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

```
#ifndef LED_H         //if not defined
#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

```
class LED {
  private:
    int pin;    // Numero del pin collegato al LED
    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

```
  public:
    LED(int p);               // Costruttore
    void on();                // Accende il LED
    void off();               // Spegne il LED
    void toggle();            // Inverte lo stato
    void setBrightness(int value); // Luminosita PWM
};
```

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

—

Chiusura della protezione

```
#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

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

`"Led.h"`: include la dichiarazione della classe.
`<Arduino.h>`: include tutte le funzioni Arduino standard.

Costruttore della classe

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

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

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

Spegnere il LED

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

Invertire lo stato (toggle)

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

Regolare la luminosità

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

---

3 Programma Principale – File `main.ino`

Inclusione della classe

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

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

—

Creazione dell'oggetto LED

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

—

Funzione setup()

```
void setup() {
  // Il costruttore ha gia configurato il pin, quindi non serve altro
}
```

—

Funzione loop()

```
void loop() {
  led1.on();     // Accende il LED
  delay(1000);   // Attende 1 secondo
  led1.off();    // Spegne il LED
  delay(1000);   // Attende 1 secondo
}
```

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

```
#ifndef LED_H
#define LED_H

class LED {
  private:
    int pin;
    int stato;

  public:
    LED(int p);
    void on();
    void off();
    void toggle();
    void setBrightness(int value);
};

#endif
```

A.2 File Led.cpp

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

LED::LED(int p) {
  pin = p;
  pinMode(pin, OUTPUT);
  stato = 0;
}

void LED::on() {
  digitalWrite(pin, HIGH);
  stato = 1;
}

void LED::off() {
  digitalWrite(pin, LOW);
  stato = 0;
}

void LED::toggle() {
  if (stato == 0)
    on();
  else
    off();
}

void LED::setBrightness(int value) {
  analogWrite(pin, value);
  stato = (value > 0) ? 1 : 0;
}
```

A.3 File `main.ino`

```
#include "Led.h"

LED led1(2);

void setup() {
}

void loop() {
  led1.on();
  delay(1000);
  led1.off();
  delay(1000);
}
```