

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.....	3
1.1	Informazioni sul progetto	3
1.2	Abstract	3
1.3	Scopo	3
2	Analisi.....	4
2.1	Analisi del dominio	4
2.2	Analisi e specifica dei requisiti	4
2.3	Pianificazione.....	2
2.4	Analisi dei mezzi	3
2.4.1	Software	3
2.4.2	Hardware	3
3	Progettazione.....	4
3.1	Design dell'architettura del sistema	4
3.2	Design procedurale	5
3.2.1	Libreria potenziometro	5
3.2.2	Libreria buzzer	6
3.2.3	Libreria bottone	7
3.2.4	Libreria Led.....	8
3.2.5	Libreria RGB	9
4	Implementazione.....	10
4.1	Libreria Potenziometro	10
4.2	Libreria Buzzer.....	10
4.3	Modulo Potenziometro-Buzzer	10
4.4	Libreria Bottone	10
4.5	Libreria LED.....	11
4.6	Modulo Bottone-LED.....	11
4.7	Libreria RGB.....	12
4.8	Modulo Potenziometro-RGB.....	13
5	Test	14
5.1	Protocollo di test	14
5.2	Risultati test	17
6	Consuntivo.....	19
7	Conclusioni	21
7.1	Sviluppi futuri	21
7.2	Considerazioni personali	21
8	Bibliografia.....	21
8.1	Sitografia	21
9	Allegati.....	21
9.1	Elenco degli allegati	21

1 Introduzione

1.1 Informazioni sul progetto

Allievi coinvolti: Thor Düblin, 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.

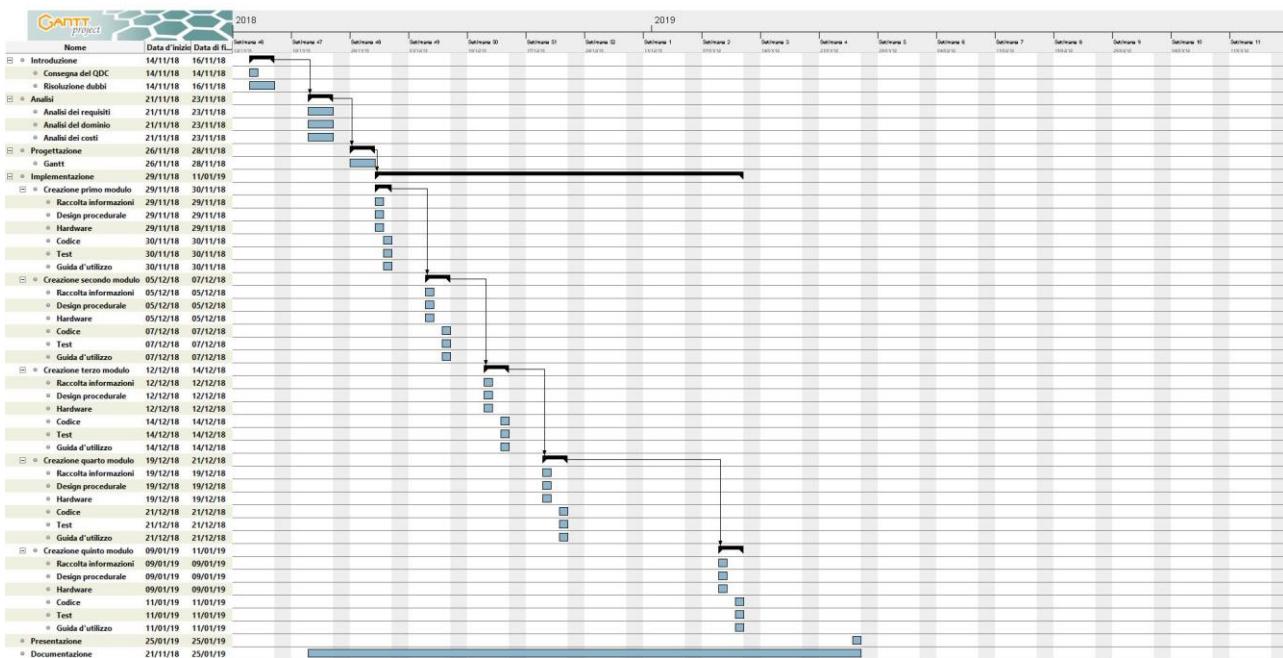
ID: REQ-07	
Nome	Moduli
Priorità	2
Versione	1.0
Note	Con moduli si intende i componenti che useremo insieme per creare delle funzioni.
Sotto requisiti	
001	Ci saranno 3 moduli
002	Ogni modulo deve avere 3 funzioni con i rispettivi livelli di difficoltà: facile, medio, difficile.

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:

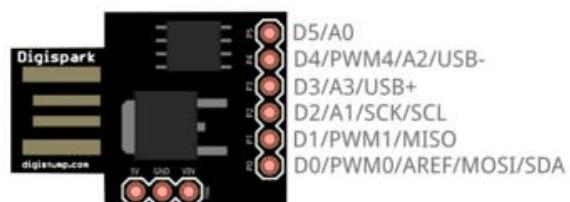
- Arduino (Arduino.h) versione incorporata nell'IDE Arduino 1.8.5 per il linguaggio C++, che permette di scrivere codice in C++ utilizzando i metodi e le funzioni di Arduino

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



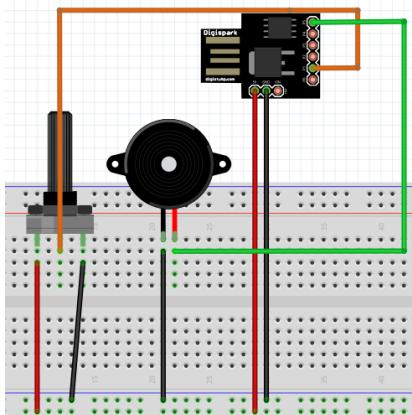
PIN 0	I2C SDA, PWM
PIN 1	PWM
PIN 2	I2C SCK(PWM), Analogico(1)
PIN 3	Analogico(3)
PIN 4	PWM, Analogico(2)
PIN 5	Analogico(0)



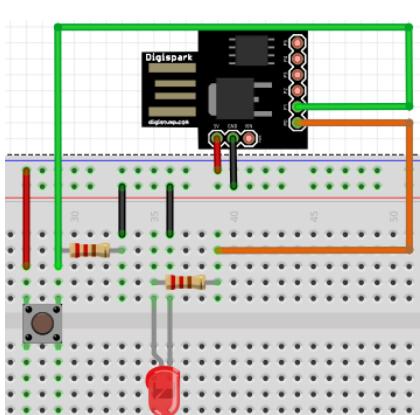
3 Progettazione

3.1 Design dell'architettura del sistema

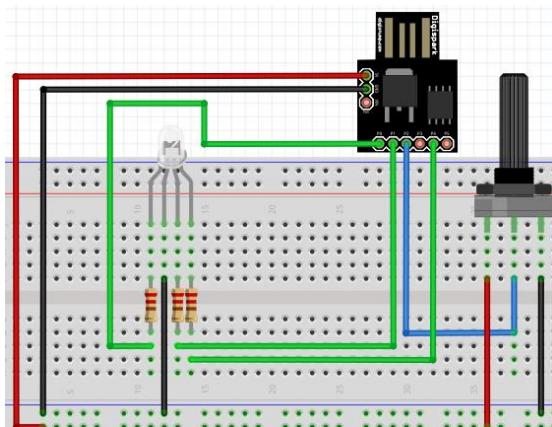
Potenziometro-Buzzer:



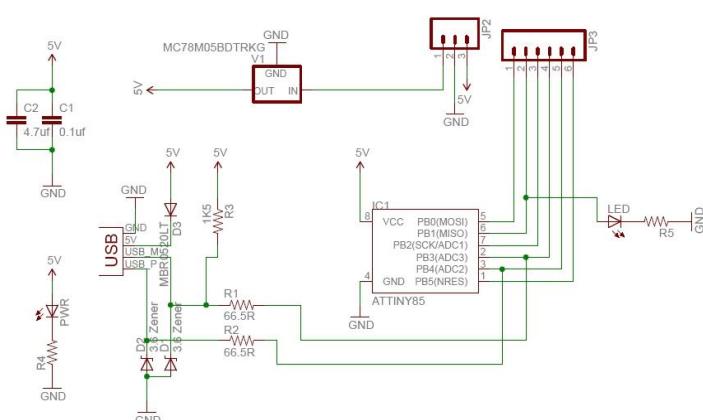
Led-Bottone:



RGB - Potenziometro:



Schema Digispark:



3.2 Design procedurale

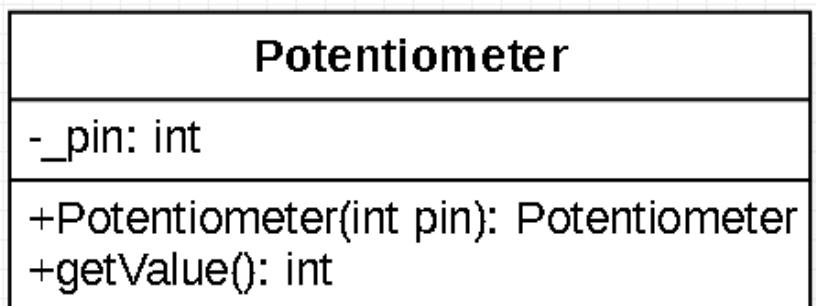
3.2.1 Libreria potenziometro

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.



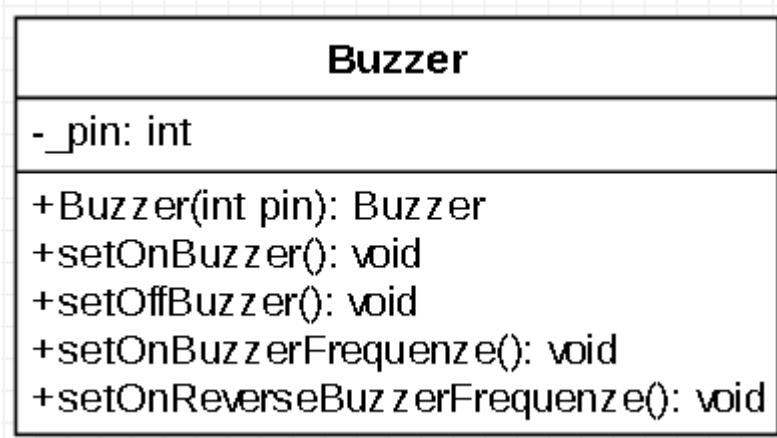
3.2.2 Libreria buzzer

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.



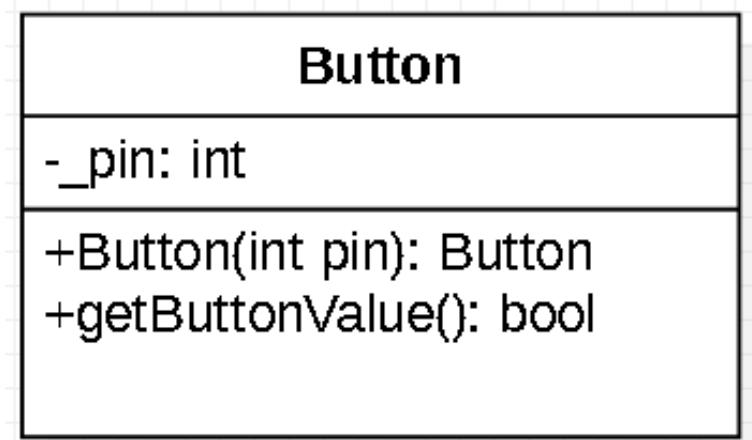
3.2.3 Libreria bottone

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

Led
-_pin: int
+Led(int pin): Led
+ledOn(): void
+ledOff(): void
+blink(int milliseconds): void
+toggle(int buttonPin): void
+getState(): bool

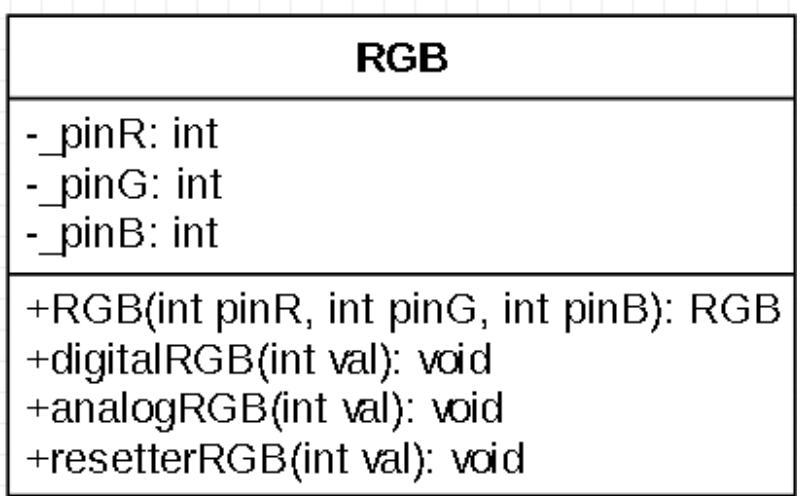
3.2.5 Libreria RGB

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 setOnReverseFrequency():

```
/**  
 * 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 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);
    }
}
```

Quando il potenziometro ha raggiunto almeno una volta 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:	TC-01	Nome:	Le librerie vengono richiamate e non danno errori
Riferimento:	REQ-01		
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:	<ol style="list-style-type: none"> 1. Creare una classe (file .INO) 2. Richiamare la libreria 3. Compilare il codice 		
Risultati attesi:	<ol style="list-style-type: none"> 1. La classe verrà creata correttamente 2. La libreria verrà richiamata correttamente 3. Il compilatore non segnalerà errori. 		

Test Case:	TC-02	Nome:	Seconda funzione del primo modulo: Potenziometro-Buzzer, setOnBuzzerFrequenze()
Riferimento:	REQ-08		
Descrizione:	Le funzioni del primo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del potenziometro e del Buzzer, componenti e Digispark.		
Procedura:	<ol style="list-style-type: none"> 1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione. 3. Diminuire il valore del potenziometro 		
Risultati attesi:	<ol style="list-style-type: none"> 1. Il programma viene compilato correttamente. 2. La frequenza del Buzzer aumenterà. 3. La frequenza del Buzzer diminuirà. 		

Test Case:	TC-03	Nome:	Terza funzione del primo modulo: Potenziometro-Buzzer, setOnReverseBuzzerFrequenze()	
Riferimento:	REQ-08	Descrizione: Le funzioni del primo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del potenziometro e del Buzzer, componenti e Digispark.			
Procedura:	<ol style="list-style-type: none"> 1. Compilare il programma con Digispark. 2. Aumentare il valore del potenziometro tramite la rotazione. 3. Diminuire il valore del potenziometro 			
Risultati attesi:	<ol style="list-style-type: none"> 1. Il programma viene compilato correttamente. 2. La frequenza del Buzzer diminuirà. 3. La frequenza del Buzzer aumenterà. 			

Test Case:	TC-04	Nome:	Seconda funzione del secondo modulo: Bottone-Led, blink()	
Riferimento:	REQ-09	Descrizione: Le funzioni del secondo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Bottone e del Led, componenti e Digispark.			
Procedura:	<ol style="list-style-type: none"> 1. Compilare il programma con Digispark. 2. Tenere premuto il bottone. 3. Rilasciare il bottone. 			
Risultati attesi:	<ol style="list-style-type: none"> 1. Il programma viene compilato correttamente. 2. Il Led lampeggerà ad una certa frequenza. 3. Il Led smetterà di lampeggiare. 			

Test Case:	TC-05	Nome:	Terza funzione del secondo modulo: Bottone-Led, toggle ()	
Riferimento:	REQ-09	Descrizione: Le funzioni del secondo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Bottone e del Led, componenti e Digispark.			
Procedura:	<ol style="list-style-type: none"> 1. Compilare il programma con Digispark. 2. Cliccare più volte il Bottone 			
Risultati attesi:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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. 			

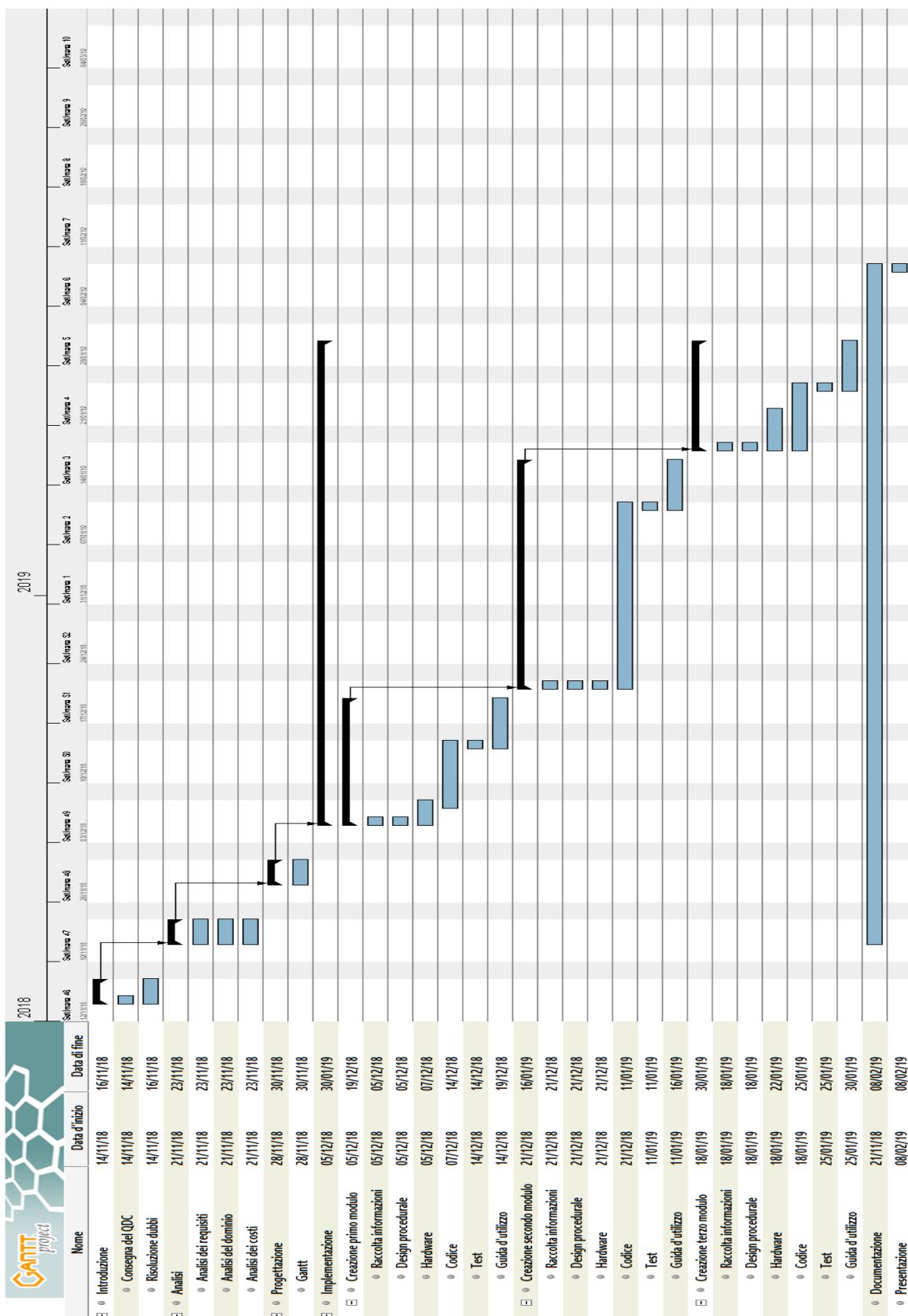
Test Case:	TC-08	Nome:	Terza funzione del terzo modulo: Potenziometro-RGB, resetterRGB()	
Riferimento:	REQ-10	Descrizione: Le funzioni del terzo modulo devono funzionare correttamente.		
Prerequisiti:	Libreria del Potenziometro e del RGB, componenti e Digispark.			
Procedura:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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 lampeggiava 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

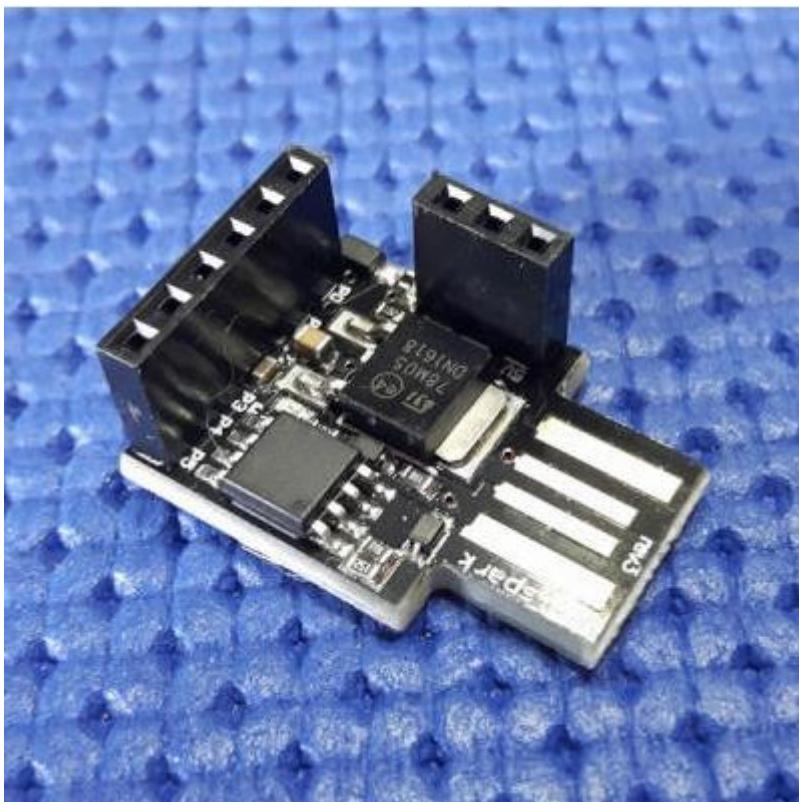
- Diari di lavoro
- Codici sorgente
- Quaderno dei compiti
- Prodotto (Librerie e Guida)
- Presentazione

Guida per l'utilizzo delle librerie

Questa guida serve a spiegare i passi della programmazione con Arduino.

Driver Digispark

Per questo progetto viene utilizzata la scheda Digispark:



I due connettori si devono saldare sulla scheda perché all'acquisto ciò non è presente per il semplice fatto che vengono scelti in base alle proprie esigenze.

I pin della scheda sono i seguenti:

- 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)

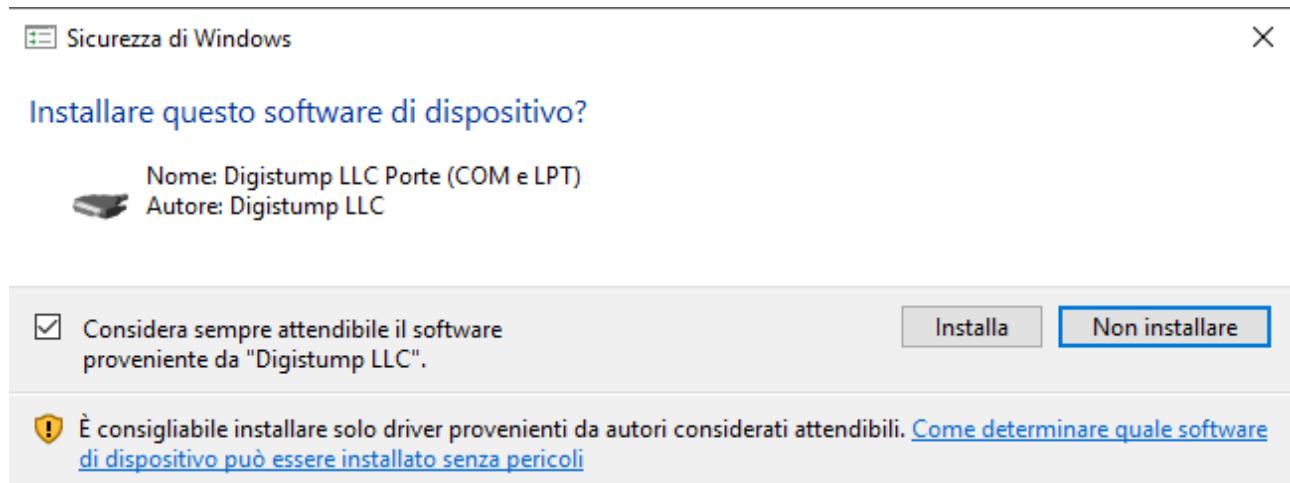
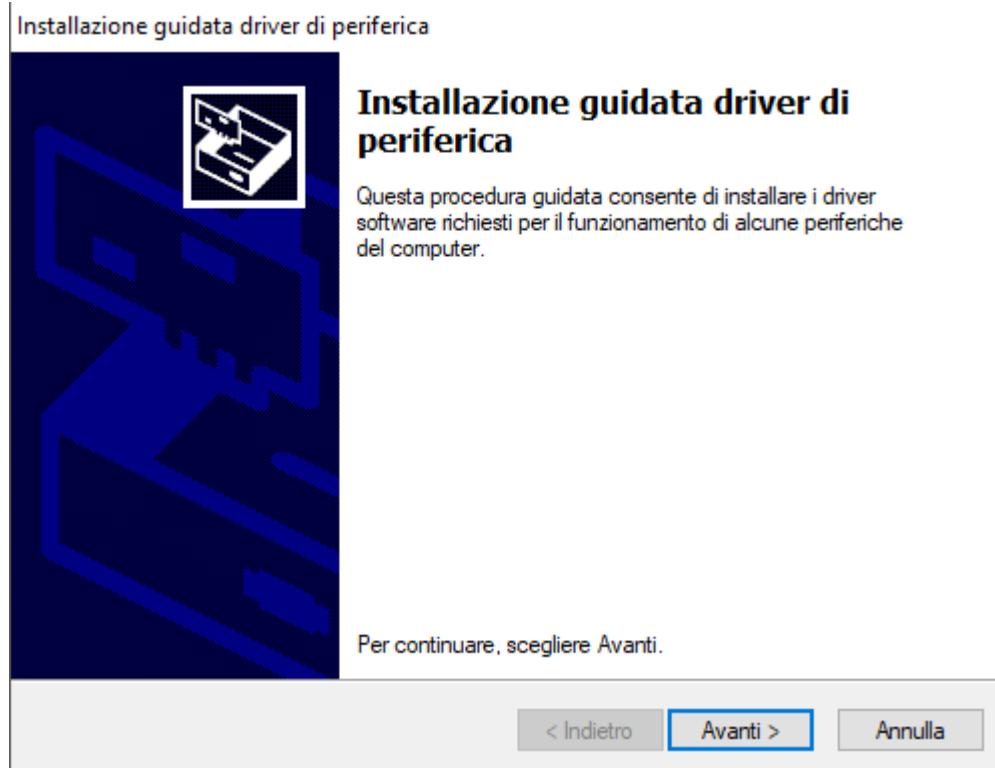
Per utilizzare questa scheda bisogna scaricare i drivers e ciò viene fatto da internet.

È necessario scaricare e installare manualmente i driver per la scheda Digispark. Bisogna scaricare, decomprimere ed eseguire "Install Drivers" (sui sistemi a 32 bit) o "DPIinst64" (sui sistemi a 64 bit). Il link

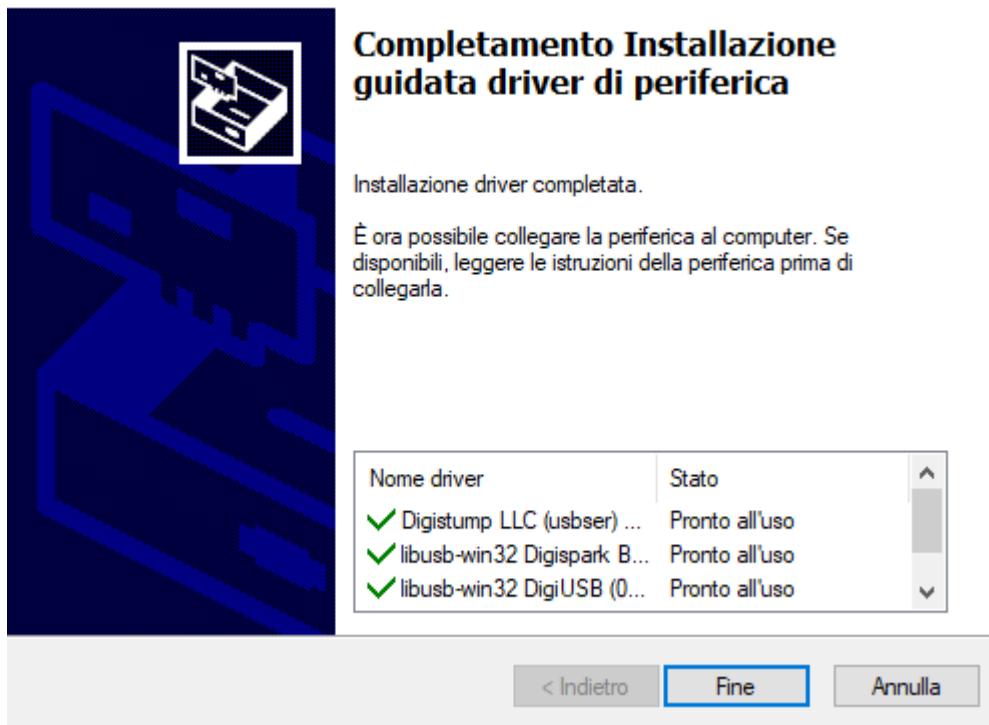
dal quale scaricare i Drivers è:

(<https://github.com/digistump/DigistumpArduino/releases/download/1.6.7/Digistump.Drivers.zip>).

Una volta eseguito il programma appare la finestra dell'installazione guidata, seguire i passi mostrati di seguito:

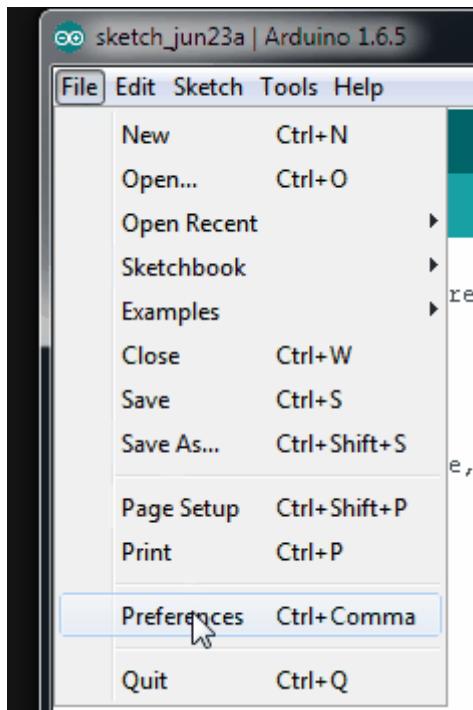


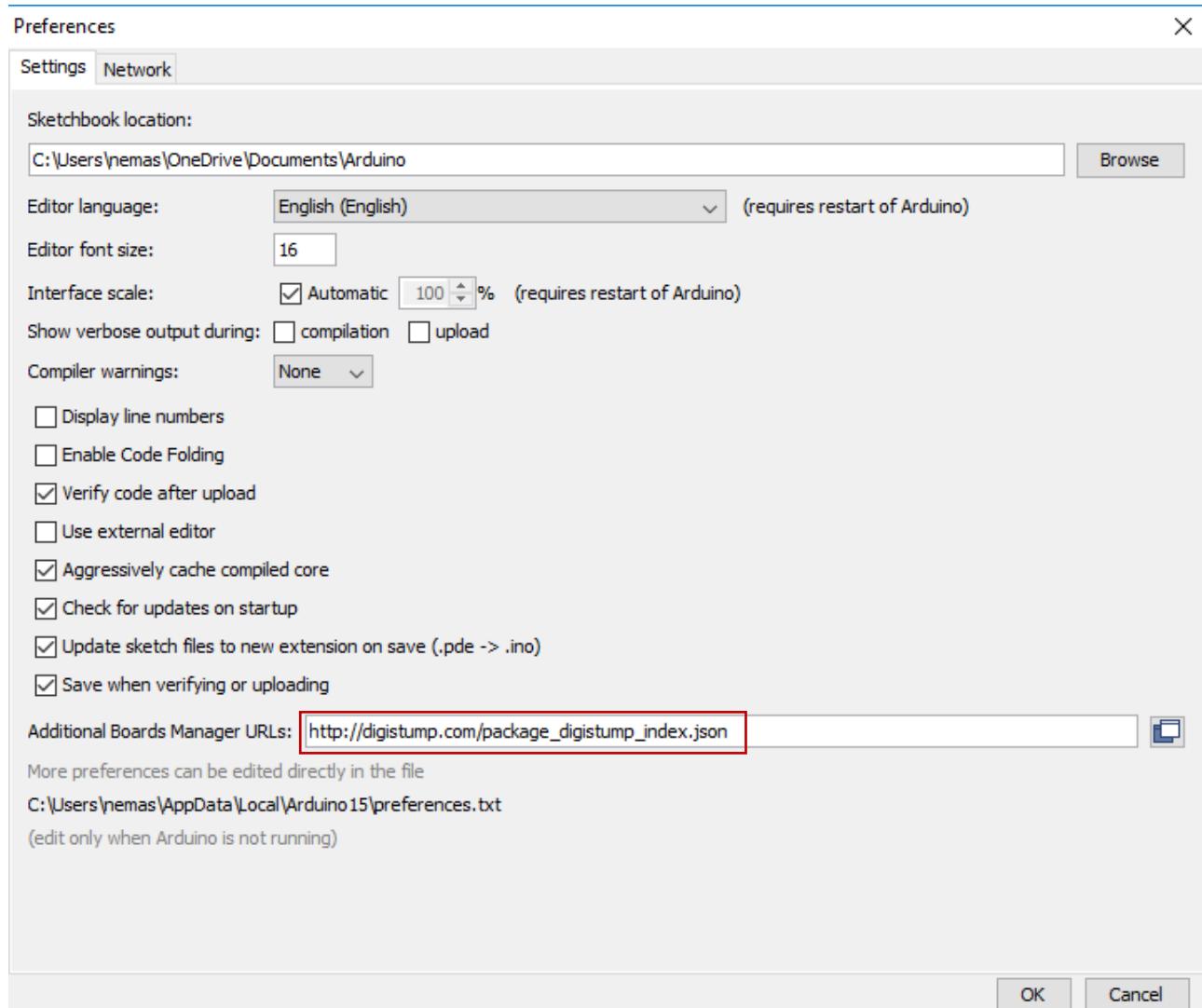
Installazione guidata driver di periferica



Quando la scheda verrà inserita in una porta USB senza che ciò venga richiesto dall'IDE è possibile che la scheda non venga riconosciuta ma è normale.

Una volta finita l'installazione, aprire il programma di Arduino e nel menu cliccare sulla voce "File" poiché scegliere "Preferences" oppure "Impostazioni".

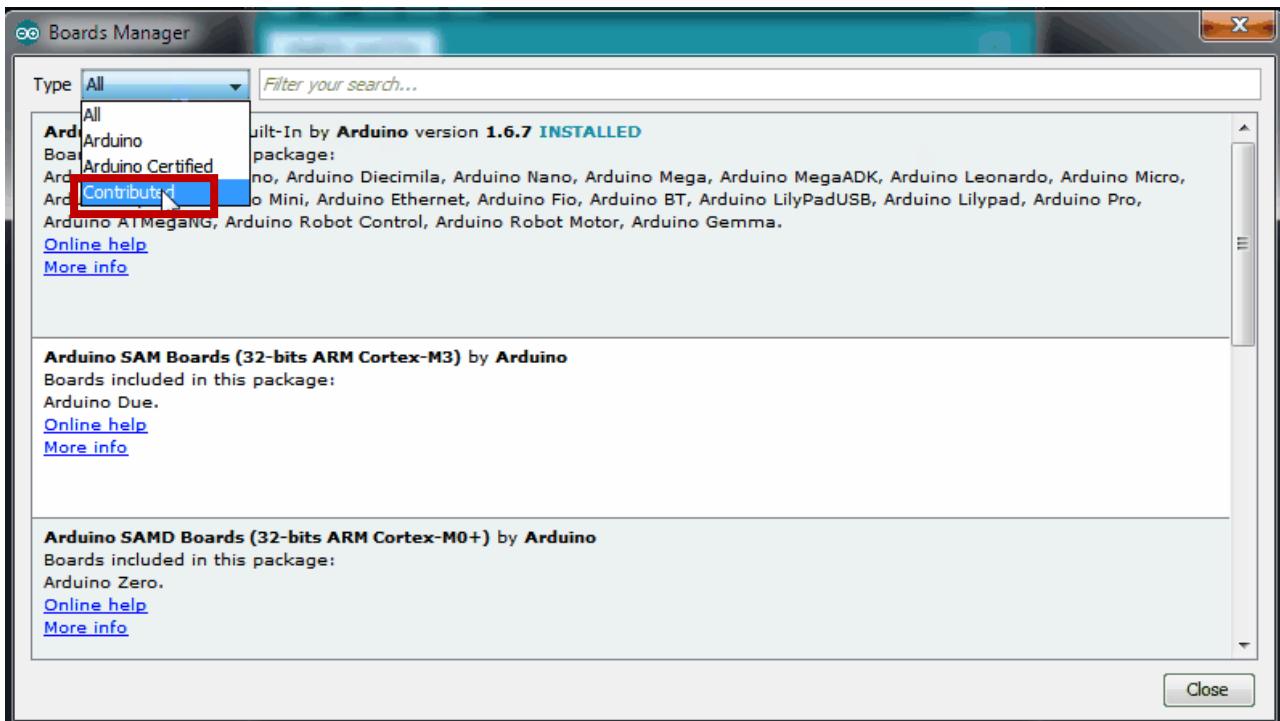




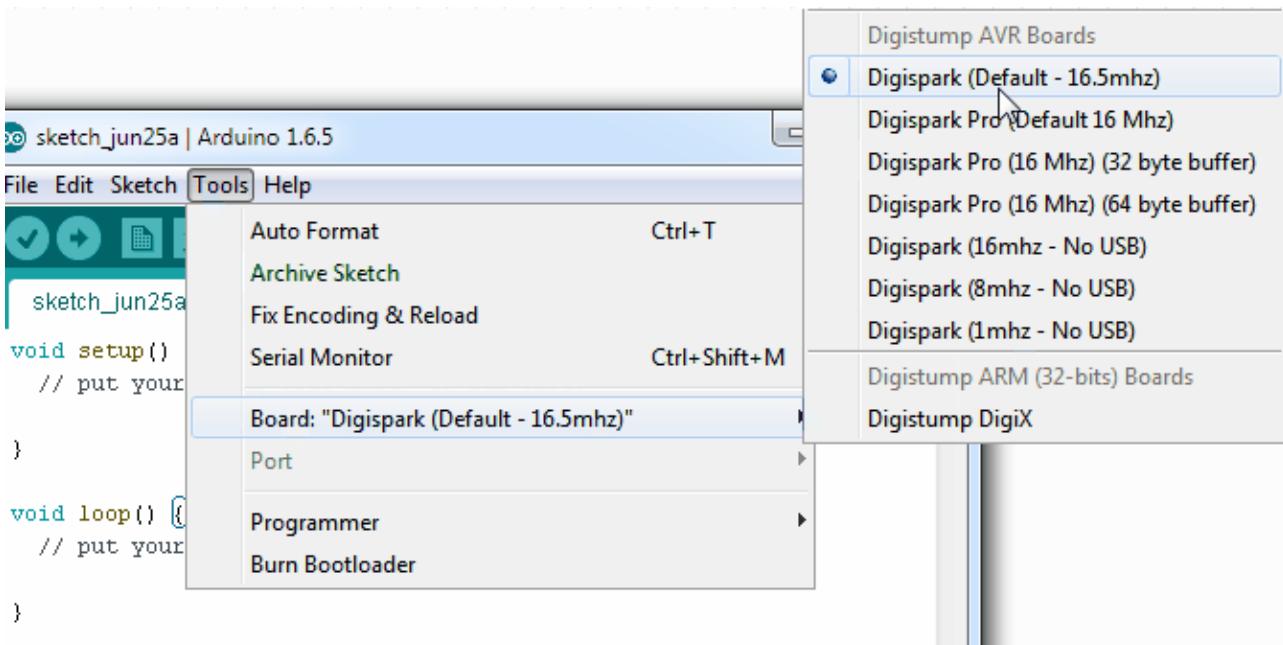
Nella casella con l'indicazione “URL” Inserire il seguente

http://digistump.com/package_digistump_index.json e cliccare su OK.

Quando questo viene terminato, nel menu scegliere “Strumenti” e poi in “Gestore schede”, quindi scegliere il tipo, il quale deve essere:



Una volta finito bisogna scegliere questa scheda per poterla utilizzare:



A questo punto l'installazione è completa.

Uso della scheda Digispark con l'IDE

Questa scheda funziona in un modo diverso rispetto agli altri prodotti Arduino. La programmazione segue una procedura diversa.

- Come prima cosa bisogna assicurarsi che il sia selezionata la scheda Digispark.
- Utilizzare un codice (scriverlo o aprire uno già creato in precedenza)
- Caricare il programma/codice. Dopo la compilazione sarà richiesto di inserire il vostro Digispark, a questo punto collegarlo oppure scollarlo e ricollegarlo.

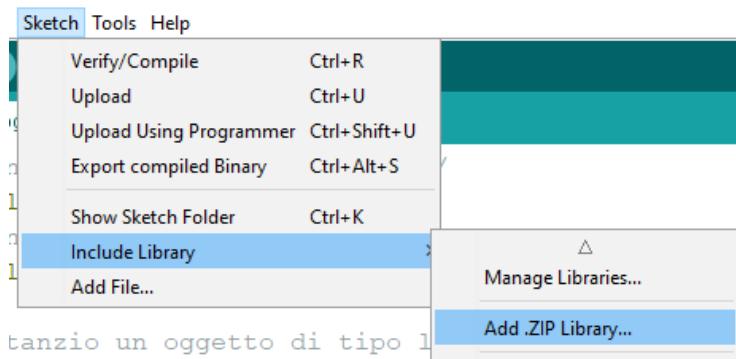
```
Le variabili globali usano 9 byte di memoria dinamica.  
Running Digispark Uploader...  
Plug in device now... (will timeout in 60 seconds)
```

Una volta inserita la scheda e se tutto andrà a buon fine apparirà la scritta “Caricamento terminato” e il codice sarà eseguito sul Digispark.

```
Caricamento completato  
> Starting the user app ...  
running: 100% complete  
>> Micronucleus done. Thank you!
```

Come integrare le librerie:

Per integrare una libreria bisogna:



In questo modo si aggiunge la libreria desiderata.

Bisogna assicurarsi che all'inizio del programma sia inclusa la libreria nel modo seguente:

```
#include <Button.h> (#include <"NomeLibreria".h>)
```

Il prossimo passo è istanziare un oggetto della libreria che si sta utilizzato, per esempio:

```
Button button(0); (Libreria "nomeogetto"(pin))
```

Per utilizzare un metodo della libreria bisogna utilizzare l'oggetto istanziato precedentemente, quindi **oggetto.metodo()** oppure come nel esempio di seguito:

```
bool state = button.getButtonValue();
```

Questo pezzo di codice ritorna il valore del bottone.

Attuatori:

- Potenziometro
- Buzzer
- Bottone
- Led
- Led RGB

Potenziometro

Il **potenziometro** è un dispositivo elettronico equivalente ad un partitore di tensione resistivo variabile (cioè a due resistori collegati in serie, aventi la somma dei due valori di resistenza costante, ma di cui può variare il valore relativo), difatti una sua parte viene disposta in parallelo al carico utilizzatore.

Buzzer

Un **buzzer** è un dispositivo di segnalazione audio, che può essere meccanico, elettromeccanico, o piezoelettrico (abbreviato anche come piezo). I tipici utilizzi del buzzer includono dispositivi di allarme, timer, e PC speaker per i feedback sugli input dell'utente, come pressione dei tasti o click del mouse, nei vecchi personal computer.

Bottone

Il bottone è un dispositivo elettronico con una sola posizione di riposo (monostabile), una volta azionato una molla lo riporta alla posizione di partenza appena viene rilasciato. Una volta premuto esso aziona qualcosa (Accensione della luce, suonare il campanello).

Led

In elettronica il LED (sigla inglese di Light Emitting Diode[1]) o diodo a emissione di luce è un dispositivo optoelettronico che sfrutta la capacità di alcuni materiali semiconduttori di produrre fotoni attraverso un fenomeno di emissione spontanea.

Led RGB

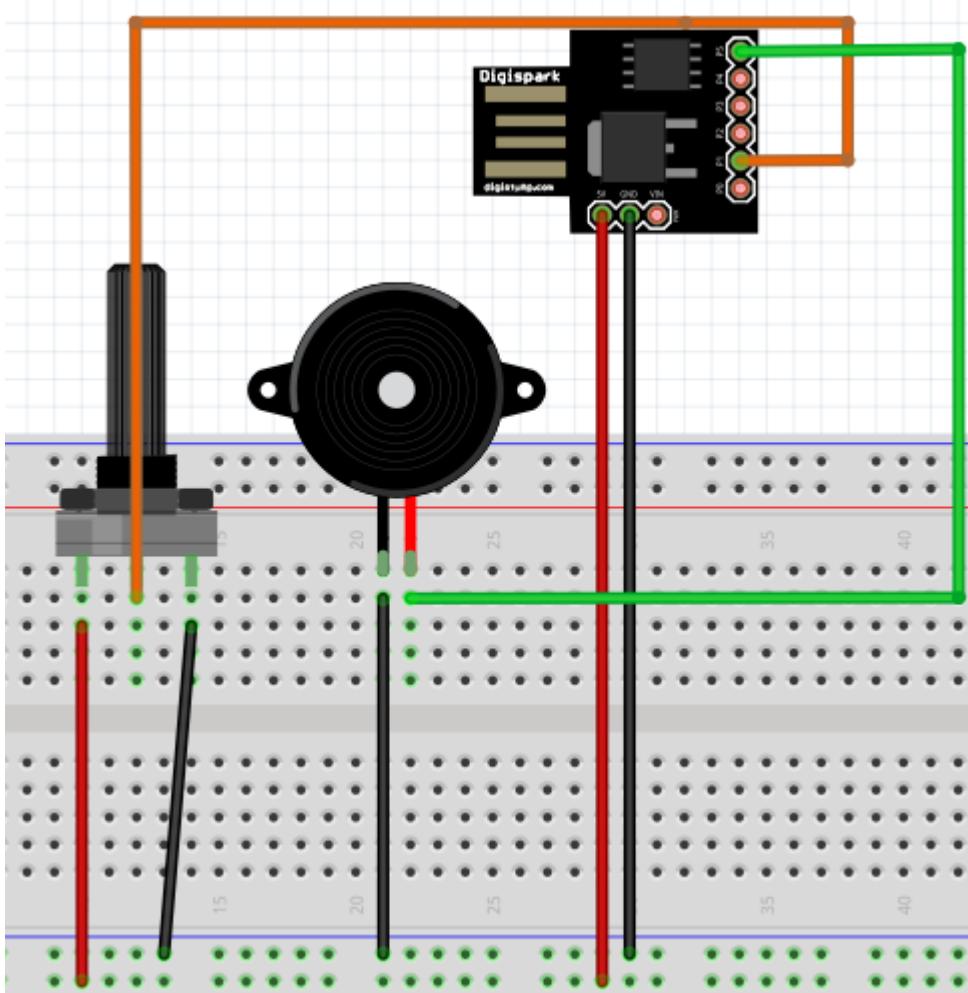
LED RGB indica i **LED** rossi, blu e verdi. I **LED RGB** combinano questi tre colori per produrre oltre 16 milioni di tonalità di luce. Non è possibile riprodurre tutti i colori. Alcuni di questi non rientrano nel triangolo formato dai LED RGB. Inoltre, i colori pigmentati come il marrone o il rosa sono difficili, se non impossibili, da ottenere.

Combinazioni di librerie

Buzzer e Potenziometro

Prima di iniziare bisogna assicurarsi che i driver del Digispark siano installati sul PC (Vedi primo capitolo “Driver Digispark”).

Prima di scrivere il codice bisogna seguire i passi del capito sull'integrazione delle librerie (Vedi secondo capitolo “Come integrare le librerie”).



Per questo circuito si necessită: 1 buzzer, 1 potenziometro, scheda Digispark.

Il potenziometro è collegato alla massa, al “5v” e al pin “P1” della scheda Digispark. Il Buzzer è collegato alla massa e al pin “P5” di Digispark.

Esempi:

BuzzerOnOff

```
/* Includere la libreria del Buzzer */
#include <Buzzer.h>
/* Includo la libreria del Buzzer */
#include <potentiometer.h>

/* Istanziare un oggetto di tipo Potentiometer */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo Buzzer */
Buzzer buzzer(2);

/*Metodo setup che viene avviato solo all'inizio*/
void setup() {
}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /*Ritorna il valore del potenziometro*/
    int stato = pot.getValue();

    /*Se il valore del potenziometro è minore di 500 spegne
     *il Buzzer e se invece è maggiore lo accende
     */
    if(stato < 500){
        buzzer.setOffBuzzer();
    }else{
        buzzer.setOnBuzzer();
    }
}
```

Questo esempio accende e spegne il buzzer in base al valore del potenziometro.

BuzzerPotentiometerFrequency

```

/* Includere la libreria del Buzzer */
#include <Buzzer.h>
/* Includere la libreria del Buzzer */
#include <potentiometer.h>

/* Includo la libreria del Buzzer */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo Buzzer */
Buzzer buzzer(2);

/*Setta il range*/
int range= 16000;

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {
}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /* Prendo lo stato del potenziometro */
    int stato = pot.getValue();

    /*Setto il range e lo stato del potenziometro che poi mi setta la frequenza al buzzer*/
    buzzer.freqence(range, stato);
}

```

Questo esempio riceve come parametro il range e lo stato del potenziometro e poi calcola la frequenza nel seguente modo:

```

/**
 * Ritorna la frequenza che verrà applicata al buzzer
 */
void Buzzer::freqence(int range, int potValue)
{
    int frequence = range/1024*potValue;
    tone(_pin, frequence);
}

```

Riceve come parametro il range di frequenza e il valore del potenziometro dopodiché fa il calcolo che viene salvato nella variabile “frequence”, la quale poi viene applicata come frequenza al buzzer tramite il rispettivo pin.

BuzzerPotentiometerFrequencyReverse

```

/* Includere la libreria del Buzzer */
#include <Buzzer.h>
/* Includere la libreria del Buzzer */
#include <potentiometer.h>

/* Includo la libreria del Buzzer */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo Buzzer */
Buzzer buzzer(2);

/*Setta il range*/
int range= 16000;

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {
}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /* Prendo lo stato del potenziometro */
    int stato = pot.getValue();

    /*Setto il range e lo stato del potenziometro che
     *poi mi setta la frequenza al buzzer all'inverso*/
    buzzer.setOnReverseBuzzerFrequenze(range, stato);
}

```

Questo esempio si comporta quasi nello stesso modo di quello precedente ma la frequenza sarà diversa poiché viene calcolata in un altro modo.

```

/**
 * 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);
}

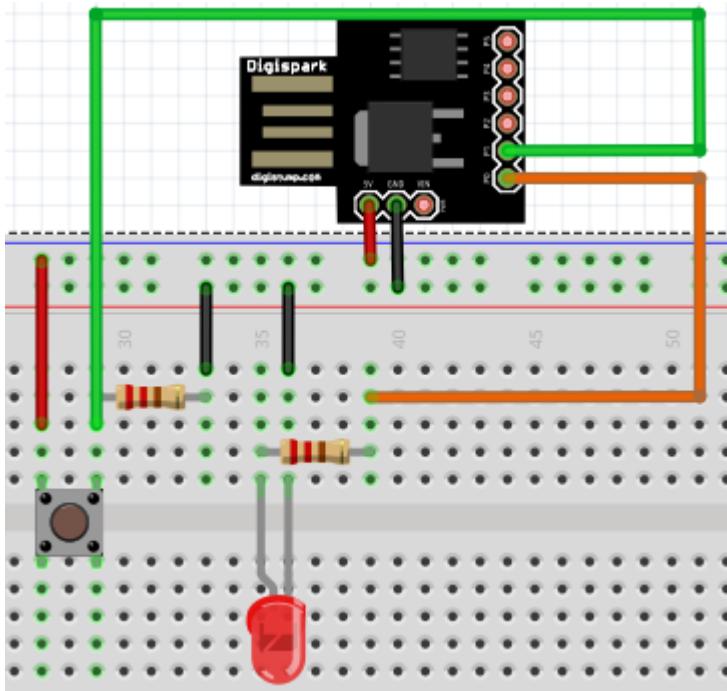
```

Riceve come parametro il range di frequenza e il valore del potenziometro dopodiché fa il calcolo che viene salvato nella variabile “frequence”, dove poi verrà fatta la differenza tra il massimo valore del potenziometro e la “perquence”, questa differenza viene applicata come frequenza al buzzer tramite il rispettivo pin.

Led e Bottone

Prima di iniziare bisogna assicurarsi che i driver del Digispark siano installati sul PC (Vedi primo capitolo “Driver Digispark”).

Prima di scrivere il codice bisogna seguire i passi del capitolo sull’integrazione delle librerie (Vedi secondo capitolo “Come integrare le librerie”).



Per questo circuito si necessită: 1 led, 1 bottone, 2 resistenze, scheda Digispark.

Il bottone è collegato alla massa tramite la resistenza da $10\text{ k}\Omega$, al “5v” e al pin “P1” della scheda Digispark.
Il led è collegato alla massa e al pin “P0” attraverso una resistenza da 330Ω .

Esempi:

LedOnOff

```
/* Include la libreria del bottone */
#include <Button.h>
/* Include la libreria del led */
#include <Led.h>

/* Istanzia un oggetto di tipo led*/
Led led(1);
/* Istanzia un oggetto di tipo button*/
Button button(0);

/* Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/* Metodo loop che viene richiamato all'infinito*/
void loop() {
    /* Prendo lo stato del bottone */
    bool state = button.getButtonValue();

    /* Se il bottone è premuto viene acceso, sennò viene spento. */
    if(state) {
        led.ledOn();
    } else {
        led.ledOff();
    }
}
```

Questo esempio accende il led se il bottone è premuto altrimenti il led rimane spento.

LedBlink

```
/* Includo la led button */
#include <Led.h>

/*Istanzia un oggetto di tipo button*/
Led led(1);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {
}

/*Variabile millisecondi che specifica ogni
*quanto bisogna fare il blink del led
*/
int milliseconds = 1000;

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /*Ritorna il valore del led*/
    bool ledstate = led.getState();

    /*Se il led è acceso allora lo fa blinkare
     *ogni (milliseconds), se non spegne il led
     */
    if(ledstate == HIGH){
        led.blink(milliseconds);
    }else{
        led.ledOff();
    }
}
```

Se il led è acceso allora esso viene fatto lampeggiare ogni tot millisecondi, i quali vengono specificati nella variabile "milliseconds". Se il led non è acceso, in quel caso rimane spento.

LedToggle

```

/* Includo la libreria button */
#include <Button.h>
/* Includo la led button */
#include <Led.h>

/*Istanzia un oggetto di tipo led*/
Led led(1);
/*Istanzia un oggetto di tipo button*/
Button button(0);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /*Ritorna il valore del bottone*/
    int buttonState = button.getButtonValue();

    /*Setta il led con il pin del buttonState*/
    led.toggle(buttonState);
}

```

Viene ricavato il pin del pulsante e poi viene dato come parametro al metodo toggle() della libreria del Led.

```

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);
    }
}

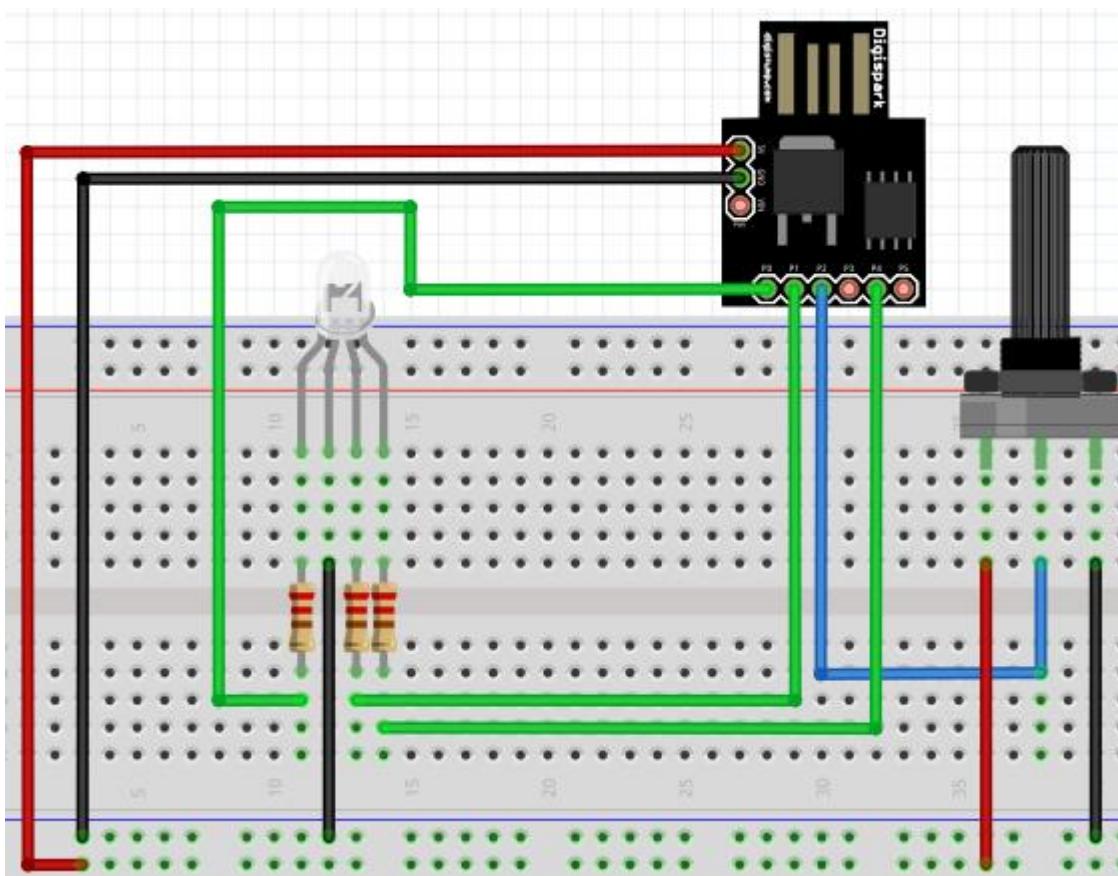
```

Il metodo toogle() si occupa di cambiare lo stato del pulsante quando viene cliccato.

Led RGB e Potenziometro

Prima di iniziare bisogna assicurarsi che i driver del Digispark siano installati sul PC (Vedi primo capitolo “Driver Digispark”).

Prima di scrivere il codice bisogna seguire i passi del capitolo sull’integrazione delle librerie (Vedi secondo capitolo “Come integrare le librerie”).



Per questo circuito si necessită: 1 led RGB, 1 potenziometro, 3 resistenze, scheda Digispark.

Il potenziometro è collegato alla massa, al “5v” e al pin “P2” della scheda Digispark. Il led è collegato alla massa e al pin “P0”, al “P1” e al “P4” attraverso 3 resistenze da 330Ω .

Esempi:

AnalogRGB

```

/* Includere la libreria del Buzzer */
#include <RGB.h>
/* Includere la libreria del potenziometro */
#include <potentiometer.h>

/* Includo la libreria del potenziometro */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo rgb */
RGB rgb(0,1,4);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /* Prendo lo stato del potenziometro */
    int stato = pot.getValue();

    /*Setto il valore del potenziometro al metodo analogRGB della libreria RGB*/
    rgb.analogRGB(stato);
}

```

Si occupa di ricavare lo stato del potenziometro e poi di settare questo valore al metodo analogRGB()

```

void RGB::analogRGB(int val)
{
    if(val > 854){
        intensity = map(val-854,0,170,255,0);
        analogWrite(redPin, 0);
        analogWrite(greenPin, 255-intensity);
        analogWrite(bluePin, 255);
    }else if(val > 680){
        intensity = map(val-680,0,170,255,0);
        analogWrite(redPin, intensity);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 255);
    }else if(val > 510){
        intensity = map(val-510,0,170,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 255-intensity);
    }else if(val > 340){
        intensity = map(val-340,0,170,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, intensity);
        analogWrite(bluePin, 0);
    }else if(val > 170){
        intensity = map(val-170,0,170,255,0);
        analogWrite(redPin, 255-intensity);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, 0);
    }else{
        intensity = map(val,0,170,255,0);
        analogWrite(redPin, 0);
        analogWrite(greenPin, 255-intensity);
        analogWrite(bluePin, 0);
    }
}

```

In base al valore del potenziometro val (stato del potenziometro passato come parametro), il range di quel valore (quello dell'if) viene mappato, calcolando l'intensità che deve avere ogni led ad una determinata fase.

DigitalRGB

```

/* Includere la libreria del Buzzer */
#include <RGB.h>
/* Includere la libreria del potenziometro */
#include <potentiometer.h>

/* Includo la libreria del potenziometro */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo rgb */
RGB rgb(0,1,4);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /* Prendo lo stato del potenziometro */
    int stato = pot.getValue();

    /*Setto il valore del potenziometro al metodo digitalRGB della libreria RGB*/
    rgb.digitalRGB(stato);
}

```

Si occupa di ricavare lo stato del potenziometro e poi di settare questo valore al metodo digitalRGB()

```

void RGB::digitalRGB(int val)
{
    if(val > 850){
        digitalWrite(_pinR, LOW);
        digitalWrite(_pinG, HIGH);
        digitalWrite(_pinB, HIGH);
    }else if(val > 680){
        digitalWrite(_pinR, LOW);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, HIGH);
    }else if(val > 510){
        digitalWrite(_pinR, HIGH);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, HIGH);
    }else if(val > 340){
        digitalWrite(_pinR, HIGH);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, LOW);
    }else if(val > 170){
        digitalWrite(_pinR, HIGH);
        digitalWrite(_pinG, HIGH);
        digitalWrite(_pinB, LOW);
    }else{
        digitalWrite(_pinR, LOW);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, LOW);
    }
}

```

In base al valore del potenziometro val (stato del potenziometro passato come parametro), vengono accesi e spenti i led.

ResetterRGB

```

/* Includere la libreria del Buzzer */
#include <RGB.h>
/* Includere la libreria del potenziometro */
#include <potentiometer.h>

/* Includo la libreria del potenziometro */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo rgb */
RGB rgb(0,1,4);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /* Prendo lo stato del potenziometro */
    int stato = pot.getValue();

    /*Setto il valore del potenziometro al metodo resetterRGB della libreria RGB*/
    rgb.resetterRGB(stato);
}

```

Si occupa di ricavare lo stato del potenziometro e poi di settare questo valore al metodo 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);
    }
}

```

Quando il potenziometro ha raggiunto almeno una volta l'intensità massima, e successivamente quella minima, viene incrementato il valore reset, che col modulo 4 definirà in che fase di colore si trova.

Diario di lavoro

Luogo	SAM Trevano
Data	14.11.2018

Lavori svolti Abbiamo scelto le coppie e il progetto. Ci è stato consegnato il QDC, il quale abbiamo letto e abbiamo pensato alle domande riguardanti il progetto stesso.

Problemi riscontrati e soluzioni adottate Nemanja Stojanovic: Nessun problema
Thor Dublin: Nessun problema

Punto della situazione rispetto alla pianificazione Stiamo seguendo il programma del modulo

Programma di massima per la prossima giornata di lavoro Raccogliere le informazioni riguardanti il progetto e iniziare con la pianificazione.

Diario di lavoro

Luogo	SAM Trevano
Data	16.11.2018

Lavori svolti

Oggi abbiamo fatto il secondo test del modulo 306 nelle prime 2 ore, nel tempo rimanente abbiamo posto al professor Mussi le domande sul secondo progetto, tempo in cui ci sono state spiegate varie cose sullo scopo del progetto.

Abbiamo scelto il primo modulo dell'Arduino, siccome il mio compagno era assente, in base alle mie preferenze ho scelto il modulo che utilizzerà il potenziometro e il buzzer.

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Nessun problema

Thor Dublin:

Nessun problema

Punto della situazione rispetto alla pianificazione

Stiamo seguendo il programma del modulo

Programma di massima per la prossima giornata di lavoro

Discutere col mio compagno che oggi era assente delle domande che abbiamo posto al professore riguardo il nostro secondo progetto.

Diario di lavoro

Luogo	Lugano
Data	21 novembre 2018

Lavori svolti

Oggi io ed il mio compagno ci siamo suddivisi i seguenti compiti:

- Lavorare sulla pianificazione del progetto creando un diagramma gant.
- Iniziare la documentazione del progetto, ed informarsi sulla creazione delle librerie

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

Diario di lavoro

Luogo	SAM Trevano
Data	23.11.2018

Lavori svolti
Nemanja Stojanovic: Le prime due ore ha fatto il test. Poi si è occupato di finire il gantt e di iniziare l'analisi dei requisiti.
Thor Dublin: Ha portato avanti la documentazione.

Problemi riscontrati e soluzioni adottate
Nemanja Stojanovic: Nessun problema
Thor Dublin: Nessun problema

Punto della situazione rispetto alla pianificazione
Stiamo seguendo il piano

Programma di massima per la prossima giornata di lavoro
Finire l'analisi dei requisiti e informarsi su come fare il primo modulo

Diario di lavoro

Luogo	SAM Trevano
Data	23.11.2018

Lavori svolti

Nemanja Stojanovic:

Ha portato a termine l'analisi dei requisiti e ha iniziato a fare la ricerca sul funzionamento del buzzer.

Thor Dublin:

Ha cercato di capire come bisogna fare per creare le librerie.

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Nessun problema

Thor Dublin:

Nessun problema

Punto della situazione rispetto alla pianificazione

Siamo un po' indietro con l'implementazione.

Programma di massima per la prossima giornata di lavoro

Iniziare a creare lo schema logico e l'implementazione

Diario di lavoro

Luogo	SAM Trevano
Data	30.11.2018

Lavori svolti

Nemanja Stojanovic:

Inizialmente ha finito l'analisi dei requisiti. Poi si è occupato di cercare su internet i driver da installare per l'arduino digispark e poi ha fatto la presentazione del primo progetto. Infine ha saldato i pin femmina sul digispark.

Thor Dublin:

Inizialmente si è concentrato sul come creare le librerie, poi è passato a fare una bozza del primo modulo potenziometro con buzzer, utilizzando l'arduino ricevuto l'anno scorso al modulo 223.

Successivamente si è tentato capire come utilizzare il digispark e ha saldato i connettori femmina al digispark.

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Nessun problema

Thor Dublin:

Non riesce a richiamare i metodi delle librerie in un programma di prova.

Punto della situazione rispetto alla pianificazione

Siamo un po' indietro con l'implementazione.

Programma di massima per la prossima giornata di lavoro

Provare il digispark e assicurarsi che funzioni.

Diario di lavoro

Luogo	SAM Trevano
Data	05.12.2018

Lavori svolti

Nemanja Stojanovic:

Siccome il digispark non funzionava si è occupato di capire il perché. Inizialmente ha cancellato i driver e ha seguito la guida per reinstallarli e implementarli nel programma di Arduino, solo che il problema si presentava di nuovo.

Thor Düblin:

Oggi si è concentrato sul funzionamento del digispark in un primo momento, in un secondo momento ha iniziato a implementare il primo modulo del progetto.

Problemi riscontrati e soluzioni adottate

Il digispark non viene riconosciuto dalle macchine se lo collegiamo ad altri componenti tipo buzzer, probabilmente per colpa di un problema di alimentazione.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Far funzionare il digispark e pensare a tutte le funzioni della libreria per il primo modulo, documentando la guida

Diario di lavoro

Luogo	SAM Trevano
Data	07.12.2018

Lavori svolti

Nemanja Stojanovic:

Inizialmente ha fatto il fritzing del circuito. Poi ha cercato di capire perché il circuito non funzionasse e il problema consiste nel collegare i componenti. Nel fritzing ha aggiunto una resistenza per il collegamento del Buzzer, ora bisogna implementarlo realmente.

Thor Düblin:

Inizialmente si è concentrato sul documentarsi sull'uso del Buzzer, a quel punto è partita un'analisi delle frequenze da cui si è stabilito il range di frequenze che andremo ad utilizzare, cioè quelle udibili dal suono, c'è ancora da analizzare il datasheet del buzzer hydz che sembra abbia problemi con delle frequenze.

Successivamente ha provato a implementare alcune prove e bozze sull'utilizzo congiunto del buzzer e del potenziometro, comprendendo così come utilizzare il circuito.

Problemi riscontrati e soluzioni adottate

Il circuito non funziona sul digispark. Il programma del led collegato al digispark inizialmente non funzionava perché abbiamo utilizzato una porta che non accettava la modalità digitalwrite. Abbiamo risolto cambiando la porta.

Analizzando il buzzer si sono riscontrate delle frequenze non accettabili (che facevano troppo rumore), questo è dovuto all'uso di frequenze non supportate dal buzzer che utilizziamo.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Collegare correttamente i componenti e far funzionare il circuito sul digispark.

Analizzare le specifiche del buzzer e provare a implementare altre funzioni del modulo.

Diario di lavoro

Luogo	SAM Trevano
Data	14.12.2018

Lavori svolti

Nemanja Stojanovic:

Si è occupato di fare la libreria del Buzzer. Nella libreria del buzzer ha creato un metodo che semplicemente accende il buzzer (HIGH) e un metodo che lo spegne (LOW). Poi ha creato un metodo che calcola la frequenza che verrà applicata in base al valore del potenziometro.

Thor Dublin:

Si è occupato di fare la libreria del potenziometro. Nella libreria del potenziometro viene semplicemente ritornato il valore del potenziometro.

```
int sensorValue=0;
int sensorrr=0;

void setup()
{
    pinMode(1, OUTPUT);
}

void loop()
{
    sensorValue = analogRead(1); // Which corresponds to P2
    //tone(1, 1600/1024*sensorValue);
    if(sensorValue > 500 ){
        digitalWrite(1, HIGH);
    }else{
        digitalWrite(1, LOW);
    }
    //delay(sensorValue);
    /*digitalWrite(4, LOW);
    delay(sensorValue);*/
}
```

Siamo riusciti a far funzionare il buzzer e il potenziometro assieme. Più tardi abbiamo iniziato a fare le librerie, una del potenziometro e una del Buzzer. Più tardi la libreria del potenziometro verrà inclusa in quella del Buzzer.

Problemi riscontrati e soluzioni adottate

Quando vengono incluse le librerie in Arduino esso ci mostra l'errore seguente:

```
no matching function for call to 'Buzzer::Buzzer()'
exit status 1
no matching function for call to 'Buzzer::Buzzer()'
```

Su internet abbiamo trovato un sito che dice di creare un file "keywords.txt" con i valori seguenti:

```
Buzzer KEYWORD1
Buzzer KEYWORD2
setOnBuzzer KEYWORD2
setOffBuzzer KEYWORD2
frequence KEYWORD2
setOnBuzzerFrequence KEYWORD2
```

Ma neanche questo ha risolto il problema.

Punto della situazione rispetto alla pianificazione

Siamo indietro con l'implementazione

Programma di massima per la prossima giornata di lavoro

Mettere appunto il problema e creare la guida.

Diario di lavoro

Luogo	SAM Trevano
Data	19.12.2018

Lavori svolti

Nemanja Stojanovic

Si è occupato della libreria del buzzer, creando un metodo che calcola la frequenza inversa che verrà applicata in base al valore del ptenziometro

Thor Dueblin:

Si è occupato di portare avanti le librerie del bottone e del Led

Problemi riscontrati e soluzioni adottate

Nemanja: Quando ha realizzato il circuito e ha provato a caricare il codice sull'Arduino appare questo errore.

An error occurred while uploading the sketch
Sketch uses 930 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 11 bytes (0%) of dynamic memory, leaving 2037 bytes for local variables. Maximum is 2048 bytes.
An error occurred while uploading the sketch

Punto della situazione rispetto alla pianificazione

Siamo indietro con l'implementazione

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione.

Diario di lavoro

Luogo	SAM Trevano
Data	21.12.2018

Lavori svolti

Abbiamo fatto la teoria su come fare una presentazione.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo indietro con l'implementazione

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione.

Diario di lavoro

Luogo	Lugano
Data	9 gennaio 2019

Lavori svolti

Thor:

In questa lezione mi sono concentrato nell'implementazione del nuovo modulo bottone led, completando la libreria del bottone, e iniziando ad inserire delle funzioni come `getValueLED` che restituisce il valore del led 0 o 1 (spento o acceso).

Nella libreria del bottone è stata aggiunta una sola funzione (oltre il costruttore di base), che ritorna lo stato del bottone.

Nemanja: assente

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Leggermente indietro rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Finire il secondo modulo/Iniziare quello nuovo

Diario di lavoro

Luogo	SAM Trevano
Data	11.01.2018

Lavori svolti

Nemanja Stojanovic:

Inizialmente si è occupato di mettere apposto lo schema del primo circuito poi ha creato lo schema del prossimo circuito. Una volta finito lo schema ha implementato lo schema nella realtà collegando i componenti alla breadboard.

Thor Dublin:

Si è occupato di portare avanti le librerie del bottone e del Led, ha creato funzioni per la libreria del Led, come il Blink ed ha iniziato a progettare quelle del bottone, di cui però sono limitate con il ritorno del suo stato (premuto o meno).

Problemi riscontrati e soluzioni adottate

Nemanja: Quando ha realizzato il circuito e ha provato a caricare il codice sull'Arduino appare questo errore.

```
An error occurred while uploading the sketch
Sketch uses 930 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 11 bytes (0%) of dynamic memory, leaving 2037 bytes for local variables. Maximum is 2048 bytes.
An error occurred while uploading the sketch
```

Punto della situazione rispetto alla pianificazione

Siamo indietro con l'implementazione

Programma di massima per la prossima giornata di lavoro

Finire il modulo corrente.

Diario di lavoro

Luogo	SAM Trevano
Data	16.01.2018

Lavori svolti

Nemanja Stojanovic:

Inizialmente con il prof. Sartori ha discusso del primo progetto e gli è stata consegnata la nota. Poi si è occupato di risolvere il problema della volta prima, quello che quando veniva caricato il codice su arduino si presentava il problema con il caricamento. Inizialmente ho controllato la sintassi del codice ma il problema non consisteva in esso. Poi ho chiesto al mio compagno di caricare lui lo stesso codice dal suo computer e ha funzionato. Infatti poi ho controllato e ho notato che non ero selezionato il dispositivo corretto (Arduino Mega).

Thor Dublin:

Si è concentrato a finire la libreria del LED, aggiungendo l'ulteriore metodo toggle, dove il led cambia stato ogni volta che il pulsante viene cliccato, oggetto di implementazione su cui io ed il mio compagno abbiamo lavorato per entrambe le ore scolastiche.

Con questo metodo possiamo dire praticamente concluso l'implementazione di questo modulo e per la prossima lezione vedremo di testarlo con digispark.

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Ha risolto quello della settimana prima con i passaggi descritti nel capitolo "Lavori svolti".

Punto della situazione rispetto alla pianificazione

Siamo indietro con l'implementazione

Programma di massima per la prossima giornata di lavoro

Finire il modulo corrente e iniziare quello nuovo.

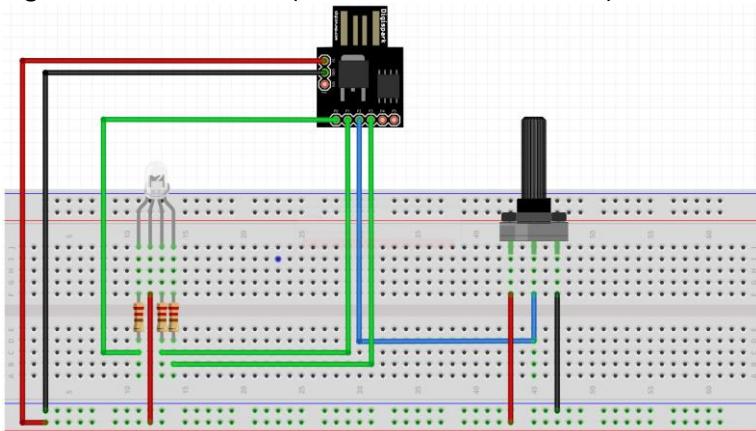
Diario di lavoro

Luogo	SAM Trevano
Data	18.01.2018

Lavori svolti

Nemanja Stojanovic:

Inizialmente si è occupato di fare i tre codici di esempio per il bottone e il led e poi ha creato lo schema logico del terzo modulo (Potenziometro e led RGB.)



Thor Düblin:

Ha iniziato la prima funzione del modulo potenziometro RGB, dove i colori si alternavano girando il potenziometro in modo digitale, cioè che i colori vengono semplicemente cambiati ad ogni fase del potenziometro, finito la prima funzione del modulo corrente.

Inizia la seconda funzione del modulo, che sarà simile alla prima ma utilizzando invece un metodo analogico, dove i colori cambieranno in modo fluido.

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Nessun problema

Thor Düblin:

I colori non cambiano in modo fluido ma continuano ad accendersi in modo digitale, pur utilizzando il comando analogWrite, ed avendo cambiato gli input dei led in pin analogici.

Punto della situazione rispetto alla pianificazione

Indietro con la creazione delle guide e con l'implementazione

Programma di massima per la prossima giornata di lavoro

Continuare il terzo modulo e fare le guide per tutti i moduli

Diario di lavoro

Luogo	SAM Trevano
Data	23.01.2018

Lavori svolti
<p>Nemanja Stojanovic: Si è occupato di creare gli esempi del modulo RGB potenziometro</p>
<p>Thor Dublin: In questa lezione ha concluso la seconda funzione del modulo potenziometro RGB, dove vengono cambiati in modo fluido i colori rosso, verde, blu per poi passare al bianco, risolvendo il problema che ha richiesto più tempo del previsto per essere risolto.</p>

Problemi riscontrati e soluzioni adottate
<p>Nemanja Stojanovic: Nessun problema</p>
<p>Thor Dublin: Risoluzione problema dell'accensione non analogica (fluida), per risolvere il problema è bastato cambiare gli input dei led che inizialmente erano analogici, in degli input digitali, in questo modo si è riuscito a cambiare i colori in modo analogico.</p>

Punto della situazione rispetto alla pianificazione
Indietro con la creazione delle guide e con l'implementazione

Programma di massima per la prossima giornata di lavoro
Continuare il terzo modulo e fare le guide per tutti i moduli

Diario di lavoro

Luogo	SAM Trevano
Data	25.01.2018

Lavori svolti

Nemanja Stojanovic:

Inizialmente si è occupato di creare gli esempi di codice per il modulo che riguarda il potenziometro e il buzzer poi ha iniziato a fare la guida del secondo modulo e mettere apposto i requisiti

Thor Düblin:

Ha iniziato a pensare a come fare la terza funzione del modulo potenziometro RGB, dove si inizierà a cambiare l'intensità del bianco per poi venir settata all'intensità del rosso, e successivamente verde e blu. Inizia a testare un po di metodi per il reset del colore che avverrà quando l'intensità di uno raggiungerà prima il massimo e poi il minimo.

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Nessun problema

Thor Düblin:

Nessun problema

Punto della situazione rispetto alla pianificazione

Indietro con la creazione delle guide e con l'implementazione

Programma di massima per la prossima giornata di lavoro

Continuare il terzo modulo e fare le guide per tutti i moduli

Diario di lavoro

Luogo	SAM Trevano
Data	30.01.2019

Lavori svolti
<p>Nemanja Stojanovic: Ha iniziato a fare le guide</p>
<p>Thor Dublin: In questa lezione si è occupato di concludere l'ultima funzione del terzo modulo, che quando il valore dell'intensità tramite il potenziometro viene portato al massimo, e poi al minimo esso attiva il flag per cambiare il colore dove si andrà nuovamente a modificarne l'intensità, nel seguente ordine: bianco, rosso, verde, blu.</p>

Problemi riscontrati e soluzioni adottate
<p>Nemanja Stojanovic: Nessun problema</p>
<p>Thor Dublin: Nessun problema</p>

Punto della situazione rispetto alla pianificazione
Indietro con la documentazione e le guide

Programma di massima per la prossima giornata di lavoro
Continuare le guide e la documentazione

Diario di lavoro

Luogo	SAM Trevano
Data	01.02.2019

Lavori svolti
<p><u>Nemanja Stojanovic:</u> Si è occupato di fare le guide, argomentando il lavoro.</p>
<p><u>Thor Dublin:</u> Si è occupato di continuare la documentazione, in specifico, i capitoli di progettazione e implementazione.</p>

<u>Problemi riscontrati e soluzioni adottate</u>
<p><u>Nemanja Stojanovic:</u> Nessun problema</p>
<p><u>Thor Dublin:</u> Nessun problema</p>

Punto della situazione rispetto alla pianificazione
Indietro con la documentazione e le guide

Programma di massima per la prossima giornata di lavoro
Continuare le guide e la documentazione

Diario di lavoro

Luogo	SAM Trevano
Data	06.02.2019

Lavori svolti
Nemanja Stojanovic: Ha portato avanti le guida e ha messo apposto gli esempi
Thor Dublin: Ha portato avanti la documentazione.

Problemi riscontrati e soluzioni adottate
Nemanja Stojanovic: Nessun problema
Thor Dublin: Nessun problema

Punto della situazione rispetto alla pianificazione
Indietro con la documentazione e le guide

Programma di massima per la prossima giornata di lavoro
Continuare le guide e la documentazione

Diario di lavoro

Luogo	SAM Trevano
Data	08.02.2019

Lavori svolti

Nemanja Stojanovic:

Oggi si è occupato di terminare la guida, mettere apposto gli esempi e la fase di progettazione.
Infine si è occupato di fare la presentazione

Thor Düblin:

Si è occupato di concludere la documentazione

Problemi riscontrati e soluzioni adottate

Nemanja Stojanovic:

Nessun problema

Thor Düblin:

Nessun problema

Punto della situazione rispetto alla pianificazione

Fine progetto

Programma di massima per la prossima giornata di lavoro

Fine progetto