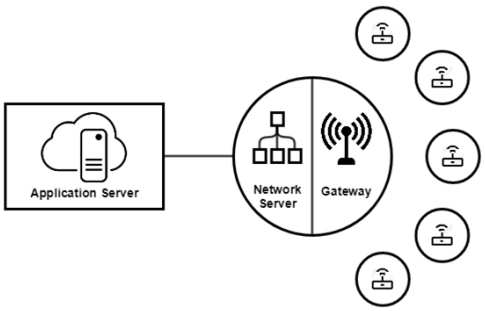
TÉCNICO LISBOA



Development of Low-cost LoRaWAN Gateway for Private Deployments

Nuno Gil Polónia Manita Nico

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. António Manuel Raminhos Cordeiro Grilo

Examination Committee

Chairperson: Prof. Horácio Cláudio De Campos Neto
Supervisor: Prof. António Manuel Raminhos Cordeiro Grilo
Member of the Committee: Prof. Renato Jorge Caleira Nunes

November 2017

To my Family

Acknowledgments

In first place, I would like to thank my thesis supervisor, Prof. António Grilo It has been an amazing opportunity to work under his supervision. Without his guidance and availability this work would have not been possible.

For NEECIST and the amazing people I worked with, a big thank you for what they taught me and for the support. My academic journey would not have been the same without them.

I also would like to thank my family and friends for supporting me during the last years.

Resumo

O principal objetivo da tese apresentada nesta dissertação é a implementação de uma *gateway* LoRaWAN de baixo custo. Para implementar esta *gateway* foram utilizados módulos LoRa capazes de receber em apenas uma frequência. Com esta limitação, surge a necessidade de resolver dois problemas expostos em seguida.

O primeiro problema é a vulnerabilidade a *jamming*, um *jammer* poderia causar interferência propositada na frequência utilizada pela *gateway*, comprometendo o correcto funcionamento do sistema. Para resolver este problema, foi desenvolvido um sistema de *frequency-hopping*. Este mecanismo faz com que as transmissões sucessivas saltem de frequência em frequência, de forma a limitar as faixas de frequências afectadas por um atacante que utilize um *jammer*. Para colmatar esta vulnerabilidade, a gateway foi implementada com dois módulos LoRa, cada um capaz de operar numa frequência distinta.

O segundo prende-se com a garantia de que os pacotes prioritários são corretamente recebidos pela *gateway*. Este problema foi resolvido utilizando um sistema de reserva de *time-slots* livres de contenção. Ou seja, um determinado nó pode reservar um determinado período de tempo para enviar os seus pacotes prioritários.

O desempenho do sistema desenvolvido foi testado experimentalmente. O teste de alcance das comunicações permite concluir que se permitem distâncias de pelo menos 4.9 quilómetros, entre os dispositivos e a *gateway*. Foi testada a aleatoriedade das sequencias de saltos na frequência, que permitiu concluir um nivel de aleatoriedade suficiente, tendo em conta as necessidades desta aplicação. Foram ainda testadas as extensões feitas ao LoRaWAN (*frequency-hopping* e reserva de *time-slots*). Foi possível concluir que os mecanismos desenvolvidos são eficazes.

Palavras-chave: IoT, LPWAN, LoRaWAN, Gateway

Abstract

The main problem addressed in this thesis is the development of a Low-cost LoRaWAN gateway. To develop this gateway LoRa single-frequency modules have been used. Using modules that are only capable of receiving in one frequency simultaneously creates two main issues, presented below.

The first is the vulnerability to jammers. A jammer can cause intentional interference and affect the proper operation of the system. To solve this issue, a frequency-hopping mechanism was implemented. Using this mechanism, the communications are constantly changing the frequency to avoid jamming. To solve this issue, were also implemented on the gateway two LoRa modules, each one capable of operating in different frequencies.

The second issue was to assure the correct sending of priority messages. This was done by implementing a time-slot reservation mechanism. Using this mechanism, a device can reserve a period in time where only it can send messages.

The performance of the developed system was tested experimentally. The gateway range was tested and transmissions between an end-device and the gateway with up to 4.9 kilometers were achieved. The randomness of the generated sequences was tested, and the results show that it has enough randomness for this application. Was also tested the implemented extensions to LoRaWAN. Concluding that the developed mechanisms are effective.

Keywords: IoT, LPWAN LoRaWAN, Gateway

Contents

List of Tables

List of Figures

List of Acronyms

3G	Third generation mobile communication
3GPP	3rd Generation Partnership Project
ADR	Adaptive Data Rate
AES	Advanced Encryption Standard
AMR	Advanced Meter Reading
AppSKey	Application Session Key
BER	Bit Error Rate
ChIndex	Channel Index
CS	Chip Select
CSS	Chirp Spread Spectrum
DevAddr	Device Address
DrRange	Data Range
FEC	Forward Error Correction
FSPL	Free Space Path Loss
Freq	frequency
FSK	Frequency Shift Keying
IC	Integrated Circuit
IoT	Internet of Things
LPWAN	Low Power Wide Area Network
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAC	Media Access Control
MaxDr	Maximum Data Range
MIC	Message Integrity Code
MinDr	Minimum Data Range
MISO	Multiple Input Single Output
MOSI	Multiple Output Single Input
Mtype	Message Type
NB-IoT	NarrowBand-IoT
NwkSKey	Network Session Key
PCB	Printed Circuit Board
RFU	Reserved For Future Use
S2AAS	Sensing as a service
SCK	Serial Clock
SPI	Serial Peripheral Interface
SS	Slave Select
TDMA	Time-Division Multiple Access
TimeSlotAss	Assigned time-slot
ToA	Time on Air
UNB	Ultra Narrow Band
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

Chapter 1

Introduction

Nowadays, the interest on the Internet of Things (IoT) has been increasing. IoT is a concept whereupon all the devices are connected to the Internet, these devices can be a smart sensor, a fridge that monitors the food stock, or even blinds that open at wake up time. IoT is creating also an interest on Machine-to-Machine (M2M) communications, that is machines communicating with each other without any human mediation - for example a fridge connected to the internet can order products from the supermarket when needed.

Typically an IoT device sends the data through the Internet to the cloud, where the data is stored and processed. By collecting all these data and processing it, it is possible to make predictions, for example, it is possible to make a relation between the temperature in a certain city and the clothes' buying trends. Obviously this kind of prediction has a commercial value. The service that allows buying and selling sensing data is called sensing as a service (S2AAS). Figure 1.1 represents a generic IoT architecture.

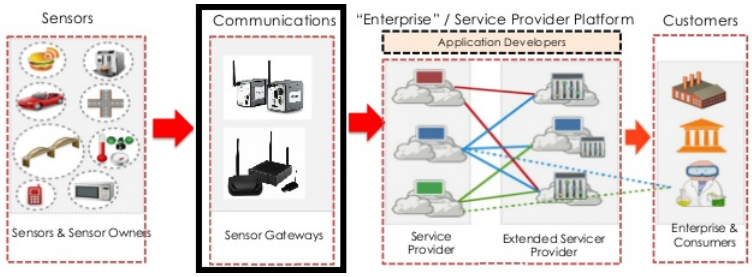


Figure 1.1: IoT architecture (extracted and adapted from [6]).

In order to support IoT communications, several technologies have been developed, these technologies are Wireless Personal Area Network (WPAN) such as Bluetooth and ZigBee, Wireless Local Area Network (WLAN) for example Wi-Fi and Low Power Wide Area Network (LPWAN). LPWAN are low power consumption technologies that provide long range communications, typically several kilometers. This long range capability is what distinguishes LPWAN from WPAN and WLAN, LPWAN technologies are also distinguished from WLAN by their much lower power consumption. The focus of this work will be a specific LPWAN technology, LoRa, together with the LoRaWAN communication protocols and functions.

1.1 Motivation

The previously mentioned factors create a high demand for technologies that support a steady communication between thousands of sensors and the cloud. This communication does not need to be very fast, bulky, or very frequent. Typically, a sensor like this sends just a few messages with few bytes a day. As already mentioned, LPWAN are communication technologies that allow a long range communication with low power consumption, and are the best option in scenarios where there are multiple battery powered nano-sensors.

LoRa is an open-source LPWAN technology and operates in free bands, which facilitate the development of IoT applications. On the other hand, the use of free bands means that it will likely compete with other communication systems, which may interfere and degrade its performance.

LoRaWAN is a Media Access Control (MAC) protocol for high capacity networks. LoRaWAN protocols are defined by the LoRa Alliance and formalized in the LoRaWAN Specification. Figure 1.2 shows the LoRa alliance logo.

Figure 1.2: LoRa alliance logo (retrived from [2]).

LoRaWAN defines several channels in the frequency domain. Each channel is independent and communications in one channel do not interfere with other channels.

To receive the data from the sensors, typically are used commercial gateways that cost several hundred euros. The main goal of this project is to develop a LoRa low-cost Gateway. The price reduction will be achieved by reducing the implementation complexity and downgrading some capabilities.

The developed gateway is only able to receive data on part of the channels available to LoRaWAN, unlike commercial gateways that can listen to all the channels. In order to mitigate channel interference and jamming, the developed gateway have a frequency-hopping system. By switching between channels it is possible to avoid channels with noise. This frequency-hopping mechanism require extensions to the LoRaWAN specification, making the developed solution in some way proprietary, which is a viable option when developing a private network. Even so, the developed solution comply with most of the LoRaWAN specifications.

LoRaWAN specifications define that a gateway must be able to receive on all the channels simultaneously. As stated before, the gateway to be developed is only able to receive on part of the channels. Although most of the LoRaWAN specifications will be fullfilled, the frequency-hopping extension must be implemented by the end-devices to communicate with the developed gateway.

As mentioned before, a large increase in the use of IoT and LPWAN technologies is expected to happen, which will obviously increase the use of free frequency bands. This creates the issue of how to ensure that priority data is sent correctly. To address this issue, the deployed system must be scalable to work with many devices, by making an extension to LoRaWAN. This extension reserve time-slots when requested by end-devices, ensuring that collisions will not affect the correct transmission of priority data.

This mechanism is, in fact, a Time-Division Multiple Access (TDMA) mechanism.

Note that one of the possible applications of IoT technologies is the farm monitoring across vast areas. This monitoring can include temperature, humidity, soil moisture, etc. Lots of end-devices, connected with sensors, can be spread in a vast area and send the measurements to a gateway. This is important, for example, to schedule the irrigation in the best way possible, according with the measurements done. The developed gateway is ideal for these scenarios where the network is private and the number of sensors is high. In rural environments ranges can reach up to 15km. Therefore, a vast area can be monitored with the proposed low-cost gateway.

1.2 Objectives

The LoRa gateway was developed taking into account the resilience against jamming, the power consumption, constraints imposed by the communications regulators, the appropriate frequency bands, data rate and range. The developed gateway is a low-cost gateway, therefore has only the ability to receive communications on part of the channels. This is a limitation to the gateway's capacity and makes it vulnerable to an eventual jammer. To solve these limitations the developed gateway has the following features:

1. Receive data through multiple channels simultaneously, gaining the ability to receive data from multiple sensors at the same time.
2. Support frequency-hopping to mitigate interference problems.
3. A time-slot reservation scheme, based on centralized scheduling algorithms to reduce the number of collisions.
4. Network server that sends the data to the cloud and stores it.
5. These features are LoRaWAN compatible, whenever possible.

To store the data received from the gateway, network and application servers have also been developed.

1.3 Thesis Outline

The following chapters are structured as follows:

1. Chapter 2 presents the relevant state of the art.
2. Chapter 3 presents a description of the work to be developed.
3. Chapter 4 presents the results of the tests made regarding this thesis
4. Chapter 5 concludes this report

Chapter 2

Background And Related Work

In this chapter, several LPWAN technologies will be introduced, such as Sigfox and LTE LPWAN technologies. Starting by introducing some general advantages and disadvantages of each technology. Then it will explain the LoRa protocol, and LoRaWAN, a network specification designed to maximize power efficiency and capacity. For this study, some technologies were selected, since they are considered as competitors for LoRa. Mobile communication technologies, such as third generation mobile communication (3G), will not be addressed here since they are not LPWAN technologies and are not suitable for use in the same scenarios.

2.1 LPWAN Technologies

LPWAN are communication technologies that allow a long range communication with low power consumption. These technologies are essential to IoT applications because they need a robust communication link to send their data, and the communication cannot spend much power as IoT devices are typically battery powered.

The frequency band is an important characteristic of any wireless technology, namely LPWAN technologies. There are frequencies that are free for anyone to use in any application such as the 863-870MHz for Europe, frequencies that are also free but reserved to a certain application and even frequencies that need to be bought from the country regulator to be used such as the frequencies used by mobile phone systems.

On the other hand, there are technologies which are open source and can be used by everyone without licensing, such as LoRaWAN and NarrowBand-IoT (NB-IoT). Sigfox is an example of a proprietary technology, therefore can only be used with licensing.

Some technologies must be managed by operators, e.g., Sigfox and NB-IoT, while LoRaWAN is suitable both for private and for operator managed solutions. When the LPWAN is managed by an operator, this operator charges fees for the communication service but the sensor creator does not need to worry about the gateways, as they are already implemented by the operator. Using an operator network comes with the advantage of having a national, international or even global coverage.

Where there are no LoRaWAN operators – as is the case in Portugal, to develop a LoRa system, there is the need to install private gateways that will receive the data from the sensors, these come with the advantage of not having to pay any fees.

Table 2.1 summarizes the mentioned characteristics about LPWAN technologies.

Table 2.1: Summary of the LPWAN technologies.

		Frequency Bands	
		Shared	Licensed
Technology	Open Source	LoRaWAN	LoRaWAN NB-IoT
	Proprietary	Sigfox	

Note that all LPWAN technologies operate at low bitrate. This is a limitation of these technologies, as they are not able to operate at higher speed. Nevertheless, this limitation is also a privacy guarantee. These technologies assure that, even if hacked, a device cannot transmit images or video.

This is useful, for example, for devices containing a camera, where images must be processed on the device and only the processed data is sent. A device like this is being tested to monitor room occupation in United Kingdom hospitals.

2.1.1 LTE LPWAN Technologies

Long Term Evolution (LTE) LPWAN are technologies developed by the 3rd Generation Partnership Project (3GPP) to use licensed bandwidths. As other IoT communication standards, it focus on low battery consumption and large number of connected devices. Table 2.2 compares some of this technologies.

Table 2.2: Comparison between LTE LPWAN technologies (adapted from [19]).

	Cat-1	Cat-0	Cat-M1	NB-IoT Cat-NB1
Bandwidth	20 MHz	20 MHz	1.08 MHz	180 kHz
UL Bitrate	5 Mbps	1 Mbps	1 Mbps	20 kbps (single-tone) 250 kbps (multi-tone)
DL Bitrate	10 Mbps	1 Mbps	1 Mbps	250 kbps
Duplex mode	full-duplex	half or full	half or full	half-duplex
3GPP release	Release 8	Release 12	Release 13	Release 13

NB-IoT is the more recent technology and the one that fits better with IoT as it provides the largest coverage and low power consumption.

NB-IoT Operators

NB-IoT is being installed by several worldwide telecommunication operators to provide IoT communication services to their clients. Operators can use the already existent infrastructures to install NB-IoT, reducing the costs. For example, in Portugal the operator NOS provides communications services based

on NB-IoT for Advanced Meter Reading (AMR). Figure 2.1 shows a power meter developed by NOS, EDP and Huawei using NB-IoT.

Figure 2.1: Power meter developed by NOS, EDP and Huawei using NB-IoT (extracted from [22]).

2.1.2 Sigfox

Sigfox is a company that provides a communication service with a proprietary protocol. This company is the most well-known in this area. Sigfox protocol uses the free 868MHz band (Europe) with an Ultra Narrow Band (UNB) of just 100Hz, providing a maximum bitrate of 100 bits per second, the lowest of the technologies addressed in this report. Sigfox also limits the number of uplink messages sent per day to 140. By reducing the bandwidth to a UNB to 100Hz is possible to produce strong signals with low-power consumption. Note that the bandwidth is much lower than the used by NB-IoT or the LoRa. With a bandwidth of 100Hz is only possible to produce low bitrate communications. Table 2.3 shows some limitations imposed by Sigfox.

Table 2.3: Limitations imposed by Sigfox (extracted from [13]).

Band used (for EU)	868 MHz
Max Uplink messages per day	140
Max messages size	12 Bytes
Bitrate	100 bps
Max Downlink messages per day	4

Sigfox has a worldwide deployed network that was mainly developed to receive messages from IoT devices, typically just a few per day, as seen before. Despite having a worldwide deployed network, Sigfox network coverage is not always assured. There is also the possibility to send data from Sigfox network to the IoT devices (downlink), but there are significant limitations and this mode is highly discouraged by Sigfox. Downlink messages have these limitations due to duty cycle regulations. A single device, in this case the gateway, can only be transmitting one per cent of the time. Therefore, to share the available time between the devices, each one must have strong downlink restrictions.

Although Sigfox was mainly developed to be used by big enterprises, it is possible for developers to use the Sigfox network. To use the Sigfox network it is needed a Sigfox certified communication module and a paid license. Figure 2.2 shows an example of a Sigfox communication module.

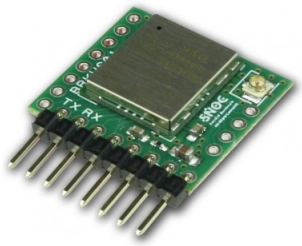


Figure 2.2: Example of a Sigfox communication module (extracted from [13]).

2.2 LoRa

LoRa stands for Long Range and is a modulation protocol designed to provide long range communication, using a variation of Chirp Spread Spectrum (CSS) with integrated Forward Error Correction (FEC). LoRa is an open source protocol, in other words anyone can develop a LoRa network without a license. The only must is to comply with the regulator's rules.

CSS is a spread spectrum technique that uses the entire channel bandwidth to broadcast the signal. By spreading the signal in the time domain, it is possible to reduce the Bit Error Rate (BER) and achieve long distance communications. LoRa can demodulate signals 19.5 dB below the noise floor while most Frequency Shift Keying (FSK) systems need a signal power of 8-10 dB above the noise floor. LoRa protocol defines 7 spreading factor levels (from 6 to 12). To increase the robustness of the connection, a higher spreading factor must be selected. By increasing the spreading factor, the time needed to transmit the data will be longer therefore increasing the power consumption. It is important to balance what are the needs of each application to select the spreading factor. The gateway developed has the ability to receive data in different spreading factors. This allows the end device to choose the spreading factor that fits better.

FEC is a technique used for controlling errors in data transmission over unreliable or noisy communication channels. To enhance data reliability FEC introduces some redundant data, using an error-correcting code. The redundancy introduced is expressed by the code rate. For example, a transmission with a coding rate of 4/5 means that one bit of redundancy is added to each block of 4 bits of useful information.

LoRa Operators

Orange, a French multinational telecommunications corporation, is building a LoRaWAN network in French metropolitan areas. Networks like this are made up of several gateways who send the data to a network server, and then to an application server.

To use this kind of network the developer implements the end-devices and pays a fee to the operator to collect and store the data.

Other services can be used as LoRaWAN operators. They have some differences from the usual telecommunications operators and will be presented in Section 2.4.

2.2.1 LoRaWAN

A typical LoRaWAN network is composed by four components (Figure 2.3 represents a typical LoRaWAN network):

1. End-device : Collects the data using sensors and sends it to the gateway.
2. Gateway : Receives the data from the network server and sends this data through the air to the end-device and vice versa.
3. Network server : Manages several gateways assuring that no downlink duplicates are sent to the air and avoiding collisions.
4. Application server : Processes and stores the data. This component can send data to the end-devices through the network server/gateway.

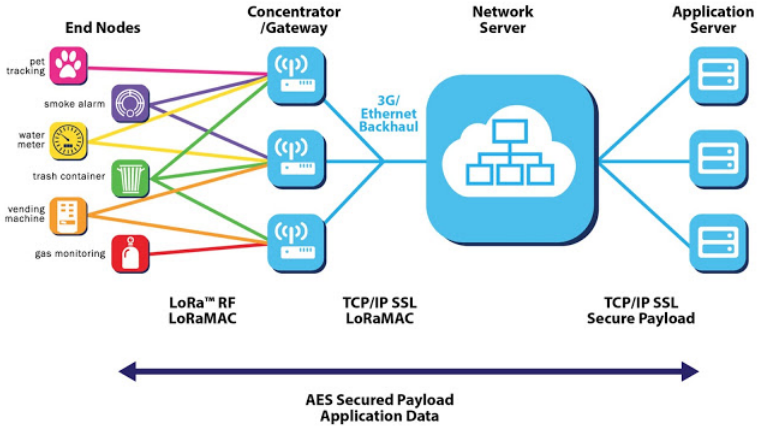


Figure 2.3: LoRaWAN typical network (extracted from [1]).

LoRaWAN Security Protocol

LoRaWAN defines protocols to implement security from the end-device to the application server. To crypt the messages the Advanced Encryption Standard (AES) 128 bits algorithm is used. AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

To assure message security from the end-device to the application server, it is encrypted twice using different keys as described below (Figure 2.4 shows the LoRaWAN security protocol):

- Application Session Key (AppSKey) is used to crypt the data field. Only the end-device and the application server know this key.
- Network Session Key (NwkSKey) is used to crypt the result of the previous encryption together with the routing data, i.e. the DevAddr and other control fields. Only the end-device and the gateway know this code.

The network server processes the frame, check that the routing data matches the end-device address and that the Message Integrity Code (MIC) is valid before forwarding it to the application server. MIC is

used to authenticate the message, in other words, to confirm that the message came from the stated sender.

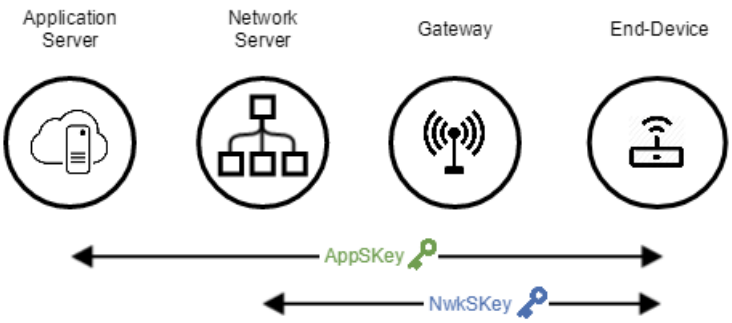


Figure 2.4: LoRaWAN security protocol.

Figure 2.5 shows what data are secured by each of the two encrypting keys.

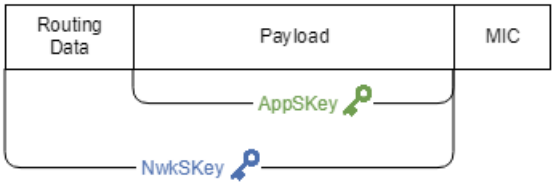


Figure 2.5: LoRaWAN message.

This security architecture assures that even if an attacker gets a message on the air, this attacker has no information about neither the data or the control fields. After the message is received by the gateway, only the control data is decrypted. This control data is used get the message to its destination.

The data is only decrypted at the application Server using the AppSKey. The security mechanisms of this architecture assures the authenticity, integrity and confidentiality of the message.

LoRaWAN Device Classes

One of the LoRaWAN main goals is to optimize power consumption by putting the end-devices into sleep mode in which they are not able to receive data. This means that end-devices need to have specific time windows to receive data. By doing this, the end-devices maximize the time in sleep mode. Therefore, the power consumption will be much lower. There are three classes of LoRaWAN devices, these classes define when a device should be listening to incoming messages (downlink).

1. Class A – Class A end-devices allow a bi-directional communication where the end-device uplink time-slot is followed by two downlink short time-slots. Using two time-slots instead of one increases the chance of success. This class provides the lowest power consumption and it's the best choice when the end-device doesn't need to be always able to receive data. Figure 2.6 shows the class A working mode.

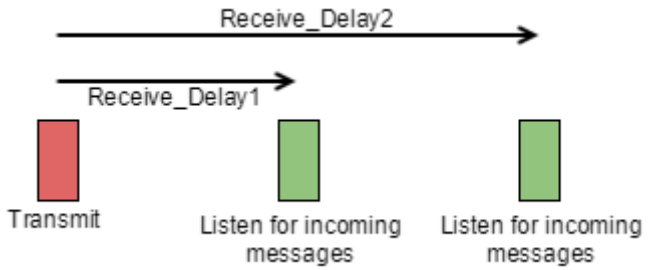


Figure 2.6: Class A working mode.

2. Class B – In addition to class A downlink time-slots, Class B end-device devices also open time-slots for downlink at a scheduled time. This allows the gateway to know when the end-devices is listening. Figure 2.7 shows the class B working mode.

Figure 2.7: Class B working mode.

3. Class C – Class C devices are always listening to incoming data, except when transmitting. Devices using this class will draw much power, for this reason it can be used only in situations where power consumption is not a problem, gateways must use this class. Figure 2.8 shows the class C working mode.

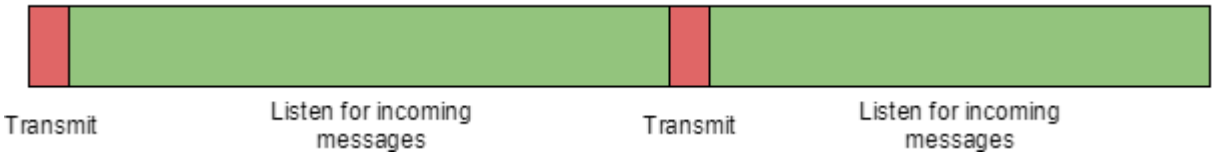


Figure 2.8: Class C working mode.

LoRaWAN class A devices use pure ALOHA access for uplink, this is, whenever there is a message to send this is sent immediately, which can cause collisions. One of the extensions to LoRaWAN done along this dissertation will schedule priority time-slots each end-device can use to send data, solving the collision issue for priority traffic.

LoRaWAN RF Protocols

As mentioned before, LoRaWAN is a MAC protocol for a high capacity star network and it is implemented on top of LoRa or FSK modulation.

LoRaWAN was optimized to work with very low power consumption sensors. This way the protocol can be implemented on battery operated sensors, operating for years without maintenance.

The frequencies, bit rates, number of channels for LoRa change from country to country. The main specifications are for Europe (EU) and for the United States (US), which are listed in Table 2.4.

Table 2.4: Europe and North America specifications (extracted from [1]).

	Europe	North America	China	Korea	Japan	India
Frequency Band	876-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz
Channels	10	64+8+8	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee
Channel BW Up	125/250 kHz	125/500 kHz				
Channel BW Dn	125 kHz	500 kHz				
TX Power Up	+14 dBm	+20dBm typ (+30dBm allowed)				
TX Power Dn	+14 dBm	+27 dBm				
SF Up	7-12	7-10				
Data rate	250bps - 50kbps	980bps - 21.9kbps				
Link Budget Up	155dB	154dB				
Link Budget Dn	155dB	157dB				

The objective is not to provide high rates, but rather to work with sensors that need to send only a few bytes per hour.

LoRaWAN can use channels with a bandwidth of either 125 KHz, 250 or 500 KHz, depending on the region. For example, in Europe only 125 KHz and 250 KHz can be used. Table 2.5 shows the spreading factors, bandwidths and indicative physical bit rate defined for EU.

Table 2.5: Bandwidths, spreading factors and indicative physical bit rates defined by LoRaWAN for EU (adapted from [17]).

DataRate	Configuration	Indicative physical bit rate (bits/s)
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa:SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

Table 2.6 shows the TX power defined by LoRaWAN for EU. Note that the maximum TX power +20 dBm increase the power consumption and is not supported by all LoRa. As mentioned before, maximum TX power is greater for other regions. For example, in North America +30 dBm power is allowed.

Table 2.6: TX powers allowed for EU (adapted from [17]).

TXPower	Configuration
0	20 dBm (if supported)
1	14 dBm
2	11 dBm
3	8 dBm
4	5 dBm
5	2 dBm
6.15	RFU

Adaptive Data Rate (ADR) is a LoRaWAN mechanism developed to optimize data rates, wait time and power consumption. This mechanism selects which spreading factor and bandwidths to use for

each end-device based on the collected connection metrics. For example, if the end-device is close to the gateway the spreading factor can be lower, this reduces the time needed to transmit. Therefore, the power consumption is lower.

2.3 LoRa Gateway

A LoRa gateway is a device that is able to receive LoRa signals and to send the data to a network server. A fully compliant LoRaWAN gateway should be able to demodulate multiple signals on multiple channels simultaneously.

There are two kinds of gateways. The first type that comes ready to be used out of the shelf, with all configurations already done and the ability to listen to all the LoRa channels, by using a SX1301 baseband concentrator. The second type are the low-cost gateways that are only a Printed Circuit Board (PCB) and need to be assembled and fully configured. Typically these gateways use a single channel module, therefore can only receive on one channel at a time. Receiving communications on just one channel limits the number of devices that can communicate with it. Figure 2.9 presents a LoRa gateway fully assembled and configured by Multitech. Figure 2.11, which is represented in Subsection 2.3.1, shows a RFM95 LoRa PCB.

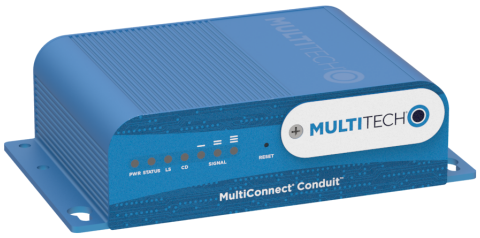


Figure 2.9: Multitech gateway (extracted from [9]).

The goal of this thesis is to develop a low-cost gateway with multi-channel receiving ability, by using several single frequency modules. Frequency-hopping will also be implemented to improve security against channel jamming. The developed solution will be compared with an already existing low-cost gateway [7]. This solution can only listen to a single frequency and does not implement frequency-hopping. Table 2.7 represents a comparison between the gateway intended to be develop, a low-cost gateway, a Raspberry Pi with a concentrator gateway and a professional out of the shelf gateway (Multitech gateway). These two last gateways differ in the software: the Multitech gateway comes with a professional easy to use software. This gateway offers also the ability to add additional modules, for example a 3G module.

Table 2.7: Comparison between the gateway Intended to develop, a Low-cost gateway, a Raspberry Pi with a concentrator gateway and a professional out of the shelf gateway (multitech gateway).

	Proposed solution	Low-cost Raspberry Pi gateway	Raspberry Pi LinkLabs gateway Board	Multitech gateway
Simultaneous channels	multiple	1	8	8
Backhaul options	Ethernet	Ethernet	Ethernet	Ethernet + 3G (extra)
Frequency-Hopping	Yes	No	Yes	Yes
Price	60€	50€	250€	450 €

As seen before, the presented solution will have resilience against jamming, offers the potential to listen to several channels at the same time keeping the price much lower than the professional out of the shelf gateways.

Arduino LoRa Shields

Arduino is an open-source hardware and software ecosystem. The company offers a range of software tools, hardware platforms and documentation. This company has announced that will release both a LoRa end-device and a LoRa gateway. Nevertheless, at the moment this thesis was published there was no information about prices and the specifications of these shields. Figure 2.10 shows a prototype of the LoRa gateway announced by Arduino.

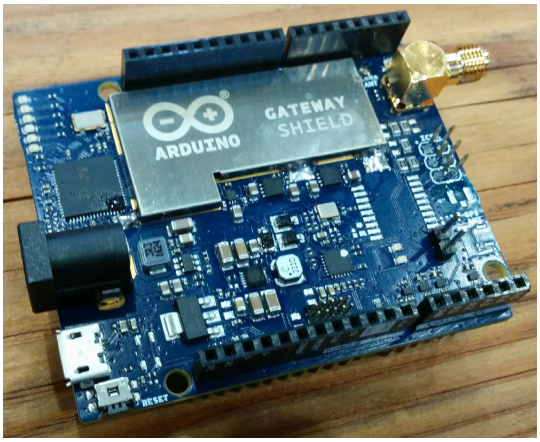


Figure 2.10: Arduino LoRa gateway (extracted from [18]).

This gateway will, probably, be a full compliant LoRaWAN gateway. This means that will have the capability to listen to all the channels defined by LoRaWAN. A frequency-hopping mechanism does not make sense in a device like this. Nevertheless, a time-slot reservation algorithm will be a gain to minimize lost of priority messages. Although is expected to be cheaper than the Raspberry Pi Link Labs gateway board option mentioned on Table 2.7 costing 250€, this solution will, probably, be more expensive than the solution proposed on this report, the low-cost LoRa gateway.

2.3.1 Equipment Selection

As stated before, multiple LoRa single channel modules will be needed, as well as a low-cost mini computer.

In this section, the equipment used to build the LoRa gateway, will be select. The options available will be presented and compared.

LoRa Module Selection

There are several available LoRa single channel modules on the market. Table 2.8 represents the comparison between LoRa modules. The prices of the presented modules are around 10 euros. Figures 2.11, 2.12 and 2.13 show the compared LoRa modules. Note that the SX1272 and SX1276 are Integrated Circuits (IC), Figure 2.12 shows a PCB with the welded IC on it.

The modules presented can operate within the frequencies allowed for EU, there are equivalent modules for other zones.

Table 2.8: Comparison between several LoRa modules.

	Semtech SX1272	Semtech SX1276	Microchip RN2483	HopeRF RFM95
Frequency	868 MHz	868/ 434 MHz	868/ 434 MHz	868 MHz
Sensivity	-130 dBm	-148 dBm	-148 dBm	-148 dBm
Supply current	125 mA	120 mA	38.9 mA	120 mA
DIO pins	6	6	14	6
Maximum Tx Power	+20 dBm	+20 dBm	+14 dBm	+20 dBm

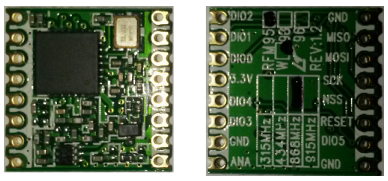


Figure 2.11: RFM95 module.

Figure 2.12: SX1272 IC mounted on a PCB (extracted from [7]).

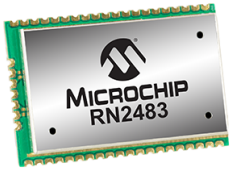


Figure 2.13: Microchip RN2483 module (extracted from [23]).

Microchip RN2483 modules have the maximum transmission power limited to +14 dBm. The feature that uses the maximum +20 dBm transmission power is named PA_BOOST. This feature increases the

transmission power, which is useful in applications where large range is needed. Nevertheless, using the modules with the maximum transmission power widely increase their power consumption.

The prices and features of single channel modules are similar. RFM95 was selected as it was the easiest to find at local stores.

Mini-Computer Selection

To build a low-cost LoRa gateway, a low-cost mini-computer is needed. This section will present the devices available on the market, and the most suitable will be selected. The selected device must have an Ethernet port to send the messages to the cloud. The selected device must also have IO pins compatible with LoRa single channel modules. These two requirements, excludes some of the available mini-computers on the market.

There are many other mini-computers available on the market, that fulfill the previous two requirements. There are cheaper mini-computers, but these do not have enough processing power to encrypt and decrypt the messages in real time. There are, also, more expensive solutions with more processing power, but these do not fulfill the low-cost requirement. Table 2.9 presents some of the mini-computers available on the market.

Table 2.9: Comparison between several mini-computers.

	Raspberry Pi 2	Raspberry Pi 3	Odroid C2	Banana Pi
Processor	ARM Cortex-A7 Quad-core 900MHz	Broadcom BCM2837 64bit Quad Core 1.2GHz	ARM Cortex-A53 Quad-core 1.5Ghz	ARM A83T Octa-Core
RAM memory	1 GB	1 GB	2 GB	2 GB
Ethernet port	Yes	Yes	Yes	Yes
IO pins	Yes	Yes	Yes	Yes
Price	40€	36€	60€	90€

Raspberry Pi 3 was found as the best match for this project. Raspberry Pi 3 offers both IO pins compatible with LoRa single channel modules and an Ethernet port. Raspberry Pi 3 fulfill the low-cost requirement needed to create a low-cost LoRa gateway. Raspberry Pi also offers the advantage of having lots of documentation available.

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. Figure 2.14 shows a Raspberry Pi 3.

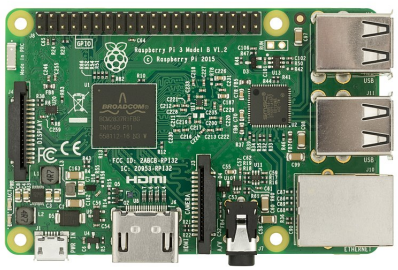


Figure 2.14: Raspberry Pi (extracted from [14]).

2.4 Network and Application Servers

This section presents the network and application servers existent on the market. The first to be presented and the most complete one is The Things Network (TTN). Then, other storage services will be presented.

2.4.1 The Things Network

TTN is a global, open, crowd-sourced IoT data network. TTN is a platform that works as a Network and application server at the same time. This platform are fully compliant with LoRaWAN protocols and regulations.

TTN allow anyone to add their own LoRaWAN gateways to the platform. Therefore, it is possible to develop an end-device and use these shared gateways to get the data and transmit it to the cloud. TTN implements the network features of decrypting incoming data and control multiple gateways, assuring that no downlink duplicates are sent to the air. Figure 2.15 shows the TTN overview.

TTN was developed to operate with LoRaWAN fully compliant gateways, as the proposed implementation has extensions to LoRaWAN it is not fully compatible with TTN, therefore was not used in this project.

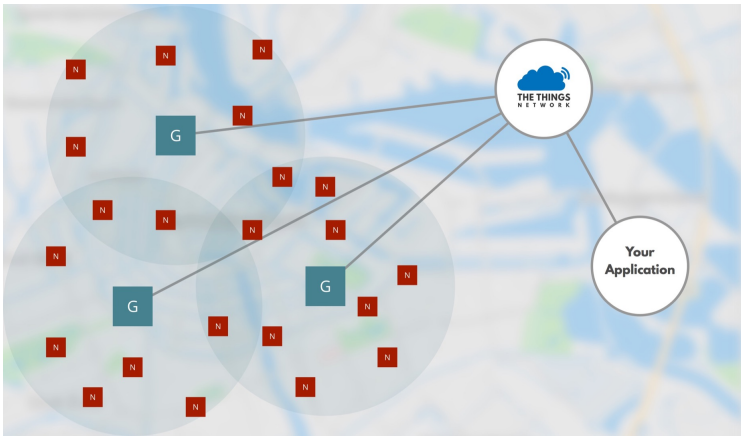


Figure 2.15: TTN overview (extracted from [17]).

TTN was developed to be very flexible, offering the option to enable and disable features. An user can add its own gateway or use the shared gateways installed by other users. Applications can be implemented on the platform or on other server.

As mentioned before, TTN platform works as a network server and application server. However, this platform allows to disable application server features. It is possible to disable decryption and storing features.

Doing this, the data will be sent encrypted to an online server. With this implementation the TTN is only responsible for managing the gateways and downlink messages. It will only operate as a network server.

TTN is, at least for now, a free service. This platform can be used as a LoRaWAN operator. Nevertheless, their service has no availability guarantee, the gateways are added by users and can go offline without warning.

2.4.2 Storage Services

There are also storage services like the ThingSpeak or GroveStream. These services are, in fact, database services. Services like these do not have the processing tools needed to decrypt the LoRaWAN messages, therefore they are not fully compliant LoRaWAN application servers. For this reason, these services have not been used to store the data.

2.5 Conclusion

In this chapter several LPWAN technologies were presented with special focus on LoRa. As seen before, Sigfox only allows up to 140 uplink and 4 downlink messages a day. This is a disadvantage when comparing with LoRa technology where the only must is to comply with the duty cycle limitation which allow a much greater number of messages per day.

LTE LPWAN technologies, such as NB-IoT, were created to use licensed frequency bands, in contrast with LoRa and SigFox, which use unlicensed bands. As mentioned before, Sigfox is a proprietary solution, which means that can only be used with a license. For these reasons LoRa was chosen, as it can be used both in private or public installations.

Table 2.10 makes a brief comparison between these technologies. Only the NB-IoT from the LTE LPWAN technologies was considered in this table.

Table 2.10: Comparison between several Communication protocols.

	Data Rate	Power Consumption	Range	Bandwidth	Free Band	Open Source
Sigfox	100 bps	Low ($\sim$20mA)	30-50km (rural) 3 - 10km (urban)	100Hz	Yes	No
LoRa	300 bps - 50kbps	Low ($\sim$20mA)	15 km (rural) 2- 5 (urban)	125-500KHz	Yes	Yes
NB-IoT	250 kbps	Medium ($\sim$100mA)	15 km	180KHz	No	Yes

Chapter 3

Development of the LoRa Gateway

This chapter presents the development of the LoRa gateway, including the gateway-driven frequency-hopping scheme, as well as the time-slot reservation algorithm that guarantees collision-free access for higher priority traffic.

3.1 System Architecture

The developed system has four components (Figure 3.1 represents these four components):

1. End-device : Collects the data and sends it to the gateway.
2. Gateway : Receives uplink and sends the downlink messages to the end-devices.
3. Network server : Manages several gateways assuring that no downlink duplicates are sent to the air and avoiding collisions.
4. Application server : Stores and processes the data.

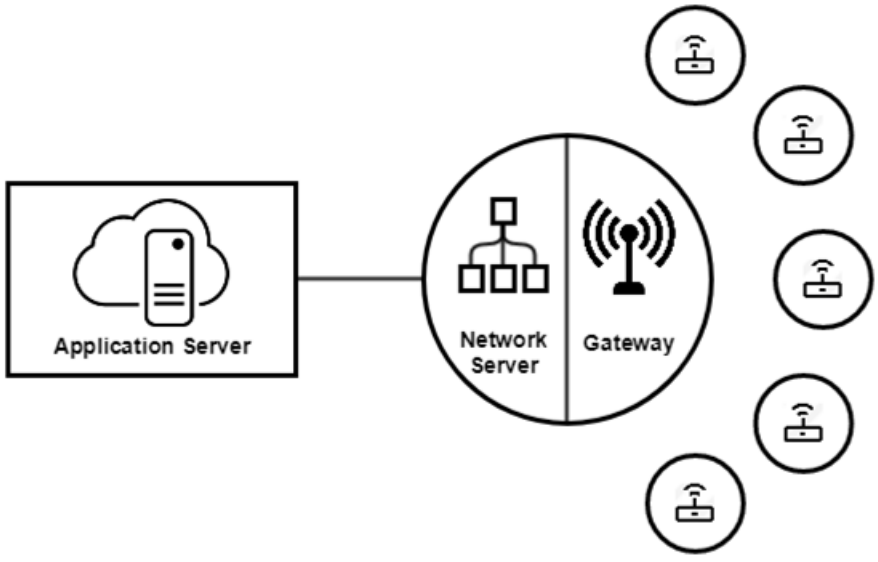


Figure 3.1: System architecture.

In a typical implementation there are several gateways managed by a network server. The network server also manages the developed frequency-hopping and time-slot reservation algorithms.

On the other hand, this thesis was focused on the frequency-hopping and time-slot reservation algorithms. For this reason, a single-gateway implementation was taken.

In this implementation network server also manages the frequency-hoping and time-slot reservation algorithms, but has no multi-gateway management function as in a multi-gateway scenario. Therefore, the network gateway was incorporated in the gateway, as expressed in Figure 3.1.

3.2 Channel Management Algorithms and Protocols

This section describes the existent LoRaWAN protocols and the frequency-hopping and time-slot reservation mechanisms implemented.

This thesis adds two important extensions to LoRaWAN, gateway-driven frequency-hopping and time-slot reservation. To add these extensions to LoRaWAN, the related control messages are sent with a message type (MType) different than the one used for data messages. MType is a 3 bits MAC header field defined by LoRaWAN and can take different values. Table 3.1 presents the values the MType can take according with LoRaWAN.

Table 3.1: MType message types according with LoRaWAN (extracted from [17]).

MType	Description
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU
111	Proprietary

Control messages regarding these extensions use MType 111, which refers to proprietary messages. Data messages use 010 for uplink and 011 for downlink. For data messages that require acknowledgement 100 is used for uplink and 101 for downlink.

Both time-slot reservation and frequency-hopping mechanisms make use of beacons. These beacons are control messages sent from the gateway to the end-devices, and contain information about the channel that should be used by the end-devices. The beacons are also used to synchronize the end-devices. The structure of these beacons is presented in Subsection 3.2.2.

3.2.1 Time-Slot Reservation Algorithms and Protocols

This subsection presents the algorithm and protocols implemented for the time-reservation mechanism. It describes the time-structure, the message exchange protocols, and the algorithms implemented for the time-slot reservation mechanism.

Structure of Time-Slots

As mentioned before, the end-devices use the beacons sent by the gateway to synchronize. This means that the end-devices count the time since the last beacon to calculate when they can send messages. The implemented protocol divides the time into equal cycles. A new cycle starts when a beacon is received. The end of each cycle is reserved for end-devices who reserved time-slots.

To calculate the Time on Air (ToA), the time needed to transmit a message, a payload of 13 bytes, Spreading Factor 12 ($10^{12} = 4096$), Bandwidth of 125 kHz and Coding rate of $\frac{4}{5}$ were used. These values were found in documentation [7] as the average values in IoT typical scenarios. With the conditions stated before, the ToA is 1155 ms. Taking into account a duty cycle[1] of 1% , a device should send a message every 115,5 or more seconds. The formulas used for the calculating ToA are the following:

$$T_{Symb} = \frac{2^{SF}}{Bandwidth * 1000} \tag{3.1}$$

$$T_{Preamble} = (N_{PreambleSymbols} + 4.25) \times T_{Symb} \tag{3.2}$$

$$N_{PayloadSymbols} = \frac{8 \times N_{PayloadBytes} - 4 \times SF + 28 + 16 - 20}{4 \times SF \times CR} \tag{3.3}$$

$$T_{Payload} = N_{PayloadSymbols} \times T_{Symb} \tag{3.4}$$

$$T_{Packet} = T_{Preamble} + T_{Payload} \tag{3.5}$$

The gateway cycles should have at least twice this value, to provide enough time for uplink messages, so 240 seconds (4 minutes) is the period selected for each cycle. From these 240 seconds, 10% were reserved for TDMA, that is, during this period only authorized end-devices can transmit. Figure 3.2 represents the superframe structure.

Figure 3.2: Proposed superframe structure.

According to the used transmission parameters, the mean time on air of each message is 1155

[1] Duty cycle is the fraction of time a device can be transmitting.

ms. To avoid collisions between different time-slots a security gap between time-slots has been set. Therefore, each time-slot has a duration of 2 seconds. The 24 seconds available for time-slots divided by these 2 seconds makes 12 time-slots available for reservation.

According to the defined protocol the end-devices know that 216 seconds after the beacon they can only send messages if they have a reserved time-slot. After receiving a new beacon, they are allowed to send messages again for 216 seconds.

Time-Slot Messages Exchange

As seen before, each time-slot has a duration of 2 seconds and there are 12 time-slots available for reservation. At this time, there are no reservation signalling protocols defined in the LoRaWAN specifications, so these protocols have to be developed from scratch. This chapter will present and explain the developed protocols.

The end-device asks the gateway for a time-slot. After this request, the gateway should answer with the time-slot the device should use. Figure 3.3 represents the reservation message handshake.

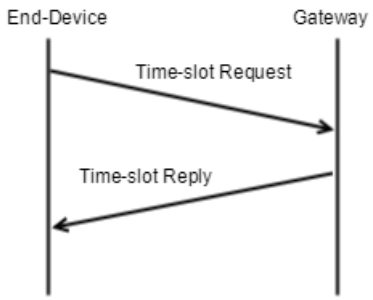


Figure 3.3: Time-slot reservation message exchange.

The request message must include the end-device identification. In this way, the gateway knows which device is requesting a time-slot. LoRaWAN specification includes this as DevAddr (Device Address). This field is a 4-byte unsigned integer, giving a total of $2^{4\times8} = 2^{32}$ available addresses, this device identification will be sent with the request for a new time-slot, in order to avoid giving two time-slots to the same end device. The second field of the request is Period, which is an 1-Byte integer. This field corresponds with the number of cycles the end-device need the time-slot to be reserved. Table 3.2 represents the structure of the time-slot request.

Table 3.2: Request time-slot message structure.

4 Bytes	1 Byte
DevAddr	Period

A soft reservation scheme is adopted, whereby the time-slot reservation is only valid for a maximum of 3600 seconds, i.e. 15 cycles of 240 seconds. This maximum assures that a time-slot doesn't stay reserved even if the end-device for any reason stops working. For this reason, there is no message to free the time-slot as it will be automatically freed. If the end-device intends to keep the time-slot

reservation, it will have to send a request message before the end of the expiration time of the previous reservation. The reservation timer is then reset to 0.

The reply message must contain the Device Address (DevAddr) and the assigned time-slot (TimeSlotAss). The TimeSlotAss field is a 1 Byte unsigned integer, with values from 0 to 12. If this field takes the value 0, it means that there are no time-slots available. The values from 1 to 12 correspond to the time-slot reserved for that device. For example, number 2 means that the device can send a message on the second time-slot, that is, 218 to 220 seconds after the beacon. Table 3.3 represents the structure for the reply of the time-slot request.

Table 3.3: Time-slot request reply structure.

4 Bytes	1 Byte
DevAddr	TimeSlotAss

Incoming Message Processing Algorithm

In order to implement time-slot reservation mechanism, there is an array that stores the device address associated with each time-slot, empty time-slots take the value -1, as address 0 is a valid one. Each module has a different array, as the reservations are only valid for the channel that module is listening to.

When a time-slot request comes from an end-device, the gateway checks in that module's array if that end-device is already registered. If so, sends a message with the time-slot already registered and resets the time counter. Otherwise it sends, if available, the time-slot assigned to the end-device.

When a message is received during reserved time-slots, the device address is checked against the array values. If the values do not match, the message is discarded. If that device has a time-slot reserved for it, but used a different one, the reservation reply message is re-sent with the same structure as in Table 3.3. Figure 3.4 shows the procedures taken when a message arrives.

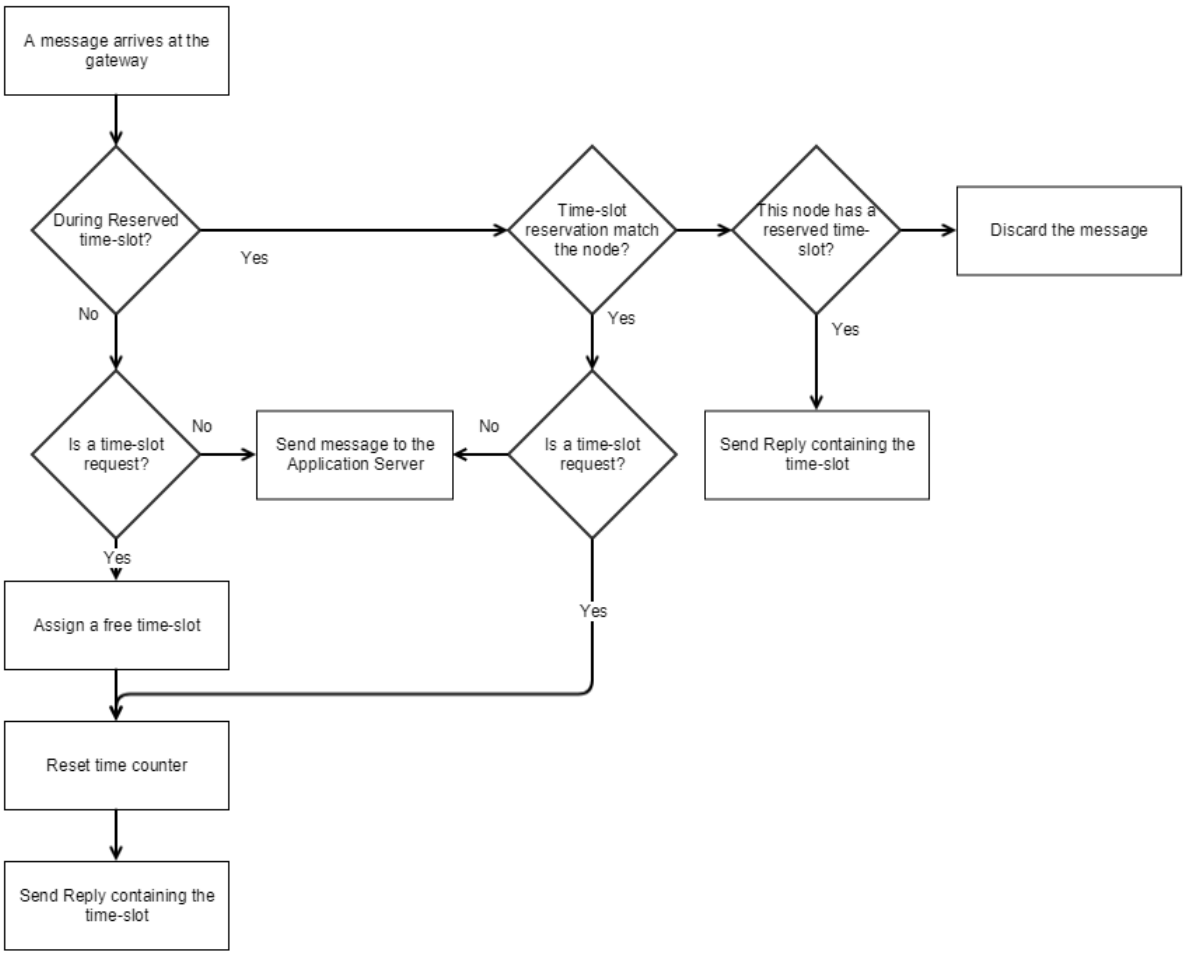


Figure 3.4: Procedures taken when a message arrives Flowchart.

3.2.2 Frequency-Hopping Algorithms and Protocols

This subsection presents the algorithm and protocols implemented for the frequency-hopping mechanism. It describes the two possible channel modes, the channel calculation algorithm and the beacon structure implemented for the frequency-hopping mechanism.

As mentioned before, LoRaWAN defines several channels in the frequency domain. Table 3.4 shows the frequency channels used in the proposed implementation.

Table 3.4: Frequency channels used in the proposed implementation.

Channel	Frequency (MHz)
0	868.1
1	868.3
2	868.5
3	867.1
4	867.3
5	867.5
6	867.7
7	867.9

Note that LoRaWAN defines 10 channels for EU, but the above table only shows 8. The other two

channels were not used as they are not suitable for the presented implementation. The first is reserved for gateway downlink messages, which is not suitable for single-channel implementations. The second is reserved for FSK.

Although an 8 channels implementation has been taken, the algorithms implemented allow to change the number of channels and corresponding frequencies. For example, according with the United States LoRaWAN regulations, there are 64 channels available. To use the implemented gateway on other region, the parameters and the LoRa modules must be changed to adopt local regulations.

Channel Modes

As seen before, the gateway will have one or more LoRa modules, each module working on a different channel. A module sends a beacon with the frequency-hopping channel and stays tuned to the new channel for 240 seconds. After this period, the module sends a new beacon and switches to other channel, keeping the loop.

Two communications can happen simultaneously in the same channel as long as they use different spreading factors. This way two operating modes were implemented:

1. Shared channels mode : The different modules have different spreading factors and each module can operate on all channels available without interference. This system provide the best spreading factor to each sensor. For example a sensor far from the gateway will need a higher spreading factor to reduce BER. On the other hand, if the sensor is close to the gateway, the spreading factor can be lower, reducing the transmission times.

2. Non-shared channels mode : The different modules have the same spreading factor, but different channels. The descriptions that follow describe how the channels are divided to ensure that two modules are never in the same channel.

It is possible, with the implementation made, to change the operating mode, the corresponding spreading factors or the number of channels. Table 3.5 presents the default values for a gateway with 2 RFM95 modules and with 8 available channels (according with EU regulation).

Table 3.5: Spreading factor, bandwidth and number of channels defaults.

Spreading factor	Shared channels mode: SF7 and SF12 Non-shared channels mode: SF12
Bandwidth	125kHz
Number of channels	8

Channel Calculation Algorithm

Flowchart 3.5 represents the procedure used to calculate the channel according with the operating mode. The first step it to generate a random number – the algorithms used to obtain the random values will be discussed below. The second step is to calculate the remainder of the division of the random number by 8, in the shared channels mode. In this mode all channels are available for the modules. In

the non-shared channels mode, the procedure is to calculate the remainder of the division of the random number by 4 and multiply it by 2. On module1 this value will be added one.

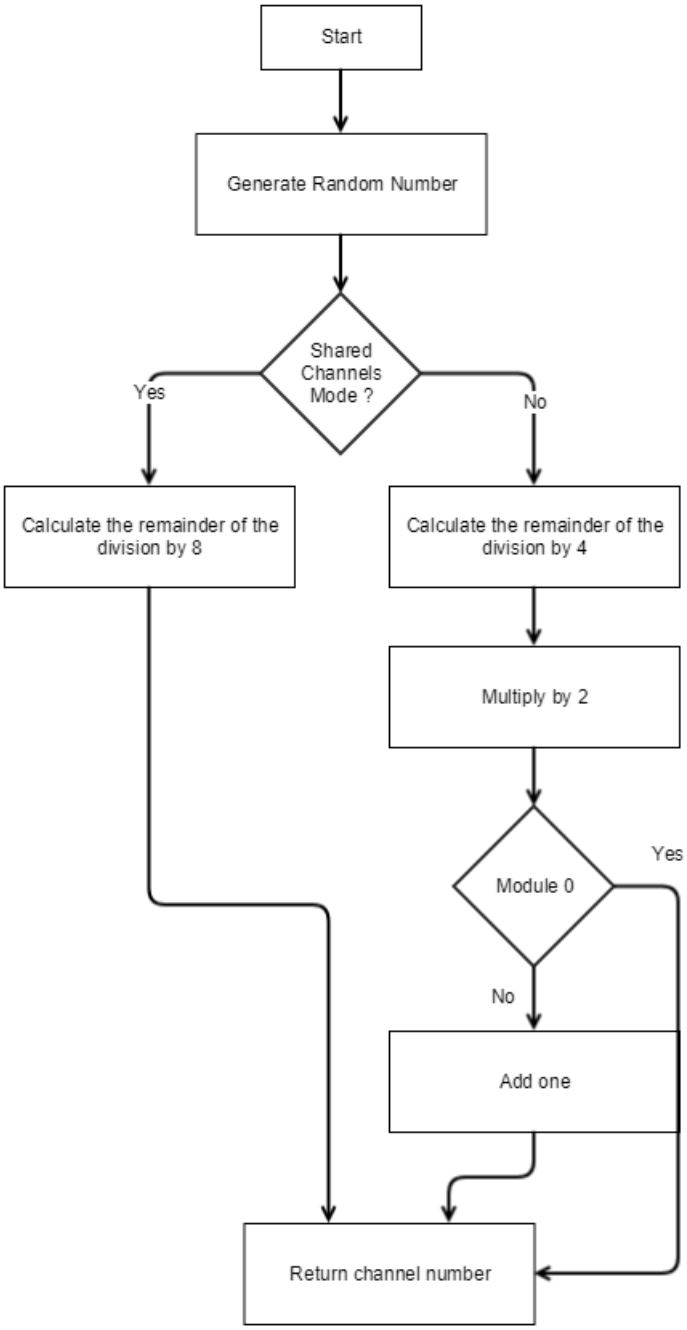


Figure 3.5: Calculation of the channels flowchart, according with the mode and the module.

Figure 3.6 represents the allocation of the channels for each module, considering 8 available channels and two modules. Module 1 can only operate on red channels, module 2 can only operate on blue channels. This could be extended to more modules.

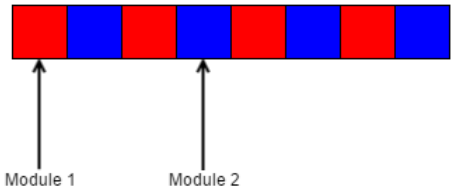


Figure 3.6: Allocation of 8 channels by 2 modules.

The information about the next channel to be used is sent encrypted and only the legit devices know the decrypt code. This assures that an attacker with jamming capability does not get the message, allowing it to change channel to interfere with the communication.

Beacon Structure

LoRaWAN defines a message to be sent from the gateway to the end-device containing the channel changing command. The name of this is NewChannelReq and its message structure is represented in Table 3.6.

Table 3.6: NewChannelReq structure according with LoRaWAN (extracted from [17]).

1 Byte	3 Bytes	1 Byte
ChIndex	Freq	DrRange

The Channel Index (ChIndex) is the number of the channel being created. It adopts values between 0 and 15. The frequency (Freq) is an unsigned integer of 24 bits. The real value of frequency is Freq x 100, allowing a frequency range between 1.67 GHz to 100MHz in 100 Hz steps. The Date Range (DrRange) specifies the allowed bandwidths. This Field splits in two 4 bits indexes represented in Table 3.7.

Table 3.7: DrRange Structure according with LoRaWAN (extracted from [17]).

MinDr	MaxDr
4 bits	4bits

Minimum Data Range (MinDr) is the minimum Bandwidth this channel can support, Maximum Data Range (MaxDr) is the maximum Bandwidth this channel can support. Both fields can take the values according with Table 2.5. For example, for SF12 / 125 KHz the data rate is DR0. If this is the only configuration available, then both MinDr and MaxDr will be DR0. This way DrRange will take the value 0x00.

3.3 Security

As presented before in Section 2.2.1, LoRaWAN defines a protocol to implement security from the end-device to the application server. This protocol uses the AES-128 algorithm to encrypt the message with two different keys. Table 3.8 shows which components of the network have each key.

Table 3.8: Network components who have each encryption key.

	AppSKey	NwkSKey
End-device	X	X
Gateway		
Network server		X
Application server	X	

The first key (NwkSKey) encrypt all the message between the end-device and the network server. The second key (AppSKey) encrypt only the data between the end-device and the application server.

In the stated architecture, the network server and the gateway have no access to the data encrypted with AppSKey. Therefore, control messages as the beacons are encrypted only using the NwkSKey. Messages containing data, for example, sensor readings from the end-device are encrypted with the both keys.

3.4 Gateway Implementation

This section presents the gateway hardware components, and the software implemented algorithms.

3.4.1 Gateway Hardware

The LoRa gateway has multiple single channel LoRa modules, each one giving the ability to receive in one channel. A Raspberry Pi was used, which communicates with the LoRa modules by SPI and sends the data to the cloud through the Ethernet cable. Figure 3.7 represents this components.

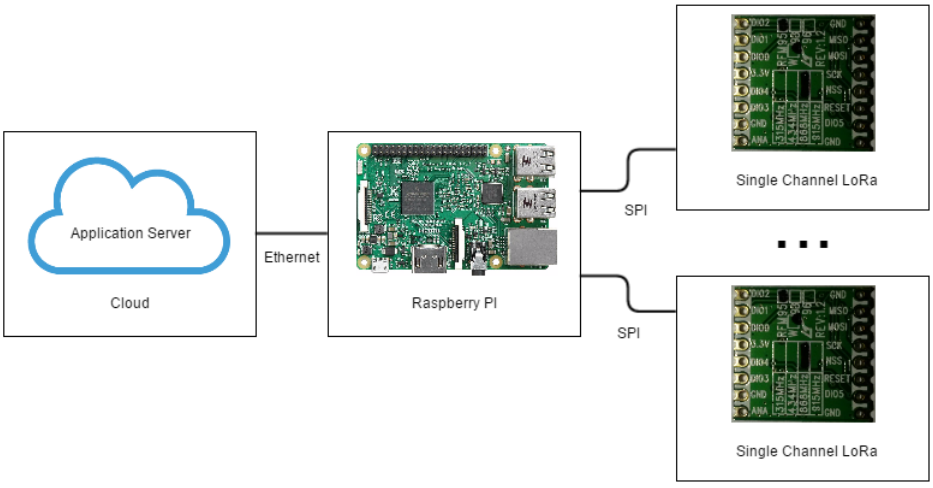


Figure 3.7: Hardware architecture.

SPI (Serial Peripheral Interface) is the protocol used to communicate between LoRa modules (SPI slave devices) and the Raspberry Pi (SPI master device). In the SPI architecture, a master device that communicate with each slave must exist – slaves cannot communicate to each other. Figure 3.8 represents the SPI pinout used.

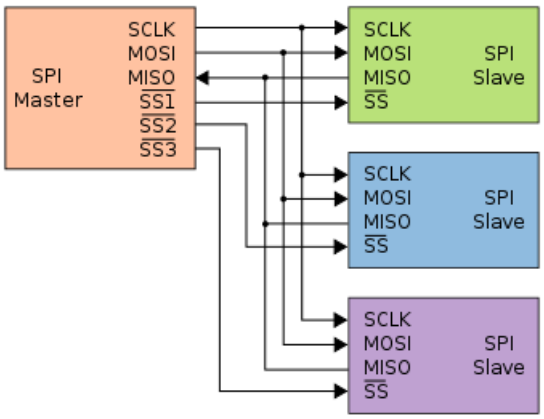


Figure 3.8: SPI architechture (extracted from [20]).

The SPI architecture defines four electronic pins:

1. SCLK or SCK (Serial Clock) pin that synchronizes all the devices: this pin is shared by all the SPI slaves.

2. MOSI (Multiple Output Single Input) and MISO (Multiple Input Single Output) pins used for data transfer: these pins are shared between all the SPI slaves.

3. Chip Select (CS) pin, also named as Slave Select (SS): used for coordinate between several masters, so they don't use the shared pins simultaneously.

According with the SPI architecture CS pin could also be shared between different modules. However, to make it simpler to implement, it was decided to adopt the implementation stated before. Using this SPI architecture, the CS pin is normally held high, which disconnects the slave from the SPI bus. Just before data is sent to the module, the line is brought to low, which activates the module. When the communication has ended, the line is made high again. The role of the CS pins will be explained later on, as they will be controlled by software to select each module the gateway is communicating with.

RFM95 Breakout Boards

As mentioned in Section 2.3, the boards used to implement the gateway were the RFM95. These boards do not have the traditional pins to connect them to an Arduino or Raspberry Pi. To solve this issue a breakout board was developed and used. The breakout board was developed using eagle, a electronic design automation application. The schematic made was sent to a PCB producer, who manufactured it. Figure 3.9 shows the developed breakout board.

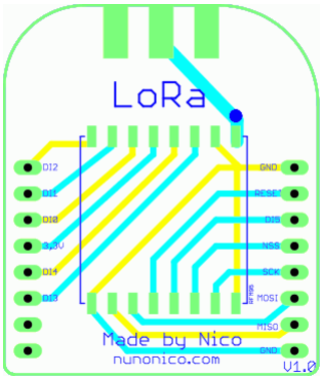


Figure 3.9: LoRa breakout board.

The produced breakout board adds the easy connector pins, but also offers option to connect it to a SMA connector antenna. This is very important as it reduces losses widely and increases gain. These breakout boards will, also, be used for the end-devices.

All the LoRa modules have been connected to +3dBm SMA half wave antennas. These antennas produce more robust communications and allow to minimize the effects of external noise when testing the gateway. Figure 3.10 shows the antennas used.

Figure 3.10: 868MHz Half wave antenna.

3.4.2 Gateway Software

This section presents the software libraries and the way they relate between them. After this, the important procedures of each library will be deepened.

As a starting point for this project a DIY low-cost gateway tutorial developed at the French University of Pau and Pays de l'Adour [7] has been used. This tutorial is about building a gateway with a Raspberry Pi and a single LoRa module. It is not resilient to channel jamming, as multi-channel, frequency-hopping and time-slot reservation capabilities are not developed which makes it very unreliable in situations of high load, where channels are often busy.

This tutorial's code uses a library to control generic LoRa modules such as the RFM95 and SX1276 made for Arduino (SX1272) with a layer that adapts this library to work on Raspberry Pi (ArduPi). The main software(LoRa_gateway) uses a *Setup* function to configure the module and then a *Loop* function to check for incoming messages. Both functions had to be adapted to support multiple modules, as will be later explained. The main software also sends commands to a Python program, through a pipe (Post_processing) that will decrypt incoming messages. Figure 3.11 represents the several libraries and the way they relate.

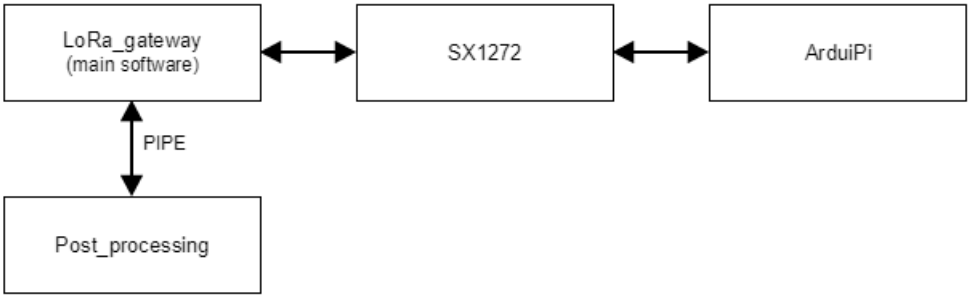


Figure 3.11: LoRa gateway libraries architecture.

Main Software

The main software first task is to launch another thread, the Beacon Processing Thread. The task of the latter, as the name implies, is to process the beacons. After this, the main software will setup the different modules (in this case there are two), and then enter an infinite loop, checking for new messages, received from the both channels available. Figure 3.12 shows the presented algorithm with a implementation of two SPI modules, module 0 and module 1.

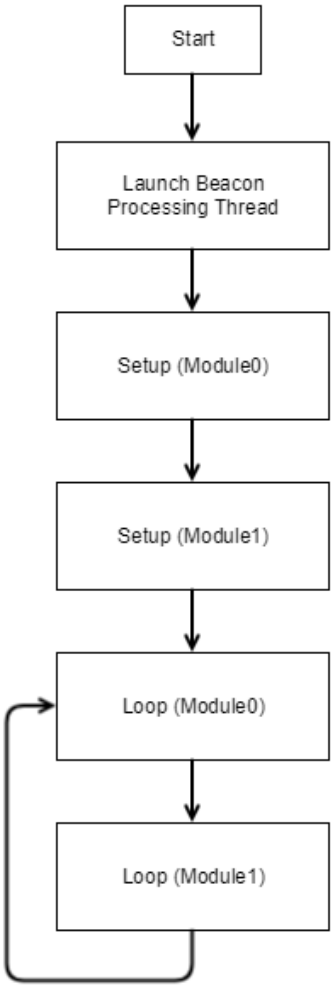


Figure 3.12: Flowchart that represents gateway main software working model with 2 modules.

Setup Function

The *Setup* function used in main software configures the LoRa module with all the parameters defined by the network Server, which should be used to receive messages such as the frequency, the spreading factor, the bandwidth, the preamble length, the power among others. Figure 3.13 shows the *Setup* algorithm.

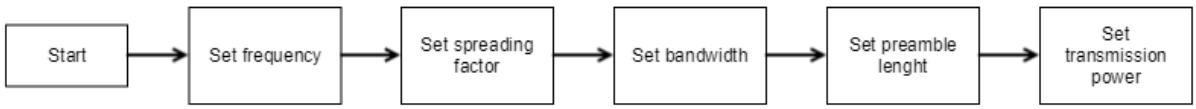


Figure 3.13: Flowchart that represents the *Setup* function.

Loop Function

The first step of the *Loop* function is to call a SX1272 procedure that assures the correct module is being controlled – this procedure will be detailed later. The second step is to restart a board if an error is detected. After ensuring the board is working normally, the *Loop* function checks for incoming messages. The message received is send through a pipe to Post_processing to be decrypted. Then, if there is a downlink message to be sent, the *Loop* function will send it. There is no risk of losing messages while processing other modules, as they are temporarily stored in the RFM95 module. Figure 3.14 shows the *Loop* algorithm.

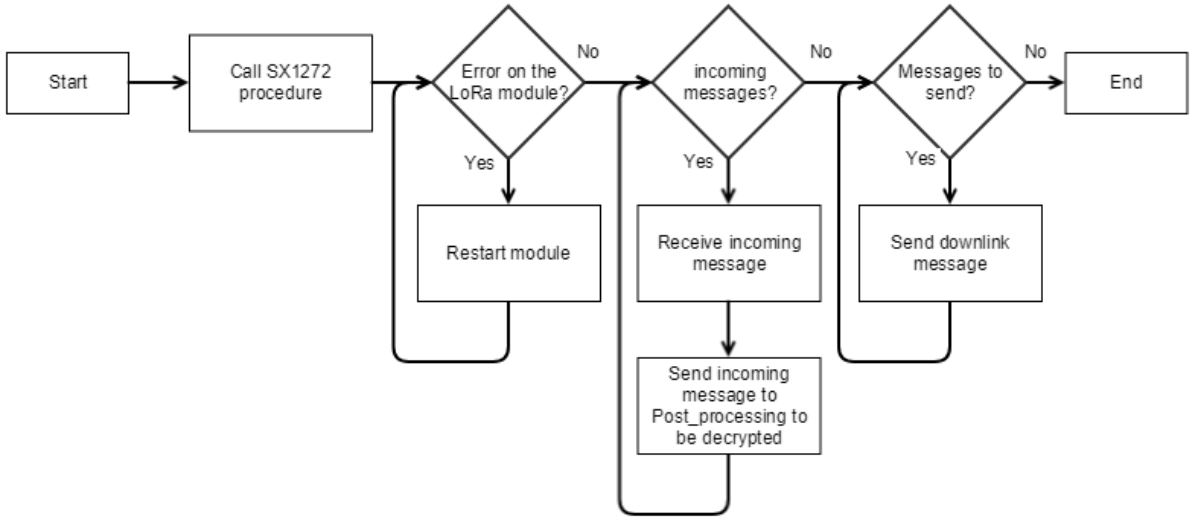


Figure 3.14: Flowchart that represents the *Loop* function.

Multiple Module Control

As mentioned before, modifications had to be made both on *Setup* and *Loop* functions, now they have a parameter which indicates what is the board being controlled. This corresponds to the CS pin that specific module is using, as Figure 3.12 shows. *Setup* and *Loop* functions, after the modifications made, call a SX1272 procedure to assure the correct board is being controlled. This SX1272 procedure has

been implemented to change the values of CS pins according to the module being controlled at that moment, as described in 3.4.1.

Reviewing the important role of the main software, its first step is to call for the Beacon Processing Thread, then setup all the boards sequentially, then sequentially check for incoming messages from the boards.

Beacon Processing Thread

As stated before, the Beacon Processing Thread is part of the network Server and is responsible for processing the beacons according with the specified rules. There is a time counter that assures that the beacon transmission procedure is called every 240 seconds, for each module. This thread is also responsible for deciding which module the beacon will be sent from and the next channel will be taken by the module. This thread only sends the beacon transmission orders – the process of sending the beacon is taken by the gateway.

Flowchart 3.15 explains how the beacon Processing Thread works.

The first step is to initialize the module counter. This counter will save the next module to be processed. After this, the timeCounter will also be initialized. This counter will save the time since the last beacon.

The second step is to wait for the time counter to reach 120 seconds. By doing this, the gateway will change a module channel every 120 seconds, alternately. There are two modules, therefore, each module will change channel every 240 seconds.

After ensuring the timeCounter reached 120 seconds, the CalculateChannel procedure will be called. After calculating the channel, a beacon command with the calculated channel will be sent. After sending the beacon the gateway will change the channel.

To finish the procedure, the ModuleCounter is toggled and the loop restarts.

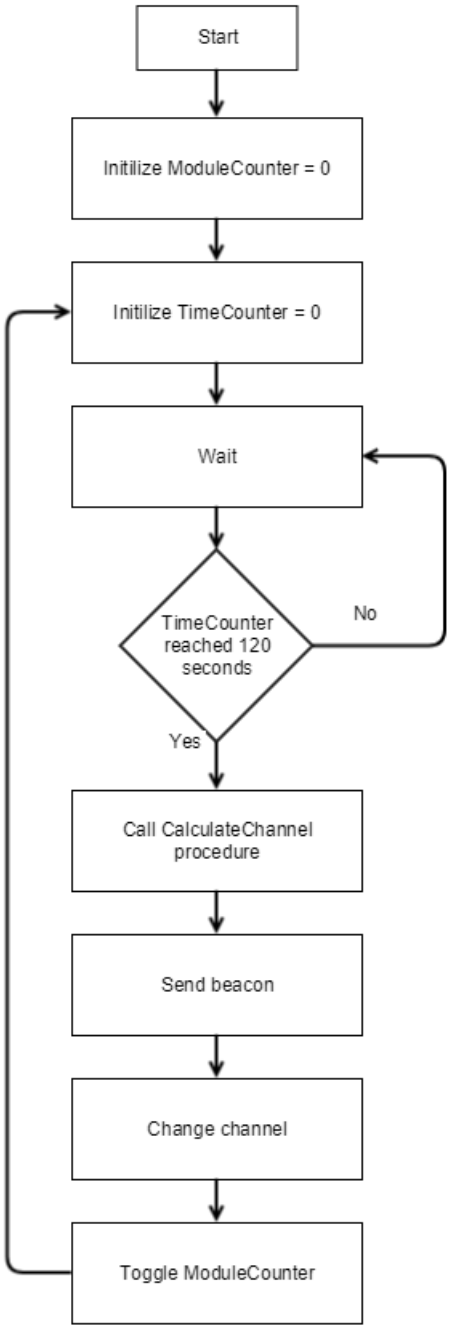


Figure 3.15: Beacon processing Thread flowchart.

3.5 Application Server Implementation

The gateway receives the data, it is processed by the network served and then sent to the application server. This application server receives the data through HTML requests, and store it on a SQL database. This server were implemented and runs on the cloud.

The message sent by the gateway to the application server contains the AES-128 encrypted message. This message is decrypted using the AppSKey. After being decrypted the message is stored in the database.

Database Architecture

The application server saves in a SQL table following fields (Table 3.9 presents the SQL fields):

- messageID : This field identifies each message. There are no messages with the same messageID. This field is a 8 Bytes integer, therefore can take values up to $2^{63} \approx 10^{19}$.
- messageTime : This field contains the date and the time the message is received.
- deviceID : This field contains the device identification code. The field is a 4 Bytes integer, the same size defined by LoRaWAN to identify the devices.
- messagePayload : This field contains the messages received by the gateway. Each message can contain up to 40 characters.

Field name	Data type	Data type format
messageID	BIGINT	numbers up to 2^{63}
messageTime	DATETIME	YYYY-MM-DD HH:MI:SS
deviceID	INT	4 Bytes integer
messagePayload	VARCHAR(40)	Strings up to 40 characters

Table 3.9: Application server database architecture.

Testing Software

Application server was used to test the frequency-hopping and time-slot reservation mechanisms. Therefore, it was implemented on application server software to analyze the data. This testing software will be presented on Chapter 4.

3.6 End-Device Implementation

In order to test the developed gateway, an end-device was also implemented. This end-device presents the following features:

1. Was developed using Arduino software found on tutorial developed at the French University of Pau and Pays de l'Adour [7].
2. Changes the communication channel according to the beacon instructions. This frequency-hopping mechanism can be disabled.
3. Can reserve time-slots and only sends data in a time-slot if that time-slot is reserved for it.
4. Only sends data if the last beacon was correctly received.

If no beacon is received within 240 seconds, the end-device knows that a beacon is missing, therefore it will not send messages until receiving a beacon. To avoid being tied on a channel for not receiving the beacon, the end-device will automatically change channel after 5 cycles, i.e., 1200 seconds. This

procedure is only taken when the frequency-hopping mechanism is enabled. This implementation will be used further to test the time-slot reservation and frequency-hopping mechanisms. Figure 3.16 presents the algorithm for the end-device.

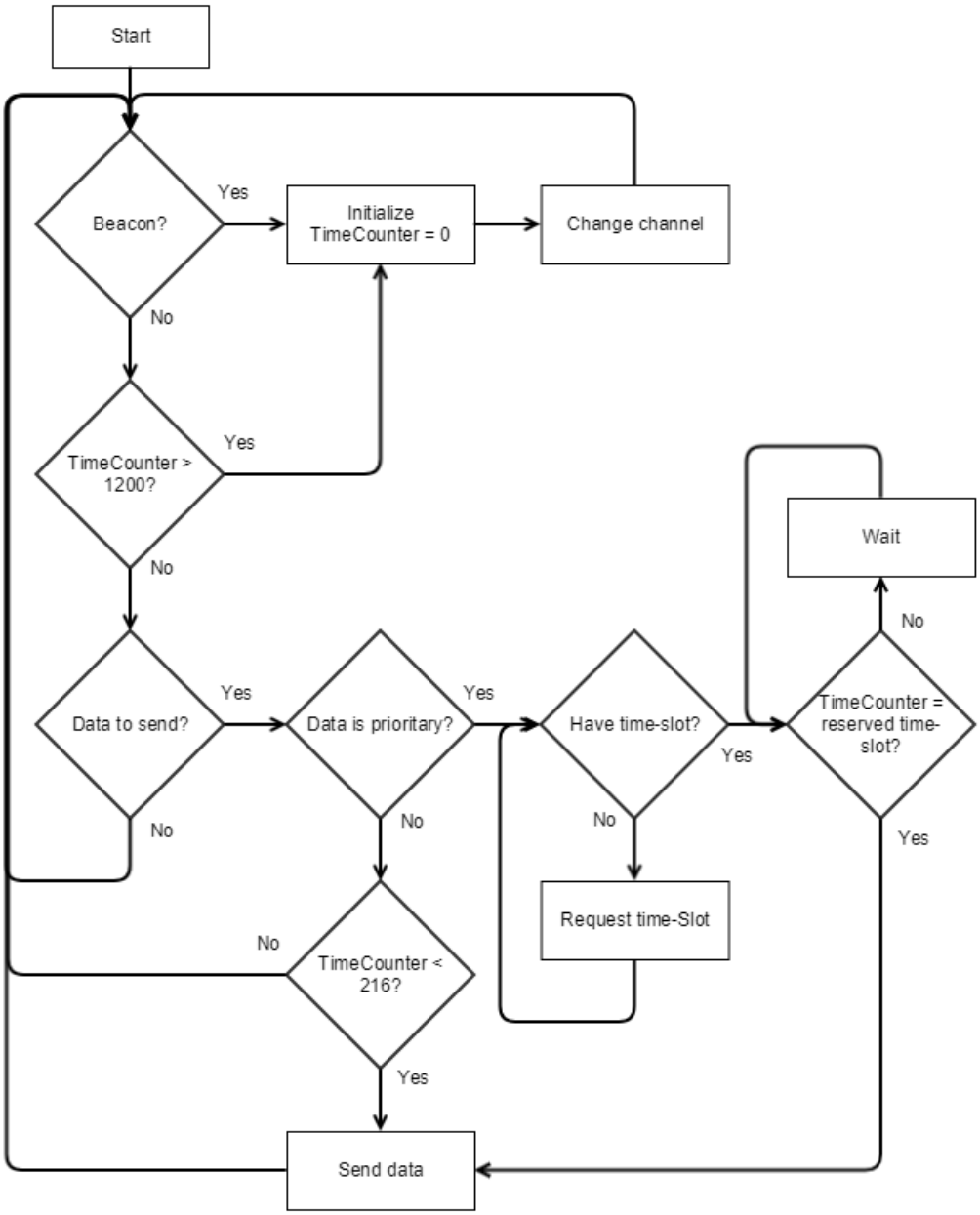


Figure 3.16: End-device algorithm flowchart.

End-Device Harware

The RFM95 module works with 3.3 Volts. This is the same voltage as the Raspberry Pi, but most Arduino boards works with 5 Volts. Therefore, there was the need for an Arduino board capable of working with 3.3 Volts and with a considerable processing power, as the low range Arduino has not power enough to encrypt and decrypt the messages in real time. Arduino M0 was found to suitably these requirements. Table 3.10 presents some of the technical specifications of Arduino M0, comparing it with the most used

one (Arduino Uno). Figure 3.17 shows an Arduino M0.

Table 3.10: Comparisson between Arduino M0 and Arduino Uno.

	Arduino M0	Arduino Uno
Working voltage (V)	3.3	5
Flash Memory (KB)	256	32
SRAM (KB)	32	2
Clock Speed (MHz)	48	16

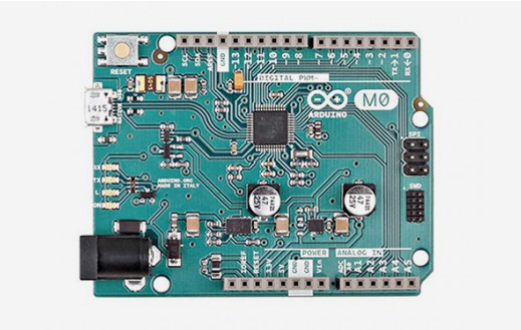


Figure 3.17: Arduino M0 (extracted from [15]).

Chapter 4

Results

This chapter presents the functional and performance evaluation of the developed system. The first set of tests aims to evaluate the communications range. The second set of tests aims to evaluate the randomness characteristics of the frequency-hopping sequences. The third set of tests aims to evaluate the implemented time-slot reservation and frequency-hopping mechanisms.

4.1 Communications Range Evaluation

Range evaluations were made to assess the gateway's ability to receive messages over long distances. These evaluations were divided in two two. The first evaluation measures how the signal power decays with the distance. The second evaluation measures the maximum distance an end-device can be from the gateway and still send successfully messages to it.

As mentioned before, one of the possible applications of LoRa technologies is the farm monitoring across vast areas. This application was taken into account and the tests were done in rural environment. These areas, typically have no buildings. As there are no attenuation factors, large ranges are expected. The tests were made at a wine producing farm near Lisbon, the capital of Portugal. Several sensors cloud be placed in this farm to monitor temperature, humidity, and soil moisture as these are factors that influence the quality of the produced wine. The gateway would be placed in a building and collect the data generated by the sensors. Figure 4.1 shows the farm where the tests were done. The area of the farm is delimited in blue, the location of the gateway is marked with green.

Figure 4.1: Farm used for the communications range evaluation.

4.1.1 Decay of Received Signal Power with Distance

To measure how the signal power decays with the distance one end-device and the developed gateway were used. The end-device was configured to send messages periodically. These messages were received by the gateway, where the received power was measured.

To perform this evaluation, the gateway was placed in a static position and the end-device was displaced at several positions, increasing the distance from the gateway. For each position, 5 signal power measurements were taken, and the mean values have been registered. The graph on Figure 4.2 shows the measured values. There is also represented in grey a trendline. Equation 4.1 shows the trendline equation. Note that the trendline follows the Free Space Path Loss (FSPL) equation. This was expected as the transmissions were done with line of sight.

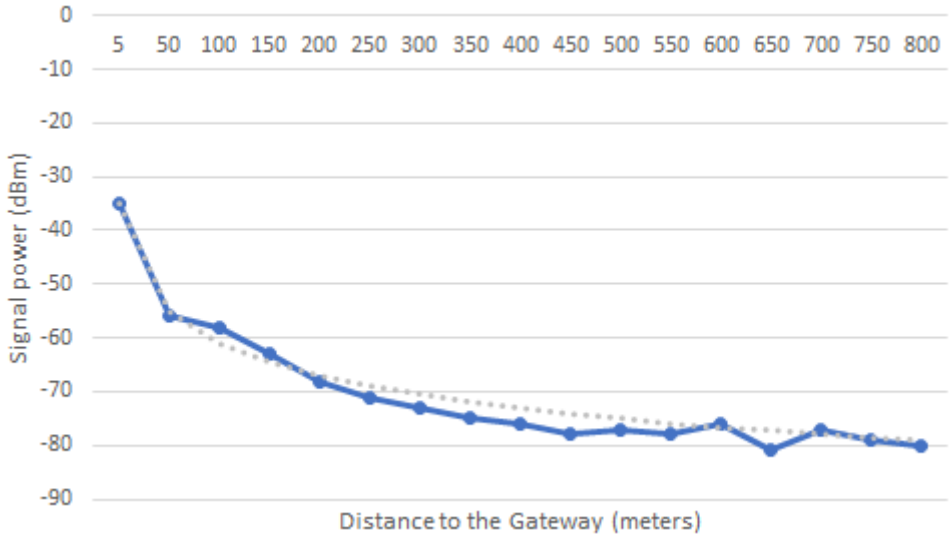


Figure 4.2: Signal power and trendline.

$$Trendline = -21.02 - 20 \log_{10} Distance \tag{4.1}$$

The measurements made show that in rural environments, where there are no obstacles. Communications with distances up to 1km can be easily achieved. For these distances, even with small obstacles like trees, the communications can be done with success.

4.1.2 Maximum Range

The maximum distance an end-device can be from the gateway and still send messages to it was measured. LoRa technology specifications claim a 15 km range in rural areas.

To perform this evaluation, the gateway was placed in a static position and the end-device was displaced at several positions, increasing the distance from the gateway. Note that points distanced several kilometers without obstacles are not common. Therefore, in order to test the success of the communications across long distances a previous planning had to be done.

This planning consisted in looking for end-device positions where it was possible to get line of sight to the gateway. To assure the pre-calculated positions were ideal, first Fresnel zone was taken into account.

A Fresnel zone, is one of a series of concentric ellipsoidal regions of space between and around a transmitting antenna and a receiving antenna system. Any obstacles inside the first Fresnel zone reflect the transmitted signal in a way that causes destructive interference at the receiver. Figure 4.3 shows a diagram of the first Fresnel zone.

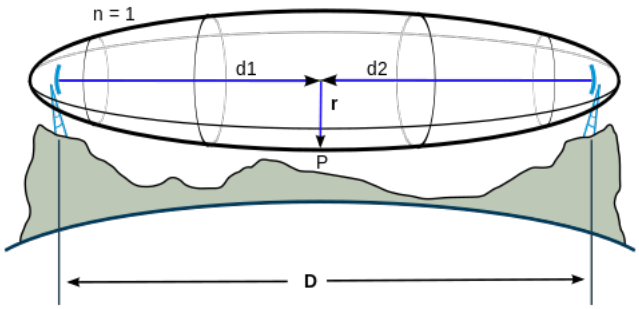


Figure 4.3: Fresnel Zone (extracted from [24]).

Several high altitude spots have been pre-selected. After this, a Google Maps script has been used. This script was used to measure the altitude in several points in the path between that spot and the gateway, for each of the pre-selected zones.

The path with the longest distance, and clear from obstacles, that was found was 4.9 km long. Figure 4.4 shows the altitude graph of the path. The lowest point represents the location of the gateway. The highest point represents the location of the end-device.

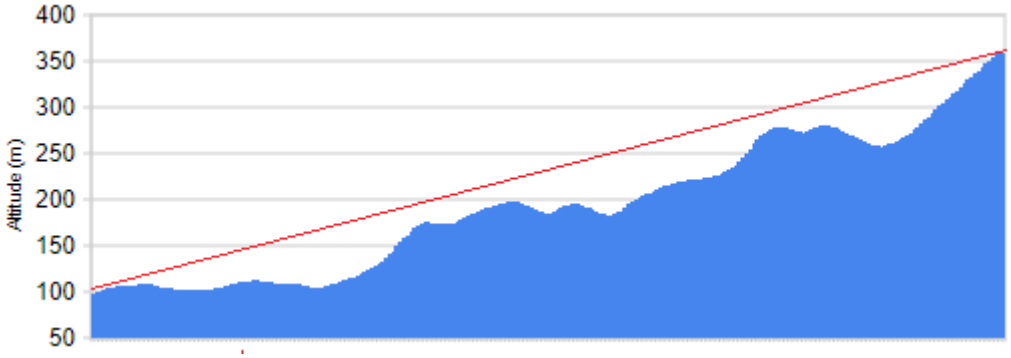


Figure 4.4: Altitude graph of the path (generated with Google Maps script).

The Figure shows where hills could be an obstacle to the line of sight. The first is on the middle of the path, the second is near the end-device location. These points were analyzed in detail and the resulting data is listed in Table 4.1.

The gateway was placed at 6 meters higher than the ground level as it was place inside an house, with the antenna on a window. The end-device altitude was one meter higher than the ground as it was being held by a person.

To calculate the line of sight altitude at the first and second hills, a rule of three has been used. Equation 4.2 was used to calculate the line of sight altitude.

$$H_{point} = \frac{H_{End} - H_{Gateway}}{D_{total}} \times D_{point} + H_{End} \tag{4.2}$$

To calculate the first Fresnel radius, Equation 4.3 has been used. *D* is the total distance of the path, *Z* is the distance from the starting point, the gateway.

$$R_{Fresnel} = \sqrt{\frac{Z(D-Z)}{D} \times \lambda} \tag{4.3}$$

To calculate λ, the wavelength, Equation 4.4 has been used. *c* is the speed of light and *f* is the frequency of the signals used, which was 868Mhz.

$$\lambda = \frac{c}{f} \tag{4.4}$$

Table 4.1: Path points.

	Gateway	First Hill	Second Hill	End-device
Distance to the gateway (km)	0	2.55	3.63	4.9
Altitude (m)	98 (+6)	195	274	361 (+1)
Line of sight altitude (m)	104	238.3	295.1	362
Fresnel radius (m)	0	20.5 (217.8-258.8)	18 (277.1-313.1)	0

The values calculated and represented on Table 4.1 show that there are no obstacles to the line of sight. Therefore this point was tested and messages were successfully sent to the gateway. The signal

power measured at the gateway shows that even bigger distances could be tested. Figure 4.5 shows the signal path in a map. This figure was taken from Google Earth and the signal path was drawn.

Figure 4.5: Map with the signal path (Taken from Google Earth).

The measurements also show that this distance was only possible because there was line of sight to the gateway. Messages sent on other locations closer to the gateway but without line of sight, were not received by the gateway.

Data Analysis

The results taken from this evaluation show that in a rural environment ranges with one kilometer can be easily achieved, even with small obstacles like trees. Larger ranges up to 5 km can be achieved with line of sight.

With the range 1km assured, it is possible with a single gateway to cover a 314 ha farm. This proves that the developed gateway is suitable to receive data from sensors spread across vast areas, in rural environments.

4.2 Assessment of the Random Characteristics of the Frequency-Hopping Sequences

The randomness of the frequency-hopping sequences is important to prevent jammers from easily predicting them. This would make the frequency-hopping worthless to prevent jamming attacks.

In this section, tests will be presented, which were made regarding the randomness of frequency-hopping sequences, addressing two aspects. The first aspect is related with how uniform the distribution is. The second aspect is the randomness of the results in a graphical view – this second aspect is more qualitative.

In Section 3.2.2, two modes of operation were introduced. The first mode uses shared channels and different spreading factors for each LoRa module. The second mode uses the same spreading

factor and different channels. These two models were tested to assess randomness of the generated frequency-hopping sequences.

4.2.1 Uniform Distribution Results

This subsection presents the uniform distribution results of two alternative random number generators: based on current timestamp, and based on thermal noise on Raspberry Pi pins.

To analyze if the results random sequences match a uniform distribution, chi-squared test was used. This test is represented by Equation 4.5, where lower values mean a more uniform distribution.

$$\chi^2 = \sum_{i=1}^{k} \frac{\left(Y_i - \frac{n}{k}\right)^2}{\frac{n}{k}} \tag{4.5}$$

In the equation, *k* is the number of available options, which are 8 in shared channel mode, and 4 in the non-shared channel mode. *n* is the sum of the generated random numbers, which in this example is 400. Y_i takes the number of times each channel number has been randomly generated.

Timestamp Based Random Number Generator

On the first mode, all channels are available for each module. Therefore, it is expected to have the time equally divided by the eight available channels. Tables 4.2 and 4.3 present the results of the number of times each channel was assigned to each radio module.

Table 4.2: Number of each channel was assigned to module number 1, in shared channels mode.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	42	50	63	43	44	59	45	54	400
Percentage	10.5%	12.5%	15.8%	10.8%	11.0%	14.8%	11.3%	13.5%	100%

Table 4.3: Number of times each channel was assigned to module number 2, in shared channels mode.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	54	57	48	52	40	54	56	39	400
Percentage	13.5%	14.3%	12.0%	13.0%	10.0%	13.5%	14.0%	9.8%	100%

Tables 4.2 and 4.3 chi-squared tests returned a 8.8 and 6.92 respectively. This means that the Table 4.3 had results that are more compliant with the uniform distribution.

With the second mode, half of the channels are available for each module. Therefore, it is expected to have the time equally divided by the four channels available. Tables 4.4 and 4.5 present the results of the number of times each channel was assigned to each module.

Table 4.4: Number of times each channel was assigned to module number 1, in non-shared channels mode.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	99	0	94	0	97	0	110	0	400
Percentage	24.8%	0%	23.5%	0%	24.3%	0%	27.5%	0%	100%

Table 4.5: Number of times each channel was assigned to module number 2, in non-shared channels mode.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	0	113	0	96	0	98	0	93	400
Percentage	0%	28.3%	0%	24.0%	0%	24.5%	0%	23.3%	100%

On Tables 4.4 and 4.5 chi-squared tests returned a 1.46 and 2.38 respectively. This means that the Table 4.4 had results that are more compliant with the uniform distribution.

The results obtained by the first and second mode cannot be directly compared as they do not have the same sample size. The Tables 4.2, 4.3, 4.4 and 4.5, along with the chi-squared results show that the algorithm assures uniform distribution of frequency-hopping sequences for both implemented modes.

Thermal Noise Based Random Number Generator

The alternative random number generator is based on the thermal noise of Raspberry Pi pins, which should result in a higher entropy. This algorithm was tested in the same way as the first. Tables 4.6, 4.7, 4.8 and 4.9 show the results for first and second modes, respectively.

Table 4.6: Number of times each channel was assigned to module number 1, in shared channels mode, using the improved algorithm.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	40	50	54	54	51	56	45	50	400
Percentage	10.0%	12.5%	13.5%	13.5%	12.8%	14.0%	11.3%	12.5%	100%

Table 4.7: Number of times each channel was assigned to module number 2, in shared channels mode, using the improved algorithm.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	59	57	51	45	38	51	44	55	400
Percentage	14.8%	14.3%	12.8%	11.3%	9.5%	12.8%	11.0%	13.8%	100%

On Tables 4.6 and 4.7 chi-squared tests returned 3.88 and 7.24 respectively. This means that the Table 4.6 had results that are more compliant with the uniform distribution.

Table 4.8: Number of times each channel was assigned to module number 1, in non-shared channels mode, using the improved algorithm.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	91	0	107	0	96	0	106	0	400
Percentage	22.8%	0%	26.8%	0%	24.0%	0%	26.5%	0%	100%

Table 4.9: Number of times each channel was assigned to module number 2, in non-shared channels mode, using the improved algorithm.

	0	1	2	3	4	5	6	7	Sum
Times Assigned	0	89	0	109	0	102	0	100	400
Percentage	0%	22.3%	0%	27.3%	0%	25.5%	0%	25%	100%

On Tables 4.8 and 4.9 chi-squared tests returned a 1.82 and 2.06 respectively. This means that the Table 4.8 had results that are more compliant with the uniform distribution.

Data Analysis

Table 4.10 summarizes the chi-squared results. The results show that the thermal noise based algorithm is slightly more compliant with the uniform distribution.

Table 4.10: Chi-squared results summary.

	Shared channels		Non-shared channels	
	Module 1	Module 2	Module 1	Module 2
Time based algorithm	8.8	6.92	1.46	2.38
Thermal noise based algorithm	3.88	7.24	1.82	2.06

4.2.2 Randomness Graphical View

It is important to bear in mind that the chi-squared function measures the compliance of the results with the uniform distribution. The results can be perfectly distributed, without being random. Take for example the binary sequence 0101010101, which is perfectly distributed, but not random. This difference can be easily seen graphically. Figures 4.6 and 4.7 show an example taken from [21] of the difference between a real random algorithm, like the based on thermal noise on Raspberry Pi, and the custom based algorithm, like the based on the current timestamp.

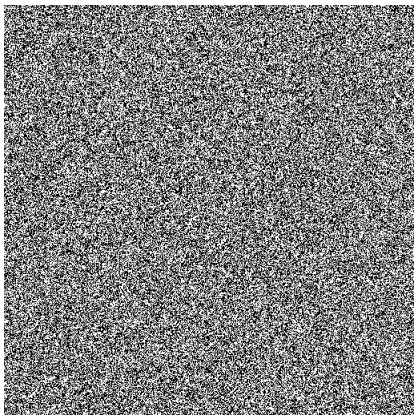


Figure 4.6: Real random algorithm (extracted from [21]).

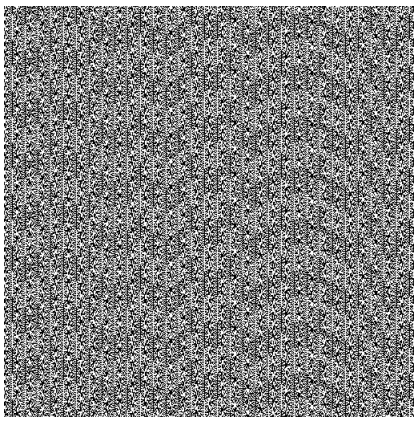


Figure 4.7: Custom random algorithm (extracted from [21]).

In order to graphically assess the randomness of both algorithms proposed in this dissertation, the results were transformed in a graphical representation. This was done using an open source software named Processing. Generated channel numbers are represented in a gray scale, considering channel 0 as white and channel 7 as black, the other channels taking different shades of grey. Figure 4.8 shows the algorithm used to create the random representation images.

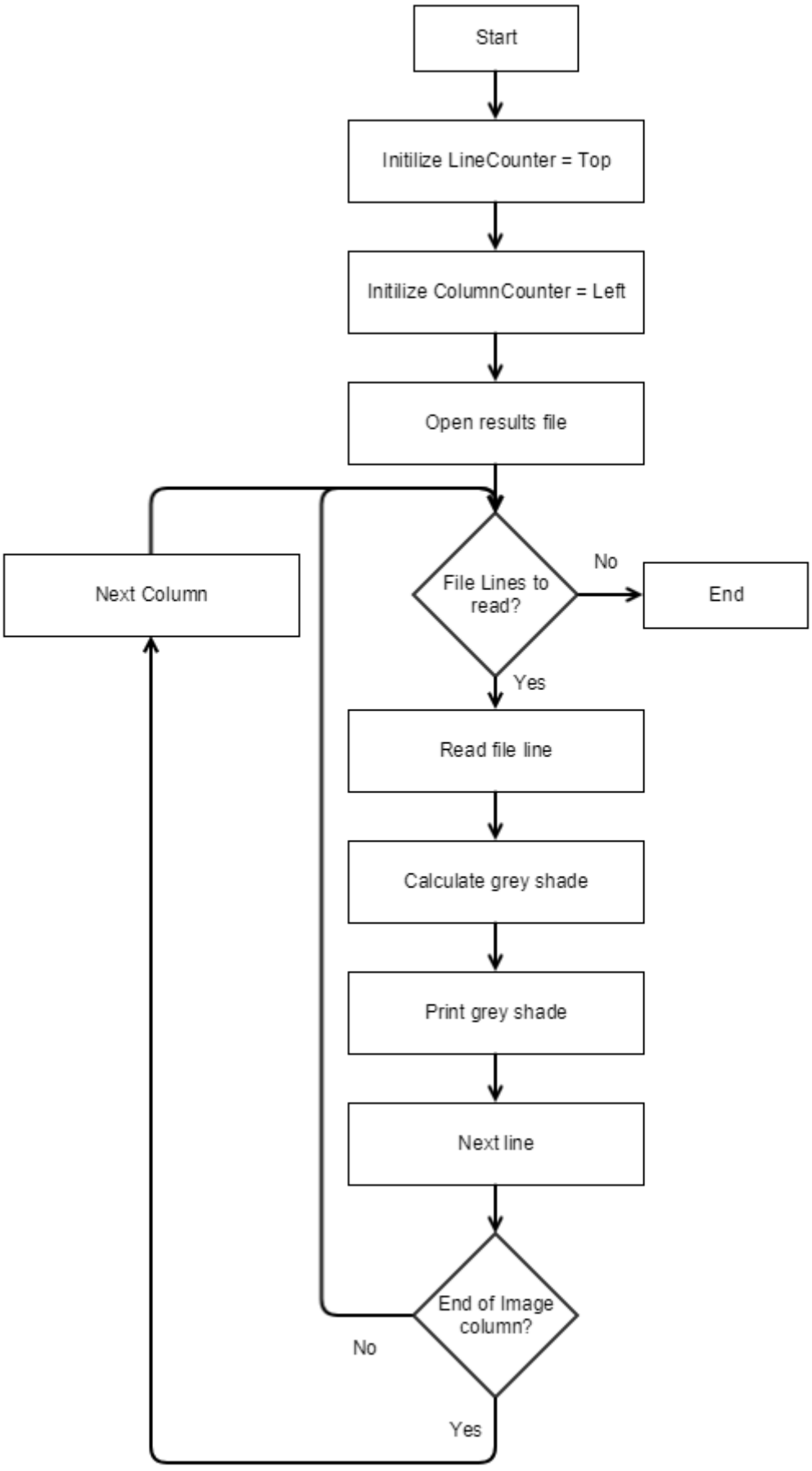


Figure 4.8: Algorithm used to create the random representation images.

To achieve representative results, the samples must be much more than the used on the chi-squared tests. Each test comprised 8100 cycles, resulting in 90 pixel squares. Note that a 8100 cycle test takes several days to complete. For this reason, and because the main purpose here is to test the random algorithm, only the shared channels mode was tested.

Figures 4.9 and 4.10 show the results of the random algorithm based on timestamp and the random algorithm based on thermal noise, respectively.

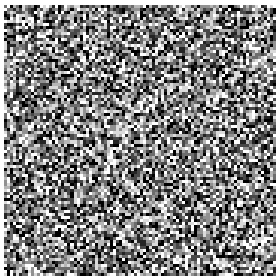


Figure 4.9: Classic random algorithm.

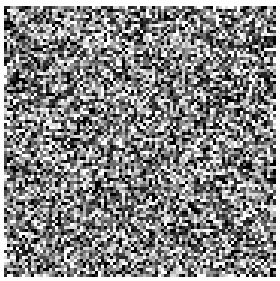


Figure 4.10: Thermal noise algorithm.

Data Analysis

Despite the number of samples was widely increased, the results still show no evidences that the thermal noise random algorithm is better than the classic random algorithm available for the Raspberry Pi. None of the above graphical representations show patterns as the examples presented in Figures 4.9 and 4.10. This is an indicator that both algorithms generate numbers with a good level of randomness.

As future work, the generated random sequences could be also analytically tested to find patterns. These tests would make the use the mathematical concept of convolution. The computer would take samples of the generated numbers and try to find the same pattern. To perform this, a computer with a high processing power must be used.

4.3 Performance Evaluation of the Time-slot Reservation and Frequency-Hopping Mechanisms

In this section, the time-slot reservation and frequency-hopping mechanisms were evaluated. There were two different evaluations. The first evaluates the time-slot reservation mechanism. The second

evaluates the frequency-hopping mechanism. These evaluations were done by comparing the number of successfully received messages.

For each mechanism, evaluation different scenarios were considered in the tests.

As mentioned before, a beacon is sent by the gateway every 240 seconds. This 240 seconds corresponds to a cycle. If the end-device does not receive the beacon, it does not send any message in that cycle.

For each scenario, 5 tests were done, each one with a duration of 100 cycles. By doing these 5 tests individually, confidence intervals can be calculated.

4.3.1 Time-Slot Reservation Results

The success rate when sending a message outside a time-slot is compared against the success rate when using a reserved time-slot.

For this evaluation, the frequency-hopping mechanism was disabled. As such, it was possible to compare the success rates of the time-slot mechanism itself. With the frequency-hopping mechanism disabled, beacons are only used by the end-device to synchronize with the gateway. This synchronization is used by the end-device to calculate time-slot schedule.

As mentioned before, the following parameters are configurable: the operating mode, the corresponding spreading factors and the number of channels. Table 4.11 presents the values used for time-slot reservation performance evaluation.

Table 4.11: Values used for time-slot reservation performance evaluation.

Spreading factor	SF12
Channel used	Channel 0 (868.1MHz)

As each 100 cycles test take almost 7 hours to complete, the tests were automated. The messages sent by the end-devices contained an identification code. These identification codes are saved and processed by the application server, which calculates the success rates presented below.

For this evaluation, the scenario considered two types of devices:

- The sender end-device, sending useful messages to the gateway. After receiving a beacon this end-device sends two messages, one on the reserved slot, the other on the contention-based time interval. The string ”reserved” is the identification code for messages sent on reserved time-slots and the string ”normal” is the identification code for messages sent outside reserved time-slots.
- The interfering device, which is at the same time, sending two messages per cycle. These messages are sent at a random time, during non-reserved periods, to simulate the real use of the gateway. The messages sent by this end-device contain the identification code ”other”, and a number to identify the interfering device that send the message.

To evaluate the time-slots reservation mechanism, three scenarios were be tested. In the first scenario, there were no interfering devices, in the second scenario a single interfering device was used and in the third scenario two interfering devices were used. Figure 4.11 shows the topologies used.

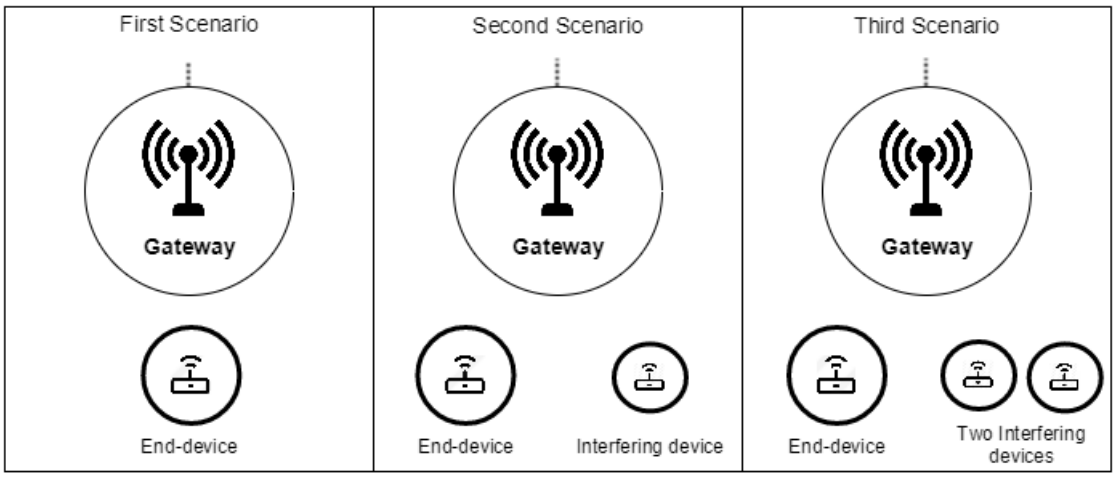


Figure 4.11: Schematic containing the tests performed for each scenario.

First Scenario

Table 4.12 shows successfully received messages. In this scenario no interfering devices were used.

Table 4.12: Number of received messages at the gateway, in the first scenario.

	Reserved time-slots	Contention-based periods
Test 1	97	98
Test 2	98	98
Test 3	93	95
Test 4	98	99
Test 5	98	97
Average	96.8 ($\pm$ 3.8)	97.4 ($\pm$ 2.4)

Second Scenario

Table 4.13 shows the number of successfully received messages. In this scenario a single interfering device was used.

Table 4.13: Received messages at the gateway, in the second scenario.

	End-device		Interfering device
	Reserved time-slots	Contention-based periods	
Test 1	91	72	130
Test 2	89	80	128
Test 3	91	68	135
Test 4	90	79	132
Test 5	86	67	131
Average	89.4 ($\pm$ 3.4)	73.2 ($\pm$ 6.8)	131.2 ($\pm$ 3.8)

Third Scenario

Table 4.14 shows the number of messages successfully received. In this scenario two interfering devices were used.

Table 4.14: Received messages at the gateway, in the third scenario.

	End-device		First interfering device	Second interfering device
	Reserved time-slots	Contention-based periods		
Test 1	84	62	102	98
Test 2	88	69	95	98
Test 3	87	64	100	98
Test 4	86	63	103	101
Test 5	84	60	101	97
Average	85.8 ($\pm$ 2.2)	63.6 ($\pm$ 5.4)	100.2 ($\pm$ 5.2)	98.4 ($\pm$ 2.6)

Data Analysis

Table 4.15 summarize the time-slot reservation results. It is clear that, when the network is being used by more than one end-device, the time-slot reservation is effective assuring the correct transmission of priority messages in the presence of additional end-devices.

The tests show that, when more than one device is operating on the network, the packet delivery rate of messages in reserved time-slots is much higher than using the contention-based periods. In fact, the success is 16.2 and 22.2 percent higher, for networks with one and two interfering devices, respectively. This allows extrapolating that with higher end node densities and higher traffic rates, the performance improvement due to the time-slot reservation mechanism will tend to increase quite significantly.

Table 4.15: Summary of the results in the three scenarios tested.

	Reserved time-slots	Contention-based periods
No interfering devices	96.8%	97.4%
One interfering device	89.4%	73.2%
Two interfering devices	85.8%	63.6%

4.3.2 Frequency-Hopping Results

For the frequency-hopping evaluation, two scenarios have been tested. In the first scenario, the frequency hopping was disabled and the end-device was using the same channel as the jammer. For the second scenario, the frequency-hopping was enabled.

For this evaluation, the time-slot reservation mechanism was disabled. As such, it was possible to compare the success rates of the frequency-hopping mechanism itself. For this evaluation, two devices were used:

- The end-device, which sent messages to the gateway. After each beacon is received, this end-device sent two messages, at a random time.
- A jammer implemented as an end-device sending tiny messages in a channel. The purpose of the jammer was only to cause interference. Therefore, the messages it sent were not LoRaWAN messages and could not be be not received by the gateway.

The channel targetted by the jammer was the same used by the end-device and the gateway, in the scenario without frequency-hopping. This simulates a situation where a jammer is being used,

and allows to evaluate the gain of a frequency-hopping algorithm in such situations. In this scenario, no messages from the end-device have been received. The jammer successfully destroyed all the messages.

Table 4.16 presents the spreading factor and the channel used by the end-device, on the scenario with the frequency-hopping disabled. The table also presents the number of channels available for the frequency-hopping mechanism.

Table 4.16: Values used for frequency-hopping mechanism evaluation.

Spreading factor	SF12
Channel jammed	Channel 0 (868.1MHz)
Frequency-hopping number of channels	8

Data Analysis

Table 4.17 shows the number of received messages with, and without the frequency-hopping mechanism. The number of received messages corresponds directly to success rate as a percentage.

Table 4.17: Number of received messages with, and without the frequency-hopping mechanism.

	Frequency-hopping enabled	Frequency-hopping disabled
Test 1	36	0
Test 2	39	0
Test 3	45	0
Test 4	22	0
Test 5	53	0
Average	39 ($\pm$ 17)	0 ($\pm$ 0)

Table 4.17 shows that the success rate in the scenario with frequency-hopping enabled is higher than the success rate with the frequency-hopping disabled. Nevertheless, 39% success rate is low. This can be solved by implementing a mechanism to detect jammers, allowing the gateway to avoid those channels. This improvement is left for future work.

Chapter 5

Conclusions

This dissertation addresses the problem of LoRaWAN implementation in small IoT installations, such as sensor networks for agriculture monitoring and control. To solve this problem, a low-cost gateway architecture is proposed. In order to improve the system resiliency and quality of service, the proposal also entails the development of time-slot reservation and frequency-hopping mechanism, which are identified in the literature as needed improvements to the LoRaWAN specification. Time-slot reservation was proposed to ensure that priority data is sent without danger of collisions. Frequency-hopping was proposed to avoid interference and jamming.

Several LPWAN technologies were analyzed with special focus on LoRa and LoRaWAN which were the selected for this proposal. Compliance with the current LoRaWAN specification was kept, whenever possible.

The gateway has been tested regarding its performance. The performance tests address the communications range, the randomness of the frequency-hopping scheme, as well as the performance improvements introduced by the time-slot reservation and frequency-hopping schemes.

Tests made with the gateway also shown that it can provide communications with line of sight within at least 4.9 kilometers. These tests have also shown that, for distances lower than 1 kilometer, communications can be done through small obstacles like trees.

Results made to both extensions made show that their use solve the problems they were implemented for.

Time-slot reservation turned out to be effective improving the packet delivery rate when more than two end nodes are concurrently transmitting to the gateway. For this situation, sending a message on a reserved time-slot offers a 22.2% higher success rate, which is expected to be even higher as the number of contending devices increases.

Frequency-hopping results show that, whenever using frequency-hopping and there is a jammer affecting one of the two available channels, the success rate is 39%.

5.1 Future Work

As mentioned before, whenever using frequency-hopping and there is a jammer, the success rates are just 39%. To improve this, a mechanism could be developed to detect the jammed channels, allowing the gateway to avoid them.

The number of channels the gateway is listening to is limited by the number of LoRa modules used. If the number of end-devices is too large, so that the gateway can not receive all the messages, more LoRa modules can be added. This can increase the cost the low-cost gateway, but one of the many LoRa modules from the end-devices can be used.

The developed time-slot reserving mechanism is not flexible. Only 12 time-slots can be reserved and the cycles have always the same length. An improvement could be done in a way that would be possible to reserve a time-slot at any time, and not just at the end of a 240 seconds cycle.

Bibliography

[1] LoRa Alliance. (2015). A technical overview of LoRa and LoRaWAN, (November), 1–20. Retrieved from https://www.lora-alliance.org/portals/0/documents/whitepapers/LoRaWAN101.pdf

[2] LoRa Alliance. (2016) Retrieved from https://www.lora-alliance.org/

[3] Huawei. (2015). NB-IOT White Paper. Manual. https://doi.org/10.1016/S0022-3913(12)00047-9

[4] Pinto, S. (2009). Redes Celulares. published by FCA.

[5] Ericsson. (2016). NB-IoT: a sustainable technology for connecting billions of devices.

[6] Sensing as-a-Service - The New Internet of Things (IOT) Business Model Retrieved from http://www.slideshare.net/mazlan1/sensing-asaservice-the-new-internet-of-things-iot-business-model

[7] Pham, C. (2016). Low-cost LoRa gateway : a step-by-step tutorial http://cpham.perso.univ-pau.fr/LORA/RPIgateway.html

[8] Link Labs: Low Power Wide Area Networks (LPWAN) and LTE-M for IoT (last access: December 2016). Retrieved from https://www.link-labs.com/

[9] MultiConnect® Conduit™: Programmable Gateway for the Internet of Things (last access: December 2016). Retrieved from http://www.multitech.co.uk/brands/multiconnect-conduit

[10] Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia, J. (2016). Understanding the limits of LoRaWAN, 8–12.

[11] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys and Tutorials, 17(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

[12] Long Range Communication For IoT Applications (last access: December 2016) Retrieved from http://iis-projects.ee.ethz.ch

[13] Sigfox - The Global Communications Service Provider for the Internet of Things (last access: December 2016) Retrieved from https://www.sigfox.com/en

[14] Raspberry Pi - Teach, Learn, and Make with Raspberry Pi (last access: December 2016) Retrieved from https://www.raspberrypi.org/

[15] Arduino (last access: December 2016) Retrieved from https://www.arduino.cc/

[16] SX1272 Long Range, Low Power RF Transceiver 860-1000MHz with LoRa Technology — Semtech (last access: March 2017) Retrieved from http://www.semtech.com/wireless-rf/rf-transceivers/sx1272/

[17] The Things Network Wiki (last access: March 2017) Retrieved from https://www.thethingsnetwork.org/wiki/

[18] Arduino shows off LoRa gateway and node shields (last access: May 2017) Retrieved from http://linuxgizmos.com/arduino-shows-off-lora-gateway-and-node-shields/

[19] NarrowBand IOT (last access: June 2017) Retrieved from https://en.wikipedia.org/wiki/NarrowBand_IOT

[20] Serial Peripheral Interface Bus (last access: August 2017) Retrieved from https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[21] Random.org - Statistical Analysis (last access: August 2017) Retrieved from https://www.random.org/analysis/

[22] PIONEER PROJECT BRINGS TOGETHER NOS, EDP DISTRIBUIÇÃO, HUAWEI, JANZ CEAND U-BLOX (last access: July 2017) Retrieved from http://www.techmezine.com/top-10-news/pioneer-project-brings-together-nos-edp-distribuicao-huawei-janz-ceand-u-blox/

[23] RN2483 - Wireless Modules (last access: September 2017) Retrieved from http://www.microchip.com/en/RN2483

[24] Fresnel zone (last access: October 2017) Retrieved from https://en.wikipedia.org/wiki/Fresnel_zone