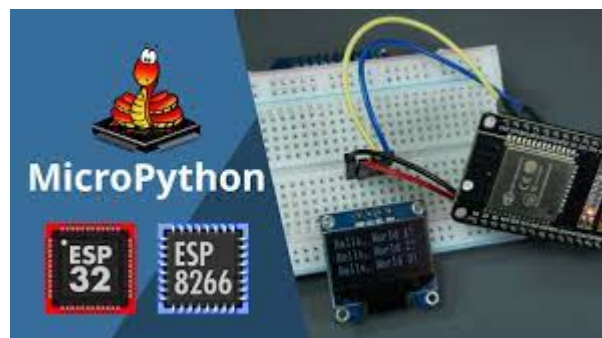




# MicroPython - Les modules Espressif ESP32 et ESP8266



[Mise à jour le : 23/7/2021]

- **Ressources**

- [Getting started with MicroPython on the ESP8266](#)
- [Getting started with MicroPython on the ESP32](#)
- [MicroPython.org](#)
- [MicroPython documentation](#)
- [IDE Thonny](#)
- Exemples de code sur [MCHobby](#)
  1. [Flashing MicroPython Firmware with esptool.py on ESP32 and ESP8266](#)
  2. [Getting Started with MicroPython on ESP32 and ESP8266](#)

- **Lectures connexes**

- **Programmez !** Juillet/Août 2019
- **Elektor 489** Mai/Jui 2021

---

## 1. Description des cartes ESPRESSIF

- Voir les Wikis Arduino
  - [La carte ESP8266 Feather Huzzah](#)
  - [La carte ESP32 Feather Huzzah](#)
  - [Les cartes ESP01\(S\)](#)

## 2. Installer MicroPython

- **Source** : [Getting started with MicroPython on the ESP32](#)

### 2.1 En ligne de commande (sous Windows)

- **Installer esptool**



esptool est un utilitaire basé sur Python, open source et indépendant de la plateforme, permettant de communiquer avec le chargeur de démarrage ROM dans les puces Espressif ESP8266 et ESP32.

Ouvrir une console et entrer la commande suivante :

\*.bash

```
pip install esptool
```

Exemple

```
CA% Invite de commandes
Microsoft Windows [version 10.0.21390.2025]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\phili>pip install esptool
Collecting esptool
  Downloading esptool-3.1.tar.gz (175 kB)
    |████████████████████████████████████████| 175 kB 1.7 MB/s
Collecting bitstring>=3.1.6
```

- **Effacer la mémoire flash**

\*.bash

```
esptool --port <portcom> erase_flash
```

Exemple

```
CA% Invite de commandes

C:\Users\phili>esptool.py --chip esp32 --port com11 erase_flash
esptool.py v3.1
Serial port com11
Connecting...
Device PID identification is only supported on COM and /dev/ serial ports.
.
Chip is ESP32-D0WDQ6 (revision 1)
```

- **Installer MicroPython**



**Télécharger** la dernière **version stable** de MicroPython pour la carte ciblée [ici](#).

Exemple

## Micrologiciel avec ESP-IDF v4.x

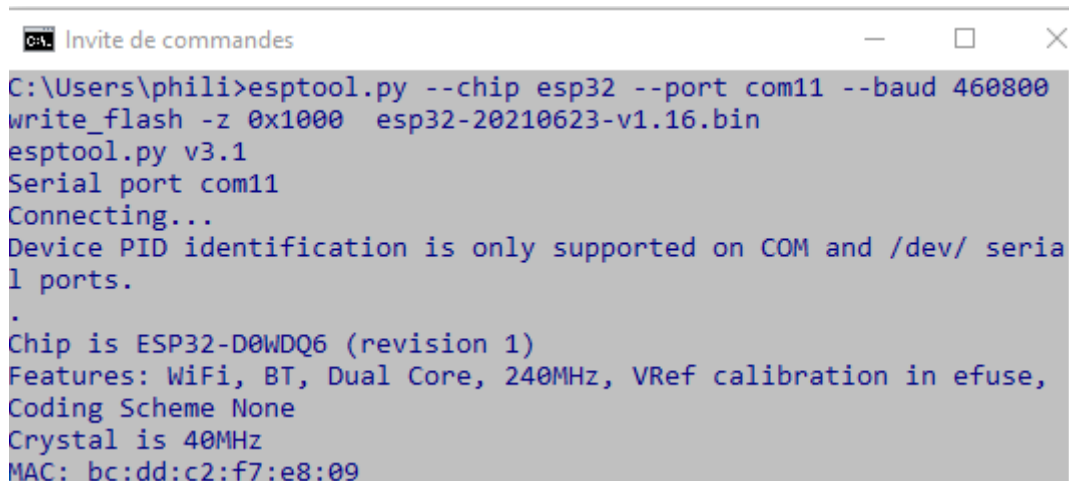
Firmware construit avec ESP-IDF v4.x, avec prise en charge de BLE et PPP, mais pas de LAN.

- GÉNÉRIQUE : [esp32-20210626-unstable-v1.16-29-gc94059731.bin](#)
- GÉNÉRIQUE : [esp32-20210625-unstable-v1.16-17-gb51ae20c0.bin](#)
- GÉNÉRIQUE : [esp32-20210624-unstable-v1.16-14-g0009a7dc3.bin](#)
- GÉNÉRIQUE : [esp32-20210624-unstable-v1.16-13-gc99e4995f.bin](#)
- GÉNÉRIQUE : [esp32-20210623-v1.16.bin](#)

\*.bash

```
esptool --port <portcom> --baud <baudrate> write_flash --  
flash_size=detect 0 <path><firmware_name>.bin
```

Exemple



```
C:\Users\phili>esptool.py --chip esp32 --port com11 --baud 460800  
write_flash -z 0x1000 esp32-20210623-v1.16.bin  
esptool.py v3.1  
Serial port com11  
Connecting...  
Device PID identification is only supported on COM and /dev/ serial  
ports.  
.  
Chip is ESP32-D0WDQ6 (revision 1)  
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse,  
Coding Scheme None  
Crystal is 40MHz  
MAC: bc:dd:c2:f7:e8:09
```

Th

## 2.2 Avec l'IDE Thonny

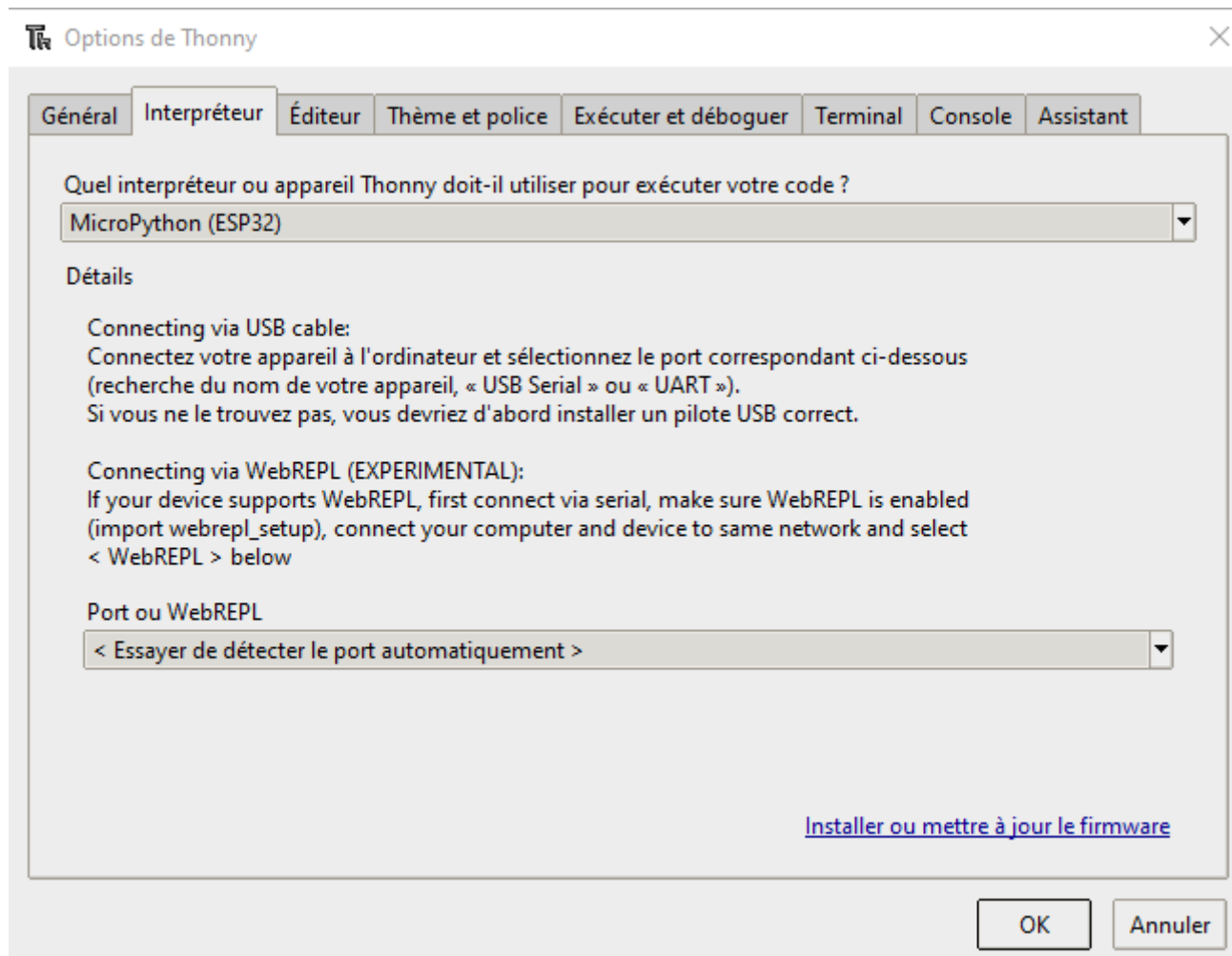
- [Thonny](#) est un environnement de développement intégré pour Python conçu pour les débutants. Il prend en charge différentes façons de parcourir le code, l'évaluation d'expression étape par étape, la visualisation détaillée de la pile d'appels et un mode pour expliquer les concepts de références et de tas.



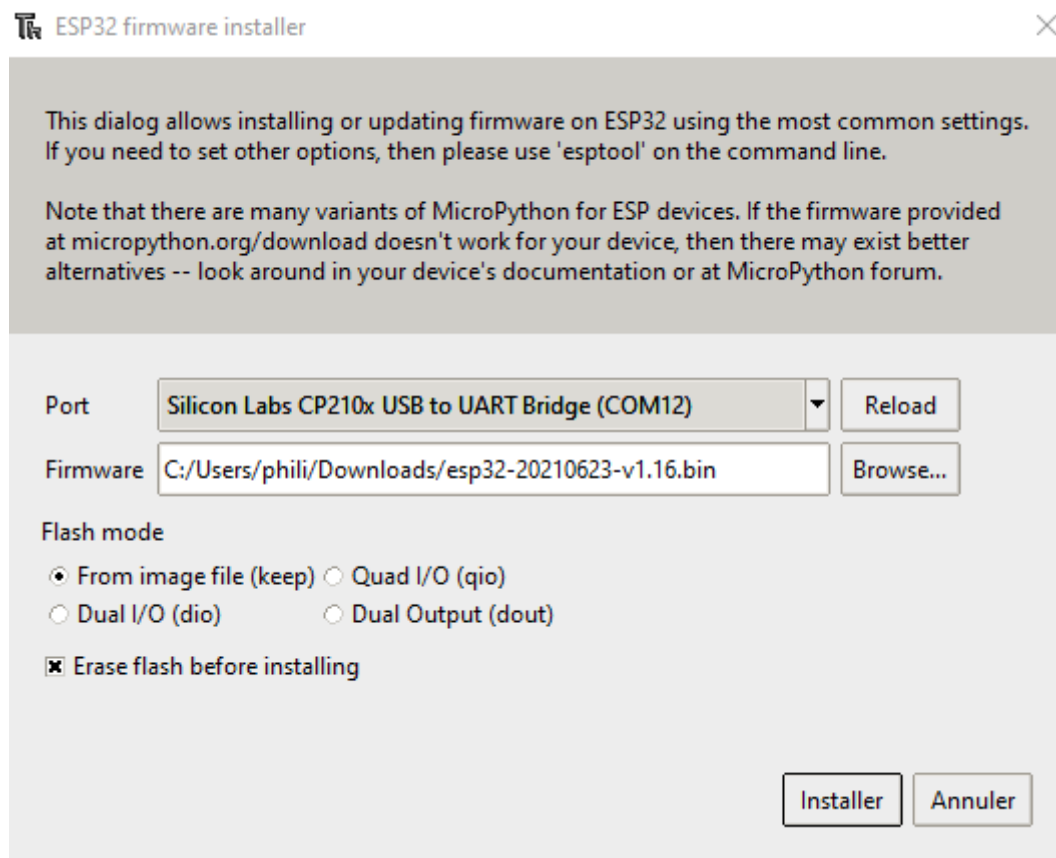
**Télécharger** la dernière **version stable** de MicroPython pour la carte ciblée [ici](#).

- **Connecter** le module au port USB du PC

- Ouvrir les options de Thonny (outils → options) et sélectionner la carte en dépliant “**Port ou WebREPL**”



- Cliquer sur “**Installer ou mettre à jour le firmware**”. Sélectionner le port et l'image préalablement chargée



### 3. Se connecter à la console REPL

- **Source** : [Getting started with MicroPython on the ESP32](#)

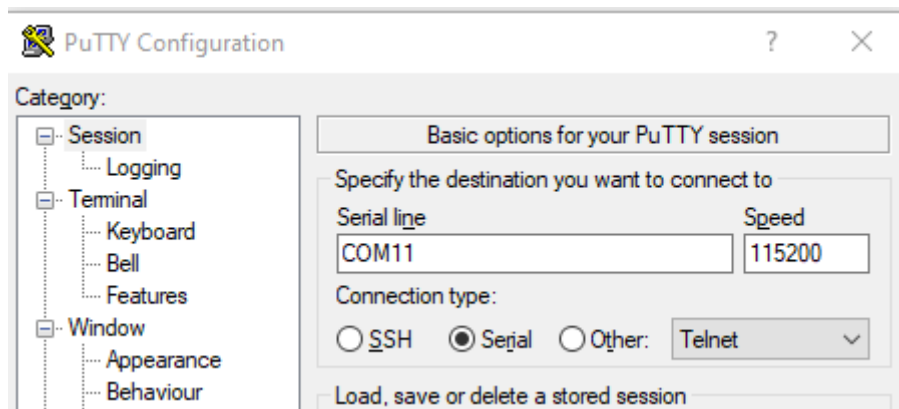


**REPL** pour **R**ead **E**val **P**rint **L**oop est un environnement de programmation informatique interactif simple qui prend les entrées d'un seul utilisateur, les exécute et renvoie le résultat; un programme écrit dans un environnement REPL est exécuté par morceaux.

#### 3.1 Connexion avec la ligne de commande (sous Windows)

- Utiliser l'utilitaire **Putty**

*Exemple*



Test

```
COM11 - PuTTY
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:5640
load:0x40078000,len:12696
load:0x40080400,len:4292
entry 0x400806b0
MicroPython v1.16 on 2021-06-23; ESP32 module with ESP32
Type "help()" for more information.
>>> █
```

### 3.2 Connexion avec l'IDE Thonny

```
Console X
MicroPython v1.16 on 2021-06-23; ESP32 module with ESP32
Type "help()" for more information.
>>>
```

## 4. Configurer le Wifi

- Source et plus loin... : [Getting started with MicroPython on the ESP32](#)
- Test avec REPL

\*.py

```
import network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
```

```
wlan.connect('MON_SSID', 'MON_PASSWD')  
print('network config:', wlan.ifconfig())
```

From:

<http://webge.fr/dokuwiki/> - **WEBGE Wikis**

Permanent link:

<http://webge.fr/dokuwiki/doku.php?id=python:micropython:materiel:espressif&rev=1685787457>

Last update: **2023/06/03 12:17**

