

Installation PlatformIO pour Netbeans

Table des matières

Installation PlatformIO pour Netbeans.....	1
1 Introduction.....	1
2 Installation de la commande python.....	1
3 Installation de platformIO.....	2
4 installation du path.....	3
5 Installation des règles udev.....	3
6 Création d'un projet Netbeans pour esp32.....	4
7 Ajouter des librairies à un projet.....	9
8 Ajouter des librairies externes à votre projet.....	10
9 Ajouter une bibliothèque globale à tous les projets.....	11
10 Code Assistance.....	12
11 Téléverser des fichiers SPIFFS dans la mémoire flash.....	14

1 Introduction

Le principal problème qui repousse les gens du monde embarqué est un 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 de systèmes embarqués et pour les développeurs de logiciels qui écrivent des applications pour des produits embarqués.

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 PlatformIO est le suivant :

- L'utilisateur choisit l'ide (netbeans) et la carte cible (esp32 lolin32)
- 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 de 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 récente de linux)

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

Comme le montre la capture d'écran si-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

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

Le script est disponible sur la page suivante.

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

Ensuite, exécutez la commande suivante :

```
python get_platformio.py ou python3 get_platformio.py
```

résultat

```
PlatformIO Core has been successfully installed into an isolated  
environment `/home/philippe/.platformio/penv`!  
  
The full path to `platformio.exe` is  
`/home/philippe/.platformio/penv/bin/platformio`  
  
If you need an access to `platformio.exe` from other applications,  
please install Shell Commands  
(add PlatformIO Core binary directory  
`/home/philippe/.platformio/penv/bin` to the system environment  
PATH variable):
```

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

4 installation du path

à la fin du fichier .profile de l'utilisateur ajouter la ligne suivante

nano .profile

```
export PATH=$PATH:~/platformio/penv/bin
```

fermer la session puis se reconnecter.

Vérification de la prise en compte

```
philippe@portable:~$ pio --version  
PlatformIO Core, version 5.2.2
```

5 Installation des règles udev

Les utilisateurs de Linux doivent 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
```

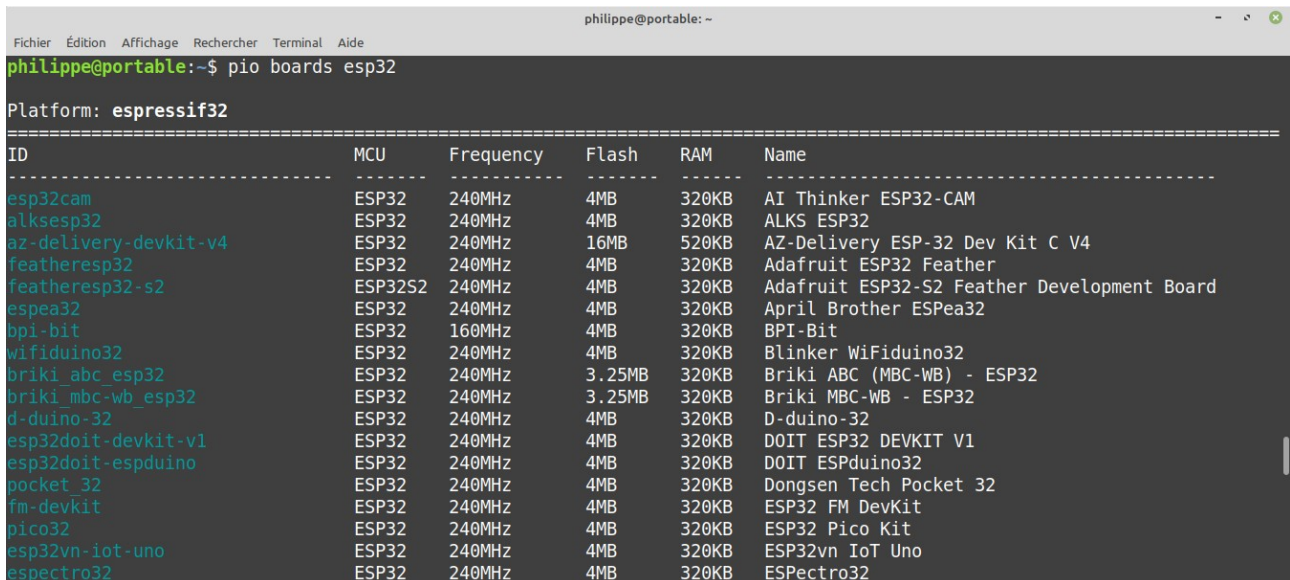
redémarrer le service udev

```
sudo service udev restart
```

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
espea32	ESP32	240MHz	4MB	320KB	April Brother ESPea32
bpi-bit	ESP32	160MHz	4MB	320KB	BPI-Bit
wifiduino32	ESP32	240MHz	4MB	320KB	Blinker WiFiduino32
briki_abc_esp32	ESP32	240MHz	3.25MB	320KB	Briki ABC (MBC-WB) - ESP32
briki_mbc-wb_esp32	ESP32	240MHz	3.25MB	320KB	Briki MBC-WB - ESP32
d-duino-32	ESP32	240MHz	4MB	320KB	D-duino-32
esp32doit-devkit-v1	ESP32	240MHz	4MB	320KB	DOIT ESP32 DEVKIT V1
esp32doit-espduino	ESP32	240MHz	4MB	320KB	DOIT ESPduino32
pocket_32	ESP32	240MHz	4MB	320KB	Dongsen Tech Pocket 32
fm-devkit	ESP32	240MHz	4MB	320KB	ESP32 FM DevKit
pico32	ESP32	240MHz	4MB	320KB	ESP32 Pico Kit
esp32vn-iot-uno	ESP32	240MHz	4MB	320KB	ESP32vn IoT Uno
espectro32	ESP32	240MHz	4MB	320KB	ESPECTRO32

La première colonne donne ID à utiliser pour chaque carte référencée.

Au lycée nous avons des cartes **ttgo-t1** pour le projet ballon

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

se placer 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

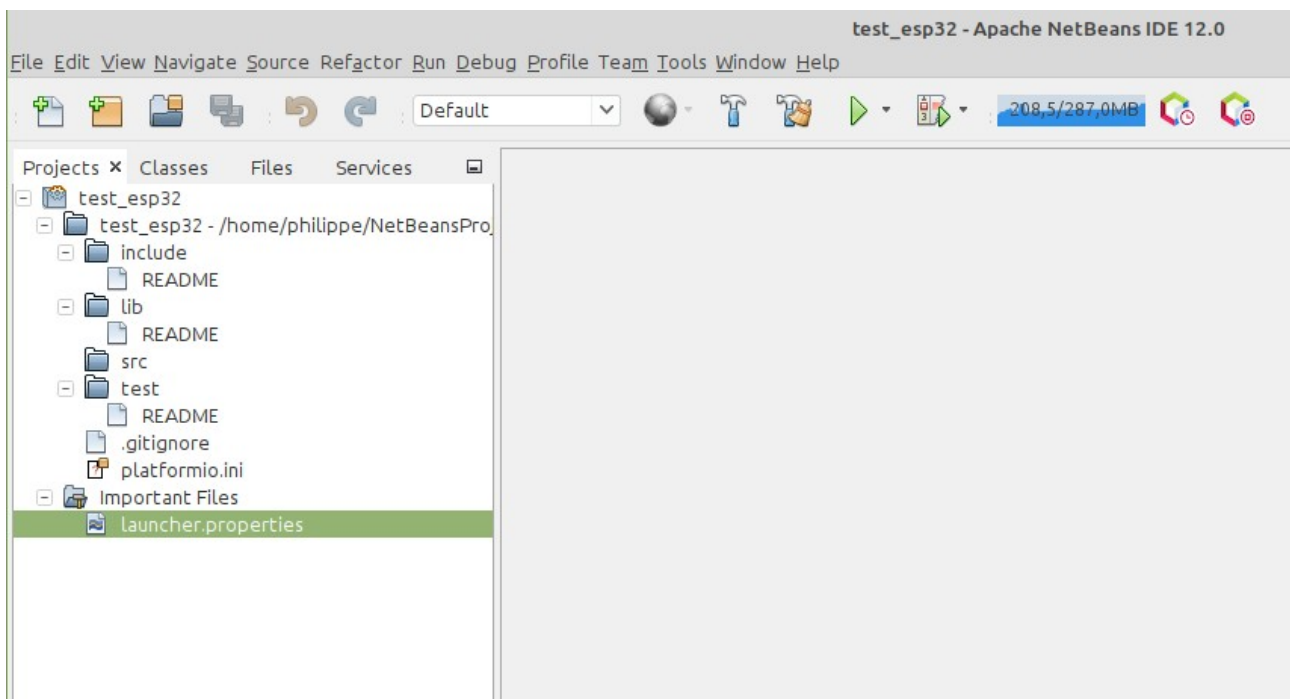
The current working directory /home/philippe/NetBeansProjects/test_esp32 will be used for the project.

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.

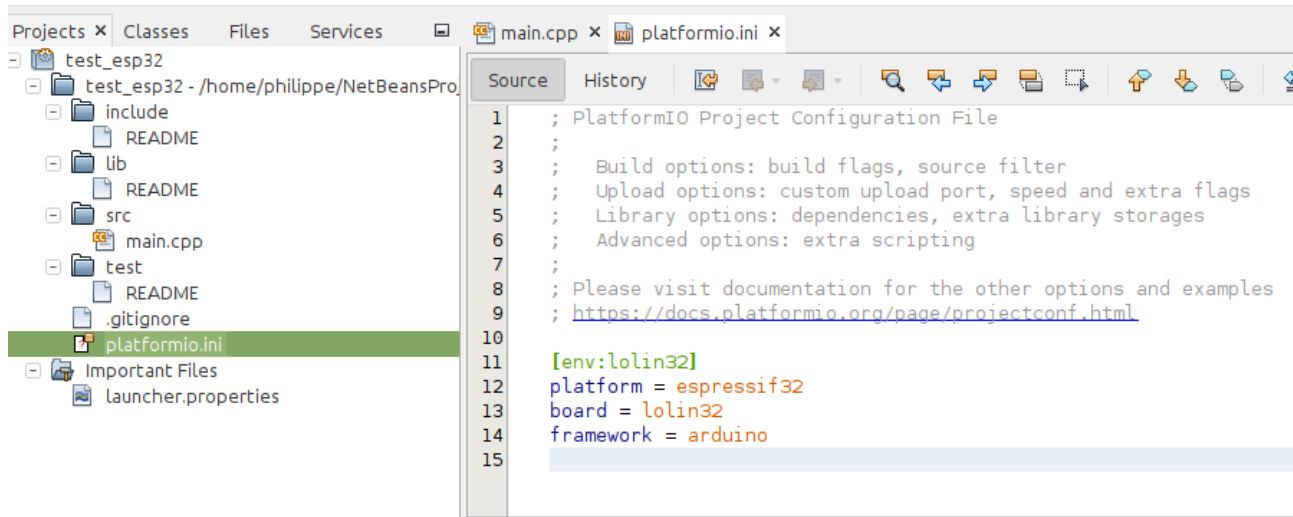
Project has been successfully initialized including configuration files for 'netbeans' IDE.
philippe@portable:~/NetBeansProjects/test_esp32$
```

La tool chaine est maintenant installée.

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



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

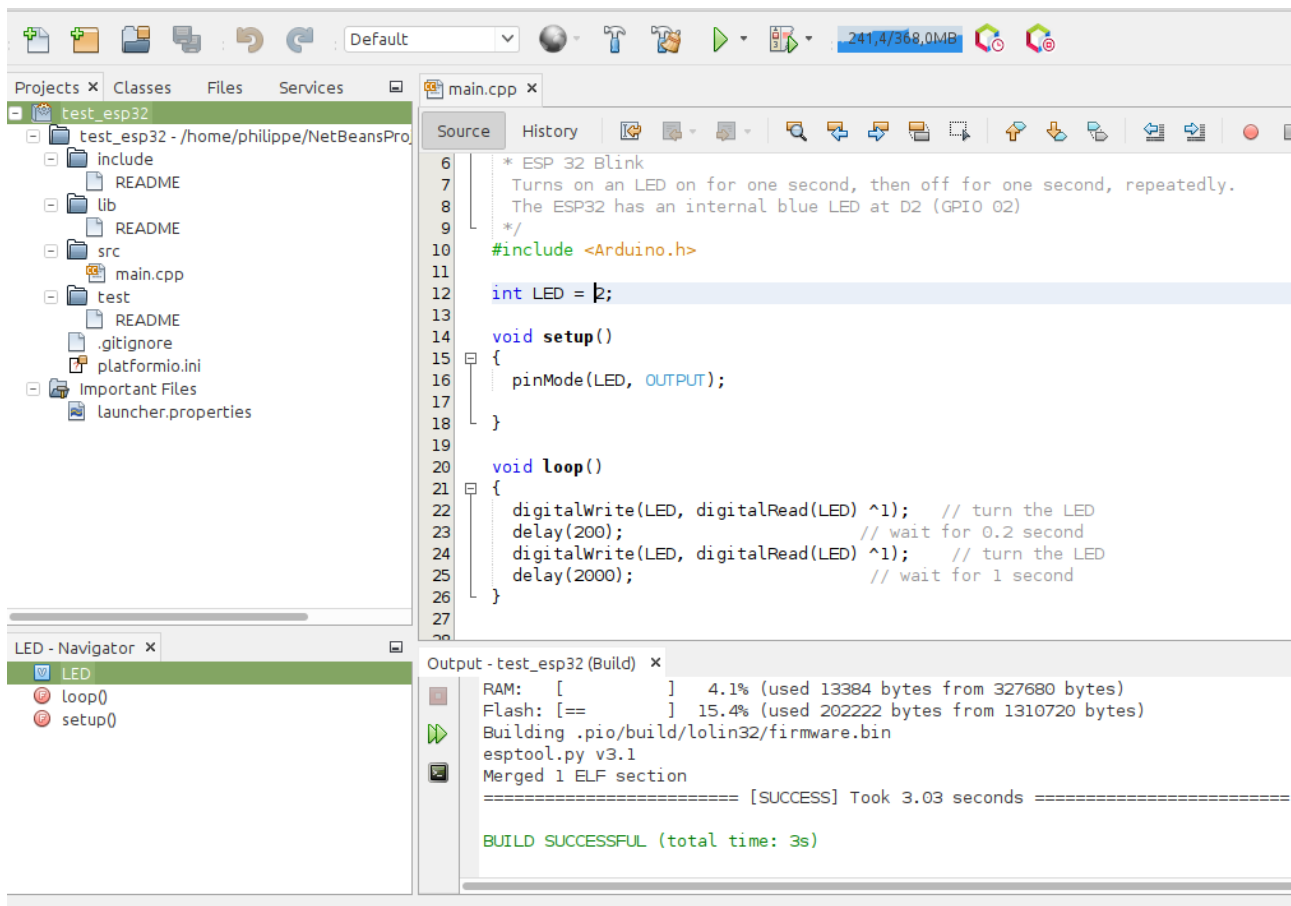


Avec ESP32 deux frameworks sont disponibles **arduino** et **espidf**

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

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

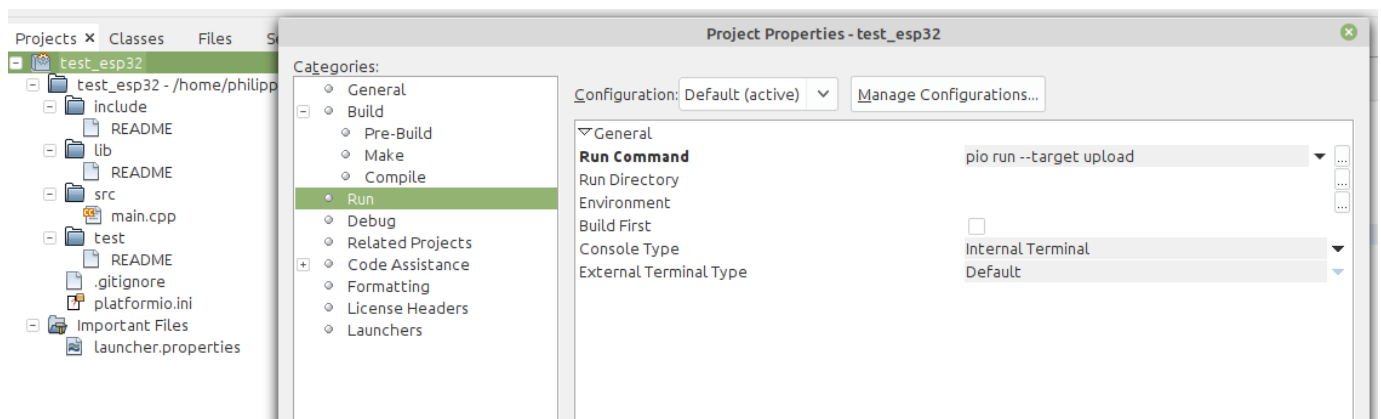
Ajoutez de nouveaux fichiers au répertoire src (*.c, *.cpp, *.ino, etc.) via un clic droit sur le dossier src dans le volet "Projects"



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

Pour uploader l'exécutable sur la carte modifier les propriétés du projet

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



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

Menu: Run > Run Project

The screenshot displays the NetBeans IDE interface. The top pane, titled 'main.cpp', contains the following C++ code:

```
6  * ESP 32 Blink
7  Turns on an LED on for one second, then off for one second, repeatedly.
8  The ESP32 has an internal blue LED at D2 (GPIO 02)
9  */
10 #include <Arduino.h>
11
12 int LED = 2;
13
14 void setup()
15 {
16     pinMode(LED, OUTPUT);
17 }
18
19
```

The bottom pane, titled 'Output', shows the results of building and running the program:

```
test_esp32 (Build) x  test_esp32 (Run) x
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 540.7 kbit/s)...
Hash of data verified.
Compressed 202336 bytes to 103728...
Writing at 0x00010000... (14 %)
Writing at 0x0001eb18... (28 %)
Writing at 0x000242ea... (42 %)
Writing at 0x0002d576... (57 %)
Writing at 0x0003397f... (71 %)
Writing at 0x00039780... (85 %)
Writing at 0x0003f2a5... (100 %)
Wrote 202336 bytes (103728 compressed) at 0x00010000 in 2.5 seconds (effective 651.1 kbit/s)...
Hash of data verified.

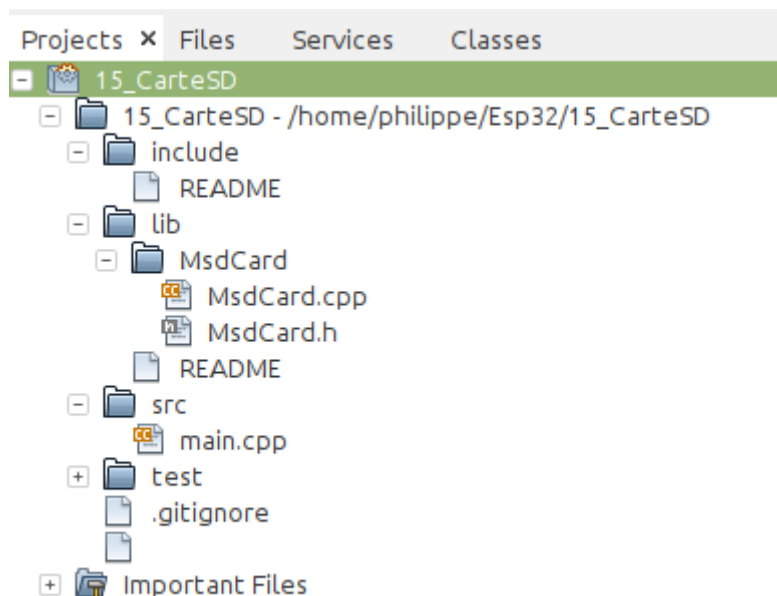
Leaving...
Hard resetting via RTS pin...
===== [SUCCESS] Took 6.24 seconds =====
```


7 Ajouter des bibliothèques à un projet

Vous pouvez mettre vos propres bibliothèques privées dans le dossier lib.

Le code source de chaque bibliothèque doit être placé dans un répertoire séparé, comme lib/private_lib/[voici les fichiers source]. Ce répertoire a la priorité la plus élevée pour Library Dependency Finder

Par exemple, voyez comment la bibliothèque MsdCard est organisée :



Ensuite, dans src/main.c, vous devez utiliser :

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

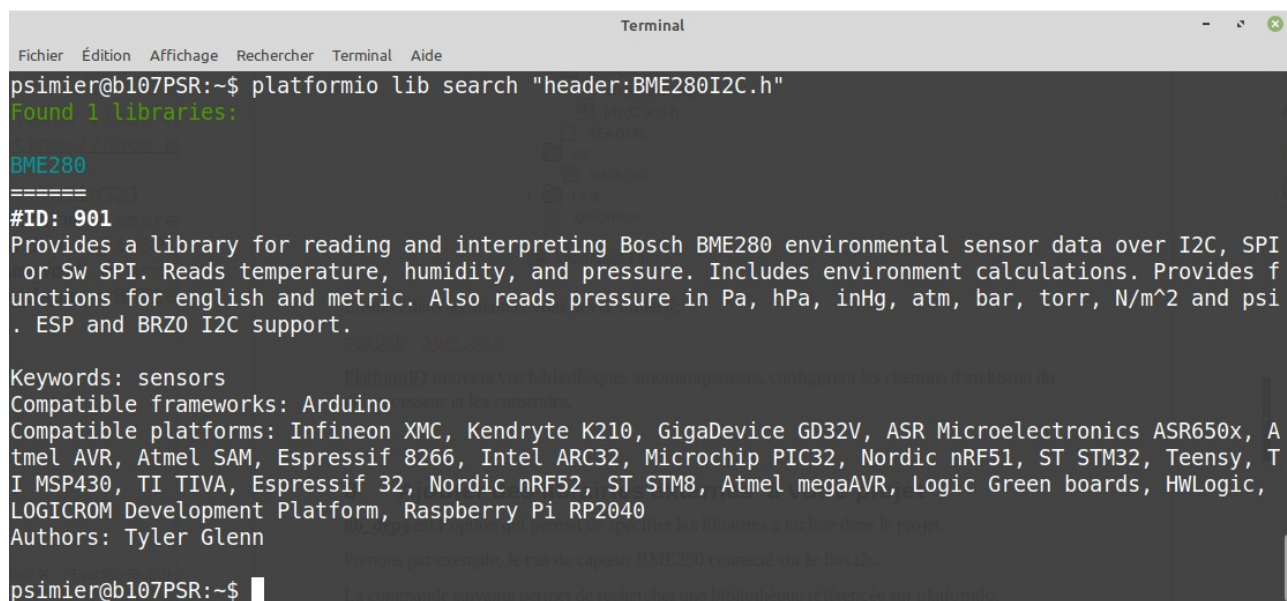
PlatformIO trouvera vos bibliothèques automatiquement, configurera les chemins d'inclusion du préprocesseur et les construira.

8 Ajouter des librairies externes à votre projet

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

La commande suivante permet de rechercher une bibliothèque référencée sur platformio.

platformio lib search "header:BME280I2C.h"



```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
psimier@b107PSR:~$ platformio lib search "header:BME280I2C.h"
Found 1 libraries:

BME280
=====
#ID: 901
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 functions for english and metric. Also reads pressure in Pa, hPa, inHg, atm, bar, torr, N/m^2 and psi. ESP and BRZ0 I2C support.

Keywords: sensors
Compatible frameworks: Arduino
Compatible platforms: Infineon XMC, Kendryte K210, GigaDevice GD32V, ASR Microelectronics ASR650x, Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, ST STM32, Teensy, TI MSP430, TI TIVA, Espressif 32, Nordic nRF52, ST STM8, Atmel megaAVR, Logic Green boards, HWLogic, LOGICROM Development Platform, Raspberry Pi RP2040
Authors: Tyler Glenn

psimier@b107PSR:~$
```

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

Installation de la librairie avec la commande **pio lib install**

pio lib install 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 lib search "header:BH1750.h"
```

```
philippe@philippe: ~/Esp32/07_I2C/PIOLib_BME280
Fichier  Édition  Affichage  Rechercher  Terminal  Aide

philippe@philippe:~/Esp32/07_I2C/PIOLib_BME280$ platformio lib search "header:BH1750.h"
Found 7 libraries:

BH1750
=====
#ID: 439
Arduino library for the digital light sensor breakout boards containing the BH1750FVI IC. Arduino, ESP8
266 & ESP32 compatible.

Keywords: bh1750fvi, light, lux, sensor, arduino, esp8266, esp32
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Espressif 32, Stm32
Authors: claws

BH1750
=====
#ID: 2188
Library for digital light sensor BH1750 (GY-30).

Keywords: bh1750, digital light sensor, gy-30
Compatible frameworks: Mbed
Compatible platforms: Atmel SAM, Freescale Kinetis, Nordic nRF51, NXP LPC, Silicon Labs EFM32, ST STM32
, Teensy, Nordic nRF52, Maxim 32, WIZNet W7500, RISC-V GAP, NXP i.MX RT
Authors: Michal Stehlik

BH1750
=====
#ID: 8593
```

Installation de la bibliothèque sur un **stockage global** en utilisant l'ID 439

```
pio lib -g install 439
```

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

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

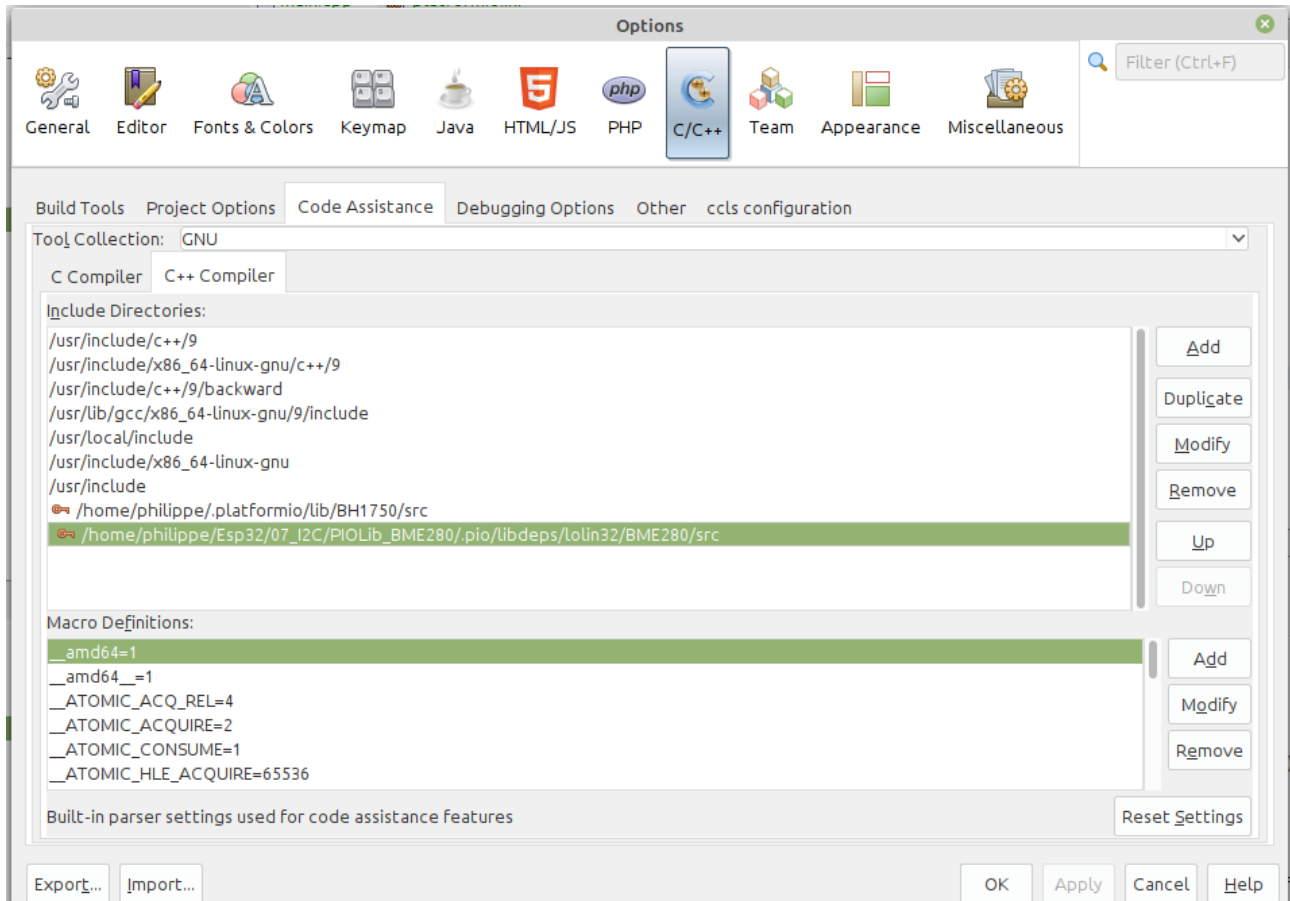
```
philippe@philippe:~/./platformio/lib$ ls
BH1750
```

10 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 sur adapter les chemins à votre configuration



The screenshot shows the NetBeans IDE interface. The top toolbar contains icons for Source, History, and various code editing functions. The main editor window displays the `main.cpp` file with the following code:

```
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 >
```

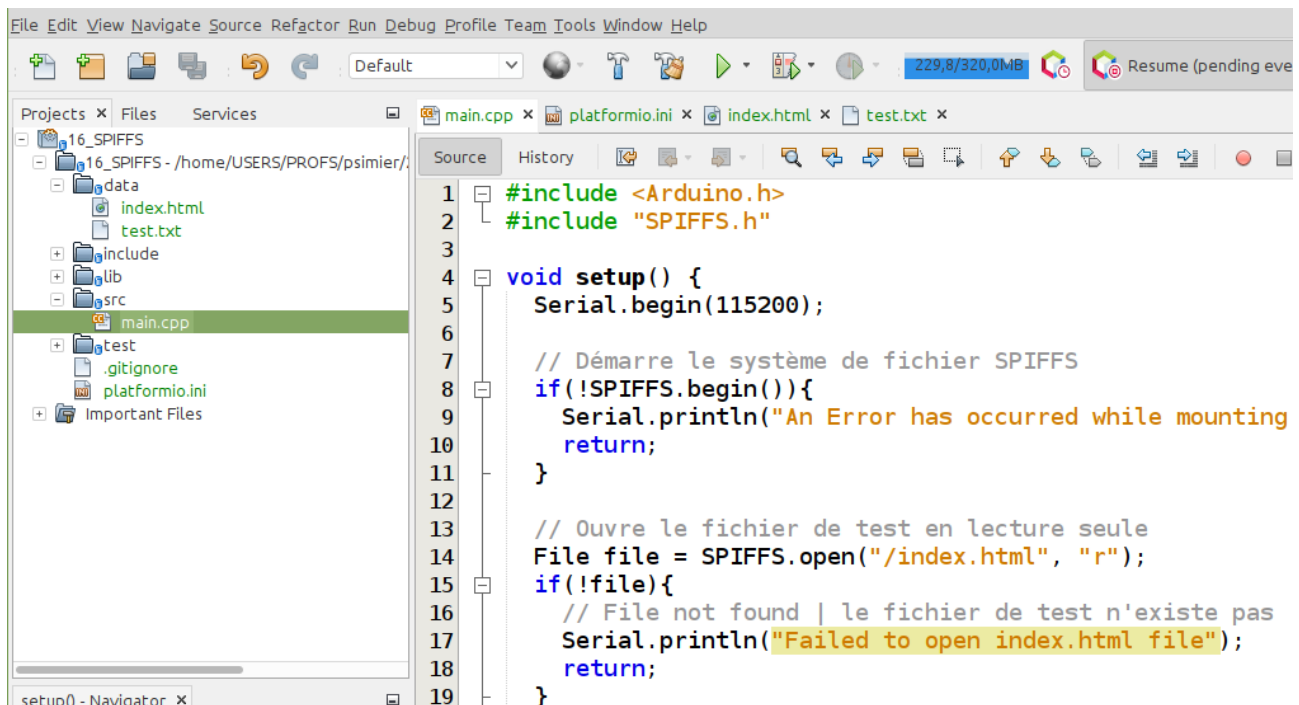
The bottom window shows the Output console with the following text:

```
16_pioBME280 (Build) x    PIOLib_BME280 (Build) x    PIOLib_BME280 (Run) x
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
```

11 Téléverser des fichiers SPIFFS dans la mémoire flash

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**).

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**.



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,		

Comme vous pouvez le constater, la zone spiffs se situe à la fin de façon à pouvoir occuper le plus d'espace disponible.

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

Pour téléverser les fichiers enregistrés dans la 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 : A chaque fois que les fichiers du dossier **data** sont modifiés, il faudra les téléverser de nouveau manuellement.