

## 1. Exploitation d'un serveur MQTT

### 1.1 Installation d'un serveur (broker) local

Dans un environnement linux, il suffit d'utiliser le gestionnaire de paquet de la distribution. Par exemple pour Ubuntu :

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

Tester l'installation :

- Dans un terminal s'abonner à un topic *test* :

```
$ mosquitto_sub -v -t test
```

- Dans un autre terminal publier sur le topic *test* :

```
$ mosquitto_pub -m hello -t test
```

- Vérifier que l'accès est possible depuis une autre machine en rajoutant l'option :

```
-h ip_du_serveur1
```

- Vérifier qu'en publiant un message avec le mode *retain* (option **-r**), le message est mémorisé.

Utilisation des "wilcards" **#** et **+** :

- **#** permet d'écouter toute la sous-arborescence qui suit<sup>2</sup>. Par exemple

```
$ mosquitto_sub -t espGEII/# -v
```

donne accès à tous les topics commençant par *espGEII/*

- **+** permet de remplacer un niveau de la sous-arborescence. Par exemple

```
$ mosquitto_sub -t /+/temperature -v
```

donne accès à toutes les températures (publiées après un seul niveau).

Sur un téléphone *Android* télécharger l'application "MQTT Dash" :

<https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=fr>



- Connectez votre téléphone dans le réseau WiFi de la salle.

- Vérifier que vous savez publier et recevoir des messages sur le serveur installé sur votre machine.

---

<sup>1</sup> Ou nom DNS du serveur si on a accès à un DNS.

<sup>2</sup> Si le broker le permet, pour écouter tout le trafic il faut "échapper" le **#** :

```
mosquitto_sub -h ip_du_server -v -t \# (ou -t '#')
```

## 1.2 Utilisation d'un broker extérieur

Pour une application permanente il faut soit installer son propre serveur avec accès à internet, soit utiliser un service mqtt payant. Pour faire des test, on peut utiliser un broker gratuit sur internet :

broker.hivemq.com , test.mosquitto.org , mqtt.eclipse.org , ...

**Attention le proxy de la salle bloque le port utilisé par mqtt (1883 par défaut), il faudra soit utiliser le wifi de l'Université soit votre connexion 4G.**

- Vérifier que vous pouvez publier et recevoir un message sur un broker gratuit en ligne de commande.
- Vérifier que vous pouvez publier et recevoir un message sur un broker gratuit avec votre téléphone.

## 1.3 Exploiter les données de la salle de TP

Dans la salle de TP des valeurs de capteurs sont publiés toutes les deux secondes. L'adresse du broker mqtt vous sera communiquée par l'enseignant.

Schéma des topics capteurs (xxyy sont les deux derniers octets de l'adresse MAC de l'ESP, il est important d'identifier correctement l'ESP pour éviter la confusion lorsque plusieurs programmes identiques connecté au même broker)

| Topic                   | Format des messages |
|-------------------------|---------------------|
| espGEIIXxyy/temperature | 21.4                |
| espGEIIXxyy/humdity     | 62.2                |
| espGEIIXxyy/light       | 243                 |
| espGEIIXxyy/ircode*     | 0xFA15              |

\*Ce topic est associé au code reçu par une télécommande infrarouge de manière à simuler des données asynchrones.

- En ligne de commande, abonnez vous à tous les topics pour découvrir le nom de base effectif des topics (espGEIIXxyy).

Il y a également des "actionneurs" : ici un relais, la led bleue (antenne) et une led RGB.

Quand on commande un actionneur il est important de s'assurer que la commande a bien été exécutée<sup>3</sup>. Pour cela il y a deux topics par actionneur :

- un topic **set** qui permet de commander l'actionneur,
- un topic **state** qui permet de connaître l'état courant de l'actionneur.

L'objet connecté va s'abonner au topic **set** de chaque actionneur. S'il reçoit une commande valide sur ce topic, il exécute cette commande et publie le nouvel état de l'actionneur sur le topic **state** correspondant. En l'absence de commande, il publie périodiquement l'état de l'actionneur sur la topic **state**<sup>4</sup>.

<sup>3</sup> Avec les qualités de service (QOS) de MQTT on peut s'assurer que le message a bien été transmis mais on ne pas être certain que la commande a bien été exécutée. Seul un retour fonctionnel permet de s'en assurer.

<sup>4</sup> On peut aussi utiliser le mode *retain* pour qu'un client qui vient de s'abonner soit informé de l'état sans attendre la publication périodique.

Schéma des topics actionneurs (xxyy sont les deux derniers octets de l'adresse MAC de l'ESP)

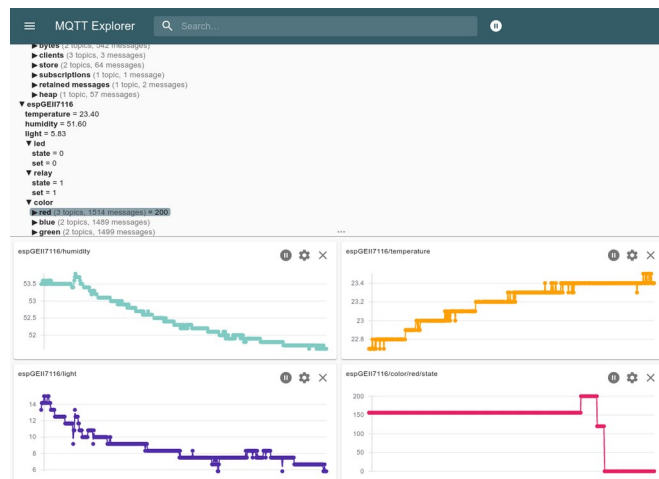
| Topic                     | Format des messages      |
|---------------------------|--------------------------|
| espGEIIXxyy/led/set       | 0 , 1                    |
| espGEIIXxyy/led/state     | 0 , 1                    |
| espGEIIXxyy/relay/set     | 0 , 1                    |
| espGEIIXxyy/relay/state   | 0 , 1                    |
| espXXYY/color/red/set     | rrr taux de r en décimal |
| espXXYY/color/red/state   | rrr                      |
| espXXYY/color/green/set   | ggg taux de g en décimal |
| espXXYY/color/green/state | ggg                      |
| espXXYY/color/blue/set    | bbb taux de b en décimal |

- En ligne de commande, changer l'état des actionneurs et assurez vous que la commande a bien été prise en compte en surveillant le message d'état correspondant.

## 1.4 Dashboard Graphique

Un "dashboard" (tableau de bord) est une application qui permet de surveiller et d'interagir avec les données qui transitent sur un serveur MQTT.

- Installez l'application MQTT Explorer
- Tracez les graphiques de la lumière de la température et de l'humidité.
- Modifier l'état des actionneurs à partir de l'application.



## 2. Programmation MQTT pour ESP8266

Installer la librairie PubSubClient :

<https://github.com/knolleary/pubsubclient>

Cette librairie permet de publier et de s'abonner à un serveur MQTT. La taille totale des messages, header+topic+message, est de 128 octets par défaut. Comme le header est sur 5 octets maximum, il reste au minimum 123 octets pour le topic **et** le message.

### 2.1 Analyse d'un exemple

On utilise l'exemple `mqtt_esp8266` de la librairie.

Modifier l'exemple pour accéder au serveur mqtt de votre machine.

Visualisez les messages publiés.

Modifier l'état de la led de l'ESP avec le message approprié.

**Il s'agit maintenant de bien comprendre cet exemple avant de commencer à programmer.**

- Comment crée-t-on l'objet client ?
- Comment sélectionner le serveur sur lequel on veut se connecter ?
- Décrire la procédure de connexion. Comment sait-on que la connexion est établie ?
- Comment faire pour publier un message sur un topic ?
- Comment faire pour s'abonner à un topic ? Quand doit-on s'abonner ?
- Que se passe-t-il lorsqu'un message est reçu sur un topic auquel on est abonné ?
- Quand et comment enregistre-t-on la fonction *callback* ?
- Expliquer le prototype de la fonction *callback* :

```
void callback(char* topic, byte* payload, unsigned int length) ;
```

### 2.2 Un premier programme

En vous inspirant de l'exemple précédent, écrire un programme qui :

- se connecte au serveur de la salle,
- affiche sur le terminal la température et l'humidité de la salle,
- publie "ON" lorsqu'on appuie sur le bouton flash (D3) et "OFF" quand on relâche,
- allume ou éteint la led antenne sur réception d'un message.

Adapter votre programme pour commander la led d'un camarade lors de l'appui sur le bouton de votre ESP.

**Attention** : Les fonctions *publish* et *subscribe* n'acceptent que les chaînes C. Si on dispose d'une String arduino il faut la convertir en chaîne C par la méthode `c_str()` pour la passer à *publish* ou *subscribe*.

Exemple :

```
String message = "ON" ;  
String topic = "/home/light/state" ;  
client.publish( topic.c_str(), msg.c_str() );
```

**Attention** : Il faut choisir un topic "personnel" pour la led et le bouton pour éviter une confusion avec les messages des camarades. Un moyen efficace de le faire est de préfixer le topic par l'adresse mac de votre esp mais à ce stade un choix personnel suffit pour ne pas compliquer.