Sistema didattico per Arduino

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](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530604)

[Figura 2: Scheda DigiSpark 9](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530605)

[Figura 3: Configurazione delle porte di DigiSpark 9](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530606)

[Figura 4: Schema elettrico 10](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530607)

[Figura 5: Circuito elettrico 10](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530608)

[Figura 6: Design libreria per un Led RGB 10](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530609)

[Figura 7: Design libreria per un led 11](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530610)

[Figura 8: Design libreria per un pulsante 11](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530611)

[Figura 9: Design libreria per un piezo buzzer 12](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530612)

[Figura 10: Design libreria per un potenziometro 12](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530613)

[Figura 11: Gantt consuntivo 22](file:///C:\Users\Paolo%20Weishaupt\OneDrive\Documenti\Terzo%20anno\Modulo%20306\Sistema-didattico-per-Arduino\Documentazione\Weishaupt_Pezzotti_doc_Sistema_didattico_per_Arduino.docx#_Toc530614)

# Introduzione

## 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

## 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.

## 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

## 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.

## 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. |

## 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.

## Pianificazione

Figura 1: Gantt preventivo

## Analisi dei mezzi

### 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 |

### 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 |

### Hardware

Il progetto verrà sviluppato su una scheda DigiSpark facilmente reperibile online. La documentazione che abbiamo utilizzato sarà reperibile cliccando [qui](https://www.adrirobot.it/arduino/digispark/digispark.htm). 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.

### Specifiche Hardware

HP Omen 2016

|  |  |
| --- | --- |
| Processore | Intel Core i7-6700HQ |
| Scheda grafica | NVIDA 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

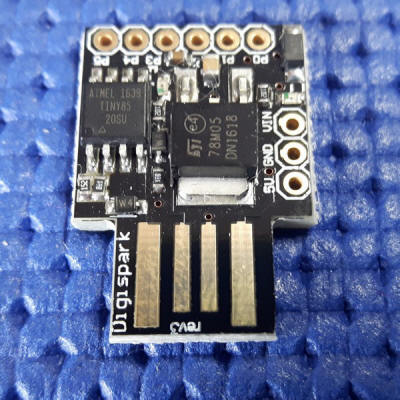
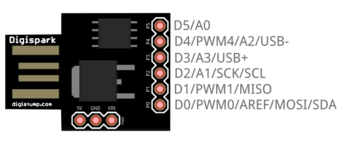
* Supporto per Arduino IDE 1.0+ (OSX / Win / Linux)
* Alimentazione tramite USB o sorgente esterna - 5v o 7-35v (consigliato 12v o meno, selezione automatica)
* Regolatore con uscita 5V - 500mA
* Integrato USB incorporato
* 6 pin I / O (2 usano solo USB se il programma comunica attivamente tramite USB, 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

# Progettazione

## Design dell’architettura del sistema

Figura 4: Schema elettrico

Figura 5: Circuito elettrico

## 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:

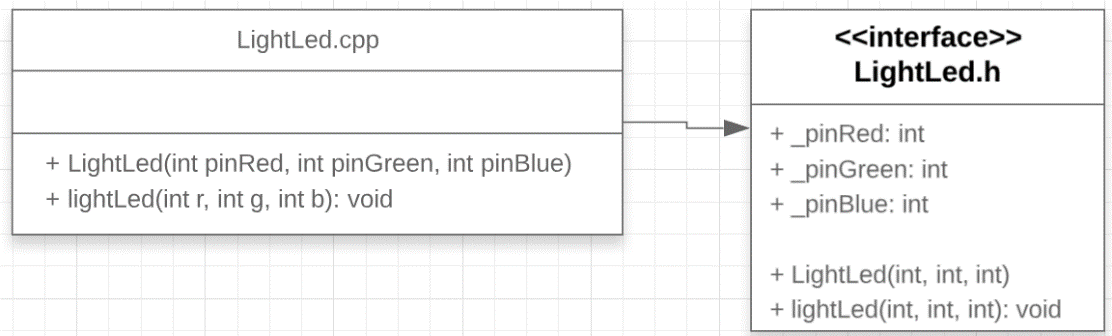
Questa libreria servirà per poter accendere un Led RGB

Figura 6: Design libreria per un Led RGB

Classi:

* LightLed.cpp
* LightLed.h

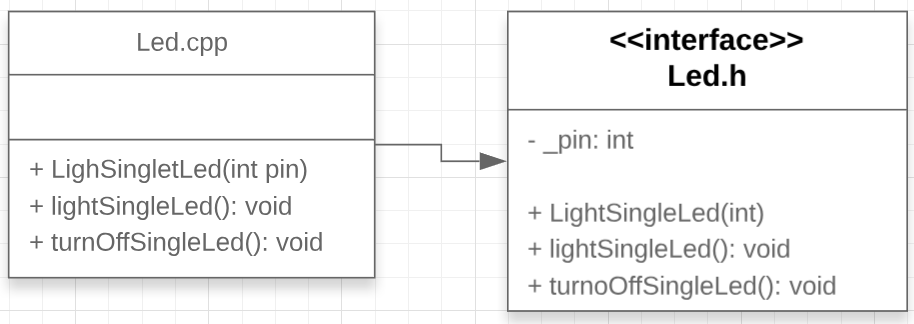
Attributi:

* 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 spegnare 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:

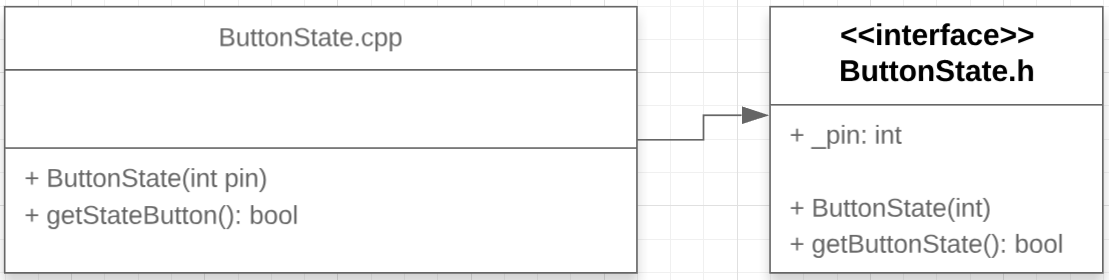
Questa libreria serve per leggere il valore di un pulsante, ossia se è premuto o no.

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:

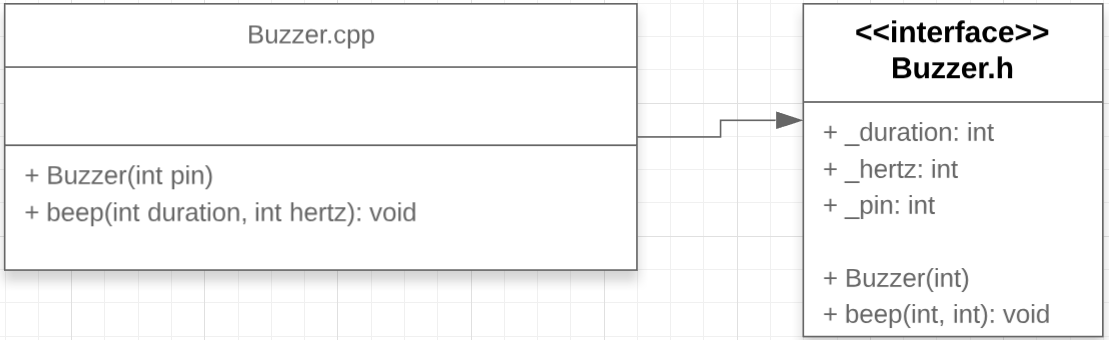
Questa libreria serve per far suonare un piezo buzzer.

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 Potenziometro:

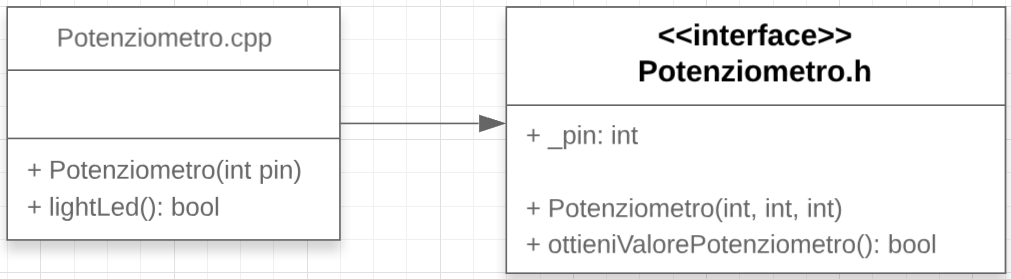
Questa libreria serve per leggere il valore di un potenziometro.

Figura 10: Design libreria per un potenziometro

Classi:

* Potenziometro.cpp
* Potenziometro.h

Attributo:

* int \_pin: pin del potenziometro

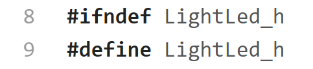
Metodi:

* 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

# Implementazione

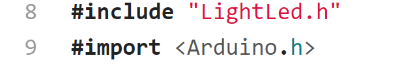
## Libreria per un Led RGB

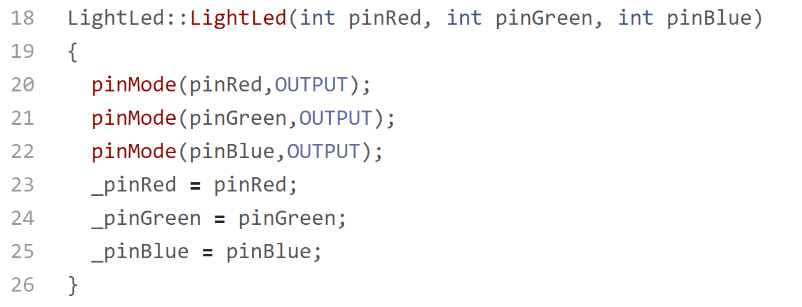
### LightLed.h

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.

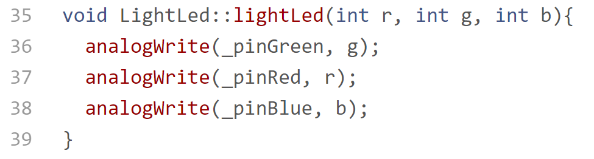
Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

### 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.

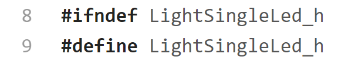
Il costruttore ci permette di definire i tre pin passati come argomento in output e di assegnarne il loro valore agli attributi corrispondenti.

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.



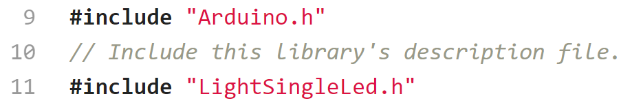
## Libreria per un led

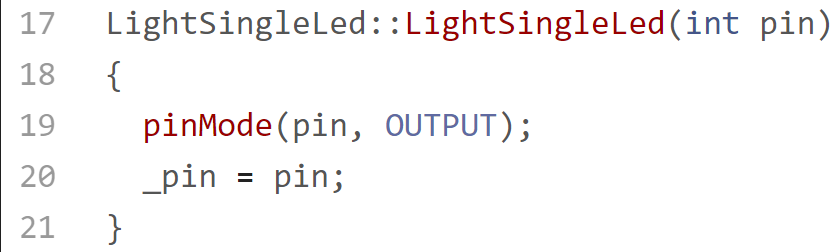
### Led.h

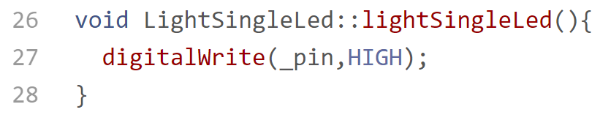
Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

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

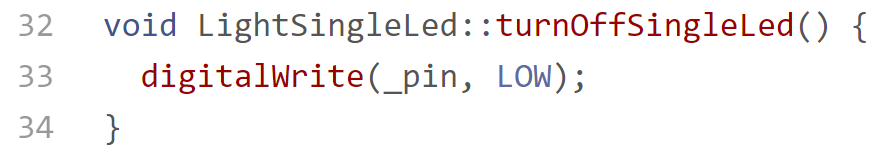
### Led.cpp

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.

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

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

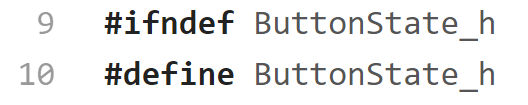
Il metodo turnOffSingleLed() tramite un digitalWrite() setta lo stato del Led si LOW e quindi si spegne.



## Libreria per un pulsante

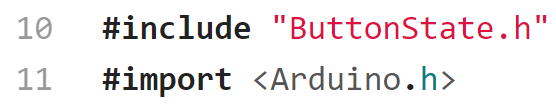
### ButtonState.h

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.

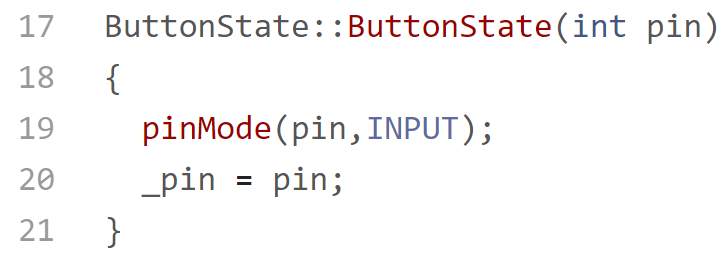


Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

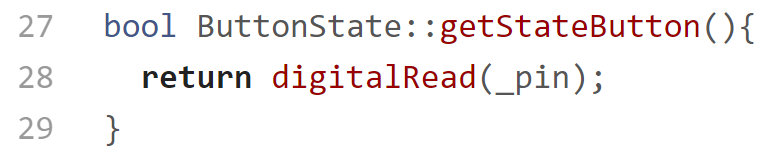
### ButtonState.cpp

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.

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

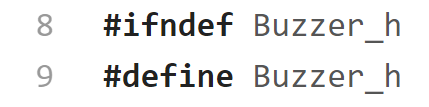


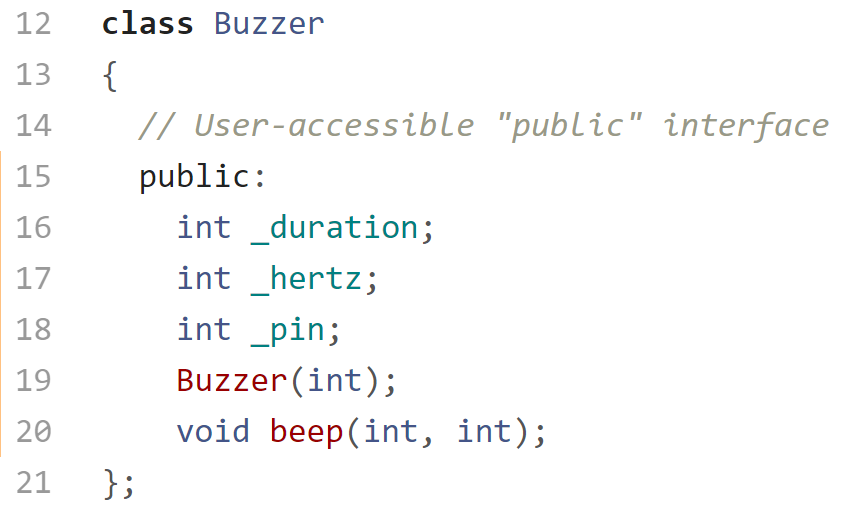
Il metodo getStateButton consente di vedere lo stato del pulsante (se è premuto o no) grazie a un digitalRead().



## Libreria per un piezo buzzer

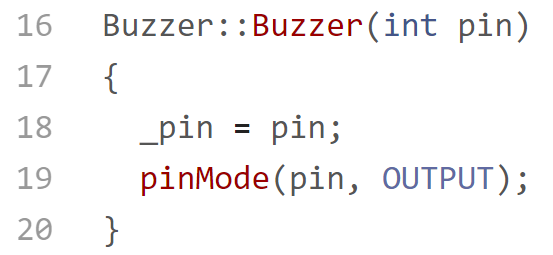
### Buzzer.h

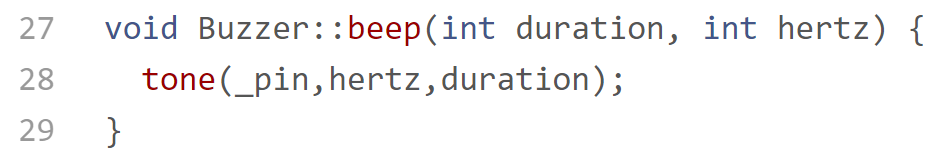
Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

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

### Buzzer.cpp

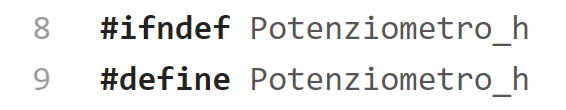
Grazie a queste due righe di codice includo l’header che definisce gli attributi e i metodi da utilizzare e importo il file Arduino.h che definisce i metodi di base di Arduino.

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

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.

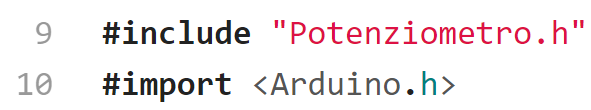
## Libreria per un potenziometro

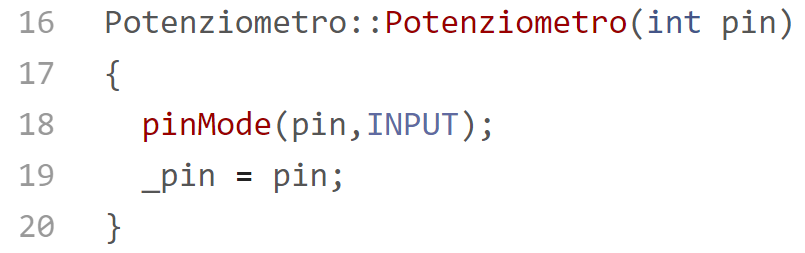
### Potenziometro.h

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.

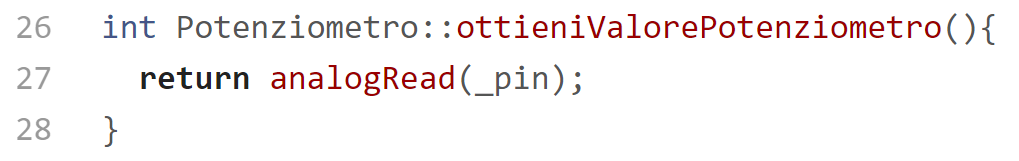
Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

### 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.

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

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



# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-001  REQ-001 | Nome: | Digispark |
| Descrizione: | Per il corretto funzionamento bisogna verificare che il digispark sia correttamente funzionante | | |
| Prerequisiti: | Nulla. | | |
| Procedura: | 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:  Riferimento: | TC-002  REQ-002 | Nome: | Attuatori |
| 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: | 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:  Riferimento: | TC-003  REQ-003 | Nome: | Ambiente ide di arduino |
| 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: | 1. Scaricare dal sito officiale l’IDE di Arduino 2. Provare a lanciare il programma 3. Far partine 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 qui è stato programmato (es. codice di prova blinking led -> il digispark farà lampeggiare il led di cui è fornito) | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-004  REQ-004 | Nome: | Funzione che legge stato pulsante |
| 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: | 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:  Riferimento: | TC-005  REQ-005 | Nome: | Funzione che incrementa stato del led |
| 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: | 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:  Riferimento: | TC-006  REQ-006 | Nome: | Funzione che scrive il valore desiderato |
| 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. | | |

## 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 rinscontrato 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 i pulsante era premuto incrementava il valore di un 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).

## Mancanze/limitazioni conosciute

La scheda Digispark è stata progettata con pochi pin quindi la possibilità di attacare 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.

# Consuntivo

Figura 11: Gantt consuntivo

# 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.

## Sviluppi futuri

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

## Considerazioni personali

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

# Bibliografia

## Sitografia

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

# Allegati

1. Quaderno dei Compiti
2. Diari di lavoro
3. Manuale di utilizzo
4. Prodotto:
   * GitHub: <https://github.com/PaoloWeishaupt/Sistema-didattico-per-Arduino>