Dany dos Santos Costa

Sistema de Navegação Assistido por RFID para Cortadores de Relva

RFID Tag Aided Navigation System for Lawnmowers

Universidade de Aveiro
2022

Dany dos Santos Costa

Sistema de Navegação Assistido por RFID para Cortadores de Relva

RFID Tag Aided Navigation System for Lawnmowers

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira, Professor Auxiliar do da Universidade de Aveiro.

Dedico este trabalho aos meus pais que me proporcionaram sempre as melhores condições possíveis durante estes anos de estudo

o júri / the jury

presidente / president

Prof. Doutor João Nuno Pimentel da Silva Matos
Professor Associado, Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Fernando José da Silva Velez
Professor Auxiliar, Departamento de Engenharia Electromecânica da Faculdade de Engenharia da Universidade da Beira Interior

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar, Universidade de Aveiro

agradecimentos / acknowledgements

Quero agradecer a toda ajuda do Engenheiro e team leader Daniel Silva da equipa Bosch Indego, que me orientou e me ajudou sempre que possível. Agradeço também ao resto da equipa Bosch Indego, que com um ambiente fantástico, sempre me acolheram da melhor forma, proporcionando todas as condições possíveis para conseguir trabalhar e ajudando-me nas dificuldades que fui obtendo ao longo deste trabalho.

Desejo também agradecer aos meus pais que, sem as condições que me foram dando, nunca seria capaz de conseguir alcançar estes objetivos, não só na realização desta dissertação como também ao longo de todos estes anos.

Aos meus amigos, agradeço pelo espirito entreajuda que sempre tiveram ao longo destes anos, sem a ajuda deles nada seria possível.

Por fim, quero agradecer á minha namorada que sempre me apoiou e ajudou a resistir aos problemas que foram ocorrendo ao longo deste trabalho e do curso.

Palavras Chave

Corta-relvas autónomo, Localização, Mapeamento, Tags ,RFID , Navegação, Emulador , Odometria , Recalibração

Resumo

Bosch Indego é um corta-relvas autónomo que utiliza um fio de perímetro que demarca a zona de corte e serve de mecanismo base de localização/navegação. Tirando esta referência absoluta, o robot apenas conta com informação de sensores inerciais e integração da deslocação da parte motora (odometeria). Isto faz com que, em jardins de grandes dimensões possa ter desvios consideráveis, podendo falhar o corte em algumas zonas.

O principal foco desta dissertação é demonstrar que é possível optimizar o processo de navegação/localização com a integração de tags RFID colocadas no relvado, que servirão como novos pontos de referência, de forma a corrigir o desvio de localização do robot sem que este tenha de voltar para a única fonte de localização absoluta que é a doca, que poderá estar muito distante da zona de corte.

Primeiro foi feito um estudo sobre qual o tipo de sistema RFID por forma a saber qual a tecnologia mais apropriada para esta aplicação. Após isto, foi feita uma implementação capaz de demonstrar através de um emulador feito para o efeito a capacidade de georreferenciar tags RFID validando o erro entre a tag e a posição do robot.

Posteriormente, foram feitos vários testes em dois relvados sintéticos, onde as tags foram colocadas no relvado quer de forma aleatória, quer organizada em cima do fio de perímetro.

No fim demonstrou-se que é possível implementar um sistema RFID com a possibilidade de ajudar o robot através da adição de mais pontos de referência no mapa. Contudo, os testes constataram que a melhor solução baseia-se na implementação de tags exclusivamente no fio de perímetro, uma vez que o movimento do robot é muito incerto fazendo com que nem sempre seja possível detetar corretamente as tags. Isto não acontece quando este está a seguir o fio de perímetro. Ainda assim, foi possível exemplificar a grande versatilidade que um sistema RFID pode oferecer, nomeadamente podendo identificar a doca sem que seja necessária intervenção do utilizador. O trabalho futuro deste sistema deve-se de focar em melhorar a precisão da deteção das tags tal como o posicionamento das mesmas relativamente ao movimento do robot

Keywords

Autonomous Lawnmower, Localization, Mapping, Tags, RFID, Navigation, Emulator, Odometry, Recalibration

Abstract

Bosch Indego is an autonomous lawnmower that uses a perimeter wire to define the mowing area as well as a basic location/navigation mechanism. Apart from this absolute reference, the robot's only sources of data are their inertial sensors and the integration of the motor's movement (odometry). This implies that in larger gardens, there may be significant deviations, resulting in miscuts in certain regions.

The main objective of this dissertation is to demonstrate that by integrating RFID tags placed on the grass as new reference points, it is possible to optimize the navigation/location process and reduce the robot's localization deviation without having to return to the only source of absolute location (dock), which may be too far away from the cutting zone.

Firstly, a research was conducted to determine the sort of RFID system that would be most suited for this application. Following that, it was implemented a system capable of proving the capacity to georeference RFID tags validating the relative error between the tag and the robot's location with an emulator designed for this purpose.

Subsequently, several tests were carried out on two synthetic fields, where tags were placed on the grass either randomly or in the perimeter wire.

In the end, it was demonstrated that an RFID system can be implemented with the potential of aiding the mower by providing more reference points to the area. However, tests showed that the best solution is based on the implementation of tags exclusively on the perimeter wire, since the robot's movement is very unpredictable, making it not always possible to correctly detect the tags. This does not happen when the mower is following the perimeter wire. Even so, it was possible to exemplify the great versatility that an RFID system can offer, namely being able to identify the dock without user intervention. Future work on this system should focus on improving tag detection accuracy as well as tag positioning relative to robot movement.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
AIDC	Automatic Identification and Data Capture
ALM	Autonomous Lawn Mower
AoA	Angle of Arrival
CM	Connect Module
cm	centimeter
CRC	Cyclic redundancy check
CSS	Cascading Style Sheets
DLE	Data Link Escape character (ascii)
EFLS	Exact Fusion Location System
ETX	End of transmission character
EPC	Electronic Product Code
FIFO	First In First Out
GNSS	Global Navigation Satellite System
GPS	Global Position System
HF	High Frequency
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
JS	JavaScript
JSON	Java Script Object Notification
LF	Low Frequency
MCU	Micro-Controller Unit
mm	millimeter
RFID	Radio Frequency Identification
RO	Read Only
rpm/s	rotations per minute
RSSI	Received Signal Strength Indication
RTLS	Real Time Location System
RTK	Real Time Kinematic
RW	Read and Write
OS	Operating System
SLAM	Simultaneous Localizing and Mapping
STX	Start of transmission character
TCP	Transmission Control Protocol
ToF	Time of Flight
TDoA	Time-Distance-of-Arrival
UHF	Ultra High Frequency
UI	User Interface
TID	Tag identifier
UART	Universal Asynchronous Receiver Transmitter
UWB	Ultra wide-band
Vdd	Voltage drain drain
WWW	World Wide Web
WORM	Write Once, Read Many
3D	three dimensional
2D	two dimensional

CHAPTER 1

Introduction

1.1 Framework

Humans have always needed basic tools to aid them in their daily lives since prehistory, whether it is for eating, transporting, or even communicating. Most basic machines are designed to reduce the amount of effort (force) required to perform a simple task. With the passage of time, these little tools evolve into sophisticated machines. Nowadays, machines are capable of performing complex tasks more successfully, efficiently, and faster than people. Machines can work with very little human assistance, but only to a certain extent [1]. With the advancement of technology, the concept of "robot" began to emerge. A machine is basically a simple device that may be programmed to carry out a command. A robot, on the other hand, is much more than a simple machine. A robot can be considered a single combination of multiple machines. Robots can be reprogrammed to perform multiple roles in situations where machines cannot [2]. They were initially designed to process resources and manufacture goods, particularly during the Industrial Revolution. This left an indelible imprint on history, since people's living standards increased dramatically. Autonomous robots initially arose as a result of this development, when robots and machines became more useful, sophisticated, and intelligent. Autonomous lawn mowers are a great illustration of the benefits that autonomous robots may provide in our daily lives [3].

An autonomous robot is a robot that can perform its tasks without the need for external help, making its decisions and actions with its own intelligence. They are essentially desirable in fields such as household maintenance, spaceflights, water treatment, and delivering services. Self-maintenance, environment sensing, task performance, and autonomous navigation are all concepts that must be mentioned while discussing autonomous robots. The first is that when the robot has an internal problem, he must take care of himself. Sensing the environment is essential for the robot to avoid issues, which is why it needs a variety of environmental sensors to complete its responsibilities. Secondly, task performance refers to the capacity to complete a job in the most feasible manner. Last but not least, autonomous navigation is the aptitude of being capable of navigating in unknown environments, and it's normally divided

into two categories: outdoor and indoor. Normally, outdoor navigation is a more difficult and complex task because usually, the environment is unstructured and unpredictable which can hamper the tracking task, making the robot's localization harder to define or predict [4].

1.2 Motivation

For a robot to associate actions with a place it's necessary to know where it is and how to navigate from point A to point B. Normally, the information available for computing the robot localization is collected using inertial sensors, to analyze the environment and its own motion. Through inertial sensors, the robot can compute its location relative to where it started. However, those sensor measurements are not perfect and can have some issues due to noisy observations giving relative errors, which makes the location obtained unreliable as the robot navigates in its environment. These errors are normally reduced with alternative scenarios, mainly retailed on external sensors placed in the environment, which will help the robot to estimate its localization.

Navigation has always been a very complex task to solve in the robotics world. In this dissertation, the application of RFID technology is studied to be an auxiliary tracking system to help an autonomous lawnmower from Bosch company in its navigation system through the lawn.

The context of this dissertation comes up with the interest of understanding how an autonomous robot performs tasks. The objective is to acknowledge the various techniques that a robot uses to navigate in a given environment and how external sensors can aid him. Moving through an unknown environment while maintaining precise localization is always a difficult task, which is where the study of RFID as an auxiliary navigation system comes in. The RFID technology will be utilized to attach the tags to strategic places that will act as reference points in order to have more absolute points of reference than simply the dock.

1.3 Objectives

The autonomous grass cutting systems from Bosch, as well as other solutions on the market, use a perimeter wire that demarcates the cutting area and serves as a basic location/navigation mechanism. Aside from this absolute reference, the robot only has information from inertial sensors and integration of the motor part's displacement. This means that this robot in large gardens can have considerable deviations and may miss the cut in some areas. In order to improve the robustness of this system, it's intended to carry out the study of integrating RFID tags in the piles that hold the wire on the lawn or other additional ones, in the middle of the garden or on the loading dock, to correct the deviation of the robot's localization. without having to go back to the only source of absolute location, which is the dock, which may be very far from the cutting zone. The goal of this dissertation is to integrate a RFID system in Bosch Indego's autonomous grass in order to improve his localization/navigation system using RFID tags in some strategic points. In response to this challenge, the development process has been separated into different phases, presented below:

- Study of the basic concepts in localization/navigation systems
- Specification of the localization system based on RFID
- Development of the simulation to prove the initial concept
- Familiarization with the necessary tools to integrate in the robot
- Architecture design and implementation of the system in the robot
- Integration and test of all the components in the system

1.4 Document outline

This dissertation is organized as the following:

- the **background status** of navigation and location, strategies in autonomous robots are presented in chapter 2, with an emphasis on autonomous lawn mowers. The Bosch Indego lawn mower will be introduced, followed by a discussion of the main elements of robotic mower navigation, as well as other techniques employed by other robotic mowers on the market.
- chapter 3 presents a more **theoretical framework** on **RFID** and its key elements, a brief explanation of how RFID system will help Bosch Indego's lawnmower in his autonomous navigation system as well as the description of the approached scenarios.
- chapter 4 relies on the description of the implemented **architecture**.
- chapter 5 debates the methods used to **implement** this system, with a detailed explanation and demonstration of all the tools used and how the work was divided.
- chapter 6 discusses the **obtained results** along with the respective interpretation and explanation.
- the last chapter presents the **conclusions**, main contributions of the work, some limitations, and some guidelines on the future work.

CHAPTER 2

Background

2.1 Introduction

The process of determining one's position in a given environment is known as localization. Higher-developed species have the ability to locate themselves in their surroundings. This ability appears to be " hardwired " into their brains, and it is mostly automatic or subconscious in humans. We take knowing where we are for granted, so we do not think about the huge amount of environmental data that must be processed, kept, compared, and then used in real-time to identify an individual's genuine location.

This chapter will start by portraying a theoretical review about localization and navigation in an autonomous robot, explaining the most important concepts, tasks required, steps, and difficulties that a robot has to face when it is trying to locate itself. Outlined that, a brief introduction about autonomous lawnmowers will be made and then will be exploited the techniques used, based on what was explained. Following the explanation made, will be presented several examples from the market and their navigation techniques, with a particular focus on Indego's lawnmower. Finally, a review of the several external sensors will be conducted, comparing each of them in the scenarios in which they will be used and determining what will be the best options.

2.2 Robot localization

Robot localization is the procedure of determining where a robot is located in relation to its environment. To comprehend where a robot is in a particular environment, it must be able to determine its own location in order to make judgments about its future actions. That is the reason why localization is one of the most important skills that an autonomous robot must possess. Normally to have knowledge about the environment, in a usual robot localization scenario, the robot needs to have a map of the area where it will navigate [5]. This can be achieved in countless different ways depending on the environment that can be dynamic or static. The ideal environment is where everything is static, there are no changes in time, and

a robot can more easily follow a trajectory to its destination. However, a static environment is rare to find. Usually, a robot that can only perform in static areas is a robot with fewer skills to work in more complex and dynamic terrains. A dynamic map implies objects moving and constant changes in time. Dynamic environments are the most common and the most complex to reach, this happens because the robot has to learn the map by himself as it moves along the area. There are a lot of different methods and techniques used to learn the environment and build a suitable map, however, they all rely on inertial or external sensors. Problems relative to dynamic environments can be divided not only in the mapping task but also with tasks like **collision avoidance** because it must sense unknown obstacles and avoid them; **sensor fusion** where the robot needs to control the different measure's techniques coming from their own sensors and **trajectory tracking** due to the **uncertainty** of the methods used to calculate his location. In the next subsections, those problems will be explored in more detail [4].

2.2.1 Obstacle detection

An autonomous robot navigating in a cluttered area has to be capable of avoiding a variety of obstacles, that can be either static or dynamic. Static obstacles are the ones that never change in time (for example a tree) and can be easily avoided with simple mechanisms. On the other hand, with dynamic obstacles, a special and more complex procedure has to be developed. In this situation, active collision avoidance, as well as precise control and manipulation of the surroundings, must be achieved to avoid those obstacles.

2.2.2 Sensors fusion

Sensor fusion is a crucial criterion in location approaches because it is this operation that collects data from several sensors. Many measurements are taken as the robot navigates, which is why algorithms must be created to collect that data and find the optimal solution to the problem. These algorithms or methods can be classified in different ways, including probabilistic methods, least-squares algorithms, and intelligent fusion techniques - Artificial Intelligence (AI).

2.2.3 Trajectory tracking

Trajectory tracking is the aptitude of tracking an organism while it is in movement, in real-time, and can be divided into three parts: **navigation**, **guidance**, and **control system**. In navigation, two methods are usually applied, absolute and relative positioning. Absolute position is the precise location where the organism is its fixed reference. Relative position is obtained in reference to other known locations (usually fixed points). In robotics, the distinction between these two ideas is made by the type of sensor employed, which is either inertial or external. External sensors, provide data that measures the correlation between the robot and external reference points. Examples of external sensors are sonar, radars, beacons, transmitters, and so on. Internal sensors will provide information that is measured on the robot. Examples of internal sensors are accelerometers, encoders, compasses, gyroscopes,

among others [4]. Internal sensors can be used to identify location using two simple methods: **odometry** and **dead reckoning**.

2.2.3.1 Odometry

A robot's location can be described by a pair of coordinates (x,y) which can either be the robot's relative location to a start point or its absolute location on the map. In each scenario, it must be known the **orientation** the robot is facing. Orientation θ, like coordinates, can also be relative or absolute, and it has to be added to the (x,y) coordinate vector to determine all aspects of a robot's location. This vector (x,y,θ) is called **pose** of the robot. The robot's pose is enough to describe all the necessary data on the robot's location. Usually, all location algorithms return a pose as the result. In odometry, the robot's wheels are used to generate this vector. This is accomplished by specifying the location of the wheels in relation to the robot's center and then connecting sensors that detect each wheel rotation (encoders) to determine how far the robot has traveled in comparison to the starting point.

$$P = [x, y, \theta] \tag{2.1}$$

" Odometry is used by some robots, whether they be legged or wheeled, to estimate (not determine) their position relative to a starting location. This method is sensitive to errors due to the integration of velocity measurements over time to give position estimates. Rapid and accurate data collection, equipment calibration, and processing are required in most cases for odometry to be used effectively" [6].

This method can be described on a simple principle: "If you know where you were, and you moved, then you know where you are". This means that, if the starting point is known and how much, and in what direction, it moved, then it is easy to calculate their current position. [7]. Odometry can be compared with a man counting kilometers based on his footsteps when he goes jogging, for example. With this, he can figure out how far he traveled by measuring and counting the footsteps.

Although it is a very good and common method, it is important to take into account that this method has its own limitations due to measurement errors coming from sensors. Using the previous example, the men's measurements will be inaccurate, counting 1.2 kilometers when he actually walked 1.4 kilometers. This occurs as a result of **uncertainty measures**. The footsteps are not always the same size even if the best effort is made, and this happens not just because it is impossible to control a perfect and concise rhythm but also because of external behaviors (uncertain ground, slippy ground, unpredictable objects, among others). In robotics, this is normally called **noise**. The accuracy and efficiency of this method will always be dependent on the precision of the sensors and, most of all, the environment. For example, if a wheel slips or passes through an inconstant ground, the measures obtained do not correspond to what happens in reality. Therefore, in a more complex dynamic environment, odometry is not suitable to handle the localization problem alone [7].

2.2.3.2 Dead reckoning

This approach, like odometry, employs wheel sensors as well as heading sensors to update the robot's location. It is feasible to compare the data with a description of the actual environment, gathering data from additional sensors. This analogy enables the determination of the position relative to the environment stored. Basically, with other auxiliary sensors capable of determining robot velocity and orientation (accelerometers, gyroscopes, compasses) it is possible to determine the robot position in a more accurate way.

As previously mentioned, if odometry is based on our footsteps, dead reckoning is based on that plus other physical measures (for e.g. velocity). If the same man controls his speed and counts his footsteps constantly, then he can calculate with more accuracy how much he walked. However, he will not have a final of 1.4 kilometers due to the same reason presented before. It is difficult for anyone to walk or run at a constant speed, and with other external adversities (slippery streets), a variable number of unexpected errors occurs, which affects the final measurement

Errors can be divided in **deterministic** and **non-deterministic** in both techniques. Deterministic errors are directly connected to the input and can be eliminated through re-calibration techniques (for example) while non-deterministic unpredictable errors (external errors) and will always lead to uncertain location estimates [8].

To summarize, these two approaches have undergone significant evolution and are still the finest and most often used techniques to implement in an autonomous robot. However, sometimes they are insufficient to complete the mission due to the buildup of uncertainty and the unpredictability of the surroundings in which the robot will traverse. In order to get more accurate location results, those techniques could be combined with external sources.

2.2.4 Mapping and localization

All the information about the area where a robot will perform its actions is provided in a form of a map (2D or 3D). When the map is unavailable, the robot must have mechanisms capable of building a map while it is navigating in that unknown environment. This technique is called Simultaneous Localizing and Mapping (SLAM). SLAM is much more like a definition than a specific algorithm. SLAM can be compared to the chicken-or-egg theory because the robot needs a map to localize himself, but it also needs an estimation of his pose to estimate his location and build the map. Having a robust SLAM technique is very complex to achieve but fortunately, there are already a lot of methods implemented that show numerous algorithms, allowing the robot to have a good SLAM implementation. Most of all, SLAM are divided into various steps: data association, state estimation, state update, and landmark update. This dissertation will focus only on the landmark update phase. Landmarks are distinguishable features that can be easily re-observed and recognized from the surrounding. The robot uses these to figure out where to localize itself. For the robot, this operates similarly to blindfolding a person. To prevent being disoriented while going around his residence, this person can reach out and touch items or hug walls. Certain qualities, such as the sense of

touching a door frame, might help this individual figure out where he is. A robot's sense of touch is provided via his inertial sensors [9].

Two types of mapping techniques can be distinguished: metric and topological mapping. Topological approaches are focused on connecting points, whereas metric maps acquire geometrical aspects and properties of the environment.

Figure 2.1: Topological map example [10].

Figure 2.2: Grid map example [10].

When using various ways to accomplish or enhance a robot's localization, it is necessary to develop a plan that includes not only the budget but also the robot's objective (for example, if it is a commercial robot, who will use it). Then another outline about the types of environments where the robot will navigate, as to be made in order to have the perception about external sources that can influence and damage the measurements. Besides that, it is equally important to determine what kind of inertial sensors the robot already possesses and then implement new algorithms based on them. Otherwise, if external sensors are the solution chosen and a priory study of the perfect technology to use is to be made. The purpose of this dissertation is exactly this: to analyze how RFID technology (external sensor), can prove that it is possible to implement a system to help an autonomous lawnmower localization system: the Bosch Indego lawnmower. .

2.3 Autonomous lawnmowers

A traditional lawnmower is a machine that cuts a lawn surface to an even height with one or more spinning blades. The height of the cut grass is sometimes fixed by the mower's

design, but it is usually adjustable by the operator, usually via a single master lever or a lever or nut and bolt on each of the machine's wheels. The blades may be operated by human force, with wheels mechanically attached to the cutting blades so that the blades spin when the mower is pushed forward, or featured with a battery-powered or plug-in electric engine. Smaller mowers are often without propulsion, relying on human muscle to move across a surface; "walk-behind" mowers, on the other hand, are self-propelled and require just a human to walk behind and direct them. Larger lawnmowers are either self-propelled "walk-behind" or "ride-on" mowers, which allow the operator to control the mower while riding on it.

ALMs is a robot capable of performing all the physical duties of a mechanical lawnmower with little (or no) human effort. The only user intervention that is necessary to make is the setup preparation. Setup preparation is normally an easy task to do, all that is required is to follow the manufacturer's simple instructions. These setups are used to limit the area where the robot will perform its actions, and they all depend on the methodologies followed by the robot's creators. With the setup installed, the user can also control other characteristics of the robot related to the type of the cut and the sensitivity of the robot (obstacle sensitivity, velocity, grass cut, etc). ALM was created to help humans to have clean and good grass without any physical effort, and its creation was a big evolution and meaningful invention in the domestic lawn areas.

There are currently many alternative solutions on the market with various methodologies that can execute the task well from a general standpoint. However, the various principles used in each of them, such as collision avoidance, navigation principles, safety features, mowing system, battery life, and so on, have a number of advantages and disadvantages. This chapter will cover everything related to navigation and location on an ALM, as well as the challenges that may arise.

2.3.1 Bosh Indego

Indego is the ALM solution from Bosch. Coming in two models, Indego M and Indego S. Bosch was the first in the market implementing a different mowing system, the **Logic Cut** 2.3.3. Bosch Indego M and S comes with different models differing in some features but most of all the range area of the mower, efficiency and battery life. Alongside this, Bosch Indego is able to:

- cut areas up to 700 m^2 (depending on the model).
- have multi area that enables the mower to trim in up to three different gardens.
- mow targeted small areas up to 3 m^2, ideal for cutting sections of the lawn that the mower could not achieve.
- cut near the edge with their border cut technique.
- have PIN protection.
- automatic charging.
- cutting on slopes.
- quick blade stop.
- have obstacle detection (collision avoidance).
- no collection of grass clippings.

- automatically schedule when to cut and ensure cutting periods are not too long or too short.
- control the mower easily with their free app, being able to communicate to the mower at a touch of a button.

" *The intelligent, efficient Indego robotic lawnmower - for beautiful lawns made easy. Thanks to Bosch LogiCut, Indego knows where it has been and where it has to go, cleverly avoiding constant running that could damage your lawn, whilst saving energy* " [11].

The next subsections will describe the essential aspects that are directly linked to the navigation and localization procedures in a robotic mower, with examples from the market and Bosch's solution to the problem.

2.3.2 Boundary area

To prepare a robot before it mows the grass, it must be followed every installation step and prepare the setup to create every condition to the ALM. The setup installation relies mostly on the limitation of the area and docking station. In order to cut the grass, the lawnmower must know where it will work, and to do that an area must be defined. Depending on brand and manufacture, there are many ways to accomplish that. The most common technique is based on a **physical wire** that is directly connected to the dock station and transmits a signal across the wire that will be recognized by the robot sensors. This perimeter wire will guide and alert him that it cannot go any further.

Implementing the perimeter wire around the lawn area where the mower will be trimming will take some time and effort. For this reason, manufactures are constantly trying to improve and take out of the list methods that require human intervention, and that is why they are exploring solutions that do not need a physical wire to limit the area. However, the ideal option is difficult to determine because there is no such thing as a flawless or perfect approach; it is always dependent on the manufacturing purpose. Even so, it is possible to mention some alternatives based on the market's offer, which is normally divided in four categories: beacons, GPS, vision-based technology and humidity sensors technology [12]. These solutions are used not only to guide the robot in terms of area limitation, but also to guide the robot when it is navigating through the lawn. Section 2.3.4 will explore more about these types of technologies, how they really work, and how they can help ALM in his localization and navigation tasks, providing examples in the market.

2.3.2.1 Mapping strategy

The mapping procedure can begin once the edges have been determined. Following the edges is the primary premise of mapping a yard. Currently, there are two techniques to doing this work. The first one relies on the user doing nothing, ALM just follows the wire through the grass making a calibration drive. The second concept is based on operating the robot via an app on the user's smartphone, similar to how a remote-controlled car is controlled. Bosch was the first company to create an ALM in the market capable of making a map of the

environment and using it to navigate and make a different type of mow. Another example is Segway ALM (navimow) that uses the control technique to map the lawn. Figure 2.3 demonstrates those approaches used by Bosch Indego (1) and Segway-navimow ALM.

Figure 2.3: Bosch Indego [11] vs Navimow [13] mapping approaches.

Besides all that, mapping the area is not a mandatory task that manufacturers have to deal with. This happens because the robot only has to create a map if it wants to know where it is in the area and how this will affect the algorithm. The need for a map is directly dependent on the type of mowing, which can be either systematic or random. Section 2.3.3 will explain these two principles but, basically, if the robot does not have to follow a designed pattern and if it can travel to the same area more than one time it means that it does not need to know its location on the map. The only thing that it has to know is where the edges are and if it faces an obstacle or not, but, this is accomplished with the boundary limitation and collision avoidance mechanisms, respectively.

2.3.3 Random and systematic mowing

ALMs can cut the grass in two ways: **random** or **systematic**. Random principle, as the name refers, is the procedure of cutting the grass without any pattern, the robot just travels through the yard in a random path, like most autonomous vacuum cleaners. The idea behind this approach is to go further until it finds the edge of an obstacle, and then it changes direction until it has hit every spot. However, the term "random" is not actually correct in many cases. Behind this "random" mowing method, there is also an algorithm more or less complex that manages the robotic behavior. The complexity and sophistication of the algorithm will always depend on the manufacturer and the model. The fact that the movement is not completely random is due to the reorientation of the obstacle and boundary [14]. Basically, when a robot hits an obstacle (for example) it can follow a different direction instead of keeping trying to pass the obstacle until it is avoided. When a typical autonomous vacuum cleaner hits an obstruction or a wall, it is usual to see that they do not instantly reverse direction, but instead continue passing through the impediment while adjusting their course slightly until they can really avoid it. Though, ALM works on a similar principle, and that is why most of them use a different technique using more sophisticated algorithms that require a more precise location.

Oppositely, systematic principle relies on mowing with predefined patterns. Different patterns are also a concept of debate in this method. According to the manufacturers, the systematic mowing approach can be 30% faster compared to the random mowing principle. Taking advantage of their mapping system, Bosch chose to build a system capable of mowing the yard in parallel lines, saving time and energy. They revolutionized the industry by being the first to employ a systematic, owning a patent on their logic mowing, the **LogiCut**. LogiCut intelligent navigation analyzes the most efficient mowing route and alerts Indego to any remaining mowing areas. By shifting the mowing direction for each cutting session, it leaves no traces, merely healthy, lush grass.

"Bosch Indego LogiCut enables Indego to cut efficiently in neat, parallel lines, freeing up your lawn sooner whilst keeping it looking healthy and beautiful" [11].

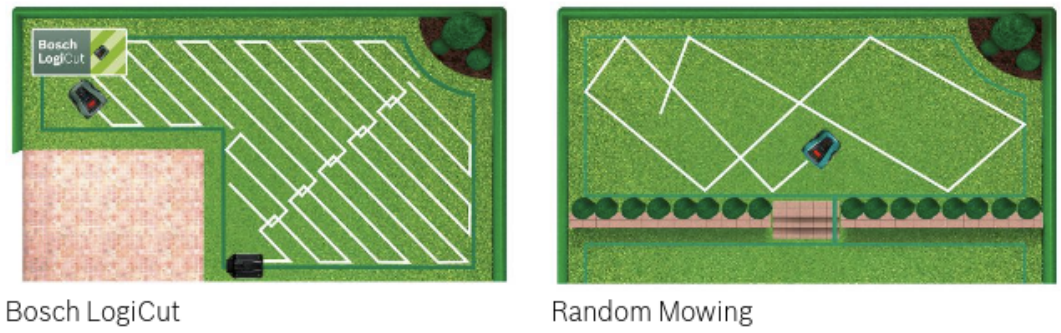

Figure 2.4: Bosch Logic Cut intelligent navigation [11].

However, there is still some debate on whether it is better to mow in a systematic or random manner. There is no right approach; as previously said, it will always rely on the manufacturers' principle. Random mowing is the most preferred method, since the advantages tend to exceed the disadvantages. Advocates of the random method argue that spending longer to mow the grass is not a disadvantage, but rather an advantage because yard maintenance is never complete. For them, ALM is never done with the work. Basically, constantly mowing the lawn is a means to keep it always in good shape, so the fact that the robot takes longer to cut is not bad at all. On the opposite side, systematic defenders, argue that the yard must be mowed in the best possible and faster way. Knowing more precisely where the robot is already mowed or not, allows the robot to save battery and be more efficient. Having a random or systematic system has its pros and cons, but one thing they can highly differ is the need for having a more or less accurate localization system. With systematic mowing, the robot follows a path to cut the grass in a predefined pattern and to do that, they should focus on obtaining a more precise position in order to be aware of the path it has taken before and the direction it must take next. Random principle, although not entirely random, does not need to be worried about this because where the robot was or will be is not actually a concern. However, as previously said, random systems vary from the mower to mower, so those who wish to enhance their random "pattern" in order to prevent mowing the same spot over and

over must consider this, even if the requirement is not the same. The majority of them employ odometry and dead reckoning to perform this task, while some rely also on external sensors. These location techniques will be discussed in further detail in the following subsections.

2.3.4 Localization strategies

2.3.4.1 Beacons

A beacon is a wireless device that is constantly transmitting a broadcast signal. These signals can be received and read by other wireless-enabled devices that follow the same protocol. Beacons send out a unique identifier that the receiver must decipher in order to figure out what is behind it. The main purpose of a beacon is to awaken the devices that are waiting. To do this, they have a protocol implemented that recognizes the signal they broadcast before. This technology is very common in robotic vacuum cleaners. The iRobot Terra mowing uses a series of wireless beacons that can be placed around the lawn, replacing the usual method of using a perimeter wire. After the installation, the user just has to drive iRobot around the yard so that it can acknowledge where the edges of the grass are. Although, on the opposite side, it requires a licensed official working as the FCC(Federal Communications Commission) prohibited the use of low fixed power radio transmitters without a license. iRobot, like Bosch Indego, also has a systematic cutting algorithm, cutting the lawn in stripes. This ALM is not in the market yet, but it will be released soon [15].

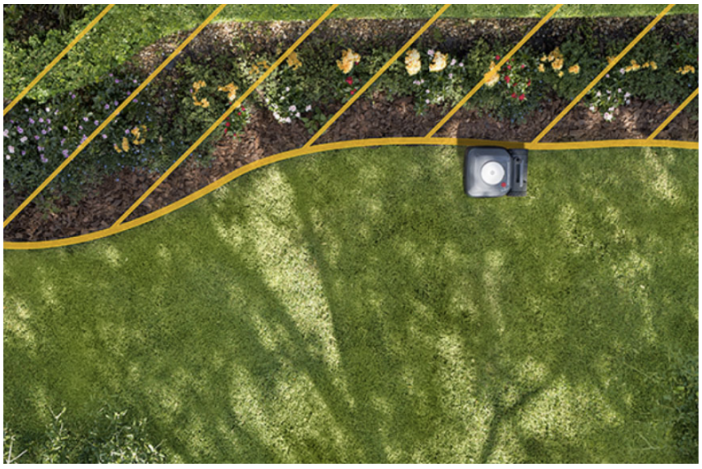

Figure 2.5: iRobot - beacon technology [15].

2.3.4.2 Vision-based technology

The purpose of this technology is to use vision-based sensors to watch the environment and analyze what is surrounding the robot. Vision technology is the same as "adding eyes" to the robot, however, this method is not easy to implement because it requires a lot of artificial intelligence that can analyze and detect all kinds of surrounding objects. To achieve that, it requires a lot of AI and complexity density. This is one of the primary technology used by self-driving cars and works extremely well. Using this technology, it is feasible to detect whether parts are grass and whether the grass has to be mowed. Not only will they be safer

and more accurate, but they will also be more efficient. Aside from that, it is still a rather costly technology, not only in terms of hardware but also in terms of the algorithm layer. Vision-based technology has grown in popularity in recent years, and in the autonomous robotics world, it is a very practical way to achieve reliable navigation. As a result, this technology is likely to be one of the best options used in ALMs to replace not only the perimeter wire, but also to make navigation more clean and efficient.

Toadi (visual robot) is one ALM that has been developed to work with this technology. " The robots use cameras and advanced AI to navigate the yard for a highly efficient cut " [12]. Toadi manufacturers also have made the possibility to control the robot in a smartphone with good features like the 4K camera that can be accessed in their application. Another great feature is that it goes beyond the purpose of a simple ALM being capable of detecting grass intruders with their night vision feature, acting as a guard. In figure 2.6 is shown an example of how the Toadi algorithm works in collision avoidance. At first, it detects the ball and then makes an "imaginary" path that surrounds the ball, giving him the possibility to change his path avoiding the ball. However, this mower is not available yet (only pre-order).

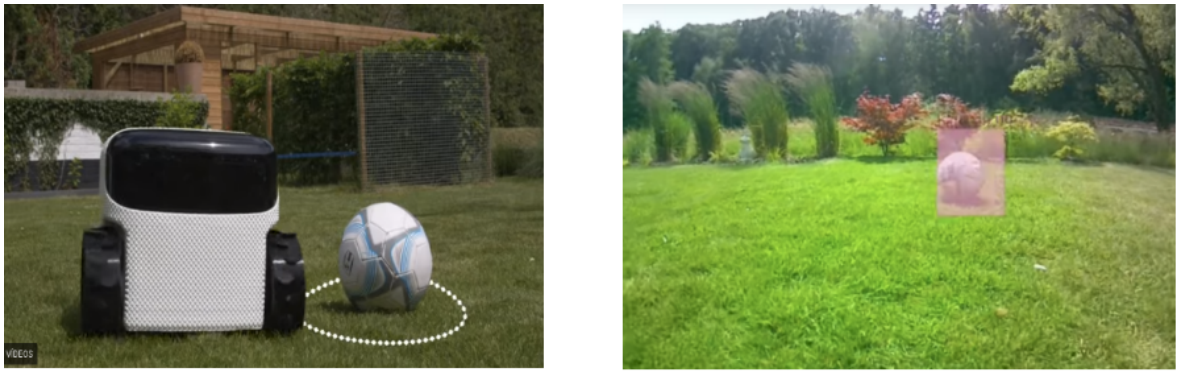

Figure 2.6: Toadi lawnmower - vision technology [16].

2.3.4.3 GPS

GPS is a location system based on satellites on a global network of satellites that transmit radio signals from medium earth to orbit. GPS is divided into three parts: satellites, receivers, and ground stations. Each satellite transmits a unique signal and orbital parameters that allow GPS devices to compute and decode the precise location of the satellite. Ground stations use radars to be sure satellites are actually where they think they are. These satellites send out a signal, which is constantly monitored by a receiver. The GPS receiver, above all, measures the distance to each satellite and the amount of time it takes to receive a signal delivered. Using distance readings from other satellites, it is possible to calculate a receiver position more precisely. Although it is the most used technology to acquire location, it does not have enough accuracy, up to 3-7 meters, to guide a robotic lawnmower through the garden. Fortunately, with the technology always improving and with the help of other technologies, it has become a more feasible choice over the years. That is the case of Husqvarna, Husqvarna's latest models use GPS navigation to guide the lawnmower in addition to odometry and other

inertial sensors. Another example is the segway navimow that uses a new system based on RTK -GPS technology. Navimow uses Exact Fusion Location System (EFLS) to control the mower position, which is a system that communicates with satellites and Global Navigation Satellite System (GNSS) antennas in the lawn as can be seen in figure 2.7. These GNSS antennas communicate to GPS satellites and then send back the signal to the mower, and with the help of other inertial sensors such as a geomagnetic sensor, odometry, differential barometer, among others, it is possible to determine the mower position more accurately.

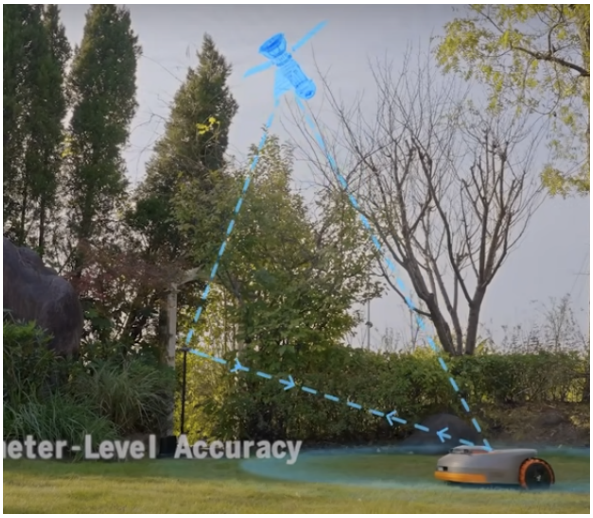

Figure 2.7: Navimow-Segway lawnmower using RTK-GPS technology [13].

2.3.4.4 Humidity sensors

Ambrogio L60 from Zucchetti Centro Sistemi SpA is the only robot currently in the market that does not use a perimeter wire and chooses sensor based technology to distinguish the grass from other materials. On the underside of the mower, a ring of humidity sensors detects the difference in humidity between the grass and other items. The mower detects the difference in humidity between the grass and a hard surface when it reaches the edge of the grass. It is then instructed to come to a complete halt and move in a different direction. As a technology, it offers both advantages and cons. The pros are that it is a relative simple technology and has a low failure rate, working well most of the time. [17]. Besides this, it imposes a number of restrictions on the operation of a robot lawnmower that employs this technology. The lawn cannot border a similarly humid area, such as flower beds or ponds, for the humidity sensor to perform properly. The Ambrogio L60 would be unable to detect this shift accurately, causing the mower to continue driving off the grass. This ALM is used only for smaller areas (up to 200 m^2).

2.4 Alternative technologies

2.4.0.1 UWB

The term Ultra wide-band (UWB) refers to a technique for transmitting data over a large section of the radio spectrum. This works by transmitting short pulses with a small

power output across a significant bandwidth. Essentially, UWB is a fairly low-power way to send a considerable amount of data. To determine location precisely in real-time, UWB solutions typically utilize techniques such as Angle of Arrival (AoA), Time of Flight (ToF), Time-Distance-of-Arrival (TDoA), and other measurement methods. Some solutions use their own unique measurement techniques as well. UWB is also particularly well-suited to operate around metal and other reflective surfaces, since its unique radio signature is a natural deterrent to multipath propagation. Furthermore, security is a strong point for UWB as well, since UWB transmissions are traditionally hard to intercept or compromise. Alongside this, UWB has some downsides, mainly retailed to infrastructural aspects. For instance, GPS is essentially ubiquitous for outdoor locations, and WiFi networks are easy to access in countless locations.UWB, hardware, on the other hand, requires new, unique hardware across a given area, and the general uniqueness of UWB hardware has not yet made wide interoperability feasible. Another disadvantage is the shorter battery life when compared to other wireless technologies. Another addressable impediment is the requirement for precise time synchronization between devices. Synchronization requirements can complicate the setup process, which is typically expensive for asset tracking systems [18].

2.4.0.2 WiFi

WiFi generally uses Received Signal Strength Indication (RSSI)-based techniques to locate and track things. This includes the use of RSSI in a process known as fingerprinting, which improves the reliability of locating systems but increases expense in terms of preparation and transmitter density. Wireless technology has a lot of difficulties with accuracy and dependability. Although accurate placement is achievable within a few meters, it is usually hampered by the presence of moving objects or other impediments. Due to the unpredictability of WiFi asset tracking, solutions are challenging to scale. It also makes WiFi inappropriate for a variety of WiFi settings or applications, such as those involving numerous moving objects. If system tracking solutions use the same wireless networks as internal communications or other essential activities, this becomes much more challenging. For starters, network congestion is frequent enough without adding hundreds of real-time location reports to the mix. Second, most existing WiFi installations do not have the best device location for an Real Time Location System (RTLS) or asset tracking application [18].

2.4.0.3 Infrared

Infrared tags are not particularly technical because they simply transmit codes with light, which are picked up by ceiling-mounted readers. While this necessitates extensive infrastructure, their technologically is very simple. The advantage of this is that infrared RTLS solutions are virtually error-free. Because they use light rather than radio waves, this solution cannot pass through walls and obstacles. When it comes to RTLS, this is advantageous because if the system informs that an asset is in room A, it is in room A without a doubt. Because radio waves can sometimes be picked up by other readers through walls, radio-based systems have a higher rate of false positives.

Technology	Accuracy	Scalability	Range	Data Transfer rates	Power consumption
UWB	10 cm -1m	very scalable	up to 100m	N/A	moderate
GPS	3-7 m	scalable outdoors	global acess	N/A	Low
Bluetooth	2-3m	limited	up to 100m	up to 2 Mbps	moderate
WiFi	-3m	limited	up to 50 m	up to 1Gbps	moderate
RFID	accurate to last scan only	very scalable	up to 600 m	N/A	very low

Table 2.1: Technologies comparison.

Previously, it was demonstrated what approaches Bosch Indego follows to cut the grass. As mentioned before, a systematic cutting system requires a more sophisticated location system. To complete this task, Indego only uses the perimeter wire and their inertial sensors to determine his location on their topological map. As proved earlier, solutions retailed on inertial sensors odometry and dead reckoning can have measures problems that can influence the behavior of the mower. With external sources continually causing inaccuracies on their sensors, Indego might become disoriented, and when this happens, it must return to the only absolute position it knows: the dock station. When it realizes is lost, it starts looking for the wire and then follows him till it finds the dock. Although this is not a critical issue, the goal is to enhance it by allowing the mower to adjust its location without having to return to the dock.

Typically, navigation challenges are related to the uncertainty of their measurement and employed algorithms, which can lead to self-lost issues. These issues are not crucial in terms of completing the task, but rather in terms of how efficiently and successfully they are doing so. The premise of the solution retails on getting more **reference points** that can help the mower to calibrate his current position. The best way to explain this is to give a daily example. When someone is lost, the first thing he looks for is someone that can assist him, or if that is not an option, he tries to find something referential, such as a gas station he is familiar with, for example. The concept is the same here: if the mower has some referential points in the lawn, it may calibrate its position automatically when it passes over them. In previous subsections, some techniques were approached but in this case, the right solution must be analyzed. Starting with UWB, which, while a highly precise technology to use, has infrastructure flaws that make it unreliable in this circumstance. GPS can be a useful tool too, but its precision constraints make it unreliable in this situation. WiFi, on the other hand, can be inaccurate in complex environments, making precise positioning difficult, and network congestion can be an issue. Infrared is out of the box primarily to its sensitivity. Beacons can also be a good solution, but it necessitates numerous networked devices, and, while the cost is low, in larger areas it may necessitate more devices, leading to higher costs, and it is not recommended in refining. As a result, RFID it can be the best solution to investigate, owing to its low cost, low power consumption, ability to function in a variety of settings, ease of use, and flexibility.

CHAPTER 3

RFID

3.1 Introduction

RFID is a technology that belongs to the Automatic Identification and Data Capture (AIDC) category. AIDC methods detect objects, collect data about them, and enter that data straight into computer systems. Radio waves are used in RFID methods to accomplish this. RFID is similar to barcoding in the fact that data from a tag or label is captured by a device and saved in a database. RFID has a number of advantages over systems that use bar code asset tracking software. The most notable difference is that RFID tags do not need to be visible in order to be read, whereas barcodes must be aligned with an optical scanner. At its most basic, RFID systems consist of three components: a **tag**, an **antenna** (which is usually integrated in the tag), and a **reader**. The tag identifies the item, while the antenna transmits and receives the signal, which is then processed by the reader. This data is then sent to a higher system for storage and software application.

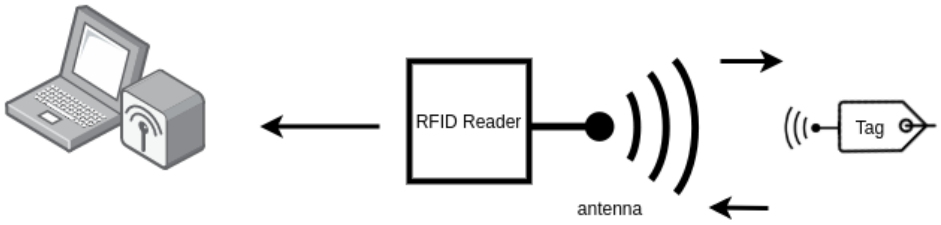

Figure 3.1: RFID system components.

3.2 Tags

Tags are composed by an antenna, for receiving and transmitting signals, and a chip Integrated Circuit (IC) that stores the tag's ID and other information. Depending on the size and sensitivity of the tag, it can be attached to a variety of objects and be several meters away from the reader. When the reader receives a signal from the tag antenna, it converts the radio waves into more usable data, which is then processed by the software [19]. Hundreds

of distinct RFID tags are available in a variety of forms and sizes, each with its own set of capabilities and choices tailored to particular environments, applications, and materials surfaces. One of the most common misconceptions about RFID tags is that they all have the same read range, regardless of their size, tagged objects, or materials. In fact, all of these elements work together to establish a tag's broad read range, but the tag's size has the biggest impact. Shortly, the larger the antenna tag, the longer the read range [20].

The IC on each chip is composed by four memory banks, **Electronic Product Code (EPC)**, **Tag identifier (TID)**, **User** and **Reserved** bank. Each of these memory banks gathers information about the tag or the item where the tag is attached, depending on the case. Each layer of memory is labeled with a number and title as follows [21]:

Bank	Memory Definition
Bank 00	Reserved
Bank 01	EPC
Bank 10	TID
Bank 11	User

Figure 3.2: Tag bank memory partition [22].

- **EPC**: contains the tag identification as well as the item information that is read by the reader, if the tag posses some software of the attached item (name, serial number or even a picture on a database).
- **User bank**: is used by the user to describe some data about the connected object, however it is not included in every IC (item type, last service date, serial number and so on)
- **TID**: contains a tag identification, which is a manufacturer-generated randomized unique number that cannot be modified.
- **Reserved memory bank**: includes the access and locks passwords that allow the user to lock the tag data and require a password to read it.

Besides this, tags can be classified with the types of memory access. Tags may be split into three categories of memory: **WORM, RO** and **RW**.

3.2.1 RO

Reading data from memory is the sole process that is possible in RO tags. Generally, they are programmed only once in the factory, containing the EPC code. Since it is only read memory, it is not possible to change any type of data. These tags do not have additional memory. They are usually associated with applications that require simple identification [23].

3.2.2 WORM

Theoretically, as the name enunciates, this type of tag can only be changed once. After that, the information is locked and cannot be ever changed again. This gives the user more

flexibility because he can control what will be written in the tag, but he cannot change the information again. WORM tags have more memory than RO tags [23].

3.2.3 RW

Users can edit or rewrite the information contained on the tag. The tag's serial number may be updated as needed, and readers can also write information to the tag, giving a read log. Once this memory has been changed, RW tags can be locked to prevent data overwriting or tag manipulation. These tags are much more expensive than RO and WORM tags, but they have a variety of applications that justify the additional cost [23].

3.3 Reader

A RFID reader is the brain of any RFID system, and it is required for RFID to work. Readers are devices that communicate with RFID tags by transmitting and receiving radio waves. Tags and readers must work in the same frequency, otherwise, communication cannot be done. Fixed RFID readers and mobile RFID readers are the two most common types of RFID readers. Fixed readers are always in the same place, generally, they are installed on desks, walls, gateways, or other immovable positions. Mobile readers are portable devices that are capable of reading the tags and communicate with a host system, while in motion. This gives RTLS, such as an autonomous robot, a lot of freedom when it comes to recognizing tags in motion.

3.4 Operating band

RFID is categorized in many ways, mainly by their operating frequency band. RFID bandwidth refers to the size of the radio waves which can be LF, HF, or UHF. Radio waves behave differently at each frequency and can influence many aspects of the RFID system, from the efficiency of the signal to the read range distance or even to the material placement.

A low frequency system, for example, has a slower read rate, but larger capability for reading near, or on metal, or liquid surfaces. A system that operates at a higher frequency, on the other side, has faster data transfer rates, and longer read ranges, but is more susceptible to radio wave interference caused by liquids and metals in the environment.

3.4.1 LF

Low Frequency systems operate in a range of 125-135 kHz, and they are categorized with the follow characteristics [24]:

- read range: up to 50 cm.
- extremely cheap price.
- works well near liquids, metals, global standards.
- slow data rate.
- usually passive tags.
- used in: animal tracking, access control, among others.

Low Frequency systems are ideal for use in regions where read range is not an issue. It can provide a variety of alternatives at extremely low prices, as well as the capacity to perform in practically any environment without requiring extensive maintenance.

3.4.2 HF

High Frequency systems operate in 13.56MHz, and they are categorized with the following characteristics [24]:

- read range: up to 1 meter.
- higher transfer rate compared to low frequencies systems.
- usually passive tags.
- read multiple tags simultaneously.
- works with metals and liquids.
- used in: personal ID cards, Poker/Gaming chips, Library books, access control, payments, and others.

3.4.3 UHF

Ultra High Frequency systems operate in range of 300 MHz - 960 MHz, and they are categorized with the following characteristics [24]:

- higher read range: up to 15 meters (passive tags) and 600 meters (active tags).
- very high speed rate.
- does not work well with metal and liquids.
- cost is low/medium.

While the read range is incredibly higher, there are some drawbacks to ultra-high frequencies. Higher frequencies result in shorter wavelengths, and shorter wavelengths result in higher sensitivity to external interference. Furthermore, they cannot be read while attached to water-containing objects because water absorbs UHF waves, and they also do not work well when attached to metals because the radio waves can be affected.

3.5 Coupling type

RFID communication between the tag and the reader is called coupling and can be either **inductive** or **backscattered** (radiative). Coupling is the process of transferring energy from one medium to another, in this case, energy from one circuit segment to another. This communication is required so that the reader can identify and retrieve data from the tag. This is similar to greeting a colleague before asking them a question. Coupling is a critical component of how RFID works. The frequency of the tag indicates which type is used, and the type of coupling used affects how far away the tag can be from the reader, whether the tag requires a battery to operate, and how the data stored in the tag is assessed [25].

3.5.1 Inductive

When the tag and the antenna are in the same range of one another, the reader generates a magnetic field that causes closer tags to couple with the reader. When the tag contacts the

magnetic field, a voltage is generated in its antenna, which serves as power for the microchip, allowing the stored tag data to be read using a technique known as load manipulation. This change will be recorded as 1's and 0's, which the reader will interpret as the tag data. This is called **inductive** coupling, and it allows the tag to work without the need of an onboard battery, as the power generated from the reader is enough to power the tag [26]. However, because it requires the reader to be closer to the tag in order to power it, this method of coupling is thought to work only in close ranges. This technique is used for **near-field communication**. In figure 3.3 it is represented a visual perspective of what was explained. This coupling is normally used in lower frequencies, LF and HF.

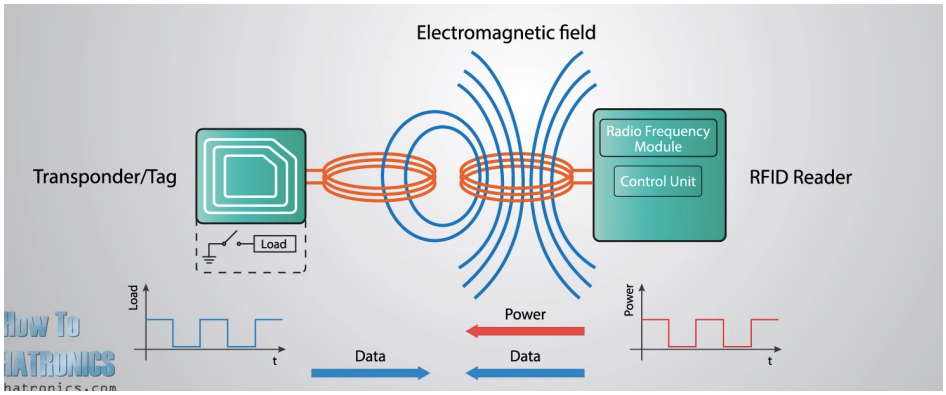

Figure 3.3: Inductive coupling [27].

3.5.2 backscattered

The reader starts to radiate electromagnetic waves, that are sent to any closer tag that captures them and reflects back to the reader - **backscattered** with their data. In this method, the signal reflected back depends on the characteristics of the tag (or any other object for that matter). This is similar to yelling in a cave and waiting to hear the echo. Basically, the tag gathers some electromagnetic fields to generate another magnetic field, which will be picked up by the reader's antenna. The main difference here is that instead of sharing a magnetic field, here is created two magnetic fields to communicate between the tag and the reader. This coupling technique is indicated for longer distances (**far-field communication**), [26] working in higher frequencies. This communication is exemplified in figure 3.4.

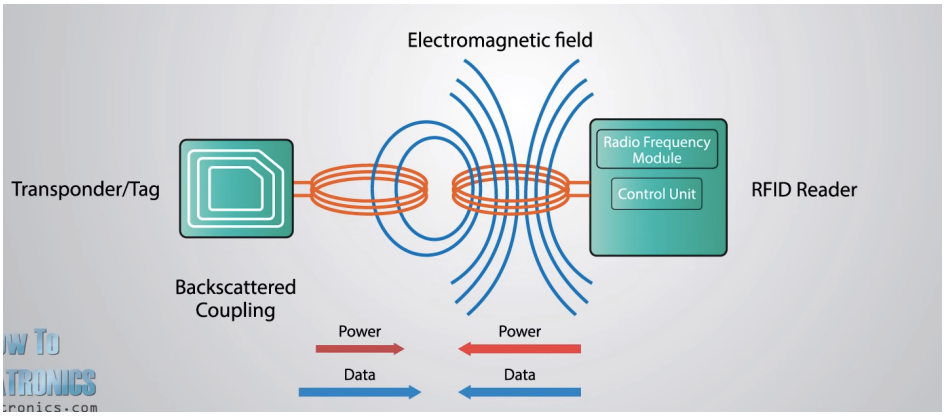

Figure 3.4: Backscattered coupling [27].

3.6 Active vs Passive

RFID systems can be either **active**, **passive**, or **semi-passive**. In passive systems, tags do not have their own power source. The reader sends a radio signal to the tag, which uses it as a source of energy to activate their circuit, and then it sends back a response with their internal data. They are cheaper, smaller, and easier to produce compared to active tags. Passive tags usually operate in LF, and HF. These tags can use both coupling types, the difference is that, as they do not have their own power source, the electromagnetic field generated (in each case) ca not be done in large distances.

On the other side, active RFID tags have their own power source. Compared to passive tags, they are relatively expensive. Having their own power source implies that they are not fully dependent on the reader's signal to communicate. Consequently, active tags can operate in higher ranges. Active tags can be divided in two types: **transponders** or **beacons**:

- **transponders** do not actively transmit signals. They are only activated when they receive a signal from the reader and send it back. Transponders typically have a longer lifetime because they do not send signals sequentially.
- **beacons** are not activated by a reader's signal; instead, they send out radio signals at pre-determined intervals. Here, the tag can start the process of communication without the need for a signal coming from the reader.

In figure 3.5 it is exposed the difference between the two systems, where it is possible to see the flow of the signal. In passive systems, the reader has to power the tag and the tag sends back the signal with the data. However, in active systems, the RFID tag can talk first sending their signal, while the reader is listening.

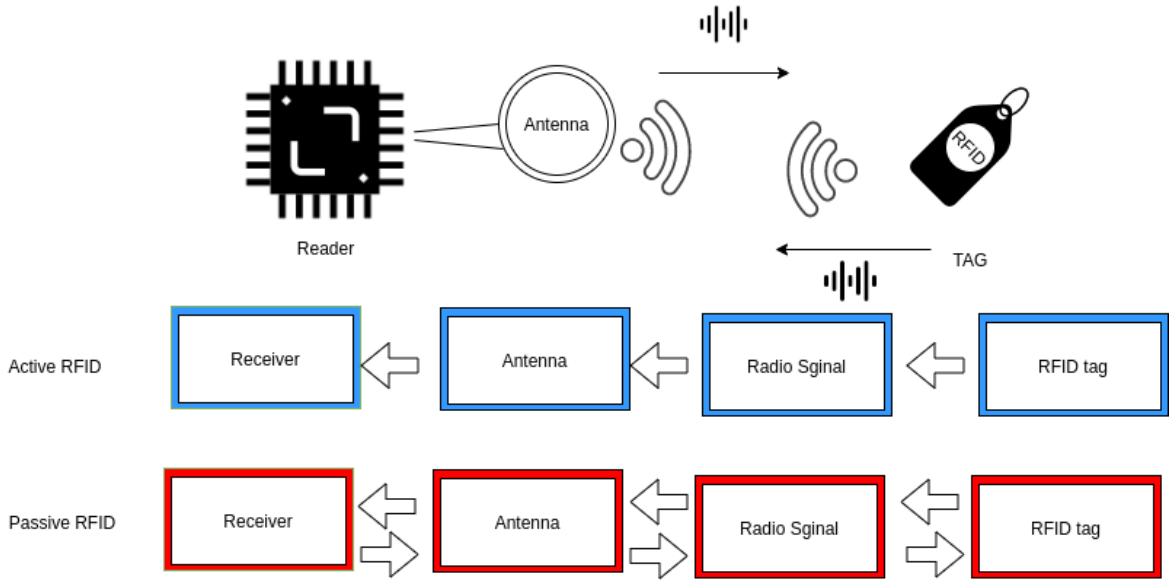

Figure 3.5: Signal flow: active vs passive RFID.

	Active RFID	Passive RFID
Read Range	20 - 100+ meters	Near Contact - 10 meters
Average Cost	15.00−50.00	0.09 −20.00
Frequency	usually UHF	LF , HF , UHF
Available signal strength from tag to reader	High	Very Low
Required signal strength from reader to tag	Very Low	Very High (tag must be powered)
Tag battery	Yes	No
Tag power source	Internal to tag	Energy transfer from the reader via RF
Applications	Vehicle tracking, Auto Manufacturing, Mining, Construction	Supply Chain Tracking, Manufacturing, Pharmaceuticals, Electronic Tolling, Inventory Tracking Race Timing, Asset Tracking
Pros	Long read range, lower infrastructure Cost, Large Memory Capacity, Very High Data Transmission Rates	Low cost per tag, Wide variety of tag sizes and shapes, Global standards, High data transmission rates, Long lifetime
Cons	Higher per Tag cost, Shipping restrictions (due to batteries), Complex software, High Interference from metal and liquids, few global standards	Moderate Memory Capacity, Lower read range,

Table 3.1: Passive vs Active RFID - review.

There are also semi-passive systems, which combine active and passive components where the circuit is powered by the battery, while communication is powered by the RFID reader [22].

3.7 Exploration of RFID in Bosch Indego

When the mower becomes lost on the map, Indego employs calibration as a technique of reallocation. The robot searches for the perimeter wire, which is then followed until the dock is founded, as part of the calibration operation. This is not a big problem in smaller yards

because the dock station can be set up quickly. In bigger fields, though, the mower might waste a lot of time returning to the dock. Even though it is not critical, avoiding this will improve Indego's calibrating efficiency significantly.

The goal of adopting an RFID system is to increase the number of referential location points, which will aid Indego not only in their reallocation efficiency but also in lowering their odometry error. The idea is to make tags acting as reference points, and the reader as the detector that sends their coordinates to the robot, which then will process the obtained coordinates and calibrate their measurements. This coordinates attribution may be accomplished in, at least, two ways:

1. with **WORM tags**: each tag has its own ID and coordinates associated. This requires a tag with more memory.
2. **RO tags database**: the reader has a database that stores the ID with the respective position associated. The tag just needs to have an ID, without having the need of additional memory to add more data. The storage procedure in the database can be:
   i imputed before the mower starts to cut.
   ii acquired while the mower is in movement.

Either in method 1 or 2 the coordinates attribution of the tags can be made in two ways: **statically** or **dynamically**. In the static method, the tag needs to be exactly in the indicated position. This requires an additional method to make sure that the tag will be exactly on the position that is supposed to be. In the dynamic method, it is the robot that gives the tag the position, writing to it (1) or reading from the database (2).

In terms of accuracy, the static approach is a better choice because if the position of the tag is absolute, it is much easier to calibrate the mower with better accuracy. However, this static method is not practically easy to achieve, mainly from a user point of view. It is very complex to create a mechanism that informs the user where he must put each tag on the yard. Contrarily, the dynamic technique uses Indego odometry measures to give the tag a position. This can be controversial since the goal is to reduce odometry uncertainties, however, it happens only the first time the mower passes through a tag (when the accumulation of the error is lower). Furthermore, in the implementation of the perimeter wire, when Indego is mapping, it can immediately attribute to each tag a position. This is almost the same as the tags being implemented statically because the mower is following the wire and the measurement error is almost null.

Summarizing, between the two approaches, the **RO tags** is the most feasible one. Tags with more memory are not only more expensive, but also more complex to implement. In contrast, with the second method, the tag information without the rest of the system is just a simple ID. In terms of security, this can be more robust because it is not possible to change data from the tag and if external sources try to read the ID it is impossible to know what that tag means. Although, this approach requires higher robustness when data is being stored. The database must be stored in a **non-volatile memory**. Non-volatile memory is a form of memory that is always accessible, meaning that data is not destroyed when the host system is

switched off or goes into sleep mode. Oppositely, volatile memory requires constant power to preserve data [28].

3.7.1 Selection of RFID technology

Before actually choosing which RFID system to implement, it is mandatory to make a statement on where the system will be implemented, and what will be the cost of the system. Two concepts must be determined when deploying a RFID system:

- **cost feasibility**: refers to determining whether or not installing a RFID system is financially feasible. RFID systems need an initial expenditure to test and work with various types of equipment and tags (which may be an expense if the technology fails). The deployment costs begin when the testing phase is completed. The timetable for getting a return on investment can only begin when a system has been deployed and is operating effectively.
- **environmental factors**: RFID systems can be harmed by specific materials and environmental conditions, resulting in reduced read ranges and a reduction in overall system accuracy. Metal and liquids are the two most frequent sources of RFID interference, but with the right RFID tags, equipment, and planning, they may be avoided.

Initially, deployment costs will not be taken into account since the system must pass through several levels of testing that are not yet fully established and completed. However, using the premise of "making it simple will make it better", the goal is to create a RFID system that is as easy and straightforward to use as possible. This is directly related to a reduced cost. Weather, water (rain), metal, and other environmental challenges are examples of external variables that might affect the reading process. Considering that, this RFID system will have the following characteristics:

- **tag memory type**: type of tag will be RO. Since the database approach only needs to read the ID of each tag and associate them to a coordinate. RO tags are much cheaper, meaning that the cost will not be a significant problem, which gives a larger testing margin.
- **reader type**: the reader will be attached to the robot while it is in movement. The objective is to have a microcontroller that will receive the information coming from the tag and store it in a database. A Hardware system must be developed. The simplicity, of the reader, is also an important factor to have in mind since tags must be easily interpreted, giving the possibility to decrease the complexity of the implementation. Simple readers have the added benefit of being less expensive than sophisticated readers.
- **frequency**: is one of the most essential characteristic to determine, since it affects all other attributes. Frequency has the greatest impact on all the parameters that might affect the read range. In this system, the read range will be determined by the distance between the tags and the reader antenna. Since the antenna will be attached to the bottom of Indego the distance between the antenna and the grass will be very low (up to 4 cm). Even with tags buried, the distance can be increased, but not significantly. This concludes that a system working with LF is more than enough to fulfill this

requirement. Low-frequency signals have longer wavelengths, and longer wavelengths mean less susceptibility to external interference, which is another advantage that LF will provide.

- **tag type**: passive tags will be the only option that must be chosen. Active tags are more reliable for systems that need higher read ranges and more accurate results, however, prices are too expensive, and the complexity is much higher. When working with low frequency, passive tags can offer more flexibility, are easier to implement, and have a longer lifetime.

	Option	Review
Frequency	LF (125-134Khz).	- Allows enough read range (up to 20 cm); - Low cost; - Less susceptibility to external sources; - Higher availability.
Passive vs active	Passive tags.	- Cost is very low; - No battery needed; - Higher availability.
Tags type	Read Only.	- Easy to use; - Cost is very low; - Do not need additional memory since the only data needed is the ID.
Reader	Mobile and open source.	- Higher flexibility; - Easy to implement; - Antenna in the bottom of Indego with enough distance of the ground where the tags will be.

Table 3.2: RFID chosen system - overview.

To show what was mentioned here, a system capable of being implemented into the robot was designed. The architecture and implementation of this system will be explained in the next chapters.

CHAPTER 4

System architecture

4.1 Introduction

In order to make the storage of the tags distributed on the lawn, it was necessary to build a system able to detect those tags, coming from the RFID reader, and associate or compare them to the robot position in the lawn. This was achieved with a microcontroller that controls the information between the RFID reader and Bosch Indego (ALM). Furthermore, in order to have a different perspective about what was happening in real-time, it was built an app viewer capable of showing Indego's behavior and the detection of the tags in a web app. Therefore, the system can be divided in three layers: **backend**, **middle-end** and **frontend**:

- **backend**: it is where all the requests made to Indego's are handled as well as the storage of the RFID tag's position, detected by the reader.
- **middle-end**: sends requests to the backend, and after being parsed, are sent to the next layer in Java Script Object Notification (JSON) format. Aside from that, all test computations, whether at the level of creating graphs or obtaining errors, are performed in this layer.
- **frontend**: demonstrates a visual tracking of the system in real-time. By receiving data from the middle-end, it is possible to watch the robot moving on a 2D map, which was later obtained after the garden map phase. With this tracking system in real-time, an **app viewer** was built successfully.

The role of each layer, and how it was designed, will be explained in-depth in the following sections. Before that, a description of the hardware components required in this project, as well as the selected components, will be provided.

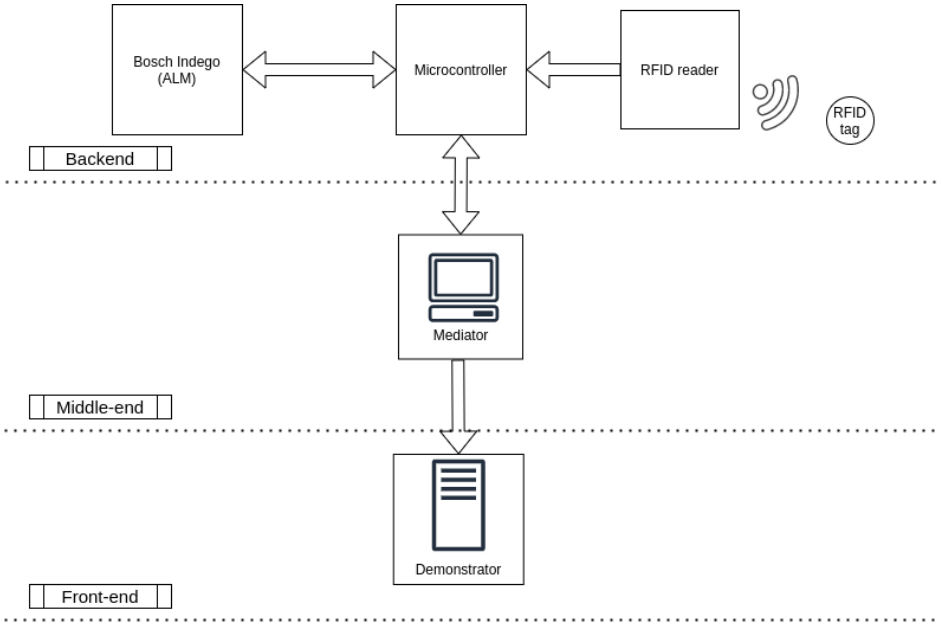

Figure 4.1: System architecture.

4.2 Hardware

This section will make a review about the hardware needed to build this system, explaining the function and requirements of each component.

Starting with the microcontroller, which is used to control the data flow between the RFID reader and the ALM data via serial port. A **microcontroller** is a small integrated circuit used in embedded systems to control a single operation. On a single chip, a microcontroller has memory, a CPU, and input/output peripherals. Microcontrollers, also known as embedded controllers or MCU, can be found in robotics, office machinery, cars, medical equipment, mobile wireless, vending machines, and household appliances, transceivers, among other things. They are basically simple little computers with no complex frontend Operating System (OS), meant to operate minor elements of a bigger component OS [29].

RFID reader is the detector of the tags. The antenna must be positioned at the bottom of Indego, within a safe distance from the ground, and work at 125kHz. In order to provide the best possible tag detection, the distance between the tag and the antenna must be established through many levels of testing. To do so, a **three dimensional (3D) component** must be created so that it fits the antenna's size and the properties of the location where it will be placed in Indego. Tags will serve as location markers on the environment and should be put in strategic areas. Tags must be RO and consistent with the reader's preferences. Table 4.1 shows the main functions, requirements of each hardware component used in this system, and the option used.

	Function	Requirements	Option used
micro-controller	- Control flow between RFID reader and ALM data; - Stores tags in a small database; - Sends data to the middle-end (depending on what is being asked).	- At least 2 serial ports for each main component; - Wifi; - Non-volatile memory.	Esp32.
RFID reader	- Reads tags ID and sends it to micro controller.	- 125 kHz; - Capable of reading tags up to 5 cm.	RFID Grover 125 Khz.
Anntena component.	- 3D component that incorporates the RFID antenna in Indego.	- Placed at the bottom of the robot within the accurate range.	Developed.
RFID tags	- Referential points in the environment; - Calibrates ALM coordinates; - Each ID is associated to a coordinate (x,y) in micro controller's database.	- Must work in 125kHz and be compatible to the used reader; - Antenna diameter >2cm.	EM4100 card and key-chain.
Indego ALM	- ALM that uses a systematic mowing type and only uses inertial sensors to control their motion.	—	Indego S

Table 4.1: Hardware overview.

4.3 Backend software

4.3.1 MCU to ALM

The Connect Module (CM) in Bosch Indego is a communication board that handles the messages from Indego to the exterior (Bosch servers, apps, and other services). The CM can be considered to be an embedded device, as the broad's firmware is completely independent of ALM. CM uses a serial interface to connect to the Indego's main circuit board.

Bosch Indego uses a protocol that defines a simple application layer for conveying command messages from the CM module to the main chip and response messages returned from the main chip to the CM module. ALM communicates with the CM module using Universal Asynchronous Receiver Transmitter (UART) with the following serial port configuration:

- baud rate at 115200 bps.
- 8 bits data packet.
- no parity bits.
- 2 stop bits.

The development of this device consists of a circuit that will substitute the Bosch connect module in its simplistic form. Instead of being connected and communicating with Bosch services, this system will communicate with the servers developed in the app viewer, following Indego protocols.

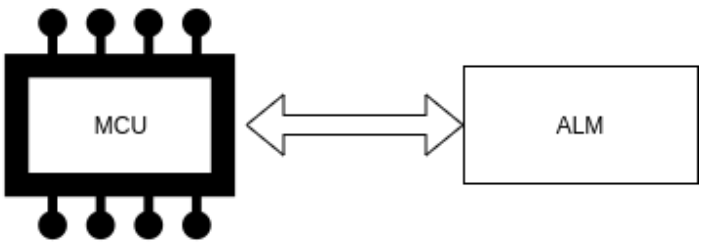

Figure 4.2: MCU and ALM data flow.

Each CM - ALM message is a sequence of contiguous binary bytes. The general message format is as follows:

Figure 4.3: ALM general message format.

- **Start of transmission character (STX)** - defines the start of a message (0x02)
- **End of transmission character (ETX)** - defines the end of a message (0x03)
- **16 bit Cyclic redundancy check (CRC)** - is a method for detecting errors. The CRC, which is a form of checksum, creates a fixed-length data set based on the construction of a file or bigger data set. CRC is a hash function that identifies unintentional modifications to raw computer data in terms of its application.[29]
- **Message size**: each message frame is a maximum of 256 bytes. For messages greater than 256 bytes, multi frame-message protocol is used.

In a simplistic way, messages from CM to ALM have a byte identification that informs ALM what type of message is required and then ALM responds with the desired answer or with an error. Basically everything that is available to the user can be requested in this layer and even more. However, for this system, the only important requests are the one's related to the robot position and the map of the area. The robot's position is used to keep track of the robot, future in the app viewer, and to associate a coordinate to a tag when it is detected. Map data is used to make the display in the frontend. The basic idea of each request is to send it within the ID of the command and the rest of the parameters. If the message is correct ALM will send the data requested in a response message format where each byte must be analyzed according to the documented protocol. This flow is demonstrated in the next figure:

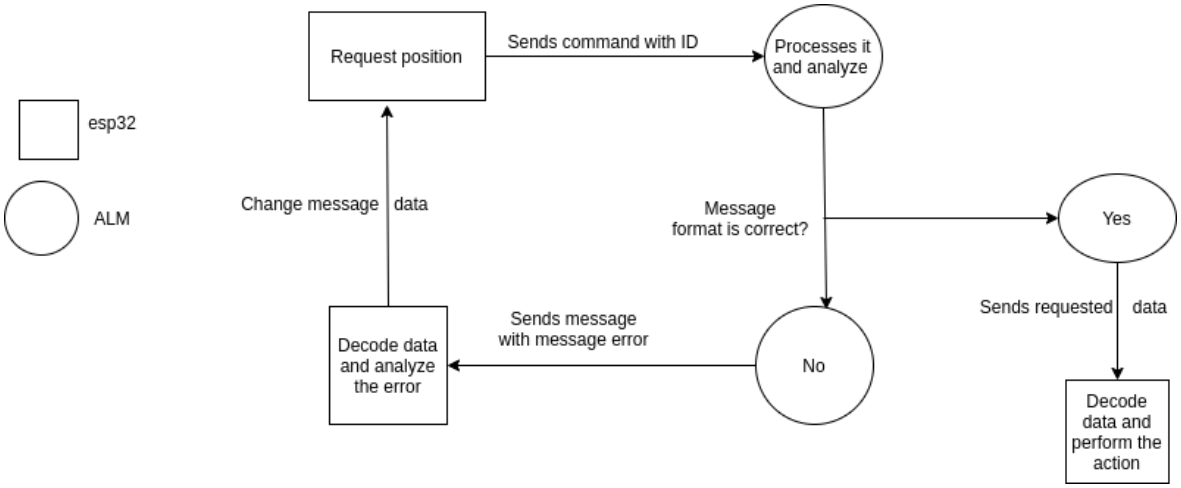

Figure 4.4: Example byte message flow.

Between these requests, mapping request is the most complex to implement, since it is a lot of data to decode, and it iss not byte wise like other requests. This is a topological map,

and the retrieved data shall be decoded into a grid where each cell has a different meaning. Oppositely to other requests, this request is bitwise, which means that each bit belongs to a sequence of bits that have different meaning in each row/column. Basically, in this message protocol, each sequence of bits have different meanings depending on what were the previous bits. Each cell should describe the garden with information like: inside, outside, border or mowed. This request is always greater than 256 bytes, even if it is a very small map, meaning that it will use a multi-frame command/response protocol. When the map is received and decoded, it has to be parsed again in order to ignore all the outsides of the map because without that, the robot position will not coincide with the position in the map (it has an offset).

4.3.2 MCU to middle-end

In this case MCU will act either as a source or as a receiver since it will receive requests from the layer above and send the corresponding data. To establish complete communication between entities that do not need a serial transmission, it is necessary to construct a communication channel that allows data to be parsed between parties, which for this situation will be the middle-end layer. Between the current methods available to make this type of communication, sockets are the ones that stand up. A socket is a two-way communication channel. This channel is created within socket procedures, and it is used to transfer data between applications, either locally or via a network. In this system, data transmissions between the backend layer and the middle-end uses Transmission Control Protocol (TCP) sockets, which guarantee data delivery.

In order to respond to these requests made by the middle-end entity, a protocol must be established. The idea is to build a similar message protocol used in ALM. The physical layer's stream of bits is split into data frames at the data link layer. The size of each frame to be sent differ in variable-length framing. As a result, a pattern of bits is employed as a delimiter to distinguish between one frame and the next. If the pattern appears in the message, measures must be implemented to prevent this circumstance. That is why **byte stuffing** should be implemented. Byte stuffing is a method where a byte is inserted in the message to distinguish the message body from the identifiers. This byte is normally called Data Link Escape character (ascii) (DLE). Whenever a pattern identifier or a DLE character is found in the message body, a DLE is added to the message before the identifier and then is added to the Xbyte in order to fully distinguish the identifier.

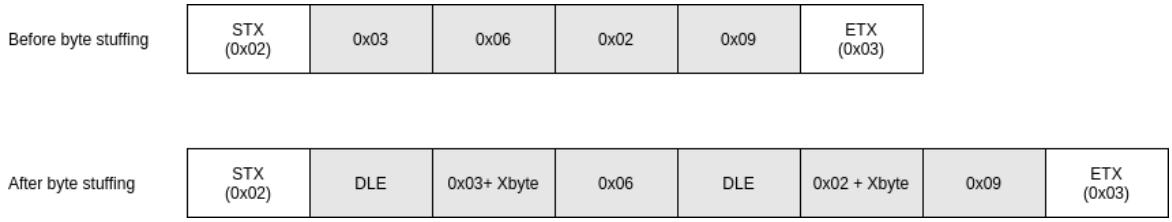

Figure 4.5: Byte stuffing example for coordinates (36,29).

In this example, the coordinate has two identifiers in the middle of the message, 0x02, and

0x03, respectively. Without byte stuffing the other entity will not have access to the data, because after a STX comes an ETX which means that it is the end of this frame. With byte stuffing and applying the inverse process, the other party can read the data correctly. This technique should be implemented in every data transmission within the socket channel.

4.3.3 MCU to RFID reader

Communication between MCU and the reader is limited to one side because the tag is RO and the only thing the RFID reader will have to do is read the ID from the tag and send it to the MCU. For that, MCU must be constantly asking the reader for new data.

Using UART with the following serial port setup, the MCU communicates with the RFID reader:

- baud rate at 9600 bps
- 8 bits data packet
- No parity bits
- 2 stop bits

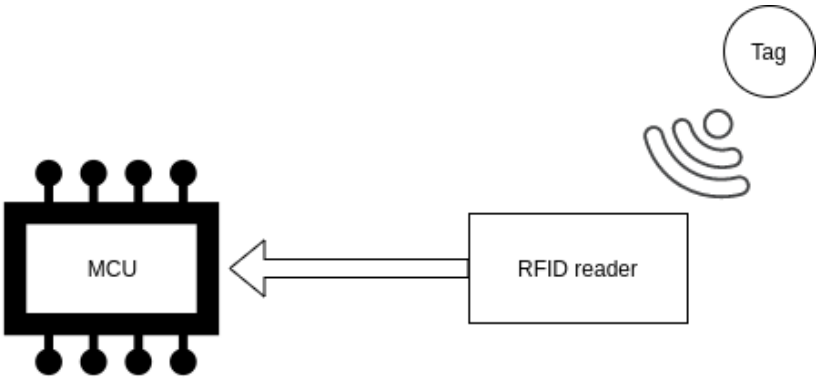

Figure 4.6: MCU and RFID reader data flow.

As shown in figure 4.3 the message coming from the tag starts and ends with an identifier. The ID is composed of 12 bits of characters. To detect a tag, the RFID port must be opened and constantly waiting for new data. When the tag is finally read, it should be stored in a database alongside the respective position.

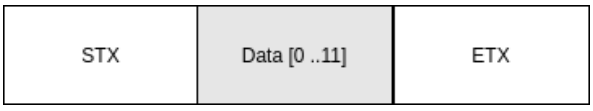

Figure 4.7: RFID message format.

4.4 Middle-end software

This system must be independent, meaning that it should be able to carry out its tasks detecting and associating tags without the need for external involvement. However, since this system is replacing Bosch CM and cannot be managed by their services, a layer capable of sending requests to ALM is required. The only accessible source is the Indego display panel,

which is not user-friendly to operate. These requests are not just for ALM operations; they're also for analyzing what's in the tags' database. Aside from that, the frontend needs some sort of data source, which is likewise made at this layer. Stating that the role of this layer can be divided into three sub-layers:

- **backend requests**: can be divided in two types: **required** and **control** requests. Control requests are those that can handle the mower's basic activities, such as starting, stopping, and returning to the dock station. On the other hand, required requests are those that must be implemented because other levels rely on them. This group includes requests such as getting the map, receiving tags, and track the ALM position. To read the data correctly, these requests must use the inverse protocol specified in figure 4.3.3.
- **frontend data source**: gathered data from required requests are used in this sub-layer. The role is to decode data coming from the backend to an understandable format that will be directly accessed in the frontend and displayed in the web app. This sub-layer acts as the "backend app viewer".
- **results analyzer**: this sub-layer is responsible for determining the outcomes, by generating and computing errors, graphs, and tables, with data parsed from the backend.

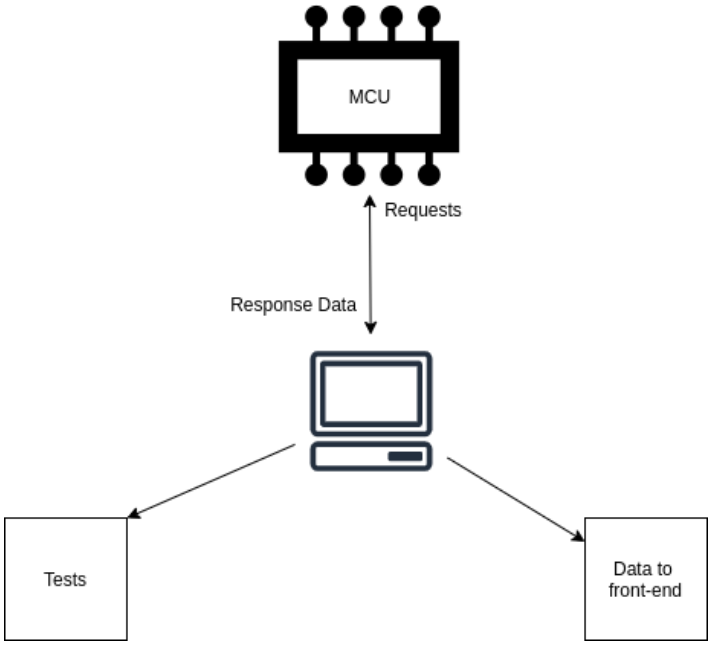

Figure 4.8: Middle-end architecture.

4.5 Frontend software

The frontend is the system presentation layer, which can be considered as the display of the app viewer. The objective of this app is to represent the behavior of the mower in real-time in an two dimensional (2D) environment. The purpose is to observe the robot in motion and the depiction of the tags as it goes through them using the map obtained previously. The

architecture of this layer is shown in figure 4.9 where it is possible to observe a server receiving data from the below layer, which then is displayed in the web app.

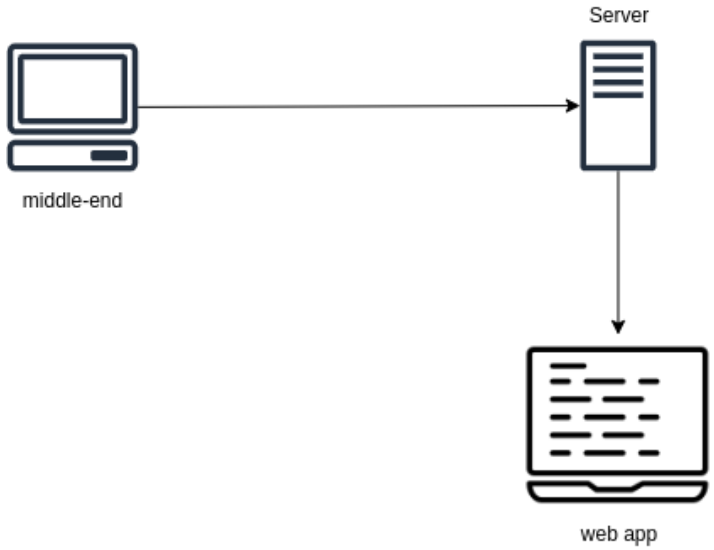

Figure 4.9: Frontend-end architecture.

The next chapter will portray the implementation that was made with this type of architecture, explaining the methods and algorithms used in each layer of the system, both at the hardware and at the software level. In addition, the final result of both, the integration of the system in the mower, the user interface, and the app viewer is also demonstrated with several illustrative images.

CHAPTER 5

Implementation

5.1 Introduction

This system was designed to operate in the same way as an embedded system. An embedded system is a piece of microprocessor-based hardware with software designed to perform certain functions, either as part of a larger system or on its own. In essence, it is an integrated circuit designed to do computing for real-time operation. In this situation, the system was designed to be embedded in Indego performing tasks that were capable of reading tags and associating them to a possible re-calibration of Indego's pose. Although this could not be achieved since the implementation in the Bosch layer was not done, the system is prepared for it. This means that despite the system being embedded in the robot, its functionalities do not affect the robot's behavior. The three layers of this system have different purposes that complement each other. The backend was created to function as an embedded system that reads the tags and compares them to the mower's location to determine whether it needs to be calibrated. Since this last part was not done and with the objective to demonstrate this concept, another layer was created. The middle-end layer was created essentially to determine the errors and analogies between the tags and mower position, that could be used to prove the concept of this dissertation. Later, the front-end was built in order to have a visual point of the system performing in real-time. Two sorts of tests are performed in this system: the first is a **static** test, in which the coordinates of the tags distributed throughout the field are already known. The second is a **dynamic** test in which, instead of being predetermined, the robot assigns a coordinate to the tags the first time it passes by them.

This chapter will explain the implementation of each layer, describing each step taken, and the results obtained, starting with the hardware circuit description followed by the implementation taken either in the lower level(backend) and at a higher level (middle-end, frontend).

5.2.1 Hardware

Table 4.1 explained the required hardware to build this system and the function of each one. This section will explain the circuit that has been mounted to be embedded in the robot, as well as the procedures for printing the 3D component.

In figure 5.1 it is possible to observe four components in the circuit: ALM, esp32, the RFID reader, and a DC-DC step down.

- **DC-DC step down**: is a voltage reducer that serves to convert the 18 Volt coming from the robot to 5 Volts. This DC-DC converter is required both to power the esp32 and the RFID reader.
- **ALM**: it has 3 connections, where the first two RX and TX serve to complete the communication by serial port with the microcontroller. RX is the one who receives the requests from the microcontroller, while the TX sends the response. Voltage drain drain (Vdd) is responsible for supplying the voltage to the converter mentioned above.
- **esp32**: it is the system's brain. RX1 receives the data from the robot, while the TX1 connection sends the commands to ALM. The RX2 link receives the data from the reader. In this specific case, the connection of TX2 is not necessary because communication with the reader is unidirectional (from reader to esp32) RX1, TX1 corresponds to UART1 while RX2 belongs to UART2, the other connections are responsible for feeding the microcontroller.
- **RFID reader**: is responsible for detecting the tags, whenever a tag is detected it is sent, via serial, to the microcontroller through the UART on the reader. As previously stated, this reader can only communicate in one direction as it is RO. Like esp32, this reader is also powered by ALM through DC-DC Step Down.

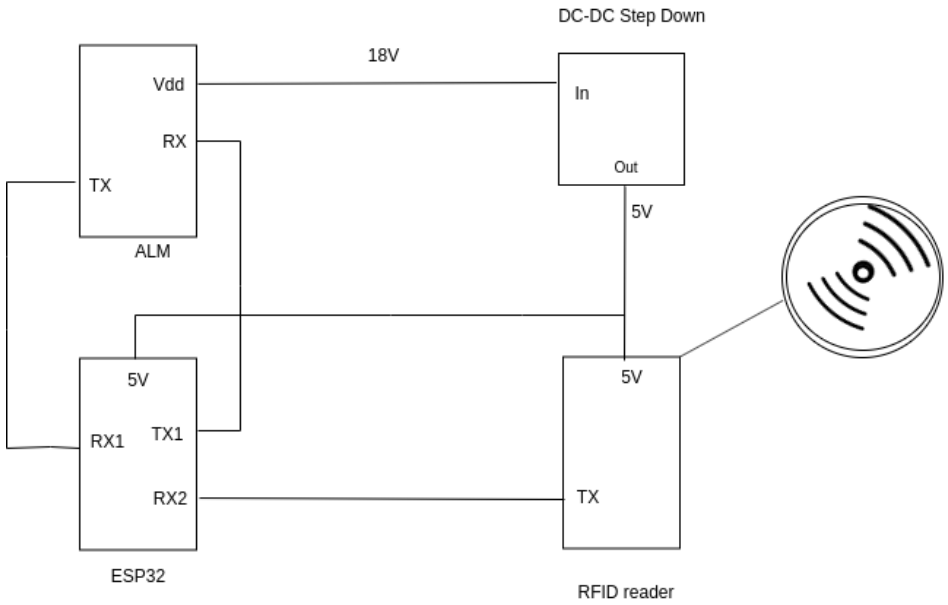

Figure 5.1: System circuit.

Alongside this, a 3D element was constructed to support the antenna in order to integrate it into the robot. The component was positioned at the robot's bottom end, where it would not interfere with the other robot mechanisms, even with the blades in place. (figure 5.3(1)). Following that, a schematic was created with the necessary measurements for both the antenna and the section where it would be installed (figure 5.2(1)). The item was then drawn using **tinkercad**. Tinkercad is a free 3D modeling platform that is well-known for its simplicity. It is entirely web-based, which make it available for anyone.[30]

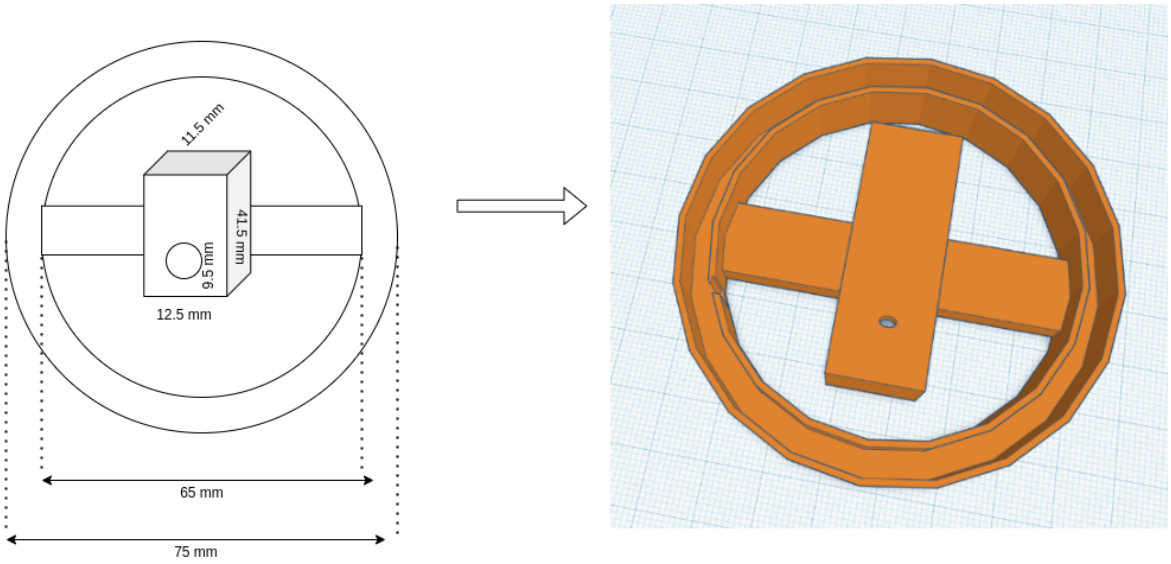

Figure 5.2: 3D component prototype and development (tinkercad).

The ultimate outcome of the 3D component and its incorporation in the robot can be seen in figure 5.3(2). The distance between the antenna and the tags may be adjusted when the piece is placed, providing flexibility to reduce the distance to detect the tags properly.

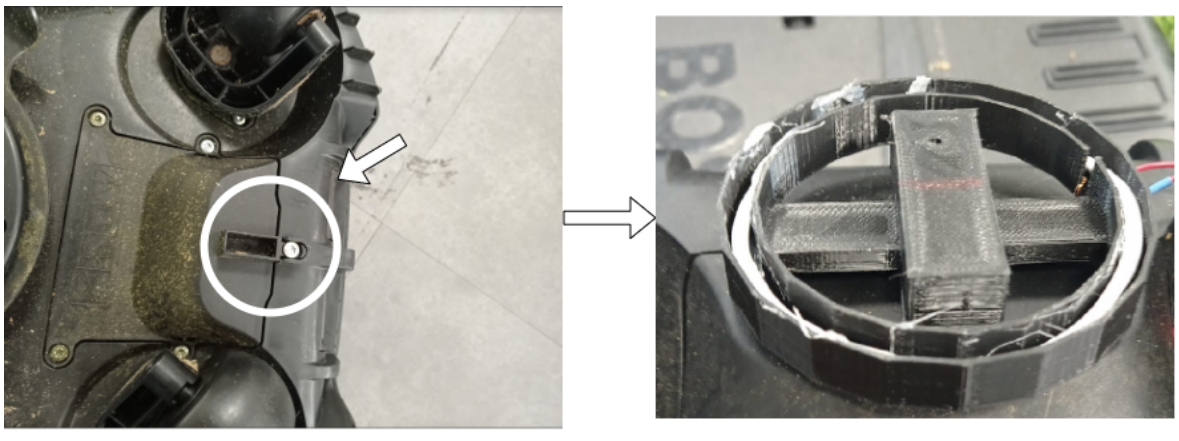

Figure 5.3: 3D component placement (1) and final product (2).

5.2.2 Software

Embedded system programming is a sort of computer software that gives a machine a set of instructions to follow in order to operate. C++ language is an object-oriented programming language that can perform as a low-level language. Low-level languages provide less abstraction from a computer's instruction set architecture. These languages are used to be close to the hardware. This layer of the project was built in C++ using PlatformIO, an embedded platform [31].

This part of the project is divided into four main classes: one that handles only the ALM tasks code another that handles the RFID reader tasks, the mediator between these 2 classes, and the main one that launches those three:

- **ALM tasks**: it is where all the requests made directly to the robot are made. All Bosch ALM message protocols are implemented here in order to communicate correctly with the mower. This class has two main threads running, where the first one is constantly asking the robot their position, and the second one manages the requests from a client (middle-end) if some is connected.
- **Reader task**: is composed by a thread that is constantly checking if there is some tag to read. Tag detection procedure is described in figure 5.4 diagram [32]. In this system, a tag is composed of an ID and a position (x,y). When a tag is successfully read, it is stored in a structure similar to a dictionary (database). Each dictionary key is made by a tag ID and its associated location. Values are made up of the different locations read with this ID. These different position measurements with the same tag are then analyzed in the middle-end.

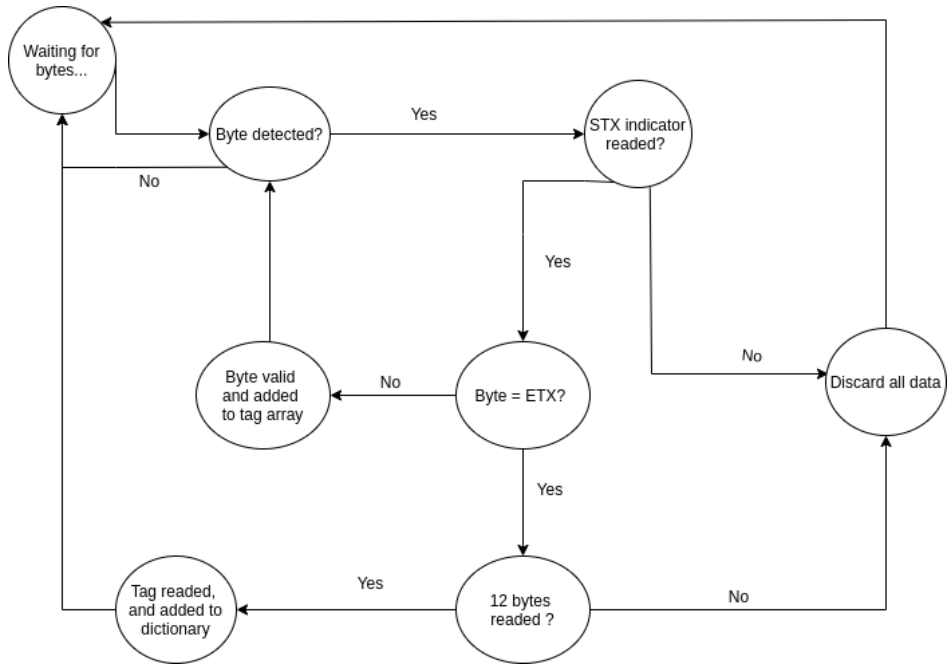

Figure 5.4: Tag detection diagram.

- **mediator tasks**: handles the control between threads, client requests and data control.

This entity can be divided in three parts: **data control**, **threads synchronization** and handling of **client requests**.

- **data control**: implements mechanisms that are capable of reading data either coming from a client, the reader, or ALM and storing it in a buffer. This data is saved in a **circular buffer**. A circular or ring buffer is a fixed-size buffer that works as if the end and the beginning are connected. Circular buffers use First In First Out (FIFO) a technique where the first element to be read is the first element that is removed. This buffer is composed of a tail and an end pointer. When new data arrives, the head pointer advances. When data is absorbed, the tail pointer advances. If the tail reaches the end of the size of the buffer means that the buffer is full. Otherwise, when the tail is equal to the end pointer, it means the buffer is empty. Another important property is the fact that when the buffer is full, it simply rewrites new data at the beginning of the buffer. With this, data is always updated. The diagram in figure 5.5 demonstrates an example of the flow of the reading and writing process in ALM thread position.

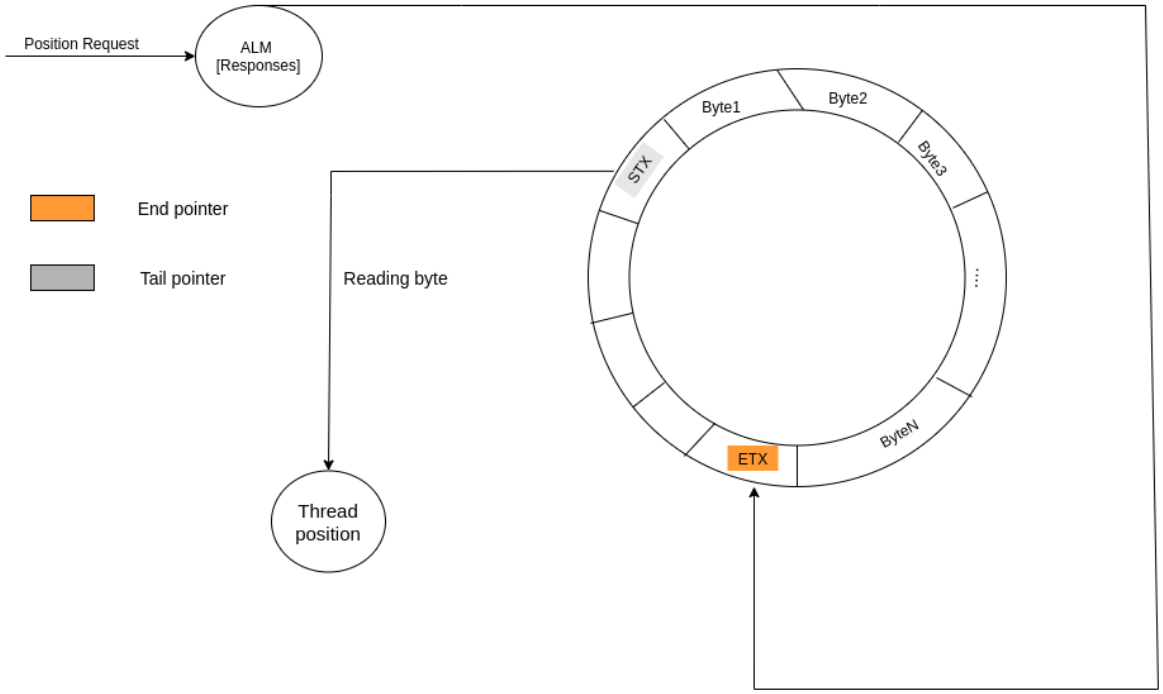

Figure 5.5: Thread reading from buffer.

- **threads synchronization**: the execution of two or more threads that share the same resources must be controlled in order to avoid critical conflicts. In this system, there are a lot of resources that are shared between threads. This synchronization is handled with the implementation of semaphores. Semaphores are used to control access to shared resources. They can be seen as a smart counter. This counter is always greater or equal to zero. When it is zero, the resource is blocked, meaning no one can access this resource. Thread synchronization is used to manage buffer

access, client requests, and thread states.

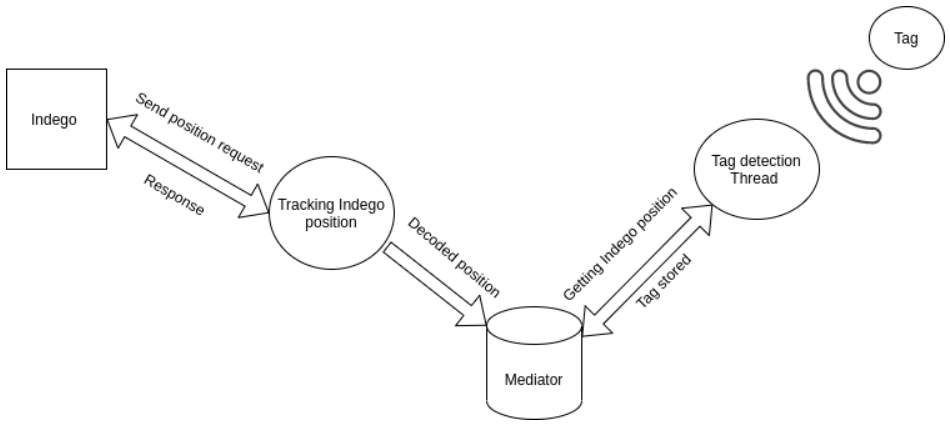

Figure 5.6: Threads basic flow without client intervention.

– **requests handler**: a socket server is initialized and waits for a new client. When a client arrives, all threads are informed. When this happens, threads start checking if the client sent anything. If the client sent something, all bytes are stored in the buffer to be interpreted later. The first message of the client is always a command ID. Depending on what was sent, **requests handler** thread, handles the request and performs the action according to the command ID. When the request is handled, this thread is blocked again and the other two are unblocked and can work normally again. To forbid blocking threads while the robot is in motion, the only requests allowed are those that do not affect the robot behavior like: stop mowing, return to the dock, pause, among others.

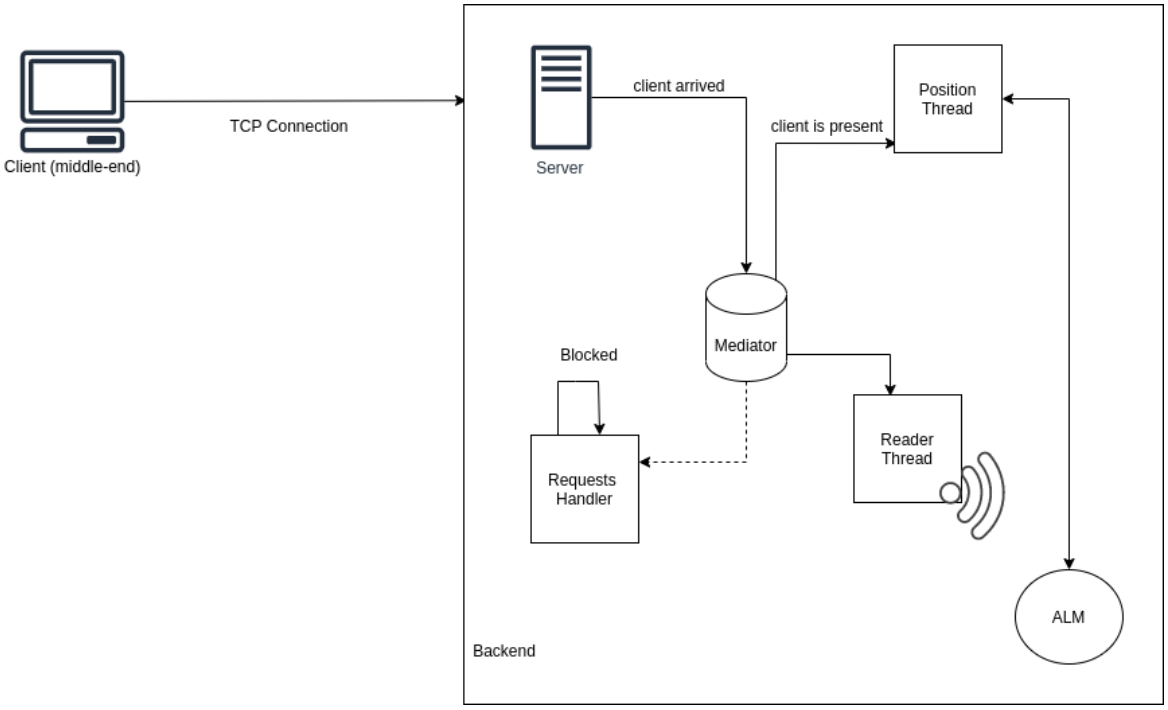

Figure 5.7: Client initialization diagram.

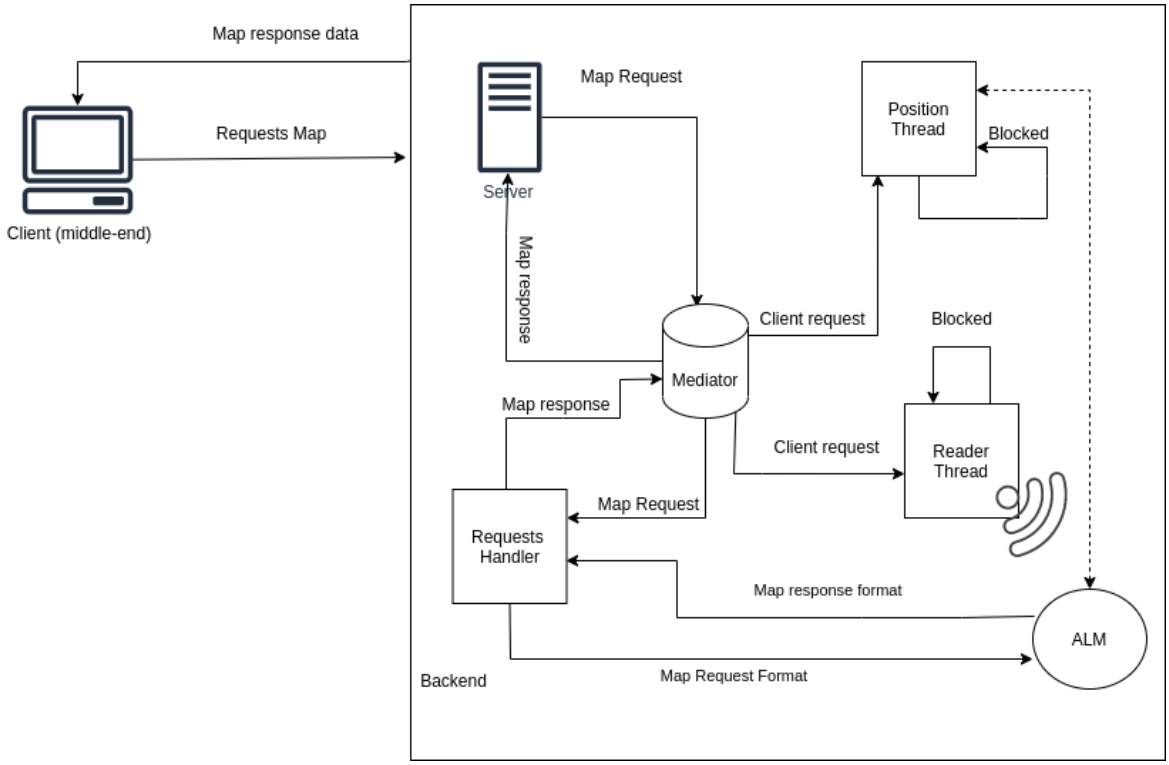

Figure 5.8: Map request diagram.

5.2.2.1 Tag Calibration

Odometry data is always computed with the robot's center as reference. In this system, the antenna is at the extreme of the robot rather than in the center, and this makes the determination of the true location of the tags challenging. This occurs because, instead of comparing two tags, the comparison is conducted between two different centers of mass. Basically, if different coordinates are compared when reading with direct assignment, the robot can be reading two different points in different locations, depending on their **orientation**. This reality is depicted in the figure below:

Figure 5.9: Orientation problem real representation.

In figure 5.10 tag is represented by the circle's center, while the robot's center of mass is represented by all the points on the circle's edge. If the assignment were done directly as represented in the figure, there would be a significant deviation from the tag's actual location, causing future results to be contradictory. This occurs because the center of the robot will vary depending on the mower's direction, which will lead to different coordinates for the same tag.

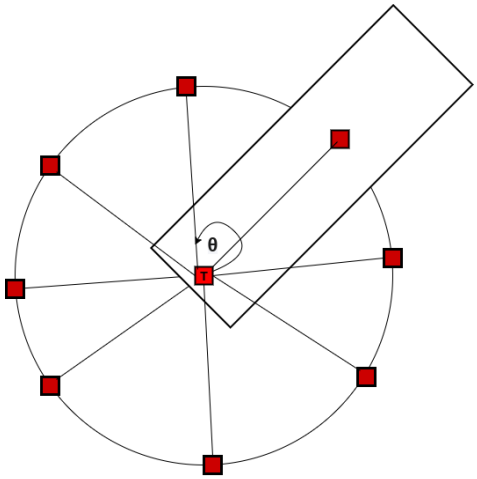

Figure 5.10: Tag attribution without calibration.

To solve this anomaly, the offset between the antenna and the robot, as well as the robot's orientation, must be known so that, regardless of the robot's direction, the true location of the tag can be calculated, as shown in the figure below.

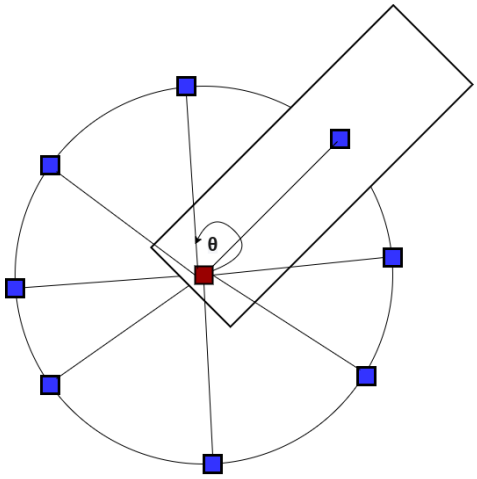

Figure 5.11: Tag attribution with calibration.

To perform the necessary estimations, it was necessary to know the distance between the center of the antenna and the robot's center (**R**), as well as the **size of the map cell** and the robot's **orientation** (θ).

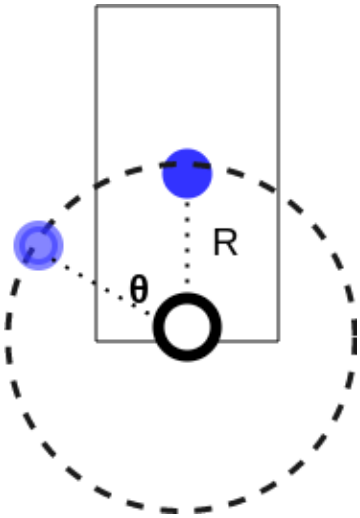

Figure 5.12: Robot range within the tag.

- **R real**: is the distance measured in reality. In this case it was obtained with a ruler, giving 150 millimeter (mm) .
- **R in cell size**: the real distance is not enough, since each coordinate in the map depends on the map cell size. With the real distance and the cell size of the map, it is possible to determine the offset.

$$R(offset) = \left\lfloor (\frac{realDistance}{cellSize}) \right\rceil \tag{5.1}$$

  In the case of one of the maps where tests were made, the cell size is 120 mm which gives an offset of 2 cell sizes.
- **robot's (θ)**: robot's orientation is obtained with two methods:
  - calculating the angle between two vectors, where the first vector is composed by the second last position and the current, and the second by the last position and the current position:

$$\theta = \arccos{(\frac{a.b}{|a| * |b|})} \tag{5.2}$$

  - in case last position and second last are equal, θ is obtained by calculating the angle between the robot's present and prior position.

$$\theta = \arctan{(\frac{(x2 - x1)}{(y2 - y1)})} \tag{5.3}$$

- **tag final position**: knowing the robot's coordinates, orientation, and the offset on the map, the position of the tag can be calculated using the following equation:

$$(x_T, y_T) = \begin{cases} R\cos(\theta + 180) + x_R \\ R\sin(\theta + 180) + y_R \end{cases} \tag{5.4}$$

  As can be observed, a 180º translation is done in each axis because the result with θ does not provide the correct direction for subtracting or adding to the coordinates in the required direction. Figure 5.13 illustrates this algorithm with different examples.

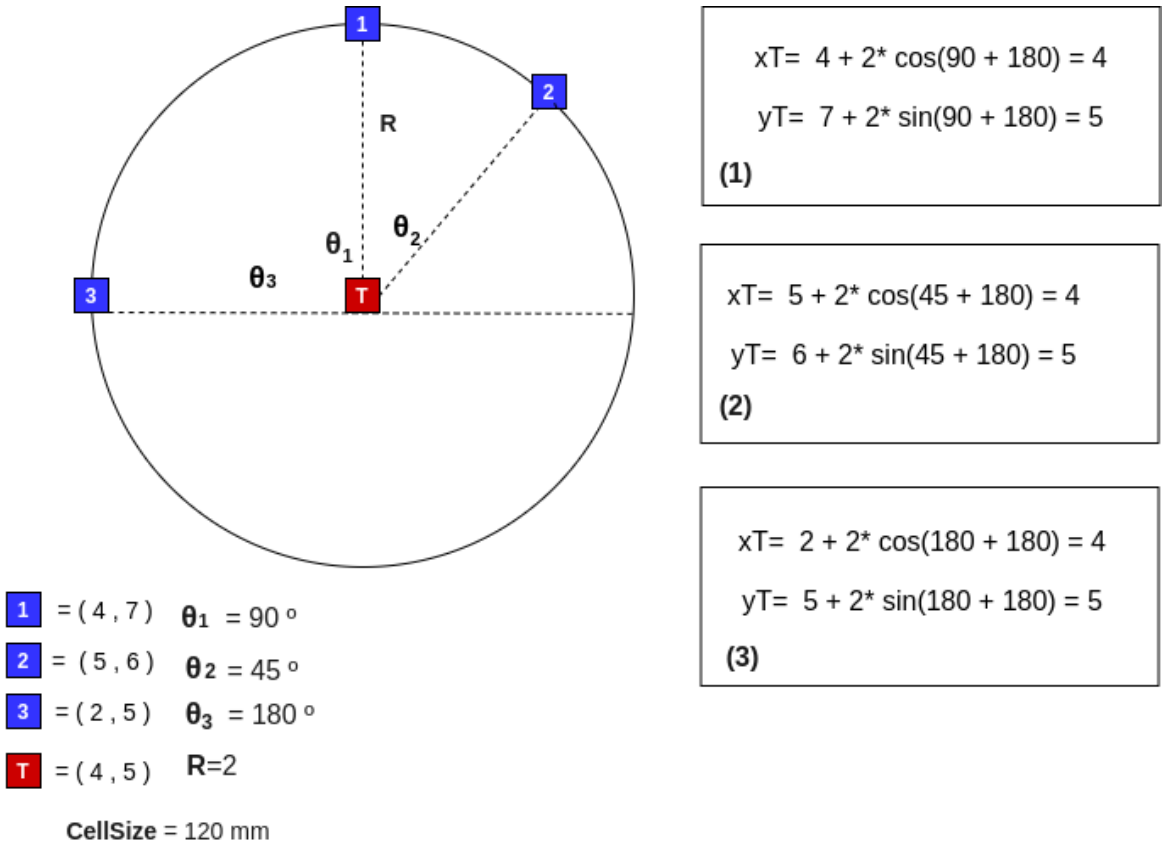

Figure 5.13: Tag calibration examples.

Through these maths, it is possible to obtain a tag's relative position with more precision. However, each of these measures has a variance related to the offset or the odometry itself. Considering that the uncertainty is determined by half of the smallest scale, the odometry and offset uncertainty can be represented as follows:

$$odometry = (x, y) \pm \frac{cellSize}{2}\text{mm} \tag{5.5}$$

$$offset = \pm\frac{R * cellSize}{2}\text{mm} \tag{5.6}$$

Following the example above, a 240 mm cell will be represented by two cells, with a cell size of 120mm, this means that if the robot is at 130 mm it will be represented as 2 cells. The same happens with the offset, and that is why, when performing the tests, all these deviations are discarded.

5.3 Middle-end and frontend development

To take benefit of the data supplied by the below layer, two layers were developed (middle-end and frontend), at a higher level. High-level languages such as python, JavaScript (JS), Hypertext Markup Language (HTML), and Cascading Style Sheets (CSS) were used to construct these layers. Python is one of the most widely used programming languages and is the preferred language for creating connections with embedded systems, processing data,

creating graphs, and running servers, due to their high productivity and huge library support. Python was used to create the frontend server, the user interface that connects with the backend, graphs, tables, and data interpretation. The server, on the other hand, is built using web app languages such as JS, HTML, and CSS. The Middle-end has three main purposes: act as the frontend data source, handle the requests to the backend with a small UI and as a data analyser, where the errors measures are made. The frontend is in charge of creating the system's display. Figure 5.16 depicts the flow of what was done at a higher level.

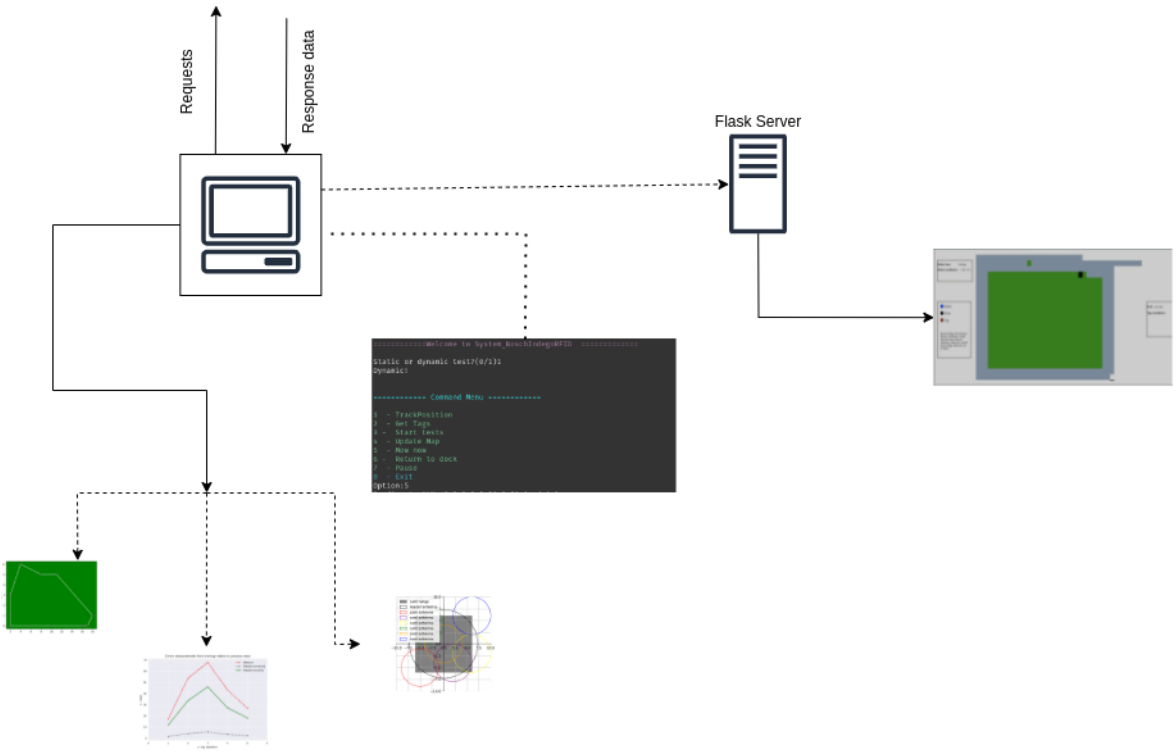

Figure 5.14: Frontend and middle-end components.

5.3.1 Requests (UI)

This part of the middle-end were created to make the possibility to manage the system requests with a small UI. The user is initially asked what type of tests want to be tested: either dynamic or static (explained in chapter 6). After that, the user can make several requests to the system implemented in the backend. Each request is sent with a proper command ID that is, then, interpreted in the lower level.

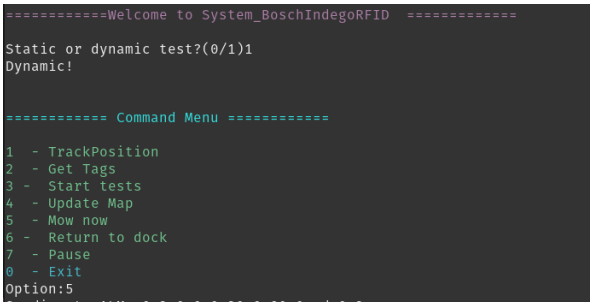

Figure 5.15: Middle-end user interface.

Requests are described as follows:

- **Track Position**: this request requires the server in frontend to be running. Basically it is the request that tracks the ALM coordinates and redirects them to frontend.

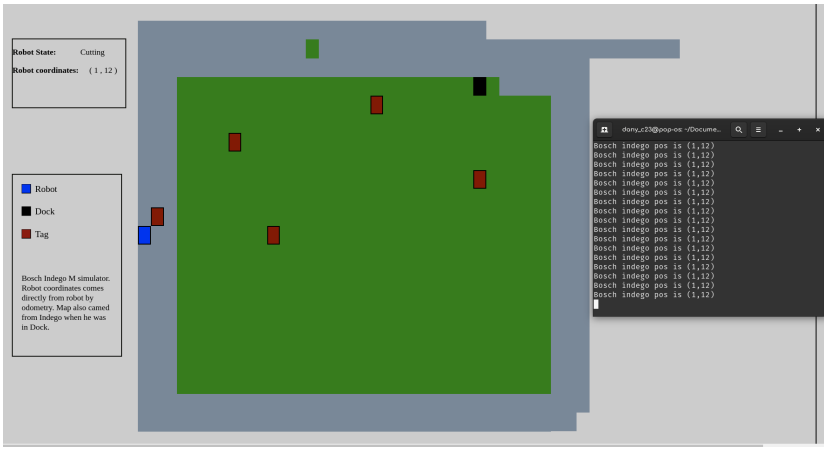

Figure 5.16: Tracking position (middle-end to frontend).

- **Get Tags**: asks backend the tags that have already been read, decodes them and stores them in a JSON file.

TagID	First detected location (x,y)	0 (x,y)	1 (x,y)
3D0081A1BDA0	[1, 2]	[1, 1]	None
3C009EBD0619	[28, 1]	[28, 2]	None
36001CF931E2	[28, 7]	[25, 7]	None
3000C4B2E1A7	[46, 20]	[22, 4]	None
3F007F2E254B	[18, 19]	[18, 18]	[17, 21]
3700304B90DC	[14, 4]	[12, 3]	None
3B006BF3882B	[13, 13]	None	None
2C00EB4AF974	[3, 9]	None	None
34006C502820	[5, 9]	None	None
34006C0AAAF8	[1, 14]	[2, 13]	[1, 17]
2C00EA902076	[22, 19]	None	None
3C009EEFAFE2	[13, 15]	None	None
3F007FF28735	[13, 3]	None	None
3D0082C68AF3	[13, 3]	None	None
3C0039DCA37A	[12, 3]	None	None
2E006FC1F575	[2, 4]	None	None
0F000DEDAD42	[2, 21]	None	None
38006688EA3C	[21, 1]	None	None
3F007C660326	[20, 3]	None	None
3C009F0B1DB5	[31, 13]	None	None
3C009EDA0179	[28, 13]	None	None

Table 5.1: Collected tags in "Get Tags" request.

- **Update Map**: as the name indicates, updates the map with sequential requests made to ALM following the messages protocols of Bosch ALM and then decoded with the same procedure. The first image of the diagram presented in figure 5.22 demonstrates the response format data from ALM. This request can only be made when the mower is docked, otherwise it will throw an exception.

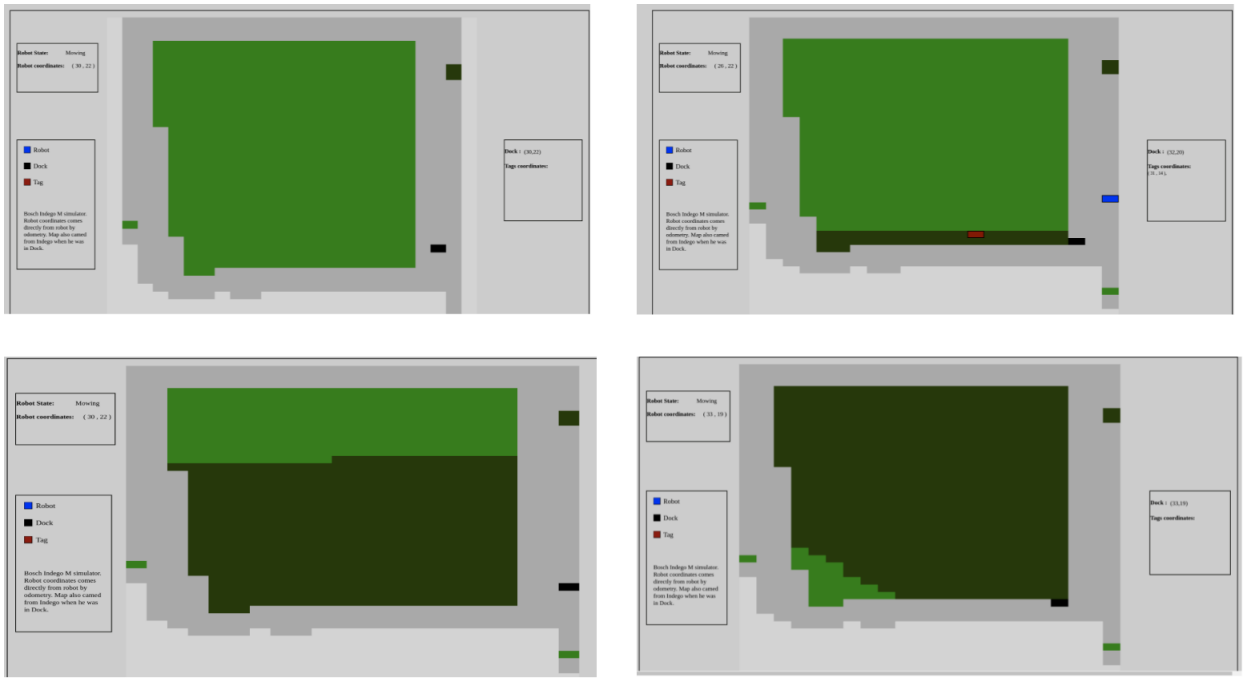

Figure 5.17: Map updating examples, frontend illustration.

- **Mow Now**: demands ALM to start mowing.

Figure 5.18: Mow now request.

- **Return to dock**: demands ALM to return to dock.

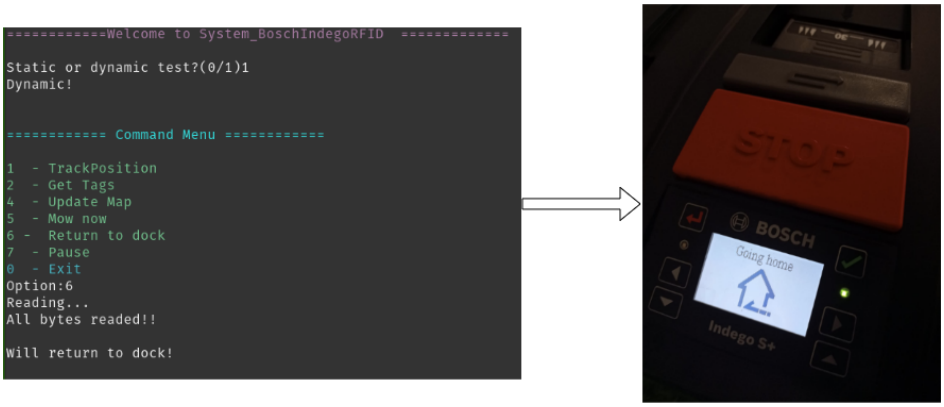

Figure 5.19: Return to dock request.

- **Pause**: demands ALM to pause

Figure 5.20: Pause request.

- **Send tag**: only used in static tests. In this case, tags were stored in a file that was sent line by line to the backend, where each line has the information about the tag and their respective coordinate, as can be seen in the following figure:

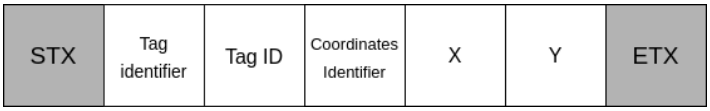

Figure 5.21: Static tags message protocol.

5.3.2 Frontend Data Source

As explained before, this part is responsible to serve the frontend as their font source. To make possible the display of the robot behavior in real-time, three main things are necessary: **grid map**, **robot's pose** and **tags locations**. Without this information, the frontend will just be a static web page with the map representation without real-time objects.

- **grid map**: to facilitate interpretation, the map is represented by a grid, with each cell representing one of three sorts of states: **inside**, **outside**, **border**, and **mowed**. Following the request to the backend, the contents of the map must be interpreted and decoded in accordance with Bosch message protocol. In the end, the map is represented as a JSON file properly decoded, along with additional map information (made with the help of other requests). This process is represented in figure 5.22.

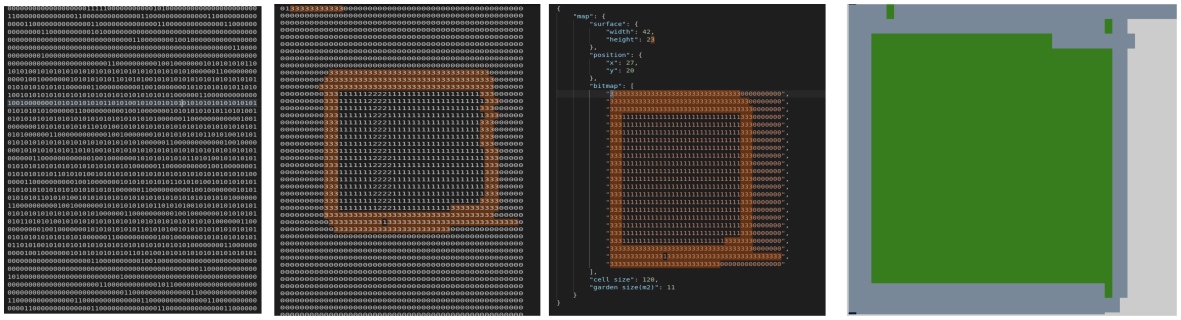

Figure 5.22: Map conversion phases.

- **ALM coordinates**: this function is responsible for constantly supplying the robot's coordinates so that a real-time display of its behavior can be provided.
- **tags**: similar to the last one, this feature is in charge of providing the tag coordinates. When a tag is detected in the lower layer, it is instantly transmitted to this layer, which is then transmitted to the frontend, where it is displayed.

5.3.3 Results analyzer

This sub-layer is responsible for analysing the results obtained in the lower layer. The comparison function compares the locations of tags that have been read several times with different positions. When a tag is discovered, its position is saved in the previously described database. When the tags are requested and decoded correctly, they are saved in a JSON file in the format given in the table 5.1. The discrepancy between the coordinates read in the same tag is then analysed. The following errors are obtained by comparing each vector with the same ID:

- **distance between the two points**:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{5.7}$$

- **coordinates difference in axis**:

$$diff_x = x_1 - x_2 \tag{5.8}$$

$$diff_y = y_1 - y_2 \tag{5.9}$$

Errors are calculated automatically whenever the detected tags are collected and saved in a JSON file. These errors are determined either relative to the first read or with the previous read. In the end, graphs are determined with a tool called **matplotlib** [33], allowing the results to be properly evaluated in chapter 6. All the tables that are in this document referred to results are also created in this sub-layer with the help of a tool called **latextable** [34].

5.3.4 Frontend emulator

To display the complete system, a web app was built using **Flask** and served via a Flask server. Flask is a Python-based web development framework. All python files run on the web host's server [35]. Flask is a highly adaptable tool that is suited for usage in

small applications. Moreover, flask provides a straightforward method to serve static files, making it simple to build a web app. This is accomplished through the use of a library that allows for the return of HTML pages with JS and CSS. Flask uses **routes** that act as the application endpoints. Flask can communicate, receiving data to their endpoints following Hypertext Transfer Protocol (HTTP) protocols, which are the most commonly used type of communication in World Wide Web (WWW). In this application, on each endpoint it is possible to make HTTP POSTs and HTTP GETs. POSTS are made by the middle-end and then consumed through GETs in the web page(JS layer).

- **HTML**: is in charge of the overall structure of the page. Initializing the complete structure of the application's components with their corresponding identifiers.
- **CSS**: define the style of the HTML components. This system is responsible for the overall look of the page, defining the text style, the colors of each map cell, the tags, and the robot.
- **JS**: is in control of the page's whole dynamic. Initially, all map attributes are consumed, specifying the color of each cell, and then the robot's location and the position of the tags are constantly updated as they are detected. With the continuous consumption (HTTP) of the implemented endpoints, it is possible to view the robot in motion as well as the representation of a new tag when is recognized.

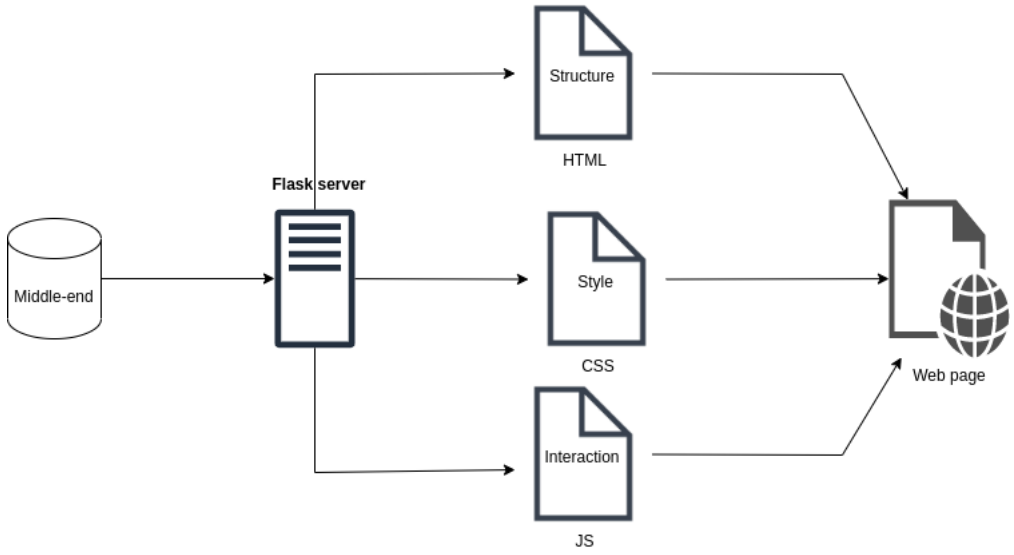

Figure 5.23: Frontend structure.

This server only operates on localhost, thus it can only be accessed by devices connected to the same network. This web app is designed with characteristics that allow it to be seen correctly even on a smartphone, as seen by figure 5.24, which shows a screenshot of the web app on a smartphone.

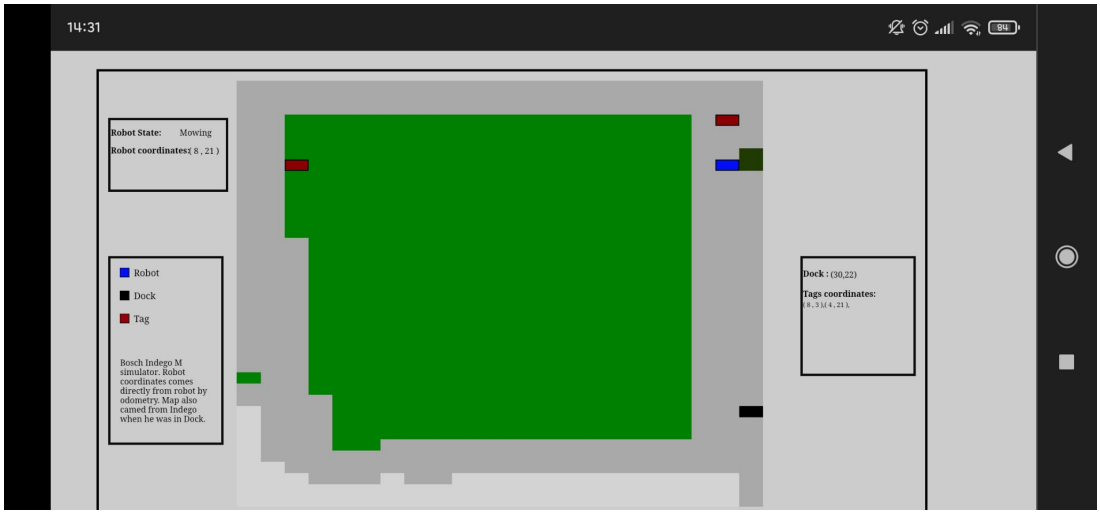

Figure 5.24: Emulator - mobile screenshot.

5.4 Programming Environment and Software Tools - review

This dissertation was built in Pop Os, a Linux operating system based on Ubuntu. Visual Studio Code Integrated Development Environment was used to write all the code. PlatformIO was used to create the backend in C++. Platform IO is an embedded system platform utility. The middle-end, as well as the frontend server, were built entirely in Python. The frontend app in the simulator was created using CSS, JS, and HTTP, as mentioned before.

Language	Used in	Characteristics	Function
Python	middle-end (fully) frontend (only server)	- interpreted, object-oriented, high-level programming language. - indicated for rapid application development - suitable language to connect existing components together - high productivity	- Decode data from backend - Convert data to JSON - HTTP requests to frontend - Requests for backend - Make graphs and tables (matplotlib) - Provides the frontend server, following flask protocol - Creates socket channels to communicate - Debug backend
C++	backend	- object-oriented computer language as part of C evolution family - high-level language that can perform as low-level language, since it is close to C. - used with platformIO	- Handles the requests and messages protocols to ALM - Tags detection - Uses multi-thread for rfid, ALM and request tasks - Multi thread synchronization - Communicate to middle-end via sockets - Stores tags in structures
HTML	frontend	- a web-standard markup language for displaying documents on a web browser - structure of a web browser	- web app structure
CSS	frontend	- programming language used to describe the presentation of a webpage - designed to enable the separation of presentation and body	- Grid map definition - Fonts, size, and color implementation - Tags, robot, dock design (color and shape) - Map cell separation (inside, outside, border, mown)
JS	frontend	- high-level programming language - used to make HTML pages better - do not need to be compiled - renders pages in interactive and dynamic way - responsible for the web page behavior - allow pages to react to data	- controls robot behavior - manages new tags added - changes position coordinates - keeps map updated

Table 5.2: Tools used in this system.

With this implementation, it is possible to observe the robot's behavior in real time in order to offer a perspective of what is happening. Furthermore, with the coordinates that are acquired by the tags, it is possible to carry out various tests through the robot that can be analysed either as proof of new reference points or as proof that the current coordinate system is not perfect. It is also important to emphasize that all the data that were tested and analyzed comes directly from the robot, which proves their viability as future-proof of implementation in the robot. The next chapter details and evaluates all the tests conducted as part of this implementation, as well as tests relating to the RFID system's reading efficiency.

CHAPTER 6

Results and analyzes

6.1 Introduction

Once the system was deployed, a number of tests directly related to the reading of tags and their position were conducted to validate the core concept of this dissertation. The purpose of these tests, and since the system was not implemented into the robot layer, is to prove that by placing the tags on the lawn it would be possible to compare the robot's position with the position of the tags. With this, it is expected to obtain different results between the robot and the tag position so that both coordinates can be compared, and the error related to these measurements can be calculated. This serves to prove that it would be possible to recalibrate the robot's positioning whenever it passes a tag. For this, two types of main tests were performed: **dynamic** and **static** tags distribution. In the first, the coordinates of the tags are assigned by the robot the first time it passes through a tag. In the second test, a tag is placed on the lawn with a defined position.

Before that, several laboratory studies were also performed in order to determine the behaviour of the tag antenna in relation to the reader's antenna. Furthermore, different experiments were carried out to illustrate the irregularity of the robot's movement, which affects tag reading, and what was predicted in the laboratory differed from what was really achieved. The findings of each test, as well as a more detailed description of each, are presented in the following sections.

6.2 Laboratory tests

As previously stated, most undesired elements that may obstruct a robot's ability to do its tasks, referred to as "noise", are removed in simulation scenarios. As a result, before advancing to real-time testing with the robot, it was required to outline and specify the capacity of the antenna, with a focus on its range in terms of both height and width. To accomplish that, two types of laboratory experiments were performed using the reader integrated into the robot to determine the range of the antenna while detecting the tags. These measurements

were performed with a handcrafted tool and both tool error and human error were discarded. Before detailing the results obtained and the methods achieved, it is necessary to emphasize that all tests were performed with the following two types of tags:

Figure 6.1: EM4100 RFID tags, card and keychain.

The first type of tag (card) has a larger antenna (diameter 4 cm), and as described in chapter 3, it is expected to have longer range. The other sort of tag employed (keychain) has a smaller antenna with a diameter of 2cm, hence its reading range will be smaller in comparison to the other tag, like will be proved in the following subsections.

6.2.1 Height

The readability was tested at the height level from cm to cm, with the tag antenna aligned with the centre of the reader antenna, until the tag was no longer detected. Human error is ruled out once more. The acquired findings are shown in the tables below:

Height - range (cm)	time(ms)	Quantity read
0	14	1
1	14	1
2	14	1
3	14	∞
4	14	∞
5	14	∞
6	X	X

Table 6.1: Height range (card).

Height - range (cm)	time(ms)	Quantity read
0	14	1
1	14	1
2	14	1
3	14	∞
4	14	∞
5	X	X

Table 6.2: Height range (keychain).

The reading duration and number of times the tag was read were recorded in these measures. The reading time was measured from the detection of the first byte to the detection of the last byte of the tag. Time has no effect on the outcomes achieved in this situation. The card-type tag could only read up to 5 cm in height, and the keychain tag up to 4 cm. The height and the communication demonstration are shown in figure 6.2. The number of readings is determined by the reader's software; as it approaches the range limit, it detects the same tag as if it were a new electromagnetic field, thus the tag is continually recognized as a new reading.

Figure 6.2: Reader antenna placement and tag communication in the lawn.

6.2.2 Width

In this experiment, the range was measured at the width level. To do that, in both cases, the tag was set to the same height (3 cm) and measured from the reader's antenna centre to the limit. The acquired findings are shown in the tables and graphs below. The wider circle indicates the reader antenna (3.75 cm radius), while the centre square represents a suitable tag placement for proper reading. This is intended to represent that the tag is read if the centre of the antenna is within the square referred.

Width - range (cm)	time(ms)	Quantity read
0	14	1
1	14	1
2	14	1
3	14	∞
4	14	∞
5	14	∞
6	14	∞

Table 6.3: Width range (card).

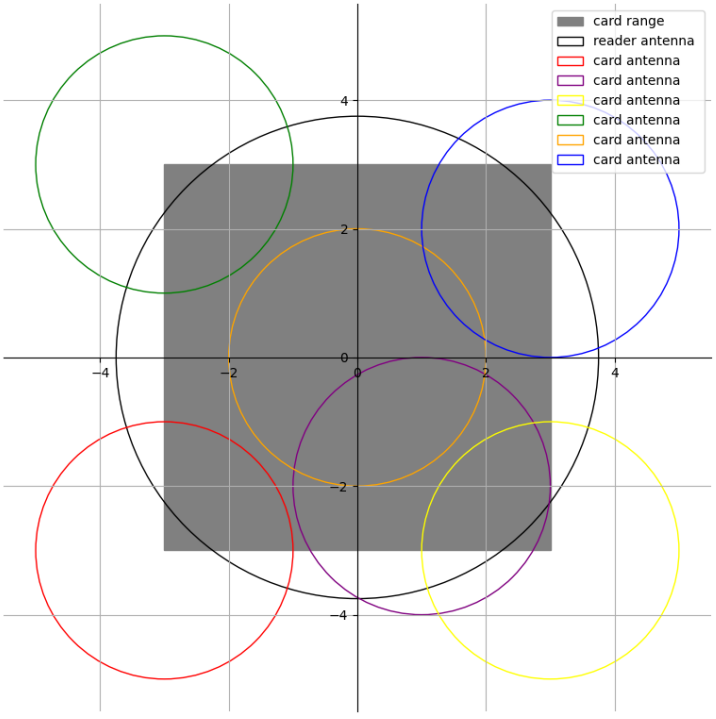

Figure 6.3: Width range graph (card).

.

Width - range (cm)	time(ms)	Quantity read
0	14	1
1	14	1
2	14	1
3	14	∞
4	14	∞

Table 6.4: Width range (keychain).

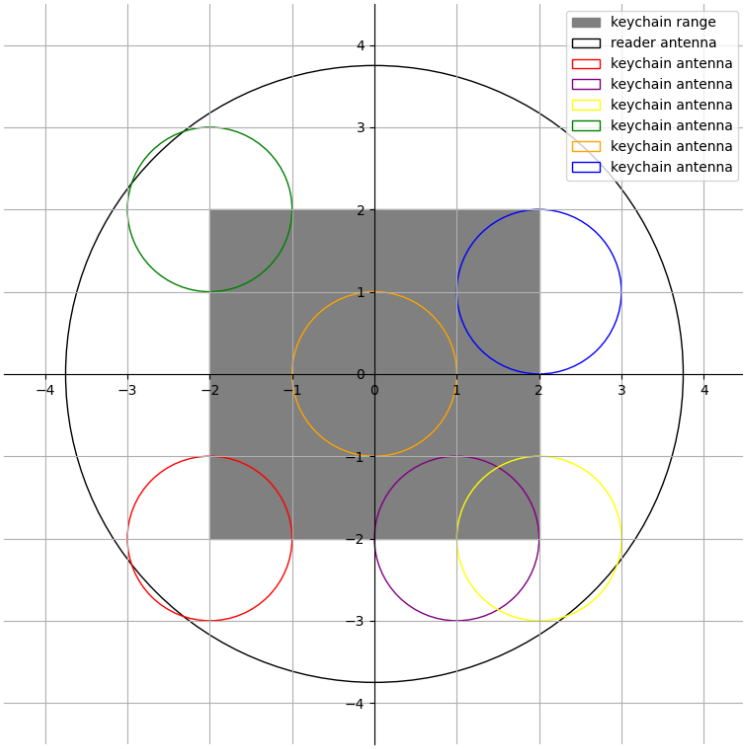

Figure 6.4: Width range graph (keychain).

By comparing the two experiments, it is clear that the smaller the antenna, the shorter the range, as stated in chapter 3, since the card read range is higher than the keychain read range.

6.2.3 Hall of tags

Before beginning to carry out the tests directly connected to the robot's position, with the robot in the ground, another type of test was carried out to determine the effectiveness of reading tags with the robot in motion. To achieve this purpose, a 3 cm (obtained in the previous test) corridor was built, in which the robot was first positioned in line with the corridor's center and then began moving forward. Each tag was put at a distance multiple of 12 cm since the garden cell size of the map on which the tests were made is 12 cm. The hall was made up of either keychain tags or card tags, as seen in the following two pictures.

Figure 6.5: Hall of tags setup.

This sort of test was designed to illustrate what was previously described in the initial chapters about the robot's shifting mobility. Even if there are no "bumps" and in a fully flat environment with no controversies, it would be assumed that the robot could read all the tags set in the corridor without problems. This was demonstrated by the following tables, where **N** represents the repetition of the test, and **velocity** is represented in rpm/s.

Tag number	time(ms)	Quantity read(x)	Probability
1	14	3	150
2	14	2	100
3	14	2	100
4	14	3	150
5	14	3	150
Average	14	2.6	130

Table 6.5: Hall of tags (card), velocity=10 rpm/s, N=2.

Tag number	time(ms)	Quantity read(x)	Probability(%)
1	14	2	100
2	14	2	100
3	14	1	50
4	14	2	100
5	14	1	50
Average	14	1.6	80

Table 6.6: Hall of tags (keychain), velocity=10 rpm/s, N=2.

In the first table and with a low speed, there were situations in which a tag was read more than once, the result obtained with this test was even better than predicted, presenting an average of 130% while the expected result was 100%. 30% of the time the tags were read

more than once, this is due to the fact that the robot's speed is relatively low. As expected, the smaller tags were not always read correctly even at a low speed, with an average of 80% instead of 100%, indicating that 20% of the time the tags were not read. This occurs because even on a flat surface, the robot's behaviour is never straight, which affects tag detection, particularly in this type of tag where the antenna is smaller and the robot's deviation can cause the tag's antenna to not be detected.

Tag number	time(ms)	Quantity read(x)	Probability(%)
1	14	10	100
2	14	8	80
3	14	7	70
4	14	8	80
5	14	6	60
Average	14	7.8	78

Table 6.7: Hall of tags(card), velocity=40 rpm/s, N=10.

Tag number	time(ms)	Quantity read	Probability(%)
1	14	6	50
2	14	5	50
3	14	4	40
4	14	3	30
5	14	4	40
Average	14	4.2	42

Table 6.8: Hall of tags (keychain), velocity=40rpm/s, N=10.

With the robot's usual pace, the same tests were performed 10 times for each tag hall. In comparison to the previous findings, these were drastically different. This illustrates how tag reading is affected by speed. The tags were read 78 % of the time with card tags, which means that 22 % of the time, even with a bigger antenna, the tags were not read. When it came to keychain tags (table 6.8), the tags were only read 42% of the time. These results show that the tags readily locate themselves beyond the square outlined in the figure 6.3 and figure 6.4 graphs in both, the first and second cases.

After testing the antenna accuracy, and to prove the initial concept of this dissertation, two other types of tests were carried out on a synthetic turf. In the first of these two tests, tags coordinate are assigned manually, while in the second, the mower is the one who assigns a coordinate to a tag. The assignment is only performed in the mower database in both circumstances. These two types of tests have two basic objectives. The first is to look for an inaccuracy in the mower coordinates by having the mower pass by a tag several times and comparing the coordinates each time, simulating a calibration of its location as it passes over a tag. The second purpose, as stated in the first chapter, was to demonstrate how these tags

may assist the mower in locating itself if it becomes lost, reducing the need to return to the dock and therefore enhancing efficiency.

6.3 Static reference points

Static reference points refer to distributing tags statically, with their location previously known before the robot starts to cut the yard. Tags coordinates do not have any relation to the robot position, they were manually placed and measured at a distance of 12 cm to provide a scale that was comparable to the map where the test was conducted. After completing a scale translation, the test was started so that the coordinates could be compared properly. Figure 6.6 depicts the tag distribution grid, with a total of 54 tags used (keychain tags are grouped with two tags due to reading restrictions).

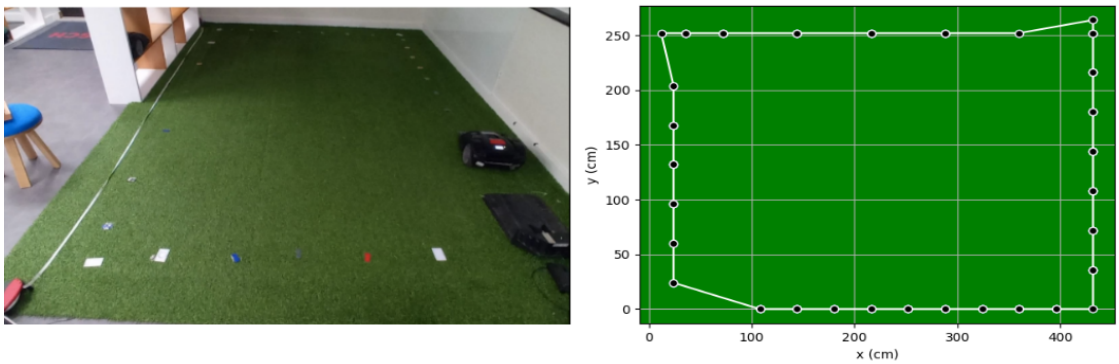

Figure 6.6: Static tags distribution.

Mower always starts with the border cut method, where most of the tags will be detected since they are placed in the perimeter wire. After the border cut, the mower starts to "trim" the rest of the yard following the Logic Cut algorithm. When the mower passes through a tag, the tag is stored in the array of the already existed tag. The results of each test done with this method are listed in the table above. Each set of tests corresponds to a complete lawn cut. To determine, the error tags were compared relative to the first read tag and to the previous one:

- **Distance (Or/Pr)**: coordinate's distance between the first tag (Or) registered or the previous tag (Pr) registered, using equation 5.7. This measure is represented in cell size, which in this case is 12 cm with an uncertainty of ±6 cm.
- **Samples**: number of samples where the same tag as more than one registered value. One tag, read twice in a test, represents one sample. It is seen as the number of times that the mower can calibrate its position.
- **New tags detected**: number of new tags detected in each round.

Statistics	Distance (Or) (cell size)	Distance(Pr) (cell size)	Samples	New tags detected (max: 54)
Test 1	5.72	5.28	17	15
Test 2	5.67	5.65	27	16
Test 3	3.93	3.47	25	7
Test 4	3.82	3.39	17	2
Test 5	3.77	3.10	41	5
Test 6	3.63	2.81	61	3
Test 7	3.55	2.75	20	1
Test 8	3.45	2.49	42	0
Test 9	3.35	2.31	61	1
Average/total	3.83/-	3.47/-	34.56 / 311	5.56/50

Table 6.9: Tags in perimeter wire with static distribution.

These experiment shows that:

- even with the tags distributed in the same position, the mower does not pass always through the same tags, detecting different tags with different coordinates in each test. This demonstrates that the mower's movement is inconsistent.
- tags have a larger distance when compared to the first read than with the previous read because the accumulation of error is bigger.
- the number of tags reads on each cut is considerably low. In this case, the total number of tags reading in the 9 tests was 50 which means that only in the ninth cut the mower read more than 90% of the tags.
- the number of samples in each test is low, that's why this test was carried out 9 times. This influences the final comparison in both aspects, either negatively or positively. For example, if a test has only one sample the mower can have the same coordinates as the tag, meaning that the average distance of this test will be null. However, this does not mean that the mower doesn't have any error in all the cuts. The same can happen on the opposite side.
- the number of samples can demonstrate that the mower coordinates could be calibrated 34.56 times in each cut.

These stats serve only to prove that the current navigation algorithm as an error that can be reduced with the aid of the tags. The distance is measured in cell size, which in this case is 12 cm and, looking at the table, even in the worst situation (5.72) the robot will have a distance error of 68.64 ± 6 cm. In reality, even in that case it is not visible a big deviation since the robot size is 40 cm and the coordinates are always given based on the mower's centre. This can be seen in the next figure:

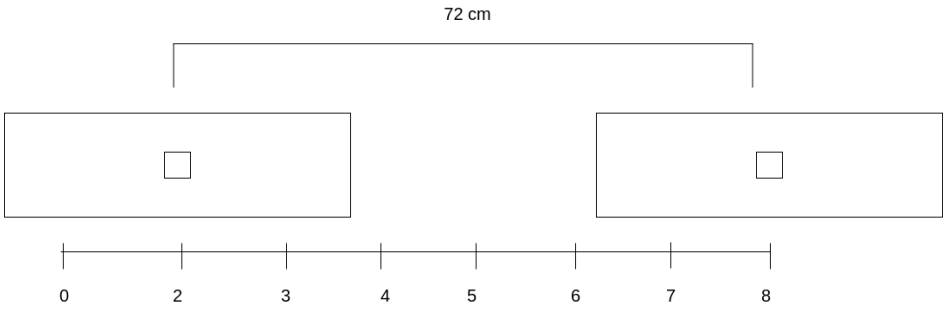

Figure 6.7: Distance error of 72 ±6 cm demonstration.

In this experiment, tags were distributed only in the wire due to measures facilities. As for the static distribution, it serves to simulate a possible case of a user measuring the distance between each tag. Since this is not user-friendly, a possible solution is to put marks, with a predefined distance, in the perimeter wire landmarks must be placed (assuming tags are attached to them). The next section will demonstrate the same sort of test but with the dynamic approach, where tags are distributed either in the perimeter wire or across the yard. In addition to proving the same purpose as this method, the dynamic technique will replicate the reality that tags can help the robot a lot when it is lost, serving as new reference points.

6.3.1 Dock identification

Another early concept was to demonstrate the versatility of this technology by using tags to designate cut zones, objects, or the dock. When Indego is mapping, it always requests the user's approval when it finds the dock. This guarantees that the "object" that the robot hits is, in fact, the dock. This system proves that the dock can be identified with tags, by placing them around the dock and introducing them into the system with their corresponding IDs. This avoids user involvement. A possible instance of this procedure can be seen in figure 6.8, which demonstrates the current process and a possible implementation with the demonstration in the emulator.

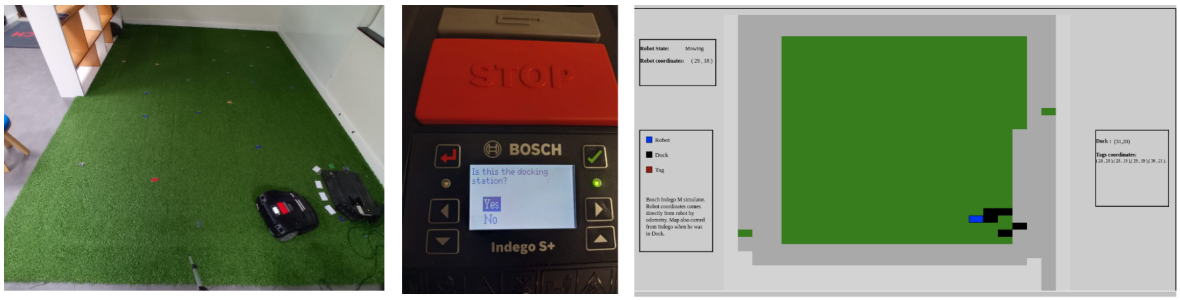

Figure 6.8: Dock identification with tags.

This situation would imitate a conceivable implementation in which the tags are also placed on the landmarks, or in a similar method in which the robot already knows the IDs of the corresponding tags. This will be quite useful because the robot will only need to know the ID of these tags since their coordinates are irrelevant. However, it would be necessary to guarantee that these tags are placed around the dock and not elsewhere.

At last, the robot's behavior was tested using the dynamic method, in which the tags are set on the lawn and the robot is the one that assigns the coordinates to the tags. As previously said, the goal would be to simply give a coordinate to the tags the first time it passes through them. To replicate this behavior, the first time the robot passes by a tag, coordinates are recorded differently, while the subsequent ones are put in an array of coordinates and compared at the end, similarly to the static test. This method was tested with the tags distributed in two different ways: the first one tags are placed on the top of the perimeter wire, the second one tags are distributed randomly across the yard.

6.4.1 Distribution in the perimeter wire

This test was carried out to validate the basic notion of placing the tags on the perimeter wire as if they were on the landmarks, just like in the static experiment. Placing the tags around the perimeter wire simplifies the detection procedure since it is simpler to get them into the range of the reader antenna because the robot's movement is more predictable and consistent since it is following the perimeter wire. Furthermore, the first time the robot goes through the tags, the cumulative error is considerably lower since the robot is guided by the wire, which makes the tag positions more precise. To simulate this, the **border cut**, in which the robot simply trims the grass along the wire, was used. However, in the future, the idea is to use this while the mower is mapping.

6.4.1.1 Map in tinny area

This test was firstly done on the smallest field, which allowed the creation of a map by merging the coordinate points provided by the tags, as can be seen in the following figure:

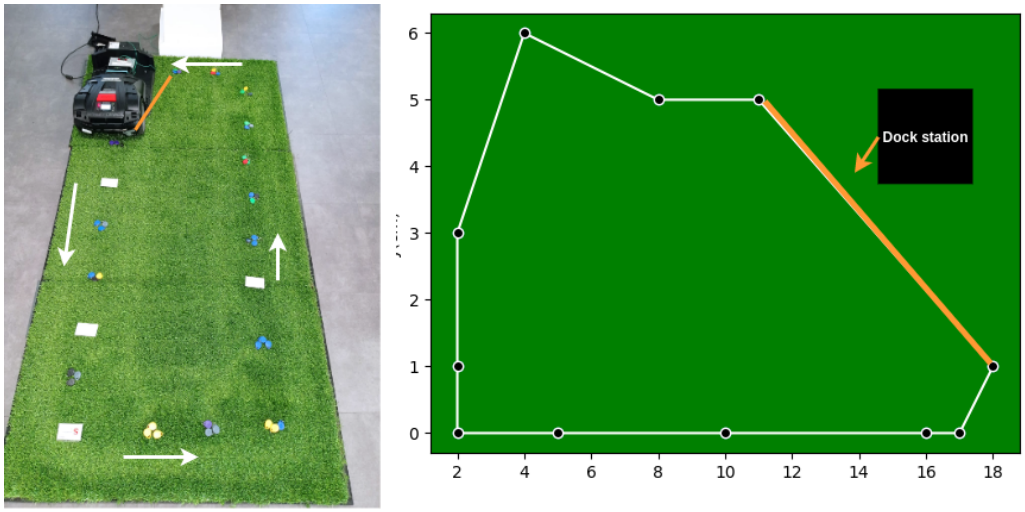

Figure 6.9: Tags in border cut demonstration (map 3m^2), scale = 1:12 cm.

As can be seen in the figure above, the implemented arrangement made use of all available tags. However, to improve the reading efficiency of the keychain tags, they were arranged in groups of three instead of one, as demonstrated earlier, because the reading efficiency of this

tag is smaller. In the end, with the union of the collected coordinate points, a representation of the map can be seen.

6.4.1.2 Map in larger area

As in the previous case, this one was only registered when the robot was making the border cut. In fig 6.10 it is possible to observe the mower behavior in the real world, and the combination of the coordinates obtained from the tags. This illustrates that by placing tags on top of the perimeter wire, new points of reference for the robot can be provided.

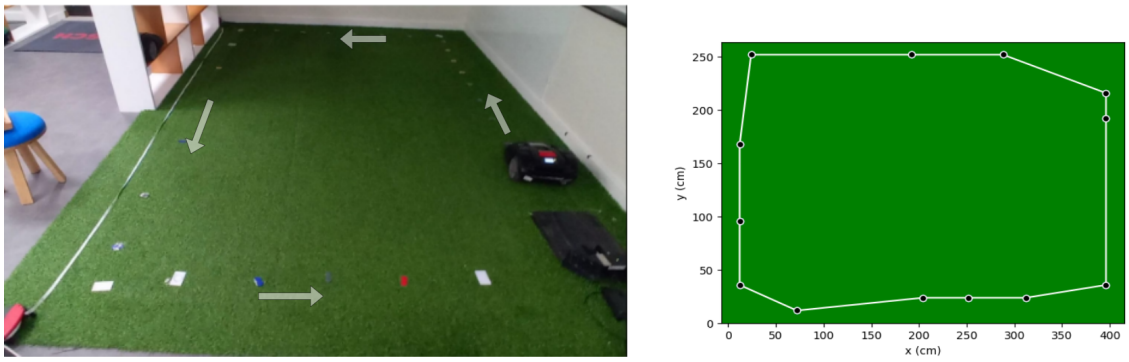

Figure 6.10: Tags detected with border cut (map 11m^2), scale 1:1 cm.

Using these extra reference locations, it is possible to reduce relocation time by having the mower move to one of these tags rather than returning to the dock when it is lost, proving that this approach can meet the other key purpose. This happens because, when the robot becomes disoriented, it searches for the perimeter wire so that it could return to the dock and recalibrate itself. After that, the mower returns to the same location and resumes its regular operation. To show this, the mower position was manually changed, for example, to simulate a large slip or a large bump, and the path and time were recorded. When the robot gets lost, as seen in figure 6.11, the path to find the wire and return to the dock can be fairly long, requiring a significant amount of time to return to the dock station. In this experiment, it took the mower 1:36 min to return to the dock and go back where it was. With tags in place, this could be avoided since the process will remain the same, but instead of returning to the dock, it will return to the nearest tag in the wire and resume its activity from there. As for the orientation, it is not a problem either, because the mower always follows the wire back to the dock in a clockwise manner.

(1) (2)

Figure 6.11: Indego finding position.

This experiment was done using the exactly same setup as in figure 6.6 using the same analyses and procedures, the only difference is that tags coordinates are allocated dynamically instead of being static, resulting in a table with the same stats recorded.

Statistics	Distance (Or) (cell size)	Distance (Pr) (cell size)	Samples	New tags detected (max: 54)
Test 1	2.0	2.0	1	14
Test 2	3.59	2.84	7	13
Test 3	2.67	2.69	20	3
Test 4	3.20	3.35	23	11
Test 5	2.96	2.65	41	2
Test 6	2.87	2.35	46	3
Test 7	2.78	2.33	21	0
Test 8	2.65	2.33	23	0
Test 9	2.32	2.28	60	1
Average/total	3.21 / -	2.66 / -	26.8 / 242	3.78 / 46

Table 6.10: Tags in perimeter wire with dynamic distribution.

In comparison to the static test, a smaller number of samples with the exact same distribution were obtained, as expected, because it is the robot that assigns the coordinates to the tags, implying that there are only samples from the moment it passes through the same tag for the second time, as opposed to the static method, which counts the first time it passes through a tag as a sample. In this experiment, the mower could calibrate its position 26.8 times in each cut. In addition, the same points mentioned in the static method were concluded, both at the level of detected tags (relatively lower in this case) and the distance of the tags compared to the initial tag and the previous tag. Furthermore, as expected, the accumulated error is significantly smaller, since the robot itself assigns the coordinates to the tags, with a lower accumulated error in the detected tags. The discrepancy between the different coordinates obtained in the same tag can be seen in the following image.

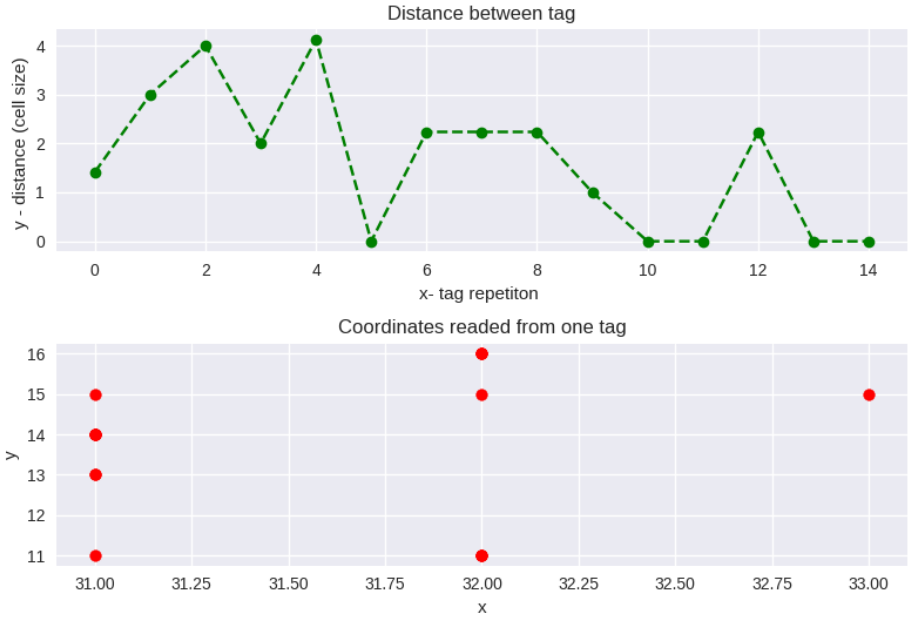

Figure 6.12: Example of coordinate discrepancy for the same tag.

As can be seen in the image, and taking the example in figure 6.7 into account, the difference between the acquired coordinates is not big enough to have a significant impact on the robot's navigation efficiency. Thus, it is possible to admit that this system could help the robot in its navigation system, reducing this distance through a hypothetical calibration of coordinates when it passes through a tag.

6.4.2 Random distribution

Finally, in the final test and using the same approach, a random distribution of tags was tested over the yard. This experiment was carried out to demonstrate that it is difficult to locate tags dispersed across the yard, as well as that the error between tags is bigger. Figure 6.13 depicts the setup used for this test.

Figure 6.13: Random tags setup.

Statistics	Distance (Or) (cell size)	Distance (Pr) (cell size)	Samples	New tags detected (max: 54)
Test 1	5.02	5.02	4	13
Test 2	6.06	6.17	3	5
Test 3	4.48	5.15	12	4
Test 4	3.80	3.94	22	4
Test 5	3.50	2.98	18	0
Test 6	3.49	2.80	26	2
Test 7	3.37	2.72	11	1
Test 8	3.33	2.60	10	1
Test 9	3.23	2.43	20	4
Average/total	4.03 / -	3.75 / -	14 / 126	3.8 / 34

Table 6.11: Tags distributed randomly with dynamic method.

First, with the tags randomly dispersed across the lawn, the robot may already have a significant error when it makes the first assignment to the tags, which means that, unlike the distribution in the perimeter wire, the coordinates of the initial tags are much less absolute. This phenomenon leads to a low detection of tags, as can be seen by the number of tags read and samples, with the exact same amount of tags as the previous experiments. Oppositely to the distribution of tags in the perimeter wire, which reached around 90% of all the tags in the ninth test, in this experiment the maximum achieved was only 63% of the same amount of tags. This happens because there is no correlation between robot movement and tag positioning, which does not happen when it is distributed in the perimeter wire. As for the distance between tags, the total amount of samples leads to more inconclusive results, however, it is possible to see that with tags distributed randomly the distance average is bigger than when they are in the perimeter wire, like expected.

CHAPTER 7

Conclusions and future work

7.1 Conclusions

The main focus of this dissertation was to explore the integration of a RFID-based system that could aid the autonomous lawnmower's navigation and location mechanism. Despite the implementation in the robot layer has not been explored, it was feasible to construct a system that demonstrates that RFID technology can be used to assist in the navigation system of an autonomous lawnmower, in this case, Bosch Indego. This was demonstrated by calculating the odometry error in a specific scenario of a 11m^2 lawn, illustrating the distance error that can be minimized by integrating this system. Furthermore, the efficiency that this technology can provide when the robot is lost was demonstrated, which can result in a large reduction of time in the robot's relocation procedure. Still, the versatility of this technology was proved with the dock identification in terms of detecting objects or specific portions of the lawn, such as an area where the robot failed to mow or even the identification of a certain sort of object. Therefore, a three-layer system with the following features was created:

- creation of a system capable of detecting RFID tags and associating them with a position on the map, either statically or dynamically.
- it was completed the integration of the system in the robot with the reader antenna at the bottom, which led to a development of an algorithm that computed the variance between the robot's centre and the true position of the tag, given the offset. This was done since direct integration in the robot's centre would interfere with the cutting blades' alignment, making it far more difficult.
- development of a simple interface capable of communicating with the robot through TCP connection, allowing it to execute simple activities like mowing the lawn, returning to the dock, and updating the map, among others.
- decoding and graphical representation of the map where the robot will cut the lawn.
- construction of a web app that shows the robot's behaviour as well as the detection of tags on a map in real-time.

However, this system has some drawbacks and limitations:

- because of reader-level limitations and the lack of a clear relationship between tag positioning and robot movement, identification of tags on the pitch is not 100% assured, especially when tags are randomly distributed around the grass.
- the tag offset was calculated using relative measurements between the antenna and the robot's "centre", thus it is not precise. This has an impact on the tag positioning process and the calibration algorithm.
- the cumulative inaccuracies in the tag calibration algorithm and data obtained from odometry constrain the veracity of the final comparison between the robot's position and the real position of the tag.

As for the type of implementation and the strategy applied for the purpose, it is concluded that:

- distribution of static tags gives better viability and efficiency since tag placement is not reliant on the robot but rather on the static method used to place the tags. This method, however, is not practical from the user's standpoint because the user would have to take measurements along the ground and enter them into a database (possibly included in the existing application) which makes this not user-friendly. Even with the proposed idea, with marks on the wire, indicating where the landmarks with the tag attached should be placed, nothing guarantees that the user actually put the tags in that mark. Besides that, in this method, it should be created another mechanism that indicates in a database the coordinates of each tag presented in the yard. With the dynamic method there is no user intervention, and no additional method is required to discover the tags coordinates since it is the mower that assigns them.
- in an early stage, the best strategy to use with this implementation would be to distribute tags exclusively over the perimeter wire, since the main goal would be to use this strategy when the robot is conducting the recognition of the area, where it would follow the perimeter wire exclusively, ensuring that the majority of tags are read.

7.2 Future work

The test scenarios used in this experiment were made in a synthetic lawn with small dimensions and a nearly flat surface. Despite being an excellent environment for early testing, this sort of grass is somewhat different from a conventional lawn, where a variety of external influences might affect the system's reading and accuracy (atmospheric conditions, bumps, slips, wheels skidding, etc). However, there are several parts of this system that could be improved before testing it in a more realistic context, such as:

- the improvement of tag reading accuracy either through the inclusion of a new antenna (preferably larger to be able to have greater range and greater reading efficiency) or through a new reader.
- the tag calibration algorithm, both in terms of offset measurement and in terms of calculating the robot's orientation. Instead of exploring a new or a better algorithm, it could be more interesting to try to incorporate the antenna in the robot's centre without

interfering with the cutting blades, allowing the tag's location to be directly linked to the robot's position.

- if the tags are placed along with the lawn (not in the wire), a correlation between their placement and the robot's movement must be explored in order to maximize the probability of the robot to read a tag.
- integration between the middle-end and front-end, providing a more extensive server with more features, allowing requests and records to be handled in a single layer.

The purpose of the comparison of the static and dynamic techniques for assigning coordinates was to determine which of the two approaches would be the most viable, both in terms of efficiency and user convenience. In the experiments, the tag coordinates were defined using the static method before the robot began cutting the grass. The dynamic strategy, on the other hand, requires the robot to recognize the tags, which means that the robot begins mowing the grass without their knowledge, with the first pass through the tags serving as the assignment of coordinates rather than the mower's calibration. The preference for the dynamic method, on the other hand, has as its central objective to make the robot identify the tags before starting to cut the grass, in the mapping phase, so that it could begin cutting with the knowledge of the tags already in its database. As a result, and with the current system's vulnerabilities addressed, the key future work would be to incorporate tag recognition in the robot mapping phase, which would already require integration in the robot layer.

References

[1] *Robot vs Machine: Difference Between A Robot and A Machine?*, en-US, Oct. 2020. [Online]. Available: https://www.robotsscience.com/resource/difference-between-a-robot-and-a-machine/ (visited on 10/14/2021).

[2] *Robot - Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Robot (visited on 10/14/2021).

[3] *A Guide to Autonomous Robots & 8 AMR Applications*, en-US, Jan. 2017. [Online]. Available: https://waypointrobotics.com/blog/what-autonomous-robots/ (visited on 10/14/2021).

[4] J. Sasiadek, Y. Lu, and V. Polotski, «Navigation of Autonomous Mobile Robots - Invited Paper», in *Lecture Notes in Control and Information Sciences*, vol. 360, Journal Abbreviation: Lecture Notes in Control and Information Sciences, Oct. 2009, pp. 187–208, ISBN: 978-1-84628-973-6. DOI: 10.1007/978-1-84628-974-3_17.

[5] S. Huang and G. Dissanayake, «Robot Localization: An Introduction», en, in *Wiley Encyclopedia of Electrical and Electronics Engineering*, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047134608X.W8318, American Cancer Society, 2016, pp. 1–10, ISBN: 978-0-471-34608-1. DOI: 10.1002/047134608X.W8318. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W8318 (visited on 08/20/2021).

[6] *Odometry*, en, Page Version ID: 1032639474, Jul. 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Odometry&oldid=1032639474 (visited on 10/06/2021).

[7] *Localization and Mapping with Autonomous Robots*. [Online]. Available: http://scholar.googleusercontent.com/scholar?q=cache:8I1brA3kONEJ:scholar.google.com/&hl=pt-PT&as_sdt=0,5&as_vis=1&scioq=Localization+and+Mapping+with+Autonomous+Robots (visited on 10/06/2021).

[8] «ARW – Lecture 01 Odometry Kinematics», en, p. 50,

[9] *SLAM for Dummies - dspace.mit.edu / slam-for-dummies-dspace-mit-edu.pdf / PDF4PRO*, en-US, Aug. 2018. [Online]. Available: https://pdf4pro.com/view/slam-for-dummies-dspace-mit-edu-4c593e.html (visited on 10/07/2021).

[10] W. Burgard, M. Bennewitz, D. Tipaldi, and L. Spinello, «Introduction to Mobile Robotics», en, p. 47,

[11] *Robotic lawnmowers | Bosch DIY*, en-GB. [Online]. Available: https://www.bosch-diy.com/gb/en/garden/robotic-lawnmowers (visited on 10/09/2021).

[12] MATTHIAS, *Are There Robot Lawn Mowers Without Perimeter Wires?*, en-US. [Online]. Available: https://robolever.com/are-there-robot-lawn-mowers-without-perimeter-wires/ (visited on 10/08/2021).

[13] *Segway's first robotic lawn mower uses GPS to stay on course*, en-US, Section: Around The Home, Sep. 2021. [Online]. Available: https://newatlas.com/around-the-home/segways-first-robotic-lawn-mower-gps/ (visited on 10/04/2021).

[14] ——, *Random vs. Systematic Robotic Mowers: Which Is Better?*, en-US. [Online]. Available: https://robolever.com/random-vs-systematic-robotic-mowers-which-is-better/ (visited on 10/05/2021).

[15] *iRobot Terra Robot Lawn Mower (First Impressions)*, en-US, Jan. 2019. [Online]. Available: https://myrobotmower.com/irobot-terra-robot-lawn-mower/ (visited on 10/08/2021).

[16] Toadi, *Meet Toadi the intelligent lawn robot*, Jun. 2020. [Online]. Available: https://www.youtube.com/watch?v=I1J7aeEh7Po (visited on 10/08/2021).

[17] *Is There A Robot Lawn Mower Without Perimeter Wire?*, en-US, May 2018. [Online]. Available: https://myrobotmower.com/is-there-a-robot-lawn-mower-without-perimeter-wire/ (visited on 10/11/2021).

[18] *A Comprehensive Guide to Asset Tracking Technologies (2021)*, en-US. [Online]. Available: https://www.wisersystems.com/blog/asset-tracking-technologies-comprehensive-guide (visited on 10/11/2021).

[19] *RFID For Dummies®*, en, ISBN: 978-0-7645-7910-3. [Online]. Available: https://learning.oreilly.com/library/view/rfid-for-dummies-r/9780764579103/ (visited on 09/13/2021).

[20] *What are RFID Tags? | UHF Tags Explained*. [Online]. Available: https://www.atlasrfidstore.com/what-are-uhf-rfid-tags/?utm_source=RFID-Beginners-Guide&utm_medium=eBook&utm_campaign=Content&utm_content=tag-guide (visited on 10/17/2021).

[21] *17 Things You Might Not Know About Gen 2 RFID Tag Memory Banks*, en. [Online]. Available: https://www.atlasrfidstore.com/rfid-insider/17-things-might-not-know-gen-2-rfid-tag-memory-banks (visited on 10/14/2021).

[22] *How does RFID work? | Aucxis*, en. [Online]. Available: https://www.aucxis.com/en/rfid/rfid-technology (visited on 10/12/2021).

[23] *What is the difference between read only and read-write RFID Tags? - everything RF*. [Online]. Available: https://www.everythingrf.com/community/what-is-the-difference-between-read-only-and-read-write-rfid-tags (visited on 10/17/2021).

[24] *What is RFID? | The Beginner's Guide to How RFID Systems Work | atlasRFIDstore*. [Online]. Available: https://www.atlasrfidstore.com/rfid-beginners-guide/ (visited on 10/12/2021).

[25] *Inductive and Backscatter Coupling - How Passive RFID works*, en-US. [Online]. Available: https://rfid4u.com/inductive-and-backscatter-coupling/ (visited on 10/13/2021).

[26] J. Miller, *RFID Coupling: The Relationship Between a Tag and a Reader*, en-us. [Online]. Available: https://www.computype.com/blog/rfid-coupling-the-relationship-between-a-tag-and-a-reader (visited on 10/13/2021).

[27] *How RFID Works and How To Make an Arduino based RFID Door Lock*, en-us, May 2017. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/rfid-works-make-arduino-based-rfid-door-lock/ (visited on 11/05/2021).

[28] *Non-volatile memory*, en, Page Version ID: 1046835537, Sep. 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Non-volatile_memory&oldid=1046835537 (visited on 10/16/2021).

[29] *What is a Microcontroller and How Does it Work?*, en. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/microcontroller (visited on 10/18/2021).

[30] *Tinkercad | Create 3D digital designs with online CAD*. [Online]. Available: https://www.tinkercad.com/ (visited on 10/26/2021).

[31] PlatformIO, *PlatformIO is a professional collaborative platform for embedded development*, en. [Online]. Available: https://platformio.org (visited on 10/26/2021).

[32] *Getting Started with RFID*, en, ISBN: 978-1-4493-2417-9. [Online]. Available: https://learning.oreilly.com/library/view/getting-started-with/9781449324179/ (visited on 09/13/2021).

[33] *Matplotlib: Python plotting — Matplotlib 3.4.3 documentation*. [Online]. Available: https://matplotlib.org/ (visited on 10/23/2021).

[34] J. Early, *latextable*, original-date: 2020-06-15T17:05:40Z, Sep. 2021. [Online]. Available: https://github.com/JAEarly/latextable (visited on 10/23/2021).

[35] *Welcome to Flask — Flask Documentation (2.0.x)*. [Online]. Available: `https://flask.palletsprojects.com/en/2.0.x/` (visited on 10/23/2021).