POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Master's Degree Thesis

Experimental implementation of a LoRa sensor network and robustness analysis of simulated TX-RX LoRa signals using LabVIEW

Supervisors

Prof. Daniele TRINCHERO

Prof. Vahid MEGHDADI

Candidate

Elena FILIPESCU

1 April 2020

Summary

The Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment. Low-Power Wide-Area Networks (LPWAN) represent a set of long-range communication technologies suitable for supporting IoT applications, which need multi-year battery lifetime and usually send small amount of data over long distances a few times per hour.

An example of LPWAN is LoRa (Long Range), which refers to the physical layer of LoRaWAN protocol standardized by LoRa Alliance, an open and non-profit association. LoRa is the first low cost implementation of Chirp Spread Spectrum (CSS) modulation for commercial usage suitable for meeting IoT requirements, such as long range of communication, low power consumption, low data rate, low cost, due to its simplified star network topology and the use of Industrial, Scientific and Medical (ISM) radio bands, and high level of robustness to interference.

In this scenario, a LoRa wireless sensor network in a real IoT application is interesting to be implemented. However, the parameters setting of LoRa transceivers sometimes can be critical, since many combinations are possible and testing all of them in a real case might be time consuming. To cope with this, a useful tool is represented by a simulated LoRa system aimed at providing a performance analysis of the network. In particular, a preliminary study of the robustness against different types of interference is essential to reduce the probability of errors due to noise sources.

The main objectives of this thesis are two: the former concerns an experimental implementation of a LoRa sensor network installed by some students of Electronic Engineering inside the building of ENSIL (École National Supérieure d'Ingénieurs de Limoges), at the University of Limoges in France; the latter consists in the robustness evaluation of a transmitter-receiver LoRa signal, simulated with the help of LabVIEW, by analysing the bit error rate under different noise environments.

For what concerns the first part, 30 sensors for monitoring temperature and humidity are all wirelessly connected to a gateway using LoRa technology and microcontrollers programmed in C++. The purpose of the infrastructure is to convey temperature and humidity information from sensors to a gateway, based on

a Raspberry Pi board, which used to upload the data to the IoT platform service ThingSpeak. However, the overall power consumption is too high, making the network last about 3 months. Thus, the goal is to increase the battery lifetime at the sensor level. Therefore, after a preliminary analysis of LoRa operating modes, some modifications at the power management level have been applied to both the sensor and gateway side. For what concerns the packet structure, some additional fields have been added to allow both sides to filter the incoming packets which belong to this network. On the other hand, all the bytes which were not strictly necessary have been removed from the node packet. A new power management technique replaced the original one, which was not optimal. The adopted choices now allow the gateway to send back a command to properly increase or decrease the transmit output power of the sensing device, if necessary, according to the Received Signal Strength Indicator (RSSI) of the LoRa node signal. Also, a new packet field has been added to the sensing node to check the value of the transmit output power. Finally, the range of possible power value has been extended to achieve lower power levels. This way, a contribution in terms of adaptive power management has been provided to the wireless sensor network. In addition to this, the computional complexity such as the bit conversion of RSSI has been moved to the gateway side when possible, since not affected by power limitations. These operations helped to lighten the burden at the sensor level and to reduce the related power needs. Furthermore, the LoRa operating mode of the sensing device after the transmission of the measured data has been changed from continuous to single receive mode, allowing thus the receiver to go to sleep mode right after the end of a given receive window, in order to reduce the consumption. Finally, some bugs were corrected as well as other methods that used to cause a useless waste of power. For example, some parameters were originally set to values which did not grant a successful packet reception or the function in charge of the error reception used to make the program stop, instead of remaining in receive mode. Moreover, the program of the end-node microcontroller has been adapted in order to retransmit the same packet in case of timeout interrupt, instead of performing a new measurement in the same transmit period. Overall, all these new or updated operations improved the performance of the microcontroller program, especially for what concerns the early stops or malfunctionings of the original version of the codes.

At the same time, ThingSpeak has been replaced by a new server, generated with PHP language and hosting a MySQL database, along with a web user interface. In particular, the data are sent via HTTP GET request from the Raspberry Pi to the server, which uploads the records to the database. The HTML webpage has been created to allow users to access the database records of interest, by means of a form to select some specific entries and to download a .csv file containing the visualized data. Eventually, some records can be deleted if the proper password is

inserted. This eased the data retrieving required by the data analysis which will be performed by students and researchers.

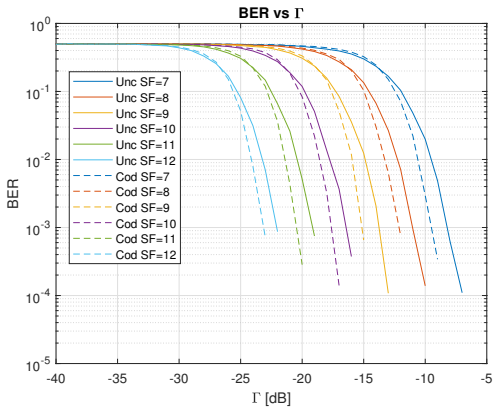
Some parts of this LoRa sensor network can be further developed and improved. This is the case of the gateway, which could be replaced by a new one able to work with more than just one channel and a unique spreading factor. Indeed, the exploitation of different channels and the adaptive control of spreading factors of LoRa nodes could enhance the orthogonality of LoRa signals and reduce the errors due to packet collisions, which could be significant in case of more sensing devices. Moreover, the web user interface is still basic and could be improved by means of a testing phase, which might be useful to get several feedbacks on the userfriendliness of the HTML application.

The second part, instead, aims at realizing a simulated TX-RX LoRa signal, by means of the graphical programming language provided by LabVIEW. The final purpose is to investigate the LoRa robustness under different noise conditions, without the need of real hardware transceivers. To accomplish this task, a deep analysis of the literature was essential to reach a sufficient level of knowledge of the proprietary LoRa protocol. In the state-of-the-art, several studies tried to reveal more details about LoRa PHYsical (PHY) layer, intercepting over-the-air LoRa signals by means of a Software Defined Radio (SDR) enabled LoRa gateway. Their results and conclusions have been exploited to implement the TX-RX LoRa signal, with some simplifying assumptions about the frame synchronization and the channel coding. The implementation of the LoRa modulation, demodulation and symbol baseband generation constitutes a potential contribution to the available LabVIEW codes that might be used by whoever is interested in LoRa.

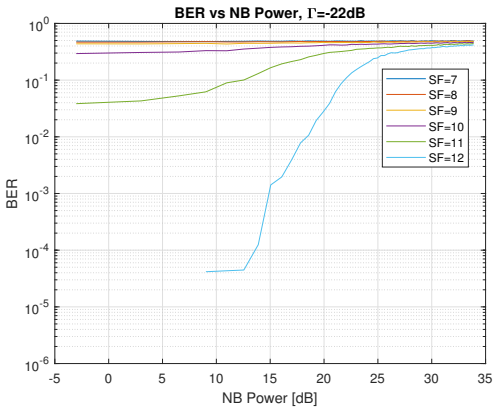
Among the related works considered in the literature, some studies tried to demonstrate the advantages of using this type of modulation and analysed the impact of an Additive White Gaussian Noise (AWGN) on the error rate, whose value is used to evaluate the robustness performance. Based on this, the Gaussian noise was added to the LoRa signal between the transmitter and the receiver simulated in LabVIEW. The simulation results in the case of uncoded signals, i.e. without considering the channel coding, have been validated with the work of Ferré and Giremus (2018). The only significant difference between the simulated and thereotical results lies in the minimum error value that can be detected, which is higher in the first case due to the limited number of packets that could be held by the simulation. After that, the advantage of introducing the channel coding technique has been shown. Indeed, considering a BER of around 10^{-3}, an improvement of $-1dB$ occurs in the value of the Signal-to-Noise Ratio (SNR) due to the AWGN, as shown in Figure 1a which represents the resulting BER curves obtained before and after applying the adopted channel coding.

Finally, the addition of a narrowband signal allowed to provide a reliable and innovative study of the effects of a narrowband interference on the Bit Error Rate

(BER) of a LoRa signal, in addition to the condition of AWGN channel. The study takes into account the BER for different combinations of spreading factor, SNR of the AWGN and carrier frequency and power of the narrowband signal. The most important result demonstrates that LoRa modulation is greatly robust to a narrowband interference, even if this is characterized by a power which is about 10 times higher than the LoRa signal. Indeed, the error rate is worsened in a significant way only for really powerful narrowband noise sources, i.e. characterized by a power of about $+15dB$ for the coded case, as shown in Figure 1b. Reasonably, the analysis for different values of SNR shows that the amplitude of the narrowband signal has a negligible impact on the BER when the error is already high due to the Gaussian Noise contribution.



(a) *Coded and uncoded BER vs SNR*



(b) *Coded BER vs NB Power,* $\Gamma = -22dB$*,* $f = 1kHz$

Table of Contents

List of Tables

List of Figures

Acronyms

LPWAN
Low-Power Wide-Area Networks

UNB
Ultra Narrow Band

SS
Spread Spectrum

IoT
Internet of Things

CSS
Chirp Spread Spectrum

CHIRP
Compressed High Intensity Radar Pulse

SF
Spreading Factor

PHY
PHYsical

ISM
Industrial, Scientific and Medical

SNR
Signal-to-Noise Ratio

CRC

Cyclic Redundancy Check

SFD

Start Frame Delimiter

FEC

Forward Error Correction

UART

Universal Asynchronous Receiver-Transmitter

SDR

Software Defined Radio

I2C

Inter-Integrated Circuit

IDE

Integrated Development Environment

RSSI

Received Signal Strength Indicator

AWGN

Additive White Gaussian Noise

SPI

Serial Peripheral Interface

RF

Radio Frequency

PLL

Phase-Locked Loop

RX

Receive

TX

Transmit

HSI

High Speed Internal

GPIO

General Purpose Input/Output

MCU

MicroController Unit

ID

Identification Number

LAMP

Linux, Apache, MySQL, PHP/Perl/Python

IP

Internet Protocol

LAN

Local Area Network

BER

Bit Error Rate

SER

Symbol Error Rate

FER

Frame Error Rate

LabVIEW

Laboratory Virtual Instrument Engineering Workbench

VI

Virtual Instrument

NB

Narrow Band

CFO

Carrier Frequency Offset

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment [1]. IoT applications can improve their decision making by exploiting the information collected by each connected device. According to Cisco, 500 billion devices are expected to be connected to the Internet by 2030 [2], implying a huge amount of data exchange.

In this scenario, Low-Power Wide-Area Networks (LPWAN) represent a set of long-range communication technologies suitable for supporting IoT applications, which need multi-year battery lifetime and usually send small amount of data over long distances a few times per hour. Indeed, the main technology attributes of LPWAN are [3]:

- Long range of communication
- Low power consumption
- Low data rate (the price for the previous two features)
- Low cost of device and deployment
- Simplified network topology and deployment (thanks to the adoption of a star topology, which allows to remove repeaters)
- Full penetration coverage
- Network scalability and capacity upgrade

In addition to the classification between licensed and unlicensed, two categories of LPWAN technologies can be identified according to the physical layer: Ultra Narrow Band (UNB), e.g. SigFox, and Spread Spectrum (SS), e.g. LoRa (LongRange).

For its low power capability, long communication range and high level of robustness to interference, Chirp Spread Spectrum (CSS) was traditionally used for military and space purposes. LoRa is the first low cost implementation of CSS modulation for commercial usage [4].

In a nutshell, LoRa is the physical layer protocol, while LoRaWAN refers to the IP network and transport layers on top of LoRa. The LoRaWAN protocol is standardized by LoRa Alliance, an open, non-profit association of members [5].

In this scenario, a practical implementation of a LoRa wireless sensor network in a real IoT application is interesting to be pursued. However, the parameters setting of LoRa transceivers sometimes can be critical, since many combinations are possible and testing all of them in a real case might be time consuming. To cope with this, a useful tool might be a simulated LoRa system aimed at providing a performance analysis of the network. In particular, a preliminary study of the robustness against different types of interference is essential to reduce the probability of errors due to noise sources.

1.2 Thesis objectives

The goals of this thesis are two: the former concerns a practical LoRa network implementation started by some students of Electronic Engineering at the University of Limoges, in France, in collaboration with the Department of Water and Environment Engineering; the latter consists in the design, implementation and validation of a simulated transmitter-receiver LoRa signal, by means of LabVIEW.

Experimental implementation of a LoRa sensor network Inside the building of ENSIL (École National Supérieure d'Ingénieurs de Limoges), 30 sensors monitoring temperature and humidity are all wirelessly connected to a gateway using LoRa technology and the data are uploaded to the IoT platform service ThingSpeak. However, the overall power consumption is relatively high for the purpose of such an IoT application. Thus, the objective is to increase the lifetime of the network. Moreover, another request is to replace ThingSpeak with our own server.

Robustness analysis of a simulated TX-RX LoRa signal This second objective aims at simulating with LabVIEW a transmitter-receiver LoRa signal, in order to analyse its robustness to interference, evaluating the behaviour of bit error rate under different noise environments.

1.3 Thesis outline

Here the structure of the thesis is presented.

Chapter 2 will provide an introductory technical overview of the basic principles of LoRa and LoRaWAN, with a particular attention to the physical layer, essential to pursue the second objective. Then, following the division of the objectives, the thesis is split into two parts as follows:

- Part I, concerning the sensors network installed at ENSIL. Chapter 3 will provide an overview of the state-of-the-art of the developed sensor network, presenting the issues, the architecture and the tools. Right after, Chapter 4 describes the new power management strategies which have been adopted. After that, the design and implementation of the new server and web user interface are shown in Chapter 5, ending with the final Chapter 6 of conclusions.
- Part II, for the LabVIEW simulation and robustness analysis. The state-of-the-art is analysed in Chapter 7, focusing on the available documents which provide relevant technical specifications on the physical layer, whose protocol is proprietary, and on the closed-form approximations for the bit error rate in the case of LoRa modulation. Then, the architecture design and the LabVIEW implementation are described in Chapter 8. After that, Chapter 9 collects the design, implementation and results of the robustness analysis of LoRa to interference. Chapter 10 is the last chapter with the conclusions of this second part.

Chapter 2

LoRaWAN and LoRa

As mentioned in Chapter 1, the LoRaWAN specification is a LPWA networking protocol designed to wirelessly connect battery operated 'things' to the Internet in regional, national or global networks, and it is particularly suitable for meeting IoT requirements such as bi-directional communication, end-to-end security, mobility and localization services [6].The underlying physical layer is LoRa.

2.1 LoRa PHY

LoRa is a proprietary spread spectrum modulation technique that is derivative of Chirp Spread Spectrum (CSS) [7]. It uses the entire channel bandwidth to broadcast a signal, providing a great robustness to channel noise. Moreover, the use of a broad band of the spectrum makes the modulation resistant to long term relative frequency error, multi-path, fading and Doppler effects [8]. The counterpart is a low data rate, with a range between 250 bps and 50 kbps.

The low cost of LoRa is also maintained by means of the use of Industrial, Scientific and Medical (ISM) radio bands which are license-free and, in particular, 169 MHz, 433 MHz, 868 MHz (Europe), 915 MHz (USA) and 923 MHz (Asia).

2.1.1 Chirp Spread Spectrum

In general, spread-spectrum techniques allow to compensate for the degradation of the Signal-to-Noise Ratio (SNR) of a radio channel, by exploiting the increase of the signal bandwidth. Chirp Spread Spectrum, based on CHIRP signal, which stands for Compressed High Intensity Radar Pulse, was developed for radar applications in the 1940s [7]. To encode the information, wideband linear frequency modulated chirp pulses are used with a frequency that increases (up-chirp) or decreases (down-chirp) over a certain amount of time and with a constant amplitude.

In the case of LoRa, the signal is transmitted using different values for the Spreading Factor (SF), which represents the length of each sub-sequence of information flow generated from the MAC layer or, equivalently, the number of bits per symbol, with $SF \in [7...12]$. The number of possible symbols S_m is thus equal to $M = 2^{SF}$. Therefore, each symbol is associated to a chirp, whose duration, determined by the value of SF, depends on the communication requirements. Intuitively, the higher the SF, the higher the robustness to interference, the lower the reachable data rate.

The chirp transmitted at time mT_s and defined as $f_c^m(t)$, where $T_s = \frac{M}{B}$ is the symbol duration and B (125kHz, 250kHz or 500kHz) is the CSS signal bandwidth, can be obtained using a cyclic shift of $\tau_m = \frac{S_m}{B}$, as shown in Figure 2.1.

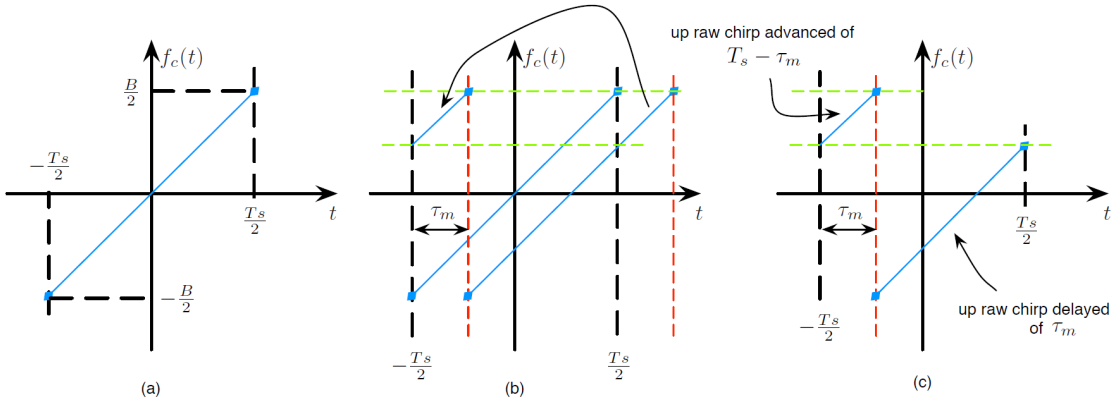


Figure 2.1: Symbol-chirp association process - (a) up raw chirp - (b) process illustration - (c) chirp associated to the m^{th} symbol [9] ©2018 IEEE

According to [10], considering the relation:

$$f_c^m(t) = \frac{1}{2\pi}\frac{d\phi_c^m(t)}{dt} \tag{2.1}$$

The instantaneous phase can be expressed as (2.2) for $t \in \left[-\frac{T_s}{2}, -\frac{T_s}{2} + \tau_m\right]$ and (2.3) for $t \in \left[-\frac{T_s}{2} + \tau_m, \frac{T_s}{2}\right]$:

$$\phi_c^m(t) = 2\pi\left[\frac{B}{2T_s}t^2 - \frac{S_m}{T_s}t\right] \tag{2.2}$$

$$\phi_c^m(t) = 2\pi\left[\frac{B}{2T_s}t^2 - \left(\frac{S_m}{T_s} - B\right)t\right] \tag{2.3}$$

Then, the complex envelope of the signal modulated with CSS can be defined as follows:

$$s(t) = \sum_{k \in \mathbb{Z}} e^{j\phi_c^k(t-kT_s)} \tag{2.4}$$

2.1.2 LoRa PHY Packet Format

From official documents provided by Semtech [11], the packet is structured as follows:

- Preamble
- Header (Header and Cyclic Redundancy Check (CRC) in explicit mode only)
- Payload
- Payload CRC

Several studies tried to reveal more details about LoRa PHY layer, intercepting over-the-air LoRa signals by means of a Software Defined Radio (SDR) enabled LoRa gateway [12], [13].

The preamble consists in a training sequence of repeated upchirps, followed by a mandatory preamble of 4.25 symbols, of which 2 synchronization word symbols and 2.25 downchirps as Start Frame Delimiter (SFD).

According to the presence or absence of CRC, uplink and downlink packets can be distinguished. Moreover, if implicit header is disabled, the header is encoded in $\frac{4}{4+CR}$, where CR is the code rate, i.e. the number of parity bits (from 1 to 4).

Then, depending on the SF, a variable number of data chirps is used to send the payload.

Finally, CRC is added at the end of uplink packets.

2.1.3 LoRa PHY Block Diagram

According to Semtech European Patent Application [14], data are encoded before being transmitted. In particular, four operations are applied to the input bits [15]:

- Hamming(M,4) Forward Error Correction (FEC), which adds a certain number of parity bits (N) according to the coding rate.
- Whitening, that randomizes data, helping the receiver synchronization.
- Interleaving, which distributes bits within the packet. The interleaver is diagonal.

- Symbol "Gray indexing" (i.e. Gray de-coding) to increase error tolerance. Indeed, in Gray code two successive values differ in only one bit. It is up to the receiver the Gray encoding procedure.

The block diagram is represented in Figure 2.2 [16].

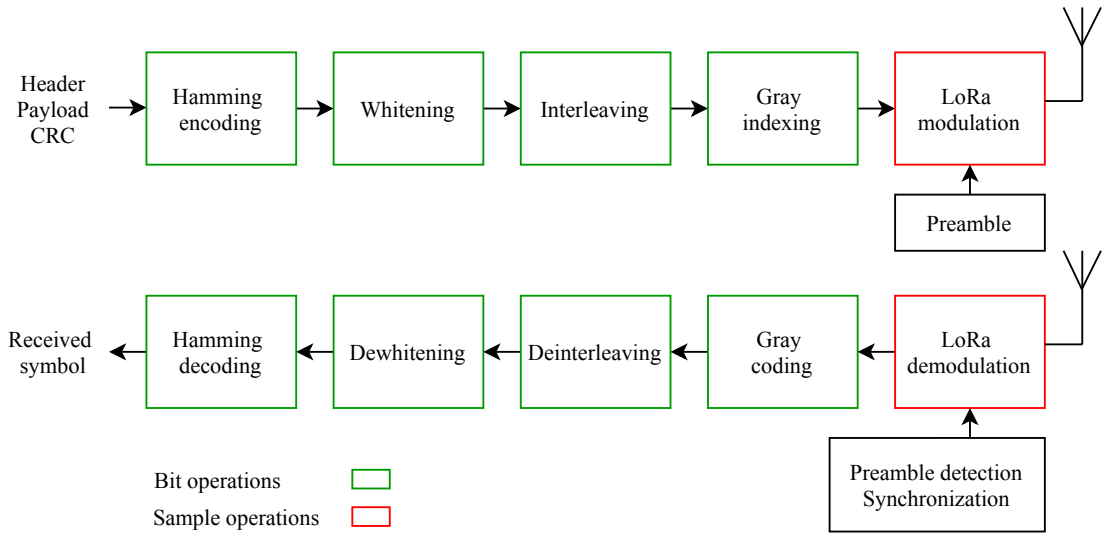


Figure 2.2: LoRa PHY Block Diagram

2.2 LoRaWAN Network Architecture

LoRaWAN network architecture is shown in Figure 2.3.

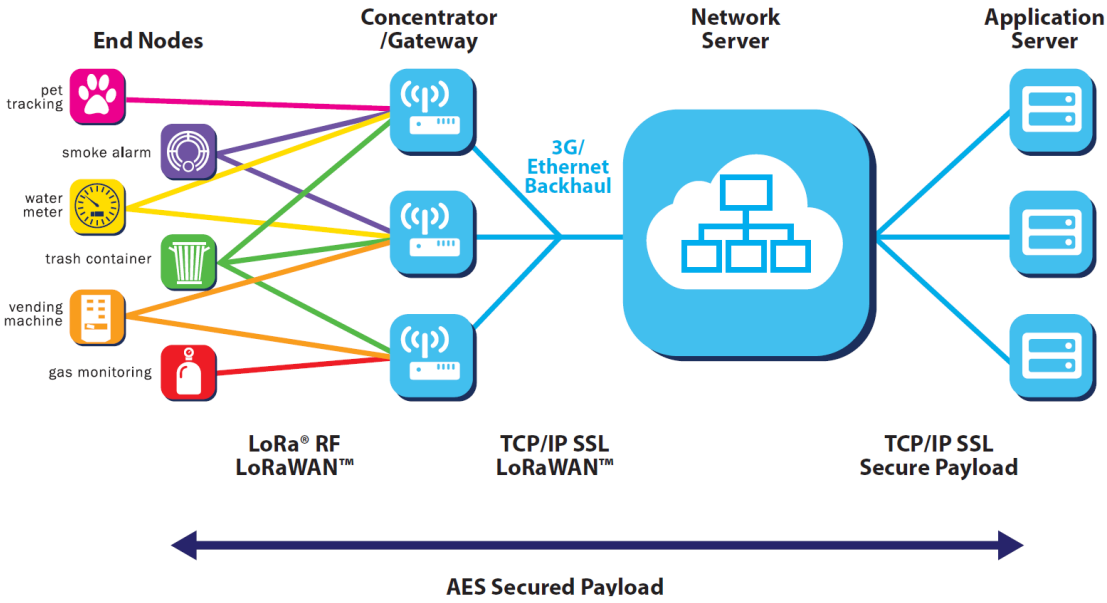


Figure 2.3: LoRaWAN Network Architecture [4]

The long range star architecture is adopted to reduce the power consumption. On the left, the end nodes transmit data packets that are typically received by several gateways that will forward them to the network server via backhaul, e.g. 3G or Ethernet. The network server manages complex operations, such as filtering redundant packets, security or adaptive data rate.

The communication protocol adopted by LoRaWAN network layer is Aloha type, where the nodes transmit messages in an asynchronous way, i.e. triggered by events or scheduled.

Since the gateway must be able to receive packets at the same time by a huge amount of end-nodes, the network capacity is increased by exploiting adaptive data rate and a multichannel multi-modem transceiver in the gateway.

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

The devices can be classified as follows:

- Class A, for bi-directional communications where each uplink transmission is followed by two short downlink receive windows. It grants the lowest power consumption to the applications that only require downlink right after the transmission.

- Class B, for battery powered actuators. It adds scheduled receive slots to Class A.

- Class C, for main powered actuators that can listen continuously with maximal receive slots.

Part I

Experimental implementation of a LoRa sensor network

Chapter 3

State-of-the-art

3.1 Overview

The development of the sensor network inside the building of ENSIL was managed by Peng Xu and Rufei Yang, students of Electronic and Telecommunication Engineering, under the supervision of professors Vahid Meghdadi and Abdul Karim Yazbek [17]. The purpose of the infrastructure is to convey temperature and humidity information from sensors to a central station.

The data analysis is performed by the research institute XLIM and the Water and Environment Engineering department, in order to identify proper solutions for energy waste minimization and study the environment aspects, relatively.

To further increase the battery lifetime, the energy consumption must be reduced at the sensor level.

3.2 Architecture

The packets are wirelessly sent using LoRa protocol to the receiver that forwards the data to the gateway, via Universal Asynchronous Receiver-Transmitter (UART) serial port. The gateway, based on a Raspberry Pi board, uploads the data to ThingSpeak [18] cloud, to be further visualized and downloaded eventually by the user via Internet. Figure 3.1 shows the project architecture.

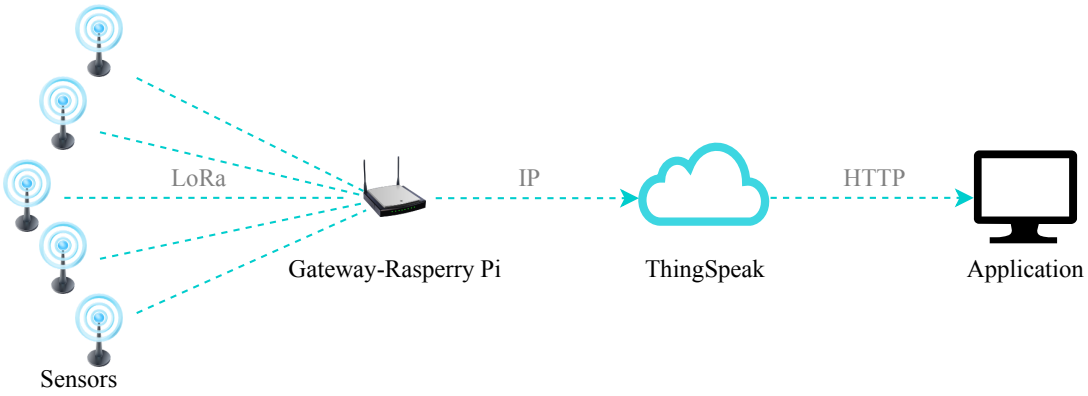


Figure 3.1: Project architecture

An example of sensing device is provided in Figure 3.2.

Figure 3.2: Example of sensing node

3.3 Tools

In this section, both hardware and software tools are presented.

3.3.1 Hardware

- AM2320 sensor: Figure 3.3 shows the temperature and humidity combined sensor AM2320 [19], which consists of a capacitive moisture element and an integrated high-precision temperature measurement device. It employs a standard Inter-Integrated Circuit (I2C) and single-bus output.

Figure 3.3: Temperature/humidity sensor AM2320

- NZ32-SC151 board (Figure 3.4), assembled with a STM32L151RC Microcontroller [20], is designed for low power applications. It contains an on board battery charger for a 3.7V Li-Ion or Li-Polymer battery and it has 3 iMod ports to add Modtronix iMod modules, such as LoRa SX1276/SX1278 Wireless iMod module. The board can be powered via USB connector, external battery or 5V pin headers. The firmware can be upgraded via USB port, by entering bootloader mode.

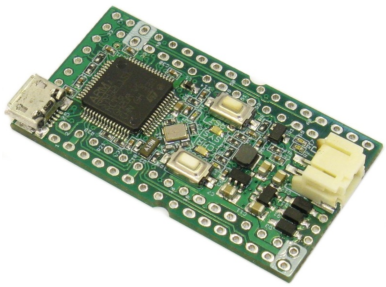


Figure 3.4: NZ32-SC151, with STM32L151RC

- Wireless SX1276 LoRa Module: Figure 3.5 shows the inAir9B board, a 868/915 MHz wireless module using the Semtech SX1276 chip [21], which is pin compatible with STM23L151RC. The SX1276/77/78/79 transceivers provide long range capability and high interference immunity whilst minimising current consumption, thanks to the use of Semtech's patented LoRa modulation technique. A sensitivity of over -148dBm can be reached and the maximum output power is +20dBm.

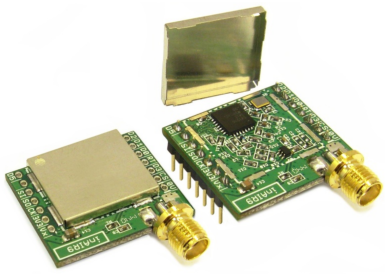


Figure 3.5: Wireless SX1276 LoRa module

- Raspberry Pi 3 Model B: Figure 3.6 shows the Raspberry Pi model [22], based on a Quad Core 1.2GHz Broadcom BCM2837 CPU .

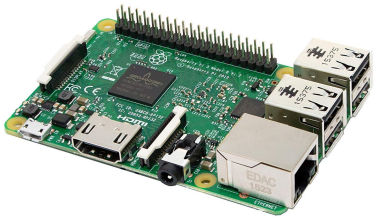


Figure 3.6: Raspberry Pi 3 Model B

- Antenna 868MHz 1/4wave
- Li-Polymer battery 3.7V 2000mAh

3.3.2 Software

- System Workbench for STM32 [23], the free Integrated Development Environment (IDE) for STM32 microprocessors developed by Ac6 Tools [24].
- Dfu file manager, to generate the .dfu file from a .hex file
- DfuSeDemo, which can be installed by downloading STSW-STM32080 in "Required software" from [25]. It is used to send the .dfu file in the microcontroller memory. The values for Vendor ID, Product ID and Version must be the same as those used by Dfu file manager;
- hercules_3-2-8.exe, for the serial interface to send and receive data through the proper USB COM port of a PC.

3.3.3 Libraries

For the transmission and reception of packets, a GitHub library [26] is required, whose folder SW4STM32 needs to be set for the workspace, while running System Workbench. It allows to exploit SX1276Lib.

3.3.4 Project building and Firmware upgrading

First of all, in order to create and run the bootloader, the IDE must generate, in addition to the binary file, the .hex file which can be obtained by:

```
Project » Properties » C/C++ Build » Setting » Build Steps »
 Post-build steps » Command:  arm-none-eabi-objcopy -O ihex
             "${BuildArtifactFileBaseName}.elf"
"${BuildArtifactFileBaseName}.hex" && arm-none-eabi-size -B
                 "${BuildArtifactFileName}"
```

Then, the project must be rebuilt and the .hex file can be found in the Debug folder. Now, Dfu file manager can be launched to generate the .dfu file.

After that, the firmware can be upgraded via USB port by means of DfuSeDemo application. In order to do that, the bootloader mode on the module has to be accessed, so once it is connected to the PC, one must press and hold BOOT button, toggle RESET button and, finally, release BOOT button.

At this point, the device is visible in the available DFU devices field of the application.

Then, the current firmware can be uploaded on the device as follows:

```
Choose » select *.dfu file » Upgrade
```

Finally, with RESET button, the program is started.

3.4 Algorithms

This section provides a quick overview of the solution used to reduce the energy consumption. Figure 3.7 shows the algorithm implemented for the LoRa network, while Figure 3.8 describes the way the transmitter output power is adjusted.

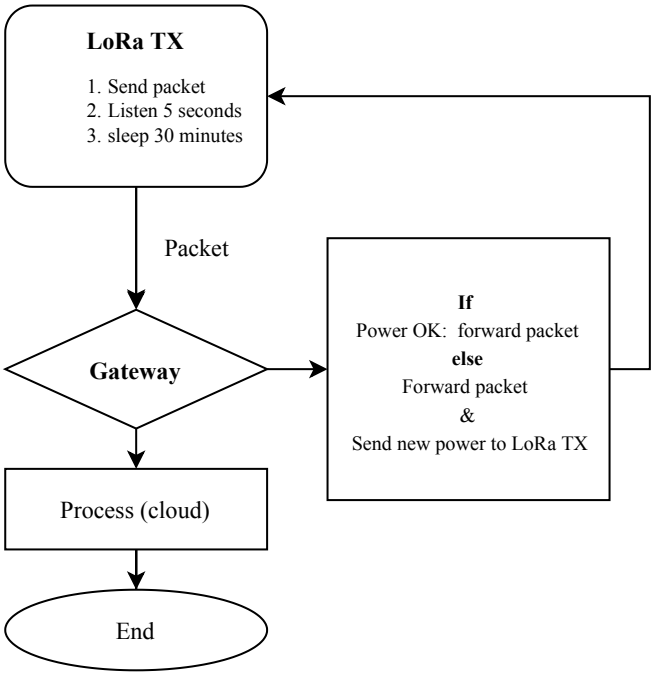


Figure 3.7: LoRa network implementation algorithm

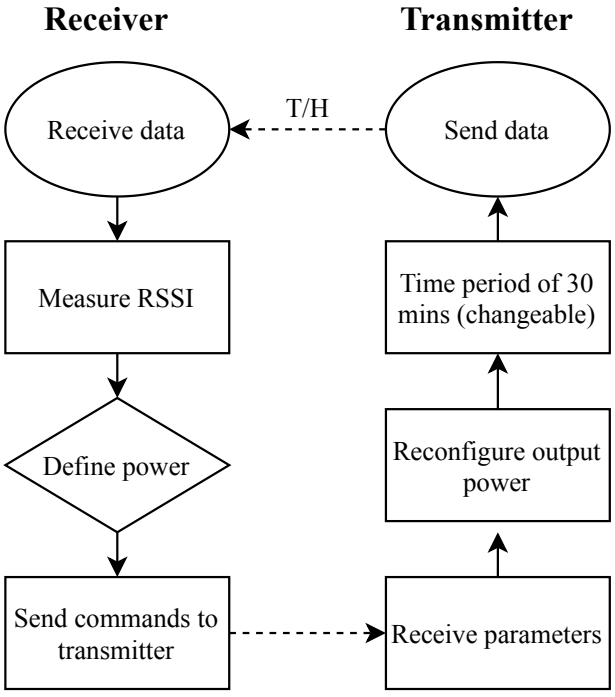


Figure 3.8: Transmitter output power adjustment

In particular, the calculation is performed according to the Received Signal Strength Indicator (RSSI) of the receiver. Indeed, the output power, between $0dBm$ and $+14dBm$, is associated to eight different RSSI intervals, that allow to estimate the distance between the node and the gateway.

Moreover, to improve the battery lifetime, the device is put in sleep mode considering each component, i.e. the microprocessor, the radio module and the sensor, as shown in Figure 3.9.

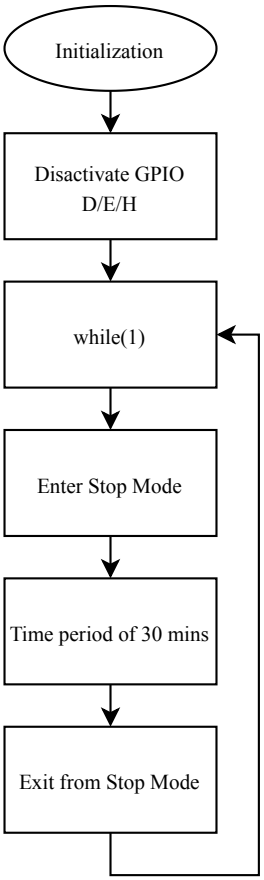


Figure 3.9: Sleep mode setting

Applying all these methods, the network can last around 3 months, which is not optimal.

Chapter 4

Power Management

For this part, the IDE has been used with C++ as programming language.

4.1 LoRa operating modes

From the Semtech datasheet of SX1276 [21], different operating modes are available, among which:

- Sleep: low-power mode in which only Serial Peripheral Interface (SPI) and configuration registers are accessible. No blocks are enabled. The supply current has a typical value of $0.2\mu A$ and a maximum of $1\mu A$;
- Standby: mode in which crystal oscillator and top regulator are turned on, while Radio Frequency (RF) part and Phase-Locked Loop (PLL)s are disabled. Typical and maximum values for the supply current in Standby mode are $1.6mA$ and $1.8mA$, respectively.
- Transmit (TX): frequency synthesizer and transmitter blocks are enabled. When the packet transmission ends, the module goes to Standby mode.
- Receive (RX) continuous: all the blocks required for reception are enabled until a new user request is made to change operating mode.
- RX single: all the blocks required for reception are enabled until a valid packet has been received and then the SX1276 returns to Standby mode.

Practically, after each transmission, so at each OnTxDone routine, the radio has to be manually set to Standby or Sleep mode, despite the datasheet specification.

Moreover, registers are readable in all the modes, but Sleep or Standby modes must be called in order to write into the configuration registers.

Single RX mode and Continuous RX mode

In battery operated systems or in systems with a limited number of timers for the companion microcontroller, the use of the timeout present in LoRa single reception mode allows to reduce the amount of time spent in reception, while not using any of the companion MicroController Unit (MCU) timers. The adoption of this mode is suitable for the power saving purposes.

At the end of the reception, the RxTimeout interrupt is triggered and used to wake up the companion MCU. Only if the data reception is corrupted by external perturbations, the interrupt is generated before the end of the reception and the device goes to Standby mode.

On the other hand, the LoRa continuous reception mode is adopted in systems with no power limitations or where the use of a companion MCU timer is preferably chosen over the radio embedded timeout system. In this configuration, the radio is continuously ready for the reception of LoRa packets, until the companion MCU timer sets it into another operating mode.

Practically, the single reception mode should only be used when the arrival time window of the packet is known, in order to allow a temporal window for receiving a possible incoming packet, after which, if nothing arrives, the timeout interrupt is raised and the radio goes to Standby. Otherwise, the RxDone routine is generated before going to Standby.

4.2 Power consumption test

A simple program has been developed to test the LoRa network and, in particular, to study the different behaviors of the main radio modes.

Initially, two devices, i.e. two couples of inAIR9B and NZ32-SC151 boards, have been programmed as follows:

- A transmitter, transmitting a packet every 20 seconds. At each OnTxDone interrupt, the module is put in single reception mode for about 10 seconds, after which, RxDone routine or the receiver timeout are called depending on the reception of a valid packet, or not.

- A receiver, in continuous mode. At each successful reception, it answers back to the transmitter. Then, it is put in transmit mode and it goes back to continuous receive mode until the next incoming packet.

It is important to properly set the value of `LORA_SYMBOL_TIMEOUT`, because if it is too short the receive timeout interrupt will be raised before the actual timeout value. In this case, its value has been found by inspection, by checking on Hercules application that the packet sent by the transmitter was effectively received.

Subsequently, to properly measure each contribution to the current consumption, the microcontroller is put in sleep mode. Moreover, after the packet reception or after 5 consecutive `RX_TIMEOUT` interrupts, the sensor power is turned off, by disabling the General Purpose Input/Output (GPIO) and then the STM32 enters the stop mode, after setting a wakeup period.

Therefore, the High Speed Internal (HSI) clock is enabled after waking up with the STM32L151RC and the sensor power is turned on.

Also, it is important to notice that the microprocessor always needs to go to deep sleep mode after SX1276 and, especially, after setting the wakeup period, otherwise the LoRa module may never wake up again.

Moreover, the UART was disconnected during the measurement to reduce the consumption.

4.2.1 Results

In order to measure the current consumption, a resistor of 10Ω was used to calculate the voltage due to the current supply. Therefore, the current values, shown in Table 4.1 have been computed by means of the simple formula $I = V/R$. The transmit case is related to a transmit output power of $+14dBm$

Table 4.1: Current measurements without UART, with antenna

	Tx	**Rx**	**Sleep LoRa**	**Sleep LoRa/STM32/Sensor**
Current [mA]	75	20	12	1.2

Then, LoRa sleep mode allows to save around $8mA$, while the microcontroller and the sensor together $10.8mA$. Considering the time spent in transmit mode, its current contribution can be neglected compared to the others.

4.3 Proposed solutions

4.3.1 LoRa node

The structure of the packet sent by the end-node has been optimized, even though the size was maintained equal to 10.

A `NET_ID` field was added and set to 56, in order to filter and recognize only the LoRa packets belonging to this network.

Moreover, the RSSI bytes have been deleted because not strictly necessary on the node side.

On the other hand, a field for `TX_OUTPUT_POWER` has been appended to help the power management.

To sum up, the new packet has been defined as follows: ID number, message number, 2 symbols for temperature, 2 symbols for humidity, 2 symbols for voltage level, network ID, transmit output power.

`LORA_SYMBOL_TIMEOUT` was increased up to 11 symbols, value found by inspection, in order to grant a successful packet reception.

In `OnRxDone` callback, a condition is added to filter only the packets whose buffer size is equal to 3, the `ID_NO` of the incoming packet is equal to the Identification Number (ID) of the current node and the network ID is correct.

`OnRxError` function has been modified too, since the testing phase allowed to notice that the program used to early block here. This issue was managed by calling again `radio.Rx(RX_TIMEOUT_VALUE)` function, in order to remain in receive mode.

`ticker_callback`, which is the function recursively called, has been changed to avoid to always read the sensor data and the battery value in case of receive timeout, which means that the answer from the gateway is not received. In fact, these operations must be done only at the first occurrence of the timeout interrupt, i.e. if the counter variable is equal to 0. In the original version, indeed, the device used to perform the measurements at each attempt of packet transmission, increasing at each time the message number value. That way, the gateway used to receive up to 5 different packets almost simultaneously, even though the first one was actually processed, with a consequent waste of energy on the sensing node.

Consequently, with this update, each device performs only one measurement every 30 minutes.

Instead of continuous receive mode, the single receive mode is used to set up the modem, by changing the code as follows:

```
  radio.SetRxConfig(
MODEM_LORA,
LORA_BANDWIDTH,
LORA_SPREADING_FACTOR,
LORA_CODINGRATE,
0,
LORA_PREAMBLE_LENGTH,
LORA_SYMBOL_TIMEOUT,
LORA_FIX_LENGTH_PAYLOAD_ON,
0,
LORA_CRC_ENABLED,
LORA_FHSS_ENABLED,
LORA_NB_SYMB_HOP,
LORA_IQ_INVERSION_ON,
false);
```

The last parameter corresponds to `rxContinuous`, that sets the reception in continuous mode if true, single mode if false. Also, another difference to be taken

into account is the value of `RX_TIMEOUT_VALUE` given to the function used to set the radio in reception mode for a given time in microseconds, which is equal to 0 for continuous receive mode.

As suggested by the removal of the corresponding packet field, the part related to the bit conversion of the RSSI of the gateway has been removed, limiting thus the computational complexity, which is better to move to the gateway side, not power constrained.

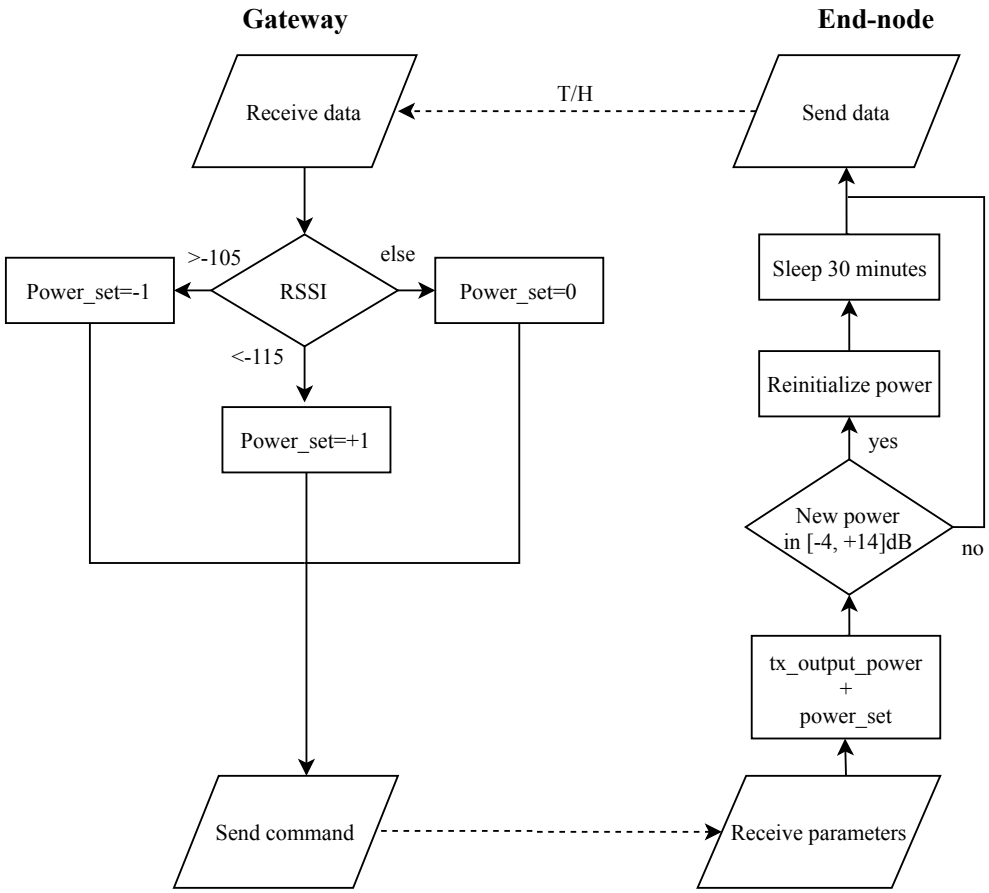


Figure 4.1: Power management algorithm

Finally, `RX_DONE` case has been updated too. After checking the destination node ID of the received packet, the value contained in `Power_set` field is considered and used to reinitialize the transmit radio modem as follows:

- `Power_set=0`: the packet is processed with no effects.
- `Power_set=`$\pm$1: this value is added to the actual `TX_OUTPUT_POWER`.

If the new value is in the range $[-4, +14]$, the `Reinitialize_TX_Modem` function is called with this new parameter for the power setting. Differently from the original version, the power range has been extended in order to achieve lower power levels.

The procedure for the transmit output power value adjustment is summarized in Figure 4.1.

4.3.2 Gateway

Similarly to the LoRa node case, the `NET_ID` constant was added, while `RX_TIMEOUT_VALUE` has been set equal to 0, since the continuous receive mode is employed here.

The packet structure has been reduced from 4 to just 3 symbols: destination node ID, network ID and the power command, which can assume the values 0, +1, -1.

In `OnRxDone`, the check on the `NET_ID` and buffer size was added. In this case, the continuous receive modem is used.

Reasonably, the old part related to the bit conversion for the RSSI symbols sent by the end-node has been removed, since the RSSI is measured only by the gateway.

Before printing the data on the Raspberry Pi to update the ThingSpeak channel, the program used to wait 17 seconds, which can be here avoided since ThingSpeak is replaced by another server, that does not require a delay.

Moreover, a filter for redundant received packets was added, since now the end-node can retransmit the same packet up to 5 times in case of 5 consecutive receive timeout interrupts, in the worst case. For this reason, a support buffer has been created to save the previous values for the node ID and message number of received packet. Any new incoming packet is processed only if these two fields are different from the old ones.

Finally, the transmit output power adjustment is modified. If the RSSI value is higher than -105, `Buffer[2]`, which is the field hosting the power command, is set to -1, in order to reduce the transmit power. Instead, if it is lower than -115 the gateway asks the end-node to increase the power of one unit, otherwise, it will forward 0.

Chapter 5

Server

5.1 Overview

The purpose of this phase is to build our own server in order to be able to replace the use of ThingSpeak, which does not offer the full ownership of data to the end user.

Initially, while waiting for the necessary permissions from the administration, a local database was designed and implemented on the Raspberry Pi by means of the Linux, Apache, MySQL, PHP/Perl/Python (LAMP) stack, a group of open-source software that is typically installed together to enable a server to host dynamic websites and web applications. Thus, an Apache web server was exploited together with a MySQL database to store the data. The dynamic content of the web user interface is processed using PHP language.

The final version of the server relies on an Internet Protocol (IP) address assigned by the administration, which makes it accessible only inside the university Local Area Network (LAN).

5.2 Database

Figure 5.1 represents the structure of the table used to store data in the MySQL database `ensil_wsn_db`. The first column is related to the ID number of the current entry in the table, therefore it has been set as an integer with an auto-increment primary key. The other columns are for the source ID of the end-node, the message number, the temperature in degree Celsius, the humidity in percentage, the battery level in millivolt, the date string and the transmit output power used by the LoRa node in decibel.

Id	srcId	msgN	temp	hum	battery	date	power
1	2	0	19.5	40.2	3000	2019-11-18 14:55:42	0
2	5	0	20.5	39.2	2000	2019-11-18 14:56:06	0
4	8	0	23.5	25	3400	2019-11-20 14:00:09	0
5	9	0	18.5	35	3900	2019-11-20 14:00:26	0
6	9	0	28.5	45	1900	2019-11-20 14:00:41	0

Figure 5.1: Table `t_h_data`

5.3 Raspberry Pi

Figure 5.2: Insertion of data in the database

Once the packet is received by the gateway, the program installed on the Raspberry Pi reads and extracts the values of each field and adds the current date and time. Then, it exploits an HTTP GET request with the format `key=value` for each parameter in order to upload the data to the server, where the file `upload.php` is in charge of inserting the new records to the table. The procedure is graphically represented in Figure 5.2.

5.4 Architecture

The server, located at `loraweb.unilim.fr`, hosts five different programs: `upload.php`, `index.php`, `db.php`, `download.php` and `delete.php`. A further file, `global.php` collects the variables used to store the names of the server, user and database and the secret password for deleting the database records.

`upload.php` is in charge of reading the data sent via HTTP by the program running on the Raspberry Pi and updates the database.

`index.php` contains an HTML form, used to apply the desired filters to the records of interest for the user. It allows to choose one specific source ID, which is

correlated to the sensor location, through a drop down menu. Moreover, a range of values can be selected for the humidity, temperature, battery and datetime. These values are then saved inside the PHP session as global variables, used to apply the MySQL query. This webpage is shown by default when accessing the server url. The procedure is shown in Figure 5.3.

Figure 5.3: Data selection: `index.php`

`db.php` applies the query to the database according to the requests submitted by the user and shows the resulting records. It allows to send a request for deleting some entries, sensitive operation that is allowed only by entering a specific password. A download button is inserted to eventually download a .csv file containing the selected records. This last function is managed by `download.php`, where the PHP session is ended and all the global variables which were previously set are destroyed. The data retrieving and download request are represented in Figure 5.4.

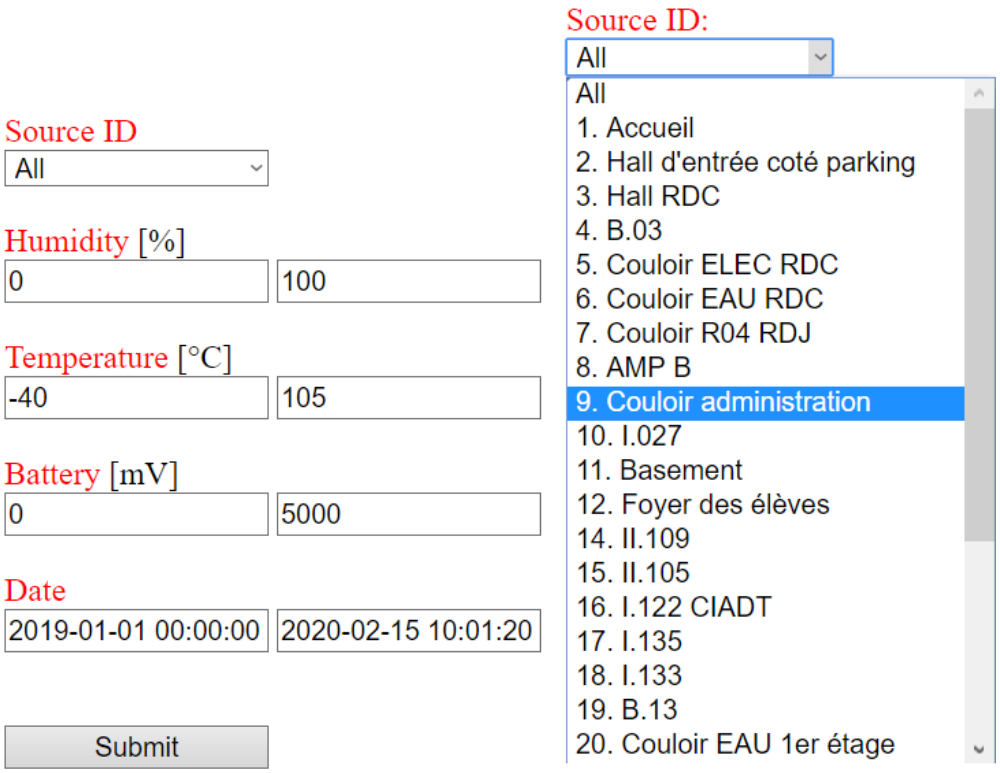
Figure 5.4: Data retrieving: `db.php`

After getting the .csv file, if triggered by the proper button in `db.php`, `delete.php` applies a delete query to the database using the parameters chosen by the user in the previous webpage in case of correct password. This last procedure is described in Figure 5.5.

Figure 5.5: .csv file download and record removal: `download.php`, `delete.php`

5.5 Web user interface

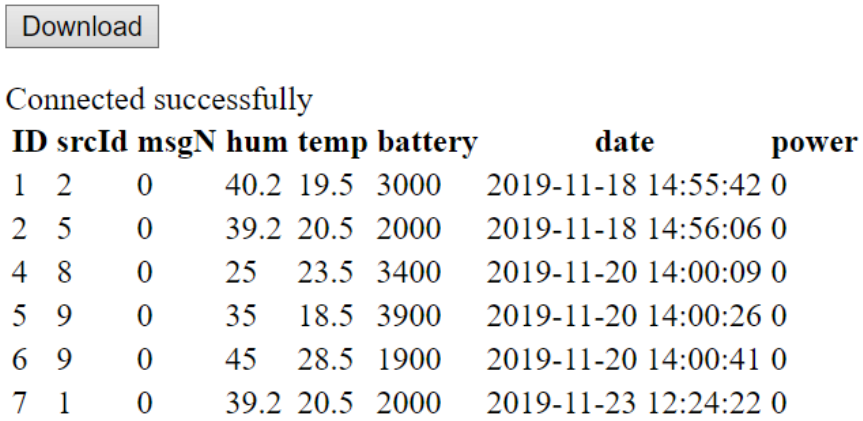
The proposed user interface consists of a first form in which the desired records can be selected by means of a list of range values for the temperature, humidity, battery and period of time, as shown in Figure 5.6a, where a drop down menu, visible in Figure 5.6b, allows to select the sensor ID of interest, or all. To ease the selection, each source node is correlated with the actual position.



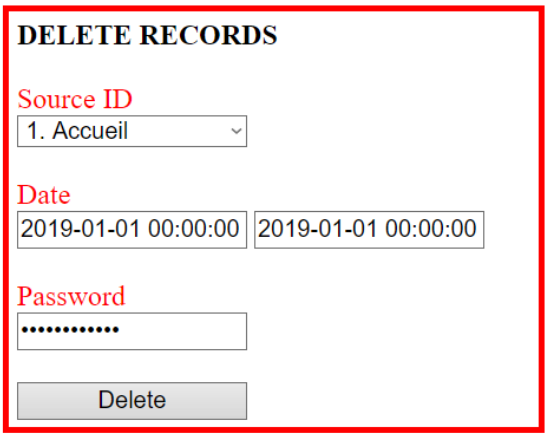
(a) *Insert filter webpage* **(b)** *Drop down menu for source ID*

Figure 5.6: `index.php` webpage

After that, the user can press the download button in order to get a .csv file containing all the current records that are visualized on the webpage, as shown in Figure 5.7a. Moreover, if necessary, some records can be deleted, by means of a new MySQL query of type `DELETE`, whose parameters are chosen again though a filtering mask, as represented in Figure 5.7b. If the password is not correct, an error message is visualized.



(a) *Data visualization*



(b) *Deleting data procedure*

Figure 5.7: `db.php` webpage

Chapter 6

Conclusions

6.1 Conclusions

The purpose of this part was to improve the adopted techniques at the power management level of the LoRa sensor network installed in the building of ENSIL. Moreover, a new server was necessary in order to replace the use of ThingSpeak, together with a web user interface to retrieve the data.

The main goal was to reduce the energy consumption of the sensor nodes, in order to increase the battery lifetime. To accomplish this task, some changes have been applied to both the node and gateway C++ programs.

For what concerns the packet structure, a field containing the network identification number has been added to both sides to perform packet filtering.

Other additional fields have been introduced since required by the new power management algorithm, such as the actual transmit output power for the LoRa node packet. This value, in fact, together with the RSSI is taken into account by the gateway to properly reinitialize the transmit radio modem. According to the RSSI, the transmit output power of the sensing node is increased or decreased by 1, in the range $[-4, +14]dB$, which has been extended with respect to the state-of-the-art version in order to achieve lower power values. To deliver this command, a field in the packet coming from the gateway is used to communicate if the power must be increased, decreased or not.

Some complex operations, such as the RSSI bit conversion have been moved to the gateway side only, to limit the computational complexity of the sensor node, indeed the gateway has no restrictions in terms of power supply.

Moreover, at the sensing node side, the Single RX mode replaced the Continuous RX mode to further limit the power consumption. Then, to avoid to consume unnecessary energy, a correction has been applied to the program to make that in case of receive timeout events, the node attempts to retransmit the same packet

and not a new one, which would derive from a new measurement.

At the same time, a filter for redundant packets has been added to the gateway side by means of a support buffer, in order to print on the Raspberry Pi only the first occurrence of that packet.

These operations gave a contribution to the wireless sensor network, in terms of performance of the microcontroller program, by means of bug correction and optimization of power management choices.

On the other side, a new server based on a MySQL database has been designed to allow the data retrieving, visualization and download to ease the data analysis performed by the researchers in charge.

Also, a web user interface has been created by means of PHP language, to allow the user to search for the data related to a particular room, a given period of time or other criteria.

6.2 Future works

Some parts of this LoRa sensor network can be further developed and improved. This is the case of the gateway, which could be replaced by a new one able to work with more than just one channel and a unique spreading factor. Indeed, the exploitation of different channels and the adaptive control of spreading factors of LoRa nodes could enhance the orthogonality of LoRa signals and reduce the errors due to packet collisions, which could be significant in case of more sensing devices.

Moreover, the web user interface is still basic and could be improved by means of a testing phase, which might be useful to get several feedbacks on the user-friendliness of the HTML application.

Part II

Robustness analysis of TX-RX LoRa signals using LabVIEW

Chapter 7

State-of-the-art

7.1 Introduction

As anticipated in Chapter 2, LoRa PHY layer protocol is proprietary, therefore, special hardware provided by some companies, e.g. Semtech, is required for the transmission and reception of LoRa packets.

Semtech, a major provider of LoRa devices and services, has published some documents about LoRa modulation basics [7]. In addition to this, the European Patent Application [14], whose inventors are Olivier Bernard André Seller and Nicolas Sornin, is available.

However, the information provided by these public documentations is not sufficient to build a decoder able to interoperate with real hardware LoRa transceivers.

The full access to the LoRa PHY layer may have interesting implications in terms of research and development of this wireless technology. Indeed, the possibility of applying modifications to the PHY layer could improve the protocol security and performance. Moreover, it could enable some software simulations of the PHY layer, in order to study the effects of different noise channel conditions without the need of multiple hardware transceivers.

Several are the attempts of decoding LoRa modulation scheme, by means of a reverse engineering approach, such as the already cited Matthew Knight, who presented a first complete analysis of the PHY layer. However, in his work the interleaving pattern is assumed to be different from the one described in the patent application, fact that was not confirmed by other studies [27].

Furthermore, different open-source implementations [27] [15] [28] found different whitening sequences, due to the fact that obtaining an error-free transmission is quite difficult.

Moreover, an error in the reverse engineering process highly affects the performance.

7.2 Related works

With reverse engineering in mind, several studies tried to exploit the acquired information, in order to be able to simulate the physical layer and analyse LoRa robustness. In particular, the Bit Error Rate (BER) performance of LoRa modulation is crucial to be able to assess the robustness. Closed-form approximations were derived for both Gaussian noise and Rayleigh fading channels [29]. The error rate of LoRa-like modulation was deeply studied in range analyses [30].

A study [31] confirmed the good transmission properties of CSS modulation over time and frequency selective channels and, then, it implemented channel coding, whitening and interleaving using GNU Radio, showing the beneficial gain. On the other hand, the large propagation losses due to the long range and the larger footprint make CSS prone to collisions with other noise sources, possibly greater than the coding gain [32].

Another one [33] performed a Frame Error Rate (FER) analysis that also includes the channel coding, interleaving and Gray mapping of LoRa physical layer.

The performance gain of channel coding was studied in [34] and compared to uncoded transmissions, in both Additive White Gaussian Noise (AWGN) and multipath channels.

Others based their research only on the theory. A numerical approximation for the LoRa BER performance under the same SF interference was derived in [35], but it does not take into account the channel coding, whitening and interleaving stages.

A further interference analysis demonstrated that interfering signals from other LoRa devices cause significant errors in reception if they have the same chirp rate [36].

None of these works focussed the attention on the interference case with narrowband signals.

Chapter 8

Architecture Design and Implementation

8.1 Assumptions

Before the description of the architecture design of the TX-RX LoRa signal, some assumptions must be taken into account, as follows:

- Preamble detection and synchronization: for simplicity, the synchronization is assumed to be perfect, therefore, the LoRa packet does not contain the preamble nor the header field in its structure. Moreover, CRC is not appended to the payload.

- Data whitening, that should be applied to induce randomness into the symbols to provide more features for clock recovery, is not included in this simulation. Indeed, it has a minimal impact on the bit error rate [34].

- Hamming Code is used to perform FEC with 3 parity bits, since Hamming(8,4) does not provide any advantage in terms of error correction with respect to Hamming(7,4).

- Diagonal interleaving, with a downward direction, even though it appears to have an upward direction [27], in contrast with LoRa patent specifications. However, this detail does not change the interleaver performance in the simulation, indeed a completely corrupted chip continues to affect only one single bit per codeword.

8.2 Design

Figure 8.1 shows the new LoRa block diagram which takes into account the assumptions.

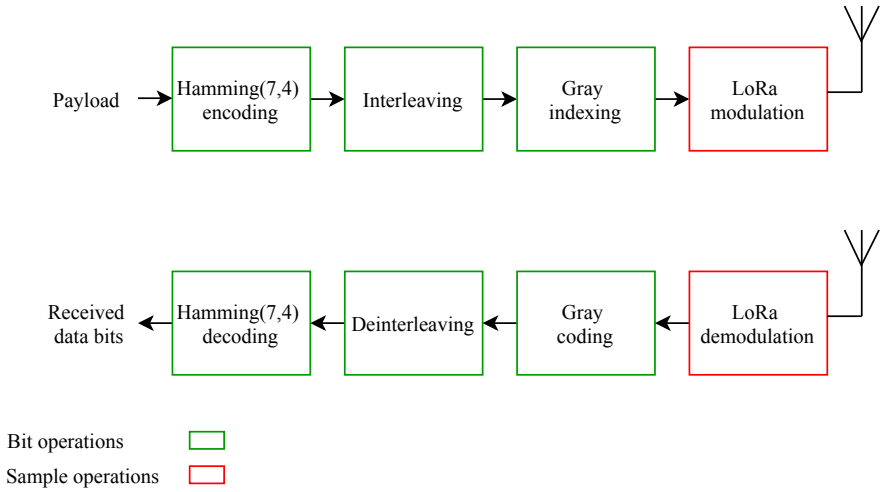


Figure 8.1: Simplified LoRa PHY block diagram

Moreover, the demodulation stage is performed as illustrated in Figure 8.2. The modulated symbol is multiplied by the reference downchirp, which can be set as the complex conjugate of the upchirp signal of the symbol 0.

Then, the magnitude of the Fast Fourier Transform is applied to the resulting signal.

Finally, the received symbol is estimated by computing the index of the maximum.

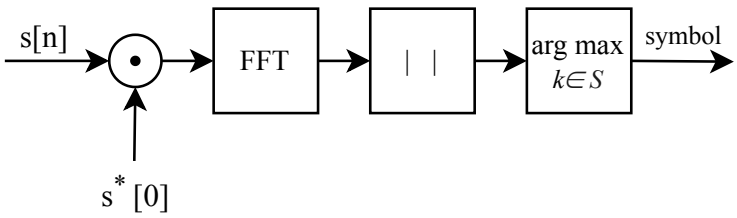


Figure 8.2: Demodulation procedure

8.3 Implementation

8.3.1 LabVIEW

Figure 8.3: LabVIEW logo [37]

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) (logo in Figure 8.3) is a proprietary product of National Instruments [38], offering a graphical programming language. It allows to visualize every aspect of the application, including hardware configuration, measurement data and debugging.

In this context, it has been used to simulate a TX-RX LoRa signal.

8.3.2 Modulation

Before applying all the coding techniques, the modulation stage must be taken into account.

Below, the generation of the LoRa symbol baseband is presented, together with the complex conjugate of the reference chirp.

Finally, the demodulation phase, which allows to retrieve the original symbol, is presented.

LoRa symbol baseband

A modification of the equation (2.2) has been derived for the phase of LoRa baseband:

$$\phi = 2\pi \left[\frac{B}{2T_s} (ndt)^2 \right] \tag{8.1}$$

Where the time has been discretized and n can assume any integer value in $[-\frac{N}{2}, +\frac{N}{2}]$. Also, dt is equal to $\frac{1}{OSF \times B}$, with OSF representing the oversampling factor.

Figure 8.4 shows the block diagram of the LoRa symbol baseband generation. On the left, the spreading factor, bandwidth and symbol are given as input, where the last one should be a natural number in the range $[0, 2^{SF}]$.

Then, a `for` loop is added, where N is the loop count, set equal to $2^{SF} \times OSF$ and i is the loop iteration. The result of the loop is the phase that is used to modulate the given symbol.

After that, the complex number is derived from the polar representation, with $r = 1$.

In Figure 8.5 the front panel with all the mentioned inputs and output is visible. For what concerns the output, for brevity's sake, only the first entries are shown.

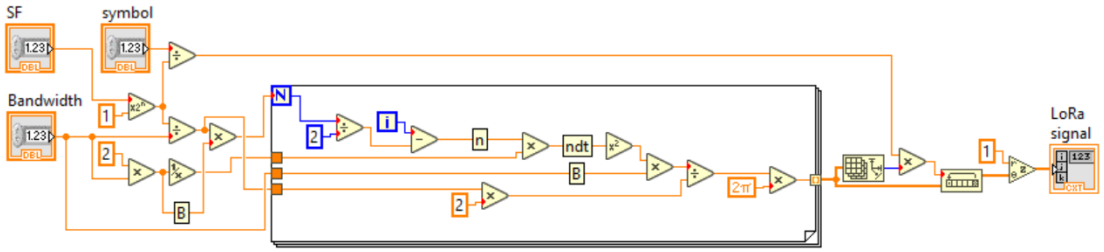


Figure 8.4: LoRa symbol block diagram

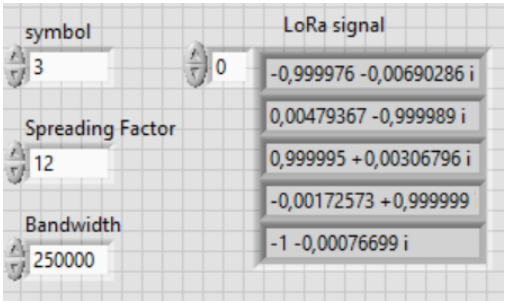


Figure 8.5: LoRa symbol front panel

Downchirp

Figure 8.6 shows the downchirp generation. It is important to maintain the same value as the LoRa symbol baseband for the oversampling factor and, in order to get the complex conjugate, the sign minus is added at the end of the phase computation.

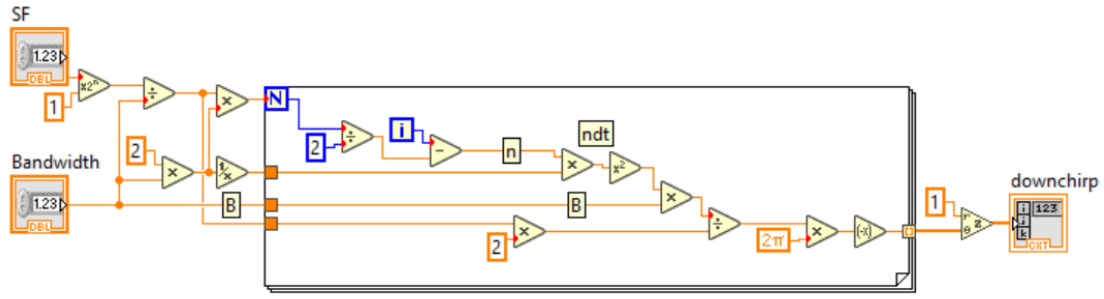


Figure 8.6: Downchirp block diagram

Demodulation

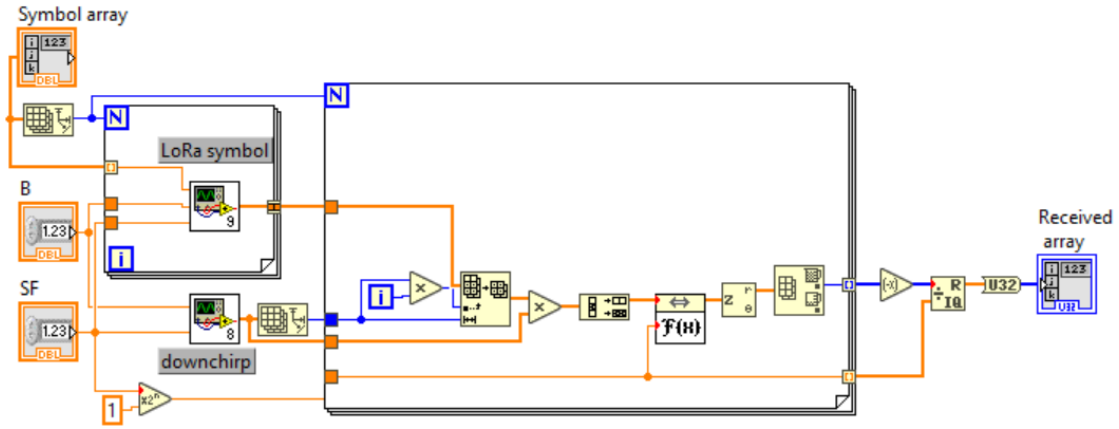


Figure 8.7: Demodulation block diagram

For what concerns the demodulation process, the LoRa symbol baseband generation is inserted into a `for` loop whose N is equal to the size of the array of input symbols, as shown in the left loop in Figure 8.7. At the same time, the downchirp is created below.

Then, a `for` loop takes as input the incoming LoRa signal and multiplies each component by the downchirp. After that, since an oversampling factor was used, the array is decimated according to the value of OSF.

At this point, to the first resulting decimated array, the FFT is applied with size equal to 2^{SF}. Afterwards, the complex number is converted into its polar representation and the modulus is considered to take the maximum index of the array.

Finally, by means of `Quotient and Remainder Function`, the received symbols

are extracted, and, since it is still an ideal case without any source of interference, they must be coincident with the original ones.

8.3.3 Coding

In order to increase the robustness to interference, several operations are applied to payload data before being modulated.

The following blocks are all inserted into a general `for` loop that allows to simulate the desired number of LoRa packets.

Hamming encoding

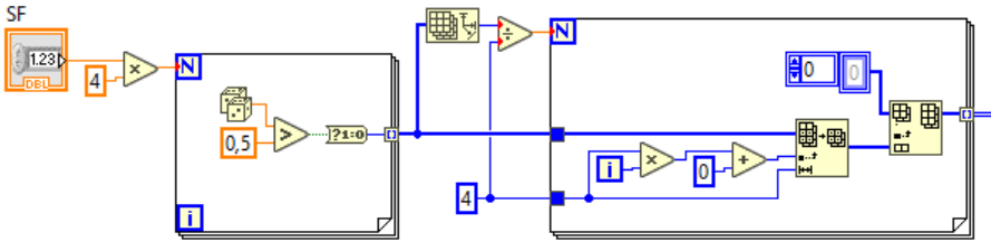


Figure 8.8: Input data bit generation block diagram

First of all, the Hamming(7,4) algorithm requires a bit matrix $\{0,1\}^{SF\times 4}$, i.e. SF arrays of 4 data bits, a monodimensional array of length $SF \times 4$ is randomly created in the first loop in Figure 8.8. After that, the second loop arranges the bits into a real 2D array, with SF arrays each of 4 bits.

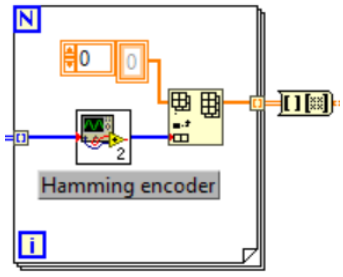
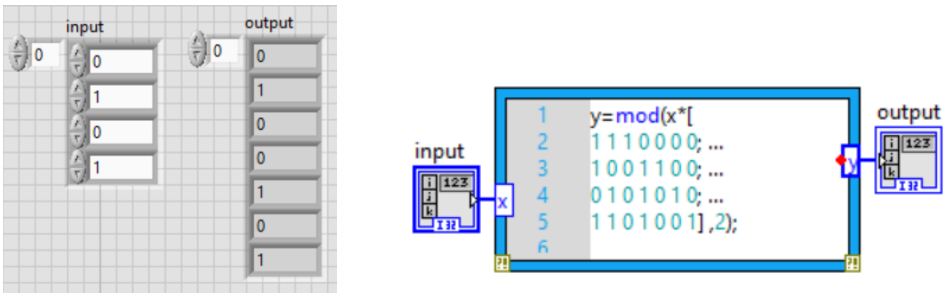


Figure 8.9: Hamming encoder loop block diagram

The loop in Figure 8.9 applies the Hamming encoding algorithm to each subarray of 4 bits. The output is a bit matrix $\{0,1\}^{SF\times(4+CR)}$, where $CR = 3$ is the number of parity bits.

Inside the loop, a Virtual Instrument (VI) manages the Hamming(7,4) algorithm, whose front panel is shown in Figure 8.10a. The block diagram in Figure 8.10b was built by exploiting `LabVIEW MathScripts Node` provided by the MathScript RT Module engine. The four input data bits, represented by vector x, are multiplied by the code generator matrix [39] and, then, taken modulo 2. This way, the original data bits are encoded and converted into an array y of 7 bits.



(a) *Hamming encoder VI front panel*

(b) *Hamming encoder VI block diagram*

Figure 8.10: Hamming encoder VI

Interleaving

Figure 8.11 represents the interleaving stage. The input is a matrix $\{0,1\}^{SF\times(4+CR)}$, which is immediately converted into a real 2D array, in order to be compliant with the type of input of the `Rotate 2D Array` VI [40], which performs a rotation of 180°.

Then, the 2D array is transposed in order to perform a downward circular shift by column index with `Rotate 1D Array Function` inside a `for` loop.

Finally the 2D array is transformed into a matrix $\{0,1\}^{(4+CR)\times SF}$.

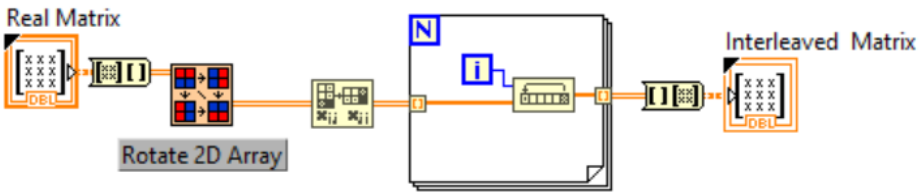


Figure 8.11: Interleaving block diagram

Gray Indexing

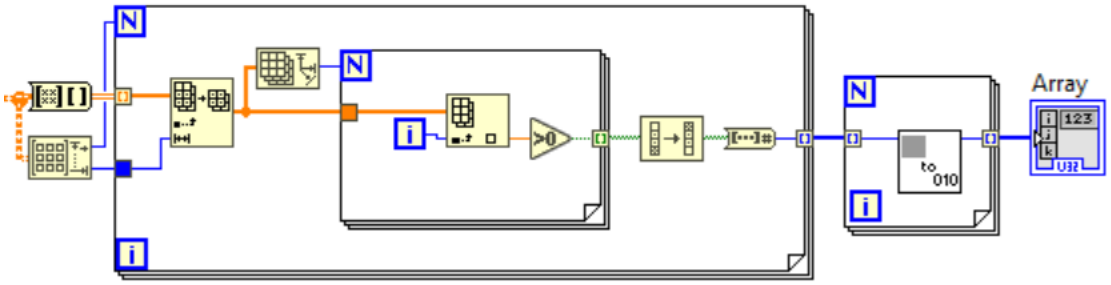


Figure 8.12: Gray indexing block diagram

In Figure 8.12, the matrix coming from the interleaving block must be converted into a 1D array of $4+CR$ symbols. The two nested loops have the function of converting each binary number with SF bits to its equivalent decimal representation.

Therefore, a further `for` loop applies the Gray decoding algorithm [41], whose front panel is shown in Figure 8.13.

The resulting array is given as input to the modulation block presented in Chapter 8.3.2.

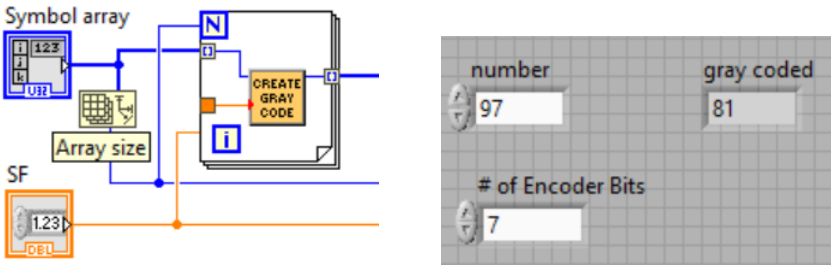
Figure 8.13: Gray indexing VI front panel

8.3.4 Decoding

After the demodulation, all the channel coding techniques are applied in reverse.

Gray Coding

The `for` loop in Figure 8.14a takes as input the array of demodulated symbols, the array size, which is still $4+CR$, and the value of the spreading factor. These parameters are required by the `Create Gray Code` VI [41], whose front panel is visible in Figure 8.14b.



(a) *Gray coding block diagram* **(b)** *Gray Code VI front panel*

Figure 8.14: Gray coding

De-interleaving

Before applying the opposite operation of interleaving, the array of symbols must be mapped into a matrix $\{0,1\}^{(4+CR)\times SF}$. In the `for` loop, each symbol is converted into an array of SF binary bits, by means of some array manipulation blocks.

Then, the 2D array is transformed into a matrix. The related block diagram is represented in Figure 8.15.

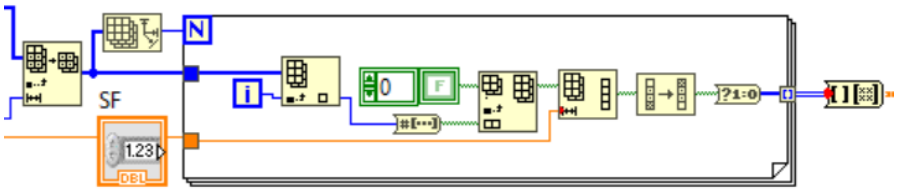


Figure 8.15: Symbol to bit matrix conversion block diagram

After that, the obtained matrix is given as input to the De-interleaving VI, whose schema is visible in Figure 8.16. The matrix is converted into a 2D array and a `for` loop is used to apply the downward circular shift by column index.

After the transposition, the 180° rotation is applied and the deinterleaved matrix is obtained.

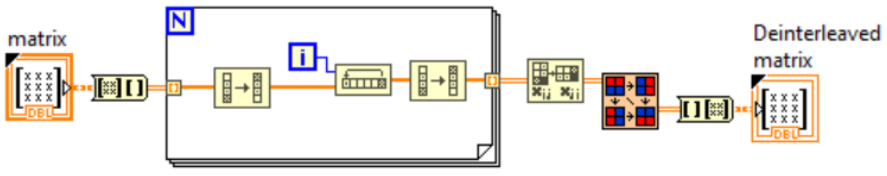


Figure 8.16: Deinterleaving VI block diagram

Hamming decoding

Figure 8.17 shows the bit operations applied for the FEC. The incoming matrix is converted into the equivalent 2D array in order to retrieve each subarray of $4+CR$ bits, that are given as input to the Hamming decoder VI. The result is an array of 4 bits, which must be coincident with the original ones sent by the transmitter side. After the `for` loop, the matrix is converted into a 1D array.

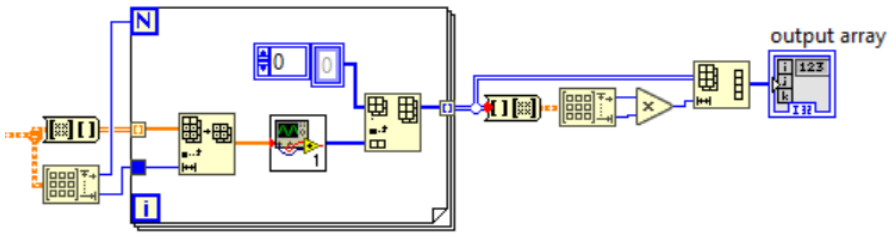
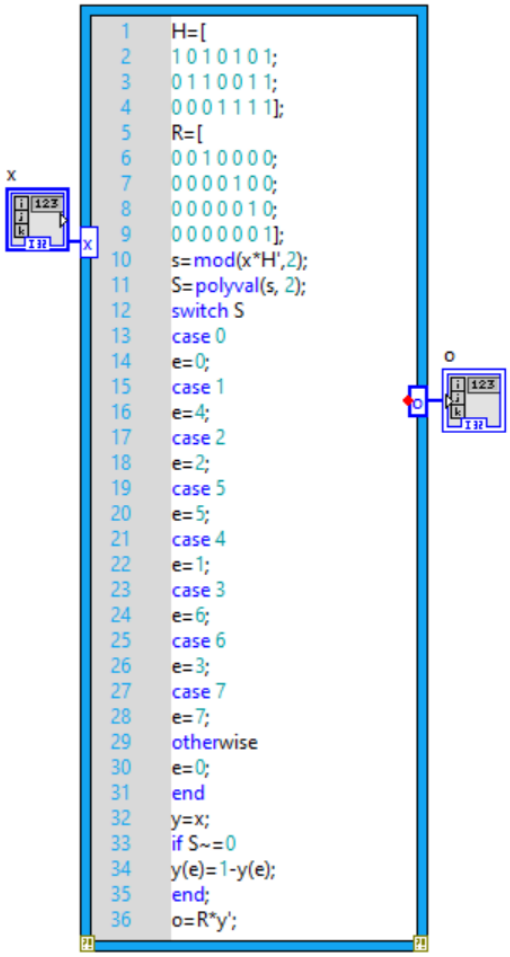


Figure 8.17: Hamming decoding block diagram

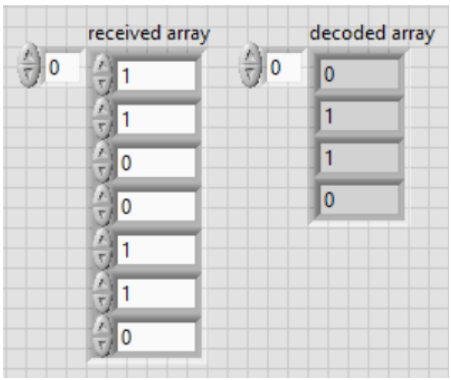
In this `LabVIEW MathScripts Node` in Figure 8.18a, the parity-check matrix $\mathbb{H}$ is defined to compute the syndrome vector at the receiver side, by means of the modulo 2 operation between the received bit array and $\mathbb{H}$. According to the non-zero value of the syndrome vector, computed by means of `polyval` function, the bit that was eventually flipped can be assessed and corrected. The correspondence between the value of $S = polyval(s,2)$ and the bit error position is defined in the `switch`.

Finally, the bits can be decoded to the original four bits, by means of the matrix $\mathbb{R}$ which is multiplied by the transposed corrected vector y.

An example of conversion is provided in the front panel in Figure 8.18b.



(a) *Hamming decoder VI block diagram*



(b) *Hamming decoder VI front panel*

Figure 8.18: Hamming decoder VI

Chapter 9

Robustness analysis

9.1 Introduction

The analysis of LoRa robustness in noise channels has been performed considering two scenarios as follows:

- LoRa modulated signal, uncoded, with Gray code.
- LoRa modulated and coded signal.

In this simulation the presence of Gray code does not much affect the error calculation, because all symbol errors are equally likely when Carrier Frequency Offset (CFO) is not introduced [34].

Figure 9.1 shows the second case, including all the channel noise sources that will be taken into account in the simulations. The Additive White Gaussian Noise (AWGN) is added to the modulated signal, together with the Narrow Band (NB) interfering signal. At the receiver side, the demodulated and decoded bits are compared with the transmitted ones, in order to compute the bit error rate.

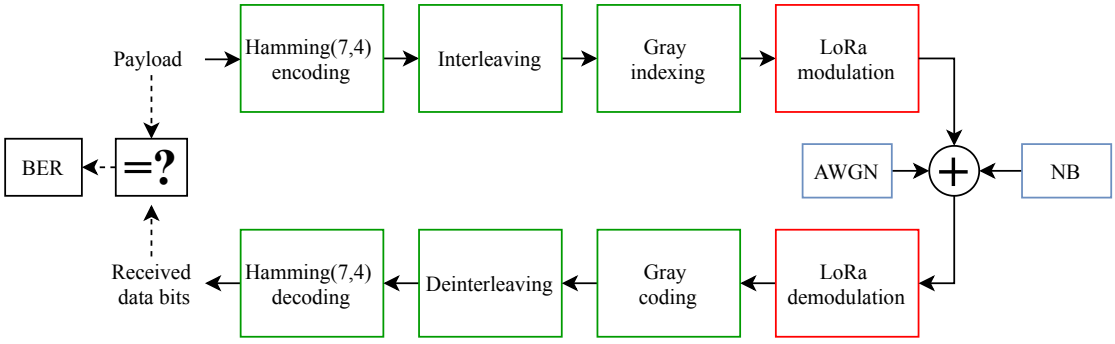


Figure 9.1: LoRa PHY block diagram with channel interference

9.2 AWGN channel

By definition, the SNR is defined as the ratio between the signal power and the noise power as follows.

$$SNR = \frac{P_{signal}}{N_0 B} \tag{9.1}$$

Since the modulus of the polar representation of LoRa symbol was set equal to 1, $P_{signal} = 1$. Consequently, the relation in (9.2) can be derived:

$$\sigma^2 = N_0 B = \frac{1}{SNR \cdot B} \cdot B = \frac{1}{SNR} \tag{9.2}$$

9.2.1 Implementation

The AWGN is added to the modulated signal as a complex number whose real and imaginary parts follow a Gaussian noise pattern with standard deviation equal to $\frac{1}{\sqrt{SNR}}$.

Figure 9.2 shows the block diagram of the LabVIEW implementation. On the left, the SNR value is converted from decibel into decimal and it is divided by two. Then, the square root is taken to derive the standard deviation, which is given as input to two `Gaussian White Noise VI` blocks, respectively for the real and imaginary part. The size of the modulated signal is used for the number of required samples.

Finally, the two parallel outputs are used to derive a complex number, which is added to the modulated signal.

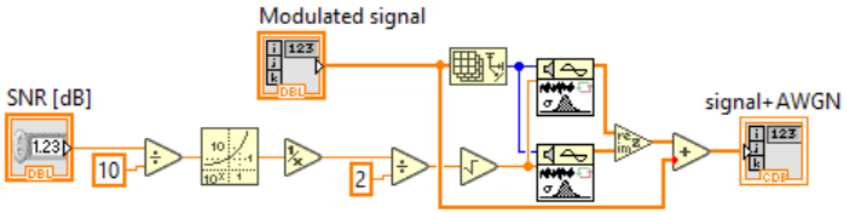


Figure 9.2: AWGN block diagram

In the real implementation, a for loop is used to compute the BER for different values of SNR, as shown in the front panel in Figure 9.3

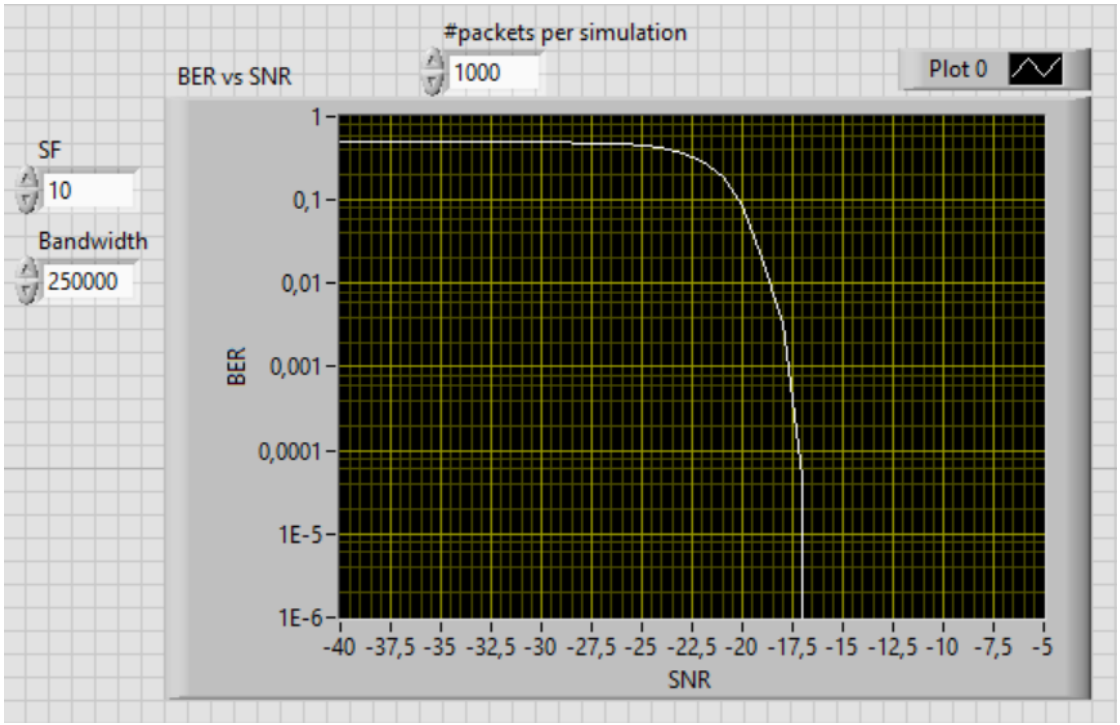


Figure 9.3: BER vs SNR front panel

9.2.2 Results

This part shows the simulation results of the error rate with varying SNR, identified as Γ, without and with channel coding and interleaving.

Uncoded

This section treates the graphical validation of the obtained SER curves, by means of the comparison with the theoretical and experimental results of Ferré and Giremus [9], under the same condition of AWGN. To do this, a conversion from BER to SER was required and derived by approximating the probability of symbol error rate with twice the probability of bit error rate.

The theoretical approximation for the SER is reported for completeness in (9.3), where $N = 2^{SF}$, N_{MC} is the number of iterations for the Monte Carlo approximation, W is a Gaussian noise sample with variance σ^2 and $F_{\chi 2}$ is the cumulative density function of the $\chi 2$ distribution with 2 degrees of freedom:

$$P_{wrong\_symbol} = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} 1 - \left(F_{\chi 2} \left(\frac{\left| \sqrt{N} + W^{(i)} \right|^2}{\sigma^2} \right) \right)^{N-1} \tag{9.3}$$

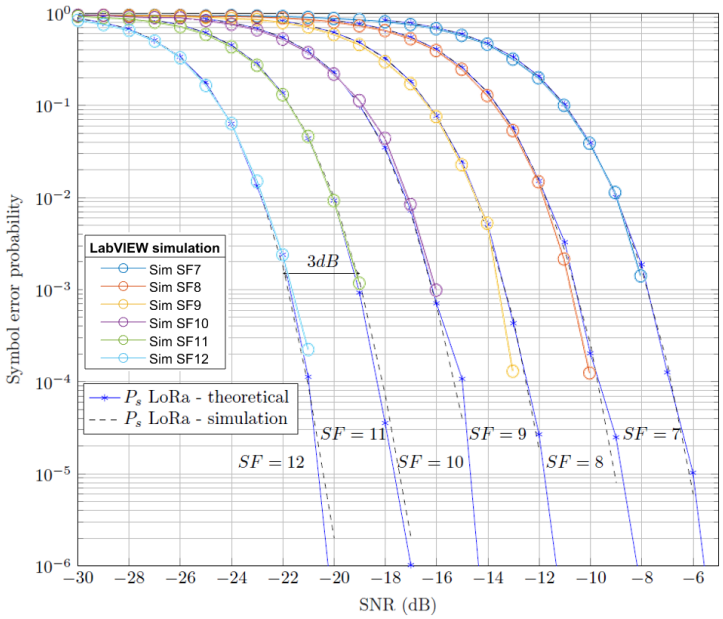


Figure 9.4: SER vs SNR, without channel coding. Comparison with reference plot ©2018 IEEE

Figure 9.4 was built by superimposing the LabVIEW simulation curves to the graphical results of the cited authors.

From the comparison between the theory and simulation, small differences can be spotted in the length of the simulated curves that are cut for low error values. The reason lies in the limited number of packets per simulation, that does not allow to detect error with a very high precision.

Besides that, the significant level of overlapping of the graphs proves the reliability of the LabVIEW simulation.

In general, increasing the SF of one unit reduces the SNR of around $3dB$.

Table 9.1 collects the number of packets that could be held by the simulation with a given SF.

Table 9.1: Number of packets per simulation

SF	**#packets**
7	9000
8	8100
9	7200
10	6400
11	5700
12	4000

Coded

Figure 9.5 represents the resulting BER curves that are obtained after applying channel coding.

Considering a BER of around 10^{-3}, this additional operation provides a gain of about $-1dB$, which is expected to increase with the decrease of the error value if the trend is held.

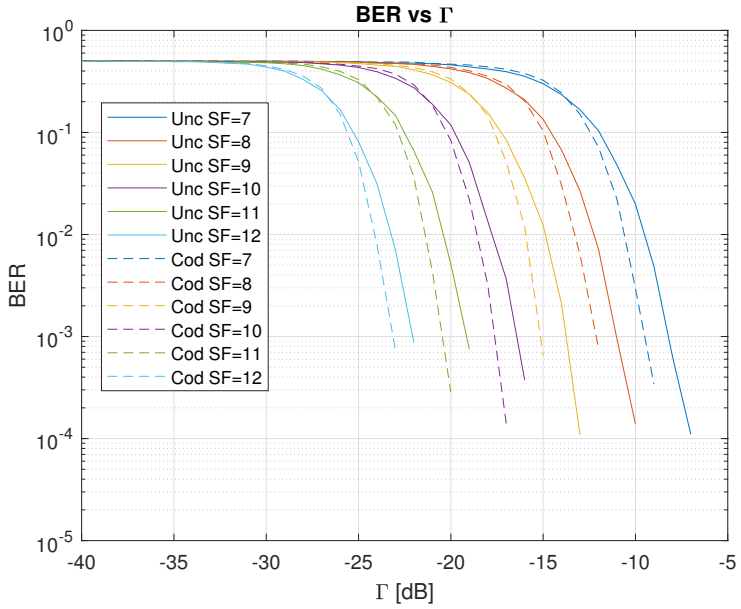


Figure 9.5: Comparison between coded and uncoded BER vs SNR

9.3 Narrowband interference

This section presents the analysis of the LoRa performance in an AWGN channel, with the addition of a narrowband interference to both uncoded and coded cases.

The study takes into account the BER performance with varying spreading factor, narrowband power and SNR.

9.3.1 Implementation

In order to generate a narrowband signal, two `Sine Wave VI` blocks have been employed, one for the cosine, the other one for the sine. The former is obtained by changing the `phase in` to 90°.

The array size of the LoRa signal is given as input for the number of samples to the VI.

The amplitude, which is the second input of the block, assumes values in the range fom 0 to 70, which are later converted into power in decibel.

The third parameter required by the `Sine Wave VI` is the frequency in normalized units of cycles per sample, that can be derived by means of $\frac{f}{B \times OSF}$. The allowed range of normalized frequency is thus $\left[-\frac{f}{B \times OSF}, +\frac{f}{B \times OSF} \right]$, where f is a value between $\left[-\frac{B}{2}, +\frac{B}{2} \right]$.

Finally, the output of cosine and sine are used as real and imaginary part, respectively, to derive the complex signal.

Figure 9.6 shows the LabVIEW block diagram used for the narrowband interference implementation.

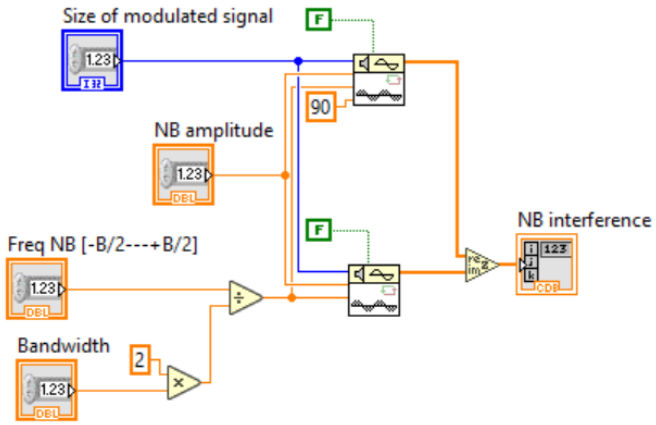
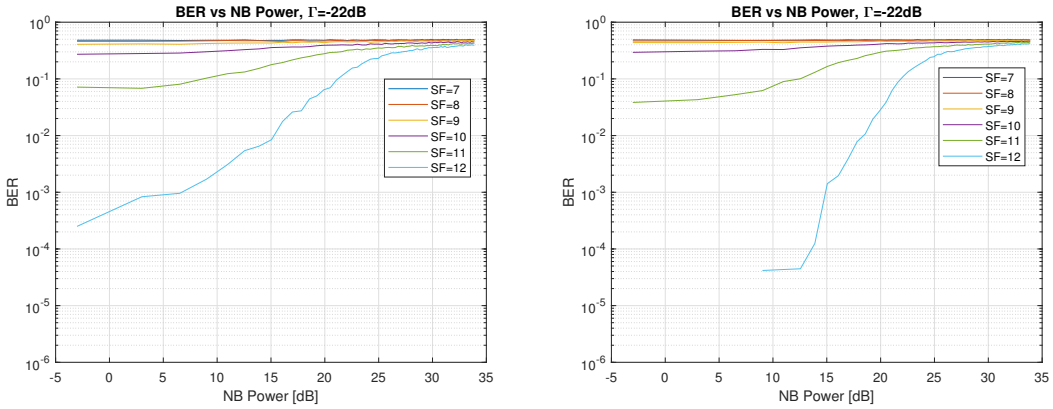


Figure 9.6: NB interference block diagram

9.3.2 Results

Here, the resulting BER curves with respect to the narrowband signal power are reported, varying SF, SNR and carrier frequency in multiple combinations.



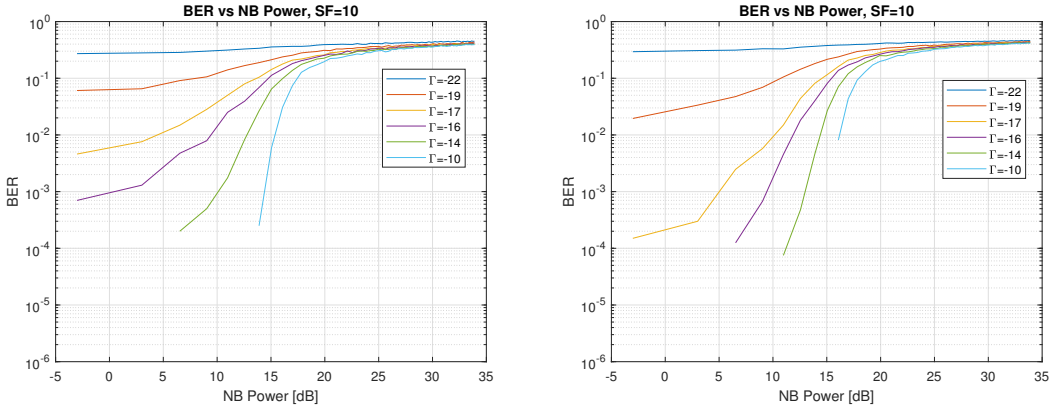
(a) *Uncoded BER vs NB Power* **(b)** *Coded BER vs NB Power*

Figure 9.7: BER vs NB Power with $\Gamma = -22dB$ and $f = 1kHz$

Figure 9.7a and Figure 9.7b represent the BER with the variation of SF, NB power and $SNR = -22dB$ for the uncoded and coded cases, respectively. The leftmost points of all the curves correspond to the BER value for that given SNR found in Figure 9.5, since the amplitude of the narrowband signal is 0.

In Figure 9.7a, the impact of the interference on the curve for $SF = 12$ starts to worsen the performance when the signal power is around $+6dB$, value for which the BER is higher than 0.001.

On the other side (Figure 9.7b), the coded one, the error rate is lower than 0.001 until a signal power equal to $+15dB$.



(a) *Uncoded BER vs NB Power* **(b)** *Coded BER vs NB Power*

Figure 9.8: BER vs NB Power with $SF = 10$, $f = 1kHz$ and varying Γ

Figure 9.8a and Figure 9.8b show the resulting BER curves with varying narrowband power for $SF = 10$ and different values of SNR (Γ).

It can be noticed that according to the value of SNR, the impact of the interference makes the BERs increase faster or slower up to the highest value.

Moreover, for signal power values higher than $+20dB$, all the curves converge to the constant value of $BER \simeq 0.5$.

Also, the amplitude of the narrowband signal has a negligible impact when the error is already consistent due to the high level of SNR of the AWGN.

In general, even though the power of the interference is 10 times higher than the LoRa signal, the BER is still low, fact that proves the good robustness of LoRa modulation.

Finally, a further analysis aimed at studying the behavior of the error rate varying the carrier frequency of the narrowband signal, as shown in Figure 9.9.

In the LoRa signal bandwidth, some frequencies were chosen close to the center of the band, such as $1Hz$, others closer to the extremes, i.e. $\pm 6kHz$, with others assuming intermediate values.

The first observation that can be made is that for a narrowband signal power greater than $+15dB$, the performance for small absolute values of f is better than intermediate and higher ones.

On the other side, before $+15dB$, the trend is the opposite, indeed, the closer to the middle point of the band, the higher the BER.

Moreover, the case for the exact $f = 0Hz$ was taken into account but not shown here, due to the inconsistent results of each simulation run.

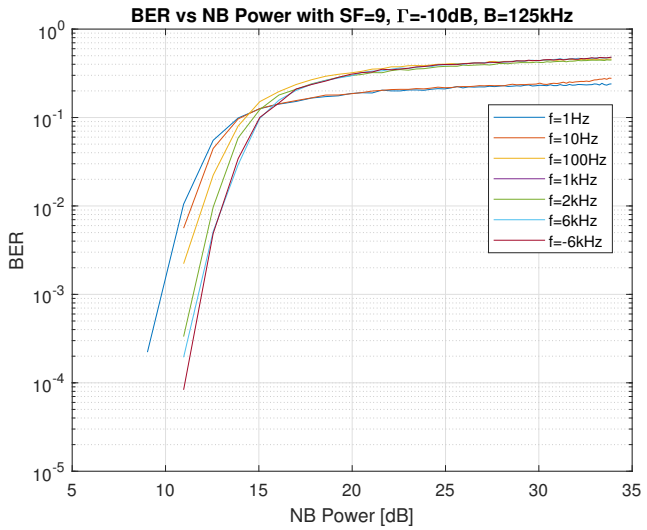


Figure 9.9: BER vs NB Power with $SF = 9$, $\Gamma = -10dB$, $B = 125kHz$ and varying f

Chapter 10

Conclusions

10.1 Conclusions

The complete knowledge of the LoRa PHY layer, which is a proprietary protocol, would imply a lot of advantages, such as the possibility of realizing software simulations to analyse the effects of different types of interference without using multiple hardware transceivers.

Consequently, a deep analysis of the literature was necessary to obtain a sufficient level of information. Indeed, several attempts of decoding the LoRa modulation have been taken into account.

The first goal of this second part was the realization of the LoRa PHY layer by means of the simulation tools provided by LabVIEW.

Before the implementation, some assumptions have been adopted together with some simplifications, such as the preamble detection and whitening that have not been employed and the hypothesis of perfect synchronization of LoRa signals.

This part can be considered as a contribution in terms of available LabVIEW codes related to LoRa modulation, demodulation and symbol baseband generation. Indeed, they might be used by whoever is interested in LoRa PHY layer.

The second goal, instead, concerns the robustness analysis of LoRa modulation, whose performance can be assessed by means of the BER.

The study of the state-of-the-art covered several works which are related to the analysis of both CSS and LoRa modulation. All of them confirmed the good quality of this technique, but none focussed the attention on the case of interference due to a narrowband signal.

The LabVIEW implementation was adapted to compute the BER for different values of SNR in AWGN channels. The results have been validated by means of the comparison with the work of Ferré and Giremus (2018) considered in the literature review phase, to prove the reliability of the simulation.

Then, the performance gain in terms of BER due to the adopted channel coding and interleaver was shown.

Finally, the additional interference source coming from a narrowband signal has been considered. The analysis covered different parameter settings and demonstrated that the LoRa modulation is able to provide a great robustness against narrowband interference. Indeed, considering the case of spreading factor equal to 10 and a reasonable value for the SNR of the AWGN, an interfering noise characterized by a power which is slightly more than 10 times higher than the LoRa signal does not worsen in a significant manner the bit error rate value.

10.2 Future works

The simulator can be further improved, by means of the addition of the whitening and removing the assumption of perfect synchronization.

Also, the impact of the Gray Code is interesting to be analysed.

Moreover, the actual program exploits the use of only Hamming(7,4) as error correction technique, which might be extended to the other types of FEC, such as Hamming(8,4).

The ideal case would be to achieve the highest level of compliance with the real hardware transceivers.

Bibliography

[1] URL: https://www.gartner.com/en/information-technology/glossary/internet-of-things (cit. on p. 1).

[2] URL: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf (cit. on p. 1).

[3] N. Naik. «LPWAN Technologies for IoT Systems: Choice Between Ultra Narrow Band and Spread Spectrum». In: *2018 IEEE International Systems Engineering Symposium (ISSE)*. 2018, pp. 1–8. DOI: 10.1109/SysEng.2018.8544414 (cit. on p. 1).

[4] URL: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf (cit. on pp. 2, 9).

[5] URL: https://lora-alliance.org/sites/default/files/2018-04/lorawan-technical-intro.pdf (cit. on p. 2).

[6] URL: https://lora-alliance.org/about-lorawan (cit. on p. 5).

[7] URL: http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf (cit. on pp. 5, 37).

[8] URL: https://www.cambridgewireless.co.uk/media/uploads/resources/Connected%20Devices%20Group/21.04.15/ConnectedDevices-21.04.15-Semtech-Jeff_McKeown.pdf (cit. on p. 5).

[9] G. Ferré and A. Giremus. «LoRa Physical Layer Principle and Performance Analysis». In: *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2018, pp. 65–68. DOI: 10.1109/ICECS.2018.8617880 (cit. on pp. 6, 53).

[10] Mohamed Amine Ben Temim, Guillaume Ferré, and Romain Tajan. «Analysis of the Coexistence of Ultra Narrow Band and Spread Spectrum Technologies in ISM Bands». In: (2019) (cit. on p. 6).

[11] URL: https://www.rs-online.com/designspark/rel-assets/ds-assets/uploads/knowledge-items/application-notes-for-the-internet-of-things/LoRa%20Design%20Guide.pdf (cit. on p. 7).

[12] Jansen Liando, Amalinda Gamage, Agustinus Tengourtius, and Mo Li. «Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study». In: *ACM Transactions on Sensor Networks* 15 (Feb. 2019), pp. 1–35. DOI: 10.1145/3293534 (cit. on p. 7).

[13] URL: https://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf (cit. on p. 7).

[14] URL: https://patents.google.com/patent/EP2763321A1/en (cit. on pp. 7, 37).

[15] Matthew Knight and Balint Seeber. «Decoding LoRa: Realizing a Modern LPWAN with SDR». In: 2016 (cit. on pp. 7, 37).

[16] Reza Ghanaatian, Orion Afisiadis, Matthieu Cotting, and Andreas Burg. *LoRa Digital Receiver Analysis and Implementation*. Nov. 2018 (cit. on p. 8).

[17] Peng Xu and Yang Rufei. *Déploiement d'un réseau de capteurs sur le site de l'ENSIL*. Report. Université de Limoges. 2019 (cit. on p. 13).

[18] URL: https://thingspeak.com (cit. on p. 13).

[19] *AM2320 Product Manual*. Datasheet. Aosong (cit. on p. 15).

[20] URL: http://modtronix.com/nz32-sc151 (cit. on p. 15).

[21] *SX1276/77/78/79*. Datasheet. Rev. 6. Semtech Corporation. Jan. 2019 (cit. on pp. 15, 21).

[22] URL: https://www.raspberrypi.org (cit. on p. 16).

[23] URL: http://www.openstm32.org (cit. on p. 16).

[24] URL: https://www.ac6-tools.com (cit. on p. 16).

[25] URL: http://wiki.modtronix.com/doku.php?id=products:nz-stm32:nz32-sc151#programming_and_debugging (cit. on p. 16).

[26] URL: https://github.com/modtronix-com/devkit_sx1276 (cit. on p. 17).

[27] Pieter Robyns, Peter Quax, Wim Lamotte, and William Thenaers. «A Multi-Channel Software Decoder for the LoRa Modulation Scheme». In: Jan. 2018, pp. 41–51. DOI: 10.5220/0006668400410051 (cit. on pp. 37, 39).

[28] URL: https://myriadrf.org/news/lora-modem-limesdr/ (cit. on p. 37).

[29] T. Elshabrawy and J. Robert. «Closed-Form Approximation of LoRa Modulation BER Performance». In: *IEEE Communications Letters* 22.9 (2018), pp. 1778–1781. ISSN: 2373-7891. DOI: 10.1109/LCOMM.2018.2849718 (cit. on p. 38).

[30] B. Reynders, W. Meert, and S. Pollin. «Range and coexistence analysis of long range unlicensed communication». In: *2016 23rd International Conference on Telecommunications (ICT)*. 2016, pp. 1–6. DOI: 10.1109/ICT.2016.7500415 (cit. on p. 38).

[31] A. Marquet, N. Montavont, and G. Z. Papadopoulos. «Investigating Theoretical Performance and Demodulation Techniques for LoRa». In: *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 2019, pp. 1–6. DOI: 10.1109/WoWMoM.2019.8793014 (cit. on p. 38).

[32] B. Reynders and S. Pollin. «Chirp spread spectrum as a modulation technique for long range communication». In: *2016 Symposium on Communications and Vehicular Technologies (SCVT)*. 2016, pp. 1–5. DOI: 10.1109/SCVT.2016.7797659 (cit. on p. 38).

[33] Orion Afisiadis, Andreas Burg, and Alexios Balatsoukas-Stimming. *Coded LoRa Frame Error Rate Analysis*. Nov. 2019 (cit. on p. 38).

[34] G. Baruffa, L. Rugini, V. Mecarelli, L. Germani, and F. Frescura. «Coded LoRa Performance in Wireless Channels». In: *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2019, pp. 1–6. DOI: 10.1109/PIMRC.2019.8904298 (cit. on pp. 38, 39, 51).

[35] T. Elshabrawy and J. Robert. «Analysis of BER and Coverage Performance of LoRa Modulation under Same Spreading Factor Interference». In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2018, pp. 1–6. DOI: 10.1109/PIMRC.2018.8581011 (cit. on p. 38).

[36] B. Dunlop, H. H. Nguyen, R. Barton, and J. Henry. «Interference Analysis for LoRa Chirp Spread Spectrum Signals». In: *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*. 2019, pp. 1–5. DOI: 10.1109/CCECE.2019.8861956 (cit. on p. 38).

[37] URL: https://en.wikipedia.org/w/index.php?curid=54205915 (cit. on p. 41).

[38] URL: https://www.ni.com/labview (cit. on p. 41).

[39] URL: https://en.wikipedia.org/wiki/Hamming(7,4) (cit. on p. 45).

[40] URL: https://forums.ni.com/t5/Example-Programs/Rotate-2D-Array-Using-LabVIEW/ta-p/3506774?profile.language=en) (cit. on p. 45).

[41] URL: https://forums.ni.com/t5/Example-Programs/Converting-Gray-Codes-to-Their-Corresponding-Standard-Binary/ta-p/3527189?profile.language=en (cit. on p. 46).