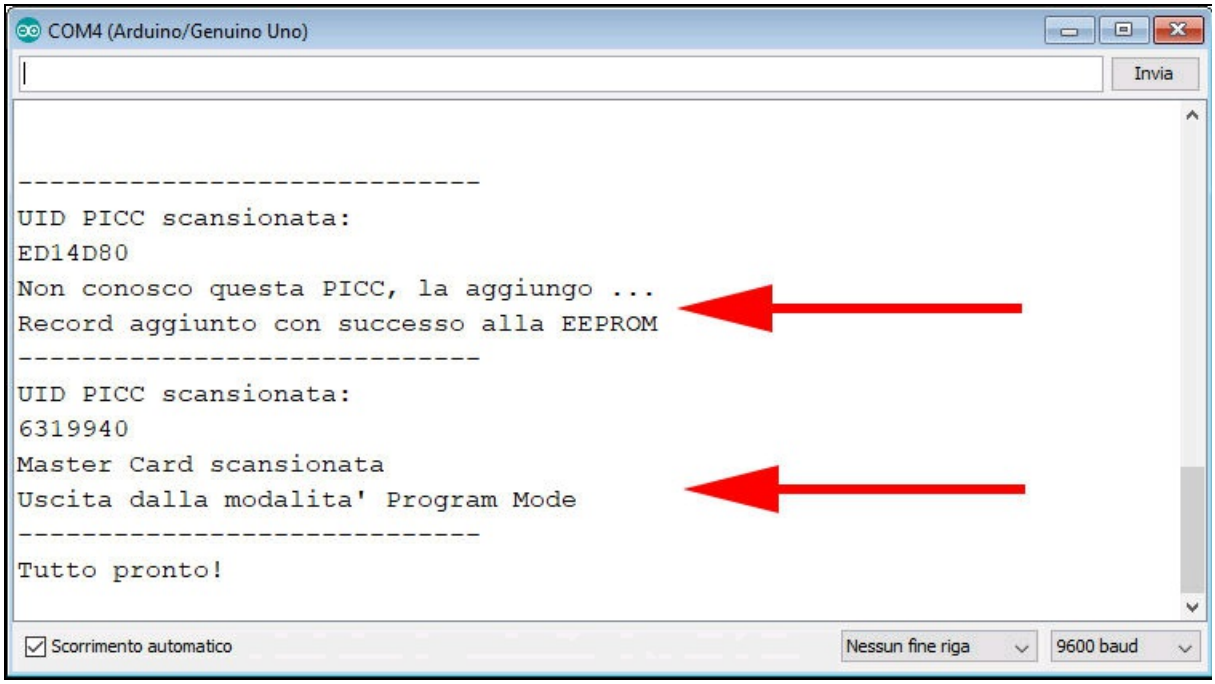


Figura 9.6 Memorizzazione di un tag sconosciuto nella EEPROM.

Riconoscimento di un tag memorizzato

A questo punto è davvero tutto pronto! Avvicinando il portachiavi al lettore, nel monitor seriale apparirà `"Accesso garantito"` e sul display apparirà `"Accesso OK!"` (Figura 9.7a).

Viene poi richiamata la funzione `accessoGarantito(3000)`, che accende il led rosso sulla porta A3 o, in alternativa, un modulo relè per 3 secondi.

Se il tag non è stato memorizzato nella EEPROM o è stato eliminato, apparirà sul monitor seriale `"Accesso negato"` e anche sul display apparirà `"Accesso negato!"` (Figura 9.7b).

Viene chiamata la funzione `accessoNegato(3000)`, che tiene spento il LED/relè per 3 secondi. Dopodiché il ciclo ricomincia e rimane in attesa di un’azione dell’utente.

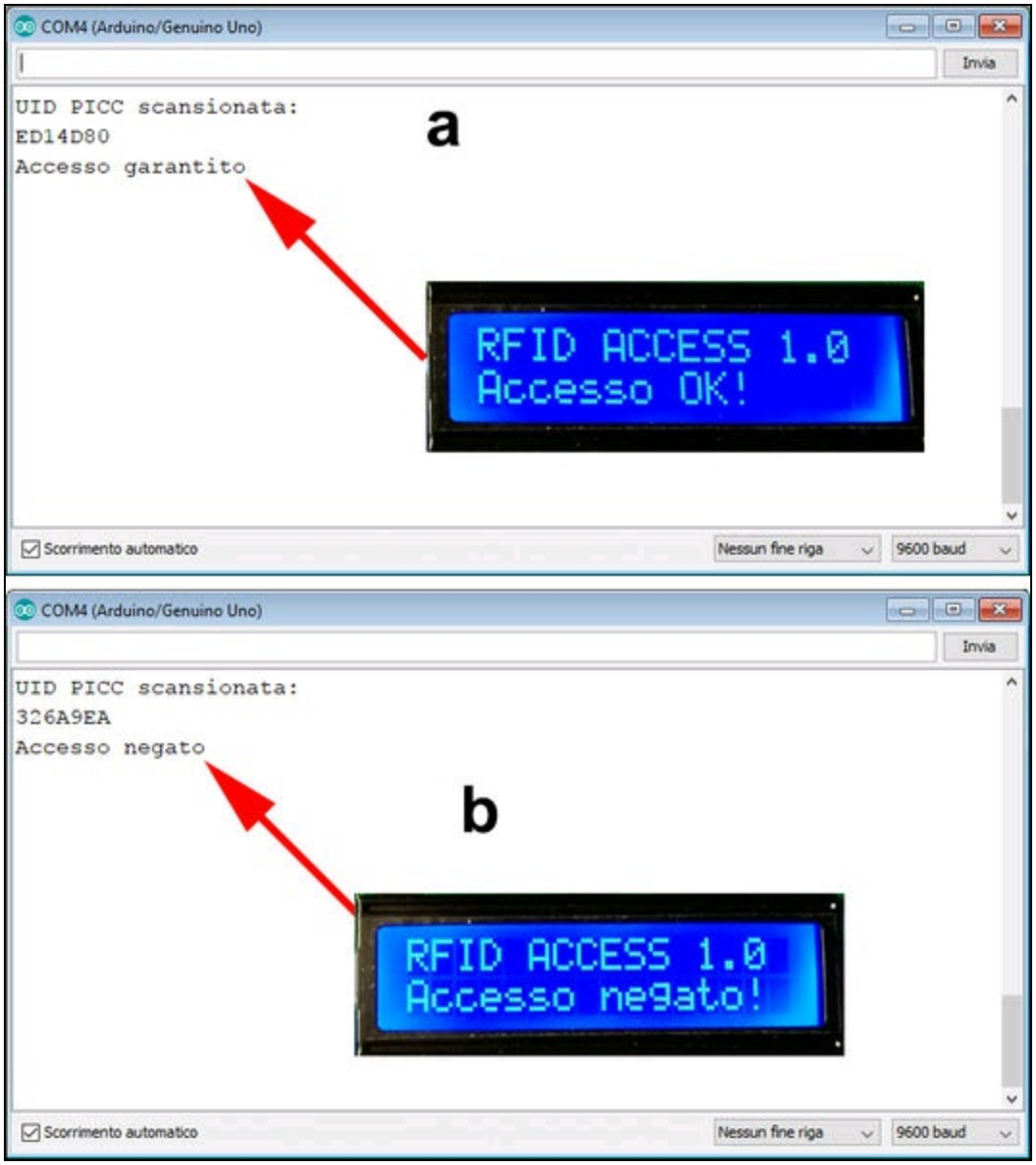


Figura 9.7 Accesso garantito (a). Accesso negato (b).

Procedura passo passo per il controllo accessi RFID

Abbiamo pensato di fare cosa utile riportare l’elenco di tutte le operazioni da seguire sia sul monitor seriale sia sul display. In seguito, dopo aver imparato come si fa, si potrà seguire solo il display.

Prima fase: definizione di una Master Card

1. Aprire il monitor seriale.
2. Riavviare Arduino tenendo premuto il tasto Reset per 5 secondi.
3. Verrà avviata la cancellazione della EEPROM.
4. A cancellazione avvenuta, il programma resta in attesa per definire una Master Card.
5. Avvicinare al lettore la card bianca del kit.
6. Questa viene definita come Master Card.

Una volta definita la Master Card, si possono seguire le operazioni anche sul display.

Seconda fase: definizione del portachiavi

1. Avvicinare al lettore un portachiavi (tag Slave) per aggiungerlo in memoria.
2. Avvicinare la Master Card al lettore per uscire dalla modalità Programma.
3. Apparirà `"Tutto pronto!"`.
4. Sul display apparirà `"Tutto pronto!"` (Figura 9.8a).

Terza fase: riconoscimento del portachiavi

1. Avvicinare il portachiavi appena memorizzato.
2. Apparirà `"Accesso garantito"`.
3. Sul display apparirà `"Accesso OK!"`.

Quelle che seguono sono operazioni per la gestione dei tag Slave.

Aggiungere ulteriori portachiavi

1. Avvicinare la Master Card per entrare in modalità Programma.

2. Sul display apparirà `"ENTER PROGRAM!"` (Figura 9.8b).
3. Il lettore rimane in attesa.
4. Avvicinare un nuovo portachiavi per memorizzarlo.
5. Sul display apparirà `"ADD CARD…"` (Figura 9.8c).
6. Avvicinare la Master Card per uscire dalla modalità Programma.
7. Sul display apparirà `"EXIT PROGRAM"` (Figura 9.8d).
8. Apparirà `"Tutto pronto!"`.

Eliminare un portachiavi

1. Avvicinare la Master Card per entrare in modalità Programma.
2. Il lettore rimane in attesa.
3. Avvicinare un portachiavi già memorizzato per eliminarlo dalla memoria.
4. Sul display apparirà `"DELETE CARD…"` (Figura 9.8e).
5. Avvicinare la Master Card per uscire dalla modalità Programma.
6. Apparirà `"Tutto pronto!"`.

La funzione accessoGarantito

Questa funzione riceve il dato di 3000 ms per tenere il LED/relè attivato per tre secondi tramite l'istruzione `digitalWrite(relay, HIGH)`. Nel frattempo si accende anche il LED RGB verde. Dopodiché appare sul display la scritta `"IN ATTESA…"` (Figura 9.8f) e il LED/relè viene disattivato. Si può modificare il tempo di accensione per adattarlo alla serratura elettrica per un progetto di serratura RFID.

La funzione accessoNegato

Questa funzione riceve il dato di 3000 ms per tenere il LED o il relè disattivato attivando solo il led RGB rosso per tre secondi, per poi

tornare a visualizzare la scritta “IN ATTESA…”. Anche in questo caso è possibile modificare il tempo di spegnimento.

Figura 9.8 Le funzioni riportare sul display.

Apertura cancello da smartphone

Descrizione

Il nome scelto per questo progetto è fuorviante. L'idea di aprire il cancello di casa con uno smartphone si può applicare a qualsiasi altro dispositivo: con lo stesso circuito si può spegnere/accendere l'illuminazione di una o più stanze, il riscaldamento, il condizionatore, la TV e quant'altro possa essere connesso a un relè.

Questo progetto impiega una scheda Arduino UNO oppure una scheda Arduino 101.

Il progetto con Arduino UNO si basa su un modem Bluetooth esterno molto diffuso. Quello con Arduino 101 sfrutta il modem Bluetooth incorporato. Entrambe le piattaforme sono valide e possono soddisfare lo scopo del nostro progetto. Per quanto riguarda la programmazione, il progetto viene proposto su tre livelli.

- *Livello base*: utilizzo di Arduino UNO e app di terze parti pronte all'uso.
- *Livello medio*: utilizzo di Arduino 101 e l'app dedicata nRF Connect.
- *Livello avanzato*: creazione di un'app Bluetooth con Visual Studio 2017.

Premesso che non vogliamo obbligare nessuno a diventare un maker esperto (anche se l'invito è questo), lasciamo al lettore la scelta della

soluzione più consona. Con questo in mente, il progetto verrà spiegato nella maniera più semplice possibile, affinché chiunque si senta soddisfatto del risultato raggiunto.

CODICE DI ESEMPIO

Tutti i file del progetto “Apertura cancello da smartphone” sono disponibili nelle risorse del libro presso il sito dell’autore all’indirizzo http://www.pierduino.com.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Arduino UNO;
- 1× scheda Arduino 101;
- 1× modulo Bluetooth HC-05 (o HC-06);
- 1× modulo relè.

Livello base

L’approccio più semplice, probabilmente, è quello con Arduino UNO e un modulo Bluetooth HC-05. Dei vari moduli della famiglia HC, il modulo HC-05 è probabilmente il più diffuso in ambiente Arduino. Può essere implementato in un progetto di controllo remoto, grazie alla facilità d’uso e a una connettività Bluetooth di buon livello, nonostante il suo basso costo.

Modulo Bluetooth HC-05

Di solito il modulo viene venduto già saldato su scheda breakout, dotata di 6 connettori a 90º e un pulsante *Enable*. Il pulsante serve solo per entrare in modalità programmazione AT (si veda la nota). Esiste anche il fratello minore, che si chiama HC-06, senza pulsante *Enable*. La programmazione può essere effettuata direttamente con comandi AT (si veda la nota che segue). La Figura 10.1 illustra un modulo HC-05 dotato di pulsante e un modulo HC-06 senza pulsante e le relative piedinature.

Caratteristiche principali.

- Sensibilità tipica: -80dBm.
- Potenza di trasmissione fino a + 4dBm RF.
- Funzionamento con alimentazioni da 1,8 a 3,6 volt.
- Controllo PIO.
- Interfaccia UART con baud rate programmabile.
- Antenna integrata.
- Connettore incorporato.
- Velocità di trasmissione predefinita: 38400 baud (bit di dati 8, bit di stop 1, parità nessuna).
- Velocità di trasmissione supportate: 9 600, 19 200, 38 400, 57 600, 115 200, 230 400, 460 800.

- Connessione automatica all’ultimo dispositivo in modalità di alimentazione come impostazione predefinita.
- Permette di collegare il dispositivo di associazione come impostazione predefinita.
- Password di accesso: “1234” come impostazione predefinita (configurabile).

NOTA

Per praticità, in questo progetto il modem HC-05 verrà usato con le impostazioni predefinite, ovvero con baud rate a 38400 e con il codice di accesso 1234 preimpostato. Per un setup personalizzato del modem si veda lo sketch per Arduino “BLUETOOTH_SETUP_HC_05” nelle risorse del libro, in cui sono descritte le procedure per modificare il nome del dispositivo, la password di accesso e la velocità di trasmissione, utilizzando semplici comandi AT. È presente anche lo schema di collegamento per il setup e il pdf di tutti i comandi AT.

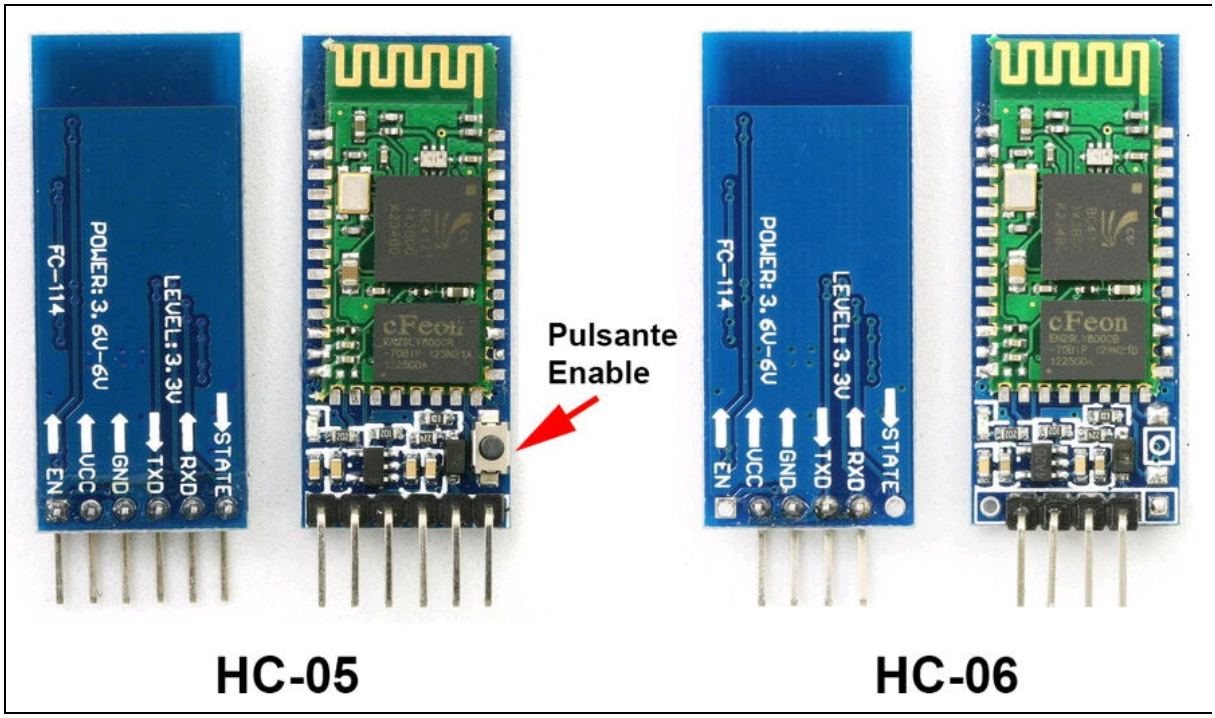


Figura 10.1 Un modulo HC-05 con pulsante Enable (a) e un modulo HC-06 senza (b).

Il circuito

Lo schema di collegamento è illustrato nella Figura 10.2. Dato l'esiguo numero di componenti e di collegamenti, non vale la pena usare una breadboard. Si può collegare il modem Bluetooth direttamente all'alimentazione 3.3 V e GND di Arduino. Il pin TX del modem va collegato al pin RX di Arduino. Il modulo relè va connesso al pin 13 e all'alimentazione 5 V e GND di Arduino. Ai contatti C (*Common*) e NO (*Normally Open*) del relè va connessa una linea di alimentazione qualsiasi (per esempio 12 VDC o 230 VAC), mentre l'altra linea di alimentazione va collegata all'attuatore, come il motore del cancello o il dispositivo che si vuol controllare.

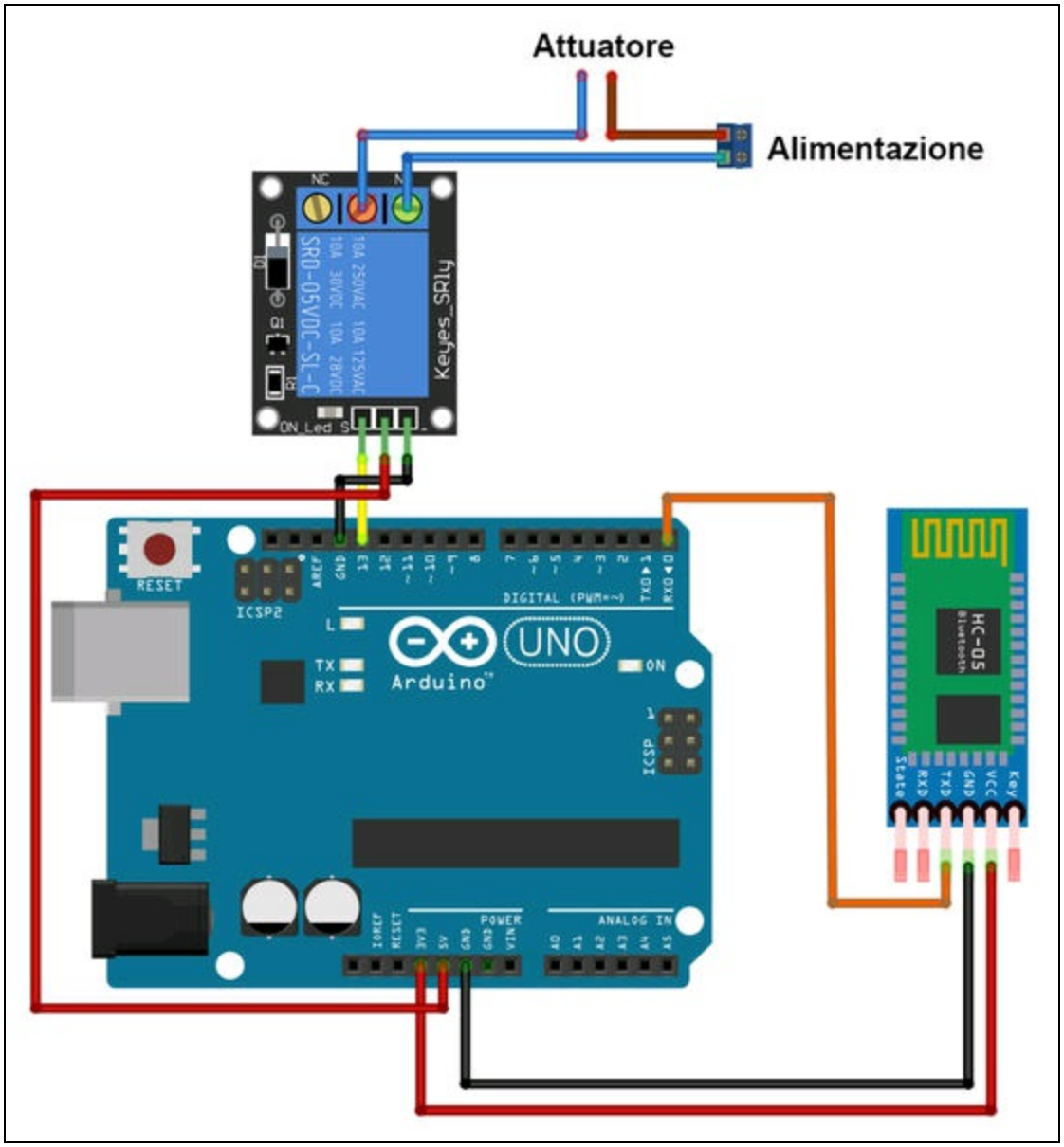


Figura 10.2 Schema di collegamento fra Arduino UNO e HC-05.

Associazione del modem Bluetooth

Per attivare una connessione fra uno smartphone e un’app Bluetooth, è necessario prima associare il modem, aprendo le impostazioni Bluetooth dello smartphone. Una volta alimentato il modem HC-05, si vedrà apparire `"HC-05"` fra i nomi dei dispositivi rilevati. Facendo clic sul

nome HC-05, apparirà la schermata per l'immissione del codice di accesso, che di default è 1234. Una volta associato il modem, qualsiasi app Bluetooth potrà connettersi.

Il codice

Fra le centinaia di app Bluetooth di terze parti disponibili online abbiamo scelto *ArduDroid*, by TechBitar, non fosse altro che è una delle storiche app Bluetooth per Arduino e perché resta ancora una delle migliori. Il sito di riferimento di TechBitar è il seguente: `http://www.techbitar.com/ardudroid-simple-bluetooth-control-for-arduino-and-android.html`. Sul sito, oltre al codice per Arduino da scaricare gratuitamente, c'è il link a Google Play per installare l'app su uno smartphone Android.

ArduDroid è in grado di controllare 12 pin digitali, di mandate testi e mandare dati verso Arduino e anche riceverli. Come si può vedere nella Figura 10.3, è possibile controllare 6 pin PWM tramite sei slider e 12 pin digitali con 12 tasti On/Off. Per controllare il nostro relè, basta premere il tasto 13 per attivarlo e ripremerlo per disattivarlo. Volendo, dalla stessa app, è possibile controllare altri 11 dispositivi collegati ad altrettanti altri pin di Arduino. Una volta caricato lo sketch dal sito (disponibile anche nelle risorse del libro), l'unica riga da cambiare è nel `setup()`:

```
Serial.begin(9600);
```

… che va cambiata in

```
Serial.begin(38400);
```

Questo perché, lo ricordiamo, il baud rate predefinito del modem è 38400.

Attenzione! Prima di caricare lo sketch ricordarsi di scollegare il filo che va al pin RX di Arduino. Una volta terminato il caricamento va

ricollegato. Come già detto altrove, il caricamento degli sketch occupa la porta RX dell'interfaccia UART.

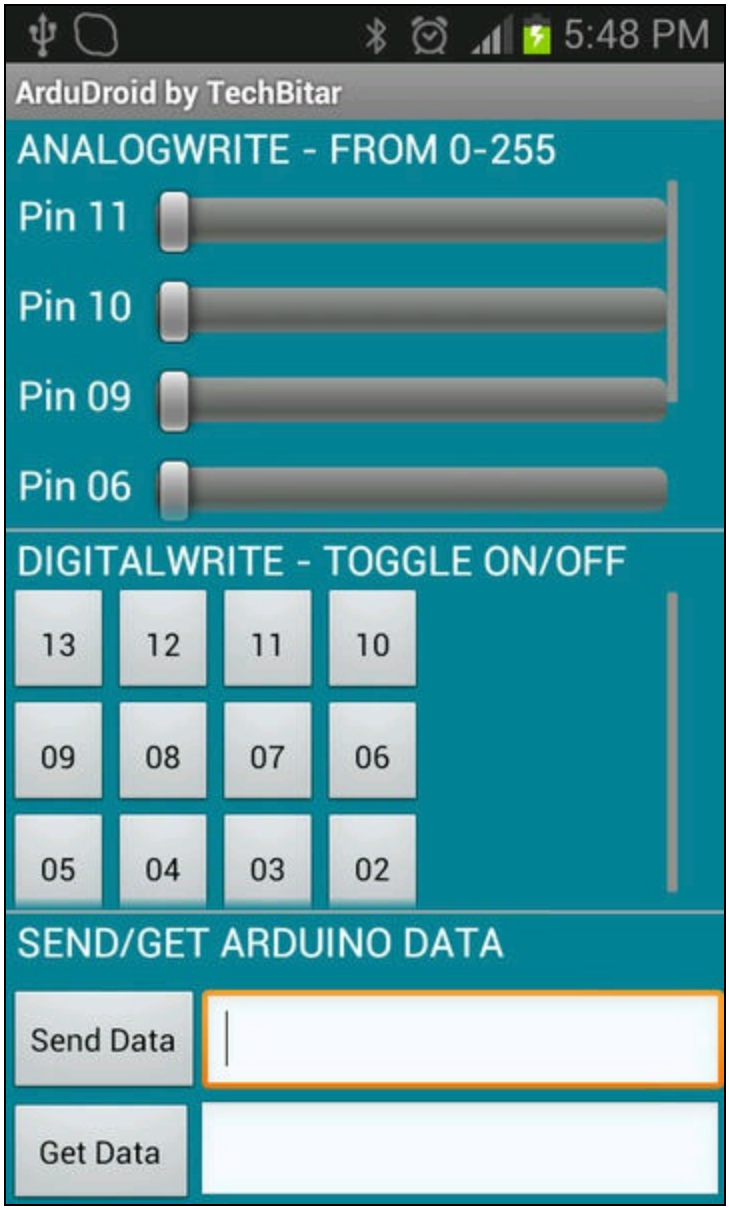


Figura 10.3 La schermata principale di ArduDroid.

Livello medio

Abbiamo già accennato alla scheda Arduino 101 nel Capitolo 2, una scheda che sfrutta le prestazioni avanzate del modulo Intel Curie a 32 bit. Il fattore di forma è identico alla scheda Arduino UNO, con l'aggiunta delle funzionalità Bluetooth e di un accelerometro/giroscopio a sei assi.

Caratteristiche principali:

- 14 pin ingressi / uscite digitali;
- 4 uscite PWM;
- 6 ingressi analogici;
- connettore USB per la comunicazione seriale;
- presa di alimentazione;
- connettore ICSP con SPI e pin dedicati I^2C;
- tensione di funzionamento e I/O della scheda TTL 3,3 V;
- tutti i pin sono protetti contro la sovratensione di 5 V.

Si tenga presente verrà usata solo la funzionalità Bluetooth e non verrà impiegato l'accelerometro/giroscopio.

Installazione della scheda Arduino 101

Prima di poter selezionare la scheda dal menu *Strumenti*, accertarsi di averla aggiunta dal gestore schede. Per i dettagli su come aggiungere una scheda si veda il paragrafo "Gestore schede" del Capitolo 3. Arduino 101 fa parte del gruppo di schede *Intel Curie Boards by Intel*. Se non è stata installata, bisogna provvedere.

Una volta installata la scheda, sarà disponibile nel menu *Strumenti* nella sezione *Intel Curie (32-bit) Boards* (Figura 10.4a). Dal menu *Esempi per Arduino/Genuino 101 > CurieBLE > Peripheral*,

selezionare lo sketch “LED” (Figura 10.4b), quindi selezionare la porta di comunicazione assegnata alla scheda e caricare lo sketch così com’è.

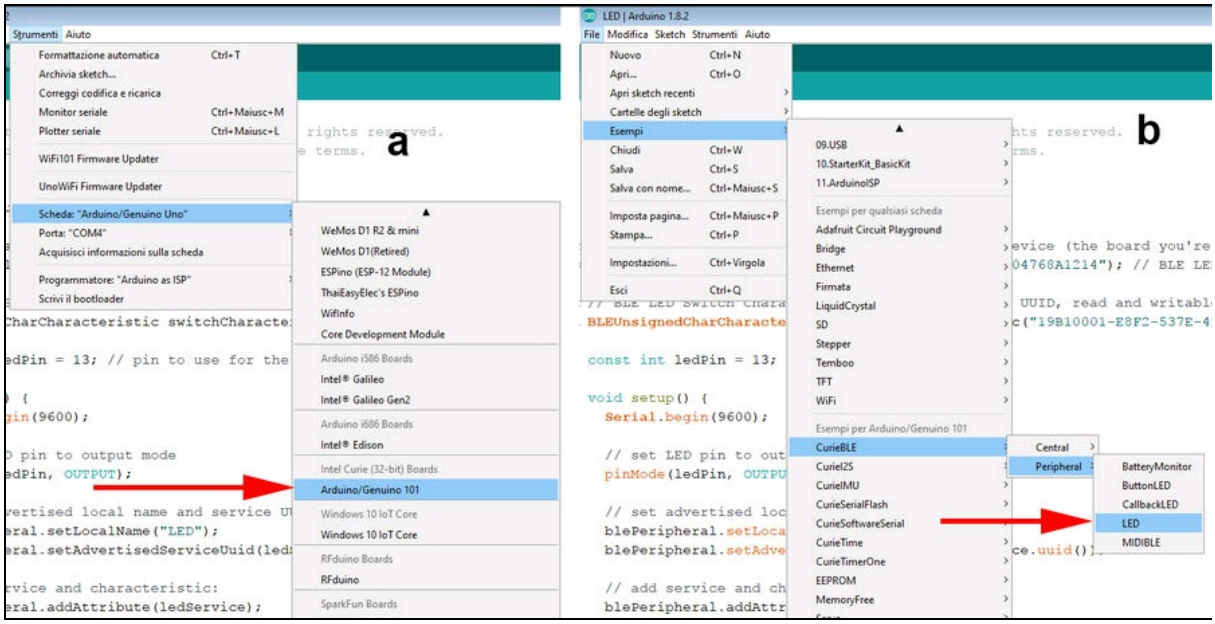


Figura 10.4 Il menu Strumenti con la scheda Arduino 101 selezionata (a). Il menu esempi della scheda (b).

Il codice

Il codice dello sketch è molto semplice ed è già impostato per accendere e spegnere il LED/relè collegato al pin 13. Vista la semplicità del codice, evitiamo di commentarlo tutto. Basti sapere che lo sketch attiva un `BLEService` con l’istanza `ledService` e `UUID` seguente:

```
BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214").
```

Questo `BLEService` verrà riconosciuto dall’app (si veda più avanti).

Nella funzione `loop()` è prevista la ricezione sulla porta seriale di qualsiasi valore inviato dall’app appositamente creata allo scopo e che si chiama nRF Master Control Panel. L’app è disponibile per Android e iOS ed è stata creata da Nordic Semiconductor, il produttore del chip Bluetooth che è stato implementato in Arduino 101. L’indirizzo per

scaricarla da Google Play è il seguente:

https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp.

L'indirizzo per scaricarla da Apple Store è il seguente:

https://itunes.apple.com/us/app/nrf-master-control-panel-ble/id1054362403.

nRF Master Control Panel

A dirla tutta, l'app non ha un aspetto molto accattivante e non è affatto facile usarla senza una guida. Cercheremo di dare alcune indicazioni in più, oltre a quelle che sono disponibili nel sito ufficiale Arduino alla pagina del prodotto: https://www.arduino.cc/en/Tutorial/Genuino101CurieBLELED. Per poter usare in modo soddisfacente l'app e controllare più facilmente il nostro LED/relè, si consiglia di impiegare le procedure seguenti.

Procedura per accendere/spegnere il LED

1. Dopo aver lanciato l'app *nRF Master Control Panel*, toccare il tasto *SCAN* per avviare la scansione dei dispositivi Bluetooth.
2. Dovrebbe apparire il dispositivo chiamato LED (il nome dello sketch) con vicino un pulsante *OPEN TAB* (Figura 10.5a).
3. Nella parte superiore della schermata seguente, toccare *CONNECT* per connettersi al dispositivo LED (Figura 10.5b). Qui verrà visualizzata la descrizione del servizio BLE istanziato da Arduino 101. Il cosiddetto servizio sconosciuto (`Unknown Service`) ha lo stesso UUID 19B10000-E8F2-537E-4F6C-D104768A1214 che è stato impostato dalla precedente istruzione `BLEService ledService ("19B10000-E8F2-537E-4F6C-D104768A1214")` dello sketch LED.
4. Toccare la scheda *Unknown Service* per aprire *Unknown Characteristic* (Figura 10.5c). Come si può vedere, le proprietà

della “caratteristica sconosciuta” sono `READ` e `WRITE`, cioè di lettura e scrittura. Accanto a queste ci sono due icone con una freccia in giù e una in su.

5. Toccare l’icona con la freccia in su (indicata nella Figura 10.5c).
6. Si aprirà una schermata (Figura 10.5d) in cui è possibile inserire un valore da scrivere sul dispositivo remoto. Selezionare dal menu il tipo di variabile, per esempio, `UINT 8` e inserire il valore `1` nel campo `Write value`, tramite il tastierino sottostante.
7. Toccare SEND per inviare il valore.
8. Il LED della scheda Arduino 101 si accenderà.

Per spegnere il LED, basta ripetere le operazioni dal passo 5 al passo 7, inserendo il valore 0 al passo 6.

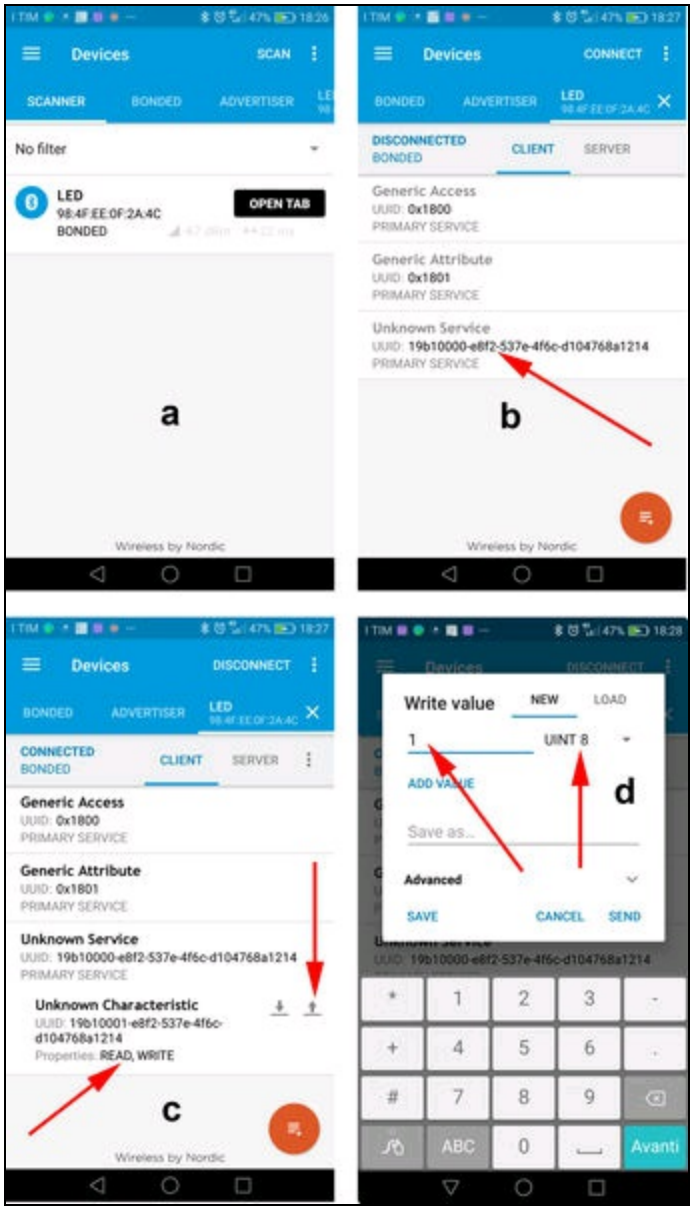


Figura 10.5 Passaggi per accendere e spegnere un LED.

Procedura per registrare una macro

Le istruzioni del sito ufficiale finiscono qui. La procedura per accendere e spegnere un LED non è certamente comoda, per cui pensiamo sia utile registrare una macro, visto che l'app rende disponibile questa possibilità.

1. Dalla schermata *Unknown Service*, toccare il tasto rosso con il menu (Figura 10.6a).
2. Apparirà il pannello *Macros* con tre icone: *Aggiungi cartella*, *Apri da…* e *Registra* (Figura 10.6b).
3. Creare una nuova cartella, toccando l'icona *Aggiungi cartella* e dare un nome, per esempio "Pier".
4. Toccare l'icona *Registra* per iniziare la registrazione della macro.
5. Ripetere tutte le operazioni della procedura precedente da 4 a 6 per accendere il LED, senza però toccare *SEND*.
6. Toccare invece il campo *Save as…* e digitare, per esempio, "Led on" (Figura 10.6c).
7. Toccare *SAVE*.
8. Si aprirà una schermata con l'elenco delle azioni.
9. Toccare *SEND*.
10. La schermata si chiuderà.
11. Toccare l'icona *Registra* per fermare la registrazione.
12. Apparirà una schermata simile alla Figura 10.6d in cui è possibile dare un nome alla macro registrata.
13. Digitare un nome, per esempio "Pier".
14. Scegliere anche un'icona fra quelle disponibili nel menu *Choose icon*, per esempio quella di un LED acceso.

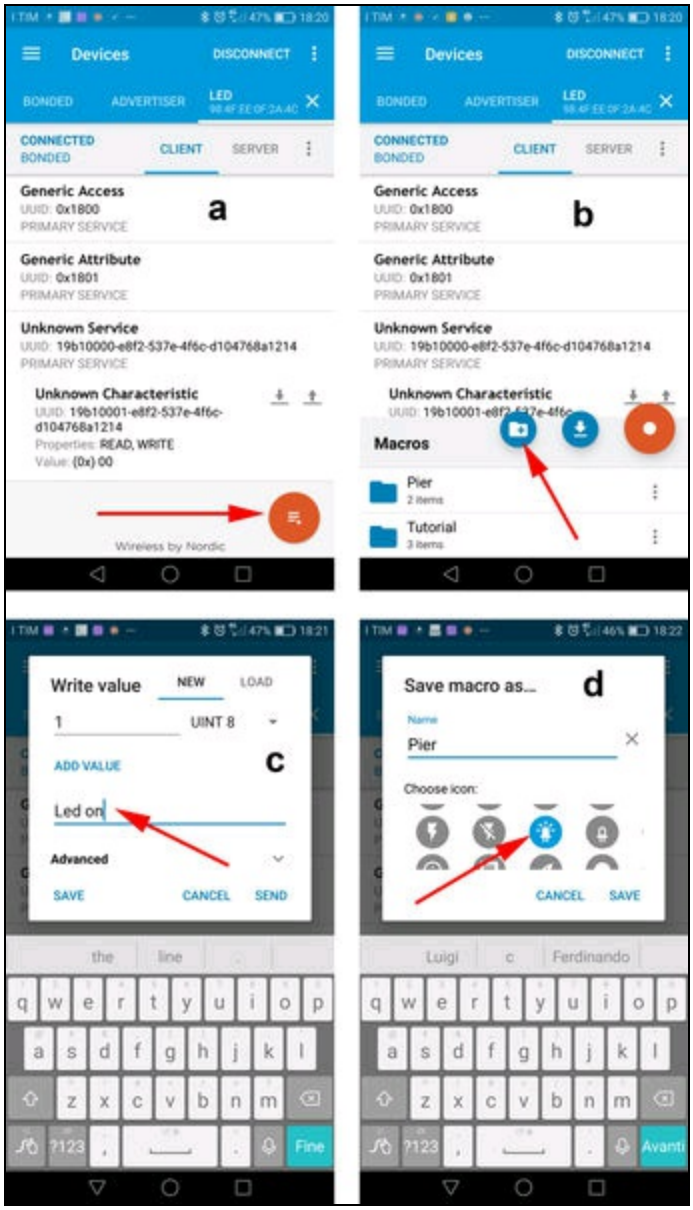


Figura 10.6 Passaggi per la registrazione di una macro.

A questo punto, ripetere la registrazione di una seconda macro, inserendo però il valore `0` e digitando “Led off” come nome dell’azione. Prima di salvare la seconda macro, scegliere anche l’icona di un LED spento.

Come attivare la macro

Dalla schermata *Macros* è possibile selezionare la cartella delle azioni registrate. Toccando la cartella, si possono visualizzare le due azioni appena registrate *Led off* e *Led on* (Figura 10.7). Toccando il simbolo *Play* (il piccolo triangolo) si attiva l'azione corrispondente. Accanto al simbolo *Play* è disponibile anche un menu (tre punti verticali) di operazioni disponibili sull'azione selezionata: *Share*, *Export to XML*, *Rename*, *Move* e *Mirror*.

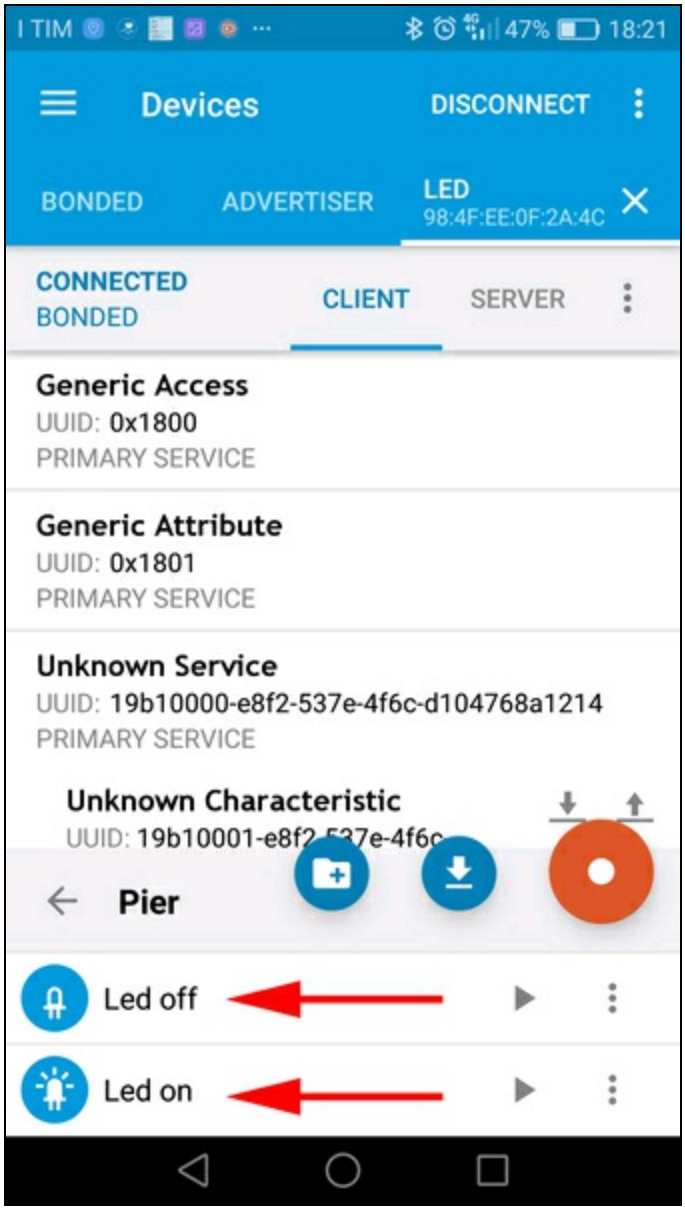


Figura 10.7 La macro contenente le due azioni Led off e Led on.

Collegamento del relè alla scheda Arduino 101

Terminiamo il progetto Bluetooth con Arduino 101 collegando un modulo relè al pin 13. La Figura 10.8 illustra il semplice collegamento da effettuare.

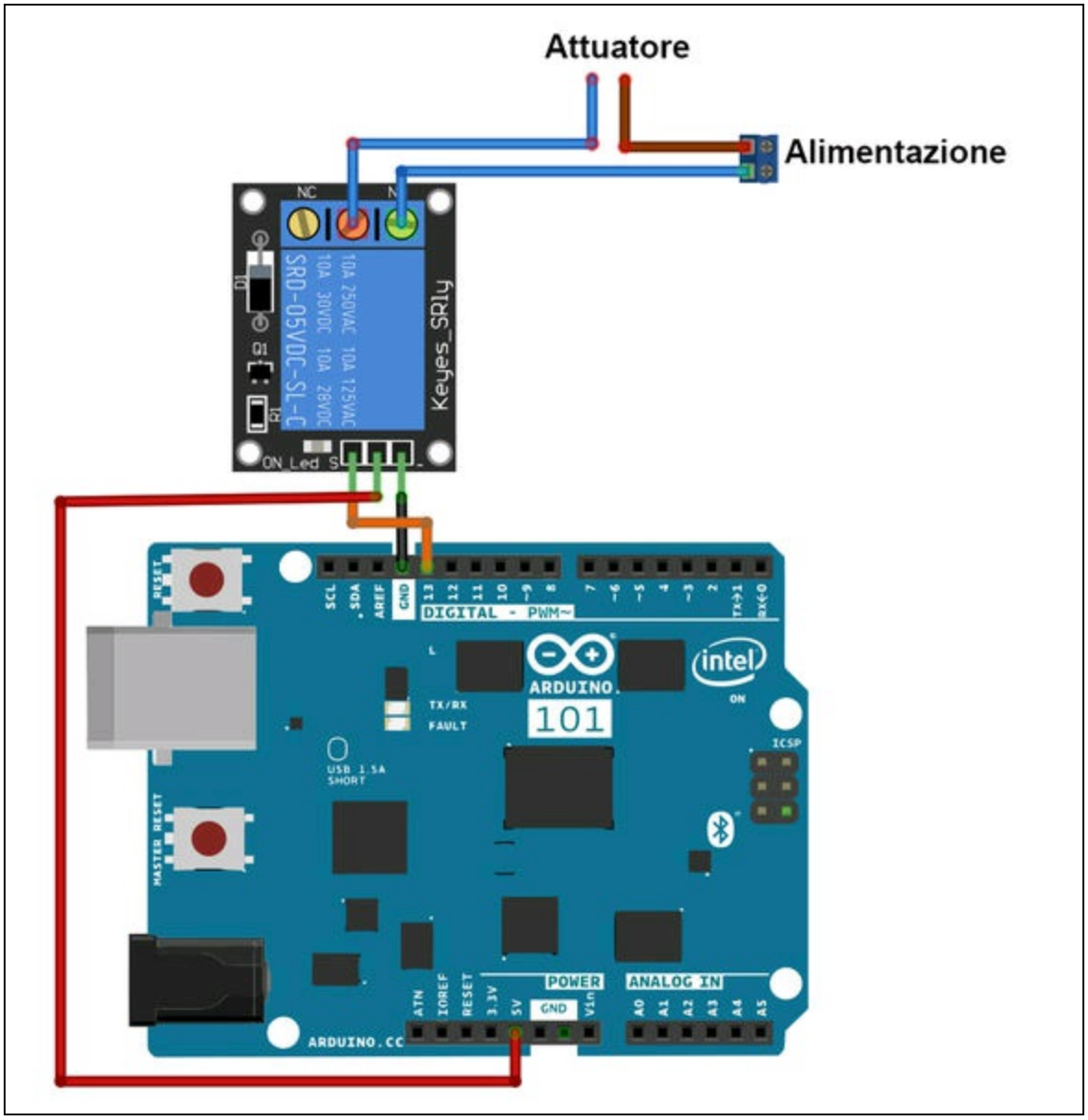


Figura 10.8 Schema di collegamento fra Arduino 101 e un modulo relè.

Livello avanzato

Quello che segue è un progetto ex novo di un'applicazione Bluetooth per Android, sviluppata con Visual Studio 2017. Chi vorrà cimentarsi nella programmazione più avanzata e seguire questa semplice guida, proverà una soddisfazione maggiore nel controllare la propria scheda Arduino usando un'app personalizzabile, invece di usarne una scaricata dal Web.

Se è stato installato tutto il pacchetto Visual Studio 2017, come suggerito nel Capitolo 4, saranno disponibili anche le piattaforme di sviluppo Android e iOS, ovvero il pacchetto installato come Xamarin. Per maggiori informazioni su Xamarin, il sito ufficiale è: `https://www.xamarin.com`.

Creazione di un'app Bluetooth con Visual Studio 2017

Dal menu *File > Nuovo progetto*, selezionare *Android* dall'elenco dei modelli e un'app vuota. Dare un nome all'app vuota, per esempio, *BlueApp*, come indicato nella Figura 10.9a.

Verrà creata una soluzione e un progetto con il nome "BlueApp", contenente un file in C#, chiamato `MainActivity.cs` (Figura 10.9b). Questo è il file che bisognerà modificare e in cui scrivere tutto il codice dell'applicazione. Aprendo questo file apparirà il codice in C# di un'applicazione vuota con una sola classe principale, `MainActivity`, all'interno del `namespace BlueApp`:

```
namespace BlueApp
{
    [Activity(Label = "BlueApp", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
{
protected override void OnCreate(Bundle bundle)
{
```

```
    base.OnCreate(bundle);
    // Set our view from the "main" layout resource
    // SetContentView (Resource.Layout.Main);
}
```

La cosa da sapere è che bisognerà scrivere il codice solo all'interno della classe `MainActivity`. Prima, però, è necessario creare l'interfaccia grafica dell'applicazione.

La cartella *Resources* contiene altre cartelle e file.

- `Drawable`: contiene l'icona di default.
- `Layout`: contiene il file `Main.axml`.
- `Values`: contiene il file `Strings.xml`.
- `AboutResources.txt`: è un file di testo con le istruzioni per le risorse.

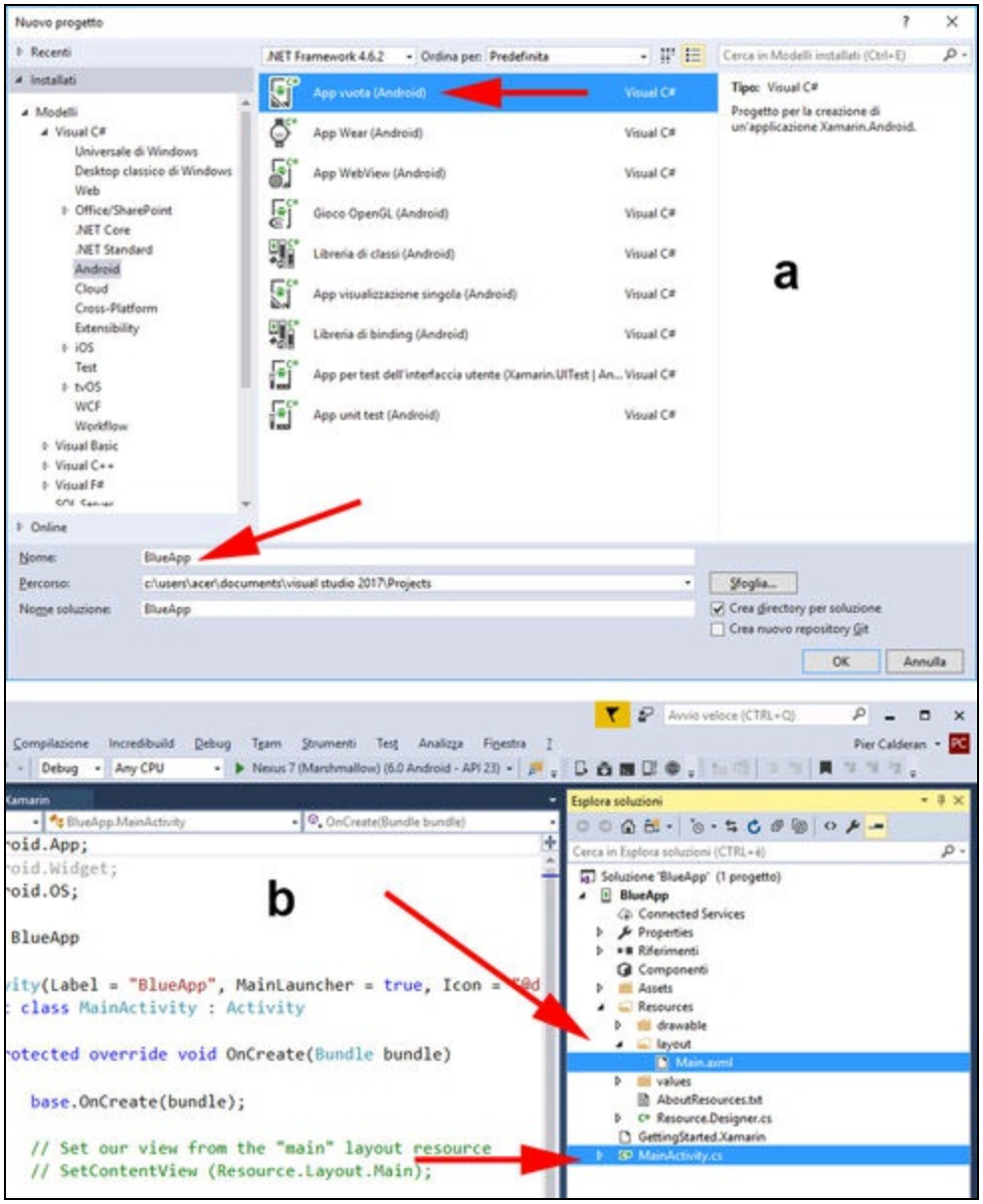


Figura 10.9 Finestra File > Nuovo progetto (a). Soluzione e progetto BlueApp (b).

Interfaccia grafica

Come in tutte le applicazioni scritte in ambiente C#, anche con Xamarin è possibile costruire l'interfaccia per il proprio smartphone.

NOTA

Se si utilizza l'emulatore Android SDK di Google, si consiglia di configurare l'emulatore per utilizzare l'accelerazione hardware. Le istruzioni per la

configurazione dell'accelerazione hardware sono disponibili a questo indirizzo: https://developer.xamarin.com/guides/android/getting_started/installation/acceler

Se si utilizza l'emulatore Android di Visual Studio, Hyper-V deve essere abilitato nel computer. Per ulteriori informazioni sulla configurazione di Visual Studio Android Emulator, consultare la pagina: https://msdn.microsoft.com/en-us/library/mt228280.aspx.

Il linguaggio del codice è molto simile a XAML, solo che è stato chiamato AXML perché è destinato esclusivamente a interfacce create per la piattaforma Android.

Innanzitutto, bisogna scegliere il dispositivo di destinazione dal menu in alto, come indicato dalla freccia a destra nella Figura 10.10a. Per installare un dispositivo virtuale fare clic su *Gestisci dispositivi virtuali…* (freccia in basso della Figura 10.10a). Si aprirà una la finestra *Xamarin Android Player* (Figura 10.10b) in cui si può scegliere il dispositivo virtuale di destinazione. Per il nostro progetto abbiamo scelto il dispositivo *Nexus 7 (Marshmallow)*, basato su Android 6.0 (API23). Dalla stessa finestra si possono installare altri dispositivi compatibili con versioni di Android precedenti o successive.

Una volta scelto il dispositivo di destinazione, con un doppio clic sul file `Main.axml`, si aprirà la finestra di progettazione con l'interfaccia del dispositivo completamente vuota e con il nome "BlueApp" sulla barra dell'applicazione (Figura 10.10b).

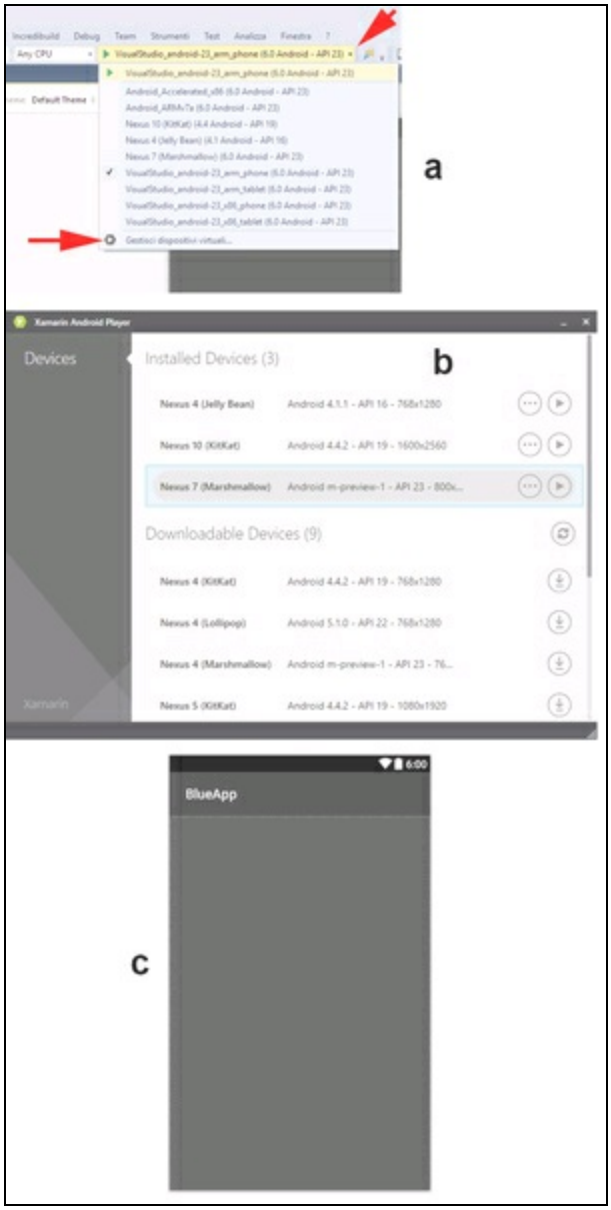


Figura 10.10 Menu del dispositivo di destinazione (a). Finestra Xamarin Android Player (b). Finestra di progettazione del dispositivo (c).

Procedura per aggiungere i componenti all'interfaccia

1. Aprire la casella degli strumenti dal menu *Visualizza*.
2. Selezionare il componente `TextView` e trascinarlo sull'interfaccia.

3. Dal menu *Proprietà* selezionare l'opzione *text* dalla sezione *Main* e digitare "LED CONTROL".
4. Dal menu *Proprietà* selezionare l'opzione *layout_marginLeft* dalla sezione *Layout - ViewGroup* e impostare il margine sinistro a 120dp. La Figura 10.11a illustra il risultato raggiunto fino a questo momento.
5. Selezionare il componente *TextView* e trascinarlo sull'interfaccia.
6. Dal menu *Proprietà* selezionare l'opzione *text* e digitare "Disconnected!".
7. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
8. Dal menu *Proprietà* selezionare l'opzione *text* e digitare "CONNECT".
9. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
10. Dal menu *Proprietà* selezionare l'opzione *text* e digitare "DISCONNECT".
11. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
12. Dal menu *Proprietà* selezionare l'opzione *text* e digitare "LED ON".
13. Dal menu *Proprietà* selezionare l'opzione *layout_marginTop* dalla sezione *Layout - ViewGroup* e impostare il margine in alto a 60dp.
14. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
15. Dal menu *Proprietà* selezionare l'opzione *text* e digitare "LED OFF".

La Figura 10.11b illustra il risultato finale. Terminata l'interfaccia, si può procedere alla scrittura del codice.

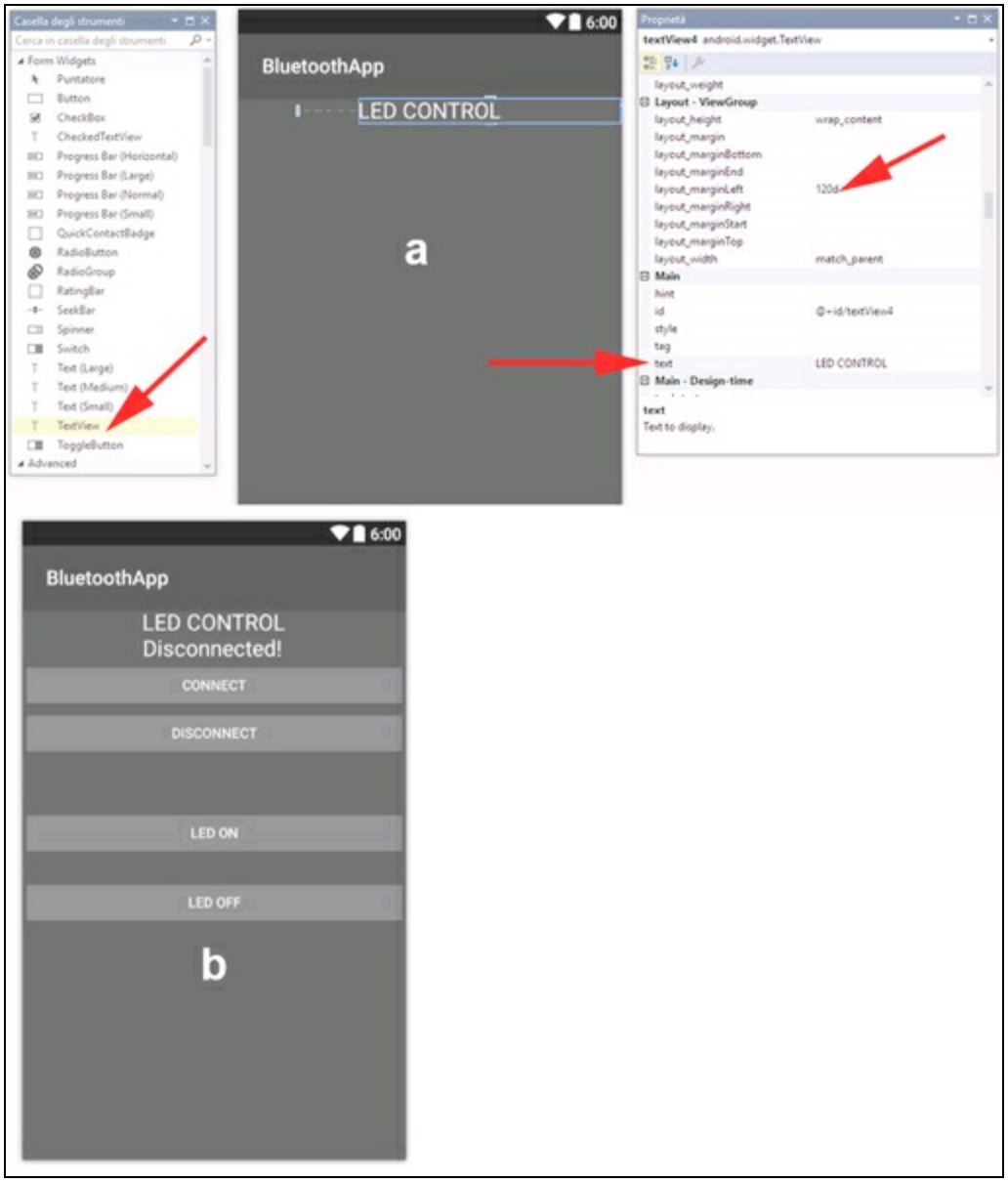


Figura 10.11 Casella degli strumenti, l'interfaccia del dispositivo e la finestra proprietà (a). Risultato finale (b).

Il codice C#

Per facilitare la comprensione e la modifica del listato in base alle necessità, vengono spiegate di seguito solo le funzioni essenziali.

Il file `MainActivity.cs` include alcune librerie Android, per cui bisogna aggiungere all'inizio del listato le seguenti direttive `using`:

```
using Android.App;
using Android.Widget;
using Android.OS;
using Android.Bluetooth;
```

In questo modo si attivano le funzionalità Android e soprattutto la funzionalità Bluetooth.

All'inizio della classe `public class MainActivity`, bisogna inserire la chiamata alla classe `BluetoothConnection`:

```
BluetoothConnection myConnection = new BluetoothConnection();
```

Questo significa che bisogna creare la classe omonima:

```
public class BluetoothConnection
{
    public void getAdapter() { this.thisAdapter = BluetoothAdapter.DefaultAdapter;
}
    public void getDevice() { this.thisDevice = (from bd in
this.thisAdapter.BondedDevices
        where bd.Name == "HC-05" select bd).FirstOrDefault(); }
    public BluetoothAdapter thisAdapter { get; set; }
    public BluetoothDevice thisDevice   { get; set; }
    public BluetoothSocket thisSocket { get; set; }
}
```

Si fa notare che il nome del dispositivo `HC-05` è quello predefinito del modem HC-05 usato con Arduino UNO. Se viene cambiato il nome, lo si dovrà cambiare anche qui.

All'interno della funzione `protected override void OnCreate(Bundle bundle)` si devono aggiungere le istruzioni per la gestione dei pulsanti che sono stati aggiunti prima all'interfaccia grafica. I nomi dei pulsanti e della casella di testo sono quelli di default: `button1`, `button2`, `button3`, `button4` e `textview2`. Per gestirli all'interno del listato, vengono tradotti rispettivamente come `buttonConnect`, `buttonDisconnect`, `buttonOn`, `buttonOff` e `connected`, utilizzando la funzione `FindViewById`.

```
Button buttonConnect = FindViewById<Button>(Resource.Id.button1) ;
Button buttonDisconnect = FindViewById<Button>(Resource.Id.button2);
Button buttonOn = FindViewById<Button>(Resource.Id.button3);
Button buttonOff= FindViewById<Button>(Resource.Id.button4);
TextView connected = FindViewById<TextView>(Resource.Id.textView2);
```

La funzione all’interno di `buttonConnect` tenterà la connessione Bluetooth:

```
buttonConnect.Click += delegate {
    myConnection = new BluetoothConnection();
    myConnection.getAdapter();
    myConnection.thisAdapter.StartDiscovery();
    …
    …
    myConnection.thisSocket.Connect();
    connected.Text = "Connected!";
    buttonDisconnect.Enabled = true;
    buttonConnect.Enabled = false;
```

Come si può vedere, una volta ottenuta la connessione, la casella di testo `connected` cambierà il testo in “Connected!” e disabiliterà il pulsante. In modo simile il pulsante `buttonDisconnect` esegue la disconnessione.

All’interno del pulsante `buttonOn`, il codice per attivare il LED è il seguente:

```
buttonOn.Click += delegate {
    try {
        myConnection.thisSocket.OutputStream.WriteByte(1);
        myConnection.thisSocket.OutputStream.Close();
    }
```

In pratica, viene mandato il byte 1 in streaming sul socket Bluetooth. Questo byte verrà ricevuto dal modem tramite la porta seriale Arduino, che, come vedremo, provvederà a leggere questo byte e a mandarlo al pin 13 per accendere il LED.

All’interno del pulsante `buttonOff`, il codice per disattivare il LED è il seguente:

```
buttonOff.Click += delegate {
    try {
        myConnection.thisSocket.OutputStream.WriteByte(0);
        myConnection.thisSocket.OutputStream.Close();
    }
```

In modo simile, viene mandato il byte 0 sul socket Bluetooth. Questo byte verrà ricevuto da Arduino, che provvederà a leggere il byte 0 e a mandarlo al pin 13 per spegnere il LED.

Compilazione e distribuzione dell’app

Terminata la scrittura del codice, si può procedere alla compilazione della soluzione e alla successiva distribuzione al dispositivo.

Prima di procedere alla compilazione, bisogna accedere ad alcune configurazioni. Dal menu *Debug*, selezionare *Proprietà* per aprire la finestra omonima ed eseguire le impostazioni seguenti.

1. Nella scheda *Applicazione* selezionare la versione di Android per la compilazione, per esempio Android 6.0 (Marshmallow) e la versione minima, per esempio Android 4.1 (livello API 16 - Jelly Bean) (Figura 10.12a).
2. Nella scheda *Manifesto Android* selezionare le funzionalità: BLUETOOTH, BLUETOOTH_ADMIN, BLUETOOTH_PRIVILEGED e impostare l'icona su *@drawableicon/Icon*, ovvero l'icona di default delle risorse (Figura 10.12b).

Non dovendo eseguire il debug del codice, si può impostare direttamente la configurazione della soluzione su *Release* (Figura 10.12c). Selezionare il dispositivo per cui effettuare la compilazione, per esempio, *Android 6.0 (Marshmallow)* e dal menu *Compilazione*, selezionare *Compila soluzione*. Se non ci sono errori la compilazione avviene senza problemi, altrimenti appariranno gli errori nella finestra di *Output*.

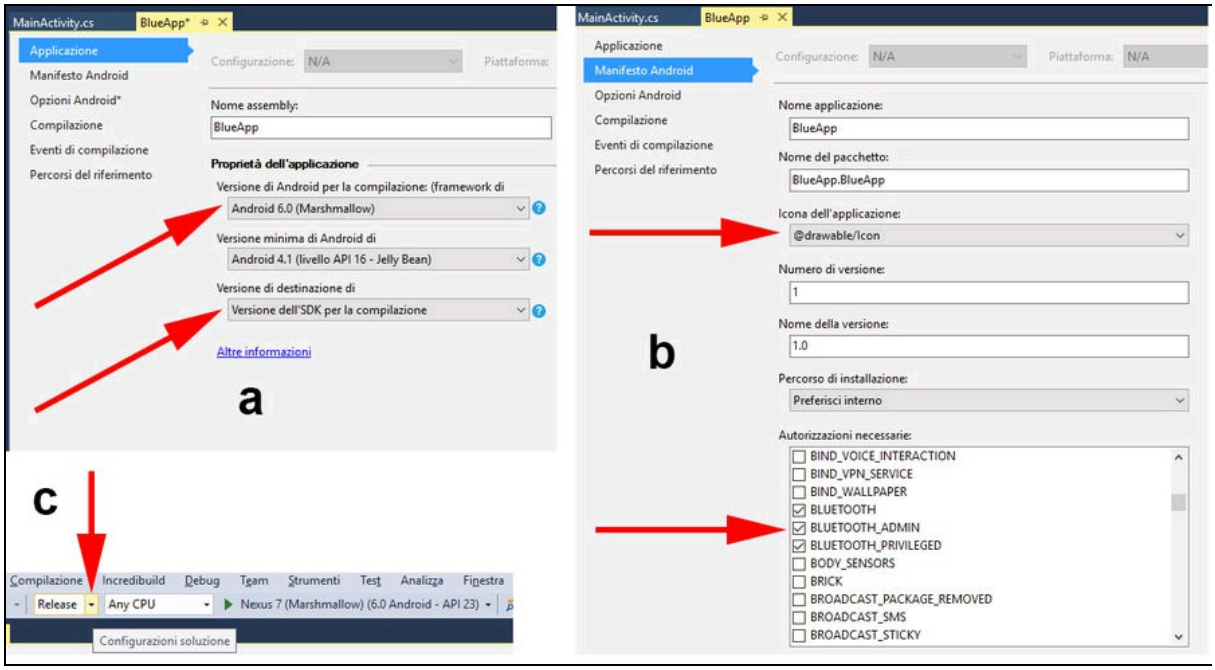


Figura 10.12 Finestra della scheda Applicazione (a) e della scheda Manifesto Android (b). Configurazione della soluzione (c).

Distribuzione

Per distribuire l'applicazione, cioè per esportare il pacchetto APK e poterlo installare sullo smartphone Android, impiegare la procedura seguente.

1. Dalla finestra *Esplora soluzioni* aprire con il tasto destro del mouse il menu contestuale e selezionare la voce *Archivia* (Figura 10.13a).
2. Si aprirà una finestra simile a quella rappresentata nella Figura 10.13b con una barra di progresso che indica lo stato della creazione del pacchetto BlueApp.
3. Al termine della creazione del pacchetto, fare clic sul pulsante *Distribuisci*.
4. Si aprirà una finestra in cui selezionare il canale di distribuzione, ovvero *Ad hoc* oppure *Google Play*. Non dovendo distribuire l'app su Google Play, selezionare il canale *Ad hoc*.

5. Si aprirà una finestra simile a quella rappresentata nella Figura 10.13c in cui selezionare una firma digitale dell'app. Fare clic sul pulsante + (più) per aggiungere una firma.
6. Si aprirà una finestra simile a quella rappresentata nella Figura 10.13d in cui si può creare una chiave. Compilare i seguenti campi.
   - *Alias*: nome della chiave.
   - *Password*: una password di almeno 8 caratteri/numeri.
   - *Validità*: lasciare 30 anni o quel che si vuole.
   - Nei campi sotto la sezione *Immettere almeno uno dei valori seguenti*, immettere i dati relativi al nome del produttore, organizzazione, città e così via.
7. Fare clic sul pulsante *Crea*.
8. Si tornerà alla finestra precedente.
9. Selezionare il nome della chiave appena creata.
10. Fare clic sul pulsante *Salva con nome*.
11. Mentre una barra di progresso indica lo stato del salvataggio si aprirà una finestra che chiederà di inserire una password per la firma (quella scelta nella finestra precedente) (Figura 10.13e).

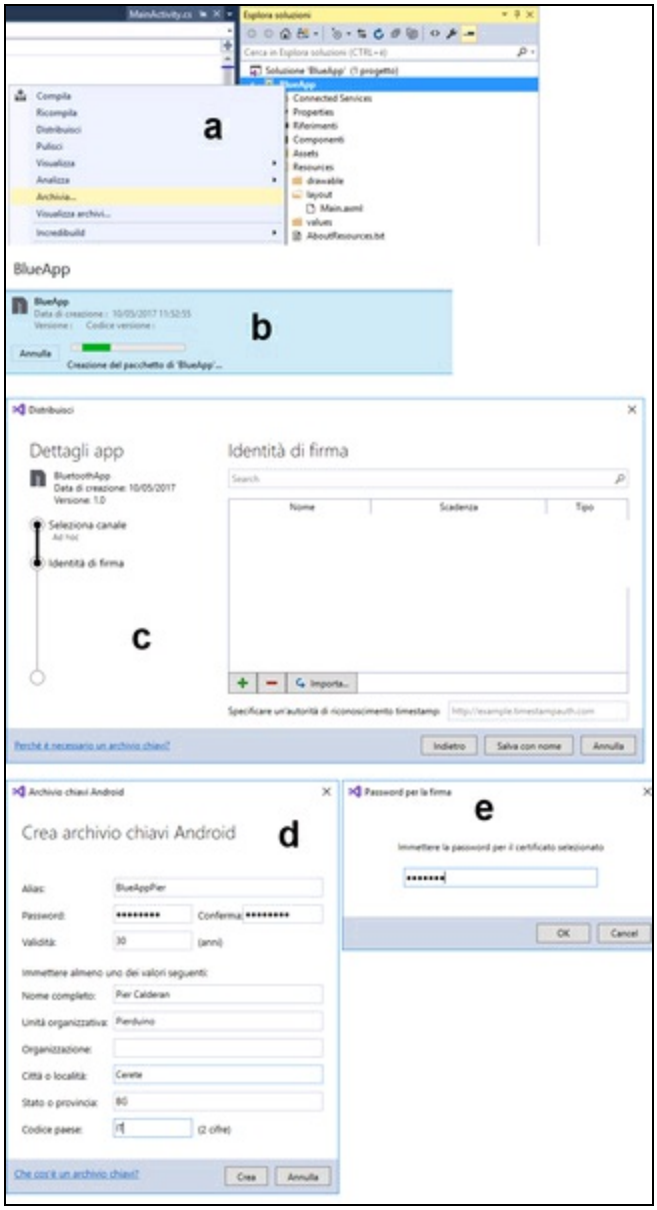


Figura 10.13 Fasi per la distribuzione dell'app.

Al termine del salvataggio si aprirà una finestra di dialogo che chiederà dove salvare il pacchetto firmato. Il nome di default del pacchetto sarà `BlueApp.BlueApp.apk`, che ovviamente si può cambiare come si vuole.

Installazione dell'applicazione

Per installare il pacchetto APK appena creato bisogna trasferirlo nello smartphone. Se lo smartphone permette l'accesso alla card, basta copiare il file APK nella card e quindi avviare l'installazione dalla card. Se lo smartphone non consente l'accesso diretto alla card si può usare una delle tante applicazioni di trasferimento remoto. Una fra le tante è Send Anywhere (`https://send-anywhere.com`).

Una volta installata l'app (Figura 10.14a) nello smartphone, apparirà l'icona BlueApp fra le applicazioni installate. Se si vuole cambiare l'icona di default, basta modificare il file `icon.png` contenuto nella cartella `Resources/drawable`.

Ricordarsi di associare il modem HC-05 come già spiegato in questo Capitolo e avviare l'applicazione. La finestra dell'app sarà identica a quella disegnata nella finestra di progettazione (Figura 10.14b).

Premendo il tasto CONNECT verrà tentata la connessione Bluetooth e, se ha successo, apparirà una schermata simile alla Figura 10.14c con la scritta *Connected!*.

A questo punto, basta premere il tasto *LED ON* per vedere accendersi il LED di Arduino e *LED OFF* per spegnerlo, ma prima è necessario caricare lo sketch di cui parliamo nel prossimo paragrafo.

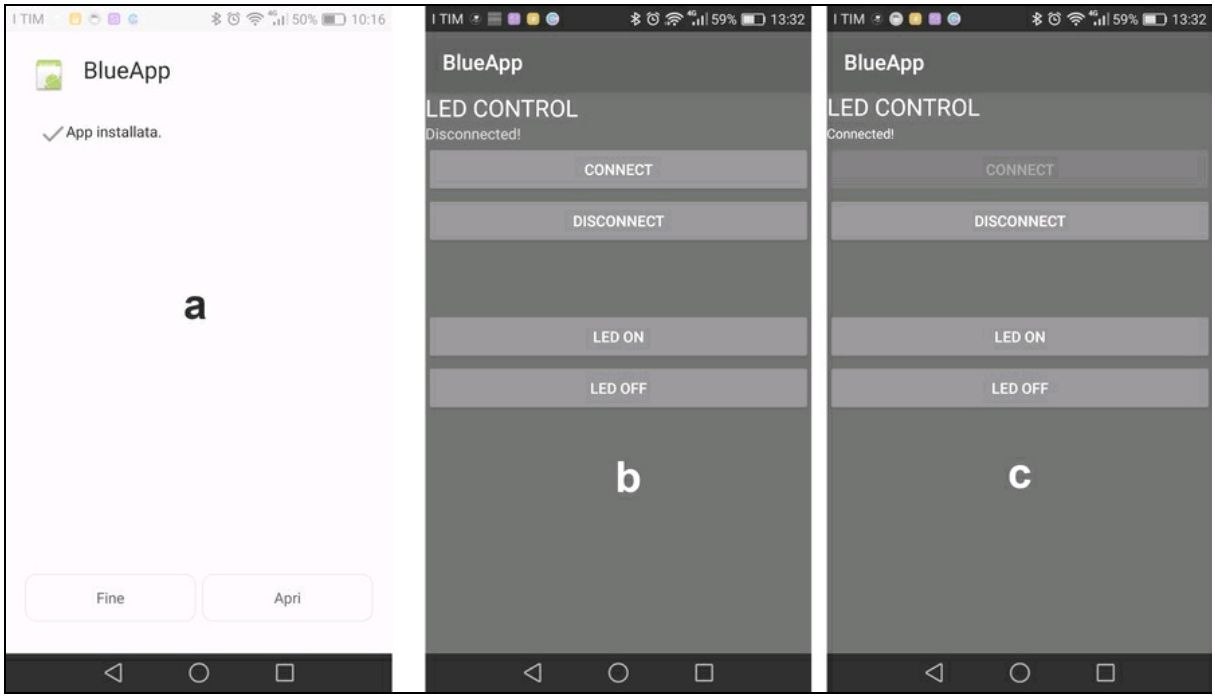


Figura 10.14 L'app installata (a). La finestra dell'app (b). Connessione avvenuta (c).

Lo sketch BlueApp per Arduino

Il circuito per il collegamento del modem HC-05 e del modulo relè ad Arduino è quello già visto all'inizio di questo Capitolo (Figura 10.2). Lo sketch da caricare si chiama `BLUE_APP_LED` ed è davvero molto semplice. Basta solo inizializzare la porta seriale e attendere il byte 1 o 0 inviato dall'app tramite la connessione Bluetooth.

Codice commentato

All'inizio si dichiarano le variabili per il dato ricevuto e il LED oppure il modulo relè collegato al pin digitale 13.

```
int dato = 0;
int led = 13;
```

Nel `setup()` viene inizializzata la porta seriale alla velocità standard del modem HC-05. Viene impostato il pin 13 come uscita.

```
void setup() {
    Serial.begin(38400);
    pinMode(led, OUTPUT);
}
```

Nel `loop()` viene letta la porta seriale. Se il dato ricevuto è `1`, accende il LED/relè e se è `0` lo spegne.

```
void loop() {
    while(Serial.available())
    {
        dato = Serial.read();
        if(dato == 1) digitalWrite(led,1);
        else if(dato == 0) digitalWrite(led,0);
    }
}
```

Fine! Modificando opportunamente l’applicazione BlueApp e lo sketch, si potranno controllare tutti i pin digitali di Arduino come si vuole. Buon lavoro!

Musica in casa

Descrizione

In mezzo a tanti assemblaggi elettronici, un po’ di relax non guasta. Questo progetto non prevede nessun tipo di programmazione né di circuitazione elettronica. Lo scopo è quello di controllare dal proprio smartphone un server musicale basato su Raspberry Pi.

La quantità di music player e di media center per Raspberry Pi è molto vasta. Dopo averne provati alcuni, abbiamo dovuto operare una scelta. Fra i software specializzati nello streaming Wi-Fi abbiamo individuato Pi MusicBox, sito ufficiale `http://www.pimusicbox.com` basato sul noto Mopidy, sito ufficiale `https://www.mopidy.com`.

Abbiamo preferito usare Pi MusicBox perché Mopidy si è rivelato troppo complesso da configurare e, anche se è molto “hackable”, non ci è sembrato utile ai nostri scopi.

La configurazione di Pi MusicBox è talmente semplice che in pochi minuti si è in grado di controllare tutta la musica stando comodamente sdraiati sul divano, utilizzando uno smartphone, un tablet, un computer portatile o scegliere il programma musicale di sottofondo dal computer di lavoro.

Inoltre, Pi MusicBox offre la possibilità di installare più lettori in più stanze della casa, per ascoltare la musica che si vuole dove si vuole.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Raspberry Pi 3 (o un qualsiasi altro modello);
- 1× impianto di diffusione (casse, cuffie, impianto audio e così via).

Pi MusicBox

Pi MusicBox è un server musicale che permette la riproduzione in streaming di sorgenti come Spotify, Google Music, SoundCloud, web radio, Podcast, così come file musicali da cartelle locali o NAS.

Questo lettore è *headless*, cioè senza interfaccia grafica e non ha bisogno di un monitor, quindi può essere avviato in modalità background. Il motore di Pi MusicBox è basato su Mopidy, per cui molte impostazioni sono simili, ma a differenza di questo, vanta eccezionali doti di immediatezza della configurazione e facilità d'uso.

Caratteristiche principali di Pi MusicBox.

- Musica in streaming da:
  - Spotify;
  - SoundCloud;
  - Google Music;
  - Podcast (con directory iTunes e gPodder);
  - file musicali locali e in rete;
  - formato dei file MP3/OGG/FLAC/AAC);
  - webradio con directory TuneIn, Dirble, AudioAddict, Soma FM e Subsonic.
- Controllo remoto da interfaccia web o da client MPD (come MPDroid per Android).
- Include anche opzioni per AirTunes/AirPlay e DLNA/OpenHome per lo streaming da smartphone, tablet (iOS e Android) o PC utilizzando software come BubbleUPnP.
- Supporto audio USB per tutti i tipi di schede audio USB, altoparlanti e cuffie.
- Supporto Wi-Fi (WPA per gli adattatori Wi-Fi supportati da Raspbian).

- Riproduce i file musicali dalla card SD, USB e Network.
- Scrobbling per Last.FM.
- Diverse schede sonore Pi supportate: HifiBerry, JustBoom, IQ Audio.
- Non è necessario effettuare difficili configurazioni e, soprattutto, non è necessario utilizzare nessuna riga di comando dal terminale Linux (come invece bisogna fare con Mopidy).

Requisiti

- Raspberry Pi (tutti i modelli).
- Altoparlanti, amplificatori o cuffie (analogici o USB).
- SD card, minimo 4 GB.
- Computer, tablet o smartphone con un browser di ultima generazione.
- L'interfaccia web è testata con versioni recenti di Firefox, Chrome, Internet Explorer e iOS (iPad/iPhone), versioni Android di Chrome Mobile, Firefox Mobile. Internet Explorer versione 10 funziona, le versioni precedenti no. È inoltre possibile utilizzare un client MPD (*Music Player Daemon*) per connettersi.

La Figura 11.1 illustra l'interfaccia web raggiungibile da qualsiasi dispositivo connesso in rete. Dal menu laterale si accede a tutte le altre pagine che compongono il sistema Pi MusicBox.

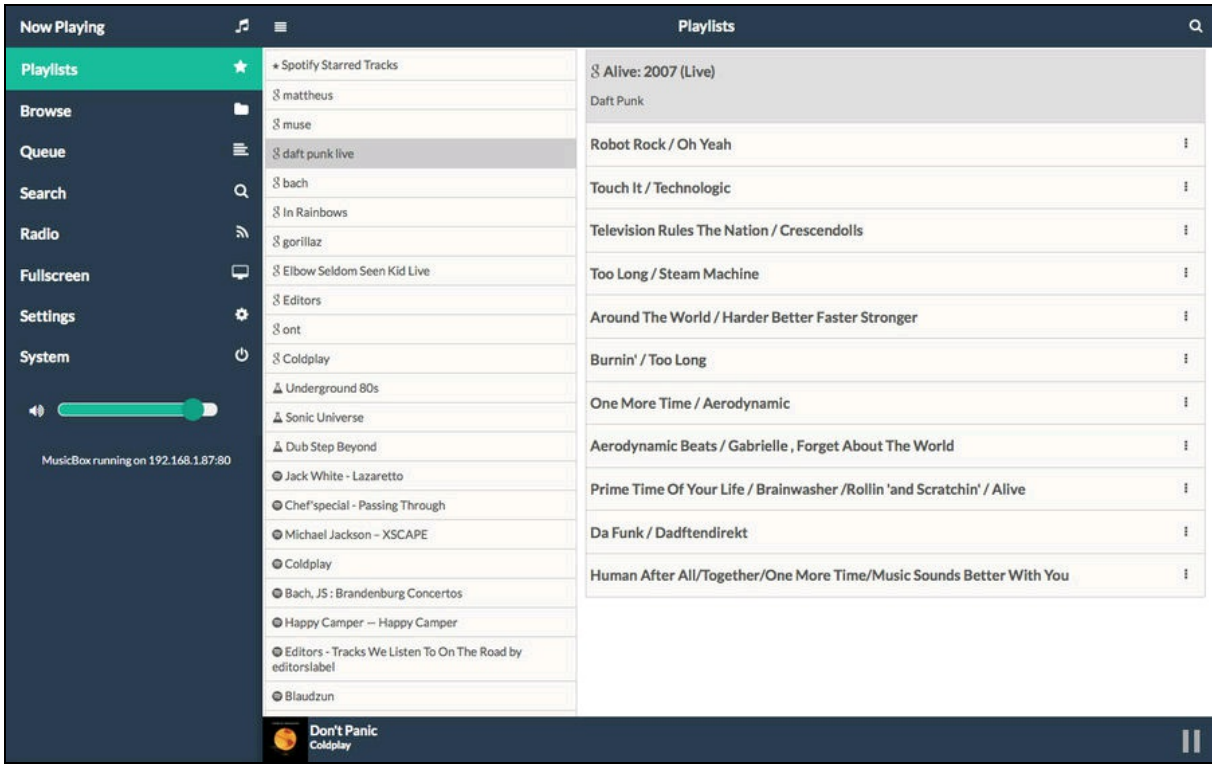


Figura 11.1 L'interfaccia web di Pi MusicBox.

Installazione

Il link per il download è nella home page del sito ufficiale: `http://www.pimusicbox.com`. Altrimenti ci si può recare direttamente al repository: `https://github.com/pimusicbox/pimusicbox/releases/tag/v0.7.0RC4`.

La versione attuale (maggio 2017 - v0.7.0 Release Candidate 4), oltre a risolvere vari bug, offre il supporto completo per Raspberry Pi 3 e Raspberry Pi Zero W, che hanno il modulo Wi-Fi incorporato. Gli altri modelli di Raspberry Pi devono montare un dongle Wi-Fi. Una volta scaricato il file compresso, bisogna scompattarlo in una cartella. All'interno si troverà l'immagine, il manuale di istruzioni in PDF e altri file. Quindi bisogna usare un programma per scrivere l'immagine su una card, per esempio *Win32DiskImager*. Si consiglia una card SD da 8 o 16 GB, in modo da non avere problemi di spazio per il salvataggio dei file

locali. Si raccomanda di usare una card di qualità e in classe 10, per aumentare la velocità di trasferimento dei file.

Win32DiskImager

Uno dei programmi più usati per scrivere immagini su card è *Win32DiskImager* (Figura 11.2a) scaricabile dal sito: `https://sourceforge.net/projects/win32diskimager/files/latest/download`. Ecco la procedura per scrivere un'immagine su card SD in quattro passaggi.

1. Inserire nel lettore di card micro-SD del PC o in un adattatore per card micro-SD la card su cui scrivere l'immagine di Pi MusicBox.
2. Dalla finestra principale di Win32DiskImager selezionare il file immagine (con estensione `.img`) dalla cartella appena scompattata.
3. Fare clic su *Scrivi* per scrivere l'immagine nella card.
4. Al termine della scrittura fare clic su *Esci*.

SD Formatter

Si consiglia di installare anche *SD Formatter* (Figura 11.2b) un software in grado di formattare tutti i tipi di card SD, SDHC e SDXC. Riesce a formattare anche card che sono state partizionate o formattate male. Fra le opzioni, c'è la possibilità di ridimensionare la formattazione. Il software è scaricabile presso il sito ufficiale: `https://www.sdcard.org/downloads/formatter_4`. Si consiglia di usare SD Formatter specialmente quando una card ha la FAT rovinata.

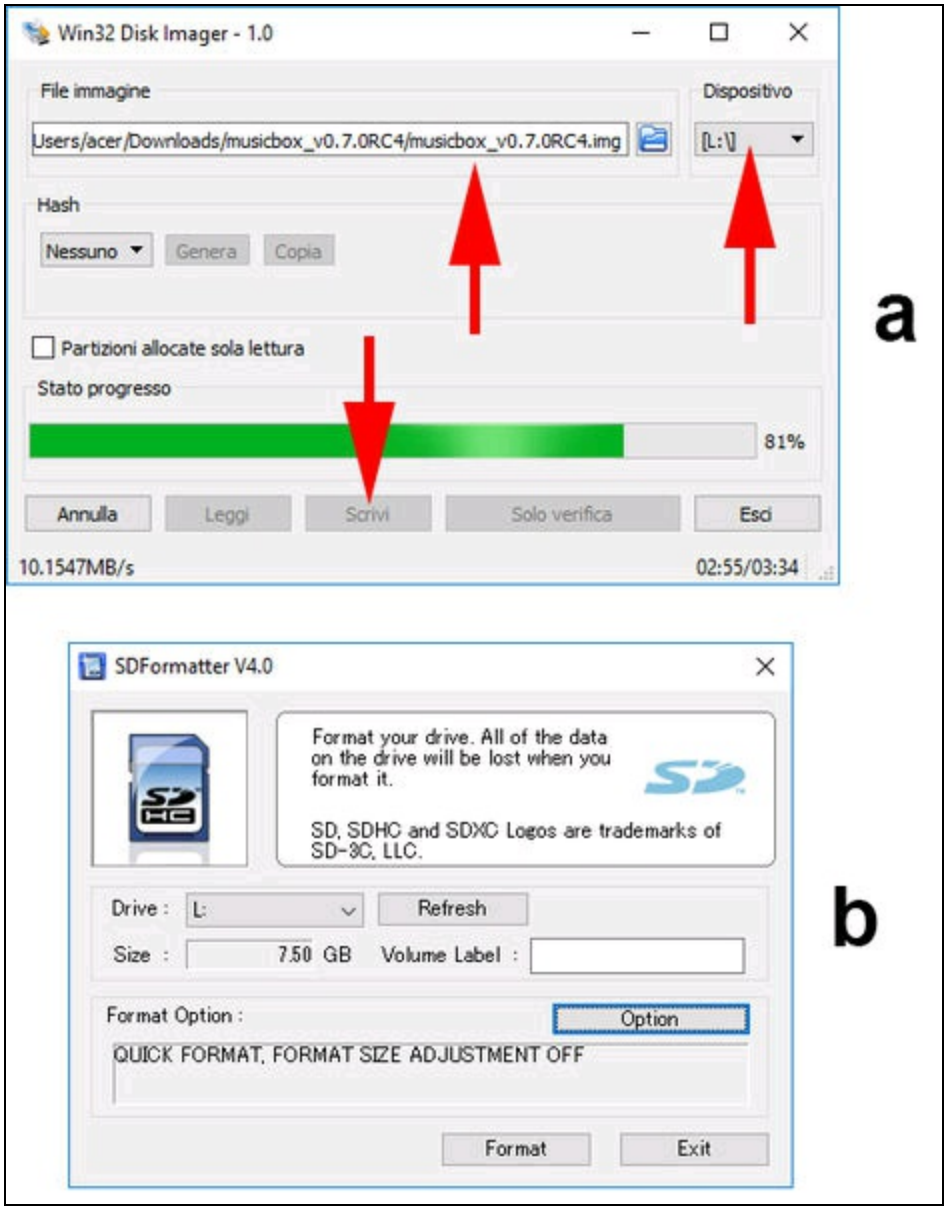


Figura 11.2 Win32DiskImager (a). SD Formatter (b).

Configurazione della rete Wi-Fi

L'unica operazione preliminare per far funzionare il server di Pi MusicBox è quella di inserire SSID e password della rete di casa nel file `settings.ini`. Questo file si trova all'interno della cartella `config` (Figura

11.3a). Aprire il file `settings.ini` (Figura 11.3b) con un editor di testi e individuare le seguenti righe:

```
wifi_network =
wifi_password =
```

Inserire il nome della rete (SSID) e la password (WPA, non WEP) fra apici semplici (Figura 11.3c):

```
wifi_network = 'nome_SSID'
wifi_password = 'password'
```

Qualche riga più sotto si trova anche il nome del server:

```
name = MusicBox
```

Si può lasciare il nome predefinito, oppure cambiarlo a piacere, nel caso si volessero installare più lettori in più stanze della casa. Per esempio si può digitare “cucina”, “salotto”, “studio” e così via.

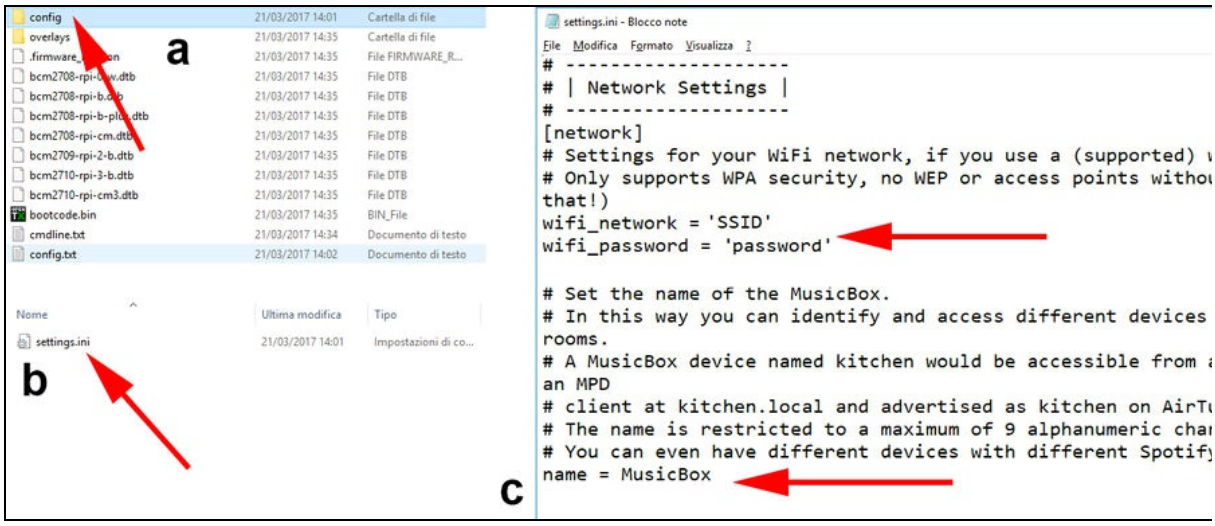


Figura 11.3 La cartella config (a). Il file settings.ini (b). SSID, password e name (c).

Fatto questo, salvare il file, inserire la card nello slot di Raspberry Pi (spento!) e accenderlo. Almeno per la prima volta è consigliabile collegare un monitor per seguire il boot del sistema e vedere che non ci siano errori fatali. Se tutto va bene, alla fine del boot si vedrà una schermata simile a quella rappresentata nella Figura 11.4.

Nella schermata apparirà il nome e l’IP del server cui collegarsi da qualsiasi browser, ovvero `http://MusicBox.local` oppure un indirizzo simile

a `192.168.1.xxx` dove *xxx* è un numero assegnato dal server DHCP. Se nel file `settings.ini`, è stato dato il nome “cucina”, si dovrà digitare `http://cucina.local`.

All’avvio, Pi MusicBox impiega un po’ di tempo per configurare il sistema, connettersi alla rete e fare la scansione dei file sulla card. In media, se non ci sono tanti file nella card, impiega circa un paio di minuti prima di essere pronto all’uso.

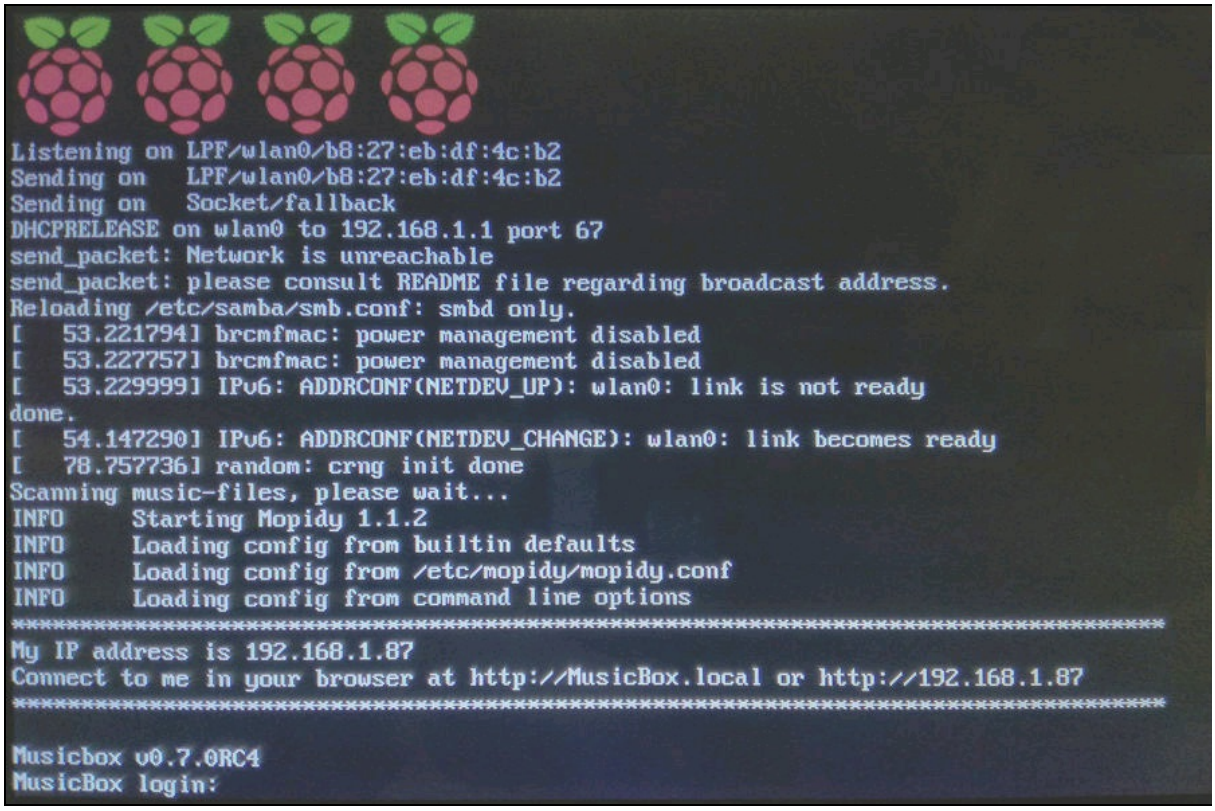


Figura 11.4 La schermata del server Pi MusicBox.

Funzionamento

L’uso di Pi MusicBox è molto intuitivo, per cui ci limiteremo a dare qualche indicazione essenziale. È comunque disponibile il manuale d’uso per i dettagli e fare un po’ di hacking. Per trovare soluzioni ai problemi esiste un forum online: `https://discuss.mopidy.com/c/pi-musicbox`.

Digitando il DNS, per esempio `http://musicbox.local` o `http://cucina.local` o l’indirizzo IP, in qualsiasi browser connesso alla stessa rete, apparirà la home page, come illustrato nella Figura 11.5. Da qui si può navigare nelle varie pagine, che sono nell’ordine:

- *Now Playing;*
- *Queue*;
- *Playlists*;
- *Browse*;
- *Streams*;
- *Search*;
- *Fullscreen*;
- *Settings*;
- *System*.

Un comodo slider nella barra laterale serve per regolare il volume audio. Facendo clic sull’icona si può silenziare o meno il suono.

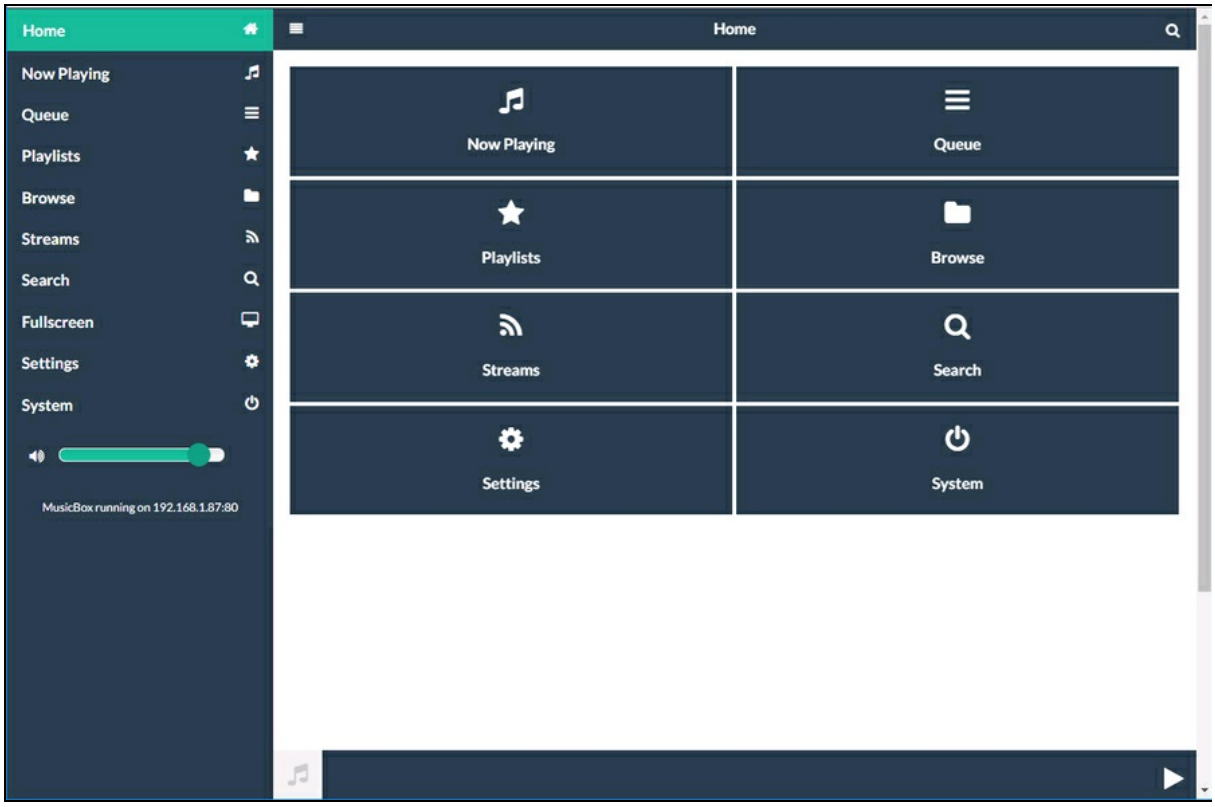


Figura 11.5 La home page di Pi MusicBox.

Streams

Nella pagina *Streams* (Figura 11.6) è possibile selezionare qualche radio preimpostata. Trovando gli URI delle migliaia di radio in streaming su Internet, è possibile ascoltare qualsiasi radio e aggiungerla alla lista.

Ecco qualche esempio.

- *Radio Swiss Jazz*: `http://www.radioswissjazz.ch/live/mp3.m3u`.
- *Radio Deejay*: `http://radiodeejay-lh.akamaihd.net/i/RadioDeejay_Live_1@189857/master.m3u8`.
- *Radio Capital*: `http://radiocapital-lh.akamaihd.net/i/RadioCapital_Live_1@196312/master.m3u8`.

- *Radio M2O*: `http://radiom2o-lh.akamaihd.net/i/RadioM2o_Live_1@42518/master.m3u8`.

Ecco come aggiungere la radio preferita.

1. Nel campo sotto *Get currently playing* incollare l'URI della radio.
2. Con il tasto *Play* si manda in riproduzione la radio (attendere qualche secondo per il buffering).
3. Nel campo *Name* digitare il nome della radio, per esempio "Radio Swiss Jazz".
4. Fare clic sul pulsante *Save* per salvare la radio che verrà aggiunta all'elenco.
5. Selezionare la radio per mandarla in esecuzione.

È da notare che l'icona a X presente in ogni campo lo svuota. Nell'elenco delle radio, elimina la radio dall'elenco.

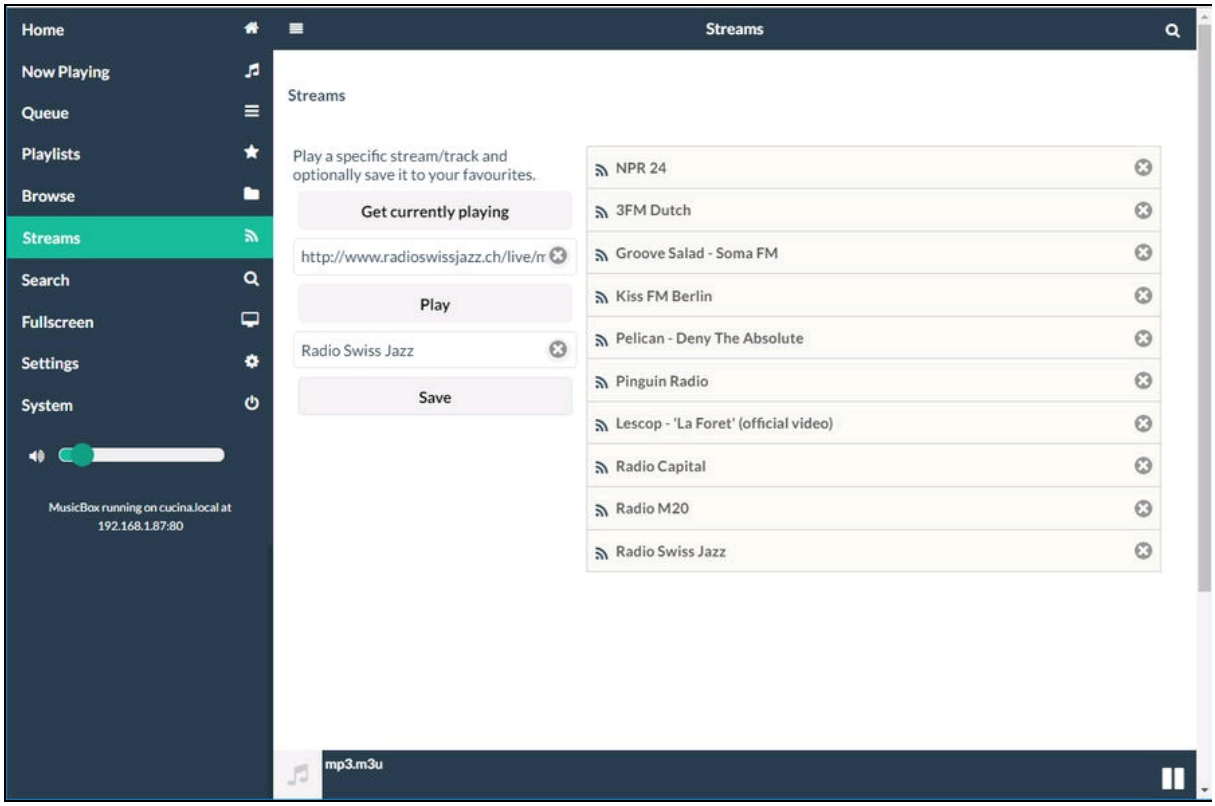


Figura 11.6 La pagina Streams.

Search

La pagina *Search* consente di effettuare comode ricerche in Rete di un titolo, di un album o di un artista. La Figura 11.7 illustra un esempio di ricerca dell'artista Thelonius Monk. Dopo qualche secondo di attesa compare la lista di tutto il materiale sonoro che riguarda quell'artista. L'icona prima di ogni traccia trovata indica la provenienza, per esempio YouTube, web radio oppure un link da un sito o altro ancora. Con il tasto *All Services* si può limitare la ricerca solo ad alcuni server.

Selezionando una traccia fra i risultati trovati, la si manda in riproduzione.

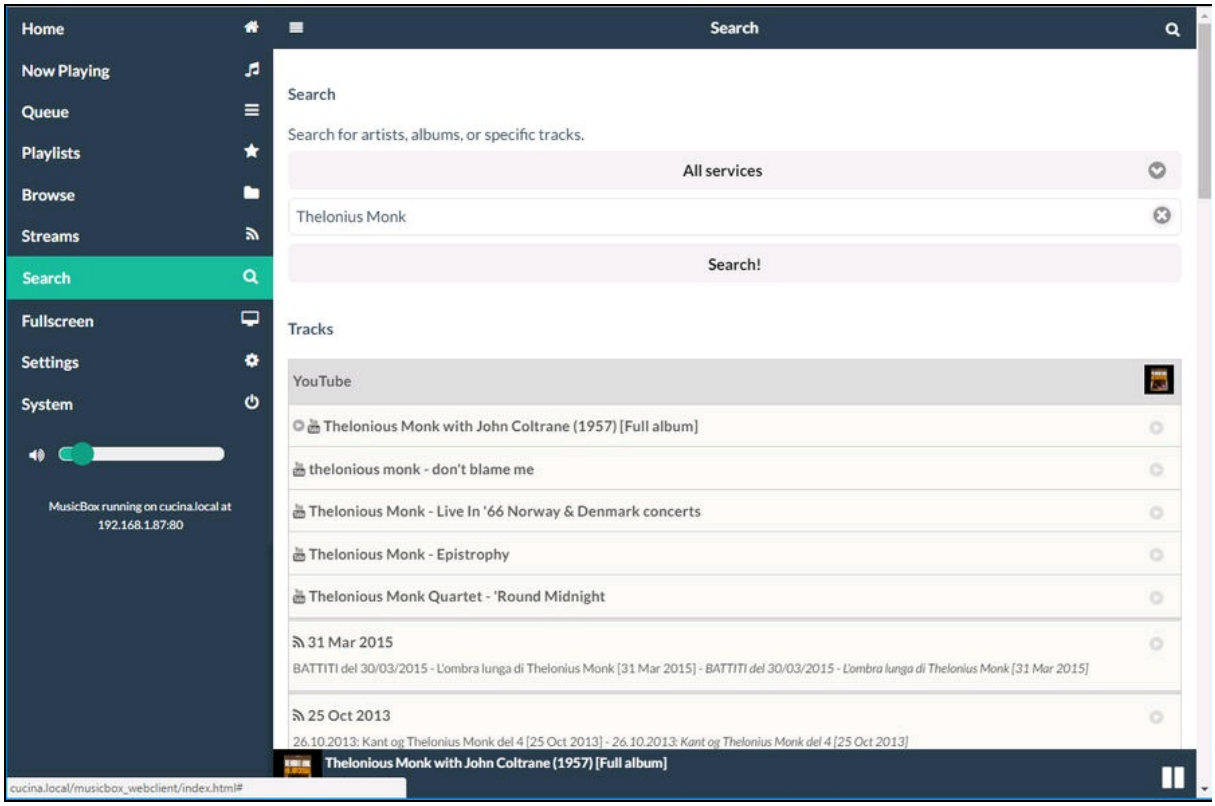


Figura 11.7 La pagina Search.

Playlists

La pagina *Playlists* visualizza l'elenco delle radio o di altre sorgenti. Ogni traccia ha un piccolo tasto *Play* che apre un menu a tendina in cui scegliere un'azione sulla traccia selezionata (Figura 11.8).

- *Play (nome della traccia)*: riproduce il brano corrente.
- *Play All*: riproduce tutti i brani.
- *Play Track Next*: riproduce la traccia mettendola in coda (Queue).
- *Add Track to Bottom of Queue*: aggiunge la traccia alla fine della coda.
- *Add All to Bottom of Queue*: aggiunge tutte le tracce alla fine della coda.
- *Show Track Info*: mostra le informazioni della traccia, ovvero URI e, se disponibili, nome e artista.

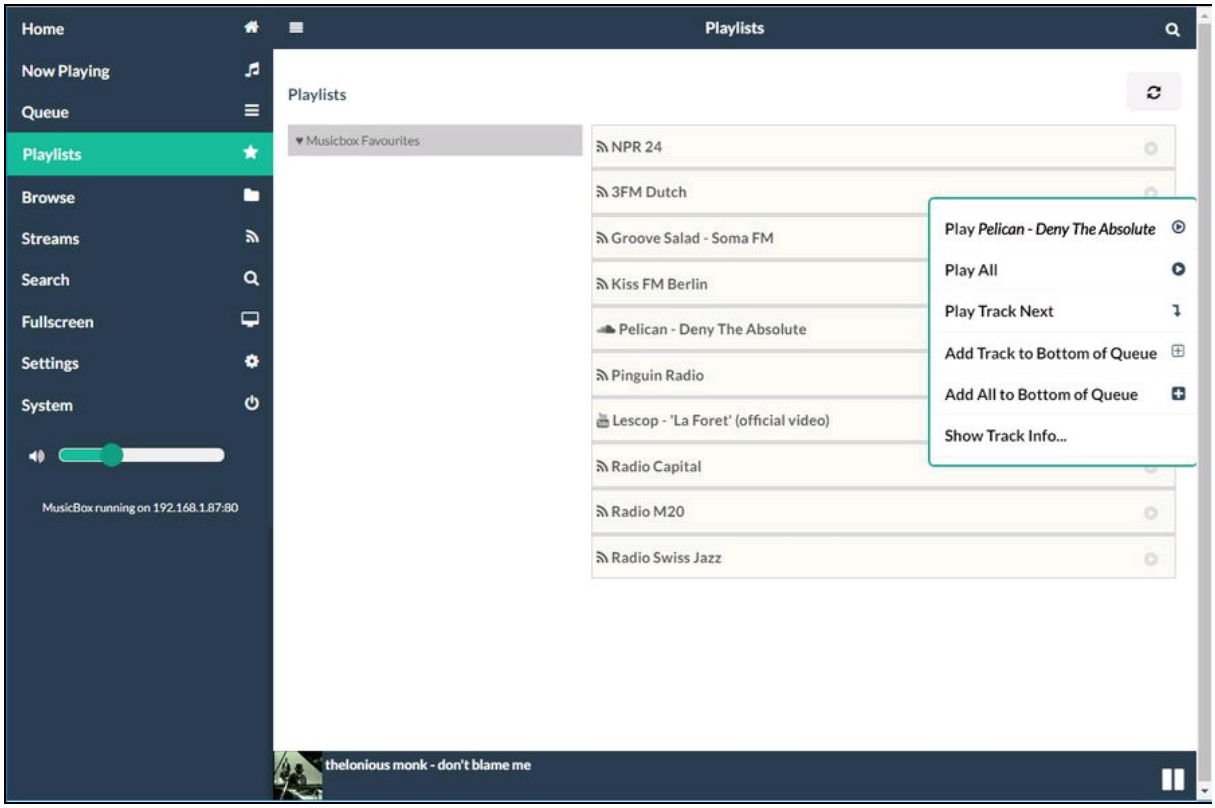


Figura 11.8 La pagina Playlists.

Now Playing

La pagina *Now Playing* (Figura 11.9) visualizza semplicemente la traccia che è in riproduzione al momento. Se sono disponibili le informazioni, vengono visualizzati il titolo dell'album, il nome della traccia, il nome dell'artista e anche la copertina del disco.

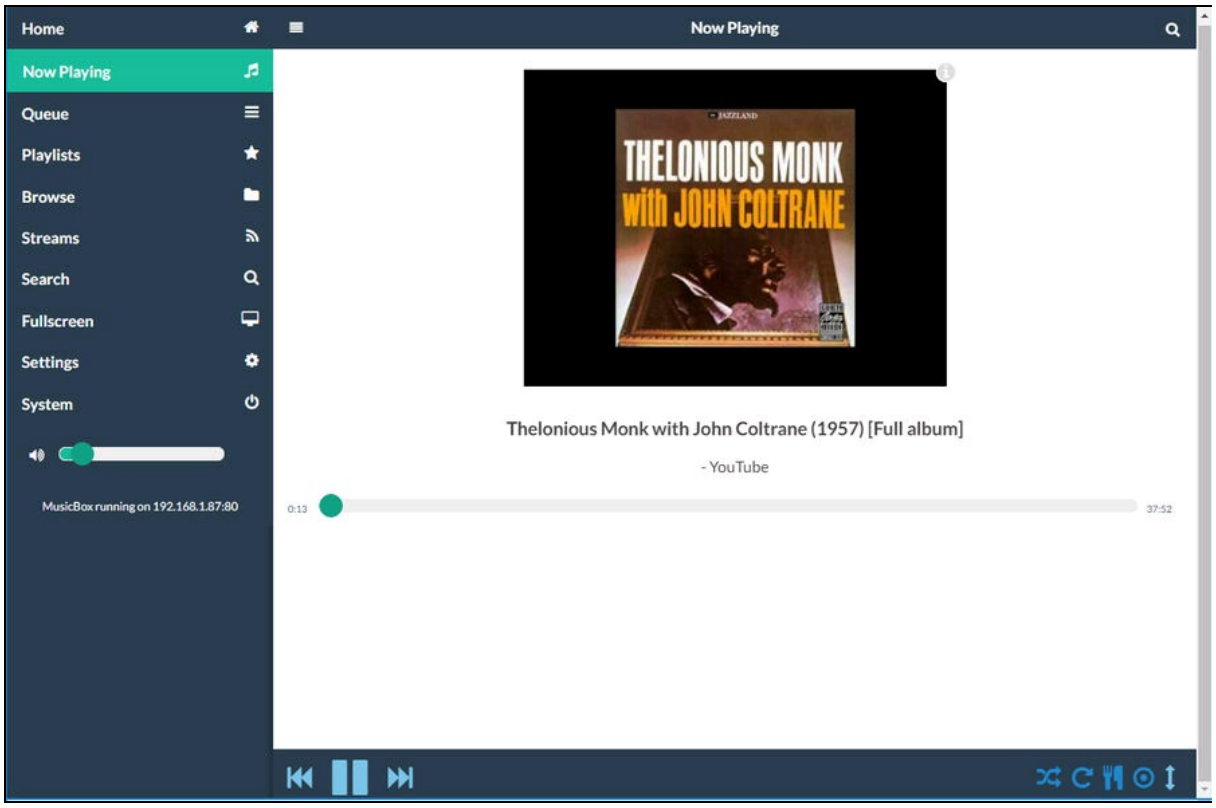


Figura 11.9 La pagina Now Playing.

Trasferimento di file alla card

Oltre allo streaming da Internet, la parte più divertente consiste nel creare delle cartelle con i file preferiti.

Per il trasferimento di file alla card è sufficiente aprire le risorse di rete e individuare il nome del dispositivo, per esempio, "CUCINA.LOCAL", come indicato nella Figura 11.10a. Quindi, basta

aprire la cartella *Music* (Figura 11.10b) e copiarvi i file preferiti (Figura 11.10c). Si possono anche creare cartelle e sottocartelle per organizzare il contenuto per artista o genere.

La pagina *Browse* (Figura 11.10d) consente di visualizzare il contenuto della cartella *Files* > *MusicBox*, contenente i file trasferiti, comprese le eventuali cartelle e sottocartelle, create per genere o artista. Se si copiano i file nella card mentre il server è in esecuzione, bisogna fare clic sull'icona con due frecce in circolo (*refresh*) per rileggere il contenuto cambiato.

In breve, dalla pagina *Browse* si può navigare nei seguenti percorsi.

- *Dirble*: servizio di streaming con migliaia di stazioni radio per tutti i gusti. Le stazioni radio internet possono essere trovate in base alla categoria, il paese, il titolo, il genere e così via.
- *Files*: file contenuti nella card SD.
- *Local media*: file organizzati in cartelle (genere, titolo, artista e così via).
- *Podcasts*: link a siti di Podcasting (gpodder.net, iTunes Store e così via).
- *Spotify Browse*: link alle liste di Spotify.
- *TubeIn*: servizio di streaming con oltre 100 000 stazioni radio e contenuti on-demand come Podcast e spettacoli.

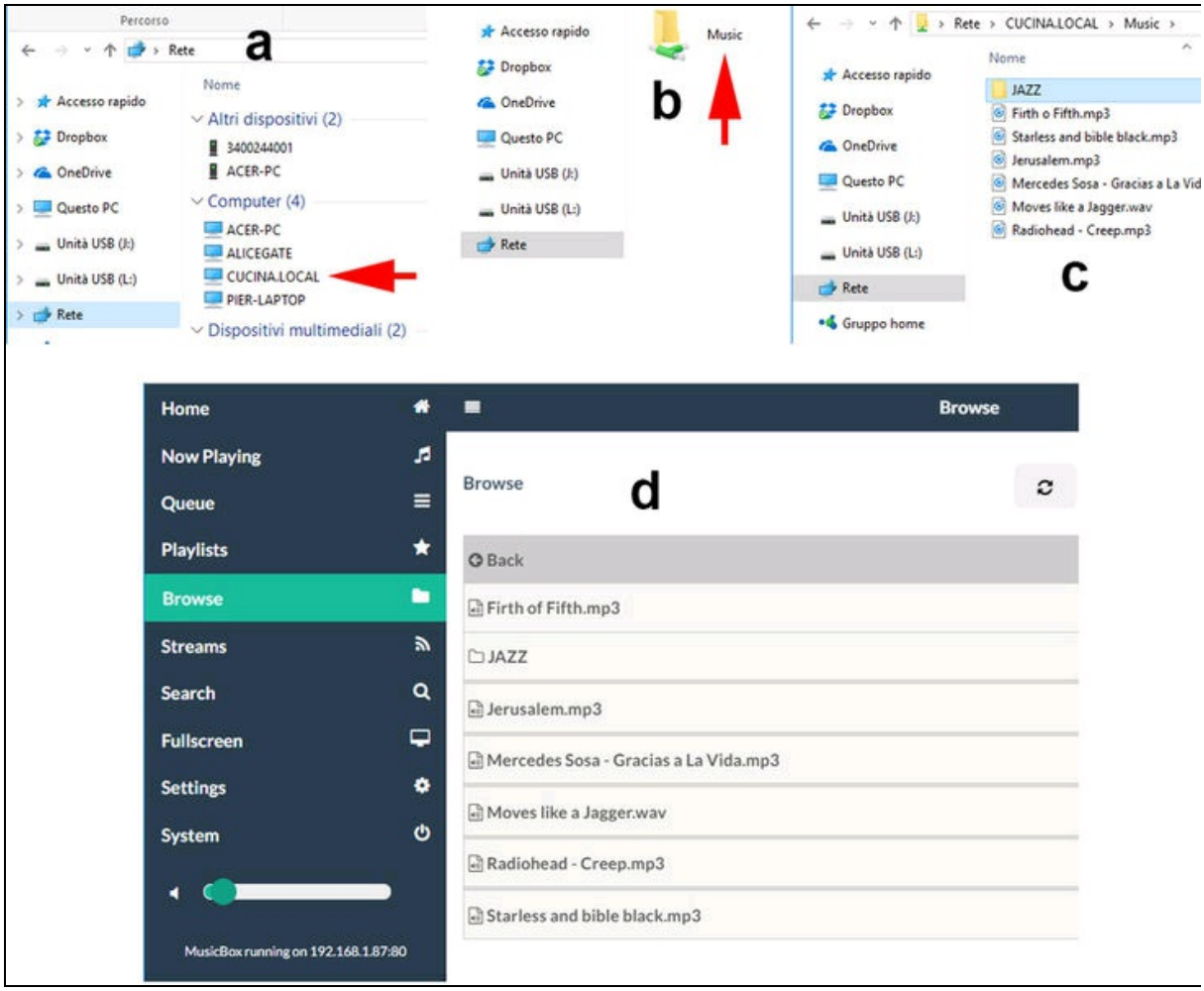


Figura 11.10 Dispositivo in rete (a). La cartella Music (b). Trasferimento di file alla card (c). La pagina Browse (d).

Settings

La pagina *Settings* (Figura 11.11) consente di impostare o reimpostare varie preferenze. Per esempio, nella sezione *Settings* sono disponibili le seguenti opzioni.

- *Network*: permette di impostare SSID e password della rete Wi-Fi, il dominio di gruppo (per esempio WORKGROUP) e la sicurezza SSH.

- *MusicBox*: permette di impostare la password di accesso come amministratore (`root`), il nome del dispositivo (per esempio `cucina`, `salotto` e così via), *AutoPlay* per la riproduzione automatica e il timer per la riproduzione automatica, *AirPlay Streaming* attiva lo streaming audio in Music Box da iPhone/iPad/Mac/iPod utilizzando ShairPort-Sync e l'audio streaming in Music Box da dispositivi DLNA/UPnP/OpenHome.

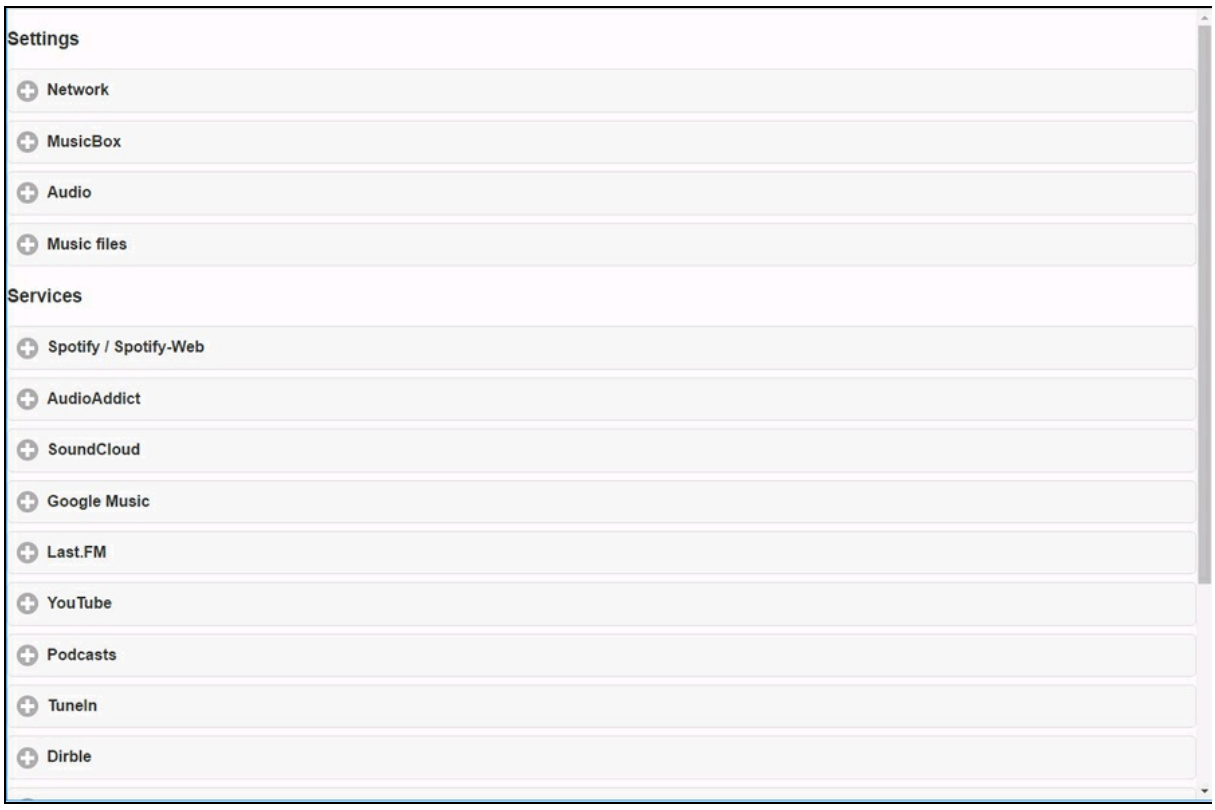


Figura 11.11 La pagina Settings.

- *Audio*: permette di rilevare in modo automatico l'audio USB se trova un dispositivo audio USB, l'audio HDMI se viene rilevato un dispositivo HDMI, altrimenti usa l'uscita analogica. Attenzione! Le schede JustBoom, HifiBerry, IQ Audio e così via non possono essere rilevate automaticamente e devono essere impostate esplicitamente. A causa delle limitazioni con alcuni USB DAC, per

impostazione predefinita MusicBox converte il suono USB a 44 KHz. È possibile disattivarlo qui. Il Mixer hardware, per alcune schede audio USB (DAC), è attivabile da qui.

- *Music Files*: utilizzare questa impostazione per consentire a MusicBox di eseguire la scansione all'avvio per i nuovi file musicali sulla scheda SD, USB o in Rete. Questo potrebbe rallentare il boot. L'opzione Network Drive serve a montare l'unità di rete di Windows durante l'avvio. Digitare l'indirizzo esattamente in questo modo: `(IP supporto samba)://nomeserver/mountpoint/directory`. Per esempio: `//192.168.1.5/musicmount` oppure `//server.local/shared/music`. Se il supporto necessita di un *nome utente*/*password*, bisogna impostarlo nei campi sottostanti. Lasciarli vuoti per l'accesso ospite (`guest`).

Nella sezione *Service* si possono configurare i servizi seguenti:

- Spotify;
- AudioAddict;
- SoundCloud;
- Google Music;
- Last.FM;
- YouTube;
- Podcasts;
- TuneIn;
- Dirble;
- Soma FM;
- Local Files;
- The Internet Archive;
- Subsonic.

Per ognuno dei suddetti servizi è disponibile una configurazione dedicata, pertanto l'accesso ai servizi gratuiti o a pagamento possono

essere attivati con le credenziali dell’account gratuito o a pagamento. Per i servizi gratuiti, basta solo attivare la ricezione.

Schede audio dedicate

Per una migliore riproduzione audio esistono in commercio varie schede appositamente progettate per Raspberry Pi. Una fra quelle compatibili con Pi MusicBox è *HiFiBerry Amp+*. Sito ufficiale: https://www.hifiberry.com.

HiFiBerry Amp+

HiFiBerry Amp+ è un amplificatore di potenza in Classe D di alta qualità e altamente efficace con Raspberry Pi.

Caratteristiche principali:

- potenza di uscita fino a 25W;
- capacità di pilotare altoparlanti da 4 o 8 Ω (funziona anche con altoparlanti ad alta impedenza);
- completamente controllabile da Raspberry Pi;
- campionamento a 44.1 e 48 kHz;
- conversione digitale-analogica inclusa, senza bisogno di DAC esterni o schede audio;
- percorso audio completamente digitale per prestazioni audio ottimali;
- si collega direttamente a Raspberry Pi senza ulteriori cavi;
- alimentazione esterna da 12 a 18 V;
- viene fornito già montato, senza bisogno di saldature.

HiFiBerry Amp+ va montato sulla parte superiore di Raspberry Pi (porta GPIO). Attenzione! Raspberry Pi verrà alimentato dall'amplificatore, per cui non bisogna assolutamente collegare i 5 V alla porta micro-USB di Raspberry Pi. Lo schema di collegamento è riportato nella Figura 11.12.

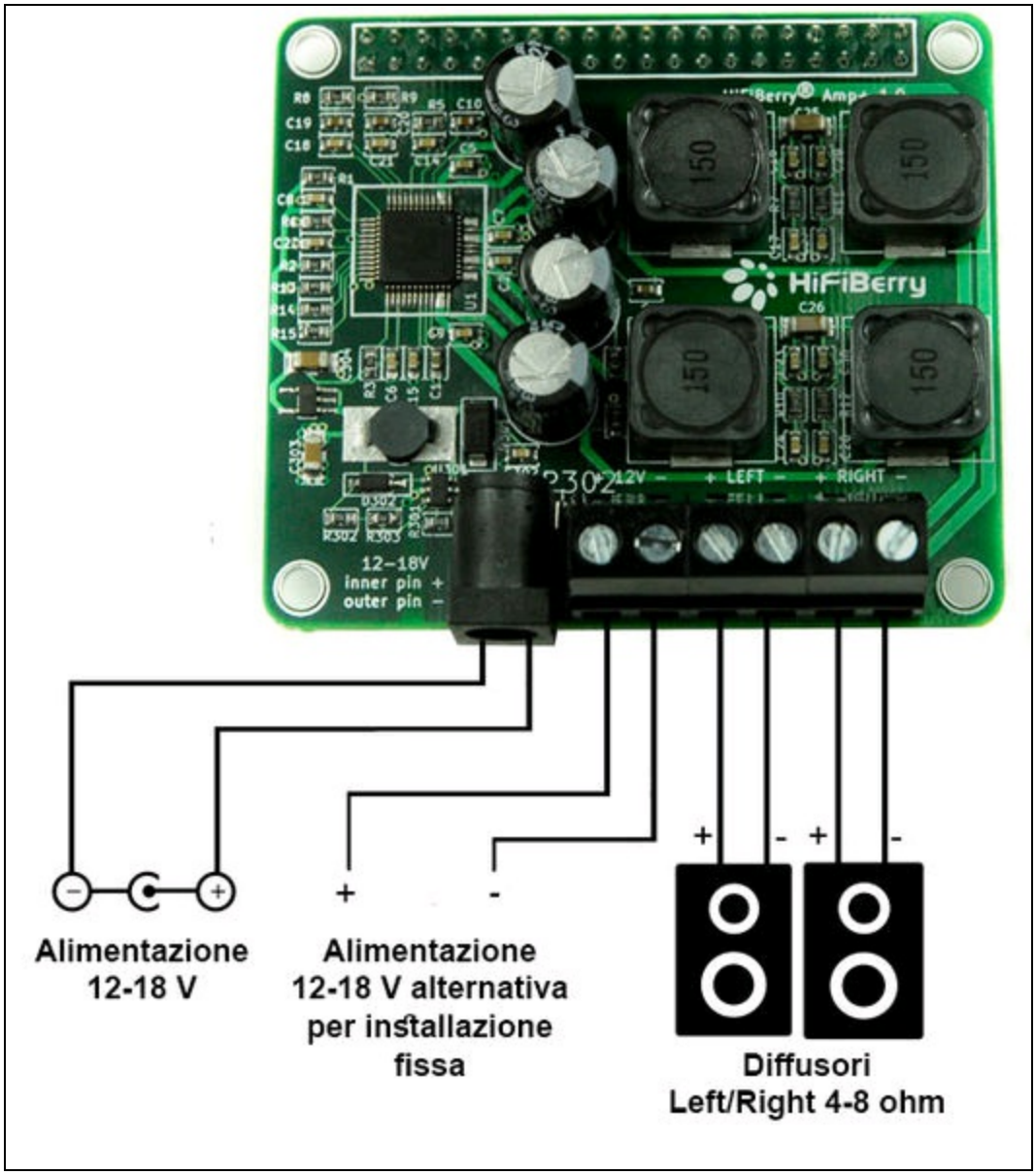


Figura 11.12 Schema di collegamento HiFiBerry Amp+.

Videosorveglianza

Descrizione

L’idea di questo progetto è quella di monitorare un locale e di rilevare il movimento di un intruso. Il sistema attiva un allarme e, opzionalmente, manda un’email con la foto dell’intruso oppure, in alternativa, segnala solo la presenza di un oggetto che si muove.

Il progetto si basa su Raspberry Pi 3, perché è già dotato di Wi-Fi per la connessione Internet al server di posta, ma si può usare anche il minuscolo Raspberry Pi Zero W con il comodo supporto ZeroView per la videocamera. Se invece si vuol usare la connessione Ethernet, va bene un qualsiasi altro modello di Raspberry Pi.

Il circuito elettronico è semplicissimo e prevede un LED che si accende al rilevamento del movimento e si spegne quando non rileva nessun movimento. Come al solito, si potrà sostituire il LED con un modulo relè, come già visto in altri progetti, e collegare un sistema acustico di allarme (sirena o simile) oppure un qualsiasi altro dispositivo, per esempio, un impianto di illuminazione o altro.

CODICE DI ESEMPIO

Tutti i file del progetto “Videosorveglianza” sono disponibili nelle risorse del libro presso il sito dell’autore all’indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× scheda Raspberry Pi 3/Zero W o altro modello;
- 1× Raspberry Pi Camera Module v2;
- 1× LED/modulo relè.

Il circuito elettrico

La Figura 12.1 illustra il semplice circuito di un LED in parallelo a un modulo relè collegati a una scheda Raspberry Pi. Per attivare il LED/relè è stato scelto il pin GPIO26, ma ovviamente si può usare un qualsiasi altro pin digitale, basta ricordarsi di modificarlo anche nel codice.

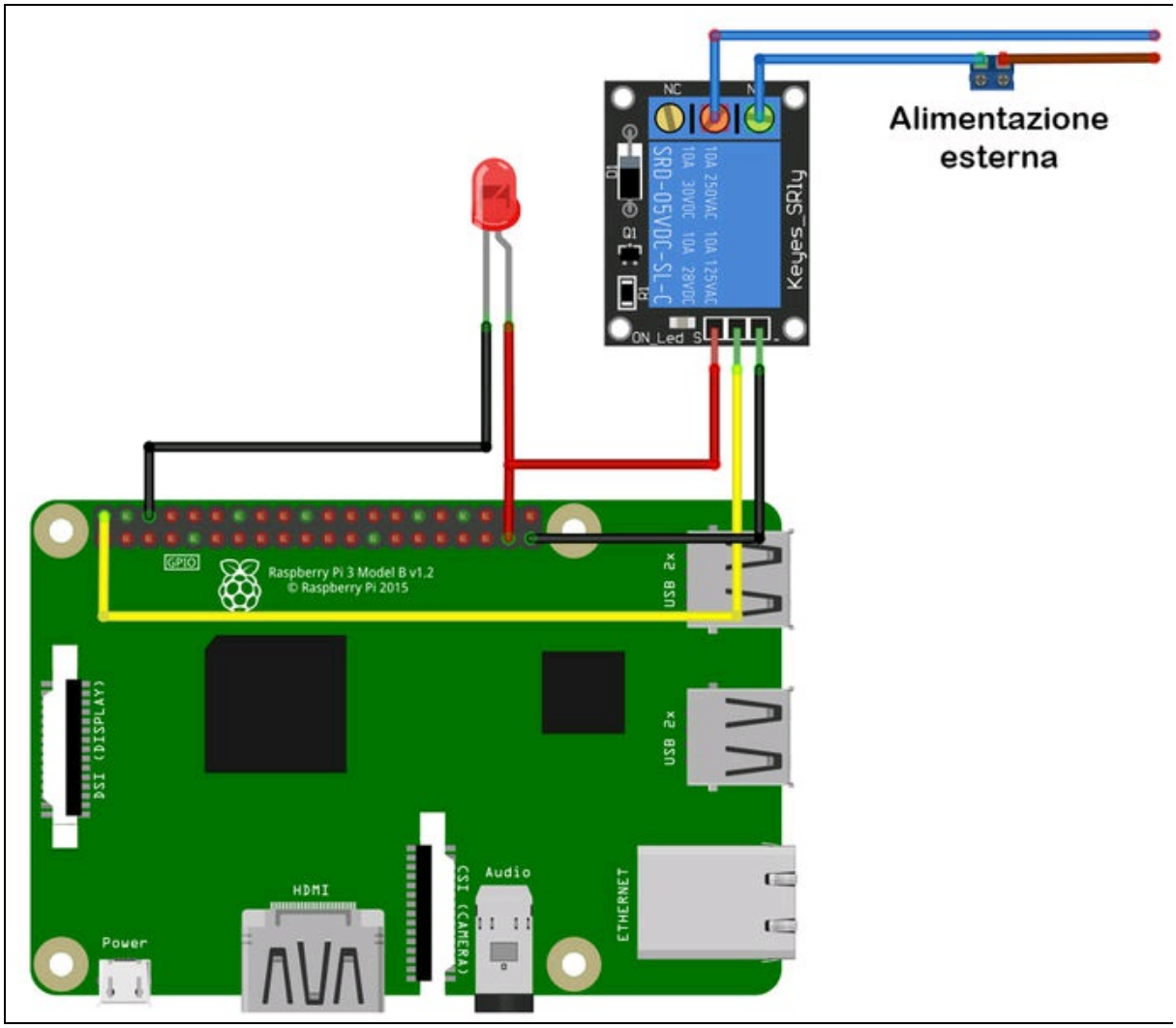


Figura 12.1 Schema di collegamento.

Il codice

Per questo progetto abbiamo optato per la piattaforma Linux, perché è l'ambiente ideale per trovare un sacco di open source e soprattutto perché si lavorerà in Python e altre librerie già presenti in RaspBian.

Una libreria open source adatta ai nostri scopi è OpenCV, abbreviazione di *Open Source Computer Vision*. Si tratta di una libreria multipiattaforma per la computer vision in tempo reale. Inizialmente sviluppata da Intel, viene mantenuta ora da Itseez di Intel. Con la visione computerizzata si possono creare software molto sofisticati per il riconoscimento facciale, il ritrovamento di immagini con i motori di ricerca, l'identificazione di targhe automobilistiche e così via.

Il linguaggio ideale per la programmazione in ambiente OpenCV è il C++, ma, grazie al porting della libreria per altri linguaggi (fra cui Python) è possibile usarla tranquillamente con RaspBian.

Il sito ufficiale di OpenCV è `http://opencv.org`. Qui si possono trovare tutte le informazioni per l'installazione e l'uso su tutte le piattaforme. La pagina di riferimento per Linux e Raspberry Pi è

`http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.ht`

Installazione di OpenCV

Le operazioni per compilare e installare OpenCV su RaspBian nella suddetta pagina del sito ufficiale sono chiare e concise. Prima di tutto, bisogna digitare i seguenti comandi per installare alcune librerie e software necessari. Se alcune di queste installazioni sono già presenti, verranno automaticamente saltate o aggiornate.

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev
libavformat-dev libswscale-dev
```

Dopodiché, bisogna clonare il pacchetto dal repository.

Dal terminale digitare il seguente comando:

```
git clone https://github.com/opencv/opencv.git
```

Verrà scaricata nella cartella `opencv` il pacchetto completo di OpenCV nella directory `home/pi`. Entrare nella cartella `opencv` con il seguente comando:

```
cd opencv
```

Creare una cartella `release`:

```
mkdir release
```

Entrare in tale cartella:

```
cd release
```

Compilare OpenCV con il seguente comando:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

Attenzione! La compilazione può impiegare anche un’ora o due. Al termine, digitare i seguenti comandi per l’installazione:

```
make
sudo make install
```

Bisognerà installare anche la libreria OpenCV per Python.

Digitare il seguente comando:

```
sudo pip install python-opencv
```

E per finire, installare anche la libreria `imutils` che serve per il nostro script di videosorveglianza. Questo modulo per Python è studiato appositamente per OpenCV e aggiunge una serie di funzionalità per eseguire l’elaborazione delle immagini come la rotazione, il ridimensionamento, la visualizzazione di immagini di Matplotlib, l’ordinamento dei contorni e l’individuazione dei bordi.

Per installare la libreria, recarsi nel sito https://pypi.python.org/pypi/imutils/0.4.3 e scaricare la versione 0.4.3 (maggio 2017) o successive facendo clic sul pulsante *Download*. Una volta scaricato, scompattare il file `tar.gz` in una cartella ed entrare nella cartella con il seguente comando:

```
cd imutils-0.4.3
```

Installare la libreria con il seguente comando:

```
sudo python setup.py install
```

Ora è tutto pronto per scrivere lo script in Python. Il nostro file si chiama `videsorveglianza.py` e si trova nelle risorse del libro.

Lo script prevede che sia installata Raspberry Pi Camera, una videocamera ad alta risoluzione appositamente progettata per Raspberry Pi. L'uso di una normale webcam USB non darebbe gli stessi risultati e sarebbe più difficile da gestire con OpenCV.

Si consiglia di acquistare il modulo Raspberry Pi Camera Module v2 presso il sito ufficiale https://www.raspberrypi.org/products/camera-module-v2 (Figura 12.2).

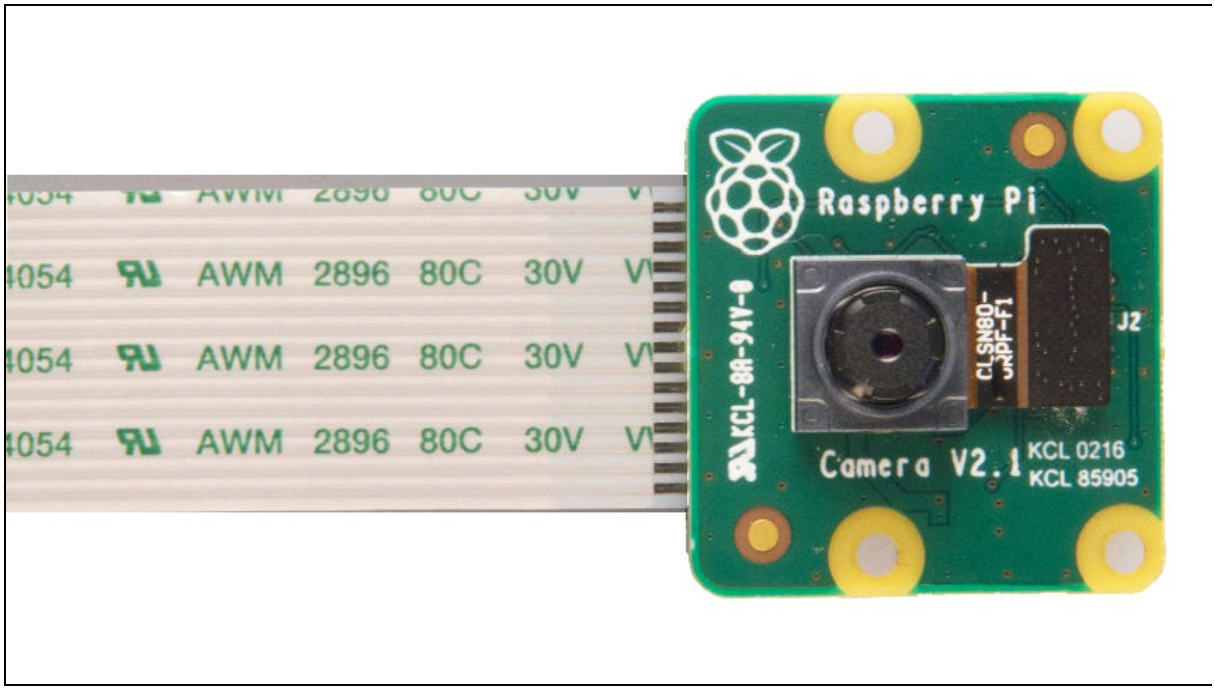


Figura 12.2 Raspberry Pi Camera Module v2.

Raspberry Pi Camera Module

Raspberry Pi Camera Module v2 ha sostituito il modulo originale v1 nell'aprile 2016. La versione 2 ha un sensore Sony IMX219 da 8 megapixel (rispetto al sensore OmniVision OV5647 da 5 megapixel della videocamera originale).

Il modulo può essere utilizzato per catturare video e scattare fotografie singole ad alta definizione. Ci sono molti esempi online per l'uso della videocamera come video time-lapse, slow-motion e altre smart-app per creare video creativi. Supporta le modalità video 1080p30, 720p60 e VGA90. Il modulo viene collegato tramite un cavo a nastro da 15 cm alla porta CSI di qualsiasi modello di Raspberry Pi.

Installazione e attivazione della videocamera

Prima di tutto, con Raspberry Pi spento, si dovrà collegare il modulo alla porta CSI di Raspberry Pi.

Fare molta attenzione all'orientamento del connettore a nastro. Una parte è colorata in blu o bianco e l'altra ha i contatti. Facendo riferimento alla Figura 12.3a, prima di inserire il connettore nella porta CSI, alzare leggermente il fermo in plastica, come indicato dalla freccia della Figura 12.3b. Dopo aver inserito il connettore, abbassare il fermo per bloccarlo. Tirare leggermente il cavo per assicurarsi che sia ben inserito nel connettore e accendere la scheda.

Dopo l'avvio del sistema, dalla voce *Preferences* del menu "lampone", aprire la configurazione di Raspberry Pi (Figura 12.3c).

Nella finestra *Preferences* (Figura 12.3d) selezionare la scheda *Interfaces*, attivare l'opzione *Camera*, fare clic su OK e riavviare il sistema.

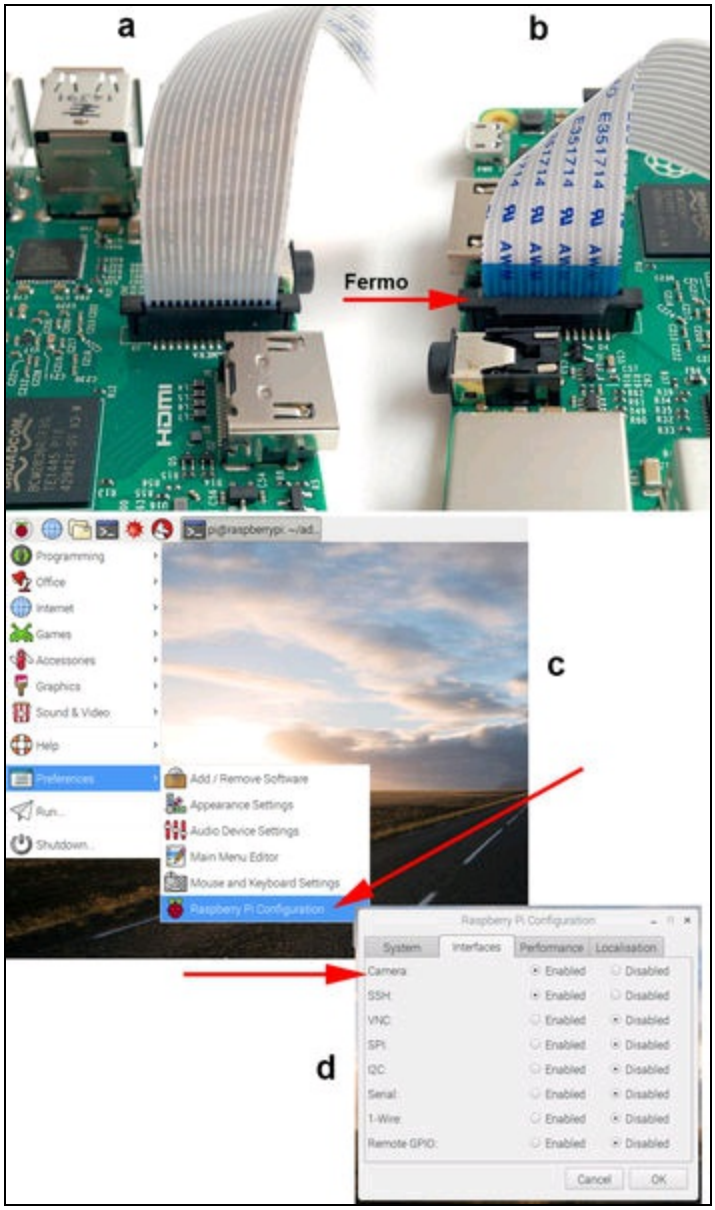


Figura 12.3 Inserimento del connettore alla porta CSI di Raspberry Pi (a). Il fermo del sconnettore (b). Il menu Preferences (c). La finestra Preferences (d).

Test della videocamera

Per vedere se la videocamera funziona correttamente con Python, si consiglia di effettuare questo semplice test, prima di provare quello di videosorveglianza.

Dopo l’avvio del sistema, dal menu “lampone”, aprire Python 2 o 3 (Figura 12.4a).

Dal menu *File* di Python creare un nuovo file e salvarlo con un nome qualsiasi. Si raccomanda assolutamente di NON usare il nome `picamera.py`.

Nella finestra dell’editor scrivere queste poche righe di codice:

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
camera.start_preview()
sleep(10)
camera.stop_preview()
```

Come si può vedere, il listato inizia con l’importazione della libreria `picamera` (ecco il motivo per cui non si può usare il nome dello script con il nome della libreria).

Quindi viene importata anche la libreria `time`.

Con l’istruzione `camera = PiCamera()` si crea l’oggetto `camera` che eredita i metodi della libreria.

Con l’istruzione `camera.start_preview()` viene aperta la finestra di anteprima e con l’istruzione `camera.stop_preview()` viene chiusa.

Dalla libreria `time`, l’istruzione `sleep(10)` applica un ritardo di 10 secondi.

Salvare il file con CTRL + S e premere F5 per avviare lo script.

Dovrebbe aprirsi per 10 secondi una schermata di preview (Figura 12.4b) e quindi dovrebbe chiudersi.

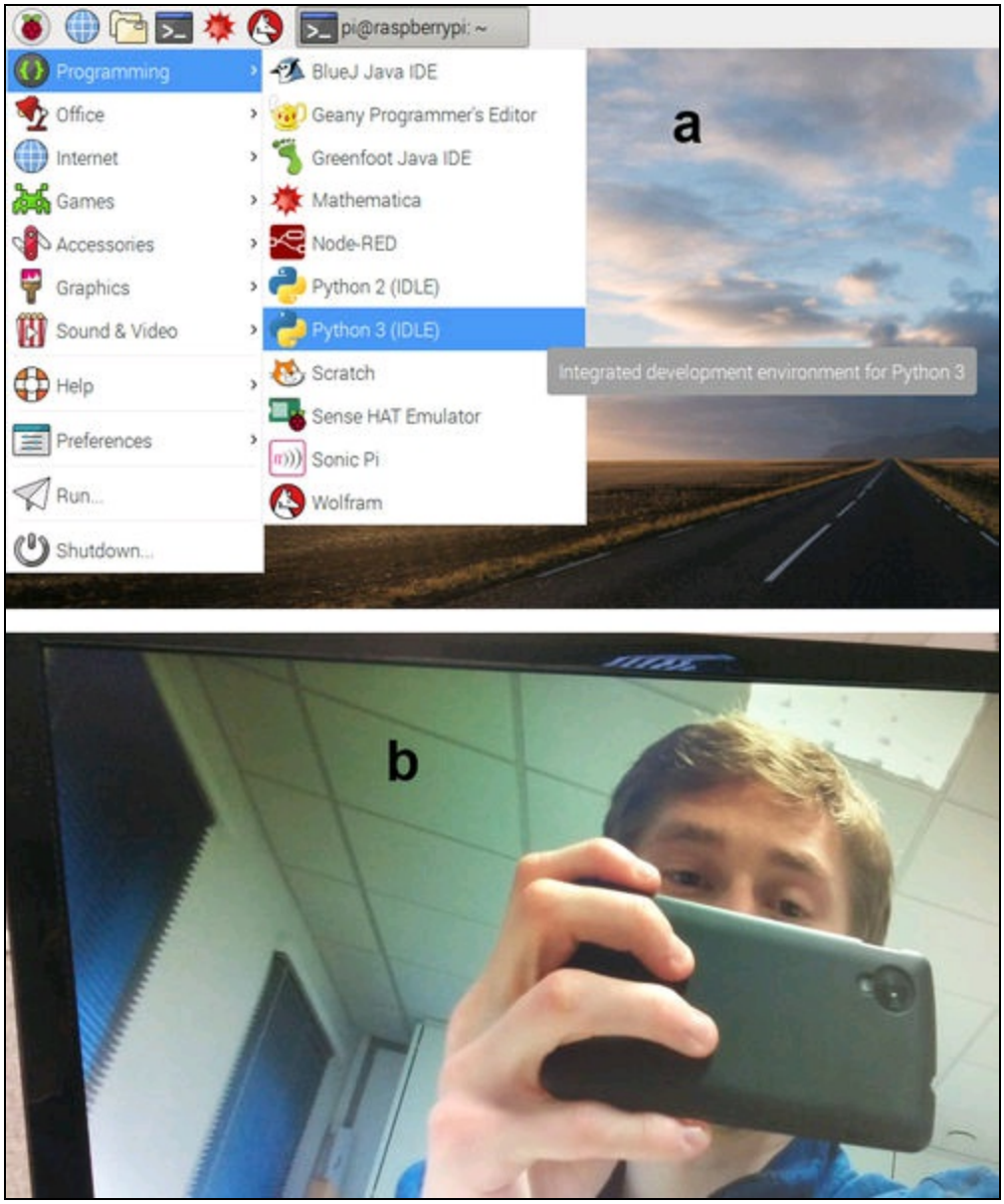


Figura 12.4 Avvio di Python 3 (a). Finestra di preview (b).

PiCamera code snippets

La libreria `picamera` consente una grande varietà di impostazioni e di effetti video. Ecco alcuni frammenti di codice per fare un po' di esperienza.

Rotazione

```
camera.rotation = 180
camera.start_preview()
sleep(10)
camera.stop_preview()
```

Scattare una foto singola

```
camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Scattare foto in sequenza

```
camera.start_preview()
for i in range(5):
    sleep(5)
    camera.capture('/home/pi/Desktop/image%s.jpg' % i)
    camera.stop_preview()
```

Registrazione video

```
camera.start_preview()
camera.start_recording('/home/pi/video.h264')
sleep(10)
camera.stop_recording()
camera.stop_preview()
```

Riproduzione della clip video

Dal terminale, digitare il seguente comando:

```
omxplayer video.h264
```

Aggiungere un testo alla foto

```
camera.start_preview()
camera.annotate_text = "Hello world!"
sleep(5)
camera.capture('/home/pi/Desktop/text.jpg')
camera.stop_preview()
```

Controllo della luminosità

```
camera.start_preview()
camera.brightness = 70
```

```
sleep(5)
camera.capture('/home/pi/Desktop/bright.jpg')
camera.stop_preview()
```

Controllo del contrasto

```
camera.start_preview()
for i in range(100):
    camera.annotate_text = "Contrast: %s" % i
    camera.contrast = i
    sleep(0.1)
    camera.stop_preview()
```

Regolazione del colore

```
from picamera import PiCamera, Color
camera.start_preview()
camera.annotate_background = Color('blue')
camera.annotate_foreground = Color('yellow')
camera.annotate_text = " Hello world "
sleep(5)
camera.stop_preview()
```

Effetti video

Con l'istruzione `camera.image_effect` si può applicare una vasta serie di effetti video: `none`, `negative`, `solarize`, `sketch`, `denoise`, `emboss`, `oilpaint`, `hatch`, `gpen`, `pastel`, `watercolor`, `film`, `blur`, `saturation`, `colorswap`, `washedout`, `posterize`, `colorpoint`, `colorbalance`, `cartoon`, `deinterlace1` e `deinterlace2`. Quello predefinito è `none`. Provare a cambiare l'effetto in questo frammento di codice:

```
camera.start_preview()
camera.image_effect = 'colorswap'
sleep(5)
camera.capture('/home/pi/Desktop/colorswap.jpg')
camera.stop_preview()
```

Si possono mettere gli effetti anche in un loop:

```
camera.start_preview()
for effect in camera.IMAGE_EFFECTS:
    camera.image_effect = effect
    camera.annotate_text = "Effect: %s" % effect
    sleep(5)
    camera.stop_preview()
```

Controllo dell’esposizione

Con l’istruzione `camera.exposure_mode` si può impostare l’esposizione su una modalità per applicare un effetto particolare. Le opzioni sono: `off`, `auto`, `night`, `nightpreview`, `backlight`, `spotlight`, `sports`, `snow`, `beach`, `verylong`, `fixedfps`, `antishake` e `fireworks`. L’impostazione predefinita è `auto`.

Provare a cambiare l’esposizione in questo frammento di codice:

```
camera.start_preview()
camera.exposure_mode = 'beach'
sleep(5)
camera.capture('/home/pi/Desktop/beach.jpg')
camera.stop_preview()
```

ZeroView

Se si usa un Raspberry Pi Zero W, si consiglia di acquistare il kit ZeroView dal sito ufficiale. Il kit è completo di supporto sagomato per la videocamera, di viti di fissaggio in plastica e di due ventose per un eventuale montaggio su una finestra o su una superficie trasparente. Se non si usano le ventose, si può montare ZeroView su una qualsiasi altra superficie, usando semplicemente quattro viti.

La Figura 12.5a illustra il supporto ZeroView montato e la Figura 12.5b mostra il connettore a nastro studiato appositamente per la porta CSI di Raspberry Pi Zero W, che è leggermente più piccola di quella standard.

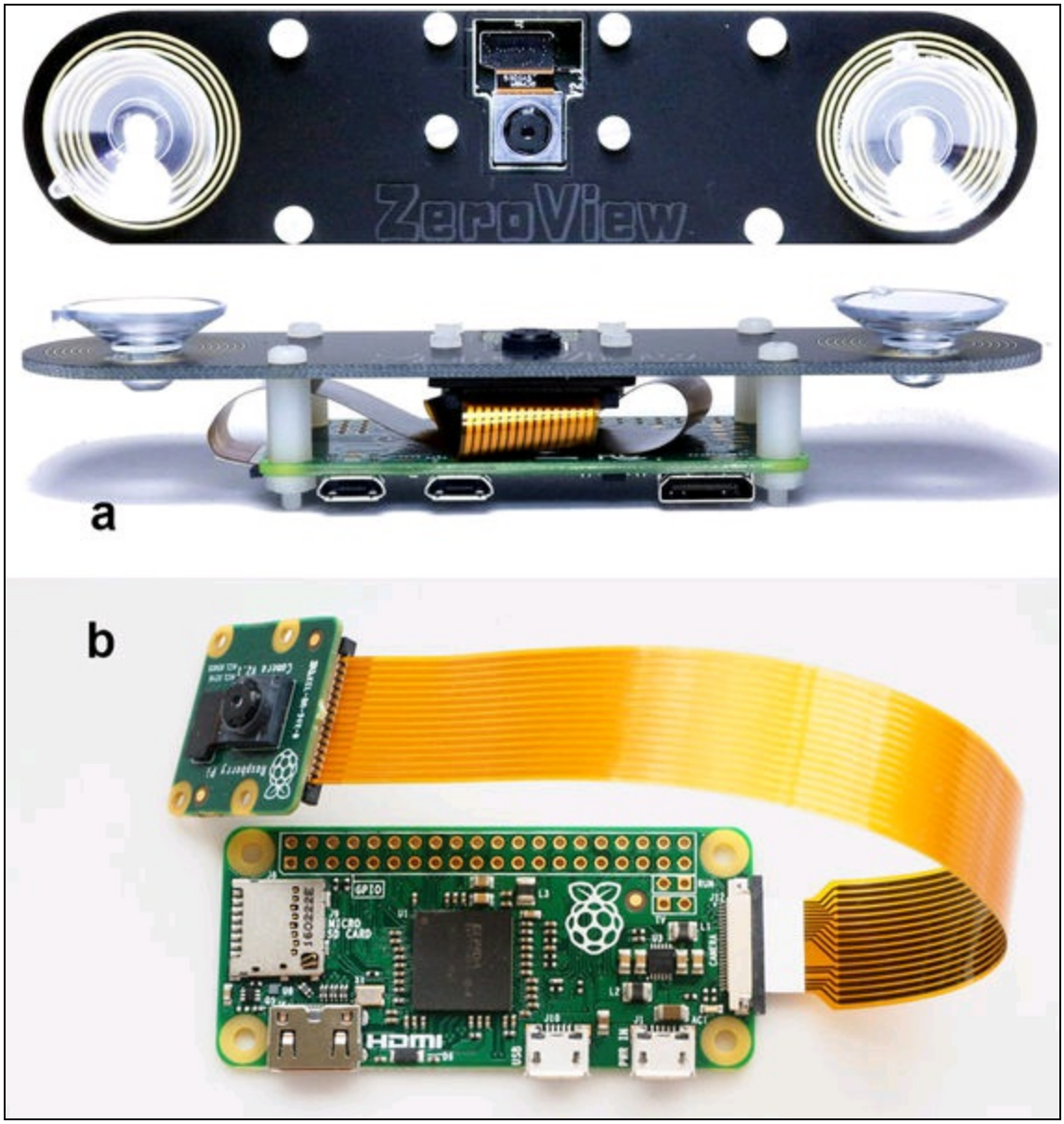


Figura 12.5 Il supporto ZeroView (a). Il connettore a nastro per Raspberry Pi Zero W (b).

Videosorveglianza

Per lo sviluppo del nostro script di videosorveglianza, un ringraziamento va a Adrian Rosebrock, autore e blogger del sito http://www.pyimagesearch.com. Il suo blog è dedicato ai programmatori che vogliono studiare il funzionamento dei motori di ricerca di immagini e

ama condividere le sue esperienze mettendo a disposizione risorse open source. Nel sito si trovano vari articoli sull'uso di OpenCV con esempi fatti con Python ed è autore di *Deep Learning for Computer Vision*.

Il codice commentato

Lo script integra un sistema basato su OpenCV e varie librerie per l'invio di email e l'uso della porta GPIO. Fortunatamente queste librerie sono già incluse in RaspBian, per cui non serve installare nient'altro.

Quelle che seguono sono solo alcune indicazioni per modificare le parti più importanti, in modo da poter adattare lo script per i propri scopi. Per praticità, non verrà pienamente rispettata l'indentazione del codice che Python vorrebbe, perché si uscirebbe dai margini della pagina stampata. Si prega quindi di fare riferimento al listato originale.

Le librerie importate all'inizio servono alla gestione SMTP, ovvero all'invio di un'email tramite un server di posta. Per questo utilizzo del programma è necessario connettere Raspberry Pi alla rete di casa (Wi-Fi o Ethernet).

```
from smtplib import SMTP_SSL as SMTP
from email.mime.text import MIMEText
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email import Encoders
import sys
import os
import re
```

Nella sezione seguente vanno cambiati i parametri per il server SMTP. Si consiglia di impiegare il server SMTP normalmente usato per l'invio di posta dal computer di casa.

Le variabili `sender` e `destination` contengono gli indirizzi del mittente e del destinatario, ovvero l'indirizzo con il quale si mandano normalmente le email e l'indirizzo cui si vuol inviare il messaggio. Le variabili `username` e `password` sono il nome utente e la password usati per l'invio al server SMTP. Il tipo di messaggio è di testo e la variabile `content` indica

il contenuto del messaggio mentre `subject` il soggetto del messaggio. L'oggetto `msg` eredita i metodi della libreria `MIMEMultipart`.

```
SMTPserver = 'smtps.server.xxx'
sender = 'sender@email.com'
destination = 'destination@email.com'
username = "my@email.com"
password = "myPassword"
text_subtype = 'plain'
content="""Messaggio"""
subject="Allarme"
msg = MIMEMultipart()
```

Quella che segue è la funzione `email_send()` per l'invio di un'email con un allegato, che nel nostro caso è un'immagine jpg dell'intruso o dell'oggetto che si è mosso nella stanza monitorata. Questa funzione di invio email può essere chiamata o meno all'interno dello script, a seconda che si voglia o meno ricevere un'email quando qualcuno o qualcosa attiva il sistema. Se l'invio viene eseguito correttamente, sul terminale verrà stampato "Invio eseguito". Se ci sono problemi di invio, apparirà "Invio fallito".

```
def email_send():
try:
    msg['Subject']= subject
    msg['From'] = sender
    msg.attach(part)
    conn = SMTP(SMTPserver)
    conn.set_debuglevel(False)
    conn.login(username, password)
try:
    conn.sendmail(sender, destination, msg.as_string())
    print("Invio eseguito")
finally:
    conn.quit()
    except Exception: print("Invio fallito")
```

Quelle che seguono sono le librerie che devono essere importate per la gestione della videocamera, della data, dell'orario, degli effetti video, dei file json, di cv2 (OpenCV2) e della porta GPIO.

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import datetime
import imutils
import json
import cv2
import RPi.GPIO as GPIO
```

Con le seguenti istruzioni si imposta l'uscita GPIO26. Se si vuole usare un altro pin, bisogna cambiarlo anche qui.

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(26,GPIO.OUT)
```

Questa istruzione permette il caricamento del file json di configurazione (si veda più avanti).

```
conf = json.load(open('configurazione.json'))
```

L'oggetto `camera` eredita i metodi della libreria `picamera` e i parametri della videocamera vengono impostati in base alla lettura del parametro `resolution` del file json di configurazione (si veda più avanti). Si fa notare l'uso della variabile `tuple` che in Python crea una lista di valori separati da virgole:

```
camera = PiCamera()
camera.resolution = tuple(conf["resolution"])
camera.framerate = conf["fps"]
rawCapture = PiRGBArray(camera, size=tuple(conf["resolution"]))
```

Vengono quindi definite alcune variabili che verranno utilizzate nello script per stabilire il tempo di inizio e di fine della cattura dei frame.

```
count = 0
avg = None
motionCounter = 0
lastUploaded = datetime.datetime.now()
```

Dopo la stampa sul terminale della scritta "Avvio…" il sistema attende 2 secondi e iniziare la cattura video. Si fa notare che nella finestra di anteprima viene sovraimposto nella parte superiore il testo "Nessun movimento" e, in basso, la data e l'orario (Figura 12.6a). Viene usata la libreria `imutils` per ridimensionare il frame e la libreria `datetime` per l'orario preso da Internet, per cui i nomi dei giorni e dei mesi sono in inglese.

Con la libreria `cv2` si imposta la maschera per rilevare un oggetto in movimento. Si fa notare che il parametro `tresh` legge il valore di soglia, ovvero il parametro `threshold` dal file json di configurazione (si veda più avanti). La variabile `motionFlag` viene posta a `False`.

```
print "Avvio..."
time.sleep(2)
for f in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    frame = f.array
    currentTime = datetime.datetime.now()
    text = "Nessun movimento"
    motionFlag = False
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)
    …
    thresh = cv2.threshold(frameDelta, conf["threshold"], 255, cv2.THRESH_BINARY)
[1]
    thresh = cv2.dilate(thresh, None, iterations=2)
```

Se viene rilevato un movimento, il soggetto viene scomposto e appaiono dei riquadri verdi attorno ai bordi del soggetto che presentano una diversità di contrasto diversa dall'immagine in background. Questa soglia di contrasto può essere modificata con il parametro `threshold`. Quando il valore di soglia viene superato, apparirà il testo "Movimento rilevato" (Figura 12.6b). La variabile `motionFlag` viene posta a `True`.

```
text = "Movimento rilevato"
motionFlag = True
ts = timestamp.strftime("%A %d %B %Y %I:%M:%S%p")
```

A questo punto, se la variabile `motionFlag` è `True`, inizia il confronto fra il tempo corrente (`currentTime`) e il tempo impostato per il rilevamento (`lastTime`) impostato dal parametro `min_time` nel file json di configurazione (si veda più avanti) e inizia il conteggio dei movimenti rilevati con la variabile `motionCounter`.

```
if text == "Movimento rilevato":
    if (currentTime - lastTime).seconds >= conf["min_time"]:
        motionCounter += 1
```

Se il numero dei movimenti rilevati supera quella impostato dal parametro `min_motion` nel file json, si può decidere se salvare localmente il file dell'immagine. Questa opzione viene impostata dal parametro `use_img` nel file json.

Si fa notare che il nome del file `img = 'foto_' + str(count) + '.jpg'` segue una numerazione progressiva. I nomi dei file che vengono salvati

localmente saranno `foto_1.jpg`, `foto_2.jpg` e così via. L'istruzione per scrivere il file su disco è `cv2.imwrite(img, frame)`.

```
if motionCounter >= conf["min_motion"]:
    if conf["use_img"]:
        count += 1
        img = 'foto_' + str(count) + '.jpg'
        cv2.imwrite(img, frame)
```

Se non si salva il file con l'immagine, si accende il LED/relè sul pin GPIO26 mentre viene stampato "LED ON" sul terminale, con le istruzioni `GPIO.output(26, GPIO.HIGH)` e `print "LED ON"`.

```
GPIO.output(26,GPIO.HIGH)
print "LED ON"
```

Se nel file json è stato impostato l'invio dell'email con il parametro `use_email`, viene creato l'oggetto `part` che imposta il formato MIME (*Multipurpose Internet Mail Extensions*) per comporre il messaggio di posta. Il messaggio viene automaticamente codificato in formato base64 e quindi viene spedito tramite la funzione `email_send()`, che abbiamo visto prima. Il messaggio di posta invierà in allegato il file `foto_1.jpg` e successivamente invierà il file `foto_2.jpg` e così via, in base alle volte che la videocamera rileva un nuovo movimento. La Figura 12.6c illustra un esempio di ricezione di un'email di allarme con allegata la foto dell'intruso all'indirizzo designato.

```
if conf["use_email"]:
    part = MIMEBase('application', "octet-stream")
    part.set_payload(open(t, "rb").read())
    Encoders.encode_base64(part)
    part.add_header('Content-Disposition', 'attachment; filename='+ img)
    email_send()
```

Viene quindi azzerato il contatore dei movimenti e viene spento il LED. Sul terminale viene stampato "LED OFF" e la variabile `motionFlag` viene posta a `False`.

```
motionCounter = 0
GPIO.output(26,GPIO.LOW)
print "LED OFF"
motionFlag = False
```

La finestra di preview della videocamera viene attivata in base al parametro `video_preview` nel file json.

```
if conf["video_preview"]:
    cv2.imshow("Videocamera di sicurezza", frame)
```

Figura 12.6 Finestra preview con “Nessun movimento” (a). Finestra preview con “Movimento rilevato” (b). L’email ricevuta con il file allegato dell’intruso (c).

Il file JSON

È stato citato varie volte e quindi scopriamo a cosa serve. Di solito un file JSON (*JavaScript Object Notation*) viene usato nell'ambiente di programmazione JavaScript. Grazie alla sua versatilità, viene usato spessissimo come file di configurazione in altri ambienti. In pratica, è uno script che supporta il linguaggio JavaScript e quindi accetta la sintassi tipica del linguaggio.

Il nostro file di configurazione si chiama `configurazione.json` contiene il codice seguente:

```
{
    "use_img": true,
    "use_email": true,
    "video_preview": true,
    "min_time": 3.0,
    "min_motion": 8,
    "threshold": 5,
    "resolution": [640, 480],
    "fps": 16,
    "min_area": 5000
}
```

Come si può vedere, i parametri possono essere posti a `true` o `false` oppure contenere valori numerici e array. Leggendo questi parametri con l'istruzione `conf = json.load(open('configurazione.json'))`, si possono impostare le funzionalità dello script di videosorveglianza, senza toccare il codice.

Per esempio, se non si vogliono salvare i file delle immagini catturate e usare la videosorveglianza solo per accendere il LED/relè di allarme, basta scrivere nel file json la seguente istruzione:

```
"use_img": truefalse
```

Allo stesso modo, se non si desidera l'invio di email:

```
"use_email": false
```

Se non si vuole visualizzare la finestra di anteprima:

```
"video_preview": false
```

Gli altri parametri del file `configurazione.json` permettono di impostare altre funzioni.

- `min_time`: predefinito 3 secondi. È il tempo minimo per rilevare il movimento.
- `min_motion`: predefinito `8` frame. È il numero minimo di frame prima di attivare il sistema di accensione del LED, di salvataggio dei file e/o di invio dell'email. Si può aumentare o diminuire il valore per rendere più o meno sensibile il rilevamento.
- `threshold`: predefinito `5`. È il valore che definisce la soglia di sensibilità per il rilevamento del movimento. Si può aumentare o diminuire questo valore a seconda che si voglia rendere il contrasto più o meno sensibile.
- `resolution`: predefinito 640x480 pixel. È la risoluzione video, ovvero la dimensione del riquadro video (frame). Si consiglia di lasciare inalterata questa risoluzione per evitare rallentamenti nel flusso video o nell'invio degli allegati via email.
- `fps`: predefinito `16`. Imposta i frame al secondo. Si consiglia di lasciare questo valore di *frame rate* per evitare rallentamenti nel flusso video.
- `min_area`: predefinito `5000`. È il valore minimo dell'area verde del riquadro attorno al soggetto da rilevare. Di solito non serve modificarla.

Avvio manuale dello script

Per avviare manualmente lo script di videosorveglianza, aprire il terminale e digitare i seguenti comandi, presupponendo che lo script sia in una cartella chiamata "videosorveglianza" creata nella directory `home/pi`:

```
cd videosorveglianza
python videosorveglianza.py
```

Per fermare lo script, digitare CTRL+C dal terminale.

Avvio automatico dello script

Quanto all'avvio automatico dello script di videosorveglianza al boot di sistema, ci sono almeno due modi per farlo. Il primo consiste nell'avviare il programma in background. In questo modo non si vedrebbe la finestra di preview. Il secondo metodo (consigliato) prevede l'avvio del programma con l'interfaccia grafica del desktop LXDE e quindi visualizzando la finestra di preview.

Primo metodo

Dal terminale digitare i seguenti comandi:

```
cd /etc
sudo leafpad rc.local
```

Si aprirà il file `rc.local` nell'editor di testo. Aggiungere le seguenti righe nella parte non commentata prima di `exit 0`:

```
cd /home/pi/videosorveglianza
python videosorveglianza.py &
exit 0
```

Secondo metodo

Rendere eseguibile lo script di Python come spiegato nel Capitolo 3 al paragrafo "Rendere eseguibili gli script di Python". Dal terminale digitare i seguenti comandi:

```
cd /home/pi/videosorveglianza
sudo chmod +x videosorveglianza.py
sudo cp videosorveglianza.py /usr/bin
sudo cp configurazione.json /usr/bin
```

Provare ad avviare lo script eseguibile che è stato copiato nella cartella `usr/bin`, digitando dal terminale solo il nome dello script, ovvero `videosorveglianza.py`. Se si avvia regolarmente, si può editare l'autostart dell'interfaccia grafica LXDE. Dal terminale digitare i seguenti comandi:

```
sudo leafpad /home/pi/.config/lxsession/LXDE-pi/autostart
```

Si aprirà l’editor di testo con l’elenco delle applicazioni che si avviano al boot del desktop. Aggiungere all’elenco la nostra applicazione `videosorveglianza.py` con il percorso di avvio `/usr/bin`, preceduto dal simbolo `@`, ovvero `@/usr/bin/videosorveglianza.py` (Figura 12.7). Salvare il file e riavviare.

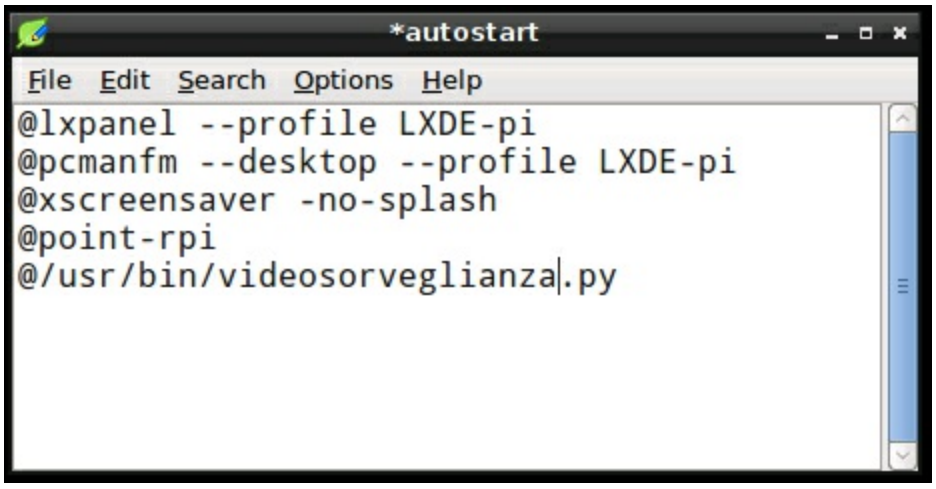


Figura 12.7 Editor di testo con il file autostart dell’interfaccia grafica LXDE.

Apriti Sesamo

Descrizione

Il nome un po’ scherzoso di questo progetto svela un po’ le nostre intenzioni. Accendere e spegnere le luci, alzare o abbassare le tapparelle, aprire e chiudere una porta, teleguidare un robot con comandi vocali… diciamolo, è davvero cool. Questo progetto si basa su Arduino UNO e uno shield appositamente creato per il riconoscimento vocale.

CODICE DI ESEMPIO

Tutti i file del progetto “Apriti Sesamo” sono disponibili nelle risorse del libro presso il sito dell’autore all’indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× Arduino UNO;
- 1× modulo EasyVR;
- 1× shield EasyVR per Arduino;
- 1× LED o un modulo relè;
- 1× servomotore;
- Attuatori vari.

EasyVR

EasyVR è un modulo molto sofisticato per il riconoscimento vocale, prodotto dall'azienda italiana Robotech SRL, che ha creato il marchio commerciale Veear. Il sito ufficiale è `http://www.veear.eu`.

La Figura 13.1 illustra il modulo EasyVR 3 che viene venduto in kit di montaggio. Il kit contiene la scheda, i connettori da saldare, il microfono con il cavetto e il connettore JST e un cavetto con il connettore JST da saldare a un altoparlante esterno (opzionale).

Ecco le caratteristiche principali.

- Fino a 32 comandi definiti dall'utente, suddivisi in gruppi fino a 15 gruppi Speaker Dependent (SD), 1 trigger SD e 1 gruppo Speaker Verification (SV) (max. 5), che possono essere addestrati in qualsiasi lingua.
- Una selezione di 26 comandi integrati per comandi Speaker Independent (SI) pronti all'uso, nelle seguenti lingue: inglese (USA); italiano; tedesco; francese; spagnolo; giapponese.
- Con la licenza Quick T2SI Lite (opzionale), fino a 28 dizionari di comandi Speaker Independent (SI), con un massimo di 12 comandi

ciascuno, per un totale di 336 comandi possibili.

- Tecnologia SonicNet per le comunicazioni wireless tra i moduli o qualsiasi altra sorgente sonora (audio CD, DVD, lettore MP3).
- Fino a circa 21 minuti di suoni o brani preregistrati.
- Fino a circa 120 secondi di registrazione e riproduzione di messaggi in diretta.
- Capacità di sincronizzazione labiale in tempo reale.
- Generazione di toni DTMF.

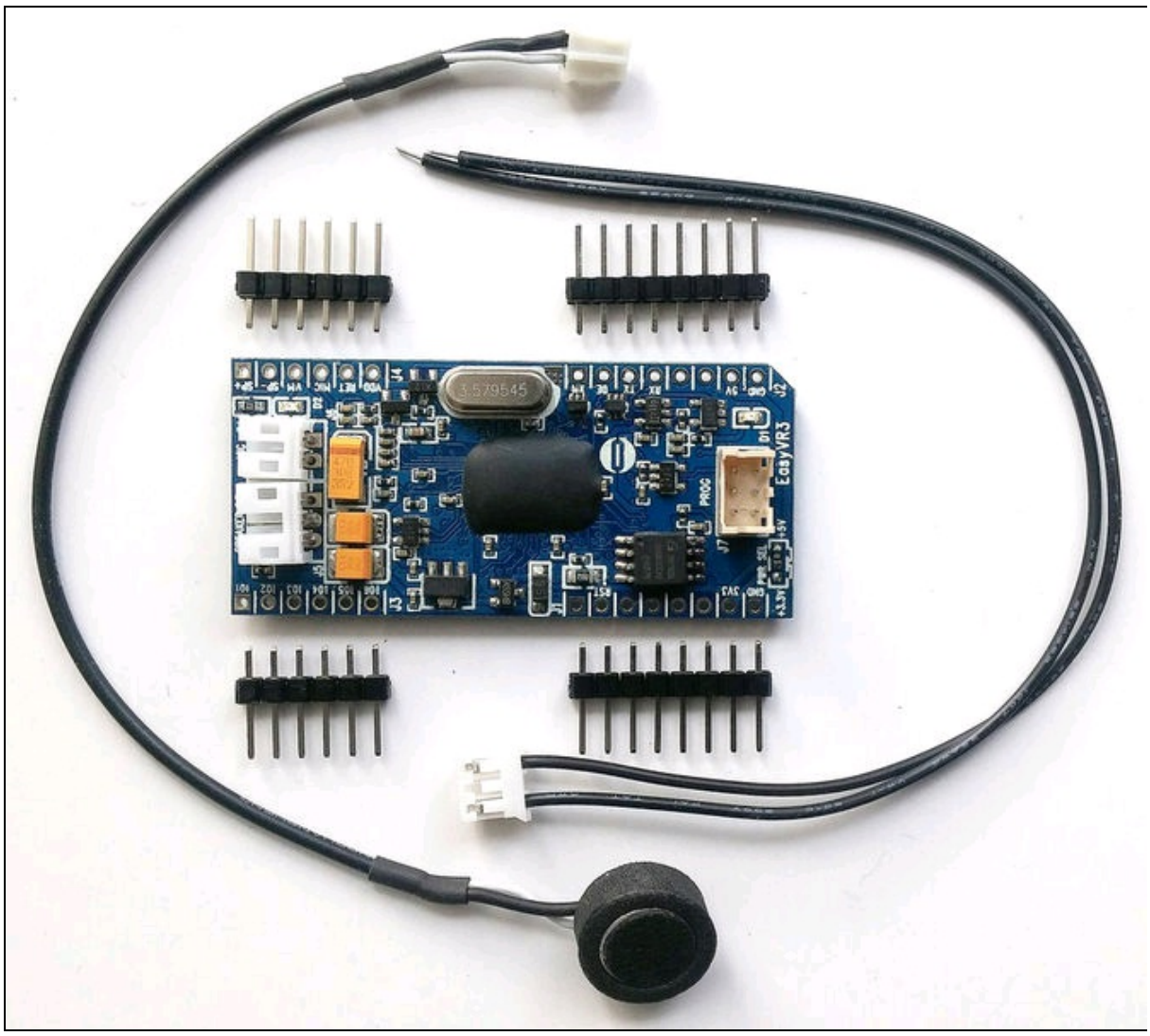


Figura 13.1 Kit di montaggio del modulo EasyVR.

- Uscita audio con supporto diretto di un altoparlante da 8Ω.
- Software facile da usare per programmare i comandi vocali e i messaggi audio.
- Interfaccia standard UART (alimentata a 3,3 V - 5 V).
- Protocollo seriale documentato, semplice e robusto per accedere e programmare attraverso l'host.
- 6 linee I/O di uso generale che possono essere controllate tramite comandi UART.

Oltre alle quattro strisce di connettori, si consiglia di saldare anche un piccolo altoparlante da 8 Ω al connettore JST che viene dato nel kit.

EasyVR shield 3

La Figura 13.2a illustra lo shield EasyVR 3 per Arduino, creato appositamente per ospitare il modulo EasyVR 3. Anche questo prodotto viene venduto con i connettori per Arduino da saldare. Dopo avere saldato i connettori al modulo EasyVR, questo va saldato allo shield, come indicato dalla freccia della Figura 13.2b.

Caratteristiche principali:

- compatibile con le schede Arduino che dispongono dell'interfaccia 1.0 Shield (UNO R3) e delle schede legacy: Arduino Duemilanove Arduino Uno Arduino Mega Arduino Leonardo Arduino Due
- supporta schede a 5 V e 3,3 V attraverso il pin IOREF;
- supporta la connessione diretta al PC con un chip separato USB seriale e una speciale modalità software "bridge" su schede con la sola interfaccia USB nativa per un facile accesso a EasyVR Commander;
- consente diversi modi di connessione seriale per il modulo EasyVR incorporato (tramite un jumper);

- supporta la mappatura dei pin seriali utilizzati dallo shield (in modalità SW);
- presa di uscita audio da 3,5 mm adatta per cuffie o uscita di linea.

Visto che bisogna effettuare molte saldature, a chi non avesse esperienza di saldatura, si consiglia il tutorial Adafruit, disponibile al seguente indirizzo: https://learn.adafruit.com/adafruit-guide-excellent-soldering/tools.

Una volta terminate tutte le saldature dei connettori, del modulo EasyVR allo shield EasyVR 3 e dell'altoparlante, si può inserire lo shield su una scheda Arduino, facendo attenzione che tutti i pin siano inseriti correttamente.

La Figura 13.2c mostra l'ingrandimento del jumper presente sullo shield. Questo jumper è molto importante per impostare le varie funzioni hardware e software come spiegato di seguito. Le posizioni del jumper sono SW, HW, PC, UP e LEO.

- *SW*: modalità SoftwareSerial, da usare per controllare il modulo EasyVR dallo sketch di Arduino attraverso una porta SoftwareSerial (usando i pin 12-13). È possibile collegare EasyVR Commander in questa modalità, a condizione che lo sketch in esecuzione abiliti la modalità bridge.
- *HW*: modalità seriale hardware, da usare per controllare il modulo EasyVR dallo sketch di Arduino attraverso la porta hardware 0-1.
- *PC*: modalità di connessione al PC, da usare per il collegamento diretto con EasyVR Commander. In questa modalità, la scheda Arduino è tenuta in reset e viene utilizzata solo come adattatore USB/seriale.
- *UP*: modalità di aggiornamento della memoria Flash, da usare per gli aggiornamenti del firmware o per scaricare i dati della soundtable e dei fonemi personalizzati nella memoria Flash da

EasyVR Commander. In questa modalità, la scheda Arduino viene tenuta in reset e viene utilizzata come adattatore USB/seriale.

- *LEO*: modalità Leonardo per schede Arduino che non dispongono di un adattatore USB/seriale separato, come Arduino Leonardo.

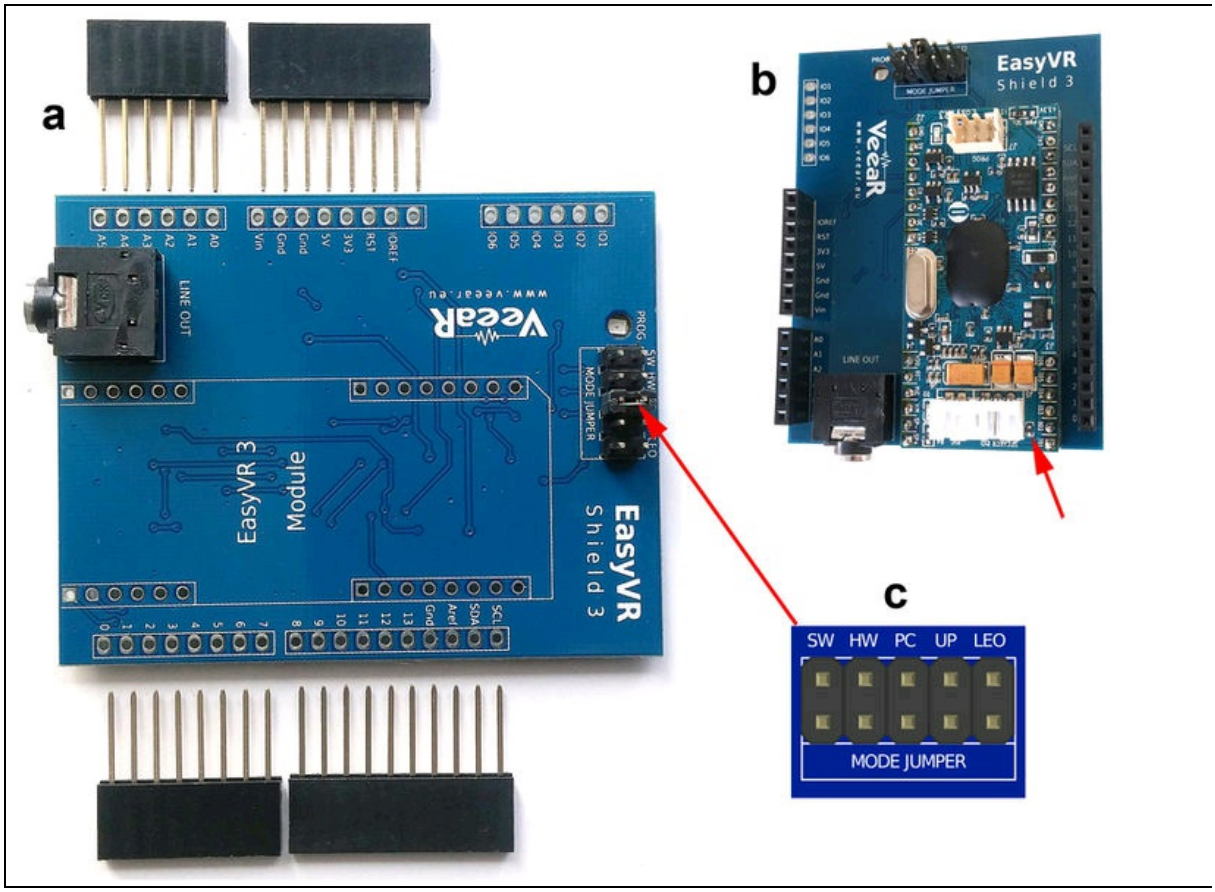


Figura 13.2 Lo shield EasyVR 3 per Arduino.

Il circuito elettrico

La Figura 13.3 illustra il semplice circuito di una scheda Arduino UNO con sopra lo shield EasyVR 3, collegato a un modulo relè al pin digitale 8 e a un servomotore al pin digitale 9. Il relè potrebbe comandare, per esempio, l'illuminazione di una stanza oppure una serratura elettrica.

Attenzione! Il servomotore collegato in questo circuito è solo di esempio e serve per vedere l'esecuzione del comando, come spiegato più avanti nel paragrafo "Voice Home Automation". Ovviamente, per aprire una porta reale serve un attuatore a pistone idraulico come quello illustrato nel Capitolo 1.

Se si vogliono controllare altri dispositivi, è sufficiente collegare altri moduli relè e comandarli tramite il software di gestione EasyVR Commander, di cui parliamo qui di seguito.

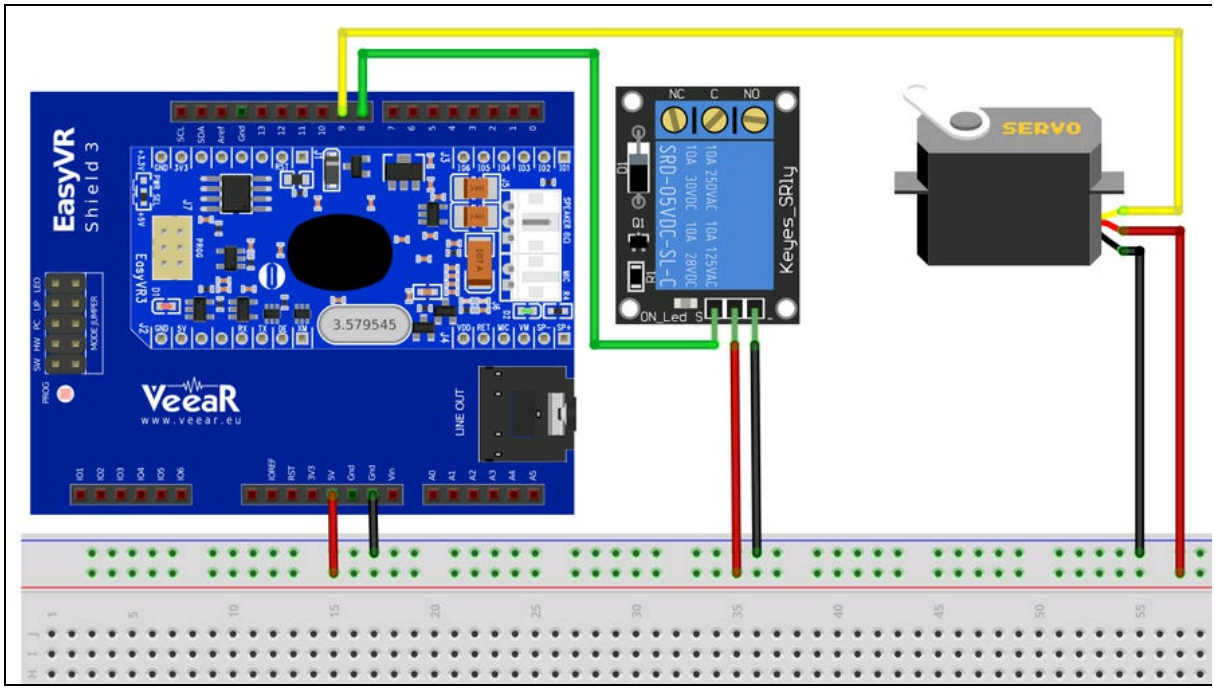


Figura 13.3 Schema di collegamento.

Il software a corredo

Dalla pagina download del sito ufficiale è possibile scaricare il manuale utente, la libreria per Arduino e soprattutto il software *EasyVR Commander v3.12.1* (aggiornato ad aprile 2017). La pagina download è disponibile all'indirizzo `http://www.veear.eu/downloads`, dove sono scaricabili diverse risorse gratuite. Una volta scaricato il software EasyVR Commander, durante la sua installazione viene chiesta l'installazione anche dei programmi *Quick Synthesis*, *FluentChip* e la versione demo di *Quick T2SI* (si veda il paragrafo dedicato più avanti).

Il software EasyVR Commander può essere utilizzato per configurare facilmente il modulo EasyVR collegato al PC mediante un cavo USB collegato alla scheda Arduino, che in questo caso funziona come un adattatore USB/seriale.

È possibile definire gruppi di comandi o password e generare il codice per gestirli, ovvero uno sketch da caricare direttamente in Arduino. Lo sketch generato automaticamente contiene tutte le funzioni e le routine per gestire le attività di riconoscimento vocale. Ovviamente, sarà necessario modificare il codice generato per implementare la logica dell'applicazione, ovvero le istruzioni per accendere o spegnere un LED e attivare altre funzioni, come si vedrà più avanti.

Inoltre, con EasyVR Commander è possibile caricare nel modulo EasyVR nuovi messaggi audio prodotti con Quick Synthesis e nuovi comandi Speaker Independent con Quick T2SI (si veda il paragrafo dedicato più avanti).

Libreria EasyVR per Arduino

Per poter funzionare con il modulo EasyVR, bisogna installare nell'IDE di Arduino la libreria dedicata. Per installare la libreria

EasyVR con le versioni recenti dell'IDE, basta aprire dal menu *Sketch* > *#include libreria* > *Gestione librerie* la finestra omonima. La Figura 13.4 illustra la libreria EasyVR pronta per l'installazione. Per trovarla subito, si consiglia di inserire nel filtro di ricerca il testo "easyvr". La versione attuale della libreria è la 1.10.1 (maggio 2017).

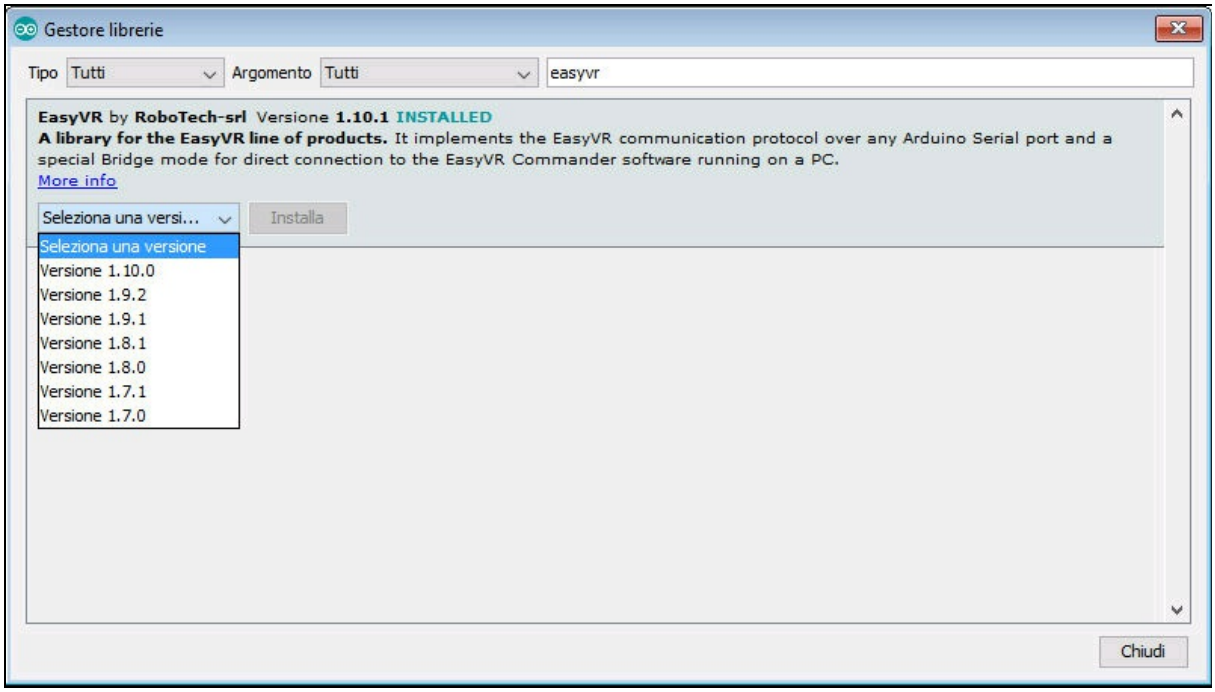


Figura 13.4 Installazione della libreria EasyVR per Arduino.

Una volta installata la libreria, dal menu degli esempi si può caricare lo sketch *TestEasyVR* per testare il modulo EasyVR seguendo le istruzioni del manuale in inglese.

Consigliamo invece di seguire il nostro tutorial su come creare un set di comandi personalizzati con il programma EasyVR Commander che, nella versione attuale (3.12.1 di maggio 2017), è in grado di generare lo sketch per Arduino e rendere disponibili i comandi vocali per controllare tutto quel che si vuole.

EasyVR Commander

Prima di iniziare, bisogna collegare il cavo USB e verificare che Arduino e lo shield EasyVR siano accesi correttamente (LED rosso acceso) e che il jumper dello shield sia messo in posizione PC. Quindi avviare EasyVR Commander.

Selezionare la porta COM da utilizzare dalla barra degli strumenti o dal menu *File*, quindi scegliere il comando *Connect* come indicato dalla freccia della Figura 13.5a. Se tutto va bene, apparirà nella parte inferiore della finestra il messaggio *Connected to EasyVR 3 (Rev 4) on COM xx* (dove *xx* è il numero della porta seriale).

Nella parte sinistra della finestra sono elencati vari tipi di comandi (elencati di seguito); in pratica, tutto il contenuto della memoria del modulo EasyVR.

- *Trigger*: è un gruppo speciale in cui è installata la parola di attivazione “Robot”. Se ne può aggiungere un’altra a piacere. Le parole di trigger possono venire utilizzate per avviare il processo di riconoscimento.
- *Group* (da 1 a 15): sono i gruppi nei quali è possibile aggiungere i comandi definiti dall’utente, organizzati in sottoinsiemi.
- *Password*: è un gruppo speciale di “password vocali” (fino a 5), utilizzando la tecnologia Speaker Verification (SV).
- *Wordset*: è un insieme di comandi incorporati pronti all’uso in diverse lingue, fra cui l’italiano.
- *SoundTable*: è un elenco di campioni audio compressi (suoni, avvisi e così via) caricati nel modulo e che possono essere ascoltati tramite l’altoparlante collegato al modulo.
- *Messages*: sono registrazioni eseguite in tempo reale e salvate nella memoria interna.

Riconoscimento vocale

La funzione di riconoscimento di EasyVR funziona su un singolo gruppo alla volta. In questo modo l'utente può usare contemporaneamente tutti i comandi desiderati appartenenti allo stesso gruppo, anche se, come vedremo, potrà passare da un gruppo di comandi all'altro con semplici istruzioni nello sketch.

Quando EasyVR Commander si collega al modulo, legge tutti i comandi e i gruppi definiti dall'utente dalla memoria del modulo EasyVR.

È possibile aggiungere un nuovo comando selezionando il gruppo in cui è necessario crearlo, utilizzando le icone della barra strumenti o il menu *Edit*.

A ogni comando va assegnata un'etichetta. Viene quindi registrato il comando vocale in due fasi, tramite il microfono del modulo. Ecco come fare.

1. Selezionare un gruppo, per esempio, *Group 1*.
2. Fare clic sull'icona + (*Add Command*) (Figura 13.5b).
3. Dare un nome all'etichetta, per esempio, "APRITI_SESAMO".
4. Fare clic sull'icona *Train Command* (Figura 13.5c).
5. Apparirà una finestra di dialogo con l'indicazione *Command 0 of Group 1* "APRITI_SESAMO" - *Phase 1/2*, ovvero "Fase 1 di 2 del comando 0 del gruppo 1 APRITI_SESAMO". È da notare che viene aggiunto automaticamente il trattino basso fra le parole.
6. Facendo clic sul pulsante *Phase 1*, apparirà una piccola finestra *Speak now* (Figura 13.5d) che invita a pronunciare la frase del comando che, nel nostro caso, sarà "apriti Sesamo".
7. Pronunciare la frase entro 5 secondi, tenendosi a una distanza a piacere dal microfono (ma non troppo distante) e possibilmente in un ambiente tranquillo, senza rumore di fondo.
8. Se non vengono visualizzati errori, apparirà nuovamente la finestra per la seconda fase.

9. Fare clic sul pulsante *Phase 2* e quando appare nuovamente *Speak now* pronunciare la stessa frase, cercando di mantenere la stessa inflessione e velocità di pronuncia.
10. Se non vengono visualizzati errori, la finestra di addestramento scompare e verrà visualizzato il comando “APRITI_SESAMO” con accanto le colonne *Trained 2* e *Conflict OK* (Figura 13.5e).
11. Se appaiono finestre di errore durante le fasi di registrazione del comando vocale, ripetere la procedura.

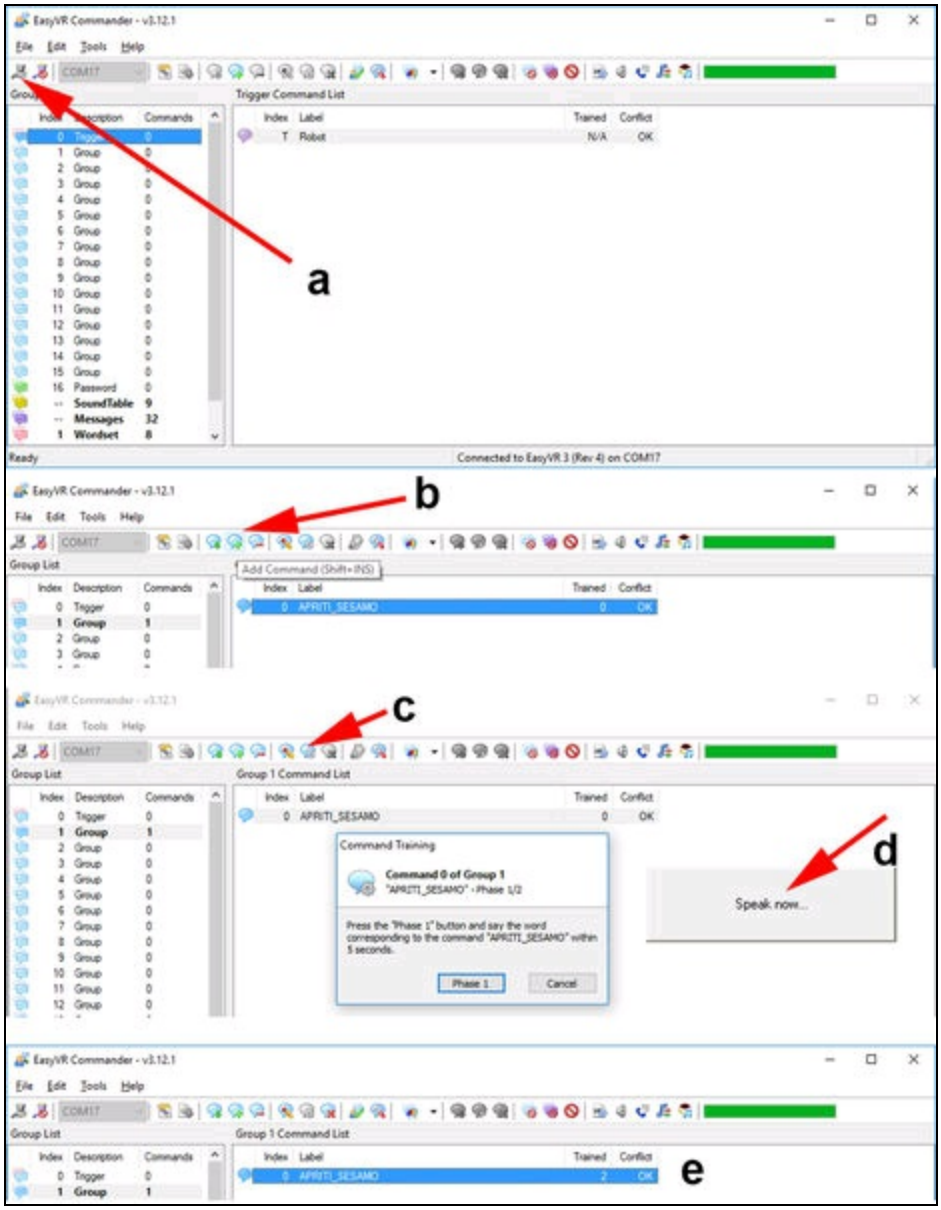


Figura 13.5 Procedura di connessione e di registrazione dei comandi vocali.

Con questo comando, come vedremo, verrà acceso il LED/modulo relè, collegato al pin 8 di Arduino.

Con le icone della barra strumenti è possibile applicare diverse azioni sul comando.

- *Insert Command*: inserisce un comando nella lista.
- *Add Command*: aggiunge un comando alla lista.

- *Remove Command*: rimuove un comando dalla lista.
- *Rename Command*: rinomina il comando.
- *Train Command*: avvia la procedura di addestramento del comando.
- *Test Group*: permette di testare uno o più comandi del gruppo e vedere se vengono riconosciuti senza problemi. Fare clic sull'icona *Test Group* e pronunciare un comando della lista (nel nostro caso c'è solo "apriti Sesamo"). Se il comando viene riconosciuto correttamente, l'etichetta lampeggerà in verde, altrimenti bisogna ripetere il comando finché non viene riconosciuto.
- *Recognition Settings*: apre una finestra per modificare le impostazioni predefinite del riconoscimento vocale (di solito non serve toccare nulla).
- *Set Language*: imposta la lingua del comando. Per esempio, impostando la lingua italiana, i tre Wordset preimpostati faranno riferimento a una serie di comandi in italiano.

Ripetere la procedura di *training* per registrare tutti i comandi vocali che possono servire allo scopo. Per esempio, si potrebbero aggiungere allo stesso gruppo oppure negli altri gruppi i comandi "accendi" e "spegni", "apri" e "chiudi" e così via. Se, per esempio, si vuole controllare un rover o un drone, si potrebbero registrare i comandi "destra", "sinistra", "stop", "indietro", "avanti" e così via.

Per restare in tema con il nostro esempio, abbiamo registrato, sempre nel gruppo 1, un secondo comando vocale etichettato come "CHIUDITI_SESAMO" (Figura 13.6a). Con questo comando, come vedremo, verrà spento il LED/modulo relè, collegato al pin 8 di Arduino.

Attenzione! Se si usano frasi composte da due o più parole (come "apriti Sesamo") bisogna etichettarle con il trattino basso (underscore). Questo è importante perché, per esempio, il Wordset in italiano che

viene dato con EasyVR Commander, purtroppo dà un errore in fase di compilazione dello sketch per Arduino, perché quattro delle sue etichette (A sinistra, A destra, In alto e In basso) non sono unite con il trattino basso (Figura 13.6). Bisognerà correggere questo piccolo bug nello sketch di Arduino che viene generato da EasyVR Commander.

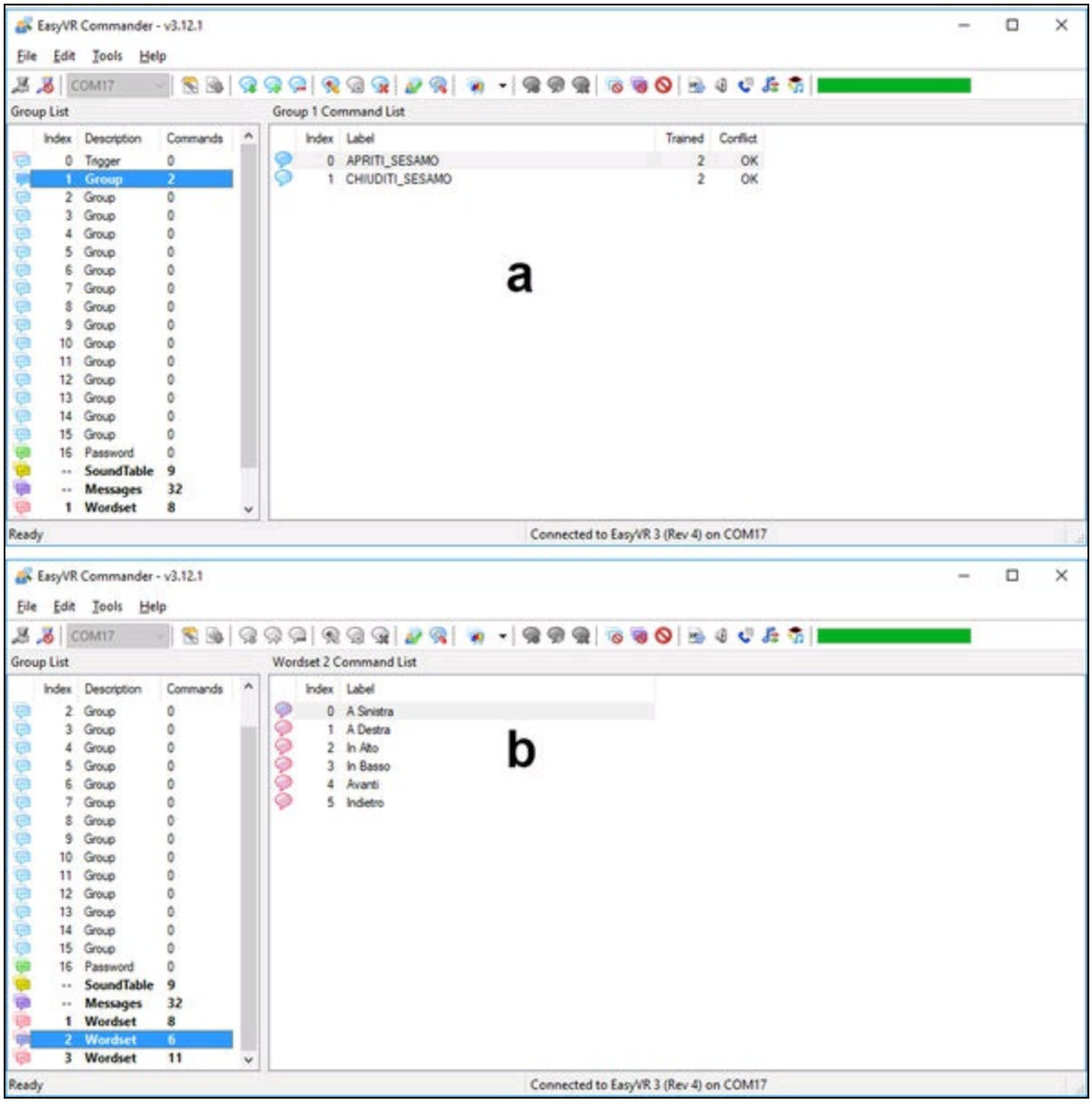


Figura 13.6 Il comando CHIUDITI_SESAMO del gruppo 1 (a). Il wordset in italiano (b).

Lo sketch di Arduino

Una volta terminato l'addestramento dei comandi, si può generare lo sketch per Arduino. Questo conterrà tutti i comandi, SoundTable e Wordset preimpostati da EasyVR Commander, più i nostri comandi appena aggiunti.

Fare clic sull'icona *Generate Code* (Figura 13.7) e dare un nome al file che verrà salvato con estensione `.ino`. Questo file andrà messo in una cartella con lo stesso nome perché Arduino non accetta che un suo file sia senza cartella.

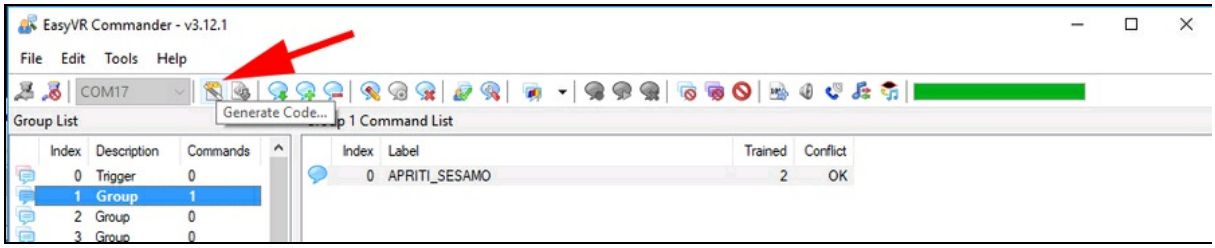


Figura 13.7 Generazione del codice per Arduino.

Perché lo sketch possa essere utile al nostro scopo, bisognerà fare qualche piccola modifica. Innanzitutto, individuare la riga seguente

```
group = EasyVR::TRIGGER; //<-- start group (customize)
```

… e cambiarla in:

```
group = 1; //<-- start group (customize)
```

In questo modo il programma chiederà di pronunciare i comandi del gruppo 1, ovvero APRITI_SESAMO e CHIUDITI_SESAMO.

Quindi, all'interno della funzione `action()`, bisogna inserire un'istruzione quando il comando viene riconosciuto.

```
void action()
{
    switch (group)
{
case GROUP_1:
switch (idx)
{
    case G1_APRITI_SESAMO:
        // write your action code here
        break;
```

```
    case G1_CHIUDITI_SESAMO:
        // write your action code here
        break;
}
```

Al posto del commento `write your action code here` si può scrivere, per esempio, `digitalWrite(LED, 1)` per fare in modo che il pin 8 attivi il LED/relè collegato. Allo stesso modo, bisogna scrivere l'istruzione per spegnere il LED/relè con `digitalWrite(LED, 0)`.

In pratica, il risultato è questo:

```
case G1_APRITI_SESAMO:
    digitalWrite(LED, 1);
    break;
case G1_CHIUDITI_SESAMO:
    digitalWrite(LED, 0);
    break;
```

Ricordarsi di definire la variabile `int LED = 8` e di inserire nel `setup()` l'istruzione `pinMode(LED, OUTPUT)`.

Attenzione! Prima di caricare lo sketch in Arduino, bisogna scollegare EasyVR Commander dalla porta seriale facendo clic sull'icona *Disconnect* e spostare il jumper dello shield su SW.

Lo sketch in funzione

Una volta caricato lo sketch, si può aprire il monitor seriale per vedere cosa succede. La Figura 13.8 illustra la finestra del monitor all'avvio dello sketch. Come si può vedere, viene rilevato il modulo EasyVR e la versione del firmware e subito dopo viene visualizzata la scritta *Say a command in Group 1*.

A questo punto si può pronunciare il comando "apriti Sesamo" e, se il comando viene riconosciuto, sul monitor apparirà *Command 0 = APRITI_SESAMO* e il LED si accenderà.

Notare anche che:

- quando un comando viene riconosciuto viene emesso un breve bip dall'altoparlante;

- se trascorre il tempo preimpostato di time out per pronunciare il comando successivo, apparirà la scritta *Timed out, try again*;
- se il comando non viene riconosciuto o l'audio è disturbato da rumori ambientali, appare un messaggio d'errore.

A ogni modo, anche dopo il time out o il messaggio di errore, riapparirà la scritta *Say a command in Group 1* e, pronunciando questa volta "chiuditi Sesamo", apparirà *Command 1 = CHIUDITI_SESAMO* e il LED si spegnerà.

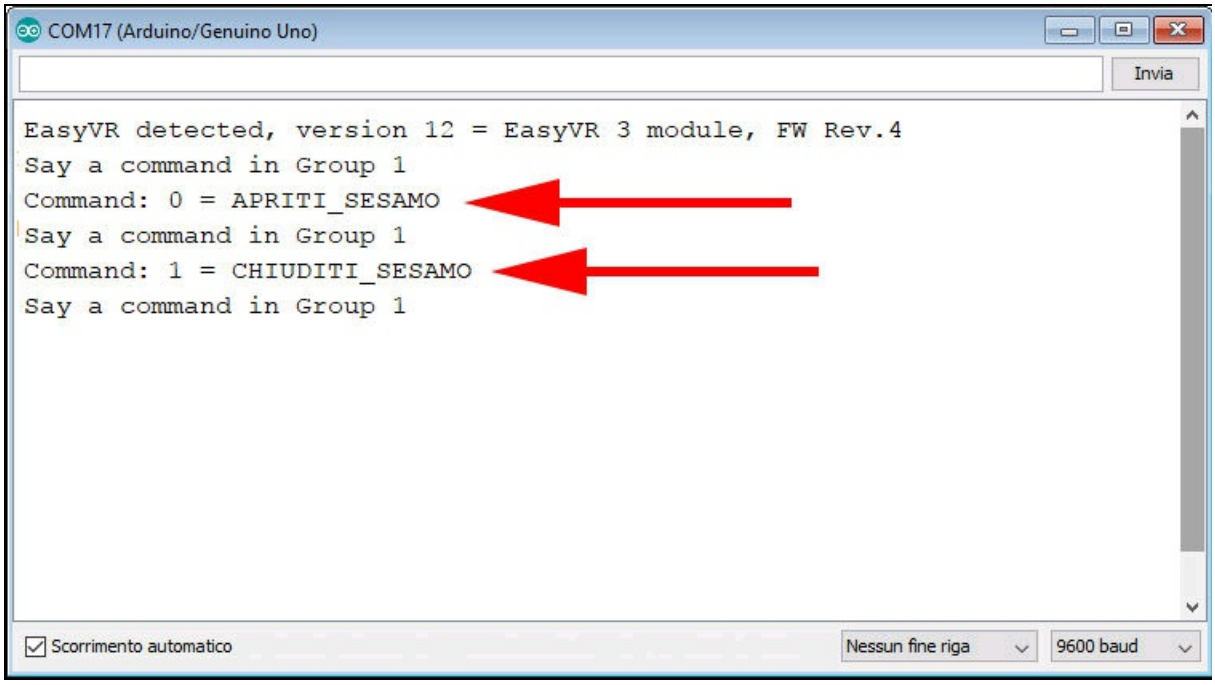


Figura 13.8 Il monitor seriale con i comandi vocali riconosciuti.

Passare da un gruppo all'altro

Come già detto in precedenza, spieghiamo come passare da un gruppo di comandi all'altro. Essendo già memorizzati i tre Wordset di comandi in italiano, è possibile sfruttarli in questo modo.

Per passare alla serie di comandi preimpostati del Wordset 2, è sufficiente aggiungere l'istruzione `group = SET_2` all'interno di un

comando del gruppo 1. Per esempio:

```
case G1_CHIUDITI_SESAMO:
    digitalWrite(LED,0);
    group = SET_2; // passaggio al set di comandi di Wordset 2
    break;
```

In questo modo, dopo aver spento il LED con il comando *CHIUDITI_SESAMO*, si passerà al set di comandi vocali preimpostati del Wordset 2:

- *A sinistra*;
- *A destra*;
- *In alto*;
- *In basso*;
- *Avanti*;
- *Indietro*.

Pronunciando uno di questi comandi verrà stampato sul monitor seriale il nome del comando. Si fa notare che per compilare lo sketch e poter usare questi comandi, sono già stati usati i trattini bassi per unire le parole.

Per tornare al gruppo 1 oppure a un altro gruppo o per passare a un altro Wordset, basta inserire l'istruzione `group = GROUP_1` in uno o più comandi. Per esempio:

```
case S1_FERMO:
    group = GROUP_1;
    break;
```

Per cui, quando si pronuncia il comando "FERMO" del Wordset 1, si potrà tornare ai comandi del gruppo 1. Come si può facilmente intuire, con questa semplice procedura si possono creare catene di comandi vocali molto complesse.

Programmi di utilità

Insieme al pacchetto EasyVR Commander vengono installati anche *Quick Synthesis* e la versione demo di *Quick T2SI*. Il primo serve a

creare nuove tabelle di suoni (SoundTable) per “far parlare” il modulo.

Con Quick T2SI, come vedremo più avanti, si possono sviluppare dizionari *Speaker Independent*. Viene installato anche il pacchetto *FluentChip*, che è una libreria vocale per sviluppatori.

Quick Synthesis

Quick Synthesis (Figura 13.9) è un software di sintesi vocale prodotto da Sensory (http://www.sensory.com) e la versione 5 messa disposizione con il pacchetto EasyVR Commander è gratuita. Con Quick Synthesis 5 è possibile importare dei normali file audio in formato wav, registrati precedentemente con un software di registrazione audio, per poi comprimerli e compilarli in un file proprietario, leggibile da EasyVR Commander.

Una volta importati i file audio in EasyVR Commander, si possono abbinare i comandi alle frasi. Per esempio, nell’esempio demo, ci sono frasi del tipo *Hello*, *Please say your name*, *Access denied* e così via. La voce che pronuncia le frasi in inglese è femminile, ma, se si vuole essere più divertenti, si possono registrare file audio con messaggi “fantasiosi” di qualsiasi tipo, usando la propria voce o quella di un’amica.

Non essendoci un manuale di istruzioni che spieghi come trasferire i file audio di Quick Synthesis al modulo EasyVR, abbiamo creato un nostro piccolo tutorial (si veda il paragrafo “Voice Home Automation”).

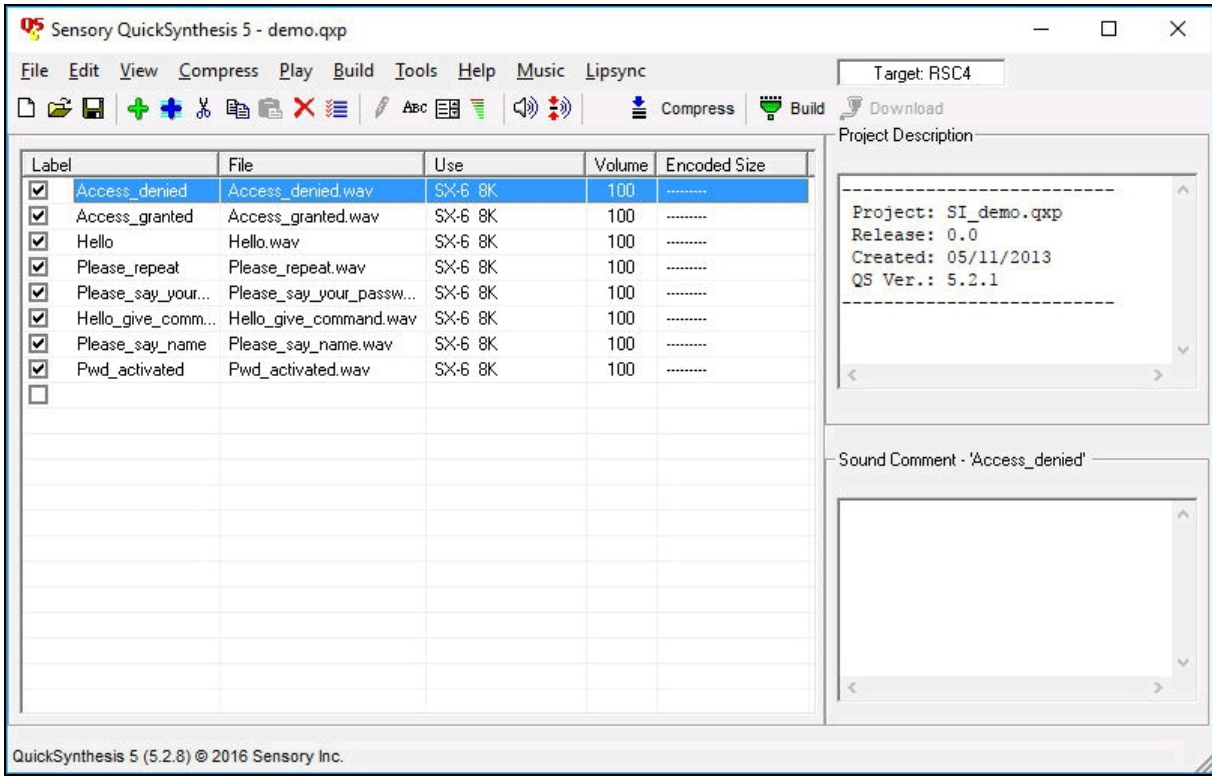


Figura 13.9 La schermata principale di Quick Synthesis.

Come funziona Quick Synthesis

Ci è sembrato utile spiegare brevemente il funzionamento di questo software.

1. Una volta registrati i file audio in formato wav, aggiungerli alla schermata principale con l'icona + (*Add wave files*).
2. Selezionare tutti i file nella lista con il mouse.
3. Fare clic sull'icona *Compress*. Apparirà in una finestra il processo di compressione dei file (Figura 13.10a). Al termine, fare clic su *OK*.
4. Fare clic sull'icona *Build* per aprire una finestra simile a quella rappresentata nella Figura 13.10b e fare clic su *OK*.
5. Al temine salvare il progetto premendo l'icona *Save*.

A questo punto il progetto è pronto per essere importato in EasyVR Commander (si veda il paragrafo “Voice Home Automation”).

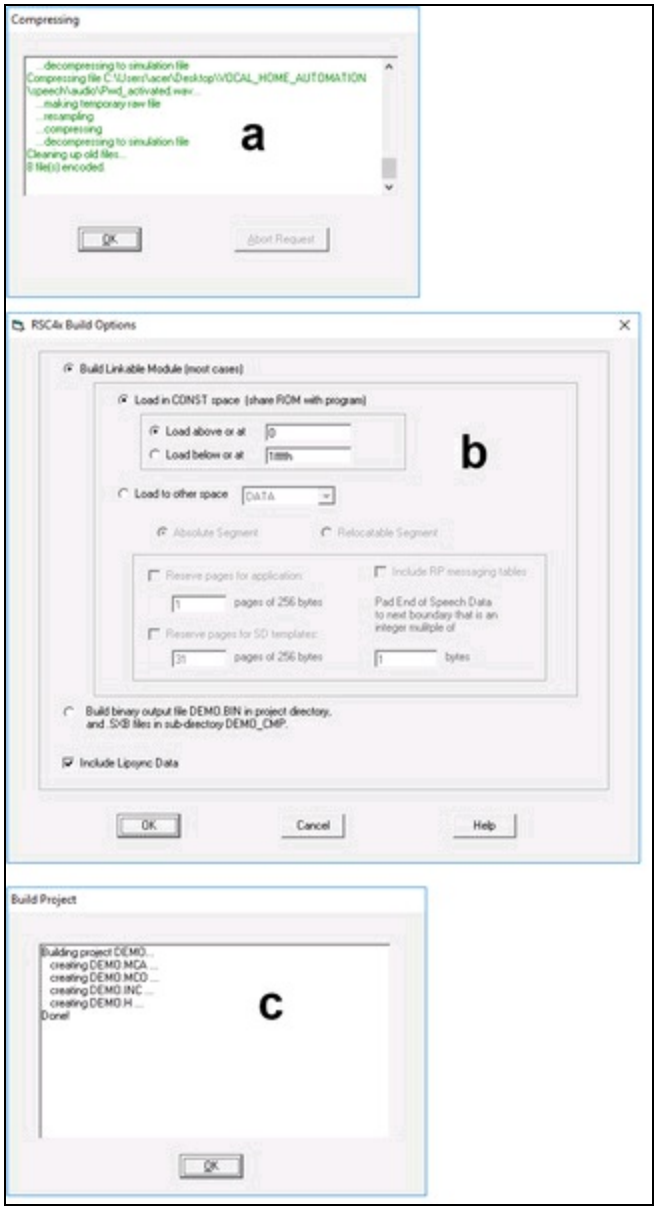


Figura 13.10 La finestra di compressione dei file audio (a). La finestra Build (b). Fine del processo Build (c).

Quick T2SI Lite

Il software Quick T2SI Lite, prodotto sempre da Sensory, consente lo sviluppo di dizionari *Speaker Independent*, ovvero indipendenti dal

parlatore, con interfaccia simile ai software *text to speech* (da testo a voce). Questo consente uno sviluppo rapido di applicazioni per il riconoscimento vocale perché basta scrivere un testo per creare un comando in una delle lingue disponibili:

- inglese US;
- francese;
- tedesco;
- italiano;
- giapponese;
- mandarino;
- spagnolo.

La versione demo deve essere acquistata online e per vederla in funzione bisogna visitare il seguente indirizzo: `http://www.veear.eu/products/quickt2si-lite`. Una volta acquistata la licenza presso un rivenditore o presso il sito italiano Robotech SRL (`http://www.robotechsrl.com`), è possibile accedere alla schermata iniziale di Quick T2SI Lite (Figura 13.11).

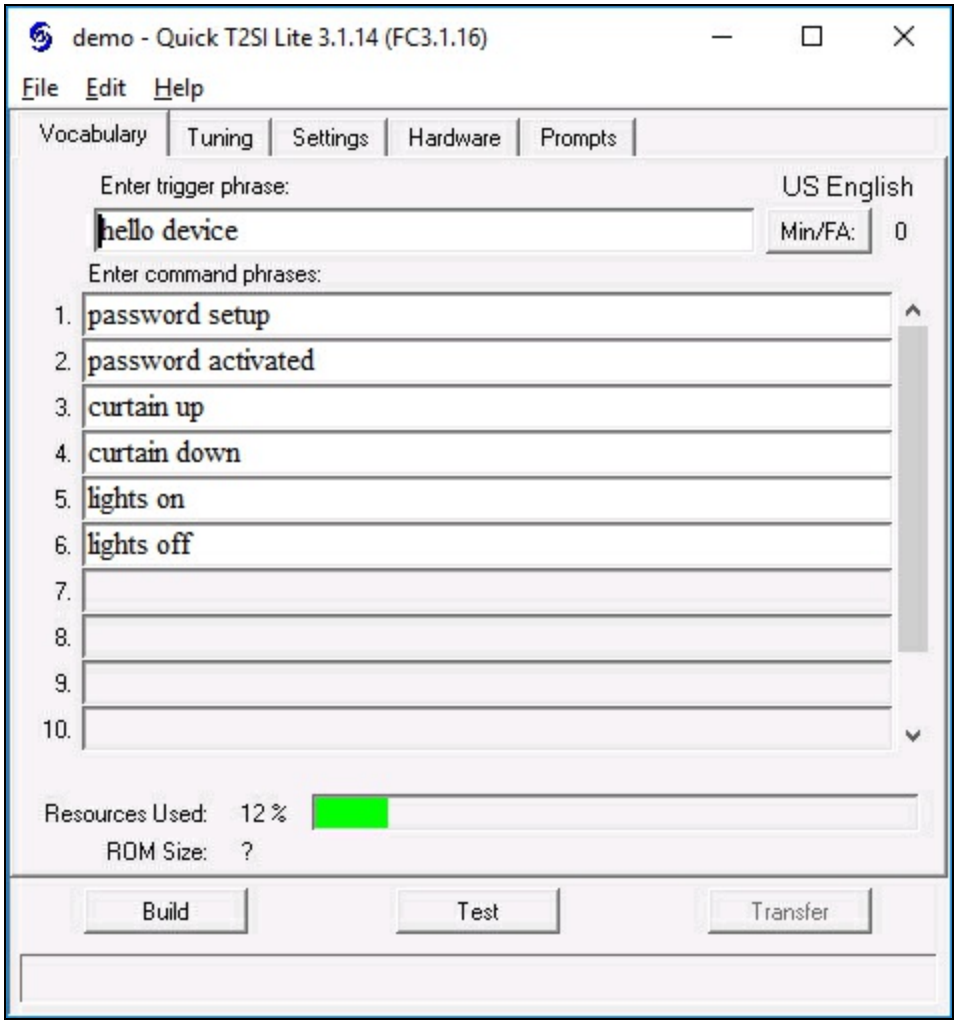


Figura 13.11 La schermata principale di Quick T2SI Lite.

Come funziona Quick T2SI Lite

Spieghiamo brevemente il funzionamento di Quick T2SI Lite, basandoci sul file dell'esempio demo (si veda il paragrafo "Voice Home Automation").

1. Nella finestra principale *Vocabulary*, digitare nella prima casella di testo una frase *Trigger*, ovvero una frase che il sistema riconoscerà per attivare le azioni successive. Per esempio "hello device".

2. Nelle caselle sottostanti inserire le frasi di comando, per esempio, “password setup”, “password activated”, “curtain up”, “ curtain down”, “lights on”, “lights off”.
3. Facendo clic sul pulsante *Test* si può avviare il test audio delle frasi.
4. Pronunciare nel microfono la frase che viene di volta in volta evidenziata in giallo.

Nelle altre schermate di Quick T2SI Lite si può mettere a punto la pronuncia usando l’alfabeto fonetico internazionale (Figura 13.12a), impostare la lingua e molti altri parametri per la frase *Trigger* e le frasi *Command* (Figura 13.12b), impostare l’hardware audio, seriale e RSC (Figura 13.12c) e impostare la risposta audio al prompt di ogni comando, importando file wav già fatti o registrandoli direttamente con il microfono (Figura 13.12d).

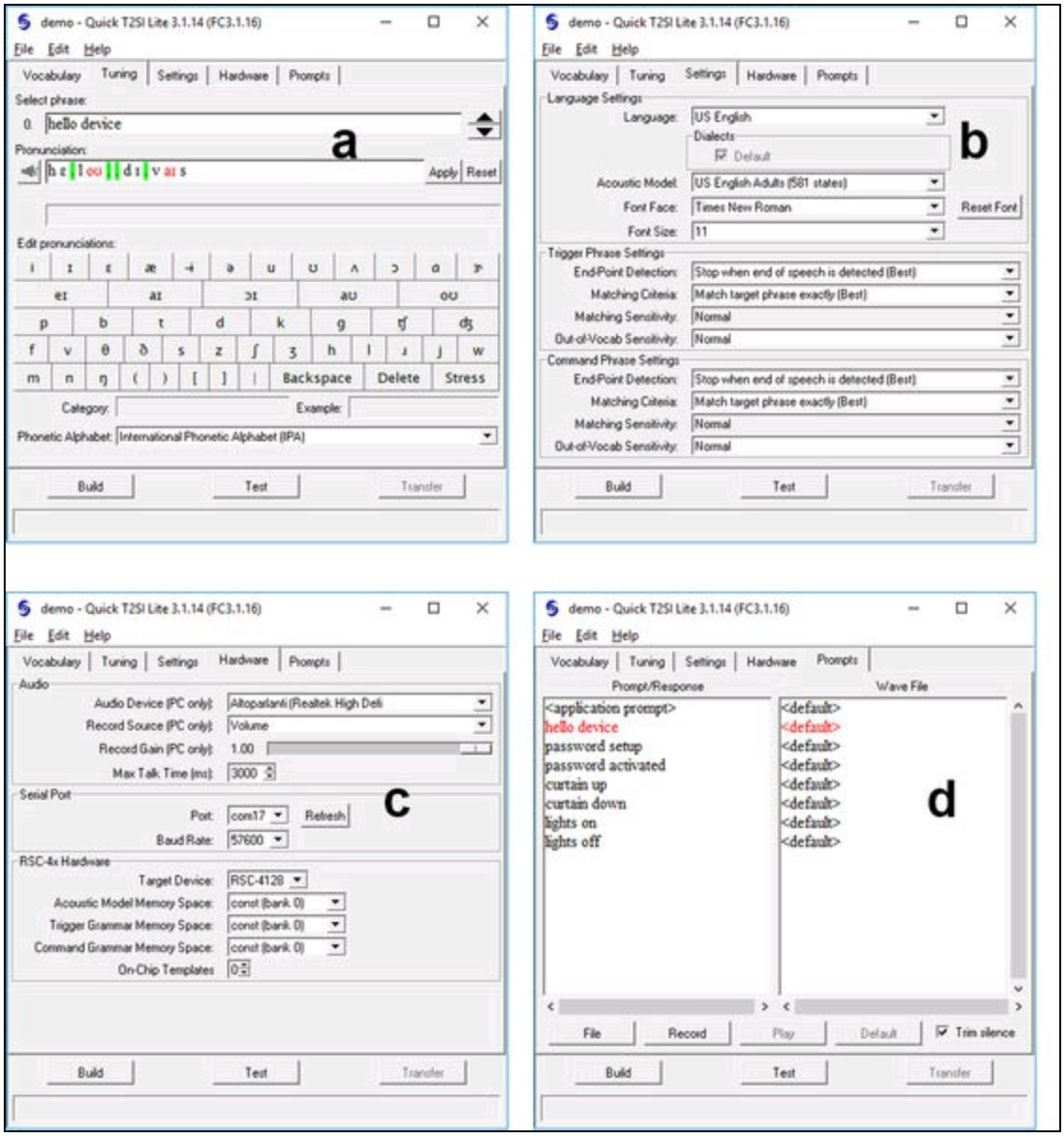


Figura 13.12 Pagina Tuning (a). Pagina Settings (b). Pagina Hardware (c). Pagina Prompt (d).

Voice Home Automation

Dalla sezione download del sito ufficiale `http://www.veear.eu` abbiamo scaricato il file `EasyVR Home Automation`. C'è anche un video, di cui si consiglia la visione, che mostra un semplice sistema di automazione con accesso al sistema tramite password che attiva un LED e un servomotore. Tutto molto bello, ma non si vede come caricare i progetti fatti con Quick Synthesis e con Quick T2SI, ovvero quelli che abbiamo usato nei paragrafi precedenti.

Una volta scompattato il file compresso, nella cartella sono presenti i seguenti file:

- lo sketch `EasyVR_HomeAutom_demo.ino` e il file *`EasyVR_custom_data.h`* da caricare nell'IDE di Arduino;
- una cartella `speech`, contenente il progetto demo fatto con Quick Synthesis;
- una cartella `t2si`, contenente il progetto fatto con Quick T2SI.

Attenzione! Bisogna creare una cartella `EasyVR_HomeAutom_demo` e spostare al suo interno i file `EasyVR_HomeAutom_demo.ino` e `EasyVR_custom_data.h`.

Caricare i progetti Quick Synthesis e Quick T2SI Lite

1. Aprire EasyVR Commander avendo cura di posizionare il jumper dello shield in posizione UP.
2. Fare clic sull'icona *Update Custom Data* come indicato nella Figura 13.13a.
3. Si aprirà una finestra simile a quella rappresentata nella Figura 13.13b.
4. Fare clic sul pulsante *Import.*

5. Individuare il file `demo.rsc` nella cartella `t2si` e importarlo.
6. Fare nuovamente clic sul pulsante *Import.*
7. Individuare il file `demo.qxp` nella cartella `speech` e importarlo.
8. Mettere il segno di spunta su *Slow Transfer*.
9. Fare clic sul pulsante *Download* come indicato nella Figura 13.13c per iniziare il caricamento dei file nella memoria del modulo EasyVR.
10. Inizierà il trasferimento dei file audio fatti con Quick Synthesis e dei comandi fatti con Quick T2SI Lite.

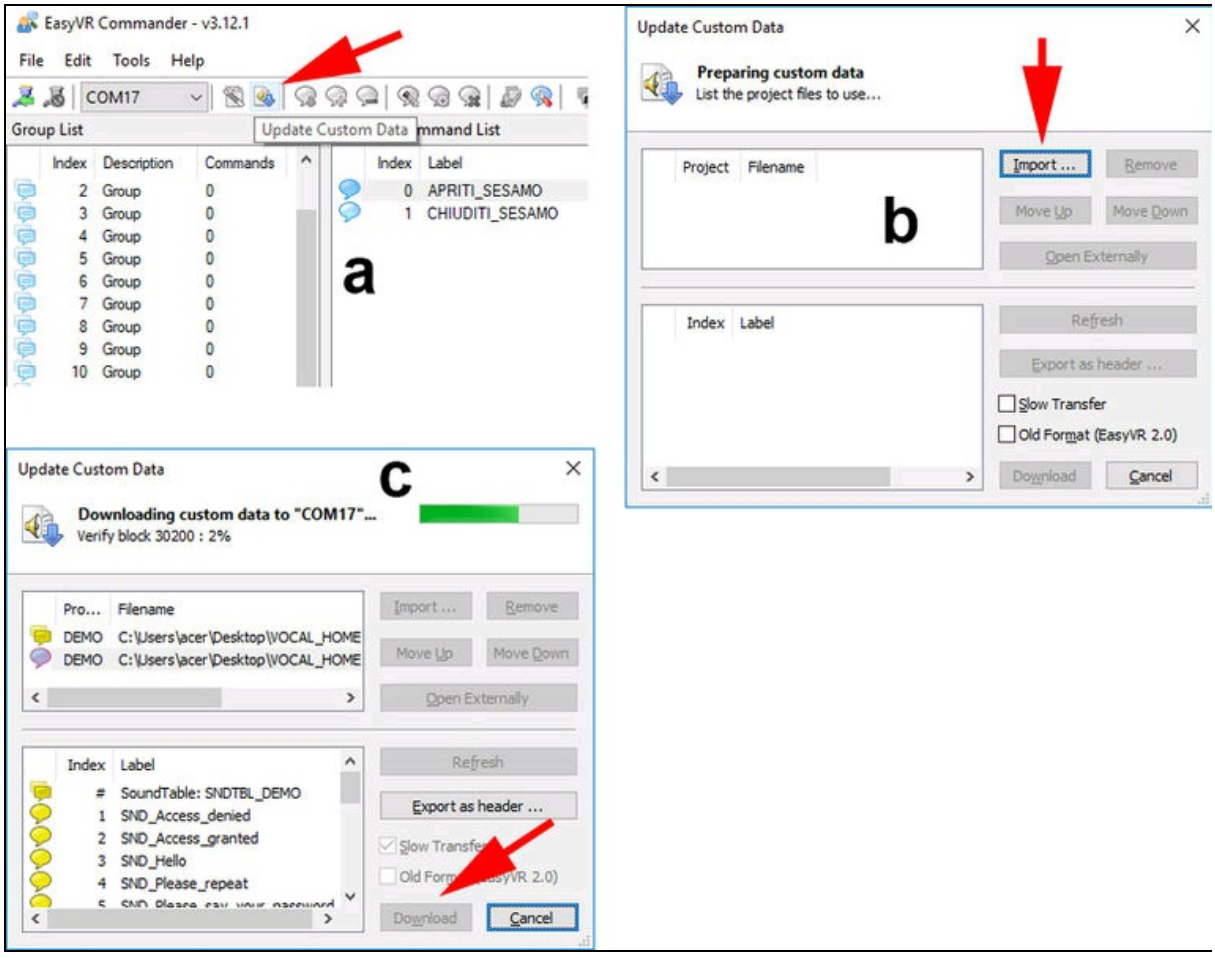


Figura 13.13 Icona Update Custom Data (a). Finestra Update Custom Data (b). Download dei file importati (c).

È tutto pronto, ma prima di caricare lo sketch per Arduino, ricordarsi di spostare il jumper nella posizione SW.

Lo sketch per Arduino

A questo punto, per vedere se tutto funziona come nel video del sito, usando però la nostra voce, non ci resta che caricare lo sketch `EasyVR_HomeAutom_demo.ino` che a sua volta caricherà dalla stessa cartella il file `EasyVR_custom_data.h`.

Si fa notare che il circuito di riferimento è lo stesso di quello usato in precedenza, cioè con un LED/relè collegato al pin 8 e un servomotore al pin 9.

Dopo il caricamento dello sketch, si sentirà dall'altoparlante una bella voce femminile che ci saluta con "*Hello*". Aprendo il monitor seriale, si vedrà anche la scritta *Wait for trigger*, ovvero "In attesa di attivazione" (Figura 13.14a).

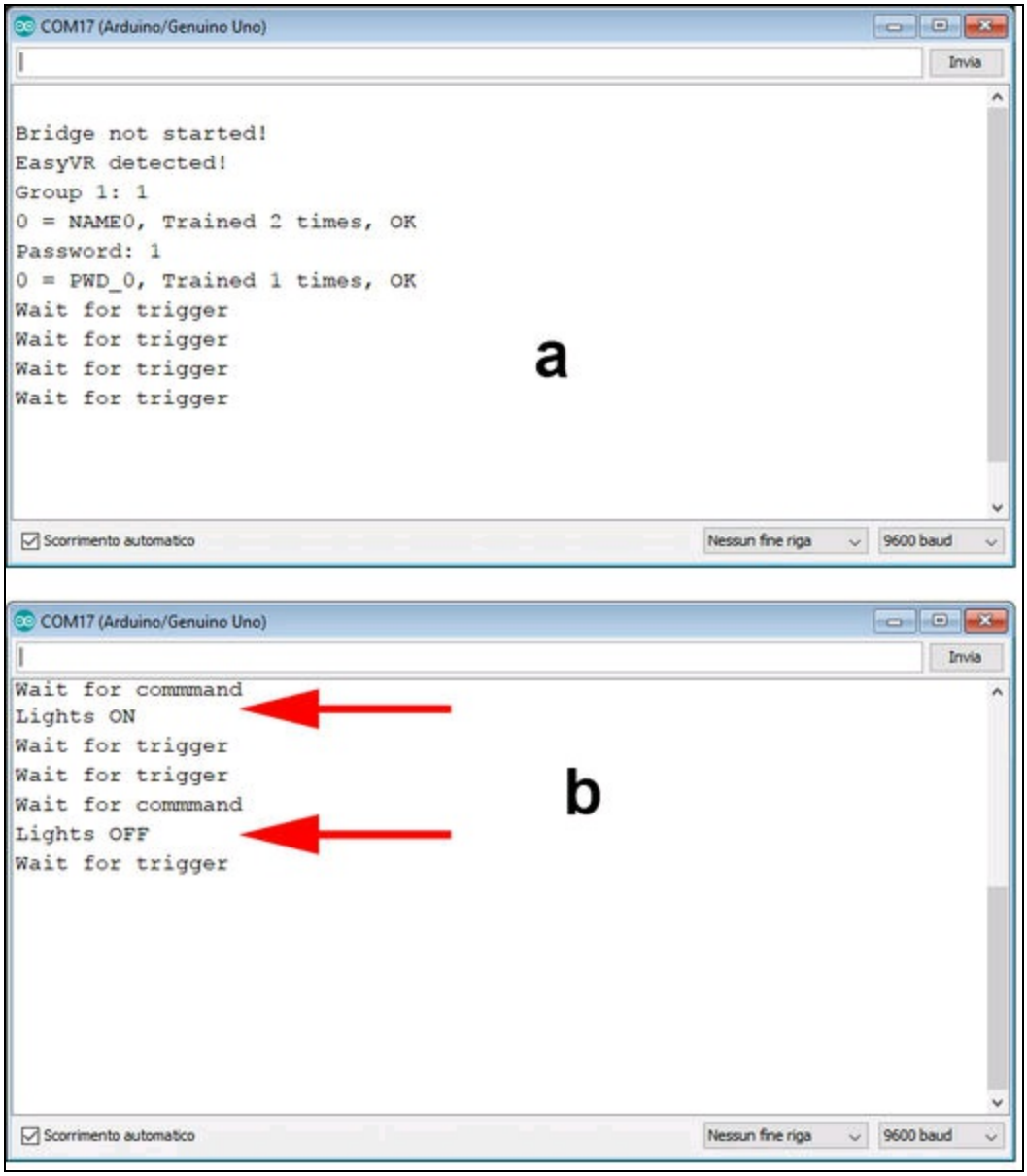


Figura 13.14 Modalità attesa del trigger (a). Modalità comando (b).

Ecco la sequenza di comandi che abbiamo impartito.

1. Nell'esempio demo la frase trigger è "hello device", quindi pronunciando "hello device" si sentirà rispondere "Hello, please say a command". Significa che si è entrati in modalità comando. Apparirà nel monitor seriale la scritta *Wait for command*.
2. Pronunciare un comando, per esempio, "lights on".

3. Se il comando viene riconosciuto, apparirà nel monitor seriale la scritta *Lights ON* e, soprattutto, si vedrà il LED accendersi.
4. Quindi il sistema ritorna in modalità di attesa del trigger.
5. Per inviare un nuovo comando, bisogna prima pronunciare la frase trigger “hello device” e in risposta si sentirà “hello please say a command”. Dopodiché, si può impartire un comando.
6. Per esempio, con la frase curtain up (tenda su) si vedrà il servomotore muoversi verso destra e con la frase curtain down (tenda giù) si vedrà il servomotore tornare nella posizione di partenza.

Impostazione della password

Ci siamo divertiti anche a impostare una password per bloccare l’accesso ad altri.

1. Con il comando “password setup” si entra in modalità di impostazione della password.
2. La voce chiederà di pronunciare un nome utente, per esempio “pippo”. Verrà chiesto di ripetere lo stesso nome per conferma.
3. Quindi chiederà di pronunciare una password, per esempio, “my password”. Verrà chiesto di ripetere la stessa password per conferma.
4. Se non intervengono errori durante questo processo, il sistema entra in modalità di attesa del trigger.
5. Pronunciando il comando “password activated” verrà chiesto di dire il nome (“pippo”) e quindi la password (“my password”). La voce risponderà con “password activated and system locked”. Altrimenti entra in modalità di attesa del trigger.
6. Quando si entra nuovamente in modalità comando, apparirà nel monitor “system locked, ask user name” (Figura 13.15). La voce

chiederà di dire il nome e la password e, se è corretta, si sentirà "access granted" altrimenti si sentirà "access denied".

Tutto questo ci rende molto soddisfatti. Soprattutto il fatto di poter creare un dizionario *speaker independent* che consente di usare comandi personalizzati in qualsiasi lingua. Tant'è che per chiudere in bellezza abbiamo creato un set di comandi in italiano con Quick T2SI Lite (Figura 13.16a) e un set di messaggi in italiano con Quick Synthesis (Figura 13.16b). Non è cool?

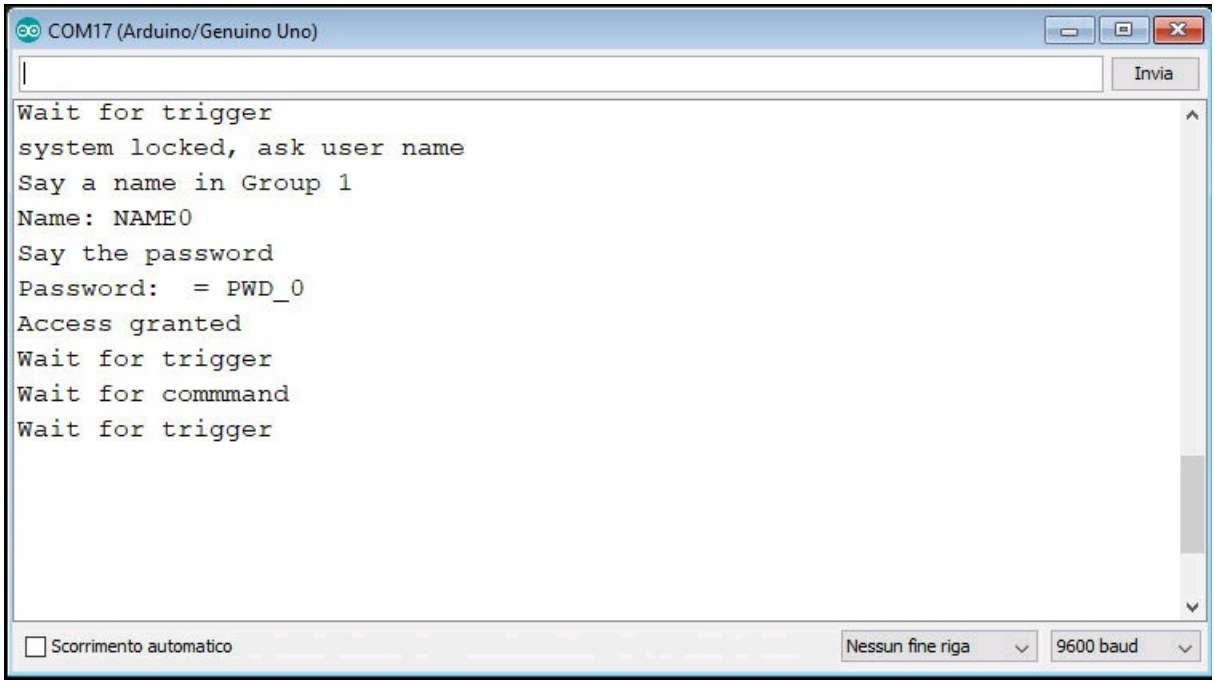


Figura 13.15 Accesso bloccato con password.

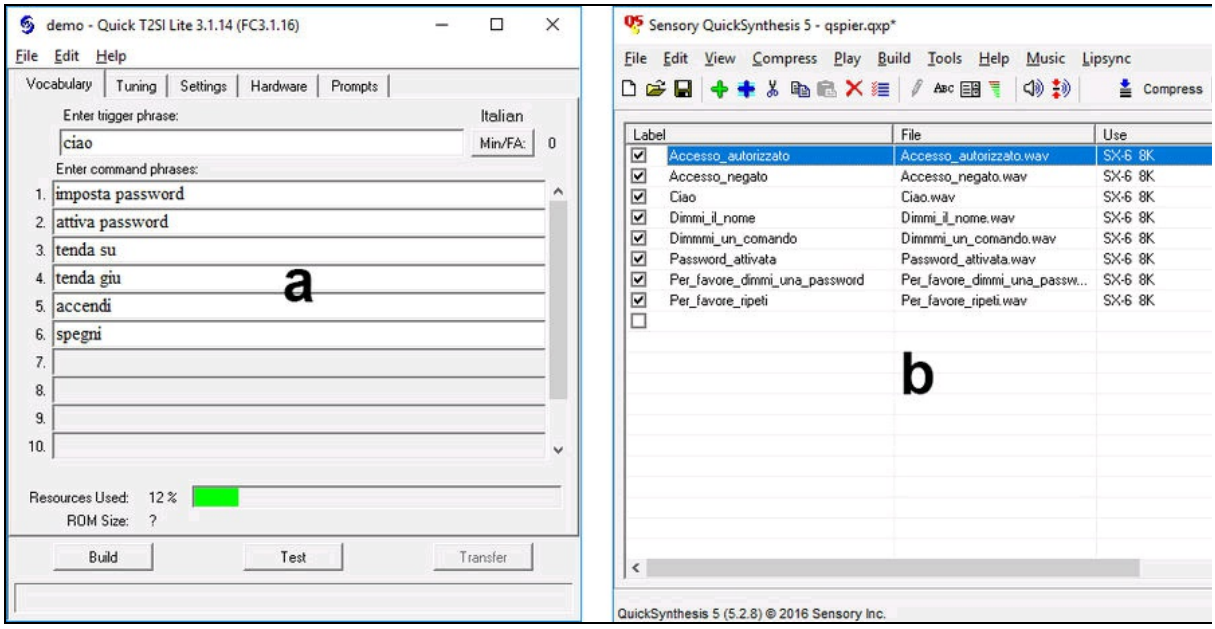


Figura 13.16 Set di comandi in italiano (a). Set di messaggi in italiano (b).

Specchio magico

Descrizione

Specchio, specchio delle mie brame, chi è… il maker più bravo del reame?

Anche se siamo molto distanti dalla fantasia dei fratelli Grimm, il progetto di specchio magico che proponiamo in questo capitolo fa vedere, oltre alla nostra immagine riflessa, la data e l'orario in tempo reale, alcuni messaggi di saluto, le informazioni meteo, le notizie del giorno e, virtualmente, qualsiasi altra cosa. Volendo, lo si potrebbe anche far parlare, ma lasciamo questo privilegio alla favola di Biancaneve.

Si possono trovare online diversi progetti di *Magic Mirror*, più o meno complessi, ma abbiamo preferito pensare a una soluzione molto semplice e personalizzabile a piacere con pochissimo sforzo.

CODICE DI ESEMPIO

Tutti i file del progetto "Specchio magico" sono disponibili nelle risorse del libro presso il sito dell'autore all'indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× Raspberry Pi Zero W (o un qualsiasi altro modello);
- 1× alimentatore per Raspberry Pi;
- 1× monitor o un apparecchio TV HDMI;
- 1× mini cavo HDMI;
- 1× specchio spia.

Specchio spia

Il cuore del progetto è un comune specchio spia o semiriflettente (Figura 14.1a). Si può trovare anche online, ma la soluzione migliore è di andare presso un vetraio specializzato in specchi e farne tagliare uno su misura.

Per il nostro progetto abbiamo utilizzato uno specchio spia fatto tagliare su misura a 410×230 mm, ovvero le misure della cornice interna di uno schermo TV da 19 pollici (Figura 14.1b) o, meglio, le misure del nostro monitor. Infatti, si consiglia di misurare con precisione le dimensioni dello schermo che si intende usare, lasciando circa un millimetro di gioco per non avere problemi di inserimento. Anche il vetraio potrebbe non essere precisissimo nel taglio.

Dopo aver visionato online diversi metodi per posizionare uno specchio davanti a un monitor, abbiamo preferito inventarne uno. Senza usare cornici di legno o altro materiale per fissare lo specchio, abbiamo preferito incollare con un due strisce sottili di biadesivo lo specchio direttamente sullo schermo del monitor. In questo modo lo specchio aderisce perfettamente alla superficie e non crea riflessioni sui bordi dei testi piccoli o errori di parallasse.

Se non si vuole vedere la marca dell'apparecchio TV o del monitor, si può pensare di verniciare a spruzzo la cornice esterna con un colore a piacere.

L'orientamento orizzontale o verticale dipende solo dai gusti ed è ininfluente ai fini del progetto, perché la rotazione delle informazioni visualizzate verrà effettuata via software.

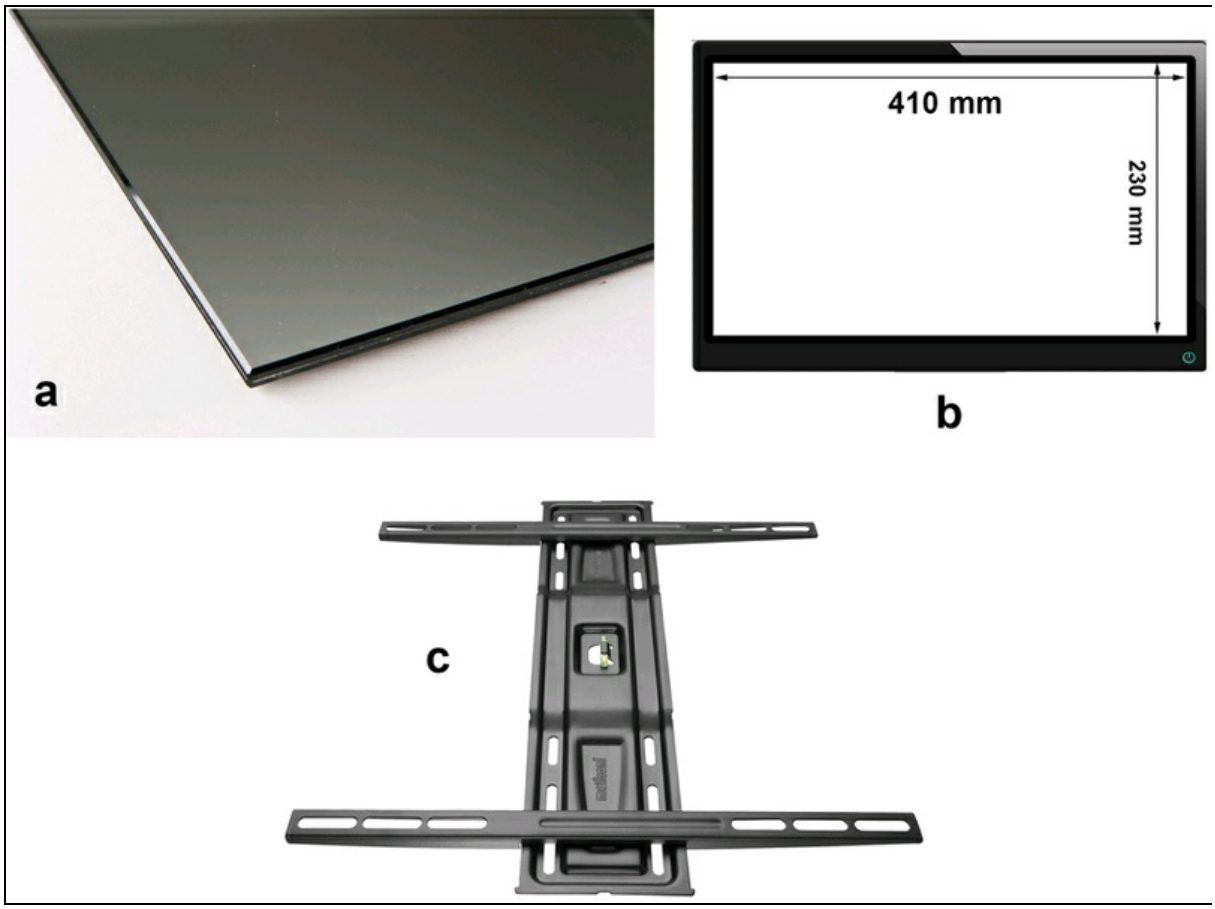


Figura 14.1 Un comune specchio spia (a). Le dimensioni del nostro specchio spia (b). Staffa per fissaggio a muro (c).

Il monitor

Va bene qualsiasi monitor per computer o un apparecchio TV con schermo a cristalli liquidi o LED, purché sia dotato di ingresso HDMI per poterlo collegare a un Raspberry Pi.

Per la dimensione dello schermo meglio non esagerare, per non spendere troppo con lo specchio. Siccome il funzionamento dello specchio spia si basa sul principio della semi riflessione ottica, lo schermo TV dovrà essere meno luminoso rispetto all'ambiente in cui viene osservato. Per questo motivo, è meglio regolare il contrasto e la luminosità video in modo che il nero sia molto… nero.

Per il fissaggio a muro, orizzontale o verticale, si può pensare a una staffa di tipo fisso, che si trova normalmente in commercio (Figura 14.1c). L'alimentazione del monitor dovrebbe essere nascosta dietro all'apparecchio per un effetto più "misterioso".

Raspberry Pi Zero W

Anche se può andare bene qualsiasi modello, si consiglia vivamente di usare il minuscolo Raspberry Pi Zero W che è già dotato di Wi-Fi e può essere alloggiato dietro a un monitor senza problemi, assieme al suo alimentatore e a un mini cavo HDMI da connettere al monitor. Notare l'adattatore HDMI per la mini presa di Raspberry Pi Zero W (Figura 14.2). Per fissarlo è sufficiente del normale biadesivo.

La connessione Wi-Fi, oltre al collegamento Internet per l'orario e le altre informazioni, consentirà di accedere alla configurazione del sistema da una postazione remota in SSH, senza dover smontare una vite dal muro.

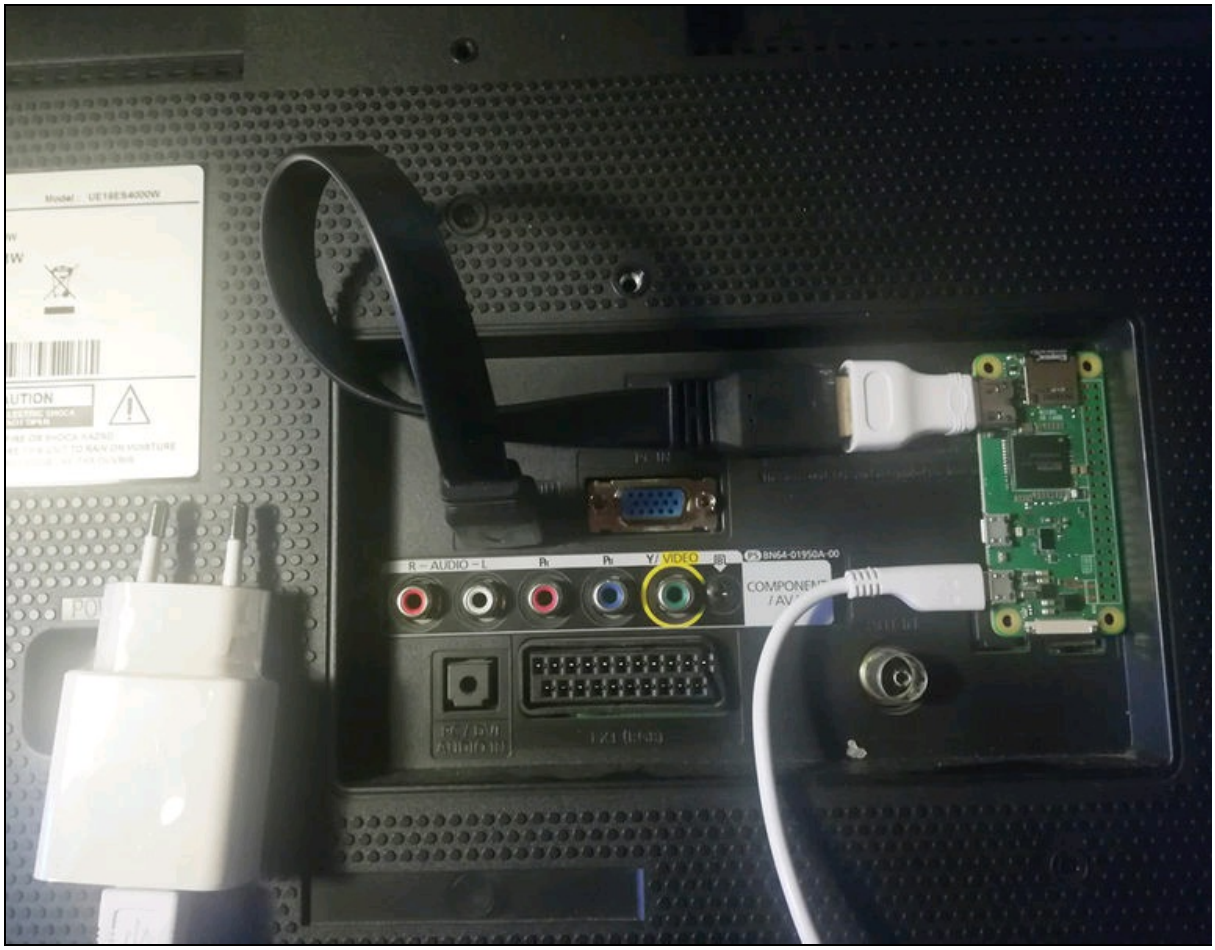


Figura 14.2 Raspberry Pi Zero W collocato dietro al monitor.

Home page

L’idea di base è quella di installare un server Apache2 nel sistema operativo RaspBian, in modo che possa ospitare una home page con contenuto attivo in PHP. Una possibile home page potrebbe avere un aspetto simile alla Figura 14.3, con uno sfondo completamente nero e i testi bianchi.

Si fa presente che l’orientamento della pagina è orizzontale, ma come vedremo, si può mettere in verticale ruotando anche il monitor.

Sulla sinistra, l’area vuota lascia spazio all’immagine di chi si specchia, l’area centrale è occupata da un orologio digitale o analogico

(a scelta) e un’area a destra con le notizie del giorno, le previsioni meteo o altro.

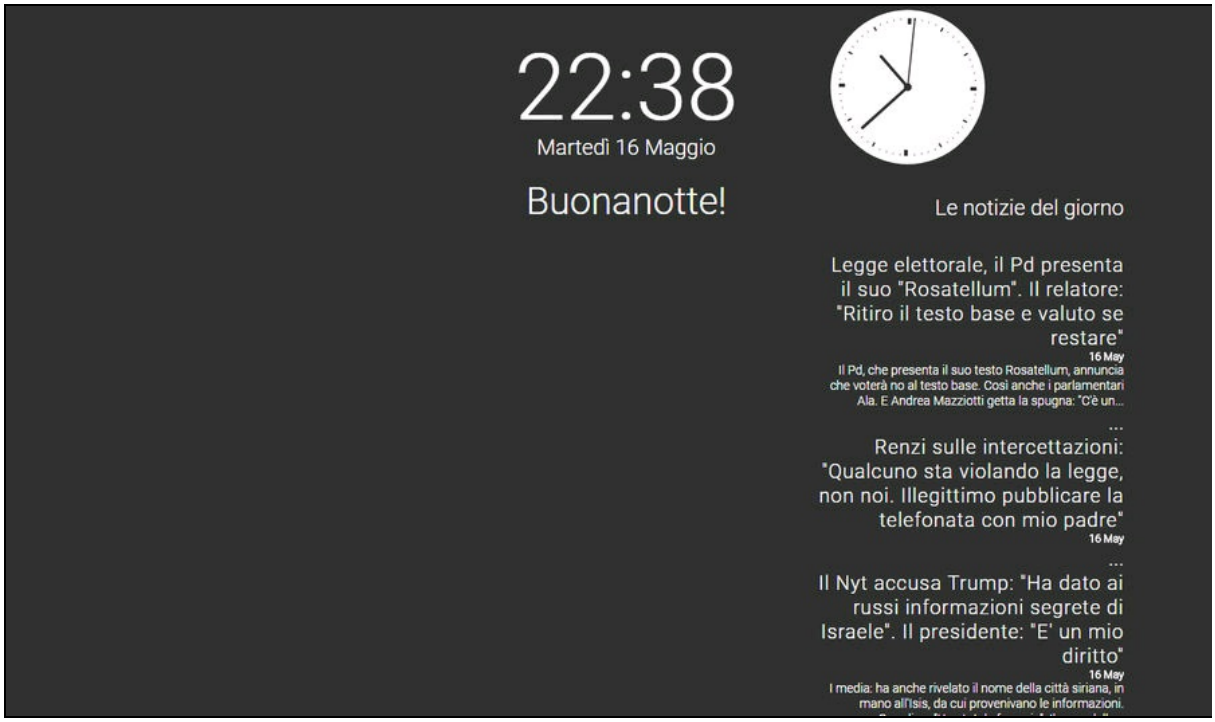


Figura 14.3 La home page del server.

Installazione del server Apache2

L’installazione si può fare direttamente da Raspberry Pi, oppure da una connessione SSH con un computer remoto, così come le successive modifiche alla home page quando lo specchio sarà fissato al muro.

Dal terminale di RaspBian eseguire i seguenti comandi per installare il server Apache2 e PHP:

```
sudo apt-get update
sudo apt-get install apache2 apache2-doc apache2-utils
sudo apt-get install libapache2-mod-php5 php5 php-pear php5-xcache
```

Dopo l’installazione, è possibile verificare il funzionamento del server digitando sul browser di RaspBian `http://localhost` oppure, su un browser collegato alla stessa rete l’IP assegnato dal router di casa, per esempio, `192.168.1.76`. Apparirà l’home page di Apache2 simile alla Figura 14.4.

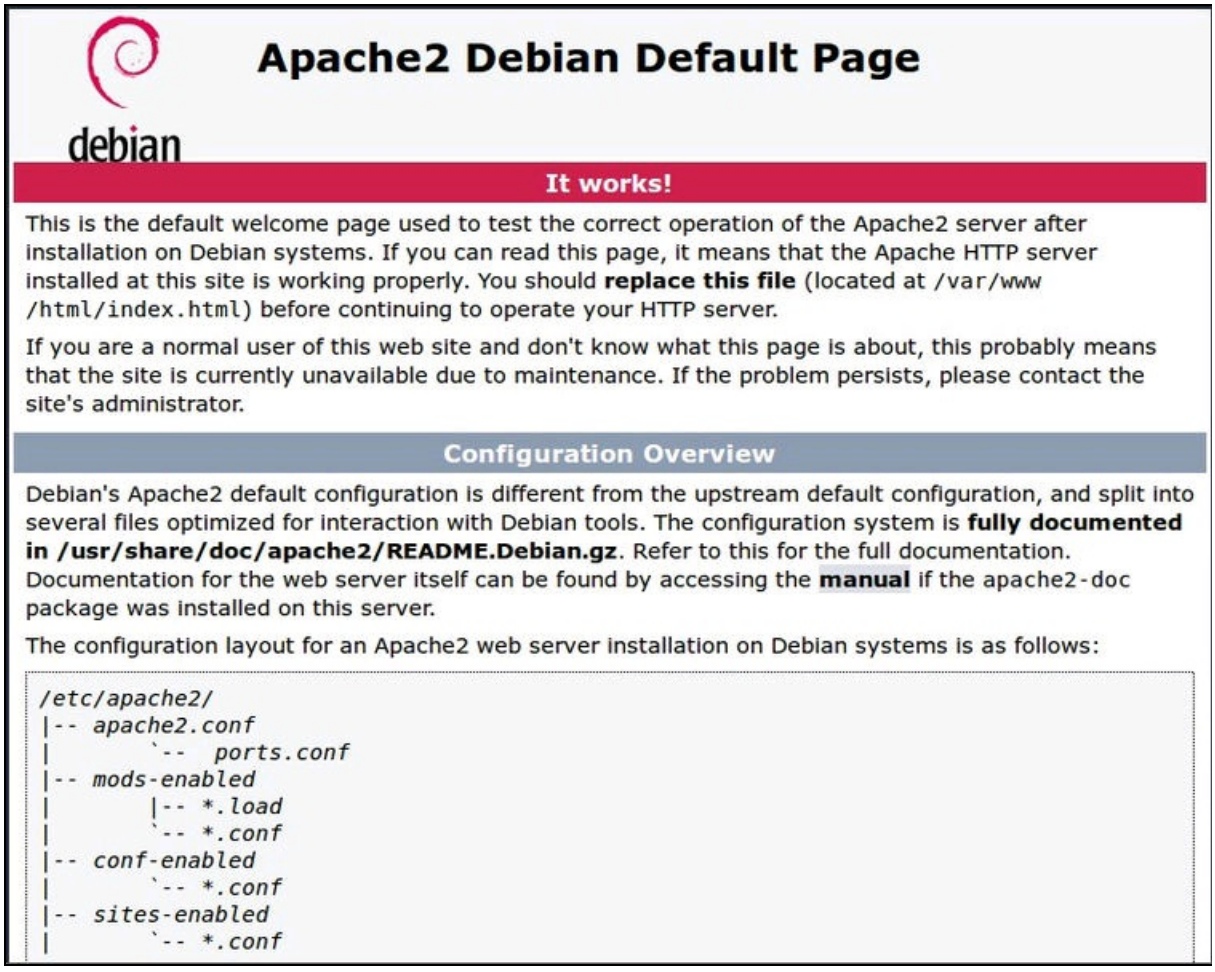


Figura 14.4 La home page di Apache 2.

Questa pagina corrisponde al documento `index.html`, il cui percorso è il seguente:

```
/var/www/html/index.html
```

Per poter modificare il contenuto della cartella principale del server, che è di sola lettura, è necessario cambiare il proprietario con il seguente comando:

```
sudo chown pi: /var/www/html
```

Dopodiché si può entrare nella cartella `/var/www/html` ed eliminare il file `index.html`. Per la nostra home page abbiamo creato due documenti:

`index.php` e `style.css`. questi andranno copiati all'interno della cartella `/var/www/html`.

Per poterli modificare, bisogna cambiare il proprietario con il comando seguente:

```
sudo chown pi: /var/www/html/index.php && sudo chown pi: /var/www/html/style.css.
```

Per la modifica dei documenti si può usare un normale editor di testo oppure Geany, un ottimo editor già installato in RaspBian.

Il documento index.php

Il documento `index.php` andrà copiato nella suddetta cartella è sarà la nuova home page del server Apache2. Per modificare il contenuto di questo documento, si prega di fare riferimento all'esempio `index.php` contenuto nelle risorse del libro. Qui di seguito vengono spiegate solo le funzioni principali per poterle adattare alle proprie esigenze.

Il contenuto del documento e i linguaggi adottati sono di vario tipo:

- PHP;
- JavaScript;
- HTML;
- CSS (file `style.css`).

NOTA

L'orientamento della home page del nostro esempio è orizzontale. Chi vuole orientarla in senso verticale dovrà modificare il codice del file CSS.

Codice PHP

Il codice PHP serve a creare una scritta amichevole in base all'ora del giorno. Ovviamente si possono creare messaggi di fantasia diversi, a seconda del proprio umore. Oltre al numero e al tipo di messaggi, si può cambiare anche l'intervallo della visualizzazione. Come si può vedere, il

codice è molto versatile. Si fa notare l'uso del carattere escape "\" per inserire gli apostrofi, come nella frase "Sogni d\'oro". L'istruzione `echo` di PHP permette di mandare all'output qualsiasi stringa.

```
<?php
    $now = date('H');
    if (($now >= 06) and ($now < 10)) echo 'Il mattino ha l\'oro in bocca!';
    else if (($now >= 10) and ($now < 12)) echo 'Buona giornata!';
    else if (($now >= 12) and ($now < 14)) echo 'Ora di pranzo!';
    else if (($now >= 14) and ($now < 17)) echo 'Ora di colazione!';
    else if (($now >= 17) and ($now < 20)) echo 'Hai pensato alla cena?';
    else if (($now >= 20) and ($now < 22)) echo 'Buona serata!';
    else if (($now >= 22) and ($now < 23)) echo 'Buonanotte!';
    else if (($now >= 23) and ($now < 24)) echo 'Sogni d\'oro...';
?>
```

Il linguaggio PHP serve anche alla decodifica dei cosiddetti feed RSS (*RDF Site Summary*), ovvero delle fonti di notizie di qualsiasi tipo, provenienti dai siti che forniscono tale servizio.

Il codice PHP che svolge questo compito crea un DOM, cioè un *Document Object Model*. Come si può vedere, il DOM viene implementato nell'oggetto *rss* che ne eredita i metodi.

Con l'istruzione `$rss->load` si può caricare la pagina RSS di riferimento. Nel nostro esempio è la home page del sito de La Repubblica. Basta effettuare una ricerca online per trovare feed RSS praticamente da ogni sito di notizie e/o servizi meteo.

```
<?php
    $rss = new DOMDocument();
    $rss->load('http://www.repubblica.it/rss/homepage/rss2.0.xml');
```

Viene quindi creata un array con l'istruzione `$feed = array()` nella quale vengono memorizzati alcuni elementi del feed, come `title`, `desc` e `date`, corrispondenti a titolo, descrizione e data.

```
$feed = array();
foreach ($rss->getElementsByTagName('item') as $node)
{
    $item = array (
    'title' => $node->getElementsByTagName('title')->item(0)->nodeValue,
    'desc' => $node->getElementsByTagName('description')->item(0)->nodeValue,
    'date' => $node->getElementsByTagName('pubDate')->item(0)->nodeValue,
    array_push($feed, $item);
}
```

Viene impostato il limite a tre feed, ma si può ridurre o aumentare il numero di feed in base alle dimensioni e all'orientamento del monitor. Con l'istruzione `echo` si stampano i tag HTML del titolo, della descrizione e della data. Anche in questo caso, se si vogliono visualizzare solo i titoli e la data, basta togliere la descrizione e cambiare l'istruzione finale in `echo '<p></p><h2>...</h2>'`. Verranno stampati solo i tre puntini.

```
    $limit = 3;
    for($x=0;$x<$limit;$x++) {
        $title = str_replace(' & ', ' &amp; ', $feed[$x]['title']);
        $description = $feed[$x]['desc'];
        $date = date('j F', strtotime($feed[$x]['date']));
        echo '<h2 class="smaller">'.$title.'</h2>';
        echo '<p class="date">'.$date.'</p>';
        echo '<p>'.strip_tags($description, '<p><b>').'</p><h2>...</h2>';
    }
?>
```

Codice JavaScript

Il codice JavaScript serve alla creazione dell'orologio digitale e di quello analogico. Nella funzione dell'orologio digitale ci sono le istruzioni per convertire il nome del giorno della settimana e il nome del mese in italiano.

Si fa notare che, nonostante la presenza del tag HTML `charset=utf-8`, abbiamo usato il codice `&igrave;` per la lettera accentata "ì", onde evitare che il browser visualizzi il tipico carattere con il punto di domanda.

Il rimanente listato è talmente semplice che non ha bisogno di ulteriori spiegazioni.

```
<script language="JavaScript">
    setInterval(function() {
        var currentTime = new Date ( );
        var currentHours = currentTime.getHours ( );
        var currentMinutes = currentTime.getMinutes ( );
        var currentMinuteswithzero = currentMinutes > 9 ?
            currentMinutes : '0' + currentMinutes;
        var currentDate = currentTime.getDate ( );
        var weekday = new Array(7);
        weekday[0] = "Domenica";
        weekday[1] = "Luned" + "&igrave;";
        weekday[2] = "Marted" + "&igrave;";
```

```
        weekday[3] = "Mercoled" + "&igrave;";
        weekday[4] = "Gioved" + "&igrave;";
        weekday[5] = "Venerd" + "&igrave;";
        weekday[6] = "Sabato";
        var currentDay = week-day[currentTime.getDay()];
        var actualmonth = new Array(12);
        actualmonth[0] = "Gennaio";
        actualmonth[1] = "Febbraio";
        actualmonth[2] = "Marzo";
        actualmonth[3] = "Aprile";
        actualmonth[4] = "Maggio";
        actualmonth[5] = "Giugno";
        actualmonth[6] = "Luglio";
        actualmonth[7] = "Agosto";
        actualmonth[8] = "Settembre";
        actualmonth[9] = "Ottobre";
        actualmonth[10] = "Novembre";
        actualmonth[11] = "Dicembre";
        var currentMonth = actual-month[currentTime.getMonth ()];
        var currentTimeString = "<h1>" + currentHours + ":" +
currentMinuteswithzero +
            "</h1><h2>" + currentDay + " " + currentDate + " " + currentMonth + "
</h2>";
        document.getElementById("clock").innerHTML = currentTimeString;
    }, 1000);
</script>
```

Il listato per l'orologio analogico è un po' più lungo, pertanto riportiamo solo la funzione `drawFace`, i cui colori possono essere modificati. I colori usati sono ovviamente il nero (`#000`) e il bianco (`'white'`) ed è sconsigliato l'uso di altri colori, anche se è possibile. Per il gradiente di riempimento del bordo, abbiamo usato un intervallo da nero a nero. Volendo, si potrebbe preferire il gradiente da nero a bianco.

```
function drawFace(ctx, radius) {
    ctx.beginPath();
    ctx.arc(0, 0, radius, 0, 2 * Math.PI);
    ctx.fillStyle = 'white';
    ctx.fill();
    var gradient;
    gradient = ctx.createRadialGradient(0, 0, radius * 0.95, 0, 0, radius * 1.05);
    gradient.addColorStop(0, '#000');
    gradient.addColorStop(1, 'black');
    ctx.strokeStyle = gradient;
    ctx.lineWidth = radius * 0.1;
    ctx.stroke();
    ctx.beginPath();
    ctx.arc(0, 0, radius * 0.05, 0, 2 * Math.PI);
    ctx.fillStyle = '#000';
    ctx.fill();
```

Codice HTML

I tag HTML sono quelli che vengono usati per impostare il layout della pagina. Come si può vedere c'è anche il titolo della pagina, anche se non verrà visualizzato nello specchio. Il `refresh` della pagina è impostato a `900` secondi, cioè ogni 15 minuti, ma si può cambiare a piacere. Il tag `link_rel` carica il foglio stile `style.css`, di cui parliamo più avanti.

Il font usato è della famiglia "Roboto", uno dei tanti che Google Fonts mette a disposizione nel sito `https://fonts.google.com` e che è anche il font di default dell'interfaccia grafica di RaspBian.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Magic Mirror</title>
<meta name="description" content="Magic Mirror">
<meta http-equiv="refresh" content="900" />
<link rel="stylesheet" href="style.css">
<link href='http://fonts.googleapis.com/css?family=Roboto' rel='stylesheet'
type='text/css'>
```

Nel `body` della pagina sono disposti tutti i vari `div` cioè le sezioni in cui mettere l'orologio digitale e/o quello analogico, il messaggio di saluto e le notizie laterali. I `div` sono stati nominati in base alla loro funzione nel layout. La loro posizione e il contenuto vengono stabiliti nel file `style.css`.

```
<div id="main">
<div id="left">
<div id="clock">
<div id="right">
```

Il `<div>` con l'ID `main` contiene tutti gli altri `<div>`. Il `<div>` con l'ID `left` contiene il `<div>` con l'ID `<clock>`, cioè l'orologio digitale. Se non si vuole l'orologio digitale basta togliere la riga `<div id="clock"></div>`. Il risultato è visibile nella Figura 14.5. Il `<div>` con l'ID `right` contiene il codice PHP con i feed rss.

Per l'orientamento verticale bisogna togliere il tag `<div id="main">`.

L'orologio analogico viene inserito in un `canvas` le cui dimensioni sono queste:

```
<canvas id="canvas" width="200" height="200">
</canvas>
```

Per ingrandire o ridurre l'orologio analogico, è sufficiente cambiare i parametri `width` e `height`.

Figura 14.5 La home page con l'orologio analogico.

Il documento style.css

Come ogni home page che si rispetti, anche la nostra ha un foglio stile che viene caricato con il tag `link_rel`:

```
<link rel="stylesheet" href="style.css">
```

I parametri più importanti del foglio stile sono relativi al colore di sfondo che è totalmente nero (`#000`) e alla dimensione dei `<div>` che andranno posizionati in modo diverso in base all'orientamento e alle dimensioni del monitor oppure secondo il proprio gusto. I parametri chiave da modificare sono `width` e `height`.

È da notare che nel `body` le dimensioni sono per il nostro schermo in orizzontale e sono state impostate in modo da non far apparire nel browser la barra di scorrimento inferiore e laterale, quindi bisogna adattarle a seconda dei casi. Anche lo spostamento da sinistra (`left:400px`) del `main` va impostato in base alle esigenze.

```
body {
    width: 1000px;
    height: 600px;
    background: #000; // colore nero dello sfondo
    color: #fff;
}
// <div> principale
#main {
    width: 750px;
    height: 500px;
    position: absolute;
    left: 400px;
    margin: 0 auto;
}
// <div> del messaggio di saluto
#left {
    width: 325px;
    margin: 50px 50px 0 0;
    height: 400px;
    float: left;
    text-align: center;
}
// <div> delle notizie
#right {
    width: 340px;
    height: 400px;
    margin: 15px 35px 0 0;
    float: left;
    text-align: right;
}
```

Il rimanente codice CSS è relativo ai tag HTML usati nel documento per i titoli (`<h1>`, `<h2>`, `<h3>`) e per il paragrafo (`<p>`). I parametri definiscono la famiglia del font “Roboto” e la dimensione del carattere. Anche in questo caso, si può modificarli a piacere.

```
h1 {
    font-family: 'Roboto', sans-serif;
    font-size: 102px;
    font-weight: 300;
    line-height: 102px; }
h2 {
    font-family: 'Roboto', sans-serif;
    font-size: 24px;
    font-weight: 300; }
h2.smaller {
    font-size: 22px;
```

```
    letter-spacing: 1px; }
h3 {
    font-family: 'Roboto', sans-serif;
    font-size: 42px;
    font-weight: 300;
    padding: 20px 0 0 0; }
p {
    font-family: 'Roboto', sans-serif;
    font-size: 14px;
    font-weight: 300; }
```

Rotazione del display

Per l’orientamento verticale del monitor bisogna ruotare di 90 gradi il display. Ecco come fare. Dal terminale digitare i seguenti comandi:

```
cd /boot
sudo leafpad config.txt
```

Nel documento `config.txt`, individuare la riga…

```
#hdmi_force_hotplug=1
```

… e togliere il commento `#`. Salvare il documento e riavviare.

Alla fine del documento aggiungere:

```
display_rotate=1
```

Lo schermo sarà ruotato di 90 gradi a destra.

Per ruotare di 90 gradi a sinistra, scrivere:

```
display_rotate=3
```

Salvaschermo e Chromium

Per eliminare il salvaschermo nero è necessario agire in questo modo.

Dal terminale digitare il seguente comando:

```
sudo leafpad /etc/kbd/config
```

Individuare nel testo le seguenti righe e cambiare il valore a 0 (zero):

```
BLANK_TIME=0
POWERDOWN_TIME=0
```

Quindi cambiare lo script in Xsession per disabilitare il salvaschermo e impostare all’avvio il browser Chromium in modalità kiosk, ovvero a pieno schermo e senza la barra dei menu.

Dal terminale digitare il seguente comando:

```
sudo leafpad .config/lxsession/LXDE-pi/autostart
```

Aggiungere alla fine della lista le seguenti istruzioni, salvare il documento e riavviare:

```
@xset s noblank
@xset s off
@xset -dpms
@chromium-browser --kiosk - incognito http://localhost
```

Al riavvio, il salvaschermo verrà disabilitato permanentemente e si avvierà automaticamente il browser Chromium in modalità incognito, ovvero senza mostrare eventuali errori di ripristino delle pagine web, nel caso venga tolta l'alimentazione a Raspberry Pi senza eseguire lo spegnimento. Il browser sarà visibile a tutto schermo con la home page del nostro Magic Mirror e senza barre di scorrimento. Il risultato è visibile nella Figura 14.6.

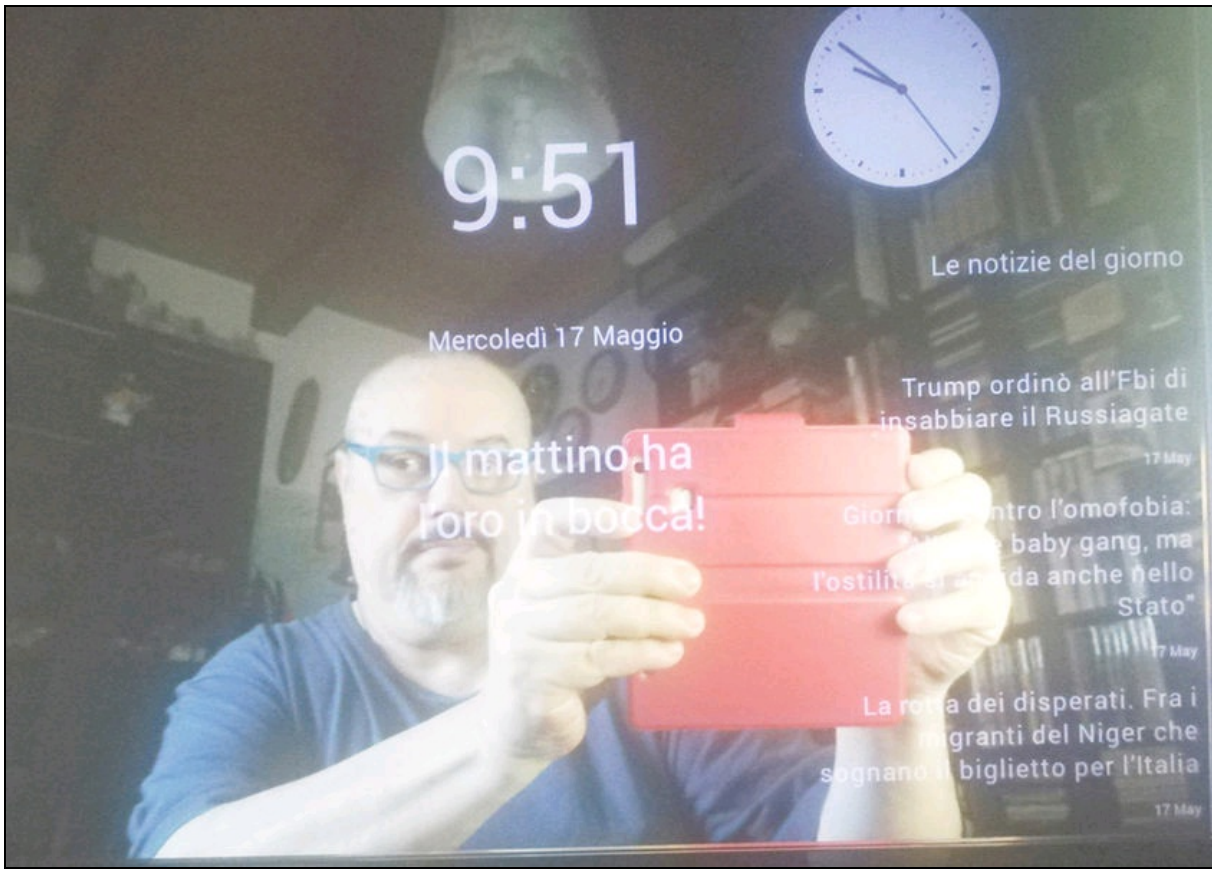


Figura 14.6 La home page del nostro Magic Mirror.

Per le modifiche al codice dei documenti `index.php` e `style.css` o per qualsiasi accesso al terminale, ci si potrà collegare tramite una connessione SSH da un computer remoto collegato alla stessa rete.

Bilancia intelligente

Descrizione

Questo è l’ultimo progetto del libro. Ultimo non perché sia meno importante, ma per chiudere in bellezza con qualcosa che “parlasse”. Oltre alla funzione Bluetooth, paragonabile alle molte bilance intelligenti che si trovano in commercio, la nostra bilancia possiede una voce. La voce ci dirà se siamo in sovrappeso o abbiamo raggiunto il peso ideale.

Il progetto di bilancia intelligente si basa su una scheda Arduino, un modem Bluetooth, una cella di carico con relativo amplificatore e un’applicazione Bluetooth “parlante” per smartphone Android.

Dato che in commercio ci sono costose bilance wireless dotate di app dedicate, abbiamo voluto cimentarci da veri maker nella programmazione da zero di una nostra app.

CODICE DI ESEMPIO

Tutti i file del progetto “Bilancia intelligente” sono disponibili nelle risorse del libro presso il sito dell’autore all’indirizzo `http://www.pierduino.com`.

Materiale occorrente

Il materiale usato per questo progetto è:

- 1× Arduino UNO;
- 1× modem HC-05 (o HC-06);
- 1× cella di carico;
- 1× amplificatore per cella di carico;
- 1× bilancia di qualsiasi tipo (opzionale);
- 1× piano in legno, plastica o altro (opzionale).

Cella di carico

A dir la verità, inizialmente avevamo sperimentato una cella di carico fatta in casa, nella speranza di ottenere un ponte di Wheatstone che potesse leggere con precisione un sensore piezoelettrico di pressione. Purtroppo, per raggiungere una buona precisione nella misurazione della pressione (intesa come forza fisica) abbiamo dovuto optare per una cella di carico commerciale. Fra le tante disponibili abbiamo scelto una cella di carico a disco, siglata TAS606 (Figura 1a) che sopporta un peso (forza) di 200 kg. Questo modello è disponibile presso Robot Italy (https://www.robot-italy.com).

Questa cella di carico è di tipo “a disco” ed è costituita da una lega di acciaio molto resistente. Può convertire con precisione pesi fino a 200 kg in un segnale elettrico. Com’è noto, ogni cella di carico è in grado di misurare la resistenza elettrica che cambia in risposta alla pressione o, meglio, alla forza applicata.

La cella di carico è composta da quattro resistori cablati nella tipica configurazione a ponte di Wheatstone (Figura 1b). Oltre alla calza di

isolamento, il cavo coassiale è composto da quattro fili, il cui cablaggio è il seguente.

- Rosso = E+ (eccitazione positiva).
- Verde/blu = O+ (uscita positiva).
- Nero = E- (eccitazione negativa).
- Bianco/giallo = O- (uscita negativa).

Si consiglia di saldare quattro cavetti con pin maschio ai quattro fili del cavo coassiale e proteggerli con della guaina termoretraibile (Figura 1c). La calza non va collegata al circuito.

Le dimensioni della nostra cella di carico sono molto ridotte (20 mm di diametro per 11 mm di altezza) e può essere alloggiata senza problemi sotto una normale bilancia.

Questo modello di cella da 200 kg è abbastanza costoso, per cui, se nel progetto dovesse bastare una cella da 100 kg, si può risparmiare qualche decina di euro.

Esistono celle di carico a barra (Figura 15.1d) che di solito sono dotate di filetto per essere fissate a un piano in legno o altro materiale. Possono andare benissimo se si vuole creare, per esempio, una bilancia ex novo.

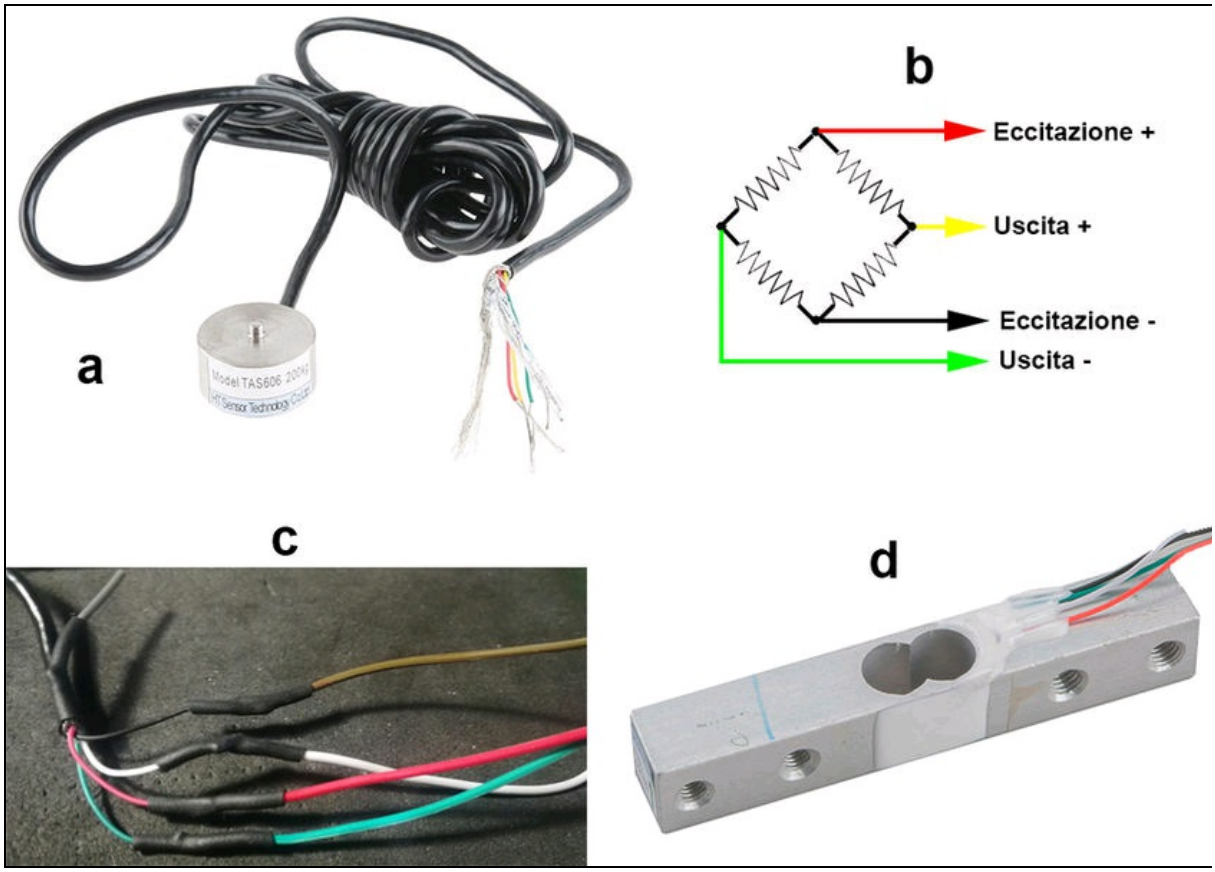


Figura 15.1 Cella di carico a disco da 200 kg (a). Ponte di Wheatstone (b). Saldatura dei 4 fili (c). Cella di carico a barra (d).

Amplificatore per cella di carico

Per poter leggere il debole segnale in uscita dalla cella di carico è necessario un amplificatore. Quello scelto per questo progetto è l'amplificatore *HX711*, non fosse altro perché esiste una libreria dedicata per Arduino.

L'amplificatore HX711 è disponibile su scheda breakout (Figura 15.2a) che facilita le connessioni alla cella e alla scheda Arduino. L'amplificatore HX711 è disponibile presso SparkFun (`https://learn.sparkfun.com`) o uno dei suoi rivenditori in Italia.

È necessario saldare due connettori femmina per lato, non inclusi (Figura 15.2b).

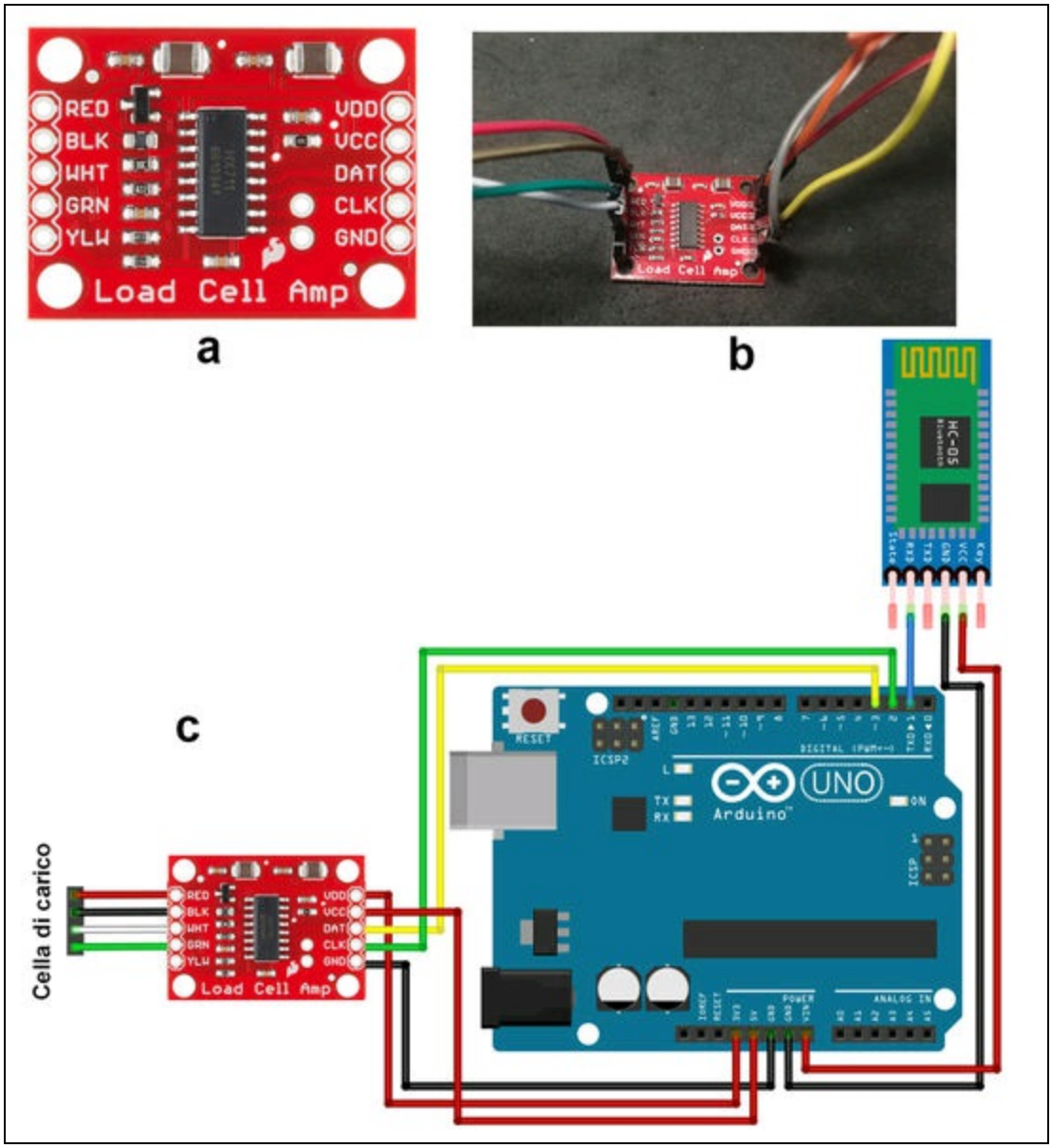


Figura 15.2 L'amplificatore HX711 (a). Saldatura dei connettori all'amplificatore (b). Cablaggio della cella di carico e di Arduino (c).

Il circuito

Quella che segue è una guida per collegare la nostra cella di carico all'amplificatore HX711, ad Arduino UNO e a un modem Bluetooth, anche se si potrebbe usare una scheda Arduino 101 già dotata di Bluetooth (si veda il Capitolo 10). In questo caso la procedura sarebbe molto più complessa.

Facendo riferimento alla Figura 15.2c, seguire la colorazione standard del cavo coassiale della cella di carico e la serigrafia riportata sull'amplificatore HX711.

I quattro fili colorati di solito riportano i seguenti nomi.

- Rosso: eccitazione + (E+) o VCC.
- Nero: eccitazione - (E-) o GND.
- Bianco: uscita+ (O+), segnale+ (S+) o amplificatore+ (A+).
- Verde: uscita- (O-), segnale- (S-) o amplificatore - (A-).

Collegare la cella all'amplificatore HX711 seguendo lo schema seguente:

Pin HX711	Filo della cella di carico
RED (red)	Rosso
BLK (black)	Nero
WHT (white)	Bianco
GRN (green)	Verde
YLW (yellow)	n.c

Collegare l'amplificatore HX711 alla scheda Arduino seguendo lo schema seguente:

Pin HX711	Pin Arduino
VDD	3.3 V
VCC	5 V
DAT	3

CLK	2
GND	GND

Si fa notare che all’amplificatore HX711 servono due alimentazioni (VDD e VCC), per cui sono stati usati i pin 3.3 V e 5 V di Arduino.

La Figura 15.2c mostra come collegare il modem Bluetooth HC-05 alla scheda Arduino. Come si può vedere, per trasmettere i dati al pin RX del modem viene usato il pin TX di Arduino. Come si vedrà più avanti, il modem provvederà a trasmettere all’applicazione Bluetooth i dati seriali della cella, ovvero il peso in kg.

Visto l’esiguo numero di componenti, non serve usare una millefori o un circuito stampato e si possono effettuare collegamenti volanti anche sotto la bilancia stessa.

In alternativa, il circuito cablato può stare comodamente dentro una scatola in plastica e, visto il basso consumo a riposo, può funzionare anche con una pila da 9 volt collegata al connettore della scheda Arduino.

Posizionare la cella di carico

La cella di carico a disco può essere comodamente posizionata sotto una normale bilancia digitale o analogica. La Figura 15.3 mostra come abbiamo sistemato la cella di carico con del semplice biadesivo, insieme a tutta la circuiteria. Se la distanza da terra non fosse sufficiente o se la bilancia venisse sollevata dalla cella, si possono usare spessori per alzare la cella o per alzare la bilancia.

Se il circuito non trovasse posto sotto la bilancia, il cavo di collegamento è abbastanza lungo per poter alloggiare il contenitore contenente il circuito dove è più comodo e, se si usa nella stanza da bagno, distante dall’acqua.

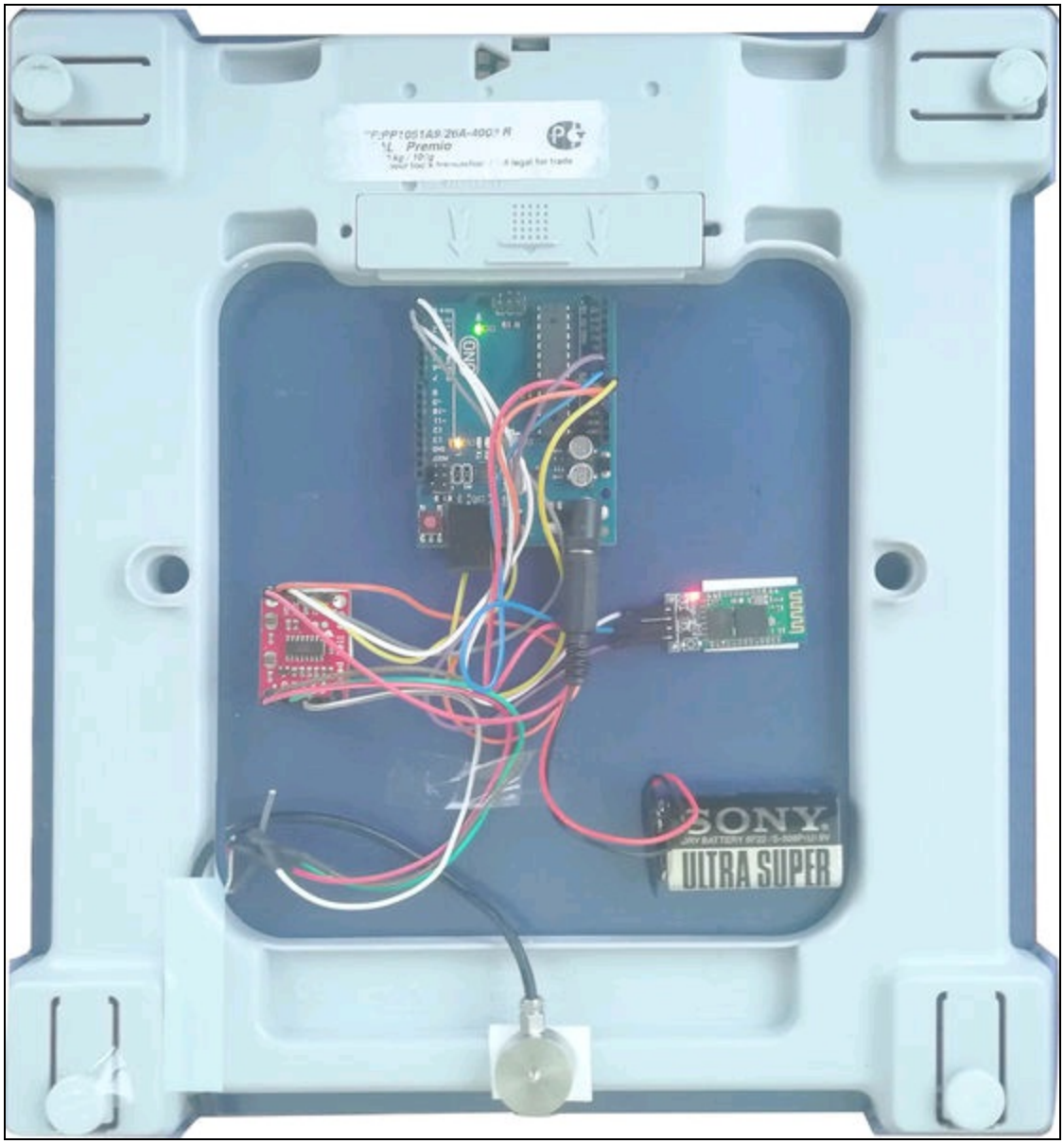


Figura 15.3 La cella di carico sotto una bilancia elettronica.

ATTENZIONE

Per questo progetto viene usata una sola cella di carico a solo scopo didattico, per cui le misurazioni potrebbero non essere attendibili. Per aumentare la precisione (e anche i costi) si possono combinare due o più celle di carico e collegarle a un combinatore, come quello proposto da SparkFun (https://www.sparkfun.com/products/13878).

Il codice

Prima di passare all'analisi del codice di trasmissione dei dati della cella di carico all'applicazione Bluetooth, bisogna effettuare la sua calibrazione. Innanzitutto, è necessario installare la libreria HX711, disponibile nelle risorse del libro e in questo repository di SparkFun: `https://github.com/sparkfun/HX711-Load-Cell-Amplifier`.

Per le istruzioni di installazione della libreria si veda il box "Come aggiungere una libreria all'IDE di Arduino" nel Capitolo 5.

Calibrazione

Fra gli esempi scaricati dal repository è presente lo sketch `SparkFun_HX711_Calibration`, ma si consiglia di usare il nostro sketch `Scale_Calibration` perché è già predisposto per una calibrazione in chilogrammi e mostra le istruzioni in italiano, mentre lo sketch originale è calibrato per una misurazione in libbre con le istruzioni in inglese.

Codice commentato

Il codice commentato illustra la procedura per la calibrazione della cella di carico. All'inizio dello sketch viene inclusa la libreria HX711 e vengono definiti i pin 3 e 2 per la comunicazione dell'amplificatore con Arduino. Poi viene istanziato l'oggetto `scale` che eredita i metodi della libreria.

```
#include "HX711.h"
#define DOUT 3
#define CLK 2
    HX711 scale(DOUT, CLK);
```

Con la variabile `float calibration_factor` viene definito il valore standard di -7050 unità di calibrazione. Questo valore preimpostato è

stato ricavato sperimentalmente da SparkFun, ma va modificato manualmente.

```
float calibration_factor = -7050;
```

Nella funzione `setup()` viene inizializzato il monitor seriale che visualizzerà le istruzioni per la calibrazione, come illustrato nella Figura 15.4a. In pratica, bisognerà collocare sulla bilancia un peso noto, per esempio un chilogrammo, e inviare il carattere + o la lettera a per incrementare il fattore di calibrazione oppure il carattere - o la lettera z per decrementarlo.

```
void setup() {
    Serial.begin(9600);
    Serial.println("HX711 sketch di calibrazione");
    Serial.println("Rimuovere il peso dalla bilancia");
    Serial.println("Appena inizia la lettura, mettere un peso noto sulla
bilancia");
    Serial.println("Premere + o 'a' per incrementare il fattore di calibrazione");
    Serial.println("Premere - o 'z' per decrementare il fattore di calibrazione");
```

Le istruzioni `scale.set_scale()` e `scale.tare()` inizializzano la cella e impostano la tara a zero. Se però esiste una tara, bisognerà impostare il valore della tara fra parentesi in unità di calibrazione, non in chilogrammi.

```
scale.set_scale();
scale.tare();
```

Nella funzione `loop()` viene effettuata la lettura del peso, ovvero della forza applicata alla cella di carico. Con l’istruzione `scale.set_scale(calibration_factor)` viene impostato il valore standard di calibrazione, che è -7050.

La variabile `kg`, di tipo `float` converte il valore da libbre in chilogrammi, moltiplicando il valore letto per 0.453592. Nonostante le sei cifre decimali, verranno visualizzate solo due cifre decimali dopo la virgola, per una limitazione del firmware di Arduino.

Quindi, nel monitor seriale apparirà la scritta “Kg:” più il valore in chilogrammi e il fattore di calibrazione (Figura 15.4b).

```
void loop() {
    scale.set_scale(calibration_factor);
```

```
    float kg = 0.453592 * scale.get_units();
    Serial.print("Kg: ");
    Serial.print(kg);
    Serial.print("\t");
    Serial.println(calibration_factor);
    delay(200);
```

A questo punto, appoggiare sulla cella di carico un peso noto, per esempio 1 Kg e inviare tramite il monitor seriale il carattere + o la lettera ‘a’ oppure il carattere - o la lettera ‘z’ per incrementare o decrementare il fattore di calibrazione a passi di 100 unità. Per regolazioni fini si può impostare l’incremento/decremento a 10 unità.

Quando il peso di riferimento (1 kg) viene visualizzato con continuità, prendere nota del fattore di calibrazione, come indicato nella Figura 15.4c.

```
if(Serial.available())
{
    char temp = Serial.read();
    if(temp == '+' || temp == 'a')
        calibration_factor += 100;
    else if(temp == '-' || temp == 'z')
        calibration_factor -= 100;
}
```

Attenzione! Se la lettura dà un valore negativo quando si appoggia un peso, invertire l’uscita - con l’uscita + della cella di carico. Nel nostro caso, il filo bianco con il filo verde.

Caricare lo sketch Smart Scale Bluetooth

Dopo aver calibrato la cella di carico, si può caricare in Arduino lo sketch che abbiamo chiamato *Smart Scale Bluetooth*. Le poche righe del listato fanno capire la sua semplicità.

Con la direttiva `#define calibration_factor` si deve impostare il fattore di calibrazione ottenuto con lo sketch di calibrazione. Nel nostro caso il fattore di calibrazione è risultato essere -8950.

```
#define calibration_factor -8950.0 // fattore di calibrazione
```

Nella funzione `setup()` viene inizializzata la porta seriale a 57600 baud, ovvero il baud rate della UART del modem HC-05 (si veda il Capitolo

10 per come cambiare i parametri del del modem).

Quindi, con l'istruzione *scale.set_scale(calibration_factor)* si imposta il fattore di calibrazione.

```
void setup() {
    Serial.begin(57600); // baud rate del modem HC-05
    scale.set_scale(calibration_factor); // imposta il fattore di calibrazione
```

Nella funzione `loop()` viene eseguita la lettura della cella con la conversione diretta in chilogrammi tramite l'istruzione `float kg = 0.453592 * scale.get_units()`.

```
void loop() {
    float kg = 0.453592 * scale.get_units();
```

Per monitorare se tutto funziona, con l'istruzione `Serial.println(kg)` viene visualizzato sul monitor seriale il peso in chilogrammi a intervalli di un secondo (Figura 15.4d). Una volta verificato che la cella di carico dà una lettura abbastanza vicina a quella della bilancia, si può commentare questa riga e decommentare quella successiva, che invece scrive il byte di lettura in kg sulla porta seriale con l'istruzione `Serial.write(byte(kg))`.

```
Serial.println(kg);        // commentare dopo il debug
//Serial.write(byte(kg));  // decommentare per il modem
delay(1000);
```

Attenzione! La conversione da `float` a `byte` arrotonda al valore intero più vicino e pertanto si perde la precisione del numero in virgola mobile. Questo è un effetto voluto, per semplificare la trasmissione seriale di un unico dato e per non complicare il codice di ricezione Bluetooth, che andiamo a spiegare nel prossimo paragrafo.

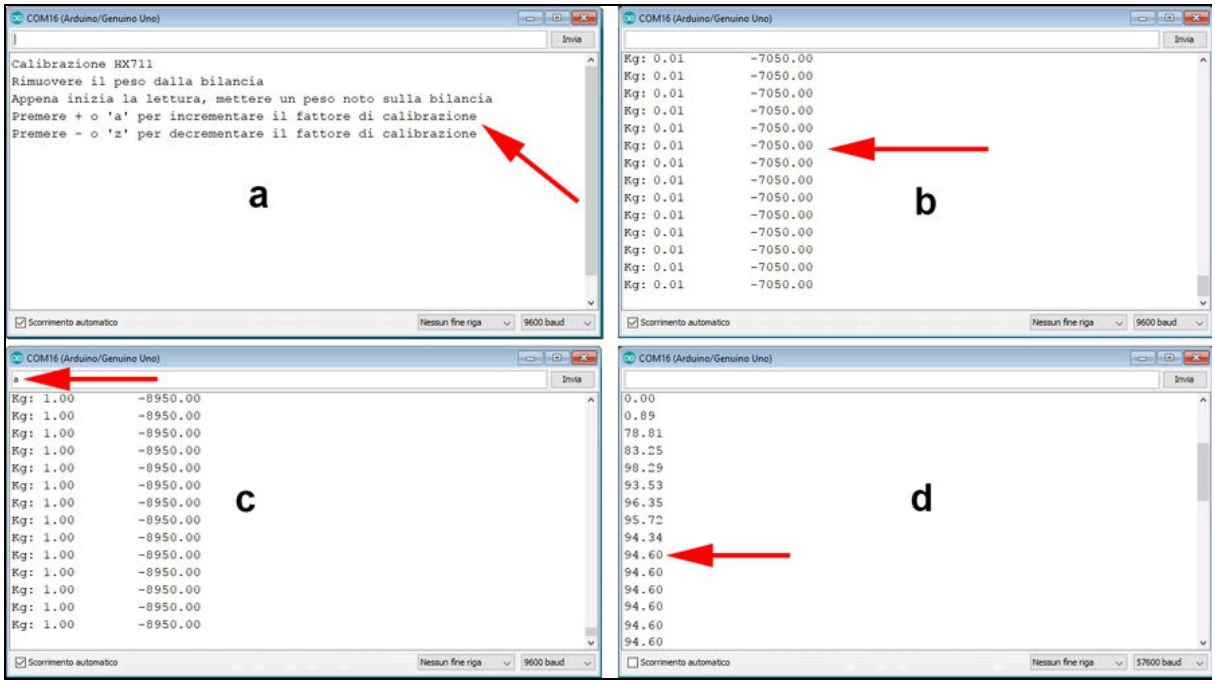


Figura 15.4 Istruzioni per la calibrazione (a). Avvio della lettura della cella di carico (b). Regolazione tramite l'invio di un carattere sulla porta seriale (c). Il peso rilevato (d).

L'applicazione Smart Scale

Nel "Capitolo 10 - Apertura cancello da smartphone" si è visto come creare un'app per la trasmissione di dati Bluetooth tramite un'interfaccia creata con Xamarin. In questo paragrafo verrà spiegato come implementare la ricezione Bluetooth e, soprattutto, come implementare l'ottimo sintetizzatore vocale *Text-to-Speech* di Google.

Creazione di un nuovo progetto

Dal menu *File* > *Nuovo progetto*, selezionare *Android* dall'elenco dei modelli e un'app vuota. Dare un nome all'app vuota, per esempio, `SmartScaleApp`, come indicato nella Figura 15.5a.

Verrà creata una soluzione e un progetto con il nome `SmartScaleApp`, contenente un file in C#, chiamato `MainActivity.cs` (Figura 15.5b). Questo

è il file che bisognerà modificare e in cui scrivere tutto il codice dell'applicazione. Aprendo questo file apparirà il codice in C# di un'applicazione vuota con una sola classe principale `MainActivity`, all'interno del `namespace SmartScaleApp`:

```
namespace SmartScaleApp
{
    [Activity(Label = "SmartScaleApp", MainLauncher = true, Icon =
"@drawable/icon")]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            // Set our view from the "main" layout resource
            // SetContentView (Resource.Layout.Main);
        }
    }
}
```

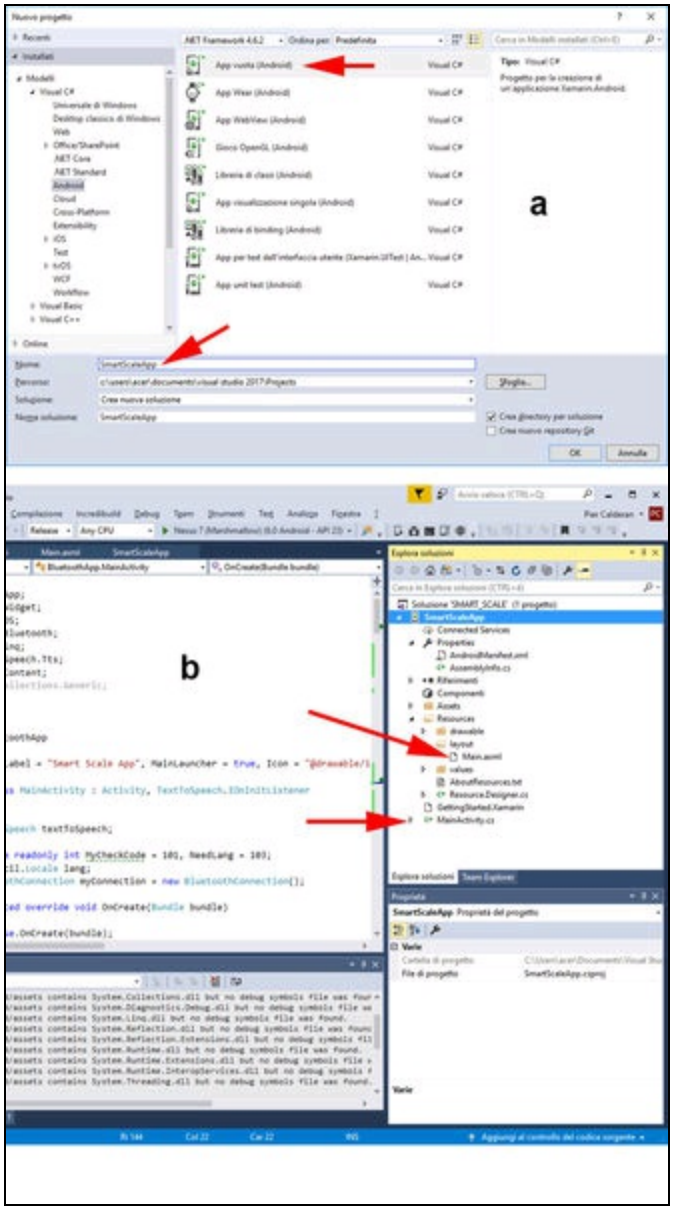


Figura 15.5 Finestra File > Nuovo progetto (a). Soluzione e progetto SmartScaleApp (b).

Bisognerà scrivere il codice solo all'interno della classe `MainActivity`. Prima però, è necessario creare l'interfaccia grafica dell'applicazione. La cartella `Resources` contiene altre cartelle e file.

- `Drawable`: contiene l'icona di default.
- `Layout`: contiene il file `Main.axml`.

- `Values`: contiene il file `Strings.xml`.
- `AboutResources.txt`: è un file di testo con le istruzioni per le risorse.

Interfaccia grafica

Come in tutte le applicazioni scritte in ambiente C#, anche con Xamarin è possibile costruire l'interfaccia per il proprio smartphone.

> **NOTA**
> Se si utilizza l'emulatore Android SDK di Google, si consiglia di configurare l'emulatore per utilizzare l'accelerazione hardware. Le istruzioni per la configurazione dell'accelerazione hardware sono disponibili all'indirizzo: https://developer.xamarin.com/guides/android/getting_started/installation/acceler
>
> Se si utilizza l'emulatore Android di Visual Studio, Hyper-V deve essere abilitato nel computer. Per ulteriori informazioni sulla configurazione di Visual Studio Android Emulator, consultare la pagina: https://msdn.microsoft.com/en-us/library/mt228280.aspx.

Il linguaggio del codice è molto simile a XAML, solo che è stato chiamato AXML perché è destinato a interfacce esclusivamente create per la piattaforma Android.

Innanzitutto, bisogna scegliere il dispositivo di destinazione dal menu in alto, come indicato dalla freccia a destra nella Figura 15.6a.

Per installare un dispositivo virtuale fare clic su *Gestisci dispositivi virtuali* (freccia in basso della Figura 15.6a). Si aprirà una la finestra *Xamarin Android Player* in cui si può scegliere il dispositivo virtuale di destinazione. Per il nostro progetto abbiamo scelto il dispositivo *Nexus 7 (Marshmallow)*, basato su Android 6.0 (API23) (Figura 15.6b). Dalla stessa finestra si possono installare altri dispositivi compatibili con versioni di Android precedenti o successive.

Una volta scelto il dispositivo di destinazione, con un doppio clic sul file `Main.axml`, si aprirà la finestra di progettazione con l'interfaccia del dispositivo completamente vuota e con il nome *SmartScaleApp* sulla barra dell'applicazione (Figura 15.6c).

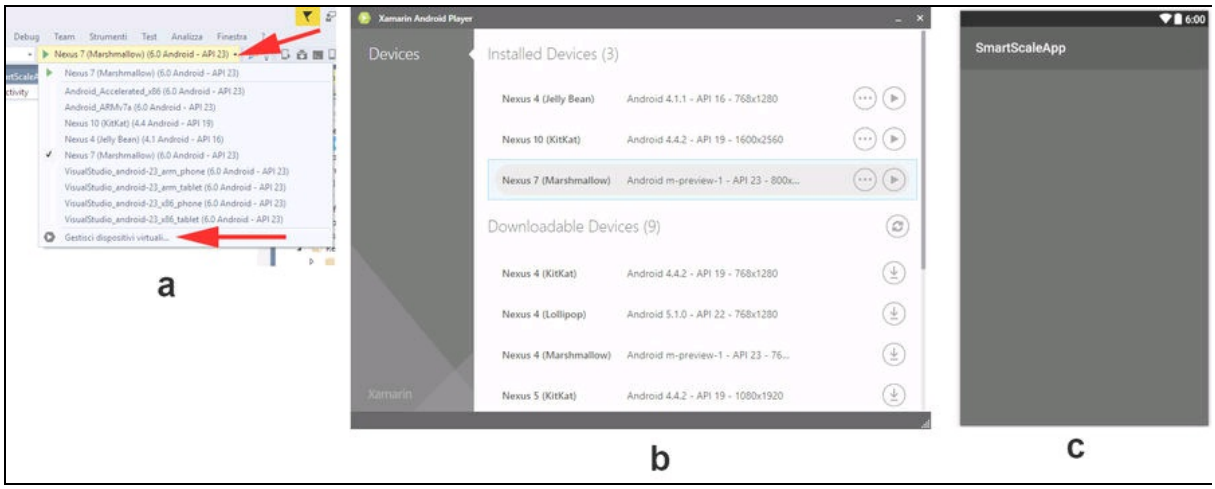


Figura 15.6 Menu del dispositivo di destinazione (a). Finestra Xamarin Android Player (b). Finestra di progettazione del dispositivo (c).

Procedura per aggiungere i componenti all'interfaccia

1. Aprire la casella degli strumenti dal menu *Visualizza*.
2. Selezionare il componente *TextView* e trascinarlo sull'interfaccia.
3. Dal menu *Proprietà* selezionare l'opzione *text* dalla sezione *Main* e digitare **0 kg**.
4. Dalla sezione *Style* selezionare l'opzione *textColor* e impostare il colore rosso (*#ff0000*) o un altro colore. Con l'opzione *textSize* impostare 60dp o un'altra dimensione del carattere.
5. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
6. Dal menu *Proprietà* selezionare l'opzione *text* e digitare **CONNECT**.
7. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
8. Dal menu *Proprietà* selezionare l'opzione *text* e digitare **DISCONNECT**.
9. Dal menu *Proprietà* selezionare l'opzione *text* e digitare **Disconnected**.

10. Dal menu *Proprietà* selezionare l'opzione *layout_marginLeft* dalla sezione *Layout - ViewGroup* e impostare il margine a sinistra a 120dp.
11. Selezionare il componente *Plain Text* e trascinarlo sull'interfaccia.
12. Dal menu *Proprietà* selezionare l'opzione *text* e digitare **My target**.
13. Dal menu *Proprietà* selezionare l'opzione *layout_marginLeft* dalla sezione *Layout - ViewGroup* e impostare il margine a sinistra a 150dp. Impostare anche l'opzione *layout_marginTop* a 30dp.
14. Selezionare il componente *Button* e trascinarlo sull'interfaccia.
15. Dal menu *Proprietà* selezionare l'opzione *text* e digitare **SPEAK**.
16. Dal menu *Proprietà* selezionare l'opzione *layout_marginTop* dalla sezione *Layout -ViewGroup* e impostare il margine in alto a 50dp. Impostare anche l'opzione *layout_marginLeft* a 150dp.
17. Selezionare il componente *TextView* e trascinarlo sull'interfaccia.
18. Dal menu *Proprietà* selezionare l'opzione *text* e digitare **Message**.

La Figura 15.7 illustra il risultato finale. Terminata l'interfaccia, si può procedere alla scrittura del codice in C#.

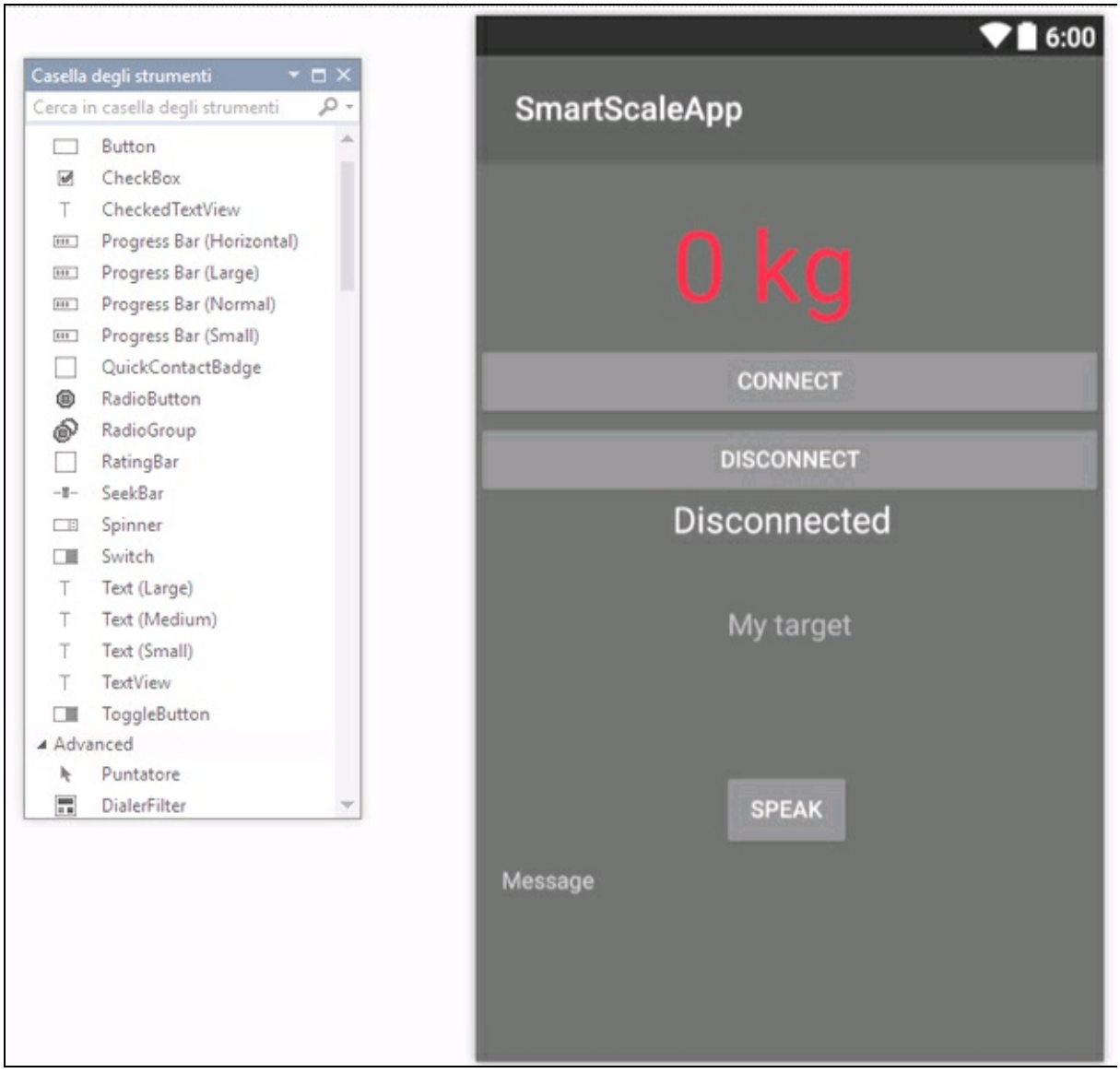


Figura 15.7 Risultato finale dell'interfaccia Smart Scale App (b).

Il codice C#

Per facilitare la comprensione e la modifica del listato in base alle necessità, vengono spiegate solo le funzioni essenziali. Il file `MainActivity.cs` include alcune librerie Android, per cui bisogna aggiungere all'inizio del listato le seguenti direttive `using`:

```
using System;
using Android.App;
```

```
using Android.Widget;
using Android.OS;
using Android.Bluetooth;
using System.Linq;
using Android.Speech.Tts;
using Android.Content;
```

Come si può vedere, si attivano le funzionalità Bluetooth e soprattutto Text-to-Speech, ovvero la possibilità di far leggere qualsiasi testo e di farlo pronunciare da una voce sintetica.

All'inizio della classe `public class MainActivity`, bisogna inserire la chiamata alla classe `BluetoothConnection` e alla funzione `TextToSpeech`. Si fa notare che il nome del dispositivo Bluetooth HC-05 è quello predefinito del modem HC-05 usato con Arduino. Se viene cambiato il nome, si dovrà cambiare anche qui.

Nella funzione `TextToSpeech` viene inizializzata la lingua locale di default, ovvero l'italiano.

```
BluetoothConnection myConnection = new BluetoothConnection();
TextToSpeech textToSpeech;
```

Questo significa che bisogna creare rispettivamente la classe `BluetoothConnection` e la funzione `TextToSpeech`:

```
public class BluetoothConnection
{
    public void getAdapter() { this.thisAdapter = BluetoothAdapter.DefaultAdapter;
}
    public void getDevice() { this.thisDevice = (from bd in
this.thisAdapter.BondedDevices
        where bd.Name == "HC-05" select bd).FirstOrDefault(); }
    public BluetoothAdapter thisAdapter { get; set; }
    public BluetoothDevice thisDevice { get; set; }
    public BluetoothSocket thisSocket { get; set; }
}
void TextToSpeech.IOnInitListener.OnInit(OperationResult status)
{
    if (status == OperationResult.Error)
        textToSpeech.SetLanguage(Java.Util.Locale.Default);
    if (status == OperationResult.Success)
        textToSpeech.SetLanguage(lang);
}
```

All'interno della funzione `protected override void OnCreate(Bundle bundle)` si devono aggiungere le istruzioni per la gestione dei pulsanti e dei testi che sono stati aggiunti all'interfaccia grafica. I nomi dei pulsanti e delle

caselle di testo sono quelli di default e, per gestirli all'interno del listato, vengono tradotti utilizzando la funzione `FindViewById`.

```
Button buttonConnect = FindViewById<Button>(Resource.Id.button1) ;
Button buttonDisconnect = FindViewById<Button>(Resource.Id.button2);
Button speak = FindViewById<Button>(Resource.Id.button3);
TextView connected = FindViewById<TextView>(Resource.Id.textView1);
TextView readTextView = FindViewById<TextView>(Resource.Id.textView2);
TextView message = FindViewById<TextView>(Resource.Id.textView3);
TextView resultView = FindViewById<TextView>(Resource.Id.textView4);
EditText myWeight = FindViewById<EditText>(Resource.Id.editSpeech);
```

Il modulo Text-to-Speech viene scaricato direttamente dal repository di Google, ovvero `com.google.android.tts`. La lingua di default è l'italiano, che viene impostata grazie alla libreria `Java.Util.Locale.Default`.

```
textToSpeech = new TextToSpeech(this, this, "com.google.android.tts");
lang = Java.Util.Locale.Default;
textToSpeech.SetLanguage(lang);
```

A questo punto basta impostare i parametri della voce sintetica perché sia più gradevole possibile. Con l'istruzione `textToSpeech.SetPitch` viene impostata l'intonazione della voce e con `textToSpeech.SetSpeechRate` viene impostata la velocità. I valori che abbiamo impostato producono una voce femminile gradevolissima. Il valore di default per entrambi i parametri è `.5f` (la "f" sta per *floating*). Provare i valori che meglio si adattano ai propri gusti.

```
textToSpeech.SetPitch(1.2f);
textToSpeech.SetSpeechRate(1.0f);
```

La funzione all'interno di `buttonConnect` tenterà la connessione Bluetooth:

```
buttonConnect.Click += delegate {
myConnection = new BluetoothConnection();
myConnection.getAdapter();
myConnection.thisAdapter.StartDiscovery();
…
…
myConnection.thisSocket.Connect();
connected.Text = "Connected!";
buttonDisconnect.Enabled = true;
buttonConnect.Enabled = false;
```

Come si può vedere, una volta ottenuta la connessione, la casella di testo `connected` cambierà il testo in "Connected!" e disabiliterà il pulsante.

In modo simile il pulsante `buttonDisconnect` eseguirà la disconnessione.

Il segreto per ricevere i dati dalla connessione Bluetooth appena instaurata sta tutto nella funzione `Listener`. Per semplicità viene letto un solo byte nell'array (quello mandato dalla porta seriale di Arduino) che viene memorizzato nella prima posizione dell'array con l'istruzione `myConnection.ThisSocket.InputStream.Read(read, 0, 1)`.

```
void Listener()
{
    byte[] read = new byte[1];
    while (true)
    {
        try
        {
            myConnection.ThisSocket.InputStream.Read(read, 0, 1);
            myConnection.ThisSocket.InputStream.Close();
```

Se il valore letto è maggiore di zero viene mandato alla casella di testo `readTextView`. Con l'aggiunta di "kg" alla stringa si potrà vedere il peso e la scritta "kg". Altrimenti, se il peso è zero, verrà visualizzata la scritta "0 kg".

```
if (read[0] > 0)
{
    weight = read[0].ToString();
    readTextView.Text = weight + " kg";
    intWeight = read[0];
}
else if (read[0] == 0)
{
    readTextView.Text = "0 kg";
}
```

Il segreto per far parlare la voce sintetica sta all'interno della funzione `speak.click`, che viene chiamata dalla pressione del pulsante *SPEAK* dell'interfaccia grafica.

Come si può vedere, viene fermata la ricezione dei dati con l'istruzione `listenThread.Abort()` e vengono convertiti i valori stringa dei campi di testo in numeri interi. In questo modo si potrà eseguire l'operazione aritmetica di sottrazione fra il peso rilevato e il peso forma, ovvero il peso impostato in "My target".

La variabile intera `result` memorizza il risultato della sottrazione fra il peso rilevato e il peso “target”.

```
speak.Click +=delegate{
try
{
    listenThread.Abort();
    String readWeight = "";
    int intweight = Convert.ToInt32(myWeight.Text);
    result = intWeight - intweight;
```

Le righe che seguono sono alcune frasi che si riferiscono a tipiche situazioni quando ci si pesa su una bilancia. Ovviamente si possono cambiare a piacere!

```
if (result >= 5 && result <= 8) readWeight = ". Peso ideale!";
if (result > 8 && result <= 20) readWeight = ". Sei in leggero sovrappeso.";
if (result > 20 && result <= 30) readWeight = ". Sei in grave sovrappeso.";
if (result > 30) readWeight = ". Sei obeso!";
if (result < 0) readWeight = ". Sei sotto peso!";
```

In pratica:

- se `result` è fra 5 e 8, la frase da aggiungere è “Peso ideale”;
- se `result` è fra 8 e 20, la frase da aggiungere è “Sei in leggero sovrappeso”;
- se `result` è fra 20 e 30, la frase da aggiungere è “Sei in grave sovrappeso”;
- se `result` è oltre 30, la frase da aggiungere è “Sei obeso”;
- se `result` è un valore negativo, la frase da aggiungere è “Sei sotto peso”.

In questo modo si può far pronunciare la frase completa “Il tuo peso è di chilogrammi *xx*” più la frase aggiunta in base al valore della variabile `result`.

Per esempio, se il peso target è di 76 kg, la voce dirà “Il tuo peso è di chilogrammi 80. Sei in leggero sovrappeso”. Non è cool tutto questo?

```
message.Text = "Il tuo peso è di chilogrammi " + weight + readWeight;
textToSpeech.Speak(message.Text, QueueMode.Flush, null);
```

Compilazione e distribuzione dell’app

Terminata la scrittura del codice, si può procedere alla compilazione della soluzione e alla successiva distribuzione al dispositivo Android.

Prima di procedere alla compilazione, bisogna accedere ad alcune configurazioni. Dal menu *Debug*, selezionare *Proprietà* per aprire la finestra omonima ed eseguire le impostazioni seguenti.

1. Nella scheda *Applicazione* selezionare la versione di Android per la compilazione, per esempio Android 6.0 (Marshmallow) e la versione minima, per esempio Android 4.1 (livello API 16 - Jelly Bean) (Figura 15.8a).
2. Nella scheda *Manifesto Android* selezionare le funzionalità: *BLUETOOTH*, *BLUETOOTH_ADMIN*, *BLUETOOTH_PRIVILEGED* e impostare l’icona su *@drawableicon/Icon*, ovvero l’icona di default delle risorse (Figura 15.8b).

Non dovendo eseguire il debug del codice, si può impostare direttamente la configurazione della soluzione su *Release* (Figura 15.8c). Selezionare il dispositivo per cui effettuare la compilazione, per esempio, *Android 6.0 (Marshmallow)* e dal menu *Compilazione*, selezionare *Compila soluzione*. Se non ci sono errori la compilazione avviene senza problemi, altrimenti appariranno gli errori nella finestra di Output.

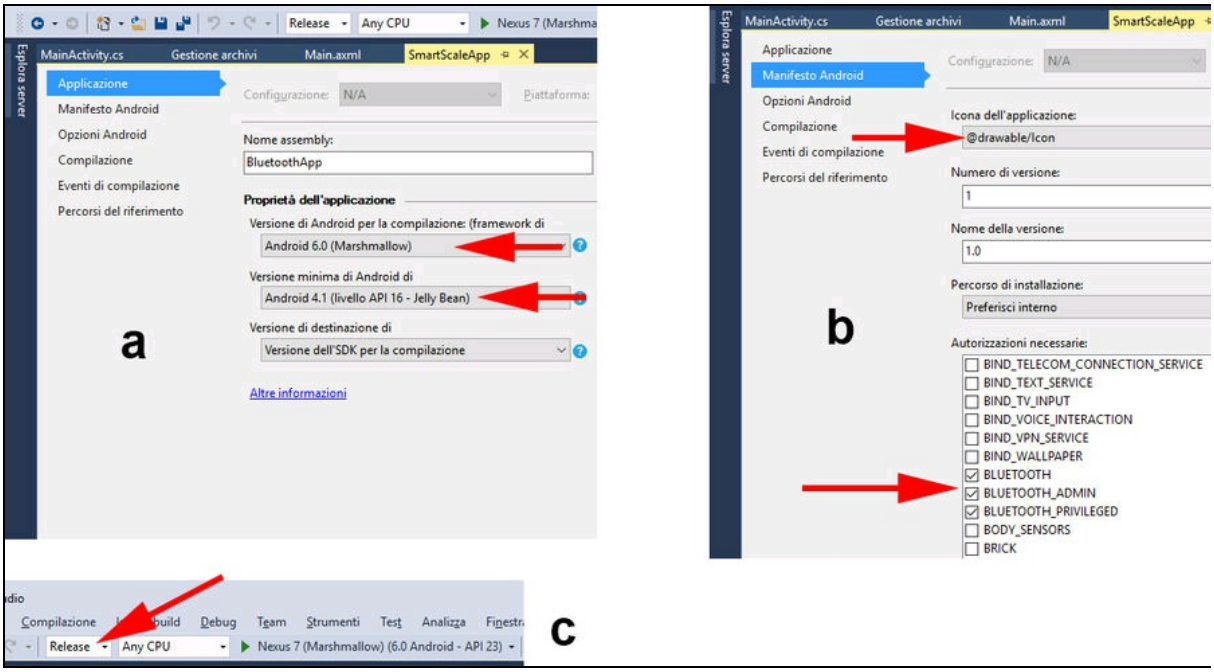


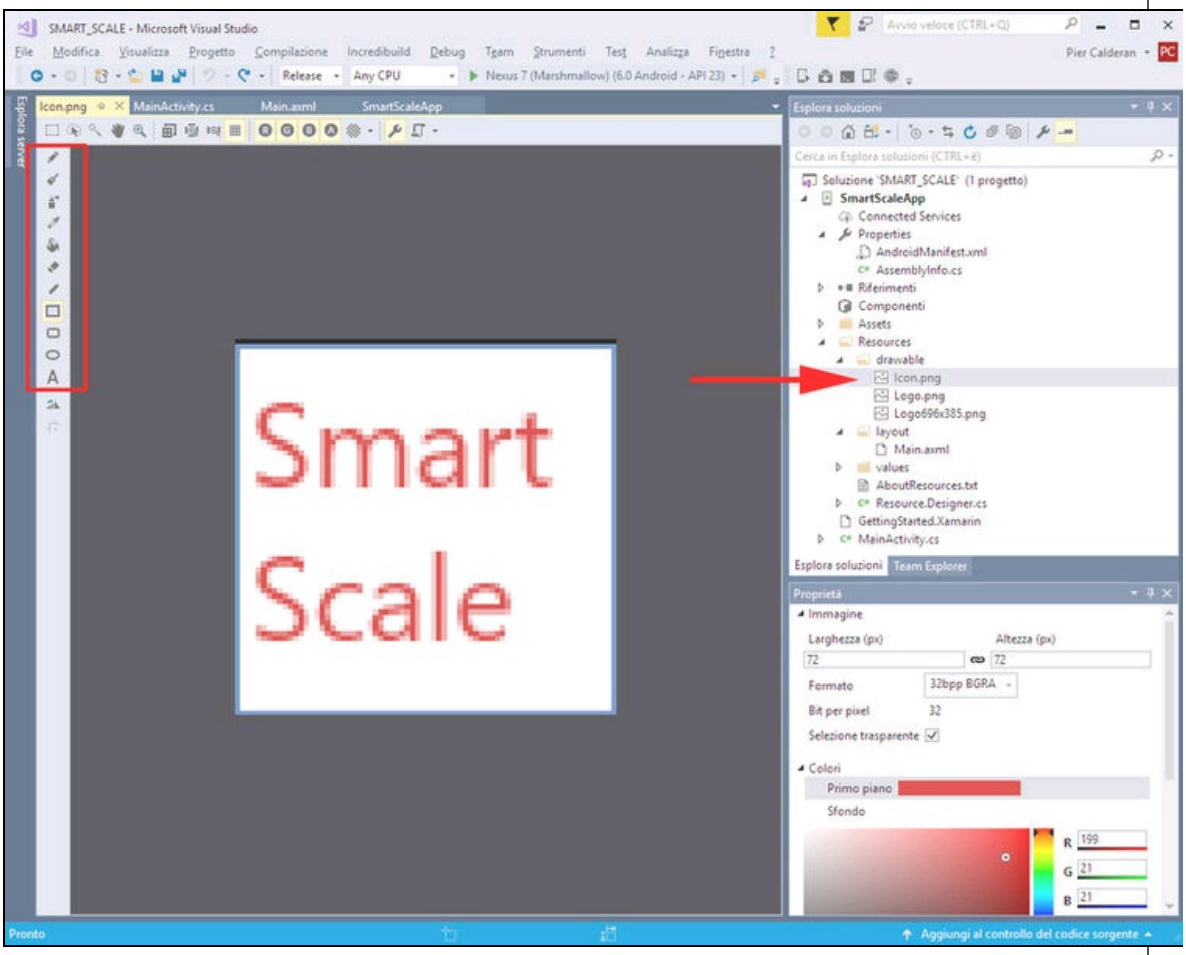
Figura 15.8 Finestra della scheda Applicazione (a) e della scheda Manifesto Android (b). Configurazione della soluzione (c).

Come creare l'icona dell'applicazione

Per creare o modificare l'icona di default dell'applicazione basta aprire l'editor grafico con un doppio clic sul nome dell'icona `Icon.png` che si trova all'interno della cartella `drawable`, come indicato nella figura.

Con gli strumenti messi a disposizione nella finestra dell'editor si possono creare sfondi, inserire forme geometriche, disegnare a mano libera, inserire testi, scegliere i colori degli strumenti di riempimento e di disegno e così via.

Nella figura è visibile l'icona creata per l'applicazione Smart Scale, che verrà visualizzata fra le app una volta installata nello smartphone.



Distribuzione

Per distribuire l'applicazione, cioè per esportare il pacchetto APK e poterlo installare sullo smartphone Android, impiegare la procedura seguente.

1. Dalla finestra *Esplora soluzioni* aprire con il tasto destro del mouse il menu contestuale e selezionare la voce *Archivia* (Figura 15.9a).
2. Si aprirà una finestra simile a quella rappresentata nella Figura 15.9b con una barra di progresso che indica lo stato della creazione del pacchetto *SmartScaleApp*.
3. Al termine della creazione del pacchetto, fare clic sul pulsante *Distribuisci*.

4. Si aprirà una finestra in cui selezionare il canale di distribuzione, ovvero *Ad hoc* oppure *Google Play* (Figura 15.9c) Non dovendo distribuire l'app su Google Play, selezionare il canale *Ad hoc*.
5. Si aprirà una finestra in cui selezionare una firma digitale dell'app, per esempio, la stessa che è stata creata per il progetto "Apertura cancello da smartphone" del Capitolo 10. Se non è stata creata la chiave procedere nel seguente modo.Fare clic sul pulsante + (più) per aggiungere una firma (Figura 15.9d). Si aprirà una finestra (Figura 15.9e) in cui si può creare una chiave. Compilare i seguenti campi.
   - *Alias*: nome della chiave.
   - *Password*: una password di almeno 8 caratteri/numeri.
   - *Validità*: lasciare 30 anni o quel che si vuole.
   - Nei campi sotto la sezione *Immettere almeno uno dei valori seguenti*, immettere i dati relativi al nome del produttore, organizzazione, città e così via.
6. Fare clic sul pulsante *Crea*. Si tornerà alla finestra precedente.
7. Selezionare il nome della chiave appena creata.
8. Selezionare il nome della chiave creata.
9. Fare clic sul pulsante *Salva con nome*.
10. Mentre una barra di progresso indica lo stato del salvataggio si aprirà una finestra che chiederà di inserire una password per la firma (quella scelta nel momento della creazione della chiave).

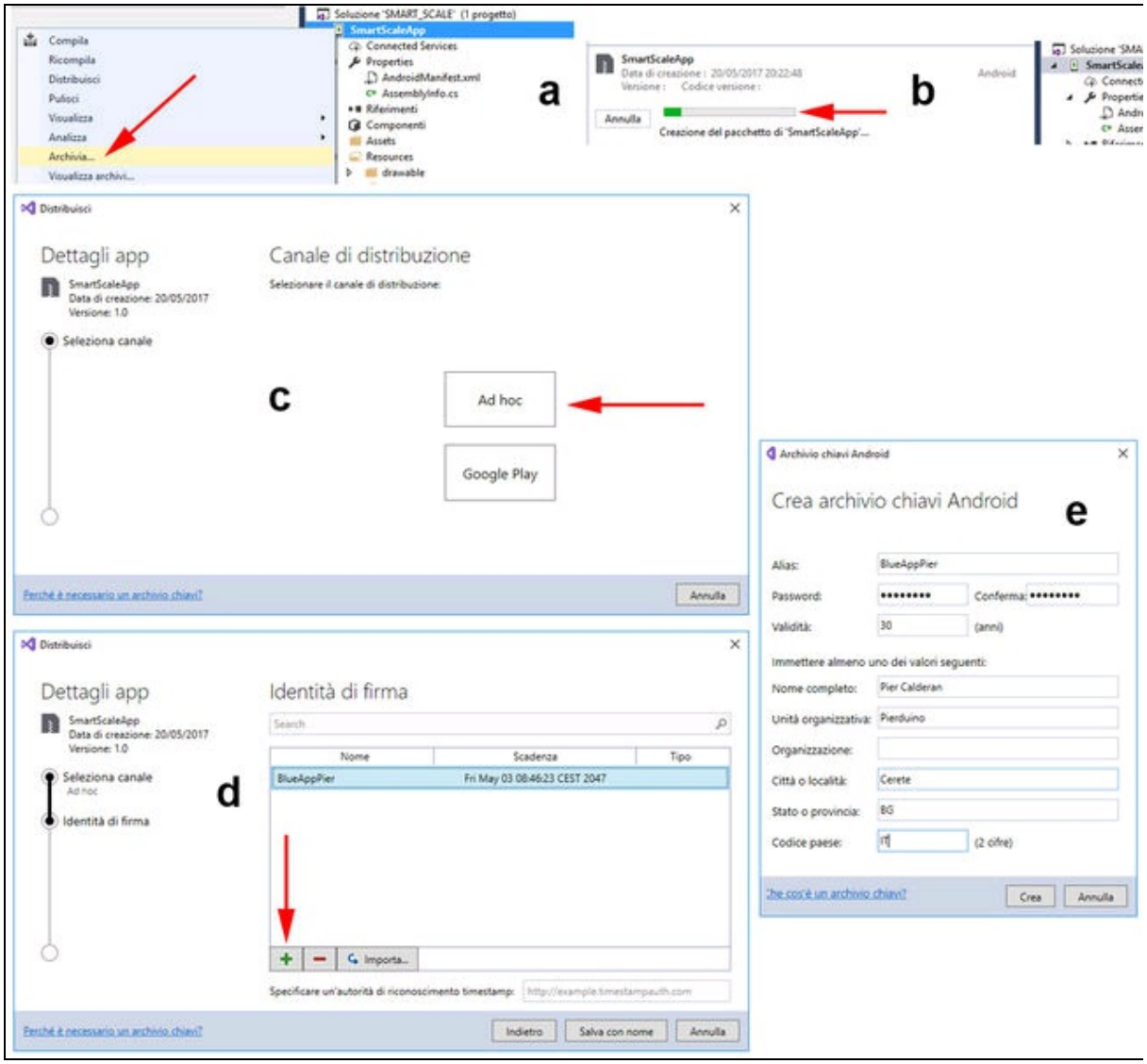


Figura 15.9 La voce di menu Archivia (a). Archiviazione dell'app (b). Finestra di distribuzione (c). Scelta della firma digitale (d). Creazione di una chiave (e).

Al termine delle operazioni, si aprirà una finestra di dialogo che chiederà dove salvare il pacchetto firmato. Il nome di default del pacchetto sarà `SmartScaleApp.SmartScaleApp.apk`, che ovviamente si può cambiare come si vuole.

Installazione dell'applicazione

Per installare il pacchetto APK appena creato bisogna trasferirlo nello smartphone. Se lo smartphone permette l'accesso alla card, basta copiare il file APK nella card e quindi avviare l'installazione dalla card. Se lo smartphone non consente l'accesso diretto alla card si può usare una delle tante applicazioni di trasferimento remoto. Una fra le tante è *Send Anywhere* (`https://send-anywhere.com`).

Una volta installata l'app nello smartphone, apparirà l'icona SmartScaleApp fra le applicazioni installate. Se si vuole cambiare l'icona di default, basta modificare il file `icon.png` contenuto nella cartella `Resources/drawable`, come spiegato nel box dedicato.

Associazione del modem Bluetooth

Per attivare una connessione fra uno smartphone e un'app Bluetooth, è necessario prima associare il modem, aprendo le impostazioni Bluetooth dello smartphone. Una volta alimentato il modem HC-05, si vedrà apparire "HC-05" fra i nomi dei dispositivi rilevati. Facendo clic sul nome HC-05, apparirà la schermata per l'immissione del codice di accesso, che di default è *1234*. Una volta associato il modem, qualsiasi app Bluetooth potrà connettersi.

Avvio dell'applicazione

Prima di avviare l'applicazione, ricordarsi sempre di associare il modem HC-05. La finestra dell'app sarà identica a quella disegnata nella finestra di progettazione (Figura 15.10a).

Toccando il tasto *CONNECT* verrà tentata la connessione Bluetooth e, se ha successo, apparirà una schermata simile a quella rappresentata nella Figura 15.10b con la scritta "Connected!".

A questo punto, si può impostare il peso ideale nella casella di testo *My target*. Toccando la casella di testo apparirà una tastiera numerica

per l’immissione del valore (Figura 15.10c). Digitare per esempio il valore 76.

Dopo essere saliti sulla bilancia, apparirà il peso in kg. Dopo che il valore si è stabilizzato, per esempio, su 80 kg, toccare il tasto *SPEAK*. Una voce pronuncerà la frase: “Il tuo peso è di chilogrammi 80. Sei in leggero sovrappeso”, come illustrato nella Figura 15.10d. Ci si può divertire a cambiare il peso target per ascoltare le altri frasi.

Siccome il peso rilevato viene memorizzato, per la lettura di un nuovo peso di riavviare l’applicazione e procedere a una nuova pesatura.

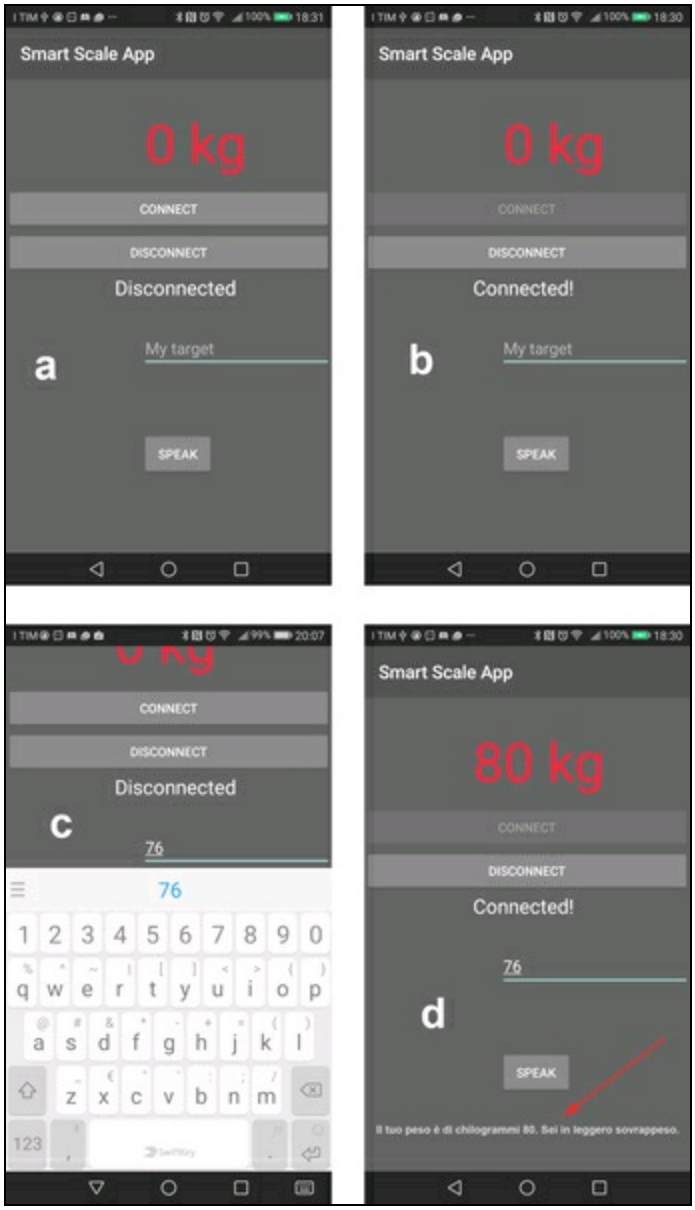


Figura 15.10 La finestra dell'app (a). Connessione Bluetooth avvenuta con successo (b). Tastierino numerico per inserire il valore del peso target (c). La frase pronunciata dalla voce sintetica (d).

Conclusioni

I progetti del libro finiscono qui. Speriamo di aver stimolato la fantasia del maker evoluto e di chi non lo è ancora ma vuole diventarlo.

Tutti progetti, per come sono stati pensati, hanno fondamentalmente uno scopo didattico e sono stati spiegati con uno spirito di sperimentazione e possono, anzi, devono essere migliorati. Non ci resta che augurare a tutti, buon lavoro e soprattutto buon divertimento!

Elementi di meccanica classica

Grandezze fondamentali

Le grandezze fondamentali della meccanica sono:

- spazio
- tempo
- massa
- forza

Spazio

Il concetto di spazio definisce la posizione di un punto P generico in un sistema di riferimento attraverso le sue coordinate nelle tre direzioni: x, y e z (Figura A.1). L'unità di misura è il metro: simbolo *m*.

Nella meccanica classica si usano spesso i suoi sottomultipli.

- Centimetro: cm (1×10^{-2} metri).
- Millimetro: mm (1×10^{-3} metri).
- Micrometro (micron): µm (1×10^{-6} metri).
- Nanometro: nm (1×10^{-9} metri).

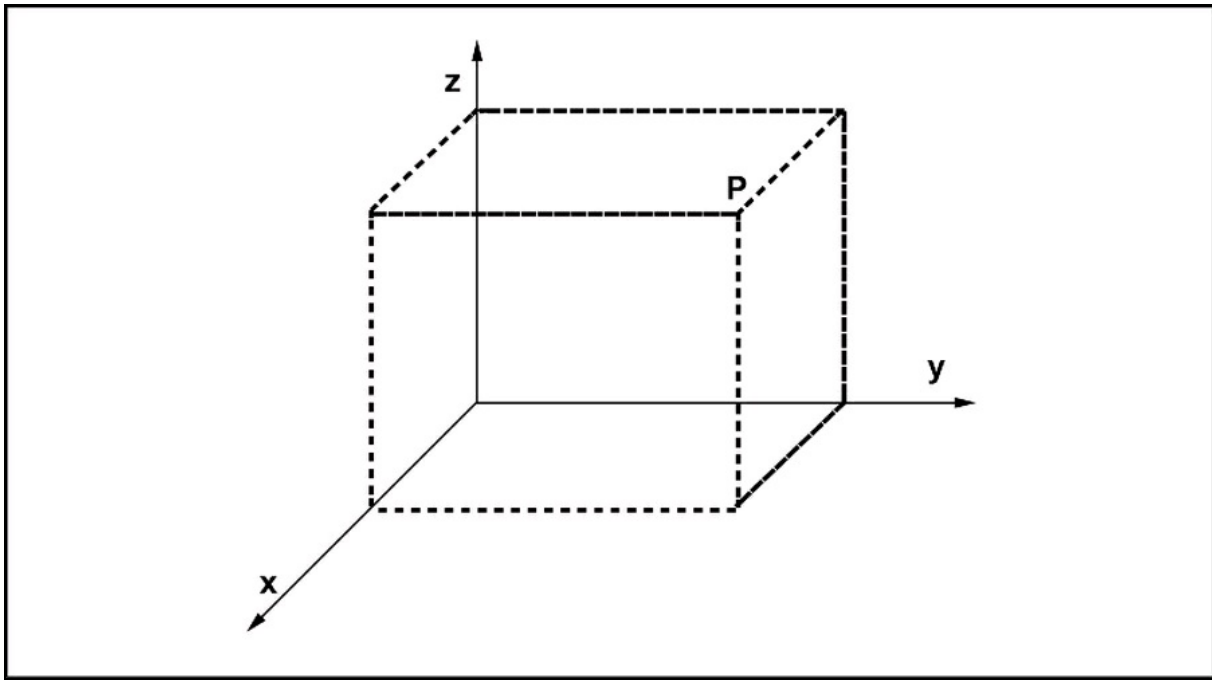


Figura A.1 Rappresentazione della posizione di un punto P nello spazio tridimensionale.

Tempo

Il concetto di tempo definisce una sequenza di eventi che si verificano in uno spazio fisico. La metafora più comune per evidenziare il trascorrere del tempo è il movimento del pendolo nel tempo di un secondo (Figura A.2). L'unità di misura è il secondo: simbolo *s*.

Nella meccanica classica si usano spesso i suoi sottomultipli.

- Millisecondo: ms (1×10^{-3} secondi).
- Microsecondo: µs (1×10^{-6} secondi).
- Nanosecondo: ns (1×10^{-9} secondi).

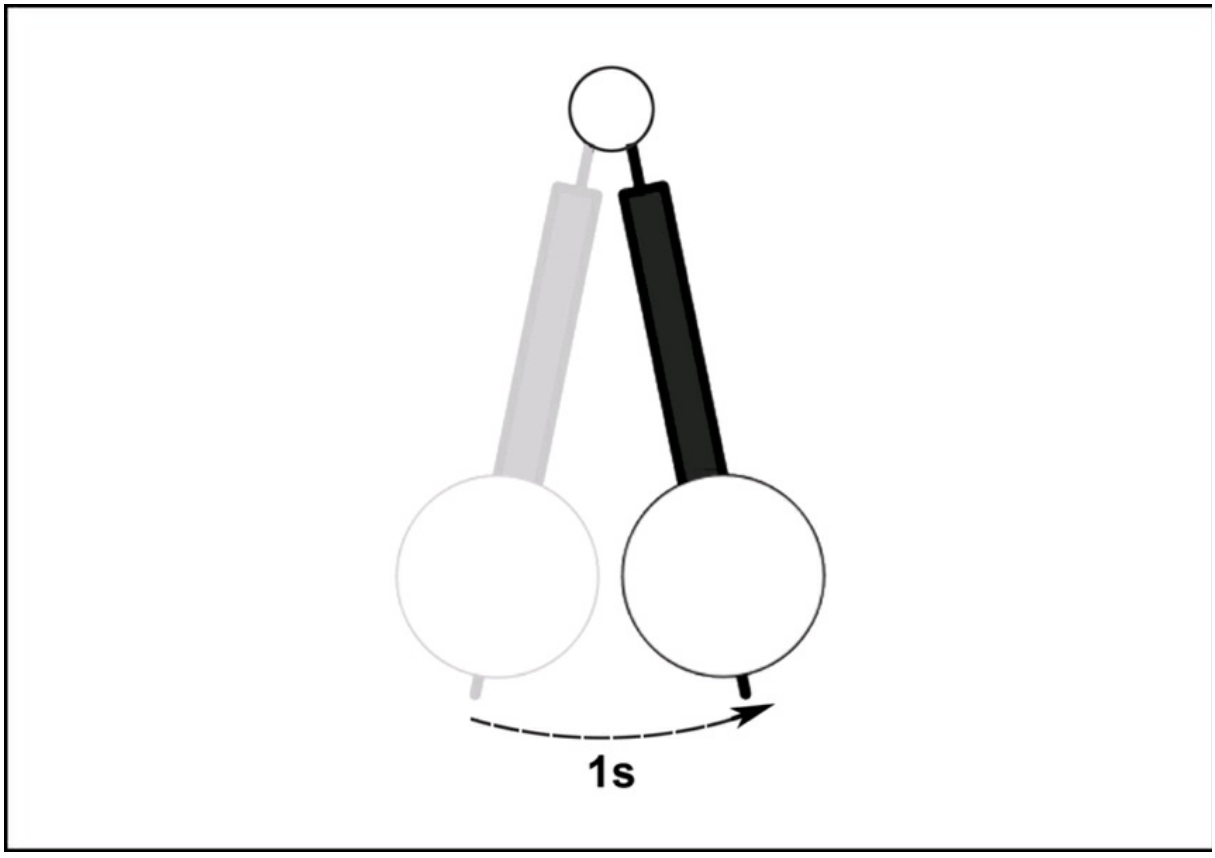


Figura A.2 Il concetto di tempo legato all’oscillazione di un pendolo.

Massa

Il concetto di massa è legato alla misura della quantità di materia presente in un corpo (erroneamente chiamato peso). È la grandezza fisica di un corpo che ne determina il comportamento dinamico quando il corpo è soggetto a una forza. La metafora più comune per evidenziare il concetto di massa sottoposta alla forza gravitazionale è il piano inclinato (Figura A.3). L’unità di misura della massa è il chilogrammo: simbolo *kg*.

Nella meccanica classica la massa può riferirsi a due grandezze fisiche.

- Massa inerziale: è la massa proporzionale all’inerzia di un corpo, quando viene applicata una forza per modificare un cambiamento di stato.
- Massa gravitazionale: è la forza di gravità, ovvero la massa proporzionale alla forza di interazione di un corpo con la forza gravitazionale.

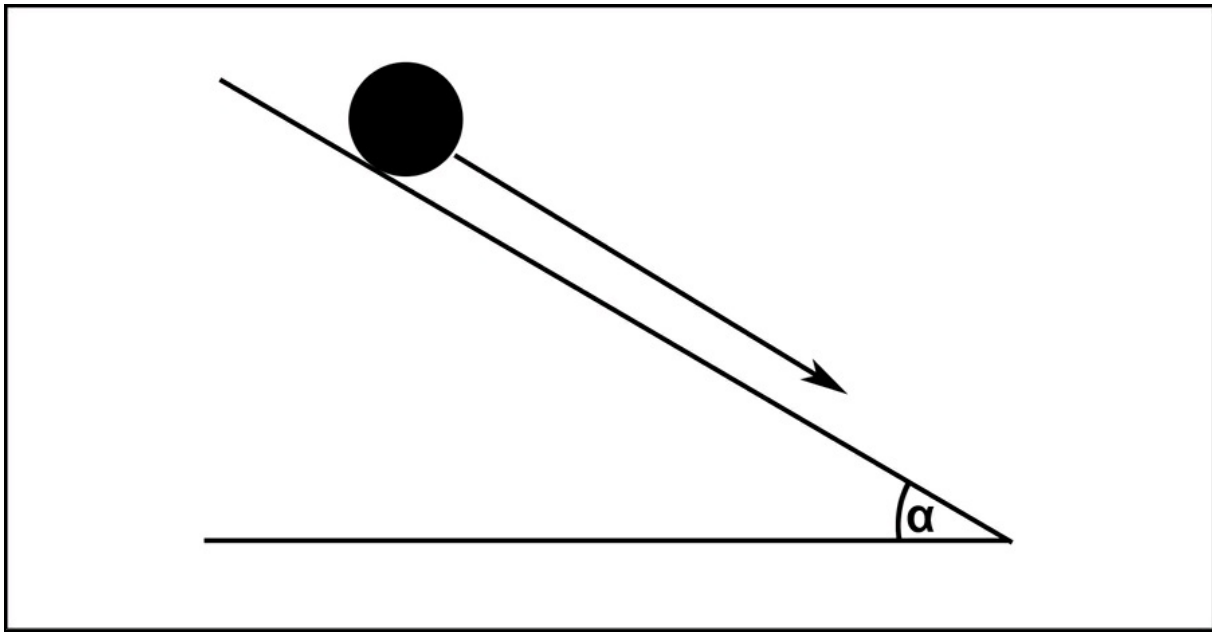


Figura A.3 Il concetto di massa legato al piano inclinato.

Forza

La forza è il concetto legato alla dinamica dei corpi, ovvero all’azione di un corpo su un altro per contatto diretto o indiretto. La metafora più comune è l’allungamento di una molla sotto l’azione di una forza peso, come nel caso del dinamometro, che è lo strumento per la misura della forza.

Secondo la legge di Hooke, lo spostamento dalla posizione di riposo di una molla è proporzionale alla forza applicata (Figura A.4):

- con *Fp* = 1 si ha 1 N e lo spostamento della molla di 1x;
- con *Fp* = 2 si ha 2 N e lo spostamento della molla di 2x;
- dove *Fp* è la forza peso, N è la misura in newton, x è la distanza dal punto di riposo.

Esistono due definizioni di forza.

- Definizione dinamica: per esempio la forza peso in grado di alterare lo stato di quiete o di moto di un corpo (piano inclinato, magnetismo, gravità e così via).
- Definizione statica: per esempio la forza in grado di produrre una deformazione di un corpo, come la forza applicata a una molla, a un materiale piezoelettrico o a una cella di carico.

L'unità di misura della forza è il newton: simbolo *N*.

Il newton è una grandezza esprimibile come la forza necessaria a imprimere alla massa di un chilogrammo l'accelerazione di un metro al secondo quadrato:

$$1\ N = (kg * m) / s^2$$

Il newton è anche l'unità di misura del peso, perché è la forza che agisce tra due corpi sottoposti alla forza di gravità. La massa di un chilogrammo subisce una forza peso di 9,81 newton.

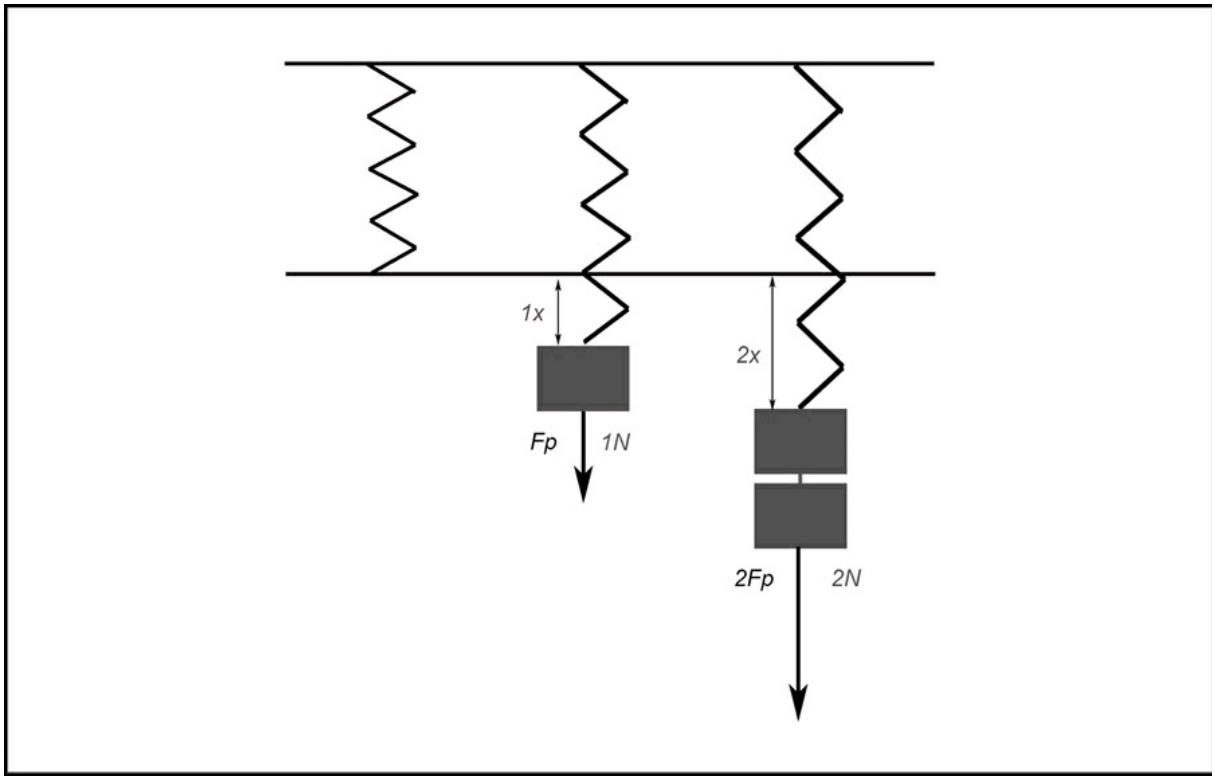


Figura A.4 La forza peso applicata a una molla secondo la legge di Hooke.

Indice