Rapport Jalon 2 | Sécurisation d'un réseau LoRaWAN

Version 0.1

Arthur Le Rest François Sevaux

oct. 25, 2019

Table des matières

CHAPITRE 1

Introduction

Projet M1 CSSE *mise en oeuvre d'un réseau LoRaWAN sécurisé* 2019-2020

Ceci est la documentation de notre projet de sécurisation d'une communication LoRaWAN.

Contexte :

L'Internet des objets (IoT, en anglais) est un paradigme dont les premiers déploiements ont quelques années (voire plus, si l'on parle de réseau de capteurs). D'un point de vue sécurité, l'IoT a une surface d'attaque très importante, du fait du nombre de technologies, de protocoles, du type de déploiement et du nombre d'acteurs différents. Ce projet s'applique aux réseaux d'objets connectés longue porté du type LoRaWAN (Long Range Wide Area Network).

Mission :

Dans ce contexte, il nous est demandé de mettre en place un réseau LoRaWAN sécurisé. Le premier et principal objectif est de créer un réseau LoRaWAN complet, mais simple et fonctionnel, dont les éléments de sécurité côté noeud et passerelle seront correctement mis en oeuvre. Toute une démarche de tests unitaires devra être mise en place, pour tester chacune des parties séparément, puis l'ensemble collectivement.

Un cas d'usage, défini avec notre responsable devra être mis en place et les aspects de sécurité devront être bien maîtrisés.

Le deuxième objectif consiste à discuter de la surface d'attaque de notre système. Un aspect analyse est donc demandé en prenant en compte les différentes versions du LoRaWAN, chacun des éléments du système etc.

1.1 Mise au point : vocabulaire

LoRaWAN : > Long Range Wide Area Network. Protocole de communication.

Noeud : > Ensemble de composants qui peuvent recevoir et/ou envoyer de l'information via le protocole de communication LoRaWAN. Branche initiale d'un réseau LoRaWAN. Par exemple, un capteur relié à une carte/microcontrôleur et une antenne pour la communication vers l'extérieur.

Passerelle ou *Gateway* : > Élément de transfert. Permet de traduire et transférer les données venant du noeud vers les serveurs.

Network server : > Cerveau du réseau LoRaWAN, il génère les clefs et authentifie les noeuds. Il déchiffre aussi une partie des trames du réseau, reçues via la passerelle.

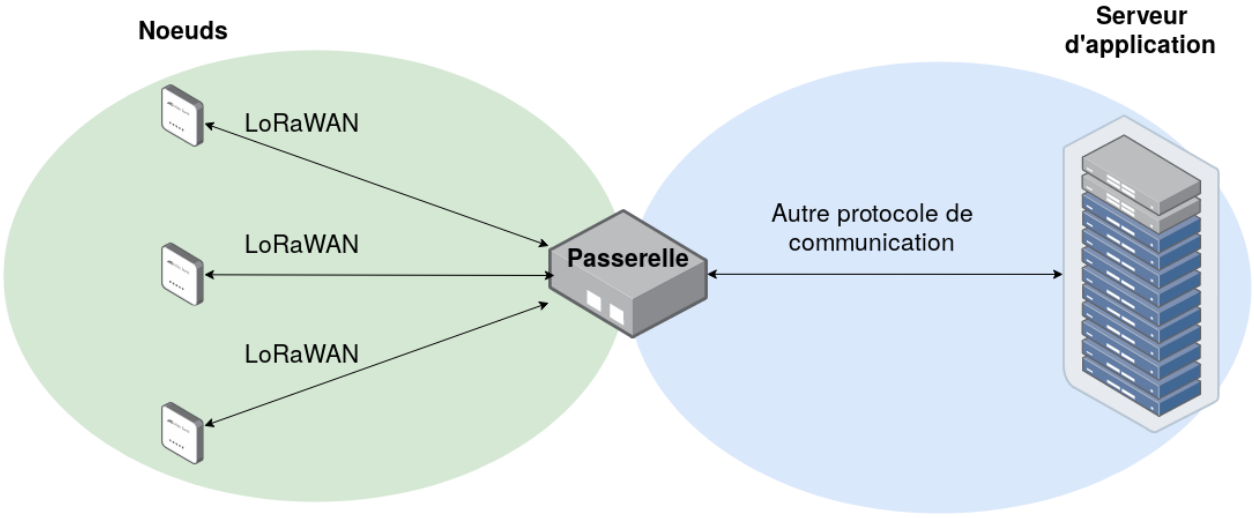
Application server : > Service qui va traiter l'information du capteur, il va déchiffrer la dernière partie du message.

Box LoRa : > Pour notre projet, sera un micro-ordinateur *Raspberry* qui va contenir la passerelle, le *network server* et l'"*application server*

1.2 Schéma simplifié

Le schéma ci-dessous est un schéma simplifié, permettant de comprendre le fonctionnement global de notre système.

Nous avons donc plusieurs noeuds, qui vont communiquer en LoRaWAN avec une passerelle. Cette passerelle va ensuite communiquer ce qu'elle a reçu des noeuds au serveur d'application, via un autre protocole de communication.



CHAPITRE 2

Spécifications

Le protocole LoRa n'est pas fait pour envoyer de grandes quantités d'information très rapidement. On ne peut envoyer que quelques KiloOctets par intervalle de quelques minutes.

Dans sa version 1.0 le LoRaWAN spécifie déjà plusieurs directives à suivre pour le sécuriser. Il y a une clef ***AES 128bits*** à fournir pour sécuriser la commuication depuis le *noeud* jusqu'au *serveur d'application*. - **AppKey** Clef AES principale. Elle doit être connue *du noeud* et *du network manager*. Elle sert ensuite à déterminer les 2 clefs suivantes. - **NwkSKey** Network Session key : chiffre la communication entre le *noeud* et *le network server*. Elle sert à détecter une éventuelle perte d'information dans le message. - **AppSKey** Application Session key : chiffre le message entre *le noeud* et *le network server*, sans cette clef il est impossible de lire le message.

Les clefs **NwkSKey** et **AppSKey** sont actualisées à chaque nouvelle connexion d'un appareil, elles sont uniques à chaque *noeud* du réseau.

Le LoRaWAN utilise des *frame counter* à fin d'éviter les attaques par répétition. Deux compteurs sont initialisés lorsqu'un nouvel appareil est connecté. Le noeud incrémente le compteur **FCntUP** à chaque fois que qu'il envoit une information sur le *UpLink*. Le Network serveur, lui, incrémente le compteur **FCntDown** à chaque fois qu'il écrit sur le *DownLink*. Pour chaque trame du réseau la valeur des compteurs est envoyée avec. Le récepteur de la trame va comparer la valeur des compteurs à l'intérieur de la trame avec ses propres compteurs et si la valeur des compteurs de la trame est inferieure au compteur du récepteur, ce dernier va ignorer le message.

2.1 Le noeud

Le *noeud* sera composé d'un microcontrôleur, d'un capteur (ou plusieurs) et d'un module permettant la communication en LoRa. Pour le noeud nous allons utiliser un kit de développement provenant de STMicroelectronics.

Nous regroupons ci-dessous les attaques possibles. Ce sera notre base de travail pour sécuriser la communication.

Surface d'attaque : - Gestion des Clés AES - Modification du code source - Interception des données directement sur le capteur - SPA - DPA - Analyse EM - Memory dumping - Valeur des Frame Counters

Les secrets à protéger sont : - La valeur du capteur - Les clefs **NwkSKey** et **AppSkey** et la clef AES **AppKey**

2.2 La passerelle

La *passerelle* sert de traducteur entre le protocole *LoRa* et un autre protocole de communication. Elle sera hébergée sur un micro-ordinateur.

2.3 Network Server

Le *Network server* est le cerveau du réseau LoRaWAN, il génère les clefs et authentifie les noeuds. Il déchiffre aussi une partie des trames du réseau.

Surface d'attaque : - Enregistrement clef AES - Création des clefs **NwkSKey** et **AppSKey**

Secrets à protéger - La valeur des clefs **NwkSKey**, **AppSKey** et **AppKey**

2.4 Application Server

Le *Application server* est le service qui va traiter l'information du capteur, il va déchiffrer la dernière partie du message.

Surface d'attaque : - Réception de la clef **AppSKey** - Gestion de la clef **AppSKey**

Secret à protéger : - La valeur de la clef **AppSKey**

CHAPITRE 3

Points d'action (format poupées russes) :

Mise en place d'un réseau LoRaWAN sécurisé

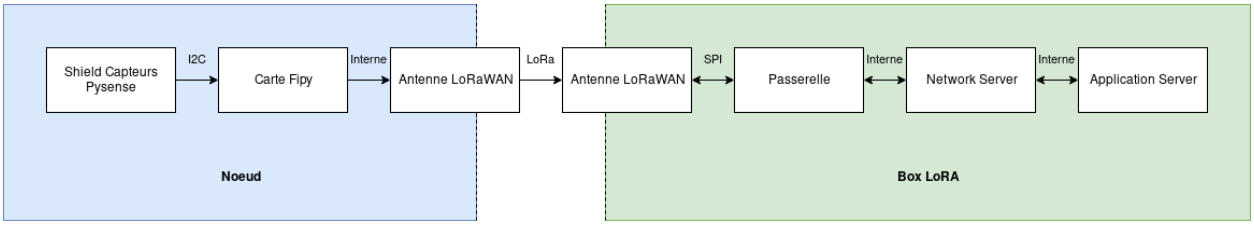
- Mise en place d'un réseau LoRaWAN avec sécurité basique (mot de passe) - Création d'un premier réseau (facile) entre le microcontrôleur *Fipy* et le capteur *Pysense* pour la partie noeud et une Raspberry pour la partie box LoRa. Ce premier réseau nous permet de prendre en main le fonctionnement global du LoRa, sans ajouter les complexités d'une carte STM, en travaillant avec un environnement de noeud plus simple.
  - Rédaction d'un tutoriel pour le déploiement de ce réseau. **Chapitre 9** pour une prise en main simple.
  - Création d'un deuxième réseau identique identique au précédent, mais en remplaçant le noeud par une carte STM32 équipé d'un shield Motion MEMS and environmental (Nucleo expansion board). La finalité de notre réseau est, en effet, d'utiliser une carte STM32 pour le noeud.
  - **Construction des services de la Box LoRa**
    - **Création d'un OS vs. Utilsation d'un OS existant (LoRaServer io)** Notre choix se portera sur l'utilisation d'un OS déjà existant. Sa rapidité de mise en oeuvre et son adaptabilité nous font pencher en sa faveur. Beaucoup de temps de développement est ainsi gagné en prenant l'OS *LoRaserver.io*
    - Choix de prendre un OS *Full*, qui contient *gateway* + *network server* + *application server* en interne, et permet une gestion simplifiée.
- **Tests unitaires de fonctionnement :**
  - **Vérification** [la valeur du capteur est-elle correcte ?]
    - Afficher la valeur de celle-ci dans le terminal et comparer avec la température ambiante réelle.
  - **Vérifier que la valeur est émise correctement.**
    - Émission des trames LoRa. Travail à l'analyseur de spectres de fréquences.
  - **Vérifier que la valeur est arrivée**
    - Réception des trames LoRa, via l'interface graphique.
  - **Vérifier que la valeur est bien transmise dans la box LoRA**
    - Regarder sur la partie *application server* que la valeur est la même que celle affichée dans le terminal du microcontrôleur
- **Mise en place d'un réseau avec des couches de sécurité renforcées**
  - **Sécurisation du noeud**
    - **Développer le software du noeud LoRaWAN**
      - Sécurisation du noeud en cachant les clefs dans la mémoire (composant de sécurité : ATEC508A ou autre)
      - Sécurisation du noeud en cachant les clefs logiciellement

- **Sécurisation de la Box LoRa**
  - **Cas 1** [La box LoRa est le seul composant du serveur]
    - Sécurisation vis à vis d'intrusion externe au systeme : VPN
    - Maintenance : Mise à jours, SSH
    - Verification des services
  - **Cas 2** [La box LoRa ne sert pas uniquement à au LoRa WAN]
    - Sécurisation par rapport aux autres service présent et/ou utilisateurs : Vérifier les droits d'accès

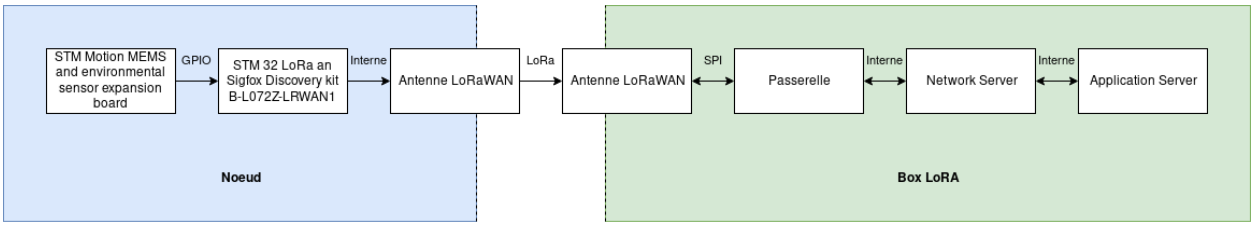
CHAPITRE 4

Schéma technique des prototypes :

Ce premier schéma ci-dessous nous montre la chaine technique des composants du premier réseau simple pour la création du prototype.



Ce deuxième schéma ci-dessous nous montre la version finale du réseau tel qu'il sera construit.



CHAPITRE 5

Analyse des risques

Pour le cas d'usage, nous avons défini que les aspects de non-répudiation et de confidentialité ne sont pas les plus critiques, car nous voulons transmettre uniquement la température et l'humidité. Nous traiterons donc les paramètres d'intégrité et d'authenticité. Il faut éviter qu'une personne vienne altérer l'information envoyée. Nous devons être sûrs que le *noeud* qui envoit l'information est bien le *noeud* que nous avons créé et pas celui d'un éventuel attaquant (par exemple : *man in the middle*).

Tableau 1 – Analyse des risques

Menaces envisageables	Risques à considérer	Contres mesures
Dump mémoire STM32 et Raspberry	✓	Hasher la clef / Composant de sécurité pour le firmware et clef
Canaux cachés (DPA, SPA)	x	
Autre Noeud usurpant l'identité de notre Noeud (altération des données)	✓	Signature et certificat
Mise à jour venant d'une entité autre que le serveur de mise à jour officiel	✓	Signature des MAJ / certificat
Interception des mises à jour	✓	VPN
Execution d'un OS malicieux sur la box LoRA	✓	Secure boot
Modification du programme du noeud	✓	Condamnation des GPIO de debogage
DoS attaque par envoi massif de données sur la Box LoRa	✓	Limiter la réception d'un nombre de trames par X temps

CHAPITRE 6

Contraintes :

— Protocole de communication LoRaWAN entre la passerelle et les noeuds
— Utilisation du matériel fourni par l'encadrant

CHAPITRE 7

Matériel :

Voici le matériel dont nous disposons :

- Lorawan discovery kit : https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html
- MEMS environmental shield : https://www.st.com/en/ecosystems/x-nucleo-iks01a1.html
- LoRaWAN concentrator : https://shop.imst.de/wireless-modules/lora-products/8/ic880a-spi-lorawan-concentrator-868-mhz?number=404802
- Carte Fipy : https://docs.pycom.io/datasheets/development/fipy/
- Carte pysense : https://pycom.io/product/pysense/
- Antenne : https://www.gotronic.fr/art-kit-antenne-pour-lora-et-sigfox-25376.htm
- Raspberry pi 3b : https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

CHAPITRE 8

Répartition des tâches

Pour faire ce travail, nous sommes deux personnels travaillant à plein temps, Arthur et François. Nous devons donc répartir équitablement les tâches.

Dans un premier temps, nous allons tous les deux travailler au déploiement du réseau LoRaWAN.

François va ensuite se charger de la sécurisation de la passerelle, ainsi que des connections avec le *network server* et l'*application server*.

Arthur se chargera de travailler sur la sécurisation du noeud, et le transfert des informations vers la passerelle.

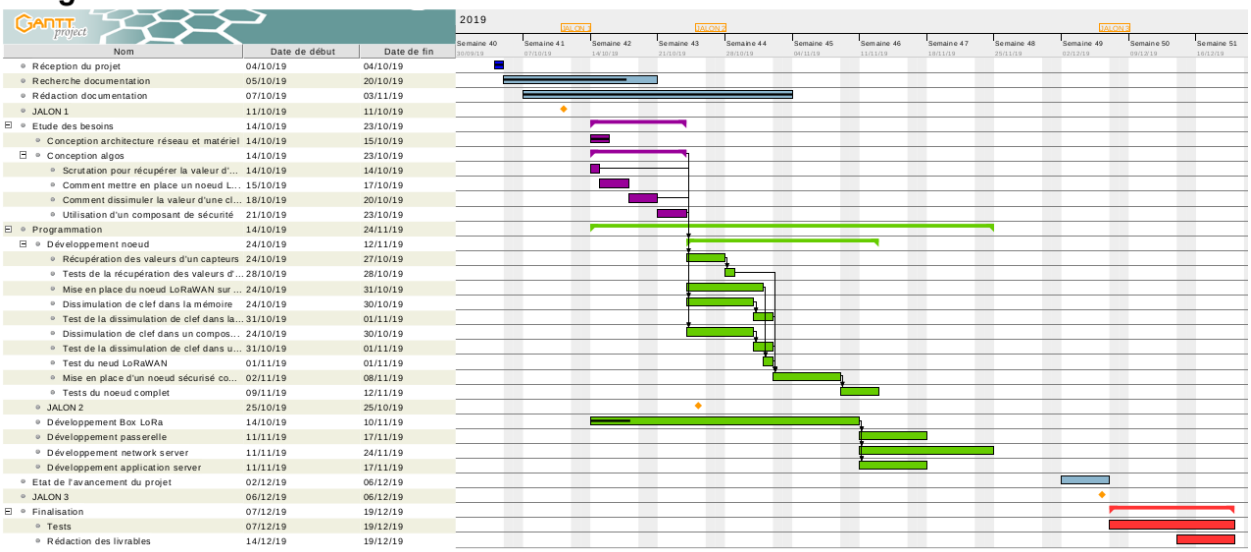
Tous les deux s'occuperont de rédiger constamment une documentation fournie ainsi que tous les documents livrables attendus.

8.1 Méthodologie

Pour gérer le projet nous utilisons un outil de *versionning* appelé Github, où on y met tout le code du projet, les sources ainsi que la documentation. Pour nous organiser tout au long de la période du projet nous avons créé un diagramme de GANTT. Nous le garderons à jour pendant toute la durée du projet. Pour avoir une gestion de projet plus précise (tâches à effectuer chaques semaines), nous utilisons l'onglet *Project* de notre *repository* Github. Dans cet onglet nous indiquons pour chaque semaine les différentes tâches à faire. Les tâches ont 3 états **À faire**, **En cours** et **Fini** nous déplaçons et nous ajoutons des tâches au cours de la semaine.

Le diagramme de Gantt ci-dessous récapitule notre organisation tout au long du projet.

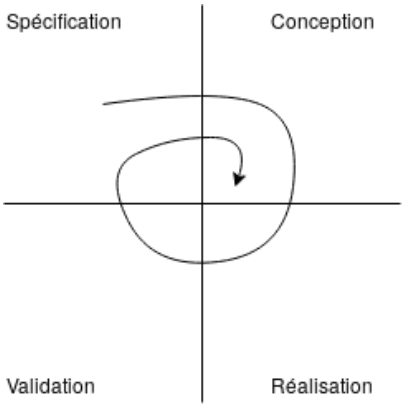
Diagramme de Gantt



8.2 Organisation

Nous avons choisi une approche en spirale pour notre organisation. En effet, sur les conseils de notre encadrant, ce modèle nous permet de tester les différentes couches de sécurisation une à une et de revenir aux étapes précédentes si besoin pour modifier et compléter le dispositif.

Le schéma ci-dessous montre simplement le fonctionnement d'une organisation en spirale.



CHAPITRE 9

Mise en place d”un réseau LoRaWAN simple

Nous allons voir comment mettre en place un réseau LoRaWAN simplement entre une *Raspberry Pi* et une carte *Fipy*. Le noeud devra envoyer la valeur de température jusqu”au serveur d”application.

9.1 Matériel :

- Carte *Fipy*
  - Carte *Pysense*
- Raspberry Pi 3b / 3b+
  - Carte IMST iC880A

9.2 Mise en place du noeud / carte *Fipy*

- Dans un premier temps, installer dans *Visual Studio Code* ou *Atom* et le plugin *Pymakr*
- Ensuite il va falloir mettre à jour le Firmware de la carte d”extension *pysense*, vous pouvez trouver la procédure ici
- Après la mise à jour, débranchez la carte *Pysense* de l”USB
- Mettre la carte *Fipy* sur la carte *Pysense*, **il faut que le bouton reset de la carte *Fipy* soit du coté du port USB de la carte *Pysense***
- Mise à jour de la carte *Fipy* :
  - Avant de commencer, il est recommandé d”installer la mise à jour de la carte *Fipy*. Vous pouvez trouver les informations d”installation pour Windows / Mac OS / Linux à cette adresse. Nous utiliserons une distribution Linux.
  - Installez les paquets *dialog* et *python-pyserial*
  - Télechargez le logiciel de mise à jour ici
  - Télechargez la dernière version du firmware de la carte *Fipy* à cette adresse
  - Après avoir extrait le logiciel de mise à jour, allez dans *pycom_firmware_update_1.16.1-amd64/pyupgrade*
  - Branchez à l”ordinateur la carte *Pysense* avec la carte *Fipy* installée dessus.
  - Executez la commande pour connaitre le port sur lequel est branché la carte :

```
./pycom-fwtool-cli list
```

— Pour écrire la mise-à-jour dans la carte *Fipy*, entrez la commande suivante :

```
sudo ./pycom-fwtool-cli -p /dev/ttyACM1 flash -t ../../FiPy-1.20.0.rc13.tar.gz
```

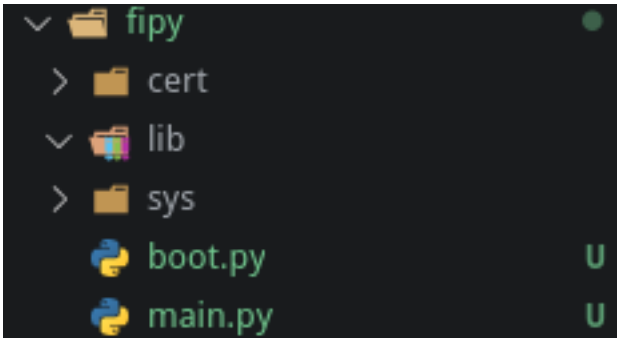
Dans notre cas, la version du firmware est *1.20.0* et le port */dev/ttyACM1*

9.3 Programmation du noeud

Ouvrez *Visual Studio Code* ou *Atom*. Créez un dossier pour le projet, nous l'appellerons *reseau_simple*.
— Créez un fichier de configuration pour le noeud. Cliquez sur *All commands* en bas de l'écran, puis dans le menu déroulant qui s'affiche, séléctionnez *Project Settings*
— Créez un sous-dossier pour y écrire le programme du noeud. Nous l'avons appelé *Fipy*
— Dans le fichier JSON créé précedement, ajoutez le nom de ce dossier :

```
"sync_folder": "fipy",
```

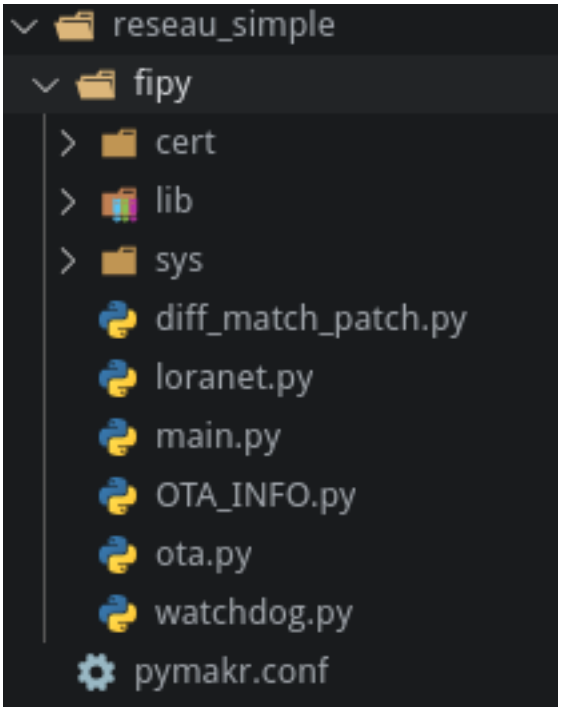
— Créez les dossiers et fichiers suivants dans ce dossier :
  — *boot.py* permet d'exécuter du code uniquement au démarrage de la carte
  — *main.py* permet d'exécuter du code pendant que la carte est allumée
  — *cert* contient les certificats
  — *lib* contient des bibliothèques



9.4 Programmation du noeud à partir d'"exemples

Dans notre cas nous allons baser notre programme sur un exemple que vous pouvez trouver sur github à cette adresse.

Dans un premier temps téléchargez le repository. Ensuite copiez le contenu *pycom-libraries/examples/OTA-lorawan/firmware/1.17.1/flash* dans le dossier **Fipy** créé précédement.



Dans le fichier *main.py*, recopiez le code suivant.

```
#!/usr/bin/env python
#
# Copyright (c) 2019, Pycom Limited.
#
# This software is licensed under the GNU GPL version 3 or any
# later version, with permitted additional terms. For more information
# see the Pycom Licence v1.0 document supplied with this file, or
# available at https://www.pycom.io/opensource/licensing
#

from loranet import LoraNet
from ota import LoraOTA
from network import LoRa
import machine
import utime

def main():
    print('Booting with firmware version 1.17.1')

    LORA_FREQUENCY = 868100000
    LORA_NODE_DR = 5
    LORA_REGION = LoRa.EU868
    LORA_DEVICE_CLASS = LoRa.CLASS_A
    LORA_ACTIVATION = LoRa.OTAA
    LORA_CRED = ('240ac4fffe0bf998', '948c87eff87f04508f64661220f71e3f',
↪'5e6795a5c9abba017d05a2ffef6ba858')

    lora = LoraNet(LORA_FREQUENCY, LORA_NODE_DR, LORA_REGION, LORA_DEVICE_CLASS, LORA_
↪ACTIVATION, LORA_CRED)
    lora.connect()
```

(suite sur la page suivante)

(suite de la page précédente)

```
    ota = LoraOTA(lora)

    while True:
        rx = lora.receive(256)
        lora.send(bytes("Hello World", "utf-8"))
        print("In while")
        if rx:
            print('Received user message: {}'.format(rx))

        utime.sleep(60)

main()

#try:
#    main()
#except Exception as e:
#    print('Firmware exception: Reverting to old firmware')
#    LoraOTA.revert()
```

9.5 Mise en place de la partie passerelle / network server / application server

Pour toute cette partie nous allons utiliser une carte *Raspberry Pi 3b+* avec une carte d"extension *IMST iC880A*

— Dans un premier temps, téléchargez l"image de *lora-getway-os-full* à l"adresse suivante.
— Une fois téléchargée, il faut extraire l"archive.
— Après cela vous devez écrire l"image extraite sur la carte SD de la *Raspberry*
    — Insérez la carte SD dans votre ordinateur
    — Repérez son point de montage à l"aide de la commande : `lsblk`
    — Puis, écrivez l"image sur la carte SD avec la commande suivante (en veillant à bien remplacer le chemin de l"image et le point de montage de la carte SD) :

```
sudo dd bs=4M if=lora-gateway-os-full-raspberrypi3--20190810092349.sdimg of=/dev/
↪mmcblk0 conv=fsync
```

    — Mettez la carte SD dans la *Raspberry* et testez si celle-ci *boot*.
    — Attention ! Le clavier est en QWERTY.
    — Connectez-vous avec les identifiants suivants : Login = admin / Password = admin

9.5.1 Configuartion du WIFI

```
   sudo gateway-config
```

Sélétionnez *Configure WIFI* puis *ok* et *ok*

```
enable wifi
scan wifi
services #Pour voir les réseaux disponibles
agent on
#Choisser un des reseaux dans la liste de service
# Exemple :
# MyNetwork               wifi_dc85de828967_68756773616d_managed_psk
```

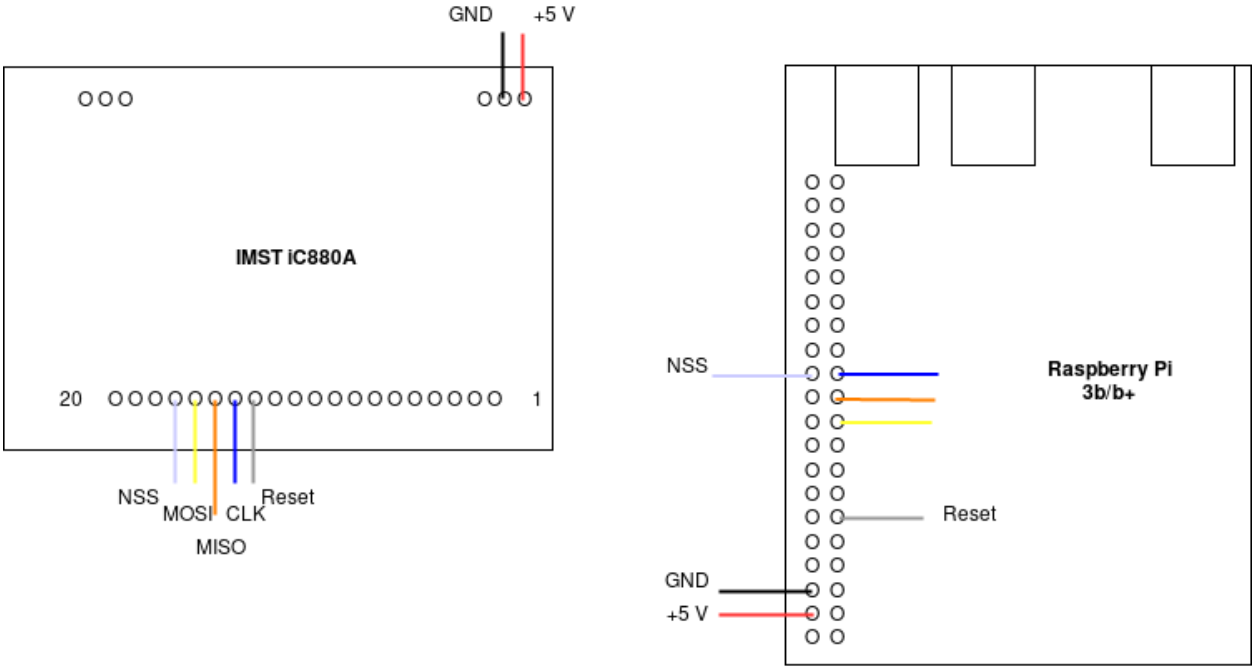
(suite sur la page suivante)

(suite de la page précédente)

```
connect wifi_dc85de828967_68756773616d_managed_psk
#Entrer le mot de passe
quit
```

9.5.2 Configuration de la passerelle

Faites tout le câblage nécéssaire avant de brancher la *Raspberry*.




```
sudo gateway-config
# 2 setup LoRa concentrator shield
# 1 IMST - ic880A
# Entrer 17 si vous avez suivi notre cablage
# OK
# Ok
# Ok
# Ok
# Ok
```

Maintenant à chaque démarrage, l'"OS va se connecter automatiquement à ce réseau wifi.

9.5.3 Paramétrage du serveur de réseau pour notre appareil

Connectez vous à l'"inteface web. Pour y accéder ouvrez votre navigateur et entrez l'"adresse IP de la *Raspberry* suivi de : *8080*. Dans notre cas : **http ://192.168.43.134 :8080** Les identifiants de connection sont les mêmes que pour vous identifer en ssh.

Création d"un *network server*

Allez dans l"onglet *network-servers* et cliquez sur *add*. Vous pouvez mettre comme nom ce que vous voulez, nous l"avons appelé **Fipy_Serv**. Pour *Network-server server* entrez : **localhost :8000**. Cliquer ensuite sur *ADD NETWORK-SERVER*

Création d"un *Gateway-profile* :

— Name : **Fipy_GW_profile**
— Enable channels : **0, 1, 2**
— Network Server : **Fipy_Serv**

Creation d"une *Gateway* :

— Gateway Name : **Fipy_GW**
— Gateway description : **OTAA Fipy Gateway**
— Gateway ID : **b2 1a d4 c0 7d c6 be f6**
— Network-server : **Fipy_Serv**
— Gateway-profile : **Fipy_GW_profile**
— Gateway discovery enabled : **Autoriser**

Création d"un *service profile* :

— Service-profile name **M1-CSSE**
— Network-Server **Fipy_Serv**
— Add gateway metadata **Autoiser**

Création d"un *Device-profile* :

— Device-profile name : **Fipy_Hello_World**
— Network-Server : **Fipy_Serv**

Device-profiles/Create

— Device-profile name : **Fipy_dp**
— Network-server : **Fipy_Serv**
— LoRaWAN MAC Version : **1.0.2**
— LoRaWAN MAC version supported by the device : **B**

Device-profiles/ota_dp

— Device support OTAA : **Autoriser**

Application

— Application name : Hello_World
— Application description : Hello world App
— Service-profile : M1 CSSE
— Payload codec : **None**

Application / Hello_world / Create

— Device name : **Fipy**
— Device déscription : **Fipy**
— Device EUI : **240ac4fffe0bf998**
— Device profile : **Fipy_dp**

Application / Hello_world / Devices / Fipy

— Application key : **5e6795a5c9abba017d05a2ffef6ba858**

9.6 Problèmes rencontrés

9.6.1 (Pymakr) « There was an error with your serialport module »

Ce problème apparait au démarrage de visual studio code après l'"installation de *Pymakr*. Vous pouvez trouver des informations pour résoudre le problème ici.

Résolution du problème :

1. *Dans le cadre de ce projet nous utilisons un fork de **Visual Studio Code** appelé **code** les noms de dossier sont suceptibles de changer en fonction du logiciel que vous utilisez.*
2. *Nous utilisons pour ce projet la distribution Linux **Manjaro** qui est basé sur **Arch Linux** le gestionnaire de paquet sera peut-être différent du votre*

```
$ sudo pacman -Sy npm
$ npm install -g prebuild-install
$ cd ~/.vscode-oss/extensions/pycom.pymakr-1.1.3/
$ cd node_modules/@serialport/bindings
$ prebuild-install --runtime electron --target 4.2.5 --tag-prefix @serialport/
↪bindings@ --verbose --force
```

Il faut ensuite relancer Visual Studio.

9.7 Sources :

— Mise en place Fipy : https ://docs.pycom.io/ ; https ://docs.pycom.io/gettingstarted/connection/fipy/ ;
— Mise en place Pymakr : https ://docs.pycom.io/pymakr/installation/vscode/ ; https ://docs.pycom.io/pymakr/toolsfeatures/
— Programmation Noeud : https ://docs.pycom.io/tutorials/lora/lorawan-otaa/ ;
— LoRa Server pour Raspberry : https ://www.loraserver.io/lora-gateway-os/install/raspberrypi/
— Partie Passerelle Box LoRa : https ://www.loraserver.io/guides/first-gateway-device/ ; https ://docs.pycom.io/tutorials/all/ota-lorawan/

CHAPITRE 10

Indices and tables

— genindex
— modindex
— search