Interaction Homme Machine - Microcontrôleurs

Introduction

Une interface Homme Machine est une interface permettant d'**interagir avec une machine**. Elle peut se faire à l'aide de **microcontrôleurs**.

Le principe de base des microcontrôleurs repose sur l'inclusion dans le même boîtier du microprocesseur et de divers périphériques, de manière à avoir **un composant autonome**.

Le microcontrôleurs sont des **circuits intégrés** qui **rassemblent les éléments essentiels** d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties.

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de **diminuer la taille**, **la consommation électrique** et le **coût des produits**. Par contre, ils ont une une vitesse de fonctionnement plus faible.

Les microcontrôleurs sont fréquemment utilisés dans les **systèmes embarqués**, pour effectuer une tâche particulière en temps réel, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc. Ces systèmes sont constitués d'une **carte programmable**, de **capteurs** et d'**actionneurs**.

Ci-dessous, deux exemples de cartes avec microcontrôleurs, connectiques d'entrées/sorties et alimentation :

Les **capteurs** permettent de connaître l'environnement, **envoient des données à la carte**, réagissent en fonction d'un phénomène physique (la température, la luminosité...).

Les **actionneurs** produisent un phénomène physique (lumière, son...), ils agissent sur un système pour en **modifier le comportement**.

Présentation d'Arduino

Nous allons travailler avec une **carte arduino uno**.

L'avantage de cette carte est que le **matériel est libre**, et qu'il y a beaucoup d'utilisateurs de part le monde. Ainsi on trouve énormément de documentation sur son utilisation.

Arduino propose toute une gamme de cartes, suivant l'usage que l'on veut en faire, ainsi que tout le matériel nécessaire pour créer toute sortes de systèmes permettant par exemple de contrôler des éclairages, de piloter des robots, de créer un détecteur de mouvement ou une station météo...

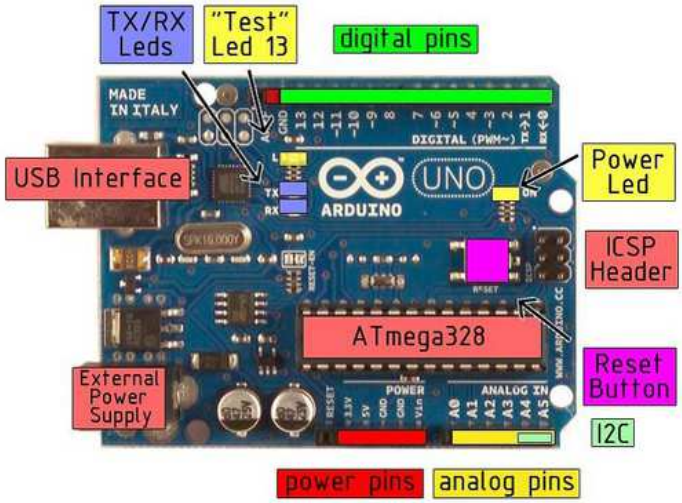
Pour travailler il faut donc le **matériel** (hardware) et un **logiciel** (le software).

★ Nous utiliserons un simulateur pour le matériel, celui du site :

https ://www.tinkercard.com

Vous pouvez si vous le souhaitez, vous procurer une carte pour essayer de faire les expériences vous-même ; on trouve facilement des kits de démarrage complets à bas prix. Le matériel arduino est lui-même un peu cher, mais on peut trouver d'autres marques qui fournissent le même matériel comme Quimat ou Elegoo à un prix plus bas.

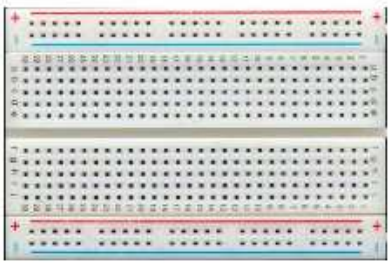
Dans un premier temps on va découvrir la **carte Arduino**, avec le schéma ci-dessous :

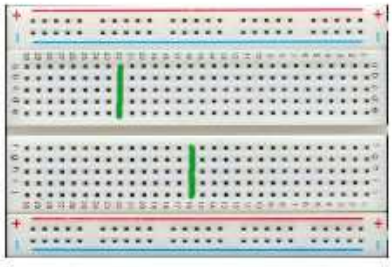


Il y a entre autre :

- une **prise USB** afin de brancher la carte à l'ordinateur ;
- une prise pour l'**alimentation** extérieur (pile) ;
- le **microcontrôleur**, ici l'ATmega328 ;
- des **broches digitales** d'entrées/sorties (pins) qui peuvent émettre ou recevoir une valeur numérique binaire et des **broches analogiques** qui peuvent émettre une valeur numérique entière comprise entre 0 et 1023.
- des **Leds** de contrôle : TX/RX (clignotent lorsque la carte communique avec l'ordinateur), Test (permet de faire des tests sans autre matériel) et Power (s'allume quand la carte est alimentée)

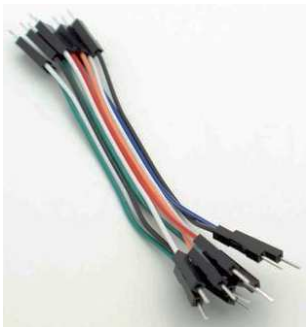
Pour pouvoir tester différents systèmes, on utilise une **Breadboad** :



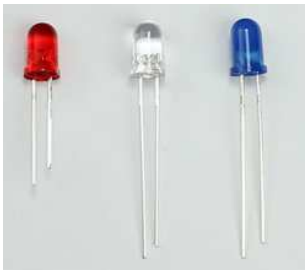


On placera les composants dans les trous, ce qui évitera d'avoir à les souder. En effet, ceux-ci sont reliés entre eux, mais attention, pas de n'importe quelle façon ! Sur la photo de droite, on voit que pour la partie centrale, les trous d'une même colonne sont reliés entre eux ; de plus les trous des lignes + d'une part et - d'autre part sont également reliées.

Ensuite il faut des **fils de connexion** ou **jumpers** :



Enfin, on peut utiliser des composants très divers tels que des **leds**, des **résistances**, des **boutons pressoir**, des **potentiomètre**, des **servo-moteurs**, des **capteurs** de température, de luminosité...



★ Le **logiciel** s'appelle également **Arduino**. C'est un **IDE** (environnement de développement) permettant d'écrire un programme dans un langage basé sur le langage de programmation C. Celui-ci est ensuite **compilé** (traduit de façon à être compris par la carte), puis **envoyé à la carte**, afin qu'elle puisse l'exécuter.

La carte exécutera le programme **tant qu'elle sera alimentée** en électricité.

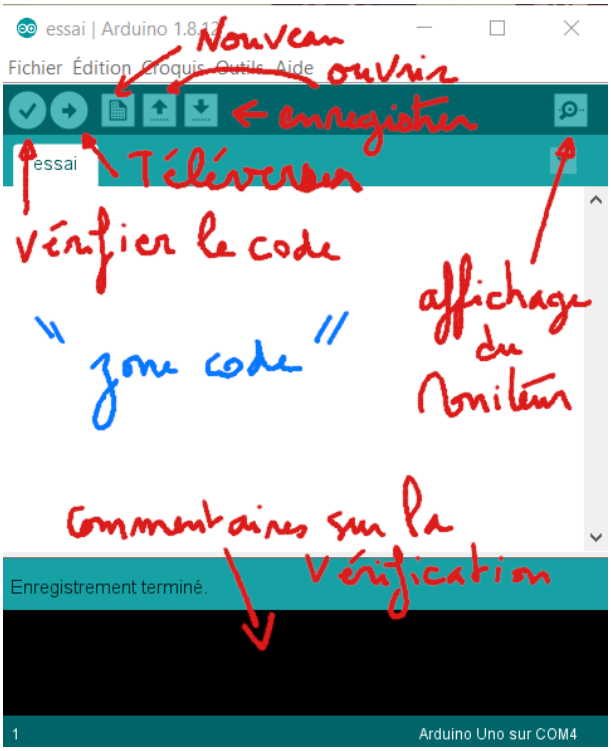
Vous trouverez le logiciel **Arduino** sur le site officiel :

https ://www.arduino.cc/en/Main/Software

TP 1 : Découverte de la Lumière

1. **Télécharger l'IDE Arduino** correspondant à votre système d'exploitation, puis installez-le.
2. Lancez le logiciel :

   Si vous voulez utiliser une vraie carte, il vous faudra **indiquer son type** dans le menu : *Outils/Type de carte*. De plus, il faudra indiquer sur quel **port de communication** vous brancherez la carte dans le menu : *Outils/Choix du port*.



Les icônes :

Vérifier : lorsque l'on a tapé le code, le logiciel vérifie qu'il est fonctionnel.

Téléverser : c'est l'action d'envoyer le code à la carte.

Nouveau : ouvre une nouvelle fenêtre.

Ouvrir : permet d'ouvrir un ancien code ou les exemples Arduino.

Enregistrer : permet de sauvegarder le code.

La loupe : permet d'ouvrir un moniteur où l'on peut lire les données reçues par la carte.

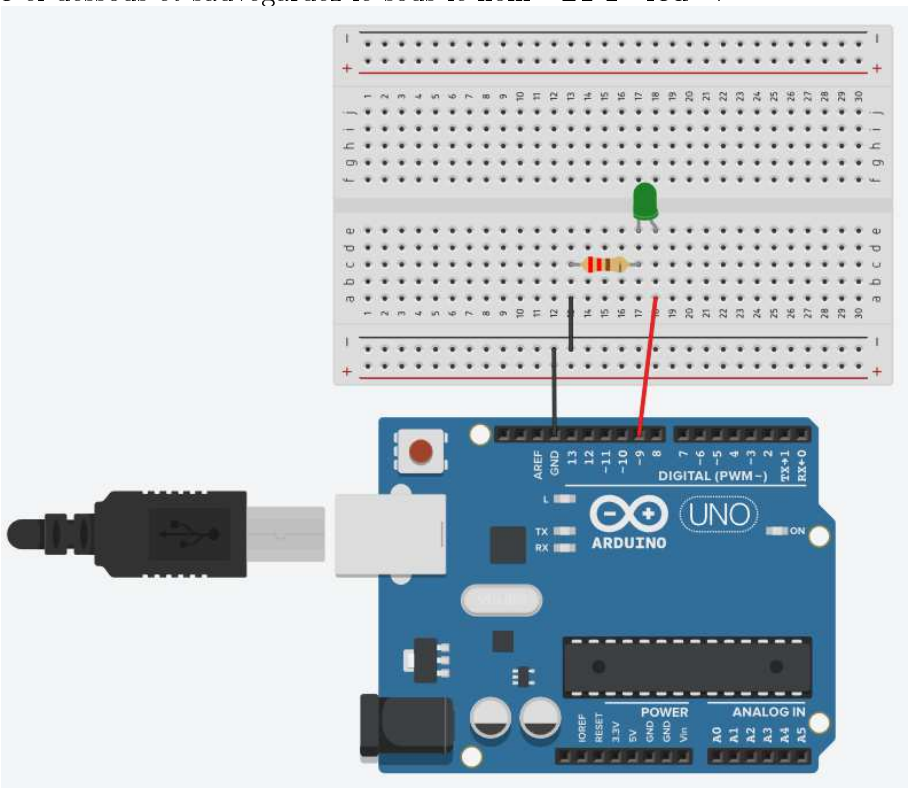
3. **Recopiez le code** suivant, sans les commentaires signalés par le symbole //. Sauvegardez-le et vérifiez-le.

```
// On stocke la valeur 9 dans la variable "led", pour indiquer que la
   led se trouve sur la broche 9 (pin)
// La variable "led" est une contante : c''est un nombre entier (int)
   égal à 9
const int led = 9;

// la fonction setup() est exécutée lorsque l'on alimente la carte ou
   que le bouton reset est pressé
void setup() {
  // indique que la broche de la led 9 est une sortie
  pinMode(led, OUTPUT);
}

// le code dans cette fonction est exécuté en boucle tant que la carte
   est alimentée
void loop() {
  digitalWrite(led, HIGH); // allumer la led (tension 5V sur la broche)
  delay(1000);             // attendre 1000ms = 1s
  digitalWrite(led, LOW);  // éteindre la LED (tension 0V sur la broche)
  delay(1000);             // attendre à nouveau 1s
}
```

4. Rendez-vous sur le site de simulation **tinkercard**. Pour cela, je vous donne accès à une classe où nous pourrons partager nos travaux, grâce au lien ci-dessous. Vous devrez juste utiliser le pseudo que je vous ai communiqué pour y accéder.

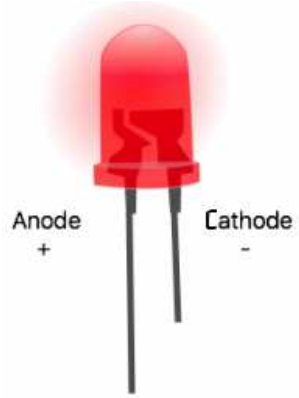
   https ://www.tinkercad.com/joinclass/A4GN54UC2BSD

5. Faites **le montage** ci-dessous et sauvegardez-le sous le nom **"TP1 led"** :



Remarques :

- il y a un sens pour **brancher les leds** :



la **broche GND** représente la terre (0 Volt), elle est relié au moins (-) de la led ; le plus (+) est relié à la **broche** 9 ;

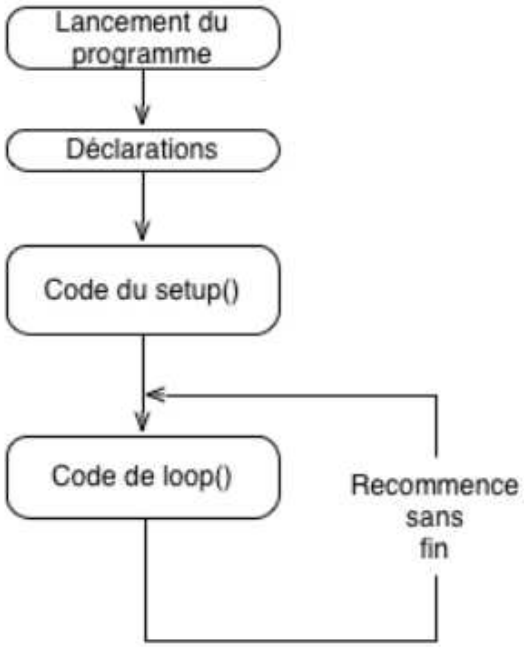
- on insère une **résistance** pour ne pas griller la led ; entre 100 et 1000 Ω = 1 $k\Omega$ (ohms) pour notre led (plus la résistance est élevée et moins l'intensité de la lampe est forte) ;

- la vitesse d'exécution du programme est trop rapide pour être perçue à l'oeil nu, c'est pour cela que l'on utilise la fonction **delay()**, qui permet de faire une pause dans l'exécution des instructions du code.

6. Collez le code précédent dans le simulateur et **lancez la simulation**.

   Remarque : Si rien ne se passe, vérifiez vos branchements !

7. **Structure de base d'un programme Arduino** :



On commence par **déclarer les variables**, en spécifiant leur **type**, puis dans la fonction **setup()**, on indique si ce sont des **entrées** (INPUT) ou des **sorties** (OUTPUT), on les **initialise**...

Les **commentaires** s'écrivent avec le symbole //.

TP 2 : Guirlande de led

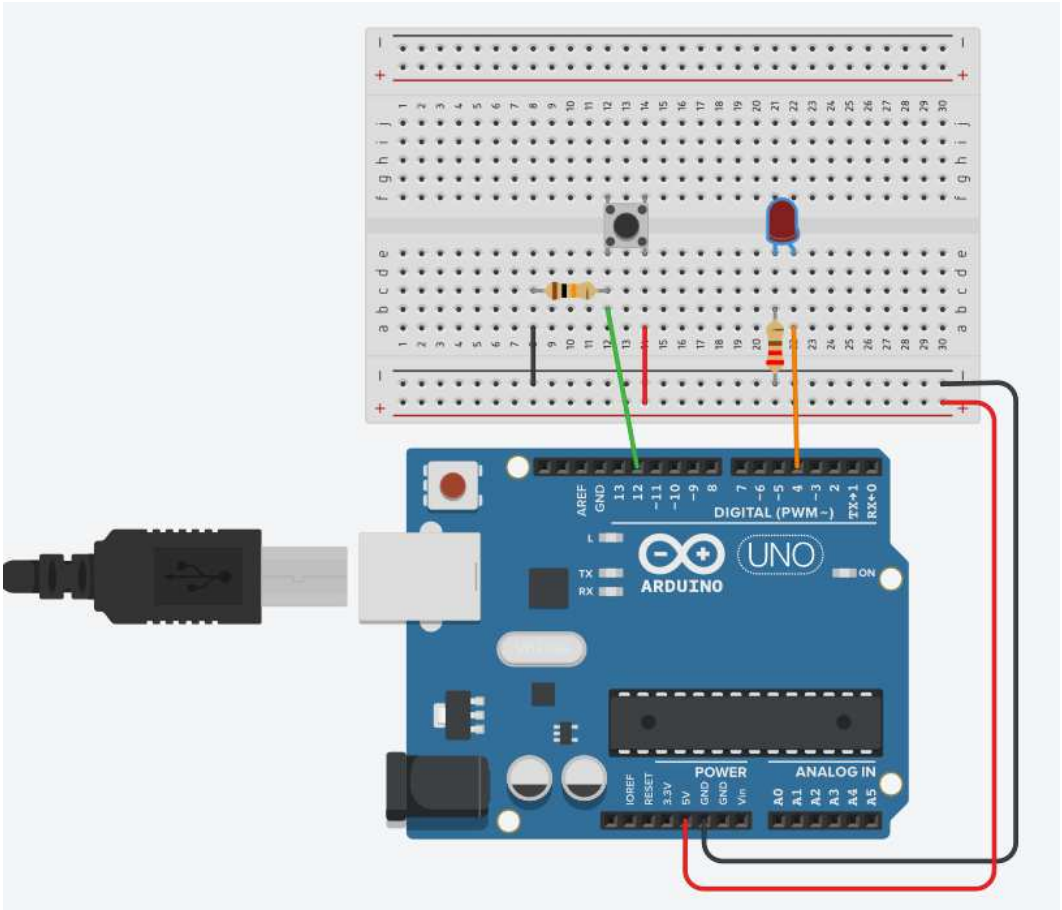
Á vous maintenant !

1. Vous devez modifier le montage précédent pour mettre 5 leds de différentes couleurs en ligne.
   Vous les connecterez sur les broches de 7 à 11.
2. Maintenant, écrivez un premier programme pour que les leds s'allument, les unes après les autres (1 seule restant allumée à la fois).
3. Modifiez votre programme, pour que les leds s'allument toutes les unes après les autres, et restent allumées ; puis, lorsque les 5 leds sont allumées, elles devront s'éteindre l'une après l'autre.

TP 3 : Interrupteur

Nous allons utiliser un bouton pressoir, afin d'avoir un interrupteur pour allumer une led.

1. Reproduisez le montage ci-dessous :



Remarques :

- le bouton est un dipôle, les pattes sont reliées deux par deux
- on utilise une résistance de 10 $k\Omega$ pour ce bouton.

2. Copiez le code suivant :

```
const int led = 4;        //on branche a led sur la broche 4
const int bouton = 12;    //on branche le bouton sur la broche 12
boolean etatBouton = 0; //on crée une variable booléenne pour les deux
   état du bouton : pressé (0) ou relâché (1)

void setup() {
  //indique que la broche de la led est une sortie
  pinMode(led, OUTPUT);
  //indique que la broche du boutton est une entrée
  pinMode(bouton, INPUT);
  //au départ la led est éteinte
  digitalWrite(led, LOW);
}

void loop() {
  //on lit l'état du bouton (préssé ou ralâché)
  etatBouton = digitalRead(bouton);
  if  (etatBouton == 1){    //si le bouton est pressé on allume la led
   pendant une seconde, puis on l'éteint
     digitalWrite(led, HIGH);
     delay(1000);
     digitalWrite(led, LOW);
  }
  delay(1000);              //on laisse un délais d'une seconde pour avoir
    le temps de relâcher le bouton
}
```

3. Lancez la simulation, appuyez sur le bouton !

   Remarques :

   - par moment, il faut appuyer un peu plus longtemps pour déclencher l'allumage ;
   - la syntaxe de la condition (si, alors, sinon) :

4. On peut utiliser le **moniteur** pour **afficher la valeur de la variable etatBouton**, en ajoutant les lignes suivantes au code :

```
void setup() {
  //on initialise la communication
  Serial.begin(9600);
}

void loop() {
  //après avoir lu la valeur de la variable, on l'affiche sur le
   moniteur
  Serial.println(etatBouton);
}
```

Après avoir ajouté ces lignes à votre programme, lancez la simulation, en laissant la fenêtre **code** ouverte ;

en bas ouvrez le **moniteur série**.

Remarque :

Si l'on veut compter le nombre de pressions sur le bouton, cela pose problème car la fonction loop() va trop vite, le temps de relâcher le bouton, l'incrémentation se fait plusieurs fois au lieu d'une.

Pour contourner ce problème, on utilise une variable supplémentaire qui garde en mémoire l'état précédent du bouton :

si l'on appui sur le bouton et que l'état était à 0, il passe à 1 ; s'il était à 1, il passe à 0.

5. On veut créer un vrai **interrupteur** : la led reste allumée tant que l'on n'a pas à nouveau pressé sur le bouton.

   Modifiez votre code, en tenant compte de la remarque précédente. Ajoutez une variable "etatAllumage", que vous afficherez également sur le moniteur, et testez le code.

TP 4 : Clignotement accéléré

La syntaxe d'une boucle Pour :

```
for (int i = 0; i <= 10; i = i + 1){
    //code à répéter
}
```

On **initialise** la variable i qui va servir de compteur ; on indique ensuite la **condition à tester** pour continuer ou non la répétition de la boucle ; enfin on indique l'**action sur le compteur** à chaque tour de boucle.

Dans l'exemple, i est un entier qui prend les valeurs de 0 à 10, en augmentant de 1 à chaque tour.

1. Reprenez le montage du TP 1.
2. Programmez la led pour quelle clignote de plus en plus vite puis s'éteigne pendant 2 secondes.

   Pour cela, pensez à utiliser la boucle Pour.

TP 5 : Son et lumière

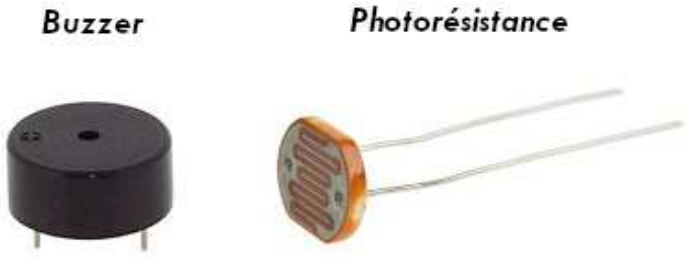
Les **capteurs analogiques** peuvent distinguer tout un panel d'états et les convertir en une valeur numérique exploitable dans nos programmes.

Par exemple, une **photorésistance** dont la résistivité varie en fonction de la quantité de lumière reçue.

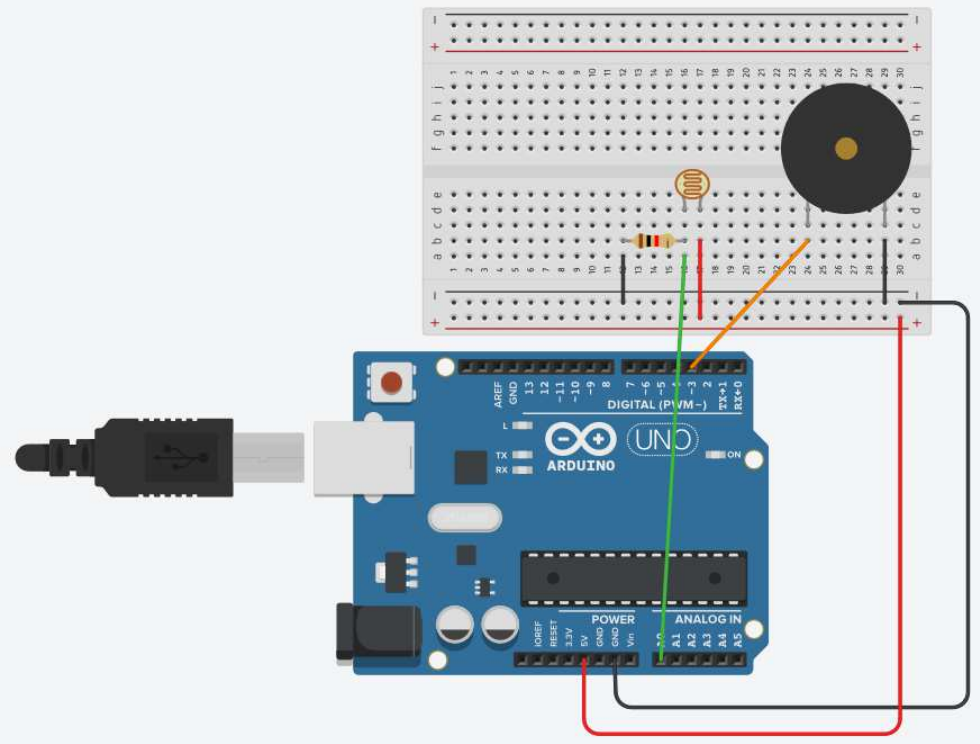
Ces capteurs sont reliés aux broches analogiques $A0$ ... $A5$.

Le **buzzer piézoélectrique** transforme l'impulsion électrique envoyée par une broche en une onde sonore de fréquence identique et audible.

Nous allons utiliser le buzzer piézoélectrique, et faire varier le son qu'il émet en fonction de l'intensité lumineuse reçue par une photorésistance.



1. Reproduisez le montage ci-dessous :



2. Copiez le code suivant :

```
const int buzzer = 3;        //numéro de la broche à laquelle est
   connectée le buzzer
const int photoResistance = A0;  //numéro de la broche à laquelle est
   connecté la photorésistance

int sensorValue = 0;            // Valeur lue sur la photorésistance
int outputValue = 0;            // Valeur envoyée au buzzer

void setup() {
  // Initialise la communication avec l'ordinateur
  Serial.begin(9600);
  // Indique que la broche 3 est une sortie :
  pinMode(buzzer, OUTPUT);
  // Indique que la broche A0 est une entrée :
  pinMode(photoResistance, INPUT);
}

void loop() {
  // lit la valeur de la photorésistance et
  // stocke le résultat dans sensorValue :
  sensorValue = analogRead(photoResistance);
  // change sensorValue vers une intervalle de 50 à 30 000 Hz
  // et stocke le résultat dans outputValue :
  outputValue = map(sensorValue, 0, 1023, 50, 30000);
  // envoie de cette nouvelle valeur sur le buzzer
  tone(buzzer, outputValue);
```

```
    // affiche les valeurs sur le moniteur
    Serial.print("photoresistance = " );
    Serial.print(sensorValue);
    Serial.println("Frequence Buzzer = ");
    Serial.println(outputValue);
}
```

Remarques :

- la photorésistance nécessite une résistance de 1 $k\Omega$.

- la fonction **analogRead()** permet de lire l'état d'une broche analogique

- le fonction **map()** permet de renvoyer une valeur numérique proportionnelle à la tension reçue.

Il faut transformer une valeur numérique entière comprise entre 0 et 1023 envoyée par la broche analogique en une intensité comprise entre 50 à 30 000 Hz lisible par le buzzer ;

- la fonction **tone()** envoie la nouvelle valeur vers le buzzer.

3. Lancez la simulation, et faite varier la lumière en cliquant sur le capteur pour faire apparaître un curseur.