TP : Découverte de l'ARDUINO

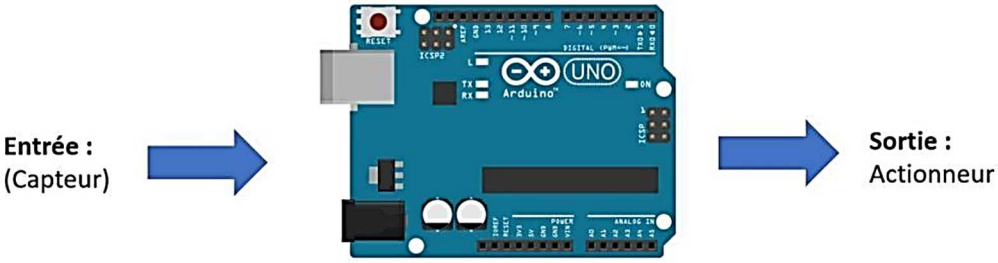
D'après *ArduinoFactory – Guide de démarrage*

L'Arduino est une **carte électronique** créée pour simplifier l'électronique à ceux qui veulent débuter ou se perfectionner dans l'électronique.

Elle permet de rendre accessible le monde de l'électronique et de créer des projets facilement, notamment la domotique, le pilotage de robot, les systèmes embarquées…

La carte Arduino possède des entrées et des sorties programmables qui permettent de contrôler beaucoup de composants : moteur à courant continu, servomoteur, photorésistance, télécommande, bouton poussoir, capteur de distance…

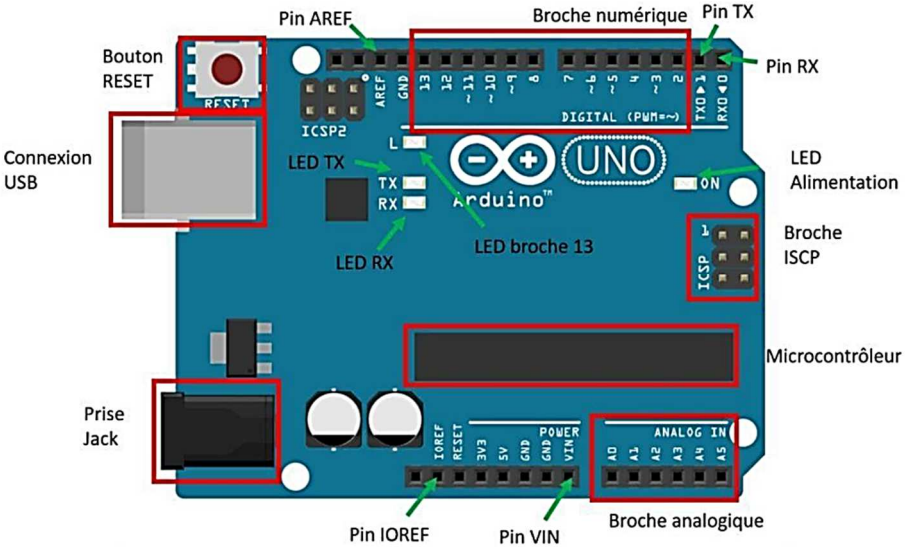
La carte Arduino est équipée d'un microcontrôleur. Celui-ci va permettre de traiter l'information entrante, comme la valeur d'un capteur, grâce à un programme et de commander des actionneurs (= sortie de la carte), on pourra ainsi réaliser un asservissement.



Comme tout système numérique, la carte Arduino fonctionne avec des grandeurs logiques 0 et des 1. Pour la carte Arduino :

- 0 ➔ état bas (LOW) : 0V
- 1 ➔ état haut (HIGH) : 5V

Voici plus en détail les composants situés sur une carte Arduino UNO :



Connexion USB :

- Permet d'alimenter la carte Arduino en 5V.
- Permet de téléverser le programme sur la carte Arduino et d'exporter des données.

Broche numérique :

- *Permet* de lire ou d'écrire sur la carte Arduino une tension soit de 0V ou de 5V, ce qui correspond pour la carte à 0 ou 1.
- Permet de contrôler des composants comme des servomoteurs (PWM : Pulse Width Modulation, une entrée pour commander un hacheur) ou même des capteurs nécessitan*t un signal numérique.*

Broche Analogique :

- Permet de lire ou d'envoyer une tension entre 0V et 5V. Un signal analogique est continu entre 0V et 5V.

Bouton RESET :

- Permet de redémarrer la carte et de réinitialiser le programme qu'elle contient.

Le logiciel : ARDUINO IDE

☐ A télécharger à l'adresse suivante : https://www.arduino.cc/en/software et à installer.
☐ Vous pouvez basculer l'interface en français : File/Preferences/Language.

Connecter la carte à l'ordinateur

☐ Brancher votre carte Arduino sur votre ordinateur à l'aide du câble USB.
☐ Lancer Arduino IDE.

A. Choisir le bon type de carte

Le logiciel Arduino IDE ne va pas téléverser le programme de la même manière si c'est une carte Arduino Uno ou Nano, car le microprocesseur à l'intérieur n'est pas le même.

☐ Dans la version actuelle d'Arduino IDE, en branchant votre carte Arduino celle-ci est directement reconnue par le logiciel. Si ce n'est pas le cas, vous pouvez la sélectionner (***Arduino Uno***) dans le menu *Outils* puis *Arduino AVR Boards.*

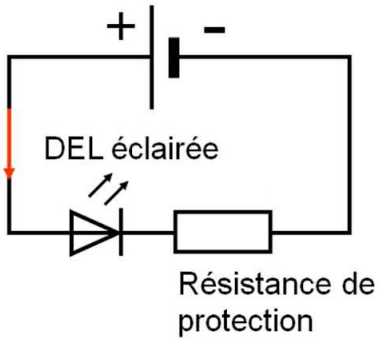
B. Choisir le Port COM

☐ Si votre carte Arduino n'est pas reconnue directement, vous devez choisir le Port COM sur lequel est branché la carte Arduino. Souvent, le logiciel Arduino IDE vous le propose directement.

Câbler sur plaquette d'essai

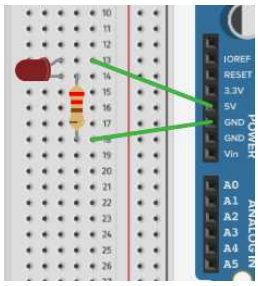
On souhaite réaliser le montage ci-contre afin de d'éclairer une LED. On utilisera la borne 5V pour la borne positive et une borne GND (ground = masse en anglais) pour la borne négative.

- La LED ne laisse passer le courant que dans un sens, celui qui correspond au triangle schématisant la LED. Sur le composant, le méplat correspond au trait vertical au bout de la flèche.
- On prendra une résistance de protection de 220 Ω (couleur rouge rouge noir noir marron).



En vous aidant de l'aide ci-dessous, câbler ce schéma où le + correspond à la broche « 5V » et où le – correspond à une broche « GND ».

☐ Vérifier que la LED s'éclaire. Si ce n'est pas le cas essyer d'inverser le sens de la LED.

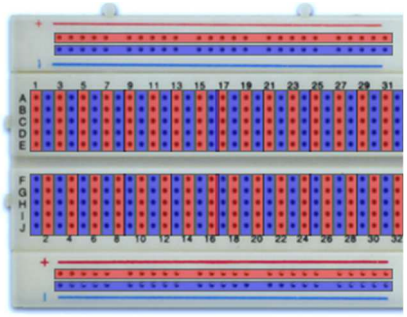


Outil : Plaque d'essai (*breadboard*)

Une plaque d'essai est d'une très grande utilité pour réaliser des montages électroniques sans soudure, elle s'utilise avec des straps, bouts de fils en cuivre.

Il existe des connexions internes sur la paque :

- Tous les points d'une même ligne du bus d'alimentation (en haut et en bas de la plaque) sont connectés entre eux.
- Tous les points d'une demi-colonne sont connectés entre eux.
- Les colonnes sont coupées en deux par le rail central qui permet de mettre des composants « à cheval ».



https://iea.org.lb/CMGuide/material.html

Langage Arduino

Le langage Arduino va vous permettre de programmer la carte Arduino UNO sur Arduino IDE.

A. Structure du programme

Il y a deux règles importantes pour commencer à coder en langage Arduino :

- Toutes les actions écrites doivent être terminées par un point-virgule afin que la carte Arduino comprenne que l'action est terminée.
- Toutes les fonctions commencent et se terminent par des accolades pour que la carte Arduino comprenne quand commence et termine la fonction.

Ce sont deux choses que l'on oublie souvent lorsqu'on écrit un programme et qui vont poser problème lors de la compilation de celui-ci.

B. Void setup()

Le *void setup* est une fonction que l'on écrit au début du programme, entre l'initialisation des variables et le *void loop*. Le *void setup* contient l'initialisation des composants comme entrée ou sortie de la carte Arduino, de l'initialisation du moniteur série que l'on va utiliser dans le reste du programme. Le *void setup* est une fonction qui va s'exécuter qu'une seule fois au début du programme.

C. Void loop()

Contrairement à la fonction *void setup* qui s'exécute une seule fois, la *void loop* s'exécute à l'infini. Ceci va permettre de contrôler vos composants sans jamais avoir à relancer le programme. La fonction *void loop* contient l'ensemble des fonctions pour lire les mesures de vos capteurs et les afficher sur le moniteur série. Vous pouvez aussi contrôler des composants, comme des LEDS, des servomoteurs…

Remarque : Les fonction *void loop* et *void setup* sont obligatoires dans tous vos programmes Arduino, même s'il n'y a rien écrit dedans, ne pas la mettre va créer une erreur.

Premier programme : faire clignoter une DEL

Nous allons maintenant faire clignoter la LED grâce à un script Arduino. Pour cela, nous aurons besoin de brancher notre LED sur une entrée/sortie de notre carte Arduino afin de pouvoir contrôler son allumage.

☐ Modifier le circuit précédent pour connecter la LED à la broche 11 à la place de la broche « 5V » de la carte Arduino.

☐ Télécharger le script « clignottements.ino » sur le cahier de prépa ou taper le code ci-dessous. Attention le fichier .ino doit être placé dans un dossier portant le même nom.

```
1   int led_Broche = 11;   // on assigne la variable led_Broche a 11
2
3   void setup() {
4     // put your setup code here, to run once:
5   pinMode(led_Broche,OUTPUT);   // on definit la broche de la led comme une sortie
6   }
7
8   void loop() {
9     // put your main code here, to run repeatedly:
10  digitalWrite(led_Broche,HIGH);   // on met la sortie led_Broche à 5V
11  delay(500);                      // on attend 500 ms
12  digitalWrite(led_Broche,LOW);    // on met la sortie led_Broche à 0V
13  delay(500);                      // on attend 500 ms
14  }
```

☐ Téléverser le programme sur la carte Arduino (aide-ci-après). Vérifier que la LED clignote.

Outil : Interface Arduino IDE

Le logiciel Arduino IDE possède plusieurs fonctionnalités qui vont vous être utile pour créer vos programmes. Voici les deux boutons les plus importants :

Bouton Vérifier

Le bouton *vérifier* permet au logiciel de savoir si votre programme a bien été écrit. Il va vérifier si vous avez bien téléchargé la librairie que vous souhaitez utiliser ou bien qu'il ne manque ni accolade, ni parenthèse, ni point-virgule à votre programme. Cela n'assure pas que votre programme va fonctionner comme vous le souhaitez.

Bouton Transférer

Le bouton *transférer* permet d'envoyer votre programme vers votre carte Arduino.

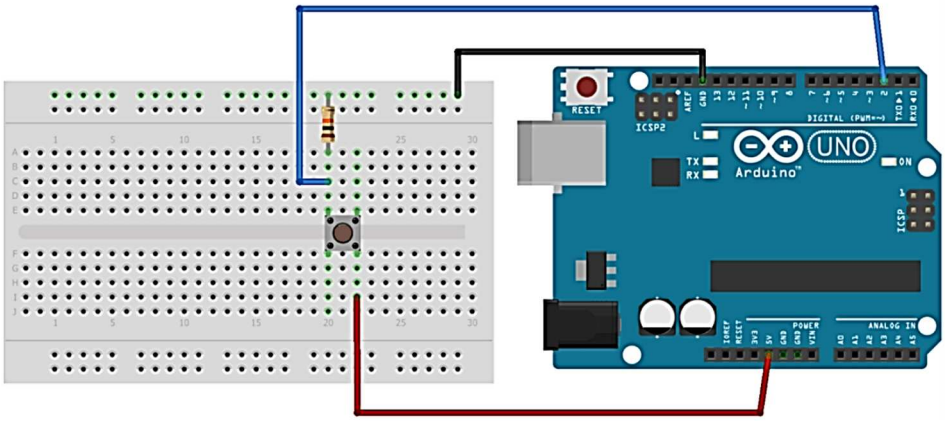
Le logiciel va d'abord vérifier votre programme comme avec la fonction vérifier, les problèmes de syntaxe seront affichés. Puis ce programme sera envoyé par un câble sur la carte Arduino.

☐ Rajouter une deuxième LED qui s'allume en alternance avec la première. Ne pas oublier de déclarer la deuxième broche utilisée.

Entrée digitale : bouton poussoir

Le bouton poussoir est un interrupteur électrique dont le bouton revient à sa position initiale après avoir été actionné. Dans un circuit sur Arduino vous pouvez l'utiliser pour avoir une interaction de l'utilisateur. Pour ce premier circuit on va voir comment lire les valeurs des boutons poussoirs depuis la carte Arduino.

☐ Réaliser le circuit suivant (R = 10 kΩ).



La résistance de 10 kΩ (couleur marron noir noir rouge marron) permet de bien avoir un signal à 0V lorsque l'on n'appuie pas sur le bouton poussoir.

☐ Télécharger le script « bouton_poussoir.ino » sur le cahier de prépa ou taper le code ci-dessous.

```
const int pin_INTERRUPTEUR = 2; // L'interrupteur est attaché au pin 2

void setup() {
    Serial.begin(9600);  // Initialisation de la communication avec le moniteur
série
    pinMode(pin_INTERRUPTEUR,INPUT);
}

void loop() {
    delay(1000); // Attente de 1000 ms
    boolean etatBouton = digitalRead(pin_INTERRUPTEUR); // Récupère l'état du
bouton
    Serial.println(etatBouton); // Affiche l'état du bouton sur le moniteur
}
```

- ☐ Téléverser le programme sur la carte Arduino.
- ☐ Ouvrir le moniteur série (aide ci-dessous) sur Arduino IDE avec la fréquence 9600 bd afin de lire les valeurs du bouton poussoir. Vérifier que la valeur renvoyée correspond bien à l'état du bouton poussoir.

Outil : Moniteur série

Le moniteur série est une interface entre l'utilisateur et la carte Arduino qui va recevoir des informations de la carte Arduino et les afficher sur Arduino IDE. Ces informations peuvent-être la température en degré donnée par un capteur ou bien l'angle d'un servomoteur.

Le moniteur série peut s'ouvrir seulement quand la carte Arduino est connectée et que le programme est téléversé dessus. Vous pouvez l'ouvrir avec le bouton en haut à droite du logiciel  ou dans l'onglet *Outils*.

Pour avoir des valeurs sur le moniteur série, vous devez le paramétrer dans votre code :

```
void setup() {
    Serial.begin(9600) ; // Initialisation de la communication avec le moniteur
série (9600 bits/s)
}
void loop() {
    Serial.print("Distance en cm :"); // Permet d'afficher sur le moniteur série
sur la même ligne
    Serial.println("la mesure"); // Permet de sauter une ligne après le texte
}
```

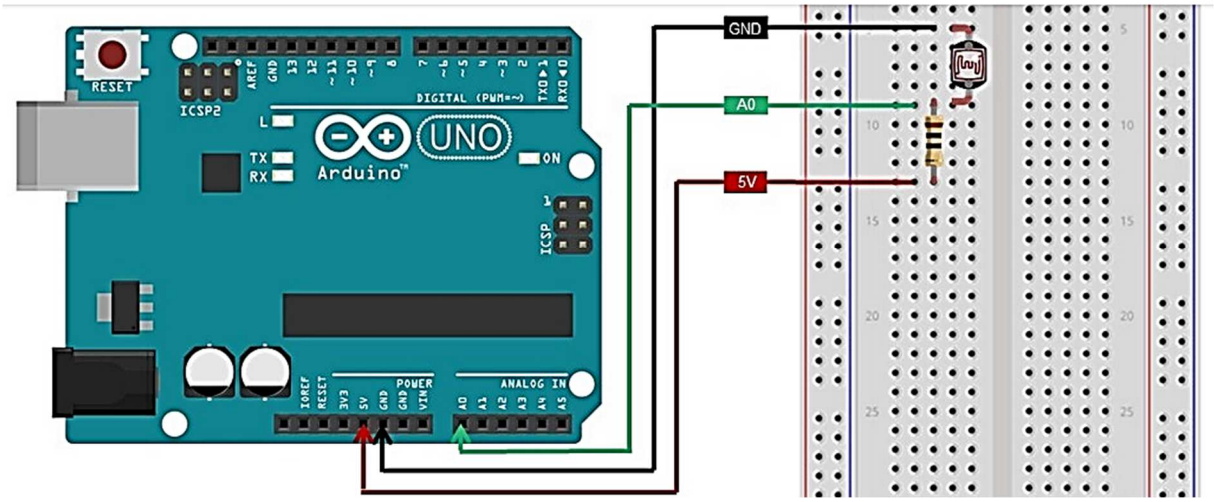
Le moniteur série doit être configuré avec un nombre précis de bauds (vitesse de transmission équivalente à des bit/s), ici 9600 bd. Ceci va être utile pour recevoir les informations depuis le moniteur série. Assurez-vous que débit du moniteur série correspond à la valeur que vous avez défini sur la carte Arduino. Vous pouvez laisser par défaut 9600 bd si votre projet ne nécessite pas de débit spécifique.

Entrée analogique : Le lampadaire

D'après https://fr.vittascience.com/learn/tutorial.php?id=8/mesurer-la-luminosite-avec-une-photoresistance-sur-arduino

Le but est maintenant d'allumer une LED lorsque la luminosité devient trop faible, comme un lampadaire, à l'aide d'une photorésistance comme capteur de luminosité. Une photorésistance est un composant dont la résistance change en fonction de la quantité de lumière à laquelle il est exposé.

☐ Brancher la photorésistance et la résistance selon le schéma suivant (R = 1 kΩ (couleur marron noir noir marron marron)) :



Outil : Entrée analogique

Les entrées analogiques traduisent la tension d'entrée lue sur le port utilisé en un nombre compris entre 0 et 1023 (codage sur 10 bits). La valeur 0 correspond à une tension de 0 V et la valeur 1023 correspond à une tension de 5 V.

☐ Télécharger le script « lampadaire_0.ino » sur le cahier de prépa ou taper le code ci-dessous.

```
void setup() {
  // put your setup code here, to run once:
pinMode(A0,INPUT);
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
int V = analogRead(A0);  // on lit la valeur sur l'entree analogique A0
Serial.println(V);
delay(1000);
}
```

☐ Téléverser le programme sur la carte Arduino.

☐ Ouvrir le moniteur série sur Arduino IDE avec la fréquence 9600 bd afin de lire les valeurs de V. Vérifier que la valeur renvoyée évolue bien en fonction de la luminosité.

☐ En vous inspirant de ce qui a été fait précédemment, rajouter une LED commander dans le circuit.

☐ Modifier le code afin de commander la LED à l'aide d'une condition de la forme ci-contre :

```
if (V > ...) {
  ...;
} else {
  ...;
}
```

☐ Téléverser le programme sur la carte Arduino et vérifier le bon fonctionnement du lampadaire.

Utilisation d'une bibliothèque : Mesure de température

D'après https://www.moussasoft.com/max6675-module-thermocouple-avec-arduino/

Certains capteurs ou actionneurs peuvent être difficile à gérer. Il existe alors des modules nous facilitant la tâche. Des bibliothèques permettent alors de récupérer facilement les informations ou de commander l'actionneur. Nous utilisons ici un capteur de température.

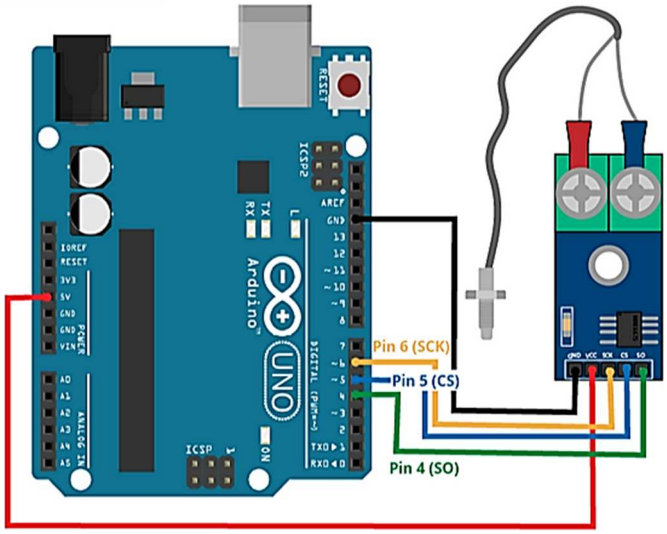
Le module MAX6675 est composé du circuit intégré MAX6675, qui mesure la température à l'aide d'un thermocouple de type K. Ce thermocouple est constitué de deux fils de matériaux différents qui génèrent une force électromotrice en fonction de la différence de température entre les deux extrémités. Le MAX6675 utilise une interface SPI pour communiquer avec le microcontrôleur et transférer les lectures de température en sortie.

Pour faciliter l'utilisation du module MAX6675, nous pouvons installer la bibliothèque MAX6675 dans l'IDE Arduino. La bibliothèque est fournie par Adafruit.

☐ Allez dans le menu "Croquis" > "Importer une bibliothèque" > "Gérer les bibliothèques".

☐ Recherchez "max6675 (par Adafruit)" et installez la dernière version disponible.

☐ Réaliser le montage suivant (en passant par la plaquette d'essai) :



☐ Télécharger le script « thermocouple.ino » sur le cahier de prépa ou taper le code ci-dessous.

```
#include "max6675.h"   // on utilise la bibliotheque associee au MAX6675

// on définit les différents ports utilisés pour communiquer avec le capteur
int SO = 4;
int CS = 5;
int sck = 6;
MAX6675 module(sck, CS, SO);

void setup() {
  Serial.begin(9600);    // on démarre la liaison série
}

void loop() {
  float temperature = module.readCelsius();  // la température est mesurée
  Serial.println(temperature);               // la valeur de la température est envoyée sur le port série
  delay(1000);
}
```

☐ Ouvrir le moniteur série sur Arduino IDE avec la fréquence 9600 bd afin de lire les valeurs de la température. Vérifier que la valeur renvoyée évolue bien.

Récupérer des données et les stocker

On souhaite maintenant stocker dans un fichier les informations envoyées par l'Arduino sur le port série. Pour cela, nous allons lire le moniteur (ou le traceur) série avec un script python. On pourra ensuite se servir de Python pour traiter ces données directement.

- ☐ Reprendre le script « lampadaire_0.ino » utilisé précédemment.
- ☐ Téléverser le programme sur la carte Arduino.
- ☐ Fermer Arduino IDE afin de libérer la liaison série et de permettre sa lecture via Python.
- ☐ Ouvrir Spyder et ouvrir le script Python « recup_donnees.py » disponible sur le cahier de prépa.
- ☐ Charger le package python pyserial dans votre IDE afin d'installer la bibliothèque pyserial nécessaire pour récupérer les données de la liaison série (à faire une seule fois). Pour ceux qui utilisent pip, dans la **console** Python, taper et exécuter la commande « pip install pyserial ».
- ☐ Choisir le nombre de mesures que vous souhaitez récupérer (de l'ordre de la dizaine pour commencer).
- ☐ Exécuter le script Python et constater que les données ont bien été récupérées.
- ☐ Reprendre la procédure mais récupérer cette fois la valeur de la tension en volt (modifier le script Arduino).

Les données sont alors stockées dans un fichier .csv dans le même dossier que le script python. Ce fichier peut ensuite être exploité avec python, Excel ou Regressi par exemple.

Remarque : Si on a besoin de récupérer deux données simultanément, on pourra les afficher successivement sur le port série et utiliser le script Python « recup_donnees_multi.py ».

A vous de jouer :

- ☐ Tracer avec Python le suivi de la température pendant une minute toutes les 2 secondes.