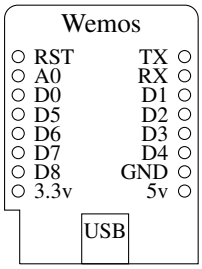
IoT et ESP8266

ESP8266 & MicroPython



Le Wemos est une carte de développement combinant :

- un composant USB ↔ série ;
- un adaptateur 5v (depuis l'USB) vers 3,3v (vers l'ESP8266) ;

On peut vérifier qu'il est connecté et que son chip série est reconnu :

xterm
```
pef@cube:~$ lsusb
...
Bus 001 Device 005: ID 10c4:ea60 Cygnal Integrated Products,
Inc. CP210x UART Bridge / myAVR mySmartUSB light
```

Installation du « *cross-compiler* »

La procédure d'installation est à l'URL `https://github.com/pfalcon/esp-open-sdk`.

xterm
```
sudo apt-get install make unrar-free autoconf automake libtool gcc g++ gperf \
flex bison texinfo gawk ncurses-dev libexpat-dev python-dev python python-serial \
sed git unzip bash help2man wget bzip2

git clone --recursive https://github.com/pfalcon/esp-open-sdk.git
make STANDALONE=y
```

Pour utiliser le compilateur, il faut l'intégrer dans son `PATH` :

xterm
```
export PATH=/home/toto/esp-open-sdk/xtensa-lx106-elf/bin:$PATH
```

Installation de micropython

La procédure est à l'URL : `https://github.com/micropython/micropython`

xterm
```
$ git clone https://github.com/micropython/micropython.git
$ cd micropython
$ git submodule update --init
$ make -C mpy-cross
$ cd ports/esp8266
```

Aller dans le répertoire dédié à l'ESP8266 et modifier le Makefile :

xterm
```
$ cd ports/esp8266
$ cat Makefile
...
FWBIN = $(BUILD)/firmware-combined.bin
PORT ?= /dev/ttyUSB0
BAUD ?= 115200
FLASH_MODE ?= dio
FLASH_SIZE ?= 32m
CROSS_COMPILE = xtensa-lx106-elf-
ESP_SDK = $(shell $(CC) -print-sysroot)/usr
...
```

xterm
```
$ make axtls
$ make
$ make erase
$ make deploy
```

Documentation Micropython : `https://docs.micropython.org/en/latest/esp8266/index.html`

Pour se connecter au micro python par le port série :

xterm
```
$ sudo screen /dev/ttyUSB0 115200
MicroPython v1.9.2-124-g2f7827ba-dirty on 2017-10-01; ESP module with ESP8266
Type "help()" for more information.
>>> import os
>>> os.uname()
(sysname='esp8266', nodename='esp8266', release='2.0.0(5a875ba)',
version='v1.9.2-124-g2f7827ba-dirty on 2017-10-01', machine='ESP module with ESP8266')
>>> import port_diag
```

Pour configurer l'ESP8266 en point d'accès

`http://docs.micropython.org/en/v1.8.7/esp8266/esp8266/tutorial/network_basics.html` :

xterm
```
>>> import network
>>> ap_if = network.WLAN(network.AP_IF)
>>> ap_if.ifconfig()
('192.168.4.1', '255.255.255.0', '192.168.4.1', '208.67.222.222')
>>> ap_if.active(True)
>>> ap_if.config(essid='toto',channel=13,authmode=network.AUTH_OPEN)
```

Pour configurer l'ESP8266 en client :

xterm
```
>>> import network
>>> sta_if = network.WLAN(network.STA_IF)
>>> sta_if.ifconfig()
('192.168.4.1', '255.255.255.0', '192.168.4.1', '208.67.222.222')
>>> sta_if.active(True)
>>> sta_if.connect('<your ESSID>', '<your password>')
```

Pour quitter il faut taper sur 'CTRL-a' puis k.

Pour utiliser MicroPython avec des programmes :

xterm
```
$ sudo -H pip install adafruit-ampy
```

Exemple de programmes

Code source du programme « `pins.py` » :

```
import machine
import time

def clignotement(p):
   while 1:
      p.off()
      time.sleep(1)
      p.on()
      time.sleep(1)


LED = 2
p=machine.Pin(LED,machine.Pin.OUT)
clignotement(p)
```

xterm
```
$ export AMPY_PORT=/dev/ttyUSB0
$ ampy run pins.py
```

La commande `ampy run -n pins.py` permet de revenir au shell directement.

Code source du programme « `simple_http.py` » :

```
import machine
pins = [machine.Pin(i, machine.Pin.IN) for i in (0, 2, 4, 5, 12, 13, 14, 15)]

html = """<!DOCTYPE html>
<html>
    <head> <title>ESP8266 Pins</title> </head>
    <body> <h1>ESP8266 Pins</h1>
        <table border="1"> <tr><th>Pin</th><th>Value</th></tr> %s </table>
    </body>
</html>
"""

import socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]

s = socket.socket()
s.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
s.bind(addr)
s.listen(1)

print('listening on', addr)

while True:
    cl, addr = s.accept()
    print('client connected from', addr)
    cl_file = cl.makefile('rwb', 0)
    while True:
        line = cl_file.readline()
        if not line or line == b'\r\n':
            break
    rows = ['<tr><td>%s</td><td>%d</td></tr>' % (str(p), p.value()) for p in pins]
    response = html % '\n'.join(rows)
    cl.send(response)
    cl.close()
```

xterm

```
$ ampy run -n simple_http.py
```