

# Découvre Arduino IDE et la logique d'un programme

## À quoi sert l'Arduino IDE ?

### 1. C'est quoi un microcontrôleur ?

Un **microcontrôleur** est un **petit cerveau électronique**.

Il est capable de recevoir des informations, de **les traiter**, et de **faire des actions**.

Par exemple :

- Allumer une LED quand on appuie sur un bouton
- Ouvrir une barrière quand on passe une carte
- Mesurer la température et afficher la valeur

### Exemples de cartes avec microcontrôleur :

Ces cartes sont comme des mini-ordinateurs que l'on peut programmer :

Carte	Caractéristiques
Arduino Uno	Simple, idéale pour débiter
Arduino Nano	Plus petite, mais similaire à l'Uno
ESP32	possède le Wi-Fi et Bluetooth intégrés
...	...

Chacune de ces cartes contient un **microcontrôleur**, c'est-à-dire un cerveau qui exécute les programmes.

## ☐ 2. L'Arduino IDE : l'outil pour programmer ces cartes

L'**IDE Arduino** (IDE = Environnement de Développement Intégré) est le **logiciel** qui permet d'écrire des programmes pour ces cartes.

Avec l'Arduino IDE, on peut :

- Écrire le **code** en langage **C/C++ simplifié**
- **Vérifier** si le code contient des erreurs
- **Envoyer** le programme à la carte par un câble USB

**Si le logiciel n'est pas encore installé sur votre ordinateur.**

Vous pouvez le télécharger ici : <https://www.arduino.cc/téléchargement>

Il vous suffit ensuite de sélectionner le système d'exploitation que vous utilisez.



 **Arduino IDE 2.3.6**

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

**SOURCE CODE**

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

**DOWNLOAD OPTIONS**

**Windows** Win 10 and newer, 64 bits  
**Windows** MSI installer  
**Windows** ZIP file

**Linux** AppImage 64 bits (X86-64)  
**Linux** ZIP file 64 bits (X86-64)

**macOS** Intel, 10.15: "Catalina" or newer, 64 bits  
**macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

# 3. Structure d'un programme Arduino : les bases à connaître

Un programme Arduino est toujours organisé de la **même manière**, avec **différentes parties bien définies** :

```
// 1. Importation des librairies
// 2. Déclaration des variables et constantes
// 3. setup() : s'exécute une fois au démarrage
// 4. loop() : s'exécute en boucle
```

## 1. Importation des **librairies** (*facultatif*)

Une **librairie** est un morceau de code déjà écrit par d'autres, que l'on peut utiliser pour **communiquer avec des capteurs, des écrans, des moteurs, etc.**

On les déclare **tout en haut** du programme. Et elle se note de cette façon

Exemple :

```
#include <Nom_de_la_librairie.h>
```

## 2. Déclaration des **constantes** et **variables globales**

Juste **sous les librairies**, on déclare :

- Les **constantes** (valeurs fixes, comme les numéros de broches sur lesquels l'on s'est connecté)
- Les **variables globales** (utilisées partout dans le programme)

## 3. `void setup()` : au démarrage

- Cette fonction est **exécutée une seule fois** au début.
- On y met tout ce qu'il faut **initialiser** : la communication série, les broches d'entrée/sortie, les capteurs, etc.
- Par exemple, si je souhaite programmer un capteur qui me donnera une information à un moment donné. J'obtiendrai un code similaire à :

```
void setup() {
    capteur(INPUT); // Prépare le capteur en entrée. INPUT = entrée
```

Attention : Dans votre code vous pouvez écrire des commentaires. Ces commentaires sont donc du texte qui ne sera pas lu par le programme mais qui vous permettra d'expliquer votre

programme au fur et à mesure.

Cela vous permettra donc de mieux vous repérer dans votre code, de mieux déboguer votre code mais aussi de le rendre plus lisible pour un tiers.

Comment faire ? :

Il vous suffit d'écrire // devant chaque ligne de code. Si vous écrivez un commentaire sur plusieurs lignes vous pouvez commencer par /\* et finir par \*/

## □ □ void loop() : la répétition

- Cette fonction est **exécutée en boucle sans arrêt**
- C'est là qu'on écrit le comportement du programme (réagir aux boutons, clignoter, mesurer, etc.)

Exemple :

```
void loop() {  
  Allumer capteur;  
  Attendre (10 secondes);  
  Éteindre capteur;  
}
```

Attention les codes que je vous indique ne sont pas fonctionnels pour le moment, ils permettent de comprendre petit à petit le fonctionnement et la structure d'un code sur Arduino IDE.

Si vous avez bien fait attention. Vous pouvez voir que lorsque je déclare une fonction. Il y a des parenthèses à côté et pour ouvrir la fonction une { et pour la fermer une autre } mais dans l'autre sens.

## 4. Exercice

Essayer sur papier (tableau, ardoise, etc...) de récapituler le fonctionnement et la structure d'un code Arduino.

---

Revision #4

Created 6 July 2025 17:27:49 by Gaëtan Carron

Updated 6 July 2025 19:17:42 by Gaëtan Carron