

Les capteurs analogiques

1. Les capteurs analogiques

Pour comprendre la différence entre ces deux familles de capteurs, prenons deux exemples :

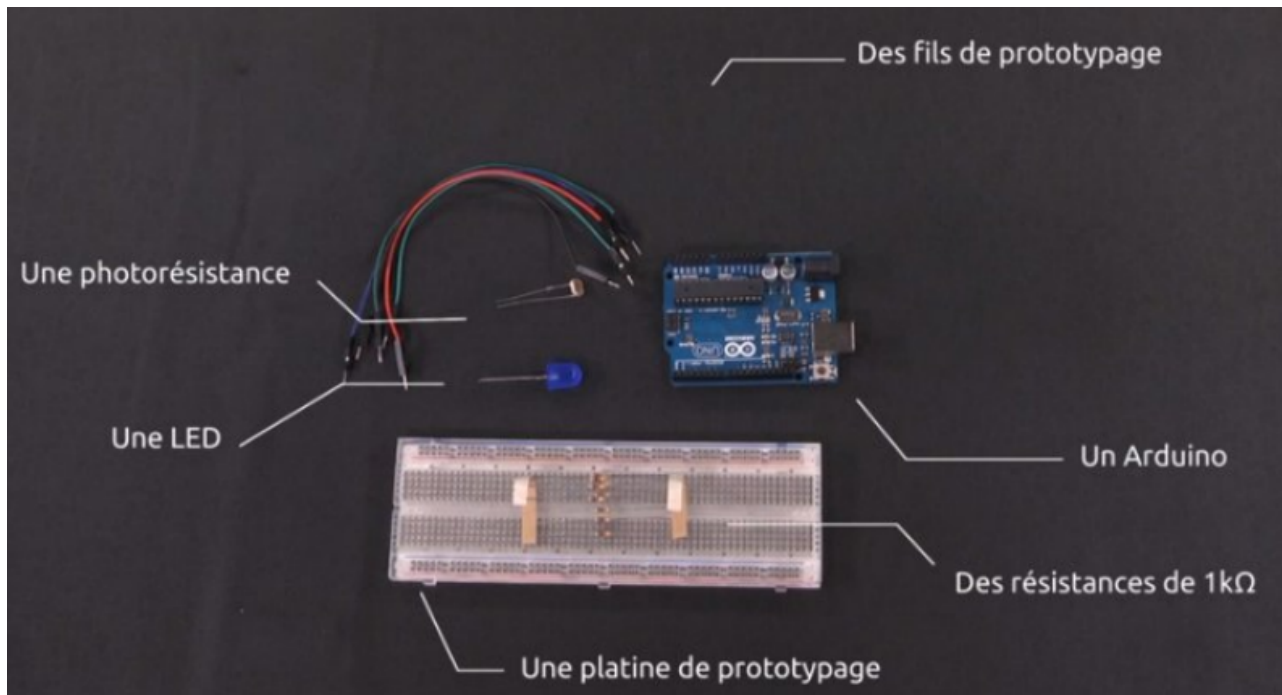
1. Le bouton poussoir est un capteur numérique qui peut nous donner deux informations sur l'état d'un bouton lorsqu'on le branche sur Arduino. Soit le bouton est appuyé (HIGH) soit il ne l'est pas (LOW)
2. Pour le capteur de luminosité qui est un capteur analogique, c'est différent ! Les valeurs renvoyées par Arduino si on lui branche ce capteur peuvent prendre des valeurs comprises entre 0 et 1023.

Nous pourrions ainsi savoir précisément si notre environnement est sombre ou lumineux selon si la valeur reçue est basse ou élevée.

On teste ça avec un montage qui va nécessiter :

- Un Arduino
- Un câble USB
- Une platine de prototypage
- Une photorésistance
- Deux résistances de $1k\Omega$
- Une LED de votre couleur préférée
- Des fils de prototypage

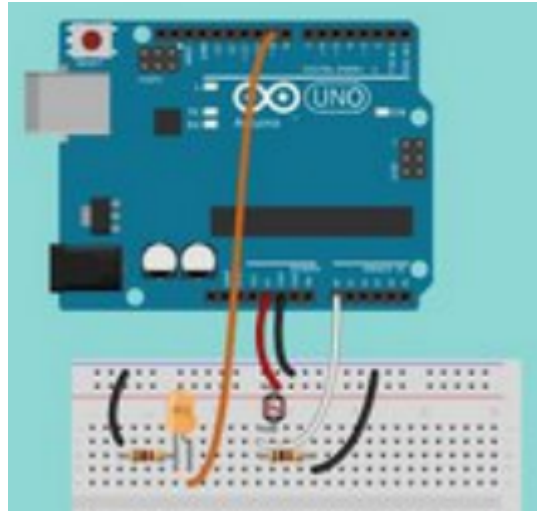
Voici le montage à réaliser :



2. Le code Arduino

On passe au code et je vous invite à ouvrir l'exemple AnalogInOutSerial ou à copier le code situé en dessous de la vidéo. Vous devriez avoir ceci :

```
const int analoInPin = A0;
const int analoOutPin = 9;
```



Intéressons nous maintenant aux instructions présentes dans le programme AnalogInOutSerial :

Après quelques lignes de commentaires, nous trouvons deux déclarations de constantes appelées analogInPin et analogOutPin qui correspondent à la photorésistance branchée sur la broche A0 et analogOutPin qui est la LED branchée sur la broche 9.

Nous déclarons ensuite deux variables entières appelées sensorValue et outputValue qui contiendront respectivement la valeur reçue par la photorésistance et la valeur envoyée à la LED.

```
int sensorValue = 0;
int outputValue = 0;
```

2.1. Bloc setup

On passe au setup qui contient une seule instruction :

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

C'est une nouveauté, vous noterez que cette fonction est séparée par un point. Cela signifie qu'elle appartient à une bibliothèque qui s'appelle ici **Serial**.

Nous reviendrons sur ce qu'est une bibliothèque mais sachez que la fonction begin qui suit le point permet de dire que l'on souhaite communiquer avec l'ordinateur en utilisant notre port USB.

En effet, le port USB que nous utilisons pour téléverser nos programmes sur l'Arduino permet aussi d'envoyer et de recevoir du texte avec un ordinateur.

Le paramètre 9600 est la vitesse d'échange des caractères.

```
void setup()
{
    // Initialise la communication avec l'ordinateur
    Serial.begin(9600);
}
```

2.2. Bloc loop

Nous passons ensuite au bloc loop.

Nous allons utiliser `analogRead` qui renvoie une valeur numérique comprise entre 0 et 1023.

Nous allons donc récupérer dans la variable `sensorValue` un nombre que nous allons ensuite renvoyer dans une LED pour définir son intensité lumineuse.

Il est en effet possible d'allumer une LED un petit peu grâce à la fonction `analogWrite` qui prend deux paramètres : le premier est le numéro de la broche qui va recevoir le courant. Le second paramètre est la valeur que nous voulons écrire comprise cette fois-ci entre 0 et 255.

Cependant, comme vous pouvez le remarquer, nous avons ici un problème : avec `analogRead`, nous recevons une valeur comprise entre 0 et 1024 et nous devons ensuite écrire une valeur comprise entre 0 et 255.

```
analogWrite(analogOutPin, outputValue);
```

Deux solutions existent pour répondre à ce problème, la première est de faire une règle de trois de ce type :

```
outputValue = (sensorValue * 255) / 1024;
```

Mais nous avons choisi d'utiliser une fonction appelée `map` qui est offerte par le langage Arduino et qui permet de faire passer une valeur située dans une intervalle vers un autre.

Les paramètres de ces fonctions sont les suivants :

1. valeur dans l'intervalle initiale
2. début de l'intervalle initiale
3. fin de l'intervalle initiale
4. fin de l'intervalle visée
5. début de l'intervalle visée

Grâce à cette fonction, nous allons donc nous retrouver avec une valeur proportionnelle comprise entre 0 et 255 que nous allons envoyer dans la LED grâce à la fonction `analogWrite`.

```
// et stocke le résultat dans outputValue :
outputValue = map(sensorValue, 0, 1023, 0, 255);
```

Nous passons ensuite aux instructions `Serial.print`

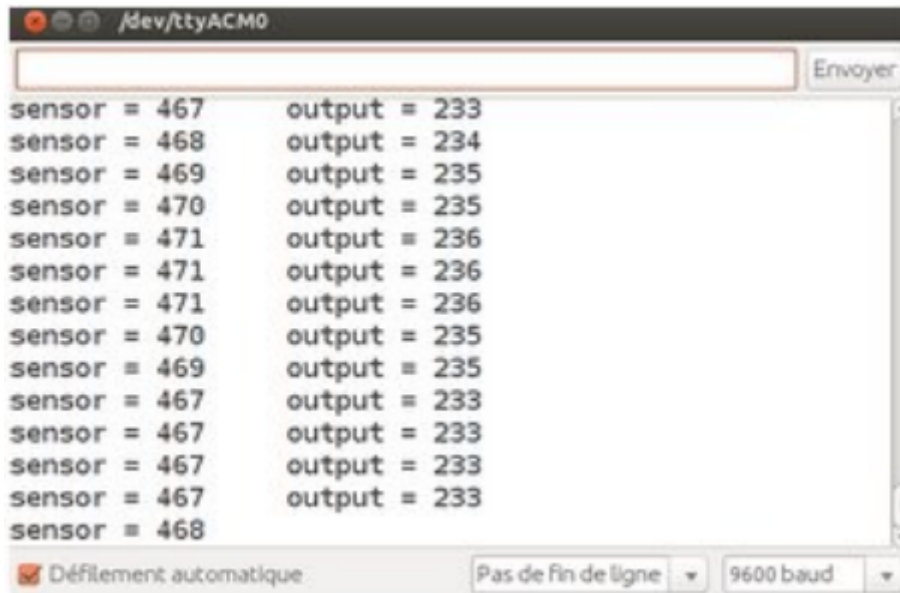
Ces fonctions font suite à l'instruction `Serial.begin` dans le `setup`. Les fonctions `print` et `println` permettent d'envoyer du texte vers l'ordinateur qui pourra ensuite être visualisé dans le moniteur série que nous aborderons dans quelques secondes après avoir téléversé le programme.

```
// envoie tout ça vers l'ordinateur
Serial.print("sensor = ");
Serial.print(sensorValue);
Serial.print("\t output = ");
Serial.println(outputValue);
```

Une fois le code chargé sur l'Arduino, la première chose à faire est de vérifier le bon fonctionnement de notre montage. Pour cela, passer la main sur le capteur de luminosité et regarder

le résultat. Normalement, vous devriez voir baisser la luminosité de la LED lorsque vous cachez la lumière arrivant sur la photorésistance.

Ce changement peut être observé depuis l'ordinateur en utilisant le moniteur série. Une icône située ici permet d'ouvrir une fenêtre où l'on va voir s'afficher les caractères envoyés par les fonctions Serial.print. Selon le niveau de luminosité reçu, nous pouvons voir varier en temps réel les valeurs sensorvalue et outputValue.



The screenshot shows a serial monitor window with the title bar "/dev/ttyACM0". It features a text input field at the top with an "Envoyer" button to its right. The main area displays a list of sensor and output values. At the bottom, there are controls for "Défilement automatique" (checked), "Pas de fin de ligne" (dropdown), and "9600 baud" (dropdown).

sensor	output
467	233
468	234
469	235
470	235
471	236
471	236
471	236
470	235
469	235
467	233
467	233
467	233
467	233
468	