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: Thor Dublin, Nemanja Stojanovic
Classe: I3AA
Anno scolastico: 2018/19
Docente responsabile: Adriano Barchi, Francesco Mussi, Luca Muggiasca, Massimo Sartori

1 Introduzione

1.1 Informazioni sul progetto

Allievi coinvolti: Thor Dùblin, Nemanja Stojanovic
Classe: I3AA SAMT (Scuola Arti e Mestieri di Trevano)
Docenti responsabili: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori
Data Inizio: 14.11.2018
Data Fine: 08.02.2019

1.2 Abstract

The aim of the project is to create more libraries that allow the use of various modules for our Arduini USB (mini DigiSpark), thanks to this the middle school children will be able to approach in an easy and interactive way to what is the field of computer science, using a library that allows the use of multiple functions, the children will be able to see how the various modules of the Arduino are programmed.

1.3 Scopo

Lo scopo del progetto è quello di creare più librerie che permettano l'utilizzo dei vari moduli per i nostri Arduini USB (mini DigiSpark), grazie a ciò i ragazzi delle medie potranno approcciarsi in modo facile ed interattivo a quello che è il campo dell'informatica, utilizzando una libreria che permetta l'utilizzo di più funzioni, i ragazzi potranno vedere come sono programmati i vari moduli dell'Arduino.

2 Analisi

2.1 Analisi del dominio

Con questo progetto cerchiamo di far sì che i ragazzi delle medie si approccino all'informatica in modo semplice, interattivo e interessante per l'utenza in quella fascia di età ma anche più grande, il progetto funzionerà su hardware Digispark, molto simile ad Arduino ma più piccolo e comodo da utilizzare, verranno forniti schemi dettagliati su come utilizzarlo nella guida.
Per utilizzare questo progetto non si deve far altri che seguire Step-by-Step la guida, grazie a ciò non sono richiesti prerequisiti o conoscenze teoriche fondamentali.

2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Librerie
Priorità	1
Versione	1.0
Note	La creazione delle librerie si realizzeranno in C++
Sotto requisiti	
001	Realizzazione di librerie compatibili con Arduino (che poi verranno implementate su Digispark)
002	Ogni componente dovrà avere una Libreria.

ID: REQ-02	
Nome	Libreria Potenziometro
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	La libreria deve implementare un metodo che ritorna il valore del potenziometro.

ID: REQ-03	
Nome	Libreria Buzzer
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	La libreria deve implementare metodi che settano il Buzzer.

ID: REQ-04	
Nome	Libreria Bottone
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	La libreria deve implementare un metodo che ritorna lo stato del bottone.

ID: REQ-05	
Nome	Libreria LED
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	La libreria deve implementare un metodo che ritorna lo stato del LED.
002	La libreria deve implementare metodi che settano lo stato del LED.

ID: REQ-06	
Nome	Libreria RGB
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	La libreria deve implementare metodi che cambiano gli stati dei Led.

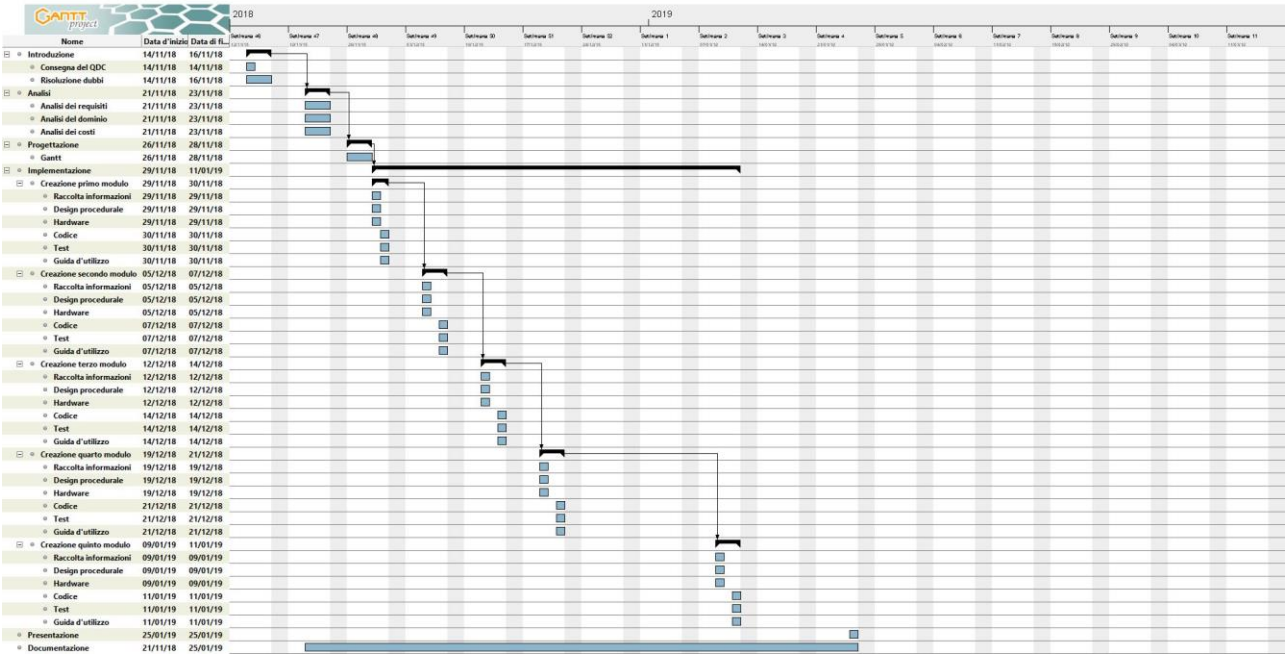
ID: REQ-07	
Nome	Moduli
Priorità	2
Versione	1.0
Note	Con moduli si intende i componenti che useremo insieme per creare delle funzioni.
Sotto requisiti	
001	Ci saranno 3 moduli
002	Ogni modulo deve avere 3 funzioni con i rispettivi livelli di difficoltà: facile, medio, difficile.

ID: REQ-08	
Nome	Modulo 1, Potenziometro-Buzzer.
Priorità	2
Versione	1.0
Note	
Sotto requisiti	
001	La prima funzione(facile) deve implementare la mappatura del potenziometro tramite un parametro.
002	La seconda funzione(medio) deve implementare il cambiamento di frequenza tramite la funzione di mappatura(prima funzione) analogalmente alla rotazione del potenziometro.
003	La terza funzione(difficile) deve implementare il cambiamento di frequenza inverso tramite la funzione di mappatura(prima funzione) analogalmente alla rotazione del potenziometro.

ID: REQ-09	
Nome	Modulo 2, Bottone-Led.
Priorità	2
Versione	1.0
Note	
Sotto requisiti	
001	La prima funzione(facile) deve implementare l'accensione del LED quando il bottone è premuto
002	La seconda funzione (medio) deve implementare il Blink(lampeggiamento) del LED, con un parametro che ne definisce la frequenza.
003	La terza funzione (difficile) deve implementare il toggle del LED, cioè ogniqualvolta che il bottone viene cliccato esso cambia stato.

ID: REQ-10	
Nome	Modulo 3, Potenziometro-RGB.
Priorità	2
Versione	1.0
Note	La sequenza di base dei colori del RGB sono: rosso, rosso-verde, verde, verde-blu, blu, bianco.
Sotto requisiti	
001	La prima funzione digitalRGB, facile, deve implementare il cambio di stato dei LED(digitale), alla rotazione del potenziometro.
002	La seconda funzione analogRGB, medio, deve implementare il cambio di stato dei LED (analogico, cioè in modo fluido), alla rotazione del potenziometro.
003	La terza funzione resetterRGB, difficile, deve implementare: -cambiamento di intensità di un colore, a partire dal bianco, successivamente, rosso, verde blu per poi ricominciare . -quando il colore raggiunge l'intensità massima, per poi raggiungere quella minima, viene cambiato colore.

2.3 Pianificazione



In questo diagramma Gantt abbiamo una pianificazione abbastanza omogenea, che ci permette di affrontare il progetto, con la creazione delle librerie e delle funzione dei moduli con un tempo equivalente tra loro, cosa che probabilmente non sarà possibile, data l'ipotetica differenza di difficoltà tra un modulo e l'altro di cui momentaneamente non siamo a conoscenza.

2.4 Analisi dei mezzi

In questo progetto abbiamo utilizzato maggiormente il programma Arduino, che ci ha permesso di creare degli esempi soddisfacenti delle librerie, e soprattutto dove abbiamo prima testato le funzioni su Arduino Genuino Mega, prima di implementarle su Digispark.

2.4.1 Software

Per realizzare questo progetto abbiamo utilizzato i seguenti software indicandone anche le versioni:

- Word 2016
- PowerPoint 2016
- Adobe Acrobat Reader DC
- Notepad++
- GanttProject 2.8
- GitHub
- Fritzing 0.9.3
- Arduino 1.8.7

Le librerie utilizzate comprendono:

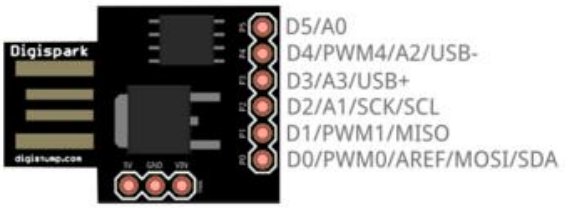
- Arduino (Arduino.h) versione incorporata nell'IDE Arduino 1.8.5 per il linguaggio C++, che permette di scrivere codice in C++ utilizzando i metodi e le funzioni di Arduino

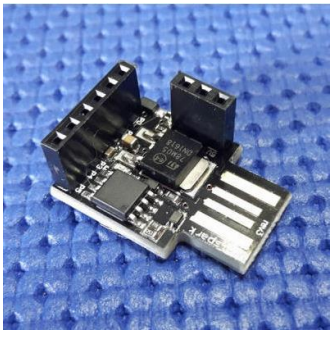
2.4.2 Hardware

Il progetto è interamente su Digispark, tuttavia per implementarlo completamente, io ed il mio compagno abbiamo utilizzato i nostri PC personali, cioè un Acer Aspire F 17 e un Acer Aspire V Nitro, entrambi con sistema operativo Windows 10, inoltre si è utilizzato l'Arduino Genuino Mega per testare tutte le funzioni su di esso prima di implementarle sul Digispark.

Successivamente le specifiche del Digispark:

- IDE 1.0+ (OSX/Windows/Linux)
- Alimentazione 5v (USB)
- 500ma
- 6 I/O Pin
- 8k Flash Memory
- I2C and SPI
- 3 pin PWM
- 4 pin ADC



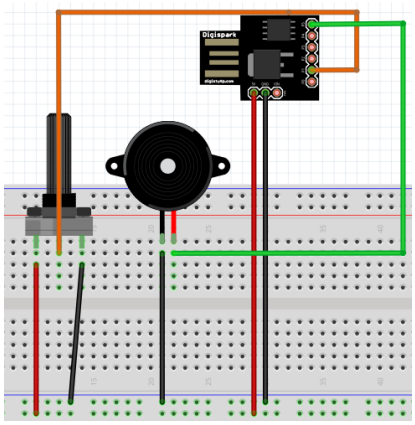


PIN 0	I2C SDA, PWM
PIN 1	PWM
PIN 2	I2C SCK(PWM), Analogico(1)
PIN 3	Analogico(3)
PIN 4	PWM, Analogico(2)
PIN 5	Analogico(0)

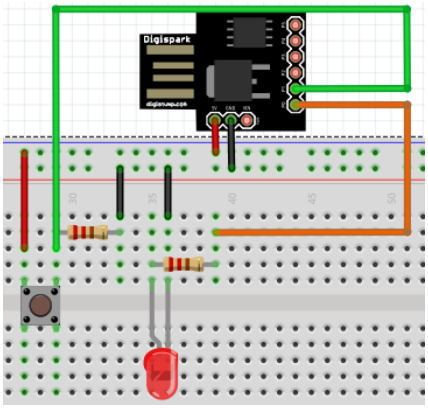
3 Progettazione

3.1 Design dell'architettura del sistema

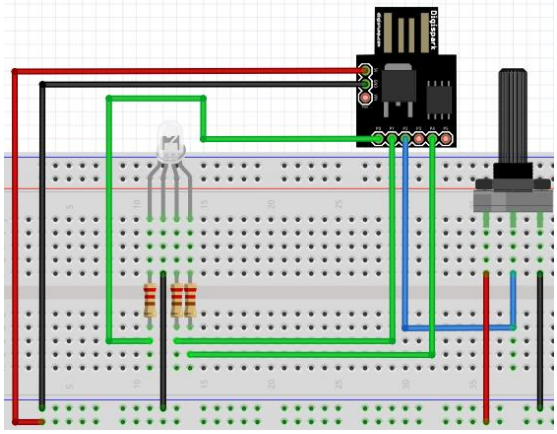
Potenziometro-Buzzer:



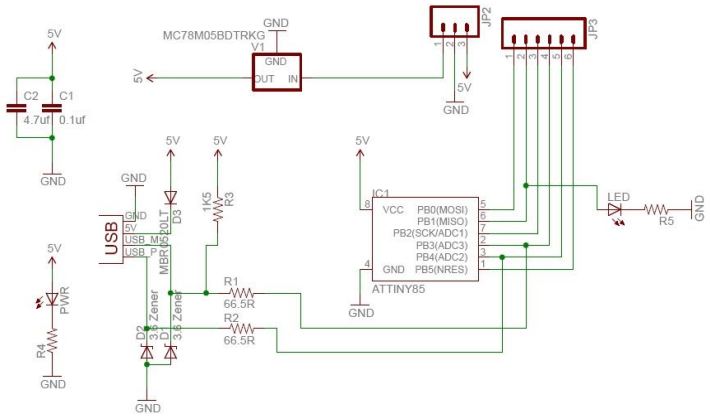
Led-Bottone:



RGB - Potenziometro:



Schema Digispark:



3.2 Design procedurale

3.2.1 Libreria potenziometro

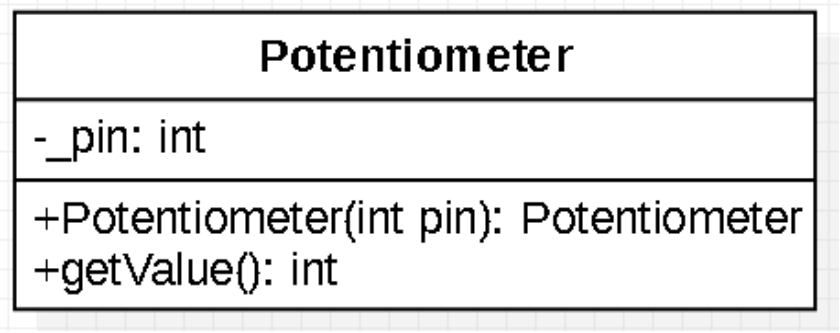
Classi:

- potentiometer.cpp
- potentiometer.h

Metodi:

- Potentiometer(int pin)
  Metodo costruttore che crea l'oggetto tramite il parametro pin che definisce il suo valore.

- getValue():int
  Metodo che ritorna il valore del potenziometro.



3.2.2 Libreria buzzer

Classi:

- Buzzer.cpp
- Buzzer.h

Metodi:

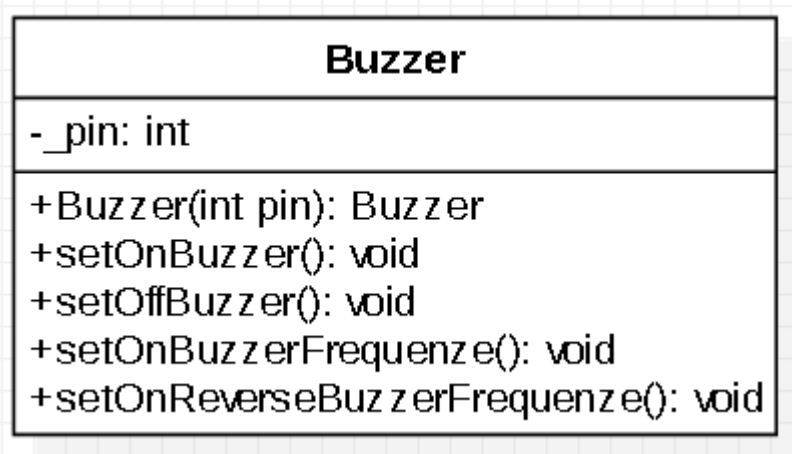
- Buzzer(int pin)
  Metodo costruttore che crea l'oggetto tramite il parametro pin che definisce il suo valore.

- setOnBuzzer():void
  Metodo che accende il buzzer.

- setOffBuzzer():void
  Metodo che spegne il buzzer.

- setOnBuzzerFrequenze():void
  Metodo che setta la frequenza del buzzer in base al valore del potenziometro.

- setOnReverseBuzzerFrequenze():void
  Metodo che setta la frequenza inversa del buzzer in base al valore del potenziometro, invertito.



3.2.3 Libreria bottone

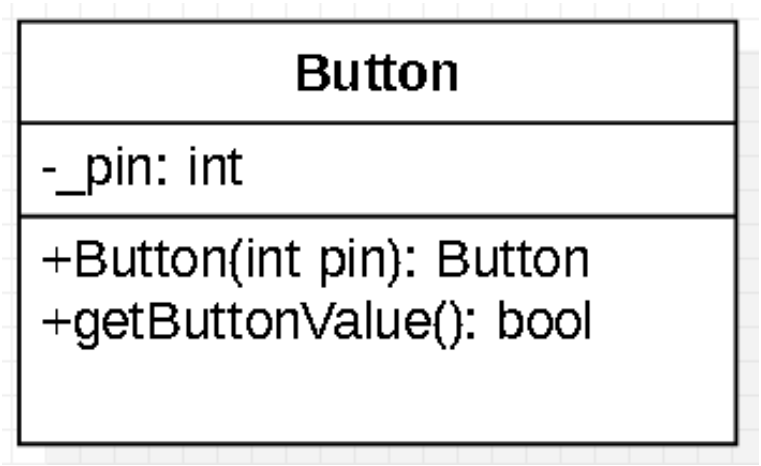
Classi:

- Button.cpp
- Button.h

Metodi:

- Button(int pin)
  Metodo costruttore che crea l'oggetto tramite il parametro pin che definisce il suo valore.

- getButtonValue():bool
  Metodo che ritorna lo stato del bottone.



3.2.4 Libreria Led

Classi:

- Led.cpp
- Led.h

Metodi:

- Led(int pin)
  Metodo costruttore che crea l'oggetto tramite il parametro pin che definisce il suo valore.

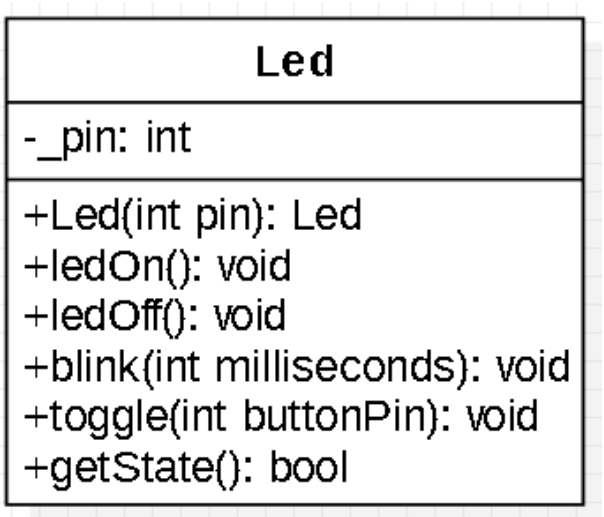
- getState():bool
  Metodo che ritorna lo stato del Led.

- ledOn():void
  Metodo che accende il Led.

- ledOff():void
  Metodo che spegne il Led.

- blink(int milliseconds):void
  Metodo che fa lampeggiare il led, con il parametro che definisce con che frequenza far avvenire il blink.

- toggle(int buttonPin):void
  Metodo che fa il toggle del led, tramite il parametro gli viene passato il pin del bottone che definisce quando far avvenire il toggle del Led



3.2.5 Libreria RGB

Classi:

- RGB.cpp
- RGB.h

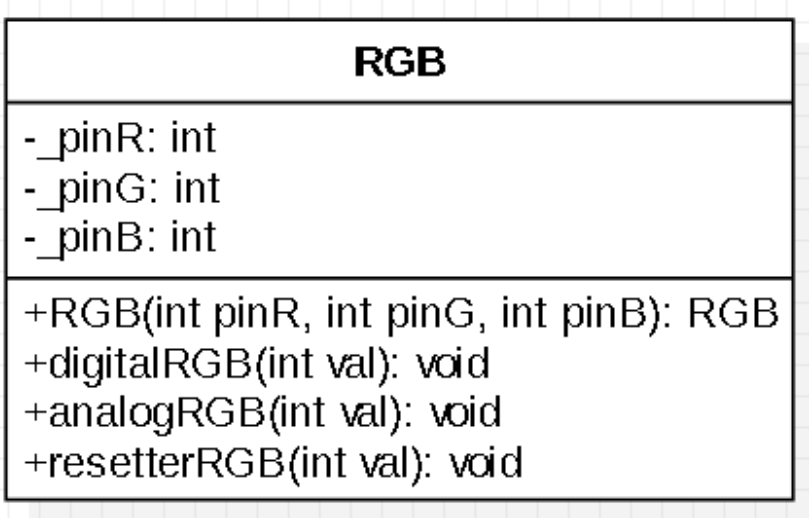
Metodi:

- RGB(int pinR, int pinG, int pinB)
  Metodo costruttore che crea l'oggetto tramite i parametri pin che definiscono i valori dei Led.

- digitalRGB(int val):void
  Metodo che accende e spegne in modo digitale, i Led RGB, in base al valore del potenziometro, che verrà impostato come il parametro val.

- analogRGB(int val):void
  Metodo che cambia il colore dell'RGB in modo analogo, in base al valore del potenziometro, che verrà impostato come il parametro val.

- resetterRGB(int val):void
  Metodo che permette di modificare l'intensità dei colori uno per volta e resettarli, cambiando colore, dopo che si è raggiunto prima il valore massimo e poi quello minimo.
  Tutto in base al valore del potenziometro, che verrà impostato come parametro val.



4 Implementazione

4.1 Libreria Potenziometro

Bisogna creare due classi, una “Potentiometer.cpp” e una “Potentiometer.h”.
Nella classe “Potentiometer.h” bisogna dichiarare un attributo, il pin del potenziometro (_pin). Ci sono due metodi, il metodo Potentiometer(int pin), il metodo getValue ().
Nella classe “Potentiometer.cpp” invece come prima cosa bisogna includere la classe “Potentiometer.h” e Arduino.h”. Il metodo Potentiometer (int pin) è il costruttore che definisce il pin del potenziometro, il metodo intero getValue () che ritorna il valore del potenziometro.

4.2 Libreria Buzzer

Bisogna creare due classi, una “Buzzer.cpp” e una “Buzzer.h”.
Nella classe “Buzzer.h” bisogna dichiarare un attributo, il pin del Buzzer (_pin). Bisogna dichiarare 5 metodi, il metodo Buzzer(int pin), il metodo setOnBuzzer(), il metodo setOffBuzzer(), il metodo frequence(int range, int potValue) e il metodo setOnReverseBuzzerFrequenze(int range, int potValue).

4.3 Modulo Potenziometro-Buzzer

Nella classe “Buzzer.cpp” invece come prima cosa bisogna includere la classe “Buzzer.h” e “Arduino.h”.
Il metodo Buzzer (int pin) è il costruttore che definisce il pin del Buzzer, il metodo setOnBuzzer() e setOffBuzzer() che rispettivamente accendono e spengono il Buzzer, il metodo frequence(int range, int potValue), il quale ritorna la frequenza che verrà applicata al buzzer e infine il metodo setOnReverseBuzzerFrequenze(int range, int potValue), il quale setta la frequenza al buzzer all'inverso.

Nella seguente immagine possiamo vedere il metodo setOnReverseFrequence():

```
/**
* Setta la frequenza al buzzer all'inverso
*/
void Buzzer::setOnReverseBuzzerFrequenze(int range, int potValue){
    int frequence = range / 1024 * potValue;
    int reverse = 1024-frequence;
    tone(_pin, reverse);
}
```

4.4 Libreria Bottone

Bisogna creare due classi, una “Button.cpp” e una “Button.h”.
Nella classe “Button.h” bisogna dichiarare un attributo, il pin del bottone (_pin). Ci sono due metodi, il metodo Button(int pin) , il metodo getButtonValue().
Nella classe “Button.cpp” invece come prima cosa bisogna includere la classe “Button.h” e Arduino.h”. Il metodo Button(int pin) è il costruttore che definisce il pin del bottone, il metodo booleano getButtonValue() che ritorna il valore del bottone.

4.5 Libreria LED

Bisogna creare due classi, una “Led.cpp” e una “Led.h”.
Nella classe “Led.h” bisogna dichiarare un attributo, il pin del Led (_pin). Ci sono 5 metodi, il metodo Led (int pin), il metodo ledOn(), il metodo ledOff(), il metodo blink(int milliseconds), il metodo toggle(int buttonPin).
Nella classe “Led.cpp” invece come prima cosa bisogna includere la classe “Led.h” e Arduino.h”. Il metodo `Led` (int pin) è il costruttore che definisce il pin del `Led`, il metodo getState() che ritorna il valore del Led, il metodo ledOn() e il metodo LedOff() che rispettivamente accendono e spengono il Led, il metodo blink(int milliseconds) che fa lampeggiare il Led con la velocità in base al parametro e toggle(buttonPin) che accende o spegne ad ogni click del bottone.

4.6 Modulo Bottone-LED

Nella seguente immagine possiamo vedere il metodo toggle:

```
void Led::toggle(int buttonPin)
{
    bool mode = 0;
    bool buttonState = 0;

    if (!digitalRead(buttonPin)){
        if (!buttonState) {
          buttonState = true;
          Mode = !Mode;
        }else{
          buttonState = false;
        }

        digitalWrite(_pin, Mode);
        delay(5);
    }
}
```

Qui possiamo vedere il flag mode che viene modificato solo dopo un cambio di stato del buttonState, che avviene quando il bottone viene cliccato.

4.7 Libreria RGB

Bisogna creare due classi, una “RGB.cpp” e una “RGB.h”.
Nella classe “RGB.h” bisogna dichiarare tre attributi, il pin dei Led (_pinR, _pinG, _pinB). Ci sono 4 metodi, il metodo RGB(int pinR, int pinG, int pinB), il metodo digitalRGB(int val), il metodo analogRGB(int val) e il metodo resetterRGB(int val).
Nella classe “RGB.cpp” invece come prima cosa bisogna includere la classe “RGB.h” e Arduino.h”. Il metodo `RGB(int pinR, int pinG, int pinB)` è il costruttore che definisce l’RGB, il metodo digitalRGB(int val) che che cambia lo stato dei led in modo digitale, il metodo analogRGB(int val) e il metodo resetterRGB(int val) che rispettivamente cambia l’intensità dei colori nell’ordine: bianco, rosso, verde, blu, che portando l’intensità al prima al massimo e successivamente al minimo resettando il colore a cui si andrà a cambiare l’intensità in base al valore del potenziometro.

4.8 Modulo Potenziometro-RGB

Successivamente possiamo vedere il metodo più interessante, cioè il resetterRGB():

```
void RGB::resetterRGB(int val)
{
    bool maxed = false;
    int reset = 0;
    int intensity = 0;

    if(val > 1000){
        maxed = true;
    }

    if(maxed && val < 2){
        reset++;
        maxed = false;
    }
    if(reset%4 == 0){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, intensity);
        analogWrite(greenPin, intensity);
        analogWrite(bluePin, intensity);
    }else if(reset%4 == 1){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, intensity);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, 255);
    }else if(reset%4 == 2){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, intensity);
        analogWrite(bluePin, 255);
    }else if(reset%4 == 3){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, intensity);
    }

}
```

Quanodo il potenziometro ha raggiunto almeno una voltà l'intensità massima,e sucessivamente quella minima, viene incrementato il valore reset, che col %4 definirà in che fase di colore si trova.

5 Test

5.1 Protocollo di test

Test Case: **Riferimento**:	TC-01 REQ-01	**Nome:**	Le librerie vengono richiamate e non danno errori
Descrizione:	È importante che le librerie vengano create correttamente, e che è possibile richiamarle all'interno di dei file .INO, in questo modo ci assicuriamo che le Librerie sono funzionanti.		
Prerequisiti:	Una qualsiasi libreria, una classe vuota, che richiami solo la libreria.		
Procedura:	1. Creare una classe (file .INO) 2. Richiamare la libreria 3. Compilare il codice		
Risultati attesi:	1. La classe verrà creata correttamente 2. La libreria verrà richiamata correttamente 3. Il compilatore non segnalerà errori.		

Test Case: **Riferimento**:	TC-02 REQ-08	**Nome:**	Seconda funzione del primo modulo: Potenziometro-Buzzer, setOnBuzzerFrequenze()
Descrizione:	Le funzioni del primo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del potenziometro e del Buzzer, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione. 3. Diminuire il valore del potenziometro		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. La frequenza del Buzzer aumenterà. 3. La frequenza del Buzzer diminuirà.		

Test Case: **Riferimento**:	TC-03 REQ-08	**Nome:**	Terza funzione del primo modulo: Potenziometro-Buzzer, setOnReverseBuzzerFrequenze()
Descrizione:	Le funzioni del primo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del potenziometro e del Buzzer, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione. 3. Diminuire il valore del potenziometro		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. La frequenza del Buzzer diminuirà. 3. La frequenza del Buzzer aumenterà.		

Test Case: **Riferimento**:	TC-04 REQ-09	**Nome:**	Seconda funzione del secondo modulo: Bottone-Led, blink()
Descrizione:	Le funzioni del secondo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Bottone e del Led, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Tenere premuto il bottone. 3. Rilasciare il bottone.		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. Il Led lampeggerà ad una certa frequenza. 3. Il Led smetterà di lampeggiare.		

Test Case: **Riferimento**:	TC-05 REQ-09	**Nome:**	Terza funzione del secondo modulo: Bottone-Led, toggle ()
Descrizione:	Le funzioni del secondo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Bottone e del Led, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Cliccare più volte il Bottone		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. Il Led si accende e si spegne ad ogni click.		

Test Case:	TC-06	**Nome:**	Prima funzione del terzo modulo: Potenziometro-RGB, digitalRGB()
Riferimento:	REQ-10		
Descrizione:	Le funzioni del terzo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Potenziometro e del RGB, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione. 3. Diminuire il valore del potenziometro tramite la rotazione.		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. Il Led si accenderanno e spegneranno nel seguente ordine: rosso, verde, blu e bianco. 3. Il Led si accenderanno e spegneranno nel seguente ordine: bianco, blu, verde e rosso.		

Test Case:	TC-07	**Nome:**	Seconda funzione del terzo modulo: Potenziometro-RGB, analogRGB()
Riferimento:	REQ-10		
Descrizione:	Le funzioni del terzo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Potenziometro e del RGB, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione. 3. Diminuire il valore del potenziometro tramite la rotazione.		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. I Led cambieranno la propria intensità dalla massima alla minima (led spento) nel seguente ordine: rosso, verde, blu e bianco. 3. I Led cambieranno la propria intensità dalla massima alla minima (led spento) nel seguente ordine: bianco, blu, verde e rosso.		

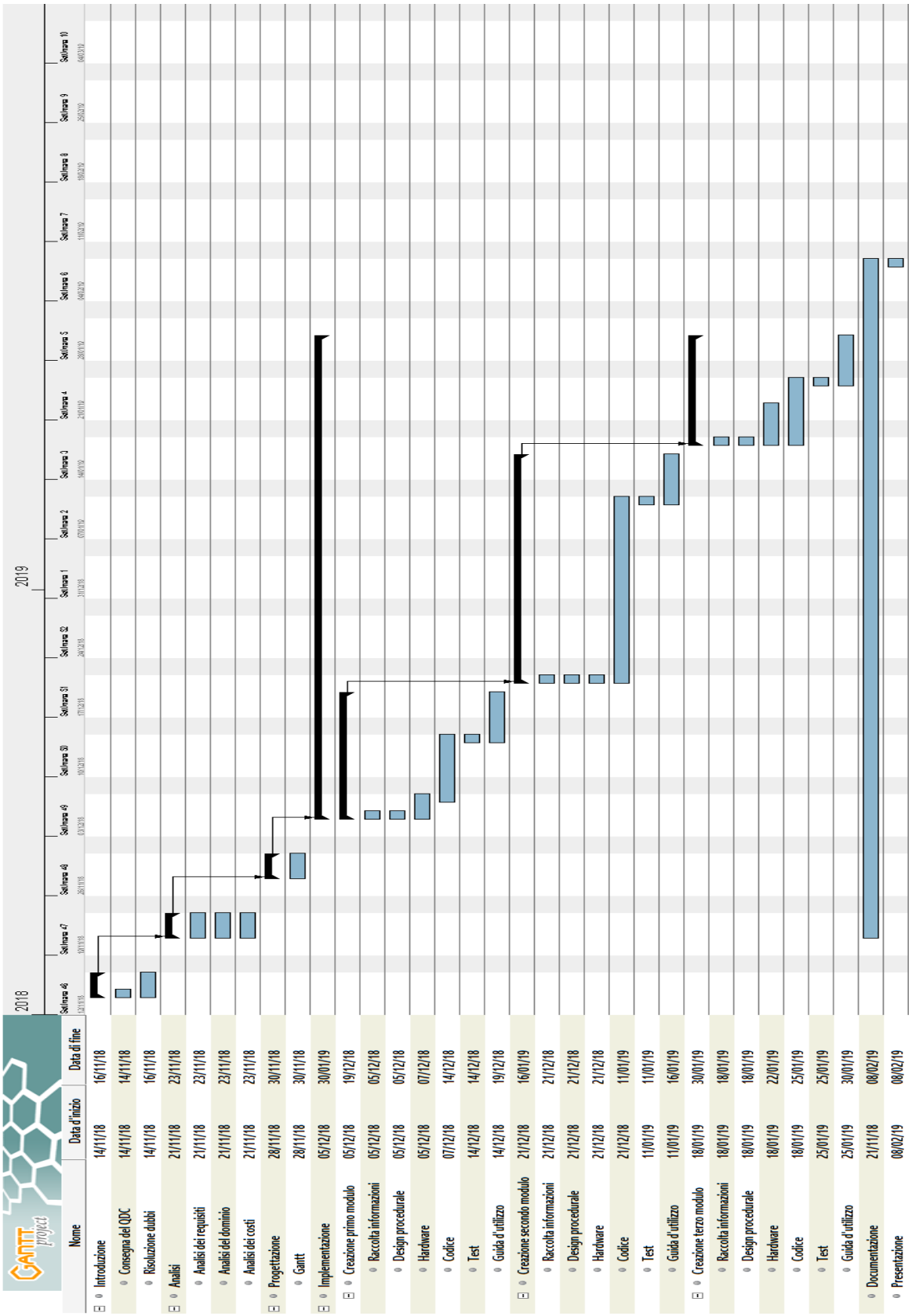
Test Case:	TC-08	**Nome:**	Terza funzione del terzo modulo: Potenziometro-RGB, resetterRGB()
Riferimento:	REQ-10		
Descrizione:	Le funzioni del terzo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Potenziometro e del RGB, componenti e Digispark.		
Procedura:	1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione al massimo. 3. Diminuire il valore del potenziometro tramite la rotazione fino al minimo. 4. Ripetere il punto 2 e 3 più volte		
Risultati attesi:	1. Il programma viene compilato correttamente. 2. L'intensità di colore dei led aumenterà. 3. L'intensità dei colori dei led diminuirà al minimo. 4. Il ogni volta che si compiono questi 2 passaggi il colore dei led cambia.		

5.2 Risultati test

Test Case	Nr. Passaggio	Risultato
TC-01	1	La classe di esempio viene creata correttamente.
	2	Le Librerie vengono richiamate correttamente.
	3	Il codice viene compilato correttamente.
TC-02	1	La classe di esempio viene compilata correttamente.
	2	La frequenza del buzzer aumenta correttamente
	3	La frequenza del buzzer diminuisce correttamente
TC-03	1	La classe di esempio viene compilata correttamente.
	2	La frequenza del buzzer diminuisce correttamente
	3	La frequenza del buzzer aumenta correttamente
TC-04	1	La classe di esempio viene compilata correttamente.
	2	Il Led lampeggia correttamente
	3	Il Led rimane spento
TC-05	1	La classe di esempio viene compilata correttamente.
	2	Il Led si accende e si spegne ad ogni click del bottone.
TC-06	1	La classe di esempio viene compilata correttamente.
	2	Il Led si accendono e spengono nel seguente ordine: rosso, verde, blu e bianco.
	3	Il Led si accendono e spengono nel seguente ordine: bianco, blu, verde e rosso.
TC-07	1	La classe di esempio viene compilata correttamente.
	2	I Led cambiano la propria intensità dalla massima alla minima (led spento) nel seguente ordine: rosso, verde, blu e bianco.
	3	I Led cambiano la propria intensità dalla massima alla minima (led spento) nel seguente ordine: bianco, blu, verde e rosso.

TC-08	1	La classe di esempio viene compilata correttamente.
	2	L'intensità dei colori aumenta
	3	L'intensità dei colori diminuisce
	4	Ogni volta che si porta l'intensità al massimo e poi al minimo, il colore che cambia intensità viene cambiato nel seguente ordine: bianco, rosso, verde, blu. Per poi ricominciare.

6 Consuntivo



Rispetto alla pianificazione ci sono stati dei cambiamenti notevoli, poiché anche la consegna è stata posticipata di due settimane, e siamo riusciti a concludere 3 moduli (numero minimo di moduli pianificati).
inoltre, oltre le 2 settimane di posticipo di consegna, il tempo di creazione di ogni modulo e delle sue rispettive librerie si è ingrandito molto in tutte.
Senza calcolare il secondo modulo, dove siamo stati interrotti dalle vacanze di natale, il terzo anche se sarebbe dovuto essere quello più elaborato date le funzioni più complicate delle altre, avevamo già comunque utilizzato lo stesso componente(potenziometro), utilizzato nel primo modulo, che quindi ci ha fatto risparmiare del tempo nella creazione della sua libreria.

7 Conclusioni

Questo progetto offre ai ragazzi delle medie, ma soprattutto a qualsiasi utente finale, la possibilità di approcciarsi in modo facile e diretto a quello che è il mondo dell'informatica.
Esso concede all'utente, la possibilità di utilizzare 3 moduli di componenti, che implementano più funzioni.
Da questo progetto abbiamo imparato come creare delle librerie per Arduino, ed ovviamente utilizzarle, posso dire che questo lavoro ci ha resi attenti all'importanza di lavorare con un'altra persona, con ritmi diversi da quelli propri, questo ci ha aiutato a capire l'importanza della coordinazione e cooperazione.

7.1 Sviluppi futuri

Per migliorare il prodotto si potrebbe semplicemente ampliare il numero delle librerie, con i suoi effettivi moduli ed ovviamente inserendogli nella guida.
Sarebbe comodo per i futuri utenti finali, allegare una guida per il montaggio dei componenti su digispark, in questo modo anche chi ha meno manualità, o paura di fare errori o anche solo danneggiare il digispark, potrebbe usufruirne con maggiore sicurezza.

7.2 Considerazioni personali

In questo progetto abbiamo imparato ad utilizzare e programmare alcuni componenti dell'Arduino, con cui non avevamo mai avuto a che fare, ma cosa più importante abbiamo appreso tutto quello che riguardava la creazione delle librerie, che era un requisito, che all'inizio di questo progetto ci mancava.
In più ci siamo sentiti più motivati nella creazione di questo progetto che aveva uno scopo concreto.

8 Bibliografia

8.1 Sitografia

1. https://www.adrirobot.it/arduino/digispark/digispark.htm , da 14.11.2019 al 08.02.2019
2. http://digistump.com/products/1, da 14.11.2019 al 08.02.2019
3. https://www.arduino.cc/reference/en/language/functions/math/map/ , da 14.11.2019 al 08.02.2019

9 Allegati

9.1 Elenco degli allegati

- Diari di lavoro
- Codici sorgente
- Quaderno dei compiti
- Prodotto (Librerie e Guida)
- Presentazione