Middleware for IoT

Compte rendu de TP

Aude Jean-Baptiste & Romain Moulin, 5ISS 2023
Enseignants de TP : David Gauchard; Jeremie Rodez *jeremie.rodez@univ-tlse3.fr*

Table des matières

Introduction

Ces TPs ont pour objectif de nous faire explorer les standards fréquemment utilisés dans l'IoT, avec pour but final la réalisation d'une petite application de domotique.

TP 1 & 2 : MQTT

2. Caractéristiques de MQTT

a. Quelle est l'architecture typique d'un système IoT basé sur le protocole MQTT?

MQTT repose sur un système de publish / subscribe (publication / abonnement). Un broker centralise les communications. Les appareils publient la valeur de leurs données sur des topics et ceux qui en ont besoin les récupèrent en s'abonnant aux topics nécessaires.

b. Quel est le protocole IP sous MQTT? Qu'est-ce que cela implique en termes d'utilisation de bande passante, type de communication, etc?

MQTT se base typiquement sur TCP et IP. Cela suppose des communications avec garanties de fiabilité et d'ordre. En termes d'utilisation de bande passante, cela suppose également une utilisation de la bande passante pour le handshake TCP.

c. Quelles sont les différentes versions de MQTT?

La version la plus actuelle d'MQTT est la version 5. La version 2.0.18 de Mosquitto utilisée dans ce TP supporte les versions 5.0, 3.1.1 et 3.1.

Vous pouvez trouver les spécifications de chaque version ici [1].

d. Quel type de sécurité/authentification/chiffrement est utilisé par MQTT?

MQTT s'appuyant sur TCP, il n'est par défaut pas encrypté. Il peut cependant reposer sur TLS, et par défaut, s'appuie exclusivement sur TLS sur le port 8883[2]. L'accès au broker MQTT est également protégé par une authentification user / mot de passe, qui sont cependant envoyés en clair. Elle peut également être désactivée, ce que nous faisons dans ce TP pour des raisons de simplicité.

Supposons que vous avez des appareils incluant un bouton, une source de lumière et un capteur de luminosité. Vous souhaitez créer un système intelligent pour votre maison avec ce comportement:

[1] https://mqtt.org/mqtt-specification/
[2] https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/

- *vous souhaitez pouvoir allumer la lumière manuellement avec le bouton*
- *la lumière s'allume automatiquement quand la luminosité passe sous une certaine valeur.*

e. Quels différents topics seront nécessaires pour obtenir ce comportement? Quid des publishing et subscribing?

Nous aurions théoriquement besoin d'un seul topic :

- pour l'état de la LED

Les publications sur ce topic seront les suivantes :

- le bouton publie des on / off selon si on y a appuyé ou pas
- le capteur de luminosité publie on si la luminosité passe sous le seuil critique

Quant aux abonnements :

- la LED s'abonne au topic pour savoir l'état qu'elle doit adopter

4. a. Donnez les caractéristiques principales du board nodeMCU en termes de communication, langage de programmation et capacités en input/ output.

Le board NodeMCU possède un module Wifi intégré qui nous permet de nous connecter facilement au broker. Il est équipé d'un UART, d'I2C et de nombreux ports en input et output. Il est programmable en C++, comme une carte arduino.

5. Création d'une application simple

Le TP1 ayant été consacré à l'installation et la prise en main du matériel nécessaire, nous n'avons pas eu le temps de réaliser cette partie.

6. Création d'une application complexe

Architecture implémentée

Nous implémentons, avec un autre binôme, l'architecture suivante :

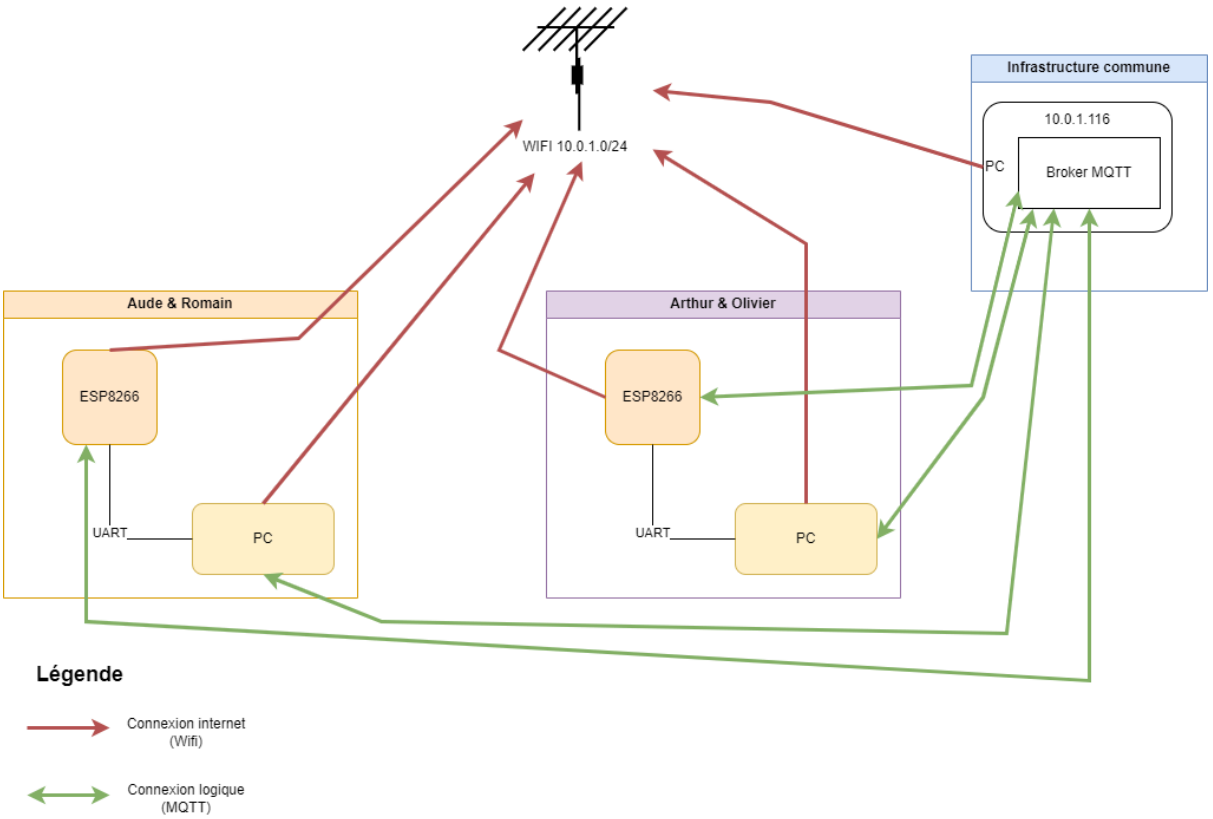


Figure 1: Architecture implémentée lors du TP2[3]

NB : Le broker que nous utilisons est situé sur le PC d'Olivier.

Chaque ESP8266 est connecté à une LED et un bouton. L'objectif est que le bouton d'Olivier et Arthur allume notre LED, et vice versa.

Topics choisis

Pour cela, nous utilisons les topics suivants :

- insa/gei/romainled
- insa/gei/olivierled

Une publication du message "off" éteint la LED correspondante, une publication du message "on" allume la LED correspondante.
Ainsi par exemple, pour allumer la LED d'Arthur et Olivier, on appuie sur notre bouton. Cela publie "on" sur le topic insa/gei/olivierled, ce qui allume leur LED, car ils sont abonnés au topic insa/gei/olivierled.
Pour éteindre leur LED, on ré appuie sur le bouton. On garde en mémoire l'état de leur LED, on envoie donc "off" cette fois-ci.

NB : Le booléen utilisé pour garder l'état de la LED nous sert également à éviter d'envoyer "on" ou "off" en continu lorsque le bouton est maintenu enfoncé.
NB2 : Pour des raisons de simplicité et en raison de la sérialisation des messages, nous utilisons :

- "0" pour "off"

[3] figure faite avec draw.io par Aude Jean-Baptiste

- “1” pour “on”

Phase de test

Pour tester le comportement de notre programme, on utilise le client mosquitto pour générer les messages souhaités sur les bons topics et observer les leds.

Retours et perspectives

Nous notons tout de même quelques anomalies au niveau de l’allumage, par exemple maintenir le bouton appuyé génère souvent des anomalies (extinction au moment du relâchement etc), probablement dues au *bouncing*. Nous aurions pu utiliser une librairie (software) pour régler ce problème.

Nous aurions pu implémenter cette application différemment, d’une manière plus proche d’une implémentation réelle d’application de domotique. Dans cette version alternative, au lieu de garder en mémoire l’état de la LED, on enverrait simplement “toggle” pour changer son état. L’ESP d’Arthur et Olivier nous renverrait alors son état actuel.

Nous avons choisi notre implémentation actuelle en raison d’une plus grande simplicité, car nous ne sommes guère à l’aise en développement C++, surtout en temps limité.

TP3 : Middleware for IoT basé sur le standard oneM2M

Expliquez dans le rapport comment vous avez créé votre device *et donnez le code.*

Nous avons réalisé quatre fonctions en Python pour interagir avec le CSE :

- createAE
- createCNT
- createCIN
- get_last_CIN

Le code de ces fonctions peut être trouvé dans la partie suivante. Vous constaterez que nous avons apporté quelques améliorations au code proposé, par exemple la suppression du *json.dumps(json.loads…)* ou d’un point virgule qui empêchait l’exécution. Nous avons également ajouté des arguments pour une plus grande maniabilité des fonctions.

Code des fonctions d'interaction avec le CSE

```
def createAE(cse,ae_name, api,host,originator,ri):
    url=host+cse
    print("requesting " + url + "...")
    payload = {
        "m2m:ae": {
            "rn": ae_name, # nom du payload
            "api": api,
            "rr": True,
            "srv": ["3"]
        }
    }
    _headers = {'X-M2M-Origin':originator,'X-M2M-RI':ri,'X-M2M-RVI':'3', "Content-
Type":"application/json;ty=2", "Accept":"application/json"}
    _payload=json.dumps(payload, indent=4)
    r = requests.post(url,data=_payload,headers=_headers)
    handleResponse(r)
```

Figure 2: Le code de notre fonction pour créer un nouvel AE[4]

```
def createCNT(cse,ae_name,host,cont_name,originator,ri):
    url=host+cse+'/'+ae_name
    print("requesting " + url + "...")
    payload = {"m2m:cnt": {"rn": cont_name}}

    _headers = {'X-M2M-Origin':originator,'X-M2M-RI':ri,'X-M2M-RVI':'3', "Content-
Type":"application/json;ty=3", "Accept":"application/json"}
    _payload=json.dumps(payload, indent=4)
    r = requests.post(url,data=_payload,headers=_headers)
    handleResponse(r)
```

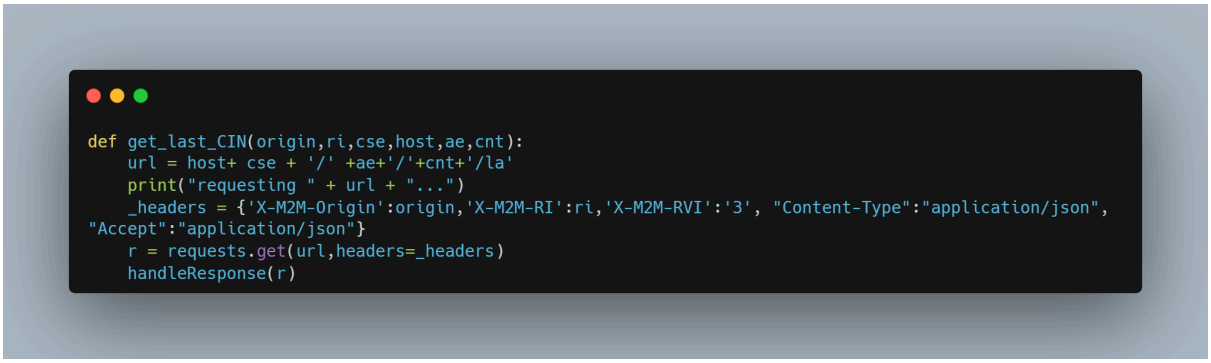
Figure 3: Le code de notre fonction pour créer un nouveau container

[4] Ces superbes images de notre code sont réalisées avec https://carbon.now.sh

```python
def createCIN(ae, cnt,host, cse,value, ri, origin):
    url = host+ cse + '/' +ae+'/'+cnt
    print("requesting " + url + "...")

    payload = {
        "m2m:cin": {
            "cnf": "text/plain:0",
            "con": value
        }
    }
    _headers = {
        'X-M2M-Origin': origin,
        'X-M2M-RI': ri,
        'X-M2M-RVI': '3',
        'Content-Type': 'application/json;ty=4',
        'Accept': 'application/json'
    }
    _payload = json.dumps(payload, indent=4)
    r = requests.post(url, data=_payload, headers=_headers)
    handleResponse(r)
```

Figure 4: Le code de notre fonction pour créer une nouvelle content instance



```python
def get_last_CIN(origin,ri,cse,host,ae,cnt):
    url = host+ cse + '/' +ae+'/'+cnt+'/la'
    print("requesting " + url + "...")
    _headers = {'X-M2M-Origin':origin,'X-M2M-RI':ri,'X-M2M-RVI':'3', "Content-Type":"application/json",
"Accept":"application/json"}
    r = requests.get(url,headers=_headers)
    handleResponse(r)
```

Figure 5: Le code de notre fonction pour récupérer la dernière CIN postée

Simulation de l'application des TP1 et 2

Nous utilisons le CSE fourni par le Notebook, cse-in. Nous choisissons de créer un AE pour la gestion de l'allumage des LED. Cet AE contient deux containers, LED-Romain et LED-Aude, sur lesquels on poste au fur et à mesure une CIN dont le payload contient l'état de la LED souhaité (on ou off). Idéalement ceci aurait dû être associé à un système de notifications, qui n'a pas pu être implémenté faute de temps.

```
createAE(cse="cse-in",ae_name="lightAE",
api="Nlight",host="http://localhost:8080/",originator="CRomain",ri="376")
createCNT(cse='cse-in',ae_name="lightAE",host="http://localhost:8080/",cont_name='LED-
Romain',originator='CRomain',ri='34')
createCNT(cse='cse-in',ae_name="lightAE",host="http://localhost:8080/",cont_name='LED-
Aude',originator='CRomain',ri='39')
createCIN(ae="lightAE", cnt="LED-Romain",host="http://localhost:8080/", cse= "cse-in",value="on",
ri="297", origin="CRomain")
createCIN(ae="lightAE", cnt="LED-Aude",host="http://localhost:8080/", cse= "cse-in",value="on", ri="297",
origin="CRomain")
```

Figure 6: Le code de notre simulation (ébauche)

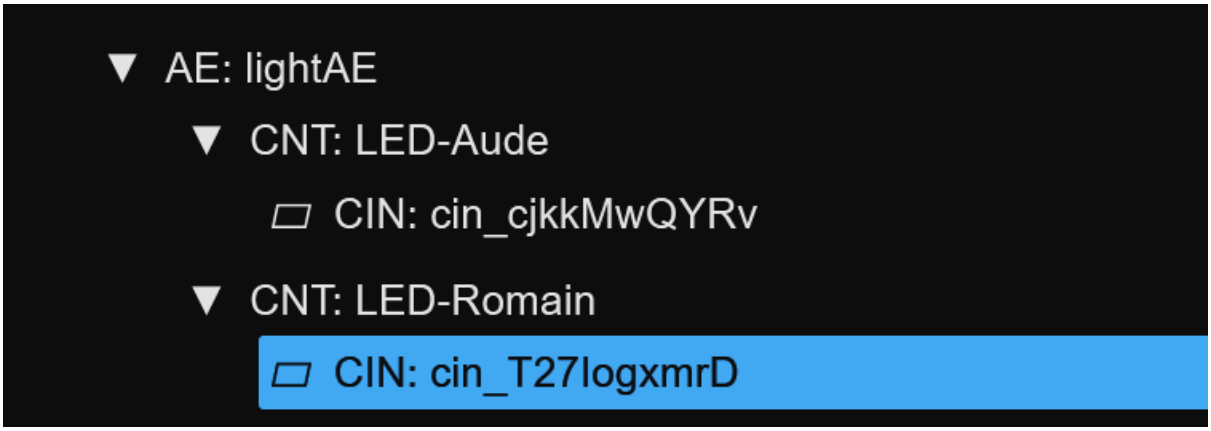


Figure 7: capture d'écran de notre AE, les containers et CIN

Figure 8: Capture d'écran d'une CIN "on"

TP4 : Prototypage rapide d’applications pour l’IoT

Node Red Flows : Capteurs & Actionneurs, Dashboard…

Nous avons choisi d’utiliser deux topics MQTT :

- insa/gei/romain/button
- insa/gei/romain/led

Nous utilisons également un AE et container OneM2M :

- lightAE
- LED-Romain

Avec l’application Node Red, on subscribe au topic /button et on transfère le message dans le container LED-Romain. Ensuite, on extrait le contenu de la CIN grâce à une écoute active, avant de republier dans le topic /led qui active ou désactive la LED.

Bien sûr, dans une application plus complexe, les topics MQTT pourraient être sur des brokers différents et l’application Node Red se chargerait de faire la liaison.
Vous trouverez dans le fichier flows.json joint à ce rapport les flux Node Red associés.

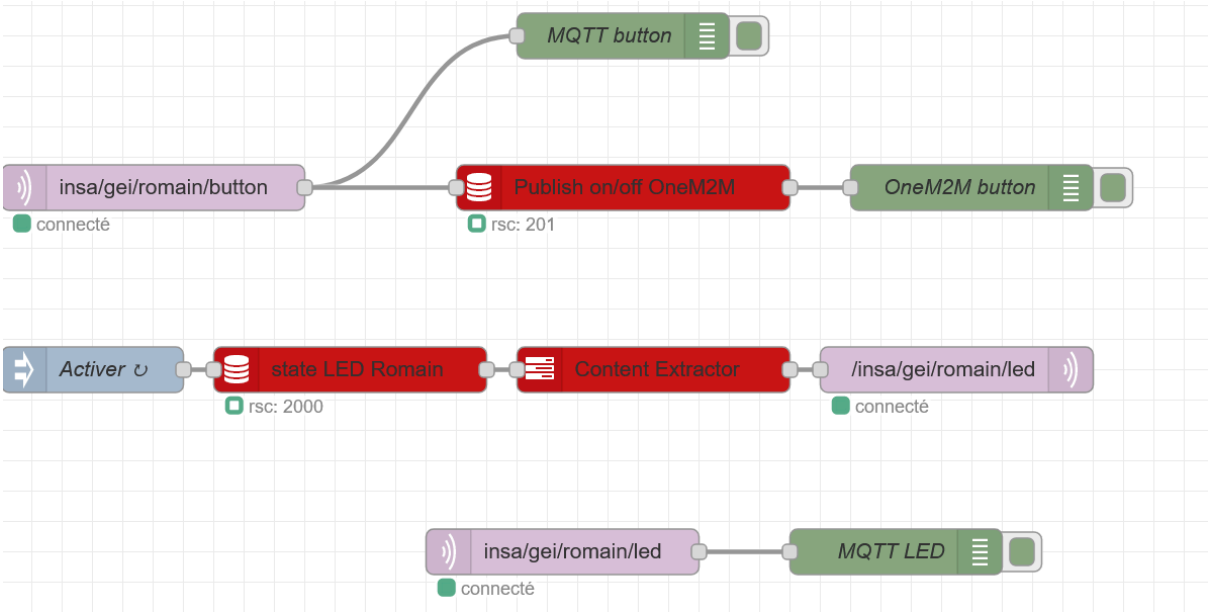


Figure 9: Le workflow de notre application Node Red

Avantages et inconvénients d'une application avec Node Red

L'avantage principal de Node Red pour construire une application est sa simplicité. Le niveau d'abstraction offert par le codage par bloc permet une vision globale sur son application ce qui rend le débogage plus simple.

Un désavantage s'accompagnant avec la simplicité de Node Red est que l'on voit justement moins en détail ce que l'application fait en pratique. Le codage d'une application à l'aide d'un langage de programmation comme python ou javascript permet une meilleure compréhension en détail de ce qui est fait.

Concernant ce sujet de TP, on aurait cependant pu simplement continuer à utiliser le Broker MQTT. Node Red nous permet d'interfacer facilement OneM2M et MQTT, mais pour cette application précise, assez simple, l'overhead n'était pas forcément nécessaire.

Conclusion

Ces 4 séances de TP nous ont permis de comprendre davantage et d'expérimenter le fonctionnement du protocole MQTT, du standard OneM2M et de Node Red.
Les deux premières séances furent également l'occasion de nous familiariser avec le fonctionnement des cartes arduino et autres composants électroniques (LED, interrupteurs, fonctionnement des In/Out). En effet, en tant qu'étudiants issus de la filière Informatique et Réseaux, nous n'en avons que peu eu l'occasion.
Malheureusement, nous n'avons pas pu aller au bout des séances de TPs en raison d'une bonne partie de chaque TP qui a dû être consacrée à l'installation des logiciels nécessaires sur nos machines personnelles.