Master thesis:

MECHATRONIC DESIGN OF AN OMNIDIRECTIONAL ROBOTIC PLATFORM AND ITS NAVIGATION BASED ON GRAPH SLAM IMPLEMENTATION

Present:

Carlos Prados Sesmero

Supervised by:

Manuel Ferre Pérez

Mario Di Castro

Master in Automatic and Robotic

Robotic Department (ETSII)

January 2020

Acknowledgments

I would like to thank, first of all, to Mario Di Castro and Manuel Ferre for organizing this experience. I am always going to be grateful for this huge chance you have given me that has allowed me to grow professionally and humanly.

To my family, especially my parents and sister, who have made my stay in CERN more pleasant. Thank you for your unconditional support and help, for your continuous encouragement and for always have time and space to dedicate to me.

To Silvia for being closer than never in spite of the long distance. This experience would not have been possible without your positive energy and help. Thanks very much.

To all of you who have taken part in the robotic group.

Abstract

One of the most significant problems in underground tunnels is the survey of the proper performance of the security sensors available along the whole corridor, especially when dealing with the safety of personnel working in one of the largest underground tunnel complexes in the world.

Intending to enhance the process of inspection in long tunnels, this project presents the mechatronic design of an autonomous robot for the SPS accelerator of CERN, where any amount of restriction is present due to its actual state. Besides, its kinematic and dynamic control is specified.

Furthermore, a graph-SLAM algorithm, which uses the robot odometry, inertial motion, visual odometry, and point clouds acquisition, for the robot location while generating a reconstruction of the environment is presented.

Besides, to find a radiation-resistant system, a study of microcontrollers under radioactivity has been prepared.

Keywords

Robotic Platform - Omnidirectional Robot - GraphSLAM - Selflocation - Control.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

One of the most significant problems in underground tunnels is the survey of the proper performance of the security sensors available along the whole corridor, especially when dealing with the safety of personnel working in one of the largest underground tunnel complexes in the world. Inspection of large tunnels can be laborious when performed by operators. This process can be automated by robots, reducing inspection time and avoiding possible risks to people.

To enhance the process of inspection, a complex net of robots is being developed in the robotic group of CERN[1]. SPS, one of the particle accelerators, has particularly high radioactivity during its operation, so it is strongly desired to replace the maintenance personal by robots.

The main objectives of this thesis are the mechatronic design and the development of the autonomous location system with the goal of the inspection of long tunnels. After that, a study of some microcontrollers under radioactivity is prepared. The objective of this part is to determine the resistant capability of them in different cases (radioactive exposure time, radioactivity quantity, with and without cover, etc.).

The structure of this project is the following:

In this chapter, an introduction to the project is written. In the next one, the objectives of this project are raised. With these goals, a study of the state of art is developed, putting the project into context.

Within the project development, thorough development of the robotic platform is carried out, selecting the proper mechanical, electronic and electrical devices, making studies about the platform stability and presenting the mechanical, electronic and electrical design.

The kinematic and dynamic control of the omnidirectional platform is detailed in the following chapter.

Then, to localize the robot within the environment, a graph-SLAM approach is explained.

Finally, a study of some microcontrollers under radioactivity is prepared. This study is expected to consist of test the resistance of some electronic devices under different cases of radiation.

[1] CERN: Center for European Nuclear Research

Within the project evaluation, some conclusions and future development lines are exposed before the bibliography, in order to continue with the present project.

Finally, some appendices have been included.

Here, a project Gantt is shown. This graphic tool exposes the dedicated time planned for the different tasks throughout the project development time.

Besides, a study of the staff salary, the used materials and resources are detailed for an economical study.

Lastly, a working guide for the study of the microcontrollers and some datasheets are presented.

1.1. Objectives and project motivation

In this section, the project objectives are presented.

The **first three parts** of the project consist in develop a mechatronic design of a robotic platform, control it and develop a self-location system that determines the robot position with an accuracy of 10cm.

The Super Proton Synchrotron, SPS, was switched on in 1976 and nowadays it is the second largest machine in CERN's accelerator complex [**3**]. Measuring nearly 7 kilometers in circumference, the corridor of the tunnel becomes completely monotonous, so typical location systems that use ultrasonic sensors and lasers are hopelessly useful.

Radiation sensors, which measure the radiation steadily, are located at completely random distances in the ring. These sensors are essentials to guarantee material and personal security. Due to the necessity of sensors surveys, inspections must be carried out every month. The actual surveillance is performed by internal employees, so it is strongly recommendable to replace these workers with automatic systems.

The Large Hadron Collider, LHC, disposes of the TIM, which performs this hard task. However, SPS doesn't dispose of rails, so other automatic techniques must be developed.

A robotic autonomous system must be developed to perform the task of radiation survey, so the **first part** of this project must present the mechanical design of the platform, while the **third part** must present the self-location approach. Some specific objectives are presented in each chapter of the corresponding part.

Currently, there are very few autonomous tasks in underground tunnels made by robots, almost everything is teleoperated. For this reason, this project is expected to present innovative techniques for a lot of companies related to the robot self-location. These techniques can be used by any robot that meets the minimum characteristics.

The **fourth part** of the project development is the preparation of a study of microcontrollers under radioactivity. The main objective in this part is to dispose of a PCB[2] able to support high quantities of irradiation in a contaminated environment to its performance like an integrated controller in the robotic arm joints. In these environments, the main problem is found in the microcontrollers, since it is the first that breaks down due to radiation, while the rest of the components withstand greater amounts of time.

This last part is being worked on due to a collaboration between the Polytechnic University of Madrid and DONES, which is looking for robots able to work under very high radioactive levels, far superior to those working in other environments.

[2] PCB: Printed circuit board

CHAPTER 2

State of the art

During this chapter, several topics related to the project will be addressed with the objective of compare and contrast them in order to take suitable options in the several decisions that are presented during the development of the project.

There are three main topics to tackle. The first treats about the facilities with radioactivity where the robots are present, the second treats about robots for inspection in a variety of environments and the third treats about some SLAM algorithms.

2.1. Facilities with radioactivity where robots take part

Radioactivity refers to the particles (with or without matter) which are emitted from nuclei as a result of nuclear instability[**1**]. The most usual types of radiation are called alpha, beta, and gamma, whose penetration force is represented in Figure 2.1.1.

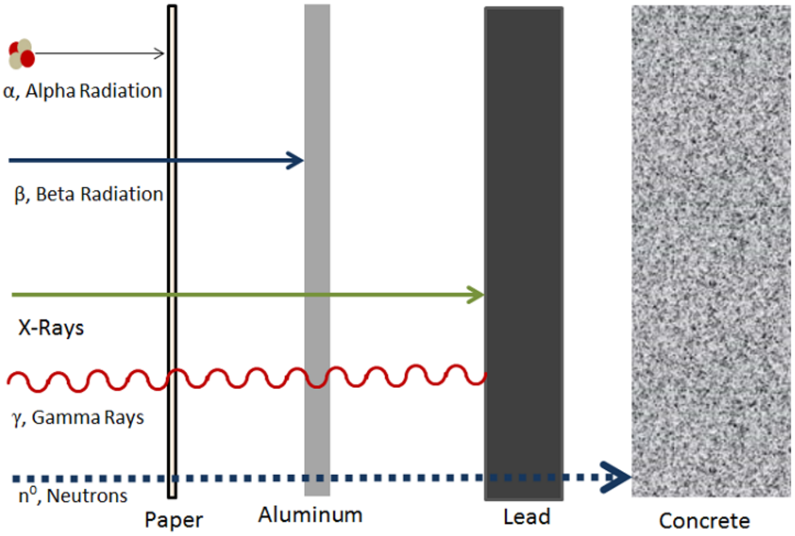


FIGURE 2.1.1. Penetration force of the radiation types

In this section, some facilities where robots work under radiation are going to be mentioned. The selection of this sort list has been set due to the proximity with them, since

Chapter 7 has been developed for DONES, while the rest of the chapters of the ”Project Development” have been developed for CERN..

2.1.1. DONES.

Between the different facilities which work with radioactivity, it is possible to find DONES[3]. DONES is a facility located in Granada (Spain) whose main objective is to recreate the terms of a fusion reactor and irradiate samples of material for the posterior analysis and studies. One of the principle applications consists of the capability to develop resistant materials for future facilities of DEMO[4]..

Three important areas exist within DONES:.

- AS (Accelerator System). The deuterium is generated.
- TS (Test System). Three rooms are available:
  - TC (Test Cell). It contains the samples, with which the accelerated deuterium collide.
  - AC (Access Cell), over the TC, contains the necessary equipment for the remote maintenance (cranes, robots, tools, etc.).
  - IWTC. Where the samples are extracted from the TC. These components are, among others, the HFTM[5] and the Target Assembly (TA).
- LS (Lithium System). Where the lithium samples are stored.

Some of these areas and rooms can be observed in Figure 2.1.2.

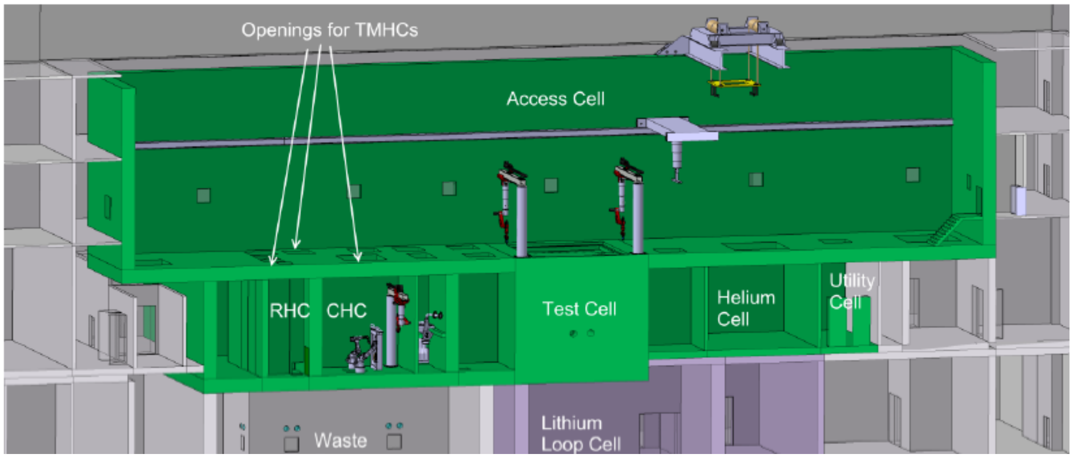


Figure 2.1.2. Layout of DONES

[3] DONES: DEMO oriented Neutron Source.
[4] DEMO: DEMOstration Power Plant.
[5] HFTM: High Flux Test Module.

In these facilities, the samples are irradiated with accelerate deuterium to study the impact of the different materials. Then, the process can be summarized in the next sequence:

- The deuterium is accelerated in AS.
- The accelerated deuterium impact against the sample in the TC, irradiating it. The samples are located in a complex system known such us HFTM.
- The HFTM is extracted from the TC through the equipment located in AC. It is transported throughout the said room and is introduced in the room IWTC.
- The components of HFTM are uninstalled to obtain the samples thanks to the use of robots, due to the existent radiation level of the materials.
- The materials are treated, packed and driven out.

The use of robots in this facility is crucial for the safety of the staff, to avoid their exposure to high levels of radiation.

2.1.2. CERN.

Probably, the most known place related to nuclear research is the European Organization for Nuclear Research. CERN is a European research organization that operates the largest particle physics laboratory in the world [**3**]. Surely, the most remarkable part of this organization is the particle accelerators complex, where various experiments are located strategically at some points. All of them are shown in Figure 2.1.3 [**4**].

CERN operates a network of 6 accelerators and a decelerator, increasing the energy of particle beams. Currently, active machines are the following [**3, 4**]:

- Two linear accelerators generate low energy particles. LINAC 2 accelerates protons for injection into the Proton Synchrotron Booster (PSB), while LINAC 3 provides heavy ions for injection into the Low Energy Ion Ring (LEIR).
- The Proton Synchrotron Booster increases the energy of particles generated by the proton linear accelerator.
- The Low Energy Ion Ring (LEIR) accelerates the ions from the ion linear accelerator, before transferring them to the Proton Synchrotron (PS).
- The Proton Synchrotron (PS) operates as a feeder to the more powerful SPS.
- The Super Proton Synchrotron (SPS), a circular accelerator with a diameter of 2 kilometers built in a tunnel. It is been used to inject protons and heavy ions into the Large Hadron Collider (LHC).
- The On-Line Isotope Mass Separator (ISOLDE), which is used to study unstable nuclei.
- The Antiproton Decelerator (AD), which reduces the velocity of antiprotons to about 10% of the speed of light for research of antimatter.

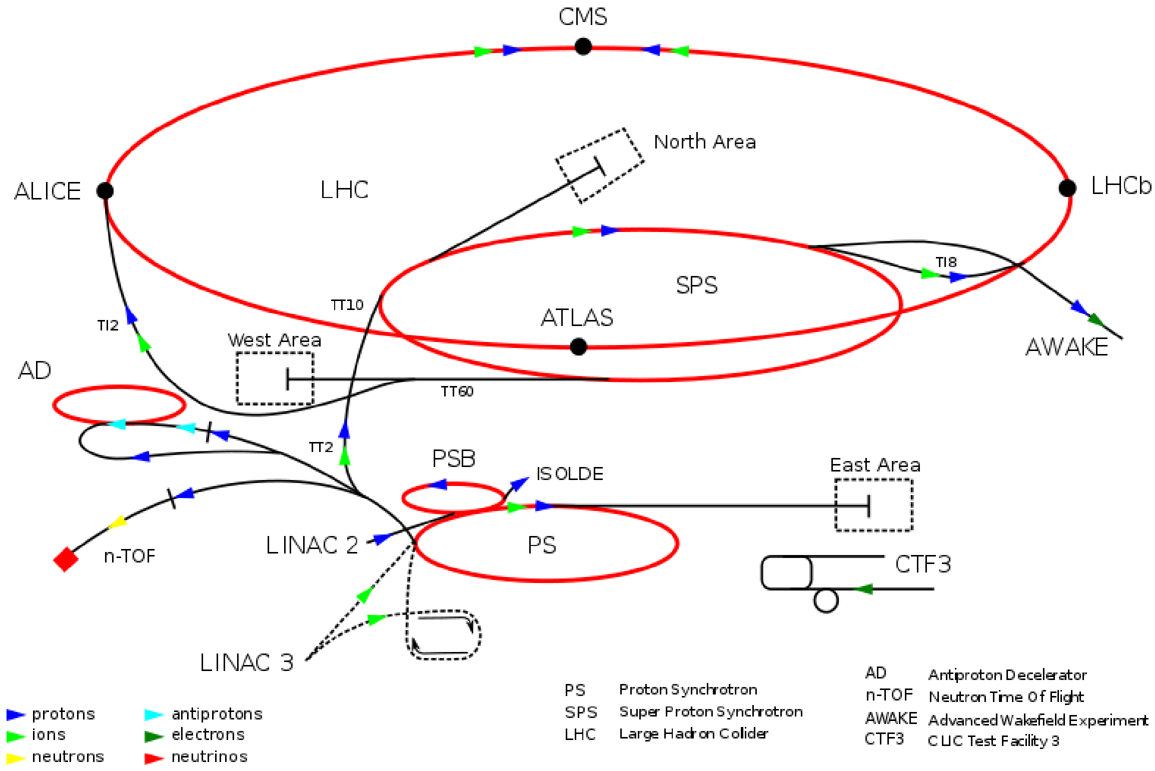


FIGURE 2.1.3. Current CERN accelerator complex.

- The Compact Linear Collider Test Facility, which studies feasibility for the future normal conducting linear collider project.
- The AWAKE experiment, which is a proof-of-principle plasma wake-field accelerator.
- The Large Hadron Collider (LHC) is a tunnel located 100 meters underground, with a length of 27 km. Insert the pre-accelerate protons and lead ions from PS/SPS and contains seven experiments.

The use of robots in this complex is crucial for the safety of the employees, to avoid their exposure to high levels of radiation. CERN disposes of a great set of robots for manipulation and survey in continuous development.

2.2. Robots for inspection

Reconnaissance and inspection might be laborious and even dangerous in hazardous environments. The best solution for those tasks can be found in the proper use of robots. These electromechanical machines can actuate in a complicated environment where radioactive, electrical and other dangers are present. Furthermore, robots improve the process of survey, reducing the time, optimizing the results and taking a lot of useful data of the place features.

In this section, some robots working for reconnaissance and inspection are described. Robots are extremely necessaries due to the danger in underground tunnels. A kind of danger can be the high level of radioactivity during the performance of the accelerators.

Robots are used for material handling and transport, person detection, sensor inspection, environment state survey, parts change and bug fix.

2.2.1. CERNbots.

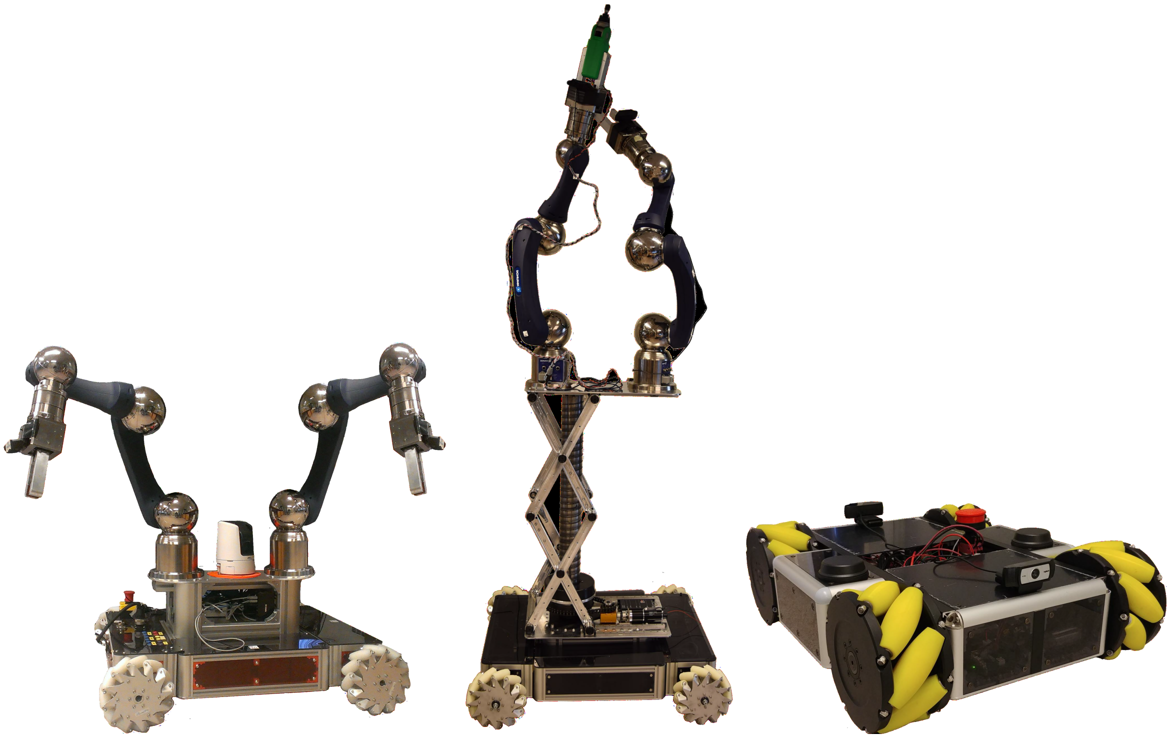
CERNbots are a set of robotic platforms developed at CERN for complex interventions in the presence of hazards like ionization radiation.

As the first CERNbot version, the second one is an omnidirectional platform, modular and flexible. Its upper module can have two robotic arms installed and can be deployed without the mechanical chassis onto other structures/platforms such as cranes etc. In addition, the chassis is very stable and can be safely operated with two robotic arms installed on a lifting chariot, which allows operations at a height of up to 3m, thus further expanding the versatility of the platform.

CERNbot uses standard industry components for most of its electronics and controls hardware making it a constantly evolving platform as the hardware is upgraded by the manufacturer. This also keeps the cost at a very competitive level for a platform with a payload of up to 250 kg and the capabilities of CERNbot.

The main applications are:

- Teleoperated interventions in hostile environments in particular in the presence of ionizing radiation.
- Complex interventions involving coordination of two robotics arms in heights up to 3m.
- Search and rescue tasks.



(A) CERNbot 1 (B) CERNbot 2 (C) CHARMbot

FIGURE 2.2.1. CERNbots and CHARMbot

All the versions can be observed in Figure 2.2.1, where CHARMbot is focused on visual inspections (it is also used for the testing algorithm).

2.2.2. TIM.

TIM[6] is a robot for LHC tunnel inspection. Its functions are the monitoring of the whole corridor of 27 km, record images (both visual and infrared), bring materials and small robots and store data of the oxygen percentage, bandwidth of the communication and temperature.

[6] TIM: Train Inspection Monorail

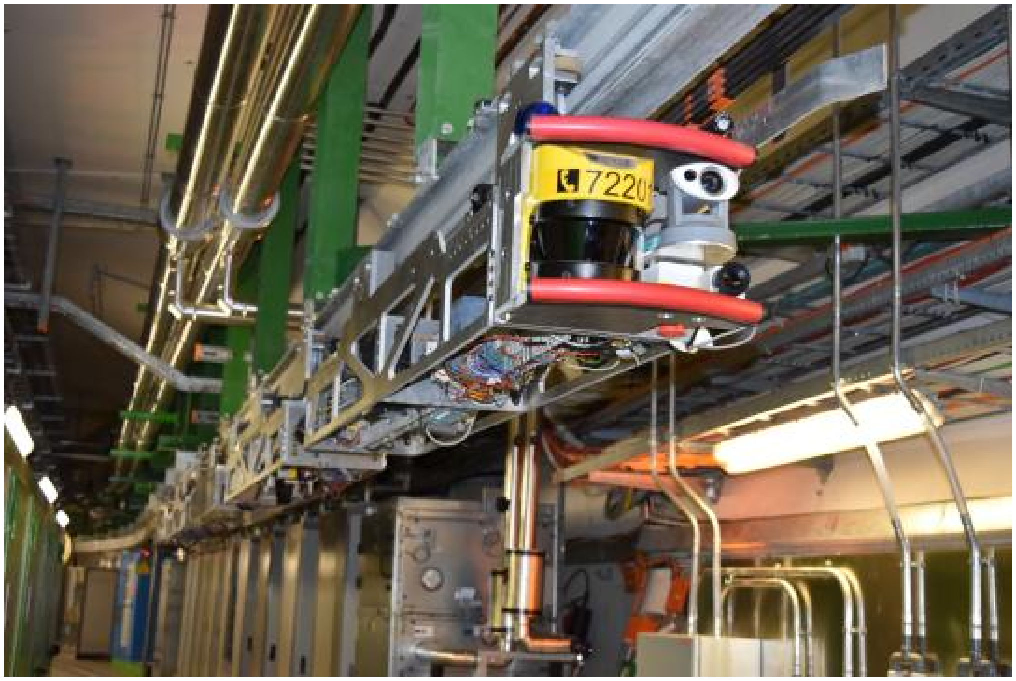


FIGURE 2.2.2. TIM

The train (shown in Figure 2.2.2) operates on a monorail that is installed on the ceiling all around a tunnel. Communication is done via 4G or Wi-Fi. It consists of five coupled wagons: control, battery, motor, payload and reconnaissance wagon, each one designed for a different purpose (modularity). This flexible robot is an autonomous system that can be operated from the surface through an advanced HMI[7]. Besides, all parts are featured with different types of cameras, sensors, PCs, controllers (almost everyone PLCs) and mechatronic systems [**5**].

2.2.3. Telemax Pro.

Telemax PRO is a versatile robot of Telerob, with a robotic arm of 7 degrees of freedom, capable of handling significant gradients and obstacles [**6**]. This robot (shown in Figure 2.2.3) allows a high range of manipulation thanks to a telescopic joint and the possibility of fold the track wheels. The control is made through a friendly HMI, where the operator can observe images provided for the 5 cameras located along with the robot.

[7] HMI: Human Machine Interface

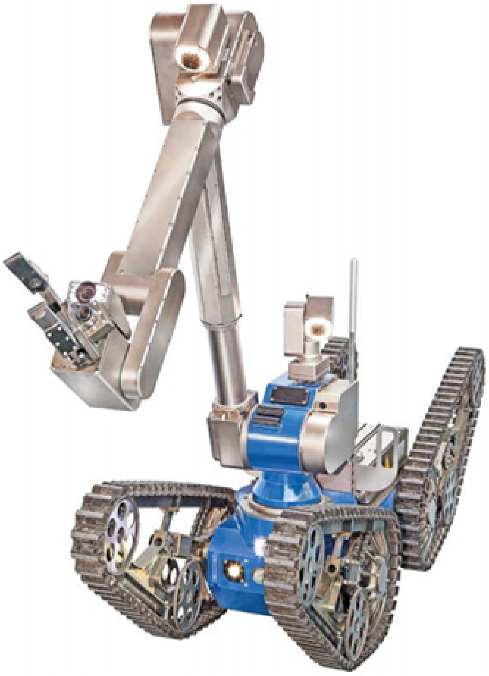


FIGURE 2.2.3. Telemax PRO

This robot is used to material handling and transport, general inspection, parts change and bug fix. Telemax is widely used at CERN in a big quantity of tasks where space is not a critical restriction.

2.3. Self-location

Robot location in an unknown or monotonous environment can become an arduous task. The problem of constructing and updating a map (usually in two or three dimensions) of an unknown environment while, at the same time, keeping track of the robot location is known as SLAM..

In this section, some SLAM algorithms are described.

2.3.1. EKF SLAM.

EKF SLAM is a class of algorithms features based witch utilizes the extended Kalman filter (EKF) for SLAM. It provides analytical estimates of the complete a posteriori probability, incrementally.

It is not an optimal solution. Computational cost raises quadratically with the number of items in the map. It is only used in small environments (unlike tunnels).

Similar to EKF SLAM, it is possible to find SEIF SLAM which is dual to the Kalman filter, where the correction step has a less computational cost.

2.3.2. FastSLAM.

Factored Solution to SLAM, FastSLAM, is an approach of the SLAM problem from a Bayesian point of view [**7**]. It decomposes the SLAM problem into a robot localization problem and a collection of landmark estimation problems that are conditioned on the robot pose estimate.

FastSLAM uses a modified particle filter for estimating the posterior over robot paths. Each particle possesses Kalman filters that estimate the landmark locations conditioned on the path estimate. Landmark estimations are efficiently represented using tree structures [**7**].

2.3.3. GraphSLAM.

GraphSLAM is a Simultaneous localization and mapping algorithm that uses sparse information matrices produced by generating a factor graph of observation interdependencies (two observations are related if they contain data about the same landmark) [**9**].

A graph-based SLAM approach constructs a simplified estimation problem by abstracting the raw sensor measurements. These raw measurements are replaced by the edges in the graph which can then be seen as “virtual measurements” [**8**].

Project development

Throughout this project, different activities are developed with the aim to comply with the set goals and achieve in the best way all the defined requirements and specifications. Every solution has been focused on functionality, quality and security point of view.

The development has been structured as follows:

Firstly, the mechatronic design of a robotic platform is carried out. Here, the platform requirement and specification are presented, making the decisions that are most affected to face the problem. A set of equations is proposed in order to find the best motor model (with gearhead and encoder), verifying the selection thanks to some simulations. In parallel form to the motor selection, some devices have been chosen as well and the mechanical design has been carried out. This design has two different phases, a preliminary design (with two extensible wheels to improve the stability) and a second model without those wheels, which have been removed thanks to a stability study. Finally, electronic and electrical design is presented, where the connexions of the motors are shown.

The next chapter presents the control system of the robotic platform from a kinematic and dynamic point of view. Here, the models are obtained, which will be used to control properly the robot, improving the quality of its movements.

In the following chapter, a graphSLAM approach for the robot is detailed. Here, the self-location system requirement and specification are presented, making the decisions that are most affected to face the problem. The main procedure to create a graph that reflects the location system is explained. After that, the used tools and libraries to create and optimize the graph are detailed. With these tools, the SLAM algorithm is done, which indicates the steps to follow to build the system. Finally, the results of the self-location system are shown.

In the last chapter, the preparation of a study of microcontrollers under radioactivity has been carried out. Here, a set of four microcontrollers is communicated with a master through RS-485 modules. The approach, the high-level communication protocol, and the electronic scheme are presented. Furthermore, some tests are shown, demonstrating the performance of the system in different cases.

CHAPTER 3

Mechatronic design of a robotic platform

In this chapter, the mechatronic design of a robotic platform is presented. How is typical in engineering projects, all the following Sections have been carried out in parallel, so some aspects are explained in later sections. The structure of this chapter is the following:

Firstly, the project requirements and specifications are explained in order to contextualize this document. According to them, important decisions have been taken to solve the problem as well as possible, proposing the development of a robotic platform. The next step has been the selection of the motor set and the batteries. They have been calculated to guarantee the robotic platform power in any case, of course, developing a study to verify its performance.

Then, the mechanical design is presented. Firstly, a preliminary design is shown and, after some stability studies of the platform, a final and compact design is presented.

Finally, the device selection and the electrical and electronic design are detailed, closing the mechatronic design in this way.

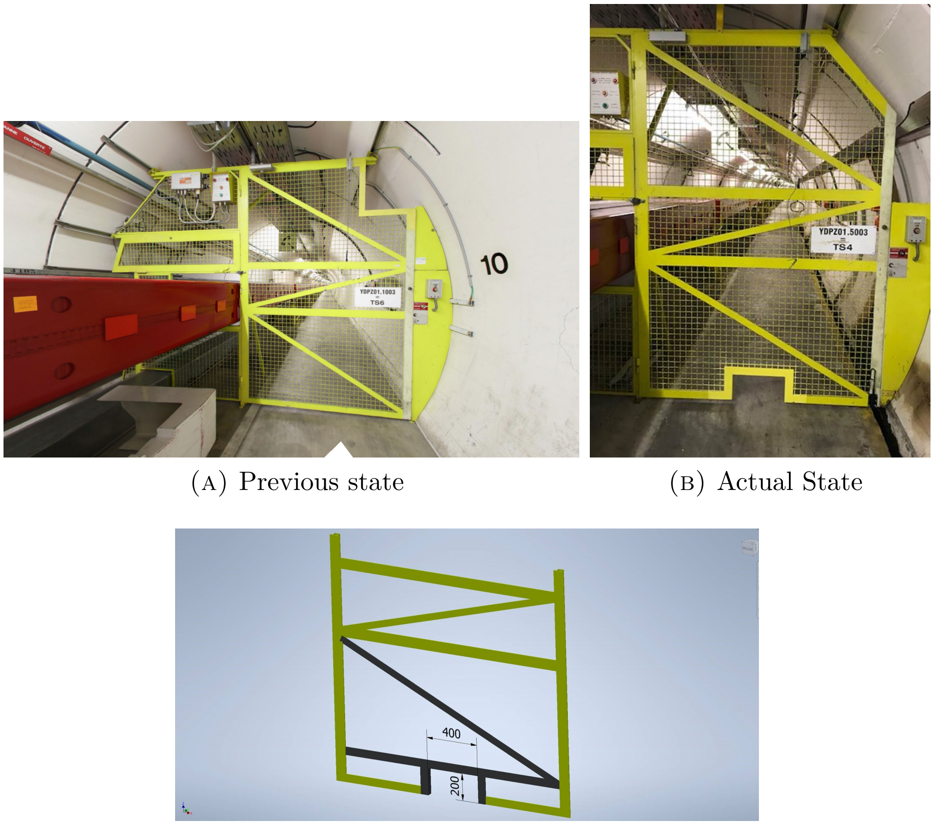
3.1. Requirements and specifications

A robotic system must be developed to perform the task of a radiation survey. A robot must start from its home, take a full turn to the ring and come back to the initial position. Since the radiation sensors are localized at 1m of height, the robot must get this heigh during the inspection.

It is expected that the robot circulates throughout the tunnel performing the survey in 80 minutes, this is, 5.5km/h or 1.5m/s.

SPS has 19 sector doors. Current internal law obligates that the doors remain closed for security reasons, so there is a problem to perform the survey with robots. In order to solve this problem, some modifications to the doors have been carried out to create an aperture that allows the robot passing through the doors.

The door modification has been approved by the committee with the constraint that the hole will be of dimensions 400x200mm (see Figure 3.1.1), so, due to this small size, the robot has a lot of constraints that must be solved. Since this project has a lot of constraints, the following requirements must be followed:



(A) Previous state

(B) Actual State

(C) Approved modification

FIGURE 3.1.1. Sector doors. A small robot is expected to cross

- In order to achieve good maneuverability to cross the small doors, arrive at the charger station through the very narrow corridors and park the robot in the charger station, the robot is expected to be holonomic.
- The accelerator complex is almost flat, with maximum slopes of 1º, so the motors must keep the velocity in any cases.
- Due to the dimension constrains and since there will not be very big obstacles, the robot is not expected to have a damping and suspension system.
- To guarantee the robot autonomy during the survey, the batteries must apply energy for 3 hours at least.

3.2. Decision making

Due to the dimension constraints to cross the doors, the robot is expected to be (~700) x 350 x 190mm. Furthermore, the robot is expected to bring a robotic arm which brings a radiation sensor. In this way, if the robot is needed in other tasks, it can be used if the tool on the robotic arm is changed.

About the locomotion system, the possible options are the following:

- Differential: A high power is required because there are only two traction wheels. The control is simple but the odometry calculation is very sensitive to errors and timely responses. If a motor fails during the survey, it would be a big problem because the robot would stay in the ring.
- Ackerman: Almost impossible to access to the tunnel and park in the charger station due to the narrow spaces (non-holonomic system).
- Tricycle: Completely discarded.
- Synchronous: Four mechanically coupled wheels, all of them turn at the same time.
  - Advantages: Reduction of the number of encoders, spin synchronism, high accuracy.
  - Disadvantages: The steering belt can become loose and generate errors, high spin friction, difficult to implement.
- Omni-directional: Four mecanum wheels.
  - Advantages: High maneuverability, easy to implement, the controllers have been already developed in the robotic group of CERN.
  - Disadvantages: High slip, bad energetic efficient and low speed (the last one is not a problem).

The final decision has been to implement an omni-directional robot. With this structure, the system is more simple, more compact and the risk when a motor fails is lower. This last point is very important, setting four wheels in rectangle, the system is redundant, this is, there are more modifiable d.o.f (4) than motion d.o.f (3, [x, y, orientation]). In case a motor fails, the system is able to continue making movements.

3.2.0.1. *Sensorization.*

The following sensors should be added:

- Two 2D lidars for object detection. They will be used to prevent the robot from crashing with obstacles, with the walls, and with the accelerator.
- Cameras for teleoperation.
- A tracking camera for visual odometry (this camera can be the same as the camera for teleoperation).
- A point-cloud generator sensor for the self-localization. It can be a 3D lidar or a RGBd camera.
- Encoders and an IMU for the localization system.

3.3. Motor set selection

In this Section, all the calculations are detailed in order to determine the proper motor and its gearhead to move the robotic platform, with his corresponding structure and locomotion system. The first step in the election of the proper components is the force and torque calculation.

3.3.1. Nominal force, torque, and power.

Since the robot is supposed to be autonomous, the maximum speed is limited. In our case, this maximum limit has been established to 2m/s (higher than required just in case of teleoperation). The equation that relates the desired linear velocity with the velocity of thee wheels (in rpm) is shown in Eq. 3.3.1 (a factor is applied to ensure good performance), while Eq. 3.3.2 shows the case this document is working on.

$$n_{in} = \frac{60}{\mathit{\Pi}} \cdot \frac{v_L}{d \cdot \eta} \tag{3.3.1}$$

$$n_{in} = \frac{60}{\mathit{\Pi}} \cdot \frac{2m/s}{0.1524m \cdot 0.8} = 313.2rpm \tag{3.3.2}$$

The input torque to move the entire robot (at nominal velocity) is established in Eq. 3.3.3. This torque depends proportionally on the force necessary to apply, which is determined in Eq. 3.3.4. In our case, since the maximum slope in the tunnel is 1º, taken the friction coefficient of the bearings as 0.01 (high value to ensure the proper robot performance) and the bearing factor such as 0.8, the results are shown in Eq. 3.3.5 and 3.3.6.

$$M_{in} = \frac{d}{2} \cdot \frac{F_L}{\eta} \tag{3.3.3}$$

$$F_L = m \cdot g \cdot \sin\theta + \mu_v \cdot m \cdot g \cdot \cos\theta \tag{3.3.4}$$

$$F_L = 40kg \cdot 9.8m/s^2 \cdot \sin 1^{\text{º}} + 0.01 \cdot 40kg \cdot 9.8m/s^2 \cdot \cos 1^{\text{º}} = 10.76N \tag{3.3.5}$$

$$M_{in} = \frac{0.1524m}{2} \cdot \frac{10.76N}{0.8} = 1.025N \cdot m \tag{3.3.6}$$

The input torque must be split on the number of traction wheels in the robot, in this case, four wheels. The resulting torque on each wheel is expressed in Eq. 3.3.7.

$$M_{Wheel} = \frac{M_{in}}{n} = \frac{1.025}{4} = 0.256N \cdot m \tag{3.3.7}$$

Until now, the torque when the nominal velocity is reached has been calculated. In order to calculate the acceleration torque, the average and the maximum torque, the velocity profile of Figure 3.3.1 has been created. This profile has been created according to the distance between the doors when the robot would drive at a nominal velocity.

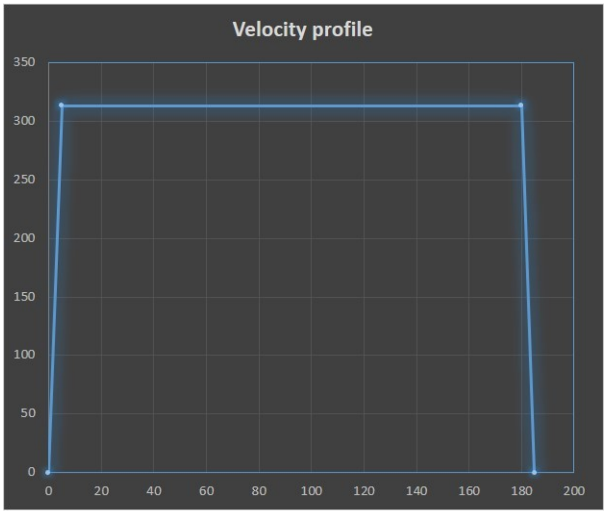


FIGURE 3.3.1. Velocity profile of the robotic platform

Thanks to this velocity profile, where it takes 5 seconds to reach de maximum velocity (long time to avoid the abrupt movements), the acceleration has been set in Eq. 3.3.8.

$$a = \frac{V_{max}}{t} = \frac{2m/s}{5s} = 0.4m/s^2 \tag{3.3.8}$$

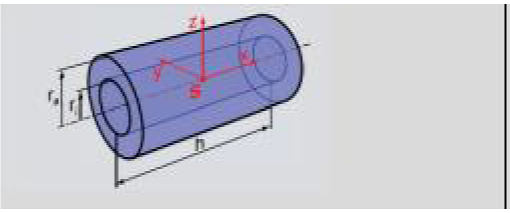
With all these calculations, it is possible to calculate the necessary power of the motor, following Eq. 3.3.9, resolved in Eq. 3.3.10. In both equations, a confidence factor of 0.8 has been set. Of course, the chosen motors must have more power than the calculated. For this calculation, a power factor of 1.1 has been applied to ensure the good performance of the motors.

$$Power = PF \cdot \frac{m \cdot V_L \cdot (a + g \cdot \sin\theta)}{2 \cdot \Pi \cdot \kappa} \tag{3.3.9}$$

$$Power = 1.1 \cdot \frac{40 \cdot 2 \cdot (0.4 + 9.8 \cdot \sin 1^{º})}{2 \cdot \Pi \cdot 0.8} = 10W \tag{3.3.10}$$

3.3.2. Maximum force, torque, and power.

One of the most important parts in the selection of a motor is found in the inertial forces. With the first selection of the motor, it is possible to know its inertial moment. Since the inertial moment of the wheels is well know, the inertial moment of the set can be calculated. The inertial moment of the wheels has been calculated through the equations shown in Figure 3.3.2 [**13**]. For the case of a wheel, the important axis would be the 'x' one.



$$m = \rho \cdot \pi \cdot (r_a^2 - r_i^2) \cdot h$$

$$J_x = \frac{1}{2} \cdot m \cdot (r_a^2 + r_i^2)$$

$$J_y = J_z = \frac{1}{4} \cdot m \cdot \left(r_a^2 + r_i^2 + \frac{h^2}{3}\right)$$

Figure 3.3.2. Inertial moments of a hollow cylinder

Thanks to it (value obtain in Eq. 3.3.11 and 3.3.12), it is possible to calculate the acceleration torque for the whole robot, which is shown in Eq. 3.3.13 and solved in Eq. 3.3.14, where the inertial moment of the motor is supposed to be the inertial moment of a motor that satisfies the previous specifications.

$$m = \frac{2.38kg(mass)}{0.01363m^3(volume)} \cdot \pi \cdot ((0.0762m^2) - (0.01m^2)) \cdot 0.076m = 0.24kg \tag{3.3.11}$$

$$J_x = \frac{1}{2} \cdot 0.24 \cdot ((0.0762m)^2 + (0.01m^2)) = 70260g \cdot cm^2 \tag{3.3.12}$$

$$M_{in,\alpha} = (n \cdot J_{in} + n \cdot J_w + \frac{m_L + m_F}{\eta} \cdot \frac{d^2}{4}) \cdot \frac{\Pi}{30} \cdot \frac{\Delta n_{in}}{\Delta t_a} \cdot PF \tag{3.3.13}$$

(3.3.14)

$$M_{in,\alpha} = (4 \cdot 0.007026kg \cdot m^2 + 4 \cdot 14.7 \cdot 10^{-7} kg \cdot m^2 + \frac{40kg}{0.8} \cdot \frac{(0.1524m)^2}{4}) \cdot \frac{\Pi}{30} \cdot \frac{313.2rpm}{5s} \cdot 1.1 = 2.3N \cdot m$$

Then, knowing the torque during the nominal moment (Eq. 3.3.6) and during the acceleration (Eq. 3.3.14), it is possible to calculate the maximum and RMS torque in each wheel, according to Eq. 3.3.15 and 3.3.17, solved in Eq. 3.3.16 and 3.3.18 respectively.

$$M_{max/wheel} = \frac{M_{in} + M_{in,\alpha}}{n_{wheels}} \tag{3.3.15}$$

(3.3.16)
$$M_{max/wheel} = \frac{1.025Nm + 2.3Nm}{4} = 0.83N \cdot m$$

(3.3.17)
$$M_{RMS/wheel} = \sqrt{\frac{1}{t_{total}} \cdot (t_1 \cdot M_1^2 + t_2 \cdot M_2^2 + t_3 \cdot M_3^2 + t_4 \cdot M_4^2}$$

(3.3.18)
$$M_{RMS/wheel} = \sqrt{\frac{1}{185s} \cdot (5s \cdot (0.83Nm)^2 + 175 \cdot (0.256Nm)^2 + 5s \cdot (0.83Nm - 0.256Nm)^2} = 0.3Nm$$

At this moment, the maximum and RMS power can be calculated according to Eq. 3.3.19 and 3.3.21, and they have been solved in Eq. 3.3.20 and 3.3.22.

(3.3.19)
$$P_{max} = M_{max} \cdot n_{max} \cdot \frac{\Pi}{30}$$

(3.3.20)
$$P_{max} = 0.83Nm \cdot 313.2rpm \cdot \frac{\Pi}{30} = 27.28W$$

(3.3.21)
$$P_{RMS} = M_{RMS} \cdot n_{RMS} \cdot \frac{\Pi}{30}$$

(3.3.22)
$$P_{RMS} = 0.3Nm \cdot 313.2rpm \cdot \frac{\Pi}{30} = 9.92W$$

3.3.3. Motor and gearhead selection.

Thanks to the calculations made previously, it is possible to do a proper selection of the gearhead according to the instructions detailed in the following text box:

Result

A reducer with torque in continuous performance at least of 0.3Nm is necessary, and an intermittent torque at least of 0.83Nm.

The necessary reduction is calculated in Eq. 3.3.23, so the chosen reduction has to be smaller.

(3.3.23)
$$N = \frac{n_{motor}}{n_{wheel}} = \frac{9690rpm}{313.23rpm} = 30.94$$

The chosen reducer has been the model **GP 26A** of **Maxon**, with a reduction of **1:27** and fulfilling all the specification of the text box (more information in Appendix D.1).

Since the output velocity is higher than required and since it is necessary motion transmission between the motor system and the wheels, a couple of pulleys have been chosen (with the corresponding belt). The necessary relationship to get 2m/s is 1:1.15, however, a **1:1.1** relation has been selected.

In this way, the motor must get the velocity indicated in Eq. 3.3.24, with a continuous and intermittent torque detailed in Eq. 3.3.25 and Eq. 3.3.26 respectively.

$$n_{motor} = n_{wheel} \cdot r = 313.22rpm \cdot 27 \cdot 1.1 = 9303rpm \tag{3.3.24}$$

$$M_{RMS} = \frac{M_{RMS/Wheel}}{r \cdot \eta} = \frac{0.3Nm}{27 \cdot 1.1 \cdot 0.8} = 0.0126Nm \tag{3.3.25}$$

$$M_{max} = \frac{M_{max/Wheel}}{r \cdot \eta} = \frac{0.83Nm}{27 \cdot 1.1 \cdot 0.8} = 0.035Nm \tag{3.3.26}$$

In summary, to select the motor:

Result

With the chosen reducer, with a reduction number (r) and a performance (η), the motor must apply at least of 9303rpm, with a minimum torque of 0.0126Nm in continuous operation and at least 0.035Nm in intermittent operation, with a power higher than 9.92W (checking the ability to withstand 27.28W in specific moments).

The chosen motor has been the model **RE 25, Graphite Brushes, 20W** of **24V** (more information in Appendix D.2).

Finally, to ensure the good performance of the chosen motor, the speed constant of the motor must be higher than the calculated in Eq. 3.3.27.

$$K = \frac{n_{empty}}{V_{supply}} = \frac{n_{motor+gradient \cdot M_{max}}}{V_{supply}} = \frac{8457rpm + 34000\frac{rpm}{Nm} \cdot 0.04Nm}{24V} = 407\frac{rpm}{V} \tag{3.3.27}$$

The motor must have a speed constant higher than 407rpm/V, in this way it is possible to compensate with the controller the desired velocity, being able to reach higher velocities (the motor speed constant is 460rpm/V, this is, higher). Additionally, the chosen encoder has been the **Encoder MR Type ML** which allows **500 counts per turn** (shown in Appendix D.3).

3.3.4. Motor verification.

In order to guarantee the proper performance of the chosen motor, an experiment has been carried out. This experiment has been approached according to the methodologies explained in [**14**]. The case of study is a cycle of motor performance, from null velocity to maximum velocity, keeping this velocity in a nominal state.

For this study, the tool Simulink of Matlab has been used. In this way, the blocking scheme of Figure 3.3.3 has been developed, where a PID controller has been included in the motor model (defined by its features such as the terminal resistance, terminal inductance, torque constant, inertial moment and viscous constant).

The results for the desired speed of 1.5m/s are shown in Figure 3.3.4, where it is possible to observe different graphs with interesting data.

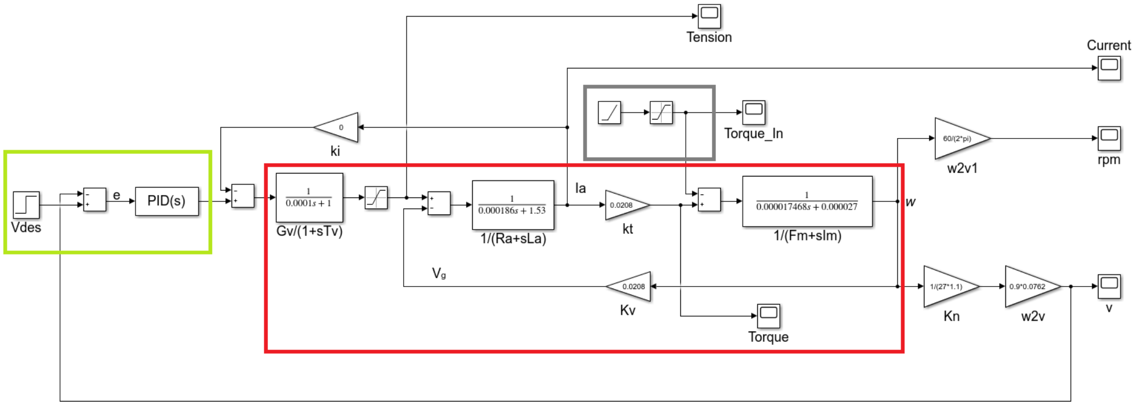
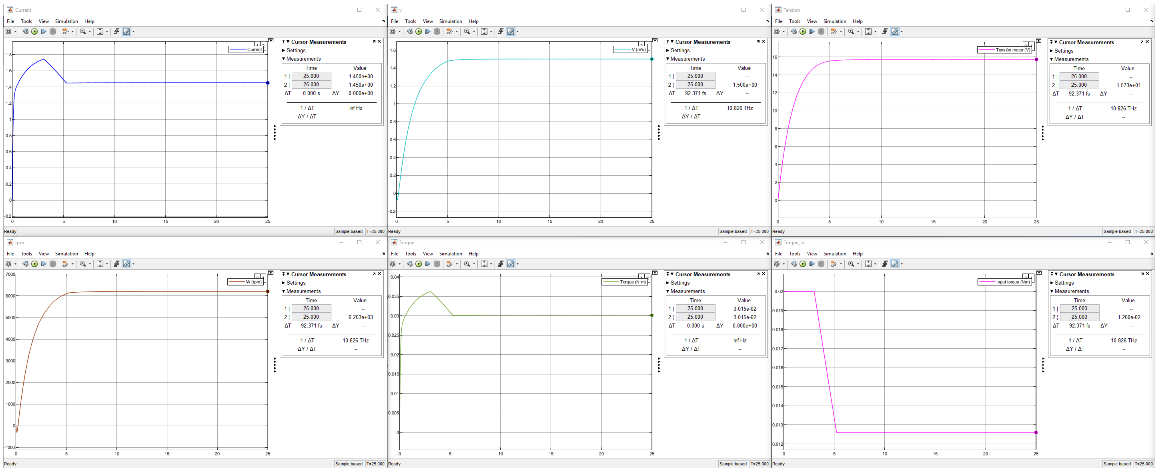


Figure 3.3.3. Block diagram for the motor verification.
Green square: Velocity controller of the robot. Red square: Motor model. Gray square: Input torque due to the necessity of motion.



Cursor Measurements
▸ Settings
▾ Measurements

	Time		Value
1	25.000		1.450e+00
2	25.000		1.450e+00
ΔT	0.000 s	ΔY	0.000e+00
	1 / ΔT		Inf Hz
	ΔY / ΔT		--

Cursor Measurements
▸ Settings
▾ Measurements

	Time		Value
1	25.000		--
2	25.000		1.500e+00
ΔT	92.371 fs	ΔY	--
	1 / ΔT		10.826 THz
	ΔY / ΔT		--

Cursor Measurements
▸ Settings
▾ Measurements

	Time		Value
1	25.000		--
2	25.000		1.573e+01
ΔT	92.371 fs	ΔY	--
	1 / ΔT		10.826 THz
	ΔY / ΔT		--

Cursor Measurements
▸ Settings
▾ Measurements

	Time		Value
1	25.000		--
2	25.000		6.203e+03
ΔT	92.371 fs	ΔY	--
	1 / ΔT		10.826 THz
	ΔY / ΔT		--

Cursor Measurements
▸ Settings
▾ Measurements

	Time		Value
1	25.000		3.015e-02
2	25.000		3.015e-02
ΔT	0.000 s	ΔY	0.000e+00
	1 / ΔT		Inf Hz
	ΔY / ΔT		--

Cursor Measurements
▸ Settings
▾ Measurements

	Time		Value
1	25.000		--
2	25.000		1.260e-02
ΔT	92.371 fs	ΔY	--
	1 / ΔT		10.826 THz
	ΔY / ΔT		--

FIGURE 3.3.4. Results for a desired speed of 1.5m/s.
The following graphs are shown: 1.a. Current in the terminals (less than the nominal current in the nominal state). 1.b Speed of the motor in rpm (less than the nominal velocity of the motor). 2.a. Platform velocity (exactly the desired). 2.b. Torque applied by the motor (less than the nominal torque). 3.a. Voltage in the motor (less than the nominal voltage). 3.b. Input torque (a simulation of the input torque when the robot is accelerating).

Thanks to this study, it is possible to verify the following facts:

- The current during the desired speed is not higher than the nominal current.
- The current does not overcome the maximum current.
- The torque during the desired speed is not higher than the nominal torque.
- The torque does not overcome the maximum torque.
- The PID controller (it will be implemented externally) allows the robot to achieve the desired velocity in 5 seconds, as calculated.

3.4. Devices selection

Between the device to be selected, it is possible to mention:

- Two 2D lidars for object detection. In this case, the model **UST-20LX** of **Hokuyo** has been selected because of its features:
  - Range from 0.06m to 60m (high to detect the obstacle with enough time).
  - 0.25º of angular resolution (small to detect every type of obstacle).
  - A scan angle of 270º (more than required for the front and back view).
- Cameras for teleoperation (or for visual odometry). In this case, the model **UI-3080CP Rev. 2** of **IDS** has been selected. The camera carries a lens **LM5JC10M** of **Kowa** to increase the field of view, so, in this way, it is possible to extract more

features for the images and improve the results of the visual odometry. The main characteristics of the camera are:
  - Resolution of 5MPix.
  - Global shutter to avoid bad distortion in the picture taking.
  - A frame-rate of 77fps (high to reduce shaking).
- A tracking camera for visual odometry. In this case, the model **R265** of **Intel** has been selected. This camera can be replaced by the cameras for teleoperation (and the needed software).
- A point-cloud generator sensor. In this case, we have two options (the second one has been implemented in this project):
  - The 3D lidar Velodyne **HDL-32E**, which has a 360x40º field of view and 2cm of accuracy. The range is up to100m.
  - The RGBd camera **D415** of **Intel**, which has a 65x40x72º (HxVxD) field of view and 10m of range.
- The IMU **VMU931** with a sensing range of 2000º/s.
- The batteries **NP7-12**, the most efficient of the seller **Yuasa** and suitable for a future design.
- Radiation sensor (internal component).
- 4G module for communication and Wi-Fi antenna (internal components).

3.5. Mechanical design

The mechanical design has been approached using aluminum profiles for the main structure, giving consistency and strength to the robot body. Furthermore, the bottom plates have been chosen of aluminum material, since this material is proper for a radioactive environment (information obtained from ActiWiz application).

3.5.1. Preliminary design.

In the first place, a mechanical design with two stabilizer wheels has been approached (Figure 3.5.1). This design has the following components:

- An aluminum structure (Figure 3.5.2a) with the bottom plate to fix the components.
- Four traction set (Figure 3.5.2b), composed by:
  - A mecanum wheel to convert the robot to omnidirectional.
  - A motor.
  - A reducer.
  - An encoder.
  - System support for the motor.
  - A shaft to move the wheel.
  - Two bearing to support the shaft.

  - A pulley set, composed of the transmitter and transmitted pulleys and by a belt.
- A stabilizer set (Figure 3.5.2c) to ensure stability. It is composed by:
  - Two 90º mecanum wheels. With this kind of wheels, if the robot goes forward the wheels turn normally, while in case the robot goes laterally, the rollers allow the movement without friction.
  - Two shafts with bearings that allow the wheels to move freely.
  - Two linear guides.
  - Two supports for the linear guides
  - Two supports to couple the shaft and the linear guide.
  - A stepper motor for the motion (with a controller externally).
  - Support for the motor.
  - Gear for the motor.
  - Two linear gears coupled to the motor gear and linked to the linear guide.
- Four battery packs.

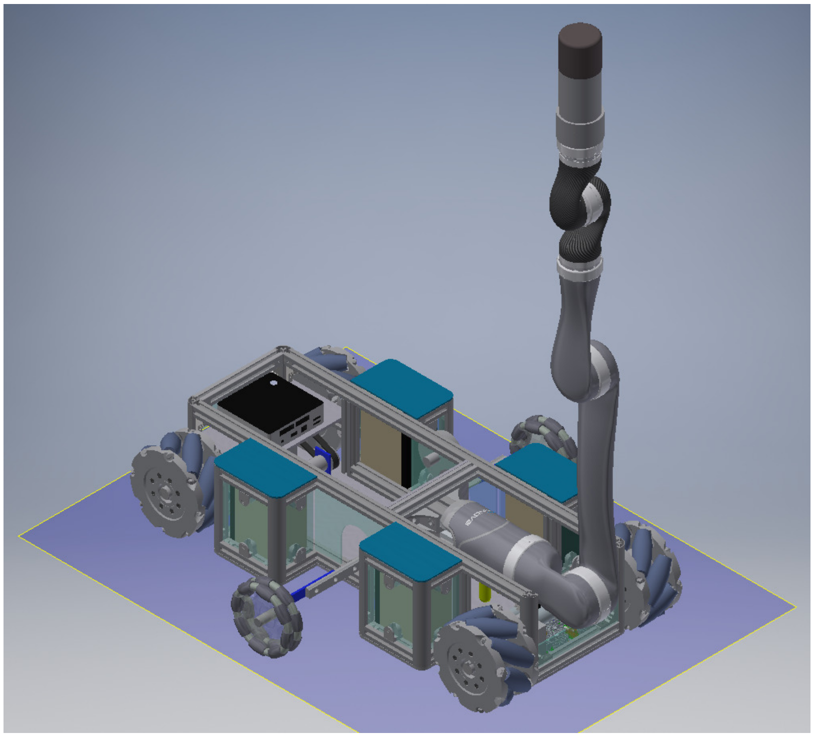
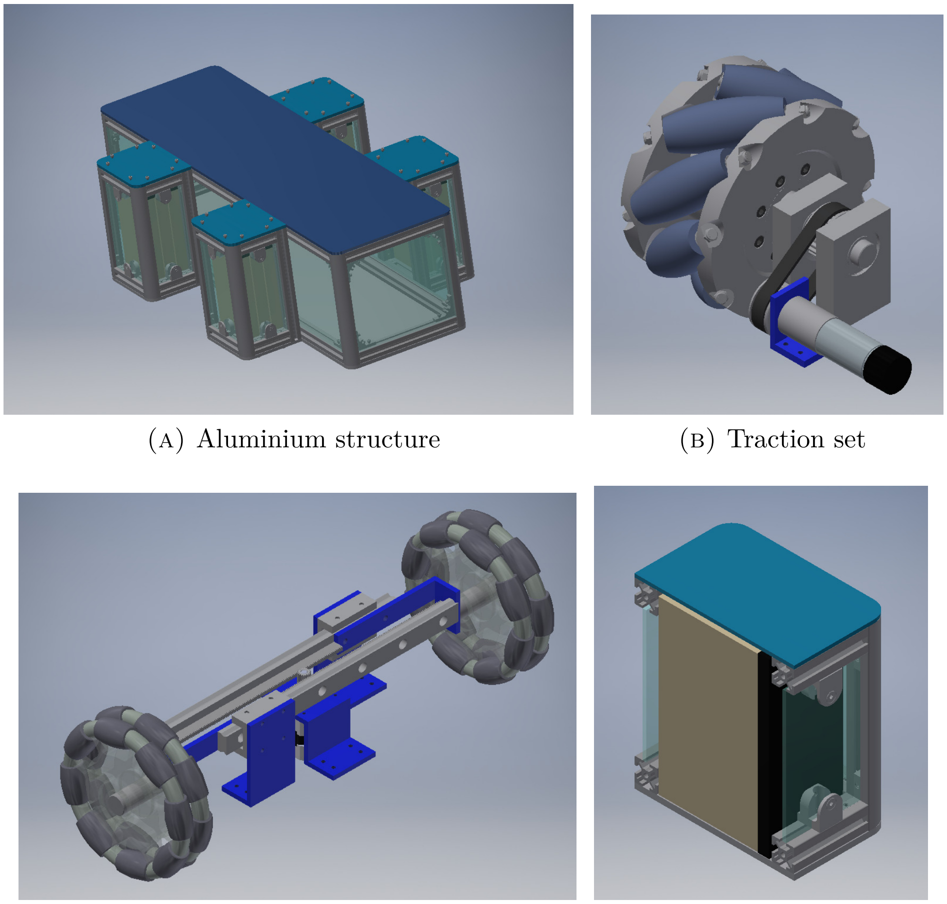


FIGURE 3.5.1. The entire system of the preliminary design

The main characteristic of this design is the possibility to extract the telescopic system to improve the stability of the robot when it is driving at the maximum velocity (Figure 3.5.1). However, the width in that state is bigger than required to cross the doors. To solve this



(A) Aluminium structure

(B) Traction set

(C) Stabilizer set

(D) Battery package

Figure 3.5.2. Components of the preliminary design of the robotic platform. All the pieces shown in dark blue have to be manufactured in aluminum material.

problem, the telescopic system can shrink and the robotic arm can be placed horizontally, like Figure 3.5.3 shows.

In all the cases, the dimensions of the robot can be observed in Figure 3.5.4.

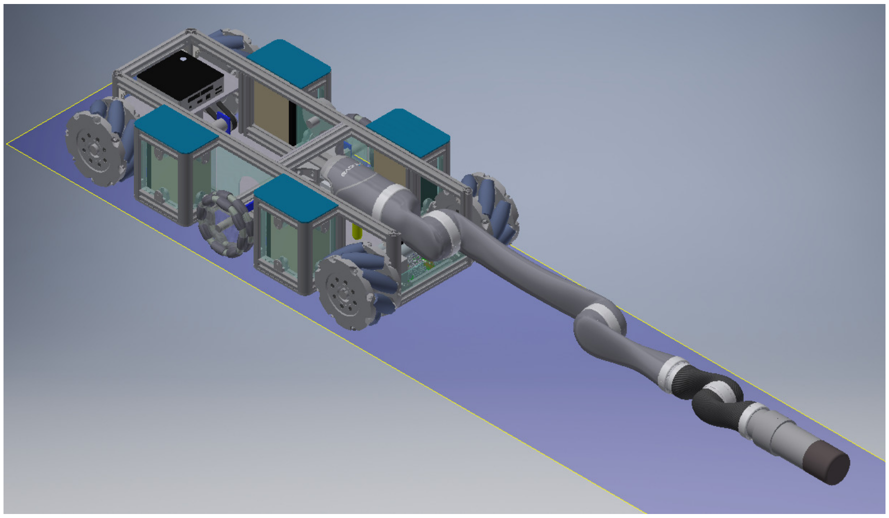
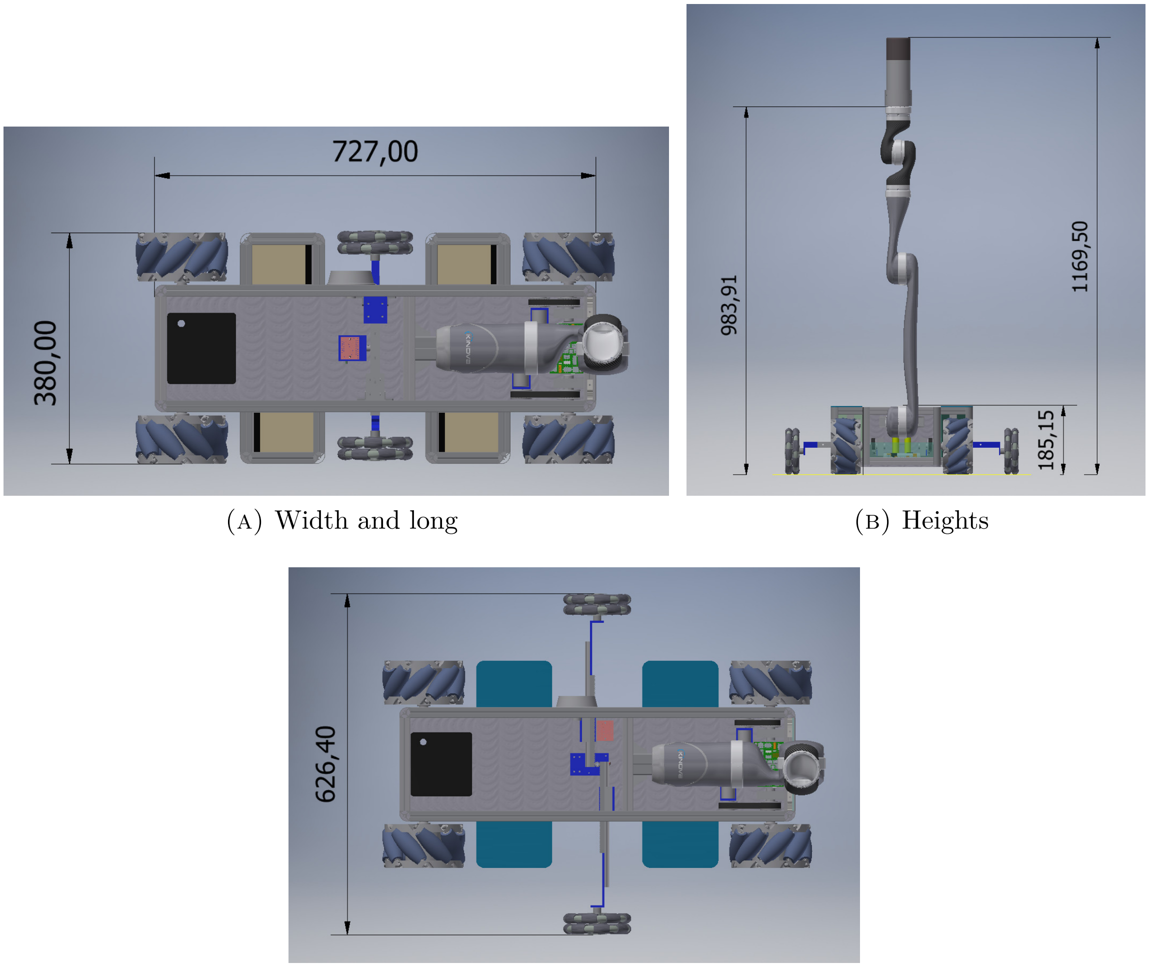


Figure 3.5.3. The robotic platform crossing the door



(a) Width and long

(b) Heights

(c) Stability system width

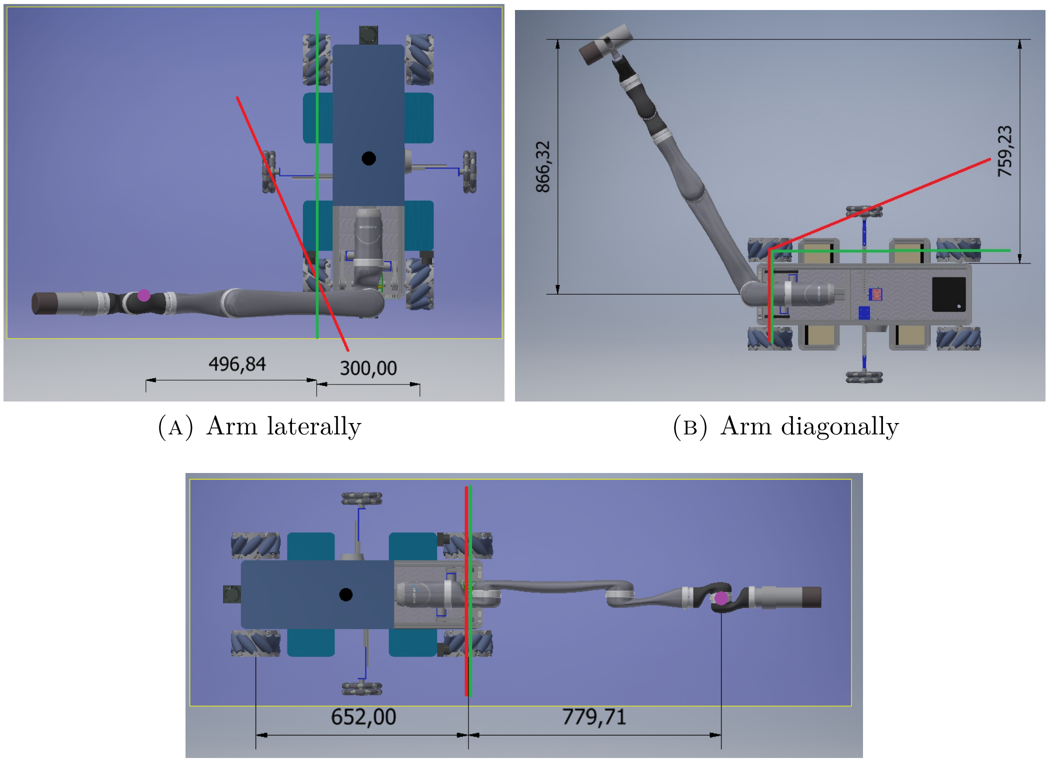
Figure 3.5.4. Dimensions of the robotic platform for the preliminary design

A study should be carried out from the stability point of view, in order to know if the telescopic wheels are useful or can be suppressed.

3.5.2. Stability study.

With the preliminary design, a stability study has been carried out in different situations:

- Arm laterally (Figure 3.5.5a).
- Arm diagonally (Figure 3.5.5b).
- Arm frontally (Figure 3.5.5c).



(A) Arm laterally

(B) Arm diagonally

(C) Arm frontally

FIGURE 3.5.5. Situations of the stability study

In all cases, the c.o.g. of the platform (black) and robotic arm (purple) are indicated, so the calculations use these distances.

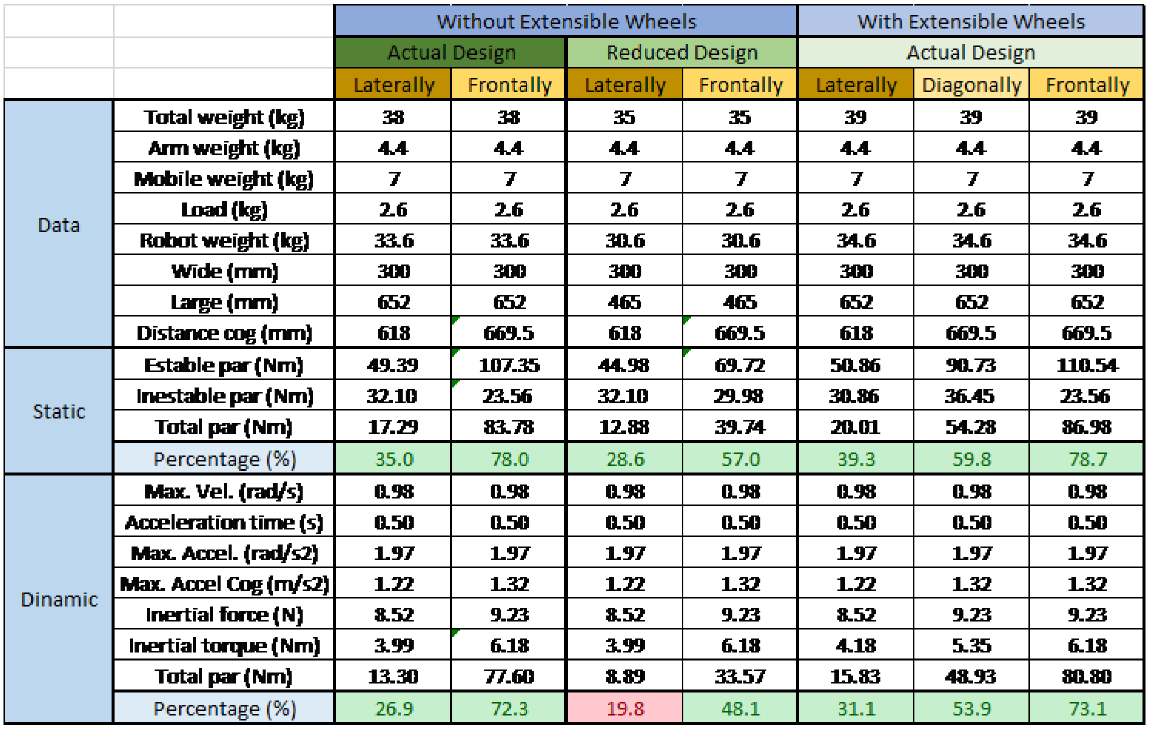
This study presents three different cases:

(1) The previous design, this is, explained in Section 3.5.1. In this case, the critical tipping line is shown in red color in Figure 3.5.5.
(2) The previous design eliminating the extensible wheels. In this case, the critical tipping line is shown in green color (fictitious).

(3) A compact design eliminating the extensible wheels. In this case, the critical tipping line is the green line as well (fictitious line).

Since there is an estimation of the robotic platform weight, a stability study is presented in Table 3.5.1, understanding stability as the ability to avoid overturning. This study contemplates both the static case, where the robotic arm is still, both the dynamic case, when the robotic arm is at its most critical point, moving from zero speed to maximum speed or vice versa.

Furthermore, the study has been developed considering the maximum load in the robotic arm in the worst configuration (the robot is not expected to bring the maximum load either the worst configuration). As it is possible to see in Table 3.5.1, the only critical case is when the robotic arm is set laterally with a reduced design (without the extensible wheels).



		Without Extensible Wheels				With Extensible Wheels		
		Actual Design		Reduced Design		Actual Design		
		Laterally	Frontally	Laterally	Frontally	Laterally	Diagonally	Frontally
Data	Total weight (kg)	38	38	35	35	39	39	39
	Arm weight (kg)	4.4	4.4	4.4	4.4	4.4	4.4	4.4
	Mobile weight (kg)	7	7	7	7	7	7	7
	Load (kg)	2.6	2.6	2.6	2.6	2.6	2.6	2.6
	Robot weight (kg)	33.6	33.6	30.6	30.6	34.6	34.6	34.6
	Wide (mm)	300	300	300	300	300	300	300
	Large (mm)	652	652	465	465	652	652	652
	Distance cog (mm)	618	669.5	618	669.5	618	669.5	669.5
Static	Estable par (Nm)	49.39	107.35	44.98	69.72	50.86	90.73	110.54
	Inestable par (Nm)	32.10	23.56	32.10	29.98	30.86	36.45	23.56
	Total par (Nm)	17.29	83.78	12.88	39.74	20.01	54.28	86.98
	Percentage (%)	35.0	78.0	28.6	57.0	39.3	59.8	78.7
Dinamic	Max. Vel. (rad/s)	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	Acceleration time (s)	0.50	0.50	0.50	0.50	0.50	0.50	0.50
	Max. Accel. (rad/s2)	1.97	1.97	1.97	1.97	1.97	1.97	1.97
	Max. Accel Cog (m/s2)	1.22	1.32	1.22	1.32	1.22	1.32	1.32
	Inertial force (N)	8.52	9.23	8.52	9.23	8.52	9.23	9.23
	Inertial torque (Nm)	3.99	6.18	3.99	6.18	4.18	5.35	6.18
	Total par (Nm)	13.30	77.60	8.89	33.57	15.83	48.93	80.80
	Percentage (%)	26.9	72.3	19.8	48.1	31.1	53.9	73.1

Table 3.5.1. Stability study results with the maximum load

Since in the worst case (maximum load with the more critical configuration of the robotic arm at the maximum velocity) the confidence factor (stable torque divided by tipping or unstable torque) in a reduced design without telescopic wheels is around 20%, it is expected that the robot will never tip over, and the design has to be modified to a new configuration.

3.5.3. Final design.

With the certainty that the robot does not need the extensible wheels, a new more compact design is presented in Figure 3.5.6. This design does not change the traction system, but its size and some characteristics are modified.

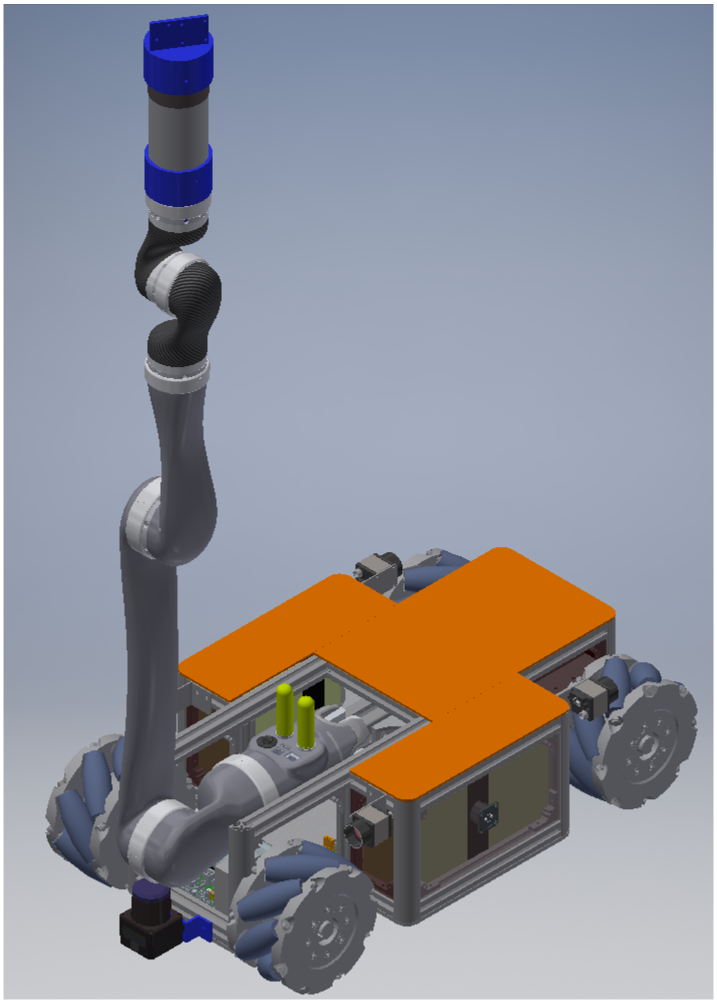


FIGURE 3.5.6. The compact design of the robotic platform for the survey in underground tunnels with space constraints.

The new features are the following:

- A magnetic connector for charge the batteries localized on one side.
- Four possible localization for IDS cameras behind the wheels.
- Two 2D lidars for obstacle and wall detection (one in front and one in the back). Both of them are linked to the structure with a piece that has to be manufactured in aluminum material.
- Possibility to localize a 3D lidar (especially design for Velodyne HDL-32E) behind the plate that supports the robotic arm (this plate can be moved forward if needed).

- A piece to support the radiation sensor in the end effector of the robotic arm.
- A piece to link in the radiation sensor to place a SONAR in order to determine the distance between the sensor and the radiation source.
- Figure 3.5.6 shows the robotic arm in an inverted way to appreciate the necessary space of the connectors (green pieces).

The size of the robot is shown in Figure 3.5.7, where it is possible to appreciate that the main dimensions are 706.6x350x184.22mm, a size that is expected to be perfect to cross the doors.

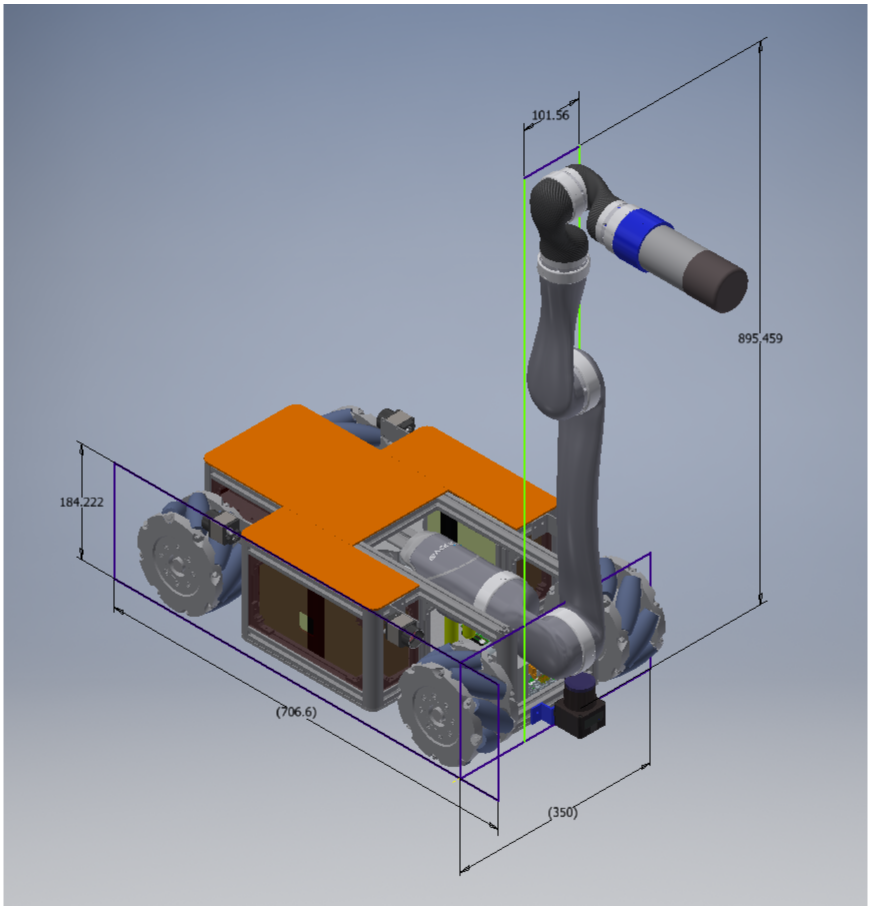
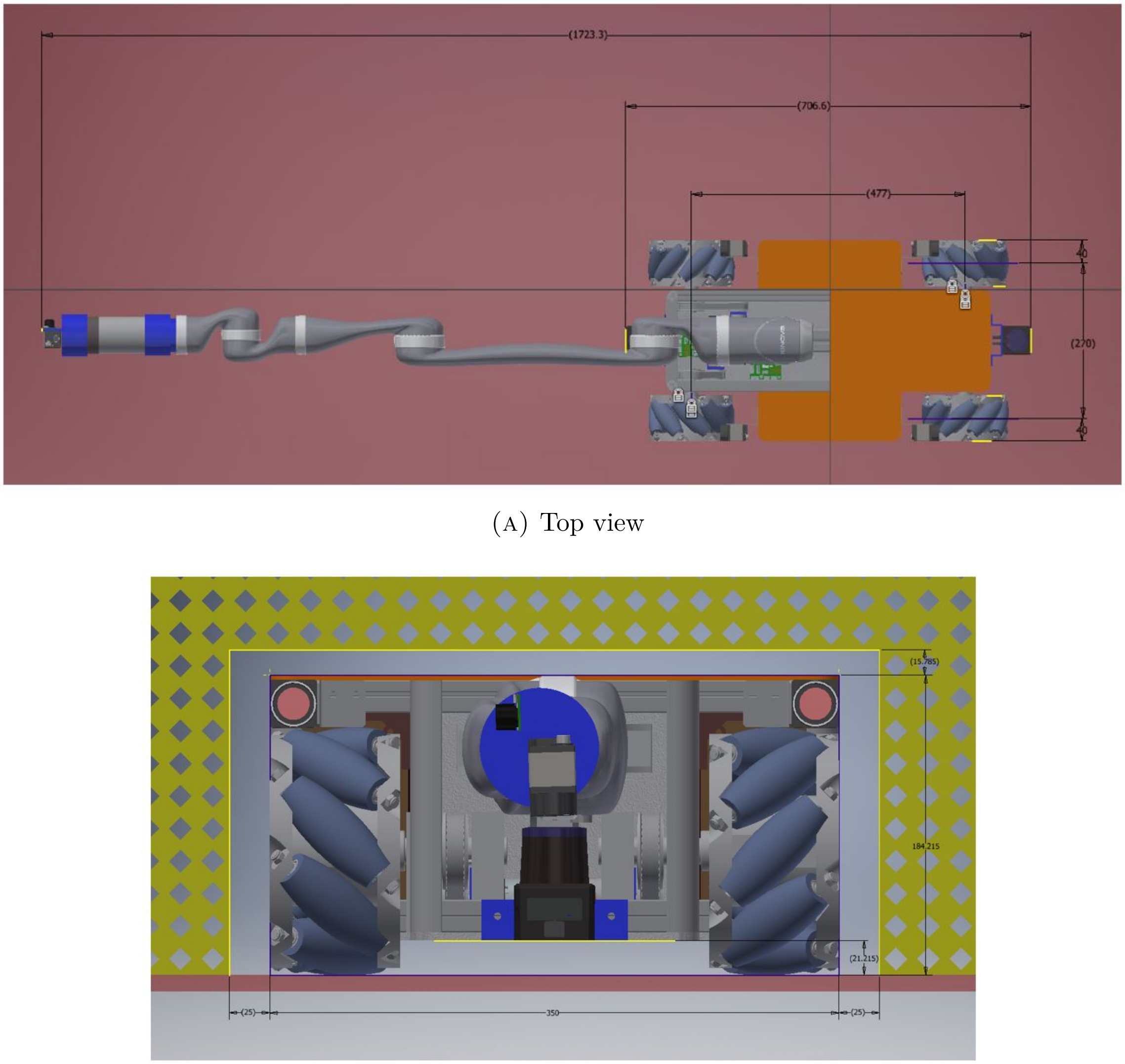


FIGURE 3.5.7. Some dimensions of the compact design of the robotic platform for the survey in tunnels are shown in 'mm'

The expected weight of the robot is around 45kg, taking into account everything. The reach of the sensor is expected to be at 850mm, however, it can reach until 1100mm in its main point.

The robot is expected to adopt the position shown in Figure 3.5.8when it is crossing the door.



(A) Top view

(B) Front view

Figure 3.5.8. Robot posture crossing the doors

As shown in those pictures, a piece has been attached to the radiation sensor. This piece has a SONAR, which is responsible for measure the distance between the sensor and the radioactive source. The connexions between the control system and the SONAR are carried out thanks to the internal connexions in the robotic arm.

In those pictures is also possible to appreciate the distance when crossing the door, concluding that there is enough space in that situation.

3.6. Electronic and electrical design

In this section, electronic and electrical design is shown. Since the motor, the gearhead, the encoder, and the controller have been selected from Maxon Motor seller, the cables have been chosen as well. The following guidelines have been taken:

- The power supply has been taken from the batteries.
- The communication between the computer and the controller is through CAN bus[8].
- The controller must control the motor position, taking the information from the encoder.

An electronic and electrical scheme to control one motor and determinate its position is shown in Figure 3.6.1, where the cables identifier is indicated (cable identifier of Maxon Motor).

Following the simple scheme, a complex scheme of the four motors has been developed and shown in Figure 3.6.2.

Summary

A robotic system for remote inspection and environmental measurements is proposed to be implemented in the SPS to reduce personnel exposure to hazards and machine downtime. In particular, the robotic system will be used to survey the radiation levels in the tunnel during a planned technical stop or during a repair intervention.
The robot is based on a four-wheeled mobile platform and equipped with several cameras, LIDARs and a radiation sensor. The robot can be monitored remotely using the 4G mobile connection available in the underground tunnels.
The estimated weight of the robot is 45kg. The robot is expected to be equipped with an active collision-avoidance system to cause no damage to the components (magnets, etc..).
The robot will be permanently stored in the SPS, in an area protected from radiation during the beam operation. An automatic charging station will be implemented; the station and charging parameters (charge status, battery temperature, and voltage, etc.) will be monitored through the Ethernet network and IT infrastructure.

[8] CAN bus: Controller Area Network. It is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer.

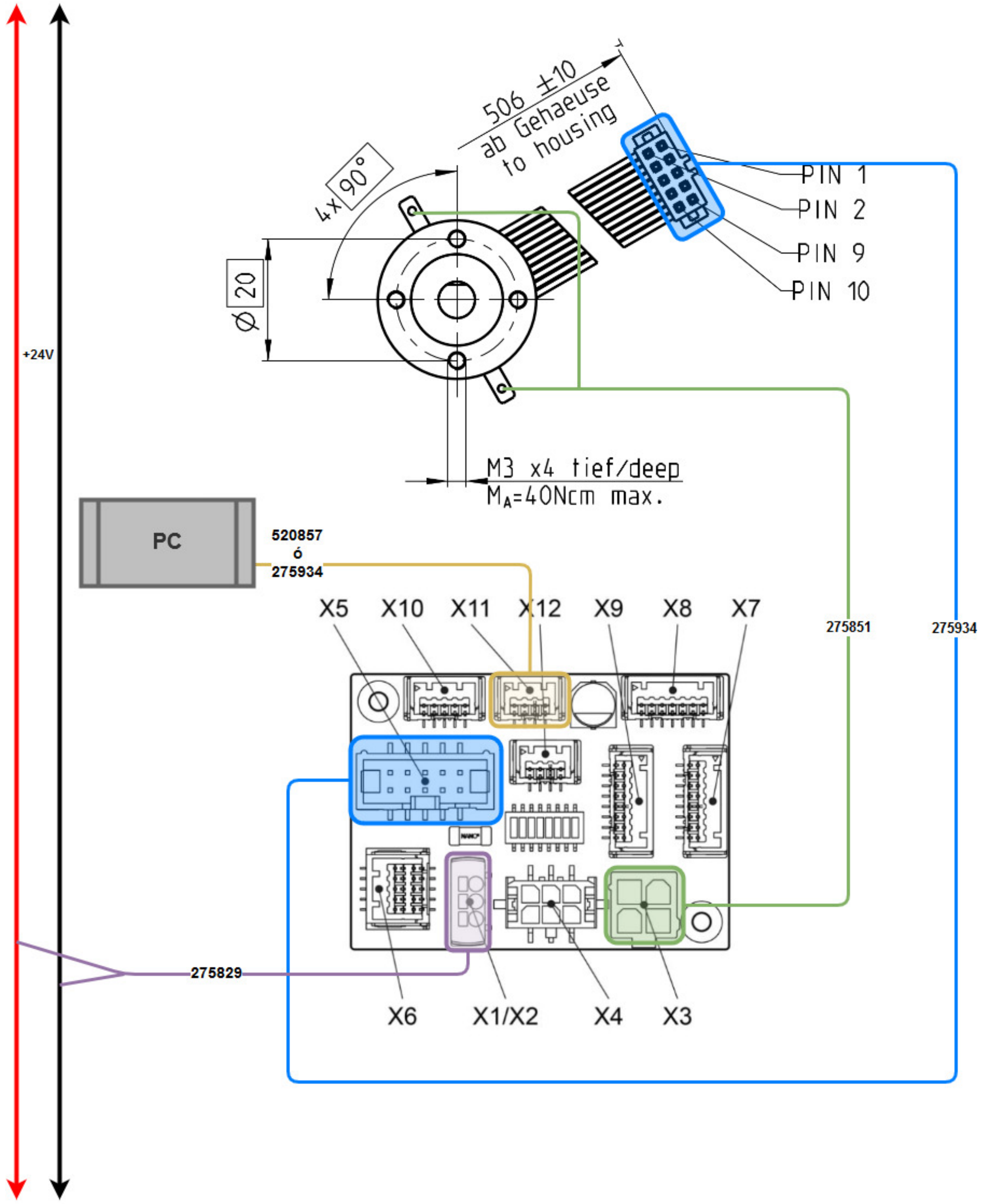


FIGURE 3.6.1. Electronic and electrical scheme to control only one motor

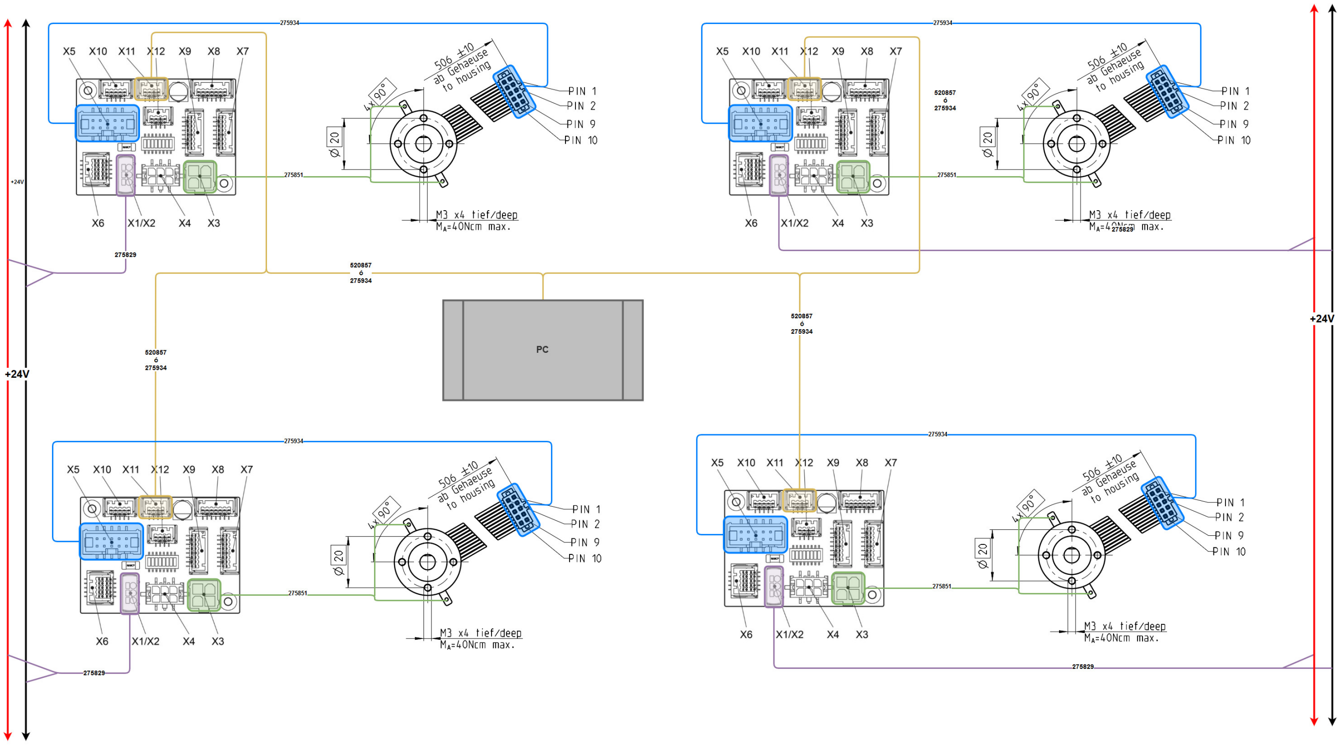


FIGURE 3.6.2. Electronic and electrical scheme to control four motors

CHAPTER 4

Control of the robotic platform

In this chapter, the kinematic and dynamic control of the platform is presented from a theoretical point of view.

4.1. Kinematic calculation

In this section, the process of obtaining the kinematic model has been developed. Firstly, to describe the robot posture in the ground plane, the reference system shown in Figure 4.1.1 has been created. The used sources in this section can be found in [**15, 16, 17**].

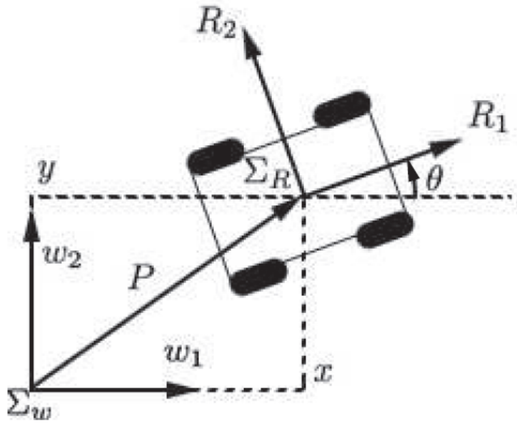


Figure 4.1.1. Reference systems $\sum_W$ and $\sum_R$

These systems are:

- $\sum_W$ denotes the world coordinates system.
- $\sum_R$ denotes the robot coordinates system.

The position of the robot is described by 3 d.o.f., the coordinates ’x’ and ’y’ (relation between the robot and world systems), and the orientation ’θ’. So, the vector ξ which defines the posture of the robot is presented in Eq. 4.1.1.

$$\xi = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \epsilon\, \mathbb{R}^3 \tag{4.1.1}$$

It is possible to define the rotation matrix Eq. 4.1.2 and 4.1.3 between both systems.

$$R_R^w(\theta) = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1.2}$$

$$R_w^R(\theta) = \begin{bmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1.3}$$

Considering the reference systems related to the omnidirectional wheel shown in Figure 4.1.2, β_i is known since the dimensions of the robot are known (α_i is well known), and γ_i is known as well since the chosen wheels are 45º omnidirectional wheels.

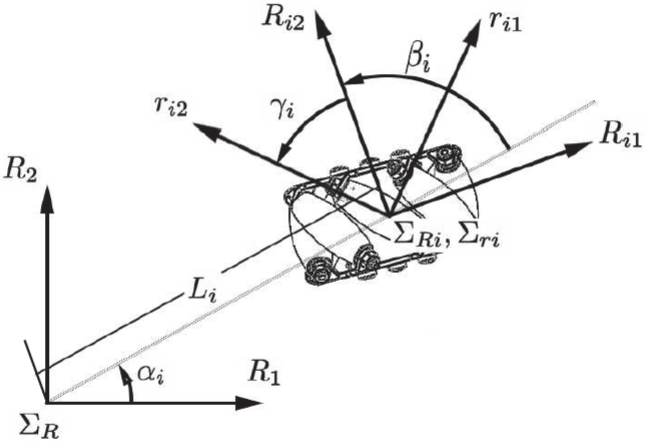


Figure 4.1.2. Reference systems related to the used omnidirectional wheel

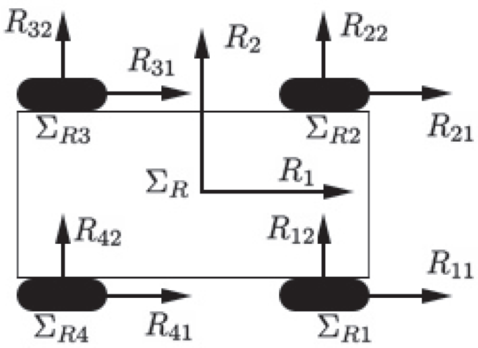


Figure 4.1.3. Reference systems related to the robot and the wheels

In that figure, the reference systems related to the wheel and related to the rollers appear, $\sum_{R_i}$ and $\sum_{r_i}$ respectively, where the R_{i1} is situated in the forward direction of the wheel, while r_{i1} in the forward direction of the rollers. The set of reference systems is shown in Figure 4.1.3, where the robot frame ($\sum_R$) and the wheel frames ($\sum_{R1}$, $\sum_{R2}$, $\sum_{R3}$, $\sum_{R4}$) are shown.

As before, it is possible to define the rotation matrix Eq. 4.1.4 between both reference systems.

$$R_R^{R_i}(\phi_i) = \begin{bmatrix} c_{\phi_i} & s_{\phi_i} & 0 \\ -s_{\phi_i} & c_{\phi_i} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \phi_i = \alpha_i + \beta_i + \gamma_i \tag{4.1.4}$$

Since the posture of the robot has been referred to $\sum_W$, the velocity vector η referred to $\sum_R$ is obtained in Eq. 4.1.5.

$$\eta = R_w^R(\theta) \cdot \dot{\xi} = R_w^R(\theta) \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{4.1.5}$$

The needed velocity in the wheel i in order to get η is calculated in Eq. 4.1.6, obtained from [**18**], where the velocities are referred to $\sum_R$.

$$\begin{bmatrix} \dot{O}_{R_{i1}} \\ \dot{O}_{R_{i2}} \\ 0 \end{bmatrix}^R = \begin{bmatrix} \dot{x}^R - L_i \cdot s_{\alpha_i} . \dot{\theta} \\ \dot{y}^R - L_i \cdot c_{\alpha_i} \, . \, \dot{\theta} \\ 0 \end{bmatrix} \tag{4.1.6}$$

Since the motion generator is the rollers, Eq. 4.1.7 transform the needed velocity to $\sum_{r_i}$ thanks to Eq. 4.1.4.

$$\begin{bmatrix} \dot{O}_{R_{i1}} \\ \dot{O}_{R_{i2}} \\ 0 \end{bmatrix}^{R_i} = R_R^{R_i}(\phi_i) \cdot \begin{bmatrix} \dot{O}_{R_{i1}} \\ \dot{O}_{R_{i2}} \\ 0 \end{bmatrix}^{R} \tag{4.1.7}$$

Since $\dot{\varphi}_{R_i}$ is the wheel speed, r_{R_i} is the wheel radius i, $\dot{\varphi}_{r_i}$ is the roller speed, r_{r_i} is the roller radius. The delivered speed by the wheel is $r_{R_i}\dot{\varphi}_{R_i}$ in R_{i1} direction and the roller lineal velocity is $r_{r_i}\dot{\varphi}_{r_i}$ in r_{i1} direction. So, passing the speed delivered by the wheel to $\sum_{r_i}$ and using Eq. 4.1.7, Eq. 4.1.8 is obtained.

$$\begin{bmatrix} r_{R_i}\dot{\varphi}_{R_i}c_{\gamma_i} \\ r_{r_i}\dot{\varphi}_{r_i} + r_{R_i}\dot{\varphi}_{R_i}s_{\gamma_i} \\ 0 \end{bmatrix} = \begin{bmatrix} s_{\phi_i} & -c_{\phi_i} & L_i c_{\delta_i} \\ c_{\phi_i} & s_{\phi_i} & L_i c_{\delta_i} \\ 0 & 0 & 0 \end{bmatrix} \eta,\ \delta_i = \beta_i + \gamma_i \tag{4.1.8}$$

Motion constraint (Eq. 4.1.9) is obtained from the first row of the equation system shown in Eq. 4.1.8.

$$\begin{bmatrix} -s_{\phi_i} & c_{\phi_i} & L_i \cdot c_{\delta_i} \end{bmatrix} \cdot R_w^R(\theta) \cdot \dot{\xi} + r_{R_i} \cdot \dot{\varphi}_{R_i} \cdot c_{\gamma_i} = 0 \tag{4.1.9}$$

In our case, the robot values are shown in Table 4.1.1.

Rueda i	L_i	α_i	β_i	γ_i	r_{R_i}
1	L	$360-\alpha$	$\frac{\pi}{2}-\alpha_1$	$\frac{\pi}{4}$	r
2	L	α	$\frac{\pi}{2}-\alpha_2$	$-\frac{\pi}{4}$	r
3	L	$180-\alpha$	$\frac{\pi}{2}-\alpha_3$	$\frac{\pi}{4}$	r
4	L	$180+\alpha$	$\frac{\pi}{2}-\alpha_4$	$-\frac{\pi}{4}$	r

L	274.06mm
α	29.5º
r	76.2mm

TABLE 4.1.1. Kinematic parameters of our robot (recommended to take a look at Figure 4.1.4). These parameters can be observed in Figure 3.5.8.

As before, the motion constraints equation of a robot composed of omnidirectional wheels is shown in Eq. 4.1.10 (with Eq. 4.1.11 and 4.1.12). Substituting values, the results are shown in Eq. 4.1.13 and 4.1.14.

$$J_1 \cdot R_w^R(\theta) \cdot \dot{\xi} + J_2 \cdot \dot{\varphi}_i = 0 \tag{4.1.10}$$

$$J_1 = \begin{bmatrix} -s_{\phi_1} & c_{\phi_1} & L_1 \cdot c_{\delta_1} \\ -s_{\phi_2} & c_{\phi_2} & L_2 \cdot c_{\delta_2} \\ -s_{\phi_3} & c_{\phi_3} & L_3 \cdot c_{\delta_3} \\ -s_{\phi_4} & c_{\phi_4} & L_4 \cdot c_{\delta_4} \end{bmatrix} \tag{4.1.11}$$

$$J_2 = \begin{bmatrix} r_{R_1}.c_{\gamma_1} & 0 & 0 & 0 \\ 0 & r_{R_2} \cdot c_{\gamma_2} & 0 & 0 \\ 0 & 0 & r_{R_3} \cdot c_{\gamma_3} & 0 \\ 0 & 0 & 0 & r_{R_4} \cdot c_{\gamma_4} \end{bmatrix} \tag{4.1.12}$$

$$J_1 = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 & -1 & L_{s_\alpha} - L_{c_\alpha} \\ -1 & 1 & L_{s_\alpha} + L_{c_\alpha} \\ -1 & -1 & L_{s_\alpha} - L_{c_\alpha} \\ -1 & 1 & L_{s_\alpha} + L_{c_\alpha} \end{bmatrix} \tag{4.1.13}$$

$$J_2 = \frac{\sqrt{2}}{2} \cdot r \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1.14}$$

More easily, it is possible to describe Eq. 4.1.10 according to Figure 4.1.4, where the distances are well known. In this case, the results are detailed in Eq. 4.1.15.

$$J_1 = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 & -1 & -l_1 - l_2 \\ -1 & 1 & l_1 + l_2 \\ -1 & -1 & l_1 + l_2 \\ -1 & 1 & -l_1 - l_2 \end{bmatrix} \tag{4.1.15}$$

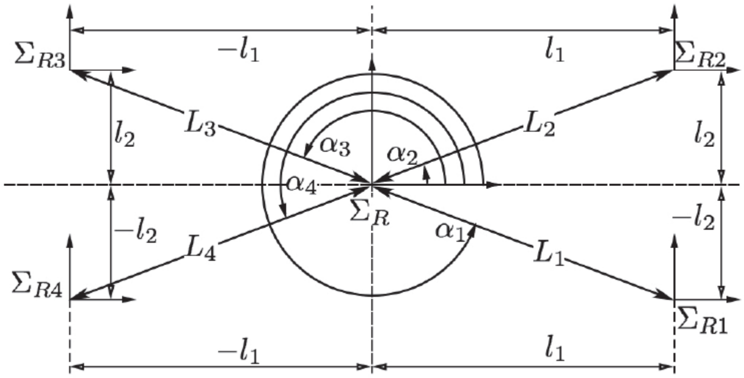


FIGURE 4.1.4. Wheel parameters relative to the robot

The kinematic posture model for an omnidirectional robot is given by Eq. 4.1.5, while the kinematic configuration model for that robot is given by Eq. 4.1.16, where $S(q)$ is described in Eq. 4.1.17.

$$\dot{q} = \begin{bmatrix} \dot{\xi} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{\varphi}_4 \end{bmatrix} = S(q) \cdot \eta \tag{4.1.16}$$

$$S(q) = \begin{bmatrix} R_w^R(\theta) \\ E \end{bmatrix}, \; E = -J_2^{-1} \cdot J_1 \tag{4.1.17}$$

Substituting in Eq. 4.1.16 it is possible to obtain Eq. 4.1.18.

$$E = \begin{bmatrix} \frac{1}{r} & \frac{1}{r} & \frac{(l_1+l_2)}{r} \\ \frac{1}{r} & -\frac{1}{r} & \frac{-(l_1+l_2)}{r} \\ \frac{1}{r} & \frac{1}{r} & \frac{-(l_1+l_2)}{r} \\ \frac{1}{r} & -\frac{1}{r} & \frac{(l_1+l_2)}{r} \end{bmatrix} \tag{4.1.18}$$

So, the kinematic model is presented in Eq. 4.1.19.

Kinematic model

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{\varphi}_4 \end{bmatrix} = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \\ \frac{1}{r} & \frac{1}{r} & \frac{(l_1+l_2)}{r} \\ \frac{1}{r} & -\frac{1}{r} & -\frac{(l_1+l_2)}{r} \\ \frac{1}{r} & \frac{1}{r} & -\frac{(l_1+l_2)}{r} \\ \frac{1}{r} & -\frac{1}{r} & \frac{(l_1+l_2)}{r} \end{bmatrix} \cdot \begin{bmatrix} \dot{x^R} \\ \dot{y^R} \\ \dot{\theta}^R \end{bmatrix} \tag{4.1.19}$$

4.2. Dynamic calculation

In this section, the process of obtaining the robot dynamic model has been developed. The main source used in this section to obtain this model has been [**15**].

The dynamic model for an omnidirectional robot can be calculated thanks to Eq. 4.2.1, considering Eq. 4.2.2 and 4.2.3. So, Eq. 4.2.2 is a generic equation that will be used to determine $[T]_\xi$ and $[T]_\varphi$ of Eq. 4.2.1.

$$R_w^R(\theta)\,[T]_\xi + E^T\,[T]_\varphi = E^T \tau_\varphi \tag{4.2.1}$$

$$[T]_\psi = \frac{d}{dt}\left(\frac{\partial T}{\partial \psi}\right) - \frac{\partial T}{\partial \psi} \tag{4.2.2}$$

$$\tau_\varphi = \begin{bmatrix} \tau_{\varphi 1} & \tau_{\varphi 2} & \tau_{\varphi 3} & \tau_{\varphi 4} \end{bmatrix} \tag{4.2.3}$$

In these equations, T is the sum of robot kinetic energies, $\tau_{\varphi i}$ is the applied torque by the wheel i, m_R is the robot total mass, I_{Rz} is the robot total inertial moment and $I_{\varphi y}$ is the wheel inertial moment. For an omnidirectional robot, T can be expressed as shown in Eq. 4.2.4, where M, I_φ and $\dot{\varphi}$ have the values shown in Eq. 4.2.5-4.2.7.

$$T = \dot{\xi^T}\left(R_w^R(\theta)\right)^T \cdot M \cdot R_w^R(\theta) \cdot \dot{\xi} + \varphi^T \cdot \dot{I}_\varphi \cdot \dot{\varphi} \tag{4.2.4}$$

$$M = \frac{1}{2} diag\left\{m_R, m_R, I_{Rz}\right\} \tag{4.2.5}$$

$$I_{\varphi} = \frac{1}{2} diag \left\{ I_{\varphi y}, I_{\varphi y}, I_{\varphi y}, I_{\varphi y} \right\} \tag{4.2.6}$$

$$\dot{\varphi} = \left[\begin{array}{cccc} \dot{\varphi}_1 & \dot{\varphi}_2 & \dot{\varphi}_3 & \dot{\varphi}_4 \end{array} \right]^T \tag{4.2.7}$$

Developing Eq. 4.2.4, Eq. 4.2.8 is obtained.

$$T = \frac{m_R}{2} \left(\dot{x}^2 + \dot{y}^2 \right) + \frac{I_{Rz}}{2} \dot{\theta}^2 + \frac{I_{\varphi y}}{2} \left(\dot{\varphi}_1^2 + \dot{\varphi}_2^2 + \dot{\varphi}_3^2 + \dot{\varphi}_4^2 \right) \tag{4.2.8}$$

$[T]_{\xi}$ and $[T]_{\varphi}$ terms are given using Eq. 4.2.2, whose results are reflected in Eq. 4.2.9 and 4.2.11.

$$[T]_{\xi} = M_R \cdot \ddot{\xi} \tag{4.2.9}$$

$$M_R = diag \left\{ m_R, m_R, I_{Rz} \right\}, \ \ddot{\xi} = \left[\begin{array}{c} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{array} \right] \tag{4.2.10}$$

$$[T]_{\varphi} = M_{\varphi} \cdot \ddot{\varphi} \tag{4.2.11}$$

$$M_{\varphi} = diag \left\{ I_{\varphi y} \right\}, \ \ddot{\varphi} \left[\begin{array}{c} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \\ \ddot{\varphi}_3 \\ \ddot{\varphi}_4 \end{array} \right] \tag{4.2.12}$$

Upgrading Eq. 4.2.1 with the previous calculations, Eq. 4.2.13 is obtained. In order to pass the accelerations from the world frame to the robot frame, Eq. 4.2.14 and 4.2.15 are presented. The second one uses and derives the top of Eq. 4.1.19.

$$R_w^R(\theta) \cdot M_R \cdot \ddot{\xi} + E^T \cdot M_{\varphi} \cdot \ddot{\varphi} = E^T \tau_{\varphi} \tag{4.2.13}$$

$$\ddot{\xi} = R_R^w(\theta) \cdot \dot{\eta} + \dot{R}_R^w(\theta) \cdot \eta \tag{4.2.14}$$

$$\ddot{\varphi} = E \cdot \dot{\eta} \tag{4.2.15}$$

The result is shown in Eq. 4.2.16, with Eq. 4.2.17 and 4.2.18, where the dynamic model of the omnidirectional robot is presented as the final model related to the operational space.

Robot dynamic model in the operational space

(4.2.16)
$$\bar{M} \cdot \dot{\eta} + \bar{C} \cdot \eta = E^T \cdot \tau_\varphi$$

(4.2.17)
$$\bar{M} = R_w^R(\theta) \cdot M_R \cdot R_R^w(\theta) + E^T \cdot M_\varphi \cdot E$$

(4.2.18)
$$\bar{C} = R_w^R(\theta) \cdot M_R \cdot \dot{R}_R^w(\theta)$$

Since the compact form described in [**19**] (specified in Eq. 4.2.19) works in the robot space (unlike η that is described in the operational space), it is necessary to apply the inverse kinematic transformation calculated in Eq. 4.1.19 and specified in Eq. 4.2.20.

(4.2.19)
$$M\left(q\right) \cdot \ddot{q} + C\left(q, \dot{q}\right) \cdot \dot{q} + g\left(q\right) = \tau$$

(4.2.20)
$$\dot{q} = \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{\varphi}_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{1}{r} & \frac{(l_1+l_2)}{r} \\ \frac{1}{r} & -\frac{1}{r} & -\frac{(l_1+l_2)}{r} \\ \frac{1}{r} & \frac{1}{r} & -\frac{(l_1+l_2)}{r} \\ \frac{1}{r} & -\frac{1}{r} & \frac{(l_1+l_2)}{r} \end{bmatrix} \cdot \begin{bmatrix} \dot{x}^R \\ \dot{y}^R \\ \dot{\theta}^R \end{bmatrix} = K \cdot \eta$$

Since K is not quadratic, the transformation must use the pseudo-inverse matrix, as shown in Eq. 4.2.21.

(4.2.21)
$$\eta = K^+ \cdot \dot{q}, \qquad K^+ = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \frac{1}{l_1+l_2} & \frac{-1}{l_1+l_2} & \frac{-1}{l_1+l_2} & \frac{1}{l_1+l_2} \end{bmatrix}$$

For the same reason (transform the data from the operational to the robot space), frame transformation matrices, such as $R_w^R(\theta)$ or $R_R^w(\theta)$ and the respecting derivates, must be transformed thanks to K^+. Since the only operational variable used by the frame transformation matrices is θ, just know $\dot{\theta}$ (Eq. 4.2.22) and the transformation is done replacing the value of θ.

(4.2.22)
$$\dot{\theta} = \frac{r}{4 \cdot (l_1 + l_2)} \cdot (\dot{\varphi}_1 - \dot{\varphi}_2 - \dot{\varphi}_3 + \dot{\varphi}_4)$$

Lastly, to convert the dynamic model to the compact form [**19**], Eq. 4.2.23 is presented as the final model related to the robot space.

Robot dynamic model in the robot space

$$\bar{M} \cdot \ddot{q} + \bar{C} \cdot \dot{q} = \tau_\varphi \tag{4.2.23}$$

$$\bar{M} = \left(E^T\right)^{-1} \cdot R_w^R(q) \cdot M_R \cdot R_R^w(q) \cdot K^+ + M_\varphi \cdot E \cdot K^+ \tag{4.2.24}$$

$$\bar{C} = \left(E^T\right)^{-1} \cdot R_w^R(q) \cdot M_R \cdot \dot{R}_R^w(q) \cdot K^+ \tag{4.2.25}$$

In the compact form, $\bar{M}$ is the inertial matrix, $\bar{C}$ is the centrifuge and Coriolis matrix, $g(q)$ is the gravity vector and τ is the torque applied by the motors. The content of these matrices is strongly important and relevant in the design of robotic arms, not so much for the platform. Its use is quite relevant in the design of joint controllers. Even so, it is possible to detail some important features about these matrices [**19**]:

- The inertial matrix is very important for the dynamic model (related to the kinematic energy) and for the robot controller design (related to stability studies for robotic arms).
- The centrifuge and Coriolis matrix are also important for stability studies for control systems in robotic arms.
- The gravity vector is presented in a not designed robot, this is without gravity torques of compensation, and in robot destined to move out of the horizontal plane (since the robot presented in this thesis is destined to move in the horizontal plane, this vector is not presented in the dynamic model).

Summary

In this chapter, the kinematic and dynamic models have been calculated.
The kinematic model relates the angular and linear velocity of the robot with the velocity in each wheel.
The dynamic model has been determined to improve the quality of the robot movements, making them softer, quality and providing security to the system and the environment.

CHAPTER 5

GraphSLAM for the robot self-location

In this chapter, the development of the self-localization of the robot has been carried out. The structure of this chapter is the following:

Firstly, the project requirements and specifications are explained in the introduction section with the aim of contextualizing this document. According to them, important decisions have been taken in order to resolve the problem as well as possible, proposing the methodology shown in Section 5.2.

Then, the position estimation is explained, detailing the used sensors and algorithms.

Later, with the objective of reducing the position error due to the inaccuracies of the estimation, a loop closure system is developed and explained.

Finally, the obtained results testing the system are shown, detailing some conclusions.

5.1. Requirements and specifications

The developed robot in Chapter 3 must be localized within the environment to provide its position with the corresponding radiation measurement.

It is known that the maximum permissible error in the relation between the position and the radiation measurement is 10cm. This value is extremely reduced to guarantee that, in case that the supposed and the real radiation in a determinate point of the accelerator are quite different, operators or robots can go to the exact point and fix the problem (leakage, breakage of a sensor, etc.).

The position in the ring is indicated in the main tunnel every ~32m, however, the position of the posting signs is at a very high height and they are very small, so it is very complicated to read the position and comply with the time constrains. So, in order to comply with the objectives, another solution must be proposed.

5.2. Decision-maker

The proposal system of self-localization is composed of:

- An Extended Kalman filter. The output of this system is an estimated position. This block has been developed by Julia Kabalar, a colleague from the workgroup. This Kalman filter integrates three different sources of data:
  - Wheel odometry: It is the use of data from motion sensors to estimate the change in position over time. In this case, the change in position is estimated from the encoders linked to the wheels.
  - IMU odometry: An inertial measurement unit is an electronic device that measures and reports a body's specific force, angular rate and the orientation of the body.
  - Visual odometry: It is the process of determining the position and orientation of the robot by analyzing the associated camera images. In this case, a tracking camera has been used temporarily, while other technologies are developed.
- A 3D Mapper. This block has been developed by Sergio Villanueva, a colleague from the workgroup. The 3D Mapper, known as scan matching system, takes an estimated position as the input and improve this position thanks to some point clouds recorded at the same time as the position estimation. Its methodology is throughout the algorithm ICP[9], using PCL[10]. All the information about the performance of this block can be found in [**20**].
- A graph generator and graph optimizer, which takes as its input the scan matching output (second estimated position). This block generates a graphSLAM problem, where each node is a position. When a position is well known, the loop closure takes part and all the positions are correct.

The performance structure of the SLAM system can be observed in Figure 5.2.1.

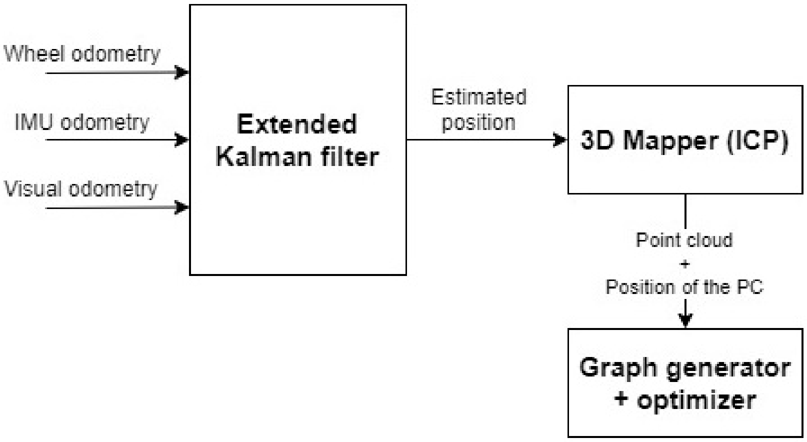


FIGURE 5.2.1. Structure of the SLAM system

[9] ICP: Iterative closest point

[10] PCL: Point Cloud Library

Since the Kalman filter and the 3D Mapper have been developed by other people, this chapter has paid more attention to the Graph generator and Graph optimizer.

5.3. Main procedure

Attending to Figure 5.2.1, the SLAM process begins with the Extended Kalman filter, which is used as sensor fusion and non-linear state estimation. The output of this block generates results with a 1-2% error in around 100m (software tests done with the robot CHARMbot mentioned in Section 2.2.1).

Although this value could seem tiny, it is very critical when the robot is working long term due to the accumulative error. For instance, if the angular position θ is wrong, the following positions, x, and y are getting worse. For this reason, it is necessary to improve the results thanks to other means.

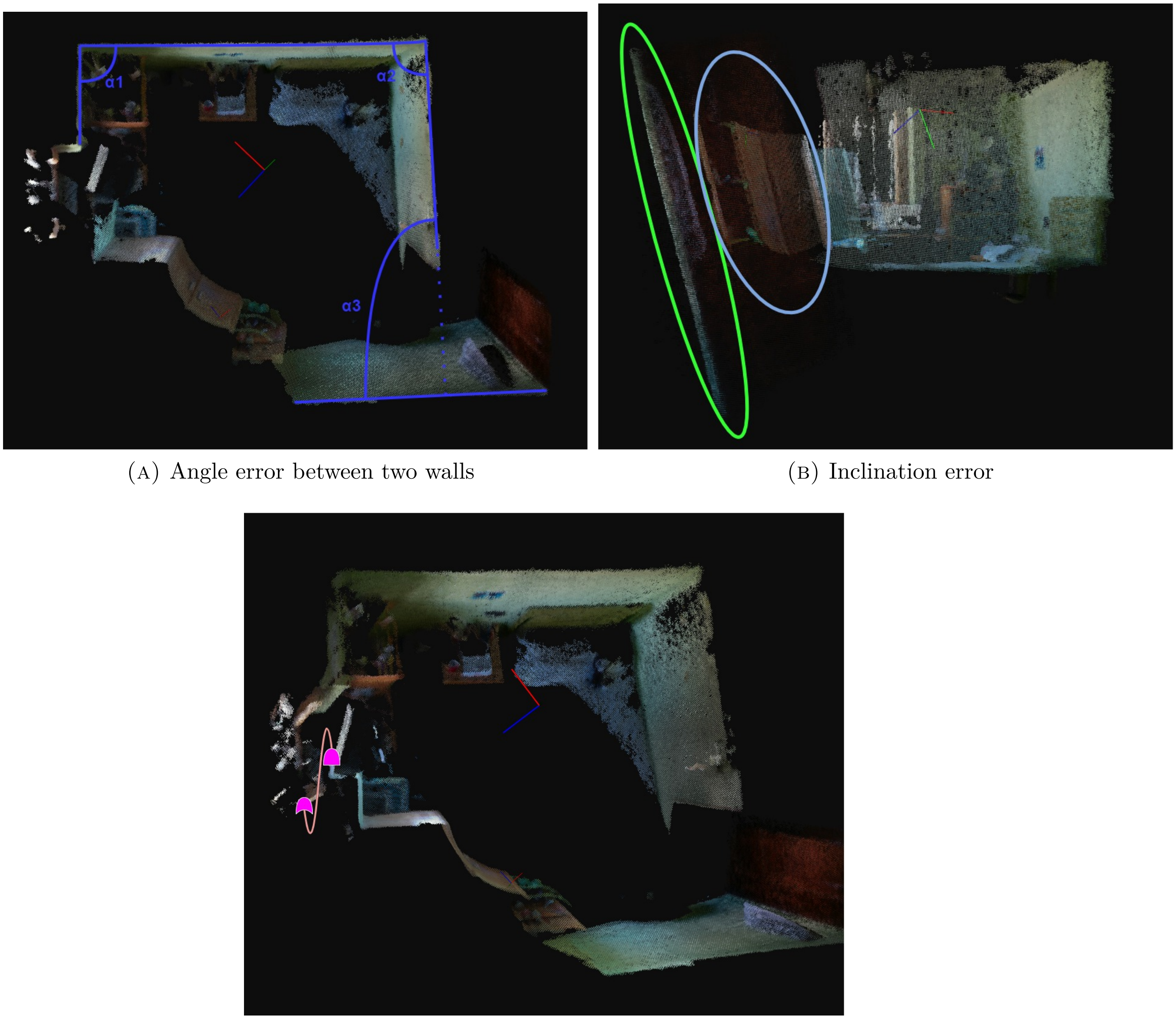
The output of the Extended Kalman filter is the input of the 3D Mapper block. This part generates a 3D reconstruction of the environment and it can be used for scan matching. The goal of scan matching is to find the relative pose or transform, between two robot positions where the data is taken, in our case, where two point clouds (with color data in the case of the RGBd camera) are taken.

Figure 5.3.1 shows the result of a scan matching in a reduced environment, this is, in a room.

In that figure, it is possible to observe the following details:

- Figure 5.3.1a shows how α_1, α_2, and α_3 between the walls are not exactly 90º when the reconstruction is increasing. That is, those angles are 90º locally, but a small error produces cumulative errors. It is possible to observe also in that picture that the line between α_1 and α_2 suffers an error of curvature.
- Figure 5.3.1b shows how the walls are inclined regarding the previous one (reconstruction carried out clockwise). This image shows how a simple reconstruction error (green ellipse) can destroy the following steps (blue ellipse) in case there is no loop closure.
- Figure 5.3.1c shows the results of accumulative errors during reconstruction. If there is a loop closure, point clouds would come together by the pink line. In that case, it is expected that all the previous positions are corrected and the reconstruction would be better.

This problem is because the scan matching works perfectly locally, but not globally, where there are accumulative errors. Any error in the estimation of the transformation between a pair of two consecutive clouds will be passed on to all the subsequent clouds.



(A) Angle error between two walls

(B) Inclination error

(C) Results of the accumulative errors

Figure 5.3.1. Errors and problems of scan matching in a reduced environment

Due to the problems produced by accumulative errors in the Kalman filter and the scan matching, a Graph Generator, and Graph Optimizer have been implemented. The objective of this block is the loop closure, to correct the point cloud positions and, in this way, improve the reconstruction and the location of the robot in the environment.

5.4. Visual omodetry

The tracking camera previously detailed is used for visual odometry. However, other techniques to find features in the taken pictures by the cameras are being developed, such as:

- SIFT[11]. It is a feature detection algorithm in computer vision to detect and describe local features in images [**21**]. Taking two consecutive images (with a distance threshold), it is possible to find the translation vector and rotation matrix between them. This can be used by the EKF as visual odometry. In each picture, SIFT finds features. Then, a matching system finds the correspondence between the features of both images. A RANSAC filter is applied to the correspondence, removing the wrong correspondences that are out of range. Lastly, an error filter is applied, removing the correspondence whose average error is higher than a threshold. The results can be observed in Figure 5.4.1.
- DSO[12]. It is a visual odometry method based on sparse and direct structure and motion formulation [**22**]. To work the algorithm, it is necessary to calibrate de camera (intrinsic, distortion and photometric calibration), having a fish-eye camera, as detailed in [**23**], where the algorithm has been implemented in a whole system. Several tests on the accelerator have been carried out with the code used in [**23**], finding the following conclusions: "This is very vulnerable to scale issues, it is necessary a very accurate intrinsic and photometric calibration, almost impossible to achieve without a calibration table. There are problems with proportionalities of the distance, destroying the EKF. There would be problems closing the loop because it closes its internal loop, so it is not recommendable to use this code in our system".

Other possible algorithms are SURF[13], ORB[14], BRIEF[15], FAST[16], and HARRIS. The performance of each descriptor must be compared and the best one must be selected.

[11] SIFT: Scale-invariant feature transform

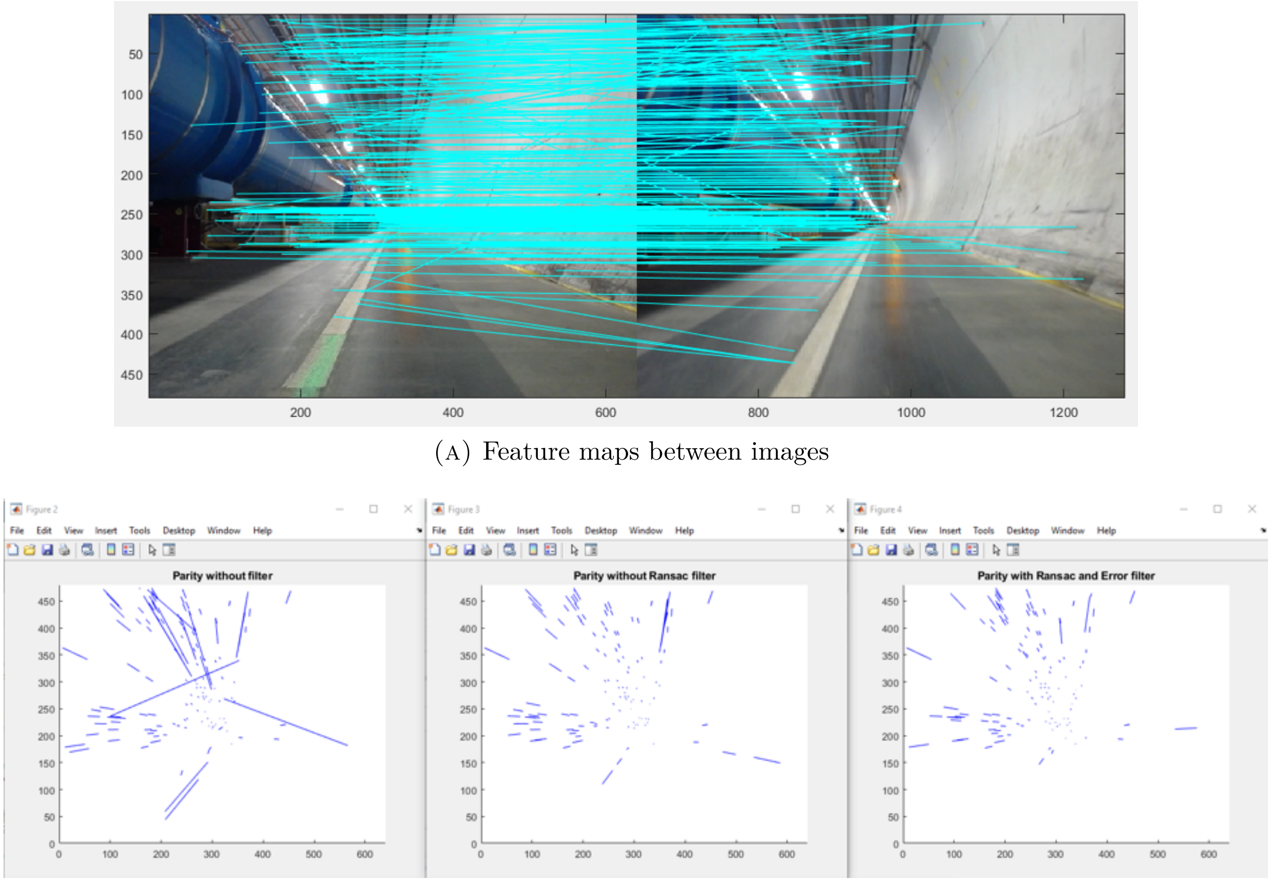
[12] DSO: Direct Sparse Odometry

[13] SURF: Speeded-Up Robust Features

[14] ORB: Oriented FAST and Rotated BRIEF

[15] BRIEF: Binary Robust Independent Elementary Features

[16] FAST: Features from accelerated segment test



(A) Feature maps between images

(B) Results when applying recursive filters

Figure 5.4.1. Results of the SIFT algorithm on accelerator images. Results when iterations are 80, the distance threshold is 50px and confidence is 90% as RANSAC filter parameters.

Since visual odometry has not been fully developed, the tracking camera has been used for testing.

5.5. Graph Generator and Optimizer

Thanks to the Kalman filter and the scan matching system, it is possible to estimate the position of a taken point cloud with good accuracy. In order to remove (or minimize) the accumulative error, a graph generator and optimizer has been developed.

The third block of the process, the Graph Generator, and Optimizer is an algorithm developed with the objective of achievingto achieve SLAM, this is, to get a good position of a robot in any kind of environment. However, this algorithm can be used to improve the reconstruction of the environment since the position of the different point cloud is optimized and improved.

To mount a graph and optimize it, g2o[17], a general framework for (hyper) graph optimization, has been used. It is a C++ framework for performing the optimization of non-linear least squares problems that can be embedded as a graph or in a hyper-graph[18] [**24**].

5.5.1. Graph structure.

GraphSLAM is an over-constrained problem that shall be solved by least-squares. Some basic features about the implemented graph are:

- Each node or vertex in the graph is a robot position (supposed by the scan matching system).
- Each vertex has an associated point cloud, recorded in that position.
- Each edge in the graph is a position relationship between two vertices, given by the scan matching system.
- The graph is over-constrained, this is, usually each vertex has more than one edge connecting to other vertices.

The ideal situation (SLAM as perfect as possible) would be possible if the PC[19] of each vertex is compared with all the others, including the corresponding edge in case that the correspondence between PCs is found. However, this situation is impossible since the system must be a real-time application.

On the contrary, a new position of the scan matching can be registered in the graph in case that there has been an increase in the output values of the EKF sufficiently large in terms of angle, translation or time (meaning that any of these terms surpass a determined threshold. This guarantees uniform distribution of frames in the scene, avoiding unnecessary repetitions and guaranteeing the correct correspondence between PCs by the scan matching system. In order to improve performance, an over-constrained graph has been implemented. There are two options to link vertices with others (excluding the previous one):

- **k-nn method**[20]: Each vertex is compared with the K closest vertices. In this case, it is known that the maximum comparative between PCs when a vertex is included in the graph is K. This is a good notice because in case the robot is several times in the same place, the scan matching would perform only with the closest vertices.

[17] g^2o: General Graph Optimization

[18] Hyper-graph: An extension of a graph where an edge can connect multiple nodes and not only two

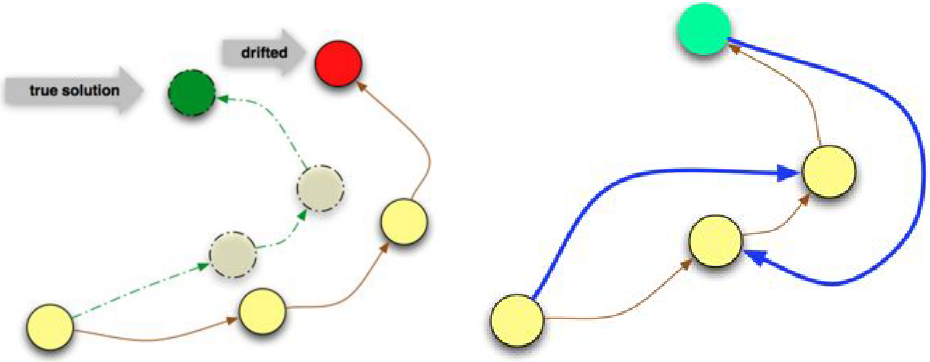
[19] PC: Point cloud

[20] k-nn: k-nearest neighbors

- **Neighbors within a radius**: Each vertex is compared with the vertices that are within a circle of radius R that surrounds the position of the vertex. In this case, it is possible to select a known radius where the scan matching could find correctly the relationship between PCs. Although the number of vertices to compare is very high, it is possible to compare only with the K closest vertices to save time and ensure the real-time feature.

In both cases, the system improves against cumulative errors, as shown in Figure 5.5.1, where the first picture shows the problem of the accumulative errors, this is, a position error induces error in the following positions. The second picture shows the previously explained methodology, creating an optimizable over-constrained graph.

Both methodologies have been implemented thanks to PCL library, which allows using the algorithm KdTree to search the neighbors. It has been explained in Section 5.5.3.



(A) Result of an accumulative error in the graph

(B) Consequence of comparing PCs of a vertex with the closest PCs

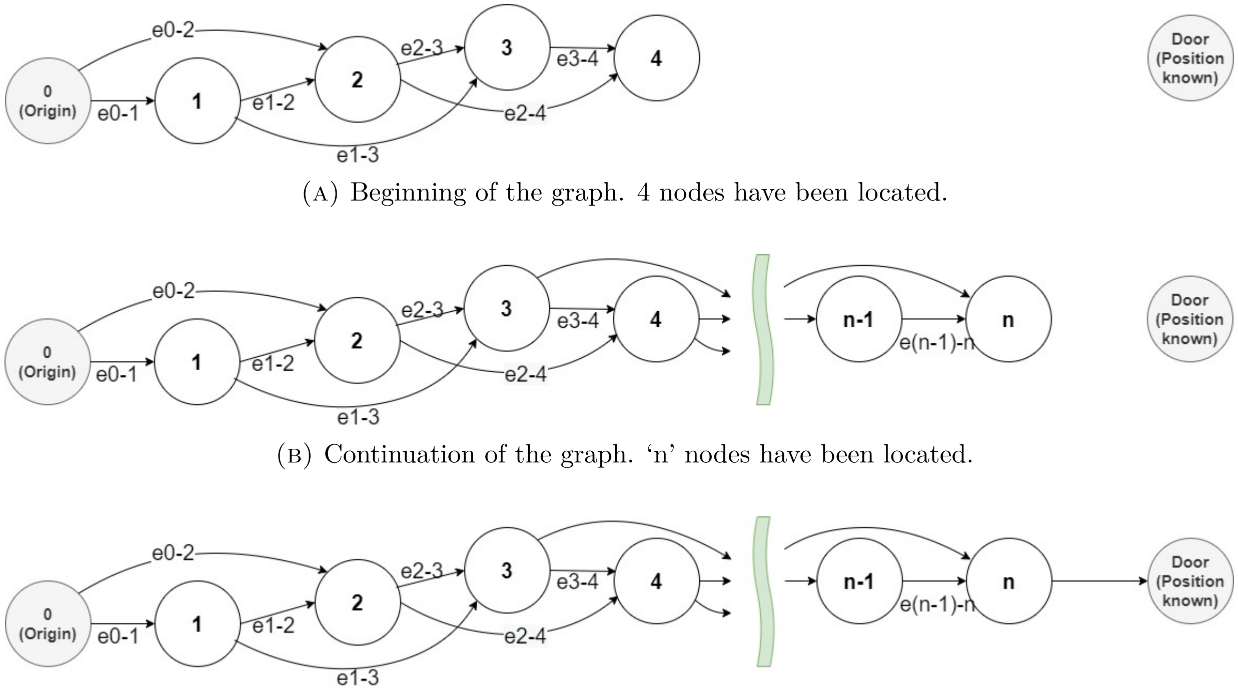
FIGURE 5.5.1. Cumulative error problem and way to reduce

Another advantage of this technique (neighbor search) is the loop closure. When the robot is in a position where it already was, the system generes a loop. These loops are useful to correct the previous position of the robot.

It is important to know that a graph whose vertices are connected only with the previous and the following vertex is not optimizable since all the edge's constraints are always satisfied. The same occurs with a graph without loops, although an over-constrained graph could be optimized, the error would be higher and higher if there is no loop closure.

Since the project demands that the robot has to complete a turn in the accelerator complex with time constraints, it is expected that the robot does not come back to previous positions, making it difficult to close the loop in the graph.

To solve this problem, another technique has been implemented. Since the accelerator door situation is well known (explained in Section 3.1), the idea is to detect them, set a fixed vertex in the graph with its precise position and add the constraints between the robot and that position. As shown in Figure 5.5.2, the origin position and the door position are well known in the large environment (fixed vertices), so all the other vertices can be referred to the fixed vertices and their position can be optimized when a known position is found.



(A) Beginning of the graph. 4 nodes have been located.

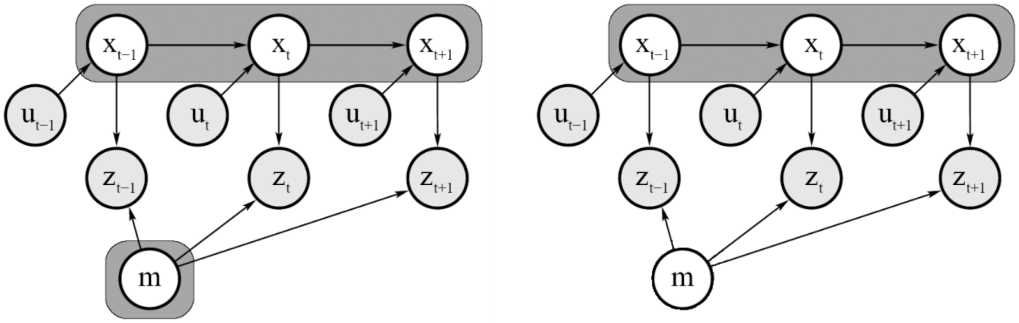
(B) Continuation of the graph. ‘n’ nodes have been located.

(C) Graph completed. The door detector has found the door and the scan matching has calculated the relationship between the door position and the vertex ’n’, adding the constraint and correcting the positions of all the vertices of the graph.

Figure 5.5.2. The technique to close the loop and correct the previous robot positions.

In that picture, gray vertices are fixed, this is, its position is well known. The position of the white vertices can be modified by the optimizer. The shown example uses the k-nn method, being $k = 2$.

Everything together seems to complete a variation of a full SLAM system, understanding that like a system that estimates the entire path in each iteration, unlike a full SLAM system which estimates the entire path and the map in each iteration. In our case, the map is formed at the end of the algorithm.



(A) Full SLAM system. Positions (x) and map (m) are optimized in each iteration.

(B) Partial full SLAM system. Positions (x) are optimized in each iteration

FIGURE 5.5.3. Graphical model of full and partial full SLAM system. In the gray color box, the optimizable items. Gray circles are landmarks and constraints.

The last point to consider is the edge weights. In order to ease the optimizer work, it possible to assign different weights to the constraints. An edge weight must reflect the importance of that edge in the graph. They are reflected in the information matrix Ω, which is the inverse of the covariance matrix.

5.5.2. g2o.

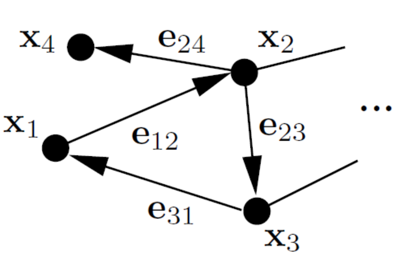
g^2o implements a non-linear graph optimization using least-squares. The least-squares method is a standard approach in regression analysis to approximate the solution of overdetermined systems by minimizing the sum of the squares of the residuals made in the results of every single equation [**25**]. So, g^2o solves the problem by finding the minimum of the function $F(x)$ declared in Eq. 5.5.1. The main source for this section is the official paper of g^2o [**24**], where all the information can be found. It is strongly recommended to read this paper.

$$F(x) = \sum_{(i,j)\epsilon C} e(x_i, x_j, z_{ij})^T \cdot \mathit{\Omega}_{ij} \cdot e(x_i, x_j, z_{ij}) = \sum_{(i,j)\epsilon C} e_{ij}(x)^T \cdot \mathit{\Omega}_{ij} \cdot e_{ij}(x) \tag{5.5.1}$$

$$x^* = argmin_x (F(x)) \tag{5.5.2}$$

In that equation, $x = \left(x_1^T, \ldots, x_n^T\right)^T$ is a vector of robot positions, z_{ij} and Ω_{ij} represent respectively the mean and the information matrix of a constraint relating the positions x_j and x_i, and $e(x_i, x_j, z_{ij})$ is a vector error function that measures how well the robot positions x_i and x_j satisfy the constraint z_{ij}. It is 0 when x_i and x_j perfectly match the constraint, this is, there is no error.

A graph node i represents the robot position x_i (6D pose, this is, position and orientation) and an edge between the nodes i and j represents an ordered constraint between the two robot positions x_i and x_j (expressed as a transformation matrix homogeneous), as shown in Figure 5.5.4 [**24**], where the way of representing $F(x)$ in a graph is detailed.



$$\begin{aligned}\mathbf{F}(\mathbf{x}) = {} & \mathbf{e}_{12}^{\top}\,\mathbf{\Omega}_{12}\,\mathbf{e}_{12}\\ & + \mathbf{e}_{23}^{\top}\,\mathbf{\Omega}_{23}\,\mathbf{e}_{23}\\ & + \mathbf{e}_{31}^{\top}\,\mathbf{\Omega}_{31}\,\mathbf{e}_{31}\\ & + \mathbf{e}_{24}^{\top}\,\mathbf{\Omega}_{24}\,\mathbf{e}_{24}\\ & + \dots\end{aligned}$$

FIGURE 5.5.4. An example that illustrates how to represent an objective function by a graph.

If a good initial guess $\breve{x}$ of the parameters is known, it is possible to obtain a numerical solution using Gauss-Newton or Levenberg-Marquardt algorithms. Approximating the error function by the first-order Taylor expansion around $\breve{x}$, Eq. 5.5.3 is obtained, where J_{ij} is the Jacobian of $e_{ij}(x)$ computed in $\breve{x}$ and $e_{ij} = e_{ij}(\breve{x})$.

$$e_{ij}(\breve{x}_i + \Delta x_i, \breve{x}_j + \Delta x_j) = e_{ij}(\breve{x} + \Delta x) \simeq e_{ij} + J_{ij} \cdot \Delta x \tag{5.5.3}$$

Substituting Eq. 5.5.3 in Eq. 5.5.1, the local approximation Eq. 5.5.4 is obtained.

$$F_{ij}(\breve{x} + \Delta x) \simeq \underbrace{e_{ij}^T \Omega_{ij} e_{ij}}_{c_{ij}} + 2\,\underbrace{e_{ij}^T \Omega_{ij} J_{ij}}_{b_{ij}} \Delta x + \Delta x^T \underbrace{J_{ij}^T \Omega_{ij} J_{ij}}_{H_{ij}} \Delta x \tag{5.5.4}$$

$$= c_{ij} + 2b_{ij}\Delta x + \Delta x^T H_{ij} \Delta x \tag{5.5.5}$$

Thanks to the local approximation, from the initial function of $F(x)$ it is possible to obtain Eq. 5.5.6, where $c = \sum c_{ij}$, $b = \sum b_{ij}$ and $H = \sum H_{ij}$.

$$F(\breve{x} + \Delta x) \simeq \sum_{(i,j)\epsilon C} c_{ij} + 2b_{ij}\Delta x + \Delta x^T H_{ij} \Delta x \tag{5.5.6}$$

$$= c + 2b^T \Delta x + \Delta x^T H \Delta x \tag{5.5.7}$$

At this point, it is possible to minimize in Δx (deriving $F(\breve{x} + \Delta x)$ concerning Δx) by solving the linear system indicated in Eq. 5.5.8, where H is the information matrix of the system.

$$H \cdot \Delta x^* = -b \tag{5.5.8}$$

Then, the solution is obtained by adding the increment Δx^* to the initial guess, as shown in Eq. 5.5.9.

$$x^* = \breve{x} + \Delta x^* \tag{5.5.9}$$

5.5.2.1. *Non-linear solvers.*

To solve these equations, two different non-linear solvers are presented. The general approach assumes that space is Euclidean, which is not valid for several problems in SLAM. Assuming this can lead to sub-optimal solutions in non-Euclidean cases. For this reason, non-linear solvers have been implemented in the g^2o framework.

The first one is the **Gauss-Newton** algorithm, which iterates the linearisation in Eq. 5.5.6, the solution in Eq. 5.5.8, and the update step in Eq. 5.5.9. In every iteration, the previous solution is used as the linearization point and the initial guess until a given termination criterion is met.

The second one is the **Levenberg-Marquardt**, or LM, algorithm, which introduces a damping factor and backup actions to Gauss-Newton to control the convergence, replacing the previous equations with Eq. 5.5.10. In that equation, λ is the damping factor: the higher λ is the smaller the Δx are. This is useful to control the step size in case of non-linear surfaces and to control dynamically this factor. If the new error is lower than the previous one, λ is decreased for the next iteration, otherwise, it is increased.

$$(H + \lambda I) \cdot \Delta x^* = -b \tag{5.5.10}$$

5.5.2.2. *Linear solvers.*

In this case, five different linear solvers can be used:

(1) **Dense Cholesky Decomposition**. In linear algebra, it is a decomposition of a Hermitian, positive definite matrix into the product of a lower triangular matrix and its conjugate transpose, which is useful for efficient numerical solutions, this is, $A = L \cdot L^*$, where L is the lower triangular matrix and L^* denotes the conjugate transpose. When it is applicable, the Cholesky decomposition is roughly twice as efficient as the LU decomposition (diagonal of L is composed of 1 values) for solving systems of linear equations [**26**].
(2) **CHOLMOD** is a set of routines for factorizing sparse symmetric positive definite matrices of form A or $A \cdot A^T$ to solve linear systems of the form $L \cdot x = b$. Both this information and the code source can be obtained in [**27**].
(3) **CSparse** refers to "Direct Methods for Sparse Linear Systems". Its objective is to find the solution of the linear system $A \cdot x = b$ where A is a general square non-singular sparse matrix. This kind of solver provides different types of iterative

solvers based on factorization methods. As the CHOLMOD method, everything can be found in [**27**].

(4) **Eigen linear solver**. This solver is implemented in the library Eigen [**28**], which uses the sparse Cholesky solver. Its performance should be similar to the CSparse solver. This solver allows that x and b would be dense or sparse.

(5) **PCG**. Preconditioned Conjugate Gradient is an iterative method that needs n iterations from a $n\,x\,n$ matrix. This method is usually slower than the Cholesky decomposition [**24**]. It applies to sparse systems that are too large to be handled by a direct implementation such as the Cholesky decomposition and it is also strongly recommended to be used to solve unconstrained optimization problems [**29**].

5.5.3. KdTree.

A k-d tree[21] is a data structure used for organizing some number of points in a space with k dimensions [**31**]. K-d trees are very useful for range and nearest neighbor searches, so it is a good solution to find the closest vertices in the graph.

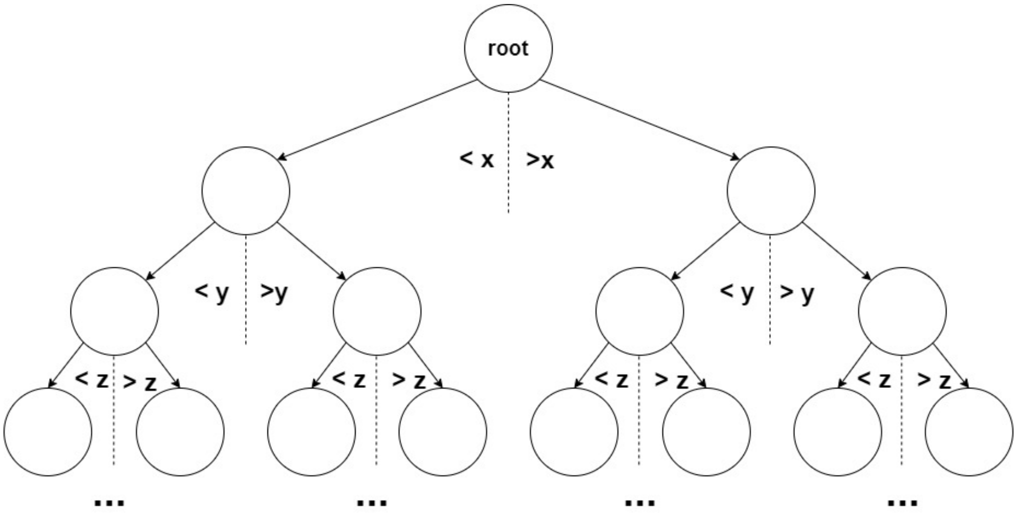


Figure 5.5.5. General 3-d tree

Our case is a three-dimensional tree since it is required to find the closest vertices in (x, y, z). Each level of a 3-d tree splits all children along a specific dimension using a hyperplane that is perpendicular to the corresponding axis. At the root of the tree, all children will be split based on the first dimension, this is, x value. Each level down in the tree divides on the next dimension (y and after it continues with z), returning to the first dimension once all others have been exhausted (Figure 5.5.5).

[21] k-d tree: k-dimensional tree

This information has been found in [**31**], where it is possible to find the source code and practical examples to understand the algorithm.

5.6. SLAM Algorithm

An algorithm has been developed in order to solve the SLAM problem. The flowchart of this implemented algorithm is shown in Figure 5.6.1, where all the points are explained.

Firstly, **point 0 "Pose estimation (Kalman filter)"** is responsible for determining the estimate of the robot's first position. This position is generated as mentioned before, this is, like a sensor fusion. The output of this block is a position increment (a homogeneous transformation matrix, as shown in Eq. 5.6.1) called R_{4x4}.

$$T = \begin{bmatrix} Rot_{3x3} & tra_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotation & Translation \\ 0 & 1 \end{bmatrix} \tag{5.6.1}$$

Point 1 "Coordinate system transformation" consists in a frame change, this is, a conversion between two coordinate systems (that of the robot movement and that of the RGBd camera or LIDAR) thanks to the transformation matrix homogeneous, as shown in Figure 5.6.2, which is known thanks to the mechanical design.

It is necessary to clarify that the test onboard has been carried out with the CHARMbot, named in Section 2.2.1, and with the RGBd camera, so Point 1 is useful since the depth camera is located according to Figure 5.6.2.

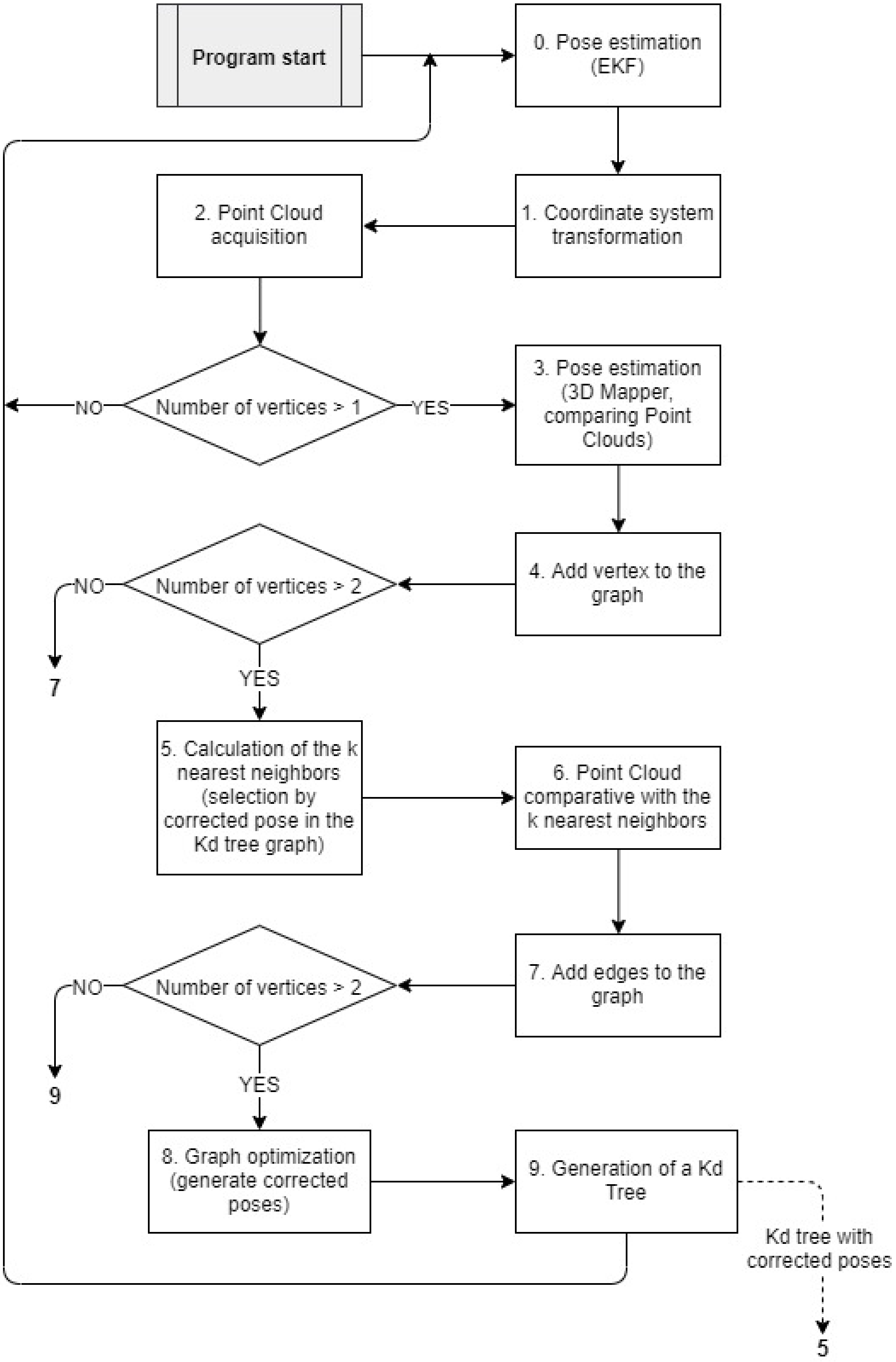

Program start
0. Pose estimation (EKF)
1. Coordinate system transformation
2. Point Cloud acquisition
Number of vertices > 1
NO
YES
3. Pose estimation (3D Mapper, comparing Point Clouds)
4. Add vertex to the graph
Number of vertices > 2
NO
7
YES
5. Calculation of the k nearest neighbors (selection by corrected pose in the Kd tree graph)
6. Point Cloud comparative with the k nearest neighbors
7. Add edges to the graph
Number of vertices > 2
NO
9
YES
8. Graph optimization (generate corrected poses)
9. Generation of a Kd Tree
Kd tree with corrected poses
5

FIGURE 5.6.1. Flowchart of the graph optimization

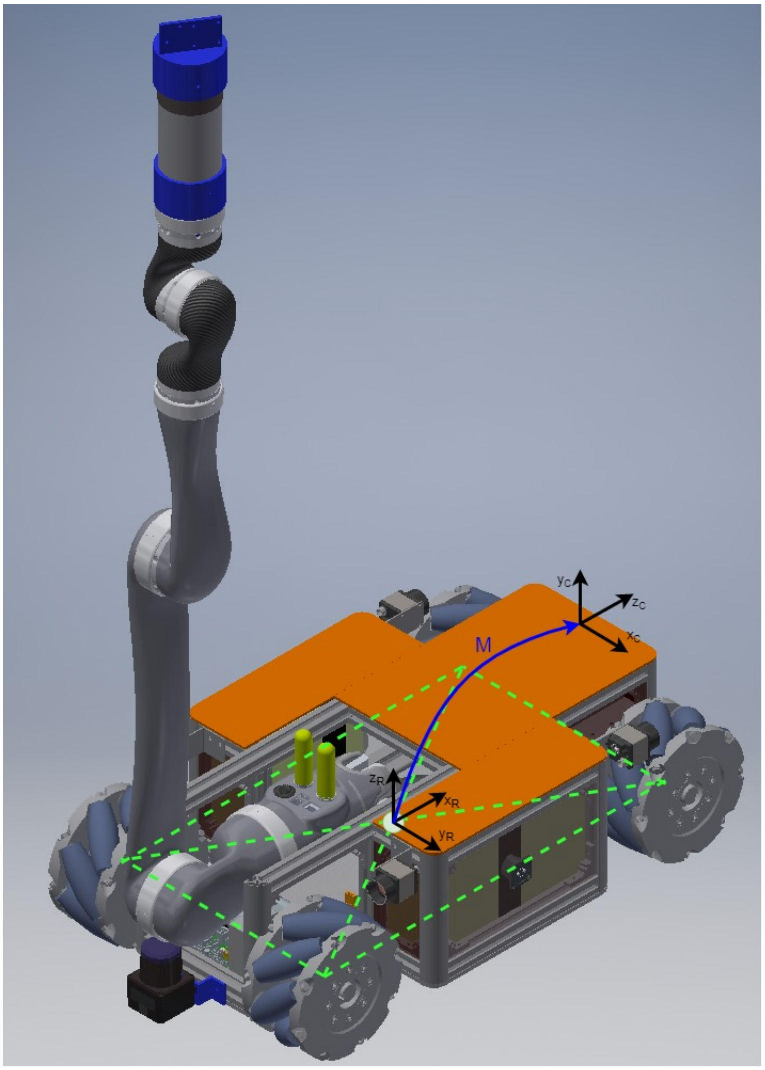


Figure 5.6.2. Coordinate system transformation between the robot movement center point and the sensor central point.

Since the presented design in Chapter 3 is still a prototype and it can be modified before starting its construction, some topics are in discussion. One of them is the location of the 3D LIDAR, which seems to be the chosen sensor for the SLAM task instead of the RGBd camera, whose maximum range is very limited, hindering this task. Some compelling reasons to make this decision are:

- With a higher range, it is easier to find the correspondence between point clouds since more features will appear in the large environment of the accelerator.
- With a higher range, since the correspondence is easier, it is possible to increase the value of "Voxel size" that indicates the accuracy when capturing data. Increasing this value the accuracy of the points and the recording time is lesser.

- Reducing the value of "Voxel size" is also useful to reduce the correspondence time, which means a faster system, less likely to fail.
- Another advantage is the easiness of the loop closure. Since the range is higher, it is possible to find known point clouds in the stored data.
- Since the 3D LIDAR does not provide color information, the correspondence time between two point clouds will be lesser at the cost of being less precise.

The most likely location of the 3D LIDAR is the center of the robot. Removing the top orange plate of the center, there is enough space for this sensor. The main problem is space constrains when crossing the doors. Since the maximum space during this time is 400x200mm, the sensor must not reach this heigh. However, if the sensor is not at a higher height, its field of view is completely useless. In order to fix this problem, a telescopic system must be developed to lift the sensor during the survey and hiding it when it crossed the doors.

Then, the output of this block is a position increment in the sensor reference system, called C_{4x4}, calculated according to Eq. 5.6.2.

$$C_{4x4} = M_{4x4} \cdot R_{4x4} \tag{5.6.2}$$

Point 2 "Point Cloud acquisition" consists in capture a point cloud (with RGB information in case of the RGBd camera) to use that to improve the estimated position by the Kalman filter. When the first PC is acquired, the pose estimation is the origin and the system comes back to point 0. However, when there are some of them, the system continues to Point 3.

Point 3 "Pose estimation (3D Mapper, comparing Point Clouds)" uses the scan matching to improve the estimated position by the EKF. This block compares the taken point cloud with the previous one, generating a homogeneous transformation matrix (Eq. 5.6.1) which contains the relationship between both positions (the place where the point clouds were taken). The output of this block is a position increment in the sensor reference system, called S_{4x4}.

Point 4 "Add vertex to the graph". The position increment S_{4x4} is summed to the previous vertex position V_{n-1}, taken from the graph, obtaining the new vertex position V_n according to Eq. 5.6.3. Then, the new position and the corresponding PC are added to the graph in the form of a vertex. At this moment, the vertex is not connected to any. If there are more than two vertices in the graph, the algorithm continues in point 5, otherwise, it jumps to point 7.

$$V_n = V_{n-1} \cdot S \tag{5.6.3}$$

Point 5 "Calculation of the k-nn". When the number of vertices is higher than two, it means that it is possible to find a correspondence between the current PC and some PCs from other vertices (excluding the previous one that has already found).

Previously to get this point, point 9 has been already got, so a kdTree is available in the system. Thanks to that, it is possible to explore the tree, looking for the nearest neighbors. Of course, as indicated in Figure 5.6.1, the vertices position in the graph are the most correct, since there is an optimizer.

Depending on the chosen mode (k-nn or radius search), point 5 gives a specific number of vertices, of course, the closest.

Point 6 "Point cloud comparative with the k nearest neighbors". At this point, the PC associated with the vertex is compared with the PCs of the closest vertices. So, depending on the number of the nearest neighbors found in point 5, the same number of comparisons is made. The higher this number, the more precise and slow the algorithm will be, so the midpoint must be found (usually between 2-4). The result of this point is a series of position transformation.

Point 7 "Add edges to the graph" is responsible for adding the edges between vertices. Since the added vertex is alone, all its edges are added to the graph, completing the graph in this iteration. If the number of vertices is higher than two it is possible to optimize the graph and the algorithm continues to point 8. Otherwise, the system jumps to point 9, since between two vertices with a simple relationship the optimization is useless.

Point 8 "Graph optimization" is responsible for reducing the accumulative error and correcting the vertices position when the loop is closed. The quality of the system is completely dependent on the performance of the optimizer. The optimization is carried out solving the least-squares problem explained in Section 5.5.2, using the non-linear and linear solvers explained in the same section.

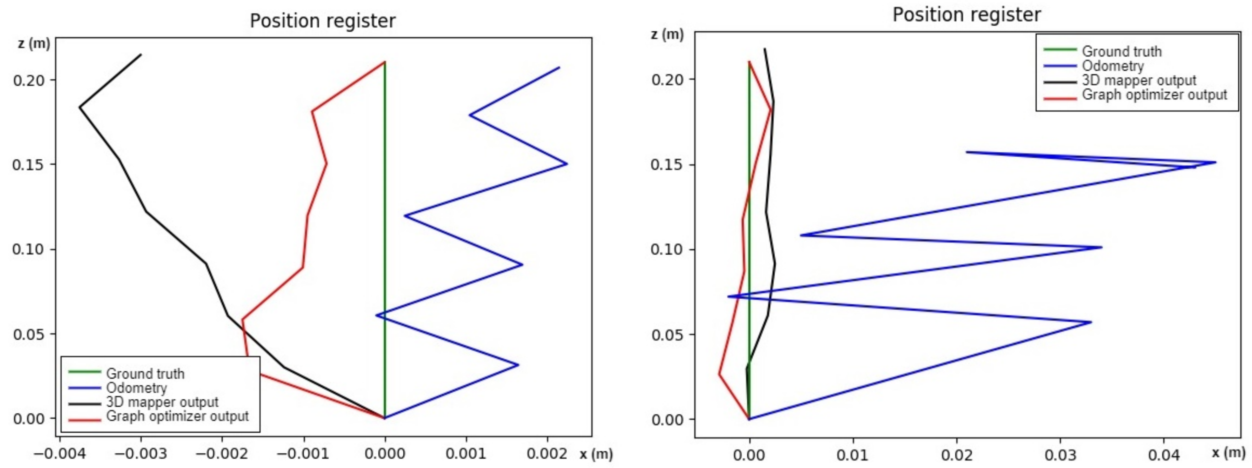
Point 9 "Generation of a KdTree". Once the graph is optimized, the position of the vertices is usually changed. The vertices position in the graph is introduced in the kdTree algorithm explained in Section 5.5.3, generating a tree with the position of the vertices, classified according to their values. It is used by point 5, where the work of finding the nearest neighbors is significantly easier.

CHAPTER 6

GraphSLAM results

The first test has been carried out in only one dimension (1D), testing the performance of the system and obtaining the result shown in Figure 6.0.1.

This test has been developed with good and bad odometry (generated in a simulated way), obtaining good results in both cases, even when odometry is a disaster.



(A) Good odometry.

(B) Bad odometry.

FIGURE 6.0.1. 1D test results.
In both cases, the camera has been moved in 3cm steps on the 'z' axis (important to observe the scale).

The second test has been carried out in two dimensions (2D), using a grid in a table, where each corner reflects a known position. The result of this test is shown in Figure 6.0.2.

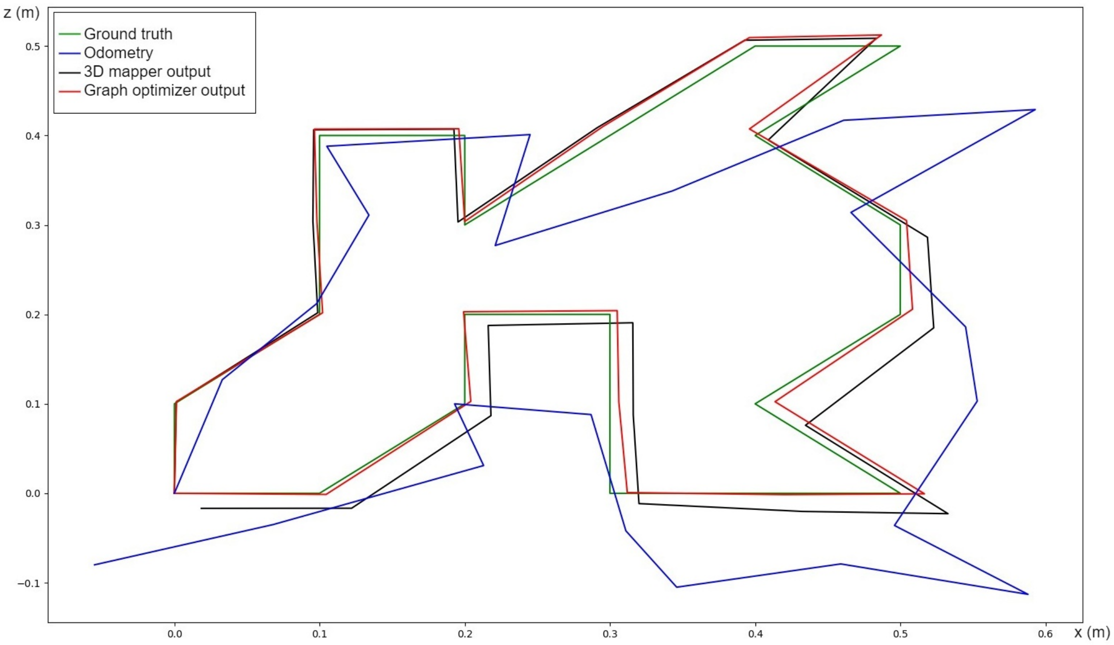


Figure 6.0.2. 2D test results.
The camera has been moved in different steps on the 'x' and 'z' camera axis.

In that figure, it is possible to observe the following details:

- Since the odometry has been generated in a simulation way, the blue line separates from the ground truth line, as expected to happen with robot odometry.
- The 3D mapper output (the scan matching) generates a very good output, however, this signal accumulates errors that have to be corrected.
- The graph optimizer output follows faithfully the ground truth trajectory since the accumulative errors of the scan matching have been reduced.

When this test has been done, a radiation sensor was the object of the environment reconstruction. The result is shown in Figure 6.0.3. In this picture, it is possible to observe that the reconstruction is extremely good since the words of the radiation sensor can be read easily, unlike if only the scan matching were used.

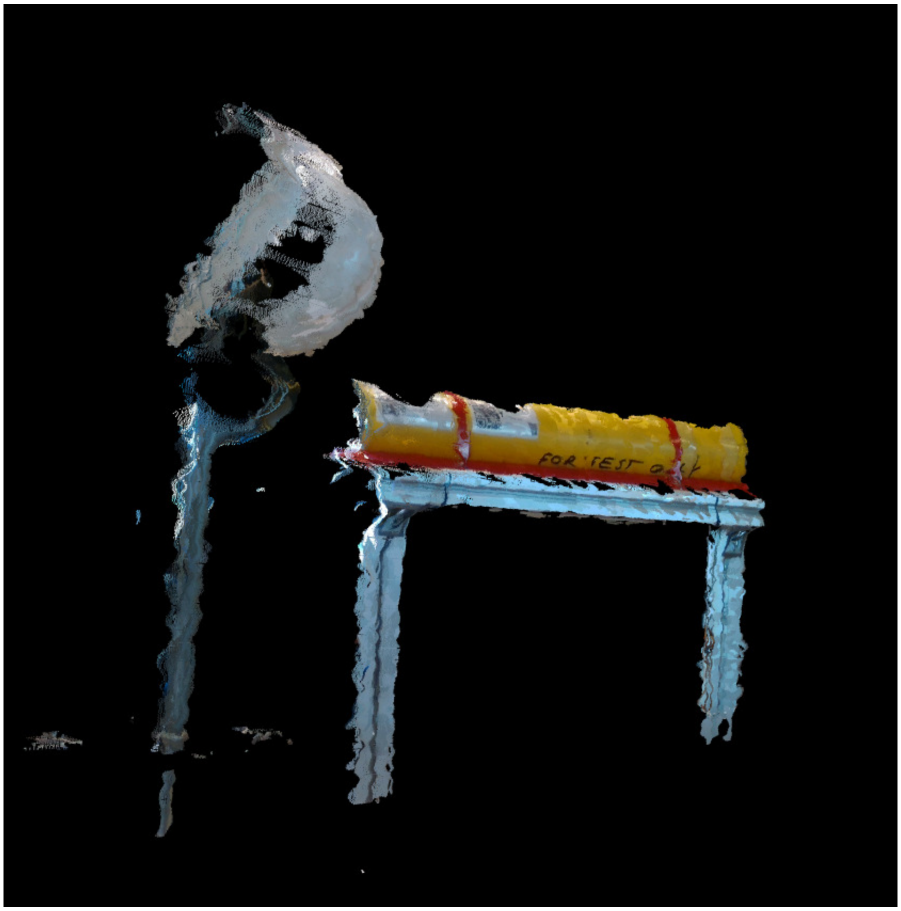


Figure 6.0.3. Radiation sensor reconstruction

To test the good performance of the system, another two tests have been developed, where the followed path has been a zigzag trajectory, starting and finishing in the same point:

- Good behavior test of the over-constrained graph. This test consists of not closing the loop with fixed vertices and lets the system close loops thanks to the edges between nearby vertices. The result is shown in Figure 6.0.4. In this picture it is possible to observe that, even if the endpoint is not fixed, interconnections between vertices reduce the cumulative error, leaving the endpoint practically at the same starting point.
- Good behavior test of the loop closure with fixed vertices in a non-over-constrained graph. This test consists of starting and ending in a known position, correcting the rest positions. The result is shown in Figure 6.0.5. In this picture, it is possible to observe that, since the graph is not over-constrained, the vertices position has not been modified notably, but it has improved according to the imposed restriction.

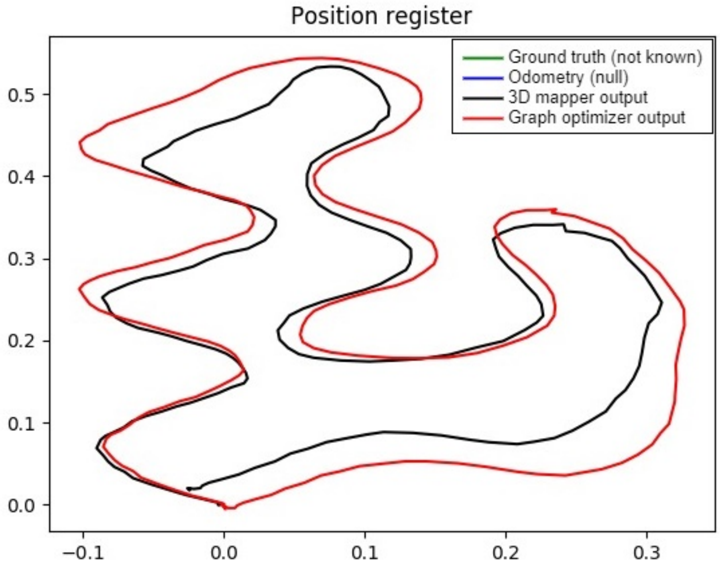


FIGURE 6.0.4. Good behavior test of the over-constrained graph

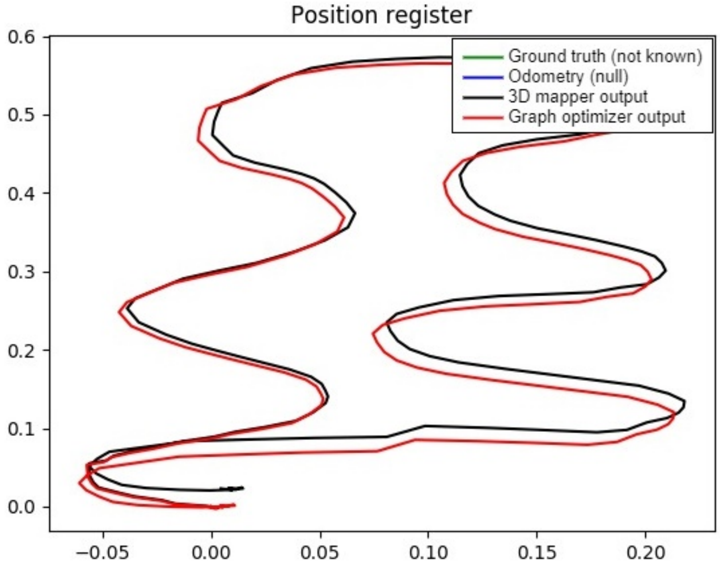


FIGURE 6.0.5. Good behavior test of the loop closure with fixed vertices

Summary

The responsible robotic platform of checking the good state of the sensors in tunnels must have a proper localization in the ring to link the position with the radioactivity. The only accessible landmarks from the robot (meeting the maximum times specified) are the yellow doors located at a variable distance.

To get the objectives, a graphSLAM system has been proposed. Each vertex of the graph corresponds to a robot position, while an edge corresponds to a constraint between two positions.

The system starts estimating the position thanks to an EKF, which can use the wheel odometry, the IMU odometry and the visual odometry (according to the user selects). With the estimated position, a scan matching system (known as 3D Mapper) computes a 3D transformation between two point clouds taken in two robot positions (the current and the previous one). Finally, the graph generator and graph optimizer take part.

This last system uses the information given by the scan matching to generate a graph according to the proposed algorithm in Section 5.6.

The obtained results show a superlative improvement concerning the generated odometry and a reasonable improvement respecting to the scan matching output.

CHAPTER 7

Study of microcontrollers under radioactivity

In this chapter, a study of microcontrollers under radioactivity has been prepared. The main objective is to dispose of a PCB able to support high quantities of irradiation in a contaminated environment to its performance like an integrated controller in the robotic arm joints. In these environments, the main problem is found in the microcontrollers, since it is the first that breaks down due to radiation, while the rest of the components withstand greater amounts of time.

The structure of this chapter is the following:

Firstly, the microcontrollers that have been studied are presented, then how they can be programmed is explained. After that, the layout of the microcontrollers in the system is detailed, from the electronic, electrical and communication point of view, deepening in the communication protocol. Later, some motion patterns of servomotors controlled by the microcontrollers are shown. Finally, different possible cases of failure in the microcontroller are detailed.

7.1. System approach

The test consists of the radiation of four different cases of microcontrollers that control a servomotor. Each case is set for different devices or different configurations. In this way, the set is formed by:

(1) A low-cost circuit board, which has been developed manually, where an ATmegaS64M1 has been instanced. This microcontroller is the Radiation Hardened model of its family and can be replaced by his homolog, the ATmega64M1. This PCB will be put with its servomotor within the radioactive area.
(2) An Arduino board with its motor within the radioactive area.
(3) An Arduino board covered by shielding of lead, with its servomotor within the radioactive area.
(4) An Arduino board outside of the radioactive area, with the servomotor inside.

Both microcontrollers, ATmega64M1 and ATmegaS64M1, are from the same manufacturer and the same family. For this reason, they have the same pin configuration (shown in Figure 7.1.1).

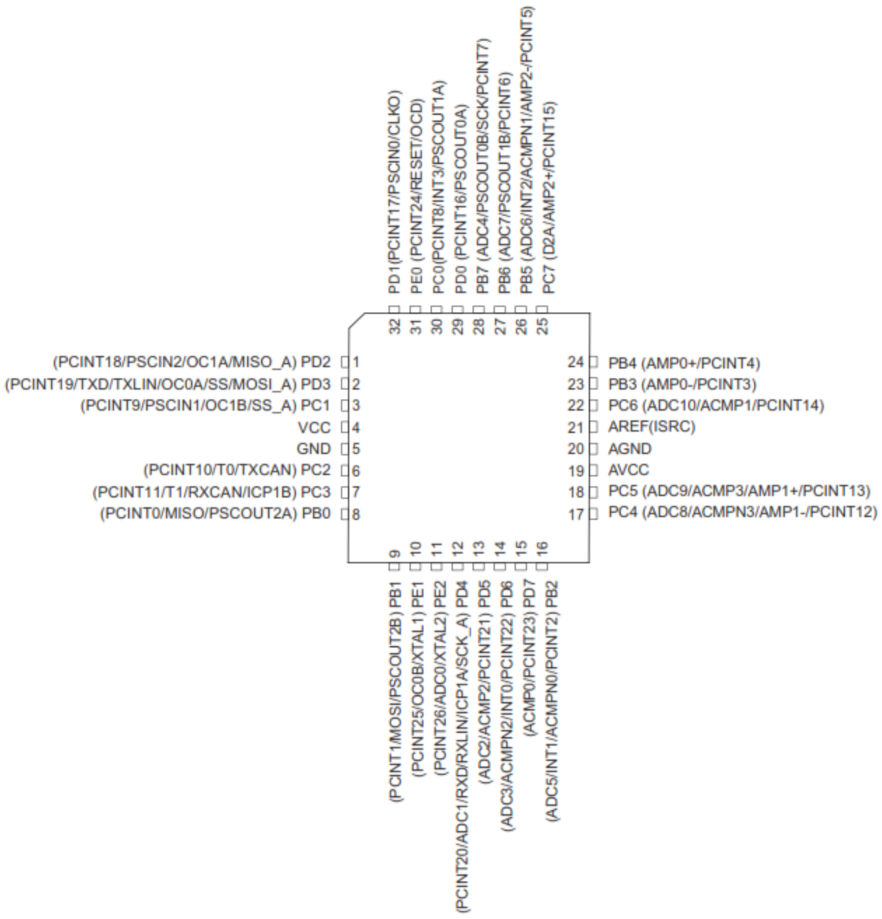

PD1(PCINT17/PSCIN0/CLKO) 32
PE0 (PCINT24/RESET/OCD) 31
PC0(PCINT8/INT3/PSCOUT1A) 30
PD0 (PCINT16/PSCOUT0A) 29
PB7 (ADC4/PSCOUT0B/SCK/PCINT7) 28
PB6 (ADC7/PSCOUT1B/PCINT6) 27
PB5 (ADC6/INT2/ACMPN1/AMP2-/PCINT5) 26
PC7 (D2A/AMP2+/PCINT15) 25
(PCINT18/PSCIN2/OC1A/MISO_A) PD2 1
(PCINT19/TXD/TXLIN/OC0A/SS/MOSI_A) PD3 2
(PCINT9/PSCIN1/OC1B/SS_A) PC1 3
VCC 4
GND 5
(PCINT10/T0/TXCAN) PC2 6
(PCINT11/T1/RXCAN/ICP1B) PC3 7
(PCINT0/MISO/PSCOUT2A) PB0 8
24 PB4 (AMP0+/PCINT4)
23 PB3 (AMP0-/PCINT3)
22 PC6 (ADC10/ACMP1/PCINT14)
21 AREF(ISRC)
20 AGND
19 AVCC
18 PC5 (ADC9/ACMP3/AMP1+/PCINT13)
17 PC4 (ADC8/ACMPN3/AMP1-/PCINT12)
9 (PCINT1/MOSI/PSCOUT2B) PB1
10 (PCINT25/OC0B/XTAL1) PE1
11 (PCINT26/ADC0/XTAL2) PE2
12 (PCINT20/ADC1/RXD/RXLIN/ICP1A/SCK_A) PD4
13 (ADC2/ACMP2/PCINT21) PD5
14 (ADC3/ACMPN2/INT0/PCINT22) PD6
15 (ACMP0/PCINT23) PD7
16 (ADC5/INT1/ACMPN0/PCINT2) PB2

Figure 7.1.1. ATmega64M1 and ATmegaS64M1 Pinout

There are two possible ways to program a microcontroller:

(1) Through another microcontroller that performs like a firmware[22] code source, this is, as a programmer. This is the case of Arduino boards, which have two microcontrollers, one for reproducing the logic sequence and another for the programming action.
(2) Through an external programmer which overwrites the bootloader [23].

The first option has been used in the case of Arduino boards, while the second option in the case of the low-cost PCB (and in the cases when the serial port to communicate with the programmer microcontroller of the Arduino boards does not work).

[22] Firmware: A computer program which establishes the logic that controls the electronic circuits of any device from the lowest level

[23] Bootloader: Simple computer program that does not have the totality of the functionalities of the operating system, which is designed exclusively to prepare everything the operating system needs to work.

The second option involves the AVR Pocket Programmer, which has been chosen due to its low price and its proven performance to program ATmega microcontrollers. The process guide to program a microcontroller thanks to this programmer is detailed in Appendix C, where its components, the methodology to connect the items, the electrical scheme, the way of programming and the way to upload the programs are described, following the requirements imposed on [**10, 11, 12**].

The structure of the system to store important data during radioactive exposure is the following:

Between the existing topologies, a bus topology (multipoint connexion way, this is, decentralized architecture) has been chosen, where an Arduino Mega performs like the master of the communication, controlling the times when the others devices will send data. The master sends a request of information sequentially to the different nodes or slaves (Arduinos Uno and PCB). When a node receives a request from the master, it sends the data following the established protocol. Furthermore, there is an initial and final sequence that has to be followed to guarantee proper communication.

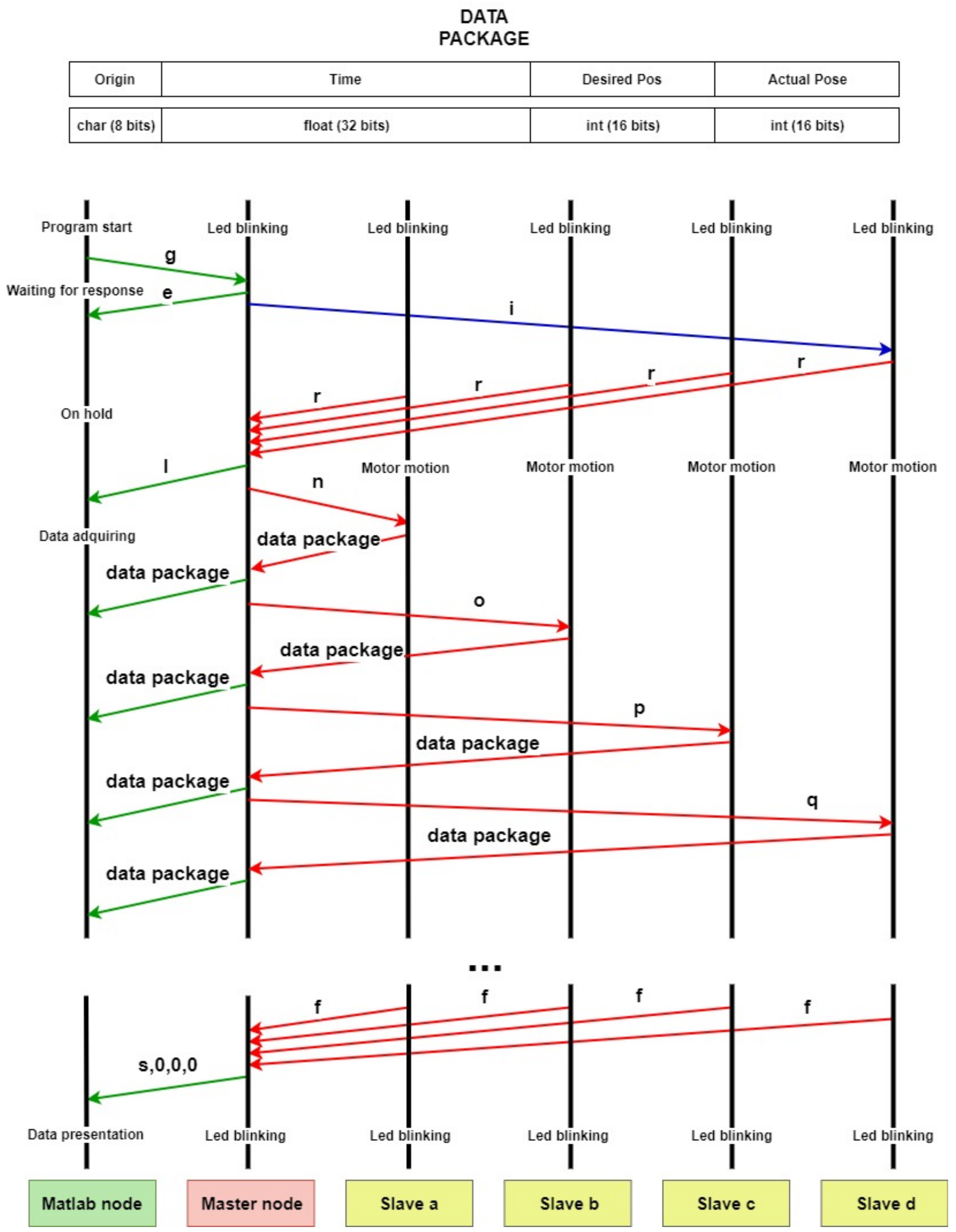
Externally, a computer indicates the beginning of the sequence. When a program is executed, the initial sequence is carried out and the master starts to send data to the computer. It stores all the information while the motors are moving. After everything is finished, the information is shown and stored in files.

In the case that a slave fails and stops sending data to the master, it is the turn of the next slave. In this way, if this fact happens, the other slaves can continue sending information.

7.2. Communication protocol

The previous explanation is reflected in Figure 7.2.1, where the different letters indicate the following:

- **g**: Global started.
- **e**: Global started signal received properly.
- **i**: Broadcast signal to start the motor motion.
- **r**: Motor motion has started.
- **l**: Start with data acquisition.
- **n, o, p, q**: Request for data to the slaves (identifiers for each one).
- **data package**: Information within a data package protocol where the slave origin (a, b, c or d, each for each one), the actual time and the desired position and the actual position of the motor are sent.
- **f**: Motor motion finished.
- **s,0,0,0**: Every motor motion finished.



DATA PACKAGE

Origin	Time	Desired Pos	Actual Pose
char (8 bits)	float (32 bits)	int (16 bits)	int (16 bits)

Figure 7.2.1. Nodes and communication protocol between them

The communication between the different nodes has been established through the RS-485 protocol[24], using the RS-485 module, which performs like a transformer of Serial TTL[25]. The RS-485 protocol is an industrial protocol widely used for its robustness, easy implementation and good performance. Using it as a physical layer it is possible to implement an industrial field bus.

7.3. Electronic scheme

The electronic scheme can be observed in Figure 7.3.1, where the following important lines and components are shown:

- Arduino boards and low-cost PCB. Each one uses two TTL lines (serial communication lines), a PWM digital output, a digital output, and an analogic input.
- Converter DC/DC (12V-3.3V). Its function is to work as the power supply of the low-cost PCB.
- RS-485 modules. On one side, the RS-485 lines (A and B) and the power lines (V+ and V-). On the other side, the receiver/sender switch and the TTL lines (one for send and the other for receive).
- Servomotors. Each one has four lines: V+, V-, PWM signal (control) and an analogic signal which indicates the actual position of the rotor.
- Line A and B: Communication channel that works with differential voltage. RS-485 protocol is defined as a differential multipoint transmission bus system, it is ideal for transmitting at high speeds over long distances and through noisy channels since it reduces the noise that appears in the voltages produced in the transmission line.
- The power supply of 12V, supplied by a source. It feeds all the motors and boards, this is the whole system.

[24] RS-485 protocol: Standard defining the electrical characteristics of drivers and receivers for use in serial communications systems.

[25] TTL Technology (Transistor-transistor logic): Method of serial communication, whose digital electronic circuit uses bipolar transistors.

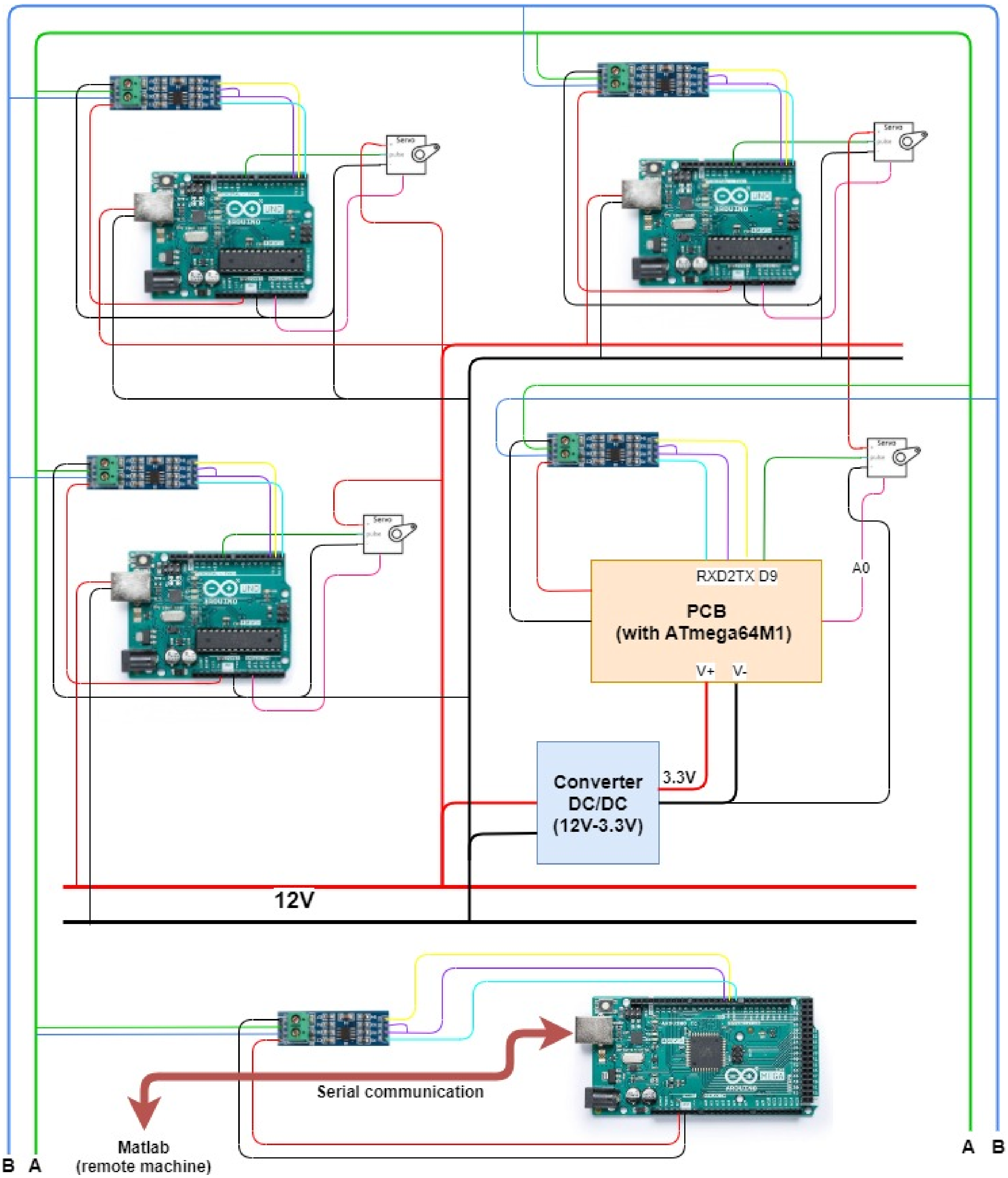


FIGURE 7.3.1. Electronic scheme of the set

7.4. Tests

Before bringing the whole system to the testing environment, some laboratory tests have been carried out. Due to the material constraints, only one servomotor is available. The first step has been to open the servomotor and get the position signal.

One this is done, the programs have been executed, where a step signal has been sent, having the performance of Figure 7.4.1. As it is possible to see in this figure, some noise signals are received when the servomotor is set at 0º. This noise is due to the cable that connects the position signal of the motor and the analogic input in the microcontroller. Furthermore, it is possible to appreciate that the servomotor takes around 750ms to go from 0 to 180º, with 20ms of response time.

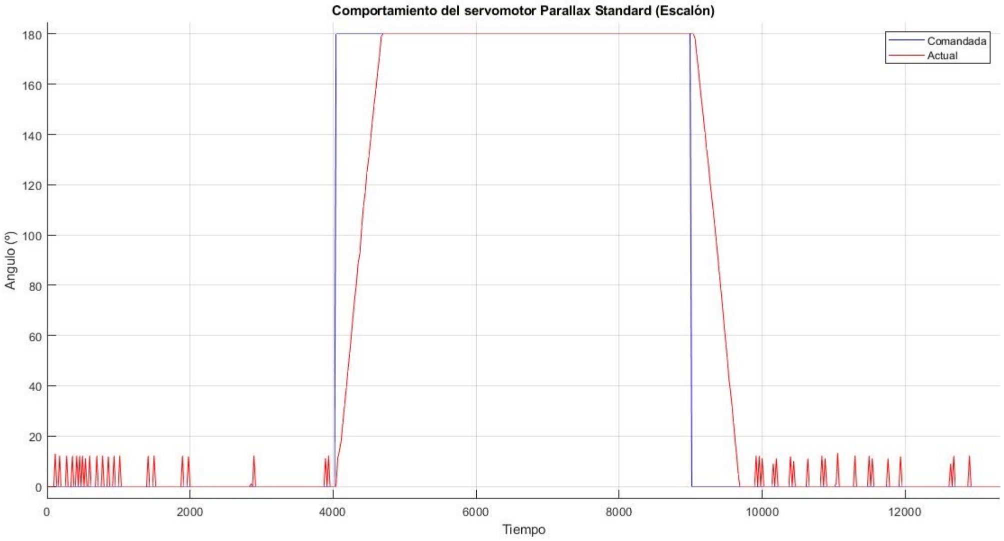


Figure 7.4.1. Servomotor Parallax Standard performance (Step input)

Another program launched has been the corresponding of several ramp and step signals. In this case, the result is shown in Figure 7.4.2, where it is possible to appreciate the mentioned noise from low angles. Besides, it is possible to see that the static position is not the same in the steps.

When the program testing has been done, the number of connected microcontrollers to the bus has been increased. Firstly, two microcontrollers have been connected, obtaining the result of Figure 7.4.3. In this case, there is no signal of the servomotor's position in any of the microcontroller, having the analogic input in the air.

The next step is the connexion of the four microcontrollers to the bus line. In this case, the result is shown in Figure 7.4.4, where the three Arduino boards and the low-cost board are sending data to the master and it is sending to the computer.

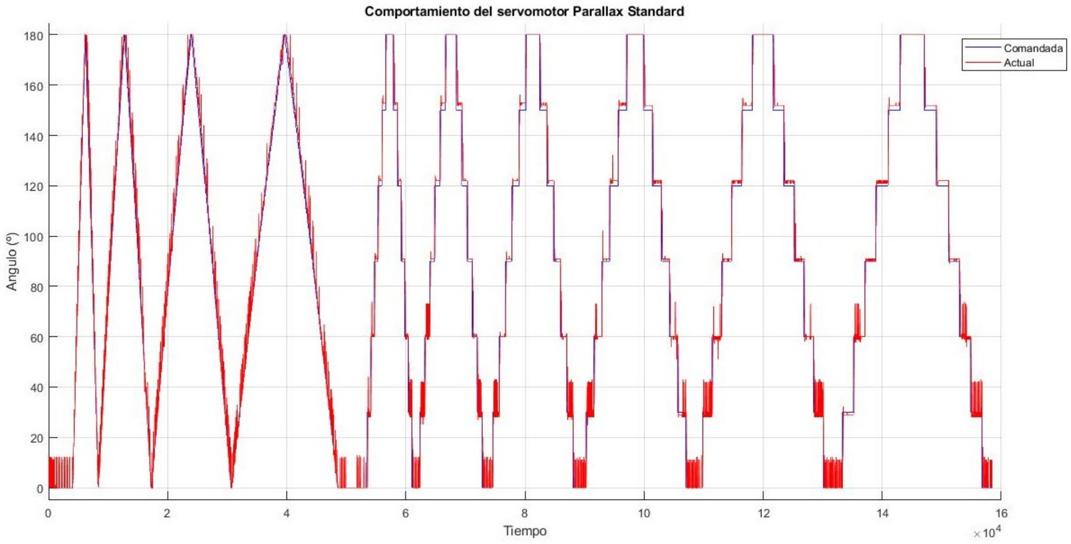


Figure 7.4.2. Servomotor Parallax Standard performance (Ramp and Step input)

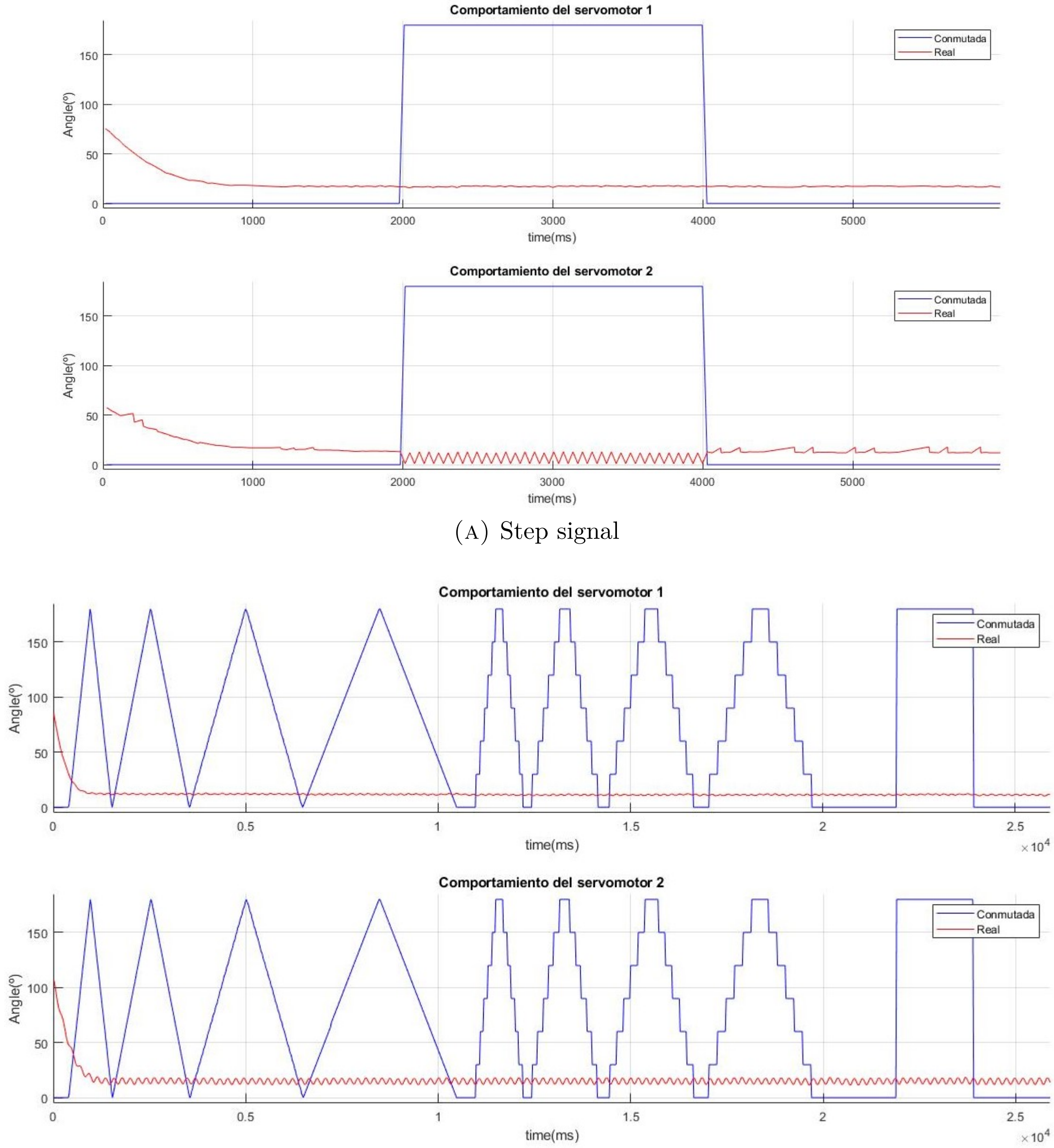
The last thing to do is the test of the performance of the microcontrollers when a failure occurs. This is the case of Figures 7.4.5, 7.4.6, 7.4.7 and 7.4.8, where the occurred failures are explained in their caption.

Thanks to this background (generated causing the failures), it would be possible to determinate what, when and how has been the problem in each microcontroller.

All the programs (Arduino and Matlab) can be found in the following repository: https://github.com/ca
servomotor-system-controlled-from-a-remote-machine-through-RS-485-protocol.git.

Summary

In order to prepare a study of microcontrollers under radioactivity, a set of four microcontrollers has been prepared. The selected items have been explained and also the communication between them. It has been carried out developing a custom high-level protocol to achieve that the master node collects the data from the others and it sends the data to the computer. The physical connexion has been presented in a scheme and some tests have been shown and explained. The result of this chapter is a system ready for being tested in a laboratory.



(A) Step signal

(B) Ramps and steps signal

FIGURE 7.4.3. Received signals from two microcontrollers

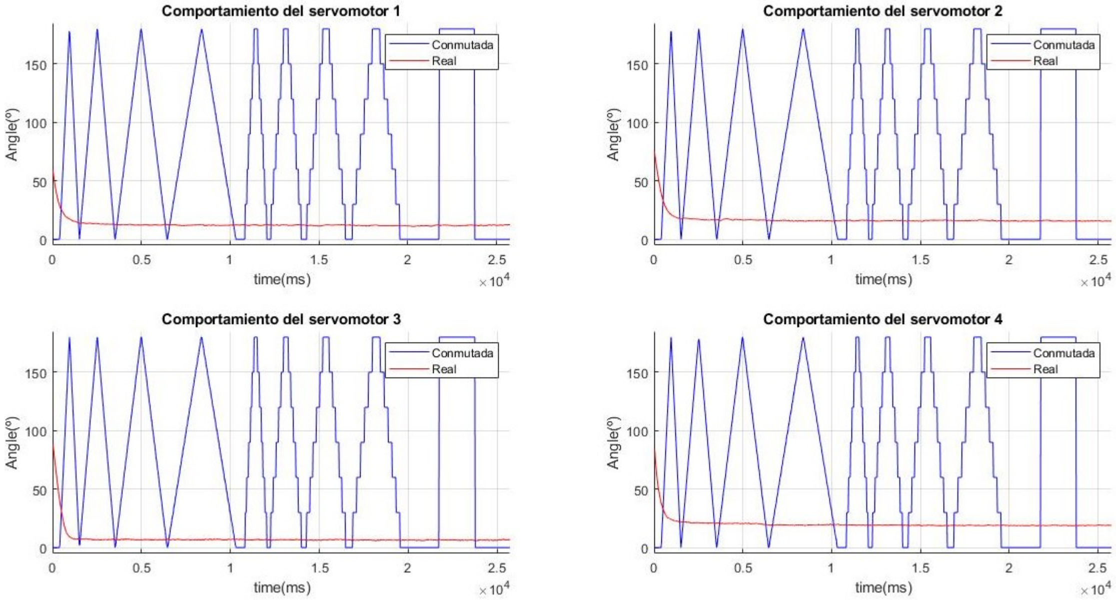


FIGURE 7.4.4. Received signals from four microcontrollers

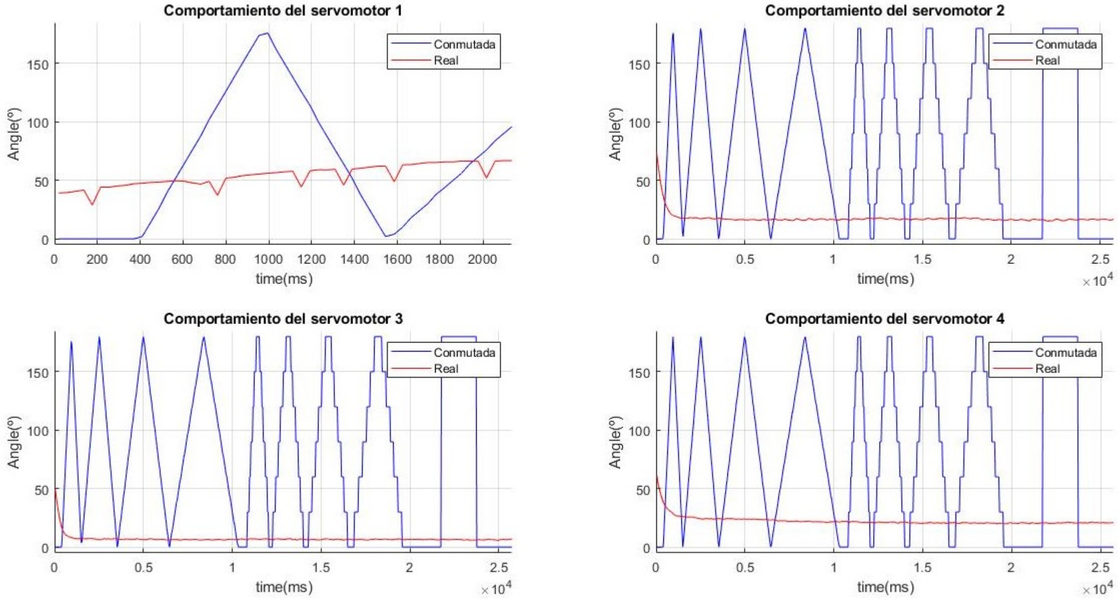


FIGURE 7.4.5. Failure of one microcontroller

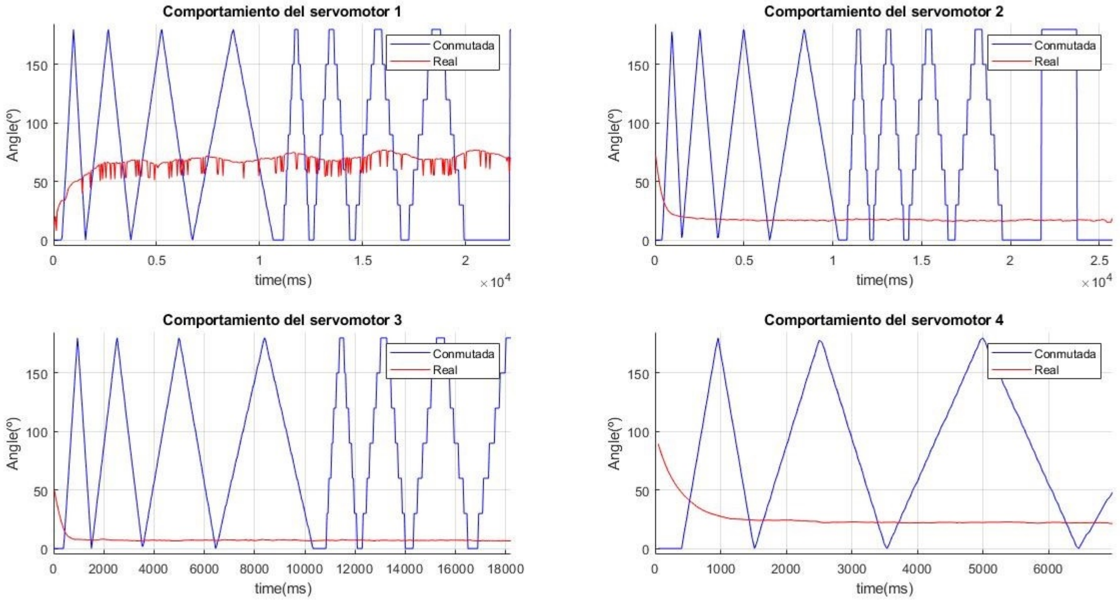


Figure 7.4.6. Failure of several microcontrollers in different moments

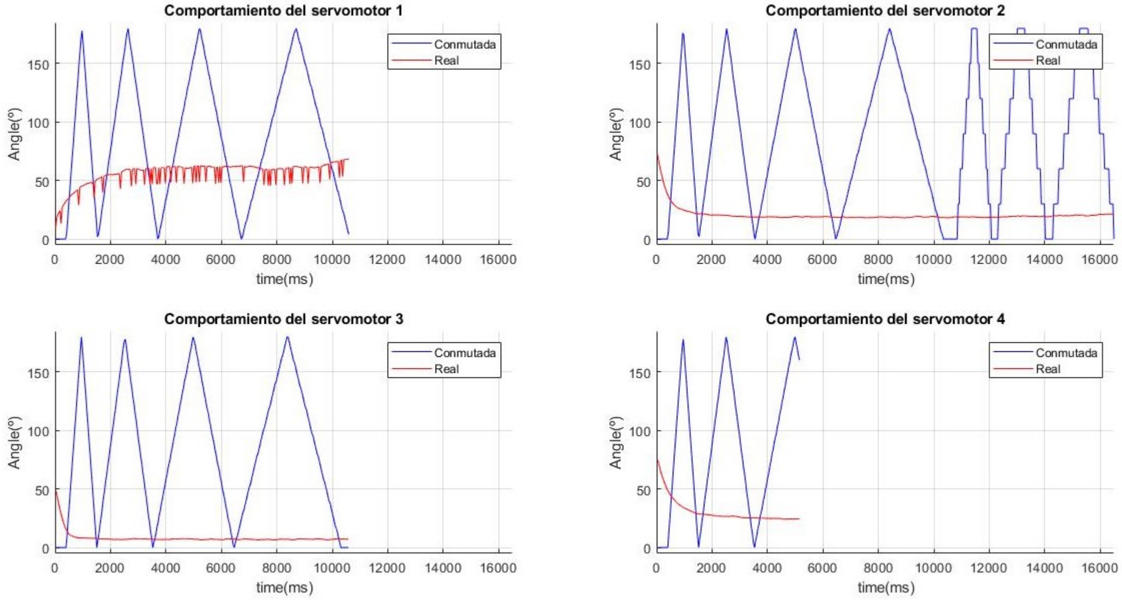


Figure 7.4.7. Completely signal lost failure

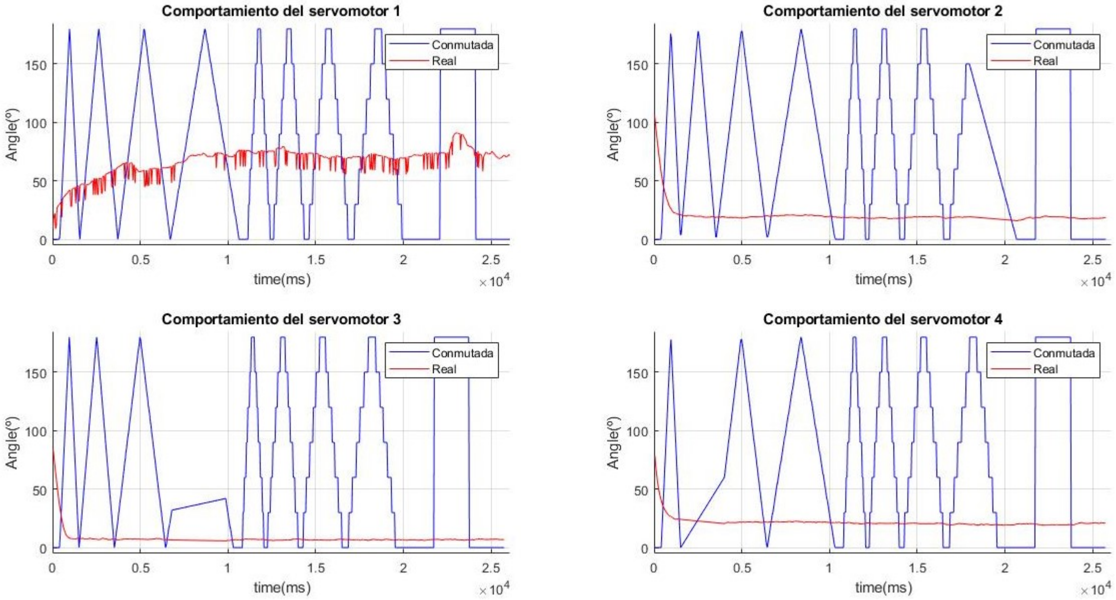


FIGURE 7.4.8. Temporally signal lost failure

Project evaluation

CHAPTER 8

Conclusions and future development lines

In this chapter, the actual project state and the future development lines are presented to continue with this project.

Objectives met.

During this project the following objectives have been satisfied:

- The mechatronic design of a robotic platform has been carried out according to the requirements and specifications.
  - Some decision has been taken, selecting the omnidirectional locomotion model and selecting the proper sensors.
  - The motor set has been chosen thanks to several equations of force, torque, power, inertial moments, etc. The motor, the gearhead, the encoder, and the controller have been selecting according to the results of the equations.
  - To guarantee the proper performance of the chosen motor set, a motor verification experiment has been carried out.
  - Two mechanical designs are presented, selecting the most compact after a stability study has been done.
  - An electronic and electrical design has been presented for the motor connexions.
- The control of the robotic platform has been developed.
  - The kinematic control has been calculated to move the robot.
  - The dynamic control has also been calculated to move the robot in the best way, this is, with gentle accelerations, guaranteeing a long life of the robot's materials.
- A graphSLAM approach has been presented to achieve a robot's self-location.
  - According to the requirements and specifications, a proposal system of self-location has been presented, formed by three main blocks.
  - The main procedure has been explained, detailing the performance of each block and the steps and relation between them.
  - The third block, the Graph Generator and Optimizer, has been specially explained. The graph structure has been presented to ease understanding.
  - The performance of the used libraries for the third block has been explained.

  - The SLAM algorithm follows a flowchart to work. It has been explained and detailed step by step, explaining each one as well as possible.
  - The results show how the robot location is improved regarding EKF output and regarding scan matching system output. Although odometry is very bad, the system is able to locate correctly in the environment. It is also possible to observe that the quality of the reconstructions is much better.
- A study of microcontrollers under radioactivity has been prepared, to find the more proper microcontroller to locate in robots that work in radioactive environments.
  - A high-level communication protocol has been created to communicate different microcontrollers through RS-485 low-level protocol.
  - An electronic scheme has been designed to connect all the components of the system.
  - The corresponding programs have been developed and uploaded to the microcontrollers.
  - Several tests have been carried out to check the performance of the whole system. The result is a functional system ready to be taken to a test lab.

Future development lines.

The following future development lines are presented:

- Development of a telescopic system that lifts the 3D LIDAR in the robot. This system must be down during the crossing of doors and it must be up during the normal performance.
- Implement the controllers (kinematic and dynamic) at the software level.
- Development of a system that detects the yellow doors of the accelerator. Since the position of these doors is well know, this system is very useful to correct the vertices position in the graph and, then, a good SLAM system can be got.
- SIFT, SURF, ORB, BRIEF, FAST and HARRIS must be tested. The performance of each descriptor must be compared and the best one must be selected to improve the odometry and the EKF.
- Respect to the SLAM system, there are some improvements to do:
  - With the RGBd camera, the scan matching system fails sometimes. This is caused due to the small range of the camera. This range produces points very located in a certain area of the environment and a moderate rotation movement or a large translation movement can cause a bad correspondence between PCs. For future development lines, it is strongly recommendable to use a 3D LIDAR with a large range and field of view or reduce the linear and angular velocity of the robot if the RGBd camera is used.

  - With the RGBd camera, it is important to check the angle difference between PCs. This point is especially important when the graphSLAM system compares PCs between the actual vertex and the nearest neighbors. Although the points are close, if the camera orientation is contrary, the RGBd camera field of view is completely different and then the scan matching system can not find a correspondence between PCs. For future development lines, it is strongly recommendable to develop a system that checks the estimated position of the PCs and, if there is much difference between the orientations, skip the comparison between them.
  - With the RGBd camera, it is known that the system can fail. In that case, usually, it fails due to the lack of correspondence between PCs. If this occurs in the first comparison (between the current position and the previous one), it is strongly recommendable to guess that the new position has suffered the same increase as the previous one.
- Concluding from the previous point, the most important future development line is to change the RGBd camera for the 3D LIDAR, which is expected to provides better results.
- Since the graphSLAM is expected to work efficiently according to the obtained results, a future development line consists of testing the entire system (EKF, scan matching, and graph generator and optimizer), which is expected to work in refined spaces with the RGBd camera and spacious environments with the 3D LIDAR.
- It can be useful to play with the information matrix, providing higher values to more reliable edges.
- Since all the solvers (linear and non-linear) have been implemented. A future development line consists in test all the solvers (when everything is integrated), to find the best solution. It is also required to determine the best value to 'k' when the system is looking for the nearest neighbors.

An economic study of the project is presented for the future development lines, where the complete cost is shown in Table 8.0.1, since this project beginning (six months ago), up to the estimated time when the project is expected to be finished. It is strongly recommendable to read previously in Appendix B..

Study type	U. Cost (€)	Future time (mos)	Time perc. (%)	Cost (€)
Material	26358	0	100	26358
Staff	8190	18	25	2047.5
Resource	1019	18	25	254.75
			Total cost (€)	28660.25

Table 8.0.1. The total cost to continue with the project and get a functional robot

Since the staff company schedule is not known, the estimation time to finish this project is two years since the project beginning.

Bibliography

[1] R. Nave. (2019, Sept 16). Radioactivity (not known ed.) [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Nuclear/radact.html 5

[2] KEK. (2019, Sept 17). KEK (not known ed.) [Online]. Available: https://www.kek.jp/en/

[3] CERN. (2019, Sept 21). CERN home (not known ed.) [Online]. Available: https://home.cern/ 2, 7

[4] Wikipedia Org. (2019, Sept 21). CERN (2019, Sept 15 edition). [Online]. Available: ht-tps://en.wikipedia.f/wiki/CERN 7

[5] CERN. (2019, Sept 23). CERN knowledge transfer (not known ed.) [Online]. Available: https://kt.cern/technologies/train-inspection-monorail-tim 11

[6] Telerob. (2019, Sept 23). Telemax PRO (2019 edition) [Online]. Available: https://www.telerob.com/en/products/telemax-family/telemax-pro 11

[7] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". 13

[8] G. Grisetti, R. Kümmerle, C. Stachniss and W. Burgard, "A Tutorial on Graph-Based SLAM". 13

[9] Wikipedia Org. (2020, Jan 06). GraphSLAM (2017, Dec 18 edition). [Online]. Available: ht-tps://en.wikipedia.org/wiki/GraphSLAM 13

[10] Arduino. (2019, June 20). Arduino as ISP and Arduino Bootloaders (2019 edition) [Online]. Available: https://www.arduino.cc/en/Tutorial/ArduinoISP 75

[11] Sparkfun. (2019. June 20). Pocket AVR Programmer Hookup Guide (2019 edition) [Online]. Available: https://learn.sparkfun.com/tutorials/pocket-avr-programmer-hookup-guide/all#pogo-pins 75

[12] Sparkfun. (2019. June 20). Installing an Arduino Bootloader (2019 edition) [Online]. Available: https://learn.sparkfun.com/tutorials/installing-an-arduino-bootloader/all 75

[13] "Mecatrónica", class notes for 42396, System engineering and automatic, Universidad de Valladolid, 2018. 22

[14] ‘Diseño y Control de Robots’, class notes for 53001562, System engineering and automatic, Universidad de Madrid, 2019. 25

[15] A. Sáenz, E. Bugarin and V. Santibáñez, "Modelado Cinemático y Dinámico de un Robot Móvil Omnidireccional de 4 Ruedas Considerando Dinámica de Actuadores", pp. 115-120. 39, 45

[16] Z. Hendzel and L. Rykala, "Modelling of dynamics of a wheeled mobile robot with mechanum wheels with the use of lagrange equations of the second kind", vol. 22, pp. 81-99. 39

[17] N.M. Adam, A. Irawan, M. R. Daud, Z. M. Zain and S. N. S. Ali, "Dynamic Modeling and Analysis of Omnidirectional Wheeled Robot: Turning Motion Analysis", vol. 10, pp. 103-108. 39

[18] F. P. Beer, E. R. J. Johnston, and E. R. Eisenberg, Mecánica Vectorial para Ingenierios: Dinámica. McGraw-Hill Interamericana, 2007. 41

[19] R. Kelly, V. Santibáñez and A. Loria, "Control of robot manipulators in joint space". Second Edition. Springer, London, U.K., 2005. 47, 48

[20] S. Villanueva, "Sistema de reconstruccion 3D y estimación de la posición de objetos en entornos no estructurados mediante cámaras RGB-D", Master thesis, CAR, UPM, Madrid, Spain, 2020. 50

[21] Tony Lindeberg. Scale invariant feature transform. Scholarpedia, 7(5):10491, 2012. 53

[22] J. Engel, V. Koltun and D. Cremers, "Direct Sparse Odometry," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 3, pp. 611-625, 1 March 2018. 53

[23] TUM (2020, Jan 21). DSO: Direct Sparse Odometry (2016, Nov 12 edition). [Online]. Available: https://vision.in.tum.de/research/vslam/dso 53

[24] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "G2o: A general framework for graph optimization," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 3607-3613. 55, 58, 59, 61

[25] Wikipedia Org. (2020, Jan 08). Least squares (2019, Dec 19 edition). [Online]. Available: https://en.wikipedia.org/wiki/Least_squares 58

[26] Wikipedia Org. (2020, Jan 09). Cholesky decomposition (2019, Dec 28 edition). [Online]. Available: https://en.wikipedia.org/wiki/Cholesky_decomposition 60

[27] Github. (2020, Jan 09). SuiteSparse (v.4.0.2 edition). [Online]. Available: https://github.com/PetterS/SuiteSparse/tree/master/CHOLMOD 60, 61

[28] Eigen. (2020, Jan 09). Eigen (v.3.3.7 edition). [Online]. Available: http://eigen.tuxfamily.org/index.php?title=Main_Page 61

[29] Wikipedia Org. (2020, Jan 09). Conjugate gradient method (2019, Dec 18 edition). [Online]. Available: https://en.wikipedia.org/wiki/Conjugate_gradient_method 61

[30] Alexandru and E. Ichim, "RGB-D handheld mapping and modeling", Master thesis, School of Computer and Communication, EPFL, Lausanne, Switzerland, 2013.

[31] PCL. (2020, Jan 10). How to use a KdTree to search (v.1.9.1 edition). [Online]. Available: http://pointclouds.org/documentation/tutorials/kdtree_search.php

61, 62

Appendices

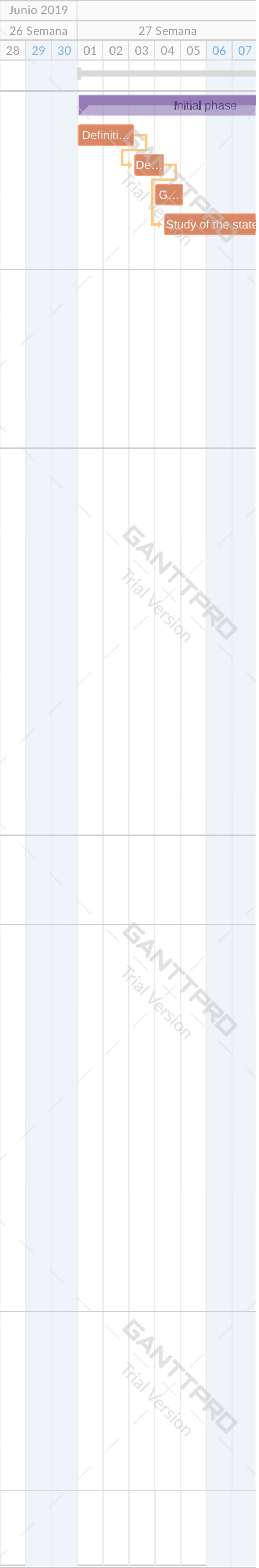
APPENDIX A

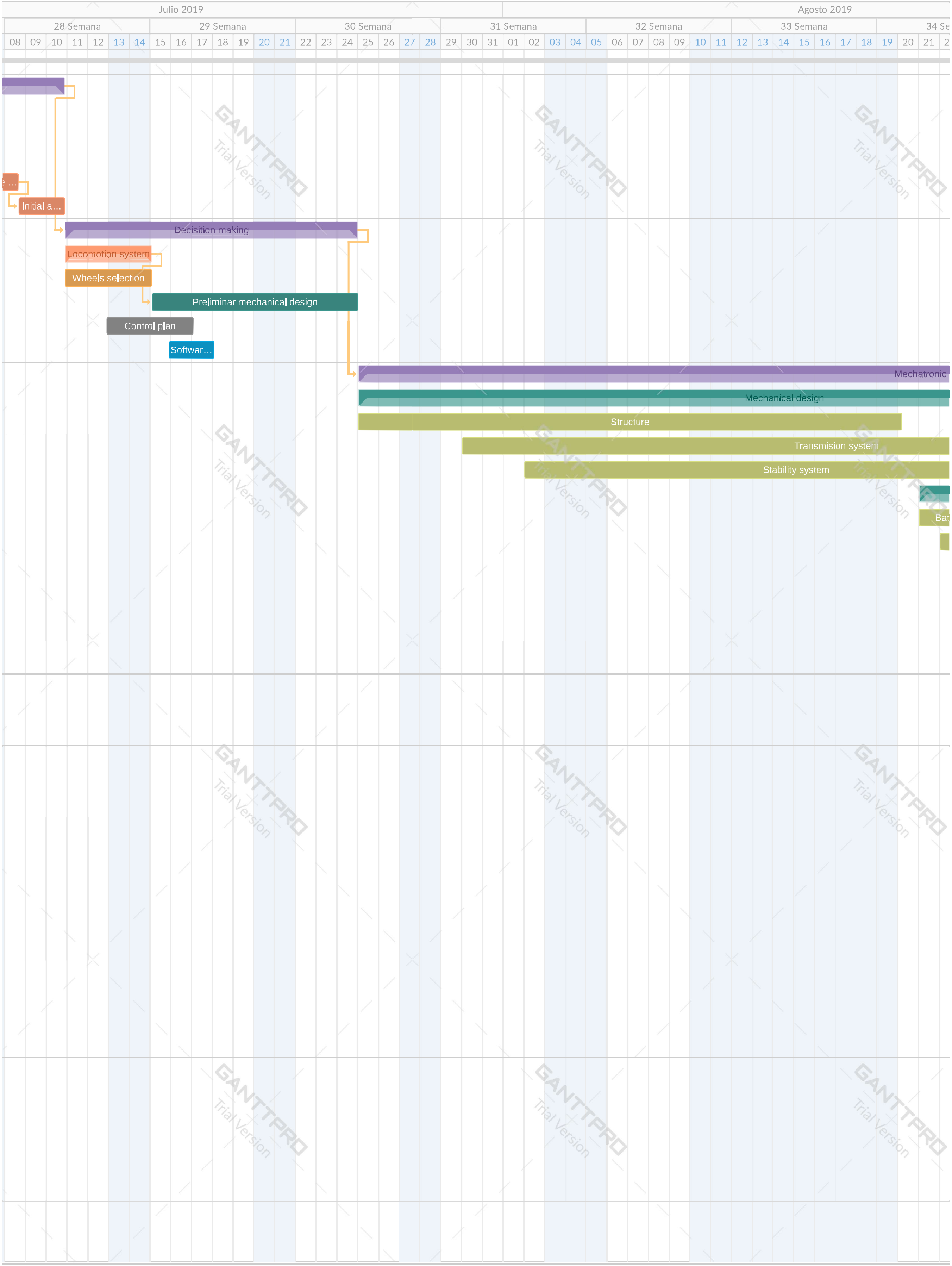
Project Gantt

GANTTPRO

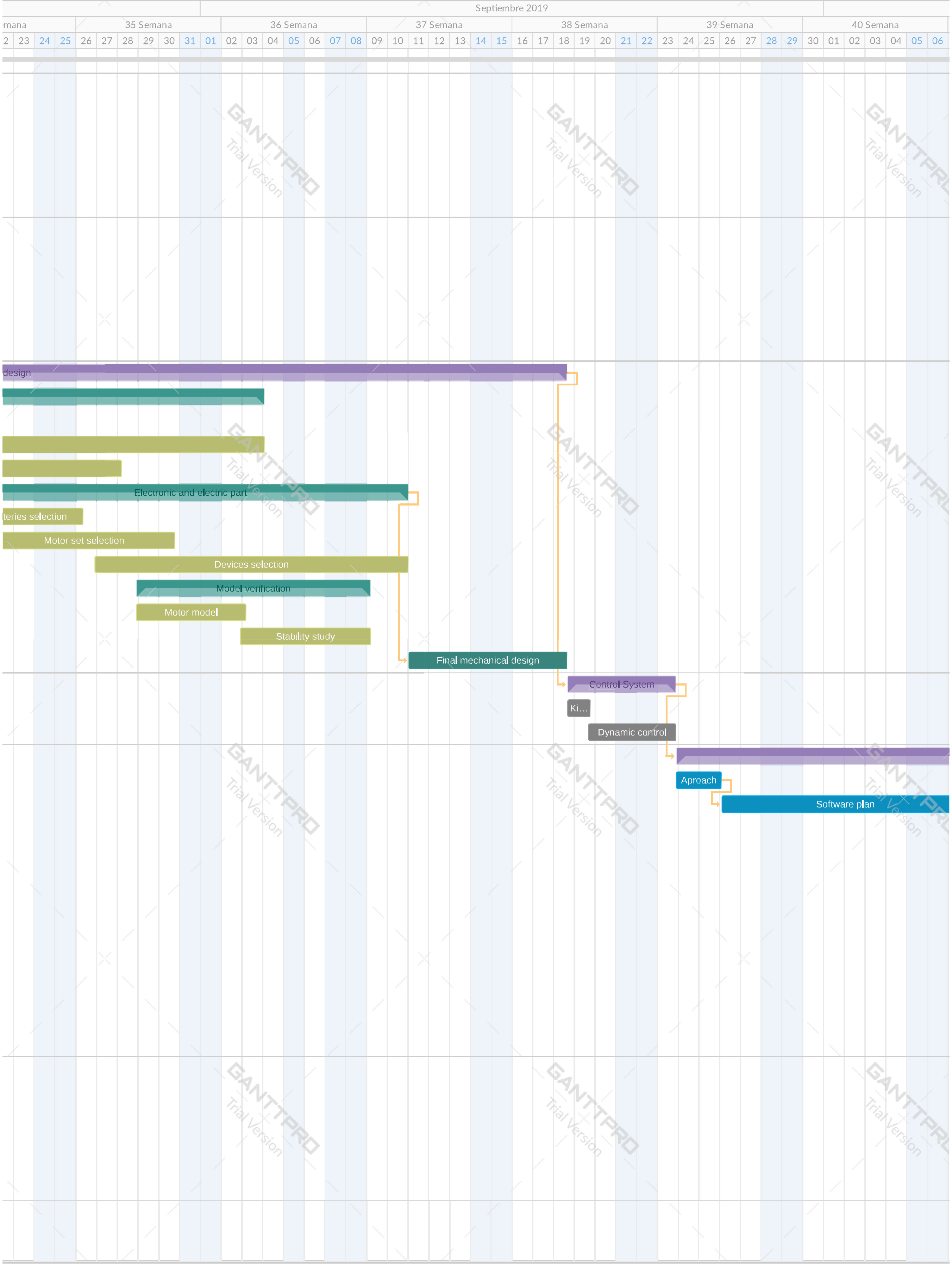
TFM

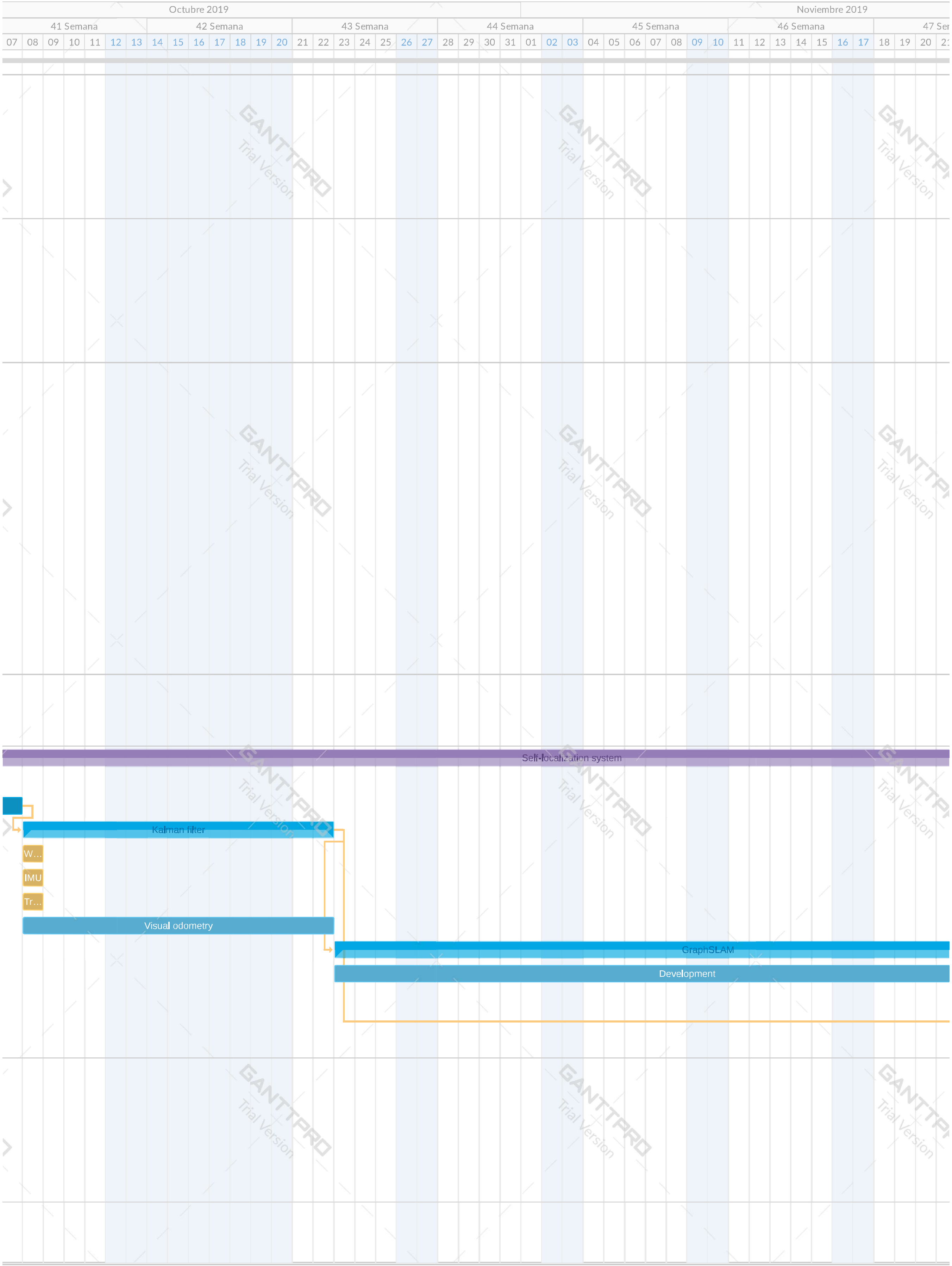
	Nombre de la tarea	Fecha de inic	Fecha final	Duración (Semana)	Estado	Prioridad
		01-07-2019	20-12-2019	22w 1d		
1	⊟ **Initial phase**	01-07-2019	10-07-2019	1w 2d 7h		
1.1	Definition of requirements and specifications	01-07-2019	03-07-2019	2d 2h	● Terminado	↑ Alto
1.2	Definition of objectives	03-07-2019	04-07-2019	1d 1h	● Terminado	↑ Medio
1.3	Gantt development	04-07-2019	05-07-2019	1d 1h	● Terminado	↓ Más bajo
1.4	Study of the state of art	04-07-2019	08-07-2019	2d 2h	● Terminado	↑ Medio
1.5	Initial aproach of the proyect	08-07-2019	10-07-2019	2d 2h	● Terminado	↓ Bajo
2	⊟ **Decisition making**	10-07-2019	24-07-2019	2w 1h		
2.1	⊟ **Locomotion system**	10-07-2019	15-07-2019	2d 2h		
2.1.1	Wheels selection	10-07-2019	15-07-2019	2d 2h	● Terminado	↑ Medio
2.2	Preliminar mechanical design	15-07-2019	24-07-2019	1w 2d 7h	● Terminado	↑ Alto
2.3	Control plan	12-07-2019	17-07-2019	2d 2h	● Terminado	↓ Bajo
2.4	Software initial plan	15-07-2019	18-07-2019	2d 2h	● Terminado	↓ Bajo
3	⊟ **Mechatronic design**	25-07-2019	18-09-2019	6w 1d 5h		
3.1	⊟ **Mechanical design**	25-07-2019	04-09-2019	4w 2d 1h		
3.1.1	Structure	25-07-2019	20-08-2019	2w 1d 2h	● Terminado	↑ Más alto
3.1.2	Transmision system	30-07-2019	04-09-2019	3w 4d 1h	● Terminado	↑ Más alto
3.1.3	Stability system	02-08-2019	28-08-2019	2w 1d 2h	● Terminado	↑ Medio
3.2	⊟ **Electronic and electric part**	21-08-2019	10-09-2019	2w 4d		
3.2.1	Batteries selection	21-08-2019	26-08-2019	3d 3h	● Terminado	↑ Alto
3.2.2	Motor set selection	22-08-2019	30-08-2019	1w 1d 6h	● Terminado	↑ Más alto
3.2.3	Devices selection	26-08-2019	10-09-2019	2w 1h	● Terminado	↑ Alto
3.3	⊟ **Model verification**	28-08-2019	09-09-2019	1w 1d 3h		
3.3.1	Motor model	28-08-2019	03-09-2019	3d 3h	● Terminado	↑ Medio
3.3.2	Stability study	02-09-2019	09-09-2019	3d 3h	● Terminado	↑ Alto
3.4	Final mechanical design	11-09-2019	18-09-2019	1w 5h	● Terminado	↑ Más alto
4	⊟ **Control System**	18-09-2019	23-09-2019	3d 2h		
4.1	Kinematic control	18-09-2019	19-09-2019	1d 1h	● Terminado	↑ Medio
4.2	Dynamic control	19-09-2019	23-09-2019	2d 2h	● Terminado	↑ Medio
5	⊟ **Self-localization system**	23-09-2019	13-12-2019	10w 3d 1h		
5.1	Aproach	23-09-2019	26-09-2019	2d 2h	● Terminado	↑ Alto
5.2	Software plan	26-09-2019	07-10-2019	1w 2d 7h	● Terminado	↑ Medio
5.3	⊟ **Kalman filter**	08-10-2019	22-10-2019	1w 1d		
5.3.1	Wheels odometry	08-10-2019	08-10-2019	1d	● Terminado	↓ Más bajo
5.3.2	IMU	08-10-2019	08-10-2019	1d	● Terminado	↓ Más bajo
5.3.3	Tracking camera	08-10-2019	08-10-2019	1d	● Terminado	↓ Más bajo
5.3.4	Visual odometry	08-10-2019	22-10-2019	1w 1d	● Cerrado	↑ Medio
5.4	⊟ **GraphSLAM**	23-10-2019	27-11-2019	5w 1d		
5.4.1	Development	23-10-2019	25-11-2019	4w 4d	● Terminado	↑ Más alto
5.4.2	Tests and adjustment	26-11-2019	27-11-2019	2d	● Terminado	↑ Alto
5.5	Kalman filter and graphSLAM Integration	28-11-2019	05-12-2019	1w 1d	● Terminado	↑ Más alto
5.6	Tests in a robotic system	09-12-2019	13-12-2019	1w	● Terminado	↑ Alto
6	⊟ **Final phase**	16-12-2019	20-12-2019	1w		
6.1	⊟ **Software error treatment**	16-12-2019	18-12-2019	3d		
6.1.1	Code tests development	16-12-2019	17-12-2019	2d	● Terminado	↑ Alto
6.1.2	Exceptions tests	17-12-2019	18-12-2019	2d	● Terminado	↑ Más alto
6.2	Documentation	19-12-2019	20-12-2019	2d	● Terminado	↑ Medio
6.3	Future lines of development	19-12-2019	20-12-2019	2d	● Terminado	↑ Medio

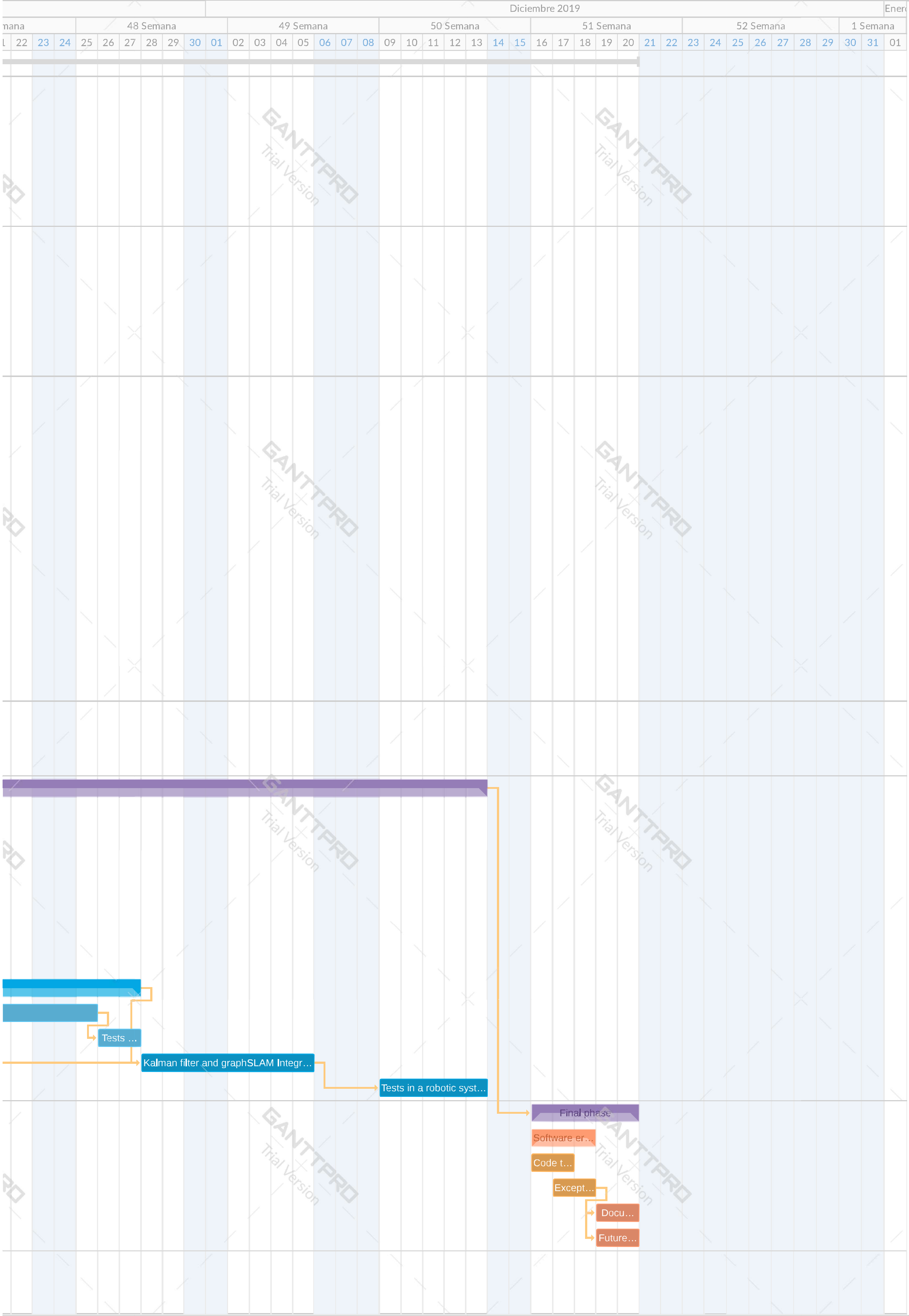




Julio 2019
Agosto 2019
28 Semana
29 Semana
30 Semana
31 Semana
32 Semana
33 Semana
34 Se
08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 2
e ...
Initial a...
Decisition making
Locomotion system
Wheels selection
Preliminar mechanical design
Control plan
Softwar...
Mechatronic
Mechanical design
Structure
Transmision system
Stability system
Bat


Septiembre 2019
mana
35 Semana
36 Semana
37 Semana
38 Semana
39 Semana
40 Semana
2 23 24 25 26 27 28 29 30 31 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 01 02 03 04 05 06
design
Electronic and electric part
teries selection
Motor set selection
Devices selection
Model verification
Motor model
Stability study
Final mechanical design
Control System
Ki...
Dynamic control
Aproach
Software plan


Octubre 2019
Noviembre 2019
41 Semana
42 Semana
43 Semana
44 Semana
45 Semana
46 Semana
47 Ser
07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 2
Self-localization system
Kalman filter
W...
IMU
Tr...
Visual odometry
GraphSLAM
Development


Diciembre 2019
Enero
mana
48 Semana
49 Semana
50 Semana
51 Semana
52 Semana
1 Semana
22 23 24 25 26 27 28 29 30 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 01
Tests ...
Kalman filter and graphSLAM Integr…
Tests in a robotic syst…
Final phase
Software er…
Code t…
Except…
Docu…
Future…

APPENDIX B

Economical study

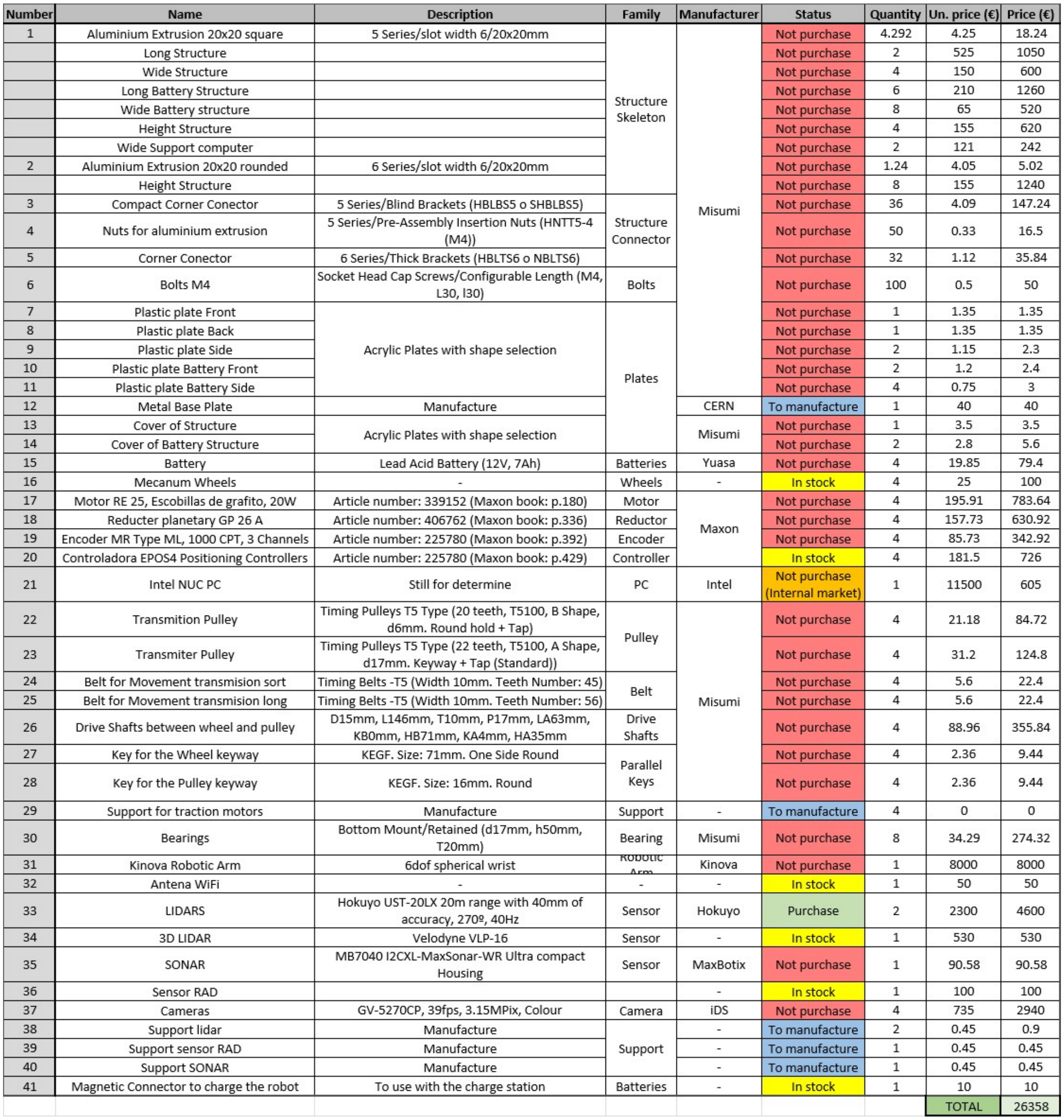
The economical study has been split into three different parts: material costs study for the robot construction, staff costs study and resources cost study.

Firstly, a **material cost study** has been carried out. Here, the necessary material equipment has been included, as shown in Table B.0.1. Since most of the items are pending to be purchased to build the robot, this study is not the one corresponding to these months, but it is useful to know the cost of robot materials.

In the second place, a **staff costs study** has been carried out. Here, the costs related to the salary has been included, as shown in Table B.0.2.

In third place, a **resource costs study** has been carried out. Here, all the costs related to the used materials to work, energy, tools, etc., have been included, as shown in Table B.0.3.

The sum of all the costs is reflected in Table B.0.4, where the cost relative to these months are detailed.



Number	Name	Description	Family	Manufacturer	Status	Quantity	Un. price (€)	Price (€)
1	Aluminium Extrusion 20x20 square	5 Series/slot width 6/20x20mm	Structure Skeleton	Misumi	Not purchase	4.292	4.25	18.24
	Long Structure				Not purchase	2	525	1050
	Wide Structure				Not purchase	4	150	600
	Long Battery Structure				Not purchase	6	210	1260
	Wide Battery structure				Not purchase	8	65	520
	Height Structure				Not purchase	4	155	620
	Wide Support computer				Not purchase	2	121	242
2	Aluminium Extrusion 20x20 rounded	6 Series/slot width 6/20x20mm			Not purchase	1.24	4.05	5.02
	Height Structure				Not purchase	8	155	1240
3	Compact Corner Conector	5 Series/Blind Brackets (HBLBS5 o SHBLBS5)	Structure Connector		Not purchase	36	4.09	147.24
4	Nuts for aluminium extrusion	5 Series/Pre-Assembly Insertion Nuts (HNTT5-4 (M4))			Not purchase	50	0.33	16.5
5	Corner Conector	6 Series/Thick Brackets (HBLTS6 o NBLTS6)			Not purchase	32	1.12	35.84
6	Bolts M4	Socket Head Cap Screws/Configurable Length (M4, L30, l30)	Bolts		Not purchase	100	0.5	50
7	Plastic plate Front	Acrylic Plates with shape selection	Plates		Not purchase	1	1.35	1.35
8	Plastic plate Back				Not purchase	1	1.35	1.35
9	Plastic plate Side				Not purchase	2	1.15	2.3
10	Plastic plate Battery Front				Not purchase	2	1.2	2.4
11	Plastic plate Battery Side				Not purchase	4	0.75	3
12	Metal Base Plate	Manufacture		CERN	To manufacture	1	40	40
13	Cover of Structure	Acrylic Plates with shape selection		Misumi	Not purchase	1	3.5	3.5
14	Cover of Battery Structure				Not purchase	2	2.8	5.6
15	Battery	Lead Acid Battery (12V, 7Ah)	Batteries	Yuasa	Not purchase	4	19.85	79.4
16	Mecanum Wheels	-	Wheels	-	In stock	4	25	100
17	Motor RE 25, Escobillas de grafito, 20W	Article number: 339152 (Maxon book: p.180)	Motor	Maxon	Not purchase	4	195.91	783.64
18	Reducter planetary GP 26 A	Article number: 406762 (Maxon book: p.336)	Reductor		Not purchase	4	157.73	630.92
19	Encoder MR Type ML, 1000 CPT, 3 Channels	Article number: 225780 (Maxon book: p.392)	Encoder		Not purchase	4	85.73	342.92
20	Controladora EPOS4 Positioning Controllers	Article number: 225780 (Maxon book: p.429)	Controller		In stock	4	181.5	726
21	Intel NUC PC	Still for determine	PC	Intel	Not purchase (Internal market)	1	11500	605
22	Transmition Pulley	Timing Pulleys T5 Type (20 teeth, T5100, B Shape, d6mm. Round hold + Tap)	Pulley	Misumi	Not purchase	4	21.18	84.72
23	Transmiter Pulley	Timing Pulleys T5 Type (22 teeth, T5100, A Shape, d17mm. Keyway + Tap (Standard))			Not purchase	4	31.2	124.8
24	Belt for Movement transmision sort	Timing Belts -T5 (Width 10mm. Teeth Number: 45)	Belt		Not purchase	4	5.6	22.4
25	Belt for Movement transmision long	Timing Belts -T5 (Width 10mm. Teeth Number: 56)			Not purchase	4	5.6	22.4
26	Drive Shafts between wheel and pulley	D15mm, L146mm, T10mm, P17mm, LA63mm, KB0mm, HB71mm, KA4mm, HA35mm	Drive Shafts		Not purchase	4	88.96	355.84
27	Key for the Wheel keyway	KEGF. Size: 71mm. One Side Round	Parallel Keys		Not purchase	4	2.36	9.44
28	Key for the Pulley keyway	KEGF. Size: 16mm. Round			Not purchase	4	2.36	9.44
29	Support for traction motors	Manufacture	Support	-	To manufacture	4	0	0
30	Bearings	Bottom Mount/Retained (d17mm, h50mm, T20mm)	Bearing	Misumi	Not purchase	8	34.29	274.32
31	Kinova Robotic Arm	6dof spherical wrist	Robotic Arm	Kinova	Not purchase	1	8000	8000
32	Antena WiFi	-	-	-	In stock	1	50	50
33	LIDARS	Hokuyo UST-20LX 20m range with 40mm of accuracy, 270º, 40Hz	Sensor	Hokuyo	Purchase	2	2300	4600
34	3D LIDAR	Velodyne VLP-16	Sensor	-	In stock	1	530	530
35	SONAR	MB7040 I2CXL-MaxSonar-WR Ultra compact Housing	Sensor	MaxBotix	Not purchase	1	90.58	90.58
36	Sensor RAD			-	In stock	1	100	100
37	Cameras	GV-5270CP, 39fps, 3.15MPix, Colour	Camera	iDS	Not purchase	4	735	2940
38	Support lidar	Manufacture	Support	-	To manufacture	2	0.45	0.9
39	Support sensor RAD	Manufacture		-	To manufacture	1	0.45	0.45
40	Support SONAR	Manufacture		-	To manufacture	1	0.45	0.45
41	Magnetic Connector to charge the robot	To use with the charge station	Batteries	-	In stock	1	10	10
							TOTAL	26358

Table B.0.1. Material cost study

Monthly salary (€)	Number of months	Total salary (€)
1365	6	8190

Table B.0.2. Staff costs study

Resource	Type	Used (mos)	Life (y)	Used (%)	Qty.	U. cost (€)	Cost (€)
Computer	Material	6	8	6.25	1	800	50
Screen		6	10	5	2	50	5
Mouse		6	10	5	1	10	0.5
Keyboard		6	15	3.33	1	15	0.5
CHARMbot		2	20	0.83	1	30000	250
Tools		4	20	1.67	1	300	5
RGBd cam.		2	10	5	1	150	7.5
Light	Energy and water	6				55	330
Water		6				8	48
Heater		6				40	240
Accelerator	Maintenance	1				15	30
Office		6				10	60
						Total (€)	1026.5

Table B.0.3. Resource costs study

Study type	Cost (€)
Material	4600
Staff	8190
Resource	1019
Total	13809

Table B.0.4. The total cost of the project until now

Another economic study has been carried out in Chapter 8, where the future development lines are explained, including the cost to continue with this project.

APPENDIX C

Methodology to program a microcontroller with an external programmer

This Appendix shows how to program a microcontroller without a programmer microcontroller. Instead, the AVR Pocket Programmer has been used to program an AVR microcontroller, shown in Figure C.0.1, where it is possible to appreciate the most important components.

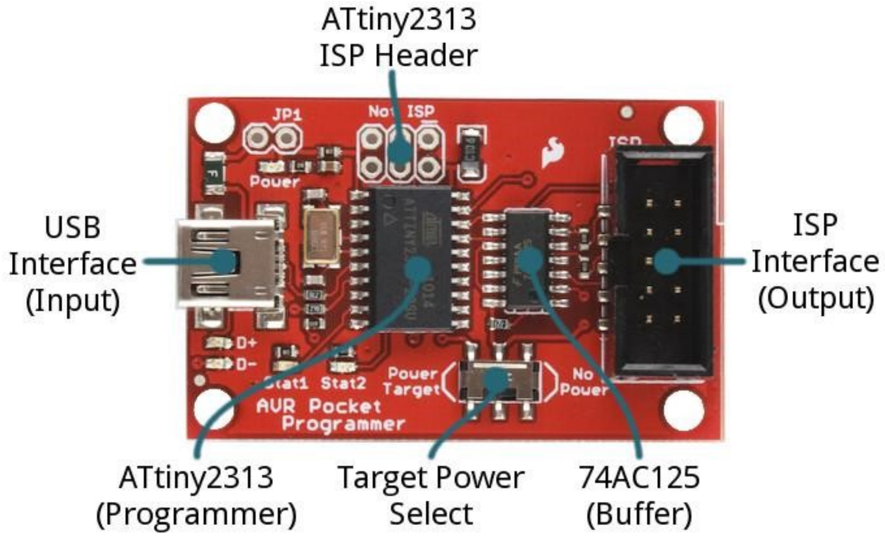


FIGURE C.0.1. Components of AVR Pocket Programmer

The function of each component is:

- USB Interface: Data and power input (from a computer).
- ISP Interface: Output to send the programming signals to AVR.
- Target Power Select: Switch that allows selecting the power supply of the AVR microcontroller (from the programmer or with its power).
- ATtiny2313: Programmer chip. Relation of the computer commands (input) with the programming commands (output).
- 74AC123: Protection of the programmer chip, storing the previous programming data (they are reset in each programming step).

In this Appendix, two different cases of programming are shown. The first case would be the fact of the program an Arduino microcontroller (Figure C.0.2). It can be useful in the case that the programming microcontroller is spoiled or to practice to program another microcontroller.

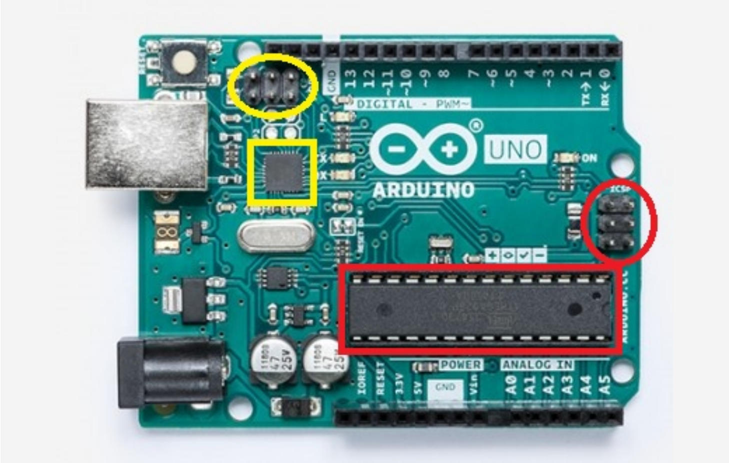


Figure C.0.2. Arduino Uno Rev3.
Yellow square: Programming microcontroller. Yellow ellipse: Pinout for the communication with the programming microcontroller. Red square: Logic microcontroller. Red ellipse: Pinout for the communication with the logic microcontroller.

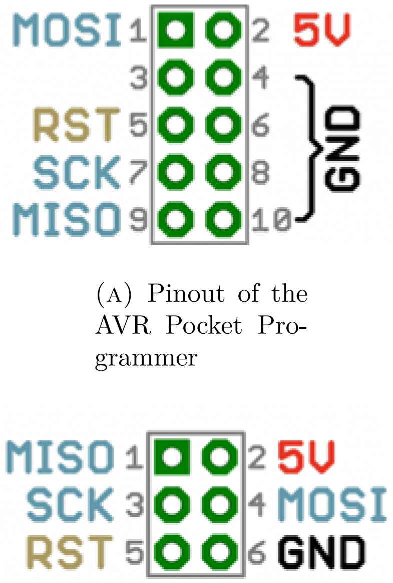
In both cases, the programming methodology is through SPI[26], which has four pins:

- SCK: Clock signal that indicates the transmission velocity.
- MISO: Master Input Slave Output. Data is received from the other integrated circuit (from the programmer to the AVR).
- MOSI: Master Output Slave Input. Data is sent to the other integrated circuit (from the AVR to the programmer).
- RST (SS, Slave Select, o CS, Chip Select): Enable the integrated circuit to which the data is sent.

Delving into the **first case**, Arduino boards dispose of internal connections between the logic microcontroller pinouts and the board pinouts. In this way, it is possible to communicate with the AVR and program it.

So several programming signals join both devices (AVR microcontroller and programmer). These constraints can be observed in Figure C.0.3.

[26] SPI: Serial Peripheral Interface. Common bus interface used to send data between microcontroller and peripherals. It uses a clock (synchronous solution), a device selector and data lines.



(A) Pinout of the AVR Pocket Programmer

(B) Pinout of Arduino board for the programming

Figure C.0.3. Pinout of each device

The supply voltage is transmitted if the board Switch is activated, otherwise, the supply voltage is not connected to the target (the board Switch must not be connected in the case that the microcontroller works with 3V, otherwise it will be damaged). In both cases, it is high recommendable to use the own power supply of the microcontrollers.

In the **second case**, a PCB must be developed and manufactured, to be able to access to all the pins we want to use. In this case, regarding Figure 7.1.1, where the microcontroller ATmega64M1 pinout is shown, it is possible to find the following useful pins for the SPI communication (shown in Figure C.0.4):

- Pin 8 (PB0): MISO.
- Pin 9 (PB1): MOSI.
- Pin 28 (PB7): SCK.

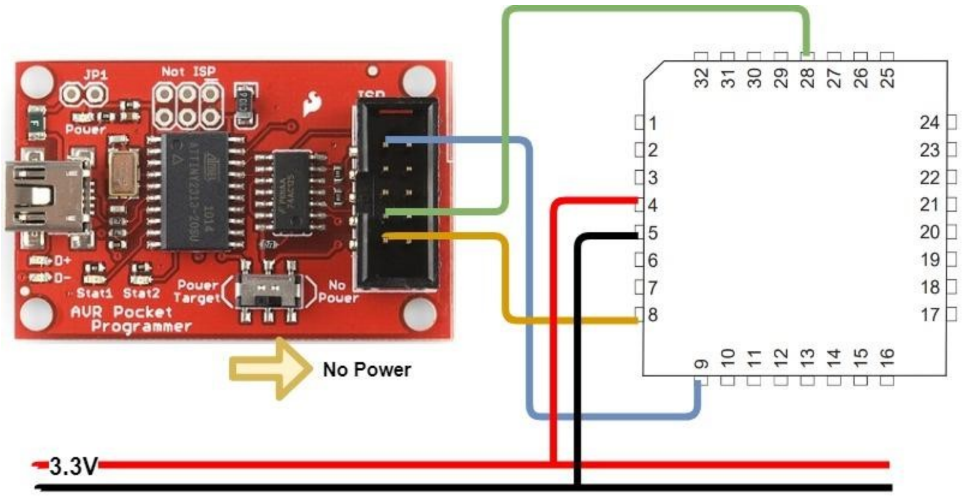


FIGURE C.0.4. The necessary connection between the programmer and the microcontroller

Then, to go forward and program the microcontroller or the Arduino microcontroller (both cases) using the SDK[27] of Arduino, follow the next steps is necessary:

- Perform the connections shown in Figure C.0.4.
- Switch off the Arduino or the microcontroller of the power supply.
- Supply it with the power supply of the programmer or through an external power supply of 3.3V, how is indicated in Figure C.0.4.
- Stablish the programmer way in Arduino IDLE like: Tools -> Programmer -> USBtinyISP.
- Upload the program: Program -> Upload using programmer.
- Disconnect the programmer.

To recover the default way to program Arduino, with the programmer connected, it must be selected: Tools -> Burn Bootloader, and the protocol must be restarted: Tools -> Programmer -> AVRISP mkll.

[27] SDK: A Software Development Kit is a collection of software development tools in one installable package.

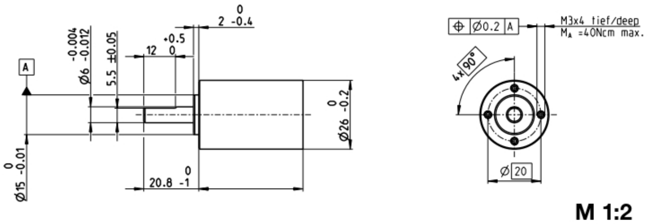
APPENDIX D

Datasheets

D.1. Reducer

maxon gear

Planetary Gearhead GP 26 A ∅26 mm, 0.75–4.5 Nm

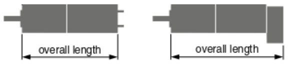


M 1:2

Technical Data		
Planetary Gearhead		straight teeth
Output shaft		stainless steel, hardened
Bearing at output		preloaded ball bearings
Radial play, 5 mm from flange		max. 0.1 mm
Axial play at axial load	< 6 N	0 mm
	> 6 N	max. 0.4 mm
Max. axial load (dynamic)		120 N
Max. force for press fits		120 N
Direction of rotation, drive to output		=
Max. continuous input speed		8000 rpm
Recommended temperature range		-30…+100°C
Extended range as option		-40…+100°C
Number of stages	1 / 2	3
Max. radial load, 12 mm from flange	70 N / 110 N	140 N

🟥 Stock program
□ Standard program
▨ Special program (on request)

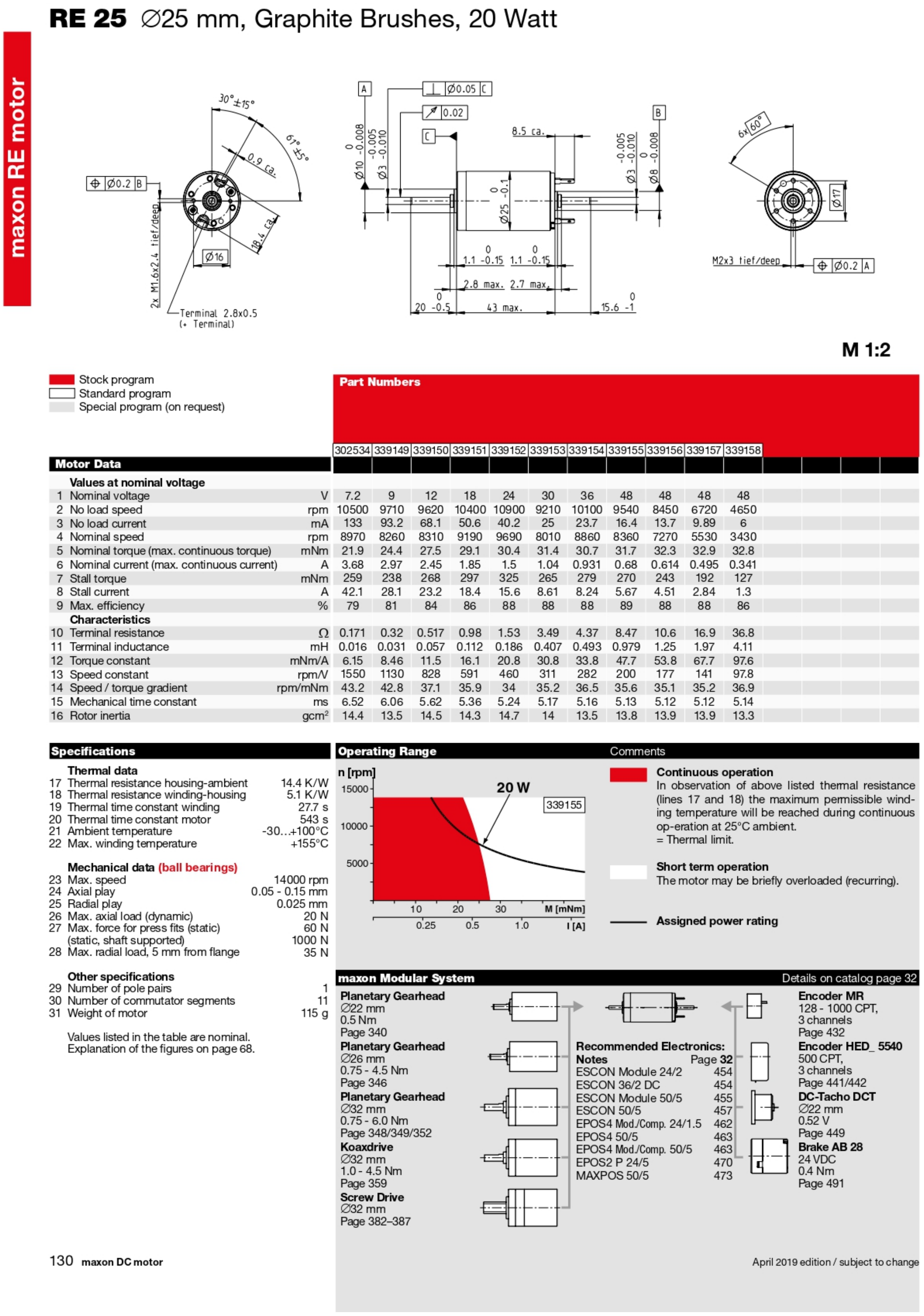
	Gearhead Data		406757	406762	406764	406767	406128	406769	406770	406771	406092
1	Reduction		5.2 : 1	19 : 1	27 : 1	35 : 1	71 : 1	100 : 1	139 : 1	181 : 1	236 : 1
2	Absolute reduction		57/11	3591/187	3249/121	1539/44	226233/3179	204687/2057	185193/1331	87723/484	41553/176
3	Max. motor shaft diameter	mm	3	3	3	3	3	3	3	3	3
4	Number of stages		1	2	2	2	3	3	3	3	3
5	Max. continuous torque	Nm	0.75	2.25	2.25	2.25	4.5	4.5	4.5	4.5	4.5
6	Max. intermittent torque at gear output	Nm	1.1	3.2	3.2	3.2	6.2	6.2	6.2	6.2	6.2
7	Max. efficiency	%	90	80	80	80	70	70	70	70	70
8	Weight	g	53	77	77	77	93	93	93	93	93
9	Average backlash no load	°	0.5	0.7	0.7	0.7	0.8	0.8	0.8	0.8	0.8
10	Mass inertia	gcm²	0.96	0.54	0.54	0.54	0.31	0.31	0.31	0.31	0.31
11	Gearhead length L1	mm	23.4	32.9	32.9	32.9	39.5	39.5	39.5	39.5	39.5
13	Max. transmittable power (continuous)	W	60	35	35	35	20	20	20	20	20
14	Max. transmittable power (intermittent)	W	90	50	50	50	30	30	30	30	30



maxon Modular System

+ Motor	Page	+ Sensor/Brake	Page	Overall length [mm] = Motor length + gearhead length + (sensor/brake) + assembly parts								
RE 25	125/127			78.0	87.5	87.5	87.5	94.1	94.1	94.1	94.1	94.1
RE 25	125/127	MR	419	89.0	98.5	98.5	98.5	105.1	105.1	105.1	105.1	105.1
RE 25	125/127	Enc 22	426	92.1	101.6	101.6	101.6	108.2	108.2	108.2	108.2	108.2
RE 25	125/127	HED_ 5540	429/431	98.8	108.3	108.3	108.3	114.9	114.9	114.9	114.9	114.9
RE 25	125/127	DCT22	438	100.3	109.8	109.8	109.8	116.4	116.4	116.4	116.4	116.4
RE 25, 20 W	126			66.5	76.0	76.0	76.0	82.6	82.6	82.6	82.6	82.6
RE 25, 20 W	126	MR	419	77.5	87.0	87.0	87.0	93.6	93.6	93.6	93.6	93.6
RE 25, 20 W	126	HED_ 5540	430	87.3	96.8	96.8	96.8	103.4	103.4	103.4	103.4	103.4
RE 25, 20 W	126	DCT 22	438	88.8	98.3	98.3	98.3	104.9	104.9	104.9	104.9	104.9
RE 25, 20 W	126	AB 28	480	100.6	110.1	110.1	110.1	116.7	116.7	116.7	116.7	116.7
RE 25, 20 W	126	HED_5540/AB 28	430/480	117.8	127.3	127.3	127.3	133.9	133.9	133.9	133.9	133.9
RE 25, 20 W	127	AB 28	480	112.1	121.6	121.6	121.6	128.2	128.2	128.2	128.2	128.2
RE 25, 20 W	127	HED_ 5540/AB 28	431/480	129.3	138.8	138.8	138.8	145.4	145.4	145.4	145.4	145.4
A-max 26	151-158			68.2	77.7	77.7	77.7	84.3	84.3	84.3	84.3	84.3
A-max 26	151-158	MEnc 13	408	75.3	84.8	84.8	84.8	91.4	91.4	91.4	91.4	91.4
A-max 26	151-158	MR	419	77.0	86.5	86.5	86.5	93.1	93.1	93.1	93.1	93.1
A-max 26	151-158	Enc 22	426	82.6	92.1	92.1	92.1	98.7	98.7	98.7	98.7	98.7
A-max 26	151-158	HED_ 5540	430/432	86.6	96.1	96.1	96.1	102.7	102.7	102.7	102.7	102.7

D.2. Motor


maxon RE motor
30° ±15°
61° ±5°
0.9 ca.
⌖ ∅0.2 B
18.4 ca.
∅16
2x M1.6x2.4 tief/deep
Terminal 2.8x0.5
(+ Terminal)
A
⊥ ∅0.05 C
↗ 0.02
C
B
∅10 0 -0.008
∅3 -0.005 -0.010
8.5 ca.
∅25 0 -0.1
∅3 -0.005 -0.010
∅8 0 -0.008
1.1 0 -0.15
1.1 0 -0.15
2.8 max.
2.7 max.
20 0 -0.5
43 max.
15.6 0 -1
6x 60°
∅17
M2x3 tief/deep
⌖ ∅0.2 A
M 1:2
n [rpm]
15000
10000
5000
20 W
339155
10
20
30
M [mNm]
0.25
0.5
1.0
I [A]

RE 25 ∅25 mm, Graphite Brushes, 20 Watt

🟥 Stock program
□ Standard program
▨ Special program (on request)

Part Numbers

Motor Data			302534	339149	339150	339151	339152	339153	339154	339155	339156	339157	339158
	Values at nominal voltage												
1	Nominal voltage	V	7.2	9	12	18	24	30	36	48	48	48	48
2	No load speed	rpm	10500	9710	9620	10400	10900	9210	10100	9540	8450	6720	4650
3	No load current	mA	133	93.2	68.1	50.6	40.2	25	23.7	16.4	13.7	9.89	6
4	Nominal speed	rpm	8970	8260	8310	9190	9690	8010	8860	8360	7270	5530	3430
5	Nominal torque (max. continuous torque)	mNm	21.9	24.4	27.5	29.1	30.4	31.4	30.7	31.7	32.3	32.9	32.8
6	Nominal current (max. continuous current)	A	3.68	2.97	2.45	1.85	1.5	1.04	0.931	0.68	0.614	0.495	0.341
7	Stall torque	mNm	259	238	268	297	325	265	279	270	243	192	127
8	Stall current	A	42.1	28.1	23.2	18.4	15.6	8.61	8.24	5.67	4.51	2.84	1.3
9	Max. efficiency	%	79	81	84	86	88	88	88	89	88	88	86
	Characteristics												
10	Terminal resistance	Ω	0.171	0.32	0.517	0.98	1.53	3.49	4.37	8.47	10.6	16.9	36.8
11	Terminal inductance	mH	0.016	0.031	0.057	0.112	0.186	0.407	0.493	0.979	1.25	1.97	4.11
12	Torque constant	mNm/A	6.15	8.46	11.5	16.1	20.8	30.8	33.8	47.7	53.8	67.7	97.6
13	Speed constant	rpm/V	1550	1130	828	591	460	311	282	200	177	141	97.8
14	Speed / torque gradient	rpm/mNm	43.2	42.8	37.1	35.9	34	35.2	36.5	35.6	35.1	35.2	36.9
15	Mechanical time constant	ms	6.52	6.06	5.62	5.36	5.24	5.17	5.16	5.13	5.12	5.12	5.14
16	Rotor inertia	gcm²	14.4	13.5	14.5	14.3	14.7	14	13.5	13.8	13.9	13.9	13.3

Specifications

	Thermal data	
17	Thermal resistance housing-ambient	14.4 K/W
18	Thermal resistance winding-housing	5.1 K/W
19	Thermal time constant winding	27.7 s
20	Thermal time constant motor	543 s
21	Ambient temperature	-30...+100°C
22	Max. winding temperature	+155°C
	Mechanical data (ball bearings)	
23	Max. speed	14000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	20 N
27	Max. force for press fits (static)	60 N
	(static, shaft supported)	1000 N
28	Max. radial load, 5 mm from flange	35 N
	Other specifications	
29	Number of pole pairs	1
30	Number of commutator segments	11
31	Weight of motor	115 g

Values listed in the table are nominal.
Explanation of the figures on page 68.

Operating Range

Comments

🟥 **Continuous operation**
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous op-eration at 25°C ambient.
= Thermal limit.

□ **Short term operation**
The motor may be briefly overloaded (recurring).

— **Assigned power rating**

maxon Modular System Details on catalog page 32

Planetary Gearhead
∅22 mm
0.5 Nm
Page 340

Planetary Gearhead
∅26 mm
0.75 - 4.5 Nm
Page 346

Planetary Gearhead
∅32 mm
0.75 - 6.0 Nm
Page 348/349/352

Koaxdrive
∅32 mm
1.0 - 4.5 Nm
Page 359

Screw Drive
∅32 mm
Page 382–387

Recommended Electronics:	
Notes	Page **32**
ESCON Module 24/2	454
ESCON 36/2 DC	454
ESCON Module 50/5	455
ESCON 50/5	457
EPOS4 Mod./Comp. 24/1.5	462
EPOS4 50/5	463
EPOS4 Mod./Comp. 50/5	463
EPOS2 P 24/5	470
MAXPOS 50/5	473

Encoder MR
128 - 1000 CPT,
3 channels
Page 432

Encoder HED_ 5540
500 CPT,
3 channels
Page 441/442

DC-Tacho DCT
∅22 mm
0.52 V
Page 449

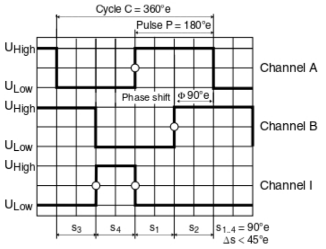
Brake AB 28
24 VDC
0.4 Nm
Page 491

130 maxon DC motor

April 2019 edition / subject to change

D.3. Encoder

Encoder MR Type ML, 128–1000 CPT, 3 Channels, with Line Driver



Direction of rotation cw (definition cw p. 60)

🟥	Stock program
□	Standard program
▨	Special program (on request)

	Part Numbers				
	225771	225773	225778	225805	225780
Type					
Counts per turn	128	256	500	512	1000
Number of channels	3	3	3	3	3
Max. operating frequency (kHz)	80	160	200	320	200
Max. speed (rpm)	37 500	37 500	24 000	37 500	12 000

overall length
overall length

maxon Modular System

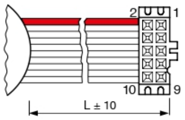
+ Motor	Page	+ Gearhead	Page	∅ Enc [mm]	Overall length [mm] / ● see Gearhead				
RE 25	125/127			25	65.5	65.5	65.5	65.5	65.5
RE 25	125/127	GP 26, 0.75 - 4.5 Nm	340	25	●	●	●	●	●
RE 25	125/127	GP 32, 0.75 - 6.0 Nm	342-347	25	●	●	●	●	●
RE 25	125/127	KD 32, 1.0 - 4.5 Nm	352	25	●	●	●	●	●
RE 25	125/127	GP 32 S	374-379	25	●	●	●	●	●
RE 25, 20 W	126			25	54.0	54.0	54.0	54.0	54.0
RE 25, 20 W	126	GP 22, 0.5 Nm	333	25	●	●	●	●	●
RE 25, 20 W	126	GP 26, 0.75 - 4.5 Nm	340	25	●	●	●	●	●
RE 25, 20 W	126	GP 32, 0.75 - 6.0 Nm	342-347	25	●	●	●	●	●
RE 25, 20 W	126	KD 32, 1.0 - 4.5 Nm	352	25	●	●	●	●	●
RE 25, 20 W	126	GP 32 S	374-379	25	●	●	●	●	●
A-max 26	152-158			25	53.5	53.5	53.5	53.5	53.5
A-max 26	152-158	GP 26, 0.75 - 4.5 Nm	340	25	●	●	●	●	●
A-max 26	152-158	GS 30, 0.07 - 0.2 Nm	341	25	●	●	●	●	●
A-max 26	152-158	GP 32, 0.75 - 6.0 Nm	342-347	25	●	●	●	●	●
A-max 26	152-158	GS 38, 0.1 - 0.6 Nm	353	25	●	●	●	●	●
A-max 26	152-158	GP 32 S	374-379	25	●	●	●	●	●
EC-max 30, 40 W	224			25			54.2		54.2
EC-max 30, 40 W	224	GP 32, 1 - 8.0 Nm	347/350	25			●		●
EC-max 30, 40 W	224	KD 32, 1.0 - 4.5 Nm	352	25			●		●
EC-max 30, 40 W	224	GP 32 S	374-379	25			●		●
EC-max 30, 60 W	225			25			76.2		76.2
EC-max 30, 60 W	225	GP 32, 1 - 8.0 Nm	347/350	25			●		●
EC-max 30, 60 W	225	KD 32, 1.0 - 4.5 Nm	352	25			●		●
EC-max 30, 60 W	225	GP 42, 3 - 15 Nm	355	25			●		●

Technical Data

Supply voltage V_{CC}	5 V ± 5%
Typical current draw	14 mA
Output signal	TTL compatible
Phase shift Φ	90°e ± 45°e
Index pulse width	90°e ± 45°e
Operating temperature range	-25…+85 °C
Moment of inertia of code wheel	≤ 0.7 gcm^2
Output current per channel	max. 5 mA

The index signal I is synchronized with channel A or B.

Pin Allocation

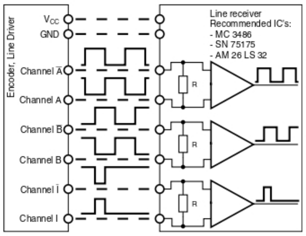


1	N.C.
2	V_{CC}
3	GND
4	N.C.
5	Channel $\overline{A}$
6	Channel A
7	Channel $\overline{B}$
8	Channel B
9	Channel $\overline{I}$ (Index)
10	Channel I (Index)

RE/A-max	L = 506
EC-max	L = 500

DIN Connector 41651/
EN 60603-13
flat band cable AWG 28

Connection example



Opt. terminal resistance R > 1 kΩ