

Sistema didattico per Arduino

Titolo del progetto: Sistema didattico per Arduino
Alunno/a: Paolo Weishaupt, Carlo Pezzotti
Classe: Info I3AC
Anno scolastico: 2018/2019
Docente responsabile: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori

Indice

1	Introduzione.....	4
1.1	Informazioni sul progetto.....	4
1.2	Abstract.....	4
1.3	Scopo.....	4
	Analisi.....	4
1.4	Analisi del dominio	4
1.5	Analisi e specifica dei requisiti.....	4
1.6	Analisi dei costi	6
1.7	Pianificazione.....	7
1.8	Analisi dei mezzi	8
1.8.1	Software	8
1.8.2	Drivers.....	8
1.8.3	Hardware.....	8
1.8.4	Specifiche Hardware.....	9
2	Progettazione	10
2.1	Design dell'architettura del sistema.....	10
2.2	Design procedurale	10
3	Implementazione	13
3.1	Libreria per un Led RGB	13
3.1.1	LightLed.h.....	13
3.1.2	LightLed.cpp.....	13
3.2	Libreria per un led	14
3.2.1	Led.h	14
3.2.2	Led.cpp.....	14
3.3	Libreria per un pulsante.....	15
3.3.1	ButtonState.h.....	15
3.3.2	ButtonState.cpp	15
3.4	Libreria per un piezo buzzer	16
3.4.1	Buzzer.h	16
3.4.2	Buzzer.cpp	16
3.5	Libreria per un potenziometro.....	17
3.5.1	Potenziometro.h	17
3.5.2	Potenziometro.cpp.....	17
4	Test.....	18
4.1	Protocollo di test	18
4.2	Risultati test	20
4.3	Mancanze/limitazioni conosciute	21
5	Consuntivo	22
6	Conclusioni.....	23
6.1	Sviluppi futuri	23
6.2	Considerazioni personali.....	23
7	Bibliografia	23
	Sitografia	23
8	Allegati	23

Indice delle figure

Figura 1: Gantt preventivo	7
Figura 2: Scheda DigiSpark	9
Figura 3: Configurazione delle porte di DigiSpark	9
Figura 4: Schema elettrico	10
Figura 5: Circuito elettrico	10
Figura 6: Design libreria per un Led RGB	10
Figura 7: Design libreria per un led	11
Figura 8: Design libreria per un pulsante.....	11
Figura 9: Design libreria per un piezo buzzer	12
Figura 10: Design libreria per un potenziometro.....	12
Figura 11: Gantt consuntivo	22

1 Introduzione

1.1 Informazioni sul progetto

Allievi coinvolti: Paolo Weishaupt, Carlo Pezzotti

Classe: Informatica 3AC presso la Scuola di Arti e Mestieri a Trevano

Docenti responsabili: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori

Data inizio: 14 / 11 / 2018

Data fine: 25 / 01 / 2019

1.2 Abstract

In this project you will find some very useful libraries to begin you training with the Arduino platform and a very well made manual on how to use them. At the end of the manual you will be able to use the very basics components of the Arduino and you will know the basics of it's language.

1.3 Scopo

Lo scopo del progetto è quello di fornire agli allievi del terzo anno delle scuole medie che partecipano al Promtec delle librerie e degli esempi di codice per poter usare un Arduino USB (mini DigiSpark) e un manuale che spieghi al meglio come usarle e montare i circuiti.

Analisi

1.4 Analisi del dominio

Il cliente vuole una libreria che permetta di chiamare delle funzioni che facilitino la stesura del codice. L'idea sarebbe quella di semplificare il più possibile il codice che dovrà utilizzare un terzo utente. L'utente finale saranno dei ragazzi di terza media, quindi con competenze informatiche basse o addirittura nulle.

Le librerie dovranno utilizzare tutti gli attuatori utilizzabili sul DigiSpark. Il mio team ha inizialmente il compito di sviluppare una libreria su un led RGB e tre pulsanti.

1.5 Analisi e specifica dei requisiti

ID	REQ-001
Nome	Digispark
Priorità	1
Versione	1.0
Nota	Digispark è un componente elettronico che serve per integrare programmazione e elettronica.
Sub-ID	Requisito
001	Digispark deve funzionare completamente.

ID	REQ-002
Nome	Attuatori
Priorità	1
Versione	1.0
Nota	Per far implementare dell'elettronica con la programmazione bisogna avere dei componenti elettronici chiamati attuatori.
Sub-ID	Requisito
001	Attuatori deve funzionare completamente.

ID	REQ-003
Nome	Ambiente di programmazione Arduino
Priorità	1
Versione	1.0
Nota	Per programmare si può utilizzare un qualsiasi editore di testo, per caricare il codice sulla scheda bisogna però utilizzare l'IDE (Integrated Development Environment) di Arduino.
Sub-ID	Requisito
001	Computer funzionante.

ID	REQ-004
Nome	Funzione che legge stato pulsante
Priorità	2
Versione	1.0
Nota	Per realizzare del codice semplice bisogna scrivere una funzione che in base al pin passato come argomento riconosce e ritorna lo stato di un pulsante, se è premuto oppure no.
Sub-ID	Requisito
001	Pulsanti funzionanti collegati ad una BreadBoard o VeroBoard.

ID	REQ-005
Nome	Funzione che incrementa valore led
Priorità	2
Versione	1.0
Nota	Ci dovrà essere una funzione nascosta all'utente, o che comunque non potrà utilizzare, che incrementerà la potenza di uscita dei pin collegati al Led RGB. Questi valori dovranno essere 3: R, G, B e dovranno avere un massimo di potenza 255 e un minimo di 0.
Sub-ID	Requisito
001	Funzione lettura pulsante.

ID	REQ-006
Nome	Funzione che scrive il valore
Priorità	2
Versione	1.0
Nota	Per realizzare del codice semplice bisogna scrivere una funzione che in base al pin passato come argomento scrive lo stato che può essere analogico oppure digitale. Nel nostro caso dove bisogna utilizzare un Led RGB dovremmo utilizzare delle uscite analogiche per rendere migliore l'esperienza.
Sub-ID	Requisito
001	Led RGB funzionante collegato ad una BreadBoard o VeroBoard.

1.6 Analisi dei costi

Nome	Salario	Ore di lavoro	Costo
Paolo Weishaupt	62 CHF/h	54	3348
Carlo Pezzotti	62 CHF/h	54	3348

In base a quanto calcolato sopra, con un salario di 62 CHF all'ora ognuno di noi dovrebbe costare 3348 CHF all'azienda.

1.7 Pianificazione

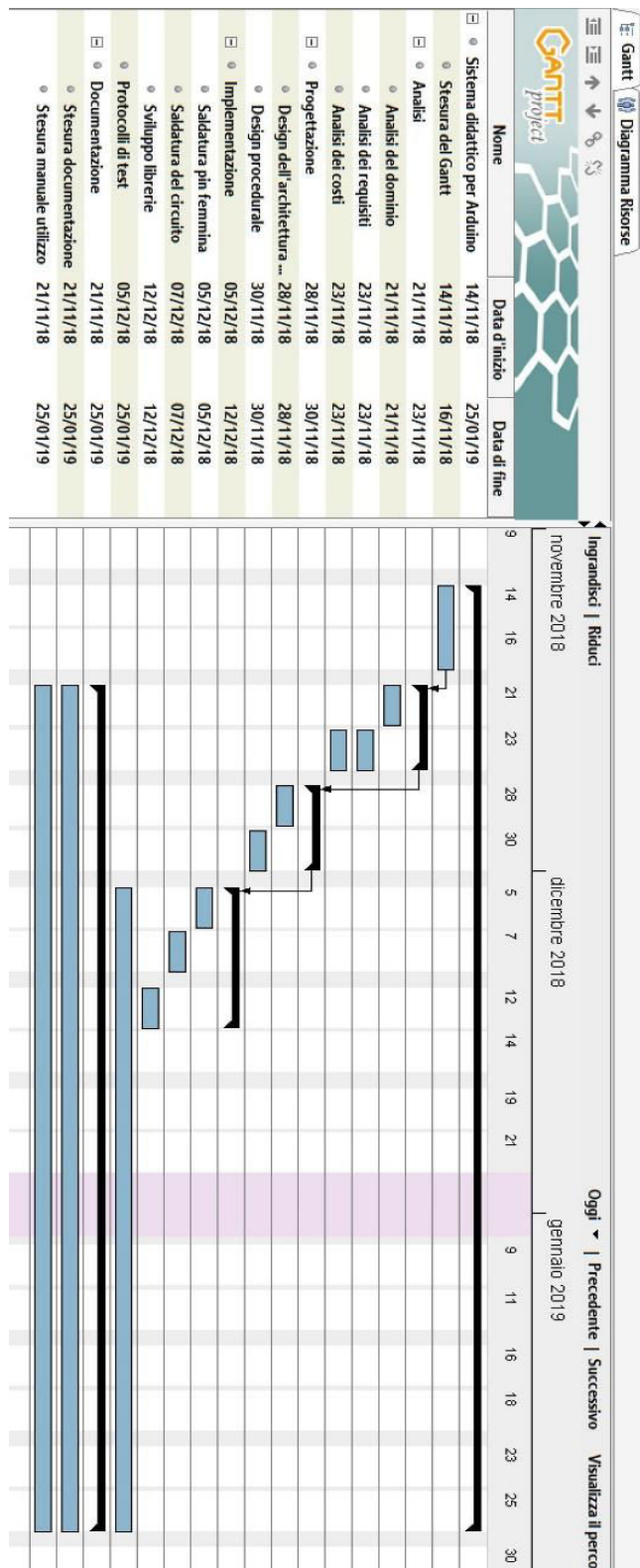


Figura 1: Gantt preventivo

1.8 Analisi dei mezzi

1.8.1 Software

Nome Software	Versione
Arduino IDE	1.8.7
Fritzing	0.9.3b
Atom	1.33.0
GanttProject	2.8.9 B2335
Microsoft Office Professional Plus	2016 16.0.4266.1001
GitHub Desktop	1.5.1
OneDrive (solo Paolo)	18.222.1104.0007

1.8.2 Drivers

Nome driver	Versione
Digistump LLC (digistump.com)	09/02/2014 5.1.2600.1
Digistump LLC	08/16/2014 1.1.0.0
Libusb-win32 Digispark Bootloader	01/17/2012 1.2.6.0
Libusb-win32 DigiUSB	09/02/2014 1.2.6.1

1.8.3 Hardware

Il progetto verrà sviluppato su una scheda DigiSpark facilmente reperibile online. La documentazione che abbiamo utilizzato sarà reperibile cliccando [qui](#). La scheda ha un problema, ossia il suo limitato numero di porte che ci costringe all'utilizzo di pochi attuatori. Carlo userà un HP Omen del 2016 con Windows 10 Pro mentre io utilizzerò un Huawei Matebook X Pro del 2018 con Windows 10 Home.

1.8.4 Specifiche Hardware

HP Omen 2016

Processore	Intel Core i7-6700HQ
Scheda grafica	NVIDIA GeForce GTX 965M
Ram	16GB
Display	15.6" 1920 x 1080px
Storage	128GB SSD + 1TB HDD

Huawei MateBook X Pro

Processore	Intel Core i7-8550U
Scheda grafica	NVIDIA GeForce GTX 150M
Ram	8 GB
Display	13.9" 3000x 2000px
Storage	512GB SSD

DigiSpark

- Supporto per Arduino IDE 1.0+ (OSX / Win / Linux)
- Alimentazione tramite USB o sorgente esterna - 5v o 7-35v (consigliato 12v o meno, selezione automatica)
- Regolatore con uscita 5V - 500mA
- Integrato USB incorporato
- 6 pin I / O (2 usano solo USB se il programma comunica attivamente tramite USB, altrimenti è possibile utilizzare tutti i 6 pin anche se si sta programmando tramite USB)
- 8k Memoria Flash (circa 6k dopo il bootloader)
- I2C e SPI
- PWM su 3 piedini (altri con Software PWM)
- ingresso ADC su 4 pin
- LED di alimentazione e LED di prova/stato

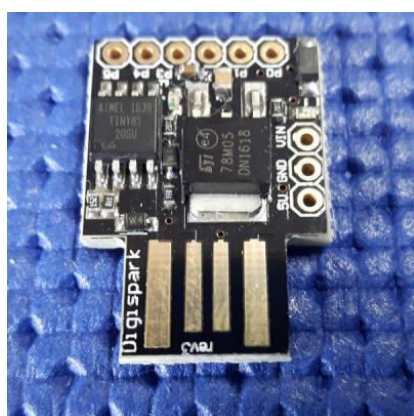


Figura 2: Scheda DigiSpark

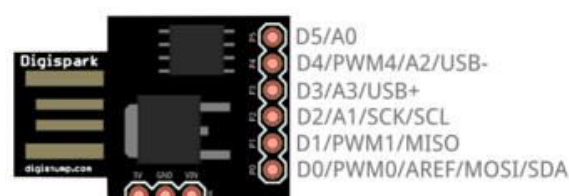


Figura 3: Configurazione delle porte di DigiSpark

2 Progettazione

2.1 Design dell'architettura del sistema

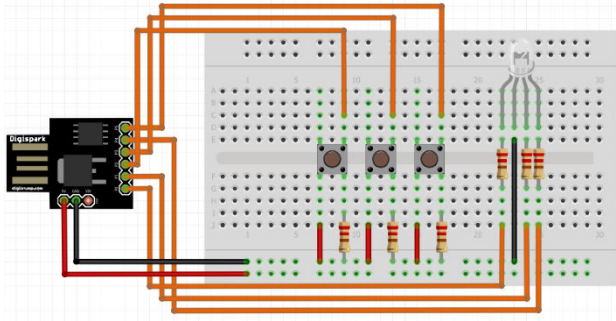


Figura 4: Schema elettrico

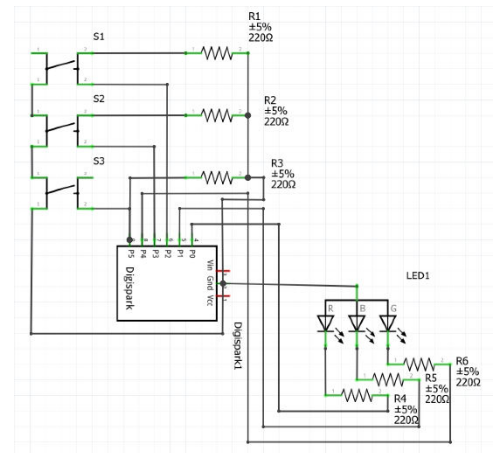


Figura 5: Circuito elettrico

2.2 Design procedurale

In tutte le librerie ci sarà un import della libreria Arduino.h per importare i metodi di base dell'Arduino.

Libreria per un Led RGB:

Questa libreria servirà per poter accendere un Led RGB

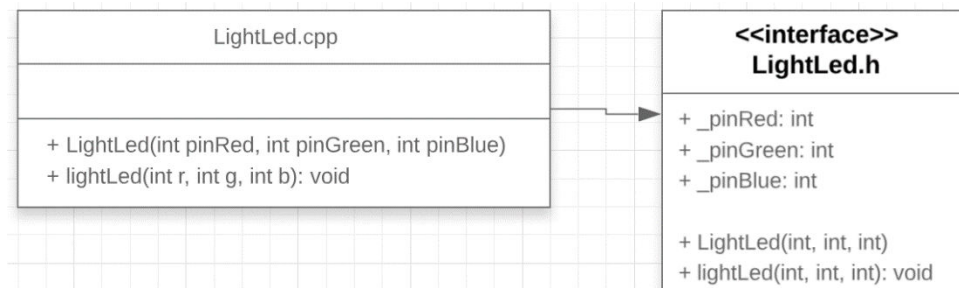


Figura 6: Design libreria per un Led RGB

Classi:

- LightLed.cpp
- LightLed.h

Attributi:

- int _pinRed: pin collegato al colore rosso
- int _pinGreen: pin collegato al colore verde
- int _pinBlue: pin collegato al colore blu

Metodi:

- LightLed (int pinRed, int pinGreen, int pinBlue): costruttore che dati i tre pin del Led RGB istanzia un oggetto di tipo LightLed
- void lightLed (int r, int g, int b): metodo che accende i led del Led RGB tutti assieme in base al valore passato come parametro

Libreria per un Led:

Questa libreria serve per accendere e spegnere un normale Led.

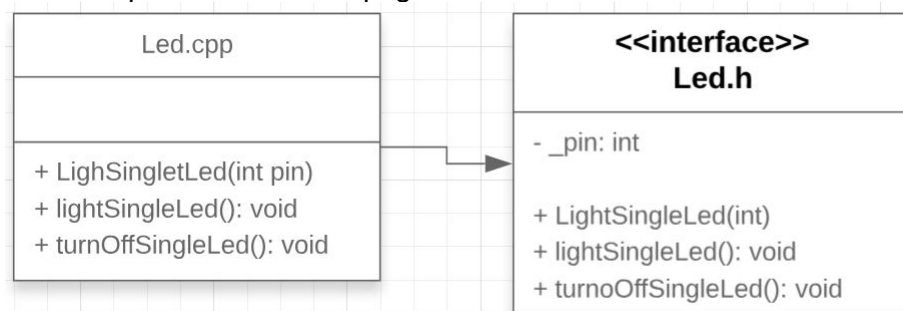


Figura 7: Design libreria per un led

Classi:

- Led.cpp
- Led.h

Attributi:

- int _pin: pin del Led

Metodi:

- LightSingleLed (int pin): costruttore che dato il pin del Led istanzia un oggetto di tipo LightSingleLed
- void lightSingleLed (): metodo che accende il Led
- void turnOffSingleLed (): metodo che spegne il Led

Libreria per un Pulsante:

Questa libreria serve per leggere il valore di un pulsante, ossia se è premuto o no.

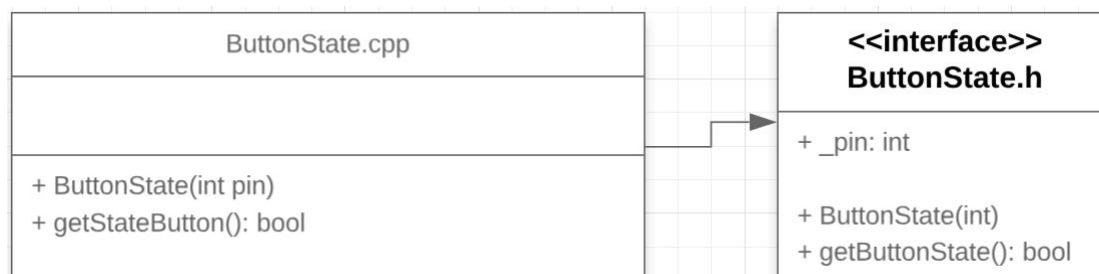


Figura 8: Design libreria per un pulsante

Classi:

- ButtonState.cpp
- ButtonState.h

Attributi:

- int _pin: pin del pulsante

Metodi:

- ButtonState (int pin): costruttore che dato il pin del pulsante istanzia un oggetto di tipo Button
- bool getStateButton (): metodo che ritorna lo stato del pulsante. Se è premuto ritorna 1 altrimenti ritorna 0.

Libreria per un Piezo Buzzer:

Questa libreria serve per far suonare un piezo buzzer.

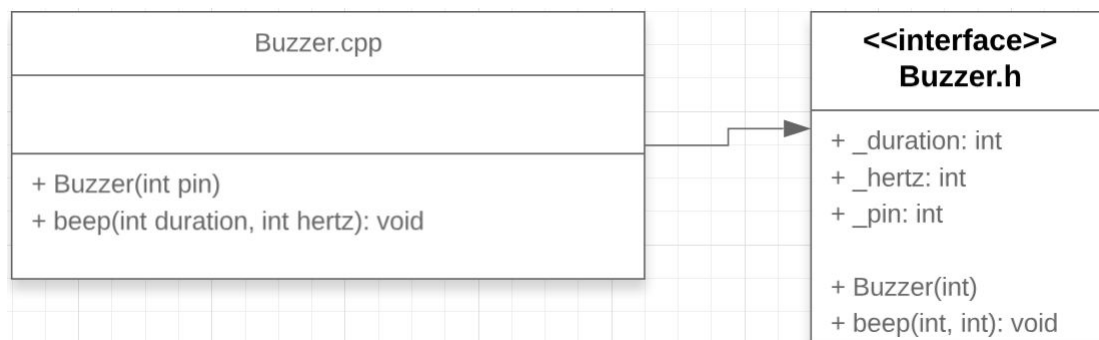


Figura 9: Design libreria per un piezo buzzer

Classi:

- Buzzer.cpp
- Buzzer.h

Attributi:

- int _duration: durata del suono da emettere
- int _hertz: frequenza del suono da emettere
- int _pin: pin del buzzer

Metodi:

- Buzzer (int pin): costruttore che dato il pin del buzzer istanzia un oggetto di tipo Buzzer
- void beep (int duration, int hertz): metodo che data una certa frequenza e una durata, fa suonare il piezo buzzer.

Libreria per un Potenzimetro:

Questa libreria serve per leggere il valore di un potenziometro.

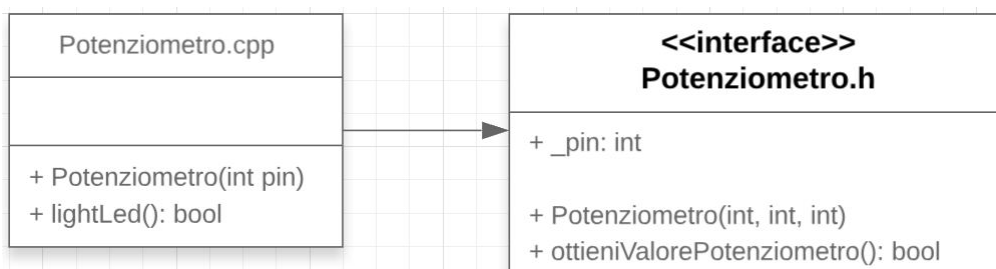


Figura 10: Design libreria per un potenziometro

Classi:

- Potenziometro.cpp
- Potenziometro.h

Attributo:

- int _pin: pin del potenziometro

Metodi:

- Potenziometro (int pin): costruttore che dato il pin del buzzer istanzia un oggetto di tipo Potenziometro
- void ottieniValorePotenziometro(): Metodo che ritorna il valore del potenziometro

3 Implementazione

3.1 Libreria per un Led RGB

3.1.1 LightLed.h

Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

```
8 #ifndef LightLed_h
9 #define LightLed_h
```

Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

```
11 // Library interface description
12 class LightLed
13 {
14     public:
15         int _pinRed;
16         int _pinGreen;
17         int _pinBlue;
18         LightLed(int, int, int);
19         void lightLed(int, int, int);
20 };
```

3.1.2 LightLed.cpp

Grazie a queste due righe di codice includo l'header che definisce gli attributi e i metodi da utilizzare e importo il file Arduino.h che definisce i metodi di base di Arduino.

```
8 #include "LightLed.h"
9 #import <Arduino.h>
```

Il costruttore ci permette di definire i tre pin passati come argomento in output e di assegnarne il loro valore agli attributi corrispondenti.

```
18 LightLed::LightLed(int pinRed, int pinGreen, int pinBlue)
19 {
20     pinMode(pinRed, OUTPUT);
21     pinMode(pinGreen, OUTPUT);
22     pinMode(pinBlue, OUTPUT);
23     _pinRed = pinRed;
24     _pinGreen = pinGreen;
25     _pinBlue = pinBlue;
26 }
```

Il metodo lightLed riceve come parametri i valori da settare ai pin in output per accendere il Led RGB. Al metodo AnalogWrite() bisogna passare il pin da accendere e il suo valore.

```
35 void LightLed::lightLed(int r, int g, int b){
36     analogWrite(_pinGreen, g);
37     analogWrite(_pinRed, r);
38     analogWrite(_pinBlue, b);
39 }
```

3.2 Libreria per un led

3.2.1 Led.h

Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

```
8 #ifndef LightSingleLed_h
9 #define LightSingleLed_h
```

Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

```
14 // Library interface description
15 class LightSingleLed
16 {
17     // User-accessible "public" interface
18     public:
19         LightSingleLed(int);
20         void lightSingleLed();
21         void turnOffSingleLed();
22
23     // Library-accessible "private" interface
24     private:
25         int _pin;
26 };
```

3.2.2 Led.cpp

Grazie a queste due righe di codice includo l'header che definisce gli attributi e i metodi da utilizzare e importo il file Arduino.h che definisce i metodi di base di Arduino.

```
9 #include "Arduino.h"
10 // Include this Library's description file.
11 #include "LightSingleLed.h"
```

Il costruttore ci permette di settare il pin passato come argomento in output e di assegnarlo alla variabile _pin.

```
17 LightSingleLed::LightSingleLed(int pin)
18 {
19     pinMode(pin, OUTPUT);
20     _pin = pin;
21 }
```

Il metodo lightSingleLed() permette tramite un digitalWrite() di settare lo stato del Led su HIGH e quindi di accendersi.

```
26 void LightSingleLed::lightSingleLed(){
27     digitalWrite(_pin,HIGH);
28 }
```

Il metodo `turnOffSingleLed()` tramite un `digitalWrite()` setta lo stato del Led su LOW e quindi si spegne.

```
32 void LightSingleLed::turnOffSingleLed() {
33     digitalWrite(_pin, LOW);
34 }
```

3.3 Libreria per un pulsante

3.3.1 ButtonState.h

Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

```
9  #ifndef ButtonState_h
10 #define ButtonState_h
```

Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

```
13 class ButtonState
14 {
15     // User-accessible "public" interface
16     public:
17         int _pin;
18         ButtonState(int);
19         bool getStateButton();
20 };
```

3.3.2 ButtonState.cpp

Grazie a queste due righe di codice includo l'header che definisce gli attributi e i metodi da utilizzare e importo il file `Arduino.h` che definisce i metodi di base di Arduino.

```
10 #include "ButtonState.h"
11 #import <Arduino.h>
```

Il costruttore permette di settare il pin passato come parametro in input e lo assegna alla variabile `_pin`.

```
17 ButtonState::ButtonState(int pin)
18 {
19     pinMode(pin, INPUT);
20     _pin = pin;
21 }
```

Il metodo `getStateButton` consente di vedere lo stato del pulsante (se è premuto o no) grazie a un `digitalRead()`.

```
27 bool ButtonState::getStateButton(){
28     return digitalRead(_pin);
29 }
```

3.4 Libreria per un piezo buzzer

3.4.1 Buzzer.h

Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

```
8 #ifndef Buzzer_h
9 #define Buzzer_h
```

Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

```
12 class Buzzer
13 {
14     // User-accessible "public" interface
15 public:
16     int _duration;
17     int _hertz;
18     int _pin;
19     Buzzer(int);
20     void beep(int, int);
21 };
```

3.4.2 Buzzer.cpp

Grazie a queste due righe di codice includo l'header che definisce gli attributi e i metodi da utilizzare e importo il file `Arduino.h` che definisce i metodi di base di Arduino.

```
8 #include "Buzzer.h"
9 #import <Arduino.h>
```

Il costruttore permette di settare il pin passato come parametro in output e lo assegna alla variabile `_pin`.

```
16 Buzzer::Buzzer(int pin)
17 {
18     _pin = pin;
19     pinMode(pin, OUTPUT);
20 }
```


Il metodo `beep` (`int duration`, `int hertz`) permette di far suonare il piezo buzzer per una durata e una frequenza in hertz passati come parametro.

```
27 void Buzzer::beep(int duration, int hertz) {
28     tone(_pin,hertz,duration);
29 }
```

3.5 Libreria per un potenziometro

3.5.1 Potenziometro.h

Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

```
8 #ifndef Potenziometro_h
9 #define Potenziometro_h
```

Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

```
12 class Potenziometro
13 {
14     // User-accessible "public" interface
15     public:
16         Potenziometro(int);
17         int ottieniValorePotenziometro();
18         int _pin;
19 }
```

3.5.2 Potenziometro.cpp

Grazie a queste due righe di codice includo l'header che definisce gli attributi e i metodi da utilizzare e importo il file `Arduino.h` che definisce i metodi di base di Arduino.

```
9 #include "Potenziometro.h"
10 #import <Arduino.h>
```

Il costruttore permette di settare il pin passato come parametro in input e di assegnarne il valore alla variabile `_pin`.

```
16 Potenziometro::Potenziometro(int pin)
17 {
18     pinMode(pin,INPUT);
19     _pin = pin;
20 }
```

il metodo `ottieniValorePotenziometro()` permette, tramite un `analogRead()` eseguito sul pin del potenziometro, di leggere il valore del potenziometro.

```
26  int Potenziometro::ottieniValorePotenziometro(){
27      return analogRead(_pin);
28  }
```

4 Test

4.1 Protocollo di test

Test Case:	TC-001	Nome:	Digispark
Riferimento:	REQ-001		
Descrizione:	Per il corretto funzionamento bisogna verificare che il digispark sia correttamente funzionante		
Prerequisiti:	Nulla.		
Procedura:	<ol style="list-style-type: none"> 1. Collegare il digispark al computer e verificare che il led rosso si accenda 2. Per verificare se tutti i pin funzionano correttamente bisogna fare un piccolo programma dove sequenzialmente si fanno accendere i led. 		
Risultati attesi:	Se tutto dovesse andare a buon fine allora mi aspetterai che il digispark e tutti pin funzionino correttamente.		

Test Case:	TC-002	Nome:	Attuatori
Riferimento:	REQ-002		
Descrizione:	Per il giusto funzionamento di un attuatore bisogna verificare la scheda tecnica di esso quindi come utilizzarlo.		
Prerequisiti:	Una scheda digispark funzionante vedi TC-001		
Procedura:	<ol style="list-style-type: none"> 1. Recarsi sulla scheda tecnica dell'attuatore verificato e visualizzare il giusto funzionamento (es. I poli) 2. Provare a fare un piccolo circuito o utilizzare uno già esistente e verificare che seguendo le istruzioni scritte sulla scheda tecnica l'attuatore svolga il lavoro desiderato. 		
Risultati attesi:	Se tutto dovesse andare a buon fine allora il risultato finale sarà uguale a quello aspettato (es. un led si accende se lo collego)		

Test Case:	TC-003	Nome:	Ambiente ide di arduino
Riferimento:	REQ-003		
Descrizione:	Per poter programmare con Arduino/digispark bisogna avere un ambiente di sviluppo e di carica dati. Arduino ci fornisce il suo.		
Prerequisiti:	Computer funzionante.		
Procedura:	<ol style="list-style-type: none"> 1. Scaricare dal sito ufficiale l'IDE di Arduino 2. Provare a lanciare il programma 3. Far partire un codice di esempio e verificare che il digispark faccia quello che deve fare 		
Risultati attesi:	Se tutto dovesse andare una volta che si esegue un codice allora digispark farà quello per cui è stato programmato (es. codice di prova blinking led -> il digispark farà lampeggiare il led di cui è fornito)		

Test Case:	TC-004	Nome:	Funzione che legge stato pulsante
Riferimento:	REQ-004		
Descrizione:	Se il programma e il circuito fornito dovessero funzionare allora con l'avvio del programma digispark sarà capace di leggere lo stato di un pulsante con una semplice funzione		
Prerequisiti:	Tutti i requisiti precedenti.		
Procedura:	<ol style="list-style-type: none"> 1. Istanziare un nuovo oggetto ButtonState (vedi manuale allegato) 2. Chiamare la funzione e verificare che l'Arduino legga ciò che deve leggere 		
Risultati attesi:	Se il programma è stato scritto in modo giusto allora l'Arduino svolgerà l'operazione per il quale è stato programmato.		

Test Case:	TC-005	Nome:	Funzione che incrementa stato del led
Riferimento:	REQ-005		
Descrizione:	In base alla nostra logica, quando un pulsante viene premuto il led RGB incrementa il colore (in base al pulsante).		
Prerequisiti:	Funzione di lettura pulsante funzionante.		
Procedura:	<ol style="list-style-type: none"> 1. Istanziare un nuovo oggetto ButtonState (vedi manuale allegato) 2. Quando il pulsante viene premuto incrementare il valore del colore rosso/verde/o blu 		
Risultati attesi:	Se si preme il pulsante si incrementa il colore.		

Test Case:	TC-006	Nome:	Funzione che scrive il valore desiderato
Riferimento:	REQ-006		
Descrizione:	Una volta che si ha il colore bisogna solo scrivere il colore sul led RGB, ciò si fare grazie alla libreria da noi ideata e scritta (vedi manuale)		
Prerequisiti:	Tutti i requisiti precedenti		
Procedura:	1. Utilizzare la funzione da noi ideata e scritta per scrivere i dati sul led RGB, (vedi manuale)		
Risultati attesi:	Quando si preme un pulsante il valore viene incrementato e scritto.		

4.2 Risultati test

Risultato TC-001:

Nel nostro caso la scheda *Digispark* ha funzionato al primo tentativo senza riscontrare errori.

Risultato TC-002:

Il primo attuatore che abbiamo provato è stato un diodo led normalissimo rosso, e abbiamo riscontrato un problema. Abbiamo verificato che la polarità fosse quella corretta e lo era, abbiamo verificato che la corrente fornita fosse abbastanza per il funzionamento del led e lo era. Quindi siamo giunti alla conclusione che il problema fosse l'attuatore, l'unica soluzione è stata quella di cambiare led. Con il secondo ha funzionato

Risultato TC-003:

Abbiamo installato correttamente l'ide dal sito ufficiale di arduino e abbiamo seguito la guida fornitaci dal docente Adriano Barchi per la corretta installazione dei driver e delle librerie di Digispark. Sul computer HP Omen non ci sono stati errori di caricamento del codice quindi il codice di prova che ho caricato ha funzionato correttamente, su Huawei MateBook X Pro la scheda *Digispark* non veniva riconosciuta pur essendoci i driver. La soluzione adottata è stata: prima di inserire il *Digispark* nella porta USB bisognava compilare il programma e caricarlo.

Risultato TC-004:

Per verificare il giusto funzionamento della libreria abbiamo attaccato al *Digispark* un led e un pulsante, utilizzando la libreria per la lettura dello stato del pulsante, accendevamo il led se il pulsante era premuto o no.

Risultato TC-005:

Per verificare il giusto funzionamento abbiamo collegato al *Digispark* un pulsante e abbiamo fatto un piccolo programma che se il pulsante era premuto incrementava il valore di una variabile e poi andava a scriverla utilizzando la libreria per il led RGB sul led RGB. Siamo incappati in un errore, ovvero non il led RGB non accendeva il colore che ci saremmo aspettati, la soluzione è stata cambiare l'ordine dei pin perché era sbagliato.

Risultato TC-006:

Per verificare il giusto funzionamento abbiamo svolto gli stessi passaggi del test precedente, perché, a differenza di arduino, questa scheda non ha la possibilità di utilizzare il monitor seriale, quindi l'unico modo per verificare che un dato sia corretto è testarlo con componenti fisici reali (attuatori).

4.3 Mancanze/limitazioni conosciute

La scheda Digispark è stata progettata con pochi pin quindi la possibilità di attaccare attuatori e sensori è molto limitata. Inoltre alcuni sensori e attuatori, come il potenziometro, possono richiedere più volte di quelli che la scheda può fornire, ciò porta a malfunzionamenti o addirittura crash da parte della scheda. Alcuni pin non fanno il lavoro che dovrebbero fare.

5 Consuntivo

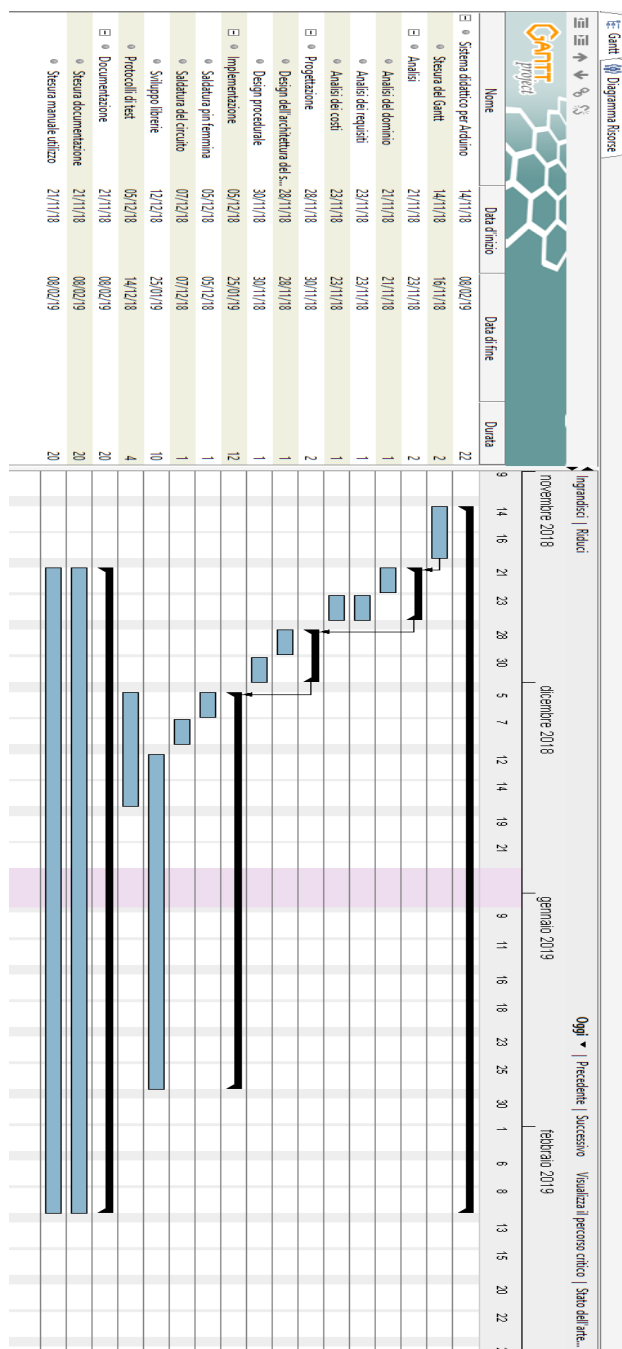


Figura 11: Gantt consuntivo

6 Conclusioni

Questo progetto potrà tornare molto utile per poter mostrare ai giovani che vogliono intraprendere un futuro nel mondo della programmazione quello che si può fare con un modulo Arduino e delle conoscenze di base di elettronica. Secondo noi la creazione di queste librerie può essere un'utile aggiunta nel bagaglio di esercizi che per esempio una scuola potrebbe utilizzare durante una sua giornata informativa per i futuri nuovi allievi.

6.1 Sviluppi futuri

In futuro si potrebbe estendere il numero di attuatori utilizzati, documentati e testati con dei codici ad-hoc.

6.2 Considerazioni personali

Abbiamo imparato a lavorare in team. Inoltre, abbiamo imparato a sviluppare librerie per schede Arduino USB (mini DigiSpark) in C e C++.

7 Bibliografia

Sitografia

1. <https://stackoverflow.com/>
2. <https://www.arduino.cc/en/Hacking/LibraryTutorial>





8 Allegati

- A. Quaderno dei Compiti
- B. Diari di lavoro
- C. Manuale di utilizzo
- D. Prodotto:
 - GitHub: <https://github.com/PaoloWeishaupt/Sistema-didattico-per-Arduino>

Allegato A

Titolo del progetto: Sistema didattico per Arduino
Alunno/a: Paolo Weishaupt, Carlo Pezzotti
Classe: Info I3AC
Anno scolastico: 2018/2019
Docente responsabile: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori

1 INFORMAZIONI GENERALI

Allievo/i	Nome:	Cognome:
		
Luogo di lavoro	Aula 417 (ex A-413) – Scuola d'Arti e Mestieri Trevano	
Orientamento	<input checked="" type="checkbox"/> 88602 Informatica aziendale	
Docente responsabile	Nome:	Cognome:
		
Periodo		
Orario di lavoro	Secondo orario scolastico 1° semestre	
Numero di ore lezione		
Pianificazione (in H o %)	Analisi: 10%	
	Implementazione: 40%	
	Test: 20%	
	Documentazione: 30%	

2 PROCEDURA

- L'allievo realizza il lavoro autonomamente o con il gruppo al quale è assegnato, sulla base del quaderno dei compiti ricevuto il 1° giorno.
- Il quaderno dei compiti è presentato, commentato e discusso con l'allievo. Con la sua firma, l'allievo accetta il lavoro proposto.
- L'allievo ha conoscenza della scheda di valutazione all'inizio del lavoro.
- L'allievo è responsabile dei suoi dati.
- In caso di problemi gravi, l'allievo avverte immediatamente il docente responsabile.
- L'allievo ha la possibilità di chiedere aiuto, ma deve menzionarlo nella documentazione.
- Alla fine del tempo a disposizione per la realizzazione del progetto, l'allievo deve inviare via e-mail il progetto al docente responsabile. In parallelo, una copia cartacea della documentazione dovrà essere fornita sempre al docente responsabile. Quest'ultima deve essere in tutto identica alla versione elettronica.

3 TITOLO

Sistema didattico per Arduino con libreria per attuatori e relativa documentazione

4 HARDWARE E SOFTWARE DISPONIBILE

1 PC della scuola + programmi concordati con i formatori

5 PREREQUISITI

Conoscenza Arduino

6 DESCRIZIONE DEL PROGETTO

Si tratta di sviluppare un nuovo prodotto didattico da utilizzare con gli allievi del terzo anno delle scuole medie che vengono a visitare la scuola durante le giornate di porte aperte Promtec.

Il prodotto da sviluppare è composto da un Arduino USB (mini DigiSpark), dei componenti elettronici da usare come attuatori, delle librerie/esempi di codice per l'utilizzo dei componenti e una guida che aiuti l'utente all'uso dell'Arduino e al montaggio del circuito.

In particolare:

- Verificare che tutto i componenti del progetto siano funzionanti.
- Creare per ogni combinazione di attuatori una libreria con del codice d'esempio.
- Per ogni combinazione di attuatori, creare almeno tre differenti esempi di codice.
- Tutto il codice deve essere ben commentato, in modo che gli allievi possano utilizzare/modificare il prodotto in maniera facile e veloce.
- Preparare una procedura di test, per controllare che tutti gli Arduino possano funzionare con tutte le combinazioni di attuatori che verranno proposti.
- Produrre una guida/scheda d'utilizzo, per l'installazione dell'Arduino e per ogni tipologia di attuatori, da consegnare insieme al prodotto durante il corso.
-

7 RISULTATI FINALI

L'allievo è responsabile della consegna al docente responsabile di:

- Una pianificazione iniziale (entro la prima settimana)
- Una documentazione della pianificazione e progettazione
- Una documentazione dell'implementazione e test
- Un diario di lavoro giornaliero entro le 18:00
- *Le librerie prodotte, gli schemi elettrici di collegamento ed i rispettivi Sketch di applicazione delle singole librerie con le rispettive spiegazioni*

8 PUNTI TECNICI SPECIFICI VALUTATI

La griglia di valutazione definisce i criteri generali secondo cui il lavoro dell'allievo sarà valutato (documentazione, diario, rispetto dei standard, qualità, ...).

Inoltre, il lavoro sarà valutato sui seguenti 7 punti specifici (punti da A14 a A20):

1. *159, Analisi del problema (programmazione)*
2. *124, Ipotesi di test, casi di test*
3. *185, Rilevamento di errori*
4. *228, Manuale utente*
5. *229, Valutazione*
6. *164, Codifica: Gestione degli errori*
7. *123, Commenti del codice sorgente*

9 FIRMA

Allievo

(luogo e data)

Docente responsabile

(luogo e data)

Allegato B

Titolo del progetto: Sistema didattico per Arduino
Alunno/a: Paolo Weishaupt, Carlo Pezzotti
Classe: Info I3AC
Anno scolastico: 2018/2019
Docente responsabile: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-14

Lavori svolti

Tempo: 2 ore

Ci è stato consegnato il nuovo QdC e ci è stato spiegato il nuovo progetto.

Abbiamo creato il repository su GitHub per il nuovo progetto e ci abbiamo caricato le cartelle di base.

Abbiamo stilato una lista delle domande da fare ai docenti la prossima lezione.

Vedere allegato.

Problemi riscontrati e soluzioni adottate

Abbiamo avuto un problema nel creare un repository pubblico che non permettesse a degli estranei di contribuire. Abbiamo risolto grazie al seguente sito:

<https://stackoverflow.com/questions/7920320/adding-a-collaborator-to-my-free-github-account>

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Esporre le nostre domande ai docenti e iniziare con la stesura del Gantt preventivo.

Domande da fare ai committenti

1. Cosa si intende per ogni combinazione di attuatori?
2. Nella documentazione bisogna aggiungere una guida su come usare l'Arduino?

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-16

Lavori svolti

Tempo: 2 ore

Abbiamo svolto il test.

Tempo: 2 ore

Abbiamo esposto le nostre domande al docente(Mussi). In allegato il resoconto riguardante l'incontro.

Abbiamo scelto quale componente usare. Noi useremo il Led RGB.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Progettare il circuito da consegnare a Manetti.

Resoconto

1. Per attuatori si intendono i singoli componenti del DigiSpark.
2. Ogni gruppo avrà inizialmente un componente attorno al quale sviluppare il progetto il più semplicemente possibile.
3. Se un gruppo finisce il suo componente prima riceverà il componente di un altro gruppo e dovrà sviluppare una libreria per quel componente.
4. Le librerie potranno essere sviluppate tutte in un singolo file oppure essere all'interno di diversi file.
5. Se si sviluppa in un solo file l'utente finale potrebbe capire più facilmente il funzionamento. Con file separati invece è più comodo per noi da sviluppare.
6. Bisognerà saldare dei contatti femmina al DigiSpark e progettare i circuiti che andremo ad utilizzare.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-16

Lavori svolti

Tempo: 1 ora

Abbiamo seguito una guida per poter utilizzare il DigiSpark sui nostri computer.

Tempo: 1 ora

Carlo ha finito i capitoli su analisi del dominio, dei costi e ha testato il funzionamento del suo DigiSpark.

Io ho eseguito la stesura del Gantt Preventivo e ho disegnato lo schema del nostro primo circuito con un Led RGB e tre bottoni.

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

Eseguire l'analisi dei requisiti.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-23

Lavori svolti

Tempo: 2 ore

Nella prima ora Carlo ha saldato i pin femmina al suo DigiSpark e ha fatto il capitolo sull'analisi dei requisiti.

Io ho fatto i capitoli 1.1, 1.2, 1.3, 1.7 e li ho aggiunti alla documentazione.

Tempo: 2 ore

Mi sono documentato su come creare delle librerie per Arduino e ho fatto la presentazione.

Carlo ha fatto un codice per testare il DigiSpark.

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

Seguire la pianificazione del Gantt.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-28

Lavori svolti

Tempo: 2 ore

Carlo ha implementato un codice di esempio per il Led RGB e i tre bottoni.
Io ho continuato con la documentazione.

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

Seguire la pianificazione del Gantt.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-30

Lavori svolti

Tempo: 4 ore

Abbiamo testato il funzionamento dei nostri DigiSpark e una volta finito gli abbiamo saldato i Pin femmina.

Carlo inoltre ha implementato gli ultimi due codici di esempio per l'attuatore assegnatoci.

Io ho iniziato con la stesura di una bozza iniziale della guida.

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

Seguire la pianificazione del Gantt.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12.05

Lavori svolti

Tempo: 2 ore

Ho provato a trasportare il circuito che avevamo svolto la scorsa volta sull'Arduino Mega nel DigiSpark.

Carlo era malato.

Problemi riscontrati e soluzioni adottate

Ho avuto dei problemi con la rilevazione del DigiSpark. Se lo si collegava senza nessun componente attaccato Windows lo riconosceva, ma appena si aggiungeva un componente Windows non lo riconosceva più. Per consiglio del docente Barchi ho provato a scollegare il Led RGB dal circuito e a vedere se lo riconosceva con solo i bottoni collegati, ma non ha funzionato.

Punto della situazione rispetto alla pianificazione

In ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Riuscire a collegare il DigiSpark con gli attuatori attaccati.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-07

Lavori svolti

Tempo: 4 ore

Carlo ha fixato i due codici implementati la settimana scorsa.

Ci siamo informati su come creare delle librerie per Arduino e su come utilizzarle su DigiSpark.

Io ho finito la stesura della libreria per il primo esempio di codice GetButtonState.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

In ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Riuscire a collegare il DigiSpark con gli attuatori attaccati.

Finire le librerie degli altri due codici di esempio per DigiSpark.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-12

Lavori svolti

Tempo: 2 ore

Abbiamo migliorato le tre librerie secondo gli standard.

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

Fare i codici di esempio.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-14

Lavori svolti

Tempo: 4 ore

Carlo ha implementato 3 codici di esempio per ogni libreria.

Io ho migliorato la documentazione e ho iniziato la stesura del manuale.

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

Fare i codici di esempio.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-19

Lavori svolti

Tempo: 2 ore

Carlo ha fatto gli esempi per la libreria del potenziometro.
Io ho aggiustato i commenti e alcuni pezzi di codice.

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

Continuare con la documentazione e il manuale.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-09

Lavori svolti

Tempo: 2 ore

Ci è stata riconsegnata la valutazione del primo progetto.

Abbiamo discusso riguardo all'utilizzo di nuovi attuatori, ma alla fine non ce ne sono stati assegnati altri.

Abbiamo continuato la documentazione e il manuale.

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

Continuare con la documentazione e il manuale.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-11

Lavori svolti

Tempo: 4 ore

Abbiamo continuato con la stesura del manuale e della documentazione.

Carlo si è occupato del manuale e ha finito una prima bozza.

Io ho continuato la documentazione con l'aggiunta degli schemi su BreadBoard e gli schemi elettrici.

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

Continuare con la documentazione e il manuale.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-18

Lavori svolti

Tempo: 4 ore

Carlo ha fatto i test case e ha aggiunto il capitolo 4.1

Io ho continuato la documentazione e ho corretto la grammatica del manuale. Ho aggiunto il capitolo 2.2 e ho modificato lo stile dei capitoli 1.7, 1.8.1 e 1.8.2. Ho aggiunto anche l'indice delle figure.

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

Continuare con la documentazione e il manuale.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-23

Lavori svolti

Tempo: 2 ore

Carlo ha fatto il capitolo sui test case.

Io ho fatto i diagrammi delle classi UML da aggiungere nel capitolo 2.2.

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

Continuare con la documentazione e il manuale.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-25

Lavori svolti

Tempo: 4 ore

Paolo ha aggiunto i diagrammi delle classi al capitolo 2 e ha fatto il capitolo 3.1 della documentazione con annessi sottocapitoli

Io ho finito tutta la guida d'uso per l'utilizzo delle librerie e la guida all'installazione dell'ambiente di sviluppo per Digispark.

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

Continuare con la documentazione

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-30

Lavori svolti

Tempo: 4 ore

Ci siamo portati avanti con la scrittura della documentazione. Abbiamo corretto i punti che il docente Barchi ci aveva segnalato.

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

Continuare con la documentazione

Diario di lavoro

Luogo	SAM Trevano
Data	2019-02-06

Lavori svolti

Tempo: 4 ore

Ho continuato la stesura della documentazione.

Carlo era assente poiché malato, ma da casa ha iniziato la realizzazione della presentazione.

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

Finire la documentazione e la presentazione.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-02-08

Lavori svolti

Tempo: 4 ore

Ho finito la documentazione. Ho sistemato delle parti nel Manuale.

Carlo ha finito la presentazione.

Ho unito i pdf dei diari alla documentazione, al QdC e al Manuale.

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

Ricevere il nuovo progetto

Allegato C

Titolo del progetto: Sistema didattico per Arduino
Alunno/a: Paolo Weishaupt, Carlo Pezzotti
Classe: Info I3AC
Anno scolastico: 2018/2019
Docente responsabile: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori

2019

Manuale d'utilizzo

INTRODUZIONE ALL'UTILIZZO DELLE LIBRERIE
PAOLO WEISHAAPT, CARLO PEZZOTTI

Indice

Indice	1
Manuale d'uso delle librerie.....	2
Premessa:	3
Installazione della scheda Digispark	3
Uso della scheda Digispark con l'IDE	7
Primo test	8
Led RGB:	9
Come utilizzare il Led RGB	9
Come utilizzare la libreria del Led RGB:	10
Push button:	11
Come utilizzare un pulsante:.....	11
Come utilizzare la libreria del pulsante:.....	11
Led:	12
Come utilizzare un Led:	12
Come utilizzare la libreria del Led:	12
Potenziometro:.....	13
Come utilizzare un potenziometro:.....	13
Come utilizzare la libreria del potenziometro:	13
Piezo Buzzer:.....	14
Come utilizzare un Piezo Buzzer:	14
Come utilizzare la libreria del Piezo Buzzer:	14
Sitografia.....	15

Manuale d'uso delle librerie

L'utilizzo di queste librerie è raccomandato a delle persone che si stanno avvicinando allo sviluppo sulla piattaforma Arduino e hanno bisogno di una guida nei primi passi della programmazione. Al loro interno ci sono anche tre codici di esempio di utilizzo della libreria per ogni attuatore elencato:

Guida alle librerie

1. Led RGB
2. Push Button
3. Led
4. Potenzziometro
5. Piezo buzzer

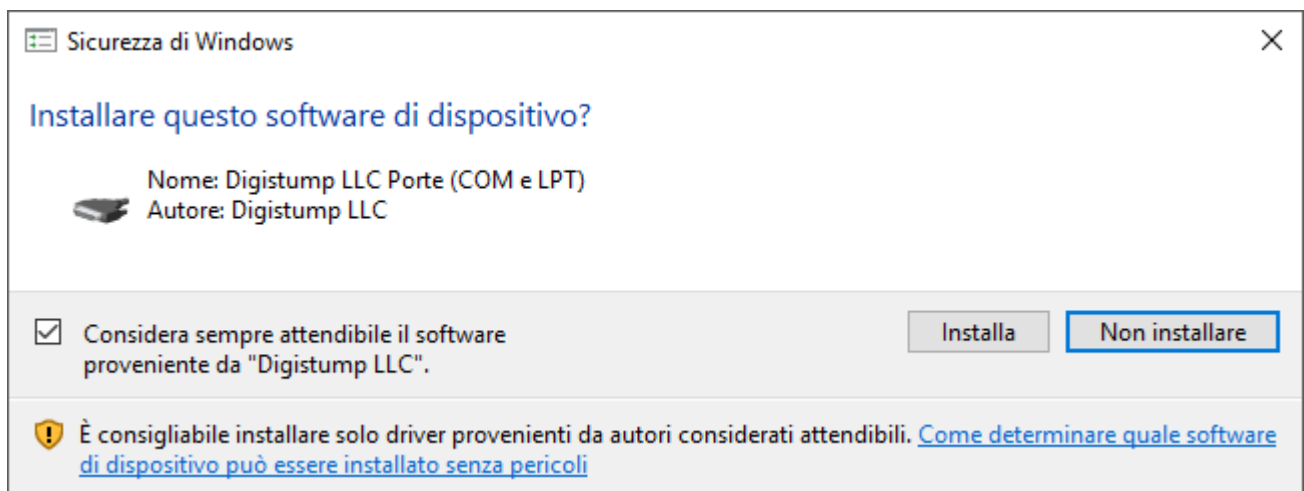
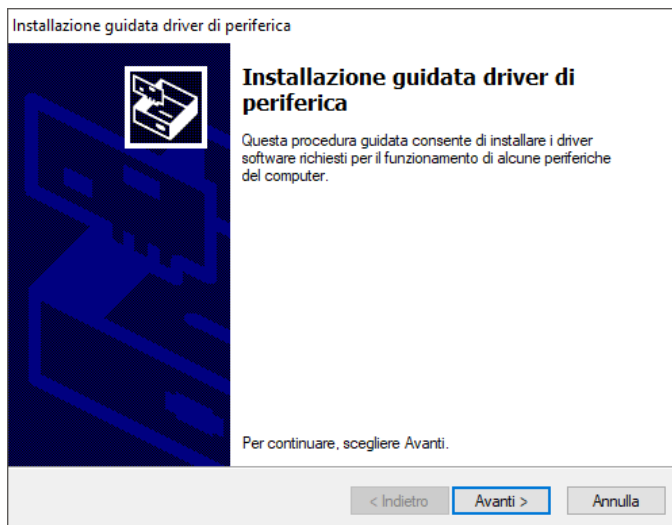
Premessa:

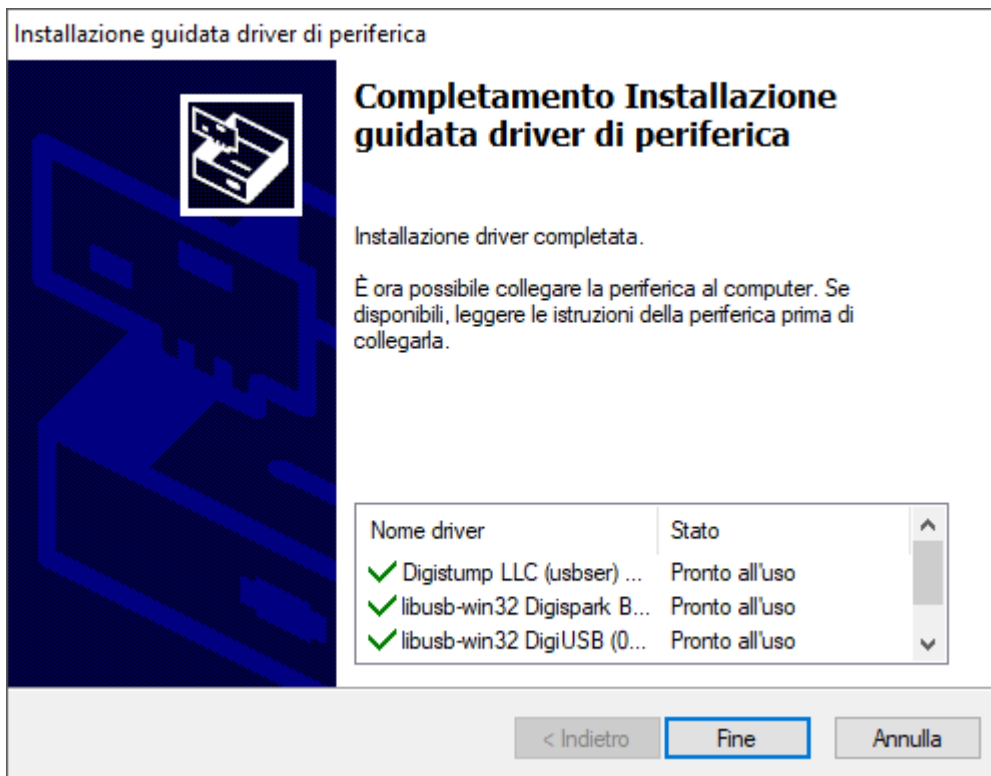
Installazione della scheda Digispark

Dopo questa breve introduzione vediamo come muovere i primi passi con questa piccola scheda.

Lo procedura è leggermente diversa a quella tipica per le altre schede Arduino.

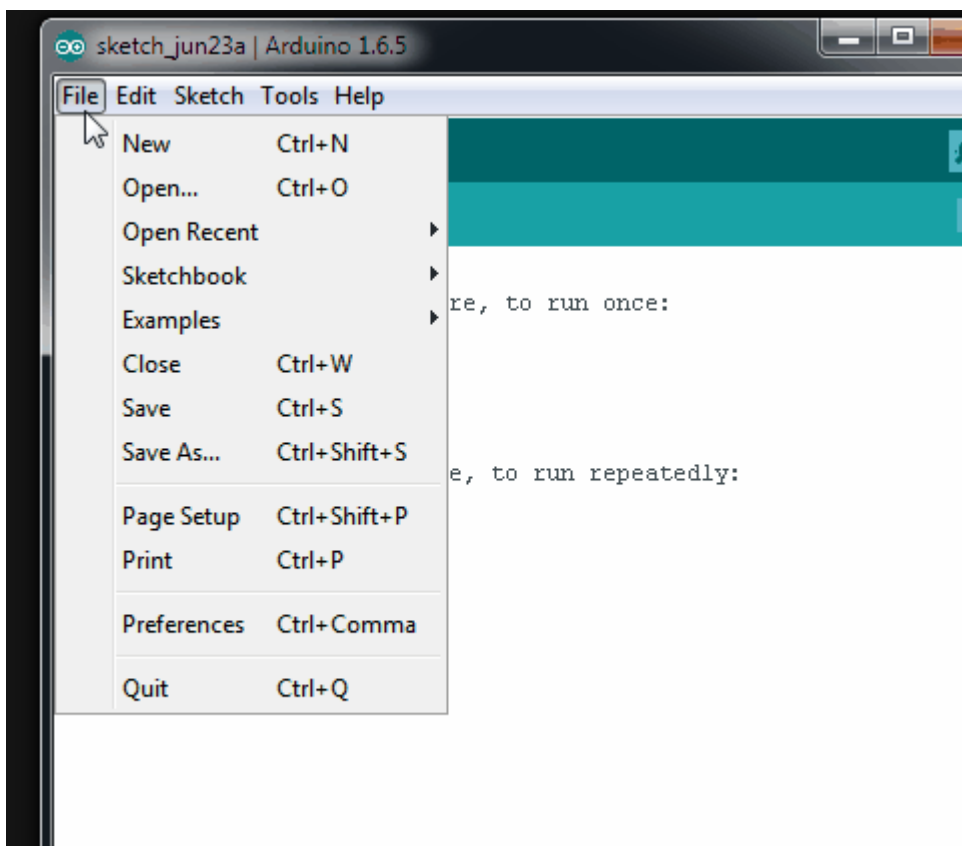
- Sarà quindi necessario scaricare e installare manualmente i driver per la scheda [Digispark](#), Scaricare, decomprimere ed eseguire "**Install Drivers**" (su sistemi a 32 bit) o "DPIInst64" (su sistemi a 64 bit). I file del driver si trovano [in questo link](#).



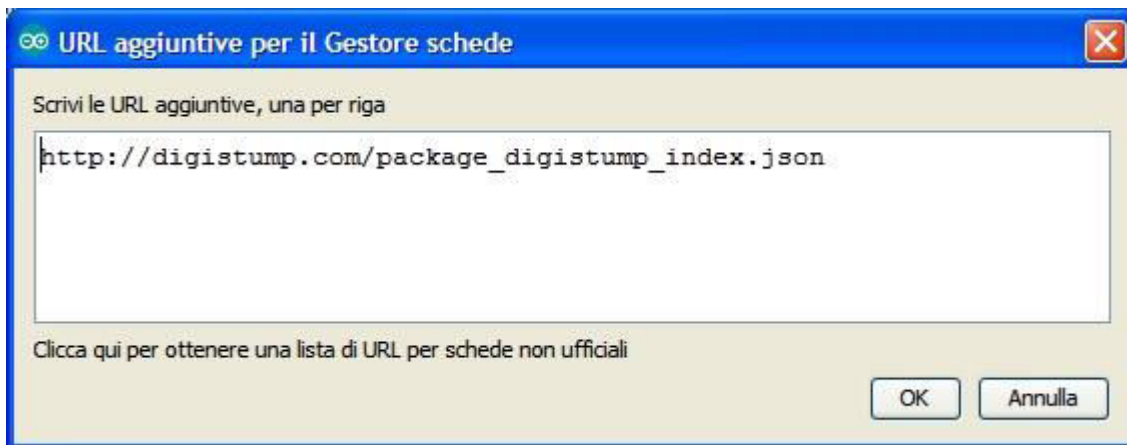


Nota: Attenzione, se inserite la scheda in una porta USB quando non richiesto dall'IDE, la scheda non sarà riconosciuta, ma questo è normale.

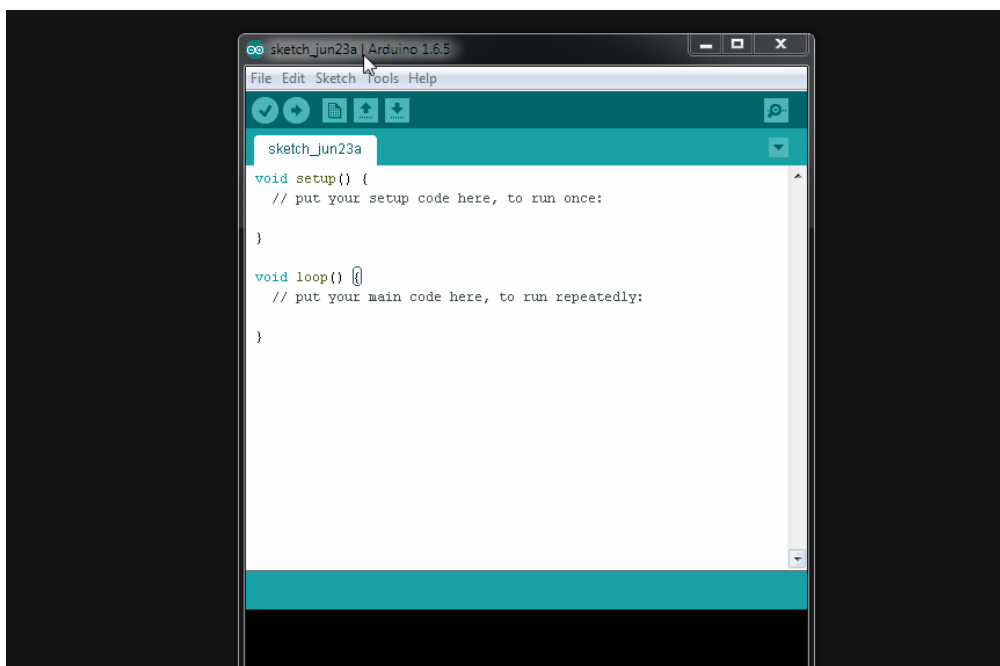
A questo punto, supponendo che l'IDE sia già installato (il test è stato fatto con la versione 1.8.3) andare al menu "**File**" e selezionare "**Impostazioni**"



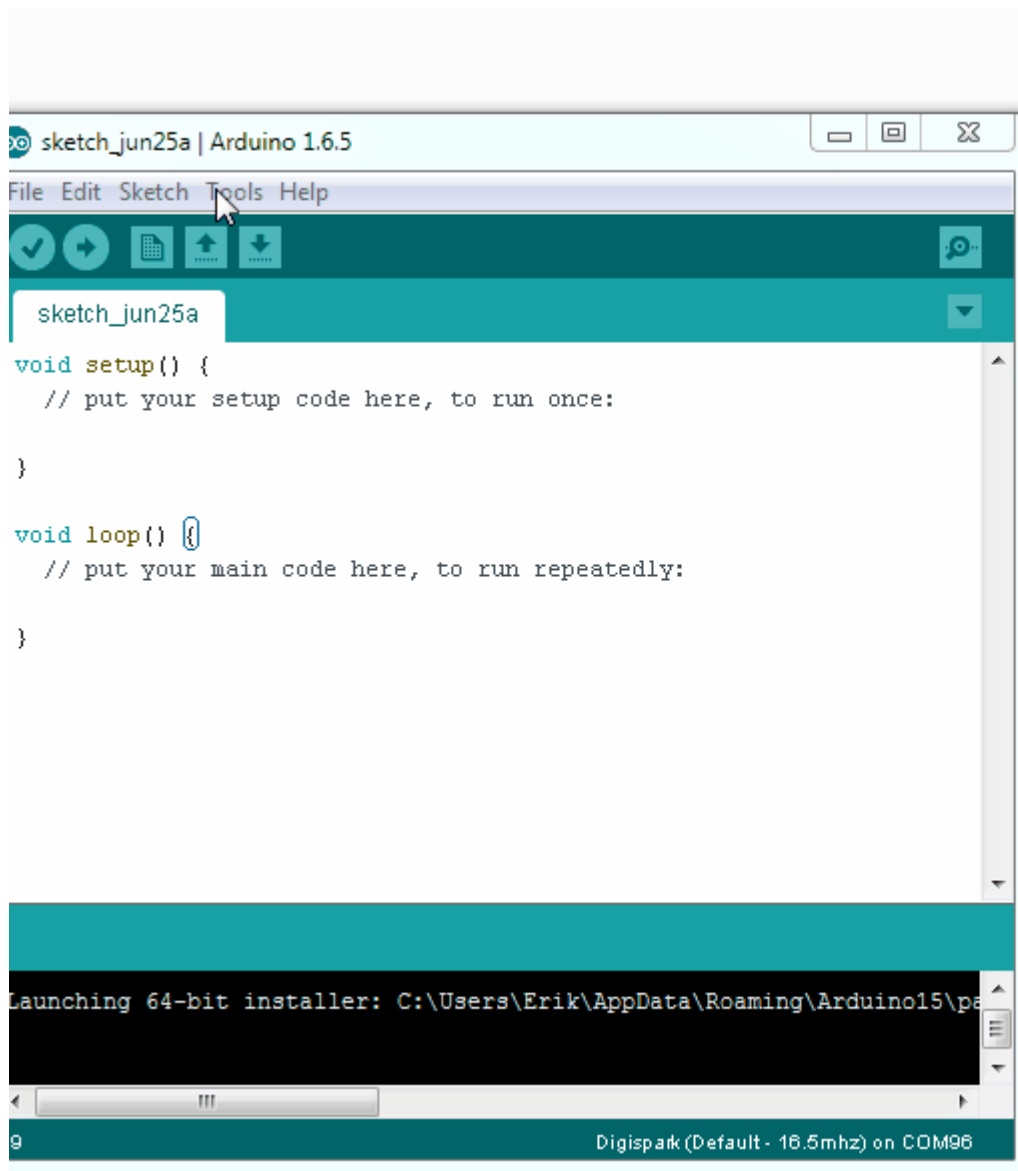
Nella casella con l'indicazione "**URL aggiuntive per il Gestore schede**" immettere: "**http://digistump.com/package_digistump_index.json**" e fare clic su OK.



Andare ora al menu "**Strumenti**" e poi nel sottomenu "**Gestore schede**": dopo una breve attesa, dal menu a discesa **Tipo** selezionare "**Fornito da terzi**". Selezionare il pacchetto "**Digitump AVR Boards**" e fare clic sul pulsante "**Installa**" e fare clic su OK.



- Verrà visualizzato il progresso di download nella barra di fondo della finestra "Gestore schede", quando sarà completata verrà visualizzato "**INSTALLED**" accanto a quella voce dell'elenco.
- Con l'installazione completa, chiudere la finestra "Gestore schede" e selezionare **Digispark** dal menu Strumenti → Schede. "**Digispark (Default - 16.5mhz)**"

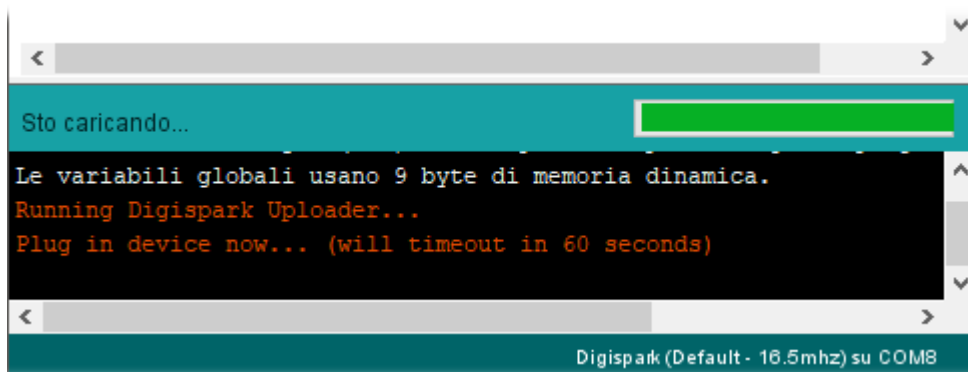


- a questo punto l'installazione può considerarsi completa

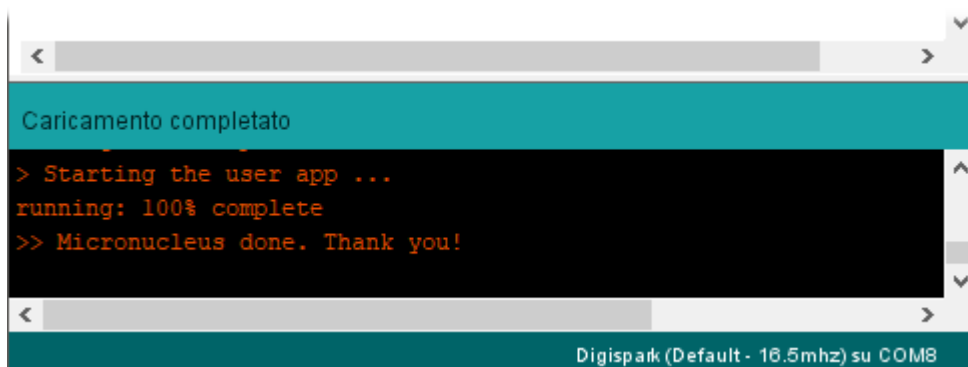
Uso della scheda Digispark con l'IDE

La scheda Digispark funziona un po' diversamente da altri prodotti compatibili con Arduino. La programmazione segue una procedura diversa.

- Verificare che dal menu Strumenti sia selezionata la Scheda → Digispark (default- 16.5Mhz), non è necessario modificare la voce programmatore.
- Scrivere un codice, aprire un codice precedentemente salvato o aprire un esempio Digispark.
- **Non è necessario collegare il Digispark prima di richiamare il caricamento**
- Premere il pulsante di caricamento. Dopo la compilazione nella casella di stato inferiore sarà richiesto di collegare il vostro Digispark - a questo punto è necessario collegarlo - oppure scollegarlo e ricollegarlo.



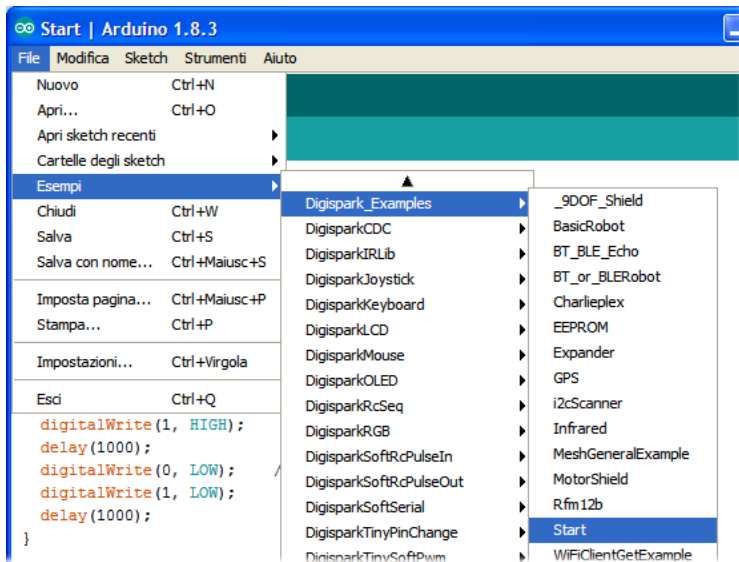
- Si vedrà il progresso del caricamento e al termine la scritta "**Caricamento terminato**", il codice sarà eseguito immediatamente sul Digispark.



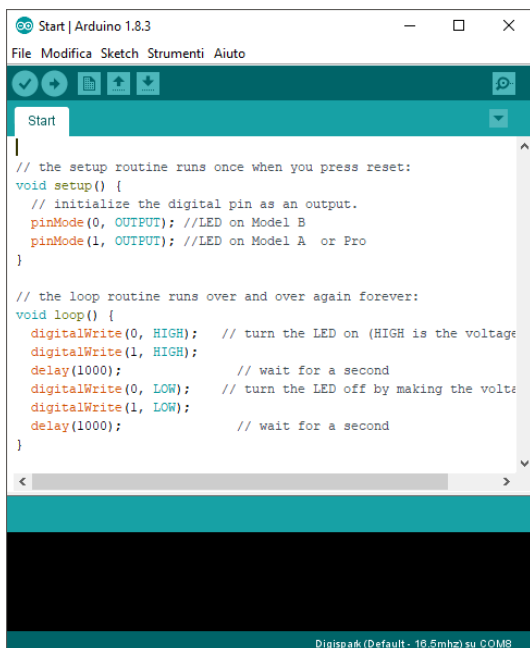
- Se si scollega il Digispark e si inserisce nuovamente o si collega ad un'altra sorgente di alimentazione, si verificherà un ritardo di 5 secondi prima che il codice programmato sia eseguito. Questo ritardo è dovuto al controllo di Digispark per verificare se si sta cercando di programmarlo.

Primo test

Per testare la scheda e la procedura, si può utilizzare l'esempio che si può trovare passando a **File** → **Esempi** → **Digispark_Examples** → **Start**



Ecco il codice che verrà caricato:



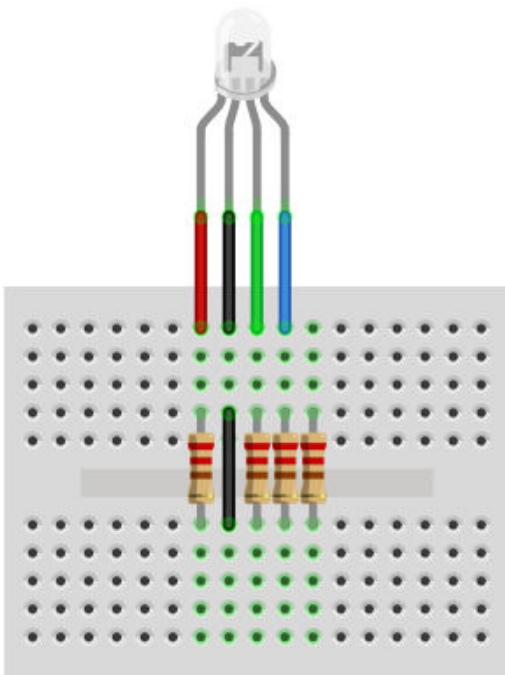
A questo punto inserite la vostra Digispark ed attendete che il caricamento vada a buon fine. Il Led sulla scheda lampeggerà al ritmo di 1Hz.

Led RGB:

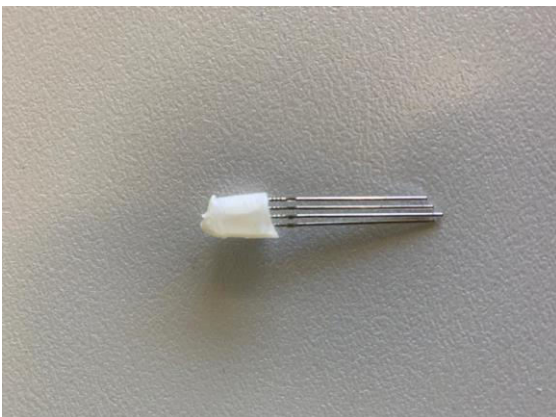
Come utilizzare il Led RGB

Per l'utilizzo di un led RGB bisogna avere come minimo a disposizione 3 pin liberi, e un pin che deve essere collegato alla massa.

Ai pin che controllano il colore del led bisogna attaccarci 3 resistenza da 150 Ohm.



Nell'immagine a sinistra si può vedere come bisogna attaccare il Led RGB. Al primo pin partendo da sinistra bisogna collegare il pin che imposterà il colore rosso al led. Nel secondo bisogna attaccare la massa, ovvero quel pin sull'Arduino con scritto GND oppure "-". Nel terzo bisogna collegare il pin che imposterà il colore verde al led. Nell'ultimo per esclusione si andrà a collegare il pin che imposterà il colore blu al led. Le resistenze che ho utilizzato nello schema sono solo indicative e non rispecchiano il vero colore della resistenza.



Un aiuto per migliorare l'esperienza con il Led RGB è di attaccarci un giro di scotch intorno così da rendere più omogeneo il colore che si andrà a vedere.

Come utilizzare la libreria del Led RGB:

L'accensione del Led RGB può avvenire grazie ad una libreria creata e ideata da me e Paolo Weishaupt.

Per il giusto funzionamento bisogna fare i seguenti passaggi:

- **Accertarsi che la libreria sia stata inclusa nel progetto:**

Facilmente verificabile controllando che all'inizio del programma ci sia scritta una stringa di codice simile a questa:

```
#include "LightLed.h"
```

- **Verificare che la variabile sia stata creata in modo corretto:**

Il modo più facile per dichiarare una variabile è il seguente:

LightLed* [nomeVariabile]; e nel metodo setup istanziarla.

- **Istanziare variabile:**

A questo punto bisognerà istanziarla, quindi recarsi nel metodo setup e istanziare la variabile dichiarata in precedenza nel seguente modo:

```
[nomeVariabile] = new LightLed(pin rosso, pin blu, pin verde);
```

- **Utilizzare il metodo per l'accensione del led:**

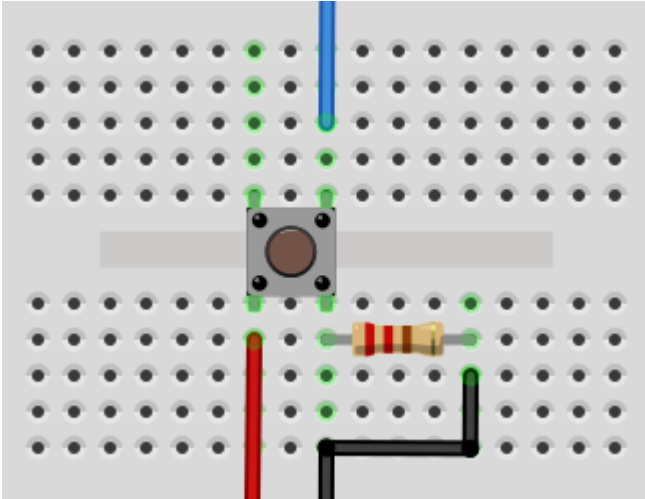
Per accendere il led bisognerà quindi chiamare un semplice metodo facilmente utilizzabile nel seguente modo:

```
[nomeVariabile]->lightLed(255, 0, 0);
```

Il seguente codice colorerà di rosso il led. I valori possono andare da 0 fino a 255.

Push button:

Come utilizzare un pulsante:



Nell'immagine a sinistra si può vedere come bisogna attaccare il pulsante al circuito. Il pin rosso sta a significare il "+", quello nero il "-" e quello blu il dato che si andrà a leggere che sarà 0 se il pulsante non sarà premuto o 1 quando lo sarà.

Come utilizzare la libreria del pulsante:

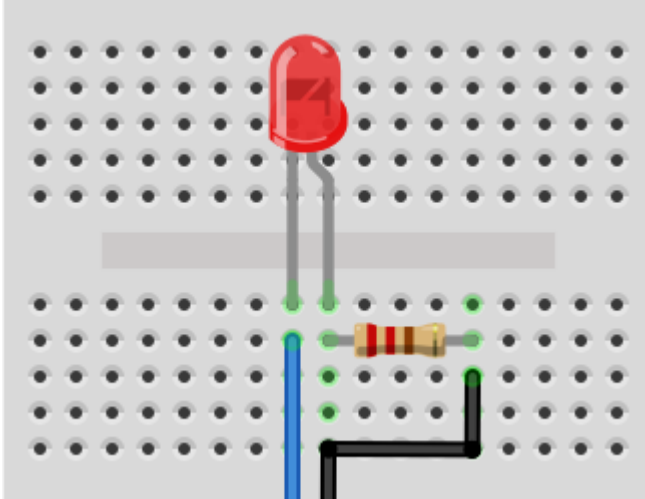
La lettura del pulsante può avvenire grazie ad una libreria creata e ideata da me e Paolo Weishaupt.

Per il giusto funzionamento bisogna fare i seguenti passaggi:

- **Accertarsi che la libreria sia stata inclusa nel progetto:**
Facilmente verificabile che all'inizio del programma ci sia scritta una stringa di codice simile a questa:
`#include "ButtonState.h"`
- **Verificare che la variabile sia stata creata in modo corretto:**
Il modo più facile per dichiarare una variabile è il seguente:
`ButtonState* [nomeVariabile];` e nel metodo setup istanziarla.
- **Istanziare variabile:**
A questo punto bisognerà istanziarla, quindi recarsi nel metodo setup e istanziare la variabile dichiarata in precedenza nel seguente modo:
`[nomeVariabile] = new ButtonState(pin del pulsante);`
- **Utilizzare il metodo per la lettura del pulsante:**
Per leggere il pulsante bisognerà quindi chiamare un semplice metodo facilmente utilizzabile, per esempio nel seguente modo:
`[nomeVariabile]->getStateButton()`
Il seguente codice ritornerà lo stato del pulsante.

Led:

Come utilizzare un Led:



Nell'immagine a sinistra si può vedere come bisogna attaccare il Led al circuito. Il cavo blu sta ad indicare lo stato di scrittura. Il cavo nero invece il "-". La resistenza bisogna metterla per non rischiare di bruciare il led.

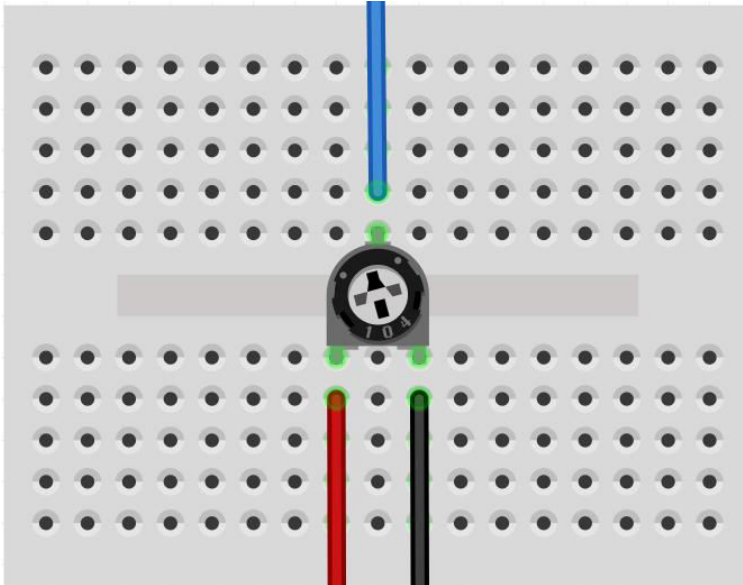
Come utilizzare la libreria del Led:

L'accensione del led può avvenire grazie ad una libreria creata e ideata da me e Paolo Weishaupt. Per il giusto funzionamento bisogna fare i seguenti passaggi:

- **Accertarsi che la libreria sia stata inclusa nel progetto:**
Facilmente verificabile che all'inizio del programma ci sia scritta una stringa di codice simile a questa:
`#include "LightSingleLed.h"`
- **Verificare che la variabile sia stata creata in modo corretto:**
Il modo più facile per dichiarare una variabile è il seguente:
`ButtonState* [nomeVariabile];` e nel metodo setup istanziarla.
- **Istanziare variabile:**
A questo punto bisognerà istanziarla, quindi recarsi nel metodo setup e istanziare la variabile dichiarata in precedenza nel seguente modo:
`led = new LightSingleLed(pin del led);`
- **Utilizzare il metodo per l'accensione del led**
Per accendere il led bisognerà quindi chiamare un semplice metodo facilmente utilizzabile nel seguente modo:
`led->lightSingleLed();`

Potenziometro:

Come utilizzare un potenziometro:



Nell'immagine a sinistra si può vedere come bisogna attaccare il potenziometro al circuito. Il cavo blu sta ad indicare lo stato di lettura. Il cavo nero invece il "-". Il cavo rosso invece sta a rappresentare il "+". La resistenza bisogna metterla per non rischiare di bruciare il led.

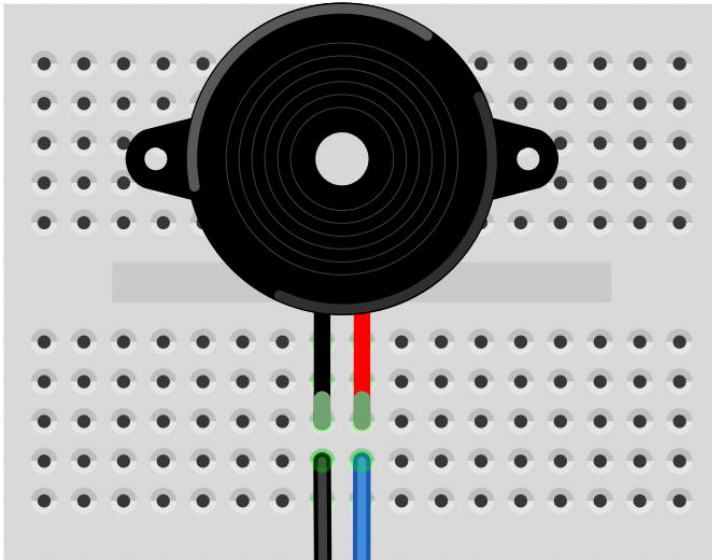
Come utilizzare la libreria del potenziometro:

La lettura dello stato del potenziometro si può eseguire grazie alla libreria scritta e ideata da me e Paolo Weishaupt.

- **Accertarsi che la libreria sia stata inclusa nel progetto:**
Facilmente verificabile controllando che all'inizio del programma ci sia scritta una stringa di codice simile a questa:
`#include "Potenziometro.h"`
- **Verificare che la variabile sia stata creata in modo corretto:**
Il modo più facile per dichiarare una variabile è il seguente:
`Potenziometro* [nomeVariabile];` e nel metodo setup istanziarla.
- **Istanziare variabile:**
A questo punto bisognerà istanziarla, quindi recarsi nel metodo setup e inizializzare la variabile dichiarata in precedenza nel seguente modo:
`potenziometro = new Potenzometro(pin di lettura del potenziometro);`
- **Utilizzare il metodo per la lettura del potenziometro**
Per leggere lo stato del potenziometro bisognerà quindi chiamare un semplice metodo facilmente utilizzabile nel seguente modo:
`potenziometro->ottieniValorePotenziometro();`
Il metodo ritorna un valore di tipo int che va di solito da 0 a 1023, dove 0 è il valore minimo di resistenza che il potenziometro può assumere e 1023 il valore massimo.

Piezo Buzzer:

Come utilizzare un Piezo Buzzer:



Nell'immagine a sinistra si può vedere come bisogna attaccare il Piezo Buzzer al circuito. Il cavo blu sta ad indicare lo stato di scrittura. Il cavo nero invece il "-".

Come utilizzare la libreria del Piezo Buzzer:

La scrittura dello stato del piezo buzzer si può eseguire grazie alla libreria scritta e ideata da me e Paolo Weishaupt.

- **Accertarsi che la libreria sia stata inclusa nel progetto:**
Facilmente verificabile controllando che all'inizio del programma ci sia scritta una stringa di codice simile a questa:
`#include "Buzzer.h"`
- **Verificare che la variabile sia stata creata in modo corretto:**
Il modo più facile per dichiarare una variabile è il seguente:
`Buzzer* [nomeVariabile];` e nel metodo setup istanziarla.
- **Istanziare variabile:**
A questo punto bisognerà istanziarla, quindi recarsi nel metodo setup e inizializzare la variabile dichiarata in precedenza nel seguente modo:
`buzzer = new Buzzer(pin buzzer);`
- **Utilizzare il metodo per la lettura del potenziometro**
Per far suonare il piezo buzzer bisognerà quindi chiamare un semplice metodo facilmente utilizzabile nel seguente modo:
`buzzer->beep(tempo di suono, tonalità del suono);`

Sitografia

- <https://www.adrirobot.it/arduino/digispark/digispark.htm>