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

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 3E 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

ID: REQ-01	
Nome	Libreria led RGB
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
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.

ID: REQ-02	
Nome	Libreria potenziometro
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
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	
Nome	Libreria bottone
Priorità	1
Versione	1.0
Note	
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	
Nome	Libreria led
Priorità	1
Versione	1.0
Note	
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.

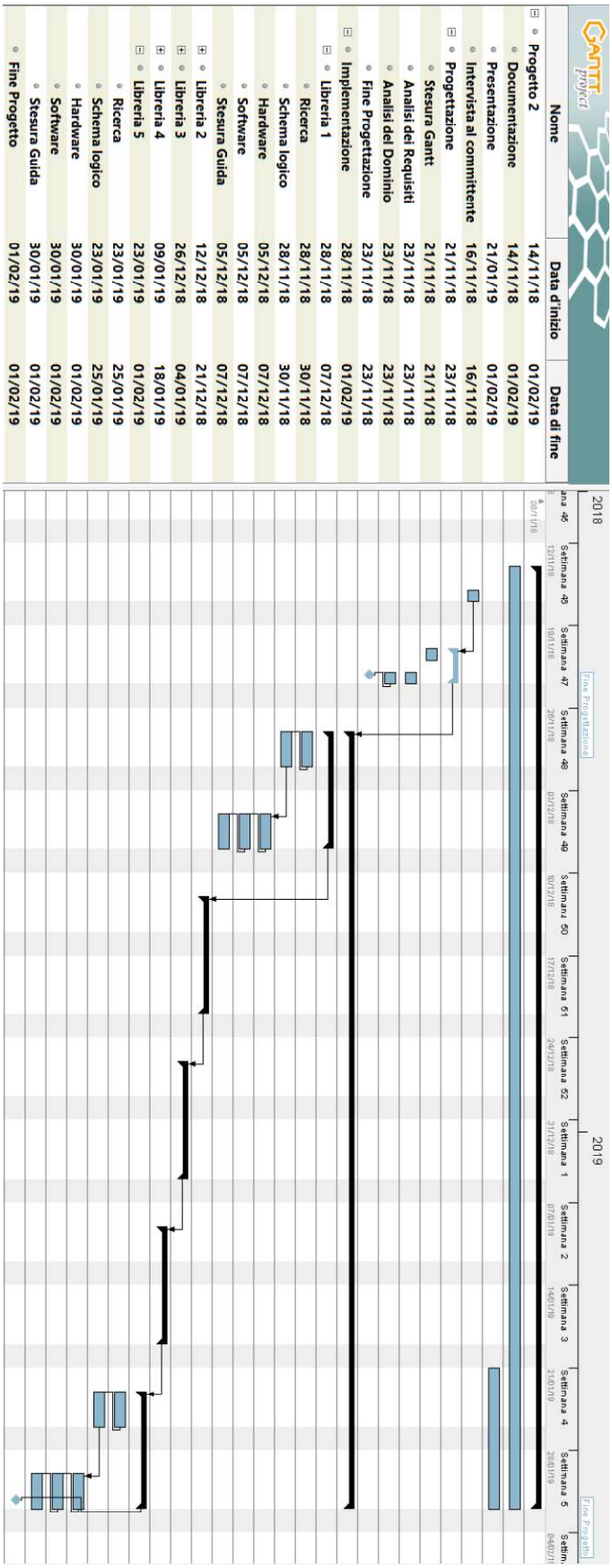
ID: REQ-05	
Nome	Libreria sensore ultrasuoni
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Realizzare la libreria per il sensore ultrasuoni.
002	Implementare metodo che setta le porte del sensore ultrasuoni.
003	Implementare metodo che ritorna la distanza misurata dal sensore in cm.

ID: REQ-06	
Nome	Libreria sensore cicalino
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
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	
Nome	Realizzare tre attuatori.
Priorità	1
Versione	1.0
Note	Il nostro progetto è composto da tra attuatori, led RGB e potenziometro, led e bottone, cicalino e sensore ultrasuoni, non ho fatto una tabella per attuatore perché ci sembrava ripetitivo.
Sotto requisiti	
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	
Nome	Realizzare una guida
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Spiegare come funziona, cosa fare per iniziare a usarlo e come funziona.
002	Spiegare come utilizzare ogni singola libreria.
003	Spiegare come funzione ogni singolo metodo delle librerie.
004	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

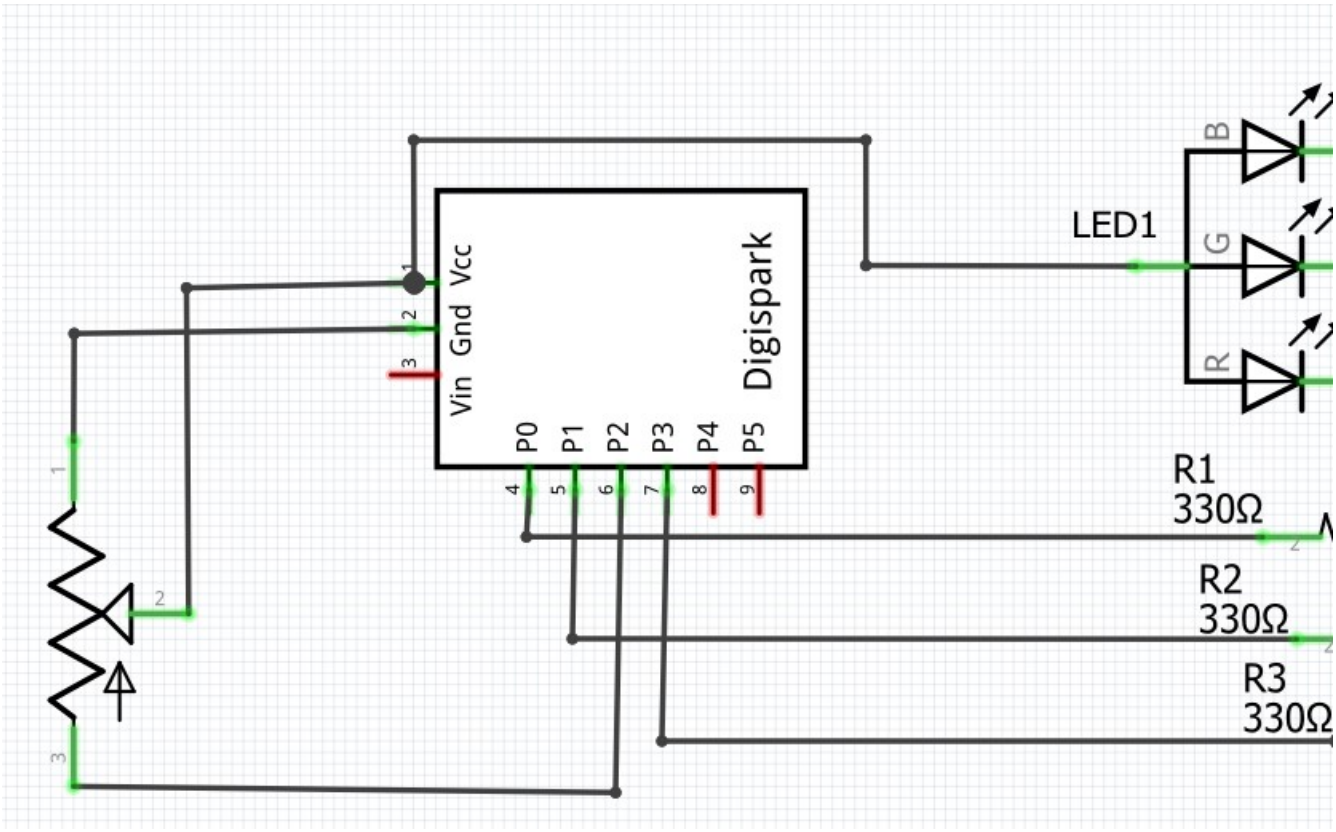
1.7.2 Hardware

- 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;

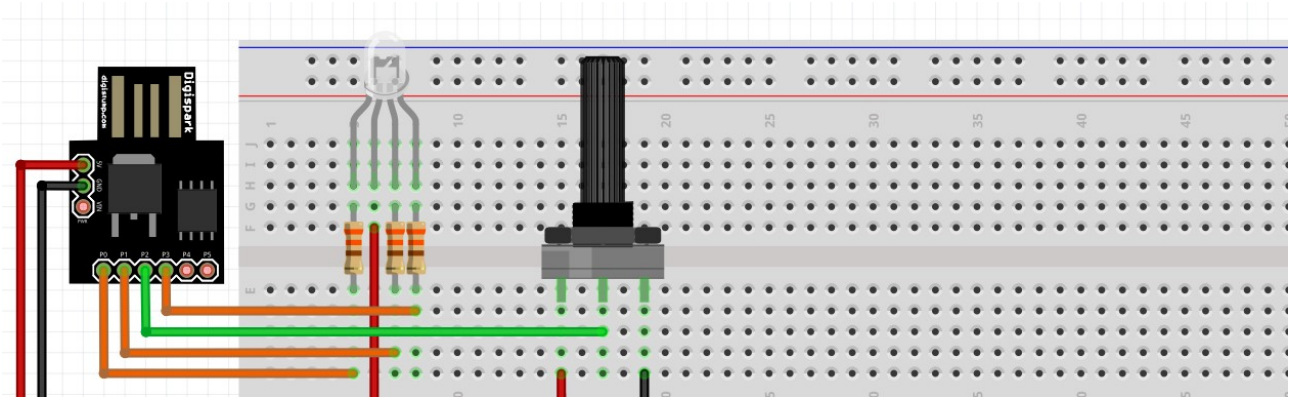
2 Progettazione (Modulo Potenziometro e Led RGB)

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 "Aruino.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 "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;

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:

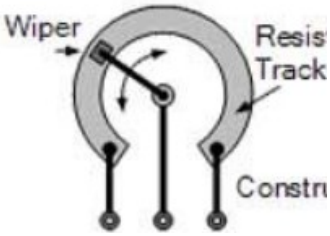
- **setUltraSoundPin**: in base ai parametri che riceve, attribuisce tali valori agli attributi che indicano i pin di emissione e ricezione del sensore;
- **getDistance**: ritorna la distanza misurata dal sensore in cm;

3.2 Modulo Potenziometro e led RGB

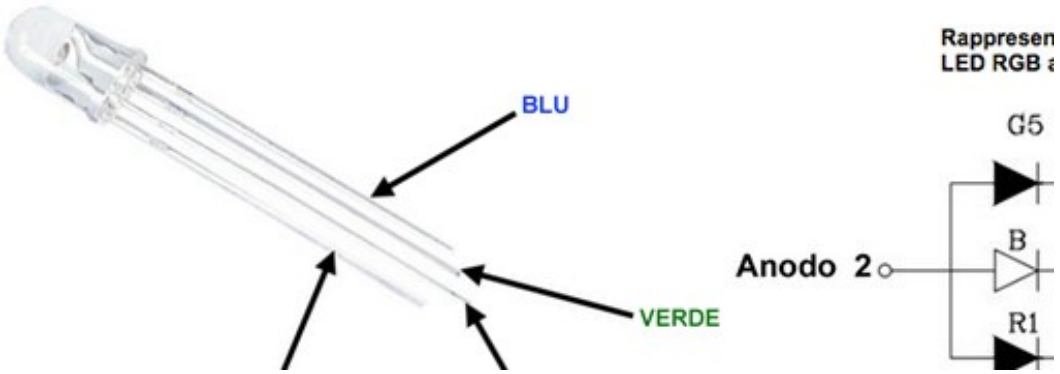
Presentiamo un solo modulo per evitare ripetizioni.
Se si desidera approfondire, consultare le guide allegate.

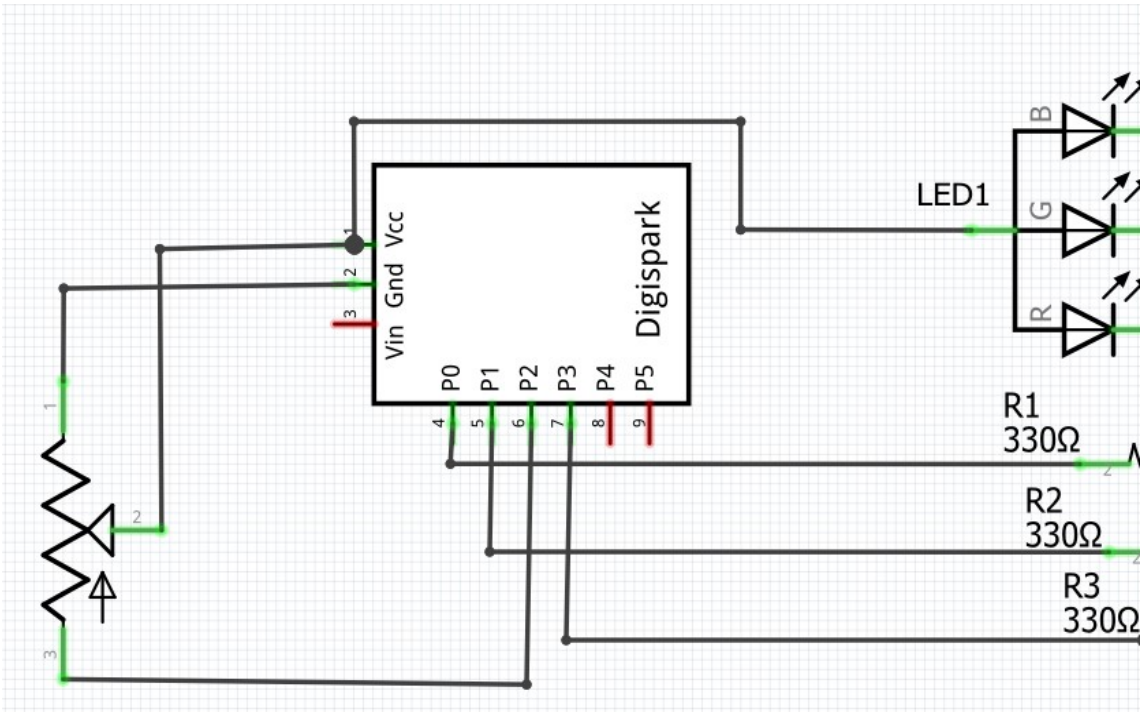
Hardware

Per realizzare questo modulo abbiamo utilizzato, oltre ai componenti specifici (potenziometro e led RGB), delle resistenze da 330Ω 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

Test Case: **Riferimento**:	TC-001 REQ-01	**Nome:**	Verificare che la libreria del led RGB funzioni.
Descrizione:	Includere la libreria, compilare il codice e verificare che non dia errori.		
Prerequisiti:			
Procedura:	• Aprire un nuovo file di Arduino • Andare su “Sketch” • Premere su “#includi libreria” • Scegliere la nostra libreria, in questo caso LibraryRGB • Compilare il file • Istanziare LibraryLedRGB libraryLedRGB; • Richiamare il metodo libraryLedRGB.setLedPin(0,1,4); • Richiamare il metodo libraryLedRGB.setColor(255,255,255) • Richiamare il metodo libraryLedRGB.setRed(255) • Richiamare il metodo libraryLedRGB.setGreen(255) • Richiamare il metodo libraryLedRGB.setBlue(255)		
Risultati attesi:	• La compilazione del file non deve dare errori. • Se il metodo che setta i pin funziona il led RGB deve illuminarsi • Se il metodo setColor(255,255,255) tutte e tre i colori del led devono illuminarsi al massimo • Se il metodo setRed(255) funziona deve accendersi solo il rosso • Se il metodo setGreen(255) funziona deve accendersi solo il verde • Se il metodo setBlue(255) funziona deve accendersi solo il blue		

Test Case:	TC-002	**Nome:**	Verificare che la libreria del potenziometro funzioni.
Riferimento:	REQ-02		
Descrizione:	Includere la libreria, compilare il codice e verificare che non dia errori.		
Prerequisiti:			
Procedura:	• Aprire un nuovo file di Arduino • Andare su “Sketch” • Premere su “#includi libreria” • Scegliere la nostra libreria, in questo caso LibraryPotentiometer • Compilare il file • Istanziare LibraryPotentiometer libraryPotentiometer; • Richiamare il metodo memorizzando il valore in int valuePotentiometer = libraryPotentiometer.getValue(1); • Richiamare il metodo memorizzando il valore in int rangeValuePotentiometer = libraryPotentiometer.getMappedValue(valuePotentiometer, 0, 1023, 0, 255);		
Risultati attesi:	• La compilazione del file non deve dare errori. • 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)		

Test Case:	TC-003	**Nome:**	Verificare che la libreria del bottone funzioni.
Riferimento:	REQ-03		
Descrizione:	Includere la libreria, compilare il codice e verificare che non dia errori.		
Prerequisiti:			
Procedura:	• Aprire un nuovo file di Arduino • Andare su “Sketch” • Premere su “#includi libreria” • Scegliere la nostra libreria, in questo caso LibraryButton • Compilare il file • Istanziare LibraryLed libraryLed • Richiamare il metodo libraryButton.setButtonPin(0) • Richiamare il metodo memorizzando il valore in boolean stato_bot = libraryButton.getStateButton() • Richiamare il metodo boolean state = libraryButton.toggle()		
Risultati attesi:	• La compilazione del file non deve dare errori. • Se il metodo setButtonPin funziona sarà possibile utilizzare il bottone • 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. • 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.		

Test Case: **Riferimento**:	TC-004 REQ-04	**Nome:**	Verificare che la libreria del led funzioni.
Descrizione:	Includere la libreria, compilare il codice e verificare che non dia errori.		
Prerequisiti:			
Procedura:	• Aprire un nuovo file di Arduino • Andare su “Sketch” • Premere su “#includi libreria” • Scegliere la nostra libreria, in questo caso LibraryLed • Compilare il file • Istanziare LibraryLed libraryLed; • Richiamare il metodo setLedPin(1) • Richiamare il metodo libraryLed.blink(600) • Richiamare il metodo libraryLed.powerOff() • Richiamare il metodo libraryLed.powerOn() • Richiamare il metodo libraryLed.setLed(<stato>)		
Risultati attesi:	• La compilazione del file non deve dare errori. • Se il metodo setLedPin funziona il led si illuminerà • Se il metodo blink(600) funziona il led si accederà e spegnerà con un intermittenza di 600 millisecondi • Se il metodo powerOn funziona il led si accende • Se il metodo powerOff funziona il led si spegne		

Test Case:	TC-005	**Nome:**	Verificare che la libreria dell'ultrasuoni funzioni.
Riferimento:	REQ-05		
Descrizione:	Includere la libreria, compilare il codice e verificare che non dia errori.		
Prerequisiti:			
Procedura:	• Aprire un nuovo file di Arduino • Andare su "Sketch" • Premere su "#includi libreria" • Scegliere la nostra libreria, in questo caso LibraryUltraSound • Compilare il file • Istanziare l'oggetto LibraryUltraSound libraryUltraSound; • Richiamare il metodo libraryUltraSound.setUltraSoundPin(1, 2) • Richiamare il metodo nel valore int distance = libraryUltraSound.getDistance()		
Risultati attesi:	• La compilazione del file non deve dare errori. • Se il metodo setUltraSoundPin(1, 2) sarà possibile utilizzare il sensore ultrasuoni • Se il metodo getDistance funziona ritornerà la distanza misurata in cm (Testato con il monitor seriale dell'arduino Mega)		

Test Case:	TC-006	**Nome:**	Verificare che la libreria del cicalino funzioni.
Riferimento:	REQ-06		
Descrizione:	Includere la libreria, compilare il codice e verificare che non dia errori.		
Prerequisiti:			
Procedura:	• Aprire un nuovo file di Arduino • Andare su "Sketch" • Premere su "#includi libreria" • Scegliere la nostra libreria, in questo caso LibraryBuzzer • Compilare il file • Istanziare la libreria LibraryBuzzer libraryBuzzer; • Richiamare il metodo libraryBuzzer.setPinBuzzer(0) • Richiamare il metodo libraryBuzzer.setTone(100) • Richiamare il metodo libraryBuzzer.powerOff()		
Risultati attesi:	• La compilazione del file non deve dare errori. • Se il metodo setPinBuzzer funziona il cicalino funzionerà • Se il metodo setTone(100) funziona il cicalino suonerà con una frequenza di 100 • Se il metodo powerOff funzionerà il cicalino non deve suonare		

Test Case: TC-007 **Riferimento**: REQ-07		**Nome:**	Test sul primo esempio del modulo led RGB e potenziometro.
Descrizione:	Testare il funzionamento del primo esempio del modulo led RGB e potenziometro.		
Prerequisiti:			
Procedura:	• Aprire il primo esempio • Staccare il pin 4 • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Quando il caricamento del codice è completato riattaccare la porta 4 • Ruotare il potenziometro più volte in entrambi i sensi.		
Risultati attesi:	• Il led si deve illuminare di sei colori diversi a dipendenza della posizione del potenziometro partendo dal rosso, giallo, verde, azzurro, blue e viola.		

Test Case: TC-008 **Riferimento**: REQ-07		**Nome:**	Test sul secondo esempio del modulo led RGB e potenziometro.
Descrizione:	Testare il funzionamento del secondo esempio del modulo led RGB e potenziometro.		
Prerequisiti:			
Procedura:	• Aprire il secondo esempio • Staccare il pin 4 • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Quando il caricamento del codice è completato riattaccare la porta 4 • Ruotare il potenziometro più volte in entrambi i sensi.		
Risultati attesi:	• 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.		

Test Case:	TC-009	**Nome:**	Test sul terzo esempio del modulo led RGB e potenziometro.
Riferimento:	REQ-07		
Descrizione:	Testare il funzionamento del secondo esempio del modulo led RGB e potenziometro.		
Prerequisiti:			
Procedura:	• Aprire il terzo esempio • Staccare il pin 4 • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Quando il caricamento del codice è completato riattaccare la porta 4 • Ruotare il potenziometro più volte in entrambi i sensi.		
Risultati attesi:	• Il led deve passare tutte le toanalità 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.		

Test Case:	TC-010	**Nome:**	Test sul primo esempio del modulo led e bottone.
Riferimento:	REQ-07		
Descrizione:	Testare il funzionamento del primo esempio del modulo led e bottone.		
Prerequisiti:			
Procedura:	• Aprire il primo esempio • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Schiacciare il bottone, rilasciarlo e tenerlo premuto.		
Risultati attesi:	• Quando il bottone è premuto il led deve lampeggiare, quando il bottone viene rilasciato il led deve spegnersi.		

Test Case:	TC-011	**Nome:**	Test sul secondo esempio del modulo led e bottone.
Riferimento:	REQ-07		
Descrizione:	Testare il funzionamento del secondo esempio del modulo led e bottone.		
Prerequisiti:			
Procedura:	• Aprire il secondo esempio • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Schiacciare il bottone, rilasciarlo e tenerlo premuto.		
Risultati attesi:	• 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.		

Test Case:	TC-012	**Nome:**	Test sul terzo esempio del modulo led e bottone.
Riferimento:	REQ-07		
Descrizione:	Testare il funzionamento del terzo esempio del modulo led e bottone.		
Prerequisiti:			
Procedura:	• Aprire il terzo esempio • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Schiacciare il bottone, rilasciarlo e tenerlo premuto.		
Risultati attesi:	• 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.		

Test Case:	TC-013	**Nome:**	Test sul primo esempio del modulo cicalino e sensore ultrasuoni.
Riferimento:	REQ-07		
Descrizione:	Testare il funzionamento del primo esempio del modulo cicalino e sensore ultrasuoni.		
Prerequisiti:			
Procedura:	• Aprire il primo esempio • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Prendere un oggetto che permetta al sensore ultrasuoni di misurare la distanza, avvinarlo, allontanarlo e restare fermi.		
Risultati attesi:	• 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.		

Test Case:	TC-014	**Nome:**	Test sul secondo esempio del modulo cicalino e sensore ultrasuoni.
Riferimento:	REQ-07		
Descrizione:	Testare il funzionamento del secondo esempio del modulo cicalino e sensore ultrasuoni.		
Prerequisiti:			
Procedura:	• Aprire il primo esempio • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Prendere un oggetto che permetta al sensore ultrasuoni di misurare la distanza, avvinarlo, allontanarlo e restare fermi.		
Risultati attesi:	• 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.		

Test Case: **Riferimento**:	TC-015 REQ-07	**Nome:**	Test sul terzo esempio del modulo cicalino e sensore ultrasuoni.
Descrizione:	Testare il funzionamento del terzo esempio del modulo cicalino e sensore ultrasuoni.		
Prerequisiti:			
Procedura:	• Aprire il primo esempio • Staccare l'arduino dalla porta USB • Caricare il programma, aspettare che compila, quando dice che l'arduino può essere inserito inserire l'arduino • Prendere un oggetto che permetta al sensore ultrasuoni di misurare la distanza, avvinarlo, allontanarlo e restare fermi		
Risultati attesi:	• Se il sensore ultrasuoni misura una distanza minore ai 20cm inizia a suonare con una frequenza stabile sempre uguale altrimenti si spegne.		

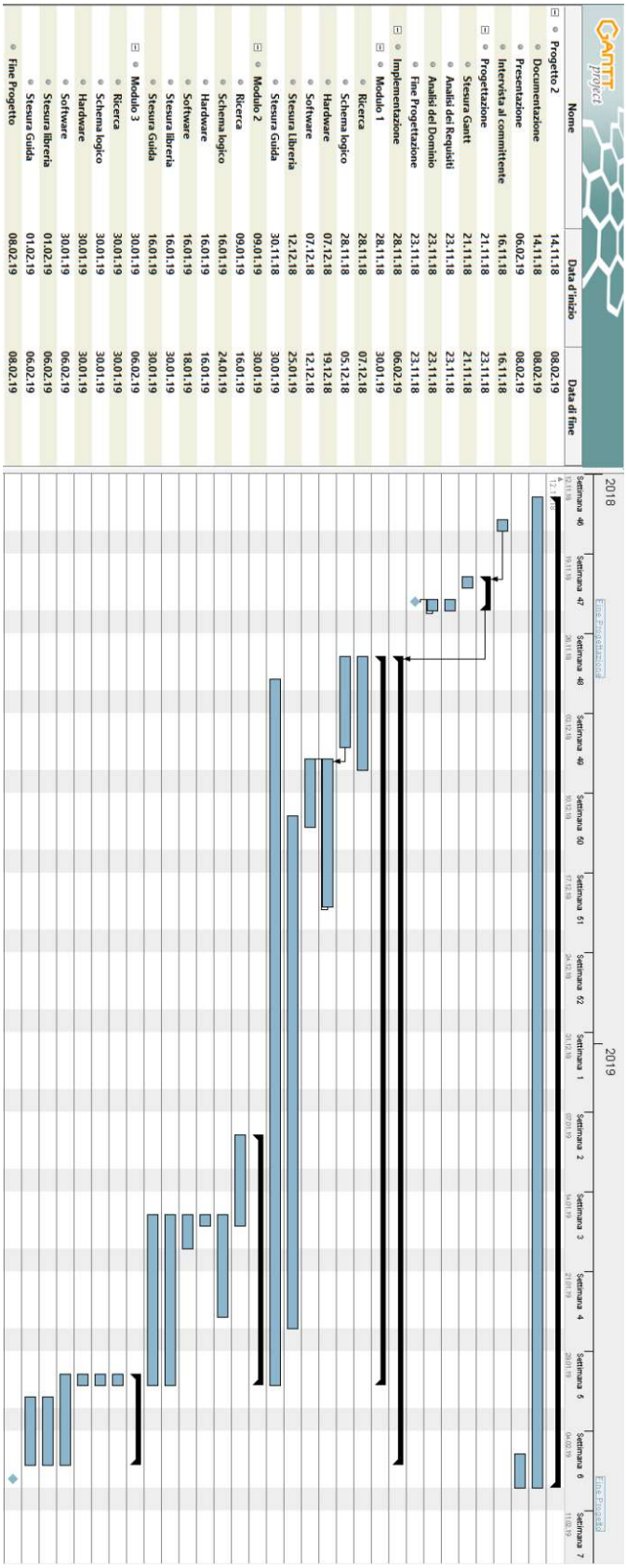
4.2 Risultati test

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://digistump.com/wiki/digispark/tutorials/debugging?s[]=serial&s[]=monitor
- 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;