Programmation avec µPython

Installation d'un serveur, « *broker* » MQTT

Deux possibilités :

- Utilisation d'un serveur MQTT central offert par le poste enseignant :
  - WiFi : SSID : `IoT`, PSK : `12344321`
  - adresse ip serveur MQTT : `10.20.30.1`
  - utilisateur : `esp32` ;
  - mot de passe : `esp32` ;
- Utilisation d'un Raspberry Pi :
  - installation d'un point d'accès WiFi avec `hostapd` (voir fiche correspondante) ;
  - installation et configuration d'un serveur MQTT avec `mosquitto` ;

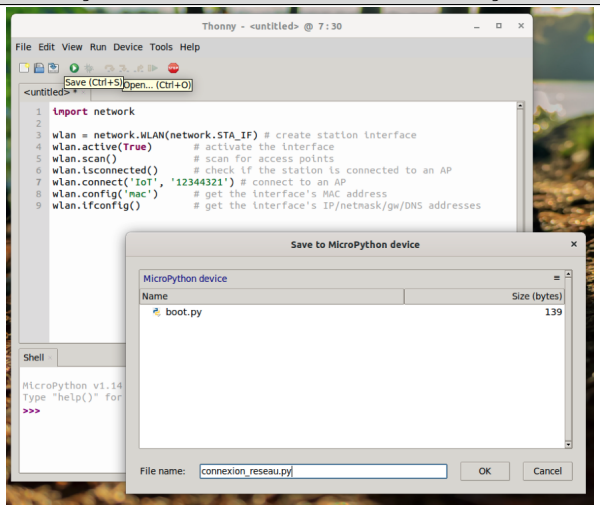
Configuration de l'ESP32 et teste de la connexion WiFi

Pour configurer l'ESP32, vous pouvez vous reporter à la fiche correspondante.

Pour tester la connexion WiFi depuis l'ESP32 :

```
import network

wlan = network.WLAN(network.STA_IF) # create station interface
wlan.active(True)       # activate the interface
wlan.scan()             # scan for access points
wlan.isconnected()      # check if the station is connected to an AP
wlan.connect('essid', 'password') # connect to an AP
wlan.config('mac')      # get the interface's MAC address
wlan.ifconfig()         # get the interface's IP/netmask/gw/DNS addresses
```



Ce qui produit sur le Raspberry Pi :

xterm

```
wlan0: STA 24:0a:c4:83:30:60 IEEE 802.11: associated
wlan0: AP-STA-CONNECTED 24:0a:c4:83:30:60
wlan0: STA 24:0a:c4:83:30:60 RADIUS: starting accounting session B5CAD8028304EFCB
wlan0: STA 24:0a:c4:83:30:60 WPA: pairwise key handshake completed (RSN)
dnsmasq-dhcp: DHCPDISCOVER(wlan0) 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPOFFER(wlan0) 10.33.33.140 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPDISCOVER(wlan0) 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPOFFER(wlan0) 10.33.33.140 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPDISCOVER(wlan0) 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPOFFER(wlan0) 10.33.33.140 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPREQUEST(wlan0) 10.33.33.140 24:0a:c4:83:30:60
dnsmasq-dhcp: DHCPACK(wlan0) 10.33.33.140 24:0a:c4:83:30:60 espressif
```

MQTT : installation et configuration du serveur sur le Raspberry Pi

xterm
```
$ sudo apt install mosquitto mosquitto-clients
```

Pour configurer le serveur mosquitto :

- ▷ modifier le fichier de configuration :

xterm
```
sudo vi /etc/mosquitto/mosquitto.conf
```

- ▷ ajouter les lignes suivantes :

```
password_file /etc/mosquitto/pwfile
port 1883
```

- ▷ relancer le serveur mosquitto :

xterm
```
$ sudo systemctl restart mosquitto.service
```

- ▷ pour vérifier que le serveur mosquitto est bien démarré :

xterm
```
$ ss -tln
State              Recv-Q     Send-Q     Local Address:Port         Peer
Address:Port
LISTEN             0          32          10.33.33.254:53                0.0.0.0:*
LISTEN             0          128              0.0.0.0:22                0.0.0.0:*
LISTEN             0          100              0.0.0.0:1883              0.0.0.0:*
LISTEN             0          128                 [::]:22                   [::]:*
LISTEN             0          100                 [::]:1883 Le serveur MQTT [::]:*
```

- ▷ pour l'ajout d'un utilisateur autorisé à s'y connecter :

xterm
```
$ sudo mosquitto_passwd -c /etc/mosquitto/pwfile esp32
```

- ▷ pour tester :
  - ◇ dans un premier shell :

xterm
```
$ mosquitto_sub -h localhost -p 1883 -u esp32 -P esp32 -t message
```

  - ◇ dans un second shell :

xterm
```
$ mosquitto_pub -h localhost -p 1883 -u esp32 -P esp32 -t message -m "hello"
```

MQTT : installation et configuration du client sur l'ESP32

Pour la récupération de la bibliothèque MQTT pour l'ESP32 :

xterm
```
$ wget
https://raw.githubusercontent.com/RuiSantosdotme/ESP-
MicroPython/master/code/MQTT/umqttsimple.py
```

Vous pouvez l'installer par thonny ou directement avec « *ampy* » :

xterm
```
$ ampy -p /dev/ttyUSB0 put umqttsimple.py
```

Un programme de démonstration sur l'ESP32 :

```
from umqttsimple import MQTTClient
import time
import machine
import ubinascii
import network

ssid = "IoT"
password = "12344321"
client_id = ubinascii.hexlify(machine.unique_id())
mqtt_server= "10.20.30.1"
user_id = "esp32"
user_pwd = "esp32"
topic_sub = "message"
topic_pub = "reponse"
last_message = 0
message_interval = 5
counter = 0

def sub_cb(topic, msg):
  print((topic, msg))
  if topic == b'notification' and msg == b'received':
    print('ESP received hello message')

def connect_and_subscribe():
  global client_id, user_id, user_pwd, mqtt_server, topic_sub
  client = MQTTClient(client_id = client_id, server = mqtt_server, port = 1883, user
= user_id, password = user_pwd)
  client.set_callback(sub_cb)
  client.connect()
  client.subscribe(topic_sub)
  print('Connected to %s MQTT broker, subscribed to %s topic' % (mqtt_server, to
pic_sub))
  return client

def restart_and_reconnect():
  print('Failed to connect to MQTT broker. Reconnecting...')
  time.sleep(10)
  machine.reset()

station = network.WLAN(network.STA_IF)
if (not station.isconnected()):
  station.active(True)
  station.connect(ssid, password)
while station.isconnected() == False:
  pass
print('Connection successful')
print(station.ifconfig())

try:
  client = connect_and_subscribe()
except OSError as e:
  restart_and_reconnect()

while True:
  try:
    client.check_msg()
    if (time.time() - last_message) > message_interval:
      msg = b'Hello #%d' % counter
      client.publish(topic_pub, msg)
      last_message = time.time()
      counter += 1
  except OSError as e:
    restart_and_reconnect()
```

Pour tester dans deux shells différents :

▷ premier shell :

```
xterm
$ mosquitto_pub -h localhost -p 1883 -u esp32 -P esp32 -t message -m "hello"
```

Vous devriez voir des messages « hello *» en sortie de l'ESP32 sur son port série.*

▷ second shell :

```
xterm
$ mosquitto_sub -h localhost -p 1883 -u esp32 -P esp32 -t reponse
```

Écrire un programme μPython utilisant le module `MQTTClient` transmettant la valeur du « *Hall sensor* » périodiquement.