



## Sistema didattico per Arduino con libreria per attuatori e relativa documentazione

**Titolo del progetto:** Sistema didattico per Arduino con libreria per attuatori e relativa documentazione

**Alunno/a:** Matteo Ghilardini & Mattia Ruberto

**Classe:** Info 3AC

**Anno scolastico:** 2018/2019

**Docente responsabile:** Luca Muggiasca, Adriano Barchi, Francesco Mussi, Massimo Sartori

1	Introduzione.....	3
1.1	Informazioni sul progetto .....	3
1.2	Abstract .....	3
1.3	Scopo .....	3
1.4	Analisi del dominio .....	3
1.5	Analisi e specifica dei requisiti .....	4
1.6	Pianificazione .....	8
1.7	Analisi dei mezzi.....	9
1.7.1	Software .....	9
1.7.2	Hardware.....	9
2	Progettazione (Modulo Potenziometro e Led RGB).....	10
2.1	Schema Elettrico .....	10
2.2	Schema BreadBoard.....	10
3	Implementazione .....	11
3.1	Librerie .....	11
3.1.1	Libreria Led .....	11
3.1.2	Libreria Led RGB .....	11
3.1.3	Libreria Buzzer (Cicalino).....	12
3.1.4	Libreria Potenziometro .....	12
3.1.5	Libreria Bottone (Pulsante) .....	12
3.1.6	Libreria UltraSound (Sensore a Ultrasuoni).....	13
3.2	Modulo Potenziometro e led RGB.....	14
Hardware .....	14	
Software .....	15	
4	Test.....	19
4.1	Protocollo di test.....	19
4.2	Risultati test.....	27
4.3	Mancanze/limitazioni conosciute.....	27
5	Consuntivo.....	28
6	Conclusioni .....	29
6.1	Considerazioni personali.....	29
7	Bibliografia.....	29
7.1	Sitografia .....	29
8	Allegati.....	29

## 1 Introduzione

---

### 1.1 Informazioni sul progetto

Questo progetto è stato assegnato dai docenti Luca Muggiasca, Adriano Barchi, Francesco Mussi e Massimo Sartori il 14.11.2018 a tutti gli studenti di entrambi le classi 3<sup>E</sup> della SAM di Trevano (sezione Informatica). Il termine di consegna è fissato per il termine della lezione del 08.02.2018.

### 1.2 Abstract

In this document there is the documentation where the following will be explained how to use Arduino USB mini Digispark. There is a bookshelf, for each modulus steering will be an explain. These bookshelves are simple to use for beginners. The circuit are soldered on plates and via the connectors you can connect it directly to the Digispark. There are about five modules.

### 1.3 Scopo

Lo scopo di questo progetto è realizzare delle librerie per ogni attuatore e per ogni “reattore” presenti nel set di articolo di Arduino. Ogni libreria deve poter essere utilizzata da chiunque possieda competenze minime per quanto riguarda la conoscenza della programmazione in C (il linguaggio Arduino è basato su C++). Precisamente, il nostro lavoro verrà utilizzato per le giornate di prom-teck della scuola, pertanto dobbiamo considerare che le nostre librerie verranno utilizzate da ragazzi delle medie che probabilmente non sapranno nemmeno cosa voglia dire programmare.

### 1.4 Analisi del dominio

Le nostre librerie dovranno avere l’ambito di utilizzo il più vasto possibile dal momento che per ogni attuatore e per ogni reattore dovranno essere create 3 librerie distinte in modo che il committente possa scegliere quella che ritiene migliore.

Per eseguire i test delle nostre librerie eseguiremo dei test su 5 moduli diversi che saranno composti da attuatori e reattori diversi.

Ogni modulo ci verrà commissionato dopo che avremo terminato il precedente. Per ogni modulo dovremo a sua volta realizzare una guida per l’utente finale.

Ogni guida dovrà essere facilmente comprensibile da tutti gli utenti che potrebbero avere bisogno di consultarla, perciò deve contenere sia elementi che siano sufficienti per gli utenti più inesperti, ma anche elementi che possano soddisfare gli approfondimenti degli utenti più preparati.

**1.5 Analisi e specifica dei requisiti**

<b>ID: REQ-01</b>	
<b>Nome</b>	Libreria led RGB
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
001	Realizzare la libreria per il led RGB.
002	Implementare metodo che permette di settare le porte dei led.
003	Implementare metodo che permette accendere il led inserendo i tre valori dei tre colori.
004	Implementare metodo che permette di setta la tonalità di rosso.
005	Implementare metodo che permette di setta la tonalità di verde.
006	Implementare metodo che permette di setta la tonalità di blu.

<b>ID: REQ-02</b>	
<b>Nome</b>	Libreria potenziometro
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
001	Realizzare la libreria per il potenziometro.
002	Implementare metodo che ritorna il valore del potenziometro.
003	Implementare metodo che ritorna il valore nel range desiderato.

ID: REQ-03	
<b>Nome</b>	Libreria bottone
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
001	Realizzare la libreria per il bottone.
002	Implementare metodo che setta la porta del bottone.
003	Implementare metodo che ritorna lo stato del bottone.
004	Implementare metodo che effettuare l'anti-rimbalzo che ritorna lo stato del bottone.

ID: REQ-04	
<b>Nome</b>	Libreria led
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
001	Realizzare la libreria per il led.
002	Implementare metodo che setta la porta del led.
003	Implementare metodo che accende il led.
004	Implementare metodo che spegne il led.
005	Implementare metodo che setta il led con lo stato che gli viene passato.
006	Implementare metodo che fa lampeggiare il led con la frequenza che gli viene passata.

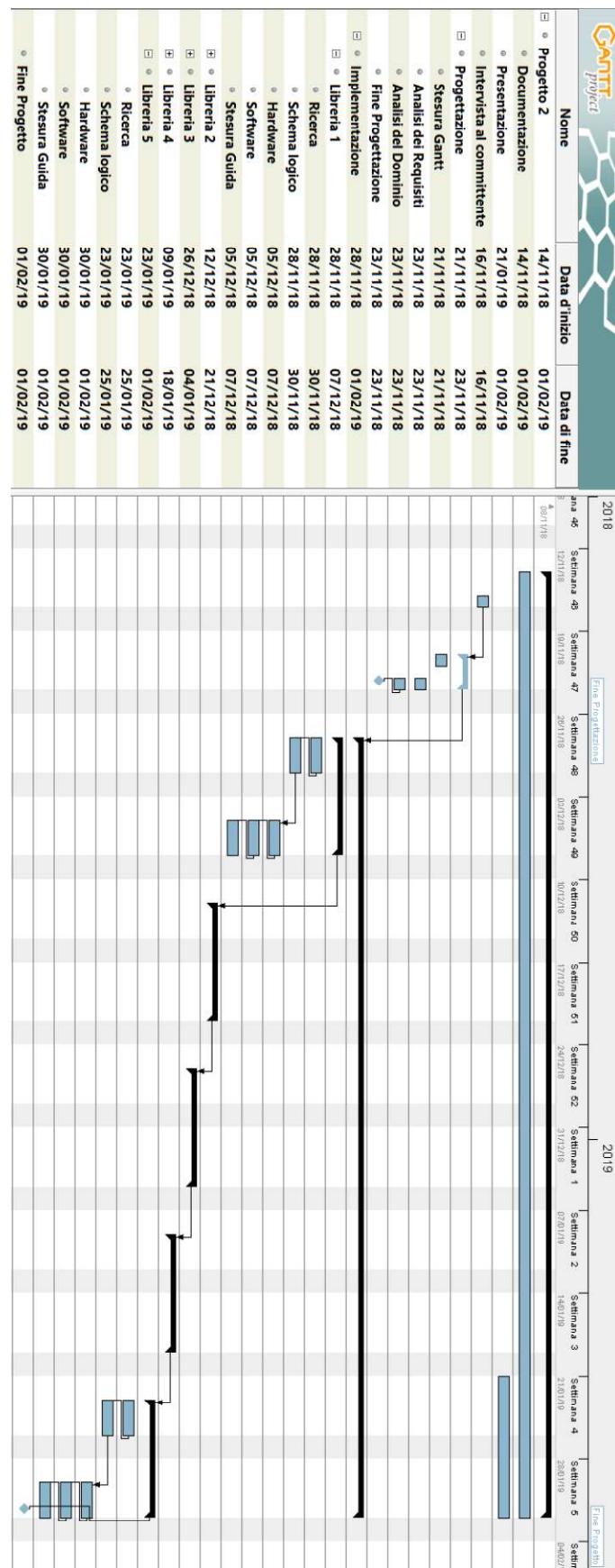
ID: REQ-05	
<b>Nome</b>	Libreria sensore ultrasuoni
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
001	Realizzare la libreria per il sensore ultrasuoni.
002	Implementare metodo che setta le porte del sensore ultrasuoni.
003	Implementare metodo che ritorna la distanza misurata dal sensore in cm.

ID: REQ-06	
<b>Nome</b>	Libreria sensore cicalino
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
001	Realizzare la libreria per il cicalino.
002	Implementare metodo che setta le porte del cicalino.
003	Implementare metodo che fa suonare il cicalino.
004	Implementare metodo che spegne il cicalino.

ID: REQ-07	
<b>Nome</b>	Realizzare tre attuatori.
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Il nostro progetto è composto da tre attuatori, led RGB e potenziometro, led e bottone, cicalino e sensore ultrasuoni, non ho fatto una tabella per attuatore perché ci sembrava ripetitivo.
<b>Sotto requisiti</b>	
001	Realizzare primo esempio, molto semplice.
002	Realizzare primo esempio, di un livello intermedio.
003	Realizzare primo esempio, di un livello superiore.

ID: REQ-08	
<b>Nome</b>	Realizzare una guida
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	Spiegare come funziona, cosa fare per iniziare a usarlo e come funziona.
<b>002</b>	Spiegare come utilizzare ogni singola libreria.
<b>003</b>	Spiegare come funziona ogni singolo metodo delle librerie.
<b>004</b>	Spiegare cosa fa, come lo fa e come farlo per ogni esempio fatto.

## 1.6 Pianificazione



Il progetto ci è stato commissionato il 14 novembre 2018 e programmiamo di completarlo entro il 1° febbraio 2019.

Alla prima lezione utile (venerdì 16/11/2018) svolgeremo l'intervista al committente procurandoci tutte le risposte alle domande che potrebbero sorgerci durante lo svolgimento del progetto.

La settimana successiva sarà adibita alla progettazione svolgendo la varie analisi, sia quella dei requisiti, sia quella del dominio (in seguito verranno inserite entrambe nella presente documentazione).

Il 28 novembre prgrammiamo già di iniziare l'implementazione eseguendo il primo modulo (con le relative librerie). Il lavoro di ogni modulo dovrebbe durare 2 settimane (per quanto riguarda "Libreria " 2, 3 e 4, abbiamo deciso di nasconderle per facilitare la lettura del gantt visto che comunque possiedono le stesse fasi delle 2 "Librerie" mostrate).

In parallelo alla "Libreria 5" (ultime 2 settimane) dobbiamo realizzare la presentazione del Progetto.

Durante tutto il corso del progetto, con il lavoro "Pratico" (di implementazione), svolgeremo in parallelo anche il lavoro legato alla presente Documentazione Di Progetto.

## 1.7 Analisi dei mezzi.

### 1.7.1 Software

- GanttProject [2.8.9]
- Microsoft Office 2016
- Github (Web e Desktop)
- Arduino
- Fritzing

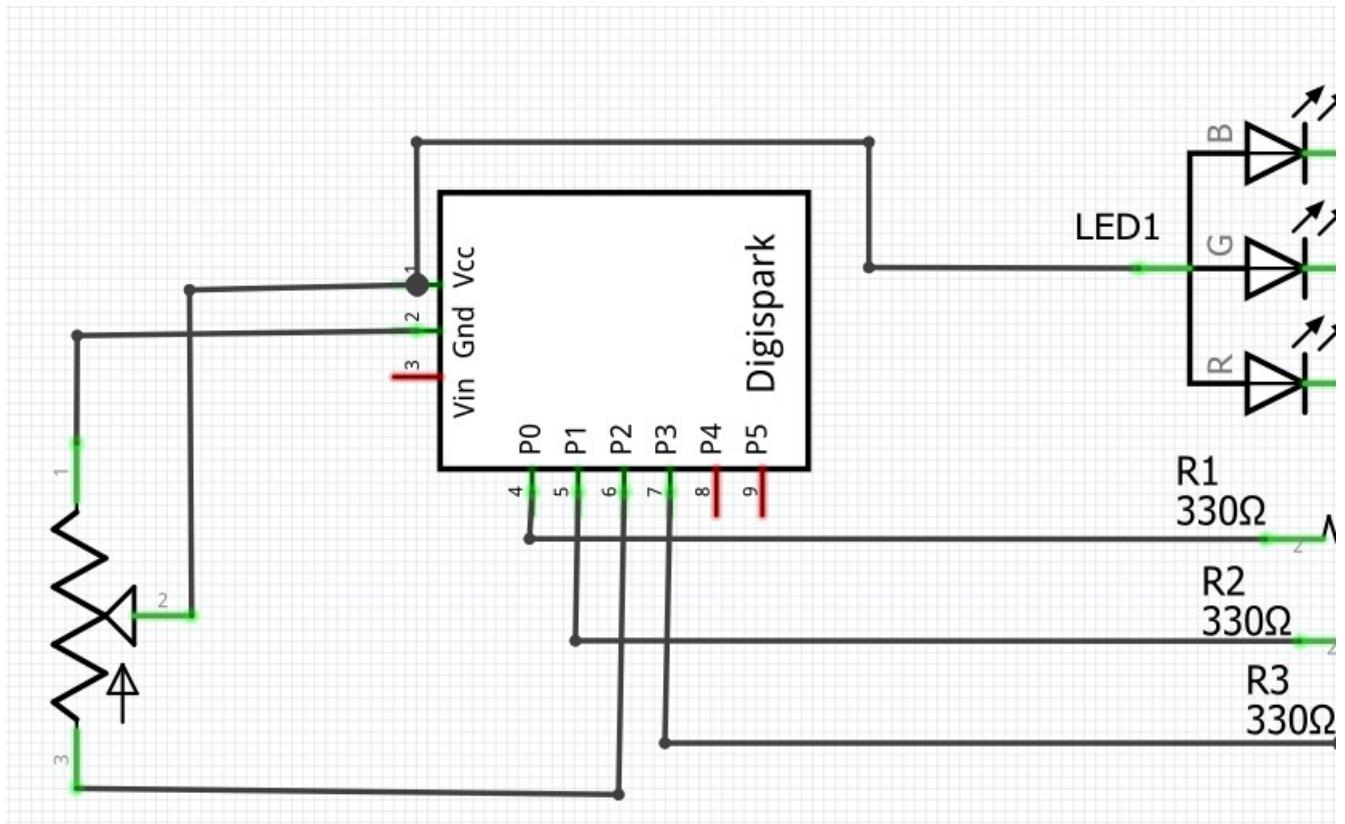
### 1.7.2 Hardware

- Computer personali:
  - Matteo Ghilardini, ASUS ZenBook Pro UX501VW:
    - Processore: Intel® Core™ i7-6700HQ CPU 2.60GHz
    - RAM: 16.0 GB DDR4
    - Sistema: 64bit, processore x64
  - Ruberto, ASUS ROGUE:
    - Processore: Intel® Core™ i7-7700HQ CPU 2.80GHz
    - RAM: 16.0 GB DDR4
    - Sistema: 64bit, processore x64
- Arduino Digispark;
- Attuatori Elettronici:
  - Potenziometro;
  - Pulsante;
  - Sensore a Ultrasuoni;
- Reattori Elettronici:
  - Led RGB;
  - Led;
  - Cicalino ("Buzzer");
- Altri componenti Elettronici:
  - Resistenze;
  - Cavi per il collegamento dei componenti;

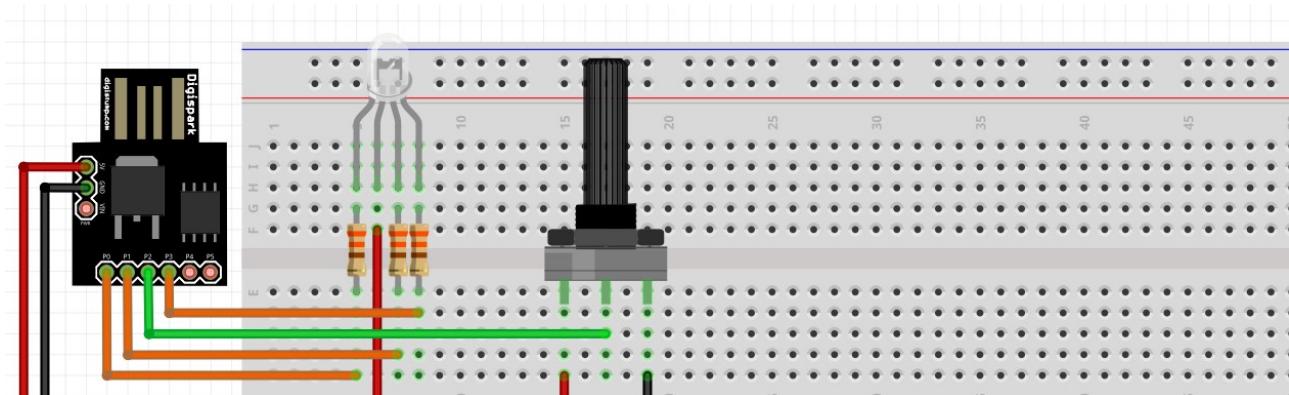
## 2 Progettazione (Modulo Potenziometro e Led RGB)

*Presentiamo un solo modulo per evitare ripetizioni.  
Se si desidera approfondire, consultare le guide allegate.*

### 2.1 Schema Elettrico



### 2.2 Schema BreadBoard



### 3 Implementazione

#### 3.1 Librerie

Tutte le librerie di Arduino soddisfano dei requisiti comuni:

- Ogni libreria deve possedere un relativo Header: l'Header ha la medesima funzione di un'interfaccia, ossia definisce quali attributi e quali metodi dovranno essere presenti nella libreria;
- Sia l'Header che la libreria estendono a loro volta l'Header base di Arduino "Arduino.h";
- Sia le librerie che gli Header sono realizzati con il linguaggio C++, ma gli Header vengono definiti in file con estensione ".h", mentre le librerie ".cpp";

##### 3.1.1 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";

##### 3.1.2 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.

### 3.1.3 Libreria Buzzer (Cicalino)

L'Header contiene:

- ◆ 1 attributo: corrisponde al pin al quale è collegato il cicalino;
- ◆ 3 metodi:
  - `setPinBuzzer(int buzzerPort);`
  - `setTone(int frequency);`
  - `powerOff();`

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

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

Ecco cosa fa nello specifico ogni metodo:

- ◆ **setPinBuzzer**: in base al parametro che riceve, attribuisce tale valore all'attributo che indica il pin del cicalino;
- ◆ **setTone**: fa emettere al cicalino dei suoni a seconda della frequenza ricevuta come parametro;
- ◆ **powerOff**: spegne il cicalino.

### 3.1.4 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.

### 3.1.5 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 "toggled";
- ◆ 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;

### 3.1.6 Libreria UltraSound (Sensore a Ultrasuoni)

L'Header contiene:

- ◆ 2 attributi: corrispondenti ai pin di emissione e ricezione del sensore;
- ◆ 3 metodi:
  - `setUltraSoundPin(int pinTrigPort, int pinEchoPort);`
  - `getDistance();`

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

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

Ecco cosa fa nello specifico ogni metodo:

- ◆ **setUltraSoundPin**: in base ai parametri che riceve, attribuisce tali valori agli attributi che indicano i pin di emissione e ricezione del sensore;
- ◆ **getDistance**: ritorna la distanza misurata dal sensore in cm;

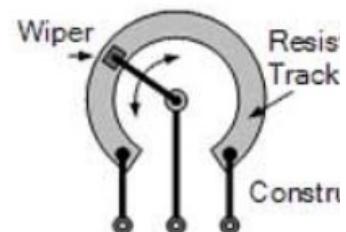
### 3.2 Modulo Potenziometro e led RGB

*Presentiamo un solo modulo per evitare ripetizioni.  
Se si desidera approfondire, consultare le guide allegate.*

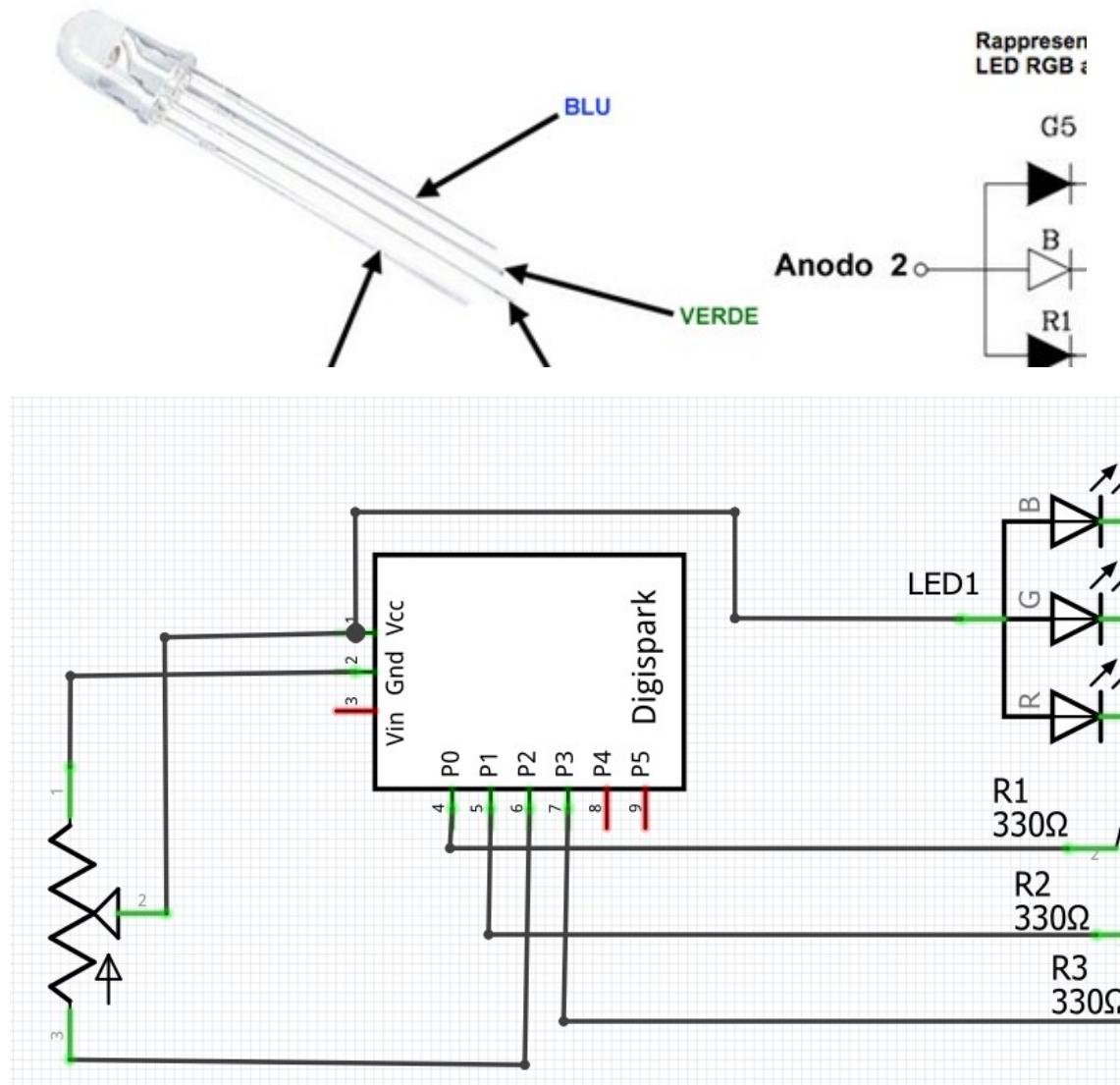
#### Hardware

Per realizzare questo modulo abbiamo utilizzato, oltre ai componenti specifici (potenziometro e led RGB), delle resistenze da  $330\Omega$  e dei cavi.

Il potenziometro può essere fissato sulla breadboard in una qualunque posizione a patto che abbia senso e che quindi tutti i pin si trovino su piste diverse. I pin esterni devono essere collegati uno al +5V e uno al GND (la polarità è indifferente, cambierà solo il senso di rotazione), il pin centrale deve invece essere collegato ad una porta che supporta la lettura analogica del Digispark, useremo P2.



Il led RGB è anodo comune, quindi il pin più lungo (l'anodo) deve essere collegato al +5V, gli altri pin possono essere collegati a qualunque porta in grado di eseguire una scrittura analogica (P0, P1, P4). Per identificare i colori relativi ad ogni pin del Led, vedi immagine sottostante.



## Software

### Esempio 1: Selezione del colore tramite serie di if leggendo il valore del potenziometro

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);  
 }
```

**Esempio 2:** Ogni colore viene rappresentato in ognuna delle sue tonalità per 1/3 del percorso del potenziometro

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);
}
```

**Esempio 3:** Scorrimento di tutte le tonalità di ogni colore; quando il valore del potenziometro torna a 0, si passa al colore successivo in ordine Red – Green – Blue.

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);  
}
```

## 4 Test

### 4.1 Protocollo di test

<b>Test Case:</b>	TC-001	<b>Nome:</b>	Verificare che la libreria del led RGB funzioni.
<b>Riferimento:</b>	REQ-01		
<b>Descrizione:</b>	Includere la libreria, compilare il codice e verificare che non dia errori.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"><li>• Aprire un nuovo file di Arduino</li><li>• Andare su "Sketch"</li><li>• Premere su "#includi libreria"</li><li>• Scegliere la nostra libreria, in questo caso LibraryRGB</li><li>• Compilare il file</li><li>• Istanziare LibraryLedRGB libraryLedRGB;</li><li>• Richiamare il metodo libraryLedRGB.setLedPin(0,1,4);</li><li>• Richiamare il metodo libraryLedRGB.setColor(255,255,255)</li><li>• Richiamare il metodo libraryLedRGB.setRed(255)</li><li>• Richiamare il metodo libraryLedRGB.setGreen(255)</li><li>• Richiamare il metodo libraryLedRGB.setBlue(255)</li></ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"><li>• La compilazione del file non deve dare errori.</li><li>• Se il metodo che setta i pin funziona il led RGB deve illuminarsi</li><li>• Se il metodo setColor(255,255,255) tutte e tre i colori del led devono illuminarsi al massimo</li><li>• Se il metodo setRed(255) funziona deve accendersi solo il rosso</li><li>• Se il metodo setGreen(255) funziona deve accendersi solo il verde</li><li>• Se il metodo setBlue(255) funziona deve accendersi solo il blue</li></ul>		

<b>Test Case:</b>	TC-002	<b>Nome:</b>	Verificare che la libreria del potenziometro funzioni.
<b>Riferimento:</b>	REQ-02		
<b>Descrizione:</b>	Includere la libreria, compilare il codice e verificare che non dia errori.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire un nuovo file di Arduino</li> <li>• Andare su “Sketch”</li> <li>• Premere su “#includi libreria”</li> <li>• Scegliere la nostra libreria, in questo caso LibraryPotentiometer</li> <li>• Compilare il file</li> <li>• Istanziare LibraryPotentiometer libraryPotentiometer;</li> <li>• Richiamare il metodo memorizzando il valore in int valuePotentiometer = libraryPotentiometer.getValue(1);</li> <li>• Richiamare il metodo memorizzando il valore in int rangeValuePotentiometer = libraryPotentiometer.getMappedValue(valuePotentiometer, 0, 1023, 0, 255);</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• La compilazione del file non deve dare errori.</li> <li>• Il metodo getValue(1) se il potenziometro è collegato senza resistenze ritorna un valore tra 0 e 1023 (Testato con il monitor serial dell'arduino Mega)</li> </ul>		

<b>Test Case:</b>	TC-003	<b>Nome:</b>	Verificare che la libreria del bottone funzioni.
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Includere la libreria, compilare il codice e verificare che non dia errori.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire un nuovo file di Arduino</li> <li>• Andare su “Sketch”</li> <li>• Premere su “#includi libreria”</li> <li>• Scegliere la nostra libreria, in questo caso LibraryButton</li> <li>• Compilare il file</li> <li>• Istanziare LibraryLed libraryLed</li> <li>• Richiamare il metodo libraryButton.setButtonPin(0)</li> <li>• Richiamare il metodo memorizzando il valore in boolean stato_bot = libraryButton.getStateButton()</li> <li>• Richiamare il metodo boolean state = libraryButton.toggle()</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• La compilazione del file non deve dare errori.</li> <li>• Se il metodo setButtonPin funziona sarà possibile utilizzare il bottone</li> <li>• Se il metodo getStateButton() funziona sarà possibile utilizzare lo stato che ritorna per settare il led, se il bottone è premuto il led si accende altrimenti si spegne.</li> <li>• Se il metodo toggle() funziona possiamo utilizzare lo stato che ritorna per settare il led, se il bottone è premuto il led cambia lo stato altrimenti niente.</li> </ul>		

<b>Test Case:</b>	TC-004	<b>Nome:</b>	Verificare che la libreria del led funzioni.
<b>Riferimento:</b>	REQ-04		
<b>Descrizione:</b>	Includere la libreria, compilare il codice e verificare che non dia errori.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"><li>• Aprire un nuovo file di Arduino</li><li>• Andare su “Sketch”</li><li>• Premere su “#includi libreria”</li><li>• Scegliere la nostra libreria, in questo caso LibraryLed</li><li>• Compilare il file</li><li>• Istanziare LibraryLed libraryLed;</li><li>• Richiamare il metodo setLedPin(1)</li><li>• Richiamare il metodo libraryLed.blink(600)</li><li>• Richiamare il metodo libraryLed.powerOff()</li><li>• Richiamare il metodo libraryLed.powerOn()</li><li>• Richiamare il metodo libraryLed.setLed(&lt;stato&gt;)</li></ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"><li>• La compilazione del file non deve dare errori.</li><li>• Se il metodo setLedPin funziona il led si illuminerà</li><li>• Se il metodo blink(600) funziona il led si accederà e spegnerà con un intermittenza di 600 millisecondi</li><li>• Se il metodo powerOn funziona il led si accende</li><li>• Se il metodo powerOff funziona il led si spegne</li></ul>		

<b>Test Case:</b>	TC-005	<b>Nome:</b>	Verificare che la libreria dell'ultrasuoni funzioni.
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Includere la libreria, compilare il codice e verificare che non dia errori.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire un nuovo file di Arduino</li> <li>• Andare su “Sketch”</li> <li>• Premere su “#includi libreria”</li> <li>• Scegliere la nostra libreria, in questo caso LibraryUltraSound</li> <li>• Compilare il file</li> <li>• Istanziare l’oggetto LibraryUltraSound libraryUltraSound;</li> <li>• Richiamare il metodo libraryUltraSound.setUltraSoundPin(1, 2)</li> <li>• Richiamare il metodo nel valore int distance = libraryUltraSound.getDistance()</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• La compilazione del file non deve dare errori.</li> <li>• Se il metodo setUltraSoundPin(1, 2) sarà possibile utilizzare il sensore ultrasuoni</li> <li>• Se il metodo getDistance funziona ritornerà la distanza misurata in cm (Testato con il monitor seriale dell’arduino Mega)</li> </ul>		

<b>Test Case:</b>	TC-006	<b>Nome:</b>	Verificare che la libreria del cicalino funzioni.
<b>Riferimento:</b>	REQ-06		
<b>Descrizione:</b>	Includere la libreria, compilare il codice e verificare che non dia errori.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire un nuovo file di Arduino</li> <li>• Andare su “Sketch”</li> <li>• Premere su “#includi libreria”</li> <li>• Scegliere la nostra libreria, in questo caso LibraryBuzzer</li> <li>• Compilare il file</li> <li>• Istanziare la libreria LibraryBuzzer libraryBuzzer;</li> <li>• Richiamare il metodo libraryBuzzer.setPinBuzzer(0)</li> <li>• Richiamare il metodo libraryBuzzer.setTone(100)</li> <li>• Richiamare il metodo libraryBuzzer.powerOff()</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• La compilazione del file non deve dare errori.</li> <li>• Se il metodo setPinBuzzer funziona il cicalino funzionerà</li> <li>• Se il metodo setTone(100) funziona il cicalino suonerà con una frequenza di 100</li> <li>• Se il metodo powerOff funzionerà il cicalino non deve suonare</li> </ul>		

<b>Test Case:</b>	TC-007	<b>Nome:</b>	Test sul primo esempio del modulo led RGB e potenziometro.		
<b>Riferimento:</b>	REQ-07	<b>Descrizione:</b> Testare il funzionamento del primo esempio del modulo led RGB e potenziometro.			
<b>Prerequisiti:</b>					
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il primo esempio</li> <li>• Staccare il pin 4</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Quando il caricamento del codice è completato riattaccare la porta 4</li> <li>• Ruotare il potenziometro più volte in entrambi i sensi.</li> </ul>				
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Il led si deve illuminare di sei colori diversi a dipendenza della posizione del potenziometro partendo dal rosso, giallo, verde, azzurro, blue e viola.</li> </ul>				

<b>Test Case:</b>	TC-008	<b>Nome:</b>	Test sul secondo esempio del modulo led RGB e potenziometro.		
<b>Riferimento:</b>	REQ-07	<b>Descrizione:</b> Testare il funzionamento del secondo esempio del modulo led RGB e potenziometro.			
<b>Prerequisiti:</b>					
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il secondo esempio</li> <li>• Staccare il pin 4</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Quando il caricamento del codice è completato riattaccare la porta 4</li> <li>• Ruotare il potenziometro più volte in entrambi i sensi.</li> </ul>				
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Il led durante il giro del potenziometro deve passare da tutte le tonalità di rosso, verde e blue, quando si torna in dietro deve fare l'opposto.</li> </ul>				

<b>Test Case:</b>	TC-009	<b>Nome:</b>	Test sul terzo esempio del modulo led RGB e potenziometro.		
<b>Riferimento:</b>	REQ-07	<b>Descrizione:</b> Testare il funzionamento del secondo esempio del modulo led RGB e potenziometro.			
<b>Prerequisiti:</b>					
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il terzo esempio</li> <li>• Staccare il pin 4</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Quando il caricamento del codice è completato riattaccare la porta 4</li> <li>• Ruotare il potenziometro più volte in entrambi i sensi.</li> </ul>				
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Il led deve passare tutte le tonalità di rosso, quando il potenziometro torna a zero il led deve diventare verde e passare tutte le tonalità e stessa cosa quando torna a zero deve cambiare la tonalità da verde a blu. Così via ricominciando dal rosso.</li> </ul>				

<b>Test Case:</b>	TC-010	<b>Nome:</b>	Test sul primo esempio del modulo led e bottone.		
<b>Riferimento:</b>	REQ-07	<b>Descrizione:</b> Testare il funzionamento del primo esempio del modulo led e bottone.			
<b>Prerequisiti:</b>					
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il primo esempio</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Schiacciare il bottone, rilasciarlo e tenerlo premuto.</li> </ul>				
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Quando il bottone è premuto il led deve lampeggiare, quando il bottone viene rilasciato il led deve spegnersi.</li> </ul>				

<b>Test Case:</b>	TC-011	<b>Nome:</b>	Test sul secondo esempio del modulo led e bottone.		
<b>Riferimento:</b>	REQ-07	<b>Descrizione:</b> Testare il funzionamento del secondo esempio del modulo led e bottone.			
<b>Prerequisiti:</b>					
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il secondo esempio</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Schiacciare il bottone, rilasciarlo e tenerlo premuto.</li> </ul>				
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Quando il bottone viene premuto il led deve cambiare lo stato nell'opposto dei quello che è adesso, se il bottone viene premuto senza mai rilasciarlo il led non deve cambiare fino a quando il bottone non viene rilasciato e ripremuto.</li> </ul>				

<b>Test Case:</b>	TC-012	<b>Nome:</b>	Test sul terzo esempio del modulo led e bottone.
<b>Riferimento:</b>	REQ-07		
<b>Descrizione:</b>	Testare il funzionamento del terzo esempio del modulo led e bottone.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il terzo esempio</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Schiacciare il bottone, rilasciarlo e tenerlo premuto.</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Il bottone quando viene schiacciato deve accendere il led, se il bottone resta premuto per più di un secondo il led deve incominciare a lampeggiare.</li> </ul>		

<b>Test Case:</b>	TC-013	<b>Nome:</b>	Test sul primo esempio del modulo cicalino e sensore ultrasuoni.
<b>Riferimento:</b>	REQ-07		
<b>Descrizione:</b>	Testare il funzionamento del primo esempio del modulo cicalino e sensore ultrasuoni.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il primo esempio</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Prendere un oggetto che permetta al sensore ultrasuoni di misurare la distanza, avvinarlo, allontanarlo e restare fermi.</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Questo esempio riprende i sensori delle macchine, se la distanza misurata è minore di 100cm il cicalino inizia a suonare, più si avvicina e più deve suonare veloce.</li> </ul>		

<b>Test Case:</b>	TC-014	<b>Nome:</b>	Test sul secondo esempio del modulo cicalino e sensore ultrasuoni.
<b>Riferimento:</b>	REQ-07		
<b>Descrizione:</b>	Testare il funzionamento del secondo esempio del modulo cicalino e sensore ultrasuoni.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ul style="list-style-type: none"> <li>• Aprire il primo esempio</li> <li>• Staccare l'arduino dalla porta USB</li> <li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li> <li>• Prendere un oggetto che permetta al sensore ultrasuoni di misurare la distanza, avvinarlo, allontanarlo e restare fermi.</li> </ul>		
<b>Risultati attesi:</b>	<ul style="list-style-type: none"> <li>• Se il sensore ultrasuoni misura la massima distanza il cicalino suona con una frequenza di 1000, più la distanza diminuisce, la frequenza diminuisce e il volume del suono diminuisce.</li> </ul>		

<b>Test Case:</b>	TC-015	<b>Nome:</b>	Test sul terzo esempio del modulo cicalino e sensore ultrasuoni.		
<b>Riferimento:</b>	REQ-07				
<b>Descrizione:</b>	Testare il funzionamento del terzo esempio del modulo cicalino e sensore ultrasuoni.				
<b>Prerequisiti:</b>					
<b>Procedura:</b>	<ul style="list-style-type: none"><li>• Aprire il primo esempio</li><li>• Staccare l'arduino dalla porta USB</li><li>• Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino</li><li>• Prendere un oggetto che permetta al sensore ultrasuoni di misurare la distanza, avvinarlo, allontanarlo e restare fermi</li></ul>				
<b>Risultati attesi:</b>	<ul style="list-style-type: none"><li>• Se il sensore ultrasuoni misura una distanza minore ai 20cm inizia a suonare con una frequenza stabile sempre uguale altrimenti si spegne.</li></ul>				

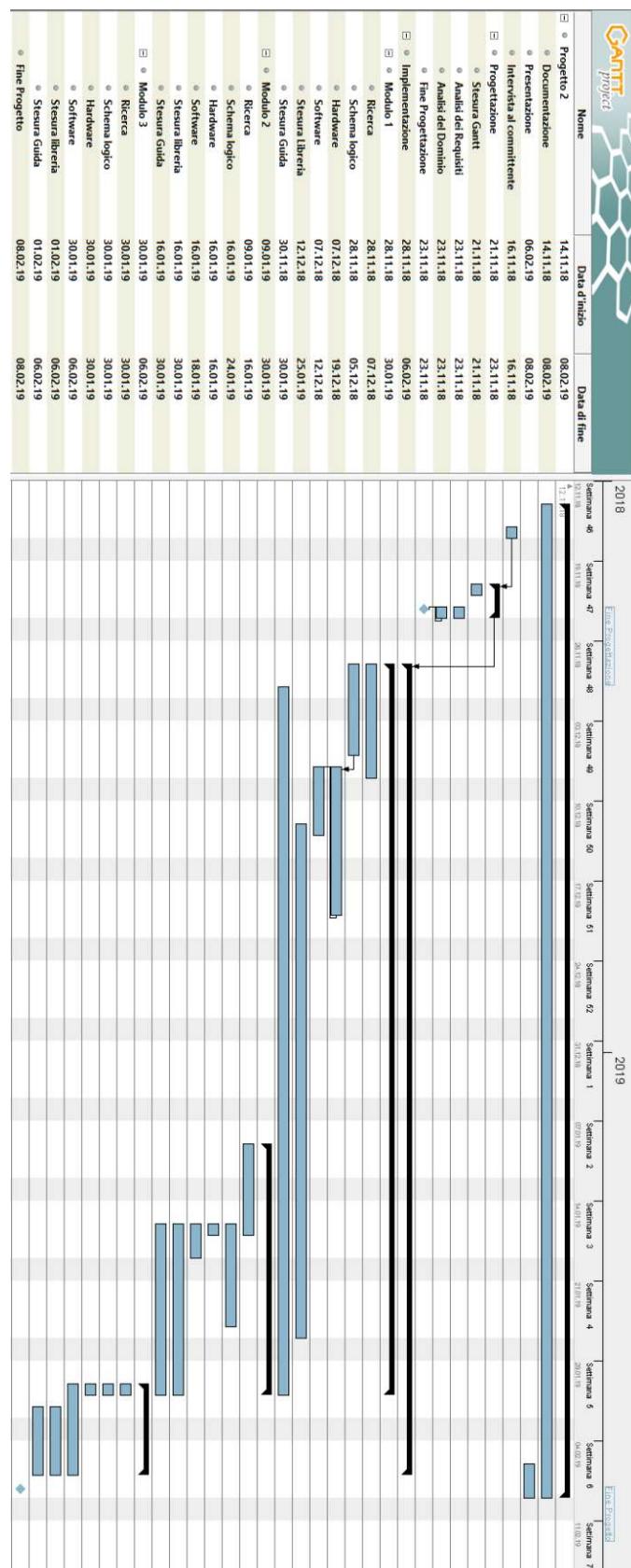
## 4.2 Risultati test

Test Case	Nome	Descrizione	Risultato
TC-001	Libreria led RGB	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-002	Libreria potenziometro	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-003	Libreria bottone	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-004	Libreria led	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-005	Libreria ultrasuoni	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-006	Libreria cicalino	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-007	Esempio1 led RGB e potenziometro	Verificare che il primo esempio sul modulo led RGB e il potenziometro funziona correttamente.	Corretto
TC-008	Esempio2 led RGB e potenziometro	Verificare che il secondo esempio sul modulo led RGB e il potenziometro funziona correttamente.	Corretto
TC-009	Esempio3 led RGB e potenziometro	Verificare che il terzo esempio sul modulo led e il bottone funziona correttamente.	Corretto
TC-010	Esempio1 led e bottone	Verificare che il primo esempio sul modulo led e il bottone funziona correttamente.	Corretto
TC-011	Esempio2 led e bottone	Verificare che il secondo esempio sul modulo led e il bottone funziona correttamente.	Corretto
TC-012	Esempio3 led e bottone	Verificare che il terzo esempio sul modulo led e il bottone funziona correttamente.	Corretto
TC-013	Esempio1 ultrasuoni e cicalino	Verificare che il primo esempio sul modulo led e il bottone funziona correttamente.	Corretto
TC-014	Esempio2 ultrasuoni e cicalino	Verificare che il secondo esempio sul modulo ultrasuoni e il cicalino funziona correttamente.	Corretto
TC-015	Esempio3 ultrasuoni e cicalino	Verificare che il terzo esempio sul modulo ultrasuoni e il cicalino funziona correttamente.	Corretto

## 4.3 Mancanze/limitazioni conosciute

Il pin P4 di Arduino Digispark, al momento del push del software, deve essere scollegato dal circuito. Per ovviare leggermente a questa problematica, abbiamo deciso di inserire nel circuito uno switch che apre o chiude il contatto con P4 e il relativo pin sulla bread-board

## 5 Consuntivo



Durante lo svolgimento del primo modulo abbiamo riscontrato numerosi ritardi dovuti ad altrettanto numerosi imprevisti causati dal fatto che l'Arduino Digispark è molto più diverso dall'Arduino Mega che conoscevamo già. Questo ci ha ritardati molto sulla realizzazione dell'hardware dal momento che ogni volta che sistemavamo qualcosa, riscontravamo una problematica diversa dovuta alle porte del Digispark.

La stesura della prima guida è durata così tanto perché avendo continuamente bisogno di modificare parti di hardware e di software, il ritardo si è sommato.

Avendo però riscontrato tutti questi problemi con il primo modulo, con i successivi 2 siamo riusciti ad impiegare veramente poco tempo rimanendo così nei tempi richiesti per la consegna del prodotto.

## **6 Conclusioni**

---

### **6.1 Considerazioni personali**

Tornare a lavorare con Arduino dopo quasi un anno di pausa, l'abbiamo trovato molto interessante.

Riconosciamo che specialmente nei primi tempi abbiamo avuto molti momenti di crisi dovuti più che altro alla mancata conoscenza del nuovo micro-controllore (Digispark), infatti abbiamo perso gran parte del tempo a cercare informazioni e a documentarci sul quale porta potesse fare ciò che ci serviva.

Superato questo momento di ricerca, siamo riusciti a ristabilirci e a tornare in linea con la programmazione. Detto ciò, pensiamo di aver imparato abbastanza da questo progetto e tutto sommato ci è piaciuto nonostante tutti i problemi riscontrati.

## **7 Bibliografia**

---

### **7.1 Sitografia**

- <https://digi-stump.com/wiki/digispark/tutorials/debugging?sj=serial&sj=monitor>
- [http://tuxamito.com/wiki/index.php/File:Digispark\\_with\\_pinout.jpg#filelinks](http://tuxamito.com/wiki/index.php/File:Digispark_with_pinout.jpg#filelinks)
- <https://www.critics-corporation.com/RaspberryPi/come-scrivere-e-creare-una-libreria-per-arduino>
- <https://it.emcelettronica.com/come-scrivere-libreria-arduino>
- <https://forum.arduino.cc/index.php?topic=126156.0>
- <https://www.swzone.it/Arduino--come-creare-e-gestire-una-libreria---43873.html>

## **8 Allegati**

---

- Diari di lavoro;
- Potenziometro & Led RGB:
  - User-Guide;
  - Esempi di codice;
- Pulsante & Led:
  - User-Guide;
  - Esempi di codice;
- Ultrasuoni & Cicalino:
  - User-Guide;
  - Esempi di codice;
- Librerie:
  - Led;
  - Led RGB;
  - Cicalino (Buzzer);
  - Potenziometro;
  - Pulsante;
  - Sensore Ultrasuoni;

# GUIDA ARDUINO DIGISPARK

## Potenziometro + Led RGB



Mattia Ruberto & Matteo Ghilardini

# SOMMARIO

<i>Sommario</i> .....	2
<i>Scopo</i> .....	3
Arduino Digispark.....	3
Potenziometro .....	4
Led RGB .....	4
<i>Schema Elettrico</i> .....	5
<i>Librerie</i> .....	6
Libreria Led RGB .....	6
Libreria Potenziometro.....	Errore. Il segnalibro non è definito.
<i>Utilizzo</i> .....	7
<i>Hardware</i> .....	7
<i>Software</i> .....	8

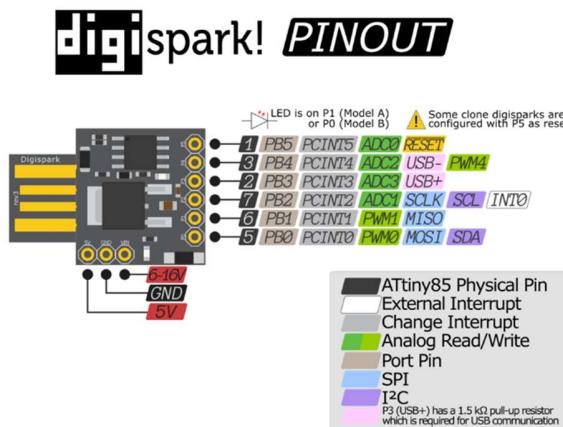
# Scopo

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](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:

#### Digistump AVR Boards by Digistump versione 1.6.7 INSTALLED

Schede incluse in questo pacchetto:

Digispark (Default - 16.5mhz), Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte buffer), Digispark Pro (16 Mhz) (64 byte buffer),  
Digispark (16mhz - No USB), Digispark (8mhz - No USB), Digispark (1mhz - No USB).

[Online help](#)

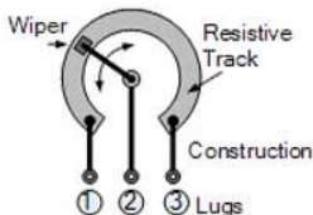
[More info](#)

nell'angolo in basso a destra 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).

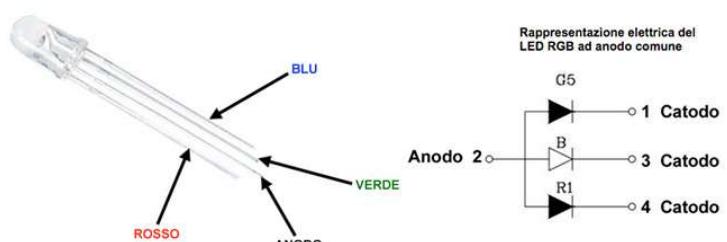


I potenziometri hanno un multiplo di 3 pin (il più comune e quello che usiamo anche noi che ne ha infatti 3) dove i 2 laterali vengono collegati al “+” e al “-”, mentre quello centrale ritorna il valore desiderato. Come nelle resistenze normali, anche i potenziometri non hanno polarità perciò è indifferente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

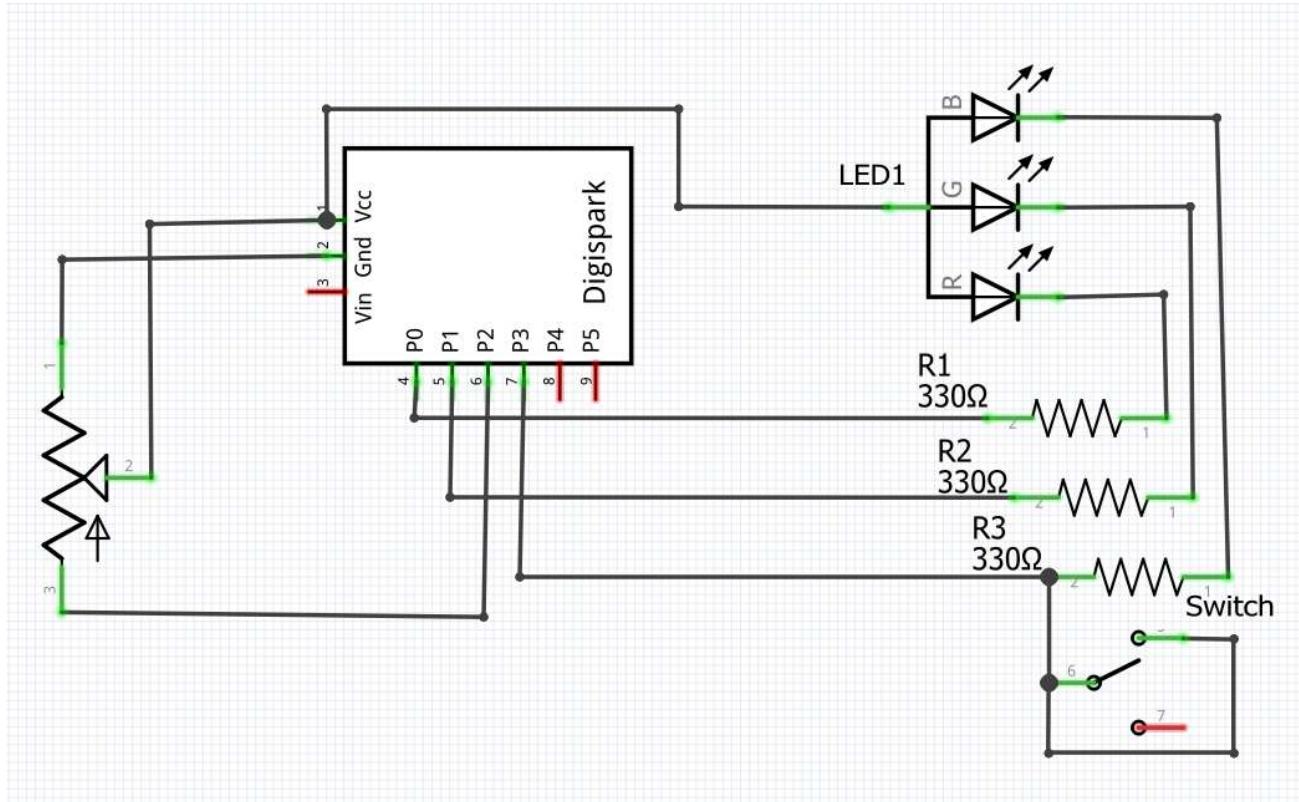
## Led RGB

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  $330\ \Omega$ .

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

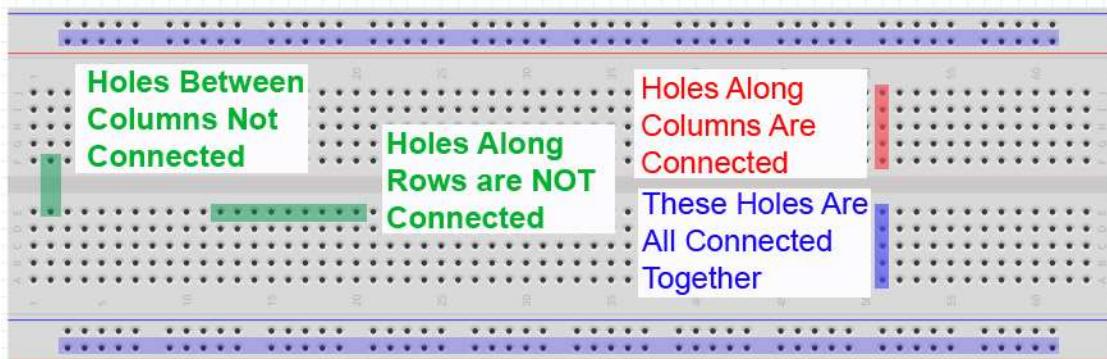
I componenti da utilizzare per questo progetto sono i 3 citati più e più volte all'interno di questa guida:

- 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 collegiamo delle resistenze da  $330\ \Omega$  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);  
}
```

# GUIDA ARDUINO DIGISPARK

## Pulsante + Led



Mattia Ruberto & Matteo Ghilardini

# SOMMARIO

<i>Sommario</i> .....	2
<i>Scopo</i> .....	3
<i>Componenti</i> .....	3
Arduino Digispark.....	3
Pulsante.....	4
Led RGB .....	4
<i>Schema Elettrico</i> .....	5
<i>Librerie</i> .....	6
Libreria Led .....	6
<i>Utilizzo</i> .....	6
<i>Hardware</i> .....	8
<i>Software (ogni codice dovrà essere mostrato)</i> .....	<i>Errore. Il segnalibro non è definito.</i>

# 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.

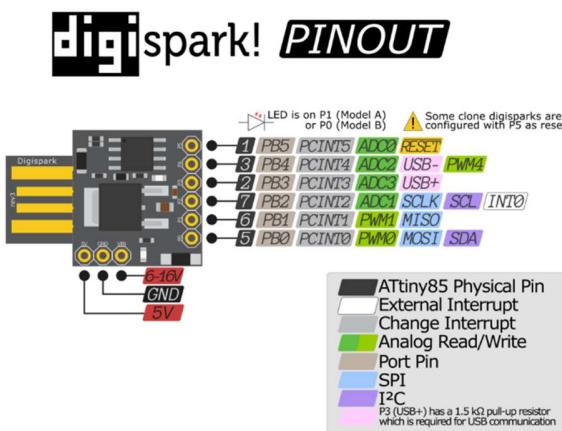
# Componenti

## 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](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:

#### Digistump AVR Boards by Digistump versione 1.6.7 INSTALLED

Schede incluse in questo pacchetto:

Digispark (Default - 16.5Mhz), Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte buffer), Digispark Pro (16 Mhz) (64 byte buffer),  
Digispark (16mhz - No USB), Digispark (8mhz - No USB), Digispark (1mhz - No USB).

[Online help](#)

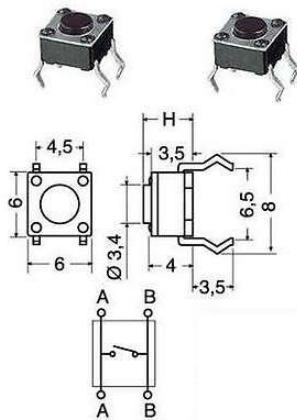
[More info](#)

nell'angolo in basso a destra 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).

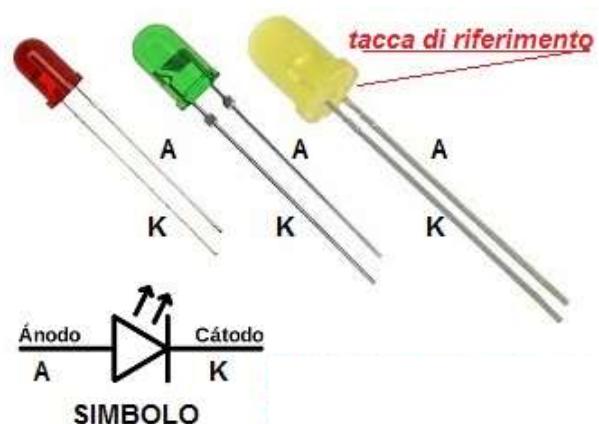


I pulsanti non hanno nessun circuito sensibile al loro interno, quindi non è necessario prestare attenzione ai voltaggi che gli vengono impressi o alla loro polarità. Questo perché, come già detto, i pulsanti sono esattamente come se fossero due cavi che vengono collegati e scollegati a seconda del fatto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

## Led

Le 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  $330\ \Omega$ .

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.



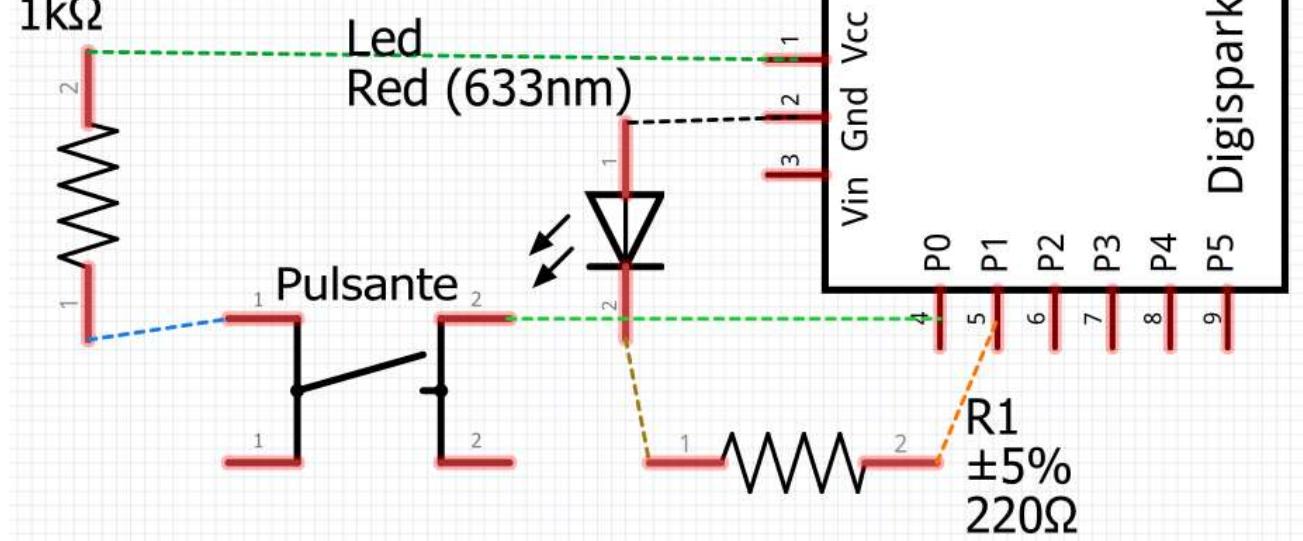
DIODI LED diam. 5 mm

# Schema Elettrico

Resistenza

$\pm 5\%$

$1k\Omega$



# 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

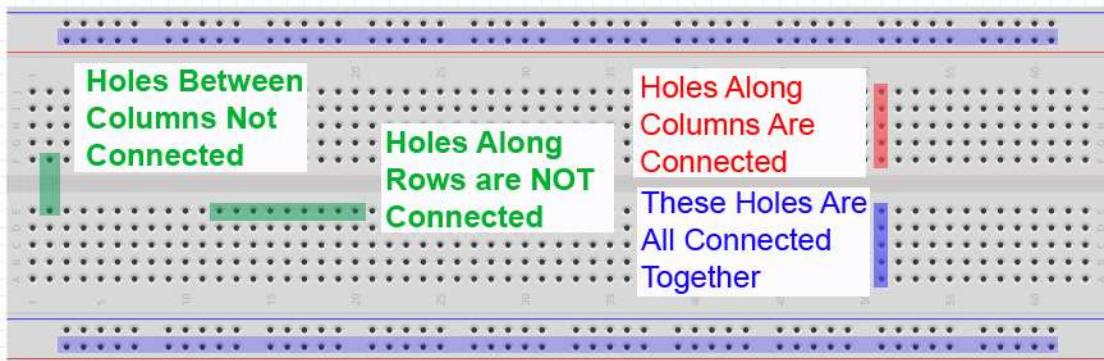
I componenti da utilizzare per questo progetto sono i 3 citati più e più volte all'interno di questa guida:

- 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

- $220\Omega$  per il led: deve essere collegata in serie fra il collegamento al Digispark (P1) e il polo positivo del led (quello più lungo).
- $10K\Omega$  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 lampeggerà, 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:

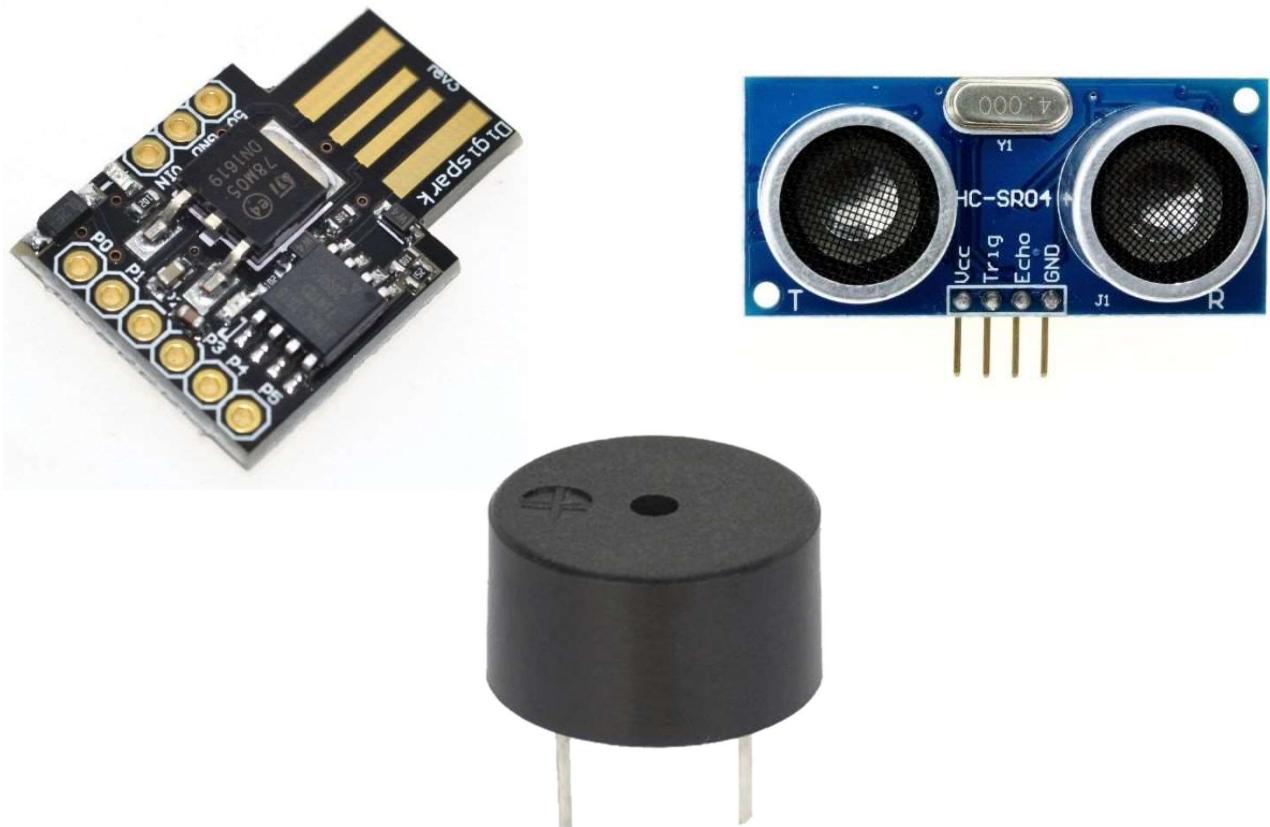
```
boolean stato_bot = libraryButton.getStateButton();
```

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();
}
```

# GUIDA ARDUINO DIGISPARK

## Ultrasuoni + Cicalino



Mattia Ruberto & Matteo Ghilardini

# SOMMARIO

<i>Sommario</i> .....	2
<i>Scopo</i> .....	3
<i>Componenti</i> .....	3
Arduino Digispark.....	3
Sensore a Ultrasuoni .....	4
Cicalino .....	4
<i>Schema Elettrico</i> .....	5
<i>Librerie</i> .....	6
Libreria Buzzer (Cicalino) .....	6
Libreria UltraSound (Sensore a Ultrasuoni).....	6
<i>Utilizzo</i> .....	7
<i>Hardware</i> .....	7
<i>Software</i> .....	9

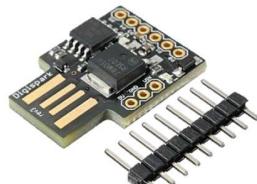
# Scopo

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.

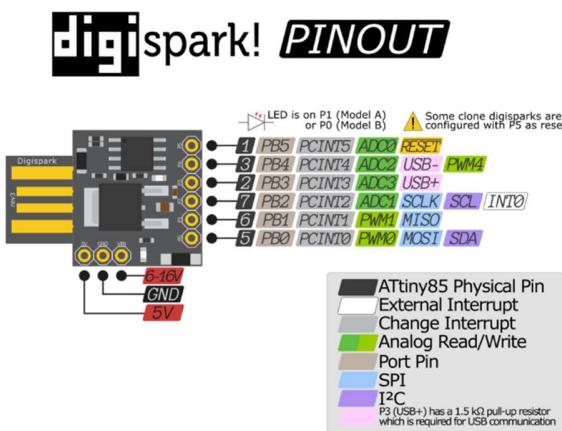
# Componenti

## 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](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:

#### Digistump AVR Boards by Digistump versione 1.6.7 INSTALLED

Schede incluse in questo pacchetto:

Digispark (Default - 16.5Mhz), Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte buffer), Digispark Pro (16 Mhz) (64 byte buffer),  
Digispark (16mhz - No USB), Digispark (8mhz - No USB), Digispark (1mhz - No USB).

[Online help](#)

[More info](#)

nell'angolo in basso a destra 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.

## Sensore a Ultrasuoni

Un sensore che lavora con gli ultrasuoni sfrutta le onde emesse da qualunque rumore per percepire la distanza dell'oggetto che ha fatto rimbalzare tale impulso.

Esistono diversi tipi di sensori ad ultrasuoni che si differenziano gli uni dagli altri da potenza, frequenza o portata del segnale, ma concettualmente sono tutti pressoché identici nel funzionamento e nella struttura.

Quelli più basilari, come anche il nostro, sono composti da 2 “antenne” (un'emittente e una ricevente), 2 pin per l'elettricità (VCC e GND) e 2 pin per la lettura dei dati.



Il funzionamento di tale apparecchio si basa in realtà sul tempo che impiega il segnale a tornare alla sorgente, in seguito a questo, utilizzando la semplice formula fisica  $S = V * T$  (conoscendo la velocità del suono che è di circa  $343 \frac{m}{s}$  alla temperatura di  $20^\circ C$ ) è in grado di definire la distanza dall'oggetto con il calcolo  $S = 343 * T$  (tenendo conto che il tempo dovrà essere in secondi per mantenere valida questa formula).

## Cicalino

Concettualmente il cicalino è un apparecchio estremamente banale:

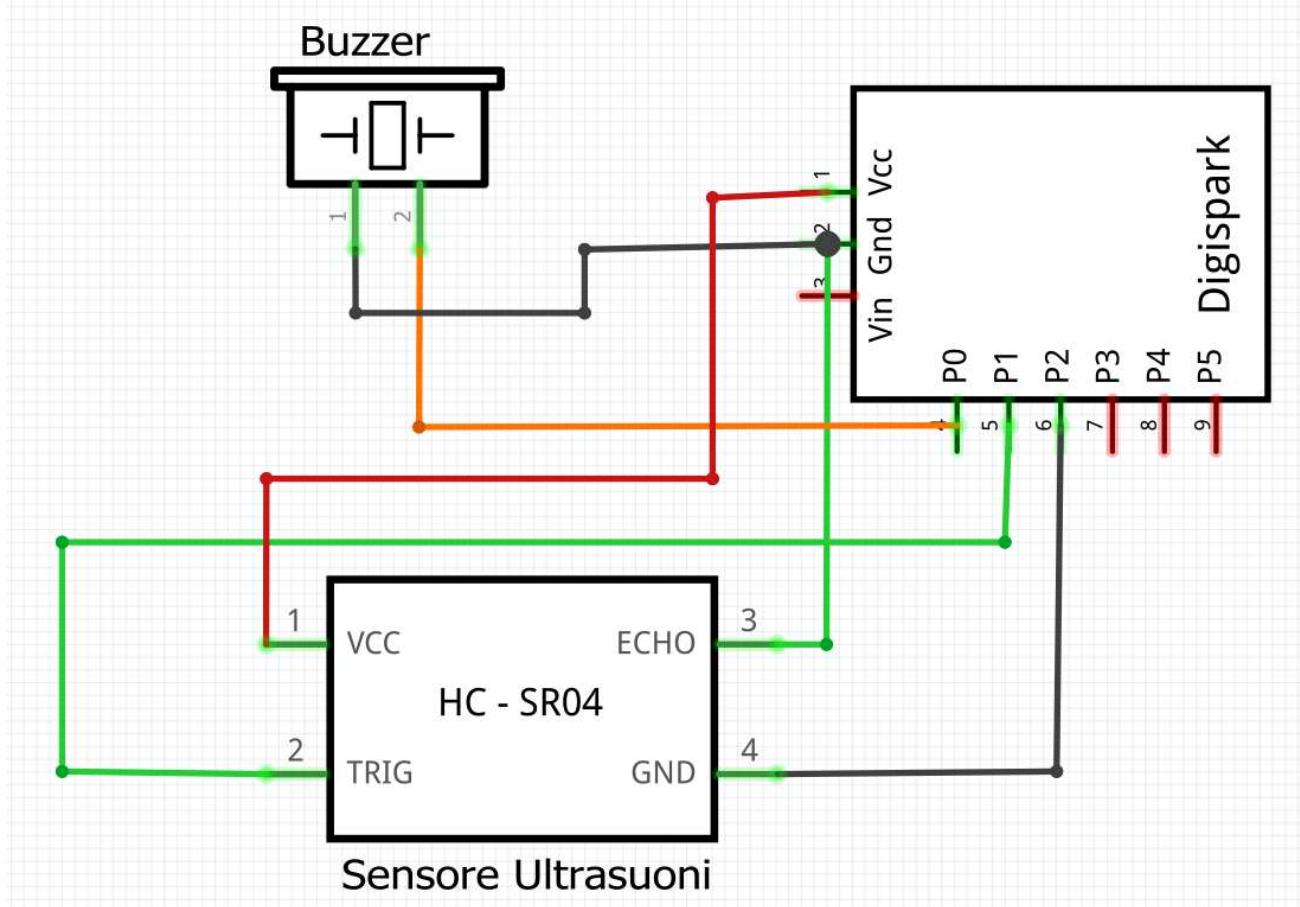
al suo interno troviamo una membrana che viene mossa da degli impulsi elettromagnetici e questi movimenti generano il suono. Per modificare il suono si procede col impostare una frequenza diversa nel segnale che viene mandato al circuito.

Non può essere perciò modificato il volume sonoro, ma come già detto, solo la l'intervallo fra un impulso e il successivo (ossia la frequenza).

Alcuni cicalini vengono illustrati con un polo rosso e uno nero, ma in realtà (quando si tratta di cicalini come il nostro, ossia con 2 pin) non c'è una polarità.



# 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 Buzzer (Cicalino)

L'Header contiene:

- ◆ 1 attributo: corrisponde al pin al quale è collegato il cicalino;
- ◆ 3 metodi:
  - `setPinBuzzer(int buzzerPort);`
  - `setTone(int frequency);`
  - `powerOff();`

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

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

Ecco cosa fa nello specifico ogni metodo:

- ◆ **setPinBuzzer**: in base al parametro che riceve, attribuisce tale valore all'attributo che indica il pin del cicalino;
- ◆ **setTone**: fa emettere al cicalino dei suoni a seconda della frequenza ricevuta come parametro;
- ◆ **powerOff**: spegne il cicalino.

## Libreria UltraSound (Sensore a Ultrasuoni)

L'Header contiene:

- ◆ 2 attributi: corrispondenti ai pin di emissione e ricezione del sensore;
- ◆ 3 metodi:
  - `setUltraSoundPin(int pinTrigPort, int pinEchoPort);`
  - `getDistance();`

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

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

Ecco cosa fa nello specifico ogni metodo:

- ◆ **setUltraSoundPin**: in base ai parametri che riceve, attribuisce tali valori agli attributi che indicano i pin di emissione e ricezione del sensore;
- ◆ **getDistance**: ritorna la distanza misurata dal sensore in cm;

# Utilizzo

## Hardware

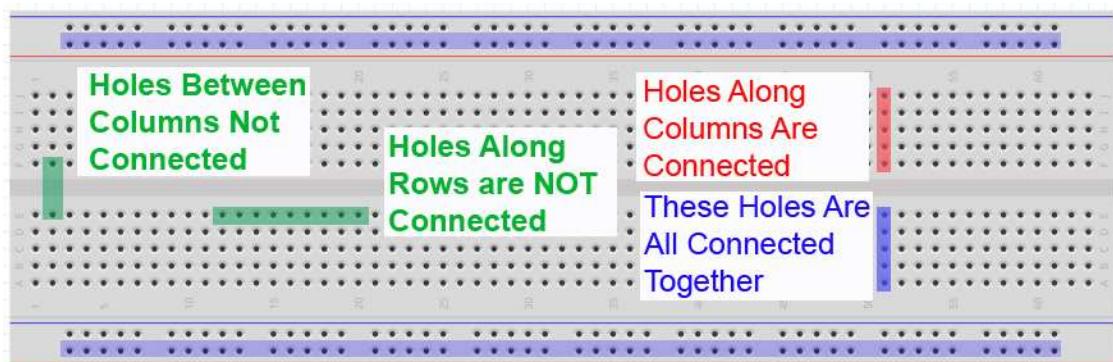
I componenti da utilizzare per questo progetto sono i 3 citati più e più volte all'interno di questa guida:

- Arduino Digispark;
- Sensore a Ultrasuoni;
- Cicalino (o Buzzer);

Per costruire il circuito dobbiamo fissare il sensore e il cicalino 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 sensore a ultrasuoni in modo che tutti i 4 pin non siano in contatto fra loro.

Il buzzer deve essere collegato con un pin al GND dell'arduino, mentre con l'altro ad una porta del Digispark (noi usiamo P0). Il buzzer non ha polarità quindi è indifferente quale pin viene collegato a cosa.

Il Sensore a ultrasuoni ha i 2 pin esterni che corrispondono a VCC (polo positivo) e GND (polo negativo) che vanno collegati rispettivamente a VCC e ad una porta del digispark (per noi P2). I 2 pin più interni che sono "trig" e "echo" si occupano del segnale; echo va collegato al GND dell'arduino, mentre trig ad un pin di lettura (per noi P1).



## Software

- Il primo esempio di codice che abbiamo realizzato con questi componenti si occupa di suonare sempre più velocemente il cicalino man mano che la distanza registrata diminuisce.

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 <LibraryUltraSound.h>
#include <LibraryBuzzer.h>
```

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

```
LibraryUltraSound libraryUltraSound;
LibraryBuzzer libraryBuzzer;
```

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire ai pin di emissione e ricezione del sensore a ultrasuoni e al pin del Buzzer un pin di Digispark:

```
void setup() {
    libraryBuzzer.setPinBuzzer(0);
    libraryUltraSound.setUltraSoundPin(1, 2);
}
```

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

La prima cosa che facciamo nel loop è memorizzare la distanza registrata dal sensore:

```
int distance = libraryUltraSound.getDistance();
```

A questo punto, dopo aver controllato che la distanza registrata sia minore di 1 metro, mappiamo tale distanza in modo da ottenere l’intervallo fra un suono e l’altro del cicalino. Al termine di tutto impostiamo un delay di 50 millisecondi per evitare interferenze:

```
if(distance <= 100){
    int risultato = map(distance, 3, 100, 20, 500);
    libraryBuzzer.setTone(100);
    delay(risultato);
    libraryBuzzer.powerOff();
}
delay(50);
```

- Il nostro secondo esempio di codice invece utilizza la frequenza, ossia la distanza letta dal sensore e la frequenza del cicalino sono direttamente proporzionali.

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 <LibraryUltraSound.h>
#include <LibraryBuzzer.h>
```

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

```
LibraryUltraSound libraryUltraSound;  
LibraryBuzzer libraryBuzzer;
```

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire ai pin di emissione e ricezione del sensore a ultrasuoni e al pin del Buzzer un pin di Digispark:

```
void setup() {  
    libraryBuzzer.setPinBuzzer(0);  
    libraryUltraSound.setUltraSoundPin(1, 2);  
}
```

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

La prima cosa che facciamo nel loop è memorizzare la distanza registrata dal sensore:

```
int distance = libraryUltraSound.getDistance();
```

A questo punto mappiamo tale distanza in modo da ottenere la frequenza dei suoni con un delay di 100 millisecondi fra un suono e l'altro:

```
int distance = libraryUltraSound.getDistance();  
int frequenza = map(distance, 0, 100, 0, 1000);  
libraryBuzzer.setTone(frequenza);  
delay(100);
```

- Il terzo esempio che abbiamo pensato fa in modo che il cicalino cominci a suonare a partire da una certa distanza.

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 <LibraryUltraSound.h>  
#include <LibraryBuzzer.h>
```

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

```
LibraryUltraSound libraryUltraSound;  
LibraryBuzzer libraryBuzzer;
```

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire ai pin di emissione e ricezione del sensore a ultrasuoni e al pin del Buzzer un pin di Digispark:

```
void setup() {  
    libraryBuzzer.setPinBuzzer(0);  
    libraryUltraSound.setUltraSoundPin(1, 2);  
}
```

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

La prima cosa che facciamo nel loop è memorizzare la distanza registrata dal sensore:

```
int distance = libraryUltraSound.getDistance();
```

Se la distanza registrata è inferiore (o pari) a 20, il buzzer comincia a suonare pressoché ininterrottamente (intervallo di 10 millisecondi). In caso contrario, si spegne:

```
if(distance <= 20){  
    libraryBuzzer.setTone(100);  
    delay(10);  
}else{  
    libraryBuzzer.powerOff();  
}
```

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-14

### Lavori svolti

Nella prima parte della lezione ci è stato assegnato il nuovo “Quaderno dei compiti” del secondo progetto ossia “Sistema didattico per Arduino con libreria per attuatori e relativa documentazione”, poi abbiamo creato il progetto su github. Nella seconda parte invece abbiamo assistito alle presentazioni dei nostri compagni di classe.

### Problemi riscontrati e soluzioni adottate

-

### Punto della situazione rispetto alla pianificazione

(Non ancora svolta)

### Programma di massima per la prossima giornata di lavoro

Iniziare la progettazione

**(TEST 2, mod.306)**

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-16

## Lavori svolti

Nelle prime due ore della lezione è stato svolto il secondo test del modulo, mentre nelle ultime due ore invece è stata fatta l'analisi dei dati con il cliente, in questo caso il docente, sono state chiarite tutte le domande che c'erano a riguardo.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

-

## Programma di massima per la prossima giornata di lavoro

Per la prossima lezione si dovrà scrivere l'analisi dei dati sulla documentazione e iniziare la progettazione.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-21

**Lavori svolti**

Oggi abbiamo svolto la prima fase della progettazione, ossia il diagramma Gantt.

**Problemi riscontrati e soluzioni adottate****Punto della situazione rispetto alla pianificazione**

In linea con la pianificazione

**Programma di massima per la prossima giornata di lavoro**

Analisi dei requisiti

Analisi del dominio

# Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-23

## Lavori svolti

Oggi abbiamo realizzato l'analisi del dominio, dei requisiti e dei mezzi, l'abstract, lo scopo.

Inoltre ci siamo informati sul come creare delle librerie per Arduino e ne abbiamo creata una come test delle informazioni trovate.

Abbiamo poi preparato hardware e software per il potenziometro che presenta sul led RGB una gamma di 6 colori.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In anticipo sulla pianificazione

## Programma di massima per la prossima giornata di lavoro

Eseguire i test del modulo.

Generare la libreria del primo modulo.

Eseguire i test della libreria.

Generare la guida relativa alla prima libreria.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-28

## Lavori svolti

Oggi abbiamo svolto una breve ricerca riguardo i componenti necessari a questo primo sotto-progetto, ossia il potenziometro e il led RGB.

Dopo esserci informati abbiamo cominciato a realizzare lo schema logico utilizzando frtzing.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

Non siamo riusciti a terminare lo schema logico (leggermente in ritardo).

## Programma di massima per la prossima giornata di lavoro

Terminare lo schema logico.

Realizzare il circuito hardware

Realizzare il codice software

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-30

## Lavori svolti

In queste quattro ore ci sono stati consegnati gli arduino digispark, abbiamo preso i connettori femmina e li abbiamo saldati ai due arduino, inoltre abbiamo iniziato la guida iniziale dell'utilizzo dell'arduino mini e abbiamo continuato a implementazione del modulo led RGB e potenziometro, provandolo con il nuovo arduino e migliorando il tutto.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

-

## Programma di massima per la prossima giornata di lavoro

Completare il circuito, fare il fritzing e continuare la guida.

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-12-05

### Lavori svolti

Oggi abbiamo realizzato l'hardware definitivo e lo abbiamo collegato al digispark.  
Abbiamo inoltre terminato il fritzing.

### Problemi riscontrati e soluzioni adottate

Il digispark con il circuito attaccato, quando viene collegato al pc non viene individuato. Ho provato a collegare resistenze diverse per verificare quale fosse il problema e ho riscontrato che probabilmente era la resistenza ad essere troppo bassa, quindi ne ho prese 2 più potenti e nelle ore seguenti testerò se è effettivamente quello il problema

### Punto della situazione rispetto alla pianificazione

In linea

### Programma di massima per la prossima giornata di lavoro

Sistemare il circuito

Completare la guida

Generare le librerie

## Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-07

### Lavori svolti

Oggi abbiamo dovuto ricreare tutto il circuito hardware e tutto il software perché una volta collegato il circuito al Digispark niente funzionava.

### Problemi riscontrati e soluzioni adottate

Non avevamo trovato quali porte del Digispark fossero predisposte a fare cosa, quindi abbiamo dovuto effettuare un'approfondita ricerca a riguardo che si è dimostrata generalmente infruttuosa. Per questo abbiamo cominciato a fare diversi test per ogni porta in relazione al potenziometro fino ad arrivare alla fine dove abbiamo collegato le porte a seconda delle loro predisposizioni tentando di utilizzare gli indici più bassi possibile.

Abbiamo anche dovuto sistemare il fatto che sul pc di Ruberto, non veniva riconosciuto il Digispark.

### Punto della situazione rispetto alla pianificazione

In ritardo di circa 1 ora (lezione)

### Programma di massima per la prossima giornata di lavoro

Realizzare la libreria finale

Completare la guida utente

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-12-12

## Lavori svolti

Oggi ci è stato detto di testare tutti i programmi di base di Digispark, e ciò abbiamo fatto. Dopo di questo, abbiamo proseguito con la redazione della guida utente (manca solo la parte relativa alle librerie). Abbiamo anche cominciato a realizzare la prima libreria.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In ritardo di circa 2 ore (lezione)

## Programma di massima per la prossima giornata di lavoro

Completare le librerie

Completare la guida utente

# Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-14

## Lavori svolti

Oggi ci siamo concentrati sulla libreria, l'abbiamo completata, nella libreria abbiamo deciso di inserire i metodi setColors() che riceve tre valori da 0 a 255 e settano i colori rosso, verde e blu, altri tre metodi setColorRed(), setColorGreen() e setColorBlue() che ricevono solo il proprio valore da settare, un metodo getValuePotenziometer() che ritorna il valore del potenziometro. Abbiamo ridato un'occhiata alla documentazione.

## Problemi riscontrati e soluzioni adottate

Abbiamo avuto dei problemi a capire quali metodi inserire all'interno della libreria, infatti all'inizio volevamo inserire altri metodi, oltretutto il digispark ci sta dando problemi dato che abbiamo bisogno 4 porte analogWrite ma abbiamo notato che vanno solamente singolarmente mentre tutte e 4 assieme non vanno, pensiamo si un problema di voltaggio.

## Punto della situazione rispetto alla pianificazione

-

## Programma di massima per la prossima giornata di lavoro

Completare il circuito, fare il fritzing e continuare la guida.

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-12-19

### Lavori svolti

Anche oggi abbiamo dedicato la maggior parte della lezione a tentare di risolvere il problema secondo il quale P4 (se collegata) non permette al pc di riconoscere il Digispark.

### Problemi riscontrati e soluzioni adottate

P4, se collegata non permette al pc di riconoscere il Digispark e quindi al software di caricare il programma. Come soluzione temporanea abbiamo trovato che è sufficiente scollegare il pin prima di collegare il Digispark al pc, e ricollegarlo dopo aver caricato il programma.

### Punto della situazione rispetto alla pianificazione

In ritardo di 6 ore lezione circa (1 settimana)

Penso che recupereremo tempo nel corso delle prossime lezioni perché stiamo riscontrando tutti i problemi che poi saranno già risolti per i prossimi moduli.

### Programma di massima per la prossima giornata di lavoro

Completare le librerie

Completare la guida utente

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-12-21

## Lavori svolti

Oggi abbiamo svolto la teoria sul come fare e non fare una presentazione. Al termine abbiamo visto 2 esempi di presentazioni: quella di Carlo e Quella di Mattia L.

## Problemi riscontrati e soluzioni adottate

Abbiamo risolto il problema relativo al P4 mettendo uno switch da commutare in modo che passi la corrente solo dopo aver caricato il programma sul resto dell'hardware.

## Punto della situazione rispetto alla pianificazione

In ritardo di 6 ore lezione circa (1 settimana)

## Programma di massima per la prossima giornata di lavoro

Completare le librerie

Completare la guida utente

# Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-09

## Lavori svolti

Oggi abbiamo utilizzato la maggior parte del tempo per fare il punto della situazione. Il resto del tempo, l'abbiamo invece utilizzato per sistemare la documentazione e il codice fino a questo punto.

Abbiamo inoltre iniziato a pensare al nuovo modulino che sarebbe composto da 2 bottoni e un led RGB. Non siamo ancora certi che questo sia possibile, dobbiamo verificare che l'Arduino permetta l'utilizzo adeguato per ognuna di queste porte.

## Problemi riscontrati e soluzioni adottate

Problemi di funzionamento della libreria, ma senza capirne il motivo. (Ruberto)

## Punto della situazione rispetto alla pianificazione

Molto in ritardo rispetto alla pianificazione, non avevamo preso in considerazione le vacanze dove non abbiamo potuto lavorare come invece programmavamo.

## Programma di massima per la prossima giornata di lavoro

Completare (finalmente) le librerie

Cominciare il secondo modulino con relativa guida

## Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-11

### Lavori svolti

Oggi ci siamo divisi il lavoro:

- Ruberto: sistemato la libreria del led RGB e fatto tutti i 3 codici di esempi relativi alla libreria citata in precedenza.
- Ghilardini: proseguito con la documentazione facendo l'analisi dei requisiti, proseguito con la guida utente aggiornando lo schema elettrico, composto il capitolo "Librerie", ampliato il capitolo "Utilizzo" aggiornando la parte di "Hardware" e introdotto la parte di "Software" (mancano solo gli esempi di codice con relativa spiegazione).

Nell'ultima ora di lezione abbiamo discusso sui prossimi 2 moduli da svolgere e abbiamo concluso che faremo:

- Bottone & Led;
- Infrarossi & Cicalino;

### Problemi riscontrati e soluzioni adottate

Ruberto ha risolto il problema relativo alla libreria semplicemente disinstallando e poi ri-installando il software di Arduino.

### Punto della situazione rispetto alla pianificazione

Abbiamo scoperto di dover fare solo 3 moduli (invece di 5), quindi pensiamo di farcela a rimanere nei tempi senza troppi problemi.

### Programma di massima per la prossima giornata di lavoro

Realizzare hardware e software per il secondo modulino

Realizzare la guida utente per il secondo modulino

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2019-01-16

### Lavori svolti

Oggi abbiamo completato l'hardware e il software per il modulo del led con pulsante.

In seguito ci siamo nuovamente suddivisi il lavoro:

- Ruberto: cominciato a realizzare la libreria per il led realizzando l'header della libreria e i primi 6 metodi;
- Ghilardini: cominciato la guida utente per il secondo modulo;

### Problemi riscontrati e soluzioni adottate

### Punto della situazione rispetto alla pianificazione

In linea con la pianificazione della settimana scorsa

### Programma di massima per la prossima giornata di lavoro

Completare la libreria per il secondo modulino

Completare la guida utente per il secondo modulino

# Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-18

## Lavori svolti

Quest'oggi siamo andati avanti con la documentazione e la libreria del bottone, che è stata completata e la libreria del led, nella libreria del led sono stati inseriti i seguenti attributi:

- Int led
- Int state led
- Bool lastButtonState
- Unsigned long lastDebounceTime = 0
- Unsigned long debounceDelay = 50;

metodi:

- Void setLedPin(int ledPort)
- Void powerOn()
- Void powerOff()
- Void setLed(bool stato\_led)
- Void blink(int frequency)
- Void toggle(bool stato\_bot)

Dopo abbiamo fatto due, esempi di utilizzo, il terzo è da finire, il primo quando il bottone è premuto il led lampeggia e nel secondo abbiamo fatto il toggle.

Nella documentazione siamo andati avanti con la guida di utilizzo delle librerie e con i test degli esempi.

## Problemi riscontrati e soluzioni adottate

Abbiamo riscontrato problemi nella programmazione del metodo toggle della libreria del led, infatti abbiamo avuto problemi a capire come funzionava.

## Punto della situazione rispetto alla pianificazione

-In linea

## Programma di massima per la prossima giornata di lavoro

Finire il terzo esempio e l'ultimo modulo

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2019-01-23

### Lavori svolti

Le ore di oggi le abbiamo investite per completare/migliorare il lavoro svolto finora, nello specifico abbiamo completato gli esempi di codice del secondo modulo e completato le User-guide dei 2 moduli fatti fino ad ora.

### Problemi riscontrati e soluzioni adottate

### Punto della situazione rispetto alla pianificazione

Leggermente in ritardo, venerdì dobbiamo svolgere hardware e software del terzo modulo per poter completare documentazione e presentazione settimana prossima

### Programma di massima per la prossima giornata di lavoro

Realizzare tutto ciò che è relativo al ultimo modulo

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2019-01-25

### Lavori svolti

Oggi abbiamo:

- Revisionato le varie librerie;
- Rifatto gli esempi applicando le modifiche apportate alle librerie;
- Modificato leggermente la guida utente del modulo 1;
- Continuato la guida utente del modulo 2;
- Realizzato lo schema fritzing del modulo 2;

### Problemi riscontrati e soluzioni adottate

### Punto della situazione rispetto alla pianificazione

Avendo ricevuto una settimana in più, in linea con la pianificazione

### Programma di massima per la prossima giornata di lavoro

Realizzare hardware e software del 3° modulo

Realizzare la guida del 3° modulo

# Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-30

## Lavori svolti

Oggi abbiamo (tutto riguardo il 3° modulo):

- Realizzato Hardware;
- Realizzato schema Fritzing;
- Realizzato il primo esempio di codice;
- Realizzato la libreria per il cicalino e quella per l'infrarossi;
- Sistemato le librerie dei moduli precedenti;

Inoltre abbiamo proseguito nella documentazione finale e iniziato la guida per il 3° modulo.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

## Programma di massima per la prossima giornata di lavoro

Realizzare gli altri 2 esempi di codice

Completare la guida del 3° modulo

Iniziare la presentazione

Proseguire la documentazione

## Diario di lavoro

---

Luogo	SAM Trevano
Data	2019-02-01

### Lavori svolti

Oggi abbiamo revisionato tutto il lavoro svolto finora:

- Documentazione di progetto (proseguita e migliorata);
- Guide utenti per moduli 1, 2 e 3(migliorate 1 e 2, proseguita 1 e 3);
- Librerie Led RGB e Potenziometro entrambe completate;
- Sistemato un'incoerenza nella libreria del led col metodo “toggle”;

Inoltre abbiamo realizzato:

- 2 esempi per il 3° modulo

### Problemi riscontrati e soluzioni adottate

### Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

(forse leggermente in ritardo con la presenzazione)

### Programma di massima per la prossima giornata di lavoro

Realizzare la presentazione

Completare tutti i frammenti di codice in sospeso

Completare i test per la documentazione

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2019-02-06

## Lavori svolti

Ghilardini era assente.

Io ho completato e migliorato l'analisi dei requisiti, ho messo apposto e ho controllato che tutto fosse apposto nelle librerie e negli esempi e ho commentato tutte le librerie e tutti gli esempi.

## Problemi riscontrati e soluzioni adottate

Non sapevo bene quante tabelle fare.

## Punto della situazione rispetto alla pianificazione

In linea

## Programma di massima per la prossima giornata di lavoro

Completare i test case.

# Diario di lavoro

Luogo	SAM Trevano
Data	2019-02-08

## Lavori svolti

Oggi abbiamo completato ciò che ci mancava:

- Guida 3;
- Documentazione generale;
  - Test case e esiti dei test;
  - Gantt Consuntivo;
  - Conclusione personale (realizzata come entità singola, duo)
- Presentazione;

Alla fine abbiamo riordinato tutte le cartelle e abbiamo consegnato il lavoro.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

## Programma di massima per la prossima giornata di lavoro