FACULTAT
DE CIÈNCIES
I TECNOLOGIA

UVIC | UVIC·UCC

Degree Final Project

ROS Educational Robot Kit

Sergi de las Muelas Acosta i Marc Sala Codina

Mechatronics Engineering

Tutor: Alberto Olivares Alarcos

Co-tutors: Marc Genevat Travesa and David Reifs Jiménez

Vic, June of 2022

Acknowledgments

First of all, we would like to give our most sincere thanks to a person without whom it would not have been possible to obtain the much desired results obtained from the project, our tutor and professor of Mobile Robotics at the University of Vic, Alberto Olivares Alarcos.

We are also grateful for the great contribution of one of our co-tutors for his guidance and collaboration in carrying out the Innova section of the work, Marc Genevat.

We would also like to thank the professor of Embedded Systems and also second tutor of the work, David Reifs, for his contributions and follow-up in the field of network communications.

Next, we say thanks to Mr. Joan Antoni Castejón, professor of Business Management at UVic, and Olga Codina, for their support in the economic and financial sections of the business plan.

Finally, say thanks to our friends, colleagues and especially family, for their support and encouragement during the development of the work, sincerely, thank you.

Title: *ROS Educational Robot Kit*

Authors: Sergi de las Muelas Acosta i Marc Sala Codina

Tutor: Alberto Olivares Alarcos

Co-tutors: Marc Genevat Travesa and David Reifs Jiménez

Data: 6th June, 2022

Key words: *ROS, sockets, multiprocessing, Python, Arduino, PyQt5, CAD design, SolidWorks, SLAM*

Abstract

This document presents the concept and first prototype of an educational robot to allow beginners and intermediate ROS users to learn more and test their work on a cheap and simple platform.

In order to achieve that, it has been developed an academic robot, which runs with ROS, to make interested people learn and practice their programming on this platform interactively.

The main knowledge fields in this project are: Python programming (Sockets, PyQt5, multiprocessing), Micro-controller programming on Arduino, CAD design with SolidWorks, Electronics, ROS, and business planning.

As a result of the project, it has been possible to develop a first prototype of the robot explained above, which can be programmed via ROS, being able to perform SLAM and more. Moreover, you can also interact via a computer application from which the user can create virtual maps and routes that lately can be executed on the robot, besides performing SLAM and connecting the robot to different networks.

With the business plan carried out, it has been possible to show that the product could take place within the market, triggering with benefits for the academic entities, interested individuals and finally for the company itself.

Títol: *Kit Robòtic Educatiu de ROS*

Autors: Sergi de las Muelas Acosta i Marc Sala Codina

Tutor: Alberto Olivares Alarcos

Co-tutors: Marc Genevat Travesa and David Reifs Jiménez

Data: 6 Juny, 2022

Paraules clau: *ROS, sockets, multiprocessament, Python, Arduino, PyQt5, disseny CAD, SolidWorks, SLAM*

Resum

Aquest document presenta el concepte i primer prototip d'un robot educatiu per permetre als usuaris principiants i intermitjos de ROS aprendre més i testejar el seu treball en una plataforma econòmica i senzilla.

Per aconseguir-ho, s'ha desenvolupat un robot acadèmic que funciona amb ROS, per fer que les persones interessades aprenguin i practiquin la seva programació en aquesta plataforma de manera interactiva.

Els principals camps de coneixement d'aquest projecte són: Programació Python (Sockets, PyQT5, multiprocessament), Programació de microcontroladors en Arduino, Disseny CAD amb SolidWorks, Electrònica, ROS i Planificació empresarial.

Com a resultat del projecte, s'ha aconseguit desenvolupar un prototip del robot explicat anteriorment, el qual es pot programar mitjançant ROS, sent capaç de realitzar SLAM i més, amb el qual també es pot interactuar mitjançant una aplicació informàtica des de la que l'usuari pot crear mapes i rutes virtuals, que posteriorment poden ser executades pel robot, a més de fer SLAM i configurar les xarxes de connexió.

Amb el pla d'empresa realitzat, s'ha pogut demostrar que el producte podria tenir lloc dins del mercat, desencadenant amb beneficis per a les entitats acadèmiques, persones interessades i finalment per a la pròpia empresa.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mechatronics is a multidisciplinary branch of engineering. It mainly merges knowledge from 4 different fields: Mechanics, Electronics, Informatics, and Control (see Fig. 1.1). It is so related to robotics that it is utterly difficult to distinguish between them. Indeed, both pretend to use robots or other devices in our daily tasks, and develop new technology to help people.

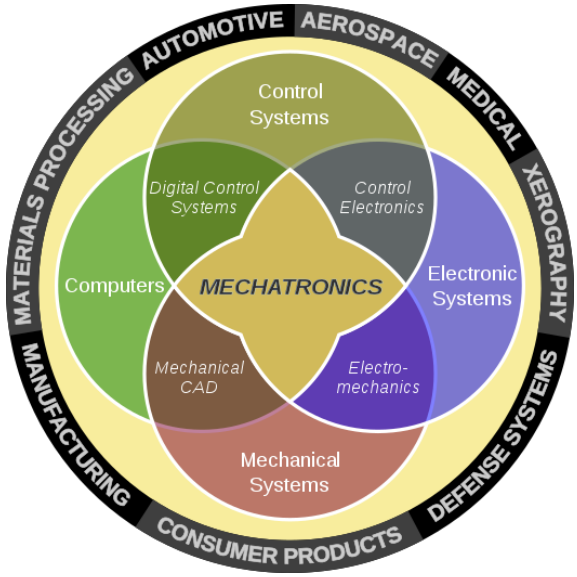


Figure 1.1: Mechatronics and its knowledge fields.

Robots consists in both hardware and software elements to be able to perform their tasks and accomplish their goals (e.g., to help people). For instance, ROS (Robot Operating System) is one of the most popular frameworks in robotics. It allows the writing of flexible robot software and simplifies the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Indeed, ROS is commonly used in research institutes and universities, and even in some new robotics companies. However, since the

learning process is quite complex, ROS is still unknown to most non-expert robotics fans. For that reason, many companies are working on teaching people about 'What ROS is' and 'How it works' (e.g., The Construct[1]). However, there is still room for improvement, thus, the idea of developing this project was born: we want to bring Robotics and ROS closer to people.

1.1 Motivation

A great way to develop a new affordable and easy-to-use platform to allow people to learn robotics, is to create a Starter kit so that people can introduce themselves to advanced robotics. Indeed, there are many starter kits in the market for electronics, mechanics and programming, besides platforms that simplify the task of building devices and program them (see Fig. 1.2).

(a) (b)

Figure 1.2: (a) Arduino Starter Kit, (b) Raspberry Pi Starter Kit.

Even though there are many options to learn electronics, we considered that there were still opportunities to enhance the way people teach all the important concepts in robotics. This fact is a barrier between people and the development of robotics devices. In general, people might be scared of the complexity of advanced robotics, but a few years ago, electronics was probably in the same position as robotics is today and platforms such as Arduino managed to break that barrier.

We are motivated to bring robotics closer to beginners and intermediate users, besides letting non-expert people enjoy robotics just by playing with it, and maybe waking up the desire of learning more about it. In order to do so, we decided to design, build, and

[1] `https://www.theconstructsim.com/`

program a mobile robot with its core running in ROS, and create a multi-platform app that, when connected to the robot, allows you to configure it or just play with it.

1.2 Objectives

The main objective of this project is to develop a low-cost ROS initiation kit and study the feasibility of introducing the product to the market, in other words, to create a business plan. This kit will supply the necessary tools (hardware and software) to allow beginner and intermediate users to interact with a real robot, either to configure it, play with it, or learn the background knowledge in this field. Furthermore, it will also let inexpert people use some robot functionalities (create routes and maps) from an App. This principle goal can be achieved by the accomplishment of the following specific objectives:

- Development of a ROS-based starter kit that can compete with the solutions already available on the market.
- Design and implementation of a multi-platform app that allows to design virtual maps and routes and use some robot functionalities remotely (e.g., perform SLAM, etc.).
- Validate the main functionalities of the designed robot, software and application in a set of use cases.
- Develop a business plan studying the viability of introducing the product into the market.

Chapter 2

Background and theory

This section supplies all the necessary knowledge to understand the background behind the different decisions made along this project. Specifically, we provide the different resources that were available during the process of development. We divided this section into four main concepts:

- Mobile robots: wheel configurations and kinematics.
- Open-source robotics hardware and software.
- App creation and design frameworks.
- Business plan creation.

2.1 Mobile Robots

Mobile robots can move in their environment, thus are not fixed to it. This ability of moving along the space as they want is a double-edged weapon. It allows much more freedom than static robots, but as the robot does not have any fixed frame, it becomes tedious to localize itself on the environment and control its position. Hence, robots need sensors and algorithms that use that sensor data combined with its kinematics to understand how they move along the environment through time. Depending on the medium through which they move they can be differentiated as ground, aquatic, and aerial robots. From now on, we will focus on the ground ones, since they are easier to design and build than the rest. They can be included in two large groups: wheeled robots, and non-wheeled robots.
Non-wheeled robots include legged robots and also other techniques of movement different from legs such as creeping. Legged robots can move through more difficult surfaces

than wheeled robots. However, they are much harder to control as they have more joints (degrees of freedom) and present more stability challenges.

On the other hand, wheeled robots are usually restricted to move through flat surfaces. Nevertheless, they are much easier to control since they usually have one joint per wheel and it is easier to ensure static stability. This simplicity, makes this type of robots suitable for our project, thus, in the following sections, we will explain more about them.

2.1.1 Wheel configurations

Wheeled robots are basically differentiated by three elements: type of wheels, number of wheels, and distribution of the wheels. In this section we will have a look to the most common configurations and we will briefly see their advantages and disadvantages.

The differential robot (see Fig.2.1) is a two wheeled robot with a third point of contact to the ground to ensure the static stability. This third element usually is a free wheel to allow movements in all directions, but it could be used a fixed contact, but in this case it would friction to the movement. The two wheels have one motor each one, that means the robot can perform just three movements on the ground, it can move front and backward, and turn in place (spin).

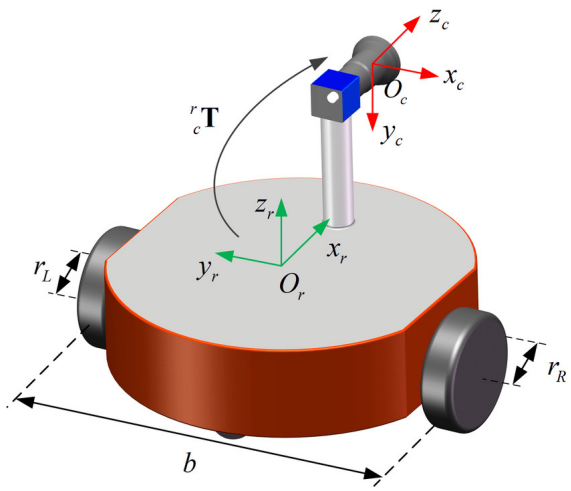


Figure 2.1: Differential drive robot. It is shown a sensor and the robot frames and the relationship between them, the homogeneous matrix.

Since it only has 2 wheels, and thus, 2 motors, it has a limited combination of movements between them (see Fig. 2.2). If it moves both wheels in the opposite direction and speed, it will turn in place either clockwise or counter-clockwise. If it moves both wheels in same direction and same speed, it will move front or backward. Finally, these combinations with different speed in each wheel will produce different curved trajectories.

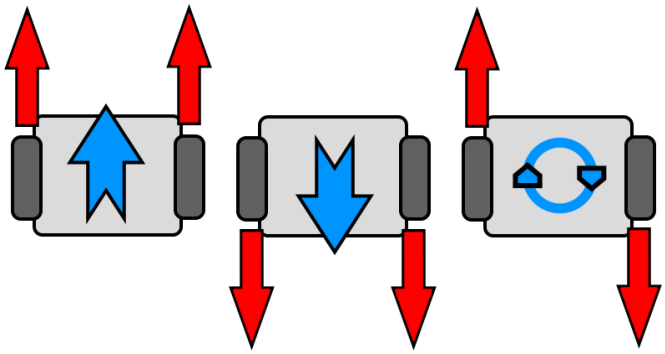


Figure 2.2: Differential drive movements depending on the direction of the wheels.

The 3 wheeled omnidirectional robot (see Fig. 2.3) is a quite common configuration that allows free movement in any direction thanks to its omnidirectional wheels. Each wheel contains little rubber rollers that allow free sliding on the perpendicular direction of the movement axis of the wheel. This fact combined with a properly angular speed results in a that free movement that we said before. However, it becomes more difficult to control, as it has more equations to take into account when computing its kinematics (in this document we will not explain the kinematics of an omnidirectional robot) in contrast to other robot distributions like the differential robot (see Fig.2.1)

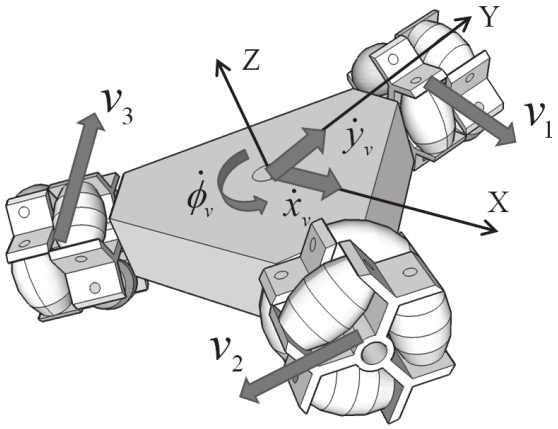


Figure 2.3: 3 Wheeled Omnidirectional Robot base with global frame and local wheel frames.

2.1.2 Kinematics: forward and inverse

As we have seen in the previous section, moving a differential robot all around the ground is quite simple, since you only need to control two motors. In this section we will see how to compute the wheels speed in order to obtain a global robot movement and vice versa, and it will be done with forward and inverse kinematics.

They are processes to understand the relation between the robot's joints movements and the actual motion of the robot in the space. Actually we focus only on the forward kinematics equations as the inverse kinematics is the result of isolating the previous equations. This process can have infinite solutions so you must apply constrains and other methods to simplify it, but it is a part of path planning algorithms and AI, which is out of scope this project.

Forward Kinematics allows to know how the whole robot is moving given the position of the joints [4]. In fixed robots, as there is an absolute reference frame, knowing the final joint positions is enough to compute the final robot position, in this case how the joints have moved during time is not really important, however, in mobile robotics, it is necessary to know the instantaneous position and velocities of each joint if we want to know the final position of the robot. It is necessary to know all the robot's geometry to get the equations. Besides, if we know how wheels move, we can predict the robot's trajectory. An important concept in forward kinematics is the Instantaneous Center of Rotation/Center (ICR/ICC). It provides information of where in the space it is placed the center of a virtual circle that describes the robot trajectory (see Fig. 2.4).

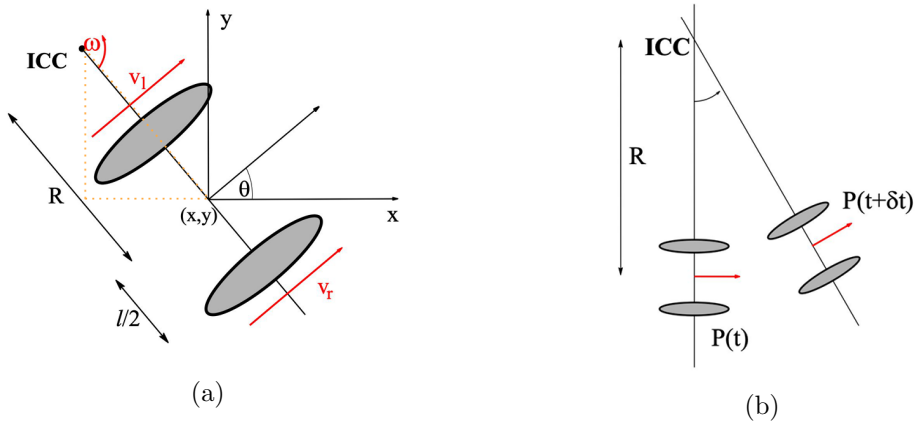


Figure 2.4: ICC for a differential drive robot and its relationship with the robot's architecture (a), and relation between the ICC and the robot's pose in time (b).

Using the variables defined in Figure 2.4a the equation of the ICC will be the following (equation 2.1):

$$ICC = [x - Rsin\Theta, y + Rcos\Theta]; \quad \text{where} \quad R = \frac{l(v_l + v_r)}{2(v_r - v_l)} \tag{2.1}$$

Together with the ICC equation, there are other useful equations to compute either the angular/linear speed of the whole robot (ω, V), or the linear velocity of the wheels

(v_l, v_r). These other useful equation are shown in 2.2

$$\omega = \frac{v_r - v_l}{l}; \quad V = \frac{v_r + v_l}{2}; \quad \omega(R + \frac{l}{2}) = v_r; \quad \omega(R - \frac{l}{2}) = v_l \tag{2.2}$$

Finally, if we want to get the position (x, y) of the robot and its orientation knowing the joints velocities, we can apply all the equations shown before (2.1 and 2.2) to get the complete forward kinematics matrix system (eq. 2.3):

$$\begin{bmatrix} x' \\ y' \\ \Theta' \end{bmatrix} = \begin{bmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \Theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix} \tag{2.3}$$

The equation 2.3 can be translated into words. First we compute the ICC (its x and y coordinates), then we translate the ICC to the (0,0) coordinates. Once in the origin we compute the X and Y of the robot depending on time and angular speed (ω) and we transform the result with the rotational matrix, and finally we undo the first translation of the ICC.

2.2 Open-source robotics software

Nowadays, ROS is the most widely used framework for advanced robotics programming, which extends to the industry, services, research, development, and academic centers, among others. For this reason, ROS has been deeply explored and developed during the last years, becoming a core element in many robots. For instance, autonomous AGVs, waiter/waitress robots, autonomous mobile manipulators, etc. (see Figure 2.5).

(a) (b) (c)

Figure 2.5: Examples of robots that are usually programmed using ROS. (a) AGV, (b) Waiter-bot, (c) Turtlebot.

As learning ROS is quite tough at the beginning and it requires some previous knowledge of programming using other languages like Python or C++, it hasn't become as

popular as other open-source platforms like Arduino or RaspberryPi. However, once you understand how ROS works, it becomes intuitive and powerful. The ROS structure is based on packages, nodes, publisher/subscriber, topics, services, and actions [5]. In this section we will see this structures in more detail.

2.2.1 ROS Packages

Software in ROS is organized in packages. A package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically constitutes a useful module. The goal of these packages is to provide this useful functionality in an easy-to-consume manner so that software can be easily reused.

A ROS packages is simply a directory that has a *package.xml* file and is registered in the ROS_PACKAGE_PATH Linux environment variable. Packages are the smallest individual element you can build in ROS and it is the way software is bundled for release. So, inside your ROS workspace, you will have packages that will contain useful tools for ROS (e.g nodes, services, libraries, etc.).

2.2.2 ROS Nodes

The most basic unit in ROS is the 'Node'. It is and executable file that, at least, performs an independent task. Each node can be created separately and no matter the programming language used for it. They can interact between them using different methods depending on the needs of the system. This methos are Publisher/Subscriber, Services, and Actions, and we will explain them in future sections.

When ROS starts a special node is launch at the beginning. It is called ROS Master. If ROS is running in different independent devices, you can have one ROS Master per device, but they will not be able to communicate between them. If you want to have them working together exchanging information, then you need to use only one ROS Master and connect the other nodes to it. After that, when a node connects to a specified master, the master registers details about: the message streams they publish, services and actions that they provide; and streams, services and actions they consume from other nodes.

2.2.3 Publisher/Subscriber architecture

One of the architectures we said before was the Publisher/Subscriber. Once the ROS master and every other node are initialized, they can send and received messages between them using it. A graphical representation of the process is shown in Figure 2.6.

As we saw in Figure 2.6, nodes publish data into one or many topics. These topics can

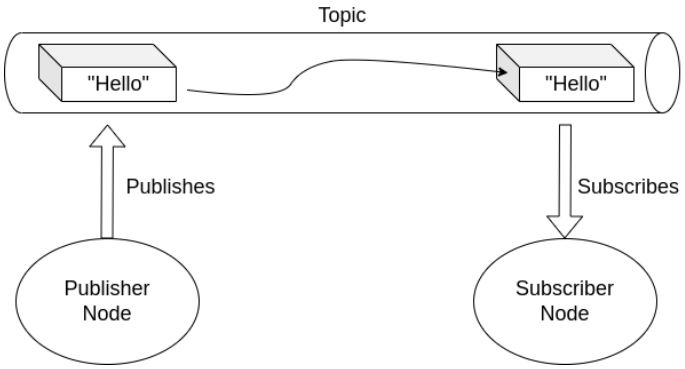


Figure 2.6: Publisher/Subscriber concept.

be seen as a pipe where the published data travel through and stays there until its lifetime ends. Nodes can also subscribe to topics and get the data that is traveling inside them. Nodes can find topics or other nodes to interact with them thanks to the ROS Node Master that provides connection information to nodes, so that they can communicate and exchange messages.

Besides, when a new node appears, the master provides it with the information that it needs to form a direct peer-to-peer TCP-based connection with other nodes publishing and subscribing to the same message topics and services.

A common example of use of the Publisher/Subscriber architecture is found in mobile robotics localization, where sensor data is sent from one node to another. For instance, if you have a LIDAR sensor that needs a special protocol to transfer its data, you can implement a ROS node with that protocol and publish the data in an standard format. Therefore, if in the future it is necessary to change the sensor, you will only need to change one node and implement the new one with the new sensor protocol. Latter, you can use this published data from other nodes to make decisions such as obstacle avoidance. An example of publisher/subscriber diagram in mobile robotics localization is depicted in Figure 2.7).

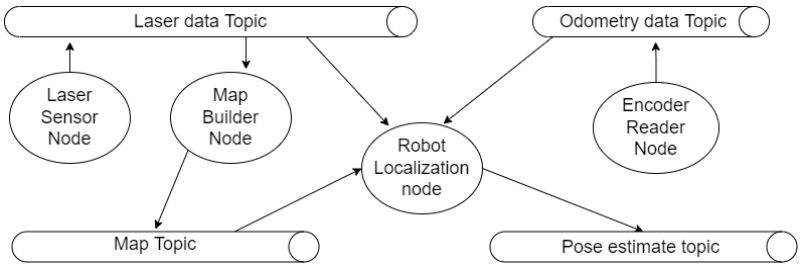


Figure 2.7: Publisher/Subscriber diagram of a mobile robot localization. The Localization node gets information of distances from the Laser topic, how the robot moves from the odometry topic, and matches that information with a map to get a pose estimation.

2.2.4 Service/Action/Client architecture

In some applications it is needed a synchronous communication between nodes instead of the publisher/subscriber asynchronous architecture. The Service/Client architecture allows to encapsulate some processes that we want to execute synchronously. A Service server is created containing the desired task or functionality that the robot would be able to execute. A Service client would ask the server to perform the task and wait (blocking the execution) until the server answered with the result (see Fig. 2.8).

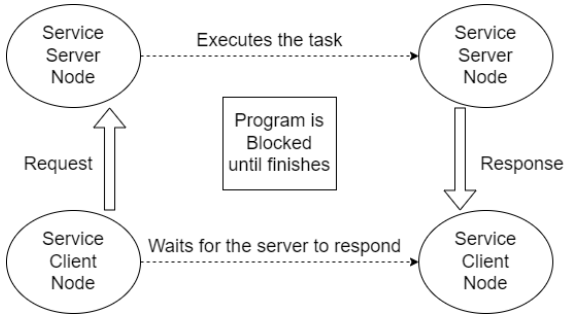


Figure 2.8: Service/Client concept. Client sends a Request to the server and blocks the code waiting for a response.

Actions are a standardized interface, they are provided by the *actionlib* stack. Actions allow to execute processes synchronously without blocking the code. It is pretty useful when running different tasks at the same time (e.g. a robot navigation running and moving arm actuators). It is possible to check the state of the action in all the moment (feedback) to see if it is still running or it has finished, either success or failure (result).

2.3 Open-source robotics hardware

In this section we describe the main and most common hardware elements in a robotics starter kit: controllers, sensors and actuators.

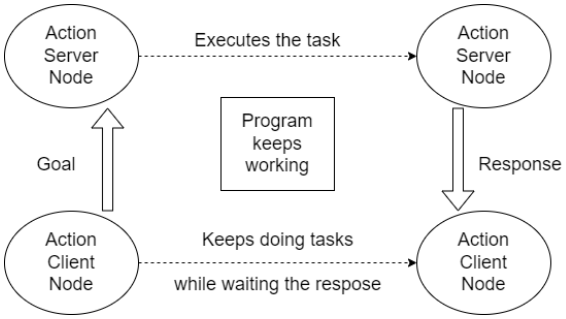


Figure 2.9: Action/Client concept. Client sends a Goal to the server and keeps doing tasks while waiting the response.

2.3.1 Controllers in mobile robotics

Starting with the controllers, Arduino and Raspberry Pi are among the two most popular open-source platforms. Both are available from a low cost and provide a high potential to develop prototypes and advanced devices. There are other controllers such as Microbit, Teensy, STM32, Nvidia Jetson, etc. Indeed, the list is quite large and since they all share the main functionalities needed for this project, we just focused on the two we used Arduino and Raspberry Pi.

2.3.1.1 Arduino

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. There are several boards and microcontroller configurations, but we are going to focus on the Arduino Nano board (see Fig. 2.10), because it is the one we have used in this project.

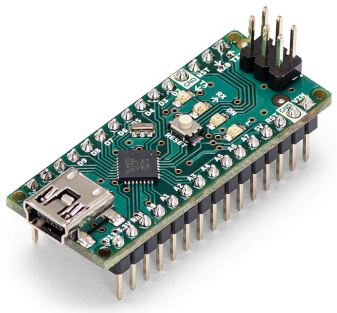


Figure 2.10: Arduino nano board.

The technical features of this board are:

- Microcontroller: ATmega328.

- Operating voltage: 5V
- Input voltage: 7-12V
- Digital I/O pins: 22 (6 of which are PWM[1])
- Analog Pins: 8
- Power consumption: 19mA

2.3.1.2 Raspberry Pi

Raspberry Pi is a series of small single-board computers (SBCs) developed by the Raspberry Pi Foundation in association with Broadcom [2]. Raspberry Pi SBCs feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). There are several series and generations of the Raspberry Pi SBCs, but we focus on the Raspberry Pi 4 model B 1GB (see Fig. 2.11), as it is the one we have used in this project.

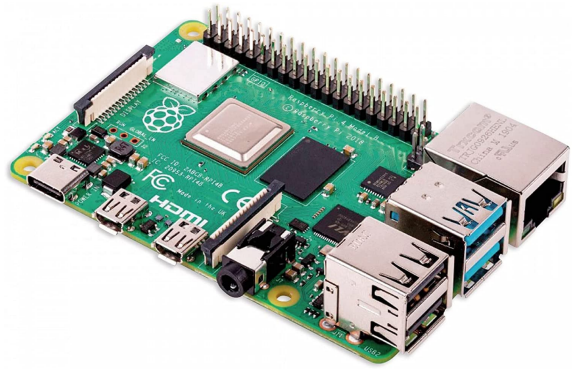


Figure 2.11: Raspberry Pi 4 model B.

The most relevant technical features of this board for this project are:

- CPU: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- RAM: 1GB LPDDR4-3200 SDRAM
- Connections: 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- USB pins: 2 USB 3.0 ports; 2 USB 2.0 ports
- Storage: Micro-SD card slot

[1]Pulse-width modulation, used to change average power an electrical signal has.
[2]`https://www.broadcom.com/`, last accessed May 30, 2022.

- Power consuption: 5V DC via USB-C connector (minimum 3A*)/5V DC via GPIO header (minimum 3A*)

2.3.2 Commonly used sensors in mobile robotics

There are a lot of available sensors in robotics, and depending on the kind of robot you desire to build it is necessary to choose the sensors wisely.
We want to build a mobile robot with the purpose to move and localize within the environment. Hence, we need sensors to scan the surroundings of the robot (e.g., range sensors), and to read the real movement of the motors (e.g., encoders), and thus, the approximated real robot movement.

Lidar (see Fig. 2.12) is an acronym of "light detection and ranging" and is a method for determining ranges (variable distance) by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. In robotics it is usually used to build maps and localize robots in the environment.

Figure 2.12: Lidar sensor model YDLidar X4.

An incremental encoder is a linear or rotary electromechanical device that has two output signals, A and B, which issue pulses when the device is moved. Together, the A and B signals indicate both the occurrence and direction of movement. Many incremental encoders have an additional output signal, typically designated index, which indicates the encoder is located at a particular reference position (see Fig. 2.13). In mobile robotics it is used to compute the linear movement of each wheel. This information is usually used as feedback to navigation packages.

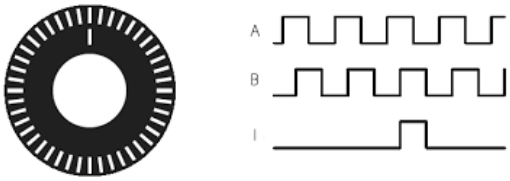


Figure 2.13: Incremental encoder.

The camera is a good sensor for perception, it allows to get information from the environment in a similar way as humans do. With all the computer vision techniques and artificial intelligence (e.g. segmentation, recognition, etc..) we can get very nice results some applications such as manipulation, where we need to identify objects and obstacles. There are 2 main types of most used cameras, RGB cameras (see Fig. 2.14a, they provide color information of the environment) and depth cameras (see Fig. 2.14b, they can be transformed into point clouds and get similar results as the lidar provides). The problem of cameras is that the depend a lot on the environmental light and processing images supposes a huge computational cost that a cheap and simple starter kit can not provide.

(a)

(b)

Figure 2.14: (a) RGB camera, (b) Depth camera (usually called RGBD camera).

Another very common sensor is the ultrasonic sensor (see Fig. 2.15). It measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target). It is very common in all different robotics applications, the only problem it has is that it is not really accurate and its dynamic range is quite short, in other words, it can only measure small distances. In mobile robotics is usually used for obstacle avoidance, but only when that obstacles are close enough.

Figure 2.15: Ultrasonic sensor.

All these sensors allow the robot to understand its environment. There is a technique that uses the sensor information to localize and the robot while building a map of it. This technique is called *SLAM* (Simultaneous Localization And Mapping, see Fig. 2.16). It is a complex computational problem but there are several algorithms already known for solving it. Popular approximate solution methods include the particle filter, extended Kalman filter, and covariance intersection. SLAM algorithms are based on concepts in computational geometry and computer vision, and are used in robot navigation, robotic mapping and odometry for virtual reality or augmented reality.

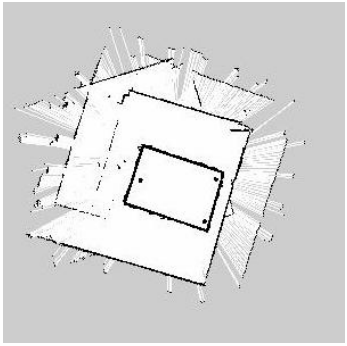


Figure 2.16: Result example of a Map built using SLAM with a laser sensor.

2.3.3 Actuators in mobile robotics

In mobile robotics there are not much actuators to choose, in fact the only actuator used is the motor. But there are several types of motors to choose and drivers to control them, so in this section we will have a look to the most popular motors in robotics.

The DC motor (direct current motor) is the mots popular one (see Fig. 2.17a). It is any of a class of rotary electrical motors that converts direct current (DC) electrical energy into mechanical energy. Their torque and velocity is proportional to its input voltage, so if you want to modify the speed, you need to modify the voltage. This modulation is

commonly done with motor drivers (see Fig. 2.17b) that allow some PWM control signal and modulates the voltage according to it. Their problem is that there is not a precise control of position and they rotate too fast, so they usually comes with a encoder and a reduction gear box.

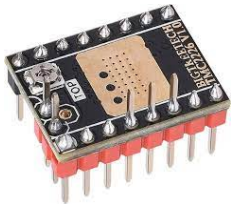
(a)

(b)

Figure 2.17: (a) DC Motor, (b) DC Motor Driver.

The Stepper motor is an other of the most popular ones (see Fig. 2.18a). It is not a DC motor, but it is a AC motor which is controlled by varying the frequency of the input voltage signal. The Stepper motor is a very accurate motor, it is used in industry to build very precise machines, as 3D printers, milling machines, etc. Each pulse it is supplied to the motor corresponds to a step, and depending the motor model this step is a fixed number of degrees. This variation of frequency and the enforcement of the steps is done by the motor drivers (see Fig. 2.18b). The problem they have is that they are slower than the DC Motors and they have less torque, and also they have more power consumption.

(a)



(b)

Figure 2.18: (a) Stepper motor, (b) Stepper motor driver.

All these actuators need to be controlled and one of the most popular control loops in industrial aplications is the PID [2]. A PID controller (proportional–integral–derivative) is a control loop mechanism employing feedback that is widely used in industrial control

systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value ($e(t)$) as the difference between a desired set point (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.

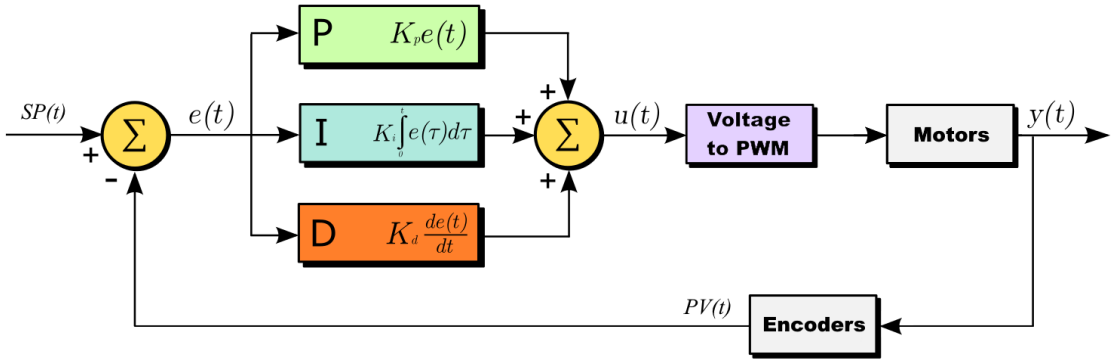


Figure 2.19: Control Loop diagram

As we can see in the figure 2.19 the error obtained by the difference between the SP and the PV enters inte the PID block and the results of multiply, integrate and derivate are added and used as compensation to the process block. Each part of the PID has its own constants, which need to be tunned. We have done it empirically, specifically the method is the following:

- Set Ki and Kd values to 0.
- Increase de Kp until the output oscillates or takes the desired shape.
- Set Kp to half that value.
- Increase Ki until the offset between SP and PV becomes 0.
- Increase Kd to clean the overshoot.

2.4 Robot design: CAD software

In order to design the first version of the robot's prototype, it's common to use a CAD software such as AutoCAD, FreeCAD, SolidWorks, etc. The acronym CAD, stands for Computer Aided Design. Therefore, this type of software is a virtual tool that allows users to design in three dimensions by using a computer substituting the manual drawing. A CAD software comes with essential advantages when creating a product. For instance,

there's no material cost, the piece sizes can be instantly modified and the created piece can be digitally assembled and tested.

Most of the CAD software options share the same functionalities and a deep analysis of the different options is out of the scope of this work.

In this project, we used SolidWorks, because we already had experience of how to utilize it. Furthermore, it is one of the most popular CAD software in the industry nowadays. This software is really complete, because it allows doing simulations, calculation of stresses and strains, and generate blueprints from the designed pieces, among others. In Figure 2.20, it can be seen the environment of the software.

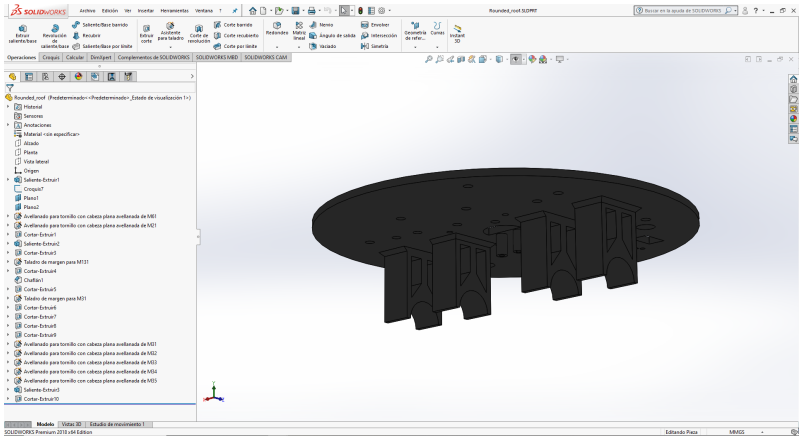


Figure 2.20: SolidWorks workspace.

2.5 Prototype production

Additive manufacturing is an industrial sector that is gaining more and more momentum in the world of the industrial manufacturing. It's manufacturing process, it's based on adding layers of material, one on top of the other, in order to achieve a piece made without subtraction of material.

2.5.1 Additive manufacturing technologies

There are a lots of additive manufacturing technologies such as stereolithography, binder jetting, selective laser sintering, 3D printing and material jetting. See the article [7].

- **Fused Deposition Modeling**

  The Fused Deposition Modeling, is the most popular of the additive manufacturing technologies nowadays. Its process is based on the fused deposition of material layers one over the other (see Fig. 2.21a). The material used in this technology is a

filament, usually made of plastic or metal, which, when passing through an extruder with the melting temperature of the material, this last enters in a semi-molten state, thus being able to be molded and unified with the other layers.

- **Stereolithography**

  It's an additive manufacturing system that using a laser, the liquid material deposited on the base of the machine is solidified. The materials used in this technology are liquid polymer resins. It's usually used to carry out prototypes as the 3D printing.

- **Binder jetting**

  It's a technology that using the addition of powders of a material and binder, alternating them layer by layer, creates pieces and objects previously designed with a CAD software. With the addition of binder, it's possible to solidify the powdered material to achieve the structure desired. The materials used in this technology, as commented before, are powdered polymers. It's used to carry out parts for automotive, aerospace's pieces, medicine and jewelry.

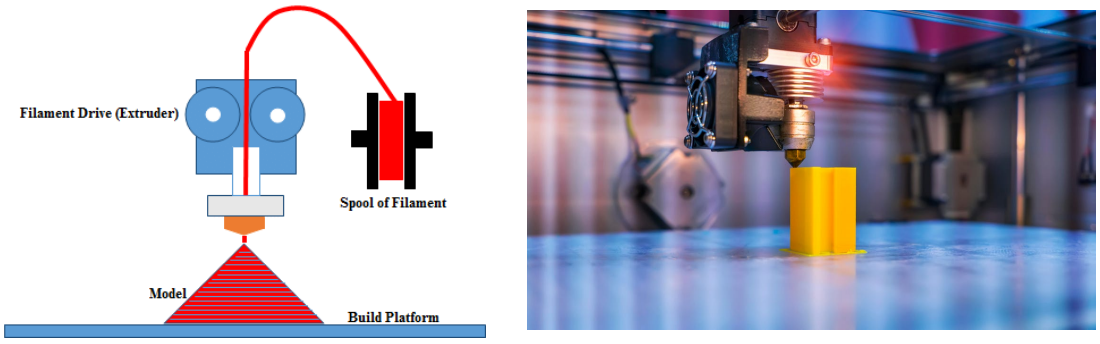
- **Material jetting**

  The material jetting process, consists on depositing drops of liquid polymer, thus creating layers of material (when solidified) and giving shape to a structure or piece. This technology is commonly used in prototypes, medical applications and jewelry.

- **Selective laser sintering**

  The selective laser sintering (SLS) is a type of additive manufacturing that is based on the use of lasers or electron beams to melt and fuse powdered material. The uses of the pieces obtained with this technique are mainly non-permanent prosthesis and prototypes.

In order to carry out the prototype, we decided to use the 3D printing system, as it is currently the cheapest and most widespread one. Recall that one of our objectives is to create a competitive product in terms of cost. Furthermore, people could more easily access to 3D printing than the other additive manufacturing technologies. Even though this kind of technology is effective in order to make a prototype, it's not so useful for chain manufacturing, as it's a slow technology that produces relatively poor surface finishes (see Fig. 2.21b).



(a) (b)

Figure 2.21: (a) 3D printing process scheme, (b) 3D printing process where appears a part of a 3D printer machine and half of a 3D printed piece.

2.5.2 3D printing materials

As explained previously, the materials used with the 3D printing technology are polymers filaments (see Fig. 2.22). These filaments are thermoplastics, non-recyclable polymers that can just be fused once. The most commonly used filaments are the ones of PLA, ABS, PETG, PVC and PP, although there exist lots of materials and filaments compatible with this technology. It'll depend on the specifications desired for the manufacturer, the chosen of the material.

The filament used in order to create the prototype, was the PLA filament. It was been chosen this material because it's a so easy material to print and it has good properties. In order to take care of the environment, it has been decided to just use this technology with the prototype and look for another technology that allows recyclable materials to manufacture the product.

Figure 2.22: Different kind of 3D printing filaments.

2.5.3 3D printing software

To print a 3D piece, it's necessary to generate a g-code of the piece to manufacture. An easy and interactive way of achieving this last one, is by designing a CAD model. Once the CAD model designed, it has to be saved as a .stl file in order to can generate later the g-code. Once saved, there exists lots of programs to convert .stl files to g-code, which are based on the 3D printing. This programs are so dynamic and so intuitive, and most of the 3D printers in the market has its own software. The most popular ones are the Ultimaker Cura, the Creality 3D and the PuraSlicer among others. The one that has been used to create the prototype is the BCN3D Stratos because its the one that uses the 3D printer with which the pieces where printed. See the Figure 2.23 to see the framework of a 3D printing software.

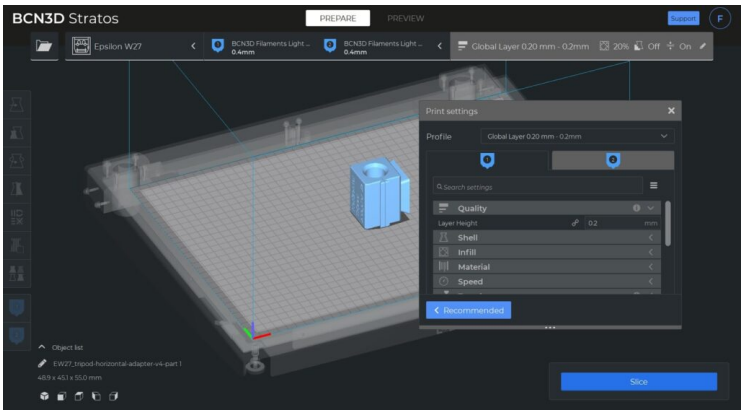


Figure 2.23: Software to generate a g-code and its workspace (BCN3D Stratos).

2.6 App creation and design frameworks

Nowadays, there exist lots of computer and mobile phone apps. These applications, have been developed using different programming languages, as C, C++, Java, JavaScript, and Python among the most known. As during the career it has been learned to program with Python, it was decided to develop the app using this programming language.

Python is a widely used, high-level programming language and it has many libraries that allow the user take profit of them in order to develop different kind of programs and functions. Over these libraries, there exist several to develop graphical interfaces as Tkinter, Kivy, PyQt5, PySide, etc. Below, there is an explanation of the ones that were considered before starting developing the graphical interface. To know more about these frameworks, please visit the webpage [1].

- **Tkinter**

  Tkinter, is the Python library of the Tcl/Tk GUI toolkit. Tk is an open-source, cross-platform software used by many different programming languages to build GUI programs. Tkinter, instead, is the result of the Python module of Tk. It allows the programmer develop graphical interfaces using the Python language. It's easy to learn and it works in Windows, macOS and Linux.

- **Kivy**

  Kivy is a free and open source Python framework, used for developing computer and mobile applications. Its graphics are processed over OpenGL instead of its own widgets, what allows that the apps created with this platform be the same in any operating system. It runs on Linux, Windows, OS X, Android, iOS, and Raspberry Pi.

- **PyQt5**

  Qt consists in a cross-platform work environment for creating programs with a graphical user interface. The combination of the both Python results in a library with the ability of creating graphical interfaces through Python programming. It can run on Windows, Linux, Raspberry Pi, macOS, iOS and Android.

It has been chosen PyQt5 as a framework to program the app, because Python is a simpler and more understandable programming system than C or C ++, and during the degree, it has been used this programming tool. In addition, many of the applications that are used every day have been designed with this language. Some examples are Netfilx, Instagram, Pinterest, Uber, and more. However, Python also has other libraries such as Tkinter, Kivy and PySide, but PyQt is the platform that has been used during the degree and a good point of it is that it also allows the user to create both computer and mobile applications.

2.7 External communication: Sockets

Sockets is a communication system between two processes in order to share data in an orderly and structured manner. There exist three kind of protocols that can be used in order to establish a socket communication, the TCP, the UDP and the IPX [6].

- **TCP**

  TCP, is the acronym of Transmission Control Protocol, and as its name indicates

it's based on the transmission of data between two or more protocols in a controlled manner. It means that before transmitting the data, it makes sure that the communication has been successfully established. Another characteristic of the TCP protocol is that it assures that the data transmitted has no errors before arriving in its destiny.

- **UDP**

  The User Datagram Protocol (UDP), is a datagram transmission system that, unlike the TCP protocol, it don't need to establish a previous communication to send data. This is because the same datagram, contains enough information of its destiny address. It neither has any control of the data sent, so it has to compile less data, and for this reason it's faster than the TCP protocol. Its main difference with the TCP, is that the UDP don't allows a bidirectional communication. To have a clearer idea of the differences between them two, please, see the Figure 2.24.

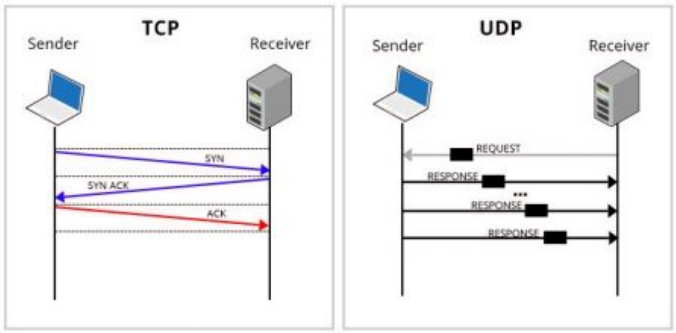


Figure 2.24: Diagram of the operation of the TCP and UDP communication protocols.

- **IPX**

  The IPX or Internet Packet Exchange, is a protocol based in the same idea of the UDP. It also transmits data in form of datagrams and have no control of the data sent. The difference between the UDP and the IPX is that this last one is a Netware layer protocol, which just allows a communication with Local Area Networks (LANs).

Knowing the types of protocols to transmit data between two protocols, and basing on our needs, it was chosen the TCP protocol to carry out the communication system of the kit, specifically the TCP/IP protocol. A TCP/IP socket, is a protocol to communicate two devices from the server IP identification of one of the two devices. This protocol allows electrical devices be wireless connected between them using the IP of both devices.
There exist lots of programming languages that allow the programming of TCP/IP sockets,

but as the application has been created using python, and most of the robot too, the programming of the sockets has also been programmed with python.

It has been explained what is a TCP/IP socket communication, but how does it woks? In order to create a wireless communication between two devices, there have to exist a server and a client. The server is the one that waits for a connection, while the socket client is the one that via recognising the IP of the server, tries to establish connection with it.

The idea of mounting a socket protocol, is to have a two-way connection between the server, which in the case of the project would be introduced inside the robot, and the client, which would be included in the application. To understand the scheme that follows a TCP/IP socket protocol, please, see the Figure 2.25

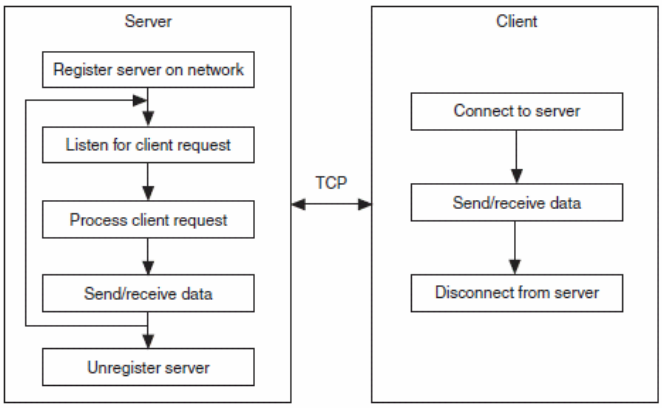


Figure 2.25: Structure of a TCP/IP protocol diagram (Server/Client).

2.8 Business Plan Conception

A business plan is a document in which appears the goals and strategies of a business, in order to be introduced or expanded in a market sector. Although the main part of a business plan are the strategies that the company plan to achieve its goals, some analysis of the market must be realized to let the business prepare good attacking and defending strategies.

Every business plan presents a similar structure: analysing the market sector of their interest, the competitiveness of its sector and creating some attacking and defending plans. For this reason, this section has been divided in five main parts, the two first sections are about analysing the market sector (market study and competitiveness), and the last three sections has the goal of planning good strategies to attack, defend or consolidate a place in a market sector.

2.8.1 Market Study

A market study is a detailed analysis of the market sector that a company has interest in. This analysis has the aim of finding the virtues and shortcomings of the own company in order to be useful to later prepare good plans and strategies. A market study is based on doing some research about the following points:

- **Business Advantages**

  In this part, the business has to analyse itself in order to detect the advantages of its product or service. This will be useful to compare its product or service with the other companies in the same sector to then propose improvements or to take profit of its this advantages.

- **Target**

  The target of the product or service that the company offers are the public and customers in which is focused. Making a good analysis of the target, the business will be able to know which sector of the market to attack.

- **Potential Customers**

  The potential customers are those who could be interested in the product or service that the company offers, but has not yet purchased it. In many cases the target and the potential customers are not differentiated and they are put in common.

- **Potential Market**

  If the target and the potential customers were those ones who are or could be interested in the product or service, the potential market is the same but wholesale. The analysis of the potential market will make the company know the countries to establish its commerce (depending on the culture and habits of it), the age of its customers, the gender, the occupation, and more.

- **PESTEL Analysis**

  PESTEL analysis is the acronym of Political Economical Social Technological Environmental and Legal analysis. This study is an internal study of the business that helps it to see the limitations, advantages pros and cons of itself, and let the business know what must they improve of every specific aspect.

2.8.2 Competitiveness

In this section, the business is able to compare its product or service with the similar ones that other companies offer in the market. The competitiveness analysis is an study of the possible companies that offer a similar product or service as the own business. It's useful to perform improvements and see the advantages of the own product or service in order to plan a good marketing strategy.

2.8.3 Marketing Plan

The marketing plan is the set of analysis and strategies that the company must consider and perform to achieve the goals it has set itself in a specific time. The main analysis and strategies of this part are the following ones:

- **SWOT Analysis**

  The SWOT analysis is the study of the internal and external factors that will make the company be better than the competence and attract more customers. In this study, the company analyses their strengths, their weaknesses, the market opportunities and the threats. With this data, the enterprise is able to create temporary strategies to take profit of the competence and the market.

- **CAME Analysis**

  If the SWOT analysis was the study of the internal and external factors of the own company, the CAME analysis is the one that propose solutions and alternatives in order to make the business go further. The acronym CAME makes reference to Correct the weaknesses, Adapt to the threats, Maintain the strengths and Explore the opportunities, and as it has been explained above, and knowing the meaning of CAME, it's easy to deduce that this analysis has the aim of taking advantage of the SWOT analysis.

- **Advertising Plan**

  The advertising plan is a vital point in order to attract clients. With a good advertising strategy, customers will decide if they go for the own product or service or for the competence one. If the product or service that the company offers better than the competence one but the publicity of the competence is better than the own one, probably the customer will decide to purchase the competence product/service. This plan, will depend on the kind of public that the company wants to attract and

their interests.

- **Business Action Plan**

  The business action plan, as its name indicates, has the mission of making the company reach their attacking goals, by taking profit of the analysis realized previously. This plan must contain the quantity of products to be sold or the quantity of clients to achieve in a specific period of time, and the way to achieve that number in that time.

2.8.4 Sales Plan

A sales plan has the same objectives as the marketing plan, but just focused in the sales of the product or service offered by the company. It must also be as specific as the business action plan, and it is the base of the economic and financial plan. Based on the predictions of this plan, the economic and financial plan will take better or worse results.

2.8.5 Economic and Financial Plan

All previous plans were based on analysis to create strategies and achieve certain goals that the company set itself. However, the economic and financial plan is the set of tables and calculations that allow the business extract numerical data from the company. With the previous plans and analysis, it cannot be proved if the enterprise could generate any profit or if it would result in loses. This plan is able to compute the temporary results of the business, as the investment and financing, the provisional results accounts, the profitability threshold, and more.

Chapter 3

Development of the ROS starter Kit

In this section we provide an introduction to the complete novel product: ROSNI (robot + app). Then, we explain both elements separately in more detail. The section is structured as follows:

- Complete product concept: ROSNI (Robot + Application).
- Robot development: from scratch to prototype.
- Application development: design and implementation of a Graphical Interface.
- Robot-app communication and integration schema.

3.1 Complete product concept: ROSNI (Robot + Application)

The main work developed in this project is: ROSNI, which involved the design and construction of a differential robot, and an interactive application to allow any not-expert to use the robot. The app should let the user set up the robot, create maps, design routes and execute them (at least in a simulated way).
More details about the robot and the application are explained bellow.

3.1.1 Robot presentation, requirements and decisions

One of the parts of the product is the differential robot. In this section we present this element of the product ROSNI, the requirements and our decisions, and in future sections

we will explain the technical development of it.

We have designed all the structure, chassis and electronics.
The aesthetic design is inspired by some actual robots such as Roomba [1] and TurtleBot [2]. Before starting designing and building it, we make some decisions how the robot should be, and we came up with a set of requirements that the robot shall satisfy:

- **Do-It-Yourself approach**

  This project aims to be educational and to let everyone build the robot on their own while learning through the process. Hence, all the pieces have been designed considering that they could be produced using a 3D printer, which added some minor constrains such as the precision and tolerances (the plastics can dilate and contract), the mechanical structure (shape limitations because of 3D printing).

- **Affordable product mentality**

  As we want to create a robot reachable to most of the people we tried to spend as little money as possible. Using generic components and materials, easy to find online. And bearing in mind to keep under the prices of the other robots abailable in the market (e.g Turtlebot 600€).

- **Open-source hardware and software**

  Open-source platforms are a really good option if you want to build a product to be used by the community. Indeed, we have also made use of other open-source tools such as Arduino (for actuator control), Raspberry (CPU), and ROS (core software). It is a very powerful tool because people can keep improving your device from where you leave it, instead of losing time trying to reach what you have already achieved.

3.1.2 Interactive app presentation, requirements and decisions

The other part of the product is the Interactive App. In this section we present this element of the product ROSNI, the requirements and our decisions about it, and in future sections we will explain the technical development of it. This App allows the user to manipulate the robot and interact with it dynamically. This application is multi-platform, thus it can be used in Windows, Mac, Android, Linux, Raspberry PI and IOS.
The app has some different functionalities: sending commands to the robot, creating virtual maps and routes, accessing to the robot's wiki, and more. Below, there are explained

[1] Autonomous robotic vacuum cleaner sold by iRobot.
[2] `www.turtlebot.com`

the requirements that the application must fulfill:

- **Create virtual routes and maps**

  This first requirement, is about making the user able to design virtual routes and maps, by inputting the coordinates of the points that the robot has to reach to arrive to its destiny. It is accompanied with the functionality of introducing, modifying and suppressing obstacles, simulating the robot's movement over the route designed, and more other minor functionalities.

- **Communicate with the robot**

  The app, must be able to establish connection with the robot in order to communicate with it. It includes the HotSpot and Wi-Fi network connections and the bidirectional message transference.

- **Have access the robot's wiki**

  The last one of the requirements is that the app shall has access to the robot's wiki. Explaining it differently, the user must be able to open the robot's wiki by pressing a button of the app.

3.2 Robot development: from scratch to prototype

In this section we explain the main tasks that has been carried out in order to obtain the robot, from the first conception of it to the final result.

3.2.1 Robot chassis design using a CAD software

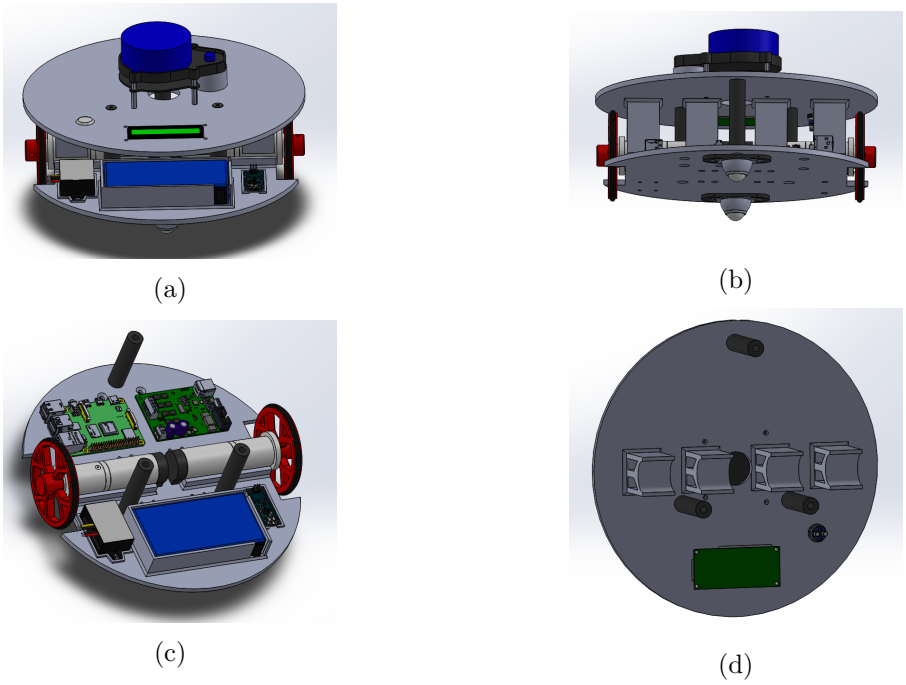
The CAD design, is so necessary in order to consolidate the concept of the product or prototype.

The first concept of the robot and its CAD design, were too different from the definitive one. In the first CAD prototype, the roof and the base of the robot were rectangular, and all the electrical components of the robot were supported on the top of them.

It was developed a second prototype changing the shape of the base and the roof from rectangular to rounded. The reason of this change, was because in case of collision, with a rectangular base and roof, it would be easier for the robot to get stuck. Instead, a rectangular shape, would allow the robot turn on itself don't getting stuck.

Once this new concept of the robot was established (the rounded one), it was developed the CAD design (see Fig. 3.1). As it can be seen in the Figure 3.1, all the electrical

components lay on the base of the robot, except for the LIDAR and the LCD screen, which were decided to be located on the roof.

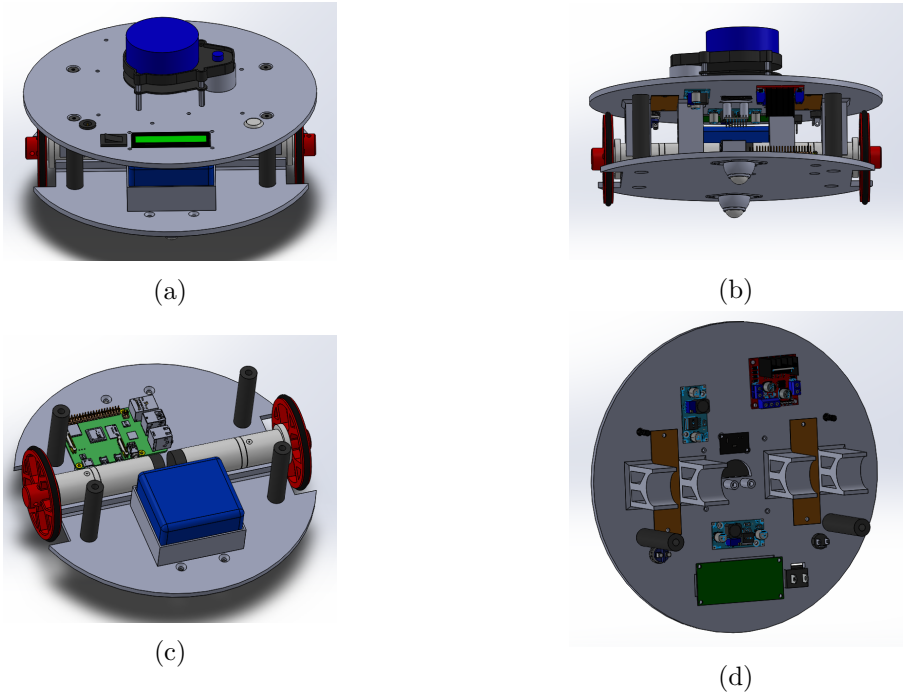


(a) (b) (c) (d)

Figure 3.1: First rounded prototype: (a) Top front face of the robot assembly, (b) Back bottom face of the robot assembly (c) Base + Electronic components, (d) Roof + Electronic components.

This concept didn't work, as a large part of space in the robot was wasted, thus a second prototype was developed. In addition, we also changed some electronic components for reasons related to efficiency, effectiveness and performance.

In order to solve the existent problems of this last prototype, a new prototype was developed. The dimensions of the robot remained the same, but the electrical organization and most of the electrical components were substituted and many others were added. The new idea was to take profit of the base and the roof of the robot to place electronic components attached to the roof upside down (Fig. 3.2). This way, the connections of the electronic devices were more organized and the new components that had to be added to the robot had enough space. There was also an improvement in the placement of the Raspberry PI, as in the first prototype, certain USB connection points were hidden, preventing potential connections. It was also noted that the Raspberry could damage the plastic of the base due to the high level of heat it generated. In order to solve it, there were extruded four little supports that separates the Raspberry Pi from the base. It was also added a now column and redistributed the other ones, in order to avoid vibrations and because the front column prevented the front crazy wheel from being attached.



(a) (b) (c) (d)

Figure 3.2: Definitive prototype: (a) Top front face of the robot assembly, (b) Back bottom face of the robot assembly (c) Base + Electronic components, (d) Roof + Electronic components.

Once the definitive prototype designed, we developed some design plans of the pieces that needed to be manufactured. In the Appendix A, there can be seen each plan: in Fig. A.1 the columns plan, in Fig. A.2 the base plan, in Fig. A.3 the roof plan, A.4 wheel plan, and in Fig. A.5 the assembly plan.

3.2.2 Robot electronic components and connections

The robot contains many different electronic components that need to be connected to each other, either to transfer data, or to get or supply power. However, there is a big challenge to properly do the connections: the reduced space. The whole space needs to be optimized to fit in all the components in an ordered manner.
To achieve such a goal, 2 PCB boards have been designed and built. In this section we show the list of all the electronic components and their connection schemes.

3.2.2.1 List of electronic components

As we can see in the following list, there are components that need 12V of power supply and others that need 5V. Since the used battery provides 14.8V at nominal voltage (16.8V max. and 11V min.) we need to convert the voltages and provide the system 2 different

voltage power sources. This sources are the buck converters.

- 1x Raspberry Pi 4 1Gb
- 2x Arduino Nano board
- 1x Li-ion battery 14.8V 5200mAh
- 1x Battery charger 16.8V
- 2x motor-reducer 12V 146rpm
- 2x Hall effect encoder 5V
- 1x Dual DC motor driver 12V
- 1x Lidar sensor
- 1x Lidar data/power adapter 5V
- 2x buck converter 28V-5V 3A
- 1x LCD screen I2C 5V
- 1x Push Button
- 1x Switch

3.2.2.2 Design of PCBs for motors and user interaction

From the two PCBs to connect the components, one was designed to connect the encoders and all the driver control pins to one of the arduinos Nano. This Arduino reads the encoder pulses to get the motor movement feedback, and controls the motor speed by writing a PWM for each motor connected in the motor driver. Figure 3.3 shows the designed circuit to connect and power all the cited components.

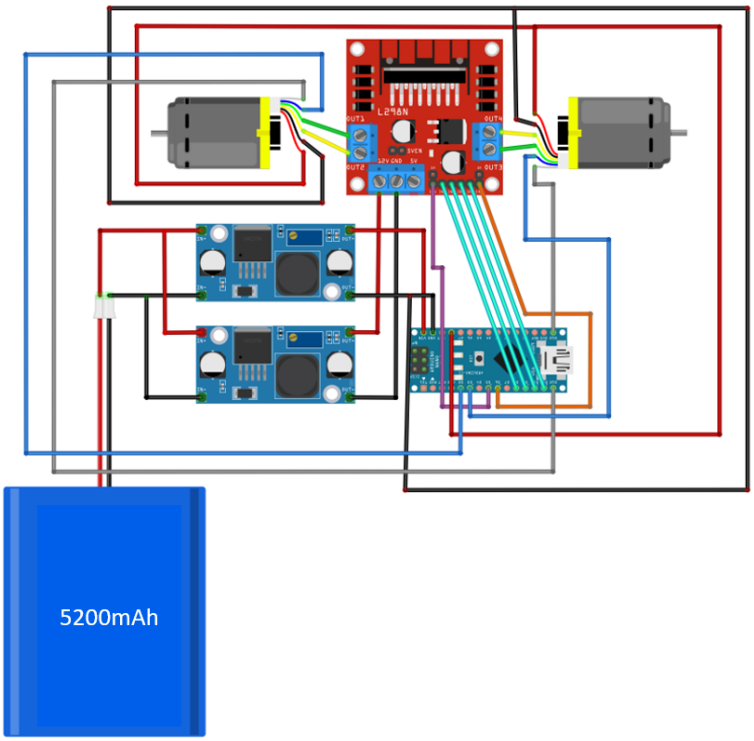


Figure 3.3: Connections of the Motors board. There are 2 motors connected to a dual driver powered by a buck converter at 12V and controlled by an Arduino powered by other buck converter at 5V.

In table 3.1 we can see how all the elements are connected to the Arduino Nano of the motors.

Table 3.1: List of connected pins Arduino of the Motors.

Arduino of the Motors			
Pin	**Connection**	**Pin**	**Connection**
2	Encoder R, channel A	9	Wheel R, back direction
3	Encoder L, channel A	10	Wheel L, front direction
5	Wheel R, enable pin	11	Wheel L, back direction
6	Wheel L, enable pin	12	Encoder R, channel B
8	Wheel R, front direction	4	Encoder L, channel B

To read the encoders pulses, 2 hardware interruptions in the Arduino Nano board were configured to trigger at any pulse change at digital pins 2 and 3. The UNO and Nano models of Arduino only support hardware interruptions in those pins.

The second implemented board was designed to act as the master. It connects the

LCD screen, a button that sends the signals to launch all the ROS packages, and the voltage divisor to get a rough reading of the state of the battery. The connection schema is shown in Figure 3.4.

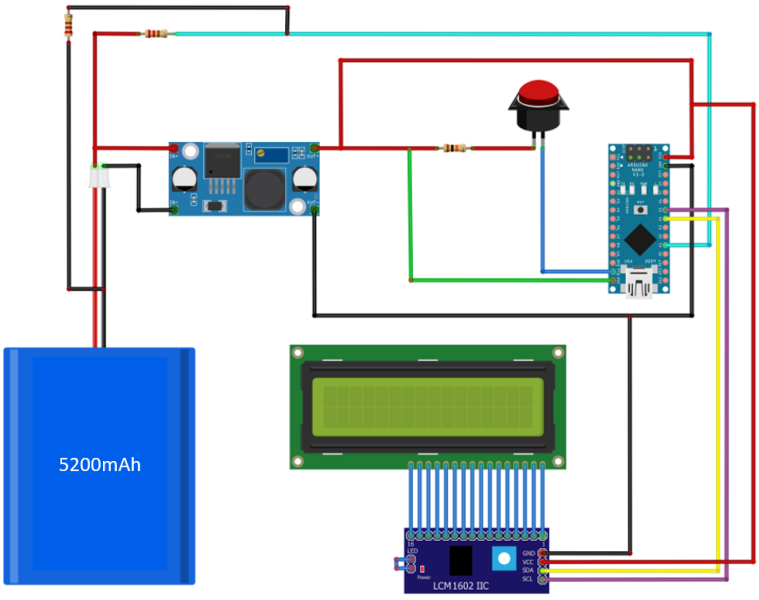


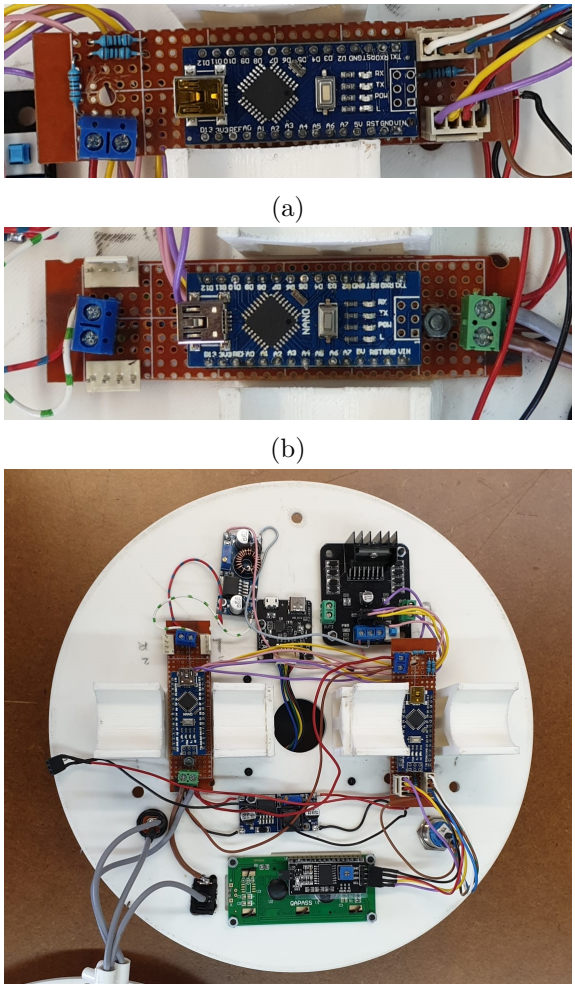
Figure 3.4: Connections of the Main board. We can see the LCD screen connected to the I2C communication module to control it easier from the Arduino. A button is also connected to Arduino. Everything is powered by the 5V buck converter.

In table 3.2 we can see how all the elements are connected to the Arduino Nano for user interaction.

Table 3.2: List of connected pins Arduino of the Motors.

Arduino for user interaction	
Pin	**Connection**
11	Button Led
12	Interactive button
A1	Voltage divisor, estimate state of charge
A4	SDA of LCD scree
A5	SCL of LCD scree

Once both boards were built and tested, all the components were installed on the robot following the distribution in Figure 3.5.



(a)

(b)

(c)

Figure 3.5: (a) Board to connect the LCD, the button and estimate the state of charge, (b) Board to connect motors and encoders, (c) Full assembly of the robot boards.

3.2.2.3 Choice of the battery

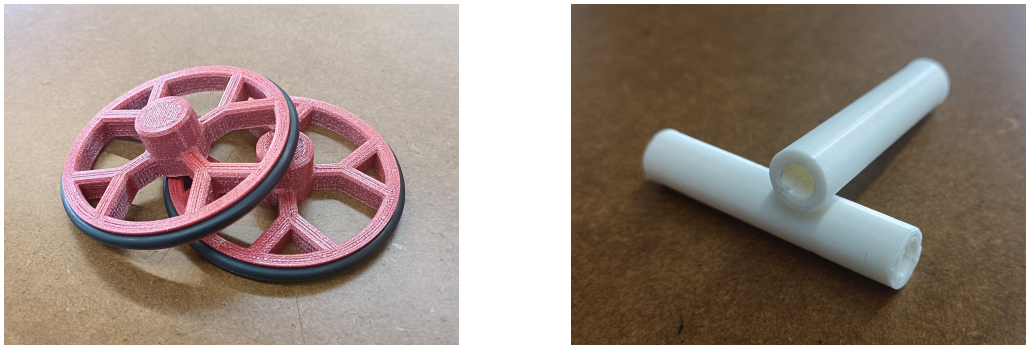
To choose the Battery capacity, we assumed to have the worst consumption case, where all components were using the maximum current. The motors would use 1A, the Raspberry Pi 1.5A, the Arduinos Nano 50mA, the LCD 55mA, and the lidar 400mA. This adds a total of 3A/h, so if we want to have a robot with 1 or 2 hours of autonomy we need minimum a capacity of 5200mAh. Another important point related to the battery is deciding which charger is the more appropriate. In our case, as we had a 4 cell battery, 2 rows in parallel, the needed voltage to charge the 4 cells at maximum is 16.8V (since every single cell max. voltage is 4.2V), so we need a 16.8V charger with 2A current (this was the recommended charging current in the cell's datasheet).

Figure 3.6: Chosen battery, 14.8V 5200mAh

3.2.3 Chassis manufacturing and assembly

Using the CAD design and the manufacturing plans, the designed parts were manufactured. The manufacturing process was done via additive manufacturing, specifically 3D printing, which fulfills our requirements for a first prototype. In the Section 2.5, there's a detailed explanation about the pros and cons of using the 3D printing technology, and why it has been chosen to obtain the robot parts. The material used in order to manufacture the pieces was the PLA, a really easy material to use in 3D printing due to its great properties.

The first pieces printed were the wheels, as we had to test the assembly of them with the motors and with the tires. Secondly, there were printed the columns of the robot, as they were too simple to manufacture. See Figure 3.7 to see the resultant pieces.

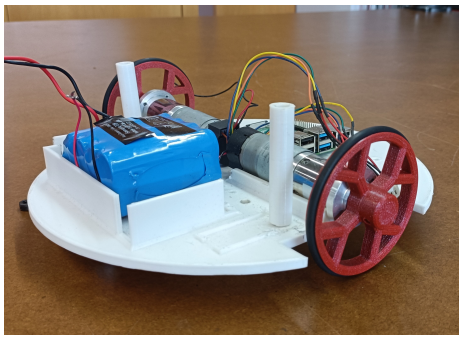


(a) (b)

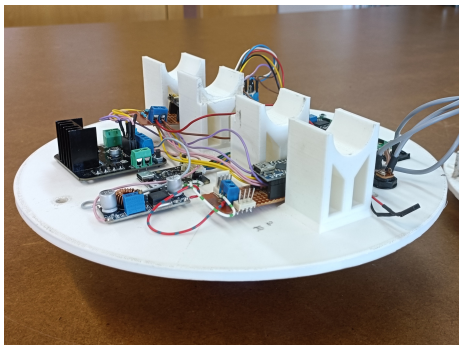
Figure 3.7: (a) 3D printed robot wheels (red PLA), (b) 3D printed robot columns (white PLA).

The next and last pieces to manufacture, were the base and the roof of the robot. It is relevant to note that due to their dimension, it was complicated to find a 3D printer able to produce them. Figure 3.8a depicts the 3D printed base with the wheels and the electrical components mounted and Figure 3.8b the robot's roof with also the electrical

components assembled.



(a)



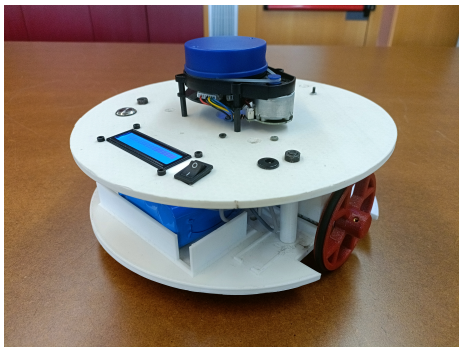
(b)

Figure 3.8: (a) 3D printed robot base (white PLA) with electrical components and wheels, (b) 3D printed robot roof (white PLA) with electrical components.

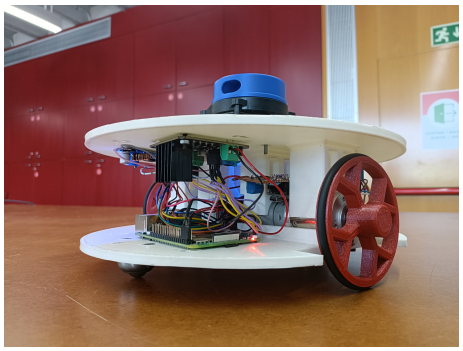
The results of the pieces were as expected and it wasn't necessary to make any superficial treatment.

Once the pieces were obtained, they were assembled together with the electronics. In order to glue the wheels to each motors shaft, it was mounted an insert in every wheel to later be attached with a screw.

Second, some electrical components were mounted on the base and the other ones on the roof, depending on the component and using the CAD assembly as a reference. Once the electrical parts were joined to the base and the roof, the assembly of the motors were located at the base, and finally, the columns were mounted on the base, along with the roof. Concluding the assembly of the robot, the crazy wheels were glued at the bottom of the base, allowing the robot to stay in static balance. Figure 3.9 depicts the resultant base and roof.



(a)



(b)

Figure 3.9: Robot assembly with all the electronic and mechanical parts.

3.2.4 Robot programming and control

In order to control the robot, we used Arduino, for low level instructions (e.g., motors control), and ROS for the high level programming (e.g., SLAM, navigation, etc.). First, there is an Arduino Nano microcontroller that controls the motors. Second, another Arduino Nano to manage the simple user-robot interface. It detects when the robot's button is pushed (e.g., to launch/kill ROS), estimates the state of the battery, and also prints all information in the LCD (e.g., IP address, State of charge, etc.). Finally, the ROS code runs on the Ubuntu Server 20.04 LTS distribution installed in the Raspberry Pi.

3.2.4.1 Arduino-based motors control and programming

The Arduino Nano of the motors has to perform 3 tasks in parallel. It needs to be reading all the time the pulses of the encoders, besides reading the serial port to see if the target speed has changed and keep that speed applying a PID control loop, which is implemented in this section. However, since the integrated microcontroller can not execute threads, we need to fake this performance.
First of all we are going to explain how we read the pulses of the encoders. Each encoder has 2 signal channels, A and B. Both A channels are attached to a interruption pin of the microcontroller, so that we make sure we do not miss any pulse. Each interruption must be connected to a callback function that is executed when an event happens in that pin. The available events are:

- LOW: to trigger the interrupt whenever the pin is low
- CHANGE: to trigger the interrupt whenever the pin changes its value
- RISING: to trigger when the pin goes from low to high,
- FALLING: for when the pin goes from high to low.

For this project we decided to get the maximum precision using the CHANGE mode. As we said before, each encoder has 2 channels: A and B. These are used to determine the direction of the rotation. The produced pulses are 90 degrees out of phase (see Fig.3.10), so, to determine the direction of the rotation we only need to compare both signals.

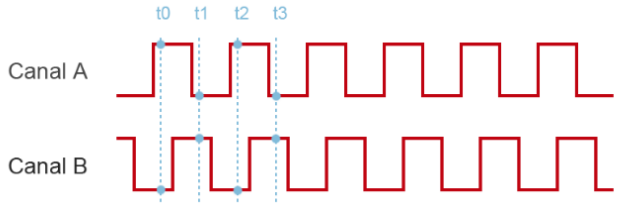


Figure 3.10: Encoder channels' pulses (A and B) and interruption triggers (t_0, t_1, t_2, t_3).

Each time the interruption fires, its callback function is attended. The main code remains blocked until this function finishes its execution, so, to do not disturb the main execution, it is necessary to perform the minimum number of tasks on it. Thus, we only increment a variable counter, but, we have to make sure the value of the variable is not modified while coping. To protect the data, what we do is to disable all the interruptions just to perform the copy and when it finishes we enable them again. We use the *Atomic* module available in *AVR microcontrollers*.

We compute the estimated velocity of each wheel using the number of pulses, and the time between updates. This information is used latter to feed the PID control loop. We get this information from the Arduino using Serial communication. As we want to read many variables, we need to define a serial communication message structure (see Fig. 3.11).

"X ; M1_pulses ; M2_pulses ; time ; Y"

(a)

"X ; M1_dir ; M1_vel ; M2_dir ; M2_vel ; Y"

(b)

Figure 3.11: (a) Send feedback message structure, (b) Receive info. message structure

We have created 2 message structures, one is used to send motion feedback from Arduino to ROS (Fig. 3.25a) and the other is used to receive messages from ROS to the Arduino (Fig. 3.25b).

Arduino's built-in serial read functions are pretty slow, that is why we have created our own read functions. When some data arrives by serial, we check if it is a 'X' character (capital X), if it is not, we ignore the value and keep checking if any 'X' arrives. When we get the 'X' we start building the message until we find a 'Y'. Then we split the message by the ';' and we get the useful information (velocities and direction) and latter it is used as consign target value in the PID controller.

In figures 3.12, 3.13, and 3.14 is shown the response graphics to a P controller, a PI controller and a PID respectively. As you can see, tuning the Kp (proportional constant) we get the 'shape' of our response (fig. 3.12), if we choose high values we get close to

instability, and else, we get very slow responses. But there is a big offset between the PV and the SP. This offset error can be reduced with the Integral of the error.

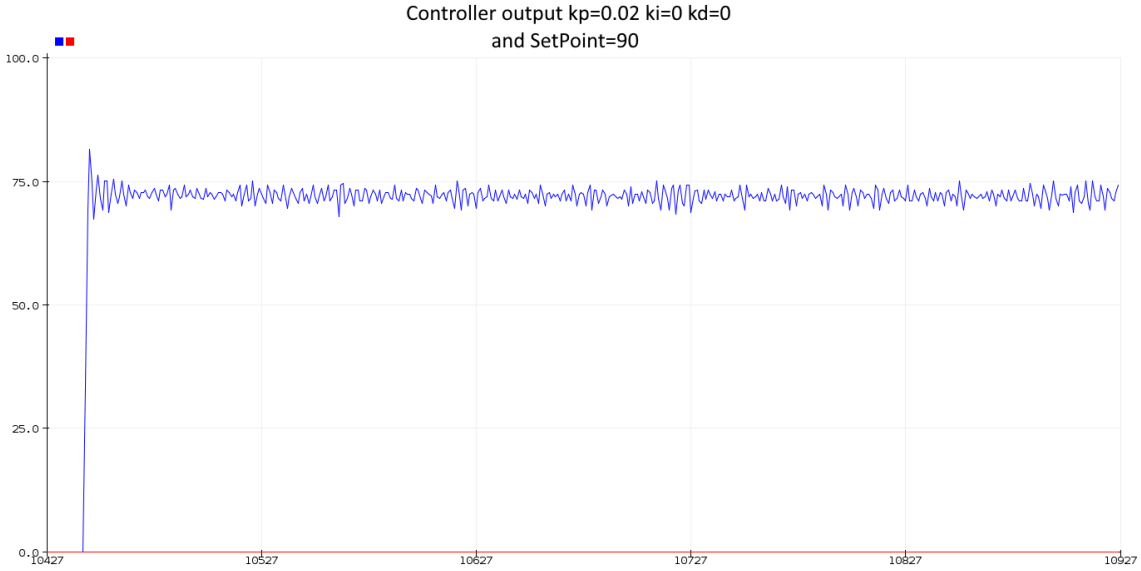


Figure 3.12: P controller.

Tuning the ki (integration constant) we can reduce the error and we can control how fast we want to do it (3.13).

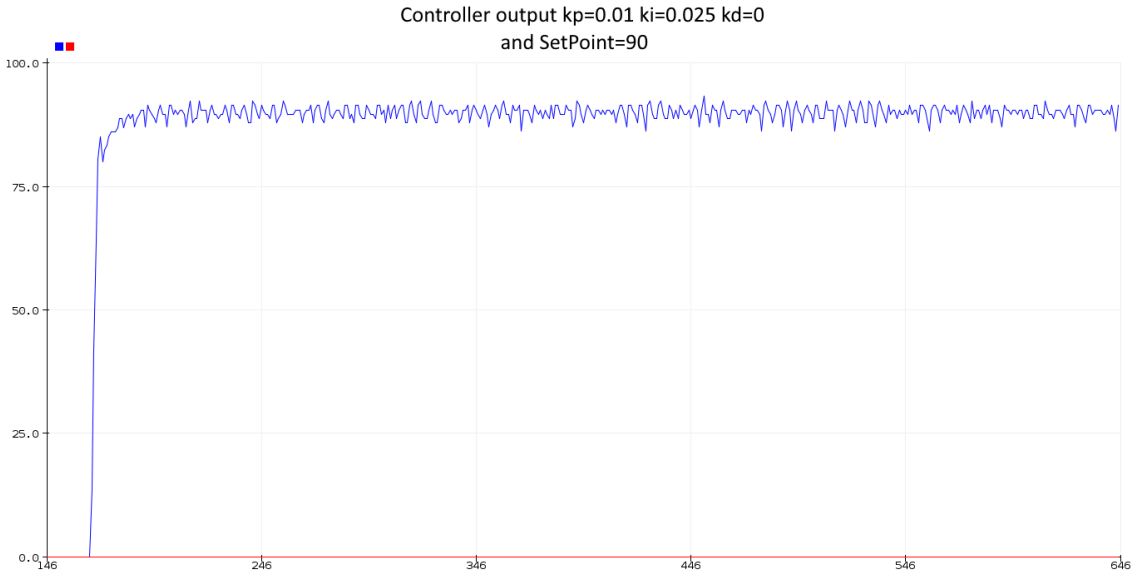


Figure 3.13: PI controller.

And finally, the Derivative part of the controller reduces the instantaneous errors (or oscillations) but it is also the most difficult part to tune. When the response is not clean

and has much noise, it can turn the it unstable very quick, and as you can see in Figure 3.14 the system turns unstable, or at least, much less precise.

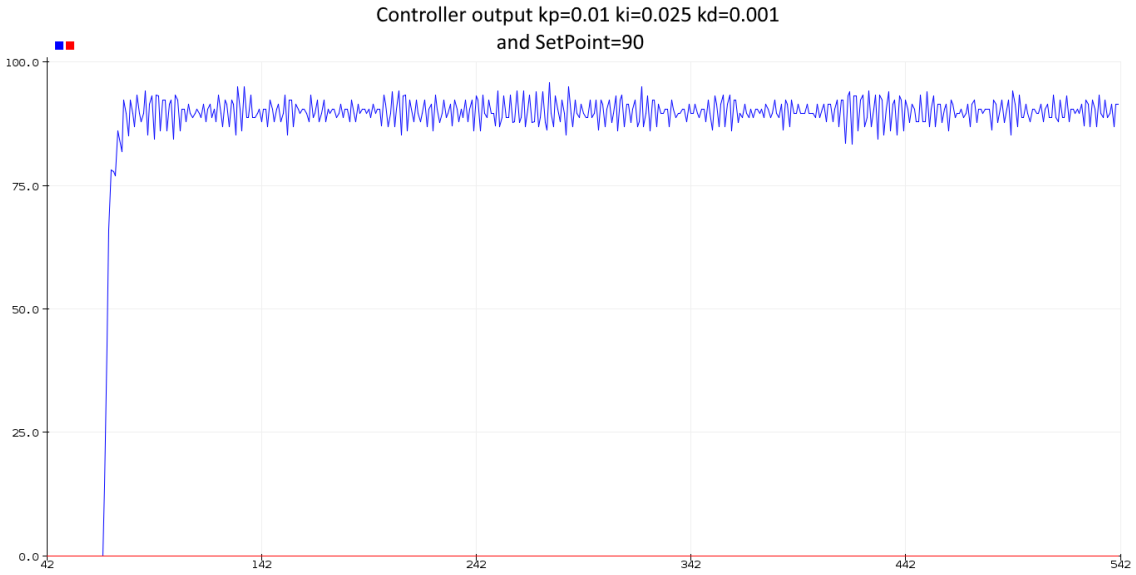


Figure 3.14: PID noisy controller.

So, finally we decide to reduce both Ki and Kd until we get a nice output (see fig. 3.15).

Figure 3.15: PID final response.

If we zoom in we can check how accurate is our controller, in this case we have approximately an accuracy of +-3.5 rpm (see. Fig. 3.16).

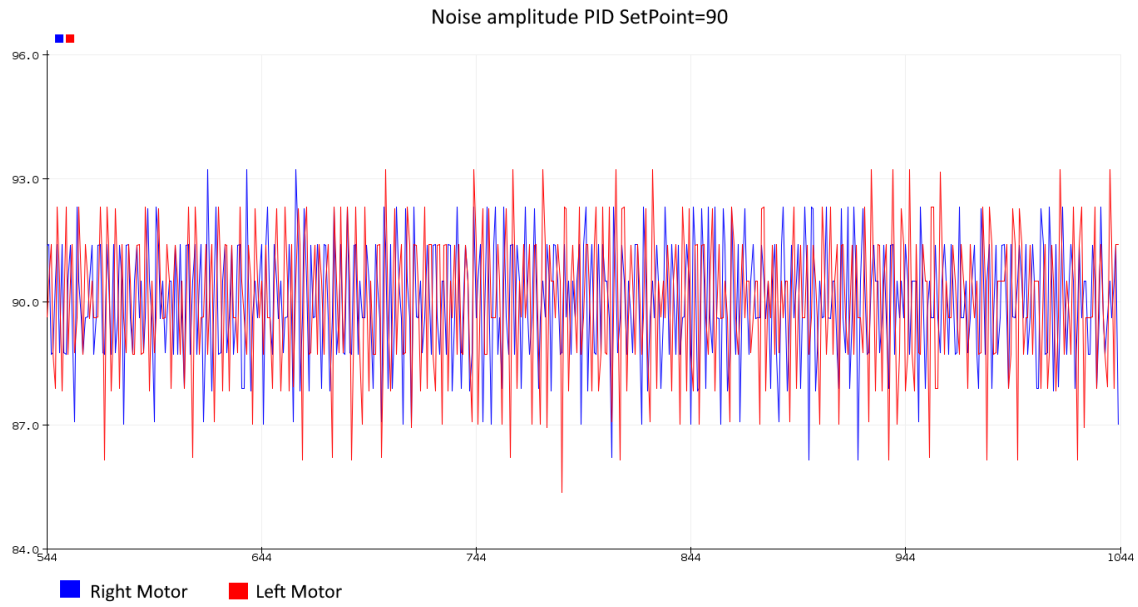


Figure 3.16: Accuracy of +-3.5rpm in PID controller.

3.2.4.2 Arduino Robot-User interface programming

The Arduino Nano used to deal with the robot-human interface offers different functionalities. On the first hand, it allows serial communication with the Raspberry Pi. It uses exactly the same protocol to read and send messages explained before (3.11). The only thing that changes is that, instead of the motors velocity, we send the *command ID* we want to run on the Raspberry Pi and, if it is necessary, we add extra attributes (e.g. the IP address when connecting to a new network).

On the other hand, it allows a robot-user interaction with the button and the LCD screen. It prints information messages to let the user know what is happening (see Fig. 3.40a). Finally it computes and updates in the LCD screen an approximated percentage of the state of charge of the battery. It is very difficult to get an exactly state of charge as the curve of discharge is different for each battery, besides it is not linear. But, since we only want to have an idea of the approximated state of charge, we suppose the battery discharges linearly.

To compute the percentage, we check the operating range of the battery from its datasheet (11V to 16.8V) and we put them into the the ADC (analog to digital converter) to found their corresponding readings. At last, the formula to get a percentage of charge and the

result was the following:

$$\%charge = (ADC.read() - 769) * 100/267$$

3.2.4.3 ROS packages and nodes structure

In this section, we explain all the ROS Packages we have developed and all nodes they contain. Bellow you can see listed all the different packages we have created or used (from other repositories) to have our robot working properly:

- **Robot description (created)**: It contains the URDF (universal robot description file) file that specifies how many joints the robot has and where they are. It has been created by us and it contains some other robot features as the mass and inertia properties, the kind of movements allowed in each joint, etc. We load this file into the ROS parameter server, and later, other packages as *TF* compute the frame transformations (e.g. transformations between the sensor frames to the base frame).

- **Robot start up (created)**: It contains the launch files that start all the necessary nodes to initialize the robot (e.g. base controller node, lidar sensor node, SLAM node). With this launch file is more easy to start all ROS nodes at once and get control of what is happening.

- **Robot teleop (created)**: It contains the node that transforms raw joystick data into Twist message (Vx, Vy, ω), allowing a human to teleoperate the robot.

- **Robot base controller (created)**: It contains the nodes to subscribe into topic that publishes the twist messages (cmd_vel). It also applies all the kinematics explained in earlier sections to generate wheels speed and write them into the motors through the Arduino Nano.

- **Node controller (created)**: It contains services that start or kill other requested ROS nodes. Actually, it is implemented to start/stop the SLAM node.

- **Node controller msgs (created)**: It contains the description of the custom services and messages used in the previous package.

- **Socket Node (created)**: It contains the socket server node that listens for requests from the App.
- **YDLidar ROS (from YDLidar repository)**: package provided by YDLidar company containing the drivers and other files to be able to read and use the lidar in ROS.
- **Joystick drivers (from ROS repositories)**: package provided by the ROS community. It contains a collection of wired and wireless controllers drivers to be able to get their data in ROS.

From the list, we can see that several nodes have been implemented in this work, in order to supply the robot all the necessary tools to perform the required tasks. In this section, we explain how these nodes are connected between them and how they interact. If you want to get into more detail in the code, it is available on Github (link in appendix D).

One of the robot's functionalities is to be controlled by a Bluetooth controller (see Fig. 3.17). This device is paired with the robot and all the data sent from the joysticks is captured by a ROS node called *Joy*, which publishes the data in the *raw_data* topic. The node *Robot teleop* subscribes to the latter topic and converts the raw data from the joysticks into velocities, using the Twist message format [3], and publishes it into the *cmd_vel* topic. Finally the *Base controller* node subscribes to the *cmd_vel* topic and writes the velocities into the Arduino, which controls the motors.

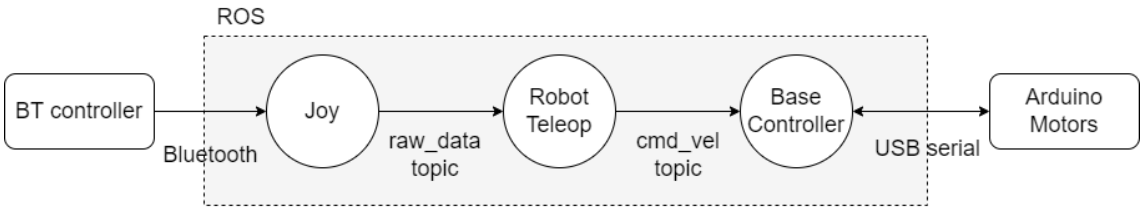


Figure 3.17: Data flow from the Bluetooth controller, through all ROS nodes, to the writing of velocities in the motors.

Another functionality of the robot is to perform SLAM. When SLAM is activated, the robot builds a map given the laser scan messages and the odometry from the encoder (see Fig. 3.18). If we want to obtain the map of a complete room, the robot must move to create it using the SLAM technique. The robot could move autonomously, selecting the best positions to build the map. However, in this case, we decided to use the Bluetooth

[3]Two vectors containing 3 axis (X,Y,Z) velocities, angular and linear

controller explained before to control the robot manually, as both nodes structures are compatible.

The socket node seen in Figure 3.18 connects the robot to the external App. Hence, this

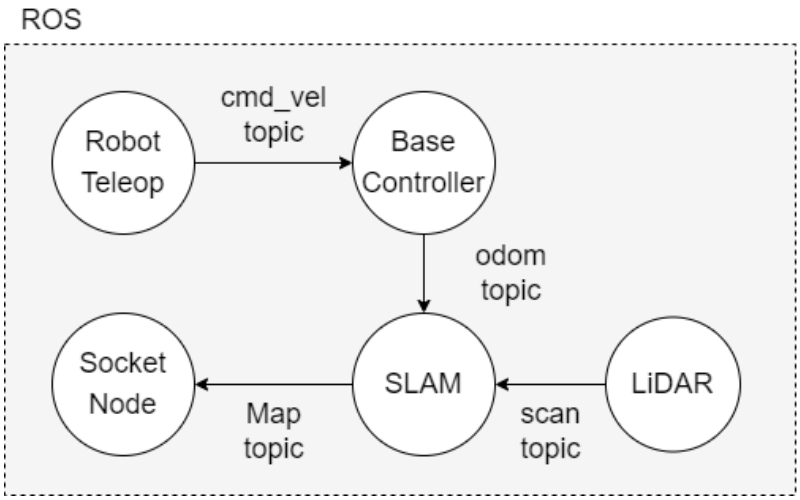


Figure 3.18: SLAM node gets information from the odometry and the lidar sensor to build a map. The robot moves thanks to the Bluetooth controller and the Socket node streams the resulting map.

node receives the users' commands: to start/stop the SLAM, to stream the Map to the App, or in future versions, to execute pre-programed routes.

The Socket node receives any the messages sent by the App, but for now the only ones that are implemented to have an effect into the robot are the Slam Start/Stop request, any other message is ignored. In the *Future work* section we will plan how the route execution could be implemented, because we can receive the coordinates of each point of the route created from the App, but there is not any implemented module to convert that information into velocities to send them to the base controller and move the robot.

3.3 Application development: design and implementation of a Graphical Interface

This section talks about the application that allows the user to interact with the robot. This app is an executable file created with Python 3 and Qt5, useful to communicate with the robot and making it execute some commands. In order to explain this section properly, it has been divided in two main parts, the framework and the design.
In Figure 3.19, is represented a diagram of the app's structure.

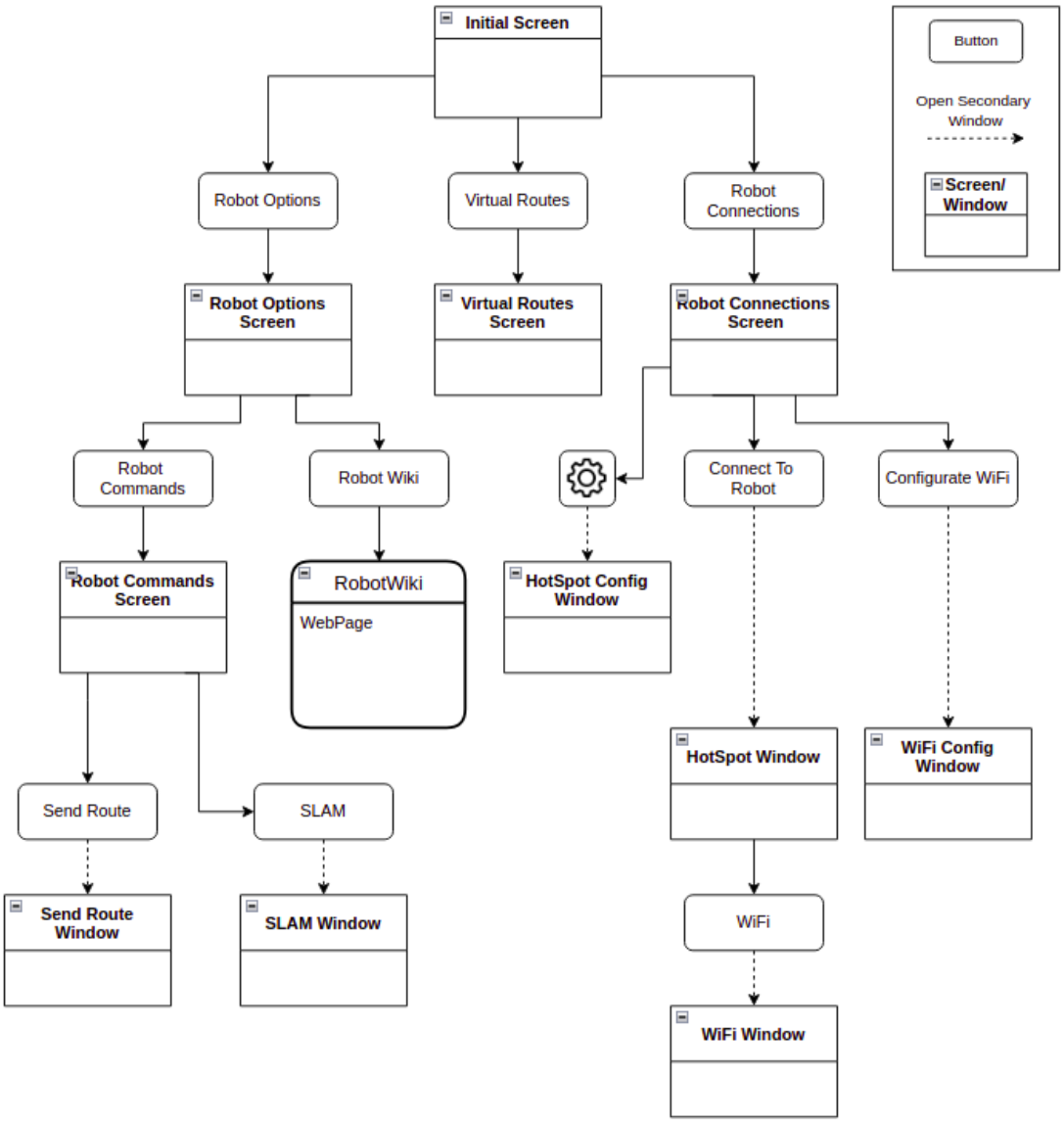


Figure 3.19: Diagram of the structure the app follows. It can be seen how it is possible to navigate through the different app's windows.

3.3.1 Application design and operation

This part is about the structure and design of the application. Here we explain the parts of the graphical interface and its functions, the graphical environment, and the functionalities and capabilities of each of its windows and widgets.

When the application is opened, it appears an initial screen consisting of three main buttons: Robot Options, Virtual Routes, and Robot Connections, as it can be seen in Figure 3.20. Those three buttons are part of the three major sections of the application. The first one, the robot options, allows the user interact with the robot and access to it's wiki. The virtual routes part, allows the user create virtual maps and routes to be

saved and later be executed for the robot. Finally, the robot connections section, has the necessary tools to connect the app with the robot and make this last one establish connection with a Wi-Fi network.

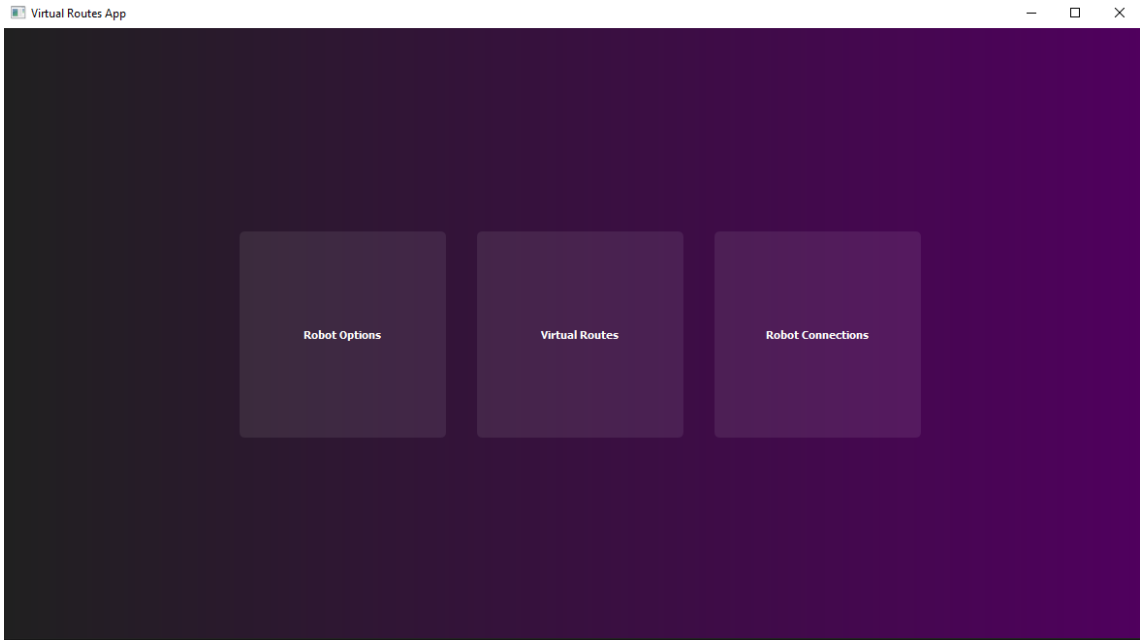


Figure 3.20: App's initial screen in which appears three buttons that allow to navigate through its three main functionalities.

By pressing one of the buttons, in addition to the widgets included in each section, it will appear a button called Menu, at the top right of the screen. By pressing it, it will automatically take you to the app's home screen.
Considering this general introduction to the three main sections of the application, let's properly explain every one of them.

3.3.1.1 Robot options

As explained before, this part is the one in which the user can interact with the robot, sending it commands. It is divided in two main parts: the Robot Commands and the Robot's Wiki. Once the Robot Options button is clicked, it redirects the user to another window with two different buttons, the Robot Commands button and the Robot Wiki button. The names of those buttons make reference to the two principal functionalities of the robot options section.
First, when the Robot Commands button is clicked, it automatically redirects the user into another window in which there are also two buttons, the Send Route button and the SLAM button. Instead, if the Robot Wiki button has been pressed, it will be opened a

web-page where will be the wiki of the robot (see Fig. 3.21).

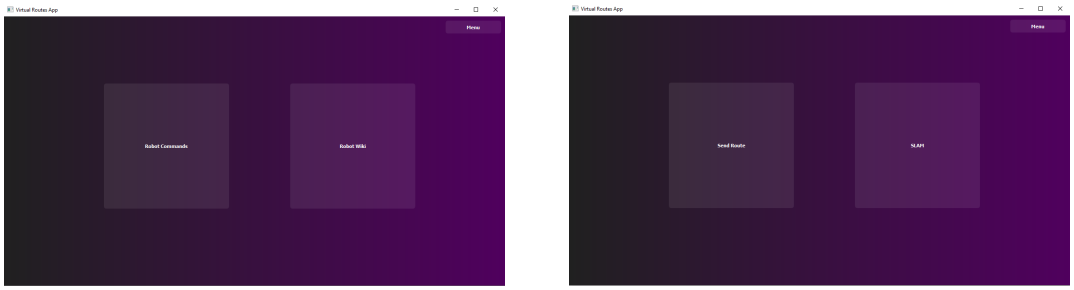


Figure 3.21: (a) Robot options window and its main widgets (Robot Commands button and Robot Wiki button), (b) Robot commands screen with its main widgets (Send Route button and SLAM button).

Returning to the Robot Commands section, if the Send Route button has been clicked, it is opened a small window with a combo box widget and a push button. The first one is a drop-down widget that, once deployed, consists of some different options to choose from. This widget allows the user choose one of the routes saved in the app and send them to the robot to be executed. See the Figure 3.22a. If the Send button of this last window has been pressed, it will be sent a message of "Initialize route" to the robot and a new window will emerge. In this window, it will be represented the map or route that the robot is performing. At the top-left side of this window, there is located a button called Stop, that allows the user send a message to the robot to make this last one stops executing the route (Fig. 3.22b).

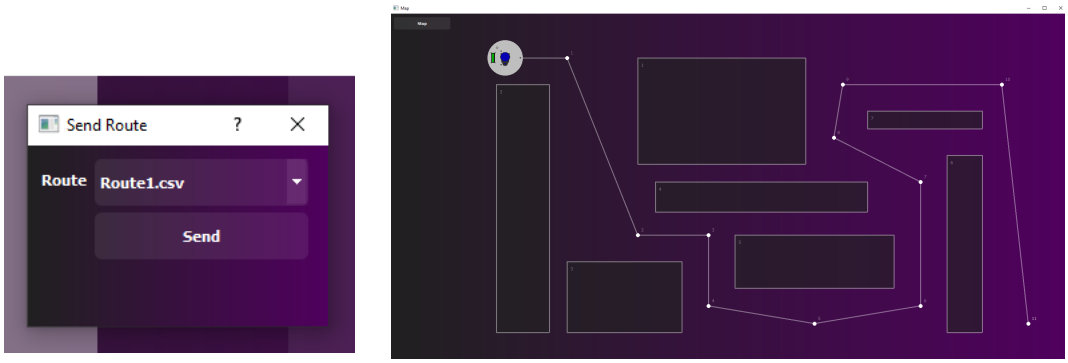


Figure 3.22: (a) Send route window with its combo box (to select the route) and its Send button, (b) Route visualizer window where is displayed the route sent.

The second button of the robot commands section, the SLAM button, if pressed, a message of starting SLAM is sent to the robot. Simultaneously, a new window will appear. In this window, is represented the image of the map that the robot generates when performing SLAM, being updated every five seconds. As in the case of the Send Route

window, there also exists a Stop button to make the robot stops performing SLAM. Figure 3.23 represents the environment of this window.

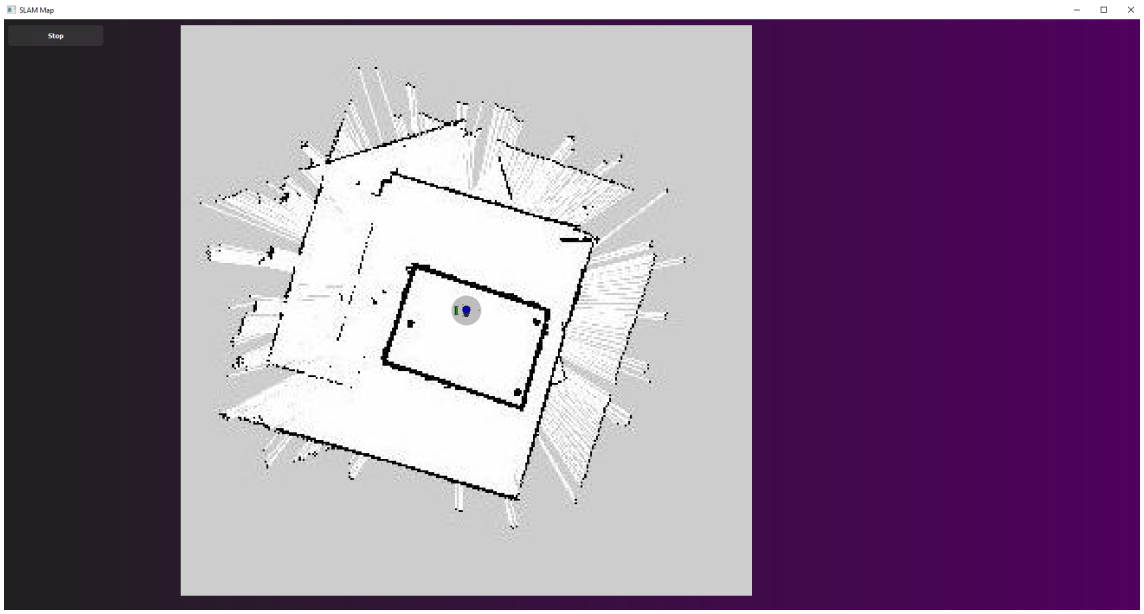


Figure 3.23: SLAM visualizer window where is represented the SLAM map that the robot computes.

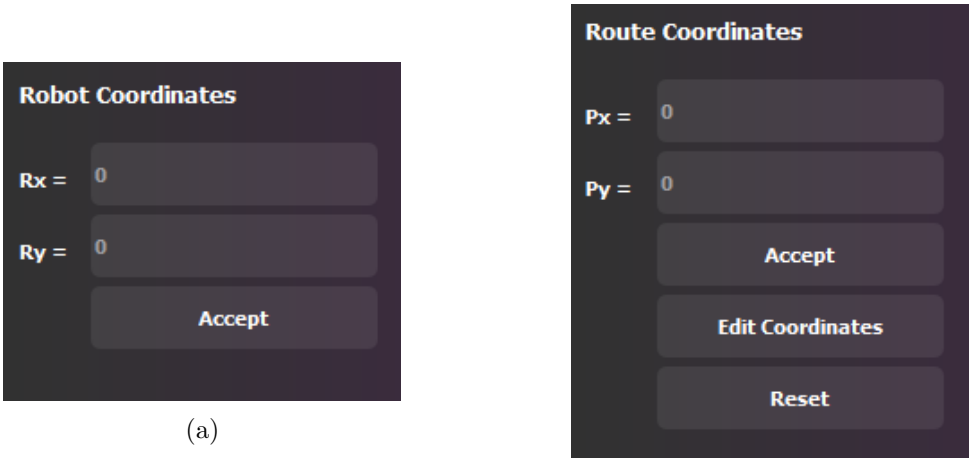
3.3.1.2 Virtual routes

In this work, a virtual route is a series of coordinates in two dimensions which together with a graphical interface, generates a sequence of points, joined by vectors, in order to create a path. This is the most complex part of the application, and its purpose is to allow the user creating virtual routes and maps (including obstacles), to then make the robot execute them.

Being in the main window, once the central button is pressed (the one named Virtual Routes), the same window is totally refreshed. When this screen appears, there can be seen three different sections: The A, B and C sections (See Fig. 3.24).

Figure 3.24: Virtual routes window with all its widgets and its differentiated sections (A,B and C). Section A, route edition part. Section B, route parameters part. Section C, Map zone where the route/map is represented.

Starting with the main part of this section, in the A section, the user can create a virtual route or map using its widgets. At the top of this section, there is a text label with the text "Robot Coordinates" written on it. Below it, there can be seen two line edits and a push button (see Fig. 3.25). The first line edit allows the user to change the initial x coordinate, in which the robot will be located at the beginning of the route. The second of them, allows to do the same but for the coordinate y. Finally, to accept these coordinates, it will be necessary to press the button below this last line edit. Once the button has been pressed, the position of widget that represents the robot, is automatically updated and the widget is moved to the coordinates inputted previously.



(a) (b)

Figure 3.25: (a) Robot Coordinates section where to edit the first robot position, (b) Route Coordinates section where to add, edit and suppress the route coordinates.

Following with the section A, there exists a section similar to the one explained above. The first thing that can be seen in this part, is a text with the words "Route Coordinates". Below the text, there are two line edits and three buttons, as it has been seen in the Figure 3.25. The two line edits will be used to generate the x and y coordinates respectively for the creation of the route. By pressing the push button below the second line edit, there will be represented, in the map section, a point and a line that unites the first robot coordinates with this new coordinate. If more coordinates are being inputted and accepted, more points will be appearing in the map section and these ones will be united (by a line) with the one created before it, generating a virtual route as the one in the Figure 3.26.

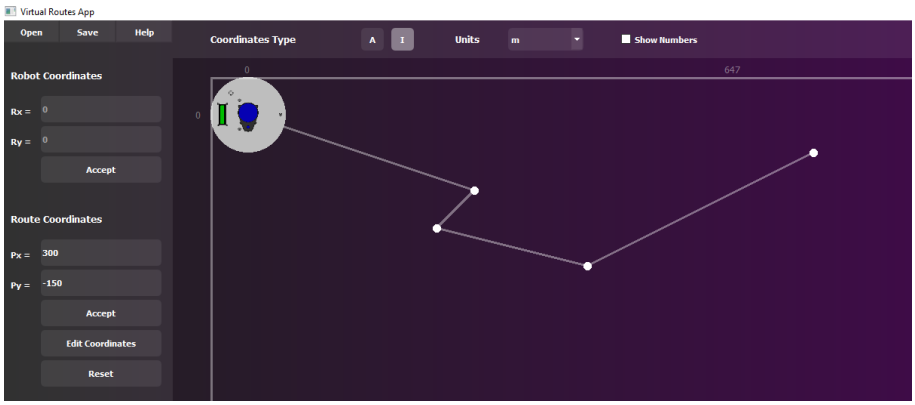


Figure 3.26: Virtual route representation example.

Despite having some resemblance to the previous section, this one is composed by two more push buttons. The first one of them is named Edit Coordinates, and its utility is to allow the user change and edit the coordinates of the route. The second one, if pressed,

the route created is erased.

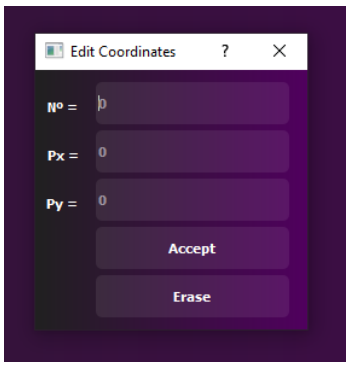


Figure 3.27: Edit coordinates window, where to edit or erase the route coordinates.

By pressing the button to edit coordinates opens a small window with three line edits and two push buttons. To see the distribution of the window, please, see the Figure 3.27. The first line edit is where the user has to input the number of the coordinate that they want to edit. The following ones, are useful to input the new x and y coordinates respectively. If the Accept button of this secondary window is pressed, the coordinates will be changed, however, if the Erase button is clicked, the coordinate in the position inputted in the first line edit will disappear from the route, and the points before and after this one will be united.

It has to be noticed that there have been created some warning windows, in order to don't collapse or block the app. One of these windows appears when a coordinate is out of range of the map area, another one emerges when, in the section of editing a coordinate, the user inputs a non existent one. Finally, the last warning window pops up when the user wants to edit the coordinate number 0, which has to be edited by the Robot Coordinates' line edits.

Following with the A section, there can be found a button called Simulate, another one called Reset Simulation and on its bottom, there can be seen three small buttons called x1, x10 and x100 from left to right respectively (see Fig. 3.28a). As the name of the first one of the buttons indicates, it is used to start a simulation of the robot's displacement over the route. This simulation will depend on the speed of the robot and the distance it has to travel to reach each of the coordinates of the designed route. The Reset Simulation button, will obviously reset the simulation, moving the robot to its initial position.

The last three buttons of this simulation part, allows the user make the simulation goes faster. By pushing the x10 button, the simulation will be represented ten times faster than with the normal speed (the initialized x1 button by default), and with the x100 button the simulation will go 100 times faster.

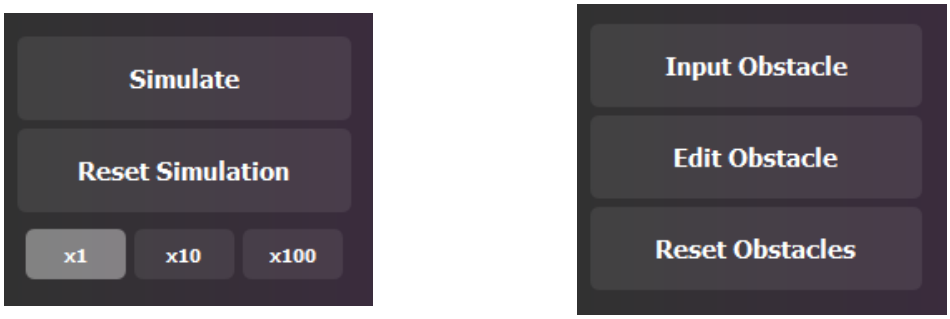


Figure 3.28: (a)Route simulation section with its widgets, (b)Route obstacles section, where to add, edit or remove obstacles from the map zone.

The last section of the A section, is based on three buttons that allows the user create, edit and remove obstacles from the map zone (see Fig. 3.28b). When the Input Obstacle button is pressed, a small secondary window emerges, as in the case of edit coordinates. This small window, is composed by four line edits and a push button as it can be seen in the Figure 3.29a. The first and second the second line edits, allows the user to input the x and y coordinates of the obstacle respectively. However, the other two buttons are there to be filled with the width and the height of the obstacle. If the button located at the bottom of the window is pressed (the Accept button), an obstacle with the parameters assigned to it, will appear on the map zone.

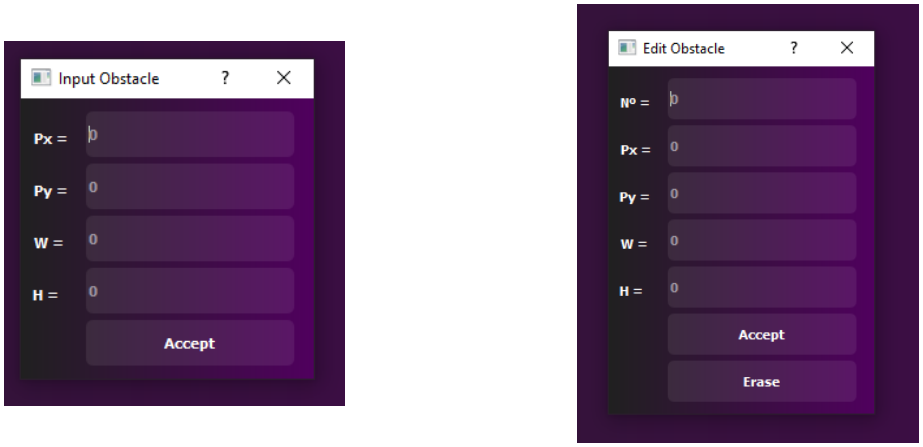


Figure 3.29: (a)Input obstacle window with its widgets to input the coordinates and dimensions of the obstacle, (b)Edit obstacle window, where to edit or erase the route obstacles.

Returning to the A section, the second of the buttons that can be found in the obstacles part, is the one called Edit Obstacle, which when pressed, it emerges a window with the same format of the Input Obstacles one, but with another line edit above the x coordinate one, and a push button named Erase down the Accept button, as it has been seen in the Figure 3.29b. This new line edit lets the user input the number of the obstacle to edit,

and the Erase button, when clicked, the obstacle on the position inputted is removed from the map.
The last button of this part, is called Remove Obstacles and, by clicking it, all of the obstacles in the map are erased.

Starting with the second part of the virtual routes window, the B section, on the top-left of it, there are three buttons (see Fig. 3.30). The firs one (the Open button), when pressed, a directory of the device where the app is executing is opened. In this directory, there appears all the routes that the user has created and saved. Once there, the user will be able to open the route and, automatically, it will be represented in the area of the map being editable and with the parameters that was saved. The push button next to it, by clicking it, the same directory window will emerge and instead of opening a route, the user will be able to save the route that has been designed. Ending with these three push buttons, the last one is called Help, and it opens a text file with some instructions on the virtual routes screen once it has been pressed.

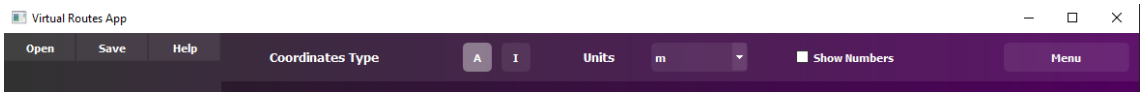


Figure 3.30: Route parameters menu with its widgets.

Following with this section there can be found a the text "Coordinates Type", followed by two buttons, the "A" button and the "I". Since the first one is pressed, the coordinates that the user inputs when setting the initial position of the robot, the ones to create and editing the route, the coordinates of editing the obstacle and its width and height will be absolute coordinates. In the case of pressing the "I" button, these coordinates will become of incremental type.
At the right side of the "I" button, there appears a label with the text Units in it, and next to the text, there is a combo box, that when being deployed, users can choose the units in which they want the route to be made, meters (m), decimeters (dm), centimeters (cm) and millimeters (mm).
To conclude this big section, at the right side of the previously explained combo box, there is situated a checkbox that if checked, there appears the numbers of each coordinates and obstacles to make it easier for the user to edit or erase the coordinates and obstacles.

3.3.1.3 Robot connections

This is the last one of the three main sections of the application. Its main functionality is to allow the user to connect via socket TCP/IP to the robot in order to send messages and commands to it. This socket connection is bidirectional, as the application can send commands to the robot, and this last one can also respond and send messages to the application (e.g., the environment map). With this kind of communication, it can be proved that the sent commands or messages have arrived to their destiny.
By pressing the third of the main buttons, the Robot Connections button, the application redirects the user to a screen where two buttons appear (Fig. 3.31). The Connect to Robot button, allows the user to connect to the robot via Hot-Spot or Wi-Fi. The second one, however, is called Configure Wi-Fi and allows the user to send the SSID and password of a close Wi-Fi network to the robot.

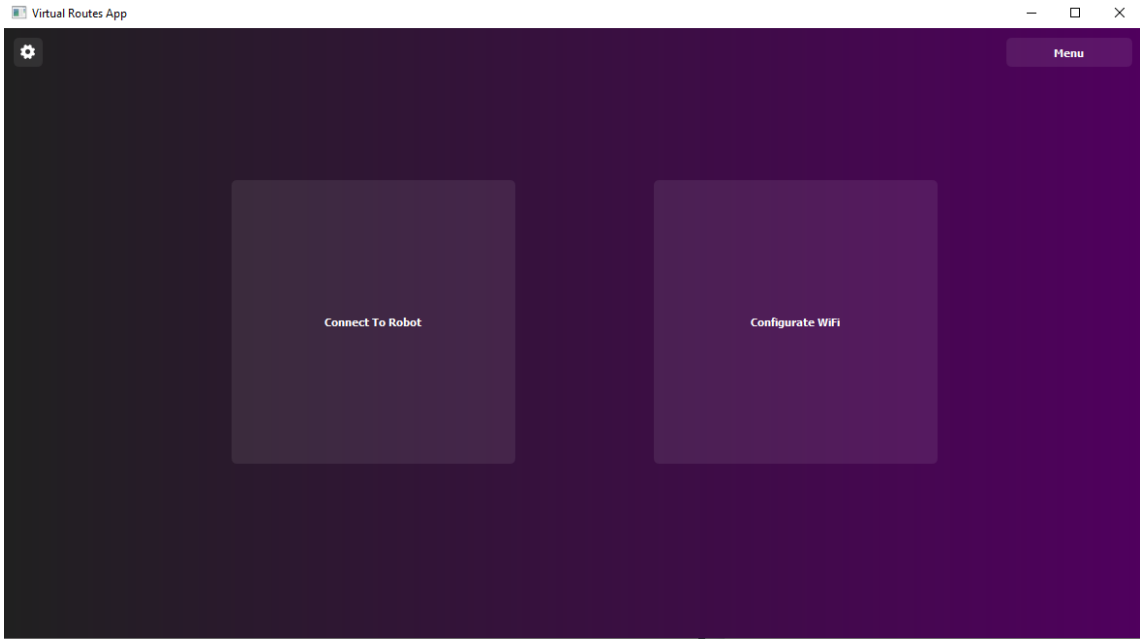


Figure 3.31: Robot connections window with its main widgets (Connect to Robot and Configure Wi-Fi buttons) and its secondary ones (HotSpot configuration and Menú buttons).

Once the Connect to Robot button pressed, it appears a small secondary window named Hot-Spot with three button (Connect, Disconnect and Wi-Fi), and the following information: IP, SSID, port and ROS port of the robot's Hot-Spot. It can be seen a representation of this window in the Figure 3.32. If the user presses the connect button, the app will try to connect to the robot (the server) using the IP and the port that appears in the screen, appearing a warning window if the connection has failed or appearing an

information window if the the connection has been successful.

When explaining the contents of the HotSpot window, there has been talked of a button called Wi-Fi. By clicking this button, the same window changes its content and name (Wi-Fi window) and there can be seen a button on the top right called Hot-Spot, a line edit and finally the two buttons connect and disconnect from the previous screen. The button named Hot-Spot has the same function as the previously explained Wi-Fi button, but instead of directing the user to the Wi-Fi window, it redirects him to the Hot-Spot window. The line edit widget allows the user to enter an IP address to be sent to the robot via the connect button, and finally establish connection with the robot once it has been connected to a Wi-Fi network. The disconnect button allows the user to disconnect from the robot as explained above. But which IP the user has to write in the line edit? Once the robot connected to the Wi-Fi, it displays by its own LCD screen an IP, this IP is the one that has to be written in the line edit.

Figure 3.32: (a)HotSpot secondary window where to connect to the robot via HotSpot, (b)Wi-Fi secondary window where to connect to the robot via Wi-Fi.

In order to connect the robot to a Wi-Fi network, there must be sent the Wi-Fi credentials to the it. If the Configure Wi-Fi button seen in the Figure 3.31 clicked, it emerges a secondary window which is composed by two line edits and a button called Send (see Fig. 3.33). Talking about the line edits, the one on the top of the window, lets the user input the SSID of a close Wi-Fi network, and the one down this previous one, has to be used to enter the password corresponding to the Wi-Fi network with the SSID written in the line edit above. Once both line edit full, if the user press the send button, the SSID and the password written, are sent to the robot to make him connect to a Wi-Fi network.

Finally, in order to conclude this section, there exists a button located at the top-left of the window, in which appears a white gear. This button, when pressed, opens a secondary window where the user can change some parameters of this section as the Hot-Spot IP

of the robot, its SSID and also its ports, the normal server port and the ROS port (Fig. 3.33b). This parameters will also be changed automatically in the Hot-Spot window.

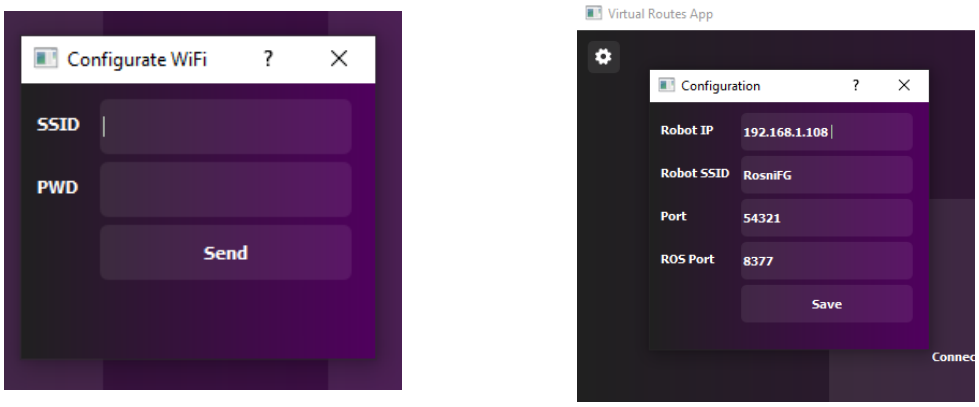


Figure 3.33: (a) Configure Wi-Fi window with its widgets to send the Wi-Fi credentials, (b) HotSpot parameters configuration window where to modify some HotSpot configuration.

3.3.2 Programming and Development

In this section, we explain the programming details of the application. In the Section 2.6, it is provided a brief explanation of the platform used for the programming of the application, PyQt5.

The most fundamental element of the applications programmed with this platform are the windows and the widgets. A window is an element that allows the user input widgets in it. The widgets, can be connected to Python functions in order to compute them. PyQt5 has its own widgets as, the QPushButton, the QLabel, the QLineEdit, and more. In order to create one, it's necessary to use these kinds of functions. To create different widgets and learn about their properties and functionalities, a good method is by accessing to the Qt Documentation web-page [3]

To have an idea of the main PyQt5 functions and commands used to create the app, in the Appendix B there's a table with the name of the function/command and its its utility (Tab. B.1).

3.4 Robot-app communication and integration schema

3.4.1 Robot-app communication: TCP/IP sockets

In order to connect the application with the robot, it has been decided to create a TCP/IP socket system based on the client and server approach. The socket server runs on the robot and the application acts as a client. Below, there's a detailed explanation of the socket

structure and the communication protocol. It has been chosen this protocol because as it has been seen in the Section 2.7, the TCP protocol proves if the message has been received successfully and, in addition, Python has a great TCP/IP socket library.

3.4.1.1 TCP/IP socket server

First of all, it is worthy to highlight that the robot can process two types of commands: the *python commands* or also called *system commands*, and the *ROS commands*. The *system commands* are the ones related with the network connections of the robot (e.g., Wi-Fi configuration). However, the *ROS commands* are those commands that allow the robot move (e.g., SLAM). Some of the commands (e.g., Stop SLAM) could block the server execution, producing some clients' request losses. Hence, it was decided to create two different servers in the robot, so that a single server did not have to run more than one simultaneous process Both servers were implemented using the same method and the same platform, Python. The unique difference is that the ROS server was implemented as a ROS node. This allowed that the commands sent to the ROS server, could be executed directly by ROS, without having to carry out a bridge to communicate the robot's system with ROS.

3.4.1.2 TCP/IP socket client

As explained above, relating some concepts, it's easy to get to the conclusion that if there exist two, in order to establish connection with both simultaneously, there are needed a minimum of two clients. Thus, there were created two clients, the system client, which send the commands to the system server and the ROS client, which send the commands to the ROS server. In this way, the clients were introduced to the application, so that, when the robot was launched, the application would be able to connect to it using two different TCP/IP processes, one before the other, and being able to send the respective commands to the ROS or system servers.

3.4.1.3 TCP/IP messages structure

Once a server and a client have established a connection, they must communicate with each-other by transferring messages bidirectionally. In order to ensure that a server and a client understand each other, the messages shall follow a structure.
The process of messages' transference starts with the construction of the message. It is counted the length of the message and it is inputted as a the header of the message (see

Fig. 3.34). This header has a maximum length of ten bits, so the maximum length of the message to send, can not reach the ten thousand millions of characters. Characters so if the number of bits that compose the length of the message is lower than ten, zeros are added before the length of the message in order to reach ten bits. Finally, the message is encoded with *ASCII* and transferred via a TCP/IP socket connection.

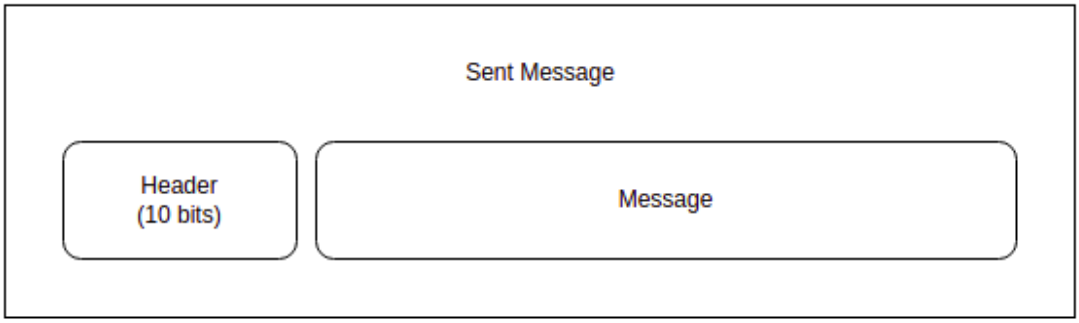


Figure 3.34: Structure of the defined messages for the Socket TCP/IP communication.

In order to make the app receive and read the SLAM image, we first read ten bits, the header. With the header, the receiver knows the length of the message that has been sent and it starts reading it. The message is read in chunks of ten bits and constructed by concatenating a newly received chunk to the previous ones (see Fig. 3.35). Finally this message is decoded for later manipulation.

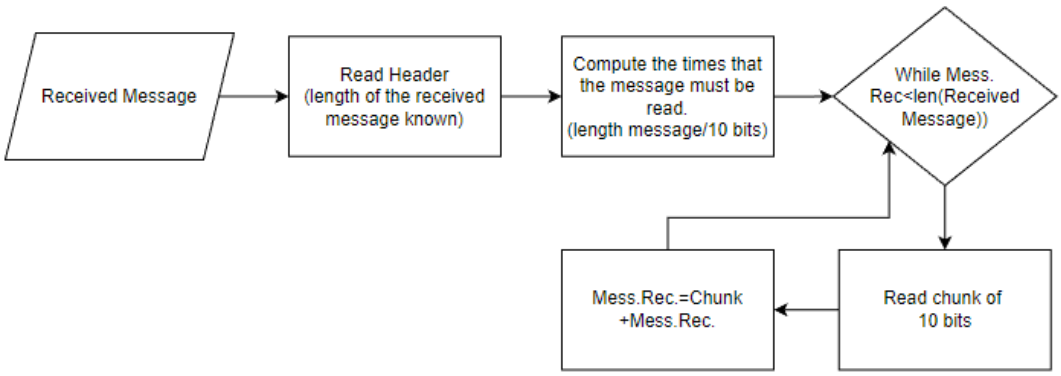


Figure 3.35: Diagram of the protocol followed to receive a message (Socket TCP/IP).

3.4.2 Robot-app integration schema

This section shows how all the different elements of the product (Robot+App) are connected between them, in other words, a integration schema.

As shown in Figure 3.36, the robot and the application are two independent elements

that communicate between them using TCP/IP Sockets. There are two Socket servers in the robot: one of them directly running on a Python script named *god.py* (launched at boot), and another one running in a ROS node. Later in this section, we explain some details about the script *god.py*.

When users want to execute or use some ROS functionalities (e.g. do SLAM, execute routes), the app allows to send a request from its *ROS socket client*. If they want to set up the robot (e.g. change the Wi-Fi network) the app sends the request from the *System socket client*.

Once the requests are received by the robot, they are managed as it is explained in Section *3.2.4 Programming*.

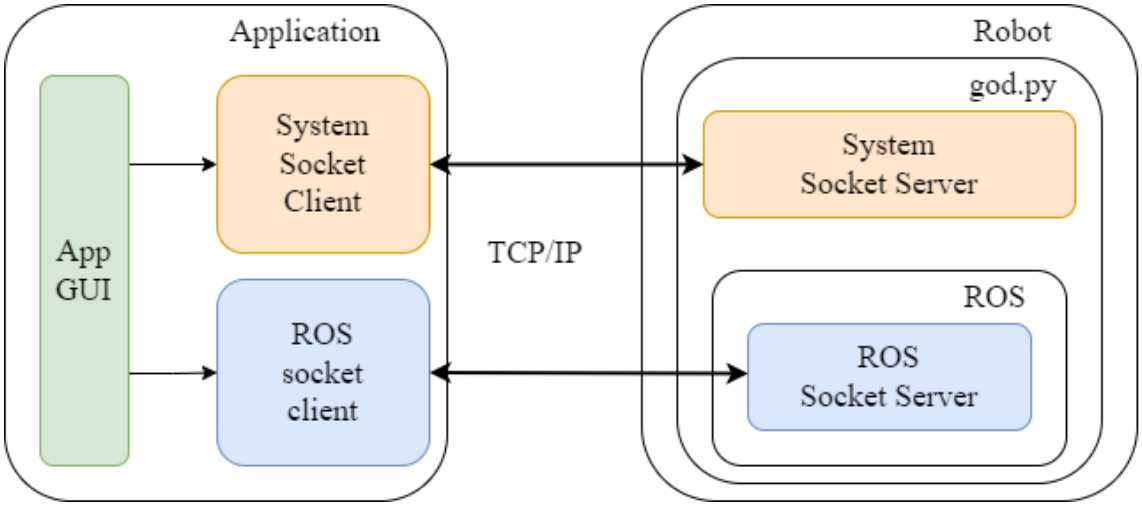


Figure 3.36: App and robot integration. On the left is the App with its socket clients and on the right the Robot and its socket servers, one inside ROS and the other in the god.py script.

As we said before, there is a script running on the robot that handles all the user requests, *god.py*. This script starts running when booting the robot's Raspberry Pi operating system (Ubuntu), and it opens communication with the Arduino Master. It also reads the *command ID* from the Arduino of robot-user interface (e.g. start ROS, kill ROS) and executes it.

The *god.py* starts a System Socket Server and, when the user starts ROS, the ROS Socket Server is started. To connect the app to the robot, both Socket Servers must be running, if not, connection will be refused.

The two App socket clients, connect to the robot socket servers. All ROS requests are attended by the ROS server, and all system requests are attended by the system server. If the user requests to connect to another network, the *god.py* must disconnect from the

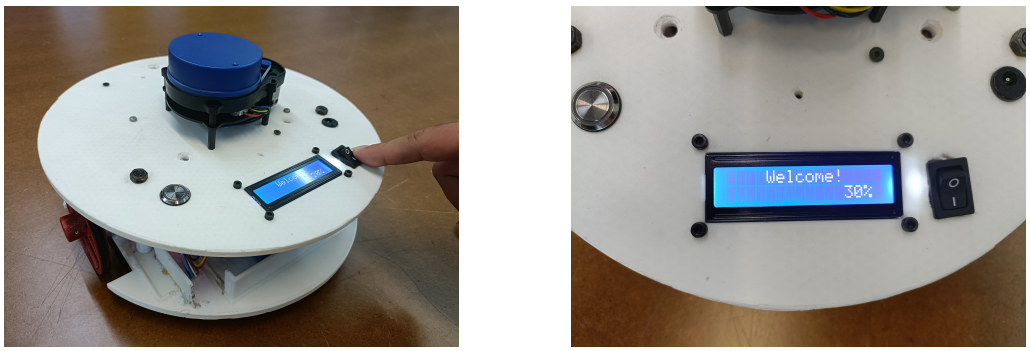
App, kill all the socket processes and run some *bash* commands to connect to the new network, and if it connects successfully, then it launches the two socket servers again with its new IP address.

3.5 Use cases

In this section we explain how the robot plus the app can be used in two different real cases as a whole. First, we describe how to start up and configure the networking of the robot using the app. Second, how to make the robot perform SLAM via the app.

3.5.1 Robot network setup

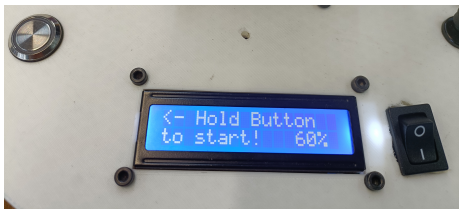
The first step to carry out the network setup is turning on the robot. In order to do this, it is necessary to press the switch located at the right side of the LCD. The LCD will be turned on and it will show the message 'Welcome!' and the percentage of the battery's energy (see Fig. 3.37).



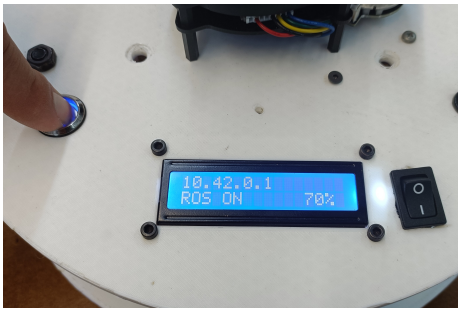
(a) (b)

Figure 3.37: (a) Switch pressed and robot turned on, (b) Representation of the LCD at initializing the robot.

Later, once the robot's system is completely working, in the LCD screen, will appear the following message: '$<-$ Hold Button to start!'. If the push button at the left of the LCD is pressed during two seconds, the message of the LCD screen changes. Specifically it shows the IP of the HotSpot of the robot and a message of 'ROS OFF', and the led of the push button starts to make a flashing light. Then, the system server of the robot is initialized and waiting for connections. If the push button is again pressed for eight seconds, the message of the LCD screen changes to 'ROS ON', indicating that the ROS system of the robot is initialized (see Figure 3.38).



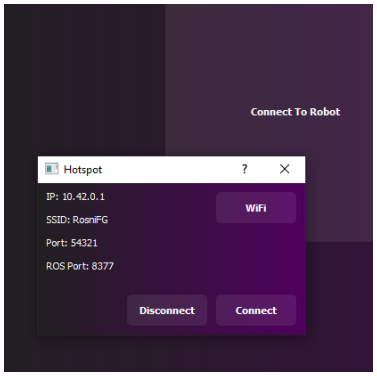
(a)



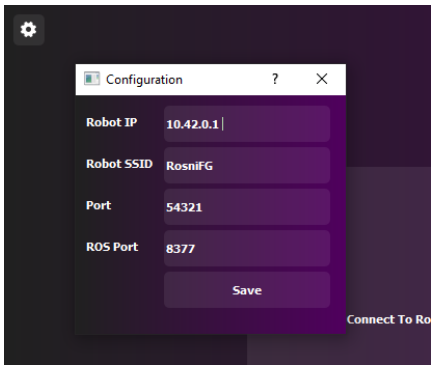
(b)

Figure 3.38: (a) Representation of the LCD once the robot's system completely operative, (b) Representation of the LCD when ROS has been initialized.

Now, the app is able to establish a connection with the robot. First of all, the app must be opened and the user has to press the Connections button of the app. While on the connections screen, if the Connect To Robot button has been pressed, a secondary screen will appear. An IP will be displayed on this screen, and it must be checked that it matches with the one of the robot. If it does not match, it will be needed to access the HotSpot settings screen of the app, by pressing the button at the top left of the Connections screen and change the IP for the one that appears on the robot's LCD (see Fig. 3.39). In order to establish a connection between the app and the robot, it must be pressed the Connect button of the secondary screen.



(a)

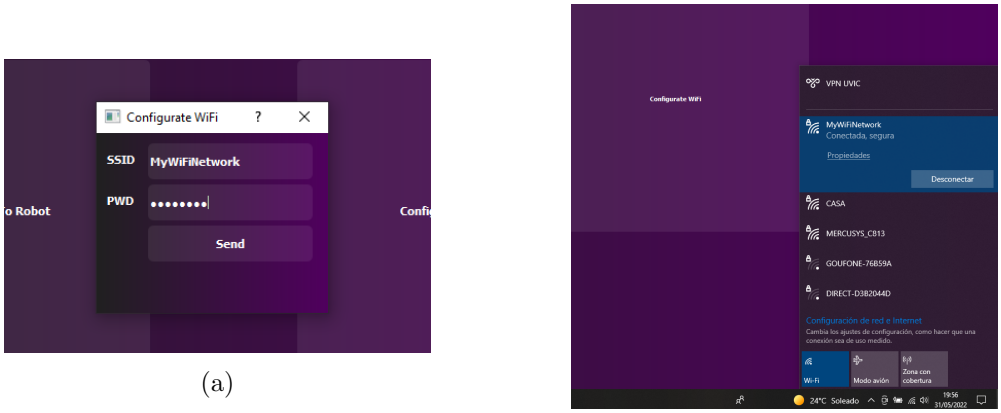


(b)

Figure 3.39: (a) Hotspot window with the same IP as the one in the robot's LCD, (b) Hotspot configuration window to change the HotSpot parameters if needed.

Once the robot and the application are connected using the robot's HotSpot, it is possible to establish this connection via Wi-Fi. When the Configure Wi-Fi button pressed, it emerges a secondary screen. In this screen, a user can put the SSID and the password of the Wi-Fi to which they want to establish the connection, and press the Send button to send the credentials to the robot. If the credentials are successfully sent, the app is

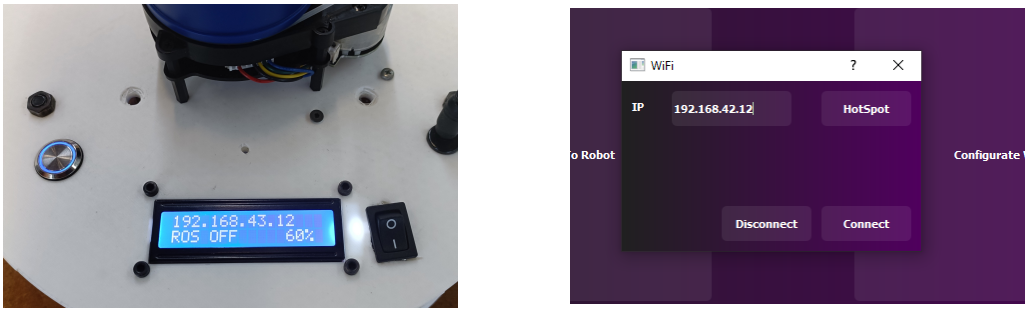
disconnected from the robot, while the robot tries to connect to the Wi-Fi network. Then, the device in which the application is running, has to connect to the Wi-Fi to which the robot is connected. In Figure 3.40 it can be seen that the SSID of the Wi-Fi sent to the robot is the same to the one that the device is connected to.



(a) (b)

Figure 3.40: (a) Wi-Fi configuration window with the SSID and password of the Wi-Fi network to establish communication with, (b) Device connected to the same Wi-Fi as the one sent to the robot.

Finally, returning to the Robot Connections screen and accessing to the Hotspot secondary window, if the button Wi-Fi is pressed, this screen will change, indicating the same IP that appears in the LCD screen of the robot (see Fig. 3.41), and by pressing the Connect button, the app will try to establish connection with the robot.



(a) (b)

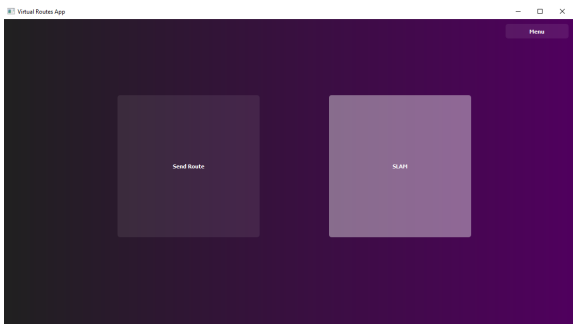
Figure 3.41: (a) LCD with the new IP of the robot, (b) Wi-Fi window with the new robot's IP.

3.5.2 Map creation using SLAM

In order to allow the robot to perform SLAM, with the ROS system operational and with the LCD showing the message 'ROS ON', the user must connect the wireless controller to the robot and access to the Robot Commands screen of the app to press the SLAM

button (see Fig. 3.42). The wireless controller is necessary to make the robot to move around the room, exploring it with the lidar and creating a more extended map.

(a)

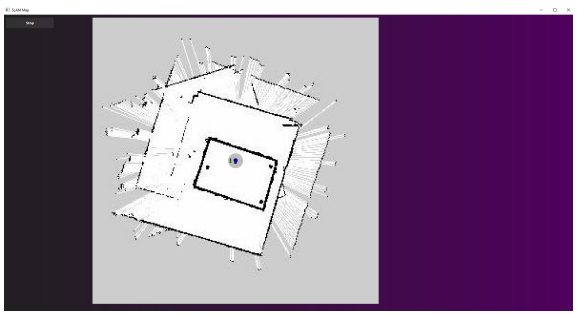


(b)

Figure 3.42: (a) Wireless controller connected in order to make the robot move, (b) SLAM button of the Robot Commands screen to make the robot start performing SLAM.

When the SLAM button has been pressed, it is sent a message of 'Initialize SLAM' to the robot (ROS server), and the robot starts to create a map and send it to the application. Via the app, it can be seen the lively representation of the map that the robot is creating (see Fig. 3.43).

(a)



(b)

Figure 3.43: (a) Robot performing SLAM and being teleoperated, (b) App screen with the lively representation of the SLAM's output map.

Chapter 4

Business Plan: MCubic

4.1 Argumentation and Prototype Development

4.1.1 Idea

The main idea of the company MCubic is to introduce the learning of ROS in Catalan and later also Spanish academic centers. In order to do this, a product called ROSNI has been created which is a mobile robot capable of performing SLAM from the ROS platform. The purpose of this robot is to be able to be programmed from the ROS platform by the user and thus achieve a dynamic learning. With the intention of introducing the dynamic learning, it has also been created a wiki of the robot in which can be found some ROS commands to take the most advantage of the robot capabilities. To complement the robot it has been created a graphical interface to interact with it and being able to move the robot, design virtual routes for it to follow, make it perform SLAM and connect the robot to a Wi-Fi network.

4.1.2 Why?

Increasingly, mobile robotics is gaining more weight in both industry and home ownership. Seeing that robotics is growing exponentially over time and that society is becoming more and more interested in domestic and industrial robotics, it has been thought to create a dynamic learning system based on programming a robot through the ROS platform, as it is currently the most optimal and well-known platform for robot programming.
At present, the use of ROS as a robotics programming platform is not as standard in Spain as in other more technologically developed countries, although every day more local companies make use of it. With the idea of introducing and standardizing the ROS as

a first-use platform for robot programming, the aim is to commercialize a product to be programmed with this platform and be an interactive solution for the teaching of robotics.

4.1.3 Promoters

The members of the MCubic company are Sergi De Las Muelas and Marc Sala, two students of Mechatronics Engineering at the University of Vic with a shared interest in mobile robotics and a desire to introduce the teaching of ROS to academic centers and universities.

- **Sergi de las Muelas**

  Sergi is a 4th year student of Mechatronics Engineering at the University of Vic and member of the robotics team of the same university, the Garrins Metàllics. He is passionate about mobile robotics and has experience working with this sector, using the ROS as a programming platform. He has worked in PAL Robotics programming and developing robots.

- **Marc Sala**

  Marc is a 4th year student of Mechatronics Engineering at the University of Vic. He really likes mobile robotics and is passionate about computer aided design. He has experience working in computer aided design, designing industrial machines and developing prototypes.

4.1.4 Mission

The main mission of the company is to become one of the largest sellers of ROS academic robots in Spain, aspiring to a possible world trade.
Its mission is also to be able to offer virtual and physical courses to learn how to use the ROS platform in an interactive way, using the ROSNI robot.

4.1.5 Objectives

The objectives set by the company have been classified chronologically, according to their priority. To put them in perspective, they have also been classified according to the year in which they are intended to be carried out, although there may always be some variation. The sales and finance plan will be based entirely on these goals.

- First Year

- Create the company and introduce it into the market making some publicity.
- Achieve a number of 50 sales.
- Be able to have a minimum salary of 200€ per month.

- Second Year
  - Offer some courses about learning ROS via the ROSNI.
  - Achieve a number of 100 sales.
  - Be able to have a minimum salary of 400€ per month.

- Third Year
  - Expand the company introducing the market beyond Spain.
  - Achieve a number of 200 sales
  - Be able to have a minimum salary of 800€ per month.

- Fourth Year
  - Establish the market of the company for Europe
  - Achieve a number of 400 sales.
  - Be able to have a minimum salary of 1200€ per month.

- Fifth Year
  - Having returned the bank loan and being able to finance our product and obtain profits at the end of the year.
  - Achieve a minimum number of 500 sales.
  - Be able to have a salary of 1500€ per month.

4.2 Market Study

4.2.1 Advantages

The ROSNI is a product that offers to the client many functions, starting with the wiki, that allows learning how to program it with ROS, making it easier to understand than the well-known ROS wiki.

On the other hand, this product also offers a great learning environment providing a

dynamic ROS learning, being able to physically see the programming of the robot with ROS, leaving aside the also well-known "turtlesim" of ROS.
Although this next part isn't related to ROS, it makes the robot more complex and easier to communicate with. The ROSNI comes with an application that allows the user to connect with him to provide communication between the user and the robot. This application makes the robot easy to use, making him do SLAM, or following a route previously created with the virtual route designer integrated in the app.

4.2.2 Target and Potential Customers

The ROSNI, as it has been explained in the previous sections, is intended for the use of learning, so the main customers of the product will be institutions for the education of robotics, private use and also technological centers and R+D departments.
The institutions for the education of robotics as universities and top grades, will be able to make their students learn dynamically providing them a great education and also a great knowledge about ROS.
In the case of the technological centers and R+D departments, the product will be useful for the research, learning and development of their own prototypes.
On the other hand the particular use of the ROSNI will make the customer able to also learn ROS dynamically and easily using the ROSNI wiki provided with the robot and making people be more interested about ROS, the present and the future of the programming of robots.
Talking about the market, the aim of this product is to be sold in any country in the world with an interest in mobile robotics, with the interest of introducing or improving mobile robotics to interested students or people who enjoy or want to learn mobile robotics.
The commercialization of the product is meant to be done via eCommerce, because nowadays, there aren't many robotic physical shops and most of the market of this sector is done via the Internet.
With the growth of social media, it would be interesting to create an Instagram profile to attract customers interested in the product. Through this platform we could announce news, make promotional videos, upload offers, etc. Another interesting point of Instagram is that you could also upload products in the Shop section and if someone clicks the product, they will be redirected to the official web-page of the product.
As most of the customers aren't expected to be particulars, it has to be found another method more effective to make promotion and sell the product, and this method could be

the publicity of the product in other virtual technological shops and also publishing the product on Amazon.
So finally to conclude this part, it would be great to create a simple and attractive web-page to captivate and engage visitors and finally make them be sure to buy the product.

4.2.3 Potential Market

In the previous sections it has been talked about the clients that the enterprise could actually have, but what about the future clients?
In the near future, robotics is expected to be one of the most popular industrial sectors over the world, so the market of this sector will increase. It means that the possibilities to attract more clients are higher and the potential market will be more extensive.
In conclusion, the customers of the product won't be as limited as nowadays, because in the future, more and more countries will join mobile robotics and with this growth, more clients will be attracted about learning mobile robotics and the market of the product will increase.
In order to take advantage of this potential market, a study of the technological centers and R+D departments in the area, which may be interested in the product, has been carried out.

4.2.4 PESTEL Analysis

This analysis will highlight the environmental, technological, social and economic factors, as political factors do not affect the marketing of this product and there are not too many legal factors that could limit it.
Starting with the political factors, the Spanish government is increasingly committed to the introduction of robotics in the industry, as are most of the governments around the world with some technological development.
Due to this increase of the interest in robotics, enterprises, industries and also particulars are investing more and more every day in this sector, and also the banks are so interested in this sector and many parts of the investments they carry out are related to robotics and technological development. Mobile robotics is a sector in this big branch of robotics that is growing more and more so these economical factors will also benefit the mobile robotics industry.
Nowadays, people feel the need to have robots in their lives. The society wants the robots to disregard housework, to simplify their job and also to communicate and socialize. Most

of the people in the world want a domestic robot to cook for them, to clean for them, etc. So to be able to achieve that goal, the world needs to learn mobile robotics, in addition to machine learning and artificial intelligence. For this reason, ROSNI offers the possibility to introduce you to this great world of mobile robotics.

It's obvious that the technological factors are the most prominent factors and will directly affect the enterprise and the marketing of the product. With the increase of the robotics market, more companies related to the mobile robotics sector have emerged, so the market has become more competitive. ROS has its own robot called Turtlebot on the market, as do other brands such as York and Ackerman, so MCubic needs to offer something that the other ones don't have or have it but worse.

Caring for the environment is a really important issue today, so in order to satisfy that aspect, the carcass of the robot could be made with thermoplastics replacing the 3D printing for another additive manufacturing process that allows thermoplastics. Another aspect to take into account is the battery of the robot, which is made of lithium (Li-Ion) and dissipates less load when it's not operating than other batteries. In addition, lithium batteries have a high energy density and, however, they are the ones with the lowest level of contamination.

Ending this analysis as it has been said previously, the legal aspects are not so relevant in the commercialization of the ROSNI. Currently, there is no rule that gives rights and obligations to the mobile robots. However, the product must be CE certified, which means that the manufacturer or importer affirms the good's conformity with European health, safety, and environmental protection standards. There also exists a legal personality, but it does not have the capacity to act, it is a diminished capacity (See Tab. 4.1).

Table 4.1: Main PESTEL analysis aspects.

PESTEL ANALYSIS	
Political	· The Spanish government is increasingly committed to the introduction of robotics in the industry.
Economical	· Enterprises, industries and also particulars are investing more and more every day in robotics. · Many parts of the investments banks carry out are related to robotics.
Social	· The society wants the robots to disregard housework, to simplify their job and also to communicate and socialize. · The world needs to learn mobile robotics.
Technological	· Lots of companies related to the mobile robotics sector have emerged. · The market has become more competitive.
Environmental	· The carcass of the robot could be made with thermoplastics. · Lithium battery with high energy density and a low level of contamination.
Legal	· Needs CE certificate · There exists a legal personality, but without capacity to act.

4.2.5 Future Keys

The future keys that will allow the project go ahead and make the ROSNI have a place in the current and future market are the following ones:

1. Upgrade the robot as the technology sector is updated to never fall behind the competence.

2. Keep social media active to capture customer attention at all times.

3. Conduct talks and events with the goal of going beyond social media and capturing the attention of new customers and improving the experience of the existing

customers.

4. Globalize the product going beyond Catalonia, in order to generate international trade.

4.3 Competitiveness

As in every sector, there exists competitors, and they make you put in more effort and improve your product to be more successful than competence.
Our product allows the customer or the user to learn ROS by dynamically learning from the robot's programming and its own wiki, it also allows you to use its graphical interface to interact with it without having to program, so the competence may have a similar product as the ROSNI.
Taking that into account, there are some enterprises that have a product with some similarities to our robot and with the same goals as our enterprise. The products that offer these enterprises are the following ones:

1. **Turtlebot**

   Turtlebot is the most popular ROS robot in the world, it consists of a mobile base, a 2D and 3D distance sensor, a single board computer and its hardware. It is designed to be easy to build and assemble. It is made only to work with ROS and it is nowadays the best one to learn to program with this operating system. It has four generations of robots, though the first one and the second one are discontinued. The Turtlebot 3 and the Turtlebot 4 are the ones that are available nowadays.

   (a) **Turtlebot 3**

   TurtleBot3 is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. The goal of TurtleBot3 is to dramatically reduce the size of the platform and lower the price without having to sacrifice its functionality and quality, while at the same time offering expandability. The TurtleBot3 can be customized into various ways depending on how you reconstruct the mechanical parts and use optional parts such as the computer and sensor. In addition, TurtleBot3 is evolved with cost-effective and small-sized SBC that is suitable for robust embedded systems and a 360 degree distance sensor. (ROBOTIS official web page description)

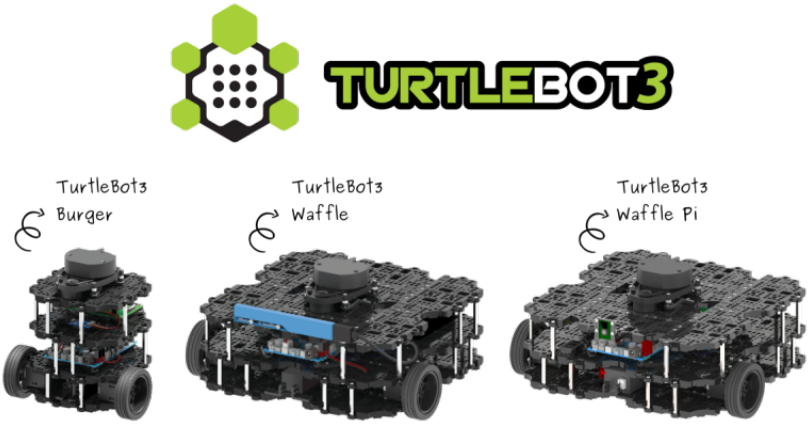


Figure 4.1: TurtleBot3 products.

(b) **Turtlebot 4**

TurtleBot 4 is the next-generation of the world's most popular open source robotics platform for education and research, offering better computing power, better sensors and a world class user experience at an affordable price point. It has two models and both are equipped with a Raspberry Pi 4 running ROS 2, an OAK-D spatial AI stereo camera, a 2D LiDAR and more. (Clearpath Robotics official web page description)

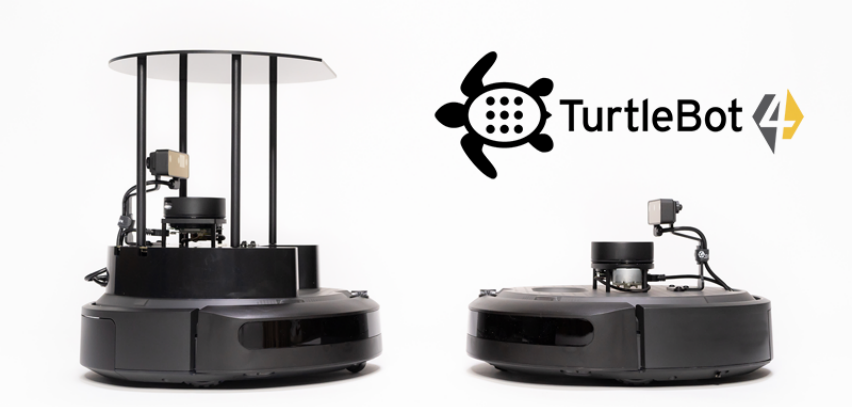


Figure 4.2: TurtleBot4 products.

2. **ROSbot**

ROSbot is a 4x4 drive autonomous mobile robot platform equipped with LIDAR, RGB-D camera, IMU, encoders and distance sensors powered by ROS. It's a direct creation of ROS, and it's equipped with some tutorials that introduce you to the robot. Its carcass is made with aluminum and it has its own Gazebo simulation model. It presents two distinguished robots, the ROSbot 2.0 and the ROSbot 2.0

PRO.

(a) **ROSbot 2.0**

It is the first generation of ROSbot and it includes a CORE2-ROS controller with Asus Tinker Board with Rockchip RK3288 up to 1.8GHz, 2GB DDR3 RAM and 32 GB MicroSD and a RPLIDAR A2 laser scanner (see Fig. 4.3a).

(b) **ROSbot 2.0 PRO**

It is a developed and updated version of the previous one and it includes a CORE2-ROS controller with UpBoard UP with Intel ATOM x5-Z8350 Processors 64 bits up to 1.92GHz, 4GB DDR3L RAM and 32GB eMMC and a RPLIDAR A3 laser scanner(see Fig. 4.3b).

Figure 4.3: (a)ROSbot 2.0 product, (b)ROSbot 2.0 PRO product.

3. **DuckieBots**

DuckieBots are low-cost mobile robots that are built almost entirely from off-the-shelf parts. The best of these robots is the Duckietown project, which consists of different parts, which are based on dynamic teaching and entertainment.

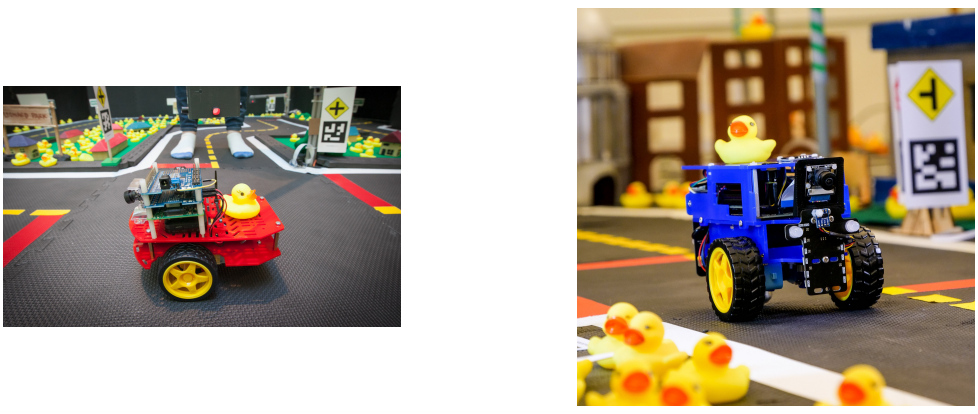


Figure 4.4: DuckieBot products.

4. **Linorobot**

   Linorobot is a suite of Open Source ROS compatible robots that aims to provide students, developers, and researchers a low-cost platform in creating new exciting applications on top of ROS. The good points of Linorobot are that it supports multiple types of robot base and you can build your own robot with the components that you want and that the platform supports depending on the type of robot that you want to achieve.

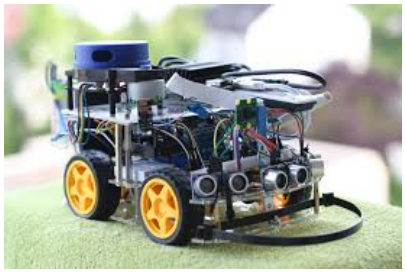


Figure 4.5: Robots made with the Linorobot source.

There are also some other companies in the market that could be competitors because their products are similar to the ROSNI and we share some marketing strategies and ideas but they are not comparable with the previous ones. Unlike the other products, ROSNI has its own wiki not only of the robot, also of ROS. It also has a graphical application to interact with the robot from which you can connect it to any Wi-Fi network, make it do SLAM, and even design routes for it to follow.

4.4 Marketing Plan

4.4.1 SWOT Analysis

By carrying out this analysis, it has been possible to detect the weaknesses, strengths, threats and opportunities that will allow the company to face the market and competition. It can be seen that the analysis is divided into two major parts, the internal factors, and the external factors.

On the one hand we would have the internal factors, they would be those that depend exclusively on the company and that would be reflected with the strengths and weaknesses. On the other hand, the external factors are those that do not depend only on the firm, but

also on the market, the economy, politics, and other factors that the firm cannot control, within which we find the opportunities and the threats.

Analyzing the strengths, the first thing to highlight is that the members of the company have studied mechatronics engineering, which are basically studies based specifically on robotics. Next, the young age of the members should be highlighted, as the desire to push the company forward and not giving up on the slightest problem is one of the biggest strengths. Another strength is how complete the product is, having its own wiki and its graphical interface to control and communicate with the robot, a point that differentiates the ROSNI from the other similar robots. Finally, another strength of the product is that it would be easy to globalize as its wiki and interface are made in english.

However, not all are strengths, the company also has weaknesses. The greatest of the weaknesses and that has also been put as strength, is the young age of the members of the company, that due to the inexperience in the labor sector, what before has been characterized as strength, also has become a weakness. Another weakness is that the company is newly created so it's not known and has no customers. Finally, it is worth noting the lack of business and marketing knowledge to move the company forward and have a better organization.

Starting with the external factors, the threats are something that the enterprise cannot control but can solve by creating some strategies and a good marketing plan. The low prices that China offers in the technological sector is a threat that all the enterprises are afraid of and that can't be competed. Another threat is the increase of the popularity of the robotics sector, as new enterprises will appear in the market and the competence will be higher. The update of technology day in and day out, can be a strength but also a threat as most of the technological enterprises will update their products in parallel with technology, so it means that the enterprise can't stay behind if it doesn't want to lose the competitiveness.

Following with external factors, as threats can be harmful for the enterprise, there also exist opportunities that can make the company grow up and be a reference in the market. The high growth of digitization and technology is a key point for the introduction of the company in the market, as the need of human beings to learn robotics every day is more noticeable. One of the biggest opportunities for the company to enter and grow in the market is the popularity that e-commerce is gaining day by day.

In addition, companies like Amazon facilitate the marketing of the product through the Internet, thus attracting more customers. Finally, the low prices that China offers, are

difficult to compete with, but also can be an opportunity for the company to make their own products cheaper, buying the robot components there, making China one of the largest suppliers.

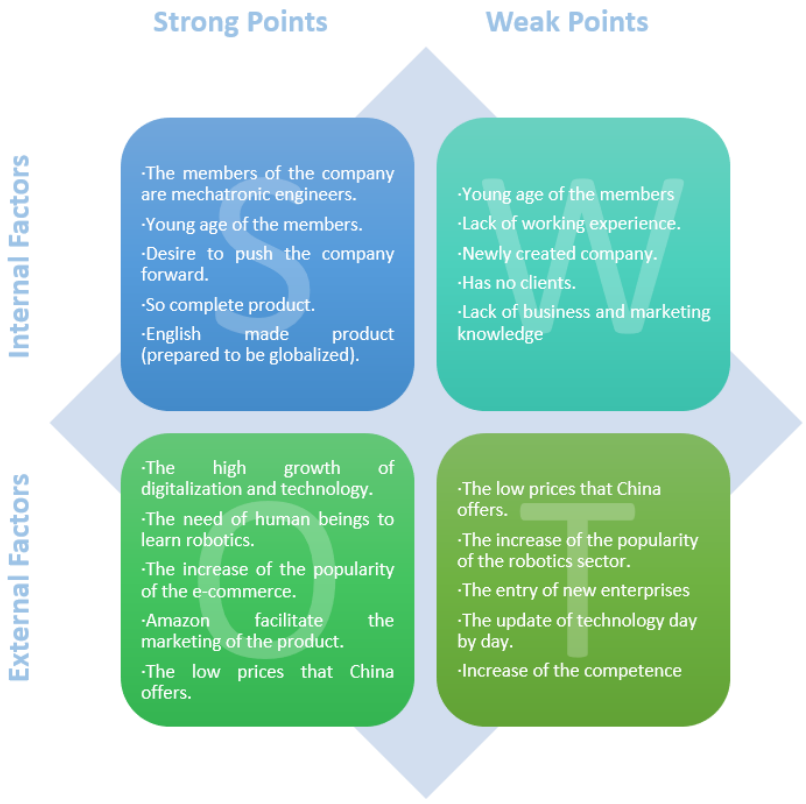


Figure 4.6: DAFO Analysis, most important aspects.

4.4.2 CAME Analysis

The CAME Analysis is a method to correct the weaknesses, adapt to the threats, maintain the strengths and exploit the opportunities that have been analyzed before, in the SWOT analysis.

In order to correct the weaknesses we have to create a survival strategy. The lack of experience will be improved as time goes on and the company will be facing the adversities that will be presented to it. Also the clients will arrive at the product if the enterprise creates a good marketing plan and prepare good market strategies. The lack of business and marketing knowledge can be solved by doing some courses or a master in marketing and communication or also contracting a new member in the company able to make the enterprise go on preparing strategies and marketing plans to arrive to the customers.

Sometimes, threats can become opportunities for the company, for this reason, MCubic has to face the threats using its strengths, as in the case of the low prices that China offers to its

customers. Taking advantage of this, the company can choose to make the Chinese market one of the largest suppliers of the company and thus obtain a more competent product in the market. With the increase of the popularity of the robot sector, new enterprises appear day by day, and with the update of the technology the market is becoming every day more competitive, but this fact will make the enterprise upgrade its product to be the most popular one and attract more customers.

Being up-to-date educationally is a key factor for the robotics industry, so to maintain one of the company's greatest strengths, we'll need to be educationally active to not fall behind with new technologies. Also related to this point, if the knowledge of the members must be updated day by day, the robot will also be upgraded due to the robotics sector development. It will also be interesting to not only have the robot platforms in english, it would be great to implement different languages depending on the language of the customer.

Finally, the main method to exploit the opportunities is to create strategies and plan actions to turn them into strengths. This kind of strategy is called reorientation strategy. One of the main weaknesses was the fact of being a newly created enterprise, but with the growth of the robotics sector, the market has also grown and more industries and people sense the need of having robots. To take advantage of that, we have the opportunity to do things right from the start, creating good publicity and making the customer be satisfied with the product and with the customer service. Another opportunity to exploit is that Amazon, one of the biggest e-commerce companies of the world nowadays, can become one of the best intermediaries, advertising and making product recommendations, as most people today buy through this platform, so it would be easier to attract customers.

4.4.3 Product Policy

ROSNI is a product offered by MCubic for teaching ROS through dynamic learning. This product consists of a mobile robot that has a LIDAR to be able to SLAM, a Raspberry Pi 4 as a CPU, two 146 rpm and 12V DC gear motors and a 14.8V and 2200mAh Li-Ion battery as its main components. It is complemented by a wiki to facilitate its programming by ROS, explaining step by step all the necessary to maximize the capabilities and functionalities of the robot. It's also equipped with a graphical interface to interact with it without having any notions of programming. The same application allows the user to connect the robot to any Wi-Fi network, order the robot to perform SLAM and even create their own virtual routes for the robot to follow.

The product is completely assembled, no installation or assembly should be carried out after delivery. All you have to do is install the app and start interacting with the robot. As with all lithium batteries, routine maintenance will be required before it runs out of energy, otherwise it will be damaged.

4.4.4 Advertising and Communication Strategy

The company will be known not only through advertising, but also through communication strategies.

1. **Instagram:** The main advertising of MCubic will be done through social media, mainly via Instagram. Within this platform will be carried out the advertising of the products, the ROSNI in this case, and will also be used to inform of promotions, to announce novelties, courses done by MCubic and attendances to fairs.

2. **Mail:** Secondly, some publicity will be done using the mailing method. This method will be a mix between advertising and communication, as it will be used to announce new products, promotions and also courses. It won't work as well as Instagram with advertising as it will be more focused on the company's customers that want to receive notices.

3. **Web-page:** It will also be a good point to create a web-page of MCubic to commercialize the products, do online courses, attract the customers, etc.

4. **Communication and events:** Another method to make publicity and catch clients is by assisting technological events, robotics fairs and also by going to universities and academic centers interested in presenting the product and thus make yourself known within the academic field.

4.4.5 Business Action Plan

This plan has to be based in the SMART method, Specific, Measurable, Achievable, Relevant and Timely.
The main objective of MCubic is to sell an amount of 50 robots during the first year in order to be introduced in the market and reach the goal of having some benefits the third year. To achieve that objective, some publicity will be done using social networks and as it has been explained previously, going to some robotics fairs, going to academic centers to present the robot and also introducing the product in some web pages such as Amazon

to catch more clients.

The main strength of the ROSNI is its price, that compared with the other similar products in the market, it is much cheaper.

The Table 4.2 shows the most prominent products on the market for the educational robotics sector of ROS and its sell prices excluding VAN and excluding it.

Table 4.2: Most prominent ROS kits on the market and its prices.

Product	**Price (VAN exc.)**	**Price (VAN inc.)**
TurtleBot3 Burger	509 €	616 €
TurtleBot3	1.199 €	1.451€
Turtlebot 4	1.263 €	1.599 €
Turtlebot 4 PRO	2.499 €	3.024 €
ROSbot 2.0	1.758,28 €	2.127,52 €
ROSbot 2.0 PRO	2.823,07 €	3.415,91 €
ROSNI	473,21 €	599 €

4.5 Sales Plan

The sales plan is the base of the economic and financial plan, and it will be done computing the number of robots that would need to be sold in a year to obtain a salary of 200€ per month, increasing year by year and also adding the expenses involved in having a limited society (Ltd.).

Below, in the economic and financial plan are the calculations and tables that show the results of selling an amount of 50 robots the first year and the requirements that the enterprise has to meet to be established as a limited society. A five year plan has also been computed to see the evolution of the company, increasing the number of robots to be sold year by year and doing the same with the salary.

The first idea was to create a plan based on not having any losses at the end of the year by selling a total of 200 robots. By selling less than 170, the company wouldn't generate profit, and being optimistic, selling about 200 robots a year, the company would generate some profit and could give a salary of 14 annual payments of 1200€ for each member

But carrying out this plan, a very high loan of money would have been requested and it would be very difficult for a bank to have it available, and the interests would also have been too high.

In order to carry out a realistic business plan, it has been developed the five year plan explained previously having losses at the first and second year but extracting good results the following years.

4.6 Economic and Financial Plan

4.6.1 Initial Investment Plan

Table 4.3: Initial investment plan to be able to undertake and carry out MCubic.

Initial Investment Plan	
Concept	**Cost/euros**
Buildings, premises and land (purchase in property) (*)	-
Computer applications (software)	0
Facilities (Adaptation of the premises: high + reforms)	-
Machinery	-
Tools	158
Furniture and equipment	-
Computer equipment	-
Transport elements	-
Transfer reights/ Patents and marks	-
Deposits and bonds for local rental	840
Constitution and commissioning expenses	-
Initial stocks of raw materials (*)	15405
Other expenses	-
Provision provision (unforeseen expenses)	-
Supported VAT (investment elements)	3268,23
Total	19.671 €
Note: All amounts are not including VAT, being a tax is not subsidized. (As a general rule is 21% although it can vary)	

4.6.2 Financing Plan

Table 4.4: Financing plan with the loan, the own capital and the share capital.

Financing plan	
Concept	**Amount/euros**
Share capital (according to deeds)	3000
Own resources (partner contributions)	10000
Loans (already granted or in approval phase)	8500
Incentives (already granted)	-
Capitalization (unemployment benefits: single payment charged)	-
Others (leasing, contributions without financial costs, etc.)	-
Total	**21500 €**
Note: The sum of social capital and your own resources have to be at least 25% of the investment	
MANEUVERING BACKGROUND	**1.828,77 €**

Table 4.5: Results of applying for a loan with an interest of a 4% and a return of five years.

Interest (%)	Return (months)
4	60
Total interest euros (Year):	680 €
Loan + interests euros (Year):	2380 €

4.6.3 Provisional Results Account

Table 4.6: Bills scheduled for the first year by manufacturing 50 ROSNIs.

Bills	First year
Raw and auxiliary matter purchases	15405
Initial stocks	-
Own remuneration	-
Autonomous insurance	7056
Personal salary or collaborators	5600
H.H. In charge of the company	-
Loan financial expenses	680
Other financial expenses	300
Publicity and promotion	-
Taxes (contributions, fees, etc.)	100
Supplies (light, water, telephone, gasoline, etc.)	840
Rentals	-
Insurance	50
Maintenance and repairs	-
Transports	-
Foreign Services (Management, ...)	-
Constitution expenses (notaries, records, ...)	500
Diverse expenses (stationery, uniforms, restoration expenses, messaging, locomotion, etc.)	-
Material immobilized amortization endowment	-
Intangible immaterial amortization provision	-
Provision provision (unforeseen expenses)	-
Total	**-30531 €**

Table 4.7: Incomes scheduled for the first year by selling 50 ROSNIs.

Income	First year
Sales	23660,5
Final existence	-
Financial income (loans)	-
Incentives (already granted)	-
Others (extraordinary income)	-
Total	**23660,5 €**

Table 4.8: Benefits before and after taxes of the first year.

Benefits after taxes (Income - expenses) =	**-6870,5**
Imposition over benefits	**0**
Benefits after taxes (25%)	**-6870,5 €**

The provisional results account, has been computed for the first five years. In order to see the results of these years, please go to the Appendix C.

4.6.4 Profitability Threshold

- **First Year**

Table 4.9: Profitability threshold of the first year with the final year balance.

MONTH	1	2	3	4	5	6	7	8	9	10	11	12	TOTAL n
INITIAL BALANCE	**3.000,00**	**21.011,40**	**20.572,79**	**17.770,56**	**17.331,96**	**16.893,36**	**15.406,13**	**14.967,52**	**14.528,92**	**13.541,69**	**13.103,09**	**12.664,48**	**3.000,00**
Charges													
Own Resources	10.000,00												**10.000,00**
Banking loom	8.500,00	0	0	0	0	0	0	0	0	0	0	0	**8.500,00**
Customers	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	**28.629,21**
Aid and subsidies	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Total charges	**20.885,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**47.129,21**
PAYMENTS													
Suppliers	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	**18.640,05**
Rentals and canons	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Software													
Maintenance and repairs	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Professional services	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Transportation	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Promotion and publicity	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Supplies	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	**1016,4**
Other external expenses	0	0	1815	0	0	100	0	0	0	0	0	0	**1915**
Insurance	50	0	0	0	0	0	0	0	0	0	0	0	**50**
Autonomous Workers (+IRPF)	400	400	400	400	400	800	400	400	400	400	400	800	**5.600,00**
Workers GENERAL REGIME	0	0	0	0	0	0	0	0	0	0	0	0	**0**
SECT SOCIAL AUTONOMOUS WORKERS	588	588	588	588	588	588	588	588	588	588	588	588	**7.056,00**
Social Seg Workers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Loans	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	**2.380,00**
Immobilized suppliers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Imposition over benefits	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Value Added Tax	0	0	548,625	0	0	548,625	0	0	548,625	0	0	548,625	**2.194,50**
Total payments	**2.874,37**	**2.824,37**	**5.188,00**	**2.824,37**	**2.824,37**	**3.873,00**	**2.824,37**	**2.824,37**	**3.373,00**	**2.824,37**	**2.824,37**	**3.773,00**	**38.851,95**
Final balance	**21.011,40**	**20.572,79**	**17.770,56**	**17.331,96**	**16.893,36**	**15.406,13**	**14.967,52**	**14.528,92**	**13.541,69**	**13.103,09**	**12.664,48**	**11.277,26**	**11.277,26**

As in the case of the provisional results account, the profitability threshold has been also carried out for the first five years of the company. In the Appendix C, there appear the tables with the results of the profitability threshold of these first five years.

4.6.5 Annual Deadlock

The annual deadlock will let us know when the total costs equal the total revenue per sale annually and it's computed by dividing the fixed costs over the sum of the unitary sale cost and the unitary variable cost (see Eqn. (4.1)).

$$Qc = \frac{CF}{PVu - CVu} \tag{4.1}$$

- First Year

$$Qc = \frac{CF}{PVu + CVu} = \frac{20.211,9}{599 - 390} = 96 units \tag{4.2}$$

- Second Year

$$Qc = \frac{CF}{PVu + CVu} = \frac{26.091,4}{599 - 390} = 125 units \tag{4.3}$$

- Third Year

$$Qc = \frac{CF}{PVu + CVu} = \frac{42.154,4}{599 - 390} = 202 units \tag{4.4}$$

- Fourth Year

$$Qc = \frac{CF}{PVu + CVu} = \frac{65.887,9}{599 - 390} = 315 units \tag{4.5}$$

- Fifth Year

$$Qc = \frac{CF}{PVu + CVu} = \frac{80.704,65}{599 - 390} = 387 units \tag{4.6}$$

Where:

CF=Fixed Costs

PVu=Unitary Sale Price

CVu=Unitary Variable Cost

4.6.6 Investment Payback Time

The Investment Payback Time is the time that takes to the company to return the initial investment, taking the pretax profit as income.

Having an initial income of 24.187€ and a pretax profit of -7.230,5€ the investment payback cannot be computed as there are no benefits the first year. Another reason of why it cannot be computed, is that the sales will be increasing over the years so the payback time will not be linear over the years but seeing that in the first year the company will have some little benefits, it can be said that the payback time of the enterprise will be about 3 years (see Eqn. (4.7)).

$$IPB = \frac{Inv.Cost}{PretaxProf.} = \frac{19671}{-6870.5} = -2.86 years = -34 months \tag{4.7}$$

4.6.7 NPV and IRR

The Net Present Value (NPV) is the difference between the present value of cash inflows and the present value of cash outflows over a period of time and is used to analyze the profitability of a projected investment.
To compute that value is necessary to follow the Equation (4.8).

$$NPV = -I + \sum_{t=1}^{n} \frac{Rt}{(1+i)^t} = 40642 \tag{4.8}$$

Where:

Rt=Net cash inflow-outflows during a single period t

i=Discount rate or return that could be earned in alternative investments

t=Number of timer periods

The Internal Rate of Return (IRR) is a directly related parameter of the NPV because it is used to analyze the profitability of a projected investment, as the NPV. The IRR is the interest rate that will bring a series of cash flows to a Net Present Value (NPV) of zero and it is computed with the following formul (Eqn. 4.9):

$$NPV = \sum_{n=0}^{N} \frac{CFn}{(1+IRR)^n} = 13.52\% \tag{4.9}$$

Where:

CF=Cash flows

n=Each period

N=Holding period

NPV=Net present value

IRR=Internal rate of return

4.7 Conclusions

As can be seen in the market study, the robotics sector is gaining more popularity every day and taking more weight in the technology sector. This fact allows the entry and growth of new companies in the market, with the plus that mobile robotics is increasingly needed by society, thus creating new opportunities in the sector of teaching mobile robotics. Once the company enters into the market, with a good market and advertising plan, lots of opportunities can be explored, selling robots and teaching mobile robotics.
Another great opportunity is that the government and lots of enterprises are interested in developing this sector, so it wouldn't be so difficult to find investors for the company avoiding the pay of high interests.
With this increase, new companies enter the market every day, increasing the competitiveness of the sector. It will make the company take more risks to be more competitive day by day, reducing costs and prices and being developed.
One of the biggest problems of robotics is the great development and renewal to which the sector is subjected day after day, which means that in order to compete within it, the company must be up to date, to not stop being a competitor in the market.
As it has been computed in the economic and financial plan, the company can be profitable as there are some benefits in the third year since the creation of the company, increasing the fourth and the fifth years. It will be difficult to carry on the company, because in the first and the second years, it won't generate benefits, but with ambition and perseverance it can result in a good project.

Chapter 5

Final Result

The principal aim of the project was to create a ROS educational kit composed by a mobile robot, an application and a wiki. The robot had to be able to be programmed with ROS commands and to execute orders from the app, and the app had to be able to create virtual routes, contain the robot's wiki and send specific commands to the robot (perform SLAM, execute the route and connect to a Wi-Fi network). Actually the robot can execute ROS commands, perform SLAM, be remotely controlled via an Xbox controller, establish connection with a Wi-Fi network and receive commands from the app. Meanwhile, the application is able to send commands to the robot, such as SLAM and execute a route, create virtual routes and maps, access to the robot's wiki, establish connection to it and make it connect to a Wi-Fi network. The app is also able to visualize, in real time, the map that the robot creates, while is performing SLAM.

Based on the objectives proposed at the beginning of the project, it has been done an analysis of each one of them to determine the degree of compliance of them. In the previous paragraph there are explained the functions that the robot and the app are actually able to do, so knowing them, it's possible to do a evaluation of the objectives.

The first of one of the specific objectives purposed, was to develop a ROS-based starter kit able to face the competence of the actual market. The ROS-based starter kit has been developed and it could be able to become a competent product in the market, performing a good business plan.

The second one of the objectives, was to design and implement a multi-platform app able to design virtual maps and routes and makes the robot execute some commands as performing SLAM, executing routes, and more. As explained in the introduction of this section, the app is able to perform all these functions and it has also been improved

implementing new functionalities on it as the visualizing of the SLAM map, in real time, making the robot connect to a Wi-Fi network and access to the robot's wiki among others. As the last objective, we set out to develop a business plan in order to study the viability of introducing the product into the market. The business plan has been developed and it has been studied the viability of introducing the product into the market by making some plans and analysis as the (PESTEL analysis, the SWOT analysis, CAME analysis, the economical plan, the financial plan, and more).

Finally, we have performed a qualitative evaluation, proving that all of the objectives have been completed successfully, even adding some improvements in the final product.

Chapter 6

Conclusion

In general terms we consider that this project has been successfully performed, achieving the major part of the objectives purposed at the beginning of it. We have built a functional robot prototype that works properly and an application to interact with it. Even though the project involved several tasks (robot design, construction and programming; App design and implementation; integration; and business plan; robot wiki), we managed to succeed in it's performance. However, we would like to improve the actual project by including more integration between the App and the Robot. These improvements will be explained in the Section 7.

During the development of the project, we have faced some difficulties and obstacles that have been solved successfully. However, we have had some limitations at the end of the project because of the lack of time. These limitations will be purposed as future objectives to achieve.

Focusing on the robot, during the course of conceptualization and design (mechanic and electronic), a lot of changes have been made. At the beginning of the project, it had been thought of with certain electronic components that in the end have been replaced by more effective and efficient ones. And the same happens with the mechanical parts.

In the case of the app, at the beginning, it should have been a simple graphical interface, just being able to send commands to the robot and to create virtual routes. However, it has been upgraded during the development of the project by adding on it a bidirectional socket communication to could see the robot SLAM lively, and more.

In the part of the business plan, we have tried to perform it as realistic as possible, in order to prove that the product has future in the market. To see the detailed conclusions of the business plan, please, go to the Section 4.7.

As a final conclusion, we would like to express our happiness and pride about the work we have done. It has been a difficult process, but also so satisfactory, as we have learned lots of new concepts and we have had a good organization during all the project.

Chapter 7

Future work

For us, all the work that has been shown in this document is just the beginning of building a nice product to let people come close to robotics. In this section we propose the next steps we would like to do in order to improve the project.

- We would like to implement the **App's routes execution**. As we saw in previous sections, the user is able to create routes in the App and also to store and send them to the robot, even though the robot does not executes them. We propose to implement a new ROS node able to get the different points of the route and compute the necessary wheel velocities to perform that trajectory.

- Other nice feature would be the **autonomous path planning and navigation**. There are several ROS packages that implement different path planners but they need the robot to be set up in a concrete way. We propose to adapt the current packages and configure the full *Navigation Stack* of ROS, which includes local and global Path Planners, localization algorithms, etc.

- Once the robot becomes autonomous on path planning, we could improve the actual SLAM method by supplying the possibility of doing **autonomous SLAM** instead with teleoperation. There are algorithms that analyze the map that the robot is building, and generate the next best view point. Then the robot navigates to the selected points updating the map.

- On the Application side, we can add a feature to **store, load and send maps to the robot**. This would be an upgrade of the map creation feature already implemented and would allow the robot to use maps with virtual obstacles it should avoid while planning paths and navigating.

Many things could be improved before getting a final product to put on the market but here we have mentioned some of them we would like to implement in a short period of time. We look forward to keeping improving the product and learning during the process.

Bibliography

[1] Oyelekan Bukunmi. *Comparing the top Python GUI frameworks*. en. https://blog.logrocket.com/comparing-top-python-gui-frameworks/. Accessed: 2022-6-4. Nov. 2021.

[2] A B Campo. "PID Control Design". In: *MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 1*. Ed. by Vasilios N Katsikis. London, England: InTech, 2012.

[3] *Qt for Python*. https://doc.qt.io/qtforpython-6/. Accessed: 2022-6-4.

[4] Renaud Ronsse. *BEST2015 -autonomous mobile robots lecture 2: Mobile robot kinematics and control*. https://perso.uclouvain.be/renaud.ronsse/BEST2015/2015_slides_L2_2pages.pdf. Accessed: 2022-6-4. 2015.

[5] *ROS: Home*. en. https://www.ros.org/. Accessed: 2022-6-4.

[6] *Socket Types and Protocols*. https://sites.ualberta.ca/dept/chemeng/AIX-43/share/man/info/C/a_doc_lib/aixprggd/progcomc/skt_types.htm. Accessed: 2022-6-4.

[7] *What is Additive Manufacturing? Definition, Types and Processes*. en. https://www.twi-global.com/technical-knowledge/faqs/what-is-additive-manufacturing. Accessed: 2022-6-4.

Appendix A

CAD Plans

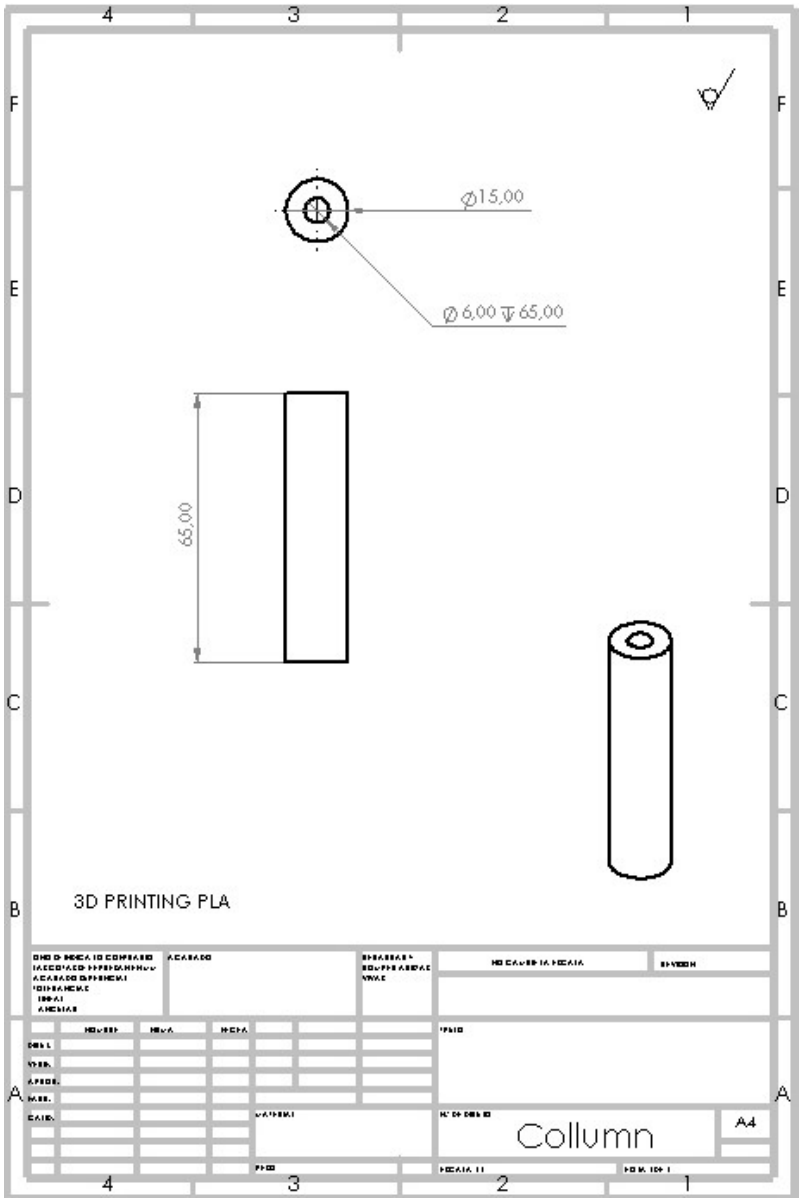


Figure A.1: Robot columns plan.

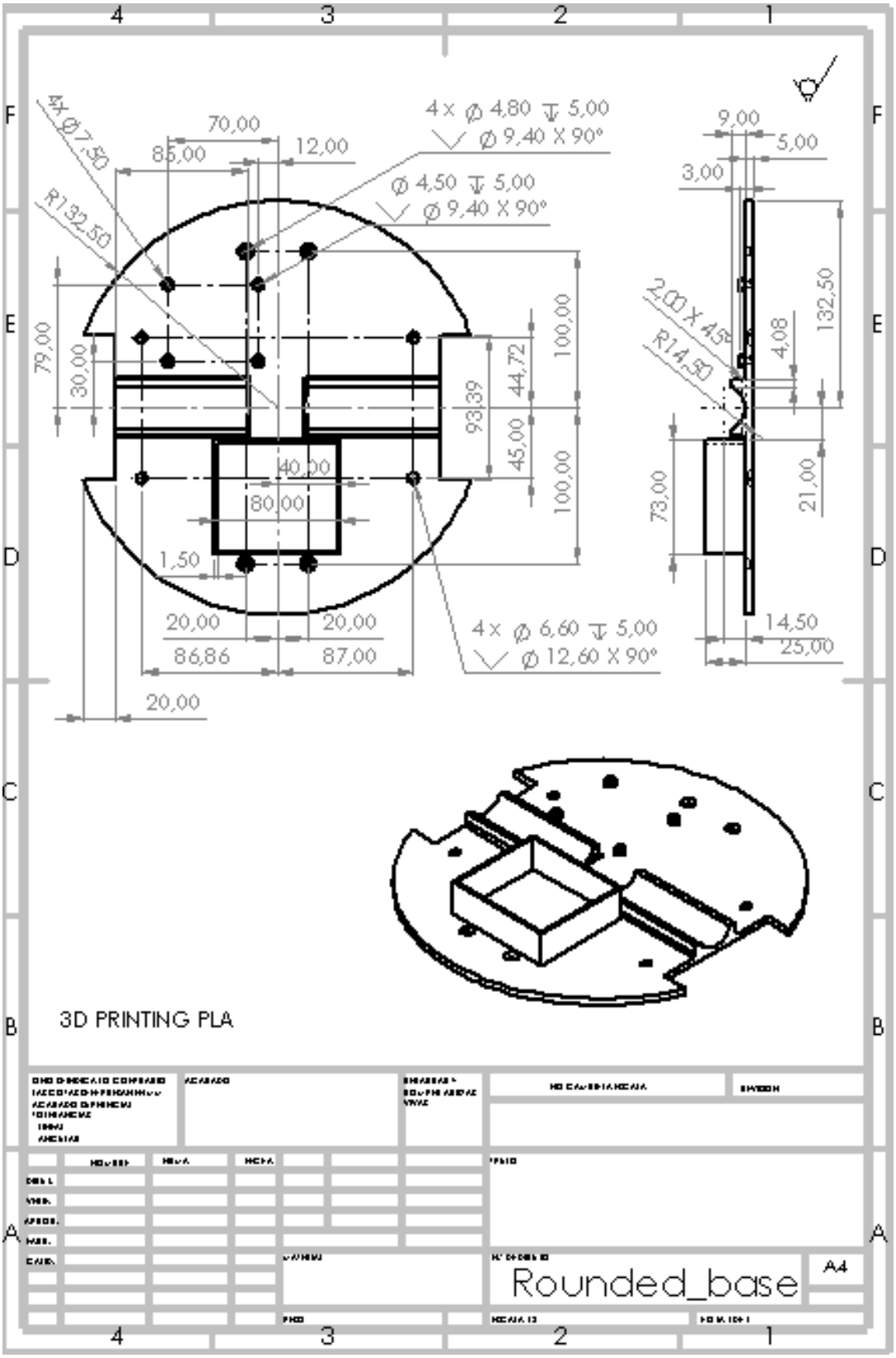


Figure A.2: Robot base plan.

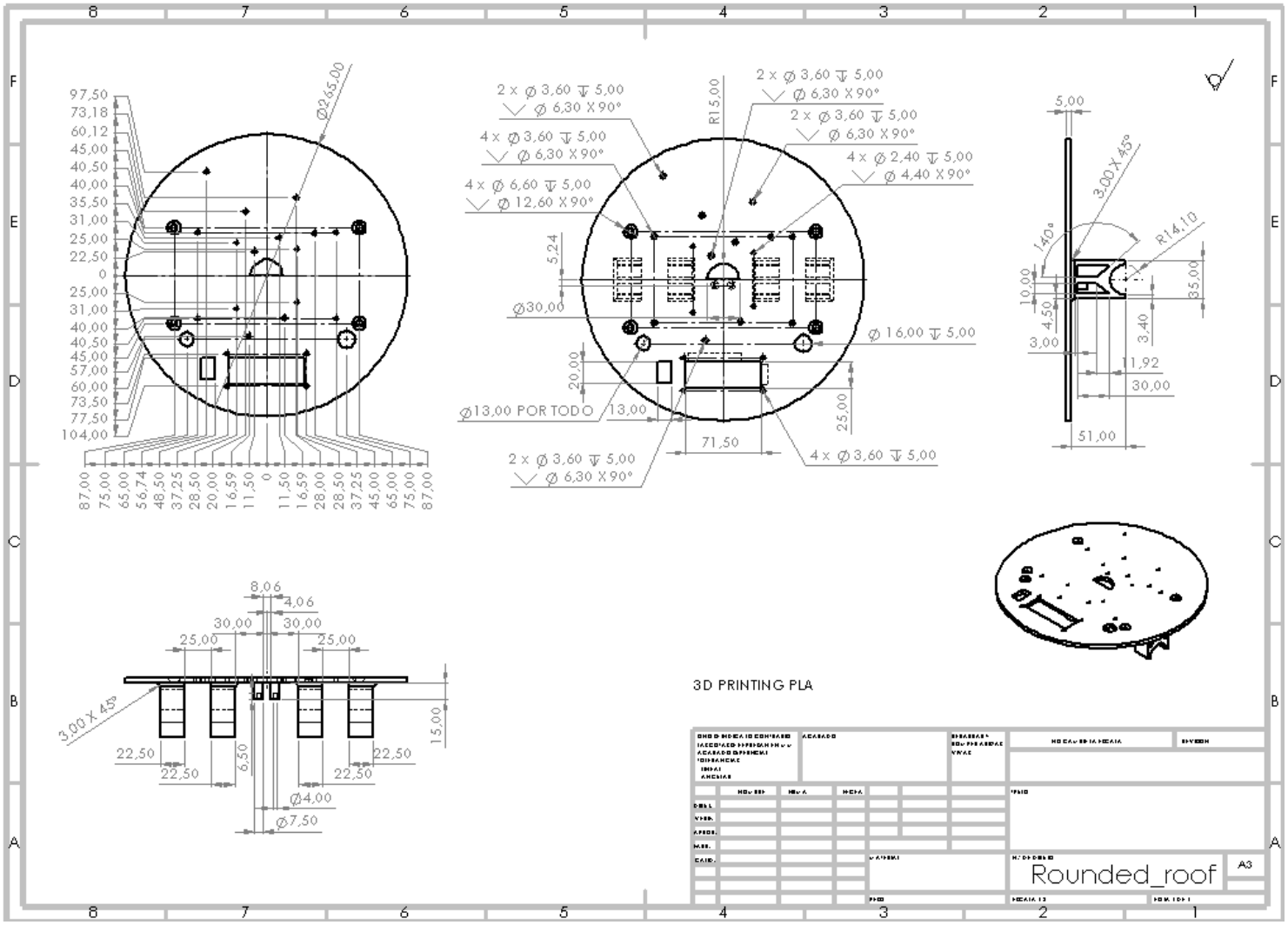


Figure A.3: Robot roof plan.

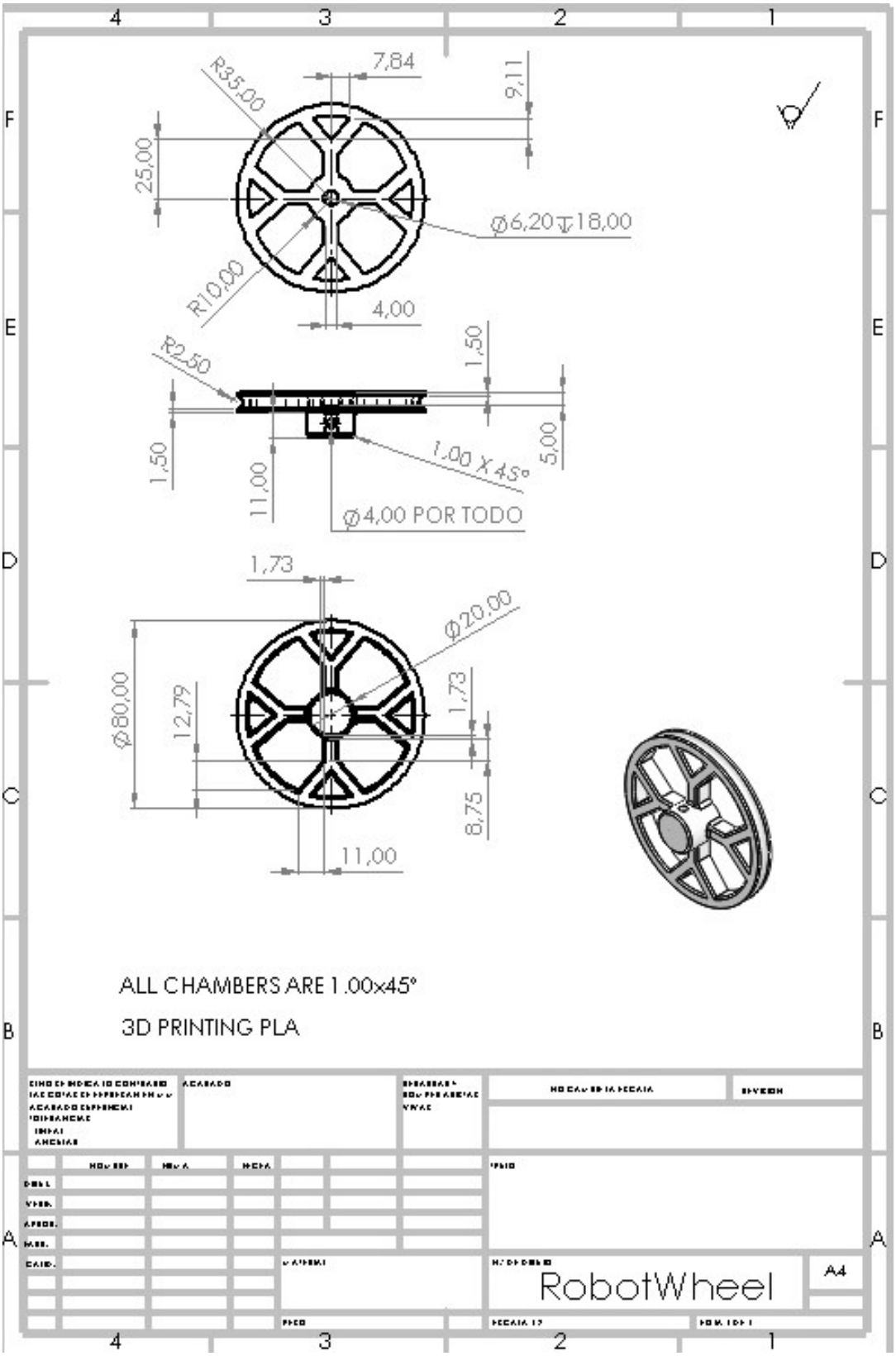


Figure A.4: Robot wheel plan.

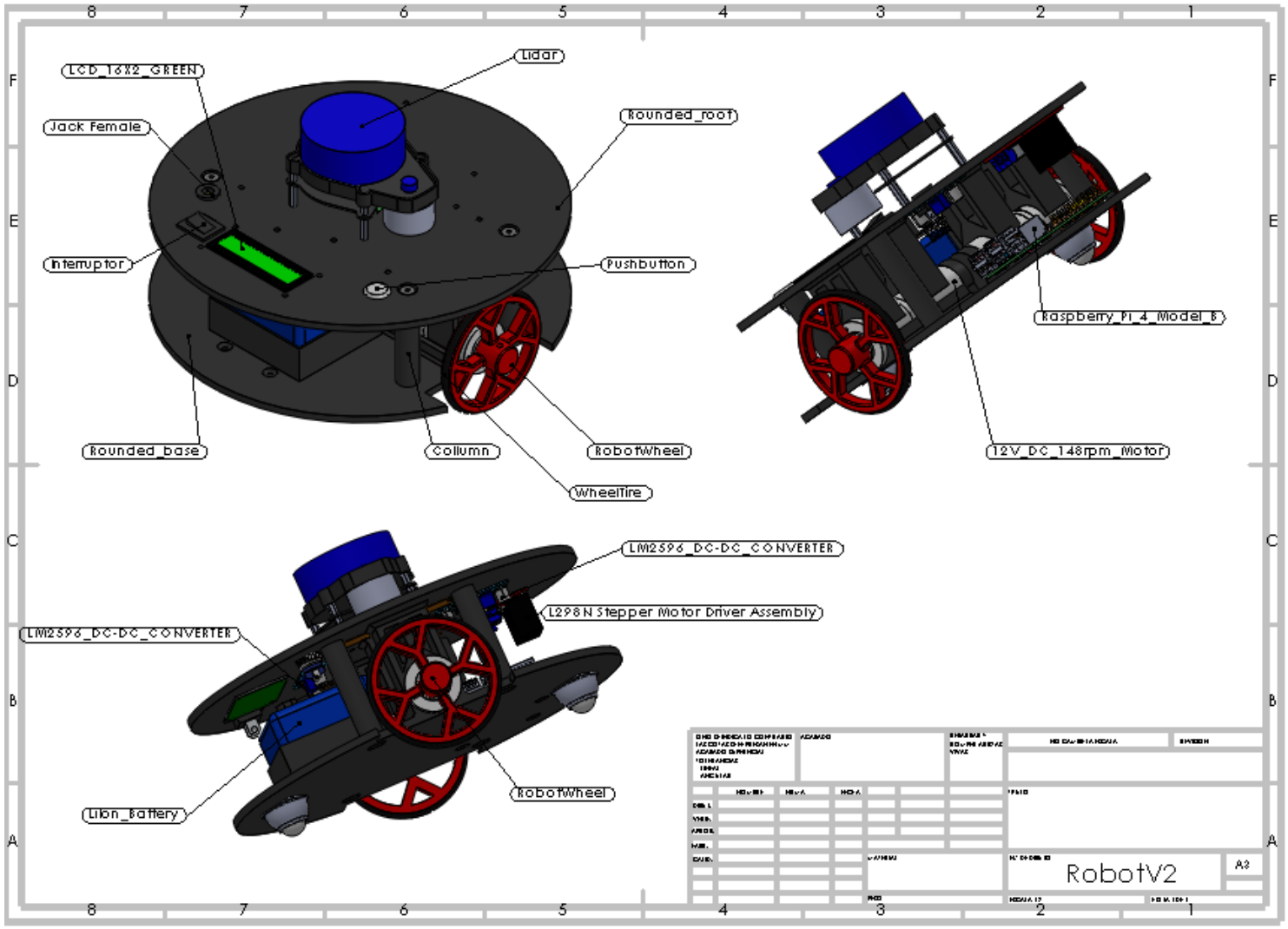


Figure A.5: Robot assembly plan.

Appendix B

PyQt commands (Interactive app)

Table B.1: Qt functions and commands used to program the app.

Widget	**Description**
QApplication	The base of a Qt program, it drives the loop of the application.
QPushButton	Widget able to execute an action when clicked.
QLabel	Widget capable of containing non editable text or images within it.
QLineEdit	Widget in which can be written a line of text.
QComboBox	Widget that combines a button and a pop-up window where a list of items is displayed when the button is pressed.
QCheckBox	Widget that allows us to check or not an option and perform a different operation if the option is checked or not.
QWidget	Widget that can be used to display custom content, it can work as a window, as a picture, as an object, and more.
QIntValidator	Ensures that a string contains a valid integer, commonly used with the QLienEdit widget.
QMainWindow	Provides a framework for building an application's user interface.

Table B.2: Continuation of Tab. B.1

Widget	**Description**
QDialog	Provides a top-level window mostly used for short-term tasks.
QFileDialog	Provides a dialog window that allow users to select files or directories
QMessageBox	Provides a modal dialog to display some informational, warning or error message, and optionally ask the user to respond by clicking any one of the standard buttons it contains.
QPixmap	Off-screen image representation that can be used as a paint device and also to deploy an image.
QPainter	Class used to realize drawing operations.
QColor	Provides colors based on RGB values, normally used with the QPainter.
QRect	Provides a rectangle in the screen, commonly used with the QPainter.
QPen	Defines the properties of the draws done with QPainter.
QFont	Specifies a font used for drawing text.
QPropertyAnimation	Interface used to animate widgets. It is based on the properties of the widgets, so it can be used to animate a move, a resize and the change of a the appearance of a widget.
QSequentialAnimationGroup	Allows executing a group of QPropertyAnimation respecting the order they have been inserted, executing them one by one.
QPoint	Defines a point in the screen using integer coordinates.

Appendix C

Economic and financial plan

C.1 First Year

C.1.1 Provisional Results Account

Table C.1: Incomes scheduled for the first year by selling 50 ROSNIs.

Income	First year
Sales	23660,5
Final existence	-
Financial income (loans)	-
Incentives (already granted)	-
Others (extraordinary income)	-
Total	**23660,5 €**

Table C.2: Bills scheduled for the first year by manufacturing 50 ROSNIs.

Bills	First year
Raw and auxiliary matter purchases	15405
Initial stocks	-
Own remuneration	-
Autonomous insurance	7056
Personal salary or collaborators	5600
H.H. In charge of the company	-
Loan financial expenses	680
Other financial expenses	300
Publicity and promotion	-
Taxes (contributions, fees, etc.)	100
Supplies (light, water, telephone, gasoline, etc.)	840
Rentals	-
Insurance	50
Maintenance and repairs	-
Transports	-
Foreign Services (Management, ...)	-
Constitution expenses (notaries, records, ...)	500
Diverse expenses (stationery, uniforms, restoration expenses, messaging, locomotion, etc.)	-
Material immobilized amortization endowment	-
Intangible immaterial amortization provision	-
Provision provision (unforeseen expenses)	-
Total	**-30531 €**

Table C.3: Benefits before and after taxes of the first year.

Benefits after taxes (Income - expenses) =	**-6870,5**
Imposition over benefits	**0**
Benefits after taxes (25%)	**-6870,5 €**

C.1.2 Profitability Threshold

Table C.4: Profitability threshold of the first year with the final year balance.

MONTH	1	2	3	4	5	6	7	8	9	10	11	12	TOTAL n
INITIAL BALANCE	**3.000,00**	**21.011,40**	**20.572,79**	**17.770,56**	**17.331,96**	**16.893,36**	**15.406,13**	**14.967,52**	**14.528,92**	**13.541,69**	**13.103,09**	**12.664,48**	**3.000,00**
Charges													
Own Resources	10.000,00												**10.000,00**
Banking loom	8.500,00	0	0	0	0	0	0	0	0	0	0	0	**8.500,00**
Customers	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	2.385,77	**28.629,21**
Aid and subsidies	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Total charges	**20.885,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**2.385,77**	**47.129,21**
PAYMENTS													
Suppliers	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	1.553,34	**18.640,05**
Rentals and canons	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Software													
Maintenance and repairs	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Professional services	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Transportation	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Promotion and publicity	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Supplies	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	**1016,4**
Other external expenses	0	0	1815	0	0	100	0	0	0	0	0	0	**1915**
Insurance	50	0	0	0	0	0	0	0	0	0	0	0	**50**
Autonomous Workers (+IRPF)	400	400	400	400	400	800	400	400	400	400	400	800	**5.600,00**
Workers GENERAL REGIME	0	0	0	0	0	0	0	0	0	0	0	0	**0**
SECT SOCIAL AUTONOMOUS WORKERS	588	588	588	588	588	588	588	588	588	588	588	588	**7.056,00**
Social Seg Workers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Loans	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	**2.380,00**
Immobilized suppliers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Imposition over benefits	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Value Added Tax	0	0	548,625	0	0	548,625	0	0	548,625	0	0	548,625	**2.194,50**
Total payments	**2.874,37**	**2.824,37**	**5.188,00**	**2.824,37**	**2.824,37**	**3.873,00**	**2.824,37**	**2.824,37**	**3.373,00**	**2.824,37**	**2.824,37**	**3.773,00**	**38.851,95**
Final balance	**21.011,40**	**20.572,79**	**17.770,56**	**17.331,96**	**16.893,36**	**15.406,13**	**14.967,52**	**14.528,92**	**13.541,69**	**13.103,09**	**12.664,48**	**11.277,26**	**11.277,26**

C.2 Second Year

C.2.1 Provisional Results Account

Table C.5: Bills scheduled for the second year by manufacturing 100 ROSNIs.

Bills	Second year
Raw and auxiliary matter purchases	30810
Initial stocks	-
Own remuneration	-
Autonomous insurance	7056
Personal salary or collaborators	11200
H.H. In charge of the company	-
Loan financial expenses	680
Other financial expenses	-
Publicity and promotion	-
Taxes (contributions, fees, etc.)	100
Supplies (light, water, telephone, gasoline, etc.)	840
Rentals	-
Insurance	50
Maintenance and repairs	-
Transports	-
Foreign Services (Management, ...)	-
Constitution expenses (notaries, records, ...)	-
Diverse expenses (stationery, uniforms, restorating expenses, messaging, locomation etc)	-
Material immobilized amortization endowment	-
Intangible immaterial amortization provision	-
Provision provision (unforeseen expenses)	-
Total	**-50736**

Table C.6: Incomes scheduled for the second year by selling 100 ROSNIs.

Bills	Second year
Sales	47321
Final existence	-
Financial income (loans)	-
Incentives (already granted)	-
Others (extraordinary income)	-
Total	**47321**

Table C.7: Benefits before and after taxes of the second year.

Benefits after taxes (Income - expenses) =	-3415
Imposition over benefits	0
Benefits after taxes (25%)	-3415 €

C.2.2 Profitability Threshold

Table C.8: Profitability threshold of the second year with the final year balance.

MONTH	1	2	3	4	5	6	7	8	9	10	11	12	TOTAL n
INITIAL BALANCE	**11.277,26**	**11.221,08**	**11.214,91**	**10.111,48**	**10.105,31**	**10.099,13**	**8.195,71**	**8.189,54**	**8.183,36**	**7.079,94**	**7.073,76**	**7.067,59**	**11.277,26**
Charges													
Banking loom	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Customers	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	4.771,53	**57.258,41**
Aid and subsidies	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Total charges	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**4.771,53**	**57.258,41**
PAYMENTS													
Suppliers	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	3.106,68	**37.280,10**
Rentals and canons	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Maintenance and repairs	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Professional services	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Transportation	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Promotion and publicity	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Supplies	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	**1016,4**
Other external expenses	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Insurance	50	0	0	0	0	0	0	0	0	0	0	0	**50**
Autonomous Workers (+IRPF)	800	800	800	800	800	1.600,00	800	800	800	800	800	1.600,00	**11.200,00**
Workers GENERAL REGIME	0	0	0	0	0	0	0	0	0	0	0	0	**0**
SECT SOCIAL AUTONOMOUS WORKERS	588	588	588	588	588	588	588	588	588	588	588	588	**7.056,00**
Social Seg Workers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Loans	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	**2.380,00**
Immobilized suppliers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Imposition over benefits	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Value Added Tax	0	0	1097,25	0	0	1097,25	0	0	1097,25	0	0	1097,25	**4.389,00**
Total payments	**4.827,71**	**4.777,71**	**5.874,96**	**4.777,71**	**4.777,71**	**6.674,96**	**4.777,71**	**4.777,71**	**5.874,96**	**4.777,71**	**4.777,71**	**6.674,96**	**63.371,50**
Final balance	**11.221,08**	**11.214,91**	**10.111,48**	**10.105,31**	**10.099,13**	**8.195,71**	**8.189,54**	**8.183,36**	**7.079,94**	**7.073,76**	**7.067,59**	**5.164,16**	**5.164,16**

C.3 Third Year

C.3.1 Provisional Results Account

Table C.9: Bills scheduled for the third year by manufacturing 200 ROSNIs.

Bills	Third year
Raw and auxiliary matter purchases	61620
Initial stocks	-
Own remuneration	-
Autonomous insurance	7056
Personal salary or collaborators	22400
H.H. In charge of the company	-
Loan financial expenses	680
Other financial expenses	-
Publicity and promotion	-
Taxes (contributions, fees, etc.)	100
Supplies (light, water, telephone, gasoline, etc.)	840
Rentals	-
Insurance	50
Maintenance and repairs	-
Transports	-
Foreign Services (Management, ...)	-
Constitution expenses (notaries, records, ...)	-
Diverse expenses (stationery, uniforms, restorating expenses, messaging, locomation etc)	-
Material immobilized amortization endowment	-
Intangible immaterial amortization provision	-
Provision provision (unforeseen expenses)	-
Total	**-92746**

Table C.10: Incomes scheduled for the third year by selling 200 ROSNIs.

Income	First year
Sales	94642
Final existence	-
Financial income (loans)	-
Incentives (already granted)	-
Others (extraordinary income)	-
Total	**94642**

Table C.11: Benefits before and after taxes of the third year.

Benefits after taxes (Income - expenses) =	1896
Imposition over benefits	474
Benefits after taxes (25%)	1422 €

C.3.2 Profitability Threshold

Table C.12: Profitability threshold of the third year with the final year balance.

MONTH	1	2	3	4	5	6	7	8	9	10	11	12	TOTAL n
INITIAL BALANCE	**5.164,16**	**5.972,85**	**6.831,53**	**5.495,72**	**6.354,40**	**7.213,09**	**4.277,27**	**5.135,96**	**5.994,64**	**4.658,83**	**5.517,51**	**6.376,20**	**5.164,16**
Charges													
Banking loom	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Customers	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	9.543,07	**114.516,82**
Aid and subsidies	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Total charges	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**9.543,07**	**114.516,82**
PAYMENTS													
Suppliers	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	6.213,35	**74.560,20**
Rentals and canons	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Maintenance and repairs	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Professional services	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Transportation	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Promotion and publicity	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Supplies	84,7	84,7	84,7	**84,7**	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	**1016,4**
Other external expenses	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Insurance	50	0	0	0	0	0	0	0	0	0	0	0	**50**
Autonomous Workers (+IRPF)	1.600,00	1.600,00	1.600,00	1.600,00	1.600,00	3.200,00	1.600,00	1.600,00	1.600,00	1.600,00	1.600,00	3.200,00	**22.400,00**
Workers GENERAL REGIME	0	0	0	0	0	0	0	0	0	0	0	0	**0**
SECT SOCIAL AUTONOMOUS WORKERS	588	588	588	588	588	588	588	588	588	588	588	588	**7.056,00**
Social Seg Workers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Loans	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	198,33	**2.380,00**
Immobilized suppliers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Imposition over benefits	0	0	0	0	0	0	0	0	0	0	0	474	**474**
Value Added Tax	0	0	2194,5	0	0	2194,5	0	0	2194,5	0	0	2194,5	**8.778,00**
Total payments	**8.734,38**	**8.684,38**	**10.878,88**	**8.684,38**	**8.684,38**	**12.478,88**	**8.684,38**	**8.684,38**	**10.878,88**	**8.684,38**	**8.684,38**	**12.952,88**	**116.714,60**
Final balance	**5.972,85**	**6.831,53**	**5.495,72**	**6.354,40**	**7.213,09**	**4.277,27**	**5.135,96**	**5.994,64**	**4.658,83**	**5.517,51**	**6.376,20**	**2.966,38**	**2.966,38**

C.4 Fourth Year

C.4.1 Provisional Results Account

Table C.13: Bills scheduled for the fourth year by manufacturing 400 ROSNIs.

Bills	Fourth year
Raw and auxiliary matter purchases	123240
Initial stocks	-
Own remuneration	-
Autonomous insurance	7056
Personal salary or collaborators	33600
H.H. In charge of the company	-
Loan financial expenses	680
Other financial expenses	-
Publicity and promotion	-
Taxes (contributions, fees, etc.)	100
Supplies (light, water, telephone, gasoline, etc.)	840
Rentals	-
Insurance	50
Maintenance and repairs	-
Transports	-
Foreign Services (Management, ...)	-
Constitution expenses (notaries, records, ...)	-
Diverse expenses (stationery, uniforms, restorating expenses, messaging, locomation etc)	-
Material immobilized amortization endowment	-
Intangible immaterial amortization provision	-
Provision provision (unforeseen expenses)	-
Total	**-165566**

Table C.14: Incomes scheduled for the fourth year by selling 400 ROSNIs.

Income	Fourth year
Sales	189284
Final existence	-
Financial income (loans)	-
Incentives (already granted)	-
Others (extraordinary income)	-
Total	**189284**

Table C.15: Benefits before and after taxes of the fourth year.

Benefits after taxes (Income - expenses) =	**23718**
Imposition over benefits	**5929,5**
Benefits after taxes (25%)	**17788,5 €**

C.4.2 Profitability Threshold

Table C.16: Profitability threshold of the fourth year with the final year balance.

MONTH	1	2	3	4	5	6	7	8	9	10	11	12	TOTAL n
INITIAL BALANCE	**2.966,38**	**6.446,45**	**9.976,52**	**9.117,59**	**12.647,67**	**16.177,74**	**12.918,81**	**16.448,88**	**19.978,95**	**19.120,02**	**22.650,08**	**26.180,15**	**2.966,38**
Charges													
Banking loom	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Customers	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	19.086,14	**229.033,64**
Aid and subsidies	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Total charges	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**19.086,14**	**229.033,64**
PAYMENTS													
Suppliers	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	12.426,70	**149.120,40**
Rentals and canons	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Maintenance and repairs	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Professional services	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Transportation	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Promotion and publicity	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Supplies	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	**1016,4**
Other external expenses	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Insurance	50	0	0	0	0	0	0	0	0	0	0	0	**50**
Autonomous Workers (+IRPF)	2.400,00	2.400,00	2.400,00	2.400,00	2.400,00	4.800,00	2.400,00	2.400,00	2.400,00	2.400,00	2.400,00	4.800,00	**33.600,00**
Workers GENERAL REGIME	0	0	0	0	0	0	0	0	0	0	0	0	**0**
SECT SOCIAL AUTONOMOUS WORKERS	588	588	588	588	588	588	588	588	588	588	588	588	**7.056,00**
Social Seg Workers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Loans	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	**680**
Immobilized suppliers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Imposition over benefits	0	0	0	0	0	0	0	0	0	0	0	5929,5	**5.929,50**
Value Added Tax	0	0	4389	0	0	4389	0	0	4389	0	0	4389	**17.556,00**
Total payments	**15.606,07**	**15.556,07**	**19.945,07**	**15.556,07**	**15.556,07**	**22.345,07**	**15.556,07**	**15.556,07**	**19.945,07**	**15.556,07**	**15.556,07**	**28.274,57**	**215.008,30**
Final balance	**6.446,45**	**9.976,52**	**9.117,59**	**12.647,67**	**16.177,74**	**12.918,81**	**16.448,88**	**19.978,95**	**19.120,02**	**22.650,08**	**26.180,15**	**16.991,72**	**16.991,73**

C.5 Fifth Year

C.5.1 Provisional Results Account

Table C.17: Bills scheduled for the fifth year by manufacturing 500 ROSNIs.

Bills	Fifth year
Raw and auxiliary matter purchases	154050
Initial stocks	-
Own remuneration	-
Autonomous insurance	7056
Personal salary or collaborators	42000
H.H. In charge of the company	-
Loan financial expenses	680
Other financial expenses	-
Publicity and promotion	-
Taxes (contributions, fees, etc.)	100
Supplies (light, water, telephone, gasoline, etc.)	840
Rentals	-
Insurance	50
Maintenance and repairs	-
Transports	-
Foreign Services (Management, ...)	-
Constitution expenses (notaries, records, ...)	-
Diverse expenses (stationery, uniforms, restorating expenses, messaging, locomation etc)	-
Material immobilized amortization endowment	-
Intangible immaterial amortization provision	-
Provision provision (unforeseen expenses)	-
Total	**-204776**

Table C.18: Incomes scheduled for the fifth year by selling 500 ROSNIs.

Income	Fifth year
Sales	236605
Final existence	-
Financial income (loans)	-
Incentives (already granted)	-
Others (extraordinary income)	-
Total	**236605**

Table C.19: Benefits before and after taxes of the fifth year.

Benefits after taxes (Income - expenses) =	31829
Imposition over benefits	7957,25
Benefits after taxes (25%)	23871,75 €

C.5.2 Profitability Threshold

Table C.20: Profitability threshold of the fifth year with the final year balance.

MONTH	1	2	3	4	5	6	7	8	9	10	11	12	TOTAL n
INITIAL BALANCE	**16.991,72**	**21.536,65**	**26.131,58**	**25.240,26**	**29.835,19**	**34.430,12**	**30.538,80**	**35.133,73**	**39.728,66**	**38.837,34**	**43.432,27**	**48.027,20**	**16.991,72**
Charges													
Banking loom	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Customers	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	23.857,67	**286.292,05**
Aid and subsidies	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Total charges	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**23.857,67**	**286.292,05**
PAYMENTS													
Suppliers	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	15.533,38	**186.400,50**
Rentals and canons	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Maintenance and repairs	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Professional services	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Transportation	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Promotion and publicity	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Supplies	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	84,7	**1016,4**
Other external expenses	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Insurance	50	0	0	0	0	0	0	0	0	0	0	0	**50**
Autonomous Workers (+IRPF)	3.000,00	3.000,00	3.000,00	3.000,00	3.000,00	6.000,00	3.000,00	3.000,00	3.000,00	3.000,00	3.000,00	6.000,00	**42.000,00**
Workers GENERAL REGIME	0	0	0	0	0	0	0	0	0	0	0	0	**0**
SECT SOCIAL AUTONOMOUS WORKERS	588	588	588	588	588	588	588	588	588	588	588	588	**7.056,00**
Social Seg Workers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Loans	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	56,67	**680**
Immobilized suppliers	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Imposition over benefits	0	0	0	0	0	0	0	0	0	0	0	7957,25	**7.957,25**
Value Added Tax	0	0	5486,25	0	0	5486,25	0	0	5486,25	0	0	5486,25	**21.945,00**
Total payments	**19.312,74**	**19.262,74**	**24.748,99**	**19.262,74**	**19.262,74**	**27.748,99**	**19.262,74**	**19.262,74**	**24.748,99**	**19.262,74**	**19.262,74**	**35.706,24**	**267.105,15**
Final balance	**21.536,65**	**26.131,58**	**25.240,26**	**29.835,19**	**34.430,12**	**30.538,80**	**35.133,73**	**39.728,66**	**38.837,34**	**43.432,27**	**48.027,20**	**36.178,63**	**36.178,62**

Appendix D

GitHub Repository: our software container

All code has been uploaded in our GitHub organization. There are 2 repositories. On contains all the ROS Packages and nodes, and the other one contains all the code used to develop the App.

Here you have the link to our organization: `https://github.com/ROSNI-educational-robot`

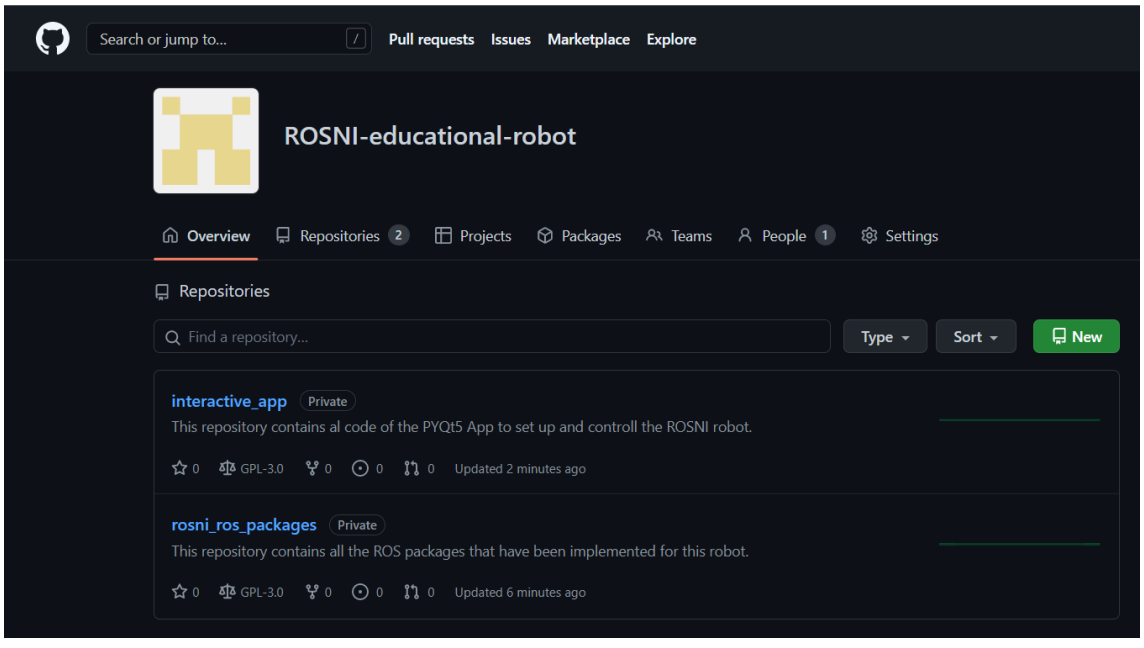


Figure D.1: How our GitHub organization looks like.