Azzolini Riccardo 2020-10-22

Arduino — Introduzione all'uso e output digitali

1 Come si usa Arduino

Esistono moltissime diverse schede Arduino (originali, cloni e altre schede compatibili), che si differenziano per il tipo (e la potenza) del microcontrollore, il numero e le funzionalità delle porte di I/O, ecc. Tutte queste schede vengono programmate in un linguaggio simile al C, usando un semplice IDE (ambiente di sviluppo) che è open source, gratuito e multipiattaforma.

L'IDE di Arduino permette di scrivere programmi (chiamati in gergo "sketch"), compilarli, e caricarli su una scheda tramite USB.

Per funzionare, una scheda Arduino ha bisogno di un'alimentazione, che può essere fornita dallo stesso collegamento USB usato per programmarla, oppure da un comune alimentatore a 7–12 V, che si collega alla scheda mediante un apposito spinotto. Per evitare di mettere a rischio la porta USB del proprio computer (nel caso ci fossero, ad esempio, errori di progettazione o scariche di tensione inaspettate nel circuito a cui la scheda è collegata), è consigliabile usare sempre un alimentatore, anche mentre Arduino è collegato al PC per la programmazione.[1]

2 Output digitali

Ogni scheda Arduino, ha una serie di terminali, chiamati *piedini* (in inglese *pin*),[2] che permettono di collegare la scheda a circuiti elettronici esterni. La maggior parte di questi sono porte di I/O digitali, ciascuna identificata da un numero. Ad esempio, sul tipo di scheda più comune (Arduino UNO, e i suoi tanti cloni), le porte di I/O digitali sono 14, numerate da 0 a 13, e alla porta 13 è già collegato un LED "integrato", che si accende quando questa porta (in modalità di output) viene messa allo stato logico high. Allora, il

[1] Su una scheda Arduino sono presenti dei transistor che selezionano la sorgente di alimentazione; quando sono disponibili sia l'USB che un alimentatore, viene data la priorità all'alimentatore, e l'USB viene impiegato solo per i dati.

[2] Più in generale, si chiamano piedini i terminali di un qualunque circuito integrato, come ad esempio un microcontrollore. I piedini di una scheda Arduino sono appunto collegati ai piedini del microcontrollore montato sulla scheda.

più semplice esempio di output digitale con Arduino consiste nel far lampeggiare questo LED:

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

Questo programma accende il LED e lo tiene acceso per un secondo, poi lo spegne e lo tiene spendo per un secondo, e così via, all'infinito (finché la scheda Arduino rimane collegata all'alimentazione).

Come mostrato da questo esempio, un programma Arduino è composto, in generale, da almeno due funzioni: `setup()`, che viene eseguita una volta in fase di inizializzazione del microcontrollore, e `loop()`, eseguita ripetutamente fintanto che il microcontrollore è acceso.

Esistono poi vari comandi che possono essere usati in queste funzioni (l'elenco completo è disponibile sul sito di Arduino: https://www.arduino.cc/reference). Nel presente esempio sono impiegati alcuni dei più importanti:

- `pinMode` imposta la modalità di una porta di I/O, che può essere output, input, o input con pull-up interno. Questo viene solitamente usato nella funzione `setup()` (e si consiglia di metterlo all'inizio di tale funzione, per evitare di dimenticarselo).
- `digitalWrite` imposta il livello logico di una porta di I/O che è stata precedentemente messa in modalità di output.
- `delay` mette il programma in pausa per il numero di millisecondi specificato come argomento.

Nel codice riportato sopra, il primo argomento di `pinMode` e `digitalWrite`, che specifica il numero della porta a cui si applicano tali comandi, è la costante `LED_BUILTIN`, che identifica la porta a cui è collegato il LED integrato; sulla scheda Arduino UNO e i suoi cloni, questa costante è equivalente al valore numerico `13`.

2.1 Collegamento di un LED esterno

Invece di far lampeggiare il LED integrato sulla scheda, si può scegliere di collegare un LED esterno, che può essere messo:

- in *current sourcing*, collegandone l'anodo a una porta di I/O e il catodo al piedino etichettato GND (**ground**, **massa**, che in pratica significa 0 V);
- in *current sinking*, collegandone il catodo a una porta di I/O e l'anodo a un piedino di alimentazione a 3.3 V o 5 V.

Come al solito, serve anche una resistenza in serie al LED, che deve essere calcolata in base alla tensione di alimentazione (3.3 V o 5 V) e alla V_F del LED.

Supponendo di collegare il LED alla porta di I/O numero 8, il programma per farlo lampeggiare sarebbe:

```
void setup() {
    pinMode(8, OUTPUT);
}

void loop() {
    digitalWrite(8, HIGH);
    delay(1000);
    digitalWrite(8, LOW);
    delay(1000);
}
```

(rispetto al programma di prima, è cambiata solo la porta usata). Una cosa da ricordare è che, se si scegliesse il collegamento in sinking, il funzionamento del LED risulterebbe inverso: esso si accenderebbe con il comando `digitalWrite(8, LOW)` e si spegnerebbe con `digitalWrite(8, HIGH)`.

Come altro esempio, il seguente programma fa lampeggiare in modo alternato il LED integrato e un LED esterno (collegato in sourcing alla porta 8), tenendo acceso quello integrato per due secondi e poi quello esterno per mezzo secondo:

```
void setup() {
    pinMode(8, OUTPUT);
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    digitalWrite(8, LOW);
    delay(2000);
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(8, HIGH);
    delay(500);
}
```