

Middleware for the IoT: TP2



LIEVRE Agathe
NGUYEN Assia

5ISS

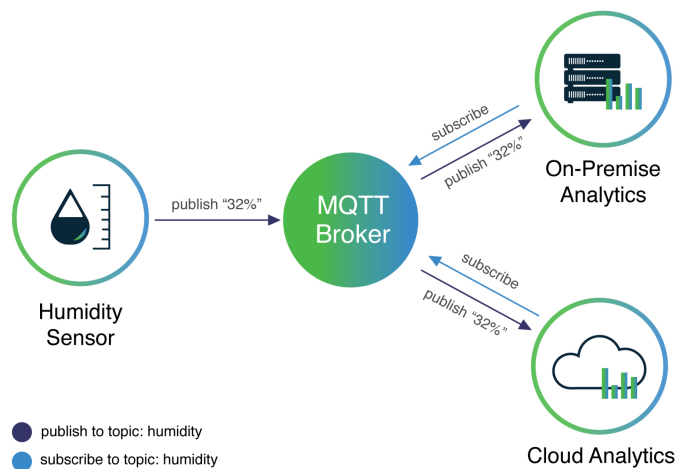
2021/202

1. Objective

The goal of this lab is to explore the capabilities of the MQTT protocol for IoT. To do so, we did a little bit of research about MQTT. Then, we installed the software we needed to use MQTT. Finally, we developed an application using the MQTT protocol and an ESP8266 board.

2. MQTT

- What is the typical architecture of an IoT system based on the MQTT protocol?



The MQTT is a publish/subscribe messaging protocol for lightweight M2M communications. In an IoT system, MQTT has a client/server model, where sensors are considered clients and connect to a server called a broker. A broker can be installed on any machine in the cloud. Any number of clients or end devices can subscribe to the topics or events by sending "publish" messages to the broker. event with the broker. When a parameter changes or an event occurs, the broker notifies the subscribed clients.

- What is the IP protocol under MQTT? What does it mean in terms of bandwidth usage, type of communication, etc ?

MQTT relies on the TCP/IP protocol which means the client and the broker need to have a TCP/IP stack. Its lightweighness allows it to be implemented on both heavily constrained device hardware as well as high latency (~120 milliseconds)/limited bandwidth networks. The communication has to be bi-directional, ordered and lossless.

- What are the different versions of MQTT?

There are two different variants of MQTT and several versions: MQTT v3.1.0, MQTT v3.1.1 (in common use), MQTT v5 (currently in limited use) and MQTT-SN. On one hand, the original MQTT was created in 1999, has been in use for many years and is designed for TCP/IP networks. MQTT v3.1.1 is the common use version and there is very little difference between v3.1.0 and v3.1.1. The v5 has been approved in January 2018. On the other hand, MQTT-SN was specified in 2013 and designed to work with UDP, ZigBee, etc. This variant is not very popular and has not changed for several years but is expected to change for the IoT deployment.

- What kind of security/authentication/encryption are used in MQTT?

Security: in the MQTT protocol, security is divided in different layers. Each layer has its own role to prevent different kinds of attacks. The protocol itself only specifies a few security mechanisms. But you can commonly use other state-of-the-art security standards like, for example, SSL/TLS for transport security.

Authentication: the MQTT protocol allows authentication with username and password. It provides username and password fields in the CONNECT message for authentication. Therefore, the client has the option to send a username and password when it connects to an MQTT broker. In addition to that, MQTT clients provide other information that can be used for authentication: with its unique client identifier or with a X.509 certificate (which is a digital certificate that uses a public key infrastructure to verify that a public key belongs to a client).

Encryption: MQTT payload encryption is the encryption of application-specific data on the application level. All MQTT PUBLISH metadata stays intact and only the payload of the message gets encrypted. Usually, the encryption is only applied to MQTT PUBLISH packets but it is possible to implement a custom broker plugin for decryption so that PUBLISH topics, CONNECT username/password, SUBSCRIBE topic or UNSUBSCRIBE topic can be encrypted. There are two popular mechanisms for encrypting data: Asymmetric encryption and Symmetric encryption. Asymmetric encryption requires two keys: One (public) key for encrypting data and one (private) key for decrypting data. Once data is encrypted with the public key, it is not possible to retrieve the original message with the public key. Only the private key can decrypt the data. Whereas, for Symmetric encryption, it

is possible to encrypt and decrypt a message with the same key (which can be a password).

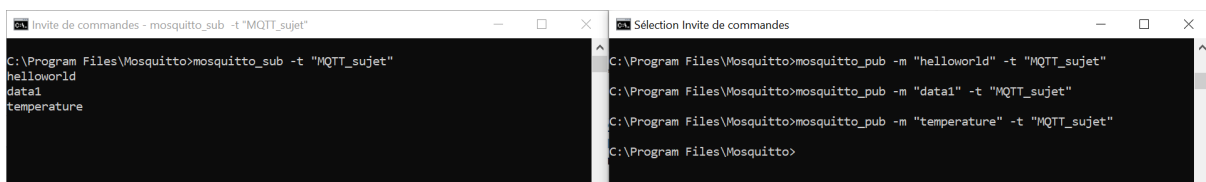
- Suppose you have devices that include one button, one light and luminosity sensor. You would like to create a smart system for you house with this behavior:
 - You would like to be able to switch on the light manually with the button
 - The light is automatically switched on when the luminosity is under a certain value

What different topics will be necessary to get this behavior and what will the connection be in terms of publishing or subscribing?

We will need 2 topics : luminosity (sensor publishes a topic "luminosity and an app subscribes to it and calculates when to switch the light based on the luminosity) and button state (app subscribes and turns on/off the light based on the state of the button).

3. Install and test the broker

- We install the broker.
- We run the mosquitto broker.
- We tested mosquitto_sub and mosquitto_pub by creating a topic and publishing some messages on this topic.



```
Invite de commandes - mosquitto_sub -t "MQTT_sujet"
C:\Program Files\Mosquitto>mosquitto_sub -t "MQTT_sujet"
helloworld
data1
temperature

Sélection Invite de commandes
C:\Program Files\Mosquitto>mosquitto_pub -m "helloworld" -t "MQTT_sujet"
C:\Program Files\Mosquitto>mosquitto_pub -m "data1" -t "MQTT_sujet"
C:\Program Files\Mosquitto>mosquitto_pub -m "temperature" -t "MQTT_sujet"
C:\Program Files\Mosquitto>
```

4. Creation of an IoT device with the nodeMCU board that uses MQTT communication

- a. Give the main characteristics of nodeMCU board in term of communication, programming language, Inputs/outputs capabilities.

NodeMCU boards have built-in Wi-Fi/Bluetooth features. It supports UART, SPI and I2C interface. We can program it with Arduino. There are also GPIO pins and an analog pin.

- b. We have the Arduino IDE.
- c. We add the board NodeMCU 09 (ESP-12 module)

- d. We add the library (and necessary dependency) ArduinoMqtt by Oleg Kovalenko
- e. Based on examples in the library we build our own application
 - We open the file in the menu: exemples/arduinoMqtt/connectESP8266wificlient and have a look at the different parts of the code.

There are 3 different parts. The first part is the definition of a class of object to supply system functions. The two functions are millis() and yield(). The second part is the setup of the objects. It sets up the wifi connection and the MQTT client. The last part is the main loop. It checks the connection status, starts a new MQTT connection and adds a “subscribe” or a “publish” if needed.

- We modify the network characteristics to connect to the local wifi network
- We modify the address of the MQTT server (IP address : ipconfig) to be able to connect to the broker on our laptop (remember that our laptop should be on the same wifi network)
- We open the serial monitor on Arduino IDE and run our Arduino code, we validate that the MQTT connection is done between our device and the broker.

We cannot connect to mosquitto because mosquitto is connected and executed in local mode. We have to change the mosquitto.conf file and add two lines: “listener 1883 0.0.0.0” and “allow_anonymous true”. When running mosquitto, we have to add the new configuration “mosquitto -c ../mosquitto.conf”.

```
Adresse IPv6 de liaison locale. . . . : fe80::28e0:81ef:f9c5:b9b9%7
Adresse IPv4. . . . . : 172.20.10.2
Masque de sous-réseau. . . . . : 255.255.255.240
Passerelle par défaut. . . . . : 172.20.10.1

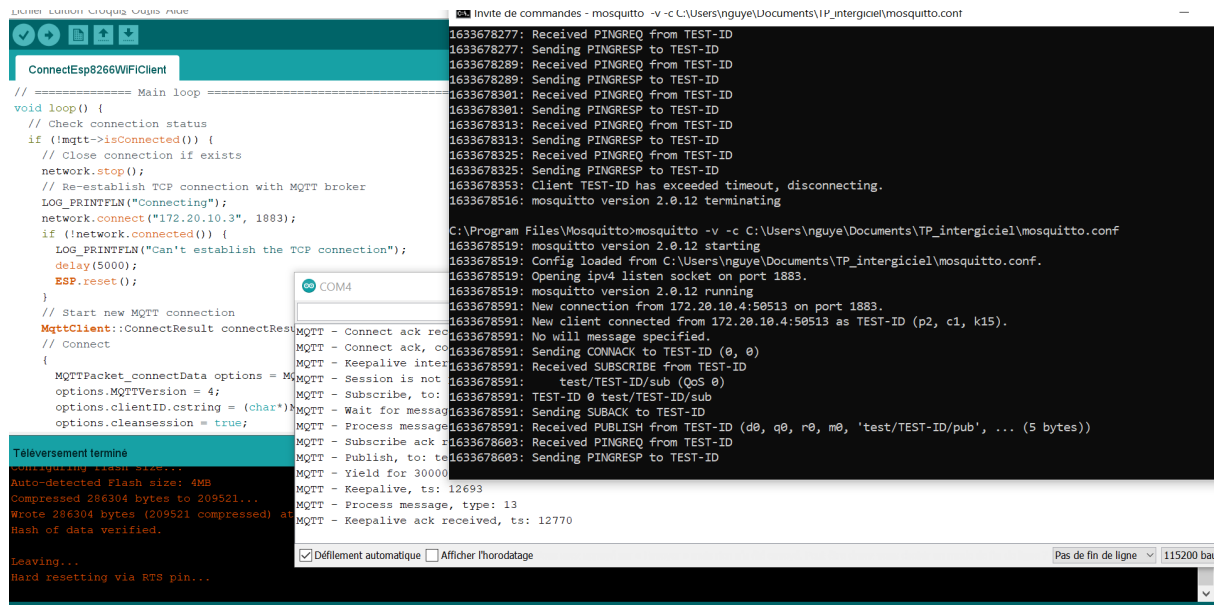
Carte Ethernet Connexion réseau Bluetooth :

Statut du média. . . . . : Média déconnecté
Suffixe DNS propre à la connexion. . . :

:\Program Files\Mosquitto>mosquitto -c C:\Users\nguye\Documents\TP_intergiciel\mosquitto.conf

:\Program Files\Mosquitto>mosquitto -v -c C:\Users\nguye\Documents\TP_intergiciel\mosquitto.conf
633677976: mosquitto version 2.0.12 starting
633677976: Config loaded from C:\Users\nguye\Documents\TP_intergiciel\mosquitto.conf.
633677976: Opening ipv4 listen socket on port 1883.
633677976: mosquitto version 2.0.12 running
```

- f. We add a publish/subscribe behavior in our device
 - We open the file in menu: exemples/arduinoMqtt/PubSub
 - We extract the specific part: publish and subscribe and necessary declaration and put that in the previous example
 - We open the serial monitor on Arduino IDE and run our Arduino code, using the command mosquitto_pub and mosquitto_sub validate that our device publishes values and can receive the result of subscription



5. Creation of the application

By using what we did on MQTT and the necessary sensors and actuators, we program the application's light management behavior through MQTT exchanges.

We will use the NodeMCU LED GPIO16, a light sensor on A0 and a button on D3. When the luminosity is under a limit of 500, the light turns on. We can see that we successfully subscribe and publish to the 2 topics "luminosity" and "button-state". When we receive the message, we analyze the payload and depending of the message, we turn on or off the LED.

