

Manipulation d'Arduino N°1 :

1- Définition :

Arduino est un **circuit imprimé** en **matériel libre** sur lequel se trouve un **microcontrôleur** qui peut être programmé pour analyser et produire des **signaux électriques**. C'est une carte électronique programmable, avec un certain nombre d'entrées/sorties

Circuit imprimé : C'est une sorte de plaque sur laquelle sont soudés plusieurs composants électroniques reliés entre eux par un circuit électrique plus ou moins compliqué. L'Arduino est donc un circuit imprimé **Fig. 1**.

Microcontrôleur : C'est le cœur de la carte Arduino. C'est une sorte d'ordinateur minuscule (mémoire morte, mémoire vive, processeur et entrées/sorties) et c'est lui que nous allons programmer. Sur la **Fig. 1** c'est le grand truc rectangulaire noir avec plein de pattes. Une fois lancé et alimenté en énergie, il est autonome. La force de l'Arduino est de nous proposer le microcontrôleur, les entrées/sorties, la connectique et l'alimentation sur une seule carte. La carte Arduino est construite autour d'un microcontrôleur Atmel AVR avec une capacité de mémoire de 32000 octets pour l'Arduino UNO. Soit 32 Ko, ce qui n'est vraiment pas beaucoup et qui permet pourtant de réaliser un max de projets. Une petite carte comme celle-ci peut gérer des moteurs, des systèmes d'affichage, des capteurs (accéléromètres, température, pression atmosphérique, luminosité, et la liste est loin d'être exhaustive).

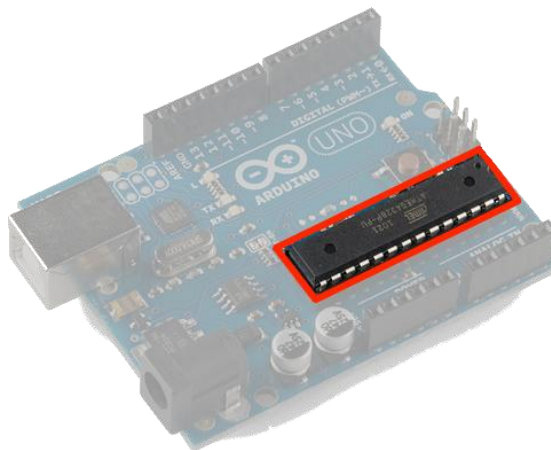


Fig 1 : Carte Arduino

Signaux électriques : L'Arduino fonctionne avec de l'électricité. Un signal électrique est un passage d'électricité dans une partie du circuit. L'Arduino est donc capable de produire ou de capter ces signaux à notre demande grâce à la programmation.

L'Arduino est donc une carte qui se connecte sur l'ordinateur pour être programmée, et qui peut ensuite fonctionner seule si elle est alimentée en énergie. Elle permet de recevoir des informations et d'en transmettre depuis ou vers des matériels électroniques : diodes, potentiomètres, récepteurs, servo-moteurs, moteurs, détecteurs...

Donc Arduino est une carte électronique open source, constituée essentiellement de :

- Un microcontrôleur fabriqué par Atmel,
- Un port USB,
- Des connecteurs d'entrées/sortie (plus ou moins nombreux selon les modèles).

Voici schéma **Fig.2** qui résume les principales interactions en jeu lorsque l'on programme une carte Arduino pour contrôler du matériel. Les flèches **vertes** indiquent la circulation des signaux électriques, la flèche **orange pointillée** représente l'envoi du programme vers l'Arduino et les flèches **bleues** les interactions avec le monde réel.

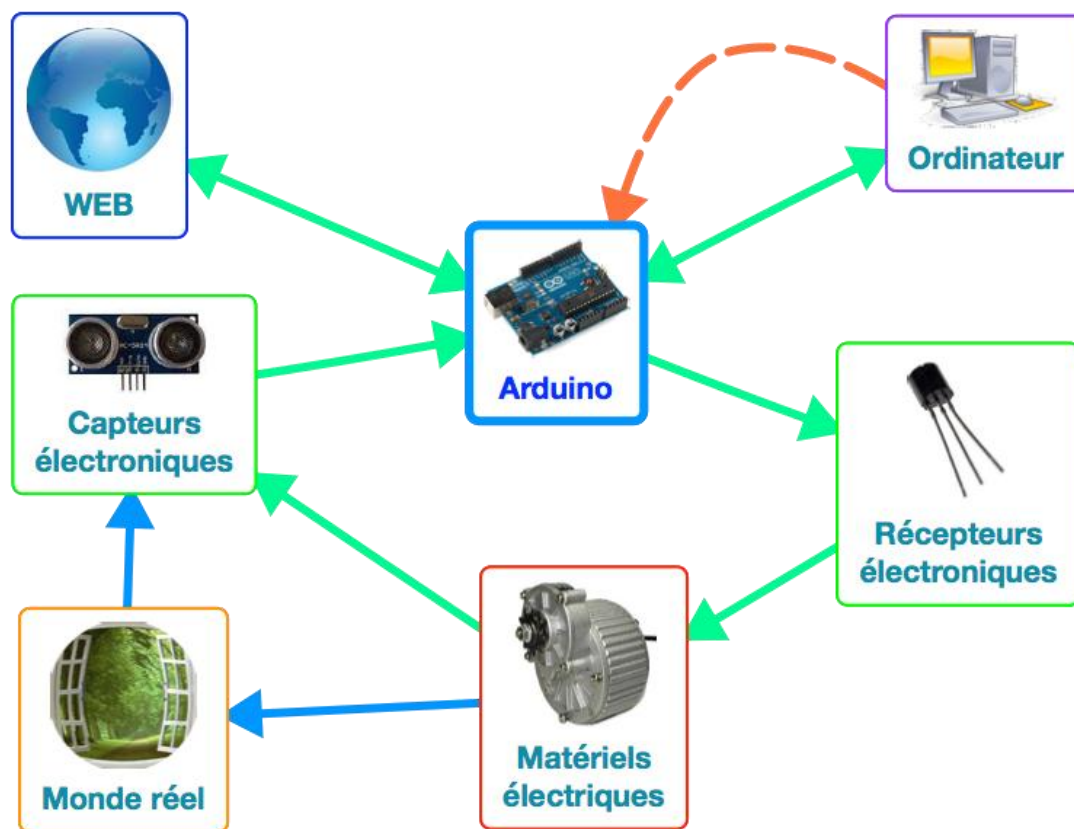


Fig 2 les principales interactions de la carte Arduino

2- Principe de fonctionnement

Le principe est d'utiliser les pins (i.e. les connecteurs numérotés que l'on peut voir sur la photo) pour lire ou écrire des informations ; les informations sont de très bas niveau, vu que c'est un signal soit de 5 V, soit de 0 V.

3- Logiciel :

Arduino, c'est aussi un IDE (Un éditeur de code) qui permet d'envoyer les programmes sur la carte à travers un port USB.

Pour manipuler l'Arduino veuillez télécharger le logiciel Arduino sur le site officiel ; Il est open source : <https://www.arduino.cc/>

(Installation version Windows ou Linux)

4- Créer votre premier programme :

4.1 La structure de base d'un programme Arduino :

- ✚ Le langage utilisé par le logiciel Arduino pour programmer le microcontrôleur est basé sur les langages C/C++.
- ✚ Le programme principal consiste en deux fonctions.

```
1 void setup()
2 {
3
4 }
5
6 void loop()
7 {
8
9 }
```

`setup()` est appelée une seule fois, au moment de la mise sous tension de la carte.

`loop()` est appelée, comme son nom l'indique, en boucle. Elle est lancée après `setup()`, et tourne à fond à l'infini (tant que la carte est alimentée).



Le premier bouton à gauche : Le V de vérifier.

Ce bouton permet de vérifier votre programme. C'est-à-dire que le logiciel Arduino va chercher si ce que vous avez écrit est conforme à ce qui est attendu.



Le La flèche de téléverser :

Une fois connectés, vous pouvez cliquer sur l'icône avec la flèche pour téléverser le programme. Le logiciel va donc transférer votre programme compilé (transformé en langage machine) dans la mémoire du microcontrôleur de l'Arduino. Une fois cette opération effectuée, l'Arduino gardera ce programme en mémoire et l'exécutera tant qu'il sera alimenté en électricité. Il sera donc autonome et ne dépendra plus de l'ordinateur.

4.2 La LED 13

Exemple : Faire allumer La LED 13 de l'Arduino.

Sur la **Fig. 3** Il y a 3 petits rectangles notés L, TX et RX, et un autre à droite noté ON.

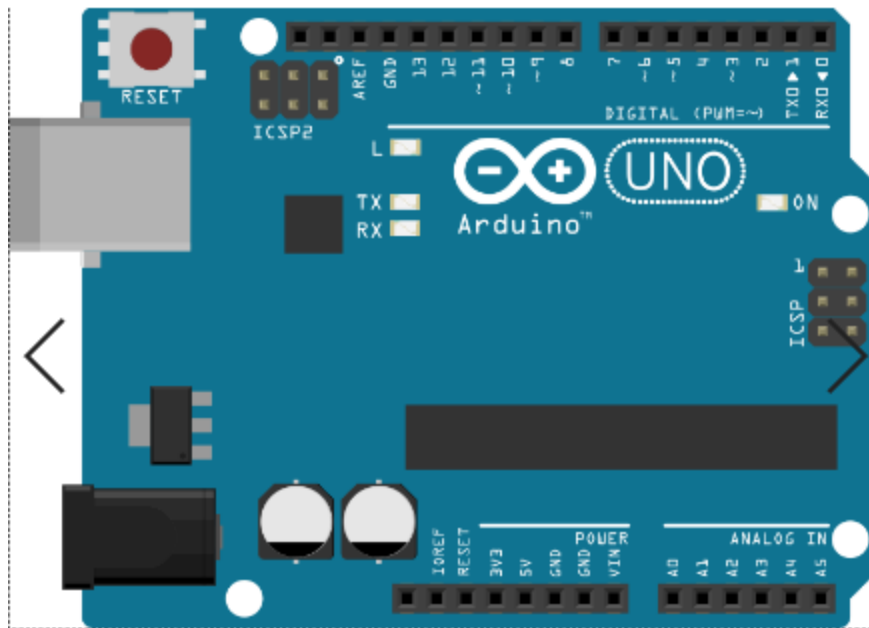


Fig 3. LED de la carte ARDUINO

Ces rectangles sont des LED (**L**ight-**E**mitting **D**iodes). Ces 4 LEDs peuvent briller dans plusieurs cas :

- La LED **ON** est **verte** quand l'Arduino est sous tension.
- Les LEDs **TX** et **RX** **clignotent** quand l'Arduino reçoit ou envoie des informations.
- La LED **L** **clignote** si on appuie sur le bouton reset, sinon elle ne fait rien ;
- C'est la LED **L** que nous allons allumer grâce à notre petit programme.

Une diode s'allume quand elle est parcourue par le bon courant, dans le bon sens.

La force de l'Arduino est d'envoyer du courant, ou non, par les connexions numérotées de 0 à 13. Et la diode **L** s'allume quand on dit à l'Arduino d'envoyer du courant dans la connexion **13**.

Les étapes du programme pour allumer la LED 13 de l'Arduino sont assez simples :

1. On dit à l'Arduino que nous voulons que la connexion 13 puisse envoyer du courant (et pas en recevoir).
2. On dit à l'Arduino d'envoyer du courant dans la connexion 13.

Voici donc le programme à taper et à envoyer à l'Arduino :

```

1 void setup()
2 {
3   pinMode(13,OUTPUT);
4   digitalWrite(13,HIGH);
5 }
6
7 void loop()
8 {
9
10 }

```

- ✓ Uploader le code.
- ✓ Eteignez la LED en utilisant la commande digitalWrite (13, LOW)

Exercices :

Exercice :1

Programme « Blink ! » :

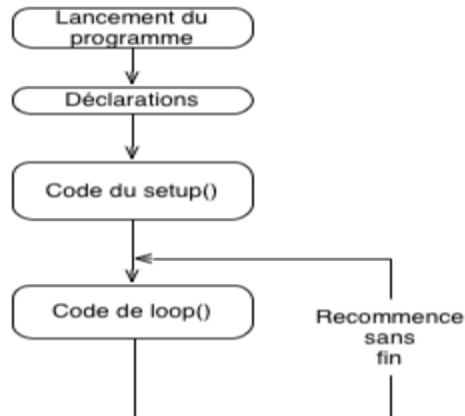
Nous voulons faire clignoter la LED 13 donc les étapes seraient :

1. Nous indiquons à l'Arduino que la connexion 13 doit pouvoir envoyer du courant
2. Nous indiquons d'envoyer du courant dans la connexion 13 (la diode s'allume)
3. Nous indiquons d'arrêter d'envoyer du courant dans la connexion 13 (la diode s'éteint)
4. Nous voulons ensuite recommencer au point numéro 2 à l'infini.

Note :

Les instructions mises dans les accolades de "loop" seront lues et activées **à l'infini** !

Ce qui donne en représentation graphique :



Code :

```

int led = 13;

void setup() {
  pinMode(led, OUTPUT);    // Définit la pin 13 comme pin de sortie
}

// Boucle qui s'exécute à l'infini :
void loop() {
  digitalWrite(led, HIGH);  // Allume la LED
  delay(1000);              // Attend une seconde
  digitalWrite(led, LOW);   // Eteint la LED
  delay(1000);              // Attend une seconde
}

```

- *setup()* qui s'exécute qu'une fois ;
- *loop()* qui s'exécute à l'infini ;
- *pinMode()* qui permet de définir une sortie en mode envoi d'électricité ou non ;
- *digitalWrite()* qui envoie de l'électricité par une sortie ;
- *delay()* qui met le programme en pause pendant un nombre défini de millisecondes.

Moniteur série :

L'Arduino de base n'a pas d'écran. C'est là qu'entre en scène le **moniteur série**, qui est une fenêtre que l'on peut ouvrir en cliquant sur la loupe du logiciel Arduino :



Cette fenêtre reçoit et envoie des infos sous forme de séries de bytes aux ports concernés. Donc grâce au moniteur série, l'Arduino peut (à condition d'être connecté à un PC) envoyer des informations à l'ordinateur qui va pouvoir les afficher en temps réel.

Pour ce faire on fait appel à une bibliothèque spéciale, déjà incluse dans l'IDE : la bibliothèque **Serial**.

Pour utiliser la possibilité de communication entre l'ordinateur et l'Arduino, on procède en deux étapes :

1. On initialise la communication (ça se fait dans le **setup()**).
2. On communique (ça se fait souvent dans la **loop()** mais possible dans le **setup()**).

```
setup()
{
  Serial.begin(9600) ;// initialisation de la communication
  Serial.println("Communication initialisée") ;// envoi d'un message
}
void loop()
{
  Serial.println("Je suis dans la boucle !") ;//envoi d'un autre message
}
```

Remarque :

- Si vous uploader ce programme rien ne se passe, sauf que la diode TX de votre Arduino est allumée.
- La LED TX montre une transmission envoyée par l'Arduino, la LED RX montre une transmission reçue par l'Arduino.

Modifier ce programme pour que le message dans la boucle ne soit envoyé que toutes les 2 secondes en utilisant la fonction **delay(2000)** après le message ;

delay() : est une fonction qui permet d'arrêter l'exécution des instructions qui précèdent sa déclaration pendant une durée bien déterminée.

Vous remarquerez que la diode TX ne clignote qu'à l'envoi du message.

Exercice 2 :

Programmer un message qui s'affiche sur trois lignes "Hello Arduino World !" avec un mot par ligne.

```
void setup()
{
  //initialisation de la communication
```

```

    Serial.begin(9600);
    //affichage du texte
    Serial.println("Hello");
    Serial.println("Arduino");
    Serial.println("World !");
}
void loop()
{

}

```

Exercice 3 :

L'objectif de ce programme est de faire compter l'Arduino tout seul :

- Tout d'abord, l'Arduino doit afficher sur le moniteur série le nombre 1 et allumer une fois la LED 13.
- Puis, après un temps d'arrêt d'une seconde, il doit afficher le nombre 2 et faire clignoter 2 fois la LED 13.
- Encore un arrêt d'une seconde, puis affichage du chiffre 3 avec 3 clignotements
- L'opération se répète jusqu'au nombre 20, puis l'Arduino envoie "Fin » sur le moniteur série.

Voici le code :

```

boolean affichage; //variable pour stopper l'affichage
int numPin;

void setup()
{
    numPin=13;
    pinMode(numPin,OUTPUT);
    Serial.begin(9600);
    affichage=true; //initialisation de la variable à true
    Serial.println("*** Debut du programme ***");
}

void loop()
{
    if (affichage) // test si affichage vaut true

```

```

{
    //boucle de comptage
    //compteur s'augmente de 1 à chaque tour
    for (int compteur=1;compteur<=20;compteur=compteur+1)
    {
        Serial.println(compteur);

        //boucle de clignotement
        //compteur sert de limite à la boucle
        //donc le nombre de clignotements augmente à chaque tour
        for (int nbClignote=0;nbClignote<compteur;nbClignote=nbClignote+1)
        {
            //allume
            digitalWrite(numPin,HIGH);
            delay(250);
            //eteind
            digitalWrite(numPin,LOW);
            delay(250);
        }
        delay(1000); //attente de 1s
    }
    affichage=false; // on passe affichage à false
    Serial.println("*** Fin ! ***");
}
}

```

Exercice 4 : Table de multiplication

- Afficher la table de multiplication par 7 des nombres de 0 à 14.

Voici quelques règles à respecter :

- ✓ Le programme devra afficher la table de multiplication une seule fois, il faudra donc utiliser une condition pour vérifier si elle a été affichée.
- ✓ Pour rendre le programme facile à modifier, vous stockerez le nombre 7 dans une variable de type *int*.
- ✓ Le moniteur devra afficher une table qui ressemble à **Fig. 4** :

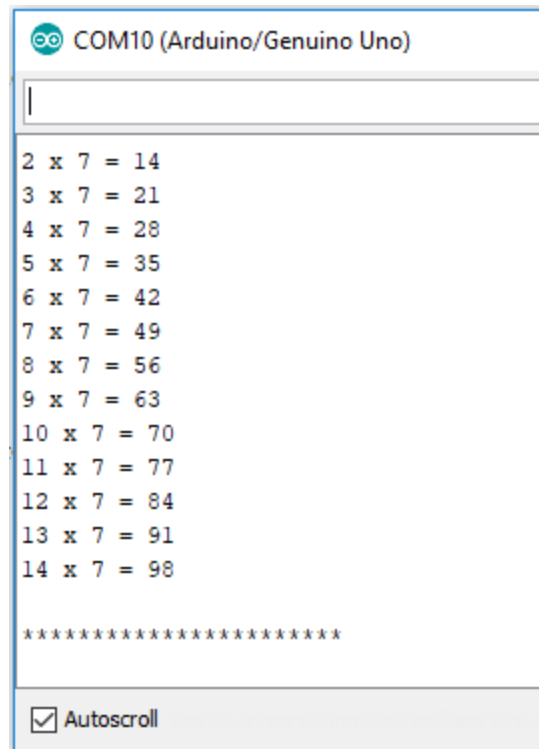


Fig. 4 : Table de multiplication

- Ecrire les instructions d’affichage de la table de multiplication sous forme de fonction. :

Voici le code :

```
int numTable; // variable pour la table concernée
boolean affiche; // variable d'affichage

void setup() {
  affiche = true; // initialisation à true
  numTable = 7; // initialisation à 7
  Serial.begin(9600); //initialisation de l'affichage

  //Affichage de l'entête du programme
```

```

Serial.println("*****");
Serial.println("Table de multiplication");
Serial.print("La table de : "); //pas de retour à la ligne
Serial.println(numTable); // affichage de la variable
Serial.println(); // saut de ligne pour aérer
}

void loop() {
  if (affiche) // test si vrai
  {
    // boucle de progression pour la multiplication
    for (int t = 0; t < 15; t++)
    {
      int resultat = numTable * t; // variable pour stocker le
résultat

      // Affichage de la ligne
      Serial.print(t);
      Serial.print(" x ");
      Serial.print(numTable);
      Serial.print(" = ");
      Serial.println(resultat);
    }
    Serial.println(); // saut de ligne
    Serial.println("*****");
    affiche=false; // passage à false pour ne plus afficher
  }
}

```

Résumé :

- *setup()* qui s'exécute qu'une fois ;
 - *loop()* qui s'exécute à l'infini ;
 - *void* (qui se met toujours devant loop et setup et dont vous comprendrez le sens plus loin dans ce cours) ;
 - *pinMode()* qui permet de définir une sortie en mode envoi d'électricité ou non ;
 - *digitalWrite()* qui envoie de l'électricité par une sortie ;
- delay()* qui met le programme en pause pendant un nombre défini de millisecondes.