IUT de LANNION

Département Réseaux & Télécommunications

THE PLACE TO BEE

Solution pour des ruches connectées

Etudiants

Hugo BELNARD

Titouan HAMON

Yann KERDONCUFF

Titouan HEYRENDT

Enseignants

Baptiste VRIGNEAU

Matthieu GAUTIER

Techniciens

Cyrille RENAULT EVEN

Yvan GUILLOIT

Année 2020-2021

Introduction

Nous sommes une équipe de quatre étudiants en seconde année de DUT Réseaux & Télécommunication à l'IUT de Lannion (22) de l'Université de Rennes1. La sauvegarde des abeilles et de la biodiversité est un sujet important et complexe de notre génération. À ce jour nous avons déjà des exemples de pays européens qui ont dû importer des abeilles à cause d'un taux de mortalité trop élevé. En effet, à cause de la surexploitation des sols et des produits phytosanitaires employés dans les cultures, la mortalité des essaims d'abeilles suit une inquiétante courbe exponentielle. C'est pourquoi il faut agir au plus vite pour sauvegarder les nôtres, et ainsi conserver notre biodiversité. Dans le cadre d'un projet tuteuré nous nous sommes intéressés à la création d'une ruche connectée afin d'aider les apiculteurs. A l'heure du tout internet, nous avons aussi conscience de l'impact de la quantité d'information qui transitent dans le cloud sur deux aspects. Le premier est la consommation d'énergie car à l'instar du circuit court en alimentation, éviter de parcourir des kilomètres au travers d'équipement et de centre de serveurs permet d'économiser de l'énergie. Le second est la gestion des données afin de les sécuriser et de les pérenniser. Lorsque la quantité reste limitée comme c'est le cas pour une ruche, il est facile de réaliser des sauvegardes personnelles pouvant échapper à un incendie, péripétie qui peut arriver aussi aux professionnels.

Nous participons au concours avec comme sujet l'acquisition de données afin d'aider la gestion d'une ruche. L'idée est d'assister les apiculteurs professionnels pour un gain de temps et une meilleure réactivité. Ils pourront suivre l'état de leurs ruches sans avoir à les ouvrir, d'obtenir un suivi constant et régulier laissant espérer une détection d'une attaque d'un parasite ou le départ d'un essaim. Un tel système peut aussi intéresser les particuliers qui n'ont pas nécessairement la possibilité de s'occuper à plein temps d'une ruche.

Les livrables du concours s'organisent selon la structure :

- Cette première partie présente les principes de notre projet dans les grandes lignes avec un état de l'art des solutions existantes,
- Des annexes et des fiches explicatives d'installation et de configuration des différentes parties permettront à des apiculteurs de mettre en place notre solution. Cependant, il n'est pas facile de mettre tous les fichiers de code PyCom. Ils ne sont pas tous présents et nous les fournirons avec plaisir sur demande.

Nous avons essayé de simplifier au maximum les notions techniques afin de permettre à un apiculteur de réaliser son installation. Elle demande cependant un minimum de connaissance en informatique, ou du moins ne pas y être allergique. Les notions abordées sont d'un spectre très large car il faudra installer un système d'exploitation puis les logiciels nécessaires, mettre en place la configuration réseau, programmer des cartes microcontrôleur et relier entre eux tous les services. Nous indiquerons également des possibilités d'utiliser des services disponibles sur internet simplifiant la mise en place mais requérant une connexion. Attention également à la constante évolution de ce domaine avec notamment les nouvelles versions des différents éléments logiciels ou matériels avec ses lots de bonnes et mauvaises surprises : par exemple le matériel Pycom et son plugin de programmation Pymakr ou encore le serveur LoRaWan de ChirpStack dont la version 3 vient tout juste de sortir.

Présentation du projet

Le contexte

Dans ce projet, nous nous sommes basés sur le principe des objets connectés avec l'internet des objet (Internet Of Things ou IoT). Le principe colle totalement avec le but du projet qui est de récolter les données sur un serveur puis de l'afficher sur un site internet. Ces données sont le poids, l'humidité, la température externe et interne. Ce ne sont pas des données très volumineuses (quelques octets) avec des fréquences d'émission faibles (1 à 2 fois par jour). C'est pourquoi nous choisissons de suivre les normes de l'IoT, car elles permettent d'envoyer peu de données sur de longues distances ce qui est très intéressant pour un apiculteur qui a des ruchers éloignés ou un regroupement de plusieurs apiculteurs par zones géographiques. De plus, les normes de l'IoT sont basées sur une faible consommation électrique afin d'augmenter l'autonomie des nœuds et de limiter les interventions sur la ruche. En effet, l'énergie liée à la transmission sans fil est la plus importante au regard du reste de consommation, c'est-à-dire le fonctionnement du microcontrôleur et des capteurs.

Etat de l'art des ruches connectées

Il existe plusieurs solutions actuellement répondant aux besoins de collectes d'informations pour le suivi et également contre le vol. Vous trouverez dans ce tableau les caractéristiques des ruches connectées que l'on peut trouver sur le marché.

Nom	BeezBee	ICKO Ruche connectée Bee2beep	ICKO Module de surveillance pour ruche	Label Abeille ruche connectée
Capteurs	Poids, température, humidité	Poids	Poids et Humidité	Position, orientation, luminosité, température, humidité, poids, localisation, consommation électrique, Pression atmosphérique
Prix	300€	209€	134€	768€
Abonnement	??	60€/mois		6€/mois
Protocole de transmission	LoRa et SigFox	SigFox	LoRa WAN ou SigFox suivant le pays	Double mode d'émission : 3G/4G LoRa
Support d'affichage des informations de la ruche	Application Web	Application web	Application Android	Application Web, Application mobile compatible Android et IOS, SMS
Information supplémentaire délivrée par la ruche	géolocalisation	Météo	-	Système antivol Météo
Autonomie	2 ans	9 mois	2 ans	1 an (2500 cycles de charge)

- https://www.beezbee.fr/
- Label Abeille ruche connectée
- https://www.icko-apiculture.com/
- https://www.beeguard.fr/ (le site internet est assez pauvre en données techniques)

Les choix de la solution

Comme souvent, la solution proposée n'est pas unique et nous résumons ici les choix que nous avons été amenés à trancher. Il est aussi nécessaire d'avoir un ordinateur avec un port USB, un lecteur de carte SD et une connexion à internet.

Norme de transmission sans fil

Parmi les solutions disponibles, nous avons choisi la technologie LoRa pour les raisons suivantes :

- Orientée IoT, la norme LoRa permet d'envoyer des messages sur de longues distances. En milieu rural, la connexion peut se faire dans un rayon de 15 km s'il n'y a pas trop de perturbation (arbres, collines...). En contrepartie, le débit assuré par LoRa est plus faible avec quelques kbit/s, bien plus lent que le Wifi. Cela n'est pas bloquant car les paquets de données sont petits, typiquement 20 octets et sachant qu'un résultat de capteur peut se coder entre 1 et 3 octets.
- Des opérateurs sont disponibles, comme le réseau de téléphonie mobile, mais il est possible de mettre en place son propre réseau. C'est ce que nous proposons dans cette solution afin de gérer de bout en bout son réseau et ses données.
- Une autre norme englobant le LoRa, dénommée LoRaWan, permet de gérer les accès au réseau (identification des nœuds) et la récupération des messages ensuite.
- Le matériel se démocratise de plus en plus et il est maintenant possible de trouver plusieurs solutions proposant la norme LoRa.

Le matériel embarqué sur la ruche

Notre choix s'est porté sur le fabricant PyCom qui propose tout un éventail de solutions matérielles associant un microcontrôleur ultra faible consommation ESP32 du fondeur Expressif Systems et des modems de transmission sans fil. Par exemple, les deux modules que nous avons utilisés sont la FiPy qui est capable de transmettre en Wifi, Bluetooth, SigFox, LoRa, NB-IoT ou Cat-M et la LoPy qui est limitée aux Wifi, Bluetooth, SigFox et LoRa. Si ces cartes sont vendues à l'unité, elles ont besoin d'un environnement matériel pour fonctionner, par exemple pour gérer l'alimentation et l'interface pour la programmation depuis l'ordinateur. Elles vont être associées à des cartes d'accueil développées par PyCom. Quelques exemples :

- Elles proposent toutes de programmer en USB, brancher/charger une batterie LiPo, ajouter une carte microSD et la connectique pour les entrées/sorties du microcontrôleur.
- La carte Expansion Board permet de connecter des capteurs sur les entrées/sorties.
- La carte PySense embarque différents capteurs comme la température, l'humidité ou la pression.
- La carte PyTrack embarque un GPS.

Il en existe d'autres et nous vous encourageons à visiter le site www.pycom.io afin de trouver toutes les documentations techniques, des exemples et de se tenir informé des nouveautés. Il est donc possible de combiner les solutions de PyCom mais aussi des capteurs externes classiques dans les montages de microcontrôleur. Nous avons utilisé une FiPy avec une carte ExpansionBoard complétée par :

- un DHT11 de GoTronic pour la température externe et l'humidité,
- une sonde DS18B20 pour la température interne

- un HX711 et 4 jauges de contraintes 50kg (cette partie demande à être améliorée avec un concentrateur de capteurs de Sparkfun).

Il est également possible d'utiliser la PySense. Pour récupérer les données de la ruche, le module récupère les informations des capteurs, les traite, se connecte au réseau LoRa, les transmets puis se met en veille profonde. Il est important de bien utiliser le mode Veille afin d'économiser la batterie entre deux messages. Il est cependant possible de réveiller le nœud pour une prise d'information ou lors d'une action de l'environnement. Un message asynchrone peut être transmis comme une alarme lorsque le poids varie brutalement (départ d'un essaim) ou lors de mouvements de la ruche (manipulations d'un tiers non autorisé ou vol).

Le point d'accès et le serveur LoRaWAN

Le nœud présenté sera répliqué pour chaque ruche et il faut maintenant un point d'accès LoRa accessible. La figure 1 résume les architectures réseaux évoquées ici. Il est possible d'utiliser le réseau LoRa de The Things Network ou TTN qui est gratuit et ne demande que la création d'un compte sur https://www.thethingsnetwork.org/. Il faut cependant qu'un point d'accès couvre la zone d'intérêt et il restera encore à stoker les données. La solution TTN1 requiert un accès LoRa TTN et internet puis le traitement des données via le protocole MQTT. Cette solution n'est pas présentée ici.

Afin de récupérer les données, nous utilisons un autre produit de la marque Pycom : la carte d'accueil PyGate. Combinée avec une LoPy, elle permet de gérer 8 connections LoRa simultanées. En une journée près de 30 000 Ruches pourraient se connecter à la Pygate. Une fois la connexion effectuée avec une ruche, la PyGate récupère les données et les envoie au serveur LoRaWAN. Une option a été de prendre le module Power over Ethernet afin d'alimenter la carte et de récupérer les données avec un seul câble ethernet, ce qui peut être pratique pour déporter le point d'accès. Cela demande aussi d'avoir soit un switch avec un port PoE, soit un injecteur PoE. Concernant le serveur LoRaWAN, il est possible d'utiliser TTN (Solution TTN2) avec des accès internet. Cette solution n'est pas présentée ici.

Dans une optique de fournir une solution à faible coût, nous utilisons un Raspberry Pi 4 (ou un RPi 3B+) comme serveur LoRaWAN et des autres services. Les Raspberry sont analogues à des ordinateurs minimalistes mais leurs capacités permettent tout à fait de répondre à notre demande. Le point faible est la carte SD utilisée pour le système d'exploitation et les technologies actuelles sont limitées dans le nombre d'écriture. Il serait envisageable de l'équiper d'un disque dur externe ou de mettre en place une sauvegarde des données délocalisée. Il est nécessaire de relier la PyGate et le RPi sur le même réseau grâce à un switch. Il faut aussi que le RPi ait un accès à internet pour l'installation des logiciels. Le serveur LoRaWan est celui proposé gratuitement par ChirpStack.

Les services

Une fois les informations collectées, il faut les stocker et les exploiter.

Le stockage est assuré par une base de données (BDD). Nous avons choisi InfluxDB qui est une solution gratuite et spécialisée dans les séries temporelles, c'est-à-dire typiquement les relevés de capteurs. Le serveur LoRaWAN formate correctement les données et les injecte dans la base de données. Celles-ci sont alors étiquetées d'un Timestamp (temps) et de Tags (mots-clés pour le filtrage).

L'affichage utilise l'outil Grafana qui permet d'interroger la BDD puis de créer un graphique visible sur un tableau de bord ou intégrable dans une page web. Il permet également de générer des alertes simples (dépassement de seuil par exemple).

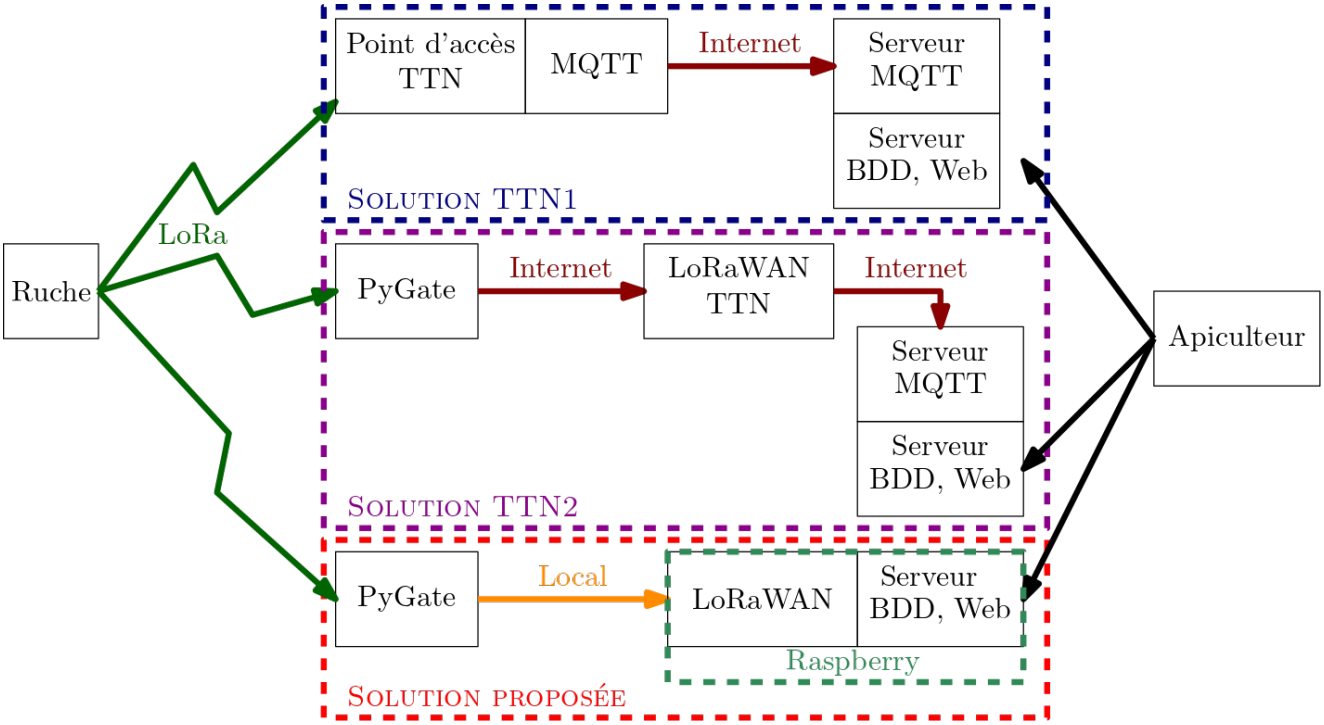


Figure 1 : synoptique des réseaux pouvant être autonome (notre solution) ou utilisant TTN.

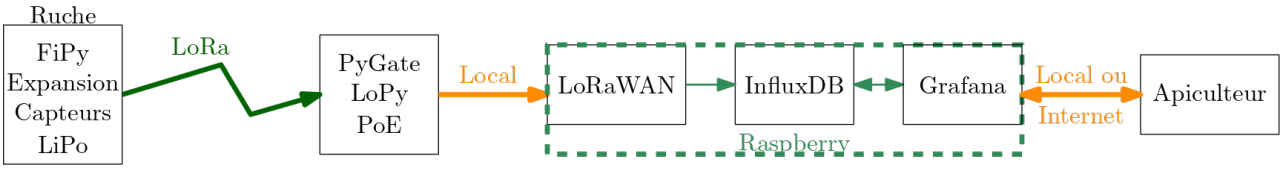
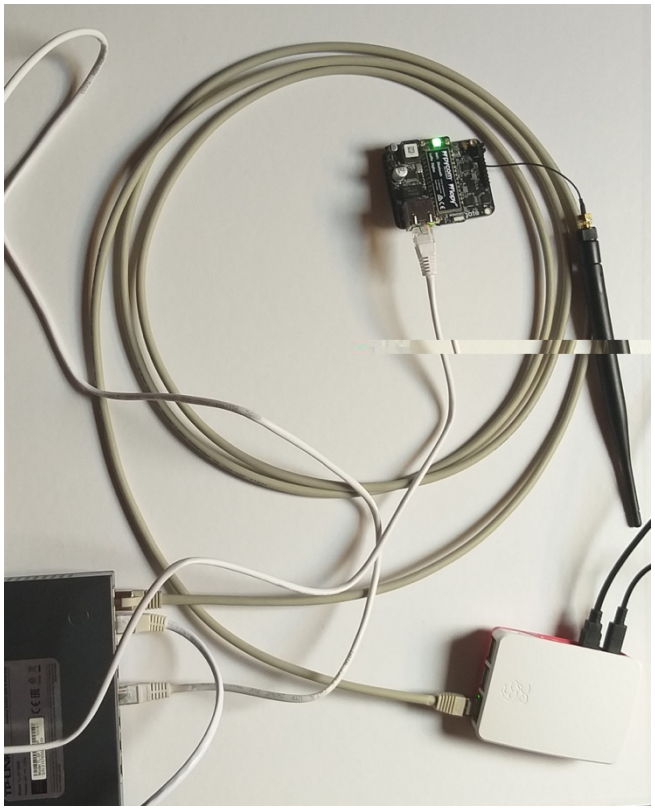


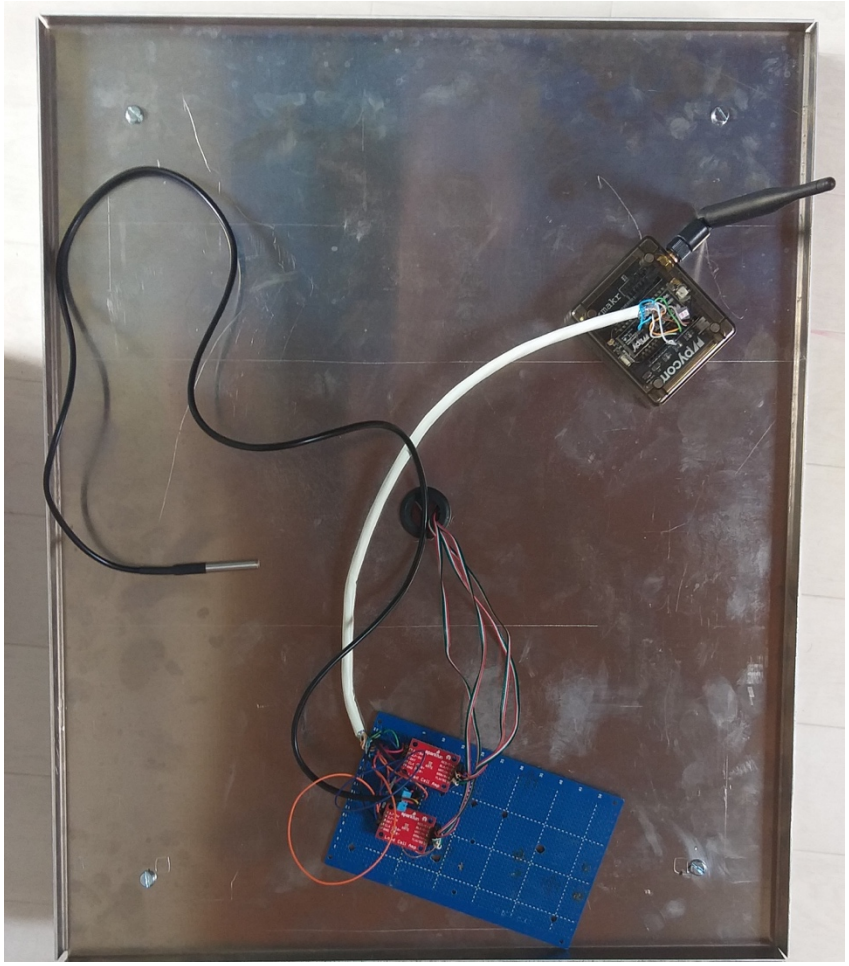
Figure 2 : synoptique détaillé de notre solution

Preuve de concept

Prototype



Point d’accès : PyGate avec son antenne en haut (la LED en vert signifie qu’elle est bien connectée), le RPi assurant les services et le stockage, le switch PoE et le câblage. Il manque encore le boitier pour la PyGate mais il n’était pas disponible.

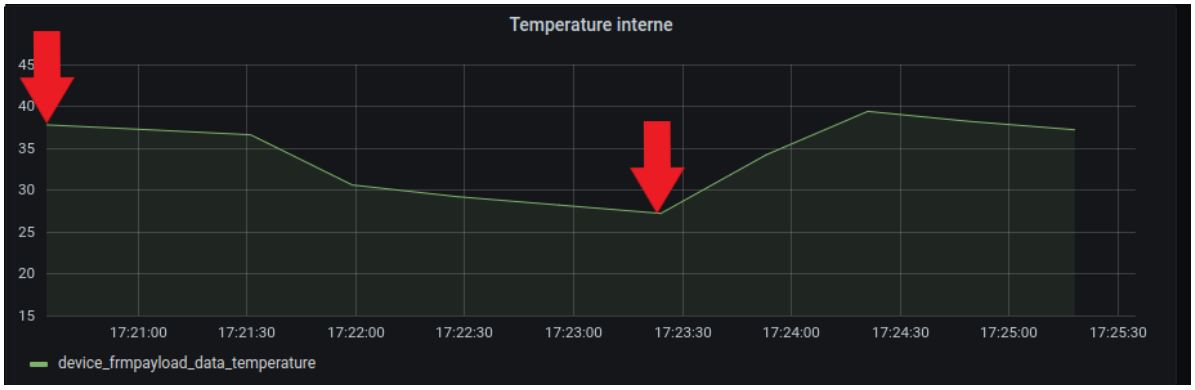


Nœud de la ruche : FiPy et ExpansionBoard en haut avec le prototype d'interfaçage avec les capteurs (l'étape suivante est de l'intégrer dans un boitier).

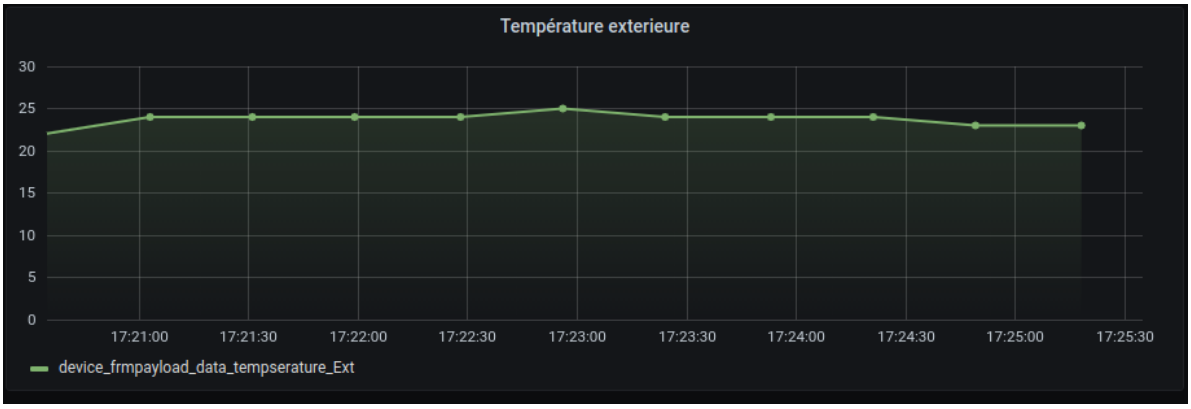
Premier test en conditions réelles

Test avec un scénario de stimuli sur les capteurs

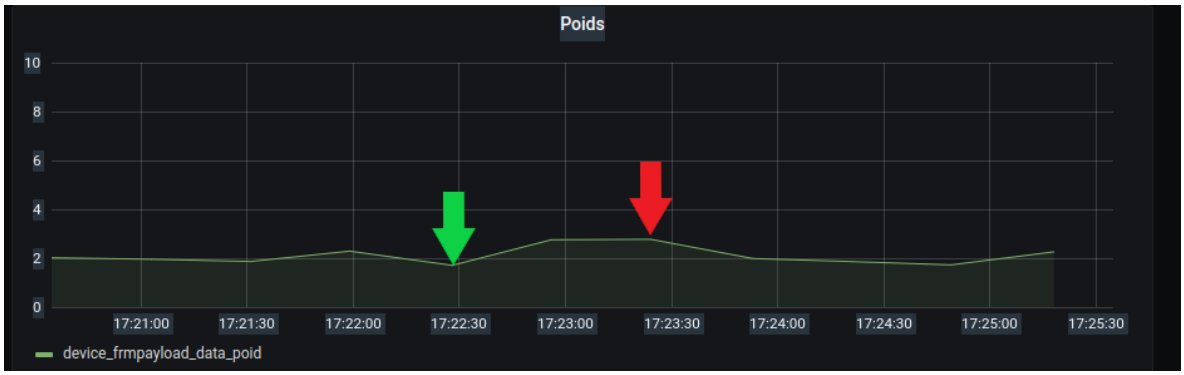
Après avoir tout mis en place nous avons fait des tests. Pour ces tests, nous avons envoyé des messages toutes les 30 secondes en influant sur les différents capteurs de la ruche. La fréquence des messages est choisie élevée pour visualiser rapidement les résultats. Les courbes de Grafana sont présentées ci-dessous.



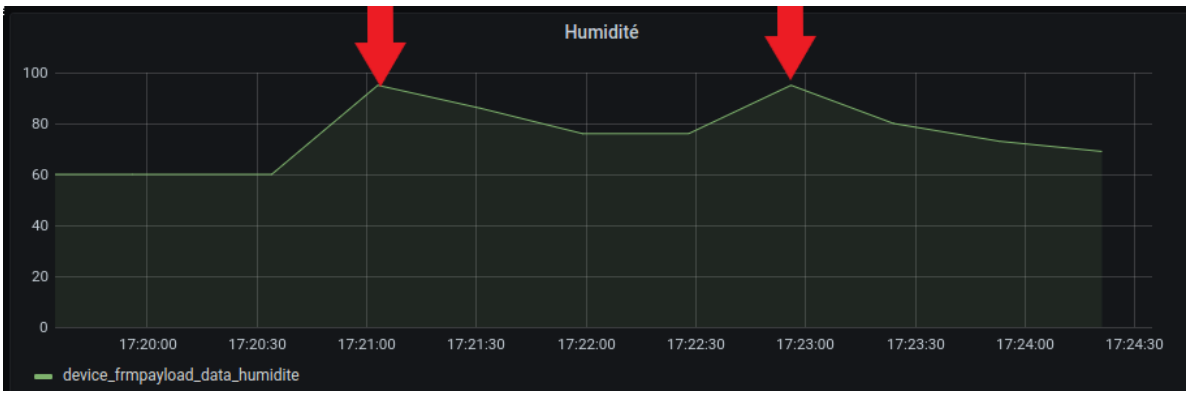
Comme vous pouvez le voir sur cette courbe, nous avons mis la sonde de température dans de l'eau chaude. Au début, la courbe descend car l'eau se refroidit, juste avant 17h23 et 30 sec. Nous avons rajouté de l'eau chaude et la température a augmenté à nouveau.



Le capteur de température externe n'est pas modifié sur la fenêtre d'observation et représente bien la température ambiante.



Pour le poids, nous avons ajouté un lest à 17h22 et 30 sec puis enlevé une minute après. Si les valeurs sont bien récupérées avec une chaîne de transmission validée, ce capteur demande encore du travail sur la partie matérielle et sur son exploitation (notamment sur le bruit de mesure et la conversion en gramme). Il faut réussir à descendre sous les 100g d'imprécision afin de surveiller l'essaimage.



Quant au capteur d'humidité, il nous a suffi de souffler dessus car l'haleine est chargée d'humidité. On voit très distinctement sur la courbe les deux fois où nous avons sollicité le capteur.

Ces tests nous ont permis de valider les parties suivantes de notre ruche connectée :

- La programmation des nœuds PyCom sur la ruche à base de FiPy et la récupération des données des capteurs.
- La configuration de la transmission LoRa avec le formatage des données dans le paquet LoRa, la connexion et l'identification au point d'accès puis l'émission des données.
- La réception des données sur le point d'accès grâce à notre serveur LoRaWAN. Il est possible de superviser le réseau en observant l'état des gateways et le nombre de transmission.
- Le bon fonctionnement du serveur d'application : extraction des données et formatage correcte pour les injecter dans une base de données InfluxDB.
- Le stockage dans la base de données et la visualisation grâce à Grafana.

Travaux à venir

L'état actuel du projet permet de valider une partie importante mais il reste encore des points à améliorer. Notamment, le point faible est la réalisation définitive de la ruche. C'est en effet le domaine de compétences le plus éloigné de la formation Réseaux&Télécommunications. Il faut améliorer les capteurs de poids et protéger tout le système contre les intempéries. Il faut également éprouver la fiabilité sur du long terme. Le projet est pour l'instant dans l'esprit du Do It Yourself et demande quelques connaissances en informatique et électronique. Il faudrait encore franchir une étape pour donner toutes les pièces du puzzle et qu'une personne non spécialiste arrive à mettre en place toute la chaîne de la ruche connectée.

Nous voulons également réaliser des tests de couverture avec le point d'accès LoRa. Pour cela, il faut regarder la législation et positionner notre PyGate sur un point élevé. En positionnant la ruche sur l'ensemble du campus de l'IUT de Lannion et en testant, nous pourrons évaluer les performances de LoRa et l'impact de bâtiments ou de végétations.

Un autre point important est la consommation du nœud sur la ruche. Il sera autonome et alimenté par une batterie et il est primordiale d'estimer la consommation pour la réduire au maximum. Des premières mesures ont été effectuée en laboratoire pour connaitre la consommation d'un module. Par exemple, en fonctionnement normal, la FiPy avec la PySense consomme un courant autour de 1 mA alors que le mode veille le plus économe permet de descendre en dessous de 10 µA soit un rapport 100. Le code Python doit être optimisé en conséquence afin de limiter le temps de réveil avec un traitement et une transmission les plus courts pour un retour en mode veille.

Il faudra alors valider une utilisation correcte et acceptable par un apiculteur en conditions réelles (pluie, vent), avec des abeilles et sur un temps long (au moins une année pour un cycle complet de la vie de la ruche). Ce sera aussi l'occasion de tester des alertes.

Conclusion

Pour conclure, toutes les parties que nous traitons dans notre DUT sont fonctionnelles, à savoir la programmation, l'envoi des données et les mises en place des services. Pour le moment, nous avons fait un prototype fonctionnel. Nous allons continuer à travailler sur notre projet afin d'améliorer certaines parties et mettre en place le système afin de faire des tests grandeur nature. En tout dernier point, deux étudiants du groupe continuent leur stage dans ce domaine et voici leurs deux impressions.

Titouan HAMON : Tout ce que j'ai appris lors de ce projet m'a beaucoup intéressé. Mon stage porte sur les cartes Pycom et les télécommunications. Au cours de mon stage, je réutilise ce que j'ai appris sur la programmation des cartes et l'affichage des données. Cela me permet d'avancer efficacement et d'apprendre des choses différentes. J'ai découvert le principe de l'internet des objets connectés et cela m'a beaucoup plu. J'ai pour projet de continuer à travailler dans cette voie.

Hugo BELNARD : Personnellement, ce projet m'a permis d'apprendre beaucoup sur les télécommunications ainsi que la création de site web. Même si ce n'est pas apparent dans la partie finale car tout est fait en local. J'y ai notamment beaucoup appris sur le JavaScript ainsi que l'utilisations de différentes API dans un code. Je vois ce projet comme une très bonne expérience qui m'a confirmé le secteur dans lequel je voulais me diriger.

Bill Of Materials BOM

Description

Point d'accès

- 1 WiPy, LoPy ou GPy
- 1 PyGate avec module Ethernet PoE
- 1 kit antenne LoRa/SigFox
- 1 boitier PyGate Case

Noeud

- 1 FiPy ou LoPy
- 1 PySense ou Expansionboard 3.1
- 1 Universal IP67 Case
- 1 kit antenne LoRa/SigFox
- 1 kit antenne IP67
- 1 batterie LiPo
- 1 HX711
- 4 jauges de contraintes
- 1 concentrateur de jauges
- 1 DS18B20
- 1 DHT11 GoTronic (option si PySense utilisée)

Serveur

- 1 kit Raspberry Pi 3B+ ou 4 (alimentation, carte SD et boitier)

Réseau

- 1 Switch POE 4 ports minimum
- 3 câbles réseau

Estimation des coûts pour 1 ruche et 1 point d'accès

Les prix sont arrondis au supérieur pour plus de simplicité.

Nom	Prix	Lien
Lopy4 Multipack (Lopy4 + PySense 2.0 x)	67 €	https://pycom.io/product/lopy4-multipack/
PyGate with Power over ethernet 868MHz	80 €	https://pycom.io/product/pygate/
Lopy4	35 €	https://pycom.io/product/lopy4/
Raspberry 400	110 €	https://www.kubii.fr/raspberry-pi-400/3084-kits-raspberry-pi-400-3272496302914.html
Switch POE	48 €	https://www.ldlc.com/fiche/PB00267565.html
PyGate case	10 €	https://pycom.io/product/pygate-case/
Universal IP67 case	15€	https://pycom.io/product/universal-ip67-case/

Kit antenne IP67	4€	https://pycom.io/product/ip67-antenna-cable/
2 Antennes LoRa	18 €	https://pycom.io/product/lora-868mhz-915mhz-sigfox-antenna-kit/
Batterie LiPo 1000mAh	10 €	https://www.gotronic.fr/art-accu-lipo-3-7-v-1000-mah-pr523450-5813.htm
DS18B20	8	https://www.gotronic.fr/art-capteur-de-temperature-grove-101990019-23842.htm
Concentrateur de jauge	3 €	https://www.gotronic.fr/art-combinateur-capteurs-de-force-bob-13878-31169.htm
HX711	11 €	https://www.gotronic.fr/art-amplificateur-pour-capteur-de-force-sen-13879-25334.htm
Jauge de contraintes 50kg x4	44 €	https://www.gotronic.fr/art-capteur-de-force-50-kg-czl635-50-17601.htm
Boitier HX711	2€	https://www.gotronic.fr/art-boitier-clipsable-g13m-19757.htm
Plaque aluminium	20€	
Total	466 €	
Total pour une ruche seule	173 €	

Présentation du Matériel PyCom

LoPy ou FiPy

La LoPy et la FiPy sont deux modules de la marque PyCom. Il se ressemblent beaucoup dans leurs caractéristiques : 4Mb de RAM, 8Mb de Stockage, fonctionne en Micro-Python, Consomme 25uA et possède le même processeur. La seule différence se trouve dans leurs moyens de télécommunication. La LoPy peut utiliser le Wifi, le Bluetooth, le LoRa et le Sigfox mais la FiPy peut utiliser en plus le LTE CAT M1/NB1 qui est une extension de la 4G. Ce sont donc des cartes très complètse qui ont un très large domaine d'applications. Dans notre projet nous utilisons la technologie LoRa que les deux cartes peuvent utiliser. La LoPy étant moins chère il est préférable de la prendre sauf si vous souhaiter utiliser le LTE par la suite.

La Pygate avec PoE

La PyGate est une carte d'accueil qui permet de gérer 8 connexions LoRa en même temps. Cela permet de créer un réseau d'objets connectés (IoT) sans passer par des solutions payantes ou par des serveurs étrangers (préservation et gestion de nos données personnelles). Grâce à l'extension PoE cela permet d'utiliser une prise électrique de moins et d'alimenter le point d'accès par le câble réseaux (très pratique pour déporter le point d'accès sur un point culminant pour plus de couverture).

L'ExpansionBoard

C'est la carte d'accueil minimaliste mais elle permet de programmer en USB, de brancher/Charger une batterie LiPo, d'utiliser une carte microSD et d'accéder aux entrées/sorties du module.

La PySense

Tout comme la Pygate, la Pysense est une carte d'accueil. Elle possède des capteurs directement intégrés, comme un capteur de température, d'humidité, de lumière et de pression. Elle possède aussi des broches non utilisées qui permettent d'ajouter d'autres capteur ou des actionneurs.

Les boitiers PyCase

Les deux boites permettent de garder les cartes à l'abris de la poussière et de l'eau. Surtout pour la Pysense qui se trouve sur la ruche, la Pygate à une prise réseaux par lequel passe le courant il est donc important qu'il soit protégé.

Les antennes

Attention à bien choisir les antennes car ce ne sont pas les mêmes selon la norme utilisée. Il faut aussi bien veiller à les connecter aux bons endroits. Utiliser une antenne non appropriée pourrait causer des problèmes sur la carte ou une mauvaise transmission des données donc une surconsommation.

Choix d'un serveur local

Le Raspberry Pi ferra guise de serveur pour des différents services mais aussi le stockage de données. Il est préférable d'utiliser un Raspberry Pi car le but est de gérer les données en local. Les solutions cloud sont souvent hébergés dans d'autres pays (question sur la protection et la gestion des données) et demandent aussi un accès à internet. Cependant, l'avantage des solutions cloud est d'avoir moins de configuration à faire.

Matériel Réseau

Le switch PoE va permettre deux choses, tout d'abord sa fonction principale sera de transférer les données de la PyGate au Raspberry et permettra au Raspberry d'accéder à internet. Mais aussi d'alimenter la PyGate car celle-ci doit être en extérieur et cela peut poser problème de devoir sortir une source d'alimentation vers l'extérieur. Grace au switch il faudra seulement le câble réseau qui transférera les données et alimentera la carte en même temps.

Nouvelles cartes Pycom

Pycom propose de nouvelles solutions clé en main, les Pygo 1 et 2. On peut comparer les Pygo aux FiPy et LoPy : il y a une correspondance entre la PyGo1 et la LoPy et entre la PyGo 2 et la FiPy.

Les PyGo sont légèrement plus grandes que les autres mais intègre un accéléromètre et un GPS. Elles sont aussi certifiées IP67 et peuvent présenter une solution intéressante pour une ruche en extérieur. Une batterie LiPo est est également disponible ainsi qu'un écran Piezo électrique tactile. Nous n'avons pas eu l'occasion de travailler avec ce matériel pour le moment. Un point fort de la Pygo est sa RAM car elle en possède 2 fois plus que ces consœurs.

Voici un petit tableau récapitulatif :

Nom	PyGo 2	PyGo 1	FiPy	LoPy
Communication	LTE		LTE	
	LoRa	LoRa	LoRa	LoRa
	Wifi	Wifi	Wifi	Wifi
	BLE	BLE	BLE	BLE
	Sigfox	SIGFOX	SIGFOX	Sigfox
Capteur	GPS	GPS		
	Accéléromètre	Accéléromètre		
Protection	IP67	IP67		
Recharge	Wireless	Wireless	Wire	Wire
Processeur	ESP32	EXP32	ESP32	ESP33
NBR Proc.	Dual Proc	Dual Porc	Dual Proc	Dual Proc
Freq.	32khz	32khz	32khz	32khz
RAM	4MB	64Mbit	4MB	4MB
Stock	8MB	8MB	8MB	8Mb
GPIO	6	6	22	24
Prix	89	69	54	34

Guide d’installation

Afin de programmer les cartes Pycom, il faut procéder à différentes installations. Tout d'abord, il faut choisir entre deux éditeurs Atom et Visual Studio Code qui sont les plus simples pour utiliser Pymakr. Visual Studio Code est moins long à installer. Atom nécessite plus de manipulation mais est plus ergonomique. Vous pouvez utiliser un ordinateur ou le RPi. La procédure présentée ici est pour Windows mais les deux logiciels sont disponibles sur tous les OS.

Table des matières

Installation de Visual Studio Code (VSC) 2

Installation des packages sur Visual Studio Code (VSC) 2

Installation de Node.js 2

Installation de Python 3

Installation de Atom 4

Installation de Pymakr (Atom) 4

Installation de Visual Studio Code (VSC)

Pour utiliser Visual Studio Code, rendez-vous sur le site.

Cliquer sur download et sélectionner le système d'exploitation que vous utilisez.

Une fois téléchargé cliquer sur le fichier pour lancer l'installation.

Lisez et acceptez les conditions d'utilisation (entre chaque étape cliquer sur 'next').

Si vous le souhaitez, vous pouvez modifier l'emplacement de l'installation.

Si vous le désirez, vous pouvez choisir de mettre en place un dossier qui s'ouvrira à chaque démarrage de VSC (votre répertoire de travail sur la ruche par exemple).

Si vous le souhaitez, cocher 'Create a desktop icon' afin de mettre une icône sur votre bureau.

Enfin cliquer sur 'install' et après un temps de chargement, vous pourrez lancer VSC.

Installation des packages sur Visual Studio Code (VSC)

Cliquer sur l'icône un menu s'ouvrira.

Dans la barre de recherche, tapez 'C/C++' puis cliquer sur le bouton 'install' tout en haut de la liste.

Faites de même avec 'Python' et 'Pymakr' (recherche puis installation).

Installation de Node.js

Rendez-vous sur le site de Node.js.

Sélectionner le système d'exploitation sur lequel vous êtes pour le télécharger.

Exécuter le fichier.

Lisez et accepter les conditions d'utilisation. Et choisissez le dossier où il sera sauvegardé.

Cliquer sur 'next' jusqu'à ce que Node.js s'installe.

Après cela vous pourrez programmer les cartes Pycom avec VSC.

Avant d'installer Atom, il faut installer quelques programmes afin de pouvoir utiliser pymakr.

Installation de Python

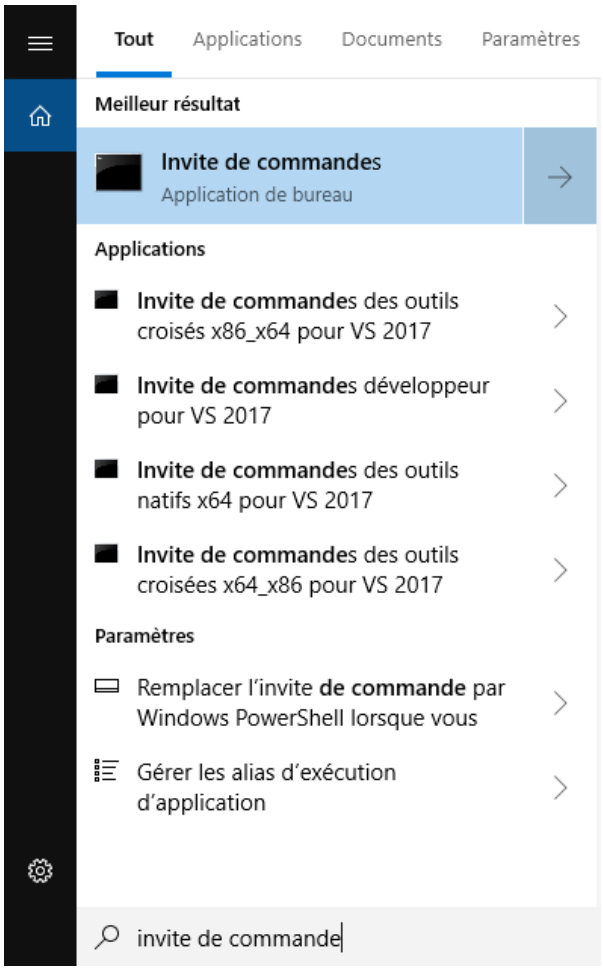
Rendez-vous sur le site de Python. Et cliquer sur 'download'. Cela permet de télécharger la dernière version de Python.

Cliquer sur le paquet pour installer python.

Cocher la case 'Add Python 3.X to PATH' (*Ajouter Python 3.5 au PATH*). Il est très important de cocher cette case sinon vous devrez le faire vous-même.

Cliquer sur installer maintenant.

Une fois fini, testez que Python est bien installé en lançant votre invite de commande.



Une fois lancé, taper 'python'

```
C:\Users\tihamon>python
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Si vous obtenez les symboles '>>>' python est bien installé.

Installation de Atom

Rendez-vous sur le site de Atom.

Cliquer sur ‘download’.

Cliquer sur le fichier télécharger pour installer Atom.

Avant d’installer Pymakr, il faut installer un compilateur C/C++. Pour cela vous pouvez regarder sur Internet ou installer Visual Studio Code comme expliquer au début.

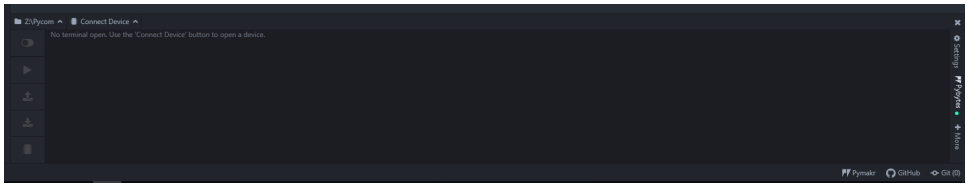
Installation de Pymakr (Atom)

Lancer Atom et cliquer sur ‘file’ puis ’settings’.

Un menu s’ouvre sur la gauche, cliquer sur

Taper ‘pymakr’, sélectionner le premier et appuyer sur le bouton ‘install’.

Une fois installé une fenêtre s’ouvre en bas de l’écran. Pymakr est bien installé.



Guide d'installation et de configuration des serveurs

Attention, une connexion à internet est nécessaire !

Installation du LoRa Server Application 2

Le LoRa Gateway Bridge 2

Installation du LoRa Server 3

Installation du LoRa App Server 4

Installation des modules de base de données et d'affichage des courbes 5

Installation de InfluxDB 5

Installation de Grafana 6

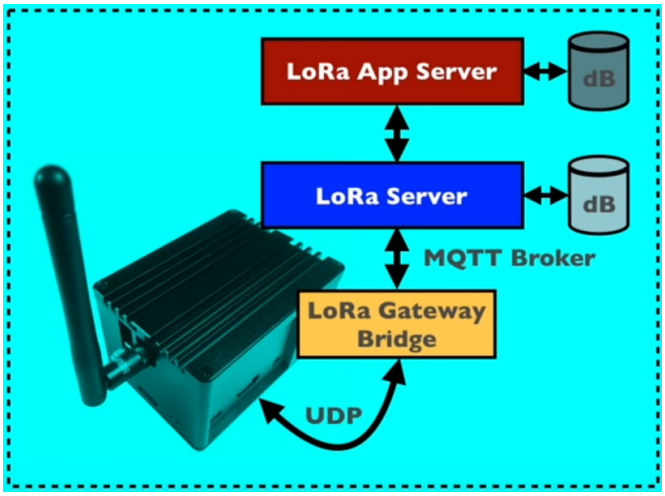
Installation du service NTP 6

Configuration du serveur LoRa et de la BDD 7

Configuration du LoRa Server Application 8

Configuration de la BDD et de Grafana 13

Ce document a pour but d'aider à installer le LoRa Server qui permet de réceptionner les données, les traiter et les rediriger. Le LoRa Server est composé de trois modules distincts comme vous pouvez le voir ici :



Le LoRa Gateway Bridge sert à faire le lien entre la Pygate et le server, le LoRa Server sert à gérer les différentes Gateway comme la PyGate. Enfin, le LoRa App Server sert à récupérer les données du LoRa server afin de les traiter. Le serveur LoRaWAN est celui mis à disposition par StackChirp. Cette fiche technique est fortement inspirée de celle proposée par StackChirp.

Installation du LoRa Server Application

Avant de commencer l'installation sur le Raspberry Pi, il faut avoir un OS de disponible. Nous avons opté pour le classique Raspbian. Si votre carte SD n'est pas prête, il faut la préparer à l'aide du logiciel disponible sur le site officiel. Il faut ensuite installer les mises à jour grâce à la commande :

```
sudo apt-get update && sudo apt-get upgrade -y
```

Le LoRa Gateway Bridge

Afin que la Gateway puisse communiquer avec les autres modules, il faut installer Mosquitto qui permet de gérer le transfert de données entre deux entités.

```
sudo apt-get install mosquitto
```

Afin de sécuriser les installations, il faut installer dirmngr :

```
sudo apt install apt-transport-https dirmngr
```

Une fois installé, on peut récupérer une clé de certification pour télécharger d'autres paquets.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 1CE2AFD36DBCCA00
```

Pour installer le LoRa Gateway Bridge, il faut mettre dans un fichier la source où l'on peut trouver les paquets que l'on souhaite grâce à la commande :

```
sudo echo "deb https://artifacts.loraserver.io/packages/2.x/deb stable main" | sudo tee /etc/apt/sources.list.d/loraserver.list
```

Une fois fait, il suffit de mettre à jour les dépôts du Raspberry Pi et de demander d'installer le LoRa Gateway Bridge :

```
sudo apt-get update
```

```
sudo apt-get install lora-gateway-bridge
```

L'installation terminée, il faut lancer le module et voir s'il est bien activé :

```
sudo systemctl start lora-gateway-bridge
```

```
sudo systemctl status lora-gateway-bridge
```

Vous pouvez aussi faire cette commande pour vérifier que le lien avec Mosquitto est en place :

```
journalctl  -f -n 50 -u lora-gateway-bridge
```

Enfin, vous pouvez faire en sorte que ce module soit lancé à l'allumage de la carte sinon vous devrez le faire à la main à chaque fois.

```
sudo systemctl enable lora-gateway-bridge
```

Voilà le LoRa Gateway Bridge est en place.

Installation du LoRa Server

Pour que le Lora Server fonctionne, il faut installer PostgreSQL. C'est un gestionnaire de Base de Données qui stockera les informations de la configuration du server.

Remarque : attention à bien mémoriser les couples login-mot de passe !

```
sudo apt-get install postgresql
```

Afin de gérer les différentes bases qui seront créées il faut installer Redis server :

```
sudo apt-get install redis-server
```

Une fois ces modules installés, nous allons créer un utilisateur loraserver_ns avec un mot de passe qui modifiera automatiquement les données quand on configurera le server.

```
sudo -u postgres psql

create role loraserver_ns with login password 'dbpassword';
```

(Changer dbpassword par un mot de passe plus sécurisé)

Une fois l'utilisateur créer on lui assigne une base de données.

```
create database loraserver_ns with owner loraserver_ns;
```

Pour vérifier que la base est bien créée, on quitte postgres et on se reconnecte avec l'identifiant loraserver_ns.

```
\q

psql -h localhost -U loraserver_ns -W loraserver_ns

\q
```

Vous devriez obtenir un texte comme se ci au début de la ligne "loraserver_ns=>". Si c'est bon, vous pouvez lancer l'installation du LoRa Server.

```
sudo apt-get install loraserver
```

Une fois installé, il faut modifier un fichier effectuer les commandes suivantes :

```
sudo su

cd /etc/loraserver

nano loraserver.toml
```

Il faut modifier une ligne afin que le LoRa Server accède à la base de données créée. Faites ctrl+W et tapez 'dsn' puis entrée. Refaire jusqu'à tomber sur cette ligne :

dsn="postgres://localhost/loraserver_ns?sslmode=disable"

Remplacer la par :

dsn="postgres://loraserver_ns:dbpassword@localhost/loraserver_ns?sslmode=disable"

(Mettez le mot de passe que vous aviez choisi précédemment à la place de dbpassword)

Quitter en sauvegardant en faisant ctrl+X puis O ou Y suivant si vous êtes en Français ou en Anglais. Puis entrée. Tapez ‘exit’ pour sortir du mode administrateur.

Lancer le module, vérifiez qu’il est lancé et faites en sorte qui se lance à l’allumage de la carte :

sudo systemctl start loraserver

sudo systemctl status loraserver

sudo systemctl enable loraserver

Voilà le LoRa Server est en place.

Installation du LoRa App Server

De même que pour le LoRa Server, le LoRa App Server à besoins d’une base de données. Il faut donc créer un nouvel utilisateur et une nouvelle base.

sudo -u postgres psql

create role loraserver_as with login password 'dbpassword';

create database loraserver_as with owner loraserver_as;

(Pour plus de sécurité, ne mettez pas le même mot de passe que le précédent)

Il faut créer une extension à cette base grâce aux commandes suivantes :

\c loraserver_as

create extension pg_trgm;

Une fois fait, on quitte et on se reconnecte pour vérifier que cela marche.

\q

psql -h localhost -U loraserver_as -W loraserver_as

\q

Si c’est bon, vous pouvez lancer l’installation du LoRa Server comme ceci au début de la ligne “loraserver_as=>".

Lancer l’installation du module grâce à :

sudo apt-get install lora-app-server

Une fois terminé, on génère une clé qui servira de certificat afin de sécuriser la connexion au serveur.

```
openssl rand -base64 32
```

(Copier la ligne obtenue et garder là. Elle est importante pour plus tard !)

Il faut maintenant compléter le fichier de configuration de ce module.

```
sudo su

cd /etc/lora-app-server

nano lora-app-server.toml
```

Il faut modifier une ligne afin que le LoRa App Server accède à la base de données créée. Faites ctrl+W et tapez ‘dsn’ puis entrée. Refaire le jusqu’à tomber sur cette ligne :

```
dsn="postgres://localhost/loraserver_as?sslmode=disable"
```

Remplacer la par :

```
dsn="postgres://loraserver_as:dbpassword@localhost/loraserver_as?sslmode=disable"
```

Il faut modifier une ligne afin que le LoRa App Server sécurise l’accès. Faites ctrl+W et tapez ‘jwt_secret’. Entre les guillemets coller la clé précédemment générée. Puis sauvegarder et quittez (ctrl+X).

Quittez le mode administrateur ‘exit’.

Lancer le module, vérifiez qu’il est lancé et faites en sorte qui se lance à l’allumage de la carte :

```
sudo systemctl start lora-app-server

sudo systemctl status lora-app-server

sudo systemctl enable lora-app-server
```

Voilà le LoRa App Server est en place.

Installation des modules de base de données et d’affichage des courbes

Installation de InfluxDB

InfluxDB est une base de données temporelle. Ça veut dire qu’elle gère les données en fonction du temps. Pour l’installer, il faut commencer par faire ces commandes :

```
sudo apt-get install apt-transport-https

sudo apt-get install curl
```

Curl nous permet d’obtenir une clé grâce à la commande :

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
```

Puis il faut ajouter une source d’installation pour influxdb :

```
echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

Enfin installer InfluxDB grâce à ces commandes :

```
sudo apt-get update
sudo apt-get install influxdb
```

Voilà InfluxDB est installé.

Installation de Grafana

Grafana est un logiciel qui permet la visualisation de données. Il nous permettra de voir les données stockées dans InfluxDB. Pour installer Grafana, il faut obtenir une nouvelle clé :

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

On ajoute une nouvelle source d'installation :

```
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

Pour l'installer on fait comme influxDB :

```
sudo apt-get update
sudo apt-get install -y grafana
```

Mais il ne faut pas oublier de la lancer et de le démarrer à l'allumage de la carte.

```
sudo systemctl start grafana-server
sudo systemectl enable grafana
```

Voilà Grafana est installé.

Installation du service NTP

NTP est un protocole qui permet de synchroniser, via un réseau informatique, l'horloge locale d'ordinateurs. Dans notre projet, la PyGate se connecte au Raspberry pour obtenir l'heure pour cela, il faut installer NTP sur le Raspberry Pi. Une condition est que RPi ait un accès à internet pour récupérer l'heure. Une autre piste à explorer est l'utilisation d'un module RTC avec une pile pour un coût de l'ordre de 10 à 15€.

```
sudo apt install ntp
```

Pour que le Raspberry devienne server NTP il faut modifier sa configuration.

```
sudo nano /etc/ntp.conf
```

Et ajouter cette ligne à la fin du fichier :

```
restrict 10.254.90.0 mask 255.255.255.0
```

Remplacer 10.254.90.0 et 255.255.255.0 par l'adresse Ipv4 de votre Raspberry Pi.

Pour cela faites la commande :

```
netstat –rn
```

Et prenez la ligne dont l’”iface” est “eth0” remplacer 10.254.90.0 par la “destination” et 255.255.255.0 par le “Genmask”.

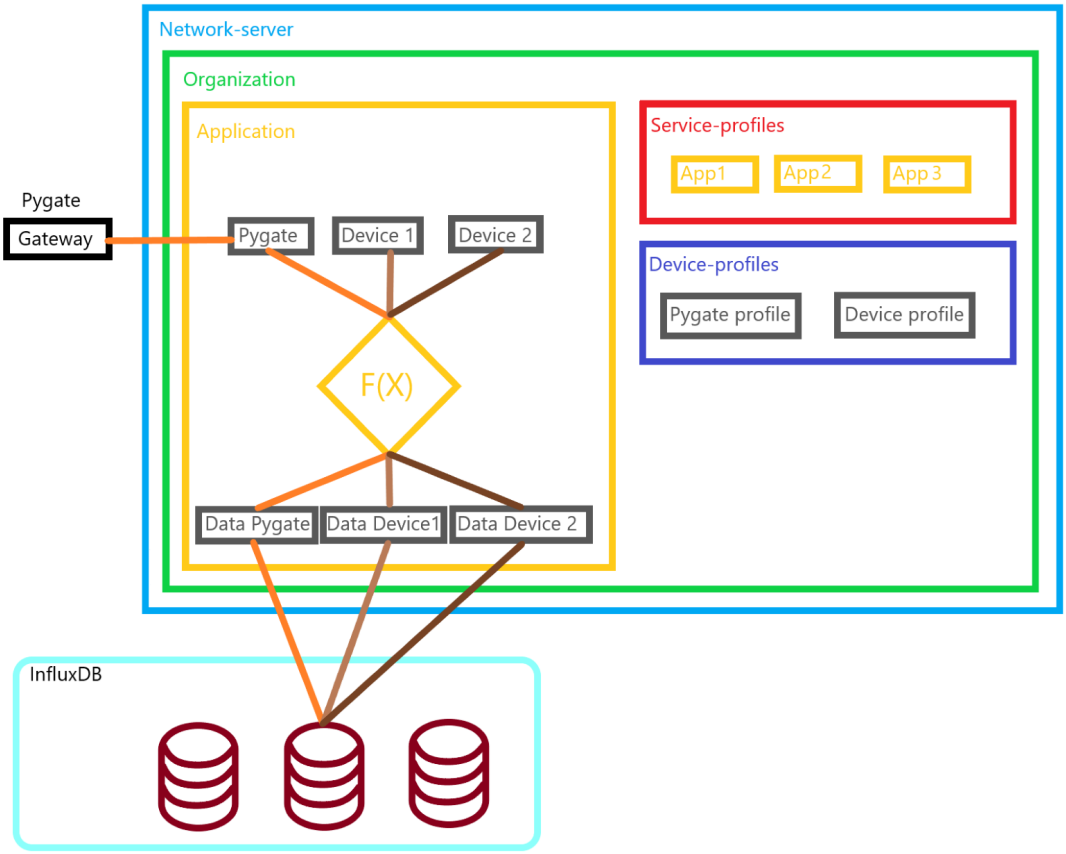
Quitter et sauvegarder le fichier, puis redémarrer le service.

```
sudo service ntp  start
```

Configuration du serveur LoRa et de la BDD

<u>Fonctionnement ChirpStack LoRa Server</u>

Le fonctionnement du ChripStack LoRa Server n’est pas simple à comprendre au premier abord alors voici un résumé de son fonctionnement. Vous serez libre de choisir les noms et ce schéma a pour objectif d’éclairer l’organisation afin de nommer correctement les différents éléments.



Comme vous pouvez le voir il y a beaucoup de choses différentes pour envoyer des données dans une BDD. Sur ce schéma les noms des boites en haute à gauche correspondent à une partie du LoRa Server.

Pour procéder dans l’ordre il y a tout d’abord le Network-Server qui correspond à l’application ChirpStack LoRa Server. Le Network-Server peut contenir plusieurs Organisations. On peut imaginer un Network-Server qui est utilisé par plusieurs apiculteurs où chaque Organisation correspond à un apiculteur.

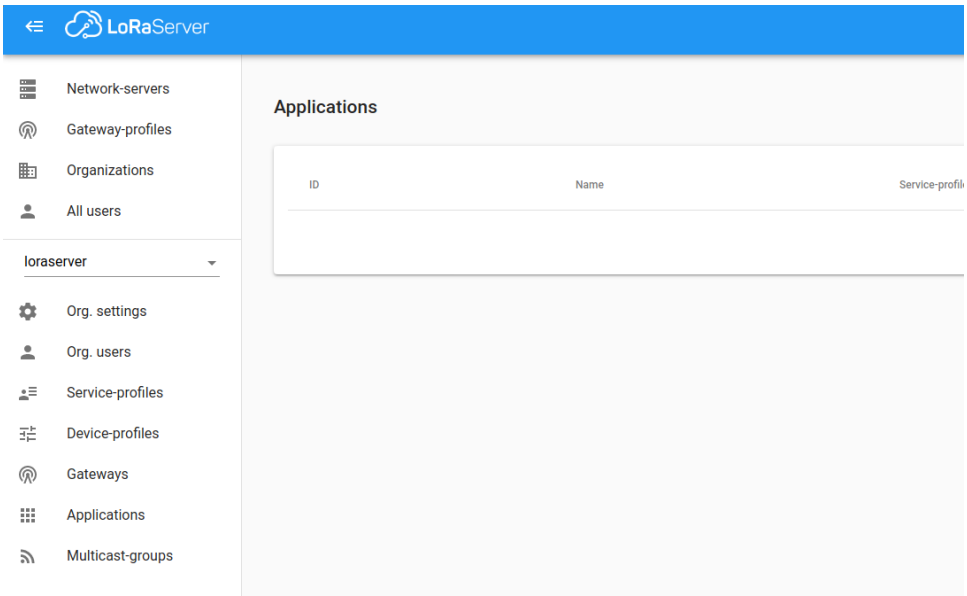
Ensuite, chaque organisation possède ses propres Service-profiles, Device-profile, applications et des utilisateurs. Une Organisation peut correspondre à un apiculteur ou à un rucher.

Pour mieux comprendre les Service-profiles et les Device-profiles il faut comprendre comment marche une application. L'application sert à transformer les données reçues par la Pygate en données compréhensibles pour les stocker. Dans notre cas les données sont transformées grâce au langage Javascript. Ces données sont ensuite envoyées dans une base de données InfluxDB, une application enverra toujours dans la même BDD. Pour utiliser une autre BDD il faut créer une autre Application.

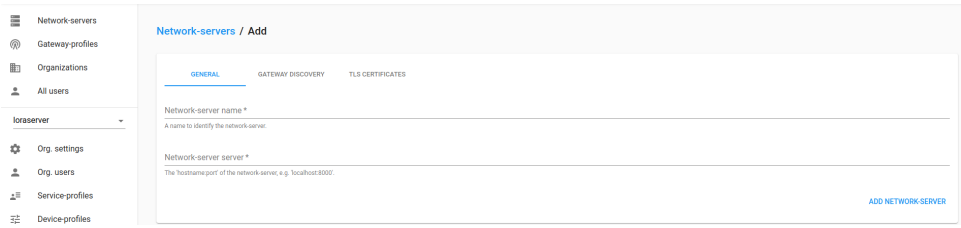
Configuration du LoRa Server Application

Une fois que tout est bien installé, il faut configurer le serveur LoRa. Pour cela, ouvrez le navigateur sur le Raspberry Pi. Dans la barre de recherche, entrez "127.0.0.1:8080". Il est également possible d'y accéder depuis un ordinateur sur le réseau en tapant l'adresse du Raspberry avec ":8080".

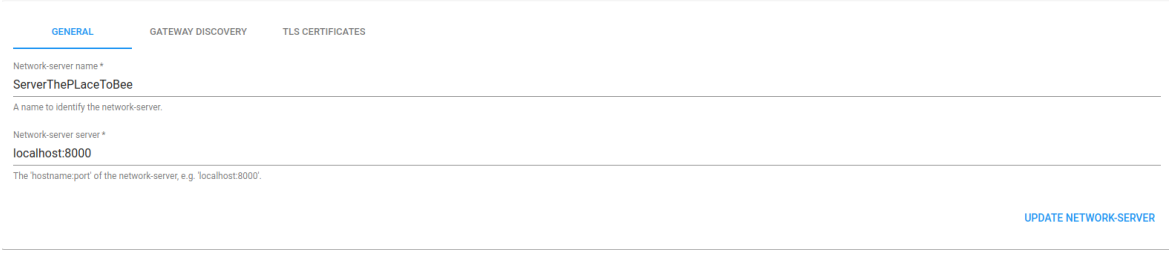
Pour vous connecter, il faut utiliser le login "admin" et le mot de passe "admin" (pensez à le modifier pour des raisons de sécurité). Vous arriverez sur cette page.



Rendez-vous dans Network-servers et cliquez sur "add" en haut à droite.



Il faut remplir les deux lignes comme ci-dessous :

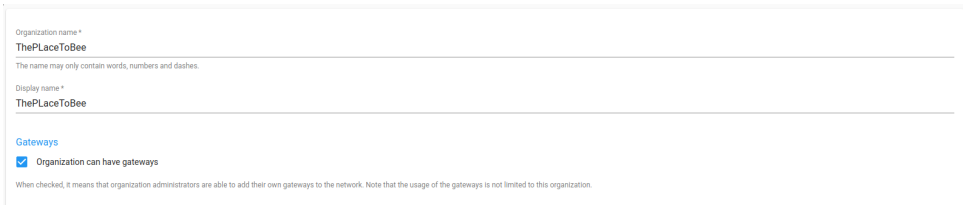


Pour le “network-server-name" vous pouvez mettre ce que vous voulez, mais laisser pour le deuxième, il est conseiller de mettre "localhost:8080". Dans “Gateway Discovery” décochez la case et dans “TLS Certificates” laisser tout vide. Cliquer sur "Add network server".

Dans le menu cliquer sur "organisations". Il y en a déjà une, mais nous allons créer une nouvelle.

Cliquer sur "Create" en haut à droite.

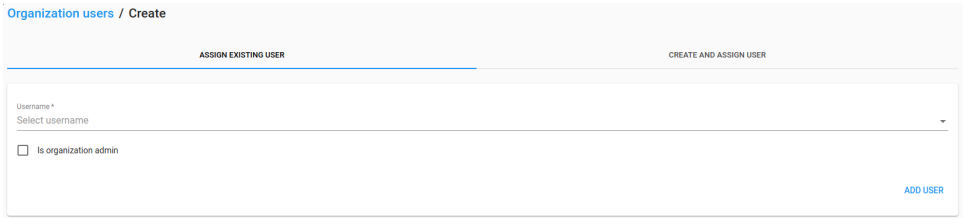
Rempliez les deux champs, il est préférable de mettre les mêmes noms pour ne pas trop complexifier le tout.



Et cocher la case avant d'appuyer sur "create organization" en bas à droite de la boite.

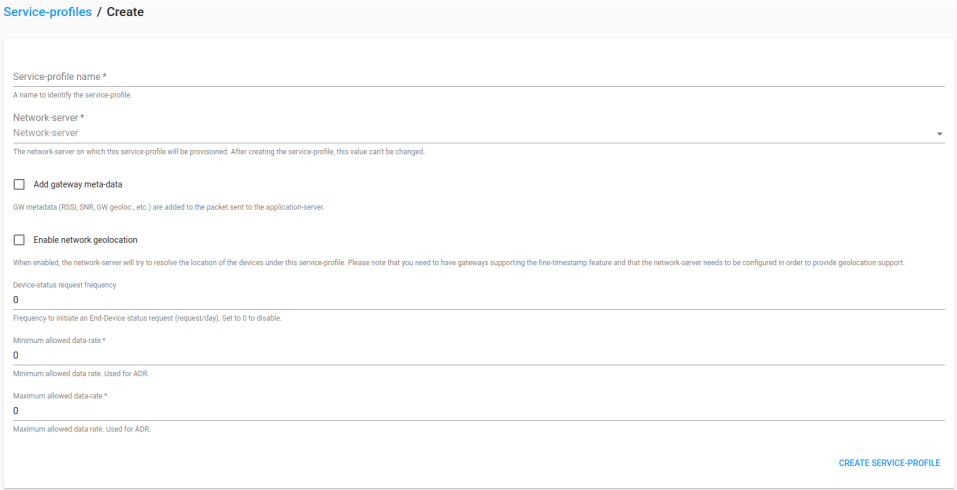
Nous allons maintenant passer à la partie application du serveur. Cliquer sur “Loraserver” puis sélectionner votre organisation.

Dans le menu et aller dans "Org. Users" pour ajouter un nouvel utilisateur en cliquant sur "ADD".

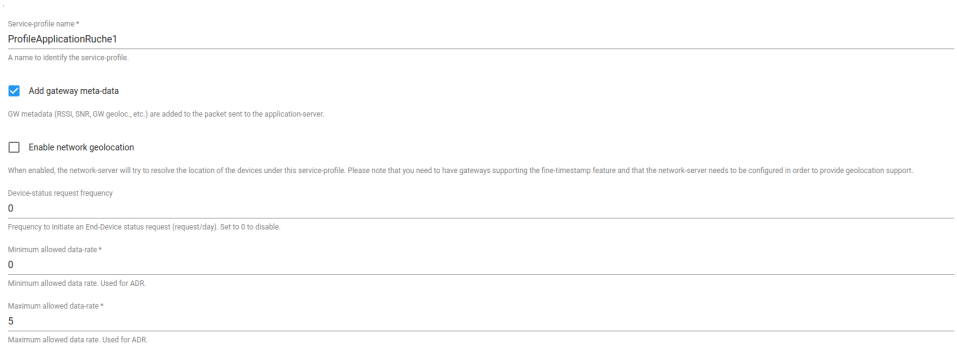


On ajoute un utilisateur déjà existant. Cliquez sur "select username" et sélectionner "admin". Cochez la case pour que l'utilisateur soit administrateur de l'organisation. Puis cliquer sur "Add users".

Maintenant, il faut créer un profile pour l'application. Dans le menu sélectionner "Service-Profiles". Créez un profile en cliquant sur "create".

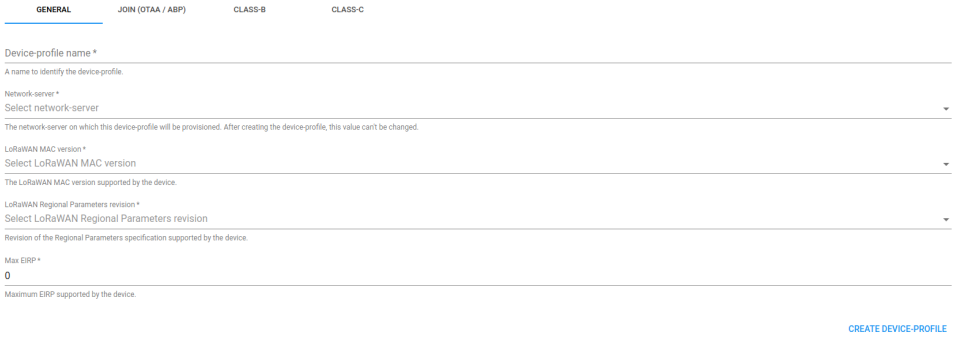


Dans le premier champ, mettez un nom explicite, dans le second champ sélectionner votre serveur. Cocher la première case et passer la dernière ligne à 5 cela permet de diminuer la consommation électrique de la carte qui se trouve sur la ruche. Dans notre cas nous avons ceci :



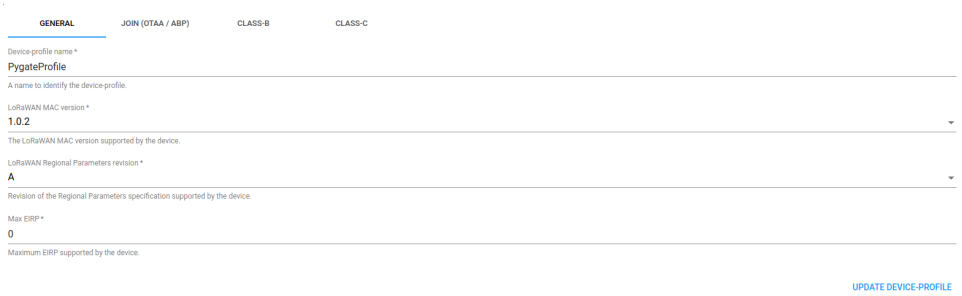
Enfin cliquer sur "create service profile".

Il faut maintenant créer un profile pour la Pygate. Dans le menu sélectionnez "Device-profiles" puis "create".

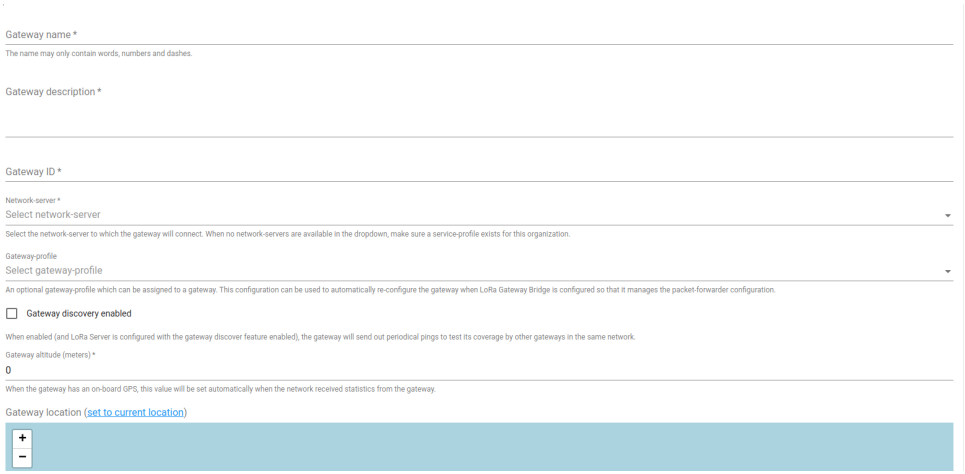


Dans le premier, champ mettez un nom explicite qui rappelle la Pygate. Dans le second champ sélectionné la version 1.0.2 de LORAWAN mac version. Dans le troisième sélectionnez la version B et laisser le dernier champ à zéro.

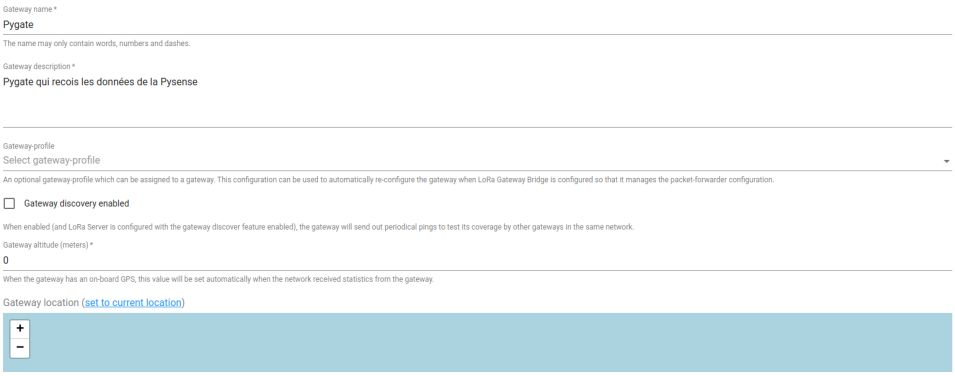
Dans la fenêtre "JOIN(OTAA/ABP)" cocher la case. Et laissez-les autres fenêtres vides et cliquez sur "create device profile".



Dans le menu sélectionner "gateway" pour configurer la Pygate. Dans "create" vous aurez cette boite :

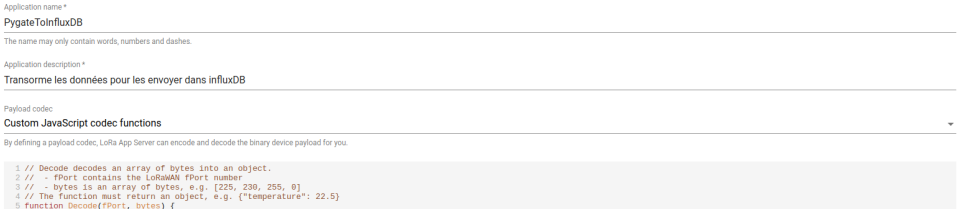


Dans le premier champ donner un nom qui permet de différencier (si vous en utilisez plusieurs) vos Pygate. Dans la description, vous pouvez mettre un petit texte pour plus de différentiation. Vous pouvez laisser le reste vide. Dans notre cas, nous avons ceci.

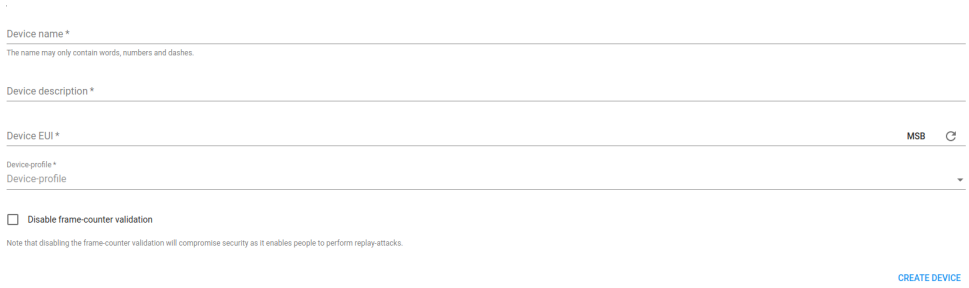


On peut maintenant créer l'application, sélectionner "application" dans le menu puis "create". Dans le premier champ donner un nom explicite à votre application et apporter des précisions dans le second champ si besoin. Dans le troisième champ, sélectionner le profile que l'on vient de créer. Dans le dernier champ passer de "none" à "Custom JAVAScript codec functions".

Vous pouvez appuyer sur "create application". Dans notre cas, nous avons ceci.



Une fois fait, restez dans le menu “application” et cliquez sur l'application que vous venez de créer. Il faut maintenant créer un device en cliquant sur "create".



Dans le premier champ, mettez un nom qui permet de différencier les différentes Pygate (si vous en avez plusieurs). Dans le deuxième champ, vous pouvez apporter des précisions. Le troisième champ permet de générer une clé qu'il faut recopier dans le code de la Pygate. Puis choisissez le profil que vous avez créé précédemment et cocher la case. Vous pouvez cliquer sur "create device". Dans notre cas, nous avons ceci :

Vous serez redirigé vers une nouvelle boite "keys (OTAA)". Comme précédemment, vous aller générer une clé qu'il faudra copier dans le code de la Pygate.

Vous pouvez maintenant brancher vos cartes avec les codes, vous devriez voir apparaître des trames dans "application/VOTREAPPLI/device/VOTREDEVICE/Live loraWan Frames".

Il faut maintenant vous rendre dans "application/VOTREAPPLI/application configuration". Ne touchez pas aux premiers champs, nous les avons déjà configurés rendez-vous dans la première fenêtre, la fonction DECODE(). Dans cette fonction, vous allez récupérer les données et les mettre en forme pour qu'on puisse les envoyer dans InfluxDB.

```
function Decode(fPort, bytes) {
  var temp = bytes[0]
  //var var1 = bytes [1]
  //var var2 = bytes [2]
  return {
    //'var1': var1
    //'pression':(var2/128)-1
    'temperature': temp
  };
}
```

Voici un exemple avec la ruche qui renvoie la température. Chaque variable envoyée sera stockée dans le tableau bytes. Pour récupérer une donnée, il faut faire bytes[x] où x va de 0 au nombre des variables moins 1. Une fois récupéré dans la balise retour, il suffit de faire comme dans l'exemple. (Les lignes avec "//" ne sont pas prise en compte par le code). Il est important d'être cohérent avec le codage effectué sur le nœud de la ruche.

Il ne reste plus qu'à relier le LoRa Server Application avec InfluxDB mais avant, il faut configurer cette dernière.

Configuration de la BDD et de Grafana

Ouvrez le terminal et tapez la commande :

Influx

Si influxDB est bien installé vous obtiendrez ceci

```
Connected to http://localhost:8086 version 1.8.5
InfluxDB shell version: 1.8.5
>
```

Sinon allez voir comment installer influxDB plus haut.

Avant de créer une base de données, il faut créer un utilisateur. Nous allons créer un utilisateur qui correspond au LoRa Server Application afin qu'il puisse écrire les données dans la BDD.

create user NomUtilisateur with password MotDePasse

Utilisez la commande SHOW USERS pour voir s'il est bien créé. (Dans l’exemple, il s'appelle "loraserver")

```
> SHOW USERS
user       admin
----       -----
loraserver false
```

On crée ensuite la BDD avec la commande :

create database NomDeVotreBDD

Utilisez la commande “SHOW DATABASES” pour vérifier qu'elle est bien créée.

```
> SHOW DATABASES
name: databases
name
----
_internal
ruche1
```

Une fois l'utilisateur créé, on retourne sur le LoRa Server Application dans "Application/VOTREAPPLI/intégration" puis cliquer sur "create".

Il vous proposera "influxDB" et vous le sélectionnez. Remplir les champs de cette manière :

Integration kind *
InfluxDB integration

InfluxDB integration configuration

API endpoint (write) *
http://localhost:8086/write

Username

Password

Database name *

Retention policy name
Sets the target retention policy for the write. InfluxDB writes to the DEFAULT retention policy if you do not specify a retention policy.

Timestamp precision *
Second
It is recommented to use the least precise precision possible as this can result in significant improvements in compression.

Dans le champ "username", mettez le nom de l'utilisateur que l'on a créé sur influxDB avec son mot de passe dans "password". Dans "database name" mettez le nom de la BDD que vous avez créée. Vous pouvez cliquer sur "create integration".

Si vous avez tout bien configuré et que vos cartes sont en marche, vous devriez pouvoir voir les données dans influxDB grâce aux commandes (Pour aller dans la BDD faites “use NomDeVotreBase”) :

SHOW MEASUREMENTS

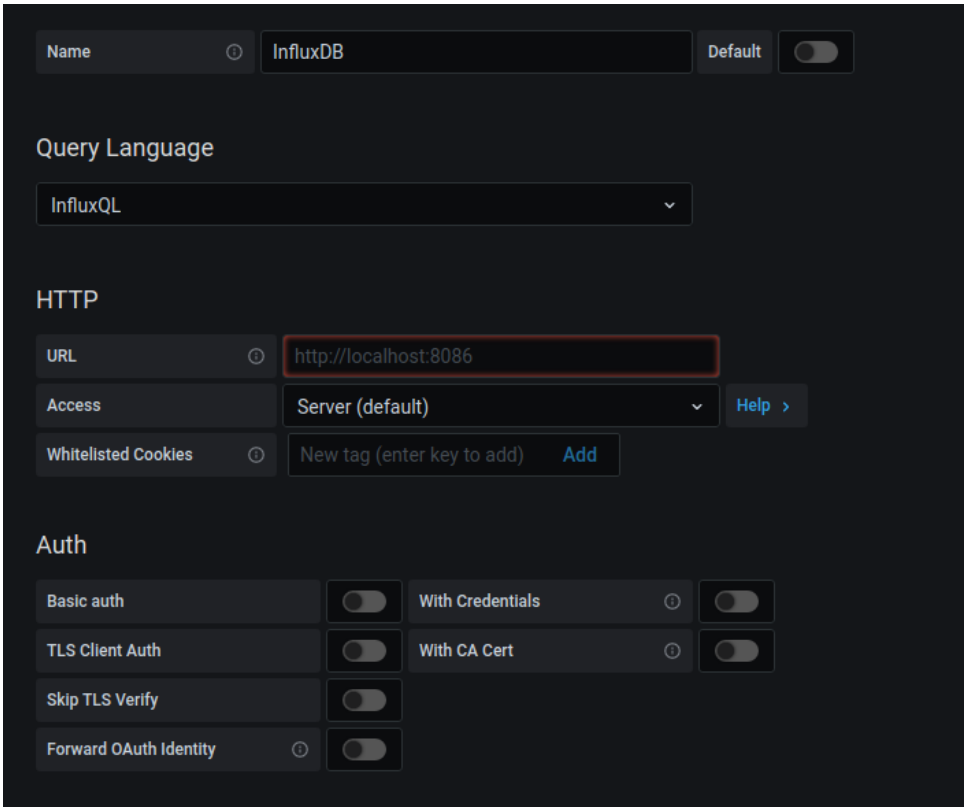
SELECT * FROM uneDesValeursObtenuAvecLaCommandePrecedente

On peut désormais configurer Grafana, rendez-vous sur votre navigateur et taper "127.0.0.1:3000". De la même manière que pour le server LoRaWAN, Grafana est accessible avec un ordinateur sur le réseau. Si l'installation a bien été faite, vous tomberez sur la page de Grafana sinon retournez voir comment installer Grafana et le lancer.

Connectez-vous avec le login/mot de passe "admin/admin" (encore une fois, pensez à modifier le mot de passe pour la sécurité).

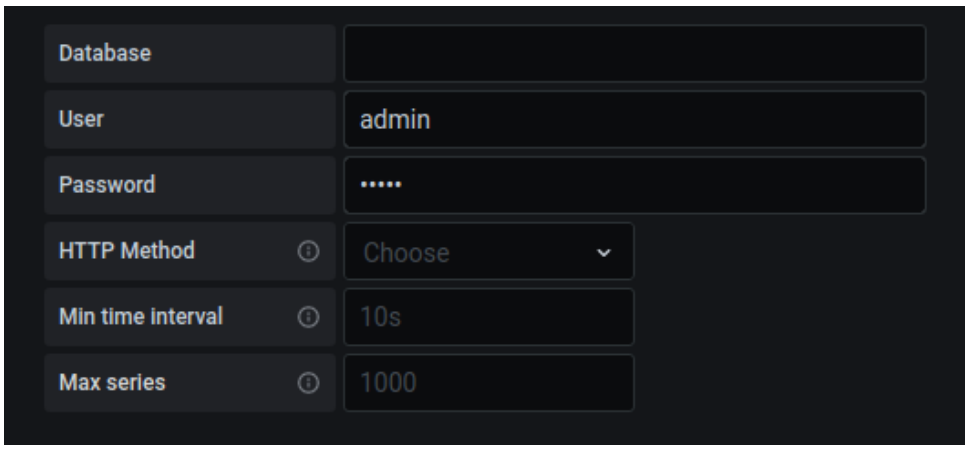
Sur le menu de gauche, cliquer sur l'engrenage et sélectionner "data source". Cliquer sur le bouton "add data source". Et sélectionner influxDB.

Vous arriverez sur cette page :



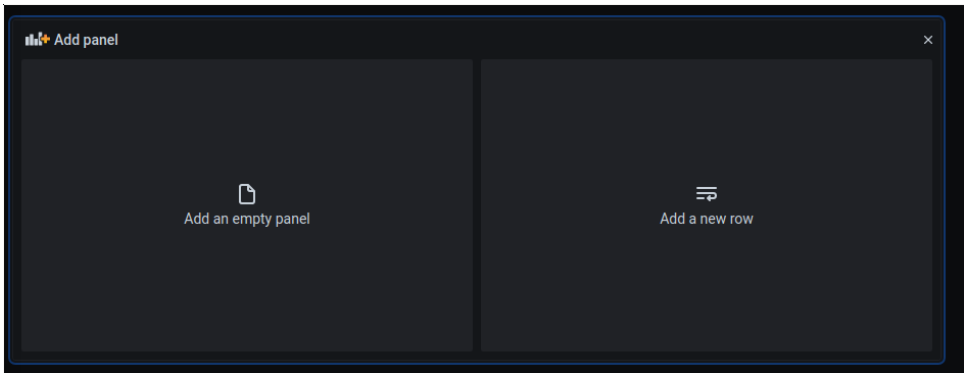
Dans "name" il est conseillé de mettre le même nom que la base de données créée sur influxDB.

Dans url tapez "http://localhost:8086" et laisser le reste de cette partie. Allez directement au bas de la page.

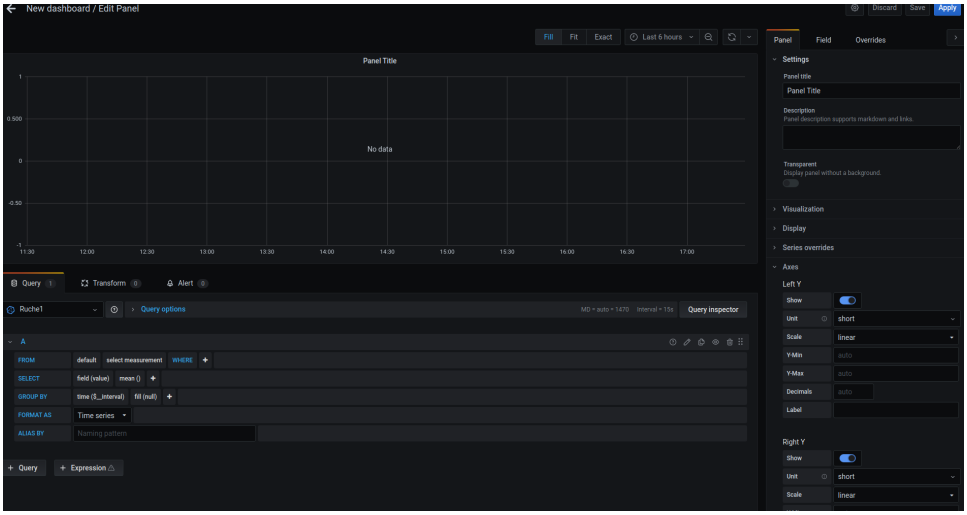


Dans "name” il est conseillé de mettre le même nom que la base de données créée sur influxDB.

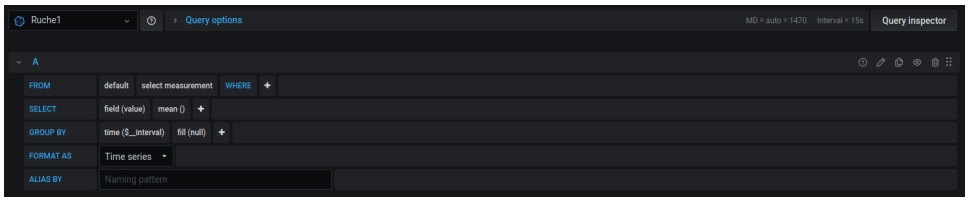
Dans url taper "http://localhost:8086" et laisser le reste de cette partie. Allez directement au bas de la page.



Cliquer sur "add an empty panel" une fenêtre va s'ouvrir :



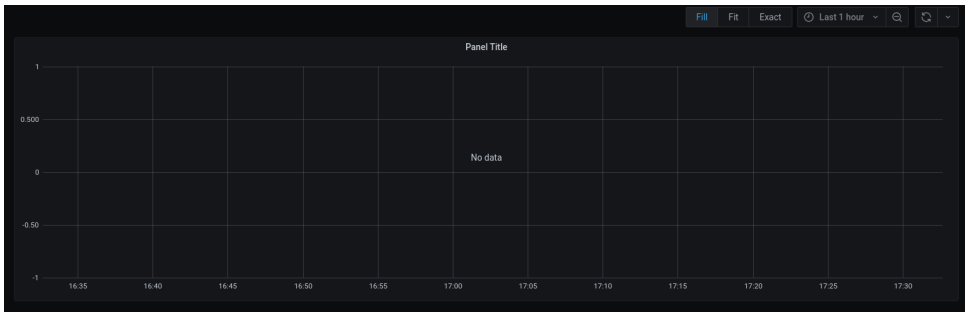
Dans cette partie, cliquer sur le petit crayon qui se trouve à droite du 'A'



Grâce à cela vous pourrez taper vos commandes de requête à la base de données, en haut à gauche vous pouvez sélectionner la BDD source. Pour obtenir les données il faut faire une requête qui ressemble à :

SELECT value FROM device_frmpayload_data_temperature

Après le "FROM" mettez la réponse à la requête "SHOW MEASUREMENTS" que vous avez fait dans InfluxDB.



En haut à droite vous pouvez sélectionner l'intervalle de l'affichage, et l'intervalle entre les rafraichissements du graphique. Sauvegardez et vous verrez vos données apparaitre et mises à jour régulièrement.

Si vous avez d'autres variables à afficher, vous pouvez créer une deuxième requête sur le même graphique mais pour plus de clarté nous conseillons de créer un nouveau "panel".

Configuration des cartes PyCom

Dans cette partie, nous allons voir comment configurer la PyGate et la connexion LoRa avec quelques explications sur le fonctionnement du code.

Configuration des cartes Pycom 1
Mise à jour des firmwares 1
PyGate et LoPy 1
Main.py 2
Config.json 3
Connexion LoRa du nœud de la ruche 4

Mise à jour des firmwares

Il est important de mettre à jour les firmwares des modules PyCom mais aussi des cartes d'accueil. Ce sont les programmes bas niveau permettant l'utilisation du matériel et notamment le langage de programmation MicroPython. Afin de profiter de toutes les fonctionnalités et être en cohérence avec les commandes, il faut utiliser la dernière version. Attention, il y a deux procédures distinctes pour la mise à jour du module et celle de la carte d'accueil. Il est possible de récupérer la version installée grâce au plugin Pymakr.

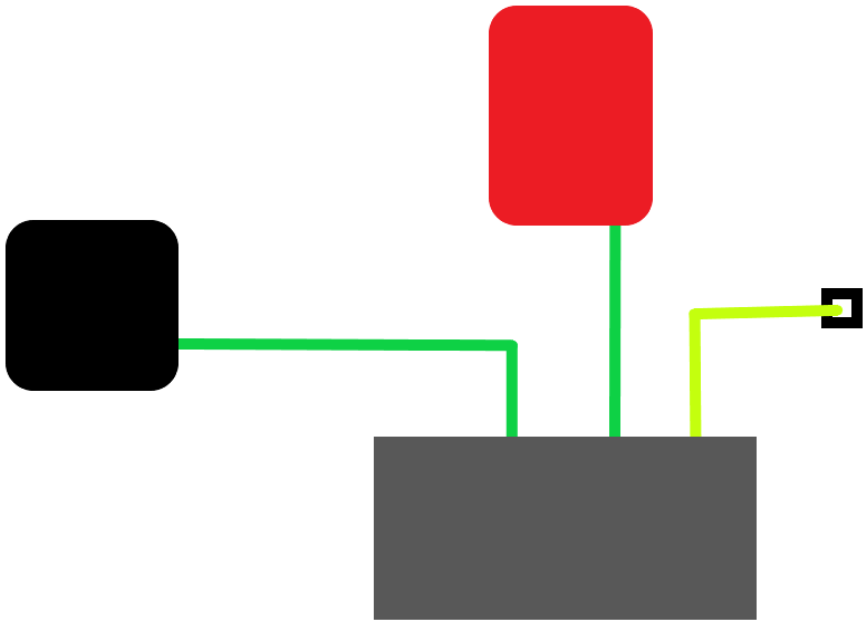
PyGate et LoPy

Plus précisément, le point d'accès est formé de 3 produits PyCom : une LoPy dont le microprocesseur ESP32 sera le cerveau, la carte d'accueil PyGate qui contient le concentrateur LoRa et le module Ethernet PoE.

Le fonctionnement de la LoPy est modifié afin de l'utiliser en point d'accès. Il faut alors la reprogrammée avec l'outil de mise à jour du firmware en précisant PyGate. Remarque : les modules WiPy et GPy sont aussi compatibles avec la PyGate.

Pour sa configuration, il y a deux fichiers. Le premier main.py, c'est le fichier dans lequel on met le code et qui gère la connexion au serveur. Le second config.json, c'est un fichier qui configure les modes de télécommunication et aussi de connexion au serveur. Il faut les placer dans le même répertoire de projet. Ils seront ensuite téléversés dans le module PyCom.

Avant de commencer, branchez le réseau de la manière indiquée sur le schéma suivant : votre Pygate (Noir), votre Raspberry Pi (Rouge) et votre Switch (Gris).



Veillez à ce que la Pygate ne soit pas connectée à un port PoE pendant qu'elle est branchée en USB à votre PC car il y a un risque de destruction. En Vert, ce sont les câbles réseaux. Le plus clair, à droite est un câble qui peut être branché sur votre box offrant un accès à internet.

Main.py

Revenons-en à la Pygate. Dans le main, commencez par récupérer l'identifiant de votre Pygate grâce au code suivant :

```
from network import LoRa
import ubinascii

lora = LoRa()
print("DevEUI: %s" % (ubinascii.hexlify(lora.mac()).decode('ascii')))
```

Il suffit de cliquer sur "Run" une fois que votre Pygate est connectée à Atom ou Visual Studio Code.

Récupérer la chaîne de caractère elle est importante pour la configuration de la Pysense.

Supprimer les lignes précédentes et remplacer les par ces lignes :

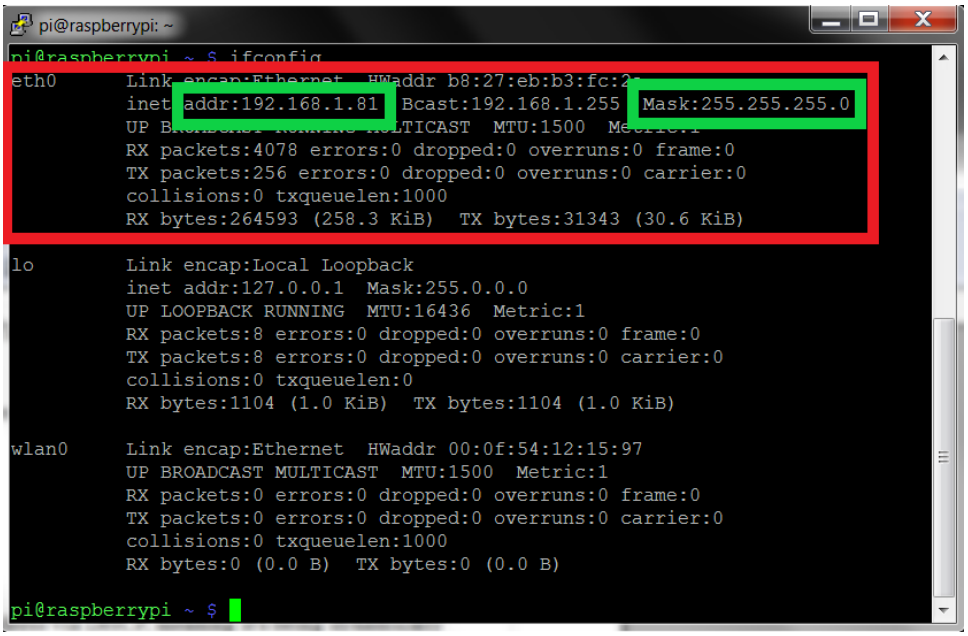
```
from network import WLAN
import ubinascii
wl = WLAN()
ubinascii.hexlify(wl.mac())[:6] + 'FFFE' + ubinascii.hexlify(wl.mac())[6:]
```

Vous obtiendrez un code de ce type "b'240ac4FFFE008d88'" récupérer juste la partie entre guillemet,"240ac4FFFE008d88" il sera important pour le fichier config.json.

Vous pouvez supprimer les lignes précédentes et copier le code fourni pour le main de la Pysense. Rendez-vous dans la partie de la configuration réseau et modifier cette ligne.

```
eth.ifconfig(config=('0.0.0.0',0.0.0.0','0.0.0.0','0.0.0.0'))
```

Remplacez les quatre premiers zéros par l'adresse IP de votre Raspberry plus un. Pour l'obtenir rendez-vous sur le terminal de la Raspberry Pi. Tapez le commande "ifconfig", trois parties devrait s'afficher. C'est la partie avec "eth0" qui nous intéresse ici.



Remplacer par l'adresse, ici "192.168.1.81" donc on met "192.168.1.82". Puis pour les quatre zéros suivants, remplacez par le masque, ici "255.255.255.0". Pour les suivants remplacer par l'adresse de la Raspberry. Les derniers zéros sont pour le DNS, mais vous pouvez laisser tel quel.

Rendez-vous dans la partie suivante, la section NTP et modifier la ligne suivante.

```
rtc.ntp_sync(server="0.0.0.0")
```

Si vous avez installé NTP sur votre Raspberry Pi, changez les zéros par l'adresse de la Raspberry, sinon cela veut dire votre Pygate devra toujours avoir un accès à internet. Dans ce cas la remplacer les zéros par "pool.ntp.org". Cela permet à la Pygate de connaître l'heure. Pour un fonctionnement en autonomie totale, il faut ajouter un module RTC avec une pile pour sauvegarder l'heure en cas de coupure.

Voilà le fichier main.py est configuré.

Config.json

Copier le ficher fourni dans le fichier. Si vous habitez dans une autre région du monde que l'Europe rendez-vous sur ce site et récupérez le fichier correspondant à votre région du monde. Puis suivez les instructions de ce site pour le configurer.

Rendez-vous directement à la fin du fichier et remplacer les deux lignes suivantes :

```
"server_address": "0.0.0.0",
```

Remplacer les quatre zéros par l'adresse IP de votre Raspberry. Une fois fait, modifiez cette ligne :

```
"gateway_ID": "240ac4FFFE008d88",
```

C'est ici que vous devez mettre le code obtenu au tout début de cette partie.

Voilà le fichier config.json est configuré.

Avant de vous débrancher, faites “Run” avec le fichier main.py une fois que la DEL se sera allumée en vert cela voudra dire que la Pygate à accès à l’heure, elle est donc bien configurée. Vous pouvez téléverser le code vers la Pygate en faisant “upload”. Une fois terminé, vous pouvez vous débrancher et mettre la Pygate sur le port PoE. Vérifiez qu’elle est bien allumée avec les LED puis bien connectée avec la LED verte.

Connexion LoRa du nœud de la ruche

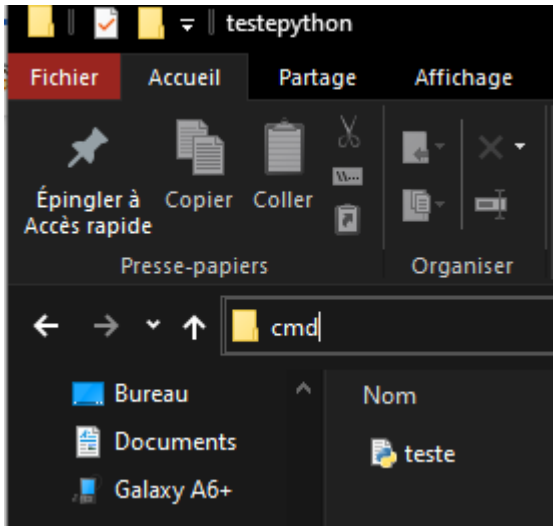
Avant de commencer à configurer la Pysense, veillez à avoir configuré le Serveur LoRa et la PyGate, le nœud a besoin de clés qui s’obtiennent pendant la configuration de la PyGate et du LoRa Server.

Dans notre cas, le nœud initialise tout ce dont il a besoin pour communiquer et récupère les données. Puis il tente de se connecter à la PyGate. Une fois connecté, il va récupérer les données, les mettre en forme pour envoi puis enfin se mettre en veille. Grâce à cela, la batterie pourra tenir plus d’un an sans avoir à la recharger.

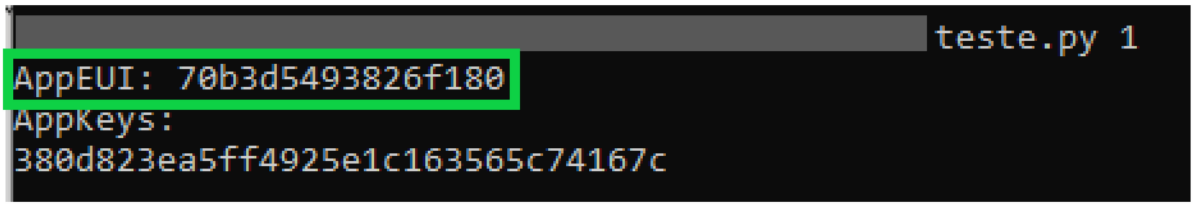
Rendez-vous dans la partie clé. Il y a trois clés, pour l’app_key il faut renseigner la clé que vous avez généré quand vous avez créé l’application sur le serveur. Pour le dev_eui il faut mettre l’identifiant de la PyGate que vous avez récupéré au tout début de la configuration de la Pygate. Pour l’app_eui créer un fichier teste.py et copier ce code :

```
DEFAULT_PREFIX= '70b3d549'
import secrets
import argparse



def gen_list(prefix, n):
    print("AppEUI: {}{}".format(prefix, secrets.token_hex(4)))
    print("AppKeys: ")
    for i in range(0, n):
        print(secrets.token_hex(16))


parser = argparse.ArgumentParser(description='Generate an App EUI and list
of AppKeys')
parser.add_argument('number_of_keys', type=int, help='number of AppKeys to
generate (default: 1)')
parser.add_argument('--prefix', type=str, default=DEFAULT_PREFIX, help='EUI
prefix to use (default:' + DEFAULT_PREFIX+ ')')
args = parser.parse_args()


gen_list(args.prefix, args.number_of_keys)
```

Ensuite exécutez ce code en mettant 1 comme paramètre. Pour cela, si vous êtes sous Windows, utiliser l’explorateur de fichier retrouver le fichier que vous avez créé.



Une fois trouvé, taper “cmd” comme sur l’image. Puis tapez “teste.py 1” et copier l’app_eui dans le main.py



Voilà votre nœud est prêt à être mis en place sur la ruche et à communiquer en LoRa.