



UNIVERSITAT DE BARCELONA

Final Degree Project
Biomedical Engineering Degree

**“Development of an AGV robot based on ROS for
disinfection in clinical environments.”**

Barcelona, 14 June, 2021

Author: Marta Córdoba

Director/s: Dr. Llorenç Servera, Dr. Manel Puig

Abstract

The aim of this project was to develop a final degree project in the field of biomedical engineering that consisted in an AGV prototype for disinfection tasks inside a hospital or clinical environment. There were different parts inside the project, and the one reported in this documentation refers to the mobility of the robot using ROS (Robot Operating System) to develop nodes to make the robot move as an holonomic robot. An holonomic robot refers to systems with capabilities to slide directly sideways, which is very useful for complex spaces, as it can reach the objectives easier.

Simulations were done in order to verify the code was working and once they were proved to be successful, the code was implemented into a small prototype for testing, whose characteristics and components are specified in this report. This was a complex step as several errors were reported, which caused a delay in the last activities, making it not possible to achieve implementing SLAM routines for mapping and trajectory planning.

The other parts of the project consisted in designing and building a bigger prototype with air quality and ozone sensors and the development of the code for the sensors. These two parts were developed by two other engineering students.

This project was developed as a collaboration project between EUSS School of Engineering and Universitat de Barcelona.

Contents

1. Introduction	5
1.1. Motivation	5
1.2. Objectives	5
1.3. Software.....	6
1.4. Scope and span	7
1.5. Methodology and report structure	8
2. Antecedents	9
2.1. Antecedents	9
2.2. State of the situation	9
2.3. State of the art	10
2.4. Information sources	11
3. Market Analysis.....	12
3.1. Sectors to which it is directed.....	12
3.2. Historical evolution of the market.....	13
3.3. Future perspectives of the market.....	14
3.4. Product environment.....	14
4. Engineering design	15
4.1. Study on different solutions.....	15
4.2. Proposed solution	23
5. Detail Engineering.....	24
5.1. Organization of the project.....	24
5.2. Flux diagram	25
5.3. Software.....	26
5.4. Hardware	29
5.5. Code description	37
5.6. Results verification.....	38
5.7. Limitations and improvements	47
6. Execution timelines	47
6.1. Work Breakdown Structure (WBS).....	47
6.2. Task matrix	50
6.3. Gantt.....	51
7. Technical feasibility	52
7.1. SWOT Analysis.....	52
8. Economic feasibility	53

9. Regulations and legal aspects	55
10. Conclusions and future work.....	56
11. Bibliography	58
12. Annexes.....	62
12.1. Tutorial: How to export a SolidWorks Assembly into an URDF file. Gazebo visualization.	62

1. Introduction

Robotics is an engineering field that consists of developing, designing, and programming robots that can produce many tasks. One of the potential fields in which robotics is being developed and there have been very significant signs of progress in the last years is the biomedical field. In this project, programming tools will be used to establish nodes that will communicate with a biomedical device using Robot Operating System (ROS). The software will be designed to be used in an AGV (Automatic Guided Vehicle) robot. The function of this robot will be involved in measuring and improving air quality in a hospital; the research of the needed components for the prototype and the software was previously done in another project. This report is written as a final degree project of Biomedical Engineering at Universitat de Barcelona.

1.1. Motivation

I started studying Biomedical Engineering because I thought it was the perfect combination between engineering, healthcare and biomedicine. During these years, I realized I was more interested in the engineering and electronic part of the degree. When choosing my final degree project, I knew I wanted to be involved in a project to improve my programming skills, especially python, since it is the programming language I preferred.

Between the projects I found, this one was the one that most adjusted to my needs and interests and allowed me to learn from a new field. I did not know that there was such a wide range of career opportunities involving robotics apart from the well-known Da Vinci surgery system. After finishing the project, I firmly believe robots are the future of hospitals; in many different tasks and chores, they can be accommodating for the sanitary professionals and help stop the spread of diseases.

Also, another part of the project that I liked is that it involves teamwork and different disciplines, which means I was able to work with engineering students from other universities and specialities. It allowed me to learn about the whole project, including the parts developed by other students and has helped me develop my teamwork abilities.

1.2. Objectives

The main objective of this project is (i) **to develop** the necessary code for the mobility and obstacle avoidance of an AGV robot that will work in a clinical environment using ROS, Python, VS Code and a Linux virtual machine. The function of the final developed robot is to move around the hospital using autonomous navigation and measure air quality. If it is considered to be not clean enough, emit UV light and ozone to improve it. There is, therefore, a more significant project that comprehends different parts:

- (i) To design and build the robot platform and to assemble the components
- (ii) To develop an air quality measurement system
- (iii) To start up the SLAM routines in the ROS environment and make a fusion from odometry and Lidar data.

This project consists of the third part. The first and second parts were developed simultaneously by other engineering students to put together the work done to obtain the final product.

Also, there are other specific objectives included in this project, which involve both learning objectives and developing and achievement objectives:

- Develop a final degree project in the field of Biomedical Engineering:

It mainly involves learning objectives, including elaborating a report with sections corresponding to an engineering project that accurately explains all the steps.

- Get familiar with the different environments used and learn how to manage the various tools.

The different environments include all software: VirtualBox, Ubuntu, Visual Studio Code, Arduino, Raspberry Pi 4 and all the communications between them.

- Understand ROS functionalities and which are helpful on an AGV robot

Learn how nodes communicate between them, what topics, messages, how the complex ROS environment works, and the code structure. Also, it is essential to determine which packages are necessary to develop an AGV robot concretely.

- Test the developed code on a virtual environment using simulations

Before testing the code directly on the final prototype, it is essential to verify that it carries out the desired processes. This objective also includes obtaining a URDF file of the robot prototype, and while it is not achieved

- Test the developed code on a small device using Arduino

It includes building a small prototype equipped with Raspberry Pi, Arduino and wheels and motors similar to the user on a final prototype and testing the software on the small device for a later integration to the bigger robot, which is equipped with the sensors and other extra functionalities.

- Develop teamwork skills: communication, organization, problem-solving, listening.

This specific objective is because the project involves other students and professors, and it is needed to establish fluent communication and coordination to produce the final product.

1.3. Software

ROS

ROS stands for Robot Operating system and consists in a platform that allows developing robotic software. The processing takes place in nodes that can send and receive messages that can contain a wide variety of information types. This operating system was developed in 2007 by Willow garage, a company dedicated to open access robotic software development [1].



Figure 1. ROS logotype

ROS allows the creation of software application by providing libraries and tools that make the development of robot software easier. One of the essential features of this platform is that it enhances sharing the work done in this field and collaboration between experts, simplifying the process as one's work can be built upon previous projects done by others. Robotic Operating System allows carrying out many functions such as hardware abstraction, low-level device control, device drivers, libraries, visualizers, message-passing, package management, etc. [2]

ROS can be implemented in many programming languages such as Python, C++, and Lisp. In this project, the chosen programming language is Python. To allow users the connection between ROS and Python there is a package called *rospy*. A more detailed description of ROS functioning is provided in the section *Detail Engineering*.

VirtualBox

ROS is a handy software; however, it only works on Linux; therefore, a Linux virtual machine is needed in case the computer employed does not use this OS. VirtualBox works on Windows and can be used if a specific operating system is required punctually. In this case, Ubuntu 18.04 is installed on the virtual machine installed on the computer's Windows OS. This allows using ROS in a Windows (or macOS) computer. Figure 2 shows the parameters used in the virtual Ubuntu 18.04 machine.

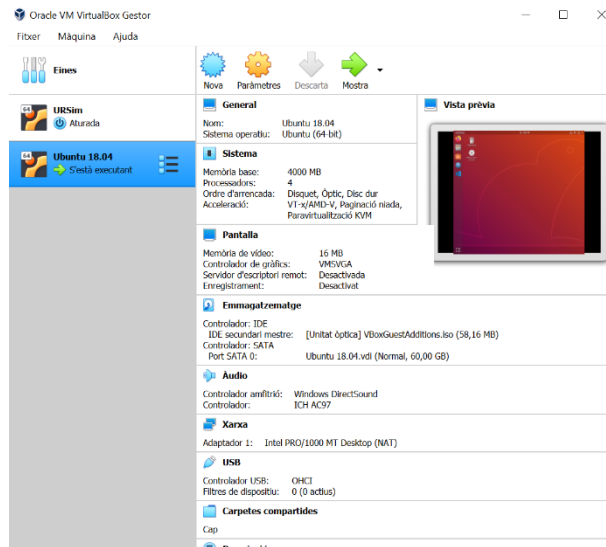


Figure 2. VirtualBox environment

Visual Studio Code

VS code is a handy tool that allows cloning and synchronising GitHub repositories, which is very convenient when working in a team with several people. VS Code allows the visualization of a wide range of file formats (.pdf, .py, .ipynb, .urdf, ...) and the structure of the folders and subfolders of the documentation packages. This environment also has access to opening and managing terminals to navigate, launch and create a file. Many tools are available called extensions that allow the manipulation and visualisation of the different types of files. Figure 3 shows VS Code environment on the virtual machine. The left part shows the structure of the folders, and the right side is used to visualise the files, in this case, a python Jupyter notebook. At the bottom of the screen, a terminal can be seen.

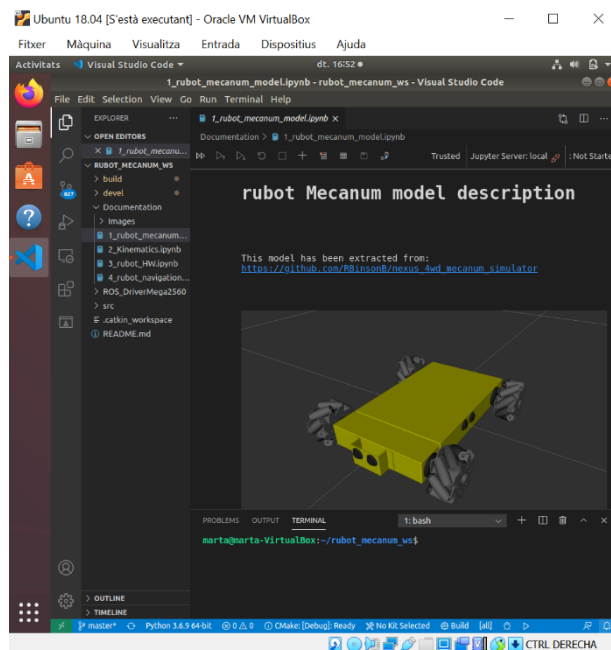


Figure 3. VSCode in VirtualBox

1.4. Scope and span

As mentioned in the objectives section, one of the goals of this project is to develop a final degree project. Therefore, this project does not involve the commercialization of the biomedical product, but the development of the software needed for the device. However, the regulations required for marketing the final product have been studied. This project does not intend to build the final

prototype, but it is focused on developing the code using Python and ROS and testing it on simulations and a small prototype. However, the development of the prototype is being done in another final degree project and the connections of air quality and ozone sensors to databases that store their readings.

The software developed in this project will be designed to be used in an Automatic Guided Vehicle. The robot's objective is to move around a hospital (or other clinical environments), measuring and, if needed, improving air quality. Even though it is an application that can be used in many industries, this project is focused on the application in hospitals. To do so, this AGV will be designed to work in Hospital Clínic, in Barcelona, but later it can be extended to other hospitals and even other purposes of an AGV robot.

The final degree project was carried out from February to June 2021, with a presentation at the end of the period.



Figure 4. Universitat de Barcelona logo (left) and EUSS School of Engineering logo (right)

As a final degree project, it was supported by Universitat de Barcelona, concretely the work was supervised by professors of the Department of Electronics and Biomedical Engineering. However, it was developed at EUSS School of Engineering (Escola Universitària Salesiana de Sarrià), becoming a collaboration project between both institutions.

1.5. Methodology and report structure

This report is structured in the sections that appear in the index (see page 2). After the introduction, which includes the objectives and motivation of the project, the area of antecedents is presented. In it, the robotic applications in medicine and the previous work done are analysed. After the next section is dedicated to market analysis, the sectors and potential users of the technology developed in this project and the past and future of the market in this field.

The design engineering section presents the different solutions that were proposed to develop the project, including all other parts (hardware and software). The proposed solution is explained at the end of this part. Then the Detail Engineering describes the implementation of the explanation in-depth, followed by the timelines followed during the project showing the organization in a time of the different tasks, which changed along with the project due to problems that came up and required to extend the time of some activities.

One of the last sections is an analysis of the economic feasibility regarding all the parts of the project. Also, the regulations and standards that apply to the final product are considered in the Regulations and Legal aspects part.

Finally, there is a conclusions section that relates the objectives with what was achieved as well as a bibliography and extra annexe sections.

The methodology consists of research on the functioning of ROS and Raspberry Pi 4 and understanding the code, and testing it on simulations and a small device before verifying its functioning on the final prototype. This project was carried out at EUSS School of Engineering and the Department of Electronic and Biomedical Engineering at the University of Barcelona.

2. Antecedents

2.1. Antecedents

As it has been said, this project is the continuation of a previous final degree project that mainly focused on the design of the prototype and the description of the components and software, as well as the study of the application and developing localization and mapping code routines (SLAM).

However, the objective of the final product has changed: the work done previously intended to design an AGV robot to develop transportation tasks, but currently, the goal is to achieve a robot with autonomous navigation that analyses and, if needed, improve air quality in clinical environments.

In this previous project [3], the conclusions stated that the components described were compatible and that AGV has a great future in hospitals, mainly getting the nursing professionals out of tasks that are not focused on health to dedicate their time to patient quality care. It was also concluded that a perfect tool to start developing the ROS code was Turtlebot3, which is a personal robot kit that can be used for educational, research and hobby purposes related to robot programming. However, in this new stage of the project, a different robot has been used, which is described in further sections of this document. This change was that even though Turtlebot3 was an excellent tool for getting started in mobile robotics, it did not allow the freedom of changing the components for different ones, such as the wheels, sensors and motors. The platform and other parts of the robot were developed by another student in his final degree project.

2.2. State of the situation

The first AGV robot was a modified tractor programmed to follow an overhead wire [4]. Since then, the industry has been growing, and now they have applications in many fields in transportation, storage, assembling and other tasks. AGV robots have already been introduced to the healthcare field with functions related to food transportation to the patient rooms, collection and transportation of used and clean laundry and waste transportation [5]. The AGV robot developed by Jaiganesh et al. was not programmed with ROS, but it did have the functions that the robot developed in this project is expected to have.

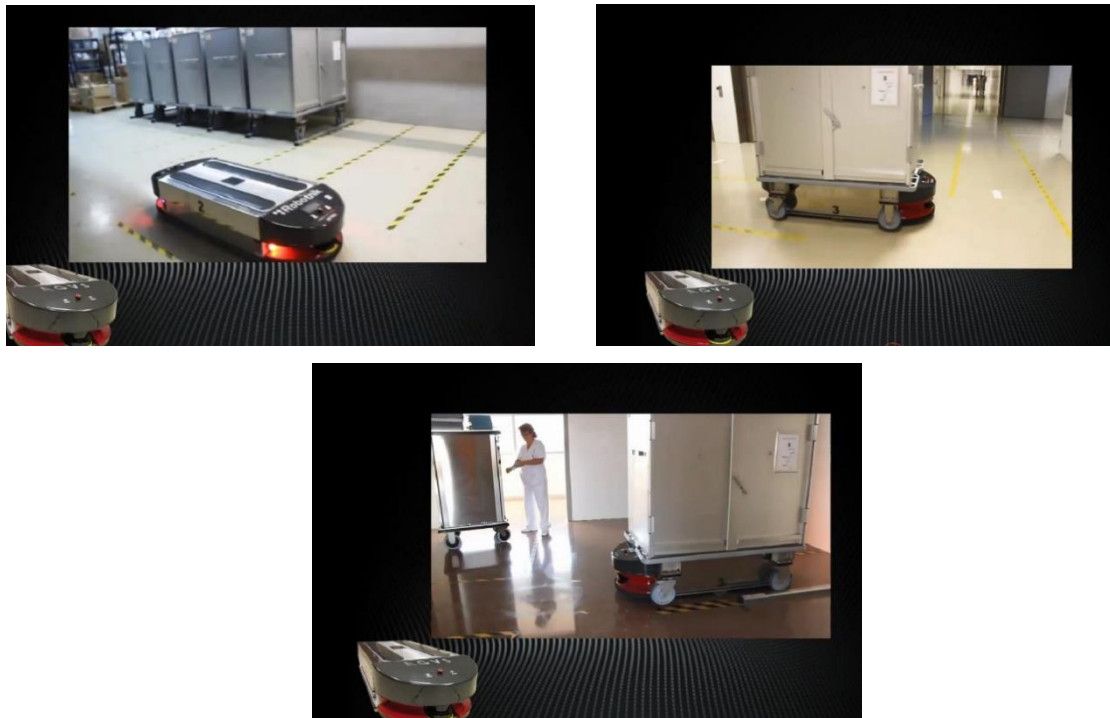


Figure 5. Robotnik AGV robot in a hospital environment [6].

Robotnik is a company that in 2014 launched an AGVS platform, and this AGV robot system has already been implemented in some European hospitals doing transportation tasks. It uses ROS architecture for simulation and control, and the designed software is available on the official ROS website. The robot moves around the hospital and carries trolleys that need to be transported to other locations inside the hospital. The robots' localisation system consists of *amcl*, a localization system for robots moving in 2D, and magnetic landmarks and two lasers that scan the environment and provide safety preventing collisions with people who work in the Hospital [7]. It was built with many packages, including Gazebo, VREP and Navigation packages. The users can program commands to be carried out at a specific time or add them to a queue, and these commands or missions are managed by FCS (Fleet Control System). Most of the AGV working in hospitals are platforms that can carry trolleys and oversized loads.

2.3. State of the art

Applications of robotics in the medical field extend to a wide variety that includes both arm robotics and mobile robotics. There are currently six main applications of robotics in medicine and healthcare: telepresence, surgical assistants, rehabilitation, medical transportation, sanitation and disinfection and robotic prescription dispensing systems.

Telepresence robots in healthcare enhance the interaction between sanitary professionals and patients who live in remote or rural areas using live video and audio. It is beneficial for older adults because the population is ageing, and soon healthcare personnel will not be enough to assist all the elderly population. These telepresence systems can also be helpful for older people to interact with their relatives and friends and cope with isolation or loneliness. The robots usually consist of a platform with wheels that support a screen or tablet and a robotic arm. Some of these systems have monitors to check the patient's vital signs and the videoconference system, which allows doctors and nurses to encourage patients to do daily activities independently. The acceptance of these types of robots might be difficult because the elderly population is the least used to new

technologies. However, every day more people are starting to use them and probably, in the coming years, the popularity of these systems will increase [8].

Robots involved in surgeries can have different roles; however, currently, no systems can operate by themselves. The systems can be supervisory-controlled systems, telesurgical systems and shared-control robotic systems. The most popular ones are telesurgical systems, which is the DaVinci robotic system. DaVinci is an operator with two parts: a console, where the surgeon sits and performs the movements and a body with different robotic arms that hold the various surgical tools that copy the surgeon's movements on the console. The advantage of these systems is that they allow operating from a distance, thus having fewer people in the operating room while maintaining all the benefits of laparoscopic interventions. Also, it enables more degrees of freedom of the tools due to an EndoWrist movement that permits a rotation of almost 360°, simulating the human hand movements and 3D vision is possible, giving the surgeon the perception of depth that was not feasible in laparoscopic interventions. Another advantage of these systems is that they allow performing minimally invasive surgeries, which have many benefits for the patient during recovery; also surgeons do not need to sterilize their hands, and the system confers excellent stability to the surgeon, avoiding trembling.

Medical transportation robots and sanitation and disinfection robots are both AGV robots that perform autonomous navigation and whose purpose is obstacle avoidance, while their goals are different. Medical transportation robots are involved in transporting tasks inside the hospital, and they are handy for heavy loads such as trolleys that carry food trays or dirty laundry, they can be directed to a specific location of the facilities. And with different programming, the places can be ordered according to the emergency degree. These robots are usually platforms that attach to trolleys, not big trolleys themselves, and there is a specific place of the hospital where they are directed when the batteries are low. Regarding sanitation and disinfection tasks, the robots are also AGVs, but the feature of leading them to a specific location is not as helpful. In this application, it is more interesting that robots can scan the rooms to detect air quality and make sure the robot inspects all rooms. It would be interesting to consider uniting both applications to make a robot that can carry both tasks simultaneously to scan and sanitize the rooms where it is required to bring something. However, it would be difficult to guarantee that all rooms would be scanned.

Robotic prescription dispensing systems were developed to avoid dispensing the wrong medication to patients, preventing severe reactions and even deaths. These also free up pharmacist's time which allows them to focus on more critical tasks. The tasks of these systems include storing medication, counting pills and labelling prescription vials, among others. The robots are massive storage containers controlled by a computer and work similarly to a vending machine. They can handle up to 200 medications and patient information simultaneously, are faster than humans, increase efficiency and increase productivity. Pharmacy staff had to count pills and label medications manually before these robots existed, giving them a great value because they can spend their time in more practical and essential activities [9].

2.4. Information sources

The leading information sources are the official ROS webpage (<http://wiki.ros.org/>) for the tutorials, articles available online and official webpages of the Raspberry system and others. Another vital information source is the final degree project that has been done before this one. The articles consulted were found using generic search engines, and all were available online for free or by using the SIRE button, which allows access to e-resources. Also, information from the Robotics

and Control of Biomedical Systems course was used as a programming guide. A GitHub repository¹ was used to share the information, code and other documents used during this project.

3. Market Analysis

3.1. Sectors to which it is directed

This section will explain which sectors are directed both the developed code and the AGV robot once it is fully functional.

3.1.1. The AGV robot

In this project, the robot is intended to travel autonomously through a hospital's corridors and rooms. It can be used in many areas and even some parameters can be changed depending on the demanding air quality in each zone. The robot will be designed considering the characteristics of the environment of Hospital Clínic, in Barcelona, studying logistical issues.

However, the robot can be used in other hospitals interested in integrating this technology to their facilities. Therefore, it is directed to the sanitary sector, even though cleaning the air might be interesting for many other industry sectors, even offices where there are several people working. Acquiring this technology can be very interesting for many different sectors to decrease the spread of viruses and bacteria, especially during this COVID-19 pandemic.

3.1.2. The developed code

As it has been mentioned, the software will be directed to an AGV robot that is expected to work in a clinical environment. To elaborate the code and nodes, the robot used was developed specifically for this project, even though for the initial stages of the code, a nexus robot model was virtually used in gazebo and rviz environments. Therefore, the nodes are directed to robots that use this developed robot, even though modifications and changes of the code might be needed for different applications or components of the robot. Even though some changes might work for other projects that use this robot, it is not an objective of this project to reach others but to develop the code, particularly for this project.

Also, AGV robots are widely used in many industries for transporting tasks, especially for weighty loads. It makes these tasks more straightforward, and some risks are reduced as workers do not have to lift the heavy loads themselves, and the incidents related to the drop of the loads are reduced. This is not the only advantage they have; they can also liberate workers from tasks that are not directly related to their profession, giving them the chance to dedicate more time to the tasks they are meant to do. In the medical field, the nursery is a perfect example of professions that spend time on jobs not directly related to their function. The elaborated code developed in this project can also be used for AGV robots in transporting tasks. In clinical environments, nurses can dedicate more time to patient care than other tasks, such as spending time going to the warehouse to get supplies transporting samples. It is one example of further clinical studies in which the code can be used, but it can actually work for any autonomous navigation task with the appropriate modifications.

¹ https://github.com/manelpuig/rUBot_mecanum_ws

3.2. Historical evolution of the market

The first-ever Automated Guided Vehicle was a modified tractor that followed wires, and it was released in 1953, being the first truck with no driver. After this, the available and growing technology allowed the development of more complex, new, and better AGV robots that were introduced to the industry massively twenty years after this first release. The first industry that integrated this technology was the automotive sector [10]. The robots have been evolving, and while the first robot used a magnetic wire, now there are also optical sensors that follow coloured lines on the floor, laser distance sensors (LDS), etc.

In hospitals, AGVs were introduced during the 80's and have been used for transporting jobs, which allows sanitary professionals to focus more on patient care. This is not the only advantage that AGVs have; another important aspect is that they reduce the propagation of infectious diseases as they are spread due to people moving and robots reduce this movement of people and do not touch all the hospital surfaces. Moreover, nursing personnel have less contact with potentially dangerous samples as they are not managing them. These robots can help save time from going to the warehouse in an emergency of lack of material. AGVs are designed not to solve medical problems but logistics problems in hospitals.

Nowadays, there are many examples of AGV robots that work in a clinical environment:

- EK Automation: this company offers highly secure vehicles for the healthcare sector as they are in contact with people as well as a hygienic design using the appropriate materials, also, the robots are able to use lifts to work in different floors of the facilities [11].



Figure 6. EK Automation's AGV robot for the healthcare sector [11]

- MIR: this company offers several models of AGVs that have different maximum load capacities and different sizes to adapt to the application and space where it is meant to work. It is not focused specifically on the healthcare or clinical environment [12].



MiR1000



Designed to optimize internal transportation of pallets and heavy loads

MiR500



Automate your internal transportation of heavy loads and pallets easily and cost-effectively

MiR250



Smaller footprint and height for narrow places with up to 250 kg payload!

Figure 7. Different models of MIR AGV robots that can be used in hospitals [12]

- Diligent robots: this company offers a different robot than the previous ones; it is not a platform that carries loads but a robot that has a compliant arm and hand that allows it to grab materials, supplies, and samples itself. This robot is called Moxi, and it has social intelligence [13].



Figure 8. Moxi robot performing a delivery of supplies task [13]

- Another application of AGV robots in hospitals is ultraviolet disinfection, which can kill harmful bacteria and improve patient safety. One company that offers robots with this function is Wellwit Robotics, their devices use pulsed light to disinfect risky contagion areas in hospitals, but they can also be used in airports, malls, offices, etc [14].



Figure 9. Ultraviolet disinfection robot [14]

3.3. Future perspectives of the market

The future of AGVs in a clinical environment is uncertain nowadays. Still, as mentioned in the report, the preferred application of these is transporting and carrying big load trolleys through the hospital. There are other applications for which these devices are being developed, such as cleaning and disinfection. Still, it seems that the market will grow in transporting tasks as in different industries such as the automotive sector.

However, it is exciting to explore other applications such as disinfection tasks or other chores that require the robot's autonomous navigation. Also, new features could be added to the robot and increase its functionalities; some of these characteristics are explained in one of the last sections of this document *Conclusions and future lines*.

Regarding other clinical applications, it seems that robots still have a wide range of activities to cover and will probably grow in all medical fields, including surgery, transporting, disinfecting and even carrying patients around the hospital.

3.4. Product environment

This part describes the political, economic, social and technological environment in which this project is being developed. It considers different factors of each briefly studied field and weigh whether the conditions are favourable or not. Currently, the most advanced countries in robotics

are the United States, Japan, Germany and China. Spain is in the top 10 countries in robot density, according to N. Blanco [15].

3.4.1. Political environment

Robotics is a field that has grown massively during the last years, and this is why it is needed to establish new regulations and solve legal issues concerning them because they are becoming autonomous in some tasks and activities. The fear is that their results are often unpredictable. In the European Union, some organisms have taken the initiative to start making and updating regulations that involve robotics and their ethical development and basic directives that consider them. These institutions are the European Parliament and the European Commission and the European Economic and Social Committee. [16]

3.4.2. Economic environment

According to the World Economic Forum, robots will destroy up to 85 million job positions in the following five years. Due to the COVID-19 pandemic, the digitalization and automatization of work has been rapidly accelerated with the integration of robots in all fields. In general, it is said that jobs are being eliminated faster than they are being created, and half of the jobs that will not be destroyed will probably require new skills that will have to be developed. At the same time, however, new jobs are being created due to the robotic revolution, which is said to be the 4th Industrial Revolution. These new jobs, include Artificial Intelligence (AI). [17]

3.4.3. Sociocultural environment

Technology is sometimes hard to implement. It is difficult to obtain the population's acceptance of the new developed technologies, especially when substituting tasks usually done by a human being [18]. The fear to the unknown, losing jobs and other factor such as media (TV shows and films) play a significant role when it comes to tolerating new technologies in the occidental cultures, while countries like Japan have always seen integrating robots into different fields as a very positive aspect [19]. However, the new generations are more used to new technologies, and our society's perspective of robots is changing, and in the coming years, researchers will continue developing and integrating them with human interactions and tasks. Also, education helps to increase the acceptance of these technologies.

3.4.4. Technologic environment

The technology environment in which this device is being developed is very favourable and promising because technology is arising in all areas in a quick way. Also, lots of open-source code sites are available, and many other AGV robots are being developed nowadays and disinfecting robots and new and more accurate sensors. There are many possibilities of tools available for developing the product, and cutting-edge-technology components are being created every day.

4. Engineering design

4.1. Study on different solutions

This project has different parts, which means there were other solutions to propose for the different parts.

4.1.1. Robot application

As mentioned previously, many clinical applications can take advantage of an AGV robot, some of which have been presented in previous document sections. The following applications were proposed for this project:

- AGV robot involved in transportation tasks. One of the solutions proposed consisted in building a robot with different drawers that would transport medical supplies such as samples, bandages, syringes, etc. The main tasks of robots in the market involve transportation. However, they are focused on big trolleys that carry food trays, laundry and other heavy loads. Therefore it would be an excellent opportunity to design a robot involved in transporting smaller items as the previously mentioned ones because they are not as present in the market. The proposed robot would have 4 drawers, each of them for a different purpose. The last drawer would be adapted with lead walls to carry radiotoxic samples in a nuclear medicine unit. This project would allow the nurses and nursery assistants to spend less time going from one place to another, which would give them more time to spend on patient care. Also, it would decrease the movement of people inside the hospital, and therefore less infectious diseases would be spread, which is very interesting in a pandemic. The transportation of tasks would liberate sanitary professionals from carrying toxic samples, therefore avoid illness developed by these. However, there are also some disadvantages which include sanitary professionals not adapting to the new technology. Also, it needs to be taken into account that the robot would contain samples of real patients and that the data of these patients must be confidential and only accessible by sanitary professionals. Therefore it must be guaranteed that the drawers cannot be open by someone that is not allowed and finds the robot in a hospital corridor. Another disadvantage is that the robot would be functioning during the day mostly, when patients and their relative are around the hospital, and it is possible that, even though it would be programmed to avoid obstacles, people could unintentionally hit it.

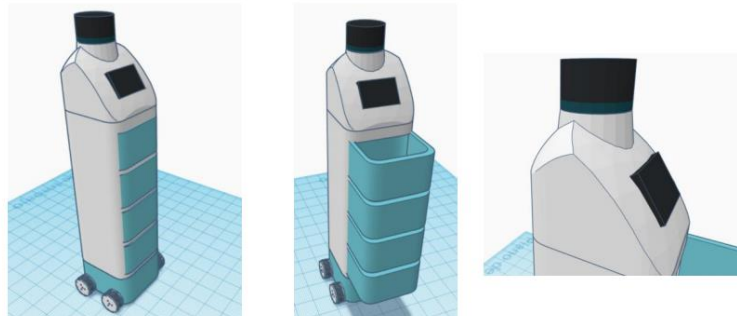


Figure 10. Different views of the proposed robot modelled with tinkercAD [3]

- Air quality detection and disinfection. This solution consists in implementing different sensors that would scan different air parameters such as CO₂ concentration, temperature, humidity, etc. With this information, if the air quality was considered to be not good enough, ozone would be expelled in order to improve it and therefore disinfect it. This solution is exciting specially during an epidemic episode, such as the current COVID-19 pandemic. It reduces infectious diseases and allows having better conditions at hospitals. A significant advantage of this application is that hospitals could reduce the spread of diseases inside the facilities, which would be very good for patients, especially for critical patients in the ICU. Similar to the previously proposed application, it is not a very common application for AGVs which would be a good opportunity in the market. Moreover, this can be applied to

many other sectors and buildings such as offices, industrial facilities, factories and even homes. Another benefit is that the disinfection could occur at night, which means the hospital would not be crowded, and there would be less risk of people colliding with it. Also, this is a less complex solution, and less security would be needed as it would not carry any patient information. Another advantage is that the received data would be stored, and different statistical analysis could be performed, such as finding relations between variables and time or between variables. The main disadvantage of this application is that patients and sanitary professionals would need to leave the room during the disinfection because when certain amounts of ozone are reached, it becomes dangerous for human beings.

4.1.2. Robot platform

The code had to be tested on a small robot before trying it on the final prototype, and there were different options of platforms or robot kits to check the code was well developed. The other considered options were retrieved from Aliexpress (<https://www.aliexpress.com/>) the following:

- ROS SLAM Robot Mecanum Wheel Car Chassis with Lidar Raspberry Pi Navigation with DC 12V Motor DIY Arduino STEM Program Toy parts

This kit has almost all the necessary components, even though they can be changed to more complex parts if desired. It includes the frame, motors, wheels, motor brackets, screws and couplings, an Arduino 2560 board, a shield to connect the motors, the LIDAR sensor and cables required. A version of the kit includes Raspberry Pi 3 or 3B, but in this case, it was wanted to develop the project with a Raspberry Pi 4 due to its faster processor and more and faster RAM. Therefore, the proposed solution was the kit that did not include Raspberry Pi 3 or 3B nor its accessories.



Figure 11. First proposed solution for small testing device [20]

As seen on Figure 11 this robot has two platforms, on the top platform, the LIDAR is fixed and between the two, there are the PCBs and wiring. This is useful because the prototype is small enough to use it as a testing device and at the same time, there is space for the LIDAR sensor and the wiring is not exposed.

Also, the kit does not include a battery, but the developers do offer an installation manual and source code including the requirements of the components that are not provided. The price of this kit is between 230 and 280€, and free shipping to Spain from China.

- Four Wheel Mechanical Arm Car Open Source Code App Control Automatic Obstacle Avoidance Intelligent robot.

The advantage of this option is that it has a low price compared to the others, it costs between 70 and 85 €. However, this platform does not include mecanum wheels, and the specifications are not as accurate; for instance, they do not expose the included components of the kit.



Figure 12. Second proposed solution for small testing device [20]

- New 5kg Load Double Chassis Mecanum Wheel Robot Car Chasses Kit with 4 pcs 12V Encoder Motor for Arduino Raspberry Pi DIY STEM

Another option was to buy a platform and all the components separately, such as batteries, LIDAR, Arduino board, shield to connect the motors, etc. This platform is bigger than the previous ones, it also has two floors, but between platforms, there are the motors, and all the wiring could get tangled with the wheels or motors if it is fixed on this bottom platform. If the components were fixed on the top platform, there could probably be a lack of space, and elements should have to be stacked.

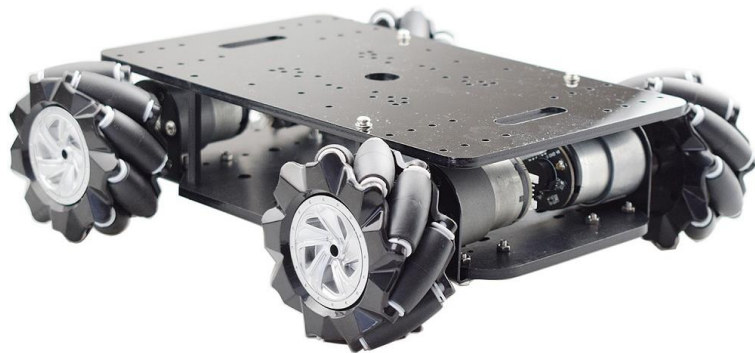


Figure 13. Third proposed solution for small testing Device [20].

- OSOYOO 4WD Omni Wheels Robotic Mecanum Wheels Robot. Automobile Platform chassis with velocity codifying motor for Arduino/Raspberry Pi/ Micro: bit DIY

This platform was found in Amazon (<https://www.amazon.es/>) and it was created by the brand OSOYOO, which produces robotic kits for different applications. This platform is intended to be used to learn programming and mobile robotics. However, it only has one platform and it is possible that not all the components fit on it, so they would need to be stacked, which is not a feasible possibility, or another platform should be



Figure 14. Fourth proposed solution for small testing device [21]

bought. The kit includes mecanum wheels, motors and the platform. Arduino, Raspberry Pi 4, LIDAR and their corresponding wires should be bought separately.

Apart from smaller platforms for a testing device, more powerful and more professional robotic platforms were studied to be later used on a prototype, including the sensors and actuators. All the presented options were 30x40cm approximately, with a wheels diameter of around 120 or 150mm. The considered items were the following:

- Nexus Robot

This company offers several robots that can be adapted to different applications, from this firm, the following platforms were considered:





				
Skus	RB-Nex-03	RB-Nex-06	RB-Nex-33	RB-Nex-04
Price	USD \$770.00	USD \$654.00	USD \$276.00	USD \$624.00
Rating	★★★★★	★★★★☆	★★★★★	★★★★☆
Add to cart	<input type="button" value="ADD TO CART"/>	<input type="button" value="ADD TO CART"/>	<input type="button" value="ADD TO CART"/>	<input type="button" value="ADD TO CART"/>
Product Code	RB-Nex-03	RB-Nex-06	RB-Nex-33	RB-Nex-04
Brand	Nexus Robot	Nexus Robot	Nexus Robot	Nexus Robot

Figure 15. Comparison from Robotshop (<https://www.robotshop.com/>) between different Nexus platforms [22]

These platforms range from 280 to 815€ approximately and have very different sizes. The most appealing items for this project were RB-Nex-03 and RB-Nex-04, which have mecanum wheels and have similar properties to the smaller prototypes proposed to test the code.

- Moebius Tech Store: shock-absorbing.

This company offers an Omni Mecanum-Suspension absorbent of 40kg load, wheels, chassis with 24V Arduino controller. It includes a shock-absorbing kit, four mecanum wheels, four 24V motors, cables for the motors, screws, and optionally a controller kit for a Bluetooth application. Batteries are not included but it is almost already fully assembled. This information was retrieved from

<https://www.aliexpress.com/>,

where the price is between 595 and 760 €. This platform has the

extra feature of integrating the shock-absorbing kit and even though it is a very interesting approach, taking into account that the robot is intended to work inside a hospital or other clinical environments where the floors are flat and with no holes or big imperfections, maybe it is not an interesting feature for this application.

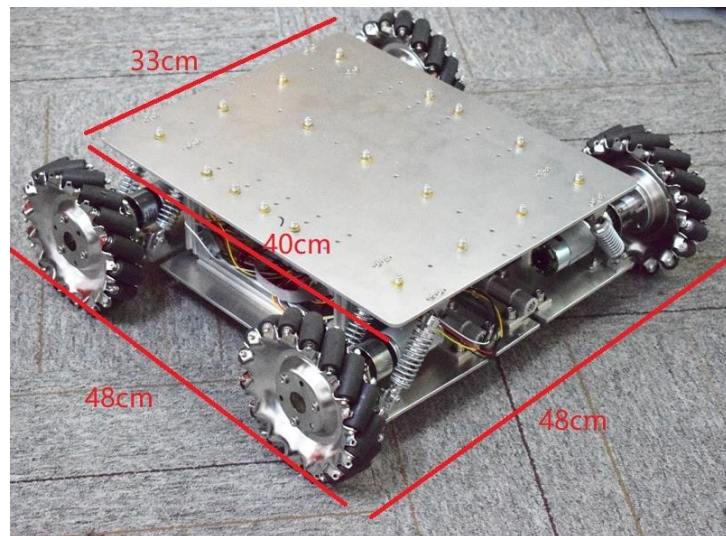


Figure 16. First Moebius proposed solution for final prototype [20]

- Moebius Tech Store: high-resistance wheels

The prototype offered by this company is a mecanum car with high-resistance wheels and it is directed to research. The kit includes wheels, platform, motors and needed accessories for them, screws and an optional kit for a Bluetooth application. The robot is not sold assembled but the creators offer the manual corresponding to these instructions.

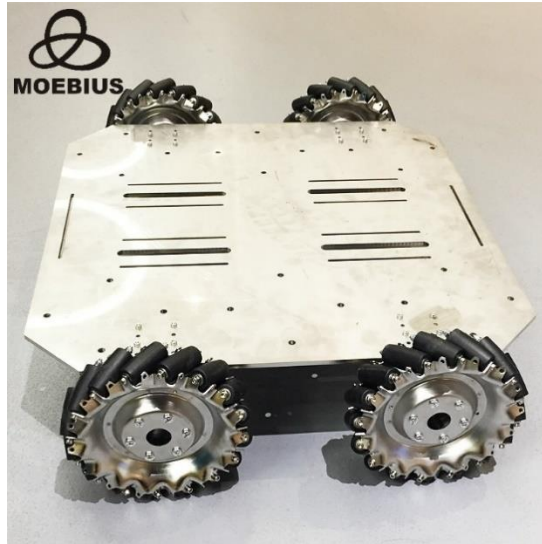


Figure 17. Second Moebius proposed solution for final prototype [20]

The information about this prototype was also retrieved from Aliexpress and its price is around 700€. It does not include the shock-absorbing feature, but, as mentioned previously, it is not something required for this application.

- HansaRobot Store

This robotics company also offers a shock-absorbing platform that has a maximum load of 40kg, it is designed for Arduino, STM32 and Raspberry Pi. The full kit includes the platform, wheels, motors, screws, shock-absorbing kit, and a PS2 controller. As in the previous seen platforms, it does not have a battery. This robot is very similar to Moebius shock-absorbing platform, and, as mentioned, this feature is not important in this project. This platform can be obtained on Aliexpress for between 590 and 640€ approximately.

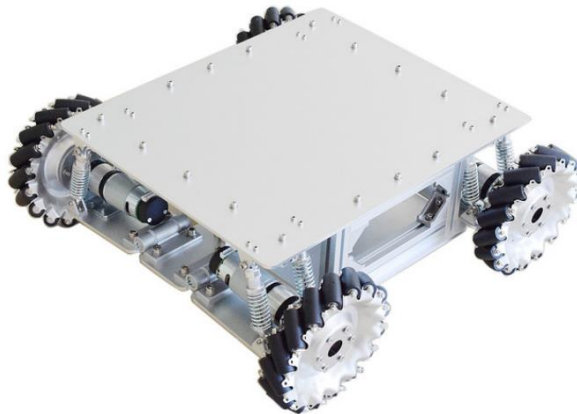


Figure 18. HansaRobot proposed solution for final prototype [20]

- Magical Tech Store

This a more advanced prototype that uses ROS and Raspberry Pi 4, it has an Ubuntu OS already and has different functions such as navigation, obstacle avoidance, etc. It is the most expensive platform (more than 1700€) due to its capabilities and already developed code. This was not proposed as a solution but as an example to look up to.



Figure 19. Magical Tech Store advanced prototype [20].

- Personalized robot

This option allows a very important advantage which is that its size can be adapted to the needs of the application, and to the characteristics of Hospital Clínic, where it is going to be tested. It can be fully personalized, including wheels, motors, platform and even supports for the sensors could be built and attached to the platform however it is desired. Different software should be used to design the platform and a 3D printer could be very useful to print personalized supports.

4.1.3. Robot wheels

From the start of the project, the robot was intended to be a holonomic robot. These robots use mecanum wheels or omnidirectional wheels, which allow the movement of the robot sideways without having to turn to the direction it is going to, in this case, it was decided that the robot would have mecanum wheels. These wheels have rollers inclined 45° on their circumferences that move freely while the wheel only moves forward or backward, and combining these two movements, the robot can slide sideways. See *Detail Engineering* section for more accurate information on the robot and wheels movement.

An online robotics shop was used to find the most appropriate wheels for this project, mainly considering the diameter and maximum load and the price. All mecanum wheels from <https://www.robotshop.com/> were considered, even though some were initially eliminated because they were made of plastic and directed to toy applications. The following table sums up the information retrieved from RobotShop on some of the characteristics of wheels, it must be taken into account that the web was visited in Spanish version to have accurate information on the price. Also, all of the listed options are sets of 4 wheels, two left and two right wheels.

OPTION	DIAMETER	WIDTH (mm)	BRAND	PRICE (€, taxes included)	MAXIMUM LOAD
1	254	95	Nexus Robot	614,21	>150kg/set
2	203,2	88,9	AndyMark	450,18	500 pounds/wheel=227kg/wheel
3	203	78	Nexus Robot	438,58	>150kg/set
4	203	78	Nexus Robot	531,6	>150kg/set
5	152,4	55,52	Nexus Robot	119,98	15kg/wheel

6	152,4	55,52	Nexus Robot	194,74	15kg/wheel
7	127	51	Nexus Robot	152,28	15kg/wheel
8	127	51	Nexus Robot	122,75	45kg/set
9	101,6	55,4	AndyMark	244,72	200 pounds/wheel=91kg/wheel
10	100	50	Nexus Robot	73,83	45kg/set
11	100	50	Nexus Robot	123,67	15kg/wheel
12	60	31	Nexus Robot	71,84	10kg*
13	60	31	Nexus Robot	91,37	10 kg*
14	54	34	Fingertech Robotics	72,75	0,5-13kg/set
15	48	25,5	Nexus Robot	34,8	3 kg*

*It is not specified if the maximum load corresponds to the whole set or one wheel.

Table 1. Options of mecanum wheels and their main characteristics [22]

Notice that the options are ordered from largest to smallest diameter. After the previous table was built, wheels of diameters larger than 200mm and smaller than 100mm were dismissed, and the choice was made between options 5, 6, 7, 8, 9.

4.1.4. Software

There are several software options to develop the code for robot movement obstacle avoidance and other aspects such as LIDAR plugins.

- ROS
ROS stands for Robot Operating System, and it is a framework that helps robotic software developers create a code that can control their robotic systems using nodes, topics, messages. Nodes are scripts that can be written in python or C++, they communicate passing messages to topics that other nodes read. Each node has one specific function, which, for example, can be reading information about the distance to obstacles or moving motors, among others. The best feature of this software is that it is open-source code, which means that the nodes, plugins and others are shared between developers, making it more accessible because some features are already done. Creators can work on previously developed code instead of having to do it from scratch. However, it only works on Linux, which is less popular than Windows. That is why extra software is needed to install ubuntu on a Windows computer, using for example, VirtualBox is an easy option, widely used, and instructions are available online. The most significant drawback is that the computer needs enough memory (RAM), otherwise turning on the Linux virtual machine and working on it can become a prolonged process.
- Microsoft Robotics Developer Studio
Microsoft designed this platform, and it is an Integrated Development Environment, which means it is a software that provides the main facilities for programming such as a source code editor, build automation tools and a debugger. This software is based on CCR (Concurrency Coordination Runtime) which is used to manage parallel tasks [23]. Both ROS and MRDS are SOAs (Service Oriented Architectures) but there are some differences between this software and ROS, the first one is that ROS works with Linux and can work with programming languages such as Python, C++ and Lisp, while MRDS runs in Windows and uses .NET, a software framework. Another difference is that ROS is open source while

MRDS is a closed source, this means that ROS codes are available on the Internet and can be modified while the ones developed in MRDS are private [24].

- Mobile Robot Programming Toolkit

This software is open source and works in both Linux and Windows with C++ programming language. It is very related to SLAM (Simultaneous Localization and Mapping), which was studied in the work done before this one together with ROS. It also includes algorithms for computer vision and motion planning (obstacle avoidance). This software is distributed under the New BSD License [25]. MRPT includes different C++ libraries, so users only depend on one part of MRPT.

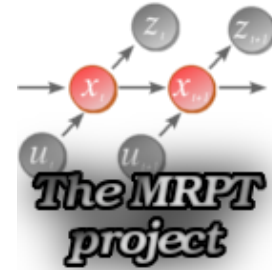


Figure 20. MRPT Project logo [25].

- CARMEN: Carnegie Mellon Robot Navigation Toolkit

This is also an open-source software for mobile robots, it includes base and sensor control, logging, obstacle avoidance, localization, path planning and mapping. CARMEN uses IPC, an inter-process communication platform that is used for message passing and synchronization. CARMEN works in Linux and is written in C, but it provides Java support [26].

4.2. Proposed solution

The final proposed solution includes all the sections described above:

- For the application, it was decided that the application would be air quality detection and disinfection. This is easier to implement and faster to obtain a prototype. Also, a student from EUSS School of Engineering was developing an air quality sensor, which was an excellent opportunity to unite the projects. Also, it is a very good opportunity to develop a robot that can help with the COVID-19 pandemic, making it a current topic project.
- Regarding the platform, the best option was considered to be the first one described. It includes almost all the needed accessories and components and has a good price and size. The assembly manual of the robot is also provided by the developers. Regarding the platform for the final prototype, it was united to the final degree project of another student that worked on the design and materials that could be used on the prototype. This had many advantages because wheels, supports and other components could be adapted to the needs of the project.
- The wheels that were bought were the eighth option, which had a 127mm diameter and 51mm width, with a cost of 122€. These were very reasonable in size, made of metal and had good reviews on Robotshop. Each wheel had 12 rollers inclined 45° to the plane of the wheel in a plane parallel to the axis of rotation of the wheel, and they made of polyurethane. The two sides of the wheels were made of an aluminium alloy. Each wheel weighted 500g, and all the set supported a load of 45kg. The four-wheel kit also included the screws needed to fix the wheels.

Nexus Robot, the manufacturer of this component, offers a datasheet describing the functioning of the kit and how the wheels should move to go in each direction.



Figure 21. Different perspectives of the final solution for mecanum wheels. 15cm ruler for reference.

- The software that was finally used was ROS, it was the best option due to previous knowledge and work using this software, as well as python. Another important feature, as mentioned above, is that it is an open-source code, and it is widely used, meaning that code can be found already developed online, and also errors that can come up have probably been resolved in online forums.

5. Detail Engineering

This section describes how the solution was implemented: software, code description, connection between the different items that took part in this project.

5.1. Organization of the project

Weekly online meetings were scheduled in order to have regular feedback from the different parts of the project, which ease the process of sharing tasks.

The most important item regarding organization was a GitHub repository where the documentation and code of the project were stored. Also, it contained instructions on how to perform the installation of software.

5.2. Flux diagram

This section, shows the general flux diagram of activities that were followed during the project.

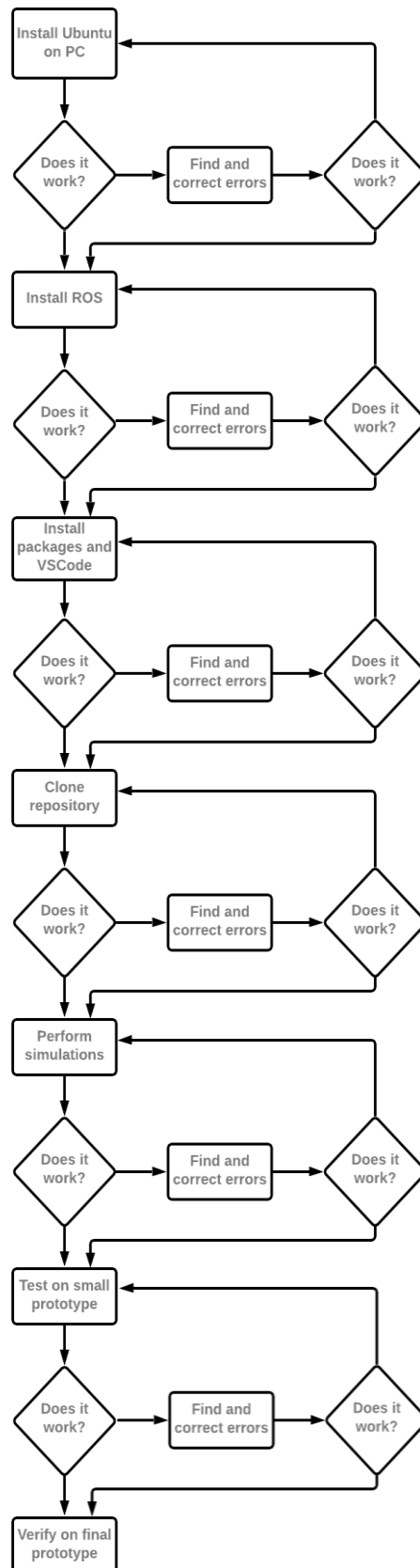


Figure 22. General flux diagram of activities

5.3. Software

This section provides an accurate description of the software used.

5.3.1. ROS: Robot Operating System

After the overview of ROS in the *Introduction* section, its functioning must be understood. It is not an operating system but a framework on top of an OS. First of all, ROS consists of code and tools organized in folders.

The main concepts are the following:

- Nodes:

These are processes, each node performs one task. The whole functioning of a project can be divided in several tasks, each of them controlled by one node.

- Master:

It registers nodes and allows the communication between them. It is the centre of the network, if it is not started, nodes cannot communicate. It tracks nodes to the topics.

- Messages:

Information passed between nodes. There are different message types, each of which has a specific data structure filled by nodes.

- Topics:

Where messages are published so that other nodes can obtain the information.

Nodes can be either publishers or subscribers. In order to pass the information, a node publishes a message to a topic and another node can subscribe to that topic to receive the message, as shown in the scheme below (Figure 23). It is very important that sent and received messages are the same type so that nodes can communicate. Topic type is defined by message type.

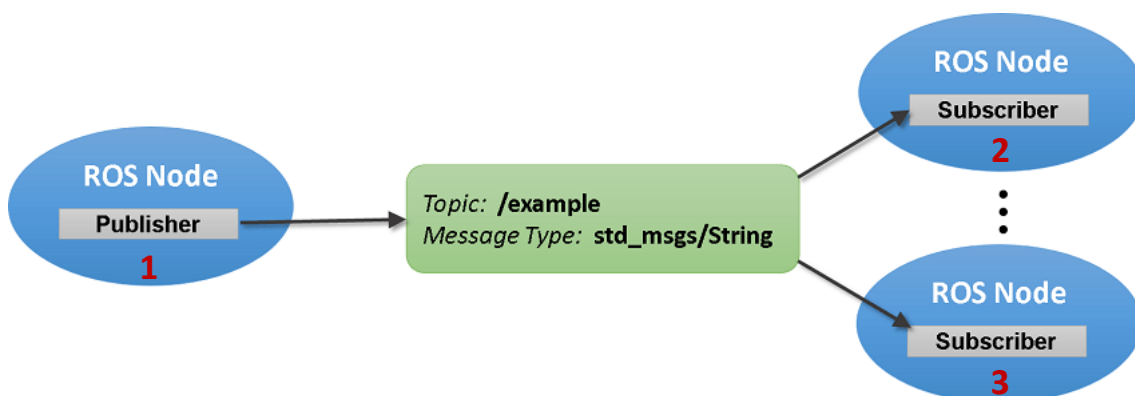


Figure 23. Nodes, messages and topics scheme [27]

Figure 23 shows a scheme where ROS Node number 1 publishes a message of type *std_msgs/String* to */example* topic. Nodes number 2 and 3 are subscribed to */example* topic and therefore receive the message that the first node publishes.

Regarding the code, a node is an executable file inside a package which is a directory, a software organization unit, inside them, there might be nodes, ROS-independent libraries, datasets, etc. A package is the smallest thing that can be built in ROS, and there are some which are already built in ROS, but developers can create their own.

The structure of *mecanum_control* package is shown in Figure 24. There are 4 folders inside the package, *launch* contains launch files while *src* is the code source, where there are python files, which are the nodes of the package. Also, there is a *package.xml* file, as in all ROS packages.

To execute the nodes, launch files and perform other ROS actions, different commands are used. There are also commands that allow listing information, such as running nodes or currently used topics. The three following commands are ROS basics to start executing code.

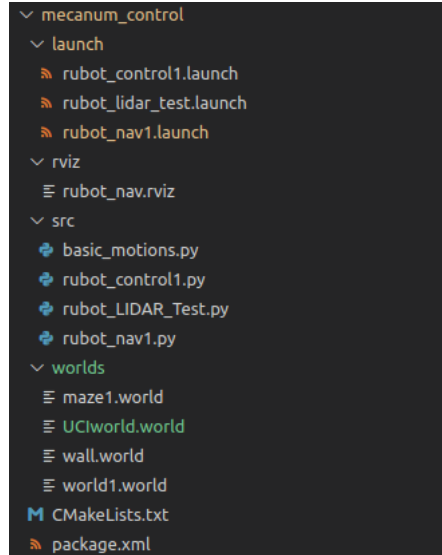


Figure 24. Structure of package *mecanum_control*

```
roscore
```

roscore executes a collection of nodes and programs under the name of *roscore*. This command starts the ROS Master so that nodes are registered and can communicate with each other. It also lists computer information. Roscore starts the master, parameter server and *rosout* node.

```
roslaunch
```

Allows executing a specific node using the name of the package containing it, there is no need to *cd*. To do so, the command is used the following way: `roslaunch package_name executable`.

```
roslaunch
```

Executes launch files, where the execution of nodes is defined. There is no need to have *roscore* running, as *roslaunch* starts the master if it has not been started yet.

Other commands such as `rostopic list` give information about the current used nodes. Topics and messages information can also be obtained by using `rostopic [subcommand]` or `rosmmsg [subcommand]`. To have a view of everything that is being executed, there is a very interesting ROS framework called *rqt*, which has many tools, one of which is `rqt_graph` that pops up a graph showing the connection between nodes, topics and messages when executed.

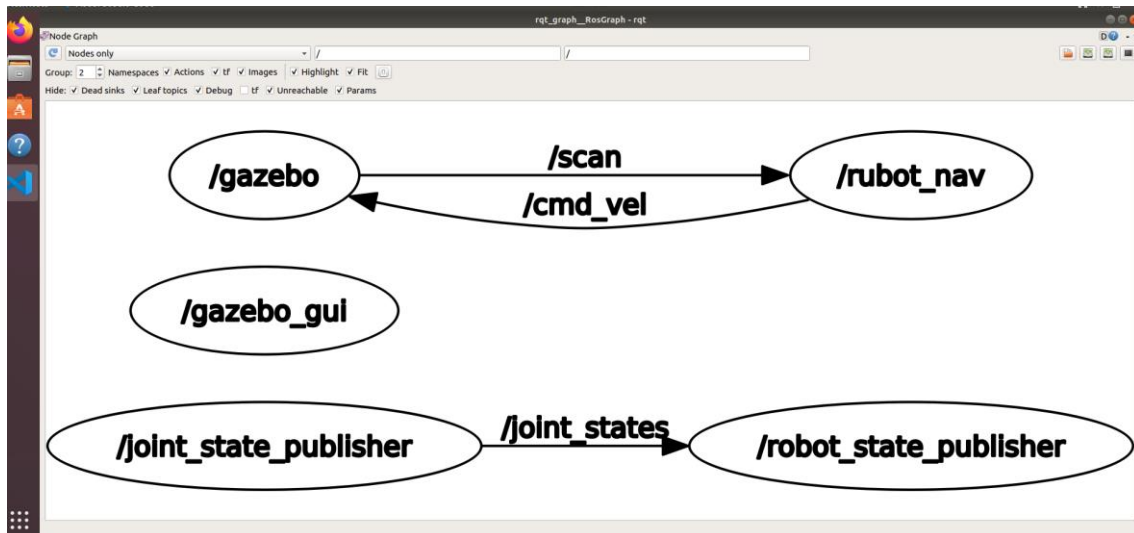


Figure 25. rqt graph that related nodes and topics

Most of the needed packages were already installed with ROS, however, some of them had to be specifically installed using `sudo apt install` commands on a terminal:

- `teleop_tools` or `teleop_twist_keyboard`: These are used to move the robot using the computer keyboard. The first one is used for non-holonomic robots and the second one is used for holonomic robots, which is the case of this project.
- `teleop_twist_joy`: This package is used for controlling the robot with a joystick connected to the computer instead of the keyboard, which can be more comfortable depending on personal preferences.
- `navigation`: This package takes information on odometry and sensors and computes the velocity commands.
- `slam_gmapping`: It is used to create a map taking information from, in this case, the LIDAR sensor and odometry.

To test the code on the small prototype, there is an extra component needed, which is Arduino Mega 2560 board. In order for ROS to reach Arduino, `rosserial` package was installed, which allows the communication over serial. This is only needed when working with a real-life robot, while simulations only need ROS packages because they do not use Arduino hardware components. After installing `rosserial` Arduino, some commands need to be executed in order to generate libraries when running the program for the first time. When running commands on Arduino through Ubuntu installed on VirtualBox, every time the Arduino board is connected through USB port it is needed to give access and permissions to the USB port by using `chmod` commands.

5.3.2. Arduino IDE

Arduino IDE is the software needed to write the Arduino scripts that will later be uploaded to the Arduino Mega2560 board. It should be easy to install by following instructions available online, however, some errors were found during the process that make it mandatory to mention them for future work when installing this software. This software needs to be installed in Raspberry Pi 4, which will be connected to the Arduino board.

First, the latest Arduino IDE version was installed (2.0 version), however, it led to some problems when installing libraries. It was realized that previous versions worked better so Arduino 1.8.15 Linux ARM 64 bits was downloaded. After downloading the zip file, it was relocated and unzipped

in the *Tools* folder (*~/Tools/Arduino-1.8.15*). It should be mentioned that a mistake was made when downloading this version without having previously uninstalled the 2.0 version, which later led to library problems and having to uninstall all versions to download the correct version. Once it is located in the correct folder, execute, from the directory:

```
./install.sh  
cd ~  
gedit .bashrc
```

Once the *.bashrc* file opens, add this line at the end: *export PATH=\$PATH:\$HOME/Tools/Arduino-1.8.13*. Save and close the file. Install *rosserial* Arduino using *sudo apt-get install* command. The next step is significant; you need to remove *ros_lib* from the *Arduino/libraries* directory. Next, execute the following command:

```
roslaunch rosserial_arduino make_libraries.py .
```

Arduino IDE and *rosserial* are now installed, you can test it using examples from *ros_lib* library.

5.4. Hardware

5.4.1. Raspberry Pi 4

Raspberry Pi is a computer with the size of a credit card mainly used for teaching and learning programming purposes, explicitly using Python language. It was created in the United Kingdom to promote science in schools and developing countries. One of the reasons this device is widely used is that it has a low price affordable for most of the population, which makes it a very suitable option for learning purposes. There are various models of Raspberry Pi, such as 3, 3B and 4. In this project, the used one was Raspberry Pi 4 Model B, which can be used with a screen connected through micro-HDMI port, a keyboard and a mouse that are connected through a USB port and a power supply that is be connected through a USB-c power connector (this should be connected last) [19]. Raspberry Pi 4 has 4 different USB ports, two of them are 2.0 and two of them are 3.0, which offers a higher transfer rates, one of these is used for the LIDAR sensor. There is a also an SD card slot, the inserted card usually contains the information on the OS.

One important characteristic of the Raspberry Pi computer is that it allows connecting to it remotely using free software available online. In this project, NoMachine was downloaded in both Raspberry Pi 4 and PC to use the computer as a remote desktop for Raspberry, but there are many other free options, such as connecting through *ssh* from a virtual Ubuntu machine.

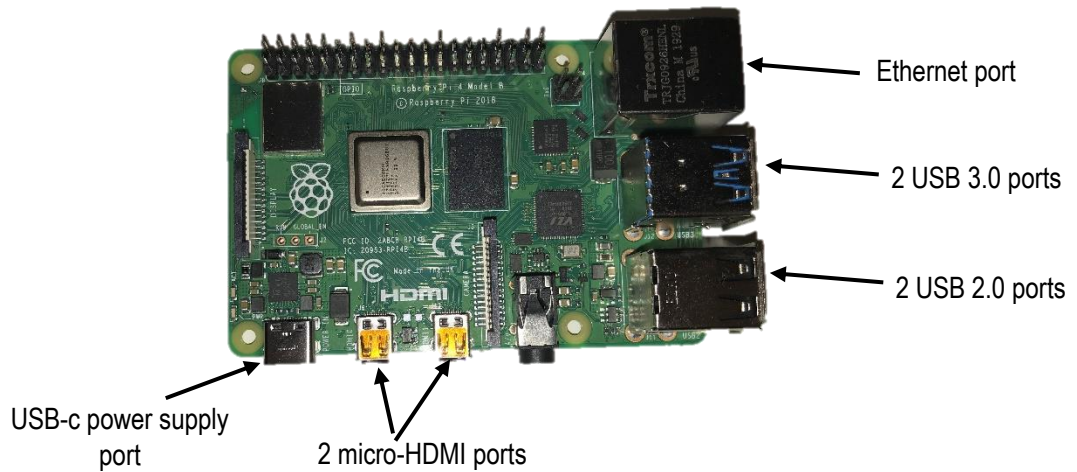


Figure 26. Front part of Raspberry Pi 4 Model B



Figure 27. Back part of Raspberry Pi 4 Model B

This tiny computer offers the possibility of installing ubuntu, even though it has its own OS, where also ROS can be installed. However, ubuntu is a more robust OS and it has more features than Raspbian, which is the default OS system installed on the SD RB card. To install the desired OS, the image of Ubuntu 18.04 is downloaded on a computer and it is then transferred into the SD card (inserted in the computer) using BalenaEtcher, which is a free software available online. Then, the SD card is inserted in the Raspberry Pi 4 and Ubuntu desktop is installed using a mouse, screen and keyboard. Afterwards, ROS and all the required packages for the project are installed on the Ubuntu OS.

Raspberry Pi is fixed onto the robot's platform, and it is connected to Arduino, LIDAR and a 5V battery. The connection of components is explained in the *Assembly* subsection.

5.4.2. Arduino

Arduino Mega 2560 is a microcontroller board based on Atmega16 microcontroller, and it is programmed using Arduino IDE Software. The basic advantage of this board against previously developed Arduino boards is the increase in memory to store the code and a higher number of I/O pins, concretely 54 digital and 15 analog pins. It is connected to a battery through a DC power jack. It is designed for more complex projects than previously developed Arduino boards [arduino].



Figure 28. Arduino 2560 board

There are shields, which are boards that can be plugged on top of the Arduino Mega 2560 board to extend its capabilities. In this project a Moebius shield was used to connect different features that cannot be plugged directly into the Arduino board. For instance, the four aligned white pin lines were used to plug in the motors.



Figure 29. Moebius shield

The wiring diagram of the shield is the following:

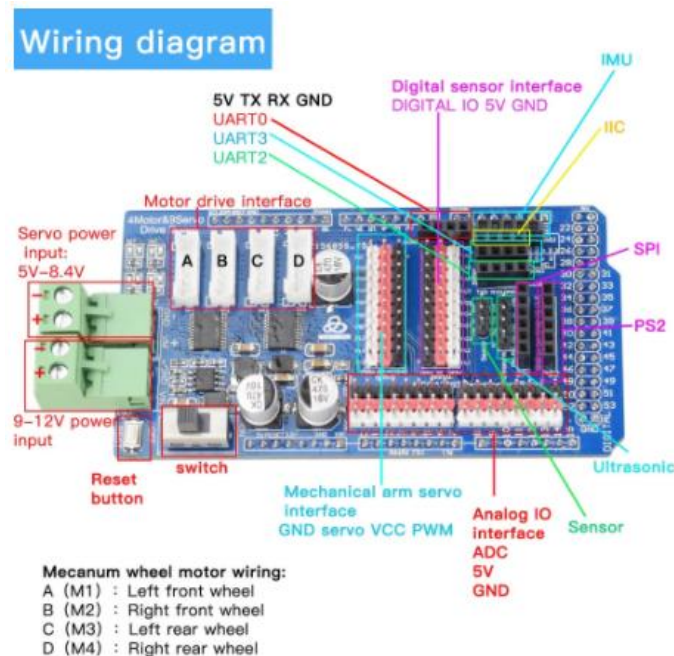


Figure 30. Wiring diagram of Moebius shield for Arduino Mega2560

5.4.3. LIDAR

LIDAR stands for *Light Detection and Ranging*, and it is used as a sensor to detect obstacles. It is continuously spinning and records the distance to obstacle every 0.5° , that is 720 recordings for every spin. The model used in this case was developed by 3irobotix, concretely model delta 2A, whose specifications are shown in the table in Figure 32.

Range	0.13m-8m (White Wall)	Sampling Rate	5K/s
scanning frequency	6.2Hz	Laser wavelength	780nm
Laser power	3mW (The most)	Accuracy	<1%@5m
Distance Variance Coefficient	DCV<0.2%	Communication Interface	RS232 (TTL)
Voltage	5V	Power consumption	1.5W
Baud rate	230400	Working current	500mA
weight	175g \pm 2g	volume	107*76*53mm
Working temperature	0°C-45°C	Level	0°- 1°
Working environment humidity	<90%	Working environment illumination	<1000lux

Figure 32. 3i LIDAR Delta 2A specifications

It is connected to the Raspberry Pi through an USB 3.0 port, plugged into the signal connector on its PCB (seen in front of the LIDAR in Figure 31), and to a battery plugged into the motor connector that makes the LIDAR spin. The code receives information about distance to object from the LIDAR sensor and gives instructions to the wheels to turn to the opposite direction where there is an obstacle if it is too close. The LIDAR sensor is fixed on top of the robot or in its frontal part so that the scanner does not find any obstacles.



Figure 31. LIDAR fixed on small prototype.

5.4.4. Holonomic robot

Holonomic robots are those that can slide directly sideways by using a concrete type of wheels and combining forward and backwards movements in each of them. Each wheel moves with a different motor so that they can move in different directions at the same time, making the characteristic sideways movements of these type of robots.

Holonomic robots can employ either mecanum wheels or omnidirectional wheels. Omni wheels have rollers along its circumference, in the direction of the perimeter of the wheel and can be used in three-wheeled and four-wheeled robots. However, it is essential that they are placed in the correct angle, they are not placed as common wheels (such as car wheels) but with an angle. Mecanum wheels also have rollers on their circumference, but they have an axis of rotation of 45° to the wheel plane, and they are placed the same way as common wheels, not rotated. Having them roll in different directions makes the robot turn, slide and move however it is needed, the wheels do not turn as car wheels, they only go forward or backwards.



Figure 33. Omnidirectional wheels (Wikipedia, Omni wheel) (left) and mecanum wheels (right)

Mecanum wheels were the choice for this project because they are easier to implement and more efficient, while omnidirectional wheels are more difficult to drive.

The wheels have to be placed explicitly, as seen in Figure 34, if the rollers are facing the wrong direction, then the robot will go to the wrong direction when driving it.

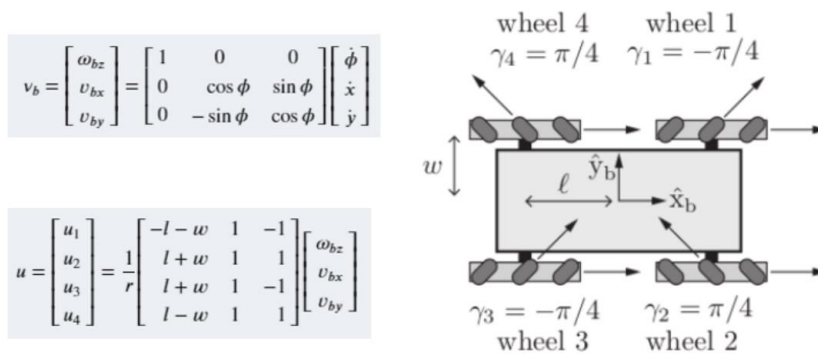


Figure 34. Kinematic model of the mobile robot with four mecanum wheels [28].

Where:

- u is the vector containing each wheel driving speed,
- r is the radius of the wheels,
- l and w are the dimensions of the chassis,
- γ (in the figure) is the angle at which free "sliding" occurs

The direction of the wheels for each movement is the following:

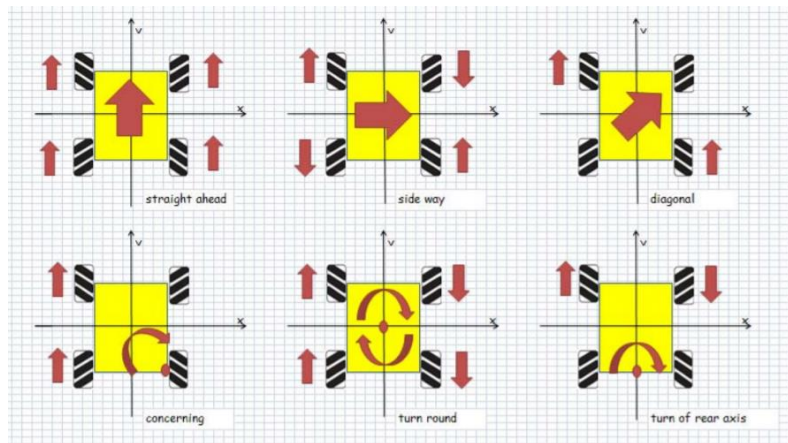


Figure 35. Direction of the wheels for each movement (Nexus Robot. Mecanum wheels (Ilon Wheel))

5.4.5. Assembly

- Small testing device

This robot was provided on AliExpress by MoebiusTech Store. There are available online resources provided by them that detail how to fix components on the different platforms and which connections should be done for hardware elements to communicate. All the information is available on their GitHub repository: https://github.com/MoebiusTech/ROS_STM32.

However, the instructions are very short and some of them need an extra description. The first steps, regarding fixing the motors and wheels on the black metallic platform were easy to follow, having to pay close attention to the direction of the rollers on the wheels. Regarding Raspberry Pi, Arduino, LIDAR and the connection between components it was different in some ways.

First of all, it is needed to know that this project used a Raspberry Pi 4 Model B, because of this, the prototype required two batteries: one for Arduino board and the other one for the Raspberry computer. Raspberry Pi 3 and 3B models can be powered by the Arduino board. Therefore, the components that needed to be fixed on the platforms were two batteries, LIDAR sensor on its support, Arduino board and Raspberry Pi.

Arduino shield was placed on top of Arduino Mega2560 board, making the pins coincide. The motors were directly connected to the shield following the wiring instructions on the Arduino shield. Afterwards, Raspberry Pi 4 was fixed on the bottom platform next to the Arduino board and shield. Regarding the LIDAR sensor, it was fixed on the top platform, and having this the connections made were the following: LIDAR signal connector was plugged into the RB through a USB 3.0 port, Another USB RB port was used to connect the Arduino board (blue cable), and two batteries were used the grey one is a 5V battery and is the RB supply, and the blue is a 12V battery that was firstly connected to the Arduino 2560 board.

However, these connections were not correct and were later changed, LIDAR was missing the motor connector, which needed to be connected to a battery in order for the LIDAR to spin. Also, the battery that was plugged directly into the Arduino board through a jack connector, was later changed to two stripped wires plugged into the green connectors of the shield.

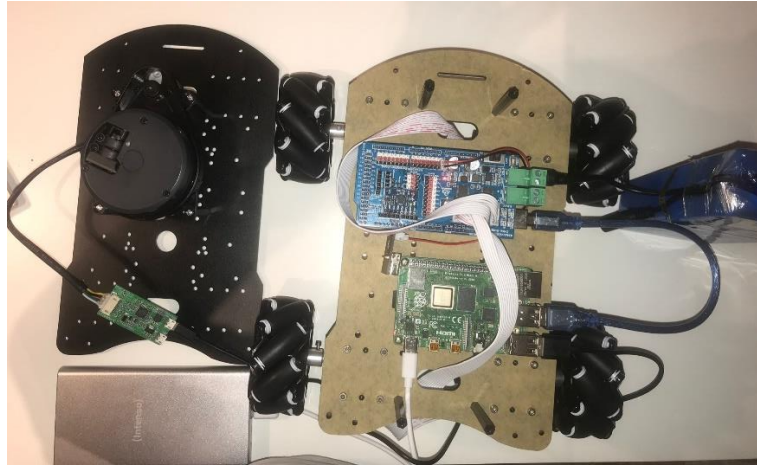


Figure 36. First version of the fixed components and their connections.

The blue battery was supposed to be placed behind the LIDAR sensor on the top platform, stacking the grey battery on top. However, it was seen that it was more practical to put the 12V battery where the Raspberry Pi computer was supposed to be placed, while putting the raspberry on the top platform below the LIDAR sensor, and the 5V battery would be placed where the 12V battery was initially supposed to be fixed. LIDAR, RB, Arduino board and shields and the two platforms were fixed using screws, while the rest of the batteries and PCB of the LIDAR sensor were fixed using double-sided tape.



Figure 37. Final assembled prototype for testing.

- Final prototype

The final prototype, as mentioned, was designed by another student from EUSS School of Engineering. Several proposals were made, but the final decided option consisted of a platform divided in four pieces with a cover to protect the electronic components of the robot. This option was later modified, and an extra platform was added on top of the cover for further applications that could be added to the robot, such as transporting medical supplies, samples and others.

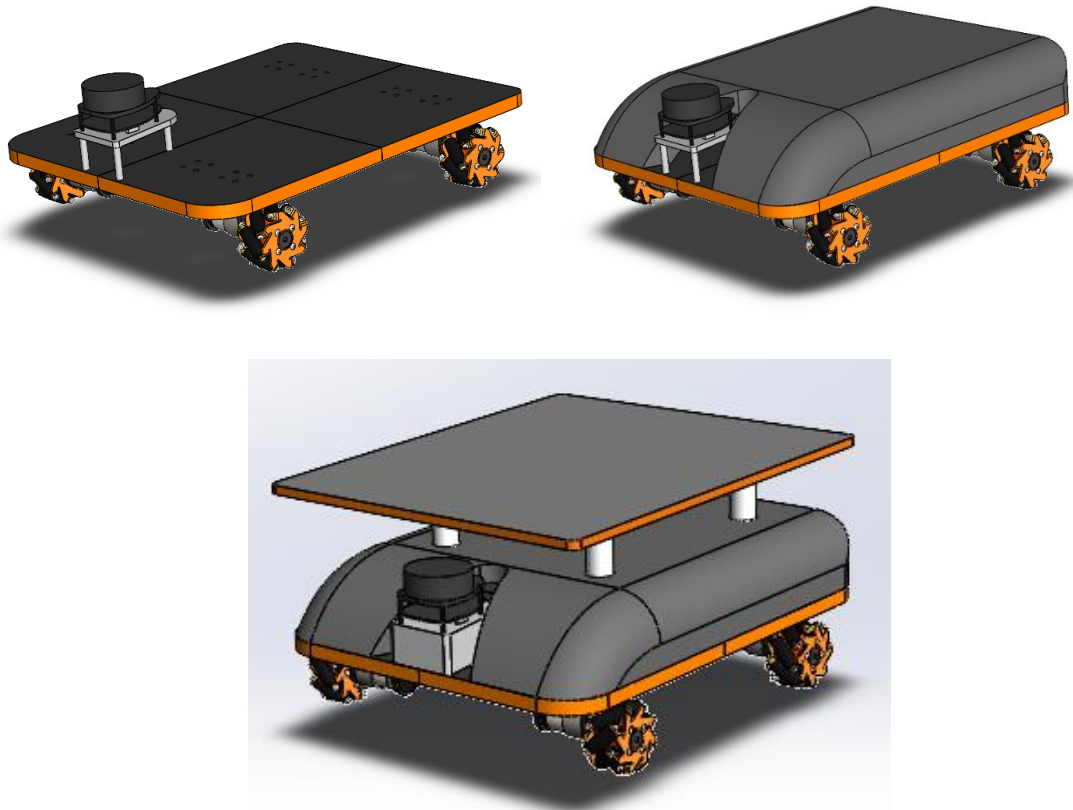


Figure 38. 1st proposed prototype without the cover (top left) with the cover (top right) and prototype with extra platform (bottom)

5.4.6. Sensors

The sensors are controlled by a Raspberry Pi 3 Model B, and there are two of them that will be fixed on the robot platform:

- Air quality sensor

It measures seven air parameters that will be used to determine if the air quality is good enough. The model of the sensor was SM300D2-V02 sensor, and the seven variables that it gives are CO₂ concentration, formaldehyde concentration, volatile organic compounds, 2.5µm diameter particles, 10µm particles, temperature and humidity.

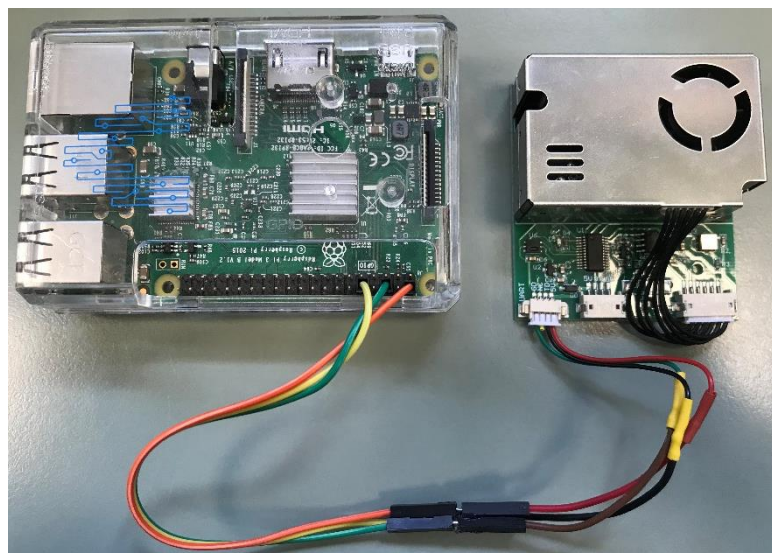


Figure 39. Air quality sensor connected to RB Pi 3

It outputs the data of the seven parameters every second. A script was developed in another final degree project by an EUSS School of Engineering student in order to obtain the data and upload it to different databases to finally visualize it and develop directions to start the ozone emission.

- Ozone sensor

This sensor is supposed to sense ozone concentration when this substance is being emitted in order to give an alert when the concentration is low enough for people to enter the room. The sensor used was Winsen O₃ sensor.

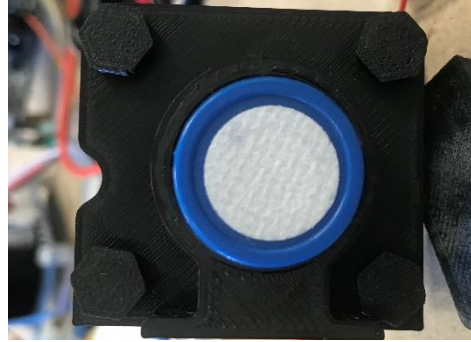


Figure 40. Ozone sensor

5.5. Code description

5.5.1. ROS nodes

As mentioned in previous sections, this project was developed using a GitHub repository [28] where all the information was stored. There are several folders inside the repository, the most important ones:

- Documentation

Contains *.ipynb* files with information on software installation, how to execute commands, holonomic robot, etc.

- src

It is the source of the code, it contains packages that store nodes, launch files, maps, rviz and gazebo files as well as robot model descriptions. It is the folder needed to execute all the commands. Inside this folder, there are several packages (Figure 41).

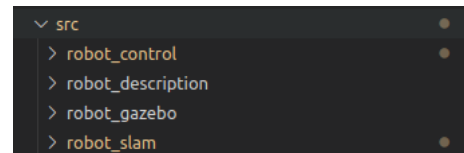


Figure 41. src folder.

- o *robot_control*: this package contains files to spawn the robot on Gazebo environments and test the LIDAR and autonomous navigation features.
- o *robot_description*: contains robot model descriptions (Nexus robot and final prototype robot) and launch files to visualize them in Gazebo environment. URDF files have plugins to execute commands to move the models using the keyboard.
- o *robot_gazebo*: It contains other packages with launch files to spawn different robots in Gazebo, multiple robots simultaneously, and modified models.
- o *robot_slam*: The files in this package allow scanning maps and saving them using SLAM and later creating trajectories to go from one point of the map to another one.

5.5.2. Arduino scripts

There is a main program and six extra files for libraries:

- Main script:

In the case of the final prototype, this file obtains the value of O₃ concentration from the Winsen sensor, and communicates with Raspberry Pi through rosserial to obtain odometry information.

However, in the case of the small prototype the code lines that referred to the ozone sensor were eliminated because it was not implemented in this testing device.

- config.h

The pins of the motors are defined in this script. Also the distance to the centre of the robot from the wheel rotating axis, distance from the centre of the robot to the width of the wheel and wheel radius are defined, and the maximum rpm allowed by the motors.

- encoder.h

Pins of Hall sensor of encoders are defined.

- imu.hpp

This is used in case an IMU is implemented, it initializes the accelerometer and gyroscope and shows the values of acceleration in m/s^2 and gyroscope in rad/s. An IMU is used to know the angular rate and orientation of a body, in this case, the robot.

- kinematics.hpp

Definition of functions to convert angular and linear velocities to PWM values, which define pulses to send to the wheels to achieve certain velocities. It computes both forward (velocities to PWM) and inverse (motor velocity to angular and linear velocities) kinematics.

- motor.h

It defines motor pins. A PWM value is assigned to define velocity.

- pid.hpp

It computes velocities to reduce and increase the robot velocity gradually.

5.6. Results verification

5.6.1. Simulation

The developed code was not intended to be tested directly on the final prototype, instead, virtual environments were used and later it was tested on a smaller prototype.

The virtual simulation required the described software above, as well as Gazebo and rviz. Gazebo is a tool for robot simulation that includes mechanical and physical features such as friction and gravity. rviz on the other hand is a visualization tool, it helps in robot visualization as well as obtaining information from sensors and cameras. Gazebo is used to simulate the robot moving and rviz is used to see the LIDAR information.

The first simulations were tested using a Nexus robot. Nexus Robot is a Chinese company that is specialized in mecanum and omnidirectional wheel robots, and one of their prototypes was virtually used in Gazebo and rviz. The first commands were simple and consisted in only seeing the robot in both environments with no errors reported, as in the following figure:

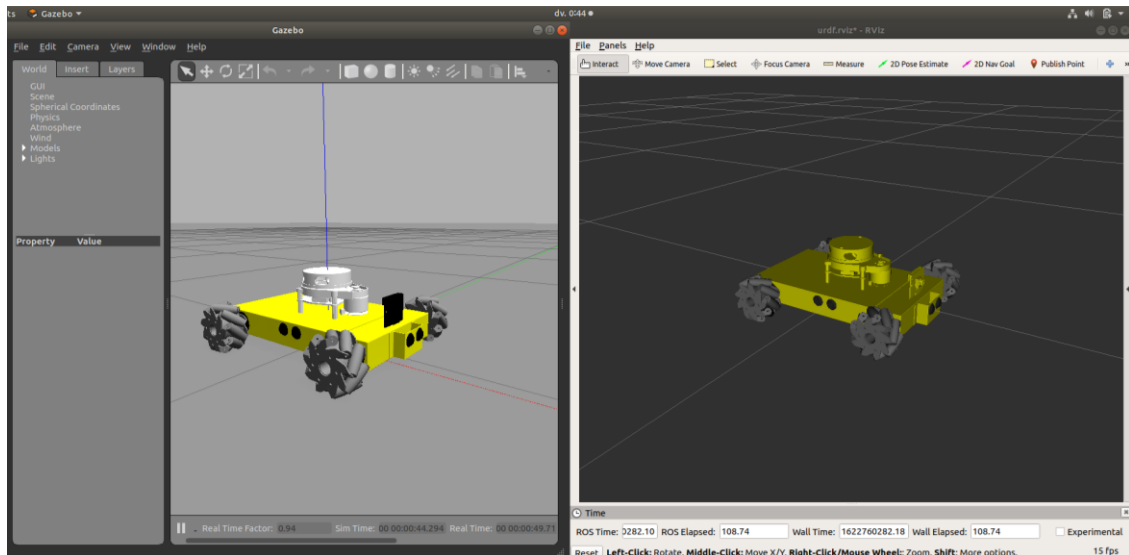


Figure 42. Nexus robot in Gazebo (right) and rviz (left) environments

Sometimes, rviz gives an error regarding the fixed frame, to solve it first click on Panels and make sure the box *Displays* is shown. In the Displays panel, which will be shown at the left part of rviz screen, go to General Options and use the drop-down to change the fixed frame, usually *base_link* works fine.

The robot in Figure 43 is a nexus model with a LIDAR and camera added. This cannot be seen or used on Gazebo but in rviz, which allows seeing the data acquired from sensors such as LIDAR and the visualization of the camera, which is in the front part of the robot (black in Gazebo and yellow in rviz). The difference in colours is because in Gazebo, colours are added inside the URDF file using already predefined colours from this application, while rviz works differently. However, they do show the robot simultaneously and simultaneously, which means that if a moving plugin is used in Gazebo, the robot will also move in rviz environment.

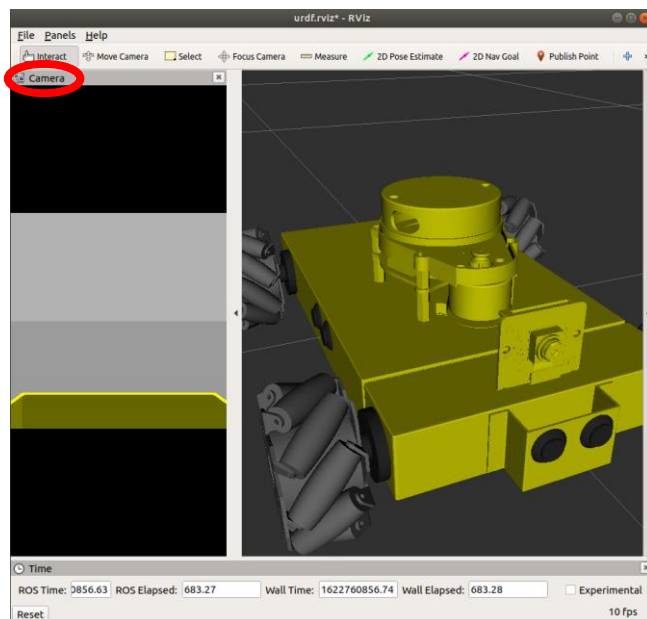


Figure 43. rviz camera vision and robot

Figure 43 shows the camera vision in rviz environment, which is one of the panels that can be presented using rviz, the view of the laser-scanned area can also be shown, as it will be seen.

After verifying the robot is visualized, the code testing begins. First of all, a Gazebo plugin needs to be added to the URDF file so that the robot can be driven around, in the case of the Nexus robot, there was a specific Nexus plugin was already built by the developers of this prototype, but it should be changed depending on the robot used. By typing the command:

```
roslaunch teleop_twist_keyboard teleop_twist_keyboard.py
```

The robot could be controlled using the computer keyboard. This was achieved rapidly after fixing some errors that appeared due to installation mistakes and source errors. Also, first the nodes used were directed to control non-holonomic robots, and it was later changed to `teleop_twist_keyboard.py` to simulate a holonomic robot.

```
Reading from the keyboard and Publishing to Twist!
-----
Moving around:
  u   i   o
  j   k   l
  m   ,   .

For Holonomic mode (strafing), hold down the shift key:
-----
  U   I   O
  J   K   L
  M   <   >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

When executing the previous command, the terminal shows the output seen in Figure 44, this output gives the instructions to move the robot according to its characteristics. As this project focuses on a holonomic robot, uppercase letters need to be used. `/` and `<` will result in forward and backwards movements, while `J` and `L` refer to sliding sideways movements. `K` is used to stop and `U`, `O`, `M` and `>` move the robot in diagonal movements.

When using lowercase characters, the robot behaves like a non-holonomic robot (similar to a car). The package that was previously used to simulate this behaviour was `key_teleop` instead of `teleop_twist_keyboard`.

Figure 44. Output of controlling robot through keyboard command.

The next step was to try the robot navigation, using the created package `robot_control` shown in Figure 41. This package contains three launch files that call three different `.py` files that have three different functionalities when they are executed: movement control, lidar test and autonomous navigation and obstacle avoidance.

In the first launch file, there are parameters that can be changed: x and y linear velocities, angular velocity in z direction and a maximum displacement of direction.

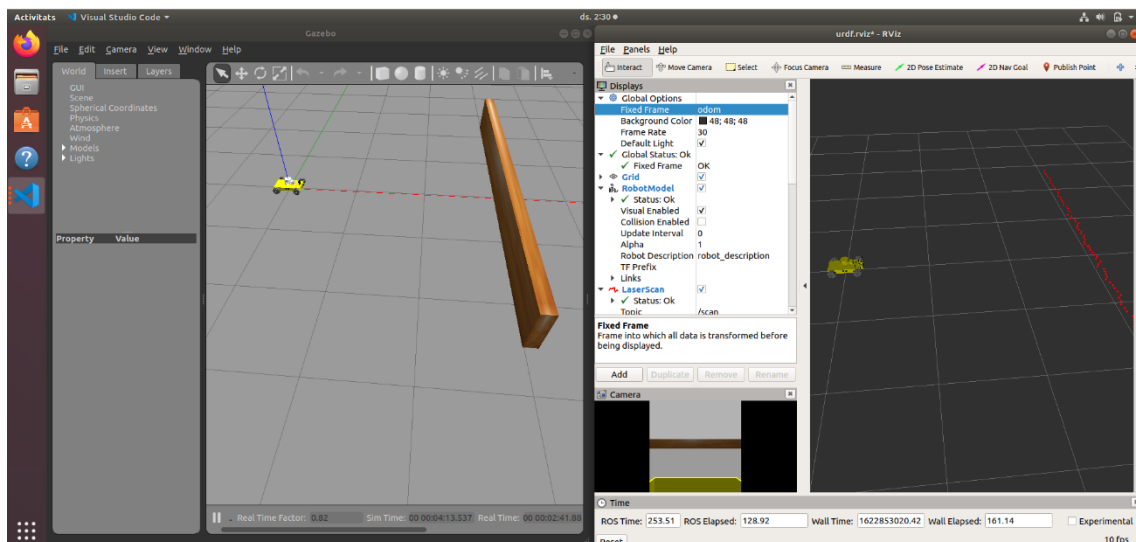


Figure 45. Output of the first robot_control launch file. Gazebo (left) and rviz (right) environments.

The Figure above (Figure 45) shows the result of the execution of the launch file that controls the robot movement, Gazebo shows the robot and the world where it is placed, and rviz shows the prototype and the information acquired from its sensors, in this case, the LIDAR sensor sends information about obstacle distance. Therefore the red dots appear as a result of the detected distance to the wood barrier that is seen in Gazebo.

The second launch file is designed and directed to test the LIDAR sensor. Inside the launch files, there is a parameter where the world can be specified, Figure 46 shows the UCIworld.world defined, which was used as an example of a clinical environment, simulating different rooms on an Intensive Care Unit (ICU). Blue dots refer to eight rooms, and the green one shows nursery control (from where patients' monitored data is controlled) and the red dot is a room used as a storage for medical supplies and others.

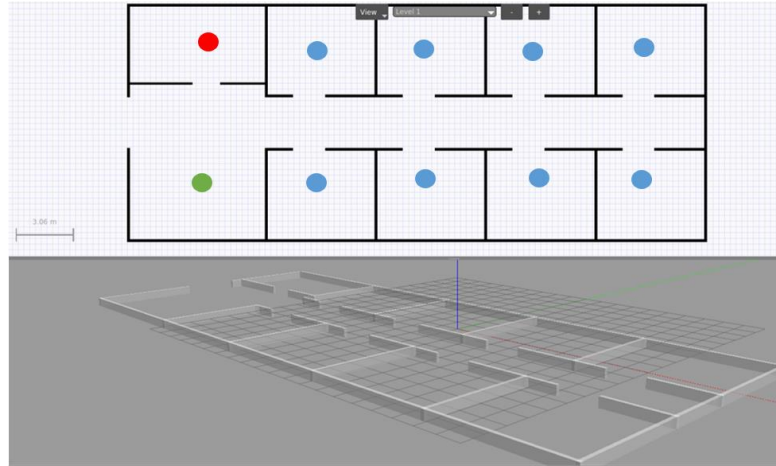


Figure 46. simulation of a clinical environment (ICU)

When testing the LIDAR sensor inside this world, rviz shows the walls detected by the sensor and the corridor and entrance to two of the rooms and their walls can be clearly distinguished. Rviz parameters can be changed to see bigger dots, change their shape and even their colours.

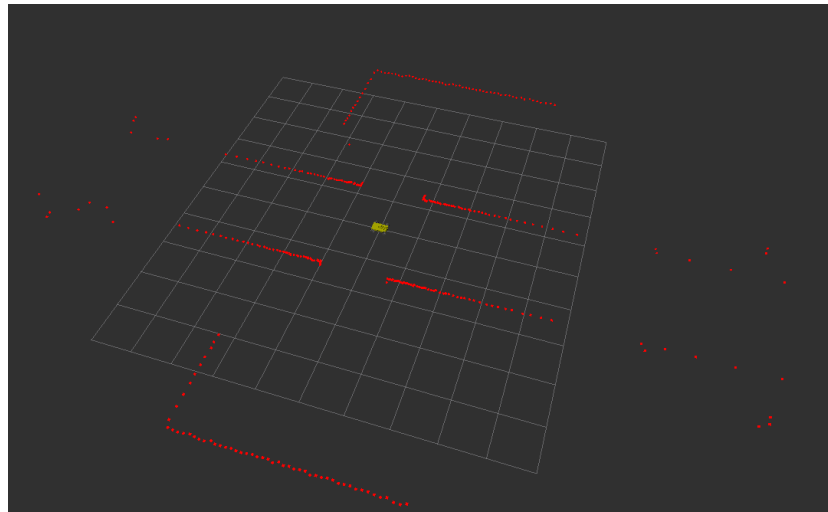


Figure 47. rviz environment showing LIDAR-detected obstacles

The third launch file is used for autonomous navigation, that is to say, to make the robot move autonomously, avoiding obstacles by defining a maximum distance to a block. If the LIDAR detects a distance to obstacle smaller than the defined one, then an order is sent to turn the robot into the opposite direction of the barrier.

After testing the control and robot navigation, the SLAM routines were tested, also simulating the virtual environments. First of all, *navigation* and *gmapping* packages were downloaded and *robot_slam* package was created. To test SLAM routines, first the map needs to be generated which is done following these steps:

- Launch the robot within the world that wants to be scanned.
- Launch SLAM mapping
- Execute a command that allows moving the robot and scan the whole map
- Save the scanned map

When launching mapping, a rviz window will pop up showing the already scanned data. The difference between this and the LIDAR test is that SLAM mapping shows the currently and previously scanned points, that is to say, SLAM saves the positions where the robot found an obstacle, instead of showing current scanned obstacles as in Figure 48. After launching SLAM mapping, when running the commands to move the robot, the environment will look like the following:

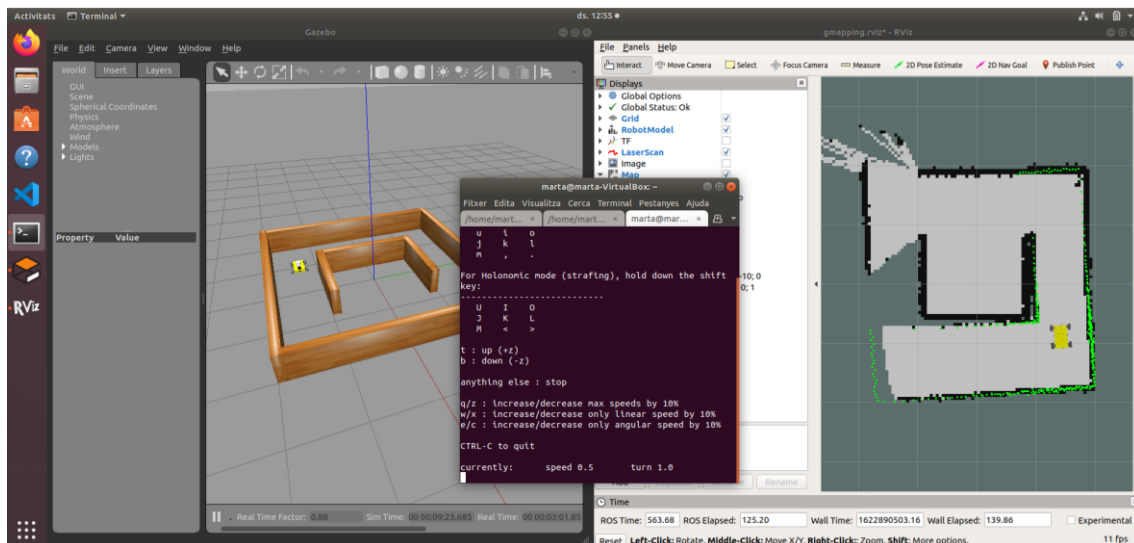


Figure 48. Robot in Gazebo (left) scanned points in rviz (right) and terminal where the command is executed to move the robot.

As it can be seen, rviz shows where the robot has been, it can also be noticed that not the whole map had been scanned when the screenshot was taken. When the entire map is scanned, then it is saved, and it can be used to plan trajectories and direct the robot to specific locations, as seen in Figure 49, which shows the scanned ICU map.

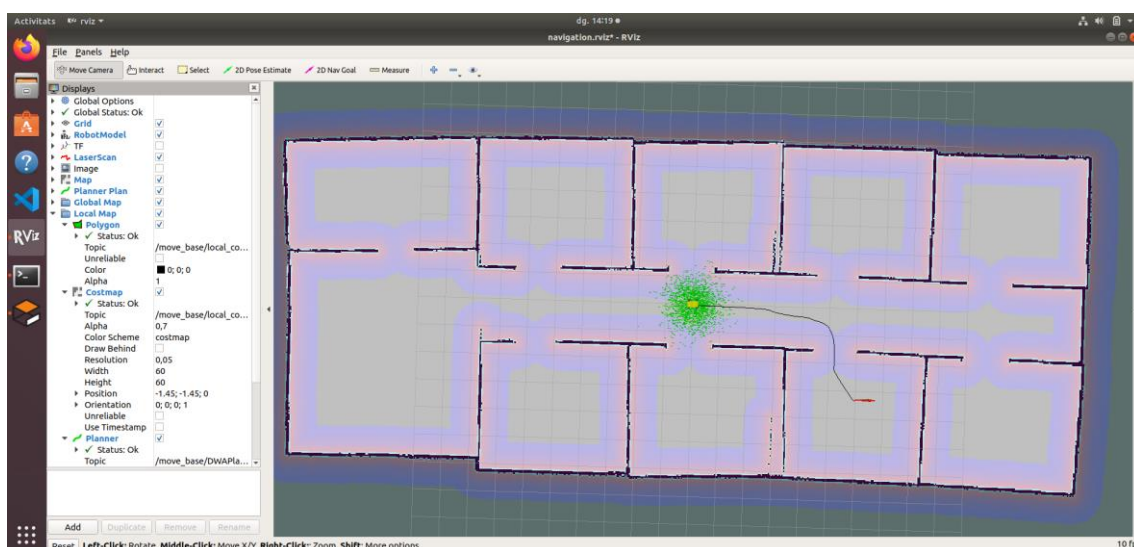


Figure 49. Trajectory drawn in ICU map generated using SLAM.

Another important task during the simulation process was creating an URDF file with the description of the final prototype to later use it as the model in the simulations. The prototype was defined using SolidWorks, and it was exported to a URDF file using a plugin that can be downloaded in ROS official webpage. Several errors were found regarding robot orientation when the robot was spawned in Gazebo environment, but they were solved changing the Global Origin Coordinate system. As the information on how to use the exporter was very limited, a tutorial was created in order to ease the process for future work in this project or others. This tutorial can be found at the end of this document in *Annex* section (Tutorial: How to export a SolidWorks Assembly into an URDF file. Gazebo visualization.). When using this exporter, a folder containing the URDF file, launch files and meshes is created. The robot can be shown in gazebo by launching the *gazebo.launch* file.

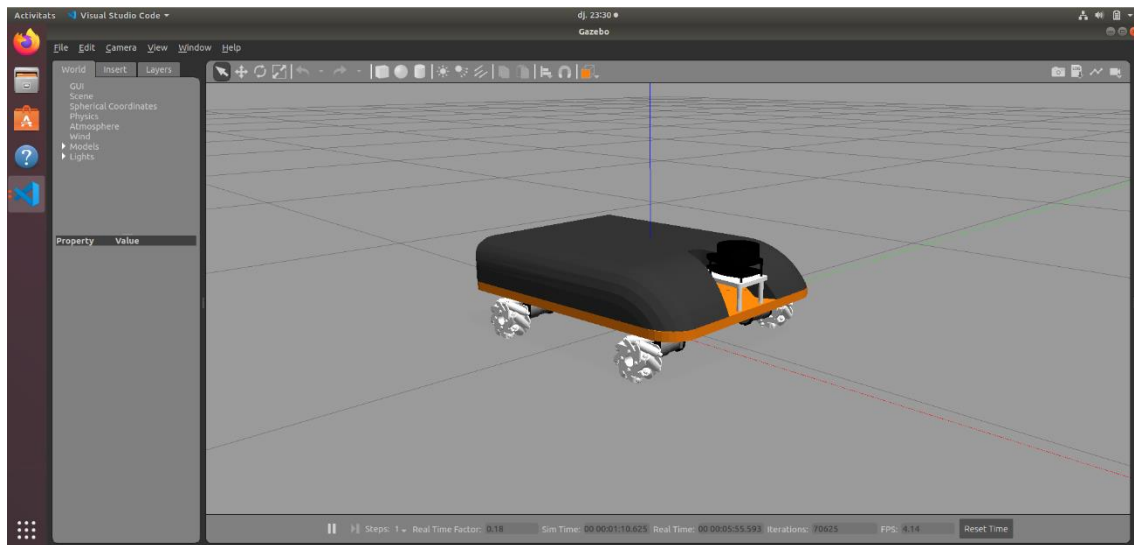


Figure 50. First achieved URDF model spawned in Gazebo

5.6.2. Small prototype

To test the code on the small prototype, first there were some features to verify. First of all, it must be known that the difference between testing the code on a simulation and testing on a real-life prototype is the Arduino board, which, as aforementioned, communicates with ROS through rosserial. The first thing that was tested, was that the Arduino board was communicating with Raspberry Pi by using an example script from *ros_lib* library. This script aims to receive through a terminal the phrase "hello world!", which is done by compiling and uploading the script to the Arduino Mega2560 board (it has to be specified on Arduino IDE, as well as the serial port it is connected to). Afterwards, roscore is run in a terminal, as well as a rosserial node (`roslaunch rosserial_python serial_node.py /dev/ttyUSB0`). When checking the `/chatter` topic it should output what is seen in Figure 51.

```

10:42
ubuntu@ubuntu: ~
File Edit View Search Terminal Tabs Help
ubuntu@ub... x roscore http... x ubuntu@ub... x ubuntu@ub... x
ubuntu@ubuntu:~$ rostopic echo /chatter
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---

```

Figure 51. Messages received by `/chatter` topic

Once the functioning of Arduino board was checked, the next test was to verify the wheels were receiving information from the Arduino board and shield. This was done by executing the following commands:

```
roscore
```

To start the ROS master and be able to use the `roslaunch` command.

```
roslaunch rosserial_python serial_node.py /dev/ttyUSB0
```

Keep in mind that it is needed to check the serial port and specify it, in this case it was USB0. This enables the communication through rosserial, as in the previous example. Afterwards a command was executed to publish a Twist message containing information on forward, lateral and angular velocities:

```
rostopic pub /cmd_vel geometry_msgs/Twist -r 10 -- '[0.5, 0.5, 0.5]' '[0.0, 0.0, 0.0]'
```

The two vectors define the linear velocities in x, y and z axis and angular velocities in x, y and z axis.

However, when executing this command, it was seen that the wheels were not moving. Wiring was checked and once verifying it was correct, it was deduced that there was a mistake when defining the pins in Arduino scripts. The previously used pins were retrieved from the official documentation, which was proved to be wrong. The correct pins were the following:

MOTOR	Encoder A	Encoder B	PWM	DIR1	DIR2
M1	18	31	12	34	35
M2	19	38	8	37	36
M3	3	49	9	43	42
M4	2	23	5	A4	A5

Table 2. Pinout connection of the motors

The two marked pins were the ones that were found to be wrong in the shield documentation. According to Figure 30, M1 corresponds to the left front wheel motor, M2 to right front wheel, M3 to left rear wheel and M4 to right rear wheel.

Once this was changed in the Arduino scripts and the three commands were executed again, the wheels did move, which meant they were now receiving the information from the Raspberry Pi through the Arduino board and shield.

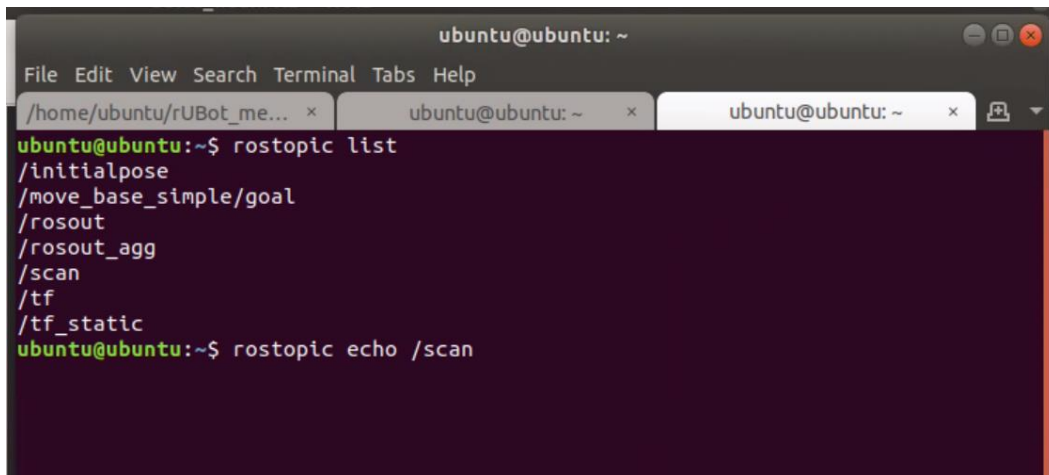
Having achieved this, the `teleop_twist_keyboard` package was used to move the robot. Executing the `.py` file, the same output as the one in Figure 44 was obtained, with the instructions to move the robot using the keyboard. It was verified that this package worked and the robot was obeying the orders sent from the keyboard.

Another verification that was being done at the same time was the LIDAR test. 3i robotics offers a LIDAR and also a package to start it and visualize in rviz the points scanned by the laser. When executing the corresponding launch file, rviz did not show any points scanned by the LIDAR. There was a warning regarding the fixed frame, but it was solved by running the following command:

```
roslaunch tf static_transform_publisher 0 0 0 0 0 0 1 map base_link 10
```

This allowed having a transform from one frame to another one and no warnings were shown while this was being executed in a parallel terminal. The LIDAR was spinning and connection was being established with no errors but the cloud point did still not show any points.

It was deduced that the LIDAR sensor was not sending data, therefore as the topic that registered the data was `/scan`, commands were run in order to look for what this topic was receiving. However, when looking at the topic list, it appeared when it came to analysing what the topic was receiving nothing appeared on the terminal (Figure 52).



```
ubuntu@ubuntu: ~  
File Edit View Search Terminal Tabs Help  
/home/ubuntu/rUBot_me... x ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x  
ubuntu@ubuntu:~$ rostopic list  
/initialpose  
/move_base_simple/goal  
/rosout  
/rosout_agg  
/scan  
/tf  
/tf_static  
ubuntu@ubuntu:~$ rostopic echo /scan
```

Figure 52. `/scan` analysed through commands

To ensure it was not due to having only-charge wires, another LIDAR was tested, concretely RPLIDAR A1M8, by Slamtec. This had a 12 meters range with 1 degree resolution, which means 360 points for every spin. This LIDAR was simpler due to the fact that there was only one wire to connect it, while the other one had two micro-USB ports, one for the signal and one for the motor. When testing this new device with its corresponding `rplidar` package, `rviz` showed the points that were being scanned (Figure 53).

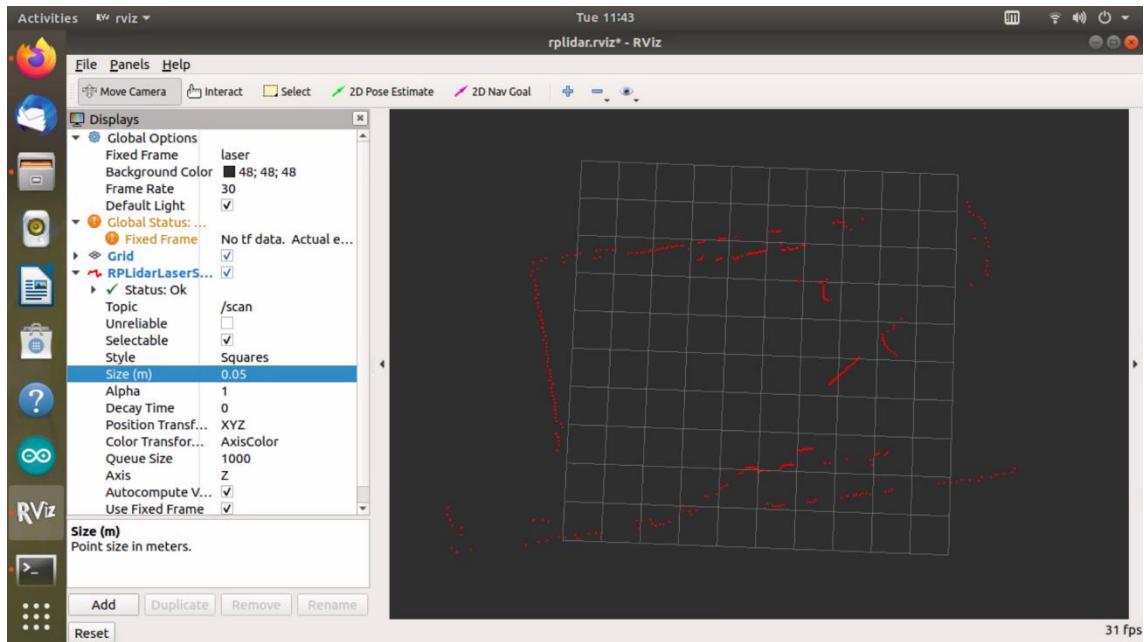


Figure 53. RPLIDAR detected points in a laboratory at EUSS

As the results were easily obtained with this LIDAR, it was decided to equip the small prototype for testing with this one after errors were not fixed for 3i LIDAR. This decision implied having to change the organization of the components in the upper platform, because both LIDARS did not fit in the same place. Also, to take advantage of the situation, Raspberry Pi was equipped with a bigger and better heat dissipater. After rearranging the components, the robot was equipped as shown in Figure 54.

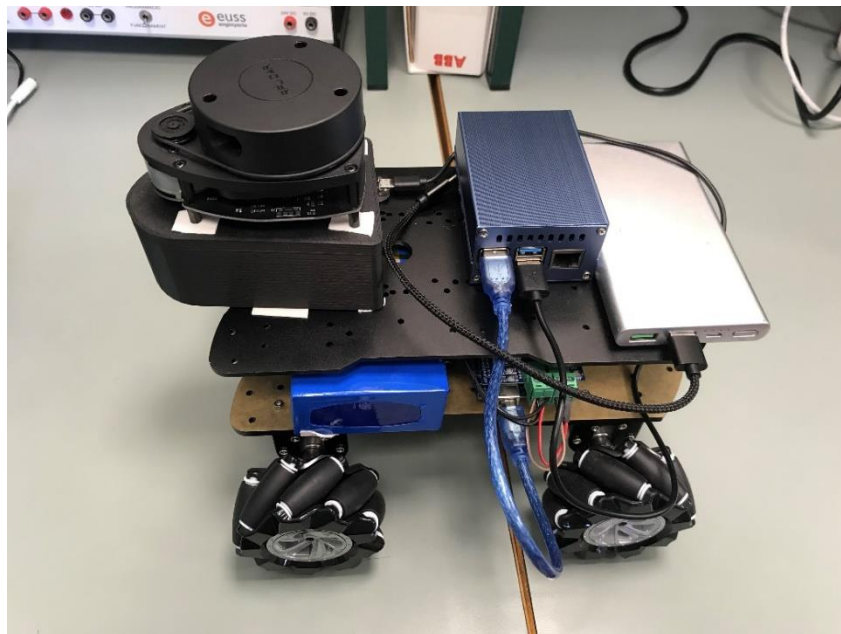


Figure 54. Rearanged components of the small prototype

After having Arduino and LIDAR sensor ready, the next step was to test an autonomous navigation file, that means, executing nodes (.py files). However, when trying to launch LIDAR and move the wheels at the same time it did not function well.

By trying to connect them separately and together using different physical USB ports, it was deduced that there was a collision of USB ports, because both were connecting to `/ttyUSB0` at the same time. This made the LIDAR stop working when the `rosserial` node was executed. To solve this, one of the ports needed to be changed so the script of the RPLIDAR package `node.cpp` was modified so that the LIDAR connected to a different port (in this case, USB1 was specified). The port was specified in line 205 of the script provided by the developers, using the `sudo su` command, because the `rplidar_ros` package was blocked:

```
203     nh_private.param<std::string>( tcp_ip , tcp_ip, 192.168.0.1 );
204     nh_private.param<int>("tcp_port", tcp_port, 20108);
205     nh_private.param<std::string>("serial_port", serial_port, "/dev/ttyUSB0");
206     nh_private.param<int>("serial_baudrate", serial_baudrate, 115200/*256000*/);
207     nh_private.param<std::string>("frame_id", frame_id, "laser_frame");
```

Figure 55. Specification of serial port in `node.cpp` file (RPLIDAR GitHub repository: https://github.com/Slamtec/rplidar_ros/)

After solving this problem, the next steps for this project are to execute autonomous navigation nodes and implement SLAM routines for mapping and trajectory planning.

5.6.3. Final equipped prototype

The first commands were also tested in the equipped prototype designed by an EUSS engineering student [36]. The achieved movements were the following:

- Sending a twist message to move the wheels
- Using `teleop_twist_keyboard` package to move the robot using the computer keyboard, achieving an holonomic controlled movement.

In this case, the LIDAR was not tested together with the Arduino, which is a step that has to be taken in a near future.

5.7. Limitations and improvements

The main limitation of this project was the time, which played a major role when it came to achieving objectives. Due to the delay in correcting errors, the final activities could not be developed. An important limitation was the time spent in fixing 3i Robotix LIDAR, which was included in the kit of the small prototype. For other versions of this project, a good recommendation would be to buy the components separately instead of buying a kit, so it can be personalized and it can be guaranteed that it will work. Also, it is very important to make sure there is available official documentation on the components, as one of the problems with the 3i LIDAR was that it was very difficult to find information and datasheets or other users who had experienced similar problems, which is also very useful when errors come up. The more popular is the component, the more possibilities to find solutions to problems that might show up.

6. Execution timelines

This section shows the activities and tasks that were carried out during the project, as well as their organization along time and the order of execution.

6.1. Work Breakdown Structure (WBS)

The Work Breakdown Structure is a scheme that organizes tasks in groups, it is shown in the figure below (Figure 56). Some of the tasks were not only carried out by the author of the project but also by other members of this project.

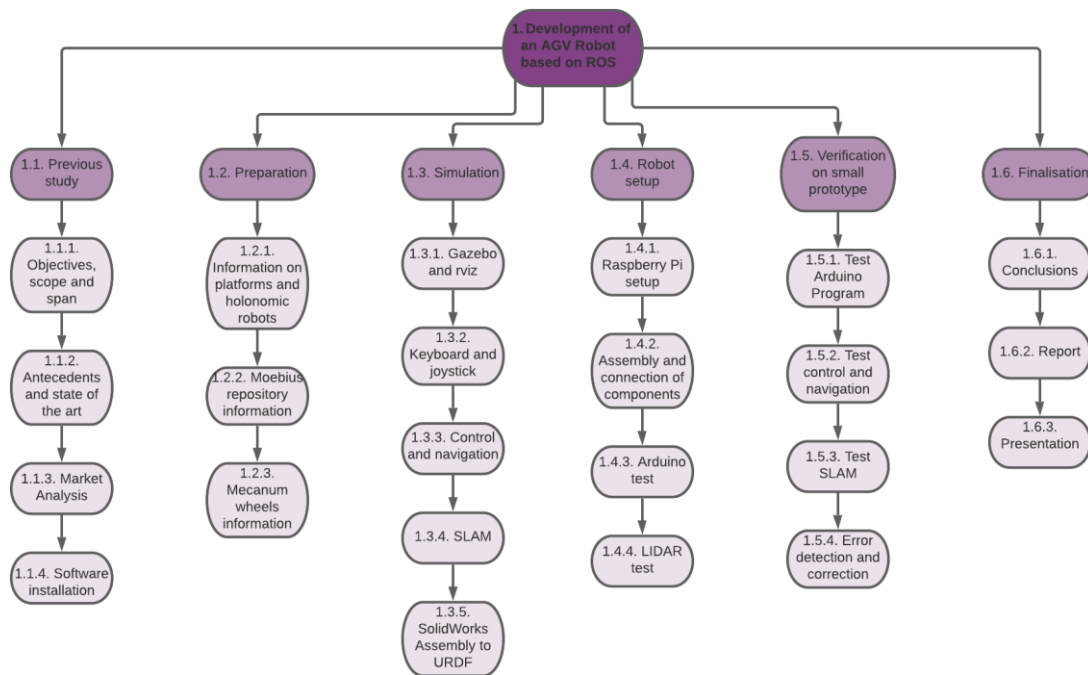


Figure 56. WBS showing from darkest to lightest colour: project, groups of tasks, tasks.

Activity	Description
1.1. Previous study	This includes tasks to study the background of the project and define what wants to be achieved in this project.
1.1.1. Objectives, scope and span	Definition of concrete goals of the project as well as potential users of the product.
1.1.2. Antecedents and state of the art	Research of information on already developed products and current state of robotics in healthcare.
1.1.3. Market Analysis	Historical evolution of the market, potential sectors and users of the final product.
1.1.4. Software installation	Installation of VirtualBox, Ubuntu, ROS, Visual Studio Code and any other software required, as well as packages and extensions.
1.2. Preparation	This group of tasks includes those that are a preparation for further tasks, for finding components of the robot and getting started on the assembly.
1.2.1. Information on platforms and holonomic robots	Research of information about what is an holonomic robot, characteristics and possible platforms and robots to be used.
1.2.2. Moebius repository information	Deep look at Moebius repository of the small testing device: assembly, components...
1.2.3. Mecanum wheels information	Research of available professional mecanum wheels for the final prototype, list of characteristics of each, and choose the best fit for the project.

1.3. Simulation	Use of virtual environments (Gazebo and rviz) on Ubuntu in VirtualBox, to test the packages and nodes. Simulate the developed code on a computer before trying it on a real-life robot.
1.3.1. Gazebo and rviz	First commands to test Gazebo and rviz environments, verify that the robot can be seen to perform other actions afterwards.
1.3.2. Keyboard and joystick	Download the proper packages to move the robot using the keyboard of the computer or a joystick connected to it.
1.3.3. Control and navigation	Testing control and navigation packages, which consist in nodes that make the robot move autonomously and avoid obstacles using LIDAR.
1.3.4. SLAM	Testing SLAM packages, which acquire the information on robot position and obstacles, obtaining a map.
1.3.5. SolidWorks Assembly to URDF	Exporting a SolidWorks Assembly to URDF file to test the commands simulating the final designed prototype.
1.4. Robot setup	Acquiring all hardware and software to finally test the code.
1.4.1. Raspberry Pi setup	Downloading Ubuntu image, Ubuntu desktop, ROS, Arduino IDE, rosserial, and packages required.
1.4.2. Assembly and connection of components	Using the Moebius tutorial, fixing all components on the platform and performing the corresponding connections between RB Pi, LIDAR and batteries.
1.4.3. Arduino test	Testing Arduino Mega 2560 board using an example program.
1.4.4. LIDAR test	Executing first commands of LIDAR and viewing on rviz the scanned surfaces.
1.5. Verification on small prototype	Putting together all the tested software and making sure it works on a real-life robot.
1.5.1. Test Arduino program	Making sure the Arduino IDE program can be uploaded to the Arduino board and testing first commands to make sure the wheels spin.
1.5.2. Test control and navigation	Testing control and navigation package on a real-life robot.
1.5.3. Test SLAM	Testing of SLAM routines on a real-life robot.
1.5.4. Error detection and correction	Final identification of errors on the robot and software when running the code on the robot and suggestion of improvements that can be implemented.
1.6. Finalisation	Closure of the project and final presentation.
1.6.1. Conclusions	Conclude the acquired knowledge during the project.
1.6.2. Report	Elaboration of a report with appropriate sections of an engineering project describing the work done.
1.6.3. Presentation	Elaboration of a presentation about the most important aspects of the project.

Table 3. WBS Dictionary. Description of each activity and group of activities.

6.2. Task matrix

The following table shows the task matrix, which defines the relationship between tasks, by which task is each preceded and which come afterwards.

Task	Activity	Description	Previous	Consequent	Duration (weeks)
1.1.1.	A	Objectives, scope and span	-	B	1
1.1.2.	B	Antecedents and state of the art	A	C	1
1.1.3.	C	Market Analysis	B	D	1
1.1.4.	D	Software Installation	C	E, F	2
1.2.1.	E	Information on platforms and holonomic robots	D	G	2
1.2.2.	F	Moebius repository information	D	G	2
1.2.3.	G	Mecanum wheels information	E, F	H	2
1.3.1.	H	Gazebo and rviz	G	I, L	1
1.3.2.	I	Keyboard and joystick	H	J	1
1.3.3.	J	Control and navigation	I	K	1
1.3.4.	K	SLAM	J	M	2
1.3.5.	L	SolidWorks Assembly to URDF	H	M	4
1.4.1.	M	Raspberry Pi setup	K, L	N	1
1.4.2.	N	Assembly and connection of components	M	O, P	1
1.4.3.	O	Arduino test	N	Q	1
1.4.4.	P	LIDAR test	N	Q	1
1.5.1.	Q	Test Arduino program	O, P	R	1
1.5.2.	R	Test control and navigation	Q	S	1
1.5.3.	S	Test SLAM	R	T	1
1.5.4.	T	Error detection and correction	S	U, W	1
1.6.1.	U	Conclusions	T	-	1
1.6.2.	V	Report	-	-	19 (Duration of the project)
1.6.3.	W	Presentation	T	-	1

Table 4. Task matrix.

6.3. Gantt

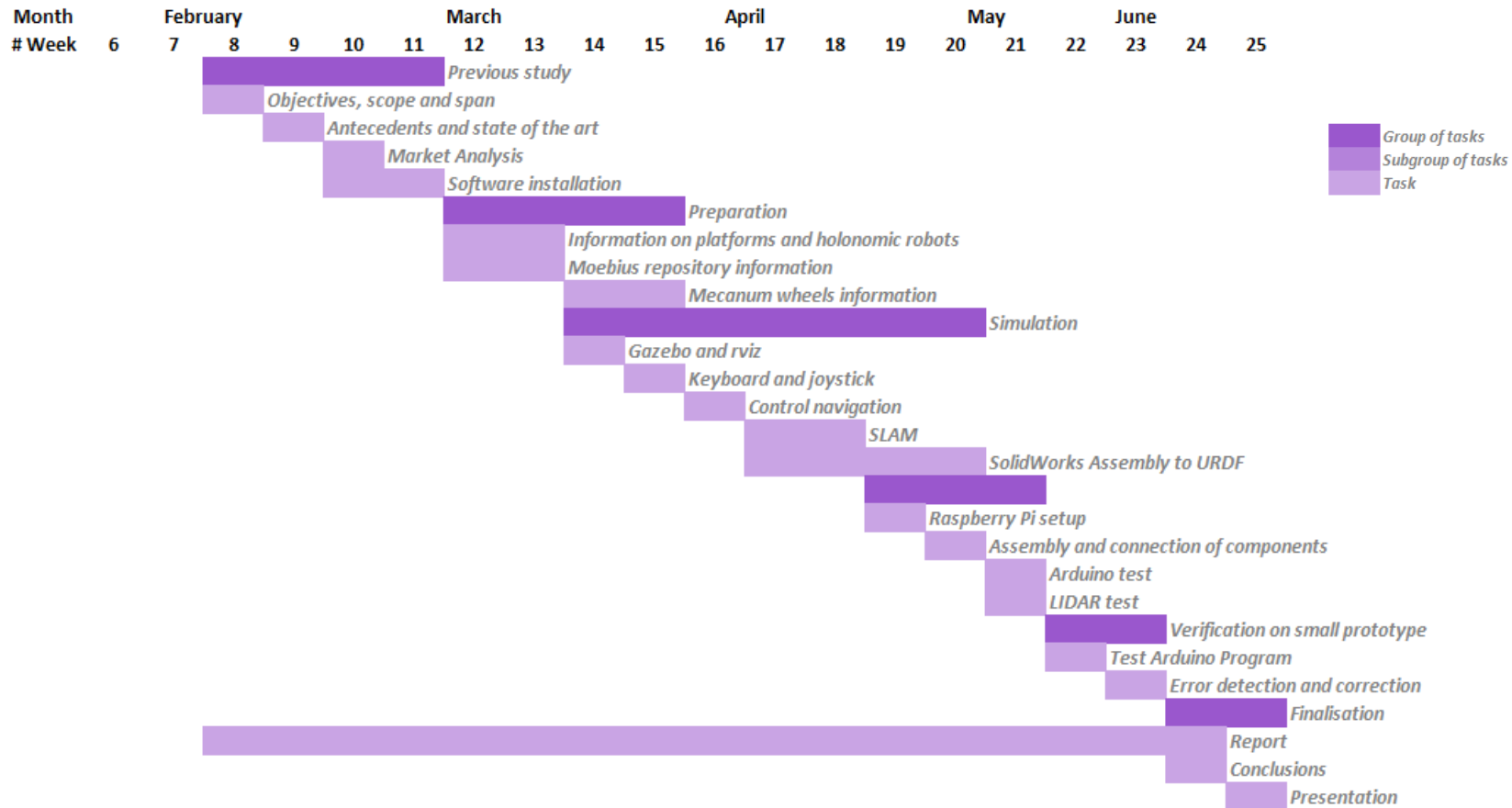


Figure 57. Gantt chart

The previous figure (Figure 57) shows the timeline and activities that were carried out along time. Note that not all the activities explained in WBS are included. That is due to some activities whose time was extended due to unexpected problems that came up during the project. This led to a lack of time to complete some of the final activities, but the project can be extended up to two weeks after the publication of this report in order to finish the remaining tasks.

7. Technical feasibility

The proposed solution's technical feasibility is studied with a SWOT analysis (Strengths, Weaknesses, Opportunities and Threats). The SWOT analyses all the aspects: the robot, ROS, Raspberry Pi and the robot application.

7.1. SWOT Analysis

7.1.1. Strengths

- Raspberry Pi is very affordable, small-sized, allows working from home and installing Ubuntu and ROS
- Experience in Python, which is the programming language with which Raspberry Pi works
- ROS is an open-source platform that is widely used; therefore, it could probably be solved using forums available on the internet and the official ROS webpage if there was any programming problem.
- ROS includes SLAM libraries, which are used for the localization of the robot, also ROS allows using Python.
- Robot does not disturb sanitary professionals because it works during night.
- The robot disinfects and sanitises air, therefore reducing infectious agents.
- Less infectious diseases are spread inside the hospital, which means it is safer for critical patients.

7.1.2. Weaknesses

- Short experience in robotics, which could complicate the node programming even though the author has experience in Python programming from other courses at Universitat de Barcelona. This could lead to extend the needed time for some activities.
- Raspberry Pi is designed for teaching and learning programming mostly for beginners which means it is not as strong as a computer for some tasks.
- Raspberry needs other hardware components (mouse, keyboard, screen, power supply) to be set up as well as an SD card that carries all system information.

7.1.3. Opportunities

- Proximity to Hospital Clínic in Barcelona, where the robot can be tested once it is ready, also belonging to the University of Barcelona, eases communicating with infrastructures and logistic departments from the hospital to test the device.
- The current available AGV robots are mainly meant to transport trolleys and big loads, while this robot pretends to disinfect rooms, which is now an emerging technology and it can find its spot in the market.
- Hospitals in Barcelona are now integrating this type of technology, which can be a market opportunity.
- The developed code can be modified to adapt the robot to the different hospitals, environments and even other components of the robot.

- COVID-19 pandemic is a very good opportunity to explain the usefulness of these robots and increase awareness on hospital safety.

7.1.4. Threats

- Sanitary professionals and patients not adapting to the robot, especially older people that are not used to this type of technology and might find it uncomfortable or a threat due to
- If the code is not programmed correctly it might lead to difficulties when trying to adapt it to other hospitals.
- Security: it must be taken into account that the intention is that the robot navigates by itself through the hospital with no vigilance, meaning it can be stolen if no security features are implemented.
- Ozone is a toxic substance, if the robot sensor fails it can lead to severe repercussions on hospital staff and patients' health.

As a result of the SWOT Analysis, it is concluded that as a new product it has very good opportunities in the market especially with current arising technologies, and threats can be overcome by integrating a security system and acquiring an accurate ozone sensor, performing the corresponding calibrations and testing it in empty environments before taking it to a real hospital.

8. Economic feasibility

The following tables show the cost of the current project, including the hours of development. Some of the components were retrieved from Amazon as an example of what could be used such as the screen, mouse, keyboard and 5V battery by searching for simple components that can have the work done, even though more expensive components can also be used, therefore it is the minimum that needs to be spent on them. The chosen models, however, are all well-reviewed by users.

Component	Individual price (€)	# needed	Total amount (€)
Raspberry Pi 4 Model B 4GB [30]	59,90	1	59,90
ROS	0	1	0
Python	0	1	0
Ubuntu 16.04	0	1	0
Mouse [31]	5,99	1	5,99
Keyboard [32]	13,99	1	13,99
Screen [33]	61,99	1	61,99
VirtualBox	0	1	0
Small Prototype	333,39	1	333,39
12V Battery [34]	49,96	1	49,96
5V Battery[35]	22,99	1	22,99

Extra assembly components ²	-	-	100
RPLidarA1M8[36]	83,09	1	83,09
Development hours	10/h	300	3000
TOTAL			3731,30

Table 5. Economic Study on the components needed for the project

The economic feasibility of the whole project, however, is higher because it includes more students and components. The costs of building the prototype were analysed by the EUSS Engineering student who designed it:

ELEMENT	UNITS	INDIVIDUAL PRICE	TOTAL
3D Printer	1	169.56€	169.56€
Raspberry Pi 4B (4GB) + Heat dissipater	1	60.99€ + 15.59€	76.58€
SSD + frame	1	24.99€ + 19.99€ (ORICO)	44.98€
Official Raspberry Pi 4B power supplier	1	8.95€	8.95€
Arduino Mega	1	35€	35€
Mecanum wheel kit (4 units)	1	60.55€	60.55€
Motor and encoder	4	10.50€	42€
RPLidarA1M8	1	83.09€	83.09€
Winsen ozone sensor	1	22.21€	22.21€
Power driver	2	0.74€	1.48€
12V battery	2	7.39€	14.78€
Adjustable source XL4015E1	1	5.88€	5.88€
IMU	1	3.99€	3.99€
Screws M6x30 DIN 933	32	0.08€	2.56€
M6 DIN 934	32	0.04€	1.28€
PLA 1Kg 1.75mm	2	24.99€	49.98€
PETG 1Kg 1.75mm	1	18.99€	18.99€
Particle board 800x600x16mm	1	7.40€	7.40€
TOTAL			648.26€

Table 6. Estimated cost of materials for final prototype [36].

For further replications of this project, the estimated cost of building both prototypes was analysed and is shown in the tables below:

Component	Price (€)	Where to buy
Raspberry Pi 4 Model B 4GB	51,87 ³	https://es.aliexpress.com/item/4000054878108.html
Arduino Mega2560	41,94	https://www.robotshop.com/es/es/microcontrolador-arduino-mega-2560-rev3.html
Arduino shield	21,72	https://www.aliexpress.com/i/4000261831657.html
12V Battery	49,96	https://www.amazon.es/Eleoption-122000-12-recargable-inal%C3%A1mbrico/dp/B075487FRL/

² It includes wiring components, it is an estimated price, computed as the maximum that should be spent on the components.

³ Minimum price that needs to be spent.

5V Battery	22,99	https://www.amazon.es/SWEYE-Capacidad%EF%BC%BDCargador-M%C3%BAltiples-Protecciones-Smartphones/dp/B0814JMRLR/
Extra assembly components ⁴	100	-
RPLidarA1M8	101,52	https://www.robotshop.com/es/es/rplidar-a1m8-kit-desarrollo-escaner-laser-360-grados.html
Base platform, motors and wheels	61,97	https://es.aliexpress.com/item/4001121790912.html
Heat Dissipater	13,32	https://www.robotshop.com/es/es/estuche-aluminio-c-ventilador-doble-y-disipador-calor-para-raspberry-pi-4.html
TOTAL	451,97	

Table 7. Estimated cost for replicating small prototype

Component	Price (€)	Where to buy
Raspberry Pi 4 Model B 4GB	51,87 ⁵	https://es.aliexpress.com/item/4000054878108.html
Arduino Mega2560	41,94	https://www.robotshop.com/es/es/microcontrolador-arduino-mega-2560-rev3.html
RPLidarA1M8	101,52	https://www.robotshop.com/es/es/rplidar-a1m8-kit-desarrollo-escaner-laser-360-grados.html
Particle board 800x600x16mm	7,40	Servei Estació (C/ d'Aragó, 270, 272, 08007 Barcelona)
Mecanum wheel kit	122,75	https://www.robotshop.com/es/es/ruedas-mecanum-aluminio-127mm-c-rodillos-basicos-2x-izquierda-2x-derecha.html
Motor and encoder	42 (10,50x4)	https://es.aliexpress.com/item/4000979634560.html
Winsen ozone sensor	22,21	https://www.winsen-sensor.com/sensors/o3-gas-sensor/me3-o3.html
Power driver	1,36	https://es.aliexpress.com/item/4000046656442.html
12V battery	7,47	https://es.aliexpress.com/item/32794930186.html
Adjustable source	5,88	https://www.e-ika.com/fuente-de-alimentacion-regulable-4-38v-5a-con-display
IMU	8,06	https://es.aliexpress.com/item/4001178597929.html
Screws M6x30 DIN 933	2,56	-
PLA 1Kg 1.75mm	49,98	-
PETG 1Kg 1.75mm	18,99	-
TOTAL	483,99	

Table 8. Estimated cost for replicating bigger equipped prototype.

9. Regulations and legal aspects

As it has been mentioned in the scope section, the objective of this project is not the commercialization of the final product (AGV robot) but the development of the software. However, the regulations that can be applied to the software are analysed in this section and the regulations

⁴ It includes wiring components, it is an estimated price, computed as the maximum that should be spent on the components.

⁵ Minimum price that needs to be spent.

that the developed AGV robot will have to fulfil in future projects so it can be sold in the European Market.

To analyse the different regulations that can be applied to the developed software first it is needed to define if it is a medical device, as the regulations are very strict for devices of this category.

According to article 2 from Directive 93/42/EEC:

'medical device' means any instrument, apparatus, appliance, software, material or other article, whether used alone or in combination, including the software intended by its manufacturer to be used specifically for diagnostic and/or therapeutic purposes and necessary for its proper application, intended by the manufacturer to be used for human beings for the purpose of:

- diagnosis, prevention, monitoring, treatment or alleviation of disease,
- diagnosis, monitoring, treatment, alleviation of or compensation for an injury or handicap,
- investigation, replacement or modification of the anatomy or of a physiological process,
- control of conception

and which does not achieve its principal intended action in or on the human body by pharmacological, immunological or metabolic means, but which may be assisted in its function by such means; [37]

By knowing this, one may think it is a medical device, however, according to Annex IX, rule 15: "All devices intended specifically to be used for disinfecting medical devices are in Class IIa, unless they are specifically to be used for disinfecting invasive devices in which case, they are in class IIb", this means that only disinfecting devices directed to be used on medical devices are actually considered medical devices, therefore, this product is not considered a medical device, nor are the software or the components. The AGV robot will not need to bear the CE marking, as it is only a must for medical devices. However, the robot needs to fulfil requirements on product safety and certification. [38]

Directive 2001/95/EC on general product safety (GPSD) ensures items are safe and it is applied to any product to be sold in the European Market. Following standards acquire the certification and if they do not exist, the certification on this rule is made through commission guidelines and consumer safety expectations among others. [39]. Also, Directive 85/374/EEC on liability for defective products establishes that 'the producer of a product is liable for damages caused by a defect in his product' [40].

Also, it must be taken into account that the robot will emit UV light to achieve the disinfection desired and Directive 2006/25/EC considers the exposure of workers to risks arising from physical agents, including UV radiation [41].

10. Conclusions and future work

10.1. Conclusions

The project's closure consisted of drawing conclusions, which need to be tightly related to the objectives to analyse the work done and the progress of the project.

- Develop a final degree project in the field of Biomedical Engineering

This objective was achieved successfully, moreover, not only the author learned about a biomedical engineering application, but from other engineering specialities, which made the experience very rewarding.

- Get familiar with the different environments used and learn how to manage the different tools.

The environments were finally managed successfully and having had the errors that came up also led to the development of problem-solving skills and acquiring experience to later avoid making the same mistakes.

- Understand ROS functionalities and which are useful on an AGV robot

As the main software, ROS was studied in depth and was very well understood when it came to general terms such as nodes, packages, launch files and others. Also having worked a lot with the software, apart from having studied it, also gave good skills and experience on it, which is very successful for further projects.

- Test the developed code on a virtual environment using simulations

Even though there were problems at the beginning regarding compilation of repositories, editing `.bashrc` file and working on the correct directory, the steps taken for simulation were all successful and thanks to the experience acquired during the project it allowed later copying the process very fast compared to the time taken at the start of the project.

- Test the developed code on a small device using Arduino

This objective was partially achieved, the wheels did move and LIDAR did so, but the problem regarding port collision led to having no time for implementing autonomous navigation or SLAM routines.

- Develop teamwork skills: communication, organization, problem-solving, listening...

During the project there was a fluent communication between students and professors which allowed an easygoing flow of events. Both weekly reunions and fast e-mail replies from all the members played a major role regarding the advance of the project.

10.2. Future Work

Regarding future lines of this project, the main task that should be done is to implement autonomous navigation to the robot. The nodes are already developed so the code should be slightly modified in order to make the robot work. After achieving autonomous navigation it will be very interesting to integrate SLAM routines to scan and map a real-life room and be able to plan trajectories to direct the robot to the place it is needed.

Once this is achieved, a very important step would be to test the device working for some time at EUSS or UB and once its functioning is verified, it can also be tested at Hospital Clínic de Barcelona, verifying it useful and successful in a real clinical environment and that it does not cause problems in logistics nor with human interaction. This process should start by testing it on a small low-risk area and if it works, it can afterwards be implemented in larger areas.

11. Bibliography

- [1] Wikipedia contributors. (2020, May 31). Robot Operating System. Retrieved May 6, 2020, from https://en.wikipedia.org/wiki/Robot_Operating_System
- [2] ROS.org. (2018). Retrieved May 6, 2020, from <https://wiki.ros.org/>
- [3] Massot, A. (2020, January). *Application of ROS and SLAM for transportation in a clinical environment*. Barcelona, Spain.
- [4] Hafiz, A. (2007, November). *DESIGN AND DEVELOPMENT OF AUTONOMOUS GUIDED VEHICLE (AGV)*. Retrieved from http://umpir.ump.edu.my/id/eprint/2253/1/ABDUL_HAFIZ_BIN_MOHAMAD.PDF
- [5] Pedan, M., Gregor, M., & Plinta, D. (2017). Implementation of Automated Guided Vehicle System in Healthcare Facility. *Procedia Engineering*, 192, 665–670. <https://doi.org/10.1016/j.proeng.2017.06.115>
- [6] Robotnik Automation. (2015, March 17). AGVS for intrahospital logistics with ROS. Retrieved May 16, 2020, from <https://www.robotnik.eu/agvs-platform-for-intrahospital-logistics-ros/>
- [7] Guzman, R., Navarro, R., Beneto, M., & Carbonell, D. (2016). Robotnik—Professional Service Robotics Applications with ROS. *Studies in Computational Intelligence*, 253–288. https://doi.org/10.1007/978-3-319-26054-9_10
- [8] Koceski, S., Koceska, N. Evaluation of an Assistive Telepresence Robot for Elderly Healthcare. *J Med Syst* 40, 121 (2016). <https://doi-org.sire.ub.edu/10.1007/s10916-016-0481-x>
- [9] Soffar, H. (2018). *Robotic Pharmacy (Automated dispensing machines pharmacy) cons, pros and uses | Science online*. Science online. Retrieved 20 May 2021, from <https://www.online-sciences.com/robotics/robotic-pharmacy-automated-dispensing-machines-pharmacy-cons-pros-and-uses/>.
- [10] Wikipedia contributors. (2019, November 23). Vehicle de guiat automàtic. Retrieved from https://ca.wikipedia.org/wiki/Vehicle_de_guiat_autom%C3%A0tic
- [11] EK Automation. (n.d.). Healthcare. Retrieved May 22, 2020, from https://ek-automation.com/en/industries/healthcare/?utm_source=AGVNETWORK&utm_medium=BANNER&utm_campaign=HOSPITAL
- [12] Frontpage | Mobile Industrial Robots. (n.d.). Retrieved May 22, 2020, from <https://www.mobile-industrial-robots.com/en/>
- [13] Moxi. (n.d.). Retrieved May 22, 2020, from <https://diligentrobots.com/moxi>
- [14] Shenzhen Wellwit Robotics Co.,Ltd. (n.d.). Shenzhen wellwit Robotics Co., Ltd. official website - AGV handling robot mobile robot laser AGV las. Retrieved May 22, 2020, from <https://www.wellwit.com.cn/tcn/WDR01C.html>
- [15] Blanco, N. *La robótica en la economía*. E Innova BUCM. Retrieved 27 May 2021, from <http://webs.ucm.es/BUCM/revcul/e-learning-innova/183/art2550.pdf>.

- [16] Andreu, M. (2019). *Robótica en el ámbito sanitario y de los cuidados: implicaciones para la privacidad y la protección de datos*. Dilemata.net. Retrieved 29 May 2021, from <https://www.dilemata.net/revista/index.php/dilemata/article/view/412000292>.
- [17] *La COVID-19 acelera la implantación de robots como mano de obra*. Computerworld.es. (2020). Retrieved 8 June 2021, from <https://www.computerworld.es/tecnologia/la-covid19-acelera-la-implantacion-de-robots-como-mano-de-obra>.
- [18] Dolic, Z., Castro, R., & Moarcas, A. (2019). *Robots in healthcare: a solution or a problem?*. European Parliament. Retrieved 27 May 2021, from [https://www.europarl.europa.eu/RegData/etudes/IDAN/2019/638391/IPOL_IDA\(2019\)638391_EN.pdf](https://www.europarl.europa.eu/RegData/etudes/IDAN/2019/638391/IPOL_IDA(2019)638391_EN.pdf).
- [19] Rodríguez, C. (2021). *Los robots, un nuevo valor en situación de pandemia*. La Vanguardia. Retrieved 27 May 2021, from <https://www.lavanguardia.com/tecnologia/20201111/49398424256/los-robots-un-nuevo-valor-en-situacion-de-pandemia.html>.
- [20] *AliExpress - Compra online de Electrónica, Moda, Casa y jardín, Deportes y ocio, Motor y seguridad, y más. - AliExpress - AliExpress*. AliExpress. (2021). Retrieved 1 June 2021, from <https://es.aliexpress.com/>.
- [21] *Amazon.es: compra online de electrónica, libros, deporte, hogar, moda y mucho más.* Amazon.es. (2021). Retrieved 1 June 2021, from <https://www.amazon.es/>.
- [22] *RobotShop | Robot Store | Robots | Robot Parts | Robot Kits | Robot Toys*. Robotshop.com. (2021). Retrieved 10 June 2021, from <https://www.robotshop.com/>.
- [23] Raghavan, R. (2019, September 12). AN OVERVIEW OF THE MICROSOFT ROBOTICS DEVELOPER STUDIO. Retrieved May 20, 2020, from <https://acodez.in/microsoft-robotics-developer-studio/>
- [24] Robot Staff. (2014, June 18). Service Oriented Architectures – Two leading systems, MRDS and ROS, point to the future of robotics. Retrieved May 20, 2020, from <https://www.botmag.com/service-oriented-architectures-two-leading-systems-mrds-and-ros-point-to-the-future-of-robotics/>
- [25] MRPT – Empowering C++ development in robotics. (2020). Retrieved May 20, 2020, from <https://www.mrpt.org/>
- [26] CARMEN. (n.d.). Retrieved May 20, 2020, from <http://carmen.sourceforge.net/intro.html>
- [27] *Exchange Data with ROS Publishers and Subscribers*. Es.mathworks.com. (2021). Retrieved 11 April 2021, from <https://es.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html>.
- [28] Puig, M. *rUBot_mecanum_ws*, (2021), GitHub repository, https://github.com/manelpuig/rUBot_mecanum_ws
- [29] Aqeel, A. (2018). *Introduction to Arduino Mega 2560 - The Engineering Projects*. The Engineering Projects. Retrieved 1 June 2021, from <https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-mega-2560.html>.

- [30] *Raspberry Pi 4 Modelo B 4GB*. Pccomponentes.com. (2021). Retrieved 1 June 2021, from [https://www.pccomponentes.com/raspberry-pi-4-modelo-b-4gb?codesf=10507899363&utm_medium=cpc&utm_campaign=Shopping-feed&utm_term=Raspberry+Pi+4+Modelo+B+4GB&utm_source=googleShopping\(via+Shopping+Feed\)&gclid=CjwKCAjwtdcFBhBAEiwAKOly56FKjYFnLLuQrQr8X1EcrHk6KdJwMYxx-_gLRxBY5eKHEEFbM02BNxoC87EQAxD_BwE](https://www.pccomponentes.com/raspberry-pi-4-modelo-b-4gb?codesf=10507899363&utm_medium=cpc&utm_campaign=Shopping-feed&utm_term=Raspberry+Pi+4+Modelo+B+4GB&utm_source=googleShopping(via+Shopping+Feed)&gclid=CjwKCAjwtdcFBhBAEiwAKOly56FKjYFnLLuQrQr8X1EcrHk6KdJwMYxx-_gLRxBY5eKHEEFbM02BNxoC87EQAxD_BwE).
- [31] Logitech M90 - Ratón con Cable, Color Negro. (n.d.). Retrieved June 5, 2020, from https://www.amazon.es/Logitech-M90-Rat%C3%B3n-cable-color/dp/B003D8ZT0C/ref=sr_1_14?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=rat%C3%B3n&qid=1591387563&sr=8-14
- [32] Rii RK907 USB - Teclado con Cable, QWERTY español, Negro. (n.d.). Retrieved June 5, 2020, from https://www.amazon.es/Rii-RK907-USB-Teclado-espa%C3%B1ol/dp/B0832FHGKD/ref=sr_1_5?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=teclado&qid=1591430524&sr=8-5
- [33] Acer Essential - Monitor de 19.5" (pantalla LED, 1600 x 900 píxeles, puerto VGA, 16.2o W), color negro. (n.d.). Retrieved May 6, 2020, from https://www.amazon.es/Acer-Essential-Monitor-pantalla-p%C3%ADxeles/dp/B00DAST78K/ref=sr_1_5?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=pantalla+ordenador&qid=1591430566&sr=8-5
- [34] *Eleoption 122000 - Batería de alta calidad, CC, 12 V, potente: 20000 mAh, iones de litio, recargable, para cámara de vigilancia con transmisor inalámbrico*. Amazon.es. (2021). Retrieved 1 June 2021, from https://www.amazon.es/Eleoption-122000-12-recargable-inal%C3%A1mbrico/dp/B075487FRL/ref=pd_sbs_60_6/259-4385486-7341636?_encoding=UTF8&pd_rd_i=B075487FRL&pd_rd_r=c6b138b0-dffa-4f0c-94c4-5ac6325d656a&pd_rd_w=5aG4A&pd_rd_wg=XEKDf&pf_rd_p=dd738298-9698-4d96-aa46-cc7f9fd38c59&pf_rd_r=TP3WYBM675BTGZT1K24K&pvc=1&refRID=TP3WYBM675BTGZT1K24K.
- [35] *SWEYE Batería Externa 26800mAh Carga Rápida de Power Bank 2 USB Cargar y Luces LED [Ultra Capacidad] Cargador Móvil Portátil Compacto con Múltiples Protecciones para Android Smartphones Tabletas, etc.* Amazon.es. (2021). Retrieved 6 June 2021, from https://www.amazon.es/SWEYE-Capacidad%EF%BC%BD-Cargador-M%C3%BAltiples-Protecciones-Smartphones/dp/B0814JMRLR/ref=sr_1_9?__mk_es_ES.
- [36] Maynau, S. (2021) *Disseny d'un sistema omnidireccional de quatre rodes per a entorns hospitalaris*.
- [37] European Parliament. (1993, June). COUNCIL DIRECTIVE 93/42/EEC of 14 June 1993 concerning medical devices. Retrieved June 1, 2020, from <https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1993L0042:20071011:EN:PDF>
- [38] *Classification - are disinfecting products medical devices?*. Medica-tradefair.com. (2014). Retrieved 2 June 2021, from https://www.medica-tradefair.com/en/News/Archive/Classification_-_are_disinfecting_products_medical_devices.
- [39] *EUR-Lex - I21253 - EN - EUR-Lex*. Eur-lex.europa.eu. (2015). Retrieved 2 June 2021, from <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISSUM%3AI21253>.

[40] Directive 85/374/EEC - liability for defective products - Salud y seguridad en el trabajo - EU-OSHA. Osha.europa.eu. (2021). Retrieved 2 June 2021, from <https://osha.europa.eu/es/legislation/directives/council-directive-85-374-eec>.

[41] McGinn, C., Scott, R., Donnelly, N., Roberts, K., Bogue, M., Kiernan, C., & Beckett, M. (2021). Exploring the Applicability of Robot-Assisted UV Disinfection in Radiology. *Frontiers In Robotics And AI*, 7. <https://doi.org/10.3389/frobt.2020.590306>

12. Annexes

12.1. Tutorial: How to export a SolidWorks Assembly into an URDF file. Gazebo visualization.

1. Install ROS exporter `sw_urdf_exporter` installer from the following website: https://wiki.ros.org/sw_urdf_exporter. Once it is downloaded, execute it.
2. Open your SW assembly. Select *Tools->Export as URDF*, as shown in the picture below:

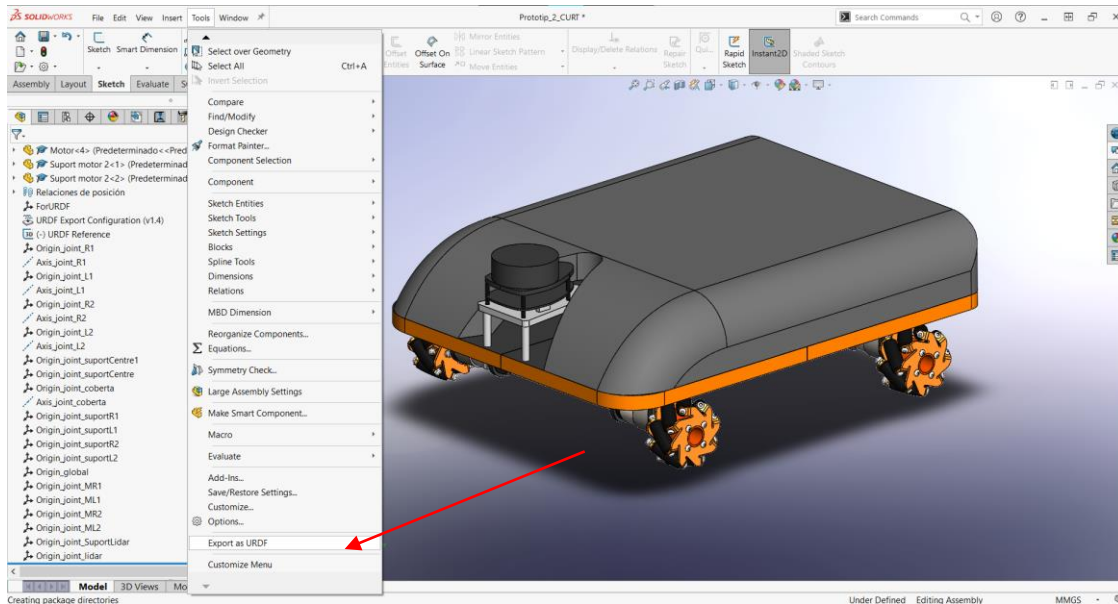


Figure 58. URDF plugin in SW

3. Choose coordinate System. It can be generated automatically but you might need to generate it manually using the option *ForURDF* in the *Global Origin Coordinate System* drop-down, so that the robot does not appear upside down when using gazebo.

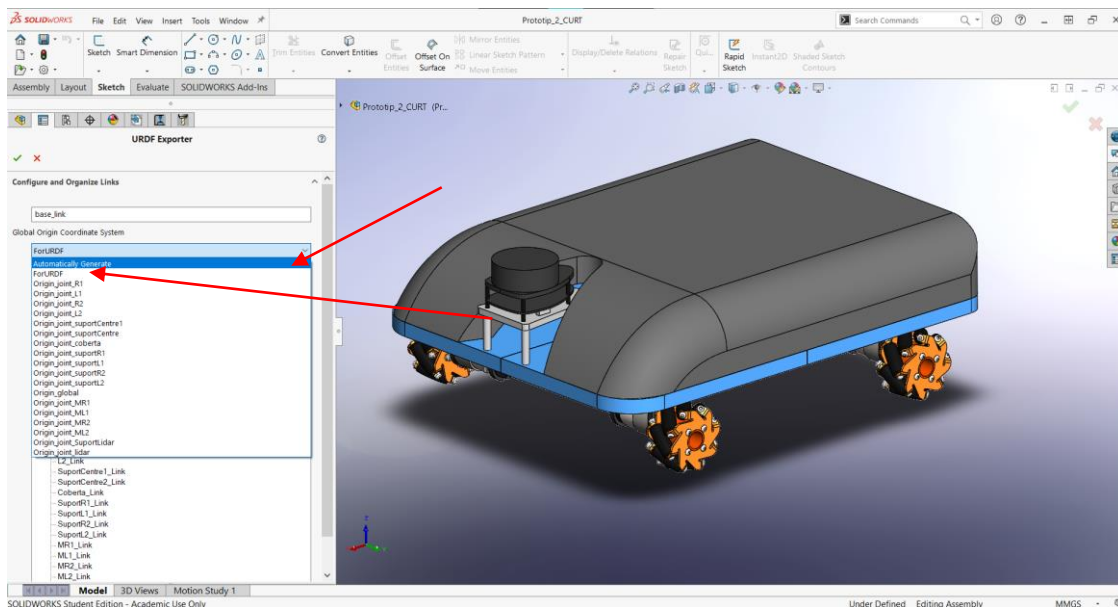


Figure 59. Coordinate system

- Choose the base link. To do so, select the base of your robot at the right part of the screen (3D prototype). Once the base is selected, add as many child links as you wish. In this case, the platform was formed by 4 parts.

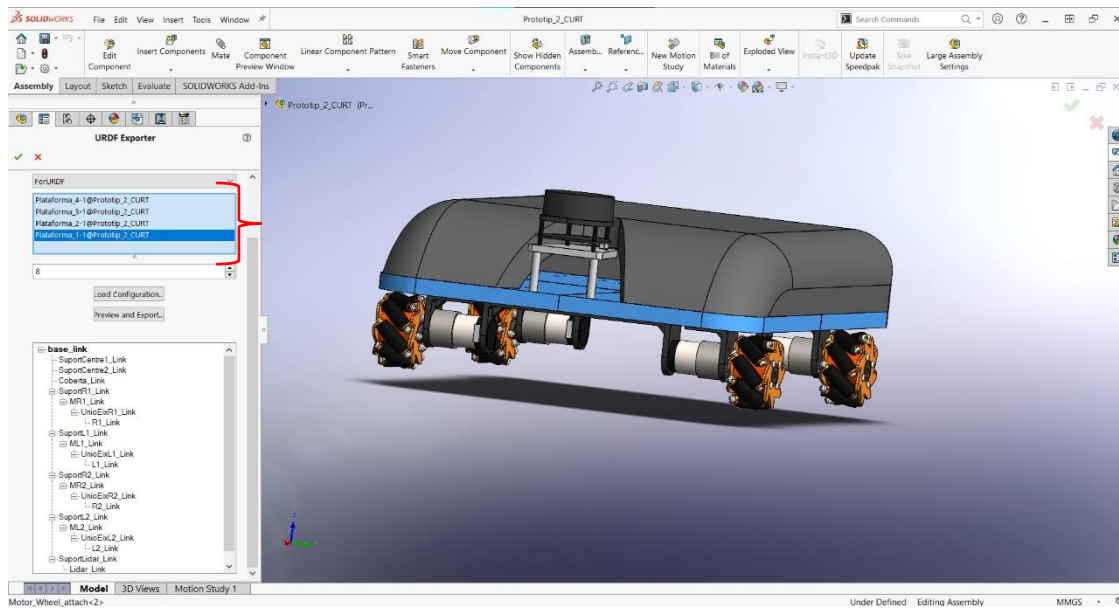


Figure 60. Base link selection

- To build the prototype, child links can be added to the tree structure. To do so, right-click on the link and select *Add Child*, if they are added to the wrong link, drag it to the correct one. Each link can have several child links but only one parent link. To edit each link, click on its name on the tree structure.

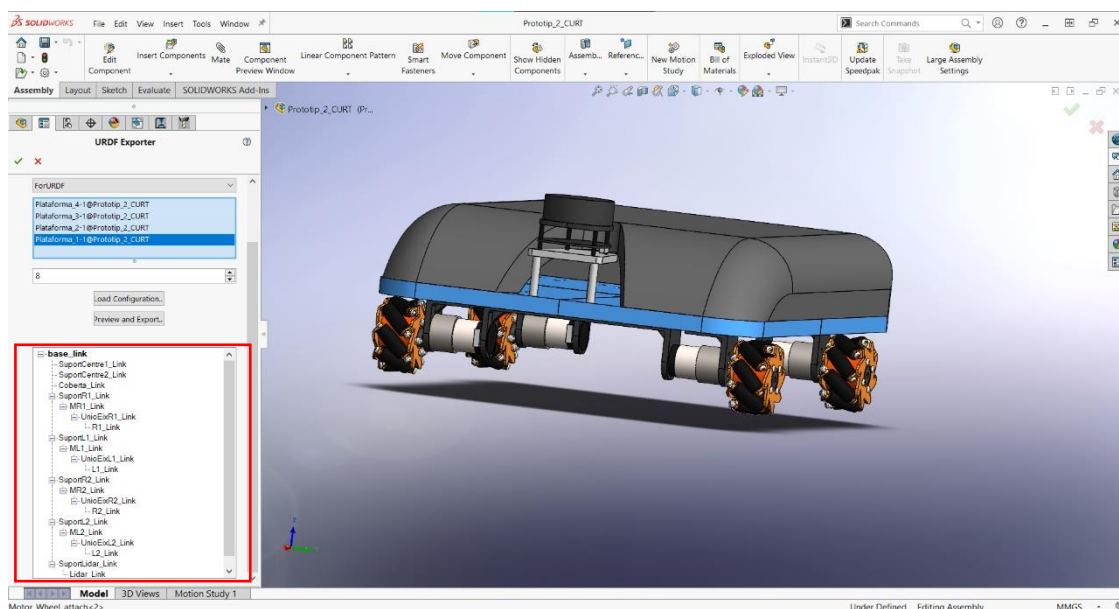


Figure 61. Tree structure of the robot.

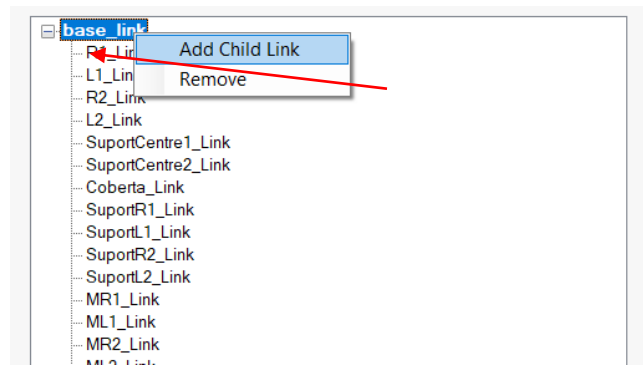


Figure 62. How to add child links.

- When editing each link, the left part of the screen will look like the following figures. A reference coordinate system, a reference axis and the type of joint need to be defined for each link. To select the link components, click on them on the 3D model. Also, set a joint name for each link. In this case, the reference coordinate system and the reference axis were automatically generated and all the joint types were set to fixed except for the wheels, which were continuous.

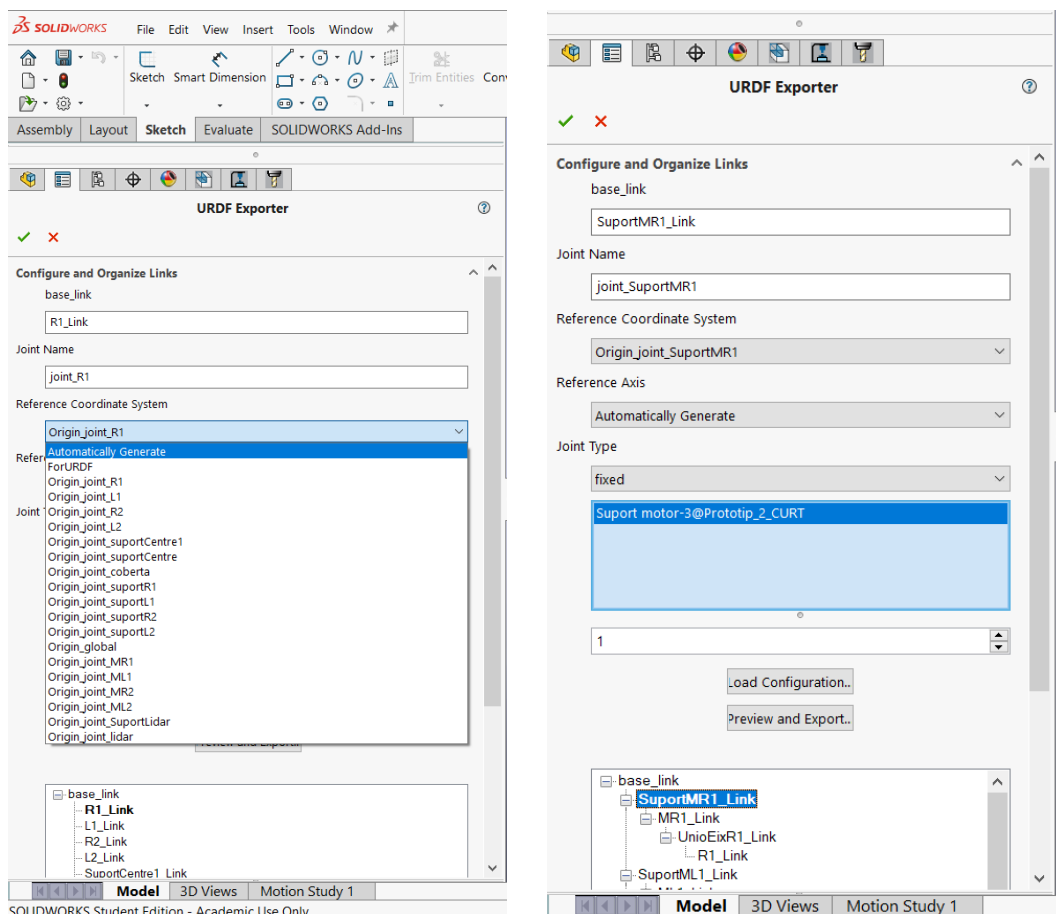


Figure 63. Link configuration

- Once all the links are well defined, click *Preview and Export*.

8. A new window pops up to edit joint properties. If everything is correct, click *Next*.

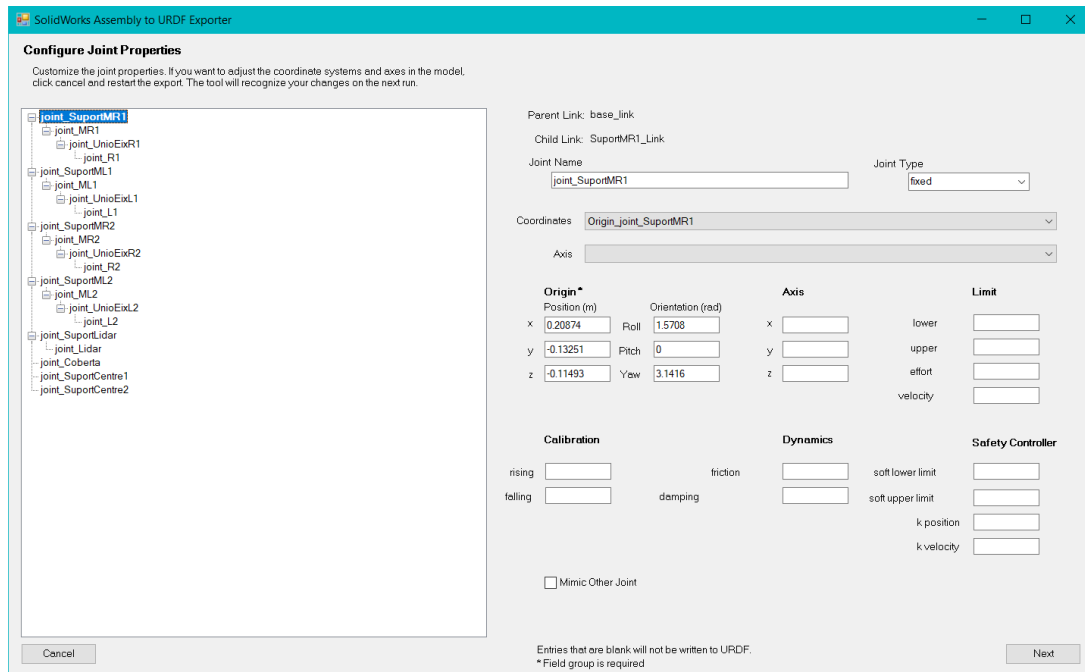


Figure 64. Joint properties window.

9. Another window pops up to configure link properties. If they are correct, URDF or URDF and meshes can be exported. A package is created.

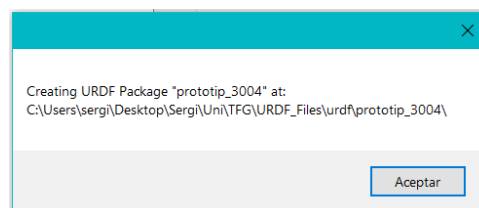
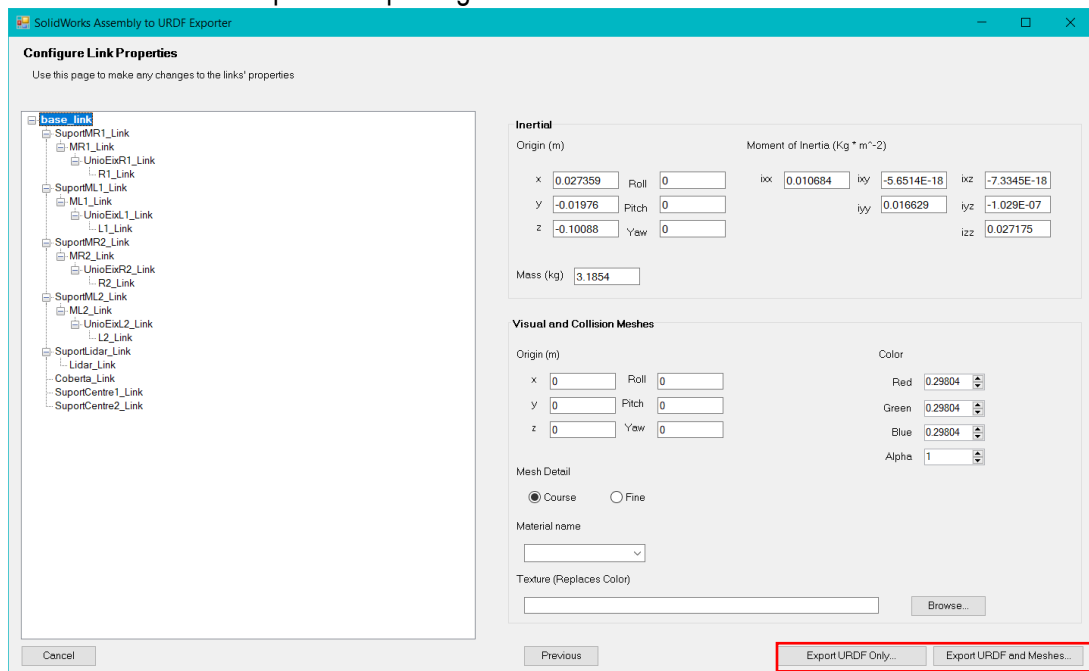


Figure 65. Link properties window.

10. Reference axis and systems are generated in SW if they were chosen to be generated automatically.

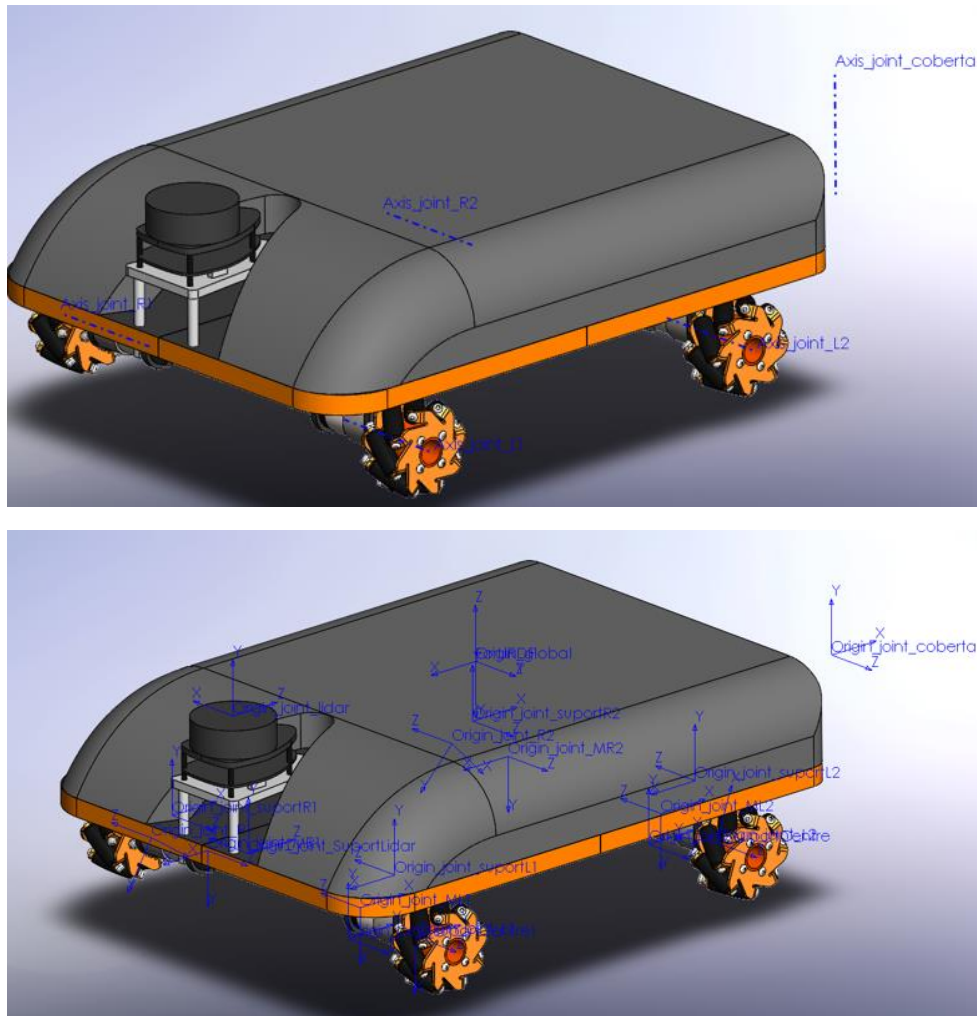


Figure 66. Reference axis and systems.

11. For the visualization in Gazebo, move or copy the package to your `src` folder in ubuntu. and use the following command:

```
catkin_make
```

This will create `build` and `devel` folders and a `.txt` file. In this case, the package is inside `demo_car` folder in the following path:

```
marta@marta-VirtualBox:~/catkin_workspace/src/urdf_demo/demo_car$
```

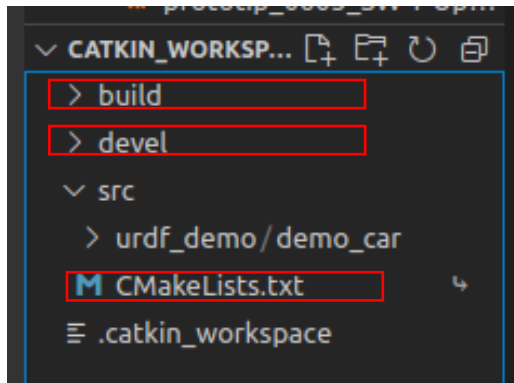


Figure 67. catkin_workspace structure

12. Open a new terminal in the directory of your package and execute the following command to see your prototype in Gazebo:

```
roslaunch gazebo.launch
```

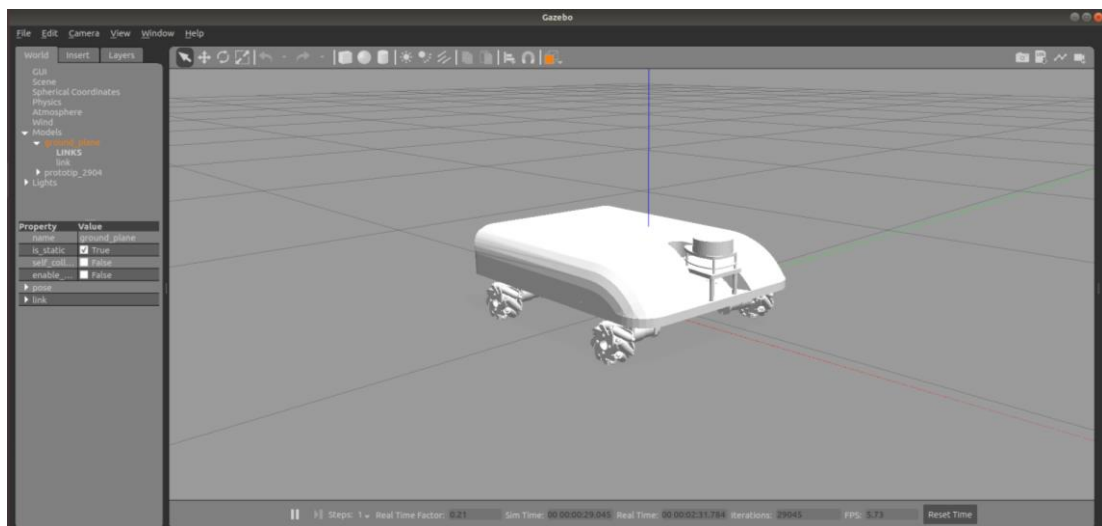


Figure 68. URDF prototype in gazebo.

Notice the robot appears in white colour. You can change the colour of each link by adding the following lines to the URDF file:

```
<gazebo reference="link_name">
  <material>Gazebo/Colour</material>
  <gravity>>true</gravity>
</gazebo>
```

Example:

```
952 <gazebo reference="base_link">
953   <material>Gazebo/Orange</material>
954   <gravity>>true</gravity>
955 </gazebo>
```

Figure 69. Gazebo colour plugin example

Insert the name of your link in "link_name" and write its colour in Colour. Find the predefined Gazebo materials in: http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials.

Once the colour are added, save the file and execute again the `roslaunch` command.

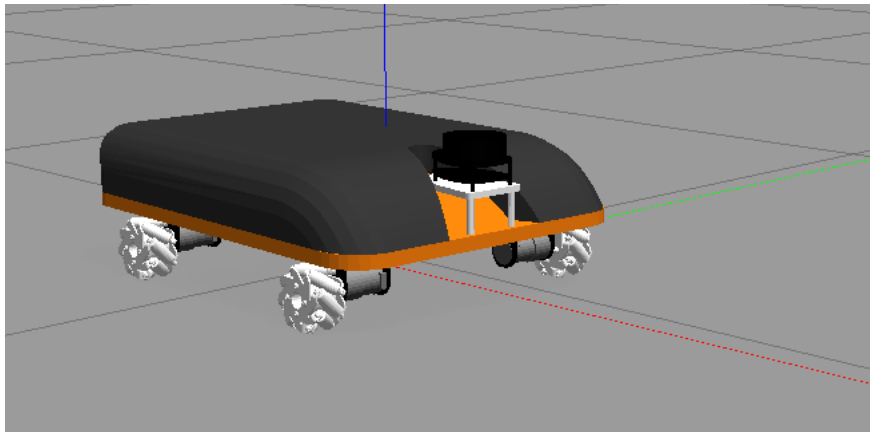


Figure 70. URDF spawned in gazebo applying colour.

13. Add gazebo plugins to make the robot move to the URDF file and save it.

```
<gazebo>
  <plugin name="Mecanum_controller" filename="libgazebo_ros_pl
  anar_move.so">
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <odometryRate>50.0</odometryRate>
    <robotBaseFrame>base_footprint</robotBaseFrame>
  </plugin>
</gazebo>
```

Now that the plugin is added, launch the gazebo file again and use the following command to use the keyboard to move the robot in Gazebo:

```
roslaunch teleop_twist_keyboard teleop_twist_keyboard.py
```

```
Reading from the keyboard and Publishing to Twist!
-----
Moving around:
u   i   o
j   k   l
m   ,   .

For Holonomic mode (strafing), hold down the shift key:
-----
U   I   O
J   K   L
M   <   >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

Figure 71. Instructions for keyboard driving.

14. Add LIDAR plugin to the URDF file to add the scan, remember to change the reference "base_scan", use the name of your LIDAR link:

```
<!-- Laser Distance Sensor YDLIDAR X4 controller-->
<gazebo reference="base_scan">
  <sensor name="lds_lfcd_sensor" type="ray">
    <pose>0 0 0 0 0 0</pose>
    <visualize>>false</visualize>
    <update_rate>5</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>720</samples>
          <resolution>0.5</resolution>
          <min_angle>0.0</min_angle>
          <max_angle>6.28319</max_angle>
        </horizontal>
      </scan>
    <range>
      <min>0.12</min>
      <max>10</max>
      <resolution>0.015</resolution>
    </range>
    <noise>
      <type>gaussian</type>
      <!-- Noise parameters based on published spec for YDLI
DAR X4 is 1.5% at half range 4m (= 60mm, "+-160mm" accuracy at m
ax. range 8m). A mean of 0.0m and stddev of 0.020m will put 99.7
% of samples within 0.16m of the true reading. -->
      <mean>0.0</mean>
      <stddev>0.02</stddev>
    </noise>
  </ray>
  <plugin filename="libgazebo_ros_laser.so" name="gazebo_ros
_lds_lfcd_controller">
    <!-- topicName>/gopigo/scan</topicName -->
    <topicName>scan</topicName>
    <frameName>base_scan</frameName>
  </plugin>
</sensor>
</gazebo>
```