

Strutturare un Progetto Arduino con File `.h` e `.cpp`

Prof. Bernardis Pierluigi

8/10/2025

Introduzione

Creare una classe in Arduino consente di scrivere codice più ordinato, riutilizzabile e facile da mantenere. In questa guida costruiremo passo passo una classe `Led` per controllare un LED con metodi come `on()`, `off()`, `toggle()` e `setBrightness()`.

Struttura del Progetto

Creare dunque una cartella chiamata `ProgettoLED`. All'interno della cartella dovranno essere salvati questi file:

```
ProgettoLED/  
|-- main.ino  
|-- Led.h  
|-- Led.cpp
```

- **main.ino**: il programma principale.
- **Led.h**: file di intestazione, contiene la dichiarazione della classe.
- **Led.cpp**: file di implementazione, contiene il codice dei metodi.

1 Dichiarazione della Classe – File Led.h

Protezione dall'inclusione multipla

```
1 #ifndef LED_H          //if not defined
2 #define LED_H          //define
```

Quando un progetto è composto da più file, può capitare che lo stesso file venga incluso più volte. Queste due istruzioni servono a evitare errori dovuti all'inclusione multipla: il file viene incluso solo la prima volta.

Dichiarazione della classe

```
3 class LED {
4     private:
5         int pin;      // Numero del pin collegato al LED
6         int stato;    // Stato del LED: 0 = spento, 1 = acceso
```

La parte `private` contiene le variabili interne alla classe:

- `pin`: memorizza il numero del pin.
 - `stato`: tiene traccia dello stato del LED.
-

Dichiarazione dei metodi pubblici

```
7     public:
8         LED(int p);          // Costruttore
9         void on();           // Accende il LED
10        void off();          // Spegne il LED
11        void toggle();       // Inverte lo stato
12        void setBrightness(int value); // Luminosità PWM
13};
```

La parte `public` contiene i metodi accessibili dal programma principale.

Chiusura della protezione

```
14 #endif
```

Chiude la protezione contro le inclusioni multiple, facendo riferimento alla prima riga di codice di questo file.

2 Implementazione della Classe – File Led.cpp

Inclusione dei file necessari

```
1 #include "Led.h"
2 #include <Arduino.h>
```

"Led.h": include la dichiarazione della classe.

<Arduino.h>: include tutte le funzioni Arduino standard.

Costruttore della classe

```
3 LED::Led(int p) {
4     pin = p;                // Salva il numero del pin
5     pinMode(pin, OUTPUT);   // Imposta il pin come uscita
6     stato = 0;              // LED spento all'inizio
7 }
```

Il simbolo `::` (chiamato *operatore di risoluzione di ambito*) indica che il costruttore (o metodo) `Led()` appartiene alla classe `Led`. In altre parole, `LED::Led()` significa “il costruttore `Led()` della classe `LED`”. Questo operatore serve anche per distinguere tra funzioni con lo stesso nome appartenenti a classi diverse.

Accendere il LED

```
8 void LED::on() {
9     digitalWrite(pin, HIGH); // Pin a livello alto = LED acceso
10    stato = 1;
11 }
```

Spegnere il LED

```
12 void LED::off() {
13     digitalWrite(pin, LOW);  // Pin a livello basso = LED spento
14     stato = 0;
15 }
```

Invertire lo stato (toggle)

```
16 void LED::toggle() {
17     if (stato == 0)
18         on();    // Se era spento, accendi
19     else
20         off();   // Se era acceso, spegni
21 }
```

Regolare la luminosità

```
22 void LED::setBrightness(int value) {
23     analogWrite(pin, value); // Scrive un valore PWM (0-255)
24     stato = (value > 0) ? 1 : 0; // Aggiorna lo stato interno
25 }
```

3 Programma Principale – File main.ino

Inclusione della classe

```
1 #include "Led.h"
```

Serve per poter utilizzare la classe `Led` creata nei file precedenti.

Creazione dell'oggetto LED

```
2 LED led1(2); // LED collegato al pin digitale 2
```

Funzione setup()

```
3 void setup() {  
4   // Il costruttore ha già configurato il pin, quindi non serve altro  
5 }
```

Funzione loop()

```
6 void loop() {  
7   led1.on(); // Accende il LED  
8   delay(1000); // Attende 1 secondo  
9   led1.off(); // Spegne il LED  
10  delay(1000); // Attende 1 secondo  
11 }
```

4 Esercizi Pratici

Creare una cartella per ogni esercizio proposto qui di seguito:

1. Accendere un LED per 2 secondi e spegnerlo per 2 secondi in modo ciclico.
2. Far lampeggiare un LED con periodo di 1 secondo.
3. Alternare due LED in modo che uno sia acceso mentre l'altro è spento.
4. Usare `toggle()` per invertire lo stato ogni 500 ms.
5. Creare un effetto “respiro” con `setBrightness()`, aumentando e diminuendo la luminosità da 0 a 255.

A Appendice – Codici Completi del Progetto

A.1 File Led.h

```
1 #ifndef LED_H
2 #define LED_H
3
4 class LED {
5     private:
6         int pin;
7         int stato;
8
9     public:
10         Led(int p);
11         void on();
12         void off();
13         void toggle();
14         void setBrightness(int value);
15 };
16
17 #endif
```

A.2 File Led.cpp

```
1 #include "Led.h"
2 #include <Arduino.h>
3
4 LED::Led(int p) {
5     pin = p;
6     pinMode(pin, OUTPUT);
7     stato = 0;
8 }
9
10 void LED::on() {
11     digitalWrite(pin, HIGH);
12     stato = 1;
13 }
14
15 void LED::off() {
16     digitalWrite(pin, LOW);
17     stato = 0;
18 }
19
20 void LED::toggle() {
21     if (stato == 0)
22         on();
23     else
24         off();
25 }
26
27 void LED::setBrightness(int value) {
28     analogWrite(pin, value);
29     stato = (value > 0) ? 1 : 0;
30 }
```

A.3 File main.ino

```
1 #include "Led.h"
2
3 LED led1(2);
4
5 void setup() {
6 }
7
8 void loop() {
9     led1.on();
10    delay(1000);
11    led1.off();
12    delay(1000);
13 }
```