Technical University of Denmark

IoT and tracking capabilities in LPWANs

Bernat Carbonés Fargas

Master's Thesis

February 2017

DTU Fotonik, Technical University of Denmark

Supervised by:

Martin Nordal Petersen

DTU Fotonik
Department of Photonics Engineering

IoT and tracking capabilities in LPWANs

Bernat Carbonés Fargas
Master's Thesis

February 2017

DTU Fotonik, Technical University of Denmark

Supervised by:

Martin Nordal Petersen

Title of the thesis:
IoT and tracking capabilities in LPWANs

Author:
Bernat Carbonés Fargas

Supervisor:
Martin Nordal Petersen

This report is a part of the requirements to achieve the Master of Science (MSc) in Telecommunication at Technical University of Denmark.

The report represents 30 ECTS points.

Department of Photonics Engineering
Technical University of Denmark
Ørsteds Plads, building 343
DK-2800 Kgs. Lyngby
Denmark

www.fotonik.dtu.dk
Tel: (+45) 45 25 63 52
Fax: (+45) 45 93 65 81

E-mail: info@fotonik.dtu.dk

Abstract

Internet of Things (IoT) has been growing over the last few years in multiple applications and due to the high consumption of the GPS for tracking capabilities, an innovative opportunity arises.

This thesis aims to design and implement a tracking IoT system in a LPWAN which is capable of transmitting the current position using low power technologies such as LoRa.

The whole system consisted of an end-node, four gateways, a server and a java application to store the obtained data in a MySQL database. A GPS was also used in the end-node to obtain the error in the estimated position computed by the multilateration algorithm. The data was analyzed in Matlab to present results of the accuracy.

Acknowledgements

Many people have contributed to the realization of this project. First of all, I would like to thank my supervisor Martin Nordal Petersen for giving me the opportunity to work in a ground-breaking field and helping me with my stay in Denmark. Thank you for your time and dedication.

Secondly, I would also like to thank Søren Manicus, Rune Domsten and Patrik Särenfors from IndesmaTech for providing the Kerlink gateways and giving me advice during the project.

Finally, I wish to acknowledge my parents, my sister, my girlfriend and my friends for their support and patience.

Table of contents

Abstract **5**

Acknowledgements **6**

Table of contents **8**

Acronyms **10**

List of Figures **11**

List of Tables **13**

1 Introduction **14**

1.1 Motivation *15*

1.2 State of the Art *15*

1.3 Methodology *18*

1.4 Thesis outline *18*

2 Overview of IoT technologies **19**

2.1 SIGFOX vs. LoRa *19*

2.2 LoRaWAN *21*

3 System design **25**

3.1 End-node *25*

3.1.1 Waspmote 26

3.1.2 GPS receiver 26

3.1.3 LoRaWAN module 27

3.2 Gateways *28*

3.3 Server *30*

3.4 Application *32*

3.4.1 Java application 32

3.4.2 MySQL Database 32

4 Algorithm **34**

4.1 Overview of geolocation techniques *34*

4.2 Algorithm Structure *35*

4.3 Extraction of TDOA *36*

4.4 Detection of Outliers *36*

4.5 Non-iterative algorithm *37*
4.5.1 Coordinates transformation 37
4.5.2 Linear multilateration 40
4.6 Iterative algorithm *41*
4.6.1 Haversine formula 41
4.6.2 Non-linear multilateration 43
5 Tests and results 45
5.1 Test spots *45*
5.2 Extraction of TDOA *46*
5.3 Detection of Outliers *47*
5.4 Non-iterative algorithm *49*
5.5 Iterative algorithm *50*
5.6 Required samples *51*
5.7 Required time *53*
6 Conclusions 56
6.1 Future work *57*
7 References 58
8 Appendix A. Linear multilateration algorithm 60
9 Appendix B, C, D, E, F 62

Acronyms

ABP	Activation By Personalization
AES	Advanced Encryption Standard
AWS	Amazon Web Services
BPSK	Binary Phase Shift Keying
CSS	Chip Spread Spectrum
ESD	Extreme Studentized Deviate
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IDE	Integrated Development Environment
IoT	Internet of Things
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network (protocol)
LPWAN	Low Power Wide Area Network
MALE	Mean Absolute Localization Error
MQTT	Message Queuing Telemetry Transport
OTAA	Over The Air Activation
RSSI	Received Signal Strength Indicator
SSH	Secure Shell
TDOA	Time Difference Of Arrival
TOA	Time Of Arrival
TOF	Time Of Flight
TTN	The Things Network
UART	Universal Asynchronous Receiver Transmitter
UNB	Ultra-Narrow Band
WSN	Wireless Sensor Network

List of Figures

Figure 1.1 Ericsson's estimation of connected devices [5] 14
Figure 1.2 Power consumption breakdown of a wildCENSE node [9] 16
Figure 1.3 Sherlock bar (a) 16
Figure 1.4 Sherlock bar (b) 16
Figure 1.5 Spylamb tracker (715 DKK) 17
Figure 1.6 Seatpost tracker (715 DKK) 17
Figure 1.7 Top cap tracker (845 DKK) 17
Figure 1.8 Connected Cycle pedal 17
Figure 2.1 LoRaWAN stack [3] 21
Figure 2.2 Receive windows Class A [3] 22
Figure 2.3 Application and session keys 24
Figure 2.4 Encryption of the payload 24
Figure 3.1 Elements of the whole system 25
Figure 3.2 GPS coordinates in the payload 26
Figure 3.3 GPS receiver 27
Figure 3.4 GPGLL output format 27
Figure 3.5 Waspmote with the LoRaWAN module 28
Figure 3.6 rxpk JSON [19] 29
Figure 3.7 TTN website interface 31
Figure 3.8 TTN website interface with JSON 31
Figure 3.9 MySQL Database. *gateway_data* table example 32
Figure 3.10 UML Class Diagram 33
Figure 4.1 Trilateration 34
Figure 4.2 Multilateration 34
Figure 4.3 Algorithm structure 35
Figure 4.4 Non-iterative algorithm structure 37
Figure 4.5 Geoid surfaces 38
Figure 4.6 Geodetic and Cartesian coordinates 38

Figure 4.7 Scenario of the algorithm 40
Figure 4.8 Distances over the globe 42
Figure 4.9 Distances over a 2D projection 42
Figure 4.10 Iterative algorithm structure 43
Figure 5.1 Map of the gateways and spots 45
Figure 5.2 TDOA extraction 46
Figure 5.3 Probability distribution of TDOA 47
Figure 5.4 Average TDOA vs alpha 47
Figure 5.5 TDOA extraction without outliers 48
Figure 5.6 Probability distribution of TDOA without outliers 48
Figure 5.7 Grid area iterative algorithm 50
Figure 5.8 Standard normal distribution 52
Figure 7.1 Scenario of the algorithm 60

List of Tables

Table 2.1 Comparison between SIGFOX and LoRa 20
Table 5.1 Alpha values 48
Table 5.2 Non-iterative algorithm MALE 49
Table 5.3 Non-iterative algorithm ALE. Sample mean. 49
Table 5.4 Iterative algorithm MALE 50
Table 5.5 Iterative algorithm ALE. Sample mean. 51
Table 5.6 Number of required samples 53
Table 5.7 Number of required sample without outliers 53
Table 5.8 Values transmission 54
Table 5.9 Required time 55
Table 5.10 Required time without outliers 55

1 INTRODUCTION

Low Power Wide Area Networks (LPWANs) are becoming more popular due to the growth of Internet of Things (IoT) technologies such as LoRa or SIGFOX. LPWANs are mainly characterized by three features [1]:

- The long range that can be reached, up to 50 km in rural areas and 10 km in urban areas, when the SIGFOX technology is used [2]. The sensitivity of the receivers (-130 dBm) is higher than that of traditional wireless technologies (between -90 dBm and -110 dBm). Consequently, the total path of the link budget could reach up to 160 dB.
- The low data rate, which usually ranges from 10 bps to 50 kbps depending on the deployed technology. Therefore, the maximum size of the payload that can be transmitted is around 256 bytes in order to fulfil the duty cycle [3].
- The low power consumption: the end-nodes only use the network when data has to be transmitted, which results in a very long battery lifetime (using a 2.5 Ah battery) of up to 10 years.

The use of IoT in LPWANs is a powerful system that is currently implemented in a wide variety of fields, such as medical and manufacturing sectors, sport activities, scientific studies or home automation, mainly driven by the low cost of the devices.

According to a BI Intelligence report, 34 billion devices will be connected to the internet by 2020, which is more than 4 devices for every human on earth, since the global population is estimated to be 8 billion by then [4].

Another report by Ericsson presents a similar forecast and predicts that there will be around 28 billion connected devices in 2021. IoT devices are expected to increase at a CAGR (Compounded Annual Growth Rate) of 23% from 2015 to 2021 due to new use cases. Cellular IoT devices are expected to have the highest growth as a result of the interest of the telecom companies and also due to 3GPP standardization of cellular IoT. This type of end-nodes has some advantages over the non-cellular one: for example, in terms of security issues and device management [5].

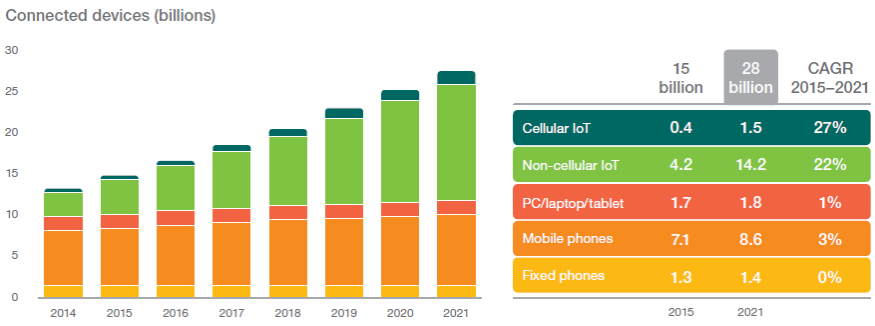

Figure 1.1 Ericsson's estimation of connected devices [5]

1.1 Motivation

In the next few years, more than half of IoT applications, like tracking devices or smart cities, will need geolocation. However, up until now only GPS is available, whose features are different from LoRaWAN technology.

Nowadays, there are cheap GPS receivers available on the market, but their main problem is their battery lifetime. These devices run out of energy quickly, so in a few days they need to be recharged. The current consumption of a GPS receiver is about 30-50 mA, far from the energy required in IoT devices. For instance, a LoRaWAN module working at a 868 MHz band consumes 2.8 mA, 38.9 mA and 14.2 mA in the "on" state, transmitting data and receiving data, respectively [6].

Therefore, the idea of the project arises from the need of designing a low-power consumption system capable of transmitting the location without GPS. At this point, the IoT technology plays an important role since it is also considered to be the next revolution in communications.

A low-cost device capable of sending the current position with a long battery lifetime would be useful for many of applications in everyday life. For example, a lot of bicycles are being stolen every day around the world: without having to recharge the battery every 2 or 3 days, it would be possible to track them. Moreover, parents could use this device to watch their children when they are playing outside. Tracking pets like dogs, cats or even birds could be another application for the system. Finally, it would also be useful to take care of elderly people who no longer fend for themselves.

1.2 State of the Art

Researchers are increasingly studying new tracking systems to be applied mainly on farming and animal monitoring. Studies and reports have been published over the last years showing several techniques, but all of them need the GPS to transmit the location.

GNSS collars were used for animal geolocation and also to monitor their behavior, health and complementary information. In order to face the high-energy consumption, the positions of the animals were acquired only hourly, so the autonomy was better, up to 7 months. SIGFOX was the IoT technology used to transmit the information from the GNSS collars due to its great features such as long range, low rate and low power consumption. A user platform was also used, composed of a MySQL database, a Tomcat server, the SIGFOX backend, a set of Java web services and an HTML5 web interface with maps [7].

In other experiments to monitor cattle behavior in the environment, GPS collars are combined with satellite images in a Wireless Sensor Network (WSN). Therefore, the location and the behavioral preferences are acquired by GPS and remote sensing is used in order to understand the interactions with the landscape [8].

The system called wildCENSE was developed mainly to track deer using different kinds of sensors (WSN): a light sensor, a temperature sensor, a humidity sensor, an accelerometer and a GPS receiver. An XBEE transceiver was used for communication and a Li-ion battery with a solar rechargeable circuit was implemented to solve the battery lifetime issue [9]. Considering that each node took measurements every 3 hours, Figure 1.2 illustrates the power requirements of various components of the node. Almost half of the power (43.65%) was drained by the GPS receiver. This clearly shows that removing the GPS from the system would reduce drastically the consumption of the whole device.

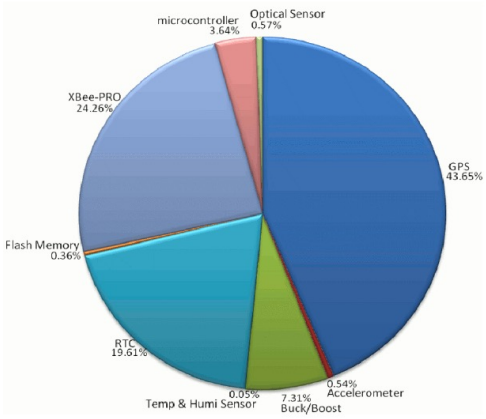

Figure 1.2 Power consumption breakdown of a wildCENSE node [9]

Regarding the research in another field, for example the theft of bicycles, there are some products on the market offering the possibility to track your bicycle. For instance, *Sherlock*, which is a small and compact bar that fits to handlebars of all shapes. It uses a GPS and is connected to a mobile app that tracks the bicycle in real-time. According to the company, the battery lasts up to one week on a single charge. It also contains a gyroscope to detect movement of the bicycle. This "invisible GPS" is still in test phase and it will be released in the next months [10].

Figure 1.3 Sherlock bar (a)

Figure 1.4 Sherlock bar (b)

Spybike is already on the market and offers three different kinds of trackers, depending on the shape (see Figure 1.5, 1.6 and 1.7). The embedded anti-theft GPS has an accuracy of 5-25 meters and uses the GSM band to send SMS messages. A pre-pay SIM card is included with a monthly cost of 30 DKK. *Spybike* also offers a web platform to track the bicycle in real-time. The prices of the products start at 715 DKK [11].

Figure 1.7 Spylamb tracker (715 DKK)

Figure 1.6 Seatpost tracker (715 DKK)

Figure 1.5 Top cap tracker (845 DKK)

A French company called *Connected Cycle* is going to lunch "pedal" this year, an innovative product which records the speed, route, incline and calories burned on each trip. It also includes an accelerometer to detect movement and a GPS in order to display the user's position at any time. These sophisticated pedals do not need to be recharged thanks to its unique bike electric circuit [12].

Figure 1.8 Connected Cycle pedal

Other companies such as *Helios* [13] or *Lattis* [14] are selling products with similar features: a GPS receiver, an accelerometer or gyroscope, a smart battery and a GSM or Bluetooth connection. However, none of them implement IoT for tracking stolen bikes.

Some smart devices with embedded GPS receivers for elderly people and children are also available in the market, such as bracelets and watches.

Various studies about LoRa technology for geolocation have been carried out in recent years, in which Sagemcom obtained good accuracy results (up to 4 meters). However, 42 gateways were used in a hexagonal layout to improve the results. The signal from the end-node was received by at least 10 gateways, when the data rate was at the highest. [15]

1.3 Methodology

The methodology used in this project is based mainly on experimental work. A brief research was conducted at the beginning to be aware of the state of the art in tracking devices and geolocation techniques.

The scheme of the system has continuously changed due to the limitations, advantages and drawbacks in each part of the project, which are described in the following pages.

The application was programmed in Java with the software Netbeans and the data was stored in a MySQL database. Both programs were used because I was already familiarized with them.

The results on accuracy were obtained using the software MATLAB, since it is a powerful data processing tool. All MATLAB code can be found in Appendix E.

1.4 Thesis outline

The structure of this document is organized as follows. Chapter 2 presents an overview of the main IoT technologies: LoRa and SIGFOX. It includes the reasons for choosing LoRa, as well as a description of the LoRaWAN protocol.

In chapter 3, the entire tracking IoT system is explained: the end-node, the gateways, the server and the third-party application.

Chapter 4 explains the two different designed algorithms to estimate the position: a non-iterative technique and an iterative one, both based on multilateration. A method for detecting outliers in the observed data is also analyzed beforehand.

Chapter 5 presents the results obtained from the algorithms explained in the previous chapter: for instance, the localization accuracy and the number of samples needed to achieve such accuracy in a confidence interval.

Finally, chapter 6 summarizes the conclusions of the project and suggests some lines of future work.

2 OVERVIEW OF IOT TECHNOLOGIES

IoT technologies in LPWANs are growing and improving fast, and they are currently present in the market: Nwave, LTE-M or Weightless-N are some examples. However, SIGFOX and LoRa are the two best-known technologies, with the highest deployment and with the best-defined features.

All information about the other technologies is not yet available and they will be launched in the next months. For this reason, SIGFOX and LoRa are analyzed taking into account the two main requirements of the project:

- Feasibility of tracking;
- Low power consumption in order to cope with the GPS.

2.1 SIGFOX vs. LoRa

Both technologies have their own infrastructure, so the user does not need to deploy new antennas over the region where the end-node is going to work. They also operate in the same unlicensed bands: 868 in Europe and 815 in the US. The topology of the network is the same as in all other IoT technologies: star. The central node is the gateway and the surrounding elements are the end-nodes.

One of the main differences between them is the bandwidth employed in the communication. SIGFOX uses an ultra-narrowband bandwidth (UNB), which allows for larger range since there is less noise in the channel. The noise spreads through the spectrum, so if the bandwidth is wide as the one employed in LoRa, the noise will be big as well. Regarding the payload, SIGFOX does not allow to transmit more than 12 bytes per packet, while LoRa can transmit up to 255 bytes [2] [3]. The data rate in LoRa (50 kbps) is also higher than the one in SIGFOX (100 bps), which allows to send more data in less time.

One of the main reasons why those technologies are low-power is the type of synchronization implemented. The devices only listen to the medium after each transmission, instead of continuously listen like other technologies do.

The main features for each technology are summarized in Table 2.1.

	SIGFOX	LoRa
Modulation	Ultra-Narrow Band BPSK	Chip Spread Spectrum modulation (CSS)
Bandwidth per channel	100Hz	EU: 125 kHz and 250 kHz US: 125 kHz and 500 kHz
Frequency Band	EU: Unlicensed 868 MHz US: Unlicensed 915 MHz	EU: Unlicensed 433, 868 MHz US: Unlicensed 915 MHz
Link budget	162 dB	155 dB
Data Rate	100 bps	From 250 bps to 50 kbps
Limitation msgs/day	140 msgs/day	Unlimited
Packet Size	12 bytes	Up to 255 bytes
Synchronization	Asynchronous	Asynchronous
Network characteristics:	Star	Star
Security issues	Frequency hopping and antireplay. No encryption	128 bits AES encryption
Open Source?	No	Yes
REST APIs? Servers?	REST API SIGFOX, SIGFOX servers, …	TTN, API Google Maps, …
Range	Rural: 30-50 km. Urban: 3-10 km	Rural: 10-15 km Urban: 3-5 km
Scalable	Yes	Yes
Feasibility of tracking	No information about it	Yes, some studies about it
Price	Low	Low

Table 2.1 Comparison between SIGFOX and LoRa

LoRa was selected as the IoT technology for the tracking system due to several reasons:

- **Range:** SIGFOX has a better range (from 10 to 15 km), but the range in LoRa is enough for the purpose of the project.
- **Open Source:** There is a lot of information about its implementation, layers, packet structures, protocols of communication and other features of LoRa.
- **Tracking capability:** Recent studies prove the possibility to perform geolocation in LoRa, as explained in section 1.2, while no information is found about SIGFOX.
- **Bandwidth:** The bandwidth in LoRa is bigger, so it is better to distinguish different paths from the same signal (useful for tracking capabilities in urban scenarios where reflections are present) [16].

2.2 LoRaWAN

First of all, it is important to distinguish the difference between LoRa and LoRaWAN. As a first approach, LoRa refers to the physical layer and LoRaWAN to the upper layers in the OSI model (Figure 2.1).

LoRa (Long Range) is a modulation based on a variation of chirp spread spectrum (CSS) with integrated forward error connection (FEC). Therefore, it uses the entire channel bandwidth to broadcast a signal, making it robust to the channel noise and resistant to multipath, fading and the Doppler effect, even at low power. In the CSS modulation, 6 spreading factors (SF) are defined, from SF = 7 to SF = 12, that ensure orthogonal transmissions at different data rates.

LoRaWAN is a media access control (MAC) protocol designed to allow low-powered devices to communicate with Internet connected applications over LPWAN. It is fully bidirectional and was architected by security experts to ensure reliability and safety. LoRaWAN can be mapped to the second and third layer of the OSI model. It is implemented on top of LoRa modulation. The LoRaWAN protocol is defined and standardized by the LoRa Alliance.

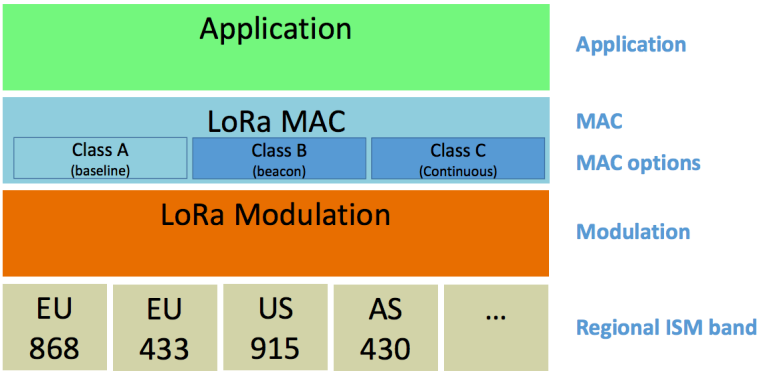

Figure 2.1 LoRaWAN stack [3]

At the medium access sublayer, LoRaWAN supports three types of devices according to their functionality:

- **Class A** devices use pure ALOHA access for the uplink, followed by two short downlink receive windows at predefined intervals (1 s and 2 s). Uplink messages can be sent at any time randomly. If the server does not respond in either of these receive windows, the next opportunity will be after the next uplink transmission from the device (Figure 2.2).
- **Class B** devices are synchronized using periodic beacons sent by the gateway to allow the schedule of receive windows for downlink messages from the server.
- **Class C** devices are always listening to the channel except when they transmit. Consequently, they consume much more energy than Class A devices.

Although the three classes are defined in the standardization, only class A must be implemented in all end-nodes.

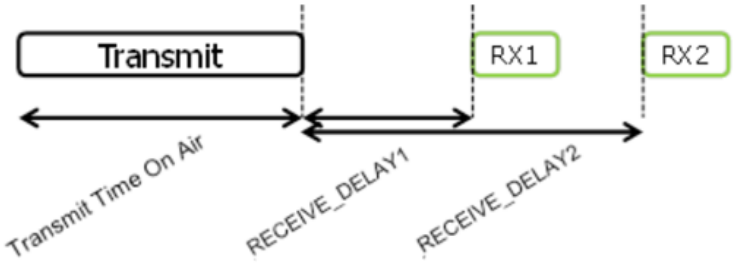

Figure 2.2 Receive windows Class A [3]

The network channels can be freely selected by the network operator. However, three default channels must be implemented in the EU 868MHz band. These channels are 868.1, 868.3 and 868.5 MHz.

The data rate depends on the bandwidth and the spreading factor. LoRaWAN can use channels with a bandwidth of 125 kHz, 250 kHz or 500 kHz depending on the region, as described in Table 2.1. The spreading factor is chosen by the end-device and influences the amount of time it takes to transmit a symbol, as can be seen in the following equation:

$$T_{sym} = \frac{2^{SF}}{BW} \tag{2.1}$$

The value of the data rate has also an impact in the sensitivity of the device. Transmitting a low data rate, allows a higher range. However, end-nodes can transmit on any channel available at any time fulfilling the duty cycle specific for each sub-band (1% or 0.1% in Europe). This also apples to gateways. The European

Telecommunications Standard Institute (ETSI) imposes such frequency specifications. This intends to make the network smarter in scheduling messages on gateways that are less busy or on channels that have a higher duty cycle.

The duty cycle is defined as the percentage of time in which the channel can be occupied:

$$d = \frac{T_{air}}{T_{air} + T_{off}} \tag{2.2}$$

For instance, for a duty cycle (d) of 1% and a time required to transmit a packet (time on air, T_{air}) of 100 ms, the channel will be unavailable (T_{off}) for 9.9 s.

Therefore, the data rate is a trade-off between range and duty cycle. If the data rate is low, packets will be in the air for a longer time and consequently the channel will be unavailable for a longer time, but the range will be larger.

In order to maximize the battery life of end-nodes and overall network capacity, the LoRaWAN network infrastructure manages the data rate and RF output for each end-device individually by means of an ADR (Adaptive Data Rate) algorithm.

ETSI also restricts the maximum allowed power that can be transmitted in the unlicensed band of 868 MHz to 14 dBm.

There are several identifiers and keys that are exchanged during the activation procedure of an end-node:

- **Device EUI**: The 64-bit device identifier. IEEE EUI64 address space that uniquely identifies the end-device.
- **Device Address:** 32 bits to identify the device in the joined network (not unique).
- **Application EUI:** Global application ID in IEEE EUI64 address space that uniquely identifies the application provider of the end-device.
- **Application Key:** Identifier used to derive the next two session keys during the activation procedure.
- **Application Session Key:** 128-bit key, which ensures end-to-end security on the application level and encrypts the payload. It is sent by the server (Figure 2.4).
- **Network Session Key:** 128-bit key, which ensures security on the network level, encrypts messages and is also sent by the server.

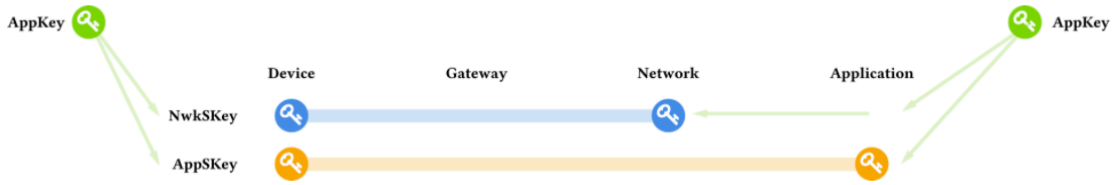

Figure 2.3 Application and session keys

An end-node willing to join a LoRaWAN network has to be personalized and activated. The activation can be achieved in two ways:

- **Over-The-Air (OTAA)**: This is the most secured way to connect to the network. Devices only set the application EUI and the application key. During the join procedure, a dynamic device address is assigned and the application and network session key are randomly generated by the server.
- **ABP (Activation By Personalization)**: The device address and the two session keys are directly stored in the end-device. Therefore, the end-node is equipped with the required information for participating in the LoRa network when started. This type of procedure can compromise the security of the communications.

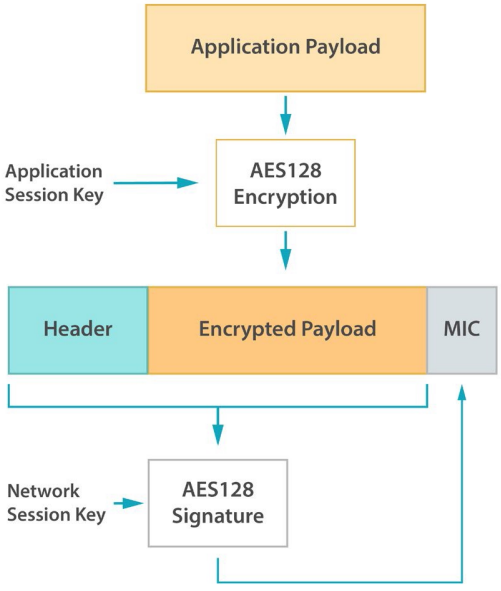

Figure 2.4 Encryption of the payload

3 SYSTEM DESIGN

The overall system consisted mainly of four blocks: an end-node or transmitter, four gateways, a server and a third-party application (Figure 3.1). It is important to notice that the algorithm could be applied also with three gateways instead of four, but the accuracy will be lower. The end-node sent the data over the air using the LoRaWAN protocol and the gateways that were close enough (around 5 km) received the data. Then, the gateways forwarded the packets via UDP/IP to the server, together with information from the received signal such as the exact time when the packet was received, the RSSI, the working frequency, etc. Afterwards, the server processed the data from the different gateways and routed the messages to the application using a MQTT client. Finally, the algorithm to estimate the position was applied in the third-party application.

The algorithm is explained in more detail in the next section (4. Algorithm).

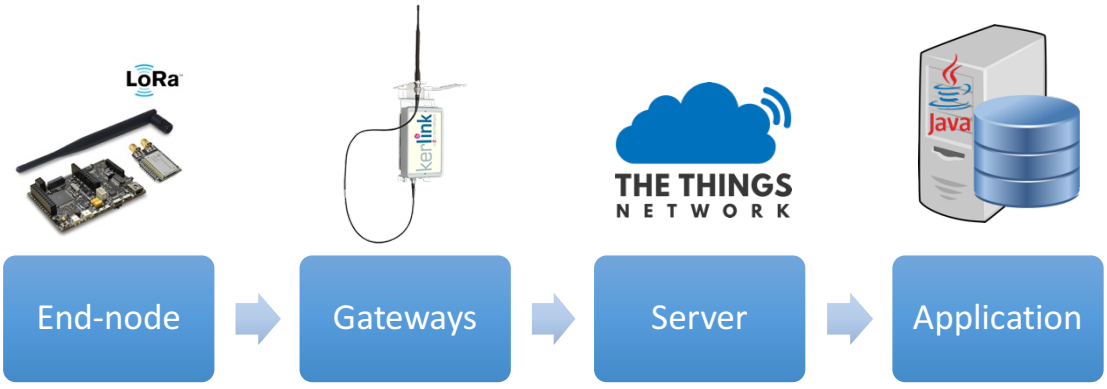

Figure 3.1 Elements of the whole system

3.1 End-node

The end-node was responsible for sending the data acquired from a GPS receiver over the air through a LoRaWAN module.

The code implemented in the end-node can be found in Appendix B.

3.1.1 Waspmote

The Waspmote (v1.2) was the main device of the end-node, so it processed and gathered information from the other modules. The Waspmote's core is based on the Atmel ATmega1281 microcontroller and is suitable for IoT applications. The modular architecture allows to integrate only the modules needed in each situation.

Waspmote was selected as the core processor because of its low power consumption, which was one of the main requirements of the system. The feasibility for allowing to connect a LoRaWAN module was also relevant for choosing this device.

The Waspmote IDE, provided by the Libelium company, was used for writing code and uploading it to the Waspmote board. All libraries needed for the software compilation are in the IDE. The program structure is always divided into two basic parts with sequential execution: setup and loop. The setup is executed once, when the code is initialized. At this point, the GPS receiver and the LoRaWAN module were initialized. The part named "loop" runs continuously, forming an infinite loop. In this part, GPS coordinates were received and such information was transmitted approximately every 10 seconds over the LoRaWAN. Therefore, the Waspmote parsed the coordinates and transmitted them as a payload in the packet.

The LoRaWAN module only allows to transmit hexadecimal characters, so 11 bytes were sent in each packet with the following structure:

- 5 bytes for the latitude without commas.
- 4 bits for the direction of the latitude: North (= 1) or South (= 2).
- 5 bytes for the longitude without commas.
- 4 bits for the direction of the longitude: East (= 1) or West (= 2).

The next figure shows an example of how the matching between the received GPS coordinates and the transmitted payload would be.

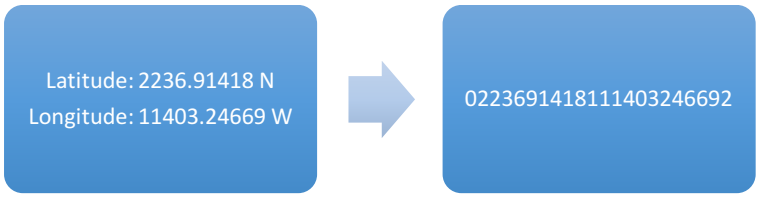

Figure 3.2 GPS coordinates in the payload

3.1.2 GPS receiver

The main function of the GPS receiver was to obtain the current position of the end-node, that is, to acquire the latitude and longitude coordinates.

All features and technical parameters can be read in the datasheet [17].

Figure 3.3 GPS receiver

The NMEA0183 protocol to transmit all the gathered data is used by the GPS. As the only important data needed for the device was the latitude and longitude values, the GPGLL message was selected. The output format is the following one:

```
       1       2 3       4 5         6 7
       |       | |       | |         | |
$--GLL,llll.ll,a,yyyyy.yy,a,hhmmss.ss,A*hh

1) Latitude
2) N or S (North or South)
3) Longitude
4) E or W (East or West)
5) Time (UTC)
6) Status A - Data Valid, V - Data Invalid
7) Checksum
```

Figure 3.4 GPGLL output format

The connection between the Waspmote and the GPS was made through an asynchronous serial communication, UART. Therefore, just two wires were needed: one for transmitting data and another for receiving.

A custom library was designed to manage this connection, which can be read in the Appendix B.

3.1.3 LoRaWAN module

The LoRaWAN module transmitted the GPS coordinates over the air to be received by any gateway.

The following parameter configuration was used in order to initialize the LoRaWAN module:

- Setting the main identifiers for joining the network using the OTAA, like the device EUI (identifier of the end-node), the application EUI (identifier of the application) and the application key (used to exchange the network and application session keys).
- Configuration of the hardware parameters: maximum power (14 dBm), working frequency (868 MHz) and adaptive data rate to maximize the battery life and the

overall network capacity. The network server was managing the data rate depending on the conditions of the channel.

Figure 3.5 Waspmote with the LoRaWAN module

Two antennas were tested and the one with the largest gain was chosen in order to ensure a good coverage. The first one was the black antenna (Figure 3.5) and had a gain of 2.5 dB (4.5 dBi), whereas the other one was bigger and had a gain of 3 dB (5 dBi).

A custom library can also be found in the Appendix B together with the GPS and the Waspmote code, as mentioned above.

3.2 Gateways

The main function of the four gateways was to route the data received from the end-node to the server via UDP / IP. In order to estimate the location of the device, the received time of the packet from each gateway was needed to apply the multilateration algorithm.

There are several gateways supporting LoRa technology on the market. The chosen one was Kerlink because a GPS receiver is embedded in the gateway. Therefore, all gateways are synchronized by using the timestamp from the GPS satellites.

The protocol between the gateway and the server is set in a binary file called “packet forwarder” that runs inside the gateway. There is no authentication of the gateway nor the server, and the acknowledges are only used for network quality assessment, not to correct lost packets. Some types of packets are exchanged between the gateway and the server. Since the purpose of the thesis was not to explain the whole protocol, the overall description can be found in the references [18]. One of the packet exchanged contains the payload of the packet received from the end-node. This is transmitted in a JSON object called “rxpk” (Figure 3.6). The “time” field contains the information which is relevant for the algorithm. Furthermore, other information about the received signal is forwarded to the server: the data rate, the received power (RSSI), the signal-to-noise ratio (SNR) or the coding rate.

Name	Type	Function
time	string	UTC time of pkt RX, us precision, ISO 8601 'compact' format
tmst	number	Internal timestamp of "RX finished" event (32b unsigned)
freq	number	RX central frequency in MHz (unsigned float, Hz precision)
chan	number	Concentrator "IF" channel used for RX (unsigned integer)
rfch	number	Concentrator "RF chain" used for RX (unsigned integer)
stat	number	CRC status: 1 = OK, -1 = fail, 0 = no CRC
modu	string	Modulation identifier "LORA" or "FSK"
datr	string	LoRa datarate identifier (eg. SF12BW500)
datr	number	FSK datarate (unsigned, in bits per second)
codr	string	LoRa ECC coding rate identifier
rssi	number	RSSI in dBm (signed integer, 1 dB precision)
lsnr	number	Lora SNR ratio in dB (signed float, 0.1 dB precision)
size	number	RF packet payload size in bytes (unsigned integer)
data	string	Base64 encoded RF packet payload, padded

Figure 3.6 rxpk JSON [19]

At the beginning, the value set in the "time" field had microseconds accuracy, which means around 300 meters in distance. Consequently, in order to enhance the accuracy, the binary file was modified to have nanoseconds (0.3 meters):

```
/* Packet RX time (GPS based), 37 useful chars */
if (ref_ok == true) {
    /* convert packet timestamp to UTC absolute time */
    j = lgw_cnt2utc(local_ref, p->count_us, &pkt_utc_time);
    if (j == LGW_GPS_SUCCESS) {
        /* split the UNIX timestamp to its calendar components */
        x = gmtime(&(pkt_utc_time.tv_sec));
        j = snprintf((char *)(buff_up + buff_index), TX_BUFF_SIZE-buff_index, ",\"time\":
            \"%04i-%02i-%02iT%02i:%02i:%02i.%06liZ\"", (x->tm_year)+1900, (x->tm_mon)+1,
            x->tm_mday, x->tm_hour, x->tm_min, x->tm_sec, (pkt_utc_time.tv_nsec)/1000); /
            * ISO 8601 format */
        if (j > 0) {
            buff_index += j;
        } else {
            MSG("ERROR: [up] snprintf failed line %u\n", (__LINE__ - 4));
            exit(EXIT_FAILURE);
        }
    }
}
```

The *%06* field was changed to *%09* and the 1000 was removed from *(pkt_utc_time.tv_nsec)/1000*. This procedure was not easy, since all raw C code had to be cross-compiled on a Linux system using an ARM toolchain provided by Kerlink. The C code that provides the binary file and the steps on how to do the cross compilation can be found in a GitHub repository [19] and also in Appendix C. However, even solving this problem to have a better accuracy, the clock of the gateway is not fast enough to set an accurate time in the received packet. For this reason, in a few months Kerlink will launch a new version of gateways with enhanced timestamps, ideal for geolocation.

Two JSON configuration files inside the gateway contain several parameters to configure the RF transceiver and also to establish the connection with the server:

global.conf and *local.conf*. The program first looks for the *global.conf* and parses it, then searches for the next JSON file *local.conf*. If this file exists and some parameters are defined in both global and local configuration files, the local definition overwrites the global one. It includes the information about the subchannels in the 868 MHz band necessary for the communication with the end-node. Furthermore, the IP address of the server and also the ports (downlink and uplink) are in these files. The local configuration file should contain parameters that are specific to each gateway (MAC address, frequency for backhaul radio channels). In each configuration file, the program looks for a JSON object named *SX1301_conf*, which should contain the parameters for the Lora concentrator board (RF channels definition, modem parameters), and another JSON object called *gateway_conf*, which should contain the gateway parameters (gateway MAC address, IP address of the server, keep-alive time). An example of a JSON file is in the Appendix C as well.

A SSH connection was established to log into the Linux system, which is running inside the gateway in order to set the binary file and the two JSON configuration files mentioned before.

The gateway can be connected to the Internet in two different ways: with an Ethernet cable or with the embedded GSM module (a SIM card with a subscription needed). Some problems were experienced with the GSM module. When it is working with other gateways connected via Ethernet, their connection with the server is faster, so the packets received from the GSM module are discarded by the server because the delay is greater than 200 ms [20]. The server opens a window of 200 ms to receive all the packets with the same identifier starting when the first packet is received.

3.3 Server

The server was responsible for decoding data from the four gateways and transmitting it to the third-party application. It was important that the four packets from the different gateways, with the same payload but with different times, arrived at the server. The most relevant feature for the algorithm computation was the value of the time field in the JSON which was sent.

Several options were tried before making a choice. First of all, the Loriot server, which was a first approach to realize and learn how the data was collected by a LoRa server. The main problem of this server is the impossibility of setting the custom binary file in the gateway, since it is given by Loriot.

The second option was to set up an own LoRa network using the code designed by Petr Gotthard [21]. However, this server is unable to receive data from more than one packet because the deduplication is done just once.

Finally, the solution was The Things Network (TTN) server, since it is an open-source platform and allows a wide variety of third-parties to be connected. Furthermore, more people are joining to this community and it has been growing in the recent years.

Figures 3.7 and 3.8 display the interface in the TTN website when a device is registered and transmitting data. The *cntr* value is the identifier of the LoRa packet and the *payload* contains the GPS coordinates in the format explained in section 3.1.1. The interface is very useful to see the data that has been sent. However, if the goal is to process and analyze such information, a communication with a third-party is required. Thus, a MQTT client was used to forward the data to a java application.

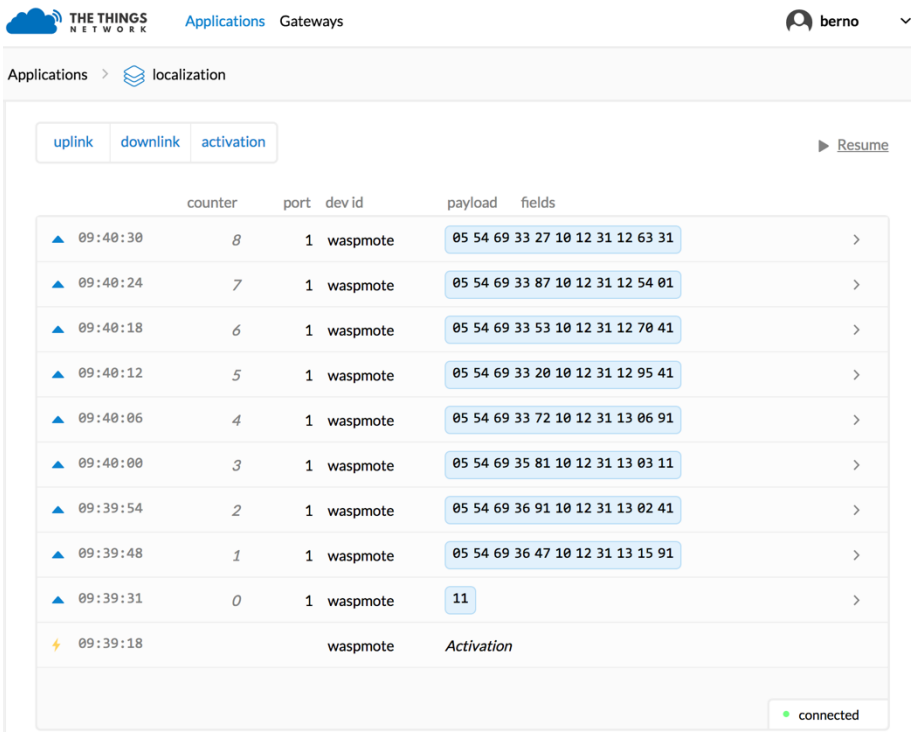

Figure 3.7 TTN website interface

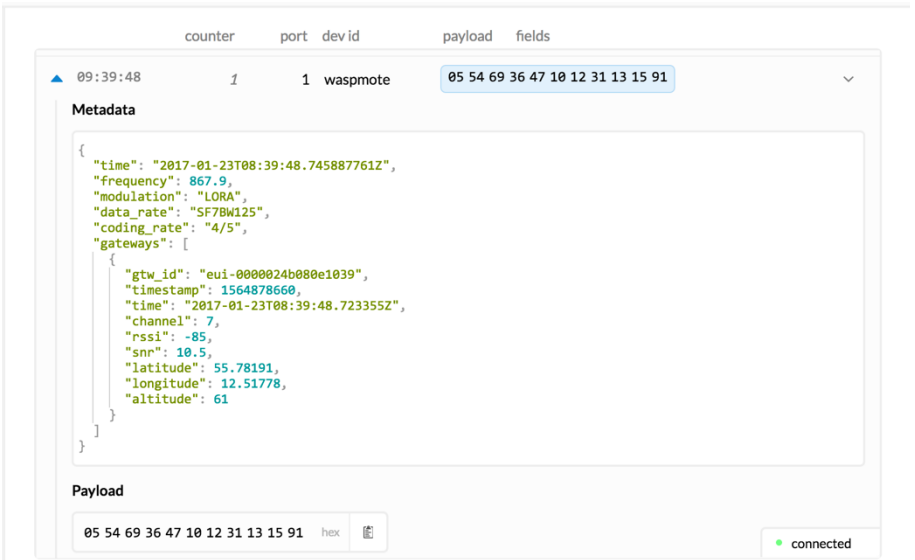

Figure 3.8 TTN website interface with JSON

3.4 Application

The third-party application consisted mainly of two parts: a Java application and a MySQL database. The main function was to obtain the data from the server, parse it and insert it in the database for processing in the following step.

3.4.1 Java application

As already stated, a MQTT client library [22] was used to establish the communication with the server. The MQTT is a machine-to-machine (M2M) connectivity protocol which is designed as an extremely lightweight publish and subscribe messaging transport. The TTN uses MQTT to publish device activations and messages, but also allows the user to publish a message for a specific device in response. Therefore, a subscription was done in the Java code to the desired topic with the suitable device identifier in order to obtain all packets from the gateways.

The object-oriented Java code was composed by several classes to parse and manage the data from the server, as well as a JDBC driver to establish connection with the MySQL database. Figure 3.10 displays the UML Class Diagram, which describes the structure of the Java application: classes, attributes, methods and relationships among the objects. All the code can be found in the Appendix D.

3.4.2 MySQL Database

Four tables were designed in the database in order to store all the data from the server:

- *uplink_packets*: All the packets received with the frequency, coding rate, data rate and modulation.
- *activations*: Activations done for the device together with the device address and the application eui.
- *uplink_data*: Raw data encoded in base 64 transmitted by the end-node.
- *gateway_data*: All the information from the gateways, including the time of the received packet and the GPS coordinates of the gateway.

The structure of the database and all the data inserted during the tests, can be found in Appendix D.

100% 20:93

Result Grid | Filter Rows: Search | Edit: | Export/Import: | Fetch rows:

id	id_uplink	gtw_id	timestamp	time	ns_time	channel	rssi	snr	latitude	longitude	altitude
33281	22072	eui-0000024b080e1039	1584438275	Mon Dec 19 12:20:38 CET 2016	739825260	4	-122	-7.5	55.78188	12.51783	58
33282	22072	eui-0000024b080e0ffc	1649924371	Mon Dec 19 12:20:38 CET 2016	739814260	4	-83	10.8	55.76842	12.45721	19
33283	22072	eui-0000024b080e0fdf	4007757227	Mon Dec 19 12:20:38 CET 2016	739818260	4	-116	5	55.78203	12.48078	67
33284	22072	eui-0000024b080e104c	4212951715	Mon Dec 19 12:20:38 CET 2016	739813260	4	-116	6.5	55.76842	12.47625	21
33472	22167	eui-0000024b080e1039	2749440603	Mon Dec 19 12:40:03 CET 2016	741554258	4	-123	-2.5	55.78191	12.51781	58
33473	22167	eui-0000024b080e0ffc	2814926619	Mon Dec 19 12:40:03 CET 2016	741554000	4	-111	8.2	55.76844	12.45719	21
33474	22167	eui-0000024b080e0fdf	877792251	Mon Dec 19 12:40:03 CET 2016	741552258	4	-116	1.2	55.78194	12.48049	94
33475	22167	eui-0000024b080e104c	1082986755	Mon Dec 19 12:40:03 CET 2016	741552000	4	-123	-5.2	55.76845	12.47629	25

Figure 3.9 MySQL Database. *gateway_data* table example

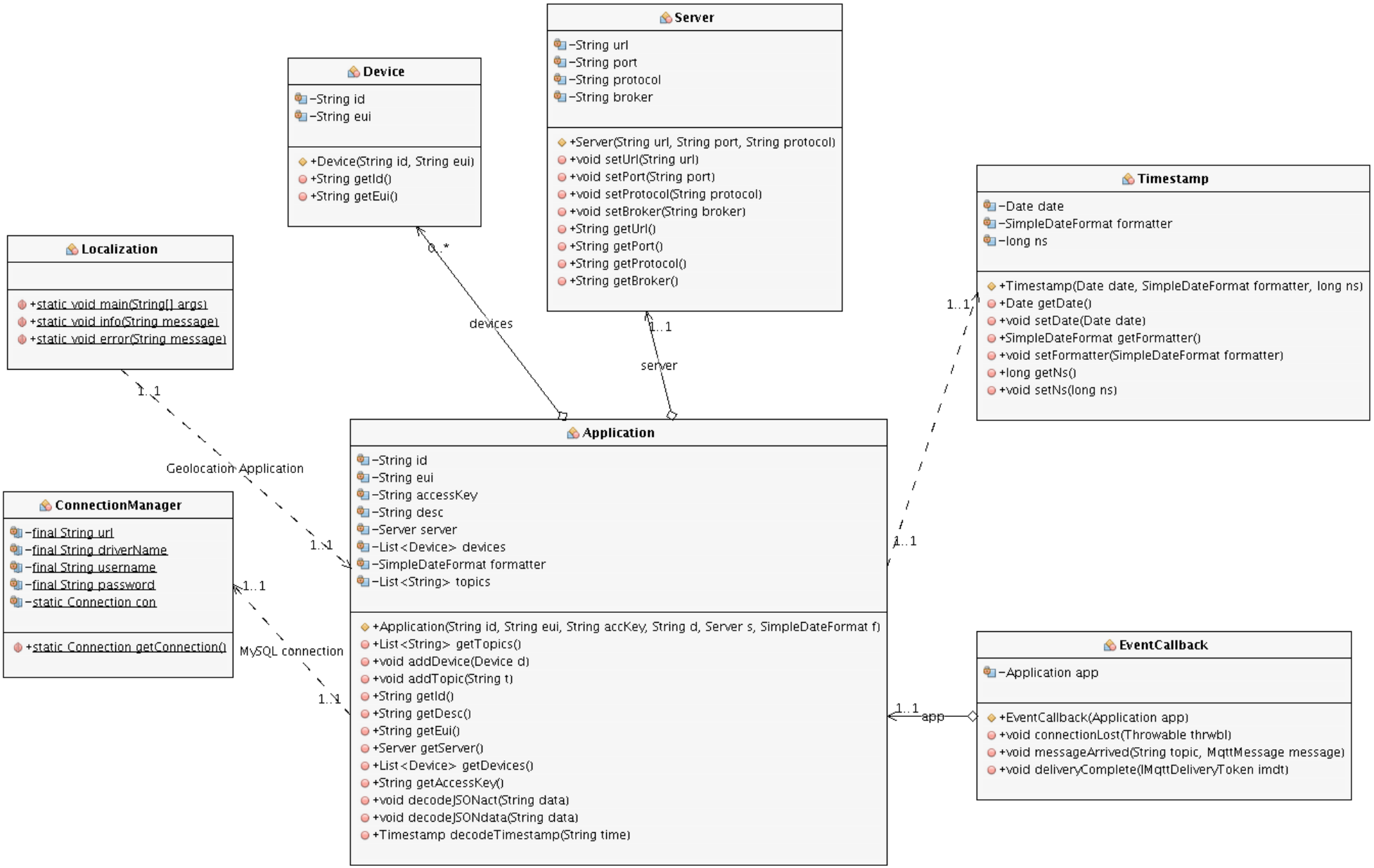

Figure 3.10 UML Class Diagram

4 ALGORITHM

4.1 Overview of geolocation techniques

There are several techniques which can be used to estimate the position of the device, each one of them with different features. It is important to select the most suitable one depending on the known information from the end-node. The three most common methods used for performing the geolocation are triangulation, trilateration and multilateration.

Triangulation uses angles of incidence of the signal received from the transmitter. A triangle is defined with two of them and the end-node position is estimated applying trigonometric formulas.

Trilateration (Figure 4.1) requires the distance between the transmitter and the receiver, which can be obtained from the time of arrival (TOA), the time of flight (TOF) or from the received signal strength indicator (RSSI). Therefore, it requires synchronization between the transmitter and the receiver. The position is the intersection of the three circles obtained from the different distances.

Multilateration (Figure 4.2) is quite similar to trilateration; however, the main feature to compute the location is the time difference of arrival (TDOA). The transmitters are synchronized to each other, whereas the receiver does not need to be. Thus, the location in this technique is the intersection of at least two hyperbolas (three antennas required).

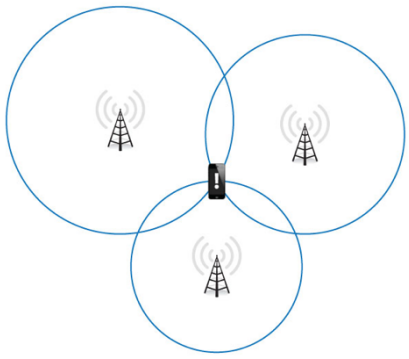

Figure 4.1 Trilateration

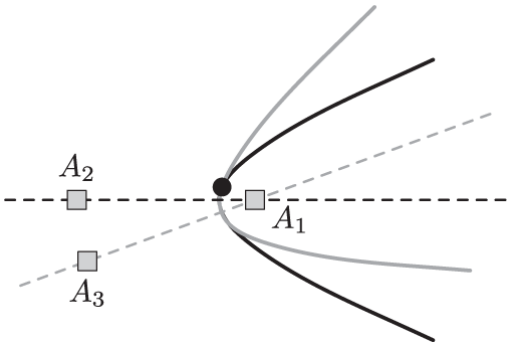

Figure 4.2 Multilateration

4.2 Algorithm Structure

The tracking IoT system did not have synchronization with the end-node, only the gateways were synchronized with each other. Therefore, the information available was the time when the packet was received by each gateway. The TDOA was computed with this information and, for this reason, the multilateration algorithm was chosen. The RSSI was also known, so trilateration could also be applied. However, recent studies demonstrate a better accuracy using TDOA instead of RSSI [15].

The algorithm consisted in several phases, as shown in Figure 4.3:

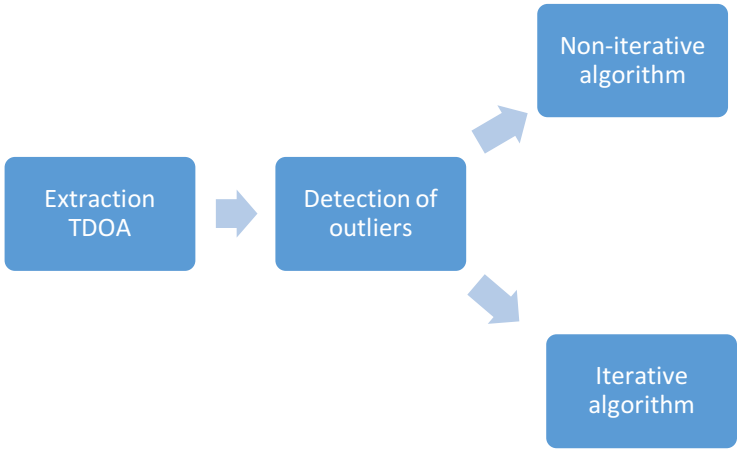

Figure 4.3 Algorithm structure

First of all, the calculation of the TDOA received from the four gateways. Then, outliers detection in the TDOA dataset due to erratic measurements, shadowing or multipath in the channel. Finally, the application of two different multilateration algorithms designed to estimate the position of the device: a non-iterative and an iterative algorithm. The techniques implemented in each block are explained in more detail in the following pages.

It is important to note that the detection of outliers was only applied in the static spots. As it is explained in the next section (5. Test and results), three static spots were tested separately. Statistical techniques can be used since a big dataset of samples from each location is compared with the actual one. However, in “real-time” calculation, only one sample is compared with the true one. Some approaches averaging with neighbor samples could be done, but this is discussed as future work at the end of the project.

4.3 Extraction of TDOA

The first step consisted in computing the TDOA based on a pair of UTC times. Therefore, once a packet was received from the end-node, four different UTC times (one from each gateway) were inserted into the database. Then, the TDOAs (t_{ij}) were calculated as follows:

$$t_{ij} = t_i - t_j \qquad \forall\, i, j = 1{:}4 \quad j \neq i \tag{4.1}$$

Therefore, twelve TDOAs were computed, but only six of them are useful since one is the the opposite from the other:

$$t_{ij} = -t_{ji} \tag{4.2}$$

4.4 Detection of Outliers

According to Barnett and Lewis [23], an outlier is defined as "an observation which appears to be inconsistent with the remainder of that set of data". Keeping outliers in a dataset can lead to wrong results, so it is important to detect the true outliers. However, in some cases it may not be possible to determine if an outlying point is bad data.

There are different methods to detect outliers. The first one is the Grubbs' test, which detects one outlier at a time assuming a normal distribution. The outlier is removed from the dataset and the test is iterated until no outliers are detected. However, it is suitable to detect a single outlier because if more outliers are present, a masking problem can occur. For instance, it is possible that there are two outliers, but they both increase the standard deviation so much that neither of them can be detected.

The second one is the Tietjen-Moore test, which is a generalization of the Grubbs' test to the case of multiple outliers. Nevertheless, it has a limitation: the number of outliers must be specified exactly.

Finally, the Generalized Extreme Studentized Deviate (ESD) test is also a generalization of the Grubbs' Test to the case of more than one outlier, but it does not require to know the number of them. This test only requires only an upper bound for the suspected number of outliers. Given the upper bound *r*, the Generalized ESD test performs *r* separate tests: a test for one outlier, a test for two outliers and so on up to *r* outliers.

As the number of outliers was unknown, the Generalized ESD was applied to detect them in the six TDOA obtained before. A MATLAB library was used [24].

4.5 Non-iterative algorithm

The non-iterative algorithm was based on using a linear multilateration technique to estimate the position. In order to do so, the following inputs were needed: the TDOAs and also the locations of the four gateways.

The position over the globe can be expressed in Cartesian coordinates (x, y, z) or in Geodetic coordinates (latitude, longitude and height). The first set of coordinates is useful for mathematical calculations and easier to manipulate, but not for providing understandable information. The second one provides understandable information, but it is useless for mathematical calculations. Therefore, the known Geodetic coordinates from the gateways could not be used directly into the algorithm and a transformation was needed.

The structure of the algorithm is shown in the next figure (Figure 4.4):

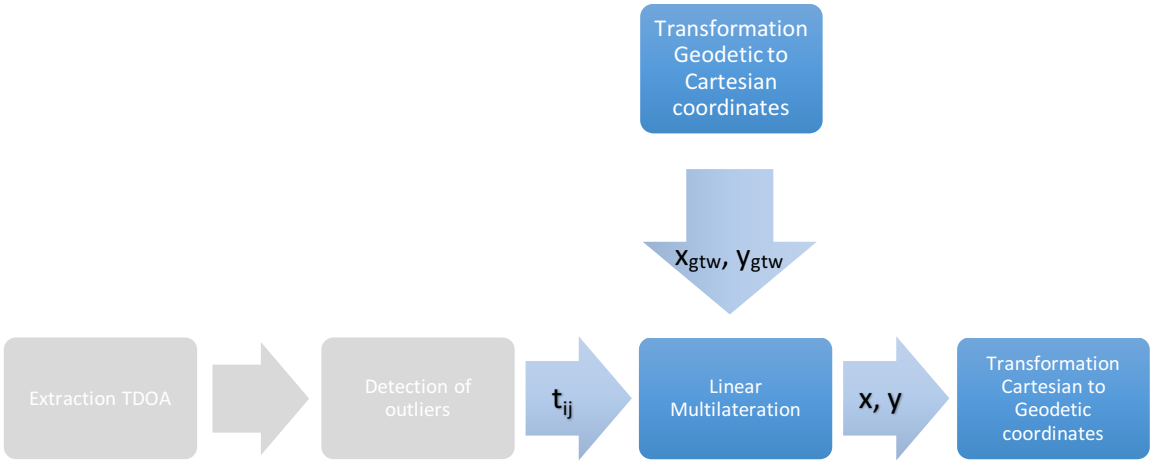

Figure 4.4 Non-iterative algorithm structure

4.5.1 Coordinates transformation

The shape of the Earth is determined by the mean sea level surface, which is called geoid. This represents an equipotential surface corresponding to an overall absolute elevation of 0 meters. As it is quite complex to represent, ellipsoids are used as a mathematical reference to express positions over the Earth in an accessible way. These positions are called geodetic coordinates. There are several ellipsoids depending on how the quadratic error with respect to the geoid has been minimized:

- Global minimization of standard deviation like WGS-84 used in GPS.
- Local minimization adapted to a particular country or continent like ED50 Europe, or NAD27 and NAD83 USA.

The geodetic coordinates are latitude, longitude and ellipsoidal height (see Figure 4.5 for more details about the height).

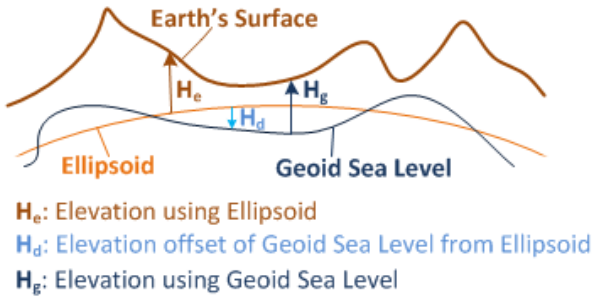

Figure 4.5 Geoid surfaces

Cartesian coordinates assume an origin at the Earth's center of mass and rotate with the Earth. This is called as the Earth-Centered Earth-Fixed (ECEF) or conventional terrestrial coordinate system. The Cartesian coordinates are x, y and z.

Figure 4.6 displays the Geodetic together with the Cartesian coordinates over the globe:

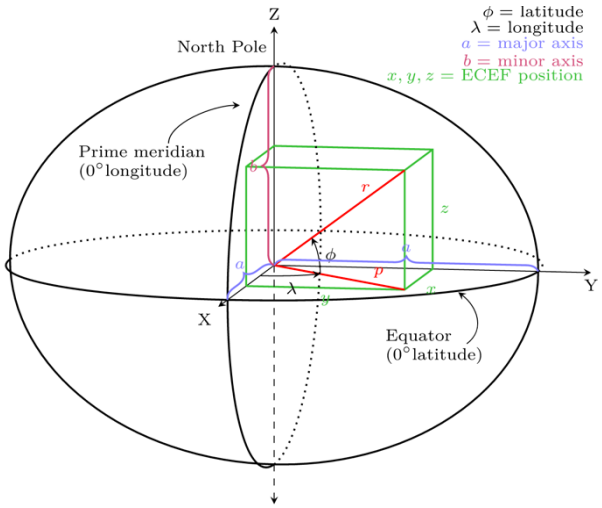

Figure 4.6 Geodetic and Cartesian coordinates

On the one hand, the transformation from Cartesian to Geodetic coordinates was attained as follows [25]:

$$\begin{aligned} x &= (v + h) * cos\varphi * cos\lambda \\ y &= (v + h) * cos\varphi * sin\lambda \\ z &= (v * (1 - e^2) + h) * sin\varphi \end{aligned} \tag{4.3}$$

Where ν is the radius of curvature in the primer vertical, h is the ellipsoidal height, e^2 is the first numeric eccentricity, φ is the longitude and λ is the latitude.

$$\nu = \frac{a}{\sqrt{1 - e^2 * sin^2\varphi}} \tag{4.4}$$

$$e^2 = \frac{a^2 - b^2}{a^2} \tag{4.5}$$

The ellipsoids are defined by their semi-major (a) and semi-minor axis (b). The WGS-84 was selected since all the coordinates from the gateways are referred to this ellipsoid and it is the one used in GPS. The values of the axis are:

$$\begin{aligned} a &= 6378137\ m \\ b &= 6356752{,}314\ m \end{aligned} \tag{4.6}$$

On the other hand, the reverse transformation does not produce a closed formula for the longitude. However, Bowring presented a closed one, which is the following [25]:

$$\lambda = tan^{-1}\frac{x}{y} \tag{4.7}$$

$$\varphi = tan^{-1}\frac{z + (e')^2 * b * sin^3\theta}{p - e^2 * a * cos^3\theta} \tag{4.8}$$

Where $(e')^2$ is the second numeric eccentricity, and θ and p are defined as:

$$(e')^2 = \frac{e^2}{1 - e^2} \tag{4.9}$$

$$\theta = tan^{-1}\frac{z * a}{p * b} \tag{4.10}$$

$$p = \sqrt{x^2 + y^2} \tag{4.11}$$

4.5.2 Linear multilateration

Once the coordinates from the gateways were transformed to Cartesian, the linear multilateration algorithm could be applied. The gateways coordinates were denoted as (x_i, y_i) and the unknown position of the end-device as (x, y). The scenario was the following:

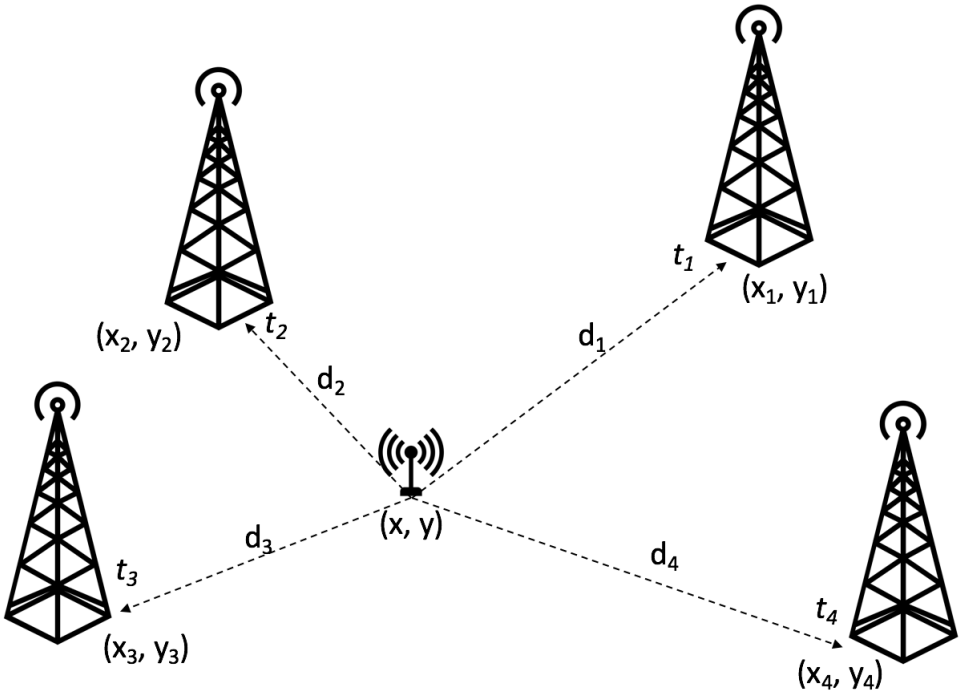

Figure 4.7 Scenario of the algorithm

The distances from each gateway to the end-node were defined as:

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \qquad \forall \, \mathrm{i} = 1{:}\,4 \tag{4.12}$$

In order to create the system of equations, the time of a gateway had to be selected as a reference. Then, the differential distances between the reference gateway and the others were:

$$D_{i1} = d_i - d_1 = c_0 * t_{i1} \qquad \forall \, \mathrm{i} = 2{:}\,4 \tag{4.13}$$

Where c_0 is the speed of the light: 299792458 m/s

In the end, after applying some mathematic techniques, the problem was to solve a system of linear equations, since four gateways were used. In the case of doing the algorithm with three gateways, the equations are non-linear. The entire procedure and demonstrations are in the Appendix A. Therefore, the system to resolve was:

$$\begin{pmatrix} -2*(x_2-x_1) & -2*(y_2-y_1) & -2*D_{21} \\ -2*(x_3-x_1) & -2*(y_3-y_1) & -2*D_{31} \\ -2*(x_4-x_1) & -2*(y_4-y_1) & -2*D_{41} \end{pmatrix} * \begin{pmatrix} x \\ y \\ d_1 \end{pmatrix} = \begin{pmatrix} -{x_2}^2 - {y_2}^2 + {D_{21}}^2 + {x_1}^2 + {y_1}^2 \\ -{x_3}^2 - {y_3}^2 + {D_{31}}^2 + {x_1}^2 + {y_1}^2 \\ -{x_4}^2 - {y_4}^2 + {D_{41}}^2 + {x_1}^2 + {y_1}^2 \end{pmatrix} \tag{4.14}$$

The unknown variables were x, y and d_1. Considering d_1 as unknown made the procedure to solve the problem easy (linear equations). This algorithm was applied four times, each one changing the reference gateway. Therefore, four different Cartesian coordinates were estimated and the optimum one was an average of the previous four, since the values were close.

Finally, the estimated position of the end-node (x, y) was transformed to Geodetic coordinates to compute the error and display it in a map.

4.6 Iterative algorithm

4.6.1 Haversine formula

The iterative algorithm arose from the possibility of committing errors in the calculation of distances using Cartesian coordinates in the previous method. There are two types of distances in navigation (Figure 4.8 and 4.9):

- Loxodrome or rhumb line: The path between two points which crosses meridians with the same angle. In a 2D projection (like Mercator), the distance is a straight line.
- Orthodrome or great circle: The shortest distance between two points over the globe.

The radio waves follow the shortest distance, the curvature of the Earth, so the "correct" way to compute the distances should be using the great circle one.

Figure 4.8 Distances over the globe

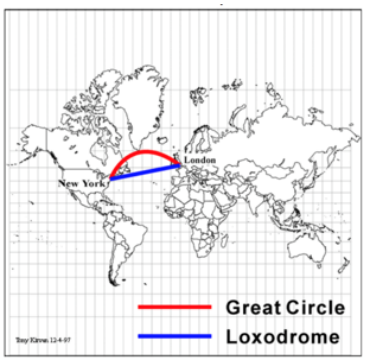

Figure 4.9 Distances over a 2D projection

Hence, a new proposal was designed with the goal of always working with Geodetic coordinates. This new technique was based on the Haversine formula to compute the distance between two points over the globe. Considering the coordinates of a point A as (λ_A, φ_A) and a point B as (λ_B, φ_B), the distance between them would be:

$$d_H(P_A, P_B) = R * c \tag{4.15}$$

Where R is the Earth's radius at the suitable latitude (λ) and c is a variable defined as:

$$R = \sqrt{\frac{(a^2 * cos\lambda)^2 + (b^2 * sin\lambda)^2}{(a * cos\lambda)^2 + (b * sin\lambda)^2}} \qquad \begin{matrix} a = \text{semi-major axis WGS-84} \\ b = \text{semi-menor axis WGS-84} \end{matrix} \tag{4.16}$$

$$c = 2 * atan2(\sqrt{k}, \sqrt{1-k}) \tag{4.17}$$

Where atan2 is the arctangent function with two arguments and k is:

$$k = sin^2\frac{\Delta\lambda}{2} + cos\lambda_A * cos\lambda_B * sin^2\frac{\Delta\varphi}{2} \qquad \begin{matrix} \Delta\lambda = \lambda_B - \lambda_A \\ \Delta\varphi = \varphi_B - \varphi_A \end{matrix} \tag{4.18}$$

4.6.2 Non-linear multilateration

The structure of the iterative algorithm was the following (Figure 4.10):

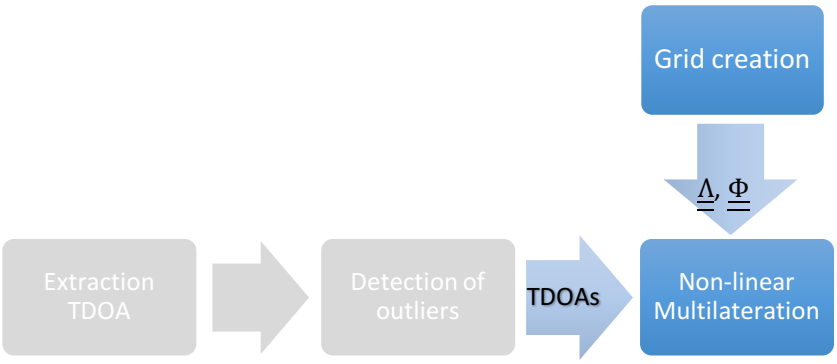

Figure 4.10 Iterative algorithm structure

The first step was to create a grid of possible latitude (Λ) and longitude (Φ) values of the end-node. As the region where the movement of the end-node was known, the values were restricted to this area. The scenario was the same as in the non-iterative algorithm (Figure 4.7), but the procedure to follow was different. Therefore, two matrices of *N* samples were created:

$$\underline{\underline{\Lambda}} = \begin{pmatrix} \lambda_1 & \cdots & \lambda_N \\ \vdots & \ddots & \vdots \\ \lambda_1 & \cdots & \lambda_N \end{pmatrix} \qquad \underline{\underline{\Phi}} = \begin{pmatrix} \varphi_1 & \cdots & \varphi_1 \\ \vdots & \ddots & \vdots \\ \varphi_N & \cdots & \varphi_N \end{pmatrix} \tag{4.19}$$

Then, for all possible latitude and longitude values, the distances from each of the four gateway to the end-node were computed:

$$\underline{\underline{d_i}} = \underline{\underline{d_{Hi}}}(gateway_i, node) \qquad \forall \, \mathrm{i} = 1{:}\,4 \tag{4.20}$$

The next step consisted in calculating the three values of the differential distances with the TDOAs:

$$D_{1i} = c_0 * t_{1i} \qquad \forall \, \mathrm{i} = 2{:}\,4 \tag{4.21}$$

It is important to notice that the previous values were scalars, since the TDOAs were scalars as well. The error ($\underline{\underline{\mathrm{E}_{1i}}}$) for each differential distance was computed and the final error was the sum of the previous three:

$$\underline{\underline{\mathrm{E}_{1i}}} = \left|\left(\underline{\underline{d_{H1}}} - \underline{\underline{d_{Hi}}}\right) - D_{1i}\right| \qquad \forall\, \mathrm{i} = 2{:}\,4 \tag{4.22}$$

$$\underline{\underline{\mathrm{E}_{F}}} = \sum_{i=2}^{4} \underline{\underline{\mathrm{E}_{1i}}} \tag{4.23}$$

The optimum values of latitude (λ) and longitude (φ) were selected by finding the minimum on the final error matrix. The error ε was also saved for the final step:

$$(\lambda, \varphi, \varepsilon) = \min\left(\underline{\underline{\mathrm{E}_{F}}}\right) \tag{4.24}$$

As in the non-iterative algorithm, this procedure was carried out three more times changing the reference gateway. Consequently, four possible values of latitude and longitude were candidates. Unlike in the previous method, the optimum pair of values $(\lambda_{opt}, \varphi_{opt})$ was selected comparing the error in each of the four cases. Again, the one with the minimum error was chosen:

$$(\lambda_{opt}, \varphi_{opt}) = \min\left(\underline{\varepsilon}\right) \tag{4.25}$$

5 TESTS AND RESULTS

5.1 Test spots

The gateways were distributed around the area taking into account some aspects. First of all, they had to form a four-sided polygon, each side with a length of around 2 or 3 km. The location was also selected according to the altitude, since the higher the antenna, the larger the coverage. This improves the sensitivity of the device, because if the height of the antenna is doubled, a gain of 6 dB is achieved considering the flat Earth model. Therefore, the positions of the gateways were the following (Figure 5.1):

- DTU (1): One gateway set permanently on the roof of the 344 building at DTU.
- Virum (2): One gateway located at the top of a high building in Virum.
- Bagsværd (3): Another one set on the roof of a house in Bagsværd, in front of the lake.
- Nybrovej (4): The last one located on the roof of a house near Nybrovej street.

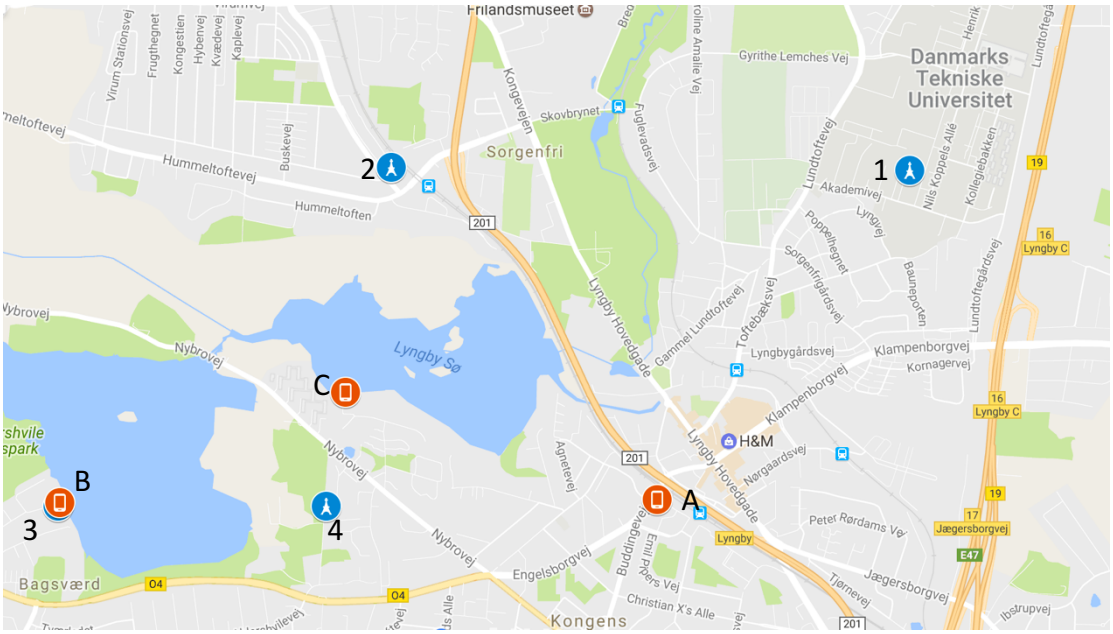

Figure 5.1 Map of the gateways and spots

Some places were tested in order to find the best spots to receive data from the four gateways. The end-node was switched on all the time and the java application was running as well. All the information about the coordinates from the GPS receiver and the received time by each gateway were inserted in the MySQL database. The number of gateways receiving packets from the end-node were checked until a suitable place was found. The previous figure (Figure 5.1) also shows the selected static locations, which were the following ones:

- Lyngby building (A): Located at the highest floor of a high building. 454 samples were recorded.
- Bagsværd (B): Location set in front of the lake, next to a gateway (3). 1728 samples were received.
- Lyngby lake (C): The most centered spot, in the center of the four gateways. 168 samples were inserted in the database.

The two different algorithms were applied in the three spots separately, as well as in all of them together with the other samples recorded in the "real-time" manner. As it is also explained in section 4.2, the detection of outliers was carried out only in the three static spots.

5.2 Extraction of TDOA

The following figures show the results of the TDOA calculation: the values of the acquired samples and their probability distribution.

Figure 5.2 displays the difference between two received times from two gateways in a temporal scale (identifiers of the packets). The actual value of the TDOA is also plotted in a red line which was computed before using the true position of the end-node in the spot. Therefore, all the obtained samples should be near this red line. However, one point is far from the real value, so it is a potential outlier.

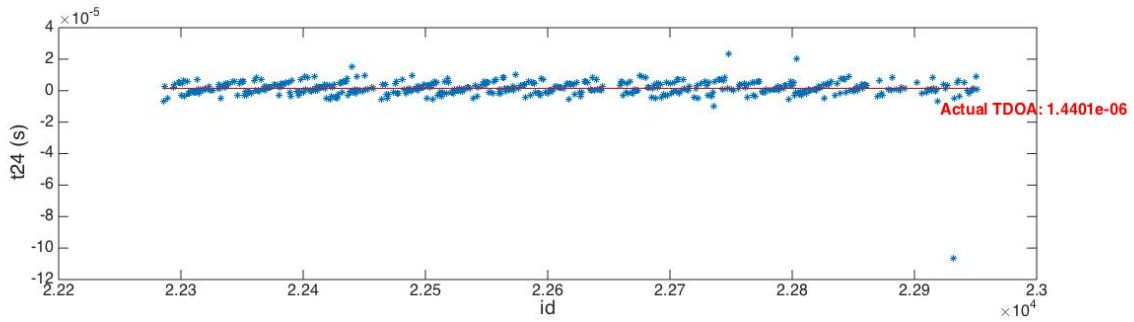

Figure 5.2 TDOA extraction

Figure 5.3 proves that the probability distribution of the TDOA can be considered as Gaussian. The samples are distributed around the actual TDOA, also plotted in the figure. The possible outlier is also present at approximately -10^{-4}.

This is just one of the examples of TDOA extraction. All the plots of the Lyngby spot can be found in Appendix F. The rest can be created by using the *mainScript.m* file in the Appendix E.

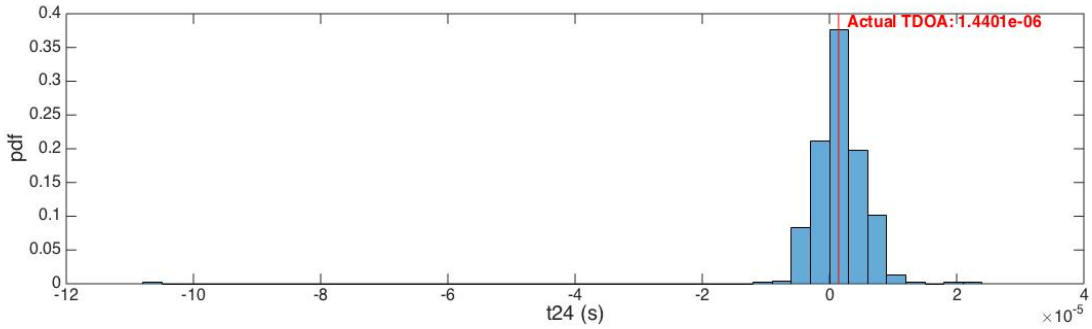

Figure 5.3 Probability distribution of TDOA

5.3 Detection of Outliers

The goal in this step was to remove the potential outliers in order to improve accuracy. The value that restricts how far the outliers are is the α parameter, called significance level, since it has influence in setting the upper bound. In other words, it is the probability of incorrectly reject outliers.

The optimum value of α was computed for the three different locations. The procedure to do so was the same for each spot. At first, for each alpha value, the mean of the TDOAs was plotted without the rejected outliers and the results were compared with the other five TDOAs. As it can be seen in Figure 5.4, the optimum value for t_{24} should be from 0.071 to 0.412, when the mean is closer to the actual TDOA. Therefore, if the alpha value is higher, which means rejecting more outliers, the mean can get worse again.

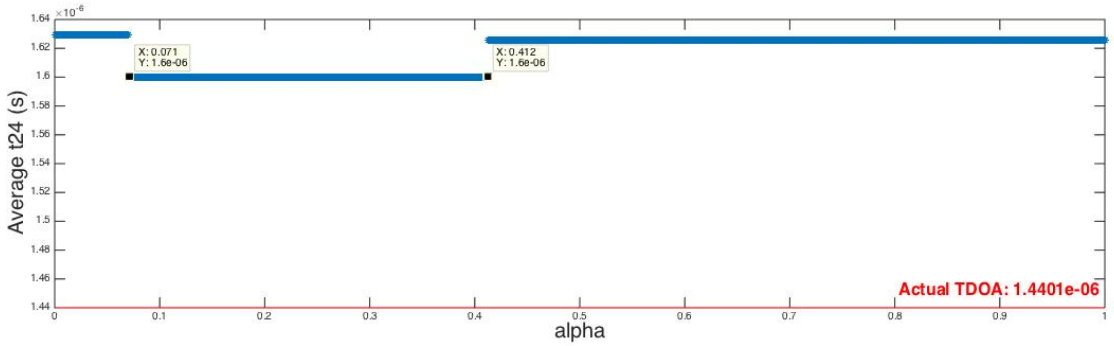

Figure 5.4 Average TDOA vs alpha

The results for each TDOA and each spot were obtained and the optimum was selected, as shown in Table 5.1:

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
t1-t2	0,387	0,126	[0 -> 0,38]
t1-t3	0,082	0,192	0,061 / 0,293
t1-t4	no changes	[0 -> 0,055]	0,039
t2-t3	0,435	no changes	[0 -> 0,522]
t2-t4	[0,071 -> 0,412]	[0 -> 0,099] or 0,136	[0,039 -> 0,082] / 0,269 / 0,468
t3-t4	[0 -> 0,098]	0,676	[0 -> 0,133]
Optimum	0,082	0,676	0,293

Table 5.1 Alpha values

The ideal case should be having approximately the same number in the first six rows for each location. However, this was not the case and the optimum one was discovered by using the trial and error method: selecting the one that provided a better accuracy.

The outliers were removed from the dataset. The following figures (Figure 5.5 and 5.6) show the improvement made with that procedure. The samples that were far away from the actual TDOA were disappeared.

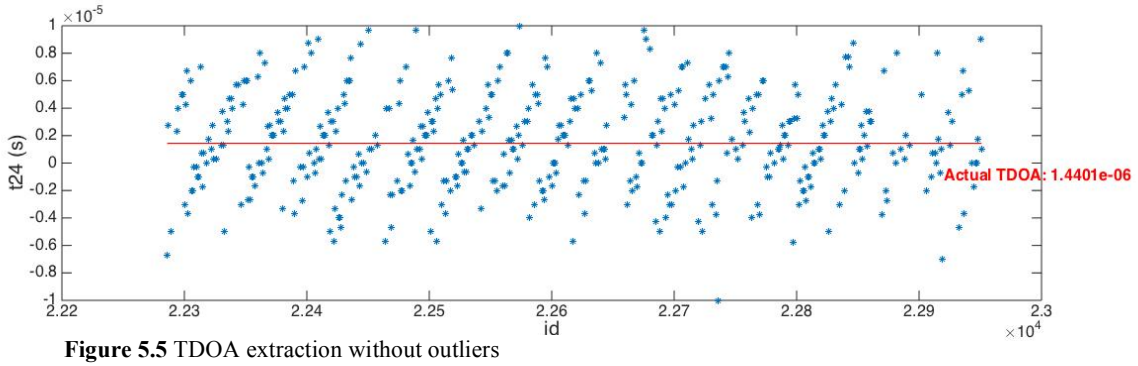

Figure 5.5 TDOA extraction without outliers

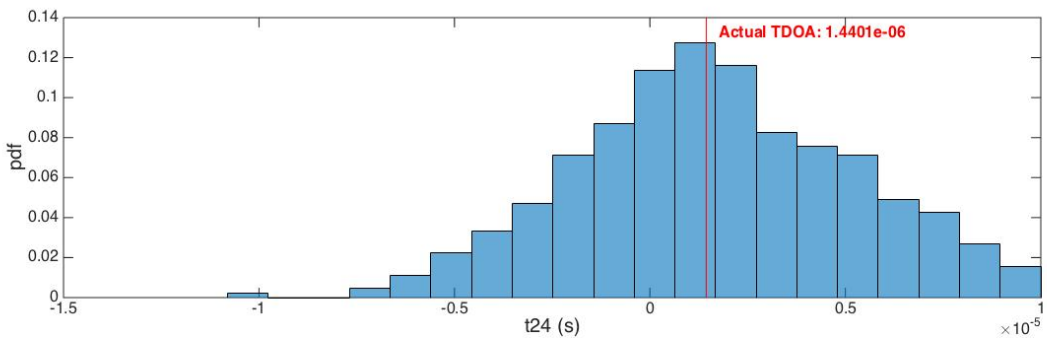

Figure 5.6 Probability distribution of TDOA without outliers

5.4 Non-iterative algorithm

In order to obtain some results from the estimated coordinates, the Mean Absolute Localization Error (MALE) was computed. The error consists in comparing the distance (error) between two coordinates using the Haversine formula, one versus one, and then doing the average of the whole distance error dataset.

For the case using all samples, the coordinates were compared with the ones acquired from the GPS receiver, since the position could change anytime. However, for the static spots, the comparison was done with the actual position of the device to avoid possible errors from the GPS.

Table 5.2 presents the results of the MALE in all the scenarios. In the first row, the overall dataset is used, while in the second row the outliers are removed. The error is fairly lower without outliers, although in the Bagsværd spot it is almost the same.

	All	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
MALE	9,58 km	5,54 km	10,72 km	9,76 km
MALE - No outliers TDOA	-	5,46 km	10,71 km	9,67 km

Table 5.2 Non-iterative algorithm MALE

The previous values were pretty poor, so the mean estimator -also called sample mean- was computed to improve the results:

$$\bar{t}_{ij} = \frac{1}{N}\sum_{k=1}^{N} t_{ijk} \tag{5.1}$$

Where t_{ijk} is the value of the TDOA t_{ij} for the sample k, and N is the size of the t_{ij} dataset. In this way, the algorithm only used the mean estimator of each TDOA. The results were much better (Table 5.3).

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
ALE - Sample mean TDOA	167 m	89 m	427 m
ALE - No outliers, Sample mean TDOA	116 m	65 m	206 m

Table 5.3 Non-iterative algorithm ALE. Sample mean.

5.5 Iterative algorithm

The grid of latitude and longitude coordinates was created near the area delimited by the gateways, as can be seen in Figure 5.7. The iterative algorithm tested all the possible values inside this region. The step between latitude and longitude values was configured to 0.0001 degrees, which means around 11 meters of precision at the equator, increasing towards the poles [26].

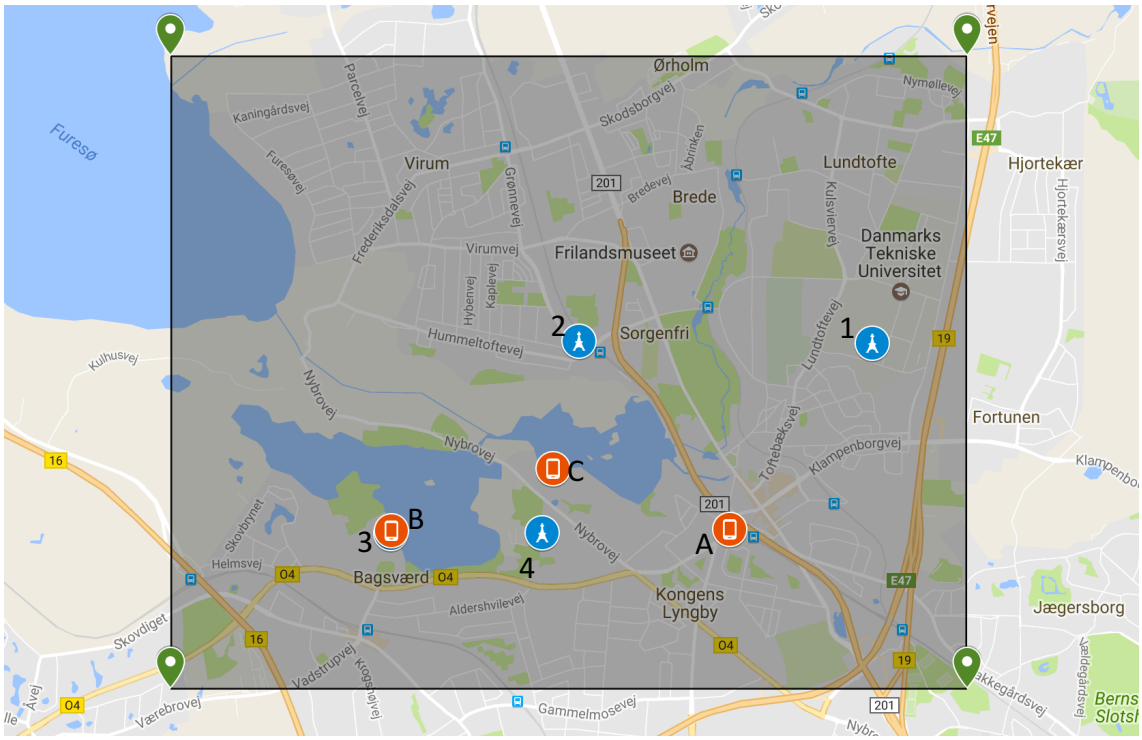

Figure 5.7 Grid area iterative algorithm

The MALE (Table 5.4) was computed in the same manner as in the previous algorithm. The results were better because the possible values of latitude and longitude were always inside the grid.

	All	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
MALE	1,16 km	1,39 km	1,09 km	1,15 km
MALE - No outliers TDOA	-	1,38 km	1,08 km	1,12 km

Table 5.4 Iterative algorithm MALE

The sample mean improved the accuracy of the results, as it is shown in Table 5.5. Removing outliers had a big impact when averaging, since the difference was larger.

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
ALE - Sample mean TDOA	167 m	175 m	136 m
ALE - No outliers, Sample mean TDOA	119 m	127 m	114 m

Table 5.5 Iterative algorithm ALE. Sample mean.

5.6 Required samples

The number of required samples to make the device real-time was estimated. Four thresholds were determined as the maximum allowed error in distance: 50 m, 100 m, 500 m and 1000 m.

The thresholds were transformed to the time domain because the data to be processed was the six different TDOAs, and not the estimated position. The approach was to define these thresholds using the typical formula to convert distance to time:

$$t_{ijTH} = \frac{d_{TH}}{c_0} \qquad c_0 \equiv speed\ of\ the\ light \tag{5.2}$$

The probability density function of the sample mean estimators was Gaussian, so each $\bar{t}_{ij}$ was assumed to be a Normal random variable $N(\mu_{ij}, \sigma_{ij}^2)$. The mean ($\mu_{ij}$) was the actual value of t_{ij} and the variance (σ_{ij}^2) was considered as the variance of the sample set. Therefore, both values were known. From now on, the random variable $\bar{t}_{ij}$ is defined as $\overline{T}$ with mean μ and variance σ^2. $\overline{T}$ was normalized, obtaining a random variable Z whose probability distribution is clearly a normal standard one:

$$Z = \frac{\overline{T} - \mu}{\sigma/\sqrt{n}} \equiv N(0,1) \tag{5.3}$$

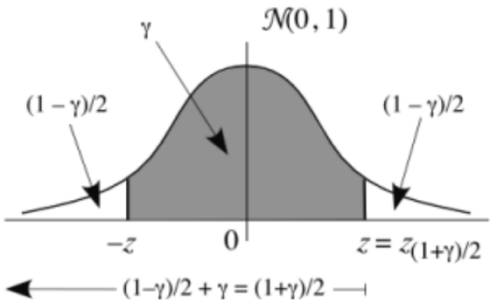

Figure 5.8 Standard normal distribution

A 95% confidence interval was defined, so $\gamma = 0.95$. The upper bound and lower bound (z) were the quantiles $z_{(1+\gamma)/2}$ of the standard normal distribution (Figure 5.8). Then, the next equation was solved in order to find the number of required samples n:

$$P\left(|\bar{T} - \mu| < t_{ijTH}\right) \geq 0.95 \tag{5.4}$$

$$\begin{aligned} &P\left(|\bar{T} - \mu| < t_{ijTH}\right) = P\left(-t_{ijTH} < \bar{T} - \mu < t_{ijTH}\right) = \\ &= P\left(\frac{-t_{ijTH}}{\sigma/\sqrt{n}} < \frac{\bar{T} - \mu}{\sigma/\sqrt{n}} < \frac{t_{ijTH}}{\sigma/\sqrt{n}}\right) = P(-z < Z < z) \end{aligned} \tag{5.5}$$

$$\begin{aligned} z &= \frac{t_{ijTH}}{\sigma/\sqrt{n}} \\ n &= \left(\frac{z * \sigma}{t_{ijTH}}\right)^2 \end{aligned} \tag{5.6}$$

The expression above (5.6) was the final formula to find the number of samples, where z was 1.96, since γ was 0.95. Different values of n were obtained from the six TDOAs. The worst case (the largest value) was the final solution. Table 5.6 shows the results for each spot considering all the dataset.

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
50 m	5637	2003	41457
100 m	1410	501	10366
500 m	57	21	415
1000 m	15	6	104

Table 5.6 Number of required samples

The interpretation of the results is the following: the estimated mean with n samples is inside the defined interval with a probability of 95%. For instance, in the Lyngby building spot, the error obtained from the mean estimator computed with 57 samples will be below 500 meters in 95% of the cases.

The number of required samples calculated using the dataset without outliers is shown in Table 5.7:

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
50 m	1838	1929	2297
100 m	460	483	575
500 m	19	20	23
1000 m	5	5	6

Table 5.7 Number of required sample without outliers

Less samples were needed because the standard deviation was lower due to the removed outliers. Taking the same example as before, 19 instead of 57 samples were needed to obtain an error of 500 meters.

5.7 Required time

Another important matter was to know the required time to transmit the n samples previously computed. This time was mainly limited by the duty cycle and the low data rate of LoRa: the lower the data rate, the larger the required time.

The first step was to compute the time needed to transmit a single packet [27]:

$$T_{symb} = \frac{2^{SF}}{BW} = \frac{2^7}{125\ kHz} = 1{,}024\ ms \tag{5.7}$$

$$T_{preamble} = \left(n_{preamble} + 4{,}25\right) * T_{symb} = (8 + 4{,}25) * 1{,}024 = 12{,}544\ ms \tag{5.8}$$

$$T_{payload} = T_{symb} * P = T_{symb} * \left(8 + max\left(ceil\left(\frac{8PL - 4SF + 28 + 16}{4(SF - 2DE)}\right)(CR + 4), 0\right)\right) = 1{,}024 * (8 + 35) = 44{,}032\ ms \tag{5.9}$$

$$T_{packet} = T_{preamble} + T_{payload} = 12{,}544 + 44{,}032 = 56{,}576\ ms \tag{5.10}$$

The following values were taken:

	Description	Value
SF	Spreading Factor	7
BW	Bandwidth of the signal	125 kHz
$\boldsymbol{n_{preamble}}$	Number of programmed preamble symbols	8
PL	Number of payload bytes	13 + 1 bytes
DE	Low data rate optimization enabled (1), otherwise 0	1
CR	Coding Rate:	4/5 -> 1

Table 5.8 Values transmission

Kerlink gateways use eight different subchannels: three with a duty cycle of 1% and five with a duty cycle of 0.1%. Applying the formula of the duty cycle, the unavailable time for each subchannel is (T_{off}):

$$T_{off} = T_{air} * \left(\frac{1}{DutyCycle} - 1\right) \tag{5.11}$$

Where T_{air} is the time required to transmit a single packet T_{packet}. The following results were obtained: $T_{off1\%} = 5{,}601\ s$ and $T_{off0{,}1\%} = 56{,}519\ s$.

A program was designed in order to estimate the time taking into account the previous values of the duty cycles and considering a processed time between transmissions null. The following results were obtained:

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
50 m	2,7 h	1,1 h	19 h
100 m	50 min	17 min	5 h
500 m	85 s	28 s	13 min
1000 m	17 s	0.4 s	164 s

Table 5.9 Required time

	Lyngby building (A)	Bagsværd (B)	Lyngby lake (C)
50 m	1 h	1 h	1,25 h
100 m	15 min	16 min	20 min
500 m	23 s	22,8 s	28,5 s
1000 m	0,28 s	0,28 s	0,4 s

Table 5.10 Required time without outliers

Considering the same example as before, 85 seconds should be needed to obtain an error of 500 meters and 23 seconds without outliers. However, a real-time device with an accuracy of 50 meters is not feasible, since in the best case would require 1 hour.

6 CONCLUSIONS

New applications for Internet of Things in LPWANs are arising in the recent years due to the attractive features that it provides, such as low cost, low power consumption, low data rate and long range. For instance, LoRa geolocation, which aims to face the drawbacks of the GPS using IoT.

In this thesis, a whole IoT tracking system was designed and implemented in order to present accuracy results using the LoRa technology. The two designed algorithms demonstrated that it can be feasible to locate a device in a static spot with an accuracy of around 100 meters. However, for a real-time tracking application it can only be seen as a first approach, and not as a usable one.

In general, the results of the iterative algorithm were better because the potential solutions of geodetic coordinates were restricted to a specific area. Nevertheless, the computational resources needed in such algorithm are much higher since all the coordinates are tested. But, as the algorithm is in the server site, this should not suppose any problem because the available resources there are unlimited.

The Generalized ESD test allowed to detect the main outliers to improve the accuracy. However, this was a difficult task because the alpha parameters were not the same for all three different scenarios and it would be difficult to implement in a usable device.

The mean estimator clearly enhanced the results and both algorithms presented approximately the same values. 460 samples would be necessary to estimate a position with an accuracy of 100 meters in order to make it a real-time device. This is quite difficult to achieve due to the limitations in the duty cycle. The device would require 15 minutes to transmit those 460 samples. Therefore, some improvements must be implemented in order to make this device real-time.

The enhancements should not only be done in the algorithm, but also in the gateways, since they are responsible for recording the time when the packet is received. The clock of the Kerlink gateways was not intended for using in geolocation techniques, so the company will launch a new version with a faster clock in a few months. This will probably allow to obtain an accurate time. Another solution could be to increase the number of gateways in order to increase the number of TDOA and consequently improve the accuracy. However, this would require to deploy more antennas, which would increase the cost of the system.

Another problem is the multipath of the signal, which causes wrong measurements of the TDOAs. Gateways do not always receive the direct path of the signal due to the reflections with terrestrial objects like buildings, forests or mountains. The ability to resolve this phenomenon depends on the bandwidth of the signal. If the bandwidth is large, the resolution is better and vice versa. The bandwidth employed in LoRa is small (125 kHz), so the recorded times in the gateways can be the time of a multipath signal instead of the direct one.

6.1 Future work

As a future development of the system, it would be interesting to work on different parts: the hardware, the algorithm and the third-party application.

The hardware could be improved by designing an embedded board with the LoRaWAN module and also by adding an accelerometer to detect movement. For instance, the frequency of sending packets to the server could be modified according to the movement of the device. If the device is still, there is no need of transmitting a lot of information. As a low power consumption device, a power supply could be designed using solar cells or the movement of the wheels in bicycles. In this way, the user would not need to recharge the battery and it would become an autonomous system.

Regarding the algorithm, other lines could be followed to improve the results. First of all, Machine Learning techniques like Decision Tree, Naive Bayes or Support Vector Machine could be applied to combine TDOAs with RSSI measurements. Low values of RSSI might mean that the received signal is not the direct path and can be discarded. In this case, a large dataset would be needed in order to divide it in three different groups: one for training, one for validating and one for assessing the suggested model. In order to improve the accuracy in “real-time”, the k-Nearest Neighbors technique could be used, since it processes closer samples. The Haversine formula used in the non-iterative algorithm can be substituted by the Vicenty formula which is more accurate.

The first step as a future work for the third-party application is to rewrite the MATLAB code into Java code in order to run the software in real-time. Then, it could incorporate a user interface to display the device and allow the user to track it. Using the Google Maps API and a simple JavaScript running in a local server as a first approach is quite straight forward. Finally, a smartphone application to communicate with the device could be created as well with some extra features.

7 REFERENCES

[1] LinkLabs, “Low Power, Wide Area Networks,” p. 16, 2016.

[2] Sigfox, “M2M and IoT redefined through cost effective and energy optimized connectivity,” *White Pap.*, 2014.

[3] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, “LoRaWAN Specification,” pp. 1–82, 2015.

[4] P. Sector, “The Internet of Everything,” pp. 1–13, 2013.

[5] I. Update, O. N. The, P. Of, and T. H. E. Networked, “Mobility Report,” no. February, pp. 4–7, 2013.

[6] © Libelium Comunicaciones Distribuidas S.L., “Waspmote LoRaWAN Networking Guide,” p. 56, 2016.

[7] A. Llaria, G. Terrasson, H. Arregui, and A. Hacala, “Geolocation and monitoring platform for extensive farming in mountain pastures,” *2015 IEEE Int. Conf. Ind. Technol.*, pp. 2420–2425, Mar. 2015.

[8] R. N. Handcock, D. L. Swain, G. J. Bishop-Hurley, K. P. Patison, T. Wark, P. Valencia, P. Corke, and C. J. O’Neill, “Monitoring Animal Behaviour and Environmental Interactions Using Wireless Sensor Networks, GPS Collars and Satellite Remote Sensing,” *Sensors*, vol. 9, no. 5, pp. 3586–3603, May 2009.

[9] V. R. Jain, R. Bagree, A. Kumar, and P. Ranjan, “wildCENSE: GPS based animal tracking system,” *ISSNIP 2008 - Proc. 2008 Int. Conf. Intell. Sensors, Sens. Networks Inf. Process.*, pp. 617–622, 2008.

[10] “Sherlock.” [Online]. Available: https://www.sherlock.bike/.

[11] “Spybike.” [Online]. Available: http://www.spybike.com/.

[12] “Connected Cycle.” [Online]. Available: http://connectedcycle.com/.

[13] “Helios.” [Online]. Available: http://www.ridehelios.com/.

[14] “Lattis.” [Online]. Available: https://www.lattis.io/.

[15] S. Sas, “Location-Enabled LoRa ™ IoT Network : ‘ Geo -LoRa- ting ’ your assets.”

[16] “LoRa geolocation - Multipath.” [Online]. Available: https://www.link-labs.com/lora-localization/.

[17] A. Village, H. I. Park, B. District, L. Town, and S. City, “VK2828U7G5LF.”

[18] A. Note, “Gateway to Server Interface LoRaWAN Network Server Demonstration : Gateway to Server Interface Definition Gateway to Server Interface,” no. March, pp. 1–17, 2015.

[19] “Packet forwarder.” [Online]. Available: https://github.com/Lora-net/packet_forwarder.

[20] “The Things Network backend.” [Online]. Available: https://www.thethingsnetwork.org/wiki/Backend/Home.

[21] “LoRa server.” [Online]. Available: https://github.com/gotthardp/lorawan-server.

[22] “MQTT Client.” [Online]. Available: https://eclipse.org/paho/.

[23] V. Barnett; T. Lewis, *Outliers in Statistical Data*, 3rd ed. Wiley Series in Probability and Mathematical Statistics, 1994.

[24] “Generalized ESD MATLAB library.” [Online]. Available: https://se.mathworks.com/matlabcentral/fileexchange/28501-tests-to-identify-outliers-in-data-series/content/gesd.m.

[25] O. Survey, “The ellipsoid and the Transverse Mercator projection,” *Geod. Inf.*, vol. version 2., no. 1, p. ?, 1998.

[26] “Decimal degrees.” [Online]. Available: https://en.wikipedia.org/wiki/Decimal_degrees.

[27] Semtech, “LoRa Modem Design Guide,” no. July, pp. 1–9, 2013.

8 APPENDIX A. LINEAR MULTILATERATION ALGORITHM

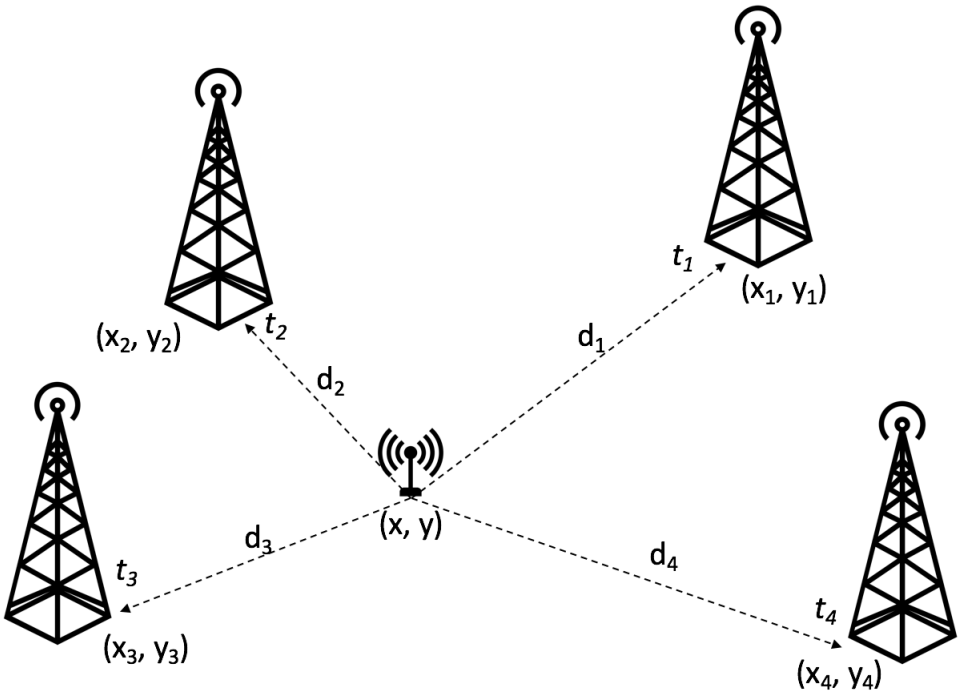

Figure 8.1 Scenario of the algorithm

The distances from each gateway to the end-node were defined as:

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \qquad \forall \, \mathrm{i} = 1{:}\,4$$

Considering gateway 1 as a reference, the differential distances between the reference gateway and the others were:

$$D_{i1} = d_i - d_1 = c_0 t_{i1} \qquad \forall \, \mathrm{i} = 2{:}\,4$$

Where c_0 is the speed of the light: 299792458 m/s

The system of linear equations was formulated as follows:

$$\begin{cases} d_2 - d_1 = c(t_2 - t_1) = D_{21} \\ d_3 - d_1 = c(t_3 - t_1) = D_{31} \\ d_4 - d_1 = c(t_4 - t_1) = D_{41} \end{cases} \rightarrow \begin{cases} d_2 = D_{21} + d_1 \\ d_3 = D_{31} + d_1 \\ d_4 = D_{41} + d_1 \end{cases}$$

$$\begin{cases} {d_2}^2 = (D_{21} + d_1)^2 = {D_{21}}^2 + {d_1}^2 + 2D_{21}d_1 \\ {d_3}^2 = (D_{31} + d_1)^2 = {D_{31}}^2 + {d_1}^2 + 2D_{31}d_1 \\ {d_4}^2 = (D_{41} + d_1)^2 = {D_{41}}^2 + {d_1}^2 + 2D_{41}d_1 \end{cases}$$

$$\begin{cases} (x - x_2)^2 + (y - y_2)^2 = {D_{21}}^2 + (x - x_1)^2 + (y - y_1)^2 + 2D_{21}d_1 \\ (x - x_3)^2 + (y - y_3)^2 = {D_{31}}^2 + (x - x_1)^2 + (y - y_1)^2 + 2D_{31}d_1 \\ (x - x_4)^2 + (y - y_4)^2 = {D_{41}}^2 + (x - x_1)^2 + (y - y_1)^2 + 2D_{41}d_1 \end{cases}$$

$$\begin{cases} x^2 + {x_2}^2 - 2xx_2 + y^2 + {y_2}^2 - 2yy_2 = {D_{21}}^2 + x^2 + {x_1}^2 - 2xx_1 + y^2 + {y_1}^2 - 2yy_1 + 2D_{21}d_1 \\ x^2 + {x_3}^2 - 2xx_3 + y^2 + {y_3}^2 - 2yy_3 = {D_{31}}^2 + x^2 + {x_1}^2 - 2xx_1 + y^2 + {y_1}^2 - 2yy_1 + 2D_{31}d_1 \\ x^2 + {x_4}^2 - 2xx_4 + y^2 + {y_4}^2 - 2yy_4 = {D_{41}}^2 + x^2 + {x_1}^2 - 2xx_1 + y^2 + {y_1}^2 - 2yy_1 + 2D_{41}d_1 \end{cases}$$

$$\begin{cases} x(-2x_2 + 2x_1) + y(-2y_2 + 2y_1) + d_1(-2D_{21}) = -{x_2}^2 - {y_2}^2 + {D_{21}}^2 + {x_1}^2 + {y_1}^2 \\ x(-2x_3 + 2x_1) + y(-2y_3 + 2y_1) + d_1(-2D_{31}) = -{x_3}^2 - {y_3}^2 + {D_{31}}^2 + {x_1}^2 + {y_1}^2 \\ x(-2x_4 + 2x_1) + y(-2y_4 + 2y_1) + d_1(-2D_{41}) = -{x_4}^2 - {y_4}^2 + {D_{41}}^2 + {x_1}^2 + {y_1}^2 \end{cases}$$

$$\begin{pmatrix} -2(x_2 - x_1) & -2(y_2 - y_1) & -2D_{21} \\ -2(x_3 - x_1) & -2(y_3 - y_1) & -2D_{31} \\ -2(x_4 - x_1) & -2(y_4 - y_1) & -2D_{41} \end{pmatrix} * \begin{pmatrix} x \\ y \\ d_1 \end{pmatrix} = \begin{pmatrix} -{x_2}^2 - {y_2}^2 + {D_{21}}^2 + {x_1}^2 + {y_1}^2 \\ -{x_3}^2 - {y_3}^2 + {D_{31}}^2 + {x_1}^2 + {y_1}^2 \\ -{x_4}^2 - {y_4}^2 + {D_{41}}^2 + {x_1}^2 + {y_1}^2 \end{pmatrix}$$

9 APPENDIX B, C, D, E, F

The rest of the appendices can be found in the attached file *Appendices.zip*. Each appendix contains the following:

- **Appendix B. End-node:** Waspmote code, and LoRaWAN and GPS C++ custom libraries.
- **Appendix C. Gateway:** Binary file (*gps_auto*), raw C code, two configuration JSONs and cross compilation steps.
- **Appendix D. Application:** Java code and also the structure of the database (*localizationDataBase.sql*) and the recorded data.
- **Appendix E. Matlab:** Matlab code to obtain the results. The main file is *mainScript.m*.
- **Appendix F. Lyngby Spot results:** Plots of the Lyngby spot.