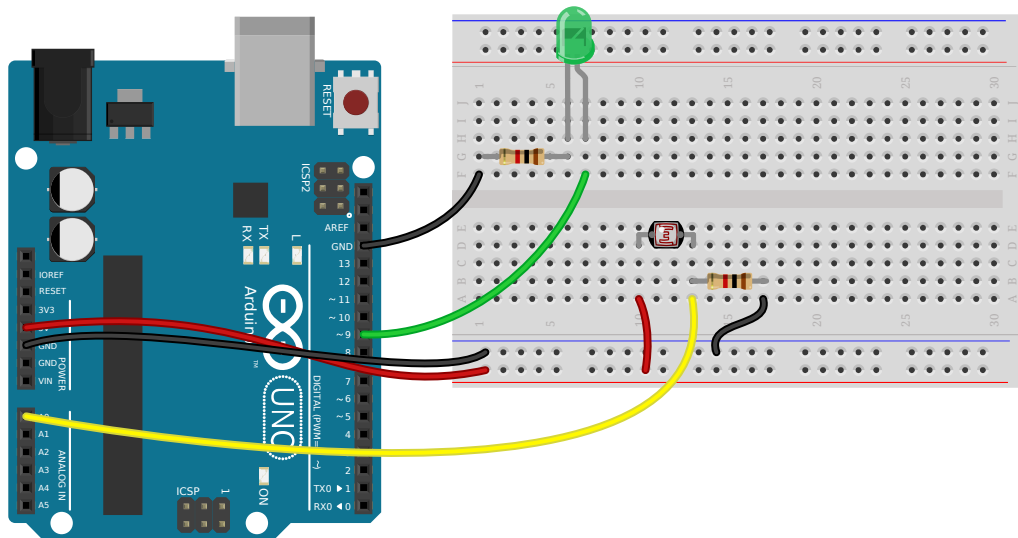
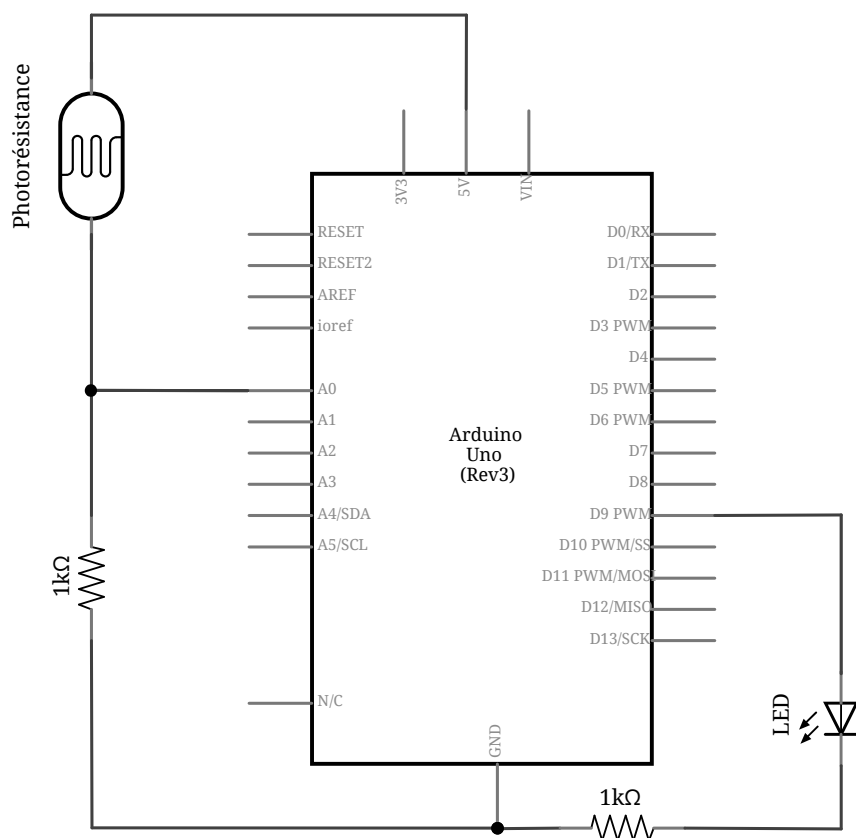


# Lecture Analogique

## Montage



## Schéma



Pour le réaliser, vous aurez besoin de :

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

## Code

Cette suite d'instructions va allumer une LED branchée sur la broche 9. L'intensité lumineuse de la LED sera proportionnelle à la luminosité captée par la photorésistance branchée sur la broche A0 (notez bien le A0, le A qui précède le 0 signifie que c'est une entrée Analogique).

Lorsque vous utilisez le logiciel Arduino, le code peut être trouvé en cliquant sur *Fichier* → *Exemples* → *03.Analog* → *AnalogInOutSerial*.

```
/*
  Entrée et sortie analogiques + communications série

  Ce programme va allumer une LED branchée sur la broche 9.
  L'intensité lumineuse de la LED sera proportionnelle à la luminosité
  captée par la photorésistance branchée sur la broche A0.

  */

// Initialisation des constantes :
const int analogInPin = A0;    // Numéro de la broche à laquelle est connecté la
                                // photorésistance
const int analogOutPin = 9;    // Numéro de la broche à laquelle est connectée la LED

int sensorValue = 0;           // Valeur lue sur la photorésistance
int outputValue = 0;           // Valeur envoyée à la LED

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

  // Indique que la broche analogOutPin est une sortie :
  pinMode(analogOutPin, OUTPUT);
  // Indique que la broche analogInPin est une entrée :
  pinMode(analogInPin, INPUT);
}

void loop()
{
  // lit la valeur de la photorésistance et
  // stocke le résultat dans sensorValue :
  sensorValue = analogRead(analogInPin);
  // change sensorValue vers une intervalle de 0 à 255
  // et stocke le résultat dans outputValue :
```

```
outputValue = map(sensorValue, 0, 1023, 0, 255);
// envoie de cette nouvelle valeur sur la LED
analogWrite(analogOutPin, outputValue);

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

### Remarques :

- Copiez-collez ce code dans le simulateur pour ne pas avoir à tout retaper. Saviez-vous que vous pouvez accéder à la documentation d'une fonction en cliquant avec le bouton droit sur celle-ci puis en cliquant sur *Trouvez dans la référence*.
- Aussi, il faut bien attendre la fin du téléversement avant d'ouvrir le moniteur série dont on parle en dessous.

## Instructions

Voici une description des nouvelles fonctions utilisées (n'hésitez pas à cliquer sur les liens ci-dessous afin d'arriver sur [la référence Arduino](#)).

- [analogRead](#) permet de lire l'état d'une broche analogique et de renvoyer une valeur numérique proportionnelle à la tension reçue. La carte Arduino comporte 6 voies, A0 à A5, connectées à un convertisseur analogique-numérique 10 bits. Cela signifie qu'il est possible de transformer la tension d'entrée entre 0 et 5V en une valeur numérique entière comprise entre 0 et 1023.

```
analogRead(analogInPin);
```

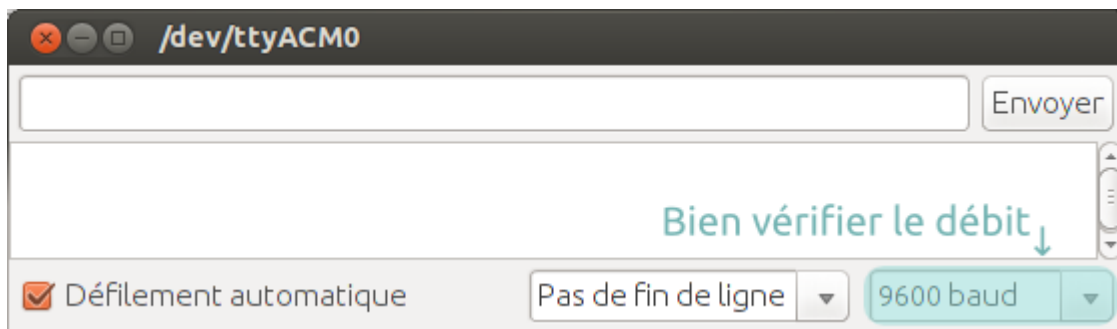
La valeur de retour de [analogRead](#) peut être stockée dans une variable entière (c'est pour cela que nous devons déclarer une variable de type [int](#) pour stocker le résultat) :

```
sensorValue = analogRead(analogInPin);
```

- [Serial](#) est une bibliothèque (un ensemble de fonctions) utilisée pour les communications par le port série entre la carte Arduino et un ordinateur ou d'autres composants. Ce port série permet l'envoi et la réception de suites de caractères sur les broches 0 (RX) et 1 (TX) avec l'ordinateur via le port USB. C'est pourquoi, si vous utilisez cette fonctionnalité, vous ne pouvez utiliser les broches 0 et 1 en tant qu'entrées ou sorties numériques. Si vous souhaitez visualiser le texte envoyé depuis l'Arduino vers votre ordinateur, vous pouvez utiliser le terminal série intégré à l'environnement Arduino. Il suffit pour cela de cliquer sur le bouton du moniteur série dans la barre d'outils (**Rappel** : il faut bien attendre la fin du téléversement avant d'ouvrir le moniteur série) :



Dans la fenêtre qui s'ouvre, vérifier que vous êtes au même débit de communication que celui utilisé dans l'appel de la fonction `Serial.begin`. Par défaut le débit est de 9600 :



Les fonctions de la librairie `Serial` sont :

- `Serial.begin` permet d'initialiser la communication entre Arduino et votre ordinateur. Cette fonction doit être placée dans le bloc `setup`, elle doit être suivie d'un seul paramètre qui correspond au débit de communication en nombre de caractères échangés par seconde (l'unité est le baud) pour la communication série. De manière classique, nous choisirons de communiquer à 9600 bauds. Sans le `Serial.begin` dans le bloc `setup`, on ne peut utiliser les autres fonctions `Serial.print` et `Serial.println`.

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

- `Serial.print` Cette fonction permet d'envoyer sur le port série une suite de caractères indiqués en paramètre. De la même façon, `Serial.println` envoie une suite de caractères suivie d'un retour à la ligne :

```
Serial.print("Texte envoyé vers l'ordinateur"); // sans retour à la ligne
Serial.println("Texte avec retour à la ligne"); // avec retour à la ligne
```

`Serial.print` nous sera particulièrement utile pour une tâche que vous avez déjà faite sans vous en rendre compte : **le débogage**. Il n'est pas rare que vous téléversiez votre code sur Arduino sans problème, mais une fois sur Arduino le comportement de votre programme n'est pas celui attendu. Grâce à `Serial`, il nous sera possible d'indiquer quand nous allumons une LED ou lorsque nous faisons un test... Il nous sera ainsi possible de suivre le déroulement de notre programme ! Mais ce n'est pas tout, `Serial.print` et `Serial.println` peuvent également afficher la valeur de variables si nous indiquons un nom de variable en paramètre :

```
Serial.print("sensor = ");
Serial.print(sensorValue);
```

- [map](#) permet de faire passer une valeur située dans une intervalle vers un autre. Les paramètres de ces fonctions sont les suivants :
  - variable qui se trouve dans l'intervalle initial
  - début de l'intervalle initial
  - fin de l'intervalle initial
  - début de l'intervalle visé

Grâce à cette fonction, nous allons donc nous retrouver avec une valeur proportionnelle

```
map(sensorValue, 0, 1023, 0, 255); // sensorValue passe de l'intervalle 0→1023  
vers 0→255
```

- [analogWrite](#) va être utilisé dans ce programme pour moduler l'intensité lumineuse d'une LED branchée sur la broche spécifiée avec le premier paramètre. L'intérêt de [analogWrite](#) avec est de pouvoir régler l'intensité lumineuse en spécifiant un nombre (compris entre 0 et 255) dans le second paramètre de la fonction :

```
analogWrite(11, 0); // éteint complètement la LED branché sur la broche 11  
analogWrite(11, 90); // allume un tout petit peu la LED branché sur la broche 11  
analogWrite(11, 255); // allume complètement la LED branché sur la broche 11
```

## Références

- [Référence standard du langage Arduino](#) par Xavier Hinault
- [Chapitre dédié aux capteurs du manuel Arduino](#) de chez FlossManuals
- [Compléments sur les capteurs](#) sur le wiki des petits débrouillards