

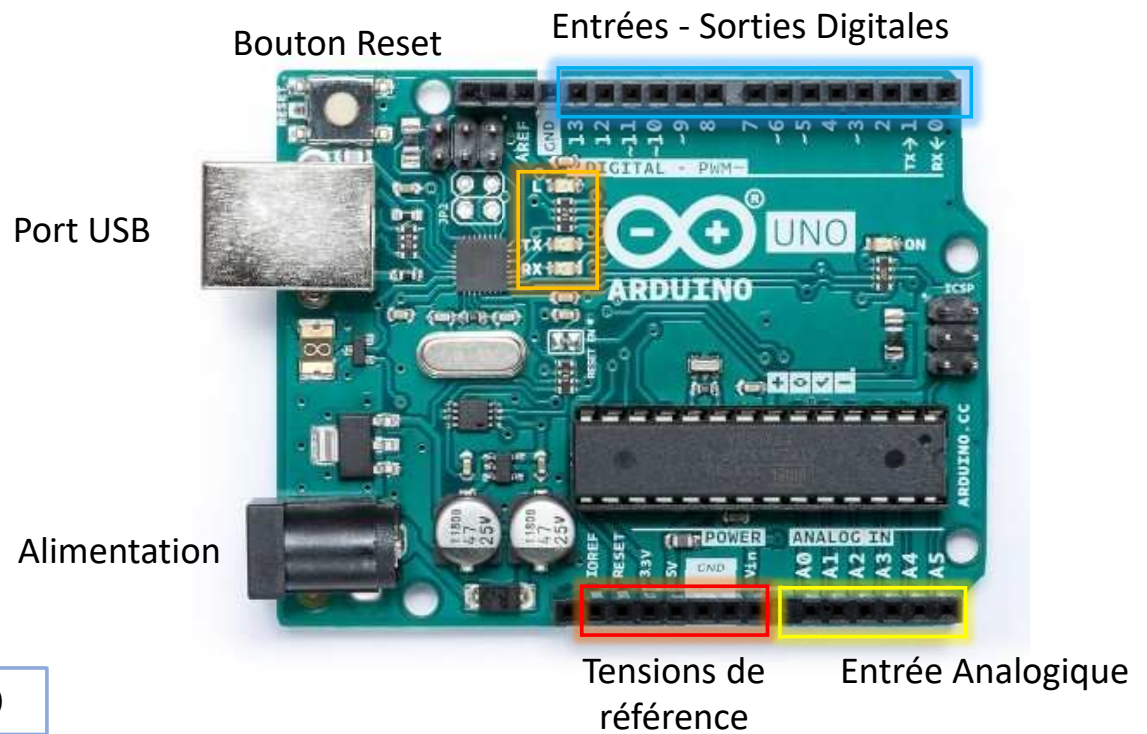
Introduction à la programmation en Arduino

Mesures et pilotage

Pourquoi choisir la programmation Arduino ?

- Carte Arduino : carte électronique (avec processeur et mémoire) programmable et totalement autonome, avec des entrées et des sorties électriques
 - Programmable : Flexibilité de l'utilisation
 - Autonome : indépendant, pas besoin d'ordinateur
 - Des entrées : possibilité de faire des mesures via des capteurs
 - Des sorties : permet d'alimenter et de piloter des organes
 - Très bas coût et libre

Présentation de la carte



Arduino UNO

Programmation de la carte

- Editeur libre (distribution Arduino)

Entête du code : déclaration des variables et constantes

Séquence « set-up » : préparation du programme

Séquence « loop » : programme se réalisant indéfiniment en autonomie

```
sketch_oct13a | Arduino 1.8.7
Fichier Édition Croquis Outils Aide

sketch_oct13a $
/*
  Déclaration des constantes
  */

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

13 Arduino/Genuino Uno sur COM7
```

Programmation de la carte

- Déclaration des constantes

→ Comme en programmation « C »

Déclaration des noms et des types

Types connus : int, float, bool, char...

Un exemple :

```
const int OutPin = 9;           // Led de fonctionnement
const int OutPin_circuit = 3;  // Alimentation du circuit de détection

int Max_sensor=73;             // Valeur max du capteur
int Min_sensor=18;             // Valeur min du capteur

int sensorValue = 0;           // valeur aux borne de la photoresistance
int outputValue = 0;           // valeur pour la diode d'éclairage
```

Programmation de la carte

- Séquence set-up

→ Séquence d'initialisation

→ Définition de la vitesse de communication avec la carte

Un exemple :

```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  pinMode(OutPin, OUTPUT);
  pinMode(OutPin_circuit, OUTPUT);
  pinMode(OutPin_eclaire, OUTPUT);
  pinMode(InPin_Pt100, INPUT);
}
```

Programmation de la carte

- Séquence loop

→ Programme se réalisant en boucle ...

→ ... sans aucun arrêt !

Un exemple :

```
void loop() {  
    // put your main code here, to run repeatedly:  
  
    // change the analog out value:  
    analogWrite(OutPin, 0);  
    delay(100);  
    analogWrite(OutPin, 50);  
    delay(100);  
    analogWrite(OutPin, 100);  
    delay(100);  
    analogWrite(OutPin, 150);  
    delay(100);  
    analogWrite(OutPin, 200);  
    delay(100);  
    analogWrite(OutPin, 250);  
    delay(100);  
}
```

Quelques fonctions de bases

- pinMode(pin, mode)

pin : numéro desortie digitale choisie

mode : INPUT ou OUTPUT

Ne renvoie rien

- digitalRead(pin)

pin : numéro d'entrée à lire

Renvoie HIGH ou LOW suivant la valeur de la tension aux bornes de l'entrée

- digitalWrite(pin, value)

pin : numéro de sortie

Value : HIGH ou LOW

Ne renvoie rien

Quelques fonctions de bases

- `analogRead(pin)`

pin : numéro d'entrée analogique à lire

Renvoie une valeur en 0 et 1023, à faire correspondre à la tension de référence de la carte (habituellement 0 → 5V)

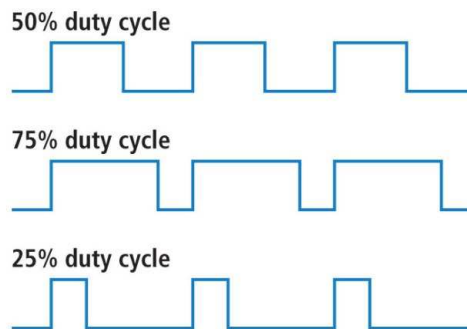
- `analogWrite(pin, value)`

pin : numéro de sortie digitale avec le symbole ~

Value : entier de 0 à 255, correspondant au « Duty cycle » (de toujours éteint à toujours allumé)

Ne renvoie rien

L'idée : faire varier la valeur moyenne du signal pour alimenter ou piloter des organes lents



Fréquence du signal :
490 Hz ou 980 Hz

Quelques fonctions de bases

- `delay(value)`

Value : entier

Stop l'exécution pendant un temps donné par « value »

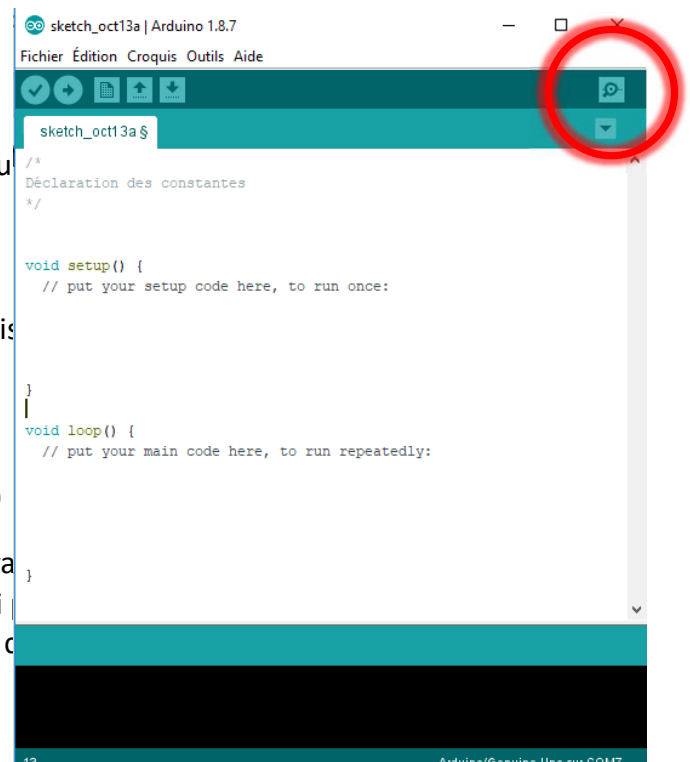
- `millis()`

Renvoie un entier correspondant au nombre de millisecondes depuis le début du programme

- `Serial.print(value)` et `Serial.println(value)`

Value : chaîne de caractères (entre " ") ou nom de variable

Écrit sur le port série de la carte en format ASCII, qui peut être lu en lisant ce port avec un ordinateur (déjà intégré au code)



Quelques structures

```
while(condition)
{ // statement(s) }
```

```
for (initialization; condition; increment)
{ //statement(s); }
```

```
if (condition)
{ //statement(s) }
```

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