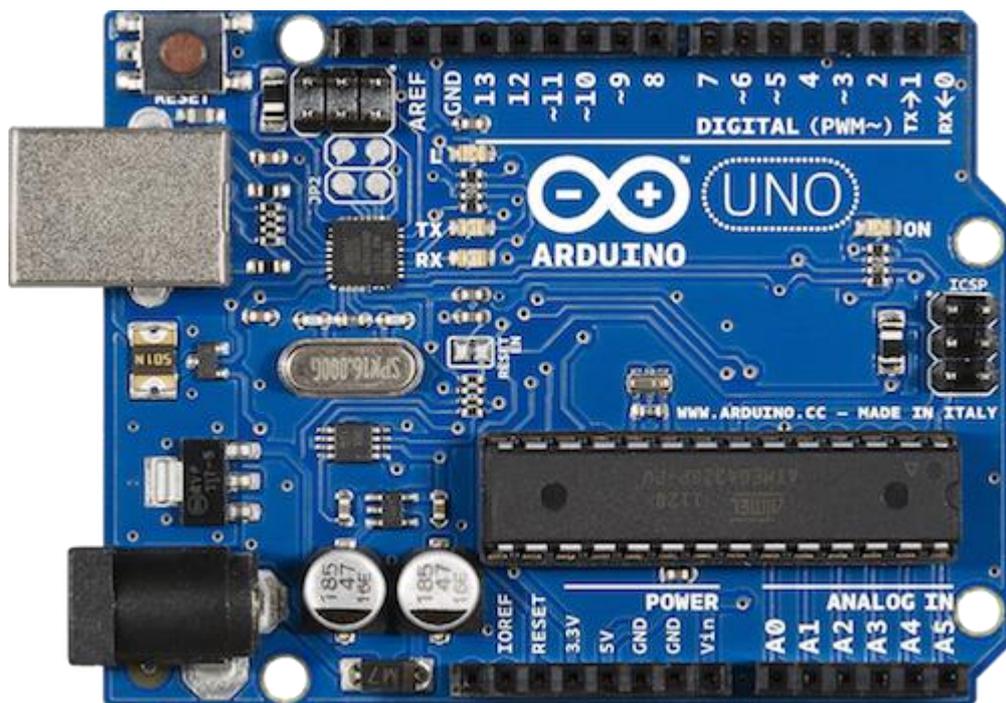


MICROCONTRÔLEUR ARDUINO

POUR LES PLP

MATHÉMATIQUES - PHYSIQUE CHIMIE



4^{ème} version

Jean-François THULLIER – Mars 2020

SOMMAIRE

Introduction.....	3
Partie A : Découverte et prise en main d'ARDUINO.....	4
I – Quelques présentations avant de commencer	
1) L'ARDUINO, qu'est-ce que c'est ?	5
2) Le matériel nécessaire	6
3) Présentation de la carte ARDUINO UNO.....	7
4) Présentation de la plaque de montage (Breadboard).....	7
5) Présentation du logiciel Arduino IDE	8
6) Rappel (plutôt utile) sur la LED	9
II – Premiers montages avec des LED	
1) Faire allumer une LED	10
2) Faire clignoter une LED	11
3) Faire clignoter 10 fois une LED	11
4) Faire clignoter quatre LED.....	12
5) Réaliser un chenillard à 4 LED (ou « l'œil de K2000 »).....	13
6) Les feux de circulation	14
III – Les entrées numériques : exemple du bouton poussoir	
1) Préambule.....	16
2) Actionner l'allumage d'une LED avec un bouton poussoir	16
IV – Les entrées analogiques	
1) Exemple du potentiomètre.....	18
2) Exemple de la thermistance.....	20
V – Utilisation des bibliothèques de code	
1) Exemple du servomoteur.....	22
2) Prolongement : barrière avec feu bicolore et bouton poussoir.....	23
Partie B : Applications en physique-chimie.....	26
I – Utilisation d'une LED RGB (module optique – seconde bac pro)	27
II - Utilisation du module ultrasons (module signaux – première ou terminale bac pro)	
1) Mesurer une distance	30
2) Créer un détecteur de présence ou d'obstacle	31
III – Utilisation d'une photorésistance ou du module photorésistance (module optique – seconde bac pro)	
1) Faire clignoter une LED avec une vitesse proportionnelle à l'éclairage ambiant.....	33
2) Réaliser un système d'éclairage automatique.....	34
IV – Utilisation d'un haut-parleur : produire la note La ₃ (module acoustique – seconde bac pro)	37
V – Utilisation du module capteur sonore (module acoustique – seconde bac pro)	39
Annexes	41
Annexe 1 : Simulation d'un dé à 6 faces.....	42
Annexe 2 : Matériels supplémentaires pour aller plus loin	44
Webographie	45

PARTIE A :

**DÉCOUVERTE ET
PRISE EN MAIN
D'ARDUINO**

I – Quelques présentations avant de commencer

1) L'ARDUINO, qu'est-ce que c'est ?

ARDUINO est une carte électronique qui comporte un **microcontrôleur**, c'est-à-dire une carte électronique programmable par un ordinateur. Elle utilise un même logiciel de programmation (environnement de développement ou IDE) appelé également Logiciel Arduino. Elle peut ensuite fonctionner seule si elle est alimentée en énergie.

On peut y brancher des capteurs (températures, humidité, pression, présence, distance, position, luminosité, ...) et des actionneurs (LED, moteurs, lampe, résistance chauffante, ...)

Comme ARDUINO est « open source », il existe un grand nombre de clones et de platines compatibles.



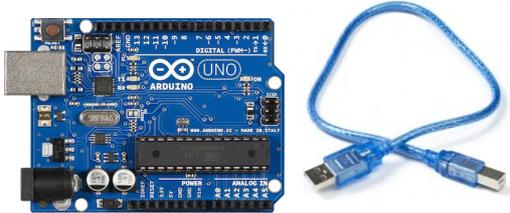
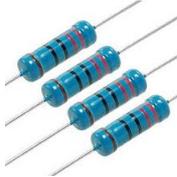
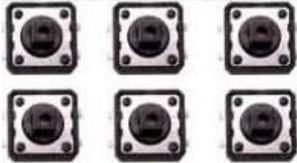
Ici, nous utiliserons le plus connu : l'ARDUINO UNO.

Pour plus de facilités, on peut acheter un **KIT STARTER**.

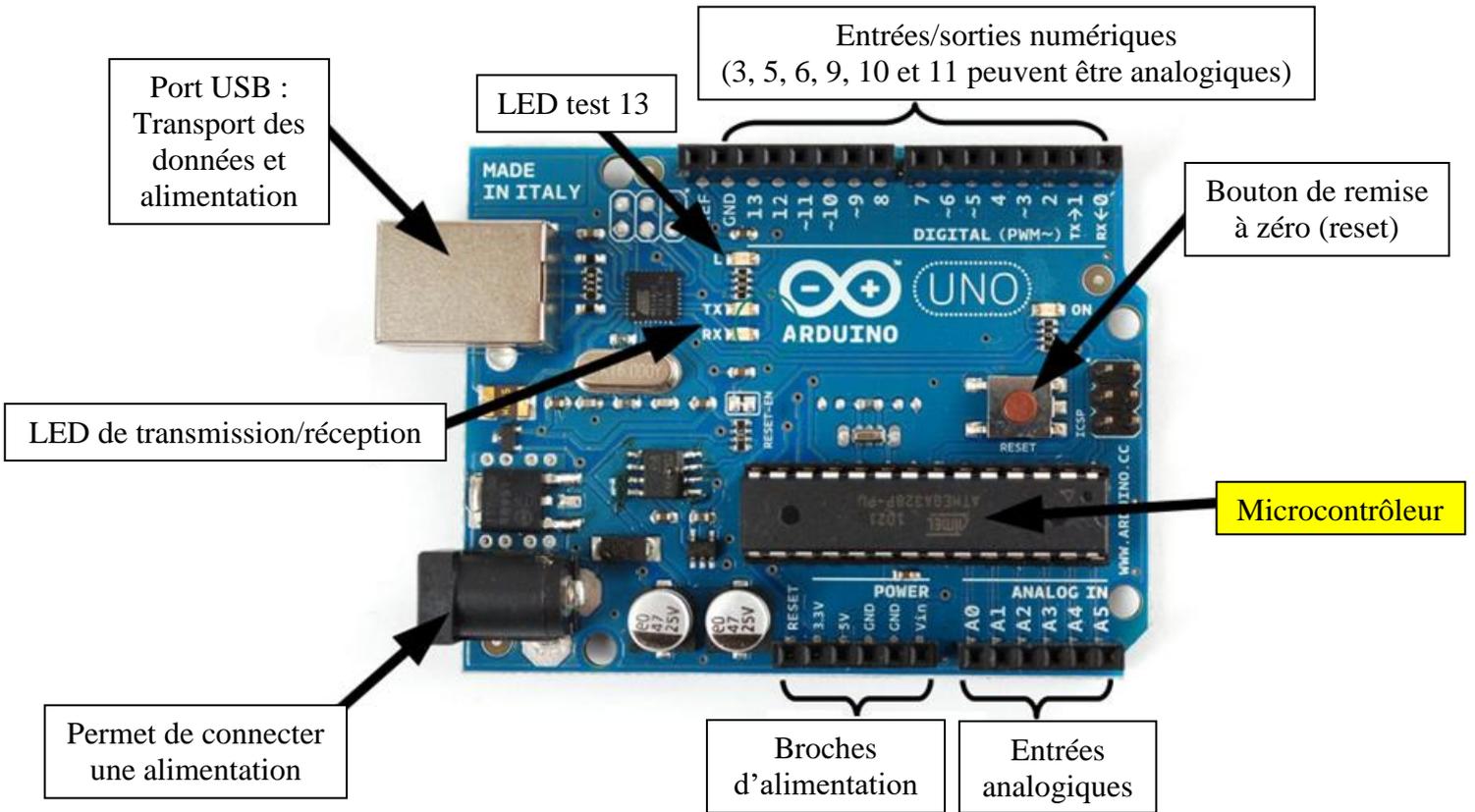
Par exemple : https://www.cdiscount.com/informatique/cartes-meres/kit-starter-arduino-uno-r3-avec-support-step-motor/f-10765-usi1909073329614.html#cm_rr=FP:7583423:SP:CAR
(48,88 euros au lieu de 208,88 € le 23 octobre 2018)

Conseil sur la carte Arduino Uno : pour éviter qu'un fil ou qu'un composant branché au + vienne endommager un port USB dans l'ordinateur, on peut isoler le métal du port USB avec un scotch d'électricien. Attention également au dessous de la carte, il ne faut pas la poser sur un support conducteur.

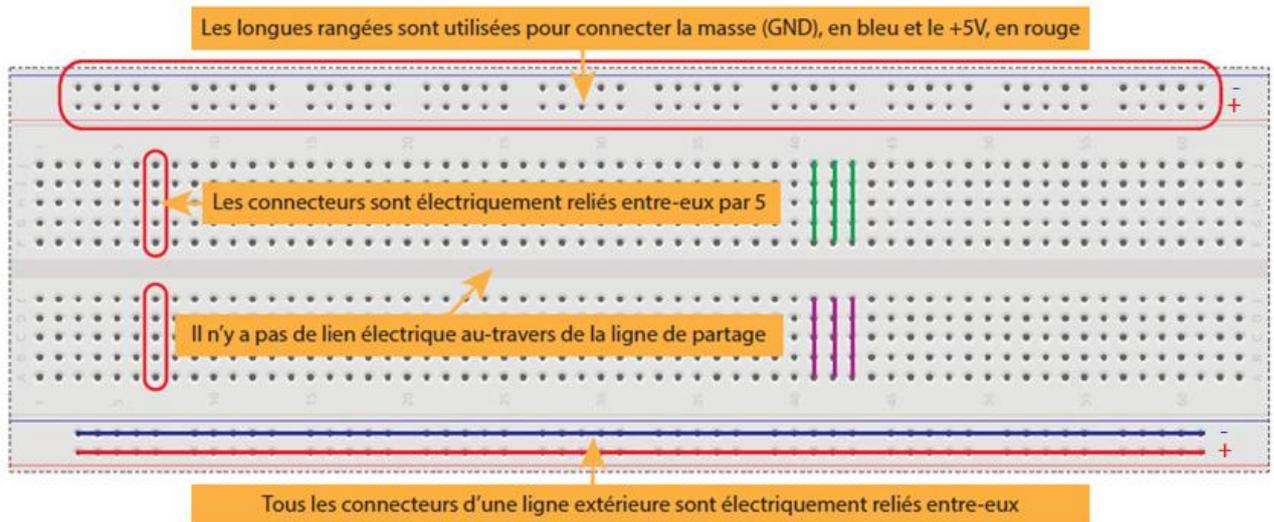
2) Le matériel nécessaire

 <p>Carte Arduino Uno et son câble USB A/B</p>	 <p>Plaque de montage (Breadboard) et câbles de connexion (Jumpers)</p>
 <p>Différentes LED</p>	 <p>Différentes résistances (dont 220 Ω et 10 kΩ)</p>
 <p>Boutons poussoir</p>	 <p>Potentiomètre analogique (10 kΩ)</p>
 <p>Thermistance CTN $R_0 = 10 \text{ k}\Omega$ à 25°C, $P = 0,25 \text{ W}$ et $B = 4300$</p>	 <p>Tower Pro Micro Servo 9g SG 90</p>
 <p>LED RGB (à anode ou à cathode commune)</p>	 <p>Module ultrasons (capteur de distance)</p>
 <p>Photorésistances ou module photorésistance</p>	 <p>Haut-parleur miniature - 8 Ω - 5 W</p>
 <p>Module capteur sonore</p>	 <p>Logiciel Arduino IDE (gratuit) Logiciel Fritzing (gratuit)</p>

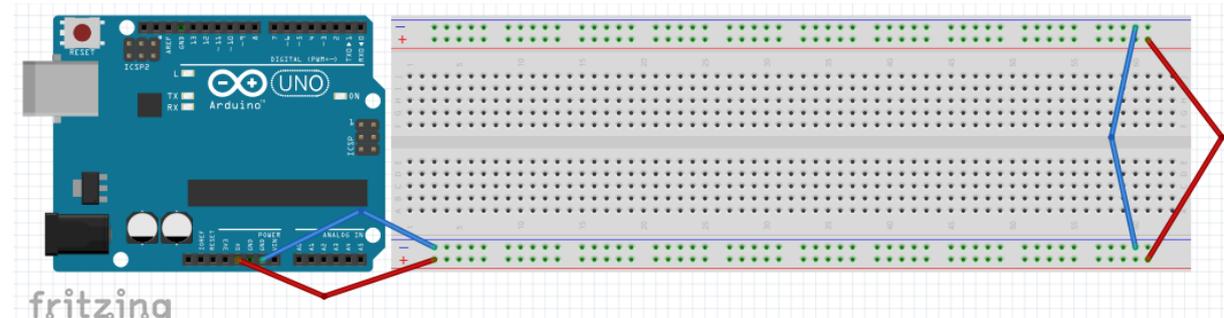
3) Présentation de la carte ARDUINO UNO



4) Présentation de la plaque de montage (Breadboard)



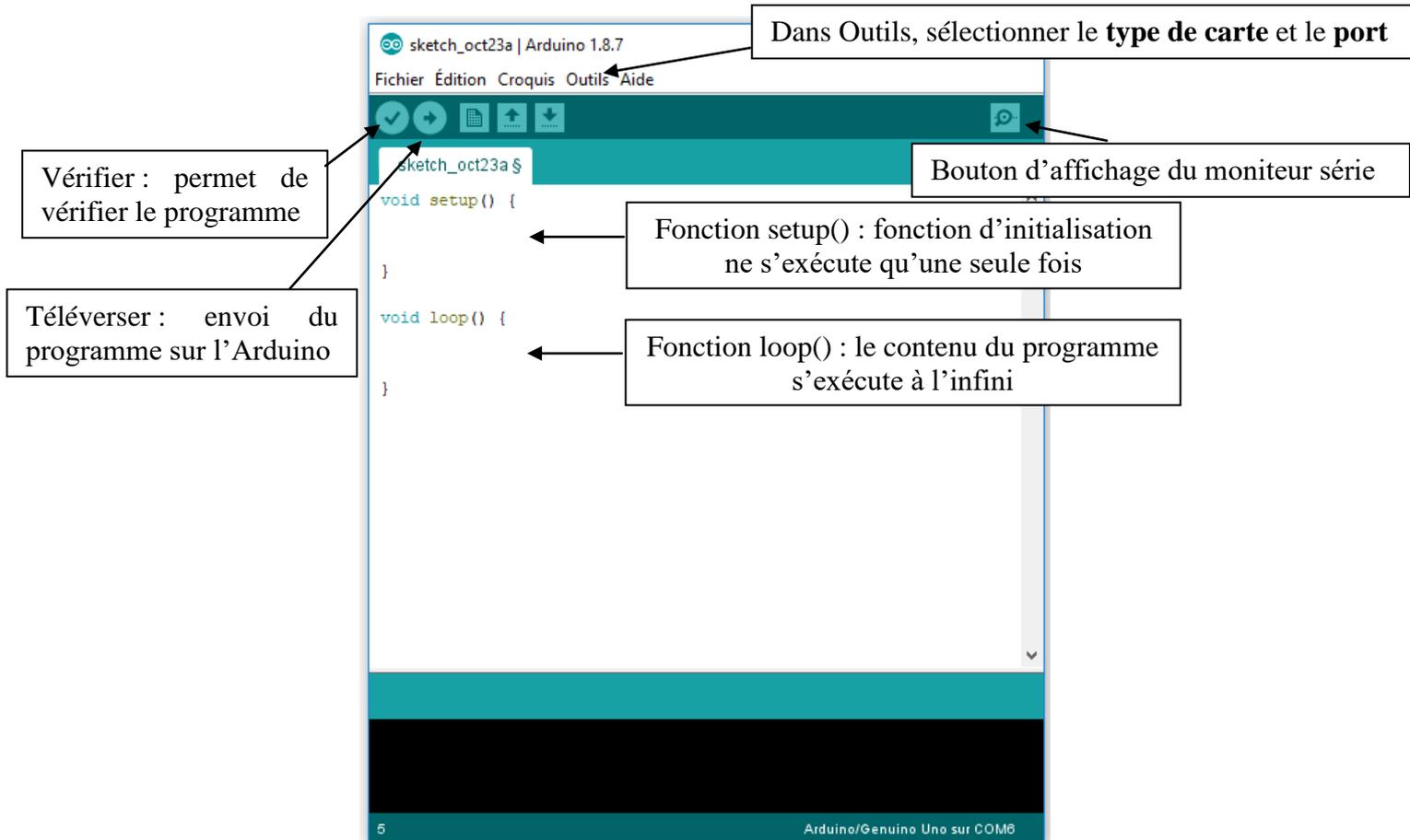
Tout ce qui sera connecté aux trous liés à +5 V recevra du +5 V et tout ce qui sera connecté aux trous liés à GND ira vers le 0 V.



5) Présentation du logiciel Arduino IDE

Il est gratuit et téléchargeable sur le site officiel Arduino : <https://www.arduino.cc/en/Main/Software>

Le langage utilisé pour programmer le microcontrôleur est basé sur les langages C/C++.

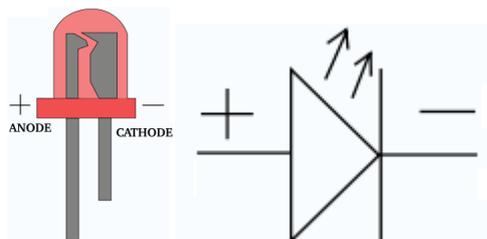


- Les accolades sont les « conteneurs » du code du programme.
- Les points virgules terminent les instructions.
- Les commentaires (lignes de codes ignorées par le programme) s'effectuent de la manière suivante :
 - // ligne unique de commentaire
 - /* ligne ou paragraphe sur plusieurs lignes */

Syntaxe du langage Arduino :

Commandes de structure du programme	Variables
<p>Contrôle et conditions</p> <ul style="list-style-type: none"> • if (si...) • if...else (si...alors...) • for (pour...) • while (pendant que ...) <p>Opérations de comparaison</p> <ul style="list-style-type: none"> • == (équivalent à) • != (différent de) • < (inférieur à) • > (supérieur à) • <= (inférieur ou égal à) • >= (supérieur ou égal à) <p>Opérations booléennes</p> <ul style="list-style-type: none"> • && (et) • (ou) • ! (et pas) 	<p>Variables</p> <ul style="list-style-type: none"> • char (variable « caractère ») • int (variable « nombre entier ») • long (variable « nombre entier de très grande taille ») • float (variable « décimal ») • boolean (true ou false) <p>Niveaux logiques des connecteurs numériques</p> <ul style="list-style-type: none"> • HIGH (état 1) • LOW (état 0) • INPUT (configuré en entrée) • OUTPUT (configuré en sortie)
	Fonctions
	<p>Entrées-sorties numériques</p> <ul style="list-style-type: none"> • pinMode(broche, état) (configuration des broches) • digitalWrite(broche, état) (écrire un état sur une broche numérique) • digitalRead(broche) (lire un état sur une broche numérique) <p>Gestion du temps</p> <ul style="list-style-type: none"> • delay(ms) (attente, en millisecondes) • delayMicroseconds(us) (attente, en microsecondes)

6) Rappel (plutôt utile) sur la LED



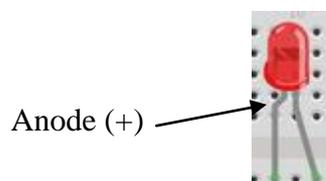
Une patte est plus longue que l'autre : la courte est la cathode (moyen mnémotechnique)

Si l'on veut qu'une LED s'allume, on connecte la patte la plus longue (anode ou +) du côté du 5 V et la patte la plus courte (cathode ou -) du côté du 0 V ou GROUND (GND).

ATTENTION : pour connecter une LED, il faut lui joindre une **résistance** d'au moins 100 Ω , **220 Ω** est souvent la valeur conseillée.

Explications : D'après la loi des mailles, on a : $U_{gen} = U_{led} + U_{res}$ avec $U_{gen} = 5 \text{ V}$ (tension aux bornes de l'Arduino) et $U_{led} = 1,8 \text{ V}$ donc : $U_{res} = 5 \text{ V} - 1,8 \text{ V} = 3,2 \text{ V}$. L'intensité de fonctionnement qui traverse une LED est d'environ 20 mA. On applique maintenant la loi d'Ohm : $R = U/I = 3,2/0,02 = 160 \Omega$ donc 220 Ω est un bon choix.

Avec le logiciel Fritzing, la patte la plus longue (anode ou +) de la LED est représentée par un petit décroché :



II – Premiers montages avec des LED

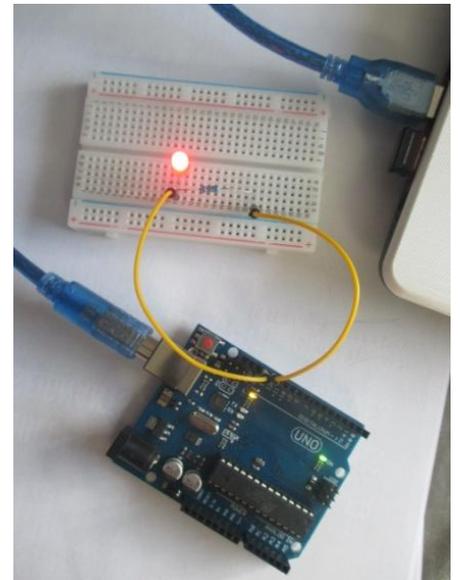
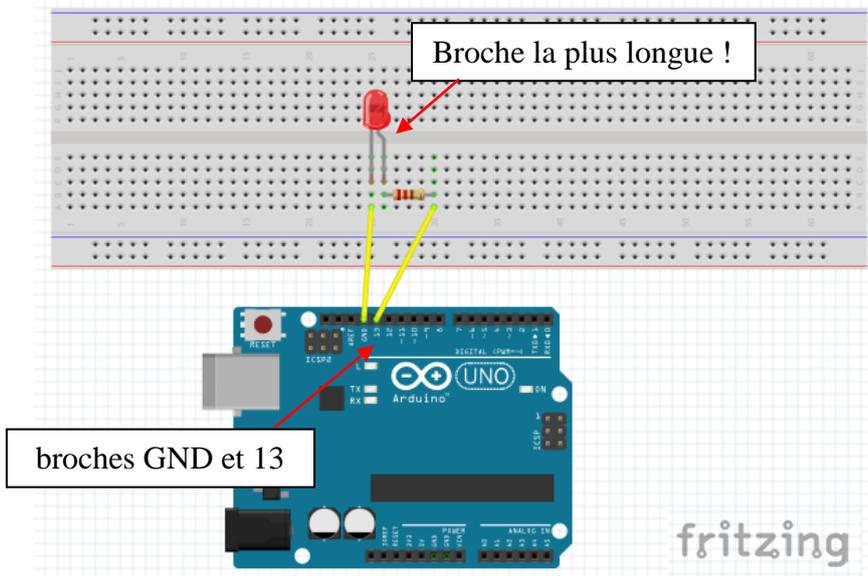
1) Faire allumer une LED

Nous allons faire allumer une LED, connectée sur la broche 13.

La résistance utilisée est de 220 Ω (comme les résistances de tous les prochains montages).

Voici le schéma de montage (réalisé avec le logiciel Fritzing) :

En photo :



Dans le logiciel Arduino IDE, la programmation se fait de la façon suivante (au choix) :

Sans variable	Avec variable
<pre>void setup() { pinMode(13, OUTPUT); } void loop() { digitalWrite(13, HIGH); }</pre>	<pre>int led1=13; void setup() { pinMode(led1, OUTPUT); } void loop() { digitalWrite(led1, HIGH); }</pre>

On appuyera sur  (vérifier) pour vérifier que le programme fonctionne puis sur  (téléverser) pour envoyer le programme sur Arduino.

On peut trouver que définir les broches allonge le code mais si on veut changer la broche utilisée, il suffit de corriger la variable au départ, sans devoir corriger tout le code. De plus, quand nous aurons de nombreuses broches en fonction, cela nous permettra de les identifier plus facilement.

Analyse du code :

- L'instruction ***pinMode(led1, OUTPUT);*** initialise la broche 13 du microcontrôleur comme sortie, c'est-à-dire que des données seront envoyées depuis le microcontrôleur vers cette broche (on va envoyer de l'électricité).
- Avec l'instruction ***digitalWrite(led1, HIGH);*** le microcontrôleur connecte la broche 13 au +5V ce qui a pour effet d'allumer la LED (de l'électricité sort de la broche 13).

2) Faire clignoter une LED

Il s'agit du même montage.

La programmation est la suivante :

```
int led1=13;

void setup()
{
  pinMode(led1, OUTPUT);
}

void loop()
{
  digitalWrite(led1, HIGH);
  delay(1000);
  digitalWrite(led1, LOW);
  delay(1000);
}
```

Analyse du code :

- L'instruction ***delay(1000);*** indique au microcontrôleur de ne rien faire pendant 1 000 millisecondes, soit 1 seconde.
- Avec l'instruction ***digitalWrite(led1, LOW);*** le microcontrôleur connecte la broche 13 à la masse (GND) ce qui a pour effet d'éteindre la LED (on coupe l'alimentation en électricité).

Pour voir la vidéo correspondante :



3) Faire clignoter 10 fois une LED

Il s'agit du même montage.

La programmation est la suivante :

```
int led1=13;
int compteur;

void setup()
{
  pinMode(led1, OUTPUT);

  for(compteur=0;compteur<10;compteur++)
  {
    digitalWrite(led1, HIGH);
    delay(1000);
    digitalWrite(led1, LOW);
    delay(1000);
  }
}

void loop()
{
}
```

Analyse du code :

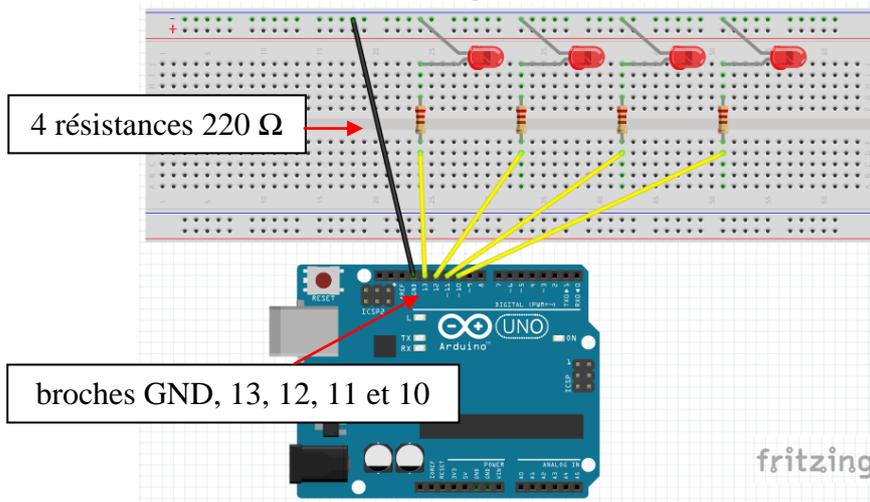
- L'instruction ***compteur++*** peut remplacer l'instruction ***compteur=compteur+1.***
- La boucle ***for*** va être exécutée 10 fois (le compteur part de 0 puis 1, puis 2, ... puis 9).
- La fonction ***setup()*** n'est exécutée qu'une fois donc si on veut que la LED s'éteigne après les 10 clignotements, il faut faire exécuter la boucle dans cette fonction. Si nous l'avions mis dans la fonction ***loop()***, le clignotement ne se serait jamais arrêté (s'exécute à l'infini).

Pour voir la vidéo correspondante :

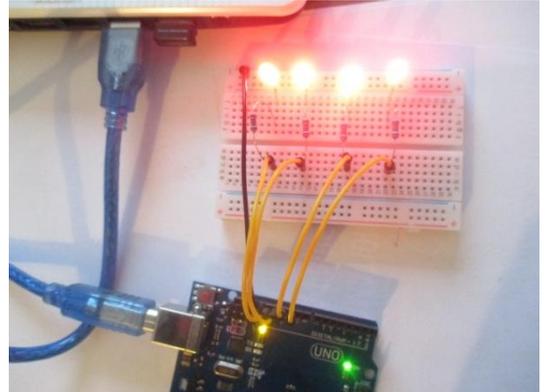


4) Faire clignoter quatre LED

Voici le schéma de montage :



En photo :



La programmation est la suivante :

Sans boucle	Avec une boucle <i>for</i>
<pre>int led1=13; int led2=12; int led3=11; int led4=10; void setup() { pinMode(led1, OUTPUT); pinMode(led2, OUTPUT); pinMode(led3, OUTPUT); pinMode(led4, OUTPUT); } void loop() { digitalWrite(led1, HIGH); digitalWrite(led2, HIGH); digitalWrite(led3, HIGH); digitalWrite(led4, HIGH); delay(1000); digitalWrite(led1, LOW); digitalWrite(led2, LOW); digitalWrite(led3, LOW); digitalWrite(led4, LOW); delay(1000); }</pre>	<pre>int led; void setup() { for (led = 10; led < 14; led++) { pinMode(led, OUTPUT); } } void loop() { for (led = 10; led < 14; led++) { digitalWrite(led, HIGH); } delay(1000); for (led = 10; led < 14; led++) { digitalWrite(led, LOW); } delay(1000); }</pre>

Pour voir la vidéo correspondante :



5) Réaliser un chenillard à 4 LED (ou « l'œil de K2000 »)

C'est le même montage.

La programmation est la suivante :

Sans boucle	Avec une boucle <i>for</i>	Avec un tableau
<pre>void setup() { pinMode(10, OUTPUT); pinMode(11, OUTPUT); pinMode(12, OUTPUT); pinMode(13, OUTPUT); } void loop() { digitalWrite(10, HIGH); digitalWrite(11, LOW); digitalWrite(12, LOW); digitalWrite(13, LOW); delay(100); digitalWrite(10, LOW); digitalWrite(11, HIGH); delay(100); digitalWrite(11, LOW); digitalWrite(12, HIGH); delay(100); digitalWrite(12, LOW); digitalWrite(13, HIGH); delay(100); digitalWrite(13, LOW); digitalWrite(12, HIGH); delay(100); digitalWrite(12, LOW); digitalWrite(11, HIGH); delay(100); }</pre>	<pre>int led; void setup() { for (led = 10; led < 14; led++) { pinMode(led, OUTPUT); } } void loop() { for (led = 10; led < 14; led++) { digitalWrite(led, HIGH); delay(100); digitalWrite(led, LOW); } for (led = 13; led > 9; led--) { digitalWrite(led, HIGH); delay(100); digitalWrite(led, LOW); } }</pre>	<pre>int led[4] = {10, 11, 12, 13}; int i; void setup() { for (i = 0; i < 4; i++) { pinMode(led[i], OUTPUT); } } void loop() { for (i = 0; i < 4; i++) { digitalWrite(led[i], HIGH); delay(100); digitalWrite(led[i], LOW); } for (i = 3; i >= 0; i--) { digitalWrite(led[i], HIGH); delay(100); digitalWrite(led[i], LOW); } }</pre>

Analyse du code :

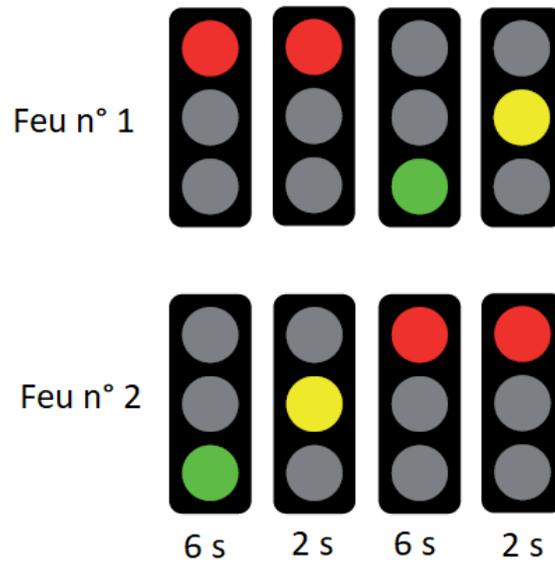
- L'instruction *led--* peut remplacer l'instruction *led=led-1*.
- Un tableau (3^{ème} version) permet de stocker des variables dans un même endroit au lieu de les créer séparément et leur donner des noms différents. Attention, la position de la dernière case d'un tableau n'est pas la taille du tableau, mais sa taille moins 1. C'est parce que la première case d'un tableau est numérotée à 0, pas à 1.

Pour voir la vidéo correspondante :

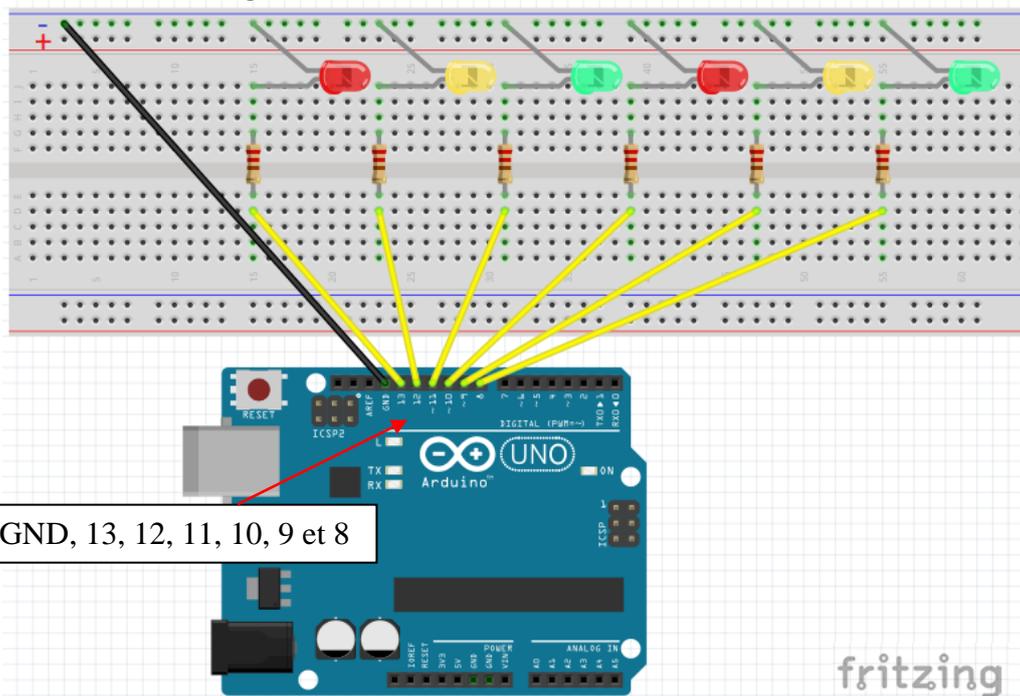


6) Les feux de circulation

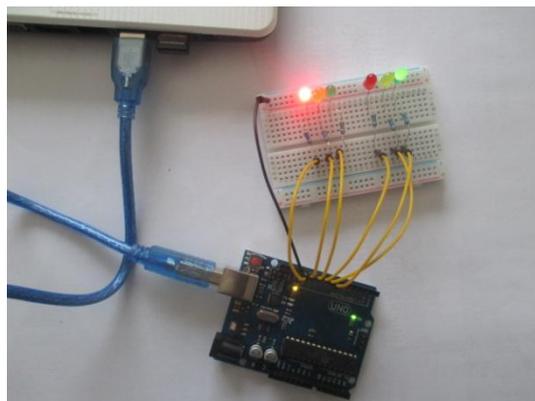
L'objectif de cet exercice est de créer deux feux de circulation et de les faire fonctionner selon les phases suivantes :



Voici le schéma de montage :



En photo :



La programmation est la suivante :

```
//FEU 1
int R1 = 13;
int J1 = 12;
int V1 = 11;
//FEU2
int R2 = 10;
int J2 = 9;
int V2 = 8;
//TEMPS
int temps1 = 6000;
int temps2 = 2000;
void setup()
{
  pinMode(R1, OUTPUT);
  pinMode(J1, OUTPUT);
  pinMode(V1, OUTPUT);
  pinMode(R2, OUTPUT);
  pinMode(J2, OUTPUT);
  pinMode(V2, OUTPUT);
}
void loop()
{
  //Phase 1: R1 et V2 allumés, J1, V1, R2 et J2 éteints, 6 s
  digitalWrite(R1, HIGH);
  digitalWrite(V2, HIGH);
  digitalWrite(J1, LOW);
  digitalWrite(V1, LOW);
  digitalWrite(R2, LOW);
  digitalWrite(J2, LOW);
  delay(temps1);
  //Phase 2: J2 allumé, V2 éteint, 2 s
  digitalWrite(J2, HIGH);
  digitalWrite(V2, LOW);
  delay(temps2);
  //Phase 3: V1 et R2 allumés, R1 et J2 éteint, 6 s
  digitalWrite(V1, HIGH);
  digitalWrite(R2, HIGH);
  digitalWrite(R1, LOW);
  digitalWrite(J2, LOW);
  delay(temps1);
  //Phase 4: J1 allumé, V1 éteint, 2 s
  digitalWrite(J1, HIGH);
  digitalWrite(V1, LOW);
  delay(temps2);
}
```

On pourrait passer au carrefour à 4 feux mais cela multiplie les LED et les résistances sur la plaque de montage !

Pour voir la vidéo correspondante :



III – Les entrées numériques : exemple du bouton poussoir

1) Préambule

Jusqu'alors, nous avons traité des sorties (outputs), c'est-à-dire qu'un signal sortait du microcontrôleur sous la forme d'un courant électrique (HIGH) ou de son absence (LOW) grâce à la commande DigitalWrite. On a utilisé jusqu'ici ce signal (un courant électrique) pour allumer des LED.

L'utilisation d'un bouton poussoir, et plus généralement d'un contacteur, va amener à gérer non plus les sorties, mais les **entrées (inputs)** de la carte Arduino : il va être possible d'envoyer un signal au microcontrôleur depuis ce bouton poussoir. Ce signal est un courant électrique entrant dans la broche. En fonction du signal reçu, le microcontrôleur effectuera une tâche prévue : allumer une LED lorsqu'on appuiera sur le bouton poussoir par exemple. Pour cela, nous utiliserons la commande DigitalRead.



Un bouton poussoir peut se trouver sous plusieurs formes dans le commerce. Celui qui nous intéresse est un bouton avec 4 pattes, une forme carrée, et un rond au centre qui est le bouton lui-même.

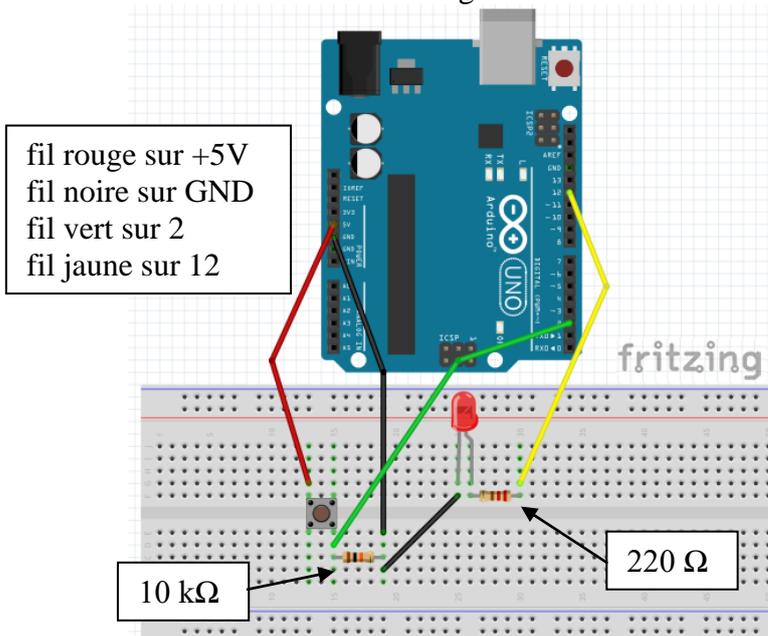
Attention, même s'il a 4 pattes, le bouton poussoir est un dipôle. En fait les pattes sont reliées deux par deux. Si la LED reste allumée en permanence, il faut tourner le bouton d'un quart de tour.

Tous les montages suivants contiennent une résistance de 10 k Ω , qu'on appelle **pull-down** : cette résistance permet de **tirer** le potentiel vers le **bas**. Le but d'une résistance de pull-down est donc d'évacuer les courants vagabonds et de donner un signal clair.

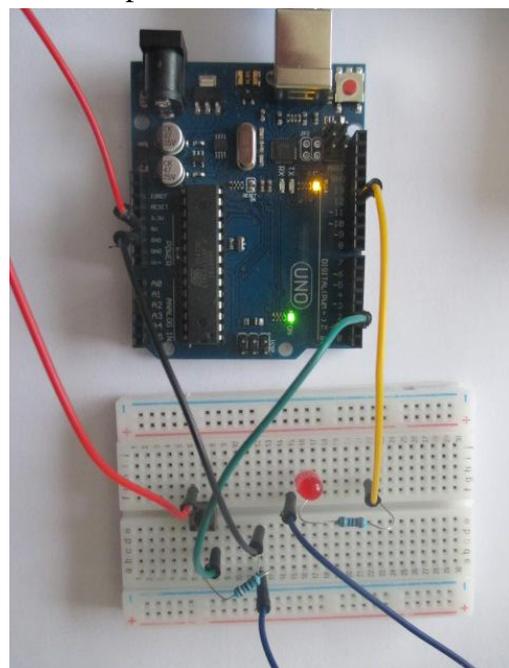
2) Actionner l'allumage d'une LED avec un bouton poussoir

Lorsqu'on appuie sur le bouton poussoir, La LED doit s'allumer. Et naturellement, lorsqu'on relâche le bouton poussoir, la LED s'éteint. Cette action n'est pas mécanique, mais logique. Ce n'est pas la fermeture d'un circuit électrique qui allume la LED, mais l'information transmise à l'Arduino, par le biais d'un **INPUT** qui lui ordonne d'allumer la LED.

Voici le schéma de montage :



En photo :



La programmation est la suivante :

Version classique	Version « élégante »
<pre>int bouton = 2; int led = 12; void setup() { pinMode(led, OUTPUT); pinMode(bouton, INPUT); } void loop() { if (digitalRead(bouton) == HIGH) { digitalWrite(led, HIGH); } else { digitalWrite(led, LOW); } }</pre>	<pre>int bouton = 2; int led = 12; boolean boutonEtat = 0; void setup() { pinMode(led, OUTPUT); pinMode(bouton, INPUT); } void loop() { boutonEtat = digitalRead(bouton); if (boutonEtat == 1) { digitalWrite(led, HIGH); } else { digitalWrite(led, LOW); } }</pre>

Analyse du code :

- ***pinMode(led, OUTPUT)***; initialise la broche 12 comme entrée, ***pinMode(bouton, INPUT)***; initialise la broche 2 en sortie.
- Le type ***boolean*** est approprié pour la variable boutonEtat car elle ne prend que les valeurs 1 ou 0 (HIGH ou LOW) ;

Pour voir la vidéo correspondante :



IV – Les entrées analogiques

Jusqu' alors, nous avons utilisé des entrées numériques qui ne prennent que deux valeurs : 0 ou 1. Nous allons désormais découvrir comment utiliser des entrées analogiques qui permettent de recevoir des données électriques plus subtiles.

Il y a **6 entrées analogiques notées A0, A1, A2, A3, A4, et A5.**

En réalité, le microcontrôleur n'est pas capable de comprendre un signal analogique. Il faut donc d'abord le convertir en signal numérique par un convertisseur analogique/numérique (CAN). Ce convertisseur va échantillonner le signal reçu sous la forme d'une variation de tension et le transformer en valeurs comprises entre 0 et 1023.

Attention à ne pas faire entrer une tension supérieure à 5V, ce qui détruirait l'Arduino.

Cette découverte des entrées analogiques se fera à travers la prise en main du **potentiomètre analogique** et de la **thermistance**.

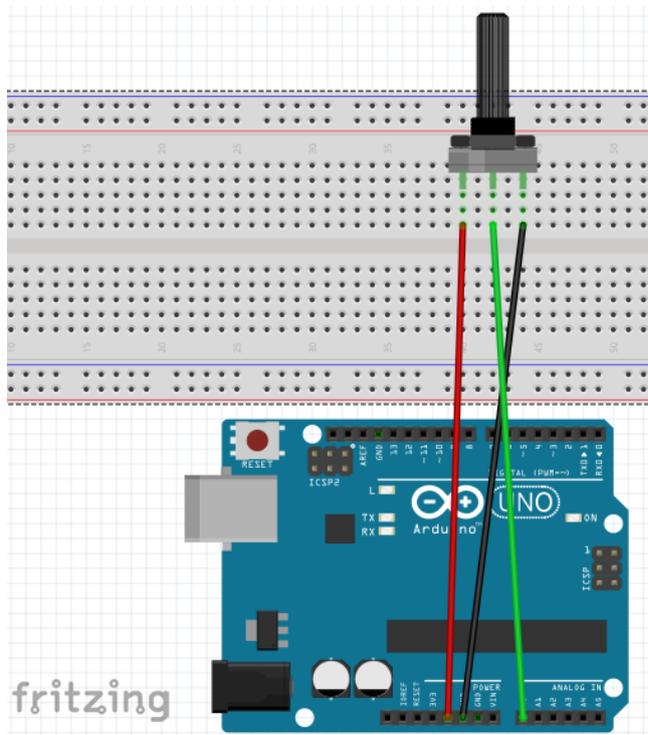
1) Exemple du potentiomètre

Le branchement est simple :

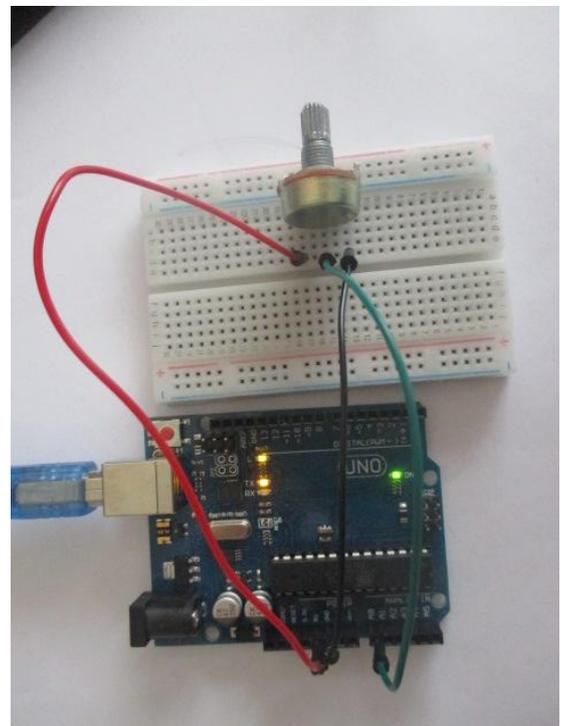
- l'une des extrémités au +5V,
- l'autre au GND,
- et le curseur à l'entrée analogique A0.



Schéma du montage :

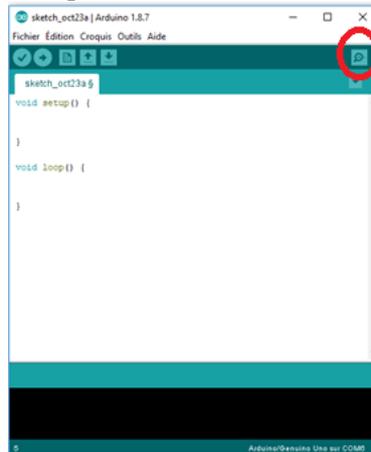


En photo :



fil rouge sur +5V, fil noir sur GND, fil vert sur A0

Pour récupérer la valeur, il faudra cliquer sur le **bouton d'affichage du moniteur série**.



La programmation est la suivante :

```
int valeurPot;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  valeurPot=analogRead(A0);
  Serial.println(valeurPot);
}
```

Analyse du code :

- ***analogRead(0)***; lit la donnée à l'entrée analogique A0
- ***Serial.begin(9600)***; initialise la communication série avec une vitesse à 9 600 bauds (c'est-à-dire maximum 9600 caractères par seconde).
- ***Serial.println*** affiche avec retour à la ligne.

En tournant la molette du potentiomètre, on peut remarquer que la valeur varie de 0 à 1 023.

Le pin analogique transforme donc une tension reçue entre 0 et 5 V en valeur entre 0 et 1 023.

Pour voir la vidéo correspondante :



Pour faire afficher la tension :

```
int valeurPot;
float U;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  valeurPot=analogRead(A0);
  U=5.0*valeurPot/1023;
  Serial.println(U);
}
```

Analyse du code :

Pour récupérer la tension aux bornes du potentiomètre, il faut saisir 5.0 et non 5.

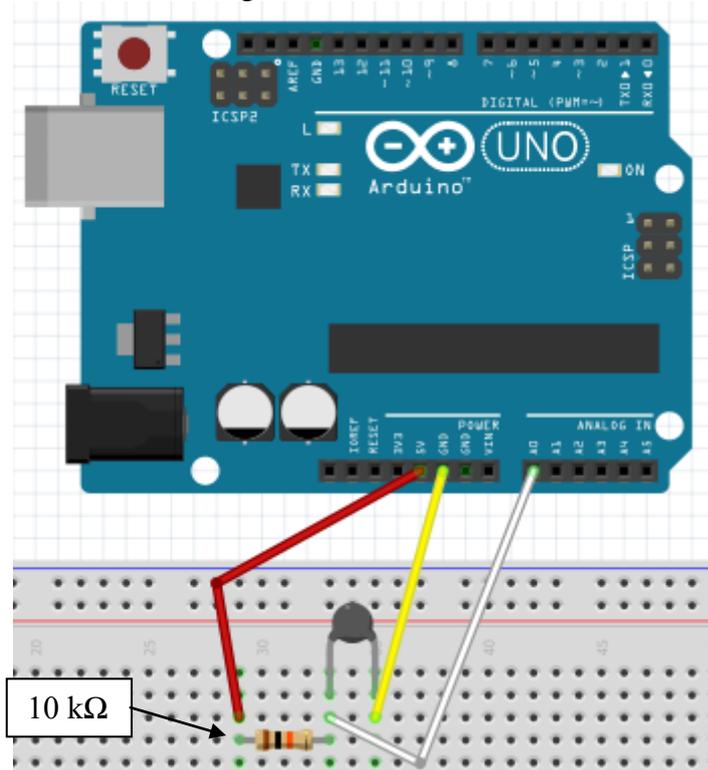
2) Exemple de la thermistance

La thermistance est une sonde de température dont la résistance varie en fonction de la température. Pour une CTN, sa résistance décroît lorsque la température augmente.

Nous en utilisons une dont les caractéristiques sont : $R_0 = 10 \text{ k}\Omega$ à 25°C , $P = 0,25 \text{ W}$ et $B = 4300$.

Si j'utilise une valeur inférieure à $10 \text{ k}\Omega$, la résistance va chauffer, ce qui faussera la mesure. Si j'utilise une valeur supérieure, on perd en précision sur le convertisseur analogique de l'Arduino.

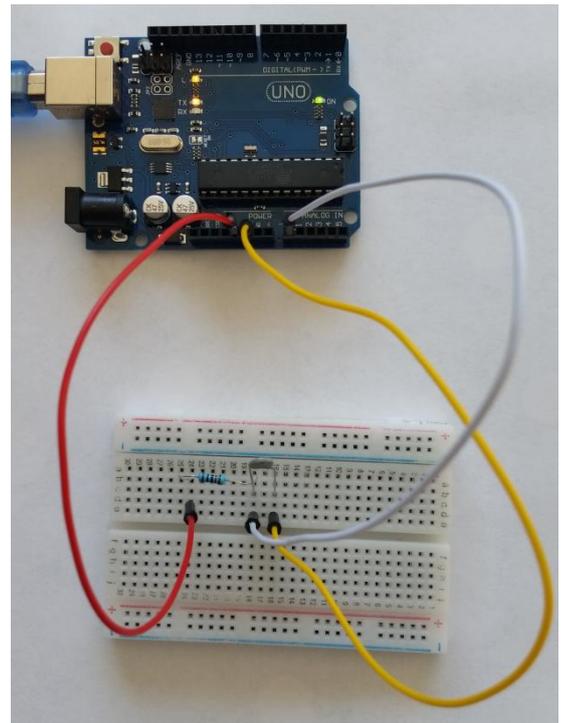
Schéma du montage :



10 kΩ

fil rouge sur +5V, fil jaune sur GND, fil blanc sur A0

En photo :



La résistance additionnelle doit être au moins équivalente à la plus forte valeur de la thermistance : nous prendrons donc ici une résistance de $10 \text{ k}\Omega$.

C'est un pont diviseur de tension donc
$$R = \frac{R_1}{V_{ref} - U} \cdot U = \frac{10}{5 - U} \cdot U$$

La température T (en kelvin) modifie la valeur de la résistance par la relation :

$$R = R_0 e^{B \left(\frac{1}{T} - \frac{1}{T_0} \right)}$$

donc
$$T(K) = \frac{1}{\frac{1}{T_0} + \frac{1}{B} \ln \left(\frac{R}{R_0} \right)}$$
 avec $R_0 = 10 \text{ k}\Omega$ et $B = 4300$.

$T_0 = 273,15 + 25 = 298,15$ K. Pour convertir le résultat en degré Celsius, il suffit de soustraire au résultat la valeur 273,15. On a donc :

$$T(^{\circ}\text{C}) = \frac{1}{\frac{1}{298,15} + \frac{1}{4300} \ln\left(\frac{R}{10}\right)} - 273,15$$

Pour récupérer la valeur, il faudra cliquer sur le bouton d'affichage du moniteur série comme pour le cas précédent du potentiomètre.

La programmation est la suivante :

```
int Valeur;
float U,R,T;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Valeur=analogRead(A0);
  U=Valeur*5.0/1023;
  R=U*10/(5-U);
  T=1/(1/298.15+log(R/10)/4300)-273.15;
  Serial.print("Température en degré Celcius = ");
  Serial.println(T);
}
```

Analyse du code :

- Pour récupérer la tension aux bornes de la thermistance, il faut saisir 5.0 et non 5.
- L'instruction à utiliser pour le logarithme népérien est log (!).

Pour voir la vidéo correspondante :



V – Utilisation des bibliothèques de code

Les bibliothèques (bibliothèques) pour Arduino sont nombreuses et abordent la plupart des besoins courants. On trouve ainsi les bibliothèques standards, pour par exemple gérer le Wifi, les écrans à cristaux liquides, une carte SD, ou encore des moteurs. Pour l'occasion, nous allons nous intéresser à la bibliothèque de moteurs un peu particuliers que l'on retrouve dans le monde du modélisme : les servomoteurs.

1) Exemple du servomoteur

Pour charger cette bibliothèque et obtenir du même coup la boîte à outils correspondante, il faut saisir l'instruction :

```
#include <Servo.h>
```

Les servomoteurs sont des moteurs pour faire tourner quelque chose jusqu'à une position bien précise et capable de maintenir cette position jusqu'à l'arrivée d'une nouvelle instruction. Ils sont très utilisés dans le modélisme (direction des voitures télécommandées, commande des gouvernes de dérive et de profondeur sur les avions, etc.), mais ont aussi leur place dans la robotique et l'industrie par exemple dans des vannes pour réguler des flux de liquides.

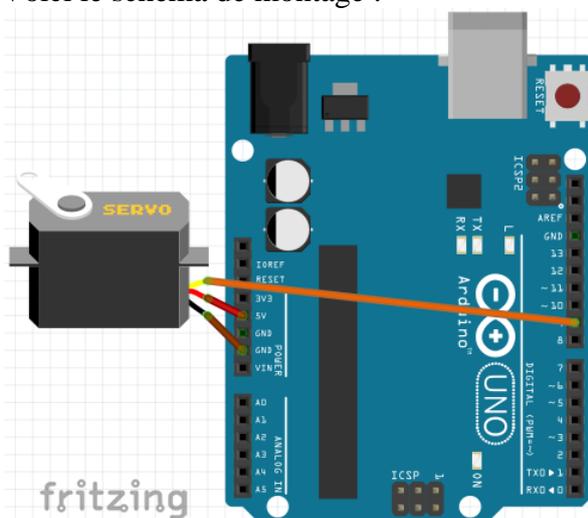
Nous allons utiliser le plus répandu des servomoteurs en modélisme et dans la petite électronique, il s'agit des modèles dits 9g, pour 9 grammes.



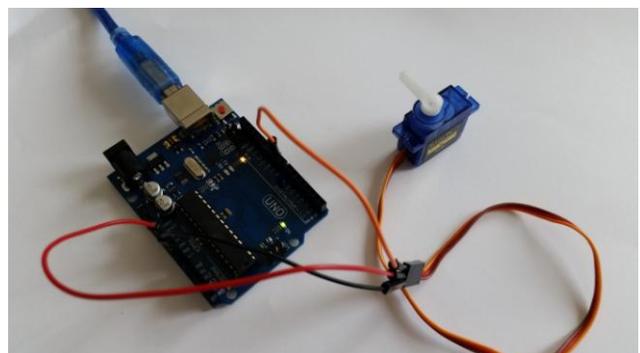
Tower Pro Micro Servo 9g SG 90

La connexion d'un servomoteur ne pose pas de difficulté. Le fil rouge se connecte au 5V, le fil marron (ou noir) se connecte au GND et le fil orange (ou jaune ou blanc) à une sortie numérique de la carte.

Voici le schéma de montage :



En photo :



fil rouge sur +5V, fil marron sur GND, fil orange sur 9

La programmation est la suivante :

```
#include <Servo.h>

Servo monmoteur;
int pos = 0;

void setup()
{
  monmoteur.attach(9);
}

void loop()
{
  for(pos = 0; pos <=180; pos++)
  {
    monmoteur.write(pos);
    delay(15);
  }
  for(pos = 180; pos >=1; pos--)
  {
    monmoteur.write(pos);
    delay(15);
  }
}
```

Analyse du code :

- L'instruction *Servo monmoteur;* crée un objet appelé monmoteur.
- La variable *pos* sera la position courante du servo (elle varie de 0 à 180° puis de 180° à 0°)
- L'instruction *monmoteur.attach(9);* attachera notre objet monmoteur au servomoteur branché sur la broche 9.
- L'instruction *monmoteur.write(pos);* fera aller à la position stocké dans *pos*.
- On laisse quelques millisecondes (ici 15 ms) au moteur afin qu'il ait le temps d'aller une marche plus loin, concrètement de tourner de 1°. On utilise ce délai dans une **boucle** pour ralentir le processus. Plus la valeur du délai sera importante dans le programme, plus le déplacement du moteur sera « doux ».

Pour voir la vidéo correspondante :



2) Prolongement : barrière avec feu bicolore et bouton poussoir

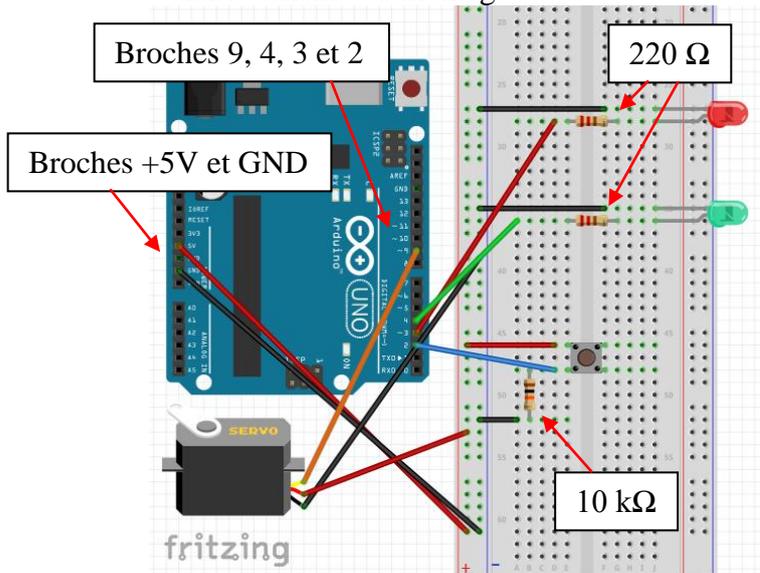
Le montage à réaliser comporte :

- un servomoteur qui jouera le rôle de barrière,
- un bouton pour demander l'ouverture de la barrière,
- un feu bicolore qui passera au vert lorsque la barrière sera complètement ouverte.

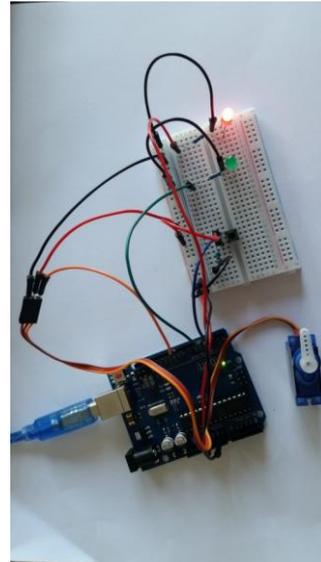
Le fonctionnement normal est un feu allumé au rouge (le feu vert est éteint) et une barrière fermée (0°). Le fonctionnement normal est interrompu par l'appui sur un bouton poussoir.

Si l'appui du bouton est détecté, alors la barrière (actionnée par le servomoteur) se relève doucement. Lorsque la barrière est à la verticale (90°), le feu vert s'allume pendant 5 secondes pendant lesquelles la barrière reste ouverte (90°) et le feu rouge s'éteint. Après les 5 secondes, le feu repasse au rouge, la barrière redescend doucement et le fonctionnement normal reprend.

Voici le schéma de montage :



En photo :



La programmation est la suivante :

```
#include <Servo.h>

Servo barriere;
int pos = 0;
int rouge = 3;
int vert = 4;
int bouton = 2;
int etatbouton = 0;

void setup()
{
  barriere.attach(9);
  pinMode(rouge, OUTPUT);
  pinMode(vert, OUTPUT);
  pinMode(bouton, INPUT);
}

void loop()
{
  digitalWrite(rouge, HIGH);
  digitalWrite(vert, LOW);
  etatbouton = digitalRead(bouton);
  if (etatbouton == HIGH)
  {
    for (pos = 0; pos <= 90; pos++)
    {
      barriere.write(pos);
      delay(15);
    }
    digitalWrite(vert, HIGH);
    digitalWrite(rouge, LOW);
    delay(5000);
  }
}
```

```
digitalWrite(rouge, HIGH);
digitalWrite(vert, LOW);
for (pos = 90; pos >= 0; pos--)
{
  barriere.write(pos);
  delay(15);
}
}
```

Pour voir la vidéo correspondante :

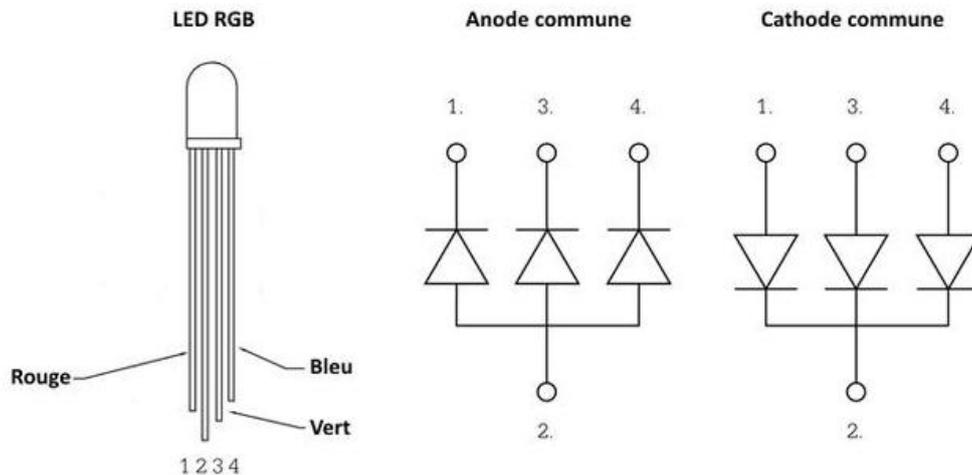


PARTIE B :

APPLICATIONS
EN PHYSIQUE-
CHIMIE

I – Utilisation d'une LED RGB (module optique – seconde bac pro)

L'objectif est d'utiliser une LED RGB (Red Green Blue ou RVB en français) capable de produire les 3 couleurs primaires et de réaliser ainsi la **synthèse additive des couleurs**.



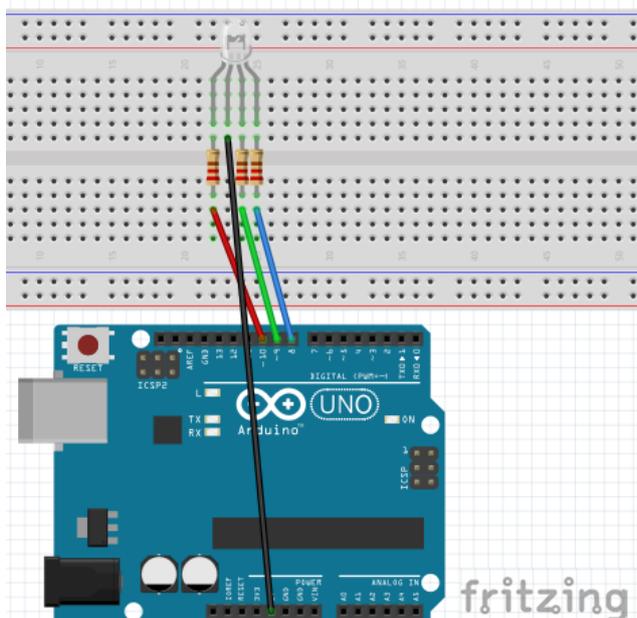
Les LED RGB existent en deux versions : à anode commune ou à cathode commune.

- Dans la version à anode commune, les trois anodes (le « + ») des LED sont reliées ensemble. Cela signifie qu'il faut câbler la tension d'alimentation (5 V) sur la broche commune pour les faire s'allumer.
- Dans la version à cathode commune, les trois cathodes (le « - ») des LED sont reliées ensemble. Cela signifie qu'il faut câbler la masse (GND) sur la broche commune pour les faire s'allumer.

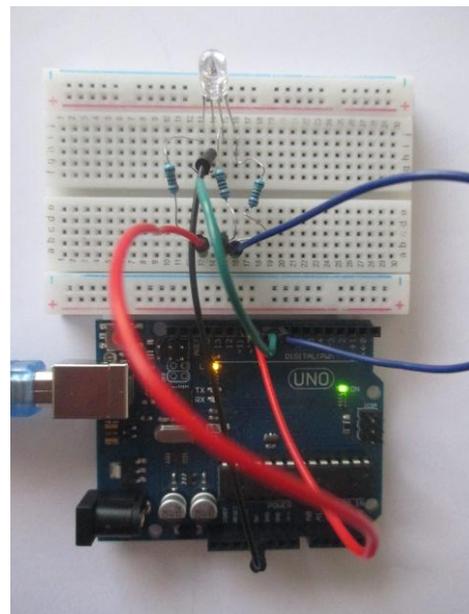
Nous utiliserons ici celle fournie dans le KIT STARTER cité précédemment : **une LED RGB à anode commune** (les plus répandues). Elles sont moins simples à utiliser, car dans cette configuration, une tension de 5 V éteint la LED et une tension de 0 V l'allume. C'est le monde à l'envers !

Nous rajouterons 3 résistances de 220 Ω (une résistance de limitation de courant par LED).

Voici le schéma de montage :



En photo :



fil noir sur +5V, fil rouge sur 10, fil vert sur 9, fil bleu sur 8

On souhaite l'alternance rouge, vert, bleu, magenta, jaune, cyan, blanc.

Comme la LED RGB est à anode commune, il faut raisonner « à l'envers », c'est-à-dire que pour obtenir par exemple la couleur rouge, il faut mettre en LOW la LED rouge et en HIGH les LED bleue et verte. Pour obtenir la couleur magenta (bleu+rouge), il faudra mettre en HIGH la LED verte et en LOW les LED bleue et rouge.

La programmation est la suivante :

```
int ROUGE=10;
int VERT=9;
int BLEU=8;

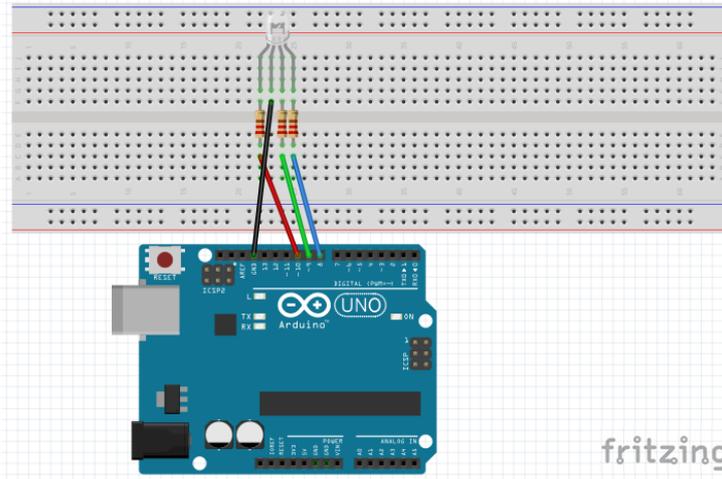
void setup()
{
pinMode(ROUGE, OUTPUT);
pinMode(VERT, OUTPUT);
pinMode(BLEU, OUTPUT);
}

void loop()
{
digitalWrite(ROUGE, LOW); // Rouge
digitalWrite(VERT, HIGH);
digitalWrite(BLEU, HIGH);
delay(1000);
digitalWrite(ROUGE, HIGH); // Vert
digitalWrite(VERT, LOW);
delay(1000);
digitalWrite(VERT, HIGH); // Bleu
digitalWrite(BLEU, LOW);
delay(1000);
digitalWrite(ROUGE, LOW); // B+R=Magenta
delay(1000);
digitalWrite(VERT, LOW); // R+V=Jaune
digitalWrite(BLEU, HIGH);
delay(1000);
digitalWrite(ROUGE, HIGH); // V+B=Cyan
digitalWrite(BLEU, LOW);
delay(1000);
digitalWrite(ROUGE, LOW); // V+B+R=Blanc
delay(1000);
digitalWrite(ROUGE, HIGH);
digitalWrite(VERT, HIGH);
digitalWrite(BLEU, HIGH);
}
```

Pour voir la vidéo correspondante :



Cas d'une LED RGB à cathode commune :



La programmation est la suivante :

```
int ROUGE=10;
int VERT=9;
int BLEU=8;

void setup()
{
  pinMode(ROUGE, OUTPUT);
  pinMode(VERT, OUTPUT);
  pinMode(BLEU, OUTPUT);
}

void loop()
{
  digitalWrite(ROUGE, HIGH); // Rouge
  delay(1000);
  digitalWrite(ROUGE, LOW);
  digitalWrite(VERT, HIGH); // Vert
  delay(1000);
  digitalWrite(VERT, LOW);
  digitalWrite(BLEU, HIGH); // Bleu
  delay(1000);
  digitalWrite(ROUGE, HIGH); // B+R=Magenta
  delay(1000);
  digitalWrite(BLEU, LOW);
  digitalWrite(VERT, HIGH); // R+V=Jaune
  delay(1000);
  digitalWrite(ROUGE, LOW);
  digitalWrite(BLEU, HIGH); // V+B=Cyan
  delay(1000);
  digitalWrite(ROUGE, HIGH); // V+B+R=Blanc
  delay(1000);
  digitalWrite(ROUGE, LOW);
  digitalWrite(VERT, LOW);
  digitalWrite(BLEU, LOW);
}
```

II – Utilisation du module ultrasons (module signaux – première ou terminale bac pro)

1) Mesurer une distance

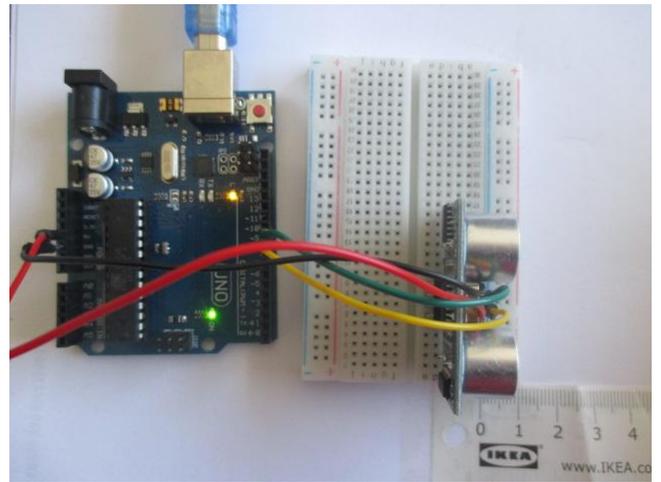
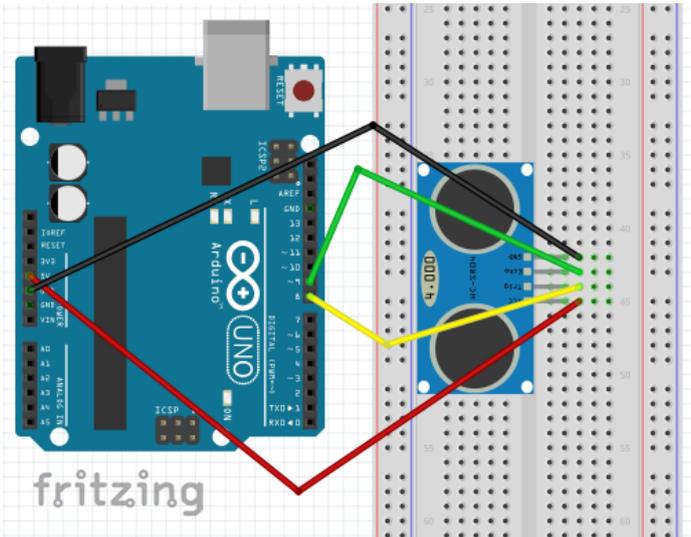
On place le module ultrasons sur 4 lignes de la plaque de montage puis on relie :

- le Vcc du capteur au +5V de la carte Arduino (fil rouge),
- le Trig du capteur à la borne 8 de la carte (fil jaune),
- l'Echo du capteur à la borne 9 de la carte (fil vert),
- le Gnd du capteur au GND de la carte Arduino (fil noir).

A l'aide du bouton d'affichage du moniteur série , on récupèrera la mesure de la distance.

Voici le schéma de montage :

En photo :



La programmation est la suivante :

```
int trig = 8;
int echo = 9;
long temps;
long distance;

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

void loop()
{
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  temps = pulseIn(echo, HIGH);
  distance = temps*340*100/1000000/2;
  Serial.print("Distance en cm : ");
  Serial.println(distance);
  delay(5000);
}
```

Analyse du code :

- Les 3 instructions ***digitalWrite(trig, HIGH); delayMicroseconds(10); digitalWrite(trig, LOW);*** lancent une impulsion de 10 microsecondes.
- ***temps = pulseIn(echo, HIGH);*** mesure le temps en microsecondes entre l'envoi de l'ultrason et sa réception.
- Explication de l'instruction ***distance = temps*340*100/1000000/2;*** : Pour calculer la distance, on utilise la formule $d = t \times v$ avec $v = 340$ m/s (vitesse du son dans l'air). Il faut multiplier par 100 pour convertir les m en cm, diviser par 1 000 000 pour convertir les μ s en s et penser à diviser par 2 pour tenir compte de l'aller-retour.
- ***Serial.print*** affiche sans retour à la ligne, ***Serial.println*** affiche avec retour à la ligne.

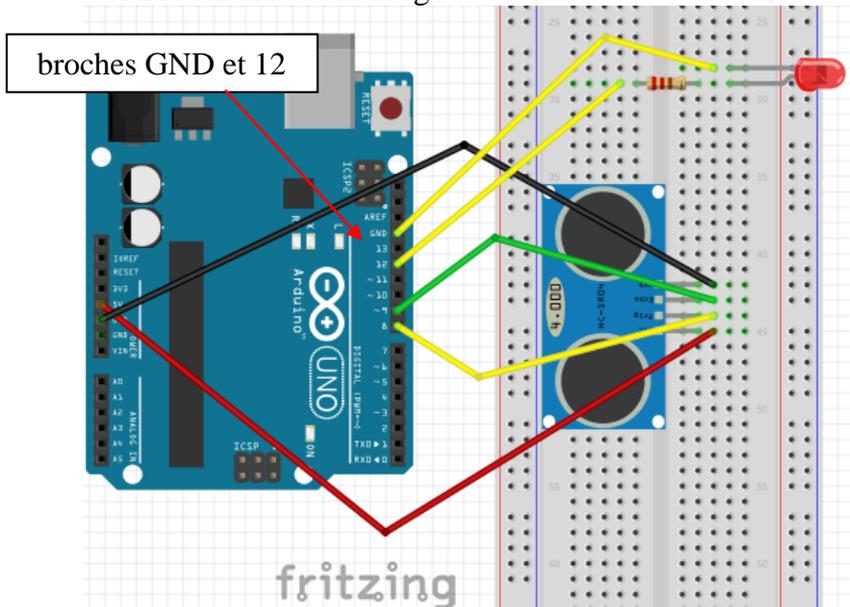
Pour voir la vidéo correspondante :



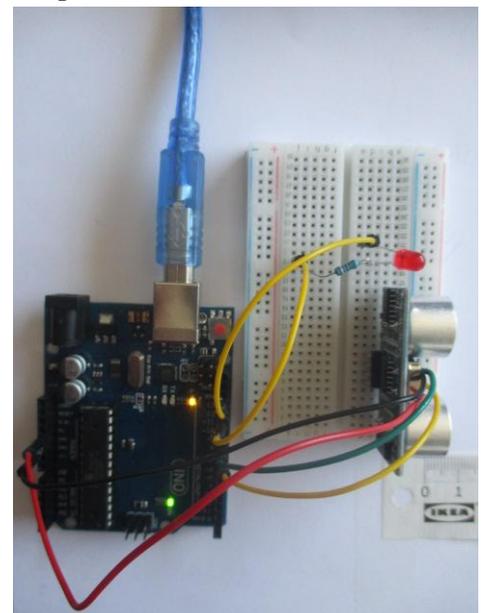
2) Créer un détecteur de présence ou d'obstacle

On souhaite qu'une LED s'allume si la distance à l'obstacle est inférieure à 10 cm.

Voici le schéma de montage :



En photo :



La programmation est la suivante :

```
int trig = 8;
int echo = 9;
int avertisseur=12;
int dmin=10; // distance (en cm) en dessous de laquelle la LED s'allume
long temps;
long distance;

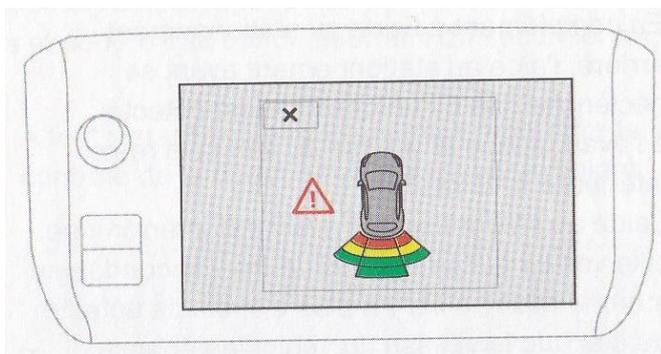
void setup()
{
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(avertisseur, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  temps = pulseIn(echo, HIGH);
  distance = temps*340*100/1000000/2;
  if (distance<dmin)
  {
    digitalWrite(avertisseur, HIGH);
  }
  else
  {
    digitalWrite(avertisseur, LOW);
  }
}
```

Pour voir la vidéo correspondante :



Prolongement possible :



(Source : guide d'utilisation Peugeot 2008)

On pourrait réaliser un radar de recul pour voiture avec les conditions suivantes :

- Une LED rouge s'allume si la distance à l'obstacle est inférieure à 30 cm,
- Une LED jaune s'allume si la distance à l'obstacle est comprise entre 30 et 60 cm,
- Une LED verte s'allume si la distance à l'obstacle est comprise entre 60 et 90 cm.

III – Utilisation d’une photorésistance ou du module photorésistance (module optique – seconde bac pro)



La photorésistance est une forme de capteur de lumière. Le principe est assez simple : plus il y a de lumière, plus la résistance est basse. L'obscurité provoque une résistance importante. Il s'agit donc d'un capteur de variation qu'il faudra connecter à l'Arduino avec un pin analogique.

On peut grâce à ce capteur, déclencher un événement en fonction de la luminosité : pièce qui s'éclaire lorsqu'on entre, inclinaison des stores pour gérer l'entrée de lumière, guidage d'un robot grâce à une source de lumière, allumage d'une lampe si trop d'obscurité...

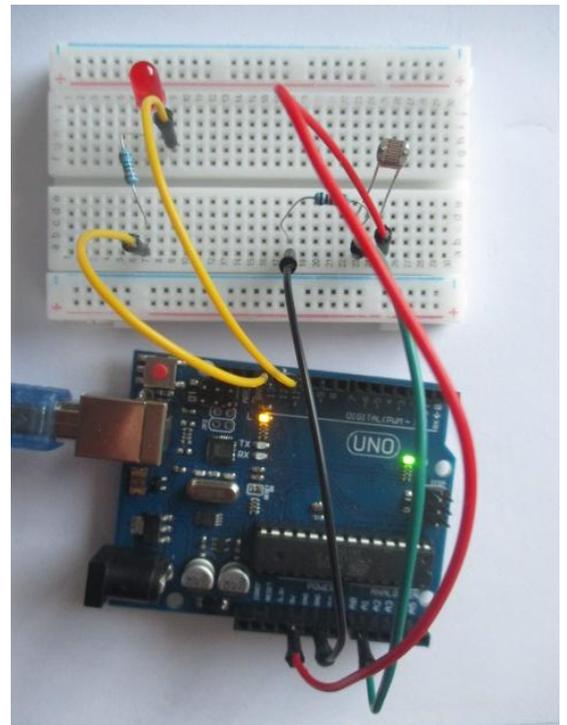
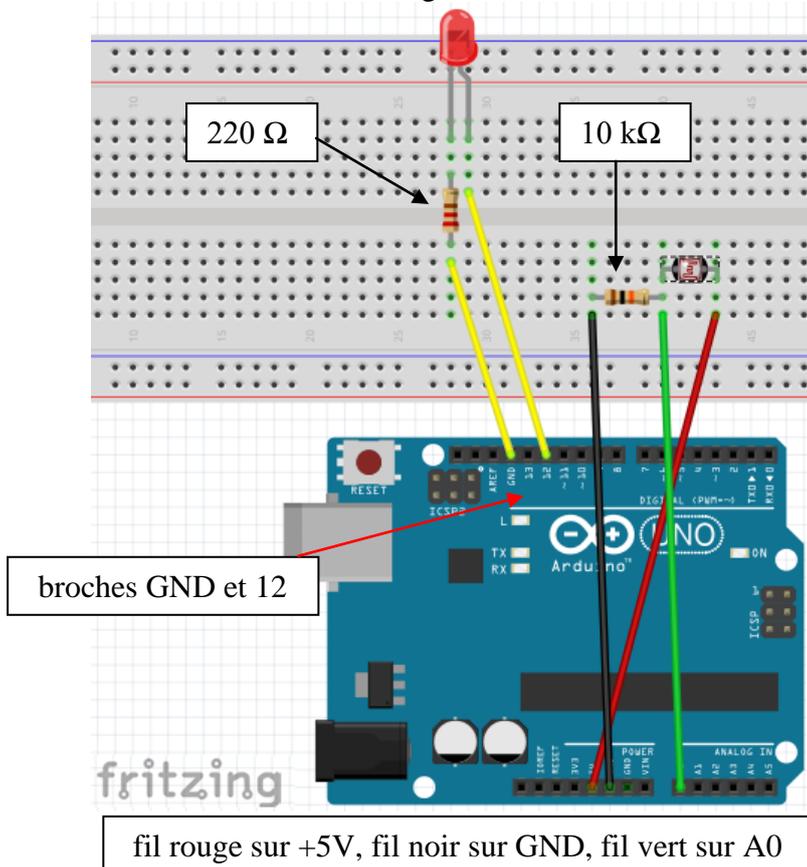
1) Faire clignoter une LED avec une vitesse proportionnelle à l'éclairage ambiant

On relie la LED au port 12 avec une résistance de $220\ \Omega$. On relie la photorésistance à l'entrée analogique A0. La résistance additionnelle doit être au moins équivalente à la plus forte valeur de la photorésistance : nous prendrons ici une résistance de $10\ k\Omega$.

Pour faire varier l'éclairage, on posera un doigt sur la photorésistance.

Voici le schéma de montage :

En photo :



La programmation est la suivante :

```
int led=12;
int luminosite;

void setup()
{
  pinMode(led, OUTPUT);
}

void loop()
{
  luminosite=analogRead(0);
  digitalWrite(led, HIGH);
  delay(luminosite);
  digitalWrite(led, LOW);
  delay(luminosite);
}
```

Pour voir la vidéo correspondante :



2) Réaliser un système d'éclairage automatique

On souhaite qu'une LED s'allume en dessous d'une valeur « seuil »

Le montage est le même.

A l'aide du bouton d'affichage du moniteur série , on déterminera la valeur du seuil. Avec la photorésistance utilisée, quand on pose un doigt sur la photorésistance, le moniteur série indique une valeur de 520 environ. A la lumière ambiante, elle est d'environ 930.

On propose donc une valeur de seuil de 600.

La programmation est la suivante :

```
int led=12;
int luminosite;
int seuil=600; // à modifier selon le résultat trouvé au moniteur série

void setup()
{
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
```

```

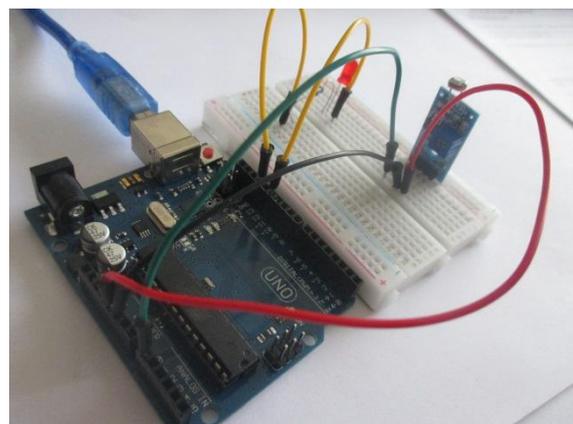
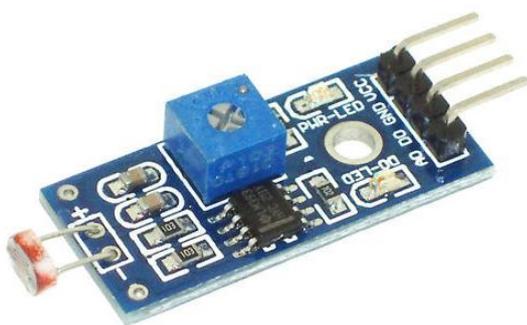
void loop()
{
  luminosite=analogRead(0);
  Serial.println(luminosite);
  /* permet de déterminer la valeur du seuil
   avec le bouton d'affichage du moniteur série */
  if(luminosite<seuil)
  {
    digitalWrite(led, HIGH);
  }
  else
  {
    digitalWrite(led, LOW);
  }
}
/* quand le seuil est déterminé, on peut effacer les 2 instructions
  Serial.begin(9600);
  Serial.println(luminosite); */

```

Pour voir la vidéo correspondante :



Autre possibilité : on peut utiliser le module photorésistance en le plaçant sur 4 lignes de la plaque de montage puis en reliant de la façon suivante :



On place le module photorésistance sur 4 lignes de la plaque de montage puis on relie :

- le A0 du capteur à l'entrée analogique A0 de la carte (fil vert),
- le D0 du capteur à rien,
- le Gnd du capteur au GND de la carte Arduino (fil noir),
- le VCC du capteur au +5V de la carte Arduino (fil rouge).

Le branchement de la LED et de sa résistance de 220 Ω est le même que précédemment (broches GND et 12).

A l'aide du bouton d'affichage du moniteur série , on déterminera la valeur du seuil.

Avec le module photorésistance utilisée, quand on pose un doigt sur la photorésistance, le moniteur série indique une valeur de 197 environ. A la lumière ambiante, elle est d'environ 65.

On propose donc une valeur de seuil de 150.

La programmation est la suivante :

```
int led=12;
int luminosite;
int seuil=150; // à modifier selon le résultat trouvé au moniteur série

void setup()
{
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  luminosite=analogRead(0);
  Serial.println(luminosite);
  /* permet de déterminer la valeur du seuil
   avec le bouton d'affichage du moniteur série */
  if(luminosite>seuil)
  {
    digitalWrite(led, HIGH);
  }
  else
  {
    digitalWrite(led, LOW);
  }
}
/* quand le seuil est déterminé, on peut effacer les 2 instructions
  Serial.begin(9600);
  Serial.println(luminosite); */
```

Pour voir la vidéo correspondante :

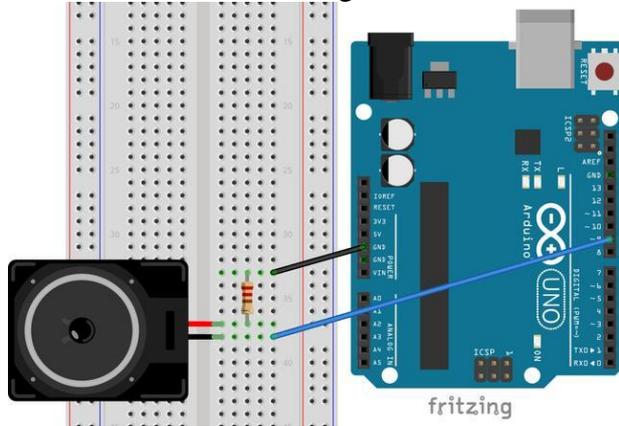


IV – Utilisation d'un haut-parleur : produire la note La₃ (module acoustique – seconde bac pro)

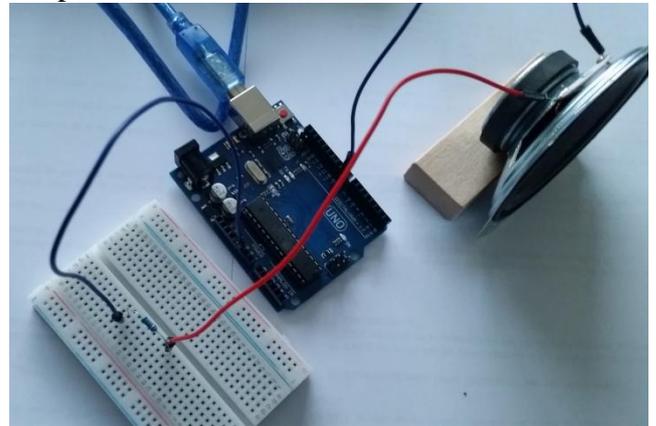
Nous allons produire la note La₃ (440 Hz) grâce à un haut-parleur 8 Ω 5 W, connecté sur la broche 9.

La résistance utilisée est de 220 Ω (résistance de limitation de courant).

Voici le schéma de montage :



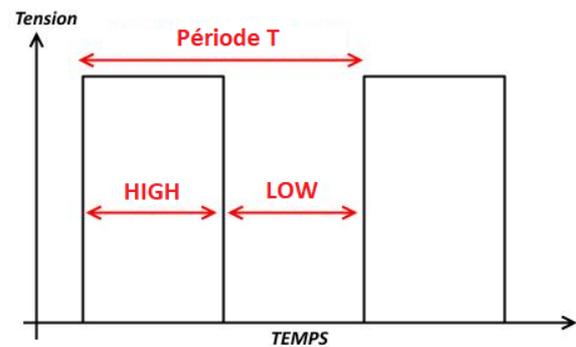
En photo :



La programmation est la suivante :

```
int hp=9;
void setup()
{
  pinMode(hp,OUTPUT);
}

void loop()
{
  digitalWrite(hp,HIGH);
  delayMicroseconds(1136);
  digitalWrite(hp,LOW);
  delayMicroseconds(1136);
}
```



$$T = 1/440 = 0,0022727272... \text{ s}$$

$$T = 2272,7272... \mu\text{s}$$

$$2272,7272 / 2 \approx 1136 \mu\text{s}$$

Pour voir la vidéo correspondante :



Autre alternative : la fonction **tone()**

Elle permet de générer un signal carré (50 % du temps à 5 V, 50 % du temps à 0 V) d'une fréquence et d'une durée donnée : *tone(broche,fréquence,durée);*

```
int hp=9;
void setup()
{
  pinMode(hp,OUTPUT);
}

void loop()
{
  tone(hp,440,400);
}
```

Prolongement : Au clair de la lune



Dans la gamme 3 : Do (261,63 Hz), Ré (293,66 Hz), Mi (329,63 Hz)

```
int hp=9;
void setup()
{
  pinMode(hp,OUTPUT);
}

void loop()
{
  tone(hp,262,400);
  delay(800);
  tone(hp,262,400);
  delay(800);
  tone(hp,262,400);
  delay(800);
  tone(hp,294,400);
  delay(800);
  tone(hp,330,800);
  delay(1000);
  tone(hp,294,800);
  delay(1000);
  tone(hp,262,400);
  delay(800);
  tone(hp,330,400);
  delay(800);
  tone(hp,294,400);
  delay(800);
  tone(hp,294,400);
  delay(800);
  tone(hp,262,1600);
  delay(2000);
}
```

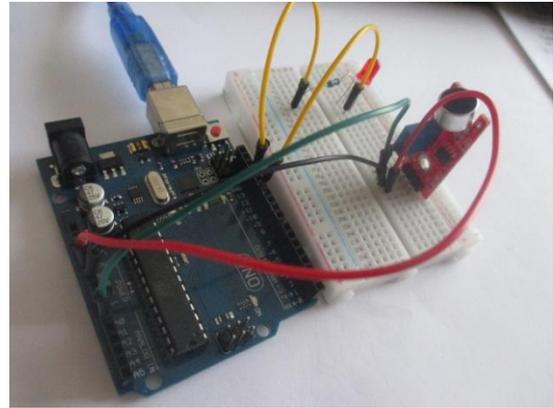
Pour voir la vidéo correspondante :



V – Utilisation du module capteur sonore (module acoustique – seconde bac pro)

On veut réaliser un montage qui allume une LED quand le niveau sonore est trop élevé. Le raisonnement est donc dans le même esprit que pour la photorésistance.

On utilise le module capteur sonore en le plaçant sur 4 lignes de la plaque de montage puis en reliant de la façon suivante :



On place le module photorésistance sur 4 lignes de la plaque de montage puis on relie :

- le A0 du capteur à l'entrée analogique A0 de la carte (fil vert),
- le G du capteur au GND de la carte Arduino (fil noir),
- le + du capteur au +5V de la carte Arduino (fil rouge),
- le D0 du capteur à rien.

Le branchement de la LED et de sa résistance de 220 Ω est le même que précédemment (broches GND et 12).

A l'aide du bouton d'affichage du moniteur série , on déterminera la valeur du seuil.

Avec le module capteur sonore utilisée, quand on approche du micro un signal sonore (chanson sur un Smartphone par exemple) assez élevé, le moniteur série indique une valeur de 302 environ. Sinon, elle est d'environ 292.

On propose donc une valeur de seuil de 300.

La programmation est la suivante :

```
int led=12;
int son;
int seuil=300; // à modifier selon le résultat trouvé au moniteur série

void setup()
{
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
```

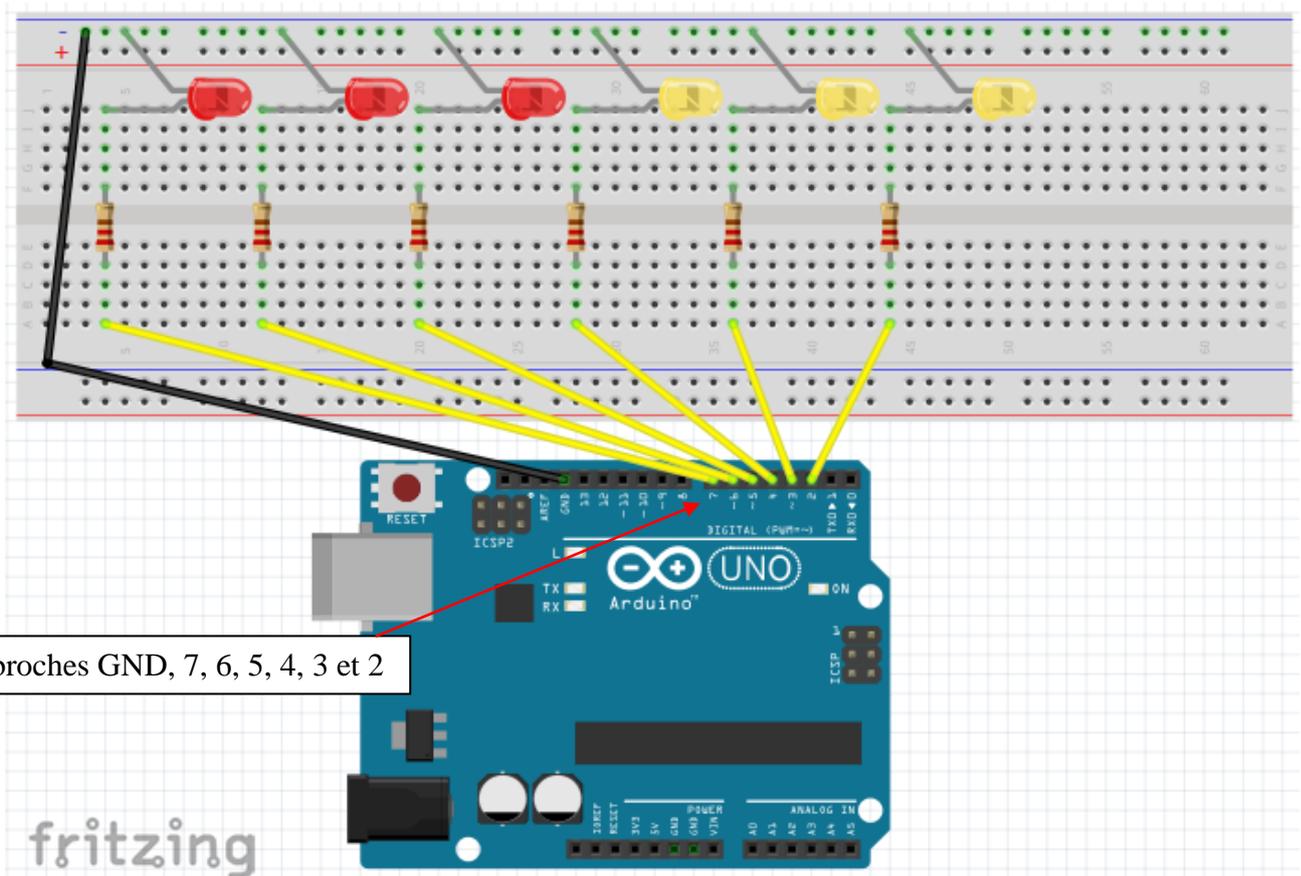
```
son=analogRead(0);
Serial.println(son);
/* permet de déterminer la valeur du seuil
avec le bouton d'affichage du moniteur série */
if(son>seuil)
{
digitalWrite(led, HIGH);
}
else
{
digitalWrite(led, LOW);
}
}
/* quand le seuil est déterminé, on peut effacer les 2 instructions
Serial.begin(9600);
Serial.println(son); */
```

Pour voir la vidéo correspondante :



ANNEXES

Annexe 1 : Simulation d'un dé à 6 faces



La programmation est la suivante :

```
int de;
int led;
int led1=2; // les 6 LED sont sur les broches 2 à 7
int led2=3;
int led3=4;
int led4=5;
int led5=6;
int led6=7;

void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0)); // sinon, le hasard sera toujours le même !
  for (led = 2; led < 8; led++)
  {
    pinMode(led, OUTPUT); // toutes les LED en output
    digitalWrite(led, LOW); // on les éteint toutes
  }
}

void loop()
{
  de=random(1,6); // un nombre au hasard entre 1 et 6
  Serial.println(de); // affiche le nombre sur le moniteur série
  affichage(de); // appel de la fonction d'allumage des LED
  delay(5000);
}
```

```
void setZero() //cette fonction sert à éteindre toutes les diodes
```

```
{  
  for (led=2;led<8;led++)  
  {  
    digitalWrite(led,LOW);  
  }  
}
```

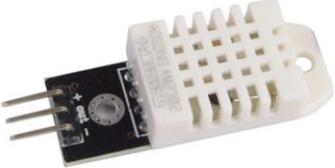
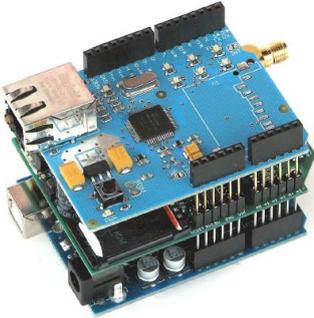
```
void affichage(int de) //cette fonction récupère un nombre et allume les LED en conséquence
```

```
{  
  setZero(); //appel de la fonction qui éteint toutes les LED  
  if (de==1)  
  {  
    digitalWrite(led1,HIGH);  
    return;//sortie de la fonction  
  }  
  if (de==2)  
  {  
    digitalWrite(led1,HIGH);  
    digitalWrite(led2,HIGH);  
    return;  
  }  
  if (de==3)  
  {  
    digitalWrite(led1,HIGH);  
    digitalWrite(led2,HIGH);  
    digitalWrite(led3,HIGH);  
    return;  
  }  
  if (de==4)  
  {  
    digitalWrite(led1,HIGH);  
    digitalWrite(led2,HIGH);  
    digitalWrite(led3,HIGH);  
    digitalWrite(led4,HIGH);  
    return;  
  }  
  if (de==5)  
  {  
    digitalWrite(led1,HIGH);  
    digitalWrite(led2,HIGH);  
    digitalWrite(led3,HIGH);  
    digitalWrite(led4,HIGH);  
    digitalWrite(led5,HIGH);  
    return;  
  }  
  if (de==6)  
  {  
    digitalWrite(led1,HIGH);  
    digitalWrite(led2,HIGH);  
    digitalWrite(led3,HIGH);  
    digitalWrite(led4,HIGH);  
    digitalWrite(led5,HIGH);  
    digitalWrite(led6,HIGH);  
    return;  
  }  
  delay(5000);  
}
```

Pour voir la vidéo
correspondante :



Annexe 2 : Matériels supplémentaires pour aller plus loin

 <p>Adaptateurs d'alimentation 9V</p>	<p>Pour alimenter la carte sans ordinateur.</p>
<p>Autres modules</p>	 <p>module capteur de température/d'humidité</p>  <p>Capteur de flamme infrarouge</p>
 <p>shield</p>	<p>Un shield est une petite carte qui se connecte sur une carte Arduino pour augmenter ses fonctionnalités.</p> <p>Exemples :</p> <ul style="list-style-type: none">- Ethernet- GSM- Wifi- Relais- LCD- GPS- etc.

WEBOGRAPHIE :

GENEVEY Frédéric et DULEX Jean-Pierre. *Arduino à l'école* [en ligne]. Disponible à l'adresse : http://arduino.education/wp-content/uploads/2018/08/Arduino_cours_sept2018.pdf

ROUSSEAU Jean-Noël. *Programmez vos premiers montages avec Arduino*. [en ligne]. Disponible à l'adresse : <https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino>

MONTAGNÉ Jean-Noël. *Atelier Arduino* [en ligne]. Disponible à l'adresse : <http://www.craslab.org/arduino/LivretArduinoFr06.pdf>

CASSEAU Christophe. *La physique computationnelle au lycée*. [en ligne]. Disponible à l'adresse : https://ent2d.ac-bordeaux.fr/disciplines/sciences-physiques/wp-content/uploads/sites/7/2018/10/physique_computationnelle.pdf