Azzolini Riccardo 2020-10-27

Arduino — Input digitali

1 Lettura di un input digitale

Il comando `digitalRead` di Arduino legge e restituisce lo stato logico (`LOW` o `HIGH`) della porta di I/O digitale passata come argomento (che deve essere in modalità di input).

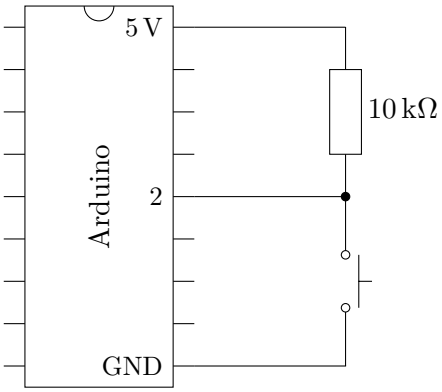
Ad esempio, se si collega un pulsante alla porta numero 2, il seguente sketch fa accendere e spegnere il LED integrato di Arduino a seconda dello stato del pulsante:

```
const int LED = LED_BUILTIN;
const int BUTTON = 2;

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(BUTTON, INPUT);
}

void loop() {
    int buttonState = digitalRead(BUTTON);
    digitalWrite(LED, buttonState);
}
```

Così, se il pulsante è collegato in pull-up,



il LED integrato (che è messo in current sourcing) risulterà acceso quando il pulsante non è premuto, e spento quando il pulsante è premuto. Per ottenere invece la logica

"corretta", cioè far accendere il LED quando il pulsante è premuto, bisogna negare lo stato logico del pulsante prima di applicarlo al LED:

```
const int LED = LED_BUILTIN;
const int BUTTON = 2;

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(BUTTON, INPUT);
}

void loop() {
    int buttonState = digitalRead(BUTTON);
    digitalWrite(LED, !buttonState);
}
```

2 Debouncing

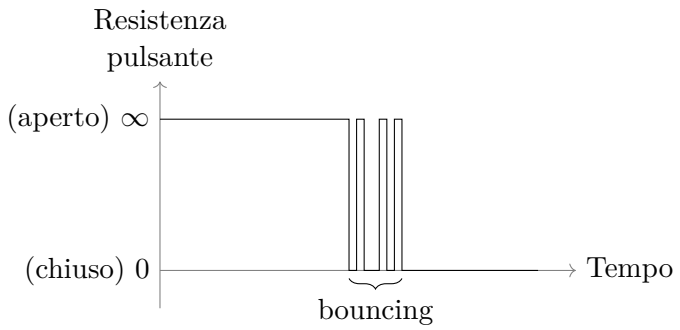
Con il codice scritto finora, il LED rimane acceso solo finché si tiene premuto il pulsante, poi si spegne. Adesso, invece, si vuole far sì che il LED si accenda con una pressione del pulsante e si spenga alla pressione successiva (l'azione che in inglese viene chiamata *toggle*). Un modo per ottenere questo effetto è memorizzare gli stati del pulsante e del LED, e invertire lo stato del LED ogni volta che il pulsante passa da "non premuto" a "premuto" (da `HIGH` a `LOW`, se è collegato in pull-up):

```
const int LED = LED_BUILTIN;
const int BUTTON = 2;
int ledState = LOW;
int lastButtonState = HIGH;

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(BUTTON, INPUT);
}

void loop() {
    int buttonState = digitalRead(BUTTON);
    if (buttonState == LOW && lastButtonState == HIGH) {
        ledState = !ledState;
        digitalWrite(LED, ledState);
    }
    lastButtonState = buttonState;
}
```

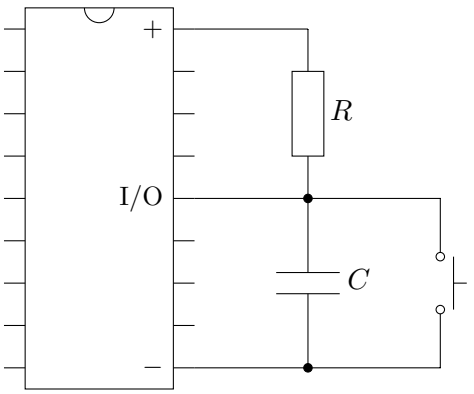
Se però si provasse questo programma, sembrerebbe che non funzioni sempre correttamente: a volte, premendo il pulsante, lo stato del LED cambierebbe, mentre altre volte rimarrebbe invariato. La causa di questo comportamento strano non è un errore nel software, bensì un comportamento fisico del pulsante, chiamato **bouncing**: quando esso viene premuto (e quando viene rilasciato), può succedere che i contatti al suo interno "rimbalzino", aprendo e chiudendo il circuito più volte in un breve periodo di tempo (tipicamente nell'ordine di pochi millisecondi, o anche meno), prima di stabilizzarsi definitivamente nello stato chiuso (o aperto, se il pulsante è stato rilasciato).



Ciascuna chiusura del circuito viene rilevata dal microcontrollore come una nuova pressione del pulsante, quindi lo stato del LED cambia ripetutamente (e se in totale cambia un numero pari di volte, lo stato finale rimane uguale a quello iniziale).

Per risolvere questo problema, esistono diverse tecniche di **debouncing**, sia di tipo hardware che software. Le più comuni si basano sull'introduzione di un breve ritardo, durante il quale lo stato del pulsante deve rimanere costante perché la pressione venga rilevata.

- Nell'hardware, ciò può essere fatto mediante una rete RC, come ad esempio la seguente:

  Con questa configurazione, inoltre, il pulsante risulta temporaneamente "premuto" quando il circuito viene inizialmente alimentato, finché il condensatore non si

carica: ciò può essere utile per un pulsante di reset di un microcontrollore, in modo da ritardare l'avvio di quest'ultimo finché il resto del circuito non è "pronto".

- Nel software, il ritardo può essere realizzato mediante opportuni contatori e/o timer.

Allora, per implementare correttamente la funzione di toggle del LED integrato (sempre con un pulsante collegato in pull-up alla porta 2), si può utilizzare ad esempio il seguente codice:[1]

```
const int LED = LED_BUILTIN;
const int BUTTON = 2;
const unsigned long DEBOUNCE_DELAY = 50; // milliseconds

int ledState = LOW;
int lastButtonState = HIGH;
int lastInput = HIGH;
unsigned long lastDebounceTime = 0;

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(BUTTON, INPUT);
}

void loop() {
    int input = digitalRead(BUTTON);
    if (input != lastInput) {
        // Reset the timer, because the input has changed.
        lastDebounceTime = millis();
    }
    lastInput = input;

    if (millis() - lastDebounceTime >= DEBOUNCE_DELAY) {
        // Accept the new input value, as it's been stable for a while.
        if (input == LOW && lastButtonState == HIGH) {
            // Toggle the LED, because the button has just been pressed.
            ledState = !ledState;
            digitalWrite(LED, ledState);
        }
        lastButtonState = input;
    }
}
```

[1] Questo codice è basato su uno degli esempi forniti con l'ambiente di sviluppo Arduino: https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce