

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Time management through indoor location

André Costa Teixeira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Rui Esteves Araújo

External Supervisor: Hélder Tiago Silva

July 27, 2021

Resumo

Os sistemas de localização interior evoluíram muito nas últimas décadas e a sua importância tem crescido progressivamente em vários aspectos da vida quotidiana, especialmente em ambientes empresariais.

Esta dissertação descreve a conceção e avaliação de um possível sistema de posicionamento "indoor" para recolha de dados sobre os colaboradores e os ativos da empresa *RACELAND S.A.*. Este sistema recorre a tecnologia BLE (Bluetooth Low Energy) iBeacon que possibilita a identificação e recolha da força do sinal para determinar as distâncias aos pontos de referência.

O principal objetivo é conceber um sistema com suficiente exatidão que pudesse recolher dados sobre as tarefas que estão a ser executadas para depois avaliar com base em indicadores de desempenho.

O trabalho aqui apresentado versa sobre uma abordagem diferente aos aparelhos BLE Beacon, os quais são usados como alvos móveis. O sistema implementado utiliza pontos de acesso, os chamados "gateways", para recolher e enviar os dados enviados pelos cartões beacon. Cada cartão beacon será atribuído a um empregado ou ativo da empresa. Através da recolha dos sinais recebidos de diferentes pontos de acesso é possível estimar a posição do beacon.

Uma pesquisa do estado da arte em tecnologias de localização interior é disponibilizada e uma comparação entre algumas das possibilidades ou tecnologias já investigadas ou presentes no mercado é apresentada.

O tema de localização nas suas várias vertentes diferentes é explicado e são discutidas as questões relativas à privacidade e segurança dos utilizadores. A estrutura base relativa ao tema da localização é definida e os conceitos fundamentais sobre a localização e o projeto são explicados. Também é feita uma referência às questões relativas à privacidade e segurança dos utilizadores.

Em seguida, é feita uma apresentação do projeto a realizar, onde são definidos os requisitos e características que a empresa pretende conseguir. O sistema é modelado e é descrita a arquitetura do sistema. A metodologia de desenvolvimento é descrita, decompondo os passos e testes efetuados até se atingir o sistema completo. Depois do sistema ter sido concluído, é posto em prova e avaliado com algumas métricas e indicadores.

Por fim, termina-se expondo o que foi conseguido, as limitações e o que pode ser feito para melhorar o sistema.

Abstract

Indoor location systems have evolved greatly in recent decades and their importance has progressively grown in various aspects of everyday life, especially in business environments.

This dissertation describes the design and evaluation of a possible indoor positioning system to collect data about the assets and personnel of the company *RACELAND S.A.*. This system uses the iBeacon BLE (Bluetooth Low Energy) technology that enables the identification and collection of the signal strength to determine the distances to the reference points.

The main objective is to design a system with sufficient accuracy that it could collect data on the tasks being performed and then evaluate it based on key performance indicators.

The work presented here is about a different approach to BLE Beacon devices, which are used as moving targets. The implemented system uses access points, called "gateways", to collect and send the data sent by the beacon cards. Each beacon card will be assigned to an employee or company asset. By collecting the signals received from different access points, it is possible to estimate the position of the beacon.

A research of the state of the art in indoor location technologies is made available and a comparison between some of the possibilities or technologies already investigated or present in the market is presented.

The theme of location in its various different aspects is explained and issues relating to user privacy and security are discussed. The basic framework for the subject of localization is defined and the fundamental concepts of localization and design are explained. A discussion is also made to address the issues relating to user privacy and security.

Then, a presentation of the project to be developed is made, where the requirements and features that the company intends to achieve are defined. The system is modeled and the architecture of the system is described. The development methodology is described, decomposing the steps and tests carried out until reaching the complete system. Once the system is completed, it is evaluated with some metrics and benchmarks.

Finally, it ends by exposing what has been achieved, the limitations and what can be done to improve the system.

Acknowledgments

First and foremost, I would like to thank the professor and my supervisor, Rui Manuel Esteves Araújo, for the opportunity to work on this project, as well as for his constant availability and advice on the best paths to take.

To my external supervisor Hélder Tiago Silva from *Net Worker - Soluções Informáticas, Lda.* who accompanied me daily in the company *RACELAND S.A.*. Thank you for your tremendous encouragement and motivation, as well as your assistance in comprehending and solving issues that have arisen throughout this project.

There should be no lack of acknowledgement to the company *RACELAND S.A.* itself, which welcomed me and provided all the means, personnel and materials necessary to carry out the project. Thanks for the pleasant working environment and all the investments made. I hope I didn't let you down.

Among the elements of *RACELAND*, I would also like to highlight the technician Xavier Ribeiro for his help in setting up the access points and for his patience in collecting the data in the experiments. To the remaining elements of *RACELAND S.A.* I also leave my acknowledgments.

I would like to express my gratitude to my family. To my parents, Carlos Teixeira and Anabela Teixeira for their constant support and motivation throughout my academic career, as well as for their efforts, which made my academic path possible. To my brother Miguel Teixeira, who, for his image editing skills, helped me with some of the figures in the thesis.

To my colleagues in the course, without you I wouldn't get this far.

André Costa Teixeira

*“The difficulty lies not so much in developing new ideas
as in escaping from old ones.”*

John Maynard Keynes

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Opportunities for indoor positioning	2
1.3	Context and Objectives	3
1.4	Document's structure	4
2	Literature review and fundamentals	5
2.1	State of Art	5
2.1.1	Technology comparison and discussion	11
2.2	Systems engineering framework	12
2.3	Location Information Quality	14
2.4	Localization-enabling information	16
2.5	Location measurement techniques	18
2.6	iBeacon protocol fundamentals	22
2.7	Message Queuing Telemetry Transport (MQTT)	23
2.8	Privacy and security issues	23
3	System Computational Architecture	27
3.1	Overview	27
3.2	Features required	28
3.2.1	Requirements	29
3.3	System's Computational Architecture	32
3.3.1	Database Structure	33
4	System Development	35
4.1	Sensors	35
4.1.1	Parsing Results and Conclusions	37
4.2	Filtering	40
4.2.1	Filter Techniques	40
4.2.2	Test Results and Conclusions	43
4.3	Localization Estimator	44
4.3.1	Experiments and Results	49
4.4	System Performance Evaluation	53
5	Conclusions and Future Work	59
5.1	Conclusions	59
5.2	Achievements and Future Work	59
5.2.1	Achievements	60

5.3 Final remarks	62
References	63
A Hardware	67
A.1 G1 Minew Gateways	67
A.2 C7 iBeacon Tags	68
B User Interface Prototype	71
C Developed Server Programs	75
D Cramer-rao Bound Matlab Code	113

List of Figures

1.1	Global Indoor Positioning and Indoor Navigation Market by Application.	2
1.2	Global Indoor Positioning and Indoor Navigation Market by Technology.	3
2.1	Frequency range of sound with distinction of the three main sections: infrasound, audio and ultrasound.	9
2.2	Comparison of indoor positioning technologies.	11
2.3	7 layer model of the Location Stack.	13
2.4	Aspects representing quality of location and orientation information along the spatial and temporal dimensions.	14
2.5	Trilateration method overview in a 2 dimensional plane.	19
2.6	Triangulation method overview.	21
2.7	MQTT protocol illustration.	23
2.8	Three pillars of information security.	25
3.1	Simple warehouse floor plan and basic indoor localization concept.	28
3.2	System's subsystems and information flow.	32
3.3	Class diagram for parsing software's database.	33
4.1	Parsing Sequence Diagram.	36
4.2	Proximity experiment illustration.	37
4.3	Raw RSSI Value Results, meter by meter (Left) and 2 by 2 meters (Right).	38
4.4	Ideal value and estimation average fitting curve for $n = 4$ and $A_0 = -27$ dBm.	39
4.5	Estimated Distance Value Results, meter by meter (Left) and 2 by 2 meters (Right).	40
4.6	Example of one iteration of the Particle Filter algorithm.	42
4.7	Actual and filtered RSSI (dBm) over time (seconds).	43
4.8	Filtered RSSI Value Results, meter by meter (Left) and 2 by 2 meters (Right).	44
4.9	Estimated Distance Value Results, meter by meter (Left) and 2 by 2 meters (Right).	44
4.10	Positioning Sequence Diagram.	45
4.11	Clustering technique demonstration, Blue dots are the set of estimated distances, Red dot is the output result.	48
4.12	Overview Sketch of the testing zone as implemented.	50
4.13	Gateways' Placement.	51
4.14	Demonstration of how the tests were conducted.	51
4.15	Position Estimate Results and their real positions for comparison.	52
4.16	Graphical plot of percentage of realizations while considering tolerance (defined as the maximum Euclidean error in which a realization is considered).	54
4.17	Positioning Engine's Confusion Matrix.	55

4.18	Average localization error (calculated as the square root of the MSE) as a function of the standard deviation of the RSS measurements. The Cramer–Rao bound is shown as comparison.	57
4.19	MSE Heatmap based on the location.	58
A.1	Gateway G1.	67
A.2	G1 Web configuration interface	68
A.3	C7 Tag iBeacon	69
A.4	Configuration menu of "BeaconSet+" mobile application	69
B.1	Innitial Menu	71
B.2	Dashboard's Positioning Map	72
B.3	Dashboard's Configuration Menu	73

List of Tables

2.1	Structure of a iBeacon payload.	22
3.1	RACELAND S.A. requested features.	29
3.2	Functional requirements.	30
3.3	Performance requirements.	31
4.1	Average RSSI values for each distance measured.	38
4.2	Average Estimated distance values for each distance measured.	39
4.3	Beacon's Hash Table Structure.	46
4.4	Anchor Gateway's Hash Table Structure.	46
4.5	Gateways' Center Position Coordinatates.	50
4.6	Mean Squared Error (MSE), Standard Deviation, Maximum and Minimum error values using Euclidean distances for each situation presented.	53
5.1	Functional requirements.	60
5.2	Performance requirements.	61

Abreviaturas e Símbolos

U.S.	United States
IoT	Internet of Things
EPA	Environmental protection Agency
GPS	Ground Positioning System
GNSS	Global Navigation Satellite System
IEEE	Institute of Electrical and Electronics Engineers
NIC	Network Interface Card
WLAN	Wireless Local Area Network
WAN	Wide Area Network
LP-WAN	Low Power - Wide Area Network
WSN	Wireless Sensor Network
RSSI	Relative Signal Strength Indicator
BLE	Bluetooth Low Energy
WPAN	Wireless Personal Area Network
IPIN	Indoor Positioning Indoor Navigation
LoRa	LoRa Wide Area Network
CSMA/CA	Carrier sense multiple access with collision avoidance
EMR	Eletromagnetic Radiation
AIDC	Automatic Identification and Data Capture
RFID	Radio Frequency Identification
UWB	Ultra Wide Band
IR	Infra Red
RF	Radio Frequency
FCC	Federal Communications Commission
OSI	Open System Interconnection
TOF	Time of Flight
TOA	Time of Arrival
TDOA	Time Difference of Arrival
AOA	Angle of Arrival
GUI	Graphical User Interface
MS	Microsoft Corporation
MQTT	Message Queuing Telemetry Transport
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
OEM	Original Equipment Manufacturer
ERP	Enterprise Resource Planning
PK	Primary Key
FK	Foreign Key
SSMS	Server Management Studio
OOP	Object Oriented Programming
UID	Unique Identifier
LSE	Least Squared Error
MSE	Mean Squared Error

Chapter 1

Introduction

1.1 Motivation

The localization and tracking of users and objects in specific spaces has been a popular theme among researchers and engineers. With the advent of smart environments and IoT (Internet of Things) devices, precise user and object localization has become even more important than before the rise of these paradigms. If a system or environment could precisely follow the movement of the user, it could customize its interface and behaviour to match the references, history, and context of that particular ambient to intervene in the case of necessity or danger.

It's often said that outdoor environments, GPS, plays a dominant role in localization. On the other hand, GPS technologies are inefficient for indoor scenarios due to innacuracy for small spaces, weakness of signals emitted by this system and their disability to penetrate solid walls or structures.

According to the U.S. Environmental Protection Agency (EPA):

"People spend approximately 93% of their time indoors, 2% outdoors, and 5% in transit (e.g car, train, bus). Thus, from a time budget standpoint, indoor environments dominate the total exposure spectrum [1]."

Considering that people spend most of their time in indoor environments, other effective technologies are demanded for indoor human/object location and some techniques have been developed these last decades. However, due to restrictions of position timing, position accuracy and complex indoor environments, a flawless positioning technique for each situation is yet to be achieved [2].

This thesis will incorporate the concepts of localization and IoT protocols to collect data and manage a company's manufacturing tasks, which is an uncommon use and requires an uncommon approach.

1.2 Opportunities for indoor positioning

The location-based analytics segment dominated the overall indoor positioning and indoor navigation market share in 2017 and is expected to continue this trend during the forecast period due to increased demand from the retail industry, as it allows end users to understand customers in real-time and send relevant messages related to their identity and needs. Furthermore, the asset and personnel tracking segment is expected to grow rapidly during the forecasted period [3].

In terms of application, the market can be segmented into asset & personnel tracking, location-based analytics, navigation & maps, and others.

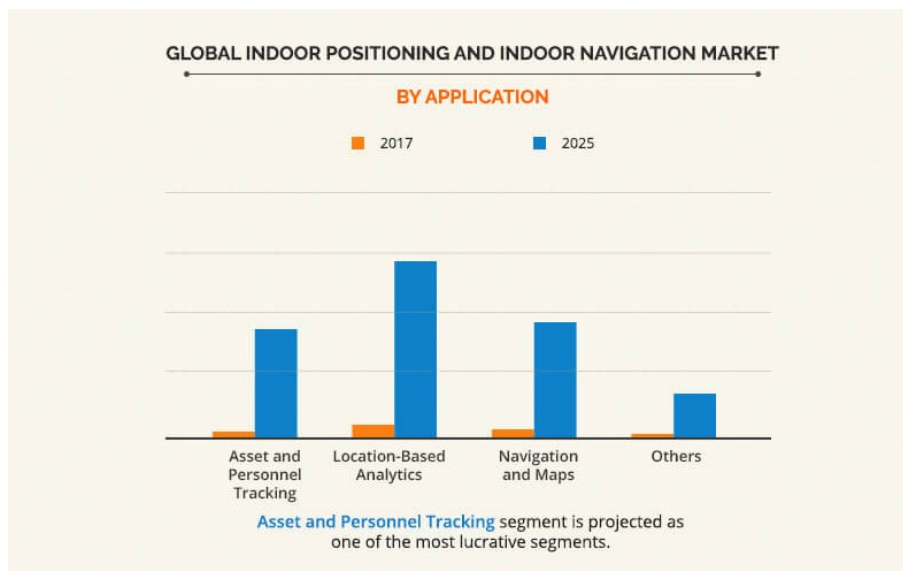


Figure 1.1: Global Indoor Positioning and Indoor Navigation Market by Application [3].

The global indoor positioning and indoor navigation (IPIN) market is segmented based on component, technology application, end-use industry, and region. In terms of component, the market is categorized into software, hardware, and services. Based on technology, it is divided into ultra-wideband technology, Bluetooth low energy, Wi-Fi, and others [3].

Factors such as increased Bluetooth beacon adoption, increased demand for proximity marketing, and increased IoT adoption in indoor positioning are expected to drive market growth. However, operational challenges are expected to hamper the global indoor positioning and indoor navigation market growth during the forecast period. Furthermore, IPIN systems in the healthcare sector are projected to present lucrative chances for the worldwide indoor positioning and indoor navigation market to expand [3].

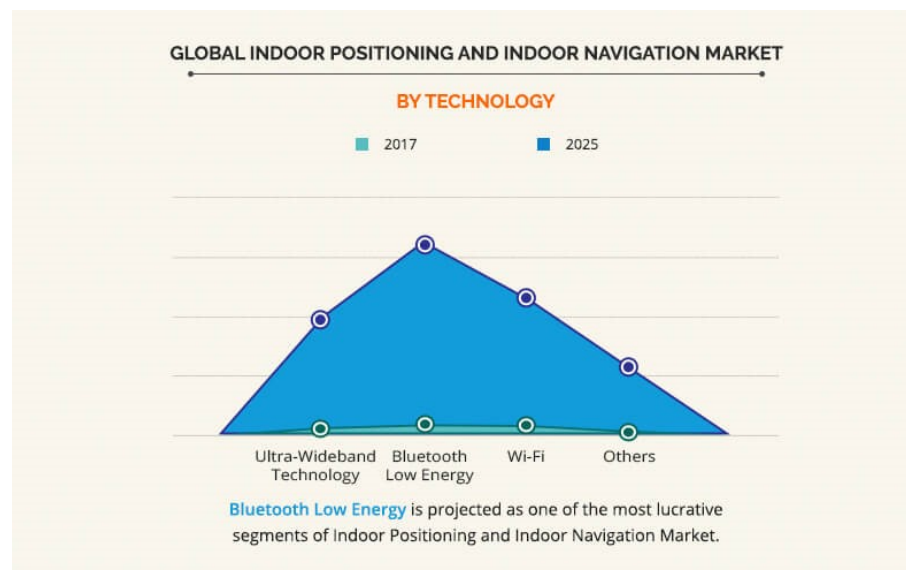


Figure 1.2: Global Indoor Positioning and Indoor Navigation Market by Technology [3].

Beacon infrastructure contributes significantly to the generation of contextual and hyper-local data, which is critical for the retail industry. Retailers and wholesalers can engage customers by providing highly relevant content for shopping based on their precise location, as well as collecting relevant data for marketing and streamlining operations. Furthermore, factors such as its low deployment cost and wide applicability in a variety of mobile devices are expected to drive the growth of the indoor positioning and indoor navigation market. Beacons are also simple to deploy and have reasonable features, which has resulted in a significant increase in their adoption rate over the years [3].

1.3 Context and Objectives

RACELAND S.A., an international vehicle manufacturer, has been faced with the need to manage production times. Given the dispersion of tasks, mostly handmade, resources expended could be optimized, suspecting that excessive hours are consumed, harming customers.

The company invests in management by objectives, in the planning and dynamic management of the performance of processes and in its continuous improvement.

It is intended that the most varied tasks distributed are tracked according to the position of the employee and the location of those tasks, saving an history of the activity and data for performance evaluation.

It is believed that with this system the company will improve their future budgeting, by optimizing their costs and human resources and create procedures for task automation.

The singularity of the tasks, makes it extremely difficult to manage times for the budgeting of future identical works.

The main objectives for this project are:

- Develop a positioning system for tracking personnel and assets;
- Evaluate the accuracy and viability of the estimator implemented;
- Integrate the data obtained with the company's management system (PHC CS Advanced);

1.4 Document's structure

This document is organized in 5 chapters that present the thesis' theme:

- Chapter 1 states the motivations behind the project's development, and the opportunity for indoor positioning systems. It also introduces the context and objectives for the work.
- Chapter 2 presents the state of art in localization with the most notable possibilities and technologies already investigated or available on the market. Afterwards, presents a systems engineering framework, which is a useful standard when developing a localization system. This chapter also approaches the different dimensions and aspects that can influence the location's information quality. The most notable techniques that enable localization and the message transport protocols used in the project are then described. The last part of this chapter is used to discuss the privacy and security issues when developing these kind of systems.
- Chapter 3 will present the project to be developed with an overview of the environment where it will be implemented. In this chapter will also present the features required by the company with an explanation of how those features were decomposed in technical requirements for development purposes. Afterwards, the system's architecture is modeled and the database structure explained.
- Chapter 4 describes the path and methodology used. The algorithm for each process decomposed in Chapter 3 is illustrated using a sequence diagram. All the concepts that constitute them are explained and the experiments and evaluations performed are presented.
- Chapter 5 exposed what has been achieved with some considerations. This chapter is used to reflect the system's strengths, limitations, and what can be done to improve it. It ends with some final remarks.

Chapter 2

Literature review and fundamentals

The state-of-the-art in indoor localization systems will be given in this chapter. To this end, certain prominent technologies and recent applications will be discussed.

Afterwards, the subject of localization will be analysed in different dimensions. A systems engineering framework is proposed. This should be considered as a foundation when designing a modular and configurable system. Once the structure is settled, a research about the different aspects that can affect the information quality is presented. These aspects will be split into two categories: spatial and temporal.

The most significant type of information that can enable localization are then exposed. Following that, some of the most notable techniques are presented, along with examples of how they function.

The iBeacon and the MQTT messaging protocols are then explained, which will be important for the project. The last part of this chapter is dedicated to discuss the privacy and security issues when developing these kind of systems.

2.1 State of Art

A review on the state of the art shows that the indoor localization problem was already addressed by many approaches, each of them with their advantages and disadvantages.

By understanding each possible approach it is possible to define the most fitting technology to solve our problem. Following this section, the most notable techniques will be listed alongside an explanation of their use cases.

Wi-Fi - IEEE 802.11

WiFi was released in 1997 and is a group of wireless network protocols, based on the IEEE 802.11 family of standards, which are commonly used for local area networking (WLAN) of devices and Internet access. Their main operating band frequencies are 2.4GHz or 5GHz and to connect to the Wireless Local Area Networks (WLAN) a wireless access point is required [4].

The main idea to use WiFi for localization purposes is the surge of IoT (Internet of Things) devices, which are devices that have a WiFi Network interface card or chip (NIC) and thus can connect to the access points and send data payloads. Time of flight and RSSI measurement values are the most common ways to retrieve the devices location, with access point serving as anchors or fixed reference points.

In the meanwhile, there are some concerns as WiFi networks are deployed for communication, so while connectivity and data rate are a high priority, localization is not their main focus. At the same time, the wide availability of WiFi can pose some challenges in the near future. As the number of the devices that have access to the medium increases, it becomes overcrowded and interference problems may arise [4].

BLUETOOTH LOW ENERGY - BLE

Bluetooth Low Energy (BLE) was first presented in 2010 by Bluetooth Special Interest Group. The main focus for BLE was to transfer large amounts of data while reducing the power consumption and the cost of the devices [4]. Similar to wifi, the surge of BLE for localization purposes came with the increase number of IoT devices, due to the lower power consumption and the number of applications that could be used.

The main advantage of BLE compared to WiFi is that BLE has much lower energy requirements, hence, its bitrate is also lower. BLE is ideal for short range periodic transmission of small amounts of data [4].

One type of BLE device that has been created is known as a beacon. Beacons are small, inexpensive devices which periodically transmit packets of information to all nearby BLE enabled devices [5].

Two of the most common protocols for beacons to this date are Eddystone and iBeacon. Even though both serve the same purposes, they have slight differences.

iBeacon is a class of Bluetooth Low Energy (BLE) devices that broadcast their identifier to nearby portable electronic devices. iBeacon differs from some other location-based technologies as the broadcasting device (the beacon) is only a 1-way transmitter to the receiving device, and necessitates a specific application or system to interact with the beacons [5].

Eddystone is a Bluetooth Low Energy beacon profile released by Google in July 2015. Though similar in concept to the iBeacon profile, Eddystone can be implemented without restriction. Eddystone also contains a telemetry frame designed for reporting on a beacon's health, including, for example, battery level. Like any other beacon technology, beacons with Eddystone can give devices a better indication of what objects and places are around them, specially at close range [6].

LoraWAN

LoRa (Long Range) is a low-power wide-area network (LPWAN) modulation technique originally developed by the LoRa Alliance.

LoRa uses license-free sub-gigahertz radio frequency bands like 433 MHz, 868 MHz (Europe), 915 MHz (Australia and North America), 865 MHz to 867 MHz (India) and 923 MHz (Asia).

LoRa enables long-range transmissions up to 15km with low power consumption. The technology covers the physical layer, while other technologies and protocols such as LoRaWAN (Long Range Wide Area Network) cover the upper layers. It can achieve data rates between 0.3 kbit/s and 27 kbit/s depending upon the spreading factor [7].

LoRaWAN is more secure than other wireless technologies for IoT since it can transmit encrypted data at a different frequency. Its wide transmission range makes it ideal for a variety of smart city programs. The disadvantage of using such a low frequency is a reduction in the data rate that can be transmitted between devices. However, for indoor localization the data rate is not an issue as the nodes are not transmitting large amounts of information. Due to the 868MHz band for Europe being unlicensed, it is free for anyone to use for their personal networking needs. However, cost is often an issue when using LoRaWAN as large antennas and extra hardware are required to access the medium. LoRaWAN is effective for long range outdoor localization, but in short range indoor localization can pose some challenges [4].

ZIGBEE - IEEE 802.15.4

ZigBee is a wireless communication protocol for IoT devices, focusing on low power devices. These, in turn, are equipment that have a limited energy demand, with autonomy that can reach years with a single battery. The technology was designed to interconnect these units of smart devices, creating a kind of ZigBee network, independent of Wi-Fi.

Zigbee is known for its simplicity, low-power usage, and secure networking capabilities. Zigbee is based on the IEEE 802.15.4 standard, which defines the operating point for wireless personal area networks (WPANs) with low-data rate antennas. Devices using IEEE 802.15.4 are able to control the flow of information and prevent any loss of data by using carrier-sense multiple access with collision avoidance (CSMA/CA) [8].

Devices using Zigbee are designed with features such as link quality and energy detection, which allow for measurements such as the RSSI to be easily determined [4].

Zigbee has a greater range than BLE as it can transmit farther by using a mesh network of relay nodes to reach a destination. Zigbee is commonly used for WSN (wireless sensor network) localization due to its low power requirements. However, extra hardware is required which makes it less popular among current IoT users [4].

RFID

RFID belongs to a group of technologies referred to as Automatic Identification and Data Capture (AIDC). AIDC methods automatically identify objects, collect data about them, and apply that data directly into computer systems with little to no human intervention. RFID methods utilize radio waves to accomplish this. At a simple level, RFID systems consist of three components: an RFID tag or smart label, an RFID reader, and an antenna. RFID tags contain an integrated circuit and an antenna, which are used to transmit data to the RFID reader (also called an interrogator). The reader then converts the radio waves to a more usable form of data. Information collected

from the tags is then transferred through a communications interface to a host computer system, where the data can be stored in a database and analyzed at a later time [9].

There are two kinds of RFID systems that exist: passive and active.

Passive RFID systems use tags with no internal power source and instead are powered by the electromagnetic energy transmitted from an RFID reader. Passive RFID tags are used for applications such as access control, file tracking, race timing, supply chain management, smart labels, and more. The lower price point per tag makes employing passive RFID systems economical for many industries.

Active RFID systems use battery-powered RFID tags that continuously broadcast their own signal. Active RFID tags are commonly used as “beacons” to accurately track the real-time location of assets or in high-speed environments such as tolling. Active tags provide a much longer read range than passive tags, but they are also much more expensive [9].

Using RFID in indoor positioning can have great advantages as it's usually a simple system with high portability, flexibility and ease of maintenance. The devices are cheap, capable of providing both identification and location and have relative long effective range (up to 1000 m for a single transmitter in free space). However, there are also some disadvantages to consider when building such system: they have only one-way communication links, may have problems with penetration and multipath effects and have unstable Received Signal Strength (RSS) [10].

InfraRed

Infrared (IR), sometimes called infrared light, is a type of electromagnetic radiation (EMR) with wavelengths longer than those of visible light. It is therefore invisible to the human eye.

Infrared signals are most of the times used for coarse localization as the precision of the system greatly depends on having line-of-sight between the infrastructure and the device to be tracked. It's low transmission range of the transducers is also a constriction for use, especially in conditions where there is sunlight influence [11].

Between the many possibilities of this type of systems, one of the first and the most known is the “Active Badge Location System” from 1992. In this system, a person wears a “badge” that broadcasts a unique code every 15 seconds using InfraRed light and this burst of invisible light would be received by sensors strategically placed [2]. The position of an active badge can be specified by the information sent from these sensors, which are connected by wires and forwards the location information of the tracked active badges to a central server [12].

Some of the advantages of such technology is that infrared radiation does not penetrate walls and therefore localization is at least confined to the inside of the room. Moreover this radiation does not interfere with electromagnetic signals. However, these systems typically provide room-granularity accuracy and are not considered as a possibly universal solution for indoor localization as they require proprietary hardware and have limited possibilities of use, like line-of-sight and controlled illumination [11].

Ultra Wide Band (UWB)

Since the FCC first opened up the ultra-wideband (UWB) radio spectrum for unlicensed commercial use in 2005, UWB location systems have surged in interest as one of the IoT technologies to lookout for. This is largely due to the level of location precision and penetration of obstacles afforded by ultra-wideband positioning systems. UWB indoor positioning systems are able to pinpoint location in real time to within 20 centimeters or less [13].

UWB can achieve a high percentage of accuracy and penetration thanks to the large swath of frequencies it uses to transmit messages. Because the ability to accurately measure the “time of flight” of any radio wave is a function of bandwidth, signals that are very broad in frequency can be accurately measured in terms of the time it takes for an encoded signal to travel from a transmitter to a receiver. The use of a wide-frequency spectrum means that the signal can be short in length. UWB signals are transmitted with low power, preventing interference with other systems using the radio spectrum [13].

For applications where pinpoint location accuracy is needed for productivity or safety, ultra-wideband location technology might be the best option. However due to the cost and complexity of installing a UWB real time location system, the level of accuracy for UWB is usually disregarded when compared, for example, to the cost of a BLE system with beacons, which can achieve accuracy within a one-to-two meter range.

In addition, UWB location systems require at least three receivers to receive signals from tags. These receivers are very expensive and they must be precisely synchronized down to the nanosecond for the system to calculate location accurately. In many installations, the receivers are connected by cables to keep them in sync, which adds to the installation cost and difficulty mentioned [13].

Ultrasonic (sound-based systems)

Another potential source of location data is ultrasonic sound. The ultrasound signals are positioned above the frequency range of human audibility, as Figure 2.1 depicts. Ultrasonic indoor localization technology measures distance between fixed-point reference devices and the target mobile device. It typically requires special transducers either for transmission either of reception.



Figure 2.1: Frequency range of sound with distinction of the three main sections: infrasound, audio and ultrasound [11].

Generically, the several approaches require synchronization between the mobile device and the infrastructure, which may be achieved by infrared or radio frequency signals. In the latter, the transmitter sends simultaneously a radio frequency signal and an ultrasonic wave. The radio frequency signal reaches receivers almost instantaneously (at approximately the speed of light),

providing a good enough synchronization signal. At the receiver's end, the time difference between the synchronization signal and the detection of ultrasonic waves is measured, enabling the distance estimation.

The ultrasound-based localization systems have narrowband or wideband approaches with associated advantages and limitations. Narrowband signals travel longer distances and need less power to be emitted. However, in situations where there are several simultaneous emitters, difficulties may arise in the identification of each emitter due to the fact that the receiver may not be able to distinguish them. A wideband usage overcomes this multiple access problem using spread spectrum signaling techniques which provide more robustness to overcome interference. Yet, a wider frequency band will also fade more in distance and therefore will require more power on the emission [11].

Because ultrasound transducers are relatively hard to find and to use in different applications, they are not applicable in a large scale. The user needs to carry a custom device capable of receiving ultrasounds radiated by a network of emitters in the infrastructure. The problem is that these solutions require specially developed hardware. Furthermore, ultrasonic signals require as much as line-of-sight possible between emitting the emitting devices and the target mobile device as the wavelength of this type of signal is relatively short and the diffraction or spreading may occur. As such, to ensure a wide coverage of area, it would be required a large number of devices transmitting ultrasonic signals.

The advantages of ultrasound systems are associated with the fact that these signals are relatively slow (when comparing with radio-frequency and other faster signals) and clock synchronization is easier and does not require any expensive hardware or techniques. Therefore it provides accuracy at smaller costs. Using ultrasound or near ultrasound also simplifies the risk of annoying people or disturbing the acoustic environment. In the systems that use audio hardware and explore the upper frequency limits of audio off-the-shelf devices, the only apparent problem is related with directivity and consequently signal coverage. More loudspeakers and more sensitive microphones become necessary increasing costs significantly [11].

Computer Vision-based systems

Artificial intelligence and computer vision are also a possibility when it comes to the design of a localization system.

There are many ways to approach localization with computer vision-based systems. For example, in a research paper with title "*Indoor Localization using Computer Vision and Visual-Inertial Odometry*" demonstrated the feasibility of a novel indoor localization approach. Their method provided indoor wayfinding for people with visual impairments, who are often unable to see visual cues such as informational signs, land-marks and structural features that people with normal vision rely on for wayfinding [14].

Their localization approach used a smartphone, held by the user or worn on the chest, to acquire several video frames per second. By using OpenCV, which is an open source computer vision framework, it is possible to recognize barcode-like signs printed on paper. Using the visual

queues and the identification of the barcode, they could translate that information to a 2D map of the indoor environment to specify the location and orientation of the user [14].

Although it's a pretty novel solution, it also comes with some limitations as the accuracy greatly depends on the computer vision algorithm implemented. Furthermore, it's pretty demanding and requires a computer or microcontroller with fast processing speed, which usually means higher power consumption.

2.1.1 Technology comparison and discussion

A flawless solution for the indoor localization problem doesn't exist yet. Each technology has its own set of advantages and disadvantages. Some will necessitate pricey installations, while others will require custom hardware. Some technologies are more susceptible to environmental conditions than others, while others are not. The optimum indoor localization technology is determined by the application's requirements and limits. The ideal scenario would be to have a technology that can accommodate all situations and possibilities. However, there is no candidate indoor localization technology that achieves this, and the majority of them are far from it [11].

The Figure 2.2 below is used to compare each technology previously presented in terms of cost, power consumption and accuracy.

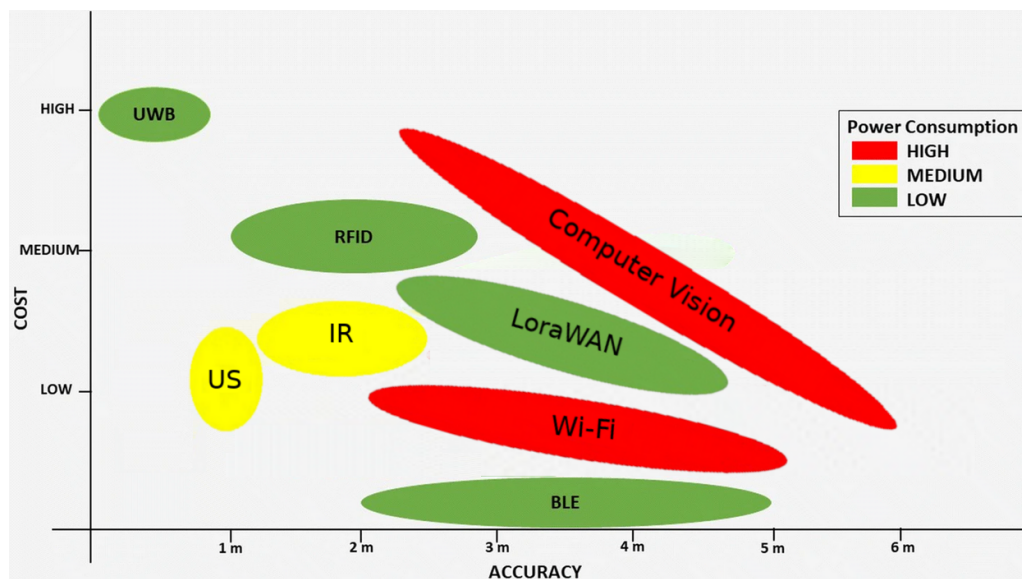


Figure 2.2: Comparison of indoor positioning technologies [15].

Ultra Wide Band (UWB) and Ultrasound (US) are ideal for demanding applications where accuracy is critical, as they may attain accuracy of less than one meter. Although they strive in precision and accuracy, there are some concerns when using these technologies. UWB based positioning systems can provide centimeter-level accuracy, which is far better than the other methods, however they are the most costly and complex to implement and manage. Audio sound systems have a possibly serious disadvantage of being heard by people if there is no attention with the

signal used. They are however not expensive to implement since they use very simple technology, present everywhere and with everyone, such as mobile phones and microphones.

In terms of RSS-based (relative signal strength based) systems, the BLE technology stands out for its low cost of equipment and implementation, making it a very attractive technology when high accuracy is not crucial. LoRa based systems also stand out for their signal reach, with transmissions of up to 15km, making it preferable for wide areas, such as parking lots. Some concerns with these technologies are the environmental problems like wall attenuation, reflections, obstacles or significant changes in the electromagnetic environment, causing bad localization results [11].

In case the project involves a great number of IoT devices, the Wi-Fi technology should be considered, since the telemetry signals coming from these devices can also be used to calculate their position. In other scenarios, it is not advised because the same outcomes can be obtained at a lower cost and with less energy use.

RFID systems have several advantages, such as portability, flexibility, and ease of maintenance, but they can be costly because they require nearly as many beacons as spots to be located.

Artificial Vision systems are expensive and require more development since recognition in uncontrolled spaces still produce many false positives. Occlusions and lighting conditions affect significantly their results [11].

2.2 Systems engineering framework

In the article paper, "*The Location Stack: A Layered Model for Location in Ubiquitous Computing*" from 2002, Jeffrey Hightower, Barry Brumitt and Gaetano Borriello defined and proposed the Location Stack, a layered software engineering model for location in ubiquitous computing. The main idea for this framework is to create standard infrastructures and common vocabulary, permitting members of the ubiquitous computing community to easily evaluate and build on each other's work [16].

Their model is similar to the seven-layer Open System Interconnect (OSI) model for computer networks and can be shown in the following Figure 2.3.

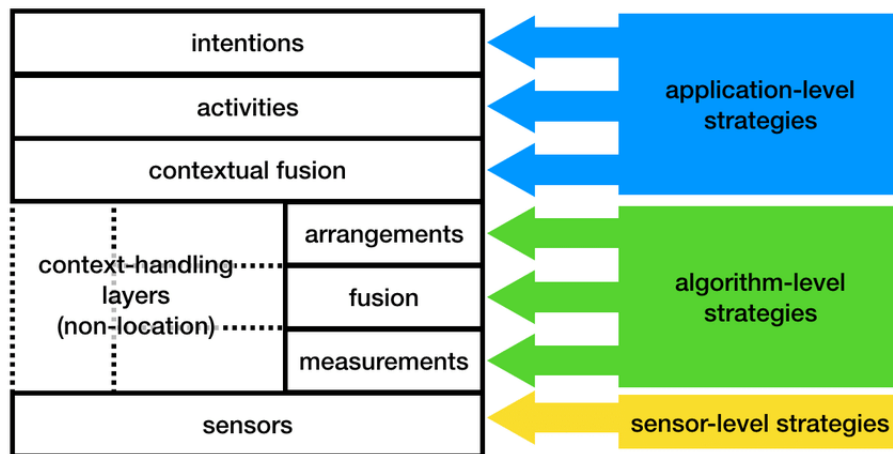


Figure 2.3: 7 layer model of the Location Stack [17].

The 7 layers can be briefly explained, from bottom to top as follows:

- Sensors (Layer 1)

This layer is responsible to deal with all data coming from the sensors of the localization system.

It outputs the raw data of those sensors to the layer above.

- Measurements (Layer 2)

Belonging to the algorithm-level strategy, it's main focus is to convert the raw data into its canonical format like: proximities, distances or angles.

It exports the converted stream of distance, angle, proximity, asserted position, or non-geometric feature measurements [16].

- Fusion (Layer 3)

This layer is a general method of continually merging "layer 2's" streams of measurements into a time-stamped probabilistic representation of the positions and orientations of objects. Through measurement fusion, differing capabilities, redundancies, and contradictions are exploited to reduce uncertainty [16].

It exports, at minimum the immediate estimated location of individual objects and the uncertainty of those locations.

- Arrangements (Layer 4)

It's the engine for interrelating the positions of the objects, for example by converting their relative coordinates to an absolute coordinate system. It probabilistically reasons the relationships (e.g. proximity, containment, geometric formations) between two or more objects [11].

The Arrangements layer creates a query or event interface to the relationships between two or more objects which are individually locatable using the previous layer interface.

- Contextual fusion (Layer 5)

The contextual fusion is a system for merging location data with other non-location contextual information. For example: tasks, temperature, calendar date, color and so forth [16].

It's an interface to allow applications to recognize interesting states, sequences, or situations presumably to take predictive or responsive action on behalf of users.

- Activities (Layer 6)

It's a high level system, such as a machine learning system, for categorizing all available context information including location into tasks or activities. To note that activities differ from context in that activities are an application's interpretation of the state of the world given both location information and other state information, whilst context is what defines the activity [16].

- Intentions (Layer 7)

It's the highest layer and the objective to what the localization system or application should do to fulfil the cognitive desires of the users, like, for example, providing routing or in the case of the project at hand: provide a BI (Business Intelligence) interface for managing tasks in a company.

2.3 Location Information Quality

In a relative recent paper from 2018, named "*Location Information Quality: A Review*", Champika Ranasinghe and Christian Kray combined seven different quality aspects in a model, as can be seen in Figure 2.4. With some research they figured that accuracy, precision, granularity, coverage, conflicts, update rate and recency showed more frequently to describe quality or used to represent quality. They also split these aspects into two different dimensions, being that accuracy, precision, granularity, coverage, conflicts belong to spatial dimension and the other two remaining, update rate and recency, belong to the temporal dimension [17].

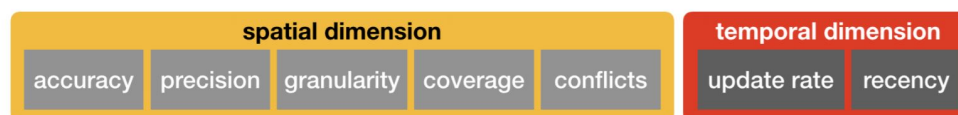


Figure 2.4: Aspects representing quality of location and orientation information along the spatial and temporal dimensions [17].

Each aspect of these two dimensions will be described, with a brief explanation of what they mean.

Spatial dimension aspects

The spatial dimension's aspects can be described as follows:

- **Accuracy** is a measurement of how closely it matches the accepted standard for that measurement or the true value. For example, we can model location accuracy as a radius in meters around the measured location of a person or object and assume that in that radius it's the real position.
- **Precision** is a measurement of how much the values of repeated measurements differ from other measures of the same thing, under the same conditions.
- **Granularity** is the amount of detail considered in a model or decision making process. A greater amount of granularity is correlated to a deeper level of detail [18]. For defining a location, this value can be given in meters, where as for cell-based localization systems the cell size may define the granularity and for continuous localization systems it is correlated to the maximum achievable accuracy and precision [17].
- **Coverage**, is defined coverage as the "[...] *extent of space in which an entity's position can be sensed*". [19]. In the example of the thesis' project it can mean the amount of range at which the access point can't sense the data advertised by the beacon or the interference that could cause a similar behavior.
- **Conflicts** or ambiguity in information is the uncertainty or error in the orientation or position measurements of a person or object. Like for example in the BLE Beacon case it can give contradictory data due to variations in signal strength.

Temporal dimension aspects

As for the Temporal dimension's aspects:

- **Update rate** is the frequency or the sampling interval, meaning how often the "sensor" measures a property. In the case of the thesis the frequency of payload advertisements.
- **Recency** or "update time" can be described as the time elapsed since the last measure. "The less recent location information is, the more uncertainty can arise [17]."

2.4 Localization-enabling information

This section is aimed to present the different ways sensors and devices can provide meaningful data to determine a person or object location. A localization system can estimate or provide cues of the position of certain individuals or objects in a contextual environment through the acquisition of measurements.

The most significant type of information for localization systems are:

- Time of flight (TOF) or Time of Arrival (TOA);
- Phase shift or Time Difference of Arrival (TDOA);
- Angle of Arrival (AoA)
- Signal strength (RSSI)

As previously mentioned in section 2.2 (Systems engineering framework), this concepts of information are the foundation of the "*sensors layer*" of the "*location stack*" by outputting time, phase or signal strength to the "*measurements layer*" in order to obtain proximity and distances.

Time of flight

Time of flight (TOF), also known as Time of Arrival (TOA), is the measurement of the time taken by an object, particle or wave (be it acoustic or electromagnetic) to travel a distance through a medium. Because time can be converted to distance by multiplying it with the speed of the signal, it is a measure of the distance between the receiver and the transmitter. This distance can then be used to determine the location. Given two devices with their clocks synchronized. If the time of the wave emission and arrival are known [11]. Let's define t_0 as the emission time, and t_f the time of arrival, the time of flight (TOF) can be calculated as:

$$t_{TOF} = t_f - t_0 \quad (2.1)$$

As for the distance between the objects, d_f , for the measurements layer, it can be derived from the travel velocity, v_w , of the wave as such:

$$d_f = v_f \cdot t_{TOF} \quad (2.2)$$

One of the main drawbacks of this measurement is that it requires precise time synchronization between the transmitter and the receiver for accurate results.

Phase shift

Phase shift or time difference of arrival (TDOA) takes the difference between two TOF (Time of flight) measurements between the device to be located and two (or more) synchronized fixed devices. Given a continuous periodic signal, if the signal generated and transmitted by the anchors are the same, it is possible to perform a cross correlation with the received signal. If both sides

are perfectly synchronized, the result of this operation yields the phase difference Φ . This phase difference is proportional to the distance between the two objects d_Φ . This distance can then be computed as

$$d_\Phi = v \cdot T \cdot \frac{\phi}{2\pi} \quad (2.3)$$

where T represents the signal's period and v the wave's propagation velocity. To avoid solutions that are multiple to the period of the signal the condition $d < v \cdot T$ must hold. This eliminates the requirement of precise time synchronization between the transmitter and the receiver. However, clock synchronization at the network level is required [11].

Angle of arrival

Angle of Arrival (AoA) based systems use the direction that the received signal approaches from to determine the relative angle from the known location. Measurements of AoA can be done by determining the direction of propagation of a radio-frequency wave incident on an multi-directional antenna array or determined from maximum signal strength during antenna rotation [17].

In the case of an antenna array station, the AoA can be calculated by measuring the time difference of arrival (TDOA) between individual elements of the array. This TDOA measurement is made by measuring the difference in received phase at each element in the antenna array as mentioned previously. The delay of arrival at each element is measured directly and converted to an AoA measurement.

This measurement is independent of clock synchronization and accuracy can be high, however it can add significant hardware cost.

Signal strength

Received Signal Strength Indicator (RSSI) is one of the most well-known and commonly used characteristics for indoor localization. It is the measurement of the power present in a signal sent from a device, like a beacon in the case of the thesis, at the moment of the reception, like at the gateway. RSSI can translate the measurements to distance estimates in meters [20]. More precise, it is possible to describe the relation between RSSI and distance using a model, the Log-Distance pathloss model:

$$RSSI = -10 \cdot n \cdot \log_{10} \left(\frac{d}{d_0} \right) + A_0 \quad (2.4)$$

In this model, d describes the distance between the transceiver and recipient, n the signal propagation exponent and A_0 a referenced RSSI value at d_0 . Usually d_0 is taken to be 1 such that A_0 becomes the signal strength measured at a distance of 1 meter of the device.

A_0 can usually be retrieved from the device itself as it is often part of the broadcast message. Alternatively, it is possible to approximate A_0 by collecting data and averaging the signal strength

measurements. The signal propagation exponent n is a constant that differs from environment to environment. For indoor applications, n is often set to values greater than 2 [20].

RSSI techniques are among the cheapest and easiest methods to implement, but they do not provide the best accuracy. Filtering is necessary to improve system accuracy using RSSI-based localization [4].

2.5 Location measurement techniques

The location estimation techniques determine the target device's location using distance, direction (angle) or proximity information that is acquired by the use of measurement techniques. This section will present the most notable methods. A better understanding of these methods allows to make an educated choice for the most effective method for the thesis' project.

Proximity-based method

Proximity-based localization determines the position of a target device based on its presence in a given area designated by a fixed reference point. The location of the reference point is taken as the location of the target.

A fixed reference point, also known as anchor could be, for example, a cell tower, a beacon, a wireless access point, an IR reader or an RFID reader. For example, in a cellular network, the location of the base station of the cell with the highest RSSI can be taken as the location of the user [17].

Robustness is the mostly appreciated advantage of these techniques. If the localization of these systems are known, then the localization of the detector is the same. The granularity of localization can be as small as the proximity of the device [11].

Distance-based methods - Trilateration

Distance-based Methods use distance measurements derived from TOA, TDOA or RSSI measurements to find the location of a target. Os dois mais comuns são trilateração e multilateração

In this subsection the trilateration algorithm, one of the most used distance-based methods, will be presented, describing the set of equations and principles to achieve the coordinates of a target device.

The trilateration technique computes the localization of a target device, P , given the distances d_1, d_2, d_3 to three fixed non-collinear reference anchors.

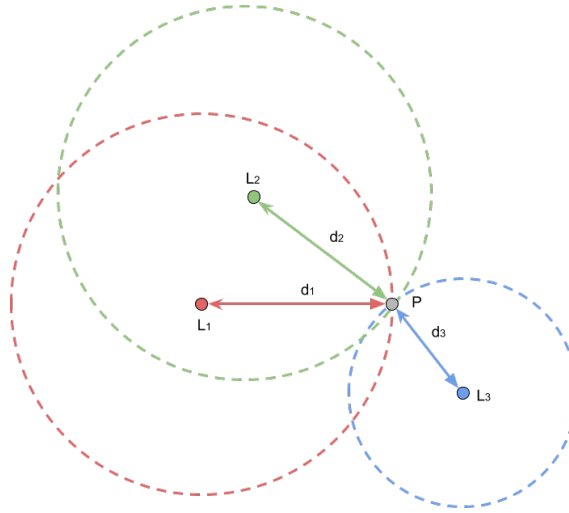


Figure 2.5: Trilateration method overview in a 2 dimensional plane.

In this method a 3 dimensional plane will be considered, but the result of the coordinates will be in a 2D plane. For each of the three anchors the feasible positions of P can be described as a general equation of a sphere centered at that anchor device. The expressions for these spheres are shown in the following equations:

$$d_1^2 = (x_P - x_1)^2 + (y_P - y_1)^2 + (z_P - z_1)^2 \quad (2.5)$$

$$d_2^2 = (x_P - x_2)^2 + (y_P - y_2)^2 + (z_P - z_2)^2 \quad (2.6)$$

$$d_3^2 = (x_P - x_3)^2 + (y_P - y_3)^2 + (z_P - z_3)^2 \quad (2.7)$$

These equations are independent and non-linear which cannot be solved directly. However, using the method proposed by [20], which assumes that each anchor device is at the same height difference of Z from the target device P , equation (2.7) can be subtracted from equation (2.6) to get the following linear equation:

$$d_2^2 - d_3^2 = 2 \cdot x_P \cdot (x_3 - x_2) + x_2^2 - x_3^2 + 2 \cdot y_P \cdot (y_3 - y_2) + y_2^2 - y_3^2 \quad (2.8)$$

The same can be made by subtracting equation (2.5) from equation (2.6) to obtain a second linear equation:

$$d_2^2 - d_1^2 = 2 \cdot x_P \cdot (x_1 - x_2) + x_2^2 - x_1^2 + 2 \cdot y_P \cdot (y_1 - y_2) + y_2^2 - y_1^2 \quad (2.9)$$

From the equation (2.9) it is possible to simplify and reduce the computational task by producing the change of variable v_1 as follows:

$$x_P \cdot (x_3 - x_2) + y_P \cdot (y_3 - y_2) = \frac{(d_2^2 - d_3^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2)}{2} = v_1 \quad (2.10)$$

The same goes for equation (2.9) to produce the change of variable v_2 as follows:

$$x_P \cdot (x_1 - x_2) + y_P \cdot (y_1 - y_2) = \frac{(d_2^2 - d_1^2) - (x_2^2 - x_1^2) - (y_2^2 - y_1^2)}{2} = v_2 \quad (2.11)$$

Solving the equation (2.10) and equation (2.11) for the intersection point (x_P, y_P) will result in the following equations for y_P and x_P value respectively:

$$y_P = \frac{v_2 \cdot (x_3 - x_2) - v_1 \cdot (x_1 - x_2)}{(y_1 - y_2) \cdot (x_3 - x_2) - (y_3 - y_2) \cdot (x_1 - x_2)} \quad (2.12)$$

$$x_P = \frac{v_1 - y_P \cdot (y_3 - y_2)}{(x_3 - x_2)} \quad (2.13)$$

The result will be the coordinates of the position of the device P.

Distance-based methods - Multilateration

Another commonly used distance-based method is the multilateration algorithm. Unlike the trilateration, the multilateration can take in more than three anchor measurements to compute the position of the target at once. This subsection will be dedicated to explain the multilateration algorithm while also providing one of the possible approaches to estimate a target location.

Even with accurate sensors, it is unlikely that at least three circles would intersect at exactly the same point at which the target is located. This means that attempts to tackle this problem analytically will almost always fail. Instead, an optimization-based approach is used [21].

One common approach to infer a target's location is by finding the point which minimizes the squared euclidean distances between this point and all the circle circumferences.

Let p denote the target's position, x_i the i^{th} gateway's location and r the radius of the circle. The function L which calculates the euclidean distance between the target and all k circles is then:

$$L(\mathbf{p}) = \sum_{i=0}^{k-1} (\|x_i - \mathbf{p}\|_2 - r_i)^2 \quad (2.14)$$

The optimal value of p , p^* is then found as:

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} L(\mathbf{p}) \quad (2.15)$$

To solve for the optimal position an optimization procedure is used. One example could be the Newton-Conjugate-Gradient algorithm. Newton-Conjugate Gradient algorithm is a modified

Newton's method and uses a conjugate gradient algorithm to approximately invert the local Hessian matrix [22]. Newton's method is based on fitting the function locally to a quadratic form:

$$f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0) + \frac{1}{2} \cdot (x - x_0)^T \cdot H(x_0) \cdot (x - x_0) \quad (2.16)$$

where $H(x_0)$ is a matrix of second-derivatives (the Hessian). If the Hessian is positive definite then the local minimum of this function can be found by setting the gradient of the quadratic form to zero, resulting in

$$x_{opt} = x_0 - H^{-1} \cdot \nabla f \quad (2.17)$$

The inverse of the Hessian is evaluated using the conjugate-gradient method.

Angle-based method

Angle-based methods constitute a class of localization algorithms that use direction or angle measurements to estimate distances to anchors or reference points, and from these distances a procedure is used to determine the optimal position of a target device.

Triangulation is one of best known and is the process of determining the location of a point by forming triangles to it from known points. In triangulation, initially angle of arrival (AoA) measurements are gathered at the sensor node from at least three fixed anchor nodes and, to localize the target node device, geometric operations and properties of triangles are applied on the measurements to estimate the position, as Figure 2.6 depicts [23].

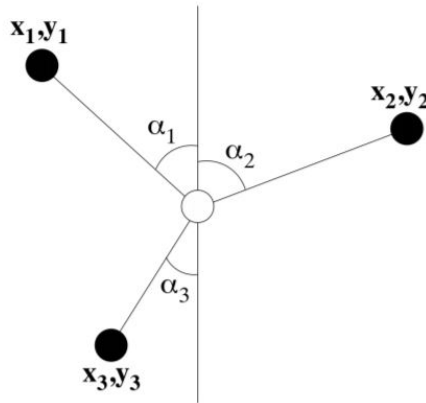


Figure 2.6: Triangulation method overview [23].

The expression 2.18 below translates the relationship from each angle, α_i , and the coordinates of the respective anchor, (x_i, y_i) and the target device, (x_P, y_P) .

$$\tan(\alpha_i) = \frac{y_i - y_P}{x_i - x_P} \quad (2.18)$$

The estimate position is then given by solving these expressions and using an optimization procedure, like for example the Gauss–Newton algorithm, to enhance the accuracy and minimize the error of the results.

2.6 iBeacon protocol fundamentals

Considering all the technologies and techniques for localization measurement, the technology chosen was BLE with the iBeacon protocol, as it's practical and cheap to implement in medium scale environments. This section aims to explain the iBeacon technology, it's design, common uses, the advantages and characteristics.

Apple introduced, at the Apple Worldwide Developers Conference in 2013, a proprietary standard called "iBeacon" based on the Bluetooth 4.0 Low Energy (BLE) specification. This technology was designed to enable additional location-based services. BLE introduced a new advertisement mode. Their first purpose for this advertisement mode was to build low-cost and low-power devices or sensors like smart-watches or fitness wristbands. The key idea behind this concept is to use cheap and durable Bluetooth senders. This senders, now called iBeacons, broadcast advertisement packets in a specific interval using the BLE advertisement channel, comparable to a lighthouse lighting showing their location. The iBeacon protocol defines a specific data structure for these advertisement packets, which is shown in Table 2.1 below.

Header	Manufacturer	Company ID	ADV ID	UUID	major	minor	Tx Power
64bit	8bit	16bit	16bit	128bit	16bit	16bit	8bit

Table 2.1: Structure of a iBeacon payload [24].

For the project at hand, the most important fields are the last four, UUID, major, minor and Tx Power, which can be configured to our needs, and thus, for localization. The first three, UUID, major, minor are usually used to identify a specific sender and give context. UUID is typically used to describe the company whereas major is used to identify the environment, like buildings or floors, and minor is used to identify specific beacons, like type of object or person. The last field is the most important one and can be used to measure distances and proximity. the Tx Power field, usually calibrated by the manufacturer of the beacon device, describes the RSSI (Relative signal strength) decibel value in 1 meter distance [25].

The iBeacon specification describes no additional field for payload data which can be used for localization. The combination of several of these devices is what makes it possible to apply location techniques, as described above to determine the position of an individual or object.

For signal quality and accurate estimation, besides the correct calibration of the sender the advertising interval and power level are of utmost importance. The advertising interval, can be set to have a interval of 100 ms up to some seconds of payload data advertisement. The power level, which defines the signal strength and therefore has impact on the transmission range can have a maximum transmission power is 2,5mW, which should cover approximately 10 m by air.

The higher the power level and the lower the advertising interval makes the power consumption of the iBeacon go higher, but it also has great impact on the accuracy of the localization [24].

2.7 Message Queuing Telemetry Transport (MQTT)

One of the protocols considered for the transport of messages to the server in the project was the Message Queuing Telemetry Transport (MQTT).

MQTT is a lightweight, publish-subscribe network protocol that transports messages between devices, using the TCP/IP protocol [26].

The MQTT protocol specifies two types of network entities: a message broker and a set of clients. An MQTT broker is a server that receives all messages from clients and forwards them to the appropriate destination clients. Any device (from a microprocessor to a full-fledged server) that runs a MQTT library and connects to a MQTT broker over a network is considered a MQTT client [26].

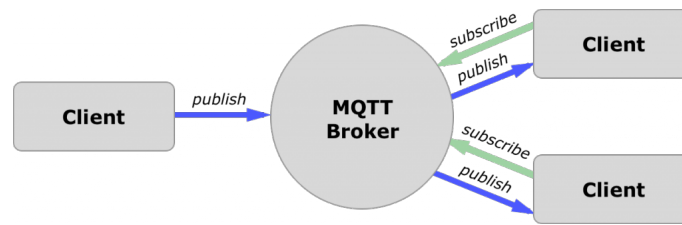


Figure 2.7: MQTT protocol illustration [26].

Information is organized in a hierarchy of topics. When a publisher has new data to distribute, it sends a control message to the connected broker including the data. The information is then distributed to any clients who have subscribed to that topic by the broker. The publisher does not need to have any information about the number or location of subscribers, and subscribers do not need to be configured with any information about the publishers.

One of the main advantages of using MQTT protocol to transport messages instead of other alternatives is the ability to easily scale from few devices to thousands, which is one of the project's requisites.

To comply with information privacy and security issues, it's important to state that MQTT eliminates vulnerable and insecure client connections and has a reduced network strain without compromising the security. With MQTT broker architecture, the devices and application becomes decoupled and more secure. MQTT uses Transport Layer Security (TLS) encryption with user name, password protected connections and the clients are unaware of each other's IP address.

2.8 Privacy and security issues

A common concerning challenge with indoor localization systems are the privacy protection and security matters. When designing a localization system or technology it is required to consider not

only the technical aspects of the project but also the ways the data provided might compromise the privacy of a person or how easy it is for someone of malicious intent to compromise the system's integrity.

In order to understand these issues, in this section the terms "privacy" and "security" will be presented alongside a description of their main concerns.

Privacy can be defined as “the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others.” and “the selective control of access to the self” [27].

The major concerns of location privacy in terms of location-related information processing are: collection, retention, use and disclosure of location-related information.

- Collection of location information refers to the ways of determining a target device location. This could be internally and independently by the device itself or externally by other devices it communicates with.
- Retention of location information refers to the location of where the localization data is stored. In the same way concerns in how much is stored and for how long, in other words, the quantity and persistence of location information and other characteristics of the stored information itself.
- Use of location information refers to the ways this information can be used to provide some kind of service, being it, for example, real-time routing assistance, destination guides, wireless advertising, finders for people and objects or for managing a company's tasks.
- Disclosure of location information concerns with how the data is shared with a third party user or company, being it intentional or in an unauthorized manner.

Security, in the other hand, is related with the use of privacy and in the characteristics of the localization system. A secure localization system protects privacy by avoiding either collection, retention, disclosure and the undesirable use of the localization information. For instance, an employee monitoring system may have privacy concerns in its many dimensions, but may be a secure system for disclosure just by not allowing any leaks of the information [11].

The Figure 2.8 below presents the three main pillars of information security and their description.

THREE PILLARS OF INFORMATION SECURITY

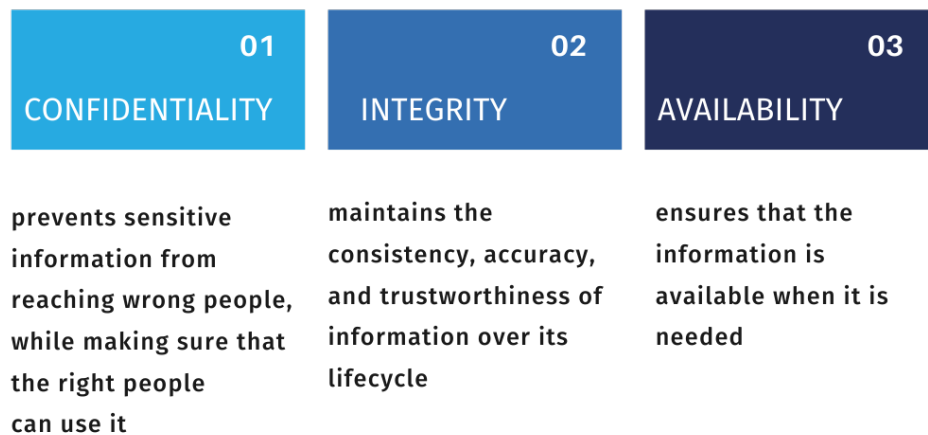


Figure 2.8: Thee pillars of information security [28].

These are very often referred to as the C-I-A triad, and they all have to be addressed in order to achieve a satisfactory level of information security.

Regulation and control of someone's localization information may come from several sources. Many governments are now considering new privacy laws covering location information even in indoor spaces, and courts are extending existing legislation into this new areas of technology [11][27].

Chapter 3

System Computational Architecture

In the realization of the project, one of the fundamental parts was preparation and planning. This chapter aims to present the system's concept and what is intended to develop. Starting with an overview, the location where the system will be implemented is presented alongside the general idea of how the system should proceed. Afterwards the requirements for the system are defined, assuring that the project meets the company's needs and has all the features that they expect from it. An entire section will also be devoted to presenting the planned system architecture and how its various sub-services will interact with each other. At the end, the database structure is shown and each relationship is explained.

3.1 Overview

The indoor location system will be placed in an assembly warehouse of the company RACELAND S.A. In this warehouse, 6 main zones were identified:

- Recreation place: resting place for RACELAND employees.
- Paint section: paint treatment, drying and other vehicle finishes.
- Production: Place where vehicle parts are assembled and produced.
- Tachograph Center: Vehicle testing and testing site.
- Secretary and Administration: General service area, divided between the floor below for the secretary and commercial area and the second and third floor for the engineering and administration department.
- Main Corridor: a large hallway in the middle that allows access to the other 5 main zones.

The intention is to make a scalable and adaptable system in which only the relative distance between each access point and beacon are known.

Each employee will hold a tag card that will periodically advertise data as a beacon, at a certain power and frequency.

Each gateway will capture the data advertised and send it to the server with information on the time they were issued, it's identification and the strength of the signals retrieved.

The server will then take into account all the measurements and will try to estimate the positions of each beacon. The estimations are stored in a database so that they can later be compared with the tasks being carried out.

A simplified floor plan, in Figure 3.1 below, is used to depict the system's concept.

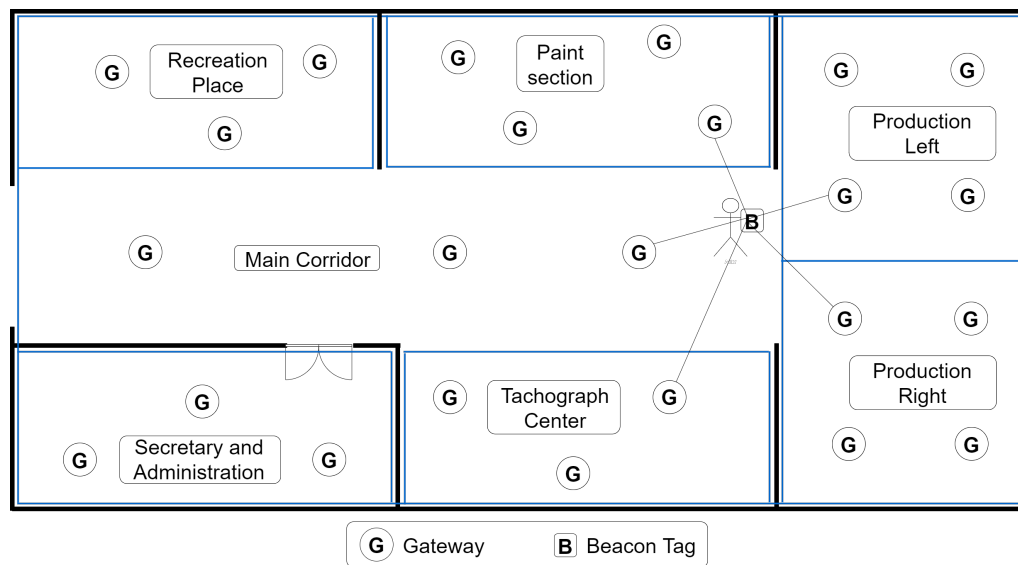


Figure 3.1: Simple warehouse floor plan and basic indoor localization concept.

3.2 Features required

In order to understand the specifications requested by the company in technical and functional terms, a list of feature requirements were compiled in a list. After obtaining this list, these features can be decomposed into performance and functional requirements.

The list of requirements presented in Table 3.1 is divided into three fields. The first field is the feature code, which serves as a label of that feature so that it can be identified and referenced. Besides the feature code, there is the description of the feature, which in a few words explains what is required by the company. A priority scale is then assigned to each feature, this is understood as the importance that the company attaches to the project, ranging from low to high.

Feature Code	Description	Priority
F1	The system must be able to be integrated into PHC billing	High
F2	The system must be scalable for any number of beacons and gateways	High
F3	The history of all positions must be stored in the database	High
F4	Must be able to assign a task to the employee or to a defined object depending on the position	High
F5	Must have a GUI that allows you to view task times and sequences	Average
F6	Sending messages from Gateways must be done using MQTT	Low
F7	The software must have a modular architecture and be configurable	High
F8	Beacon's protocol should be iBeacon	Average

Table 3.1: RACELAND S.A. requested features.

Each feature assigned with "High" priority is related with the overall objectives for the project. For example, the first four features (F1, F2, F3 and F4) are related with the company's necessity to automate and evaluate the manufacturing tasks when it comes to billing process. Feature F7 is also regarded as "High" priority to facilitate the use of the software and future upgradability.

The only feature assigned as "Low" priority is the feature F6. Consulting the manufacturer's datasheet for the gateways, it is possible to configure the IoT protocol to either HTTP REST or MQTT protocols. Both of them are accepted by the company and accomplish the same goals, the choice is up to what is considered the most suitable for the project.

3.2.1 Requirements

As previously mentioned, the feature list from Table 3.1 represents the broad characteristics required by the company for the project. In order to translate these characteristics in engineering terms, they need to be decomposed in functional and performance requirements. This section aims to present how those features were interpreted, beginning with a list for the functional requirements and then for the performance ones.

3.2.1.1 Functional Requirements

Functional requirements refer to the tasks that the system must perform, in a generic and high level manner. It does not include specific implementations, or detailed definitions of how to perform the task, or how to develop the system.

The list of functional requirements, presented on the following Table 3.2, is organized in 4 fields. The first field, similar to the feature code is used to label or identify the corresponding functional requirement. The following field is used to illustrate what features are associated with that specific requirement, for example, the first requirement RF1 was derived from features F1

and F3. The description field is used to briefly and concisely describe the functional requirement. Similar to the feature list, each requirement is categorized from low to high in priority. The intention with the last field is to understand how the time and effort should be focused when developing the system.

Code of the Requirement	Features Associated	Description	Priority
RF1	F1, F3, F6	Gateways should be able to communicate the information with the RACELAND server	High
RF2	F2, F4	Gateways must be strategically positioned with distances between them less than 10m	High
RF3	F2, F7	Gateways positions must be measured with coordinates relative to some fixed points	Average
RF4	F3	Gateways values must be parsed and placed in an MS SQL Server database	High
RF5	F1, F3	There should be a class distinction between objects and employees	Low
RF6	F3	Implementation of an algorithm capable of processing RSSI values of tags in coordinates	High
RF7	F5	Implement a BI dashboard to evaluate the results in the database	Average
RF9	F8	Each of the tags must be configured to use the iBeacon protocol	Average

Table 3.2: Functional requirements.

In order to integrate the system with the company's existing billing and ERP (enterprise resource planning) solution, namely PHC CS software, RF1 and RF4 are regarded as high priority when developing it. The main processing should be made in the company's server and each data must be then stored in their database for ease of access. The other requirements are related to the flexibility and usability of the system and will be further addressed in chapter 4.

The only functional requirement classified as low priority was RF5. RF5 states that there should be a distinction between a person and an object. The company RACELAND clearly mentioned that the main objective for the project was to have a functional and high accuracy system that could precisely detect the beacon tags to collect data of the company's activities. Therefore, the effort should be placed in the development of the estimator. Furthermore, since the idea was to develop a modular architecture for the software, this is seen as an extra feature that shouldn't be hard to add.

3.2.1.2 Performance requirements

Performance requirements typically comprise a set of criteria which stipulate how things should perform or the standards that they must achieve in a specific set of circumstances. These requirements are quantitative and low level specifications of how the system should be implemented and run.

In a similar manner to the functional requirements, Table 3.3 below is divided in the same four columns: code of the requirement, features associated, description and each is assigned in a priority scale.

Code of the Requirement	Features Associated	Description	Priority
RP1	F3	Position accuracy should have a maximum error of 2 meters	High
RP2	F6	The frequency of sending data from gateways should be every 30 seconds	Low
RP3	F2	Data processing should be done asynchronously	Low
RP4	F1	The system should run uninterruptedly as a service on the server	High
RP5	F5	The data processing should handle sets of 40 to 50 multiple payloads without any loss of information	High

Table 3.3: Performance requirements.

The company RACELAND made clear that their intentions for the project was to have an high accuracy system that could uninterruptedly handle lots of information from the gateways. For that purpose RP1, RP4 and RP5 are assigned with "High" priority and that's where the development should mainly focus on.

RP3 states that the data processing should be done asynchronously, this is classified as "Low" priority since a threaded option is also accepted. The form of processing will be chosen depending on the how the system will perform on practice.

The frequency of payload data, RP2, is also classified as "Low" priority since it's a configurable parameter and will be chosen depending on the results.

3.3 System's Computational Architecture

This section aims to present the proposed software architecture for how the information should flow between each subsystem and how the processing is going to be done. Having in mind the requisites from section 3.1, the architecture chosen is known as "*Microservice Architecture*". The main idea behind a microservice architecture is that applications are simpler to develop and upgrade when broken down into smaller pieces that work seamlessly together. When using microservices, each subsystem functionality is isolated into multiple independent modules that are individually responsible for performing standalone tasks [29].

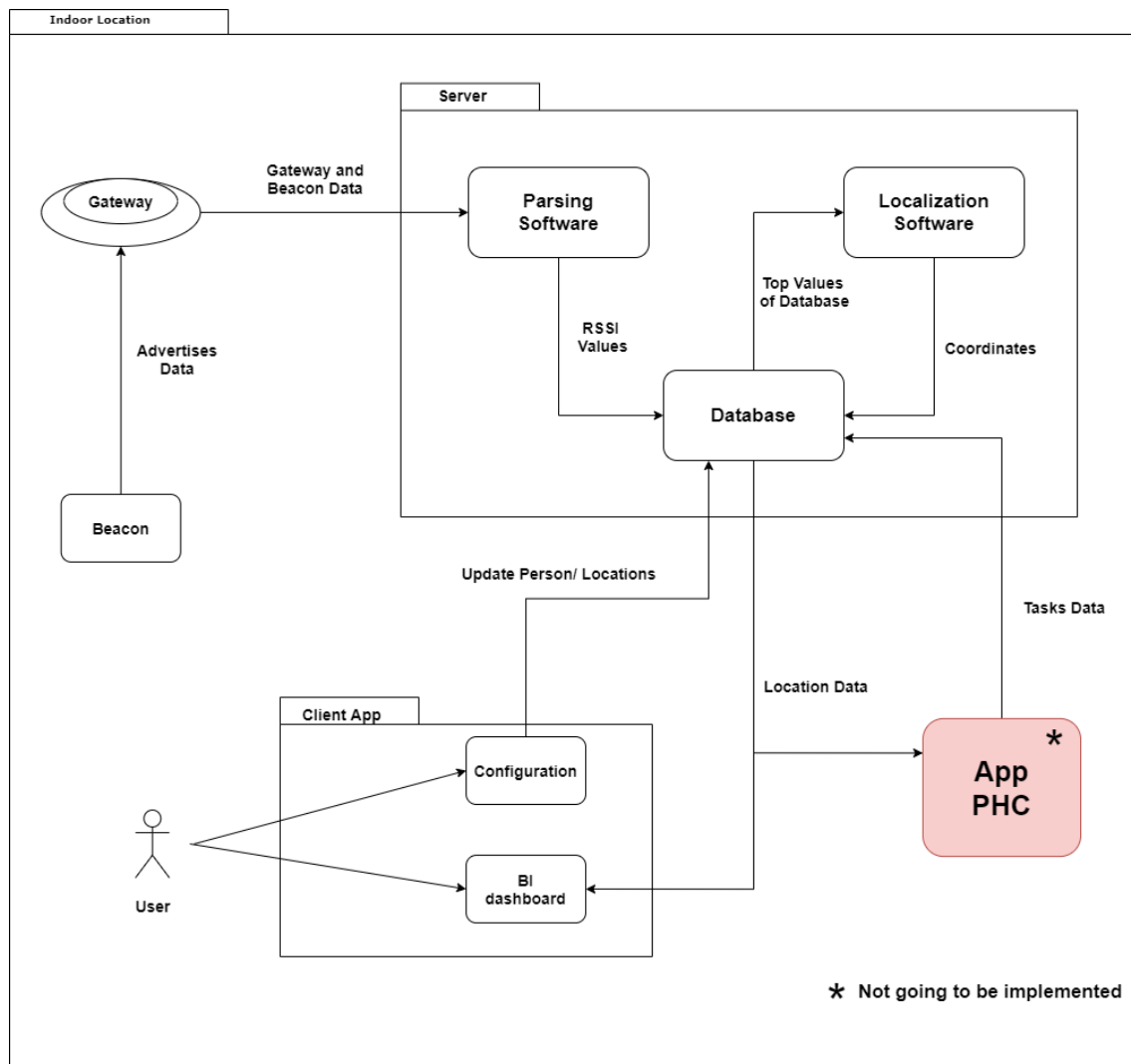


Figure 3.2: System's subsystems and information flow.

As Figure 3.2 illustrates, the system has three major subsystems: the server-side, the client-side and the PHC application which will not be implemented. In the server-side the system has three microservices: the database, the parsing and the localization software. Each beacon will advertise data that will be interpreted and the signal strength will be measured by the gateways.

The "Parsing Software" task is to parse the payload and the signal strength measurements that each gateways sends and stores them in the server's MS SQL database. The "Localization software" will process a batch of the most recent measurements from the database at a certain interval and will output and store the estimated coordinates into another table of the database.

The company PHC CS's application will be in charge of introducing to the database the manufacturing tasks that are in progress, so that the position of each beacon could be assigned to a certain task.

Finally, with the "Client App", the user will be able to see and configure the results present in the BI (Business intelligence) dashboard. This dashboard will present to the user an overview of the warehouse that will show the location of each beacon device and interactive graphs and key performance indicators (KPI) for further evaluation.

3.3.1 Database Structure

As mentioned previously, each module will perform it's task isolated and independently from each other, making the database the most critical part to ensure interoperability between them. This part of the section is used to present and explain the structure of the database by showing the classes, their attributes, and their relationships. To this effect the following class diagram was made:

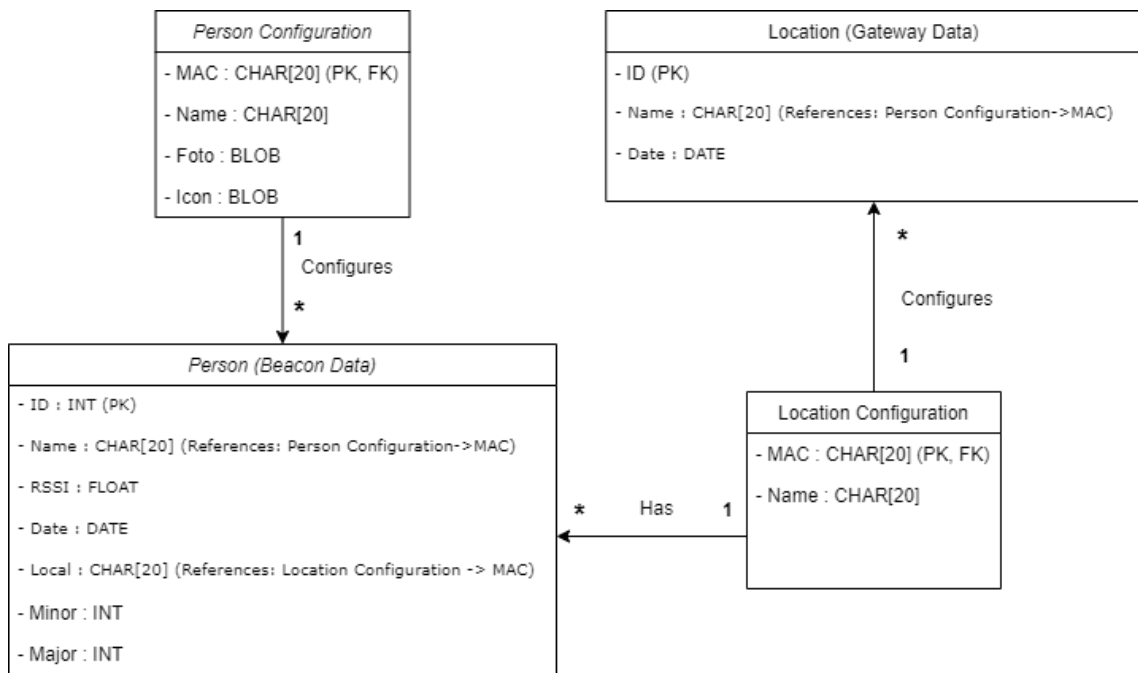


Figure 3.3: Class diagram for parsing software's database.

Figure 3.3 represents a class diagram with 4 main tables in the database, two of them for the data parsed and two of them for configuration purposes. The main idea is that the "Parsing software" will process the information and will organize it into two tables: the beacon or person related information and the gateway or the location where that information was retrieved. The

"Configuration" microservice will be responsible for updating the values for two the remaining tables, changing how each device is assigned to each person or location. The "Localization software" will then be able to produce a query and get a batch of the most recent records and, after the processing is done, output the coordinates, the person assigned and the timestamp to a new table to be utilized for application-level strategies.

Chapter 4

System Development

This chapter will present the procedures taken when the system was being developed. From start to finish, as much as possible, there will be an attempt to follow the steps and reasoning as each feature was being developed.

Firstly, the development of the parsing microservice will be explained with the appropriate conclusions. Afterwards, filtering techniques were analyzed to improve the measurements that were given. Finally, an explanation of the positioning microservice will be provided, which will be responsible for estimating the Cartesian coordinates of the target using various techniques, some of which have already been presented.

4.1 Sensors

To start analysing and graphing the raw signal strength values that the gateways output, it was necessary to develop a subsystem to receive the published MQTT message packets and separate each piece to correctly identify the information given. That subsystem is called the "parsing microservice". The sequence diagram in Figure 4.1 illustrates the processing sequence of the parsing microservice as implemented.

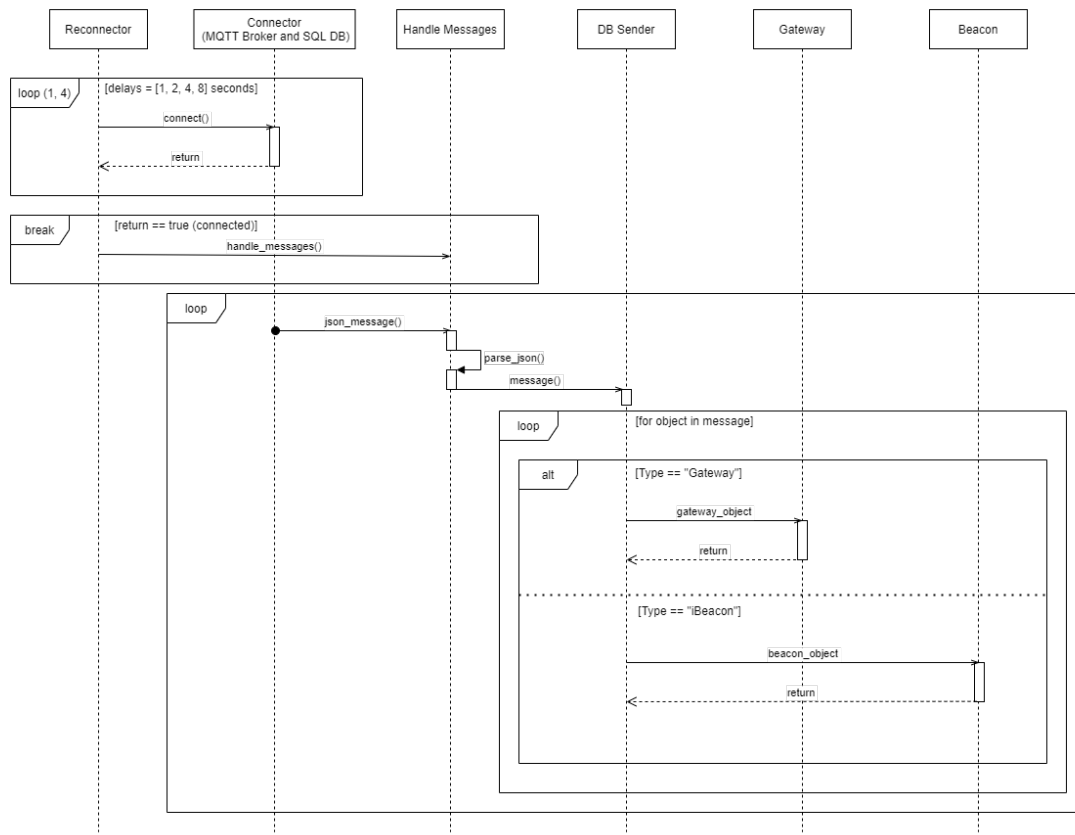


Figure 4.1: Parsing Sequence Diagram.

As the figure depicts, the microservice processing is asynchronous and follows an object oriented pattern with 6 main classes. Each of these classes can be explained as follows:

- **Reconnector**

The "**Reconnector**" class is responsible for initiating and maintaining connections with the SQL database and MQTT broker. It initially sends a request to connect to the "Connector" class which tries to make the connections and returns if they were successful.

In case the connection is not successful, it waits one second and tries again. If it still doesn't succeed, it waits twice as long as the previous attempt and tries again. This process is repeated several times until the connection is succeeded.

In case the connection is successful, the program is able to continue and as such sends a request to start handling the messages.

- **Connector**

The "**Connector**" class has two main focuses. The first one is assuring the connections to the SQL database and MQTT broker, given a configuration string. The second part is subscribing to the right topics and sending the messages once they arrive to the "Handle Messages" class.

- **Handle Messages**

"**Handle Messages**" class starts an infinite loop and whenever it receives one or more messages from the gateways in json binary format, it decodes it and passes it to an array of string objects and sends to the "DB Sender" class.

- **DB Sender**

The "**DB Sender**" class receives the message as an array of string objects. This array contains both the information regarding the gateway that published the message and the information and packets regarding the beacons. Afterwards, it prepares objects from gateways or beacons to be sent to the database.

In case it is a Gateway object, it is sent to the "**Gateway**" class, which will be in charge of interpreting and organizing the values and waiting for the return to be sent to the "Location" table of the database, as specified in 3.3.

In case it is a beacon object, it is sent to the "**Beacon**" class. For each Beacon, an object is created where the values are interpreted and organized to also be sent to the database for the "Person" table (Figure 3.3).

4.1.1 Parsing Results and Conclusions

With the parsing service in place, sufficient conditions were established to gather raw RSSI data and observe behaviors. An experiment was carried out to this goal. To obtain proximity measurements, one of the gateways was fixed placed, the distance was measured using a tape measure and the data recorded using only one beacon, as the Figure 4.2 below illustrates.



Figure 4.2: Proximity experiment illustration.

A range of distances between 0.6 and 8 meters was determined. The results were taken meter by meter, with the exception of 0.6 meters as a measurement below the reference (1 meter) was needed. At each distance, 15 measurements were taken and recorded in a spreadsheet to graph the values. The graphs below, Figure 4.3, show the RSSI values in dBm of the 15 measurements taken for each distance.

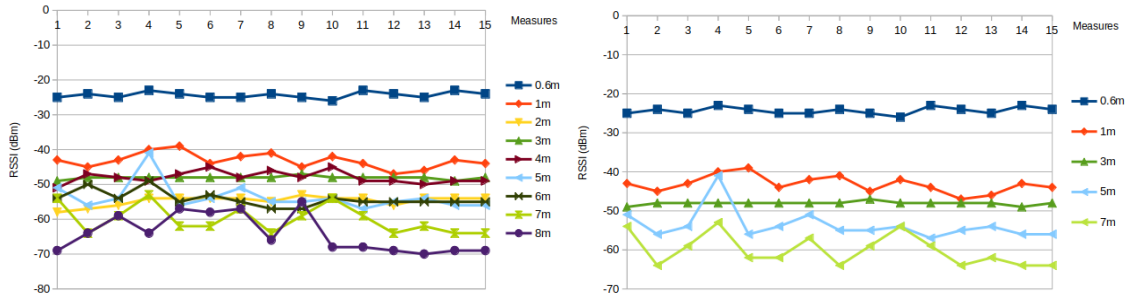


Figure 4.3: Raw RSSI Value Results, meter by meter (Left) and 2 by 2 meters (Right).

Given the non-linearity of the signals due to possible reflections or disturbances outside, for close distances the values received are quite similar, even overlapping. A second graph with RSSI values every 2 meters was placed to the right to better observe the behavior of the signal strength as the beacon moves away from the gateway. From the results presented above for the raw rssi values, it is possible to confirm that there is some noise that creates the fluctuations of the signal. This noise can arise for several reasons, such as signal reflections, presence of obstacles and dust or simply poor sensor calibration.

It is also observed that, in general, the greater the distance, the greater the fluctuations of the signal. This is because the smaller the signal strength, the greater the influence of the noise.

Below, a table with the average measured RSSI values and standard deviations for each distance is also presented.

Measured Distance (m)	0.6	1	2	3	4	5	6	7	8
Average RSSI (dBm)	-24.33	-43.20	-54.73	-48.07	-48.00	-53.67	-54.20	-60.07	-64.13
Standard Deviation (σ)	0.899	2.178	1.387	0.457	1.732	3.904	2.178	4.008	5.423

Table 4.1: Average RSSI values for each distance measured.

Taking a look at the results, it is possible to infer that for close distances the raw average values of the signal strength poorly reflects the corresponding distances. To reinforce this statement, the average relative strength value at 5 meters (-53.67 (dBm)) and at 6 meters (-54.20 (dBm)) have a difference of less than 1 dbm and the results at 3 and 4 meters are almost identical (-48.07 (dBm))

and -48.00 (dBm) respectively). The result at 2 meters distance was considered a defective value, probably due to the conditions under which the experiment was carried out or a sensor defect. As for the standard deviation, it is clear that as the distance to the gateway increases, the higher the standard deviation value, increasing the signal fluctuations, as noted above.

Converting from RSSI to meters

As explained in detail in the section 2.4, it is possible to describe the relationship between signal strength (RSSI) and distance using the Log-Distance pathloss model:

$$RSSI = -10 \cdot n \cdot \log_{10} \left(\frac{d}{d_0} \right) + A_0 \quad (4.1)$$

In this model, d describes the distance between the beacon and the gateway, n the signal propagation exponent and A_0 a referenced RSSI value at d_0 .

By rearranging the equation, it is possible to estimate the distance d as:

$$d = 10^{\frac{(Measured_{RSSI} - A_0)}{(-10 \cdot n)}} \quad (4.2)$$

To obtain the distance in meters in relation to the RSSI value in dBm (decibels to milliwat), it is necessary to set the values of the variables n and A_0 . The choice was made through successive educated guesses until the curve with the averages of the estimates approached the values that ideally should have for each distance. The values found for those variables were $n = 4$ and $A_0 = -27$ dBm.

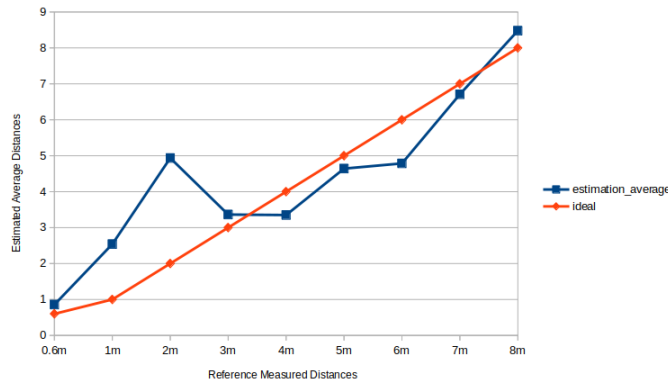


Figure 4.4: Ideal value and estimation average fitting curve for $n = 4$ and $A_0 = -27$ dBm.

Using the above values, the table below serves to show how the mean estimation values compare to the distances at which they were measured.

Measured Distance (m)	0.6	1	2	3	4	5	6	7	8
Estimated Average (m)	0.86	2.54	4.94	3.36	3.35	4.64	4.79	6.71	8.48

Table 4.2: Average Estimated distance values for each distance measured.

While adjusting the values, one of the main focus was to closely match the values for further distances, since it's where the system will mainly operate and the fluctuations are higher, thus increasing the accuracy as a whole.

In the same way as with the RSSI results, Figure 4.5 below shows two graphs with the values of the estimated distances for each measured distance.

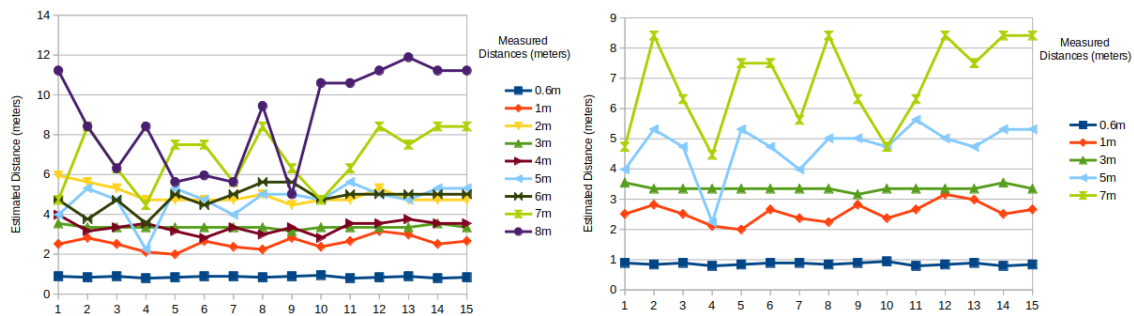


Figure 4.5: Estimated Distance Value Results, meter by meter (Left) and 2 by 2 meters (Right).

Having the values converted it is possible to see the effect that fluctuations have on the distance estimates. During the tests, the measurements registered a variation with a maximum amplitude of about 8 meters with errors of more than 4 meters in relation to the reference. These errors are more noticeable at large distances where fluctuation is greatest. In the graph on the right of Figure 4.5, the fourth estimation at 5 meters inclusive was smaller than at 3 meters. At short distances these variations are no longer so noticeable, however the system will rarely work at these distances, making it necessary to implement some type of signal processing.

4.2 Filtering

As previously shown, the raw relative strength results showed large fluctuations due to signal noise, significantly affecting stability and efficiency. One method to mitigate this problem is to apply a discrete filter that reduces these variations.

The two discrete filtering methods that were likely to be used in the system will be provided in this subsection, along with a practical explanation.

Following that, the two methods will be compared using a test signal, and it will be determined which method is best suited for this case based on the results.

Having chosen the filter method, it will be applied to the test results performed in the parsing and conclude how they've improved.

4.2.1 Filter Techniques

This subsection presents 2 different filtering components implemented: Discrete Kalman Filter and Particle Filter.

Both filters are capable of minimizing the fluctuations, are widely used in indoor localization and navigation and use state space equations making them the two preferred candidates to use for this project.

Kalman Filter

The Kalman filter is a state estimator that uses noisy observations to estimate an unobserved variable. It is a recursive algorithm since it considers the history of measurements. The key goal in this scenario is to determine the actual RSSI from a set of measurements. The regular or discrete Kalman filtering module attempts to estimate RSSI values by portraying the RSSI time evolution as a mixture of signal noise (measurement noise) and maximum signal evolving (process noise). A linear stochastic equation models the RSSI evolution, with signal/process noise assumed to be independent of each other, white, and with normal probability distribution (standard deviation $\frac{R}{Q}$) [30].

The filter works by minimizing process noise (w) through a two phase algorithm. First, a predictor performs the estimation for the next RSSI (equations 4.3 and 4.4). Then a corrector improves the RSSI estimation by exploiting the current RSSI measurement (equations 4.5, 4.6 and 4.7).

An iteration (k) of the filtering module is done with the following sequence of equations:

$$\hat{\hat{x}}_k = A \cdot \hat{x}_{k-1} + w_{k-1} \quad (4.3)$$

$$\bar{P}_k = A \cdot P_{k-1} \cdot A^T + Q \quad (4.4)$$

$$K_k = \hat{\hat{P}}_k \cdot H^T \cdot (H \cdot \hat{\hat{P}}_k \cdot H^T + R)^{-1} \quad (4.5)$$

$$\hat{x}_k = \hat{\hat{x}}_k + K_k \cdot (z_k - H \cdot \hat{\hat{x}}_k) \quad (4.6)$$

$$P_k = (1 - K_k \cdot H) \cdot \hat{\hat{P}}_k \quad (4.7)$$

where P_k is the covariance matrix of the state estimate error at step k , with initial value Q .

In this project scenario x and z are RSSI values, the state coincides with the output (A is a 1×1 identity matrix), and the estimation of the next state estimate is equal to the current state (H is a 1×1 identity matrix). R and Q are the measurement noise and the process noise respectively and their values are fixed and determined by successive tests until a good trade-off between RSSI fluctuation mitigation and filtered-to-actual RSSI delay is obtained. The best values found were $Q = 0.1$ and $R = 2$.

Particle Filter

Like the Kalman filter, the Particle filtering module tries to estimate RSSI by minimizing measurement and process noise, but without imposing a linear equation modeling and, more important in the deployment scenario, without imposing a normal distribution for signal noise. The main idea at its basis is to have an algorithm that computes, at each step, several possible filtered RSSI values for each measured RSSI. Then, the filter associates each candidate value with a weight and chooses the most promising values among them when a new measured RSSI is available. Finally it perturbs candidate values according to the rules shown below, thus obtaining a new filtered RSSI (the average value of the most promising candidates) [30].

To better and practically understand how the Particle Filter works, below a rapid example of the algorithm is shown with 10 particles, which represents 10 possible filtered values (Figure 4.6):

1. Starting step - there are 10 possible filtered RSSI values (yellow circles in figure 4.6), all with same weights;
2. Importance weight step - by exploiting state estimate probability (the black curve in figure 4.6) obtained from RSSI measurement, the algorithm assigns a weight at each filtered RSSI value (Blue circles);
3. Re-sampling step - heavy RSSI are spread in different RSSI values, all with the same weight (yellow circles), while light RSSI values are discarded;
4. Sampling/Prediction step - filtered RSSI are randomly perturbed (yellow circles).

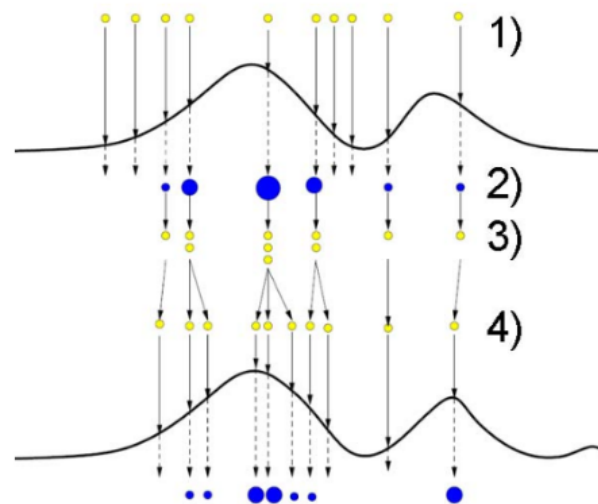


Figure 4.6: Example of one iteration of the Particle Filter algorithm [30].

The number of particles strongly influences the Particle Filter performance. In general, the greater the particle number, the better the filtered RSSI follow the actual RSSI sequence. However, differently from the Kalman Filter, the Particle Filter has a non-negligible computational

load, which deeply depends on the particle number. Due to performance issues, the best trade-off between performance and results was with 5 particles.

Method Comparison

To give a comparison of the proposed methods, the filters were applied to a measured test signal. To give consistency to the results, the same values of $Q = 0.1$ and $R = 2$ were used, and A and H are identity matrices.

Figure 4.7 shows a comparison of the RSSI fluctuations between the actual and the filtered signals, over almost 40 seconds.

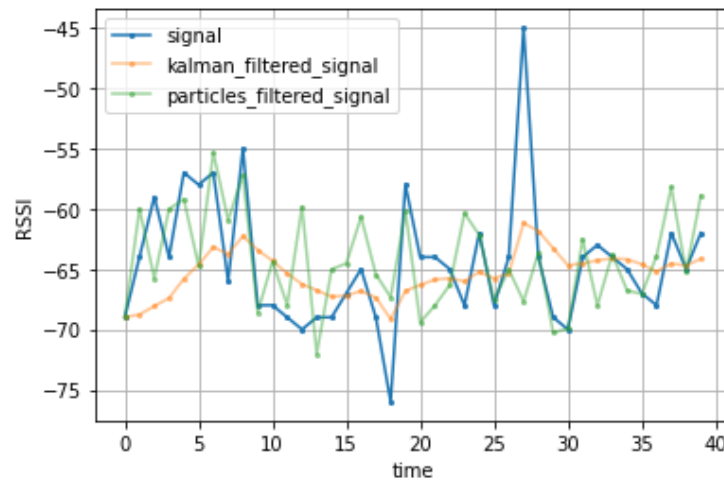


Figure 4.7: Actual and filtered RSSI (dBm) over time (seconds).

By merely observing the results, it is possible to conclude that both methods were able to minimize RSS fluctuations. However, it is verified that the signal filtered by the Kalman Filter method has a smoother behavior than the signal filtered by the Particle Filter. As for RSS fluctuations, the signal filtered by the Kalman filter seems to have lower amplitude in the fluctuations when compared to the Particle, for this case. Having these results in mind, the choice was clear that the Kalman Filter was the best candidate when processing the RSS signals in this project.

4.2.2 Test Results and Conclusions

Using the Kalman Filter as the preferred method, the filter was applied to the results of the experiment conducted when developing the Parsing service, subsection 4.1.1, to observe how the RSS signals have changed.

In the same way as was done before, in Figure 4.8, two graphs are provided showing the RSSI values in dBm of the 15 measurements taken for each distance.

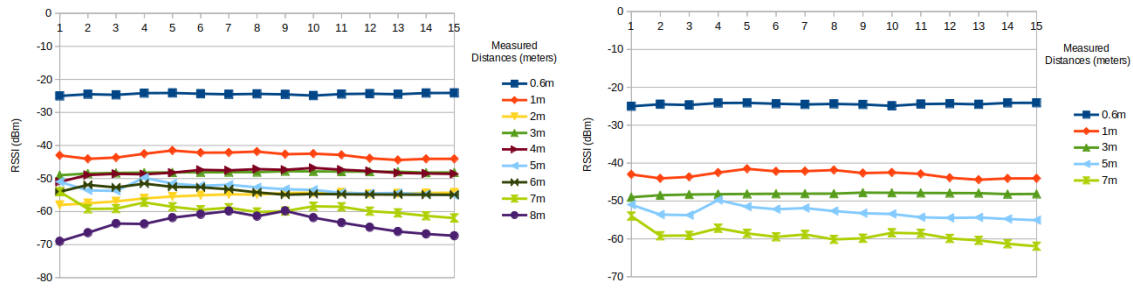


Figure 4.8: Filtered RSSI Value Results, meter by meter (Left) and 2 by 2 meters (Right).

Compared to previous results, the signals have a smoother behavior as expected. Although some fluctuations can still be noticed, they are much more minimized. In addition, observing the graph on the right, none of the RSS measures of the signals overlap. In this way it is already possible to assign a defined range of RSSI values for each distance at which it was measured, greatly benefiting the estimation of proximity distances.

In Figure 4.9, two graphs are provided to show the results of the proximity distance estimates with the filtered RSS values.

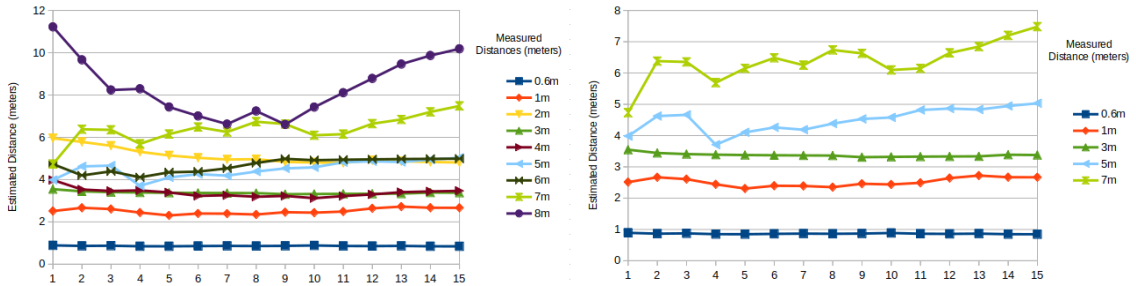


Figure 4.9: Estimated Distance Value Results, meter by meter (Left) and 2 by 2 meters (Right).

These two graphs clearly show how the application of a discrete signal method can greatly improve the efficiency and consistency of the distance estimation. Even tho there's still some noise at greater, the amplitude of fluctuations decreased by more than half compared to when the estimation was conducted on raw RSS values, recording in this example less than 4 meters of amplitude. The estimation error as whole has also decreased greatly, registering an average error of 0.969 meters.

4.3 Localization Estimator

Having the measurement results, the next step is to create an algorithm that can convert them in real time to positions in the plane. This subsection will explain the service behind the positioning system, decomposing each of its constituent parts.

As in this case efficiency was not a requirement, and the results will be presented every 30 seconds or more, the processing of this service was done synchronously, contrary to how parsing was implemented.

The sequence diagram below, figure 4.10, illustrates the processing sequence of the positioning microservice as it was implemented.

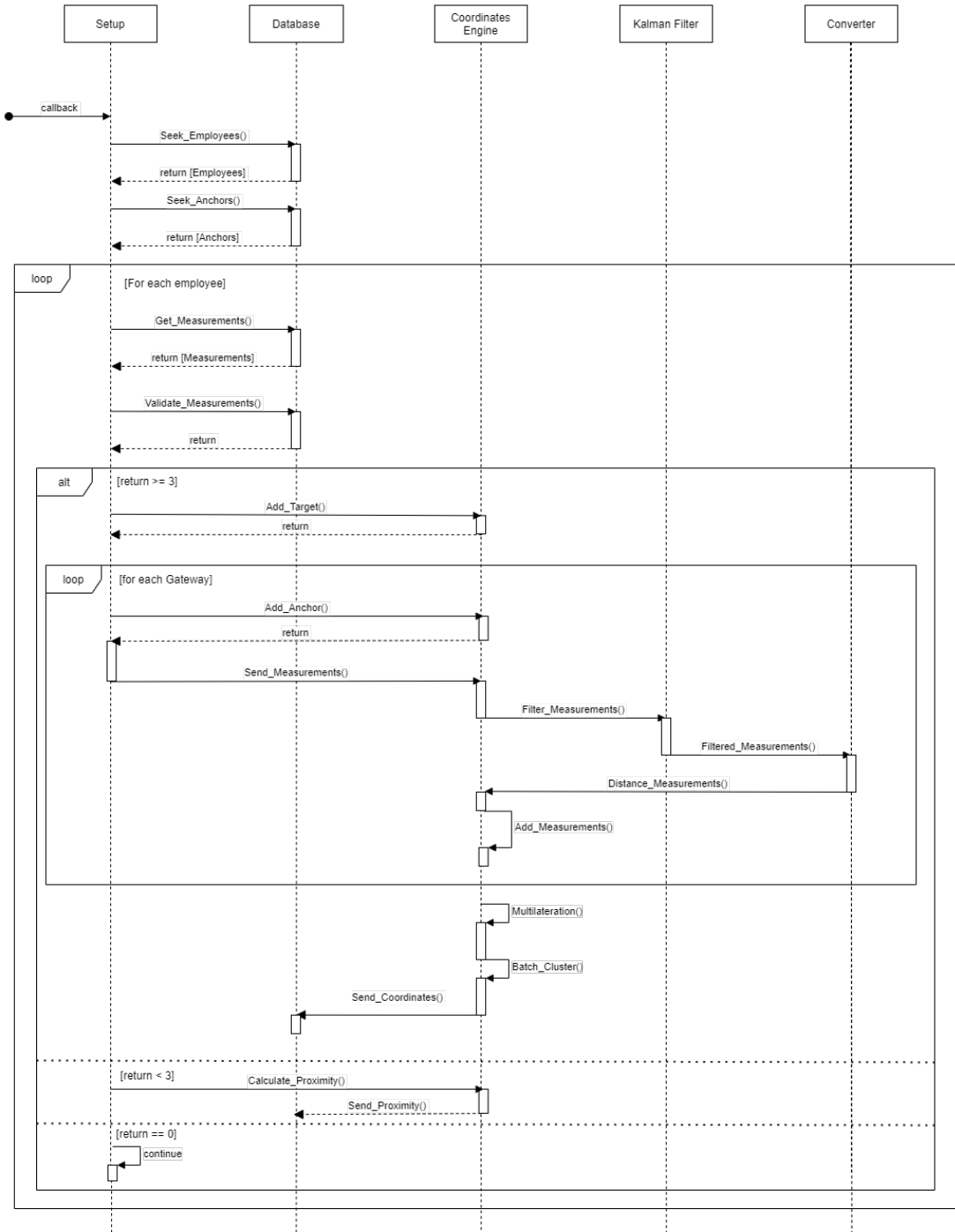


Figure 4.10: Positioning Sequence Diagram.

As in the case of parsing, this microservice follows an OOP (Object Oriented Programming) structure with five classes. For each class an explanation of its functions will be made.

- **Setup**

The "**Setup**" class, at the moment the callback is made, performs the first procedures and organizes the necessary data before sending it to the "Coordinate Engine".

Initially, it sends a query to the database looking for information on employees, or targets, and organizes the result in a hash table. This hash table is a data structure that can map keys to values, where in this case the keys are the MAC addresses beacon, which will be unique for each employee, and the values are the full names of each employee in string format. Thus, it is possible to identify the person to which the beacon was assigned and use the MAC as UID (Unique Identifier) to identify the target in the "Coordinate Engine".

Key	Value
Beacon's MAC Address (Char[10])	Employee's Full Name (String)

Table 4.3: Beacon's Hash Table Structure.

Right after that, it retrieves the stored information relative to the anchors, or gateways in this case, and organizes that into another hash table. In this case the UID will be the MAC Address of each G1 Gateway, and the values will be it's location name and it's 3D coordinates, stored in a doubly link list.

Key	Value
Gateway's MAC Address (Char[10])	[location name, X, Y, Z coordinates] (List)

Table 4.4: Anchor Gateway's Hash Table Structure.

Afterwards, the class starts looping and for each employee in the hash table it gets a batch or a set of the top most recent measurements stored in the database, given a certain time interval (for example: the most recent 30 measurements, in the last 30 seconds).

Once the measurements are retrieved, it evaluates how many measurements and distinct gateways are on that batch.

In case it has measurements of 3 or more distinct gateways, it has sufficient conditions to start multilateration and, therefore, it adds the target to the "Coordinate Engine" and sends the batch to be processed.

In case it has less than 3 anchors, proximity is calculated. The way the proximity is done is by comparing the values in the batch. The position of the beacon is assumed to be the position of the closest gateway. in case there are two gateways, the results of each are averaged and the distance result is compared, the one with the smallest distance being chosen as the

one representing the relative position of the beacon. In case there is only 1 gateway, this is automatically chosen as the reference position.

If there are no measurements, it is considered that it is absent from the location and the loop is continued for the next employee.

- **Database**

"**Database**" is the class that interacts with an instance of a Microsoft's SQL Server service. As explained in detail in the system architecture, section 3.3, this service will be responsible for maintaining the interoperability and consistency between the different subsystems. This class enables the execution of simple store, retrieve, update and delete data tasks in a relational database.

Depending on the query sent, it will execute the desired function and return the output as a table. To send the query a prepared statement is used. Prepared statement or parameterized statement is a feature used to execute the same or similar database statements repeatedly with high efficiency, greatly reducing the processing time and minimizing the bandwidth to the server by not sending the whole query.

- **Coordinate Engine**

The "**Coordinate Engine**" is the class that performs position estimation. After the targets and anchors involved have been aggregated and validated, it starts iterating through each employee in the hash table and checks the batch of raw RSS measurements sent by the gateways concerning that employee. This batch of measurements is sent to the "Kalman Filter" class to process the data and waits for a set of estimated distances in order to associate these measured distances to the gateway and target.

After iterating through all the gateways for that target, the multilateration process itself begins. The multilateration method chosen was the "circular positioning algorithm" based on the least squared error (LSE) propagation estimation. The basic idea of the circular positioning algorithm is to find the position (x, y) of the mobile target using an optimization procedure that minimizes the sum of the squared errors in the set of estimated distances.[31] If (x_i, y_i) is the position of the gateway i ($i = 1, 2, \dots, N$, where N is the number of gateways) and $\tilde{d}_i$ is the distance estimation to the gateway i , the LSE error is given by

$$\varepsilon = \sum_{i=1}^N \left(\sqrt{(x_i - x)^2 + (y_i - y)^2} - \tilde{d}_i \right)^2 \quad (4.8)$$

The error ε will be considered in the program as the cost function to be minimized.

The position (x,y) of the target can be then calculated iteratively using for example a stochastic gradient descent method, with learning rate $\alpha = 0.1$.

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}_{k+1} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}_k - \alpha \cdot \begin{bmatrix} \frac{\partial \varepsilon}{\partial x} \\ \frac{\partial \varepsilon}{\partial y} \end{bmatrix}_{x=\hat{x}_k, y=\hat{y}_k} \quad (4.9)$$

This method requires an initial location estimation. The method recommended by *Gau, Y., Chu, H. and Jan, R.* [32] was followed, in which the initial estimation is given as the sum of the gateway center points (x_i, y_i) multiplied by the inverse of the normalized distance (W_i) , as expressed in equations 4.10 and 4.11 below.

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \sum_{i=1}^N \left(W_i \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) \quad (4.10)$$

where W_i is the inverse of the normalized measured distance $\tilde{d}_i$ and is calculated as show in the equation 4.11 below.

$$W_i = \frac{\left(\left(\sum_{j=1}^N \tilde{d}_j \right) - \tilde{d}_i \right)}{(N-1) \cdot \left(\sum_{i=j}^N \tilde{d}_j \right)} \quad (4.11)$$

After iterating over all gateways and over all measured distances the result will be a batch of position estimates. A simple centroid finding technique is then used to obtain the final result of the position estimation. The centroid or center of mass of the group of L points is then given as

$$(x_f, y_f) = \left(\frac{x_1 + x_2 + \dots + x_L}{L}, \frac{y_1 + y_2 + \dots + y_L}{L} \right) \quad (4.12)$$

To illustrate how the centroid technique helps the results, Figure 4.11 below was prepared with a real example.

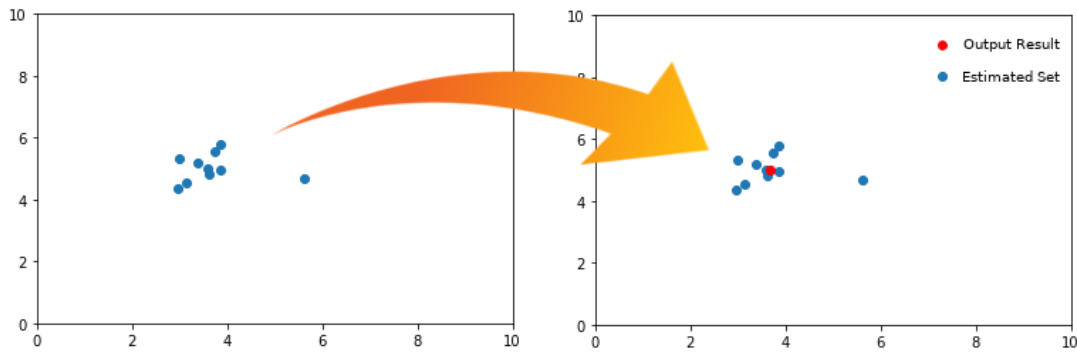


Figure 4.11: Clustering technique demonstration, Blue dots are the set of estimated distances, Red dot is the output result.

As the figure portrays, the centroid technique averages the results which in turn will help minimize the appearance of outliers due to disturbances and noise in the environment, enhancing the accuracy of the estimation.

Finally, the program cycle ends sending the final result to the database along with the timestamp in which it was executed.

- **Kalman Filter**

The "**Kalman Filter**" class, as its name implies, includes the algorithm implementation of the Kalman Filter method as specified in the 4.2 section, to iteratively filter a batch of RSS measurements and return the data processed.

- **Converter**

The "**Converter**" class is an utility class with implemented methods that can convert RSS signals in decibel miliwatt (dBm) to distances in meters.

The model used to translate signal strength to distances was the Log-Distance Pathloss model [20] as described previously in subsection 4.1.1.

4.3.1 Experiments and Results

After implementing the microservices on the server, enough conditions were met to test the system in a real environment and check the results.

A few meetings with the company were made with the intention of choosing the location and listing the material needed to proceed with the tests. It was established that the tests will only be carried out in one small area and that depending on whether the results are promising, it would be scaled up to the rest of the warehouse. As for the material, 5 G1 Gateways were arranged and each employee was assigned a C7 card tag with a QR code and it's name printed on the back.

To obtain reliable and coherent results, the site should have conditions close to ideal. The site should be a wide enough space, with no obstacles in the middle that could hinder the collection of signals and the height of the gateways should be the same to avoid measurement errors. Of the 6 main areas into which the warehouse is divided, the only area that had all these conditions was the main corridor, next to the tachograph area.

A sketch, Figure 4.12 of the location was made to delimit the zone and the layout of the gateways before being placed.

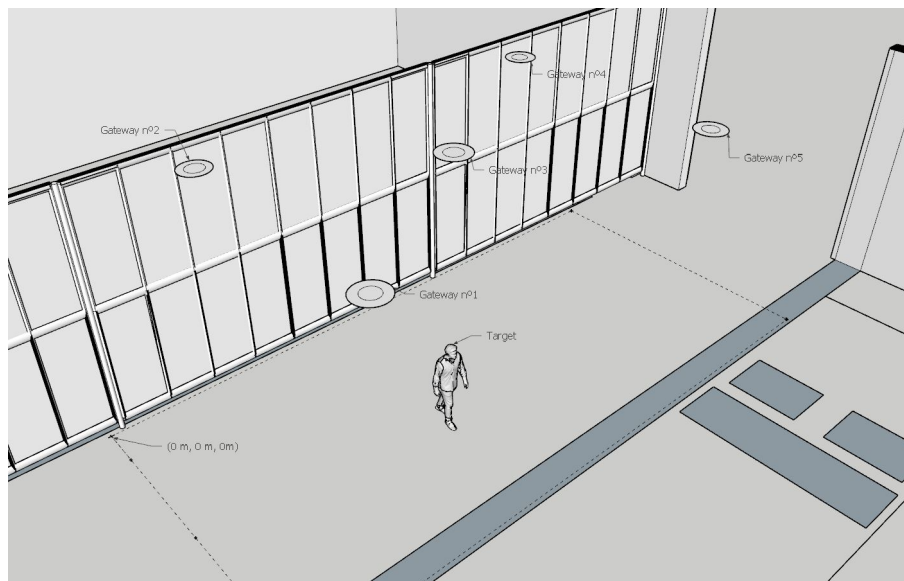


Figure 4.12: Overview Sketch of the testing zone as implemented.

The experiments were performed in a 12 meters in length by 7 meters in width area in the main corridor. To ensure minimal interference, the zone was cleared and every object was removed.

The distribution of gateways in the experiment zone followed a cross placement. This placement was considered to improve the system's accuracy in the symmetry zones, making it less sensitive to proximity estimation errors.

The gateways' center coordinates are presented in table 4.5 below.

Gateways	Gateway n°1	Gateway n°2	Gateway n°3	Gateway n°4	Gateway n°5
X coordinate	5.168	0.769	2.977	0.769	5.168
Y coordinate	2.072	2.072	5.388	9.144	9.144
Z coordinate	5.235	5.235	5.235	5.235	5.235

Table 4.5: Gateways' Center Position Coordinates.

With the help of a technician from the company, the 5 gateways were hung from the roof, all at the same height of 5.235 meters, and powered by PoE (Power over Ethernet) with network cables connected to a switch, ensuring constant power and the connection to the same subnet. Figure 4.13 shows the final result of the distribution of gateways in the main corridor of the company's warehouse.



Figure 4.13: Gateways' Placement.

During the experiment, the beacon card tags were placed on top of objects, like ladders and boxes, in fixed strategic positions and at a height of 1 meter, simulating a person standing with the tag held in his hand, as the Figure 4.14 below illustrates.



Figure 4.14: Demonstration of how the tests were conducted.

For this experiment, four beacons were placed at four sites throughout the test zone that were thought to best depict the system's diverse conditions. The four points chosen were: in the center at the position $(x = 3.38, y = 5.90)$, near the top corner at the position $(x = 4.90, y = 8.15)$, in the middle of the gateways at the position $(x = 2.25, y = 3.50)$ and outside but near the test zone at position $(x = 7.62, y = 2.17)$.

Figure 4.15 is used to present the results obtained after 15 consecutive position estimates for each of the 4 situations. A 30 seconds time cycle was used between each position estimate for each beacon.

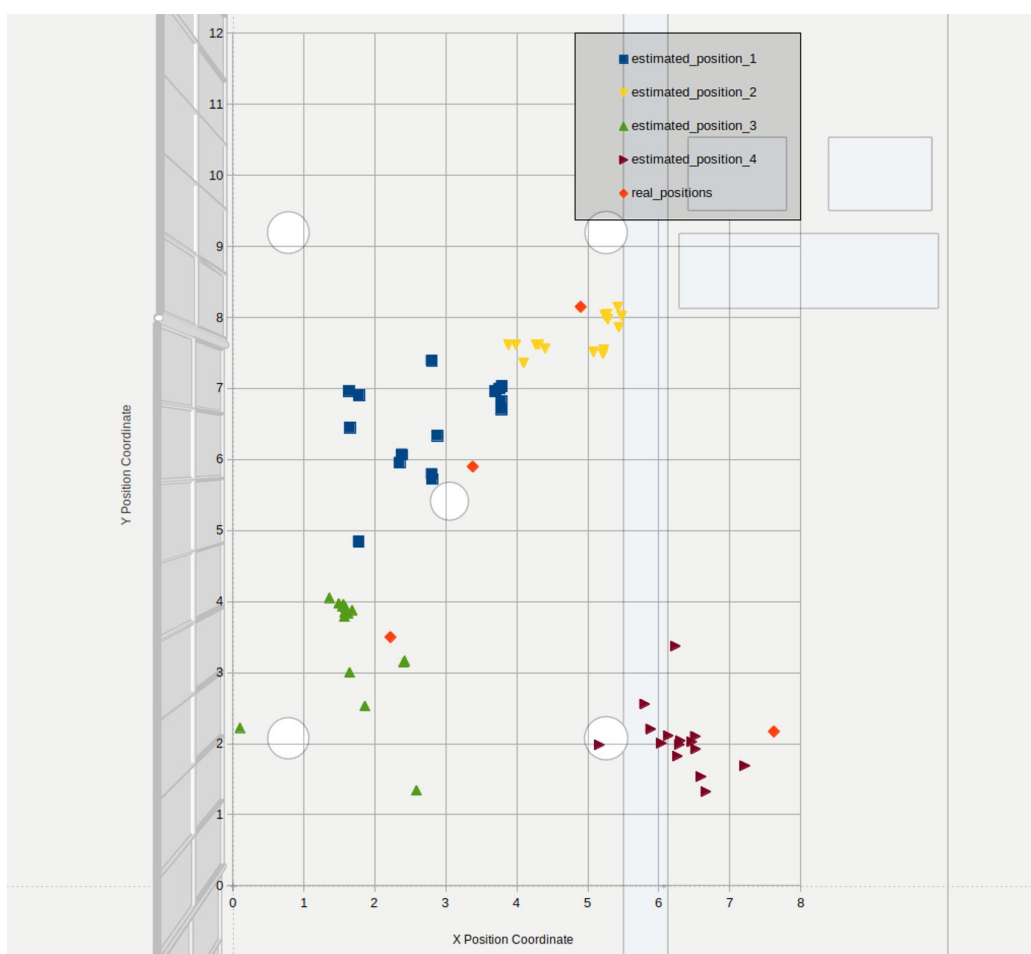


Figure 4.15: Position Estimate Results and their real positions for comparison.

To also show some metrics for evaluation in terms of error between the Euclidean distance of the estimates and the actual position, the following Table 4.6 is presented with the mean squared error (MSE), standard deviation, maximum and minimum error values for each situation.

Positions	Position 1 (3.38, 5.90)	Position 2 (4.90, 8.15)	Position 3 (2.22, 3.50)	Position 4 (7.62, 2.17)
Mean Squared Error (MSE)	1.759	0.569	1.233	2.237
Standard Deviation (σ)	0.496	0.252	1.752	0.426
Max error distance	2.039	1.150	2.474	2.464
Min error distance	0.588	0.362	0.388	0.630

Table 4.6: Mean Squared Error (MSE), Standard Deviation, Maximum and Minimum error values using Euclidean distances for each situation presented.

The estimated points are dispersed, but concentrated in an area that includes each of the real positions. Looking at Figure 4.15 and with the support of the MSE and Standard Deviation values from the 4.6 table it is possible to draw some conclusions.

The estimates for position 2, near the top corner, are both accurate and precise, while the estimates for position 1, at the center, are accurate, but not as precise. This is because the signal strength is stronger in the zones near the gateways. In the central zone, there is only one gateway capturing a strong signal, while in each corner there are at least 3 nearby gateways that do so.

Note that in position 3 the greatest dispersion results from the fact that the real position is not close to any particular gateway. As for the exceptional position 4, which is not within the experiment zone, but close enough to be captured by the five gateways, the results reveal a precise position, but not very accurate.

4.4 System Performance Evaluation

Keeping the area unoccupied and free of objects that could serve as obstacles, data collection was continued with the aim of evaluating the performance of the system.

In this section, graphs and heatmaps will be presented to evaluate the system's performance. Starting with the receiver operating characteristic curve (ROC curve), the accuracy will be evaluated in function of the defined tolerance. Afterwards, the calibration and effectiveness of the engine will be tested with a small but representative size of samples. A benchmark for the positioning algorithm will also be provided with the Cramer-Rao lower bound as the optimal estimator reference. Finally, to check the average precision at each place, a heatmap will be presented showing the zones with higher and lesser precision.

ROC Curve

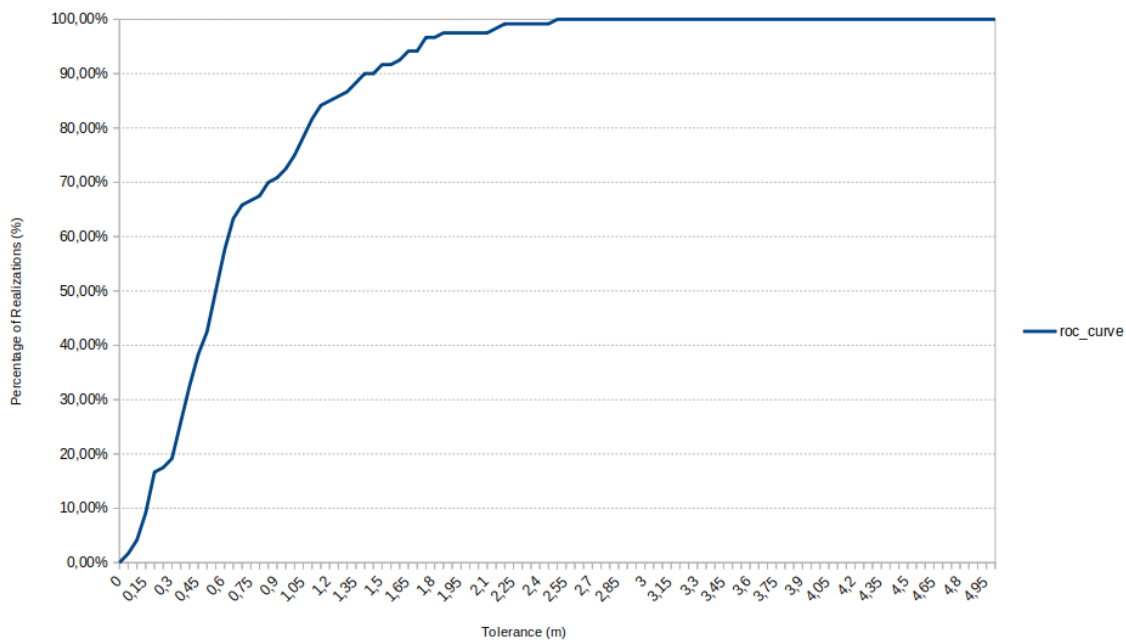


Figure 4.16: Graphical plot of percentage of realizations while considering tolerance (defined as the maximum Euclidean error in which a realization is considered).

The graphical plot of Figure 4.16 illustrates the performance of this localization estimation system in percentage of good localization estimations as its discrimination threshold is varied (tolerance, which is defined as the maximum Euclidean error in which a realization is considered). This graph may be regarded as a receiver operating characteristic curve (ROC), in which the plot is created by relating the true positive rate and the false positive rate at set threshold. The true-positive rate can be interpreted as a sensitivity measure and the false-positive rate as specificity one. It is possible to visualize the evolution in the estimation performance as the tolerance criterion changes. It is possible to observe that if the user of the system wants to have an excellent accuracy, with a tolerance of 2 meters it is already obtained a percentage of correctness of 98%. Furthermore, over 2.55 meters 100% accuracy is achieved, thus eliminating the possibility of false positives. If, on the contrary, the application requirement is higher, a tolerance of 1.5 meters is recommended, giving a 90% confidence in realizations. The usage of this technology is no longer of interest below 1 meter of tolerance, because the false-positive rate begins to rise above 30%, which is not acceptable in many applications.

Confusion Matrix

An interesting way to evaluate the implemented positioning engine calibration and the quality of the received relative strength signals could be through a table that observes the behavior of the estimates in relation to the collected measurements.

Using a sample of 300 estimate sites at random places throughout the zone, the data gathered for the position estimates and measurements was rated as correct or incorrect.

For the estimations, a tolerance of 1.5 meters was used, meaning that for an estimation to be considered correct the Euclidean distance between the real and the estimated point must be less than 1.5 meters. The relative strength signal measurements were labeled according to the number of signals with a tolerable proximity estimate.

In this experiment, it is assumed that the 5 gateways are guaranteed to receive the beacon signal at any point in the test zone, meaning that for each output coordinate 5 proximity distances are estimated. A proximity measure is considered defective if the resulting distance is greater than 3 meters than the actual distance between the beacon and the corresponding gateway. A RSS measurement is only labeled as correct measurement if at least 3 of the 5 proximity estimates fed to the system are not defective.

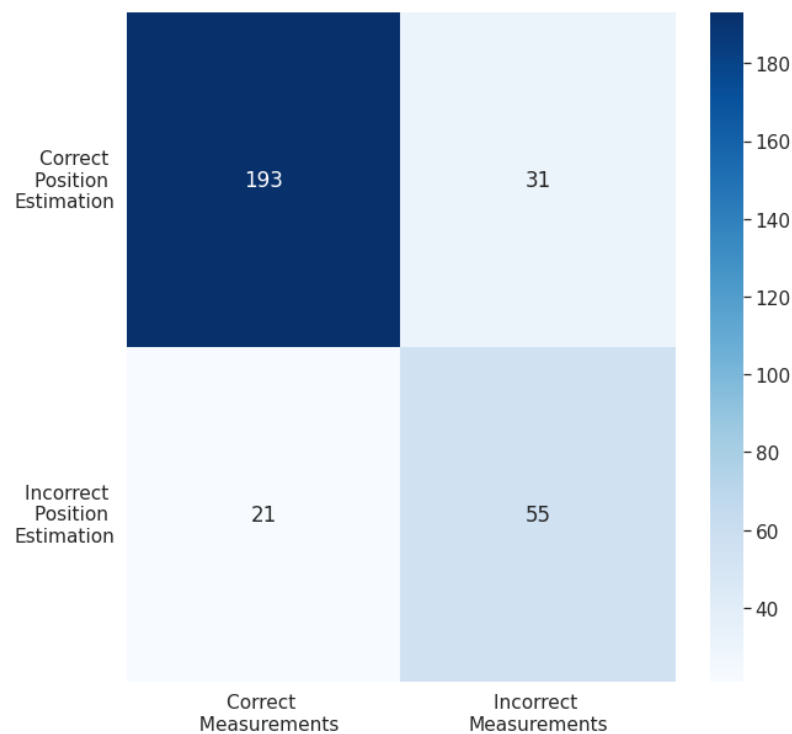


Figure 4.17: Positioning Engine's Confusion Matrix.

The result of this experiment can be observed in Figure 4.17. This can be regarded as the positioning engine's confusion matrix, defining 4 possible situations.

- **True-Positive:** the situation in which the system lived up to expectations. The measurements received were correct and the system was able to correctly estimate the position within the tolerance range.
- **True-Negative:** when the system was fed incorrect measurements which resulted in an estimation failure. Meaning a failure in the robustness or quality of the signals received by the beacon.
- **False-Positive:** the situation in which, despite the measurements being defective, the system estimated the position correctly. It may mean that even if for this particular case the estimation was correct, on another occasion it may fail due to signal noise or high fluctuations.
- **False-Negative:** this is an indication of the calibration of the system or the algorithm used, when the correct measurements are fed but the output is an incorrect position estimate.

Overall, despite the tolerance placed being a little demanding, the system showed acceptable behavior for a prototype, with a majority of 193 results in which the values matched expectations, the signals were within the expected parameters and the system estimated the position correctly. Although there were 55 situations where it unperformed, the system shown to be sufficiently calibrated with only 21 false-negatives. The 31 false-positives can be attributed due to symmetry zones in which, although the signals suffered some interference due to distance, as the distances were the same for opposing gateways the algorithm managed to cancel them out.

Cramer-Rao Lower Bound

To evaluate the proposed positioning technique in terms of the robustness of the localization results, the circular multilateration algorithm used in the project is compared with the Cramer-Rao lower bound. This will provide insight to the robustness of this technique when errors and noise are introduced into the system, in other words, it will evaluate on how the accuracy degrades as the positioning technique model differs from the optimum one. The Cramer-Rao bound indicates the minimum variance an unbiased estimator of the position can achieve, thus, giving a benchmark for the MSE error as the signal strength variance increases [31].

The Cramer-Rao lower bound for the localization problem with RSS measurements is derived in [33]. Assuming the Log-Distance pathloss model, the Fisher information matrix F for a area with N reference anchors (in this case the gateways) and one target beacon with unknown position is given by

$$F = \begin{bmatrix} F_{xx} & F_{xy} \\ F_{xy} & F_{yy} \end{bmatrix} \quad (4.13)$$

where F_{xx} , F_{xy} , F_{yy} are given by the following expressions.

$$F_{xx} = b \cdot \sum_{i=1}^N \frac{(x-x_i)^2}{d_i^4}, \quad F_{xy} = b \cdot \sum_{i=1}^N \frac{(x-x_i)(y-y_i)}{d_i^4}, \quad F_{yy} = b \cdot \sum_{i=1}^N \frac{(y-y_i)^2}{d_i^4} \quad (4.14)$$

with b being:

$$b = \left(\frac{10 \cdot n}{\sigma \ln 10} \right)^2 \quad (4.15)$$

For consistency, to plot the Crammer-rao bound, the same values for the path loss exponent ($n = 4$) and gateways center positions were used.

The data collection for the system implemented was done by varying randomly the positions of the beacon tag throughout the area, from a point where the variation of the signals was small and moving away until there was greater agitation.

Figure 4.18 shows the average squared localization errors as a function of the standard deviation σ of the RSS measurements registered in the systems database.

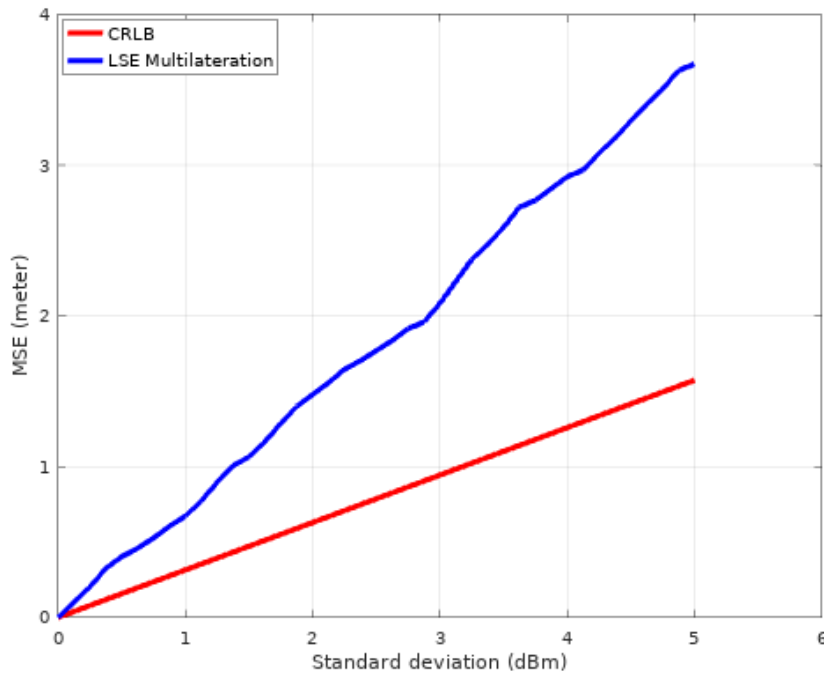


Figure 4.18: Average localization error (calculated as the square root of the MSE) as a function of the standard deviation of the RSS measurements. The Cramer–Rao bound is shown as comparison.

Observing the results, it is possible to state that as the fluctuations of the signals increase, the degradation of accuracy and, therefore, efficiency decreases in relation to the lower limit possible to reach. Furthermore, at 5 dBm of standard deviation the error results were more than double than the reference. However, the system seems to have a good response to low variations, noting that for values below 3 dBm standard deviation the system presented results close to the Cramer–Rao bound, with less than 1 meter of MSE of difference.

Average Precision Heatmap

One way to also evaluate the system is to also see how the average precision varies across the experiment zone. For this purpose, all the results of the previous experiments, present in the database, were taken into account. Points whose results were not included in the database were interpolated.

Figure 4.19 illustrates the heatmap obtained with the mean squared error values using euclidean distances along the experiment zone.

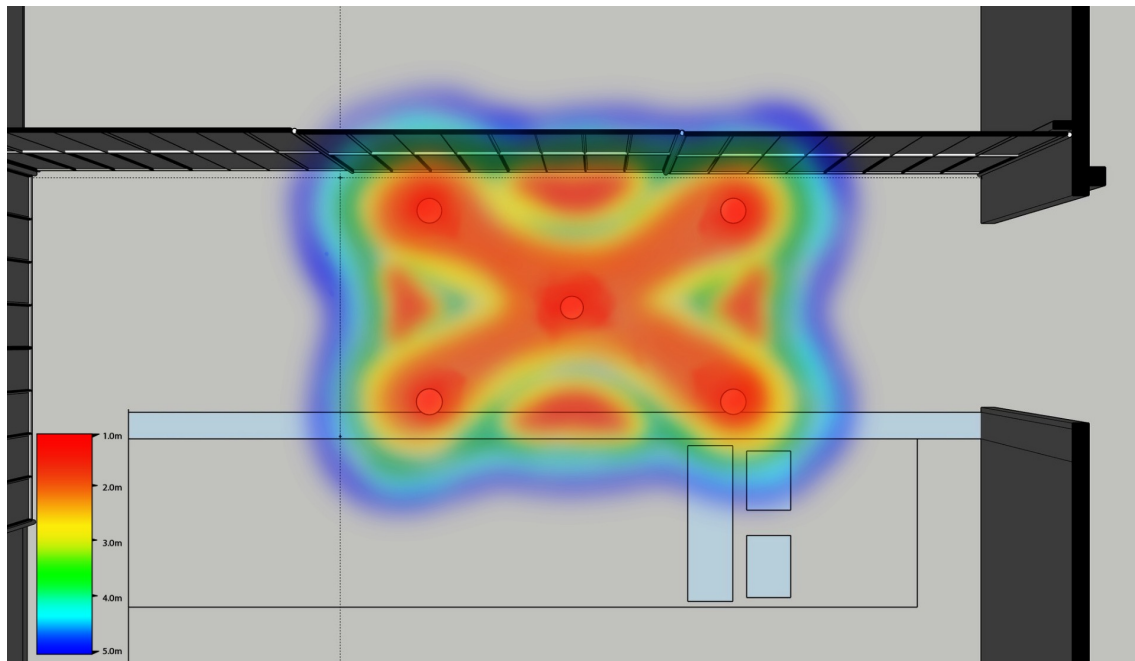


Figure 4.19: MSE Heatmap based on the location.

Observing the results, it is possible to affirm that a tendency of the results to have lower MSE in the zones in which the gateways are centered is noted. This is because the signal strength is higher in these places, giving a good indication of presence for the algorithm. A greater precision is also observed in the symmetry zones than in some interior zones, this effect is due to the fact that in these zones the signal obtained from the gateways is similar minimizing the influence of some of the fluctuations. It's also important to note that the further the beacon card moves away from the experiment zone, the greater the error, as it moves further away from the area where the gateways are placed.

Chapter 5

Conclusions and Future Work

In this thesis, a new indoor localization system has been developed to fulfill the needs of the company RACELAND S.A., which was having problems with time optimization of the tasks.

This chapter is dedicated to reflect on the work done and the results achieved. Although the system is able to comply with some of the requirements, it is also necessary to expose the limitations and what can be done in the future to improve the system.

The chapter ends with some final remarks about the broad challenges of indoor positioning systems.

5.1 Conclusions

Developing a precise positioning algorithm with BLE technology in indoor industrial environments is a sizeable challenge. However, the developed system was a worthwhile effort as it was capable of keeping track the target's location in a reasonable accuracy tolerance.

This system is composed by several small subsystems that once they are combined provide what the author believes is a true feasible solution for the problem. The architecture was modular enough that if any changes were required, only a tiny portion of the system would need to be restructured. The system is also scalable and can easily be expanded to cover larger areas.

It was demonstrated that while the target is within the zones where gateways can capture strong signals, it is possible to achieve high accuracy. The further it strays away, less accurate it will be.

The results from section 4.4 proved to be promising as a first step in the creation of a proof of concept, proving that even with cheap equipment it is possible to obtain an indoor positioning system that matches the needs of a company.

5.2 Achievements and Future Work

Future work will focus on scaling the system to the rest of the warehouse and verify if it maintains the same results in different environments. In the used experiment zone, the placement of the gateways and the interference-free space promoted conditions close to ideal for data collection.

However, these conditions will not always be possible. The warehouse contains zones with various objects and moving personnel and vehicles that will disturb beacon signals. Also, some of the sites are more complex to hang the gateways which means that the same configuration and placement of the gateways will not be possible. It would be crucial to see how the system reacts in less-than-ideal scenarios in order to make progress.

One of the important parts that has not yet been done is the integration of the task management system with the invoicing system in use. One of the company's ideas was the automation and verification of performance indicators and lead times. This system will be used as a complement to the PHC's task register to collect the necessary data on the company's ongoing activities.

An improvement to be considered will also be the optimization of the positioning algorithm by adding multiprocessing to obtain faster refresh rates. The developed positioning service was done synchronously, agglomerating all the data and processing one object at a time. It would be better if this processing were parallelized.

5.2.1 Achievements

As mentioned above, despite the contributions made, not everything was achieved. This subsection will be used to reflect on what requirements have been accomplished alongside some reasoning. To this end, the tables 3.2 and 3.3 from chapter 3 will be brought back, however this time each requirement will be categorized as being accomplished, partially accomplished or not done.

As for the functional requirements, the results can be observed in the table 5.1 below.

Code of the Requirement	Features Associated	Description	Achievement
RF1	F1, F3, F6	Gateways should be able to communicate the information with the RACELAND server	accomplished
RF2	F2, F4	Gateways must be strategically positioned with distances between them less than 10m	accomplished
RF3	F2, F7	Gateways positions must be measured with coordinates relative to some fixed points	accomplished
RF4	F3	Gateways values must be parsed and placed in an MS SQL Server database	accomplished
RF5	F1, F3	There should be a class distinction between objects and employees	not done
RF6	F3	Implementation of an algorithm capable of processing RSSI values of tags in coordinates	accomplished
RF7	F5	Implement a BI dashboard to evaluate the results in the database	not done
RF9	F8	Each of the tags must be configured to use the iBeacon protocol	accomplished

Table 5.1: Functional requirements.

With the exception of the BI dashboard and the distinction between an employee and an asset, almost all functional requirements were met. In relation to the dashboard, neither the integration with the company's tasks nor the definition of Key Performance Indicators (KPIs) or metrics that can benefit from the data stored in the database were done. The moment the system is scaled and well tested for the rest of the warehouse, it will have sufficient conditions to be able to move on to more practical uses.

As for RF5, more effort was put into creating a system that would estimate the position of the beacon tag, whether it is assigned to an asset or an employee. As such, no class was implemented yet that distinguishes the tag holder. Anyway, the system is configurable and will be a feature that will be added in the future.

As far as performance requirements are concerned, the following table 5.2 below is presented to list what has been achieved.

Code of the Requirement	Features Associated	Description	Achievement
RP1	F3	Position accuracy should have a maximum error of 2 meters	not done
RP2	F6	The frequency of sending data from gateways should be every 30 seconds	accomplished
RP3	F2	Data processing should be done asynchronously	accomplished
RP4	F1	The system should run uninterruptedly as a service on the server	accomplished
RP5	F5	The data processing should handle sets of 40 to 50 multiple payloads without any loss of information	partially accomplished

Table 5.2: Performance requirements.

The RP1 requirement was marked as "not done". Although, as shown in the ROC curve of the 4.4 section, the system has already presented a good confidence margin for tolerances above 2 meters, there is still a significant percentage of false positives. Furthermore, the tests were carried out under near-ideal conditions. These results are expected to worsen as the system is put to the test in various situations, which in turn will result in changes in the estimator and in the signal processor.

The RP5 requirement has been marked as "partially accomplished". As mentioned above, despite the system having an architecture that allows scaling to a large number of devices, the tests have not yet been performed and it is not possible to predict what will happen.

5.3 Final remarks

Along the chapters, a possible path was presented, which started by introducing the problem and ended up becoming a working prototype for an application that could be introduced in an increasingly growing market.

Indeed, indoor localization is a challenging subject because of the presence of wide range of obstacles including objects, walls, people, vehicles, to name a few, that results in increased multipath signals and effects. In addition, the wireless technology used will dictate the type of estimation algorithms that will influence the accuracy that it will be possible to achieve.

Furthermore, the localization or proximity detection accuracy requirements for indoor environments are pretty demanding comparing to GPS technologies, which have a 5-10 meters accuracy [11]. For street level applications this accuracy is feasible, however, for indoor environments like an industrial warehouse these large localization/proximity errors are not tolerable.

References

- [1] Neil E Klepeis. The national human activity pattern survey (nhaps): a resource for assessing exposure to environmental pollutants, February 2001.
- [2] João Manuel Jorge Barbosa. Position reference with direction estimator for indoor location systems, July 2015. Faculdade de engenharia da universidade do Porto, available in <https://hdl.handle.net/10216/79612>.
- [3] Pankaj Lanjudkar. Indoor positioning and indoor navigation (ipin) market outlook: 2025, November 2020. [Online], Available: <https://www.alliedmarketresearch.com/indoor-positioning-and-indoor-navigation-ipin-market>.
- [4] Sebastian Sadowski and Petros Spachos. Rssi-based indoor localization with the internet of things. *IEEE Communications Magazine*, pages 1–13, April 2018.
- [5] Apple. Getting started with ibeacon, June 2014. [Online], Available: <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>.
- [6] Google. Get started with beacons, December 2020. [Online], Available: <https://developers.google.com/beacons/eddystone>.
- [7] Adelantado Ferran. Understanding the limits of lorawan. *IEEE Communications Magazine*, pages 1–13, April 2017.
- [8] Zigbee Alliance. What is zigbee, November 2018. [Online], Available: <http://www.zigbee.org/>.
- [9] AB&R. What is rfid and how does rfid work?, June 2020. American Barcode and RFID Available: <https://www.abr.com/what-is-rfid-how-does-rfid-work/>.
- [10] Yuntian Brian Bai. Rfid-based indoor positioning technologies – where are we? *2012 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–10, November 2012.
- [11] João Neves Moutinho. Indoor sound based localization, March 2016. Faculdade de engenharia da universidade do Porto, available in <https://hdl.handle.net/10216/102449>.
- [12] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system, January 1992. Transactions on Information Systems. ISSN: 10468188. Available: <http://dl.acm.org/citation.cfm?id=128756.128759>.
- [13] Dominic Marcellino. An examination of ultra-wideband (uwb) for positioning & location tracking, November 2018. Link Labs AirFinder Available: <https://www.airfinder.com/blog/ultra-wideband-positioning-location-tracking>.

- [14] Fusco Giovanni and Coughlan James. Indoor localization using computer vision and visual-inertial odometry, June 2018. Published in final edited form as: Comput Help People Spec Needs.
- [15] J. Kunhoth, A. Karkar, and S. Al-Maadeed. Indoor positioning and wayfinding systems: a survey. *Hum. Cent. Comput. Inf. Sci.* <https://doi.org/10.1186/s13673-020-00222-0>, 2020.
- [16] Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello. The location stack: A layered model for location in ubiquitous computing. *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, pages 1–7, May 2002. University of Washington Computer Science & Engineering.
- [17] Christian Kray Champika Ranasinghe. Location information quality: A review, September 2018. Institute for Geoinformatics, University of Muenster, Germany.
- [18] Witold PedryczShyi-Ming Chen. *Information Granularity, Big Data, and Computational Intelligence*. Springer, 1st edition edition, 2015.
- [19] Stevenson, Ye J., Dobson S.A., and Nixon P. Loc8: A location model and extensible framework for programming with location, September 2010. IEEE Pervasive Computing.
- [20] O.S. Oguejiofor, Okorogu V.N., Adewale Abe, and Osuesu B.O. Outdoor localization system using rssi measurement of wireless sensor network. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, pages 1–7, January 2013. Blue Eyes Intelligence Engineering & Sciences.
- [21] Chanwoong Jo and Chaewoo Lee. Multilateration method based on the variance of estimated distance in range-free localisation. *ELECTRONICS LETTERS*, pages 1–7, June 2016.
- [22] Scipy. Optimization documentation, January 2021. [Online], Available: <https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html>.
- [23] S. R. Leelavathy and S. Sophia. Providing localization using triangulation method in wireless sensor networks. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, pages 1–4, November 2014. Blue Eyes Intelligence Engineering & Sciences.
- [24] Marcel Estel and Laura Fischer. Feasibility of bluetooth ibeacons for indoor localization, September 2015. Digital Enterprise Computing.
- [25] Marcos Brandão Duarte. Localização para smart devices tirando partido de ibeacons, February 2016. Faculdade de engenharia da universidade do Porto, available in <https://hdl.handle.net/10216/85340>.
- [26] HiveMQ. Hivemq introduction, December 2019. [Online], Available: <https://www.hivemq.com/docs/hivemq/4.6/user-guide/introduction.html>.
- [27] Dr. Robert P. Minch. Privacy issues in location-aware mobile devices. *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004.
- [28] Ana Grgat. Data privacy vs. data security, November 2020. [Online], Available: <https://dataprivacymanager.net/security-vs-privacy/>.

- [29] Hulya Vural, Murat Koyuncu, and Sinem Guney. A systematic literature review on microservices. *International Conference on Computational Science and Its Applications*, pages 203–217, 2017.
- [30] Paolo Bellavista, Antonio Corradi, and Carlo Giannelli. Evaluating filtering strategies for decentralized handover prediction in the wireless internet. *Dip. Elettronica, Informatica e Sistemistica - Università di Bologna*, 2006.
- [31] Paula Tarrío, Ana Bernardos, and José Casar. Weighted least squares techniques for improved received signal strength based localization. *Multidisciplinary Digital Publishing Institute (MDPI)*, pages 1–13, August 2011.
- [32] Y. Gau, H. Chu, and R. Jan. A weighted multilateration positioning method for wireless sensor networks, available: <https://doi.org/10.1108/17427370710856246>. *International Journal of Pervasive Computing and Communications*, pages 289–303, August 2007.
- [33] Patwari N, Perkins M, and Correal N. Relative location estimation in wireless sensor networks. *Multidisciplinary Digital Publishing Institute (MDPI)*, August 2011.

Appendix A

Hardware

In this appendix, the gateways and beacons used will be presented alongside a brief explanation of their operation and settings.

A.1 G1 Minew Gateways

Minew Gateway G1 is an open source and completely configurable device made by *Minew Technologies Co.*, a professional IoT manufacturer. This device captures data from iBeacon, Eddystone, BLE sensors and other BLE devices and sends it to the local TCP server via HTTP or MQTT protocol.



Figure A.1: Gateway G1.

G1 Gateway is based on OpenWrt. OpenWrt is a highly extensible GNU/Linux distribution for embedded devices. Unlike other distributions, OpenWrt is built from the ground up to be a full-featured, easily modifiable operating system for embedded devices.

G1 BLE & WiFi gateway can be configured through a browser with a simple Web configuration interface, once it is connected to the local network.

Figure A.2: G1 Web configuration interface

As shown in Figure A.2, the configuration interface allows the developer to change the parameters of the network, the service and others in order to better adapt to the application that is intended to be carried out.

For the configuration of the network, a range of IP addresses was made available. Each Gateway is assigned a static IP address for the 192.168.1.0 subnet with subnet mask 255.255.255.0 (/24). In this way, it is possible to identify each gateway by its own IP address and helps with stability during the execution of the program.

In the configuration of the service, it is possible to choose the message protocol, the upload interval, the BLE format data and also filter the information that is uploaded to better suit the needs. For the protocol, the MQTT was chosen for reasons mentioned in the subsection 2.7. For the BLE Data, it was chosen the "Long Json" format with an advertising interval of 5 seconds. In other words, each gateway will publish the received iBeacon data in json format each 5 seconds. The MAC address will also be published to trace-back the origin of the message.

A.2 C7 iBeacon Tags

Minew's C7 iBeacon Tags are based on an Nordic nRF52 microcontroller and its only function is the ability to advertise data via BLE Bluetooth 5.0 at a certain power and frequency in iBeacon packet format.



Figure A.3: C7 Tag iBeacon

To change the transmission parameters of the beacon, the manufacturer provides a mobile application for iphone’s IOS called "*BeaconSet+*".

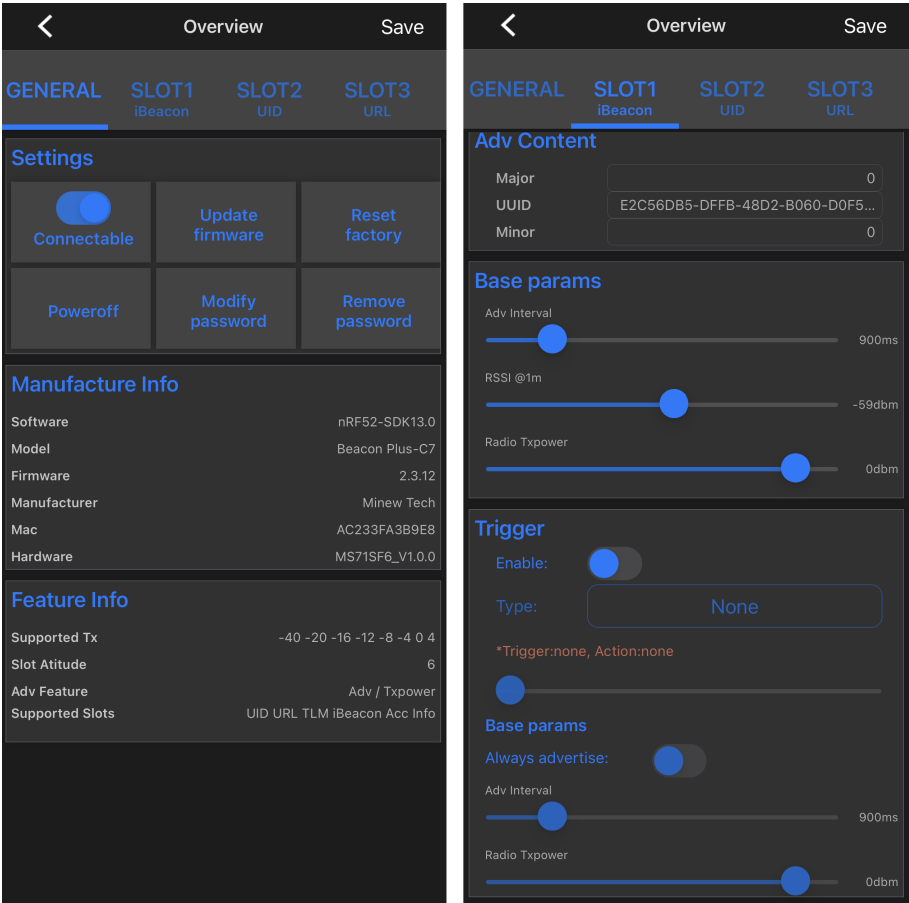


Figure A.4: Configuration menu of "BeaconSet+" mobile application

The app will automatically discover every beacon in it’s range and present a list of them. The

user can choose the beacon tag to configure and it will open the settings page as illustrated in Figure A.4. In this page the manufacturer and the feature info are shown. By navigating to the iBeacon slot it is possible to change the advertising content such as the Major, the minor and the UUID of the iBeacon packet. For this project the UUID remained unchanged. The Minor field was used to identify if the beacon's holder is an object or an employee. The Major field was used to identify the type of object or person. As for the base parameters, the default configuration was used with 900ms as advertising interval, -59dBm as the reference RSSI and 0dBm transmit power. To note that these parameters can be change to enhance the localization accuracy.

The Trigger function enables the Beacon's holder to publish a trigger message to alert to some event. This ability is not useful for the project and was disabled.

Appendix B

User Interface Prototype

An important touch that complements the project has to do with the user-friendly way in which the results are presented to the application's user.

In this appendix some images of the windows of the prototype of the developed dashboard will be presented, along with a simple explanation of the functionality of each window.



Figure B.1: Inntial Menu

In Figure B.1 the design of the initial menu is shown. When starting the application, this will be the first window that will be opened to the user. From this window the user can choose what he wants to observe or configure. 5 icons are shown, representing 5 of the main warehouse zones. Clicking on one of these icons will show a window with the results of the tasks being executed in that location.

Clicking the "Map" button will open the window shown in Figure B.2 below.

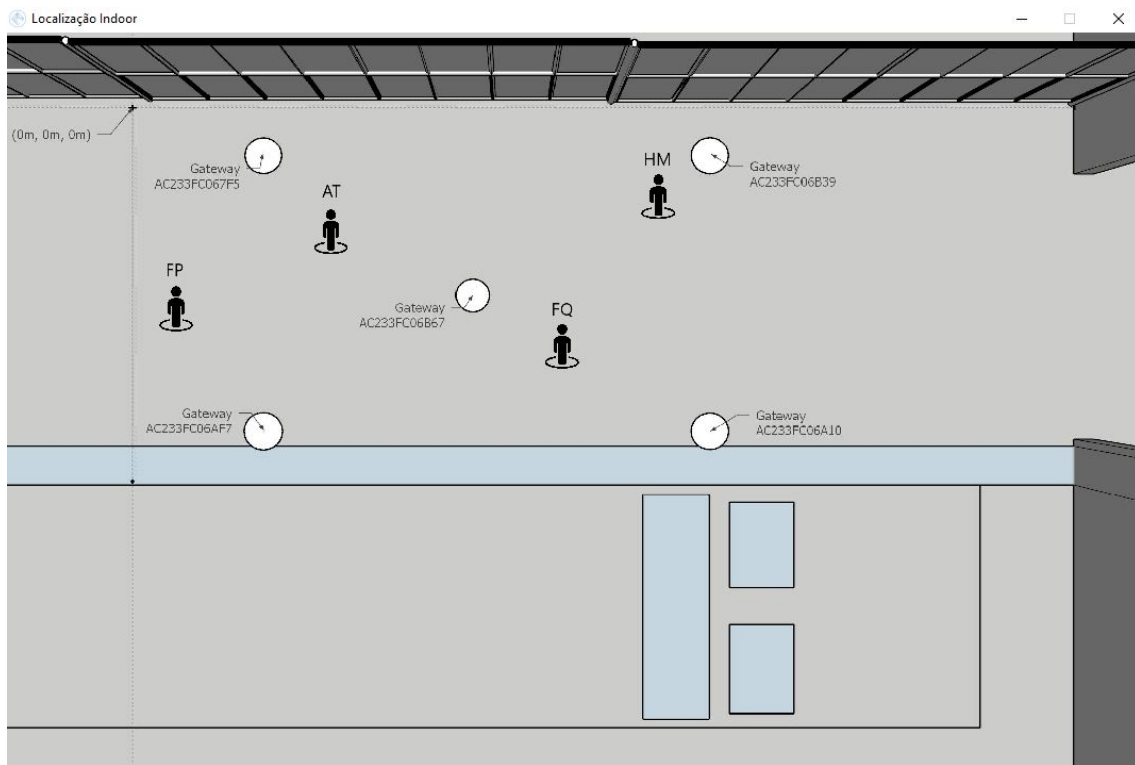


Figure B.2: Dashboard's Positioning Map

This is a rendering of the 3D model developed from the warehouse plant, where the user will be able to observe in real time the position of the personnel, identified by an icon and the initials of their name above. The gateways placed will also be shown with their MAC address to be able to identify themselves.

From the initial menu, it is possible to configure the displayed contents, with a click on the "Configure" button.

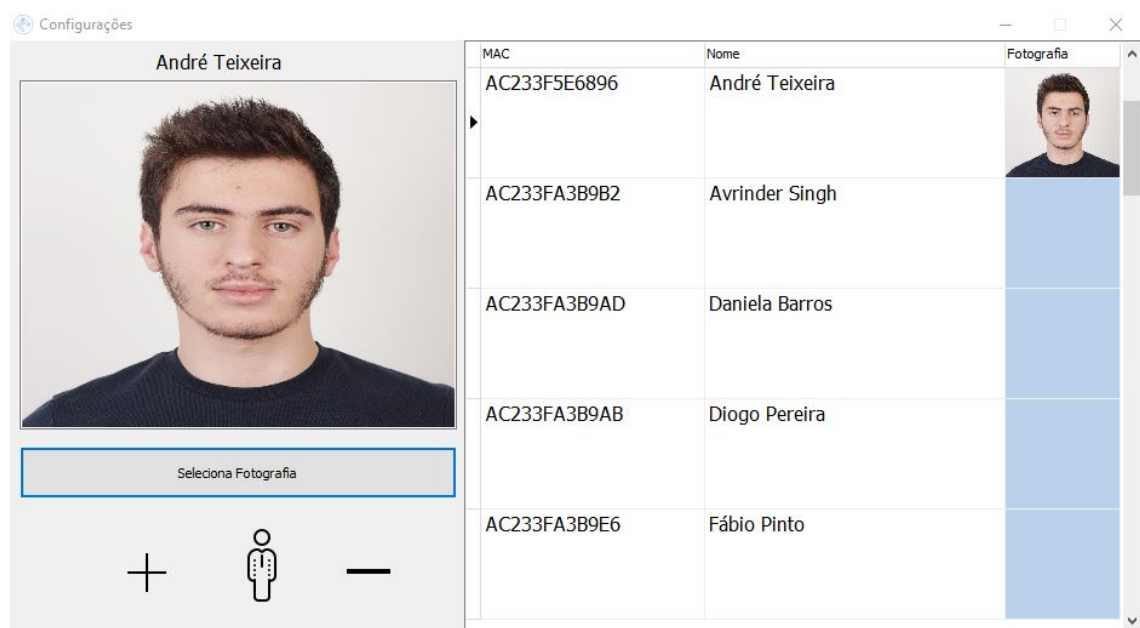


Figure B.3: Dashboard's Configuration Menu

In the configuration menu, Figure B.2, the user is able to edit the contents in the database.

This window has capabilities to add, delete or modify the employee's name and photo and the assigned beacon's MAC address.

Appendix C

Developed Server Programs

Parsing_f

June 28, 2021

```
[ ]: import asyncio
```

```
[ ]: import mqtttools
```

```
[ ]: import pyodbc
```

```
[ ]: import json
```

```
[ ]: import logging
```

```
[ ]: #####  
#                               #  
#####
```

```
[ ]: class Gateway():  
  
    def __init__(self, message):  
        self.type= message["type"]  
        self.mac= message["mac"]  
        self.timestamp= message["timestamp"]  
  
    def __str__(self):  
        return str(self.mac)  
  
    def __eq__(self, other):  
        return self.__str__() == other.__str__()  
  
    def get(self, attribute):  
        if attribute == "type":  
            return self.type  
        elif attribute in "mac":  
            return self.mac  
        elif attribute == "timestamp":  
            return self.timestamp  
  
        else:
```

```

        return None

    async def write(self, conn):
        cursor = conn.cursor()
        cursor.execute(
            'INSERT INTO edificiolocal'
            ' (Nome'
            ' ,Tempo) '
            'VALUES '
            ' (?, '
            ' ?) ',
            (self.mac , self.timestamp)
        )
        conn.commit()

    async def write_edificio(self, conn):
        cursor = conn.cursor()
        cursor.execute(
            'BEGIN TRY '
            'INSERT INTO EDIFICIO'
            ' (MAC'
            ' ,NAME'
            ' ,OP_TIMESTAMP) '
            'VALUES '
            ' (?, '
            ' ? , '
            ' ?) '
            'END TRY '
            'BEGIN CATCH '
            'IF ERROR_NUMBER() = 2627 '
            'update EDIFICIO set NAME= ?, OP_TIMESTAMP= ? where MAC_
↳ = ? ; '
            'END CATCH',
            (self.mac , self.name, self.timestamp, self.name, self.timestamp,
↳ self.mac)
        )
        conn.commit()
        #read(conn)

```

```

[ ]: class Beacon():

    def __init__(self, message, gatewaymac, gatewaytime):
        self.type= message["type"]
        self.mac= message["mac"]
        self.timestamp= message["timestamp"]
        self.ibeaconUuid= message["ibeaconUuid"]
        self.ibeaconMajor= message["ibeaconMajor"]

```

```

self.ibeaconMinor= message["ibeaconMinor"]
self.rssi= message["rssi"]
self.ibeaconTxPower= message["ibeaconTxPower"]
self.gatewaymac=gatewaymac
self.gatewaytime=gatewaytime

def __str__(self):
    return str(self.mac)

def __eq__(self, other):
    return self.__str__() == other.__str__()

def get(self, attribute):
    if attribute == "type":
        return self.type
    elif attribute in "mac":
        return self.mac
    elif attribute == "timestamp":
        return self.timestamp
    elif attribute in "ibeaconUuid":
        return self.ibeaconUuid
    elif attribute == "ibeaconMajor":
        return self.ibeaconMajor
    elif attribute == "ibeaconMinor":
        return self.ibeaconMinor
    elif attribute == "rssi":
        return self.rssi
    elif attribute == "ibeaconTxPower":
        return self.ibeaconTxPower
    elif attribute == "gatewaymac":
        return self.gatewaymac
    elif attribute == "gatewaytime":
        return self.gatewaytime

    else:
        return None

async def write(self, conn):
    cursor = conn.cursor()
    cursor.execute(
        'INSERT INTO colaborador'
        ' (Nome'
        ',RSSI'
        ',Tempo'
        ',Local) '
        'VALUES'

```

```

        '(', '
        '?', '
        '?', '
        '?) ',
        (self.mac , self.rssi, self.gatewaytime, self.gatewaymac)
    )
    conn.commit()
    async def write_employed(self, conn):
        cursor = conn.cursor()
        cursor.execute(
            'BEGIN TRY '
            'INSERT INTO EMPREGADO'
            ' (MAC'
            ',RSSI'
            ',NAME'
            ',OP_TIMESTAMP) '
            'VALUES'
            ' (', '
            '?', '
            '?', '
            '?', '
            '?) '
            'END TRY '
            'BEGIN CATCH '
            'IF ERROR_NUMBER() = 2627 '
            'update EMPREGADO set RSSI= ?, NAME= ?, OP_TIMESTAMP= ?_
↪where MAC = ? ; '
            'END CATCH',
            (self.mac , self.rssi, self.name, self.timestamp, self.rssi, self.
↪name, self.timestamp, self.mac)
        )
        conn.commit()
        #read(conn)

```

```

[ ]: #####
#           Universal Database Functions           #
#####

```

```

[ ]: async def read(conn, table="dummy"):
    print("\n\n")
    #print("-----Read-----\n")
    print(f"-----{table}-----\n")

    cursor = conn.cursor()

    cursor.execute(
        f"select * from {table}"
    )

```

```

for i, row in enumerate(cursor):
    print(f'row {i+1}: {row}')
print("\n")

```

```

[ ]: async def create_employment(conn, table="EMPREGADO"):
    print(f"_____Create_{table}_____\\n")
    cursor = conn.cursor()
    cursor.execute(
        'CREATE TABLE EMPREGADO ('
        'MAC nchar(12) NOT NULL PRIMARY KEY,'
        'RSSI int,'
        'NAME varchar(100),'
        'OP_TIMESTAMP DATETIME'
        ');'
    )
    conn.commit()

```

```

[ ]: async def create_edificio(conn, table="EDIFICIO"):
    print(f"_____Create_{table}_____\\n")
    cursor = conn.cursor()
    cursor.execute(
        'CREATE TABLE EDIFICIO ('
        'MAC nchar(12) NOT NULL PRIMARY KEY,'
        'NAME varchar(100),'
        'OP_TIMESTAMP DATETIME '
        ');'
    )
    conn.commit()

```

```

[ ]: async def insert(conn):
    print("_____insert_____\\n")
    cursor = conn.cursor()
    cursor.execute(
        'insert into dummy(a,b) values(?,?);',
        (3232, 'catzzz')
    )
    conn.commit()
    read(conn)

```

```

[ ]: async def update_gateway(conn, table= 'dummy'):
    print(f"_____Update_{table}_____\\n")
    cursor = conn.cursor()
    cursor.execute(
        'update EMPREGADO set NAME=? where MAC = ?;',
        ('dogzzz', 'AC233FA36C2C')
    )
    conn.commit()

```

```
read(conn)
```

```
[ ]: async def update_employado(conn, table= 'dummy'):  
    print(f"_____Update_{table}_____\n")  
    cursor = conn.cursor()  
    cursor.execute(  
        'update EMPREGADO set RSSI= %d, NAME=%s where MAC = %s;',  
        (-1 , 'dogzzz', 'AC233FA36C2C')  
    )  
    conn.commit()  
    read(conn)
```

```
[ ]: async def delete(conn, table= 'dummy'):  
    print(f"_____Delete_{table}_____\n")  
    cursor = conn.cursor()  
    cursor.execute(  
        f'delete TOP (100) from {table};'  
    )  
    conn.commit()  
    await read(conn)
```

```
[ ]: #####
```

```
[ ]: async def db_sender(conn, message):  
  
    #edificio_dic={"AC233FC067F5" : "Pintura"}  
  
    #colaborador_dic={"AC233FA36C2C": "Xavier", "AC233F5E6896": "Fábio"}  
  
    for m in message:  
        if m["type"]== "Gateway":  
  
            gateway=Gateway(m)  #, edificio_dic[str(m["mac"])])  
            await gateway.write(conn)  
  
            #print("debug ==> Gateway of: ", edificio_dic[str(gateway)], ' |  
↳Timestamp: ', gateway.get("timestamp"))  
  
            elif m["type"]== "iBeacon":  
  
                beacon=Beacon(m, gateway.get("mac"), gateway.get("timestamp"))  #,  
↳colaborador_dic[str(m["mac"])])  
                await beacon.write(conn)
```

```

        #print("debug ==> iBeacon of: ", colaborador_dic[str(beacon)]," |␣
↪RSSI: ", beacon.get("rssi"))

```

```

    else:
        pass

```

```

    #await read(conn, "EDIFICIO")
    #await read(conn, "EMPREGADO")

```

```

[ ]: async def handle_messages(client, conn, debug= False):
    while True:
        topic, message = await client.messages.get()

        if topic is None:
            print('Connection lost.')
            break
        message=json.loads(str(message.decode("utf-8","ignore")))
        if debug:
            ␣
↪print("-----")
            print(f'Got {message} on {topic}.')
            ␣
↪print("-----")
            print("\n")
            print("+++++")

            for i, m in enumerate(message):
                print("-----")
                print("# MESSAGE OF: ", m["type"], " Nº: ", i, "#")
                print("-----")
                for key, value in m.items() :
                    print (key,": ", value)

            ␣
↪print("+++++")
            await db_sender(conn, message)

```

```

[ ]: #ac233fc067f5/status', '/gw/ac233fc06b39/status', '/gw/ac233fc06b67/status', '/'
↪gw/ac233fc06a10/status', '/gw/ac233fc06af7/status

```

```

async def reconector():

```

```

client = mqtttools.Client('192.168.1.2',
                           1883,
                           subscriptions=['/gw/#'],
                           connect_delays=[1, 2, 4, 8])

conn = pyodbc.connect(
    "Driver={SQL Server};"
    "Server=192.168.1.2;"
    "UID=;"
    "PWD=;"
    "Database=;"
)

while True:
    #break
    await client.start()
    await handle_messages(client, conn, debug=True)
    await client.stop()
    await conn.close()

```

```

[ ]: import nest_asyncio
nest_asyncio.apply()

```

```

[ ]: ##run
logging.basicConfig(level=logging.INFO)

asyncio.run(reconnector())

```

```

[ ]:

```

FiltragemRSSI

June 28, 2021

```
[1]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
[2]: # from google.colab import files

# uploaded = files.upload()

# for fn in uploaded.keys():
#     print('User uploaded file "{name}" with length {length} bytes'.format(
#         name=fn, length=len(uploaded[fn])))
```

```
[3]: def plot_signals(signals, labels):

    """
    Auxiliary function to plot all signals.
    input:
        - signals: signals to plot
        - labels: labels of input signals
    output:
        - display plot
    """

    alphas = [1, 0.45, 0.45, 0.45, 0.45]      # just some opacity values to
    ↪ facilitate visualization

    lenght = np.shape(signals)[1]              # time lenght of original and
    ↪ filtered signals

    plt.figure()

    for j, sig in enumerate(signals):          # iterates on all signals

        plt.plot(range(lenght), sig, '-o', label=labels[j], markersize=2,
        ↪ alpha=alphas[j])

    plt.grid()
```

```

plt.ylabel('RSSI')
plt.xlabel('time')
plt.legend()
plt.show()

return

```

```

[6]: def kalman_block(x, P, s, A, H, Q, R):

    """
    Prediction and update in Kalman filter
    input:
        - signal: signal to be filtered
        - x: previous mean state
        - P: previous variance state
        - s: current observation
        - A, H, Q, R: kalman filter parameters
    output:
        - x: mean state prediction
        - P: variance state prediction
    """

    # check laaraiedh2209 for further understand these equations

    x_mean = A * x + np.random.normal(0, Q, 1)
    P_mean = A * P * A + Q

    K = P_mean * H * (1 / (H * P_mean * H + R))
    x = x_mean + K * (s - H * x_mean)
    P = (1 - K * H) * P_mean

    return x, P

```

```

[7]: def kalman_filter(signal, A, H, Q, R):

    """
    Implementation of Kalman filter.
    Takes a signal and filter parameters and returns the filtered signal.
    input:
        - signal: signal to be filtered
        - A, H, Q, R: kalman filter parameters
    output:
        - filtered signal
    """

    predicted_signal = []

```

```

    x = signal[0] # takes first value as first
    ↪ filter prediction
    P = 0 # set first covariance state
    ↪ value to zero

    predicted_signal.append(x)
    for j, s in enumerate(signal[1:]): # iterates on the entire
    ↪ signal, except the first element
        #flag = True
        #for s in signal: # iterates on the entire signal, except the
    ↪ first element
            # if flag: #code improve to save memory, signal[1:] in case big size,
    ↪ makes trouble.
            # flag = False
            # continue

    x, P = kalman_block(x, P, s, A, H, Q, R) # calculates next state
    ↪ prediction

    predicted_signal.append(x) # update predicted signal with
    ↪ this step calculation

    return predicted_signal

```

```

[8]: def choose_particle(particles):

    """
    Takes an array of particles and returns an element according to weights
    ↪ distribution
    input:
        - particles: array of particles
    output:
        - chosen particle
    """

    prob_distribution = []

    # calculates sum of weights to normalize weight vector in next step

    sum_weights = 0
    for p in particles:
        sum_weights += p['weight']

    for p in particles:
        prob_distribution.append(float(p['weight'] / sum_weights))

```

```

# choose particle according to weights distribution

a = np.random.choice(particles, size=1, replace=False, p=prob_distribution)

return a[0]['value'][0]

```

```

[9]: def particle_filter(signal, quant_particles, A=1, H=1, Q=1.6, R=6):

    """
    Implementation of Particles filter.
    Takes a signal and filter parameters and return the filtered signal.
    input:
        - signal: signal to be filtered
        - quant_particles: filter parameter - quantity of particles
        - A, H, Q, R: kalman filter parameters
    output:
        - filtered signal
    """

    predicted_signal = []

    rang = 10 # variation range
    ↪ of particles for initial step

    x = signal[0] # takes first
    ↪ value as first filter prediction
    P = 0 # set first
    ↪ covariance state value to zero

    predicted_signal.append(x)

    min_weight_to_consider = 0.07 # defines some
    ↪ needed constants in algorithm
    min_weight_to_split_particle = 5

    for j, s in enumerate(signal[1:]): # iterates on the
    ↪ entire signal, except the first element

        range_ = [predicted_signal[j-1] - rang,
                  predicted_signal[j-1] + rang] # set variation
    ↪ range for first step sampling

        particles = []

```

```

        for particle in range(quant_particles):                # loop on all
↪particles

            input = np.random.uniform(range_[0], range_[1])    # sample particle
↪value from variation range

            weight = 1 / np.abs(input-x)                        # particle weight

            if weight > min_weight_to_consider:                # it only
↪iterates on particles which weights

                                                                # are greater
↪than _min_weight_to_consider_

            x_, P = kalman_block(input, P, s, A, H, Q, R)      # calculates next
↪state prediction

            weight = 1 / np.abs(s - x_)                        # prediction
↪weight

            particles.append({'value': x_, 'weight': weight})

            # for particles with greater weights, it creates other particles
↪in the 'neighborhood'

            if weight > min_weight_to_split_particle:

                input = input + np.random.uniform(0, 5)
                x_, P = kalman_block(input, P, s, A, H, Q, R)

                weight = 1 / np.abs(s - x_)
                particles.append({'value': x_, 'weight': weight})

            x = choose_particle(particles)                      # choose a
↪particle, according to weight distribution

            predicted_signal.append(x)                          # update
↪predicted signal with this step calculation

        return predicted_signal

```

1 MAIN

```

[19]: #np.warnings.filterwarnings('error', category=np.VisibleDeprecationWarning)
np.warnings.filterwarnings('ignore', category=np.VisibleDeprecationWarning)
np.warnings.filterwarnings('ignore', category=np.ComplexWarning)

#file = pd.read_csv('sample.csv')

```

```

signal = [-69.0,-64.0,-59.0,-64.0,-57.0,-58.0,-57.0,-66.0,-55.0,-68.0,-68.0,-69.
↪0,-70.0,-69.0,-69.
↪0,-67,-65,-69,-76,-58,-64,-64,-65,-68,-62,-68,-64,-45,-64,-69,-70,-64,-63,-64,-65,-67,-68,-62
↪#file['rssi']
#print(len(signal), "\n")

# calculate filters

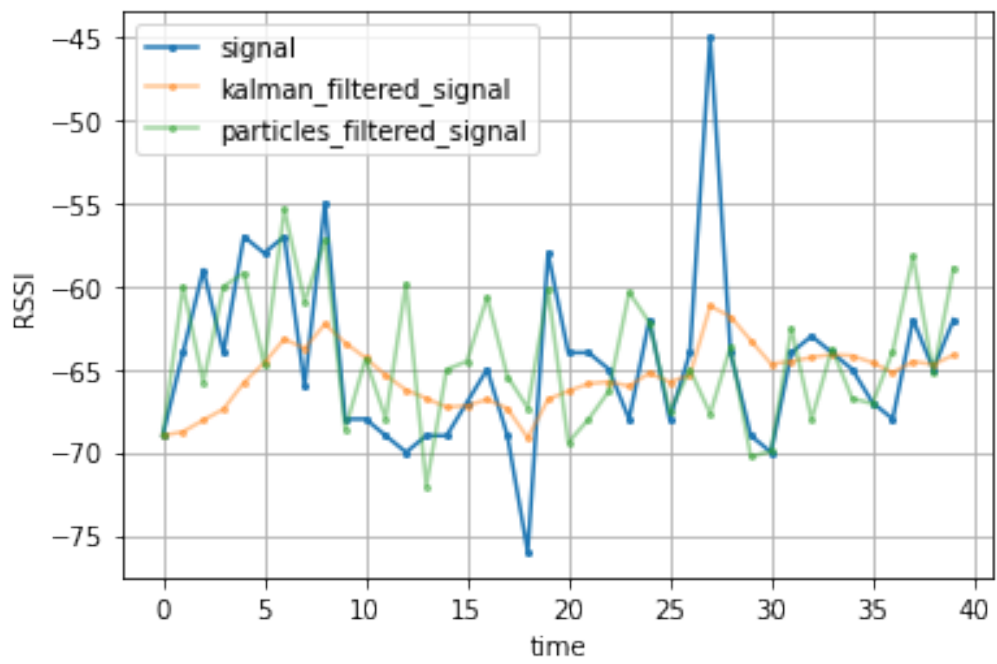
signal_kalman_filter = kalman_filter(signal, A=1, H=1, Q=0.1, R=2)
signal_particle_filter = particle_filter(signal, quant_particles=5, A=1, H=1,↪
↪Q=0.1, R=2)

#####
#      PLOT
#####
plot_signals([ signal,  signal_kalman_filter,      signal_particle_filter],↪
↪['signal', 'kalman_filtered_signal', 'particles_filtered_signal'])

#####
# Kalman Filter
#####
#plot_signals([ signal,  signal_kalman_filter],  ['signal', ↪
↪'kalman_filtered_signal'])

#####
# Particle Filter
#####
#plot_signals([ signal,  signal_particle_filter],  ['signal', ↪
↪'particles_filtered_signal'])

```



```
[11]: #!rm sample.csv
```

```
[12]: #!ls
```

```
[ ]:
```

coordenadas_f

June 28, 2021

```
[1]: #import asyncio

[2]: import pyodbc

[3]: #import logging

[4]: #####
    #import nest_asyncio
    #nest_asyncio.apply()
    #####

[5]: import math
    from math import sqrt

[6]: from functools import lru_cache
    from numba import int32, float32
    from numba.experimental import jitclass
    from numba import njit

[7]: import matplotlib.pyplot as plt
    import numpy as np
    from numpy import mean

[8]: import scipy

[9]: import multilateration

[10]: import time
    import datetime

[ ]:

[11]: #SETUP
    conn = pyodbc.connect(
        "Driver={SQL Server};"
        "Server=192.168.1.2;"
        "UID=;"
```

```

    "PWD=";
    "Database=";
)

```

```

[12]: #BUSCA COLABORADORES
colab={}
table='configcolaborador'
print("\n\n")

print(f"-----{table}-----\n")

cursor = conn.cursor()

cursor.execute(
    f"select * from {table}"
)
for i, row in enumerate(cursor):
    colab[row[0].strip()]= row[1].strip()

#print(colab)
for co in colab:
    print("key: ", co,"|value: ",colab[co])

```

```

-----_configcolaborador_-----

key:  AC233F5E6896 |value:  Anabela Bessa
key:  AC233FA36C2C |value:  Xavier Ricardo
key:  AC233FA3B9AA |value:  José Nunes
key:  AC233FA3B9AB |value:  Diogo Pereira
key:  AC233FA3B9AC |value:  Pedro Silva
key:  AC233FA3B9AD |value:  Daniela Barros
key:  AC233FA3B9AE |value:  Fernando Quinhentas
key:  AC233FA3B9AF |value:  José Freitas
key:  AC233FA3B9B0 |value:  Serafim Ferreira
key:  AC233FA3B9B1 |value:  Harinder Singh
key:  AC233FA3B9B2 |value:  Avrinder Singh
key:  AC233FA3B9B3 |value:  Xavier Ribeiro
key:  AC233FA3B9B9 |value:  André Teixeira
key:  AC233FA3B9BA |value:  José Silva
key:  AC233FA3B9BB |value:  Rui Nunes
key:  AC233FA3B9BD |value:  Henrique Silva
key:  AC233FA3B9BE |value:  Ricardo Remelgado
key:  AC233FA3B9BF |value:  José Gomes

```

```

key: AC233FA3B9C0 |value: Pedro Moreira
key: AC233FA3B9C1 |value: Mafalda Rodrigues
key: AC233FA3B9C2 |value: Timoteo Monteiro
key: AC233FA3B9E6 |value: Fábio Pinto
key: AC233FA3B9E7 |value: Ricardo Martins
key: AC233FA3B9E9 |value: Maria Silva
key: AC233FA3BA00 |value: Abel Silva

```

```

[13]: #BUSCA EDIFICIOS
edifconf={'MAC':0,'LOCAL':1,'X':2,'Y':3,'Z':4}
edif={}
edifparam=[]
table='configedificio'
print("\n\n")

print(f"-----{table}-----\n")

cursor = conn.cursor()

cursor.execute(
    f"select * from {table}"
)
for i, row in enumerate(cursor):
    for c in row:
        edifparam.append(str(c).strip())

    edif[row[0].strip()]=edifparam
    edifparam=[]

#print(edif)
for ed in edif:
    print("key: ", ed,"|value: ",edif[ed])

print("\n")

```

```

-----configedificio-----

```

```

key: AC233FC067F5 |value: ['AC233FC067F5', 'Tacógrafos', '0.769', '2.072',
'5.235']
key: AC233FC06A10 |value: ['AC233FC06A10', 'Tacógrafos', '5.168', '9.144',
'5.235']
key: AC233FC06AF7 |value: ['AC233FC06AF7', 'Tacógrafos', '5.168', '2.072',
'5.235']
key: AC233FC06B39 |value: ['AC233FC06B39', 'Tacógrafos', '0.769', '9.144',

```

```
'5.235']
key: AC233FC06B67 |value: ['AC233FC06B67', 'Tacógrafos', '2.977', '5.388',
'5.235']
```

```
[14]: import localization as lx
P=lx.Project(mode='2D',solver='LSE', detail=True)
for anchor in edif:
    print(anchor, "x: ", edif[anchor][2], "y: ", edif[anchor][3], "z: ",
    ↪edif[anchor][4])
    #etc=float(edif[anchor][2]) + float(edif[anchor][3])
    #print(etc)
    P.
    ↪add_anchor(anchor,(float(edif[anchor][2]),float(edif[anchor][3]),float(edif[anchor][4])))

t,label=P.add_target("hello")

t.add_measure("AC233FC067F5", 2.821526548426293)
t.add_measure("AC233FC06A10", 5.781175478138106)
t.add_measure("AC233FC06AF7", 0.8078378540858728)
t.add_measure("AC233FC06B39", 7.190596637693705)
t.add_measure("AC233FC06B67", 2.3292402796114073)

P.solve()

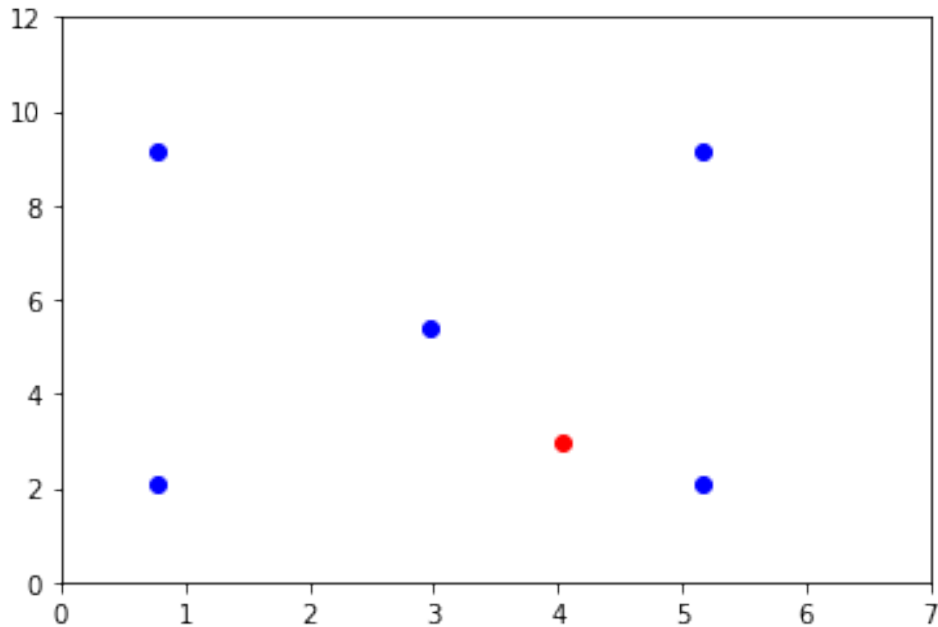
# Then the target location is:
for tar in P.TargetDic:
    print(tar, P.TargetDic[tar].loc)

plt.xlim(0, 7)
plt.ylim(0, 12)
plt.scatter(0.769, 2.072, c='blue')
plt.scatter(5.168, 9.144, c='blue')
plt.scatter(5.168, 2.072, c='blue')
plt.scatter(0.769, 9.144, c='blue')
plt.scatter(2.977, 5.388, c='blue')

plt.scatter(t.loc.x, t.loc.y, c='red')
plt.show()
```

```
AC233FC067F5 x: 0.769 y: 2.072 z: 5.235
AC233FC06A10 x: 5.168 y: 9.144 z: 5.235
AC233FC06AF7 x: 5.168 y: 2.072 z: 5.235
AC233FC06B39 x: 0.769 y: 9.144 z: 5.235
```

AC233FC06B67 x: 2.977 y: 5.388 z: 5.235
 hello p(4.046129991759708,2.9845751233629225,0.0)



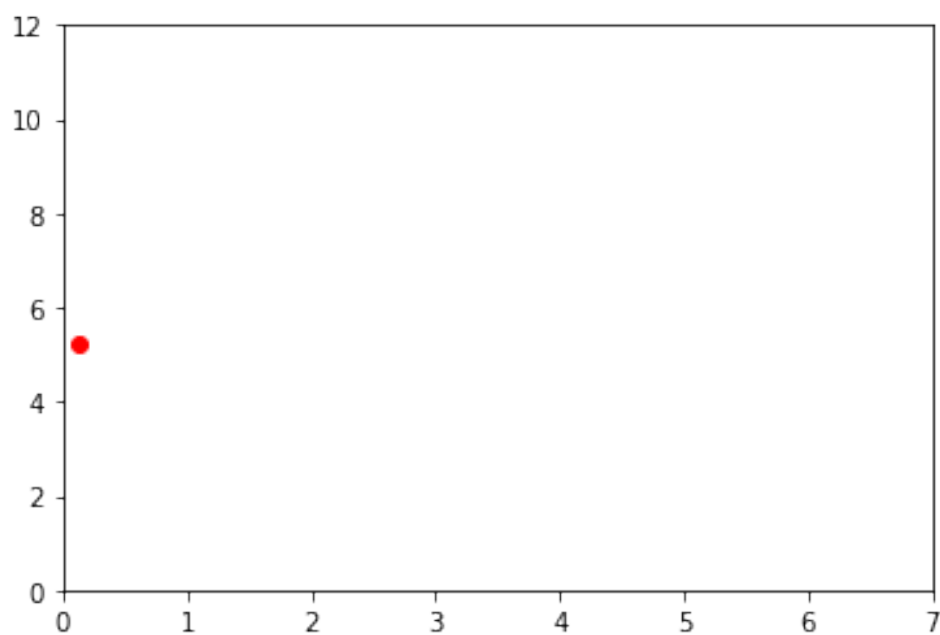
```
[15]: #inicia o engine e coloca as ancoras
P=multilateration.Engine(goal=[None, None, 0.0])
for anchor in edif:
    print(anchor, "x: ", edif[anchor][2], "y: ", edif[anchor][3], "z: ",
    ↳edif[anchor][4])
    #etc=float(edif[anchor][2]) + float(edif[anchor][3])
    #print(etc)
    P.
    ↳add_anchor(anchor,(float(edif[anchor][2]),float(edif[anchor][3]),float(edif[anchor][4])))
```

AC233FC067F5 x: 0.769 y: 2.072 z: 5.235
 AC233FC06A10 x: 5.168 y: 9.144 z: 5.235
 AC233FC06AF7 x: 5.168 y: 2.072 z: 5.235
 AC233FC06B39 x: 0.769 y: 9.144 z: 5.235
 AC233FC06B67 x: 2.977 y: 5.388 z: 5.235

```
[16]: P.add_measure_id("AC233FC067F5", 3.936808193158295)
P.add_measure_id("AC233FC06A10", 7.100517077373254)
P.add_measure_id("AC233FC06AF7", 5.849484886489524)
P.add_measure_id("AC233FC06B39", 4.0422836808817095)
P.add_measure_id("AC233FC06B67", 3.163930359781399)
```

```
[17]: loc=P.solve()

fig = plt.figure()
#ax = plt.axes(projection='3d')
plt.xlim(0, 7)
plt.ylim(0, 12)
#ax.set_zlim(0,7)
plt.scatter(loc.x,loc.y, c='red')#,loc.z, c='red')
plt.show()
print(loc)
```



Point(0.12251365266614192,5.250317969819699,3.5637769887706385)

```
[18]: #para cada colaborador:

        #verifica se nos x(6) do top ordenados por tempo num intervalo de tempo (max_
        ↳até max-30*7) tem pelo menos 3 ancoras
cursor = conn.cursor()
table='colaborador'
var='AC233FA3B9AD'
qres=0
```

```

for res in cursor.execute(f"with q as (SELECT TOP (6) [RSSI], [Local], [Tempo]
↳FROM [andredb].[dbo].[colaborador] where Nome='{var}' and Tempo > (SELECT
↳DATEADD(SECOND, -30, MAX([Tempo])) FROM [andredb].[dbo].[colaborador] ))
↳select count(distinct Local) from q; "):
    qres=res[0]
print(qres)

```

5

```

[19]: print("\n\n")
      #print("-----Read-----\n")
      print(f"-----{table}-----\n")

      cursor = conn.cursor()
      var='AC233FA3B9AD' #emp
      var2='AC233FC06B67' #local
      data=[]
      cursor.execute(
          f"SELECT TOP (6) [ID]"
          f",[Nome]"
          f",[RSSI]"
          f",[Tempo]"
          f",[Local]"
          f"FROM [andredb].[dbo].[colaborador] WHERE Tempo > (SELECT DATEADD(SECOND,
↳-30, MAX([Tempo]))"
          f"FROM [andredb].[dbo].[colaborador] ) and Nome='{var}' and Local =
↳'{var2}' order by Tempo asc"
      )
      for i, row in enumerate(cursor):
          print(f'row {i+1}: {row[2]}')
          data.append(row[2])
      print(data)

```

-----colaborador-----

```

row 1: -62
row 2: -40
row 3: -37
row 4: -38
row 5: -37
row 6: -58
[-62, -40, -37, -38, -37, -58]

```

```
[20]: #test = KalmanFilter(0.008, 0.1)
      #filtereddata=[]
```

```
[21]: #for x in data:
      #     print ("Data:", x)
      #     print ("Filtered Data: ", test.filter(x))
      #     filtereddata.append(test.filter(x))

      #print(filtereddata)
```

```
[22]: #convert=Converter(ref=-32, n=2)
```

```
[23]: #med=convert.media(filtereddata)
      #print(med)
```

```
[24]: #out_m=convert.rssi_to_m(med)
      #print(out_m, "metros")
```

```
[25]: #t.add_measure('anchor_A',sqrt(2))
```

```
[26]: #P.solve()
      #print(t.loc)
```

```
[27]: #print ("Start : %s" % time.ctime(), ": ", datetime.datetime.now())
      #time.sleep( 5 )
      #print ("End : %s" % time.ctime(), ": ", datetime.datetime.now())
```

```
[28]: '''async def setup(client, conn, debug= False):
      #aquí
      return'''
```

```
[28]: 'async def setup(client, conn, debug= False):\n    #aquí\n    return'
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```

[29]: spec = [
    ('R', float32),
    ('Q', float32),
    ('cov', float32),
    ('x', float32),
    ('A', int32),
    ('B', int32),
    ('C', int32),
    ('measurement', float32),
    #('measurement', float32[:]),
]

#@jitclass(spec)
class KalmanFilter:

    #cov = float('nan')
    #x = float('nan')

    def __init__(self, R, Q):
        """
        Constructor

        :param R: Process Noise
        :param Q: Measurement Noise
        """
        self.cov = 0
        self.x = 0

        self.A = 1
        self.B = 0
        self.C = 1

        self.R = R
        self.Q = Q

    #@lru_cache(maxsize = 1000)
    def filter(self, measurement):
        """
        Filters a measurement

        :param measurement: The measurement value to be filtered
        :return: The filtered value
        """
        u = 0
        if self.x==0: #math.isnan(self.x):
            self.x = (1 / self.C) * measurement
            self.cov = (1 / self.C) * self.Q * (1 / self.C)

```

```

else:
    predX = (self.A * self.x) + (self.B * u)
    predCov = ((self.A * self.cov) * self.A) + self.R

    # Kalman Gain
    K = predCov * self.C * (1 / ((self.C * predCov * self.C) + self.Q));

    # Correction
    self.x = predX + K * (measurement - (self.C * predX));
    self.cov = predCov - (K * self.C * predCov);

return self.x

def last_measurement(self):
    """
    Returns the last measurement fed into the filter

    :return: The last measurement fed into the filter
    """
    return self.x

def set_measurement_noise(self, noise):
    """
    Sets measurement noise

    :param noise: The new measurement noise
    """
    self.Q = noise

def set_process_noise(self, noise):
    """
    Sets process noise

    :param noise: The new process noise
    """
    self.R = noise

```

```

[30]: class Converter:

    def __init__(self, ref=-62, n=2):
        """
        Constructor

        :param ref: Process Noise
        :param n: Measurement Noise
        """
        self.ref = ref

```

```

self.n = n

def rssi_to_m(self, measurement):
    """
    Filters a measurement

    :param measurement: The measurement value to be filtered
    :return: The filtered value
    """

    self.measurement_m = 10**((measurement - self.ref)/ (-10 * self.n))

    return self.measurement_m

def m_to_rssi(self, measurement):
    """
    Filters a measurement

    :param measurement: The measurement value to be filtered
    :return: The filtered value
    """

    self.measurement_rssi=measurement

    return self.measurement_rssi

def media(self, number_list = [0, 0, 0, 0, 0, 0, 0]):
    """
    Sets measurement noise

    :param noise: The new measurement noise
    """

    self.avg = mean(number_list)
    #print("The average is ", round(avg,2))
    return self.avg

def set_ref(self, ref):
    """
    Sets measurement noise

    :param noise: The new measurement noise
    """

    self.ref = ref

def set_n(self, n):

```

```

"""
Sets process noise

:param noise: The new process noise
"""
self.n = n

```

```
[ ]:
```

```
[31]: def getMeasurements(conn, emp, local, table='colaborador', flag=0, debug= False):
```

```

    #emp='AC233FA3B9AD'          #emp
    #local='AC233FC06B67'        '#local

    if debug:
        print("\n\n")
        #print("-----Read-----\n")
        print(f"-----{table}-----\n")

    cursor = conn.cursor()

    data=[]
    cursor.execute(
        f"SELECT TOP (6) [ID] "
        f",[Nome] "
        f",[RSSI] "
        f",[Tempo] "
        f",[Local] "
        f"FROM [andredb].[dbo].[{table}] WHERE Tempo > (SELECT DATEADD(SECOND,␣
↪-30, MAX([Tempo]))"
        f"FROM [andredb].[dbo].[{table}] ) and Nome='{emp}' and Local =␣
↪'{local}' order by Tempo asc"

    )

    for i, row in enumerate(cursor):
        if debug:
            print(f'row {i+1}: {row[2]}')
        data.append(row[2])

    return data

```

```
[32]: def seekColab(conn, table='configcolaborador', flag=0, debug= False):
    colab={}
    #table='configcolaborador'
    cursor = conn.cursor()

    cursor.execute(
        f"select * from {table}"
    )
    for i, row in enumerate(cursor):
        colab[row[0].strip()]= row[1].strip()

    if debug:
        print("\n\n")
        print(f"-----{table}-----\n")
        #print(colab)
        for co in colab:
            print("key: ", co,"|value: ",colab[co])

    return colab
```

```
[33]: def seekEdif(conn, table='configedificio', flag=0, debug= False):
    #BUSCA EDIFICIOS
    edifconf={'MAC':0,'LOCAL':1,'X':2,'Y':3,'Z':4}
    edif={}
    edifparam=[]
    #table='configedificio'
    cursor = conn.cursor()

    cursor.execute(
        f"select * from {table}"
    )

    for i, row in enumerate(cursor):
        for c in row:
            edifparam.append(str(c).strip())

        edif[row[edifconf['MAC']].strip()]=edifparam
        edifparam=[]

    if debug:

        print("\n\n")
        print(f"-----{table}-----\n")
        #print(edif)
```

```

        for ed in edif:
            print("key: ", ed,"|value: ",edif[ed])

        print("\n")
    return edif

```

```

[34]: def validator(conn, colaborador, table='colaborador', flag=0, debug= False):
        cursor = conn.cursor()
        #colaborador='AC233FA3B9AD'
        qres=0
        for res in cursor.execute(f"with q as (SELECT TOP (6) [RSSI], [Local],
↵[Tempo] FROM [andredb].[dbo].[{table}] where Nome='{colaborador}' and Tempo >
↵(SELECT DATEADD(SECOND, -30, MAX([Tempo])) FROM [andredb].[dbo].[{table}] ))
↵select count(distinct Local) from q; "):
            qres=res[0]

        return qres

```

```

[35]: def setup(conn, flag=0, debug= False):
        #BUSCA COLABORADORES
        colab = seekColab(conn)

        #BUSCA EDIFICIOS
        edif = seekEdif(conn)

        return colab, edif

```

```

[381]: def localization(conn, flag=0, debug= False):
        import openpyxl
        debug=True
        print ("Start : %s" % time.ctime(), ": ", datetime.datetime.now())
        convert=Converter(ref=-25, n=3.5)
        P=lx.Project(mode='3D',solver='LSE', detail=True)
        while True:

            if flag:
                break

            colab, edif = setup(conn) #P,

            #del edif['AC233FC06B67']
            #del edif['AC233FC06AF7']

            iterater=15

```

```

srcfile = openpyxl.load_workbook('res.xlsx',read_only=False,
↳keep_vba= True)#to open the excel sheet and if it has macros
sheetname = srcfile.get_sheet_by_name('Sheet1')
#sheetname.cell(row=liner-1,column=coluner+auxer2).value = ed
liner=5+(25*iterater)-25
coluner=2 +14 +14 +14
auxer=0
auxer2=0

for colaborador in colab:
    #print("Colaborador: ", colaborador)
    #verifica se nos x(6) do top ordenados por tempo num intervalo
↳de tempo (max até max-30*7) tem pelo menos 3 ancoras
    qres = validator(conn, colaborador)
    #print("validator: ", qres)

    if (qres > 2):

        #inicia o engine #coloca as ancoras
        #P=multilateration.Engine(goal=[None, None, 1])
        t,label=P.add_target(colab[colaborador])

        for anchor in edif:
            #print(anchor)
            P.
↳add_anchor(anchor,(float(edif[anchor][2]),float(edif[anchor][3]),float(edif[anchor][4])))

        for ed in edif:
            print("#####")
            print("Gateway: ", ed)
            sheetname.cell(row=liner-1,column=coluner+auxer2).value
↳= ed

            #busca medidas
            data = getMeasurements(conn, colaborador, ed)

            if data:
                #filtra os dados com filtro de Kalman
                filtereddata=[]
                kalman = KalmanFilter(0.1, 2)#0.008, 0.1)

                auxer=0
                for x in data:
                    if debug:
                        pass
                        print ("Data:", x)

```

```

sheetname.cell(row=liner,column=coluner-1).
↪value = "RSSI Raw"
sheetname.cell(row=liner +
↪auxer,column=coluner+auxer2).value = x
x_f= kalman.filter(x)
sheetname.cell(row=liner+8,column=coluner-1).
↪value = "RSSI Filtered"
sheetname.
↪cell(row=liner+auxer+8,column=coluner+auxer2).value = x_f
print ("Filtered Data: ", x_f)
auxer=auxer+1

filtereddata.append(kalman.filter(x))

sheetname.cell(row=liner+auxer+8,column=coluner-1).
↪value = "Average"
#realiza a media dos resultados
med=convert.media(filtereddata[1:])
if debug:
    print(med)
    sheetname.
↪cell(row=liner+auxer+8,column=coluner+auxer2).value = med
auxer=auxer+1

#converte para metros
out_m=convert.rssi_to_m(med)

if debug:
    sheetname.
↪cell(row=liner+auxer+8,column=coluner-1).value = "Distance (meters)"

print("#####")
print("RESULTADO EM METROS: ", out_m)
sheetname.
↪cell(row=liner+auxer+8,column=coluner+auxer2).value = out_m
auxer=auxer+1

if out_m < 4:
    out_m=out_m
#elif out_m >10:
#    out_m=10
#else:
#    out_m=out_m*1

```

```

        #adiciona a medida
        #P.add_measure_id(ed, out_m)
        t.add_measure(ed, out_m)

        auxer2=auxer2+1

        #quando termina de passar pelos edificios:
        #loc=P.solve()
        P.solve()
        #print(colaborador, ": ", loc)
        for tar in P.TargetDic:
            print(tar, P.TargetDic[tar].loc)
        fig = plt.figure()
        #ax = plt.axes(projection='3d')
        plt.xlim(-3, 10)
        plt.ylim(-3, 15)
        #ax.set_zlim(0,7)

        plt.scatter(0.769, 2.072, c='blue')
        plt.scatter(5.168, 9.144, c='blue')
        plt.scatter(5.168, 2.072, c='blue')
        plt.scatter(0.769, 9.144, c='blue')
        plt.scatter(2.977, 5.388, c='blue')

        plt.scatter(t.loc.x,t.loc.y -1.5, c='red')#,loc.z, c='red')
        plt.show()

        sheetname.cell(row=liner+19,column=coluner).value = "x"
        sheetname.cell(row=liner+19,column=coluner+1).value = "y"
        sheetname.cell(row=liner+20,column=coluner-1).value = "
    ↪"Output Coords"

        sheetname.cell(row=liner+20,column=coluner).value = t.loc.x
        sheetname.cell(row=liner+20,column=coluner+1).value = t.loc.
    ↪y -1.5

        #Caso nao tenha condicoes para fazer localizacao
        else:
            pass

        print ("End : %s" % time.ctime(), ": ", datetime.datetime.now())
        sheetname.cell(row=liner+17,column=coluner-1).value = "Date"
        sheetname.cell(row=liner+17,column=coluner).value = datetime.
    ↪datetime.now()

        srcfile.save('res.xlsx')

```

```
#time.sleep( 30 )
```

```
break
```

```
[ ]:
```

```
[382]: def main():
        flag=0
        try:

            conn = pyodbc.connect(
                "Driver={SQL Server};"
                "Server=192.168.1.2;"
                "UID=;"
                "PWD=;"
                "Database=;"
            )
        except:
            flag=1

        while True:
            #break
            localization(conn, flag, debug=False)
            conn.close()
            break
```

```
[390]: ##run
        if __name__ == "__main__":
            main()
```

Start : Tue May 4 16:38:53 2021 : 2021-05-04 16:38:53.608257

<ipython-input-381-04996f73a957>:24: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).

```
sheetname = srcfile.get_sheet_by_name('Sheet1')
```

#####

Gateway: AC233FC067F5

Data: -52

Filtered Data: -52.0

Data: -54

Filtered Data: -52.71975019516003

Data: -46

Filtered Data: -51.2896616733484

Data: -41

Filtered Data: -48.05739007987912
Data: -41
Filtered Data: -45.39729643054969
Data: -53
Filtered Data: -46.22425664237403
-48.358116995968295

RESULTADO EM METROS: 4.649162983584633

Gateway: AC233FC06A10
Data: -50
Filtered Data: -50.0
Data: -50
Filtered Data: -50.0
Data: -48
Filtered Data: -49.4917387753585
Data: -45
Filtered Data: -48.22930071125815
Data: -46
Filtered Data: -47.22035772955713
Data: -51
Filtered Data: -47.78938588967863
-48.41869080624907

RESULTADO EM METROS: 4.667727042420246

Gateway: AC233FC06AF7
Data: -38
Filtered Data: -38.0
Data: -38
Filtered Data: -38.0
Data: -40
Filtered Data: -38.5082612246415
Data: -38
Filtered Data: -38.667190894939694
Data: -38
Filtered Data: -38.41571120592838
Data: -38
Filtered Data: -38.26320445398464
-38.38431776101479

RESULTADO EM METROS: 2.412176481892643

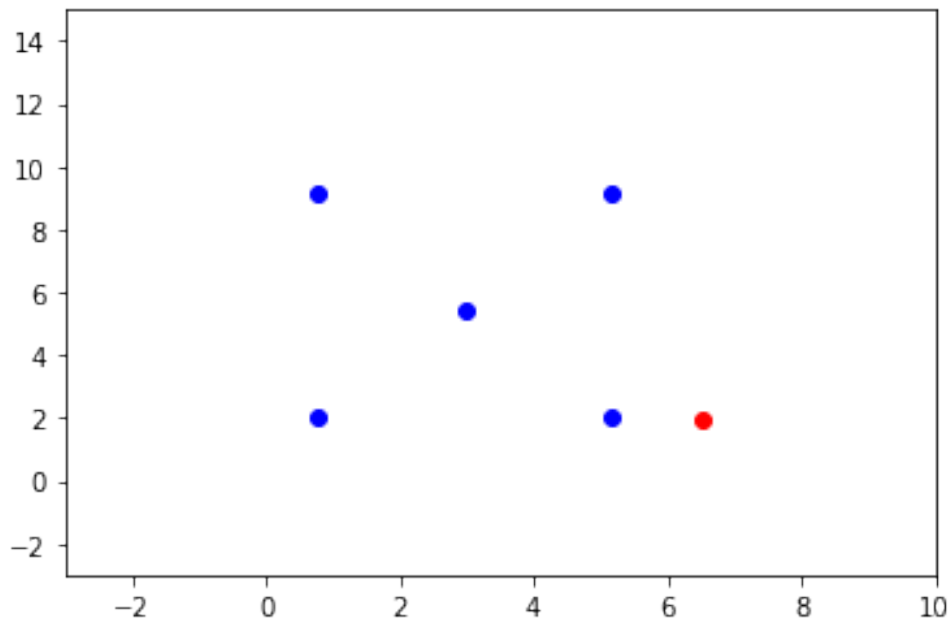
Gateway: AC233FC06B39
Data: -58
Filtered Data: -58.0
Data: -58
Filtered Data: -58.0

Data: -69
Filtered Data: -60.79543673552827
Data: -69
Filtered Data: -64.09726838853308
Data: -49
Filtered Data: -61.780117563323486
Data: -49
Filtered Data: -57.091636254072
-60.092709600133745

RESULTADO EM METROS: 10.061178306126108

Gateway: AC233FC06B67
Data: -43
Filtered Data: -43.0
Data: -45
Filtered Data: -43.71975019516003
Data: -46
Filtered Data: -44.57683718423517
Data: -47
Filtered Data: -45.37027428851423
Data: -47
Filtered Data: -45.98455562569468
Data: -43
Filtered Data: -45.54369857147533
-45.188125935255385

RESULTADO EM METROS: 3.774014793125918
Daniela Barros p(6.516576872606306,3.4244157235939157,5.234999152213878)



End : Tue May 4 16:38:54 2021 : 2021-05-04 16:38:54.840267

```
[371]: ##run
        #logging.basicConfig(level=logging.INFO)
        #asyncio.run(main())

[ ]: #print(edif['AC233FC06AF7'][1])

[ ]: import pandas as pd

[ ]:

[ ]: '''import openpyxl

srcfile = openpyxl.load_workbook('docname.xlsx',read_only=False, keep_vba=
↳True)#to open the excel sheet and if it has macros

sheetname = srcfile.get_sheet_by_name('sheetname')#get sheetname from the file
sheetname['B2']= str('write something') #write something in B2 cell of the
↳supplied sheet
sheetname.cell(row=1,column=1).value = "something" #write to row 1,col 1
↳explicitly, this type of writing is useful to write something in loops

srcfile.save('newfile.xlsm')'''
```

```
[ ]: #import openpyxl

[ ]: #srcfile = openpyxl.load_workbook('res.xlsx',read_only=False, keep_vba= True)#to
    ↪open the excel sheet and if it has macros

[ ]: #sheetname = srcfile.get_sheet_by_name('Sheet1')

[ ]: #sheetname.cell(row=1,column=1).value = "something"

[ ]: #srcfile.save('res.xlsx')

[ ]:
```

Appendix D

Cramer-rao Bound Matlab Code

```
1 % this code simulates the CRLB of RSS measurements
2 clc;
3 clear all;
4 %*****
   PARAMETERS *****
5 d0=1;% reference distance (indoor scenario) (meter)
6 PLE=4; % path loss exponent
7 target_x=4.0; %true location (meter)
8 target_y=5.0; %true location (meter)
9 s_x=[ 0.769 5.168 5.168 0.769 2.977];% x coordinate of anchor sensors ...
   (meter)
10 s_y=[ 2.072 9.144 2.072 9.144 5.388];% y coordinate of anchor sensors ...
   (meter)
11 L0=[40];% reference path loss (dB)
12 p0=[-27];% reference received power (dBm)
13 %*****
   standard deviation range [0, 1, ..., 5] ...
   *****
14 sigma=[0.0001:1:5.1]; % noise standard deviation (dB)
15 for w=1:length(sigma)
16 %%%%%%%%%%%%% CRLB BEGIN
17 for k3=1:length(s_x)
18 drivative_mu_x1(k3)=(-10*PLE/log(10)) * ( ([ target_x ]-s_x(k3)) ...
   )/ ...
19 ( (norm([target_x; target_y ]-[s_x(k3);s_y(k3)],2) )**2 );
20 end
21 drivative_mu_xx1=drivative_mu_x1;
22 for k2=1:length(s_x)
23 drivative_mu_y2(k2)=(-10*PLE/log(10))*( ([ target_y ]-s_y(k2)) )...
24 /( (norm([target_x; target_y ]-[s_x(k2);s_y(k2)],2) )**2 );
25 end
26 drivative_mu_yy2=drivative_mu_y2;
27 gama=[ eye(length(s_x))];
```

```

28 c1=[(sigma(w))**2];
29 sigma1=c1*gama;
30 inv_sig=inv(sigma1);
31 J1(1,1)=drivative_mu_xx1'*inv_sig*drivative_mu_xx1';
32 J1(2,2)=drivative_mu_yy2'*inv_sig*drivative_mu_yy2';
33 J1(1,2)=drivative_mu_xx1'*inv_sig*drivative_mu_yy2';
34 J1(2,1)=J1(1,2);
35 J1_inv=inv(J1);
36 CRLB(w)=sqrt(J1_inv(1,1)+J1_inv(2,2));
37 %%%%%%%%%%%%% CRLB  END
38 end
39 %*****plot ...
    %*****
40 pic1=plot(sigma/2,CRLB,'r','LineWidth',3);
41 hold on
42 grid on;
43 title('Figure1 ')
44 xlabel('Standard deviation (dBm)');
45 ylabel('MSE (meter)');
46 leg1 = legend({'CRLB', 'LSE Multilateration'},'Location','northwest');

```