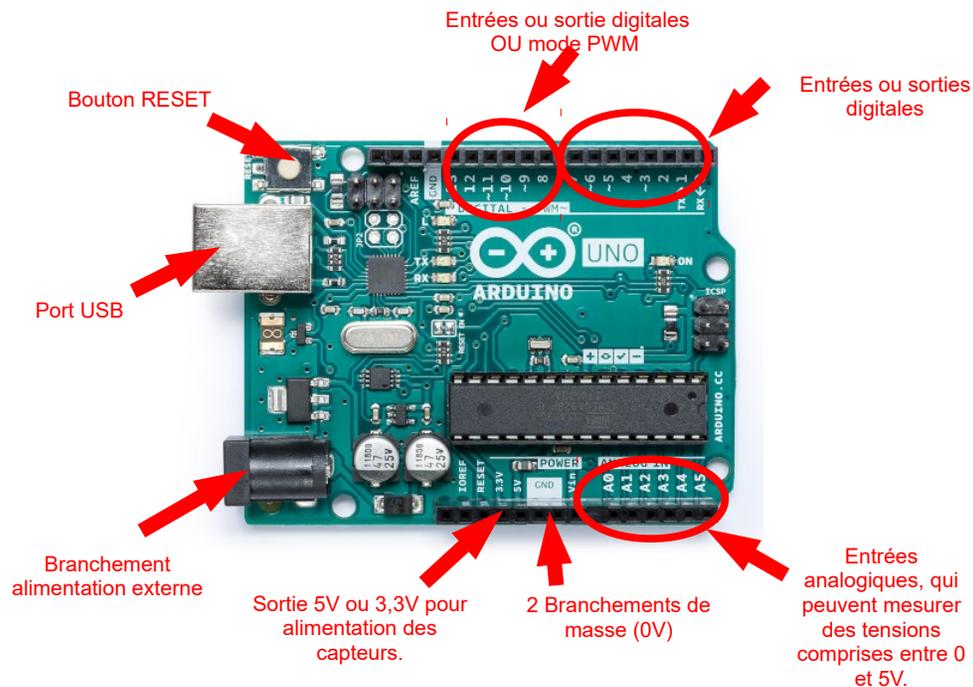


Programmer en langage Arduino



La carte Arduino est un microcontrôleur, c'est à dire une sorte de mini ordinateur qui sert d'interface entre l'environnement (actions, mesures de grandeurs...) et un utilisateur. Elle se programme nativement dans un langage dérivé du C : le langage « Arduino »

Présentation de la carte :



Les bornes 3,5,6,9,10 et 11 (avec le symbole ~) peuvent servir de sorties analogiques (PWM)

Le logiciel utilisé :

Arduino est le Logiciel permettant d'installer les drivers des cartes Arduino, de la programmer **en langage C**. Il est téléchargeable librement sur : <https://www.arduino.cc/>

```
AnalogReadSerial | Arduino 1.8.8
Fichier Édition Croquis Outils Aide

AnalogReadSerial
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
}
```

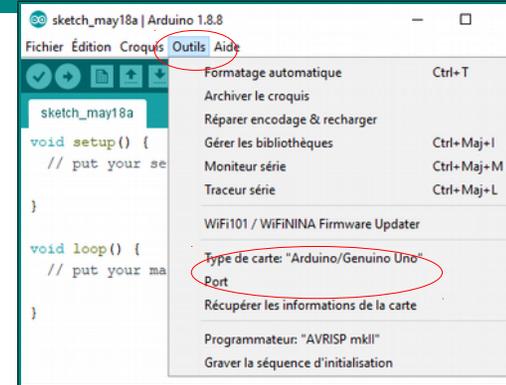
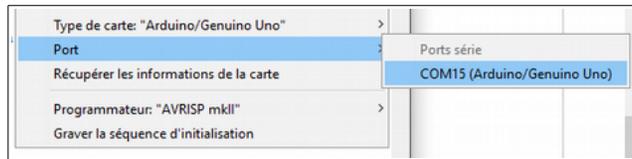
Fiche 1 : Prise en main du logiciel :

Ouvrir le logiciel Arduino :

Choisir le type de carte :

faire **Outils** > **Type de carte** > **Arduino/Genuino Uno**

Puis connecter la carte : faire **Outils** > **Port** > **COM...**



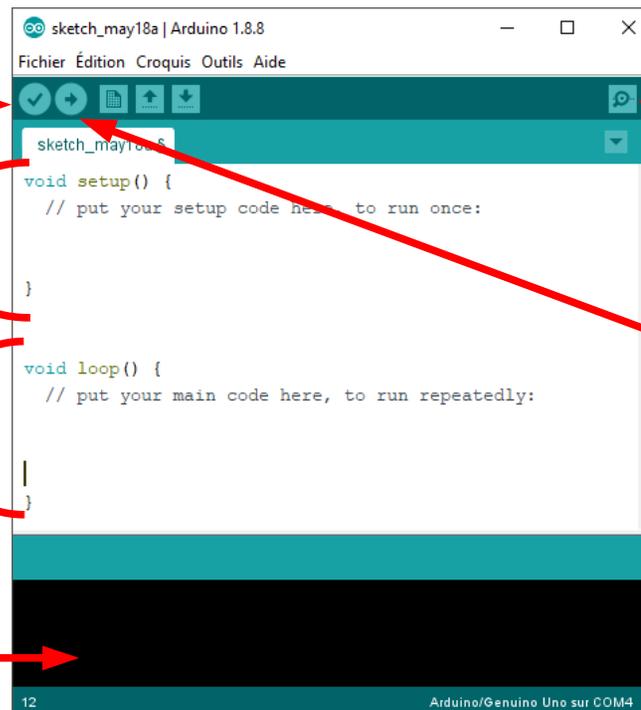
L'interface du logiciel :

Compile le programme : c'est à dire vérifie s'il n'y a pas d'erreur. Peut être utilisé, même si aucune carte n'est connectée, pour corriger la syntaxe d'un programme.

Cette boucle est exécutée **une fois** à l'initialisation du programme et quand la bouton **reset** de la carte est pressé.

La partie de programme se trouvant se répètent **indéfiniment** tant que la carte est alimentée.

Les messages d'erreur s'afficheront ici.



Lance le moniteur série qui affiche les valeurs et mesures renvoyée par la carte par la fonction :
Serial.print()
Serial.println() (pour sauter des lignes)

Téléverse le programme sur la carte Arduino, pour qu'il puisse tourner. Dès qu'une modification est faite dans le programme, il faut téléverser le programme de nouveau pour qu'elle soit prise en compte.

Fiche 2 : Prise en main du langage Arduino

De nombreux exemples pré-enregistrés peuvent être chargés et testés : ils permettent de se familiariser avec le langage plus rapidement. Nous allons tester le plus simple. Faire **Fichier** > **Exemples** > **01.Basics** > **Blink**.

- Le logiciel demande la sauvegarde le programme dans un dossier qui porte le même nom à la première compilation.

- Les majuscules et minuscules sont prises en compte dans les fonctions et variables.

- En langage Arduino, la plupart des lignes se termine par un point virgule ;

- Les blocs, boucles, fonctions sont encadrées par des accolades { }, apprendre à bien les repérer !

- Toutes les variables doivent être déclarées (nom et type).

- L'indentation (décalage) n'est pas obligatoire mais conseillée pour bien se repérer dans le programme.

Les commentaires sont définis par // en début de ligne. Les blocs de commentaires par /* au début et */ à la fin.

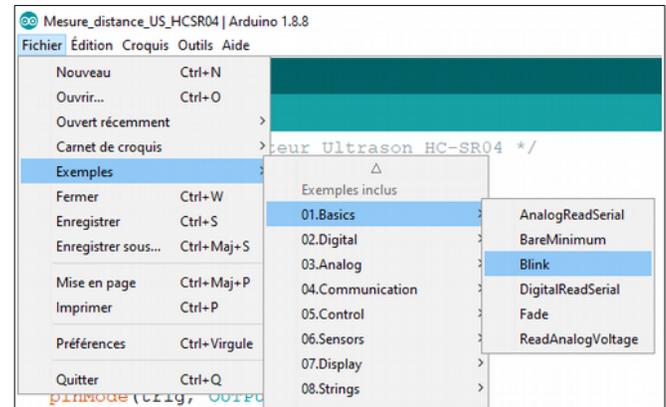
Les principales fonctions seront vues à travers quelques exemples de programmes.

Ici LED_BUILTIN correspond à la led qui se trouve sur la carte (emplacement PIN 13).

Envoie 5V sur la LED de la carte

Attendre 1000 ms

Remet la tension de la led à 0V



```
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(1000);  
}
```

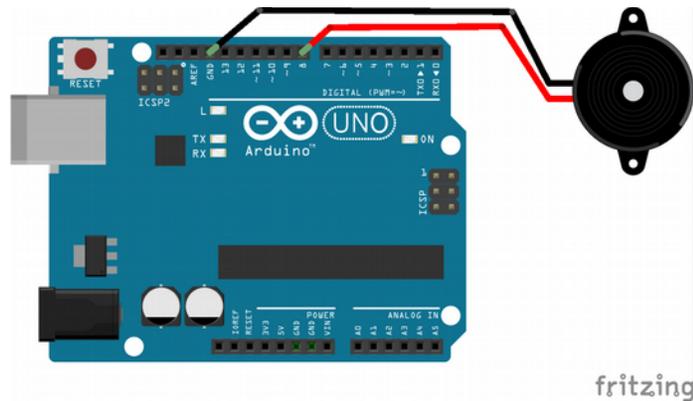
Fiche 3 : générer un son avec la carte Arduino :

La fonction **tone()** permet de générer un son de fréquence et durée choisie sur une sortie digitale de l'Arduino.

Tone(8,440,1000) : signifie jouer un son de fréquence **440Hz** pendant **1000 ms** sur la borne (pin) **8**.

Le code est mis dans la boucle « **void setup** » pour que le son ne soit joué qu'une fois (sinon ça ne s'arrête que si la carte est débranchée!!).

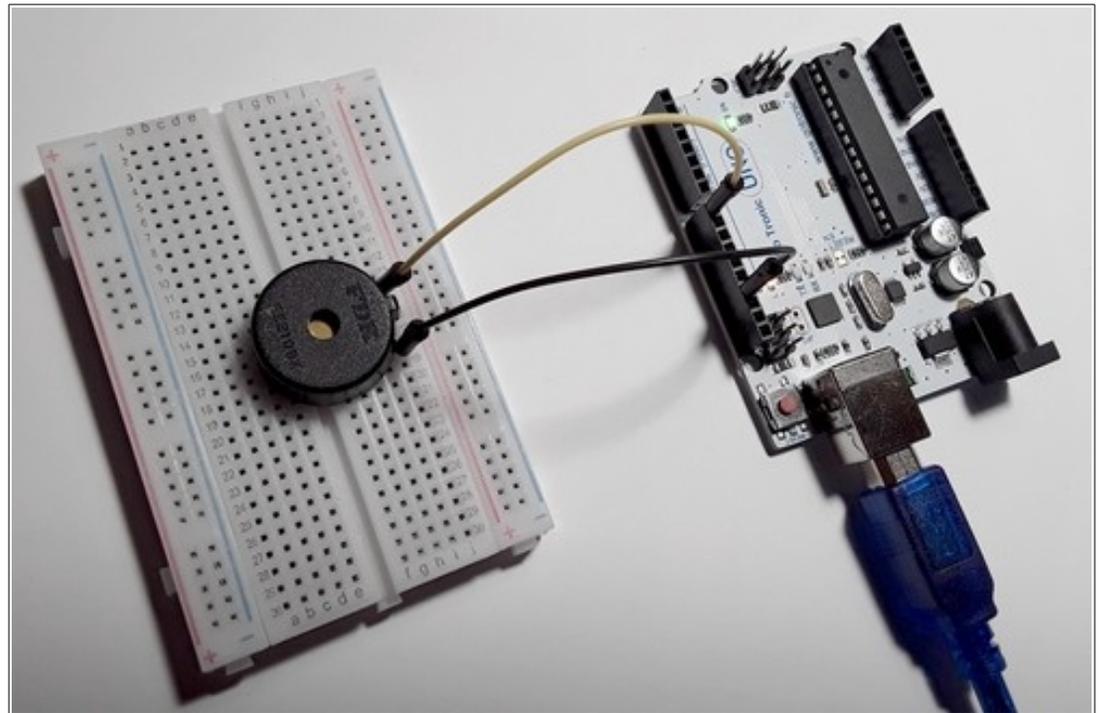
Pour relancer la mélodie il suffit d'appuyer sur le bouton **reset**.



```
produire_un_son$  
void setup() {  
  tone(8,440,1000);  
}
```

Remarque :

La borne 8 est conçue pour que l'on puisse brancher un haut parleur d'impédance 8Ω , produisant un son beaucoup plus joli que le simple buzzer.



Proposition d'activité : générer un son de fréquence donnée et jouer les octaves suivantes

Pour jouer plusieurs sons dont les fréquences sont liées à la suite, il est possible d'introduire la boucle for :

for (int i =1 ; i<5 ; i++) : jouer la boucle pour i allant de 1 à 4 en augmentant i d'une unité à chaque passage.

Comment changer la fréquence de la note de départ ?

Comment modifier le code pour jouer 5 sons en tout ?

Comment changer la durée des sons ? La pause entre les notes ?

```
int freq0 = 440;
int borne_buzzer = 8;
int duree = 1000;
void setup() {

for (int i=1;i<5;i++){
    tone(borne_buzzer,freq0*i,duree);
    delay(1000);
}

void loop() {
}
```

Un prolongement possible :

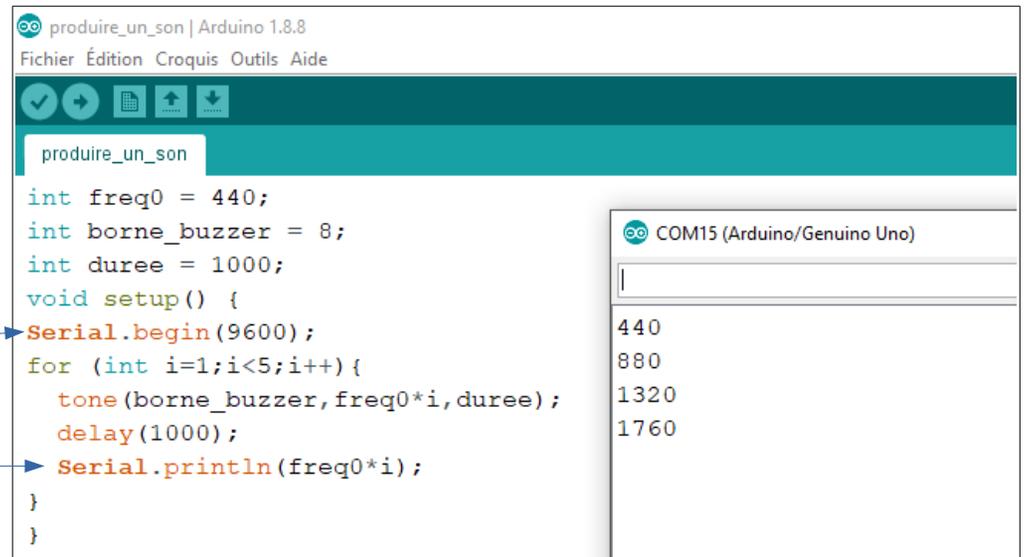
Il est possible de faire afficher la valeur de la fréquence de chaque son par la carte dans le moniteur série.

Modifier le code comme ci-contre, puis téléverser.
Le moniteur série est la console qui permet de recevoir les informations produites par la carte. Pour l'afficher il faut cliquer sur l'icône 

Les valeurs s'affichent dans la fenêtre qui apparaît.

Ouvre la communication en 9600 bps entre la carte et l'ordi.

Affiche les valeurs en sautant des lignes.



```
produire_un_son | Arduino 1.8.8
Fichier Édition Croquis Outils Aide

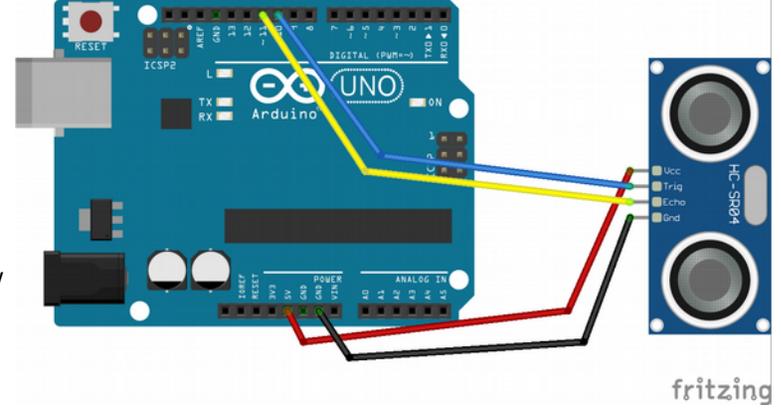
produire_un_son
int freq0 = 440;
int borne_buzzer = 8;
int duree = 1000;
void setup() {
Serial.begin(9600);
for (int i=1;i<5;i++){
    tone(borne_buzzer,freq0*i,duree);
    delay(1000);
Serial.println(freq0*i);
}
}

COM15 (Arduino/Genuino Uno)
440
880
1320
1760
```

Fiche 4 : Mesure de la vitesse du son :



Les capteurs ultrasonores type **HC-SR04** permettent de générer une impulsion ultrasonore puis de mesurer le temps que met le son à revenir d'un obstacle. Il est donc possible d'utiliser ce capteur pour mesurer la vitesse du son ou des distances.



Pour que le capteur fonctionne, il faut lui envoyer en impulsion de 5V pendant 10ms sur la borne **TRIG**.

Une fois qu'il a reçu cette information, il envoie une impulsion ultrasonore et mesure le temps que met cette impulsion à revenir.

Ce résultat est retourné par la borne **ECHO** en micro-secondes.

Voici le début du code :

Définit le numéro de la borne trig (pin)
Définit le numéro de la borne echo

Les variables qui vont nous servir pour
la mesure et les calculs.

Définit le pin « trig » comme une sortie digitale.

Définit le pin « echo » comme une entrée digitale.

Envoie une impulsion de 5V (HIGH)
pendant 10 μ s
la fonction pulseIn() renvoie la durée de l'écho
en micro-seconde (aller et retour).

Remarque :

le **nom** et le **type** de chaque variable doivent être définis au départ.

```
int trig = 10;
int echo = 11;
float duree_echo;
float distance_cm;
float vitesse_son;

void setup()
{
  pinMode(trig, OUTPUT);
  digitalWrite(trig, LOW);
  pinMode(echo, INPUT);
  Serial.begin(9600);

  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  duree_echo = pulseIn(echo, HIGH);
}
```

Proposition d'activité : Mesure de la vitesse du son dans l'air

- Placer le capteur face à un obstacle.
 - Mettre une règle à côté pour pouvoir mesurer la distance entre le capteur HCSR04 et l'obstacle (voir photo).
 - Taper le code complet ci-contre. Remplacer les **???** par la distance mesurée entre l'obstacle et le capteur.
 - Trouver la formule ******* permettant de calculer la vitesse du son en **m/s**.
 - Téléverser, puis afficher le moniteur série.
- La valeur affichée est-elle cohérente ?

Pour augmenter la précision, il est possible de faire une autre mesure :

- Déplacer le capteur, puis changer la distance dans le programme. Ne pas oublier de téléverser de nouveau.

Prolongement possible :

Une fois la valeur de la vitesse du son trouvée, il est possible de faire un nouveau programme pour transformer la carte Arduino en télémètre !

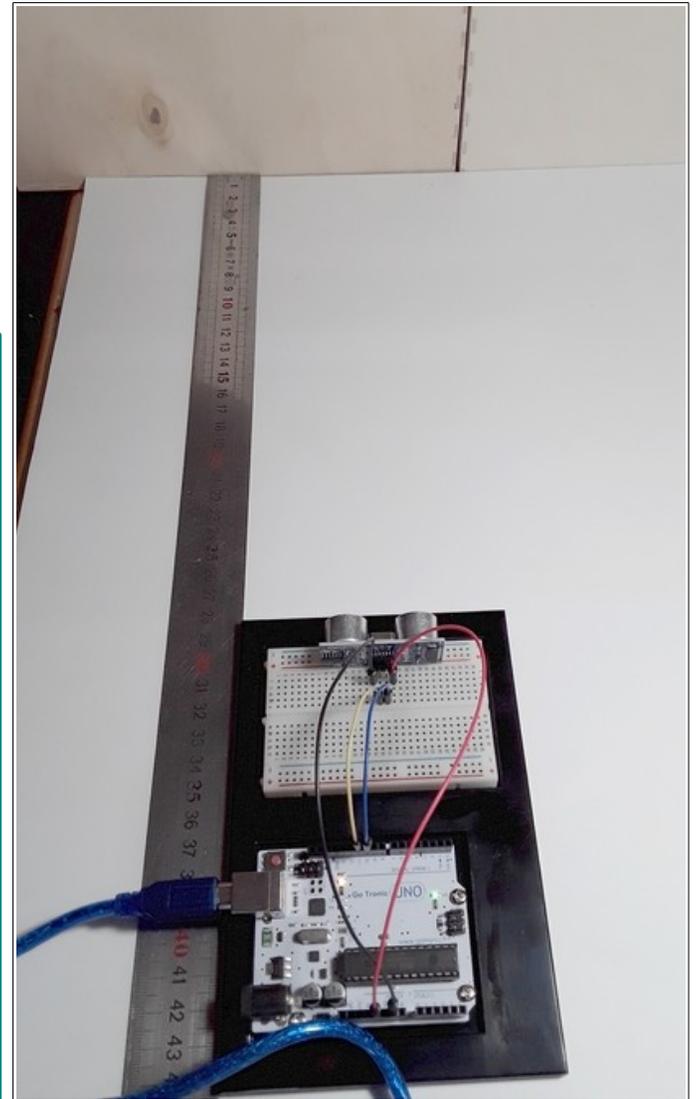
```
// le code complet
int trig = 10;
int echo = 11;
float duree_echo;
float distance_cm = ???;
float vitesse_son;

void setup()
{
  pinMode(trig, OUTPUT);
  digitalWrite(trig, LOW);
  pinMode(echo, INPUT);
  Serial.begin(9600);

  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  duree_echo = pulseIn(echo, HIGH);
  vitesse_son = *** ;
  Serial.println(vitesse_son);
  delay(500);
}

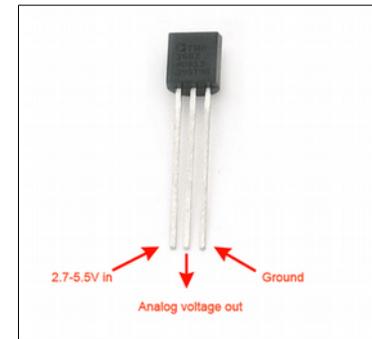
void loop(){
}
```



Fiche 5 : Mesure de température avec un capteur TMP36

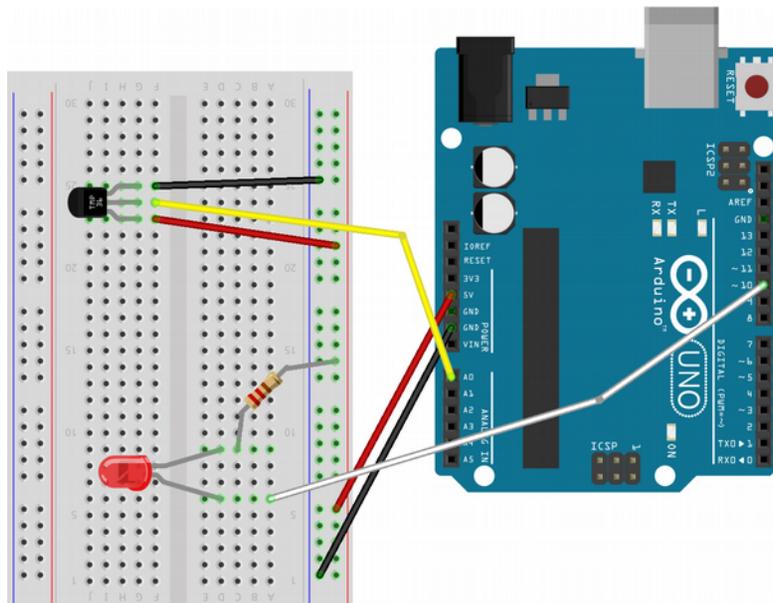
La carte Arduino peut mesurer des tensions sur les entrée analogiques A0...A5.
La fonction **analogRead**(pin) renvoie une valeur analogique lue comprise entre 0 et 1023, ce qui correspond à une tension comprise entre 0V et 5V.
Nous allons donc pouvoir mesurer la tension aux bornes d'un capteur.

Par exemple dans l'activité suivante, le capteur utilisé est un **TMP 36**, une fois correctement alimenté (voir ci contre), il sort une tension analogique proportionnelle à la température



Proposition d'activité : mesure d'une température et déclenchement d'une alarme :

Afin d'assurer une meilleure surveillance des cas d'hypothermie à l'hôpital, chaque patient d'un CHU s'est vu remettre un bracelet contenant un capteur de température, une DEL et un microcontrôleur.
Votre mission consiste à réaliser le montage permettant l'allumage de la DEL en cas d'hypothermie sévère (moins de 28°C). Dans le programme suivant, agir sur la valeur de Umes dans la boucle **if** :



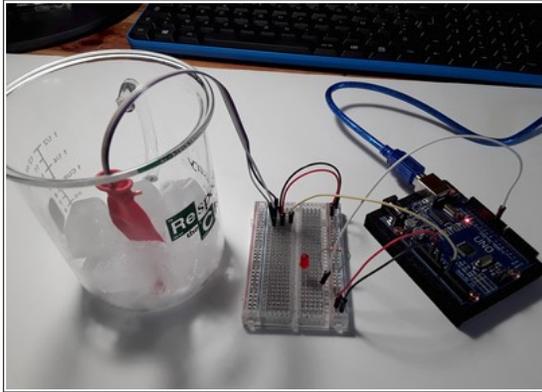
fritzing

```
void setup() {  
  Serial.begin(9600);  
  pinMode(10, OUTPUT);  
}  
  
void loop() {  
  int Umes = analogRead(A0);  
  Serial.println(Umes);  
  if (Umes < 150) {  
    digitalWrite(10, HIGH);  
  }  
  else {  
    digitalWrite(10, LOW);  
    delay(500);  
  }  
}
```

Quelques astuces

Astuce 1 :

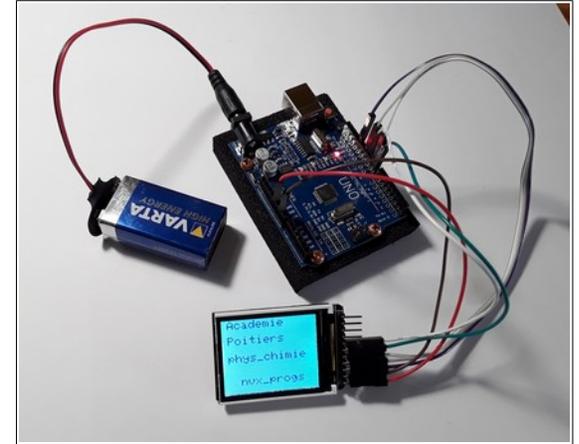
Pour pouvoir étalonner un capteur qui n'est pas étanche, il est possible de le mettre dans un ballon de baudruche.



Astuce 2 :

Une fois le microprogramme déposé sur la carte **Arduino**, celui peut continuer de tourner de manière autonome, sans PC, à condition d'alimenter la carte.

Applications :
mesures nomades sur le terrain, station météo...



Astuce 3 :

Ne jamais faire débiter à la carte une intensité totale dépassant 200mA et 40 mA par sortie : cela signifie qu'il faut toujours placer une résistance d'au moins **200 Ω** (avec une marge) entre une sortie digitale et la masse GND.

Quelques liens :

Le site officiel **Arduino** : <https://www.arduino.cc/>

La physique avec Arduino : <https://opentp.fr/card/>

La physique autrement :

http://hebergement.u-psud.fr/supraconductivite/projet/enseigner_la_physique_avec_arduino/

Fritzing, le logiciel libre et gratuit pour schématiser les montages : <http://fritzing.org/home/>

Astuce 4 :

Lors de l'achat d'un capteur ou d'un actionneur, ne pas oublier de télécharger la fiche technique du fabricant : elle contient souvent les caractéristiques techniques, les branchements et des exemples de codes permettant d'utiliser très rapidement le composant !

Guide de mise en marche du capteur à effet Hall

Matériel nécessaire :

- une carte [Arduino Uno](#)
- un [câble de connexion USB](#)
- un [capteur à effet Hall 57022](#)

Présentation du module :

Module à effet Hall utilisant une tension analogique en fonction du champ magnétique se situant à proximité (un aimant par exemple).

Alimentation : 5 Vdc
Connecteurs : 3 broches (Vcc, GND et Signal)
Dimensions : 30 x 23 mm

Connexion du capteur :

Branchez le capteur aux broches de la carte Uno comme représenté ci-dessous :

Table de correspondance :

Carte Uno	Capteur à effet Hall
GND	GND
5Vdc	+ (broche du module)
DIGITAL	-