Arduino - Mémento

Introduction aux microcontrôleurs

Arduino est la marque d'une plateforme de prototypage open-source qui permet aux utilisateurs de créer des objets électroniques interactifs à partir de cartes électroniques équipées d'un ***microcontrôleur***.

Un ***microcontrôleur*** est un ***circuit intégré*** qui intègre les éléments essentiels d'un ordinateur : processeur, unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration (taille réduite), une plus faible consommation électrique et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

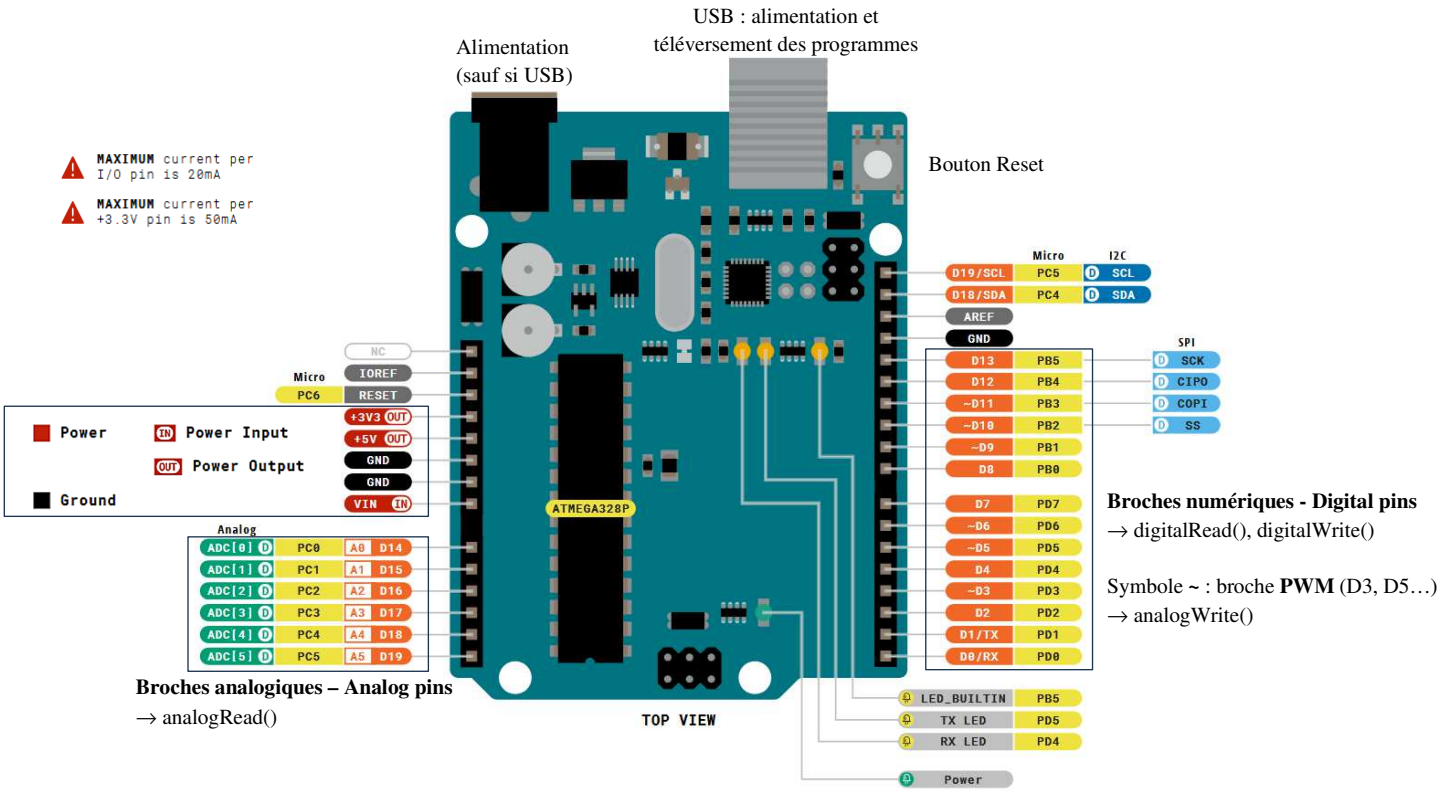
Un microcontrôleur peut être programmé pour analyser et produire des signaux électriques.
Ils sont utilisés dans les ***systèmes embarqués*** pour piloter des robots, dans les voitures, les avions, les récepteurs GPS, les télécommandes, l'électroménager, les jouets, la téléphonie mobile, la domotique, etc.

Dans le cas d'Arduino, les langages de programmation utilisés sont **C** et **C++**.
Un **IDE** (environnement de Développement Intégré) Arduino permet d'écrire et de téléverser les programmes dans la mémoire ROM du microcontrôleur (cf. ci-dessous).
Rq : Pyzo ou Spyder sont des IDE pour Python, VisualStudio est un IDE multi langages.

Le microcontrôleur des cartes Arduino utilise deux types de mémoires.

- ✓ Une ***mémoire morte*** (**ROM**, pour l'anglais read-only memory), une mémoire au contenu non volatile utilisée pour enregistrer des informations qui doivent être conservées lorsque l'appareil qui les utilise n'est plus sous tension.
  La carte Arduino utilise une **EEPROM** (Electrically-Erasable Programmable Read-Only Memory ou mémoire morte effaçable électriquement et programmable) qui peut être facilement effacée et réécrite à l'aide d'un courant électrique.
  C'est dans cette mémoire que seront stockés les programmes.
- ✓ Une ***mémoire vive*** (**RAM** acronyme anglais pour random-access memory), « mémoire à accès aléatoire » à accès rapide dans laquelle peuvent être enregistrées des données volatiles.

Carte Arduino Uno - Brochage



La modulation **PWM** (Pulse Width Modulation en anglais) ou modulation de largeur d'impulsions (MLI en français) est une technique utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits numériques.

Instructions pour lire le signal appliqué à une broche :

- ✓ **`analogRead()`** sur une entrée ***analogique*** (A0 à A5) (tension délivrée par un capteur par exemple) ;
- ✓ **`digitalRead()`** sur une entrée ***numérique*** (D2, D4, D7, D8, D12, D13).

Instructions pour envoyer un signal à une broche :

- ✓ **`analogWrite()`** sur une broche ***numérique*** **PWM** simulant un signal ***analogique*** (D3, D5, D6, D9, D10, D11) ;
- ✓ **`digitalWrite()`** sur une broche ***numérique***.

Définir le comportement d'une broche (entrée ou sortie) : **`pinMode(`***`broche, mode`***`)`** avec *`mode`* = `INPUT` ou `OUTPUT`.

Conversion analogique - Numérique

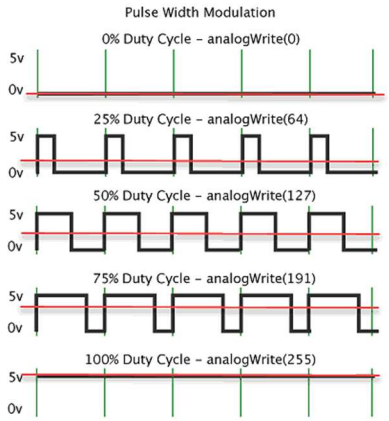
Convertisseur analogique numérique (CAN ou ADC pour analog-to-digital-converter en anglais) *n* bits : une grandeur analogique appliquée au CAN est discrétisée sur 2^n valeurs (de 0 à 2^n -1).

La carte Arduino UNO est équipée d'un **CAN 10 bits** : on dispose donc de 2^{10} = 1024 valeurs, de **0 à 1023**, pour représenter la grandeur analogique appliquée sur une broche.
La fonction `analogRead()` renvoie la valeur binaire issue du CAN.

Il faudra donc ***convertir*** la valeur binaire en fonction des caractéristiques du signal envoyé pour accéder à la valeur physique de la grandeur mesurée.

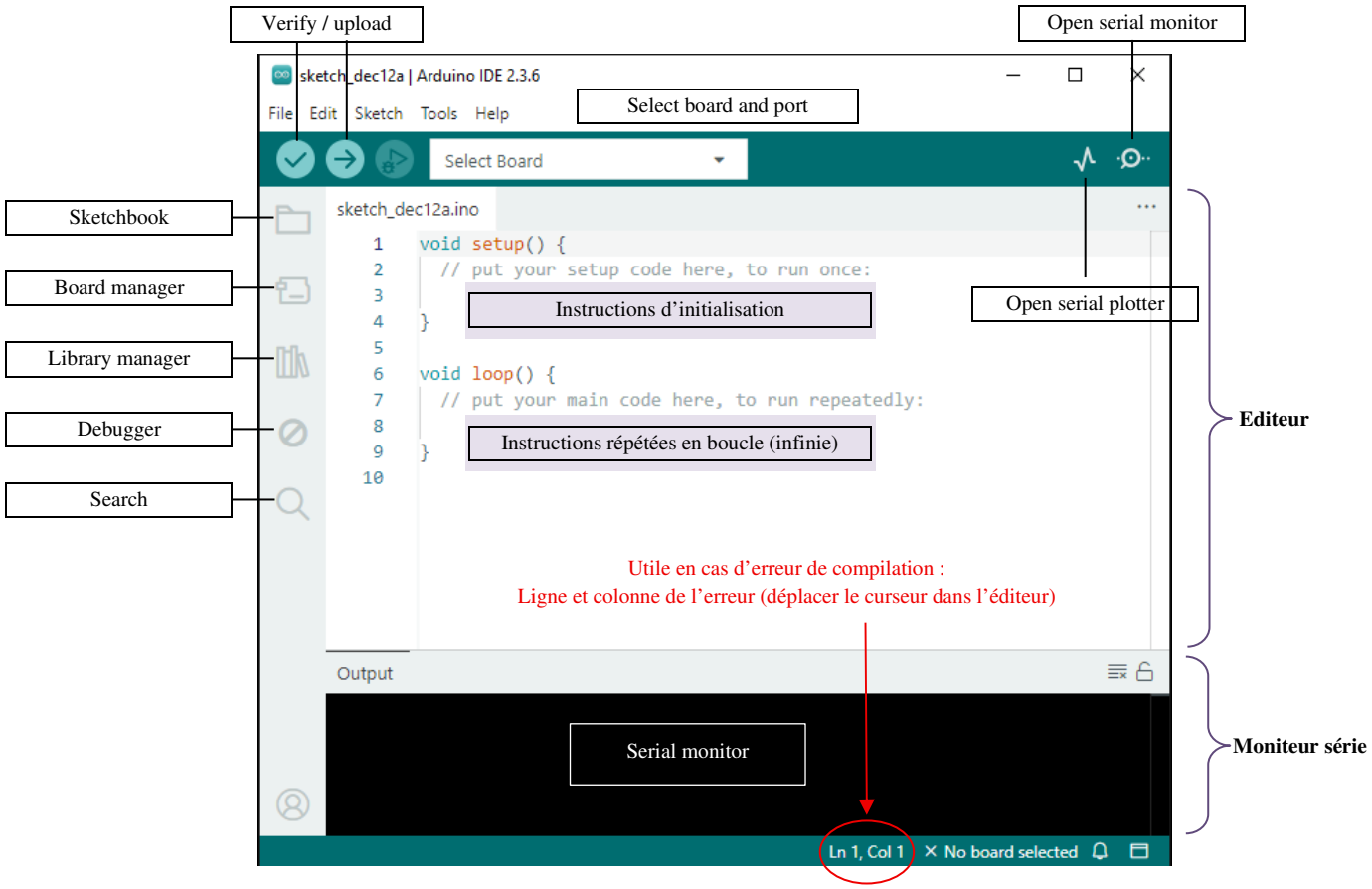
Modulation de largeur d'impulsion (sorties PWM)

Modulation de largeur d'impulsion (schéma ci-contre) : en faisant varier le ***rapport cyclique*** (durée du niveau haut / durée du niveau bas) des impulsions, on fait varier la ***valeur moyenne*** du signal.
C'est cette valeur moyenne qui simule un signal analogique. Le rapport cyclique pouvant varier rapidement, la valeur moyenne peut également évoluer au cours du temps et simuler un signal analogique variable dans le temps.



Sélectionner le type de carte Arduino et le port sur lequel elle est connectée dans la liste déroulante « Select Board ».

Un programme Arduino est appelé « ***sketch*** » (parfois traduit par « croquis » ou « esquisse »), l'extension de fichier est **.ino**
Il est tapé dans la zone d'édition de l'interface, il doit être ***compilé*** (transformation du code source en fichier binaire) puis téléversé sur la carte.
Des informations sur le processus de compilation sont affichées dans l'onglet « Output » sous l'éditeur (éventuelles erreurs, statut final, mémoire disponible…).

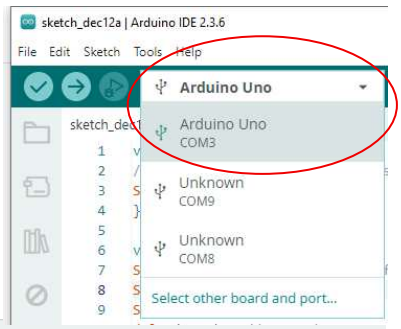


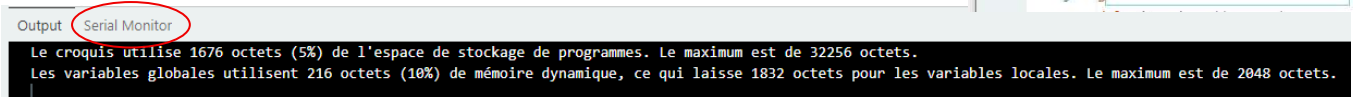
Sketchbook : programmes enregistrés sur l'ordinateur.
Board manager : packages nécessaires avec certaines cartes d'extension ajoutées sur la carte Arduino (WiFi,…).
Library manager : librairies/modules à utiliser pour les capteurs par exemple.
Serial moniter : visualisation du flux de données issu de la carte.
Les instructions **Serial.print()** et **Serial.println()** permettent d'écrire dans le moniteur série.

Serial Plotter : visualisation de graphes.

Connexion de la carte à un ordinateur

1. Brancher la carte et lancer l'IDE.
2. Sélectionner la carte et le port dans la liste déroulante (cf. ci-contre).
   En cas de non détection, essayer de changer de port et de relancer l'IDE.
3. Créer un nouveau sketch (File / New Sketch) ou ouvrir un sketch enregistré (le dernier utilisé est préchargé).
4. Taper ou modifier le code ; l'aide en ligne est abondante.
5. Compiler le code .
   Si tout se passe bien un message s'affiche en blanc (sinon lire le message affiché en rouge et remédier aux problèmes) :





6. Téléverser le code .
7. Cliquer sur l'icône du moniteur série ou sur l'onglet « Serial Monitor » s'il est déjà ouvert.