

**GUIDA ARDUINO DIGISPARK**

**Potenziometro + Led RGB**



Mattia Ruberto & Matteo Ghilardini

SOMMARIO

[SOMMARIO 2](#_Toc534980475)

[Scopo 3](#_Toc534980476)

[Componenti 3](#_Toc534980477)

[Arduino Digispark 3](#_Toc534980478)

[Potenziometro 4](#_Toc534980479)

[Led RGB 4](#_Toc534980480)

[Schema logico 5](#_Toc534980481)

[Software & Libreria 5](#_Toc534980482)

[Utilizzo 6](#_Toc534980483)

[Hardware 6](#_Toc534980484)

[Software 7](#_Toc534980485)

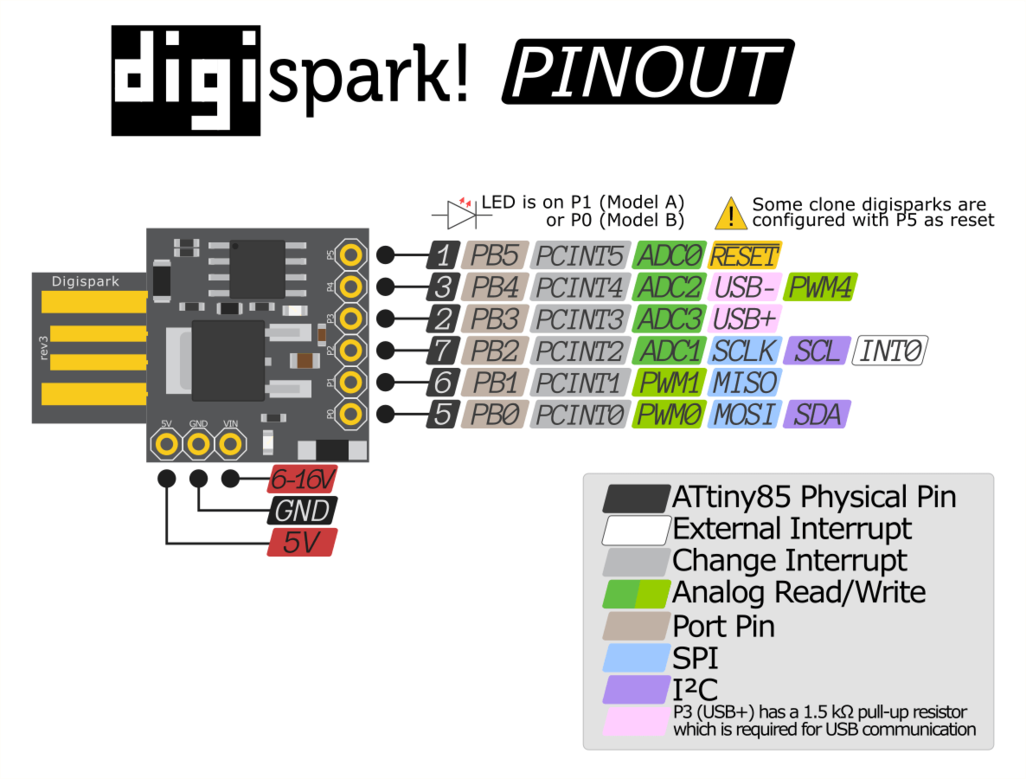
Scopo

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.

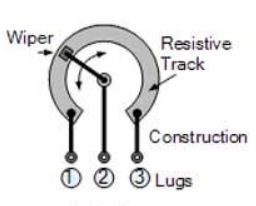
Componenti

## 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:

## Potenziometro

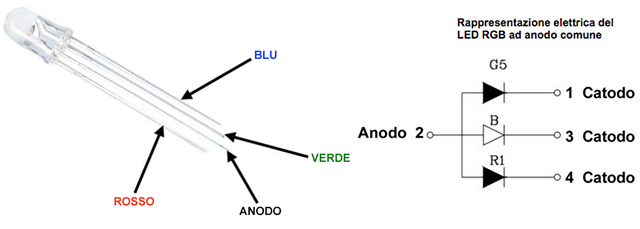
Il potenziometro è una sorta di resistenza che però può essere modificata, ossia può essere gestita la sua resistenza elettrica. Nell’elettronica “semplice” viene utilizzato per modificare delle frequenze o per modificare la luminosità dei led, mentre nell’elettronica di Arduino i suoi utilizzi aumentano 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.

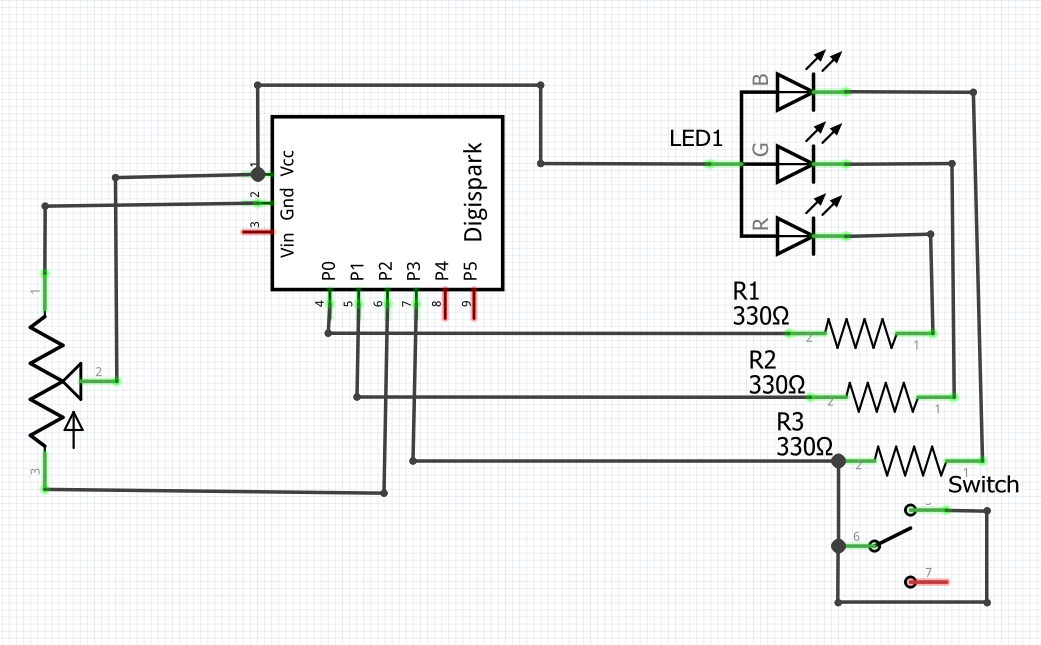
## Led RGB

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.*

La libreria realizzata per il led è composta da un’interfaccia (chiamata nel linguaggio specifico di “C” è chiamata Header ed è un file con estensione “.h”) e la libreria in sé che estende l’interfaccia.

L’interfaccia deve includere a sua volta l’interfaccia “Arduino.h”, in seguito troviamo la classe.

All’interno della classe abbiamo definito 3 attributi privati di tipo int che corrispondono ai vari pin del led RGB (uno per ogni colore), e 5 metodi, 3 dei quali che serviranno a settare il valore di ogni colore.

La libreria in sé è leggermente più complessa. All’inizio del codice dobbiamo anche in questo caso includere delle interfacce, in questo caso abbiamo sia quella base (“Arduino.h”), sia quella che abbiamo creato prima: “LedRGB.h”.

Ora parliamo dei metodi della libreria:

* **setLedPin(int, int, int)** richiede come parametri gli indici dei pin di ogni colore del led RGB. Al suo interno il metodo setta semplicemente i parametri passati con gli attributi della classe.
* **setColor(int, int, int)** richiede come parametri dei valori da 0 a 255 corrispondenti all’intensità del colore corrispondente. In seguito rappresenta il colore ricevuto sul led RGB.
* **setRed(int)** richiede come parametro il valore da 0 a 255 corrispondente all’intensità di rosso. In seguito rappresenta il valore ottenuto sul led RGB.
* **setGreen(int)** richiede come parametro il valore da 0 a 255 corrispondente all’intensità di verde. In seguito rappresenta il valore ottenuto sul led RGB.
* **setBlue(int)** richiede come parametro il valore da 0 a 255 corrispondente all’intensità di blu. In seguito rappresenta il valore ottenuto sul led RGB.

Per quanto riguarda la libreria per il potenziometro abbiamo concluso con il committente che non era necessario crearla dal momento che non sarebbe sensato creare un metodo contenete un metodo già esistente nel ambiente di Arduino.

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.

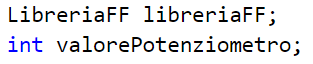
# Software

* 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).

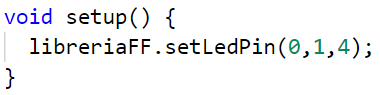
Le prime linee di codice servono ad includere le librerie necessarie al programma (quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente:



Le linee successive le utilizziamo per dichiarare le variabili che ci serviranno più tardi, tra queste anche l’istanza di LibreriaFF:

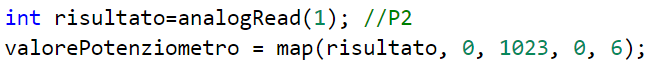


Nel metodo setup richiamiamo il metodo che abbiamo creato nella libreria che si occupa di attribuire ad ogni colore del led RGB un pin di Digispark:

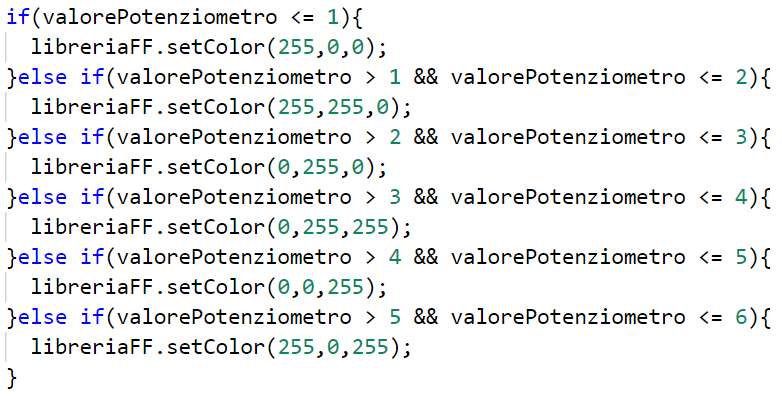


Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.

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:



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:

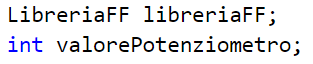


* Il secondo esempio che abbiamo pensato, scorre tutte le tonalità di ogni colore separatamente a seconda del valore del potenziometro. Ogni colore viene rappresentato per la durata di 1/3 di giro del potenziometro, all’interno di questo 1/3 vengono passati 255 valori che può assumere il colore.

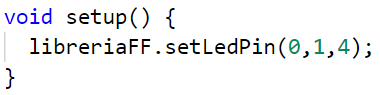
Le prime linee di codice servono ad includere le librerie necessarie al programma (quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente:



Le linee successive le utilizziamo per dichiarare le variabili che ci serviranno più tardi, tra queste anche l’istanza di LibreriaFF:



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:



Adesso