Corso di Automazione Intelligente e Robotica

Manuale di esercizi introduttivi in C++ con Arduino ed ESP32

Prof. Bernardis Pierluigi

Indice

Introduzione

Il presente manuale propone quindici esercizi finalizzati all'apprendimento della programmazione orientata agli oggetti applicata ad Arduino ed ESP32. Tutti gli esercizi utilizzano esclusivamente LED come dispositivi di output. La sequenza degli esercizi è progressiva: i primi dieci esercizi introducono la classe `Led` e i relativi metodi di controllo, mentre gli ultimi cinque simulano dispositivi reali mediante classi dedicate. Le esercitazioni sono accompagnate da indicazioni sui collegamenti elettrici, sul comportamento atteso e sull'organizzazione del codice.

1 Configurazione di base

Tutti i LED devono essere collegati in serie con una resistenza da 220 Ω tra il catodo del LED e GND. L'anodo del LED deve essere collegato al pin digitale di uscita. I pin utilizzati per PWM devono essere compatibili con la funzione `analogWrite` su ESP32. Quando si utilizzano più LED, ciascuno deve essere collegato a un pin separato con la stessa configurazione.

2 Esercizi

2.1 Esercizi 1–10: classe Led

Esercizio 1 – Accensione e spegnimento di un LED

L'obiettivo consiste nella creazione di una classe `Led` dotata dei metodi `on()` e `off()`. Si deve collegare un LED al pin 2 tramite resistenza. Il costruttore della classe deve inizializzare il pin come OUTPUT. Nel ciclo principale (`loop`) si deve accendere il LED per due secondi e spegnerlo per due secondi. Questo esercizio consente di comprendere come un oggetto possa gestire lo stato di un componente senza richiamare continuamente funzioni di basso livello.

Esercizio 2 – Lampeggio

Il LED deve lampeggiare con periodo di un secondo. È necessario utilizzare un oggetto della classe `Led` creata in precedenza. Nel ciclo principale, il LED deve alternare lo stato acceso/spento ogni 500 millisecondi, consentendo di osservare l'effetto lampeggiante.

Esercizio 3 – Toggle automatico

Il metodo `toggle()` deve essere implementato nella classe `Led` per cambiare lo stato del LED da acceso a spento e viceversa. Nel ciclo principale, il metodo deve essere richiamato periodicamente ogni 500 millisecondi. L'esercizio illustra l'uso di metodi interni per automatizzare comportamenti ripetitivi.

Esercizio 4 – Due LED alternati

Si devono collegare due LED ai pin 2 e 3 e creare due oggetti `Led`. L'obiettivo è ottenere un lampeggio alternato: quando il primo LED è acceso, il secondo deve essere spento e viceversa. Questo esercizio introduce la gestione simultanea di più oggetti e la sincronizzazione dei comportamenti.

Esercizio 5 – Sequenza di tre LED

Tre LED collegati ai pin 2, 3 e 4 devono essere controllati in sequenza. Ogni LED deve accendersi per 500 millisecondi e spegnersi prima di accendere il successivo. L'esercizio permette di comprendere l'uso di più oggetti per generare sequenze temporali.

Esercizio 6 – PWM: luminosità variabile

Un LED collegato a un pin PWM (ad esempio pin 5) deve variare la propria luminosità tramite il metodo `setBrightness(int value)`. Nel ciclo principale, il valore deve aumentare gradualmente da 0 a 255 e successivamente diminuire a 0. L'esercizio introduce il concetto di modulazione della potenza e controllo fine dei componenti elettronici.

Esercizio 7 – Semaforo

Tre LED, collegati ai pin 2, 3 e 4, devono simulare un semaforo. Si crea un oggetto `Led` per ciascun LED. La sequenza consiste nell'accendere il LED rosso per tre secondi, il LED verde per tre secondi e il LED giallo per un secondo. L'esercizio consente di combinare più oggetti e gestire logiche temporali complesse.

Esercizio 8 – Effetto rincorsa

Quattro LED collegati ai pin 2–5 devono accendersi uno alla volta con intervalli di 200 millisecondi, spegnendo il LED precedente prima di accendere quello successivo. Si utilizza un array di oggetti `Led` per realizzare l'effetto visivo. Questo esercizio illustra la manipolazione di array di oggetti e la

creazione di sequenze animate. (Se non sai cosa sono gli array, crea 5 oggetti LED)

Esercizio 9 – Lampeggio alternato su quattro LED

Quattro LED collegati ai pin 2–5 devono essere gestiti in due coppie: LED1+LED2 e LED3+LED4. La prima coppia deve lampeggiare mentre la seconda è spenta e viceversa, alternando lo stato ogni 500 millisecondi. L'esercizio introduce il concetto di gruppi e controllo simultaneo di più oggetti.

Esercizio 10 – PWM opposto su due LED

Due LED collegati ai pin 2 e 3 devono variare la luminosità in modo opposto. Quando il LED1 aumenta la luminosità, il LED2 deve diminuirla. Questo esercizio consente di comprendere la sincronizzazione di modifiche di stato variabili su più oggetti.

2.2 Esercizi 11–15: simulazioni di dispositivi reali

Esercizio 11 – Ventola

In questo esercizio si simula il funzionamento di una ventola. Il LED, collegato al pin 2, rappresenta la ventola stessa. La classe `Ventola` deve contenere due metodi: `bassa()` che mantiene il LED acceso fisso per indicare velocità ridotta e `alta()` che fa lampeggiare il LED rapidamente per indicare velocità elevata. Nel ciclo principale, le due modalità devono alternarsi con una pausa di un secondo. Questo esercizio permette di comprendere come un LED possa rappresentare un attuatore in termini di velocità variabile.

Esercizio 12 – Motore DC

Questo esercizio consiste nella simulazione di un motore DC mediante due LED collegati ai pin 2 e 3. La classe `Motore` deve implementare i metodi `avanti()`, `indietro()` e `fermo()`. L'accensione del LED1 rappresenta la rotazione del motore in senso orario, il LED2 in senso antiorario, mentre entrambi spenti indicano motore fermo. Nel ciclo principale, il programma deve alternare i due stati di rotazione con pause di un secondo. L'esercizio illustra la logica di controllo dei motori mediante oggetti software e LED.

Esercizio 13 – Valvola automatica simulata

L'esercizio consiste nel simulare il comportamento di una valvola che può essere aperta o chiusa. Collegate un LED al pin 2. La classe `Valvola` deve implementare due metodi: `apri()` che accende il LED e indica che la valvola è aperta, e `chiudi()` che spegne il LED. Nel ciclo principale, il programma deve alternare lo stato della valvola ogni due secondi. Questo esercizio consente di comprendere come i LED possano rappresentare dispositivi di azione meccanica, introducendo il concetto di attuatore simulato.

Esercizio 14 – Cancello automatico simulato

Questo esercizio simula un cancello automatico mediante due LED collegati ai pin 2 e 3. Il LED di apertura indica il movimento verso lo stato aperto, mentre il LED di chiusura indica il movimento verso lo stato chiuso. La classe `Cancello` deve implementare i metodi `apri()` e `chiudi()`. Nel ciclo principale, il programma deve alternare apertura e chiusura ogni tre secondi. L'esercizio introduce la gestione di più stati per un attuatore simulato mediante LED e consente di comprendere la logica di funzionamento di un sistema meccanico controllato digitalmente.

Esercizio 15 – Motore DC simulato con ponte H L9110S

L'obiettivo di questo esercizio è simulare un motore DC controllato da un ponte H L9110S. Si devono collegare due LED in antiparallelo ai pin 2 e 3: il LED1 acceso indica rotazione in senso orario, il LED2 acceso indica rotazione antioraria, entrambi spenti indicano motore fermo. La classe `MotoreH` deve implementare i metodi `avanti()`, `indietro()` e `fermo()`. Nel ciclo principale, il programma deve alternare i due stati di rotazione con pause di un secondo tra ciascun cambiamento. Questo esercizio permette di comprendere la logica del ponte H e la simulazione della direzione di un motore mediante LED, preparandosi all'uso reale di un L9110S o altri driver di motori.

3 Soluzioni degli esercizi

```
// Classe Led
class Led {
  int pin;
  int stato; // 0 = spento, 1 = acceso
public:
  Led(int p) {
    pin = p;    //incapsulamento
    pinMode(pin, OUTPUT);
    stato = 0;
  }

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

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

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

  void setBrightness(int value) {
    analogWrite(pin, value); // PWM
    if(value > 0){
        stato = 1;
    } else {
        stato = 0;
    }
  }
};
```

4 Esercizi 1–10: classe Led

Esercizio 1 – Accensione e spegnimento di un LED

Collegare un LED al pin 2 tramite resistenza. Accendere il LED per due secondi e spegnerlo per due secondi.

```
Led led1(2);

void loop() {
  led1.on();
  delay(2000);
  led1.off();
  delay(2000);
}
```

Esercizio 2 – Lampeggio

Il LED deve lampeggiare con periodo di un secondo.

```
Led led1(2);

void loop() {
  led1.on();
  delay(500);
  led1.off();
  delay(500);
}
```

Esercizio 3 – Toggle automatico

Implementare il metodo `toggle()` per cambiare lo stato del LED ogni 500 millisecondi.

```
Led led1(2);

void loop() {
  led1.toggle(); delay(500);
}
```

Esercizio 4 – Due LED alternati

Due LED collegati ai pin 2 e 3 devono lampeggiare alternati.

```
Led led1(2);
Led led2(3);

void loop() {
  led1.on(); led2.off(); delay(500);
  led1.off(); led2.on(); delay(500);
}
```

Esercizio 5 – Sequenza di tre LED

Tre LED collegati ai pin 2,3,4 devono accendersi in sequenza.

```
Led leds[3] = {Led(2), Led(3), Led(4)};

void loop() {
  for(int i=0; i<3; i++){
    leds[i].on(); delay(500); leds[i].off();
  }
}
```

```
Led led1(2);
Led led2(3);
Led led3(4);

void loop() {
  led1.on(); delay(500); led1.off();
  led2.on(); delay(500); led2.off();
  led3.on(); delay(500); led3.off();
}
```

Esercizio 6 – PWM: luminosità variabile

Un LED collegato a un pin PWM deve variare luminosità da 0 a 255 e ritorno.

```
Led led1(5);

void loop() {
  for(int i=0; i<=255; i+=5){ led1.setBrightness(i); delay
      (50); }
  for(int i=255; i>=0; i-=5){ led1.setBrightness(i); delay
      (50); }
}
```

Esercizio 7 – Semaforo

Tre LED collegati ai pin 2,3,4 devono simulare un semaforo.

```
Led rosso(2);
Led giallo(3);
Led verde(4);

void loop() {
  rosso.on(); giallo.off(); verde.off();
  delay(3000);
  rosso.off(); giallo.off(); verde.on();
  delay(3000);
  rosso.off(); giallo.on(); verde.off();
  delay(1000);
}
```

Esercizio 8 – Effetto rincorsa

Quattro LED collegati ai pin 2–5 devono accendersi uno alla volta.

```
// Con array
Led leds[4] = {Led(2), Led(3), Led(4), Led(5)};

void loop() {
  for(int i=0; i<4; i++){
    if(i>0) leds[i-1].off();
    leds[i].on(); delay(200);
  }
  leds[3].off();
}
```

```
// Senza array
Led led1(2);
Led led2(3);
Led led3(4);
Led led4(5);

void loop() {
  led1.on(); delay(200); led1.off();
  led2.on(); delay(200); led2.off();
  led3.on(); delay(200); led3.off();
  led4.on(); delay(200); led4.off();
}
```

Esercizio 9 – Lampeggio alternato su quattro LED

Due coppie di LED lampeggiano alternativamente ogni 500 ms.

```
// Con array
Led coppia1[2] = {Led(2), Led(3)};
Led coppia2[2] = {Led(4), Led(5)};

void loop() {
  for(int i=0; i<2; i++){ coppia1[i].on(); coppia2[i].off();
     }
  delay(500);
  for(int i=0; i<2; i++){ coppia1[i].off(); coppia2[i].on();
     }
  delay(500);
}
```

```
    // Senza array
Led led1(2);
Led led2(3);
Led led3(4);
Led led4(5);

void loop() {
  led1.on(); led2.on(); led3.off(); led4.off(); delay(500);
  led1.off(); led2.off(); led3.on(); led4.on(); delay(500);
}
```

Esercizio 10 – PWM opposto su due LED

Due LED collegati ai pin 2 e 3 variano luminosità in opposizione.

```
Led led1(2);
Led led2(3);

void loop() {
  for(int i=0; i<=255; i+=5){
    led1.setBrightness(i);
    led2.setBrightness(255-i);
    delay(50);
  }
}
```

5 Esercizi 11–15: simulazioni di dispositivi reali

Esercizio 11 – Ventola

Simulazione di una ventola: la ventola ha due stati, bassa e alta. L'output è simulato accendendo un LED.

```
class Ventola {
  int pinLED; // LED che simula la ventola
public:
  Ventola(int pin) {
    pinLED = pin;
    pinMode(pinLED, OUTPUT);
    }

  void bassa() { // ventola a bassa velocita
    digitalWrite(pinLED, HIGH);
    }
  void alta() {  // ventola ad alta velocita
    digitalWrite(pinLED, HIGH); delay(100);
    digitalWrite(pinLED, LOW); delay(100);
  }
};

Ventola ventola(2);

void loop() {
  ventola.bassa();
  delay(1000);
  for(int i=0; i<5; i++){
    ventola.alta();
    }
}
```

Esercizio 12 – Motore DC

Simulazione di un motore DC: avanti, indietro e fermo. L'output è simulato con due LED.

```
class Motore {
  int pinLEDAvanti;
  int pinLEDIndietro;
public:
  Motore(int pinA, int pinI) {
    pinLEDAvanti = pinA; pinMode(pinLEDAvanti, OUTPUT);
    pinLEDIndietro = pinI; pinMode(pinLEDIndietro, OUTPUT);
  }

  void avanti() { digitalWrite(pinLEDAvanti, HIGH);
     digitalWrite(pinLEDIndietro, LOW); }
  void indietro() { digitalWrite(pinLEDAvanti, LOW);
     digitalWrite(pinLEDIndietro, HIGH); }
  void fermo() { digitalWrite(pinLEDAvanti, LOW);
     digitalWrite(pinLEDIndietro, LOW); }
};

Motore motore(2,3);

void loop() {
  motore.avanti(); delay(1000);
  motore.indietro(); delay(1000);
  motore.fermo(); delay(1000);
}
```

Esercizio 13 – Valvola automatica simulata

Simulazione di una valvola: aperta o chiusa. LED simula lo stato.

```
class Valvola {
  int pinLED;
public:
  Valvola(int pin) { pinLED = pin; pinMode(pinLED, OUTPUT); }
  void apri() { digitalWrite(pinLED, HIGH); }
  void chiudi() { digitalWrite(pinLED, LOW); }
};

Valvola valvola(2);

void loop() {
  valvola.apri(); delay(2000);
  valvola.chiudi(); delay(2000);
}
```

Esercizio 14 – Cancello automatico simulato

```
class Cancello {
  int pinApri;
  int pinChiudi;
public:
  Cancello(int pinA, int pinC) {
    pinApri = pinA; pinMode(pinApri, OUTPUT);
    pinChiudi = pinC; pinMode(pinChiudi, OUTPUT);
  }

  void apri() { digitalWrite(pinApri, HIGH); digitalWrite(
     pinChiudi, LOW); }
  void chiudi() { digitalWrite(pinApri, LOW); digitalWrite(
     pinChiudi, HIGH); }
};

Cancello cancello(2,3);
void loop() {
  cancello.apri(); delay(3000);
  cancello.chiudi(); delay(3000);
}
```

Esercizio 15 – Motore DC simulato con ponte H L9110S

```
class MotoreH {
  int pinAvanti;
  int pinIndietro;
public:
  MotoreH(int pinA, int pinI) {
    pinAvanti = pinA; pinMode(pinAvanti, OUTPUT);
    pinIndietro = pinI; pinMode(pinIndietro, OUTPUT);
  }
  void avanti() { digitalWrite(pinAvanti, HIGH); digitalWrite
     (pinIndietro, LOW); }
  void indietro() { digitalWrite(pinAvanti, LOW);
     digitalWrite(pinIndietro, HIGH); }
  void fermo() { digitalWrite(pinAvanti, LOW); digitalWrite(
     pinIndietro, LOW); }
};

MotoreH motoreH(2,3);
void loop() {
  motoreH.avanti(); delay(1000);
  motoreH.indietro(); delay(1000);
  motoreH.fermo(); delay(1000);
}
```