

 <h1 style="margin: 0;">PlatformIO</h1> 	
Sommaire :	
<i>I - Introduction.....</i>	<i>1</i>
<i>II - Installation sous Ubuntu avec VSCode.....</i>	<i>2</i>
<i>III - Utilisation de PlatformIO avec VsCode.....</i>	<i>3</i>
<i>IV - Installation sous Ubuntu avec Atom.....</i>	<i>6</i>

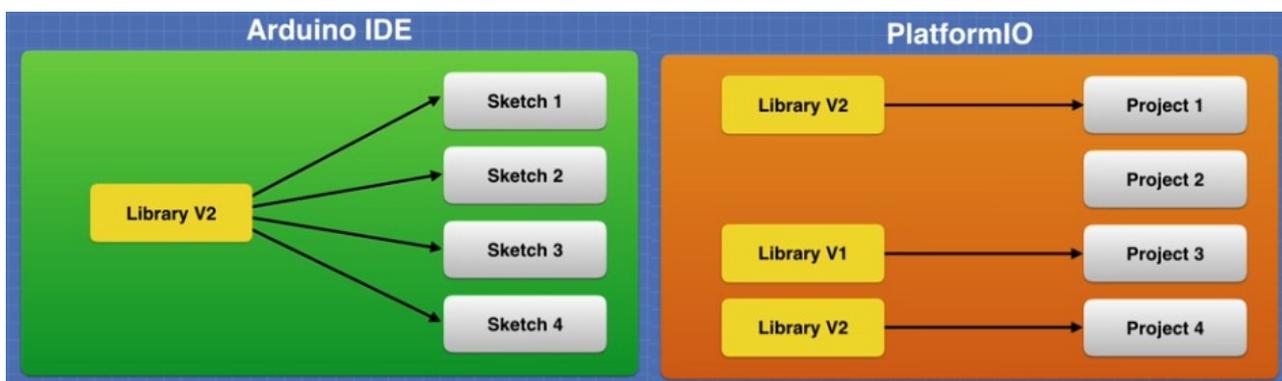
I - Introduction

PlatformIO est un écosystème *open source* dédié au développement **IoT**. **PlatformIO IDE** est l’environnement de développement C/C++ pour les systèmes embarqués. Il est multi-plateformes (Windows, Mac et GNU/Linux) et il est fourni comme une extension à **Atom** ou à **VSCode**.



PlatformIO va faciliter le développement embarqué professionnel. Il prend en charge plus de 500 cartes de développement des principaux micro-contrôleurs (Atmel, ESP8266 et ESP32, STM32, etc ...) et regroupe plus de 5000 bibliothèques.

L’avantage principal d’utiliser **PlatformIO** par rapport à l’**IDE Arduino** est que l’on va créer un projet pour chaque application et y associer des librairies et des fichiers. Cela n’aura aucune incidence sur les autres projets, on pourra utiliser par exemple des versions de librairies différentes pour chaque projet (<https://www.youtube.com/watch?v=JmvMvIphMnY>).



II - Installation sous Ubuntu avec VSCode

1. Installation de VSCode

```
apt install python3-venv
Si nécessaire : apt install snapd
snap install code --classic
python3 -version
```

Télécharger le paquet **vsc** depuis <https://code.visualstudio.com/> puis l'installer à l'aide de la commande :

```
dpkg -i code_1.59.1-1629375198_amd64.deb
```

2. Installation du plugin PlatformIO pour VSCode

<https://docs.platformio.org/en/latest/integration/ide/vscode.html>.

Lancer **VSCode** puis aller dans le menu en bas à gauche **Extensions**, rechercher et installer **PlatformIO IDE**. Puis relancer **VSCode**. L'icône de **PlatformIO IDE** apparaît dans le menu en bas à gauche.



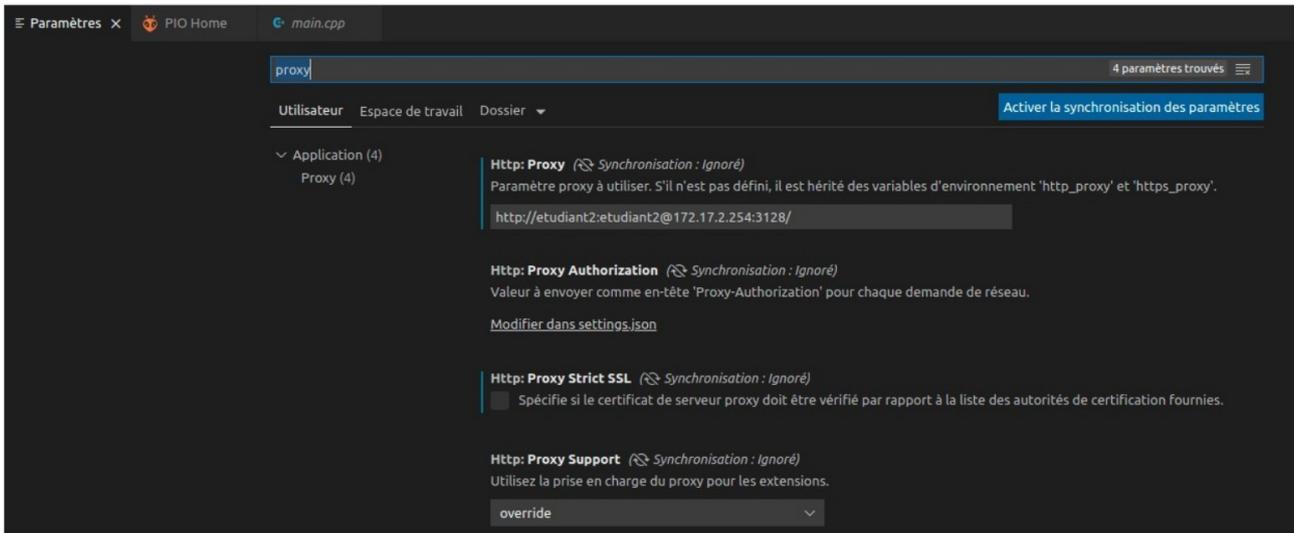
On accède à la fabrication (*build*), au téléversement (*upload*) et au moniteur série par la barre bleue en bas de l'IDE :

```
main.cpp - Sans titre(Espace de travail) - Visual Studio Code
Fichier Edition Sélection Affichage Atteindre Exécuter Terminal Aide
EXPLORATEUR
SANS TITRE(ESPACE DE TRAVAIL)
  python
  ESP32-Blink
    .pio
    .vscode
    include
    lib
    src
      main.cpp
    test
    .gitignore
    platformio.ini
  STRUCTURE
  AZURE IOT HUB

ESP32-Blink > src > main.cpp > ...
1 #include <Arduino.h>
2
3 #define LED_BUILTIN 2
4
5 // the setup function runs once when you press reset or power the board
6 void setup() {
7   // initialize digital pin LED_BUILTIN as an output.
8   pinMode(LED_BUILTIN, OUTPUT);
9   Serial.begin(9600);
10 }
11
12 // the loop function runs over and over again forever
13 void loop() {
14   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
15   Serial.println("Mise à 1");
16   delay(1000);
17   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
18   Serial.println("Mise à 0");
19   delay(1000);
20 }

PROBLEMS OUTPUT TERMINAL CONSOLE DE DÉBOGAGE PlatformIO: Monitor (ESP32-Blink) - ESP32-Blink
> Executing task in folder ESP32-Blink: platformio device monitor <
--- Available filters and text transformations: colorize, debug, default, direct, esp32_exception_decoder,
hexlify, log2file, nocontrol, printable, send on enter, time
--- More details at https://bit.ly/pio-monitor-filters
--- Miniterm on /dev/ttyUSB0 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
```

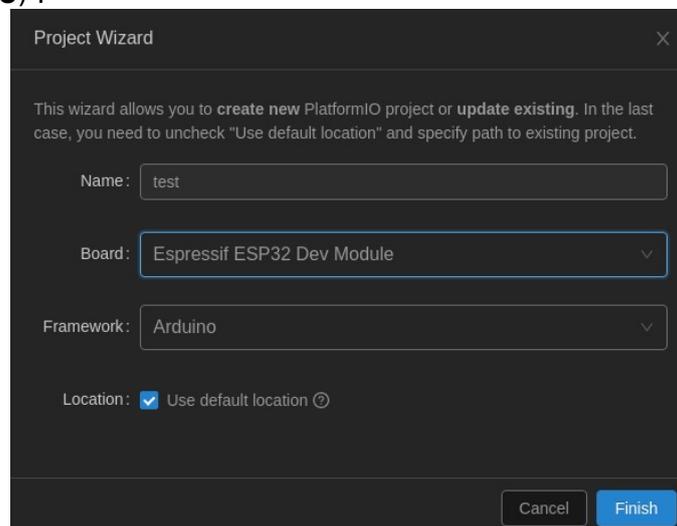
Remarque : Si on utilise **VS Code** derrière un **proxy**, il faut le configurer de la façon suivante en spécifiant le paramètre **Http:Proxy** et en désactivant **Http:Proxy Strict SSL** :



III - Utilisation de PlatformIO avec VsCode

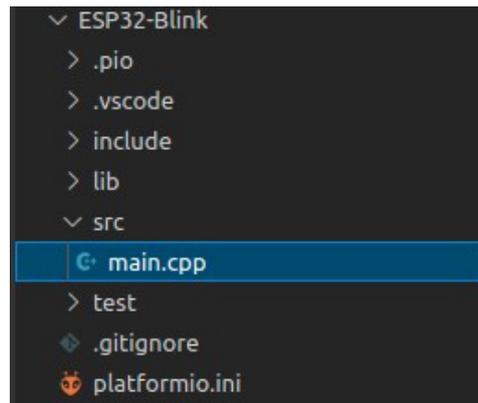
https://docs.platformio.org/en/latest/tutorials/espressif32/arduino_debugging_unit_testing.html.

Pour tester l'environnement, on commence par créer un nouveau projet. On donne un nom au projet et on choisit sa carte de développement (ici une carte **Espressif ESP32 Dev Module** et le Framework **Arduino**) :



Remarque : En fonction de la carte choisie, il peut y avoir plusieurs plateformes de développement.

L'architecture d'un projet **PlatformIO** est la suivante :



Le code source de l'application se situe dans le dossier **src** avec le fichier **main.cpp** suivant (par défaut pour un framework **Arduino**) :

```
#include <Arduino.h>
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

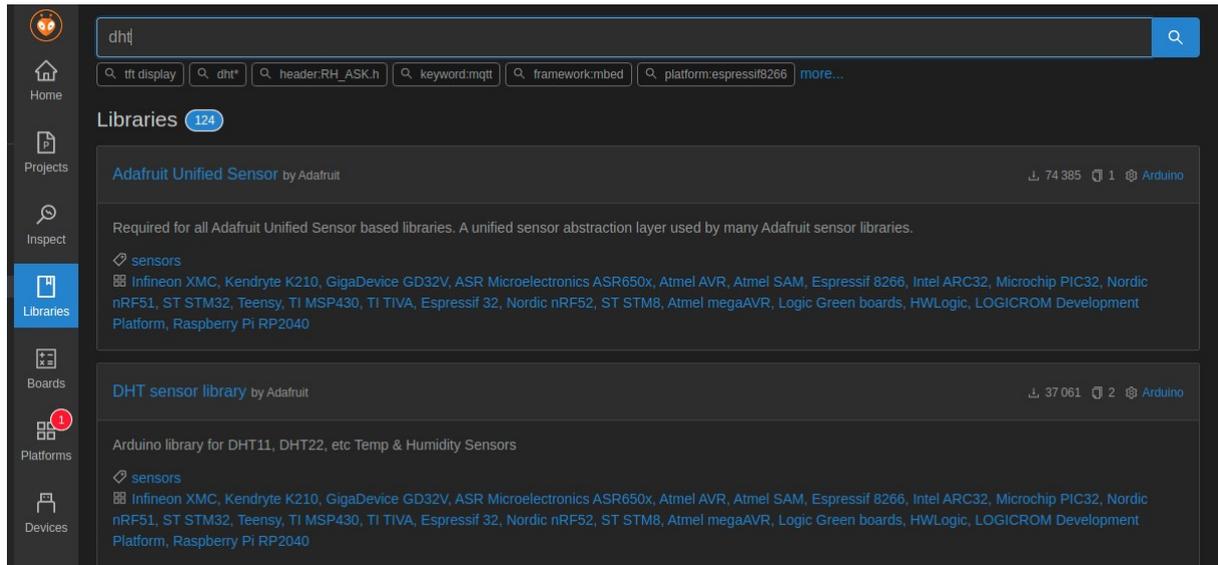
Il est possible d'ajouter ses fichiers d'en-tête (*header*) **.h** dans le dossier **include** et ses bibliothèques dans le dossier **lib**. Tout ceci sera automatiquement pris en charge par **PlatformIO**.

Le fichier **platformio.ini** contient la configuration du projet (les options du fichier platformio.ini sont décrites à l'URL <https://docs.platformio.org/en/latest/projectconf.html>) :

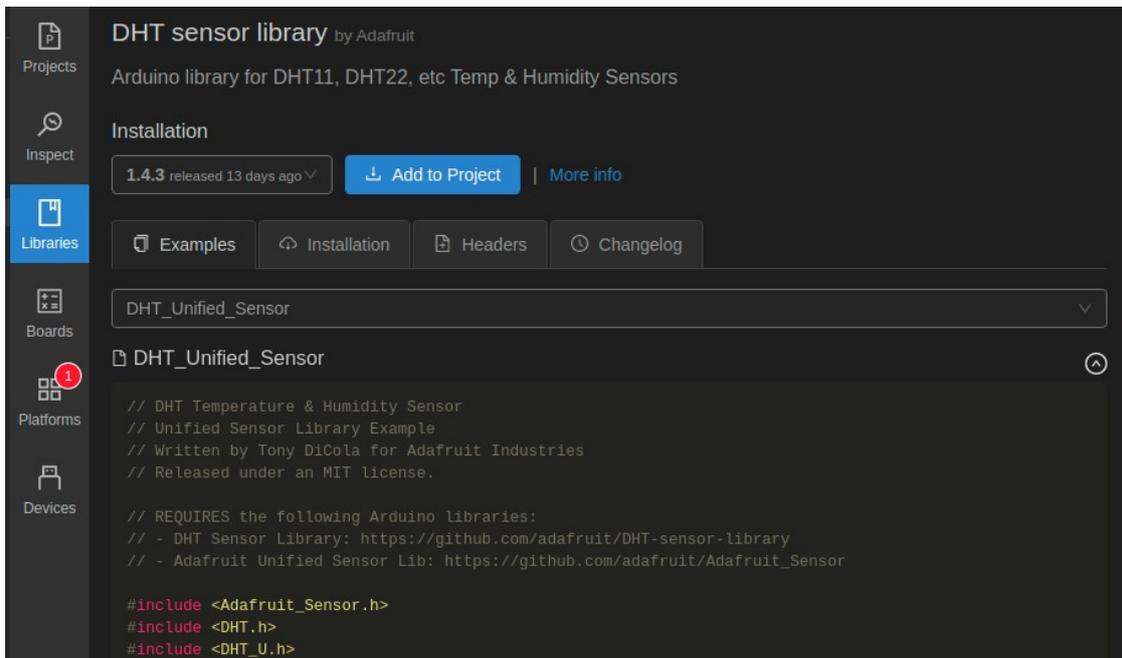
```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
```

Pour ajouter des bibliothèques de la communauté au projet, il suffit d'ajouter leur nom (ou leur ID) dans la variable **lib_deps**.

Comme le montre la figure ci-dessous, **PlatformIO** fournit un outil de recherche de bibliothèques bien pratique :



On choisit sa bibliothèque et on accède à l'ensemble de ces informations :



PlatformIO facilite le développement car il est possible maintenant de fabriquer son application et la téléverser sur sa carte sans rien faire de plus !

- Fabriquer (*build*) :



- Téléverser (*upload*) :



- Ouvrir le moniteur série (*serial monitor*) :



Après avoir téléversé le programme, on obtient :

```

--- More details at https://bit.ly/pi0-monitor-filters
--- Miniterm on /dev/ttyUSB0 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Mise à 0
Mise à 1
Mise à 0
Mise à 1
Mise à 0

```

Pour le port série les options disponibles sont :

```

$ platformio device monitor --help
Usage: platformio device monitor [OPTIONS]

Options:
  -p, --port TEXT      Port, a number or a device name
  -b, --baud INTEGER   Set baud rate, default=9600
  --parity [N|E|O|S|M] Set parity, default=N
  --rtscts             Enable RTS/CTS flow control, default=Off
  --xonxoff           Enable software flow control, default=Off
  --rts [0|1]         Set initial RTS line state, default=0
  --dtr [0|1]         Set initial DTR line state, default=0
  --echo              Enable local echo, default=Off
  --encoding TEXT     Set the encoding for the serial port (e.g. hexlify,
                    Latin1, UTF-8), default: UTF-8
  -f, --filter TEXT   Add filters / text transformation
  --eol [CR|LF|CRLF] End of line mode, default=CRLF
  --raw              Do not apply any encodings/transformations
  --exit-char INTEGER ASCII code of special character that is used to exit
                    the application, default=29 (DEC)
  --menu-char INTEGER ASCII code of special character that is used to
                    control miniterm (menu), default=20 (DEC)
  --quiet            Diagnostics: suppress non-error messages, default=Off
  -h, --help         Show this message and exit.

```

Par exemple pour modifier la vitesse à la valeur 115200 :

```
$ platformio device monitor -b 115200
```

IV - Installation sous Ubuntu avec Atom

1. Installation de Atom

Il suffit de récupérer le paquet à l'adresse <https://atom.io/download/deb> puis de l'installer à l'aide des commandes :

```

dpkg -i atom-amd64.deb
apt-get -f install // pour installer les dépendances si nécessaire

```

2. Installation du paquet PlatformIO pour Atom

Une fois **Atom** installé, il faut installer le package **PlatformIO**. Ouvrir le Menu **Edit/Preferences/Install**.

Dans le champ de recherche, saisir **platformio-ide**. Il y a 3 packages disponibles :

- platformio-ide
- platformio-ide-terminal
- platformio-ide-debugger

Installer les packages **ide** et **ide-terminal** en cliquant sur **Install**. L'installation est silencieuse mais vous pouvez la suivre en survolant la roue crantée dans le coin inférieur droit de l'écran. L'installation peut durer quelques minutes en fonction de votre débit internet. Rester sur cette page jusqu'à ce que le package soit marqué comme installé.

Remarque : Si on utilise **Atom** derrière un **proxy**, il faut le configurer en utilisant la commande **apm config** ou en créant le fichier `~/.atom/.apmrc` contenant :

```
strict-ssl=false  
http-proxy=http://etudiant2:etudiant2@172.17.2.254:3128  
https-proxy=http://etudiant2:etudiant2@172.17.2.254:3128
```

Avec la commande **apm config** cela donne :

```
apm config set strict-ssl false  
apm config set http-proxy http://etudiant2:etudiant2@172.17.2.254:3128  
apm config set https-proxy http://etudiant2:etudiant2@172.17.2.254:3128
```

Et enfin on lance **atom** à l'aide de la commande suivante :

```
HTTP_PROXY="http://etudiant2:etudiant2@172.17.2.254:3128"  
HTTPS_PROXY="http://etudiant2:etudiant2@172.17.2.254:3128" atom --proxy-  
server="http://etudiant2:etudiant2@172.17.2.254:3128"
```

Lorsque l'installation est terminée, un nouvel onglet **PlatformIO Home** est ajouté à **Atom**. Cet onglet peut s'ouvrir automatiquement au démarrage d'Atom et vous permettra de gérer vos projets et mettre à jour le package et le Core de PlatformIO. Si une nouvelle version est disponible, en profiter pour mettre à jour avant de continuer en cliquant sur **Upgrade NOW!**.

Un nouveau menu est également ajouté à la barre de menu d'**Atom**. Si vous fermez l'écran d'accueil, pas de panique, il est possible de le ré-ouvrir depuis le menu Home/Screen.

Le menu contient les options suivantes :

- **Home Screen** : ouvrir la page d'accueil
- **Projects Example** : créé un nouveau projet à partir des exemples pour les plateformes prisent en charge
- **Initialize or Update Project** : permet d'ajouter une plateforme à un projet existant. Les ressources nécessaires sont téléchargées et installées
- **Import Arduino IDE Project** : importe un projet Arduino existant
- **Open Project folder** : ouvre le répertoire des projets dans le gestionnaire de fichier (ou le Finder sur macOS)
- **Build** : compile le code avant téléversement
- **Upload** : téléverse le code sur le micro-contrôleur
- **Clean** : supprime le dossier masqué .pioenv qui contient toutes les versions compilées.
- **Test** : test le code pour toutes les plateformes cibles. Fonction non disponible pour la version Community gratuite.
- **Run other target** : permet de choisir la cible. PlatformIO détecte seul la carte connectée et téléverse le code correspondant. Dans ce cas, on peut choisir la carte (par exemple si plusieurs cartes sont connectées à la machine), ou envoyer des fichiers dans la zone mémoire SPIFFS (contenu du dossier Data)

- **Toggle Build Panel** : affiche ou masque le Terminal
- **Terminal** : ouvre le Terminal sur macOS, Linux, Raspberry et Power Shell sur Windows 10
- **Library Manager** : une page d'information qui explique comment gérer les bibliothèques. Tout se fait en ligne de commande (plus loin)
- **Serial Monitor** : ouvre le moniteur série. Sur Windows, une session Power Shell est lancée.
- **List Serial Ports** : liste les périphériques branchés en USB pouvant communiquer via le port série
- **Update** : vous pouvez directement mettre à jour le Core de PlatformIO, les packages et les bibliothèques utilisées
- **Settings** : un raccourci pour accéder au réglage d'Atom et des packages

Toutes ces fonctions prennent également place sous la forme d'une barre d'icône latérale. La barre d'icône peut être placée en haut, bas, gauche ou en bas.

PlatformIO utilise **Clang** pour l'auto-complétion des commandes. Cette installation est réalisée à l'aide de la commande **apt-get install clang**.