EQUIPE :	Date :
PROJET 2 : THEREMINE LUMINEUX	

IL EST TEMPS DE FAIRE DU BRUIT ! EN UTILISANT UNE PHOTORÉSISTANCE ET UN ÉLÉMENT PIÉZOÉLECTRIQUE, VOUS ALLEZ CONSTRUIRE UN THÉRÉMINE LUMINEUX

Objectif : faire du son avec la fonction tone(), le calibrage des capteurs analogiques.

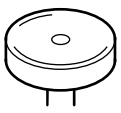
Durée : 1 heure

Difficulté : ■■■■□

				Score
++	++	++	++	
+	+	+	+	
-	-	-	-	
- -	- -	- -	- -	

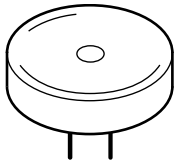
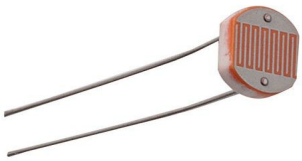
Un 'Thérémine' est un instrument qui produit des sons en fonction des mouvements des mains du musicien autour de lui. Vous en avez déjà probablement entendu dans des films d'horreur.

Vous utiliserez une photorésistance pour détecter la quantité de lumière. En bougeant vos mains autour du capteur, vous changerez la quantité de lumière atteignant le capteur. La variation de tension présente sur la broche analogique déterminera la fréquence de la note à jouer.

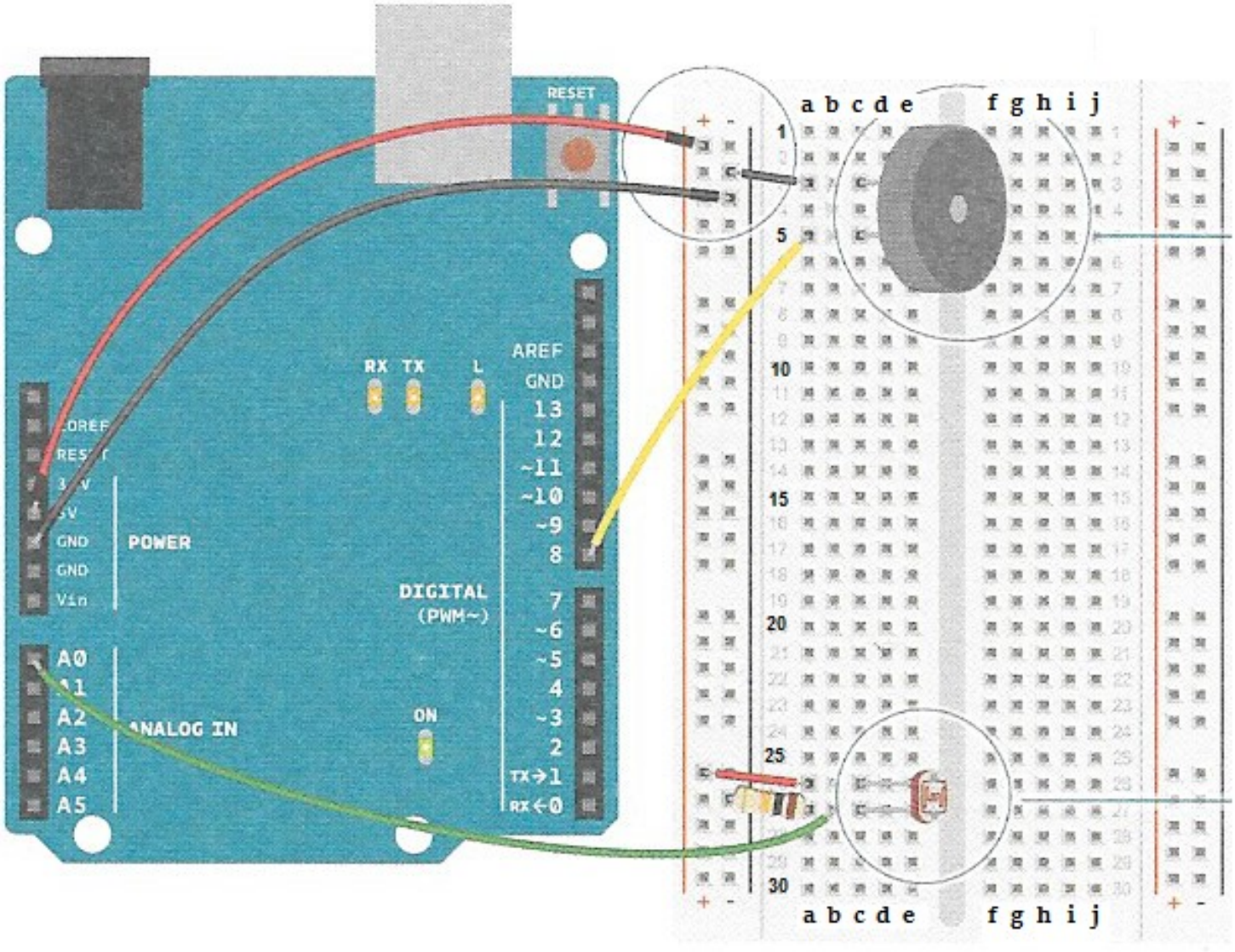


Un élément piézoélectrique (piézo) est un élément qui vibre lorsqu'il reçoit de l'électricité. Lorsqu'il bouge, il déplace de l'air autour de lui, créant des ondes sonores.

LES COMPOSANTS

		
RESISTANCE 10 kΩ	PIEZO	PHOTO RESISTANCE
x1	x1	x1

LE MONTAGE



LE CODE

Créer des variables pour calibrer le capteur

Créez une variable pour stocker les valeurs lues par `analogRead()` sur la photorésistance. Ensuite, créez des variables pour les valeurs haute et basse. Vous initialisez la variable `sensorLow` avec la valeur 1023, et `sensorHigh` à 0. Lorsque vous lancerez le programme pour la première fois, vous comparerez ces valeurs aux lectures faites sur le capteur pour trouver les réels maximum et minimum.

```
1 int sensorValue;
2 int sensorLow = 1023;
3 int sensorHigh =;
```

Nommez une constante pour l'indicateur de fin de calibrage

Créez une constante appelée `ledPin`. Vous l'utiliserez pour indiquer que votre capteur a fini de se calibrer. Pour ce projet, utilisez la LED reliée à la broche 13 déjà présente sur la carte.

```
4 const int ledPin = ;
```

Configurez la broche numérique et mettez-la à l'état haut

Dans le `setup()`, configurez le `pinMode()` de ledPin en OUTPUT, et allumez la lumière.

```
5 void setup() {

6   pinMode(ledPin, OUTPUT);
7   digitalWrite(ledPin, HIGH);
```

Utilisez une boucle while pour le calibrage

Passons au calibrage du capteur. Vous utiliserez une boucle `while()` pendant 5 secondes : celle-ci tourne en boucle jusqu'à ce que sa condition ne soit plus remplie.
`millis()` est une fonction renvoyant les millisecondes écoulées depuis le dernier reset de l'Arduino, on l'utilisera pour garder trace du temps qui passe.

```
8   while (millis < 5000) {
```

Comparez les valeurs du capteur pour le calibrage

Dans la boucle, vous lirez les valeurs en provenance du capteur : si la valeur vaut moins que `sensorLow` (initialement 1023), vous mettrez à jour cette variable. Si elle est plus haute que `sensorHigh` (initialement 0), vous mettrez à jour cette dernière.

```
9     sensorValue = analogRead(A0);
10    if (sensorValue > sensorHigh) {
11      sensorHigh = sensorValue;
12    }
13    if (sensorValue sensorLow) {
14      sensorLow = sensorValue;
15    }
16  }
```

Indiquez que le calibrage est fini

Lorsque 5 secondes se sont écoulées, la boucle while() s'arrêtera. Éteignez la LED branchée à la broche 13. Vous utiliserez les valeurs haute et basse que vous venez de déterminer pour les faire correspondre à une fréquence dans la partie principale de votre programme.

```
17  digitalWrite(ledPin, L);
18 }
```

Lisez et stockez la valeur du capteur

Dans la `loop()`, lisez la valeur sur A0 et stockez-la dans `sensorValue`.

```
19 void l() {
20   sensorValue = analogRead(A0);
```

Faites correspondre la valeur du capteur à une fréquence

Créez une variable nommée `pitch`. La valeur de `pitch` sera déterminée à partir de `sensorValue`. Utilisez `sensorLow` et `sensorHigh` comme valeurs limites pour les valeurs d'entrée. Pour la sortie, essayez d'abord 50 et 4000. Ces nombres définissent l'échelle de fréquences que l'Arduino va générer.

```
21  int pitch =
      map(sensorValue,sensorLow,sensorHigh, , 4000);
```

Jouez la fréquence

Ensuite, appelez la fonction `tone()` pour jouer un son. Elle prend trois arguments : sur quelle broche jouer le son (ici la broche 8), quelle fréquence à jouer (déterminée par la variable pitch), et pendant combien de temps jouer la note (essayez 20 millisecondes pour commencer).
Ensuite, appelez un `delay()` pour 10 millisecondes pour donner au son le temps de se moduler.

```
22   tone(8,pitch,20);


23   delay( );
24 }
```