

# Installation de PlatformIO pour Netbeans

## Table des matières

Installation de PlatformIO pour Netbeans.....	1
1 Introduction.....	1
2 Installation des prérequis : la commande python.....	2
3 Installation de platformIO-Core.....	3
4 installation des liens symboliques.....	4
5 Installation des règles udev.....	5
6 Création d'un projet Netbeans pour esp32.....	6
7 Ajouter des bibliothèques personnelles.....	10
8 Ajouter des bibliothèques externes à votre projet.....	11
9 Ajouter une bibliothèque globale à tous les projets.....	12
10 Lister les bibliothèques installées.....	13
11 Code Assistance.....	14
12 Tests du programme.....	16
13 Téléverser des fichiers SPIFFS dans la mémoire flash.....	16
14 Ajouter des commandes externes à Netbeans.....	19
2 Configuration des commandes externes.....	20
3 Téléverser les fichiers data avec Commande 1.....	21
15 Mettre à jour platformio.....	22
1 Mise à jour de la la plateforme espressif32.....	22
2 Mise à jour des librairies.....	22
3 Mise à jour de platformIO.....	23
16 Création d'un projet pour la carte Wemos D1 R1.....	23

## 1 Introduction

Le principal problème qui repousse les gens du monde embarqué est le processus compliqué pour configurer un logiciel de développement pour une carte spécifique avec ses chaînes d'outils.

PlatformIO est un outil professionnel multi plateforme, et multi architecture pour les ingénieurs sur systèmes embarqués et pour les développeurs de logiciels qui écrivent des applications pour ces systèmes.

Comment ça marche?

Sans entrer trop profondément dans les détails de la mise en œuvre de PlatformIO, le cycle de travail du projet développé à l'aide de cet outil est le suivant :

- L'utilisateur choisit l'IDE (pour nous ce sera Netbeans) et la carte cible (lolin32 ou ttgo)
- PlatformIO télécharge les chaînes d'outils requises et les installe automatiquement. Il crée aussi l'architecture du projet.
- L'utilisateur ouvre le projet créé et développe le code.
- PlatformIO assure la compilation, et télécharge le firmware vers la carte cible.

## 2 Installation des prérequis : la commande python

Comme le montre la capture ci-dessus la commande python n'est pas reconnue par défaut.

```
philippe@portable:~$ python --version  
La commande « python » n'a pas été trouvée, voulez-vous dire :  
  commande « python3 » du deb python3  
  commande « python » du deb python-is-python3
```

Installation du dépôt python-is-python3 (ce dépôt n'est pas disponible pour les versions non récentes de Linux)

```
sudo apt-get install python-is-python3  
philippe@portable:~$ python --version  
Python 3.8.10
```

Comme le montre la capture d'écran ci-dessus la version pour python est Python 3.8.10

Installation de packages d'environnements virtuels pour python.

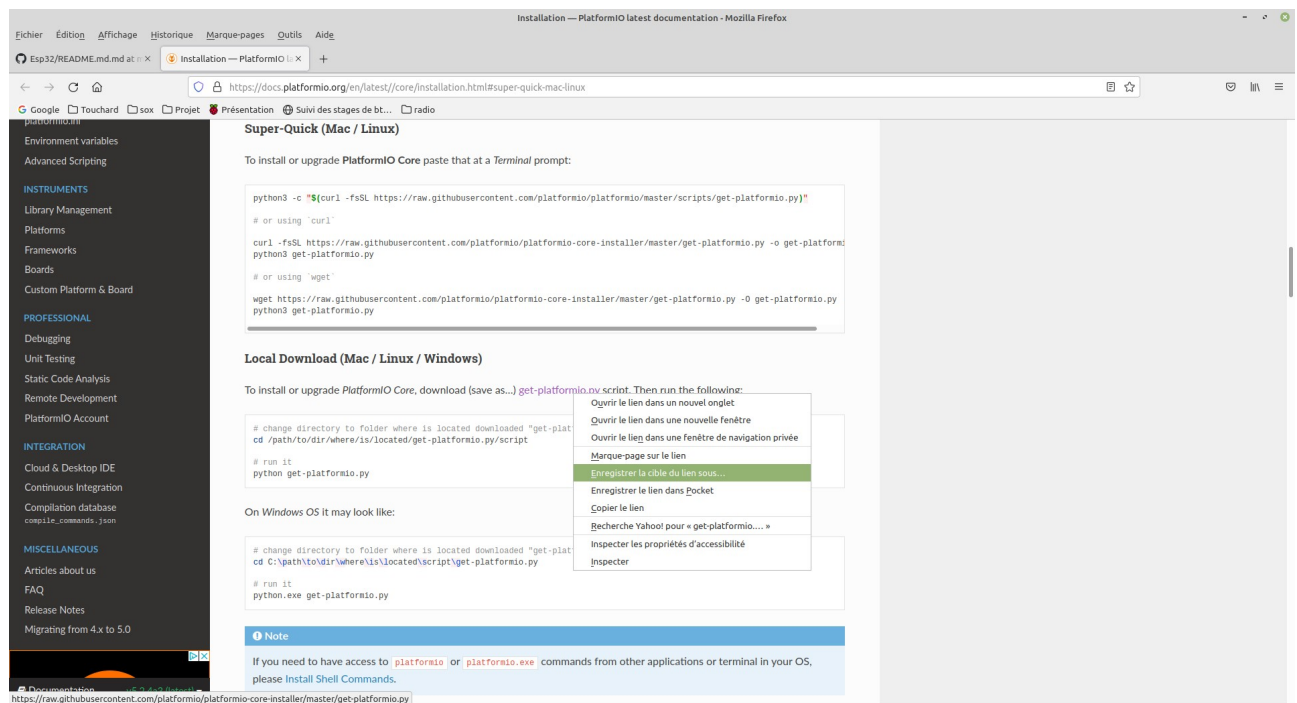
```
sudo apt-get install python3-venv
```

### 3 Installation de platformIO-Core

Pour installer ou mettre à niveau PlatformIO-Core, **téléchargez** le script **get-platformio.py**.

Le script est disponible sur la page suivante. (Clic droit sur le lien - Enregistrer la cible du lien sous...)

<https://docs.platformio.org/en/latest/core/installation.html#super-quick-mac-linux>



Ensuite, exécutez la commande suivante : en tant que root

```
PLATFORMIO_CORE_DIR=/opt/platformio python3 get-platformio.py
```

résultat

```
PlatformIO Core has been successfully installed into an isolated  
environment `/opt/platformio/penv`!
```

```
The full path to `platformio.exe` is  
`/opt/platformio/penv/bin/platformio`
```

```
If you need an access to `platformio.exe` from other applications,  
please install Shell Commands
```

```
(add PlatformIO Core binary directory `/opt/platformio/penv/bin` to  
the system environment PATH variable):
```

```
See https://docs.platformio.org/page/installation.html#install-  
shell-commands
```

## 4 installation des liens symboliques

```
cd /usr/local/bin  
sudo ln -s /opt/platformio/penv/bin/pio /usr/local/bin/pio  
sudo ln -s /opt/platformio/penv/bin/platformio  
/usr/local/bin/platformio  
sudo ln -s /opt/platformio/penv/bin/piodebuggdb  
/usr/local/bin/piodebuggdb
```

Vérification de la prise en compte

```
philippe@philippe:~$ pio --version  
PlatformIO Core, version 6.0.2
```

Remarque : **pio** est un *alias* de la commande **platformio**.

*La version affichée est celle disponible courant Juillet 2022*

## 5 Installation des règles udev

*Remarque* : attention aux droits d'accès au périphérique USB. Le plus simple étant de copier le fichier de règles udev

Les utilisateurs de Linux doivent donc installer des règles udev pour les cartes/périphériques pris en charge par PlatformIO.

```
curl -fsSL https://raw.githubusercontent.com/platformio/platformio-core/master/scripts/99-platformio-udev.rules | sudo tee /etc/udev/rules.d/99-platformio-udev.rules
```

redémarrer le service udev

```
sudo service udev restart
```

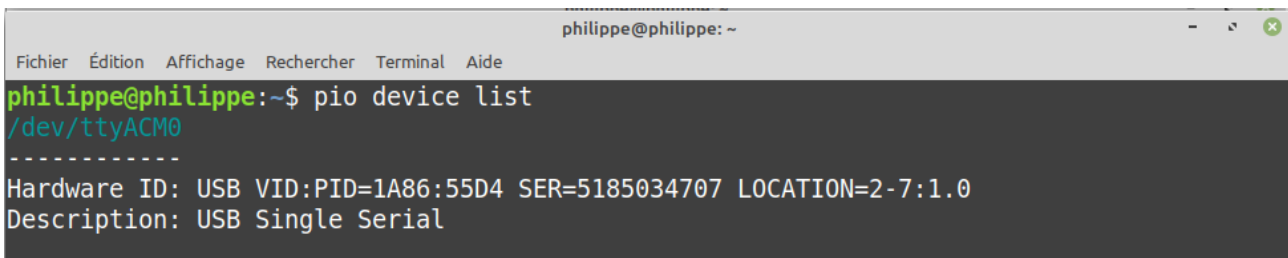
Brancher un esp32 et lister les ports série disponibles avec la commande **pio device list**



```
philippe@philippe: ~  
Fichier  Édition  Affichage  Rechercher  Terminal  Aide  
philippe@philippe:~$ pio device list  
/dev/ttyUSB0  
-----  
Hardware ID: USB VID:PID=10C4:EA60 SER=0001 LOCATION=2-7  
Description: CP2102 USB to UART Bridge Controller - CP2102 USB to UART Bridge Controller
```

L'écran ci-dessus montre qu'un esp32 est connecté sur **ttyUSB0**

Même commande avec un esp32 TTGO



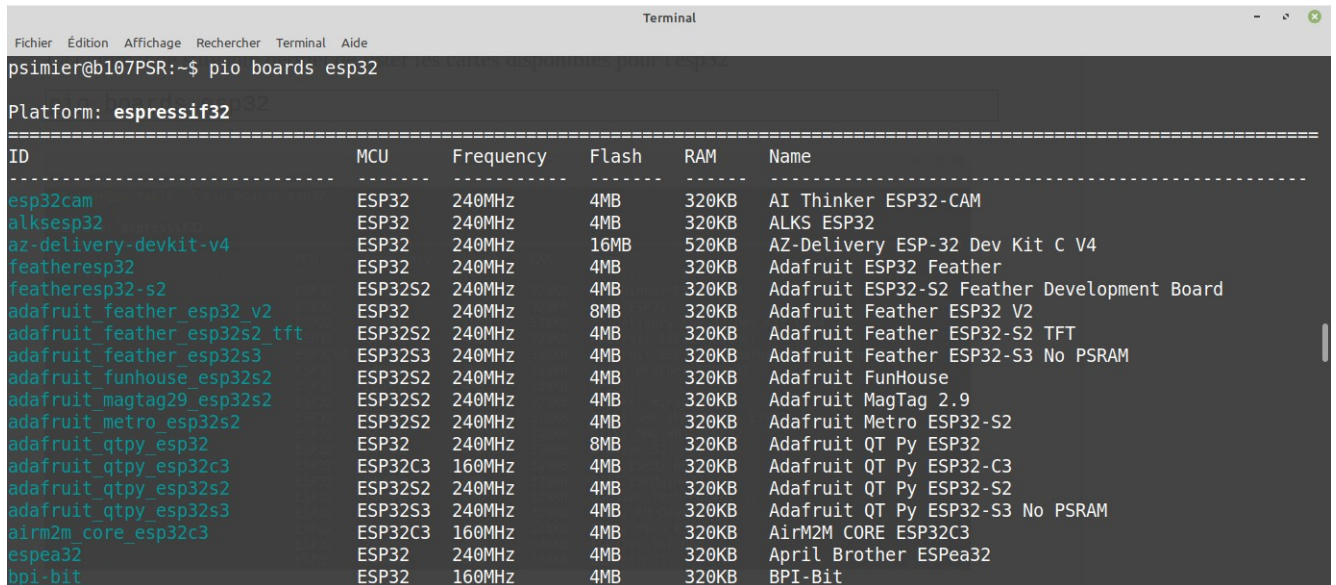
```
philippe@philippe: ~  
Fichier  Édition  Affichage  Rechercher  Terminal  Aide  
philippe@philippe:~$ pio device list  
/dev/ttyACM0  
-----  
Hardware ID: USB VID:PID=1A86:55D4 SER=5185034707 LOCATION=2-7:1.0  
Description: USB Single Serial
```

L'écran montre qu'une carte TTGO est connectée sur **ttyACM0**

## 6 Création d'un projet Netbeans pour esp32

La commande suivante permet de lister les cartes disponibles pour l'esp32

```
pio boards esp32
```



ID	MCU	Frequency	Flash	RAM	Name
esp32cam	ESP32	240MHz	4MB	320KB	AI Thinker ESP32-CAM
alksesp32	ESP32	240MHz	4MB	320KB	ALKS ESP32
az-delivery-devkit-v4	ESP32	240MHz	16MB	520KB	AZ-Delivery ESP-32 Dev Kit C V4
featheresp32	ESP32	240MHz	4MB	320KB	Adafruit ESP32 Feather
featheresp32-s2	ESP32S2	240MHz	4MB	320KB	Adafruit ESP32-S2 Feather Development Board
adafruit_feather_esp32_v2	ESP32	240MHz	8MB	320KB	Adafruit Feather ESP32 V2
adafruit_feather_esp32s2_tft	ESP32S2	240MHz	4MB	320KB	Adafruit Feather ESP32-S2 TFT
adafruit_feather_esp32s3	ESP32S3	240MHz	4MB	320KB	Adafruit Feather ESP32-S3 No PSRAM
adafruit_funhouse_esp32s2	ESP32S2	240MHz	4MB	320KB	Adafruit FunHouse
adafruit_magtag29_esp32s2	ESP32S2	240MHz	4MB	320KB	Adafruit MagTag 2.9
adafruit_metro_esp32s2	ESP32S2	240MHz	4MB	320KB	Adafruit Metro ESP32-S2
adafruit_qtpy_esp32	ESP32	240MHz	8MB	320KB	Adafruit QT Py ESP32
adafruit_qtpy_esp32c3	ESP32C3	160MHz	4MB	320KB	Adafruit QT Py ESP32-C3
adafruit_qtpy_esp32s2	ESP32S2	240MHz	4MB	320KB	Adafruit QT Py ESP32-S2
adafruit_qtpy_esp32s3	ESP32S3	240MHz	4MB	320KB	Adafruit QT Py ESP32-S3 No PSRAM
airm2m_core_esp32c3	ESP32C3	160MHz	4MB	320KB	AirM2M CORE ESP32C3
espea32	ESP32	240MHz	4MB	320KB	April Brother ESPea32
bpi-bit	ESP32	160MHz	4MB	320KB	BPI-Bit

La première colonne donne l'ID utilisé pour chaque carte référencée.

Au lycée nous avons des cartes **ttgo-t1** et **esp32cam** pour le projet ballon stratosphérique et des cartes **lolin32** pour le projet ruche et les travaux pratiques.

### Création d'un projet pour IDE Netbeans

Créer un répertoire pour votre projet

puis se déplacer à l'intérieur puis lancer la commande **pio project init** avec comme arguments l'IDE et la carte utilisée.

```
mkdir test_esp32
cd test_esp32
pio project init --ide netbeans --board lolin32
```

```
philippe@portable: ~/NetBeansProjects/test_esp32
Fichier Édition Affichage Rechercher Terminal Aide
philippe@portable:~/NetBeansProjects/test_esp32$ pio project init --ide netbeans --board lolin32

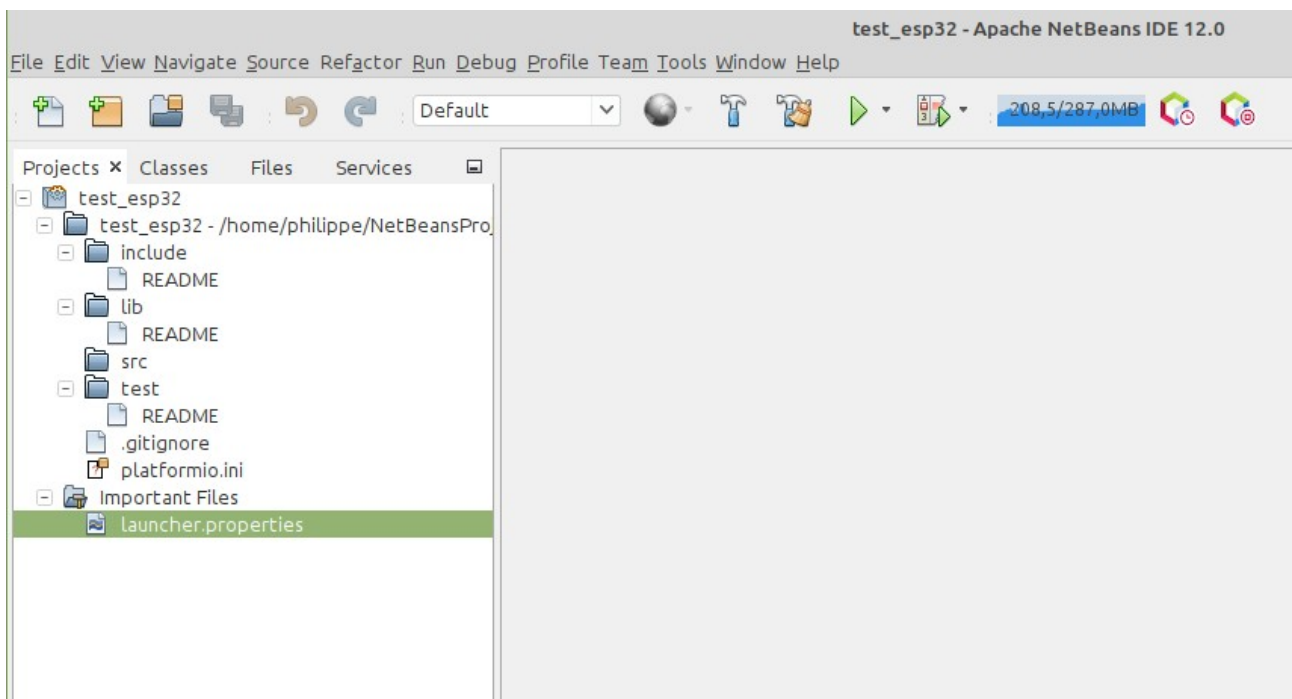
The current working directory /home/philippe/NetBeansProjects/test_esp32 will be used for the project.

The next files/directories have been created in /home/philippe/NetBeansProjects/test_esp32
include - Put project header files here
lib - Put here project specific (private) libraries
src - Put project source files here
platformio.ini - Project Configuration File
Platform Manager: Installing espressif32
Downloading [#####] 100%
Unpacking [#####] 100%
Platform Manager: espressif32 @ 3.3.2 has been installed!
Tool Manager: Installing platformio/toolchain-xtensa32 @ ~2.50200.0
Downloading [#####] 100%
Unpacking [#####] 100%
Tool Manager: toolchain-xtensa32 @ 2.50200.97 has been installed!
Tool Manager: Installing platformio/tool-esptoolpy @ ~1.30100.0
Downloading [#####] 100%
Unpacking [#####] 100%
Tool Manager: tool-esptoolpy @ 1.30100.210531 has been installed!
The platform 'espressif32' has been successfully installed!
The rest of the packages will be installed later depending on your build environment.

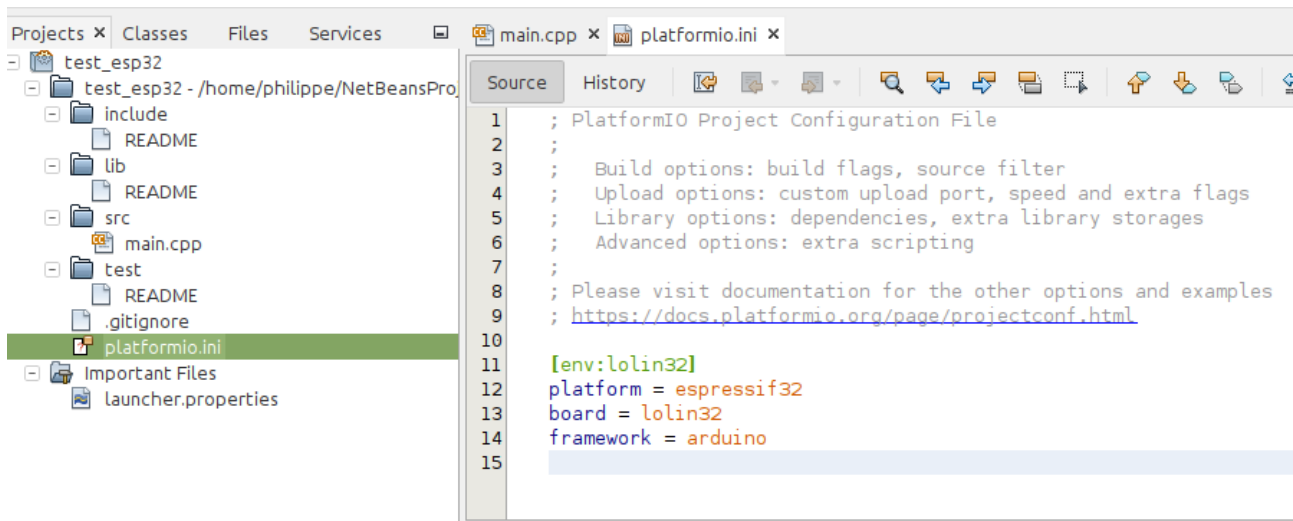
Project has been successfully initialized including configuration files for 'netbeans' IDE.
philippe@portable:~/NetBeansProjects/test_esp32$
```

La "tool chaîne" est maintenant installée.

On peut ouvrir ce projet via le Menu: File > Open Project... de Netbeans



Le fichier **platformio.ini** permet de définir le framework utilisé.

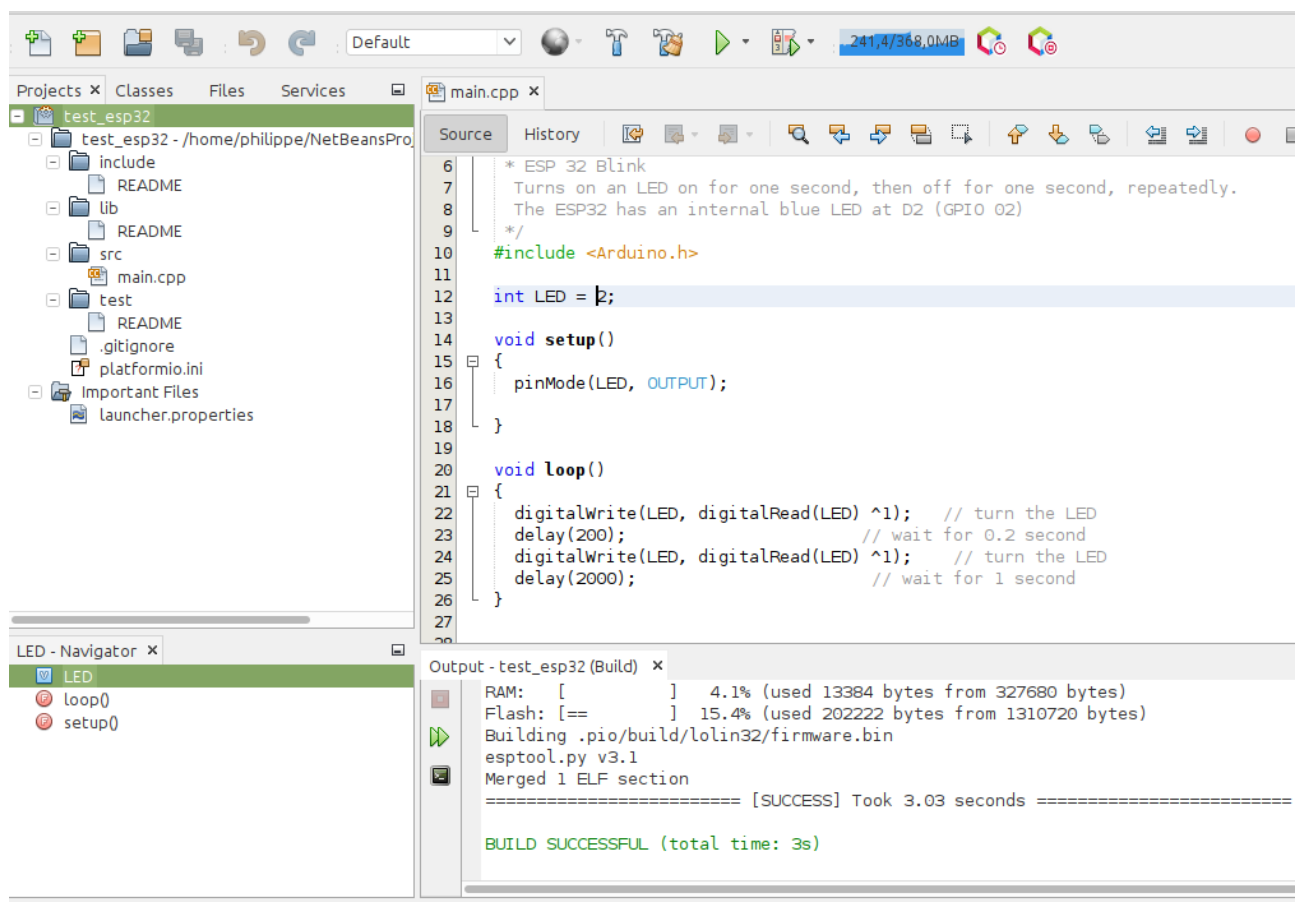


Avec ESP32 deux frameworks sont disponibles **Arduino** et **espidf**

avec le framework Arduino, il faudra juste inclure le fichier d'en-tête suivant :

```
#include <Arduino.h>
```

Ajoutez de nouveaux fichiers au répertoire src (\*.c, \*.cpp, \*.ino, etc.) via un clic droit sur le dossier src dans le volet "Projects"

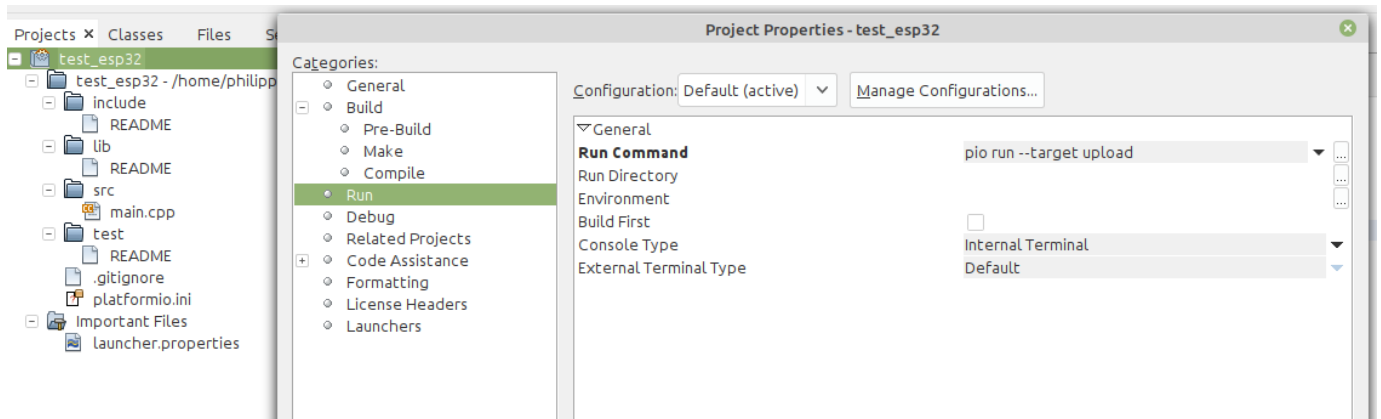




Construire le projet à l'aide du menu : Run > Build Project ou cliquer sur le marteau ou F11

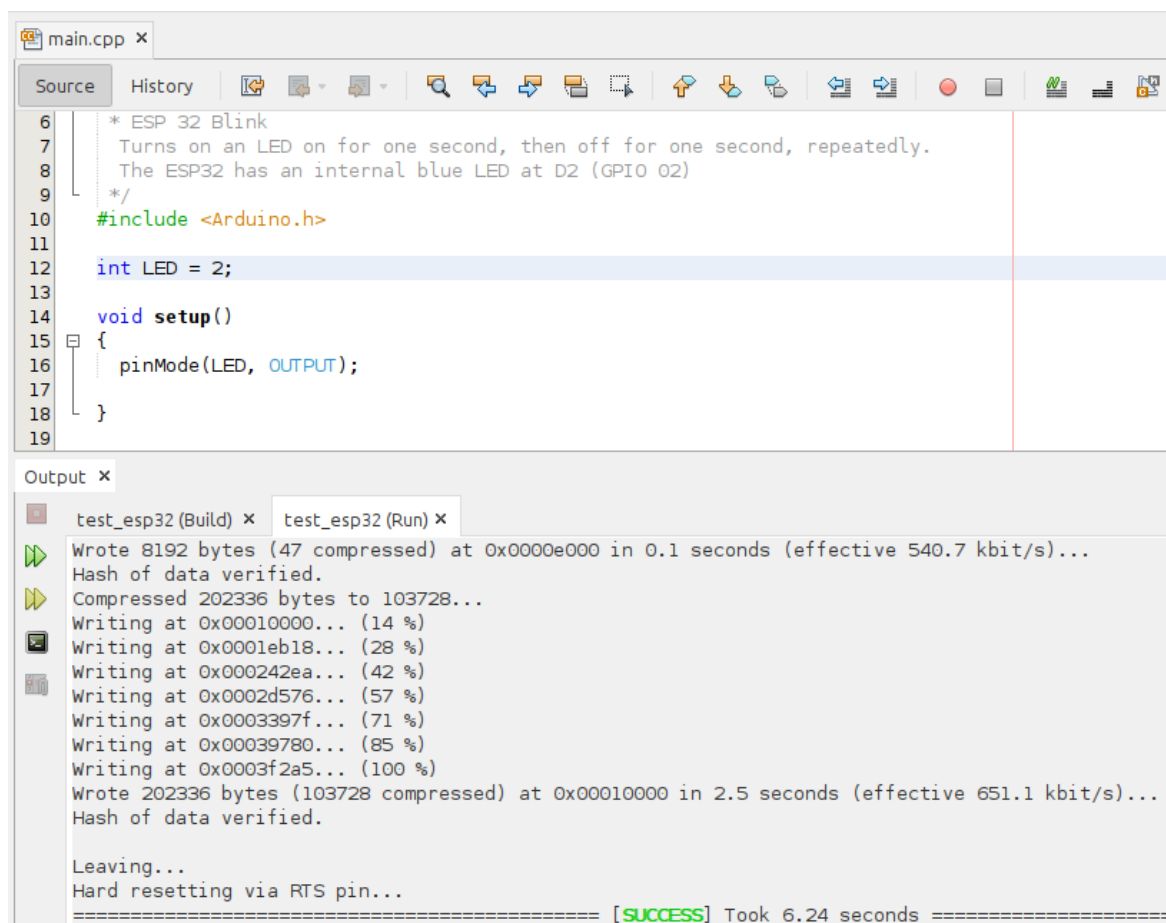
Pour téléverser l'exécutable sur la carte, modifiez les propriétés du projet

Run Command **pio run --target upload**



puis téléverser le firmware en utilisant la commande Run Project (triangle vert) ou F6

Menu: Run > Run Project

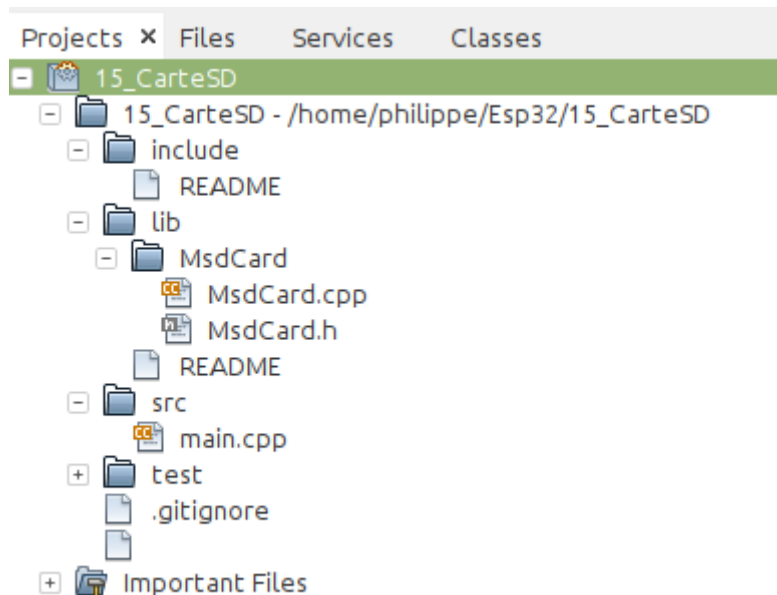


## 7 Ajouter des bibliothèques personnelles

Vous pouvez ajouter vos bibliothèques personnelles dans le dossier lib.

Le principe est de créer un sous-répertoire qui porte le même nom que le nom des fichiers du code source. Ce répertoire a la priorité la plus élevée pour Library Dependency Finder

Par exemple, voyez comment la bibliothèque MsdCard est installée :



Ensuite, dans src/main.c, vous devez utiliser l'inclusion suivante:

```
#include <MsdCard.h>
```

PlatformIO trouvera vos bibliothèques personnelles automatiquement, configurera les chemins d'inclusion du préprocesseur et les construira.

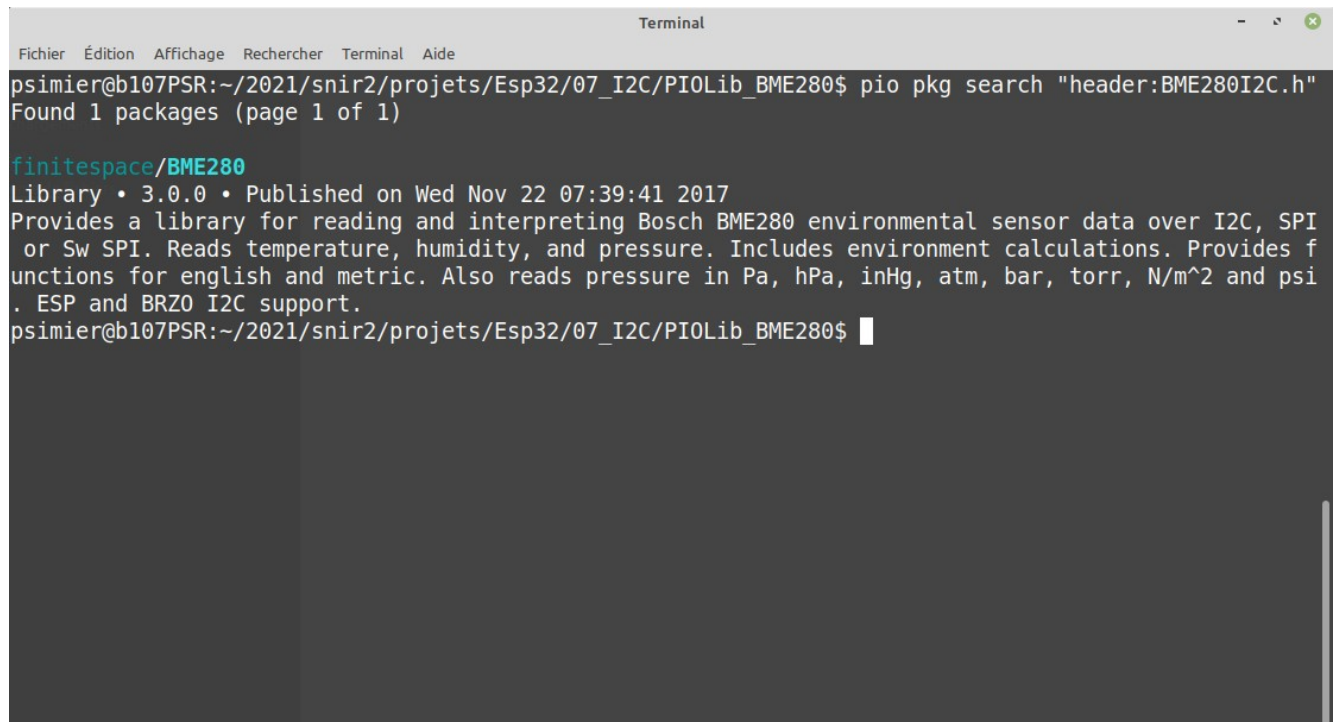
## 8 Ajouter des bibliothèques externes à votre projet

Prenons par exemple, le cas du capteur BME280 connecté sur le bus i2c.

La gestion des bibliothèques sous Platormio est contrôlée par la commande : **pio pkg**

La commande suivante permet de rechercher une bibliothèque particulière référencée sur PlatformIO.  
à partir d'un fichier header connu.

**pio pkg search "header:BME280I2C.h"**



```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
psimier@b107PSR:~/2021/snr2/projets/Esp32/07_I2C/PIOLib_BME280$ pio pkg search "header:BME280I2C.h"
Found 1 packages (page 1 of 1)

finitespace/BME280
Library • 3.0.0 • Published on Wed Nov 22 07:39:41 2017
Provides a library for reading and interpreting Bosch BME280 environmental sensor data over I2C, SPI
or Sw SPI. Reads temperature, humidity, and pressure. Includes environment calculations. Provides f
unctions for english and metric. Also reads pressure in Pa, hPa, inHg, atm, bar, torr, N/m^2 and psi
. ESP and BRZ0 I2C support.
psimier@b107PSR:~/2021/snr2/projets/Esp32/07_I2C/PIOLib_BME280$
```

Le résultat de la commande montre qu'une bibliothèque pour le BME280 est référencée.

Installation de la librairie avec la commande

**pio pkg install --library finitespace/BME280**

Le fichier platformio.ini a été modifié.

**lib\_deps** est l'option qui permet de spécifier les librairies à inclure dans le projet.

Exemple de contenu du fichier platformio.ini : après l'inclusion de la bibliothèque BME280

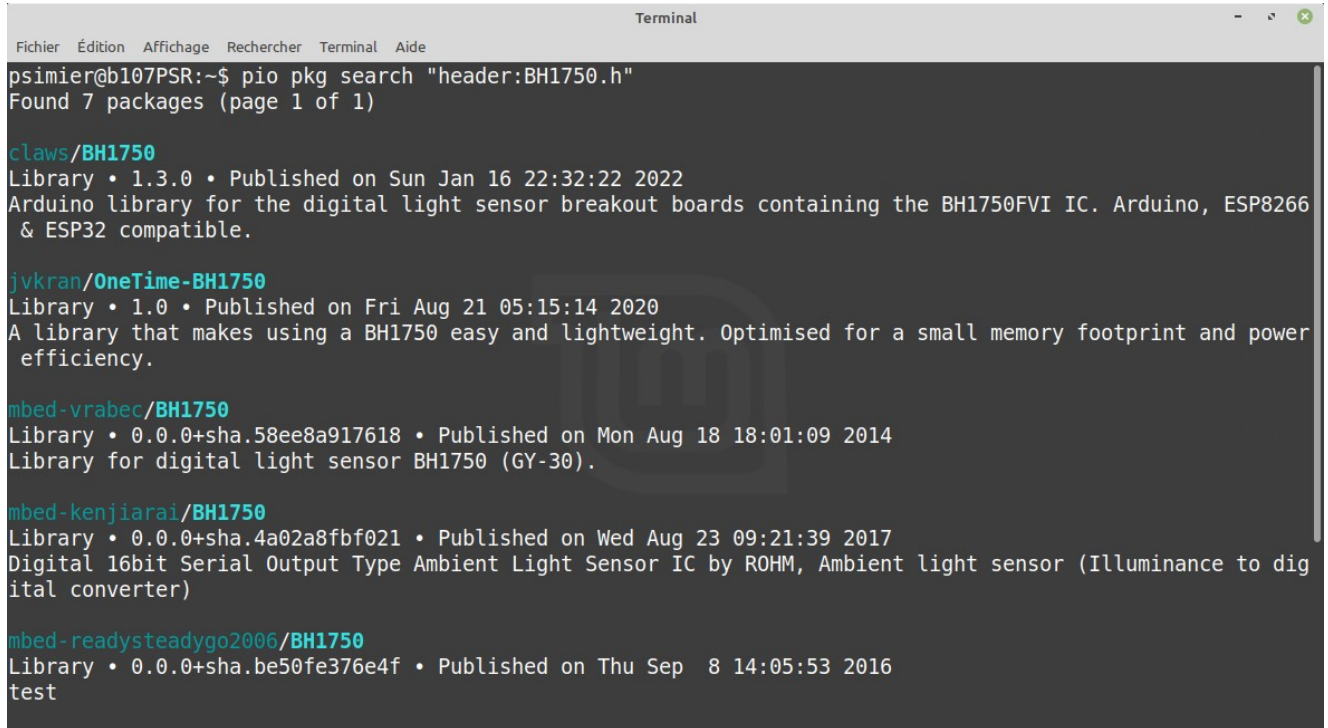
```
[env:lolin32]
platform = espressif32
board = lolin32
framework = Arduino
lib_deps = finitespace/BME280@^3.0.0
```

La bibliothèque est installée dans le sous-répertoire du projet **.pio/libdeps/lolin32/BME280**

## 9 Ajouter une bibliothèque globale à tous les projets

Recherche d'une bibliothèque pour le BH1750 (capteur d'éclairement)

```
pio pkg search "header:BH1750.h"
```



```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
psimier@b107PSR:~$ pio pkg search "header:BH1750.h"
Found 7 packages (page 1 of 1)

claws/BH1750
Library • 1.3.0 • Published on Sun Jan 16 22:32:22 2022
Arduino library for the digital light sensor breakout boards containing the BH1750FVI IC. Arduino, ESP8266 & ESP32 compatible.

jvkran/OneTime-BH1750
Library • 1.0 • Published on Fri Aug 21 05:15:14 2020
A library that makes using a BH1750 easy and lightweight. Optimised for a small memory footprint and power efficiency.

mbed-vrabec/BH1750
Library • 0.0.0+sha.58ee8a917618 • Published on Mon Aug 18 18:01:09 2014
Library for digital light sensor BH1750 (GY-30).

mbed-kenjiara1/BH1750
Library • 0.0.0+sha.4a02a8fbf021 • Published on Wed Aug 23 09:21:39 2017
Digital 16bit Serial Output Type Ambient Light Sensor IC by ROHM, Ambient light sensor (Illuminance to digital converter)

mbed-readysteadygo2006/BH1750
Library • 0.0.0+sha.be50fe376e4f • Published on Thu Sep 8 14:05:53 2016
test
```

Installation de la bibliothèque **claws/BH1750** dans un **stockage global**

```
pio pkg install --global --library claws/BH1750
```

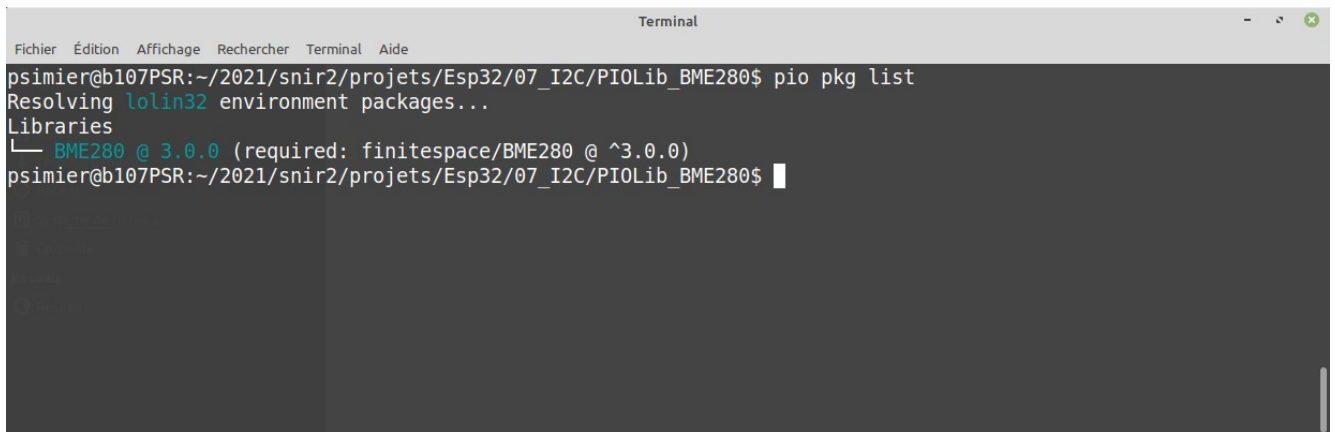
La bibliothèque peut être utilisée sans être déclarée dans le fichier **platformio.ini** du projet.

Les fichiers de la bibliothèque sont enregistrés dans le répertoire **~/ .platformio/libs**

```
philippe@philippe:~/ .platformio/lib$ ls
BH1750
```

## 10 Lister les bibliothèques installées

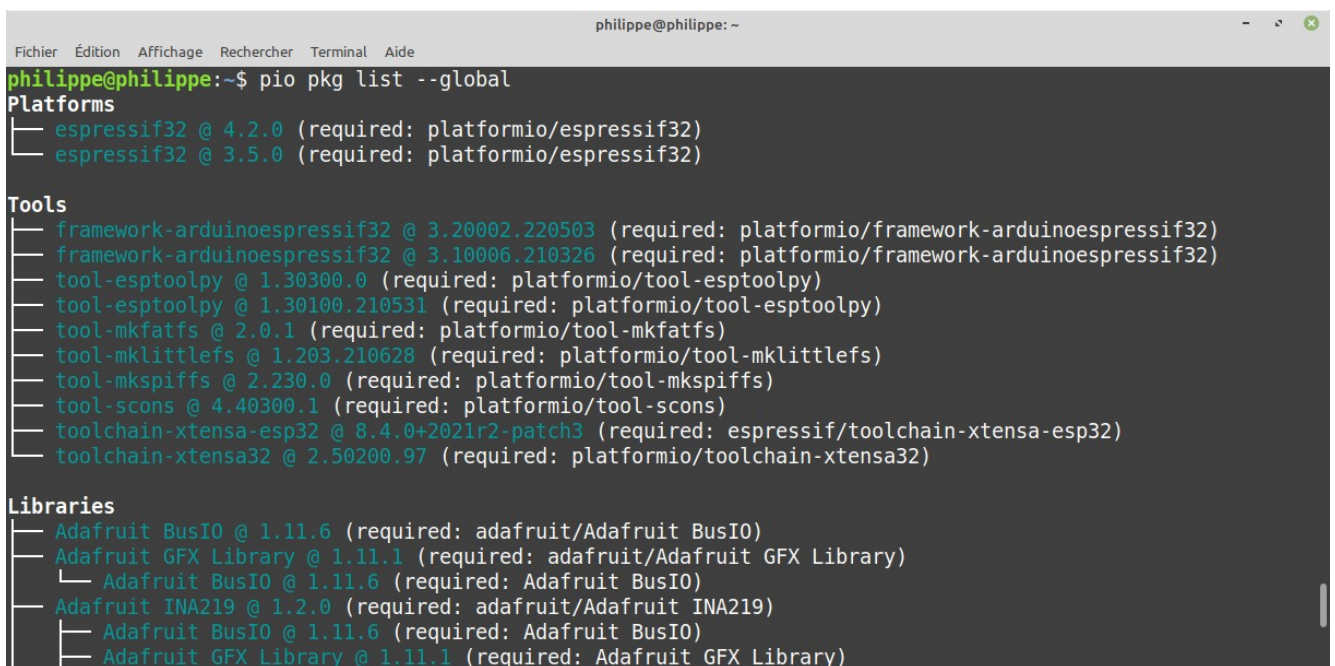
Lister les bibliothèques d'un projet **pio pkg list**



```
psimier@b107PSR:~/2021/snir2/projets/Esp32/07_I2C/PIOLib_BME280$ pio pkg list
Resolving lolin32 environment packages...
Libraries
└─ BME280 @ 3.0.0 (required: finitespace/BME280 @ ^3.0.0)
psimier@b107PSR:~/2021/snir2/projets/Esp32/07_I2C/PIOLib_BME280$
```

Lister les plateformes les outils et bibliothèques installées globalement (dans le dossier .platformio/lib de votre installation)

**pio pkg list --global** ou **pio pkg list -g**



```
philippe@philippe:~$ pio pkg list --global
Platforms
└─ espressif32 @ 4.2.0 (required: platformio/espressif32)
└─ espressif32 @ 3.5.0 (required: platformio/espressif32)

Tools
└─ framework-arduinoespressif32 @ 3.20002.220503 (required: platformio/framework-arduinoespressif32)
└─ framework-arduinoespressif32 @ 3.10006.210326 (required: platformio/framework-arduinoespressif32)
└─ tool-esptoolpy @ 1.30300.0 (required: platformio/tool-esptoolpy)
└─ tool-esptoolpy @ 1.30100.210531 (required: platformio/tool-esptoolpy)
└─ tool-mkfatfs @ 2.0.1 (required: platformio/tool-mkfatfs)
└─ tool-mklittlefs @ 1.203.210628 (required: platformio/tool-mklittlefs)
└─ tool-mkspiffs @ 2.230.0 (required: platformio/tool-mkspiffs)
└─ tool-scons @ 4.40300.1 (required: platformio/tool-scons)
└─ toolchain-xtensa-esp32 @ 8.4.0+2021r2-patch3 (required: espressif/toolchain-xtensa-esp32)
└─ toolchain-xtensa32 @ 2.50200.97 (required: platformio/toolchain-xtensa32)

Libraries
└─ Adafruit BusIO @ 1.11.6 (required: adafruit/Adafruit BusIO)
└─ Adafruit GFX Library @ 1.11.1 (required: adafruit/Adafruit GFX Library)
└─ Adafruit BusIO @ 1.11.6 (required: Adafruit BusIO)
└─ Adafruit INA219 @ 1.2.0 (required: adafruit/Adafruit INA219)
└─ Adafruit BusIO @ 1.11.6 (required: Adafruit BusIO)
└─ Adafruit GFX Library @ 1.11.1 (required: Adafruit GFX Library)
```

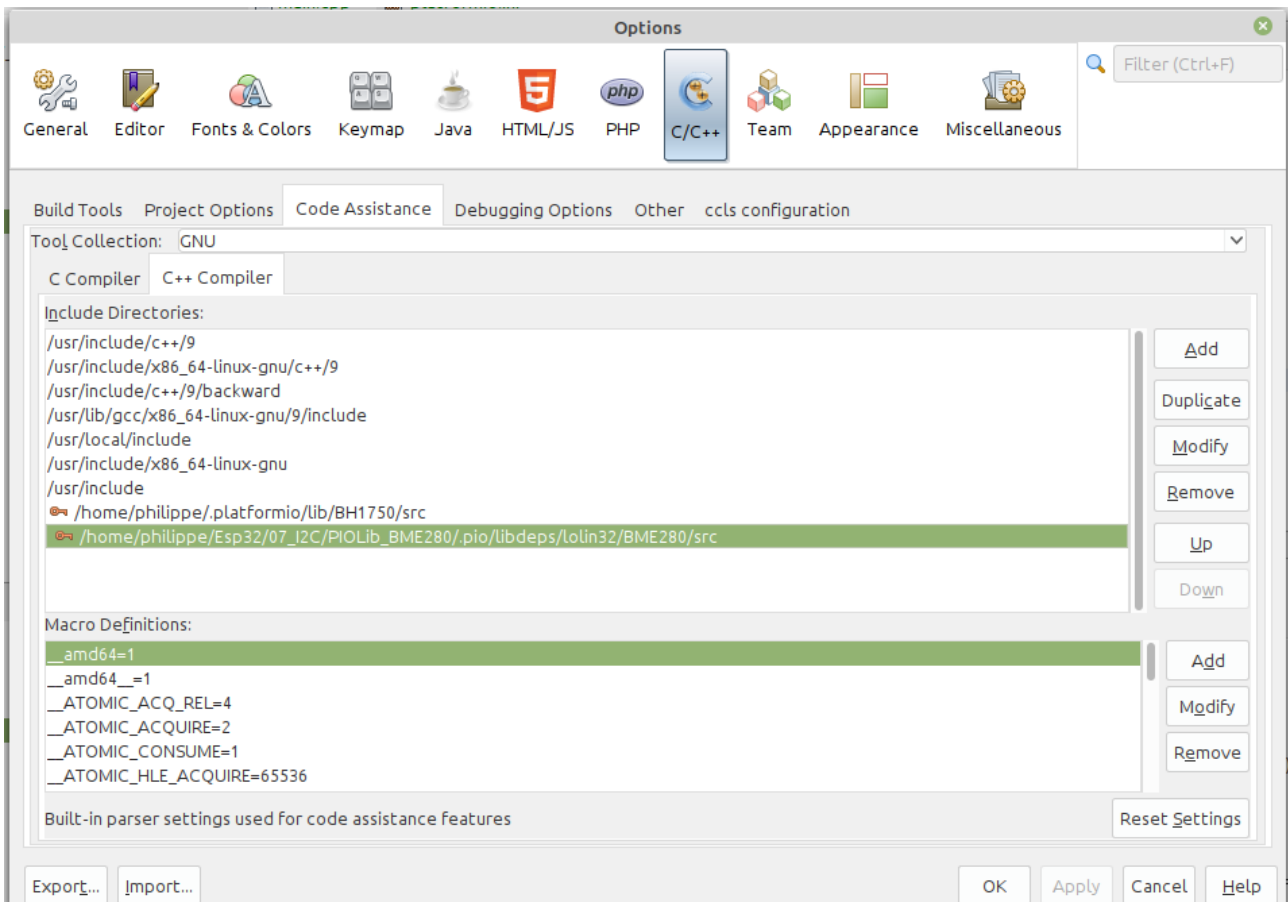
L'écran ci-dessus montre que deux plateformes espressif32 sont installées ainsi que deux frameworks arduino.

## 11 Code Assistance

Il faut renseigner les répertoires dans "code assistance" pour permettre la reconnaissance des "#include" et la complétion de code.

Pour nos deux bibliothèques BME280 et BH1750 on doit configurer les répertoires suivants :

Il faut bien sûr adapter les chemins à votre configuration



la copie d'écran suivante montre que maintenant les bibliothèques sont reconnues.

The screenshot displays the NetBeans IDE interface. The top toolbar includes icons for Source, History, and various development actions like Run, Debug, and Build. The main editor window shows the source code of `main.cpp` with the following content:

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <BME280.h>
4 #include <SPI.h>
5 #include <BME280I2C.h>
6 #include <BH1750.h>
7
8 #define SERIAL_BAUD 115200
9
10 BME280I2C::Settings parametrage(
11     BME280::OSR_X1,
12     BME280::OSR_X1,
13     BME280::OSR_X1,
14     BME280::Mode_Forced,
15     BME280::StandbyTime_1000ms,
16     BME280::Filter_Off,
17     BME280::SpiEnable_False,
18     BME280I2C::I2CAddr_0x77 // I2C address pour BME 280 Adafruit.
19 );
20
21 BME280I2C bme(parametrage);
22 BH1750 eclairement;
23
24 void printBME280Data(Stream* client);
25
26 void setup() {
27     parametrage
```

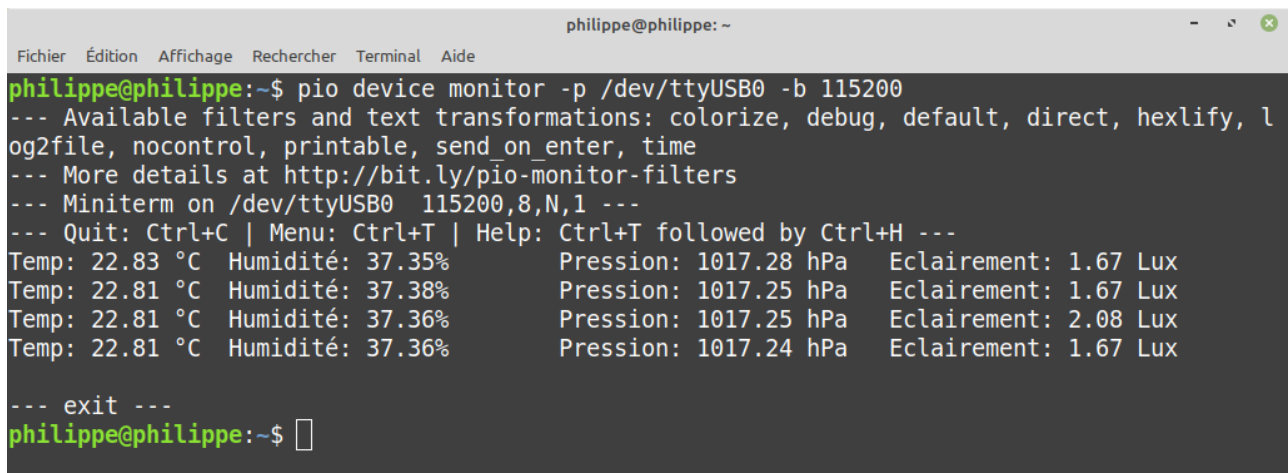
Below the editor, the Output window shows the results of the compilation and execution. It includes three tabs: `16_pioBME280 (Build)`, `PIOLib_BME280 (Build)`, and `PIOLib_BME280 (Run)`. The output text is as follows:

```
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
===== [SUCCESS] Took 9.10 seconds =====
RUN FINISHED; exit value 0; real time: 9s; user: 240ms; system: 3s
```



## 12 Tests du programme

La commande **pio device monitor -p /dev/ttyUSB0 -b 115200** permet d'ouvrir un moniteur pour afficher les messages envoyés par l'esp32.



```
philippe@philippe: ~  
Fichier  Édition  Affichage  Rechercher  Terminal  Aide  
philippe@philippe:~$ pio device monitor -p /dev/ttyUSB0 -b 115200  
--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time  
--- More details at http://bit.ly/pio-monitor-filters  
--- Miniterm on /dev/ttyUSB0 115200,8,N,1 ---  
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---  
Temp: 22.83 °C Humidité: 37.35% Pression: 1017.28 hPa Eclairement: 1.67 Lux  
Temp: 22.81 °C Humidité: 37.38% Pression: 1017.25 hPa Eclairement: 1.67 Lux  
Temp: 22.81 °C Humidité: 37.36% Pression: 1017.25 hPa Eclairement: 2.08 Lux  
Temp: 22.81 °C Humidité: 37.36% Pression: 1017.24 hPa Eclairement: 1.67 Lux  
--- exit ---  
philippe@philippe:~$
```

**Contrôle C** pour quitter

## 13 Téléverser des fichiers SPIFFS dans la mémoire flash

Nous devons parfois stocker des données dans la mémoire flash pour qu'elles persistent même après un redémarrage ou une mise hors tension.

La taille de la mémoire flash varie en fonction du module ESP32 embarqué sur la carte de développement. Les modules récents disposent généralement d'une mémoire flash de **4Mo** dont on pourra allouer 1Mo, 2Mo ou 3Mo au système de fichier (File System – **FS**).

Vous pouvez stocker votre application dans une partition et vos données dans une partition différente. Ce qui permet, par exemple, de ne mettre à jour que l'application avec une dernière version et de conserver tous vos fichiers de données intacts.

C'est la raison pour laquelle il existe deux cibles pour téléverser. Une pour l'application et une autre pour les données.

`pio run --target upload` et `pio run --target uploadfs`

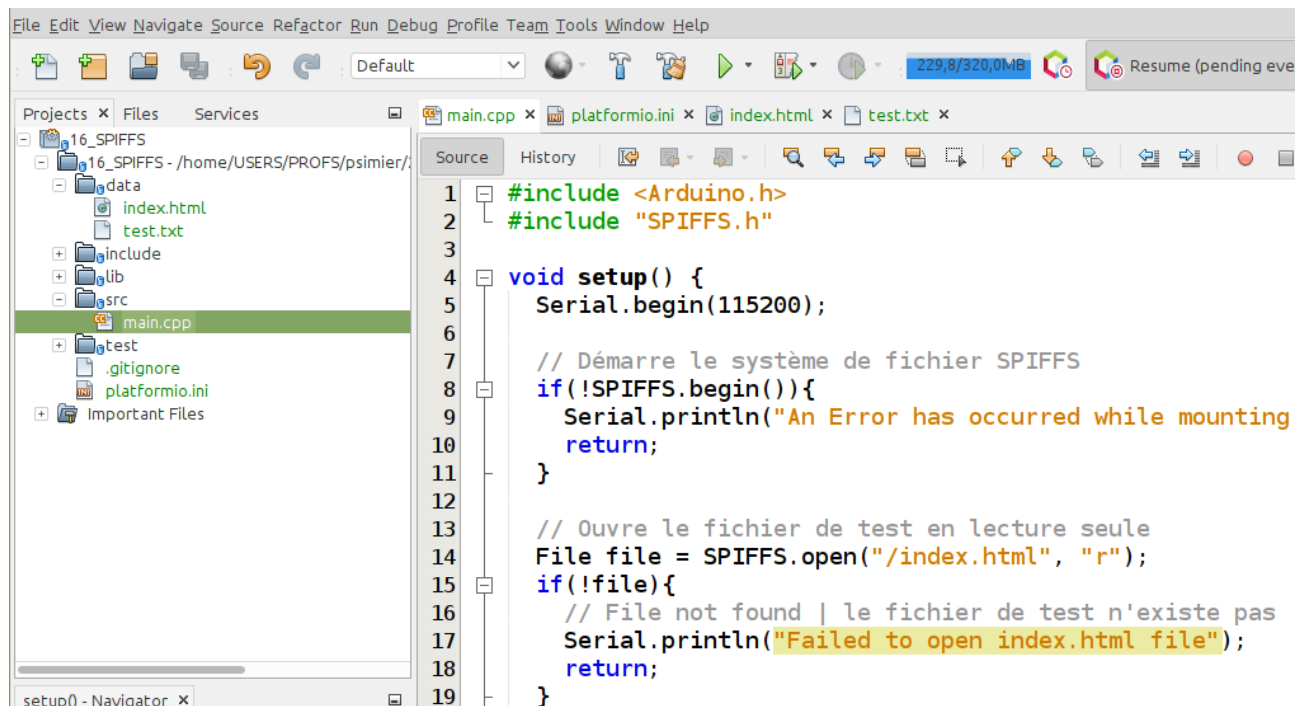
Voici un exemple d'arborescence des fichiers d'un projet ESP32 dont le code de l'interface HTML est séparé du code C++. En général, les fichiers d'un serveur WEB sont stockés dans un dossier nommé **data**.

Le dossier **data** contiendra tous les fichiers à téléverser dans la partition data de la mémoire flash de l'ESP32.

Dans l'arborescence du projet, le dossier **data** doit se trouver au même niveau que le dossier **src**.

Voici un exemple d'arborescence





La table de partition de la mémoire flash sur l'ESP32 fonctionne de manière très similaire à celle de notre ordinateur.

Par défaut, le framework alloue des portions de mémoire suivant la table appelée **Partition Table**.

Espressif a défini un schéma de partition par défaut.

<https://github.com/espressif/arduino-esp32/blob/master/tools/partitions/default.csv>

#	Name,	Type,	SubType,	Offset,	Size,	Flags
nvs,	data,	nvs,	0x9000,	0x5000,		
otadata,	data,	ota,	0xe000,	0x2000,		
app0,	app,	ota_0,	0x10000,	0x140000,		
app1,	app,	ota_1,	0x150000,	0x140000,		
spiffs,	data,	spiffs,	0x290000,	0x170000,		

Les partitions ne sont pas toutes répertoriées dans le fichier csv. Il manque la partition du chargeur de démarrage (offset 0x1000 et taille 0x7000) et une zone pour la table de partition (offset 0x8000 et taille 0x1000).

La première partition **nvs** est utilisée pour stocker l'étalonnage physique unique de l'appareil, les données WiFi, les informations de couplage Bluetooth et toute autre valeur à stocker au format NVS. La taille par défaut est de 20 Ko (0x5000 octets).

Comme vous pouvez le constater, la partition **data** au format **spiffs** se situe à la fin de façon à occuper tout l'espace disponible restant soit 1,5 Mo (0x170000 octets).

PlatformIO permet de définir finement la Partition Table à l'aide d'un fichier csv. Plus d'informations [ici](#). (toutefois, je n'ai pas testé cette possibilité).

Par défaut, les fichiers sont téléversés au format SPIFFS qui convient aux fichiers volumineux. La partition SPIFFS effectue également le nivellement de l'usure et la vérification de la cohérence du système de fichiers. Le SPIFFS ne prend pas en charge le cryptage flash.

Pour téléverser les fichiers enregistrés dans le dossier data, il suffit de lancer la commande

**pio run --target uploadfs** depuis le terminal en se plaçant dans le répertoire du projet, comme le montre la capture d'écran suivante.

```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide

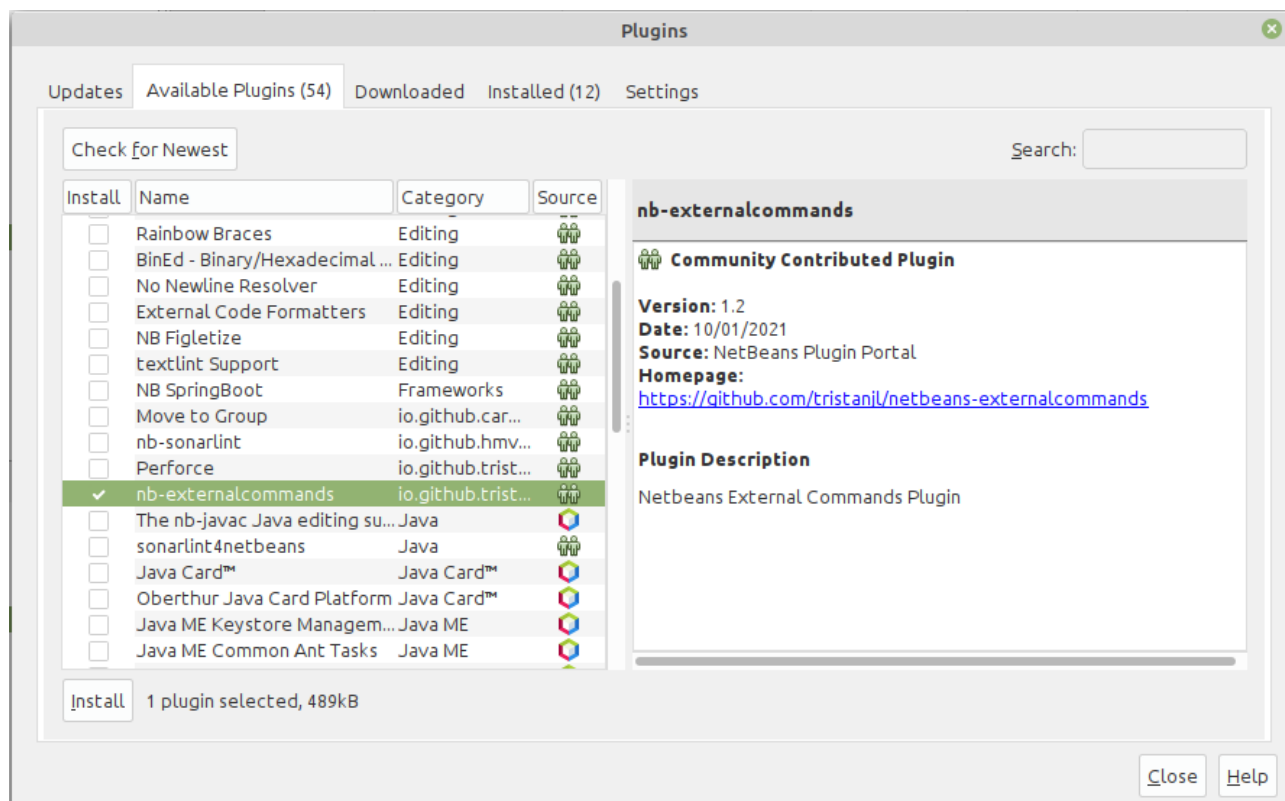
psimier@b107PSR:~/2021/snr2/projets/Esp32/16_SPIFFS$ pio run --target uploadfs
Processing lolin32 (platform: espressif32; board: lolin32; framework: arduino)
-----
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif32/lolin32.html
PLATFORM: Espressif 32 (3.3.2) > WEMOS LOLIN32
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
DEBUG: Current (esp-prog) External (esp-prog, iot-bus-jtag, jlink, minimodule, olimex-ar
limex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa)
PACKAGES:
- framework-arduinoespressif32 3.10006.210326 (1.0.6)
- tool-esptoolpy 1.30100.210531 (3.1.0)
- tool-mkspiffs 2.230.0 (2.30)
- toolchain-xtensa32 2.50200.97 (5.2.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 31 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <SPIFFS> 1.0
|   |-- <FS> 1.0
Building in release mode
Building SPIFFS image from 'data' directory to .pio/build/lolin32/spiffs.bin
/index.html
/test.txt
Looking for upload port...
Auto-detected: /dev/ttyUSB0
Uploading .pio/build/lolin32/spiffs.bin
esptool.py v3.1
Serial port /dev/ttyUSB0
Connecting....._
Chip is ESP32-D0WDQ6 (revision 1)
```

**Remarque :** À chaque fois que les fichiers du dossier **data** sont modifiés, il faudra les téléverser de nouveau manuellement.

## 14 Ajouter des commandes externes à Netbeans

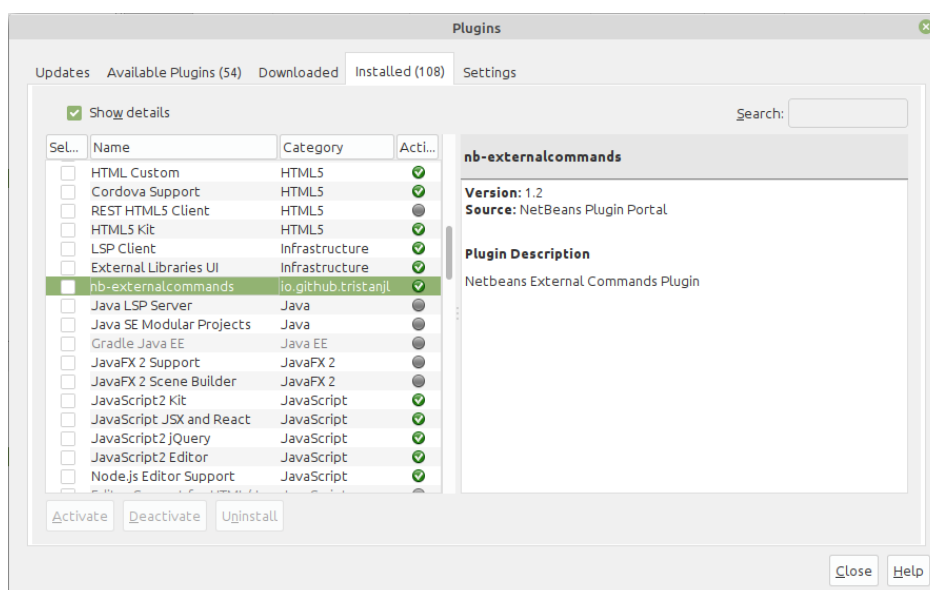
Installer le plug-in **nb-externalcommands**. Ce plug-in Netbeans ajoute un sous-menu de commandes externes à votre menu Outils qui comprend 10 commandes externes personnalisables. Nous utiliserons ces commandes externes pour lancer des scripts pio.

Menu: tools > Plugins



Sélectionner le plug-in puis cliquer sur le bouton **Install**

Après l'installation le plug-in apparaît dans le tableau des plug-ins installés. Sélectionner **Show details**



Les options peuvent être configurées dans "Outils->Options->Divers->Commandes externes"

Chaque entrée de ligne dans le menu de personnalisation fournit deux zones de saisie de texte.

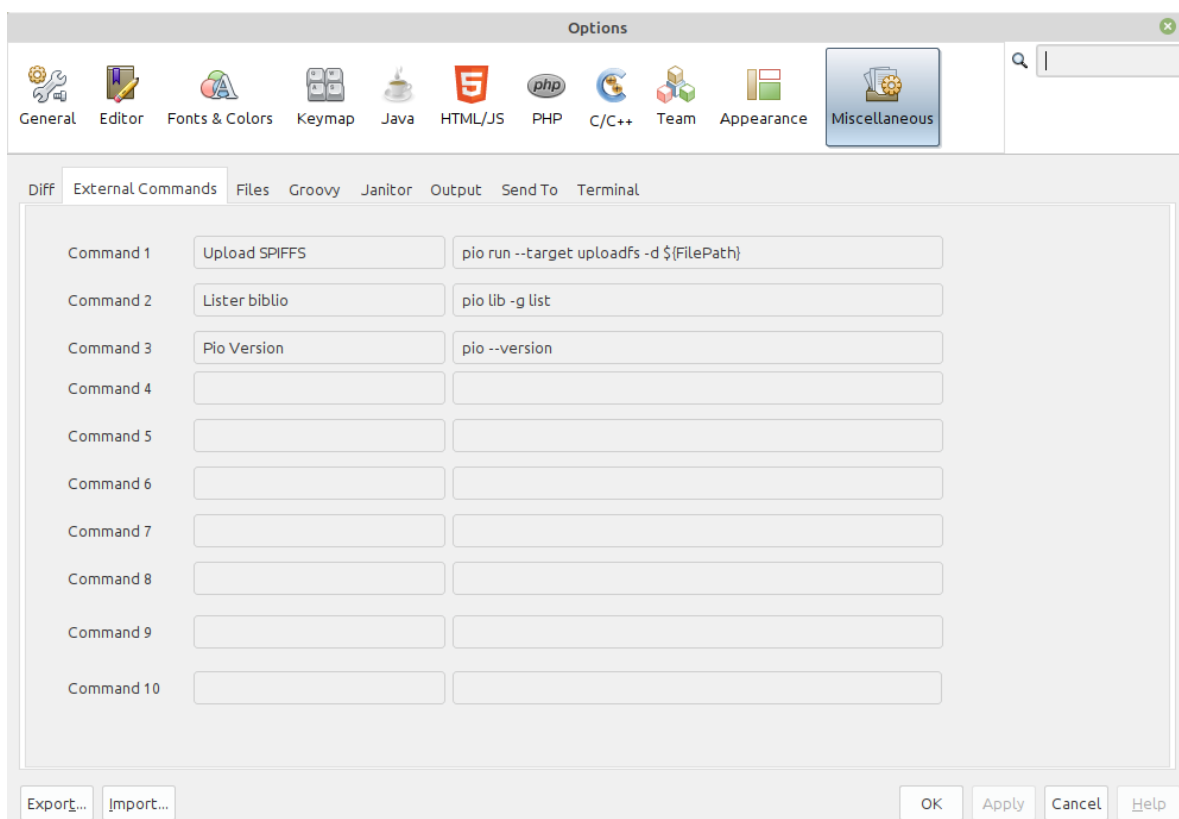
La première consiste à saisir le nom de la commande, qui sera **utilisé pour le titre de la fenêtre de sortie lors de l'exécution** de la commande.

Le second contient la commande et tous ses arguments. Les commandes prennent également en charge les remplacements de variables suivants :

`${FilePath}` - Le chemin d'accès complet du fichier actuel

## 1

## 2 Configuration des commandes externes



Command 1   Upload SPIFFS      **pio run --target uploadfs -d `${FilePath}`**

Vous devez redémarrer Netbeans pour que les modifications soient prises en compte.

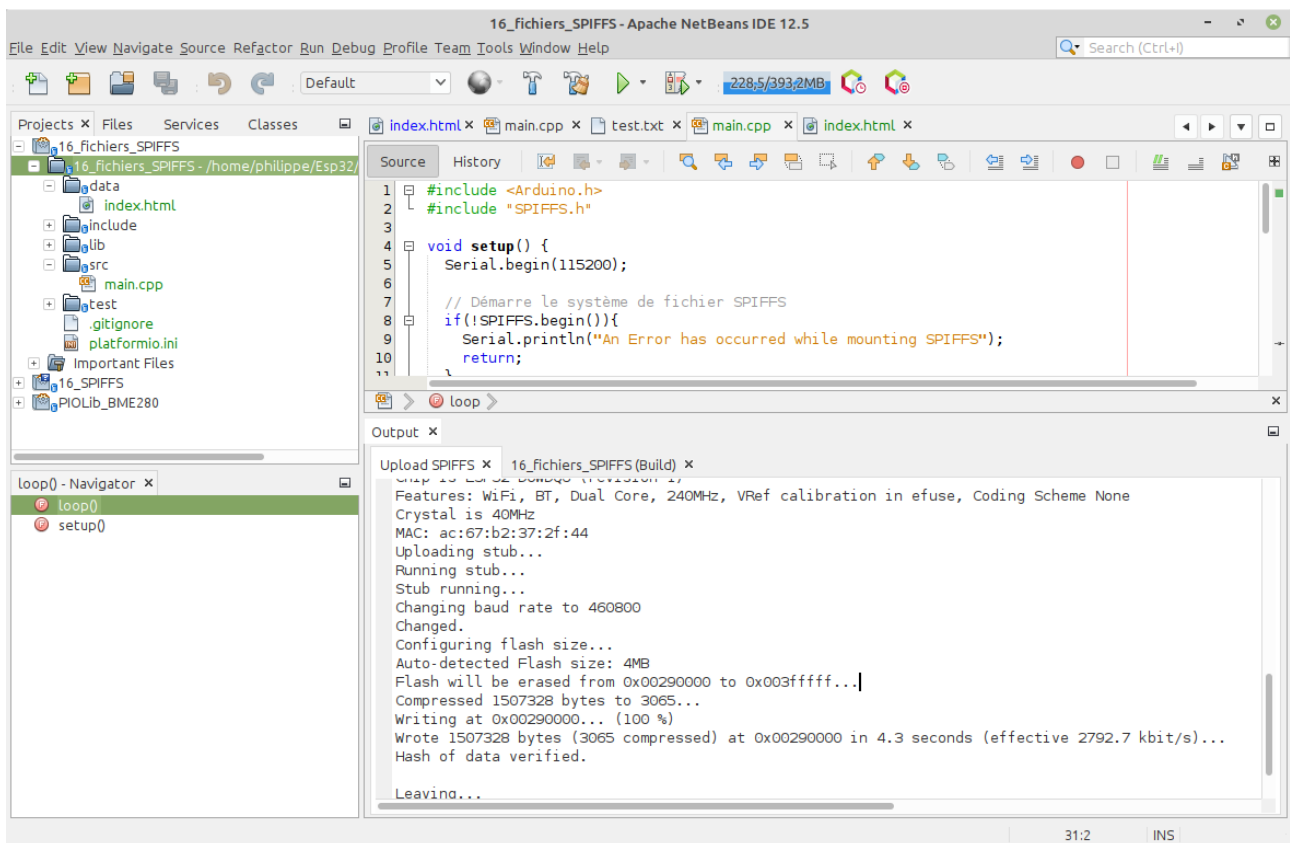
### 3 Téléverser les fichiers data avec Commande 1

Pour téléverser les fichiers du répertoire "**data** " donner le focus au dossier du projet. Le dossier ayant le focus apparaît dans l'arborescence surligné en vert. Puis sélectionner "**commande 1**"

Menu: tools > ExternalCommands > Commande 1

Le script associé à la commande s'exécute.

Comme le montre la capture d'écran suivante, la sortie de la commande apparaît dans un onglet de la fenêtre "output" à la fin de l'exécution de la commande. Soyez patient la commande peut prendre une dizaine de secondes pour s'exécuter.



## 15 Mettre à jour platformio

### 1 Mise à jour de la la plateforme espressif32

```
philippe@philippe:~$ pio platform update
Platform espressif32
-----
Updating platformio/espressif32          4.2.0
[Up-to-date]
Updating espressif/toolchain-xtensa-esp32 8.4.0+2021r2-patch3 @ 8.4.0+2021r2-
patch3[Up-to-date]
Updating platformio/framework-arduinoespressif32 3.20002.220503 @ ~3.20002.0
[Up-to-date]
Updating platformio/tool-esptoolpy        1.30300.0 @ ~1.30300.0
[Up-to-date]
Updating platformio/tool-mkspiffs         2.230.0 @ ~2.230.0
[Up-to-date]
Updating platformio/tool-mklittlefs       1.203.210628 @ ~1.203.0
[Up-to-date]
Updating platformio/tool-mkfatfs          2.0.1 @ ~2.0.0
[Up-to-date]
```

Remarque : Après une mise à jour du framwork il est possible de constater que la compilation n'est plus possible sur des codes qui se compilaient auparavant sans erreur.

Pour demander la compilation avec une version antérieur modifier le fichier platformio.ini

```
[env:lolin32]
platform = espressif32 @ ~3.5.0
board = lolin32
framework = arduino
```

### 2 Mise à jour des librairies

```
philippe@philippe:~$ pio lib -g update
Library Storage: /home/philippe/.platformio/lib
Updating adafruit/Adafruit BusIO          1.11.5          [Updating to 1.11.6]
Library Manager: Installing adafruit/Adafruit BusIO @ 1.11.6
Downloading [#####] 100%
Unpacking [#####] 100%
```



Library Manager: Adafruit BusIO @ 1.11.6 has been installed!

Library Manager: Removing Adafruit BusIO @ 1.11.5

Library Manager: Adafruit BusIO @ 1.11.5 has been removed!

Updating adafruit/Adafruit GFX Library 1.11.0

[Updating to 1.11.1]

### 3 Mise à jour de platformIO

```
root@philippe:/home/philippe# pio upgrade
```

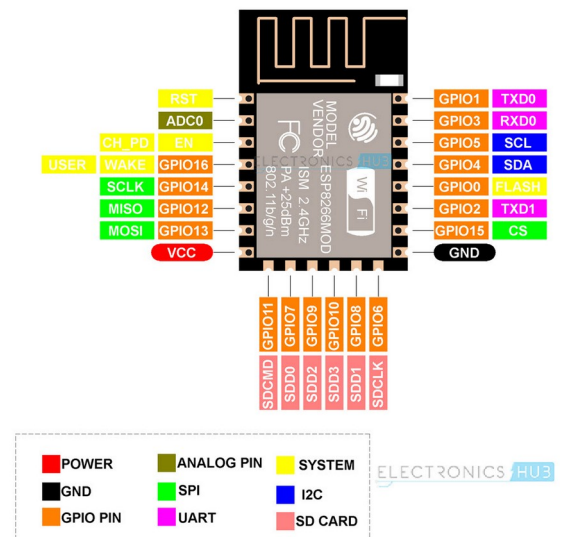
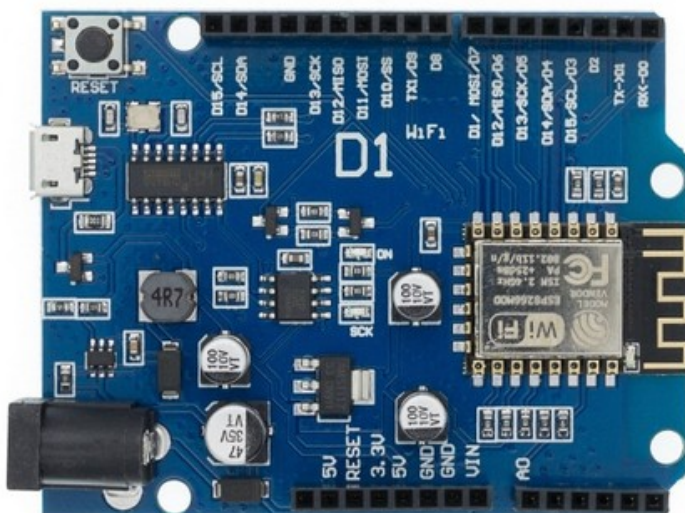
```
Please wait while upgrading PlatformIO ...
```

```
PlatformIO has been successfully upgraded to 6.0.0
```

```
Release notes: https://docs.platformio.org/en/latest/history.html
```

## 16 Création d'un projet pour la carte Wemos D1 R1

La carte WeMos D1 R1 intègre un module WIFI ESP8266-12 en natif, une mémoire SPIFFS de 3Mo. Sa tension de fonctionnement est 3,3 V – Elle possède seulement 1 Entrée Analogique , 15 Entrées / Sorties Digitales (un bus i2c, un bus spi 2 UART ).



Sur les cartes filles du lycée nous avons entre autres de connecté

GPIO0  
GPIO4  
GPIO5  
ADC0  
GPIO12

logique OUT  
Bus I2C SDA  
Bus I2C SCL  
Entrée Analogique  
logique (IN/OUT) Bus One Wire

led rouge ou verte  
OLED SSD1306  
LDR  
DS18S20

Affichage tout ou rien  
Affichage texte sur  
Afficheur 128\*64 px  
Capteur d'éclairement  
Mesure de la température

## Création du projet avec platformIO

lister toutes les cartes disponibles pour les esp8266

```
pio boards esp8266
```

d1	ESP8266	80MHz	4MB	80KB	WEMOS D1 R1
d1_mini	ESP8266	80MHz	4MB	80KB	WeMos D1 R2 and mini
d1_mini_lite	ESP8266	80MHz	1MB	80KB	WeMos D1 mini Lite
d1_mini_pro	ESP8266	80MHz	16MB	80KB	WeMos D1 mini Pro

puis créer un répertoire et exécuter la commande suivante

```
pio project init --ide netbeans --board d1
```