I trucchi di Arduino

Semplice multitasking

di Davide Achilli IZ2UUF

Arduino è entrato prepotentemente nel mondo dell'elettronica amatoriale e certo non ha bisogno di presentazioni. La combinazione di schedine a basso costo già predisposte con tutti i componenti di contorno e di un ambiente di sviluppo dall'uso elementare hanno reso la curva di apprendimento molto corta per tutti. Persone che non avevano mai scritto una riga di codice in vita loro si sono trovate in pochi minuti, con enorme soddisfazione, a vedere il piccolo microcontrollore seguire docile le loro istruzioni. Complice una quantità sterminata di librerie pronte all'uso, con pochi semplici righe di codice chiunque può domare sensori, display, attuatori e qualunque altro hardware possibile ed immaginabile. La scelta del linguaggio C/C++, potente e molto diffuso ma certo non facile da usare per un principiante, si è invece rivelata vincente: la minuscola memoria dei microcontrollori impiegati da Arduino, infatti, non permette di raggiungere un grado di complessità tale da creare difficoltà. Tant'è che, modificando gli esempi e andando ad intuito, tutti arrivano facilmente ad un risultato.

La trappola di Arduino

Purtroppo non sono tutte rose e fiori per il principiante. Dopo aver letto un sensore e gestito un pulsante con successo, il passo logico successivo è leggere il sensore e gestire il pulsante nella stessa applicazione. Questo dovrebbe essere il punto di partenza per un'irrefrenabile *escalation* di funzionalità, ma spesso si arena subito in un comportamento anomalo e incomprensibile del sistema. Pulsanti che funzionano quando vogliono, LED che non si accendono, sensori che riportano sempre lo stesso valore e via dicendo. E all'iniziale euforia, subentra la frustrazione.

Per comprendere una delle più frequenti cause di questi problemi esaminiamo il programma in figura 1, preso dal sito ufficiale di Arduino, dove viene mostrato come creare un LED lampeggiante con un periodo di due secondi.

I programmi "standard" per Arduino sono composti da due funzioni: una *setup*, che viene chiamata una sola volta all'inizio e utilizzata per inizializzare il microcontrollore, e una *loop*, che viene richiamata all'infinito.

Nel nostro esempio, dopo aver configurato come *OUTPUT* la porta a cui è collegato il LED di serie (figura 1, linea 2), il programma ha uno svolgimento molto intuitivo:

- alla linea 6 il pin collegato al LED viene messo *HIGH* e quindi il LED si accende;
- alla linea 7 si attendono 1000ms (=1 secondo) durante i quali il LED rimane acceso
- alla linea 8 si mette *LOW* il pin e il LED si spegne
- alla linea 9 si attendono altri 1000ms durante i quali il LED rimane spento

Essendo la funzione "loop" ripetuta all'infinito, avremo un'esecuzione continua di quella sequenza e di conseguenza un LED che lampeggia con un periodo di due secondi.

Procediamo ora ad aggiungere un semplice requisito: un pulsante che, non appena premuto, spenga immediatamente il LED e disabiliti il lampeggio. Per collegare il pulsante useremo il pin 3 impostato ad *INPUT_PULLUP* in modo che per dare l'impulso sia sufficiente chiudere il contatto a massa.

L'approccio in genere seguito dai meno esperti per aggiungere questa funzionalità è illustrato in figura 2. L'esempio estende quello precedente con poche aggiunte:

- alla riga 1 una variabile globale *attivo*, inizialmente impostata a *true*, ricorda se il lampeggio al momento è attivo o meno
- alla riga 11 viene letto il pin n.3, che normalmente è *HIGH* (tasto non premuto); se risulta essere *LOW* è perché qualcuno sta premendo il pulsante e quindi la variabile *attivo* viene messa a *false*

```
1  void setup() {
2    pinMode(LED_BUILTIN, OUTPUT);
3  }
4
5  void loop() {
6    digitalWrite(LED_BUILTIN, HIGH);
7    delay(1000);
8    digitalWrite(LED_BUILTIN, LOW);
9    delay(1000);
10 }
```

Fig. 1 - Esempio introduttivo presente sul sito di Arduino in cui viene mostrato come creare un LED lampeggiante

```
1  bool attivo = true;
2
3  void setup()
4  {
5    pinMode(LED_BUILTIN, OUTPUT);
6    pinMode(3, INPUT_PULLUP);
7  }
8
9  void loop()
10 {
11   if (digitalRead(3) == LOW) attivo = false;
12   if (attivo) {
13     digitalWrite(LED_BUILTIN, HIGH);
14     delay(1000);
15     digitalWrite(LED_BUILTIN, LOW);
16     delay(1000);
17   }
18 }
```

Fig. 2 - LED lampeggiante disabilitato dalla pressione di un tasto