Projet M5Stack Module 64-21

Light Detector

Esteban Flores

Sommaire

Projet Initial :

A quoi sert le projet ? :

Le projet consiste à allumer différents types de lumières dans une pièce (une chambre dans mon cas). J'utiliserai le détecteur dans le M5Stack comme déclencheur de la lumière. Dès que le détecteur repère un mouvement, une requête sera envoyée pour allumer les lumières ou les éteindre. Si je suis absent de chez moi, je recevrai une notification sur mon téléphone afin que je sois averti qu'une personne entre dans ma chambre.

Situations :

1. Je rentre chez moi et me dirige vers ma chambre. Au moment où je rentre dans ma chambre le détecteur détecte un mouvement et allume toutes les lumières qui sont connectées (lumière principale de la chambre, led du bureau…)
2. Je ne suis pas chez moi mais une personne rentre dans ma chambre, toutes les lumières s'allument et je reçois une notification qu'une personne rentre dans ma chambre.

Mise en œuvre :

Pour réaliser ce projet je vais utiliser 4 protocoles afin d'allumer les lumières : UDP, TCP, HTTP, MQTT.

Tout d'abord, le M5Stack enverra les données qu'il capte grâce à son capteur de mouvement en utilisant le protocole UDP. Il enverra les données à un serveur réel créé via PacketTracer. Un fois fait, le MCU dans PacketTracer va s'occuper d'allumer la lumière. En parallèle, le serveur réel envoie les données avec le Protocole TCP sur un Serveur TCP qui sera hébergé sur le MQTT Publisher. Le MQTT Publisher enverra via le protocole MQTT les données a un MQTT broker qui s'occupera lui-même de transférer via MQTT au subscriber. Le subscriber envoie en HTTP par la suite les données qu'il a récolté. Il envoie une requête HTTP grâce à l'API Telegram afin que le téléphone reçoive une notification de celle-ci.

Projet Final

Introduction :

Suite au commencement de mon projet et les différentes difficultés que j'ai rencontrées, j'ai décidé de modifier quelque peu celui-ci. La modification majeure par rapport à mon projet initial est l'envoi des données via mon M5Stack. En effet, j'envoie les données de mon M5stack en UDP vers un serveur réel UDP sur PacketTracer. Mon M5Stack envoie en même temps les données avec le protocole TCP afin que ma machine virtuelle qui sert de publisher puisse recevoir les données et envoyer en MQTT les données nécessaires pour la suite du projet. Pour ce qui est du reste, rien n'a réellement changé.

Le protocole UDP :

J'ai utilisé le protocole UDP pour envoyer les données de mon M5Stack, que j'ai codé sur une machine virtuelle sur Linux, afin de les envoyer sur un serveur réel UDP socket. Pour ce faire, j'ai d'abord créé un PacketTracer avec une lampe qui représente les lampes connectées initiales dans mon projet. Je branche cette lampe avec un iot « custom cable » à un iot « MCU-PT MCU0 ».

Dans l'iot j'ai créé un serveur réel socket udp. Cela m'a fourni un code dans lequel j'ai ajouté certaines lignes de code. Dans la méthode onUDPReceive() :

```
if data == "1":
        customWrite(0,1)
else:
        customWrite(0,0)
```

date représente les données envoyées du M5Stack. La suite du code permet de faire en sorte que si la donnée = à 1, alors la lumière connectée sur PacketTracer s'allume.

Concernant le code d'envoi, j'ai installé une machine virtuelle afin d'utiliser thonny. J'ai mis le code d'un client UDP ainsi que le code pour faire fonctionner le capteur de mouvement. Grâce à cela, les données du capteur de mouvement sont envoyées à mon serveur réel UDP et celui-ci peut traiter les données comme expliqué ci-dessus.

```
def udp(mouv):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.sendto(str((mouv)),("172.20.10.3", 1235))
    s.close()
```

Le protocole TCP :

J'ai utilisé le protocole TCP afin de fournir les donnée du M5Stack directement à la machine virtuelle tiny qui a comme rôle de publisher. J'ai décidé de délivrer les données en TCP directement à la machine virtuelle et de ne pas passer par PacketTracer car j'ai eu pas mal de problèmes. En effet, je n'arrivais pas à envoyer depuis PacketTracer les données, dont j'avais besoin, avec le protocole TCP à la machine publisher. Donc avec l'aide de l'assistant David Roch, j'ai décidé d'envoyer directement depuis le thonny qui contient le code du M5Stack pour le protocole UDP les données avec également le protocole TCP. Cela permet donc à mon publisher de recevoir les données du M5Stack via TCP et donc de les traiter pour ensuite les publier en MQTT au subscriber.

Pour faire tout cela, j'ai créé une fonction tcp(mouv). Cette fonction va se connecter au serveur TCP puis envoyer les données du m5Stack au serveur.

```
def tcp(mouv):
    socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socket.connect(("172.20.10.5", 1234))
    socket.sendall(str(mouv))
    socket.close()
```

Pour le serveur TCP, j'ai pris le code TCP socket dans le tp n° 3. Je l'ai mis dans un fichier python sur la machine virtuelle tiny qui sert de publisher. Cela permet donc d'exécuter le fichier et de lancer le serveur TCP et d'utiliser les données reçues afin de publier des données en MQTT.

Le protocole MQTT :

Pour commencer, j'ai d'abord créé la machine virtuelle tiny nommé « publisher » dans laquelle j'ai créé un dossier python afin de coder le programme. Le programme permet de publier un message à tous ceux qui se sont « sub » au topic que j'ai créé. Le code pour publier est dans le même ficher que le serveur TCP.

```
TCPServerSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
TCPServerSocket.bind(("172.20.10.5", 5656))
TCPServerSocket.listen()
```

Pour créer le topic, j'ai créé une nouvelle machine virtuelle tiny. Grâce à la ligne de commande ci-dessous, j'ai pu créer le broker qui va permettre de faire communiquer ma machine « subscriber » et ma machine « publisher ».

```
tc@box:~$ sudo mosquitto --daemon --config-file /usr/local/etc/mosquitto.conf
```

Enfin, j'ai créé une dernière machine virtuelle afin de créer le « subscriber ». Pour cela j'ai également créé un fichier python qui contient à une ligne près le même code que le publisher. Ce code permet de recevoir le message du publisher et permettra d'utiliser les données afin de le transmettre via le protocole HTTP pour envoyer une requête Telegram.

```
broker_address = "172.20.10.2"
print("creation de la nouvelle instance")
client = mqtt.Client("este")
client.on_message=on_message
print("connection au broker")
client.connect(broker_address)
client.loop_start()
print("Inscription au topic", "Light")
client.subscribe("Light")
print("test")
```

Le protocole HTTP :

J'ai utilisé le protocole HTTP pour envoyer une requête Telegram afin de recevoir une notification lorsque que la lumière s'allume. On peut aussi utiliser cette notification juste pour être averti si une personne rentre dans la pièce. Pour cela, j'ai juste ajouté deux lignes de code dans le fichier python que contient la machine du subscriber. Pour que cela marche, j'ai créé un bot Telegram afin qu'il reçoive la requête et envoie le message.

```
        requete = "https://api.telegram.org/bot5355618925:AAHBEJGCrBwZL_bY_KDoLi
qVIV2WIBJUWMk/sendMessage?chat_id=538957986O&text=Votre%20Lumiere%20est%20Allume
r"
        res = requests.get(requete)
```

Les difficultés que j'ai rencontrées :

J'ai rencontré quelques difficultés lors de la réalisation de mon projet. Tout d'abord, comme mentionner plus haut, je n'ai pas pu faire mon projet d'origine avec le schéma que j'avais présenté car je n'ai pas réussi à envoyer les données reçues en UDP, en TCP. J'ai donc dû trouver une alternative avec l'assistant David pour palier à ce problème.

Cela m'a conduit à une autre difficulté. J'ai en effet eu du mal à envoyer les données du M5Stack en simultané en UDP et en TCP. J'ai donc dû créé des fonctions afin que cela fonctionne.

Ensuite lors de la mise en place du protocole MQTT, je ne comprenais pas trop comment faire fonctionner cela car nous n'avions malheureusement pas pu voir en profondeur ce protocole lors du tp à cause d'un contrôle d'un autre module. J'ai donc demandé à mon enseignant afin qu'on puisse rentrer plus en détails ce qui ma permis d'y voir plus clair.

Conclusion :

Ce projet m'a permis de mieux comprendre le fonctionnement de ces 4 protocoles. Que ce soit en les travaillant lors de la réalisation du projet ou encore durant les cours théoriques et pratiques. Le fait de travailler en autonomie m'a beaucoup plu car cela nous fait découvrir les différentes méthodes, mais également de mieux comprendre le fonctionnement de toute la réalisation.

Sources :

- Ciaran Bryce
- David Roch

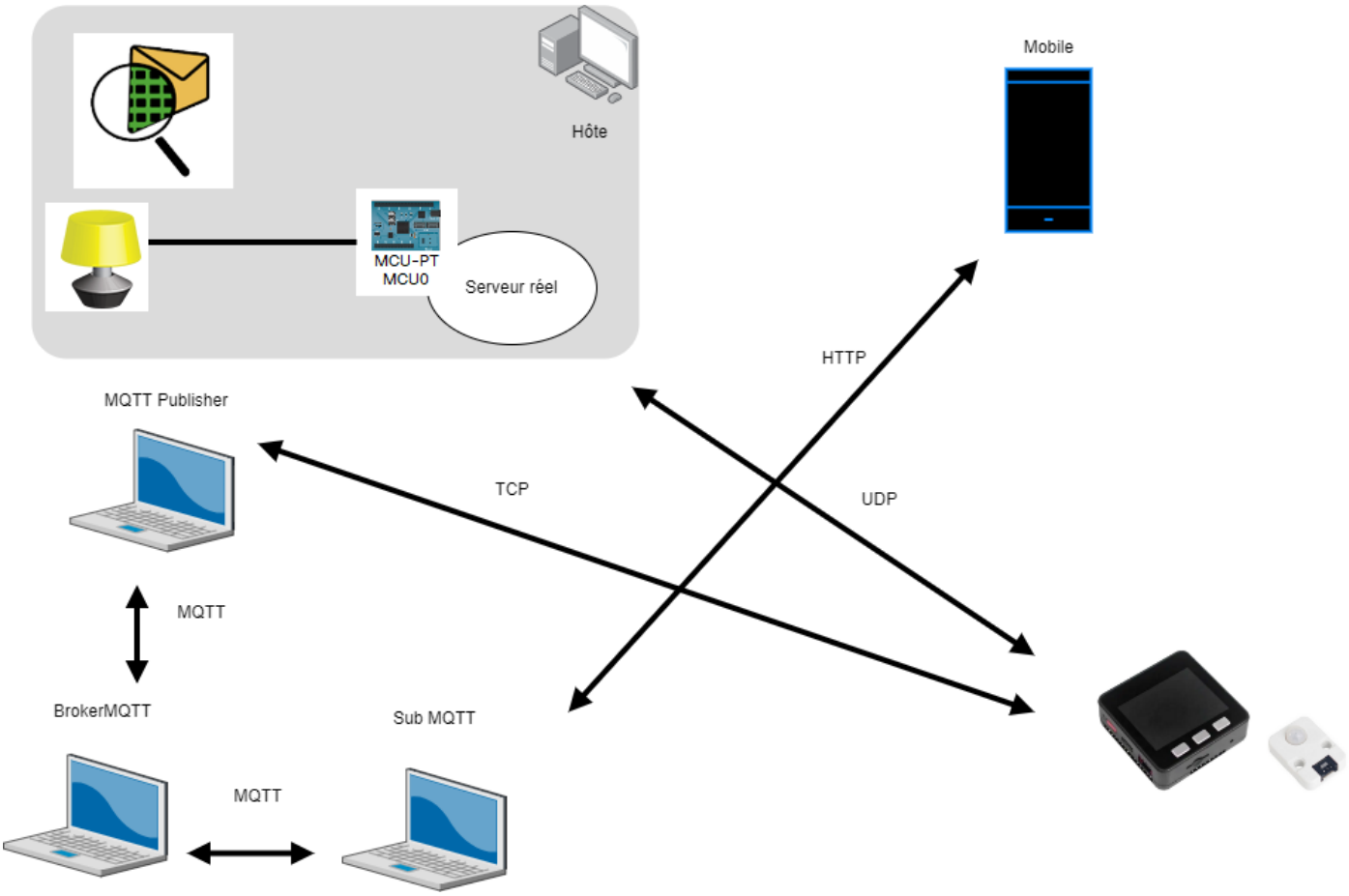
Ce sont les personnes qui m'ont aidé lors de la réalisation de mon projet.

API Télégram : https://core.telegram.org/

CyberLearn pour les codes dans le TP : https://cyberlearn.hes-so.ch/course/view.php?id=19772

Annexes

Schéma final :



Code client TCP et UDP + capteur Mouvement

```
#import usocket as socket
from m5stack import *
from time import sleep
import socket
import network
import machine
import time

AUTH_OPEN = 0
AUTH_WEP = 1
AUTH_WPA_PSK = 2
AUTH_WPA2_PSK = 3
AUTH_WPA_WPA2_PSK = 4
SSID = "Este"
PASSWORD = "qwertzuio"

def do_connect(ssid,psw):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    s = wlan.config("mac")
    mac = ('%02x:%02x:%02x:%02x:%02x:%02x').upper() %(s[0],s[1],s[2],s[3],s[4],s[5])
    print("Local MAC:"+mac) #get mac
    wlan.connect(ssid, psw)
    if not wlan.isconnected():
        print('connecting to network...' + ssid)
        wlan.connect(ssid, psw)
    start = time.ticks_ms()
    while not wlan.isconnected():
        time.sleep(1)
        if time.ticks_ms()-start > 20000:
            print("connect timeout!")
            break
    if wlan.isconnected():
        print('network config:', wlan.ifconfig())
    return wlan
def connect():
    do_connect(SSID,PASSWORD)

def udp(mouv):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.sendto(str((mouv)),("172.20.10.3", 1235))
    s.close()

def tcp(mouv):
    socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socket.connect(("172.20.10.5", 1234))
```

```
    socket.sendall(str(mouv))
    socket.close()



def app():
    connect()
    pinin = machine.Pin(22, machine.Pin.IN)
    mouvement = pinin.value()
    while True:
        udp(mouvement)
        tcp(mouvement)
        sleep(5)

app()
```

Code Serveur TCP + Publisher MQTT

```
import socket
import paho.mqtt.client as mqtt
import time
def on_message(client, userdata, message):
        print("msg recu ", str(message.payload.decode("utf-8")))
        print("msg topic ", message.topic)
        print("msg qos ", message.qos)
        print("msg retain flag =", message.retain)
TCPServerSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
TCPServerSocket.bind(("172.20.10.5", 5656))
TCPServerSocket.listen()
while True:
        print("Serveur TCP = ON")
        print("Atente connexion")
        connexion, addr = TCPServerSocket.accept()
        data = connexion.recv(1024).decode("UTF-8")
        print(data)
        if data == "1":
                broker_address = "172.20.10.2"
                client = mqtt.Client("Este")
                client.on_message=on_message
                client.connect(broker_address)
                client.loop_start()
                client.subscribe("Light")
                client.publish("Light", "Lumiere allumee")
                client.loop_stop()
        connexion.close()
        print("Connexion closed")
TCPServerSocket.close()
```

Code Subscriber + requête telegram

```
import paho.mqtt.client as mqtt
import time
import requests

def on_message(client, userdata, message):
        print("msg recu", str(message.payload.decode("utf-8")))
        print("msg topic =", message.topic)
        print("msg qos=", message.qos)
        print("msg retain flag =", message.retain)

        requete = "https://api.telegram.org/bot5355618925:AAHBEJGCrBwZL_bY_KDoLi
qVIV2WIBJUWMk/sendMessage?chat_id=5389579860&text=Votre%20Lumiere%20est%20Allume
r"
        res = requests.get(requete)


broker_address = "172.20.10.2"
print("creation de la nouvelle instance")
client = mqtt.Client("este")
client.on_message=on_message
print("connection au broker")
client.connect(broker_address)
client.loop_start()
print("Inscription au topic", "Light")
client.subscribe("Light")
print("test")

fin = False
while fin == False:
        time.sleep(5)
client.loop_stop()
```

Code serveur UPD + allumage de la lumière

```
from realudp import *
from time import *
from gpio import *

IP = "172.20.10.2"
PORT = 1235

def onUDPReceive(ip, port, data):
    print("received from "
        + ip + ":" + str(port) + ":" + data);
    if data == "1":
        customWrite(0,1)
    else:
        customWrite(0,0)

def main():
    customWrite(0,0)
    socket = RealUDPSocket()
    socket.onReceive(onUDPReceive)
    print(socket.begin(1235))

    count = 0
    while True:
        count += 1
        # s0t.send(IP, PORT, "hello " + str(count))
        sleep(1)

if __name__ == "__main__":
    main()
```