Université de Mons

Faculté des Sciences

Identifier des noeuds IoT en espionant leur signal radio

Mémoire réalisé par
Arnaud TULIPPE HECQ

Département d'Informatique
SERVICE RESEAU ET TELECOMMUNICATION
Directeur : Mr Bruno QUOITIN

Année académique 2023-2024

Remerciements

Nous remercions ...

Table des matières

Introduction **2**

1 Rappels et nomination des technologies **6**
1.1 Signal radio 6
1.2 Traitement du signal 8
1.2.1 Principe de la modulation 8
1.2.2 Modulation en amplitude 9
1.2.3 Modulation en fréquence 10
1.2.4 Modulation en phase 11
1.2.5 Gestion du bruit 13
1.2.6 Transformée de Fourier 13
1.3 LoRa 16
1.3.1 Couche physique de LoRa 17
1.3.2 LoRaWAN 22

2 Expérimentations **28**
2.1 Matériel 28
2.1.1 radio logicielle 28
2.1.2 Module d'émission Lora 32
2.1.3 Logiciels 34
2.2 Librairies python 38
2.3 Géneration et réception d'un signal LoRa 39
2.3.1 Analyse avec GQRX 40
2.3.2 Analyse avec URH 41
2.3.3 Analyse avec python 45
2.3.4 Automatisation du signal et preprocessing 46

3 Identification d'appareils LoRa par la méthode des constellations traces figures **49**
3.1 Fadio Frequency Fingerprinting avec DCTF 49
3.2 Méthode DCTF en pratique 50

3.3 Autres approches d'identification dans l'iot 53

Conclusion **56**
.1 Annexe A :Code module Arduino 59
.2 Annexe B :Code module RN2483 61
.3 Annexe C :Implémentation de la configuration RTL SDR via PyRTLSDR 62
.4 Annexe D :Implémentation de l'automatisation des captures de signaux 63

Annexes **59**

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 trois parties. Le premier chapitre sert d'aperçu global du signal radio afin d'y développer et rappeler les concepts de base. Ce chapitre présente également les technologies LoRa et LoRaWAN à travers leurs caractéristiques et leur pertinence dans l'IoT. Le deuxième 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 et automatisés pour la suite du travail. Le troisième chapitre du travail présentera la méthode utilisée pour réalisé l'objectif du mémoire, ainsi que son application pratique sur les appareils décrits au chapitre précedent. Enfin le travail sera achevé en concluant sur de potentielles implications plus larges à ce sujet ainsi que des recherches plus approfondies.

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*[31], 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*[28], 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)[8] 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[21], 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[22] exploite une faille lui permettant de récupérer le mots de passe d'appareils afin de s'en servir pour lancer des attacks *DDoS (distributed 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[10] :

- 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*[16](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[20] :

— 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[25]. 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[9].

Chapitre 1

Rappels et nomination des technologies

1.1 Signal radio

Les concepts suivants peuvent être retrouvés dans le chapitre 2 du livre *The Scientist and Engineer's guide to Digital Signal Processing*[23] ainsi que dans le cours de *Communications analogiques et numériques* donné par le professeur Véronique Moeyaertà l'université de Mons.

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 continu 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 incompatibles car il est souvent nécessaire en télécommunication de pouvoir passer de l'un à l'autre.

L'utilisation de signaux radio en télécommunication 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 câble ni de fibre optique, le signal passe dans l'air) réduit considérablement le cout de la transmission. 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. Ses ondes comportent de nombreuses caractéristiques qui les déterminent :

— la fréquence. Elle est mesurée en *Hertz (Hz)* et 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 dépend 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. C'est la force du signal, mesurée en *Watt (W)*, 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 exprimé en décibel mesure la qualité du signal. Le SNR peut se calculer de la façon suivante :

$$SNR(dB) = 10log_{10}\frac{P_{signal}}{P_{noise}} \tag{1.1}$$

où P_{signal} et P_{noise} sont respectivement la puissance du signal et du bruit. Une valeur élevée indique que le pourcentage de bruit est faible.

— le *Bit rate*. C'est le taux de transmission, une mesure de 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 Principe de la modulation

Les différentes approches de modulations et les choix de notations ont été repris du livre *An introduction to analog and digital communication* de S. Haykin[2].

La réception et l'émission d'un signal radio nécessitent 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 :

$$c = f\lambda \tag{1.2}$$

où c est la vitesse de la lumière, f est la fréquence et 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 signal en bande de base *(baseband signal)* soit 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. La combinaison des deux devient alors un signal modulé, le *modulated signal*. Les trois modulations les plus utilisés sont basées sur les attributs de la fréquence, de l'amplitude et de la phase.

La modulation est au final un décalage du signal en bande de base sur une porteuse de fréquence plus élevée. Cette relation peut s'exprimer de la façon suivante :

$$m(t) = Re\{g(t).e^{-j2\pi f_c t}\} \tag{1.3}$$

où $m(t)$ représente le signal modulé , $g(t)$ représente l'enveloppe complexe du signal en bande de base, f est la fréquence porteuse et $e^{-j2\pi f_c t}$ est le signal porteur dans sa forme complexe. $Re()$ signifie qu'on s'intéresse à la partie réelle de l'équation. Le signal porteur peut être représenté autrement (formule d'Euler) :

$$e^{-j2\pi f_c t} = \cos(2\pi f_c t) + j\sin(2\pi f_c t). \tag{1.4}$$

$m(t)$ peut donc être exprimé par la relation suivante :

$$\begin{aligned} m(t) = \mathrm{Re}\{&x(t)\cos(2\pi f_c t) + jx(t)\sin(2\pi f_c t) \\ &+ jy(t)\cos(2\pi f_c t) - y(t)\sin(2\pi f_c t)\} \end{aligned} \tag{1.5}$$

$$m(t) = x(t)\cos(2\pi f_c t) - y(t)\sin(2\pi f_c t) \tag{1.6}$$

car l'enveloppe complexe $g(t)$ peut être exprimée selon ses composantes *In Phase* et *Quadrature*. Dans le plan complexe, la composante *In phase (I)* est représentée par l'axe des réels, tanids que la composante *Quadrature (Q)* est représentée sur l'axe imaginaire. Un nombre complexe z peut être représentée selon ces composantes tel que :

$$z = I + jQ. \tag{1.7}$$

Ainsi, l'enveloppe complexe d'un signal modulé $g(t)$ peut être interprétée comme la somme de ses composantes formulées comme 1.7 c'est à dire :

$$g(t) = I(t) + jQ(t) = x(t) + jy(t). \tag{1.8}$$

On peut donc interpréter *I* comme étant aligné avec le cosinus dans le plan complexe, et *Q* aligné en quadrature avec le cosinus.

1.2.2 Modulation en amplitude

La modulation en amplitude (*Amplitude modulation, AM*) est plus ancienne et est encore utilisée dans beaucoup de systèmes. Cette technique consiste à encoder l'information en faisant varier l'amplitude en maitenant la fréquence constante. Elle possède moins de contrainte et est notamment plus simple à implémenter que les deux autres modulations citées. En effet on peut représenter le signal modulé $m(t)$ de la façon suivante :

$$m(t) = s(t).\cos(2\pi f_c t) \tag{1.9}$$

où s(t) est le signal en bande de base $\cos(2\pi f_1 t)$. le signal modulé est équivalent à :

$$m(t) = U_c(t) = U_c.\cos(2\pi f_c t + \theta_c) \tag{1.10}$$

car par l'indentité trigonométrique :

$$sin(a)cos(b) = \frac{1}{2}sin(a+b) + sin(a-b). \tag{1.11}$$

U_c est donc l'attribut de l'amplitude à faire varier pour la modulation AM.

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

$$s_{\mathrm{AM}}(t) = u(t).v(t) \tag{1.12}$$

Prenons par exemple
$u(t)$ = $\cos(2\pi f_u t)$ avec f_u = 5 Hz,

$v(t) = \cos(2\pi f_c t)$ où $f_c = 50$ Hz.

La Figure 1.1 montre le signal modulé $s\mathrm{AM}(t)$ via la modulation en amplitude.

FIGURE 1.1 – Exemple de modulation en amplitude

1.2.3 Modulation en fréquence

La modulation en fréquence (*Frequency modulation, FM*) encode les informations dans les caractéristiques temporelles du signal transmis, ce qui la rend plus robuste aux interférences liés à l'amplitude que la modulation AM. La fréquence d'un signal ne peut pas être modifiée par le bruit ou la distorsion. Cepdendant, d'autres types de distortions comme le *frequency drift* (un changement non désiré de la fréquence dans le temps) peuvent affecter la qualité d'un signal modulé en fréquence. Selon l'équation 1.10, le paramètre à faire varier est f_c. Contrairement à la modulation en amplitude, le signal porteur et modulé ne sont pas similaires ($m(t) \neq U_c(t)$). Dans 1.10, la phase $\phi(t)$ vaut $2\pi f_c t + \theta_c$ où θ_c est une constante. Pour déterminer la fréquence instantanée, il faut dériver la phase instantanée c'est à dire :

$$f_i(t) = \frac{1}{2\pi}\frac{d\phi_i(t)}{dt} \tag{1.13}$$

Ainsi, il est possible de représenter l'accumulation de la phase due à la mo-

dulation par l'équation :

$$U_c(t) = U_c.\cos(2\pi \int_0^t f_i(\tau)\, d\tau + \theta_c) \tag{1.14}$$

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

$$u(t) = \sin(2\pi f_u t) \tag{1.15}$$
$$v(t) = \cos(2\pi f_c t + \phi_c) \tag{1.16}$$
$$s_{fm}(t) = \cos\left(2\pi f_c t + \Delta f \cdot u(t) + \phi_c\right) \tag{1.17}$$

Reprennons le même signal en bande de base $u(t)$ = $sin(2\pi f_u t)$ avec f_u = 5 Hz et la même porteuse $v(t)$ = $cos(2\pi f_c t + \phi_c)$ où f_c = 50 Hz.

La Figure 1.2 montre le signal modulé $s_{fm}(t)$ via la modulation en fréquence pour une phase initale du carrier signal ϕ_c = 0 avec une dérivation en fréquence Δf = 5Hz. On remarque la présence d'une constante représentant un décalage de phase. La modulation en fréquence n'est pas détachée de la modulation en phase. Les deux modulations fonctionnent à partir de la phase instantannée $\phi_i(t)$. La modulation en fréquence fait varier linéairement la fréquence instantanée $f_i(t)$ (qui dépent de $\phi_i(t)$ par 1.13) tandis que la modulation en phase fait varier linéairement la phase instantannée $\phi_i(t)$. Ajouté à cela, $u(t)$ est un sinus, or

$$\int_0^t \sin(2\pi f \tau\, d\tau) = \frac{1 - \cos(2\pi f t)}{2\pi f} \tag{1.18}$$

Ce qui implique que le signal modulé a subi un décalage de $\frac{\pi}{2}$ radian car l'identitée trigonométrique indique que :

$$\cos(x + \frac{\pi}{2}) = \sin(x) \tag{1.19}$$

1.2.4 Modulation en phase

La modulation en phase (*Phase Modulation, PM)* permet généralement d'obtenir une meilleur utilisation de la bande passante que les autres modulations car les variations de phase peuvent encoder plus d'informations, ce qui augmente la quantité de données transmises. La modulation en phase est associée à celle en fréquence. En effet les concpets sont liées par leur dépendance à $\phi_i(t)$. Peu importe l'argument de la fonction de modulation choisi, il n'est plus une fonction linéaire du temps.

FIGURE 1.2 – Exemple de modulation en fréquence

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

$$u(t) = \sin(2\pi f_u t) \tag{1.20}$$
$$v(t) = \cos(2\pi f_c t + \phi_c) \tag{1.21}$$
$$s_{pm}(t) = \cos\left(2\pi f_c t + K_p \cdot u(t)\right) \tag{1.22}$$

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

On constate que pour un signal sinusoidal il est assez difficile de différencier la modulation en fréquence de celle en phase. La différence a lieue à l'emplacement des variations de la vitesse d'oscillation. Pour la modulation en fréquence, on constante que l'oscillation accélère jusqu'à atteindre le maximum quand l'amplitude du signal en bande de base est maximale, et inversément quand l'amplitude est au minimum. Les concepts de modulation de phase et de frénquences sont généralement regroupés sous le terme de modulation angulaire (source : http://www.telecom.ulg.ac.be/teaching/notes/total1/elen008/node39mn.html)

FIGURE 1.3 – Exemple de modulation en phase

1.2.5 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.

1.2.6 Transformée de Fourier

Les notations sont également reprises du livre *An introduction to analog and digital communication* de S. Haykin[2].

Pour effectuer une analyse de signal, sa représentation est capitale. Les Figure 1 et 2 représentent des signaux en fonction du temps écoulé, soit dans le domaine temporel. Il est possible de représenter des signaux selon une autre composante, la fréquence, c'est à dire dans le domaine fréquenciel.

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

Pour les signaux continus, la *CFT* (Transformée de Fourier continue ou juste Tranformée de Fourier) 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. La Transformée de Fourier Continue peut être calculée de la manière suivante :

$$G(\omega) = \int_{-\infty}^{\infty} g(t)e^{-j\omega t}dt \tag{1.23}$$

où : $G(\omega)$ est la Transformée de Fourier du signal $g(t)$. ω représente la fréquence angulaire $(2\pi f)$. Il est également possible de revenir dans le domaine temporelle, grâce à la Transformée de Fourier inverse (IFT) :

$$g(t) = \int_{-\infty}^{\infty} G(f)e^{j2\pi ft}df \tag{1.24}$$

Les deux équations 1.23 et 1.24 sont les complexes conjugués l'une de d'autre.

Reprenons la modulation en amplitude de la section 1.2.1. En utilisant la CFT sur les trois signaux (bande de base, porteuse et modulé) de la figure 1.1, la figure 1.4 montre leurs CFT respectives.

La première chose que l'on constate, c'est que pour le singal en bande de base et la porteuse, on observe pour chacun d'entre eux deux pics. Ces pics correspondent à la fréquence des différents signaux (pour rappel f_u = 5Hz et f_v = 50Hz) mais aussi à leur fréquence négatives. Cette duplication est due au fait que la Tranformée de Fourier produit un spectre symétrique par rapport à l'origine (dans notre cas, le centre où la fréquence vaut 0Hz). La CFT d'un signal réel possède deux valeurs (positives et négatives) pour chaques composantes du signal. On constate également que sur la CFT du signal modulé il y a 4 pics. Ces pics sont expliqués par un décalage en fréquence du à la modulation. En effet le troisième signal est un produit de deux signaux, dans le domaine fréquenciel ce produit apparait comme un décalage de la fréquence du signal en bande de base de part et d'autre de la fréquence du signal de la porteuse. C'est le théorème de la modulation. Pour des signaux $u(t)$ et $v(t)$ ayant pour CFT respectives $U(f)$ et $V(f)$, alors leur relation peut être exprimée de la manière suivante :

$$u(t)v(t) = \int_{-\infty}^{\infty} U(f)V(t-f)df \tag{1.25}$$

FIGURE 1.4 – Exemple de CFT

Ainsi, par l'identité trigonométrique 1.11, le signal modulé possède un pic à 50 - 5 Hz et à 50 + 5 Hz selon le signal en bande de base. Comme le signal est réel, les mêmes pics sont observés à l'opposé du centre de la symétrie.

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 optimisée pour les signaux discrets appelé *FFT (Fast Fourier Transform)*[13]. Il s'agit d'un moyen plus rapide et moins couteux de calculer la transformée de Fourier, en particulier pour les signaux numériques comportant un grand nombre de points de données. La DFT calcule la transformé de Fourier pour une séquence de N échantillons en $O(N^2)$ tandis que la FFT optimise le temps de calcul en $O(NlogN)$ pour la même séquence. Il existe diverses variante de la FFT (par example le Cooley-Tukey FFT[7]) ce qui peut légèrement faire varier les performances dépendant de l'algorithme utilisé. Les logiciels utilisés pour l'analyse de signaux dans la section 2.1.3 utilisent des algorithmes de FFT pour leurs affichages dans le domaine fréquenciel.

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 consom-

mation 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. Toutes les informations de base concernant LoRa sont disponibles sur le site LoRa Alliance (https://lora-alliance.org)

FIGURE 1.5 – Spectre des technologies sans fil

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. Lora fait partie des technologies appelées *LPWAN (Low Power Wide Area Network)*. La figure 1.5 montre le spcetre des technologies sans fil actuelles. Cette image est reprise du document *A comparative study of LPWAN technologies for large-scale IoT deployment*[18]. Une longue portée avec un puissance limitée induit un débit de transmission plus faible 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*, une bande dédiée 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 si-

gnal modulé linéairement en fréquence. 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 est 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 : https://resources.lora-alliance.org/technical-trainings/lorawan-device-to-device-multicast-communications)

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 est 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 *gateways*) 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 de LoRa

1.3.1.1 Découpage de la couche physique

FIGURE 1.6 – Etapes de la transformation des données dans un émetteur LoRa[26]

Les étapes de la conception de l'envoi de données dans la couche physique de LoRa sont montrés dans la figure 1.6. Cette analyse a été faite en *Reverse Engeneering* par Alexandre Marquet, Nicolas Montavont et Georgios Z. Papadopoulos [17]. 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. Les principales étapes de la conceptions de la couche physique LoRa sont les suivants :

— le blanchiment de canal (*channel withening)*. 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.

— 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 *Gray mapping*. Le code de Gray est une méthode d'ordonnancement de symboles. Le principe de cette méthode est que dans la séquence ordonnée, chaque symbole ne diffère du précedent que d'un seul bit. Utiliser le code de Gray permet de réduire les erreurs dues au bruit ou aux interférences.

— 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 démo dulation, le déblanchiment, le démellement et de décodage.

1.3.1.2 Modulation CSS

Le principe de la modulation LoRa est détaillé en profondeur dans l'article *Frequency Shift Chirp Modulation : The LoRa Modulation*[30] de Lorenzo Vangelista.

La modulation LoRa repose sur le principe de *Frequency Shift Chirp Modulation (FSCM)*, qui est une combinaison de la modulation *Frequency Shift Keying (FSK)* et *Chirp Spread Spectrum (CSS)*. La modulation CSS étale le signal sur une large bande de fréquences tandis que la modulation FSK décale la fréquence périodiquement dans le temps. Le signal modulé est composé de 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 $upchirps$ et $downchirps$. 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

$$s_{chirp}(t) = sin(2\pi(f_0 + (\frac{f_1 - f_0}{T})t)t) \tag{1.26}$$

alors la figure 1.7 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.

FIGURE 1.7 – Example d'un upchirp

Supposons que la bande passante du canal soit β, LoRa impose qu'un échantillon soit transmis tout les $T = \frac{1}{\beta}$. Un symbole $s(nT_s)$ est envoyé a l'entré de la modulation tout les $T_s = 2^{SF}$ où *SF (spreading factor)* est le facteur d'étalement. le symbole $s(nT_s)$ est un nombre réel formé en utilisant un vecteur de chiffres binaires du facteur d'étalement. Les valeurs de SF pour LoRa sont comprises entre

7 et 12. Lora utilise donc au total $2^{SF} - 1$ symboles. L'onde transmise pour une durée T_s pour un certain $s(nT_s)$ vaut :

$$c(nT_s + kT) = \frac{1}{\sqrt{2^{SF}}} e^{j2\pi((s(nT_s)+k) mod 2^{SF})kT\frac{\beta}{2^{SF}}} \tag{1.27}$$

pour k = 0...$2^{SF} - 1$ Le signal modulé est donc une onde composée de chirp. Chaque fragment de l'onde diffère de l'onde possèdant une fréquence initiale à 0, par un décalage de fréquence $s(nT_s)$. Cette caractéristiques est la raison de l'appellation FCSM.

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 permet de déterminer le taux de variation de fréquence pour un signal. Modifier le spreading factor ajuste diffé
rentes propriétés de la communications (source : The Thing Network[1]). 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 large, il y a moins de risque de subir des interférences.
— Plus petit débit de données. Le spreading factor contrôle 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 grande consommation. Les données transmises à un taux plus faible, la durée de transmission est donc plus longue ce qui prolonge le cout pour envoyer l'information.

Diminuer le spreading factor engendre l'effet inverse [1].

1.3.1.4 Structure d'un paquet LoRa

La figure 1.8 montre la structure d'un paquet Lora. Un paquet LoRa contient 3 parties différentes [26] :

FIGURE 1.8 – Structure d'un paquet Lora[12]

— 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 12,25 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 par défaut de 16 bits.

1.3.2 LoRaWAN

LoRaWAN est un protocol de type *Low Power Wide Area Network (LPWAN)* conçu 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 Aspects généraux de la technologie

Le coeur de LoRaWAN réside dans 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. Elle est également efficace dans différents environements. Le signal est capable de pénétrer diverses terrains et structures. Cela rend la technologie efficace aussi bien milieu rurale qu'urbain. Le déploiment d'une infrastructure LoRa ne nécessite pas de license, et son réseau peut être public ou privé.

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 3.3

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 sont donc autorisés à communiquer. Ensuite, une fois la communication commencée, les données sont chiffrées avant d'être transmises dans le réseau. Le framework sécuritaire de LoRa ne se limite pas à l'autentification et au chiffrement. LoRaWAN gère également les mise à jour en continu par les airs, ainsi qu'une supervision continue sur d'éventuelles intrusions. Les caractéristiques générales de LoRaWAN sont disponibles sur le site The Thing Network (https://www.thethingsnetwork.org/docs/lorawan/).

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 (liste des applications : https://

www.semtech.com/lora/lora-applications) :

— la surveillance environementale en général [19]. 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 (https://www.sensoterra.com/technology/global-lorawan-networks/), utilise notamment LoraWan pour surveiller la qualité des sols.

— Les *smart cities* [6]. LoRaWAN est actif sur différents aspects comme la gestion intelligente de l'éclairage, la gestion des déchets, la surveillance, etc.

— L'embarqué industriel [5]. La maintenance et la surveillance de matériel et de l'équipement peut être gérée par LoRaWAN. TataSteel (https://consulting.tatasteel.com/ourexpertise/plant-infrastructure-and-logistics/), une compagnie indienne, utilise LoRaWAN pour ces équipements industriels.

— La prévention de catastrophe naturelle. Que ce soit en prévision[29] ou après[4] 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 différents endroits dans le monde plus difficile. Cela peut aussi entrainer des problèmes de compatibilité entre régions, notament pour des chaines logistiques ou d'approvisionement qui en traversent 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 pour un facteur d'étalement SF = 7.

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 collisions entre messages si plusieurs appareils communiquent simultanénent. Ce système est basé sur une combinaison entre *Listen before talk (LBT)* et des delais aléatoires[14]. Il est néamoins possible que dans un environement très dense des collisions puissent encore se produire. La communication asynchrone et le système d'évitement de collision entrainent une augmentation du temps entre les envois et la réception de messages.

1.3.2.2 Topologie de LoRaWAN

FIGURE 1.9 – Topologie de l'infrastructure LoRaWAN (source : The Thing Newtork)

La figure 1.9 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 appareils 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. 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 supérieures de réception 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 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 ser-

veur réseau. Ils reçoivent les transmissions depuis les end devices dans leur zone de couverture et forwardent les messages vers le serveur réseau. Les gateways peuvent écouter plusieurs fréquences simultanénent (*multichanneling*) là où les end devices n'écoutent qu'une seule fréquence. Les gateways 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 chiffrement 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'utilisateurs finaux. Les utilisateurs intéragissent avec le serveur d'application pour n'importe quelle action à 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 axes 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 visibles par personne au sein du réseau hormi l'émetteur et le récepteur.

La sécurité repose sur le chiffrement 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[27].

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 airs, 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 authentifie 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 différentes, la *AppSKEY* pour le chiffrement des payloads d'application, et la *nwkSKEY* pour les fonctionalités du réseau (le chiffrement à la couche MAC, les vérifications 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. Le procédé des session est disponibles sur le site The Thing Network : https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/

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. La requête contient également le *Message Integrity Code*(MIC), un code calculé à partir de tous les champs de la join request.
— Le serveur réseau accepte (ou décline) la requête et vérifie les identifiants 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 serveur utilise AES en mode ECB pour chiffrer le message de join accept. Si le serveur a refusé le join request, il n'y a pas de réponse envoyée au device.
— 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. Il possède les informations additionelles suivantes : la DevAddr (son adresse assigné par le serveur réseau), la NwkSKey (utilisée pour calculer le MIC pour garantir l'intégrité des messages) et l'AppSKey (pour chiffrer et déchiffrer les payloads d'application dans les message pour garantir la confidentialité).

La seconde méthode est hardcodée et permet à un end device de rejoindre directement le réseau sans passer par l'identification. 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 déployé au préalable dans la zone de couverture du réseau LoRaWAN.
— Sans devoir initialiser 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

Expérimentations

2.1 Matériel

2.1.1 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.

2.1.1.1 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 2.1 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 2.1 – RTL SDR Dvb-T dongle

2.1.1.2 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 2.2 montre la SDR en question.

FIGURE 2.2 – RTL SDR R820T2

2.1.1.3 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 2.3 montre la SDR avec l'antenne utilisée posée sur un trepied.

(a) SDR HackRF One (b) Antenne

FIGURE 2.3 – SDR HackRf One avec son antenne

2.1.2 Module d'émission Lora

2.1.2.1 Module RN2483

FIGURE 2.4 – 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 2.4 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 2.3.

2.1.2.2 Module Pycom lopy

FIGURE 2.5 – 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 2.5 montre le module connecté à une antenne.

2.1.2.3 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 2.6 – 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.

2.1.3 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.

2.1.3.1 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 2.7 montre comment configurer simplement le logiciel. La figure 2.7a 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 2.7b 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 2.7 – Configuration de GQRX

La figure 2.8 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 2.8 – 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 2.8, 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.

2.1.3.2 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 2.9 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 2.3.2.

FIGURE 2.9 – Enregistrement d'un signal avec URH

2.1.3.3 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.

2.1.3.4 IDE Arduino

L'utilisation de l'IDE Arduino 2.2.1 est nécessaire pour la configuration du module Arduino décrit dans la section 2.1.2.3. L'IDE permet d'uploader le code dans le module. Le code est disponible dans l'annexe .1.

2.2 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.

2.3 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 .2

2.3.1 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 2.10 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 2.10 – 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 2.11 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 2.11 – capture module Arduino sur GQRX avec RTLSDR R820T2 Sample rate = 1.8MHz

2.3.2 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 2.12 et 2.13 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 2.3.

FIGURE 2.12 – Capture du signal

FIGURE 2.13 – Sauvegarde du signal

A partir de la figure 2.13, 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 2.14 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 2.15).

FIGURE 2.14 – Signal LoRa sous forme spectrogramme et analogique

FIGURE 2.15 – 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 :

$$F_{recepteur} = F_{emetteur} - \frac{BW}{2} soit : \tag{2.1}$$

$$868Mhz - \frac{125KHz}{2} = 867,9375MHz \tag{2.2}$$

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 2.16 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 2.16 – 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.

2.3.3 Analyse avec python

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 2.17 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 2.18 fait un zoom sur les 4000 premiers échantillons. Les deux ondes apparaissent grâce au zoom.

FIGURE 2.17 – Plot des échantillons I/Q avec matplotlib

FIGURE 2.18 – zoom sur les 4000 premiers échantillons I/Q

2.3.4 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 (voir annexe .3. 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 2.19 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 :

$$TimeOnAir(TOA) = \frac{N_{samples}}{samplerate} = 0.0245 \tag{2.3}$$

FIGURE 2.19 – 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 2.20), 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)*. La valeur du RMS pour un set x de n données $x_1, x_2, ..., x_n$ vaut la racine carrée de la moyenne arithmétique des carrés des valeurs soit :

$$x_{RMS} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + ... + x_n^2} \tag{2.4}$$

L'implémentation de l'automatisation et de la normalisation est disponible en annexe .4

FIGURE 2.20 – Configuration en intérieur

Chapitre 3

Identification d'appareils LoRa par la méthode des constellations traces figures

Ce chapitre a pour but d'approfondir la méthode des *differential constellations traces figures (DCTF)* developpé dans l'article [32] par Yu Jiang, Linning Peng, Aiqun Hu, Sheng Wang, Yi Huang et Lu Zhang. Dans un premier temps la méthode de l'article est présentée de manière théorique avant d'être appliquée sur les appareils afin de réaliser l'objectif du mémoire. Cependant, plusieurs modifications ont été ajoutées afin de pousser plus loin les possibilitées de la méthode.

3.1 Fadio Frequency Fingerprinting avec DCTF

L'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[24].

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

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 :

$$x(t).x(t+n)e-j2\pi on \tag{3.1}$$

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 :

$$Rs = BW/2sfN = fs/Rs \tag{3.2}$$

carrier frequency offset :

$$CFO \tag{3.3}$$

3.2 Méthode DCTF en pratique

L'analyse suivante est basée sur l'article publié [32]. 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 3.1 montre la représentation dans le temps du signal test et sa représentation sous forme de constellation est donnée par la figure 3.2.

Selon la section **??**, le diagramme de constellation seul n'est pas suffisant pour pouvoir identifier des composantes uniques au signal. Pour pouvoir observer l'émergence d'une signature, il faut appliquer la méthode différentielle décrite dans 3.1 ainsi que dans l'article [32]. La figure 3.3 montre le diagramme differentiel de constellation *DCTF* du signal. L'équation 3.1 possède deux paramètres à calculer. Le *différential interval (N)* se calcule via 3.2 et vaut 4096. Le *Carrier Frequency Offset (CFO)* peut être estimé via 3.3. 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

FIGURE 3.1 – LoRa signal

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 3.4 montre le même diagramme qu'à la figure 3.3 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 [32] 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 3.4 par les points qui ne se situent pas sur la constellation mais entre la constellation et le centre du 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 3.5 permet de distinguer clairement la région dense dans le

FIGURE 3.2 – Diagramme de constellation

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 [32]) de la zone la plus dense sont également considérés. La figure 3.6 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 3.3 – DCTF du signal test

3.3 Autres 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[11]. 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 pu-

FIGURE 3.4 – DCTF du signal test

blié par H. Kawai, S. Ata et N. Nakamura [15] 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[3]. 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.

FIGURE 3.5 – DCTF du signal test

FIGURE 3.6 – Points de haute densité dans la DCTF du signal test

Conclusion

Mettez votre conclusion ici. Dressez le bilan de votre travail effectué, en prenant du recul. Discuter de si vous avez bien réussi les objectifs du travail ou non. Présentez les perspectives futurs.

Bibliographie

[1] Lorawan spreading factors. `https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/`. Accessed on 2024-02-08.

[2] *An Introduction To Analog And Digital Communications*. Wiley India Pvt. Limited, 2009.

[3] M. Anjum, M. A. Khan, S. A. Hassan, A. Mahmood, and M. Gidlund. Analysis of rssi fingerprinting in lora networks. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1178–1183. IEEE, 2019.

[4] M. I. Arsyad, A. A. Pekerti, and B. A. Nugraha. Time-slotted lorawan implementation of gps tracker system for post-disaster utilization. pages 172–178, 2020.

[5] M. Ballerini, T. Polonelli, D. Brunelli, M. Magno, and L. Benini. Nb-iot versus lorawan : An experimental evaluation for industrial applications. *IEEE Transactions on Industrial Informatics*, 16(12) :7802–7811, 2020.

[6] P. J. Basford, F. M. Bulot, M. Apetroaie-Cristea, S. J. Cox, and S. J. Ossont. Lorawan for smart city iot deployments : A long term evaluation. *Sensors*, 20(3) :648, 2020.

[7] A. Bekele. Cooley-tukey fft algorithms. *Advanced algorithms*, 2016.

[8] R. Canetti, D. Shahaf, and M. Vald. Universally composable authentication and key-exchange with global pki. In *Public-Key Cryptography–PKC 2016 : 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II 19*, pages 265–296. Springer, 2016.

[9] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, S. X. Ng, and L. Hanzo. The evolution of quantum key distribution networks : On the road to the qinternet. *IEEE Communications Surveys & Tutorials*, 24(2) :839–894, 2022.

[10] D. Dasgupta, J. M. Shrein, and K. D. Gupta. A survey of blockchain from security perspective. *Journal of Banking and Financial Technology*, 3 :1–17, 2019.

[11] J. Deogirikar and A. Vidhate. Security attacks in iot : A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 32–37, 2017.

[12] E. Gambi, L. Montanini, D. Pigini, G. Ciattaglia, and S. Spinsante. A home automation architecture based on lora technology and message queue telemetry transfer protocol. *International Journal of Distributed Sensor Networks*, 14 :155014771880683, 10 2018.

[13] P. Heckbert. Fourier transforms and the fast fourier transform (fft) algorithm. *Computer Graphics*, 2(1995) :15–463, 1995.

[14] Q. L. Hoang, H. P. Tran, W.-S. Jung, S. H. Hoang, and H. Oh. A slotted transmission with collision avoidance for lora networks. *Procedia Computer Science*, 177 :94–101, 2020.

[15] H. Kawai, S. Ata, N. Nakamura, and I. Oka. Identification of communication devices from analysis of traffic patterns. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2017.

[16] S. Li, M. Iqbal, and N. Saxena. Future industry internet of things with zero-trust security. *Information Systems Frontiers*, pages 1–14, 2022.

[17] A. Marquet, N. Montavont, and G. Z. Papadopoulos. Towards an sdr implementation of lora : Reverse-engineering, demodulation strategies and assessment over rayleigh channel. *Computer Communications*, 153 :595–605, 2020.

[18] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *ICT express*, 5(1) :1–7, 2019.

[19] A. N. Rosli, R. Mohamad, Y. W. Mohamad Yusof, S. Shahbudin, and F. Y. Abdul Rahman. Implementation of mqtt and lorawan system for real-time environmental monitoring application. pages 287–291, 2020.

[20] M. Samaniego and R. Deters. Zero-trust hierarchical management in iot. pages 88–95, 2018.

[21] K. Sha, T. A. Yang, W. Wei, and S. Davari. A survey of edge computing-based designs for iot security. *Digital Communications and Networks*, 6(2) :195–202, 2020.

[22] H. Sinanović and S. Mrdovic. Analysis of mirai malicious software. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5. IEEE, 2017.

[23] S. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997.

[24] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar. A review of radio frequency fingerprinting techniques. *IEEE Journal of Radio Frequency Identification*, 4(3) :222–233, 2020.

[25] F. Song. A note on quantum security for post-quantum cryptography. In *International Workshop on Post-Quantum Cryptography*, pages 246–265. Springer, 2014.

[26] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg. An open-source lora physical layer prototype on gnu radio, 02 2020.

[27] P. Thaenkaew, B. Quoitin, and A. Meddahi. Leveraging larger aes keys in lorawan : A practical evaluation of energy and time costs. *Sensors*, 23(22) :9172, 2023.

[28] F. Thiesse and F. Michahelles. An overview of epc technology. *Sensor review*, 26(2) :101–105, 2006.

[29] K.-H. Tseng, M.-Y. Chung, L.-H. Chen, and Y.-W. Huang. Implementation of composite lpwan on the slope disaster prevention monitoring system. *IEEE Sensors Journal*, 22(3) :2658–2671, 2022.

[30] L. Vangelista. Frequency shift chirp modulation : The lora modulation. *IEEE Signal Processing Letters*, 24(12) :1818–1821, 2017.

[31] R. Weinstein. Rfid : a technical overview and its application to the enterprise. *IT Professional*, 7(3) :27–33, 2005.

[32] X. Wu, Y. Jiang, and A. Hu. Lora devices identification based on differential constellation trace figure. In *Artificial Intelligence and Security : 6th International Conference, ICAIS 2020, Hohhot, China, July 17–20, 2020, Proceedings, Part I 6*, pages 658–669. Springer, 2020.

.1 Annexe A :Code module Arduino

```

#define LED_BUILTIN 3

#define SX1276_NSS 10
#define SX1276_RESET 9
#define SX1276_DIO0 2

// the setup function runs once when you press reset or
    power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(57600);
  while (!Serial);

  LoRa.setPins(10, 9, 2);

  if (!LoRa.begin(868.1E6)) {
    Serial.println("Lora init failed. Check your connections
        .");
    while (true);
  }

  Serial.println("LoRa setup");
  //LoRa.setSyncWord(0x12); // private
  LoRa.setSyncWord(0x34); // LoRaWAN
  Serial.println(" Sync Word = 0x34 (LoRaWAN)");
  LoRa.setSpreadingFactor(12);
  Serial.println(" SF = SF12");
  LoRa.setSignalBandwidth(7.8E3);
  Serial.println(" BW = 7.8kHz");
  LoRa.dumpRegisters(Serial);

}

int counter = 0;

// the loop function runs over and over again forever
void loop() {

```

```
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
  Serial.println("Sending frame...");

  unsigned long t0 = micros();
  LoRa.beginPacket(0); // 0 = explicit header, 1 = implicit header
  LoRa.print("B");
  //LoRa.print(counter);
  LoRa.endPacket(0); // 0 = wait for end of packet transmission

  // Time on Air as calculated per https://loratools.nl/#/airtime
  // should be around 13 seconds, but measurements give around 10 seconds (why ?)
  Serial.print("ToA ~ ");
  Serial.println(micros()-t0);

  //LoRa.idle(); // Warning, further transmission do not work; need to reset module ?
  //LoRa.setSyncWord(0xFF); /* LoRaWAN */


  counter++;
}
```

Listing 1 – Module Arduino

.2 Annexe B :Code module RN2483

```python
import serial
import time
from utils import read_file, write_file

SERIAL_PORT = '/dev/ttyUSB0'
BAUD_RATE = 57600
SPREADING_FACTOR = 8

config_commands = [
    "sys get ver\r\n",
    "radio set mod lora\r\n",
    "radio set freq 868000000\r\n",
    "radio set pwr 14\r\n",
    f"radio set sf sf{SPREADING_FACTOR}\r\n",
    "radio set cr 4/8\r\n",
    "radio set bw 125\r\n"
]

MESSAGE_COMMAND = "radio tx 1509ACf\r\n"


def config(ser: serial.Serial):
    # Send commands and read responses
    for command in config_commands:
        write_command(ser, command)


def send_signal(ser: serial.Serial):
    write_command(ser, MESSAGE_COMMAND)


def write_command(ser: serial.Serial, command: str):
    ser.write(command.encode())
    response = ser.read(100)
    print("Command:", command.strip())
    print("Response:", response.decode().strip())
    print("------------")
```

Listing 2 – Configuration module RN2483

.3 Annexe C :Implémentation de la configuration RTL SDR via PyRTLSDR

```
import math
import time
import numpy as np
from rtlsdr import RtlSdr
import hackrf
from utils import cut_preamble, save_signal_old, read_file,
    write_file
from find_centers import SAMPLES_FOLDER


SAMPLE_RATE = 2_000_000


def capture_signal_rtlsdr():
   # Configure RTL-SDR parameters
   sdr = RtlSdr()
   sdr.sample_rate = SAMPLE_RATE
   #sdr.bandwitdh = 250000
   sdr.center_freq = 867937500
   sdr.gain = 5

   # Start signal capture
   capture_duration = 2.5 # in seconds

   print(f"Capturing signal for {capture_duration} seconds
      ...")
   nb_samples = math.ceil(capture_duration * sdr.
      sample_rate /16384)*16384
   #samples = sdr.read_samples(5046272)
   samples = sdr.read_samples(nb_samples)

   sdr.close()
   samples = np.array(samples, dtype=np.complex64)
   return samples
```

Listing 3 – Configuration RTL SDR

.4 Annexe D :Implémentation de l'automatisation des captures de signaux

```
if name == 'main':
   SER = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
   for i in range(5):
      time.sleep(1)
      config(SER)
      write_file('tmp.txt','0')
      time.sleep(1.5)
      send_signal(SER)
      while read_file('tmp.txt')[-1] != '1':
         print('emmeteur is waiting')
         time.sleep(1)
   #config(SER)
   #send_signal(SER)
   SER.close()
```

Listing 4 – Emetteur

```

if name == "main":
   PREAMBLEDURATION = 0.0245
   for i in range(2):
      while read_file('tmp.txt')[-1] != '0':
         print('recepteur is waiting')
         time.sleep(0.2)

      SIGNAL = capture_signal_rtlsdr()
      #SIGNAL = capture_signal_hackrf()
      PREAMBLE = cut_preamble(SIGNAL, 0.03, int(
         PREAMBLE_DURATION*SAMPLE_RATE))

      if len(PREAMBLE) > 0:
         save_signal_old(PREAMBLE,f'{SAMPLES_FOLDER}sample{
            i+1}', np.complex64)
         print(f'signal {i+1} saved')
      else:
         print(f"no signal found for {i+1}")

      write_file('tmp.txt', '1')
```

Listing 5 – Récepteur

```python

import numpy as np


def read_file(filename: str):
    with open(filename, 'r',encoding='utf8') as reader:
        return reader.readlines()

def write_file(filename: str, content: str):
    with open(filename, 'a',encoding='utf8') as writer:
        writer.write('\n' +content)

        def save_signal(signal: np.ndarray, filepath: str, dtype) -> None:
    signal_to_save = np.array(signal, dtype=dtype)
    print(f"Saving captured signal to {filepath}.npz")
    np.savez_compressed(filepath, signal_to_save)


def load_signal(filepath: str, dtype) -> np.ndarray:
    if filepath.endswith('.npz'):
        return np.load(filepath)['arr_0']
    return np.fromfile(filepath, dtype=dtype)

def cut_preamble(signal, treshold, preamble_size):
    start_index = 0
    consecutive_start = 0
    while start_index < len(signal) and consecutive_start <3:
        if abs(np.real(signal[start_index])) > treshold or abs(np.imag(signal[start_index])) > treshold:
            consecutive_start += 1
        else :
            consecutive_start = 0
        start_index += 1

    if start_index + preamble_size > len(signal) -1:
        return []

    return signal[start_index:start_index+preamble_size]

def rms_normalize(samples: np.ndarray) -> np.ndarray:
```

```
    rms_values = np.sqrt(np.mean(np.abs(samples)**2, axis=0)
        ) # Compute RMS values
    normalized_samples = samples / rms_values # Normalize
        samples
    return normalized_samples
```

Listing 6 – Preprocessing