

Introduction à l'Arduino

Table des matières

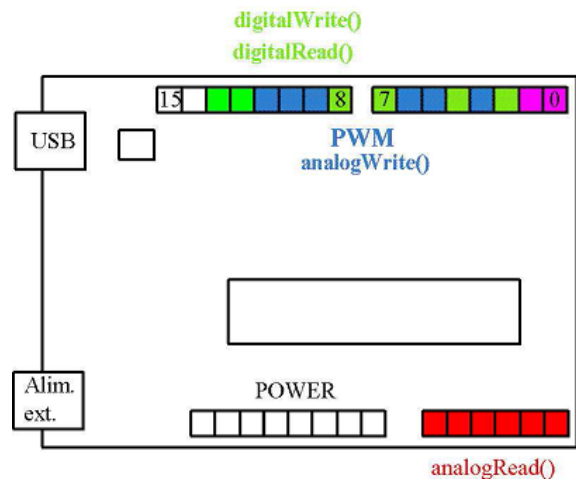
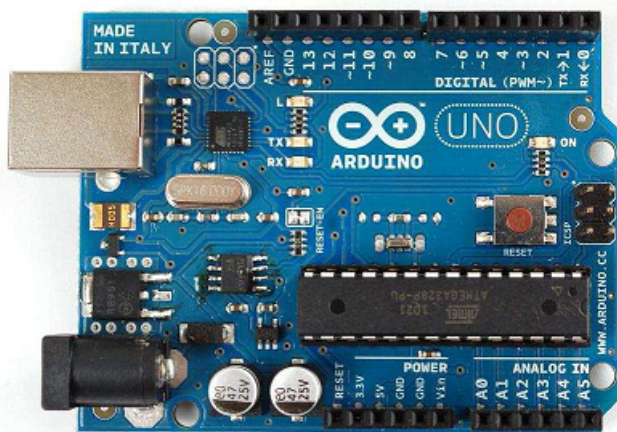
1. Installation et configuration de la carte Arduino.....	2
1.1. Carte Arduino.....	2
1.2. Interface de développement.....	2
1.3. Structure d'un programme.....	3
2. Utilisation de la sortie numérique.....	4
2.1. Clignotement d'une LED.....	4
2.2. Clignotement de 2 LED.....	5
3. Utilisation de l'entrée numérique.....	6
3.1. Détection d'un Bouton Poussoir.....	6
4. Utilisation de l'entrée analogique.....	7
4.1. Mesure de l'éclairement.....	7
4.3. Simulation store électrique.....	9
5. Références.....	10



1. Installation et configuration de la carte Arduino

1.1. Carte Arduino

Nous utiliserons une carte Arduino Uno. Elle emploie un microcontrôleur ATMEGA328P alimenté en 5 V. Il y a 14 entrées/sorties numériques. Il y a 6 entrées analogiques. Le microcontrôleur possède un CAN¹ avec 10 bits de résolution. Sur la carte, il y a un circuit qui permet de gérer facilement l'USB² qui peut alimenter la carte.




- Courant max. E/S40 mA

Pour en savoir plus, consultez la page : <http://arduino.cc/en/Main/ArduinoBoardUno>

1.2. Interface de développement

Pour utiliser l'interface de développement :

1. Lancer le logiciel Arduino .
2. Vérifier que la carte Arduino Uno est bien prise en compte par : Outils > Type de carte > Arduino Uno.
3. Brancher la carte sur un port USB. Vérifier que le port COM est bien configuré. Outils > Port série > COM X.
4. Vérifier que le port COM X est bien reconnu par windows : Panneau de Configuration > Système et sécurité > Gestionnaire de périphériques > Ports : Arduino Uno (COM X).

¹ Convertisseur Analogique Numérique

² Universal Serial Bus

1.3. Structure d'un programme

Un exemple de programme est donné ci-dessous.

```

/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
*/

const int led = 13;

// the setup function runs once when you press reset or power the board
void setup()
{
  // initialize digital pin 13 as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}

```

- Le commentaire de description du programme, appelé sketch, débute par `/*` et se termine par `*/`.
- Puis il est placé la définition des constantes et des variables avec les instructions `const` et `int`.
- Vient ensuite la configuration des entrées / sorties avec l'instruction `void setup()`. La suite d'instructions est précédé de `{` et se termine par `}`.
- On définit par l'instruction `void loop()` la programmation des interactions et comportements. La suite d'instructions est précédé de `{` et se termine par `}`.

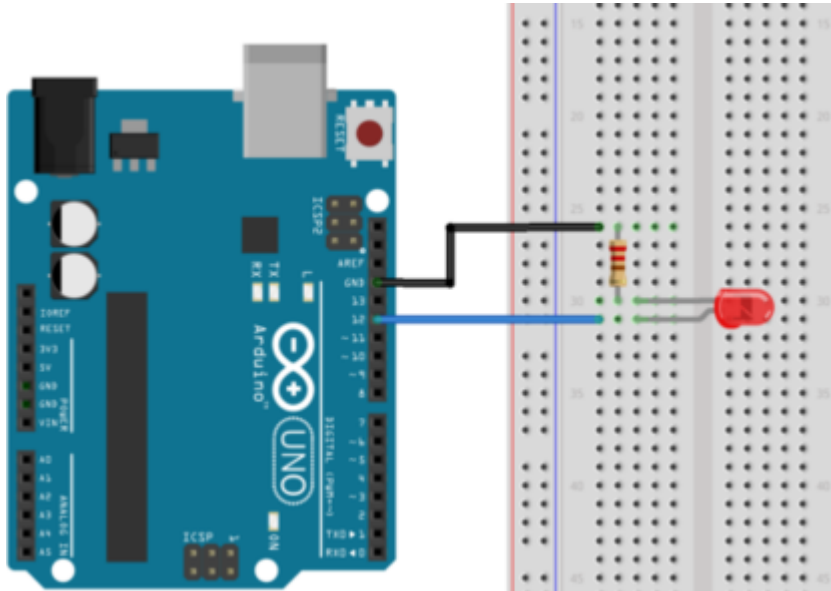
2. Utilisation de la sortie numérique

2.1. Clignotement d'une LED

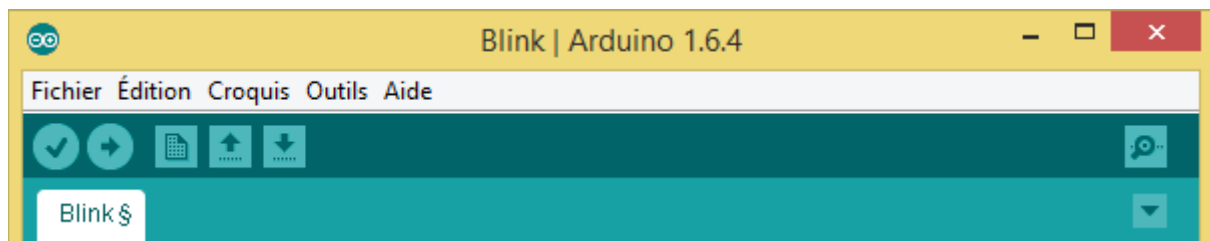
Ouvrir le fichier Blink par : Fichier > Exemples > 01.Basics > Blink

Pour comprendre la syntaxe et le rôle d'une instruction, utiliser la page : <http://arduino.cc/en/Reference/HomePage>

1. Réaliser le schéma de montage ci-dessous (faire attention au [sens de polarisation de la LED](#)).



2. Modifier le programme de telle façon que la sortie sur laquelle on connectera la LED soit la sortie 12.
3. Modifier les instructions pour que la période soit de 1 s et la durée d'allumage de la LED 100 ms.
4. Lancer l'exécution du programme en cliquant sur le bouton en dessous Édition :



```

/*
  clignotement diode
*/

const int led = 12; // broche 12 pour la LED

// cette fonction ne s'exécute qu'une seule fois
void setup()
{
  pinMode(led, OUTPUT);          // on définit la broche en sortie
}

// cette fonction boucle indéfiniment
void loop()
{
  digitalWrite(led, HIGH); // led allumée
  delay(100);              // allumé 100ms
  digitalWrite(led, LOW);  // led éteinte
  delay(100);              // éteint 100ms
}

```

Dans ce programme, nous avons :

1. **Des commentaires :**

qui sont des lignes de texte incluses dans le programme et qui ont pour but de vous aider à comprendre (ou à vous rappeler) comment votre programme fonctionne ou d'en informer les autres. Ces lignes ne sont pas envoyées à Arduino. Il y a deux façons de créer des lignes de [commentaires](#) :

```
/*
Voici des
commentaires sur
plusieurs ligne
*/

// Ceci est également un commentaire
```

2. Des instructions :

- **Déclaration d'une [variable](#)** : on vient avec cette ligne stocker la valeur à droite du signe égal dans la [variable](#) à gauche du signe égal.

```
Const int led = 13;
```

Dans notre cas, cela signifie que la [variable](#) appelée `led` qui sera un nombre (puisque elle est précédée du mot clé [int](#)) viendra prendre la valeur 13.

- **Les blocs d'instructions** : [setup](#) regroupe toutes les instructions qui seront exécutées au démarrage du programme. La fonction [setup](#) n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino. [loop](#) (boucle en anglais) contient les instructions que l'on souhaite voir exécutées encore et encore tant que l'Arduino est branché.

```
void setup() { }
void loop() { }
```

3. Les **fonctions** : sont des instructions qui permettent d'exécuter une ou plusieurs actions. Les fonctions sont définies avec :
- **Un nom** : ce qu'on devra taper pour appeler la fonction.
 - **Une ou des entrées** : ce sont des variables passées à la fonction appelées **paramètres** ou **arguments**. Ces arguments sont placés entre parenthèses.
 - **Une sortie** : le résultat de la fonction qui peut être stocké dans une variable.

Prenons l'exemple de la fonction suivante :

```
digitalWrite(led, HIGH);
```

Dans ce cas, le nom de la fonction est [digitalWrite](#). Nous passons deux paramètres à la fonction : `led` et `HIGH`. La fonction [digitalWrite](#) n'a pas de sortie. Avec cette fonction, nous allumons la broche située sur la broche passée avec le premier paramètre (qui peut être un nombre ou une variable). Lorsque le second argument est placé à `HIGH`, on vient allumer la LED. Tandis qu'on éteindra la LED si le second argument passé est `LOW`.

Les autres fonctions présentes dans le programme *Blink* sont :

- [pinMode](#) configure la broche spécifiée dans le premier paramètre pour qu'elle se comporte soit en entrée (`INPUT`), soit en sortie (`OUTPUT`) passée avec le second paramètre :

```
pinMode(led, OUTPUT);
```

- [delay](#) fait une pause dans l'exécution du programme pour la durée (en millisecondes) passée en paramètre :

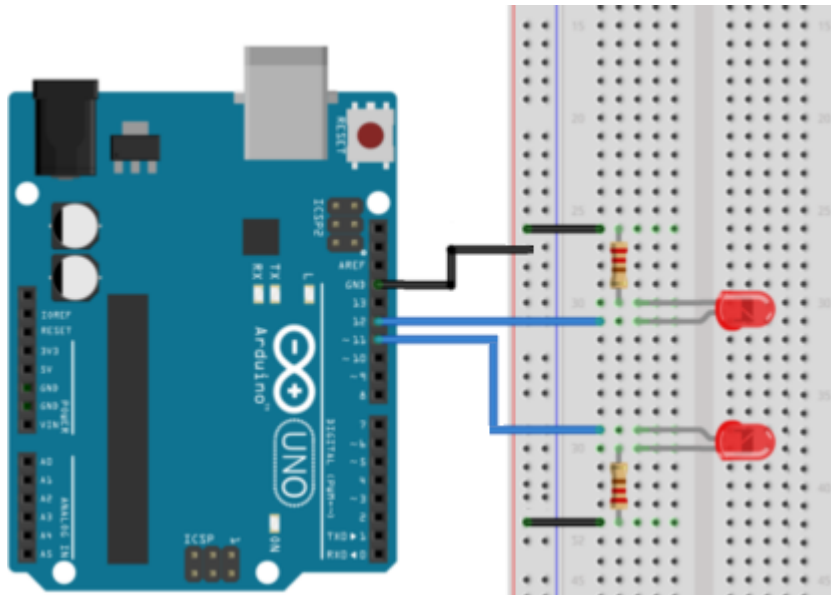
```
delay(1000);
```

2.2. Clignotement de 2 LED

1. Modifier le programme pour que deux LED s'allument en opposition de phase comme dans le tableau suivant :

LED 1	Broche 11	Allumée	Éteinte
LED 2	Broche 12	Éteinte	Allumée

2. Modifier le schéma de montage comme indiqué ci-dessous :



Broche 11 : module LED1

Broche 12 : module LED2

```

/*
   clignotement 2 diodes alternées
 */

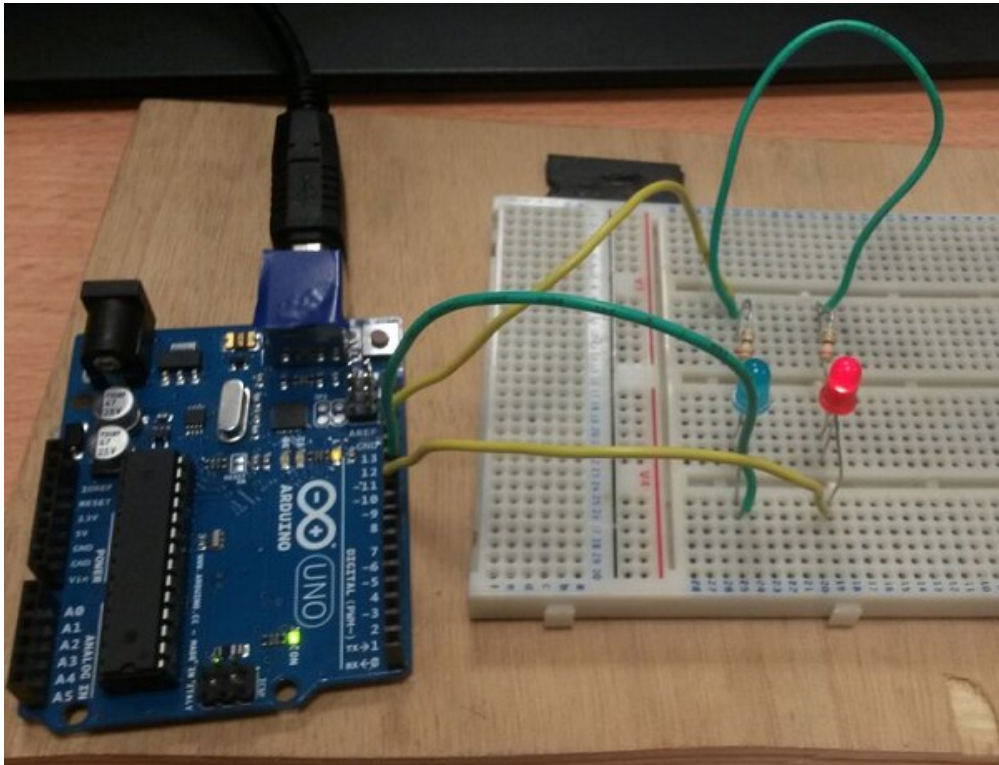
const int led1 = 11;      // broche 11 pour la LED 1
const int led2 = 12;      // broche 12 pour la LED 2

// cette fonction ne s'exécute qu'une seule fois
void setup()
{
  pinMode(led1, OUTPUT);  // on définit la LED 1 en sortie
  pinMode(led2, OUTPUT);  // on définit la LED 2 en sortie
}

// cette fonction boucle indéfiniment
void loop()
{
  // à compléter
  // ...
  delay(1000);             // durée 1s

  // à compléter
  // ...
  delay(1000);             // durée 1s
}

```



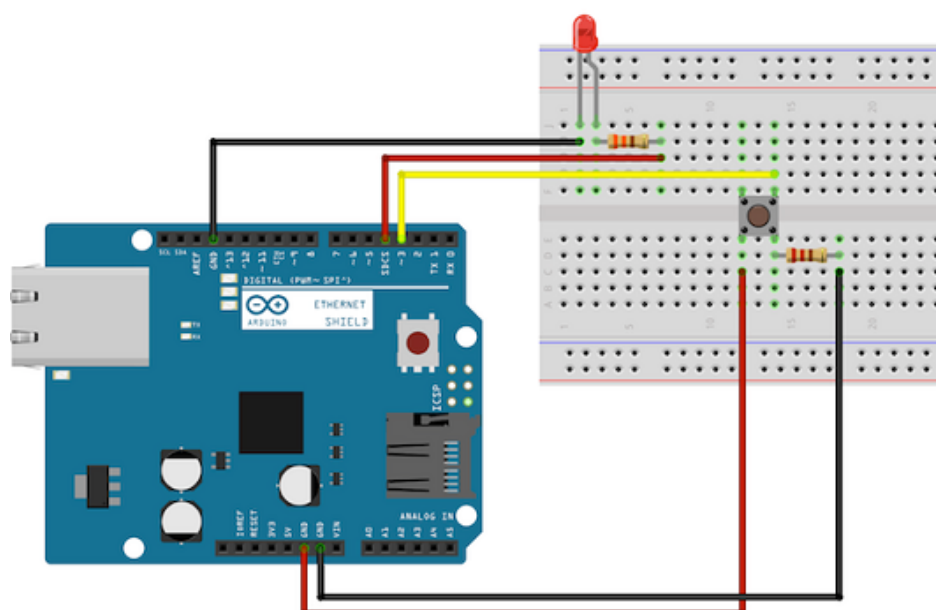
Pour voir le résultat, cliquer sur l'image

3. Utilisation de l'entrée numérique

3.1. Détection d'un Bouton Poussoir

1. Réaliser le schéma de montage ci-dessous (faire attention au [sens de polarisation de la LED](#)).

On utilisera un bouton poussoir (BP) qui commandera une LED à travers la carte Arduino.



Broche 4 : module LED
Broche 3 : module BP

2. Modifier ce programme pour que la carte Arduino commande la LED lors de la pression du bouton.


```

/*
  détection bouton poussoir
*/

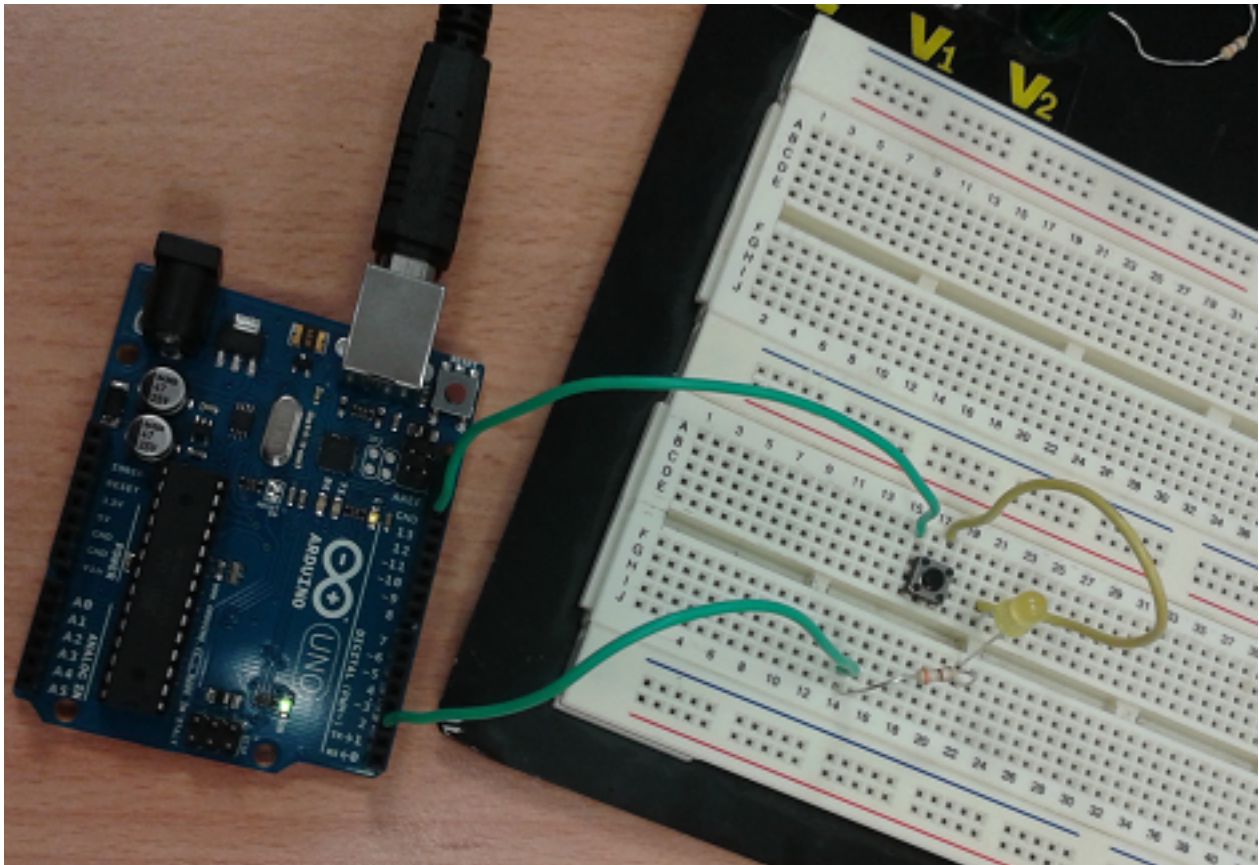
const int led = ...;          // broche 4 pour la LED
const int button = 3;         // broche 3 pour le bouton poussoir

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(led, ...);          // on définit la led en sortie
  pinMode(button, ...);       // on définit le bouton en entrée
}

// the loop function runs over and over again forever
void loop()
{
  // lecture bouton
  if ( digitalRead(button) == ... )
    digitalWrite(led, ...);   // led allumée
  else
    digitalWrite(led, ...);   // led éteinte

  delay(...); // stabilisation 200 ms
}

```



Pour voir le résultat, cliquer sur l'image

4. Utilisation de l'entrée analogique

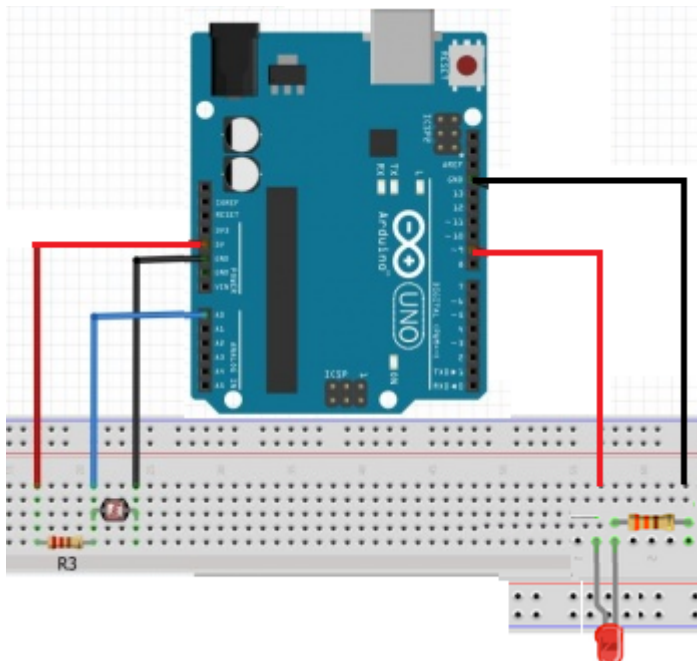
4.1. Mesure de l'éclairement

Le capteur de luminosité utilisé est le LDR VT935G (ou équivalent série VT900).

Par effet photovoltaïque, l'énergie rayonnante est transformée en énergie électrique.

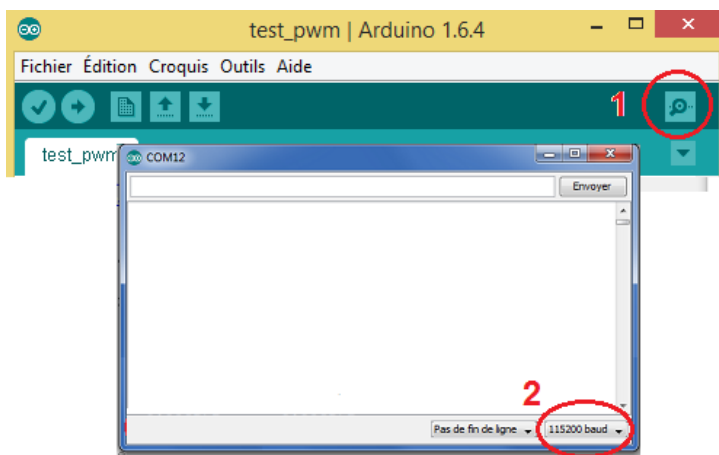
La résistance interne du capteur est inversement proportionnelle avec la luminosité sur une échelle logarithmique.

1. Réaliser le schéma de montage ci-dessous :



Broche 9 : module LED
Broche A0 : module capteur luminosité

2. Visualiser la luminosité mesurée.



1. ouvrir un terminal
2. Régler la vitesse de communication à 9600 bauds
3. Obturer le détecteur
4. Relever les valeurs :
- avec lumière
- sans lumière
5. En déduire la valeur de seuil

3. Modifier le programme ci-dessous pour régler le seuil de détection qui allumera LED.

```
/*
  détection luminosité
*/

const int led = 9;           // broche 9 pour la LED
const int lux = ...;         // broche analogique A0 pour le capteur

// cette fonction ne s'exécute qu'une seule fois
void setup()
{
  pinMode(...);             // on définit la broche 9 en sortie
  pinMode(...);             // on définit la broche A0 en entrée
  Serial.begin(9600);         // initialise la vitesse du port série
}

// cette fonction boucle indéfiniment
```

```

void loop()
{
  int value = analogRead(...);    // lecture capteur
  Serial.println(value);          // affichage valeur sur terminal série

  if ( value < 0 )                // détection du seuil de luminosité
    digitalWrite(led, ...);       // led allumée
  else
    digitalWrite(led, ...);       // led éteinte

  delay(10);                      // stabilisation 10 ms
}

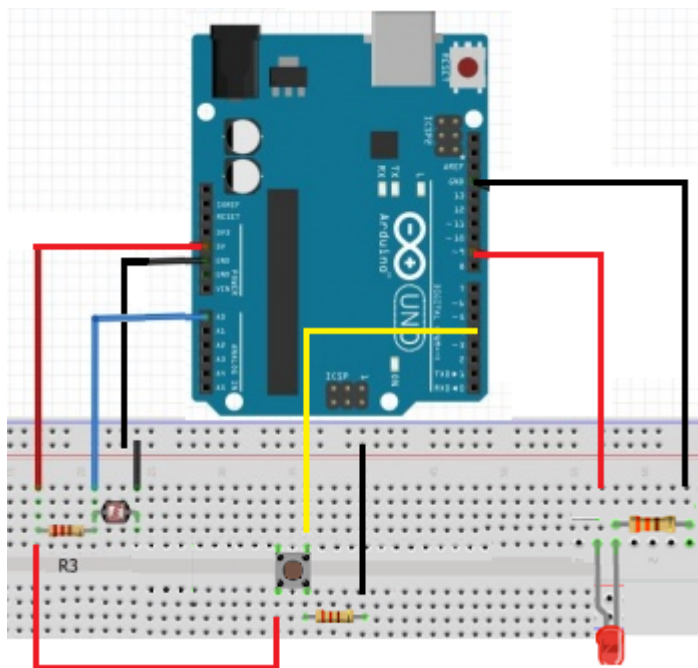
```

4.3. Simulation store électrique

On va simuler le fonctionnement d'un store électrique automatique.

A partir d'un certain seuil de luminosité, le programme déclenche un moteur (LED) qui devra s'arrêter lorsqu'il arrive en buté (BP).

1. Réaliser le schéma de montage ci-dessous :



Broche 4 : module BP
 Broche 9 : module LED
 Broche A0 : module capteur luminosité

2. Modifier le programme de la mesure de l'éclairement pour que la LED s'éteigne quand le BP est enfoncé en vous aidant du tableau ci-dessous.

Seuil atteint	BP enfoncé	LED
oui	oui	OFF
oui	non	ON
non	oui	OFF
non	non	OFF

```

/*
  simulation store électrique
*/

const int BP = ...;    // broche 4 pour le BP
const int led = ...;   // broche 9 pour la LED
const int lux = ...;   // broche analogique A0 pour le capteur

// cette fonction ne s'exécute qu'une seule fois

```

```
void setup()
{
    ...;          // on définit la broche 4 en entrée
    ...;          // on définit la broche 9 en sortie
    ...;          // on définit la broche A0 en entrée

    Serial.begin(9600);      // initialise la vitesse du port série
}

// cette fonction boucle indéfiniment
void loop()
{
    int bouton = ...;  // lecture BP
    int value = ...;   // lecture capteur
    Serial.println(value);      // affichage valeur sur terminal série

    if ( ... && ... ) { // détection du seul de luminosité
        ...;           // led allumée
    } else
    {
        ...;           // led éteinte
    }

    delay(10);         // stabilisation 10 ms
}
```

5. Références

- [Tutoriel Arduino](#)
- [Manuel de référence Aduino](#)
- [langage C](#)
- [langage C++](#)
- [la résistance](#)
- [la diode](#)
- [Référence Langage Arduino](#)