Tutoriel Arduino

Qu'est-ce qu'Arduino ?

Arduino est une plateforme électronique open-source composée d'une carte matérielle (avec un microcontrôleur) et d'un logiciel de développement (l'IDE Arduino). Elle permet de créer des projets interactifs en programmant des circuits électroniques.

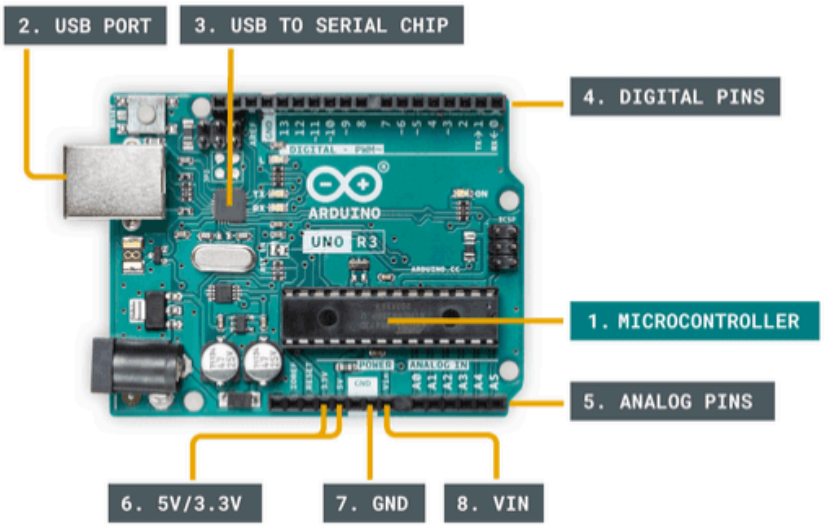
Installation de l'IDE Arduino

1. **Télécharger l'IDE Arduino**
   Accédez à Software | Arduino et téléchargez l'IDE pour votre système d'exploitation (Windows, macOS, Linux).
2. **Installer le logiciel**
   Suivez les instructions d'installation. Assurez-vous que les pilotes Arduino sont correctement installés.
3. **Connexion de la carte Arduino**
   Branchez votre carte Arduino à votre ordinateur avec le câble USB. Dans l'IDE Arduino, allez dans **Outils > Type de carte** et sélectionnez votre modèle de carte (ex : "Arduino Uno"). Sélectionnez ensuite le bon port COM dans **Outils > Port**.

Using the Arduino Software (IDE) | Arduino Documentation

Anatomie d'une carte Arduino

Chaque carte Arduino a des composants clés qui peuvent varier légèrement d'un modèle à l'autre. Cependant, voici les éléments que l'on retrouve généralement sur la plupart des cartes Arduino :



1. **Microcontrôleur** : Le cerveau de la carte, responsable de l'exécution des programmes.
2. **Port USB** : Utilisé pour connecter la carte Arduino à un ordinateur afin de charger des programmes.
3. **Chip USB-to-Serial** : Permet de traduire les données du port USB pour qu'elles soient compréhensibles par le microcontrôleur.
4. **Broches numériques (Digital pins)** : Broches qui utilisent une logique binaire (0 ou 1) pour activer ou désactiver des composants comme des LED ou des boutons.
5. **Broches analogiques (Analog pins)** : Broches capables de lire des valeurs analogiques dans une résolution de 10 bits (0-1023).
6. **Broches 5V/3.3V** : Pour alimenter des composants externes.
7. **Broche GND** : Connexion à la masse (0 volt) pour compléter un circuit.
8. **Broche VIN** : Permet de connecter une alimentation externe.

Fonctionnement de base

Chaque carte Arduino peut exécuter un programme appelé **sketch**, écrit en langage Arduino (une dérivation simplifiée du C++). Voici deux fonctions essentielles présentes dans tout programme Arduino :

- **void setup()** : Cette fonction s'exécute une seule fois au démarrage de la carte, et elle sert à initialiser les paramètres comme le mode des broches (entrée/sortie).
- **void loop()** : Cette fonction s'exécute en boucle et contient les instructions à répéter indéfiniment (par exemple, allumer/éteindre une LED en fonction d'un bouton).

```
void setup() {
  // Initialisation du programme
}

void loop() {
  // Code exécuté en boucle
}
```

Les bases des circuits

Un circuit électronique nécessite au moins un composant actif (comme une LED) et des fils conducteurs pour permettre au courant de circuler. Par exemple, pour allumer une LED avec Arduino :

- Connectez un fil depuis une broche numérique (par ex. broche 13) jusqu'à la LED en passant par une résistance (pour limiter le courant).
- Reliez ensuite la LED à la broche GND pour fermer le circuit.
- Le code allumera et éteindra la LED selon l'état de la broche.

Signaux électroniques : Analogiques vs Numériques

- **Signal analogique** : Représente une plage continue de valeurs. Dans Arduino, cela se traduit souvent par une plage de tension entre 0 et 5V (ou 3,3V), correspondant à une échelle de 0 à 1023.
- **Signal numérique** : Utilise seulement deux états (0 ou 1, bas ou haut). Cela permet par exemple d'allumer ou d'éteindre un composant.

Capteurs et actionneurs

- **Capteur** : Dispositif qui mesure une propriété physique (température, lumière, etc.) et la convertit en signal électrique. Par exemple, un capteur de température envoie un signal proportionnel à la température ambiante.
- **Actionneur** : Composant qui agit sur le monde physique (allume une lumière, fait tourner un moteur, etc.) en fonction d'un signal.

Arduino Tutorials (arduinogetstarted.com)

Mémoire dans Arduino

Arduino possède deux types de mémoire :

- **SRAM** : Utilisé pour stocker les variables pendant l'exécution du programme. Cette mémoire est volatile et se réinitialise lorsque la carte est éteinte.
- **Flash** : Utilisé pour stocker le programme principal. Cette mémoire n'est pas effacée lorsque la carte est éteinte.

API Arduino et structure du programme

Le langage Arduino, basé sur le C/C++, est simplifié pour contrôler le matériel Arduino. Voici les trois éléments principaux de l'API :

- **Fonctions** : Exécuter des actions comme lire/écrire des états, calculer des valeurs, etc. Ex : `digitalWrite()`, `analogRead()`.
- **Variables** : Gérer des données, par exemple : `int`, `boolean`, `float`.
- **Structures de contrôle** : Comme dans C++, on utilise `if`, `else`, `for`, `while`, etc., pour structurer le code.

Les Bibliothèques Arduino

Les **bibliothèques Arduino** sont une extension de l'API standard d'Arduino. Elles regroupent des milliers de bibliothèques, à la fois officielles et créées par la communauté, facilitant l'utilisation de composants et de fonctionnalités avancées.

Pourquoi utiliser des bibliothèques ?

Les bibliothèques simplifient l'utilisation de fonctionnalités complexes, comme la lecture d'un capteur spécifique, le contrôle d'un moteur, ou la connexion à Internet. Au lieu d'écrire tout le code nécessaire vous-même, vous pouvez installer une bibliothèque, l'inclure en haut de votre code, et utiliser les fonctionnalités déjà disponibles.

Toutes les bibliothèques Arduino sont **open source** et peuvent être utilisées gratuitement par n'importe qui.

Comment utiliser une bibliothèque ?

Pour utiliser une bibliothèque, il suffit de l'inclure en haut de votre code avec la directive `#include`. Par exemple : #include <NomDeLaBibliothèque.h>

Après avoir inclus la bibliothèque, vous pouvez utiliser ses fonctions dans votre programme. La plupart des bibliothèques viennent également avec des exemples qui vous aideront à bien démarrer avec la bibliothèque.

Installation d'une bibliothèque

1. **Arduino IDE** : Vous pouvez installer des bibliothèques directement depuis l'IDE Arduino en allant dans l'onglet **"Croquis"** > **"Inclure une bibliothèque"** > **"Gérer les bibliothèques..."**.
2. **Recherche** : Vous pouvez rechercher des bibliothèques spécifiques à l'aide du gestionnaire de bibliothèques.
3. **Installation manuelle** : Pour des bibliothèques téléchargées ailleurs, allez dans **"Inclure une bibliothèque"** > **"Ajouter une bibliothèque .ZIP"** pour l'installer manuellement.

Explorer les bibliothèques Arduino

Vous pouvez parcourir toutes les bibliothèques, qu'elles soient officielles ou créées par la communauté, sur la page des **Bibliothèques Arduino**. Cela vous permet de découvrir de nouvelles fonctionnalités et d'élargir les possibilités de vos projets Arduino.

Get to know Arduino Libraries | Arduino Documentation

Potentiomètre

Un **potentiomètre** est un dispositif permettant de diviser la tension et d'ajuster les niveaux de signaux en fonction de la position de sa molette. C'est essentiellement une résistance variable avec trois broches : deux pour les bornes extrêmes de la résistance fixe et une pour la borne du curseur qui se déplace le long de la résistance.

Exemple d'utilisation avec Arduino :

En connectant un potentiomètre à un Arduino, on peut lire la position du curseur avec une entrée analogique pour contrôler des éléments comme la luminosité d'une LED, la vitesse d'un moteur, ou tout autre dispositif.

Connexion :

- **Broche 1 (GND)** : Connectée à la masse (GND) de l'Arduino.
- **Broche 2 (Sortie)** : Connectée à une broche analogique de l'Arduino (par exemple A0).
- **Broche 3 (VCC)** : Connectée à l'alimentation 5V de l'Arduino.

Basics of Potentiometers with Arduino | Arduino Documentation

Bouton poussoir

- Connectez une broche du bouton à un pin numérique de l'Arduino (par exemple, **pin 7**).
- Connectez l'autre broche du bouton à la masse (GND).
- Placez une résistance de 10kΩ entre le pin 7 et l'alimentation (5V) pour éviter des signaux flottants (ou vous pouvez activer la résistance de pull-up interne de l'Arduino, comme indiqué plus bas).

Le bouton fonctionne en changeant l'état logique du pin : lorsque le bouton est pressé, le pin est connecté à la masse (LOW). Lorsque le bouton n'est pas pressé, la résistance tire la broche vers un état logique haut (HIGH).

Arduino - Button | Arduino Tutorial (arduinogetstarted.com)

Récepteur-Transmetteur Asynchrone Universel (UART)

Le **récepteur-transmetteur asynchrone universel** (UART) est un protocole de communication série utilisé pour envoyer des données série via USB ou via les broches TX/RX.

Avec la classe **Serial**, vous pouvez envoyer et recevoir des données depuis et vers votre ordinateur via USB ou vers un appareil connecté via les broches RX/TX de l'Arduino.

- Lorsque vous envoyez des données via USB, nous utilisons **Serial**. Ces données peuvent être visualisées dans le **Moniteur Série** de l'IDE Arduino.
- Lorsque vous envoyez des données via les broches RX/TX, nous utilisons **Serial1**.

La classe **Serial** comprend plusieurs méthodes essentielles, dont voici quelques-unes :

- **begin()** : commence la communication série avec un débit en bauds spécifié (de nombreux exemples utilisent soit 9600, soit 115200).
- **print()** : affiche le contenu dans le Moniteur Série.
- **println()** : affiche le contenu dans le Moniteur Série et ajoute une nouvelle ligne.
- **available()** : vérifie si des données série sont disponibles (par exemple, si vous envoyez une commande depuis le Moniteur Série).
- **read()** : lit les données du port série.
- **write()** : écrit des données sur le port série.

Universal Asynchronous Receiver-Transmitter (UART) | Arduino Documentation

Voici quelques ressources pour commencer votre projet :

Getting Started with Arduino | Arduino Documentation

Troubleshooting Arduino Sketches | Arduino Documentation

The Arduino Guide to Low Power Design | Arduino Documentation

Intelligence artificielle avec Arduino UNO

Il est tout à fait possible d'intégrer l'intelligence artificielle dans votre projet Arduino, en utilisant des algorithmes simples comme la régression linéaire. Voici les étapes à suivre :

Étape 1 : Programmer l'Arduino pour recueillir des données

Commencez par programmer l'Arduino pour collecter des données à partir de capteurs (par exemple, un capteur de température, de luminosité, ou d'humidité). Cela permettra de rassembler les informations nécessaires pour entraîner votre modèle.

Étape 2 : Collecter les données

Téléversez le programme dans l'Arduino et utilisez le *Serial Monitor* pour visualiser et collecter les données. Vous pouvez enregistrer ces données dans un fichier texte (TXT) pour un usage ultérieur. Pour faciliter la collecte de données dans différents environnements, vous pouvez ajouter un bouton à votre projet pour déclencher manuellement l'enregistrement des données, par exemple, selon les variations de luminosité ou d'humidité.

Étape 3 : Analyser les données avec Python

Une fois les données collectées, vous pouvez les analyser en utilisant un script Python. Des bibliothèques comme *pandas*, *matplotlib*, et *numpy* vous aideront à manipuler et visualiser les données. Vous pouvez ensuite appliquer un modèle de régression linéaire (ou un autre modèle, selon vos besoins) pour établir une relation entre les variables.

Étape 4 : Intégrer le modèle dans l'Arduino

Après avoir trouvé un modèle adapté à vos données, vous pouvez intégrer cette fonction directement dans le code Arduino. Cela vous permettra d'utiliser le modèle pour prédire de futures valeurs en temps réel dans votre projet.

Exemple de projet

Vous pouvez consulter le tutoriel AI Lamp - How to Use AI in Arduino! : 9 Steps (with Pictures) - Instructables pour un exemple concret et simple de l'utilisation de l'intelligence artificielle avec un modèle de régression linéaire.

Important : Assurez-vous d'enregistrer soigneusement les données que vous collectez pour démontrer que vous avez bien utilisé vos propres données pour entraîner votre modèle.

Apprentissage Automatique sur Arduino Nano 33 BLE Sense

Get Started With Machine Learning on Arduino

Nano 33 BLE Sense | Arduino Documentation