Table des matières

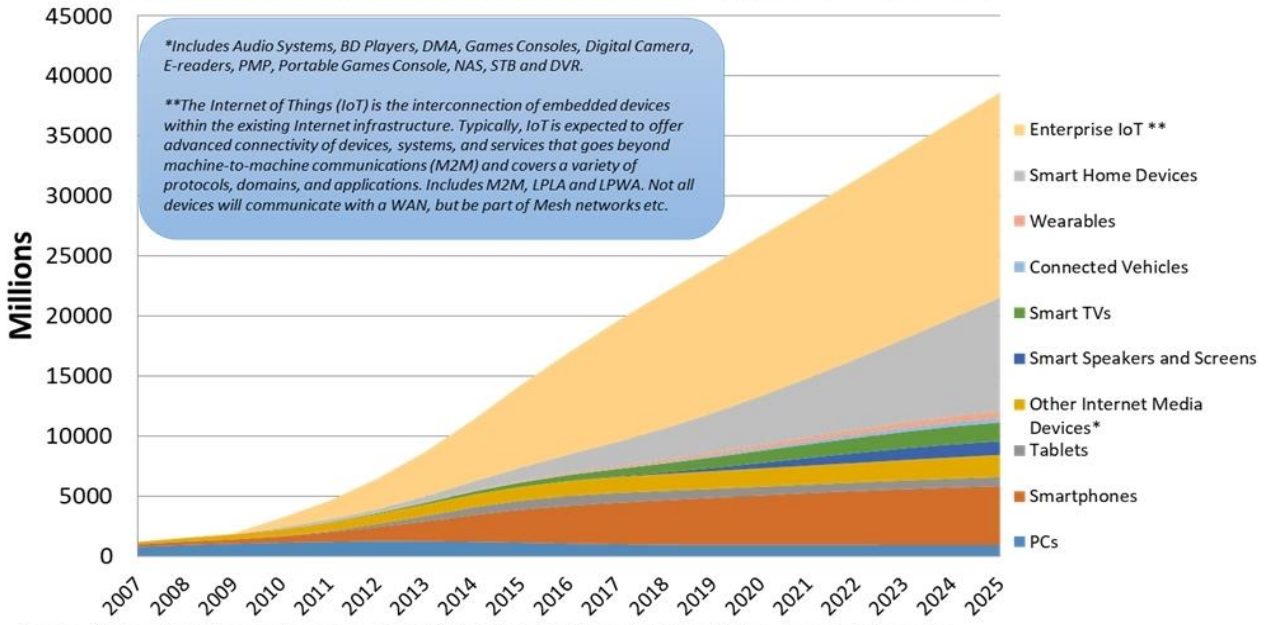
Liste des tableaux

Liste des figures

1. Introduction

L'Internet des objets est aujourd'hui devenu un incontournable dans notre société. En effet, les appareils de domotique sont de plus en plus implantés dans notre quotidien. Nous le retrouvons aujourd'hui jusqu'à nos appareils ménagers. L'avènement de l'ère de l'internet a permis la mise en place une très grande diversité d'appareils connectés.

Figure 1 : Prévision du nombre d'IoT dans le monde



Source – Strategy Analytics research services, May 2019: IoT Strategies, Connected Home Devices, Connected Computing Devices, Wireless Smartphone Strategies, Wearable Device Ecosystem, Smart Home Strategies

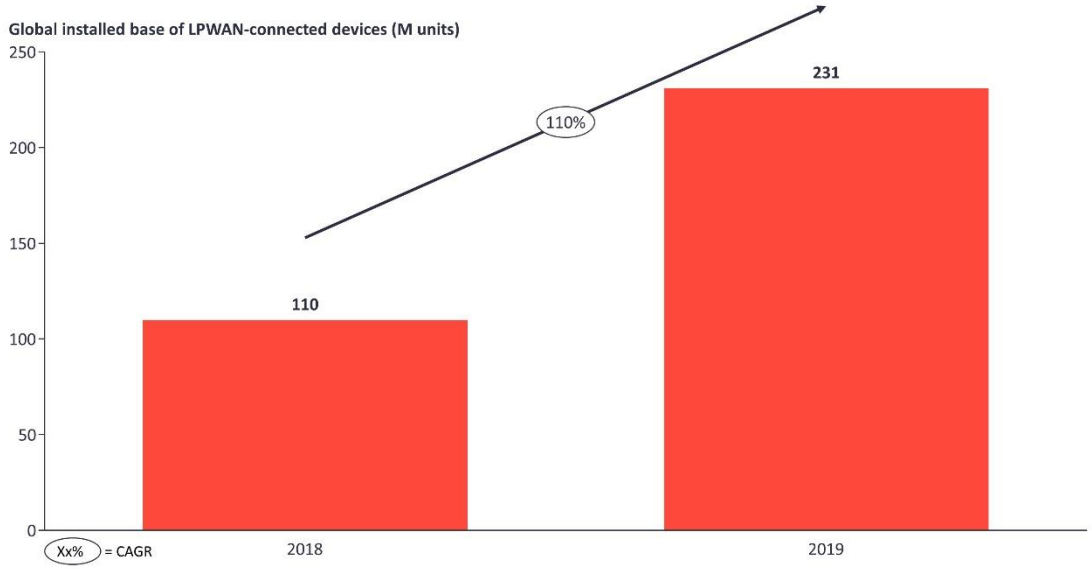
(https://www.businesswire.com/news/home/20190516005700/en/Strategy-Analytics-Internet-Things-Numbers-22-Billion)

Les IoTs continuent d'être de plus en plus valorisés et commencent à avoir des tâches essentielles. Leurs améliorations nécessitent aussi que les appareils soient plus efficaces dans leur fonctionnement. Beaucoup de ces appareils ont des exigences particulières qui peuvent être la longue portée, l'envoi de faible débit de données, la nécessité d'avoir une faible consommation d'énergie. Pour ceux-là, une transmission tel que le Bluetooth par exemple ne peut pas être envisagée. Les technologies de communication cellulaire telles que la 3G ou 4G peuvent fournir cette longue portée, mais en contrepartie, la consommation d'énergie des appareils sera plus grande. Pour contourner ces deux problèmes, une alternative émergente de communication existe :

le Low Power Wide Area Network (LPWAN ou dans sa traduction réseau étendu à basse consommation). Parmi les LPWAN, plusieurs protocoles de communications existent dont le LoRaWAN.

Aujourd'hui, la popularité du LPWAN est en constante augmentation auprès des industries et des chercheurs en raison de sa longue portée et de son faible coût. LoRa est une des technologies qui est la mieux adaptée aux IoTs qui nécessitent la plus grande autonomie et une longue portée pour la transmission d'informations.

Figure 2 : Graphique du nombre global d'appareils LPWAN connectés



(https://iot-analytics.com/5-things-to-know-about-the-lpwan-market-in-2020/)

A travers ce travail, le but est de comprendre le fonctionnement du système LoRa et d'étudier les offres existantes afin de proposer la meilleure solution pour répondre aux critères tels que la sécurité pour sa mise en place et le type de réseau à choisir (privé, collaboratif ou public). Ce document est destiné à servir aux PMEs et aux particuliers afin de pouvoir leur donner les bases pour le déploiement d'un réseau LoRa dans leur propre réseau Internet.

2. L'Internet des Objets - IoT

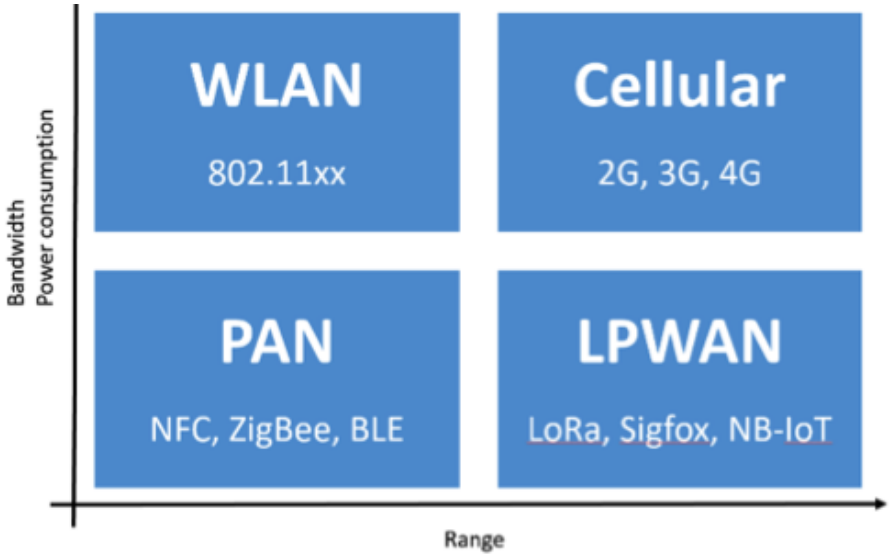
2.1 Définition

Internet of Things (IoT) ou l'Internet des Objets est un terme pour décrire un système d'appareils interagissant par internet. Ces objets connectés peuvent ainsi être utilisés à distance par un réseau de télécommunication. Cela permet notamment de pouvoir récolter des données en temps réel à plusieurs endroits différents sans devoir forcément s'y rendre pour chaque mesure. Ces réseaux de communications peuvent être le réseau des téléphones mobiles, le Wi-Fi ou les réseaux à basses consommations.

2.2 Low Power Wide Area Network

Comme son nom l'indique, il s'agit d'un réseau sans fil à basse consommation. Le LPWAN est un réseau à longue portée et à bas débit, comparé aux réseaux cellulaires où le débit est élevé. Là où le LPWAN se démarque, c'est qu'il permet d'augmenter l'autonomie d'un appareil déployé, qui n'est pas possible avec un réseau cellulaire qui consomme plus d'énergie. Elle convient donc aux appareils ne nécessitant pas d'envoyer de grande quantité de données et à longue portée. Ces appareils peuvent ainsi rester actifs bien plus longtemps. Le LPWAN permet ainsi de réduire les coûts avec des appareils plus optimisés. Comme on peut le voir sur la figure 3 ci-dessous, les technologies comme le Bluetooth ou le NFC ne permet d'avoir une grande portée malgré leur basse consommation.

Figure 3 : Différent types de réseaux sans fil selon leur portée et leur bande passante



(https://home.cern/fr/news/announcement/computing/put-sensor-your-life-new-lorawan-network)

Le Bluetooth à basse consommation (BLE) par exemple permet d'avoir une consommation très réduite comparé au Bluetooth classique permettant de pouvoir atteindre jusqu'à 1 an d'autonomie sur batterie, mais la portée est de 10 mètres. Les LPWAN répondent donc à ce manque de grande portée et de basse consommation. Parmi les technologies LPWAN, trois protocoles se démarquent : NB-IoT, Sigfox et LoRaWAN.

2.2.1 NB-IoT

De son nom complet Narrowband IoT, il s'agit d'un réseau LPWAN développé l'organisation qui s'occupe de la standardisation des réseaux cellulaires[1]. Là où certains réseaux LPWAN utilisent des fréquences radios libres, NB-IoT utilisent les réseaux cellulaires existants comme la 4G. Les réseaux cellulaires étant bien déployé selon les régions, un réseau NB-IoT peut donc posséder une très grande portée pour les appareils à basse consommation.

2.2.2 Sigfox

Sigfox est un opérateur réseau proposant leur réseau du même nom. Par le biais de partenariat avec d'autres entreprises de télécommunication, Sigfox déploie son réseau à travers les pays qui ont adopté la technologie. Le réseau utilise des fréquences radios libres pour la communication en Europe à la fréquence 868 MHz. Il s'agit donc d'un réseau propriétaire auquel il est nécessaire de souscrire à un abonnement pour pouvoir utiliser le réseau.

2.2.3 LoRaWAN

Le réseau LoRaWAN utilise la technologie LoRa pour les communications. Il utilise comme Sigfox une fréquence radio libre qui est la même pour l'Europe et les Etats-Unis. Il s'agit d'une technologie propriétaire développé par Semtech. Néanmoins, un réseau LoRaWAN n'est pas propriétaire et peut être déployé librement comme réseau privé ou public, notamment chez les opérateurs cellulaires pour le réseau public. LoRaWAN peut être déployé partout dans le monde et permet de gérer son propre réseau sans devoir passer des réseaux externes.

[1] 3rd Generation Partnership Project (3GPP)

2.3 La technologie LoRaWAN dans le secteur professionnel

L'utilisation de la technologie LoRaWAN est très variée dans le monde. Elle peut toucher tout autant le secteur du bâtiment, de capteurs sur les lieux publics. Elle peut aussi être utilisée dans le secteur agricole ou voir même dans le contexte de la Covid-19 pour notamment les traqueurs de proximité.

2.3.1 Bâtiment intelligent

Dans le secteur du Smart Building ou bâtiment intelligent, il y a plusieurs cas où des appareils LoRa peuvent être utiles à implémenter. En effet, ces appareils peuvent être utilisés comme des capteurs de présences ou capteurs d'utilisation de porte. Dans le cadre d'un bâtiment universitaire, ces capteurs serviraient par exemple de vérifier si une classe est actuellement occupée et pourrait envoyée l'information à un serveur gérant tous ces appareils. Nous aurons ainsi la possibilité d'avoir une vision globale de l'utilisation de chaque salle. Des capteurs de mouvements peuvent servir à analyser le flux de passage dans le bâtiment, qui pourrait être utile dans le cadre d'une pandémie tel que le Covid-19. Les données récupérées peuvent permettre d'obtenir des statistiques et ainsi établir des solutions en se basant sur ces résultats. Il est aussi possible de surveiller le système d'électricité, d'eau ou d'autres éléments du bâtiment à distance pour déceler d'éventuel dysfonctionnement.

De nombreuses entreprises proposent des solutions LoRa dans le smart building dont Naxoo avec leur système SHERPA.

2.3.2 Agriculture

Dans le domaine de l'agriculture, les appareils LoRa peuvent être utilisés comme capteurs d'environnement. La portée élevée des appareils permet un répartissement plus facile sur un grand terrain tout en gardant la communication avec le serveur LoRaWAN. Les mesures de débits d'eau ou d'irrigation sont de petites données et peuvent donc être transmis par LoRa.

LoRa peut aussi être utilisé dans le traçage de positions des bétails. En effet, des capteurs de positions placés sur des vaches peuvent transmettre les positions à intervalle régulier vers un serveur LoRaWAN. Il devient alors intéressant dans des

régions où les connectivités sont peu présentes et où il y aurait une nécessité de suivi du bétail.

2.3.3 Ville intelligente

Le smart city est aussi un marché en plein essor dans les grandes villes. En effet, ces dernières déploient de plus en plus de capteurs dans la ville. Il existe par exemple des capteurs de détection de véhicule. L'entreprise franco-suisse IEM propose notamment PrestoSense qui est un capteur qui peut être posé au sol des places de parking en ville et ainsi connaître en temps réel leur occupation. Nous pouvons voir en temps réel l'utilisation des parkings publics dans certaines rues équipées de ces appareils. Il y a aussi la possibilité d'utiliser des capteurs de la qualité d'air avec une transmission LoRa.

3. La technologie LoRa

Long Range ou LoRa dans son diminutif est un protocole de liaison sans fil à faible consommation développé par l'entreprise Semtech. LoRa utilise une modulation qui permet une communication à longue portée entre des appareils pourvus de ce protocole pouvant aller à plusieurs kilomètres. LoRa est la communication entre deux nœuds LoRa ou un nœud LoRa et une passerelle. LoRaWAN est l'architecture permettant à des nœuds de communiquer avec un serveur LoRaWAN. Le nœud utilise le protocole LoRa pour le type de modulation. Si nous nous basons sur le modèle OSI, nous pouvons donc observer que LoRa correspond à la couche physique et LoRaWAN à la couche de liaison de données.

LoRa utilise des fréquences libres de transmissions qui sont différentes selon les régions du monde : 868MHz en Europe et 915MHz aux Etats-Unis et 433MHz en Asie. Le type de signal radio émis est l'étalement de spectre sous la forme d'un Compressed High Intensity Radar Pulse (CHIRP). Cette modulation est composée de deux paramètres supplémentaires pour définir la transmission. En premier lieu, il y a la largeur de la bande passante. LoRa utilise des largeurs de bandes passantes à 125kHz, 250kHz et 500kHz. Le second paramètre est le Spreading Factor (SF) ou le facteur d'étalement du spectre. Le paramètre de facteur va de 7 à 12. Un SF élevé permet d'envoyer à plus longue portée, mais en échange le débit envoyé est bien plus faible. Avec un SF élevé, les données envoyés devront alors avoir une taille plus faible et le récepteur aura plus de possibilité de capter le message. Il est donc important de déterminer quel type de SF nous voulons utiliser selon la situation donnée. Dans le cas d'appareils LoRa qui se trouvent à plusieurs kilomètres de distance, il convient par exemple d'utiliser un SF élevé pour éviter des pertes de paquets. Le temps de transmissions est aussi plus grand.

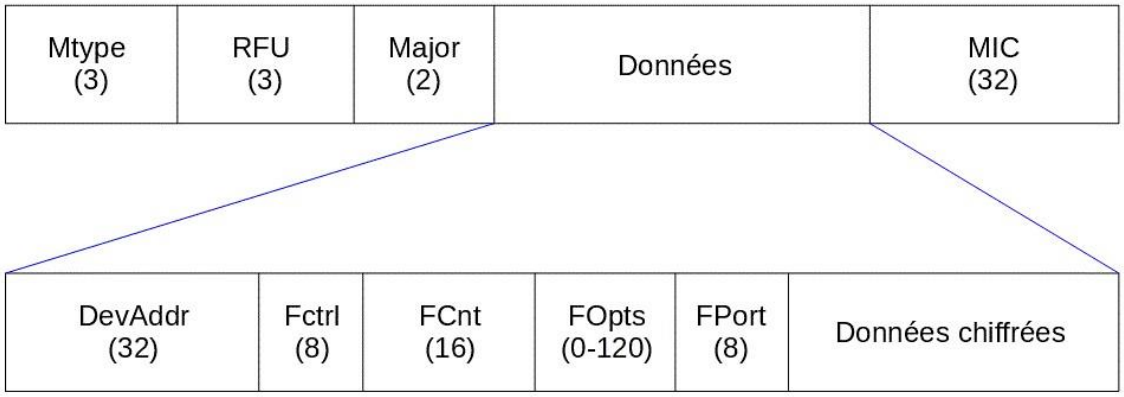
Tableau 1 : Débit physique et Taille du payload pour la bande EU 868MHz

Data Rate	Spreading Factor	Bande passante (kHz)	Taille max du payload (Octets)
0	12	125	51
1	11	125	51
2	10	125	51
3	9	125	115
4	8	125	222
5	7	125	222
6	7	250	222

3.1 LoRaWAN

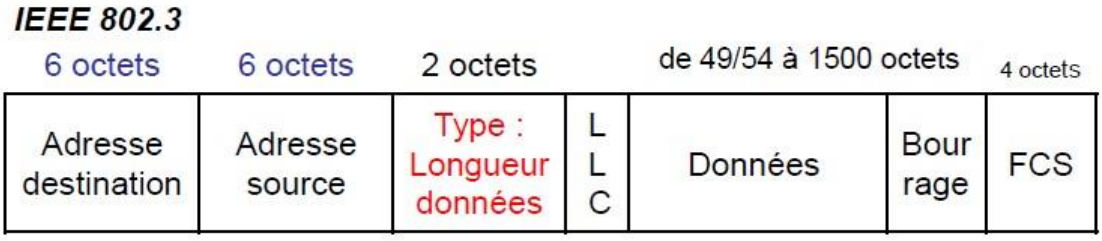
LoRaWAN utilise la couche physique de LoRa. Elle définit la couche de liaison du système de réseau. Chaque trame Lora contient dans son PHY payload la couche MAC LoRaWAN. Il y a deux types de message communiqué en LoRaWAN, les uplinks et les downlink. Un uplink est un message transmis d'un appareil et un downlink est un message envoyé vers l'appareil. Cela montre qu'une communication avec un appareil n'est pas vraiment symétrique.

Figure 4 : Trame LoRaWAN



(Germain Gaudard – Wikipédia)

Figure 5 : Trame Ethernet IEEE 802.3



(HEG)

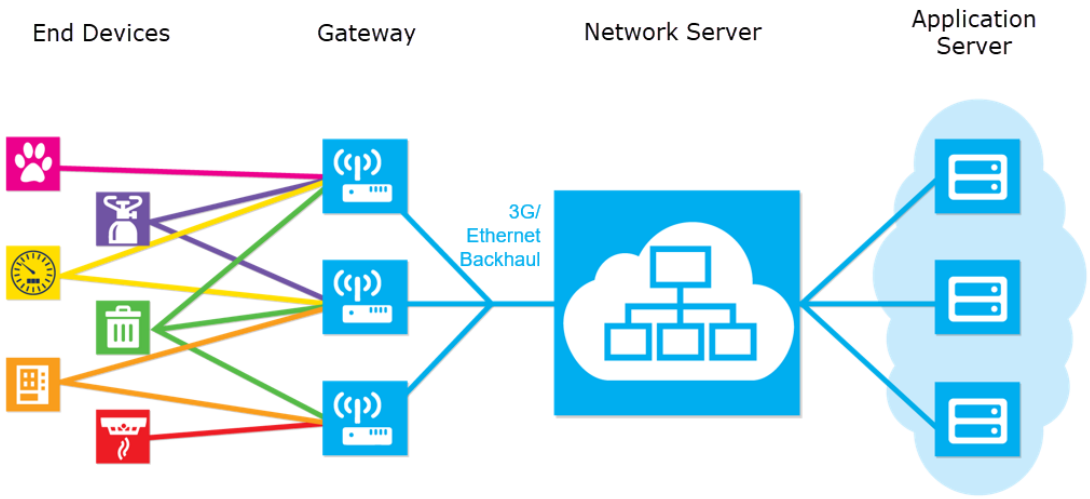
En observant une trame LoRaWAN et une trame Ethernet, on retrouve quelque similitude comme l'Adresse MAC source de l'Ethernet qui est le DevAddr dans une trame LoRaWAN et les données de la trame. LoRaWAN utilise une double encryption dans sa trame avec les données chiffrées ainsi que le MIC (Message Integrity Control). Cela permet de compenser le manque d'adresse de destination de LoRa. En effet, un appareil LoRa transmet des messages en broadcast et n'importe quel autre appareil peut

réceptionner le message. Ce double chiffrement permet d'authentifier le destinaire qui possède les mêmes clés de chiffrement pour déchiffrer la trame LoRaWAN.

3.2 Structure du réseau LoRaWAN

Un réseau LoRaWAN est composé de 4 éléments principaux : les appareils LoRa, les passerelles, le Network Server (serveur réseau) et l'Application Server (serveur applicatif). Le réseau comprend donc de notre appareil IoT jusqu'à notre partie client avec l'Application Server comme l'illustre la Figure 3 ci-dessous :

Figure 6 : Architecture LoRaWAN



(https://tech-journal.semtech.com/understanding-the-lorawan-architecture)

Après l'observation de cette figure, nous remarquons que les appareils Lora ne communiquent pas directement entre eux comme nous pouvions avoir dans des communications simples entre appareils LoRa sous le format sender et receiver. Dans le cas de LoRaWAN, nous avons une topologie plutôt sous forme d'étoile. Dans le cas où nous désirerions plutôt une communication entre les noeuds en plus, nous devrions alors conceptualiser un réseau maillé de l'ensemble, mais cela ajoutera plus de complexité à la mise en place. En effet, dans le cadre d'un réseau maillé LoRa, nous perdrions en vitesse de transfert et les noeuds devront restés actifs plus longtemps ce qui diminuerait leur autonomie sur le long terme. Les noeuds serviraient aussi d'intermédiaire entre eux pour se relayer le message. Dans notre cas, le réseau LoRaWAN nous permettra d'avoir une communication simple des noeuds vers notre serveur en utilisant les protocoles de sécurité fournit par LoRaWAN.

3.2.1 Appareils LoRa

Les End Nodes ou appareils sont les dispositifs que nous désirons mettre en place dans le réseau LoRaWAN. Il peut s'agir de senseur thermique, de système de traçage GPS ou encore des systèmes d'alarme. Ces appareils doivent être pourvu d'antenne LoRa, sans laquelle il leur est impossible de communiquer. Il peut y avoir un ou plusieurs appareils dans un même réseau.

3.2.1.1 Les différentes classes

Il existe trois catégories d'appareils LoRa. Elles dépendent principalement de leur temps d'activité et d'écoute via leur radio LoRa. Le choix des appareils selon la situation nécessaire peut être envisagé en ciblant les classes d'appareils.

3.2.1.1.1 Classe A

Un appareil de classe A est la catégorie principale des devices. En effet, chaque appareil est au moins de classe A pour pouvoir fonctionner en LoRa. Dans le cas de transmission d'informations d'une classe A vers un autre appareil ou passerelle LoRa, le transmetteur peut recevoir un retour du récepteur durant un certain un certain délai. Hors de ce délai, le transmetteur ne recevra pas de retour et les messages retours seront ignorés.

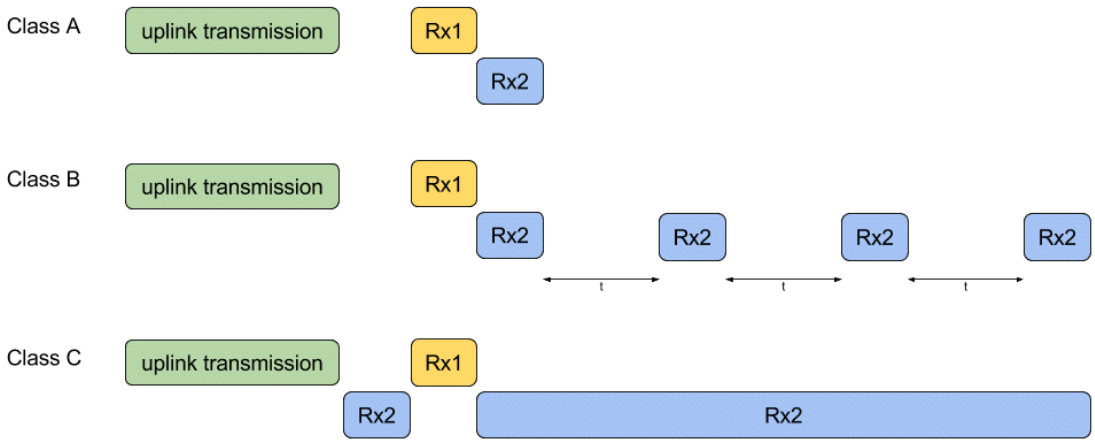
3.2.1.1.2 Classe B

L'appareil de classe B utilise le même principe que la classe A et il ajoute une fonctionnalité en plus qui est la réception périodique de message venant d'une passerelle. Cela fait qu'entre les envois de trames de l'appareils de classe B, ce dernier peut recevoir durant son temps de pause des Downlinks à intervalle régulier.

3.2.1.1.3 Classe C

Un appareil de classe C est en écoute constante sur leur fréquence entre les envois qu'il fait. Cela permet d'envoyer un message en tout temps à l'appareil depuis un serveur. L'appareil en revanche aura une autonomie sur batterie nettement moindre comme il sera constamment actif.

Figure 7 : Classes d'appareils LoRa



(https://witekio.com/lorawan-a-dedicated-iot-network/)

3.2.2 Gateway – Passerelle

La passerelle joue le rôle d'intermédiaire entre le Network Server et les noeuds. Elle s'occupe de transmettre tous les paquets qu'elle reçoit des noeuds vers le Network Server auquel elle est rattachée. La communication entre une passerelle et un Node se fait en LoRa alors qu'entre une passerelle et son Network Server, ça sera une communication par TCP/IP. La liaison peut donc être faite via Wi-Fi, Ethernet ou 4G. Il peut aussi y avoir plusieurs passerelles dans une installation de réseau LoRaWAN ce qui peut permettre d'augmenter sa portée et ainsi pouvoir placer des appareils plus loin.

Une passerelle peut tout aussi bien être un simple microcontrôleur ou un Raspberry avec un module LoRa spécifique. Ces passerelles peuvent alors communiquer via un Uplink et un Downlink s'il s'agit d'un module normalement créé pour un noeud. Des shields LoRaWAN existent pour les Raspberry leur permettant de fonctionner comme une réelle passerelle avec les 8 Uplinks maximum possible d'une passerelle. Il existe aussi des passerelles LoRaWAN proposées par Cisco, Kerlink.

3.2.3 Network Server

Le Network Server peut être vu comme le cœur du réseau LoRaWAN. Il trie les paquets reçus des passerelles en supprimant les doublons dans le cas où d'autres passerelles LoRa du même réseau transfèrent le paquet. Il décode aussi ces paquets avec les clés à sa connaissance des appareils de son réseau. Lorsqu'une trame arrive d'un appareil configuré pour un autre réseau LoRaWAN, ce dernier ne peut être décodé car les clés

d'authentifications ne correspondent pas avec ceux du Network Server. Les noeuds LoRa sont authentifiés par le Network Server grâce à une clé AES 128bits.

Le Network Server peut se trouver dans notre propre réseau interne ou dans le cloud en étant hébergé. Les passerelles doivent connaitre l'adresse IP du Network Server pour transférer le message. De son côté, le Network Server détermine l'adresse IP de ses passerelles après avoir reçu leur message.

3.2.4 Application Server

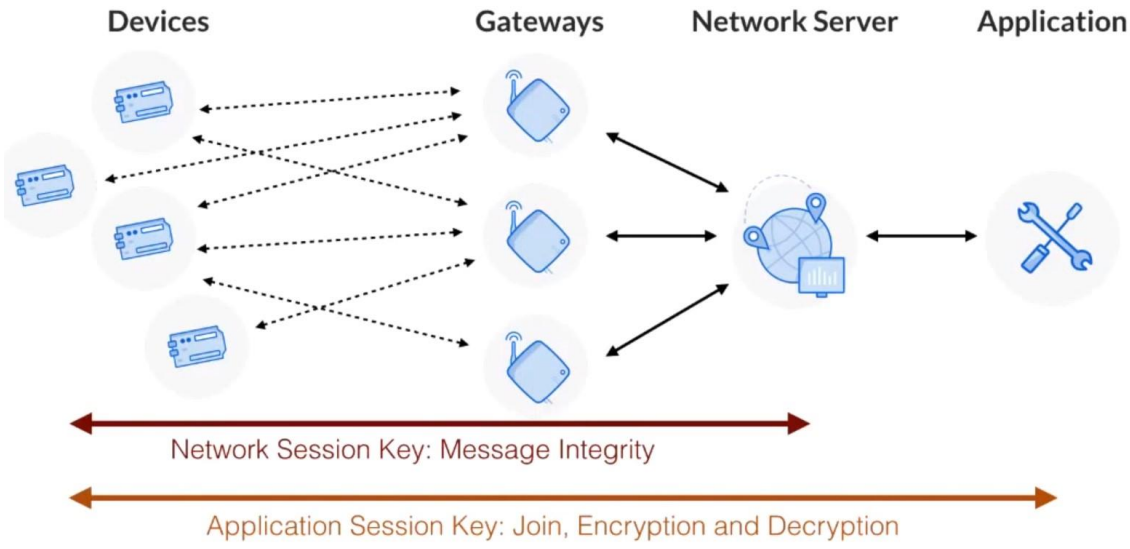
L'application gère les données transmises des noeuds et fait le lien entre le client et le serveur. C'est à travers l'Application Server que l'utilisateur peut interagir avec ses appareils. Il pourra y récolter les données envoyées des appareils ou leur transmettre des commandes

3.3 Sécurité et authentification

3.3.1 Clés de sessions

Pour garantir une sécurité lors des communications entre les appareils et le serveur, il y a deux clés qui sont importantes : le Network Session Key (NwkSKey) et l'Application Session Key (AppSKey). Une communication qui utilise le protocole LoRaWAN est forcément encryptée. Ce Protocole permet grâce à ces clés d'avoir deux couches de sécurité. Il s'agit donc de deux clés de session utilisant l'algorithme AES-128.

Figure 8 : Clés de sessions LoRaWAN



(The Things Network)

3.3.1.1 Network Session Key

Le Network Session Key permet l'interaction entre un serveur et les appareils LoRa en tant qu'authentification. Cette clé permet de valider que le champ Message Integrity Control d'une trame LoRaWAN a bien été généré correctement et que donc le message est valide. Le MIC est généré par un calcul entre le payload chiffré ainsi que le NwkSKey. Elle est créée lors de l'envoi d'un message et lors de sa réception. Cette validation est faite si les MIC générés de l'appareil et du serveur sont les mêmes. Le message est ainsi authentifié et peut être utilisé.

3.3.1.2 Application Session Key

En comparaison avec le NwkSKey, l'AppSKey sert ici de chiffrement des données du payload. L'AppSKey permet l'encodage et le décodage des données transmises entre l'appareil et l'Application Server. Les données sont ainsi chiffrées de bout en bout. Ainsi la combinaison de l'AppSKey et du NwkSKey permet d'avoir un Message Integrity Control plus sécurisé et permet une authentification sûre de la trame.

3.3.1.3 DevAddr

En plus des deux clés de sessions, une adresse d'identification nommée DevAddr doit être attribuée à l'appareil pour être reconnu dans le réseau LoRaWAN. Cette adresse agit comme une adresse IP pour l'appareil dans le réseau pour reconnaitre de et vers quel appareil les trames LoRaWAN transitent. Elle peut être configurée manuellement ou dynamiquement dans le réseau.

3.3.2 Connexions et activations des appareils

Afin de pouvoir communiquer avec ces clés de session. Il est nécessaire d'établir une connexion entre notre appareil et le réseau LoRaWAN auquel nous voudrions le connecter. Il existe deux types de connexions possibles Over The Air Activation (OTAA) et Activation By Personalization (ABP). Ces deux connexions utilisent chacune des données différentes pour établir l'authentification de l'appareil.

3.3.2.1 Activation by personalization

Pour une connexion par ABP, il est nécessaire de connaître les deux clés de sessions NwkSKeym, AppSKey ainsi que le DevAddr. Ces données sont stockées aussi dans l'appareil. Elles peuvent être définies par défaut par les fabricants sur leurs appareils.

3.3.2.2 Over the air activation

De son côté, l'OTAA nécessite d'autres données. Les clés de sessions et le DevAddr seront générés lors de l'établissement de connexion entre l'appareil et le réseau. Pour établir cette connexion, nous devons avoir le DevEUI, l'AppEUI et l'AppKey .

Le DevEUI correspond à l'identifiant d'un appareil LoRa. Il correspondrait alors à un semblant d'adresse MAC. L'AppKey est une clé AES-128 qui permet d'établir le lien avec le réseau. Il doit être connu dans le serveur du réseau ainsi que dans l'appareil. L'AppEUI est une clé supplémentaire qui permet d'identifier le serveur vers lequel notre appareil envoie sa demande de connexion.

Tableau 2 : Méthodes de connexion et activations des appareils

	ABP	**OTAA**
NwkSKey	Configurée sur le serveur et l'appareil	Générée lors de la connexion de l'appareil
AppSKey	Configurée sur le serveur et l'appareil	Générée lors de la connexion de l'appareil
DevAddr	Configurée sur le serveur et l'appareil	Générée lors de la connexion de l'appareil
AppKey	N'est pas utilisée en ABP	Générée ou configurée lors de la connexion de l'appareil
DevEUI	Configurée sur le serveur et l'appareil	Configurée sur le serveur et l'appareil
AppEUI	Configurée sur le serveur et l'appareil	Configurée sur le serveur et l'appareil

3.4 Réseaux LoRaWAN disponibles

Il existe principalement deux types de réseaux LoRa : public et privé. Les réseaux publics sont gérés par des opérateurs télécoms. Dans le cas des réseaux privés, il s'agit d'une installation dans son propre réseau pour une utilisation privée.

3.4.1 Swisscom

Parmi les opérateurs suisses, Swisscom propose un réseau LoRaWAN à leurs clients commerciaux. L'opérateur propose une couverture importante dans le pays pour leur réseau LoRa. Nous pouvons aussi trouver une liste d'appareils professionnels compatibles et utilisés sur leur réseau. Comme il s'agit d'offre en grande majorité Business To Business, les prix ne sont malheureusement pas disponibles. L'infrastructure LoRa de Swisscom est maintenue en collaboration avec la Poste, CFF et NeoVac[2]. Dans le cadre d'un déploiement d'un très grand nombre d'appareils LoRa à plusieurs différents endroits, il peut être envisagé comme une solution pour éviter le coût d'infrastructure d'installation des antennes passerelles et serveurs.

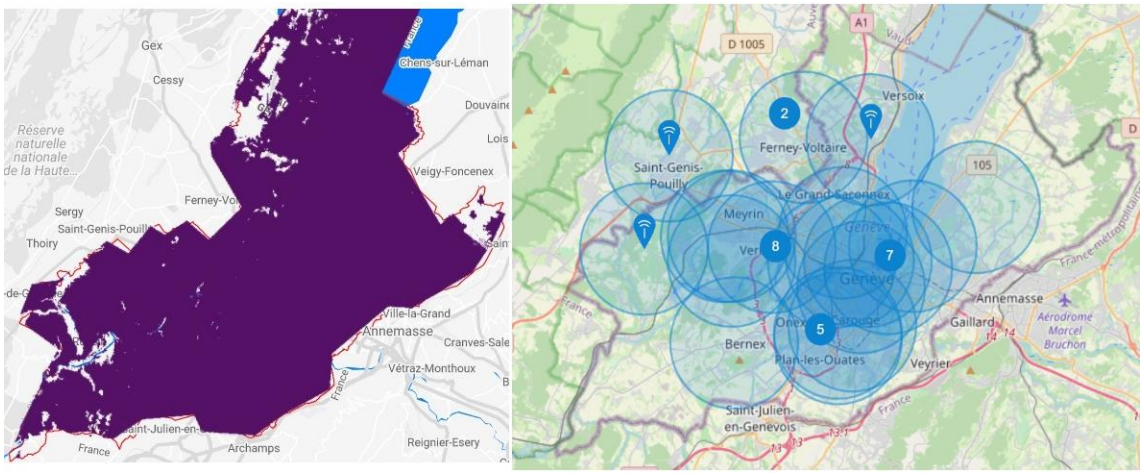
3.4.2 The Things Network

The Things Network ou TTN est un réseau LoRaWAN public. Il s'agit d'un réseau public gratuit proposant un réseau LoRaWAN complet. Sa gratuité réside dans le fait que ce sont ses utilisateurs qui mettent en place des passerelles LoRa. Ce système permet de créer un réseau maillé à très grande échelle dans le monde avec chaque contributeur pouvant proposer une passerelle de son côté. TTN de son côté n'installe néanmoins pas de passerelles. Une grande communauté s'est alors formée autour de ce réseau avec un nombre croissant d'appareils mis en place sur les réseaux TTN. De nombreux guides de mises en place sont présents et le forum communautaire de TTN est très actif. Actuellement le centre-ville de Genève est couvert par des passerelles installées par des particuliers et des entreprises. Une cartographie en temps réel affiche la position des passerelles installées dans le monde dans le cas où les positions des passerelles sont rendues publiques par leur propriétaire.

TTN propose son Network Server dans leur cloud ainsi qu'une page de console pour mettre en place et configurer nos appareils et gateways. Une offre payante existe pour les entreprises leur permettant d'installer le Network Server dans leur propre réseau cloud et la possibilité d'avoir un réseau privé. Ce dernier est sous le nom de The Things Enterprise et est donc orienté comme solution pour les professionnels.

[2] https://www.swisscom.ch/fr/business/enterprise/offre/iot/cmp-lpn.html

Figure 9 : Cartes de la couverture LoRaWAN Swisscom (gauche) et TTN (droite) dans le Canton de Genève



https://www.swisscom.ch/fr/business/enterprise/offre/iot/lpn.html et
https://www.thethingsnetwork.org/community/geneva/

Comme Nous pouvons observer sur la Figure 4 ci-dessus, la couverture réseau est nettement plus importante chez Swisscom. Du côté de TTN, la couverture peut varier grandement selon l'ajout ou la suppression de passerelle sur le territoire.

3.4.3 The Things Stack

The Things Stack (TTS) est la version open source et gratuite de The Thing Industries. Accessible via leur repository Github, il peut être installé sur un serveur local en utilisant Dockers. Il est aussi possible d'installer sa version de développement directement sur le serveur. TTS contient toutes les mêmes fonctionnalités que TTN. TTS est donc un réseau privé.

3.4.4 ChirpStack

ChirpStack anciennement nommé LoRa Server est un serveur LoRaWAN open-source privé. Créé fin 2015 par Orne Brocaar, Il contient également une solution complète pour le déploiement rapide d'un réseau LoRa. Un guide de mise en place basique est disponible sur leur site. ChirpStack propose aussi la possibilité d'installer un système d'exploitation directement sur une passerelle Raspberry pour le transformer en passerelle et serveur à la fois, gérant ainsi tout le fonctionnement.

3.4.5 Avantages et inconvénients

Dans le cas d'une mise en place d'un réseau privé, nous avons un contrôle complet sur nos machines. Nous ne sommes non plus pas limités par les temps d'utilisation du réseau. En effet, TTN impose un temps de uplink maximum par appareil connecté sur leur réseau. ChirpStack ou The Things Stack permet d'éviter ce genre de limite.

La couverture d'un réseau privé est limitée par notre infrastructure installée. Si nous désirons augmenter la portée, il devient nécessaire d'investir dans plus de passerelles, ce qui augmenterait le coût. Ce coût d'infrastructure est d'ailleurs plus important pour le réseau privé. Nous devons effectivement investir dans les passerelles et le serveur selon les besoins nécessaires. Dans le cas d'un réseau public, ce coût peut être limité si nous sommes à portée d'antenne LoRa. Si nous nous trouvons à portée d'une passerelle connectée à TTN, nous aurons alors qu'à financer les appareils LoRa que nous voudrions connecter dessus. Mais il peut y avoir le risque qu'une antenne installée par un particulier soit éteinte ou retirée par son propriétaire du réseau, coupant ainsi notre connexion à TTN. Nous n'aurions néanmoins pas ce problème avec Swisscom.

La mise en place d'un réseau LoRa privé ne nécessite pas un coût d'abonnement que nous aurions avec un opérateur. L'investissement est principalement dans le matériel réseaux. Cette solution est plus intéressante pour un investissement sur le long-terme. Un réseau public peut vite coûter cher comme il y a un prix de base en plus des prix de déploiement par appareils sur le réseau. En effet comme l'infrastructure d'un réseau public d'opérateur est très important, son coût l'est aussi. TTN dépend très fortement de sa communauté pour augmenter la couverture, vu que l'entreprise n'installe pas leurs antennes. Néanmoins, TTN jouit d'une communauté très présente dans certaines des grandes villes, ce qui donne la possibilité de pouvoir exploiter leur réseau selon la position sans devoir investir dans une passerelle LoRa.

Un réseau public ne nécessite pas de maintenance, ce dernier étant géré par le fournisseur. Dans le cas du privé, des connaissances en réseau telles que la mise en place et la maintenance est nécessaire. Il faut pouvoir configurer entièrement le Network serveur par soi-même. Il est alors plus simple de déployer une installation sur un réseau public que privé.

4. Mise en place d'un réseau LoRaWAN

4.1 Choix des installations

Après l'analyse des types de réseaux et solutions disponibles, deux solutions se démarquent : The Things Network et ChirpStack. En effet, la présence du premier sur Internet est très importante et la communauté utilisant TTN de mêmes.

Comme vu précédemment, la mise à disposition possible de passerelles par la communauté de TTN dans Genève nous permet d'avoir un accès gratuit aux serveurs de TTN et d'avoir une portée bien plus importante. Dans le cadre de ce projet, nous excluons la solution Swisscom de qui n'entrerait pas dans le budget du Travail de Bachelor. En effet, le budget étant limité, il convient de trouver des solutions moins onéreuses mais offrant tout autant un système complet.

Pour ChirpStack, la possibilité de créer un réseau privé sans frais supplémentaires est intéressante et peut nous permettre de gérer entièrement la manière que nous désirons que le serveur LoRa fonctionne. La méthode d'installation comparé à The Things Stack est aussi nettement plus simple. ChirpStack peut être directement déployé sur un Raspberry tournant sous Raspbian alors que The Things Stack nécessite de fonctionner avec Docker. Dès lors le choix de ChirpStack est plus intéressant en termes de coût et la maintenance est plus simple à gérer.

Je pense qu'à travers cette recherche il est essentiel de présenter une méthode pour un réseau privé et une pour réseau public. Tous les deux ont leurs avantages et inconvénients et proposent des solutions complètes pour le déploiement d'un réseau sécurisé utilisant le protocole LoRa. L'installation des deux types de réseaux nous permettra de mieux différencier les deux solutions et convenir finalement laquelle est la meilleure pour notre utilisation professionnelle.

4.1.1 Détails de l'installation

Le choix étant fait, il est maintenant nécessaire de démontrer le fonctionnement de ces deux solutions. Pour cela, j'ai opté pour la création de petites stations météos envoyant les données observées sur le serveur LoRaWAN en passant par une passerelle que j'aurai préalablement configurée. Je vais aussi mettre en place un site web afin de stocker les données récupérées du serveur et pouvoir les afficher par la suite sur le site. Ainsi, nous aurons fait une élaboration complète du système et montrer son fonctionnement entier.

4.1.2 Matériels utilisés

La sélection pour les matériels est principalement orientée vers des Raspberry, Arduinos et ESP32 pour leur facilité d'accès et leur prix abordable comparés aux appareils professionnels déjà configurés. N'ayant jamais configuré de microcontrôleurs auparavant, j'ai dû étudier le fonctionnement de ces derniers et comment les mettre en place. Les librairies fournies pour ces derniers sont complètes et permettent un déploiement facilité pour des tâches basiques. Le choix de faire des stations météos est venu de l'idée de pouvoir utiliser au jardin familial de mes parents qui se situent à quelques kilomètres de la maison et de pouvoir récolter la météo au jardin et principalement l'humidité pour savoir s'il est nécessaire de s'y rendre pour arroser les plantes ou de déclencher un système d'arrosoir automatique à distance.

4.1.2.1 Gateway RAK7246 (Concentrator RAK + Raspberry Pi Zero W)

Dans notre cas, l'idéal est une passerelle offrant les 8 uplinks possible et 1 downlink, afin de pouvoir connecter le maximum d'appareils sans pertes de données. Ainsi nous gagnerons en robustesse. Nous pouvons trouver des passerelles vendues par des entreprises tels que Cisco, Kerlink, TTN et Lorix, les prix pouvant varier de 300 à 2000 CHF. TTN propose leur propre routeur pour un prix nettement plus intéressant, néanmoins dans notre cas, nous ne pourrons pas l'utiliser pour ChirpStack et cela nous obligera d'avoir une deuxième passerelle.

Après quelques recherches, il ressort que construire soi-même la passerelle avec un Raspberry et un concentrateur LoRa est la meilleure solution dans notre situation actuelle. Cela permet de limiter les coûts pour la phase de développement et permet d'utiliser des types de serveurs différents selon notre configuration. C'est pourquoi, je me suis intéressé à la passerelle RAK7246 qui est composé d'un Raspberry Pi Zero W avec un concentrateur LoRa de la marque RAK, une entreprise chinoise. La passerelle vient avec un OS basé sur Raspbian déjà configuré pour fonctionner avec TTN et ChirpStack. La configuration pour ces deux réseaux sera détaillée dans leur méthode de mise en place. Elle sera faite via une connexion SSH sur la passerelle depuis un ordinateur portable.

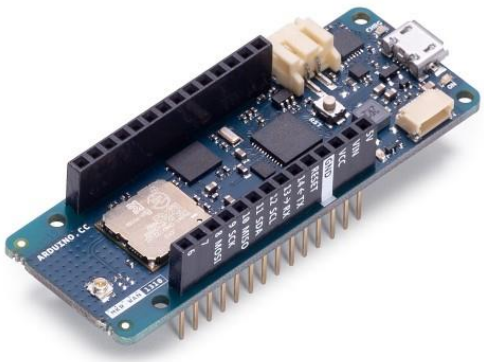
Figure 10 : Gateway RAK7246

(https://store.rakwireless.com/products/rak7246-lpwan-developer-gateway)

4.1.2.2 Arduino MKR WAN 1310

De même que pour les passerelles LoRa, il existe de nombreux appareils professionnels déjà configuré pour fonctionner en LoRaWAN. Il convient alors de concevoir ici aussi nous-mêmes un appareil LoRa à partir de micro-contrôleur pour limiter les coûts. Arduino propose leur propre modèle au nom de MKR WAN 1310. Il s'agit d'un modèle plus compact et plus puissant que l'Arduino Uno et cette version utilise une antenne LoRa et répond aux exigences LoRaWAN. Par ailleurs, Arduino fournit des librairies complètes pour déployer rapidement des prototypes fonctionnels. Arduino propose aussi des Shields pour leur gamme MKR dont nous allons utiliser celui des capteurs d'environnement.

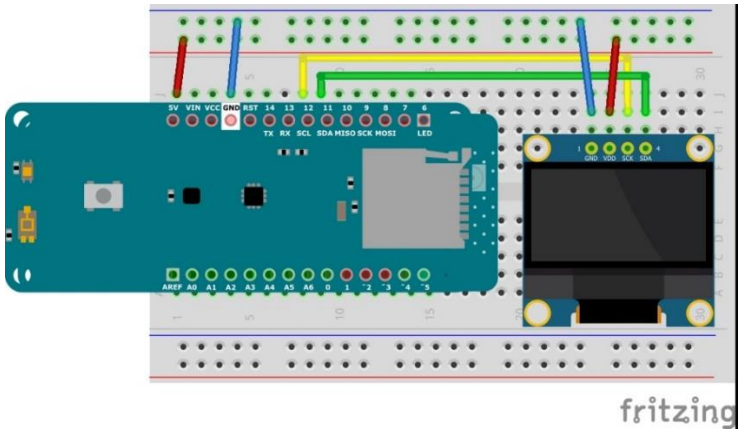
Figure 11 : Arduino MKR WAN 1310



https://store.arduino.cc/mkr-wan-1310

Dans notre cas, notre Arduino sera donc une station météo et intégrera un écran OLED pour afficher les données récoltées en temps réels. Comme nous branchons un écran dessus, il est important de faire le câblage correctement comme ci-dessous :

Figure 12 : Schéma de montage de l'Arduino

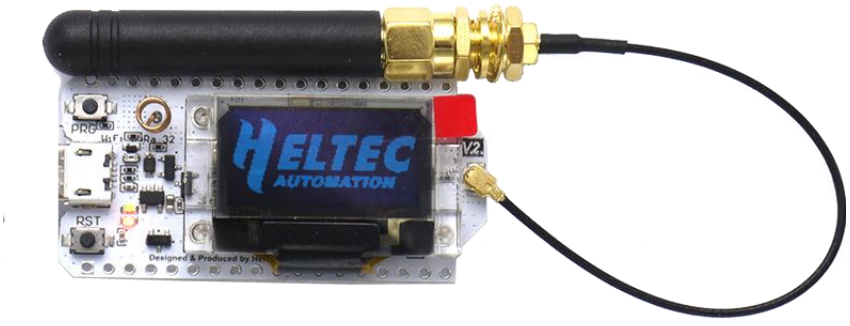


Créé sur Fritzing

4.1.2.3 Heltec LoRa 32

Les ESP32 proposent des microcontrôleurs avec des spécificités nettement plus intéressantes que les Arduino. Ils ont aussi l'avantage d'avoir un prix moindre qu'un Arduino. Dans notre cas, nous allons utiliser un Heltec LoRa 32 V2, il s'agit d'un ESP32 composé d'une antenne LoRa, WiFi ainsi qu'un écran OLED intégré. Cela nous permettra de démontrer que les réseaux LoRaWAN mis en place fonctionne avec n'importe quel modèle.

Figure 13 : Heltec LoRa 32 V2



(https://heltec.org/project/wifi-lora-32/)

4.1.3 Type de connexions des appareils au réseau

Pour la méthode de connexions de notre Arduino et de notre ESP32, nous utiliserons la méthode OTAA. En effet, les clés de sessions étant générées aléatoirement à chaque connexion de l'appareils, il y a moins de données à devoir stocker sur nos appareils. Par ailleurs avec l'OTAA, il est plus facile de changer de réseau pour passer de TTN à ChirpStack, facilitant ainsi le déploiement plus rapide des appareils. Les compteurs de trames sont pris en charge directement en OTAA redémarrant à zéro à chaque connexion, alors qu'en ABP il est nécessaire de le supprimer manuellement à chaque nouvelle connexion.

4.1.4 Programmation

Pour programmer nous utiliserons principalement l'IDE Arduino pour les microcontrôleurs. Le langage utilisé pour les microcontrôleurs sera donc en C. Le site web sera en React JS avec un backend Express JS (Node.js).

4.1.4.1 Site web de démonstration

L'interface web servira principalement de preuve du fonctionnement globale du réseau LoRaWAN. Un backend sera configuré pour récupérer les données du broker de l'Application Server et les stocker dans une base de données MySQL. Le frontend sera mis en place pour afficher ces données stockées dans la base de données et ainsi consulter les données que les appareils auront envoyées sur notre réseau LoRaWAN.

4.1.4.2 Broker MQTT et API

Il convient de signaler que dans cette démonstration, nous allons utiliser le protocole MQTT pour récupérer les données des serveurs. TTN et ChirpStack proposent de nombreuses intégrations possibles avec d'autres outils professionnelles tel que AWS IoT ou myDevices. ChirpStack propose une interface API depuis la console permettant de récupérer les données via des commandes REST aussi.

4.2 Infrastructure des réseaux

Dans la volonté d'être le plus mobile possible afin de pouvoir tester le fonctionnement tant à l'école qu'à domicile, pour une connexion stable entre nos appareils LoRa et le serveur, la passerelle sera placée au balcon de mon domicile afin d'avoir une portée suffisante jusqu'à l'école. La hauteur à laquelle l'appartement se situe permet d'avoir une portée plus élevée. Le réseau utilisé sera donc celui du domicile.

Figure 14 : Schéma d'infrastructure du réseau TTN

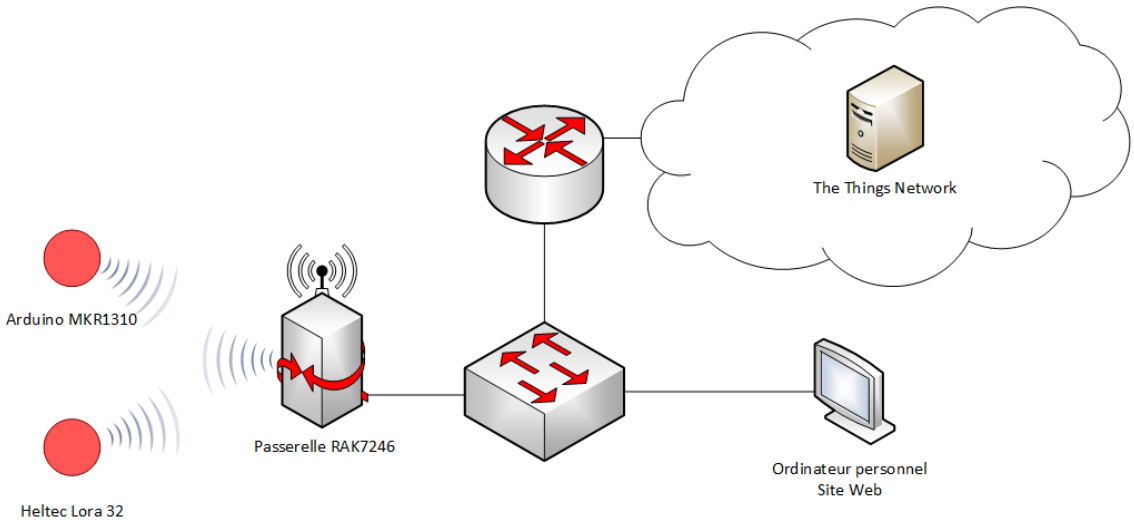
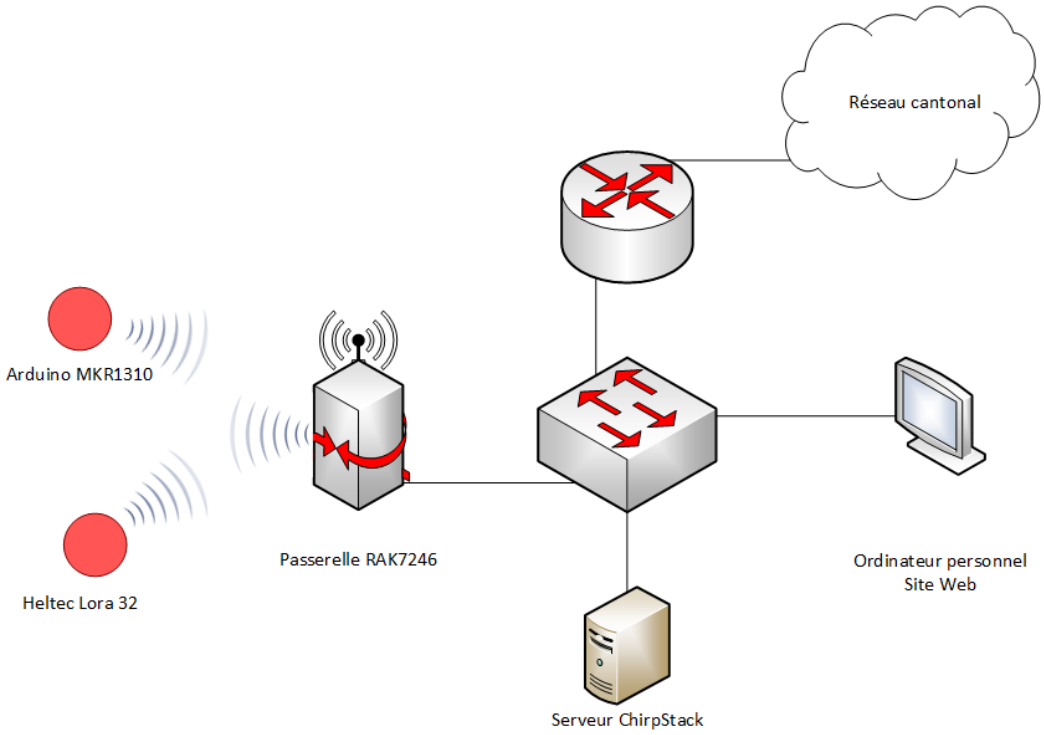


Figure 15 : Schéma d'infrastructure du réseau ChirpStack



A travers ces deux diagrammes, nous avons une vision globale de l'installation complète de nos deux réseaux. Nous allons maintenant voir comment les mettre en place.

4.3 The Things Network (réseau public)

Pour pouvoir utiliser The Things Network, il est nécessaire de créer un compte sur leur site web https://www.thethingsnetwork.org/ . Après la création d'un compte nous obtenons l'accès à la page de console où nous pouvons gérer nos passerelles et appareils LoRa.

4.3.1 Implémentation du réseau

4.3.1.1 Mise en place de la passerelle

Afin de mettre en place la passerelle RAK7246, il est d'abord nécessaire de la configurer de tel sorte que celle-ci puisse communiquer avec le serveur de TTN avant de pouvoir l'enregistrer sur ce même.

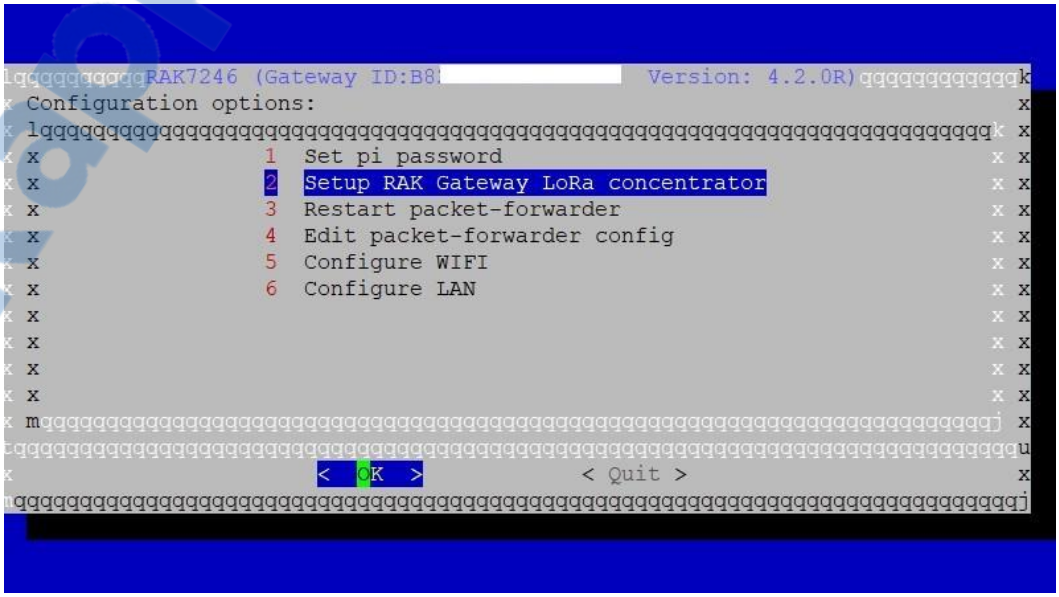
4.3.1.2 Configuration de la passerelle

Nous nous connectons sur notre passerelle Raspberry dans son mode de configuration gateway avec la commande

sudo gateway-config

Nous arrivons ainsi sur la fenêtre comme dans la Figure ci-dessous :

Figure 16 : Fenêtre de configuration de la passerelle

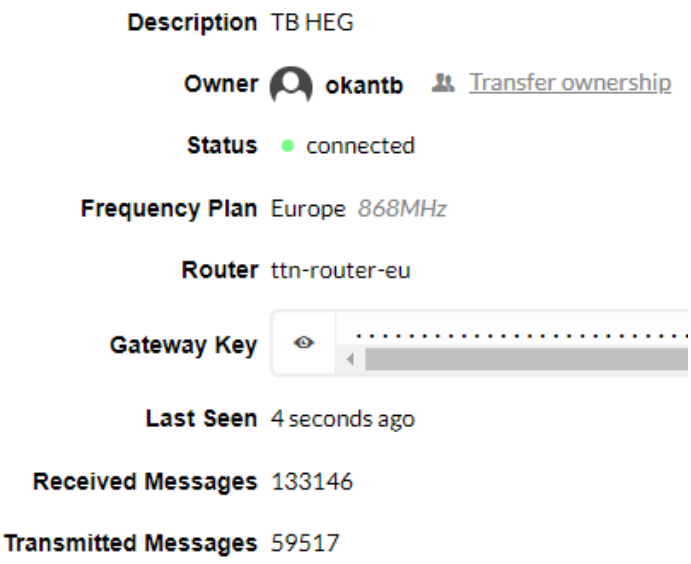


Il faut ici sélectionner *Setup Rak Gateway LoRa concentrator*, puis *Server is TTN*. Ensuite, comme nous sommes en Europe, nous sélectionnons *EU_863_870*. Enfin, nous validons les pages suivantes. Le Raspberry redémarre alors la passerelle LoRa et nous pouvons maintenant l'enregistrer sur TTN.

4.3.1.3 Enregistrement de la passerelle sur TTN

Pour l'enregistrement de notre passerelle sur TTN, il faut cliquer sur Gateway sur notre page de console TTN et cliquer sur le bouton *register gateway*. Sur cette nouvelle page, nous rentrons les données nécessaires. Pour le Gateway ID, il faut retranscrire la valeur que nous avons sur la fenêtre de configuration de notre passerelle tout en haut (voir Figure 9). Dans d'autres cas, elle peut aussi être écrite le document fourni de la passerelle ou bien être collée sur le boitier de la passerelle. Cet identifiant est propre au gateway. Il faut aussi cocher *I'm using the legacy packet forwarder*. Vous pouvez donner la description que vous désirez. Dans Frequency Plan, le choix doit être selon notre région, dans notre cas l'Europe. Vous pouvez aussi définir la location de la passerelle. En cliquant ensuite sur Register Gateway, notre passerelle est alors configurée. Pour confirmer l'enregistrement, nous trouverons sur la page de la passerelle le mot «connected » avec une icône verte à côté montrant que notre passerelle communique avec TTN.

Figure 17 : Status de la passerelle sur TTN



4.3.1.4 Appareils LoRa

Dans TTN, les appareils LoRa doivent être configurés dans des «Applications». Ces derniers servent d'identifiants intermédiaire des appareils pour les séparer en type d'applications. Dans notre cas, nous séparerons notre projet en deux applications : une pour les arduinos et le second pour l'Heltec Esp32.

Chaque application peut contenir un décodeur et un encodeur de payload. Cela permet notamment lors de la réception d'un payload en hexadécimal de le convertir au format désiré. Ainsi, nous pouvons convertir les données reçues au format JSON et avoir un traitement plus facile du côté serveur par la suite.

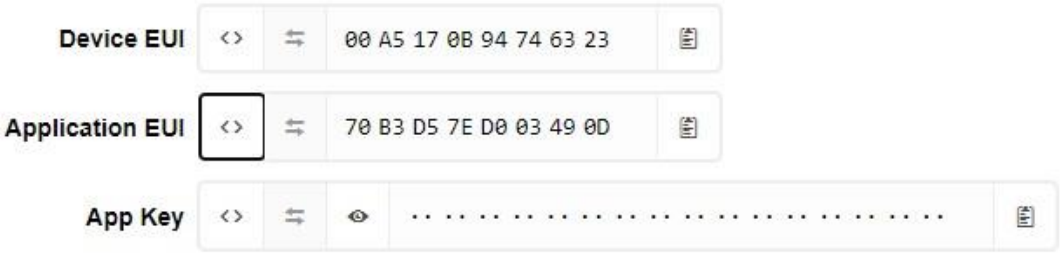
L'appareil doit être au préalable déjà prêt à son déploiement. Dans l'Annexe 3, nous trouverons une pré-configuration pour l'Arduino que nous utiliserons comme exemple ici. Nous procéderons à une connexion en OTAA. Pour cette connexion nous avons besoin de trois données : le devEUI de l'appareil, l'App EUI et l'App KEY fournit par TTN. Pour obtenir le devEUI de l'Arduino, il est nécessaire d'utiliser un des exemples de librairies fourni pour le MKRWAN. Cet exemple affichera dans le moniteur série de l'arduino le devEUI que nous devrons récupérer et le reporter sur TTN. Il est possible que nous devions générer nous même le devEUI, ce qui sera effectivement le cas un ESP32.

4.3.1.5 Connexion d'un appareil Lora

Pour connecter notre appareil LoRa au réseau TTN, il nous faut nous rendre sur la page de console et choisir cette fois Applications. Sur la page de création de l'application, il est nécessaire de définir un ID. Cet ID sert d'identification par le broker et permet de classer aussi nos appareils. Chaque application peut avoir son propre rôle. Pour l'application arduino, nous donnerons l'id « arduinotb ». Après avoir donné un id et une description, nous pouvons valider la création en cliquant sur Add application. Il est maintenant possible d'enregistrer nos appareils sur l'application.

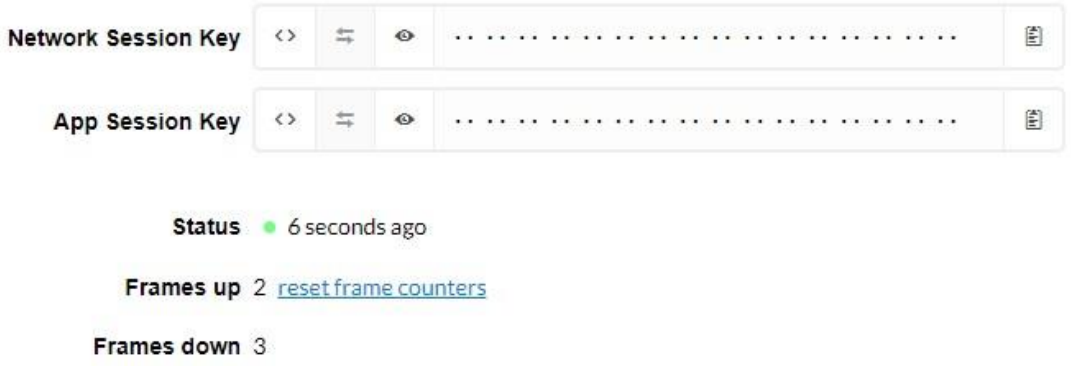
Sur la page d'application créée, il faut cliquer sur register device. Nous rentrons alors un identifiant comme pour l'application et le Device EUI que nous avons récupéré de l'appareil. Une clé d'association sera alors créée par TTN que nous devrons reporter dans la configuration de notre appareil. Maintenant que nous avons enregistré l'appareil, nous récupérons les clés App Key et App EUI de TTN sur la page de résumé de l'appareil. Ces deux clés seront alors ajoutées dans la configuration de l'appareils.

Figure 18 : Clés de connexion nécessaires en OTAA



Si les données entrées sont correctes sur TTN et sur l'appareil, une connexion sera alors établie entre eux. Elle peut être vérifiée dans la page de l'appareil dans la console comme sur la figure ci-dessous :

Figure 19 : Connexion établie d'un appareil à TTN



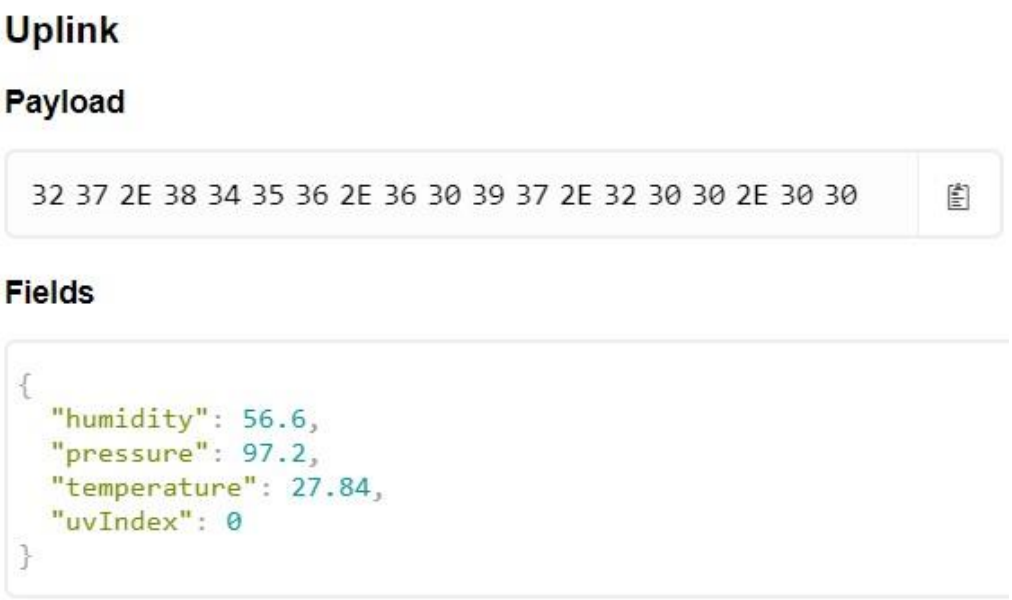
Dans le cas du Heltec LoRa 32, la démarche sera identique hormis pour le device EUI qui devra être généré par TTN, car le Heltec n'en a pas un propre à lui. Il faudra alors aussi reporter le device EUI créé sur TTN dans la configuration du Heltec.

4.3.1.6 Décodage du payload

Dans l'onglet Data, nous pouvons voir les données de la trame. Les données envoyées de l'appareils sont ici déchiffrées par le serveur et affichées en hexadécimal. Il est possible d'afficher ces données hexadécimales au format que nous désirerions. Il faut alors retourner sur la page de l'application « arduinotb » dans l'onglet Payload Formats. Nous pouvons ici configurer un décodeur de payload dans le cas d'une réception de paquet ou d'un convertisseur dans le cas d'un envoi de paquet. Il s'agit ici du décodeur qui nous intéresse pour convertir les données dans un format facilement réutilisable par

notre application futur (le site web). La fonction pour convertir le payload au format voulu se trouve dans l'Annexe 3 dans la partie Décodage. Nous utiliserons ici un envoi de données au format JSON.

Figure 20 : Payload décodé sur TTN



Il est à noter que lors du décodage du payload, les zéros inutiles après la virgule n'apparaissent pas au format JSON. Nous remarquons cela avec l'humidité où la valeur retournée est 56,6 alors qu'au format non décodé nous avons 56,60 (35 36 2E 36 30). Un autre changement est l'ordre d'affichage au format JSON. En effet, les données affichées sont dans l'ordre alphabétique, ce qui peut perturber en comparant le payload au field. Dans le payload, la première donnée est celle de la température, alors quand le field il s'agit de l'humidité.

4.4 ChirpStack (réseau privé)

4.4.1 L'installation

Nous allons maintenant décrire les étapes d'installations d'un réseau privé LoRaWan avec ChirpStack qui est un server réseau open-source.

Pour cela, il faut se rendre sur le site web de ChirpStack https://www.chirpstack.io/ . Un guide d'installation est disponible sur le site avec les étapes importantes nécessaires que nous allons détailler ici. Comme nous avons un Raspberry Pi Zero W, il nous est possible d'installer ChirpStack de deux manières possibles :

- Le concentrateur RAK7246 contiendra le Gateway Bridge, le Network Server et l'Application Server grâce au Gateway OS proposé par ChirpStack avec toute la configuration. La Raspberry est alors directement prêt à l'emploi. Cela limite néanmoins l'ajout ultérieur d'autres passerelles sur le même serveur.

- Le RAK7246 fonctionne en tant que forwarder en envoyant les trames sur un serveur séparé tournant sous Debian ou Ubuntu. Ce dernier jouera alors le rôle de Network Server et Application Server. Le GateWay Bridge peut soit être installé sur la passerelle soit sur le serveur.

Dans notre cas, nous allons opter pour le second cas et utiliser le gateway Raspberry en tant que simple forwarder et avoir un serveur dédié à la réception de ces paquets. Cela peut aussi permettre par la suite d'intégrer d'autres gateways au serveur plus facilement. Il est à noter qu'il est possible d'installer le Gateway Bridge directement sur votre propre passerelle LoRa, mais par mesure de simplicité pour passer de TTN à ChirpStack et vice-versa, il est préférable de garder le même système d'exploitation fourni par RAK.

Pour débuter, il est important d'installer premièrement les dépendances nécessaires au fonctionnement de ChirpStack. En effet, ce dernier nécessite Mosquitto (Broker MQTT), Redis (Système de gestion de base de données en cache) et PostgreSQL (Base de données). Nous allons utiliser Ubuntu tout au long pour installer les composants requis et les modifications nécessaires des fichiers.

Comme précisé avant, ChirpStack est constitué de trois programme à utiliser que nous devons au préalable installer et configurer. Ces trois installations se feront sur notre machine virtuelle Ubuntu 20.04. L'installation permettra de démarrer le serveur LoRaWan à chaque démarrage de la VM. Le Gateway Bridge joue le rôle passerelle de la trame LoRa à la trame UDP vers le Network Server.

Pour installer Mosquitto, Redis et PostgreSQL :

```
sudo apt install mosquitto mosquitto-clients redis-server redis-tools postgresql
```

Il est ensuite nécessaire paramétrer le repository de ChirpStack. La clé utilisée ci-dessous permet d'accéder à son repository d'installation.

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

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00

sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable main" | sudo tee /etc/apt/sources.list.d/chirpstack.list

sudo apt update
```

Nous pouvons maintenant installer Gateway Bridge sur le serveur. La commande ci-dessous avec start permet de démarrer le composant et la commande enable permet son exécution à chaque démarrage du serveurChirpStack fournit une aide d'installation selon le modèle que vous possédez. Dans le cas du Rak7246 il y a déjà une pré-configuration des fréquences pour fonctionner

```
sudo apt install chirpstack-gateway-bridge

sudo systemctl start chirpstack-gateway-bridge

sudo systemctl enable chirpstack-gateway-bridge
```

Avant de configurer le Network Server et l'Application Server, nous devons créer leurs bases de données ainsi que leur utilisateurs et mots de passes. Il est important que vous configuriez vos propres utilisateurs et mots de passe pour les deux bases de données.

L'installation du Network Server est de la même manière que celui du Gateway Bridge :

```
sudo apt install chirpstack-network-server
```

Avant de le démarrer, il est maintenant nécessaire de modifier le fichier de configuration du Network Server pour ajouter notre utilisateur et son mot de passe. Ce fichier se situe au chemin

/etc/chirpstack-application-server/chirpstack-application-server.toml

À la ligne dsn=''… nous le modifions avec nos identifiants créer précédemment.

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

```
sudo systemctl start chirpstack-network-server
sudo systemctl enable chirpstack-network-server
```

Pour installer l'Application Server, il faut entrer la commande suivante :

```
sudo apt install chirpstack-application-server
```

Comme pour le network serveur, il est obligatoire de modifier le fichier de configuration de l'AS qui se situe à */etc/chirpstack-application-server/chirpstack-application-server.toml* .

À la ligne dsn=''… nous le modifions avec nos identifiants créés précédemment.

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

Et à la ligne *jwt_secret* dans *[application_server.external_api]* nous ajoutons une clé secrète que nous générons dans le terminal avec la commande

```
openssl rand -base64 32
```

Nous pouvons maintenant démarrer l'application server :

```
sudo systemctl start chirpstack-application-server
sudo systemctl enable chirpstack-application-server
```

Le serveur est maintenant opérationnel, mais un redémarrage du serveur est conseillé.

4.4.2 Configuration des appareils

Nous pouvons dès à présent utiliser ChirpStack en local en accèdant via l'adresse localhost:8080 depuis le serveur ou via l'adresse ip du serveur IP_SERVEUR:8080 .

Le nom d'utilisateur et le mot de passe par défaut est 'admin' et 'admin'. Il est fortement recommandé de les changer en allant dans l'onglet *All users*.

Avant de pouvoir configurer notre passerelle, on doit créer un serveur réseau dans l'onglet Network-servers, un Service-profile et un Device-profile :

- Pour le Network server
    - Ajouter un nouveau profil en cliquant sur add
    - Dans l'onglet General, mettre le nom que vous voulez dans name et ajouter l'adresse 127.0.0.1:8080
- Service-profiles
    - Créer un nouveau profil
    - Lui donner un nom et cliquer sur Create en bas
- Device-profiles
    - Créer un nouveau Device-profiles
    - Dans l'onglet General, choisir la version LoRaWAN MAC 1.0.2 , A pour le paramètre de révision régional et 30 pour Max EIRP. Le choix du LoRaWAN MAC dépend de la comptabilité des appareils utilisés. Dans notre installation, ils sont tous comptatibles avec la version 1.0.2.
    - Dans l'onglet Join (OTAA/ABP), cocher Device supports OTAA
    - Cliquer sur le bouton Create.

Maintenant que le minimum nécessaire a été configuré pour ajouter notre gateway et notre arduino, nous allons procéder à leur mise en place. Nous devons aussi reconfigurer la passerelle RAK pour la faire fonctionner avec ChirpStack

- Configuration du RAK7246
    - Sélectionner le 2 (Setup RAK Gateway), puis Server is Other Server. Pour la fréquence, sélectionner EU_863_870) et finalement sur la dernière page entrer l'adresse IP du serveur vers lequel transférer les trames.
- Gateway
    - Créer une nouvelle gateway et donner un nom et une description à votre gateway.
    - Entrer le Gateway ID qui se trouve sur la page d'accueil de la page de configuration de la passerelle pour le RAK7246. Cet ID peut aussi se trouver sur le boitier de l'appareil ou il peut être générer directement sur

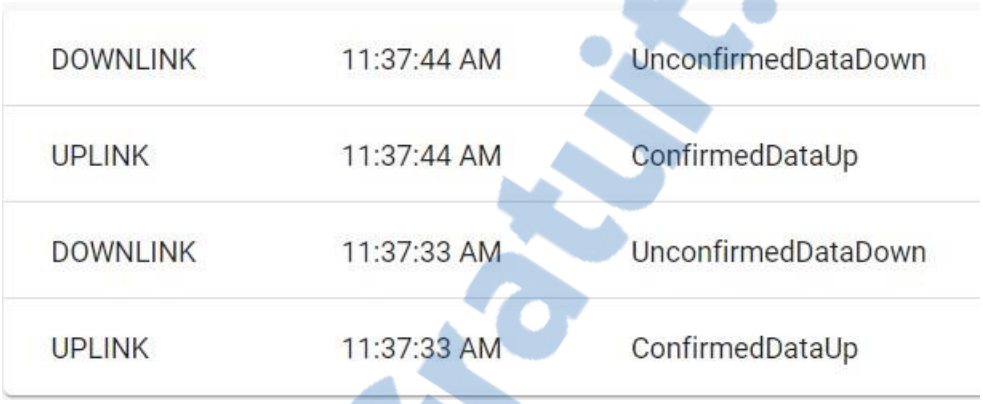
ChirpStack. Ce dernier devrait être alors reporté sur la configuration de la passerelle.

- Vous pouvez lui donner une position pour ainsi situer leur emplacement sur la carte.
- Cliquer sur Create Gateway pour confirmer.

- Applications
  - Créer une nouvelle application
  - Donner un nom et une description et choisir le service-profile créé précédemment et cliquer sur Create pour confirmer la création.
- Devices
  - Maintenant qu'une application a été créée, cliquer sur cette application et créer un nouvel appareil
  - Donner un nom et une description à l'appareil et entrer le Device EUI de l'appareil (Arduino) ou le générer pour l'entrer dans la configuration du microcontrôleur (Heltec). Choisir le device-profile créé précédemment.
  - Cliquer sur Create pour confirmer la création.
  - Maintenant que le device a été créé, cliquer dessus pour générer les clés de connexion.
    - Dans le cas d'une connection OTAA, aller dans l'onglet Keys (OTAA) et générer une Application key et une Gen Application Key et enregistrer la modification. Reporter l'Application key sur votre appareil pour permettre la connection OTAA.
    - Pour une connexion ABP, aller dans l'onglet Activation et générer le Device address, le Network session key et l'Application session key. Reporter ces 3 données sur votre appareil pour les connecter en ABP.

Arrivé ici, notre gateway et notre appareil lora sont fonctionnels. Vous pouvez consulter l'état des deux appareils sur leur page respectives où vous pouvez retrouver l'heure de leur dernier message sur ChirpStack. Lors d'une connexion entre un device et la gateway, nous pouvons constater que les messages du device dans son live data frame apparait comme un « ConfirmedDataUp ».

Figure 21 : Uplinks et Downlinks d'un appareil sur ChirpStack



Pour accéder aux données de la trame, nous les trouverons dans l'onglet Live Data de l'appareil. Ce dernier contient les éléments d'identifications de l'appareil, la fréquence et la modulation à lesquelles le message a été envoyé et contient les données encodées en base64. Ces données peuvent être décodées directement depuis ChirpStack en codant l'algorithme dans le device-profiles dans l'onglet Codec ou directement depuis notre application personnelle. Si nous décodons depuis ChirpStack, le payload décrypté se trouvera dans « objectJSON », on aura ainsi les deux données lors de la récupération du message de l'appareil sur notre serveur. Le décodage est fait avec la même fonction utilisée dans TTN.

Figure 22 : Payload décodé d'une trame sur ChirpStack

4.5 Implémentation du site web

4.5.1 Backend - Express JS

4.5.1.1 MQTT

Les Brokers MQTT sont déjà implémentés dans le serveur de TTN et dans notre serveur privé ChirpStack. Il nous faut donc implémenter dans notre backend des clients MQTT pour nous connecter sur ces brokers.

TTN fournit un SDK pour une connexion sur le MQTT avec node.js que nous utiliserons ici. Une documentation est fournie sur le site de TTN pour la mise en place de l'API.

En faisant le lien de notre client MQTT vers le broker, notre backend gérera alors la communication entre le backend et le broker. Une instanciation de la classe client MQTT sera lancé au démarrage du backend. Il sera alors en écoute constante du broker pour récupérer les derniers messages des brokers. Ces données reçues seront alors stockées dans la base de données que nous aurons configuré au préalable dans MySQL.

Figure 23 : Uplinks reçus de TTN

```
Received uplink from  arduinomkrenv
{ humidity: 50.41, pressure: 97.19, temperature: 24.67, uvIndex: 0 }
Record inserted:  110
Received uplink from  helteclora32
{ humidity: 48, pressure: 66, temperature: 29, uvIndex: 0 }
Record inserted:  51
```

Une deuxième instanciation de client MQTT est également mise en place pour ChirpStack permettant de récupérer les uplinks de notre serveur ChirpStack. Nous utiliseront ici le framework MQTT.js comme client MQTT. Nous aurons ainsi dans un même programme une connexion à TTN et ChirpStack et pouvoir récupérer en simultané les données publiées.

4.5.1.2 Modèles et Base de données MySQL

La base de données contient des tables simples pour les arduinos et les esp32. Une table pour chaque type d'appareils sera créée. Nous aurons aussi une table différente pour le réseau TTN et le réseau ChirpStack, ce qui nous fera quatre tables au total.

Chaque table sera composée d'un ID, du nom de l'appareil, des données météorologique et la date de réception. A chaque publication sur le broker, si elle correspond à un envoi de données d'un appareil LoRa, les données seront stockées alors dans la base de données.

4.5.1.3 L'implémentation

Notre backend retourne les données enregistrées par nos instanciations de nos clients MQTT. Des routes d'accès ont été mises en place pour lire les données enregistrées dans notre base de données et le renvoyé par un API au format JSON. Nous pouvons accéder ainsi aux données récoltées et stockées dans la base de données.

Figure 24 : Exemple de JSON envoyé du Backend

4.5.2 Interface web

Au niveau de l'implémentation du site web, il y a premièrement une page d'accueil et une page expliquant le but du site. Nous avons ensuite 3 pages accessible svia la barre de menu. Dans l'ordre des boutons de la barre, une page pour afficher la liste des données de l'arduino sur TTN, puis celui de l'esp32 sur TTN et finalement l'arduino sur ChirpStack. Les données récoltées sur les appareils sont affichées de manière simple sous la forme de liste. Nous avons ainsi une démonstration complète d'accès aux données récoltés sur un appareil qui ont été stockées sur une base de données et récupérerables pour être affichées ultérieurement.

Figure 25 : Affichage de la liste des données de l'arduino sur TTN

4.6 Observations

4.6.1 Maintenance et améliorations des serveurs

Les serveurs de TTN n'étant pas sur notre propre réseau, c'est l'entreprise du même nom qui gère de son côté les mises à jour et autres améliorations apportées. En observant les Git de TTN, nous pouvons voir que le projet est actuellement scindé en deux repository différent. II y a la version V2 sous le nom de The Things Network Stack et la nouvelle version V3 au nom de The Things Stack. La version gratuite proposée pour la communauté est la V2 alors que pour la version payante il s'agit de la V3. Le seul moyen d'accéder gratuitement à la V3 actuellement est d'installer The Things Stack comme réseau privé. Il y a donc un doute sur la durée de vie que propose TTN pour son réseau public. Les Git de TTN sont maintenus par une équipe de développement

L'équipe de développement de TTN est entièrement focalisée sur le développement de la version 3 dont la date n'a pas été annoncée ni si cette version allait être apportée à la Community de TTN.

Lorsque nous observons les commits effectués sur les repositories de ChirpStack et TTN V2, nous pouvons voir qu'il y a des modifications régulières sur ChirpStack par son développeur Brocaar. Pour TTN, la V2 reçoit rarement des mises à jour mineur alors que la V3 est en plein développement des ajouts réguliers sur son repository.

Un autre problème se situe sur la récupération des données via MQTT avec TTN avec cette nouvelle version V3. TTN propose des SDK pour récupérer facilement les données transmises par nos appareils avec MQTT sur d'autres langages tel Node.js, Go ou Node-Red. Mais ces SDK ne sont plus mis à jour par TTN, les rendant ainsi obsolètes sur le long terme. Ils fonctionnent toujours par la V2 de TTN mais pas pour la V3. TTN propose alors des alternatives pour pallier cela. Nous pouvons certes toujours récupérer avec MQTT, mais le SDK donnait une grande facilité par son côté fonctionnel dès l'installation.

Du côté de ChirpStack, les mises à jour se font par de simples commandes d'updates dans le terminal pour les programmes nécessaires à son fonctionnement. En consultant le repository de ChirpStack, il s'agit d'une seule personne responsable (Brocaar) du développement du projet open-source. Les autres contributeurs sur ChirpStack ont de simples ajouts minimes en comparaison. Ils corrigent principalement les erreurs ou autres bugs qui peuvent surgir dans le projet. On a donc d'un côté une équipe développeur pour TTN et un unique développeur responsable pour ChirpStack.

4.6.2 Portée des appareils

Comme annoncé dans le choix des matériels, j'ai placé la passerelle au balcon du domicile afin d'avoir la plus grande portée possible. Etant au dixième étage, cette portée ne devient donc pas négligeable pour tout un coté accessible. Comme il s'agit d'un réseau public pour TTN, sa portée sera nettement plus grande que pour ChirpStack où il n'y aura qu'une seule passerelle disponible pour le réseau privé.

Sur la Figure 8, nous avions vu les diverses passerelles mises à disposition à Genève par des particuliers et professionnels. En traversant, la ville nous pouvons effectivement observer que notre appareil arrive à communiquer avec TTN et nous pouvons ainsi toujours récupérer nos données. Il y a quand même de grandes zones d'ombre dans la ville ne permettant pas forcément une communication constante. Un autre paramètre à prendre en compte et le fait que ces passerelles peuvent être déconnectées d'un moment à un autre sans notification. En augmentant le Spread Factor des appareils, la portée devient alors plus importante et la coupure n'est presque plus existante.

Il devient alors plus intéressant de tester la portée avec ChirpStack, car nous avons la communication via une seule et unique passerelle vers notre réseau privé. J'ai varié le Spread Factor de l'Arduino pour mesurer si les portées étaient suffisantes pour atteindre la passerelle. Le premier test a été effectué depuis le batiment F de l'HEG où avec un Spread Factor de 12 il était impossible de l'atteindre depuis le rez-de-chaussée mais en prenant de la hauteur (4ème étage), l'appareil arrivait à se joindre à la passerelle.

Figure 26 : Carte de position de l'arduino autour de la passerelle

Sur la figure ci-dessus, le point rouge correspond à la position de la passerelle et les points bleus, les emplacements où la connexion était établie entre l'arduino et la passerelle. Nous pouvons observer que la portée autour de la passerelle ne reste pas négligeable au vu de la taille d'antenne de notre passerelle et de nos appareils. L'utilisation du Spread Factor 12 permet d'augmenter nettement la portée de l'appareil. En contrepartie, l'envoi de données prend plus de temps avec une taille de données réduite.

4.6.3 Problèmes survenus

Un des premiers problèmes survenus et la perte des paquets envoyé par mon Arduino sur le réseau ChirpStack. En effet, environ 90% des messages n'étaient pas reçu par le serveur. Il s'agit d'une phase de test avec un envoi d'un message toutes les 20 secondes sur le serveur. Bien qu'envoyé à ce rythme de 20 secondes, le serveur recevait environ toutes les trois minutes un paquet. L'esp32 n'avait pas le même problème avec le même temps d'envoi. Après analyse du code des deux appareils et des trames reçues de l'esp32, il est apparu que l'Arduino peine à envoyer avec un Spread Factor SF7. En augmentant ce dernier vers un SF12, il n'y avait plus de perte de paquet. Etonnamment ce souci n'apparaissait pas sur TTN avec l'Arduino. Je soupçonne que cela est dû à la librairie de l'Arduino qui est normalement conçu pour être fonctionnel sur TTN à la sortie de boite.

Pour le second problème, le microcontrôleur Heltec ne possède pas d'identifiant devEUI. Il doit être généré depuis le serveur et reporté sur l'appareil. Il a été en effet conçu pour fonctionner en simple transmetteur ou récepteur LoRa. Pour le faire fonctionner avec LoRaWAN, il est nécessaire de se rendre sur le site d'Heltec et de générer un identifiant spécial qu'il faudra reporter dans le code du microcontrôleur.

Il y a aussi eu pendant ma phase d'essai de récupération de données dans la console de TTN une coupure du service de TTN pendant une heure à la suite d'une panne de leur serveur. Il faut donc faire le choix de ne pas utiliser la version community de TTN s'il est nécessaire d'avoir une haute disponibilité des services.

5. Conclusion

Pour conclure ce travail, plusieurs observations peuvent être relevées sur l'utilisation de réseaux LoRaWAN. La première est la présence très importante de The Things Network dans les recherches sur Internet. La majorité des tutoriels sur l'utilisation d'un réseau LoRaWAN se base sur TTN. LoRaWAN propose une solution sécurisée pour le déploiement d'appareils utilisant le protocole LoRa.

Les réseaux LoRa des opérateurs ne fournissent que peu d'informations et ne permettent pas une vision claire de leur offre. C'est une des raisons pour laquelle la solution de Swisscom a été écartée dans le cadre de ce travail. Il reste néanmoins intéressant dans le cadre où nous voudrions déployer un nombre important d'appareils LoRa dans plusieurs régions différentes.

Avec le nombre croissant d'IoT, il va sans dire que les appareils LPWAN vont commencer à prendre de plus en plus d'importance dans les villes. L'autonomie de ces appareils LoRa est un de leur argument majeur, permettant des déploiements avec des maintenances nettement moins coûteuses.

L'utilisation d'un réseau public comme TTN ou d'un réseau privé avec ChirpStack relève vraiment du choix d'une PME, avec une préférence pour ChirpStack qui offre une indépendance totale sur toute l'infrastructure. Si la connexion au réseau externe se coupe, le réseau interne fonctionnera toujours et les données durant l'inaccessibilité du serveur ne seront pas perdues. Le coût de déploiement de ces deux solutions est intéressant.

TTN permet un déploiement simple d'appareils sur leur réseau, mais en contrepartie les données d'autre utilisateurs peuvent transiter par notre passerelle. ChirpStack offre une solution complète et entièrement gratuite fonctionnant de la même manière que TTN. Il nécessite néanmoins plus de connaissances quant à son utilisation. Les forums communautaires de ces deux réseaux offrent de précieux conseils selon le problème qui pourrait survenir.

D'un point de vue sécurité, nous avons pu voir que LoRaWan utilise plusieurs clés de session et d'authentification pour encrypter les trames envoyées depuis nos appareils. La sécurité est garantie dans le cas d'un réseau privé comme ChirpStack, car les trames ne transitent pas par un réseau externe. En revanche, les trames sont récupérées sur un réseau externe avec TTN. Il y aura toujours le potentiel risque d'accès non-autorisé sur notre compte TTN ou à nos clés et une personne externe pourrait alors avoir accès aux appareils LoRa. Une solution envisageable est le contrôle d'intégrité supplémentaire

entre notre appareil et de notre application en utilisant le hashage avec, par exemple, SHA-1, ou une encryption pour garantir la confidentialité. Il conviendrait également de ne pas utiliser la fonction de décodeur et encodeur de TTN et de gérer ces derniers sur notre application. L'utilisation de l'OTAA est importante aussi pour rafraichir les clés de sessions à chaque reconnexion de l'appareil.

D'un point de vue personnel, j'ai pu enrichir tout au long de ce travail mes connaissances dans le domaine de l'internet des objets ainsi que du réseau. Je m'étais toujours intéressé à l'univers des microcontrôleurs et ce travail m'a permis d'en apprendre plus, tout en étudiant une technologie de communication qui m'était encore inconnue. J'ai pu découvrir le fonctionnement et le déploiement de serveur pour mettre en place une infrastructure complète et gratuite pour pouvoir utiliser LoRaWAN.

6. Bibliographie

SEMTECH - LoRa and LoRaWAN: Technical overview | DEVELOPER PORTAL, [sans date]. [en ligne]. [Consulté le 18 juin 2020]. Disponible à l'adresse : https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/

RAKwireless Documentation Center. [en ligne]. [Consulté le 15 mai 2020]. Disponible à l'adresse : https://docs.rakwireless.com/Product-Categories/WisGate/RAK7246G/Overview/

The Things Network. The Things Network Documentation [en ligne]. [Consulté le 15 mai 2020]. Disponible à l'adresse : https://www.thethingsnetwork.org/docs/

ChirpStack - ChirpStack open-source LoRaWAN® Network Server. [en ligne]. [Consulté le 18 juin 2020]. Disponible à l'adresse : https://www.chirpstack.io/project/guides/debian-ubuntu/

Tetaneutral.net - Cryptographie et sécurité LoRaWAN - Réseau LoRaWAN. [en ligne]. [Consulté le 27 juillet 2020]. Disponible à l'adresse : https://docs.lora.tetaneutral.net/lorawan/crypto/

Swisscom - Internet of Things (IoT). [en ligne]. [Consulté le 18 juin 2020]. Disponible à l'adresse : https://www.swisscom.ch/fr/business/enterprise/offre/iot.html

ANDRADE, Roberto & YOO, Sang Guun. 2019. A Comprehensive Study of the Use of LoRa in the Development of Smart Cities. Applied Sciences. [Consulté le 27 août 2020]. Disponible à l'adresse : https://www.researchgate.net/publication/337095217_A_Comprehensive_Study_of_the_Use_of_LoRa_in_the_Development_of_Smart_Cities

BROCAAR, Orne, 2018. The state and future of LoRa Server. In : Medium [en ligne]. [Consulté le 4 août 2020]. Disponible à l'adresse : https://medium.com/@brocaar/the-state-and-future-of-lora-server-b95425eea73c .

DEBAUCHE, Olivier. 2017. LoRa et LoRaWan de la théorie à la pratique. [Consulté le 19 juin 2020]. Disponible à l'adresse : https://www.researchgate.net/publication/322138636_LoRa_et_LoRaWan_de_la_theorie_a_la_pratique

ERTÜRK, Mehmet Ali & AYDIN, Muhammed & BÜYÜKAKKAŞLAR, Talha & EVIRGEN, Hayrettin. 2019. A Survey on LoRaWAN Architecture, Protocol and Technologies. Future Internet. [Consulté le 4 août 2020]. Disponible à l'adresse : https://www.mdpi.com/1999-5903/11/10/216/pdf

GILBERT, Johann. 2018. Étude et développement d'un réseau de capteurs synchronisés à l'aide d'un protocole de communication sans fil dédié à l'Internet des objets. Traitement du signal et de l'image. Université de Toulon. [Consulté le 4 août 2020]. Disponible à l'adresse : https://tel.archives-ouvertes.fr/tel-02398271v2/document

KOON, John., 2019. Cybersecurity basics: Authentication and « key » management in LoRaWAN. In : [en ligne]. [Consulté le 27 juillet 2020]. Disponible à l'adresse : https://www.microcontrollertips.com/cybersecurity-basics-authentication-key-management-lorawan-faq/ .

LUETH, Knud Lasse., 2018. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. In : [en ligne]. [Consulté le 1 septembre 2020]. Disponible à l'adresse : https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/ .

PASQUA, Eugenio., 2020. 5 Things to know about the LPWAN market in 2020. In : [en ligne]. [Consulté le 1 septembre 2020]. Disponible à l'adresse : https://iot-analytics.com/5-things-to-know-about-the-lpwan-market-in-2020/ .

RAM, Prashant, 2018. LPWAN, LoRa, LoRaWAN and the Internet of Things. In : Medium [en ligne]. 15 octobre 2018. [Consulté le 2 septembre 2020]. Disponible à l'adresse : https://medium.com/coinmonks/lpwan-lora-lorawan-and-the-internet-of-things-aed7d5975d5d .

Wikipedia - Chirp, 2020. [en ligne]. [Consulté le 30 juillet 2020]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Chirp&oldid=963915118

Wikipedia - Narrowband IoT, 2020. [en ligne]. [Consulté le 1 septembre 2020]. Disponible à l'adresse : . https://en.wikipedia.org/w/index.php?title=Narrowband_IoT&oldid=968300943

Wikipedia - LoRaWAN, 2020 [en ligne]. [Consulté le 18 mai 2020]. Disponible à l'adresse: https://fr.wikipedia.org/w/index.php?title=LoRaWAN&oldid=170342342

i-SCOOP - LoRa and LoRaWAN: the technologies, ecosystems, use cases and market. In : *i-SCOOP* [en ligne]. [Consulté le 19 juin 2020]. Disponible à l'adresse : https://www.i-scoop.eu/internet-of-things-guide/lpwan/iot-network-lora-lorawan/.

LoRa Alliance Technical Committee,2018 - LoRaWAN™ 1.1 Specification. [en ligne]. [Consulté le 4 août 2020]. Disponible à l'adresse : https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf

Annexe 1 : Tutoriel d'installation du réseau The Things Network

Création de compte : https://www.thethingsnetwork.org/

Enregistrement d'une passerelle

Source :

https://docs.rakwireless.com/Product-Categories/WisGate/RAK7246G/Quickstart/#connecting-to-the-things-network-ttn

Connecting to the Things Network (TTN)

The Things Network is about enabling low power devices to use long range gateways to connect to an open-source, decentralized network to exchange data with Application. Learn more about the Things Network through their documentation.

- First, you should have connected your Gateway into the internet through a router according to the method which has been introduced in the Accessing the Internet section.
- Second, config your Gateway and choose TTN as the LoRa Server and choose a correct frequency according to the method which has been introduced in the Configuring the Gateway section.
- Now go to the TTN Website and Login. You will then see the following page:

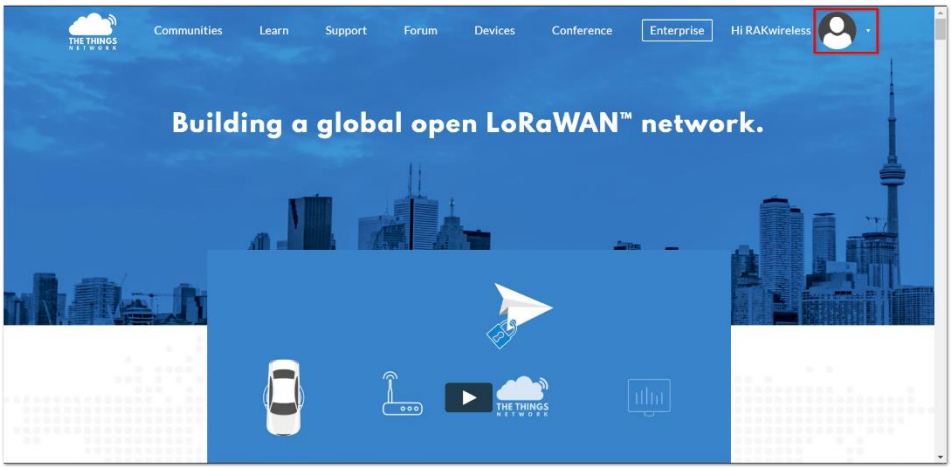


Figure 26: The Things Network Home Page

- Choose Console then Click Gateways.

The Things Network Console Page

Figure 27: The Things Network Console Page

- All of your Registered Gateways will be displayed here in this page. Click "register gateway"

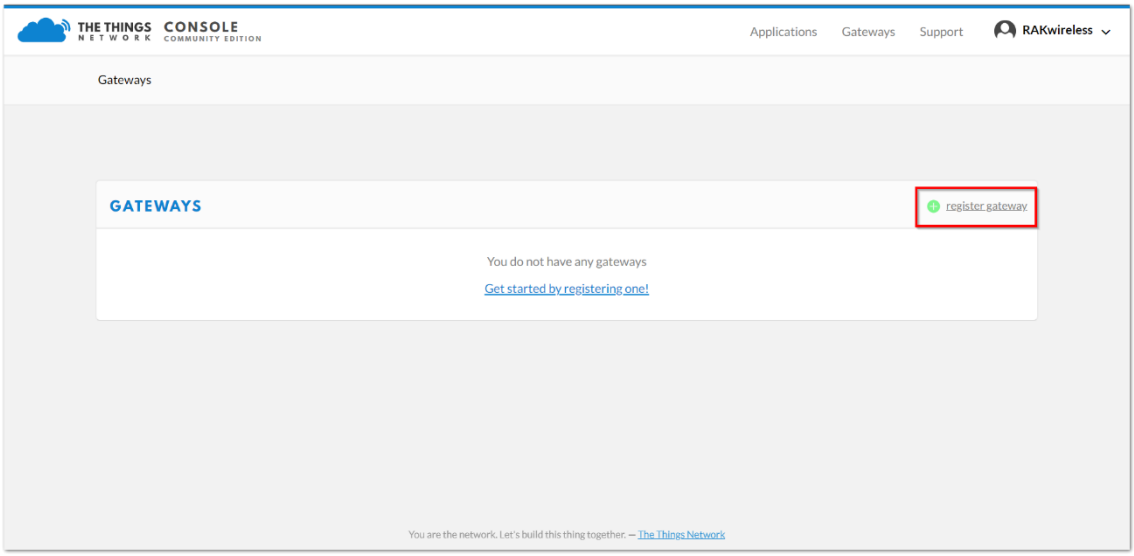


Figure 28: Adding a Gateway to TTN

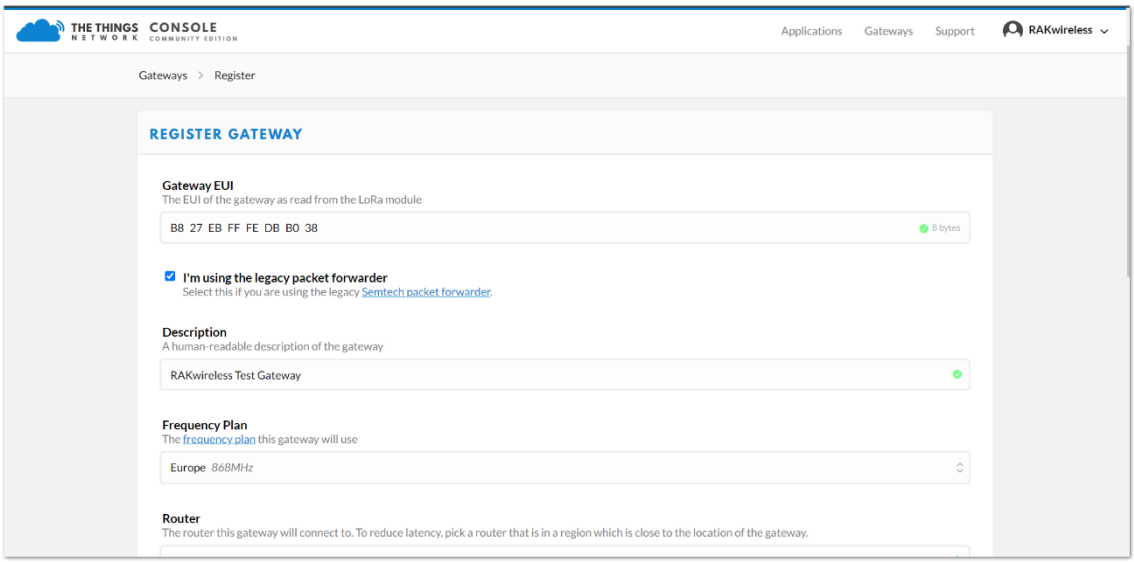


Figure 29: Registering your Gateway

- Gateway EUI - refers to the Gatway ID you obtained from the previous steps. In case you forgot, just type gateway-version in the command line. This must be the same with the Gateway's True Gateway ID otherwise you will fail to register your Gateway on TTN.

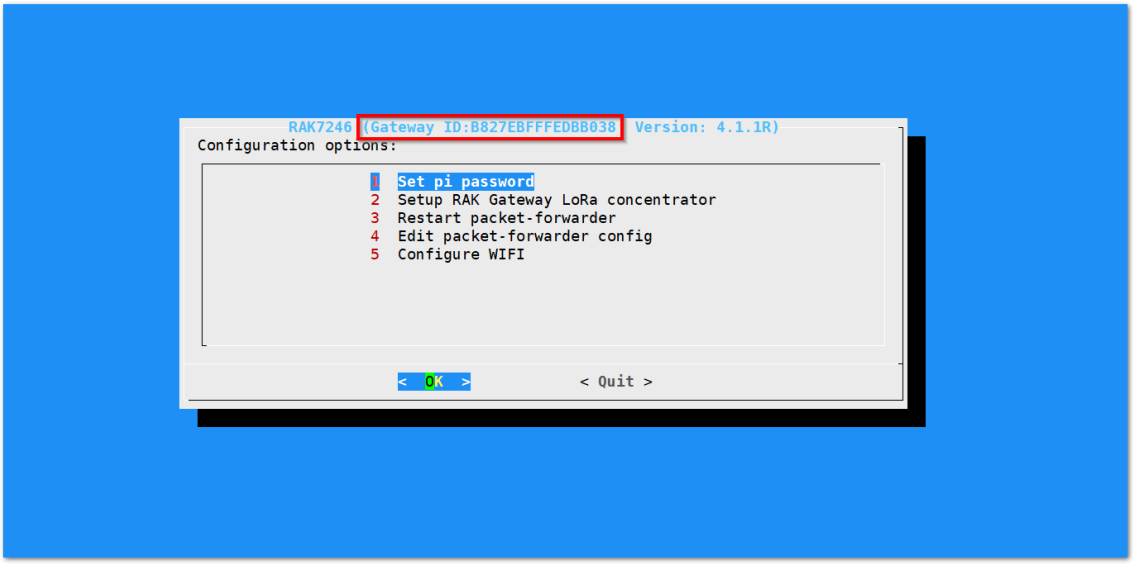


Figure 30: RAK7246G WisGate Developer D0 Gateway ID in SSH

NOTE: Make sure to select the "I'm using the legacy packet forwarder" check box.

- Description - A human readable description of your Gateway.
- Frequency Plan - This is the frequency you want to use and it must be the same with Gateway and the Node.
- Router - The router this gateway will connect to. To reduce latency, pick a router that is in a region which is close to the location of the gateway.
- Location - Choose the location of the Gateway by entering its coordinates. This is reflected on the Gateway World Map.
- Antenna Placement - Where is your antenna placed? Is it placed indoors or outdoors?

Click Register Gateway and wait for a couple of minutes. If the status of your gateway is Connected, Congratulations! Your Gateway is now connected to the The Things Network (TTN).

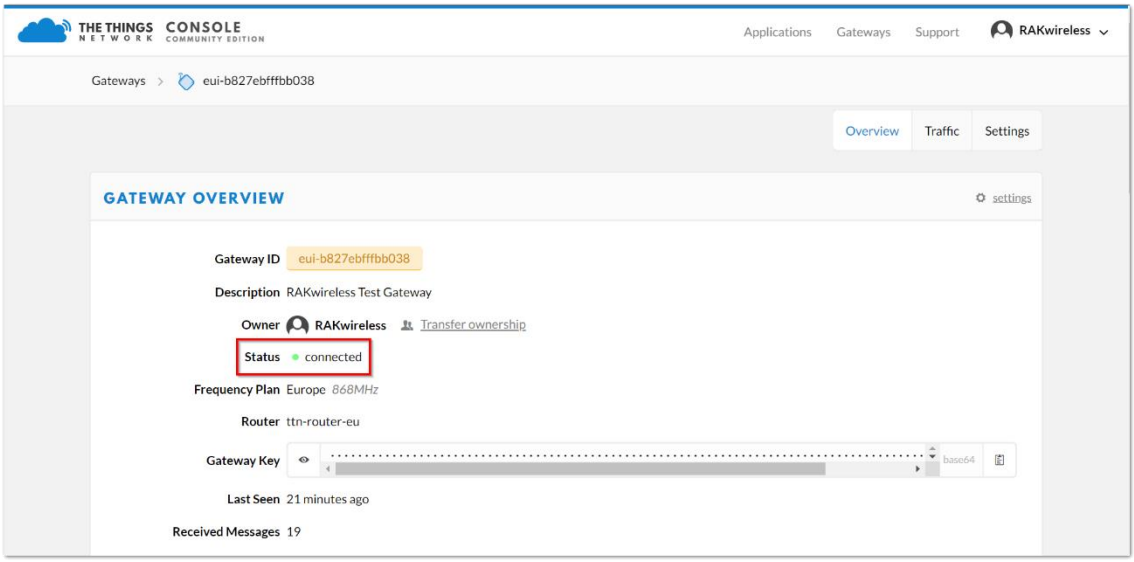


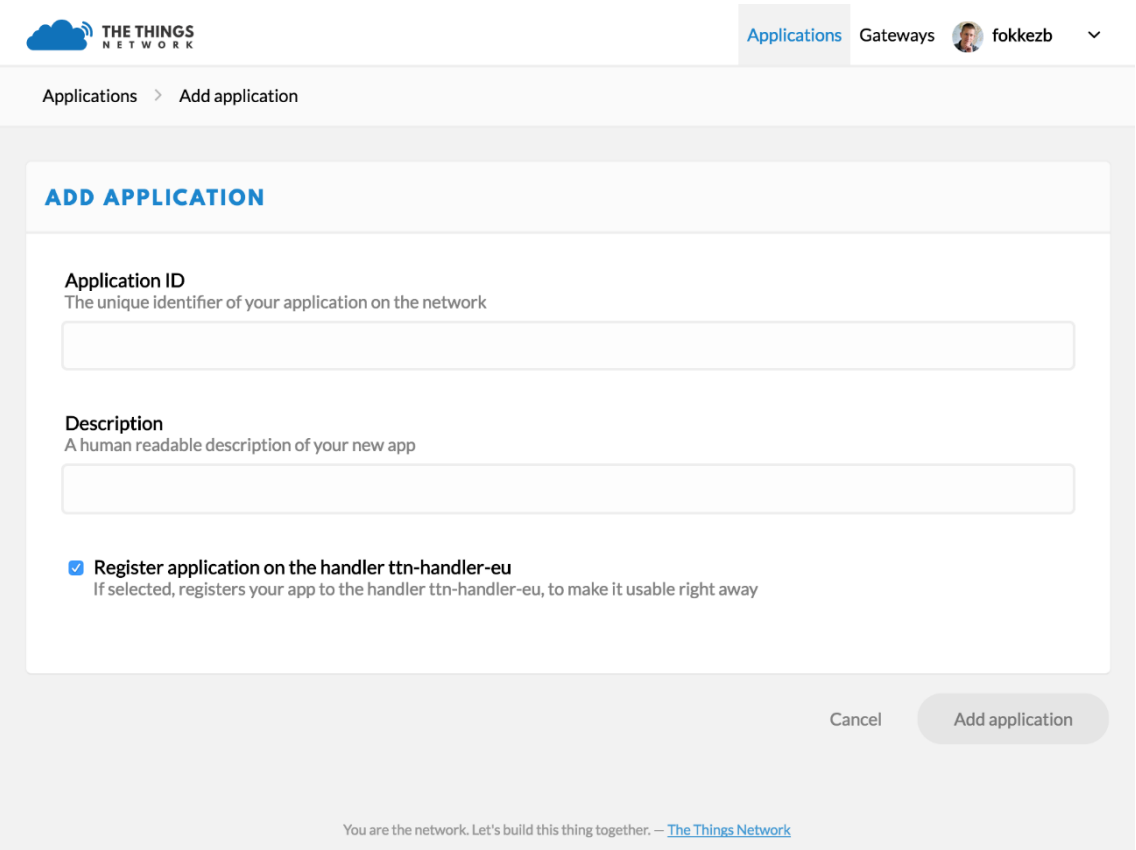
Figure 31: RAK7246G WisGate Developer D0 Gateway TTN Connection Success

Création d'une application sur TTN

Source: https://www.thethingsnetwork.org/docs/applications/add.html

Devices can communicate with applications that they've been registered to. To register a device, you'll first need to add an application.

- In the console, click add application.
    - For Application ID, choose a unique ID of lower case, alphanumeric characters and nonconsecutive - and _.
    - For Application Description, enter anything you like.
    - Leave the checkbox enabled to automatically register the application to your default region.



Click Add Application to finish.

You will be redirected to the newly added Application page where you can find the generated App EUI and Access Keys.

Enregistrement d'un appareil

Source: https://www.thethingsnetwork.org/docs/devices/registration.html

To use the default Over The Air Activation (OTAA) you will need to register your device with its Device EUI. See the documentation of your device for instructions, as for example that of The Things Uno.

Open the application to which you wish to add a device and click register device.

- For Device ID, choose a - for this application - unique ID of lower case, alphanumeric characters and nonconsecutive - and _.
- For Device EUI, copy-paste the one you retrieved from your device.
- If you plan to switch to ABP anyway, click the button have one generated for you.
- Leave the App Key to be generated.
- For App EUI, select the generated EUI from the list.

Register Device (OTAA)

Click Register to finish.

You will be redirected to the newly registered device where you can find the generated App Key needed to activate the device.

Annexe 2 : Tutoriel d'installation du réseau ChirpStack

Source : https://www.chirpstack.io/

Pour l'installation de ChirpStack, il faut se rendre sur le site web de ChirpStack https://www.chirpstack.io/ . Un guide d'installation est disponible sur le site avec les étapes importantes nécessaires que nous allons détailler ici. Nous allons utiliser la passerelle Raspberry en tant que simple forwarder et avoir un serveur dédié à la réception de ces paquets. Cela peut aussi permettre par la suite d'intégrer d'autres passerelles au serveur plus facilement. L'installation de ChirpStack decrite ici sera sur l'OS Ubuntu 20.04 tournant dans une VM avec sa propre adresse IP.

Pour débuter, il est important d'installer les dépendances nécessaires au fonctionnement de ChirpStack. En effet, ce dernier nécessite Mosquitto (Broker MQTT), Redis (Système de gestion de base de données en cache) et PostgreSQL (Base de données). Nous allons utiliser Ubuntu tout au long pour installer les composants requis et les modifications nécessaires des fichiers.

Pour installer Mosquitto, Redis et PostgreSQL :

```
sudo apt install mosquitto mosquitto-clients redis-server redis-tools postgresql
```

Il est ensuite nécessaire paramétrer le repository de ChirpStack, il est nécessaire de rentrer la bonne clé du repository de ChirpStack indiqué ci-dessous.

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

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
1CE2AFD36DBCCA00

sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable main" | sudo tee
/etc/apt/sources.list.d/chirpstack.list

sudo apt update
```

Nous pouvons maintenant installer Gateway Bridge sur le serveur. La commande ci-dessous avec start permet de démarrer le composant et la commande enable permet son exécution à chaque démarrage du serveur. Il est à noter qu'il est possible d'installer le Gateway Bridge directement sur votre propre passerelle LoRa. ChirpStack fournit une

aide d'installation selon le modèle que vous possédez. Dans le cas du Rak7246 il y a déjà une préconfiguration des fréquences pour fonctionner

```
sudo apt install chirpstack-gateway-bridge
sudo systemctl start chirpstack-gateway-bridge
sudo systemctl enable chirpstack-gateway-bridge
```

Avant de configurer le Network Server et l'Application Server, nous devons créer leurs bases de données ainsi que leur utilisateurs et mots de passes. Il est important que vous configuriez vos propres utilisateurs et mots de passe pour les deux bases de données :

```
sudo -u postgres psql
create role okanchirpstack_as with login password 'dbpassword';
create role okanchirpstack_ns with login password 'dbpassword';
create database chirpstack_as with owner okanchirpstack_as;
create database chirpstack_ns with owner okanchirpstack_ns;
\c chirpstack_as
create extension pg_trgm;
create extension hstore;
\q
```

L'installation du Network Server est de la même manière que celui du Gateway Bridge :

```
sudo apt install chirpstack-network-server
```

Avant de le démarrer, il est maintenant nécessaire de modifier le fichier de configuration du Network Server pour ajouter notre utilisateur et son mot de passe. Ce fichier se situe au chemin

/etc/chirpstack-application-server/chirpstack-application-server.toml

À la ligne dsn=''… nous le modifions avec nos identifiants créer précédemment.

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

```
sudo systemctl start chirpstack-network-server
sudo systemctl enable chirpstack-network-server
```

Pour installer l'Application Server, il faut entrer la commande suivante :

```
sudo apt install chirpstack-application-server
```

Comme pour le network serveur, il est obligatoire de modifier le fichier de configuration de l'AS qui se situe à */etc/chirpstack-application-server/chirpstack-application-server.toml* .

À la ligne dsn=''… nous le modifions avec nos identifiants créer précédemment.

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

Et à la ligne *jwt_secret* dans *[application_server.external_api]* nous ajoutons une clé secrète que nous générons dans le terminal avec la commande

```
openssl rand -base64 32
```

Nous pouvons maintenant démarrer l'application server :

```
sudo systemctl start chirpstack-application-server
sudo systemctl enable chirpstack-application-server
```

Le serveur est maintenant opérationnel, mais un redémarrage du serveur est conseillé.

Configuration des appareils

Nous pouvons dès à présent utiliser ChirpStack en local en accèdant via l'adresse localhost:8080 depuis le serveur ou via l'adresse ip du serveur IP_SERVEUR:8080 .

Le nom d'utilisateur et le mot de passe par défaut est 'admin' et 'admin'. Il est fortement recommandé de les changer en allant dans l'onglet *All users*.

Avant de pouvoir configurer notre passerelle, on doit créer un serveur réseau dans l'onglet Network-servers, un Service-profile et un Device-profile :

- Pour le Network server
    - Ajouter un nouveau profil en cliquant sur add
    - Dans l'onglet General, mettre le nom que vous voulez dans name et ajouter l'adresse 127.0.0.1:8080
- Service-profiles
    - Créer un nouveau profil
    - Lui donner un nom et cliquer sur Create en bas
- Device-profiles
    - Créer un nouveau Device-profiles
    - Dans l'onglet General, choisir la version LoRaWAN MAC 1.0.2 , A pour le paramètre de révision régional et 30 pour Max EIRP. Le choix du LoRaWAN MAC dépend de la comptabilité des appareils utilisés. Dans notre installation, ils sont tous comptatibles avec la version 1.0.2.
    - Dans l'onglet Join (OTAA/ABP), cocher Device supports OTAA
    - Cliquer sur le bouton Create.

Maintenant que le minimum nécessaire a été configuré pour ajouter notre gateway et notre arduino, nous allons procéder à leur mise en place. Nous devons aussi reconfigurer la passerelle RAK pour la faire fonctionner avec ChirpStack

Configuration de la passerelle

(Source :

https://docs.rakwireless.com/Product-Categories/WisGate/RAK7246G/Quickstart/#configuring-the-gateway)

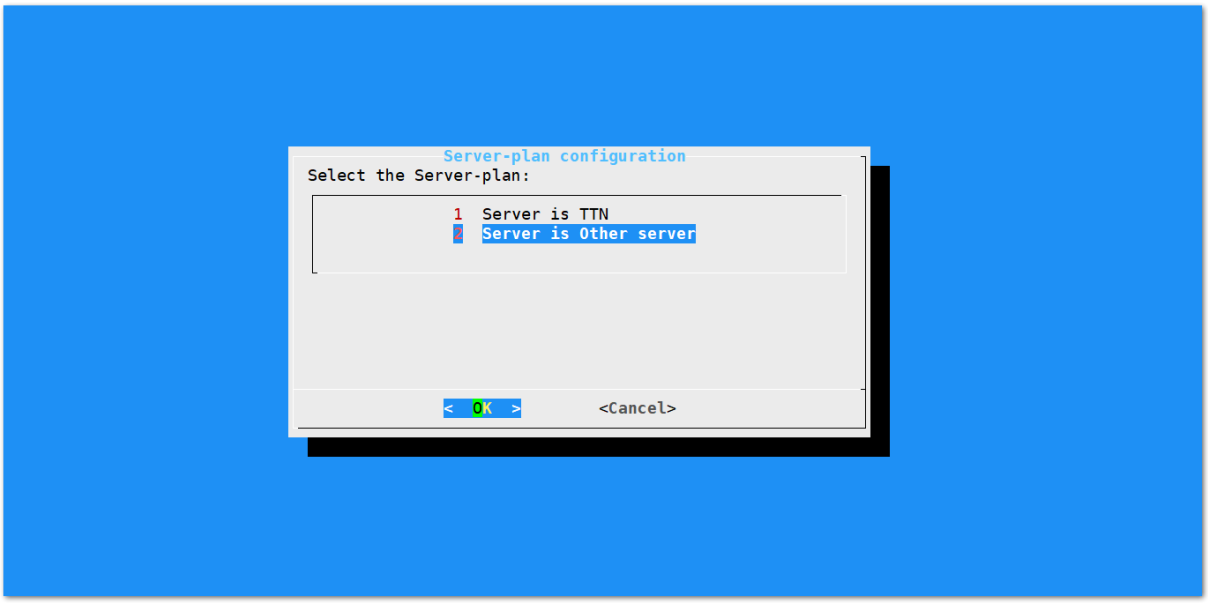


Figure 22: Server Is Chirpstack

- **ChirpStack** - If you choose Chirpstack as your LoRa Server, choose "2 Server is Other server". First, configure your Regional Frequency Band by choosing the option below:

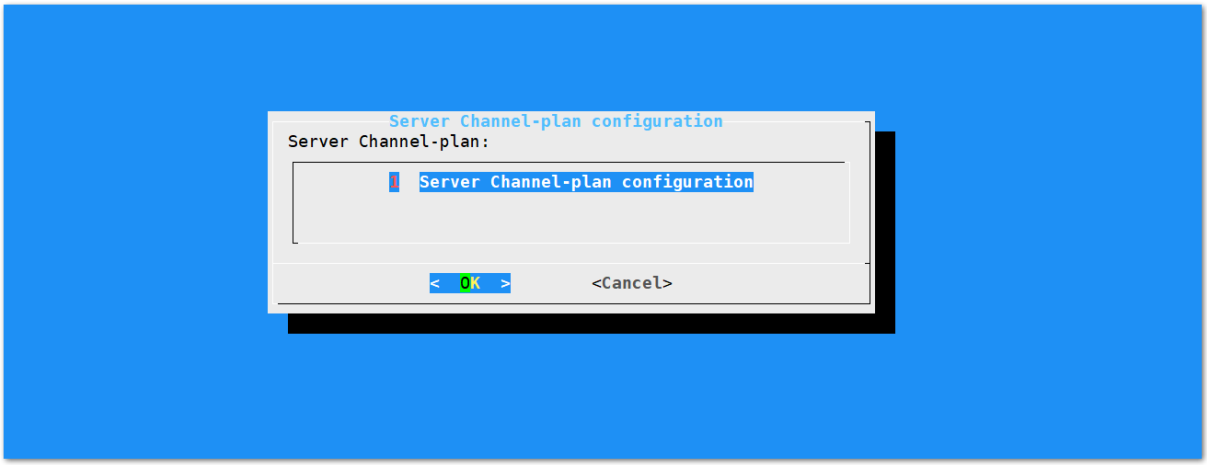


Figure 23: Regional Frequency Band Option

For this example, we will be using EU868 Frequency Plan.

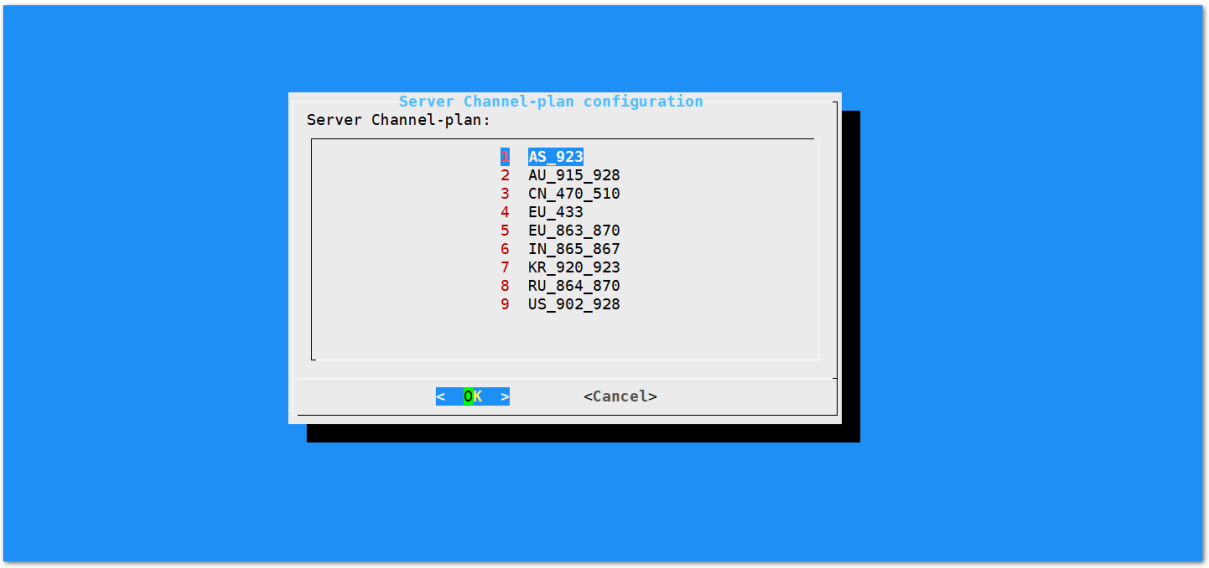


Figure 24: Selecting the Chirpstack Channel Plan

Then, set the IP address of the ChirpStack which you want your Gateway to work with:

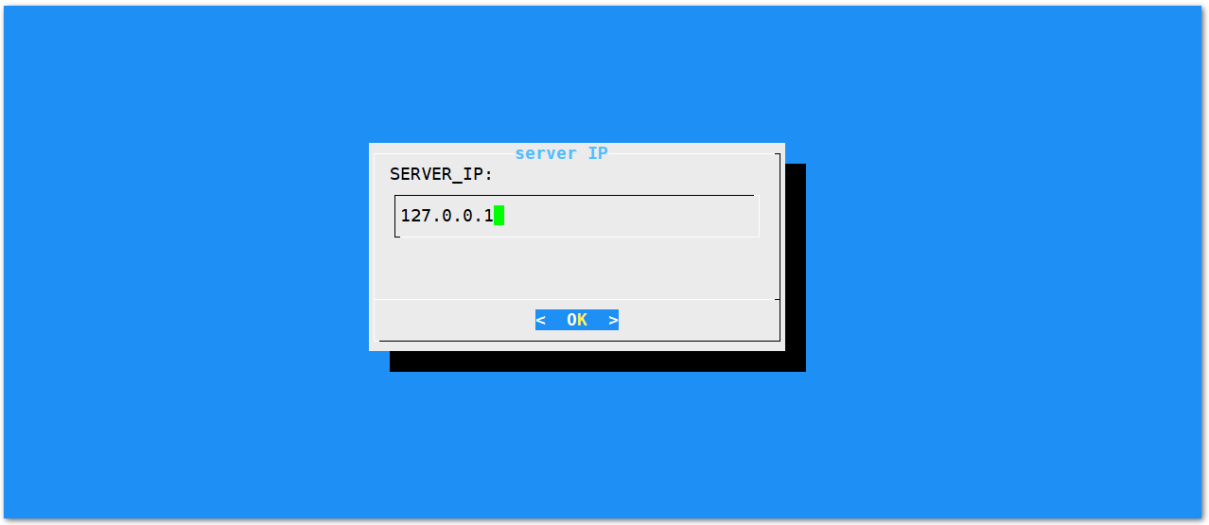


Figure 25: Default LoRaServer IP Address

Assuming you have set it up correctly, Login to your ChirpStack to register your Gateway by opening the ChirpStack's web page in a browser by entering "**IP Address of ChirpStack:8080**".

- If you are using an Independent Chirpstack, use the IP Address you have set in the Configuring the Gateway document.
- If you are using the RAK Free Cloud Server Chirpstack `209.250.251.9`

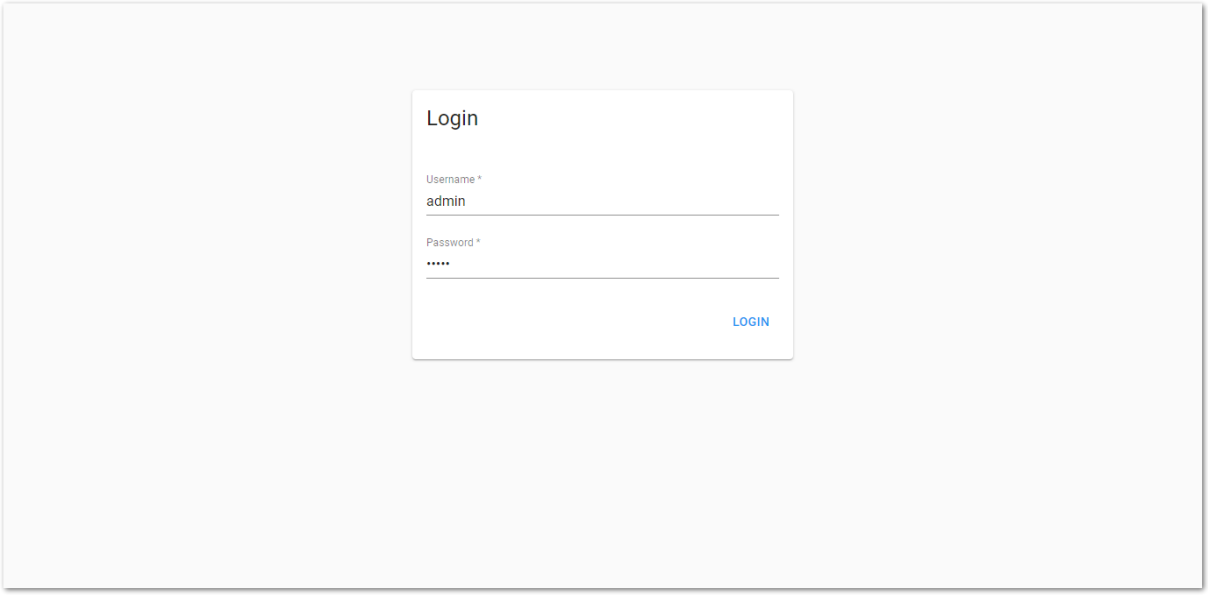


Figure 33: ChirpStack Login Page

- The default username is "**admin**" and the password is also "**admin**"

NOTE:

If you are using the RAK Cloud Testing ChirpStack, input the account and password you have asked in the forum provided beforehand.

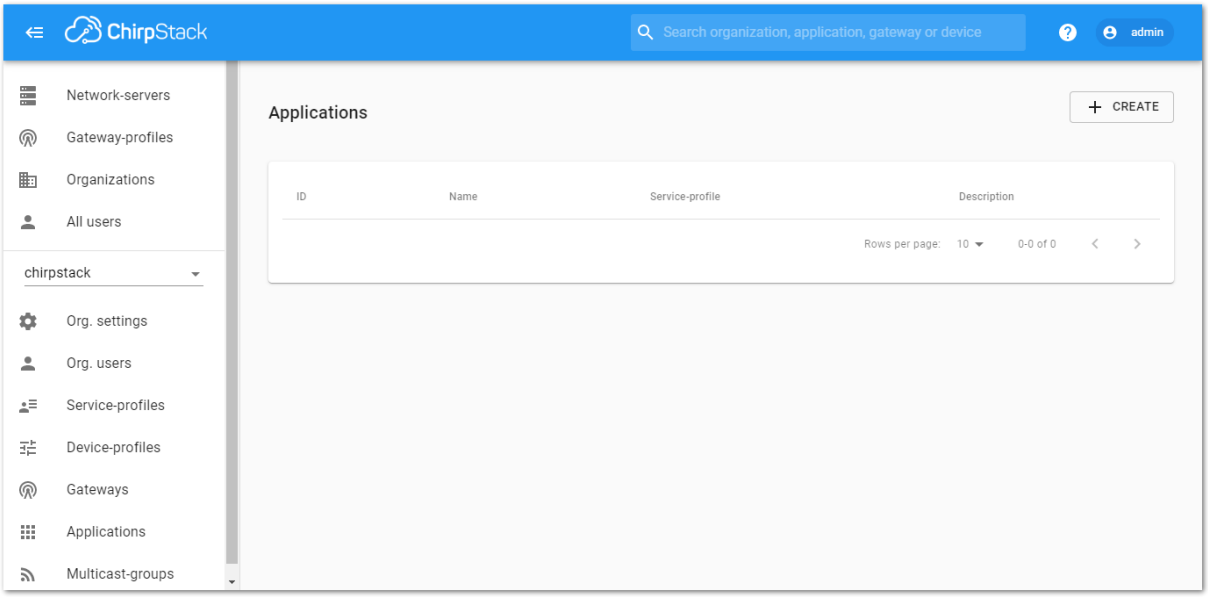


Figure 34: ChirpStack Home Page

- Click "**Gateways**" in the left menu and Press "**+ CREATE**" to register your Gateway

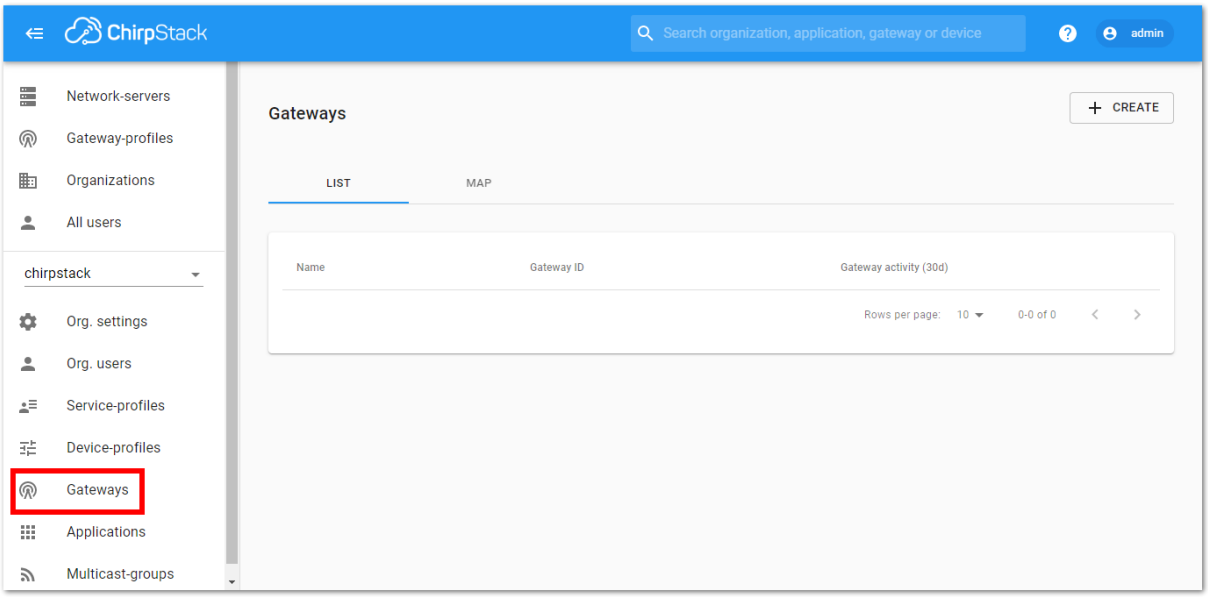


Figure 35: ChirpStack Registered Gateways

- Click "Create" to register your Gateway and fill up the necessary information.

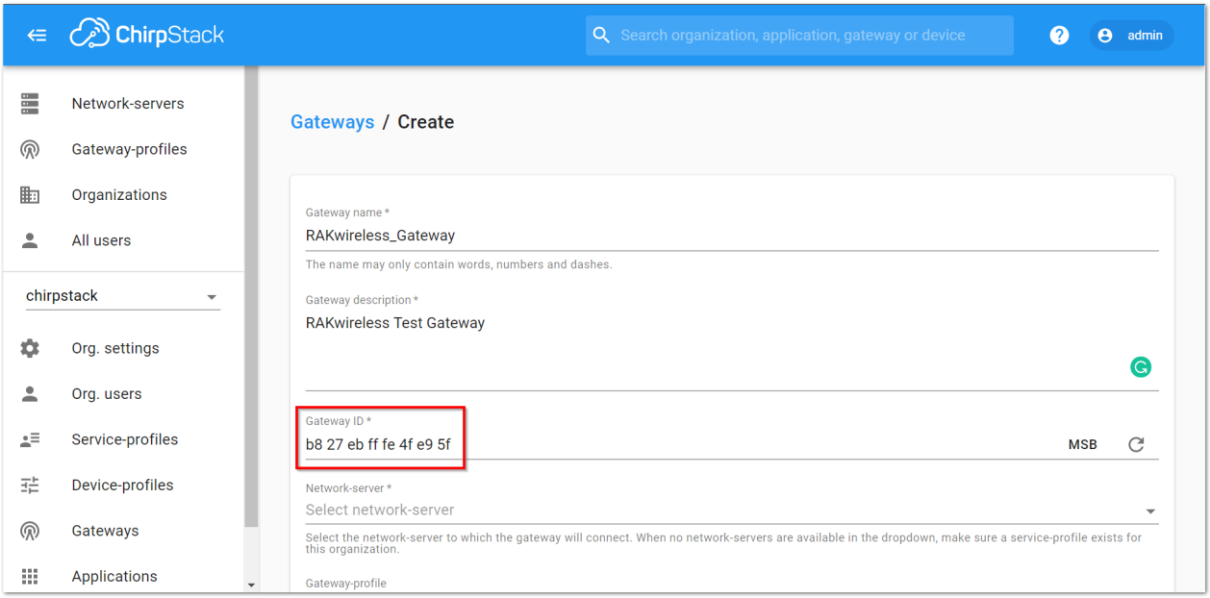


Figure 36: Registering your own Gateway

- Fill in the Gateway ID that we got from the last section (Configuring the Gateway), also called Gateway EUI.

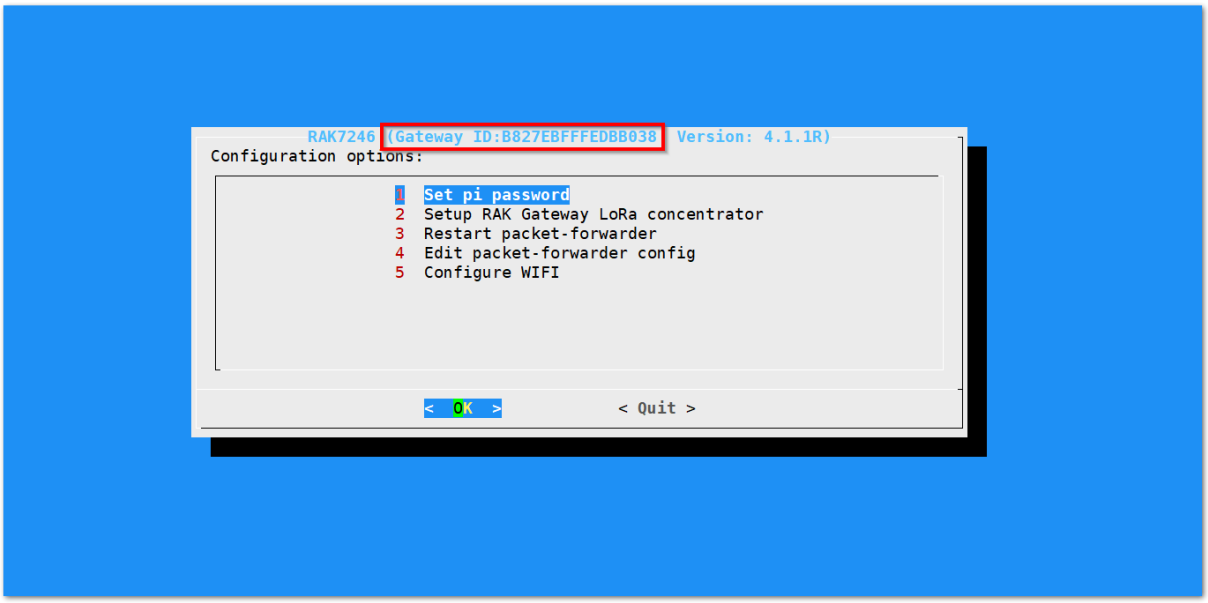


Figure 37: Gateway ID

- If you have properly configured your Gateway and there is a network connection between the external ChirpStack and your Gateway, you should see the following page and status:

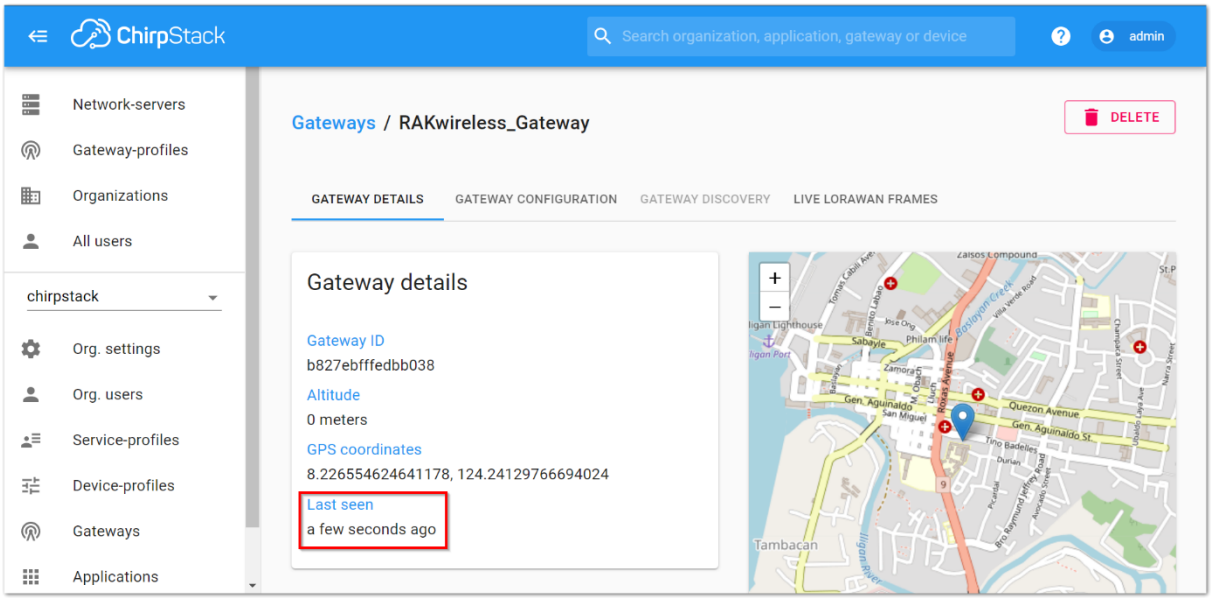


Figure 38: Successfully Registered the Gateway

- By clicking the Live LORAWAN® FRAMES tab, you can check the LoRa packets sent by the nodes into your RAK7246G WisGate Developer D0 Gateway

Congratulations! 🎉 You have connected your Gateway to an external ChirpStack Successfully!

Configuration d’un appareil

- Applications
    - Créer une nouvelle application
    - Donner un nom et une description et choisir le service-profile créé précédemment et cliquer sur Create pour confirmer la création.
- Devices
    - Maintenant qu’une application a été créée, cliquer sur cette application et créer un nouvel appareil
    - Donner un nom et une description à l’appareil et entrer le Device EUI de l’appareil (Arduino) ou le générer pour l’entrer dans la configuration du microcontrôleur (Heltec). Choisir le device-profile créé précédemment.
    - Cliquer sur Create pour confirmer la création.
    - Maintenant que le device a été créé, cliquer dessus pour générer les clés de connexion.
        - Dans le cas d’une connection OTAA, aller dans l’onglet Keys (OTAA) et générer une Application key et une Gen Application Key et enregistrer la modification. Reporter l’Application key sur votre appareil pour permettre la connection OTAA.
        - Pour une connexion ABP, aller dans l’onglet Activation et générer le Device address, le Network session key et l’Application session key. Reporter ces 3 données sur votre appareil pour les connecter en ABP.

Arrivé ici, notre gateway et notre appareil lora sont fonctionnels. Vous pouvez consulter l’état des deux appareils sur leur page respectives où vous pouvez retrouver l’heure de leur dernier message sur ChirpStack. Lors d’une connexion entre un device et la gateway, nous pouvons constater que les messages du device dans son live data frame apparait comme un « ConfirmedDataUp ».

Annexe 3 : Configuration de l'Arduino MKR 1310

Code basé sur la librairie MKRWAN d'Arduino

Fichier ArduinoEnvTTN.ino

```
/*
   ARDUINO MKRWAN avec SHIELD ENV
*/

#include <Arduino_MKRENV.h>
#include <MKRWAN.h>
#include <Arduino.h>
#include <U8g2lib.h>
#include <Arduino_JSON.h>
#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif

LoRaModem modem;

#include "arduino_secrets.h"

// OTAA
String appEui = SECRET_APP_EUI;
String appKey = SECRET_APP_KEY;

//ABP
String devAddr = SECRET_DEV_ADDR;
String nwkSKey = SECRET_NWK_SKEY;
String appSKey = SECRET_APP_SKEY;

int mode = 1; // 1 for OTAA and 2 for ABP

// Ecran config
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);

void setup() {
  Serial.begin(115200);

  //while (!Serial); // Forcer la connexion sur le Serial si besoin

  // change this to your regional band (eg. US915, AS923, ...)
  if (!modem.begin(EU868)) {
    Serial.println("Failed to start module");
    while (1) {}
  };
  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
    while (1);
  }
  Serial.print("Your module version is: ");
  Serial.println(modem.version());
  Serial.print("Your device EUI is: ");
  Serial.println(modem.deviceEUI()); // Pour récupérer le deviceEUI si
besoin
```

```
  // Ecran
  u8g2.begin();
  u8g2.enableUTF8Print();    // enable UTF8 support for the Arduino
print() function

  int connected;
  if (mode == 1) {
    connected = modem.joinOTAA(appEui, appKey);
  } else if (mode == 2) {
    connected = modem.joinABP(devAddr, nwkSKey, appSKey);
  }

  if (!connected) {
    Serial.println("Something went wrong; are you indoor? Move near a
window and retry");
    while (1) {}
  }
  delay(5000);

  int err;
  modem.setPort(3);
  modem.dataRate(5);
  modem.beginPacket();
  modem.print("HeLoRA world!");
  err = modem.endPacket(true);
  if (err > 0) {
    // Serial.println("Message sent correctly!");
  } else {
    // Serial.println("Error sending message :(");
  }
}

void loop() {


  // Lecture des données sur les sensors
  float temperature = roundf(ENV.readTemperature() * 100) / 100;
  float humidity    = ENV.readHumidity();
  float pressure    = ENV.readPressure();
  float illuminance = ENV.readIlluminance();
  float uva         = ENV.readUVA();
  float uvb         = ENV.readUVB();
  float uvIndex     = ENV.readUVIndex();

  // Affichage sur l'écran OLED
  u8g2.setFont(u8g2_font_b10_t_japanese1);
  u8g2.setFontDirection(0);
 // u8g2.clearBuffer();
  u8g2.firstPage();
  do {
    u8g2.setCursor(0, 10);
    u8g2.print("Temperature = ");
    u8g2.print(temperature);
    u8g2.print(" °C");
    u8g2.setCursor(0, 25);
    u8g2.print("Humidity = ");
    u8g2.print(humidity);
    u8g2.print(" %");
    u8g2.setCursor(0, 40);
    u8g2.print("Pressure = ");
```

```
    u8g2.print(pressure);
    u8g2.print(" kPa");
    u8g2.setCursor(0, 55);
    u8g2.print("UV Index    = ");
    u8g2.print(uvIndex);
  } while ( u8g2.nextPage() );
 // u8g2.sendBuffer();

  modem.beginPacket();
  int err;
  modem.print(temperature);
  modem.print(humidity);
  modem.print(pressure);
  modem.print(uvIndex);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
  }


  delay(300000);
}
```

Fichier arduino_secrets.h :

```
// TTN OTAA
#define SECRET_APP_EUI "ADEFINIR"
#define SECRET_APP_KEY "ADEFINIR"

// ChirpStack OTAA
//#define SECRET_APP_EUI "ADEFINIR"
//#define SECRET_APP_KEY "ADEFINIR"


// ABP
#define SECRET_DEV_ADDR "ADEFINIR"
#define SECRET_NWK_SKEY "ADEFINIR"
#define SECRET_APP_SKEY "ADEFINIR"
```

Pour décoder le payload des Arduinos

- Pour The Things Network et ChirpStack

```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var temperature = parseFloat(String.fromCharCode(bytes[0],bytes[1],byte
s[2],bytes[3],bytes[4]));
  var humidity = parseFloat(String.fromCharCode(bytes[5],bytes[6],bytes[7
],bytes[8],bytes[9]));
  var pressure = parseFloat(String.fromCharCode(bytes[10],bytes[11],bytes
[12],bytes[13],bytes[14]));
  var uvIndex = parseFloat(String.fromCharCode(bytes[15],bytes[16],bytes[
17],bytes[18],bytes[19]));
  return {
          'temperature': temperature,
          'humidity': humidity,
          'pressure': pressure,
          'uvIndex': uvIndex,

        }
}
```

Annexe 4 : Configuration de l'Heltec LoRa 32 V2

Code basé sur la librairie LoRaWAN d'Heltec

```
#include <ESP32_LoRaWAN.h>
#include "Arduino.h"

/*license for Heltec ESP32 LoRaWan, quary your ChipID relevant
license: http://resource.heltec.cn/search */
uint32_t  license[4] = {0xDC5ED31E,0x4AE9A785,0xF4E1B493,0x7D270188};

/* OTAA para*/
uint8_t DevEui[] = { 0x00, 0xEC, 0x96, 0x4E, 0x03, 0xAE, 0x27, 0x46 };
uint8_t AppEui[] = { 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x03, 0x2E, 0xD7 };
uint8_t AppKey[] = { 0xC1, 0x1B, 0x19, 0x5C, 0xCA, 0x13, 0xE0, 0x5D,
0xF6, 0x5D, 0x0A, 0x85, 0xA7, 0xBB, 0x39, 0xEE };

/* ABP para*/
uint8_t NwkSKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe,
0x3d, 0x11, 0x18, 0x1e, 0x1e, 0xc7, 0xda,0x85 };
uint8_t AppSKey[] = { 0xd7, 0x2c, 0x78, 0x75, 0x8c, 0xdc, 0xca, 0xbf,
0x55, 0xee, 0x4a, 0x77, 0x8d, 0x16, 0xef,0x67 };
uint32_t DevAddr =  ( uint32_t )0x007e6ae1;

/*LoraWan Class, Class A and Class C are supported*/
DeviceClass_t  loraWanClass = CLASS_A;

/*the application data transmission duty cycle.  value in [ms].*/
uint32_t appTxDutyCycle = 15000;

/*OTAA or ABP*/
bool overTheAirActivation = true;

/*ADR enable*/
bool loraWanAdr = true;

/* Indicates if the node is sending confirmed or unconfirmed messages
*/
bool isTxConfirmed = true;

/* Application port */
uint8_t appPort = 2;

uint8_t confirmedNbTrials = 8;

/*LoraWan debug level, select in arduino IDE tools.
* None : print basic info.
* Freq : print Tx and Rx freq, DR info.
* Freq && DIO : print Tx and Rx freq, DR, DIO0 interrupt and DIO1
interrupt info.
* Freq && DIO && PW: print Tx and Rx freq, DR, DIO0 interrupt, DIO1
interrupt, MCU sleep and MCU wake info.
*/
uint8_t debugLevel = LoRaWAN_DEBUG_LEVEL;

/*LoraWan region, select in arduino IDE tools*/
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;
```

```
long randomTemp;
long randomHumidity;
long randomUvIndex;
long randomPressure;

static void prepareTxFrame( uint8_t port )
{

    appDataSize = 4;//AppDataSize max value is 64
    float temperature = 27.84 ;
    char temperatureS[5];
    randomTemp = random(30);
    randomHumidity = random(60);
    randomUvIndex = random(2);
    randomPressure = random(120);
    Serial.println(randomTemp);
    Serial.println(randomHumidity);
    Serial.println(randomUvIndex);
    Serial.println(randomPressure);
    appData[0] = (char) (randomTemp);
    appData[1] = (char) (randomHumidity);
    appData[2] = (char) (randomUvIndex);
    appData[3] = (char) (randomPressure);

}

// Add your initialization code here
void setup()
{
  if(mcuStarted==0)
  {
    LoRaWAN.displayMcuInit();
  }
  Serial.begin(115200);
  while (!Serial);
  SPI.begin(SCK,MISO,MOSI,SS);
  Mcu.init(SS,RST_LoRa,DIO0,DIO1,license);
  deviceState = DEVICE_STATE_INIT;

  randomSeed(analogRead(37));
}

// The loop function is called in an endless loop
void loop()
{
  switch( deviceState )
  {
    case DEVICE_STATE_INIT:
    {
      LoRaWAN.init(loraWanClass,loraWanRegion);
      break;
    }
    case DEVICE_STATE_JOIN:
    {
      LoRaWAN.displayJoining();
      LoRaWAN.join();
      break;
    }
    case DEVICE_STATE_SEND:
```

```
    {
      LoRaWAN.displaySending();
      prepareTxFrame( appPort );
      LoRaWAN.send(loraWanClass);
      deviceState = DEVICE_STATE_CYCLE;
      break;
    }
    case DEVICE_STATE_CYCLE:
    {
      // Schedule next packet transmission
      txDutyCycleTime = appTxDutyCycle + randr( -APP_TX_DUTYCYCLE_RND,
APP_TX_DUTYCYCLE_RND );
      LoRaWAN.cycle(txDutyCycleTime);
      deviceState = DEVICE_STATE_SLEEP;
      break;
    }
    case DEVICE_STATE_SLEEP:
    {
      LoRaWAN.displayAck();
      LoRaWAN.sleep(loraWanClass,debugLevel);
      break;
    }
    default:
    {
      deviceState = DEVICE_STATE_INIT;
      break;
    }
  }
}
-------------------------------------------------
-------------------------------------------------
```

Pour décoder le payload des Arduinos

- Pour The Things Network et ChirpStack

```
function Decoder(bytes, port) {

 var temperature = parseInt(bytes[0]);
 var humidity = parseInt(bytes[1]);
 var uvIndex = parseInt(bytes[2]);
 var pressure = parseInt(bytes[3]);
 return {
       'temperature': temperature,
       'humidity': humidity,
       'uvIndex': uvIndex,
       'pressure': pressure,
     }
}
```

Annexe 5 : Base de données utilisée

```
--
-- Structure de la table `lora_arduino`
--
CREATE TABLE `lora_arduino` (
  `id` int(10) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `temperature` decimal(8,2) DEFAULT NULL,
  `humidity` decimal(8,2) DEFAULT NULL,
  `pressure` decimal(8,2) DEFAULT NULL,
  `uv_index` decimal(8,2) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- --------------------------------------------------------
--
-- Structure de la table `lora_arduino_chirpstack`
--
CREATE TABLE `lora_arduino_chirpstack` (
  `id` int(10) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `temperature` decimal(8,2) DEFAULT NULL,
  `humidity` decimal(8,2) DEFAULT NULL,
  `pressure` decimal(8,2) DEFAULT NULL,
  `uv_index` decimal(8,2) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- --------------------------------------------------------
--
-- Structure de la table `lora_esp32`
--
CREATE TABLE `lora_esp32` (
  `id` int(10) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `temperature` int(10) DEFAULT NULL,
  `humidity` int(10) DEFAULT NULL,
  `pressure` int(10) DEFAULT NULL,
  `uv_index` int(10) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `lora_arduino`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `lora_arduino_chirpstack`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `lora_esp32`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `lora_arduino`
  MODIFY `id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=221;
ALTER TABLE `lora_arduino_chirpstack`
  MODIFY `id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;
ALTER TABLE `lora_esp32`
  MODIFY `id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=154;
COMMIT;
```