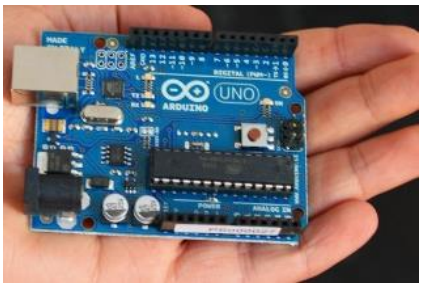
## 1- Arduino

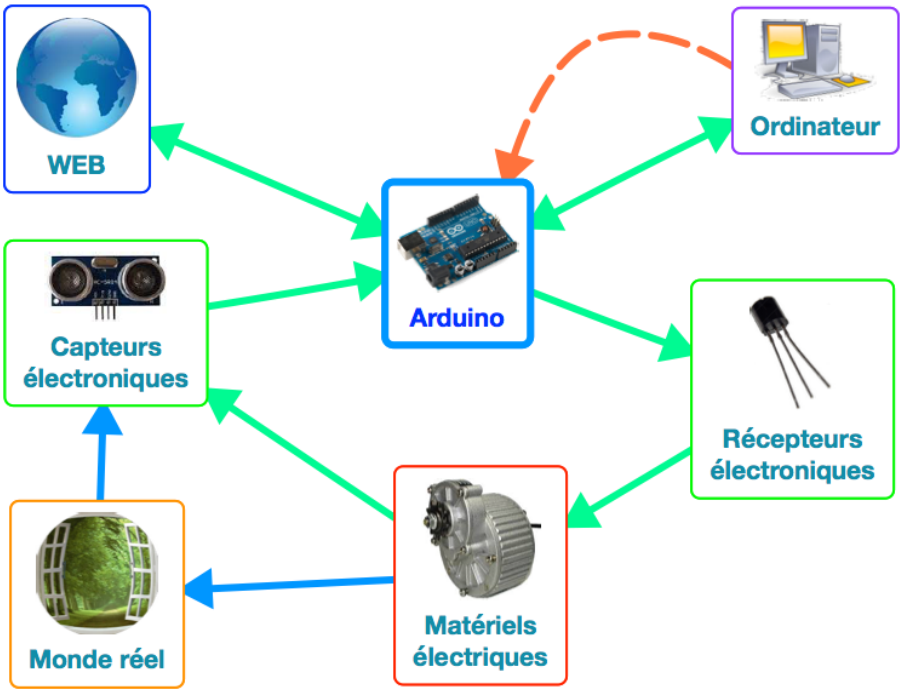
Arduino est le nom d'une gamme de cartes à microcontrôleurs, c'est-à-dire de cartes électroniques programmables.



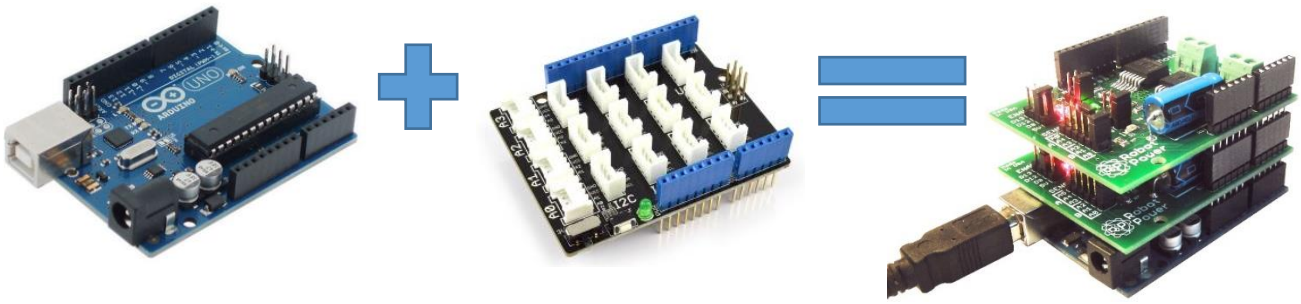
Elles utilisent toutes un même logiciel de programmation (environnement de développement ou IDE « Integrated Development Environment ») appelé logiciel Arduino.

Le langage de programmation utilisé est proche du langage C/C++.

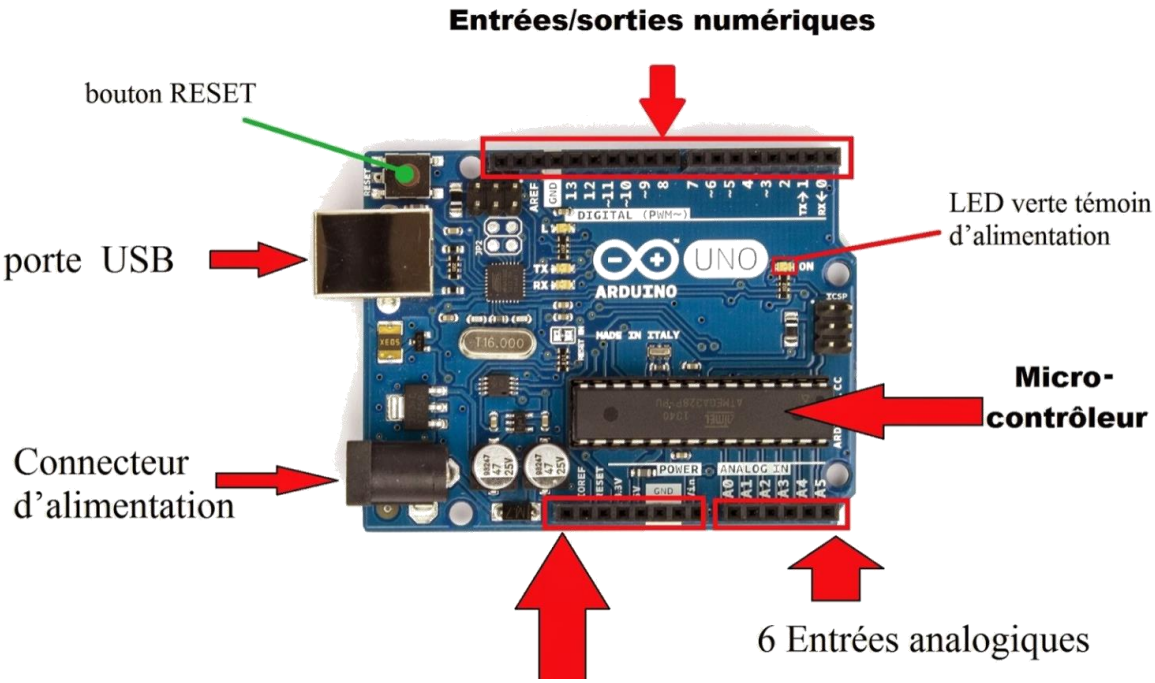
Les cartes Arduino permettent de traiter des données provenant de composants et capteurs divers (capteur de température, luminosité, mouvement ou boutons-poussoirs, …) et de communiquer des ordres pour allumer des lampes ou actionner des moteurs électriques, par exemple.




La connectique des cartes Arduino est conçue pour pouvoir y connecter des cartes additionnelles en les empilant sur la carte à microcontrôleur (sur deux rangées de connecteurs traversant). Ces cartes additionnelles sont appelées shield (« bouclier » en Anglais)



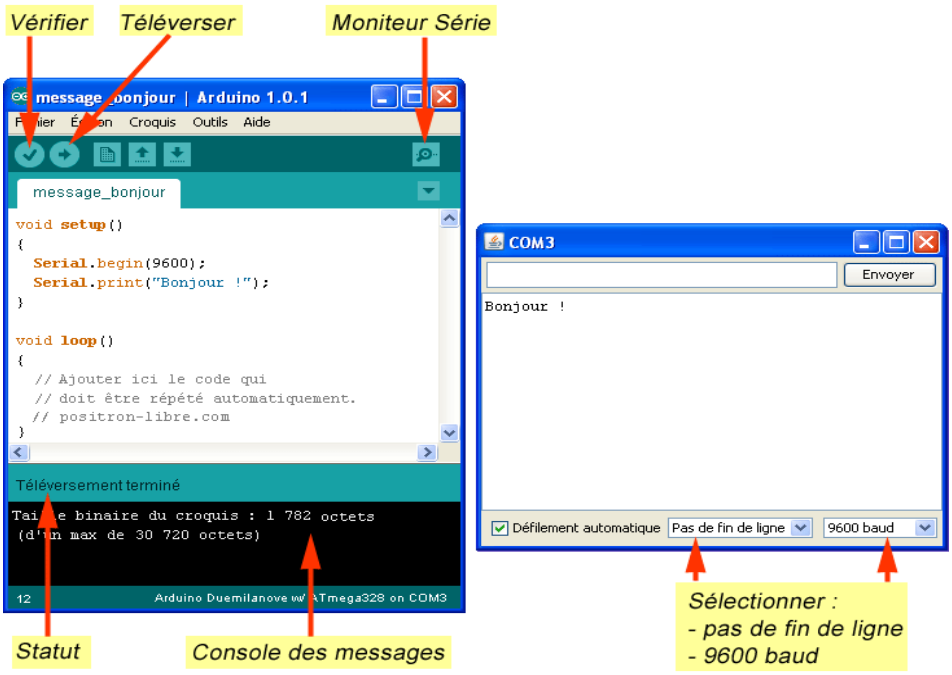
## 2- Les composants principaux de la carte




Bornier d'alimentation

## 3- L'interface logicielle

Sur l'ordinateur, le logiciel de programmation de la carte Arduino sert d'éditeur de code. Une fois le programme tapé ou modifié au clavier, il sera transféré et mémorisé dans la carte grâce à la liaison USB. Le câble USB a 2 fonctions : transférer le programme et alimenter la carte en énergie.



## 4- La structure d'un programme

| | |
|---|---|
| 1 | Déclaration des constantes<br>Déclaration des variables |
| 2 | **Setup**<br>Exécuté 1 seule fois, au lancement du programme<br>Configuration des entrées/sorties; Initialisation des variables |
| 3 | **Loop**<br>Exécuté en continu; Les instructions sont exécutées en boucle jusqu'à l'appui sur « Reset » |

1

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
```

2

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```

3

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

## 5- La ponctuation

- Toute **ligne** de code se termine par un point-virgule « ; »
- Le contenu d'une **fonction** est délimité par des accolades « { } »
- Les **paramètres** d'une fonction sont contenus entre parenthèses « ( ) »

## 6- Les variables

Présentation de quelques types de variables fréquemment utilisés :

| | |
|---|---|
| **int** | « integers » les entiers relatifs compris entre<br>« -32 768 ; ….-3 ; -2 ; -1 ; 0 ; 1 ; 2 ; ….32 767 » |
| **unsignedint** | « unsignedintegers » les entiers naturels compris entre<br>« 0 ; 1 ; 2 ; 3 ; …… ; 65 534 ; 65 535 » |
| **boolean** | Variable logique qui ne peut prendre que deux valeurs<br>« true » et « false » |
| **float** | Nombres décimaux (nombres à virgule). Attention il faut utiliser un point (et non une virgule) avant les décimales |
| **byte** | Variable codée sur 8 bits (donc 1 octet) qui peut prendre les valeurs entières comprises entre 0 et 255 |

## 7- Les opérateurs

### OPÉRATEURS ARITHMÉTIQUES

- = (égalité)
- + (addition)
- - (soustraction)
- * (multiplication)
- / (division)
- % (modulo)

### OPÉRATEURS DE COMPARAISON

- == (égal à)
- != (différent de)
- < (inférieur à)
- > (supérieur à)
- <= (inférieur ou égal à)
- >= (supérieur ou égal à)

### OPÉRATEURS BOOLÉENS

- && (ET booléen)
- || (OU booléen)
- ! (NON booléen)

### OPÉRATEURS COMPOSÉS

- ++ (incrémentation)
- -- (décrémentation)

## 8- Les structures de contrôle

### If …. else …then

« si… alors ….sinon »

**exemple :**

```
//si la valeur du capteur dépasse le seuil
if(valeurCapteur>seuil){
clignote();//appel de la fonction clignote
}
```

### while

« tant que »

**exemple :**

```
//tant que la valeur du capteur est supérieure à 250
while(valeurCapteur>250){
digitalWrite(5,HIGH); //allume la sortie 5
Serial.println(1);    //envoi le message "1" au port serie
    //en boucle tant que valeurCapteur est supérieue à 250
}
Serial.println(0);
digitalWrite(5,LOW);
```

### for

« pour une variable (exemple « i ») allant de telle valeur à telle autre… »

**exemple** :

```
//pour i de 0 à 255, par pas de 1
for (inti=0; i<= 255; i++){
analogWrite(PWMpin, i);
delay(10);
}
```

### Swich / case

« bascule en fonction de la valeur de la variable sur le cas n°…. »

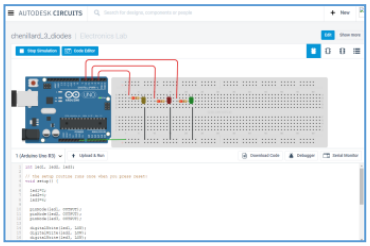
**exemple** :

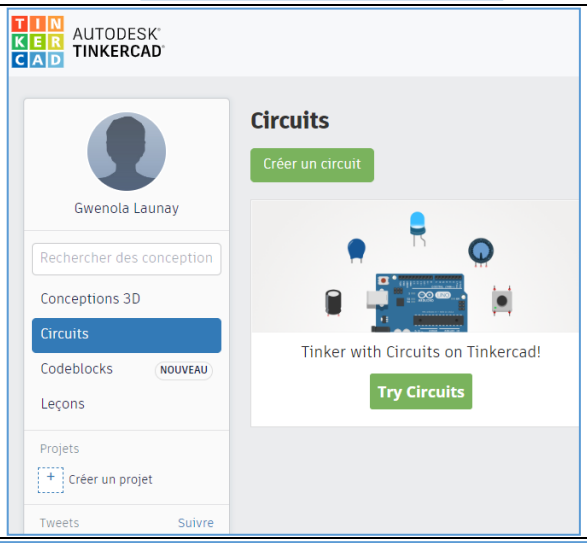
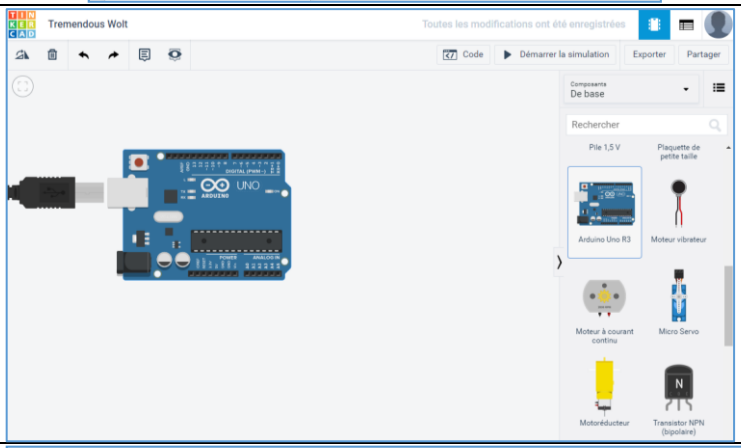
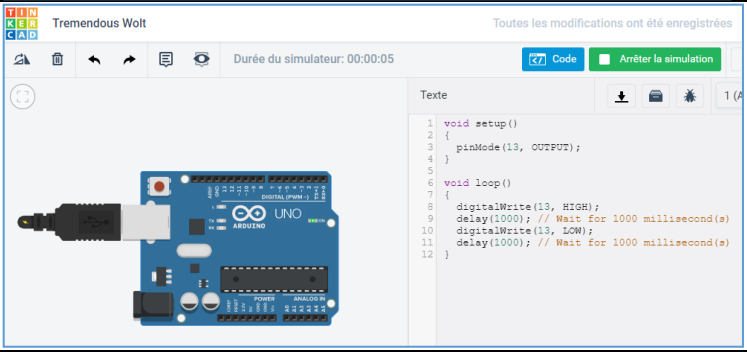
```
// fait un choix parmi plusieurs messages reçus
switch (message) {
case 0: //si le message est "0"
digitalWrite(3,HIGH);        //n'allume que la sortie 3
digitalWrite(4,LOW);
digitalWrite(5,LOW);
break;
case 1: //si le message est "1"
digitalWrite(3,LOW);
digitalWrite(4,HIGH);        //n'allume que la sortie 4
digitalWrite(5,LOW);
break;
case 2: //si le message est "2"
digitalWrite(3,LOW);
digitalWrite(4,LOW);
digitalWrite(5,HIGH);        //n'allume que la sortie 5
break;
}
```

## 9- Simulation du fonctionnement d'un programme

Pour simuler le fonctionnement d'une carte Arduino avec ses composants, on peut utiliser l'interface Tinkercad en ligne:

https://www.tinkercad.com/



| | |
|---|---|
| Créez votre compte puis cliquez sur « Circuits » |  |
| Choisir, paramétrer (exemple : valeur de la résistance) et relier les composants |  |
| Programmer le code et lancer la simulation |  |