Sistema didattico per Arduino

2 Implementazione 3

2.1 Libreria per un Led RGB 3

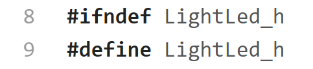
2.1.1 LightLed.h 3

2.1.2 LightLed.cpp 3

# Implementazione

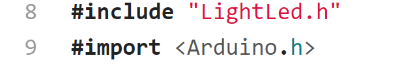
## Libreria per un led RGB

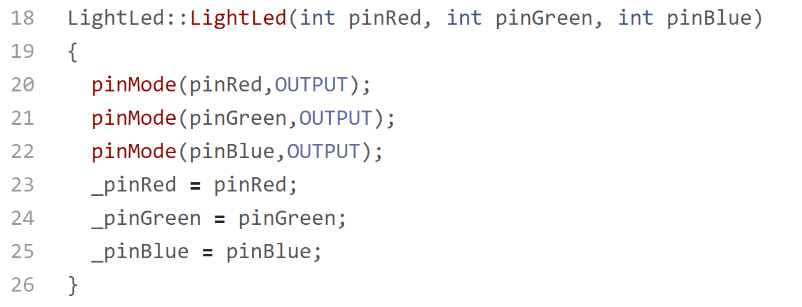
### LightLed.h

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.

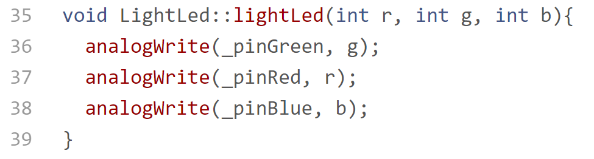
Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

### 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.

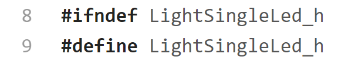
Il costruttore ci permette di definire i tre pin passati come argomento in output e di assegnarne il loro valore agli attributi corrispondenti.

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.



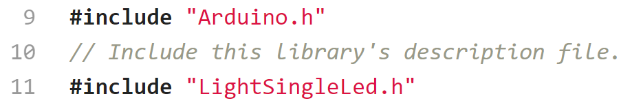
## Libreria per un led

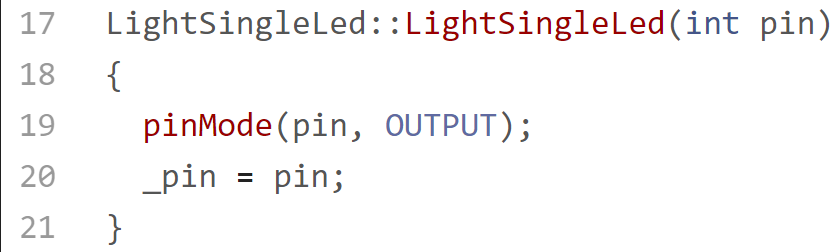
### Led.h

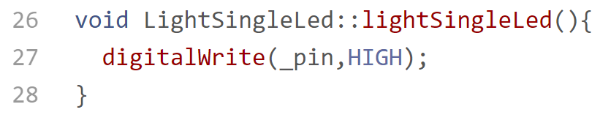
Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

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

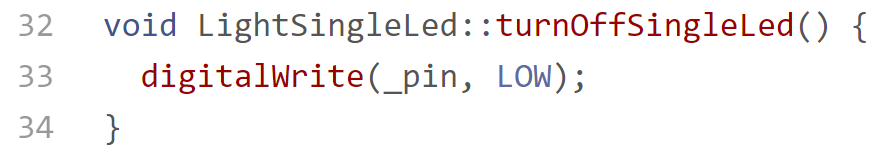
### Led.cpp

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.

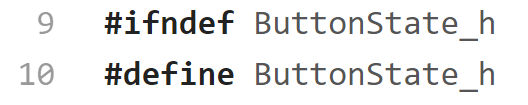
Il metodo turnOffSingleLed() tramite un digitalWrite() setta lo stato del Led si LOW e quindi si spegne.



## Libreria per un pulsante

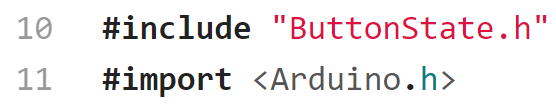
### ButtonState.h

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.

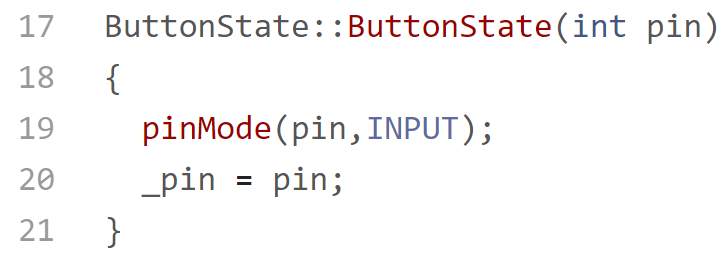


Così si definiscono gli attributi e i metodi che devono essere implementati dalle classi che utilizzano questa interfaccia.

### 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.

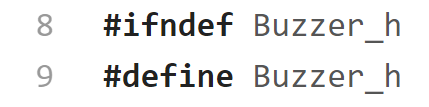
Il costruttore permette di settare il pin passato come parametro in input e lo assegna alla variabile \_pin.

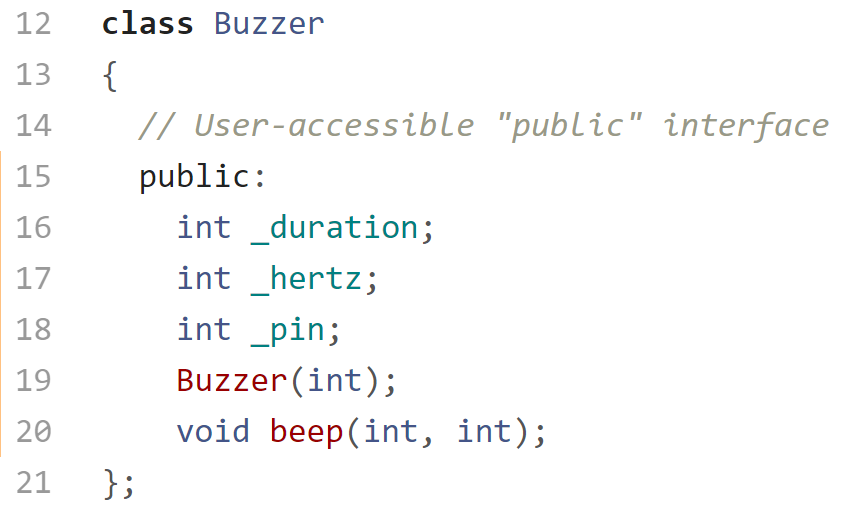


Il metodo getStateButton consente di vedere lo stato del pulsante (se è premuto o no) grazie a un digitalRead().

## Libreria per un piezo buzzer

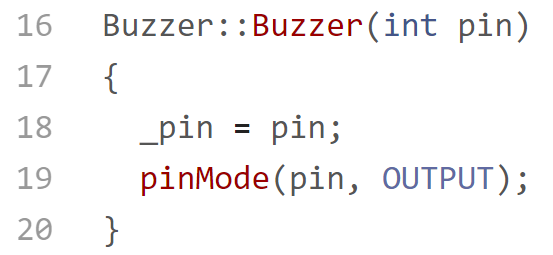
### Buzzer.h

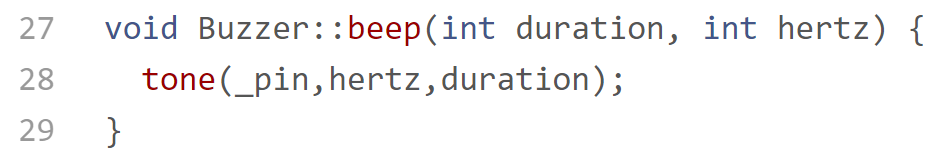
Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

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

### Buzzer.cpp

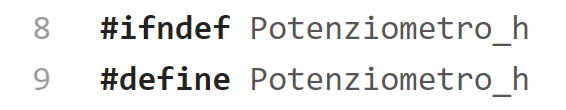
Grazie a queste due righe di codice includo l’header che definisce gli attributi e i metodi da utilizzare e importo il file Arduino.h che definisce i metodi di base di Arduino.

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.

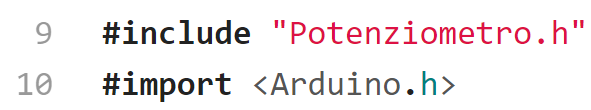
## Libreria per un potenziometro

### Potenziometro.h

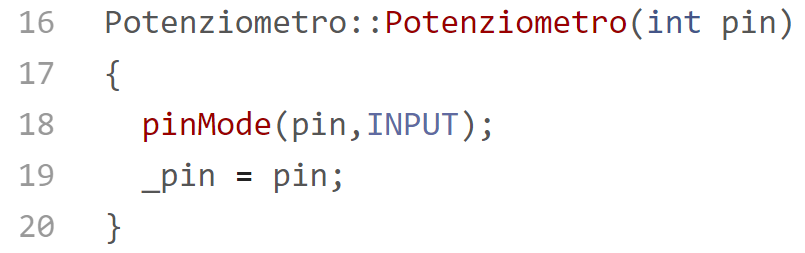
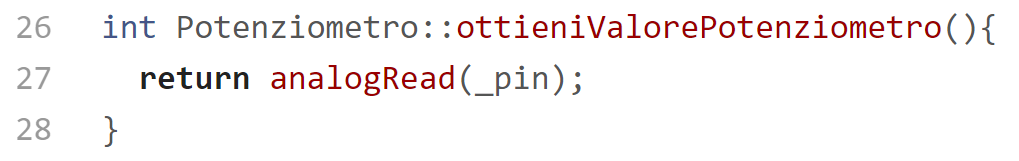
Questa parte di codice serve per impedire che venga importato lo stesso file header più di una volta altrimenti verrebbero generati degli errori a causa della presenza di più metodi definiti allo stesso modo.

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.