# GUIDA ARDUINO DIGISPARK

## Potenziometro + Led RGB


Mattia Ruberto & Matteo Ghilardini

# SOMMARIO

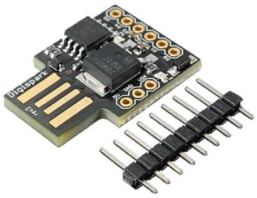
# Scopo

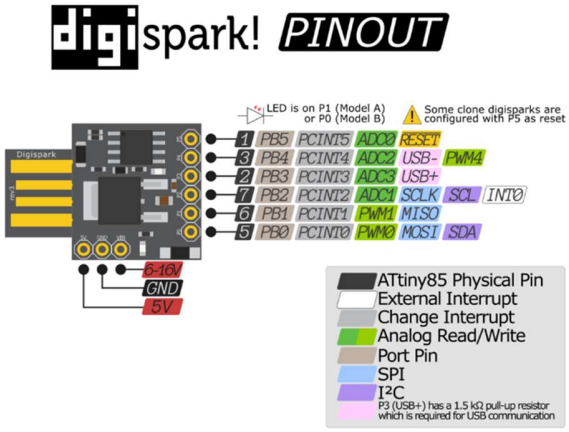
Lo scopo di questa guida è illustrare il funzionamento del circuito in modo che sia facilmente comprensibile anche agli utenti più inesperti. Illustreremo perciò ogni componente utilizzato e il funzionamento di essi singolarmente, così anche per il prodotto globale.

## 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):

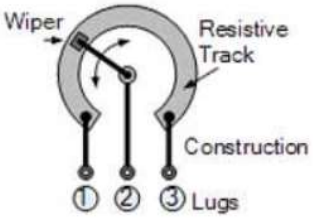
- Nelle impostazioni di arduino (File→Impostazioni), nel campo "URL aggiuntive per il Gestore schede:" inserire l'URL http://digistump.com/package_digistump_index.json;
- Riavviare il software (se procedendo qualcosa non va, riavviare il pc);
- Cliccando "Gestore schede" (Strumenti → Scheda) verrà aperta 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.

# Potenziometro

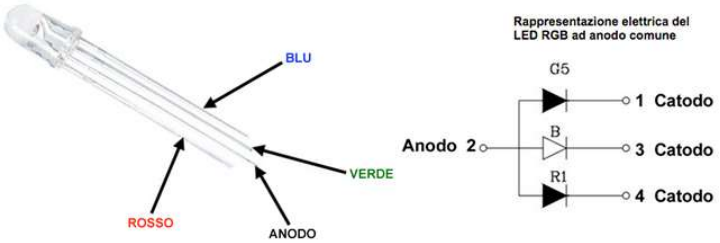
Il potenziometro è una sorta di resistenza che però può essere modificata, ossia può essere gestita la sua resistenza elettrica. Nell'elettronica "semplice" viene utilizzato per modificare delle frequenze o per modificare la luminosità dei led, mentre nell'elettronica di Arduino i suoi utilizzi aumentano a dismisura grazie ad un semplice comando utilizzato molto spesso: *map(value, fromLow, fromHigh, toLow, toHigh).* Tramite un analogRead() il potenziometro ritorna un valore fra 0 e 1023 perciò molte volte può essere utile magari rimappare l'intervallo fra magari 0 e 100 per ricevere la percentuale di rotazione (se utilizziamo un potenziometro rotativo), in questo caso usiamo il comando *map(analogRead(pinPotenziometro),0,1023,0,100)* e otterremo il valore percentuale di quanto è stato rotato il potenziometro (sempre se utilizziamo un potenziometro rotativo).




I potenziometri hanno un multiplo di 3 pin (il più comune e quello che usiamo anche noi che ne ha infatti 3) dove i 2 laterali vengono collegati al "+" e al "-", mentre quello centrale ritorna il valore desiderato. Come nelle resistenze normali, anche i potenziometri non hanno polarità perciò è indifferente quale dei pin esterni inseriamo nel "+" o rispettivamente nel "-", ma se non ritorna il valore che ci aspettiamo dovremo invertirne il senso.

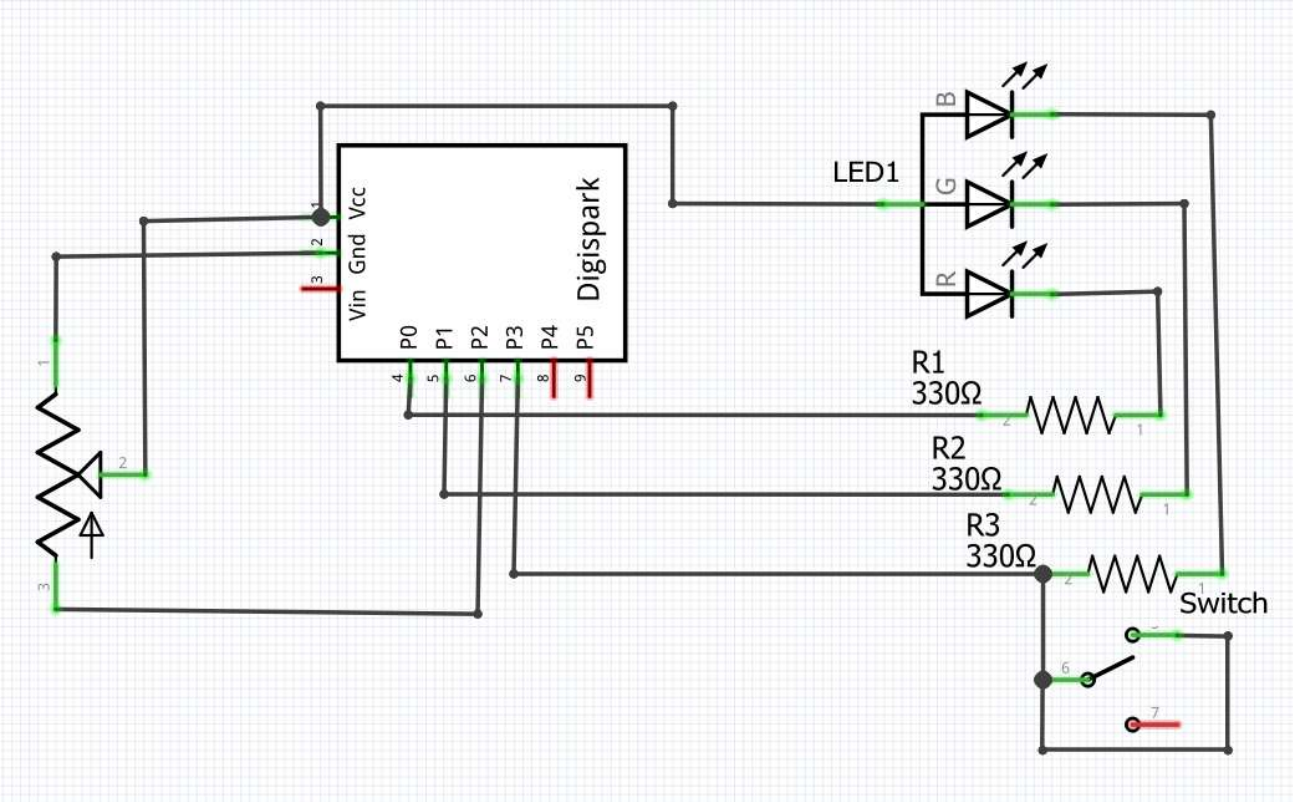
# Led RGB

Un led RGB è un solo led con al suo interno 3 led rispettivamente di colore rosso, verde e blu. I led (sia quelli RGB che quelli semplici) non possono essere collegati direttamente al polo positivo (o negativo, dipende se anodo o catodo comune. Vedi sotto) 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.




Esistono globalmente 2 tipi di led RGB, quelli con l'anodo ("+") comune o con catodo ("-") comune. Nel nostro caso utilizziamo un led anodo comune.

# 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 RGB

L'Header contiene:

- 3 attributi: uno per ogni pin corrispondente ad un colore diverso;
- 5 metodi:
  - `setLedPin(int myRedPin, int myGreenPin, int myBluePin);`
  - `setColor(int red, int green, int blue);`
  - `setRed(int red);`
  - `setGreen(int green);`
  - `setBlue(int blue);`

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 ai parametri che riceve, attribuisce tali valori agli attributi che indicano i pin dei colori;
- **setColor**: rappresenta il colore sui vari pin a seconda dei valori che riceve per ogni colore;
- **setRed, setGreen, setBlue**: modificano semplicemente solo il loro valore e lo rappresentano.

## Libreria Potenziometro

L'Header contiene:

- 2 metodi:
  - `int setRange(int valuePotentiometer, int valueMinPotentiometer, int valueMaxPotentiometer, int valueMin, int valueMax);`
  - `int getValue(int port);`

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

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

Ecco cosa fa nello specifico ogni metodo:

- **getMappedValue**: in base ai parametri che riceve, effettua una mappatura del valore del potenziometro (`valuePotentiometer`) dai valori originari (`valueMinPotentiometer` e `valueMaxPotentiometer`) ai nuovi valori scelti (`valueMin` e `valueMax`) e ritorna il valore mappato.
  Se il potenziometro è collegato direttamente al circuito (senza resistenze nel mezzo), i valori per `valueMinPotentiometer` e `valueMaxPotentiometer` sono rispettivamente 0 e 1023;
- **getValue**: riceve la porta analogica dalla quale leggere il valore del potenziometro e ritorna tale valore.

# Utilizzo

## Hardware

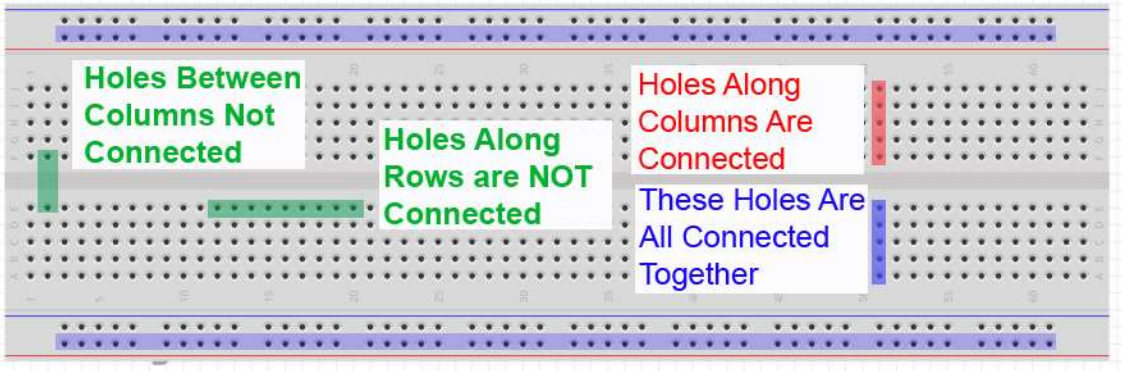
I componenti da utilizzare per questo progetto sono i 3 citati più e più volte all'interno di questa guida:

- Arduino Digispark;
- Potenziometro (rotativo);
- Led RGB anodo comune;

Per costruire il circuito dobbiamo fissare il potenziometro 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)***




Collegare i pin esterni del potenziometro ad alimentazione e GND (è indifferente quale ad uno o all'altro, nel caso in cui i valori non corrispondano a quelli desiderati è sufficiente invertirli), mentre il pin centrale del potenziometro deve essere collegato alla porta P2 del Digispark.

Collegare al pin più lungo del led RGB l'alimentazione (anodo in comune). A tutti gli altri pin colleghiamo delle resistenze da 330 Ω e in serie anche i connettori per portare il segnale alle corrispettive porte del Digispark, nello specifico:

- Il rosso alla porta P0;
- Il verde alla porta P1;
- Il blu alla porta P4;

In caso di dubbio su quale sia il pin corrispondente ad un determinato colore, consulta la documentazione del led RGB a pagina 4.

# Software

- Il primo esempio di codice che abbiamo realizzato con questi componenti si occupa di far scorrere una gamma di colori a seconda del valore del potenziometro, ossia, a seconda del valore del potenziometro, viene mostrato un colore diverso partendo dal rosso arrivando al viola passando da tutti i colori primari e secondari (rosso <-> giallo <-> verde <-> azzurro <-> blu <-> viola).

  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 <LibraryLedRGB.h>
#include <LibraryPotentiometer.h>
```

  Le linee successive le utilizziamo per dichiarare le variabili che ci serviranno più tardi tra queste anche l'istanza delle nostre librerie:

```
LibraryPotentiometer libraryPotentiometer;
LibraryLedRGB libraryLedRGB;

int valuePotentiometer;
int rangeValuePotentiometer;
```

  Nel metodo setup richiamiamo il metodo che abbiamo creato nella libreria che si occupa di attribuire ad ogni colore del led RGB un pin di Digispark:

```
void setup() {
    libraryLedRGB.setLedPin(0,1,4);
}
```

  Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.
  La prima cosa che facciamo nel loop è leggere il valore del potenziometro, memorizzarlo in una variabile e poi rimappare il valore di questa variabile passando da valori 0-1023, a valori 0-6:

```
valuePotentiometer = libraryPotentiometer.getValue(1);
rangeValuePotentiometer = libraryPotentiometer.getMappedValue(
    valuePotentiometer, 0, 1023, 1, 6
);
```

L'ultima parte di codice è una serie di IF che a seconda del valore del potenziometro, richiama il metodo setColor() della nostra libreria passandogli dei valori diversi a seconda del colore che dovrà essere rappresentato:

```
if(rangeValuePotentiometer > 0 && rangeValuePotentiometer <= 1){
     libraryLedRGB.setColor(255,0,0);
}else if(rangeValuePotentiometer > 1 && rangeValuePotentiometer <= 2){
     libraryLedRGB.setColor(255,255,0);
}else if(rangeValuePotentiometer > 2 && rangeValuePotentiometer <= 3){
     libraryLedRGB.setColor(0,255,0);
}else if(rangeValuePotentiometer > 3 && rangeValuePotentiometer <= 4){
     libraryLedRGB.setColor(0,255,255);
}else if(rangeValuePotentiometer > 4 && rangeValuePotentiometer <= 5){
     libraryLedRGB.setColor(0,0,255);
}else if(rangeValuePotentiometer > 5 && rangeValuePotentiometer <= 6){
     libraryLedRGB.setColor(255,0,255);
}
```

- Il secondo esempio che abbiamo pensato, scorre tutte le tonalità di ogni colore separatamente a seconda del valore del potenziometro. Ogni colore viene rappresentato per la durata di 1/3 di giro del potenziometro, all'interno di questo 1/3 vengono passati 255 valori che può assumere il colore.

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 <LibraryLedRGB.h>
#include <LibraryPotentiometer.h>
```

Le linee successive le utilizziamo per dichiarare le variabili che ci serviranno più tardi, tra queste anche l'istanza delle nostre librerie:

```
LibraryPotentiometer libraryPotentiometer;
LibraryLedRGB libraryLedRGB;

int valuePotentiometer;
int rangeValuePotentiometer;
```

Nel metodo setup richiamiamo il metodo che abbiamo creato nella libreria che si occupa di attribuire ad ogni colore del led RGB un pin di Digispark:

```
void setup() {
     libraryLedRGB.setLedPin(0,1,4);
}
```

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

```
valuePotentiometer = libraryPotentiometer.getValue(1);
```

L'ultima parte di codice è una serie di IF che a seconda del valore del potenziometro, mostra un colore diverso in tutte le sue tonalità (0 a 255 per ogni colore):

```
if(rangeValuePotentiometer < 256){
     libraryLedRGB.setRed(rangeValuePotentiometer);
     libraryLedRGB.setGreen(0);
     libraryLedRGB.setBlue(0);
}else if(rangeValuePotentiometer < 511){
     libraryLedRGB.setRed(0);
     libraryLedRGB.setGreen(rangeValuePotentiometer-255);
     libraryLedRGB.setBlue(0);
}else {
     libraryLedRGB.setRed(0);
     libraryLedRGB.setGreen(0);
     libraryLedRGB.setBlue(rangeValuePotentiometer-510);
}
```

- Il terzo esempio che abbiamo pensato, scorre tutte le tonalità di ogni colore separatamente a seconda del valore del potenziometro. Ogni volta che il potenziometro torna a 0 (o 255 a seconda della polarità) viene cambiato il colore da scorrere.

  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 <LibraryLedRGB.h>
#include <LibraryPotentiometer.h>
```

  Le linee successive le utilizziamo per dichiarare le variabili che ci serviranno più tardi, tra queste anche l'istanza delle nostre librerie:

```
LibraryPotentiometer libraryPotentiometer;
LibraryLedRGB libraryLedRGB;

int valuePotentiometer;
int rangeValuePotentiometer;
int counter = 0;
bool ceck = true;
```

  Nel metodo setup richiamiamo il metodo che abbiamo creato nella libreria che si occupa di attribuire ad ogni colore del led RGB un pin di Digispark:

```
void setup() {
     libraryLedRGB.setLedPin(0,1,4);
}
```

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

```
valuePotentiometer = libraryPotentiometer.getValue(1);
```

A questo punto verifichiamo che il valore del potenziometro sia "valido" (il valore è rappresentabile), in caso positivo settiamo la variabile check a true:

```
if(valuePotentiometer > 50){
     ceck = true;
}
```

Se check è true e se il valore del potenziometro è inferiore a 30 (quello che consideriamo 0 per evitare eventuali problemi di hardware), passiamo al colore successivo. Se sono già stati passati tutti i colori, ricomincia il ciclo:

```
if(ceck){
     if(valuePotentiometer < 30){
          counter++;
          if(counter == 3){
               counter = 0;
          }
          ceck = false;
     }
}
```

A seconda del valore del contatore (corrispondente ad ogni colore), mostra il colore indicato con tonalità definita dal valore del potenziometro:

```
if(counter == 0){
     libraryLedRGB.setColor(valuePotentiometer,0,0);
}else if(counter == 1){
     libraryLedRGB.setColor(0,valuePotentiometer,0);
}else if(counter == 2){
     libraryLedRGB.setColor(0,0,valuePotentiometer);
}
```