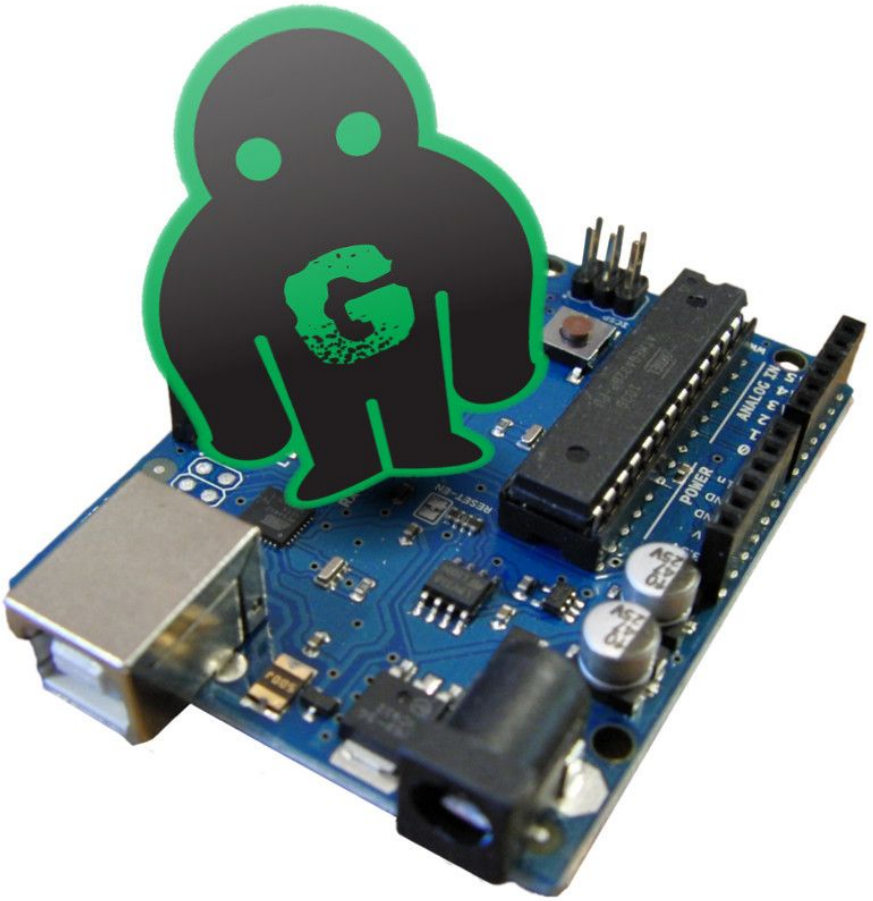
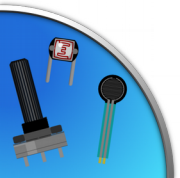
CORSO ARDUINO



Jacopo Belli
Giulio Fieramosca
Luca Mattii
GOLEM 2016

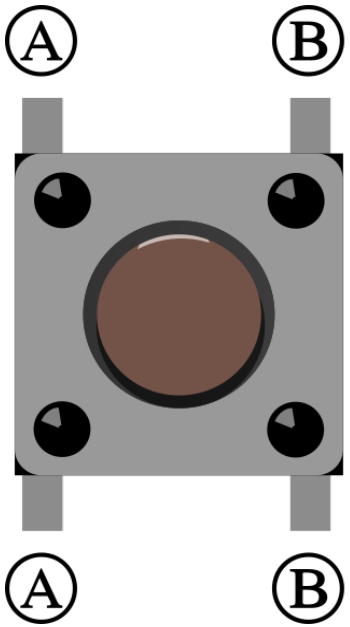


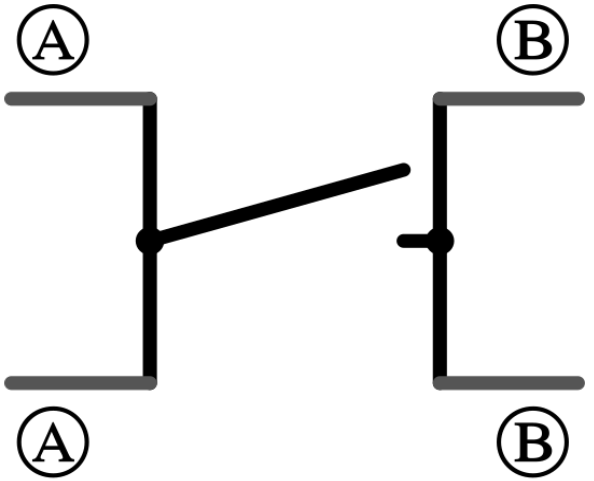
Pulsanti e interruttori

Pushbutton

Momentary button

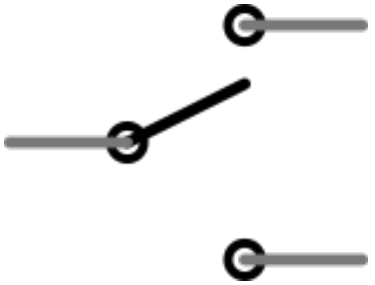
Pulsante monostabile

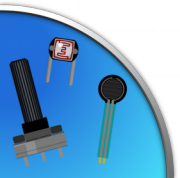




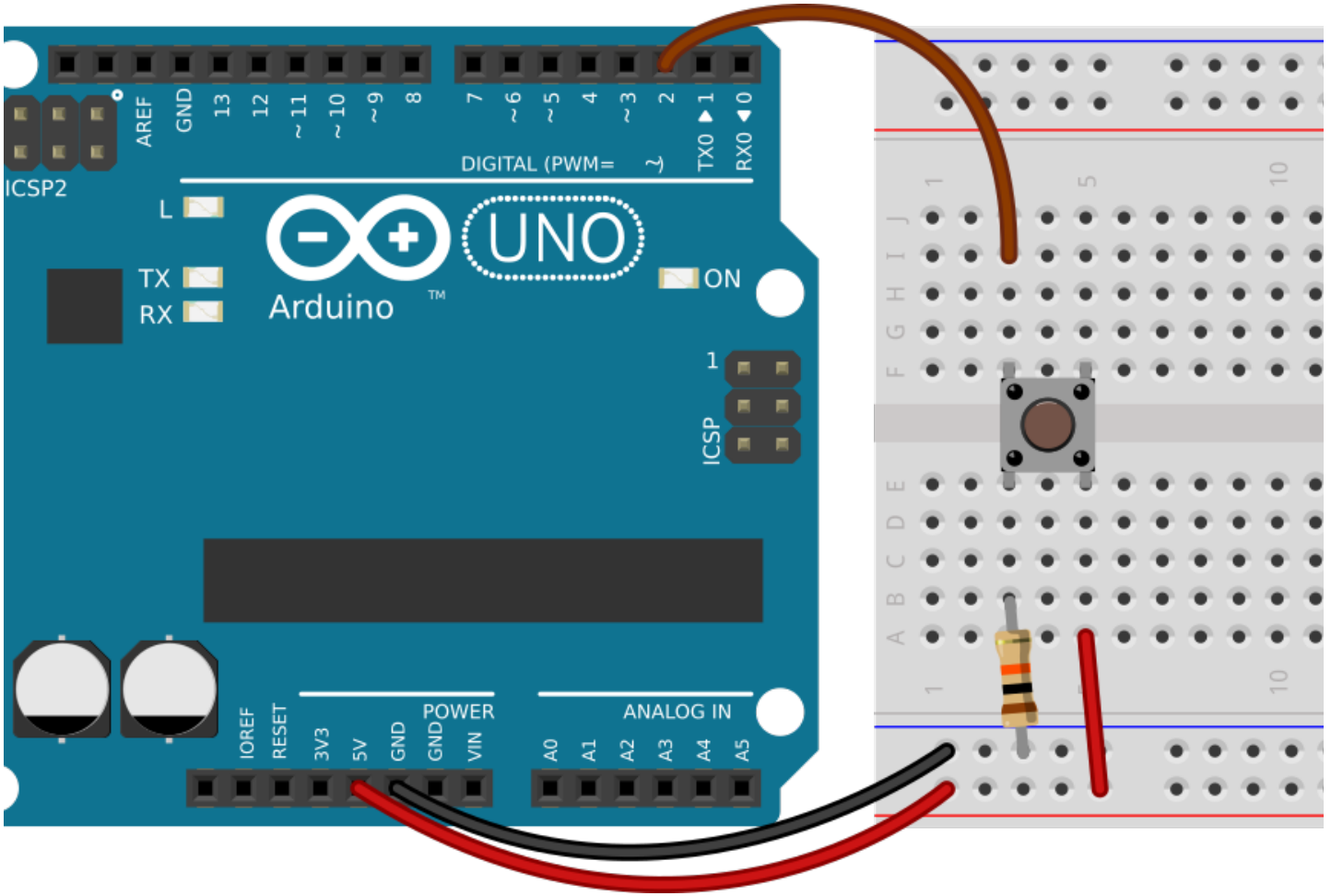
Switch

Interruttore bistabile

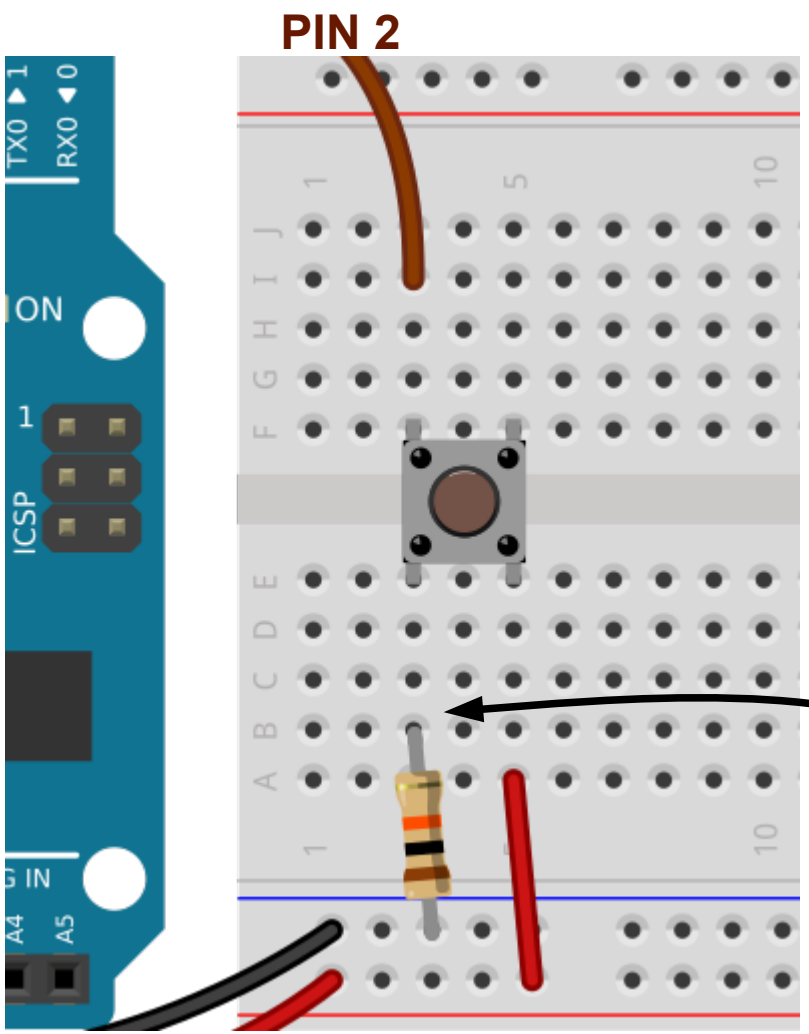


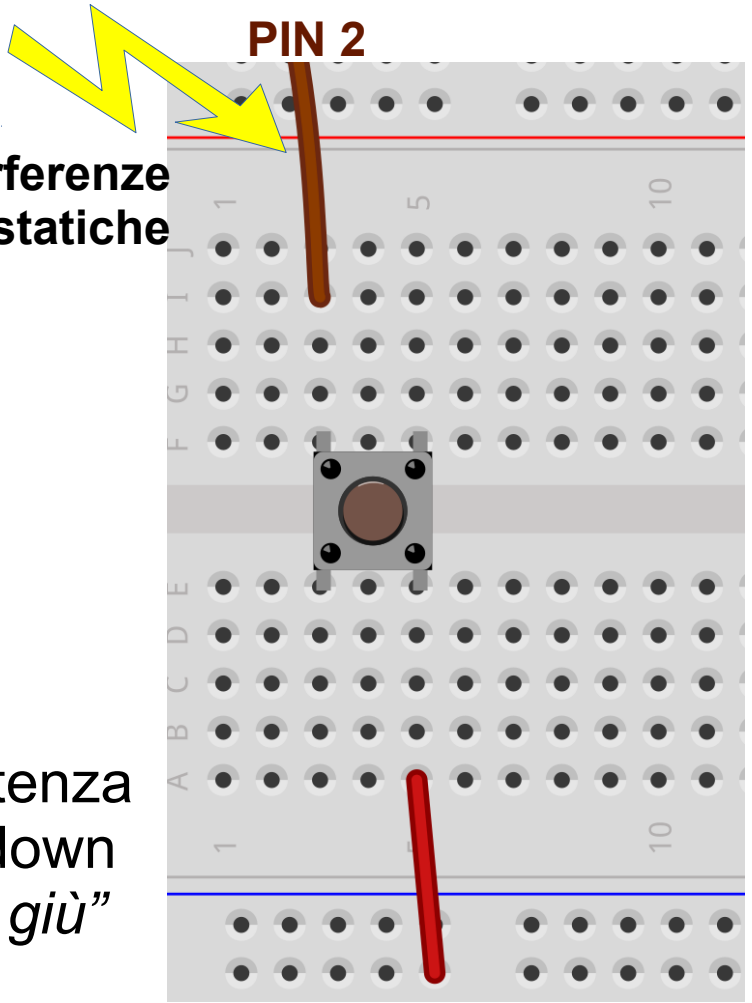


Il pulsante (connessioni)



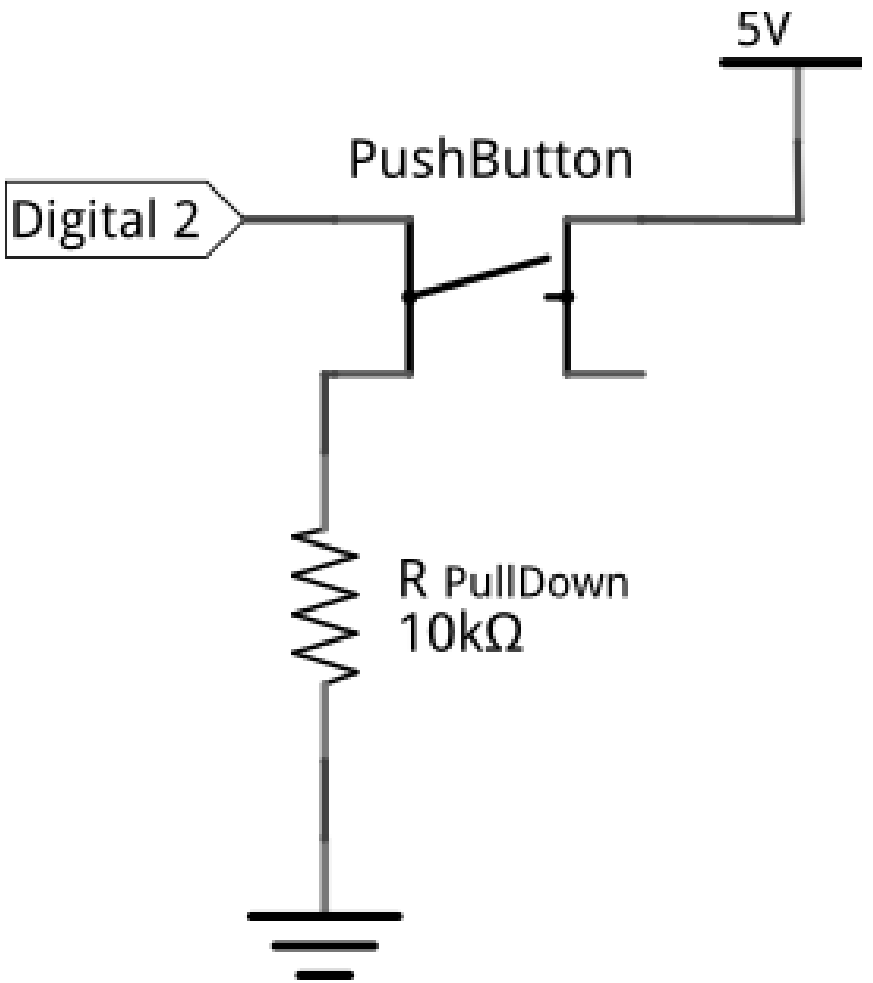
Pull-down e cortocircuiti...

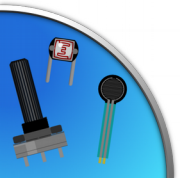




Resistenza
Pull-down
"Tira giù"

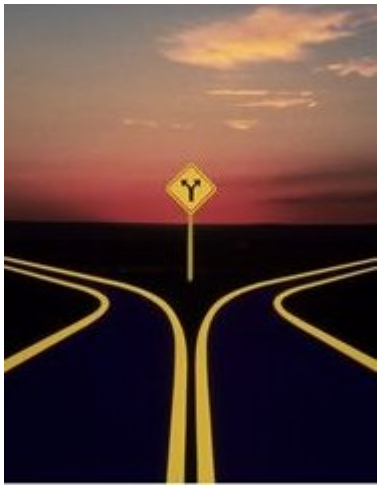
Il pulsante (schema)



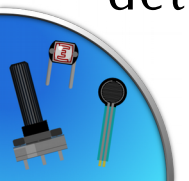


Blocco `if`: espansione

```
if (condizione #1) {
    // Blocco da eseguire #1
}
else if (condizione #2) {
    // Blocco da eseguire #2
}
else {
    // Blocco da eseguire altrimenti
}
```



Si possono aggiungere infinite **condizioni** da verificare, analoghe a più percorsi che il programma può seguire.
Può essere inserito un percorso *generico* da imboccare se nessuna delle condizioni è verificata (`else`).



Unire più condizioni

Facciamo conto di aver dichiarato due variabili a e b...

```
if ( ( a == 2 ) && ( b < 5 ) )
```

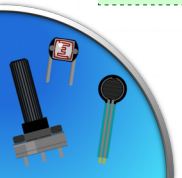
La condizione è verificata se **entrambe** le condizioni sono verificate (AND).

```
if ( ( a > 2 ) || ( b <= 3 ) )
```

La condizione è verificata se **almeno una** condizione è verificata (OR).

```
if ( !( a < 3) )
```

La condizione è verificata se la condizione **non** è verificata (NOT).

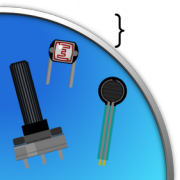


Il pulsante (listato)

```
const byte PIN_PULSANTE = 2;
const byte PIN_LED =  13;

void setup() {
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_PULSANTE, INPUT);
}

void loop(){
  boolean statoPulsante = digitalRead(PIN_PULSANTE);

  if (statoPulsante == HIGH) {          // se è HIGH
    digitalWrite(PIN_LED, HIGH);        // accende il led
  }
  else {
    digitalWrite(PIN_LED, LOW);         // altrimenti lo spegne
  }
  delay(10);
}
```

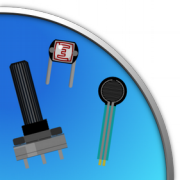
boolean → true **oppure** false

HIGH → true ← 1
LOW → false ← 0

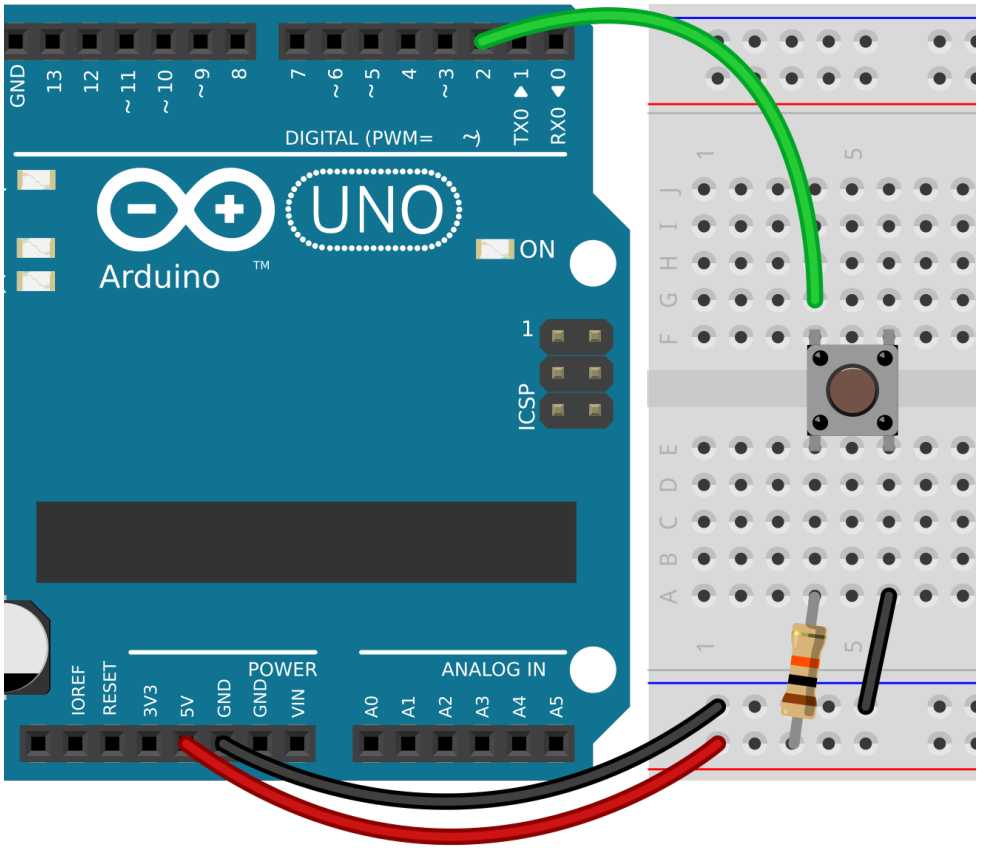


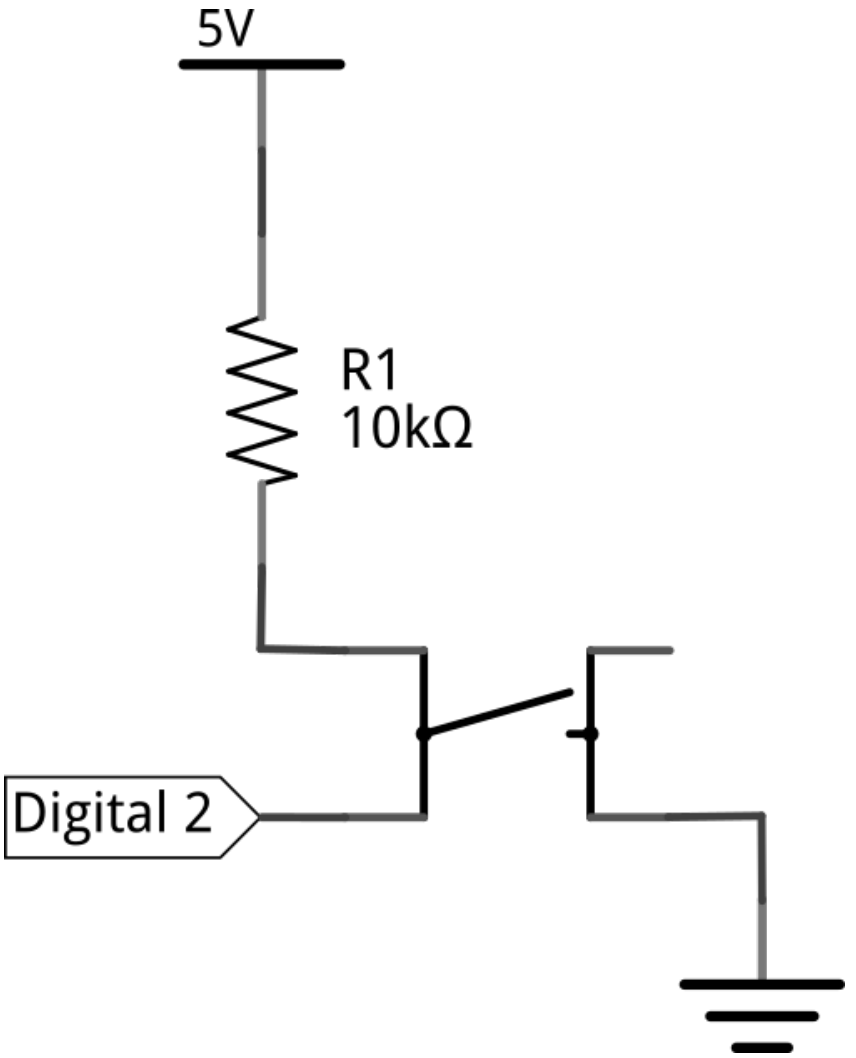
Il pulsante (Versione contratta)

```
const byte PIN_PULSANTE = 2;
const byte PIN_LED =  13;

void setup() {
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_PULSANTE, INPUT);
}

void loop(){
  digitalWrite(PIN_LED, digitalRead(PIN_PULSANTE));
  delay(10);
}
```

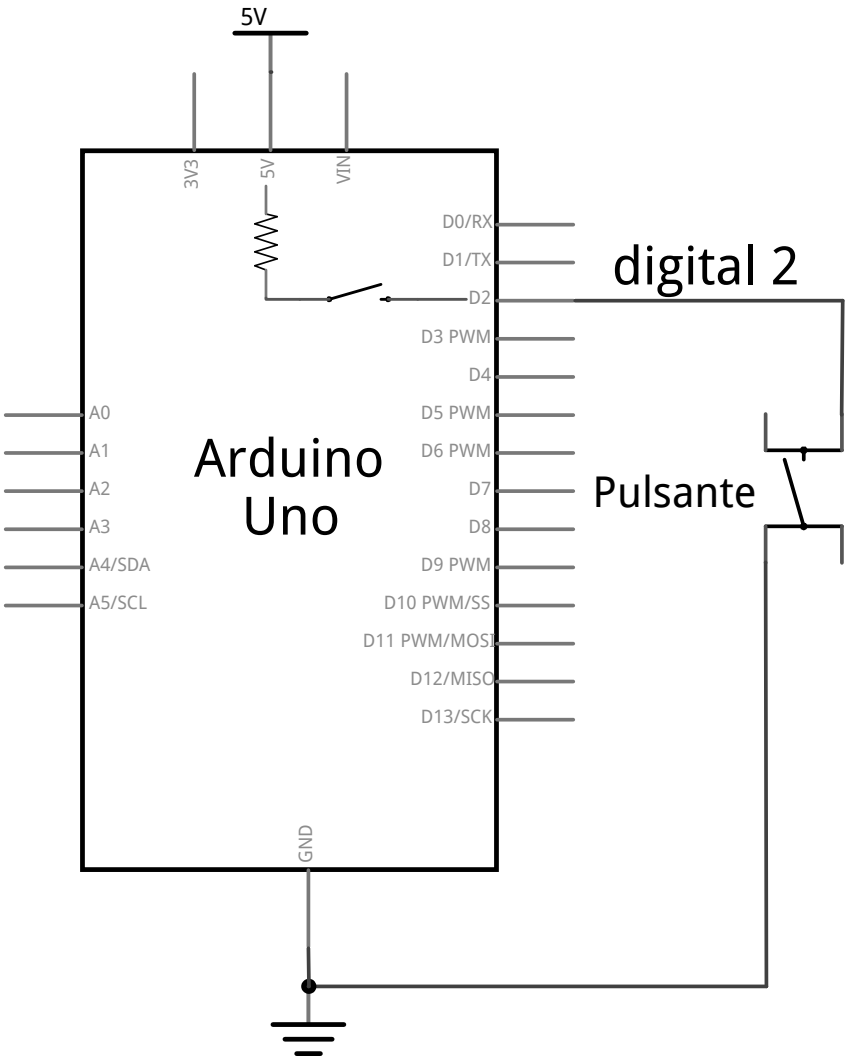


Connessione inversa (*pull-up*)

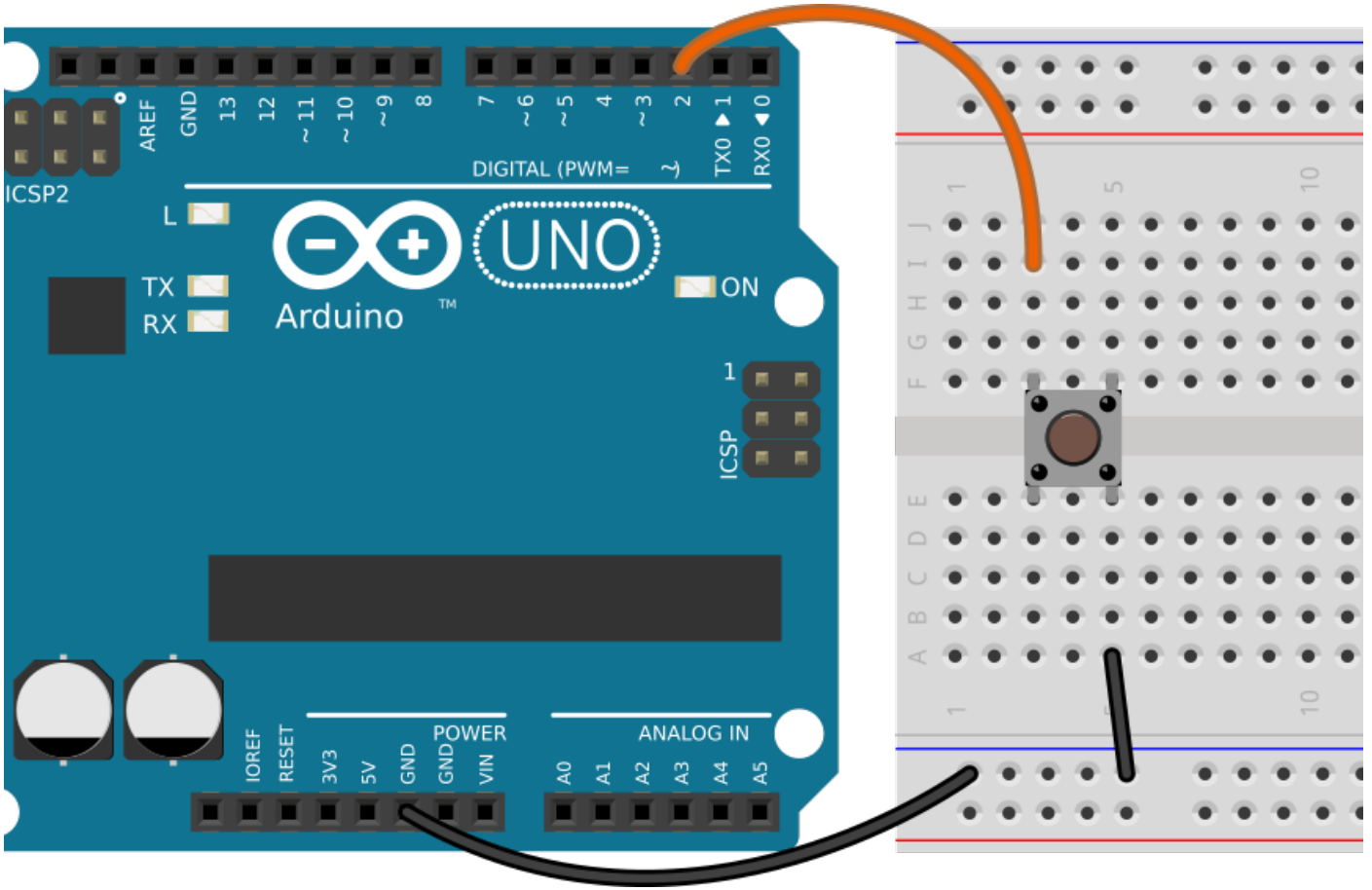




Pull-Up Interno

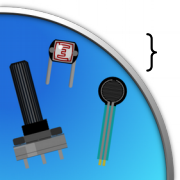


Pull-Up Interno

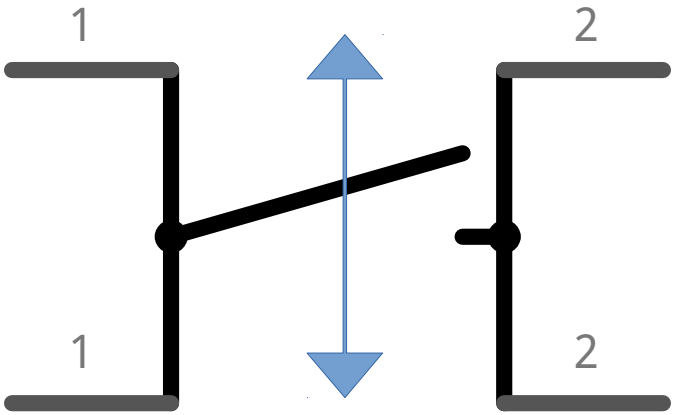


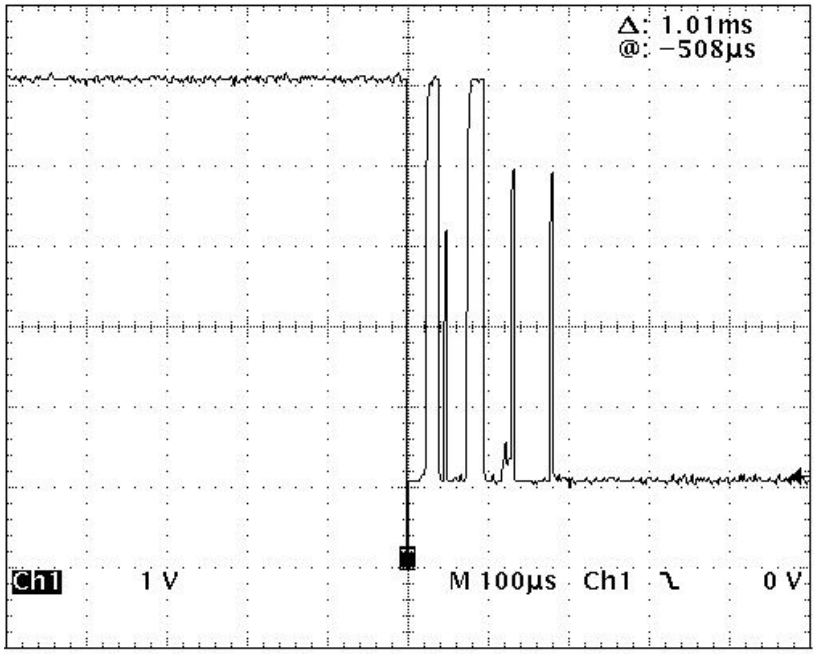
Pull-Up (listato)

```
const byte PIN_PULSANTE = 2;
const byte PIN_LED      =  13;

void setup() {
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_PULSANTE, INPUT_PULLUP);
}

void loop(){
  byte statoPulsante = digitalRead(PIN_PULSANTE);

  if (statoPulsante == LOW) {
    digitalWrite(PIN_LED, HIGH);
  }
  else {
    digitalWrite(PIN_LED, LOW);
  }
  delay(10);
}
```



Button Bouncing – letture spurie





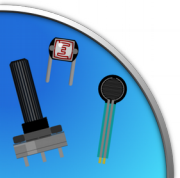

```
  else {
    digitalWrite(PIN_LED, LOW);          // altrimenti lo spegne
  }
  delay(10);
}
```

Segnali digitale e analogico

Sensori Analogici

Resistenze variabili, vanno combinati con altri componenti per leggere una variazione di tensione

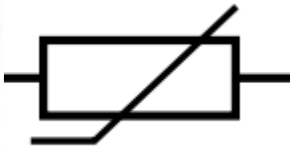
Sensori integrati, già pronti, che forniscono in output una tensione variabile, di solito compresa fra lo 0 e i 5v



Resistenze variabili

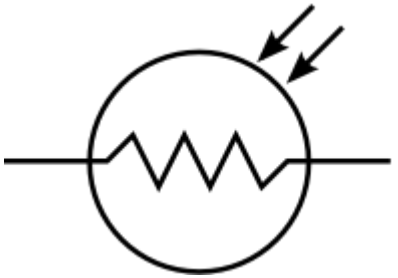
Potenziometro:

Si varia la resistenza ruotando una manopola



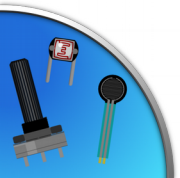
Termoresistenza (PTC o NTC):

Varia la resistenza con la temperatura



Fotoresistenza:

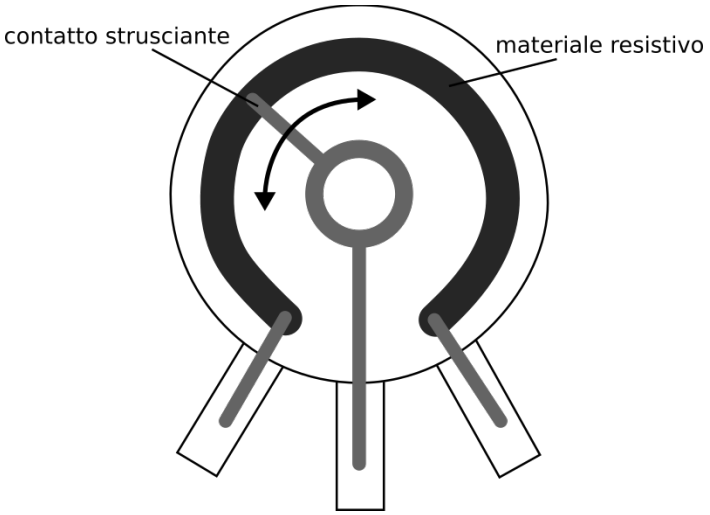
Varia la resistenza con la luminosità



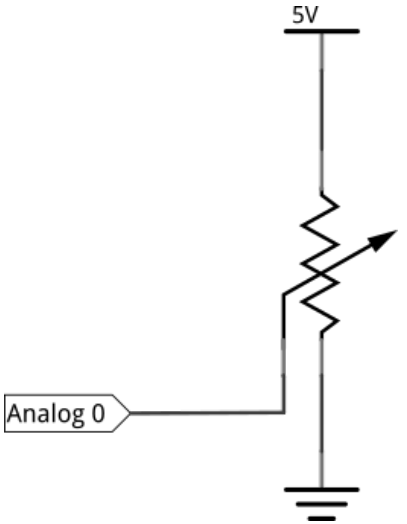
Potenziometro (e trimmer)

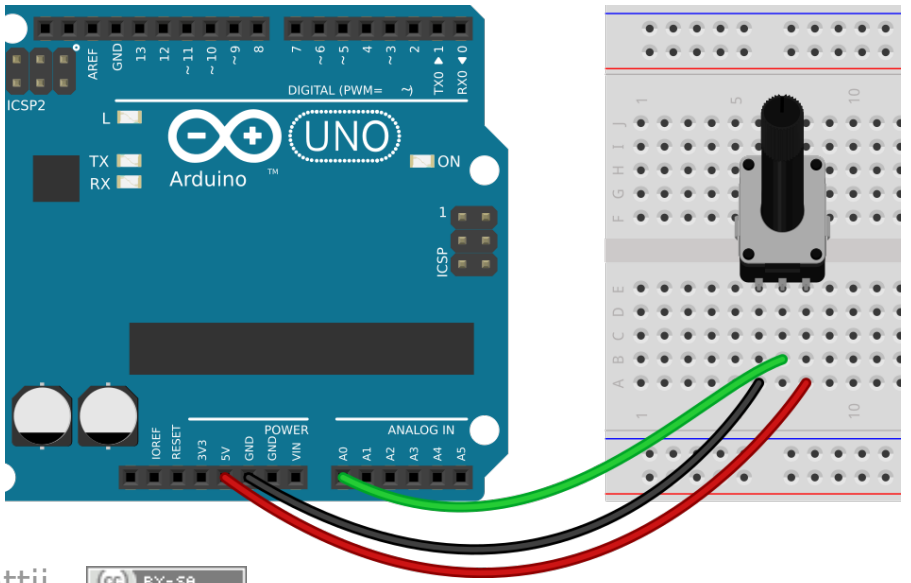
Potenziometro

Trimmer

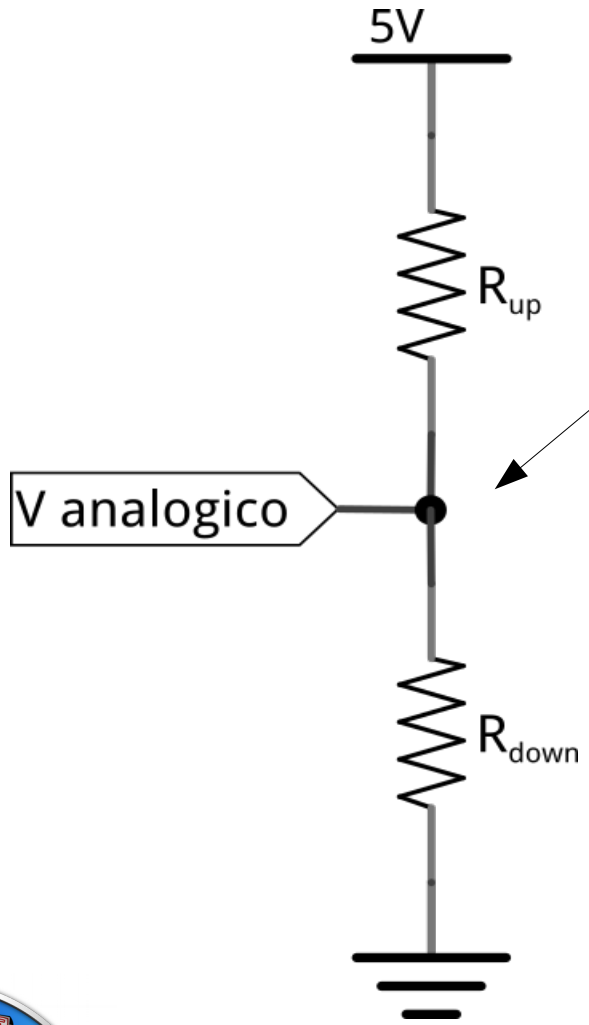


Funzionamento di un potenziometro





Il partitore di tensione



Formule utili:

$$V_{analogico} = 5\,V \cdot \frac{R_{down}}{R_{up} + R_{down}}$$

$$R_{down} = R_{up} \cdot \left(\frac{5\,V}{V_{analogico}} - 1 \right)$$

Nel potenziometro:

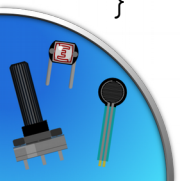
$$R_{up} + R_{down} = 10k$$

Potenziometro (codice)

```
const byte PIN_POTENZIOMETRO = A0;
void setup() {
  pinMode(PIN_POTENZIOMETRO, INPUT);
  Serial.begin(9600); // avvia la comunicazione seriale
}

void loop() {
  int lettura = analogRead(PIN_POTENZIOMETRO);
  // convertiamo la lettura in un valore di tensione
  float tensione = lettura * 5.0 / 1024.0;
  // inviamo la lettura ed il valore convertito al PC
  Serial.print("Lettura: ");
  Serial.print(lettura);
  Serial.print("/1023   ");
  Serial.print("Tensione: ");
  Serial.print(tensione);
  Serial.println("/5V");
  delay(1000);
}
```

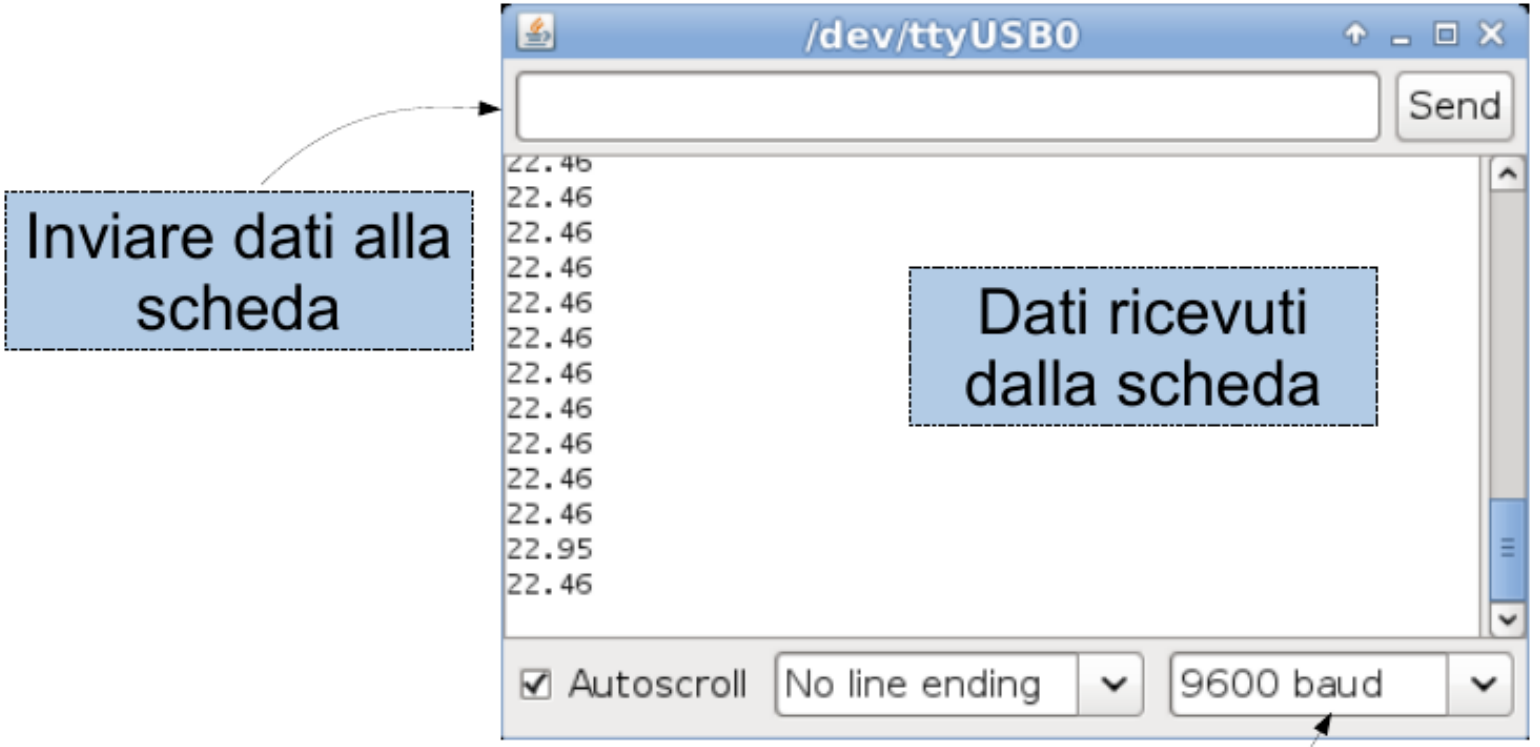
Legge Lettura ↔ Tensione

$$Tensione = 5\,V \cdot \frac{lettura}{1024}$$

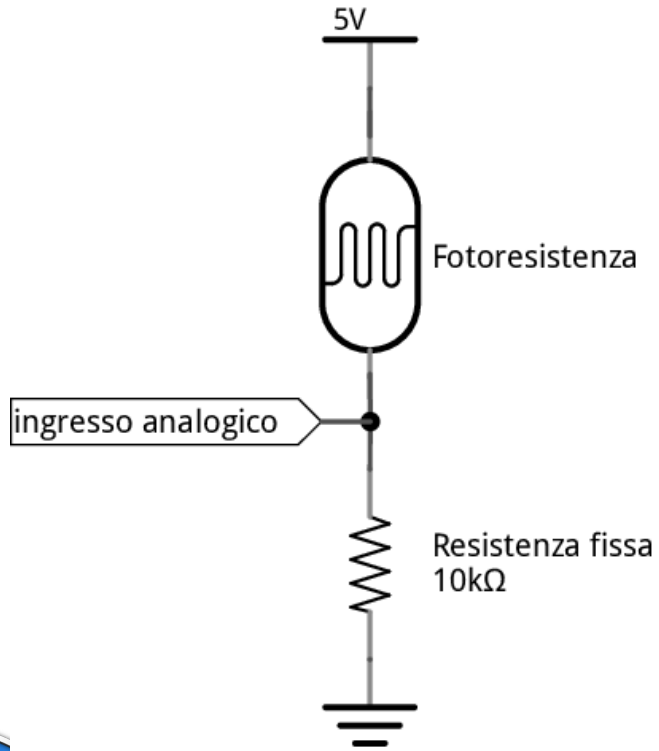


Monitor Seriale

Bottone per aprire il serial monitor

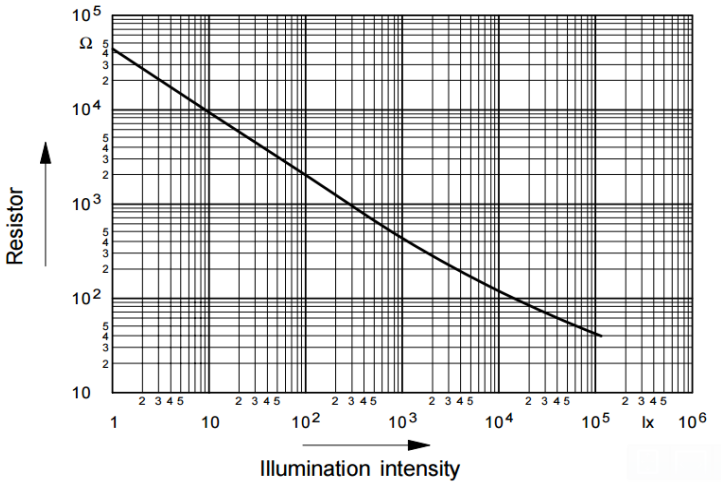


Fotoresistenza: lettura con **partitore di tensione**



→

Grafico del comportamento di una fotoresistenza

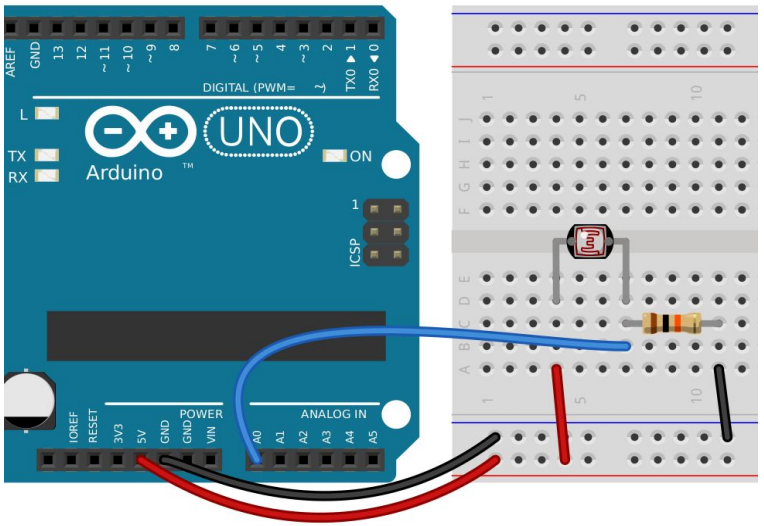


←

Partitore di tensione

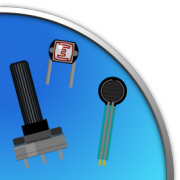
→

Connessioni su BreadBoard



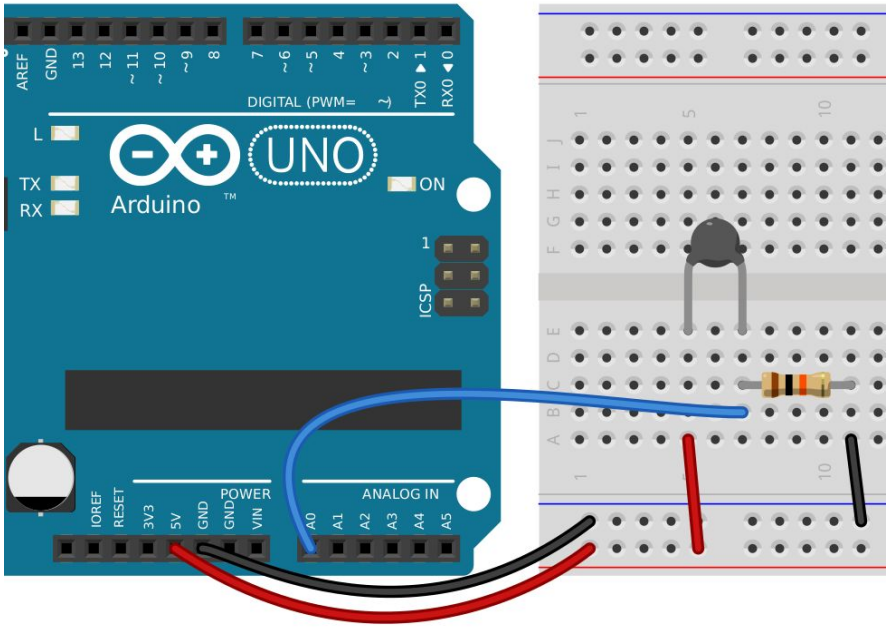
Fotoresistenza (codice)

```
const byte PIN_FOTORESISTENZA = A0;
void setup() {
  pinMode(PIN_FOTORESISTENZA, INPUT);
  Serial.begin(9600);
}

void loop() {
  int lettura = analogRead(PIN_FOTORESISTENZA);
  float tensione = lettura * 5.0 / 1024.0;
  Serial.print("Lettura: ");
  Serial.print(lettura);
  Serial.print("/1023   ");
  Serial.print("Tensione: ");
  Serial.print(tensione);
  Serial.println("/5V");
  delay(1000);
}
```



Termoresistenza:

lettura con **partitore di tensione**



$$R = 10\,k \cdot \left(\frac{1024}{\text{Valore Letto}} - 1 \right)$$

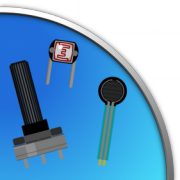
$$\frac{1}{T^{(K)}} = \frac{\log\left(\frac{R}{R_{nom}}\right)}{B_{coeff}} + \frac{1}{T_{nom}^{(K)}}$$

Parametri necessari:

Resistenza nominale: 10k
Temperatura nominale: 25°C → 298.15K
Coefficiente B: 3435

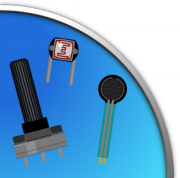
Termoresistenza – 1 (codice)

```
const int TMP_NOMINALE = 25;
// In kiloohm
const int RESISTENZA_NOMINALE = 10;
const int COEFFICIENTE_B = 3435;

// In kiloohm
const int RESISTENZA_SERIE = 10;

const byte PIN_TERMOMETRO = A0;

void setup() {
  Serial.begin(9600);
}
```



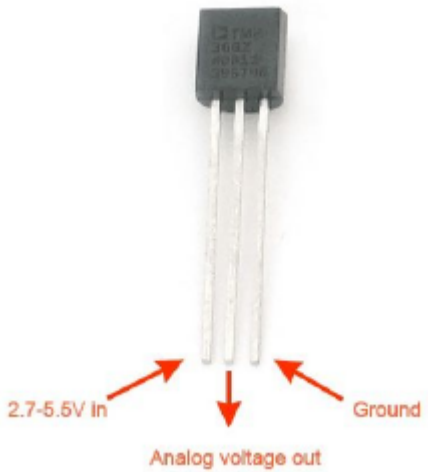
Termoresistenza – 2 (codice)

```
void loop() {
  float resistenza = analogRead(PIN_TERMOMETRO);
  resistenza = 1024.0 / resistenza;
  resistenza--;
  resistenza *= RESISTENZA_SERIE;

  Serial.print("Resistenza: ");
  Serial.println(resistenza);

  float temperatura = log(resistenza/RESISTENZA_NOMINALE);
  temperatura /= COEFFICIENTE_B;
  temperatura += 1.0 / (TMP_NOMINALE + 273.15);
  temperatura = 1.0 / temperatura;

  Serial.print("Temperatura: ");
  Serial.println(temperatura);
  Serial.println(""); // Riga vuota
  delay(500);
}
```

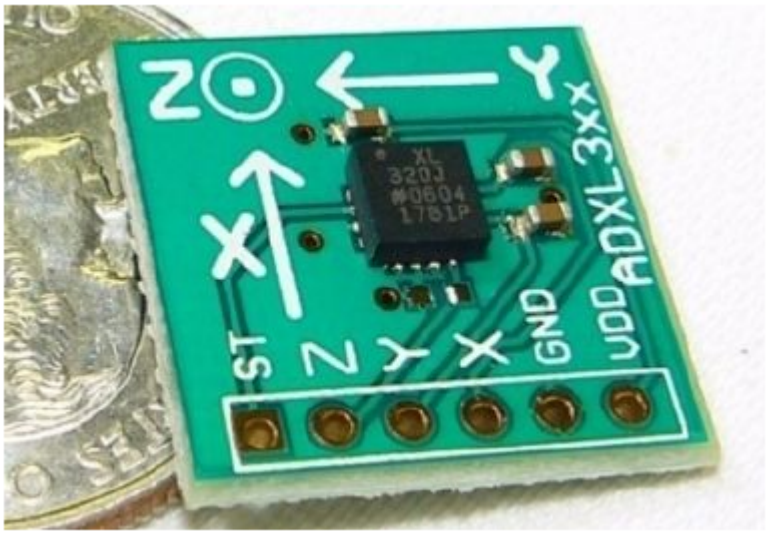


Sensori integrati

Temperatura *TMP36*



Accelerometro (ADXL3xx)



Distanza (Sharp)

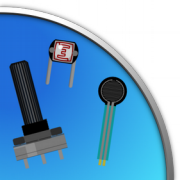
Esercizio 1

Luci passo-passo

Obiettivo: accendere o spegnere il LED quando si "clicca" sul pulsante.

Possono esserti utili gli **operatori logici**, per unire più condizioni insieme:

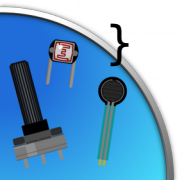
`condizione1 && condizione2`	AND
`condizione1 || condizione2`	OR



Esercizio 1

Luci passo-passo

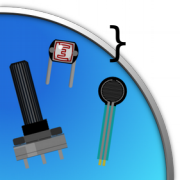
```
// Pin del pulsante
const byte PIN_PULSANTE = 2;
// Pin del LED
const byte PIN_LED = 13;
// Variabile di stato del pulsante
boolean statoPulsante;
boolean statoPulsantePrecedente;
void setup() {
    pinMode(PIN_LED, OUTPUT);       // LED in OUTPUT
    // pulsante in INPUT con PULLUP interno
    pinMode(PIN_PULSANTE, INPUT_PULLUP);
    // leggo il pulsante e memorizzo lo stato iniziale
    statoPulsantePrecendente = digitalRead(PIN_PULSANTE);
}
```



Esercizio 1

Luci passo-passo

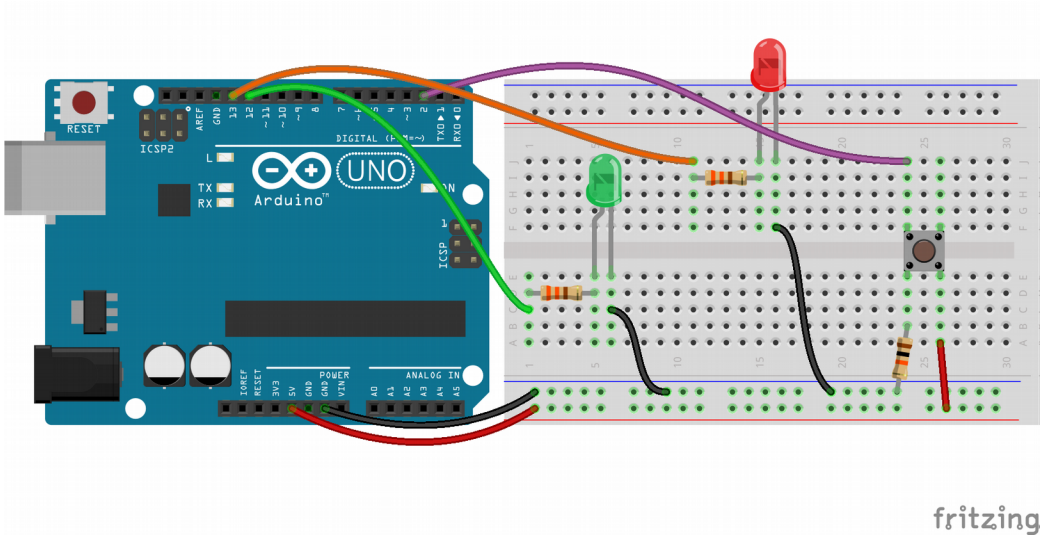
```
void loop(){
    // leggo il pulsante e memorizzo lo stato
    statoPulsante = digitalRead(PIN_PULSANTE);
    // se il pin è LOW (pulsante premuto), ma prima non lo era...
    if (statoPulsante == LOW && statoPulsantePrecendente == HIGH)
    {
        // inverte lo stato del led
        digitalWrite(PIN_LED, !digitalRead(PIN_LED));
        // aggiorno lo stato del pulsante precedente (mi servirà al
        // prossimo ciclo)
        statoPulsantePrecendente = statoPulsante;
    }
    else if (statoPulsante == HIGH && statoPulsantePrecendente == LOW)
         statoPulsantePrecendente = statoPulsante;
    delay(10);
}
```



Esercizio 2

Pulsante temporizzato

- **Obiettivo:** accendere il LED rosso se si preme il bottone per un breve istante (massimo 1 secondo). Accendere il LED verde se si preme il pulsante per un tempo maggiore.



Esercizio 2

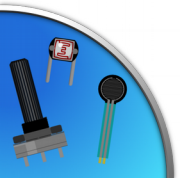
Pulsante temporizzato

- Può essere utile usare la funzione `millis()`

  Restituisce il tempo, **in millisecondi**, dall'avvio del programma. **Attenzione:** questo numero può essere grande, va memorizzato in variabili `unsigned long int`

- Se, quando si preme il pulsante, si memorizza il valore dato da millis, possiamo calcolare quanto tempo lo si sta tenendo premuto.

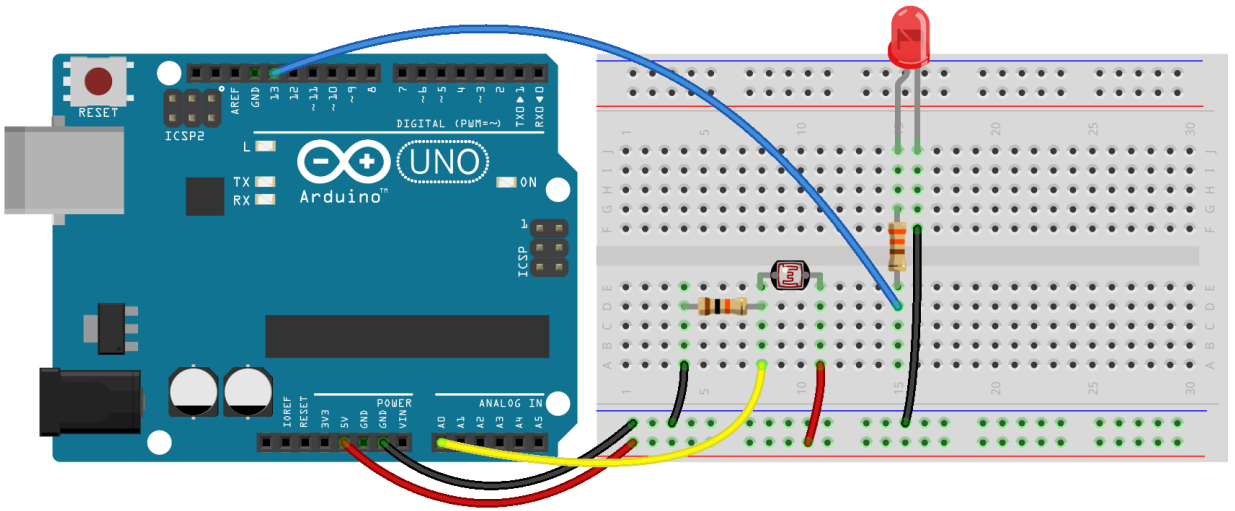
- La soluzione che fa uso di questo metodo è nel paragrafo 2.1.3



Esercizio 3

Interruttore crepuscolare

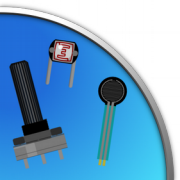
- **Obiettivo:** Usare una fotoresistenza per far accendere un LED se la luce dell'ambiente è troppo poca.



Esercizio 3

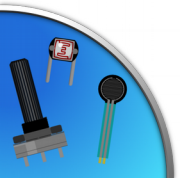
Interruttore crepuscolare

- **Problema:** quando la luce dell'ambiente è vicina a quella della soglia, il LED *sfarfalla*, si accende e spegne rapidamente.
- **Soluzione:** si inserisce una *doppia soglia* (isteresi):
  - Quando la luce cresce oltre la “soglia alta” il LED si spegne
  - Quando invece la luce scende sotto la “soglia bassa” il LED si accende
- La soluzione è nel paragrafo 2.2.2 della dispensa. Rifletti prima su come svolgerlo e poi guarda la soluzione!



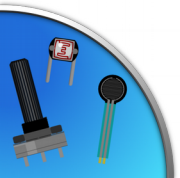
Approfondimento: Numeri casuali

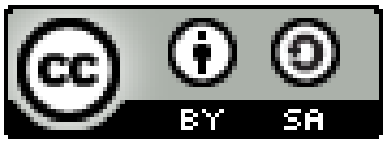
```
void setup() {
    // Usa Analog 0 come pin per la lettura
    randomSeed(analogRead(A0));
    // Attivo la comunicazione seriale
    Serial.begin(9600);
}

void loop() {
    unsigned int casuale;
    // un numero casuale da 1 a 29
    casuale = random(1, 30);
    Serial.println(casuale);
    delay(500);
}
```

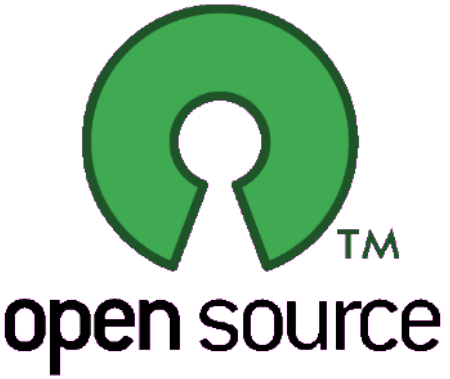


Esercizi per casa

- **Termostato:** usa la termoresistenza per realizzare un sistema a termostato per radiatori. Quando la temperatura scende sotto una certa soglia fai accendere un LED che simuli la caldaia.
  Ulteriori migliorie consigliate (opzionali!):
  - Soglia regolabile tramite un potenziometro;
  - Soglia regolabile tramite due pulsanti;
  - "Doppia soglia" per prevenire false accensioni/spegnimenti.
- **Luci natalizie:** accendi 5 LED in modo casuale *(bonus)*.







Presentazione realizzata con software open source
(LibreOffice Impress, Gimp, Arduino, Fritzing)