Découverte de l'IoT au niveau applicatif: mise en œuvre de la SAÉ 203 du BUT R&T

Manuel Munier[1] Nouha Laamech[2]

manuel.munier@univ-pau.fr laamech.nouha@univ-pau.fr

[1,2] IUT des Pays de l'Adour
Université de Pau et des Pays de l'Adour, E2S UPPA
Rue du Ruisseau – 40000 Mont de Marsan

Thèmes – *informatique, réseaux*

Résumé – *Des capteurs industriels aux smart cities, il n'y a qu'un pas aisément franchi grâce à des innovations toujours plus poussées. Bien loin de l'image de technologie "gadget" qui lui a longtemps collé à la peau, l'Internet des objets (en anglais IoT pour Internet of Things) peut être considéré comme le prolongement logique du développement d'Internet et de la connectivité réseau. La formation BUT R&T est en parfaite adéquation avec les besoins en compétences pour cette nouvelle technologie. Si cette dernière est le plus souvent abordée sous l'angle de l'électronique et des réseaux de communication, nous vous présentons dans cet article une approche plutôt orientée informatique et logiciel de la gestion des informations échangées dans un réseau IoT. Nous avons ainsi pu faire découvrir le monde de l'IoT aux étudiants de fin de 1ère année du BUT R&T de Mont de Marsan au travers de la SAÉ 203 "mettre en place une solution informatique pour l'entreprise". Au-delà du sujet IoT lui-même, les étudiants ont également découvert qu'un système d'information n'était pas forcément un seul et unique "gros" programme informatique mais pouvait aussi être constitué de plusieurs "petits" modules logiciels qui, une fois interconnectés, fournissent des services dignes de ce nom aux utilisateurs.*

Mots-clés – *IoT, MQTT, Python, web dynamique.*

1 Introduction

La réforme du DUT en BUT mise en application en 2021 a introduit les situations d'apprentissage et d'évaluation (SAÉ) permettent l'évaluation en situation des compétences des étudiants. Dans cet article nous allons vous présenter la manière dont nous avons mis en œuvre la SAÉ 203 (*"mettre en place une solution informatique pour l'entreprise"*) en fin de 1ère année du BUT RT à Mont de Marsan.

Notre idée lorsque nous avons préparé cette SAÉ qui intervenait en fin de semestre 2 était de faire travailler les étudiants en mode projet sur un sujet qui mobiliserait simultanément leurs compétences acquises dans les modules du BUT 1 en informatique bien évidemment, mais également leur "culture générale" dans les réseaux informatiques.

Nous leur avons donc proposé de concevoir par leurs propres moyens une plateforme de gestion des informations en IoT (*Internet of Things*, c'est-à-dire les objets connectés). Ceci sans passer par des outils logiciels évolués tels qu'openHAB ou Home Assistant, ou bien des plateformes telles que ThingSpeak. Nous leur avons demandé de développer par eux-mêmes les différentes briques permettant d'échanger des messages MQTT [1], de stocker les informations dans une base de données, de fournir une interface web, de produire des graphiques ou des documents PDF, etc.

Cet article introduit le sujet sur lequel les étudiants ont travaillé, présente ensuite la mise en œuvre de cette SAÉ, puis conclut par un retour d'expérience (spoiler : positif !), tant du point de vue des enseignants que de celui des étudiants.

2 Présentation du sujet

L'architecture générale de la solution que les étudiants doivent développer est représentée sur la Figure 1. Les connexions IoT se feront au travers du protocole MQTT. Soit directement depuis le device, soit par l'intermédiaire d'un programme (ex : Python) qui collecte les données et les retransmet à la plateforme via MQTT (*publish*). D'un autre côté un ou des programmes (ex : Python) qui récupèrent ces données depuis MQTT (*subscribe*) et les stockent (ex : base de données SQL). D'autres programmes pourraient également enregistrer directement des données dans la BdD. Finalement, à partir de cette BdD, différents programmes peuvent présenter les données à l'utilisateur : des scripts PHP qui génèrent des pages HTML, des programmes Python (ou autre) qui génèrent des documents PDF, des images PNG, etc.

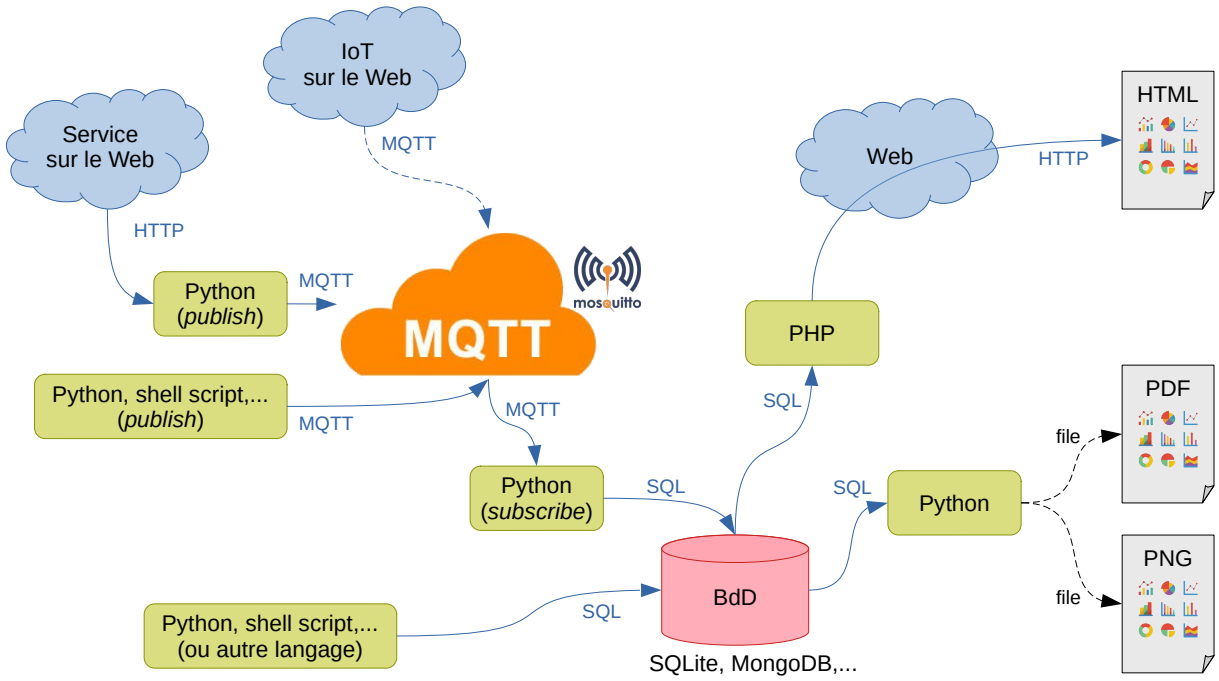

FIGURE 1 – Architecture générale

Ce projet sera donc constitué de plusieurs "modules" logiciels pouvant être développés de manière relativement indépendante les uns des autres et dans des langages de programmation différents.

2.1 Le protocole MQTT

MQTT est un protocole de type *publish/subscribe*, c'est-à-dire qu'un client produisant une donnée va se connecter à un serveur, le *broker* MQTT, et va lui envoyer un message MQTT sur un certain canal (un *topic*) : action *publish*. De l'autre côté, un client souhaitant recevoir ces informations va lui aussi se connecter au *broker* MQTT et va s'abonner au *topic* qui l'intéresse : action *subscribe*. À chaque fois que le *broker* recevra un message sur un *topic* il le diffusera à

1. MQTT : Message Queuing Telemetry Transport

tous les clients qui se seront abonnés à ce *topic*. Le fonctionnement est en quelque sorte similaire à celui d'une liste de diffusion.

Dans une telle architecture, le *publisher* ne connait pas les clients qui recevront son message, et les *subscribers* ne savent pas quel est le client qui a publié le message. Le contenu d'un message MQTT est en effet extrêmement simple : du point de vue applicatif il ne contient que 2 chaînes de caractères qui sont le nom du canal de diffusion (appelé le *topic*) et le contenu de message (appelé le *payload*).

En soi le protocole MQTT est extrêmement simple, et le fait que toutes les informations soient transmises en format texte permet d'analyser les échanges facilement à l'aide d'outils tels que Wireshark que nos étudiants sont déjà habitués à utiliser.

Dans son fonctionnement de base, MQTT ne fournit aucune garantie de service, c'est-à-dire qu'il n'est pas garanti que les messages envoyés soient reçus par tous les clients, ni dans le même ordre. Et si un client est déconnecté du réseau, il ne recevra pas les messages envoyés pendant ce laps de temps. Certains *brokers* peuvent proposer 2 niveaux de qualité de service. Le premier niveau garantit l'acheminement des messages jusqu'aux clients, dans le même ordre qu'à l'envoie, à condition que les clients soient connectés à ce moment là. Le second niveau de qualité de service stockera en plus les messages sur le *broker* afin de pouvoir les (re)distribuer aux clients lors de leur reconnexion au réseau.

Du point de vue du contenu de l'information, MQTT ne définit aucune sémantique pour le *payload* : ce n'est qu'une chaîne de caractères. À charge aux modules métier (communiquant via des messages MQTT) de donner une signification aux contenus échangés. Une solution toute trouvée pour représenter des informations structurées est bien évidemment le format JSON [2] que nos étudiants de RT MdM auront déjà eu l'occasion d'étudier dans les précédents modules R207 (*"sources de données"*) et R208 (*"analyse et traitement de données structurées"*).

Finalement, il est à noter que le protocole MQTT propose de structurer les *topics* de manière arborescente (un *topic* pourra contenir des sous-*topics*, lesquels pourront à leur tour contenir des sous-sous-*topics*, etc.). Nous n'entrerons pas dans les détails du nommage des *topics* MQTT dans le cadre de cet article. L'idée à retenir est simplement qu'un client pourra ainsi s'abonner à un "groupe" de *topics*, ce qui facilitera l'implémentation de la logique métier.

2.2 Les *publishers* MQTT

Si dans la "vraie vie" nous utiliserions de véritables objets connectés envoyant directement des messages MQTT en WiFi, dans cette SAÉ les étudiants ont pour objectif d'écrire des programmes qui publient des messages MQTT. Ils peuvent utiliser les langages de programmation qu'ils souhaitent.

Ce peut être des script shell Unix qui récupèrent des données (ex : SNMP, statistiques système issues de commandes Unix, etc.) et les envoient ensuite au *broker* MQTT (ex : via la commande `mqtt_pub` fournie avec Eclipse Mosquitto).

Une seconde solution consiste à écrire des programmes Python qui envoient des messages MQTT via la librairie Eclipse Paho par exemple. Ces programmes peuvent collecter des données de différentes manières : génération de données aléatoires, utilisation de librairies permettant d'accéder au système d'exploitation, requêtes à des sites web externes (ex : site en Open Data fournissant des jeux de données en JSON ou en XML).

2.3 Les *subscribers* MQTT

Les étudiants doivent également écrire des programmes, en Python par exemple, qui vont écouter sur certains canaux et, à réception d'un message MQTT, extrairont les données du *payload* et les enregistreront dans une base de données (ex : SQLite).

Les étudiants doivent pour cela indiquer au moment de la souscription de leur programme à un *topic* quelle sera la fonction *callback* qui sera automatiquement invoquée à réception d'un message sur ce *topic*. Tout la "mécanique" d'écoute et de déclenchement d'un certain traitement sur un certain événement est complètement cachées dans la librairie MQTT qu'ils utiliseront. Leur seul travail consiste à écrire le corps de la fonction *callback* : extraction des informations depuis le *payload* puis traitement de ces données.

2.4 La base de données

Le protocole MQTT ne gère que la diffusion des messages. Il n'y a aucune persistance des données. C'est donc au programmeur de mettre en place un système de stockage des différentes valeurs reçues à des fins d'historique, de statistiques, etc. Dans le cas de cette SAÉ les étudiants doivent réfléchir aux informations qu'ils souhaiteront stocker, de quelle manière ils devront les exploiter (les requêtes), puis concevoir les tables de la base de données SQL.

2. JSON : JavaScript Object Notation

Sur le fond, rien de nouveau par rapport à ce qu'ils ont déjà vu en 1ère année. Sur la forme, il s'agit ici d'intégrer leur code Python/SQL au code Python/MQTT de leur *subscriber* : à chaque déclenchement, leurs fonctions *callback* devront extraire les données du *payload* du message MQTT reçu puis les stocker "au bon endroit" et "dans le bon format" dans la base de données SQL.

2.5 La présentation des données

Dans cette partie de la SAÉ, l'objectif des étudiants est de développer un ou plusieurs modules de présentation des informations enregistrées dans la base de données. Une solution peut être un site web dynamique codé en HTML+PHP+SQL. Mais le rendu à l'utilisateur final ne doit pas se limiter à une "simple table HTML", tout aussi joliment habillée qu'elle puisse être avec un CSS adapté. Les étudiants doivent ici rechercher et utiliser une des librairies PHP permettant de produire, à partir d'un tableau de données, des graphiques au format PNG ou encore des graphiques dynamiques à intégrer dans leurs pages web.

Dans cette partie les étudiants peuvent également envisager de fournir un serveur qui, au travers de scripts en PHP par exemple, retournerait une extraction des données de la base (ex : requêtes SQL pour filtrer les données) au format JSON. Ce faisant, ce serveur (situé à droite sur la Figure 1) deviendrait un service web comme un autre (en haut à gauche sur la Figure 1) pour d'autres projets…

3 Mise en œuvre de la SAÉ 203

Dans cette section nous allons présenter les modalités de mise en œuvre de cette SAÉ. Certains étudiants pouvant (éventuellement) encore avoir quelques lacunes, nous souhaitions leur mettre à disposition une infrastructure minimale afin que, malgré ces lacunes, ils ne soient pas bloqués et puissent néanmoins développer telle ou telle pièce du puzzle.

3.1 Infrastructure MQTT

Pour faire simple, nous pouvons commencer par utiliser le *broker* MQTT de test de la fondation Eclipse disponible à l'URL test.mosquitto.org. Il nous a suffit pour cela de demander aux administrateurs système du réseau de l'IUT de nous ouvrir le port `1883` afin que nos *publishers* et nos *subscribers* puissent se connecter au serveur. Et les étudiants peuvent également se connecter à ce *broker* de test depuis chez eux et ainsi continuer le développement de leur projet sans avoir à maintenir plusieurs configurations différentes selon le réseau sur lequel ils sont connectés.

3.2 Tutoriel MQTT & Python

Afin de faciliter l'apprentissage de cette technologie MQTT à nos étudiants nous avons développé un petit tutoriel [3] expliquant pas-à-pas comment utiliser la bibliothèque Paho MQTT dans des programmes Python, tant du côté *publisher* que du côté *subscriber*.

S'il existe déjà bon nombre de documentations sur Internet quant à l'utilisation de MQTT dans différents langages de programmation, il nous a semblé judicieux de proposer à nos étudiants un tutoriel "sur mesure", c'est-à-dire en réduisant les programmes et les concepts utilisés au strict minimum nécessaire, et en utilisant pour les explications le même "style pédagogique" que celui auquel nous les avons habitués…

À noter qu'il existe également une version Java de ce tutoriel [4].

3.3 Programme *publisher* de test

Afin d'aider les étudiants qui auraient des difficultés à publier eux-mêmes leurs propres données via MQTT, nous leur avons écrit un programme Python qui récupère les informations météo au format JSON sur le site OpenWeather pour l'emplacement "IUT MdM" et les republie sur différents *topics* MQTT.

Les données élémentaires extraites du JSON (ex : température, température ressentie, humidité, etc.) sont publiées sur des *topics* individuels (*payload* = 1 entier ou 1 réel). L'ensemble de ces données est également publié sur le *topic* `all` sous la forme d'un dictionnaire JSON.

Ce programme a ensuite été placé dans la `crontab` d'un Linux afin qu'il soit automatiquement exécuté toutes les 2mn. Une trace d'un programme qui écouterait sur ce(s) *topic(s)* (sur le *broker* `test.mosquitto.org`) est fournie à la Figure 2.

NB : Certains étudiants m'ont demandé s'il était possible de visiter le *data center* hébergeant le projet. Je les ai donc amenés à mon bureau… où ils ont été surpris (et le mot est faible) de voir que ce qu'ils pensaient être un *"data center"*

3. https://munier.perso.univ-pau.fr/tutorial/iot/2022/20220510-mqtt_python/
4. https://munier.perso.univ-pau.fr/tutorial/iot/2022/20220113-mqtt_java/

```
Received ‘17.74‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/temp‘ topic
Received ‘17.4‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/feels_like‘ topic
Received ‘16.73‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/temp_min‘ topic
Received ‘19.42‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/temp_max‘ topic
Received ‘1020‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/pressure‘ topic
Received ‘70‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/humidity‘ topic
Received ‘{"temp": 17.74, "feels_like": 17.4, "temp_min": 16.73, "temp_max": 19.42,
  pressure": 1020, "humidity": 70}‘ from ‘/rtmdm/mm/sae203/g553dtQJS7yk2G/weather/all‘ topic
```

FIGURE 2 – Exemple de messages MQTT reçus

n'était en réalité qu'un ancien laptop Dell Latitude 2100 (sur base Intel Atom N270, tournant sous Xubuntu 18.04) datant d'avant 2010…

3.4 Base de données, serveur web

Pour cette SAÉ nous utilisons le serveur web Apache de l'IUT et dédié aux étudiants (donc inaccessible de l'extérieur) qui a été mis en place pour le module R209 (web dynamique). La base de données est une SQLite, donc stockée localement dans l'espace utilisateur, comme pour les modules R207 et R209.

4 Retour d'expérience

4.1 Compétences mises en œuvre

L'objectif de cette SAÉ étant de développer une plateforme logicielle de gestion des informations en IoT, les principales compétences mises en œuvre qui nous viennent à l'esprit sont bien évidemment celles liées à l'informatique et à la programmation :

- R107, Fondamentaux de la programmation
- R207, Sources de données
- R208, Analyse et traitement de données structurées
- R209, Initiation au développement Web

Sur un tel projet, les étudiants ont également eu l'occasion de mobiliser d'autres compétences du BUT RT, notamment en réseaux. Certains étudiants ont par exemple choisi d'installer leur propre *broker* MQTT (Mosquitto) sur un ordinateur situé chez eux. Ce faisant, ils ont aussi dû configurer leur box Internet pour pouvoir y accéder depuis l'extérieur (NAT & PAT). D'autres ont installé leur "cœur de réseau" sur un Raspberry Pi avec leur téléphone portable en routeur 4G. Avec leurs laptops connectés à ce réseau ad hoc ils avaient ainsi leur environnement de développement en permanence avec eux.

4.2 Retours des étudiants

Durant cette SAÉ les étudiants ont pu découvrir le monde de l'IoT non plus au travers des thématiques du réseau et des télécommunications (connectivité IP, routage, communications LoRa, etc.) mais du point de vue de l'infrastructure logicielle : échange de données, collecte et exploitation de ces données, etc.

En s'appuyant sur un protocole de communication très simple (MQTT), ils ont pu développer diverses (petites) briques logicielles, guère plus compliquées que les programmes qu'ils avaient réalisés en 1ère année de BUT, puis les assembler pour aboutir à un système d'information complet et fonctionnel. Les étudiants ont ainsi pu constater que sans être des informaticiens chevronnés qui développent de gros logiciels, ils peuvent parfaitement répondre à un cahier des charges en ne codant que de petits programmes qui s'échangent des informations. Ce qui leur permet également de mixer différents langages (ex : Python, JavaScript, PHP, etc.) et de faire appel à différentes technologies de l'information (ex : HTML/CSS, MQTT, JSON, SQL, etc.). À eux de savoir utiliser le bon outil pour le bon usage, en fonction bien évidemment de leurs compétences.

Les retours ont tous été très positifs, même de la part des étudiants pour lesquels l'informatique n'était pas une vocation. À défaut d'avoir un coup de cœur, ils ont néanmoins compris les différentes facettes d'une telle architecture, et tous se sont piqués au jeu de vouloir réussir à "faire tourner quelque-chose", avec plus ou moins de fonctionnalités. Tous étaient satisfaits et, disons le, un peu fiers d'avoir pu mener leur projet jusqu'au bout. Et si les étudiants les moins à l'aise en programmation sont restés sur la thématique des données météorologiques sur la base des squelettes fournis (ex :

section 3.3), d'autres ont choisi des domaines d'application bien différents. L'un a développé un plugin sous Discord pour récupérer les informations des discussions et les transmettre via MQTT avec, au niveau du tableau de bord, un affichage dynamique (quasi) en temps réel. Un autre a choisi de se connecter au site Flight Radar pour collecter les données des vols à intervalle régulier, les republier en MQTT, puis es afficher sur un tableau de bord qui cette fois n'est plus un histogramme mais un calque au-dessus d'une Google Maps ! Ce n'est finalement qu'une API de visualisation de données comme une autre…

Ils ont aussi pu constater qu'un protocole tel que MQTT, pourtant en passe de devenir un standard en IoT y compris dans le milieu industriel, ne propose aucun mécanisme pour la sécurité de l'information : le *payload* des messages circule en clair, une fois connecté au *broker* (lequel peut éventuellement demander un mot de passe) chacun peut écouter et émettre sur tous les *topics*, le consommateur d'un message ne sait pas qui l'a produit (ni identité, ni signature électronique), etc. La mise en place de tels mécanismes incombe justement au système d'information. Les étudiants ont ainsi réfléchi à l'organisation des *topics* à utiliser et sur quel *broker*, d'autres ont structuré le *payload* de leur messages en JSON, et certains ont même été jusqu'à chiffrer le contenu de leurs messages pour assurer la confidentialité. Mentionnons également le travail d'un étudiant qui a mis en place un système de licence sur son API de présentation des données : les utilisateurs doivent s'enregistrer ; ils obtiennent alors une clé de licence qu'ils doivent fournir dans les requêtes qu'ils font sur l'API, laquelle "adapte" les données transmises (en JSON) en fonction de la licence des utilisateurs ! Cette SAÉ a donc aussi été l'occasion pour eux de se sensibiliser à la cybersécurité au niveau de l'architecture logicielle cette fois.

4.3 Ressenti des enseignants

Du côté des enseignants ayant encadré cette SAÉ le constat est similaire : tous ont noté un excellent investissement de tous les étudiants ! Y compris de la part des étudiants n'ayant pas d'appétence particulière pour l'informatique et la programmation et qui, au travers de ce projet, se sont rendus compte qu'ils étaient malgré tout capables de produire de petits programmes et de les assembler pour fournir une solution informatique complète. La curiosité et le challenge ont fait leur œuvre…

5 Perspectives

Ce projet IoT nous a servi de support pour mettre en œuvre la SAÉ 203 *"mettre en place une solution informatique pour l'entreprise"* au département RT de Mont de Marsan pour les promotions 2021-2022 et 2022-2023. Parmi les perspectives envisagées il est bien évidemment prévu d'intégrer l'utilisation de "vrais" objets connectés pour collecter des informations, et non plus uniquement le téléchargement de données à partir de sites tels que OpenWeather. Un moyen simple consisterait à passer par une plateforme de domotique openHAB, laquelle intègre nativement de nombreux plugins pour connecter différents périphériques IoT du commerce. Outre ses outils intégrés, openHAB permet également de déclencher des scripts JavaScript en réaction aux différents événements de la plateforme. Via ces scripts nous pouvons, par exemple, publier les données reçues des capteurs via des messages MQTT. Nous disposons ainsi d'une alternative au programme *publisher* de test présenté à la Section 3.3. Une telle plateforme openHAB est déjà opérationnelle dans nos bureaux (Figure 3). Elle tourne sur un Raspberry Pi et gère plusieurs capteurs connectés en Z-Wave et en RFLink 433MHz (via un Arduino Mega 2560, Figure 4). L'envoi de message MQTT via JavaScript fonctionne ; reste à définir le format du *payload*.

Une autre perspective serait de réutiliser les travaux réalisés dans la SAÉ 102 (*"s'initier aux réseaux informatiques"*) dans laquelle nos étudiants ont, entre autres, connecté différents capteurs sur un Raspberry Pi pour ensuite, via des scripts en Python, collecter les données et les transmettre à une MongoDB extérieure. Il suffirait de remplacer cette dernière étape de leurs scripts Python par l'envoi de messages MQTT. 2 étudiants ont d'ailleurs déjà réalisé ce travail en 2022-2023 dans le cadre de la SAÉ 203.

La même idée pourrait être développée à un niveau encore plus bas à l'aide de capteurs branchés sur des ESP32…

Une dernière perspective évoquée très récemment serait d'ouvrir cette SAÉ 203 vers nos activités de recherche actuelles sur la gouvernance du partage de données dans les environnements connectés. Très brièvement, nos travaux concernent l'autodétermination informationnelle, c'est-à-dire que c'est le producteur d'une information qui va décider quelles sont les règles à respecter pour pouvoir utiliser cette information et comment l'utiliser. C'est en quelque sorte une politique de contrôle d'usage encapsulée dans une licence attachée à l'information. Ensuite, le consommateur de l'information devra s'assurer que l'usage qu'il compte faire de cette information est conforme à la licence. Dans le cadre de nos expérimentations du modèle et de l'architecture proposés nous avons choisi d'utiliser des données issues de l'IoT. L'idée serait donc que le *payload* des messages MQTT encode ces données sous la forme d'une `Observation` au sens de l'ontologie `SSN`[5].

5. SSN : Semantic Sensor Network Ontology

FIGURE 3 – Plateforme IoT openHAB

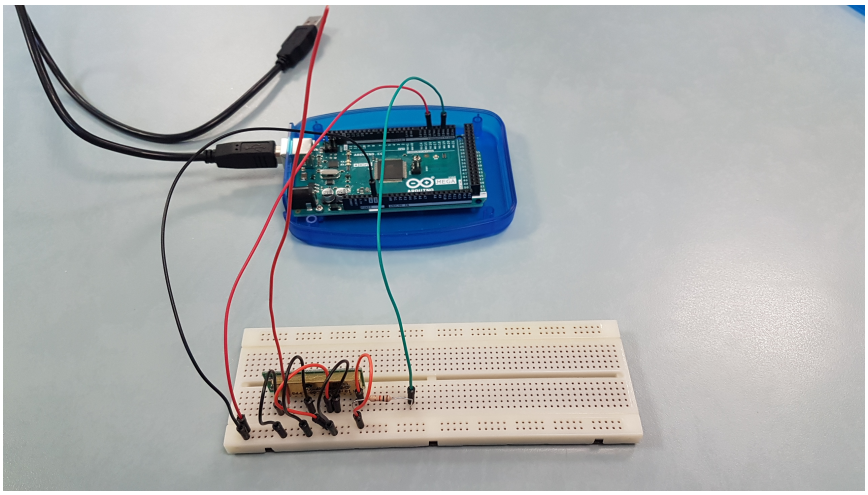

FIGURE 4 – Interface RFLink → openHAB

6 Conclusion

Au final nous sommes très satisfaits de cette implémentation de la SAÉ 203. Sous couvert d'un projet "à la mode" (domaine de l'IoT) les étudiants ont pu percevoir la complémentarité des compétences acquises en 1ère année du BUT RT dans le domaine de l'informatique, des réseaux et des systèmes. Et si certains doutaient encore d'avoir appris "quelque-chose" durant cette année, il aura suffi de leur poser la question : *"Penses-tu qu'il y a un an tu aurais été capable de faire tout ceci ?"*

Les retours ont tous été très positifs, même de la part des étudiants pour lesquels l'informatique n'était pas une vocation. À défaut d'avoir un coup de cœur, ils ont pu constater que sans être des spécialistes en informatique ils peuvent parfaitement répondre à un cahier des charges en ne codant que de petits programmes qui s'échangent des informations et concevoir ainsi une solution informatique complète.