Université de Mons
Faculté des Sciences
Département d'Informatique
SERVICE RESEAU ET TELECOMMUNICATION

Identifying IoT nodes by spying on their radio signal

Directeur : Mme Bruno QUOITIN

Rapporteurs : Mr Prénom NOM
Mr Prénom NOM

Mémoire réalisé par
Arnaud TULIPPE HECQ

en vue de l'obtention du grade de
Master en Sciences Informatiques

UMONS
Université de Mons

Faculté
des Sciences

Année académique 2023-2024

Remerciements

Nous remercions ...

Table des matières

Introduction **2**

1 Rappels et nomination des technologies **3**
1.1 Signal radio 3
1.2 Traitement du signal 4
1.2.1 Modulation 4
1.2.2 Gestion du bruit 8
1.2.3 Transformée de Fourier 8
1.3 LoRa 10
1.3.1 couche physique LoRa 11
1.3.2 LoRaWAN 14

2 Travaux similaires et autres contributions **20**
2.1 Identification d'appareils dans l'iot 20
2.1.1 historique et évolution des préoccupation de sécurité dans l'iot 20
2.1.2 Approches d'identification dans l'iot 23
2.2 Analyse de la technologie lora 24
2.3 identification de device lora 24
2.3.1 RFFI avec DCTFs 24

3 Expérimentations **26**
3.1 Matériel 26
3.1.1 radio logicielle 26
3.1.2 Module d'émission Lora 30
3.1.3 Logiciels 32
3.2 Librairies python 36
3.3 Géneration et réception d'un signal LoRa 37
3.3.1 Analyse avec GQRX 38
3.3.2 Analyse avec URH 39
3.3.3 Analyse avec matplotlib 43

3.3.4 Automatisation du signal et preprocessing 44
3.4 Méthode DCTF 46

4 Résultats **52**
4.1 Méthode des constellations Traces 52
4.1.1 paramètrage 52
4.1.2 training phase 52
4.1.3 testing phase 52
4.1.4 résultats 52

A Annexe A :Code module Arduino **53**

B Annexe B :Code module RN2483 **54**

C Annexe C :Implémentation émission réception automatisé **55**

Annexes **53**

Introduction

L'avènement de *L'Internet of Things* (IoT) a lancé une nouvelle ère d'appareils connectés, ouvrant de nouvelles possibilités de partage de l'information, d'automatisation et de protection. Bien que le concept lui même soit prometteur, la technologie qui l'accompagne est essentielle. Les premières technologies utilisées pour l'IoT étaient les technologies sans fil déja présentes comme le Wifi ou le Bluetooth. Elles ont cependant plusieurs limitations : une consommation en énergie élevée, une portée restreinte et parfois même un coût d'infractrucutre trop important. Dans ces circonstance est apparu *LoRa*, une technologie développée en particulier pour l'IoT. Sa capacité à gérer les communications longue portée même dans des environements urbain très dense est un grand atout pour le domaine.

L'expansion de L'IoT soulève une nouvelle problèmatique de sécurité. Entre autres, l'identification des noeuds au sein des réseaux est essentielle. Il a été découvert que des noeuds fabriqués avec les mêmes microprocesseurs et modèles d'émetteurs-récepteurs radio peuvent présenter de subtiles particularités dans les caractéristiques de leurs signaux. Cette variabilité intrinsèque de la transmission des signaux radio peut être exploitées pour distinguer les noeuds d'un réseau. En écoutant leurs signaux radio et en analysant leurs signatures distinctes, il devient possible de les identifier.

Ce travail est structuré en quatre parties. Le premier chapitre sert d'aperçu global du signal radio afin d'y développer et rappeler les concept de télécommunication de base. Ce chapitre présente également les technologies LoRa et LoRaWAN à travers leurs caractéristiques et leur pertinence dans l'IoT. Le second chapitre rassemble les travaux qui ont déja été effectués dans ce domaine. Le troisième chapitre est dédié à l'étude expérimental du sujet. Les aspects pratiques y seront appliqués, notamment l'utilisation de radio logicielle afin de capturer des signaux radio. Ces signaux seront ensuite analysés grâce à diverses méthodes détaillés dans ce chapitre. La dernière partie du travail présentera les résultats obtenu en suivant l'analyse effectuée au chapitre précédant. Enfin le travail sera achevé en concluant sur de potentielles implications plus larges à ce sujet ainsi que des recherches plus approfondies.

Chapitre 1

Rappels et nomination des technologies

1.1 Signal radio

Un signal est une variation dans l'espace ou dans le temps d'une quantité physique contenant de l'information. Un signal peut être continu ou discret, on le nomme alors respectivement analogique ou numérique. Le type de signal dépend notamment de l'information qu'il contient. Un signal analogique est continue en amplitude, ce qui veut dire qu'il peut contenir un nombre infini de valeur, ainsi que prendre toutes les valeurs possibles, là où un signal numérique contient généralement un nombre fini de valeur (par exemple des 0 et 1). Les deux catégories ne sont pas incompatible car il est souvent nécessaire en télécommication de pouvoir passer de l'un à l'autre.

L'utilisation de signaux radio en télécomunication confère de nombreux avantages, comme la portée, la vitesse de transmission ou encore le coût de propagation. Pouvoir transporter de l'information sans avoir recours à du support matériel complet (pas besoin de cable, le signal passe dans l'air) réduit considérablement le cout de la transmission. Ajouté à cela, il est possible d'adapter un signal pour le rendre compatible avec diverse canaux de transmission et de réception, grâce à la modulation. La modulation est une technique permettant de modifier les propriétés du signal lui permettant de transporter de l'information.

En télécommunication, les signaux sont des ondes électromagnétiques appelées signal radio. Les signaux comportent de nombreuses caractéristiques qui les déterminent :

la fréquence, mesurée en Hertz (Hz). Elle détermine le nombre de cycle qu'accomplit le signal par seconde. Une onde radio possède une fréquence entre 9kHz et 300GHz.

La largeur de spectre, elle dépent de la fréquence car c'est l'écart entre la plus haute et la plus basse fréquence du signal. Une plus grande largeur permet de transmettre plus d'informations, mais consomme plus d'énergie.

L'amplitude. Selon le type de signal l'attribut possède différentes fonctions. Dans le cas d'un signal analogique l'amplitude détermine la magnitude de l'onde pour n'importe quel point dans le temps. Dans un signal numérique l'amplitude est interprétée différemment. Les signaux numériques sont encodés avec des valeurs discrètes, où chaque valeurs repésente un niveau (par exemple 0 ou 1). L'amplitude permet de faire la disctinction entre ses niveaux.

la puissance, mesurée en Watt (W). C'est la force du signal, un attribut important pour la réception du signal notamment. Bien que le Watt soit utilisé pour décrire la puissance à l'émission ou la réception, les variations de puissances sont généralement exprimées en décibels (dB). Le décibel est une unité logarithmique permettant de mesurer plus facilement les realtions entre les différents niveaux de puissance.

le *Signal to Noise Ratio* (SNR). Cet attribut mesure la qualité du signal. une valeur élevée indique que le pourcentage de bruit est faible.

le *Bit rate*, ou le taux de transmission mesure la quantité de donnée transmise en bit par seconde. Cet attribut est exclusif aux signaux numériques. On parle de *Baud rate* pour mesurer la quantité de symboles transmise par seconde. Ce n'est pas excatement l'équivalent du bit rate car un symbole peut contenit plusieurs bits, mais le Baud rate est utilisé pour les signaux numériques et analogiques.

1.2 Traitement du signal

1.2.1 Modulation

La réception d'un signal radio nécessite une antenne dont les dimensions dépendent de la longueur d'onde du signal. La longueur d'onde d'un signal représente la distance entre deux points consécutifs de même phase dans l'onde. La longueur d'onde s'obtient par la formule suivante :

$$c = f * \lambda \tag{1.1}$$

où c est la vitesse de la lumière,
f est la fréquence,
λ est la longueur d'onde.

Il est donc possible d'adapter les charactéristiques d'un signal pour le rendre compatible à différentes antennes, via la modulation. La modulation est le procédé par lequel un ou plusieurs attributs du *baseband signal* (le signal modulant, contenant l'information à transmettre) vont être altéré par le *carrier signal* (un signal porteur, utilisé pour être combiné avec le signal modulant) pour devenir un signal modulé, le *modulated signal*.

En plus de sa compatiblité, un signal modulé a l'avantage d'être facilement transmissible sur une grand portée sans perdre en puissance.

Parmis ces différents attributs, certains sont utilisés pour effectuer une modulation. Les trois modulations les plus utilisés sont basées sur les attributs de la fréquence, l'amplitude et la phase. La modulation en fréquence (*Frequency modulation, FM*) consiste à encoder l'information en faisant varier la fréquence en maitenant l'amplitude constante. La modulation en amplitude (*Amplitude modulation,AM*) est le procédé inverse, c'est à dire encoder l'information en faisant varier l'amplitude tout en gardant la fréquence constante. La modulation en phase (*Phase Modulation, PM)* fait varier la phase de la porteuse proportionellement à l'amplitude instantannée du baseband signal.

La modulation en amplitude est plus ancienne et est encore utilisée dans beaucoup de systèmes. Cette technique possède moins de contrainte et est notamment plus simple à implémenter.

Soient $u(t)$ un baseband signal et $v(t)$ un carrier signal, la modulation en amplitude s'effectue en multipliant les deux signaux pour obtenir le signal modulé

$$s(t) = u(t).v(t) \tag{1.2}$$

Prenons par exemple
$u(t)$ = $sin(2\pi f_u t)$ avec $f_u t$ = 5 Hz,
$v(t)$ = $cos(2\pi f_c t)$ où $f_c t$ = 50 Hz.
La Figure 1.1 montre le signal modulé $s(t)$ via la modulation en amplitude.

FIGURE 1.1 – Exemple de modulation en amplitude

La modulation en fréquence permet d'obtenir des transmissions de meilleures qualités plus résistantes à leur environement tout en gardent une puissance d'émission constante.

Soient $u(t)$ un baseband signal et $v(t)$ un carrier signal, le signal modulé en fréquence $s_{fm}(t)$ est le résultat suivant :

$$u(t) = \sin(2\pi f_u t) \tag{1.3}$$
$$v(t) = \cos(2\pi f_c t + \phi_c) \tag{1.4}$$
$$s_{fm}(t) = \cos\left(2\pi f_c t + \Delta f \cdot u(t) \cdot t + \phi_c\right) \tag{1.5}$$

Prenons par exemple
$u(t)$ = $sin(2\pi f_u t)$ avec $f_u t$ = 5 Hz,
$v(t)$ = $cos(2\pi f_c t + \phi_c)$ où $f_c t$ = 50 Hz.

La Figure 1.2 montre le signal modulé $s_{fm}(t)$ via la modulation en fréquence pour une phase initale ddu carrier signal ϕ_c = 0 avec une dérivation en fréquence Δf = 10Hz.

FIGURE 1.2 – Exemple de modulation en fréquence

La modulation en phase permet généralement d'obtenir une meilleur utilisation de la bande passante que les autres modulations car les variations de phase peut encoder plus d'informations, ce qui augmente la quantité de données transmises.

Soient $u(t)$ un baseband signal et $v(t)$ un carrier signal, le signal modulé en phase $s_{pm}(t)$ est le résultat suivant :

$$u(t) = \sin(2\pi f_u t) \tag{1.6}$$
$$v(t) = \cos(2\pi f_c t + \phi_c) \tag{1.7}$$
$$s_{pm}(t) = \cos\left(2\pi f_c t + K_p \cdot u(t)\right) \tag{1.8}$$

Prenons par exemple

$u(t)$ = $sin(2\pi f_u t)$ avec $f_u t$ = 5 Hz,
$v(t)$ = $cos(2\pi f_c t + \phi_c)$ où $f_c t$ = 50 Hz.

La Figure 1.3 montre le signal modulé $s_{pm}(t)$ en phase pour une phase initale ddu carrier signal ϕ_c = 0 avec un index de modulation de phase K_p = 8.

FIGURE 1.3 – Exemple de modulation en phase

1.2.2 Gestion du bruit

L'un des attributs cités concerne le bruit. Un signal est toujours affecté de petites fluctuations plus ou moins importantes, et dont les origines peuvent être diverses. Ces perturbations, appelée bruit ou *noise* en télécommunication se définissent par l'altération non souhaitée de l'intégrité d'un signal. Le bruit peut prendre différentes formes, des perturbations essentiellement impulsionnelles engendrées par des commutations de courants ou alors du bruit de fond généré dans les câbles et les composants électroniques en raison des mécanismes statistiques de la conduction électrique. Il est possible de réduire voir éliminer l'influence des perturbations impulsionelles. En revanche, le bruit de fond est lui irreductible. Tout signal sans bruit n'existe pas, même à l'émission. Il est cependant possible que le bruit devienne invisible si son niveau est très faible. L'attribut SNR est donc un critère de la qualité du signal.

1.2.3 Transformée de Fourier

Pour effectuer une analyse de signal, sa représentation est capitale. Les Figure 1 et 2 représentent des signaux en fonction du temps écoulé. Il est possible de représenter des signaux selon une autre composante, la fréquence.

La transformée de Fourier est un outil fondamental utilisé pour analyser et décomposer des signaux complexes en composantes fréquentielles. En trans-

formant un signal dans le domaine temporel en sa représentation dans le domaine fréquentiel, la transformée de Fourier révèle les différentes composantes fréquentielles présentes dans le signal. En fonction du type de signal, la transformée de Fourier est adaptée.

Pour les signaux continus, la *CFT* (Transformée de Fourier continue) convertit une fonction du temps en fonction de la fréquence en intégrant le signal par rapport aux sinusoïdes de toutes les fréquences possibles. Cette transformation fournit les informations d'amplitude et de phase pour chaque composante de fréquence présente dans le signal.

Pour les signaux discrets et échantillonnés, la *DFT* (Transformée de Fourier discrète) calcule un ensemble fini de composantes de fréquence. Il est calculé à l'aide d'un nombre fini d'échantillons, ce qui donne des composantes de fréquence discrètes. Il existe un méthode simplifiée pour les signaux discrets appelé *FFT (Fast Fourier Transform)*[**?**]. Il s'agit d'un moyen plus rapide de calculer la transformée de Fourier, en particulier pour les signaux numériques comportant un grand nombre de points de données. L'avantage principal de cet algorithme permet de réduire le temps de calcul en divisant la DFT en sous problèmes. La FFT est une méthode très utilisée pour l'analyse de signaux.

1.3 LoRa

LoRa (Long Range) est une technologie de communication sans fil qui permet de transmettre des données sur de longues distances avec une faible consommation d'énergie. Elle a été développée par la société française Cycleo et est maintenant gérée par la fondation LoRa Alliance, qui regroupe plusieurs entreprises et organisations du monde entier.

LoRa est principalement utilisée dans l'IoT. Elle se distingue par sa portée étendue, qui peut atteindre plusieurs kilomètres en milieu urbain et plusieurs dizaines de kilomètres en milieu rural, ainsi que par sa faible consommation d'énergie, qui permet de prolonger la durée de vie des appareils connectés. Une longue portée avec un puissance limitée induit une plus faible bande passante que les autres technologies sans fil (le Wifi, la 4G, Bluetooth etc).

LoRa utilise une bande de fréquences qui varie selon les régions du monde où LoRa est déployée :

— en Europe, la bande de fréquences autorisée est comprise entre 863 et 870 MHz, ce qui correspond à l'*ISM radio band*, un bande dédié aux recherches qui ne nécessite pas de license d'émission.
— aux États-Unis, elle se situe entre 902 et 928 MHz,
— en Chine, la fréquence autorisée varie entre 779 et 787 MHz,
— les régions restantes ont elles aussi une fourchette unique.

La technologie LoRa utilise la modulation appelé *Chirp Spread Spectrum Modulation* (CSS). La modulation CSS utilise un signal chirp, c'est à dire un signal modulé en fréquence linéaire. Ce signal a une amplitude constante mais balaie tout le spectre de la bande passante de manière linéaire dans une période de temps définie. Cette technique de modulation sera détaillée à la section 1.3.1.2

La technologie LoRa utilise également une technique de multiplexage en temps partagé (*Time Division Multiple Access*) pour permettre à plusieurs appareils de partager la même bande de fréquences de manière à maximiser l'utilisation de la capacité de transmission. Elle utilise également une technique de diffusion de données (*multicast*) pour envoyer les mêmes données à plusieurs appareils simultanément, ce qui permet de réaliser des économies de bande passante et d'énergie (source : LoRa Alliance)

En plus de sa portée étendue et de sa faible consommation d'énergie, LoRa se distingue par sa sécurité de transmission, qui est assurée grâce à l'utilisation de codes de sécurité uniques et à la possibilité de chiffrer les données transmises. Lora n'est pas exclusivement lié au protocole LoraWan. Ce protocol sera décrit en détails à la section 1.3.2. Si LoRa opère à un niveau plus bas que la plupart des protocoles réseau, LoRaWan via son infrastructure (notament les *gateway*

permet entre autre aux appareils LoRa de pouvoir utiliser différents protocoles et d'être compatibles avec un grand nombre de protocoles de communications comme *TCP/IP (transport layer protocol), HTTP (hypertext transfer protocol ou MQTT(message queuing telmetry transport).*

Toutes ces particularités font de LoRa une technologie complémentaire à celles déja existantes plutot que rivale. LoRa se compose de deux éléments principaux : la couche physique de la technologie et LoRaWAN, la couche MAC (*Media Access Control*, une sous couche de la couche liaison de données dans le modèle OSI *Open Systems Interconnection*. la couche physique de LoRa gère la fréquence radio ainsi que la modulation. LoRaWAN gère les aspects réseaux comme la sécurité, la propagation, l'adressage et la sécurité.

1.3.1 couche physique LoRa

1.3.1.1 Découpage de la couche physique

FIGURE 1.4 – Etapes de la transformation des données dans un émetteur LoRa[**?**]

Les étapes de la conception de l'envoi de données dans la couche physique de LoRa sont montrés dans la figure 1.4 :

— Le codage de canal (*channel coding*) est une technique utilisée dans les systèmes de communication sans fil pour améliorer la robustesse et la fiabilité de la transmission des données. Dans le cas de LoRa, le codage de canal utilise le *Forward Error Correction (FEC)* pour corriger les erreurs causées par du bruit. La méthode FEC ajoute de l'information redondante sur les données.

— Le mélange de canal (*channel interleaving*) suit le codage de canal. Cette technique consiste à réarranger les bits ou les symboles de données en les dispersant sur plusieurs canaux (ici on fait référence à des streams ou a des bandes de fréquences spécifiques plutôt qu'à des canaux physiques). Cela permet de réduire l'impact de *burst errors*, des erreurs consécutives.

— Le blanchiment de canal (*channel withening)* est la dernière étape avant la modulation du signal. Cette technique consiste à utiliser une transformation aléatoire ou pseudo-aléatoire des données avant de les transmettre, de manière à répartir le spectre des fréquences de la transmission

sur une large gamme de fréquences. C'est une technique mathématique qui consiste a effectuer une transformation linéaire des données avec un matrice de covariance en un nouveau set de données dans la covariance est la matrice d'identitée. Le du blanchiment est de réduire la corrélation entre les différentes composante fréquencielles et assurer que le signal possède une puissance similaire tout le long de son spectre.

— La modulation CSS est l'étape principale de LoRa. En effet, les étapes précédentes sont communes à de nombreuses technologies, mais la particularité de LoRa provient de la modulation. Cette étape est détaillée dans la section 1.3.1.2.

Chacune des étapes décrites doit être inversément réalisée pour le récepteur. Ainsi pour la récupération de donnée à l'arrivée, l'appareil récepteur gère la démodulation, le déblanchiment, le démellement et de décodage.

Cette analyse a été faite en *Reverse Engeneering* par Alexandre Marquet, Nicolas Montavont et Georgios Z. Papadopoulos [**?**]. Le reverse engineering consiste à analyser un produit ou un système afin de comprendre comment il fonctionne ou d'identifier ses principes de conception. Dans le contexte de LoRa, le reverse engineering examine la technologie derrière LoRa afin de comprendre ses principes de base et sa conception.

1.3.1.2 Modulation CSS

Contrairement aux modulation classique en amplitude ou en fréquence, la modulation CSS étale le signal sur une large bande de fréquence. La modulation en fréquence est linéaire et utilise des chirps. un chirp est un signal dont la fréquence change en continue tout en conservant une amlitude constante. Il existe deux types de chirps : les $upchirp$ et $downchirp$. Dans un upchirp la fréquence augmente avec le temps tandis que dans un downchirp la fréquence diminue. Soit $s_{chirp}(t)$ un signal chirp avec

$$s_{chirp}(t) = sin(2\pi(f_0 + (\frac{f_1 - f_0}{T})t)t) \tag{1.9}$$

alors la figure 1.5 montre s_{chirp} en fonction du temps où f_0 = 10Hz, f_1 = 100Hz et T = 1 seconde. On observe que le signal oscille de plus en vite plus vite au fur et à mesure que le temps augmente.

Le signal est donc séparé sur une large bande de fréquence, permettant par exemple plusieurs transmissions sans causer d'interférence. La modulation CSS est l'une des principales contributions au fait que LoRa possède une faible consommation et une longue portée. Cette technioque est très bien intégrée aux appareils a faible puissance utilisé par LoRa.

FIGURE 1.5 – Example d'un signal modulé en upchirp en fonction du temps

1.3.1.3 Spreading factor

LoRa permet d'envoyer des paquets sur une longue distance à faible puissance. Selon l'environement dans lequel les appareils LoRa sont présents, il peut être utile de pouvoir ajuster certaines capacités.

Le facteur d'étalement (*Spreading Factor, SF*) permet de déterminer le taux de variation de fréquence pour un signal. Modifier le spreading factor ajuste différentes propriétés de la communications [**?**]. Par exemple, si on augmente le spreading factor, les quatre conséquences principales sont :

— l'augmentation de la portée. En effet augmenter le SF réduit le bitrate et augmente le *processing gain* (l'augmentation de la puissance du signal atteint en l'étalant sur une plus large bande).

— Augmentation de la résistance aux interférences. Comme le signal est étalé sur une bande plus largeur, il y a moins de risque de subit des interférences.

— Plus petit débit de données. Le spreading factor controle le taux de chirp, et du coup la vitesse de transmission de donnée. Augmenter le speading factor signifie ralentir la vitesse d'émission des chirps. Pour chaque augmentation du spreading factor, le taux de transmission de donnée est réduit de moitié.

— Plus faible consommation. Les données transmises à un taux plus faible consomment moins d'énergie, ce qui prolonge la durée de vie des appareils dont l'économie d'énergie est une priorité.

Diminuer le spreading factor engendre l'effet inverse [**?**].

1.3.1.4 Structure d'un paquet LoRa

FIGURE 1.6 – Structure d'un paquet Lora[**?**]

La figure 1.6 montre la structure d'un paquet Lora. Un paquet LoRa contient 3 parties différentes [**?**] :

— Le *preamble*. La première partie du paquet, composée d'un nombre variable d'upchirps. La valeur par défaut est fixée à 8 upchirps minimum. L'émetteur radio ajoute à cela un peu plus de 4 symboles (4.25), qui contiennet l'identificateur réseau ainsi que deux downchirps de synchronisation de fréquence. Ceci fixe le préambule à 12 symboles.

— Le *header* du paquet. Les informations sur la taille du paquet, le code rate, la présence d'un CRC (*cyclique redundancy check*) et la checksum sont incluses dans l'en-tête.

— Le *payload*. La dernière partie du paquet qui contient les données à transmettre. La taille maximale du payload est de 255 octets. En plus des données, le payload peut également contenir un CRC pour la détection d'erreurs. La longueur du CRC est généralement de 16 bits.

1.3.2 LoRaWAN

LoRaWAN est un protocol de type *Low Power Wide Area Network (LPWAN)* désigné pour la communication longue portée. Ce protocole opère avec la technologie LoRa et lui fournit une infrastructure capable de maintenir une communication à longue portée et à faible cout dans l'IoT.

1.3.2.1 Aspets généraux de la technologie

LoraWan bénéficie donc d'une faible puissance de consommation et d'une portée accrue. Elle est également efficace dans différents environements. Le signal est capable de pénétrer diverses terrains et structures. Le déploiment d'une infrastructure LoRa ne nécessite pas de license, et son réseau peut être public ou privé. Les caractéristiques générales de LoraWan sont disponibles sur le site The Thing Network.

Le coeur de LoRaWAN réside sur la gestion de l'énergie, permettant aux appareils de fonctionner avec une consommation d'énergie minimale, prolongeant leur durée de vie tout en garantissant une fonctionnalité à long terme. A cette caractéristique de faible consommation d'énergie s'ajoute ses capacités en termes de portée, capable de pénéter diverses environements. Cela rend la technologie efficace aussi bien milieu rurale qu'urbain. LoRaWAN opère sur une bande de fréquence qui ne nécessite pas de license d'émition, par exemple sur la bande ISM pour *Industrial, Scientific, and Medical*. Les bandes ISM, (868 MHz en Europe ou 915 MHz aux USA) sont disponibles pour l'utilisation de différentes technologies, incluant LoraWAN.

LoRaWAN possède des capacités de géolocalisation, permettant au réseau de détecter et de localiser précisément les appareils au sein de son domaine. LoraWAn utilise différentes méthodes pour localiser ses appareils comme *Received signal strengh indication (RSSI)*, *Time difference on Arrival (TDOA)*, une triangulation ou alors une combinaison de plusieurs des méthodes. Certaines de ses méthodes seront détaillées dans la section **??**

LoRaWAN utilise des protocoles de sécurité *end-to-end*, aussi bien dans un réseau public intégré que dans un réseau privé.L'architecture LoraWan (décrite en détails dans la section 1.3.2.2) contient plusieurs couches de sécurité. Au niveau des *end devices*, une routine d'identification est imposée avant l'accès au réseau. seul les appareils de confiance son donc autorisé à communiquer. Ensuite, une fois la communication commencée, les données sont chiffrées avant d'être transmise dans le réseau. Le framework sécuritaire de Lora ne se limite pas à l'autentification et au chiffrage. LoraWan gère également les mise à jour en continue par les airs, ainsi qu'une supervision continue sur d'éventuelles intrusions.

Avec toutes ces caractéristiques, LoraWan s'est développé dans de nombreux domaines aussi bien environementaux qu'industriels. Les principales utilisations de LoraWan actuelles sont les suivantes (toutes les applications ici) :

— la surveillance environementale en général [**?**]. Lorawan peut être déployé pour surveiller des niveaux de températures, d'humidité, de bruits ou encore d'autres paramètres dans n'importe quel milieu. Une compagnie Hollandaise, Sensoterra, utilise notamment LoraWan pour surveiller la qualité des sols.
— Les *smart cities* [**?**]. LoraWan est actif sur différents aspects comme la gestion intelligent de l'éclairage, la gestion des déchets, la surveillance, etc.
— l'embarqué industriel [**?**]. La maintenance et la surveillance de matériel et de l'équipement peut être gérée par Lorawan. TataSteel, une compagnie indienne, utilise LoraWan pour ces équipements industriels.
— la prévention de catastrophe naturelle. Que ce soit en prévision[**?**] ou

après[**?**] d'éventuelles catastrophes naturelles, la longue portée et la surveillance en temps réel sont des atoux cruciaux pour ce genre d'évènement.

Cependant, toutes ses caractéristiques entrainent un certain nombre de limitations. La restriction de la fréquence en fonction de la région peut rendre le déploiment d'une même infrastructure à différent endroit dans le monde plus difficile. Cela peut aussi entrainer des problèmes de compatibilité entre régions, notament pour des chaines logistique ou d'approvisionement qui en traverse plusieurs.

Une faible consommation de puissance avec une grande portée a un impact sur la taille et la vitesse de l'information. La taille du payload d'un message est limitée entre 51 et 241 octets. La vitesse de transmission est également peu élevée, atteignant un maximum de 5.5kbps sur une largeur de bande de 125kHz.

La communication au sein d'un réseau LoraWan se fait en grande partie de manière asynchrone. La synchronisation dépend de la classe de l'appareils, qui est détaillé dans la section 1.3.2.2. C'est un avantage pour maintenir une grande autonomie de batterie pour les appareils. LoraWan possède un système pour limiter les colisions entre messages si plusieurs appareils communiquent simultanénent. Ce système est basé sur une combinaison entre *Listen before talk (LBT)* et des delais aléatoires[**?**]. Il est néamoins possible que dans un environement très dense des collisions puissent encore se produire. La comunications asynchrone et le système d'évitement de collision entraine une augmentation du temps entre les envois et la réception de message.

1.3.2.2 Topologie de Lorawan

La figure 1.7 montre les 4 types d'appareils qui composent la topologie d'un infrastrucure LoraWan. Les end devices sont les noeuds qui collectent les informations à envoyer à travers le réseau. Ils sont catégorisés en trois sous classes : A, B et C. Les apareils de classe A sont les plus économes en énergie. Ils ont été créés pour conserver leur énergie et communiquent exclusivement en comunication asynchrone. Les appareils de classe A écoutent les messages provenant des serveurs uniquement après avoir eux-même transmis un message. la classe A regroupe les appareils les moins énergivores. Les appareils de classe B sont assez similaires à ceux de classe A, mais sont occasionellement synchronisés avec les serveurs du réseau. Ils possèdent des capacités supérieurs de reception leur permettant de se synchroniser avec le scheduler des serveurs, ce qui augmente considérablement l'efficacité du temps de réponses dans le réseau. Finalement, les appareils de classe C sont en écoute permanente de messages provenant des serveurs. Ils sont les plus réactifs mais également les plus énergivores. Les end devices sont donc classés selon deux paramètres : leur réactivité et leur consomation

FIGURE 1.7 – Topologie de l'infrastructure LoraWan (source : The Thing Newtork)

d'énergie. En fonction de leur classe, ils ont également la possibilité de recevoir des messages server après avoir transmis de l'information. L'envoie d'un message d'un end devices vers les serveurs est appelé *uplink message* et l'envoie d'un message depuis les serveur vers les end devices est appelée *downlink message*.

Les gateways jouent le role d'intermédiaire entre les end devices et le serveur réseau. Ils recoivent les transmissions depuis les end devices dans leur zone de couverture et forward les messages vers le serveur réseau. Les gateways peuvent écouter plusieurs fréquence simultanénent (*multichanneling*) là où les end devices n'écoutent qu'une seule fréquence. Les gateway gèrent la communication radio avec les end devices en utilisation la modulation de LoRa.

Le serveur réseau est la composante centrale de l'infrastructure. Il gère tout le réseau, que ce soit les données reçues des gateays, l'identification et l'activation des end devices dans le réseau, le routing ou encore l'adapation du data rate. Le serveur réseau supervise également l'aspect sécurité au sein du réseau en gèrant les clés de chiffrage et les protocoles de sécurité.

Le serveur application de LoraWan reçoit les données forwardées depuis le serveur réseau. C'est l'interface entre le réseau de LoraWan et les différents services ou applications d'utilisaturs finaux. Les utilisateurs intéragissent avec le serveur d'application pour n'importe quelle action a effectuer sur le réseau ou pour la récupération de données du réseau. Les données reçues par le serveur réseau sont traduites par le server d'application avant d'être interprétée par l'utilisateur

final.

1.3.2.3 Sécurité

La sécurité dans l'architecture LoraWan se concentre sur trois axe principaux :

— l'authentification : qui communique avec qui.
— L'intégrité : les données ne sont pas altérées entre l'émetteur et le recepteur.
— La confidentialité : les données ne sont visible par personne au sein du réseau hormi l'émetteur et le récepteur.

La sécurité repose sur le chiffrage des données. Les données sont chiffrées en utilisant l'algorithme de cryptographie AES (*Advanced Encryption Standard*). La taille des clés est de 128 bits. Ce choix est motivé par un équilibre entre une sécurité suffisante et une consommation réduite des ressources[**?**].

Il y a deux types de clés utilisées dans LoraWan. La *root key* est la clé partagée entre un end device et le serveur réseau. Cette clé est utilisée pour l'authentification initiale et l'établissement d'une communication entre deux éléments du réseau. Cette clé n'est jamais transmise par les air, elle est stockée dans un *join server*. Un join server est un server dédié au contenu sensible à l'activation du matériel dans un réseau LoRaWAN. Il autentifie le réseau et les application du servers. Il gère les root keys. il génère également le second type de clés de LoraWan, les *session keys*.

Les session keys sont des clés générées dynamiquement par le join server et utilisées durant l'échange de données pendant une session. Il y a deux session keys différente, la *AppSKEY* pour le chiffrage des payloads d'application, et la *nwkSKEY* pour les fonctionalités du réseau (le chifrage à la couche MAC, les vérification d'integrité, etc).

1.3.2.4 Session

L'établissement d'une session entre un end devices et le réseau LoraWan peut se faire de deux façons différentes.

La première méthode est une méthode dynamique appelée *Over the Air Activation(OTAA)* et se déroule de la façon suivante :

— Le device possède initialement deux indentificateurs, un DevEUI et un appEUI.
— La requête pour rejoindre le réseau est initiée par le end device. Il en envoir un message *join request* au serveur réseau. La join request contient ses identificateurs, ainsi qu'un nombre aléatoire généré par le device.

— Le serveur réseau accepte (ou décline) la requête et vérifie les identifiateurs du device dans ses enregistrements.
— Si la requête est acceptée, le server génère ensuite un nombre aléatoire appelé *DevNonce* et renvoie un message *join accept* contenant le DevNonce, l'adresse du device ainsi que les clés (NwkSKey et appSKey) de session.
— le end device reçoit le message join accept. Il extrait les clés envoyées et calcule ses propres clés de session avec ses paramètres (les clés envoyé par le join server ainsi que le devNonce).
— le device fait maintenant parti du réseau. Chaque message que le device va envoyer au serveur sera chiffré avec ses clés.

La seconde méthode est hardcodée et permet à un end device de rejoindre directement le réseau sans passer par l'indentification. cette méthode est appelée *activation by personalisation (ABP)*. Voici la procédure de la session :

— Le device possède à l'avance son adresse ainsi que ses clés de session.
— Le device est déploiyé au préalable dans la zone de couverture du réseau LoraWan.
— Sans devoir itinialiser de procedure *join request*, le end device transmet directement ses données au serveur en utilisant ses clés préconfigurées. L'échange de clés avec le serveur n'a pas lieu.

Cette seconde procèdure a comme avantage d'être plus rapide à exécuter car toute la partie d'initialisation est passée. Le processus d'initialisation peut être contraignant en ressource ce qui rend la méthode ABP moins énergivore. Cependant l'utilisation de clés hardcodées directement dans les devices est une pratique moins sécuritaire. Comme pour la taille de clés, il y a un équilibre entre consommation d'énergie et sécurité.

Chapitre 2

Travaux similaires et autres contributions

Ce chapitre a pour but d'établir un état de l'art dans le domaine de l'internet of things. Comme l'iot est très vaste, seulement certains aspects seront présentés. Tout d'abord, il est important de comprendre que la sécurité dans ce domaine a évoluée autan bien par ces méthodes que par son importance, ainsi un historique est présenté dans la section2.1.1 de ce chapitre. Ensuite, la seconde section se concentre uniquement sur la technologie Lora. Cette section donne une approche analytique des signaux. Finalement, la dernière partie de ce chapitre se consacre aux approches existantes pour l'identification d'appareils utilisant Lora. Cette section présentera les travaux qui sont au centre des expérimentations de ce travail.

2.1 Identification d'appareils dans l'iot

Avante de s'intéresser à l'identification d'appareils Lora, il est nécessaire d'étendre ce concept à l'iot. L'internet of things a connu une forte évolution depuis les début des années 2000, avec des priorités dans son évolution qui ont également changées au fil du temps.

2.1.1 historique et évolution des préoccupation de sécurité dans l'iot

Afin de bien comprendre les enjeux des différentes époques, voici une petit historique des deux décénnies précedantes. Les technologie mais également le concept même d'internet of thing ont évolué.

Les premières années de l'iot (*début des années 2000*).

Bien que le concept d'appareils connectés remonte aux années 70, l'avènement de l'internet of thing arrive en fin de milénaire. Ce concept est associé à la technologie RFID *Radio Frequency Identification*[**?**], qui permet d'utiliser les ondes radios afin d'identifier des objets ou des personnes. Le but initial était de rendre tout objet dans le monde identifiable par un code EPC ou *Electronic Product Code*[**?**], un peu comme le code barre. Durant ses années, plusieurs entreprises lancent leur première appareils connectés. Tout cet enthousiame pour la connectivité met au second plan les questions de sécurité. Ainsi la première partie du développemnt de l'iot se concentre surtout sur la qualité de communication entre les objets plutot que sur leur sécurité.

Les premières préoccupations sécuritaires (*fin des années 2000, début des années 2010*)

Vers la fin des années 2000, l'augmentation du nombre d'appareils est si grande qu'elle a atteint tous les domaines de la société. Certains domaines étant plus critique que d'autre d'un point de vue sécurité (l'énergie, les transports, la santé, etc),l'intégrité des données, la confidentialité et les accès réseaux deviennent le centre de l'attetion. Le concept de certificats x.509, initalement développé pour le world wide web avant les années 2000, a un regain d'attention dans cette période. Plus largement, la structure de la technologie PKI (*Public Key infrastructure*, qui utilise les certificats x.509)[**?**] a été adaptée pour s'intégrer aux problématiques de l'embarqué. Un certificat est un document digital permettent de vérifier d'identité d'une entité, comme d'un appareil, un utilisteur ou une organisation. Il se base sur la liaison d'une clé public à l'entité établie par une *Certificate Authority (CA)*. La CA agit en temps de tier de confiance et assure la légitimité de l'information grâce au certificat. Ainsi, les trois axes principaux de la sécurité dans l'IoT émergent : l'autentification, l'intégrité des données et la confidentialité.

L'avènement du *Edge Computing*(à partir des années 2010).

Le nombre d'appareils connecté dépasse le nombre d'êtres humains, forçant une transition vers l'*ipv6* tant le nombre d'appareils est évelé et continue d'augmenter. L'information a pris de la valeur, et de l'ampleur. Ainsi, viens se gréffer de nouveau enjeux économiques en plus des enjeux sécuritaires. La quantité de données générées nécessite de revoir le stockage de l'information. C'est ainsi que va apparaitre le Edge computing[**?**], qui est un réponse directe aux besoin des architectures de gérer autan de données en périphérie de réseau. Le concept du edge computing vise à effectuer des calculs et des analyses des données directement sur les appareils connectés, plutôt que de les envoyer vers un centre de données centralisé. Cela réduit la latence, améliore l'efficacité du réseau et permet des analyses en temps réel. Le premier malware spécialemnt centré sur l'iot fait son apparition. Mirai[**?**] exploite une faille lui permettant de récupérer le mots de passe d'appareils afin de s'en servir pour lancer des attacks *DDoS (distribu-*

ted denial of service) à grande échelle. En quelques années l'IoT est passé d'un gadget d'entreprise à un vériable enjeu économique et sécuritaire, centré autour de l'information. Les seules perspectives de législation concernant la sécurité de l'internet of thing n'apparaitront de tradivement en fin de décennies avec La loi européenne sur le *Réglément générale sur la protection des données*(source : loi RGPD. cette loi ne couvre pas la sécurité des appareils mais plutot l'utilisation des données sur internet en général.

L'apparition de la *Blockchain*(fin année 2010).

Le stockage et la transmission de données, l'authentification d'appareils ou encore la confidentialité sont au centre des préoccupations. Initialement utilisée dans les cryptomonnaie, la technologie Blockchain est un mécanisme de base de données qui permet un partage transparent des informations au sein d'un réseau. Une base de données Blockchain stocke les données dans des blocs qui sont reliés entre eux dans une chaîne. Les données sont chronologiquement cohérentes, car il n'est pas possible de supprimer ou modifier la chaîne sans le consensus du réseau. Par conséquent,la technologie Blockchain peur servir de livre inaltérable ou immuable pour le suivi des ordres, des paiements, des comptes et d'autres transactions. Le système dispose de mécanismes intégrés qui empêchent les entrées de transactions non autorisées et créent une cohérence dans la vue partagée de ces transactions. L'implémentation de la blockchain pour l'iot confère les avantages suivant[**?**] :

- l'immuabilité. la blockchain permet de créer un enregistrement immuable de toutes les interactions et communications des appareils. Cet enregistrement peut être utilisé pour détecter et empêcher l'accès non autorisé ou la modification des appareils ou des données dans l'IoT.
- La décentralisation. il est possible de créer un système décentralisé pour l'authentification et la communication des appareils. Chaque appareil IoT se connecte au réseau blockchain et se voir attribuer une identité numérique unique, qui est vérifiée grâce à l'utilisation de signatures numériques ou de contrats intelligents. Cela élimine le besoin d'une autorité centrale pour authentifier les appareils et annule ainsi les risques de *single point of failure*.
- La confidentialité.La technologie Blockchain peut sécuriser la communication entre les appareils IoT grâce à l'utilisation de la cryptographie à clé publique ou asymétrique. Cela permet l'échange sécurisé d'informations entre appareils sans avoir recours à des intermédiaires.

Le *Zero Trust Model*[**?**](début des années 2020).

les menaces de sécurité sont de plus en plus sophistiquées. Comment faire encore confiance aux infrastructures qui doivent gérer autan d'appareils ? La réponse est de ne plus leur faire confiance. Le zero Trust model est donc un modèle basé

sur l'absence totale de confiance et une vérification constante, que la demande d'accès provienne de l'intérieur ou de l'extérieur du réseau. Dans un modèle de sécurité classique, une fois qu'un utilisateur ou un appareil accède au réseau interne, on lui fait souvent implicitement confiance, ce qui lui permet une grande liberté d'actions au sein du réseau. Le Zero Trust model suppose cependant que des menaces peuvent exister à la fois à l'intérieur et à l'extérieur du périmètre du réseau et nécessite donc une vérification continue de la confiance. Un modèle qui s'applique sur ce principe devrait contenir les éléments suivants[**?**] :

— La vérification d'identité. Les utilisateur et les appareils doivent subir une authentification avant d'accèder à n'immporte quel services ou ressources du réseau.
— Le *Least privilege acces*. les permissions sont accordées de manières limités selon le besoin de l'utilisateur ou de l'appareil.
— La micro segmentation. Diviser le réseau en segments pour limiter son accès par les appareils.
— La surveillance en continue. L'analyse du traffic, du comportement et des activités des appareils.
— le chiffrage des données.

Le *quantum computing*(dans les prochaines années...).

Avec la future arrivée des ordniateurs quantiques, les méchanismes de chiffrage basés sur la comlexité mathématique comme RSA ou ECC sont voués à disparaitre[**?**]. La puissance de calcul des ordinateurs quantique est déja considérée comme une véritable menace pour la sécurité informatique. Fort heureusement, c'est également un nouveau champ de possibilité qui s'ouvre pour la sécurité, avec le développement du *post quantum cryptography*. Un premier protocol résistant au menances quantiques, *Quantum Key Distribution* permet d'établir des canaux de communications entre différents appareils dand l'iot. Ce protol n'est pas encore en service dans l'iot, mais les premiers est réalisé en laboratoire sont très prometteurs[**?**].

2.1.2 Approches d'identification dans l'iot

Comme tout appareil au sein d'un réseau, un appareil connecté dans l'iot possède de base différents moyens d'authentification. Son adresse MAC, son adresse IP, des informations relatives à sa manufacturation comme un numéro de série par example. De manière plus spécifiques, les noeuds au sein d'un réseau LoraWAN possède un DevEUI, un numéro unique accordé par le réseau à l'appareil. Cependant ses informations peuvent être compromises si appareils sont victimes de *devices spoofing*, c'est à dire qu'un appareil malveillant usurpe l'indentité de sa cible, afin d'accéder au sein du réseau. D'autres attaques comme *Man in The*

Middle ou *Replay attacks* peuvent également compromettre l'identité d'un appareil si on se base uniquement sur ses identifiant classiques[**?**]. Il faut donc pouvoir identifier les appareils mais sans se fier à leur informations. Il éxiste diverse méthodes basée sur différentes approches pour pouvoir indentifier un appareil.

La première approche possible est de fier non pas à l'appareil directement ni aux données qu'il reçoit car celles ci pourraient également être compromises, mais à la routine sur sa communication. Cette approche, appelée *Traffic related pattern* où s'intéresse au comportement d'un appareil au sein d'un réseau. L'article publié par H. Kawai, S. Ata et N. Nakamura [**?**] propose notament d'analyser via machine learning le *traffic pattern* c'est à dire le comportement du traffic.

Une autre approche s'intéresse à la position géographique d'un appareil. En effet il est possible de mesure la qualité de la réception d'un signal, le *Received Signal strengh Indicator* ou RSSI. Un étude a été menée sur des appareils lora par M. Anjum, MA. Khan et SA. Hassan[**?**]. L'article utilise le *Path Loss* pour estimer le RSSI. Via machine learning ils sont capables de recréer un système de positionement permettant l'identification d'appareil Lora.

La dernière approche se concentre sur l'analyse des charactéristiques uniques des fréquences radios. Plutot que de s'intéresser au routine entre les communication ou la distance d'où elles ont lieu, le *Radio Frequency Finguerprinting* se concentre sur les propriétés physiques des signaux. Différentes techniques sont montrées par N. Soltanieh, Y. Norouzi et Y. Yang[**?**].

2.2 Analyse de la technologie lora

2.3 identification de device lora

radio frequency finguerprinting identification (RFFI). trouver des charactéristiques hardware pour identifier des devices.

2.3.1 RFFI avec DCTFs

objectif, identification du device via son frequecy offset.

selon l'article (citer article), possible de performer la méthode soit uniquement sur le préambule, soit sur l'intégralité du singal.

idée : le received singal contient le baseband singal ainsi qu'un rotation factor instable. pour pouvoir recover cette partie du signal, besoin d'effctuer une opération différentielle suivante :

$$x(t).x(t+n)e-j2\pi on \tag{2.1}$$

apparation d'un nouveau rotation factor, mais stable. Besoin de trouver deux inconnue,*delta* f et n. n est le differential interval. il se calcule de la manière suivante :

$$Rs = BW/2sfN = fs/Rs \tag{2.2}$$

carrier frequency offset :

$$CFO \tag{2.3}$$

Chapitre 3

Expérimentations

3.1 Matériel

3.1.1 radio logicielle

La radio logicielle *(Software Defined Radio, SDR)* est une technologie qui permet de mettre en œuvre des systèmes de radio à l'aide de logiciels plutôt que de matériels.

Dans les systèmes de radio traditionnels, les différentes fonctions de la radio, comme l'accord sur une fréquence spécifique, la modulation et la démodulation du signal, et le filtrage du bruit, sont mises en œuvre à l'aide de composants matériels tels que des oscillateurs, des amplificateurs et des filtres. En revanche, les systèmes SDR utilisent des logiciels pour effectuer ces fonctions, ce qui les rends beaucoup plus flexible car chaque composante est reconfigurable. Les radios logicielles sont capables d'opérer sur une large portée de fréquences, aussi bien très basse fréquence comme haute fréquence. Les SDR peuvent jouer le rôle d'émetteur ou de récepteur ou les deux. Différentes radios logicielles sont testées pour ce travail, ce qui est motivé pour plusieurs raisons. Tout d'abords cela permet d'observer certaines variations (comme la qualité du signal, la distance) entre les différents récepteurs pour une même expérience et ainsi définir la SDR la plus adéquate. Ensuite, la diversité de radios logicielles permet d'approfondir l'apprentissage de cette technologie et de comprendre plus aisément le fonctionnement d'une radio logicielle. Une version plus simple permet d'assimiler les bases de la pratique avec une SDR, mais pouvoir utiliser des version plus abouties permets d'expérimenter de manières plus précise les expériences.

3.1.1.1 RTL SDR dvb-T

La *RTL SDR dvb-T* est la version la moins chère des radios utilisées. Cette radio est initialemnt utilisée pour recevoir des signaux dvb-T (*Digital video broadcasting Terrestrial* télévisés. La figure 3.1 montre la radio, qui est un device USB qui inclu un *chipset RTL2832U* et un *RF tuner chip*. Le chipset RTL2832U digitalise les signaux RF et les evnoie à l'ordinateur. Le tuner permet d'ajuster la fréquence pour couvrir une larger portée. La radio est raccordé à l'ordianteur via le port USB.

FIGURE 3.1 – RTL SDR Dvb-T dongle

3.1.1.2 RTL-SDR R820T2

La deuxième radio logicielle utilisée est la *RTL SDR R820T2*. Il y a deux différences majeures avec la radio dvb-T. L'antenne de cette radio logicielle est de meilleure qualité et le tuner chip de cette SDR est un r820t2 tuner. Cette version est une version améliorée du tuner qui se trouve dans la dvb-T, ce qui a pour impact une réduction du bruit, une meilleure sensibilité et une couverturee de fréquence plus large. La figure 3.2 montre la SDR en question.

FIGURE 3.2 – RTL SDR R820T2

3.1.1.3 HackRF One

La dernière radio logicielle utilisée est la *HackRF One*. Au dela du prix plus élevé, cette dernière radio est différente des deux autres. Elle est notament capable de gérer la transmission et la réception de signaux, là ou les deux autres sont uniquement des récepteurs. Si la R820T2 offrait déja une qualité de signal supérieure à la dvb-T, celui de la hack rf est encore plus net. La figure 3.3 montre la SDR avec l'antenne utilisée posée sur un trepied.

(a) SDR HackRF One (b) Antenne

FIGURE 3.3 – SDR HackRf One avec son antenne

3.1.2 Module d'émission Lora

3.1.2.1 Module RN2483

FIGURE 3.4 – 3 modules RN2483

Le *microchip RN2483* est un module de technologie spécifique à LoRa. Bien que Microchip soit une compagnie qui produit de nombreux microcontrolleurs, cet appareil n'en est pas un. Le module RN2483 de la figure 3.4 est un simple *transceiver (transmitter and receiver)* radio permetant de communiquer à longue portée et à faible coup. La documentation sur cet appareil est disponible au lien suivant : https://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf.

Voici quelques spécificités sur le module :

— Il comprend la technologie de modulation LoRa, ce qui lui donne son atout de faible consommation et de longue portée. Il gère également les modulations *FSK (frequency shift keying)* et *GFSK (Gaussian frequency shift keying)*.
— Une faible consommation induit une faible puissance, le module possède un amplificateur de puissance maximale de 14dBm.
— Les bandes de fréquences disponibles sont compatibles avec la bande ISM. Le module couvre de 863Mhz à 870Mhz pour la région européenne.
— Le data rate maximum en modulant avec LoRa est de 10937 bps (bit par seconde).

— tous les paramètres TX sont configurables.

L'utilisation du module RN2483 est détaillée dans la section 3.3.

3.1.2.2 Module Pycom lopy

FIGURE 3.5 – Un module Pycom lopy avec une antenne

Le module *Pycom lopy* est un appareil programmable en micro python. Il comprend un microcontrolleur ainsi qu'une série de pins digitaux input/output, des composantes de connectivité sans fil et un port micro USB. La figure 3.5 montre le module connecté à une antenne.

3.1.2.3 Module Arduino

L'avantage principal du module *Arduino* est qu'il est possible de configurer une largeur de bande bien plus faible que les autres modules. Les modules RN2483 ou lopy ont une largeur de bande minimal de 125KHz, tandis qu'avec

FIGURE 3.6 – Un module Arduino

l'arduino il est possible de descendre jusqu'à quelques Khz à peine. Pour rappel, une largeur de bande plus faible permet un *time on air (TOA)* plus grand, ce qui est très utile pour analyser les signaux.

3.1.3 Logiciels

Une fois les différents appareils choisis, il faut les accompagner avec les softwares adéquats. Ainsi il faut des logiciels d'analyse de signaux compatibles avec les différents modules et radios logicielles.

3.1.3.1 GQRX

Le premier logiciel choisi est *GQRX*. c'est un logiciel open source d'analyse de fréquences radios pour les SDR. Le fonctionnement du logiciel est assez direct, la figure 3.7 montre comment configurer simplement le logiciel. La figure 3.7a demande de compléter les paramètres suivants :

— **Device** : il faut sélectionner la SDR dans la liste des appareils. Le *Device String* fait référence à cet appareil dans GQRX.

— **Input rate** : le taux d'échantillonage capturé par la SDR, il apparait plus bas comme *sample rate*.

— **Decimation** : c'est un procédé qui permet de réduire le nombre d'échantillons gérés par le logiciel, pour économiser des ressources.
— **Bandwidth** : la largeur de bande supplémentaire que peut gérer la SDR.
— *Low Noise Block Local Oscillator (**LNB LO**)* : ce paramètre est spécifique à la réception satéllite et répresénte le frequency offset aplliqué par l'oscillateur locale dans un LNB downconverter.
— L'**Audio output** concerne la sortie sonnore du signal.

La figure 3.7b ajoute des paramètes pour le récepteur, c'est à dire pour l'interface de GQRX :

— **frequency** : la fréquence que l'on souhaite écouter.
— **Filter Width** : ce paramètre permet de changer la taille de la largeur de bande du filtre appliqué au signal.
— **Filter Shape** : contrairement au Filter Width ce n'est pas la taille mais la forme du filtre que l'on peut modifier.
— **Mode** : le mode de démodulation.
— **Automatic Gain Control (ACG)** : l'ACG ajuste le gain du récepteur pour maintenir un signal constant.
— **Squelch** : c'est un threshold qui coupe la sortie sonore s'il est franchi.
— **Noise blanker** : c'est une fonctionnalité qui permet de réduire l'impact des interférences sur le signal.

(a) SDR input

(b) Receiver options

FIGURE 3.7 – Configuration de GQRX

La figure 3.8 montre un exemple de ce qu'on observe quand on écoute la

fréquence radio de Classic21. GQRX affiche le signal reçu sous deux formes différentes : en spectre et en cascade.

FIGURE 3.8 – Réception de la station radio Classic21

L'affichage du spectre fournit une représentation graphique en temps réel du spectre RF sur une gamme de fréquences. Il montre la puissance du signal de différentes fréquences sur une plage de fréquences spécifiée. L'axe horizontal représente la fréquence, tandis que l'axe vertical affiche la force du signal (mesurée en dB).

L'affichage en cascade est un spectrogramme qui visualise la force du signal au fil du temps. Il montre une série de spectres instantannés empilés les uns sur les autres, où l'intensité de la couleur représente la force du signal. Chaque ligne horizontale du tracé en cascade représente une vue du spectre capturée à un moment précis, créant ainsi un enregistrement de l'activité du signal. L'axe horizontal représente la fréquence et l'axe vertical représente le temps.

Durant la réception du signal, il est possible de modifier l'affichage. A droite de l'affichage en spectre et en cascade sur la figure 3.8, il y a différents paramètres :

Pour l'affichage en spectre, le paramètre **Panadapter dB** fait référence à l'échelle verticale. Il représente la force du signal des fréquences radio reçues affichées sur l'axe vertical du graphique du spectre. Le réglage du paramètre Pa-

nadapter dB modifie l'échelle verticale de la force du signal affichée. Pour l'affichage en cascade, le paramètre **Waterfall dB** concerne l'intensité de la couleur ou l'ombrage des fréquences affichées dans le tracé en cascade. Le réglage du paramètre Waterfall dB modifie l'intensité utilisée pour afficher la force du signal, permettant ainsi d'ajuster le contraste ou la visibilité des signaux plus faibles ou plus forts. Les paramètres suivants sont communs et affectes les deux affichages :

Le paramètre **FFT size** détermine le nombre d'échantillons utilisés dans chaque calcul de la FFT. Une valeur plus large donne un meilleure résolution, mais consome plus de ressources. La valeur est une puissance de deux pour des raisons d'éfficacité de calculs.

Le paramètre **Rate** détermine le taux de rafraichissement de l'affichage. Un taux relativement faible avec une taille de frame FFT élevé induit un effet d'overlap, c'est à dire que des frames consécutives partagent des échantillons, ce qui donne un affichage plus lisse du spectre.

3.1.3.2 Universal radio hacker, URH

Universal Radio Hacker est un logiciel open source similaire à GQRX. Son role principal est l'analyse de signaux radio. Au dela de l'écoute en temps réel, URH peut sauvegarder des signaux et lire des enregistrements à partir de fichiers. Les fichiers qu'URH peut interpréter ou enregistrer sont au format *.complex*. Ce format est utilisé pour pour sauvegarder les signaux sous forme d'échantillons *I/Q (In phase / Quadrature)* Il existe plusieurs variantes de ce format supporté par URH :

- *.complex (ou .complex64)* : c'est le format par défaut. Les échantillons sont représenté par des float de 32bits pour les symboles I et Q respectivement (donc 64 bits au total)
- *.complex16u* : la représentation en utilisant 2 entiers de 8 bits non signés pour I et Q.
- *.complex16s* : la représentation en utilisant 2 entiers de 8 bits signés pour I et Q.
- *.complex32u* : la représentation en utilisant 2 entiers de 16 bits non signés pour I et Q.
- *.complex32s* : la représentation en utilisant 2 entiers de 16 bits signés pour I et Q.

La figure 3.9 montre comment enregistrer un signal. Les paramètres à configurer sont similaires à GQRX (le choix de la SDR, le fréquence, le taux d'échantillonage, la largeur de bande et le gain). Comme URH est le principal outil d'analyse pour ce travail, les analyse avec ce dernier sont détaillées dans la section 3.3.2.

FIGURE 3.9 – Enregistrement d'un signal avec URH

3.1.3.3 Visual Studio Code

L'environement de travail utilisé est Visual Studio Code (version 1.87.1). Cet IDE est open source et largement populaire ce qui permet un apprentissage pour des travaux spécifique très rapide via Internet. L'utilisiation de VSCode est motivé par son système d'extension, qui contient plusieurs extension qui permettent de manipuler directement les modules d'émission LoRa. Le module Lopy de Pycom est compatible avec VSCode via l'extension PyMakr. GitHub est également intégré dans VSCode, ce qui permet de simplifier la mise en ligne du travail.

3.1.3.4 IDE Arduino

L'utilisation de l'IDE Arduino 2.2.1 est nécessaire pour la configuration du module Arduino décrit dans la section 3.1.2.3. L'IDE permet d'uploader le code dans le module. Le code est disponible dans l'annexe A.

3.2 Librairies python

Le choix du langage pour l'implémentation du travail est Python (version 3.11.4). Python donne l'accès à plusieurs librairies très pertinentes pour faire du *signal processing*. Voici les principales librairies utilisées :

— *Numpy*. Cette librairie est fondamentale pour la gestion des *arrays*. Elle contient également de nombreuses fonctions et formats mathématiques utiles au projet.
— *Matplotlib*. Cette librairie permet d'effectuer des plots des données de manières simple et efficace.
— *PyRTLSDR*. La librairie PyRTLSDR est essentielle pour pouvoir manipuler la SDR R820T2. Il est possible de configurer la SDR sans devoir passer par un software tier comme GQRX ou URH grâce à cette librairie.
— *PyHackRf*. L'équivalent de PyRTLSDR mais pour la SDR HackrF One.
— *Datashader*. Cette librairie permet la visualisation améliorée de grande quantité de données via l'ajout de gradient coloré.
— *Serial*. La manipulation des modules rn2483 est possible grave à la librairie serial. Cette librairie permet d'intéragir avec les apparails connecté via USB.

3.3 Géneration et réception d'un signal LoRa

La première étape de l'expérimenation consiste a générer via un module un signal LoRa et à le capturer à l'aide d'une radio logicielle. Pour cette première étape, le module d'émission choisi est le module RN2483, car il est le plus facile et rapide à configurer. Via python, il est possible d'utiliser la librairie *Serial* pour se connecter au port usb reliant le module à l'ordinateur. Ensuite, via les différentes commandes, on configure le module en modifiant les paramètres suivants :

— la modulation, en LoRa.
— La fréquence. 868MHz, la fréquence de la bande ISM , la bande d'émission en Europe.
— La largeur de bande choisie est de 125Khz, ce qui est la plus petite valeur possible que permet le module.
— La puissance du module, au maximum (14W).
— Le spreading factor, la valeur la plus grande possible est choisie (SF = 12).
— Le coding rate. Il y a 4 valeur possible :s 4/5, 4/6, 4/7 and 4/8. Cela signifie que tout les 4 bits seront codé par 4, 5, 6, 7 ou 8 bits de transmissions en fonction de cette valeur. Plus la valeur est faible (la plus faible étant 4/8), plus le TOA sera élevé, car cela prend plus de temps pour transmettre le message.

Le choix des valeurs pour les différents paramètres est de maximiser le TOA. Les commandes relatives à la configuration sont disponibles dans l'annexe B

3.3.1 Analyse avec GQRX

Une fois que le module est configuré, il faut paramètrer le recepteur, la radio logicielle. La SDR choisie est la RTL SDR T820R2. Voici les paramètres de la SDR :

— la fréquence. La fréquece d'écoute, idéalement la même fréquence de celle de l'émetteur (868MHz).
— Le taux d'échantillonage. Il est possible de choisir la quantité d'échantillons traitée chaque seconde. Un taux plus élevé donnera un signal plus complet. Le taux minimal ne doit jamais être inférieur à deux fois la largeur de bande (Théorème de Nyquist-Shannon).
— Le gain. Dépendant de la qualité du signal il peut être nécessaire d'ajouter du gain, c'est à dire d'amplifier la force du signal. Un gain trop élevé peut saturer le signal, c'est à dire quand l'amplitude dépasse la portée du récepteur.

La figure 3.10 montre la capture de deux signaux LoRa. On remarque que malgré la configuration des paramètres pour maximier le TOA, il est difficile d'observer en détails le signal. On observe également que le signal se situe entre la fréquence 867,915MHz et 868,040MHz, ce qui correspond bien à la largeur de bande de 125Khz (0.125MHz) choisie.

FIGURE 3.10 – capture module rn2483 sur GQRX avec RTLSDR R820T2. Sample rate = 1.8MHz

Le module Arduino, contrairement aux modules Pycom et RN2483, permets d'utiliser une largeur de bande beaucoup plus faible. La figure 3.11 montre la capture d'un signal LoRa émis depuis le module Arduino. La largeur de bande choisie est de 7.8Khz, les autres paramètres d'émissions (la modulation, la fréquence et le spreading factor) sont similaires à ceux de la capture précédente. Cette fois ci, on distingue clairement sur l'affichage en spectre un pic à 868,070MHz. Sur l'affichage en cascade, le signal se dessine sous forme de traits (les chirps). Cela correspond bien à la théorie de la modulation LoRa où le signal modulé est composé de upchirps et downchirps. Il y a de part et d'autre du signals des silhouettes. C'est du à la sensibilité de la SDR, si on observe les pics de fréquences dans l'affichage en spectre, on constate qu'ils sont négligeables car leur échelles de puissances est bien inférieure (environ 60dB plus faible) à celle du pic principal. GQRX permet de visualiser le signal, mais pour pousser l'analyse plus en détails d'utilisation d'Universal Radio Hacker est nécessaire.

FIGURE 3.11 – capture module Arduino sur GQRX avec RTLSDR R820T2 Sample rate = 1.8MHz

3.3.2 Analyse avec URH

GQRX permet de zoomer sur les fréquences mais pas sur le temps, ce qui demande donc de générer un signal dont le TOA doit être le plus élevé possible. URH offre plus de flexibilité sur l'analyse ce qui permet de ne pas devoir utiliser

des largeurs de bande aussi faibles que celles du module Arduino. Ainsi, pour l'analyse avec URH, les modules Pycom et RN2483 sont utilisés. La figure 3.12 et 3.13 montre la capture et la sauvegarde d'un signal LoRa. Le module utilisé est rn2483, la SDR est la R820T2. les paramètres d'émission sont les suivants : BW = 125KHz, SF = 8, freq = 868MHz, Mode = LoRa, cr = 4/8, pwr = 14. Les paramètres du récepteurs sont les suivants : frequency = 868MHz, SR = 1.8MHz, gain = 0dB. Pour des raisons de sipmlicité, le signal capturé avec ses paramètres sera utilisé comme référence pour toute la partie analyse jusqu'à la fin de la section 3.3.

FIGURE 3.12 – Capture du signal

FIGURE 3.13 – Sauvegarde du signal

A partir de la figure 3.13, il est possible de couper une partie de l'enregistrement, notamment celle qui ne contient pas le signal. URH permet également d'afficher le signal sous forme de spectrogramme, la figure 3.14 montre le signal sous sa forme analogique et sous spectrogramme. On constate cependant que les deux affichages ne correspondent pas. En effet si on se fie au spectrogramme, la fréquence devrait augmenter durant le premier chirp, or quand on zoom sur le signal sur l'affichage analogique, celle ci diminue dans un premier temps. (voir figure 3.15).

FIGURE 3.14 – Signal LoRa sous forme spectrogramme et analogique

FIGURE 3.15 – Zoom d'un upchirp du signal LoRa capturé

Ce phénomène est due à une interprétation faussée par URH. La fréquence

du récepteur choisie étant la même que celle de l'émetteur, URH va interpréter 868Mhz comme étant la fréquence centrale ayant pour valeur 0, ce qui signifie que pour un signal reçu à 868Mhz, la moitié des échantillons sont interprétés comme ayant une fréquence "négative". Autrement dit, URH affiche la position inverse des échantillons qu'il considère comme négatifs. Pour éviter cette erreur, il suffit de décaler la fréquence d'écoute proportionnellement à la moitié de la taille de la largeur de bande du signal émis. Dans ce cas, le signal émis à une largeur de bande de 125Khz, il aut ajuster la fréquence d'écoute :

$$F_{recepteur} = F_{emetteur} - \frac{BW}{2} soit : \tag{3.1}$$

$$868Mhz - \frac{125KHz}{2} = 867,9375MHz \tag{3.2}$$

Selon la section 1.3.1.4, la structure du paquet LoRa devrait contenir d'abord les symboles du préambule, soit 8 upchirps suivis de 4.25 chirps. La figure 3.16 montre le spectrogramme du signal complet, dont la partie fixe du préambule (les 8 upchirps par défaut) ont été mis en évidence sous forme analogique. Pouvoir identifier et récupérer le préambule du signal LoRa est crucial pour l'identification de l'appareil qui l'a émis.

FIGURE 3.16 – Identification du préambule LoRa

Cependant, l'utilisation d'intermédiaire comme URH ou GQRX n'est pas efficace pour traiter un grand nombre de signaux. La dernière étape avant de commencer l'identification des noeuds à partir des signaux est d'automatiser le processus complet de la capture du signal.

3.3.3 Analyse avec matplotlib

Une information importante que les deux logiciels de visualisation ont occultée, c'est la composition du signal. En effet, la visualisation analogique du signal avec URH montre une seule onde continue. Cependant il s'agit d'un signal complexe composé de deux deux ondes distinctes. Les échantillons du signal capturés par la SDR sont composé de deux composantes : la composante *I (In phase)* et la composante *Q (Quadrature)*. La composante I représente l'amplitude du signal à un point spécifique du temps, cela correspond à la partie réelle du signal. La composante Q représente le déplacement de phase du signal en fontion de I, ce qui correspond à la partie imaginaire du signal. URH combine les deux pour générer la vue analogique.

La librairie *Matplotlib* de python permet d'afficher le contenu du signal capturé. La figure 3.17 est un plot des échantillons dont la partie réelle (orange) et imaginaire (bleue) ont été séparées. Le signal a été capturé avec un taux d'échantillonnage de 1.8MHz, malgré un TOA relativement court (quelques centièmes de secondes) la quantité d'échantillons capturés est très grande (plus de 130000 échantillons). Matplotlib permet également le zoom, la figure 3.18 fait un zoom sur les 4000 premiers échantillons. Les deux ondes apparaissent grâce au zoom.

FIGURE 3.17 – Plot des échantillons I/Q avec matplotlib

FIGURE 3.18 – zoom sur les 4000 premiers échantillons I/Q

3.3.4 Automatisation du signal et preprocessing

Afin de pouvoir effectuer l'identification des appareils, il faut que l'environement de test soit identique pour chaque module, donc tous les paramètres d'émissions sont identiques. La même SDR est utilisé avec les mêmes paramètres de réception pour les mêmes raisons. Avec des paramètres d'émission et de réception fixes, il est possible d'automatiser le processus d'enregistrement d'échantillons LoRa.

La librairie PyRTLSDR permet de configurer la radio logicielle depuis un script python. La partie du signal que l'on cherche à conserver est le préambule (soit les 8 premiers chirps fixes ou les 12.25 premiers chirp). Si le taux d'échantillonage et fixe et que le préambule est toujours de la même taille, alors il est possible de calculer le nombre d'échantillons à capturer.

Par exemple, la figure 3.19 indique qu'un préambule complet (12.25 chirp) contient environ 49000 échantillons. Sachant que le taux d'échantillonnage est de 2MHz pour cette example, alors le TOA du préambule vaut :

$$TimeOnAir(TOA) = \frac{N_{samples}}{samplerate} = 0.0245 \tag{3.3}$$

FIGURE 3.19 – Préambule LoRa (sampel rate = 2MHz)

Maintenant que la durée de l'enregistrement est calculée, il reste à fixer le moment à partir duquel l'enregistrement commence. La méthode la plus facile est d'implémenter un threshold qui détermine si un signal est reçu durant l'écoute. Voici en résumé comment toute l'opération se déroule :

— Les paramètres de l'émetteur sont configurés.
— Dès que l'émetteur est prêt à envoyer un signal, le récepteur commence à écouter.
— le récepteur informe l'émetteur que il est en écoute, l'émetteur envoie un signal.
— Grâce au threshold, le récepteur commence à enregistrer le TOA du signal dès que le threshold est franchi. L'émetteur est en attente pendant que le signal est enregistré dans un fichier.
— Le récepteur se remet sur écoute et informe l'émetteur qu'il peut envoyer un nouveau signal.

Finalement, il reste une dernière étape avant de pouvoir commencer l'identification des modules. Malgré que les prises d'échantillons soient en intétieur avec une faible distance entre l'émetteur et le recepteur (voir figure 3.20), les signaux sont parfois fortement atténués. Il faut donc normaliser les données pour contrer les différences d'amplitude. Les données sont normalisée en utilisant *Root means square (RMS)*.

L'implémentation de l'automatisation et de la normalisation est disponible en annexe C

FIGURE 3.20 – Configuration en intérieur

3.4 Méthode DCTF

L'analyse suivante est basée sur l'article publié par Yu Jiang, Linning Peng, Aiqun Hu, Sheng Wang, Yi Huang et Lu Zhang [**?**]. L'objectif de cette analyse est de pouvoir identifier un appareil LoRa uniquement en se fiant à l'analyse des signaux générés par ce dernier.

Le signal test pour cette analyse est le signal contenant les paramètres suivants :

— émetteur : module RN2483, modulation : LoRa, SF = 8, BW = 125KHz, frequency = 868MHz, pwr = 14, cr = 4/8.
— Récepteur : RTL SDR R820T2, f = 867,9375MHz, SR = 2MHz, gain = 5dB.

Ce signal a été normalisé avec *Root Means Square*.

La méthode DCTF se base sur l'utilisation de constellations traces pour pouvoir identifier sur base de propiétées uniques un appareil. Un diagramme de constellations est une représentation dans le plan complexe de la distribution spaciale des points du signal. La figure 3.21 montre la représentation dans le temps du signal test et sa représentation sous forme de constellation est donnée par la figure 3.22.

Selon la section **??**, le diagramme de constellation seul n'est pas suffisant pour pouvoir identifier des composantes uniques au signal. Pour pouvoir obser-

FIGURE 3.21 – LoRa signal

ver l'émergence d'une signature, il faut appliquer la méthode différentielle décrite dans 2.3.1 ainsi que dans l'article [**?**]. La figure 3.23 montre le diagramme differentiel de constellation *DCTF* du signal. L'équation 2.1 possède deux paramètres à calculer. Le *différential interval (N)* se calcule via 2.2 et vaut 4096. Le *Carrier Frequency Offset (CFO)* peut être estimé via 2.3. Dans un premier temps sa valeur est estimée à 0.

On remarque que la forme de la constellation est similaire, mais l'application de la méthode différentielle juxtapose les points les uns sur les autres à tel point qu'il devient difficile d'analyser en détail sa composition. Pour pouvoir observer une composante susceptible d'être une signature, il faut appliquer un gradient coloré pour évaluer la densité des points de la constellations. La figure 3.24 montre le même diagramme qu'à la figure 3.23 mais avec l'utilisation de la librairie python Datashader, qui ajoute une échelle de densité (en poucentage, ou 100 représente la zone la plus dense du diagramme). On observe que dans le coin supérieur droit la densité de point est un peu plus élevé que dans le reste de la constellation.

Jusqu'à présent, l'analyse a été faite en utilisant l'intégralité du signal comme donnée. Cependant l'article [**?**] a montré qu'il est possible de filtrer une partie des données et ainsi ne conserver qu'une partie suffisante du signal pour déterminer sa signature. Premièrement, d'un point de vue physique, le signal possède une partie appelée *transient part*, c'est la portion initiale du signal qui contient la transition d'un état vers un autre. Cette partie se caractérise sur la figure 3.24 par les points qui ne se situent pas sur la constellation mais entre la constellation et le centre du

FIGURE 3.22 – Diagramme de constellation

plot. Dans ce domaine, cette partie concerne le moment ou le module s'enclenche, occasionant des irrégularités dans la réception des données. Cette partie seule étant instable, elle est écarté des données analysées. Ensuite, l'intégralité du signal n'est pas nécessaire. En effet, le contenu du message (dans la partie payload du paquet LoRa) est sujet à modifications et n'est pas pertinent pour l'analyse, seule la partie incluant le préambule est conservée. Ainsi, du signal complet on ne conserve que le préambule (12.25 chirps) auquel on coupe la *transient part* au début des données. La figure 3.25 permet de distinguer clairement la région dense dans le coin supérieur droit après avoir supprimé les données jugées non pertinentes.

Maintenant que la région d'intérêt est indetifiée, il faut l'extraire. C'est ce qui sera extrait de cette DCTF qui sera la signature du signal, et donc permettra l'identification du module. Pour déterminer la meilleure valeur dans le plan à récuperer, la méthode suivante permet de récupérer le centre de la zone dense. La librairie Numpy de python permet de créer un histogramme en deux dimension de la DCTF. Afin d'extraire la valeur la plus pertinente (le centre de la zone dense), le point le plus dense de l'histogramme sert de référentiel. Sa valeur de densité est calculée (c'est à dire le nombre d'échantillons présent dans cette zone définie par l'histogramme). Afin de mieux refléter le centre de la zone dense, tous les points ayant une valeur de densité au moins égale à 90 pourcent (valeur choisie dans

FIGURE 3.23 – DCTF du signal test

[**?**]) de la zone la plus dense sont également considérés. La figure 3.26 montre les points sélectionnés pour le signal test dans la DCTF. Le centre euclidien est calculé à partir des points éligibles, sa valeur est la signature du signal.

FIGURE 3.24 – DCTF du signal test

FIGURE 3.25 – DCTF du signal test

FIGURE 3.26 – Points de haute densité dans la DCTF du signal test

Chapitre 4

Résultats

4.1 Méthode des constellations Traces

4.1.1 paramètrage

3 modules lora
module rn2483
module pycom fipy
module arduino
paramètres d'émission : SF =7, BW = 125, Freq = 868Mhz, CR = 4/5, mod = Lora
plot des DCTF (differential constellation traces figure)

4.1.2 training phase

during the training phase, you use differential constellation trace figures (DCTFs) from different modules. The objective is to identify clusters in these figures and learn the characteristics of each module.

4.1.3 testing phase

In your case, during the testing phase, you have new DCTFs from modules that the model has not seen during training. The model uses what it learned to predict the category or cluster for each DCTF.

4.1.4 résultats

Annexe A

Annexe A :Code module Arduino

```python
# This is a Python function
def my_function(x, y):
    """
    This function computes the sum of two numbers.

    Parameters:
    x (int): The first number.
    y (int): The second number.

    Returns:
    int: The sum of x and y.
    """
    return x + y

# Example usage of the function
result = my_function(3, 4)
print("The result is:", result)
```

Listing A.1 – Example Python code

Annexe B

Annexe B :Code module RN2483

```python
# This is a Python function
def my_function(x, y):
    """
    This function computes the sum of two numbers.

    Parameters:
    x (int): The first number.
    y (int): The second number.

    Returns:
    int: The sum of x and y.
    """
    return x + y

# Example usage of the function
result = my_function(3, 4)
print("The result is:", result)
```

Listing B.1 – Example Python code

Annexe C

Annexe C :Implémentation émission réception automatisé

```python
# This is a Python function
def my_function(x, y):
    """
    This function computes the sum of two numbers.

    Parameters:
    x (int): The first number.
    y (int): The second number.

    Returns:
    int: The sum of x and y.
    """
    return x + y

# Example usage of the function
result = my_function(3, 4)
print("The result is:", result)
```

Listing C.1 – Example Python code