Installation et premiers essais avec un ESP32

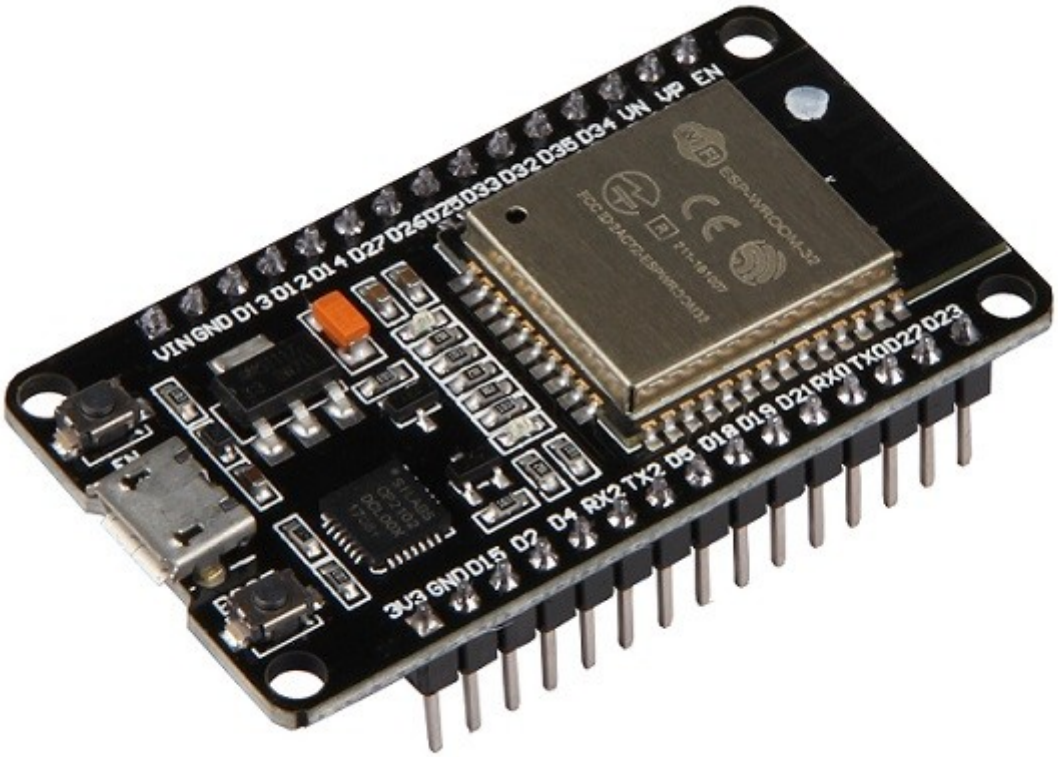
Prérequis : Pour cet article, je considère les prérequis suivants:

- VSCode installé et fonctionnel
- Base de programmation en C/C++
- Premiers programmes avec l'Arduino Uno

Présentation de l'ESP32

L'ESP32 est un microcontrôleur 32 bits, conçu par Espressif (Chine) et fabriqué par TSMC. Ce microcontrôleur à très faible consommation et coût, intègre de nombreux périphériques ainsi que le Wifi + Bluetooth. Toutes ces caractéristiques le rendent particulièrement adapté pour les projets en embarqué ou IoT.

[



](https://innovation.iha.unistra.fr/uploads/images/gallery/2023-07/image-1688566829354.jpg | width=100)

Pour être plus précis, il faut rappeler qu'il s'agit d'une famille de microcontrôleurs ESP32 dont les caractéristiques varient en fonction du modèle, ou de son intégration sur une carte de développement. Parmi les caractéristiques que nous pouvons noter :

- Microprocesseur single core ou dual core jusqu'à 240 MHz
- Jusqu'à 4MB de mémoire flash
- WiFi 802.11 b/g/n
- Bluetooth 4.2 BR/EDR ou 5 LE
- 26x E/S digitales (3.3V)
- 12x entrées analogiques
- 4x SPI, 2x I²S, 2x I²C, 3x UART, CAN 2.0, IR, Touch Sensor
- Capteur de température intégré
- Power Management Unit
- Modules de cryptographie hardware
- ...

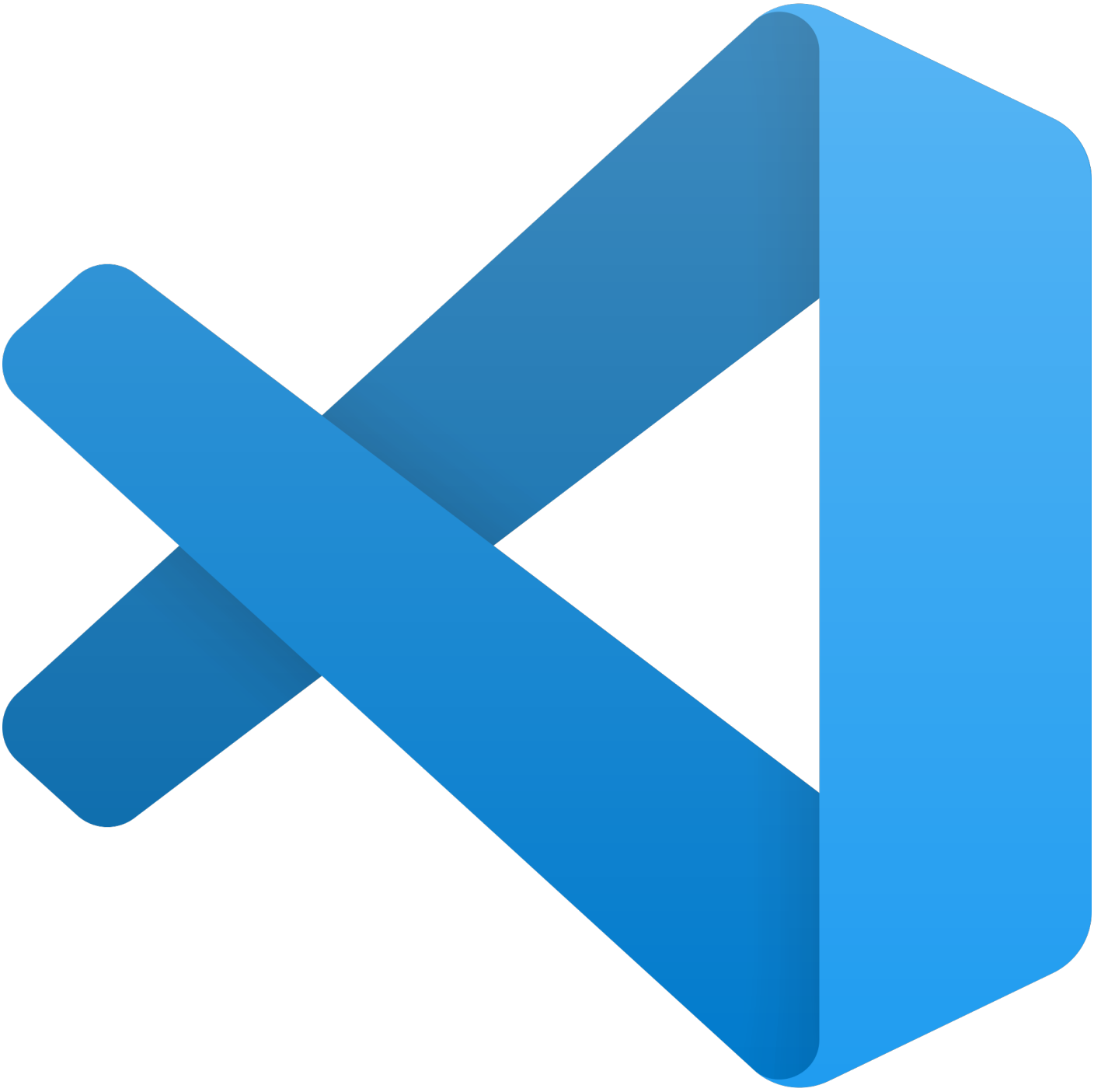
Pour le prix, les caractéristiques sont généreuses. En Europe, un ESP32 est à moins de 1.1€ et les cartes de développement se trouvent aux alentours de 10€. A titre de comparaison, un PIC16F1939, microcontrôleur 8 bits de Microchip coûte 2.6€

Avec un tarif aussi accessible, les cartes de développement (DevKit) peuvent être difficiles à déchiffrer pour connaître le modèle exact de l'ESP32 intégré, notamment quand on passe par des fournisseurs asiatiques low-cost pour économiser encore quelques euros. Les fournisseurs européens sont souvent plus à jour dans les descriptions des ESP32 intégrés dans les cartes.

Les outils pour programmer les ESP32

Kunbuntu 22.04 est utilisé pour cet article. Vscode est installé manuellement en .deb et non en Snap. Microsoft propose une documentation de l'installation de VSCode dans le lien suivant.

Visual Studio Code (VSCode)



Visual Studio Code (VSCode) est un éditeur de code extensible, multi plateforme (Windows, macOS et Linux), open source et gratuit, développé par Microsoft. Les fonctionnalités de VSCode peuvent être étendues par l'intermédiaire des extensions qui permettront par exemple:

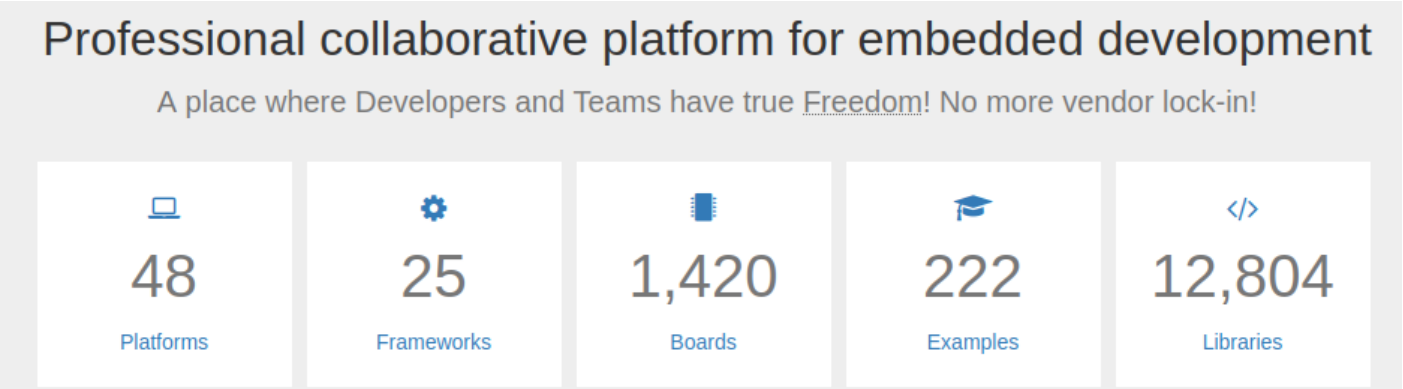
- l'intégration de l'autocomplétion IntelliSense
- l'intégration de Git
- le support du langage Python
- ...

D'une grande polyvalence, Visual Studio Code peut également servir d'IDE pour programmer les microcontrôleurs de type ESP32, STM32, TI, Microchip avec l'extension PlatformIO.

PlatformIO

PlatformIO est une platforme open source, compilant les outils nécesssaires (compilateur, debuger, librairies ...) au développement de systèmes embarqués. PlatformIO s'intègre dans de nombreux IDE dont VSCode.

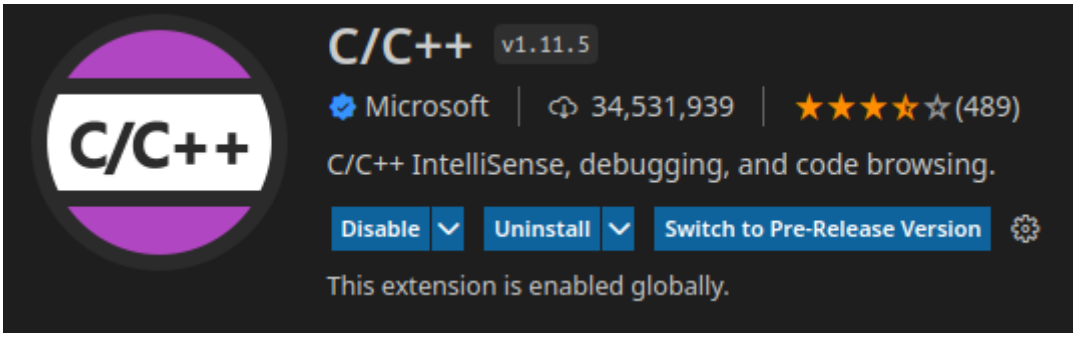
En août 2022, ce sont plus de 1 400 cartes de développement qui sont supportées avec une intégration de 12 804 librairies.

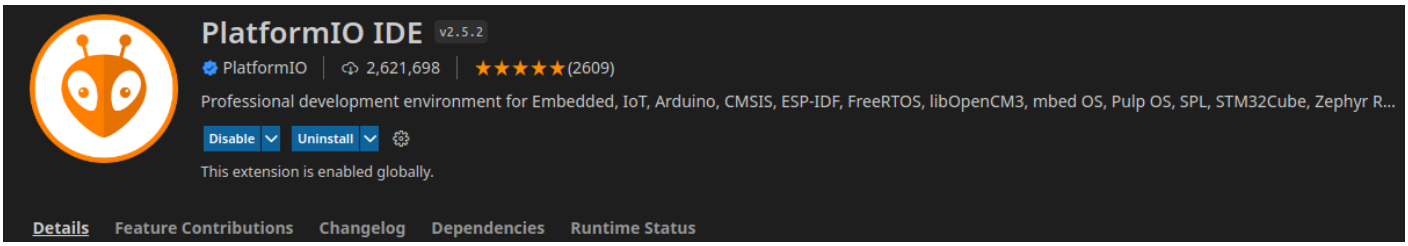


Installation des extensions dans VSCode

Pour programmer la carte de développement ESP32, j'utiliserai deux extensions VSCode:

- C/C++ IntelliSense, debugging and code browsing.
- PlatformIO IDE



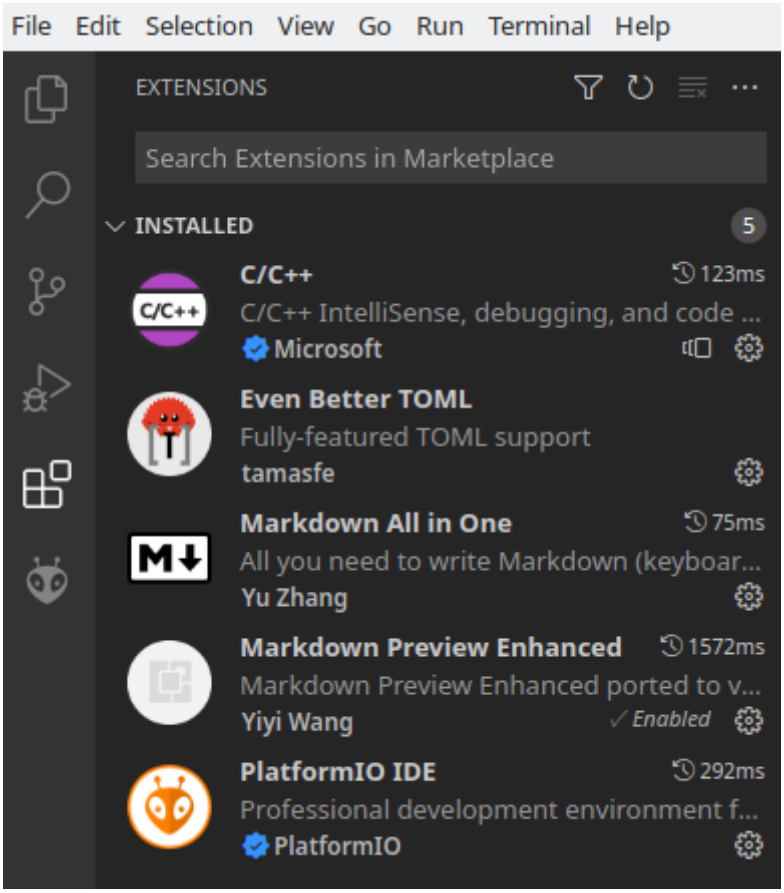


Pour accéder au menu d'installation des extensions, il y a trois méthodes :

- le raccourci : Ctrl + Shitf + X
- le menu : View -> Extensions
- Par l'icône avec les 4 petites cases à gauche, en dessous du logo Run & Debug

Pour installer le C/C++, on saisit C/C++ dans le champ Search et l'extension vous est proposée. On clique sur Install et le processus de téléchargement et d'installation se lance. Pour PlatformIO, la méthode est identique.

A l'issue de ces installations , les extensions se retrouvent sous l'onglet INSTALLED

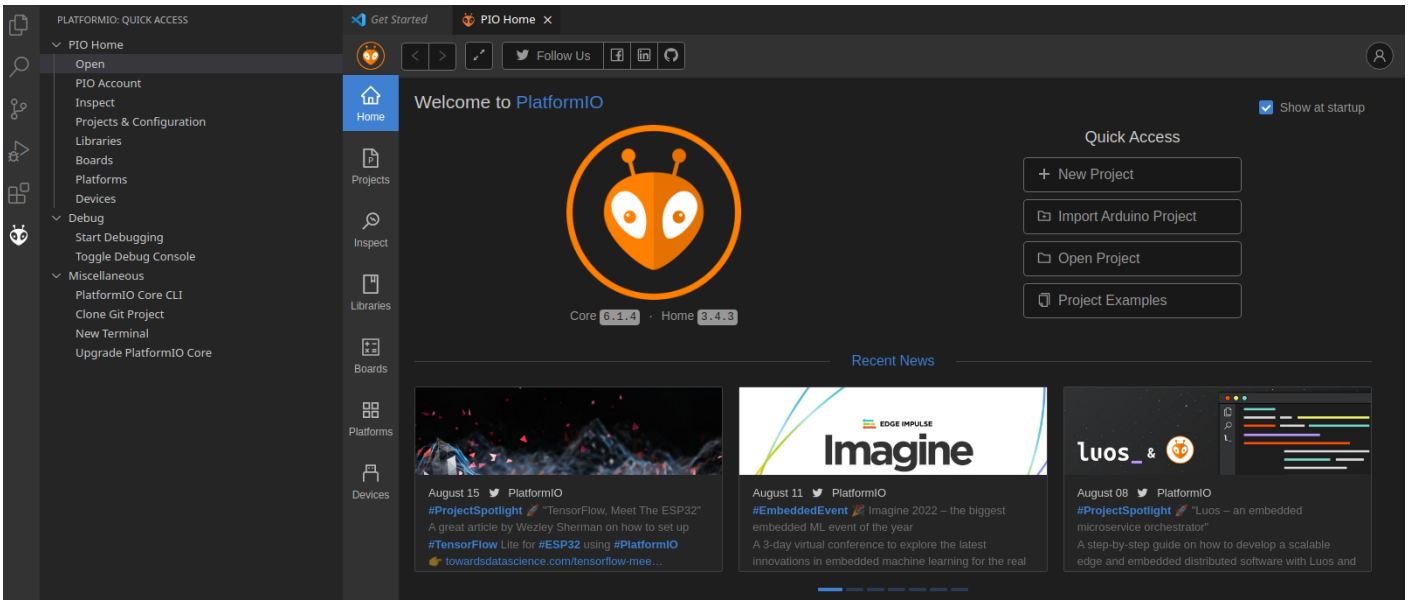


(Dans cet exemple, j'ai en plus les extensions pour le support du MarkDown et du TOML qui ne sont pas nécessaires pour l'ESP32)

Programme de Test

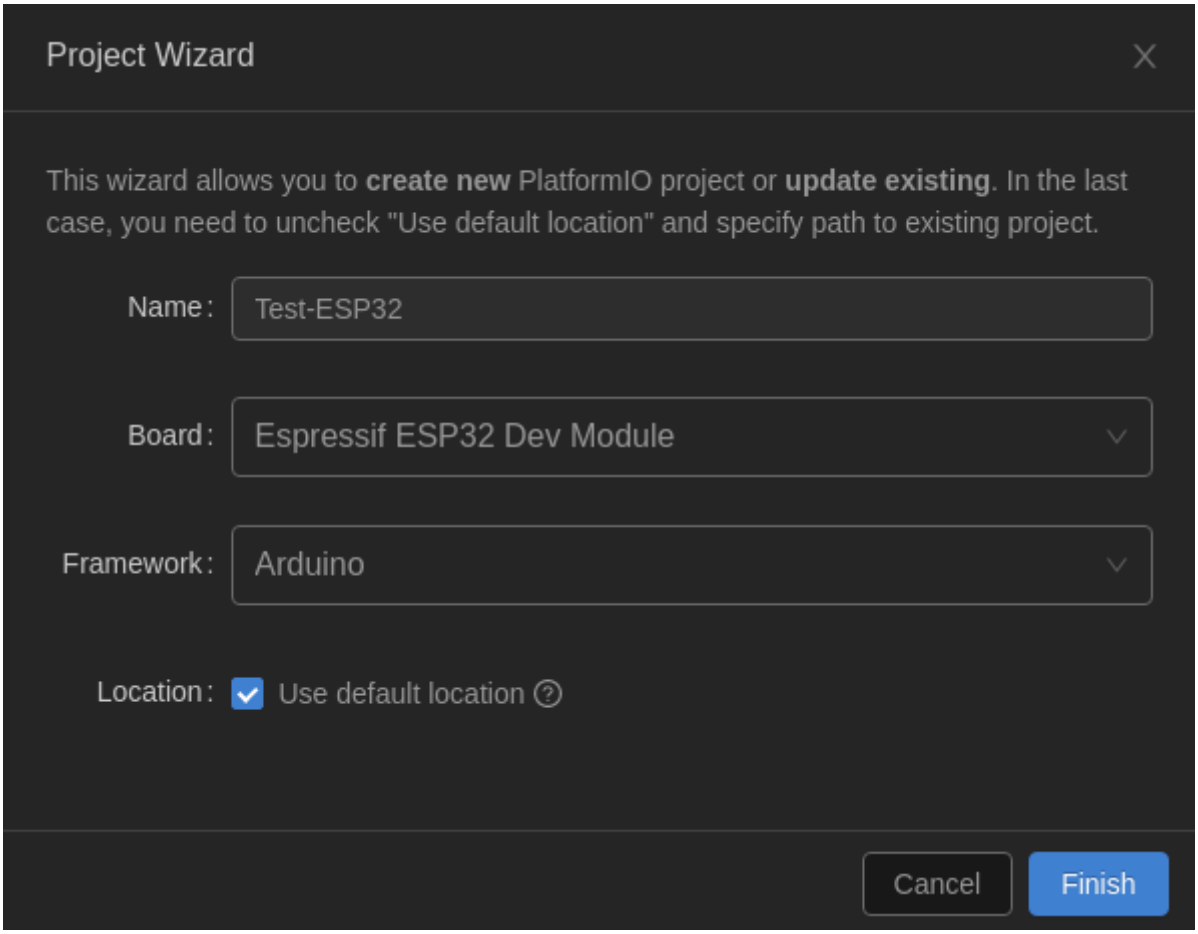
Pour créer notre premier Projet ESP32, nous passerons par le menu Home de PlatformIO et dans le Quick Access, nous ferons :

- New Project

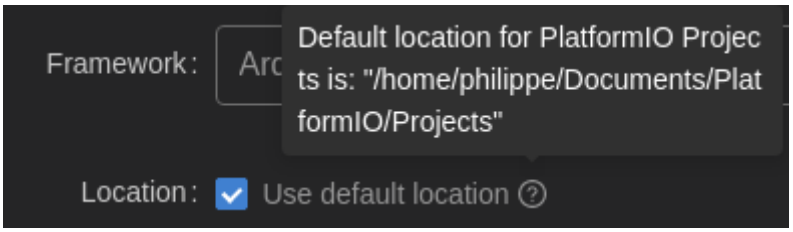


Le Project Wizard s'ouvre, pour :

- Name : Test-ESP32 (pas de caractères latin et on évite les espaces)
- Board : Espressif ESP32 Dev Module
- Framework : on choisit Arduino

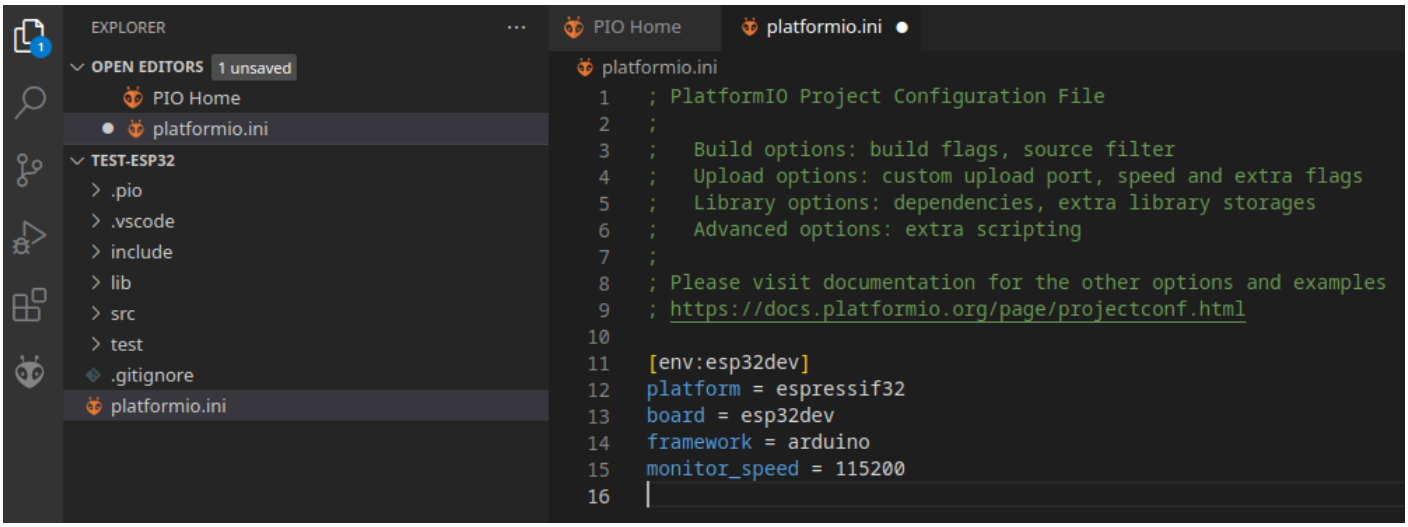


Comme on peut le constater, la localisation du Projet PlatformIO se trouve dans `/home/user/Documents/PlatformIO/Projects` L'ensemble des librairies vont également être stockés dans ce dossier, rendant le projet portable si l'on souhaite le transférer sur un autre poste.



Le fichier `platformio.ini` contient les variables d'environnement et de configuration. Dans un premier temps, on va uniquement ajouter la variable de vitesse de liaison série à 115200 afin de pouvoir transférer le code à la bonne vitesse dans la carte ESP32.

`monitor_speed = 115200`



Le fichier main.cpp se trouve dans :

- src -> main.cpp

Nous y mettrons le code suivant

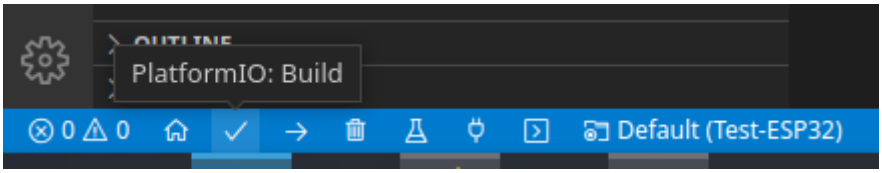
- `Serial.begin(115200);` permet d'initialiser la liaison série à 115200 bits/s
- `Serial.println("Hello World!");` permet d'afficher sur le moniteur série le message Hello World!
- `delay(1000);` est un délai d'attente de 1000 ms
- Comme en est dans une boucle infinie, le message Hello World sera répété toutes les secondes

PIO Home | platformio.ini | main.cpp

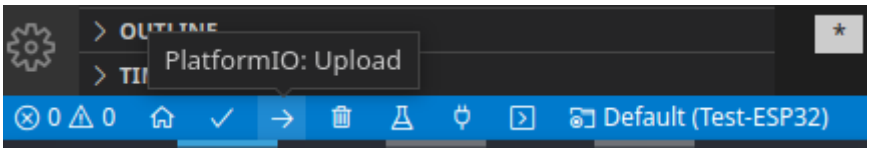
src > main.cpp > loop()

```
#include <Arduino.h>

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!");
  delay(1000);
}
```

Pour faire un Build du projet, on appui sur le petit v en bas de l'écran

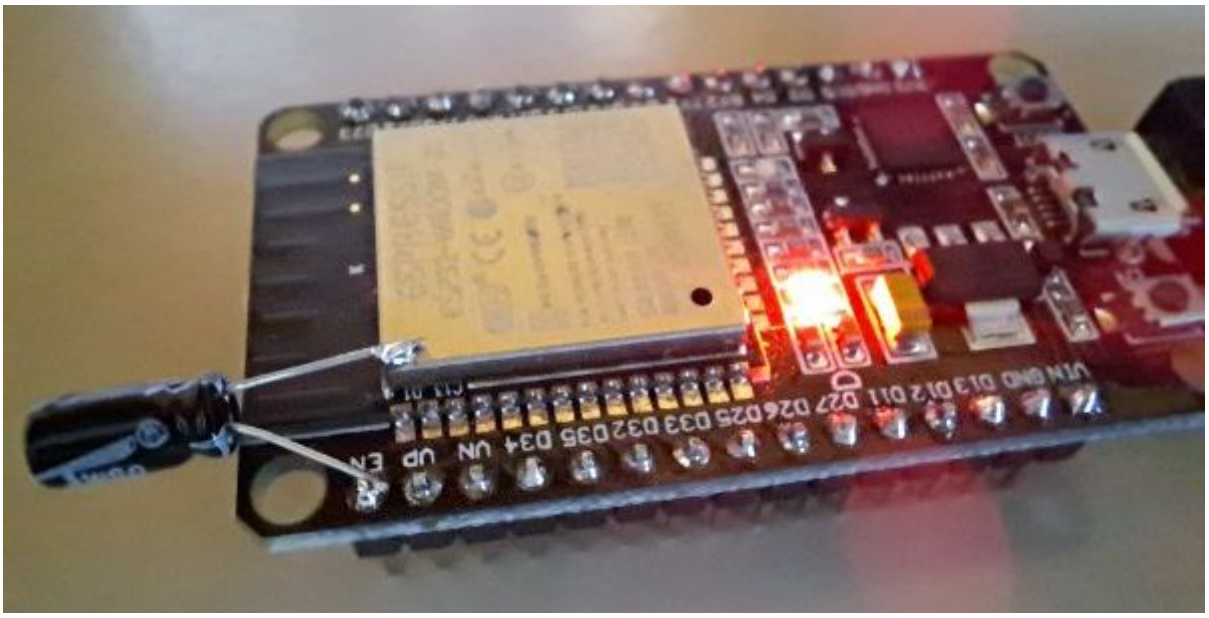


Pour transférer le code dans la carte ESP32, on appuye sur la flèche



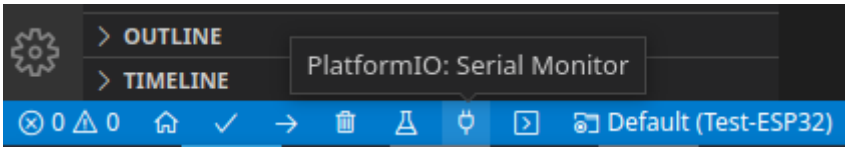
PlatformIO tente de se connecter sur /dev/ttyUSB0 pour trnaférer le code

- Si vous êtes sur Linux, il faut être dans le groupe Dialout pour avoir le droit d'écrire sur le port USB de la liaison série
  - `sudo usermod -aG dialout user` remplacer user par votre login
- Autre élement de blocage possible en fonction des PC, le transfert ne s'effectue pas et des petits points s'ajoutent après Connecting, jusqu'à générer un message d'erreur
  - Quand les petits points apparaissent, appuyer sur le bouton Boot de la carte ESP32. Si vous êtes confronté à l'obligation d'appuyer sur le bouton de Boot à chaque Upload, on peut placer une capacité électrolytique de 10 µF entre EN (+) et GND(-). Perso, j'ai soudé la broche (-) de la capacité directement sur le capot de l'ESP32.



```
Auto-detected: /dev/ttyUSB0
Uploading .pio/build/esp32dev/firmware.bin
esptool.py v3.3
Serial port /dev/ttyUSB0
Connecting.............
```

Après transfert du code, on ouvre le moniteur série en cliquant sur l'icône de prise électrique



On observe que le programme génère bien le message Hello World! toutes les secondes.

```
--- Terminal on /dev/ttyUSB0 | 115200 8-N-1
--- Available filters and text transformations: colorize, debug, default,
ntable, send_on_enter, time
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
Hello World!
Hello World!
Hello World!
```

Notre environnement de programmation de l'ESP32 avec PlatformIO est opérationnel.

Revision #3
Created 5 July 2023 14:19:13 by Philippe Celka
Updated 5 July 2023 14:29:06 by Philippe Celka