

Installation PlatformIO pour Netbeans

1 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

```
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

2 Installation de platformIO

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

<https://raw.githubusercontent.com/platformio/platformio-core-installer/master/get-platformio.py>

Ensuite, exécutez la commande suivante :

```
python 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>

3 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
```

4 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
```

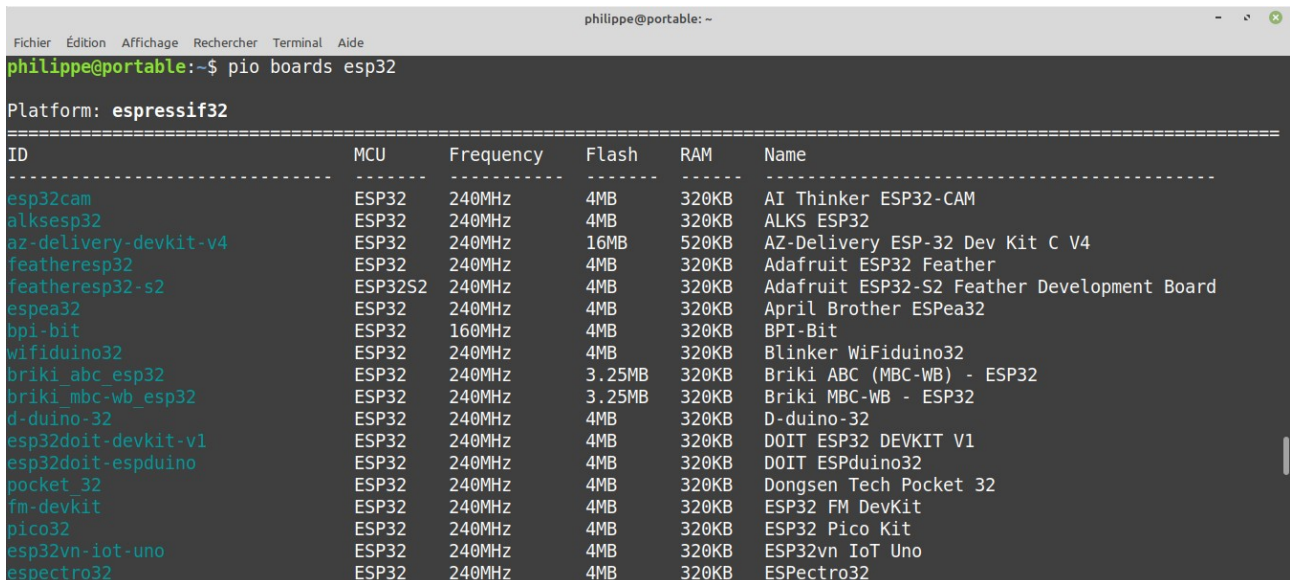
redémarrer le service udev

```
sudo service udev restart
```

5 Creation 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

se placer à l'intérieur puis lancer la commande **pio project init** avec comme argument 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

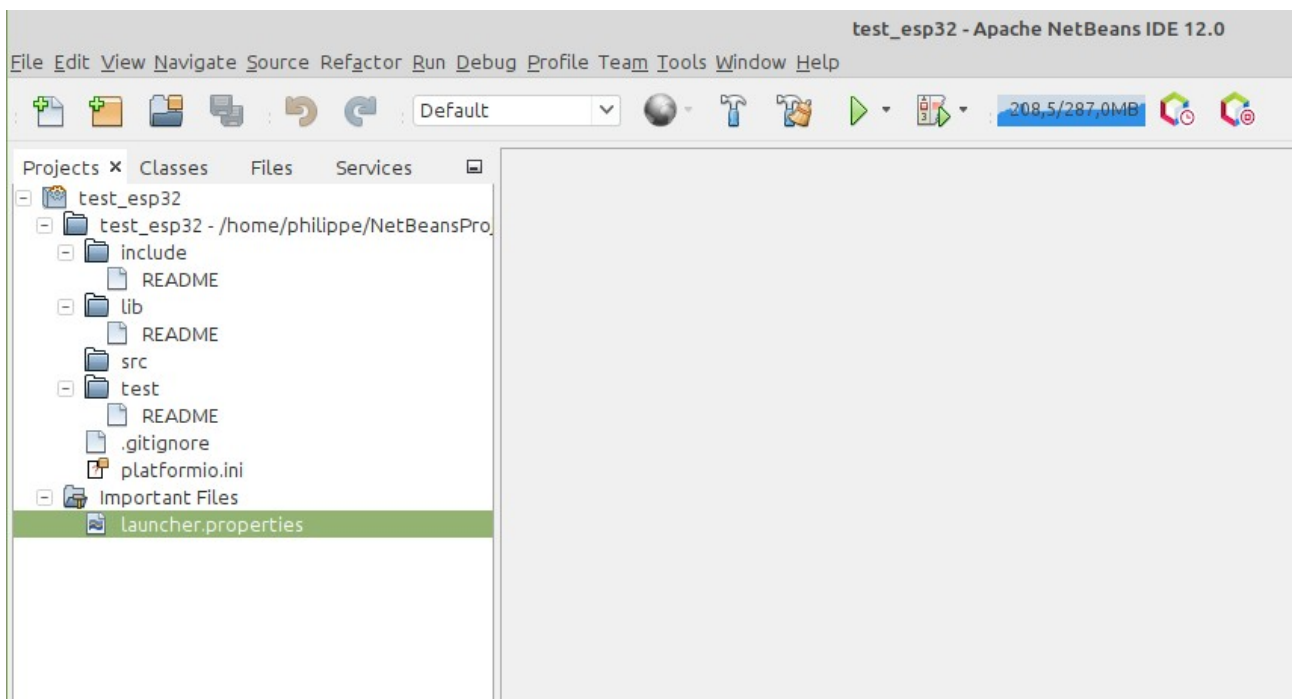
The current working directory /home/philippe/NetBeansProjects/test_esp32 will be used for the project.

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.

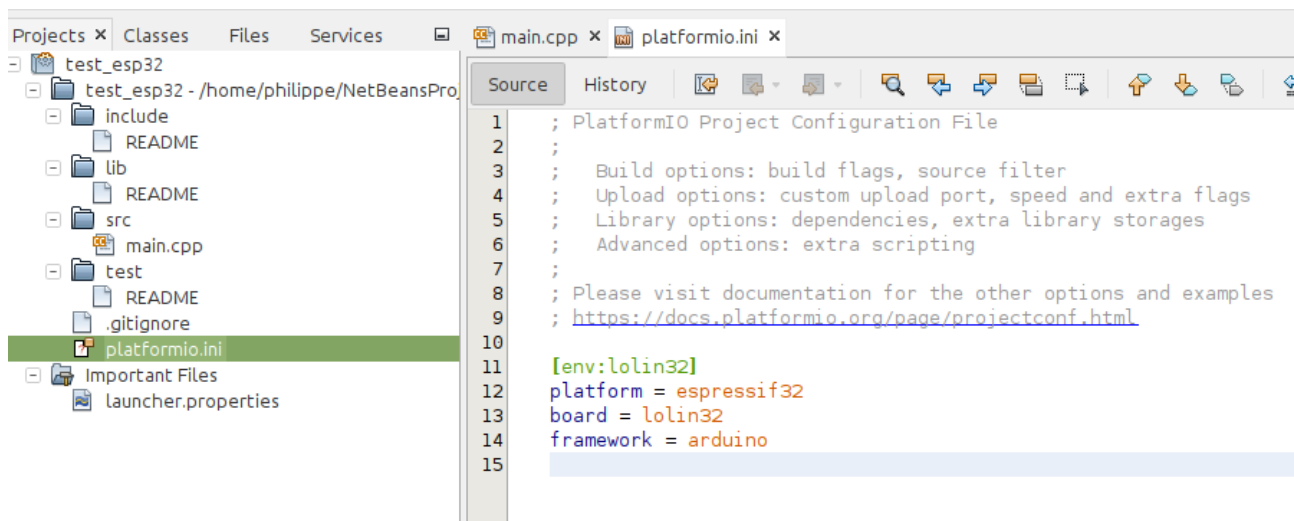
Project has been successfully initialized including configuration files for 'netbeans' IDE.
philippe@portable:~/NetBeansProjects/test_esp32$
```

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.

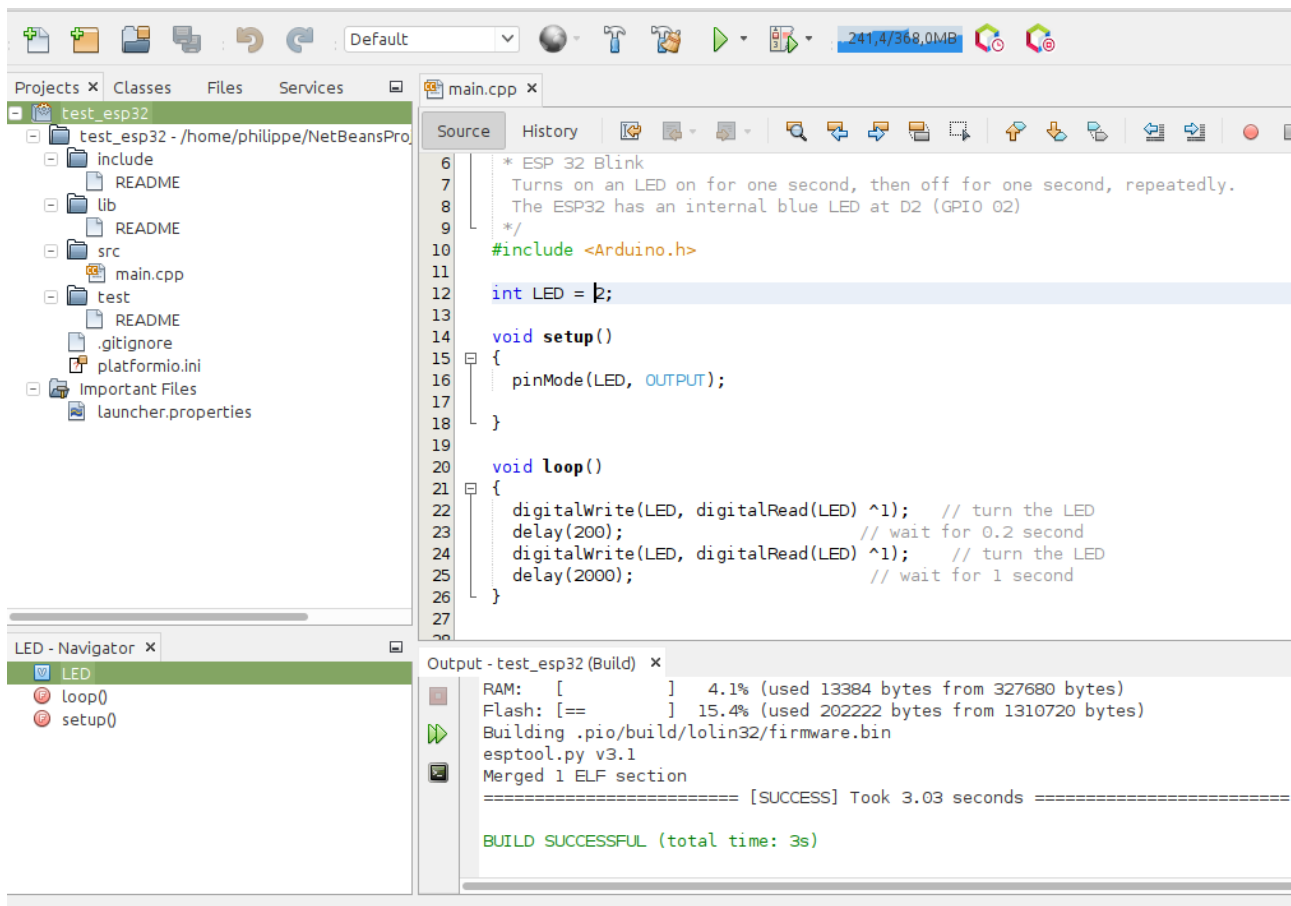


Avec ESP32 deux framework sont disponibles **arduino** et **espidf**

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

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

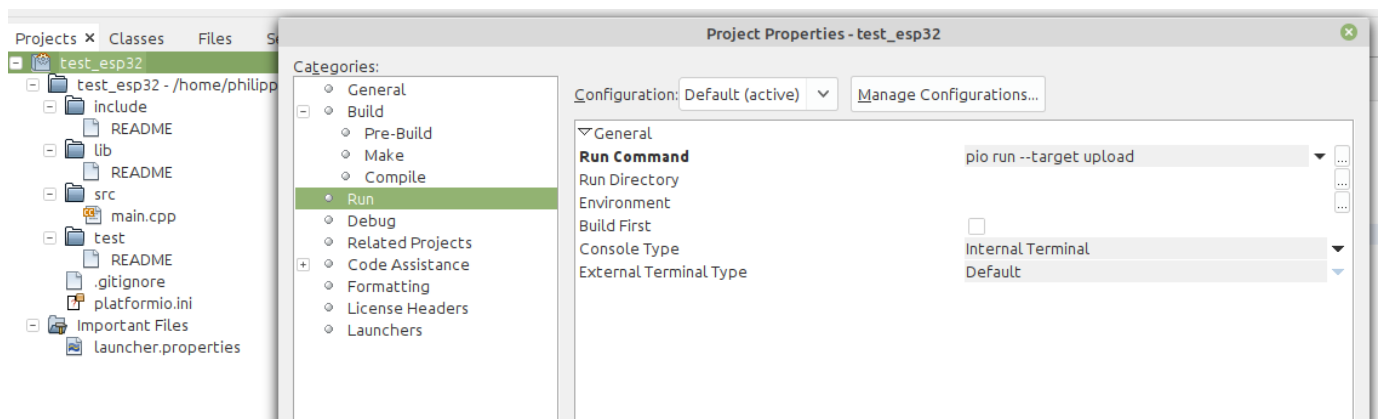
Ajoutez de nouveaux fichiers au répertoire src (*.c, *.cpp, *.ino, etc.) via un clic droit sur le dossier src dans le volet "Projects"



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

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)

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 build and run logs for 'test_esp32'. The logs indicate a successful compilation and execution:

```
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 =====
```