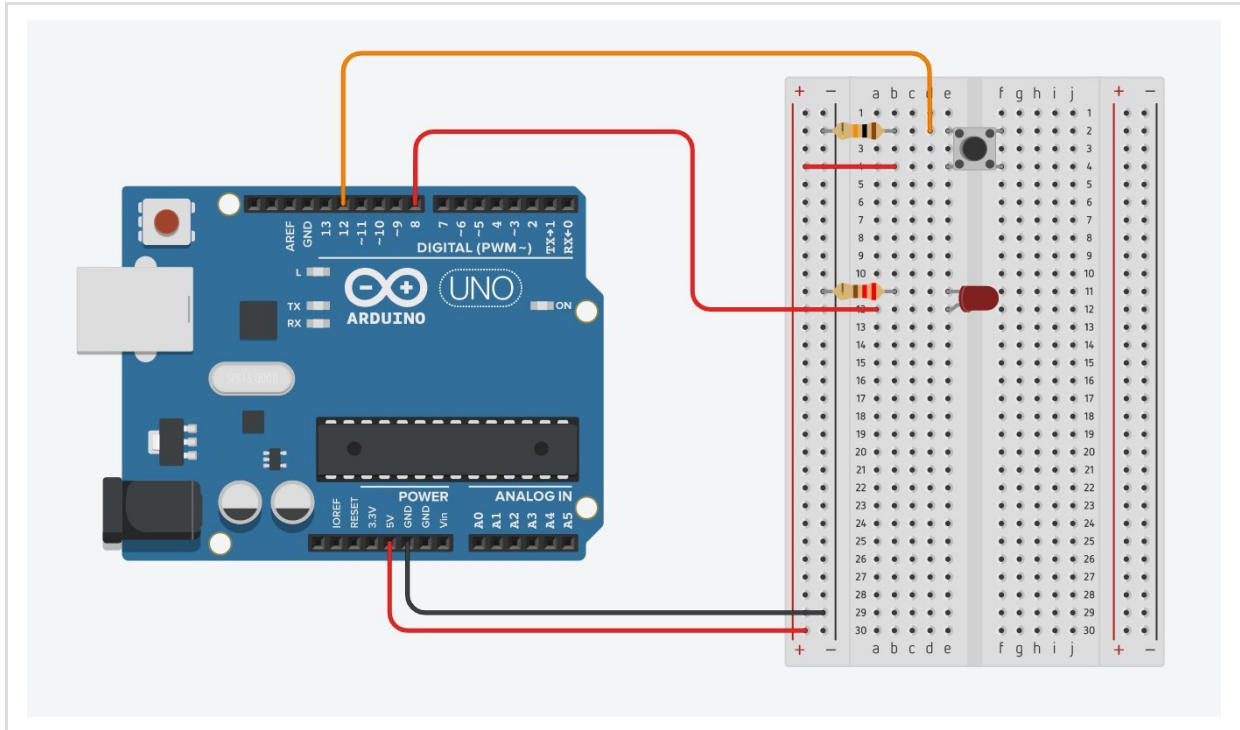


# Interrupteur

## (Allumer ou éteindre une DEL avec un bouton-poussoir)



### . Liste des composants :

- . 1 DEL rouge
- . 1 résistance de 220  $\Omega$  (résistance de protection de la DEL)
- . 1 bouton poussoir
- . 1 résistance de 10 k $\Omega$  (résistance du circuit du bouton poussoir)
- . 1 plaque d'essai
- . Fils de connexion

### . Objectif

Dans cette activité, quand la DEL est éteinte, si on appuie sur le bouton poussoir, la DEL s'allume, mais contrairement à l'activité "Bouton-poussoir", la DEL ne s'éteint pas quand on relâche le bouton poussoir. En effet, si la DEL est allumée, celle-ci s'éteint en appuyant de nouveau sur le bouton poussoir. Le principe de fonctionnement du bouton poussoir est donc comme celui d'un interrupteur.

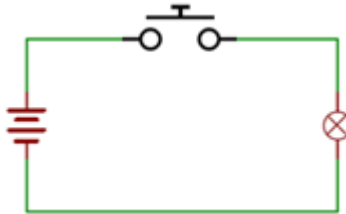
Pour réaliser cette activité, on va demander à l'Arduino d'interroger l'état logique (niveau haut ou bas) de la broche du bouton poussoir qui a été déclaré comme une entrée numérique. A l'aide de variables permettant de stocker les valeurs (actuelle et précédente) de cet état, l'Arduino pourra savoir quelle action effectuer (allumer ou éteindre la DEL) après l'appui sur le bouton poussoir.

## . Le bouton poussoir normalement ouvert

Un bouton poussoir normalement ouvert est un dispositif mécanique doté de 4 broches et d'une lamelle métallique qui met en contact toutes les broches lorsqu'on appui sur la tête du bouton. Un ressort de rappel ramène la tête du bouton lorsqu'il est relâché.



Un bouton poussoir normalement ouvert n'est jamais qu'un fil qui est connecté au circuit électrique ou non selon sa position :

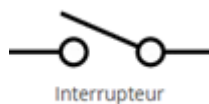
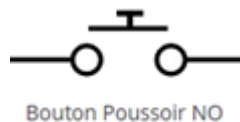


. **Relâché** : le courant ne passe pas dans le circuit électrique, le circuit est « ouvert » .

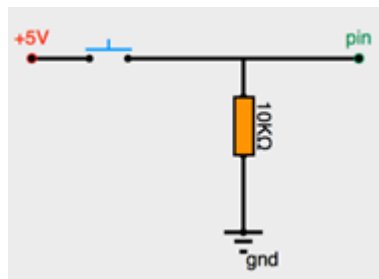
. **Appuyé** : le courant passe, on dit que le circuit est fermé.

Les mêmes effets peuvent être produits avec un interrupteur sauf que circuit reste fermé ou ouvert tant que la position de l'interrupteur n'a pas été changé.

Le bouton poussoir et l'interrupteur ne possèdent pas le même symbole pour les schémas électroniques :



Avec un Arduino, il est principalement utilisé pour envoyer une « impulsion de commande » avec ce circuit électrique :



Quand le bouton poussoir est appuyé, le potentiel de sa broche connectée à la broche « **pin** » de l'Arduino passe à 5 V (le circuit est fermé) et quand il est relâché, celui-ci passe à 0 V (le circuit est ouvert).

Avec notre circuit, on pourra alors demander à l'Arduino, d'allumer ou d'éteindre la DEL connectée sur la broche 9 en fonction de la valeur du potentiel de la broche 12 de l'Arduino.

Avec le bouton poussoir, nous allons donner, à l'Arduino, l'ordre d'effectuer une action. D'où le terme « impulsion de commande ».

### **Attention :**

**Il est impératif d'utiliser une résistance de 10 kΩ en série avec le bouton poussoir dans un montage « impulsion de commande ».**

**Ainsi, le courant dans le circuit est très faible ( $I = U/R = 0,5 \text{ mA}$ ) quand le circuit est fermé (bouton poussoir appuyé), et il n'y a pas de risque pour l'Arduino.**

## **. Gestion des entrées numériques**

En langage Arduino, pour lire l'état logique d'une entrée numérique, on utilise la fonction :

### **digitalRead()**

Lit l'état (= le niveau logique) d'une broche initialisée en entrée numérique, et renvoie la valeur HIGH (HAUT en anglais) ou LOW (BAS en anglais).

### **. Syntaxe :**

**digitalRead(broche)**

### **. Paramètre :**

broche: le numéro de la broche de la carte Arduino

### **. Valeur retournée :**

renvoie la valeur HIGH (1) ou LOW (0)

## . Le programme

### Interrupteur

```
// Déclaration des constantes et variables

const int PinLED = 8;
const int PinButton = 12;

int ValButton = 0;
int OldValButton = 0;
int State = 0;

// Initialisation des entrées et sorties

void setup() {
  pinMode (PinLED, OUTPUT);
  pinMode (PinButton, INPUT);
}

// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);
  if ((ValButton == HIGH) && (OldValButton == LOW)) {
    State = 1 - State;
  }
  OldValButton = ValButton;
  if (State == 1) {
    digitalWrite(PinLED, HIGH);
  }
  else {
    digitalWrite(PinLED, LOW);
  }
}
```

Déroulement du programme :

Déclaration des constantes et variables

- N° de la broche correspondant à la DEL: **const int PinLED = 8**
- N° de la broche correspondant au bouton poussoir: **const int PinButton = 12**
- Variable pour stocker la valeur de la broche du bouton poussoir: **int ValButton = 0**
- Variable pour stocker l'ancienne valeur de la broche du bouton poussoir: **int OldValButton = 0**
- Variable correspondant à l'action à effectuer: **int State = 0**

**void setup()**

initialisation des entrées et sorties

- La broche de la diode est initialisée comme une sortie digitale. Des données seront donc envoyées depuis le microcontrôleur vers cette broche :

**pinMode (PinLED, OUTPUT)**

- La broche du bouton poussoir est initialisée comme une entrée digitale. Des données seront donc envoyées depuis cette broche vers le microcontrôleur :

**pinMode (PinButton, INPUT)**

**void loop()**

Fonction principale en boucle

Lecture de l'état logique de la broche du bouton poussoir:

**ValButton = digitalRead(PinButton)**

Attente (10 ms) pour éviter les perturbations dues au phénomène de rebond du bouton poussoir:

**delay(10)**

bouton poussoir appuyé (broche à +5V):  
**ValButton = HIGH**  
et avant relâché (broche à 0V):  
**OldValButton = LOW**

?

Stockage de l'état logique de la broche du bouton poussoir lu:

**OldValButton = ValButton**

**State = 1 ?**

Non

Oui

Eteindre la DEL

Niveau bas (0 V) sur la broche de la DEL

**digitalWrite(PinLED, LOW)**

Allumer la DEL

Niveau haut (+ 5 V) sur la broche de la DEL

**digitalWrite(PinLED, HIGH)**

Oui

Mis à jour de la variable correspondant à l'action à effectuer;  
**State = 1 - State**  
La variable ne peut donc prendre que 2 valeurs (0 ou 1):  
**State = 0: Eteindre la DEL**  
**State = 1: Allumer la DEL**