

Systèmes interconnectés – SINT

Sujet de TP 6 : MQTT

Centrale Nantes

P.-E. Hladik, pehladik@ec-nantes.fr

—
Version 1.0.1 (8 décembre 2022)



1 Objectifs pédagogiques

- comprendre et utiliser MQTT depuis un PC et un micro-contrôleur
- déployer un *broker* MQTT, projet Paho

2 Rendu

À la fin du TP déposez sur Hippocampus une archive avec un compte rendu en pdf (voir les éléments demandés dans le texte).

3 Architecture déployée pour le TP

Nous utilisons le *broker* Mosquitto (<https://mosquitto.org>). Il a été déployé sur deux serveurs, un sur une Raspberry Pi 3B et l'autre sur une Raspberry Pi 4 qui servent de point d'accès. Il y a deux points d'accès car les Raspberry Pi limitent le nombre de stations connectées. Les points d'accès ont été configurés à l'aide de RaspAP (<https://raspap.com>). Répartissez-vous les points d'accès de manière à ne pas les surcharger.

Le point d'accès de la Raspberry Pi 3B déploie un réseau nommé **raspi-webgui** et celui de la Raspberry Pi 4 se nomme **raspi-webgui4**. La clef WEP2 utilisée pour les deux réseaux est **ChangeMe**.

Une fois établit la connexion WiFi, il est possible de se connecter à la Raspberry Pi en ssh avec l'id **pi** et le mot de passe **kywy**. L'adresse IP de la raspberry est **10.3.141.1**, l'adresse des machines connectées sur le point d'accès est de type **10.3.141.xxx**.

Le *broker* utilise le port 1883 pour les communication MQTT et 9001 pour le websocket.

4 Premier contact avec MosQuiTTo

4.1 Publication et souscription locale

Connecter vous sur la Raspi en ssh (mot de passe `kywy`) :

```
ssh pi@10.3.141.1
```

Vous pouvez vérifier que le *broker* est lancé en utilisant la commande (il faudra faire un `ctrl+C` pour quitter) :

```
systemctl status mosquitto
```

Pour tester MQTT vous allez créer un client *publisher* et un *subscriber* sur la Raspberry Pi. Lancer deux terminaux et connecter vous depuis chacun à la Raspberry Pi.

Utiliser ensuite les commandes `mosquitto_sub`¹ et `mosquitto_pub`² pour créer un *subscriber* et un *publisher* (chacun dans son terminal). Le premier client s'abonnera au topic que vous voulez et le second devra publier 'helloworld' sur ce topic. **Attention vous allez tous utiliser le même broker, pensez donc à mettre des noms explicites et uniques pour les topics (par exemple en commençant par votre nom).**

Utiliser la fonction `mosquitto_sub` pour s'abonner à un topic et `mosquitto_pub` pour publier, ces fonctions ont besoin d'arguments (notamment `-t`) que vous pourrez trouver grâce à l'aide (`mosquitto_sub -help`). Il va falloir trouver l'option pour passer le message de `mosquitto_pub` (regardez les notes de fin de page).

Lancez le *subscriber* dans un terminal et, une fois le message publié dans un autre terminal, vérifiez qu'il a bien été reçu par le *subscriber*.

Maintenant que vous avez réussi, ouvrez un troisième terminal et créez un second client abonné au même topic. En publiant un nouveau message, vérifiez que les deux clients *subscribers* le reçoivent bien.

Rapport : mettez une capture d'écran avec les terminaux pour montrer que vous avez bien réaliser une publication et souscription.

4.2 Publication et souscription distante

Installer le client open-source MQTTeX sur votre machine (<https://mqttx.app>). **Attention MQTTeX n'a été testé que sur Mac OS, si vous constatez que c'est trop compliqué à installer sur votre machine passer à la suite.**

Lancer l'application cliente pour vous connectez au *broker*. Tester l'envoi et la réception d'un message entre MQTTeX et le *broker*. Pour cela utiliser le terminal pour disposer d'un *publisher* ou d'un *subscriber*.

Rapport : mettez une capture d'écran de MQTTeX et du terminal qui montre que vous avez réussi à envoyer et recevoir des messages.

1. https://mosquitto.org/man/mosquitto_sub-1.html
2. https://mosquitto.org/man/mosquitto_pub-1.html

5 MQTT avec la carte Feather

5.1 Présentation du matériel

La carte Adafruit-feather-M0-Wifi contient un contrôleur WiFi « ATWINC 1500 » d'Atmel (maintenant Microchip), qui est connecté au micro-contrôleur par bus SPI.

Le contrôleur WiFi ATWINC 1500 :

- permet de le configurer en point d'accès ou en station,
- implémente les protocoles DHCP, DNS, TCP/IP (IPv4), UDP, HTTP, HTTPS,
- implémente les protocoles de sécurité WPA/WPA2 Personal, TLS, SSL.

La sécurité WEP n'est pas implémentée (ce qui signifie que ce contrôleur ne peut pas se connecter à un réseau WiFi sécurisé par WEP).

La librairie Arduino qui implémente le pilote est Wifi101. **Ne pas utiliser la librairie Wifi.** Attention, la documentation de la librairie n'est pas vraiment à jour.

5.2 Installation des bibliothèques

Pour mettre en œuvre la communication MQTT sur la Feather M0, nous utiliserons les bibliothèques Arduino :

- Wifi101 d'Arduino,
- PubSub Client de Nick O'Leary.

Avant de commencer, installez ces bibliothèques en utilisant le [gestionnaire de bibliothèques](#) de l'IDE Arduino.

5.3 Connexion WiFi

Ouvrez le croquis d'exemple `ConnectWithWPA` de la bibliothèque Wifi101. Faites-en une copie dans un nouveau projet et modifiez le fichier `arduino_secret` avec les identifiants de connexion au WiFi.

Ajouter au début du `setup` la ligne suivante afin de configurer les broches du port SPI pour avoir accès au contrôleur WiFi :

```
// Configure les broches pour Adafruit ATWINC1500 Feather
Wifi.setPins(8,7,4,2);
```

Observer le code dans le `setup` et la sortie sur le port série. Quelle est votre adresse ?

Rapport : mettez une capture d'écran du log sur le port série montrant que vous avez réussi à vous connecter au WiFi et avec votre adresse.

5.4 Connexion au *broker*

Utiliser le fichier précédent comme base pour la suite du TP.

Les services MQTT sont offerts par la librairie PubSub. Commencez par inclure son entête :

```
#include <PubSubClient.h>
```

La documentation de la librairie PubSub se trouve sur <https://pubsubclient.knolleary.net/api>. Vous en aurez besoin.

Ajouter les constantes et variables suivantes :

```
#define MQTT_broker "10.3.141.1" // Adresse du broker MQTT (IP or URL)
#define MQTT_PORT 1883 // port 1883 pour la connexion non securisee

WiFiClient WiFiclient; // instantiation d'un client WiFi non-SSL
PubSubClient client(WiFiclient); // Utilise le WiFi pour la communication MQTT

// Chaque client MQTT doit avoir un ID unique
String device_id = "FeatherMO-"; // Ajouter par exemple l'adresse MAC
```

Attention vous vous connectez tous au même *broker*, utiliser un ID unique !

La librairie utilise une fonction de rappel quand un message arrive suite à un abonnement. Il faut donc définir cette fonction. Pour commencer, on peut utiliser la fonction du fichier d'exemple `mqtt_basic` :

```
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived ");
    Serial.print(topic);
    Serial.print("] ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}
```

Pour initialiser la connexion avec le *broker*, il faut initialiser les paramètres du *broker*. Appeler la méthode `setServer` de la classe `PubSubClient` afin d'associer les paramètres du serveur, et `setCallback` pour lier la fonction de callback. Ajoutez dans le `setup` les éléments suivants :

```
client.setServer(MQTT_broker, MQTT_PORT); // parametres du broker
client.setCallback(callback); // fonction d'appel pour traiter les messages
```

Pour se connecter au *broker*, appelez la méthode `connect` (voir la documentation pour les arguments) :

```
if (client.connect(device_id.c_str())) {
    Serial.println("Connected to MQTT broker");
} else {
    Serial.print("MQTT connection failed, rc=");
    Serial.println(client.state());
}
```

Pour rendre un peu plus générique la connexion, allez voir le fichier d'exemple `mqtt_basic` et la fonction `reconnect`. Mettez la en place.

Rapport : Mettez dans le rapport le code que vous avez écrit pour connecter la carte au *broker*. Ajouter une capture qui montre le succès de la connexion.

5.5 Publication et abonnement

La publication d'un message se fait simplement par l'appel à la méthode `publish` :

```
client.publish("myTopic", "hello_world");
```

L'abonnement se fait à l'aide de la méthode `subscribe` :

```
client.subscribe("myTopic");
```

Il faut aussi appeler fréquemment la méthode `loop` de la classe `PubSubClient` pour permettre au client de traiter les messages entrants et maintenir sa connexion avec le serveur :

```
client.loop();
```

Testez la publication et l'abonnement avec une carte Feather M0 et MQTT X ou en se connectant via ssh à la Raspberry Pi.

Récupérez le code des TP précédents pour avoir accès aux données des boutons. Ajoutez du code pour publier un message (différent) lors de l'appui sur un bouton.

Rapport : Mettez dans le rapport les parties du code que vous avez écrit pour la publication et la souscription. Ajoutez des captures pour montrer que tout cela marche.

6 Fonctionnalités avancées

6.1 Message enregistré (Retained Message)

Pour envoyer avec Mosquitto un message avec l'option "retain" il faut ajouter `-r` à la fin de la commande :

```
mosquitto\_pub -t "/test/example" -m "ceci_n'est_pas_un_message" -r
```

Et pour supprimer l'option :

```
mosquitto\_pub -t "/test/example" -r -n
```

où `-n` permet d'envoyer un message nul (longueur zéro).

Publiez un topic en mode enregistré (*retained*) avec les commandes Mosquitto et instanciez un *subscriber* sur la carte Feather pour ce topic, que constatez vous ? Quel pourrait être l'intérêt d'un tel mécanisme ?

Rapport : Expliquez dans le rapport ce que vous avez constaté et à quoi pourrait servir un tel mécanisme.

6.2 Keep Alive

Observez les échanges de message entre le *broker* et votre carte. Pour cela vous allez utiliser l'analyseur de protocole réseau **TShark**³. Il vous permet de lire les données de trames sous une forme décodée à partir d'un réseau ou d'un fichier de capture. Le format de fichier de capture natif de TShark est le format pcapng, qui est également le format utilisé par **Wireshark**⁴ et divers autres outils. TShark est installé sur la Raspberry Pi.

Pour observer les messages circulant via le WiFi sur la Raspberry Pi, connecter vous en ssh et utiliser la commande :

```
sudo tskark -i wlan0 host 10.3.141.xxx
```

avec 10.3.141.xxx l'adresse IP de votre carte (`wlan0` précise l'interface que vous écoutez).

Pour configurer la durée du Keep Alive de votre connexion MQTT avec la librairie Arduino vous pouvez utiliser la méthode :

```
PubSubClient* setKeepAlive (keepAlive)
```

avec `keepalive` la durée en seconde du maintien en vie. Par défaut la valeur est de 15 s.

Modifiez la durée du Keep Alive, se passe-t-il quelque chose ? Analyser les messages échangés entre la carte et le *broker* (via TShark). Comment interprétez-vous le fonctionnement ?

Ajoutez du code sur la Feather pour utiliser de manière cohérente l'option Keep Alive avec une application sur le micro-contrôleur.

Rapport : Expliquez ce que fait Keep Alive et ce que vous avez observé. Mettez les éléments de code et d'explication pour montrer comment vous avez modifié votre code pour utiliser de manière pertinente l'option Keep Alive.

6.3 Acte de dernière volonté (Last Will and Testament)

Le message de *Last Will and Testament* (LWT) se configure lors de la connexion avec la méthode `connect` (<https://pubsubclient.knolleary.net/api#connect>).

Modifiez l'appel à `connect` dans votre code pour ajouter un LWT. Attention lors de l'appel de `connect` il faut énumérer les paramètres dans le bon ordre même s'ils sont optionnels (rappelez vous les valeurs par défaut en c++), il est donc nécessaire de remplir les champs `username` et `password` avec une chaîne de caractère vide pour ces deux paramètres.

Mettez en place le LWT pour votre carte, préparez un *subscriber* sur le topic du LWT (directement sur la Raspberry Pi avec mosquitto ou via MQTTX) et observez les dernières volontés de votre carte (soit en la débranchant sauvagement soit avec un Keep Alive bine configuré).

Rapport : Mettez un extrait de code qui montre comment vous avez mis en place le LWT. Expliquez le fonctionnement et mettez une capture d'écran pour illustrer.

3. <https://www.wireshark.org/docs/man-pages/tshark.html>

4. Wireshark est un analyseur de paquets libre et gratuit. Il est utilisé dans le dépannage et l'analyse de réseaux informatiques, le développement de protocoles, l'éducation et la rétro-ingénierie <https://fr.wikipedia.org/wiki/Wireshark>

6.4 Sessions persistantes et QoS

Mettez en place un *subscriber* avec un session persistante. A quoi cela peu-il servir ? Trouver dans la documentation comment faire. En quoi la QoS est liée à ce point ?

Tester et montrer que vous l'avez mis en place.

Rapport : Ajoutez tous les éléments de code et de test que vous avez réalisés. Expliquez le rôle de la QoS dans ce mécanisme.

7 Pour aller plus loin

Voici trois activités pour aller un peu plus loin dans la maîtrise des outils autour de MQTT. Il n'y a pas d'ordre pour mener ces activités et elles sont là uniquement pour votre culture. Choisissez celle qui vous intéresse. Aucun élément relatif à cette partie n'est nécessaire dans le rapport, ce n'est que pour le plaisir...

Voici trois proposition de travail que vous pouvez réaliser :

- **Déploiement du broker Mosquitto :** Déployer le *broker* Mosquitto sur votre propre machine. Pour cela il faut transformer votre machine en point d'accès puis installer les paquets nécessaires . Les paquets sont disponibles sur Windows, Mac OS et Linux (<https://mosquitto.org/download/>). A vous de suivre ensuite les tutoriaux d'installation propre à chaque système.
- **MQTT over websocket :** si vous avez envie de faire un peu de javascript et d'appeler MQTT depuis un navigateur web via un websocket, vous pouvez suivre le [tutorial de Thomas Laurensen](#). Le *broker* a été configuré pour accepter des websocket sur le port 9001 et le tutorial utilise la librairie javascript de Paho.
Le projet Eclipse Paho est un client MQTT open source qui est disponible pour une variété de langages de programmation, y compris JavaScript. La page [Eclipse Paho Downloads](#) fournit un résumé des différents langages de programmation pris en charge et des fonctionnalités fournies pour chacun d'eux.
- **MQTT Python :** Le projet Paho (voir explication ci-dessus) propose aussi un paquet en Python `paho-mqtt` pour utiliser le protocole MQTT depuis du code en Python. La librairie est très simple à utiliser et de nombreux tutoriaux existent en ligne. Le tutorial <https://www.emqx.com/en/blog/how-to-use-mqtt-in-python> explique de manière claire comment utiliser le paquet et propose un code fonctionnel. À vous de le tester et de jouer avec. Vous pouvez aussi vous lancer dans l'utilisation de Paho avec un autre langage, comme le C ou C++ par exemple à vous de choisir...