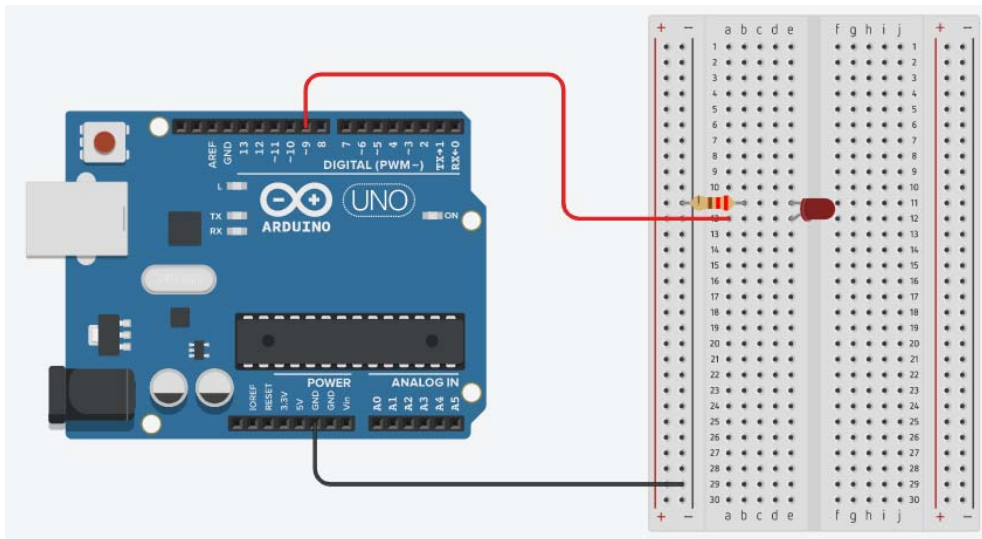


# Les activités d'apprentissage

## Activité 1 : Faire clignoter une DEL

Voici le premier circuit électrique que nous allons réaliser :



### - Liste des composants :

- . 1 DEL rouge connectée sur la broche 9 de l'Arduino
- . 1 résistance de 220  $\Omega$
- . 1 plaque d'essai
- . Fils de connexion

Avant de voir en détail l'activité basée sur ce circuit, nous allons faire quelques rappels sur l'électronique :

### . Petit rappel sur l'électricité

L'électricité est un déplacement d'électrons dans un milieu conducteur. Pour que ces électrons se déplacent tous dans un même sens, il faut qu'il y ait une différence du nombre d'électrons entre les deux extrémités du circuit électrique.

Pour maintenir cette différence du nombre d'électrons, on utilise un générateur (pile, accumulateur, alternateur...)

La différence de quantité d'électrons entre deux parties d'un circuit s'appelle la **différence de potentiel** et elle se mesure en **Volts (V)** et se note  $U$ .

Le débit d'électrons dans le conducteur correspond à *l'intensité*, aussi appelée *courant*. Elle se mesure en *Ampères (A)* et se note  $I$ .

La puissance électrique se note  $P$  et se mesure en *Watts (W)*. Elle exprime la quantité de courant ( $I$ ), transformée en chaleur ou en mouvement.

La puissance  $P$  est le produit de la tension  $U$  et de l'intensité  $I$ .

$$P = U \cdot I$$

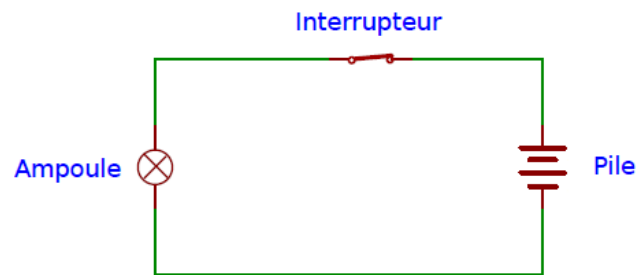
## . Le circuit électrique

Un circuit électrique est une association de dipôles (générateur, résistances, ampoules, ...) reliés par des conducteurs.

Pour qu'un courant circule, il faut que le circuit soit fermé et qu'il contienne un générateur (une pile par exemple). Il circule du pôle (+) au pôle (-) du générateur.

Une pile est constituée d'un milieu contenant de nombreux électrons en trop, et d'un second milieu en manque d'électrons. Quand on relie les deux pôles de la pile (le + et le -) avec un fil électrique (le conducteur), les électrons vont alors se déplacer du milieu riche en électrons vers le milieu pauvre.

Si on place une lampe électrique entre les deux, le passage des électrons va générer de la lumière.



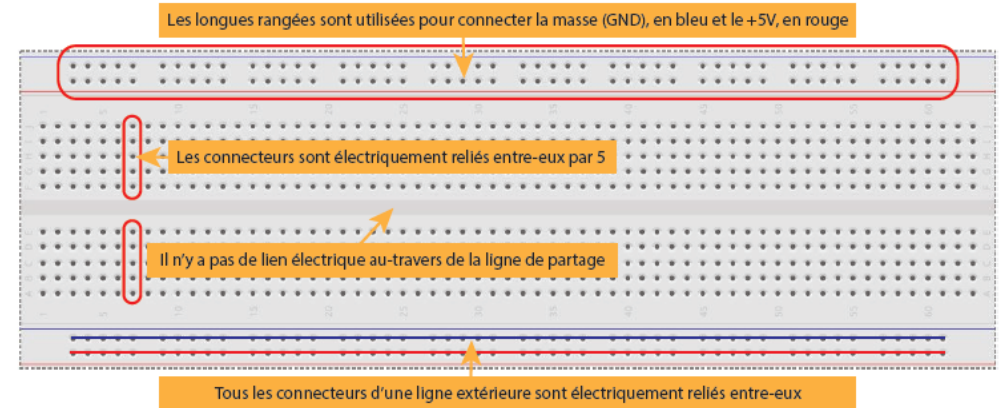
Dans le circuit ci-dessus, lorsque l'interrupteur est enclenché, on dit que le circuit est fermé. Les électrons vont alors se déplacer et l'énergie de ce déplacement pourra être exploitée pour allumer une lampe ou faire fonctionner un moteur, par exemple.

Lorsque l'interrupteur est déclenché, on dit que le circuit est ouvert. Le pôle positif n'étant alors plus relié au pôle négatif, les électrons ne peuvent plus se déplacer.

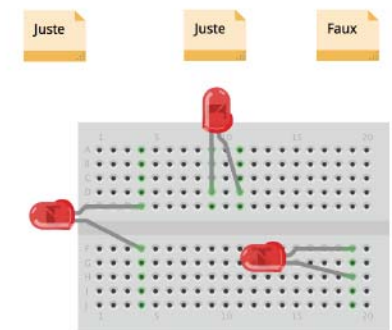
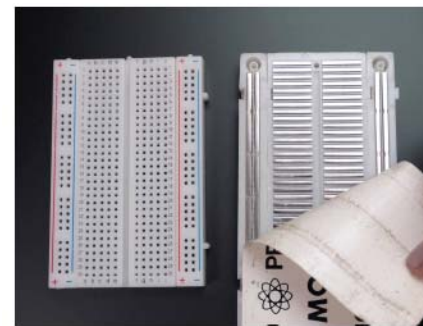
Dans les circuits électriques que nous allons étudier, L'Arduino servira d'alimentation électrique du circuit, comme une pile.

## . La platine d'expérimentation

Une platine d'expérimentation (appelée breadboard) permet de réaliser des prototypes de montages électroniques sans soudures et donc de pouvoir réutiliser les composants.



Tous les connecteurs dans une rangée de 5 sont reliés entre eux. Donc si on branche deux éléments dans un groupe de cinq connecteurs, ils seront reliés entre eux. Il en est de même des alignements de connecteurs rouges (pour l'alimentation) et bleus (pour la terre).



Les composants doivent ainsi être placés à cheval sur des connecteurs qui n'ont pas de liens électriques entre eux.

## . Les résistances

Une **résistance** est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (mesurée en ohms :  $\Omega$ ) à la circulation du courant électrique.

Ainsi, pour une tension fixe, plus la résistance est faible, plus le courant la traversant est fort. Cette proportion est vérifiée par la loi d'Ohm :

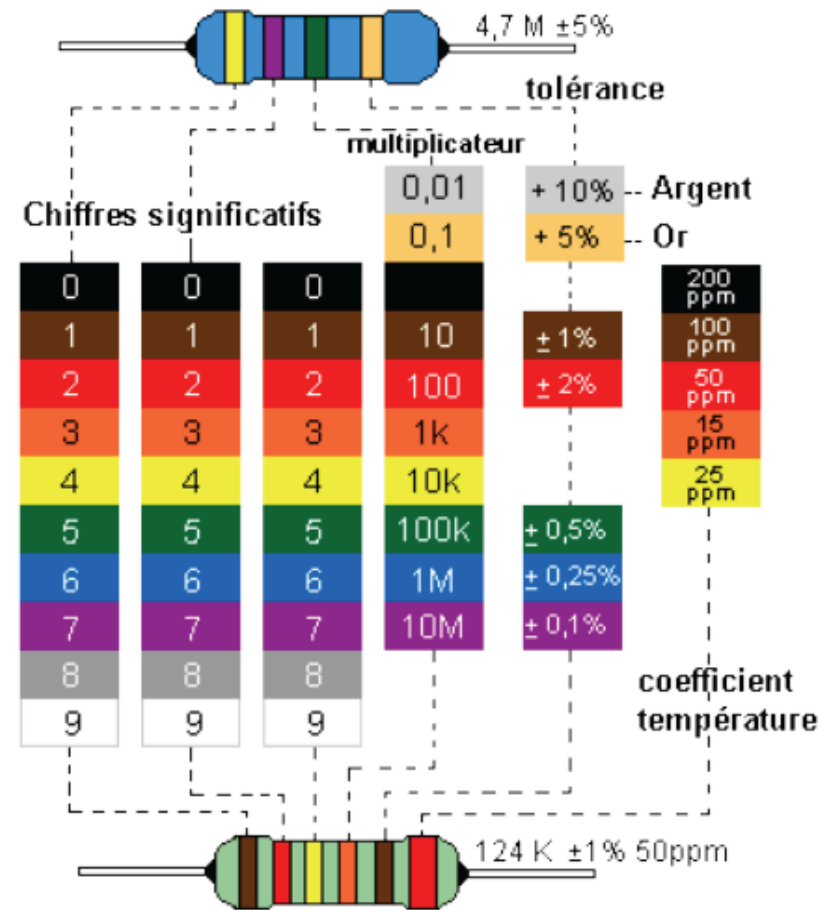
$$U = R \times I \text{ et donc } R = U/I \text{ et } I = U/R$$

Une résistance est un milieu peu conducteur. Les électrons peinent à s'y déplacer. Leur énergie se dissipe alors en général sous forme de chaleur. C'est ce principe utilisé pour les bouilloires électriques ou les ampoules à filaments.

La résistance est schématisée de cette manière :



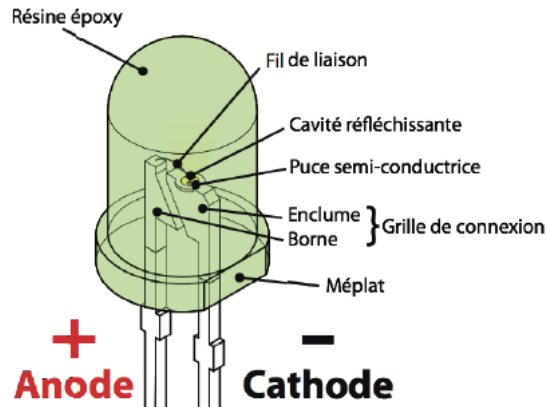
La valeur de la résistance se mesure en Ohms ( $\Omega$ ). La valeur d'une résistance est déterminée par ses bandes de couleurs :



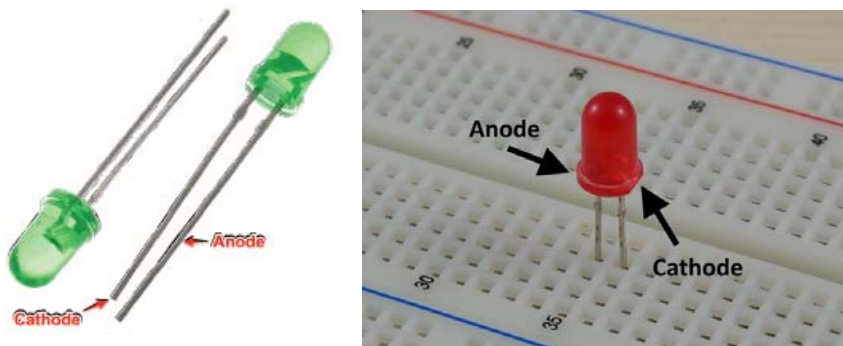
## . Les diodes électroluminescentes

La diode électroluminescente, aussi appelée DEL (ou LED en anglais), a la particularité de ne laisser passer le courant électrique que dans un sens.

Le courant électrique ne peut traverser la diode que dans le sens de l'*anode* vers la *cathode*.



On reconnaît l'anode, car il s'agit de la broche la plus longue. Lorsque les deux broches sont de même longueur, on peut distinguer l'anode de la cathode, par un méplat du côté de cette dernière.



Le symbole de la DEL est le suivant :



En utilisant divers matériaux semi-conducteurs, on fait varier la couleur de la lumière émise par la DEL et il existe une grande variété de formes de DELs.



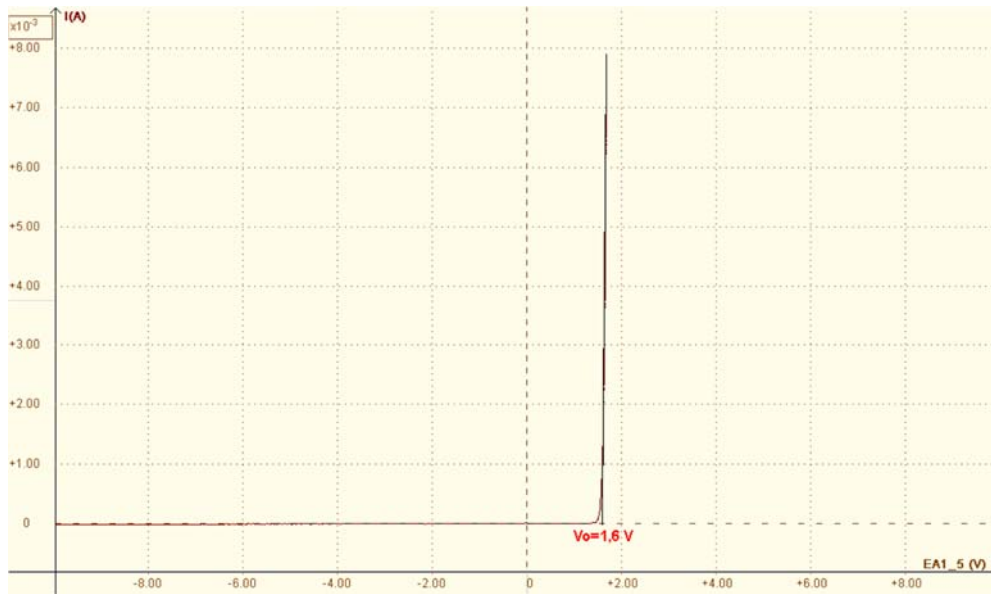
Attention : le courant produit par l'Arduino est trop important pour y brancher directement une DEL dessus.

**L'utilisation d'une résistance est obligatoire, pour ne pas griller la DEL.**

Mais si la valeur de la résistance est trop grande, la DEL ne s'allumera pas et au contraire, si la valeur de la résistance n'est pas suffisante, la DEL grillera.

Si on trace la caractéristique d'une DEL, c'est-à-dire, le graphe représentant l'intensité  $I$  traversant la DEL en fonction de la tension  $U$  à ses bornes :  $I=f(U)$ , on remarque qu'en dessous d'une certaine valeur de tension, le courant ne passe pas (la DEL ne s'allume pas).

On dit que la DEL est bloquante en dessous d'une tension seuil et passante au-dessus.



Pour que la DEL s'allume, il faut donc que la tension appliquée à ses bornes soit supérieure à sa tension seuil.

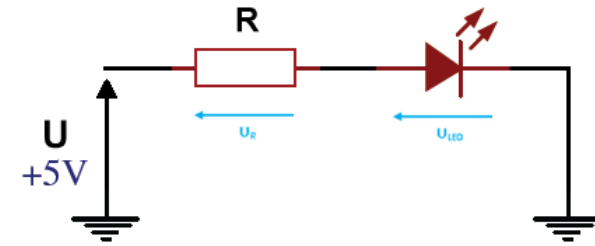
La tension seuil de la DEL,  $U_{LED}$ , dépend de sa couleur. Pour connaître sa valeur, il suffit de consulter sa fiche technique (Datasheet) donné par le constructeur de la DEL.

En général, on aura :

- . DEL Blanche :  $U_{LED} = 3,4$  à  $3,8$  V
- . DEL Jaune :  $U_{LED} = 2,1$  V
- . DEL Rouge :  $U_{LED} = 1,6$  V à  $2$  V
- . DEL Verte :  $U_{LED} = 2,2$  V
- . DEL Bleue :  $U_{LED} = 3,2$  à  $3,6$  V

Et pour qu'une DEL fonctionne dans des conditions optimales, les constructeurs préconisent généralement un courant de 20 mA (0,02 A) maximum.

Pour calculer la valeur de la résistance adéquate qu'il faut utiliser, on doit appliquer la loi d'Ohm au circuit suivant :



D'après la loi d'additivité des tensions dans un circuit en série,

$$U = U_R + U_{LED}$$

$$\text{Donc : } U = Ri + U_{LED}$$

$$R = \frac{U - U_{LED}}{I}$$

**Par exemple :**

Pour une LED rouge,  $U_{LED} = 1,6$  V, avec une alimentation de 5 V et une Intensité de 20mA maximum, la résistance minimale est :

$$R = (5 - 1,6) / 0,02 = 170 \Omega$$

En prenant une résistance de **220  $\Omega$** , nous sommes assurés de ne pas dépasser le courant maximal admissible par la DEL, et comme la DEL rouge est celle qui a une tension de seuil la plus basse, la résistance de 220  $\Omega$  peut être utilisée avec les autres DELs (l'intensité dans le circuit sera obligatoirement inférieure à 20 mA).

**A retenir :**

**Si on utilise une DEL dans un circuit alimenté par un Arduino, il est impératif de placer une résistance, par défaut de 220  $\Omega$ , en série avec elle. Cette résistance est appelée, résistance de protection, de façon à limiter le courant qui la traverse à 20 mA maximum.**

Comme première activité, nous allons faire clignoter une DEL rouge connectée sur la broche 9 (**programme "Activity1.ino" dans le dossier "Codes/Apprentissage"**).

Cette activité a pour but l'apprentissage de l'utilisation des **sorties digitales** de l'Arduino qui ne peuvent prendre que 2 valeurs : 0 (niveau bas) ou 1 (niveau haut), soit électriquement : 0 V ou +5 V.

Donc, pour allumer la DEL, la broche de l'Arduino sur laquelle celle-ci est connectée, doit être au niveau haut (+5V) et pour l'éteindre, elle doit être au niveau bas (0 V).

Pour réaliser cette activité, on va demander à l'Arduino d'allumer la DEL (donc d'appliquer un niveau haut sur la broche de la DEL) pendant une durée définie par une variable, puis de l'éteindre (donc d'appliquer un niveau bas sur la broche de la DEL) pendant une durée définie par une autre variable, puis à nouveau de l'allumer et cela indéfiniment. De cette façon, on verra la DEL clignoter.

## . Gestion des sorties numériques

- Les broches numériques de l'Arduino sont configurées en entrée ou en sortie à l'aide de la fonction :

### **pinMode()**

. Syntaxe :

**pinMode(broche, mode)**

. Paramètres :

broche: le numéro de la broche de la carte Arduino dont le mode de fonctionnement (entrée ou sortie) doit être défini.

mode: soit INPUT (entrée en anglais) ou OUTPUT (sortie en anglais)

- Pour modifier l'état logique d'une sortie numérique, on utilise la fonction :

### **digitalWrite()**

Cette fonction met un niveau logique HIGH (HAUT en anglais) ou LOW (BAS en anglais) sur une broche numérique.

Sa tension est mise à la valeur correspondante : 5V pour le niveau HAUT, 0V (masse) pour le niveau BAS.

. Syntaxe :

**digitalWrite(broche, valeur)**

. Paramètres :

broche: le numéro de la broche de la carte Arduino

valeur : HIGH ou LOW (1 ou 0)

## . Le programme

Le programme de l'activité pourra être modifié pour voir l'influence des variables (durée d'allumage et d'extinction) :

```
Activity1
// Déclaration des constantes et variables

const int PinLed = 9;
const int TimeSleep1= 500;
const int TimeSleep2= 500;

// Initialisation des entrées et sorties

void setup()
{
  pinMode(PinLed, OUTPUT);
}

// Fonction principale en boucle

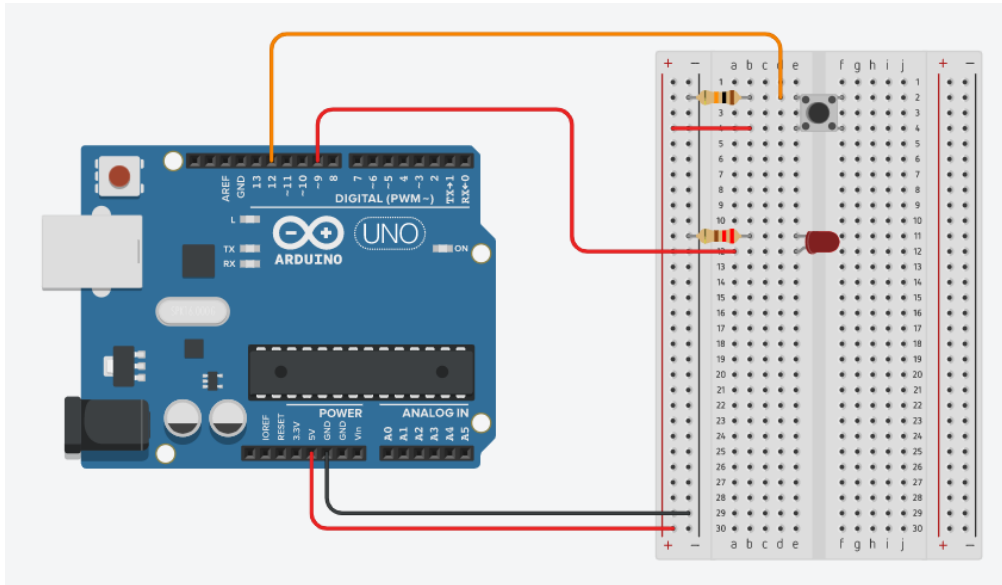
void loop()
{
  digitalWrite(PinLed, HIGH);
  delay(TimeSleep1);
  digitalWrite(PinLed, LOW);
  delay(TimeSleep2);
}
```

## Déroulement du programme :

- Déclaration des constantes et variables :
  - . **const int PinLed =9** (constante nombre entier correspondant au n° de la broche sur laquelle la DEL rouge est connectée)
  - . **const int TimeSleep1 = 500** (constante nombre entier correspondant à la durée d'allumage de la Del en ms)
  - . **const int TimeSleep2 = 500** (constante nombre entier correspondant à la durée d'extinction de la Del en ms)
- Initialisation des entrées et sorties :
  - . La broche de la DEL est initialisée en sortie digitale. Des données seront donc envoyés depuis le microcontrôleur vers cette broche :  
**pinMode(PinLed, OUTPUT)**
- Fonction principale en boucle :
  - . Niveau haut sur la broche de la Del : **digitalWrite(PinLed, HIGH)**
  - . Attente pendant TimeSleep1 ms : **delay(TimeSleep1)**
  - . Niveau bas sur la broche de la Del : **digitalWrite(PinLed, Low)**
  - . Attente pendant TimeSleep2 ms : **delay(TimeSleep2)**

## Activité 2 : Allumer une DEL avec un bouton poussoir

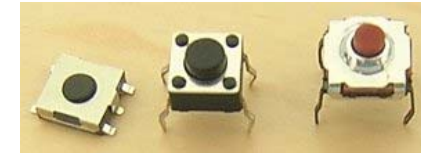
Ajoutons un bouton poussoir au circuit précédent :



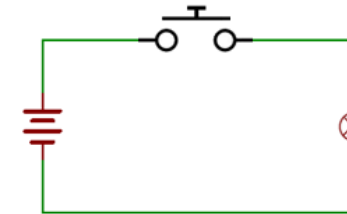
Dans cette activité, la DEL rouge s'allume en appuyant sur le bouton poussoir et s'éteint si on le relâche. L'objectif est de se familiariser avec les entrées numériques de l'Arduino.

### . 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 :

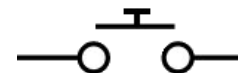


. **Relâché** : le courant ne passe pas dans le circuit électrique, le circuit est "ouvert".

. **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 :



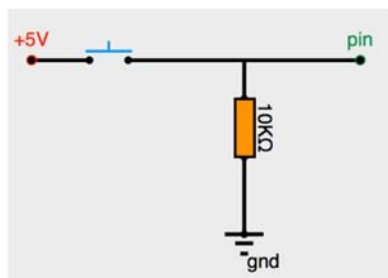
Bouton Poussoir NO



Interrupteur



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

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

Dans le programme de cette activité ("**Activity2.ino**" dans le dossier "**Codes/Apprentissage**"), on demande 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, on peut alors savoir si celui-ci est appuyé ou pas et donc lui donner l'ordre d'allumer ou d'éteindre la DEL.

```
Activity2
// Déclaration des constantes et variables

const int PinLED = 9;
const int PinButton = 12;

int ValButton = 0;

// Initialisation des entrées et sorties

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

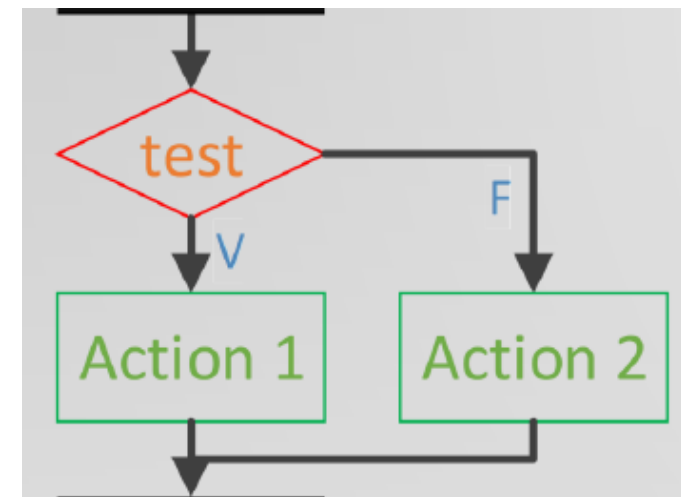
// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  if (ValButton == HIGH) {
    digitalWrite (PinLED, HIGH);
  }
  else {
    digitalWrite (PinLED, LOW);
  }
}
```

## Rappel sur la programmation d'une condition if - else:

```
Condition IF (Test)

if (test)
{
  // Action 1;
}
else
{
  // Action 2;
}
```



## Déroulement du programme :

### - Déclaration des constantes et variables :

- . **const int PinLed =9** (constante nombre entier correspondant au n° de la broche sur laquelle la DEL rouge est connectée)
- . **const int PinButton = 12** (constante nombre entier correspondant à la durée d'allumage de la Del en ms)
- . **int ValButton = 0** (variable nombre entier pour stocker la valeur de la broche du bouton poussoir)

### - Initialisation des entrées et sorties :

- . La broche de la DEL est initialisée en sortie digitale. Des données seront donc envoyés depuis le microcontrôleur vers cette broche :

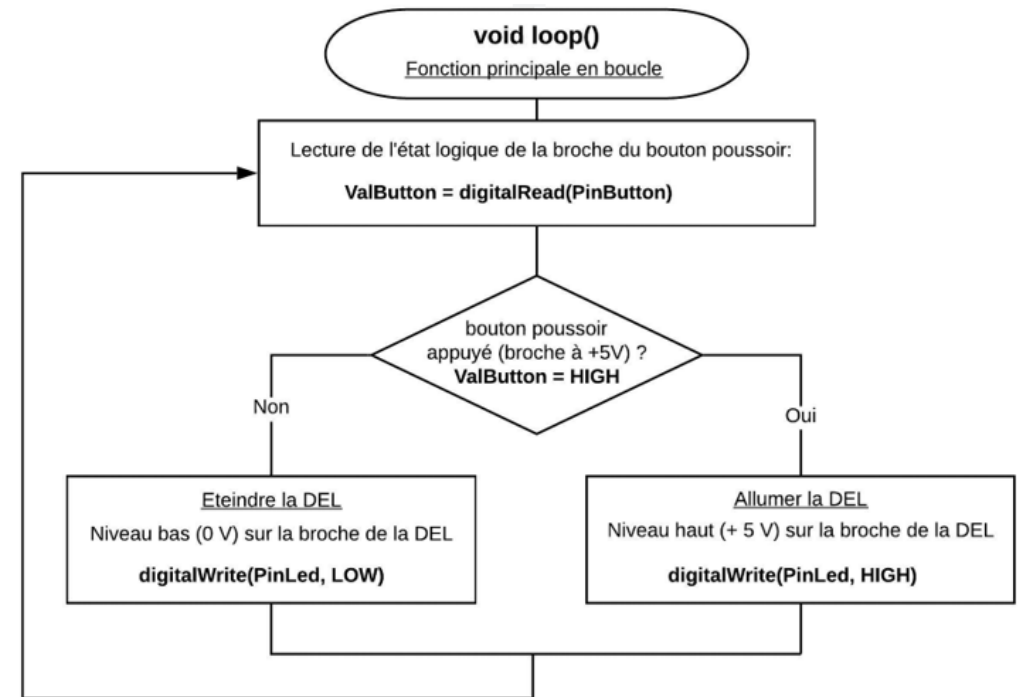
**pinMode(PinLed, OUTPUT)**

- . La broche du bouton poussoir est initialisée en entrée digitale.

Des données seront donc envoyés depuis cette broche vers le microcontrôleur :

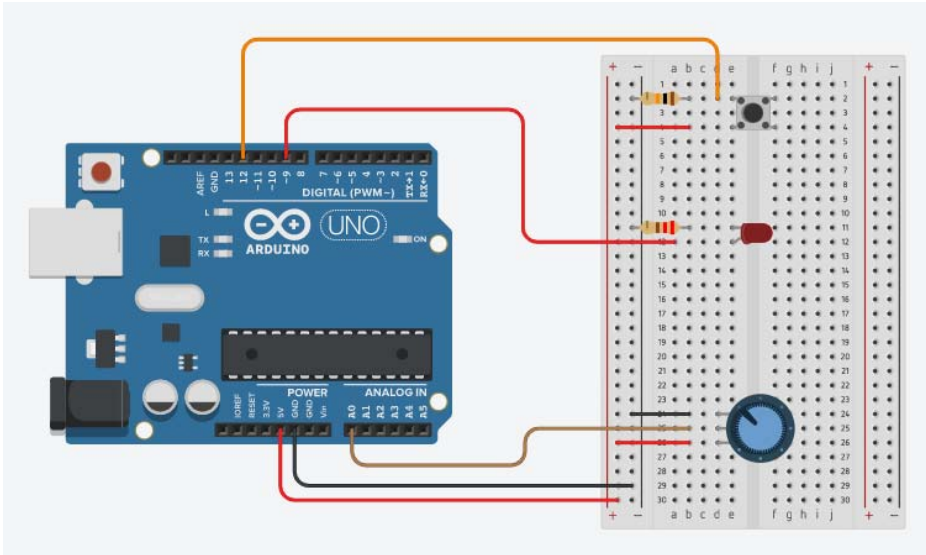
**pinMode(PinButton, INPUT)**

### - Fonction principale en boucle :



## Activité 3 : Le potentiomètre - Les entrées analogiques (CAN)

Ajoutons un potentiomètre à notre circuit d'apprentissage :



L'objectif de cette activité est l'apprentissage des entrées analogiques et de l'utilisation du moniteur série

### . Le potentiomètre

Le potentiomètre est une résistance variable. C'est le bouton de réglage du volume que l'on retrouve sur une radio. La plupart des potentiomètres sont soit rotatifs, soit linéaires.

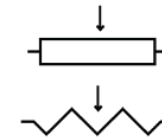


Potentiomètre rotatif



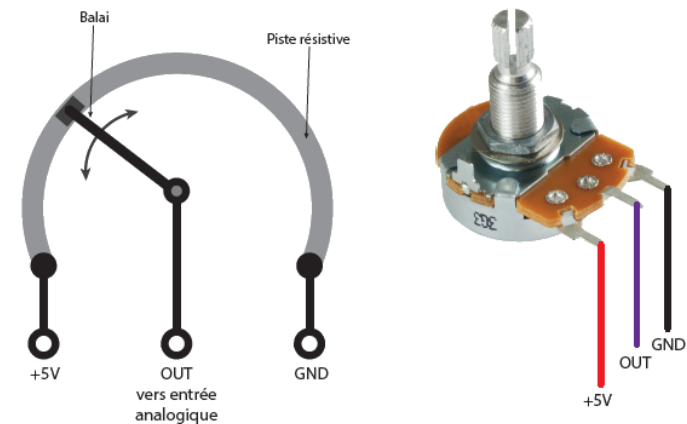
Potentiomètre linéaire

Voici les symboles électroniques (européen dessus et américain dessous) du potentiomètre :



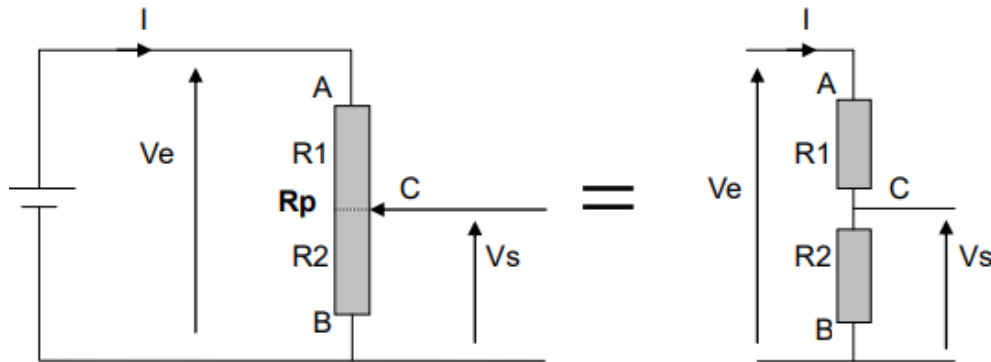
Comme toute résistance, le potentiomètre modifie la tension d'un circuit. On va donc l'utiliser principalement comme entrée (input) sur une broche analogique (A0 à A5) de l'Arduino.

Les potentiomètres ont en général trois broches :



Les broches extérieures se connectent sur l'alimentation +5V et sur la masse, alors que la broche centrale envoie le signal sur une broche d'entrée analogique de l'Arduino, comme sur le circuit ci-dessus (broche A0).

Dans ce montage, le potentiomètre de résistance totale  $R_p$  est utilisé en pont diviseur de tension :



On a :  $V_e = (R_1 + R_2) I$

$$I = \frac{V_e}{(R_1 + R_2)}$$

$$\text{Et : } V_s = R_2 I = \frac{R_2}{(R_1 + R_2)} V_e = \frac{R_2}{R_p} V_e \quad (\text{car } R_p = R_1 + R_2)$$

On peut remplacer le rapport  $R_2 / R_p$  par la position du curseur comprise entre 0 (position B) et 1 (position A).

Dans ce cas, la relation devient :

$$V_s = \alpha V_e \quad (\text{avec } \alpha \text{ la position du curseur : } 0 \leq \alpha \leq 1)$$

$V_s$ , qui est la tension appliquée sur une entrée analogique de l'Arduino, varie donc entre 0 et +5V en fonction de la position du curseur du potentiomètre.

De façon à limiter le courant dans le circuit à 0,5 mA, on utilise généralement des potentiomètres de **10 kΩ**.

## . Gestion des entrées analogiques

Pour lire la valeur de la tension d'une entrée analogique, on utilise la fonction :

**analogRead()**

. Syntaxe :

**analogRead(broche\_analogique)**

. Paramètres :

broche\_analogique : le numéro de la broche sur laquelle il faut convertir la tension analogique appliquée (0 à 5 sur la plupart des cartes Arduino)

. Valeur retournée :


valeur int (0 à 1023) correspondant au résultat de la mesure effectuée

. Remarque :

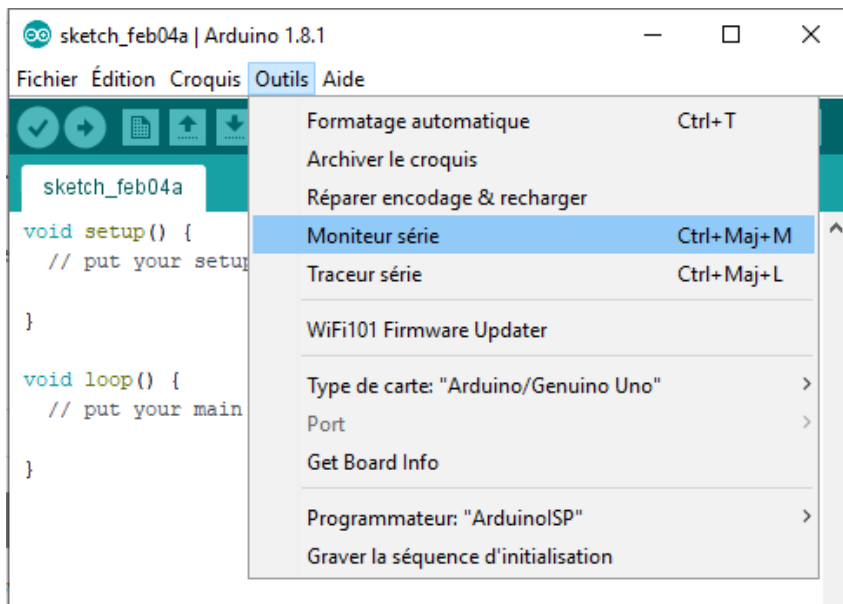
Il n'est pas nécessaire de déclarer les broches A0 à A5 en entrée avec la fonction **pinMode()** pour lire la tension appliquée sur celles-ci.

## . Le moniteur Série

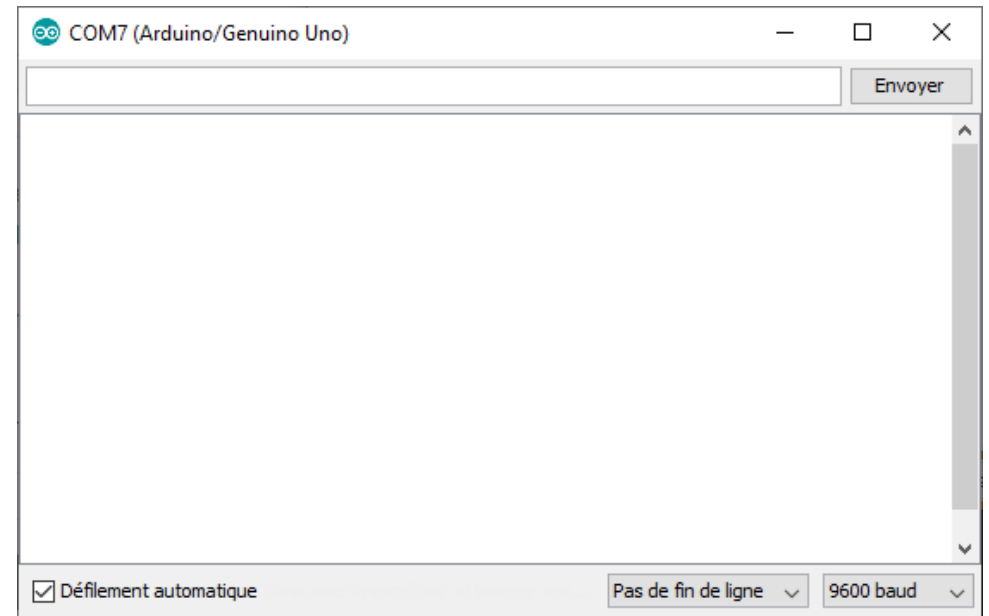
Le logiciel Arduino IDE dispose d'un moniteur série, qui permet de recevoir et d'envoyer des informations via une liaison série. Il est particulièrement intéressant pour les tests des programmes puisqu'il permet d'afficher les valeurs des variables, les états logiques des entrées et des sorties de l'Arduino, etc...

Le moniteur série est accessible depuis le logiciel Arduino en cliquant sur la loupe en haut à droite de la fenêtre du logiciel: 

Ou à partir du menu **Outils/Moniteur série** :



Une nouvelle fenêtre s'ouvre alors :



La fenêtre est composée de deux zones blanches d'exploitation et de plusieurs commandes :

- la grande zone au centre est la zone d'affichage des données reçues par le moniteur et donc envoyées par l'Arduino,
- la petite zone rectangulaire, en haut est une zone de saisie des données à envoyer à l'Arduino à l'aide du bouton "Envoyer" qui se trouve à sa droite. C'est le programme du microcontrôleur qui génère la demande d'envoi de données par l'utilisateur et qui s'occupe de traiter les informations reçues,

- la case cochée "Défilement automatique" permet d'arrêter le défilement des données retournées par l'Arduino si on la décoche,
- une liste déroulante permettant le réglage du mode de défilement des informations (Pas de fin de ligne, Nouvelle ligne, ...),
- une liste déroulante permettant de régler le débit de transmission des données par le port série en bauds.

#### Remarque :

Le baud est une unité de mesure utilisée dans le domaine des télécommunications en général, et dans le domaine informatique en particulier. Le baud est l'unité de mesure du nombre de symboles transmissibles par seconde.

Le terme "baud" provient du patronyme d'Émile Baudot, l'inventeur du code Baudot utilisé en télégraphie.

Il ne faut pas confondre le baud avec le bps ou bit par seconde, ce dernier étant l'unité de mesure du nombre d'information effectivement transmise par seconde. Il est en effet souvent possible de transmettre plusieurs bits par symbole. La mesure en bps de la vitesse de transmission est alors supérieure à la mesure en baud.

La valeur par défaut est de 9600 bauds. La carte Arduino peut monter jusqu'à une cadence de 115200 bauds. Mais un débit de communication de 9600 caractères (chaque caractère étant codé sur 8 bits, soit 1 octet) par seconde (**9600 bauds**) est généralement suffisant.

Pour utiliser le moniteur série, le programme téléversé dans la mémoire de l'Arduino doit établir la liaison série.

Pour cela, dans la fonction Setup() du programme, on utilise la fonction **begin()** de la classe "**Serial**" :

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

Le port série de la carte Arduino est alors configuré à 9600 bauds.

#### Important :

- Le microcontrôleur et le moniteur série doivent être configurés sur le même débit de transmission.
- Le programme du microcontrôleur est réinitialisé à l'ouverture du moniteur série.

La classe "**Serial**" regroupe toutes les fonctions utiles à la gestion d'un port série. Son travail est de gérer le traitement des données d'une communication série entre le moniteur série et le périphérique Arduino.

## Les principales fonctions de la classe Serial :

- . **begin();** (Configure la vitesse de transmission du port série)
- . **print();** (Envoie une donnée sous forme de chaîne de caractères sur le port série)
- . **println();** (Envoie une donnée sur le port série et fait un saut à la ligne)
- . **write();** (Ecrit des données binaires sur le port série. Ces données sont envoyées comme une série d'octets)
- . **read();** (Lis les données contenues dans la mémoire tampon (buffer) du port série)
- . **flush();** (Vide la mémoire tampon de la liaison série)
- . **available();** (Donne le nombre d'octets (caractères) disponible pour lecture dans la file d'attente (buffer) du port série.

Une fois la liaison série établie, il est possible d'envoyer des données depuis la carte Arduino vers le moniteur série avec la fonction "print()" de la classe "Serial".

On peut envoyer différents types d'informations :

- . Envoie d'une chaîne de caractères, sans saut de ligne à la fin :

```
Serial.print("Test");
```

- . Envoie d'une chaîne de caractères, avec saut de ligne à la fin :

```
Serial.println("Test");
```

- . Envoie de la valeur d'une variable, sans saut de ligne à la fin :

```
int a = 3;  
Serial.print(a);
```



## . Le programme

Dans le programme de cette activité ("**Activity3.ino**" dans le dossier "**Codes/Apprentissage**"), on demande à l'Arduino de lire la valeur de l'entrée analogique A0 et d'afficher le résultat en numérique et en analogique dans le moniteur série si la valeur lue est différente de la précédente (écart de plus d'une unité entre les valeurs lues)

```
Activity3
// Déclaration des constantes et variables
const int PinPOT = 0;
int ValPot = 0;
int OldValPot =0;
float Tension=0.00;

// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  Serial.println("Valeur A0 ; Tension (V):");
}

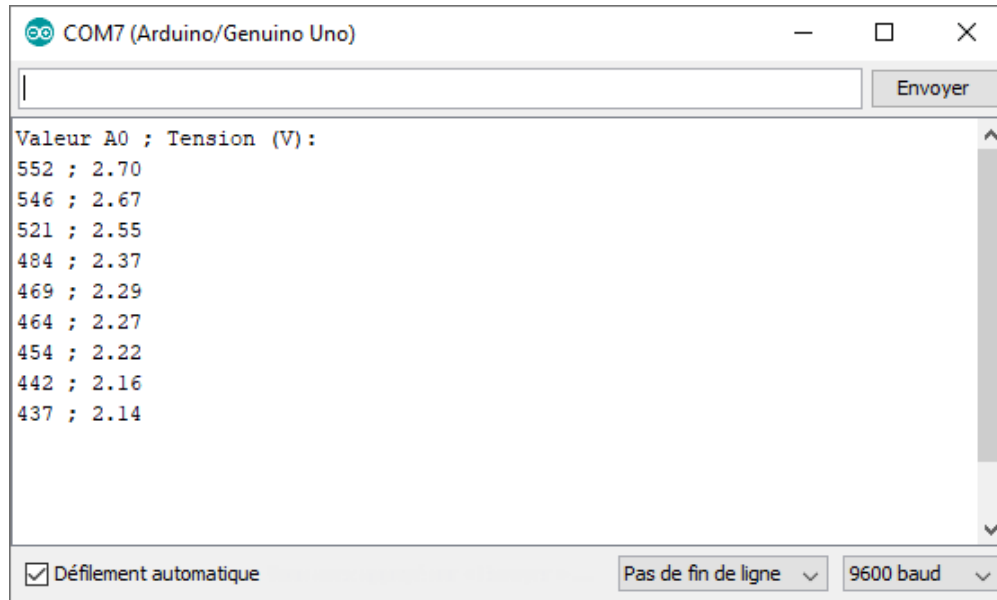
// Fonction principale en boucle

void loop() {
  ValPot = analogRead(PinPOT);
  if (abs(ValPot - OldValPot)>=2) {
    Serial.print(ValPot);
    Serial.print(" ; ");
    Tension=ValPot*5.00/1023;
    Serial.println(Tension);
    OldValPot = ValPot;
  }
  delay(100);
}
```

## Déroulement du programme :

- Déclaration des constantes et variables :
  - . **const int PinPOT = 0** (constante nombre entier correspondant au n° de la broche sur laquelle le potentiomètre est connecté)
  - . **int ValPot = 0** (variable nombre entier pour stocker la valeur de la broche du potentiomètre)
  - . **int OldValPot = 0** (variable nombre entier pour stocker la valeur précédente de la broche du potentiomètre)
  - . **float Tension = 0.00** (variable nombre décimal pour le calcul de la tension en V appliquée sur la broche du potentiomètre)
  
- Initialisation des entrées et sorties :
  - . La liaison série est initialisée à 9600 bauds :  
**Serial.begin(9600)**
  
- Fonction principale en boucle :
  - . Lecture de la valeur de la tension de l'entrée analogique du potentiomètre :  
**ValPot = analogRead(PinPot)**
  - . Affichage de la valeur lue dans le moniteur série si celle-ci est différente de la valeur précédente
  - . Calcul et affichage de la tension appliquée sur A0 en V
  - . Stockage de la valeur lue dans la variable OldValPot
  - . Attente de 100 ms avant une nouvelle mesure

## Résultat dans le moniteur série :



The screenshot shows a serial monitor window titled "COM7 (Arduino/Genuino Uno)". It features a text input field at the top with an "Envoyer" button. The main area displays a list of sensor readings. At the bottom, there are settings for "Défilement automatique" (checked), "Pas de fin de ligne" (dropdown), and "9600 baud" (dropdown).

```
Valeur A0 ; Tension (V):  
552 ; 2.70  
546 ; 2.67  
521 ; 2.55  
484 ; 2.37  
469 ; 2.29  
464 ; 2.27  
454 ; 2.22  
442 ; 2.16  
437 ; 2.14
```

Défilement automatique    Pas de fin de ligne    9600 baud

## Activité 4 : Régler la luminosité d'une DEL avec un potentiomètre

Notre circuit d'apprentissage, dispose d'un potentiomètre dont le "point milieu" est relié à la broche A0 de l'Arduino. Suivant la position du "point milieu", nous avons vu que la tension appliquée à la broche A0 varie entre 0 et 5 V. On peut donc utiliser le potentiomètre pour régler la luminosité de la DEL rouge.

En effet, la DEL est connectée sur une broche PWM et contrairement aux sorties numériques qui ne peuvent avoir que deux valeurs 0 ou 1 (0 ou 5V), une sortie analogique (ou plutôt PWM) permet d'obtenir une tension entre 0 et 5 V (les broches 3, 5, 6, 9, 10 et 11 peuvent être configurés en sortie analogique).

### . Gestion des sorties analogiques

Pour utiliser une broche de l'Arduino en sortie analogique (mode PWM), il faut au préalable la déclarer en sortie avec la fonction **pinMode()**.

La tension de la broche déclarée en sortie analogique est réglable en modifiant le rapport cyclique du signal PWM à l'aide de la fonction :

**analogWrite()**

.Syntaxe :

**analogWrite(broche, valeur)**

.Paramètres :

- broche: la broche utilisée pour "écrire" l'impulsion. Celle-ci devra être une broche ayant la fonction PWM (broches 3, 5, 6, 9, 10 ou 11).
- valeur: rapport cyclique du signal PWM, c'est à dire la proportion de l'onde carrée qui est au niveau HAUT, valeur entre 0 (0% HAUT donc toujours au niveau BAS) et 255 (100% HAUT donc toujours au niveau HAUT).

## . Le programme

Le programme ("Activity4.ino" dans le dossier "Codes/Apprentissage", lit la valeur de la broche A0 et applique cette tension sur la broche de la DEL rouge. La luminosité de la DEL est alors modifiée.

```
Activity4
// Déclaration des constantes et variables

const int PinPOT = 0;
const int PinLED = 9;
int ValPot = 0;

// Initialisation des entrées et sorties

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

// Fonction principale en boucle

void loop() {
  ValPot = analogRead(PinPOT);
  analogWrite(PinLED, int(ValPot/4));
  delay(100);
}
```

### Déroulement du programme :

- Déclaration des constantes et variables :
  - . **const int PinLED = 9** (constante nombre entier correspondant au n° de la broche sur laquelle la DEL est connectée)

- . **const int PinPOT = 0** (constante nombre entier correspondant au n° de la broche sur laquelle le potentiomètre est connecté)
- . **int ValPot = 0** (variable nombre entier pour stocker la valeur de la broche du potentiomètre)

### - Initialisation des entrées et sorties :

- . La broche de la DEL est initialisée en sortie digitale. Des données seront donc envoyés depuis le microcontrôleur vers cette broche :

**pinMode(PinLed, OUTPUT)**

### - Fonction principale en boucle :

- . Lecture de la valeur de la tension de l'entrée analogique du potentiomètre :

**ValPot = analogRead(PinPot)**

- . règle la luminosité de la DEL rouge en appliquant cette valeur sur la broche de la DEL :

**analogWrite(PinLED, int(ValPot/4))**

- . Attente de 100 ms avant une nouvelle mesure

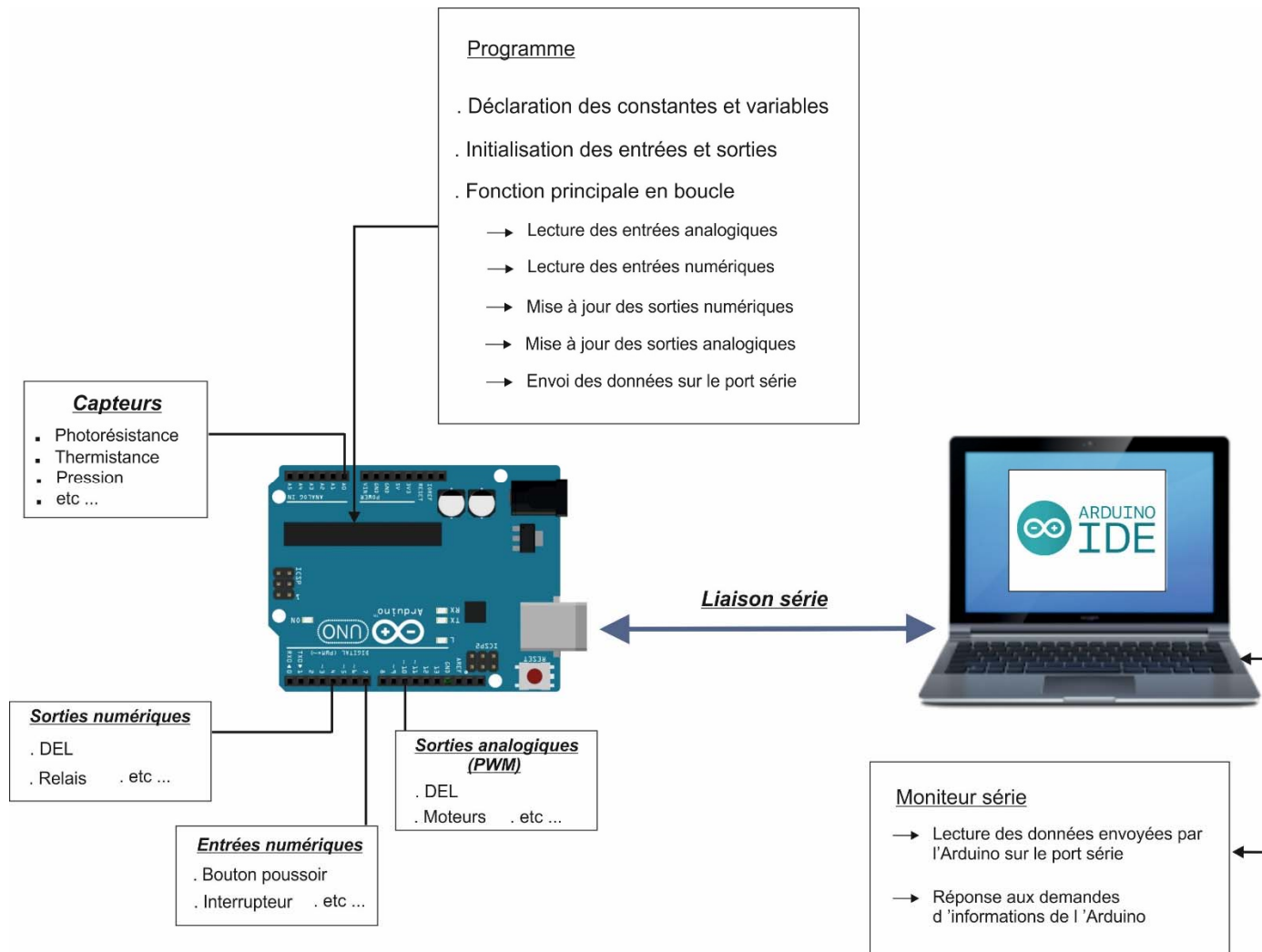
### Remarque :

La valeur lue de la broche A0 est un nombre entier compris entre 0 et 1023.

Cependant la fonction "**analogWrite()**" n'accepte que des valeurs entre 0 et 255. Il faut donc diviser par 4 la valeur lue de la broche A0 avant de l'appliquer sur la broche de la DEL (Le résultat de la division est d'abord converti en entier avec la fonction "**int()**")

# Synthèse

Nous avons vu le principe de fonctionnement des entrées et sorties, numériques ou analogiques, de l'Arduino. C'est la base de tous les programmes Arduino. En effet dans la plupart des cas, un programme, téléversé dans un Arduino, consistera à lire les valeurs de ses entrées analogiques ou numériques et de déclencher des actions en fonction des valeurs lues, celles-ci pouvant être affichées dans le moniteur série.



## . Rappels sur les fonctions principales d'un Arduino

### 1. Sorties numériques

- Pour utiliser une broche numérique de l'Arduino en sortie, il faut au préalable, généralement dans la fonction "**Setup()**", la configurer en sortie à l'aide de la fonction "**pinMode()**" :

. Syntaxe :

**pinMode(broche, OUTPUT)**

. Paramètre :

**broche**: le numéro de la broche de la carte Arduino dont le mode de fonctionnement en sortie doit être défini.

- Pour modifier l'état logique d'une sortie numérique, on utilise la fonction "**digitalWrite()**" :

. Syntaxe :

**digitalWrite(broche, valeur)**

. Paramètres :

**broche**: le numéro de la broche de la carte Arduino

**valeur** : **HIGH** ou **LOW** (1 ou 0)

### 2. Entrées numériques

- Pour utiliser une broche numérique de l'Arduino en entrée, il faut au préalable, généralement dans la fonction "**Setup()**", la configurer en entrée à l'aide de la fonction "**pinMode()**" :

. Syntaxe :

**pinMode(broche, INPUT)**

. Paramètre :

**broche**: le numéro de la broche de la carte Arduino dont le mode de fonctionnement en entrée doit être défini.

- Pour lire l'état logique d'une entrée numérique, on utilise la fonction "**digitalRead()**" :

. Syntaxe :

**digitalRead(broche)**

. Paramètre :

**broche**: le numéro de la broche de la carte Arduino

. Valeur retournée :

Renvoie la valeur **HIGH** (1) ou **LOW** (0)

### 3. Entrées analogiques

- Pour lire la valeur de la tension d'une entrée analogique, on utilise la fonction "**analogRead()**" :

. Syntaxe :

**analogRead(broche\_analogique)**

. Paramètres :

**broche\_analogique** : le numéro de la broche sur laquelle il faut convertir la tension analogique appliquée (0 à 5 sur la plupart des cartes Arduino)

. Valeur retournée :

valeur int (0 à 1023) correspondant au résultat de la mesure effectuée

- Il n'est pas nécessaire de déclarer les broches A0 à A5 en entrée avec la fonction "**pinMode()**" pour lire la tension appliquée sur celles-ci.

- Il est cependant possible d'utiliser les broches A0 à A5 en entrée ou sortie numérique. Il faut alors utiliser la fonction "**pinMode()**" pour les configurer comme telle.

### 4. Sorties analogiques (PWM)

- Pour utiliser une broche de l'Arduino en sortie analogique (mode PWM), il faut au préalable la déclarer en sortie avec la fonction "**pinMode()**" :

. Syntaxe :

**pinMode(broche, OUTPUT)**

. Paramètre :

**broche** : le numéro de la broche de la carte Arduino dont le mode de fonctionnement en sortie doit être défini. Celle-ci devra être une broche ayant la fonction PWM (broches 3, 5, 6, 9, 10 ou 11).

- La tension de la broche déclarée en sortie analogique est réglable en modifiant le rapport cyclique du signal PWM à l'aide de la fonction "**analogWrite()**" :

. Syntaxe :

**analogWrite(broche, valeur)**

. Paramètres :

**broche** : la broche utilisée pour "écrire" l'impulsion.

**valeur** : rapport cyclique du signal PWM entre 0 et 255