Costa Rica Institute of Technology

Mechatronics Engineering School

Autonomous Localization and Coordination of Tiny Aerial Robots relative to a Larger Aerial Platform

Graduation Project Report to Qualify for the Degree of

Mechatronics Engineer with the Academic Degree of Bachelor's

Isaac Méndez Barquero

Cartago, June 27, 2025

License

I declare that the present Graduation Project has been carried out entirely by myself, using and applying literature related to the topic and incorporating my own knowledge.

In the cases where I have used bibliographic sources, I have indicated them through the corresponding citations. Consequently, I assume full responsibility for the graduation project and for the content of the corresponding final report.

Isaac Méndez Barquero

Cartago, June 27, 2025

Céd: 1-1808-0219

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: Isaac Méndez Barquero

Proyecto: Autonomous Localization and Coordination of Tiny Aerial Robots relative to a Larger Aerial Platform

MSc. -Ing. Juan Carlos Brenes Torres

Asesor

27 de junio de 2025, Cartago

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: Isaac Méndez Barquero

Proyecto: Autonomous Localization and Coordination of Tiny Aerial Robots relative to a Larger Aerial Platform

Miembros del jurado evaluador

Ing. Ana Maria Murillo Morgan
Jurado

Ing. Ronald Loaiza Baldares
Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

27 de junio de 2025, Cartago

Resúmen

Se implementó una prueba de concepto para un sistema de localización relativa autónoma para cuadricópteros con respecto a una plataforma aérea mayor, así como un algoritmo de planificación. Aunque no fue completamente validado, se obtuvieron resultados prometedores y se sentaron las bases para una continuación del proyecto. Además, se siguió una metodología de diseño de ingeniería para seleccionar los conceptos a desarrollar según las necesidades del cliente y su importancia relativa.

Palabras clave: drones, robots móviles, filtro de Kalman extendido, robótica en enjambre, localización relativa autónoma, sistema operativo para robots

Abstract

A proof of concept for an autonomous relative localization system was implemented for quadcopters with respect to a larger aerial platform, as well as a scheduling algorithm. Although not completely validated, promising results were achieved, and ground was laid for a continuation on the project. Furthermore, an engineering design methodology was followed to select the concepts to develop based on the client's needs and their relative importance.

Keywords: drones, mobile robots, extended kalman filter, swarm robotics, autonomous relative localization, robot operating system

To my parents, and my life-partner Angie

Acknowledgements

Special gratitude to my parents who have supported me throughout all of my career. Also to FRPG to welcome me in their laboratories. And also my university and school to have allowed me to study this major that I am so passionate about.

Isaac Méndez Barquero

Cartago, June 27, 2025

Contents

List of Figures

List of Tables

List of Abbreviations

ROS Robot Operating System

UAV Unmanned Aerial Vehicle

EKF Extended Kalman Filter

GPS Global Positioning System

IMU Inertial Measurement Unit

LST Least Slack Time

EDF Earliest Deadline First

LCT Least Completion Time

SJF Shortest Job First

FCFS First Come First Served

UWB Ultra-Wideband

Chapter 1

Introduction

In order to understand the project, it is necessary first to present its background to justify its existence. Therefore, in this chapter, the background, the description of the problem to solve, and the objectives of the project are presented. In the end, a general description of this document is also written to prepare the reader.

1.1 Project background

The Flight Robotics and Perception Group (FRPG) is a research group of the Institute of Flight Mechanics and Controls of the University of Stuttgart, Germany, that develops deep-learning and reinforced-learning methods for flying robots and multi-robotic systems [5]. This group focuses on robots' perception, for which they develop state estimation methods for the environment through communication among the robots and sensors fusion like cameras, Inertial Measurement Units, wind speed sensors, among others [6].

WildCap is a project within the FRPG which consists of the development of a vision-based system that is capable of detecting, autonomously tracking, and following wild animals to estimate their pose and shape for environmental studies [7].

This system consists of airships and multirotor drones; specifically, these drones have a very limited battery life and therefore cannot fly for extensive time periods in the wilderness without having to land and charge them. However, the airships have bigger energy storage capacity.

1.2 Problem to solve

In order to study the animals in their natural habitats, it is necessary to have drones available for long periods of time, however, these drones have a limited energy capacity corresponding to their batteries, making it hard for them to be in the air for long. Nonetheless, these drones are actually accompanied by an airship with bigger energy capacity, creating the possibility of

using it to charge their batteries.

Battery charging would be possible through an electromagnetic coupling, which must be highly precise. In spite of the robots having sensors like GPS to have absolute, precise pose estimation in air, they cannot locate each other in a precise manner, that is, they don't have a relative pose estimation.

Besides that, the amount of charging ports is small compared to the number of drones on the area in a given moment. For this reason, apart from being able to autonomously coupling and charging, they must be able to coordinate and charge according to who needs it more at the moment, and not everyone at once.

Therefore, in this project a relative localization system will be developed, that will allow the drones to estimate their pose, precisely, with respect to the charging platform. This will be the base to develop a coordination algorithm that will optimize the drones' charging schedule according to predefined parameters. This way, this project includes several mechatronics applications: sensors and computer vision for relative localization and robotics for control and coordination of the mobile robots.

1.3 Problem synthesis

The need for the drones used in the WildCap project to be able to estimate their pose relative to a larger aerial platform, in a precise and autonomous manner, and also, to be able to coordinate going to the platform according to previously established priorities.

1.4 Objectives

1.4.1 General objective

Design a system of autonomous, relative localization and a coordination algorithm for a group of drones relative to a larger aerial platform of the WildCap project from the Flight Robotics and Perception Group of the University of Stuttgart.

1.4.2 Specific objectives

- Define the state of the art of the available methods for relative pose estimation and coordination algorithms for mobile robots.
- Develop the program for the drones' autonomous, relative localization, validating it both in simulation and hardware.

- Implement the coordination algorithm for the drones to be able to organize themselves to head to the aerial platform following an order priority.
- Validate the system in a less controlled environment that the laboratory, that is, in an external environment where there is no lab-dependant sensors, like infrared cameras.

1.5 Document overview

The next Chapter (2) of the document introduces the necessary theoretical background to understand the decisions taken in the different steps of the design methodology, which is explored in Chapter 3. With the design methodology, an engineering solution is proposed, which is developed in Chapter 4, and its results and analysis are outlined in Chapter 5. Finally, in Chapter 6 the conclusions of the project and further recommendations are discussed.

Chapter 2

Theoretical Framework

In this chapter, the concepts needed to understand the design and implemented solutions are described. It starts with an overview of robotics, with a specific section on swarm robotics. This is connected to sections on the Kalman Filter, visual tag detection, and processes scheduling.

2.1 Robotics

In this section, the definition, types of robotics and usage as a swarm is explored. This is relevant to further understand the context of the project.

2.1.1 Definition

Defining a robot is not an easy task, and it might even be a philosophical question. Its definition will depend on the context of application, so there are many definitions in academia and industry. For instance, according to ISO [8], a robot is a "programmed actuated mechanism with a degree of autonomy to perform locomotion, manipulation or positioning", where autonomy is defined as the "ability to perform intended tasks based on current state and sensing, without human intervention".

But IEEE [9] defines a robot as an "agentive device in a broad sense, purposed to act in the physical world in order to accomplish one or more tasks. In some cases, the actions of a robot might be subordinated to actions of other agents, such as software agents (bots) or humans. (...)". And ASTM [10] defines it as a "mechanical system designed to be able to control its sensing and acting for the purpose of achieving goals in the physical world".

These definitions have differences, like ISO using autonomy on its definition, or ASTM defining it as a mechanical system, but what they all have in common is that a robot is an agent that senses the world and acts on it to perform tasks. This definition will be used throughout this project.

Now that a robot has been defined, a robotics definition can be built. Once again, ISO [8] defines robotics as "science and practice of designing, manufacturing, and applying robots", which corresponds to a broad-enough definition for this project.

2.1.2 Types

There are many classifications available. ISO focuses on the application [8]:

- Service robot: These perform useful tasks for humans or equipment but does not include industrial applications like automation.
- Mobile robot: It can travel under its own control. Can be a moving platform with or without manipulators.
- Household robot: This is an actuated mechanism with a degree of autonomy, operating within a household or a similar environment.
- Industrial robot: Manipulator in three or more axes. It is autonomically controlled, reprogrammable and multipurpose. Can be fixed in place or mobile.
- Collaborative robot: In a collaborative workspace, this robot directly interacts with a human.
- Medical robot: This is intended to use as a medical electrical equipment or system.

In the other hand, IEEE focuses on how they are controlled [9]:

- Automated: A robot performs a task as an automation. It does not adapt to changes in the environment and/or follows a scripted plan.
- Fully autonomous: The robot solves a task without human intervention while adapting to external conditions.
- Remote-controlled: The robot has no initiative, but relies completely on a human operator, who controls it from a location off the robot.
- Semi-autonomous: Both a robot and a human operator plan and conduct the task. There are various levels of human interaction.
- Teleoperated: The robot performs a task in which a human operator uses sensor data and controls the actuators or assign incremental goals continuously. This is done from a location off the robot.

Finally, ASTM also classifies them according to their application [10]:

- Emergency response or response: Intended to perform tasks to assist the operator with handling the involved task.
- A-unmanned ground vehicle: Autonomous vehicle that operates while in contact with the ground without a human operator.
- Exoskeleton: Wearable that augments, enables, assist and/or enhances physical activity through mechanical interaction with the body.

In the context of the project, the classification that apply is that of a mobile, fully-autonomous robot.

2.1.3 Mobile robots positioning

As mobile robots are often moving through places they do not know, they need a strong positioning system. For this, there are different approaches, and there are new ones being researched. [11] classifies them as

1. Relative position (dead-reckoning)
    1.1. Odometry
    1.2. Inertial navigation
2. Absolute position measurements
    2.1. Magnetic compasses
    2.2. Active beacons
    2.3. Global positioning systems
    2.4. Landmark navigation
    2.5. Model matching

Odometry

This is the most used technique due to the sensors' price, and because it provides good short-term accuracy. However, it is based on cumulative position data, i.e. the position data at a given time depends on the past measurements. This also means that there is an unbound accumulated error [11].

Odometry is based on the translation of wheel motion to linear motion through basic cinematic equations that relate the wheel circumference and the rotational speed. The issue with this is that slippage, among other phenomena, is not usually included in the equations because they are hard to model. Because these small errors accumulate over time is that this method is not useful on its own over long periods [11].

Inertial navigation

Also called inertial odometry or even interchanged with simple odometry. It uses gyroscope and accelerometer data, which are integrated once (or twice, for the latter). This positioning system has the advantage of not needing external reference, but works on its own. Nonetheless, it has the same problem as before, that because of integration, the error accumulates over time, making it increasingly unreliable for long-term estimation without correction [11].

Magnetic compasses

The heading of the robot is the positioning variable more sensible to increasing error in dead-reckoning. That is why an absolute heading positioning sensor like a magnetic compass is so useful to correct the odometry measurements. The main disadvantage is that Earth's magnetic field can be distorted by power lines or electronic devices, mostly indoors [11].

Active beacons

Active beacons are precisely installed and provide high precision with low latency, making it the go-to method for ships and airplanes. There are two types: *trilateration*, in which the robot measures the distance to the beacons, and triangulation, in which the robot measures the angle to the beacons with a rotary sensor. In both cases there must be at least three beacons, and the main disadvantages of this method are that it is costly and that the beacons have to be placed with high precision [11].

Global Positioning System (GPS)

This system uses trilateration, but it uses 24 satelites as the beacons. They send radio frequency (RF) signals and the ground receivers caculate the distance to the satelites from the travel time of the signal [11].

Landmark navigation

Landmarks are characteristic sensory information, usually they can be geometric shapes, and they can contain information like in bar-codes. These have a fixed, known position relative to wich a robot can locate itself. They also have to be carefully chosen, for example, with high contrast with the background [11].

The landmarks information has to be previously stored in the robot's memory, so the main task of this method is to recognize accurately the landmark and calculate the robot's position. It is often assumed that the robot's pose is known, so that only a partial area of the landmark needs to be identified. For this reason, accurate odometry positioning is used together [11].

Artifical landmarks are easier to recognize because they are custom designed with specific geometries and optimal contrast. They are typically based on computer vision. The accuracy of the detection depends on how accurately are the geometric properties of the landmark extracted—this, at the same time, depends on the position and orientation of the robot with repect to the landmark [11].

One such an example is the AprilTag which is a fiducial visual system used, among other applications, in robotics. They have the advantage of being printable with a common printer, and the AprilTag software computes the precise 3D position, orientation and tag identification relative to the camera, Also, the tag detection software is optimized enough to be executed even on cell-phones [12] [1] [13].

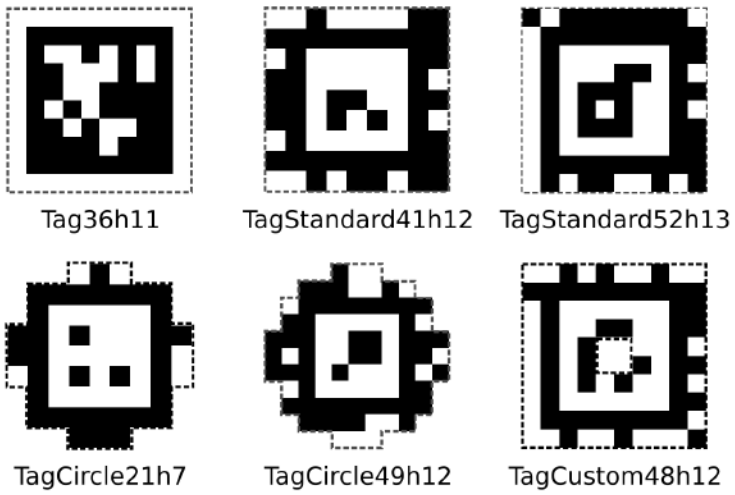

Figure 2.1: Examples of tag families possible with AprilTag 3, with dotted lines showing the tag borders. a) The old-style tag family from AprilTag 2. b) and c) Tags with one layer of bits outside of the border with 41 and 52 data bits respectively. d) and e) Circular tags with 21 and 49 bits respectively. f) Tag with empty space in the middle for recursive tag applications [1].

Map-based positioning

This is also known as "map matching" because the robot creates a local map based on its sensors data and compares it to a global map stored in the memory. If a match is found, then the robot calculates its position and orientation [11].

2.1.4 Swarm robotics

A robot can be used alone or in groups as a multi-agent system. An individual robot can typically be used for tedious or dangerous tasks for humans, like manufacturing or a specific task. However, there are challenging scenarios where deploying several robots can be more

beneficial than using just one, and that they behave cooperatively [14].

The difference between just a multi-agent system and a swarm lies in the cooperative behavior of the robots. In the first, the robots behave independently doing tasks—say, a production line with several robots that do different tasks, but they do not cooperate, they just do their individual task without caring for what the others are doing. A robotic swarm will exhibit animal-like behavior, specifically those of swarms like ants or bees.

Many animals cooperate to do their day-to-day tasks, since it would be difficult for them to do it on their own. For example, ants colonies send scouts to look for food, but as small as they are, when they find it, they cannot usually bring it on their own. So the scout leaves a pheromone trail as it comes back to the colony so that the rest of the workers can follow it and bring it back together [15]. In this way, a single ant is not very useful, multiple ants acting independent is also not very useful, but a group acting together, making decisions cooperatively lets them survive.

Applying this to robotics, according to [14], a "robot swarm is a group of three or more robots that perform tasks cooperatively while receiving limited or no control from human operators".

2.2 Kalman Filter

As previously mentioned, the mobile robots can use many different localization methods, but every one of them has disatvantages, mainly drifting or unstable signals in different scenarios. Basically, no method can be used on its own—if GPS is used, there might be a place where it does not have a signal of three satelites; if odometry is used, it will drift over time; if a magnetometer is used, it could have interference by other electronic devices.

This is one perfect application of the Kalman Filter (KF). The KF is an optimal state estimation algorithm, and it is used in scenarios similars to the one mentioned, but also for indirect measurements. Applications include guidance and navigation systems, computer vision and systems and signal processing [2]. Figure 2.2 is a usage example.

To understand the KF, in Figure 2.3 a system of a car is presented with input speed and output position. The car dynamics outputs the real-world measurement. The measurement has noise v, for example, from the GPS, and the process also has noise w, for example, from wind or velocity changes. These two noises are assumed to behave as follows [3]:

$$v \sim N(0, R)$$

$$w \sim N(0, Q)$$

This means that they behave like a Gaussian distribution, centered on mean zero, with a covariance R and Q, respectively.

Figure 2.2: Usage example for Kalman Filter [2].

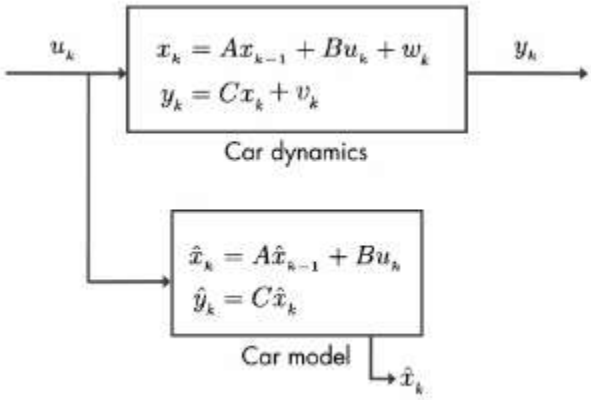

Figure 2.3: System model with a measurement subsystem and a prediction subsystem [3].

In Figure 2.4 it can be seen that at first, there will be an estimation $\hat{x}_{k-1}$ with small variance, but the next estimation $\hat{x}_k$ will have more variance because of the noise. In this new iteration, there will be a measurement y_k, so through a KF, by multiplying the prediction and the measurement, an intersecting optimal state estimate is obtained, which is a combination of the measurement and the prediction, and has a smaller variance than both [3].

The KF algorithm is represented by two steps: prediction (2.1) and update (2.2). On the first step $\hat{x}_k^-$ represents the *priori* state estimate, the one before measurement. P_k^- is the error covariance and is a measurement of uncertainty in the estimated state. It comes from the process noise and propagation of the uncertainty of $\hat{x}_k^-$. At first, they come from their initial estimates $\hat{x}_{k-1}$ and P_{k-1} [16].

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{2.1}$$

The second step of the algorithm uses the *priori* estimates and updates them to estimate the

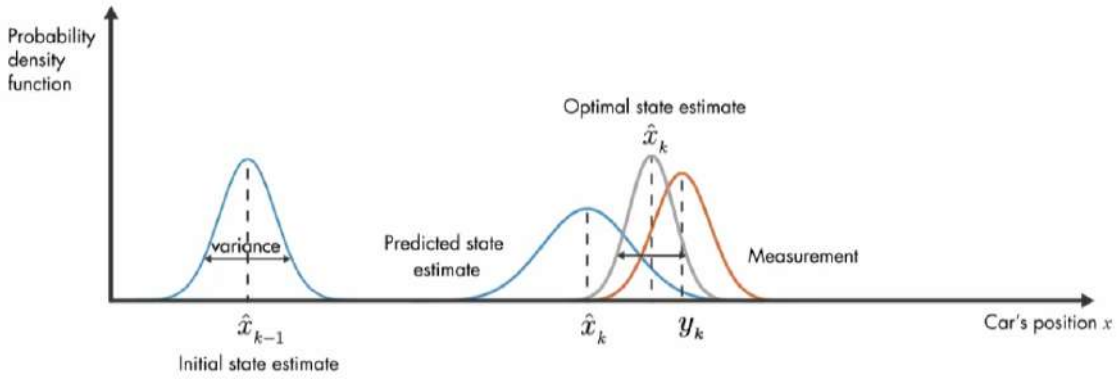

Figure 2.4: Visual statistic explanation of Kalman Filter [3].

posteriori estimates $\hat{x}_k$ and P_k. The Kalman gain K_k is calculated such that it minimizes the *posteriori* error covariance. These two steps are repeated in a loop indefinitely [16].

$$\begin{aligned} K_k &= \frac{P_k^- C^T}{C P_k^- C^T + R} \\ \hat{x}_k &= \hat{x}_k^- + K_k \left(y_k - C\hat{x}_k^-\right) \\ P_k &= (I - K_k C) P_k^- \end{aligned} \tag{2.2}$$

Chapter 3

Methodology

In this project, the engineering design methodology of Ulrich and Eppinger [17] was used. Even though the authors in their book present this methodology for products design, their presented steps for design are easily adaptable for more general engineering solutions. Furthermore, this methodology was chosen because it gives a series of guidelines that help the engineer make design decisions systematically and as unbiased as possible, prioritizing the client's needs and specifications, instead of the personal experience and likes of the designer.

In this chapter, there is a section for every step of the design methodology, explaining rigorously the thinking behind the decisions taken, and most importantly, presenting the arguments and evidence that make them objective.

3.1 PHASE 0: Problem definition and ANAFI drone's especifications

This phase is called "zero" because it happens before starting the project, and for this reason it is presented in Chapter 1. The problem definition comes from the first communications with the client, in this case, through emails and virtual meetings with Jun.-Prof. Dr.-Ing. Aamir Ahmad, the coordinator of FRPG.

After the first meetings, when the project is delimited, it becomes essential to do an analysis of the given information in order to discard biases like the two most common: "perceived problem", where a symptom of the actual problem is perceived as the problem, and the confusion between problem and solution technique, where a specific technology or technique is communicated as the desired solution, without having done the necessary methodology yet. After this analysis was done, the problem in Chapter 1 was defined.

3.1.1 ANAFI drone's especifications

The specific drone that's used in the WildCap project is the Parrot ANAFI Drone. To design a solution around this particular drone, it is first necessary to list its characteristics so that later in the design process they can be taken into account. They are shown in Table 3.1.

Table 3.1: ANAFI Drone's main characteristics [4].

Component	Specifications
Drone	• Size: 244x67x65 mm • Weight: 320 g • Max transmission range: 4 km with controller • Max horizontal speed: 15 m/s • Max vertical speed: 4 m/s
Battery	• Type: Lipo • Capacity: 2700 mAh
Imaging system	• Sensor: 1/2.4" CMOS • Lens aperture: f/2.4 • Lens focal length: 35 mm • Lens depth of field: 1.5 m • Video resolution: 4K UHD 3840x2160 24/25/30 fps or FHD 1920x1080 24/25/30/48/50/60 fps • Video HFOV: 69°
Sensors	• 6-axis IMU (gyroscope and accelerometer) • Magnetometer • Barometer • GPS • Ultrasonar (height measurement) • Vertical camera (measuring horizontal speed and height using optical flow)

3.2 PHASE 1: Determination of requirements

The problem statement is just a general overview of what the client requires, nonetheless, the actual solution requires certain aspects to be accomplished, based on the client's wishes, to deem the solution valid. This is why the **requirements** or needs must be properly defined, as they will help validate the solution later on.

Following the proposed methodology's four steps to define the requirements:

3.2.1 Step 1: Recollection of unprocessed data

Following the last phase, the communication with the client continues, in this case, through multiple conversations with a stakeholder, the PhD student Pascal Goldschmid, who is working directly with the autonomous drones' charging. This form of communications is preferred because it is straight forward to collect the information and define the client's expectations.

Also, these talks were not very formal because of similar reasons like straight-forwardness and time optimization, but it's important to consider that as the stakeholder has a Bachelor's and Master's in an engineering degree, the discussions were able to be highly technical, and concrete delimitations and expectations were set.

3.2.2 Step 2: Interpret unprocessed data in terms of client's needs

Based on the results of the conversations with the client, the requirements shown in Table 3.2 are defined.

Note that the amount of needs identified is balanced—neither too many (more than twenty), nor too few (less than ten). For this reason, further classification of the requirements is not necessary. Furthermore, they have a proper independency level, i.e. not similar enough to be worth it grouping them.

3.2.3 Step 3: Establishing requirements' relative importance

As it is not timely and economically feasible to meet all the requirements with the same level of importance, getting to know how important each individual need for the client is becomes an important step of the process.

The same approach of informal conversations was used for the same reasons as before. The list with the identified needs was given to the client with the instruction to write in each of them how important its accomplishment is, using a five for the absolutely essential (i.e. must be achieved) and a one for the ones where it is not important if they are taken into account in the design.

The resulting requirements with their importance level are shown in Table 3.2. It is noted that there are three instances with a rating of one. This is because these attributes were initially proposed to the client as potential features to simplify the design. However, the client ultimately views them as non-essential, meaning their fulfillment is not critical.

Table 3.2: Project's requirements with their relative importance.

Number	REQUIREMENTS	Importance
1	The relative localization is precise	5
2	The localization uses the existing hardware	2
3	The localization is fast	4
4	The localization of the larger platform in respect to the drone is relative	5
5	The localization is resilient against signal interruptions	4
6	The coordination is centralized	1
7	The coordination is simple	1
8	The coordination is not interdependent	1
9	The coordination works with respect to previously defined priorities	4
10	The algorithm is efficient	3

3.2.4 Step 4: Overview

As the methodology is rather long, an important step in every phase is to do an overview of the previous steps, so that any bias or misjudgment can be traced down easily and so that they do not affect the next phases.

In the last two phases, the problem proposed by the client was properly determined, without biases, as well as the requirements with their needs. Moreover, many conversations in-person were held, with which it is considered that a good enough amount of information was collected.

The information was filtered to state the requirements adequately, as desired attributes of the system. It is also important to note that many revisions of the requirements were needed to establish the expectations properly.

3.3 PHASE 2: System's specifications

Once the requirements have been determined, it is necessary to establish their level of completion, in other words, there must be a measurement with which the client can tell "this design accomplishes what I want" or "it does not". This is why in this phase the **specifications** are defined, which are statements that enable the measurement of how much are the client's needs

being met.

These specifications must refer to the overall designed system, not to a specific solution method. Furthermore, they have to be concrete, measurable, and are composed of a **metric**—a measurable quantity, and one or more **goal values**, which correspond to the values the design must meet to verify to what level are the client's requirements being met.

In order to determine the specification, the following steps were followed:

3.3.1 Step 1: Metrics list

In this step, each requirement has to be directly associated with the specifications, i.e. "how am I going to measure this requirement?". This list must be complete, that is, there can be no need without a metric, and they also have to be practical because they must be able to be properly measured. Moreover, these metrics can be subjective, in which case they are not easily quantified, and have to include standardized criteria. Finally, binary metrics should be avoided and have to include units and their importance level.

In order to define metrics, first direct information from the talks with the client was used—i.e. the measuring of the drones' precision through the absolute error. For most of them, being very specific requirements, expert criteria and intuition was used to define them, like the measurement of usage of existing hardware in the solution and the speed of localization. Lastly, for a requirement such as the algorithm's efficiency, a standard metric was used, in this case, the Big-O Complexity [18].

In Table 3.3 defined metrics associated with their respective requirement are shown.

3.3.2 Steps 2 and 3: Establishing goal values

As already mentioned, metrics were defined based on different criteria according to the individual requirement that is being analyzed. This is also the case for the goal values, so that depending on the metric's own nature, different criteria is used.

One by one:

1. **Error respect to ground truth:** This metric and goal values were determined according to direct talks with the client, in which they were explicitly mentioned.
2. **Hardware utilization compliance:** This metric is defined by the Equation (3.1), where $Total\ hardware\ used = Existing\ hardware\ used + New\ hardware\ used$. This metric was derived by expert criteria as a way of measuring how much new hardware is being used in the solution compared to the total amount of hardware used also in the solution. Hardware components may be discretely summed, like "Raspberry Pi + IMU + Camera..."

Table 3.3: Needs-metrics matrix

			METRICS									
			1	2	3	4	5	6	7	8	9	10
Number	**Importance**	**REQUIREMENTS**	Error respect to ground truth	Hardware utilization compliance	Cycle frequency	Relative localization ratio	Time of signal interruption necessary to fail localization	The swarm system is centralized	Number of parameters needed for calculation	Number of interdependent decisions	Number of predefined priorities needed for the coordination algorithm	Big-O complexity
1	5	The relative localization is precise	x									
2	2	The localization uses the existing hardware		x								
3	4	The localization is fast			x							
4	5	The localization of the larger platform with respect to the drone is relative				x						
5	4	The localization is resilient against signal interruptions					x					
6	1	The coordination is centralized						x				
7	1	The coordination is simple							x			
8	1	The coordination is not interdependent								x		
9	4	The coordination works with respect to previously defined priorities									x	
10	3	The algorithm is efficient										x

$$Hardware\ Compliance(\%) = \frac{New\ hardware\ used}{Total\ hardware\ used} * 100 \tag{3.1}$$

3. **Cycle frequency:** This one was also defined through direct talks with the client, establishing the metric and goal values.

4. **Relative localization ratio:** Similar to the hardware compliance metric, with expert criteria, a percentage metric was designed with Equation (3.2), where $AllMethodsUsed$ might include methods based on absolute localization like GPS, and relative localization methods.

$$RL\ Ratio(\%) = \frac{Relative\ localization\ methods\ used}{All\ methods\ used} \tag{3.2}$$

5. **Time of signal interruption necessary to fail localization:** Expert criteria was used to define this metric, however, the goal values were set along with the client.

6. **Number of independent processing units that make decisions in the swarm:** Also expert criteria was used for this metrix and goal values. It allows to keep the solution

as centralized as possible, being ideal that it's completely centrlized, but having a margin of not more than three possible central processing units for the algorithms.

7. **Number of parameters needed for calculation:** Similar to the previous metric. However, how "simple" an algorithm is refers to a subjective question. In this case, "simple" was defined through the quantity of parameters the algorithms needs in order to do the calculations, because the more it needs, the more steps are necessary to have an output, thus adding complexity.

8. **Number of interdependent decisions:** To measure the level of interdependence ideally no interdependent decisions should be made in the algorithm, but there is a margin of no more than 4 possible. This was determined though expert criteria.

9. **Number of predefined priorities needed for the coordination algorithm:** This one was directly determined with the client, for it was one of the main requirements from the start. The goal values were also determined with the client.

10. **Big-O complexity:** To measure efficency the standard of Big-O complexity was used [18]. In this case, through expert criteria, it was determined that ideally an $O(n)$ complexity would be achieved, but there is a margin of $O(nlog(n))$ which is still efficent considering these are real-time systems.

In Table 3.4 all metrics with their respective goal values are shown.

Table 3.4: Goal values to measure the accomplishment of the client's requirements.

Metrics No.	Requirement No.	Metric	Importance	Units	Marginal value	Ideal value
1	1	Error respect to ground truth	5	cm	<10	<5
2	2	Hardware utilization compliance	2	%	<10	0
3	3	Cycle frequency	4	Hz	>5	>2
4	4	Relative localization ratio	5	%	>10	>50
5	5	Time of signal interruption necessary to fail localization	4	s	>30	>60
6	6	Number of independent processing units that make decisions in the swarm	1	Quantity	<4	1
7	7	Number of parameters needed for calculation	1	Quantity	<7	<4
8	8	Number of interdependent decisions	4	Quantity	<5	None of them
9	9	Number of predefined priorities needed for the coordination algorithm	3	Quantity	>1	>3
10	10	Big-O complexity	3	O	$O(nlog(n))$	$O(n)$

3.3.3 Step 4: Overview

In this phase, an iteration on the previous phase (3.2) was done. Initially the requirement "The coordination algorithm works in real time" was introduced, but after talking with the client

about possible metrics, it was discarded because measuring real-timeness is a difficult task that goes beyond the scope and interest of the project. Moreover, several iterations were done in the current phase because some metrics were defined as binary, for example, the eighth metric was initially defined as "The coordination is not interdependent", with goal values: accomplishes or does not. As this type of metric is not desirable, they were finally changed to ones with quantifiable measurements.

Regarding the current phase, the metrics were correctly defined considering the client's needs from talks with them. It's important to review that they are indeed metrics because you can assign a measurement unit to them as well as goal values. Also, they are not being confused with the testing method, but they are the actual measurement. Finally, the goal valules are consistent with what the client expects, like the precision, and when they were not explicitely told, they were correctly set by standards or expert criteria.

3.4 PHASE 3: Design process conceptualization

Up until this moment in the design methodology, the actual solution hasn't been tackled, but only the characteristics to consider the solution valid have been determined. This phase marks the beginning of the solution design, but this is no easy task, for it should be organized, thourough and without biases.

To determine the solutions candidates, a combination of concepts is conceived, where each of these represents a different way of solving one subproblem of the designed solution. This way the problem is better understood, it covers an extended amount of solutions and, if a problem would arise with one particular solution, it can be improved.

The corresponding methodology is:

3.4.1 Step 1: Problem decomposition into subproblems

Solving the problem of the drones' autonomous and precise localization, as well as their coordination according to the client's specifications includes many factors to be taken into account at the same time, resulting in an overwhelming situation. That is why an engineering problem, which tends to be complex, needs to be decomposed into subproblems, to solve each one individually.

For this, the **functional decomposition** is used, in which the subproblems are defined as functions. First, the problem that is to be solved acts as a black box whose inputs and outputs are to be determined. Furthermore, it is recommended to see these inputs and outputs in terms of energy, material and signals, but the determined inputs muts be relveant to the problem, that is, they cannot be simply "something" that's used in the process, but an important part in it.

Figure 3.1 shows the black box model for this project's problem. Note that a fine line is used

for moving or converting energy, a thick line is used for material transfer, and a dotted line is for control or information signals. This convention is followed on the rest of diagrams. In the diagram, the energy input is already established as the drone's battery by the client, the input signal is the drone's physical position and the output signal is the localization and charging schedule, which is transmitted to the drones and to the user. Notice that there is no material transfer in the process nor energy output because the drones do not give energy to any other system.

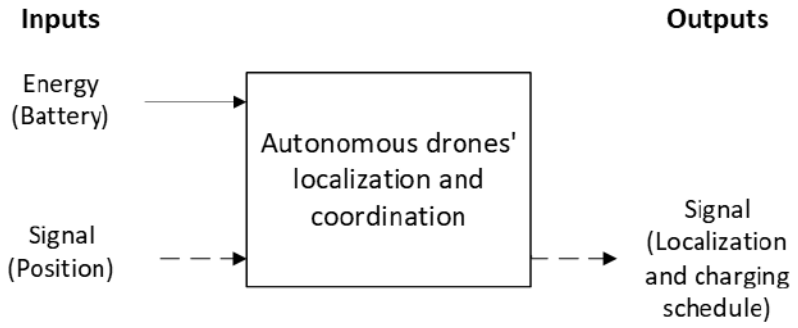

Figure 3.1: Problem to solve with its relevant inputs and outputs.

Once the black box has been determined, the inside has to be studied, which means determining what subproblems exist to convert the given inputs into the required outputs so that in the next step a list with one or more solutions for each subproblem can be generated. This process is iterative, each problem is divided until it can be "fixed by itself". In this case, the decomposition is shown in Figure 3.2

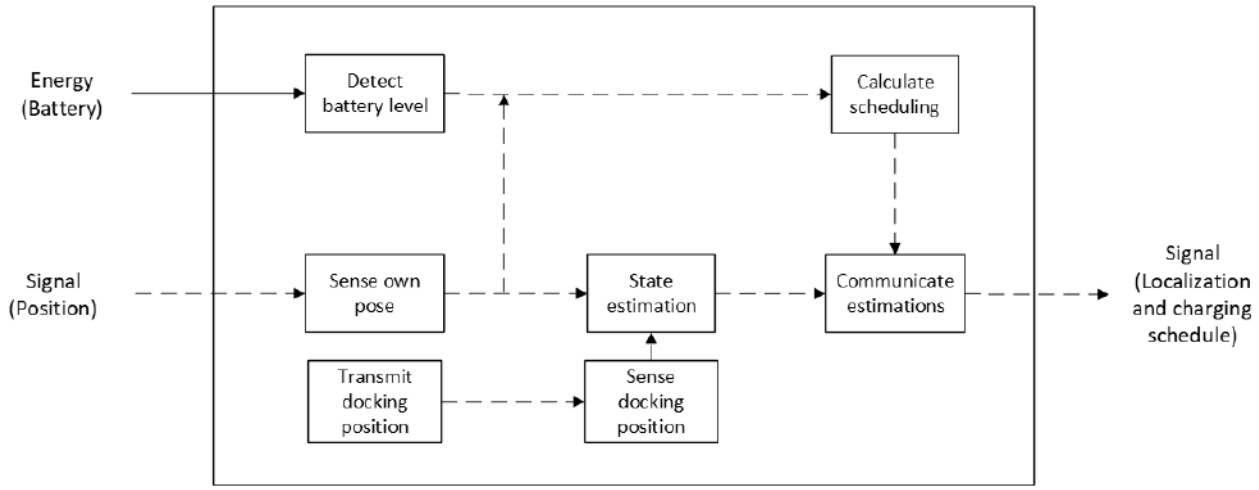

Figure 3.2: Functional decomposition for the problem to solve.

3.4.2 Step 2/3: Concepts searching

In this step a list of concepts as thourough and complete as possible is to be made for the solutions of each block. Also, no selection, elimination, categorization, or filtering is to be made, because they are done in next steps or phases, in a methodological way, to avoid biases.

This searching can be external, through sources outside of the designer's own experience, which helps to reduce biases, optimizes time because existing solutions can be found, and gives context to possible solutions. Or they can be internal, where the designer's own knowledge, habilities and experience are used.

The concepts searching for each block is now defined:

Detect battery level

The drones need to be able to detect their battery level because this is the main factor causing the problem being solved. In this case the drones already have the capacity to detect their own battery level, as voltage or percentage, so in this case this is the concept to be used because it is already implemented, so it's an efficient, practical solution that does not incur in any expenses or technological difficulty.

Sense own pose

To estimate a drone's pose relative to other object's, first the drone needs to know its own pose. To do this, the drones have their own set of sensors (Table 3.1) which includes a lot of really important options, from innertial odometry sensors, to visual odometry sensors and even a GPS. Thus, the choice is made to use the drone's own sensors since they are already integrated into the system, leaving out the complexities of adding new components, also being a commercial drone, they have been thouroughly tested and calibrated, so that's one less layer of complexity. Finally, economically this is better since there is no necessity to buy new components.

Transmit docking position

Just as explained before, the drone also needs to know the docking's pose, so it has to transmit it somehow. The first concept came from one of the stakeholders' expert criteria: using an AprilTag. Then, through *Google Scholar*'s search engine, the keyphrase "relative localization of quadcopter" was used to search solutions from other academic institutions. This phrase, although not directly related to the actual subproblem to solve, was used in order to obtain results with a similar context of this project, in this case, a robotic swarm. From this search, the following results were obtained:

In [19], [20] and [21] the position of the object to track or locate is not actually transmitted, but a detection algorithm, for example, based on reinforced learning in [19], is used to detect

the shape, motors, or other specific characteristics of the object to track or locate.

[22], [23] and [24] use ultra-wideband (UWB) modules to measure the distance between the two objects, however, [22] and [23] combine this sensor with additional localization methods. In the first, the heading in the drone's trajectory is first calculated and fed into the flight controller, and in the latter, odometry based on IMU, altimeters and optical flow is used and fused with the UWB measurements with an EKF. In [24] the UWB measurements are fed into an EKF where the system's model was mathematically determined.

Finally, LiDAR technology is used in [25], where the LiDAR sensor detects the object of interest's pose, and a fisheye camera is used in both to determine the orientation based on external objects. In summary, the following methods for transmitting the docking position were determined:

- AprilTag
- Nothing (the drones detect its shape with cameras or LiDAR)
- Ultra-wideband

Sense docking position

Following the previous blocks, the drones also need to sense where the docking is, but this is directly related to how the docking is transmitting its pose. Thus, being consistent with that block, the following technologies are determined:

- Camera to detect AprilTag or shape of docking
- LiDAR to also detect shape of docking
- Ultra-wideband

State estimation

Once the drone has sensed its own position, and the position of the docking port, the necessity of a pose estimation arises, since there might be several sensor inputs. For this it is recommended by the client's expert criteria to use a Kalman Filter, however, once again the search engine *Google Scholar* was used with the keyword "drones pose estimation" to gather concepts other academics are using in similar contexts.

The client's idea to use a KF is reinforced with [26] and [27] who use an EKF to estimate a quadcopter's pose. The first is a tutorial on how to use an EKF to estimate a quadcopter's state, while in the latter the EKF is used to estimate the state of many small rigid bodies that a representative of a swarm of miniature quadcopters.

Adding the designer's expert criteria, EKF being an already studied topic, this is chosen due to the simplicity of implementing it and the vast amount of packages that already implement this method in different technologies (e.g. Arduino, Raspberry Pi, MATLAB, etc).

Calculate scheduling

When the drones has all the necessary information about its location and battery level, now it needs to calculate a priority score in order to schedule the charging. For this, due to the client's expert criteria, a basic search was done in terms of processes scheduling, in this case, the key phrases "priorities scheduling algorithm" and "scheduling algorithm real time" were searched in *Google*, where from [28], [29], [30] the following concepts were generated:

- Earliest Deadline First (EDF)
- Least Slack Time (LST)
- Latest Start Time
- Earliest Start Time
- Shortest Job First (SJF)
- Least Completion Time (LCT)
- First Come First Served (FCFS)

In all of these, except FCFS, the priority score is calculated for each drone, and then sorted according to what the specific algorithm states.

Communicate estimations

To communicate sensors data, pose estimation, priority score, schedule, and any other data the drones already have a radio communication with their controller and computer that according to Table 3.1 has a 4 km range, but also as it is a solution that is already implemented into the drones, so similarly as other concepts, this is simpler to implement, and less expensive because there is no necessity to buy new components.

3.4.3 Step 4: Concepts organization and combination

Now that there are many concepts, it is necessary to combine to finally propose candidate solutions to the problem. In order to do that, tables with combinations of concepts are made, in which the most viable ones to solve are shown and the combinations among them are explored. In this way, creativity is fomented and biases are avoided when exploring as many combinations as possible.

In this case, two routes are proposed: the relative position estimation (Figure 3.3) and the scheduling subproblem (Figure 3.5), since the latter is just about software (independent of hardware) and the first one includes hardware.

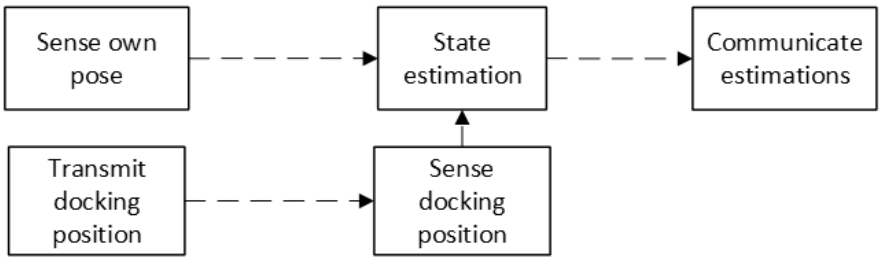

Figure 3.3: Relative position estimation route.

For the localization route, the combination table shown in Table 3.5 was made. Note, however, that certain subproblems' solutions are related to each other. Because of this, the four combinations shown in Table 3.6 and Figure 3.4 are proposed for this route. In A, the drone's camera detects the AprilTag's pose, in B the camera detects the port's shape, in C the LiDAR sensor detects the port's shape and in D the drone gets the pose through UWB. In all oof them the drone uses its own sensors to estimate its pose, uses EKF to fuse sensors' data and state estimation, and uses its own antenna to communicate with the base.

Table 3.5: Combination table for localization route.

Sense own position	**Transmit docking position**	**Sense docking position**	**State estimation**	**Communicate estimations**
Drone's sensors	AprilTag	Camera	EKF	Drone's antenna
	Nothing	LiDAR		
	UWB	UWB		

Table 3.6: Concepts to explore of the localization route.

ID	**Concepts**				
A	Drone's sensors	AprilTag	Camera	EKF	Drone's antenna
B		Nothing	Camera		
C		Nothing	LiDAR		
D		UWB	UWB		

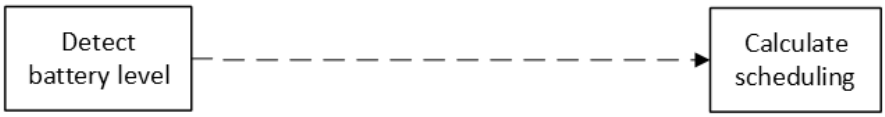

Figure 3.5: Scheduling route.

Regarding the scheduling route, in Table 3.7 the proposed concepts are shown. In this case, the proposals are not shown as images, but as the equation each of them would solve. These are shown in Equations (3.3) to (3.9).

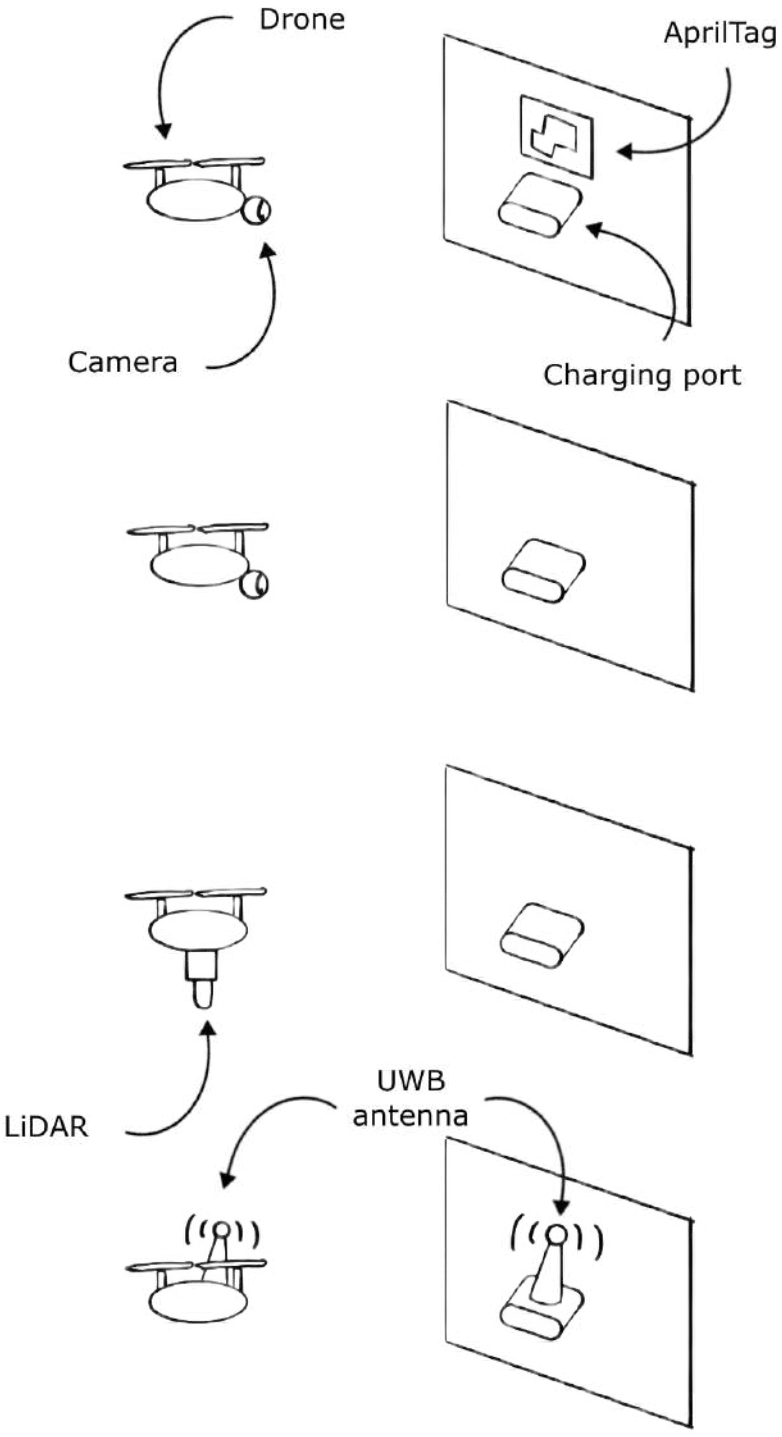

Figure 3.4: Concepts for the localization route, from top to bottom, respectively, A, B, C and D.

Table 3.7: Concepts to explore of the scheduling route.

ID	Concepts	
A	Drone's battery detection	EDF
B		LST
C		Latest Start Time
D		Earliest Start Time
E		SJF
F		LCT
G		FCFS

$$EDF = w_{battery} * battery\ left + w_{distance} * distance\ to\ charger + w_{memory} * memory\ left \quad (3.3)$$

$$LST = \frac{battery\ left}{consumption\ rate} - \frac{distance\ to\ charger}{velocity} - charging\ duration \quad (3.4)$$

$$Latest\ Start\ Time = \frac{battery\ left}{consumption\ rate} - charging\ duration \quad (3.5)$$

$$Earliest\ Start\ Time = \frac{distance\ to\ charger}{velocity} \quad (3.6)$$

$$SJF = charging\ duration \quad (3.7)$$

$$LCT = \frac{distance\ to\ charger}{velocity} + charging\ duration \quad (3.8)$$

$$FCFS = battery\ left \quad (3.9)$$

3.4.4 Step 5: Overview

An extensive space of concepts was created and all the concepts within were successfully explored with confidence and based on different sources. The research done with both external and internal searching met the project's nature, which is academic, and was thorough and complete.

3.5 PHASE 4: Concepts selection and evaluation

Now that there are concepts candidates for the solution, a system to determine which of them should be implemented as less biased as possible is necessary. At this point, proofs of concept should be made through prototypes, but this go beyond the time scope of this project.

Because of this, a concept selection is imposed, and to do that they must be evaluated with respect to the client's needs, comparing their strong and weak aspects. The selection methodology consists of two parts:

3.5.1 Step 1: Filtering

Instead of directly choosing the winner concept from an evaluation matrix, a filtering step allows to show the strengths and weaknesses of each concept relative to the client's needs. These weaknesses could be worked on to make the concept better and maybe meet a need that it did not before, or to combine two concepts that together eliminate such weakness.

In order to filter, a matrix with the concepts and evaluation criteria is built. These criteria correspond to the requirements defined in Phase 1 (Section 3.2), but only those associated to the system or subsystem to evaluate are to be used.

The filtering matrix for the localization route is shown in Table 3.8, where concept B was chosen as "neutral" because of the designer's intuition. This was actually proved right because this concept got the middle placement.

Table 3.8: Filtering matrix for the localization subsystem.

		Concept			
Criteria number	**Selection criteria**	A	B	C	D
1	Error respect to ground truth	0	0	0	0
2	Hardware utilization compliance	0	0	-	-
3	Cycle frequency	+	0	0	+
4	Relative localization ratio	0	0	0	0
5	Time of signal interruption necessary to fail localization	0	0	0	0
	Sum +	1	0	0	1
	Sum 0	4	5	4	3
	Sum -	0	0	1	1
	Total evaluation	1	0	-1	0
	Place	1	2	4	2
	Continue?	Yes	No	No	No

The criteria used to consider whether a concept was better or worse than B were the following:

1. As the accuracy depends greatly on the hardware used, and in this case the hardware has not been chosen, they all are considered equal. Nonetheless, in the future it might be important to iterate and with actual data and assign a punctuation to the concepts.
2. In A and B the only hardware used is the drone's camera, whereas in the other two, new components are added.
3. Since B and C both depend on a neural network to learn to identify the object's shape, they are slower than A and D, who use simpler methods.
4. As all the concepts use a relative and an absolute method and are fused through an EKF, they all have the same punctuation.
5. Similarly, all of them depend on an absolute method and a relative method, so if one or all signals are lost, the EKF will compensate equally for all of them.

From the results of the filtering, only one solution came up as a winner. Concept C was completely defeated by the others, being discarded for future reference. C and D got the second place, and it does not make sense to combine them because it would add more hardware and processing time (both being worse). So the winner concept, A, will be the one to be implemented without the need of an evaluation matrix.

Regarding the filtering for the scheduling subsystem, Table 3.9 shows its respective matrix, where Latest Start Time (concept C) was chosen as neutral because it uses parameters in the middle ground—not too many and not too few.

Table 3.9: Filtering matrix for the scheduling subsystem.

		Concept						
Criteria number	**Selection criteria**	A	B	C	D	E	F	G
1	Hardware utilization compliance	+	0	0	+	+	+	+
2	Number of independent processing units that make decisions in the swarm	0	0	0	0	0	0	0
3	Number of parameters needed for calculation	0	-	0	+	+	0	+
4	Number of interdependent decisions	0	0	0	0	0	0	0
5	Number of predefined priorities needed for the coordination algorithm	+	+	0	0	-	0	-
6	Big-O complexity	0	0	0	0	0	0	0
	Sum +	2	1	0	2	2	1	2
	Sum 0	4	4	6	4	3	5	3
	Sum -	0	1	0	0	1	0	1
	Total evaluation	2	0	0	2	1	1	1
	Place	1	6	6	1	3	3	3
	Continue?	Yes	No	No	Yes	No	No	No

In this case, the following criteria were used:

1. The neutral, C, uses consumption rate as a parameter, and the drone cannot on its own calculate this, so it would be necessary to add a component that measures, for example, the battery's current consumption. Since B also uses this parameter, it was deemed equal ("0"), but the rest use parameters that can be obtained with the current hardware, thus being better in this regard.

2. All the algorithms have only one independent processing unit making the decisions, since the priority score is calculated and sent to the central computer, and then it makes a decision and communicates it to the drones.

3. This case was a sum of the variables in each equation. When the sum was higher than C, it got a "-", lower got a "+" and equal, a "0".

4. Because all the concepts are centralized, there is actually just one decision being taken by the central computer—which one gets the best score and goes first, so as this decision does not depend on any other, they are all equal.

5. Concept C uses one parameter whose importance the operator can predefine at any moment (battery left) so, any other concept with the same amount of parameters whose importance can be predefined (battery left, distance to charger and memory left) got a "0", while the ones with more parameters got a "+" and the ones with less got a "-".

6. Since fundamentally all algorithms consist in a sorting of a list of priority scores, they all have the same complexity.

A and D move on to the evaluation step, as they tied the first place, but the rest do not move. They cannot be upgraded (because of their own definition) and cannot be combined because in all combination cases they win and lose, ending up even, or just lose a point, ending up worse.

Overview

In Tables 3.8 and 3.9 the usage of requirements relevant to each concept is evident. Furthermore, the reference concept was correctly chosen in both cases since they both got a final placement in the middle.

Also, the scoring process was consistent throughout—characteristics that made a concept worse, applied equally to all of the other concepts. This consistency is also applied when assigning final scores, since summing "+" and substracting "-" determines their placement in a mathematical and absolute way.

3.5.2 Step 2: Evaluation

To choose the winner concept from the filtered concepts from the last step, a more precise evaluation is systematically done. The weight assigned to each selection criteria is directly related to the importance each requirement has according to the client, i.e. how influential each of the criteria is for the final placement is a consequence of the importance grade the client put to it. Because of this, the percentage is assigned through normalization of the importance score with a percentage distribution according to the amount of criteria to reach the 100 % of scoring (see Section 3.2).

In Table 3.10 the proposed evaluation for the scheduling route is shown, from it only one winner concept comes out to be implemented.

Table 3.10: Evaluation matrix to choose concept to implement.

			Concept			
			A		D	
Criteria number	**Selection criteria**	**Weight**	Evaluation	Weighted evaluation	Evaluation	Weighted evaluation
1	Hardware utilization compliance	14%	**3**	0.43	2	0.29
2	Number of independent processing units that make decisions in the swarm	7%	3	0.21	3	0.21
3	Number of parameters needed for calculation	7%	**3**	0.21	4	0.29
4	Number of interdependent decisions	29%	3	0.86	3	0.86
5	Number of predefined priorities needed for the coordination algorithm	21%	5	1.07	**3**	0.64
6	Big-O complexity	21%	3	0.64	3	0.64
		Total points		3.43		2.93
		Placement		1		2
		Continue?		Yes		No

In this matrix, the reference is set as a bold three for each of the criteria, and they are set different for each one, not only on one concept. The references were set according to the designer's expert criteria. Additionally, if the other concept is slightly better, it will get a score of a four, and if it is absolutely better, it will get a five. Contrarily, if it is slightly worse, it will get a two, and it is absolutely worse it will get a 1. This assignment is explained as follows:

1. A does not need any other hardware at all, D needs charging duration, which needs at least one more hardware component.
2. The same explanation as in the previous step applies.
3. A is considered to have a balanced amount of parameters, but D has one less, so it is slightly better.
4. The same explanation as in the previous step applies.
5. In this case, D has only one predefined priority, but A has two more, making it absolutely better.
6. The same explanation as in the previous step applies.

It is possible to see from the matrix that the concept with a bigger score is A, so this concept will be implemented. As it was previously mentioned, for the localization route it is not necessary to do an evaluation because the filtering already filtered out one absolute winner.

Overview

The evaluation process was concluded, and the scores were systematically assigned, avoiding biasses regarding the value each concept has in the weighted evaluation, also the obtained points are given taking into consideration what the two concepts offer, reducing a wrong judgment.

As a result of this process, the winner concept of the scheduling route to implement is A, along with the (also) A concept from the localization route.

Chapter 4

Solution implementation and validation set-up

In this chapter, the detailed implementation of the AprilTag detection system and the scheduling algorithm is discussed. For the same reason they were separated in Section 3.4 it was decided to work the concepts separately. Furthermore, the concepts are implemented as proofs of concept because actual implementation goes beyond the scope in time and complexity of this project.

4.1 AprilTag detection

4.1.1 Initial simulation

From Chapter 3 the winner concept is the AprilTag detection with the drone's camera, fusing that data with the drone's own odometry with an EKF and transmitting it via its own antenna. Before implementing all of this on the actual hardware, the implementation is carried out in simulation. This allows to focus the attention on the tag detection and Kalman Filter rather than constantly interrupting the development because of trivial hardware issues.

To begin with the implementation, it was decided to use ROS because of the following reasons:

- The drone's company, Parrot, has an API called Olympe [31] to extract most of the drone's data, like sensors, camera information and image, battery, among many more, and there is a ROS bridge built with Olympe to use that data in the operating system [32].
- It is possible to program in the language Python, which is generally considered easier for being high-level, also the designer is experienced in it as well as the client, who can offer advice.
- There are lots of packages available with robotic tools like AprilTag detection, EKF and simulation.

- The client has already a framework built with ROS for these drones.

For the simulation environment, ideally the Sphynx simulator would have been used, since it's Parrot's own simulator for the ANAFI drone [33], however, it has heavy graphics requirements that the available computers could not meet. Therefore, open-source simulator Gazebo [34] was instead used, which has lighter graphics requirements, which actually is a building block for Sphynx, so the implementation is as close as it can be to a real-world simulation with the current hardware. Other packages used in the simulation are the following:

- RotorS Simulator for quadcopter simulation, including realistic physics for the drone's movements, and sensors for odometry [35].
- apriltag_ros which provides a visual fiducial detection algorithm with the detection image and detected tags' pose being readily available to just extract [36] [12].

So a simple simulation environment was built on Gazebo with a simple box with AprilTags on all sides that represent the docking port and a drone. This is shown in Figure 4.1, where the box is at (0,0,2) and the drone is at (-1,0,2). With this setup, the tag detection is straightforward, since there is only to launch the apriltag's continous_detection launch file and the tag's pose is immediately accessible through the tag_detections topic.

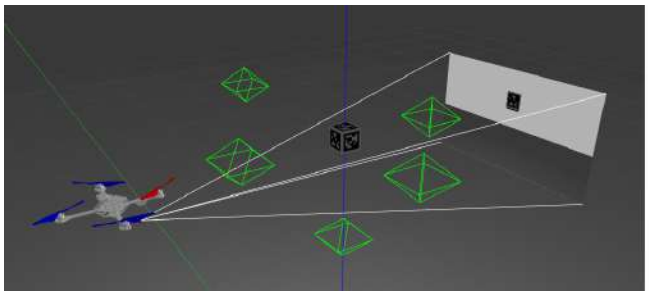

Figure 4.1: Gazebo simulation for testing AprilTag detection.

The data from the from the apriltag_ros package's custom message is shown in Table 4.1. Besides this information, it includes a header with a timestamp and the frame from which is being measured, in this case "hummingbird/camera_link_optical", just as previously seen in Figure 4.3. Also, it includes the actual detections, in this case, a tag with ID 2 was detected, and it includes its size, pose and covariance, which in this case is

$$\mathbf{Cov} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 4.1: Summary of raw tag detection message.

ID	Size	Position (m)			Orientation			
		x	y	z	x	y	z	w
2	0.08	0.000	0.135	0.860	0	-0.71	0	0.71

Note that the tag is not detected with 1 meter on the z-axis, this is because i) the measurement is done from the camera optics, not from the drone's body, ii) the camera is not exactly on the drone's center and iii) the apriltag is also not on the box's center, but has half the box's side as offset.

Two other notes: the tag is detected with a -90° rotation along the y-axis, which makes total sense when looking at Figure 4.3. Furthermore, the measurement covariance is set to zero by default—it does not calculate it.

During this implementation, many coordinate systems start to appear, thus making it necessary to clearly specify the coordinates' systems present in the current simulation. Figures 4.2 to 4.4 show all the coordinate systems from the same perspective for the case of Figure 4.1. Following an RGB homologue to XYZ, red axis is x-axis, green is y-axis and blue is z-axis.

The coordinates frames are visible using RViz, which is a 3D visualization tool for ROS [37]. Note that some frames have arrows pointing to another frame, this means that the first one is relative to the latter—i.e. the first one is a child frame to the latter (called parent frame).

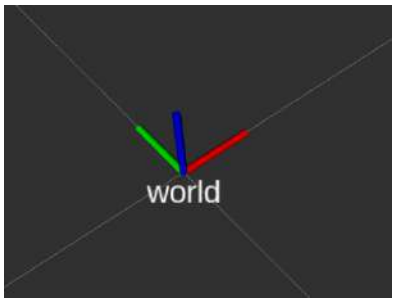

Figure 4.2: Coordinates of "world" frame.

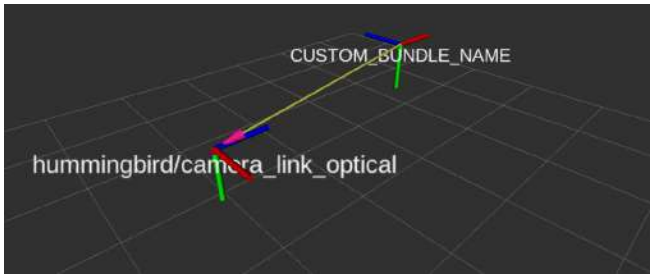

Figure 4.3: Coordinates of the tag detection and camera optical frame.

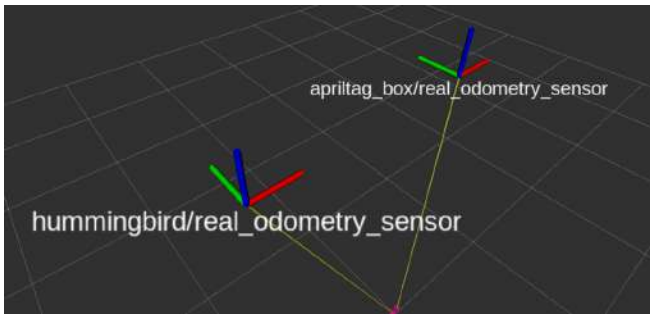

Figure 4.4: Coordinates of odometry sensors frames of the apriltag box and the drone.

Because there might be more than one tag in the drone's FOV, an intermediary ROS node is implemented that takes the raw tag detections and calculates the Euclidean distance of each of them and returns the closest ones. So it filters the detections to only keep the closest one, which is the important.

As the pose of the AprilTag relative to the drone's body ${}^{body}P_{AprilTag}$ is what is required, it is necessary to transform the known pose of the tag in the camera's frame ${}^{camera}P_{AprilTag}$. To do that, the following transformation is done with ROS package "tf2", which keeps track of multiple coordinate frames over time and handles the math internally [38]:

$$ {}^{body}P_{AprilTag} = {}^{body}_{camera}T \ * \ {}^{camera}P_{AprilTag} $$

However, the covariance matrix cannot be transformed by tf2, so a manual approach is necessary in the following form [39]:

$$ {}^{body}C_{AprilTag} = ({}^{body}_{camera}R) \ * \ ({}^{camera}C_{AprilTag}) * ({}^{body}_{camera}R)^T $$

But as the covariance matrix C is 6x6 and the rotation matrix R is 3x3, the above equation is applied to the diagonal 3x3 components of C, leaving the off-diagonal elements as zero.

The results in Table4.2 are obtained for the coordinates' transformation of the tag detection, where it is possible to see now that the apriltag box is 1 meter away from the drone in the x-axis and rotated -90° in the same axis, which is consistent with Figures 4.3 and 4.4. The covariance in this case is the following, where it is possible to see that, with the transformation, a little covariance is added in the diagonal.

$$ C = \begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix} $$

Table 4.2: Summary of transformed tag detection message.

Position (m)			**Orientation**			
x	y	z	x	y	z	w
1.008	0.011	0.045	-0.71	0	0	0.71

Now there is only to obtain the transformation from the apriltag box's odometry to the drone's. The pose transform is done by the tf2 package, the covariance is transformed as before, but as this is odometry data, it includes velocity (referred to as twist), linear and angular, so they have to be manually transformed:

$$ {}^{drone}_{world}V_{box} = {}^{drone}_{box}R * {}^{box}_{world}V_{box} $$

The result from this transformation is shown in Table 4.3, which is very similar to the last result, as expected. The covariances are 6x6 matrices filled with zeroes for the pose and the velocity—this is not strange because the simulated odometry is "perfect".

Table 4.3: Summary of transformed odometry message.

Position (m)			**Orientation**				**Linear velocity**			**Angular velocity**		
x	y	z	x	y	z	w	x	y	z	x	y	z
1.000	0	0.035	0	0	0	1	0	0	0	0	0	0

The main difference here with respect to the tag detection is that there is no rotation, and this makes sense because both odometry frames are lined up (Figure 4.4). Even then, the client is only interested in the position, not the rotation, so it is of no importance.

Finally, a Kalman Filter can be implemented to fuse together the two measurements. For this the package "robot_localization" is used, and it contains an EKF localization node that, after being configured, does all the internal calculations [40]. The non-default configuration used is the following:

EKF configuration

```
world_frame: hummingbird/real_odometry_sensor
odom_frame: hummingbird/real_odometry_sensor
base_link_frame: apriltag_box/real_odometry_sensor

odom0: /apriltag_box/transformed_odom
odom0_config: [true,  true,  true,
               true,  true,  true,
               true,  true,  true,
               true,  true,  true,
```

```
                false, false, false]
odom0_queue_size: 15

pose0: /tag_detections/transformed_pose
pose0_config: [true,  true,  true,
               true,  true,  true,
               false, false, false,
               false, false, false,
               false, false, false]
pose0_queue_size: 15
```

The world and "odom" frames are chosen as the drone's odometry because we want the pose of the "base_link" frame (i.e. april tag box frame) to be relative to the drone's frame, not to the world's frame. The first odometry sensor corresponds to the previously mentioned transformed apriltag odometry sensor, which has information on pose and twist but not in acceleration, thus leaving the last row empty, The queue size is put to a higher number than the default value because the input data has a high frequency. Similarly, the other sensor corresponds to the previously mentioned transformed tag detection, but in this case it only has pose information, leaving the last three rows as false. A more thorough explanation on the different settings parameters is found in the source *git* repository [41].

With the previous messages fed into the filter, its output is exactly the same as in 4.3. This happens because the odometry data has more weight in the filter, since it provides more information through the velocity measurements and has a smaller covariance.

To test the filter, artificial noise can be introduced into the sensors. In real hardware, the noisier odometry to be expected is the inertial from either the drone or the docking, so the noise will be put into the apriltag box odometry sensor. Specifically, a normal noise of 1 meter was introduced in each position axis and of 0.1 for each quaternion. Noise was also introduced in the velocity.

After introducing the noise to the odometry, the results in Table 4.4 are obtained, where twist is not included, but the measurements were of zero for all velocity components, as well as its covariance.

Table 4.4: Summary of filter message with artificial noise.

Position (m)			**Orientation**			
x	y	z	x	y	z	w
1.000	0.008	0.045	-0.668	0	0.004	0.744

The pose covariance is inherited from the noisier input—the tag detection, so it is the same:

$$C = \begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$$

The filter is effective because now that the odometry input is noisier (more covariance), the tag detection weighs more and, thus, a good result is still obtained. This is the expected behavior in the actual implementation.

4.1.2 AprilTag detection's covariance determination

The pose estimation from the quadcopter and docking's actual odometry sensors currently have a covariance associated with them, however, the AprilTag detection does not. For this reason, a covariance mapping is desired, in which almost every position detected by the tag has a covariance associated. To map the covariance, it is necessary to take real measurements with the hardware.

The idea is simple. The covariance is calculated from the error in measurement from the tag detection relative to the true position of the docking. In this case, the true position is called *ground truth*, and corresponds to the measurements made by a Vicon system which is assumed to have a covariance of zero and that it is the true pose of the system. A Vicon system is a motion capture system based on inrarred cameras that detect reflective markers and measures or tracks their positions with high precision and low latency [42].

First, an AprilTag was printed with the same dimensions as in simulation (8x8 cm) and pasted into a wood board. To measure its position with the Vicon system, infrared markers were put on the board with an offset from the tag, as shown in Figure 4.5. At the same time, similar markers were put on the ANAFI drone (Figure 4.6). To measure the tag's offset and write it into the Vicon system, the drone was put on top of the tag, in its center, so that Vicon on its own could precisely measure it, this is shown in Figure 4.7. In Figure 4.8 the computer with the visualization of the markers is shown.

With all of this, the Vicon system indirectly tracks the drone's and tag's pose. Now it is only a matter of detecting the tag as in simulation, and measuring the error between this measured pose and the ground truth, calculate the covariance, and associate it to the true drone's pose where that measurement was taken.

Ideally, the experiments would be designed to generate a covariance map that covers all positions of interest, varying all six degrees of freedom (x, y, z, roll, pitch, and yaw). However, this is beyond the scope of the current project. Instead, a proof of concept was conducted in which only three variables—x, y, and z—were varied. Since the physical world is continuous, it was necessary to discretize the space to collect a manageable number of measurements while

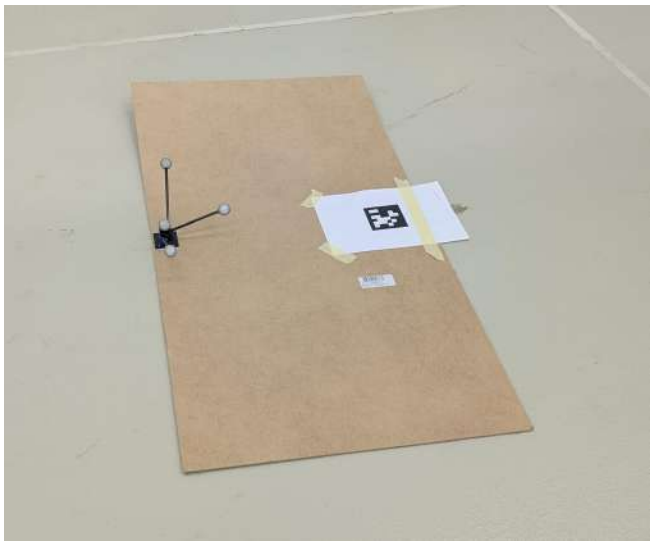

Figure 4.5: Image of the wood board with the tag and infrarred markers.

Figure 4.6: Image of the ANAFI drone with infrarred markers.

maintaining sufficient precision. After discussions with the client, a discretization step size of 30 cm for position and 15° for orientation was agreed.

The board is put vertically as it would be next to the docking and the drone is put on three different random positions labeled as “left”, “center” and “right”, and three different heights for each of them, labeled as “bottom”, “center” and “top”. To mark down the positions, a cross with tape was put on the ground, and for the height, three different objects or their combination was used: a plastic box, cardboard boxes and a suitcase. Some of these setups are shown in Figures A.1 to A.4.

Note that it is not important where the drone and tag board are located, since the Vicon measures their pose and this is the measurement taken as real or ground truth.

To use the previous frames transformation nodes, a series of new frames conversions were necessary. In Figure 4.9 the result of the conversions is shown. Particularly, the following conversions were performed:

Figure 4.7: Image of the setup to measure the markers' offset relative to the AprilTag.

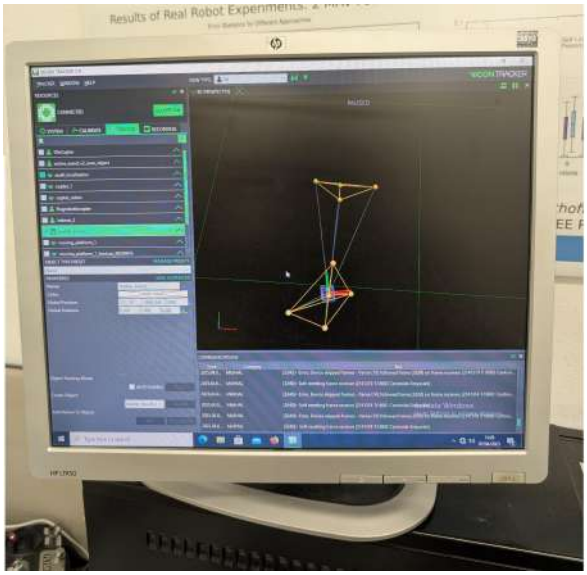

Figure 4.8: Image of the computer with the Vicon system interface.

1. **anafi_localization_combined:** The Vicon measurement of pose was used.
2. **gimbal_camera_frame:** The gimbal orientation data was used but with an offset of 8.8 cm in the y-axis, as it was measured with a ruler.
3. **tag_frame:** The tag detections' pose with respect to the gimbal frame.
4. **marker_board_1:** The Vicon measurement of where the tag is.

Then, the previous coordinates transformation for odometry and tag detection were adapted to use real-world data. The odometry would now take as input the ground truth's measurements of the drone's and marker board's pose, but the Vicon system separates the velocities and positions

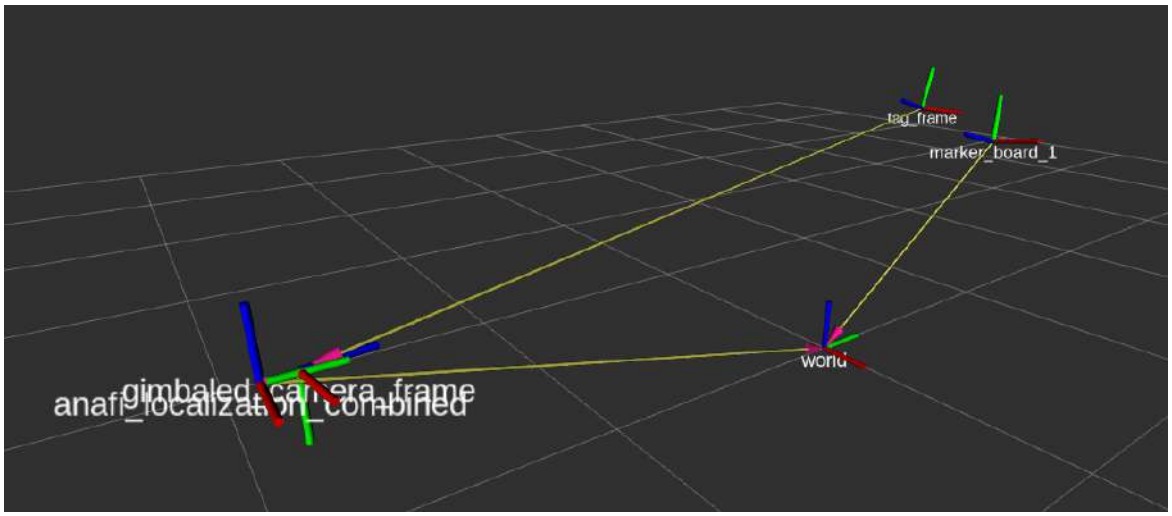

Figure 4.9: Frames used in hardware.

of the measurements, so a conversion node had to be made in order to put the data together as a full odometry message. For the tag detection, it was only necessary to change the camera from the simulated to the real one.

One last node was made to transform from the marker board's to the tag's frame. This was done to directly obtain the error measurement instead of calculating it later. In total, measurements were done in 9 different positions and more than 30 were recorded for each position in order to have a significant amount of data. The output of the measurement was the marker board's pose (x, y, z, roll, pitch, yaw). Two different files were generated per test, one for the ground truth data and the other for the error data.

In all positions the drone was manually put with 0° rotation on any axis, however, it was noted that the slightest yaw offset would increase the position error too much. Since the client wanted mainly the relative position, the orientation was sacrificed in order to take accurate position's measurements. This way, the *anafi_localization_combined* frame was forced to have a rotation of (0, 0, 0) degrees.

After the experiments in the nine different positions, Tables 4.5 to 4.8 show the average and standard deviation obtained from the tests for each position. There is to remember that the angles measurements are with respect to a fixed (0, 0, 0) degrees of drone's orientation, so they don't represent real orientation, but they are still necessary for the covariance matrix, as the program expects a 6x6 matrix.

From these tables, it is also possible to see that there is little standard deviation for the measurements, which in turn represent a good accuracy, but the average position error is generally greater than 10 cm, i.e. not the expected precision. But this is not yet the state estimation, just the tag detection.

Table 4.5: Average summary of ground truth measurements for covariance determination.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
top-right	1.2091	0.4993	2.0904	77.1322	-9.9575	4.4245
bottom-center	-1.036	0.1363	1.9714	77.108	-9.9522	4.4384
top-left	-0.0294	0.1355	2.9758	77.1391	-9.9558	4.4269
center-right	-0.0096	-0.3887	2.9351	77.1269	-9.955	4.4333
center-left	1.1998	0.1274	2.0487	77.1301	-9.96	4.4304
center-center	-1.0248	0.5077	1.9915	77.1007	-9.9492	4.4402
bottom-left	1.2168	-0.3974	2.0231	77.1379	-9.9555	4.4237
bottom-right	-1.0565	-0.3893	1.9683	77.1072	-9.9401	4.4346
top-center	-0.02	0.5076	2.9743	77.1414	-9.9596	4.4279

Table 4.6: Standard deviation summary of ground truth measurements for covariance determination.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
top-right	0.0001	0.0001	0.0002	0.0219	0.008	0.0143
bottom-center	0.0001	0	0.0002	0.0224	0.0089	0.0149
top-left	0.0001	0.0001	0.0002	0.0206	0.0078	0.0132
center-right	0.0001	0	0.0002	0.0198	0.0073	0.0125
center-left	0.0002	0.0001	0.0002	0.0233	0.0085	0.0145
center-center	0.0001	0	0.0002	0.0205	0.0079	0.0135
bottom-left	0.0001	0	0.0002	0.0185	0.0069	0.012
bottom-right	0.0001	0	0.0002	0.0191	0.0078	0.014
top-center	0.0002	0.0001	0.0002	0.0218	0.0083	0.0139

Table 4.7: Average summary of error measurements for covariance determination.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
top-right	-0.305	0.2165	0.0823	-0.4397	8.8253	-4.6267
bottom-center	-0.5345	0.2336	-0.1511	-2.9551	10.8077	-5.0106
top-left	-0.3308	0.3798	-0.2804	-0.3939	11.9671	-3.4619
center-right	-0.3207	0.1506	0.0835	-1.9207	9.4692	-4.9343
center-left	-0.3466	0.2771	-0.2953	-2.6657	11.2098	-4.4573
center-center	-0.5264	0.2272	-0.1474	-0.0222	10.5171	-5.6064
bottom-left	-0.3585	0.2566	-0.3054	-3.6357	11.2199	-4.9909
bottom-right	-0.3731	0.0683	0.0412	-4.5101	10.4796	-5.2564
top-center	-0.495	0.3298	-0.1311	4.5108	10.2937	-4.9353

Table 4.8: Standard deviation summary of error measurements for covariance determination.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
top-right	0.0008	0.0008	0.0012	0.1648	0.0956	0.0624
bottom-center	0.0004	0.0011	0.0021	0.1746	0.5102	0.0651
top-left	0.0035	0.0053	0.002	0.2311	0.115	0.085
center-right	0.0011	0.0009	0.0017	0.1677	0.1286	0.0567
center-left	0.0025	0.002	0.0021	0.2212	0.1331	0.0868
center-center	0.0015	0.0025	0.0022	7.1809	1.5305	0.311
bottom-left	0.0029	0.0022	0.006	0.1468	0.1767	0.0892
bottom-right	0.0016	0.0017	0.0025	0.1653	0.1344	0.064
top-center	0.0037	0.0059	0.0041	3.7495	3.338	0.4414

Based on these measurements, covariance matrices were generated for each position (x,y,z). In terms of position, the covariance values were effectively zero, which aligns with the standard deviation results. Since the deviations were all on the order of millimeters, squaring them to obtain variance yields even smaller values. The covariance of the orientation angles was not considered, as they were not derived from the drone's actual orientation.

$$(0.0, -0.3, 3.0): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.03 & 0.019 & 0 \\ 0 & 0 & 0.001 & 0.019 & 0.26 & 0.009 \\ 0 & 0 & 0 & 0 & 0.009 & 0.004 \end{bmatrix}$$

$$(1.2, -0.3, 2.1): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.001 & 0.001 & 0 \\ 0 & 0 & -0.001 & 0.022 & -0.014 & 0.001 \\ 0 & 0 & 0.001 & -0.014 & 0.031 & 0.007 \\ 0 & 0 & 0 & 0.001 & 0.007 & 0.008 \end{bmatrix}$$

$$(-1.2, -0.3, 2.1): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.027 & 0.004 & 0 \\ 0 & 0 & 0 & 0.004 & 0.018 & 0.006 \\ 0 & 0 & 0 & 0 & 0.006 & 0.004 \end{bmatrix}$$

$$(0.0, 0.0, 3.0): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.001 & 0 & 51.565 & -10.527 & -2.176 \\ 0 & 0 & 0 & -10.527 & 2.342 & 0.448 \\ 0 & 0 & 0 & -2.176 & 0.448 & 0.097 \end{bmatrix}$$

$$(1.2, 0.0, 2.1): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.049 & 0.011 & 0.012 \\ 0 & 0 & 0 & 0.011 & 0.018 & 0.008 \\ 0 & 0 & 0 & 0.012 & 0.008 & 0.008 \end{bmatrix}$$

$$(-0.9, 0.0, 2.1): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.028 & 0.01 & -0.002 \\ 0 & 0 & 0 & 0.01 & 0.017 & 0 \\ 0 & 0 & 0 & -0.002 & 0 & 0.003 \end{bmatrix}$$

$$(0.0, 0.6, 3.0): \begin{bmatrix} 0 & 0 & 0 & -0.001 & 0.002 & 0 \\ 0 & 0 & 0 & -0.002 & 0.002 & 0 \\ 0 & 0 & 0 & -0.002 & 0.002 & 0 \\ -0.001 & -0.002 & -0.002 & 14.059 & -12.159 & -1.596 \\ 0.002 & 0.002 & 0.002 & -12.159 & 11.142 & 1.438 \\ 0 & 0 & 0 & -1.596 & 1.438 & 0.195 \end{bmatrix}$$

$$(1.2, 0.6, 2.1): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0.053 & -0.014 & 0.006 \\ 0 & 0 & 0 & -0.014 & 0.013 & 0.001 \\ 0 & 0 & 0 & 0.006 & 0.001 & 0.007 \end{bmatrix}$$

$$(-0.9, 0.6, 2.1): \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.027 & 0.005 & -0.007 \\ 0 & 0 & 0 & 0.005 & 0.009 & -0.001 \\ 0 & 0 & 0 & -0.007 & -0.001 & 0.004 \end{bmatrix}$$

4.2 Implementation of EDF algorithm

As described in Section 3.4, the algorithm to be implemented is EDF, defined by Equation (3.3). Testing this equation is straightforward: three drone objects are created, each with a battery level, memory usage, and distance to a charger. The weights for these three factors were initially set to equal importance (0.33 each), as agreed upon with the client, but they remain configurable so that an operator can adjust them later if needed. All variables were normalized: battery level from 100%, memory usage relative to the drone's total memory capacity (as reported by the ANAFI), and distance from a maximum of 10 meters—though this distance threshold can also be modified based on operational requirements.

For real-world implementation, the same simulation environment used previously was employed, with the key difference being the presence of three identical drones instead of just one—see Figure 4.10. A simulation node was developed to manage this setup. It tracks the odometry of each drone, as well as the position of a box representing the docking port and calculates the Euclidean distance to the charger.

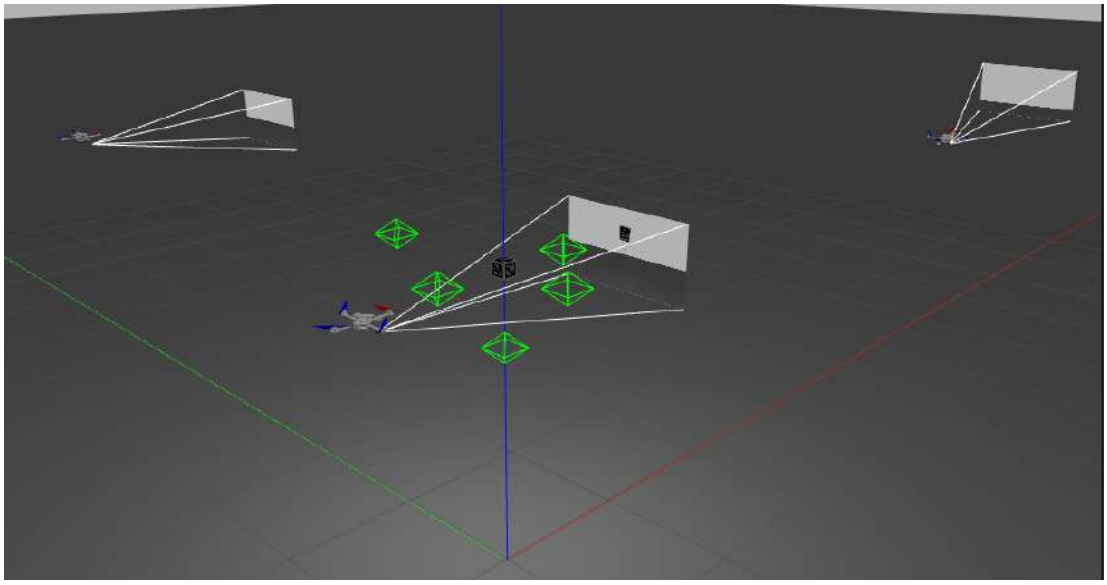

Figure 4.10: Simulation environment to simulate scheduling algorithm.

To emulate battery depletion and memory consumption, linear functions were implemented to continuously update each drone's battery and memory levels over time. LiPo batteries typically discharge in an approximately linear manner for most of their usage cycle. Based on data from [43] and [44], the battery level was estimated using the function $battery\ left = 100 - 6.25t$. For simulation purposes, the slope was scaled by a factor of 100 to accelerate the process. In the case of memory consumption, a slope of -2 was used, selected to be slightly less steep than the battery depletion rate. This reflects the typical scenario where drones tend to run out of battery before exhausting their available memory.

With this simulation, every cycle the priority scores are calculated and sorted by highest priority

because the EDF algorithm states that the most urgent goes first. Every time a simulation ends, a plot is generated to study how the priority scores change over time.

The implementation on hardware was also done, with the differences from the simulation being that the actual drone's battery and memory usage are tracked, and the odometry corresponds to the real sensors. However, testing was not possible due to the lack of available drones, but the individual workings of the code were tested, like receiving the battery, memory and calculating the distance from the charger. Thus, the program is done but with testing left.

4.3 Outside testing

The solution needs to be tested outside the laboratory environment, without relying on lab-specific sensors such as the Vicon system. While the drone already transmits GPS and odometry data, the board with the tag does not. Based on the client's recommendation, a setup was implemented using the OpenPilot Revolution flight controller [45], a GPS module, and a Raspberry Pi to estimate the board's state.

First, the center of the tag was measured with a ruler (Figure 4.11). Then, the GPS module was placed 15 cm away from this center point, as shown in Figure 4.12.

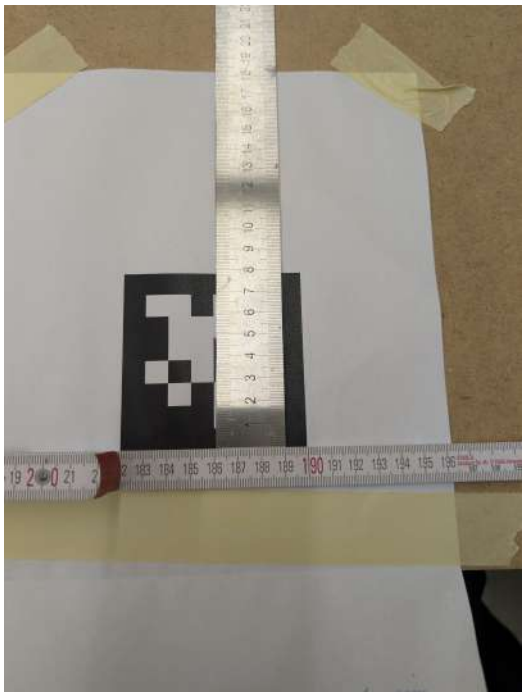

Figure 4.11: Image of tag's center identification.

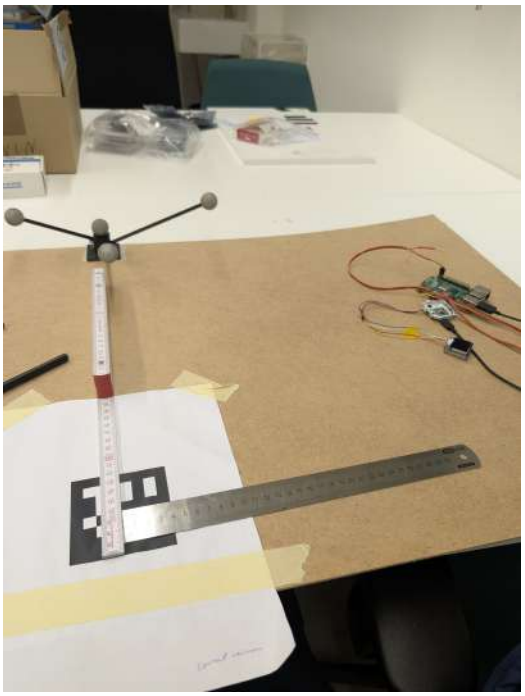

Figure 4.12: Image of GPS positioning from tag's center.

The flight controller was mounted near the GPS. Its exact position isn't critical as long as it's close enough, since we mainly need it for orientation measurements. The final outdoor setup is shown in Figure 4.13.

Since odometry is now based on GPS, localization isn't as straightforward as before. We need to convert GPS coordinates into a more familiar frame like (x, y, z, roll, pitch, yaw). The OpenPilot controller uses the North-East-Down (NED) frame, so that's what we used here.

First, the GPS coordinates are converted into ECEF (Earth-Centered, Earth-Fixed) coordinates using Python's pyproj library [46], which is built for this kind of coordinate transformation. This is done for both the board's coordinates and a fixed "home" position, which acts as a reference point. The difference between these two in ECEF is then converted into NED using the following rotation matrix [47]:

$$DCM_{ef} = \begin{bmatrix} -\sin\mu\cos\iota & -\sin\mu\sin\iota & \cos\mu \\ -\sin\iota & \cos\iota & 0 \\ -\cos\mu\cos\iota & -\cos\mu\sin\iota & -\sin\mu \end{bmatrix}$$

Here, μ is the latitude and ι is the longitude. To get the NED coordinates, this matrix is multiplied by the ECEF position vector. An example of how the home and the board's position were visualized in the controller's interface is shown in Figures 4.14 and 4.15.

The home and board's locations were written in the code due to simplicity and because these positions would not be moving in the experiment, but the ground was laid to use actual data from the controller. The locations are (latitude, longitude, altitude):

Figure 4.13: Image of setup for testing outside.

Figure 4.14: Image of the flight controller's interface showing the home's GPS location.

```
home_gps = (48.750186, 9.105734, 514.354)
borad_gps = (48.750231, 9.105686, 473)
```

Figure 4.15: Image of the flight controller's interface showing the marker's GPS location.

Regarding the other variables, because of the same reason (simplicity), they were all written as zero: linear and angular velocity and orientation. But for future usage, the implementation of this data was still worked. In this case the covariance comes in a 10×10 matrix—it includes position, linear speed and orientation in quaternions, but the standard for the odometry messages is of two separate 6×6 matrices, one for pose, in which the orientation covariances is for Euler angles, and the other for velocities.

So these matrices have to be built manually as follows:

$$C_{pose} = \begin{bmatrix} & & & 0 & 0 & 0 \\ & C_{position} & & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ 0 & 0 & 0 & & & \\ 0 & 0 & 0 & & C_{orientation} & \\ 0 & 0 & 0 & & & \end{bmatrix}$$

$$C_{velocity} = \begin{bmatrix} & & & 0 & 0 & 0 \\ & C_{linear} & & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$C_{position}$ and C_{linear} are taken directly from the output. There is no $C_{angular}$ from the output. The covariance matrix for the orientation expressed in Euler angles has to be calculated as follows, from [48]:

$$C_{Euler} = GC_qG^T$$

where C_{Euler} is the covariance matrix for the Euler angles (3×3), G is the Jacobin matrix containing the partial derivatives of the Euler angle equations with respect to the quaternions and C_q is the covariance matrix for the quaternions (4×4).

Also, the orientation is set to (0, 0, 0) degrees just as with inside tests, but the actual information from the controller's state estimation has orientation and twist information

This is implemented in the same frames transformation node as with the inside tests, but now the drone's position is taken from its GPS and is also transformed with respect to the home location, just as with the board. For the orientation, the inertial odometry data is used. All the data is fed again into the filter.

Chapter 5

Results and Analysis

5.1 Concepts validation

Here continues the design methodology with the validation tests design. Now that a concept has been chosen and implemented by taking into account the client's needs with their relative importance, determining their completion is needed. This is the concept validation.

Besides evaluating the completion of the specifications, if they were not satisfied, the validation allows determining the changes that could be made to the concept in order to satisfy them. Maybe these correspond to details in the concept description that were omitted.

For the validation, it is not enough to have a prototype, but tests must be done, which have a validation objective, influence factors with test values, the execution, and analysis and presentation of their results.

Due to time constraints, only the project's objectives (Sec. 1.4) are validated, but the general framework is still designed to validate the rest of the project's specifications in the future.

5.1.1 Objective definition

The validation objectives come directly from the client's requirements (Section 3.2) and their measuring variables or inputs come from the specifications (Section 3.3). These objectives are the ones to be validated, and in case they are not accomplished, it can be analyzed if modifications in the specifications is possible.

Table 5.1 shows how from each need and specification, the sampling objectives and variables were determined.

Table 5.1: Validation objectives with their sample variables.

Number	Requirement	Objective	Sample variable	Specification
1	The relative localization is precise	Localization precision	<5 cm	Error respect to ground truth
2	The localization uses the existing hardware	Amount of hardware used that is new and existing	0 %	Hardware utilization compliance
3	The localization is fast	Localization speed	>2 Hz	Cycle frequency
4	The localization of the larger platform in respect to the drone is relative	Amount of localization methods used, relative and absolute	>50 %	Relative localization ratio
5	The localization is resilient against signal interruptions	Localization's resilience	>60 s	Time of signal interruption necessary to fail localization
6	The coordination is centralized	Algorithm's centralization level	1	Number of independent processing units that make decisions in the swarm
7	The coordination is simple	Algorithm's complexity	<4	Number of parameters needed for calculation
8	The coordination is not interdependent	Algorithm's interdependency	None of them	Number of interdependent decisions
9	The coordination works with respect to previously defined priorities	Amount of predefined priorities needed for the coordination algorithm	>3	Number of predefined priorities needed for the coordination algorithm
10	The algorithm is efficient	Algorithm's efficiency	O(n)	Big-O complexity

5.1.2 Influence factors establishing and assignment

The influence factors are characteristics inherent to the winning concept; they are those that, when varied or parameterized, will influence the measurement variables of the objectives to be validated. Likewise, these are divided into controllable and blocked factors—the former are self-explanatory, while the latter are those that the designers cannot control.

In this case, it is decided to identify the influencing factors based on the objective(s) that are to be measured in a given test. The following sections describe a test for each objective with its influence factors.

5.2 Relative localization validation indoors

5.2.1 Test set-up

The validation objective here is the first one: localization precision. The subsystem responsible for this measurement is the AprilTag detection system, and according to [49], some influential factors for the AprilTag accuracy are:

1. Distance from the tag (position, three coordinates).

2. Viewing angle (orientation, three coordinates).
3. Tag size.
4. Tag rotations.
5. Border occlusion.
6. Overall visibility.
7. Lightning conditions.

The first four factors are controllable, whereas the rest are blocking factors, so they cannot be controlled in real-life situations. In this case, the most important factor identified to test is the first one—tag distance or position. This way, the following experiment was designed:.

To test the hardware with the covariance mapping done in Sec. 4.1.2, the same three positions and heights labeled as "left", "center", "right", and "bottom", "center", "top" are used. Some of them are shown in Figs. B.1 to B.3. These random positions were not exactly the same as for the covariance mapping because the mapping was done with ground truth data, so this time the tag detection had to detect the position of the tag as the ground truth did before.

5.2.2 Tests results

These positions were simulated in software, but even though the same camera parameters were introduced into the simulation, it was not possible to detect the apriltag further than 1 meter away. This is due to the low resolution the simulation has. The drone in real life records in 4k, which is significantly higher resolution than the simulation's.

The hardware test was possible, resulting in more than 100 measurements per position and height so that a significant sample of the population could be obtained. Measurements in the x, y and z-axis were recorded for the filter, ground truth, and error between them. The summary of averages and standard deviations are shown in Tables 5.2 to 5.7.

Table 5.2: Average summary of filter measurements for hardware validation.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
bottom_center	0.0592	3.1883	0.3415	75.5775	-0.2185	-3.2977
bottom_left	1.1634	2.2901	0.3732	72.472	-0.4514	0.7335
bottom_right	-1.2882	2.1291	0.3301	73.7185	0.0772	-4.3693
center_center	-0.0995	3.1539	0.1001	90.906	-0.7012	0.3794
center_left	1.2165	2.2396	0.278	72.6461	-0.3941	-1.2466
center_right	-0.9735	2.094	-0.0537	73.4484	-0.1554	-0.3555
top_center	-0.0314	3.1069	-0.5665	79.9783	0.9907	2.4674
top_left	1.2379	2.2569	-0.524	72.2198	-0.2759	0.1432
top_right	-0.9022	2.1156	-0.561	72.6255	-0.9741	-3.3205

Table 5.3: Standard deviation summary of filter measurements for hardware validation.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
bottom_center	0.0013	0.0142	0.003	3.6779	0.164	1.4635
bottom_left	0.0039	0.0064	0.0027	1.6387	0.3222	2.1718
bottom_right	0.0017	0.0027	0.0011	0.1211	0.0646	0.0702
center_center	0.0022	0.0139	0.0047	5.9872	0.354	1.1702
center_left	0.0032	0.0049	0.0023	2.998	0.4324	4.1062
center_right	0.0018	0.0037	0.0014	0.1714	0.1253	0.1737
top_center	0.0015	0.0132	0.003	1.1597	1.549	3.6436
top_left	0.0024	0.0038	0.0014	0.1662	0.1499	0.1457
top_right	0.001	0.0023	0.0014	0.1984	0.1266	0.1121

Table 5.4: Average summary of error measurements for hardware validation.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
bottom_center	0.4574	-0.2687	-0.0694	0.6739	-9.3766	-10.1418
bottom_left	0.3223	-0.3505	-0.052	4.3978	-8.1553	-8.6606
bottom_right	0.3159	-0.0064	-0.0297	3.6982	-9.7374	-8.0508
center_center	0.4522	-0.2444	-0.0377	-13.4055	-8.889	-13.0744
center_left	0.3199	-0.3559	-0.0478	3.5994	-9.2866	-8.7442
center_right	0.3003	-0.0181	-0.0801	2.6534	-9.0793	-6.9286
top_center	0.4752	-0.2287	-0.1286	-4.1244	-11.186	-14.6916
top_left	0.3137	-0.3634	-0.1685	1.8429	-7.9104	-8.9008
top_right	0.3234	-0.0124	-0.1243	2.3428	-9.3964	-6.0777

Table 5.5: Standard deviation summary of error measurements for hardware validation.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
bottom_center	0.0022	0.0189	0.0044	4.0657	0.2605	1.4023
bottom_left	0.0047	0.0066	0.0047	1.9925	0.7488	2.2585
bottom_right	0.0018	0.0029	0.0015	0.1299	0.0733	0.0725
center_center	0.0035	0.0142	0.0089	5.807	0.3699	1.2467
center_left	0.0041	0.0053	0.0041	2.5	1.3729	2.8788
center_right	0.0018	0.0038	0.0015	0.1722	0.1142	0.1782
top_center	0.0027	0.0131	0.003	1.8476	1.7113	3.66
top_left	0.0033	0.0041	0.0016	0.1854	0.1521	0.1635
top_right	0.0011	0.0025	0.0015	0.2178	0.1163	0.1155

Table 5.6: Average summary of ground truth measurements for hardware validation.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
bottom_center	0.5167	2.9205	0.2723	76.8956	-9.5198	-13.4227
bottom_left	1.4855	1.9396	0.3211	76.8097	-8.7239	-8.0011
bottom_right	-0.9723	2.1228	0.3004	78.1495	-9.3491	-12.4627
center_center	0.3527	2.9093	0.0623	77.5614	-9.3644	-12.5064
center_left	1.5363	1.8837	0.2301	76.3123	-9.7262	-9.8732
center_right	-0.6731	2.0757	-0.1338	76.1614	-9.2177	-7.2999
top_center	0.4436	2.8774	-0.6949	75.3408	-10.1359	-12.2779
top_left	1.5516	1.8934	-0.6925	74.0447	-8.1934	-8.766
top_right	-0.5789	2.1032	-0.6853	75.5213	-10.2166	-9.4906

Table 5.7: Standard deviation summary of ground truth measurements for hardware validation.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
bottom_center	0.0014	0.0004	0.0026	0.0536	0.0372	0.0337
bottom_left	0.0022	0.0017	0.0032	0.1006	0.0347	0.0617
bottom_right	0.0002	0.0002	0.0007	0.0242	0.0247	0.0102
center_center	0.0022	0.0003	0.0067	0.1281	0.1099	0.0623
center_left	0.002	0.0017	0.0029	0.0956	0.0556	0.0643
center_right	0.0001	0.0002	0.0001	0.0217	0.0044	0.0072
top_center	0.0021	0.0006	0.0008	0.0266	0.0292	0.0404
top_left	0.0019	0.0014	0.0007	0.031	0.0255	0.0505
top_right	0.0001	0.0002	0.0001	0.0253	0.0052	0.0085

The validation consists of determining if the accuracy or error is less or equal than 10 cm, so the

following null and alternative hypotheses are to be tested, where μ is the population average:

$$H_0 : \mu > 0.1\ m$$
$$H_1 : \mu \leq 0.1\ m$$

It would be ideal to prove the alternative hypothesis for all the populations, considering them as if they behaved similarly and the error did not change with the position. But, this cannot be assumed, so first an ANOVA should be done to actually see if they behave similarly, and then prove the original hypothesis individually or collectively according to the former result. Also, a confidence level of 90% will be used for every statistical test, since the client did not want a particularly high confidence level.

Normality check

First, data normality of the error measurements is checked with a Q-Q plot for each axis, for each position—Figures 5.1 to 5.3 show these for a single position, but a similar trend was observed for all the data. It can be seen that the data behaves closely to a normal distribution, with just the borders going outside the distribution. This means that there are outliers, but this is more common for samples as big as these, with more than 100 samples—in these cases, the outliers cause big deviations in distribution tests. But also, because the sample size is high, normality will be assumed from now on.

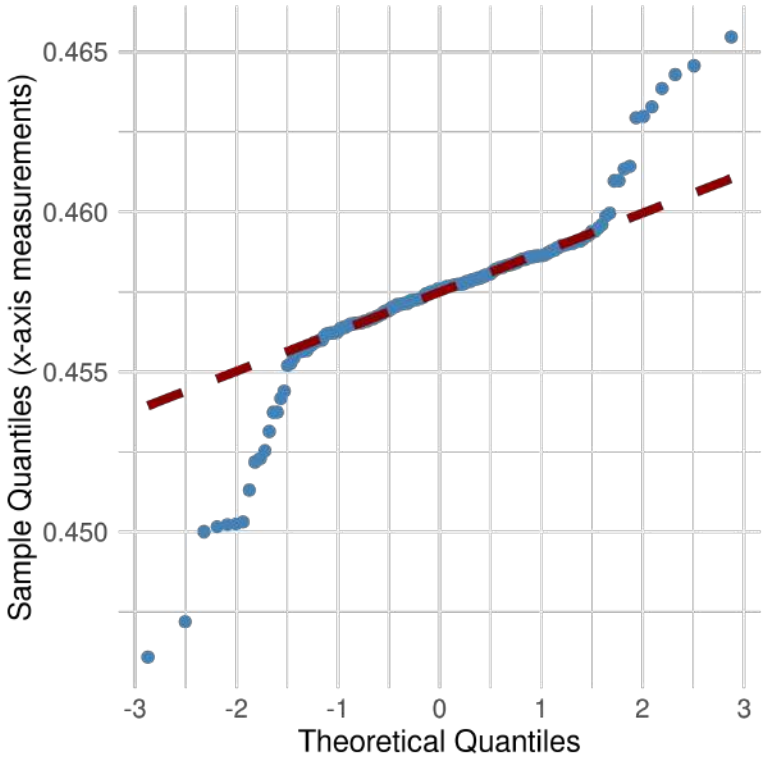

Figure 5.1: Q-Q plot of error data for x-axis measurements in the bottom center position.

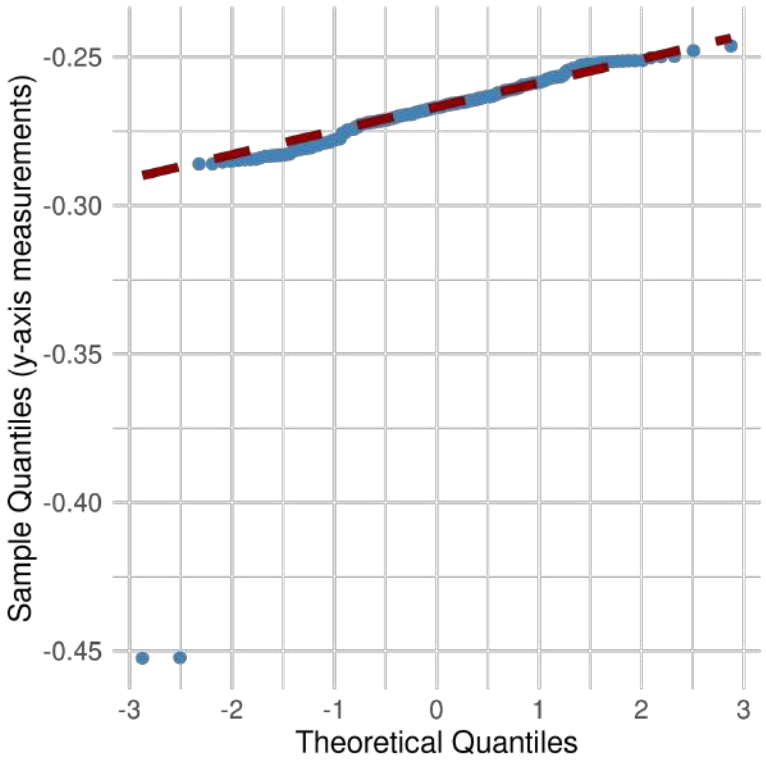

Figure 5.2: Q-Q plot of error data for y-axis measurements in the bottom center position.

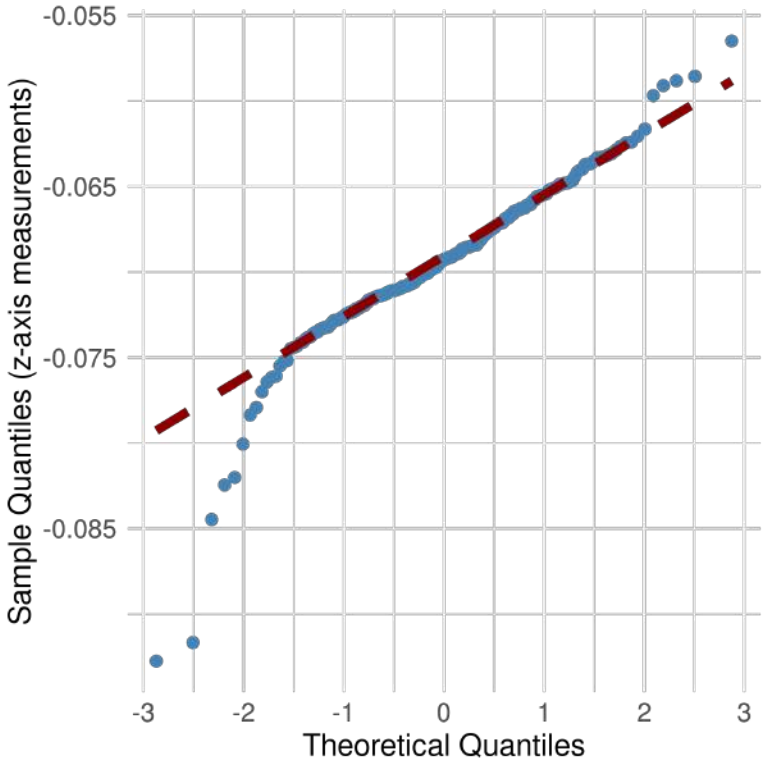

Figure 5.3: Q-Q plot of error data for z-axis measurements in the bottom center position.

ANOVA

Now the ANOVA null hypothesis is raised for the nine positions' error average:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_9$$
$$H_1 : \mu_1 \neq \mu_2 \neq \ldots \neq \mu_9$$

The results are shown in Table 5.8. The F-statistic is high, this implies that there is a big difference in the variances between the groups compared to the variance within the groups, but also the p-value is essentially zero. This indicates that there is overwhelming statistically significant evidence to reject the null hypothesis, i.e., the average error in the x, y and z-axis across the different positions are not all the same.

Table 5.8: ANOVA results for each axis.

Axis	**F**	**p(>F)**
x	138208	$2\text{x}10^{-16}$
y	61654	$2\text{x}10^{-16}$
z	31092	$2\text{x}10^{-16}$

When looking at the standard deviation of the error measurements for every position and axis, this result is consistent, for they vary a lot between themselves. In simpler words, this results means that indeed the position of the camera from the AprilTag across the different axes has different effects on the pose detection accuracy.

The tag is a flat surface, and as the camera moves to different positions, a distortion occurs to this flat surface, resulting in different accuracy values for each position. Also, as the tag has a very distinct geometry, it is expected that the distortion affects differently each axis. This is an intrinsic problem of using a normal camera, ideally, focal distance should be as big as possible, but this cannot be varied for the drone. An alternative that could reduce a little the distortion's effect would be a calibration for the distortion.

From now on, the original hypothesis for the system's accuracy will be proved separately for each position.

Hypothesis test for every position

Now the main hypothesis shall be tested. For this, a test statistic has to be chosen, and since normality is assumed, the Z-statistic is chosen. For a sample mean [50]:

$$Z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}} \tag{5.1}$$

where $\bar{x}$ is the sample mean, μ_0 is the hypothesized mean (0.1 in this case), σ is the population deviation, but for large mean samples, the sample deviation can be used, and n is the sample size.

With this, the critical Z value can be calculated as

$$Z(1-\alpha) = Z(0.9) = -1.2816$$

Finally, with $p = P(Z < Z\text{-statistic})$, the Tables 5.9 to 5.17 show the results for the nine positions. Aside from the afore-mentioned parameters, the two criteria given in [50] to estimate whether the null hypothesis is rejected are given, as well as the actual rejection or not rejection.

Table 5.9: Z-statistic and p-value for position bottom center.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 rejected?
x	0.4574	2492.1093	1	No	No	No
y	0.2687	140.2192	1	No	No	No
z	0.0694	-108.2166	0	Yes	Yes	Yes

Table 5.10: Z-statistic and p-value for position bottom left.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.3223	740.3323	1	No	No	No
y	0.3505	589.5249	1	No	No	No
z	0.052	-157.8029	0	Yes	Yes	Yes

Table 5.11: Z-statistic and p-value for position bottom right.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.3159	1923.3528	1	No	No	No
y	0.0065	-531.2825	0	Yes	Yes	Yes
z	0.0297	-719.5277	0	Yes	Yes	Yes

Table 5.12: Z-statistic and p-value for position center center.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.4522	1601.1352	1	No	No	No
y	0.0065	-531.2825	1	No	No	No
z	0.0297	-719.5277	0	Yes	Yes	Yes

Table 5.13: Z-statistic and p-value for position center left.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.3199	829.0211	1	No	No	No
y	0.3559	749.2341	1	No	No	No
z	0.0478	-200.2545	0	Yes	Yes	Yes

Table 5.14: Z-statistic and p-value for position center right.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.3003	1760.9848	1	No	No	No
y	0.0181	-342.8506	0	Yes	Yes	Yes
z	0.0801	-205.4167	0	Yes	Yes	Yes

Table 5.15: Z-statistic and p-value for position top center.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.4752	2003.435	1	No	No	No
y	0.2287	140.4049	1	No	No	No
z	0.1286	137.6156	1	No	No	No

Table 5.16: Z-statistic and p-value for position top left.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.3137	999.3004	1	No	No	No
y	0.3634	1011.2172	1	No	No	No
z	0.1685	650.0697	1	No	No	No

Table 5.17: Z-statistic and p-value for position top right.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 **rejected?**
x	0.3234	3229.0547	1	No	No	No
y	0.0124	-553.5929	0	Yes	Yes	Yes
z	0.1243	250.5473	1	No	No	No

The results are consistent when looking at the error mean for each position and axis. As the null hypothesis is rejected in most cases, there is not enough evidence yet to prove that the system's accuracy is less than 10 cm in all axes and positions. To gain more evidence, a power analysis is now performed.

Power analysis

Power is the probability of correctly rejecting a false null hypothesis [50]. When failing to reject the null hypothesis, a low power could mean that the study was underpowered and more samples would be necessary.

To calculate it, an alternative "true" mean value is chosen such that if the hypothesized value of 10 cm were true, the alternative value should also be true. This way, $\mu_a = 0.08$ is chosen, slightly smaller than the expected value. This way the Z-statistic is calculated with the alternative mean, $\beta = P(\mu > 0.1 \text{ when } \mu = 0.8)$ is calculated as the Type II probability, and finally

$$Power = 1 - \beta$$

The results for every position and axis yielded a power equal to 1 for everyone. This could be compared to a common power value like 80% to indicate whether the amount of samples was adequate, but since the power is of 100% it is a strong evidence of a good sampling amount.

Validation result

With all the results there is considerable evidence indicating that the system as a whole does not achieve the desired accuracy of 10 cm or less in the current setup. The axis that did achieve in most cases this value was the vertical (z-axis). Since the difference in heights was not that significant and the three of them were near the center, it is possible that the afore-mentioned distortion did not affect this axis as much as the x-axis that did have more extreme values.

Aside from the distortion, the accuracy was probably affected mainly by the other influencing factors. The visibility seems good, at least to the human eye; an image histogram would be necessary to analyze the contrast recorded by the camera. The drone's orientation was mainly kept at 0°, so it is not likely that this had an effect on the low accuracy.

One of the two most likely reasons for the low accuracy is the tag size, that might be too small, translating to few tag's pixels to estimate its pose. The other likely reason is that the Vicon system's measurements are too precise compared to the tag detection, causing this relative low accuracy, but it might be that compared to the odometry measured outside the laboratory, the tag detection is actually accurate.

Additional notes on the general results from Tables 5.2 to 5.7 are the following: it is possible to see that the biggest error was in the x-axis, the error got to a maximum of 47.5 cm and was always greater than 10 cm. In the y-axis, there was not such a big error, but still most of them were more than 10 cm. Finally, on the z-axis only in three occasions was the error greater than 10 cm.

Other patterns noticed are that the z-axis got noisier when the drone was above the tag. Also, the y-axis had the lowest errors when the drone was on the right of the tag.

Beyond the accuracy, a good precision is noted for the filter, with only in three scenarios being the standard deviation on the order of 1 cm.

5.3 EDF algorithm validation

Validating this objective in hardware was not possible due to the reasons mentioned in Sec. 4.2, so the validation was done in software. This validations was focused on exploring the algorithm behaviour under different circumstances. The influence factors chosen for validation are the ones that have a weight assigned: distance to charger, battery and memory left. These are blocking factors because the drones will be doing a mission and the operator should not control the distance to the charger because it would alter the mission.

Simulation speed was incremented compared to real life by introducing bigger slopes into the battery and memory depletion time functions, specifically in the order of thenths or hundreds. This was done becaus it is of no interest in simulation to see the behaviour for several hours, as it will be similar as if it is proportionally accelerated like in this case. The three drones in the simulation are called "hummingbird", with the differentiator being the numbering (from 1 to 3). The analysis is generally qualitative, with some quantitative analysis at the end.

5.3.1 Only distance changes

In this experiment, hummingbird 1 started one meter apart from the box, hummingbird 3 started 5 m apart from the box, and hummingbird 2 started 10 m apart from it. Then, hummingbird 1 moved one meter further, then hummingbird 2 moved one meter closer, and this cycle was repeated until the first one got to 10 meters apart from the box, which was the previously-defined distance limit of the drone. The battery and memory levels were kept constant to 100% to just explore the effects of distance changes.

Figure 5.4 shows the simulation result. It shows the expected behavior: the second drone starts with the highest priority, as it is farther, but gets a lower score as it gets near. The first drone has the contrary result because its behavior is exactly opposite to the second's. The third has a constant score as it stays in the same position.

Results similar to this are to be expected on all experiments because the influence factors vary linearly, and the score changes directly proportional to these factors. But this only happens because they all have the same weight, so mathematically the score proportionality looks like

$$priority \propto w(distance + battery + memory)$$

And actually, since two of them are constant, it ends up being completely proportional to the distance change. In this case, the change in distance happens incrementally, but if it happened exponentially or sinusoidally, then so would the change in the priority score.

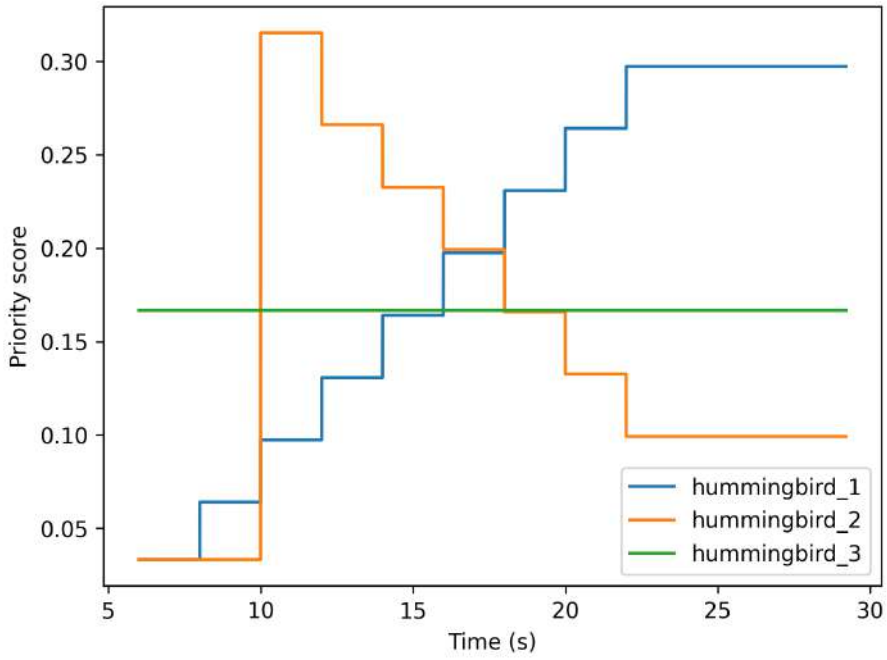

Figure 5.4: EDF algorithm simulation with three drones starting from differnet positions.

5.3.2 Only battery changes

Here hummingbird 1 started with 100% of battery, hummingbird 2 with 60% and hummingbird 3 with 35%. The distance from the charger was of 1 meter and the memory was at 100% for all of them. Figure 5.5 shows the result. This result is explained by exactly the same phenomena as in the previous test, the difference is that the increments here are smaller, thus looking more continous. These lines have the battery discharge rate's slope multiplied by the weight—$score \propto -6.25t * 0.33 \approx -2.06t$.

5.3.3 Only memory changes

For this one, hummingbird 1 started with 60% of memory, hummingbird 2 with 40% and hummingbird 3 with 25%, while mantaining the same distance as the previous test and the battery cosntant to 100%. The result is shown in Fig. 5.6. The result is just like the previous one but with a different slope—the memory filling rate multiplied by the weight. This is $priority \propto 0.33 * -2t = -0.66t$.

5.3.4 Distance and memory change

For this one, hummingbird 1 started with 80% of memory, hummingbird 2 with 90% and hummingbird 3 with 70%, while moving in the same pattern, with the same initial position as in the first test, and the battery constant to 100%. The result is shown in Fig. 5.7. This presents a result similar to the first, with the difference being that now the increments (or reductions)

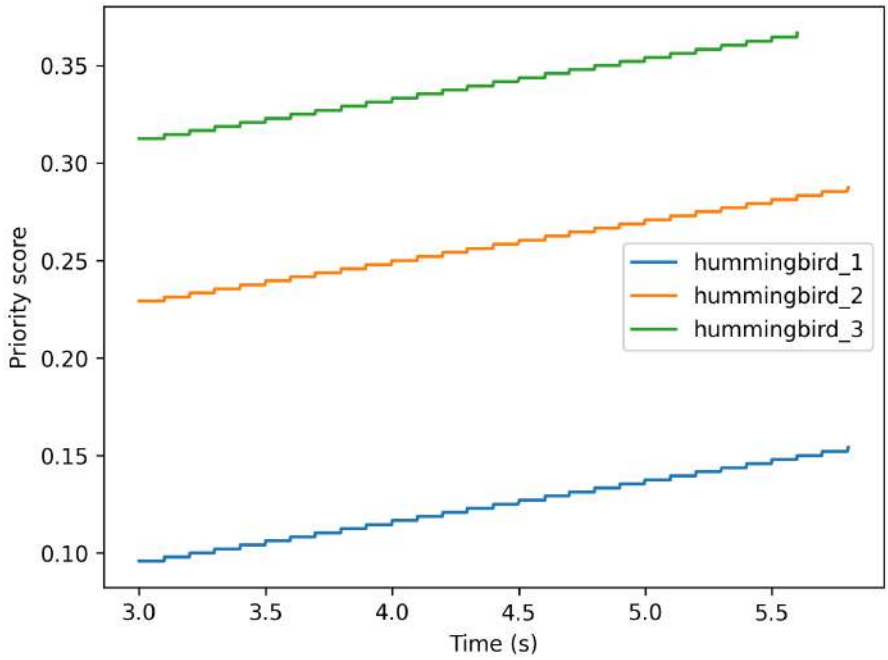

Figure 5.5: EDF algorithm simulation with three drones with different starting battery levels.

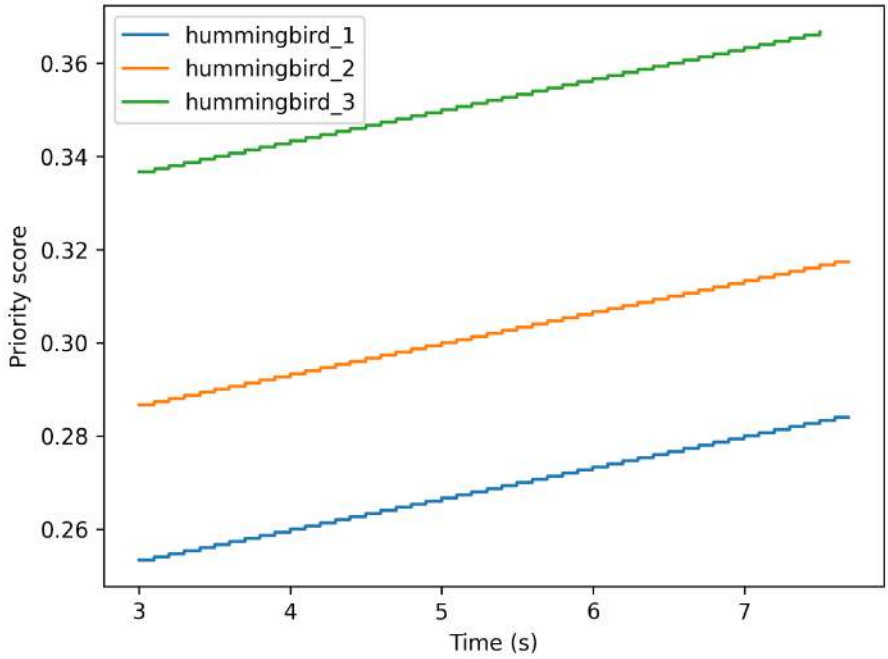

Figure 5.6: EDF algorithm simulation with three drones with different starting memory levels.

also have the memory's slope. The same rules as before regarding the proportionality applies, resulting in the slope of $\approx -0.66t$.

The test where distance and battery change would exhibit a similar behavior, with the difference being that the slope would be the battery's again, $\approx -2.06t$.

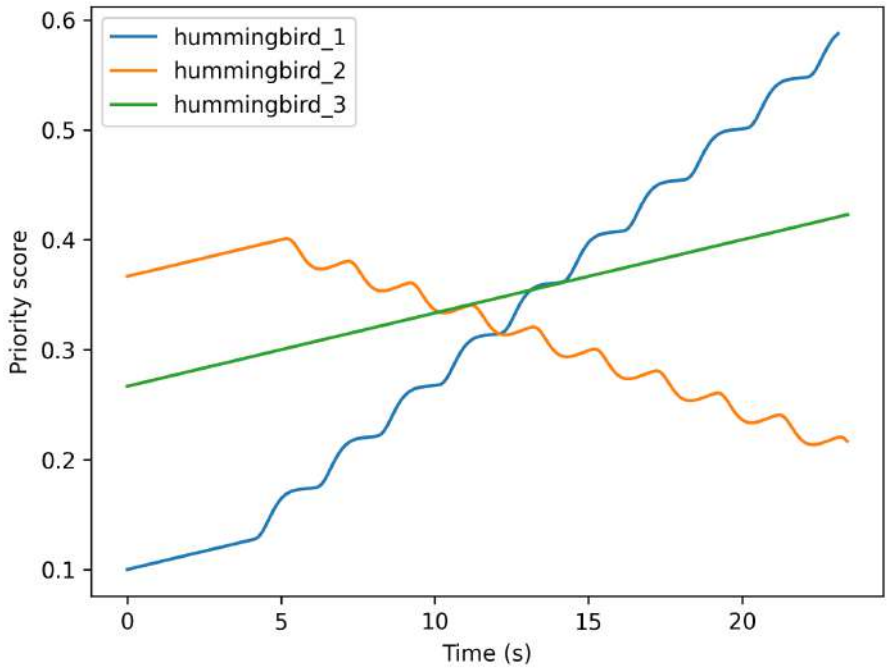

Figure 5.7: EDF algorithm simulation with three drones with different starting memory levels and different distance to the charging port.

5.3.5 Battery and memory change

For this one, hummingbird 1 started with 100% of battery, hummingbird 2 with 70% and hummingbird 3 with 85%, with the same initial memory as in the previous test, and the distance constant at 1 m from the box. The result is shown in Fig. 5.8. In this test, the slopes of the battery and memory are summed due to the distribution principle of products, so $score \propto 0.33(-6.25t - 2t) \approx -2.475t$, being a bigger slope than before, but with the same behavior as the individual parameters—just a slope.

5.3.6 All variables change

By changing all the variables, the results in Figure 5.9 and Tables 5.18 to 5.20 are obtained. The initial state was:

- **hummingbird_1:** Distance: 1 m; battery: 100 %, Memory: 80 %.
- **hummingbird_2:** Distance: 10 m; battery: 70 %, Memory: 90 %.
- **hummingbird_3:** Distance: 5 m; battery: 85 %, Memory: 70 %.

The incremental or decreased behavior from the change in distance is smoothed by the slopes of battery and memory combined. They still have a linear tendency, but it is more complex than before. The simulation stopped because a drone ran out of memory, stopping the simulation,

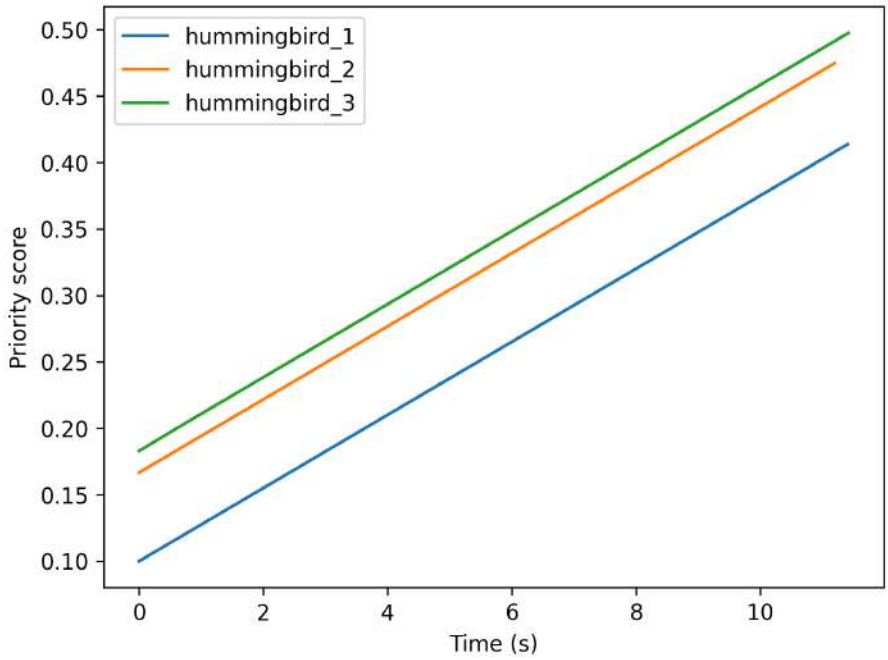

Figure 5.8: EDF algorithm simulation with three drones with different starting battery and memory levels.

but it is likely that hummingbird 3 would have surpassed 1's score and 1's score would have gone below the second's, which is always constant because it does not have the incremental behavior of the others.

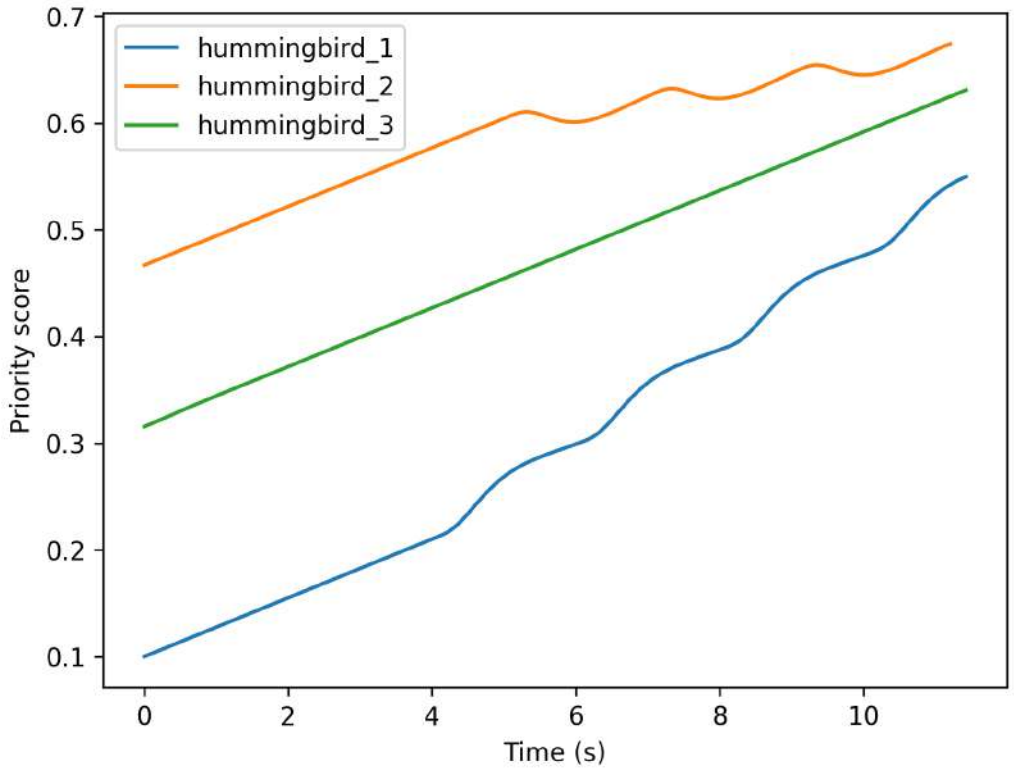

Figure 5.9: Change of priority score when all variables change.

From the numeric results, the linearity can also be seen, because as some parameters decrease, so does the score increase, but in this little simulation time, a change of priority could not be observed.

Table 5.18: First priority results of EDF algorithm with all variables changing.

Time (s)	Drone	Score	Distance (m)	Battery (%)	Memory (%)
0	hummingbird_2	0.467	3.15	70	90
2.79	hummingbird_2	0.543	6.34	52.5	84.4
5.6	hummingbird_2	0.606	9.98	35	78.8
8.38	hummingbird_2	0.628	10	17.6	73.2

Table 5.19: Second priority results of EDF algorithm with all variables changing.

Time (s)	Drone	Score	Distance (m)	Battery (%)	Memory (%)
0	hummingbird_3	0.315	4.12	85	70
2.79	hummingbird_3	0.394	3.43	67.3	64.3
5.6	hummingbird_3	0.471	5.01	49.9	58.8
8.38	hummingbird_3	0.548	5	32.4	53.2

Table 5.20: Third priority results of EDF algorithm with all variables changing.

Time (s)	Drone	Score	Distance (m)	Battery (%)	Memory (%)
0	hummingbird_1	0.1	2.45	100	80
2.79	hummingbird_1	0.178	1.06	82.3	74.3
5.6	hummingbird_1	0.29	1	64.8	68.7
8.38	hummingbird_1	0.406	1	47.2	63.1

The EDF algorithm was validated through different behavioral experiments, exploring that it works as expected in different scenarios. The simulation speed was increased because the conclusion from these results are also valid for longer experiments in real life. Even though all results were fairly linear, in real life the battery depletion rate could have non-linear zones, as well as the memory filling rate, but it was proved that the score is proportional to both rates, so it will be directly affected.

5.4 Relative localization validation outdoors

5.4.1 Test set-up

This experiment is almost the same as in Sec. 5.2. The influence factors are the same, but in this case there is a dyanamic, uncontrollable light source (the Sun). Also, due to space constraints, it was not possible to emulate the positions from the inside tests. So five positions were discretized from the far left to the far right of the image frame, labeling them as "most-left", "mid-left", "center", "mid-right" and "most-right". Also two different heights were teste for the same positions, but closer to the board, adding to the label "top" or "down" accordingly. In Figure 4.13 the center-top position is shown and in Figure 5.10 the center-down position is shown.

5.4.2 Test results

The same procedure will be used to try to prove the alternative hypothesis that the mean of the error values are less or equal to 10 cm. First, in Tables 5.21 to 5.24 the testing results can be seen. Also, because of the same reasons, the significance level will be of 90% (meaning $\alpha = 0.1$).

$$H_0 : \mu > 0.1\ m$$
$$H_1 : \mu \leq 0.1\ m$$

Figure 5.10: Image of configuration center-down.

Table 5.21: Average summary of error measurements for outside tests.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
center-down	0.0380	0.0362	0.0572	1.0673	0.1077	10.6573
center-top	0.0004	0.0107	0.0018	1.1744	0.4332	42.7448
mid-left-down	0.0076	0.0086	0.0400	1.0851	0.2652	5.1897
mid-left-top	0.0056	0.0329	0.0041	1.8253	0.6024	26.3534
mid-right-down	0.0100	0.0429	0.0770	1.2202	0.0947	4.9493
mid-right-top	0.0085	0.0510	0.0053	3.4770	1.3978	137.8612
most-left-down	0.0056	0.0085	0.1753	3.2417	0.9114	84.9476
most-left-top	0.0034	0.0132	0.0015	2.0901	0.4190	16.6005
most-right-down	0.0021	0.0057	0.0020	0.2207	0.1030	0.2370
most-right-top	0.0136	0.0437	0.0060	3.8800	1.4789	142.4265

Table 5.22: Standard deviation summary of error measurements for outside tests.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
center-down	0.3637	0.3128	0.5603	7.4349	0.1077	57.9922
center-top	0.0003	0.0077	0.0014	0.9485	0.3341	112.7531
mid-left-down	0.0879	0.0465	0.5390	6.5596	0.7366	39.1500
mid-left-top	0.0088	0.0531	0.0055	2.3855	0.7216	90.2293
mid-right-down	0.0811	0.3336	0.7347	7.3631	0.1123	37.0314
mid-right-top	0.0078	0.0480	0.0050	2.7674	1.0777	164.4050
most-left-down	0.0279	0.0346	1.6989	11.2464	2.4541	149.2563
most-left-top	00.0027	0.0108	0.0013	5.8012	0.8367	68.9307
most-right-down	0.0018	0.0047	0.0013	0.1990	0.0854	0.2010
most-right-top	0.0252	0.0724	0.0075	3.5447	1.6673	166.0027

Table 5.23: Average summary of filter measurements for outside tests.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
center-down	0.1229	2.8191	0.5429	-80.2763	1.7677	175.0241
center-top	-0.0557	4.1121	-0.4392	-76.3659	2.3359	-144.4396
mid-left-down	0.6228	2.7763	0.5372	-80.3957	2.1099	-175.3539
mid-left-top	0.647	4.1434	-0.4309	-74.7416	2.0302	-160.9742
mid-right-down	-0.4766	2.848	0.5293	-79.7929	1.5257	175.9067
mid-right-top	-0.6538	4.1672	-0.4292	-82.2866	1.1096	22.8472
most-left-down	1.4131	2.7308	0.5393	-80.5262	1.6483	107.0269
most-left-top	1.1148	4.151	-0.4208	-74.2502	2.2751	-165.9198
most-right-down	-1.0194	2.8863	0.5407	-79.583	1.588	176.2937
most-right-top	-1.2773	4.1593	-0.4499	-80.7053	1.4721	77.3309

Table 5.24: Standard deviation summary of filter measurements for outside tests.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
center-down	0.0008	0.0045	0.0015	0.2558	0.0951	36.7978
center-top	0.0007	0.0104	0.0022	1.2291	0.422	101.1901
mid-left-down	0.0012	0.0047	0.0014	2.9049	0.5572	23.725
mid-left-top	0.0077	0.0431	0.0048	2.1576	0.6608	65.184
mid-right-down	0.0016	0.0082	0.0021	0.4708	0.0853	25.8813
mid-right-top	0.0081	0.0505	0.0053	3.6079	1.2964	169.005
most-left-down	0.0033	0.0059	0.0015	6.4352	1.8869	143.1565
most-left-top	0.0042	0.0146	0.0014	3.927	0.6245	44.3429
most-right-down	0.002	0.0049	0.0047	0.2328	0.0867	0.2056
most-right-top	0.0212	0.0612	0.0075	3.8601	1.6426	155.1604

Table 5.25: Average summary of odometry measurements for outside tests.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
center-down	0.0645	2.8671	0.6521	-79.0925	1.7838	173.2962
center-top	-0.0557	4.1119	-0.4392	-76.3555	2.3385	-144.0343
mid-left-down	0.6178	2.7789	0.568	-80.052	2.1173	-174.5195
mid-left-top	0.6469	4.1431	-0.4309	-74.7273	2.0276	-161.083
mid-right-down	-0.4701	2.8746	0.588	-79.1911	1.5348	175.7583
mid-right-top	-0.654	4.1684	-0.4293	-82.4251	1.0801	18.5336
most-left-down	1.4152	2.7284	0.6756	-79.7511	1.7021	103.0618
most-left-top	1.1148	4.151	-0.4208	-74.3587	2.3006	-164.7698
most-right-down	-1.0194	2.8863	0.5408	-79.5845	1.5907	176.2974
most-right-top	-1.2766	4.1575	-0.4498	-80.7113	1.4874	77.2879

Table 5.26: Standard deviation summary of odometry measurements for outside tests.

Position	x (m)	y (m)	z (m)	roll (°)	pitch (°)	yaw (°)
center-down	0.4539	0.3758	0.8741	9.2524	0.1645	40.4453
center-top	0.0007	0.0104	0.0022	1.209	0.4245	101.6985
mid-left-down	0.0785	0.042	0.4804	5.3367	0.5046	27.9572
mid-left-top	0.0078	0.0431	0.0047	2.1542	0.6586	64.8675
mid-right-down	0.0718	0.2956	0.6492	6.5384	0.1171	23.6565
mid-right-top	0.0083	0.0513	0.0054	3.6103	1.2764	169.3027
most-left-down	0.0248	0.0318	1.5048	8.7572	1.7031	145.6952
most-left-top	0.0043	0.0149	0.0014	4.1624	0.6623	48.0597
most-right-down	0.002	0.0049	0.0047	0.2339	0.0883	0.2054
most-right-top	0.0203	0.0586	0.0073	3.8748	1.6362	155.158

Normality check

To perform further statistical tests, it is important to check normality. Once again, in Figs. 5.11 to 5.13 Q-Q plots are shown in which a very similar behavior across the axes is noticed. Just as before, in the center the data is similar to a normal distribution, but there are some outliers. This is not a problem because there is a big amount of samples (more than 100), so normality can be assumed from now on.

Also, these similarities with the tests indoors indicate that very likely there will be similar patterns compared to the previous tests.

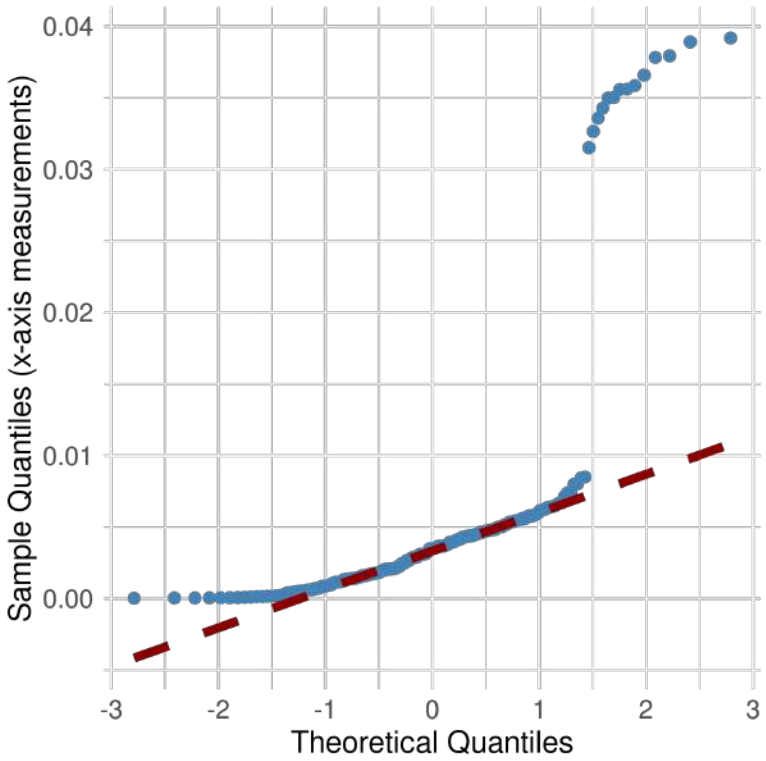

Figure 5.11: Q-Q plot of error data for x-axis measurements in the mid left top position.

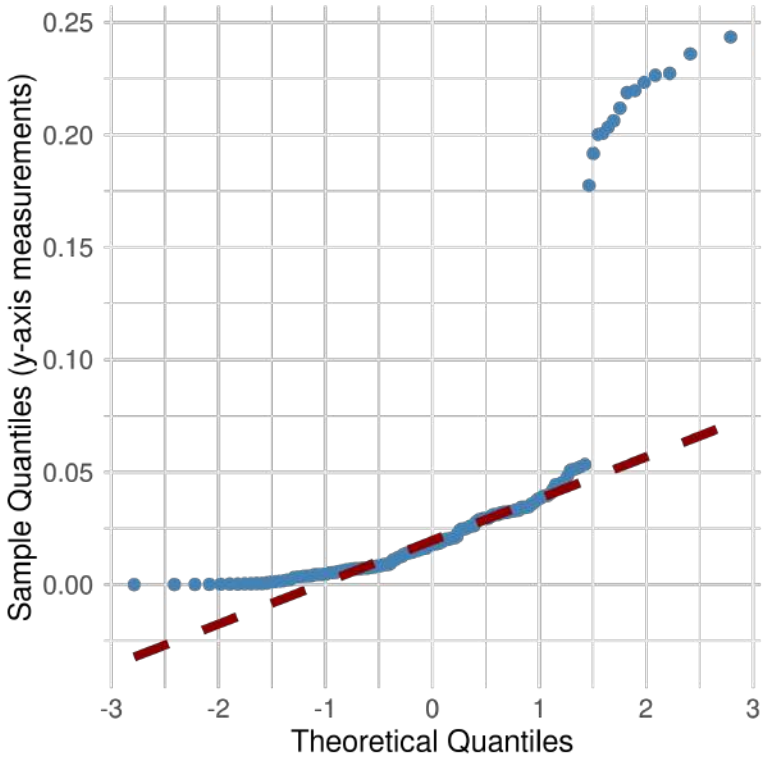

Figure 5.12: Q-Q plot of error data for y-axis measurements in the mid left top position.

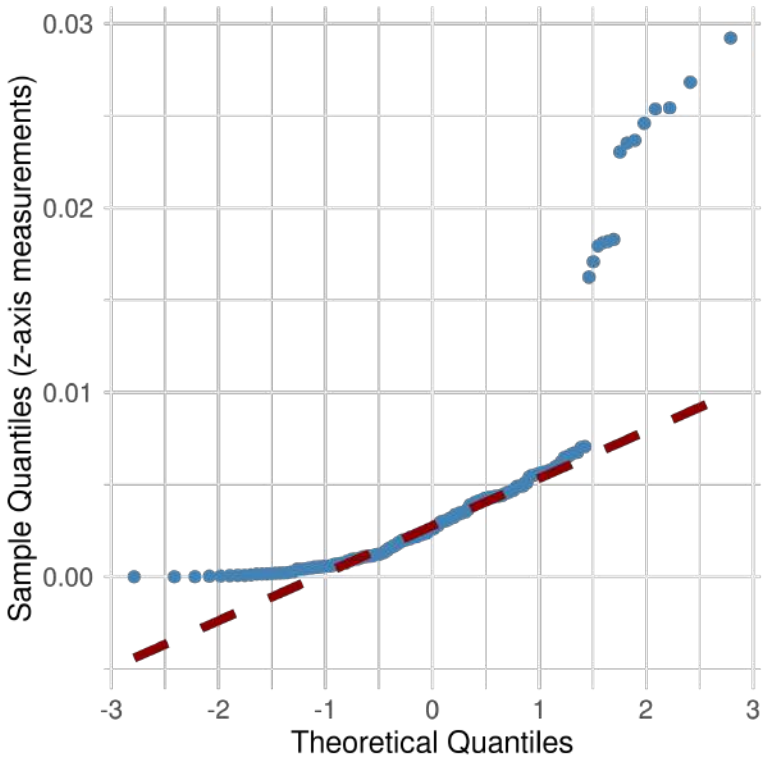

Figure 5.13: Q-Q plot of error data for z-axis measurements in the mid left top position.

ANOVA

Again, the ANOVA null hypothesis is raised for the ten positions' error average:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_1 0$$
$$H_1 : \mu_1 \neq \mu_2 \neq \ldots \neq \mu_1 0$$

In Table 5.27 the results of the ANOVA can be seen. Very differently from last time, the F values are lower, which indicates that the variances between the groups are not so different from the variances within the groups. The p-value is compared to α, and if it is greater than α, then there is no sufficient evidence to reject the null hypothesis. That is, any observed differences in sample means are likely due to random chance [50]. In this case, x and z-axis values will be treated as one only population from now on, whereas the y-axis population will be analyzed separately.

Table 5.27: ANOVA results for each axis.

Axis	F	p(>F)	$\alpha < p$?	Reject H_0
x	1.484	0.148	Yes	No
y	2.641	0.00486	No	Yes
z	1.462	0.157	Yes	No

5.4.3 Hypothesis test for every position

Just as before, the main hypothesis shall be tested. For this, the Z-statistic will be used because normality is assumed. For a sample mean (5.1) is used [50].

Following the same validation as before, Tables 5.28 and 5.29 show that the null hypothesis can be rejected for every position and axis of the system. This means that there is sufficient statistic evidence to say that the mean of the error of the system is less than or equal 10 cm, so the objective is successfully validated.

Notice that in this case, the biggest influence factor that changed from the first validation is the lightning conditions. This time the tests were performed with sunlight, although in shadow, but lumens were not measured to verify if this was the reason for good accuracy.

At this moment, other factors have to be included in the equation, because now there is no ground truth, so the output does not depend solely on the tag detection. There is odometry from both the tag board and the drone, so there are new factors: odometry precision and GPS precision.

The GPS location is less accurate than the Vicon system, and in this case, the GPS position was also manually set to remain static. Despite its lower accuracy, the filter tends to converge more closely to the tag detections—which are more precise in this setup—than to the odometry. This behavior is exactly what was intended.

The fact that the desired precision was not achieved indoors—despite using the highly accurate Vicon system—but was achieved outdoors suggests that the system performs better when relying on real-world sensor fusion rather than ideal lab conditions. Indoors, the system depended entirely on tag detection and Vicon-based ground truth, and while Vicon is extremely precise, the system's own estimation did not meet the precision goal. Outdoors, however, the absence of Vicon meant that the system had to rely on GPS and onboard odometry, with tag detections providing additional corrections. Despite GPS being less accurate than Vicon, the filter effectively prioritized the more reliable tag detections over odometry, leading to better convergence and meeting the precision objective. This implies that the system is well-tuned for real-world conditions, and that its design allows it to exploit tag information more effectively in outdoor environments, possibly due to more favorable lighting or sensor behavior in non-laboratory settings.

Table 5.28: Z-statistic and p-value for x and z-axis of the system.

Coordinate	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 rejected?
x	0.0095	-32.489	0	Yes	Yes	Yes
z	0.0372	-4.294	0	Yes	Yes	Yes

Table 5.29: Z-statistic and p-value for all positions in y-axis.

Position	$\bar{x}$	Z	p	$p < \alpha$?	$Z <$critical Z?	H_0 rejected?
center-down	0.0362	-2.8055	0	Yes	Yes	Yes
center-top	0.0107	-159.9356	0			
mid-left-down	0.0086	-27.2826	0			
mid-left-top	0.0329	-17.3941	0			
mid-right-down	0.0429	-2.3632	0.009			
mid-right-top	0.051	-13.9633	0			
most-left-down	0.0085	-36.6782	0			
most-left-top	0.0132	-111.0811	0			
most-right-down	0.0057	-273.4112	0			
most-right-top	0.0437	-10.7729	0			

5.5 Other metrics

The rest of the metrics could not be tested because of time limitations, so objectives' validation was prioritized. Even so, the proofs of concept developed here could give a general idea of whether the designed concept is good enough with more simple tests.

5.6 Economic analysis

5.6.1 Costs

The economics of this project is discussed in this section. Table 5.30 show the costs of the project. First, the average cost of materials is listed. In reality, the materials were used mainly to do the proofs of concepts, so they should not represent costs in the future. The ANAFI drone is not listed here because it is an intrinsic part of the solution—the solution is for the drone, not that the drone is part of the solution. Part of the proofs of concept were electronic components like the Raspberry Pi, the GPS module and the flight controller, but the most expensive were the Raspberry Pi and the Portable Power Station.

A total of 321.00 € would have been spent only on materials for this solution, but all of these materials were already in posetion by the laboratory. Savings could be made here if, for example, instead of a power source, a large electrical extension was used to connect all devices outside.

Then the estimated salary of the people needed in different stages of development was included. These salaries are based on Germany. The PhD student Pascal Goldschmid helped a few hours a week for example setting up the Vicon room, and very importantly, capacitating the main researcher on the advanced topics needed to develop the project. It is important to take this into account because a PhD student is an employee, so any hours he uses in this project, are actual paid hours. Nonetheless, the biggest expense was the main researcher, whose hourly salary was taken from a minimum engineer's salary in Germany. The total spent was of 16,680.00 €, which represents the largest spent in the project, mainly because most materials and assets were already owned by the client.

Finally, assets were really important because in the laptop all simulations and implementations were done throughout the entirety of the project. Its price is based on Germany and it should be used always for the solution, so the total price is used. The Vicon system is also important for the solution because it allows to measure ground truth data and establish the system's accuracy. However, since the client already owned this license at least 10 years ago, from the total estimated license cost of 45.000 € only the four months of usage were used for calculations, so that amount was divided in the 10 years and 12 months per year, and the four months were extracted from the total price. The assets' total cost of 1,980.00 € represent the second-biggest expense of the project.

Table 5.30: Project's economic analysis

Components				
Name		**Cost in Germany**		
Raspberry Pi 4		130.00 €		
OpenPIlot Revolution		13.00 €		
GPS module		10.00 €		
Raspberry Pi poer source		13.00 €		
Portable Power Station		150.00 €		
Cable		5.00 €		
Total		321.00 €		
Salaries				
Researcher	**Hour salary**	**Hours a week**	**Project weeks**	**Total**
Main researcher	48.00 €	20	16	15,360.00 €
PhD student Pascal Goldschmid	33.00 €	2	16	1,056.00 €
Total		16,416.00 €		
Assets used				
Asset		**Associated monetary value**		
Laptop		480.00 €		
Vicon license		1,500.00 €		
Total		1,980.00 €		
Total				
Components		321.00 €		
Salaries		16,680.00 €		
Assets		1,980.00 €		
Total		18,717.00 €		

5.6.2 Project's benefits

Given the research nature of the project, quantifying the benefits is difficult. The robots used in the research group are not used in any inustrial or commercial activity, but are solely used for research, so there is not an immediate economical benefit from this project.

However, the developed solution lays the ground for more research. First, the most direct consequence of the project is a saving in researchers time when going to the field to do experiments. In an 8-hour day of experiments in the field, the ANAFI drone lasts up to one hour charged. That means that in the experiment, at least 8 charging cycles have to be done. With controlling the drone to come back, replacing the battery and charging it, a researcher could last 10 minutes, so it is at least one hour a day just charging one drone. If three drones are taken to the field, this is 3 hours lost in this task, which according to the hourly salary of a PhD student from Table 5.30 corresponds to 99 € a day.

Even beyond the researchers' time, the actual experiment's time is also affected by the absence of one drone and the time it takes to get to the operator for charging and going back to its designated position. So the experiments can also be more efficient, resulting in more results and thus, more publications.

Furthermore, this project serves as a base for further research, for example, in the research group, FRPG two new calls for master's thesis have been published [5] to work on the project follow-up. This means two more researchers doing research and publications for at least one year.

Finally, the publications allow the group to participate for European, German or international reserach grants to keep doing better research. Also, it keeps the interest of the many sponsors they have and may attract more, resulting in indirect economic benefits.

Chapter 6

Conclusions

A state of the art for drones' autonomous localization and coordination algorithms or processes scheduling was defined in order to give context to the project, and also to generate a set of different concepts for each of them in the design methodology. Some of these ideas were implemented into the final solution.

A program for relative autonomous localization was developed and tested in hardware, where multiple statistical tests were performed to validate the solution. It was demonstrated that there is significant difference between the positions, so nine accuracy measurements were obtained for each axis. In the x-axis the range goes from 30.03 cm 47.52 cm; in the y-axis the accuracy varies from 0.64 cm to 36.34 cm; the z-axis has a range of 2.97 cm to 12.86 cm. Table 5.4 shows the results for all positions, however, it is noted that overall the desired accuracy was not achieved. To further understand the accuracy of the measurements, a power analysis was done, in which a power of 1 was obtained for every position and axis, concluding that the amount of measurements is good, but the solution will not achieve the desired accuracy.

When testing the solution outside, without the laboratory's precise Vicon system, an accuracy of less than 10 cm was achieved. In this case, there was not enough evidence to infer that the x and z-axis accuracies vary with position, so the accuracy was studied with all positions combined. This was validated through a hypothesis test, just as before. The accuracy for the x-axis is of 0.95 cm, the z-axis' was of 3.72 cm, and the accuracy on the y-axis varies from 0.57 cm to 5.1 cm.

Finally, an EDF algorithm for scheduling was implemented. It was not validated in hardware due to hardware limitations, but it was qualitatively tested in software. Several scenarios were explored through plotting, but for the last scenario, the most complicated one, the scores were obtained and analyzed at different times. Priority scores of up to 0.628 were obtained where the drone was the farthest (10 m from the box) and with the lowest battery (17.6%). This way, a priority was established according to the scores.

6.1 Recommendations

As the hardware validation in the Vicon laboratory was not successful, it is recommended to explore other influence factors that were already defined, to see if it can be fixed. In case it is not possible to obtain better, acceptable measurements, another concept from the alternatives could also be explored.

Regarding the scheduling algorithm, hardware tests are highly recommended to actually test the real-world behavior of the algorithm. Also, safeguards for when the drones are actually going to be discharged were not implemented, so this should be part of a following iteration.

Finally, more tests outside with the calibrated GPS information is suggested to have a better idea of the solution's validity.

Bibliography

[1] M. Krogius, A. Haggenmiller, and E. Olson, "Flexible layouts for fiducial tags," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2019.

[2] MATLAB, "Why use kalman filters? — understanding kalman filters, part 1," YouTube, Jan. 2017. [En línea]. Disponible: https://youtu.be/mwn8xhgNpFY?si=-5FJpcBnIAsqvcNR

[3] MATLAB, "Optimal state estimator — understanding kalman filters, part 3," YouTube, 2017. [En línea]. Disponible: https://youtu.be/ul3u2yLPwU0?si=6IlgDcyMt8Oyy1up

[4] Parrot, "Anafi technical specifications," 2025. [En línea]. Disponible: https://www.parrot.com/assets/s3fs-public/2020-07/white-paper_anafi-v1.4-en.pdf

[5] A. Ahmad, "Flight robotics and perception group," 2024. [En línea]. Disponible: https://www.aamirahmad.de/

[6] A. Ahmad, "Robot perception," 2024. [En línea]. Disponible: https://www.aamirahmad.de/research/robot-perception/

[7] Flight Robotics and Perception Group, "Wildcap autonomous non-invasive monitoring of animal behavior and motion," 2024. [En línea]. Disponible: https://www.aamirahmad.de/projects/wildcap/

[8] ISO 8373:2021, "Robotics—vocabulary," International Organization for Standardization, Standard, 2021.

[9] "IEEE standard ontologies for robotics and automation," *IEEE Std 1872-2015*, DOI 10.1109/IEEESTD.2015.7084073, pp. 1–60, 2015.

[10] ASTM E2521-24, "Standard terminology for evaluating response robot capabilities," American Society for Testing and Materials, Standard, 2024.

[11] L. F. D. W. J. Borenstein, H. R. Everett, "Mobile robot positioning: Sensors and techniques," *Journal of Robotic Systems*, vol. 14,

[12] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, DOI 10.1109/IROS.2016.7759617, pp. 4193–4198. IEEE, Oct. 2016.

[13] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407. IEEE, May. 2011.

[14] R. Arnold, K. Carey, B. Abruzzo, and C. Korpela, "What is a robot swarm: A definition for swarming robotics," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, DOI 10.1109/UEMCON47517.2019.8993024, pp. 0074–0081, 2019.

[15] J. H. Sudd and N. R. Franks, *The Behavioural Ecology of Ants*. Springer Dordrecht, 1987.

[16] MATLAB, "Optimal state estimator algorithm — understanding kalman filters, part 4," YouTube, 2017. [En línea]. Disponible: https://youtu.be/VFXf1lIZ3p8?si=-KIFo4UJLANzHhhk

[17] K. T. Ulrich and S. D. Eppinger, *Diseño y desarrollo de productos*. McGraw Hill Education, 2013.

[18] F. Mala and R. Ali, "The big-o of mathematics and computer science," *Journal of Applied Mathematics and Computation*, vol. 6, DOI 10.26855/jamc.2022.03.001, pp. 1–3, 01 2022.

[19] M. Pavliv, F. Schiano, C. Reardon, D. Floreano, and G. Loianno, "Tracking and relative localization of drone swarms with a vision-based headset," *IEEE Robotics and Automation Letters*, vol. 6, DOI 10.1109/LRA.2021.3051565, no. 2, pp. 1455–1462, 2021.

[20] X. Si, G. Xu, M. Ke, H. Zhang, K. Tong, and F. Qi, "Relative localization within a quadcopter unmanned aerial vehicle swarm based on airborne monocular vision," *Drones*, vol. 7, DOI 10.3390/drones7100612, p. 612, 09 2023.

[21] S. Wu, R. Li, Y. Shi, and Q. Liu, "Vision-based target detection and tracking system for a quadcopter," *IEEE Access*, vol. 9, DOI 10.1109/ACCESS.2021.3074413, pp. 62 043–62 054, 2021.

[22] K. Guo, Z. Qiu, W. Meng, T.-M. Nguyen, and L. Xie, "Relative localization for quadcopters using ultra-wideband sensors," 10 2016.

[23] T.-M. Nguyen, A. Hanif Zaini, C. Wang, K. Guo, and L. Xie, "Robust target-relative localization with ultra-wideband ranging and communication," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, DOI 10.1109/ICRA.2018.8460844, pp. 2312–2319, 2018.

[24] S. Pfeiffer, V. Munaro, S. Li, A. Rizzo, and G. C. H. E. de Croon, "Three-dimensional relative localization and synchronized movement with wireless ranging," *Swarm Intelligence*, vol. 17, DOI https://doi.org/10.1007/s11721-022-00221-0, pp. 147–172, 2023.

[25] J. Hausberg, R. Ishikawa, M. Roxas, and T. Oishi, "Relative drone-ground vehicle localization using lidar and fisheye cameras through direct and indirect observations," 2020. [En línea]. Disponible: https://arxiv.org/abs/2011.07008

[26] H. Yang1, Y. Lee1, S.-Y. Jeon1, and D. Lee, "Multi-rotor drone tutorial: systems, mechanics, control and state estimation," *Intelligent Service Robotics*, vol. 10, DOI https://doi.org/10.1007/s11370-017-0224-y, pp. 79–93, 2017.

[27] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, DOI 10.1109/ICRA.2017.7989376, pp. 3299–3304, 2017.

[28] OSDev.org, "Scheduling algorithms," 2023. [En línea]. Disponible: https://wiki.osdev.org/Scheduling_Algorithms#Shortest_Remaining_Time_Next

[29] P. Eggert, "Real time scheduling," 2017. [En línea]. Disponible: https://web.cs.ucla.edu/classes/winter17/cs111/readings/realtime.html

[30] I. Lee, "Real-time scheduling (part 1)," 2010. [En línea]. Disponible: https://www.seas.upenn.edu/~lee/10cis541/lecs/lec-RT-sched-part1-v2-1x2.pdf

[31] Parrot, "Olympe documentation," 2023. [En línea]. Disponible: https://developer.parrot.com/docs/olympe/index.html

[32] A. Sarabakha and P. N. Suganthan, "anafi_ros: from Off-the-Shelf Drones to Research Platforms," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, DOI 10.1109/ICUAS57906.2023.10155881, pp. 1308–1315, 2023.

[33] Parrot, "Parrot sphinx sdk," 2023. [En línea]. Disponible: https://developer.parrot.com/docs/sphinx/sdk_introduction.html

[34] Open Robotics, "Gazebo," 2025. [En línea]. Disponible: https://gazebosim.org/home

[35] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. Cham: Springer International Publishing, 2016. [En línea]. Disponible: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[36] D. Malyuta, "Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy," Master thesis, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, USA, Dec. 2017.

[37] D. Hershberger, D. Gossow, J. Faust, and W. Woodall, "rviz," 2018. [En línea]. Disponible: http://wiki.ros.org/rviz

[38] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, DOI 10.1109/TePRA.2013.6556373, pp. 1–6, Apr. 2013.

[39] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2022.

[40] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, Jul. 2014.

[41] Charles River Analytics, "robot_localization," 2017. [En línea]. Disponible: https://github.com/cra-ros-pkg/robot_localization/blob/kinetic-devel/params/ekf_template.yaml

[42] Vicon Motion Systems, "Vicon," 2025. [En línea]. Disponible: https://www.vicon.com/

[43] J. Shepard, "How to read battery discharge curves," 2021. [En línea]. Disponible: https://www.batterypowertips.com/how-to-read-battery-discharge-curves-faq/

[44] Grepow, "Basis of lipo battery specifications," 2021. [En línea]. Disponible: https://www.grepow.com/blog/basis-of-lipo-battery-specifications.html

[45] LibrePilot, "Openpiolot revolution," 2017. [En línea]. Disponible: https://librepilot.atlassian.net/wiki/spaces/LPDOC/pages/26968084/OpenPilot+Revolution

[46] J. Whitaker, "pyproj documentation," 2024. [En línea]. Disponible: https://pyproj4.github.io/pyproj/stable/

[47] The MathWorks, Inc, "Direction cosine matrix ecef to ned," 2025. [En línea]. Disponible: https://de.mathworks.com/help/aeroblks/directioncosinematrixeceftoned.html

[48] J. B. Schleppe, "Development of a real-time attitude system using a quaternion parameterization and non dedicated gps receivers," Master's thesis, Department of Geomatics Engineering, 1996.

[49] J. Kallwies, B. Forkel, and H.-J. Wuensche, "Determining and improving the localization accuracy of apriltag detection," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, DOI 10.1109/ICRA40945.2020.9197427, pp. 8288–8294, 2020.

[50] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probabilidad y estadística para ingeniería y ciencias*, 9th ed. PEARSON EDUCACIÓN, 2012.

Appendix A

Setup for covariance estimation

Figure A.1: Image of setting top-right for covariance estimation.

Figure A.2: Image of setting bottom-right for covariance estimation.

Figure A.3: Image of setting center-center for covariance estimation.

Figure A.4: Image of setting center-right for covariance estimation.

Appendix B

Setup for hardware validation

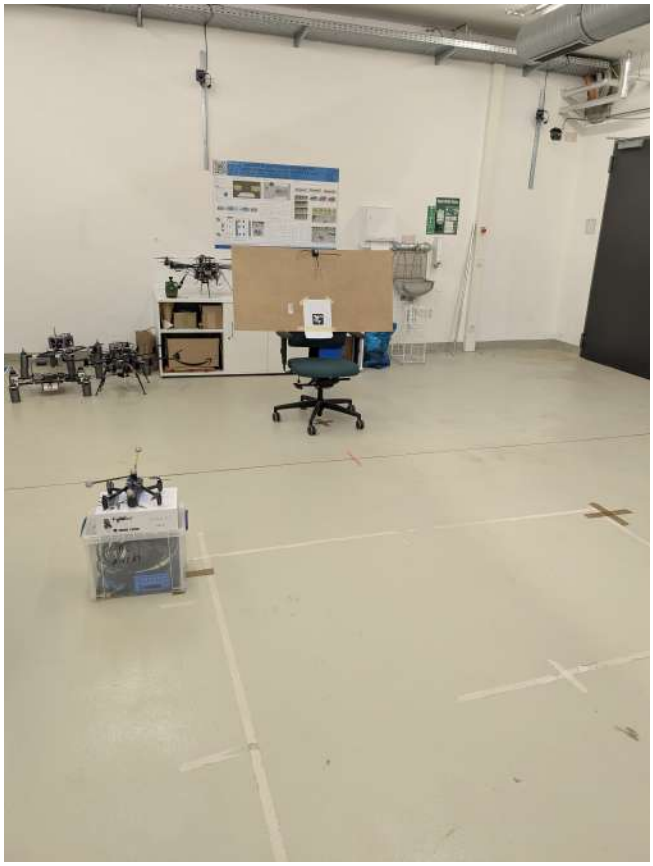

Figure B.1: Image of setting bottom-left for hardware validation.

Figure B.2: Image of setting center-center for hardware validation.

Figure B.3: Image of setting top-right for hardware validation.

Appendix C

GitHub Repository

The main branch from the public repository can be found in the following link: `https://github.com/isaacmn24/apriltags_kalman`.