ESP32 et installation de µPython

Détection de l'interface USB ↔ Série ↔ ESP32

xterm
```
$ lsusb
Bus 001 Device 012: ID 10c4:ea60 Silicon Labs CP210x UART Bridge
$ ls /dev/ttyUSB*
/dev/ttyUSB0
```

Installation de l'outil pour « *flasher* »/programmer la mémoire flash de l'ESP32

xterm
```
$ python3 -m venv .venv
$ source .venv/bin/activate
$ python3 -m pip install esptool
```

Vérification et préparation de l'ESP32

On vérifie :

- ▷ la connexion de l'ESP32 ;
- ▷ la capacité de la mémoire flash ;

xterm
```
$ esptool chip-id
esptool v5.1.0
Connected to ESP32 on /dev/ttyUSB0:
Chip type:          ESP32-D0WDQ6 (revision v1.0)
Features:           Wi-Fi, BT, Dual Core + LP Core, 240MHz, Vref calibration in
eFuse, Coding Scheme None
Crystal frequency:  40MHz
MAC:                7c:9e:bd:5b:b7:c0

Stub flasher running.

Warning: ESP32 has no chip ID. Reading MAC address instead.
MAC:                7c:9e:bd:5b:b7:c0

Hard resetting via RTS pin...

$ esptool flash-id
esptool v5.1.0
Connected to ESP32 on /dev/ttyUSB0:
Chip type:          ESP32-D0WDQ6 (revision v1.0)
Features:           Wi-Fi, BT, Dual Core + LP Core, 240MHz, Vref calibration in
eFuse, Coding Scheme None
Crystal frequency:  40MHz
MAC:                7c:9e:bd:5b:b7:c0

Stub flasher running.

Flash Memory Information:
=========================
Manufacturer: c8
Device: 4017
Detected flash size: 8MB
Flash voltage set by a strapping pin: 3.3V

Hard resetting via RTS pin...
```

vérification de l'ESP32

vérification du composant mémoire

On efface le contenu de la mémoire flash :

xterm
```
$ esptool --chip esp32 --baud 460800 erase-flash
esptool v5.1.0
...
```

Installation de μPython

On récupère le firmware de μPython :

xterm
```
$ wget
https://micropython.org/resources/firmware/LILYGO_TTGO_LORA32-20251209-v1.27.0.bin
```

On flashe la mémoire avec le firmware μPython :

xterm
```
$ esptool --chip esp32 --port /dev/ttyUSB0 --baud 460800 write-flash -z 0x1000
LILYGO_TTGO_LORA32-20251209-v1.27.0.bin
esptool.py v5.1.0
...
```

flashage du firmware μPython

Connexion série vers l'ESP32

xterm
```
$ sudo apt install tio
$ tio /dev/ttyUSB0 -b 115200
```

Si vous n'avez pas les droits d'accès sur /dev/ttyUSB0 :

xterm
```
$ chmod o+rw /dev/ttyUSB0
```

Ou vous pouvez joindre le groupe possédant l'interface :

xterm
```
$ ls -la /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 Jun 14 13:56 /dev/ttyUSB0
$ sudo usermod -aG dialout $USER
```

Vous ajouterez le groupe auquel appartient le `/dev/ttyUSBx` à votre utilisateur.
Il faudra vous déconnecter/reconnecter pour joindre le groupe.

xterm
```
$ tio /dev/ttyUSB0 115200
>>>
>>>
MPY: soft reboot
MicroPython v1.27.0 on 2025-12-09; LILYGO TTGO LoRa32 with ESP32
Type "help()" for more information.
>>>
```

Appuyez sur entrée pour obtenir le prompt μPython

Ctrl-D pour relancer l'interprète μPython

Pour quitter `tio` : `<ctrl-t>-q`, `ctrl-t` est le préfixe d'ordre, `q` la commande `quit`.

Utilisation de l'outil « *ampy* »

xterm
```
$ sudo pip3 install adafruit-ampy
```

Pour lister les fichiers du « *filesystem* » du firmware µPython :

xterm
```
$ ampy -p /dev/ttyUSB0 ls -l
/boot.py - 139 bytes
/ssd1306.py - 4921 bytes
/ulora.py - 13135 bytes
/umqttsimple.py - 6446 bytes
```

Pour lancer un programme et conserver les E/S :

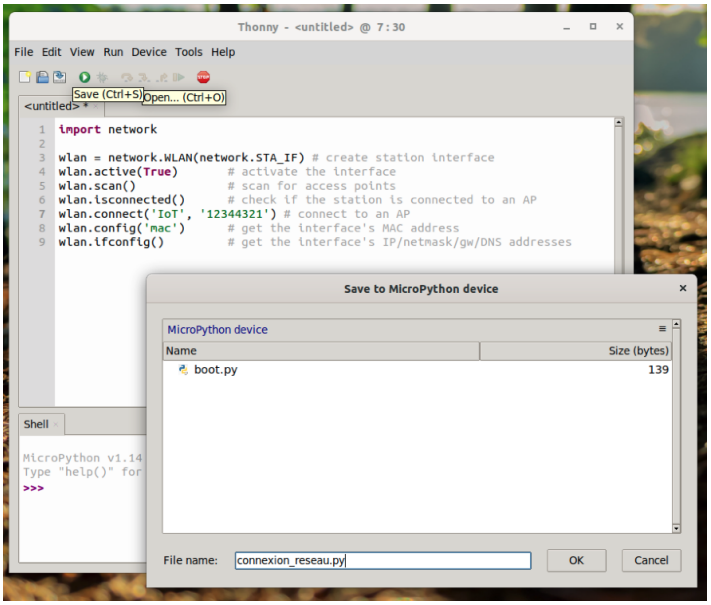
xterm
```
$ ampy -p /dev/ttyUSB0 run oled_demo_wifi.py
```

Attention

Il se peut que la commande « *ampy run* » plante et ne fournit plus les E/S de l'ESP32 : dans ce cas là le programme Python tourne toujours et vous pouvez utiliser « *tio* » pour vous y connecter.
Si vous relancez la commande `ampy ... run mon_prog.py` l'exécution est **relancée**.

Utilisation de l'IDE « *thonny* »

```
xterm
$ pipx install thonny
$ thonny
```



Cet IDE permet d'éditer un fichier, de le sauvegarder sur l'ordinateur ou l'ESP32 et d'exécuter les programmes tout en conservant leur E/S.