POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Design and Implementation of a Radio-based Indoor Localization System for Cyber Physical Systems focused on Industry 4.0 applications

Supervisors

Prof. ALESSANDRO RIZZO

Dr. STEFANO PRIMATESTA

Ing. ORLANDO TOVAR ORDOÑEZ

Candidate

EMANUELE ERRICO

DECEMBER 2021

Summary

Nowadays, technological progress has led to significant developments in the field of automation. This promoted the increase of plant productivity, the improvement of working conditions and product quality. One of the main factors of this type of industry is the localization system allowing machines and robots to move autonomously in a given work environment. The aim of this thesis is to develop and implement a localization system for a drone performing autonomous indoor navigation. To achieve this goal it is essential to use a well-known localization system, the Ultra-WideBand (UWB) technology, which has good accuracy and gives good stability to the navigation. Firstly, research has been made on the state of the art of Ultra-WideBand technology with its advantages over other localization systems. Furthermore, studies have been done on the standard using UWB technology and its progress over the years. Note that during the localization process, the positioning outputs are infected with some errors that can be bias errors or outliers, i.e., errors related to radio connectivity. For this reason, it is essential to realize a tool that improves the accuracy and stability of the whole system. Thus, after some analysis, a filter was carried out on Matlab simulation program and then translated into the Python language in order to optimize the system. The filtered systems have been simulated to check the characterization of the technology and possible problems related to it. Once good results were achieved in the simulation, a suitable configuration was built in a flight cage and some real-world tests performed to determine the successful implementation of the design localization system.

Acknowledgements

I would like to express my gratitude to my supervisors Prof. Rizzo, Dr. Primatesta and ing. Tovar and to CIM4.0 for all their help and advice with this thesis.
I wish to thank all my friends and those I have met on this university journey.

To my family that has always been close to me.
To my parents, the example of love, their desires put aside to satisfy mine.
To Maria and Marco, my happiness.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Objective of the thesis

In the last decade, there were great technological advances in industries, especially in the field of automation. Industries use many robots to make production faster and of higher quality. Indeed, these robots can be automatically controlled, reprogrammable, and serve multiple purposes. Self-driving drones are also part of this advancement and play an important role in Industry 4.0.

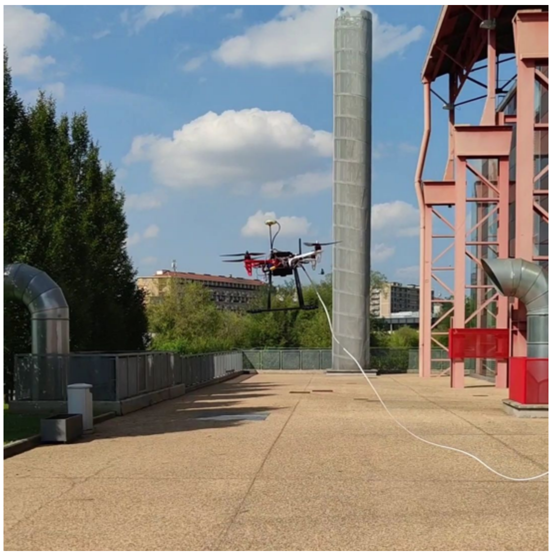


Figure 1.1: Drone flying in the outdoor environment

Generally, they are used to control what is present in an area, transport small objects, check harmful or unreachable environments for humans. To navigate in a certain environment they need a localization system. Usually, drones fly using GPS in the outdoor environment, but the GPS signal cannot determine the location in the indoor one.

Many technologies are on the market for indoor location systems. These technologies have different features and different advantages. Bluetooth and Wi-fi systems are used, but Ultra-WideBand plays an important role in this field. It is a radio technology that transmits the signal at a high frequency and has an accuracy of about 20 centimeters.

Figure 1.2: Drone flying in the indoor environment

The goal of the thesis is to create a UWB network that allows a self-guided drone to navigate an indoor environment with good accuracy and stability. To do this, a series of studies have been done on the various localization systems. In particular, it has been investigated in the state of the art of UWB technology and how the localization is calculated.
A fundamental aspect is the position of the anchor nodes because the estimate of the system accuracy depends on the anchor configuration.

Later, studies have been done to filter, these measurements decrease the disturbances and have better results. The Extended Kalman filter is the best tool to obtain optimal results. Furthermore, there may be also outlier errors that are due to loss of connectivity between anchor nodes and tag nodes, because of the distance or presence of obstacles. So, the filter has been further modified to improve outlier detection.

The code has been first written and simultaneously simulated on Matlab to make sure that the system is as efficient as possible. The position of the drone during the

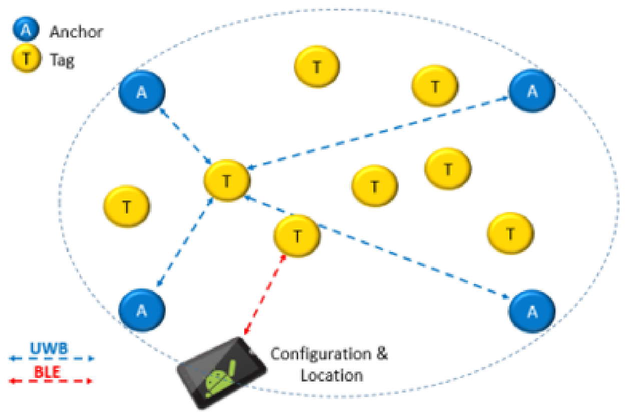


Figure 1.3: Ultra-WideBand system configuration: 4 Anchors, 8 Tags [1]

flight is determined by creating a filter, implemented and tested on Python. Finally, other simulations have been done on Gazebo, which is a simulation environment of robots.

The UWB system used to determine the position of the drone has been tested in simulation environment, where good results have been obtained for trajectories, and in real file.

1.2 Structure of the thesis

The thesis is structured as follows:

- Chapter 2: The development of this thesis begins with a state of the art analysis. Different studies on location systems and filters to decrease the error in the location calculation phase have been compared.
- Chapter 3: It describes some useful simulations on Matlab to analyze the performance of the filter and correct any errors. Finally, after implementing the filter code on Python some flight simulations have been done on Gazebo.
- Chapter 4: It describes and analyze some tests done in real life. The position of a node tag has been computed at some fixed points and following a trajectory in both closed and open environments.
- Chapter 5: In the last chapter there are conclusion on the thesis work, a description of the most important processes and some possible future improvements for the system .

Chapter 2

State of the art

2.1 Localization systems

The thesis is based on the navigation of a drone and it needs a localization system to move in a certain environment. There are systems that allow to locate and track a target device, and they have different features and operations to achieve the same goal.

2.1.1 GPS

To locate a device it is very common to use GPS which is present in many smartphones, PCs and mobile devices.
The Global Positioning System (GPS) is a satellite-based navigation system and it provides geolocation in three dimensions and time information using satellites and a receiver where there are no obstacles such as mountains and buildings [2].
It is operated by the United States Air Force under the direction of the Department of Defense (DoD) and is owned by the United States government.
GPS is divided into three segments:

- Control Segment, global monitor stations track the GPS satellites which relay satellite range and timing measurement data to the Master Control Station to replace orbit and clock data of each space vehicle every hour.
- Space Segment, GPS constellation is composed of 24 space vehicles orbiting the earth at altitudes of 20.200 km. They circle in six orbital planes inclined at an angle of 55 degrees, covering any point on the Earth with at least five space vehicles.
- User Segment, originally conceived of as a military system, consists of GPS receivers and transmitters including devices like smartphones, PCs, and mobile

devices [2].

Figure 2.1: GPS Satellite [3]

Location, velocity and elevation are computed through trilateration: in order to be read and interpreted by a GPS device capable of reading the signal of at least four satellites, each satellite in the network circles the Earth twice a day. In this way, the satellite transmits a unique signal, the orbital parameters and the time.
A single satellite sends a microwave signal which is picked up by a GPS device and used to calculate the distance from the GPS device to the satellite. The satellite cannot provide much location information because a GPS device only gives information about the distance from a satellite. Satellites do not provide information about angles, so the location of a GPS device could be anywhere on a sphere's surface area with a radius equal to the distance measured from the GPS device to the satellite [4].

With three satellites, the device's location can be determined, as the device is at the intersection of all three spheres that produces two points of intersection, so the point nearest Earth is chosen.
With every movement of the device, the distance to the satellite changes, and new spheres are produced, giving a new position. It is possible to use that data, combined with the time from the satellite, to determine velocity, calculate the distance to the destination [4].

2.1.2 Wi-Fi

Wi-Fi is a wireless networking technology that allows devices such as computers and mobile devices the Internet connection, so this makes a network and devices can exchange information with one another.

Figure 2.2: Wi-fi alliance logo [5]

The IEEE 802.11 standard defines the protocols that enable communications with current Wi-Fi-enabled wireless devices, including wireless routers and wireless access points that support different IEEE standards.
The standards, ratified over time, operate on variable frequencies, provide different bandwidths, and support a different number of channels [6]. Wi-Fi technology is also important for positioning systems, a geolocation system that discovers the localization of a device from the characteristics of nearby Wi-Fi hotspots and other wireless access points [7].

Wi-Fi technology is widely deployed in indoor environments and the device positioning can be computed based on the number of Wi-Fi access points: in presence of only one Wi-Fi access point, the target's positioning is located through a hybrid AoA/ToA system; when there are two or more Wi-Fi access points, AoA system is used to obtain higher accuracy location [8].

The goal of Wi-Fi-based indoor localization of a device is the position computation of the device with respect to access points. To reach this goal many techniques exist and they may be classified into four main types: received signal strength indication (RSSI), fingerprinting, angle of arrival (AoA), and time of flight (ToF) based techniques [9].

In most cases the first step to determine a device's position is to determine the ranging, that is the distance measured between the target client device and a few access points. Knowing the ranging and the position of access points, trilateration algorithms may be used to determine the relative position of the target device [10]. Alternatively, the angle of arriving signals at a target client device can be employed to determine the device's location based on triangulation algorithms [9].

Another solution can be the combination of these techniques to increase the accuracy of the system [9].

2.1.3 Bluetooth

Bluetooth is a short-range wireless technology for low-cost and low-bandwidth communication scenarios connecting devices such as mobile phones, headsets, and portable computers without any cables. Bluetooth devices can also be used to connect multiple devices into Personal Area Networks (PANs). Bluetooth is constantly evolving and the total versions of Bluetooth standards developed are: v1.1, 1.2, 2.0, 2.1, 3.0, 4.0, 4.1, 4.2, 5, 5.1, 5.2, 5.3. Bluetooth is widely used in everyday life: it is present in most mobile phones, and also in printers, computers, cameras, smartwatches, etc.

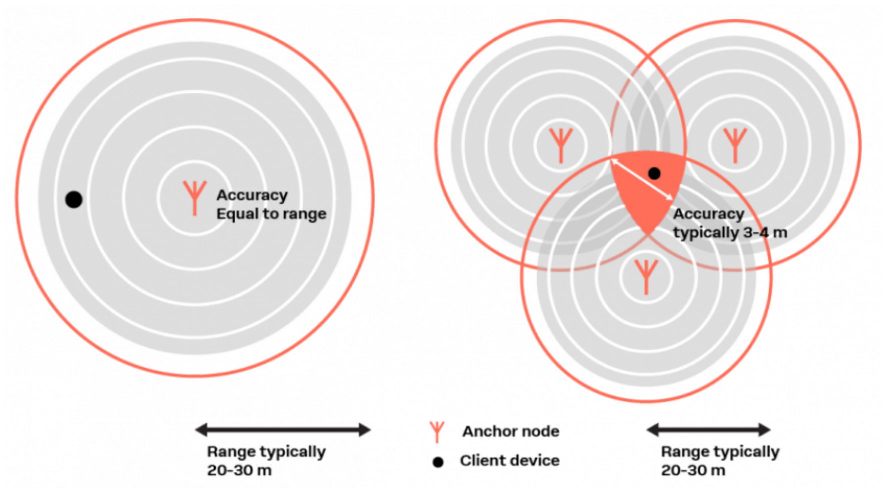


Figure 2.3: Bluetooth indoor positioning [11]

In some scenarios where positioning accuracy and response time are not critical, but device operation time is more important, Bluetooth positioning is more preferable technology than Wi-Fi, for example, city and tourism guild programs for smartphones, daily people, and logistic tracking in hospitals, companies.
However, there is much fewer Bluetooth-based positioning system in comparison to Wi-Fi because there are several technical hurdles to overcome for Bluetooth-based positioning systems.

- Bluetooth communication is low-cost and its standard does not provide precise time synchronization, so it is necessary time-based triangulation methods such as TOA, TODA.

- Bluetooth technology is present in low footprint devices but directional or array antennas are rarely used, so it is difficult to use angle measurement methods like AOA.

- RSSI approach is not designed for Bluetooth standards and it is unreliable and device-dependent. Furthermore, two devices are needed to be connected to measure the RSSI values, but this may take dozens of seconds [12].

2.1.4 UWB

Ultra-Wideband is a radio technology that can use a very low energy level for short-range, high-bandwidth communications over a large portion of the radio spectrum.
It is different from Bluetooth and Wi-Fi technology because it works at very high frequency. We can imagine it as a radar capable of continuously scanning an entire room and precisely identifying an object to discover its position and to communicate any data [13].

UWB technology was developed in the early 2000s, with limited use of military radar and covert communications; and was used briefly as a medical imaging tool, in systems such as remote cardiac monitoring.

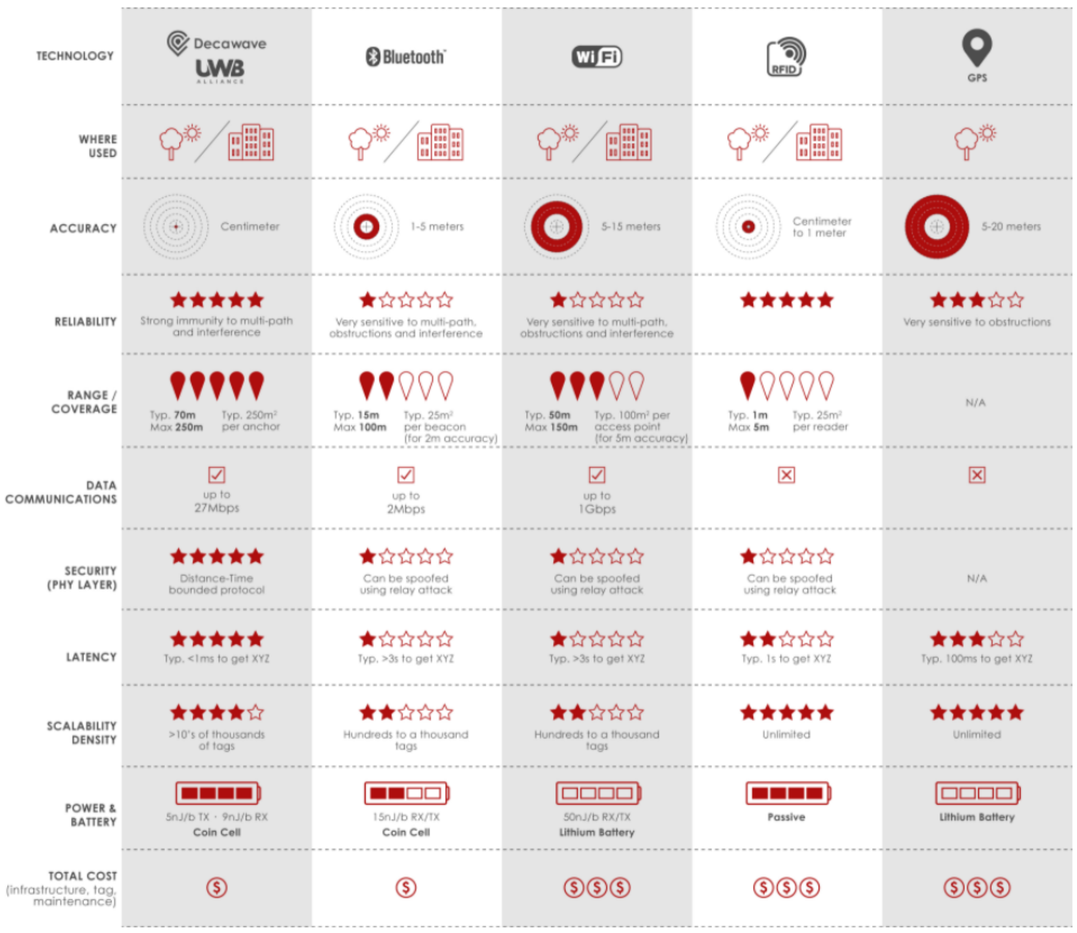
Actually it is used for tracking the animals for biological research, or the monitoring and control of machinery without any cables in a warehouse, or as in this experience to localize a drone in an indoor environment.

While both Wi-Fi and Bluetooth have been tweaked to allow for greater accuracy in locating other devices and connecting to them, UWB is natively more accurate, uses less power. Furthermore, UWB chip production is increasing and the price becomes attractive.

It is significant that companies like Samsung, Apple and Huawei are committing to this technology, in fact they are all involved in UWB projects, including chip and antenna manufacturing [14].

Position technique

To locate a node in a wireless system, it is necessary to compute the position measurements of the radio signals traveling between the target node and the reference. Depending on the positioning technique, the angle of arrival (AOA), the signal strength (SS), or time delay, information is important to determine the location of a node. The AOA technique measures the angles between a given node and the reference nodes to estimate the location, whereas the SS and time-based approaches estimate the distance between nodes by measuring the energy and the travel time of the received signal, respectively [14].



TECHNOLOGY	Decawave UWB ALLIANCE	Bluetooth	WiFi	RFID	GPS
WHERE USED	🌳 / 🏢	🌳 / 🏢	🌳 / 🏢	🌳 / 🏢	🌳
ACCURACY	Centimeter	1-5 meters	5-15 meters	Centimeter to 1 meter	5-20 meters
RELIABILITY	★★★★★ Strong immunity to multi-path and interference	★☆☆☆☆ Very sensitive to multi-path, obstructions and interference	★☆☆☆☆ Very sensitive to multi-path, obstructions and interference	★★★★★	★★★☆☆ Very sensitive to obstructions
RANGE / COVERAGE	Typ. **70m** Max **250m**; Typ. 250m^2 per anchor	Typ. **15m** Max **100m**; Typ. 25m^2 per beacon (for 2m accuracy)	Typ. **50m** Max **150m**; Typ. 100m^2 per access point (for 5m accuracy)	Typ. **1m** Max **5m**; Typ. 25m^2 per reader	N/A
DATA COMMUNICATIONS	✓ up to 27Mbps	✓ up to 2Mbps	✓ up to 1Gbps	✗	✗
SECURITY (PHY LAYER)	★★★★★ Distance-Time bounded protocol	★☆☆☆☆ Can be spoofed using relay attack	★☆☆☆☆ Can be spoofed using relay attack	★☆☆☆☆ Can be spoofed using relay attack	N/A
LATENCY	★★★★★ Typ. <1ms to get XYZ	★☆☆☆☆ Typ. >3s to get XYZ	★☆☆☆☆ Typ. >3s to get XYZ	★★☆☆☆ Typ. 1s to get XYZ	★★★☆☆ Typ. 100ms to get XYZ
SCALABILITY DENSITY	★★★★☆ >10's of thousands of tags	★★☆☆☆ Hundreds to a thousand tags	★★☆☆☆ Hundreds to a thousand tags	★★★★★ Unlimited	★★★★★ Unlimited
POWER & BATTERY	■■■■ 5nJ/b TX · 9nJ/b RX **Coin Cell**	■■□□ 15nJ/b RX/TX **Coin Cell**	□□□□ 50nJ/b RX/TX **Lithium Battery**	■■■■ **Passive**	□□□□ **Lithium Battery**
TOTAL COST (infrastructure, tag, maintenance)	$	$	$$$	$$$	$$$

Figure 2.4: Comparison diagram of location systems [13]

- **AOA**, The AOA is based on measuring angles of the target node seen by the anchor nodes using antennas. This approach is not suited to UWB positioning because the use of antenna arrays increases the system cost that is the main advantage of UWB technology. Furthermore, since the bandwidth of a UWB signal are large, the number of paths may be very large, so the accurate angle estimations become very challenging due to scattering from objects in the environment.

- **SS**, the distance between two nodes can be calculated by measuring the voltage of the received signal at one node. This is the received signal strength approach and there are necessary at least three nodes for the triangulation of the signal. The very large bandwidth is the UWB characteristic and this

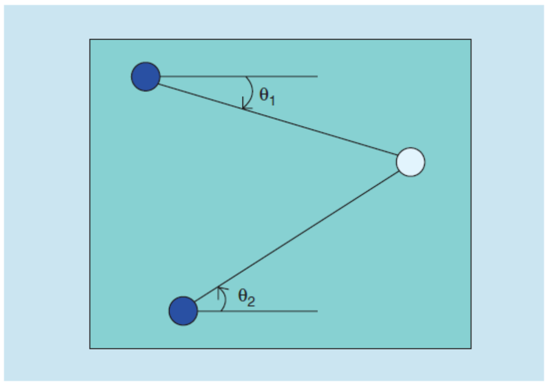


Figure 2.5: Positioning via the AOA measurement. The blue nodes are the reference nodes.[14]

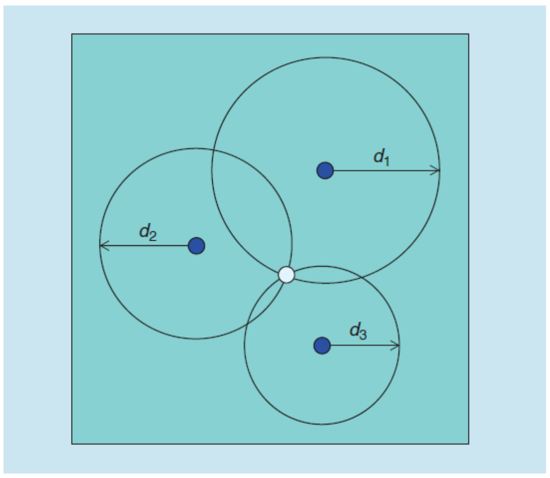


Figure 2.6: Distance-based positioning technique. The distances can be obtained via the SS or the TOA estimation. The blue nodes are the reference nodes.[14]

does not make improvements in LSS approach. A better solution can be a conjunction of SS positioning algorithms with time delay measurement of other reference nodes in a hybrid scheme.

- **Time-Based Approaches**, if two nodes have a common clock the node that receives the signal can determine the time of arrival (TOA) of the incoming signal that is time-stamped by the reference node. In this case, the accuracy of a time-based approach can be improved by increasing the effective signal bandwidth, unlike the SS-based technique. Furthermore, it is important the clock synchronization between a given node and the anchor nodes because the achievable accuracy under ideal conditions is very high. [14]

For positioning systems employing UWB radios, time-based schemes provide a very good accuracy due to the large bandwidth of UWB signals.
Moreover, they are cheaper than the AOA approach which has a strong scattering due to typical UWB signals. Although it is easier to estimate SS than TOA, the range information obtained from SS measurements is very coarse compared to that obtained from the TOA measurements. [14]

2.2 Standard IEEE 802.15.4

2.2.1 IEEE 802.15.4a

The original 802.15.4 standard is released in 2003 and adopted a wideband physical layer using a Direct Sequence Spread Spectrum technique (DSSS). The standard provided specifications for operating in three different frequency bands divided into channels. The channelization scheme was the following:

- 1 channel (Channel 0) was defined in the 868 MHz band;
- 10 channels (Channels 1 – 10) were defined in the 915 MHz band, with a channel spacing of 2 MHz;
- 16 channels (Channels 11 – 26) were defined in the 2.4 GHz band, with a channel spacing of 5 MHz.

The first channels were intended for very low bit rate operations and with user rates of 20-40 kb/s; the last ones for bit rates up to 250 kb/s per channel, thanks to the larger bandwidth allocate to each channel.
Afterward, it was evident that the range of potential applications for a low bit rate standard could be significantly increased by the capability of measuring distance between devices in the network with high accuracy. Since this capability was precluded to 802.15.4 devices due to the limited signal bandwidth, the 802.15.4a Task Group was created to define a new physical layer, able to provide the desired ranging capability, and correspondingly adapting the Medium Access Control layer.
The UWB signal adopted in 802.15.4a uses three frequency bands within the range of frequencies made available by regulation for UWB emissions released in 2002 by the Federal Communications Commission (FCC): a sub-GHz band, a low band in

Channel	Center frequency (MHz)	Bandwidth (MHz)
0	499.2	499.2
1	3494.4	499.2
2	3993.6	499.2
3	4492.8	499.2
4	3993.6	1331.2
5	6489.6	499.2
6	6988.8	499.2
7	6489.6	1081.6
8	7488.0	499.2
9	7987.2	499.2
10	8486.4	499.2
11	7987.2	499.2
12	8985.6	499.2
13	9484.8	499.2
14	9984.0	499.2
15	9484.8	1355

Table 2.1: Channelization scheme in IEEE 802.15.4a[15]

the 3–5 GHz range, and a high band in the 6–10 GHz range. As in the original physical layer, the bands are divided into channels, as presented in Table 2.1 [15].

The standard channels, characterized by a bandwidth of about 500 MHz, provide ranging with an accuracy in the order of 1 meter. To have higher ranging accuracy it may use channels 4, 7 and 15, which offer a larger bandwidth. Furthermore, it should be noted that up to two 802.15.4.a networks can operate in the same channel at the same time, thanks to the adoption of preambles, characterized by low cross-correlation. The preambles are defined within the standard.
The UWB physical layer uses an Impulse Radio approach, in which short pulses with a bandwidth matching the channel bandwidth are transmitted. Moreover, the UWB physical layer can achieve bit rates varying approximately between 0.1 Mb/s and 26 Mb/s[15].

Network Devices and Topologies

The 802.15.4/4a standards define two classes of devices: Full-Function Devices (FFD), where all network functionalities are implemented, and Reduced-Function Devices (RFD), that only support a reduced set of functionalities and are thus typically sensor nodes that measure a physical parameter and can execute simple commands.

RFD and FFD devices organize themselves in Personal Area Networks (PANs). A PAN can adopt either of the two following network topologies:

- **star topology**, where the devices can only exchange information with the PAN coordinator; better topology suited for network architectures where a device is connected to the power network.

- **peer-to-peer topology**, where FFD devices can communicate directly as long as they are within physical reach, while RFD devices, due to their limitations, can only connect with the PAN coordinator.

The peer-to-peer topology has its higher flexibility, so potentially can make more complex topologies, for example based on multiple clusters; algorithms for the creation and management of such larger network topologies are however not part of the standards [15].

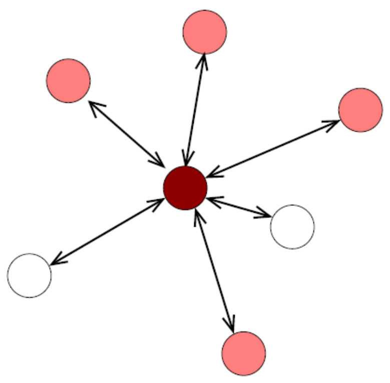


Figure 2.7: Example of star topology (Dark grey circle: PAN coordinator; Light grey circle: FFD device; White circle: RFD device) [15]

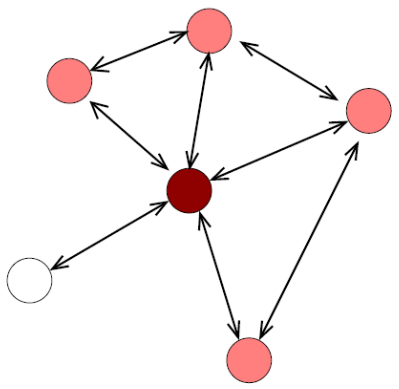


Figure 2.8: Example of peer-to-peer topology (Dark grey circle: PAN coordinator; Light grey circle: FFD device; White circle: RFD device) [15]

Ranging Support

One of the key innovations introduced in the 802.15.4a revision is the support for ranging. The need for information on the distance between network devices was indeed one of the main reasons for the definition of the 802.15.4a standard. It should be noted however that support for ranging in 802.15.4a-compliant devices will be optional.
To support the broadest possible set of network topologies, the procedure to obtain

the distance between two devices in the network is based on a two-way ranging approach, allowing for distance estimation without the need for a common time reference.

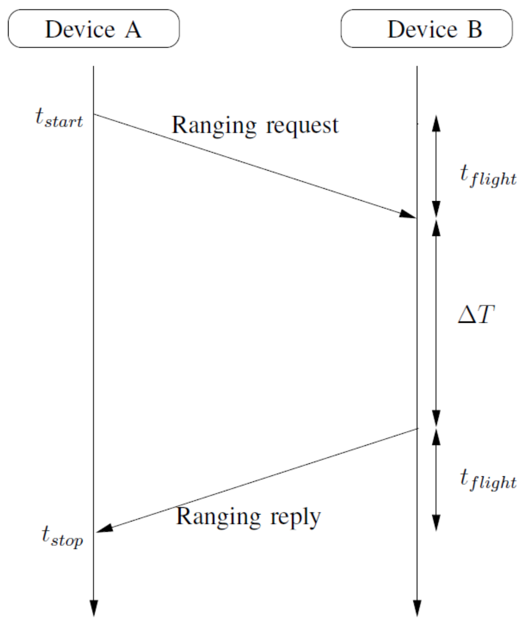


Figure 2.9: Example of two-way ranging [15]

The two-way ranging approach requires the two devices to exchange at least two packets, following the scheme represented in Figure 2.9. In this scheme, device A starts a ranging measurement by sending a ranging packet to device B at time t_{start} and recording a timestamp. Device B records a timestamp when it receives the packet from A, and replies with a second ranging packet, transmitted after a delay ΔT when a second timestamp is recorded by B. The packet is received by device A at time t_{stop}, when a new timestamp is recorded. The knowledge of the time interval t_{stop} - t_{start} and the delay ΔT, obtained from the four timestamps recorded by devices A and B, allows to determine the propagation time t_{flight}, given by the equation[15]:

$$t_{flight} = \frac{t_{stop} - t_{start} - \Delta T}{2} \tag{2.1}$$

2.2.2 IEEE 802.15.4z

There are some innovations in the standard 4z that are the enhancements in the Radio, new ranging features like the simultaneous ranging and a better security in the physical layer.

Enhancements in the Radio

The new standard tries to increase the limited range of the radio due to very strict transmission power requirements; the main website of the EIR 4z Task Group states that the typical range of the radio is up to 100 meters.
Typically, the distance between the Anchors and between the Anchors and Tags is no more than a few tens of meters due to the extremely low Tx power of the radio. The improved timestamp formats could help improve the robustness of the messages and thus improve the overall range that could be achieved.
With the introduction of the new enhancements, it can be a better location accuracy: the general First Path detection should improve helping with the reliability of the measurement and the resulting accuracy as well.
Reducing on-air times is also important: the introduction of PRFs of 128 MHz for 6.81 Mbit/s data rate and a PRF of 256 MHz for the 27 Mbit/s data rate should help immensely in comparison with the maximum 64 MHz PRF, the actual maximum value. These revisions and proposals could also increase the distances significantly.
To sum up, introducing higher PRFs, compressing the radio messages, and increasing their robustness will dramatically increase the battery lifetime, which is a crucial factor for the mobile Tags in any UWB based system [16].

New Ranging Features

The introduction of SS - TWR would make the ToF estimates more simple when compared to the DS - TWR which is actually implemented today. If the new generation of chips could handle this issue with the help of the MAC changes in the new standard, this ranging technique could be more easily implemented.
The introduction of the new Simultaneous ranging capability is also interesting for several use cases for example the wireless car key access: the receiving key-fob would be able to simultaneously handle incoming responses from several Anchors, which will dramatically increase battery lifetime. Generally, putting the UWB radio in a receiving state has a big impact on the battery so any reduction of this is a big advantage [16].

Security

Security was not a big topic in the previous UWB standards, but UWB technology is finding more and more applications and also becoming more present everywhere at the same time, so the security of the technology has become an important topic. That is why the new standard implements several ways to increase the security of wireless transactions. One of them is the introduction of the new ciphered messages between communicating devices, which require a form of authentication key to prevent any malicious form of communication. The overall security is also increased thanks to the reduced on-air time of the radio and shorter messages.
A big emphasis is also on providing the highest security possible for the UWB LRP PHY, which is newly intended for ranging applications as well enabling secure access to vehicles for example. The LRP PHY shall also utilize a form of authentication, either a one-way or a mutual one [16].

2.3 FILTER

The positioning outputs from the UWB devices have good accuracy and stable behavior but they contain some inaccuracies, so it is needed a tool that reduces the noises in the measurement.

2.3.1 Linear Least Square

The least-squares approach occurs in regression analysis and is able to approximate those systems that have more equations than unknowns by minimizing the sum of squares of the residuals, which are the differences between an observed value and the fitted value provided by a model, made in the results of each equation.

The least-squares approach is divided into the linear least squares (L-LS) approach, which has a closed-form solution, and the nonlinear least squares (N-LS) approach, which is usually solved by iterative filtering, depending on the linearity of the residuals.

The L-LS approach is a suboptimal placement technique, which can be used for applications that require a low cost implementation with reasonable placement accuracy. In addition, for applications that require precise position estimation, L-LS approaches can be used to obtain an initial position estimate to initialize high-precision positioning algorithms: good initialization can significantly decrease the computational complexity and final localization error of a high-precision technique [17].

System Model

A wireless network with N reference nodes, with the i^{th} node being located at $l_i = [x_i \quad y_i]^T$ for i=1,…,N. The aim is to estimate the position of the target node, denoted by $l = [x \quad y]^T$, based on N TOA measurements between the target node and the reference nodes:

$$z_i = f_i(x, y) + n_i, \quad i = 1, \ldots, N \tag{2.2}$$

where n_i is the noise in the i^{th} measurement, and $f_i(x, y)$ is the true distance between the target node and the i^{th} reference node, given by

$$f_i(x, y) = \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{2.3}$$

Typically, the noise is modeled by a zero-mean Gaussian random variable, when the reference node and the target node have direct line-of-sight (LOS). In contrast, under non-line-of-sight (NLOS) conditions, the noise distribution can be quite different from a Gaussian distribution.

In the absence of noise in the system, each TOA measurement specifies a circle for the possible positions of the target node, and the intersection of those circles determines the target position. This geometric technique called, trilateration, yields ambiguous solutions in the presence of noise in the system, since the circles defined by the TOA measurements may intersect at multiple points due erroneous TOA estimation, as shown in Fig. 2.10.

Due to the limitations of the geometric technique, statistical approaches are commonly employed in wireless positioning systems. A popular statistical positioning algorithm is the N-LS technique.

An alternative approach to the N-LS estimation is the L-LS approach. In an L-LS technique, a new measurement set is obtained from the measurements in (2.2) by certain operations that result in linear relations.

The L-LS approach starts with the following set of equations

$$z^2 = (x - x_i)^2 + (y - y_i)^2, fori = 1, ..., N, \tag{2.4}$$

where each distance measurement is assumed to define a circle of uncertain region. Then, one of the equations in (2.4), the rth one, is fixed and subtracted from all of the other equations. After some manipulation, the following linear relation can be obtained:

$$Al = p \tag{2.5}$$

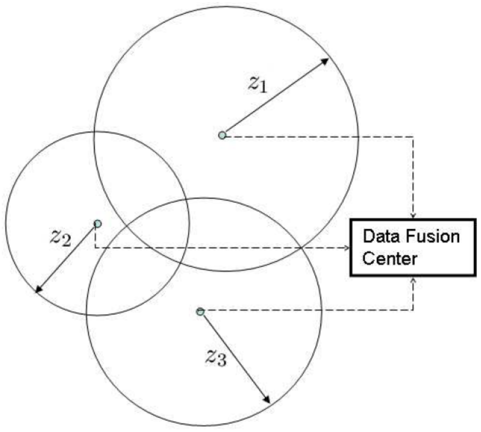


Figure 2.10: Trilateration yields multiple intersection of circles defined by TOA measurements in the presence of noise [17]

where $l = [x \quad y]^T$,

$$A = 2 \begin{bmatrix} x_1 - x_r & y_1 - y_r \\ \vdots & \vdots \\ x_{r-1} - x_r & y_{r-1} - y_r \\ x_{r+1} - x_r & y_{r+1} - y_r \\ \vdots & \vdots \\ x_N - x_r & y_N - y_r \end{bmatrix} \tag{2.6}$$

and

$$p = \begin{bmatrix} z_r^2 - z_1^2 - k_r + k_1 \\ \vdots \\ z_r^2 - z_{r-1}^2 - k_r + k_{r-1} \\ z_r^2 - z_{r+1}^2 - k_r + k_{r+1} \\ \vdots \\ z_r^2 - z_N^2 - k_r + k_N \end{bmatrix} \tag{2.7}$$

with

$$k_i = x_i^2 + y_i^2 \tag{2.8}$$

for i = 1, 2, . . . , N, and r being the selected reference node index that is used to obtain linear relations. Note that A is an $(N - 1)2$ matrix, and p is a vector of size $(N - 1)$, since the r_{th} measurement is used as a reference for the other measurements.

From (2.5), the LS solution can be obtained as

$$\hat{l} = (A^T A)^{-1} A^T p \tag{2.9}$$

This estimator is called the linear LS (L-LS) estimator. Compared to the N-LS estimator, it has low computational complexity. However, it is suboptimal in general, and the amount of its suboptimality can be quantified in terms the CRLB [17].

2.3.2 Kalman Filter

The Kalman filter plays a good role in this field: it is an algorithm that produces estimates of unknown variables based on a series of measurements observed over time, including statistical noise and other inaccuracies, by estimating a joint probability distribution over the variables for each timeframe. It is more accurate than estimates that only work on a single measurement. The filter is called for one of the main developers of this theory, Rudolf E. Kálmán [18].

Kalman filtering has numerous technological applications, for example, guidance, navigation, and control of vehicles and ships positioned dynamically [19]. Furthermore, Kalman filtering is a concept much applied in time series analysis used for topics such as signal processing and econometrics. Kalman filtering is also one of the main topics of robotic motion planning and control and plays a good role in trajectory optimization [20]. Kalman filtering is also present in some systems for modeling the central nervous system's control of movement. The presence of Kalman filters in the control systems provides a realistic model for making estimates of the current state of a motor system and issuing updated commands because of the time delay between issuing motor commands and receiving sensory feedback [21].

The algorithm develops in two phases: the prediction phase, where the Kalman filter produces estimates of the current state variables, along with their uncertainties, and the updated phase, in which, once the outcome of the next measurement corrupted with some error is observed, these estimates are updated using a weighted average, giving more weight to estimates with greater certainty. The algorithm is recursive and it can operate in real-time, using only the present input measurements and the state calculated previously and its uncertainty matrix.

For the nonlinear systems, the Kalman filter has some calculation problems, so extensions and generalizations of the method are necessary, such as the extended Kalman filter and the unscented Kalman filter. The fulcrum of the theory is a hidden Markov model such that the state space of the latent variables is continuous and all latent and observed variables have Gaussian distributions.

Kalman Filter Computation

The goal of the Kalman filter is to try to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is given by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \tag{2.10}$$

with a measurement $z \in \mathbb{R}^m$ that is

$$z_k = Hx_k + v_k \tag{2.11}$$

The random variables w_k and v_k are the process and measurement noise and they are assumed independent, white and with normal probability distributions

$$p(w) \sim N(0, Q), \tag{2.12}$$

$$p(v) \sim N(0, R), \tag{2.13}$$

with Q and R respectively the process noise covariance and measurement noise covariance.

The n x n matrix A in the equation (2.10) relates the state at the previous time step k-1 to the state at the current step k. The n x l matrix B relates the optional control input $u \in \mathbb{R}^l$ to the state x. The m x n matrix H in the measurement equation (2.11) relates the states to the measurement z_k.

In deriving the equations for Kalman filter, a posteriori state estimate x_k is seen as a linear combination of an a priori estimate x_k^- and a weighted difference between an actual measurement z_k and a measurement prediction $H\widehat{x}_k^-$:

$$\widehat{x}_k = \widehat{x}_k^- + K_k(z_k - H\widehat{x}_k^-), \tag{2.14}$$

The difference $(z_k - H\widehat{x}_k^-)$ in (2.14) is called the measurement innovation, or the residual. The residual is the discrepancy between the predicted measurement $H\widehat{x}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The n x m matrix K in (2.14) is chosen to be the gain or blending factor that minimizes the a posteriori error covariance. The optimal Kalman gain is:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.15}$$

Looking at (2.15) it is possible to see that as the measurement error covariance R approaches zeros, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1} \tag{2.16}$$

On the other hand, as the a priori estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily. Specifically,

$$\lim_{P_k^- \to 0} K_k = 0 \tag{2.17}$$

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the Kalman filter are divided into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

The time update equations can also be seen as predictor equations, while the measurement update equations can be seen as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

The specific equations for the time and measurement updates are:

$$\widehat{x}_k^- = A\widehat{x}_{k-1} + Bu_{k-1} \tag{2.18}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.19}$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.20}$$

$$\widehat{x}_k = \widehat{x}_k^- + K_k(z_k - H\widehat{x}_k^-), \tag{2.21}$$

$$P_k = (I - K_k H)P_K^- \tag{2.22}$$

The first task during the measurement update is to compute the Kalman gain, K_k. The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement as in (2.21). The final step is to obtain an a posteriori error covariance estimate through (2.22).

After each time and measurement update pair, the algorithm is repeated with the previous a posteriori estimates used to calculate or predict the new a priori estimates. The Kalman filter recursively conditions the current estimate on all of the past measurements. Figure 2.11 shows a summary of how the filter works, combining the time and measurement updates equations.

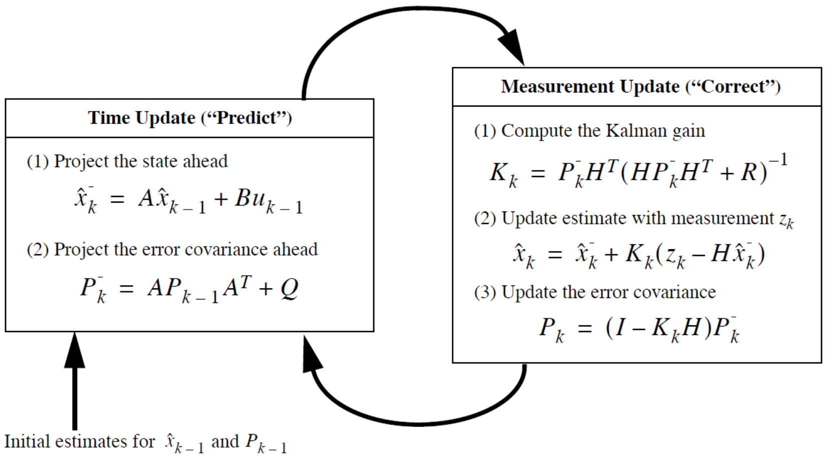

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 2.11: A complete picture of the operation of the Kalman filter [22]

In the implementation of the filter, each of the measurement error covariance matrix R_k and the process noise Q_k(given by (2.13) and (2.12) respectively) might be measured prior to operation of the filter. In the case of the measurement error covariance R_k this makes sense, as we need to be able to measure the process we should generally be able to perform some off-line sample measurements to determine the variance of the measurement error. In the case of Q_k, the choice is often less deterministic. For example, this noise source is often used to represent uncertainty in the process model. In presence of a very poor model, it can be used by adding enough uncertainty by selecting Q_k. Certainly in this case it is hoped that the process measurements are reliable. In either case, superior filter performance can be often obtained by "tuning" the filter parameters Q and R that can be performed with the help of another Kalman filter.

When Q_k and R_k are constant, both the estimation error covariance P_k and the Kalman gain K_k will stabilize quickly and then remain constant. In this case these parameters can be pre-computed by either running the filter off-line or for example by solving (2.19) for the steady-state value of P_k by defining $P_k^- \equiv P_k$ and solving for P_k.

However, it often happens that the measurement error does not remain constant. Also, the Q_k process noise is sometimes dynamically changed during filter operation to accommodate different dynamics. For example, in the case of tracing a user's head of a 3D virtual environment, it could reduce the amount by Q_k if the user appears to be moving slowly and increase the amount of dynamics that begin to change rapidly. In this case Q_k can be used to model not only the uncertainty in the model but also the uncertainty of the user's intent [22].

2.3.3 Extended Kalman Filter

The Extended Kalman filter (EKF) is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance [23]. The EKF is considered the standard in the theory of nonlinear state estimation, navigation systems and GPS [24].

The EKF equations are similar to those of the Kalman filter and are summarized:

Predict phase

$$\widehat{x}_{k|k-1} = f(\widehat{x}_{k-1|k-1}) \tag{2.23}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.24}$$

$$F_k = \frac{\delta f}{\delta x}|_{\widehat{x}_{k-1|k-1}} \tag{2.25}$$

Update phase

$$\tilde{y}_k = z_k - h(\widehat{x}_{k|k-1}) \tag{2.26}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.27}$$

$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.28}$$

$$P_{k|k} = (I_n - K_k H_k) P_{k|k-1} \tag{2.29}$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.30}$$

Below it is possible to see the equations that characterize the EKF and its covariance matrices.

Position - Velocity (PV) model

$$x = \begin{bmatrix} x & v \end{bmatrix}^T \tag{2.31}$$

The state transition function is equal to:

$$f(x_{k-1}) = F_k x_{k-1} = \begin{bmatrix} I_n & \Delta t_k I_n \\ 0_n & I_n \end{bmatrix} x_{k-1} \tag{2.32}$$

The covariance matrix of the state is:

$$Q_{\tilde{a},k} = \begin{bmatrix} \frac{\Delta t_k^2}{2} I_n \\ \Delta t_k I_n \end{bmatrix} \begin{bmatrix} \sigma_{\tilde{a}_1}^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_{\tilde{a}_1}^2 \end{bmatrix} \begin{bmatrix} \frac{\Delta t_k^2}{2} I_n \\ \Delta t_k I_n \end{bmatrix}^T \tag{2.33}$$

The EKF observation vector is then defined as:

$$z = \begin{bmatrix} d_{ref_1} & \dots & d_{ref_L} \end{bmatrix}^T, \tag{2.34}$$

where

$$d_{ref_l} = dist(x, x_{ref_l}) = \sqrt{\sum_{i=1}^{n} (x_i - x_{i,ref_l})^2} \tag{2.35}$$

Consequently, the observation function is defined as the distance between the position component of the state vector and the anchors:

$$h(x_k) = \begin{bmatrix} dist(x_k, x_{ref_1}) \\ \vdots \\ dist(x_k, x_{ref_L}) \end{bmatrix}, \tag{2.36}$$

Since $h(x_k)$ is non-linear, the Jacobian matrix H needs to be computed around the a priori state $x_{k|k-1}$:

$$H_k = \begin{bmatrix} \frac{x_{1,k|k-1} - x_{1,ref_1}}{dist(x_{k|k-1}, x_{ref_1})} & \dots & \frac{x_{n,k|k-1} - x_{n,ref_1}}{dist(x_{k|k-1}, x_{ref_1})} \\ \vdots & \ddots & \vdots \\ \frac{x_{1,k|k-1} - x_{1,ref_L}}{dist(x_{k|k-1}, x_{ref_L})} & \dots & \frac{x_{n,k|k-1} - x_{n,ref_L}}{dist(x_{k|k-1}, x_{ref_L})} \end{bmatrix} \tag{2.37}$$

The observation errors are modeled as uncorrelated white Gausssian noises:

$$R_{d,k} = diag(\sigma_{d_{ref_1},k}^2 \dots \sigma_{d_{ref_L},k}^2), \tag{2.38}$$

2.3.4 Weighted Outlier-Robust Kalman Filter

Measurements can sometimes contain errors due to the high distance or presence of obstacles between tag node and anchor nodes. This type of error is called outlier and the EKF does not react well to the outliers. It is necessary to make some change in the EKF: it is possible to add a scalar weight for each coordinate of position and velocity and modify the covariance matrices. These modifications are present in the EKF with Outlier Detection. It is worth noticing that in the Covariance matrix the observation error is divided by w_k, in this way the output will be influenced more by the states with less noise than the other ones in case of outlier.

Propagation:

$$x_k = f(x_{k-1}) \tag{2.39}$$

$$P_{k|k-1} = Q_k \tag{2.40}$$

Update:

$$S_k = H_k P_{k|k-1} H_k^T + \frac{R_k}{\langle w_k \rangle} \tag{2.41}$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.42}$$

$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.43}$$

$$P_{k|k} = (I_n - K_k H_k) Q_k \tag{2.44}$$

$$\langle w_k \rangle = \frac{a_{w_k,0} + \frac{1}{2}}{b_{w_k,0} + \left\langle (z_k - h(\widehat{x}_{k|k-1}))^T R_k^{-1} (z_k - h(\widehat{x}_{k|k-1})) \right\rangle} \tag{2.45}$$

It is worth noticing that the equations (2.41) and (2.40) have some differences respectively with the equations (2.27) and (2.22); whereas the equation (2.45) is introduced in this filter calculation to determine the weight w_k useful for the calculation of the equation (2.41).

2.4 ROS and Gazebo

2.4.1 ROS

The Robot Operating System (ROS) is a set of software libraries and tools useful to build robot applications. Ros is an open-source, meta-operating system for the robot. It provides the services typical of the operating systems, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers [25].

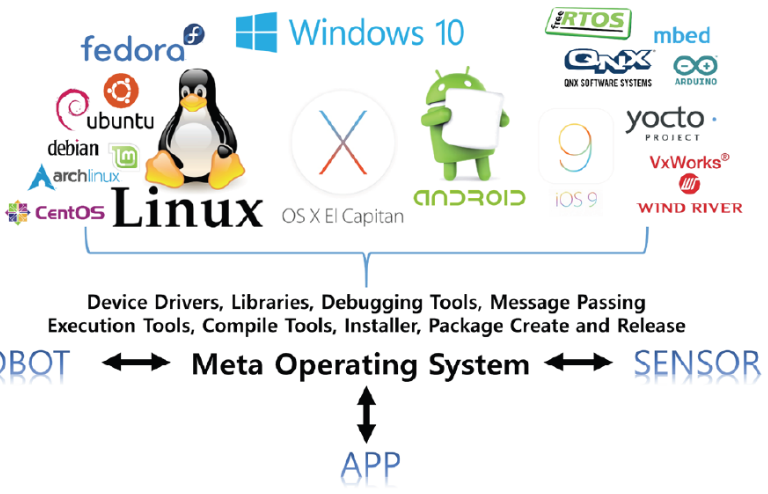


Figure 2.12: ROS as a Meta-Operating System [26]

ROS is a Meta-Operating System, that is a system that performs processes such as scheduling, loading, monitoring, and error handling by utilizing a virtualization layer between applications and distributed computing resources. Furthermore, ROS runs on the existing operating such as Ubuntu, which is one of Linux's distributions.

ROS is a supporting system for controlling a robot and sensor with a hardware abstraction and developing robot applications based on existing conventional operating systems. Moreover, ROS data communication is supported not only by one operating system, but also by multiple operating systems, hardware, and programs, making it highly suitable for robot development where various hardware are combined [26].

Objectives of ROS

The goal of ROS is to "build the development environment that allows robotic software development to collaborate on a global level". ROS is focused on maximizing code reuse in the robotics research and development, rather than orienting towards the so-called robot software platform, middleware, and framework. To support this, ROS has the following characteristics.

- Distributed process: it is programmed in the form of the minimum units of executable processes, and each process runs independently and exchanges data systematically.
- Package management: packages contain multiple similar processes and they are easy to use and develop.

- Public repository: each package is made public to the developer's preferred public repository such as GitHub.

- API: when developing a program that uses ROS, ROS is designed to simply call an API and insert it easily into the code being used.

- Supporting various programming languages: the ROS program provides a client library to support various programming languages. It is possible to develop a ROS program using a preferred programming language [26].

ROS terminology

Here some common terminology in ROS programs.

Master is the name server for node-to-node connections and message communication. The command 'roscore' is used to run the master and the name of each node can be registered and information is got. The connection between nodes and message communication such as topics and services is impossible without the master.

Node is the smallest unit of the processor running in ROS. ROS recommends creating one single node for each purpose, and it is recommended to develop for easy reusability. For example, in case of mobile robots, the program to operate the robot is broken down into specialized functions. Specialized node is used for each function such as sensor drive, sensor data conversion, obstacle recognition, motor drive, encoder input, and navigation.

Package is the basic unit of ROS. The ROS application is developed on a package basis, and the package contains either a configuration file to launch other packages or nodes. The package also contains all the files necessary for running the package.

Message, a node sends or receives data between nodes via a message. Messages are variable such as integer, floating-point, and boolean.

Topic, the publisher node first registers its topic with the master and then starts publishing messages on a topic subscriber nodes that want to receive the topic request information of the publisher node corresponding to the name of the topic registered in the master. Based on this information, the subscriber node directly connects to the publisher node to exchange messages as a topic.

Publish and **publisher**, the term 'publish' stands for the action of transmitting relative messages corresponding to the topic. The publisher node registers its information and topic with the master and sends a message to connect subscriber nodes that are interested in the same topic. The publisher is declared in the node and can be declared multiple times in one node.

Subscribe and **subscriber**, the term 'subscribe' stands for the action of receiving relative messages corresponding to the topic. The subscriber node registers its information and topic with the master, and receives publisher information that publishes relative topic from the master. Based on received publisher node and receives messages from the connected publisher node. A subscriber is declared in the node and can be declared multiple times in one node.

Service is synchronous bidirectional communication between the service client that requests a service regarding a particular task and the service server that is responsible for responding to requests.

Catkin refers to the build system of ROS. The build system uses CMake and the build environment is described in the 'CMakeLists.txt' file in the package folder. Cmake was modified in ROS to create a ROS-specific build system.

Roscore is the command that runs the ROS master. If multiple computers are within the same network, it can be run from another computer in the network. However, except for special case that supports multiple roscore, only one roscore should be running in the network.

Rosrun is the basic execution command of ROS. It is used to run a single node in the package.

Roslaunch executes multiple nodes. It is a ROS command specialized in node execution with additional functions such as changing package parameters or node names, configuring namespace of nodes [26].

Message communication

The key concept is the message communication methods among nodes. There are three different methods of exchanging messages: a topic which provides a unidirectional message transmission/reception, a service which provides a bidirectional message request/response and an action which provides a bidirectional message goal/result/feedback.

Communication over **topics** allows the subscriber node to receive the publisher node information corresponding to the identical topic name registered in the master. Since topics are unidirectional and remain connected to send or receive messages continuously, it is suitable for sensor data that requires messages to be published periodically. In addition, multiple subscribers can receive messages from a publisher and vice versa.

Service consists of a service server that responds only when there is a request and a service client that can send requests as well as receive responses. Unlike the topic, the service is one-time message communication and does not maintain the connection. A service is often used to command a robot to perform a specific action or nodes to perform certain events with specific conditions.

Communication on **action** is used when a requested goal takes a long time to be completed, therefore progress feedback is necessary. This is very similar to the service and the 'feedback' is added to report feedbacks to the client periodically when intermediate values are needed. The message transmission method is the same as the asynchronous topic. The feedback transmits an asynchronous bidirectional message between the action client which sets the goal of the action and an action server that performs the action and sends the feedback to the action client [26].

2.4.2 Gazebo

Gazebo is a 3D simulator that provides robots, sensors, environment models for 3D simulation required for robot development, and offers realistic simulation with its physics engine. Gazebo is an open-source simulator and is useful to rapidly test algorithms, design robots, perform regression testing, and train AI systems using realistic scenarios. It is a very popular simulator in the field of robotics and offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. Moreover, Gazebo is developed and distributed by Open Robotics which is in charge of ROS and its community, so it is very compatible with ROS [27].

The main characteristics used in Gazebo are the following: Dynamic simluation, 3D graphics, Sensors and noise simulation, Plug-ins, Robot model, Tcp/Ip data transmission, Cloud simulation, Command line tool [26].

Chapter 3

Simulations

The Extended Kalman filter is the most suitable tool to decrease the error during the position calculation and improve the stability of the system.
In order to have a good filter it is necessary to set precise values in the covariance matrices. This has been done as a result of various simulations done on simulation programs such as Matlab and Gazebo.

3.1 Simulations on Matlab

The Extended Kalman filter obtains the position of the tag node from the measurements of the distances between anchor nodes and tag node.

A Gaussian error with standard deviation of 0.2 m and mean error equal to 0 m was added to simulate the real world.
A configuration with 5 nodes was created with these coordinates (Table 3.1) taking into account a radio connectivity of 30 m like the devices used in real life tests.
Two types of simulations have been done to calculate the position of a tag node: in the first one the tag node positioned on the same point for 50 times in a row and in the second one the tag node follows a trajectory with straights and curves.

Anchor node	**Coordinate**
Anchor1	(1.00,1.00)
Anchor2	(19.00,1.00)
Anchor3	(10.00,10.00)
Anchor4	(1.00,19.00)
Anchor5	(19.00,19.00)

Table 3.1: Coordinates (in meters) of the anchor nodes in MatLab simulation

3.1.1 Static simulations

The positions obtained by EKF are very precise, but it is necessary to consider the presence of outlier errors related to connectivity, for example when there are obstacles (NLOS), as in reality. The Figure 3.1 shows positions obtained with the simple EKF (in blue) and those obtained with the EKF with outlier detection (in red).

To simulate outliers the Gaussian error at some points was multiplied by 10 with a standard deviation of 2 meters. This experience shows how convenient it is the EKF with outlier detection that manages to have results very close to the desired point. In blue, there are some points far from the desired point and this creates some problems in the optimization of the trajectory. Instead in red, the simulated points are close giving a stable and linear behavior to the drone.

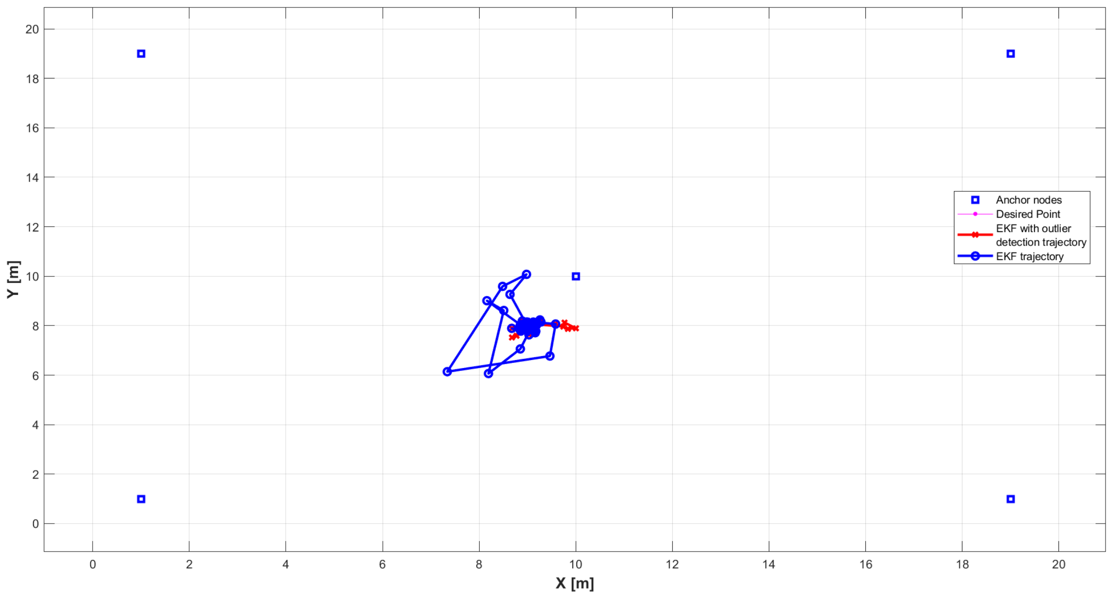


Figure 3.1: Position simulation 50 times in (9.8). In blue EKF calculation; In red EKF with outlier detection.

Below the mean errors of the filtered simulations.

	EKF	**EKF with outlier detection**
avg err	0.4230	0.2446
rms err	0.7279	0.3256

Table 3.2: Mean errors of the position simulation 50 times in (9.8).

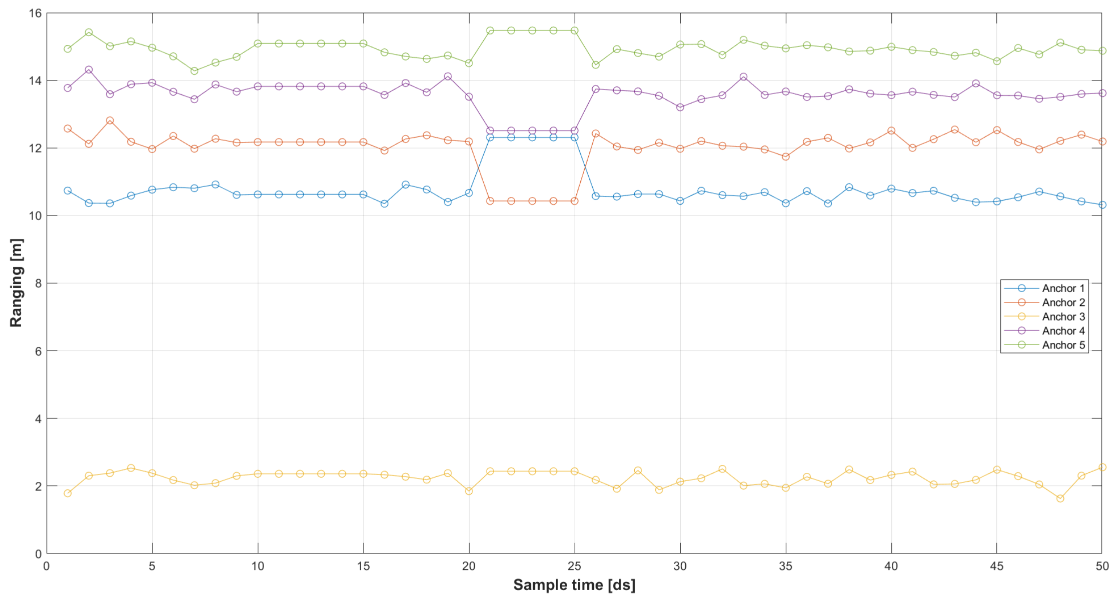


Figure 3.2: Ranging of the EKF with outlier detection in Position simulation 50 times in (9.8).

3.1.2 Dynamic simulations

Other simulations have been made in the dynamic case: The trajectory to be executed was plotted and the measurements (with disturbances) were filtered, as can be seen in Figure 3.3.

Also in this case the outlier errors (NLOS) were considered, reproduced with a standard deviation of 2 meters in two intervals: between sample time 10 and 15 and between sample time 20 and 25, where there are a curve and a straight line. The filtered trajectory (in blue) follows the desired trajectory except in the areas where outliers are present. Instead the EKF with the outlier detection (in red), has a better behavior: the trajectory generated by the output results is closer to the desired one (Figure 3.3). The Table 3.3) shows the error in both the case.

	EKF	**EKF with outlier detection**
avg err	0.3470	0.3545
rms err	0.6107	0.5531

Table 3.3: Mean errors of the filtered position simulation of a trajectory.

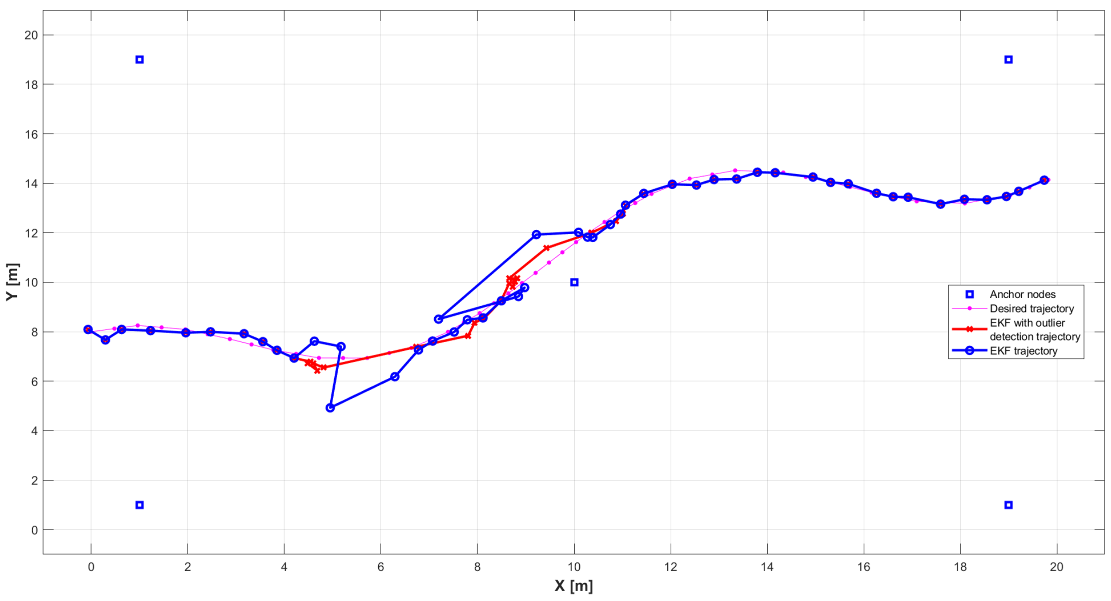


Figure 3.3: Position simulation of a trajectory

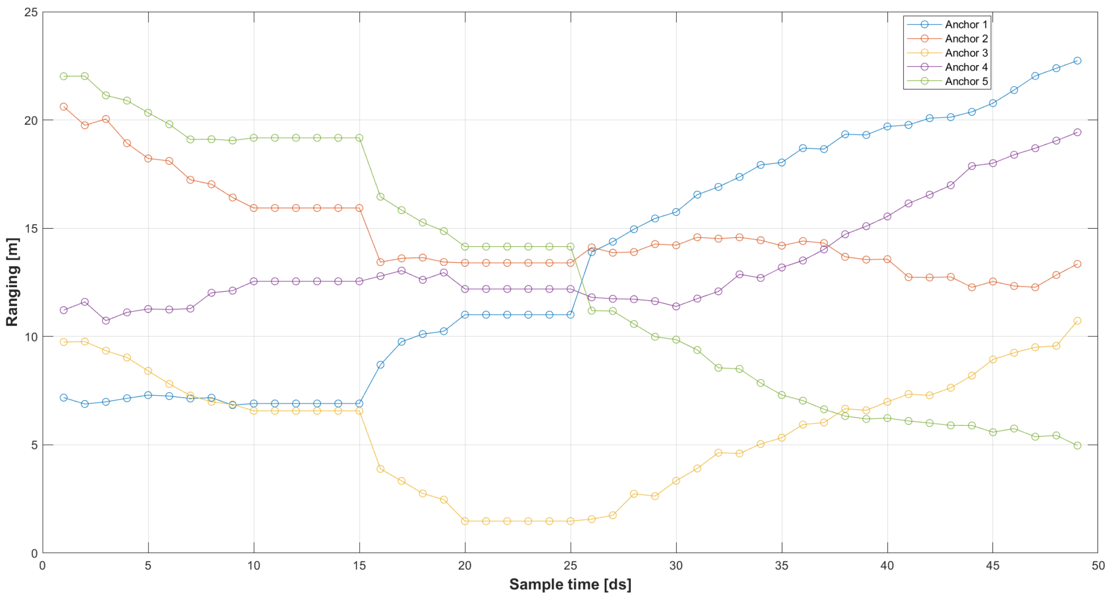


Figure 3.4: Ranging of the EKF with outlier detection in position simulation of a trajectory.

3.2 Filter implementation in Python

Once verified that the filter has a good performance, the code of the filter has been rewritten in Python to make other tests on the actual code that will be used in the real tests.

In this paragraph are commented some simulations made on Python having a precise number of outlier errors, but unknown position of outlier errors and unknown disturbances for each anchor node (standard deviation between 0.2 and 2.0 meters).

3.2.1 First simulation on Python

Figure 3.5 shows one of the 200 simulations done with an outlier ratio equal to 0.2, i.e. outlier errors in 10 points (0.2 x 50 total points).

	EKF	**EKF with outlier detection**
avg err	0.2600	0.2804
rms err	0.3580	0.3882

Table 3.4: Mean errors of 200 filtered simulations with 0.2 outlier ratio.

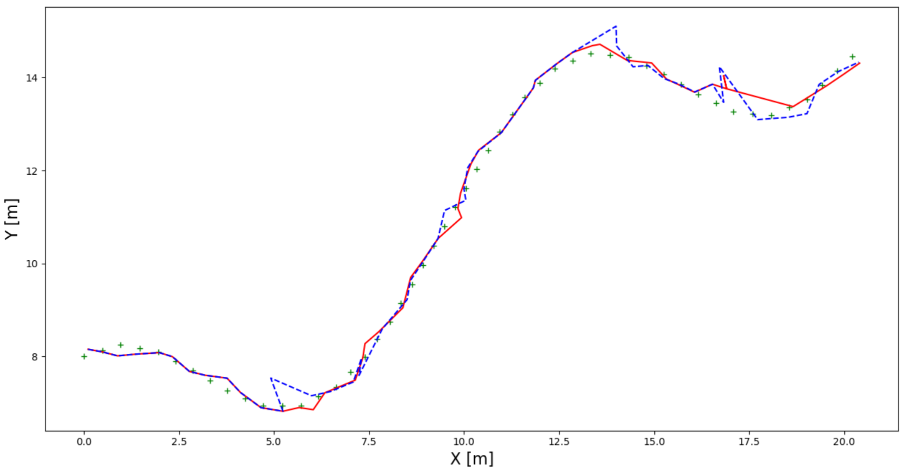


Figure 3.5: Simulation with 0.2 outlier ratio on Python. Desired trajectory in green, EKF trajectory in blue, EKF with outlier detection trajectory in red.

3.2.2 Second simulation on Python

Figure 3.6 shows one of the 200 simulations done with an outlier ratio equal to 0.35, i.e. outlier errors in 18 points (0.35 x 50 total points).

	EKF	**EKF with outlier detection**
avg err	0.3274	0.4247
rms err	0.4326	0.5893

Table 3.5: Mean errors of 200 filtered dynamic simulations with 0.35 outlier ratio.

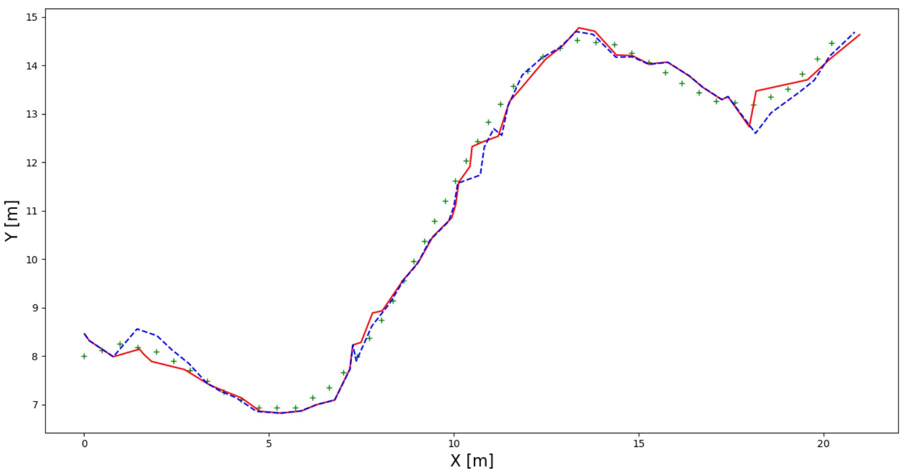


Figure 3.6: Simulation with 0.35 outlier ratio on Python. Desired trajectory in green, EKF trajectory in blue, EKF with outlier detection trajectory in red.

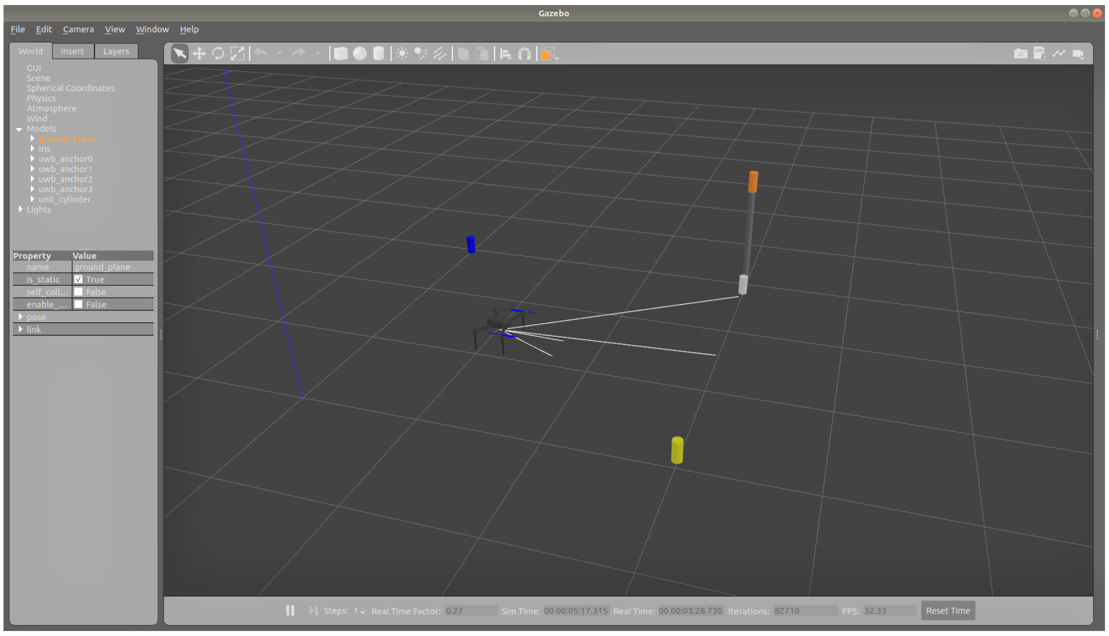
On 200 simulations the average errors show a better accuracy of the EKF instead of the one with outlier detection. But it is possible to see in the figures that for both the simulations with 0.2 outlier ratio and with 0.35 outlier ratio the behavior in red is more stable and does not have points very far from the desired trajectory.

It is worth noting that Gaussian errors with a very high standard deviation (2.0 meters) were considered to simulate the outlier errors and, especially in the second simulation, there are many of these errors along the trajectory.

3.3 Simulations on Gazebo

In parallel with the experimental tests, flight simulations have been performed on the Gazebo simulation program to simulate environments with multiple drones and obstacles between nodes.

An existing plugin [28] has been used to derive the ranges between anchor nodes and tag nodes. A tag node has been implemented on the drone model, and four anchor nodes have been placed in the environment simulating one of the two cells of the cage used for the experimental tests, as shown in Figure 3.7. The filter code has been adapted for these simulations on Gazebo. It allows calculating the position of the drone taking as input the range obtained from the topic */gtec/toa/ranging*.



Property	Value
name	ground_plane
is_static	✓ True
self_coll...	False
enable_...	False
pose	
link	

Steps: 1 Real Time Factor: 0.27 Sim Time: 00 00:05:17.315 Real Time: 00 00:03:28.730 Iterations: 82710 FPS: 32.33

Figure 3.7: Gazebo environment

The values coming out of the */gtec/toa/ranging* topic compared with the distance between nodes are not exact and cannot be compensated with a fixed bias during flight simulations. For the simulation it is necessary a different filter than the one tested in real life.

Chapter 4

Experimental tests

4.1 DECAWAVE

In this thesis UWB technology is used to create an indoor localization system. Decawave is one of the first companies to produce UWB technology in the civil field. It develops semiconductor solutions, software, modules, and reference designs that enable real-time, ultra-accurate, ultra reliable local area micro-location services [29].
Decawave's technology enables intelligent location functionality. It is employed for IoT services, smart consumer products, and applications.
Decawave is headquartered in Ireland, with regional headquarters in California and China, and a presence in South Korea, France, and Japan.

4.1.1 DWM1001

In the real life tests DWM1001 devices are used as tag node and anchor nodes acting as a network gateway device. They have a single-chip wireless transceiver working with a large bandwidth of 6.5GHz, and the accuracy is around 20 centimeters.

The DWM1001C module combines the DW1000, a Nordic Semiconductor nRF52832 MCU, and a 3-axis accelerometer. It builds scalable TWR RTLS systems up to thousands of tags and the same module is used for anchor, tag, and gateway design. Embedded DRTLS firmware reduces software development efforts. Bluetooth for connectivity and motion sensors are integrated. It uses channel 5 with a frequency of 6.5GHz and a data rate of 6.8Mbps.

The DWM1001 Development Board can be used for various applications, including asset tracking, navigation, factory automation, security, and consumer applications [30].

Figure 4.1: Decawave DWM1001 [31]

4.2 UWB configuration

The objective of this thesis is to create an indoor UWB localization system. To do this it is necessary to position the anchor nodes most properly, decreasing the presence of errors and disturbances in the system. Several studies [14] state that it is best to use: at least three anchor nodes to determine the XY plane location of a given node, at least four for localization in XYZ space, at least one of which is at a different height.

A protection cage for flights has been installed to create an indoor network, with dimensions 6.00 m x 3.00 m x 2.50 m, divided into two equal cells. Five anchors were placed in the environment, four on the upper floor and one on the lower one. They compose two cells of 4 anchors, which represents the minimum number for 3D triangulation, as mentioned before.
Setting a vertex of the cage as the origin (in blue), the coordinates of the anchor nodes (in red) have been measured, as depicted in Figure 4.2 and Table 4.1.
Instead, Figure 4.3 shows how the tag node is displayed in real-time in the Decawave app.

Anchor node	Coordinate
Anchor1	(0.00,2.95,2.04)
Anchor2	(2.99,2.95,2.23)
Anchor3	(2.99,2.95,0.38)
Anchor4	(5.98,2.95,2.18)
Anchor5	(2.99,0.10,2.23)

Table 4.1: Coordinates of the anchor nodes in meters

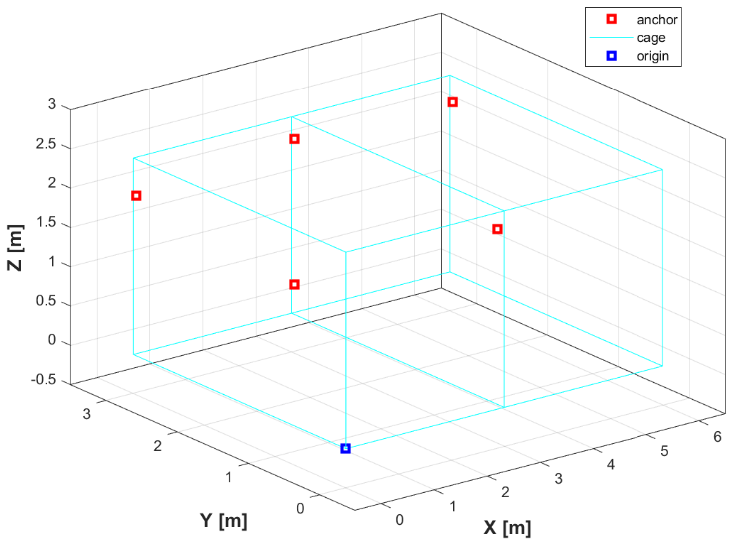


Figure 4.2: Anchors Configuration represented on Matlab

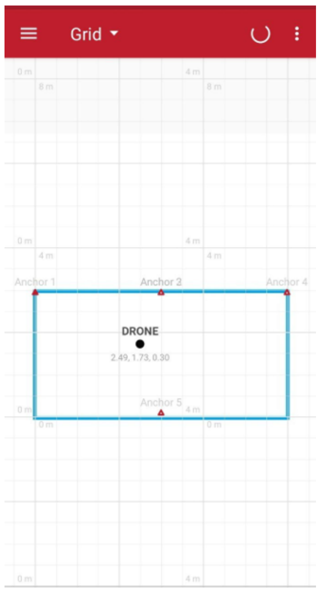


Figure 4.3: Decawave APP look

4.3 First tests

4.3.1 Tests without filter

The same simulations done on Matlab have been implemented in the real world. In the first tests, the tag node is placed in ten different positions at four different heights in a given time interval to visualize the positioning obtained by triangulation from the anchor nodes.

Figure 4.4 shows the positioning of the ten tests done with the tag node at height 0.094 m. These positions are obtained by the UWB system from the Decawave device, without any filter.
The positions of the tag node that are approximately in the center of the y-axis are the best-sampled ones. They have all the points close together, a sort of a point cloud, near the position of the tag node. The others have lower precision: the point clouds are far from the desired point. In addition, they are very large because of the poor accuracy of the measurements.

4.3.2 Tests with EKF with outlier detection

Then, it is necessary to insert an EKF with outlier detection that takes the ranging measurements calculated by the Decawave devices and determines the location of

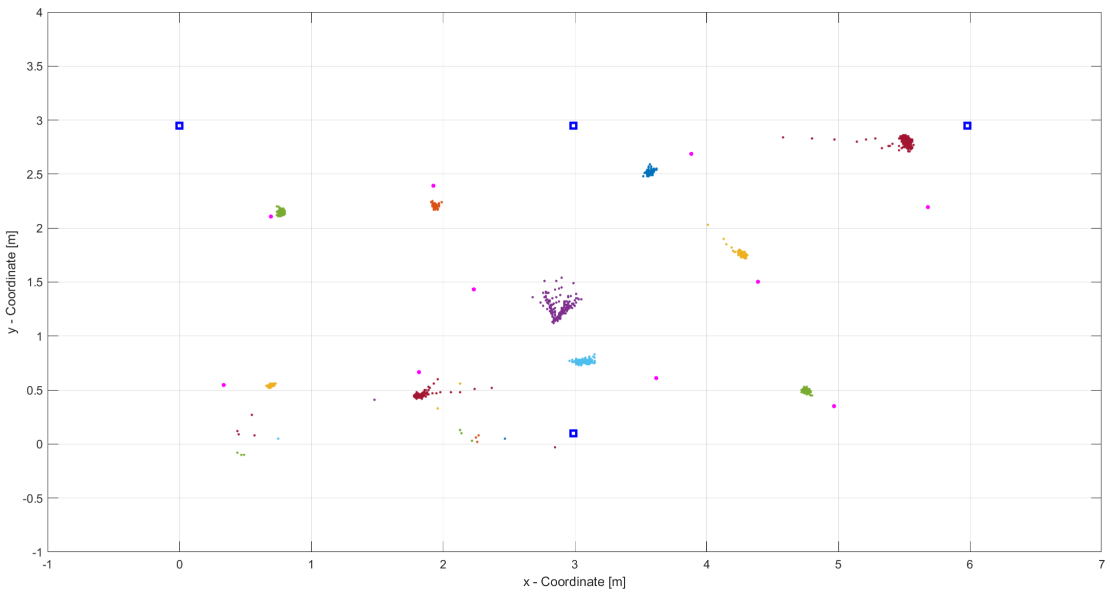


Figure 4.4: Plot of the tests at flight altitude of 0.094 m

the tag node.

The data from the previous tests have been filtered. Following figures shows how results have been obtained and compared to the previous tests.

It is worth noticing that graphically there is a difference between the two outputs: at each height, almost all ten filtered points do not have outliers errors; the points close to the transition between the two cells are most hostile to this behavior. This is because in some cases the triangulation occurs in the left cell and others in the right one. This makes the position calculation worse, as in the case without filtering. Another observation has to be made: there are still systematic errors between the desired point and the filtered point.

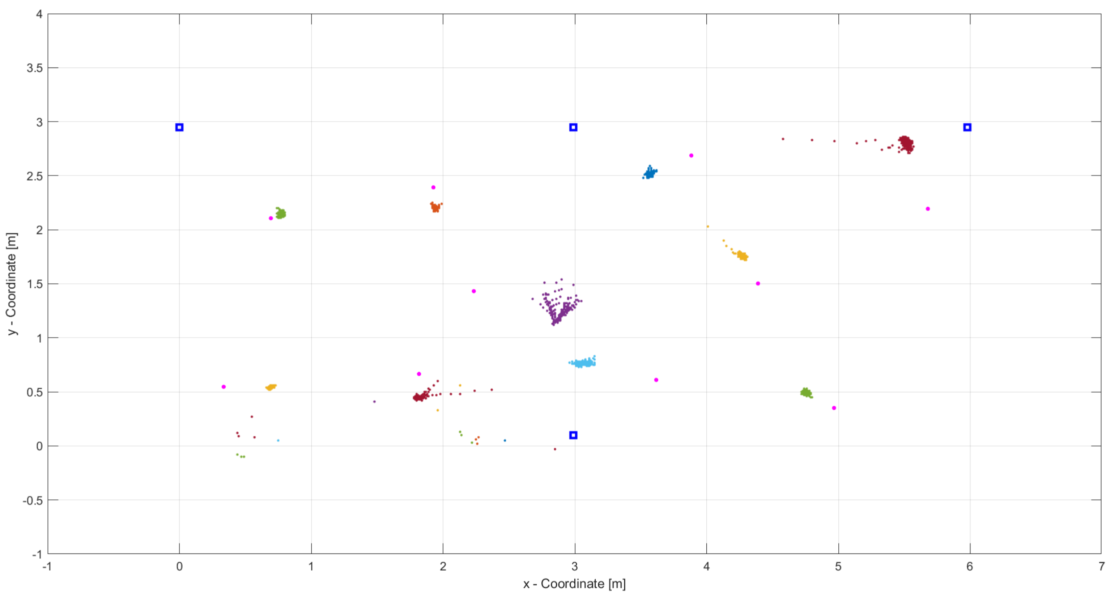


Figure 4.5: Plot of the Decawave tests at flight altitude of 0.094 m

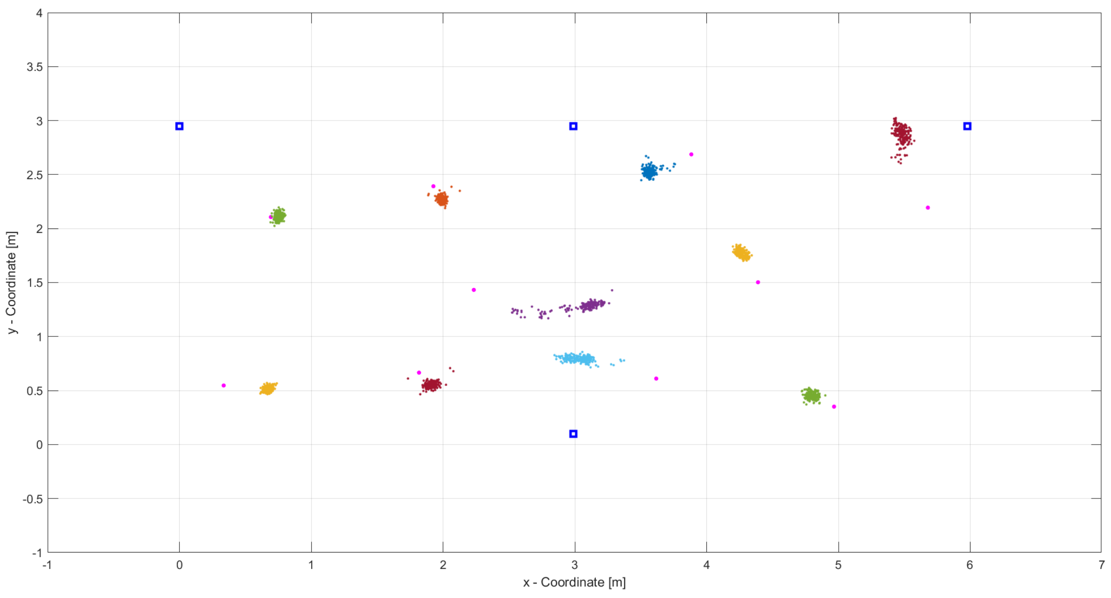


Figure 4.6: Plot of the filtered tests at flight altitude of 0.094 m

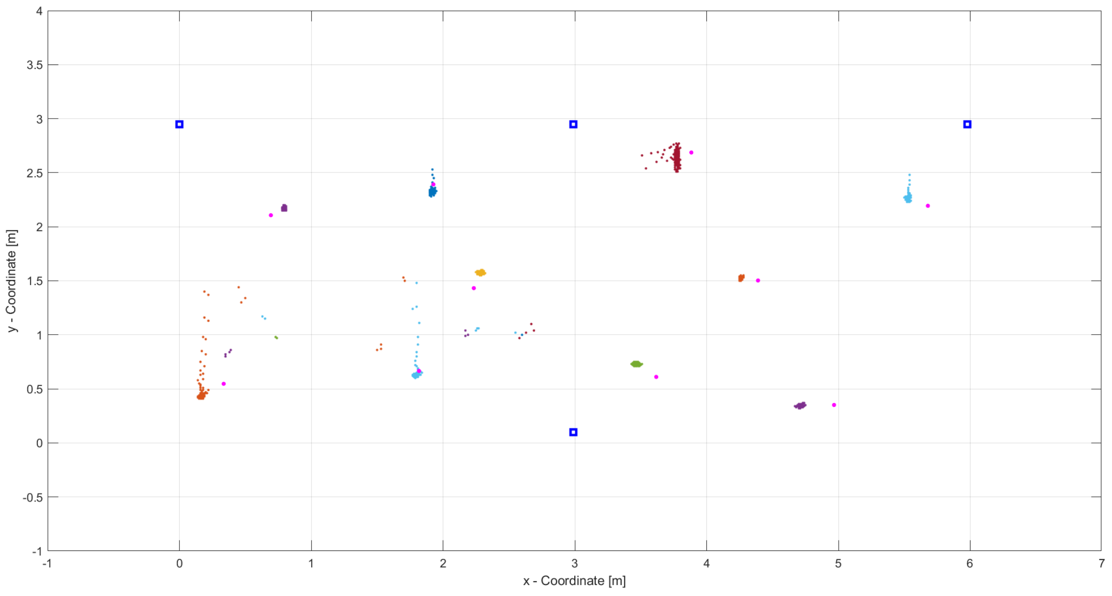


Figure 4.7: Plot of the Decawave tests at flight altitude of 0.97 m

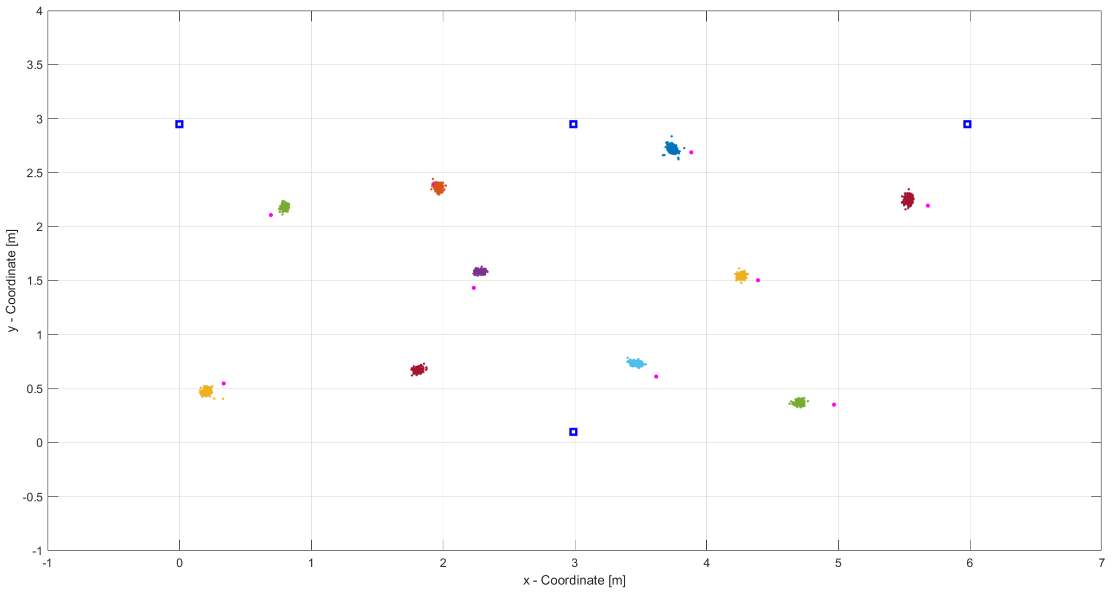


Figure 4.8: Plot of the filtered tests at flight altitude of 0.97 m

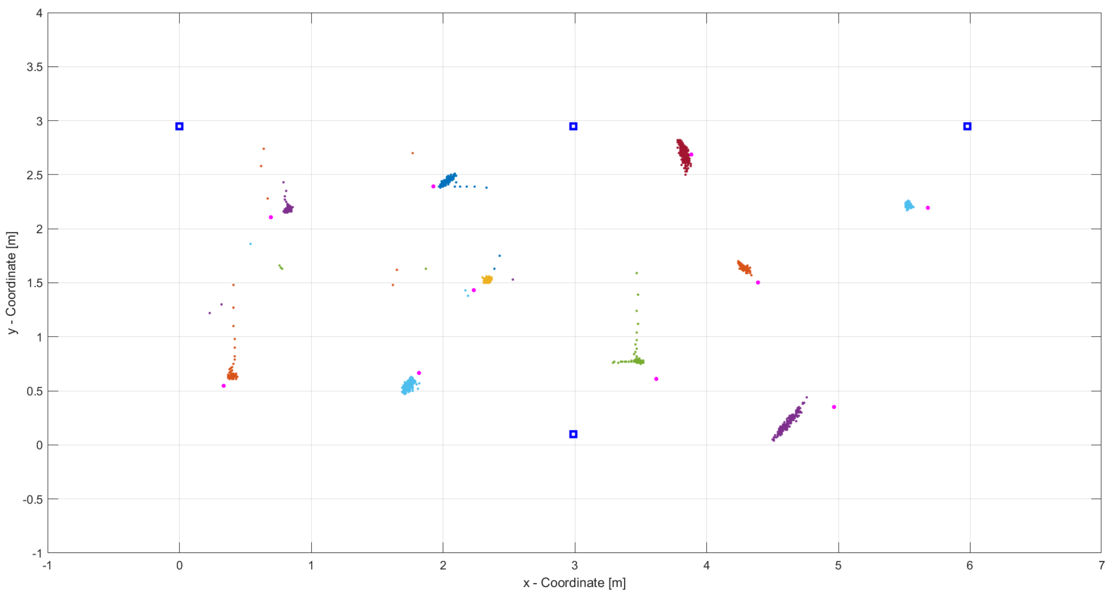


Figure 4.9: Plot of the Decawave tests at flight altitude of 1.53 m

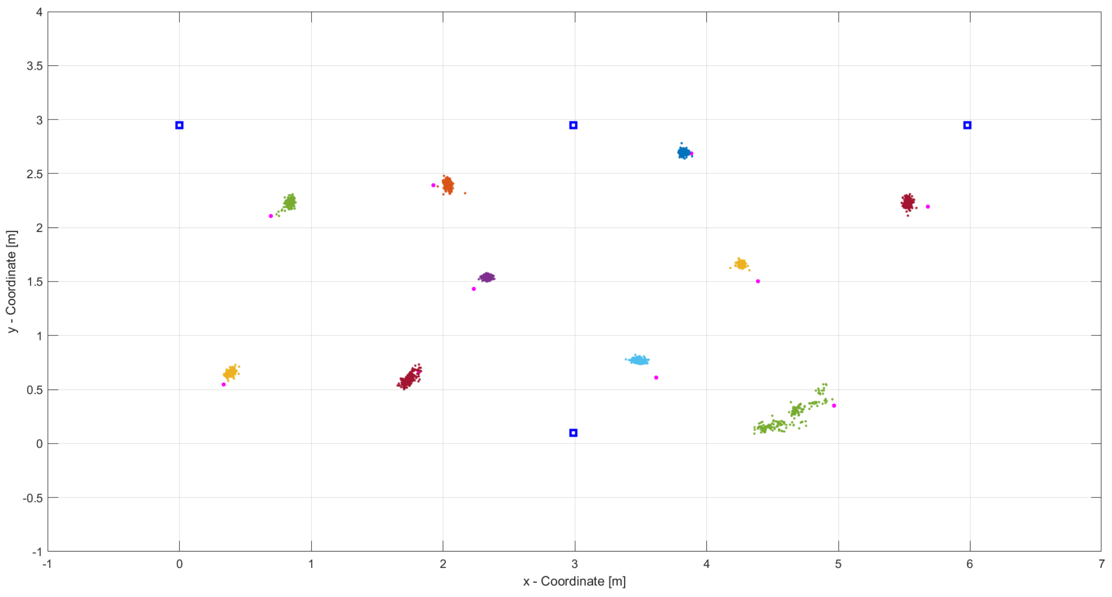


Figure 4.10: Plot of the filtered tests at flight altitude of 1.53 m

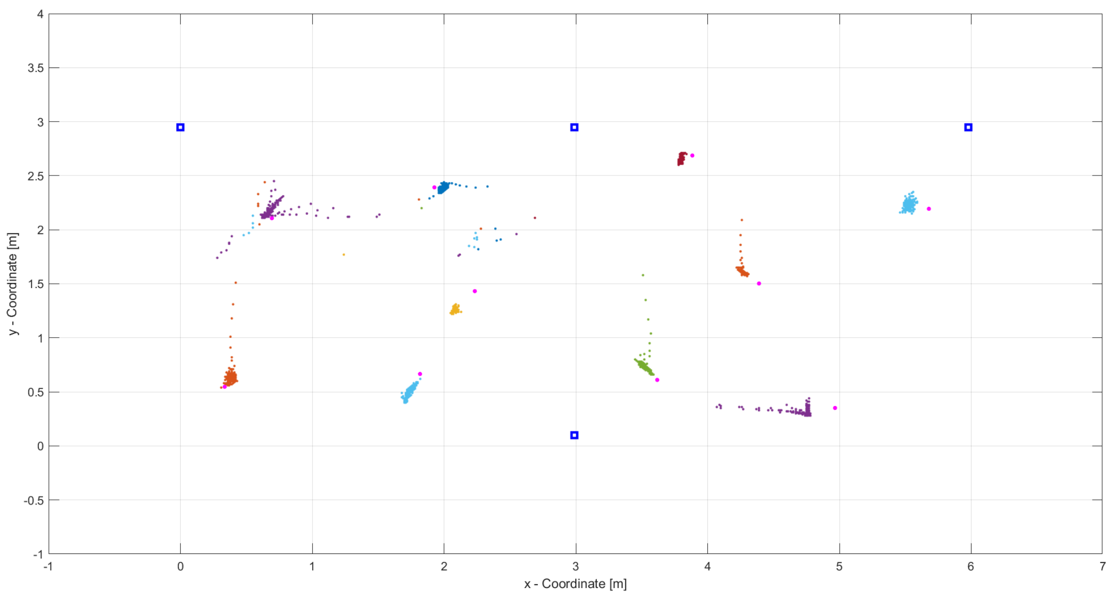


Figure 4.11: Plot of the Decawave tests at flight altitude of 2.02 m

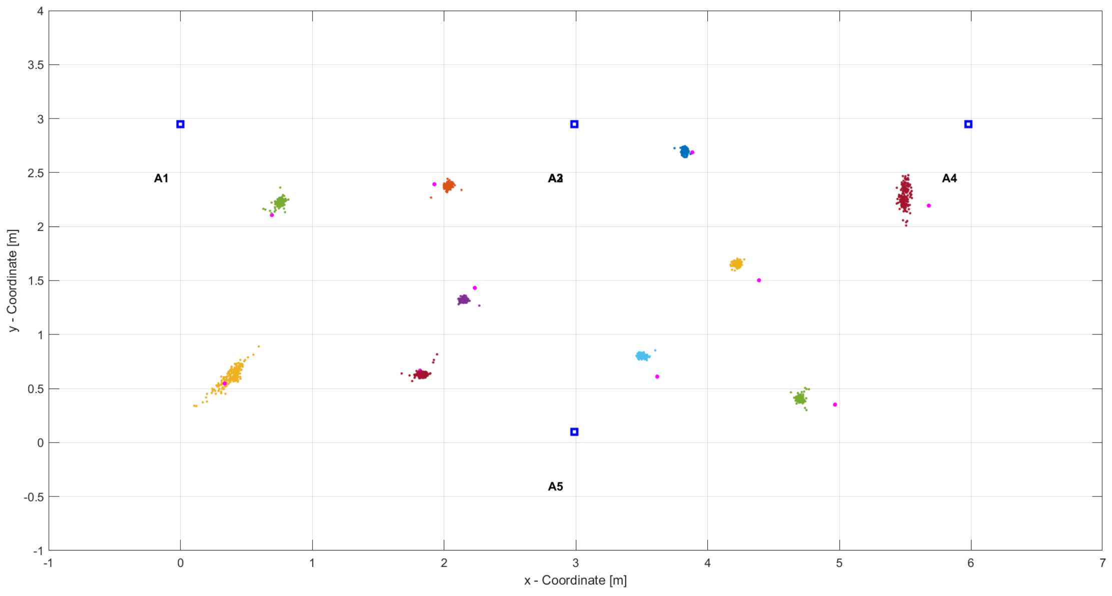


Figure 4.12: Plot of the filtered tests at flight altitude of 2.02 m

4.3.3 Improved tests with EKF with outlier detection

Systematic error *bias* can be decreased in the following way:

$$bias = z - dist \tag{4.1}$$

where *dist* is the real distances between the anchor nodes and the tag node, z is the measured distances between the anchor nodes and the tag node.

Finally, this difference, useful for the calculation of future positions, is subtracted from the range of the anchor nodes before being filtered.

The same data from the test done previously have been simulated with this modification. In this way, it is possible to see the better accuracy of the positioning determined.

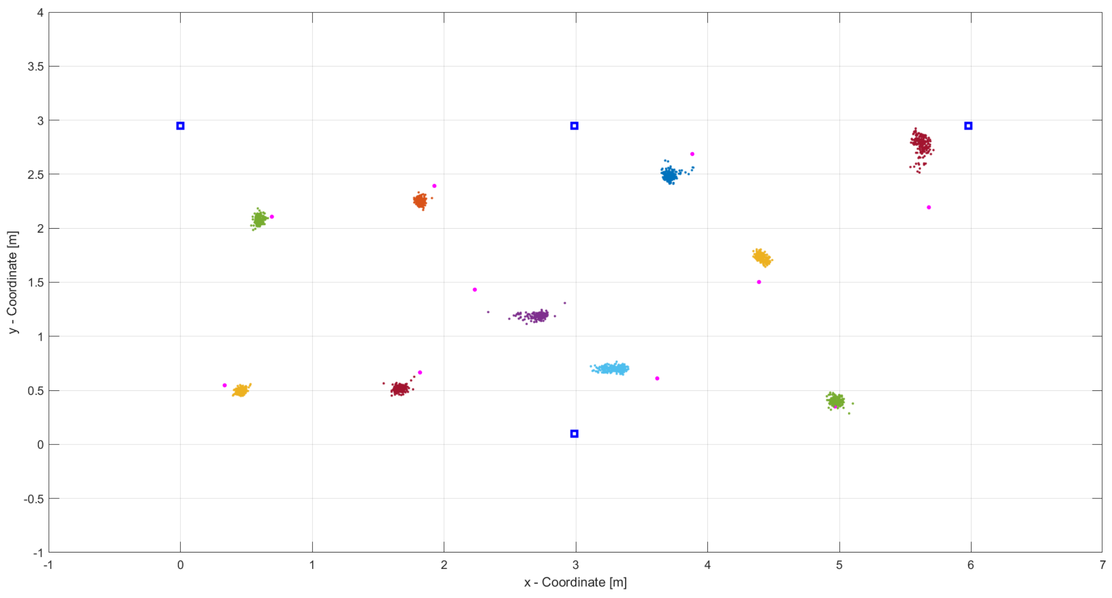


Figure 4.13: Plot of the final tests at flight altitude of 0.094 m to compare with the Figure 4.6

Average error	0.1393	0.1029	0.2157	0.1797	0.5382
	0.3396	0.2623	0.2258	0.0604	0.5861
RMS error	0.1408	0.1054	0.2179	0.1811	0.5408
	0.3454	0.2646	0.2279	0.0659	0.5905

Table 4.2: Error of 10 tests at flight altitude of 0.094 m

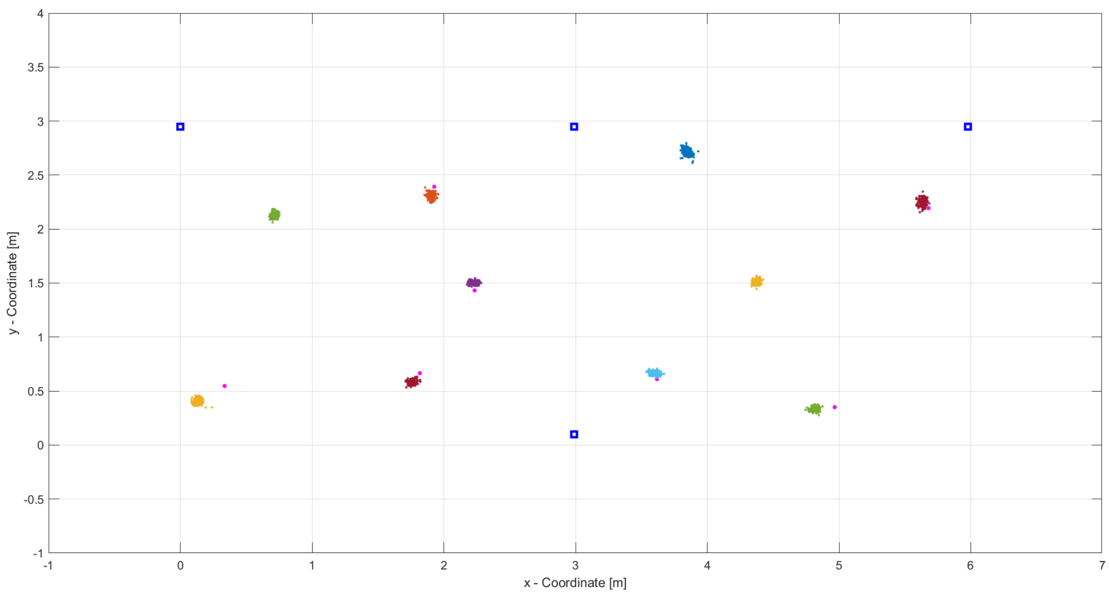


Figure 4.14: Plot of the final tests at flight altitude of 0.97 m to compare with the Figure 4.8

Average error	0.2464	0.0372	0.1048	0.0885	0.0734
	0.0620	0.0510	0.0287	0.1525	0.0746
RMS error	0.2473	0.0410	0.1065	0.0915	0.0747
	0.0649	0.0560	0.0318	0.1538	0.0786

Table 4.3: Error of 10 tests at flight altitude of 0.97 m

Average error	0.0883	0.0853	0.2433	0.0581	0.0450
	0.0895	0.0652	0.1365	0.2785	0.0823
RMS error	0.0900	0.0893	0.2483	0.0625	0.0482
	0.0911	0.0679	0.1377	0.3143	0.0870

Table 4.4: Error of 10 tests at flight altitude of 1.53 m

Average error	0.1008	0.0552	0.1582	0.0674	0.2502
	0.1249	0.0727	0.1230	0.1586	0.1059
RMS error	0.1296	0.0589	0.1598	0.0690	0.2507
	0.1255	0.0766	0.1243	0.1595	0.1180

Table 4.5: Error of 10 tests at flight altitude of 2.02 m

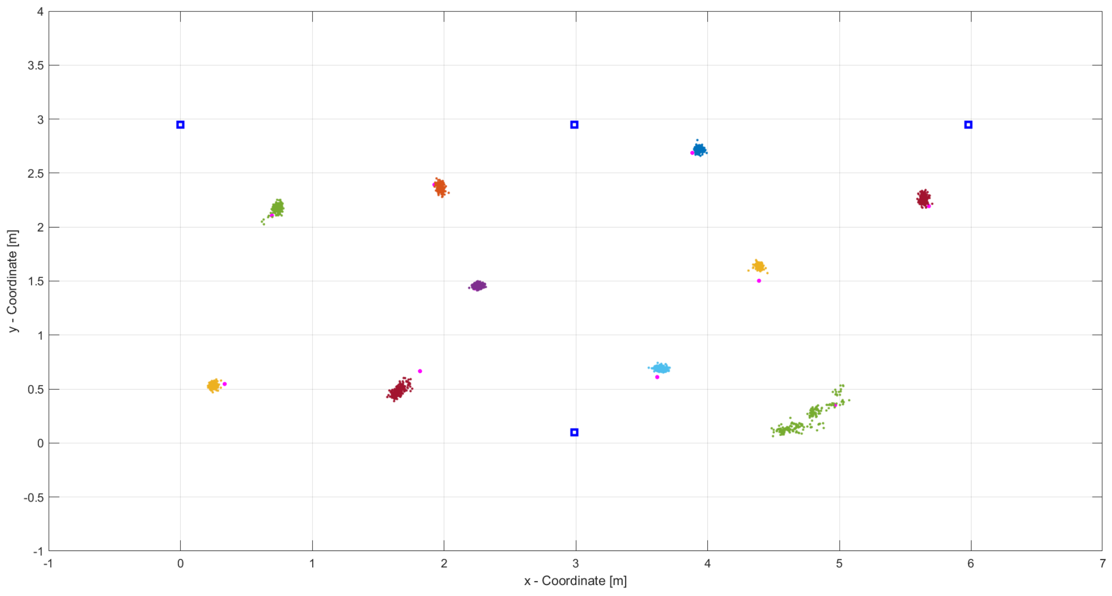


Figure 4.15: Plot of the final tests at flight altitude of 1.53 m to compare with the Figure 4.10

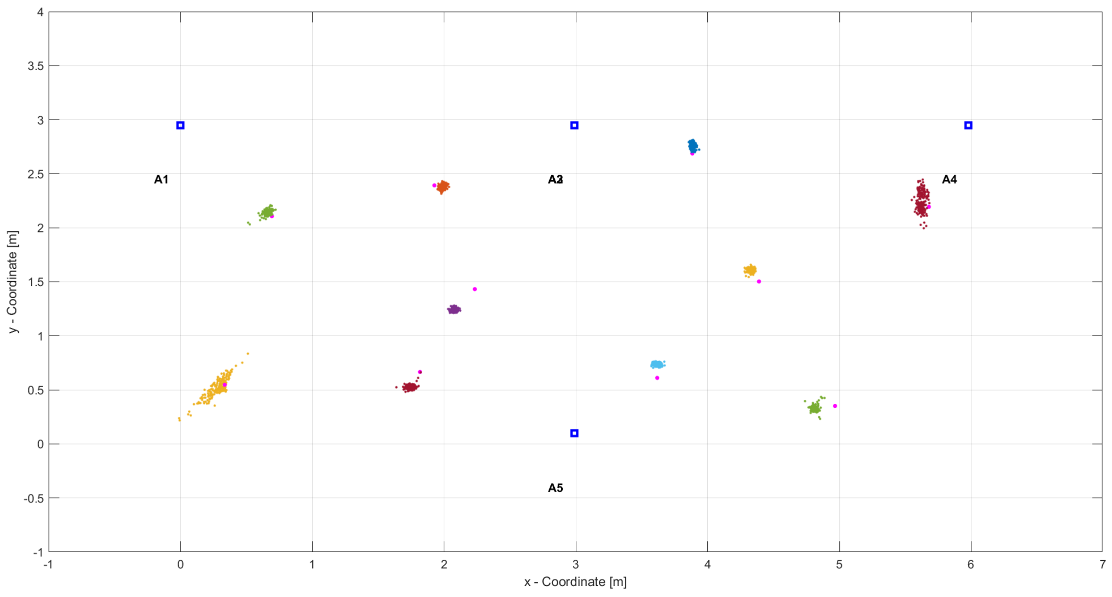


Figure 4.16: Plot of the final tests at flight altitude of 2.02 m to compare with the Figure 4.12

4.4 Indoor tests

Python code has been implemented on the drone and some real-life testings have been made.

Before testing the localization with a drone in flight, some indoor tests have been made with the drone placed on a MiR Robot. The sensor system on the drone determines its position while the robot is following a previously drawn trajectory. The tests simulate the same trajectory 20 times acquiring at each sample time the Ultra-WideBand signal.

The anchor nodes are arranged as in the configuration designed for the indoor flight cage, as in Figure 4.2.

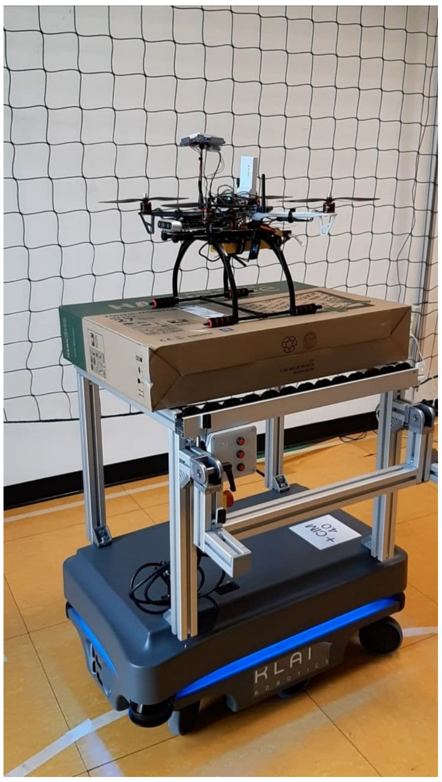


Figure 4.17: Drone positioned on MiR Robot in indoor environment.

4.4.1 First indoor test

A trajectory has been created with curves capable of covering a large part of the environment: 6 points are assigned to the MiR Robot by creating small segments. It should be noted that the MiR travels the shortest path between the start and end points avoiding obstacles.

The assigned points, listed in Table 4.6, have all been reached with the same height of 1.38 m, and there are no obstacles along the way. Therefore the final trajectory will be the result of a series of continuous breaks (Figure 4.18).

Point	**Coordinate**
A	(0.30, 1.50, 1.38)
B	(1.20, 2.00, 1.38)
C	(2.40, 1.50, 1.38)
D	(3.60, 1.10, 1.38)
E	(4.80, 1.40, 1.38)
F	(5.50, 2.00, 1.38)

Table 4.6: Coordinates(in meters) of the assigned points.

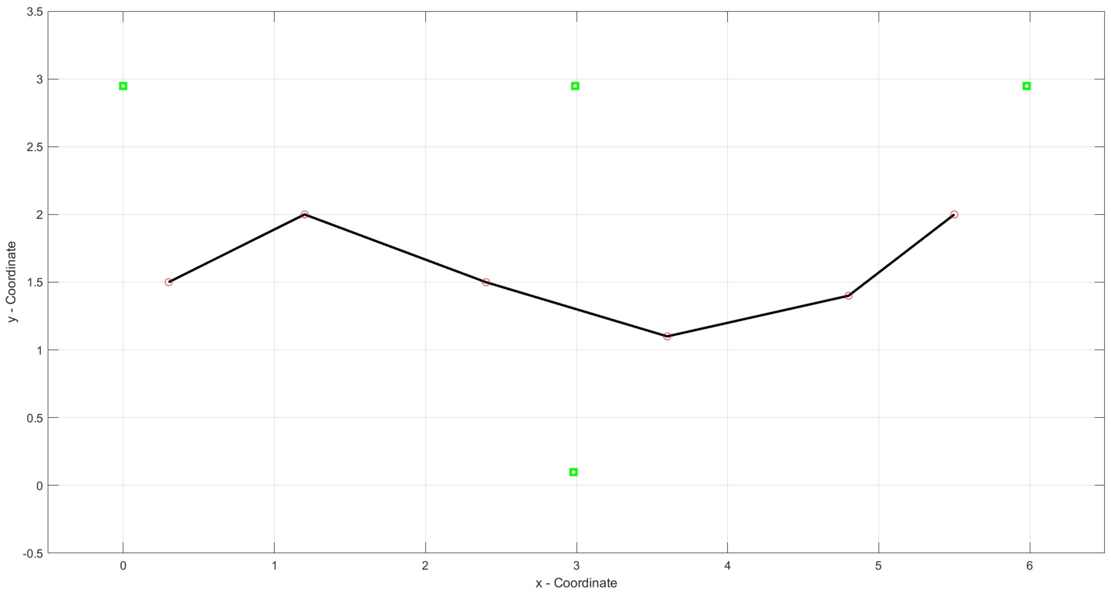


Figure 4.18: xy - Plane Representation of the trajectory

In the first test, the same path has been run 20 times to make an average estimate

$(Student'stdistribution)$. The results obtained can be considered good, but some observations must be made. Figure 4.19 shows three graphical representations of the trials performed in the XY plane.

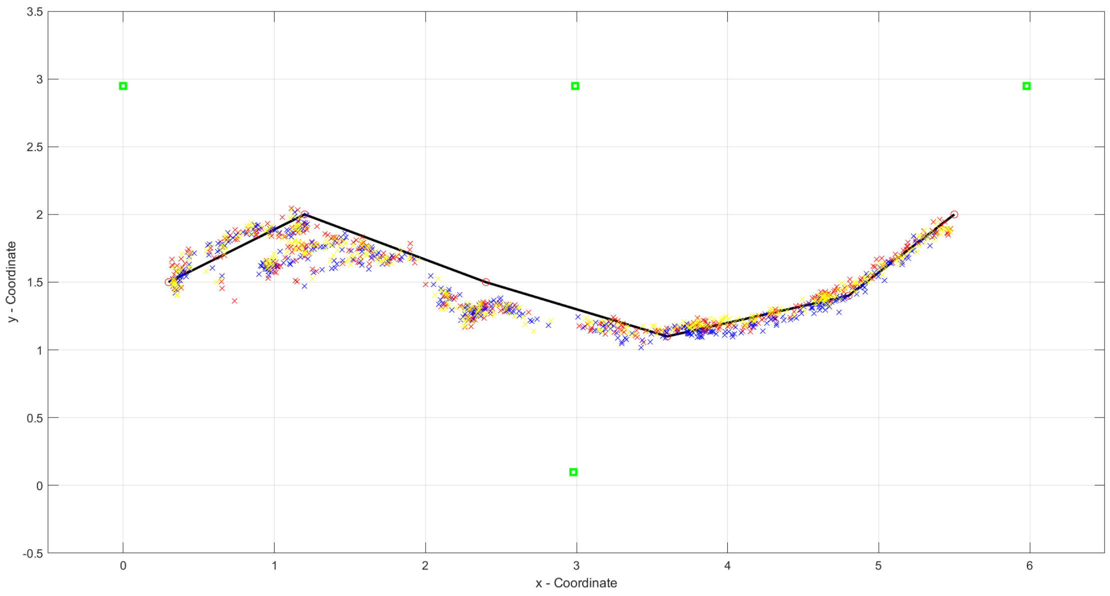


Figure 4.19: xy - Plane Representation of 3 Real Indoor Tests

The second part is much more precise than the first one: it should be noted that, during the tests, at point B the MiR Robot has an abrupt movement when changing direction. Therefore, some modifications have been made in the path making less evident the curve from A to D, lowering the y coordinates of points B and C.
During the travel on the Mir Robot, the drone oscillates affecting sensor stability.

As mentioned before, the entire system, composed of the drone and MiR Robot, travels the same path 20 times with the Decawave device always placed at the same height. Figure 4.20 shows the tests in the XZ plane. The height exiting by UWB and filter is not precise, thus leading to low stability and accuracy of the drone system.

4.4.2 Second indoor test

To solve this problem, a Time of Flight (TOF) sensor is needed to take the values obtained from this sensor as z coordinates of the system. Thus, the x and y coordinates from the filter and the z coordinate from the TOF sensor are taken to calculate location.

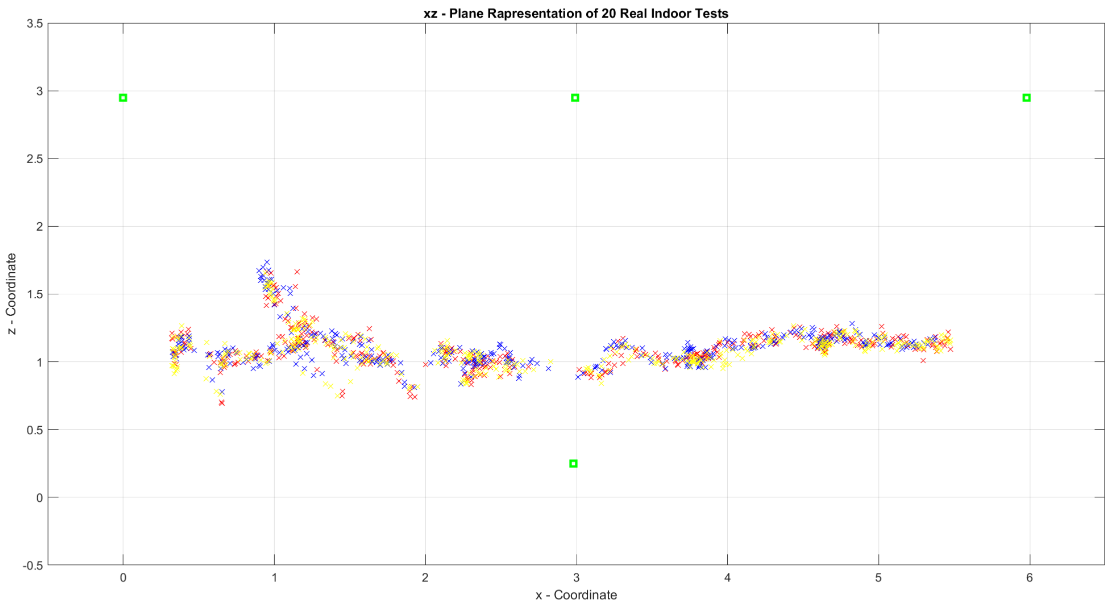


Figure 4.20: xz - Plane Representation of 3 Real Indoor Tests, same tests of Figure 4.19

After these improvements, tests have been made with these 6 points and this new trajectory, as shown in Table 4.7 and Figure 4.21):

Point	Coordinate
A	(0.30, 1.50, 1.38)
B	(1.20, 1.80, 1.38)
C	(2.40, 1.40, 1.38)
D	(3.60, 1.10, 1.38)
E	(4.80, 1.40, 1.38)
F	(5.50, 2.00, 1.38)

Table 4.7: Coordinates of the assigned points in meters

Also, in this case, 20 tests have been carried out with the tag node always placed at the same height. Figure 4.22 shows the plots of the 20 tests in the XY plane.

Figure 4.23 shows the representations of the 20 tests in the XZ plane:

The points close to the transition between the two cells are in an area where the position is a bit difficult to calculate. This is because in some cases the triangulation occurs in the left cell and in others in the right cell. However, the results are a good optimization.

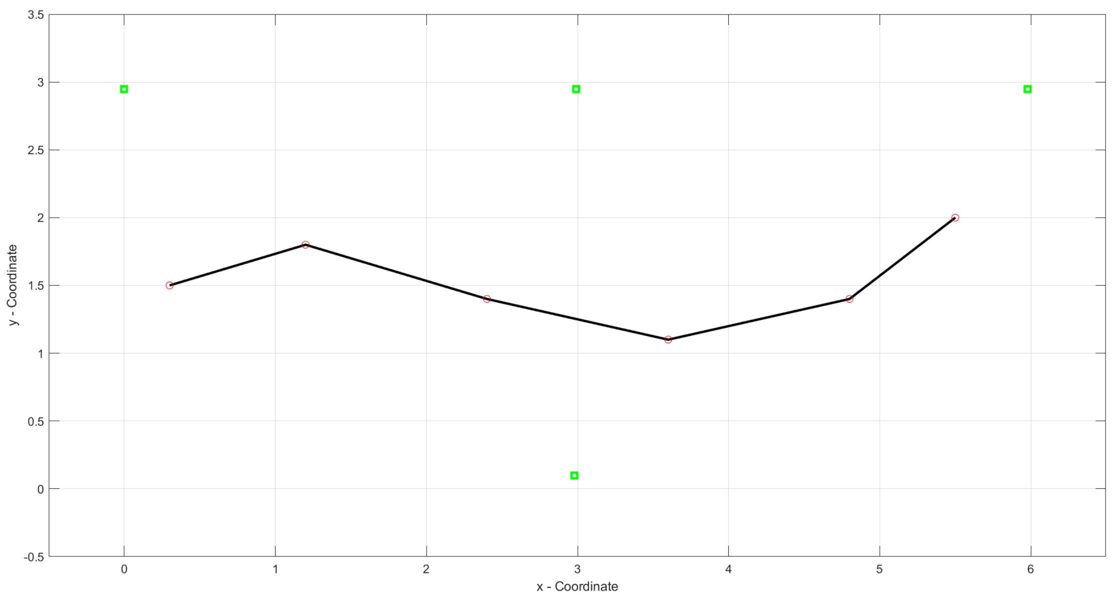


Figure 4.21: xy - Plane Representation of the new trajectory

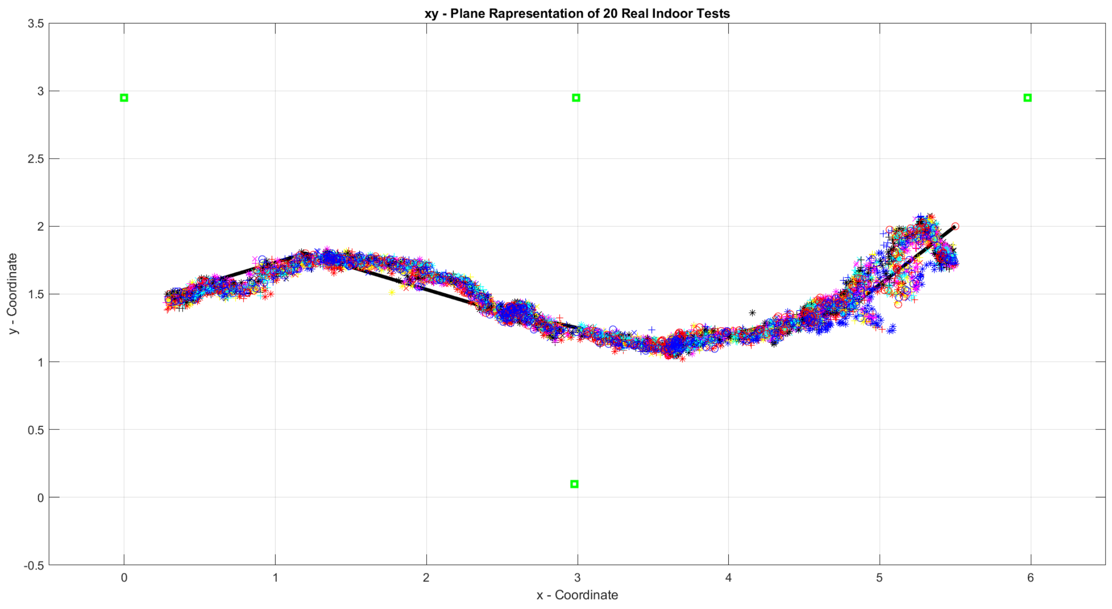


Figure 4.22: xy - Plane Representation of 20 Real Indoor Tests

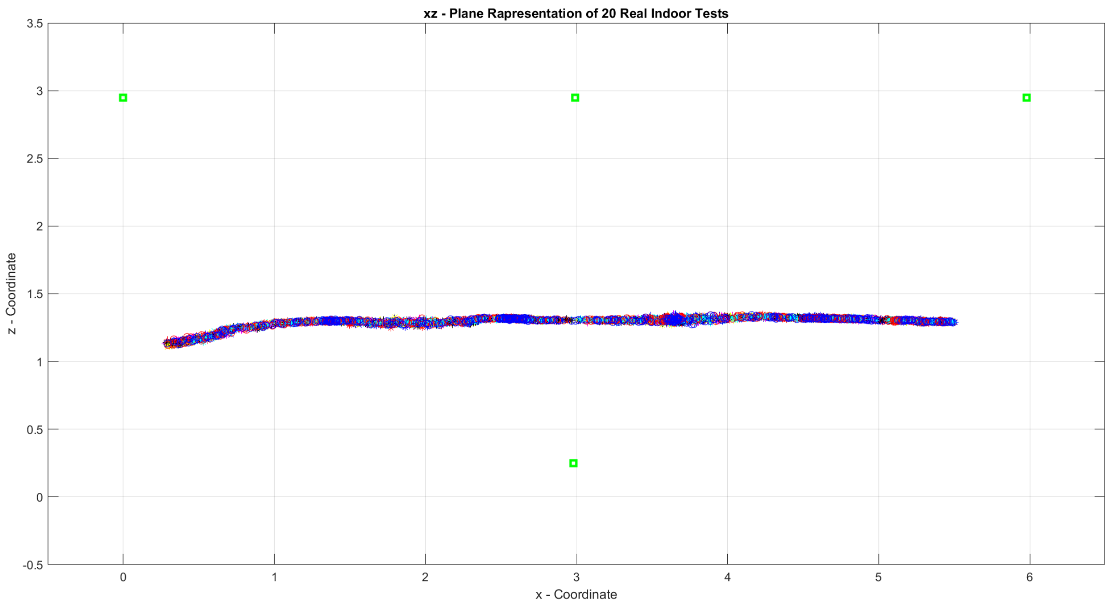


Figure 4.23: xz - Plane Representation of 20 Real Indoor Tests, same tests of Figure 4.22

Test	**avg err**	**rms err**
1	0.1821	0.2097
2	0.2086	0.2452
3	0.2051	0.2409
4	0.2029	0.2432
5	0.1971	0.2382
6	0.2646	0.3228
7	0.2174	0.2490
8	0.2187	0.2562
9	0.1844	0.2124
10	0.2043	0.2369
11	0.2151	0.2482
12	0.2352	0.2776
13	0.2398	0.2826
14	0.2033	0.2395
15	0.1683	0.2382
16	0.1883	0.2286
17	0.1983	0.2303
18	0.2268	0.2827
19	0.2116	0.2518
20	0.2057	0.2382

Table 4.8: Error of 20 Real Indoor Tests

4.5 Outdoor tests

In outdoor environment it is possible to use the GPS signal. So it is interesting doing a comparison between positioning from GPS and UWB. Two types of tests have been done: in the first one the same trajectory have been simulated 20 times acquiring each sample time the GPS signal; in the second one have been done the same path for 20 times having as localization system the Ultra-WideBand.

In the first test, the drone has been connected to the GPS signal and the positioning has been automatically calculated by the localization system. Instead, in the tests with the UWB, the anchor nodes were arranged as in the configuration designed for the indoor flight cage: the same anchor nodes in the same positions relative to the considered environment, as can be seen in Figure 4.24.

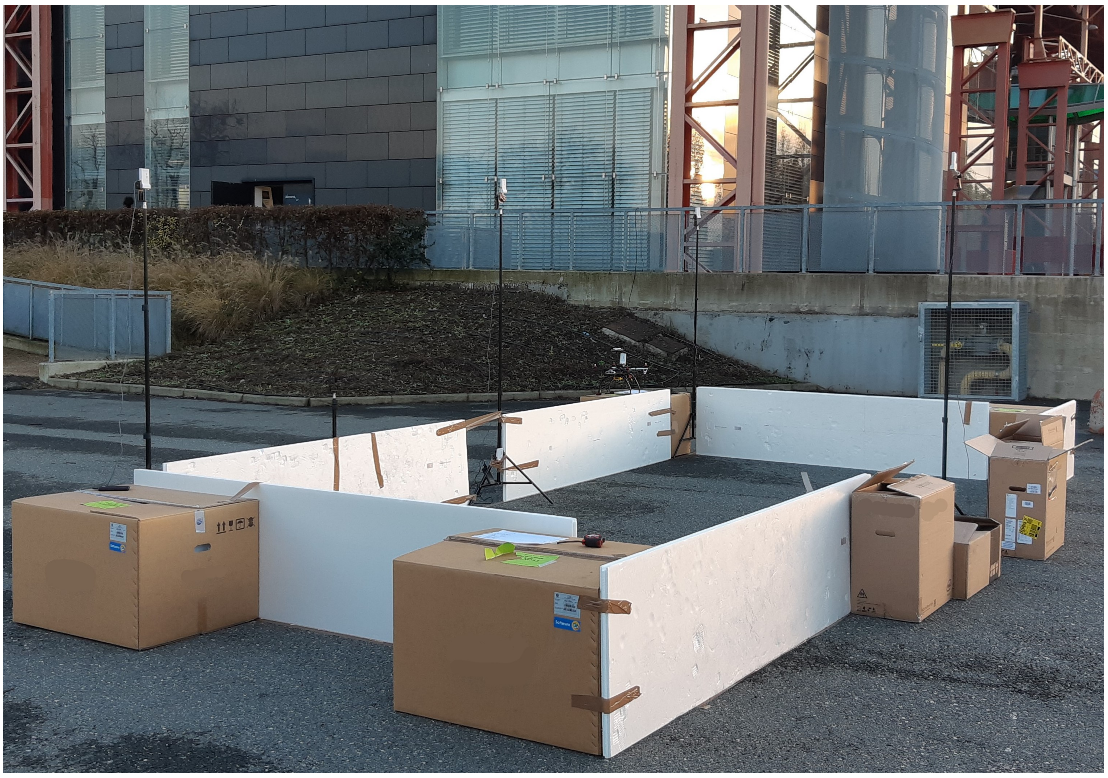


Figure 4.24: UWB configuration in outdoor environment

The trajectory of the tests done indoors has been used. 6 points (Table 4.7) are assigned to the MiR Robot, and the final trajectory is the result of a series of continuous segments since there are no obstacles along the way, as shown in Figure 4.21).

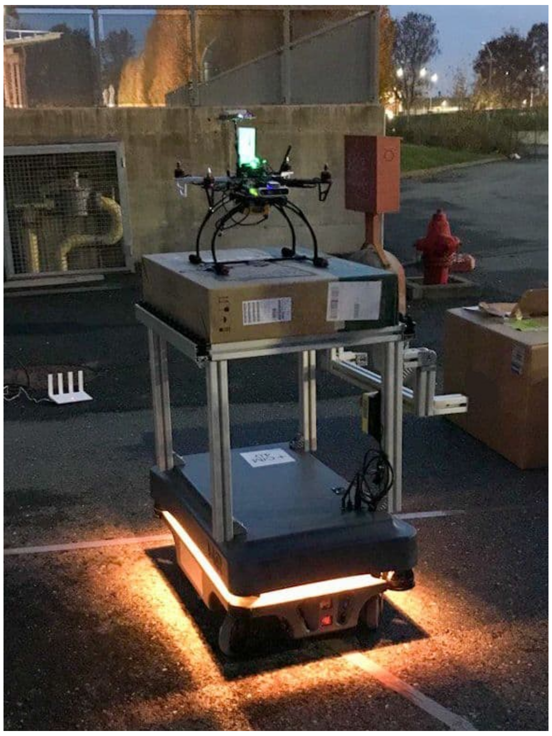


Figure 4.25: Drone positioned on MiR Robot in outdoor environment.

4.5.1 Outdoor tests with GPS

In the first test, cardboard boxes have been placed to create the boundaries of the mapping of the MiR Robot. The drone has been placed on the MiR Robot and the entire system traveled the route 20 times to create an average estimate ($Student's\ t\ distribution$). Figure 4.26 shows the plots of the 20 outdoor tests in the XY plane.

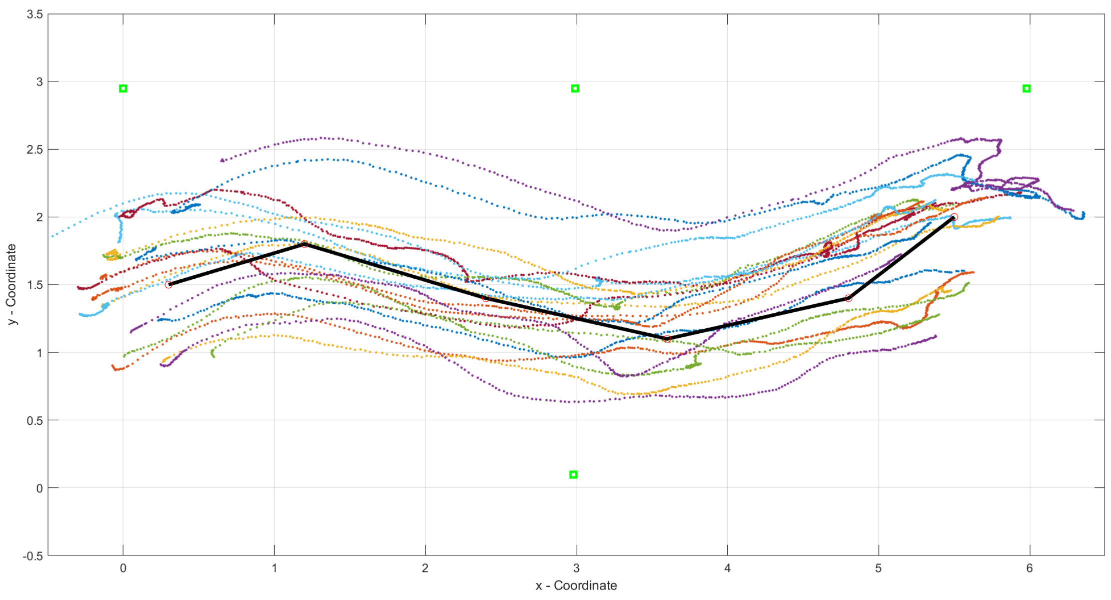


Figure 4.26: xy - Plane Representation of 20 GPS Outdoor Tests.

4.5.2 Outdoor tests with UWB

In the second test, the anchor nodes have been placed in tripods at the desired position and height creating the same configuration designed as in the indoor environment. The MiR Robot traveled 20 times following the desired trajectory to obtain an average estimate ($Student's\ t\ distribution$). Figure 4.27 shows the plots of the 20 outdoor tests in the XY plane.

Figure 4.28 shows the plots of the 20 outdoor tests in the XZ plane.

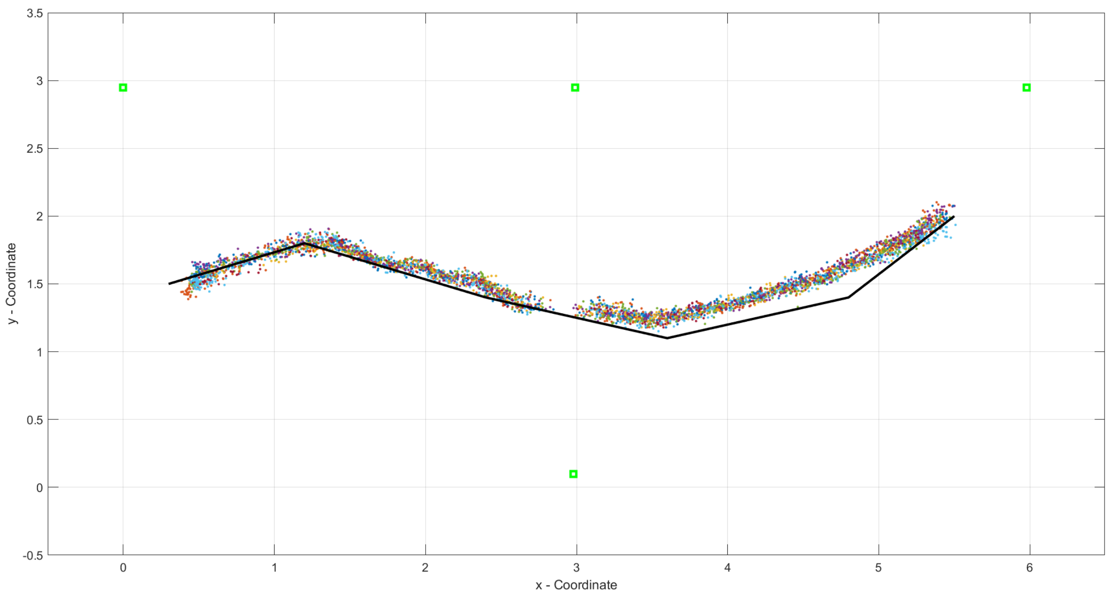


Figure 4.27: xy - Plane Representation of 20 UWB Outdoor Tests.

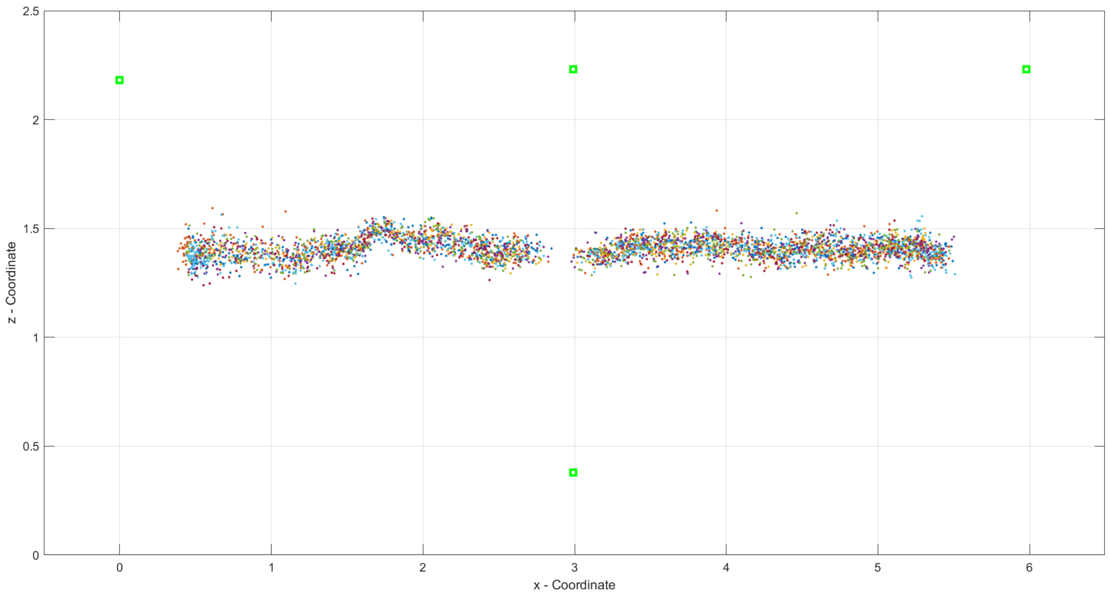


Figure 4.28: xz - Plane Representation of 20 UWB Outdoor Tests, same tests of Figure 4.27.

Chapter 5

Conclusion

Ultra-wideband is one of the most performing technologies to create an indoor localization system. It determines the position of the target node knowing the relative distance to the reference nodes.
The implementation of a Weighted Outlier-Robust Kalman Filter on UWB technology provides a performant indoor localization system. The filter decreases the disturbances derived from the calculation of the node position. It can estimate good results even in presence of obstacles between tag nodes and anchor nodes.

The simulations on Matlab show that the filter depends on the configuration of the anchor nodes. It is also necessary a correction during the calculation of the ranging between nodes due to a systematic error in the hardware of Decawave technology.

Real-life tests demonstrate the good stability and accuracy of the filter implemented on the Decawave technology in indoor environments.
Finally, the localization system obtained has been compared with the GPS, and it has been possible to see the strengths of the UWB technology also in outdoor environments.

5.1 Limits and future developments

In Gazebo environments an existing plugin has been used to simulate the position calculation from the ranging between tag node and anchor nodes in flight. It has been observed that there are some errors in the computation of the ranging. Consequently, the determination of the drone position is not accurate thus complicating the stability of the whole system.
The plugin can be modified to allow more articulated simulations, with obstacles and more drones in the same environment.

Another problem is given due to drone stability issues, which was not the focus

of this thesis. It has not been possible to test UWB localization with the drone in motion, but the experimental tests have been done only with the drone with its engines off.
The UWB technology is not susceptible to interference from the drone. Vibrations on a drone in flight can lead to a lower signal accuracy. During the tests, the MiR robot carrying the drone experienced oscillations, so no clear changes in the results are expected.

Bibliography

[1] DECAWAVE. *Quick-Start Guide for the MDEK1001.* URL: https://www.decawave.com/mdek1001/usermanual/ (cit. on p. 3).

[2] Peter H. Dana. «Global positioning system (GPS) time dissemination for real-time applications». In: *Real-time Systems* 12 (1997), pp. 9–40 (cit. on pp. 4, 5).

[3] Wikipedia. *Sistema di posizionamento globale.* URL: https://it.wikipedia.org/wiki/Sistema_di_posizionamento_globale (cit. on p. 5).

[4] Geotab. *What Is GPS?* URL: https://www.geotab.com/blog/what-is-gps/ (cit. on p. 5).

[5] Wikipedia. *Wi-Fi.* URL: https://en.wikipedia.org/wiki/Wi-Fi (cit. on p. 6).

[6] CISCO. *What Is Wi-Fi?* URL: https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html (cit. on p. 6).

[7] Thomas Lindner, Lothar Fritsch, Kilian Plank, and Kai Rannenberg. «Exploitation of Public and Private WiFi Coverage for New Business Models». In: *Building the E-Service Society.* Ed. by Winfried Lamersdorf, Volker Tschammer, and Stéphane Amarger. Boston, MA: Springer US, 2004, pp. 131–148. ISBN: 978-1-4020-8155-2 (cit. on p. 6).

[8] Chouchang Yang and Huai-rong Shao. «WiFi-based indoor positioning». In: *IEEE Communications Magazine* 53.3 (2015), pp. 150–157. DOI: 10.1109/MCOM.2015.7060497 (cit. on p. 6).

[9] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. «SpotFi: Decimeter Level Localization Using WiFi». In: *SIGCOMM Comput. Commun. Rev.* 45.4 (Aug. 2015), pp. 269–282. ISSN: 0146-4833. DOI: 10.1145/2829988.2787487. URL: https://doi.org/10.1145/2829988.2787487 (cit. on pp. 6, 7).

[10] Jie Yang and Yingying Chen. «Indoor Localization Using Improved RSS-Based Lateration Methods». In: *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*. 2009, pp. 1–6. DOI: `10.1109/GLOCOM.2009.5425237` (cit. on p. 6).

[11] u-box. *Bluetooth Indoor Positioning*. URL: `https://www.u-blox.com/en/technologies/bluetooth-indoor-positioning` (cit. on p. 7).

[12] Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. «Bluetooth positioning using RSSI and triangulation methods». In: *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*. 2013, pp. 837–842. DOI: `10.1109/CCNC.2013.6488558` (cit. on p. 8).

[13] DECAWAVE. *TECHNOLOGY*. URL: `https://www.decawave.com/technology1/` (cit. on pp. 8, 9).

[14] S. Gezici, Zhi Tian, G.B. Giannakis, H. Kobayashi, A.F. Molisch, H.V. Poor, and Z. Sahinoglu. «Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks». In: *IEEE Signal Processing Magazine* 22.4 (2005), pp. 70–84. DOI: `10.1109/MSP.2005.1458289` (cit. on pp. 8, 10, 11, 38).

[15] Luca De Nardis and Maria-Gabriella Di Benedetto. «Overview of the IEEE 802.15.4/4a standards for low data rate Wireless Personal Data Networks». In: *2007 4th Workshop on Positioning, Navigation and Communication*. 2007, pp. 285–289. DOI: `10.1109/WPNC.2007.353647` (cit. on pp. 12–14).

[16] Petr Sedlacek, Martin Slanina, and Pavel Masek. «An Overview of the IEEE 802.15.4z Standard its Comparison and to the Existing UWB Standards». In: *2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)*. 2019, pp. 1–6. DOI: `10.1109/RADIOELEK.2019.8733537` (cit. on pp. 15, 16).

[17] S. Gezici, I. Guvenc, and Z. Sahinoglu. «On the Performance of Linear Least-Squares Estimation in Wireless Positioning Systems». In: *2008 IEEE International Conference on Communications*. 2008, pp. 4203–4208. DOI: `10.1109/ICC.2008.789` (cit. on pp. 16, 18, 19).

[18] O. A. Stepanov. «Kalman filtering: Past and present. An outlook from Russia. On the occasion of the 80th birthday of Rudolf Emil Kalman)». In: *Gyroscopy and Navigation* 2 (Apr. 2011), pp. 99–110 (cit. on p. 19).

[19] Paul Zarchan. *Progress in astronautics and aeronautics: fundamentals of Kalman filtering: a practical approach*. Vol. 208. Aiaa, 2005 (cit. on p. 19).

[20] Eric Ghysels; Massimiliano Marcellino. *Applied Economic Forecasting using Time Series Methods*. New York, NY: Oxford University Press, 2018, p. 419 (cit. on p. 19).

[21] Daniel M Wolpert and Zoubin Ghahramani. «Computational principles of movement neuroscience». In: *Nature neuroscience* 3.11 (2000), pp. 1212–1217 (cit. on p. 19).

[22] Greg Welch, Gary Bishop, et al. «An introduction to the Kalman filter». In: (1995) (cit. on pp. 22, 23).

[23] S.J. Julier and J.K. Uhlmann. «Unscented filtering and nonlinear estimation». In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422. DOI: `10.1109/JPROC.2003.823141` (cit. on p. 23).

[24] Eric Wan. «Sigma-Point Filters: An Overview with Applications to Integrated Navigation and Vision Assisted Control». In: *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. 2006, pp. 201–202. DOI: `10.1109/NSSPW.2006.4378854` (cit. on p. 23).

[25] ROS. *What is ROS?* URL: `https://www.ros.org/` (cit. on p. 25).

[26] Pyo YoonSeak, Cho HanCheol, Jung RyuWoon, and Lim TaeHoon. *Ros Robot Programming*. ROBOTIS, Dec. 2017. ISBN: 9791196230715 (cit. on pp. 26–29).

[27] Gazebo. *Why Gazebo*. URL: `http://gazebosim.org/` (cit. on p. 29).

[28] Valentín Barral, Carlos J. Escudero, José A. García-Naya, and Roberto Maneiro-Catoira. «NLOS Identification and Mitigation Using Low-Cost UWB Devices». In: *Sensors* 19.16 (2019). ISSN: 1424-8220. DOI: `10.3390/s19163464`. URL: `https://www.mdpi.com/1424-8220/19/16/3464` (cit. on p. 36).

[29] DECAWAVE. *About Decawave*. URL: `https://www.decawave.com/product/mdek1001-deployment-kit/` (cit. on p. 37).

[30] DECAWAVE. *DWM1001C Module*. URL: `https://www.decawave.com/product/dwm1001-module/` (cit. on p. 37).

[31] DECAWAVE. *MDEK1001 Development Kit*. URL: `https://www.decawave.com/` (cit. on p. 38).