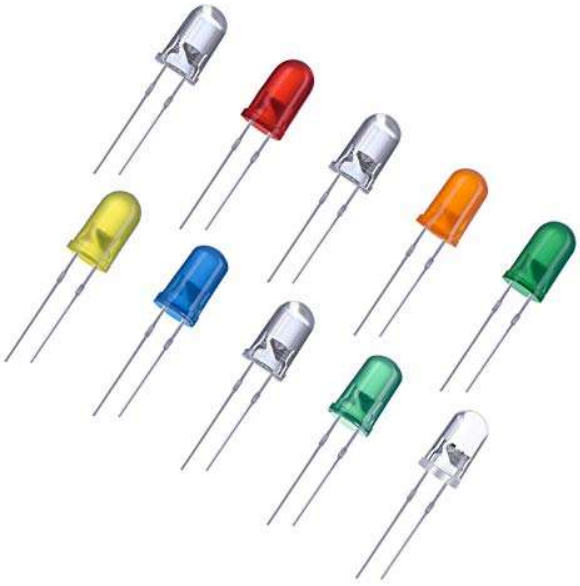
GUIDA ARDUINO DIGISPARK

Pulsante + Led



Mattia Ruberto & Matteo Ghilardini

SOMMARIO

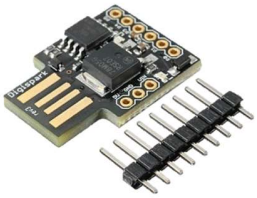
Scopo

Lo scopo di questa guida è illustrare il funzionamento del circuito in modo che sia facilmente comprensibile anche agli utenti più inesperti. Illustreremo perciò ogni componente utilizzato e il funzionamento di essi singolarmente, così anche per il prodotto globale.

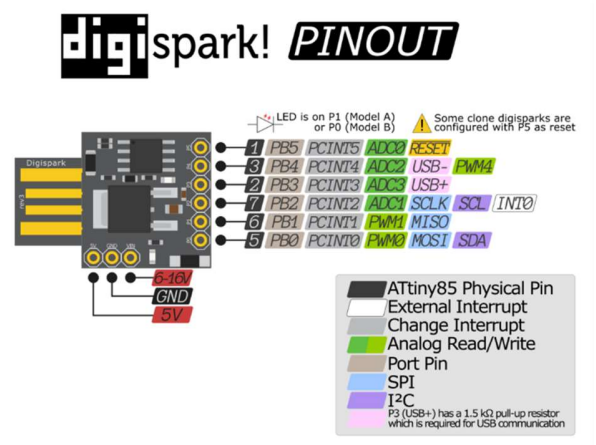
Componenti

Arduino Digispark

Arduino Digispark, così come tutti gli altri componenti della famiglia Arduino, è una scheda elettronica dotata di un microcontrollore. La funzionalità principale di Arduino è quella di realizzare in maniera pressoché semplice dei dispositivi di controllo oppure degli automatismi (specialmente nel caso di Arduino Digispark). Uno dei punti di forza di Arduino è la sua convenienza economica dal momento che le schede programmabili hanno prezzi veramente bassi (per Digispark meno di 5 CHF) e inoltre il software e il linguaggio di programmazione utilizzato sono Open Source (ossia gratis).



Per collegare elementi esterni alle schede si utilizzando dei pin che possono venir saldati sulle apposite interfacce. L'alimentazione (ossia il +) è indicata da "5V", mentre la terra (ossia il -) è indicata da "GND", mentre gli altri pin (da P0 a P5) possono assumere diverse funzionalità seguendo il seguente modello:



Per poter utilizzare il software di Arduino col Digispark sono necessari alcuni accorgimenti, per poter installare le schede è necessaria una connessione a internet (preferibilmente senza proxy):

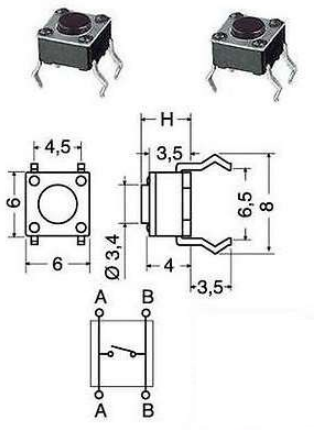
- Nelle impostazioni di arduino (File→Impostazioni), nel campo "URL aggiuntive per il Gestore schede:" inserire l'URL http://digistump.com/package_digistump_index.json;
- Riavviare il software (se procedendo qualcosa non va, riavviare il pc);
- Cliccando "Gestore schede" (Strumenti → Scheda) verrà aperta una schermata nella quale è presente una barra di ricerca, scriveteci "Digistump" e verrà mostrata una possibilità come quella da immagine:

nell'angolo in basso a detra di questa sarà presente il pulsante Installa (premerlo);

- Riavviare il software (se procedendo qualcosa non va, riavviare il pc);
- Nella selezione delle schede cercare e selezionare "Digispark (Default – 16.5 MHz)";
- Per quanto riguarda la selezione della porta (COM…) dipende dal vostro computer e da quale porta usb utilizzerete per inserire il digispark.

Pulsante

Un pulsante si comporta come se fosse un cavo che viene collegato e scollegato. La funzione corrispondente al fatto che è collegato, sarebbe quando viene premuto il pulsante, mentre quando viene rilasciato il circuito viene aperto (e quindi scollegato).



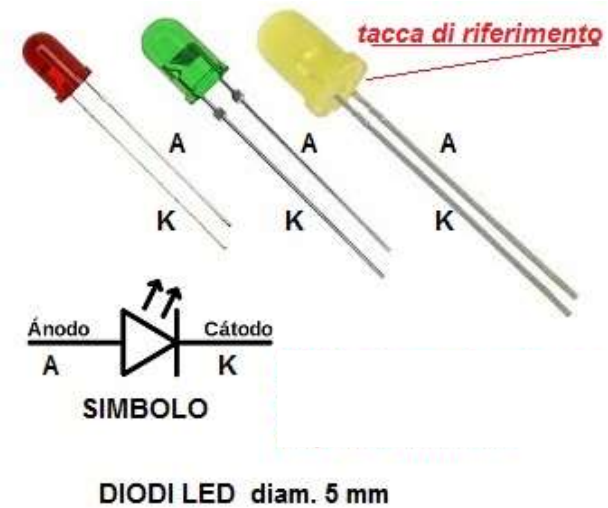
Esistono numerosi tipi diversi di pulsanti, ma quelli più comuni e più utilizzati sono quelli a 4 pin come quello mostrato nelle foto. I pin sono collegati a coppie e perciò per collegarli in modo da rilevare la pressione del pulsante bisogna seguire lo schema a sinistra (collegare un polo A, con un polo B).

I pulsanti non hanno nessun circuito sensibile al loro interno, quindi non è necessario prestare attenzione ai voltaggi che gli vengono impressi o alla loro polarità. Questo perché, come già detto, i pulsanti sono esattamente come se fossero due cavi che vengono collegati e scollegati a seconda del fatto che sia stato premuto o meno il pulsante.

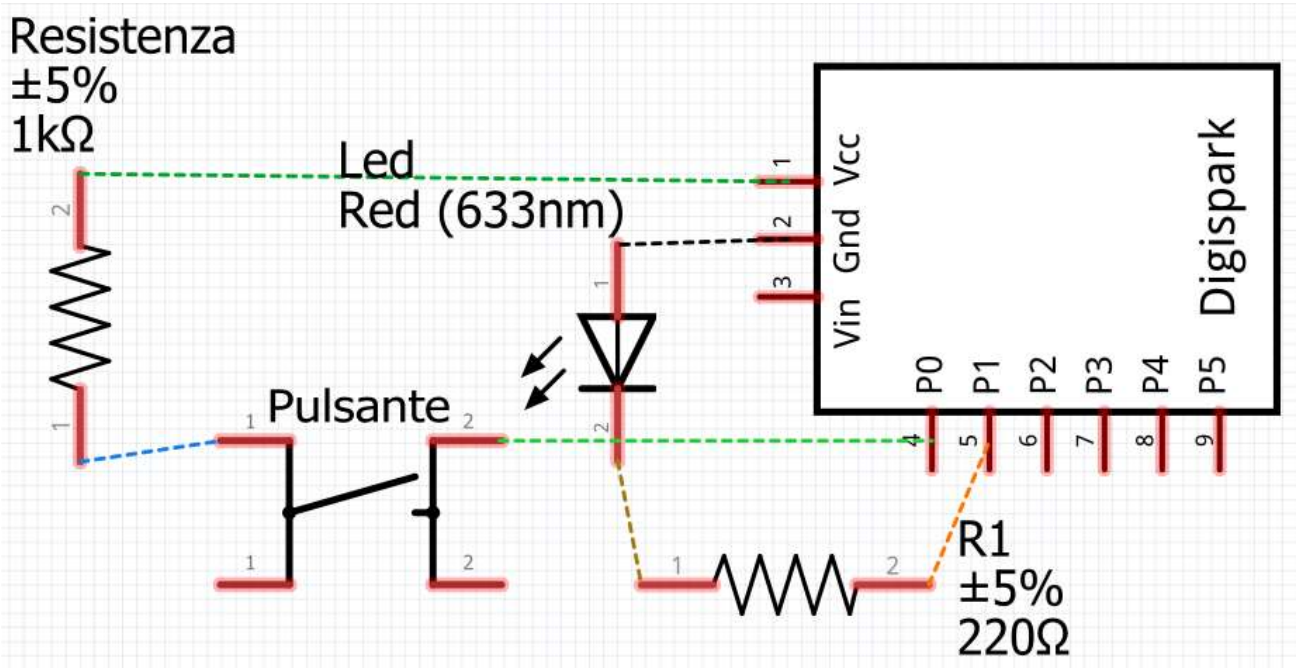
Led

Led non possono essere collegati direttamente al polo positivo o negativo della corrente perché subirebbero un voltaggio troppo alto rispetto a quello supportato, per questo dobbiamo utilizzare delle resistenze. Il minimo per il led che utilizziamo noi è una resistenza da 330 Ω.

Per distinguere il pin positivo e quello negativo è sufficiente guardare la lunghezza del suddetto pin e la posizione della tacca di riferimento (Vedi immagine a fianco). Il pin più lungo rappresenta il polo positivo, quindi il più corto quello negativo. Il polo positivo è identificabile anche dalla presenza della tacca.



Schema Elettrico



Librerie

Tutte le librerie realizzate per questo progetto sono state realizzate nel linguaggio di C++, come d'altronde anche il software di Arduino.

Per includere una libreria (o una cartella di librerie) dobbiamo spostarci nella cartella ".\Arduino\libraries" (se non la trovate, premete tasto destro sull'icona dell'editor di Arduino, quindi "Apri percorso File"), all'interno di questa cartella creiamo a sua volta una cartella chiamata come la libreria o come il componente al quale fa riferimento, all'interno di questa cartella, copiamo sia l'header che la libreria stessa.
Quando apriamo l'editor di Arduino, selezionare "Sketch", quindi "#include libreria", e ora scegliere la libreria desiderata (si chiamerà come la cartella che avete creato in precedenza).

Tutte le nostre librerie sono composte da un'interfaccia (chiamata nel linguaggio specifico di "C" *Header*, ed è un file con estensione ".h") e la libreria in sé (con estensione ".cpp") che estende l'interfaccia.

Sia l'Header, che la libreria devono includere l'interfaccia "Arduino.h".

Libreria Led

L'Header contiene:

- 2 attributi:
  - `led`: indica il pin del led;
  - `state_led`: indica lo stato del led (acceso / spento);
- 5 metodi:
  - `setLedPin(int ledPort);`
  - `powerOn();`
  - `powerOff();`
  - `setLed(bool stato_led);`
  - `blink(int frequency);`

La libreria contiene tutti i metodi dell'Header dichiarandoli in questo modo:

```
void <NomeHeader>::<metodo>(<tipoParametro> <parametro>){};
```

Ecco cosa fa nello specifico ogni metodo:

- **setLedPin**: in base al parametro che riceve, attribuisce il valore all'attributo che indica il pin del led;
- **powerOn**: setta il valore di state_led ad HIGH (che corrisponde a "1" o a "true"), a seconda della polarità del led, lo accende o lo spegne (se il contatto è al polo positivo, lo accende);
- **powerOff**: setta il valore di state_led a LOW (che corrisponde a "0" o a "false"), a seconda della polarità del led, lo accende o lo spegne (se il contatto è al polo positivo, lo spegne);
- **setLed**: rappresenta lo stato del led ricevuto come parametro (acceso o spento);
- **blink**: accende e spegne il led con un intervallo definito dal parametro "frequency";

Libreria Bottone (Pulsante)

L'Header contiene:

- 6 attributi:
  - `button`: indica il pin del bottone;
  - `state_button`: indica lo stato del bottone (premuto o meno);
  - `lastButtonState:` indica l'ultimo stato del bottone (necessario per l'anti-rimbalzo);
  - `lastDebounceTime`: memorizza i millisecondi da quando il bottone è stato premuto;
  - `debounceDelay`: indica il tempo per garantire l'anti-rimbalzo del bottone;
  - `ledState`: indica lo stato del led che potrebbe venir "toggleato";
- 5 metodi:
  - `setButtonPin(int buttonPort);`
  - `boolean getStateButton();`
  - `boolean toggle();`

La libreria contiene tutti i metodi dell'Header dichiarandoli in questo modo:

```
void <NomeHeader>::<metodo>(<tipoParametro> <parametro>){};
```

Ecco cosa fa nello specifico ogni metodo:

- **setButtonPin**: in base al parametro che riceve, attribuisce il valore all'attributo che indica il pin del bottone;
- **getStateButton**: ritorna il valore del bottone. A seconda della polarità cambia, ma un valore viene ritornato quando il bottone è premuto, mentre l'opposto quando non lo è;
- **toggle**: ritorna il valore di ledState invertito, il metodo contiene anche un controllo per l'anti-rimbalzo;

Utilizzo

Hardware

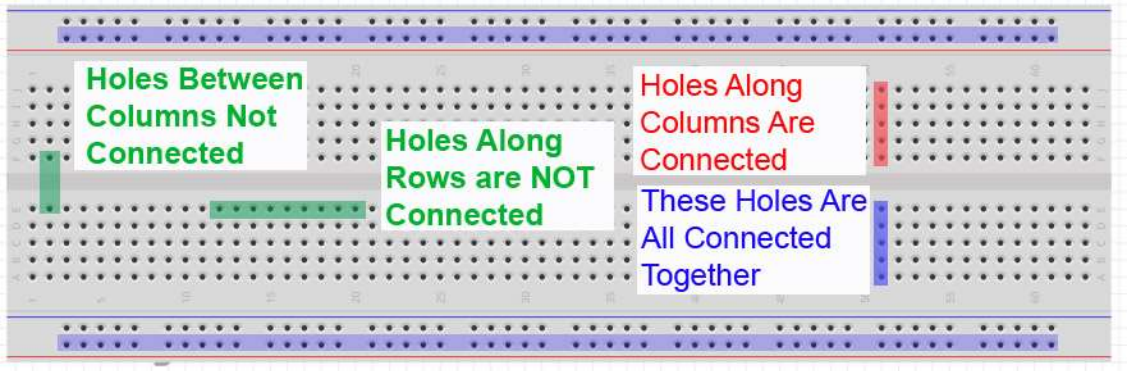
I componenti da utilizzare per questo progetto sono i 3 citati più e più volte all'interno di questa guida:

- Arduino Digispark;
- Pulsante;
- Led (di un qualunque colore);

Per costruire il circuito dobbiamo fissare il pulsante e il led su di una breadboard (circuito provvisorio) o su una veroboard (circuito definitivo), non è necessario metterli in uno schema preciso a patto che abbia un senso.

⚠ Fare attenzione alle piste delle board per evitare cortocircuiti ⚠

(In caso di dubbio, eccoti un'immagine che mostra com'è fatta una breadboard di Arduino al suo interno)



Per costruire il circuito dobbiamo fissare il pulsante in modo che le 2 coppie di pin non siano in contatto fra loro. Il led allo stesso modo può essere montato in qualunque modo a patto che i 2 pin non siano connessi.

Per gestire la corrente nel circuito abbiamo bisogno di 2 resistenze

- 220Ω per il led: deve essere collegata in serie fra il collegamento al Digispark (P1) e il polo positivo del led (quello più lungo).
- 10K Ω per il pull-Up o pull-Down del pulsante: deve essere collegata fra il pin del pulsante diagonalmente opposto a quello al quale è collegato il pin di lettura del Digispark (P0) e, a seconda del fatto che viene utilizzata per fare pull-Up o pull-Down, rispettivamente al +5V o al GND.

In caso di dubbio su come collegare il pulsante, consulta la documentazione del pulsante a pagina 4.

Software

- Il primo esempio di codice che abbiamo realizzato con questi componenti si occupa di far lampeggiare il led quando il bottone rimane premuto.

Le prime linee di codice servono ad includere le librerie necessarie al programma (solo quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente):

```
#include <LibraryLed.h>
#include <LibraryButton.h>
```

Le linee successive le utilizziamo per dichiarare le istanze delle nostre librerie:

```
LibraryLed libraryLed;
LibraryButton libraryButton;
```

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire al pin di scrittura del led e a quello di lettura del bottone un pin di Digispark:

```
void setup() {
     libraryLed.setLedPin(1);
     libraryButton.setButtonPin(0);
}
```

Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.
La prima cosa che facciamo nel loop è leggere lo stato del bottone memorizzarlo in una variabile:

```
boolean stato_bot = libraryButton.getStateButton();
```

L’ultima parte di codice controlla se il bottone è premuto o meno, in caso positivo il led lampeggierà, altrimenti rimarrà spento:

```
if (stato_bot == HIGH) {
     libraryLed.blink(600);
} else {
     libraryLed.powerOff();
}
```

Il secondo esempio che abbiamo pensato, inverte lo stato del led ogni qualvolta che il bottone viene premuto.

Le prime linee di codice servono ad includere le librerie necessarie al programma (solo quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente):

```
#include <LibraryLed.h>
#include <LibraryButton.h>
```

Le linee successive le utilizziamo per dichiarare le istanze delle nostre librerie:

```
LibraryLed libraryLed;
LibraryButton libraryButton;
```

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire al pin di scrittura del led e a quello di lettura del bottone un pin di Digispark:

```
void setup() {
     libraryLed.setLedPin(1);
     libraryButton.setButtonPin(0);
}
```

Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.
La prima cosa che facciamo nel loop è invertire una variabile chiamata stato_led tramite il metodo toggle:

```
boolean stato_led = libraryButton.toggle();
```

In fine rappresentiamo lo stato di tale variabile:

```
libraryLed.setLed(stato_led);
```

- Il terzo esempio che abbiamo pensato, quando il bottone viene premuto, accende il led per 1 secondo, se al termine di questo secondo il bottone è ancora premuto, il led comincia a lampeggiare con una frequenza di 20 millisecondi per 1500 millisecondi.

Le prime linee di codice servono ad includere le librerie necessarie al programma (solo quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente):

```
#include <LibraryLed.h>
#include <LibraryButton.h>
```

Le linee successive le utilizziamo per dichiarare le istanze delle nostre librerie:

```
LibraryLed libraryLed;
LibraryButton libraryButton;
```

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire al pin di scrittura del led e a quello di lettura del bottone un pin di Digispark:

```
void setup() {
     libraryLed.setLedPin(1);
     libraryButton.setButtonPin(0);
}
```

Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.
La prima cosa che facciamo nel loop è leggere lo stato del bottone:

```
boolean stato_bot = libraryButton.getStateButton();
```

In fine, se il bottone viene premuto, accendiamo il led per 1 secondo. Se al termine di questo secondo il bottone è ancora premuto, il led comincerà a lampeggiare per 1500 millisecondi con una frequenza di 20 millisecondi. Se invece non è premuto, il led si spegne:

```
if(stato_bot == HIGH){
     libraryLed.powerOn();
     delay(1000);
     stato_bot = libraryButton.getStateButton();
     if(stato_bot == HIGH){
          unsigned long currentMilles = millis();
          while((millis() - currentMilles) < 1500){
               libraryLed.blink(20);
          }
     }
}else{
     libraryLed.powerOff();
}
```