Faculté des Sciences & Techniques — Université de Limoges

Master 2nde année

TMC

TP n°1

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 pip install esptool
```

Vérification et préparation de l'ESP32

On vérifie :

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

```
xterm
$ esptool.py chip_id
esptool.py v3.1
Found 1 serial ports
Serial port /dev/ttyUSB0
Connecting......
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme
None
Crystal is 40MHz
MAC: 7c:9e:bd:5a:da:88
Uploading stub...
Running stub...
Stub running...
Warning: ESP32 has no Chip ID. Reading MAC instead.
MAC: 7c:9e:bd:5a:da:88
Hard resetting via RTS pin...

$ esptool.py flash_id
esptool.py v3.1
Found 1 serial ports
Serial port /dev/ttyUSB0
Connecting.....
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme
None
Crystal is 40MHz
MAC: 7c:9e:bd:5a:da:88
Uploading stub...
Running stub...
Stub running...
Manufacturer: c8
Device: 4017
Detected flash size: 8MB
Hard resetting via RTS pin...
```

vérification de l'ESP32

vérification du composant mémoire

On efface le contenu de la mémoire flash :

```
xterm
$ esptool.py --chip esp32 --baud 460800 erase_flash
esptool.py v3.1
...
```

effacement de la mémoire de l'ESP32

Installation de μPython

On récupère le firmware de μPython :

xterm
```
$ wget
https://micropython.org/resources/firmware/ESP32_GENERIC-20230426-v1.20.0.bin
```

On flashe la mémoire avec le firmware μPython :

xterm
```
$ esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800 write_flash -z 0x1000
ESP32_GENERIC-20230426-v1.20.0.bin
esptool.py v3.1
...
```

flashage du firmware μPython

Connexion série vers l'ESP32

xterm
```
$ sudo apt install screen
$ screen /dev/ttyUSB0 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.

xterm
```
$ screen /dev/ttyUSB0 115200
>>>
>>>
MPY: soft reboot
MicroPython v1.18 on 2022-01-17; ESP32 module with ESP32
Type "help()" for more information.
>>>
```

Appuyez sur entrée pour obtenir le prompt μPython

Ctrl-D pour relancer l'interprète μPython

Pour quitter `screen` : `<ctrl-a>-k-y`, `ctrl-a` est le préfixe d'ordre, `k` la commande `kill` et `y` pour confirmer.

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 :

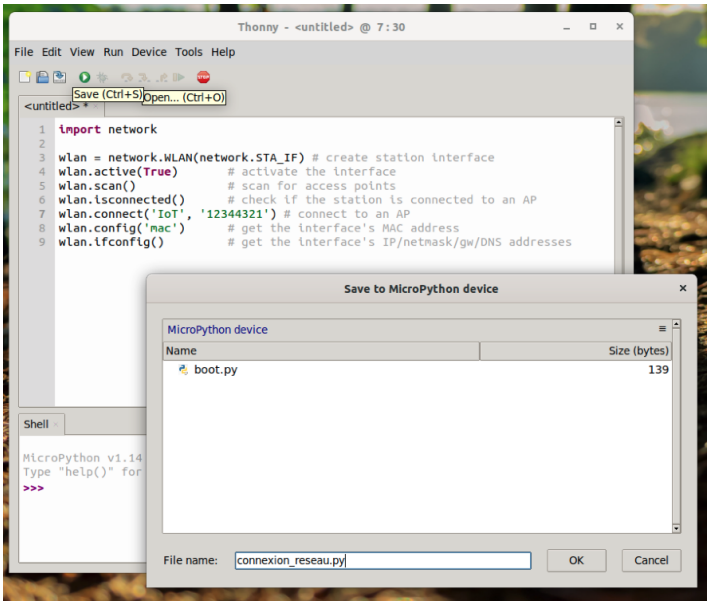
xterm
```
$ ampy -p /dev/ttyUSB0 run oled_demo_wifi.py
```

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 « *screen* » pour vous y connecter.
Si vous relancez la commande `ampy ... run mon_prog.py` l'exécution est **relancée**.

Utilisation de l'IDE « *thonny* »

xterm
```
$ sudo apt 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.