Università degli Studi di Padova

PREAMBLE DETECTION AND SYNCHRONIZATION IN LORAWAN

Master Thesis

ICT for Internet and Multimedia
School of Engineering
University of Padova
John Sengendo (Matr. No. 2006024)

June 30, 2023

Supervisor: Prof. Dr. Lorenzo Vangelista

Abstract

In this thesis, I present an approach on detecting the preamble in LoRaWAN (Long Range Wide Area Network). The preamble is a pattern of bits used to synchronize the receiver with the transmitter. Its typically 8 symbols long and serves as an identifier for the beginning of a LoRaWAN packet transmission. When a LoRaWAN device transmits data, it begins by transmitting the preamble as the first part of the packet. The receiver continuously listens for incoming signals and checks if the received signal matches the expected preamble pattern. Once the receiver successfully detects the preamble, it can synchronize with the transmitter and prepare to receive the payload of the LoRaWAN message.
If the preamble detection fails, the receiver assumes that there is no LoRaWAN transmission and continues listening for incoming signals.
In the thesis, to successfully achieve my goal of preamble detection, I first do a literature review on LoRa, LoRaWAN, modulation techniques and the various detection techniques already being used then I perform simulations in MatLab.

Contents

List of Figures

Chapter 1

Introduction

1.1 Internet of Things.

Internet of things (IoT) is an arrangement of physical objects embedded with sensors, processing ability software, and other technologies that interlink and exchange data with other multiple devices and systems over the Internet or other communications networks. The core concept behind Internet of Things is to create a vast network of devices that can communicate and share data with each other, as well as with cloud-based platforms and applications. This interconnection allows for real-time monitoring, control, and automation thus enabling new levels of efficiency, convenience, and insights across various domains.
In the consumer market, Internet of Things technology is applied in products pertaining to smart homes mainly used for home control, security systems and cameras. These devices can be part of one or multiple common ecosystems, and these can be controlled via devices associated with that domain such as smartphones. IoT based smart home technology further presents the following benefits.

1. It works together with current household appliances. This doesn't mean discarding current technologies, but to work along with them in providing a better life. Any new appliance complying with the defined architecture and protocols is capable of being added into the system. Current appliances can as well be added into the system efficiently.

2. It as well provides convenience in use with the control center continuously communicating with users.

3. The agents can work in conjunction with each other, under constraints of ontology, using service oriented language, which is defined to be a promising programming language for the next few years [17].

The expected benefits provided by IoT are widely spread and the opportunities that come with IoT are enormous and have proved to increase profits of organizations, especially those focused on digital transformation [18]. Therefore, it is evidently clear that the IoT market is growing bigger than before. Consequently, huge companies are already starting to invest in IoT in different sectors. The IoT

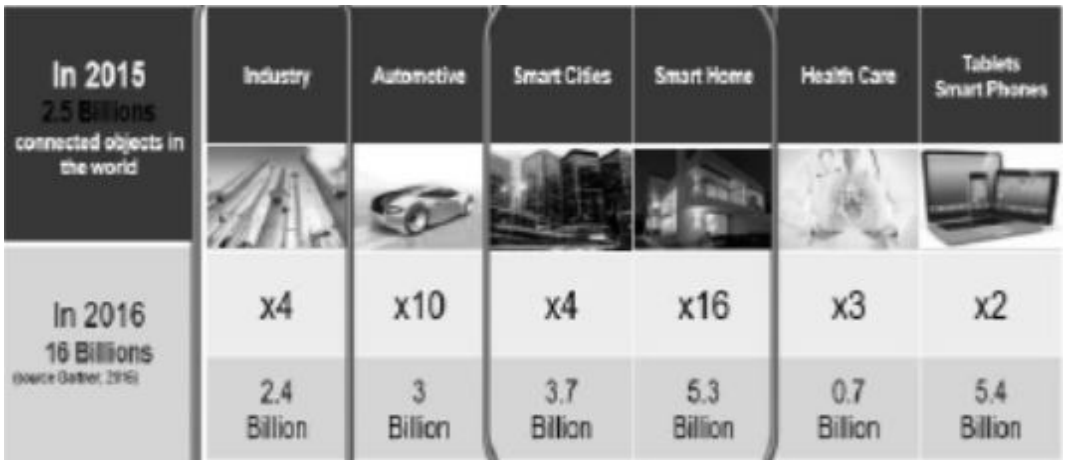

Figure 1.1: IoT device growth[1].

conjecture shows that the IoT market size is enormous and steadily growing [18].

Internet of Things has also transformed the health care sector by aiding different medical applications. A wide range of technologies are in current utilization such as, Body Sensor Networks(BSN)[19] which are a network of sensors and biosensors that can be carried along or can be worn anywhere on the body [19]. For example, wearable sensors like smartwatches and smart bands, which are readily available, and can measure blood pressure levels and heart rate. The applications are further growing exponentially and now being applied in smart cars [20] especially in preventing accidents by confirming the active state of the drivers, recognizing the drunken state of the driver, checking the air quality prevailing inside the car and keeping track of the location details of the car so that relatives are notified in case of an accident [21].

Other applications include: Monitoring of road conditions by sensor nodes, air quality measurement by various gas sensors, monitoring of environmental information in the sea such as sailing of sea boats, and geolocation tracking for effective livestock management. IoT has as well been employed effectively in both indoors and outdoors in many critical environments for health monitoring, environmental monitoring, and tracking.
In the times when the world was battling with the COVID-19 pandemic, IoT devices played a significant role in enhancing public safety solutions by assisting with tracing patient contacts and addressing the technology needs of medical professionals[4].
As shown in Figure 1.1, the Internet of Things world is predicted to comprise of billions of connections, many devices will playing a fine role. Devices are not only smartphones, smartwatches or connected cars; they are also sensors in every part of the environment. The ability to cost-effectively connect devices within the unlicensed frequency spectrum is a critical challenge. Congestion and interference have become more likely. This can be connected to trying to have a conversation in a room that is slowly getting more and more crowded with louder and louder voices.

LoRaWAN a promising wireless standard because of its low power, long range, and deep penetration capabilities is used in most IoT applications to continuously monitor the environment and surrounding conditions especially in disaster-prone regions [22].
In recent years, different wireless communication technologies such as Zigbee, Bluetooth have been frequently used. Despite them gaining a big market share, their short-range, security risks, and hard-to-configure network made them difficult to be applied in commercial buildings and other fast growing Internet of Things applications. Also majority of the applications in the present day world are embedded systems[1]. But in all these, power requirement and battery life is a major issue .These systems have very high power constrains as most of these system are battery operated and not externally powered [1]. All these challenges have lead to an accelerated development of LoRaWAN.

In Internet of Things devices, the basic service layer is the most important layer as its used in connecting to the internet. The basic services provided through access to the internet are divided into three types [5]:

(1) Device identification. This is the most simple service that devices can provide. To successfully identify them, it is recommended for devices to have unique identifiers. Currently, there is no unified standard for Internet of Things devices identifiers. The most common things identifiers contain are bar codes, bi-dimensional codes and RFIDs which are not necessarily unique identifiers [23].

(2) Data collection. When it comes to data collection, some of the fields of application are industrial control, environmental monitoring and control, automated meter reading. As a result of data security, most of this data is not pushed to the internet. The data obtained through such services is extremely relevant to the applications, and a lot of them involve industrial standards. For instance, the BoYang water resources monitoring IoT system which supports the monitoring of flow, pressure, temperature, humidity, and other nine categories of data according to the recommended standards.

(3) Things behavior control: Things behavior control is as well one of the initial ways to be put in application. This type of control is an active behavior, mainly for control in industries, water / fluids flow control and other fields. There is a new fast growing area which is intelligent homes, a good way to control home appliances and lighting bulbs with the basic services based on Internet of Things.

1.2 LoRa

LoRa (Long Range) is currently one of the fastest growing technology. Long battery life, long distance communication and low cost of application are the main focus of the Internet of Things engineers, LoRa is perfectly solves most of these requirements. It outstands as a prominent wireless communication technology used for IoT, which has several advantages when applied in conjecture with UAV-based networks, such as; improved packet reception and low complexity [24]. Internet of Things systems where LoRa mostly applies include four explicit sections which are; sensors/devices, connectivity, processing of information and a user interface [4].

LoRa follows an approach based on the following two distinct layers: i) a physical layer based on radio modulation approach; and ii) a MAC layer protocol (LoRaWAN an open standard).

Further key important aspects about LoRa are explained below:

LoRa makes use of a wider bandwidth usually of 125 kHz which is extendable to 250kHz or 500kHz depending on the application for broadcasting signals [25]. Application of a wider bandwidth enables LoRa to be resistant to channel noise, long term relative frequency, doppler effects and fading. However, spreading a narrow band signal over wider band width makes less efficient utilization of the spectrum.
LoRa is based on spread spectrum modulation techniques and operates in the unlicensed Industrial, Scientific, and Medical (ISM) bands. It utilises chirp spread spectrum modulation to achieve long-range communication with robustness against noise and interference. LoRa technology offers exceptional range, giving capability to communication over several kilometers in rural areas and up to a few hundred meters in urban environments. The range can be further extended by using repeaters , which act as intermediaries between end devices and the network server.
Low Power Consumption: One of the critical advantages of LoRa is its low power consumption. LoRa devices are designed to operate on batteries for an extended period, ranging from months to years, depending on usage patterns and the power requirements of the specific application.
Solubility: LoRa networks can support a large number of devices, making it highly scalable. It utilizes a star-of-stars network architecture, where multiple end devices communicate with a central gateway or base station. Gateways act as concentrators and forward the data to the network server for further processing.
Data Rate: LoRa is optimized for low data rates, typically ranging from a few hundred bits per second (bps) to a few kilobits per second (kbps). While it may not be suitable for applications requiring high-speed data transfer, it is well-suited for transmitting small amounts of data at regular intervals.
Applications: LoRa technology finds applications in numerous fields, including smart cities, agriculture, asset tracking, environmental monitoring especially in disaster prone regions, industrial automation, and smart home systems. It is particularly useful for applications that require long-range communication, extended battery life, and the ability to operate in remote or challenging environments.

Generally, LoRa technology offers a cost-effective and power-efficient solution for long-range, low-power Internet of Things applications. Its capabilities make it suitable for a wide range of use cases where long-range communication, extended battery life, and scalability are important considerations.

1.3 LoRaWAN

LoRaWAN is a protocol that operates on top of the LoRa physical layer. It provides a standardized communication protocol for LoRa devices, enabling

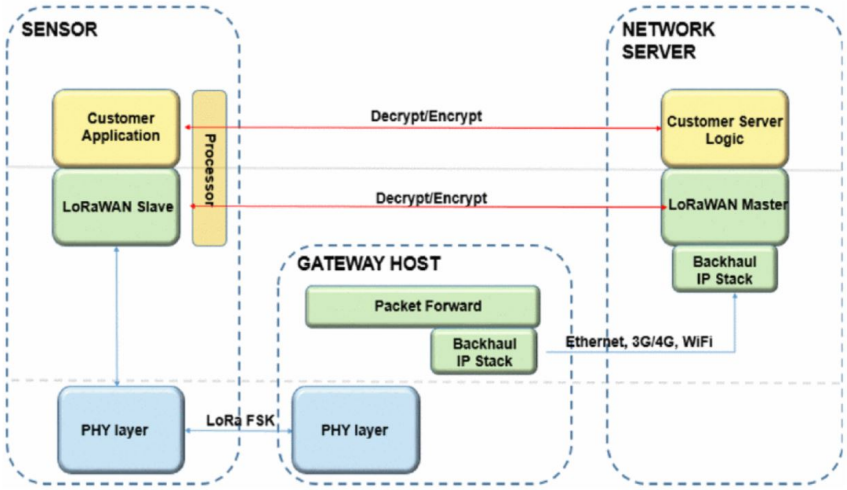

Figure 1.2: LoRaWAN network layers[2].

interoperability and seamless integration into existing network infrastructure. LoRaWAN defines the network architecture, security mechanisms, and communication protocols for LoRa-based networks. IEEE 802.15.4 forms the security protocol of LoRaWAN where every LoRaWAN node has its own App Key[26]. LoRaWAN network layers are demonstrated in figure 1.1. Thus, LoRaWAN is approached as a communication protocol and network architecture, while LoRa physical layer gives the capability for the long range link.
With the rise of LoRaWAN, further enhancements for users of Internet of Things technology can be obtained. By making use of many gateways in the network, it can deal with numerous hubs at various areas inside the region, which differs from the Wi-Fi-based framework which requires a lot more access points to cover large area. In the case of wider range coverage of IoT systems, LoRaWAN is quite more useful as compared to Wi-Fi and Bluetooth used in short range. However, there have been some shortcomings on the transmission speeds and constraint on the payload size.

LoRaWAN Classes:
LoRaWAN offers three main classes of devices through which communication takes place, these are. Class A, Class B and Class C .

1. Class A devices are characterised of requiring more latency time and are more energy efficient which makes them applicable in many low powered sensor networks. As shown in figure 1.3, they only have two very short receive windows after they have transmitted a packet[3]. After the receive windows, class A devices go to sleep in order to conserve energy.

2. Class B devices have time synchronization with a beacon. They are also associated with low power devices. As illustrated in figure 1.4, class B devices open extra receive windows at scheduled intervals[3]. The receive windows are synchronized by beacons delivered from the gateway.

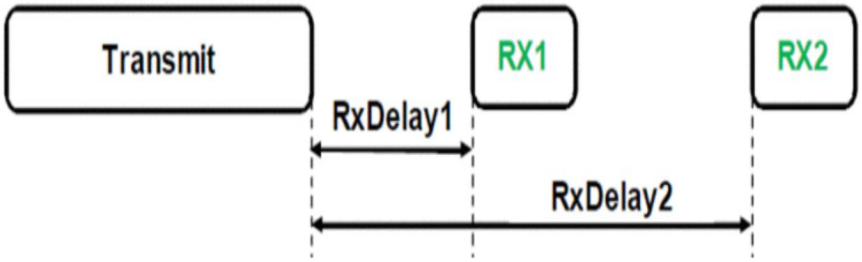

Figure 1.3: Class A[3].

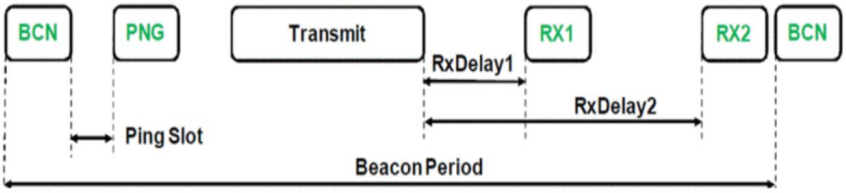

Figure 1.4: Class B[3].

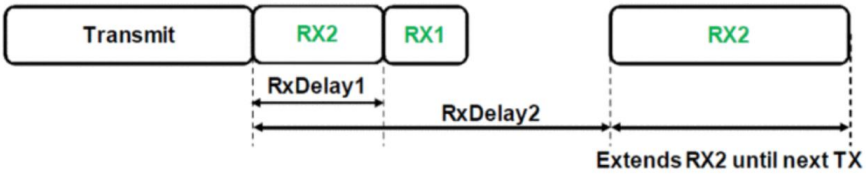

Figure 1.5: Class C[3].

3. Class C have low latency time but bare maximum reception slots[3]. These are bidirectional but need an external power supply. C devices, as they are normally not battery-powered can afford to continuously have their radio in receive mode (as long as they are not transmitting themselves), giving possibility for instantaneous transmission of data towards a device without having to wait for a receive window to open.

1.4 Problem statement

With the development of LoRaWAN technology, many challenges in Internet of Things technology where resolved and a lot of applications adopted the technology. However, with this exponential growth in its utilization, some shortcomings arose. For example, there is a lot of interference among users and as a result, this interference causes a lot of delays in communication.
When a LoRaWAN device transmit data, it begins by transmitting the preamble which is a pattern of bits used to synchronize the receiver with the transmitter.

My thesis is centred on detecting the preamble in LoRaWAN communication. Once the receiver successfully detects the preamble, further capabilities such as synchronization with the transmitter can be attained and thus proper reception of the payload message.
Generally, preamble detection in LoRaWAN communication plays a crucial role especially in achieving reliable communication and synchronization between devices and gateways in a low-power, long-range wireless networks.

	Europe	North America
Frequency Band	867-869 MHz	902-928MHz
Channels	10	64+8+8
Channel BW Up	125/250KHz	125/500KHz
Channel BW Dn	125KHz	500KHz
TX Power Up	+14dBm	+20dBm typ (+30dBm allowed)
TX Power Dn	+14dBm	+27dBm
SF Up	7 12	7 10
Data rate	250bps-500kbps	980bps-21.9kbps
Link Budget Up	155dB	154dB
Link Budget Dn	155dB	157dB

Figure 1.6: Lora/LoraWAN Operating Specifications for the different region[4].

Chapter 2

State of Research and Theoretical literature review

As highlighted in chapter 1, LoRaWAN technology demonstrate a great potential for Internet of Things as it enables communication over several kilometers with extended battery life for Internet of Things devices and sensors [7]. It as well forms the standardized MAC protocol as per the LoRa alliance. The first version of LoRaWAN was published by LoRa alliance in 2015 and the second version in 2017. LoRaWAN technology is further governed by an Internet of Things system architecture which is divided into layers [5], as demonstrated in Figure 2.1. The bottom is the IoT physical node layer subdivided into two layers, sensing device layer, and the intelligent device layer; the layer above is associated with abstraction of things in IoT, its divided into physical information layer and logical information layer; then the above is the service layer,sub-divided into the IoT basic service layer and service middle layer; the top most forms the application layer.
The devices in sensing and executing device layer are split into sensing devices and executing devices. Sensing devices can be sensors that directly connects to the intelligent devices and transform the external physical signals into signals that can trigger the intelligent devices.

Mainly, the architecture of the IoT network applied in LoRaWAN consists of end nodes that are responsible for the collection and processing of the sensors data, a gateway that allows communication of the nodes through the LoRaWAN technology and connects to the central point using the Internet, a central server that allows for the data management functions, processing, interpretation and storage of the data collected by the sensors and subsequently to be visualized towards the client. The topology used is of star type illustrated in figure 2.2.
The layer where the hardware for Internet of Things resides provides for the physical flow of device information to the upper layers, which contain the physical information layer. In communication with upper layers, It's much better to use data which is much easier to interpret and represent[5]. So information

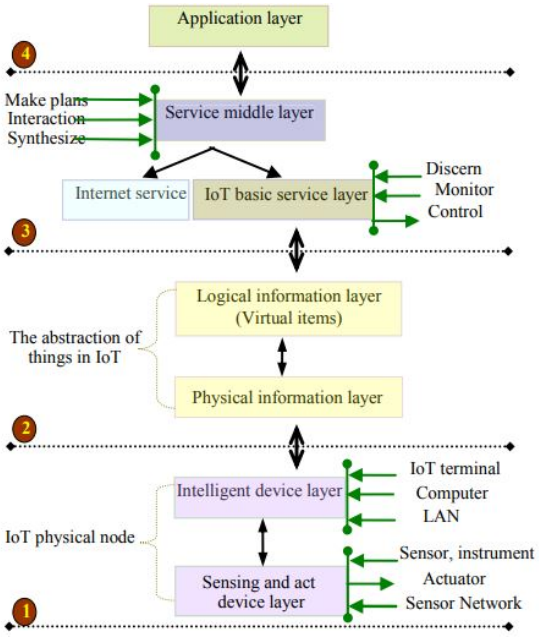

Figure 2.1: Layered architecture [5].

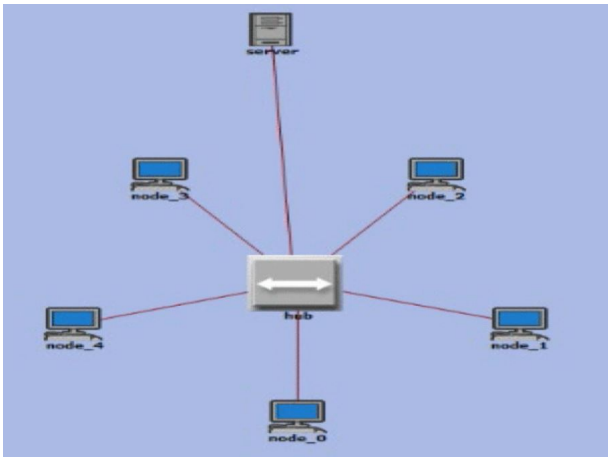

Figure 2.2: Star network topology [6].

represented by this way constitutes the logical information of things.
Below I further give more explanations on the layered architecture in figure 2.1.

1. **IoT physical nodes.** As illustrated in Figure 2.1, the IoT devices layer is the same as the IoT physical nodes, which forms the base of the bottom-up research idea for IoT systems. Whether it is a sensor technology in sensing device layer or embedded systems technology in the intelligent devices layer, they all fall in the same bracket of the research domain in IoT [5]. IoT terminal, computer and local area network. These three separate devices are classified into the IoT node intelligent device layer, which reason is that IoT nodes not only keep at hand things, but again must directly access the internet and deliver the basic services based on their application. This means that the service carriers that are able to provide services, compatible with the internet service-based architecture, make the internet transit to the IoT at the cost effective rate.
   This approach of design and hierarchical separation for IoT physical node enhances the connotation of the IoT nodes and makes them more flexible, and provides the physical nodes with a variety of granularity, different application modes and different access ways have the unified extension. Be it for moving or fixed objects, and whether for objectives with a broader or narrow range, they are all excellent adaptability.

2. **The abstraction of IoT things.** The abstraction process of Internet of Things devices is the way of setting up the information model for the devices.
   Standardizing on grounds of ensuring the versatility of the information model is one of the main goals of Internet of Things research all the time [5].

3. **Basic services based on things.** When a user application requests a service from a service provider, the service can be accomplished as a standalone integral task or as a service decomposition and completed all together [5].

2.1 LoRaWAN Network Architecture.

The LoRaWAN network topology is shown in figure 2.3. As seen in the figure, the long range star architecture is adopted to reduce the power consumption[7]. On the left, the end node devices can be a sensors, actuators, or both which are often battery operated. These end devices are wirelessly connected to the LoRaWAN network through gateways using wireless connections. They transmit data packets that are typically received by several gateways that will forward them to the network server via a network connection like 4G or Ethernet.
The end devices as well function as nodes that are connected to sensors and other interfaces which then transmit data to the gateways, which then needs to be sent to the Network Server. Each gateway is registered to a LoRaWAN network server. LoRaWAN gateways can be categorized into indoor and outdoor gateways illustrated in figure 2.4 and 2.5 respectively. Indoor gateways are very cost-effective and suitable for providing excellent coverage in places like deep-indoor locations, spaces covered by multiple walls, basements, underground tunnels and multi-floor buildings. These gateways have built-in internal

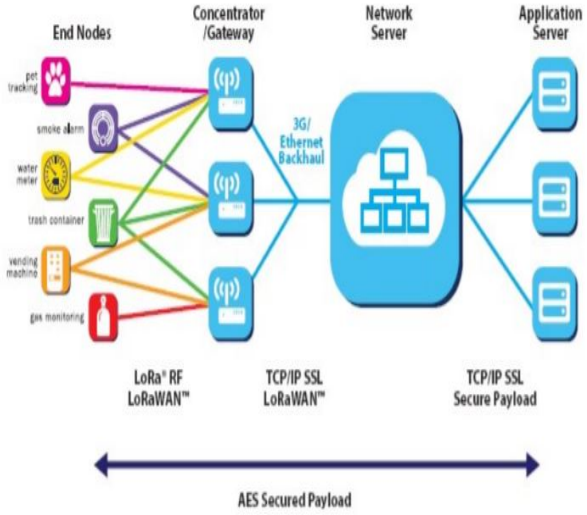

Figure 2.3: LoRaWAN Network topology[7].

antennas or external antennas. Depending on the indoor physical surroundings, some indoor gateways can receive messages from sensors located several kilometers away[8].
On the other-hand, outdoor gateways are designed to be tolerant to external weather conditions and are mostly mounted outside. The network server then manages multiple complex operations, such as filtering redundant packets caused by scenarios such as network congestion. When congestion occurs, it limits the available bandwidth for data transmission. In such cases, the sender may retransmit packets, assuming that some packets may have been lost due to congestion. However, if the original packets were not lost and the re-transmitted packets arrive, redundant packets are created on the network. Network errors also cause redundant packets because networks can generate errors during packet transmission. For instance, electromagnetic interference, noise, power failures or malfunctioning network equipment can corrupt packets. When errors occur, network protocols may trigger re-transmissions thus resulting in redundant packets. The network server as well manages security which involves operations of protecting data and applications that are connected to the network [27].
The gateway is designed to be able to receive packets at the same time from a large amount of end-nodes. This way the network capacity is increased by exploiting a technique of adaptive data rate where control at the end device is done for the following transmission parameters.

1. Spreading factor
2. Transmission power.

By the use of different spreading factors, multiple signals with different data rates can be simultaneously handled by the gateway on the same channel.

Figure 2.4: Example of an Indoor gateway[8].

Figure 2.5: Example of an outdoor gateway[9].

Among the LPWAN technologies, LoRaWAN has become one of the most popular and widely adopted solutions, of which radio frequency (RF) signal modulation is based on the chirp spread spectrum. With chirp standing for 'Compressed High Intensity Radar Pulse' which signifies a signal whose frequency either increases or decreases with time. It is very common is sonar and radar [27].
In the LoRa frequency shift chirp spread spectrum, chirp signals have a constant amplitude and use the entire bandwidth in a linear way from one end to another end in a certain time. If the frequency changes from lowest to highest, it is called up-chirp as shown in figure 2.6 and if the frequency changes from highest to lowest, its called down-chirp.

The chirp spread spectrum techniques helps in transmitting signals for very large distances [10].

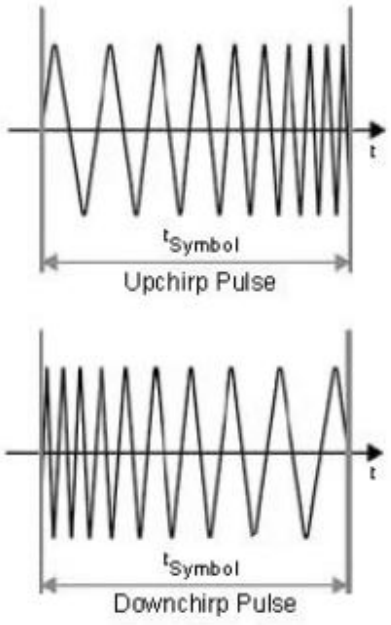

Figure 2.6: Example of an up-chip and down-chip[10].

It as well offers more benefits summarised below:

1. Resistance to interference: The wide bandwidth and spreading nature of chirp spread spectrum makes it highly resilient to narrow-band interference. It allows the communication system to maintain signal quality even in the presence of other signals or noise.

2. Robustness in multipath environments: Chirp spread spectrum is effective in combating multipath propagation, where signals reach the receiver via multiple paths, causing delay and phase differences. The spread signal helps to mitigate the effects of multipath by spreading the energy across a wide frequency range.

3. Improved range and coverage: The wide bandwidth characteristic of chirp spread spectrum enables longer communication range and improved coverage compared to narrowband systems. It increases the chances of successful data transmission even in challenging environments.

As shown in the figure 2.8, LoRa physical layer includes 8 up-chirp preamble symbols used for detecting LoRa symbols, frame synchronization and frequency synchronization. Every symbol being transmitted consists of one or more bits of data for example, a symbol could be 1111111 representing decimal 127. The second part in the LoRa physical layer are 2 down-chirp synchronisation symbols used for timing synchronisation. Finally the last section is the physical payload and optional CRC.

Spreading Factor (SF)	Chip Length 2^{SF}
7	128
8	256
9	512
10	1024
11	2048
12	4096

Figure 2.7: Chirp length variation with Spreading factor.

During transmissions, Spreading Factor 8 takes exact twice the time of Spreading factor 7 and Spreading Factor 9 takes exact twice time of Spreading Factor 8. Variations in the spreading factor as well has a significant impact on the chirp length as illustrated in figure 2.7.
Symbol Rate (Rs), Bandwidth (B) and Spreading Factor (S) relation can be shown in this equation below.

$$Rs = B/2^s$$

The higher the Spreading Factor,the higher the over-the-air time of the symbol. The lower the Spreading Factor, the higher the Data Rate. Having the spreading factor and the Bandwidth (B), we as well get the symbol period given by.

$$Ts = 2^s/B$$

2.2 LoRa Modulation.

LoRa modulation has been originally stated in terms of the instantaneous frequency based on chirp spread spectrum (CSS) [28]. It is an M-ary digital modulation technique, where the M possible waveforms at the output of the modulator are modulated signals over the frequency interval $(f_0 - B/2, f_0 + B/2)$with M different initial frequencies. For the data, the instantaneous frequency is increased steadily, and after wrapped to $f_0 - B/2$ when it reaches the maximum frequency $f_0 + B/2$, an operation that mathematically can be seen as a reduction modulo B.
$BT_s = M$, where T_s is the symbol interval and M is an integer value while B being the bandwidth. LoRa uses three different bandwidth: $125kHz, 250kHz$ and $500kHz$. The modulation bit-rate can be given as:

$$M_b = \log_2 M/T_s$$

which simplifies to:

$$M_b = S/T_s = BS/2^S$$

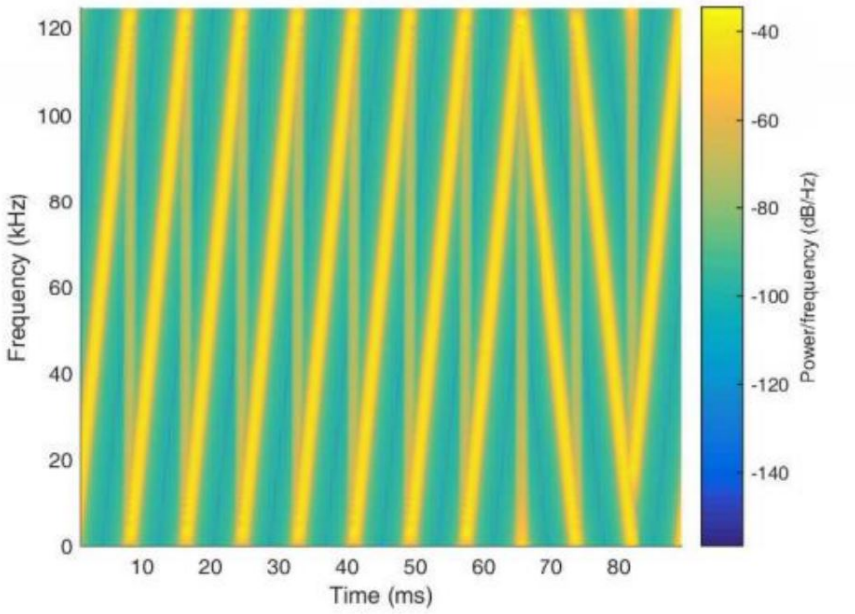

Figure 2.8: Example of LoRa physical layer [11].

To obtain the modulation spectral efficiency E, we get the ratio between M_b and the chip rate C_r.

$$C_r = M/T_s = B$$

Which bring us to

$$E = S/2^S$$

2.3 Lora block diagram.

In Lora, data are encoded before being transmitted in particular four operations are applied on the input bits that is; whitening, hamming encoding,interleaving and reverse gray coding. This can be demonstrated in the figure 2.9.

2.3.1 Gray coding.

Gray coding is performing a mapping between a symbol in numeric representation to a binary sequence[12]. The particularity of the attained binary sequence is that adjacent symbols in the original representation only differ by one bit in the gray representation. This property of gray coding finds application in many wireless communication modulations where it is more likely to mistake a symbol by an adjacent one than another random one.
Gray coding is as well commonly used in error correction, in analog-to-digital and digital-to-analog converters, and other applications where minimizing the possibility of errors during transitions between consecutive values is crucial.

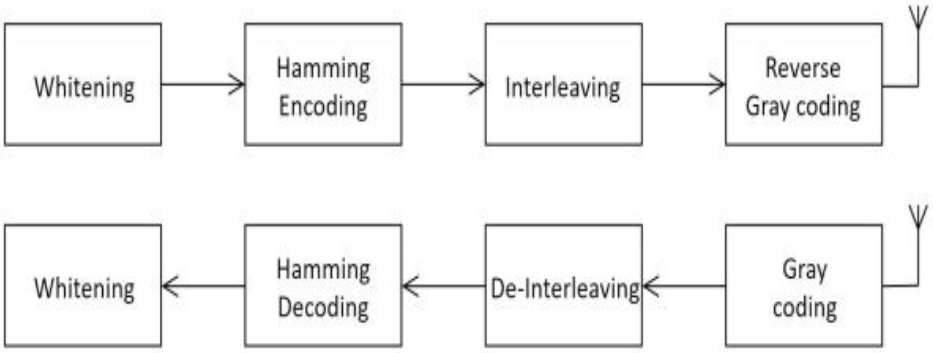

Figure 2.9: LoRa Encoding / Decoding [12].

2.3.2 Interleaving.

During signal transmissions, a symbol can be corrupted by noise or fading thus leading to multiple bits errors after demodulation because all errors are caused by one symbol, since they are linearly related. The issue with such errors is that they are not well corrected by error correction codes designed to rectify random errors in code word.
So the goal is to break this linear relation between the erroneous bits by distributing the errors over multiple code words. This operation improves the effectiveness of the error correction code with a penalty of increased latency brought about by the necessity of receiving multiple symbols before being able to recover one full code word. By distributing errors across different packets or frames, interleaving allows for more efficient error correction techniques to be employed. Error correction codes, such as Reed-Solomon codes or convolutional codes, can be applied to each packet or frame individually. If errors occur during transmission, the error correction codes can detect and correct them within each packet or frame. This approach is more effective compared to relying solely on error correction codes to handle burst errors, as the codes are optimized to handle isolated errors rather than bursts.

2.3.3 Whitening.

Whitening is performed by XORing bits with a pseudo random sequence with the goal of removing DC-bias in the transmitted data. Unlike Manchester coding, it provides the advantage of keeping the same data rate at the cost of not having the guarantee of removing any DC-bias but only a very high probability of doing it.
There are various methods to achieve whitening, depending on a particular application. Some common techniques include:

1. Using equalization filters: In digital communication systems, equalization filters are used to compensate for frequency-dependent attenuation or amplification. The filters modify the received signal to ensure that

all frequencies are treated equally, improving the overall signal quality. The filter architecture is such that the frequency response parameters are independently related to the multiplier coefficients [29].

2. Applying decorrelation filters: These filters are deployed to remove correlation between adjacent symbols in a data stream. A simple approach to decorrelation filtering is to convolve the input signal with a short white-noise sequence [30]. By decorrelating the symbols, the data becomes more independent, which can lead to better performance in applications such as channel coding, modulation, or pattern recognition.

3. Pre-emphasis and de-emphasis: In signal processing, pre-emphasis is applied to boost higher frequencies before transmission, compensating for the natural attenuation of high frequencies in the communication channel. At the receiver end, de-emphasis is performed to restore the original frequency balance.

4. Using linear prediction: This models a signal as a linear combination of past samples. By estimating the correlation between samples, the prediction coefficients can be adjusted to eliminate the correlation, resulting in a whitened signal.

By integrating whitening, LoRa communication systems can improve the efficiency of subsequent processing stages, enhance data transmission quality, and reduce the impact of signal biases or correlations.

2.3.4 Hamming code.

Hamming code is an error correcting code used to detect and correct errors in transmitted data. Its able to correct errors by adding redundancy in each code word. The main concept behind the Hamming code is the use of parity bits to detect and correct errors. Parity is a simple method of error detection that involves adding an additional bit to the data to ensure that the total number of ones in the code word is even or odd, depending on the parity scheme employed. In a Hamming code, the positions of the parity bits are determined by the powers of 2. For example, in a 7-bit Hamming code, the positions of the parity bits are 1, 2, and 4, while the data bits occupy the remaining positions (3, 5, 6, and 7). During transmission, if an error occurs and flips a bit in the code word, the parity bits can be used for detecting the error. By comparing the calculated parity bits with the received parity bits, the presence of an error can be detected. When an error is detected, the Hamming code can as well correct it by flipping the bit at the position indicated by the parity bits.

2.4 Description of the LoRa signal in Continuous time.

To describe the LoRa signal model in the time domain we start by assuming the frequency over which the signal varies is from 0 to B. For the time interval 0 to T_s, the instateneous frequency of a symbol in LoRa can be given as.

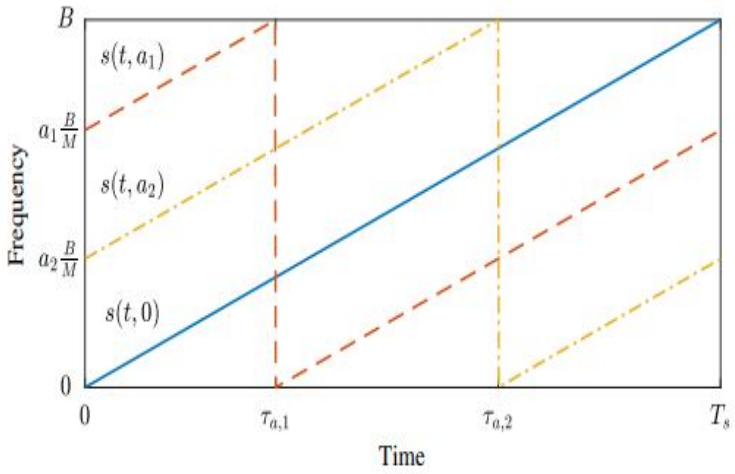

Figure 2.10: Two LoRa modulated waveforms[13].

$$f(t;a) = aB/M + Bt/T_s (mod B)$$

which further simplifies to (where M is the total number of symbols):

$$f(t;a) = aB/M + Bt/T_s - Bu(t - t_a) \quad 0 \leq t < T_s$$

where aB/M is the starting frequeny that depends on the modulating symbol and t_a is the time instant where after a linear increase and $u(t-t_a)$ is the change in time. The instantaneous frequency reaches the maximum for the remaining part of the symbol time interval the instantaneous frequency is still increasing but reduced by deducting F. As shown in figure 2.10 were we have two LoRa waveforms for arbitrary symbols $s(t, a_1)$ and $s(t, a_2)$ the starting frequencies are $a_1 \frac{B}{M}$ and $a_2 \frac{B}{M}$ respectively.

$$t_a = T_s(1 - a/M)$$

Let us assume that the modulation begins at time t = 0 and take the time interval $t \in [0, T_s]$. The phase which is denoted as $R(t;a)$ can be obtained by taking the integral of the instantaneous frequency

$$R(t;a) = 2\pi \int_0^t f(t;a)\, dt$$

$$= 2\pi \left[aBt/M + Bt^2/2T_s - B(t - t_a)u(t - t_a) \right]$$

With the above instantaneous phase, the LoRa signal with energy E_s is described by.

$$s(t,a) = \sqrt{\frac{E_s}{T_s}} exp[jR(t;a)]$$

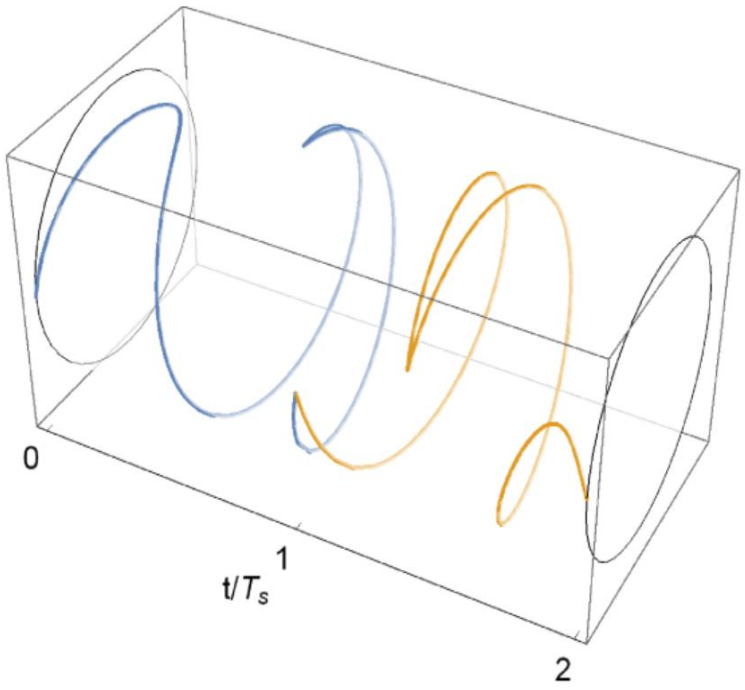

Figure 2.11: Phase as a function of time[13].

Lora modulation has the property of being a memoryless and continuous phase modulation. The transmitted waveform in each symbol interval only depends on the symbol in that interval, and not on previous or successive symbols. This can be visualized in the phase diagram which keeps tracks of the phase evolution over time. In figure. 2.11, the phase diagram for two consecutive LoRa modulated symbols is shown as a function of time. It can be noted that each waveform starts and ends with the same phase.

2.5 Discrete-time description.

In the discrete time domain, a LoRa signal can be represented as a sequence of samples, where each sample corresponds to a specific time instant. The discrete time representation is commonly used for digital signal processing and analysis. For example, in the discrete time domain, LoRa signals often undergo digital filtering to achieve various objectives such as noise reduction, interference rejection, and bandwidth shaping. Thus by representing LoRa signals in the discrete time domain, it becomes compliant to the different digital signal processing techniques and applications.

For receiver implementation, it has been proposed to sample the received signal at chip rate T_c. Which is given by $T_c = T_s/M$ seconds. So in the interval [0 to T_s] we have samples.

$$x(kT_c; a) = exp\left[\frac{j2\pi BkT_s}{M}\left[\frac{a}{M} - \frac{1}{2} + \frac{BkT_s}{2M^2} - u(\frac{KT_s}{M} - \frac{M-a}{B})\right]\right]$$

$$= exp\left[j2\pi k\left[\frac{a}{M} - \frac{1}{2} + \frac{k}{2M} - u(\frac{KT_s}{M} - \frac{M-a}{B}\right]\right]$$

$$= exp\left[j2\pi k\left[\frac{a}{M} - \frac{1}{2} + \frac{k}{2M}\right]\right]$$

where k takes on values from 0 to (M - 1).

Chapter 3

System Model

In the third chapter, I present mostly my practical work which mainly contains the simulations I performed. I present current algorithms in an abstract way and relate my work to the theoretical concepts introduced in chapter two and in the reference papers.

3.1 Preamble generation.

As highlighted before in the previous chapter, the preamble serves as a synchronization [31] and detection mechanism, allowing receivers to identify the presence of a LoRa transmission and prepare for data reception. To allow for the receiver's radio to correctly synchronize with the transmitter's radio, a preamble has to first be generated. The transmitting device broadcasts the preamble at the beginning of a communication session to allow other devices in the vicinity to detect and synchronize with the transmission. Receivers within the range of the transmitting LoRa device listen for the preamble signal. Once synchronized, the receivers can prepare for data reception and demodulation.

I first performed the simulation to generate a preamble consisting of 8 up-chips which I generated at a bandwidth of 125kHz and Spreading factor (SF) 7. So when I detect the start of the preamble, I detect starting point of an incoming packet. SF being the ratio between symbol rate and chip rate. It controls the chirp rate, and thus controls the speed of data transmission, it also provides a trade-off between data rate and range with lower spreading factors providing faster chirps and therefore higher data transmission rates. On contrary a higher spreading factor results in a longer symbol period and a lower data transmission rate. It also provides increased resistance to noise and interference, allowing for longer communication range.
Figure 3.1 illustrates the generated preamble. The simulation was done in Matlab using the scripts in the Appendix section. Within my preamble generation simulation, I used the following parameters below:

- $SF = 7$
- $BW = 125kHz$

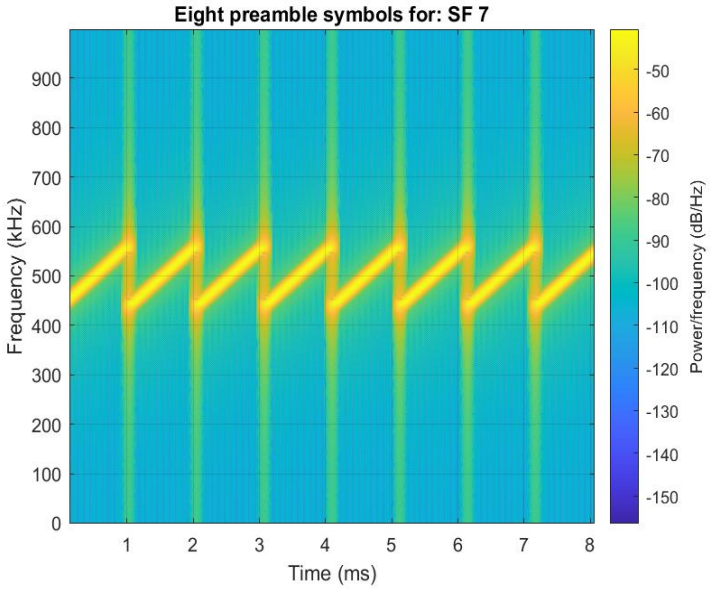

Figure 3.1: Simulation of the 8 up-chip preamble

- sampling frequency $FS = 10^6$
- number of samples $= FS\frac{2^7}{BW} = 1024$ samples

3.1.1 Simulation of the LoRa communication.

The simulation of the communication system consist of generating random bits, these bits are then LoRa modulated then they go through to the noise channel and finally demodulating this signal to bits again. I applied the following parameter below while performing the simulation. Since the simulation was performed in MATLAB, in the appendix section I include the simulation algorithms.

- $SF = 7, 8, 9, 10, 11, 12$
- $BW = 125kHz$
- Preamble Length $= 8$
- Synchronisation symbols.
- Sampling Frequency $Fs = 125kHz$
- Number of samples $= Fs\frac{2^7}{BW}$ samples

The simulated algorithm is described in the summarised steps below:

1. Bits Generation: Initially, I generated random bits that are to be transmitted. I applied random number generation libraries from MATLAB that aided in providing functions to generate random bits with high-quality randomness suitable for application.

2. Binary to Gray, Gray to Decimal: In this step I group the bits depending on their spreading factor, then they are converted to Gray codification system and, finally, they are converted to decimal numbers.

3. LoRa Signal Generation: I perform the generation of the LoRa signal with its respective preamble, synchronization and data information. To generate the signal to be transmitted, I specified parameters to define the modulation scheme. These parameters included the frequency band, spreading factor (SF), bandwidth, and sampling rate.

4. AWGN Channel: Additive White Gaussian Noise (AWGN) is added to the signal coming through the channel.This was applied as a model for simulating the effects of real-world noise in the communication system performance evaluations. For example, evaluating the performance such as determining the bit error rate (BER).

5. Multiplication Step: This spep Comprises of multiplication between the transmitted signal and the purely DownChirp signal, which is the conjugate of the signal UpChirp.

   Figure 3.2 on the right illustrates the results obtained from the multiplication.

6. Fourier transform Calculation: With the use of this transform to manipulate the signal, I extract information from the data. Its necessary to perform the Fourier transform which allows for the detection of the information from the amount of energy in the signal.

7. Decoding Data: Involves detection of every peaks presents in the Fourier transformed signal and later extract the corresponding values. This process is essential in ensuring that the received data is correctly interpreted and usable by the receiver.

8. Decimal to Gray, Gray to Binary: Is the reverse process of the second step described before.

9. Received Bits: This is the final step corresponding to receiving of the bits transmitted.

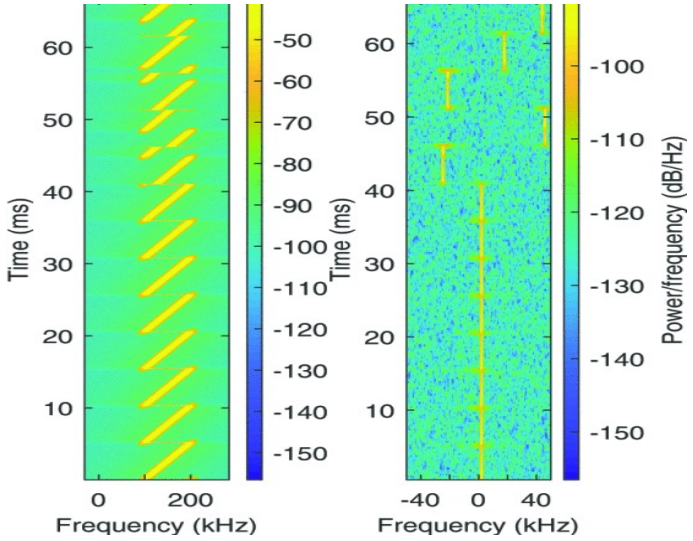

Figure 3.2: (Left) The up-converted transmitter signal. (Right) Results after multiplication of the transmitted signal with the base down-chip signal[14].

Chapter 4

Realisation and Evaluation

4.1 BER Performance of the LoRa MATLAB simulation.

In order to evaluate the efficiency of my LoRaWAN simulated system, I made use of the bit error ratio technique. Since LoRaWAN uses several spreading factors (SF) to control the bit rate, improve the range, and decrease the energy consumption. I performed my simulation on all spreading factors from; 7 to 12. The bit error ratio (BER) is estimated as the number of bit errors divided by the total number of bits transmitted during a specific time interval. Its a unit-less performance measure, often expressed in percentage form.
The probability of BER is the expected value of the bit error ratio. The bit error ratio can also be considered as an approximate estimate of the bit error probability. This estimate is approximate accurate for long time intervals and a high number of bit errors.
As detecting a LoRaWAN signal with oversampling requires adjustments to the data detection procedure, it was worthwhile to examine the performance of the resulting method. Hence, I performed bit error rate simulations at different signal to noise ratio (SNR) values for an additive white Gaussian noise (AWGN) channel.
In my transmission channel, the number of bit errors is the number of received bits of the data stream over the noisy communication channel that have been altered due to noise, distortion or bit synchronization errors. The SNR defined as E_s/N_0M, where E_s is the energy of the signal and N_0M is the power spectral density of the noise [32].

4.1.1 Algorithm description.

I developed my algorithms using MATLAB software which can be accessed in the Appendix section. Below is a step by step description of its operation.

- To have an average running time which is not too long for the algorithm, I initialise my iterations to 10 and at every iteration, I perform the following steps.

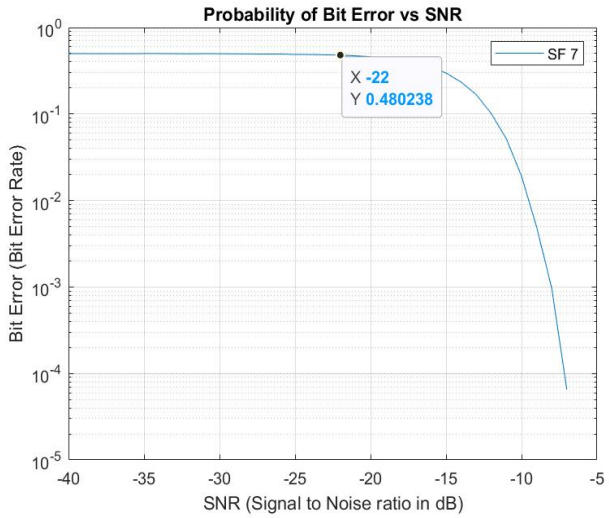

Figure 4.1: BER performance curve at SF = 7

- I generate the symbols to be transmitted following the LoRaWAN modulation technique.
- At every iteration from 1 to 10, I add additive white Gaussian noise to the transmitted symbols for different SNR values.
- Multiply the transmitted symbols with a reverse chip to obtain the decoded signal.
- Perform the Fourier Transform.
- Decode the received data.
- At every iteration and SNR, I perform the BER by summing the difference between bits received from the initially transmitted bits then divide by the total number of bits.
- I initialise a matrix of dimensions number of iterations by SNR which stores the calculated BER.
- In my case I performed 10 iterations.
- Then I calculated the mean value for the BER at every SNR and obtain a 1 by 35 matrix.

For clear visualization I present Figures. 4.1, 4.2 and 4.3, to show the generated Bit Error Rate performance curves for SF 7, SF 9 and SF 12 respectively. As expected, BER gradually decreases smoothly with the increase in SNR. The results of the MATLAB simulations where then compared to the approximate analytical results for LoRa Bit Error Rate performance presented in [15] also shown in figure 4.8. The simulated Bit Errror Rate results and those predicted through the closed-form expressions in [15] follow the same pattern.

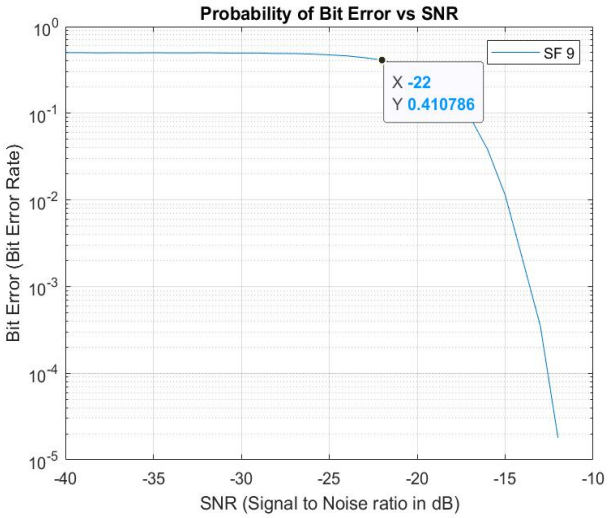

Figure 4.2: BER performance curve at SF = 9

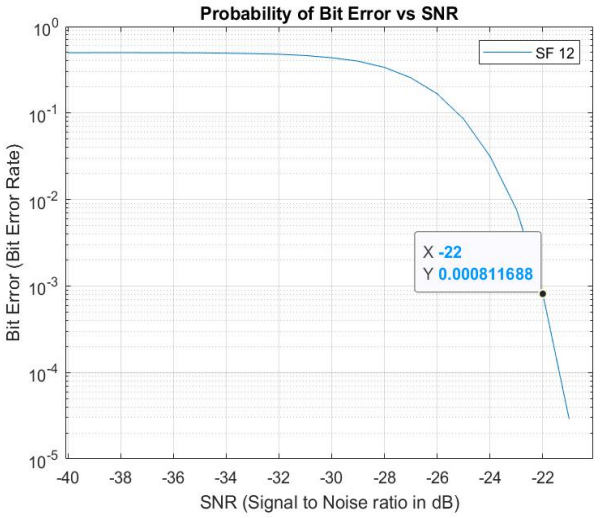

Figure 4.3: BER performance curve at SF = 12

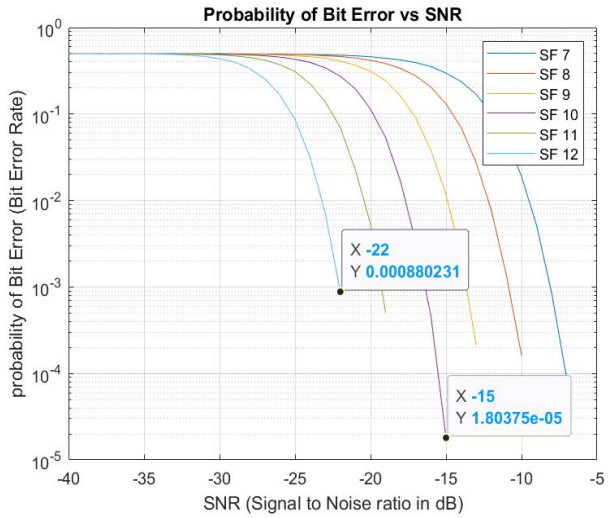

Figure 4.4: BER performance curves at all the SFs

4.2 Preamble detection and Synchronization.

As explained before, one of the most significant initial tasks in the operation of LoRaWAN systems is to detect the presence of the preamble, also commonly known as the preamble detection [33]. This allows for the receiver's radio to start synchronization with the transmitter's radio. Within [34], theoretically the optimal performance of LoRa preamble detection was investigated. To achieve the described optimal performance, the complexity of the hardware needs to increase inevitably, eventually leading to incomplete compatibility with LoRa.

Moreover, in [34] the impact of fading channel, which is very significant in a realistic wireless propagation environment was not taken into account during the preamble detection process. From in-depth research, a more compatible strategy with LoRaWAN preamble detection is the threshold based preamble detection technique as described in [33] which tries to directly discover the presence of the preamble by following a standard reception technique of LoRa, that is; the process of de-chirping, followed by taking the discrete Fourier transform (DFT). The key to the threshold-based LoRa preamble detection is comparing the detection results to some thresholds which need to be optimised for a better preamble detection process. In [33] and [**?**], few simple rules of thumbs were brought up for investigating the thresholds; however, the concluding performance is not satisfactory due to their rigid sub-optimalities (i.e., lack of taking into account of the optimization of thresholds). Also, the issue of the false alarm was ignored and not taken into consideration.

After citing all these challenges, and to have a better performance of the threshold-based preamble detection, In my simulation I aim at optimizing the detection thresholds by maximizing the detection probability of the preamble by having it as close to 1 as possible for all the spreading factors.

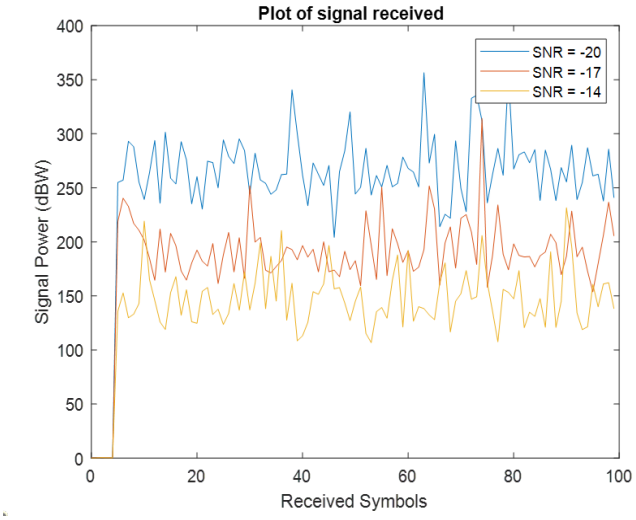

Figure 4.5: Detecting the start of the preamble at SF 7.

For accurate detection of the start of the preamble, I performed the following steps:

- I sample the received signal $r(t)$ with a sampling frequency $f_s = B \cdot$ OSF. The Over Sampling Factor (OSF) having an integer value such as; 2,3,4. In my case I used 4.
- I multiply a window of $N = K \cdot$OSF samples with a base down-chirp. With K being an integer value.
- I computes the absolute value of the FT (Fourier transform) for the mixed signal.
- At the receiver, I plot out the received signal and find the position where I get a sharp rise in the received signal.
- If the detected position is detected continuously for a number of times (e.g., 3 consecutive windows), then the algorithm declares the presence of a preamble.

I carried out this procedure continuously, for each possible spreading factor at different Signal to Noise ratio values. Here in figures 4.5, 4.6 and 4.7 I illustrate results for Spreading Factors (7,8 and 12). The algorithm succeeded in detecting the presence of the preamble even several dBs below the sensitivity threshold.

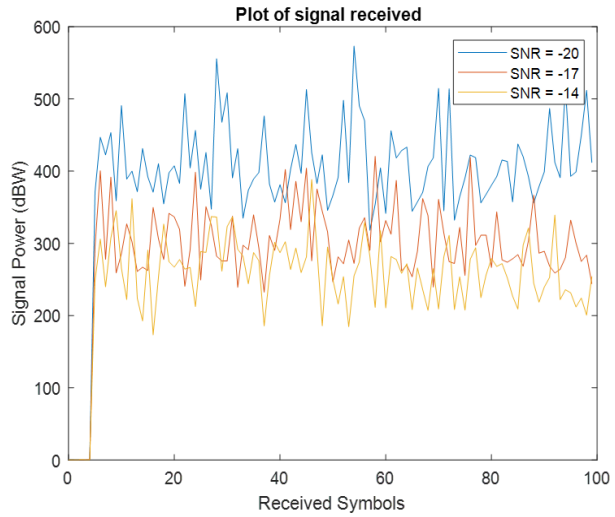

Figure 4.6: Detecting the start of the preamble at SF 8.

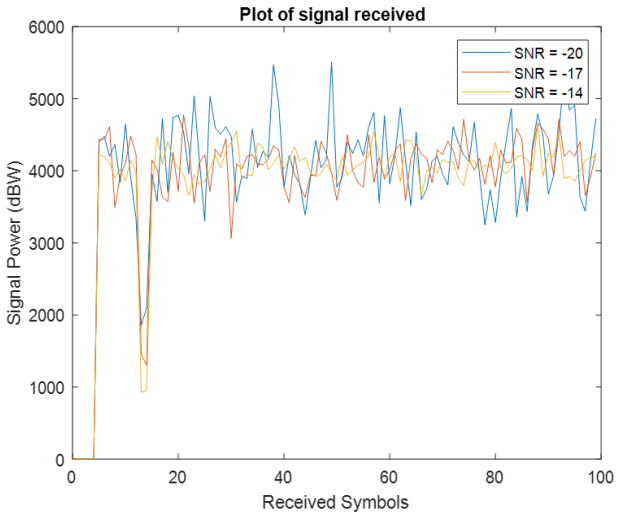

Figure 4.7: Detecting the start of the preamble at SF 12.

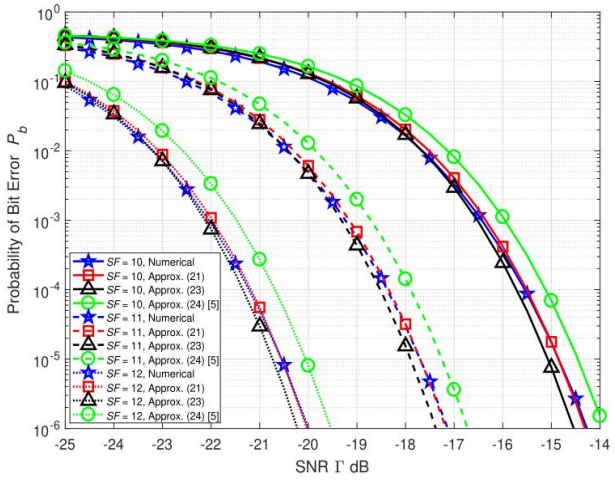

Figure 4.8: Theoretical BER performance Vs SNR at SF 10, 11, 12 [15].

4.3 Results.

4.3.1 Performance evaluation for BER in AWGN channels.

In this section, I evaluate the performance of my MATLAB simulations at varying channel quality that is; varying noise levels. In BER performance scenario, I confirm that the simulation algorithm delivers performance close to the theoretical results in [15]. This is because the generated curves exhibit an extremely good accuracy of the derived approximations when compared with the theoretical results. Also since the signal quality becomes better as SNR increases, it practically accurate to have a steadily decreasing BER pattern with increasing SNR.
I compared my results with the theoretical curves in [15]. Figure 4.8 presents the curves derived from [15] indicating comparisons of the derived LoRa BER approximations against the theoretical BER performance computed at SF=10,11,12. In comparison with my BER performance curves generated in my MATLAB simulation as shown in figure 4.9 for the same spreading factors, the results exhibit an extremely accurate approximation with coinciding curves.

Secondly, I check how good the preamble detection algorithm is by simulating the probability of detection as the SNR changes which corresponds to different radio link qualities. As shown in figures 4.10 and 4.11, the probability of detection gradually increases with increase in SNR levels.
With these simulation results, it can surely be concluded that the my MATLAB algorithms succeed in performing an accurate BER performance evaluation and an evaluation of the preamble detection.

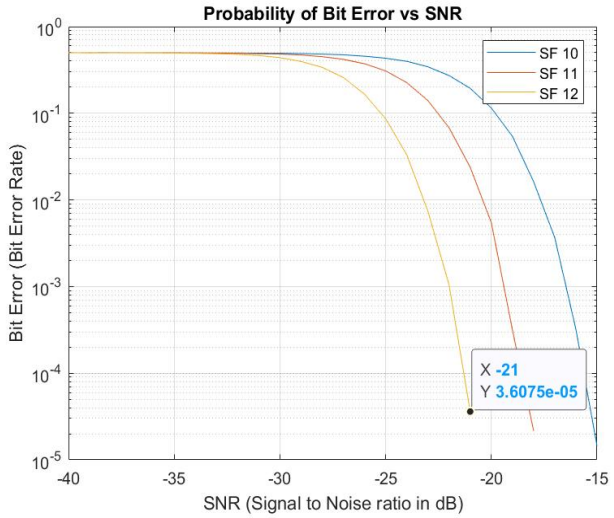

Figure 4.9: BER simulation in MATLAB for SF 10, 11, 12.

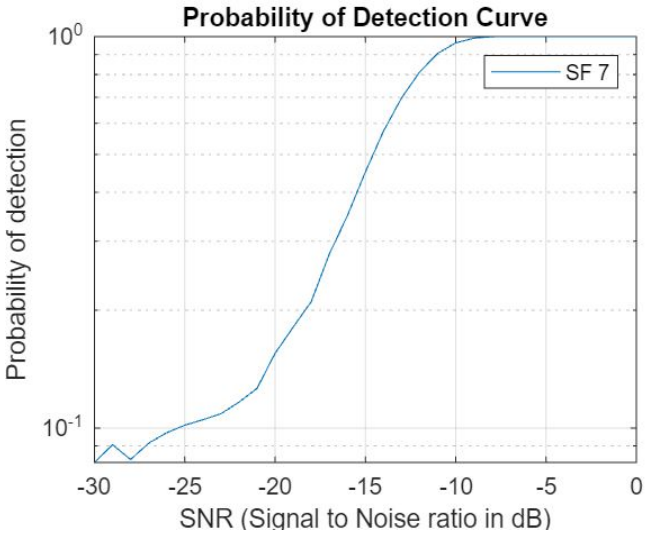

Figure 4.10: Probability of detection Vs SNR at SF 7.

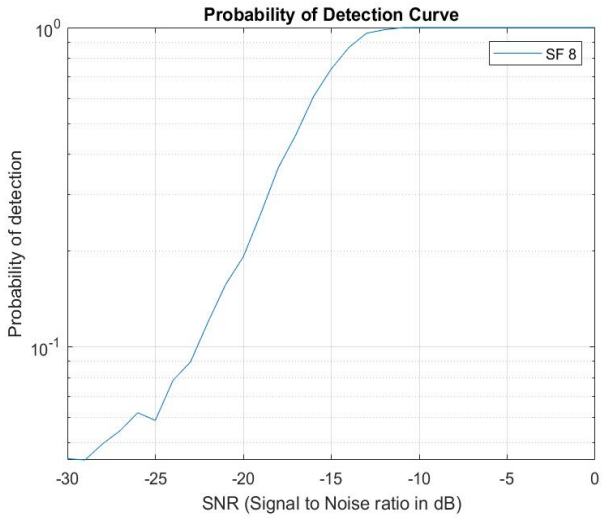

Figure 4.11: Probability of detection Vs SNR at SF 8.

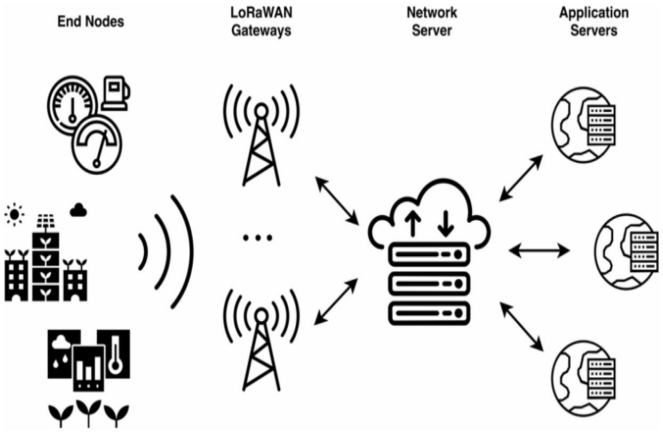

Figure 4.12: Multiple users in the network [16].

Chapter 5

Conclusions and Future Work

In the thesis, I implemented a LoRaWAN preamble detection approach. Throughout the whole implementation, I simulated algorithms to modulate and transmit the LoRaWAN signal then performed detection of the preamble. Evaluation was done with the probability of detection approach, which demonstrated fairly accurate results with the variation in signal to noise ratio.
A closed form approximation for the BER performance in Additive White Gaussian noise channel environments has as well been presented too. With variation in channel conditions (noise) comparison with numerical results affirmed a quite accurate approximation of the presented analysis. Table 5.1 illustrates a brief summary of some results obtained from the simulation.

My approach was based on a single user/transmitter and receiver, future work can be done with carrying out multiple transmissions from many users simultaneously as depicted in figure 4.12 and perform multiple preamble detections to overcome the multiple user interference problem. Most of the work and algorithms I presented in my work are perfectly suitable in attaining the future work.

Table 5.1: Table highlighting some of the simulation results

SNR	BER at SF-7	BER at SF-10	BER at SF 11	BER at SF 12
-30	0.498	0.493	0.480	0.437
-25	0.493	0.431	0.304	0.086
-22	0.479	0.273	0.067	0.00086

Bibliography

[1] S. Devalal and A. Karthikeyan, "Lora technology - an overview," *2018 IEEE 2nd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 284–290, 2018.

[2] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, "Lorawan — a low power wan protocol for internet of things: A review and opportunities," *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, pp. 1–6, 2017.

[3] P. S. Cheong, J. Bergs, C. Hawinkel, and J. Famaey, "Comparison of lorawan classes and their power consumption," *2017 IEEE Symposium on Communications and Vehicular Technology (SCVT)*, pp. 1–6, 2017.

[4] A. S. Rawat, J. Rajendran, H. Ramiah, and A. Rana, "LoRa (Long Range) and LoRaWAN Technology for IoT Applications in COVID-19 Pandemic," *IEEE Access*, pp. 419–422, 2020.

[5] W. Lv, F. Meng, C. Zhang, Y. Lv, N. Cao, and J. Jiang, "A General Architecture of IoT System," *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1, pp. 659–664, 2017.

[6] S. Deepak, H. Anandakumar, S. Pavithra, V. Keerthika, and K. Nandhini, "Performance analysis of star topology for small networks using riverbed," *2022 IEEE 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 2108–2111, 2022.

[7] B. Al Homssi, K. Dakic, S. Maselli, H. Wolf, S. Kandeepan, and A. Al-Hourani, "IoT Network Design Using Open-Source LoRa Coverage Emulator," *IEEE Access*, vol. 9, pp. 53636–53646, 2021.

[8] https://www.thethingsnetwork.org/docs/lorawan/architecture/ *Online source.*

[9] M. M. Erbati, G. Schiele, and G. Batke, "Analysis of lorawan technology in an outdoor and an indoor scenario in duisburg-germany," *2018 IEEE 3rd International Conference on Computer and Communication Systems (ICCCS)*, pp. 273–277, 2018.

[10] N. Technology, "Chirp spread spectrum," *Chirp Spread Spectrum (CSS). url:https://nanotron. com/EN/co$_t$echn − css − php/.*, 2021.

[11] N. V. E. Aiju Thomas, "Chirp spread spectrum for narrow band long range bio sensor networks," *IEEE Access, Chirp Spread Spectrum For Narrow Band Long Range Bio Sensor Networks*, vol. 9, pp. 1–6, 2020.

[12] T. Joachim, "Complete reverse engineering of lora phy," *IEEE Telecommunications Circuits Laboratory Internet of Things Journal*, pp. 1–1, 2020.

[13] M. Chiani and A. Elzanaty, "On the LoRa Modulation for IoT: Waveform Properties and Spectral Analysis," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8463–8470, 2019.

[14] S. Robson and A. M. Haddad, "On the use of lora for power line communication," *2019 54th International Universities Power Engineering Conference (UPEC)*, pp. 1–6, 2019.

[15] T. Elshabrawy and J. Robert, "Closed-Form Approximation of LoRa Modulation BER Performance," *IEEE Journal of communications letters*, vol. 22, 2018.

[16] J. Sanchez-Gomez, J. Gallego-Madrid, R. Sanchez-Iborra, and A. F. Skarmeta, "Performance study of lorawan for smart-city applications," *2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 58–62, 2019.

[17] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei, "Smart home system based on iot technologies," *2013 IEEE International Conference on Computational and Information Sciences*, pp. 1789–1791, 2013.

[18] S. Al-Sarawi, M. Anbar, R. Abdullah, and A. B. Al Hawari, "Internet of things market analysis forecasts, 2020–2030," *2020 IEEE 4th World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 449–453, 2020.

[19] H. Bhatia, S. N. Panda, and D. Nagpal, "Internet of things and its applications in healthcare-a survey," *2020 IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 305–310, 2020.

[20] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[21] R. S. Krishnan, U. Narmatha, N. Preethi, G. Subhashini, J. R. Francis Raj, and K. L. Narayanan, "Iot powered smart cars (isc)," *2023 IEEE International Conference on Inventive Computation Technologies (ICICT)*, pp. 1351–1356, 2023.

[22] A. Nessa, F. Hussain, and X. Fernando, "Adaptive latency reduction in lora for mission critical communications in mines," *2020 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–7, 2020.

[23] H. Y. C. J. Ma YW, Lai CF, "Mobile RFID with IPv6 for phone services," *IEEE journal, 2009. 169 170.*, 2009.

[24] S. Jha and N. M. Balasubramanya, "Multi-user detection and data association for lora-based uav iot networks," *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 1170–1175, 2022.

[25] W. M. B. Reynders and S. Pollin, "Range and coexistence analysis of long range unlicensed communication," *2016 IEEE 23rd International Conference on Telecommunications (ICT)*, pp. 1–6, 2016.

[26] I. Y. S. C. S. H. K. Tsai, F. Leu and H. Park, "Low-power aes data encryption architecture fora lora wan," *IEEE Access*, vol. 7, pp. 146348–146357, 2019.

[27] R. Z. Thamrin, O. N. Samijayani, S. Rahmatia, D. Adrianto, and I. K. A. Enriko, "Implementation of LoRa End-Device in Sensor Network System for Indoor Application," *2020 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*, pp. 208–212, 2020.

[28] T. Elshabrawy and J. Robert, "Closed-form approximation of lora modulation ber performance," *IEEE Communications Letters*, vol. 22, pp. 1778–1781, 2018.

[29] P. Regalia and S. Mitra, "Tunable digital frequency response equalization filters," *IEEE Access, Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 1, pp. 118–120, 1987.

[30] T. N. Galen Reich, Chris Pike, "Development and analysis of decorrelation filters for binaural rendering of audio definition model content," *IEEE Access*, no. 1, pp. 1–10, 2021.

[31] M. Xhonneux, J. Tapparel, A. Balatsoukas-Stimming, A. Burg, and O. Afisiadis, "A maximum-likelihood-based two-user receiver for lora chirp spread-spectrum modulation," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22993–23007, 2022.

[32] B. T. Bosworth, W. R. Bernecky, J. D. Nickila, B. Adal, and G. C. Carter, "Estimating Signal-to-Noise Ratio (SNR)," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 414–418, 2008.

[33] D. B. M. Xhonneux, O. Afisiadis and J. Louveaux, "A low-complexity LoRa synchronization algorithm robust to sampling time offsets," *IEEE Access*, vol. 9, p. 3756–3769.

[34] J.-M. Kang, D.-W. Lim, and K.-M. Kang, "On the LoRa Modulation for IoT: Optimal Preamble Detection and Its Performance Analysis," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

Appendix A

MATLAB scripts

Algorithms used in the project implementation

A.0.1 In the following block, I present the Lora symbol modulation code highlighting the main operations done to modulate the LoRa signal.

```
1
2  %LoRa Signal modulation
3
4  function out_preamble =
5  LoRa_Modulation(SF,BW,Fs,num_samples,symbol,inverse)
6
7      %initialization
8      phase = 0;
9      Frequency_Offset = (Fs/2) - (BW/2);
10
11     shift = symbol;
12     out_preamble = zeros(1,num_samples);
13
14     for k=1:num_samples
15
16         %output the complex signal
17         out_preamble(k) = cos(phase) + 1i*sin(phase);
18
19         % Frequency from cyclic shift
20         f = BW*shift/(2^SF);
21         if(inverse == 1)
22                f = BW - f;
23         end
24
25         %apply Frequency offset away from DC
26         f = f + Frequency_Offset;
27
28         % Increase the phase according to frequency
29         phase = phase + 2*pi*f/Fs;
```

```
30          if phase > pi
31              phase = phase - 2*pi;
32          end
33
34          %update cyclic shift
35          shift = shift + BW/Fs;
36          if shift ≥ (2^SF)
37              shift = shift - 2^SF;
38          end
39      end
40  end
```

A.0.2 In the following function, I generate the random numbers to be used as symbols during transmission.

```
1  % Inputs:
2  % total_sym: Total no. of random bits to be generated
3  % SF: Spreading Factor
4  % Output:[random_number_input, columns] that appends ...
       random_number_input with columns
5  % Input_sample: Random number in decimals
6
7  function [random_number_input, columns] = ...
       LoRa_random_number_generation(total_sym, SF)
8
9  rows = SF;
10 columns = ceil(total_sym/SF);
11 random_number_input = round(0.75*rand(1,rows*columns))';
12 % generates a rondom arrray of numbers betweeen (0 and 1) of ...
       size 1 by rows*columns
```

A.0.3 The following function performs the Binary to Gray conversion where I convert the randomly generated symbols in bits to Gray format.

```
1  %% Binary to Gray Conversion
2
3  function [Input_sample_gray] =  binary2gray(Input_sample_Bi)
4
5  [r,c] = size(Input_sample_Bi);
6  Input_sample_gray = zeros(r,c);
7
8  Input_sample_gray(1,:) = Input_sample_Bi(1,:);   % Copying ...
       First bit
9
10 for m = 1:1:c
11     for g = 2:1:r      % Xor of input bit with last input bit
12         Input_sample_gray(g,m) = xor(Input_sample_Bi(g,m), ...
               Input_sample_Bi(g-1,m));
```

```
    end
end
```

A.0.4 The following algorithm below performs the simulation of Bit Error Rate (BER) performance against Signal to Noise Ratio (SNR) for all spreading factors 7 to 12.

```

%%
clear all; close all; clc;
%% main parameters of the simulator

SF = 7:1:12;                    % Spreading Factor
BW = 125000;                    % 125kHz
Fs = 125000;                    % Sampling Frequency
preamble_len = 8;               % Preamble length
sync_len = 2;                   % Sync length
total_bits = 27720;             % total bits to be transmitted ...
    in LoRa message
SNR_dB = -40:1:-5;              % SNR in DB
SNR = 10.^(SNR_dB/10);          % SNR
Total_iterations = 10;
BER = zeros(Total_iterations,length(SNR_dB));

for sf = 1:1:length(SF)

    num_samples = Fs*(2^SF(sf))/BW;  % Number of samples

    %% Random Number Generation
    [Input_sample_Bi, input_len] = ...
        LoRa_random_number_generation(total_bits,SF(sf));

    rand_num_matrix = reshape(Input_sample_Bi, SF(sf), ...
        input_len);

    % Binary to Gray Conversion
    Input_sample_gray = binary2gray(rand_num_matrix);

    % Binary to Decimal conversion
    Input_sample = bi2de(Input_sample_gray','left-msb')';

    lora_total_sym = preamble_len + sync_len + input_len;  % ...
        Total transmitted symbols


    %% Preamble Generation
    inverse = 0;
    for i = 1:preamble_len
        [out_preamble] = ...
            LoRa_Modulation(SF(sf),BW,Fs,num_samples,0,inverse);
```

```
39          outp((i-1)*num_samples+1 : i*num_samples) = ...
                out_preamble;
40      end
41
42      %% Sync Symble Generation
43      inverse = 1;
44      for i = 1:sync_len
45          [out_sync] = ...
                LoRa_Modulation(SF(sf),BW,Fs,num_samples,32,inverse);
46          outp = [outp out_sync];
47      end
48
49      %% Symble Generation
50      inverse = 0;
51      for i = 1:input_len
52          [out_sym] = ...
                LoRa_Modulation(SF(sf),BW,Fs,num_samples,Input_sample(i),inverse);
53          outp = [outp out_sym];
54      end
55
56
57      for ite = 1:1:Total_iterations
58          for snr = 1:1:length(SNR_dB)
59              %% AWGN Channel
60              out_channel = awgn(outp,SNR_dB(snr),'measured');
61
62              %% Reverse chirp generation for receiver
63              inverse = 1;
64              [out_reverse] = ...
                    LoRa_Modulation(SF(sf),BW,Fs,num_samples,0,inverse);
65              % Multiplying with the reverse chirp
66              for n = 1:1:lora_total_sym
67                  decoded_out((n-1)*num_samples + 1 : ...
                        n*num_samples) = ...
                        (out_channel((n-1)*num_samples + 1 : ...
                        n*num_samples).*out_reverse);
68              end
69
70              %% Calculating FFT
71              for m = 1:1:lora_total_sym
72                  FFT_out(m,:) = ...
                        abs((fft(decoded_out((m-1)*num_samples + ...
                        1 : m*num_samples))));
73              end
74
75              %% Decoding the received data
76              k=1;
77              for m = ...
                    (preamble_len+sync_len+1):1:(lora_total_sym)
78                  [r,c] = max(FFT_out(m,:));
79                  data_received_De(k) = c-1;
80                  k = k+1;
81              end
82
```

```
            % Decimal to Binary Conversion
            data_received_bin = ...
                de2bi(data_received_De,SF(sf),'left-msb')';
            % Gray to Binary Conversion
            data_received_gray = ...
                gray2binary(data_received_bin);
            % Matrix to array conversion
            data_received = ...
                reshape(data_received_gray,total_bits,1);

            %% BER Calculation
            BER(ite,snr) = sum(abs(data_received - ...
                Input_sample_Bi))/total_bits;
        end
    end
    Avg_BER(sf,:) = mean(BER);  % Average BER over all the ...
        iterations
    clear FFT_out;
    clear outp;
    clear decoded_out;
    clear data_received_De;
    clear data_received_gray;
end


%% Plotting
% Plotting the BER vs SNR curve
semilogy(SNR_dB,Avg_BER);
hold on
title('Probability of Bit Error vs SNR');
xlabel('SNR (Signal to Noise ratio in dB)');
ylabel('probability of Bit Error (Bit Error Rate)');
legend('SF 7','SF 8','SF 9','SF 10','SF 11','SF 12');
grid on;
```

A.0.5 The following algorithm performs the simulation of detecting the preamble for spreading factor 12.

```

%%
clear all; close all; clc;

%% main parameters of the simulator

SF = 12;                    % Spreading Factor
BW = 125000;                    % 125kHz
Fs = 500000;                    % Sampling Frequency
Down_sample = Fs/BW;
num_of_samples = 2^SF;
preamble_len = 8;               % Preamble length
sync_len = 2;                   % Sync length
total_bits = 27720;
```

```
15 SNR_dB = -20:3:-14;
16 SNR = 10.^(SNR_dB/10);       % SNR
17 Total_iterations = 10;
18 BER = zeros(Total_iterations,length(SNR_dB)); ...
       %iteration-by-length matrix
19
20 for sf = 1:1:length(SF)
21
22     num_samples = fix(Fs*(2^SF(sf))/BW);  % Number of samples
23
24     %% Random Number Generation
25     [Input_sample_Bi, input_len] = ...
           LoRa_random_number_generation(total_bits,SF(sf)); ...
           %generates random bits upto 27720
26
27     rand_num_matrix = reshape(Input_sample_Bi, SF(sf), ...
           input_len); %reshapes input_sample_Bi into a SF by ...
           input_len matrix.
28
29     % Binary to Gray Conversion
30     Input_sample_gray = binary2gray(rand_num_matrix);
31
32     % Binary to Decimal conversion
33     Input_sample = bi2de(Input_sample_gray','left-msb')';
34
35     lora_total_sym = preamble_len + sync_len + input_len;  % ...
           Total transmitted symbols
36
37
38     %% Preamble Generation
39     inverse = 0;
40     for i = 1:preamble_len
41         [out_preamble] = ...
               LoRa_Modulation(SF(sf),BW,Fs,num_samples,0,inverse); ...
               % 1*128 matrix
42         outp((i-1)*num_samples+1 : i*num_samples) = ...
               out_preamble;
43     end
44     %% Sync Symble Generation
45     inverse = 1;
46     for i = 1:sync_len
47         [out_sync] = ...
               LoRa_Modulation(SF(sf),BW,Fs,num_samples,32,inverse); ...
               %its a 1*128 matrix
48         outp = [outp out_sync]; %outp = [outp x];
49     end
50
51     %% Symble Generation
52     inverse = 0;
53     for i = 1:input_len
54         [out_sym] = ...
               LoRa_Modulation(SF(sf),BW,Fs,num_samples,Input_sample(i),inverse); ...
               %its a 1*128 matrix
55         outp = [outp out_sym]; %outp = [outp t];
```

```
56      end
57
58    %% for ite = 1:1:Total_iterations
59          for snr = 1:1:3
60
61              %% AWGN Channel
62
63              out_channel = awgn(outp,SNR_dB(snr),'measured');
64              out_channel = downsample(out_channel,Down_sample);
65              %%out_channel_1 = ...
                    downsample(out_channel,Down_sample);
66
67
68              y1 = wgn(1,5,SNR_dB(snr));
69              %y2 = var(y1);
70              noise = abs(y1);
71
72              %% Reverse chirp generation for receiver
73              inverse = 1;
74              [out_reverse] = ...
                    LoRa_Modulation(SF(sf),BW,Fs,num_samples,0,inverse);
75              out_reverse = downsample(out_reverse,Down_sample);
76              % Multiplying with the reverse chirp
77              for n = 1:1:lora_total_sym
78                  decoded_out((n-1)*num_of_samples + 1 : ...
                        n*num_of_samples) = ...
                        (out_channel((n-1)*num_of_samples + 1 : ...
                        n*num_of_samples).*out_reverse);
79              end
80
81
82
83              %% Calculating fourier transform
84              for m = 1:1:lora_total_sym
85                  FFT_out(m,:) = ...
                        max(abs((fft(decoded_out((m-1)*num_of_samples ...
                        + 1 : m*num_of_samples))
86              end
87              tt = FFT_out; %% for visualization
88
89              conversion_of_FFT_out = tt.'; %reshape(tt,1,3970);
90              noise_plus_signal = [noise conversion_of_FFT_out];
91              noise_array_length = length(noise);
92
93              y5 = noise_plus_signal(1:100);
94
95
96              y6 = 0:1:(length(y5) - 1);
97           xlabel('Received Symbols')
98           ylabel('Signal Power (dBW)')
99           title('Plot of signal received ')
100          plot(y6,y5);
101          hold on
102
```

```
        end
        hold off
        legend('SNR = -20','SNR = -17','SNR = -14')
        %%figure
        %%findchangepts(y5,'MaxNumChanges',5)
        %%figure
        %%findchangepts(y5,'MaxNumChanges',5,'Statistic','rms')
        %%figure
        %%findchangepts(y5,'Statistic','std')
        figure
        findchangepts(y5,'Statistic','mean','MinThreshold',1)

        %%figure
        %%findchangepts(y5,'Statistic','linear','MinThreshold',0.6)
    %% end
    iterations
    clear FFT_out;
    clear outp;
    clear decoded_out;
    clear data_received_De;
    clear data_received_gray;
end
```

A.0.6 The following algorithm simulates the probability of detection against Signal to Noise Ratio for spreading factor 7

The algorithm can be adjusted for all spreading factors by changing the SF values.

```
%%
clear all; close all; clc;
%% main parameters of the simulator


SF = 7;                 % Spreading Factor
BW = 125000;                % 125kHz
Fs = 125000;                % Sampling Frequency
preamble_len = 8;           % Preamble length
sync_len = 2;               % Sync length
total_bits = 27720;         % total bits to be transmitted ...
    in LoRa message
SNR_dB = -30:1:00;          % SNR in DB
SNR = 10.^(SNR_dB/10);      % SNR
Total_iterations = 300;
BER = zeros(Total_iterations,length(SNR_dB));

for sf = 1:1:length(SF)

    % Number of samples
    num_samples = Fs*(2^SF(sf))/BW;

```

```
22      %% Random Number Generation
23      [Input_sample_Bi, input_len] = ...
            LoRa_random_number_generation(total_bits,SF(sf));
24
25      rand_num_matrix = reshape(Input_sample_Bi, SF(sf), ...
            input_len);
26
27      % Binary to Gray Conversion
28      Input_sample_gray = binary2gray(rand_num_matrix);
29
30      % Binary to Decimal conversion
31      Input_sample = bi2de(Input_sample_gray','left-msb')';
32
33      % Total transmitted symbols
34      lora_total_sym = preamble_len + sync_len + input_len;
35
36
37      %% Preamble Generation
38      inverse = 0;
39      for i = 1:preamble_len
40          [out_preamble] = ...
                LoRa_Modulation(SF(sf),BW,Fs,num_samples,0,inverse);
41          outp((i-1)*num_samples+1 : i*num_samples) = ...
                out_preamble;
42      end
43
44      %% Sync Symble Generation
45      inverse = 1;
46      for i = 1:sync_len
47          [out_sync] = ...
                LoRa_Modulation(SF(sf),BW,Fs,num_samples,32,inverse);
48          outp = [outp out_sync];
49      end
50
51      %% Symble Generation
52      inverse = 0;
53      for i = 1:input_len
54          [out_sym] = ...
                LoRa_Modulation(SF(sf),BW,Fs,num_samples,Input_sample(i),inverse);
55          outp = [outp out_sym];
56      end
57      Disp_outp = outp;
58
59
60      for ite = 1:1:Total_iterations
61          for snr = 1:1:length(SNR_dB)
62              %% AWGN Channel
63              out_channel = awgn(outp,SNR_dB(snr),'measured');
64
65              %% Reverse chirp generation for receiver
66              inverse = 1;
67              [out_reverse] = ...
                    LoRa_Modulation(SF(sf),BW,Fs,num_samples,0,inverse);
68              % Multiplying with the reverse chirp
```

```
69              for n = 1:1:lora_total_sym
70                  decoded_out((n-1)*num_samples + 1 : ...
                        n*num_samples) = ...
                        (out_channel((n-1)*num_samples + 1 : ...
                        n*num_samples).*out_reverse);
71              end
72              disp_decoded_out = decoded_out;
73              %% Calculating FFT
74              for m = 1:1:lora_total_sym
75                  FFT_out(m,:) = ...
                        abs((fft(decoded_out((m-1)*num_samples + ...
                        1 : m*num_samples))));
76              end
77              Disp_FFT_out = FFT_out;
78              %% Decoding the received data
79              k=1;
80              for m = 1:1:(lora_total_sym)
81                  [r,c] = max(FFT_out(m,:));
82                  data_received_De(k) = c-1;
83                  k = k+1;
84              end
85              Disp_data_received_De = data_received_De;
86              Disp_data_received_De1 = ...
                    Disp_data_received_De(1:1:8);
87
88
89              PD(ite,snr) = ...
                    sum(abs(Disp_data_received_De1)<10)/length(Disp_data_received_De1);
90              %end
91
92
93          end
94      end
95  end
96  Avg_BER(sf,:) = mean(PD);
97  semilogy(SNR_dB,Avg_BER);
98  hold on
99  title('Probability of Detection Curve');
100 xlabel('SNR (Signal to Noise Ratio in dB)');
101 ylabel('Probability of detection');
102 legend('SF 7');
103 grid on
```