TP : Découverte Arduino

I. Introduction

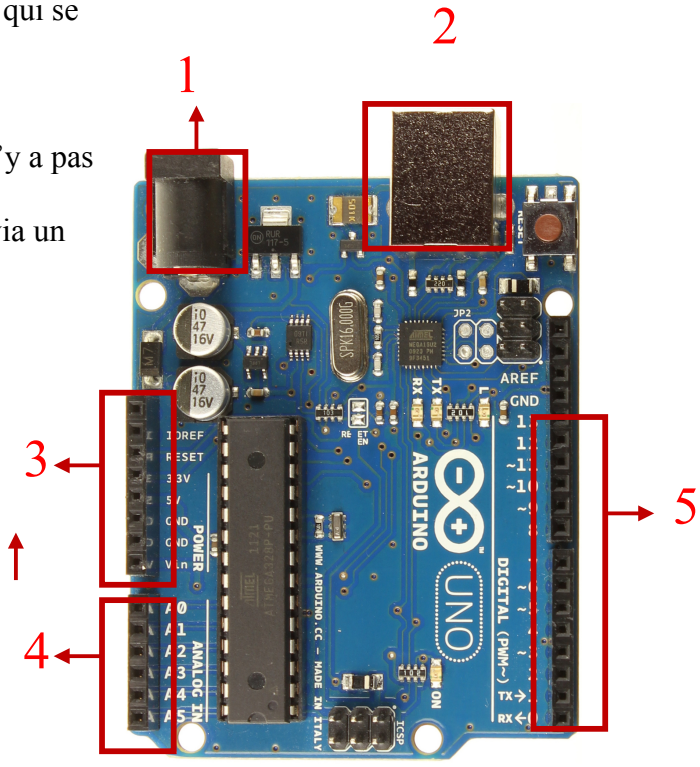
Les cartes arduino sont des microcontrôleurs qui vont vous permettre de réaliser des maquettes lors de vos projets. L'avantage de ces cartes est que la programmation est relativement simple et que tout a été réalisé en Open Source, il y a donc de nombreux exemples et bibliothèques présents sur le web qui pourront vous aider lors de vos projets.

Exemples de sites qui peuvent vous aider :

http://www.craslab.org/interaction/files/LivretArduinoCRAS.pdf

https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino/installez-vos-outils-de-travail (fortement recommandé)

Il existe diverses cartes Arduino avec des performances différentes pour chacune, nous allons utiliser la carte Arduino Uno qui se présente de la manière suivante :



1 – Alimentation de la carte dans le cas où il n'y a pas d'alimentation via le port USB (entre 7 et 12V)

2 – Alimentation/Communication avec le PC via un câble USB

3 – Masses et alimentation pour le circuit électronique

4 – Entrées Analogiques

5 – Entrées/Sorties Logiques

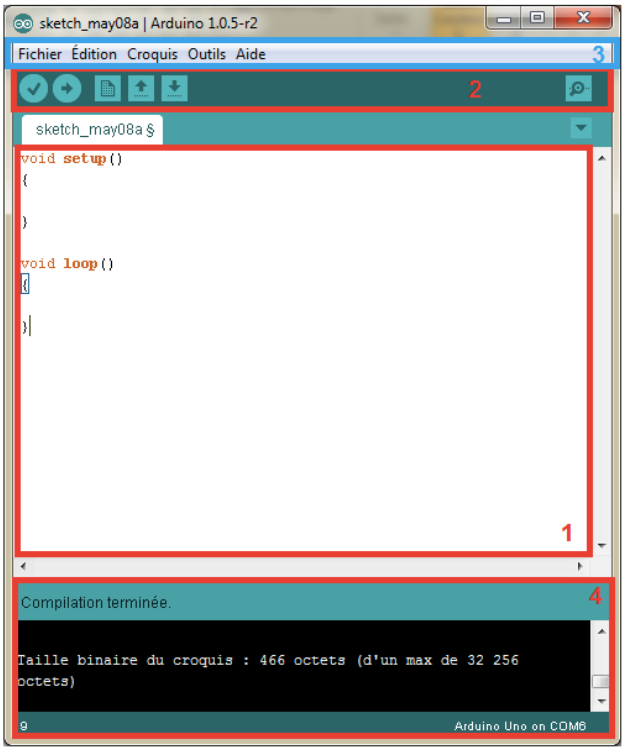
Les pins 0 et 1 sont utilisés uniquement pour la communication (via Blutooth par exemple)

Les pins 3,5,6,9,10 et 11 (symbole ~) peuvent également servir de sorties analogiques.

II. Présentation du logiciel

Le logiciel que nous allons utiliser est le logiciel gratuit distribué sur le site d'Arduino. Il se présente de la manière suivante :

1 – Editeur de code
2 – Barre d'outils rapide
3 – Barre de menu
4 – Console : affiche les résultats de la compilation et les éventuelles erreurs



Le langage Arduino est un langage informatique, il faut donc utiliser la syntaxe correcte pour que le programme puisse s'exécuter.

Lorsque vous ouvrez un nouveau programme Arduino, par défaut vous avez l'affichage ci-contre.

La fonction **setup** contiendra toutes les opérations nécessaires à la configuration de la carte (directions des entrées sorties…)

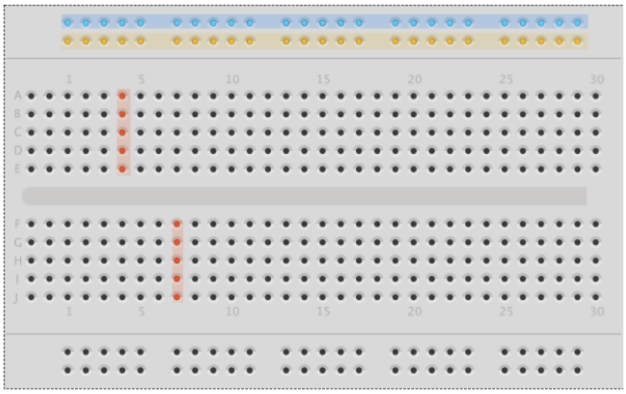
La fonction **loop** est exécutée en boucle après l'exécution de la fonction setup. Elle continuera de boucler tant que la carte n'est pas mise hors tension ou redémarrée (par le bouton reset). Cette boucle est absolument nécessaire sur les microcontrôleurs étant donné qu'ils n'ont pas de système d'exploitation. En effet, si l'on omettait cette boucle, à la fin du code produit, il sera impossible de reprendre la main sur la carte Arduino qui exécuterait alors du code aléatoire.

III. Câblage et exécution de programme

Lorsque vous branchez la carte Arduino sur l'ordinateur via le câble USB, vous devez préciser au programme sur quel port de communication vous travaillez. Pour cela, allez dans Outils/Ports et sélectionner un port de communication. **Vous ne pouvez pas travailler avec le COM1**, si le logiciel ne propose que celui-là, cela signifie qu'il y a un problème de branchement ou de carte.

1. Utilisation d'une breadboard

Pour tous les branchements que nous allons faire ensuite, nous allons utiliser une breadboard (ou protoboard).



Tous les trous surlignés de la même couleur sont connectés, cela signifie que si on connecte un composant dans un trou et un autre dans un trou lié, ils sont connectés entre eux.

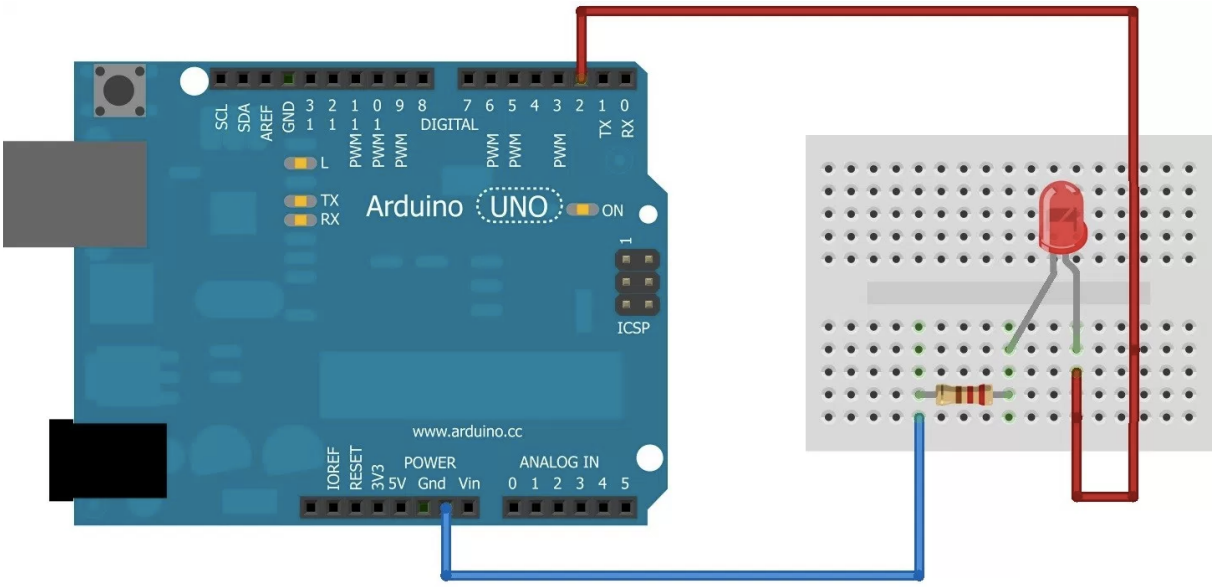
Usuellement les lignes bleu et jaune servent à connecter la masse et l'alimentation.

2. Branchement d'une LED

Il faudra connecter une résistance en série avec la Led afin que l'intensité délivrée par l'Arduino ne soit pas trop forte et qu'on ne grille pas la Led. Par contre, si l'on connecte une résistance trop grande, l'intensité lumineuse sera très faible.

Si on veut faire le calcul détaillé, il faut se reporter à la datasheet de la Led et chercher la chute de tension ainsi que l'intensité de fonctionnement. (Pour en savoir plus, c'est ici : https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino/jeux-de-lumiere-avec-une-led-et-la-breadboard)

Faites le branchement suivant : **(Réfléchissez au sens de la Led !)**



Maintenant écrivez le programme suivant et téléverser le dans la carte arduino (2ème bouton de la barre d'outils rapide)

```
int pinLed=2; //variable pour le numéro du pin utilisé
void setup()
{
    pinMode(pinLed,OUTPUT); //le pin 2 en mode sortie de courant
}
void loop()
{
    digitalWrite(pinLed,HIGH); // on passe le pin à +5V
    delay (1000); // attente de 1000 ms
    digitalWrite(pinLed,LOW); // on passe le pin à 0V
    delay(1000);
}
```

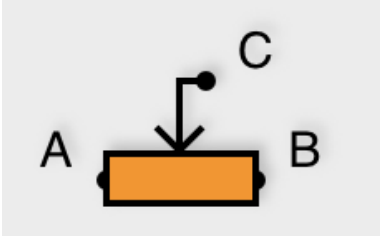
La Led va donc s'allumer lorsque l'on met le pin 2 à l'état haut donc à un potentiel de 5V.

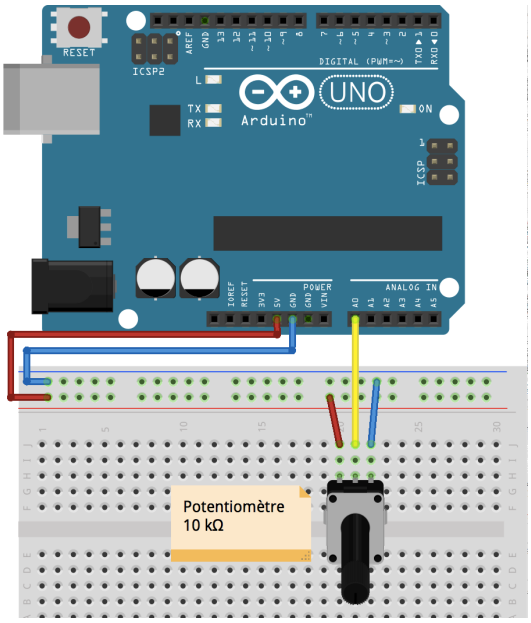
Remarque : On aurait pu brancher la petite patte de la Led sur le pin 2 et la grande patte sur le 5V, le comportement aurait alors été inversé !

Réalisez maintenant le câblage et le programme avec 3 Leds de couleur afin de modéliser le comportement d'un feu tricolore.

3. Branchement d'un potentiomètre

Un potentiomètre est une résistance variable, les pattes A et B sont respectivement branchées sur la masse et le potentiel haut (ou inversement) et la patte C doit être reliée à une entrée analogique de la carte Arduino, en effet le signal va prendre une infinité de valeurs.





Branchez le potentiomètre à la carte de la manière indiquée ci-contre.

Nous allons lire le signal analogique envoyé sur le pin A0, ce signal va être transformé en signal numérique. La carte Arduino code le signal reçu sur 10 bits, c'est à dire que le signal peut prendre 2^{10} valeurs, soit 1024 valeurs.

Tapez le code suivant, ouvrez le moniteur en allant dans Outils/Moniteur série et observez les valeurs obtenues.

```
int pinPot=A0; //variable pour définir le CAN où est connecté le potentiomètre
int valPot=0; //variable pour récupérer la tension aux bornes du potentiomètre traduite par le CAN . On l'initialise à 0.
void setup() {
  Serial.begin(9600); //Initialisation de la communication avec la console
}

void loop() {
  valPot=analogRead(pinPot); //lit la tension, la convertit en valeur numérique et la stocke dans valeurPot
  Serial.print("Valeur lue : ");
  Serial.println(valPot);
}
```

Proposez maintenant un branchement puis un programme qui permet d'allumer 1, 2 leds en fonction de la rotation du potentiomètre.

Pour cela, vous allez devoir utiliser une condition *if* dans votre programme Arduino. Elle s'écrit de la manière suivante :

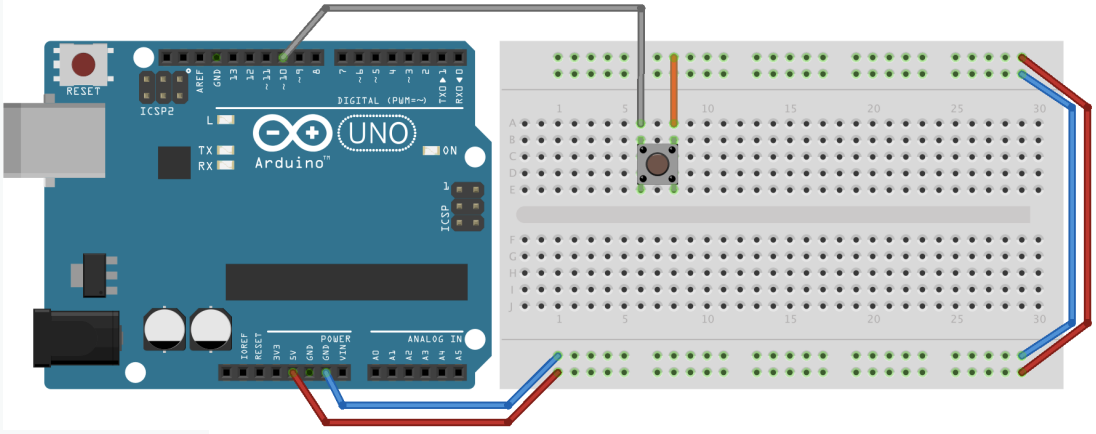
```
void loop()
{
    if (la condition qui vous intéresse)
    {
        les actions qu'il faut réaliser
    }
}
```

Une fois que vous avez réussi cela, compliquez votre programme pour allumer 0, 1, 2 ou 3 leds en fonction de la rotation du potentiomètre. Le if ne suffit plus, il faudra lui ajouter un else if.

```
if (condition1)
{
  // do Thing A
}
else if (condition2)
{
  // do Thing B
}
else
{
  // do Thing C
}
```

4. Branchement d'un bouton poussoir

On va commencer par le câblage le plus simple possible :



Et on va demander à l'Arduino de nous afficher des 1 ou des 0 en fonction du circuit, s'il est fermé ou ouvert.

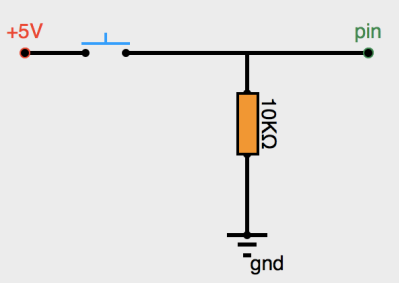
```
int pinBouton; // déclaration du bouton
void setup()
{
    Serial.begin(9600);
    pinBouton=10;        // déclaration du pin connecté au bouton
    pinMode(pinBouton,INPUT); // le pin déclaré est une entrée
}
void loop()
{
    boolean etatBouton=digitalRead(pinBouton); // Lecture de la valeur envoyée sur le pin
    Serial.println(etatBouton);                // Affichage
}
```

Testez ce programme… Il ne fonctionne pas correctement !

On dit que notre montage est erratique. Si on observe bien, le pin 10, quand le bouton est levé, n'est finalement connecté à rien. Le résultat lu par l'Arduino est donc peu interprétable. Il existe un moyen de forcer l'Arduino à lire quelque chose, tout simplement avec l'ajout d'une résistance...

Il faut savoir que l'électricité est paresseuse. Elle va toujours choisir le chemin qui lui résiste le moins. Mais si elle n'a pas le choix, elle passe tout de même là où ça résiste.

Nous allons donc ajouter une résistance à notre circuit. Une assez forte pour que le courant ne passe que s'il y est obligé (souvent de l'ordre de 10kΩ).



Quand le bouton poussoir est fermé, l'intensité va au pin. Quand le bouton poussoir est ouvert, l'intensité résiduelle présente dans le circuit va vers la masse.

Maintenant que vous avez un fonctionnement correct du bouton poussoir, créez un programme permettant de compter le nombre de fois où l'on presse le bouton.

5. To be continued…

Il existe énormément de capteurs et d'actionneurs pouvant être câblés sur des cartes Arduino (moteurs, accéléromètre, capteur infra rouge, …). En fonction de vos besoins, il faudra trouver les éléments adaptés et chercher la manière correcte de les utiliser.

Nous avons seulement vu la boucle *if* dans ce TP, nous pouvons aussi faire des boucles *for* ou *while*.