# Laboratoire Arduino avancé - Capteurs

**GNG1503 – Génie de la Conception**

## Objectif

L'objectif de ce laboratoire est d'enseigner aux étudiants comment utiliser la plate-forme logiciel Arduino EDI pour écrire des commandes logiques qui répondent à une entrée d'un capteur. Ce code sera utilisé pour contrôler les entrées et les sorties d'un microcontrôleur Arduino Uno. Ce laboratoire s'appuie sur le contenu du «GNG1103-1503- Arduino lab» et suppose que les étudiants maîtrisent les bases de l'utilisation d'Arduino. Veuillez consulter «GNG1103-1503- Arduino lab» si nécessaire.

## Téléchargements

L'EDI Arduino peut être téléchargé à partir d'ici: https://www.arduino.cc/en/Main/Software

La bibliothèque pour le télémètre à ultrasons:

NewPing.h ( https://playground.arduino.cc/Code/NewPing#Download )

## Aperçu des Appareils et Équipement

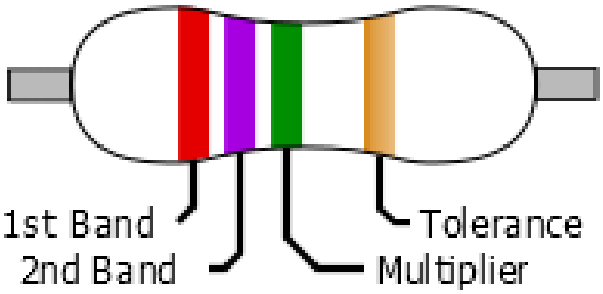
L'équipement qui sera utilisé dans ce laboratoire comprend:

- 1 × microcontrôleur Arduino Uno R3
- 1 × câble USB 2.0 type A vers USB 2.0 type B
- 1 × ordinateur de laboratoire ou personnel (sous Windows, Macintosh ou Linux)
- 1 × logiciel Arduino EDI
- 1 × photorésistance
- 1 × bouton poussoir
- 1 × 5mm DEL
- 1 × résistance de 330Ω
- 2 × résistance de 10kΩ
- 1 × télémètre à ultrasons
- 1 × capteur de mouvement infrarouge passif
- 9 × longueurs de fil
- 1 × haut-parleur piézoélectrique

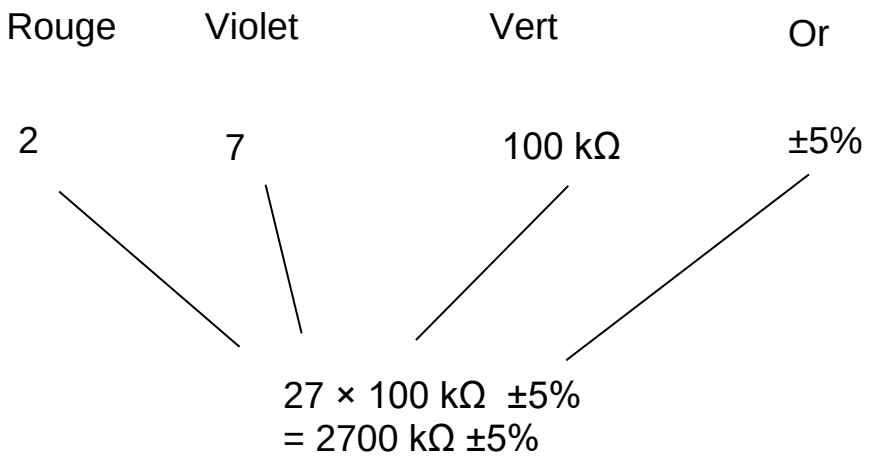
## Contexte

### Résistance

Une résistance est un composant électrique qui  s'oppose à la circulation du courant. La valeur de la résistance peut être  déterminée à partir des bandes de couleur de la résistance, qui sont lues de gauche à droite. Pour les résistances à quatre bandes, La première et deuxième bande représentent des chiffres, tandis que la troisième bande représente un multiplicateur permettant de multiplier les chiffres de première et deuxième bande. La quatrième bande est la tolérance, elle représente de combien la résistance peut différer de la valeur indiquée par les bandes. La valeur d'une résistance peut également se mesurer grâce à la fonction ohmétrique d'un multimètre. Le tableau et l'image ci-dessous montre comment lire les couleurs.




Par exemple, cette résistance a une valeur de 2700 kΩ et une tolérance de ± 5%.



| Couleur | 1er groupe | 2ème groupe | Multiplicateur | Tolérance |
|---|---|---|---|---|
| Noir | 0 | 0 | 1Ω | |
| Brun | 1 | 1 | 10Ω | ± 1% |
| Rouge | 2 | 2 | 100Ω | ± 2% |
| Orange | 3 | 3 | 1kΩ | |
| Jaune | 4 | 4 | 10kΩ | |
| Vert | 5 | 5 | 100kΩ | ± 0,5% |
| Bleu | 6 | 6 | 1 MΩ | ± 0,25% |
| Violet | 7 | 7 | 10 MΩ | ± 0,10% |
| Gris | 8 | 8 | 100 MΩ | ± 0,05% |
| Blanc | 9 | 9 | 1GΩ | |
| Or | | | 0,1Ω | ± 5% |
| Argent | | | 0.01Ω | ± 10% |

Les résistances sont souvent utilisées en série avec des composants pour réduire la quantité de courant circulant dans un circuit, souvent pour protéger les composants conçus pour des valeurs de courant inférieures.

Les résistances d’excursion haute ou basse sont utilisées pour polariser une entrée sur l'Arduino afin qu'elle soit HAUTE (HIGH) ou BASSE (LOW), respectivement. Cela doit être fait car le niveau de repos de l'entrée n'est pas nécessairement 0. Ceci est particulièrement utile lorsque vous travaillez avec des capteurs dotés d'une sortie analogique.

**Photorésistance**

Une photorésistance est une résistance dont la valeur change en fonction de la quantité de lumière à laquelle elle est exposée. Au fur et à mesure que le niveau de lumière auquel la résistance est exposée augmente, sa résistance diminue, à l'inverse, à mesure que le niveau de lumière auquel elle est exposée diminue, sa résistance augmente.

**Bouton poussoir**

Un bouton est un composant électrique qui laisse passer le courant électrique lorsque vous appuyez sur le bouton, sinon il ne laisse pas passer le courant électrique.

**Télémètre à ultrasons**

Un télémètre à ultrasons est une composante qui mesure la distance en utilisant le son. Le composant émet un son à une fréquence élevée, au-dessus de la plage de l'ouïe humaine, puis écoute l'écho de l'onde sonore. La vitesse du son dans l'air dépend des conditions atmosphériques, telles que la température, l'humidité et la pression de l'air. En conséquence, la vitesse du son varie d'un endroit à l'autre et d'un moment à un autre. Malgré la vitesse du son lors du changement d'air, le télémètre à ultrasons peut généralement déterminer les distances avec une assez grande précision. Le télémètre à ultrasons a une limite à la distance qu'il peut mesurer, de 3 cm à 400 cm environ, ainsi que la fréquence à laquelle il peut mesurer la distance, environ toutes les 29 ms.

**Capteur de mouvement infrarouge passif (MIP)**

Un capteur de mouvement infrarouge passif est un composant qui détecte les modifications de la quantité de lumière infrarouge qu'il reçoit. Lorsqu'il détecte un changement dans la quantité de lumière infrarouge qu'il reçoit, il amène la broche de sortie à l'état HAUT (HIGH), en l'absence de changement, la broche de sortie est à l'état BAS (LOW). Les MIPs sont souvent utilisés dans les systèmes de sécurité domestique.

## Préparation pré-lab

Avant d'arriver au laboratoire, les étudiants doivent consulter le manuel du laboratoire et se familiariser avec la configuration et la procédure du laboratoire. En outre, il est recommandé aux étudiants de consulter le «GNG1103-1503- Arduino lab», car ce laboratoire suppose que les étudiants comprennent parfaitement le contenu de ce laboratoire. Il est recommandé aux étudiants d'utiliser leurs propres ordinateurs pour ce laboratoire, mais des ordinateurs du laboratoire sont disponibles. Si vous utilisez un ordinateur personnel, il est de la responsabilité des étudiants de télécharger et d'installer l'EDI Arduino ainsi que toutes les bibliothèques requises, que vous pouvez trouver dans la section des téléchargements. De plus, il est conseillé

aux étudiants de réviser et de mettre en pratique la syntaxe Arduino au préalable. La référence Arduino peut être trouvée ici: https://www.arduino.cc/reference/en/

### Questions pré-lab

Que représente chaque bande, de gauche à droite sur une résistance?

_____

_____

Comment un télémètre à ultrasons détermine-t-il la distance qui le sépare d'un objet?

_____

_____

Pourquoi une résistance peut-elle être placée en série avec une DEL ou un autre composant électrique?

_____

_____

Quel est le but d'une résistance de rappel?

_____

_____

Quel est la différence principale entre les capteurs à ultrasons et MIPs?

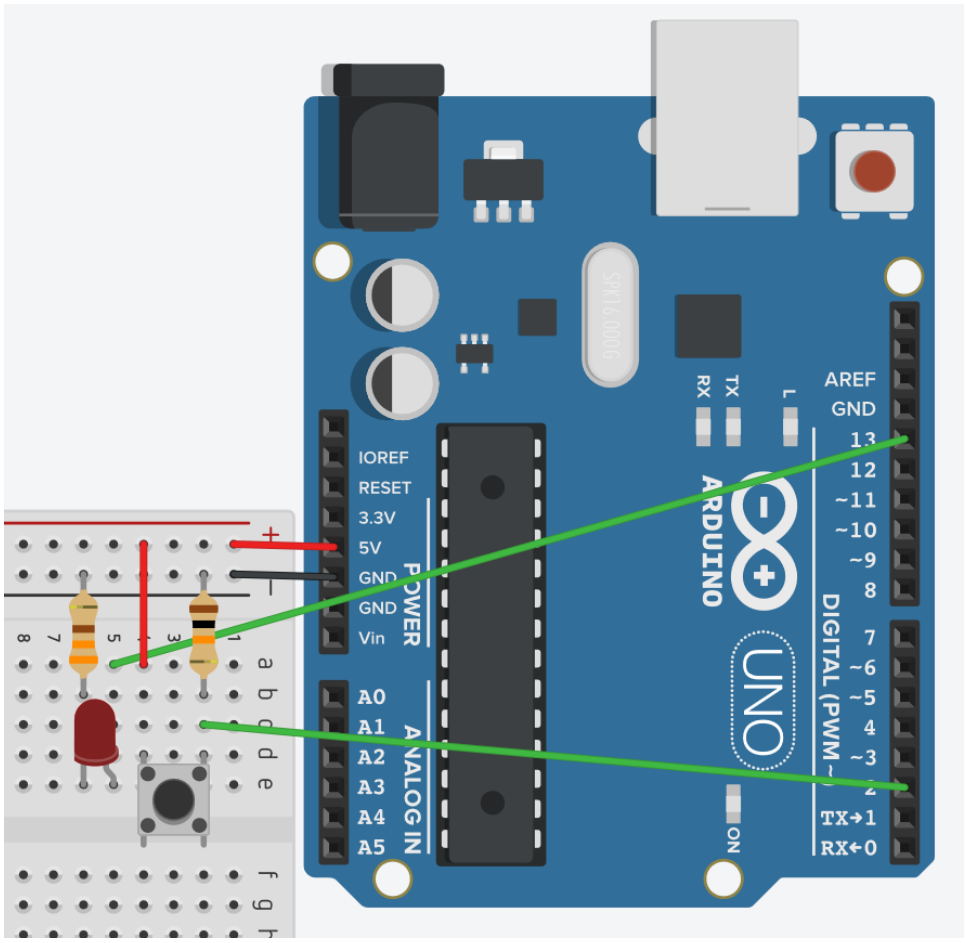
_____

_____

## Partie A - Programme de Photorésistance, Bouton et DEL

Cette partie du laboratoire traitera de la connexion d'une photorésistance, d'un bouton poussoir et d'une DEL à l'Arduino, ainsi que de la création d'un programme permettant de contrôler la DEL à l'EDI de la photorésistance et du bouton poussoir. La DEL agira comme une sortie, tandis que la photorésistance déterminera la luminosité de la DEL lorsque le bouton poussoir est enfoncé.

1. Connectez le bouton, la DEL et la photorésistance comme indiqué dans le schéma de câblage ci-dessous.

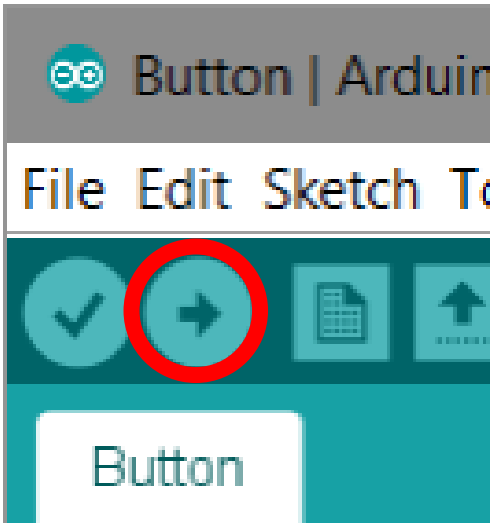



* Notez les couleurs des résistances, nous avons une résistance de 10kΩ de la sortie du bouton à la terre, il s'agit d'une résistance d'excursion basse. Il est utilisé pour polariser la broche d'entrée sur BAS (LOW).

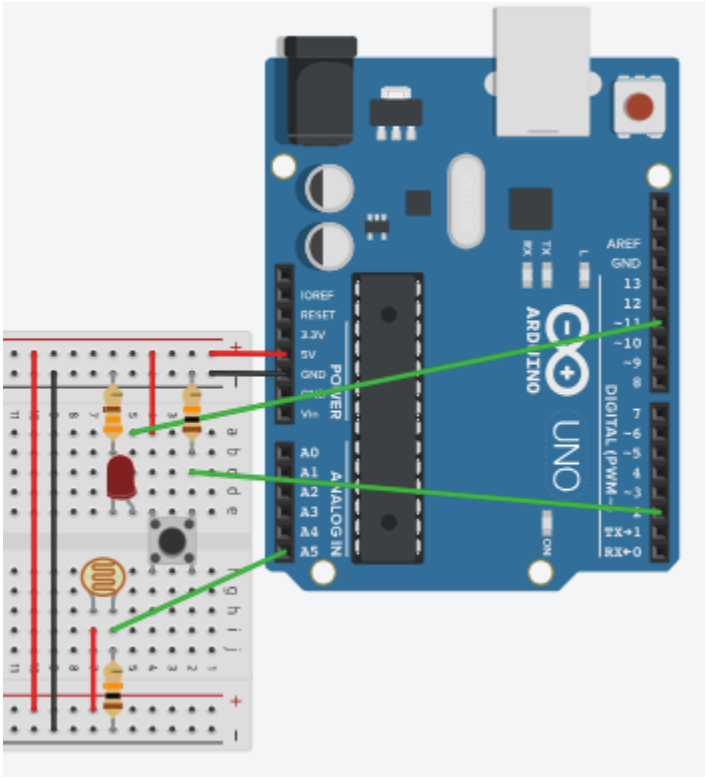
2. Lancez le logiciel Arduino EDI. Une fois démarré, sélectionnez Tools> Board> Arduino / GenuinoUno et sélectionnez le port auquel Arduino est connecté dans Tools> Port. Ouvrez File> Exemples> 02. Digital> Button. Cela ouvrira une nouvelle esquisse qui ressemble à celle ci-dessous.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

3. Téléchargez maintenant l'esquisse sur le tableau en cliquant sur le bouton en surbrillance dans l'image ci-dessous. Cela va essayer de compiler le programme et de le télécharger sur le tableau. S'il y a des erreurs dans la compilation ou le téléchargement de EDI Arduino, le message d'erreur en bas de l'EDI s'affichera dans la zone de la console.



4. Une fois le téléchargement terminé, le programme sera automatiquement exécuté sur le microcontrôleur. Dans ce cas, le voyant s'allumera chaque fois que vous appuierez sur le bouton.
5. Modifiez maintenant votre circuit pour qu'il ressemble au schéma de câblage ci-dessous.




* Notez que nous changeons la broche de sortie de 13 à 11, car nous avons besoin d'une sortie analogique. Une résistance d'excursion basse a été placée à la sortie de la photorésistance.

6. Ajoutez ou modifiez maintenant les lignes de code suivantes dans le programme.
   - Modifiez «const int ledPin = 13» sur la nouvelle broche de la DEL, broche 11.
   - Sur la ligne suivante, ajoutez «const int sensorPin = A5;»
   - Sur la ligne après «buttonState = 0;», ajoutez «int sensorValue;».
   - Dans la configuration, ajoutez «Serial.begin (9600); »
   - Au début de la boucle, ajoutez «sensorValue = analogRead (sensorPin);» et immédiatement après, ajoutez «Serial.println (sensorValue)»;
   - Dans l'instruction if, lorsque le paramètre buttonState a la valeur High, définissez digitalWrite sur «analogWrite (ledPin, sensorValue)»;

Le programme devrait maintenant ressembler à l'image ci-dessous.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  11;      // the number of the LED pin
const int sensorPin = A5;

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int sensorValue;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    analogWrite(ledPin, sensorValue);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

7. Téléchargez et testez votre nouveau programme. Que se passe-t-il quand vous appuyez sur le bouton maintenant? Essayez de changer la quantité de lumière à laquelle la photorésistance est exposée, que se passe-t-il?

---

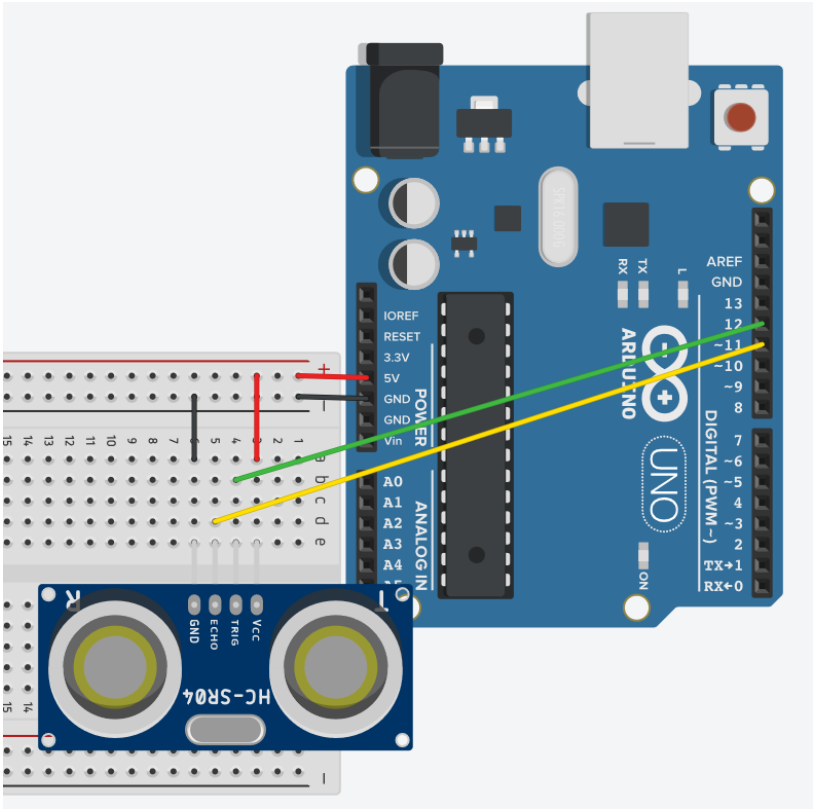
---

8. Pensez à un moyen de changer le comportement de la DEL en réponse à l'entrée de la photorésistance et mettez-le en œuvre!

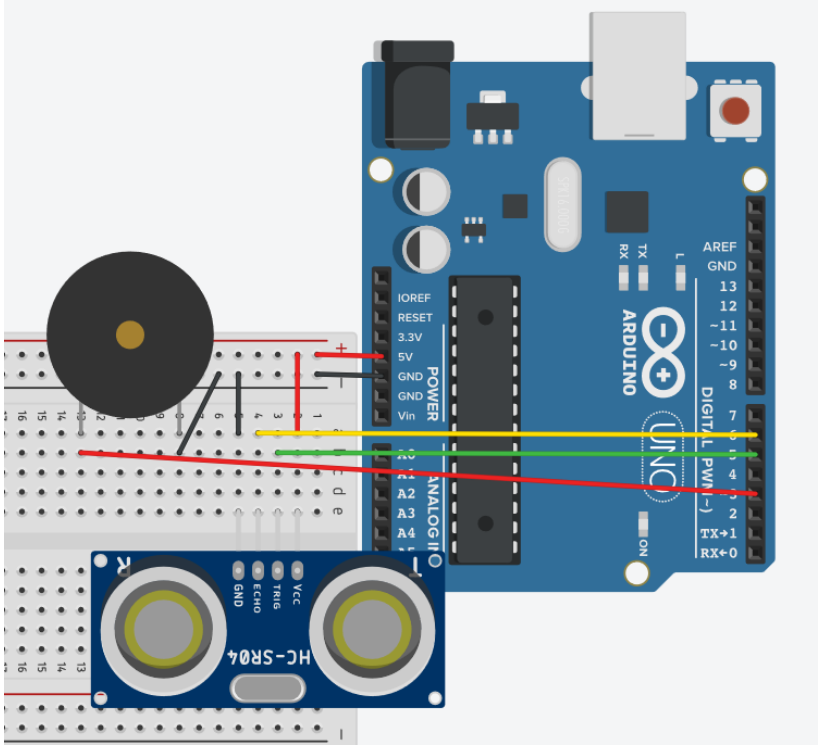
## Partie B - Télémètre à Ultrasons

Dans cette partie du laboratoire, vous créerez un programme qui indiquera la distance détectée par le télémètre à ultrasons à la console Arduino EDI. Pour tester le programme et le circuit, placez votre main ou un autre objet solide devant le télémètre à ultrasons et déplacez-le en avant et en arrière. La sortie de la console doit indiquer la distance par rapport à l'objet de manière assez précise. Nous utiliserons la sortie du télémètre à ultrasons pour contrôler le volume d'une marque de haut-parleur piézoélectrique.

1. Créez le circuit dans le schéma de câblage ci-dessous.




2. Dans le Arduino EDI trouver File> Examples> NewPing> NewPingExample, ceci devrait ouvrir une esquisse dans une nouvelle fenêtre. Recherchez Serial.begin() et remplacez le débit en bauds par 9600. Téléchargez le programme et testez-le.
3. Rebranchez les broches “TRIG” et “ECHO” aux broches 5 et 6 respectivement. Modifiez le code pour refléter les modifications apportées au câblage, puis téléchargez et testez.
4. Nous allons maintenant associer une sonnerie à l’Arduino, comme indiqué dans l’image ci-dessous.

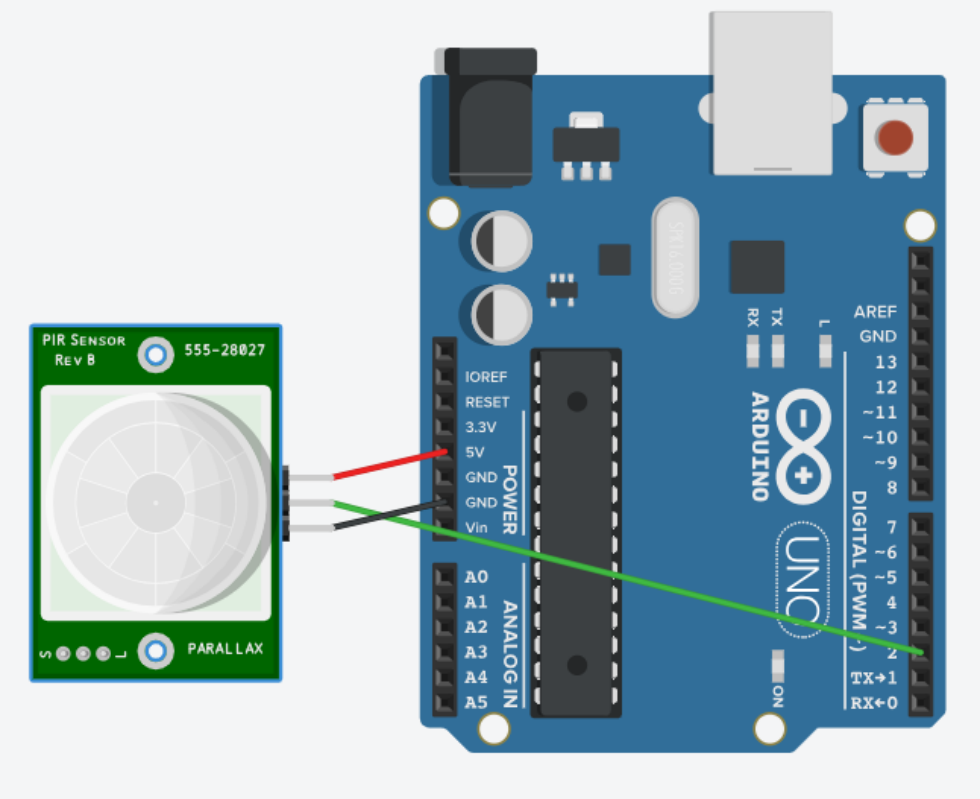



5. Nous devons maintenant modifier le code pour utiliser le haut-parleur piézoélectrique. Normalement, lorsque vous utilisez un haut-parleur piézo-électrique, la fonction tone() est utilisée, mais cette fonction est incompatible avec la bibliothèque NewPing. En conséquence, nous devrons mettre en œuvre nous-mêmes la génération de sons. Nous avons besoin que le haut-parleur piézoélectrique oscille de HAUT (HIGH) à BAS (LOW) à la fréquence souhaitée. Pour ce faire, allez chercher l'exemple sur le campus virtuel appelé «URFPiezo2_Fr.ino».

## Partie C – Capteur Mouvement Infrarouge Passif

Dans cette section du laboratoire, vous apprendrez à utiliser un capteur infrarouge passif (MIP). Il est important de noter que le capteur a un large champ de vision lors de son test. Cette partie du laboratoire couvrira le processus de réflexion derrière l'écriture du code pour le MIP. Le programme qui sera créé utilisera le MIP pour détecter les mouvements. Dès qu'un mouvement est détecté, le voyant attaché à la broche 13 d'Arduino s'allume. De plus, notre programme écrit sur la console chaque fois que le mouvement commence et s'arrête. Essayez de compléter chaque étape sans consulter le code exemple.

1. Pour commencer, branchez le circuit comme indiqué sur l'image ci-dessous.




2. Pour commencer à écrire le programme, commencez une nouvelle esquisse, allez à File> New.
3. Nous allons commencer le programme en déclarant les variables qui seront utilisées. Dans ce cas, nous avons besoin d'une variable pour représenter la broche DEL, la broche d'entrée PIR, si quelque chose a commencé à bouger ou continue de bouger. Pour cela, avant le setup(), écrivez ce qui suit

```
1 int ledPin = 13;
2 int pirInput = 2;
3 bool wasMoving = false;
```

4. Lorsque Arduino démarre et exécute le programme pour la première fois, il doit configurer les modes des différentes broches utilisées par le programme. Dans ce cas, la broche DEL doit être une sortie et la broche d'entrée PIR doit être une entrée. Dans le bloc setup(), ajoutez ce qui suit.

```
5 void setup() {
6   pinMode(ledPin, OUTPUT);
7   pinMode(pirInput, INPUT);
```

5. L'Arduino doit également générer du texte sur la console Arduino EDI. Il est donc nécessaire d'initier la connexion série. Ajoutez ce qui suit au bloc setup().

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

6. Maintenant que l'Arduino a configuré tout ce que le programme utilisera tout au long de la boucle principale, nous commençons à travailler sur le corps principal du programme, la boucle. Chaque fois que la boucle est exécutée, nous voulons savoir si le capteur MIP a détecté un mouvement. S'il y a du mouvement, le programme doit allumer la DEL, sinon le programme devrait l'éteindre. Ajoutez ce qui suit.

```
10
11 void loop() {
12   if (digitalRead(pirInput) == HIGH) {
13     digitalWrite(ledPin, HIGH);
14   } else {
15     digitalWrite(ledPin, LOW);
16   }
17 }
18
```

7. Enfin, nous voulons que l'Arduino écrive sur la console si la sortie du MIP change, c'est-à-dire que si le MIP vient de démarrer ou d'arrêter de détecter un mouvement. En fin de compte, le code devrait ressembler à ce qui suit.

```
1  int ledPin = 13;
2  int pirInput = 2;
3  bool wasMoving = false;
4
5  void setup() {
6    pinMode(ledPin, OUTPUT);
7    pinMode(pirInput, INPUT);
8    Serial.begin(9600);
9  }
10
11 void loop() {
12   if (digitalRead(pirInput) == HIGH) {
13     digitalWrite(ledPin, HIGH);
14     if(wasMoving == false){
15       wasMoving = true;
16       Serial.println("Movement started");
17     }
18   } else {
19     digitalWrite(ledPin, LOW);
20     if(wasMoving == true){
21       wasMoving = false;
22       Serial.println("Movement stopped");
23     }
24   }
25 }
```

Cette partie du laboratoire a été adaptée du didacticiel sur les capteurs Adafruit PIR.

8. Maintenant, testez le MIP, quelle est sa sensibilité? À quelle distance ou à quelle distance peut-il détecter un mouvement? Quelle est la taille de son champ de vision?