POLYTECH LILLE

Université de Lille 1 SCIENCES ET TECHNOLOGIES

Projet IMA4 P6 : LoRaWAN

Gosse Antoine
Informatique Microélectronique Automatique 4
Année universitaire 2017/2018

Remerciements

Je remercie mes professeurs référents, Thomas Vantroys, Xavier Redon et Alexandre Boé, pour leur soutient et leur disponibilité durant ce projet.
Leur aide et leur conseils ont été des éléments clés qui m'ont permis de mener à bien ce projet.

Sommaire

Introduction

On estime à 20 milliards le nombre d'appareils connectés d'ici 2020. Parmis ces objets, un grand nombre d'entre eux feront partie de ce qu'on appelle l'Internet des Objets (IoT) permettant de bâtir les villes et bâtiments connectés.

Les technologies existantes comme le wifi ne sont pas forcément adaptées à cause de la portée et de la consommation énergétique principalement. Il devient alors nécessaire de développer des nouvelles technologies conçues spécialement pour ces objets connectés. La technologie LoRaWAN (Long Range Wide Area Network) est une de ces nouvelles technologies permettant de créer des réseaux pour les objets connectés.

Cette technologie permet de créer des réseaux sans fils, sécurisés, longue portée (en théorie 2km en zone urbaine et 15km en zone rurale), basse consommation et à moindre coût puisqu'il est possible de créer son réseau privé ne nécessitant pas d'abonnement.

Elle s'appuie sur la modulation LoRa, une modulation à étalement de spectre sur la bande de fréquence libre Informatique Scientifique et Médicale à 868 Mhz en Europe. LoRa constitue donc la couche physique du protocole et a été créé par la startup française Cycléo puis racheté par la société américaine Semtech en 2012.

Le protocole LoRaWAN est quant à lui développé par l'alliance LoRa qui compte plus de 100 membres (des sponsors comme Semtech, IBM, Cisco, des contributeurs comme Microchip, des revendeurs comme IMST).

Chapitre 1

Analyse du projet

1.1 Positionnement par rapport à l'existant

De nombreux projets utilisent déjà la technologie LoRa mais se basent sur le serveur public The Thing Network. On fait ici le choix d'utiliser un serveur privé afin de contrôler au mieux la sécurité des données. (chiffrées en AES-128 de bout en bout)

1.2 Analyse de la concurrence

A l'heure actuelle, le seul vrai concurrent de LoRa en terme de possibilités est Sigfox, créé en 2009. LoRa arrive donc 3 ans plus tard, mais les nombreux avantages par rapport à son concurrent lui permettent de s'immiscer sur le marché :

Taille maximale des messages : 242 octets (12 pour Sigfox)
Nombre de messages par jour illimités (dans la limite du duty-cycle) (140 msg/jour pour Sigfox)
Immunité contre les interférence élevé grâce à la modulation à étalement de spectre (basse pour Sigfox)
Coût d'installation et de maintenance faible (Sigfox nécessite un abonnement)
Open Source, du moment qu'on achète les puces (Sigfox est propriétaire)

L'autre concurrent potentiel est la 5G. Toutefois, bien qu'elle soit déjà en développement elle prendra un certain temps avant de se standardiser. Orange à également lancé la LTE-M, une évolution de la 4G spécialement dédiée au objets connectés. LTE-M propose des débits allant jusqu'à 10Mb/s et bénéficie de l'itinérance puisqu'ils se base sur le réseau 4G existant.

1.3 Scénario d'usage du produit ou du concept envisagé

Dans le cadre de sa politique d'amélioration continue, l'école Polytech Lille déploie un réseau LoRaWAN dans son établissement. Ce réseau longue portée et faible consommation permet l'installation de petits appareils dotés de capteurs dans les salles de classe permettant d'obtenir divers informations sur les salles pour réduire la consommation en énergie ou connaître l'occupation des salles.

Des capteurs de luminosité sont installés dans les salles et permettent de repérer les endroits où les lumières sont allumées. Des capteurs de température permettent de contrôler la température des salles. Des capteurs de contact installés sur les fenêtres permettent de détecter celles restées ouvertes. Des capteurs de présence permettent de savoir si la salle est occupée.

Le réseau est composé d'un concentrateur LoRa, relié à une RaspberryPi3 (installée dans un local technique de l'école) embarquant un programme de conversion des paquets LoRa en paquets UDP. Les capteurs, associés à des microcontrôleurs de type Nucleo-F4 équipés de radios LoRa vont envoyer les informations au concentrateur. Celui-ci par le biais de la RaspberryPi va envoyer les information sous forme de paquets TCP au serveur.

L'utilisateur cible est l'appariteur de l'école, il pourra contrôler depuis une page web les différentes statuts des capteurs lors de la fermeture le soir : fenêtres fermées, lumières éteintes, chauffage baissé raisonnablement, salles vides.(figure 1.1)

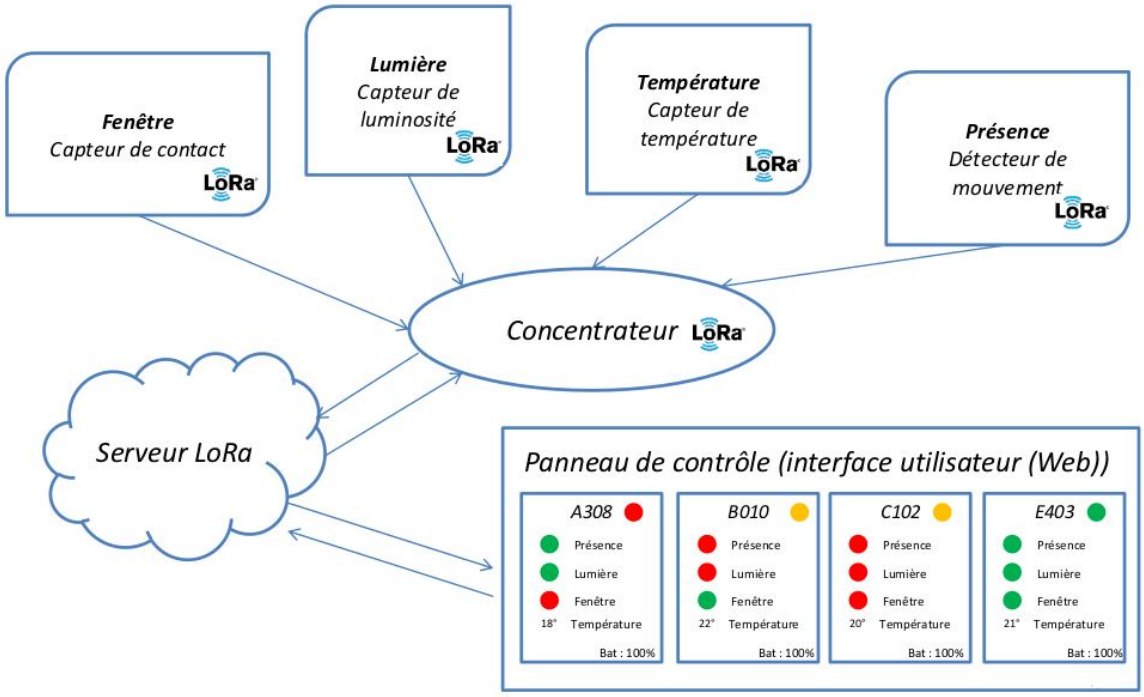


FIGURE 1.1 – Scénario d'usage

Chapitre 2

Réalisation du projet

2.1 Réseau LoRaWAN

La première étape du projet consiste au déploiement du réseau. Un réseaux LoRaWAN est constitué de 4 éléments caractéristiques : les noeuds ou objets connectés (nodes), les passerelles (gateway) embarquant des concentrateurs, le serveurs LoRaWAN (Network Serveur) et le serveurs applicatif (Application Server). (figure 2.1)

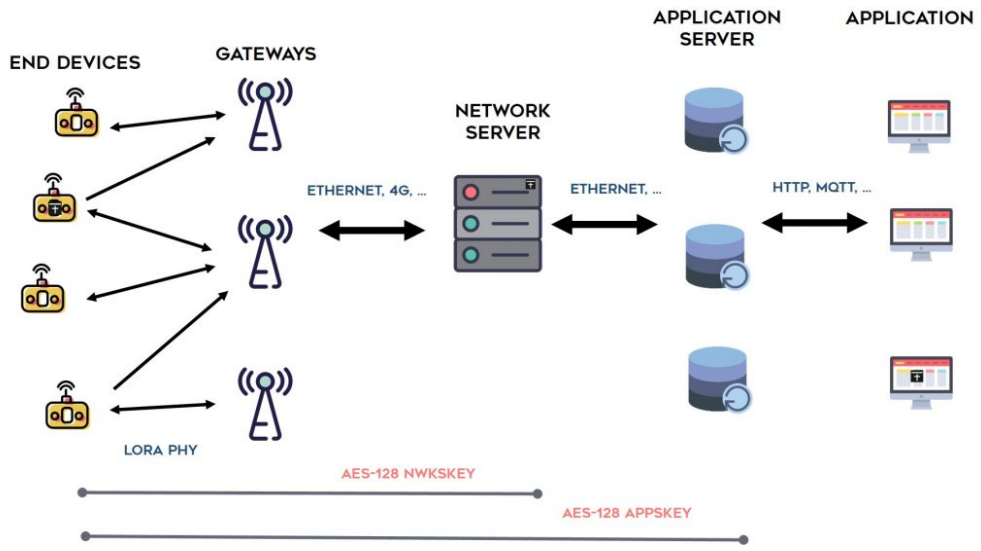


FIGURE 2.1 – Architecture classique d'un réseau LoRaWAN

2.1.1 Noeuds

On distingue 3 classes de noeuds : A, B, C différenciées par leur facultée à recevoir des informations provenant du serveur. En effet, les réseaux LoRaWAN sont surtout prévus pour les communications montantes (UpLink : depuis les noeuds vers le serveur) mais peuvent aussi dans certains cas recevoir des informations descendantes (DownLink : depuis le serveur vers le noeud)(figure 2.2)

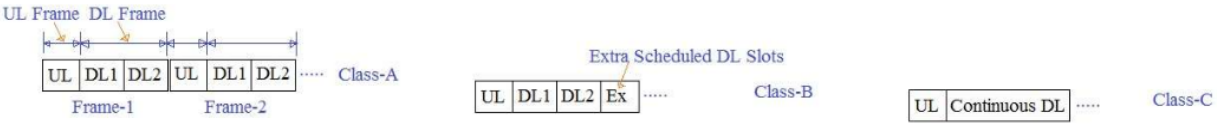


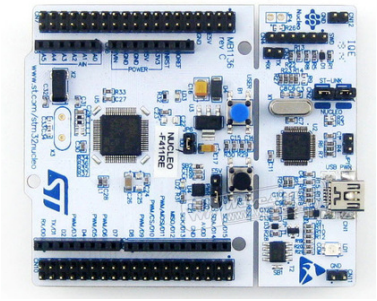
FIGURE 2.2 – Classes LoRaWAN

Les noeuds de classe A peuvent recevoir des informations provenant du serveur après une communication montante. Il ouvrent alors 2 fenêtres de réception après un envoie.

Les noeuds de classe B constituent une extension aux noeuds de classe A dans la mesure où ils disposent d'une fenêtre de réception supplémentaire.

Les noeuds de classe C sont eux capable de recevoir en permanence sauf lorsqu'ils émettent, ils consomme alors davantage que les noeud des autres classes.

Dans ce projet, les noeuds sont constitués d'un microcontrôleur Nucleo-F401-RE de chez ST, d'une radio LoRa I-NUCLEO LRWAN1 de chez USI et d'un capteur de température/humidité relative de type DHT11. (figure 2.3)



(a) Nucleo F401-RE

(b) I-Nucleo LRWAN1

(c) DHT11

FIGURE 2.3 – Éléments de l'objet connecté

La radio LoRa I-NUCLEO LRWAN1 permet la création de noeuds de classe A. Cette radio sous forme d'un bouclier Arduino (également compatible Nucleo) implémente la spécification LoRaWAN 1.0.1, La radio LoRa comporte les informations nécessaires à l'identification du noeud dans le réseau :

DevEUI : numéro d'identification unique.
AppEUI : numéro d'identification de l'application associée au noeud.

Pour être reconnue par le réseau, le noeud doit être personnalisé et activé, ceci peut soit avoir lieu lors d'une activation par personnalisation (ABP) soit par une activation "Over The Air" (OTAA).

La méthode ABP consiste à stocker les clés de session serveur NwkSKey et de session application AppSKey en dur dans l'objet connecté. Elle est maintenant dépréciée puisqu'elle présente des failles de sécurité importantes. Dans ce cas l'objet contient deux clés : une NwkSKey (clés de session serveur) et une AppSKey (clés de session application).

Pour l'OTAA, les objets connectés commencent par envoyer leur DevEUI, APPEUI et un code d'intégrité de message (MIC) à la passerelle. Celle-ci envoie transmet la requête au loraserver. Ce dernier va demander au serveur applicatif s'il connaît l'EUI et l'AppKey de l'appareil. Si oui le serveur envoie un join-accept et va dériver des clés de session (AppSKey et NwkSKey) à partir de l'AppKey. Ces clés seront utilisés pour établir une communication sécurisée entre l'appareil et les serveurs. Si la session est interrompue, par example lors d'une mise hors tension de l'objet connecté il faudra de nouveau effectuer la join-request. Dans ce mode, l'objet comporte une AppKey.(figure 2.4)

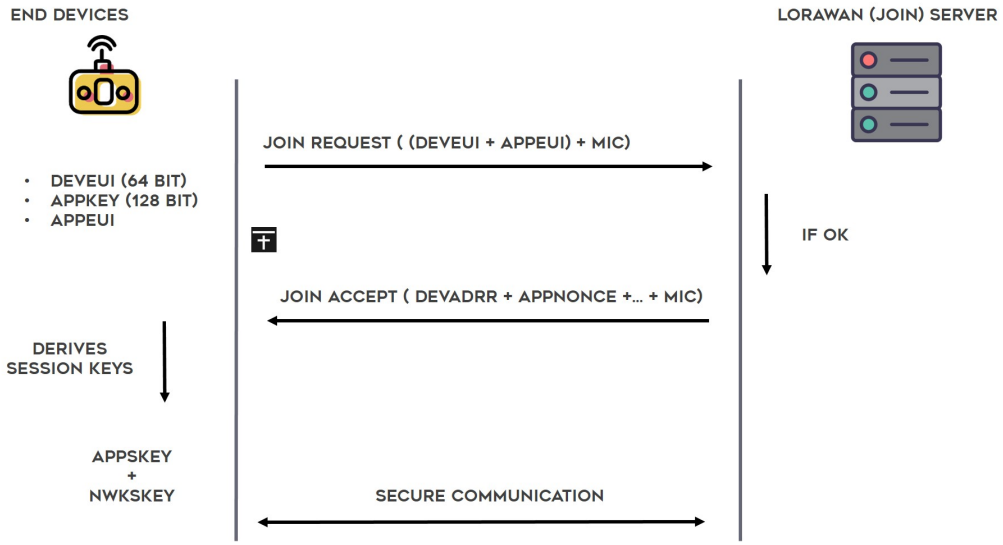


FIGURE 2.4 – LoRaWAN OTAA

Les données sont chiffrées à l'aide de la NwkSKey et de l'AppSKey. Ce double chiffrement offre une sécurité renforcée sur les réseaux, et permet de créer des réseaux fédérés comme TheThingNetwork où chacun peut ajouter une passerelle pour agrandir ce réseau mais où les données ne seront pas dévoilées ni altérés par les autres utilisateurs. La clés NwkSKey

permet de chiffrer jusqu'au serveur LoRaWAN et la seconde l'AppSKey permet de chiffrer jusqu'à l'utilisateur final.

Pour communiquer avec la radio on utilise son port série pour lui envoyer des commandes AT. Le Nucleo-F4 est alors utilisé pour automatiser l'envoi de ces commandes en les envoyant sur le port série du bouclier I-NUCLEO LRWAN1. Les Nucleo-F4 ont été programmés par le biais du compilateur os.mbed.com disponible en ligne.

Les noeuds peuvent soit faire partie d'un réseau public (géré par un opérateur de télécommunication) ou privé comme c'est le cas dans ce projet.

Le Duty Cycle défini le pourcentage du temps pour lequel l'objet occupe la bande de fréquence. En Europe il est réglementé à 1% sur la bande 868 MHz. Dans ce projet il à été désactivé pour faciliter les tests, mais il serait préférable de le réactiver si l'application venait à être utilisée en situation réelle.

Le débit adaptatif (ADR) est un mécanisme permettant d'optimiser le débit du signal, la consommation énergétique et le temps d'occupation de la bande de fréquence.

Les commandes sont envoyés dans cet ordre :
A l'initialisation de l'appareil :
AT+NTYP=0 permet de définir le réseau comme réseau privée
AT+AK=<clé> permet d'attribuer une clé d'application
AT+DC=0 permet de désactiver le duty cycle
AT+ADR=1 permet d'activer débit adaptatif
AT+JOIN=1 permet d'envoyer une "join request" au serveur LoRaWAN

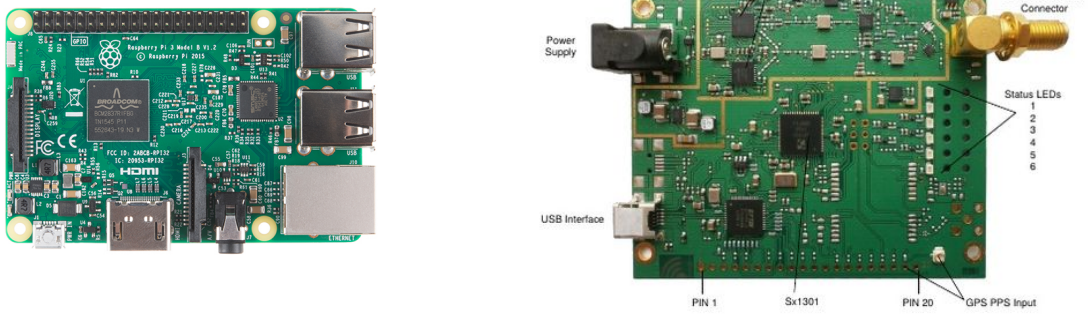
Périodiquement : AT+SEND=1,<message>,1 permet d'envoyer le <message> LoRaWAN contenant les valeurs des capteurs chiffrés en AES128 sur le port 1 et avec accusé de réception. Le message est codé sur 3 octets : le premier contient l'identifiant de l'appareil qui apparaîtra sur l'application de l'utilisateur, le second contient la température en degrés Celcius et le troisième contient le taux d'humidité relatif en pourcentage. Les capteurs peuvent varier de 0 ° C à 50 ° C avec un précision de 1 ° C, et le taux d'humidité relative peut varier de 20% à 95% avec une précision de 1% .

Ceci permet la création d'un objet prêt à l'emploi, il suffit de le brancher pour qu'il rejoigne le réseau sur lequel il aura été au préalable enregistré et commence à émettre.

2.1.2 Passerelle

La passerelle est l'élément de transition entre les paquets LoRa et les paquets UDP envoyés au serveur. Elle est constituée d'un hôte et d'un concentrateur, élément de télécommunication permettant de recevoir/émettre les paquets LoRa.

Dans ce projet, l'hôte est une RaspberryPi3 embarquant une distribution Raspbian Stretch, elle est reliée en SPI à un concentrateur IC-880A SPI de chez IMST. (figure 2.5)



(a) RaspberryPi3 (b) IC-880A SPI

FIGURE 2.5 – Éléments de la passerelle

Au début de ce projet une Raspbian Jessie avait été installée, bien que recommandée par IMST elle posait des problèmes de connexions SPI avec le concentrateur.

La RaspberryPi embarque un "packet forwarder" provenant du GitHub de Semtech Loranet [2]. Ce packet forwarder est un programme qui tourne en continue sur la RaspberryPi, son rôle est de transmettre les paquets LoRa provenant du concentrateur vers des paquets UDP qui seront envoyés au serveur. Il est également capable de procéder à l'opération inverse lors de communications descendantes.

La RaspberryPi est branchée en Ethernet sur le réseau de l'école et à pour IP fixe 172.26.145.111.

Un service à été créé afin d'automatiser le démarrage du "packet forwarder" lors de l'allumage de la RaspberryPi. Ce service est constitué d'un script qui va tout d'abord effectuer un reset du concentrateur et va ensuite tenter de démarrer le packet forwarder. Comme indiqué dans le guide de démarrage d'IMST [3], depuis la version 4 du noyau de RaspberryPi, un problème lié à la fréquence du SPI peut provoquer une erreur lors du démarrage du packet forwarder. Le script va donc effectuer jusqu'à 5 tentatives de démarrage du packet forwarder.

Ceci permet d'obtenir une passerelle prête à l'emploi lors de l'allumage, à condition qu'elle soit branchée sur le réseau de l'école au préalable.

2.1.3 Serveur LoRaWAN

Le serveur LoRaWAN est l'élément implémentant la spécification LoRaWAN, notamment pour déchiffrer les messages et pour gérer les connexions. Ce projet open source utilise l'implémentation de serveur LoRaWAN "loraserver" provenant du GitHub d'Orne Brocaar sous licence du MIT [1].

Cette implémentation composée d'une suite de trois éléments est installée sur la RaspberryPi, elle contient : un LoRa Gateway Bridge, un Loraserver et un Lora App Server permettant de créer un réseau LoRaWAN privé.

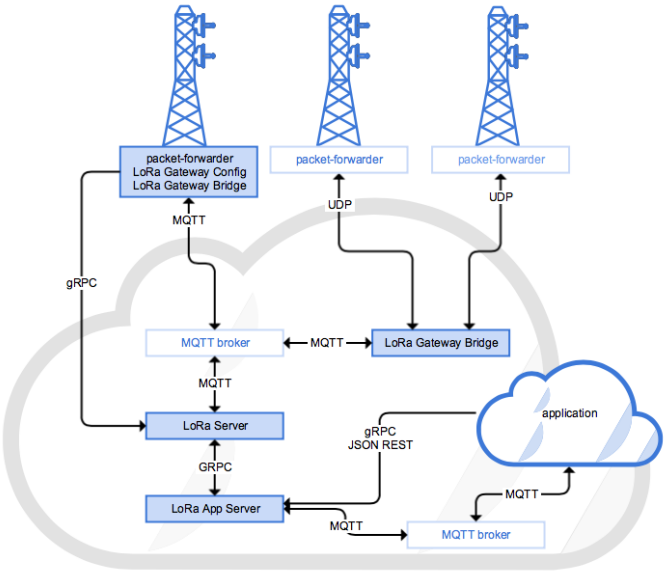


FIGURE 2.6 – Principe de fonctionnement du loraserver

Le Lora Gateway Bridge est un service qui a pour but de faciliter la transmission des paquets UDP provenant du packet forwarder de Semtech. Ce gateway bridge va transformer les paquets UDP en JSON par le protocole MQTT.

Le loraserver constitue le coeur du réseau. Il gère les sessions en cours des objets connectés. Lorsqu'un nouveau noeud tente de rejoindre le réseau, il demande au serveur applicatif si le noeud en question est autorisé à rejoindre et si oui quelle configuration il doit utiliser pour ce noeud.

Comme il est possible d'utiliser plusieurs passerelles, il est également responsable de la déduplication des données. Il va ensuite authentifier les données et leur attribuer un numéro, le Fcnt (Frame counter) pour éviter les rejeux, et enfin envoyer les données chiffrées au serveur applicatif.

Enfin il est en charge d'envoyer les commandes de paramétrage (commande MAC) au noeud pour par exemple changer le débit, ou le canal utilisé.

2.1.4 Serveur applicatif

Dernier élément de l'implémentation loraserver, le serveur applicatif est l'élément permettant à l'administrateur de pouvoir gérer son réseau. Il est responsable du déchiffrement des données finales de l'utilisateur. Il est fourni avec une interface web permettant de configurer le réseau, d'ajouter des organisations contenant des applications contenant elles même des appareils. Il est également possible de créer des profils utilisateurs et de leur définir des permissions. Enfin le serveur applicatif permet de visualiser les trames des messages provenant des appareils.(figure 2.7)

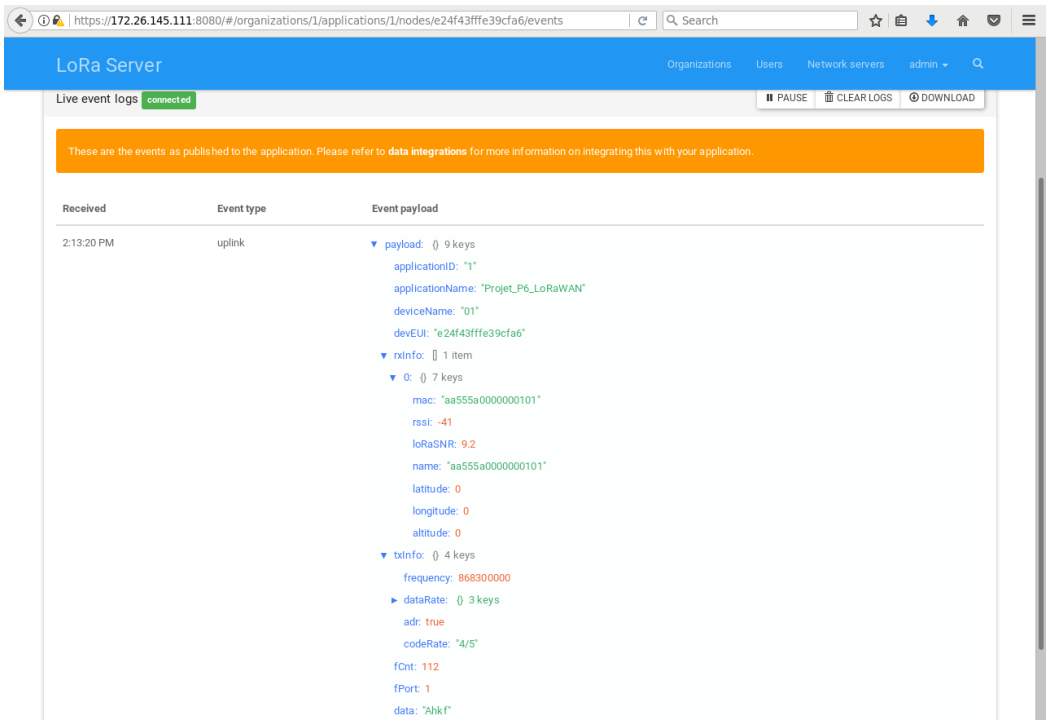


FIGURE 2.7 – Réception des trames depuis le serveur applicatif

Sur l'exemple ci dessus le champ data contient le message "Ahkf" en base64. Cela donne 02 19 1f après passage en hexadécimal. Selon le protocole du projet, cela signifie que l'appareil avec l'identifiant 2 vient d'envoyer une température de 25 ° et un taux d'humidité relative de 31%.

2.2 Récupération des données

La partie LoRaWAN à proprement parler s'arrête au serveur applicatif, mais il convient ensuite d'intégrer les données reçues afin de donner une réelle utilité à notre réseau. Toute la partie de récupération des données se trouve sur une machine virtuelle sur les serveurs de Polytech.

2.2.1 Méthodes d'intégration proposées par Loraserver

L'implémentation Loraserver est en développement continu, au moment du projet elle permet uniquement l'intégration des données par le biais de requêtes HTTP "POST".

Cette méthode d'intégration est assez facile à mettre en oeuvre puisqu'elle se paramètre directement sur le serveur applicatif, cependant, elle présente des problèmes. En effet, il n'y a pas de sécurité, un programme malveillant peut envoyer des requêtes POST en se faisant passer pour le serveur (puisqu'il s'agit de simples requêtes POST sans sécurités supplémentaires).

Il existe toutefois une autre méthode pour récupérer les données. En effet, les différents composants communiquent par le protocole MQTT, un protocole basé sur TCP/IP conçu pour les objets connecté.

Dans ce protocole les clients communiquent par le biais d'un "broker" installé sur la machine virtuelle, il peuvent publier ou souscrire à des sujets, et communiquent de cette manière.(figure 2.8) Le port 1883 à été ouvert par le service informatique afin de permettre cette communication, elle été bloquée par défaut.

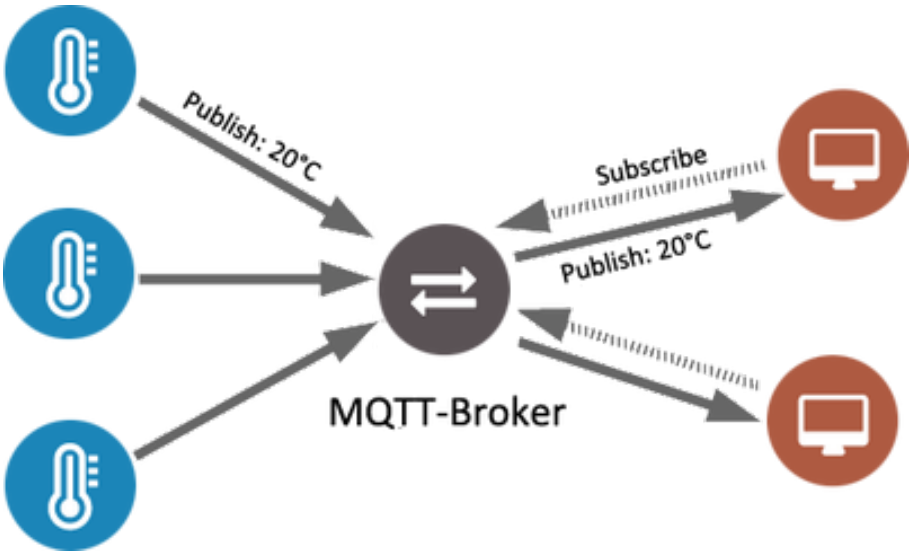


FIGURE 2.8 – Principe de fonctionnement de MQTT

Dans ce projet, le client MQTT Mosquitto est utilisé sur Debian, le sujet "#" définit l'ensemble des sujets du broker. De cette manière on peut voir tous les échanges se réalisant entre les différentes partie du loraserver.

Si on souscrit au sujet application/1/node/+/rx (le symbole "+" est le joker, ce qui permet de souscrire aux sujets de tous les noeuds de l'application "1") on verra l'ensemble des messages reçus par l'application, en JSON, contenant le message déchiffré codé en base64.

Le protocole MQTT présente l'avantage de pouvoir être utilisé en mode authentifié, et d'interdire les souscriptions anonymes. Ce mode s'active en modifiant les fichiers de configuration du broker sur la machine virtuelle. On indique alors un fichier contenant les utilisateurs autorisés (ce fichier contient également les mots de passe chiffrés) et les ACL (Access Control List) définissant pour chaque utilisateur les permissions en lecture écriture sur les différents sujets.

2.2.2 Web serveur et HTTPS

La machine virtuelle est équipée d'un serveur Apache2. Comme elle est partagée avec le groupe P5, les documents web sont dans le dossier /var/www/html/P6.

Pour permettre l'accès en HTTPS, un nom de domaine à été associé à l'IP du serveur, https ://trappe.space sur lequel est accessible l'application de démonstration du projet. L'accès au dossier P6 est protégé par authentification (login : admin, mot de passe : p0lytech). Ce nom de domaine à été réservé sur Gandi.net, le champ "@" pointant vers l'adresse publique 193.48.57.232.

Le serveur est paramétré pour chiffrer les connexions en HTTPS. Pour cela, des certificats SSL/TLS gratuits et reconnus ont été créés avec letsencrypt grâce à l'utilitaire certbot. Ils ont été renseignés sur les serveurs de l'école responsable de cela. Le domaine est enregistré sous antoine.gosse@polytech-lille.net et est valable 90 jours (renouvelable).

2.2.3 Script PHP : client MQTT

Il est possible de visualiser les différents sujets MQTT grâce au client Mosquitto. Cependant, il faut pouvoir récupérer les messages. Ceci passe par le biais d'une extension PHP "Mosquitto-PHP". un script PHP fonctionne alors en continu agissant comme un client, et permettant de stocker la partie données du message dans des fichiers ainsi qu'un horodatage.(timestamp linux)

La page web est codée en HTML/CSS, elle utilise le framework "bootstrap" pour l'affichage des barres de progression nottament. Les graphiques utilisent chart.js (JavaScript) et PHP principalement.

L'utilisation de l'utilitaire "screen" permet de faire tourner le script dans un terminal séparé, toujours actif même après avoir fermé la console SSH.

Chapitre 3

Résultats obtenus

Le projet est fonctionnel, l'utilisateur à uniquement besoin de brancher la RaspberryPi au réseau de l'école, puis de brancher les objets connectés. Pour les 2 capteurs déjà enregistrés sur le loraserver il n'y a aucune manipulation supplémentaire à effectuer.

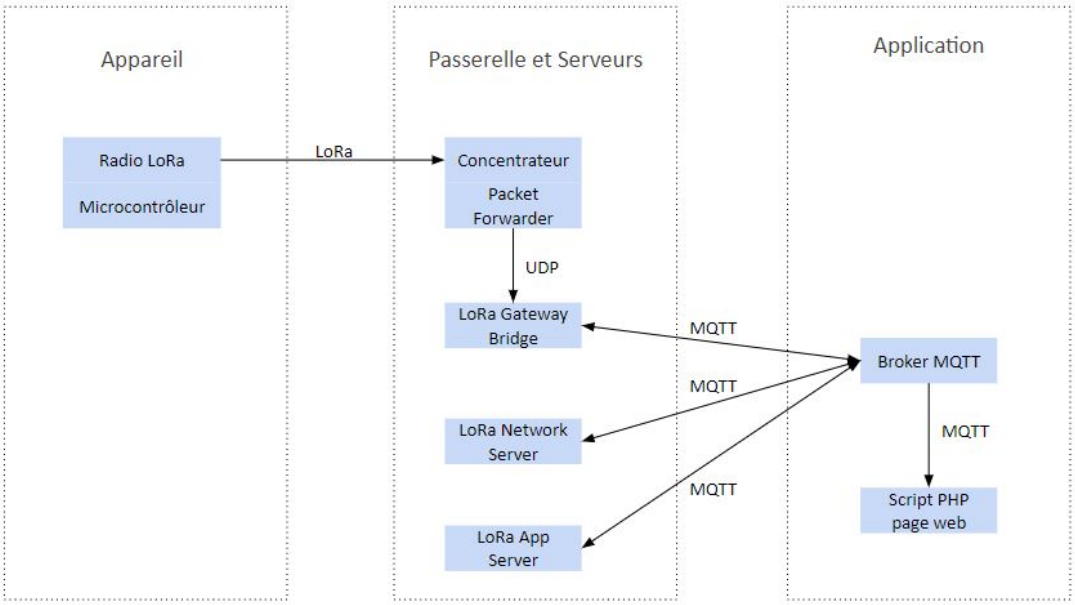


FIGURE 3.1 – Architecture finale

3.1 Procédure d'ajout d'un nouveau noeud

S'il le souhaite, l'utilisateur peut ajouter un nouveau noeud au réseau.

Il faut tout d'abord connaître le DevEUI du noeud, dans ce projet ceci passe par les commandes et AT+EUI sur la laison série du bouclier LoRa. Il faut ensuite modifier l'identifiant de l'objet dans le code du microcontrôleur afin que celui-ci soit reconnu par l'application finale, il faut également choisir une AppKey dans le code.

Une fois ces informations connues, il faut s'authentifier sur le serveur applicatif à l'adresse https ://172.26.145.111 :8080, (login : admin, mot de passe : p0lytech)

Enfin, dans l'application shouhaitée, on ajoute une nouveau noeud en indiquant son DevEUI et son AppKey. Une fois cette étape faite on peut brancher le noeud et la connexion se fera de manière automatique. (Grâce à l'envoi des commandes AT par le noeud expliqué dans la partie précédente). Un nouveau graphique est les données associées au capteur aparaîtront alors automatiquement sur la page.

3.2 Application web

L'application web constitue le point final du projet, c'est grâce à cette interface, que l'utilisateur va pouvoir visualiser les données de ses capteurs après s'être authentifié.

L'application propose pour chaque objet, un graphique contenant l'ensemble des valeurs des capteurs (humidité/température). Si l'objet est actif (il a envoyé une donnée dans les 3 minutes passées) il apparaît avec la mention "Online".(figure 3.2)

Passé ce délai, l'objet passe en mode déconnecté et apparaît alors avec la mention "Offline". Les données temps réel disparaissent pour laisser place au temps écoulé depuis la dernière connexion.(calculé à partir du timestamp linux)

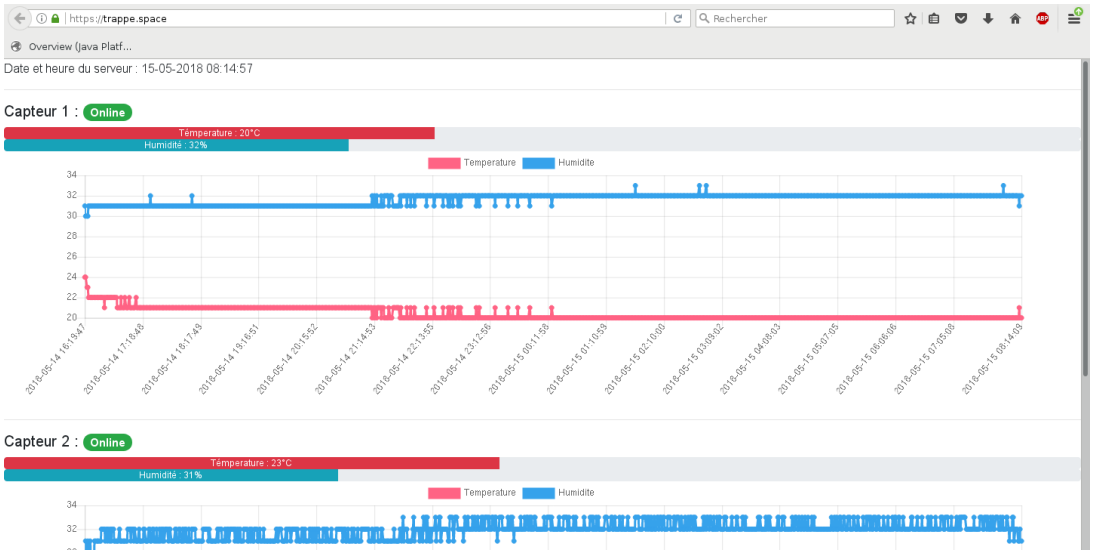


FIGURE 3.2 – Panneau de contrôle utilisateur

3.3 Difficultés rencontrées

Durant ce projet, de nombreuses difficultés ont été rencontrées, certaines d'entre elle ont été longues à résoudre. Heureusement, une fois résolues elles se sont révélées très

formatrices.

Une des premières difficultés était la RaspberryPi, n'en ayant jamais utilisé et flashé, la moindre manipulation était au début hasardeuse et prenait du temps. Je me suis donc documenté et j'ai réussi à obtenir ce que je souhaitais. J'ai notamment rencontré des difficultés pour lancer un script au démarrage de la RaspberryPi. Même en ajoutant le script dans le fichier rc.local je n'y suis pas parvenu, le script était pourtant correcte mais ne se démarrait pas avec la RaspberryPi. J'ai résolu ce problème en créant un service avec l'utilitaire systemctl.

Une difficulté majeure à été rencontrée lors du déploiement de l'objet connecté. Je n'avais jamais utilisé de Nucleo mais leur programmation n'a pas constitué de difficulté. En revanche c'est lors de l'utilisation du bouclier I-NUCLEO LRWAN1 que les problèmes sont survenus. Ce bouclier à le défaut d'être très peu documenté, de même pour le package I-CUBE LoRaWAN supposé contenir des exemples, les seules ressources sont disponibles sur le GitHub d'USI [5] et certains éléments ont été rajoutés durant le projet, après les recherches initiales.

D'autres interfaces de développement on été également testé comme l'IDE gratuit basé sur Eclipse pour STM32 "System Workbench for STM32" ou "Atollic TrueSTUDIO STM32" mais dans il était impossible de charger le projet sur TrueSTUDIO et sur l'autre IDE même en ayant réussi à charger le projet, des erreurs empêchaient la compilation. Il semble que ce code soit prévu pour être implémenté directement sur le Nucleo-F0 intégré dans le bouclier I-NUCLEO LRWAN, mais aucune certitude à ce sujet. Des utilisateurs sur le GitHub d'USI ont expliqué avoir réussi à implémenter le code en flashant par SWD ou JTAG le Nucleo-F0.

A l'aide d'un programmeur JTAG j'ai tenté de sauvegarder le firmware du Nucleo-F0 du bouclier à l'aide de l'utilitaire OpenOCD. Mais au moment de la sauvegarde, un message d'erreur est survenu indiquant que le Nucleo-F0 était protégé et que la suppression de cette protection entraînerait la remise à zéro du contenu de ce microcontrôleur. Ne voulant pas prendre le risque de rendre le bouclier inutilisable, et ayant déjà passé beaucoup de temps sur ce problème j'ai préféré abandonner cette méthode et me concentrer sur l'envoie de commandes AT par laison série qui elle fonctionnait.

Un problème mineur à été rencontré en fin de projet. Il était impossible d'accéder aux valeurs des 3 capteurs embarqués sur le bouclier (pression, magnétique, température/humidité relative).

Les commande AT+SIIC pour initialiser la liaison I2C et AT+RIIC=0xbf,0,1 (address I2C 0xbf, registre 0, lecture d'1 octet) ont été testé comme indiqué sur le forum GitHub d'USI, mais même résultat que l'utilisateur ayant posé la question, cette commande

bloque le bouclier et il faut le redémarrer. La solution à ce problème à été d'utiliser un capteur externe de type DHT11 (température/humidité relative) branché sur le Nucleo-F4 directement.

Un soucis à également été rencontré lors de l'envoi des messages au loraserver, le serveur refusait les join-request de l'appareil lorsque la RaspberryPi était connecté au réseau de l'école. Aucun soucis sur un réseau personnel. La solution à donc été de réinstaller le loraserver. Le problème était en fait une mise à jour incomplète d'un des composants de ce dernier ayant provoqué la suppression de certains fichiers de configuration du loraserver. Une fois réinstallé le loraserver acceptait de nouveau les join-request.

3.4 Ouverture envisageable et remarques

Du fait des problèmes rencontrés ayant pris beaucoup de temps à résoudre, il n'a pas été possible de tester la vulnérabilité du réseau. On trouve sur internet des documents à ce sujet, notamment une thèse [6] qui présente les différentes attaques que peut subir un réseau LoRaWAN, la majeure partie d'entre elles sont toutefois liés à l'ABP. On trouve également des test réalisé par le blog MISCmag [4] qui a deployé une infrastructure de test.

Pour bien faire il faudrait décortiquer la spécification LoRaWAN et étudier le protocole dans ses moindres recoins pour déceler d'éventuelles failles de sécurité, ceci à tous les niveaux du protocole.

Enfin des tests de portée ont été réalisés en déplaçant l'objet connecté dans l'établissement. La portée n'est pas aussi grande en pratique qu'en théorie, sans doute à cause de la structure interne de l'école.

Conclusion

J'ai choisi le sujet P6 LoRaWAN car j'avais entendu parler de cette technologie. Ne connaissant pas du tout son mode de fonctionnement, ce projet était donc l'opportunité de découvrir une technologie relativement récente. J'ai maintenant une idée précise du fonctionnement de la technologie LoRaWAN, du moins des différents éléments qui composent un tel réseau. Cette idée à en tout cas bien évoluée depuis les recherches préliminaires en début du projets comme l'atteste le wiki. C'est notamment au niveau de la partie serveurs qui me paraissait totalement nébuleuse en début de projet que ma compréhension s'est accrue.

Ce projet s'est également révélé très formateur sur plusieurs points. Il m'a permis de découvrir les RaspberryPi, de leur installer un système d'exploitation et de les utiliser pour parvenir à mes fins. Il est fort probable que je réutilise ce micro ordinateur à l'avenir.

La découverte des microcontrôleurs Nucleo-F4 à également marqué ce projet. Ces objets équipés de STM32 peuvent être programmés depuis le compilateur en ligne os.mbed.com et finalement s'apparente à de la programmation Arduino du fait du haut niveau de langage.

Enfin j'ai acquis de nombreuses connaissance côté web, j'ai appris à associer un serveur Apache à un nom de domaine, j'ai créé des certificats SSL/TLS letsencrypt reconnus pour activer les connexions HTTPS, j'ai découvert le PHP et le protocole MQTT et j'ai utilisé le JavaScript pour les graphiques.

Les nombreuses connaissances acquises tout au long de ce projet notamment au niveau de la technologie LoRaWAN, les problèmes rencontrés et les solutions apportées ont renforcé mon expérience de jeune ingénieur en devenir.

Sources

[1] Orne Brocaar. *GitHub loraserver*. URL : https://github.com/brocaar/loraserver.

[2] Semtech Corporate. *GitHub Lora-net*. URL : https://github.com/Lora-net.

[3] IMST GmbH. *iC880A-SPI QuickStartGuide*. URL : https://wireless-solutions.de/downloads/Radio-Modules/iC880A/iC880A-SPI_QuickStartGuide.pdf.

[4] MISCRedac. *LORAWAN : déploiement d'une infrastructure de test*. URL : https://www.miscmag.com/lorawan-deploiement-dune-infrastructure-de-test-partie-1-2.

[5] USI IoT Solution. *GitHub USI I-NUCLEO-LRWAN1*. URL : https://github.com/USIWP1Module/USI_I-NUCLEO-LRWAN1.

[6] Yang Xueying. *LoRaWAN : Vulnerability Analysis and Practical Expoitation*. URL : https://repository.tudelft.nl/islandora/object/uuid:87730790-6166-4424-9d82-8fe815733f1e.