Arduino Explore IoT Kit

Obiettivo: Unboxing del kit Educational di Arduino Explorer IoT Kit

Componenti:

- Arduino MKR1010
- MKT IoT Carrier, che a sua volta include:
  2 relé 24V
  Slot scheda SD
  5 pulsanti touch
  Connettori plug-and-play per diversi sensori
  Sensore di temperatura
  Sensore di umidità
  Sensore di pressione
  Sensore UV
  Accelerometro
  Display RGB 1.20″
  Slot per batteria ricaricabile Li-Ion 18650
  5 LED RGB
- Cavo Micro USB
- Sensore di umidità
- Sensore a infrarossi passivo
- Cavi plug-and-play per tutti i sensori
- Accesso ad Arduino Create, una piattaforma online integrata che consente di scrivere codice, accedere a contenuti, configurare schede e condividere progetti
- Accesso alla piattaforma online dedicata con tutte le informazioni, le attività e i contenuti per usare il kit
- 10 lezioni hands-on passo-passo, che coprono tutti gli aspetti fondamentali legati all’IoT:

Hardware
Rete
Algoritmi e programmazione
Sicurezza
Gestione dei dati

- 10 sfide aperte

Link:

https://store.arduino.cc/explore-iot-kit
https://www.campustore.it/arduino-education-explore-iot-kit.html

---

Come Utilizzare il Monitor Seriale per Determinare se un Pulsante Funziona

Obiettivo: Utilizzare il monitor seriale di Arduino per avviare una comunicazione dati Arduino-PC e comprendere se un pulsante funziona oppure no.

Componenti elettronici:

- Arduino UNO
- Breadboard
- Pulsante

- Resistenza (1kOhm)

Prerequisiti

LED e Pulsante

Teoria: Non essendo disponibile un debugger per il controllore Arduino, l'utilizzo del monitor seriale rappresenta l'unica valida alternativa per comprendere i malfunzionamenti del codice scritto.
Il monitor seriale è uno strumento integrato nell'IDE di Arduino e nella piattaforma Tinkercad per visualizzare i dati ricevuti mediante comunicazione seriale.
La comunicazione seriale è una modalità di comunicazione tra dispositivi digitali nella quale i bit sono trasferiti lungo un canale di comunicazione uno di seguito all'altro. Nel caso specifico, la comunicazione avviene tra il Computer ed Arduino.
Le istruzioni per inviare messaggi da Arduino al Personal Computer sono due: **Serial.begin** e **Serial.println**

L'**Inizializzazione** della comunicazione avviene mediante l'istruzione:

```
Serial.begin(9600);
```

Questa istruzione deve essere inserita all'interno del corpo del setup e permette di impostare la comunicazione seriale

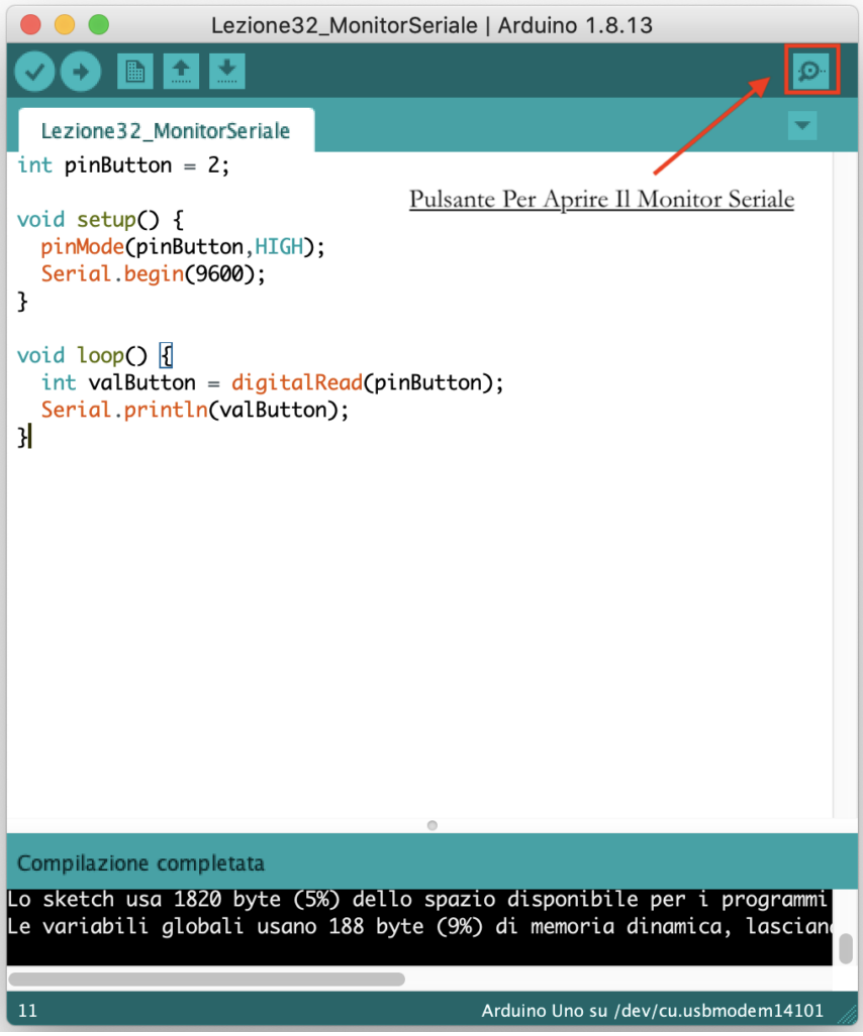
definendo la velocità della comunicazione in bits per second (baud).

La **comunicazione** vera e propria avviene invece mediante l'istruzione:

```
Serial.println("Il valore del pulsante risulta:");
```

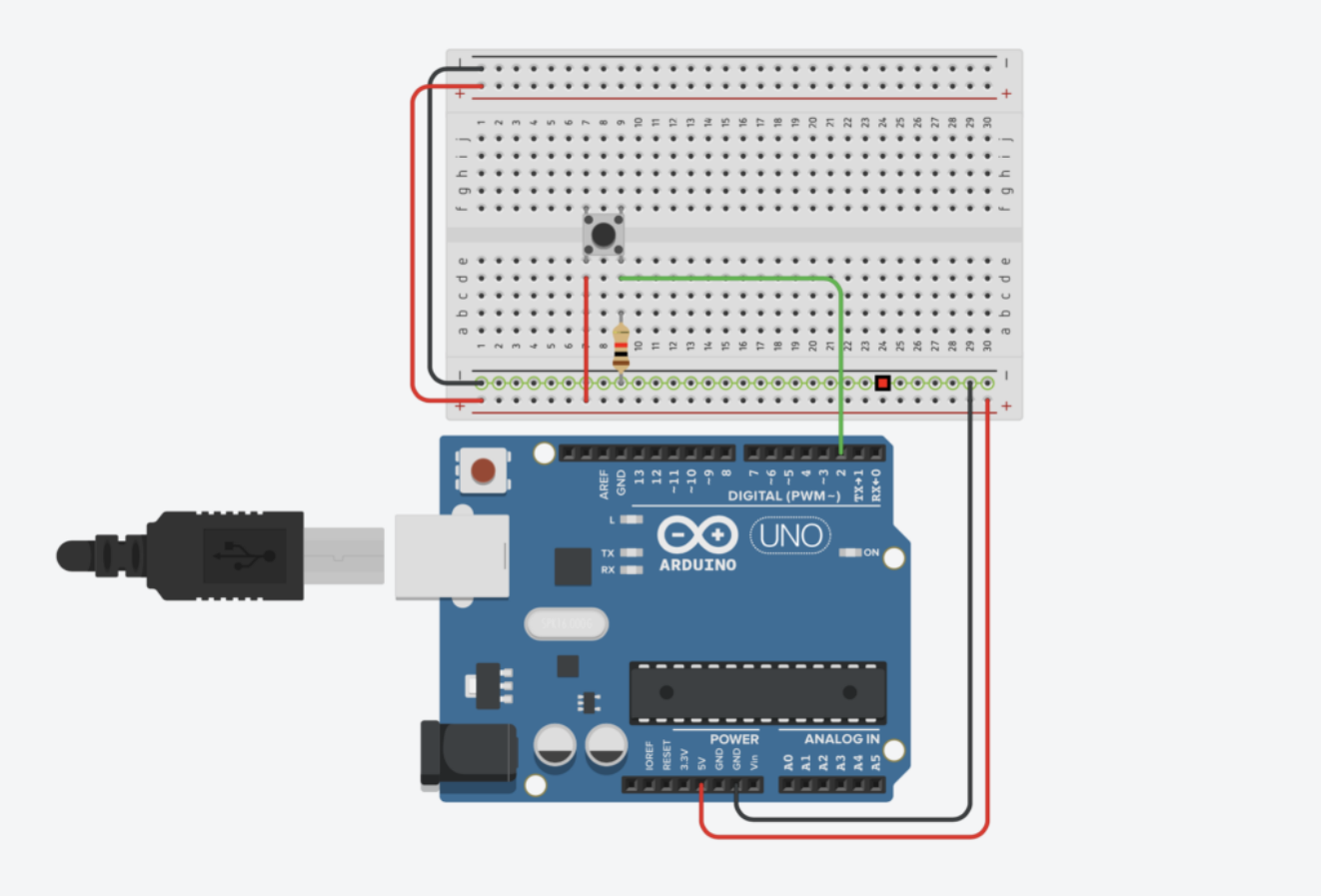
```
Serial.println(valButton);
```

Nel primo caso viene stampato nel monitor seriale il testo: "Il valore del pulsante risulta:". Mentre nel secondo caso viene stampato il valore della variabile valButton.
L'impiego delle println permette di capire il valore delle variabili e determinare il corretto funzionamento del circuito.
E' possibile visualizzare i dati inviati da Arduino al PC cliccando sullo specifico tasto:



Pulsante per aprire il monitor seriale

Collegamento Circuitale:



Collegamento Circuitale

<u>Codice</u>: A seguire viene riportato il codice utilizzato per determinare se un pulsante è stato premuto oppure no. Questo permette di comprendere se un pulsante è stato montato in modo corretto. Nello specifico il codice utilizza la variabile di stato “*valButton*” per determinare lo stato del pulsante (premuto/non premuto).
Attraverso l’istruzione Serial.println(valButton) è possibile stampare sul monitor il valore della variabile.

Quanto Tempo Hai Premuto il Pulsante?

Obiettivo: Determinare per quanto tempo un pulsante è stato premuto.

Componenti elettronici:

- Arduino UNO
- Breadboard
- Pulsante
- Resistenza (1kOhm)

Prerequisiti

Blinking Led Senza Delay: MILLIS()

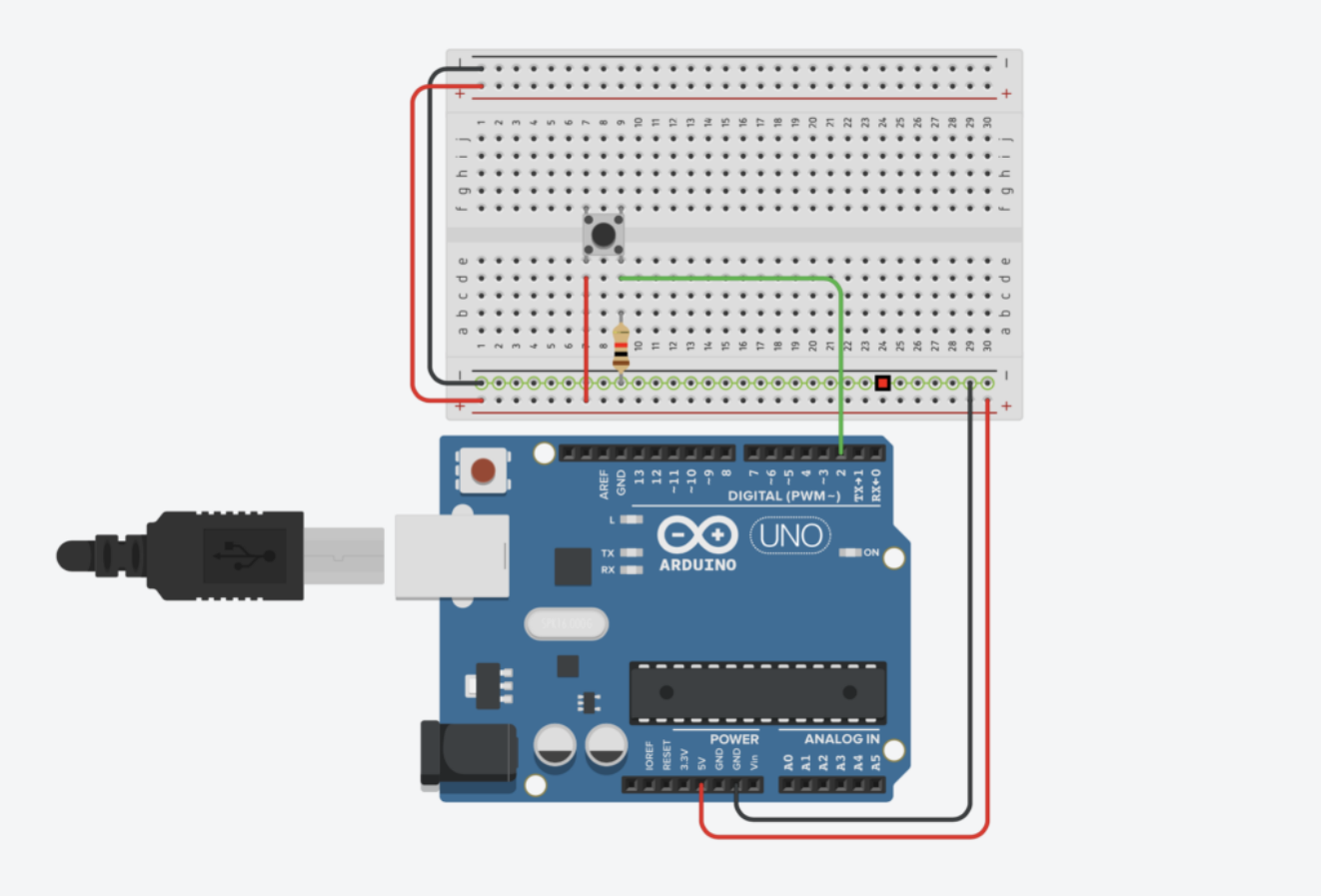
Pulsante come Interruttore

Teoria: Poter misurare il tempo di pressione di un tasto può risultare utile in molte applicazioni. Infatti, questa informazione permette di discriminare le differenti modalità di iterazione con il pulsante come il click (tasto premuto) ed

il long click (tasto premuto a lungo). Potere discriminare questi comportamenti permette di abilitare il pulsante a differenti funzioni. Ad esempio il single click potrebbe essere utilizzato per accendere un led mentre il long click potrebbe essere utile per farlo lampeggiare.
Dal punto di vista hardware il circuito necessario per realizzare questa applicazione è molto semplice ed è costituito dal singolo pulsante collegato a vcc e al ground mediante resistenza di pull-down.
Elemento centrale di questa esercitazione è la scrittura di un codice corretto che permetta di misurare esattamente lo scorrere quel tempo. Questo codice si basa sull'impiego di due elementi fondamentali:

- La funzione millis: questa funzione restituisce il numero di millisecondi che sono passati da quando la board Arduino ha eseguito il programma corrente. Questo numero si riazzera dopo circa 50 giorni. L'impiego di questa funzione è fondamentale per misurare il tempo di pressione del pulsante. Nel dettaglio, questa operazione può essere svolta semplicemente eseguendo la differenza tra le misure temporali prese quando il pulsante è premuto e quando il pulsante è rilasciato.
- La variabile di stato: questa variabile (di natura globale) permette di determinare quando un pulsante è premuto e quando il pulsante è rilasciato. Nello specifico grazie a questa variabile è possibile determinare un passaggio dallo stato logico basso a quello alto e viceversa.

Collegamento Circuitale:



Collegamento Circuitale

Codice: A seguire viene riportato il codice utilizzato per determinare il tempo di pressione di un pulsante. Nello specifico il codice utilizza la variabile di stato "*valButtonOld*" per memorizzare lo stato del pulsante relativo al ciclo passato.
Quando i valori di "*valButton*" e "*valButtonOld*" differiscono, allora c'è stato un passaggio di stato.

ValButtonOld	ValButton	Evento
LOW	HIGH	Il pulsante è stato premuto
HIGH	LOW	Il pulsante è stato rilasciato

Nel caso specifico del passaggio di stato viene effettuata una misura del tempo trascorso mediante la funzione millis(). Per determinare il tempo trascorso basta semplicemente effettuare

una differenza tra le due misure realizzate.

Personalizzazioni: E’ possibile modificare l’hardware introducendo due led. Quando il pulsante viene premuto per meno di un secondo deve accendersi il primo led, quando invece il pulsante viene premuto per più di un secondo deve accendersi il secondo.

Blinking Led Senza Delay: MILLIS()

Obiettivo: Realizzazione del classico blinking led senza utilizzare la funzione Delay

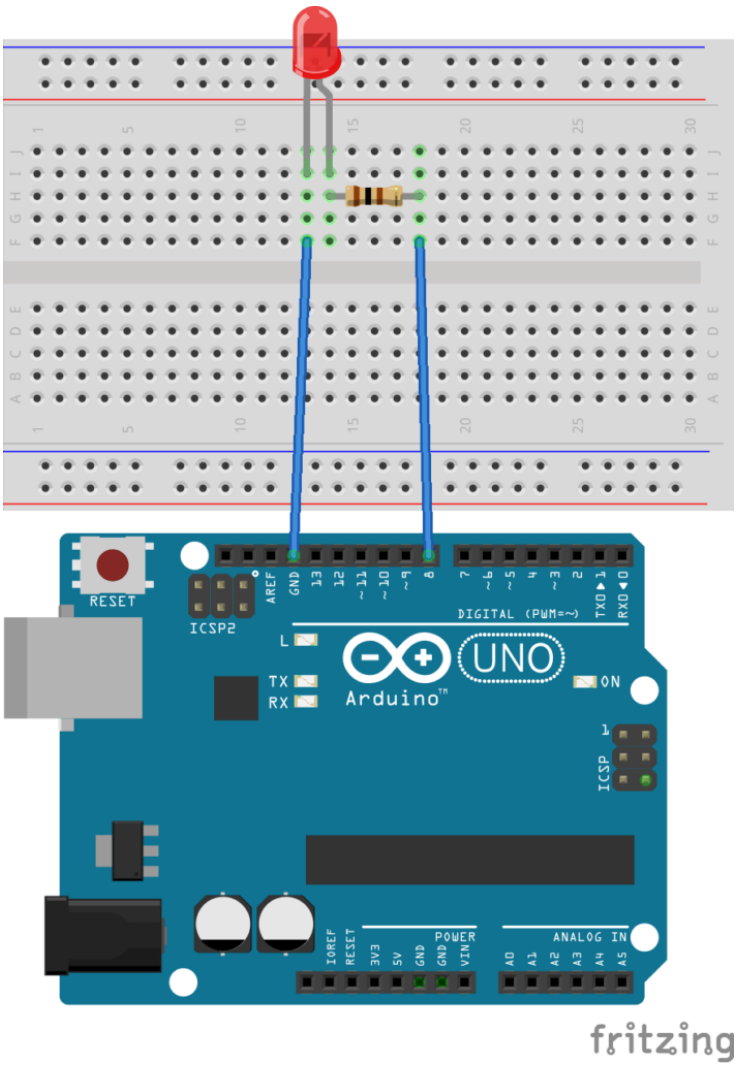
Componenti elettronici:

- Arduino UNO
- Breadboard
- Led
- Resistenza (100 Ohm)

Teoria: Se da un certo punto di vista l'impiego della funzione delay è particolarmente utile per la realizzazione di semplici applicativi; da un altro punto di vista molteplici potrebbero essere gli inconvenienti legati all'utilizzo di questa funzione.
E' infatti, molto importante, considerare che l'istruzione delay è un'istruzione bloccante. Questa istruzione "congela" Arduino nel suo stato corrente, pertanto negli istanti di delay non è possibile fare aggiornamenti, gestire input attraverso letture o comandare attuatori mediante scritture.
Ad esempio, nel caso della realizzazione di un semaforo per perdoni, dove è possibile effettuare la chiamata mediante la pressione di un pulsante, l'impiego della funzione delay è altamente sconsigliato per la gestione del semaforo perchè questo renderebbe invisibile la pressione del pulsate.
Per tutte queste ragioni è opportuno scrivere codici senza l'impiego della funzione delay. Nel dettaglio è opportuno sostituire la funzione delay con la più funzionale ma meno pratica funzione **millis.**

La funzione millis restituisce il numero di millisecondi che sono passati da quando la board Arduino ha eseguito il programam corrente. Questo numero si riazzera dopo circa 50 giorni.

Collegamento Circuitale:



Collegamento Circuitale

Codice:

A seguire viene riportato il codice utilizzato per fare lampeggiare il led senza delay. Nello specifico il codice memorizza l'ultimo istante in cui è avvenuta una azione (il led ha cambiato stato) nella variabile previousMillis mentre nella variabile currentMillis viene memorizzato il tempo corrente. Attraverso la differenza tra questi due tempi si comprendere se il led deve cambiare stato (accendersi o spegnersi). Nel dettaglio, se la differenza tra la variabile currentMillis e previousMillis è maggiore di 1000 millisecondi allora lo stato del led cambia.

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *interval*. Modificando il suo valore infatti cambia la frequenza di lampeggiamento del LED.
E’ inoltre possibile modificare il pin digitale utilizzato per pilotare il LED cambiando rispettivamente hardware e software.

L’utilizzo di una resistenza, in serie al LED, serve appunto per limitare la quantità di corrente presente sul diodo emettitore di luce.

Blinking led [Avanzato]

Obiettivo: Realizzazione, mediante breadboard, di un led che lampeggi ad una frequenza specifica (e.g., 1Hz) su un PIN differente dal 13.

Componenti elettronici:

- Arduino UNO
- Breadboard
- Led
- Resistenza (100 Ohm)

Teoria: Al fine di garantire il corretto funzionamento del LED

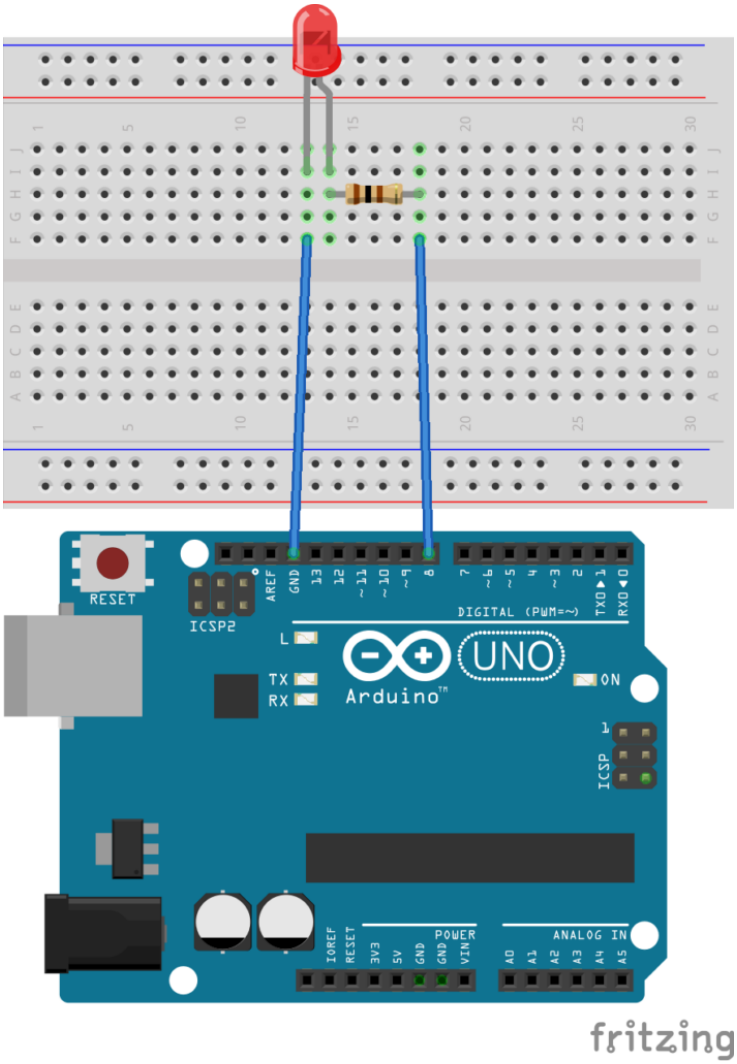
su un pin differente dal 13, è indispensabile l'utilizzo di una resistenza in serie al dispositivo. La resistenza permette di regolare fissare i corretti valori di tensione e corrente necessari ad alimentare il LED.
Ad esempio, considerando una tensione sul pin di Arduino pari a 5V ed i seguenti parametri caratteristici del LED:

- ILed = 20 mA
- VLed = 1,5 V

Data la **legge di Ohm** V=RI, la resistenza necessaria per garantire un corretto funzionamento del diodo emettitore di luce può essere così calcolata:

R = (5-2)/20*10^(-3)

Collegamento Circuitale:



Collegamento Circuitale

Codice:

[crayon-616ae908060fe116766825/]

Tinkercad:

https://www.tinkercad.com/embed/ckyY3Pamusd

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *ledTime*. Modificando il suo valore infatti cambia la frequenza di lampeggiamento del LED.
E’ inoltre possibile modificare il pin digitale utilizzato per pilotare il LED cambiando rispettivamente hardware e software.

Approfondimento Teorico:

Legge di Ohm: La legge di Ohm mette in relazione le tre grandezze elettriche fondamentali Tensione (Volt), Intensità di Corrente (Ampere) e Resistenza (Ohm) secondo la seguente relazione.

$$V = RI$$

L’utilizzo di una resistenza, in serie al LED, serve appunto per limitare la quantità di corrente presente sul diodo emettitore di luce.