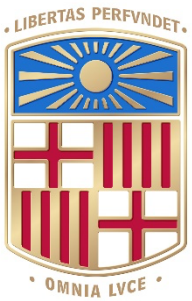


UNIVERSITAT DE BARCELONA

Final Degree Project

Biomedical Engineering Degree

“Da Vinci robot at Hospital Clínic. Haptic technology in robotic surgery“

Barcelona, June 2022
Author: Sergio Venteo Benavente
Director: Dr. Manel Puig i Vidal

Abstract

Minimally invasive surgery is growing and displacing traditional surgery. Despite the multiple benefits it presents for the surgeon and for the patients, the surgical robots used in these scenarios (mainly the Da Vinci robot) do not have a haptic feedback system.

This project it is created with the main objective of equipping and providing tactile sensitivity to an UR5e based robotic arm prototype and to adjust it as much as possible to the operation of the DaVinci minimally invasive surgical system, using an original EndoWrist tool too.

The project is divided into three main components: the pen user interface, the EndoWrist maneuverability executed by the robotic arm and the servomotors; and the communication and supervision module performed by a computer.

Obtaining the current obtained by the servomotors and applying different formulas and filters, the force exerted on the tissue is acquired. Which is used in the pen user interface to display the applied strength.

Finally, with several cheap components and with good communication between the different interfaces, the UR5e robot is intended to simulate the user's movements while giving it tactile sensitivity.

Keywords:

Minimally invasive surgery, Da Vinic, Haptic feedback, Robotic arms, Hardware and software platform.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to Dr. Manel Puig for his involvement and advise on this project. He has been a great director following my Final Degree Project and helping me in some problems that I have been encountering during the project.

What's more, I could not have carried out this project if I had not been allowed to use the laboratories and resources of the Department of Electronics and Biomedical Engineering of the University of Barcelona.

I would like to thank Estela Barrio, student of Electronic Engineering, for the mutual support and help throughout the project.

Finally, and on a more personal note, I would like to thank my family for their patience and support during this journey.

Table de Contents

1. INTRODUCTION

1.1. Introduction to minimally invasive surgery

Traditional surgery has been a technique that for many years has dealt with the diagnosis and treatment of diseases through surgical procedures. However, currently, minimally invasive surgery has become an alternative for these interventions. Minimally invasive surgery, also known as laparoscopic surgery, is a surgical technique that is performed through small incisions, inserting slender instruments and a tiny video camera so that the surgeon can see inside the patient and perform the intervention [1].

Usually, to enter on the patient body, the surgeon will make some small incisions, which are known as *ports*. Although the size of the ports depends on the procedure, they are not the same size as open surgery. That means, there won't be done extensive cuts in the skin, muscle, tissue, and nerves.

All the procedure is performed through the ports, where the surgeon inserts short, narrow tubes called *trochars*. Once these are placed, surgical instruments are placed along with a video camera equipment. Depending on the action that the surgeon needs to perform, the instruments are changed.

The patient isn't open as in traditional surgery, this condition provides several benefits for the patients and for the ease and comfort of the surgeon in the intervention. However, the procedure is more complex and longer.

What's more, advantages predominate over traditional surgery. As surgeon doesn't cut important regions of muscles, nerves, or other tissues, it is a less invasive and trauma operation [2]. This entails into a shorter hospital stays and a quicker recovery. If we compare the duration of recoveries traditional and minimally invasive surgery, we can pass from weeks to days. Furthermore, an important factor such as psychological may appear. If they tell us that they are going to have to open us in two, it is more frightening than if they are going to make a small hole in us. This fact gives us more security and what is certain is that after all it is safer because we lose less blood, and it is reduced the risk of infection. If we do a global comparison, minimally invasive surgery offers fewer complications.

After all, it is an invasive surgery, even there are less risks it can appear complications that can be rare and it is important to know the risks that can occur in each operation.

1.2. Surgical robots

During the last decade, surgical robots have got a revolutionary impact in the medicine. Giving a new version of diagnosis and treatment for doctors, as we have said, is known as minimally invasive surgery. To carry out this kind of procedures, robotic systems are necessary. In a robotic surgical procedure, three

or four robotic arms are inserted into the patient through small incisions [4]. The DaVinci Robotic System is the latest and greatest innovation and evolution in minimally invasive surgery. It is a system born in Silicon Valley, from military patents and developed by the Californian company Intuitive Surgical Inc., the da Vinci robot was launched on the market in 1999. Since then, it has revolutionized robotic surgery in the United States and the rest of the world, and over the years it is offering better results with the development of new robotic platforms.

The system is endowed with a 3D vision, equipped with Endowrist articulated instrumentation, and is equipped with a simple and intuitive control system that makes it easy for the surgeon to carry out complex operations through a minimally invasive approach [5]. This surgical robot is especially involved in prostate operations, heart valve repairs, gynecological surgical procedures cardiothoracic surgery, general surgery oncology and resolve congenital heart diseases.

This robot provides the surgeon with a very wide and clear field of vision, which offers the possibility of treating anatomical areas that are difficult to access. With the precision, minimal invasion, and safety that it offers, it becomes an advantage both for the doctor when carrying out the intervention, and for the patient and his safety.

The use of the DaVinci robot offers many benefits during the surgical intervention, such as reduced incisions, minimal bleeding, less painful post-operative recovery, among others. In the following sections we will go into more detail.

1.3. Objectives

The DaVinci robot, among the advantages that offers, there is a limitation that is still unsolved, haptic feedback. When the surgeon performs the intervention and carries out his movements on the remote to move the robot's arms, the movements are made according to what he sees. However, he does not know the force that he is exerting on the tissue nor the stiffness that it imparts in front of the movement of the arm.

The main objective of this project is to provide tactile sensitivity to an UR5e based robotic arm prototype and to adjust it as much as possible to the operation of the DaVinci minimally invasive surgical system, for this we will use an original EndoWrist tool. This would be a great modification that could be applied to the DaVinci robot and that will facilitate and give the surgeon more comfort during the operation.

The complementary objectives are:

- Development of haptic pen which as surgeon user interface, that could sense the movements of the surgeon, including rotations and translations, and tool openings.

- Include of electronic actuators devices to provide to the surgeon haptic sense while he is manipulating the haptic pen.

- Development of a mechanism to produce the maneuverability and imitation of the movements of the surgeon hand in the EndoWrist tool.

- Design and fabrication of the 3D pieces that are coupled to the Endowrist to assemble it to the UR5e robot end effector obtaining the maneuverability and haptic feedback.

1.4. Scope and limitations

The project consists of achieving the mentioned goals. As DaVinci robot is an expensive machine, we can't test our modifications on it. That's why all the implementations will be performed on an Endowrist and on UR5e robot. The framework of the project will be performed in the Physics laboratory of the Faculty of Physics of the University of Barcelona. To find out how it works the DaVinci robot, we had the opportunity to see him both in an operation and outside of it, in one of the subjects *"Aplicacions Mèdiques de l'Enginyeria".*

In addition, this project is carried out in conjunction with an electronic engineering student, Estela Barrio. She will focus on the part of creating the PCB boards for the servo motors and the pencil. This part will be taken into account in the project since it is carried together, but the creation of the PCB will be done by her.

The project development has been divided on two periods. The first one, from September 2021 to November 2021, was focused on understanding the performance of the DaVinci and testing the different platforms to do the software of the project. The second part, from February 2022 to June 2022, it was the period to execute the software, hardware, measurements and drafting of the project.

However, we need to consider some limitations as can be the time, the cost, and the knowledge of the student. The main goal of the project that is implement haptic feedback on DaVinci robot, it is a limitation of this robot that many companies are trying to solve. DaVinci is a complex robot and is laborious to implemented on the actual software and performance of the robot.

Furthermore, the time is an important restriction both to carry out the project and the prior knowledge that we must acquire. If all the conditions are met and the problems that appear in the project are solved in the agreed period, the development of the project will be performed on the agreed time.

1.5. Outline

The project is divided in 3 blocks, considering the amount of information studied, the different alternatives and solutions taken, and the results obtained.

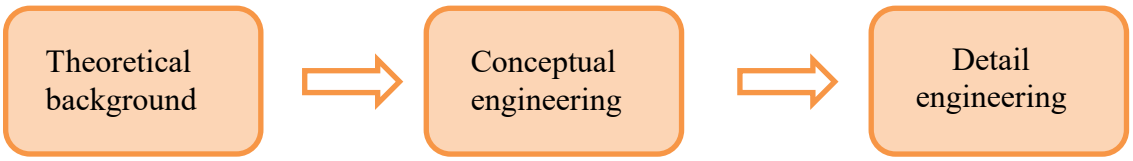


Figure 1. Project outline.

The first section encompasses all the information regarding the project. Knowing how the DaVinci was created and how it works, why there has been a growth of the minimally invasive surgery. This part also includes the state of art and analysis of the market, identifying the different sectors implicated and interested on this topic.

The conceptual engineering presents the different alternatives that could solve the main objective of the project. Studying the different options presented, a solution is proposed considering the pros and cons of each one.

Finally, the detail engineering shows the implementation of the solution and the selected materials to execute the project. The different analysis and measurements are presented along with the final results.

2. BACKGROUND

2.1. State of the art of surgical robots

If we want to do a review of the history of surgical robots, we must mention Karel Capek and his play *R.U.R.* (Rossum's Universal Robots). Capek was a playwright and, on his play R.U.R., it was the first time that the term *'robot'* was described. Its origin comes from the Slavic word *robota,* which refers to forced labor. Rapidly, this term was used for any kind of machine made for repetition of the same task. Nowadays, robots are machines that can undertake repetitive, programmed, and precise procedures [9].

A surgical procedure refers to invasive therapies that involves incisions performed to repair damage or arrest a disease from a living body. Although it has been over thirty years since the first time that a non-surgical robot, *PUMA 560*, was used in a stereotaxic operation in 1985. And, since the first surgical robot, *Arthorobot*, was used; the field of medical robotics is still growing has not reached its peak [7].

Robotic surgery, or also known as robot-assisted surgery, allows doctors to perform many types of complex procedures with more precision, control and flexibility compared to conventional techniques. This area is extremely associated with minimally invasive surgery, meaning that the procedures are performed through tiny incisions. The use of techniques to operate with less damage to the body than with open surgery, is associated with less pain, shorter hospital stays and fewer complications. This may play an important role as the medicine of the future wants to be less invasive.

As mentioned before, the modern robotic surgery started with the *PUMA 560*. After that, many companies started to create their owns models as *PROBOT*, for transurethral prostatectomy, and *ROBODOC*, for hip replacement operations; both from Integrated Surgical Supplies. However, the first robot approved by FDA was *AESOP* for abdominal surgeries. Finally, at the beginning of the 21st century, the two best-known surgical robots appeared, *Zeus System* and *Da Vinci Surgical System*. These two rivals have been dominating the field of robotic surgery for over a decade. On the one hand, the three-armed ZEUS platform, it uses one arm to hold the camera and the other two for surgical instruments. On the other hand, the four-armed Da Vinci has a central arm for the binocular lens and the instruments are used by a wrist of seven degrees of freedom. Noteworthy, this rivalry has benefited this area by providing innovations that have broken important barriers of minimally invasive surgery [8].

Currently in the surgical procedures, we found three types of robotic systems [6]:

1. Active systems have artificial intelligence, allowing the robot to perform tasks autonomously while a surgeon is supervising.
2. Semi-active systems have an automatic and a surgeon driven component, to complement these pre-programmed robots.

3. Master-slave systems, unlike the others, it doesn't have any pre-programmed or autonomous component. They depend completely on the surgeon activity, allowing him to operate directly on the robot.

2.2. Da Vinci Surgical System

The Da Vinci Surgical System is a robotic surgical system from the company Intuitive Surgical. This system is called *"Da Vinci"* in honor to Leonardo Da Vinci, a 15th-century Italian polymath, because of his work on robots and his study of human anatomy. The company was founded in 1995 with the idea of creating new robotic systems that could help surgeons to perform their surgical procedures [10].

In 2000, the Da Vinci Surgical System became the first robotic surgical platform commercially available in the United State approved by the FDA. It can be used to treat different pathologies in these fields: gynecology, general surgery, oral and maxillofacial surgery, pediatric surgery, thoracic or cardiac surgery. However, it is used especially in prostate cancer and other urological conditions [11].

Sometimes the term *'robot'* misleads people, it is the surgeon who performs the surgery with Da Vinci by using the robot instruments that he guides. The Da Vinci Surgical System is a master-slave robot, it obeys the surgeon movements while it increases the ability of surgeons to operate. Offering more precision and dexterity, reducing tremor, and providing a clear view of the patient's anatomy. What's more, the surgeon operates more comfortably seated on the console while is manipulating the robot and obtaining a 3D view of the patient's interior. The Da Vinci system translates the surgeon's hand movements at the console in real time, bending and rotating the instruments while is performing the procedure.

All these advances have favored both the patient and the surgeon. Postoperative pain has been reduced, having a great influence on postoperative recovery, and reducing the possibility of complications found in conventional surgery.

The Da Vinci system is a part of a much bigger picture, it is surrounded by a dedicated care team. The complexity of this kind of technology requires that all the personnel inside the operating room must be trained to manage the equipment.

In the market, there have been 5 generations of Da Vinci [12]. We are going to consider the fourth generation, *Da Vinci Xi*, which is the one that Hospital Clinic has. Even though, every generation presents different improvements, they all share the same structure [13]:

- Surgical Console: from this workstation the surgeon control and manage the arms of the robot through the pedals and controls. This console allows the surgeon a great vision, and at a scale of the surgeon's own hands, allowing his maneuvers to be firmer and more precise, assuming greater safety when performing surgery.

Figure 2. Da Vinci components: Surgical Cart, Surgical Console and Vision Tower [13].

- Surgical Cart: the patient cart rolls on wheels and it is positioned over the patient. It is composed by 4 robotic arms designed as human shoulder, elbow, and wrist. The four arms are fixed in space and derived to a remote center, permitting the surgical instruments move freely.

  Attach to the end of the robotic arm there are the EndoWrist, surgical instruments that provide surgeons with natural dexterity while operating through small incisions. On this surgical instrument you can add different accessories such as scissors and graspers, among others. These tools provide 7 degrees of freedom, providing ambidexterity and unparalleled precision. Each instrument can balance 180 degrees and rotate 360 degrees.

- Vision Tower: is responsible for the elaboration and processing of the image. It is a high-quality display system that generates 3D images with depth of field. Its houses advanced vision and energy technologies, provides communication across Da Vinci system components. What's more, includes a large HD display that shows the live procedure, and an electrosurgical unit with a series of racks for optional auxiliary surgical equipment.

2.3. Haptic Technology

As we have mentioned, Da Vinci provide to surgeons an incredible precision when they are working with tissues deep inside the body. However, this system presents some limitations as the lack of any force-feedback. This means that during the operation, the surgeon can't feel what's being worked on, without knowing the force you are applying or the tension you are exerting on the tissue [14]. However, thanks to the experience of the surgeons and the stereoscopic

vision of the tissue environment during surgery, they have been able to replace the ability to touch.

This issue could be solved with haptic technology, or also known as kinesthetic communication or 3D touch. This area includes any kind of technology that can create an experience of touch by applying forces, vibrations, or motions to the user.

One of the first applications on this field was in large aircraft using servomechanism system to operate control surfaces. When the aircraft approached a stall, the aerodynamic buffeting were felt in the pilot control.

Haptics are described as touch feedback, which include force (kinesthetic) and tactile (cutaneous) feedback. Nowadays, this is a limitation on minimally invasive surgery because the surgeon doesn't manipulate with the instrument directly.

The main goal of haptic technology in robot-assisted minimally invasive surgery is to is to give the surgeon the feeling that he is touching the patient with his own hands. This needs haptic sensors on the patient-side robot to get haptic information, and haptic displays to convey the information to the surgeon. Haptics can be structured as kinesthetic (refers to forces and positions of the muscle and joints) and cutaneous (tactile sense related to the skin).

Force feedback measure the forces applied to the patient by the surgical instrument and try to resolve this force via force feedback device [15]. Using conventional force display technology, the motors of the master manipulator will be programmed to recreate the forces sensed by the patient-side robot. As we have mentioned, the surgical instruments have 7 degrees of freedom and some of them are not actuated on the master, this is the reason why it cannot provide force feedback in all directions. In the case of tactile feedback, as the other method, it requires a sensor and a display. The goal is to detect local mechanical properties that indicates the health of the tissue to obtain information that can be used directly for feedback.

2.4. State of the situation

We have observed the advantages and limitations that the Da Vinci system has. We are going to focus on trying to provide it with a haptic feedback system. Nowadays, there is not haptic feedback system applied for medical healthcare. However, we can find several studies related to that.

Currently, force feedback research focuses on developing practical systems for application in fields such as education, entertainment, medicine, and training [58]. Although researchers have been studied tactile feedback during many years, there isn't a commercialization solution. On this field, it has appeared an effective application for video games using vibration feedback, giving information the video game experience as making and breaking contact.

3. MARKET ANALYSIS

On this section we will provide proof of surgical robot market. Focusing especially on robots related to minimally invasive surgery. Furthermore, we will look for the path that this sector wants to take and its corresponding future market prospects.

3.1. Evolution of the market

Since there was an increasing demand on improve precision and security on surgeries, robotic surgery took place deriving into minimal invasive surgical technologies. As we have mentioned before, the introduction of PUMA 560 in 1985 led to the first surgical robot. For the next 30 years, surgical robot have occupied an influential role while it increased its applications in modern robotic platform.

In the next picture we can observe the evolution of surgical robots applying the SEBMA acronym (Stereotaxic, Endoscopic, Bioinspired, Microrobots and Autonomous robots of the future) [16].

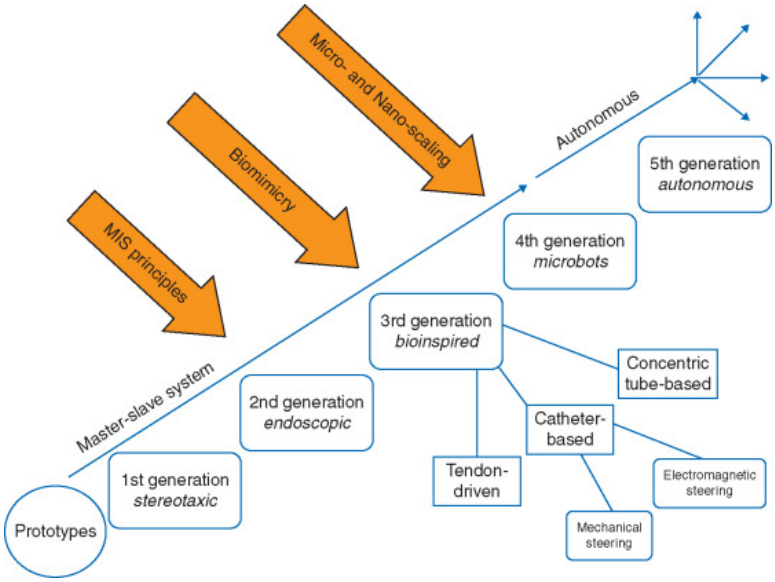


Figure 3. Evolution of surgical robots [16].

3.1.1. First generation – Stereotaxic robots

PUMA 560 was the first surgical robot, and it was used as the first laparoscopic cholecystectomy (minimally invasive surgery to remove gallbladder). Puma was utilized in stereotaxic brain biopsy by the surgeon to place the arms of the robots in a position to perform the task. This device was a precursor to a modified brain tumor excision device, Neuromate. The robots of this generation performed procedures that had a mathematic and mechanic strategy well defined, and the tissue tactility and vulnerability was limited [17].

However, robots couldn't perform multiple sequential tasks and a master-slave paradigm was introduced where the robot was a direct extension of the surgeon.

3.1.2. Second generation – Endoscopic robots

This generation is the greatest expansion of robotic surgery. The market needed more accurate robotic systems that can meet the requirements for minimally

invasive surgical technology. These surgical robots could overcome the difficulty to access to tissue places and organ systems as a result of anatomical restraints, instruments that lack precision and struggling in visualization that traditionally was limited to 2D [18].

In this generation are included two of the best-known surgical robots, ZEUS and Da Vinci System. Market forces led to a competition between both companies, Computer Motion and Intuitive Surgical Inc, respectively. Finally, Intuitive Surgical bought Computer Motion in 2003, starting the monopoly of Da Vinci System.

3.1.3. Third generation – Bioinspired robots

The third generation included principals from biomimicry and multiple articulation technology. In endoscopy we find NOTES (Natural Orifice Transluminal Endoscopic Surgery) developed for scarless surgery. Another example is one-port SPL (Single-Port Laparoscopy) that offered a platform with standard laparoscopic equipment through enhanced ergonomics [59]. These innovations offered to minimally invasive surgery instruments with articulated tips that allowed to reach hard areas with minimal access.

3.1.4. Fourth generation – Microrobots

The idea on this generation is create robots at a microscopic level that could enter in your body with minimal surgical footprint and work as a solitary robot or as a group of robots to image and treat diseases. One example is the capsule endoscopes that is swallowed by the patient, and the wireless camera on the capsule takes picture of the small intestine [60].

3.1.5. Fifth generation – Autonomous systems

The concept of fully autonomous, human-level consciousness robots remains conceptual, it is not a reality [61]. Autonomous robots will use and benefit from enhanced machine and deep learning capability.

3.2. Current situation

Nowadays, the manufacturer of the Da Vinci Surgical System, Intuitive Surgical, has been the market leader in robotic surgery since its creation. There are more than 5,500 Da Vinci robots installed worldwide. The company is firmly established, performing over seven million surgeries, that it has practically become synonymous with the term robotic surgery. The company stock price had grown 66% from US$312 in 2017 to $520 in 2019. Having a total revenue grow from $3.7 billion in 2018 to $4.5 billion in 2019. On the market there are four models: Da Vinci Si, Da Vinci X, Da Vinci Xi and Da Vinci SP.

Although it has not had great competition, in recent years, new surgical robots have been appearing, which may end its monopoly. As we have mentioned before, Da Vinci System is used for different pathologies but stands out in prostate cancer and other urological situations [19][20]. The current situation of the market is the following one:

- *Hugo Medtronic:* Medtronic, one of the global leaders in medical technology; in September 2019 unveiled the surgical robot Hugo. In 2021, it received CE (Conformité Européenne), authorizing the sale of the system in Europe. CE Mark approval is for urologic and gynecologic procedures, important fields to Da Vinci System [21].

Hugo system involves an operating console, a central power and a cart-based robotic arms. It is made up of pedestal-based module system and a surgeon console that it is inside the operating room.

- *Versius robot*: CMR Surgical, a British MedTech firm, after raising £75m in June 2018 from a host of investors including LGT and ABB, it has launched Versius robot. It has raised £195m to finance the global commercialization. It has taken a position that can threat the leading position of Da Vinci System across Europe and Asia.
  The structure is similar to Hugo and Da Vinci, it consists of individual cart-mounted arms controlled by a surgeon who sits or stands at its 3D high-definition control. It is really similar to Da Vinci, the handles used to operate are linked to video game controllers. It has been performed in gynecological, colon and renal procedures [22].
  If Da Vinci System is little available, it is due to its price. Apart from being expensive, its design does not allow much flexibility. The weight and size of Da Vinci robot can be a limitation inside of the operating room for its movement. On average, Da Vinci System operates once every day. These facts explain its lack of productivity and cast doubt on investing in it.
  However, Versius is cheaper and is characterized by its flexibility. What's more, CMR designed Versius with several separate robot arms, each one with their own surgical instrument and pillar. This allows a faster set up and is easier to carry from one operating room to another. These are the reasons why Versius is a real threat to Da Vinci System.

Of surgical robots that operates in fields different from Da Vinci System, we have:

- *ROSA robot:* it is a system composed by a robotic arm and a sophisticated software, that guides the surgeon during placement of joint prostheses. It is used for knee surgeries. It uses navigation systems, demanding anatomical reference points, ROSA can evaluate the position of the soft tissues and indicate where the cut should be made, offering a greater precision and a less invasive procedure. For its functionality, it uses preoperative X-ray images and data collected during surgery [23].
  Regarding to the equipment, it is composed by two towers, in one there is the robotic arm that guides the surgeon where to perform the cutting. On the other tower is the camera, which monitors the sensors that are placed on the axes of the patient leg.

- *MAZOR X*: Medtronic created a robotic surgery to perform orthopedic and neurological procedures for spinal surgeries. MAZOR X Stealth allows the surgeons to perform a less invasive surgical procedure [24].
  The robot-guided spine surgery system enables surgeons to create a 3D surgical plan for each patient before starting the procedure. During the surgery, the robotic guidance helps the surgeon to custom the plan with high precision.

- *Monarch ARES*: If your physician has discovered a spot on your lung, it has to perform a lung biopsy to know if it is cancerous or not. There are few options to reach those small nodules in the lung and it is difficult to perform it with manual bronchoscopes. The MONARCH® Platform is a

robotic assisted bronchoscopy introduced in 2018 to help reaching these nodules for biopsy [25].

3.3. Future perspectives

Robotics systems have been advancing quickly. Interdisciplinary collaboration is essential to develop surgical robots. It requires knowledge from different areas, as medicine, mechanical engineering, robotics, optics, and automatic control. Innovations are made to improve the clinical safety and effectiveness.

One of the objectives to improve these robots, focus as on this work, to equip them with haptic feedback. The lack of force feedback presents a challenge for a surgeon to control the amount of force that is exercising on the tissue [26].

Then, artificial intelligence can have an important role using algorithms to give machines human-like abilities to make decisions and perform the surgery autonomously. Machine learning enable robots to make predictions by the patterns that the recognize.

The idea of minimally invasive surgery is minimizing the level of invasiveness. What's more, improving visualization capacities would lead to many advantages. Many companies are pushing themselves to reduce the trauma and impact on the tissue. These would lead into a faster recovery, reducing the risk of infection and reduce postoperative time.

4. CONCEPTUAL ENGINEERING

This section outlines the different options for the hardware and software solutions. About software, we will focus on the different interfaces that would help us synchronize the distinct modules and connect them to the robot to test the program and to the computer, to perform the simulations. Regarding to the hardware, we will consider the robotic arms and the components to create the haptic pen.

The main objective is creating haptic feedback in a robotic arm. However, it is impossible to use the DaVinci robot due to its cost you cannot test projects on it. We will need a robotic arm to simulate the functioning of the DaVinci, an haptic pen which will reproduce the movements of the surgeon, and a software that will allow the control of the position and orientation.

The project is structured by three big components:

- Pen user interface that will offer and define the rotation of the surgeon hand, and that will generate the haptic feedback.

- The EndoWrist maneuverability, needing a robotic arm, an EndoWrist and an intermediary that would rotate the gears of the EndoWrist according to the rotation of the surgeon's hand. In addition, depending on the force that this intermediary has to use to rotate the EndoWrist, we will obtain the values of the force that we are applying.

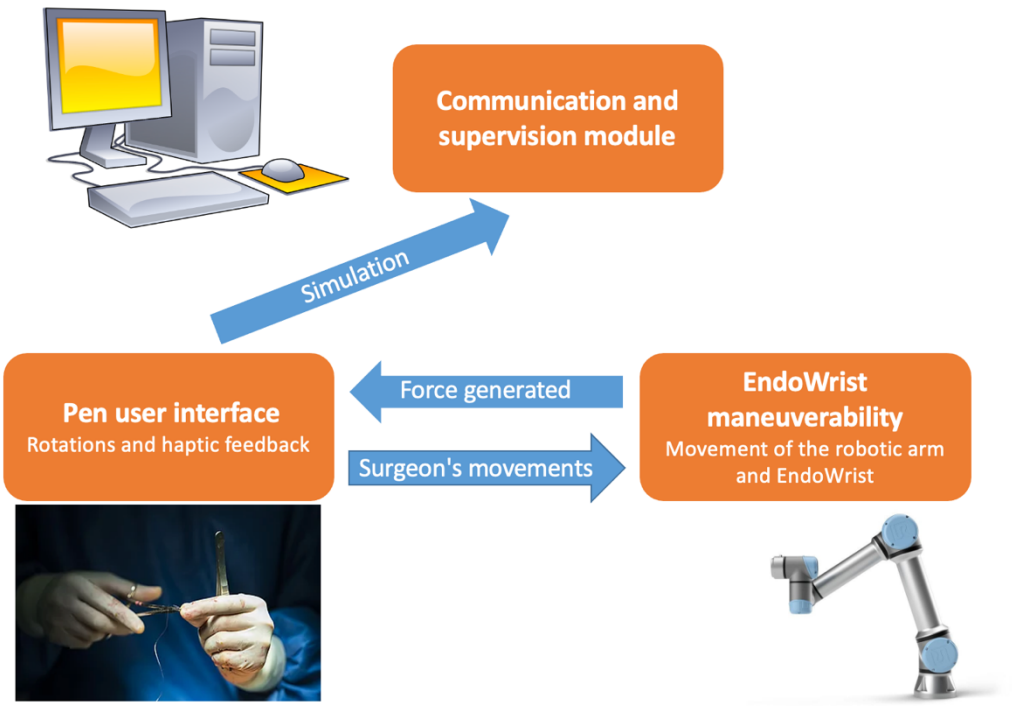


Figure 4. System schematic of the Final Degree Project.

-The last component, communication and supervision module which will need different software to perform the simulation and execute the programs in the real world.

We need to obtain the different components to meet the following specific objectives and achieve the challenge of the project.

Specific objectives:

- Understanding the performance of DaVinci robot.
- Learning about the software programs to perform the project.
- Analysis of the hardware components implemented on the haptic pen.
- Bluetooth/Wireless communication between the three main components.
- Read the torque generated due to the movement of the EndoWrist gears.
- Move the whole robot to change the orientation of the tool to enter through different entrances following the rotation of the hand-surgeon.
- Move the robot up and down following the Z-axis, simulating the entry and exit of the hole with the pen user interface.
- Move the EndoWrist following the rotations from the pen user interface.
- Creating haptic feedback with the torque received.

We will consider different options in terms of hardware and software to be used to accomplish the project objectives.

4.1. Software

An important part is the communication of the robot with the computer and with the haptic pen, apart from perform all the analysis on the program to calculate the torque, among other parameters, to achieve the haptic feedback. These objectives are fulfilled thanks to the software, and on this part, we will examine the different programs related to that.

4.1.1. Programming software

In order to perform the software, we have need different solutions for environments and languages of programming. On the following table we can observe the different options for the software:

Software	Arduino	LabView	ROS	Matlab	Python	RoboDK
Cost	Open Source	456 €/year [28]	Open Source	800 €/year [27]	Open Source	Open Source
Experience	1 year	2 years	6 months	3 years	3 years	6 months

Table 1. Options for programming software.

- Arduino is an open-source electronics platform that is able to read inputs, as can be the values from the IMU, and turn these into outputs, for example vibrate the vibration motor or ring the sound. What's more, it is free and a simple, clear programming environment really helpful for beginners [29].
- LabView is a graphical programming environment to develop and test systems. It is useful to build quickly automated test systems [30]. It enables measure physical systems with sensors and actuators and validate them.
- Robot Operating System (ROS) is a robotic middleware, a collection of frameworks for developing robot software [31]. It is an open-source operating metasystem, and provides services such as hardware abstraction, low-level device control, and package maintenance. All of this makes it easier to create robotic applications. It provides different applications as Gazebo and RViz which provide real world simulation and 3D visualization to interpret movements, respectively.
- Matlab is a programming platform created to analyze and design systems, used mainly by engineers and scientists [32]. For this project could be helpful for developing the algorithm to calculate the force applied on the tissue. Matlab is designed for a quick and accessible learning, as opposed it has a high cost.
- Python is a computer programming language used for software and data analysis [33]. It has become very popular in the recent years because of machine learning and software testing. It presents versatility and presents general-purpose language which allows creating a variety of different programs.
- RoboDK is a simulator for industrial robots [34]. It allows you to program robots outside the production environment and perform, directly on the computer, the programming and simulation of the movements and tasks made by the robot.

4.1.2. Designing software

The other part of the project is the 3D modeling of the anchor piece between the effector and servomotor. In this section, there is no previous experience, instead of dividing the table into cost and experience, we will do it in cost and skill.

Software	FreeCAD	AutoCAD	SolidWorks
Cost	Open Source	217 €/year [36]	1.197 €/year [35]
Skill	Low	High	Medium

Table 2. Options for designing software.

- FreeCAD [37] is an open-source to design real-life objects in 3D. Modifications are easy, enabling going back to model history. It works on different operating systems and reads and writes many file formats. It is a

type of modeling where the shapes of 3D objects are controlled by disparate parameters.

- AutoCAD [38] is a software of type CAD that allows the modelling in 2D and 3D. The modification of geometric models is almost infinite to develop any kind of structures. However, its difficulty is due to mechanical knowledge.
- SolidWorks [39] is a CAD program and 3D modeling software. It is used for planning, visualization and modeling of elements. It is a solid modeler that uses parametric feature-based approach to create models.

4.2. Hardware

On this section, we will observe the different devices considered to be part of the hardware of the project. Mainly, robotics arms and devices to complete the haptic feedback tool.

4.2.1. Robotic arms

The DaVinci is composed of different arms, with one of them we can perform the project. That's why we need a robotic arm, which is a mechanical arm that does similar functions to the human arm. The arm is composed by links which are connected by joints. This structure allow different movements as rotations and translations. The links compose a kinematic chain with an end effector at its extreme.

These robotic arms have a software behind programmed to execute a specific task and repeat it many times accurately. They have an important role on industrial, manufacturing and assembly sectors because they can perform heavy and repetitive procedures during a long period of time.

However, the robotic arm on our project would work as a *'cobot'*. This term is used for robots created to have a direct interaction with humans [40]. Their functions are very different compared to the traditional industrial robot applications. Having to interact with people, there are some components that are compulsory to ensure the safety of the people around them. That's the reason that they are built with lightweight materials, sensors, limitation of speed and rounded edges.

The Industrial Federation of Robotics (IFR) is an organization created to promote, reinforce, and protect the robotics industry. This one has defined different levels of collaboration between human and robot which are:

- Coexistence: both works alongside.
- Sequential Collaboration: both shared workspace but their functions are sequential.
- Cooperation: work at same time and in motion.
- Responsive Collaboration: robot responds to the movement of the human.

In the following section, we are going to present different robotics arms and different features will be consider selecting the best solution.

4.2.1.1. UR5e

UR5e is a robot from Universal Robots (UR) which is a lightweight and adaptable collaborative industrial robot that tackles medium-duty applications with ultimate flexibility [41].

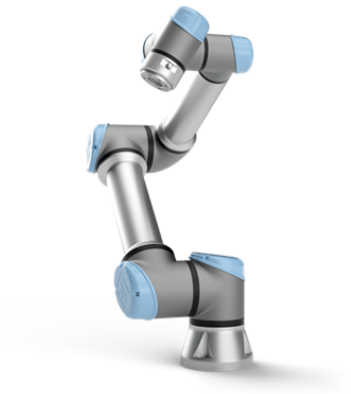


Figure 5. UR5e robot.

It is a great example of *'cobot'* able to repeat tasks. It is flexible and presents an arm with a force torque that enables to perform safety exercises. Its software consists of an easy program that helps the costumer with a very fast learning [42]. We have to highlight the 6 degrees of freedom, its fast movement and its lightweight (18 kg). What's more, at the end-effector we are able to attach a tool, as can be the EndoWrist needed on the proyect. These features allow to perform recurring tasks and flexibility to automate multiple manual tasks. The price can be the main handicap, with a cost of 30.581€.

With the recent technological developments, we have seen these kinds of robots in healthcare industry performing tasks supervised by a human.

In the case of UR5e, it has been involved in rehabilitation from injuries caused by blood clot and strokes, helping patients with repetitive movements that form part of the rehabilitation process. Doing these charges, it can support therapists allowing them to set up training programs. Apart of this, it has been present in some businesses as Aurolab, helping in manufacturing intra ocular lenses. Its presence has increased affordability, regain vision at lower cost, reduce of power consumption and a decrease of quality problems [43].

4.2.1.2. Mover 4

Mover 4 is robot from Commonplace Robotics Mover, unlike UR5e, is not suited for industrial production. The main objective is fulfilling the needs of schools, universities, or research institutions [44].

Figure 6. Mover4 robot.

Its programming environment is really intuitive, including graphical and textual program for ease of use by beginners. Its little weight, its payload of 0,5 kg and his ability to equip different grippers, they have made it an option for our project. However, it has 5 degrees of freedom, we move away from the freedom of movement of the DaVinci. On the other hand, its price is more feasible and economical, 3,730.65 €.

At the end of the arm can attach different grippers, this would allow us to assemble the EndoWrist and perform the project. What's more, its interface let us an easy control of the position of robot in real time and it has a low price.

4.2.1.3. Kuka Robot Arm

Figure 7. KUKA KR 360 robot.

KUKA is a global automation corporation known as one of the world's leading suppliers of intelligent automation solutions, as robot arms [45].

KUKA KR 360 is an example of many robot arm intended for handle heavy components. It has an arm extension of 500 mm with a repeatability of 0.08 mm providing great precision [46]. Its 6 degrees of freedom and large number of applications characterize it for being versatile and flexible. With a price about 10.000 € and the light work of large and heavy components, they allow to establish a low-space and low-cost solution.

4.2.2. EndoWrist Da Vinci

Intuitive Surgical has patented EndoWrist Instruments, these ones are designed to help the doctors on the surgery. Its main function is recreating the human wrist. However, these instruments allow more freedom of movement than a human wrist can achieve. Specifically, it as 7 degrees of freedom, swinging 180 degrees and rotate 360 degrees [47]. At the end that connects with the robotic arm, there is a mechanism that controls these degrees of freedom. There are 5 gears that control the roll, pitch and yaw of the tool.

The EndoWrist perform the movements that the surgeon is making on the surgical console, to do the motions in real time, internal cables are needed, allowing rapid, maximum responsiveness and precise action of the manipulation. These features allow to the surgeon a natural dexterity while is performing the surgery through a small incision. During a surgery, different EndoWrist tools are used, depending on the task that is performed, some examples are graspers, needle drivers and energy instruments.

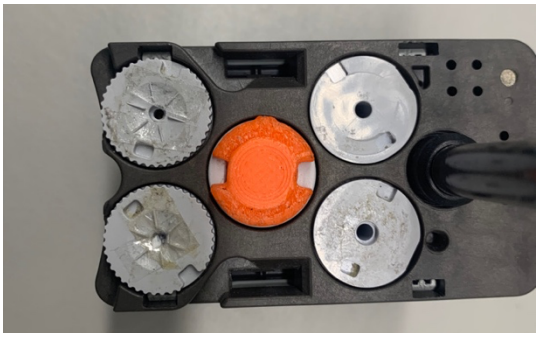


Figure 8. Mechanism rotations EndoWrist.

Figure 9. Tweezer EndoWrist.

4.2.3. Servomotors

Servomotors are a rotatory or linear actuator that can control the position, speed or acceleration of the axis. It is an electrical device that with its rotation, can spin a part of a machine with great efficiency and precision. Its mechanism is a closed-loop that uses the position feedback to control its movement and the final position.

The input is a signal that can be analog or digital, which an output that represents the position requested. The position feedback provides different parameters, as can be current, velocity or position to the servo controller, depending on the input [48].

The servomotor is an electromechanical device that can generate torque and velocity depending on the current and voltage provided. These characteristics are crucial, because these devices will be used to rotate the mechanism on the EndoWrist and provide the different rotations. We will need four servomotors, one for rolling, another for pitching and two for the yaw to open and close the tweezers.

It is important the resolution, speed of rotation and torque of the servomotors. These servomotors will have to rotate according to the movement of the surgeon, meaning that the rotation must be very precise and rotate imitating as close as possible the movement executed by the surgeon. Also, it needs enough strength to be able to move the gears of the EndoWrist.

In the following table different solutions are presented:

Servomotor	Operating Voltage	Stall Torque	Weight	Rotation range	Running Speed
Sunfounder SF180M	4,8 V – 6 V	2,5 kg/cm	9 gr	180º	0,09 s/60 °
Longruner KY66	4,2 V – 6 V	1 kg/cm	9 gr	180º	0.12 s/60º
Dynamixel AX-12A	9 V – 12 V	12 kg/cm	54.6 gr	300º	0,02s/60ª

Table 4. Options for servomotors.

4.2.4. IMU Sensors

An inertial measurement unit (IMU) is an electronic device that measure the orientation and angular rate of the body at which is attached. It is a combination of accelerometers, gyroscopes and magnetometers [49].

This device will be crucial to know the rotations that the surgeon makes and that these commands are sent to the rob arm. Using the accelerometer of the IMU, the linear acceleration will be calculated, and the rotational rate is obtained by gyroscopes.

IMU	Degrees of Freedom	Power	Weight	Price
ICM-20948	9 DoF	3,6V -1,71V	0,7 gr	13,20€

MPU 9250	9 DoF	2,4 V – 3,6 V	2,72 gr	12,99 €
MPU 6050	6 DoF	2,4 V – 3,6 V	2,1 gr	5 €

Table 5. Inertial Measurement Unit comparison.

4.2.5. Haptic stimulus

One of the main objectives is the haptic feedback, giving a stimulus to the surgeon so they know the force that are applying on the tissue. Two devices are selected as haptic stimulus: buzzer and vibration motor [50].

Vibration motor is compact size coreless DC motor with the objective of creating a vibration to inform the user that the signal or condition has been met [51]. This device will be use in our project to inform the surgeon how much force is applying on the tissue. If he is doing a lot of force, the motor will vibrate more than if you are applying less force. This vibration has to be controlled, because if it is really big it could affect on the precision of the surgeon.

Vibration Motor	Operating Voltage	Rotational Speed	Weight	Price
Seeed Studio 105020003	3V – 5V	9000 rpm	8,8 gr	3 €
PWM Vibration Motor Module	3V – 5,3V	9000 rpm	16 gr	5,9 €
Mini Vibration Motor Seeed Studio	2,5V – 3-5V	7000 rpm	2 gr	1,14 €

Table 6. Comparison of Vibration Motors solutions.

The second device is a buzzer, a vibrator-type electronic device. Its function is creating a sound, that can be continuous or intermittent, used as a signal or warning [52]. In our case, it will alert us at the moment that we are making a lot of force on the tissue or surface in question.

Buzzer	Resonance Frequency	Intensity	Operating Voltage	Price
HW-508	2300 Hz	110 dB	4V – 8V	3 €
Buzzer RS PRO	2000 Hz	86 dB	3V – 5V	2,9 €
MCKPT-G1720-3922	4000 Hz	85 dB	30V	1,14 €

Table 7. Comparison of Buzzers solutions.

4.2.6. Microcontroller

Our project will have different components that need to be connected wireless. The pencil, that will simulate the surgical console imitating the movement of the surgeon hand, needs to be connected by Bluetooth to the robotic arm, which will execute the motion [53].

For this purpose, we will use microcontrollers but the ones that contain a set of peripherals including WiFi and Bluetooth wireless capabilities. We should know that a microcontroller is an integrated circuit created to perform a specific task in an embedded system. It can do it by interpreting the information and data that receives from the peripherals using its central processor.

Microcontroller	Memory	Bluetooth	Operating Voltage	Price
LattePanda	4 GB	Yes	7.5V – 15V	125 €
ESP32	520 KB	Yes	2.2V – 3.6V	25 €
NanoPi M4	4 GB	Yes	5V – 12V	47,51 €

Table 7. Microcontrollers solutions comparison.

4.2.7. Push buttons

Apart from rotate the EndoWrist, we need to translate the robotic arm to a desired position and use the tweezers, opening and closing them. To achieve that we need to include some push buttons, that depending on the button that we press, it will perform one of the mentioned tasks.

Push button	Company	Weight	Price
COM-00097	Sparkfun	270 mg	0,32 €
1301.9314.24	Schurter	250 mg	0,45 €
B3F-1022	Omron	250 mg	0,38 €

Table 8. Comparison push buttons solutions.

4.3. Final selection

In this section we will present the selection of the software and hardware components. We have taken into account the features of each element, its prices and the resources that have been presented to us. As mentioned in the first section, our project has been developed in the Physics laboratory of the Faculty of Physics of the University of Barcelona, there we have some components that

can be useful for our solutions in the hardware section. And because we are students, we are also presented with offers for different software solutions.

Starting for the hardware, UR5e is selected although it is the most expensive. This is because this robot is already in the Department of Electronics and Biomedicine of the Faculty of Physics of the University of Barcelona. So having this robot within our reach and being able to use it for this project would be affordable, if we wanted to use any of the other alternatives it would mean a greater economic expense since we do not have any of the other alternatives. What's more, of the alternatives presented, UR5e is the one with more degrees of freedom fact that has allowed its use in some clinical activity as mentioned in the previous section. Another point in favor is that we already have a little experience with this robot due to its use in one of the subjects of the career, "Robotics and Control of Biomedical Systems".

Regarding the hardware components for the haptic feedback, some of them have been selected due to its availability and presence in the laboratory, in this way they did not have to buy or wait for their arrival. In the case of haptic pencil, we have selected MPU 9250 because it is the one that has more degrees of freedom, along with ICM-20948. However, we have used it before, and it is available in the University. Furthermore, we know how to obtain its orientation and translate this information to the servomotors. Also, it will include an ESP32 to connect wirelessly the pencil with the servomotors, and with the computer. In summary, we will have an ESP32 in the pencil, to obtain the orientation and translation; in the servomotors mechanism to rotate the EndoWrist imitating the movement of the surgeon's hand; and in the computer to get all the necessary values and perform the relevant calculations and analysis.

The rest of the plugins included are PWM Vibration Motor Module, HW-508 and COM-00097. The vibration motor will change its frequency depending on the amount of torque that we are applying. In the case of the buzzer, it will change in the same way, but increasing the intensity of the volume.

Concerning the servomotors mechanism, we have used Sunfounder SF180M. In first place we utilized the Longruner KY66 because they were available in the University. However, when we try to move the EndoWrist with the servomotors, Longruner KY66 it didn't have enough strength to move and rotate the EndoWrist as we wished. That's the reason why we use Sunfounder SF180M, because they have bigger stall torque.

With reference to the software, Arduino has been chosen to perform most part of the project. It will connect the different ESP32, calculate the torque and carry out the different haptic stimulus. RoboDK and python will be used for the simulation part and the direct connection with the UR5e, to run the program on the robotic arm. A great advantage of these programs is that they are free, except RoboDK, However, as we are student, we can obtain a free license for this program.

Finally, for the construction and design of the anchor pieces between the EndoWrist and servomotors, FreeCAD has been selected. The reason is that it is an open source and its learning is very simple for someone who has not had any type of experience with this type of software.

COMPONENT	SELECTION
Programming software	Arduino – RoboDK - Python
Designing software	FreeCAD
Robotic Arm	UR5e
Servomotor	Sunfounder SF 180M
Inertial Measurement Unit	MPU 9250
Buzzer	HW-508
Vibration Motor	PWM Vibration Motor Module
Push Button	COM 00097
Microcontroller	ESP32

Table 9. Proposed solution for each component software and hardware.

To understand better how the components are going to interact between them, a scheme representation of the prototype conception is shown below.

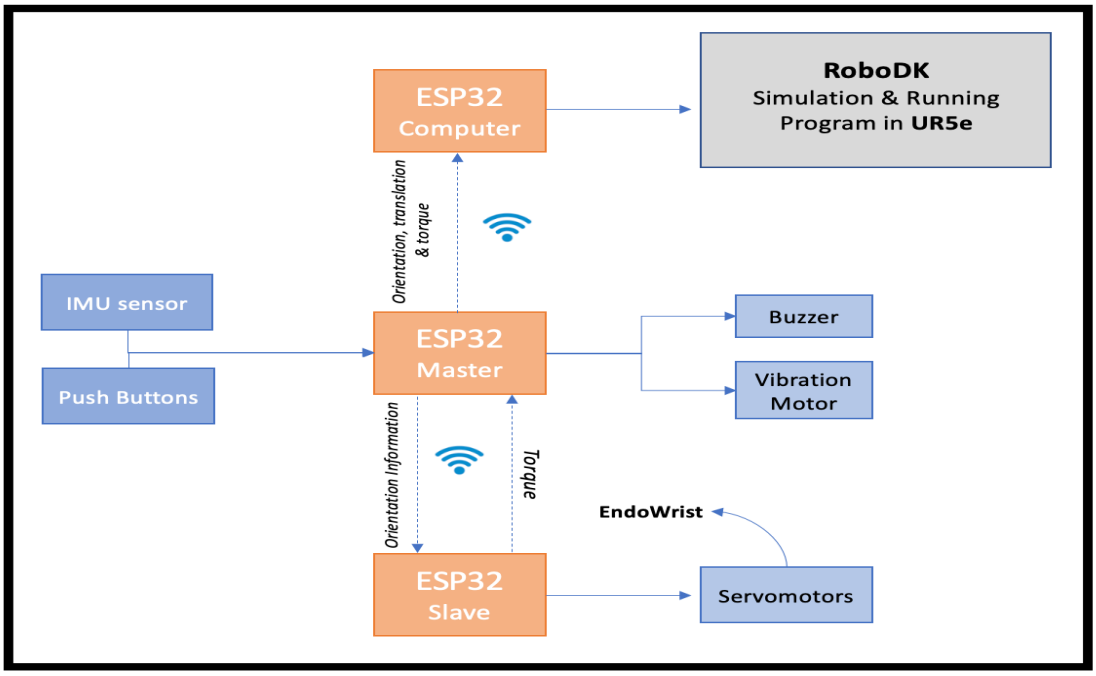


Figure 10. Schematic representation of the own project.

5. DETAILED ENGINEERING

This is the main part of the project, where all the hardware and software implementation are explained. We will use the items defined in the conceptual engineering and defined and specify how each process and part of the *Figure 9* was performed. To get the big picture, a pen, simulating the hand of the surgeon, will move the robot UR5e and the EndoWrist to his state, giving tactile sensation in the surgeon's hand when applying a lot of force with the EndoWrist on some tissue.

5.1. Prototype Implementation

First of all, to understand each main component of hardware and its relation, we will divide the section in 3 parts, the pen user interface, the servomotors with the EndoWrist and UR5e (EndoWrist maneuverability), and the communication and supervision module performed by a computer. Each one of them share a component, the ESP32. This processor will allow us to connect each module between them, via WIFI, avoiding wiring between them that would make their use very uncomfortable.

Otherwise, the pen provided to us was a PCB with an IMU and with two buttons, apart from the ESP32. The IMU will be used to read the dimensions of movement from the surgeon hand, used in the servomotors to rotate according to them.

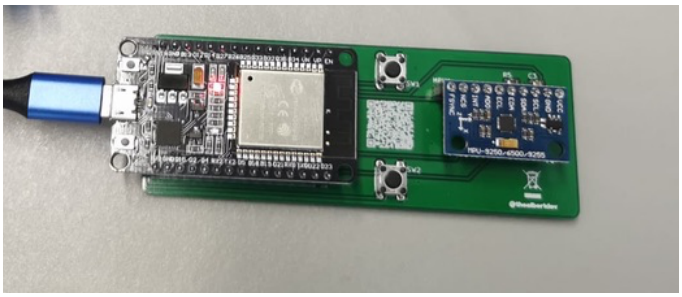


Figure 11. Pen provided to us with an ESP32, two buttons and an IMU sensor.

The EndoWrist was also provided to us from the Hospital Clínic. An EndoWrist can be used just 10 times, once this threshold is exceeded, they stop using it even if it works fine. For our project it does the job, its use is adequate because it meets the requirements.

5.1.1. Wireless communication

Each ESP32 has its own MAC direction, which is like its representative. Using a WIFI library in Arduino, we can get the address of each of the three ESP32. After that, we will create three different Arduino files that will represent where one will represent the pen (known as master), another the servomotors (called slave) and the last one will receive all the information that will be sent to the computer for the simulation.

To perform the transmission of information, each ESP32 needs to know the MAC direction of the ESP32 to which they are going to send or receive information. The master will need the MAC address of the slave and the computer; the slave

will need the direction of the master, and the computer will need the address of the master. Apart from that, we need to define the structure of the data that we will receive and transmit on each ESP32. Now we are going to observe which information is going to be passed between the ESP32.

5.1.2. Pen and dimensions of movement

We needed to understand how we are going to use the information received by the IMU. From this component we will use the roll, pitch and yaw. They are three dimensions of movement defining how an object move. In that case, we are going to define how is moving the surgeon its hand.

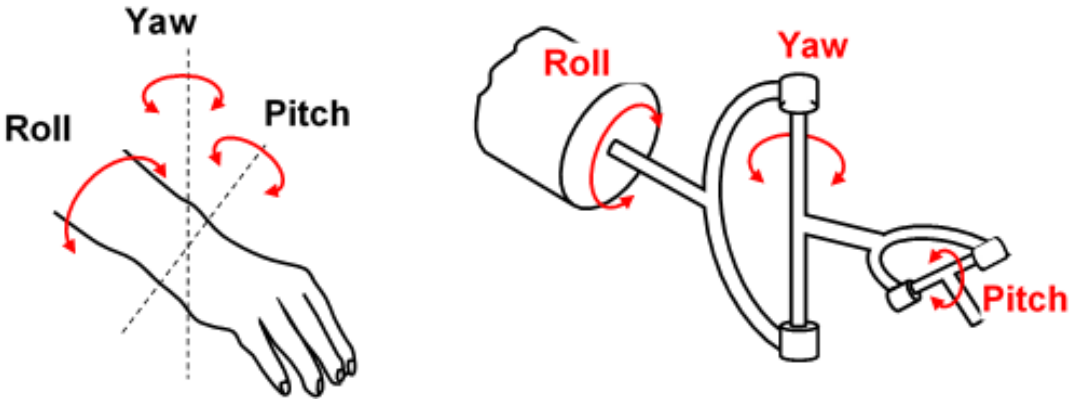
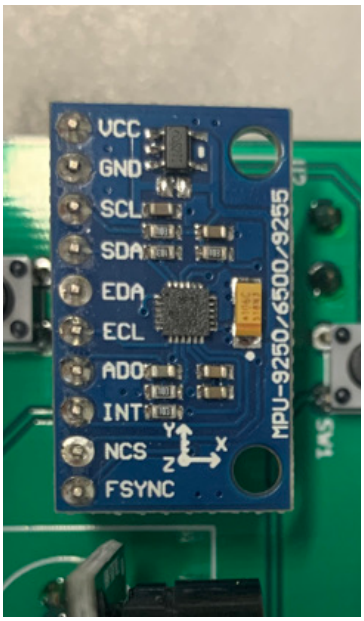


Figure 12. Representation of the roll, pitch and yaw in a human hand.



The IMU has a representation of the three axis as shown in *Figure 12.* The roll is the rotation about the x-axis, the pitch about y-axis and yaw z-axis. These rotations will be sent to the servomotors which will move the gears of the EndoWrist. As shown in *Figure 13,* although there are 5 gears, just 4 of them are used. For the roll (1) and pitch (2) we need just one, however for the yaw we need two (3,4). This two will rotate oppositely to move the two grippers together, will rotate the same way when they want to open the clamps. Considering that, for each interruption of the IMU sensor we read the data that it provides us. We will send the roll, pitch and yaw from the master to the slave and the computer.

Figure 13. IMU sensor.

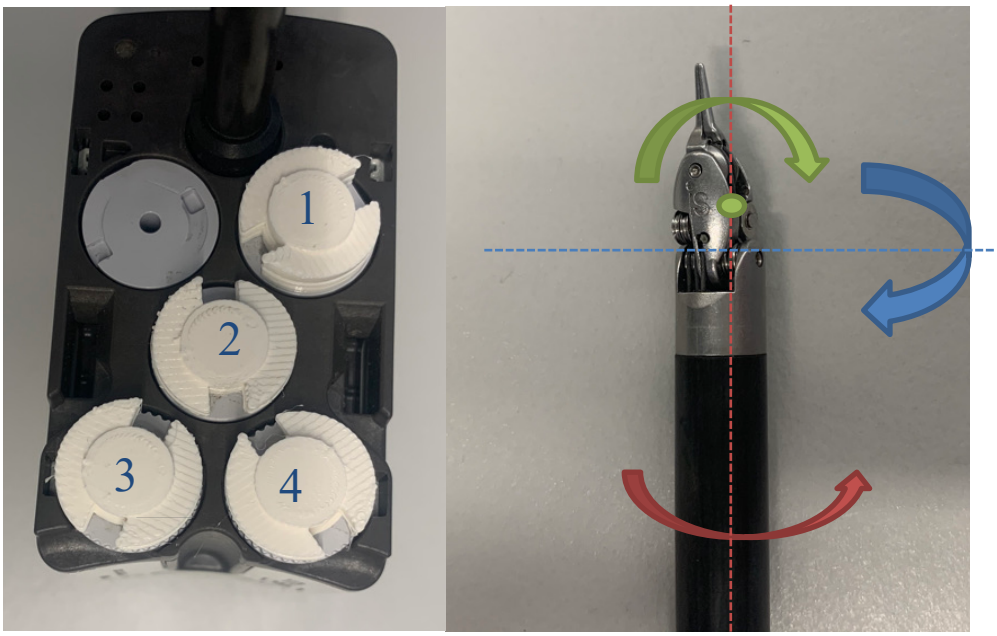


Figure 14. Rotations of the servomotors and their corresponding movement in the gripper (1.Roll-red, 2.Pitch-blue, 3. And 4. Yaw-green).

Regarding to the buttons, we are going to read their status in Arduino to, subsequently, move the robot forward or backward.

5.1.3. Reading torque

To know which force are be applying on the tissue, we are going to consider the torque that the servomotors are producing. In that case we will measure their electrical power, P=I*V.

We will measure the current that is going through the servomotor because the voltage is going to be the same. To achieve that we need a resistor in series with the motor, measuring its voltage drop we can know the current. As the voltage drop is practically constant because the voltage is the same, we can assume that the electrical power is directly proportional to the current that we are measuring and its torque. We need a small resistance, if not it will be very difficult to the current to go through. That's why, the resistance used is 1.6 Ohms for all the servomotors.

In *Figure 15*, we can observe how the pins and the resistor are connected to be in series. We need to connect the GND of the motor and terminal of the resistance to a GPIO and ADC pin.

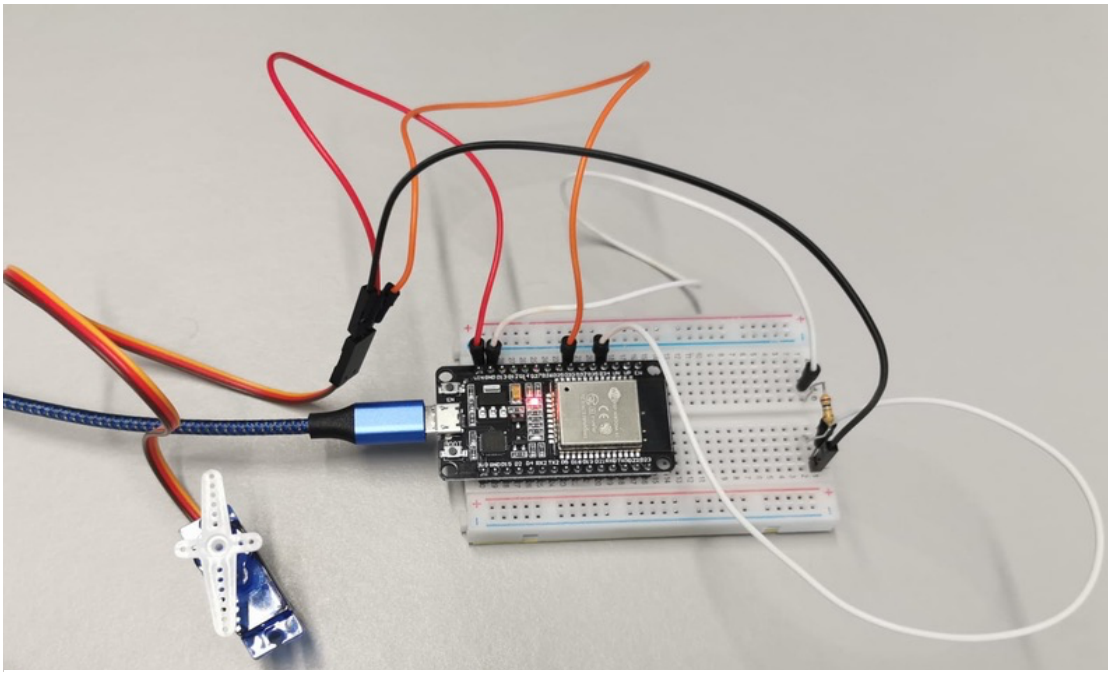


Figure 15. Servomotor connected to a ESP32 and a resistor in series.

This information will be sent to the master, which later from this, will be sent to the computer. Finally, we apply this to 4 servomotors. The hardware of the prototype is represented in *Figure 19,* we can observe the 3 components, the pen/master, the slave composed with 4 servomotors and its resistor in series, and the ESP32 computer.

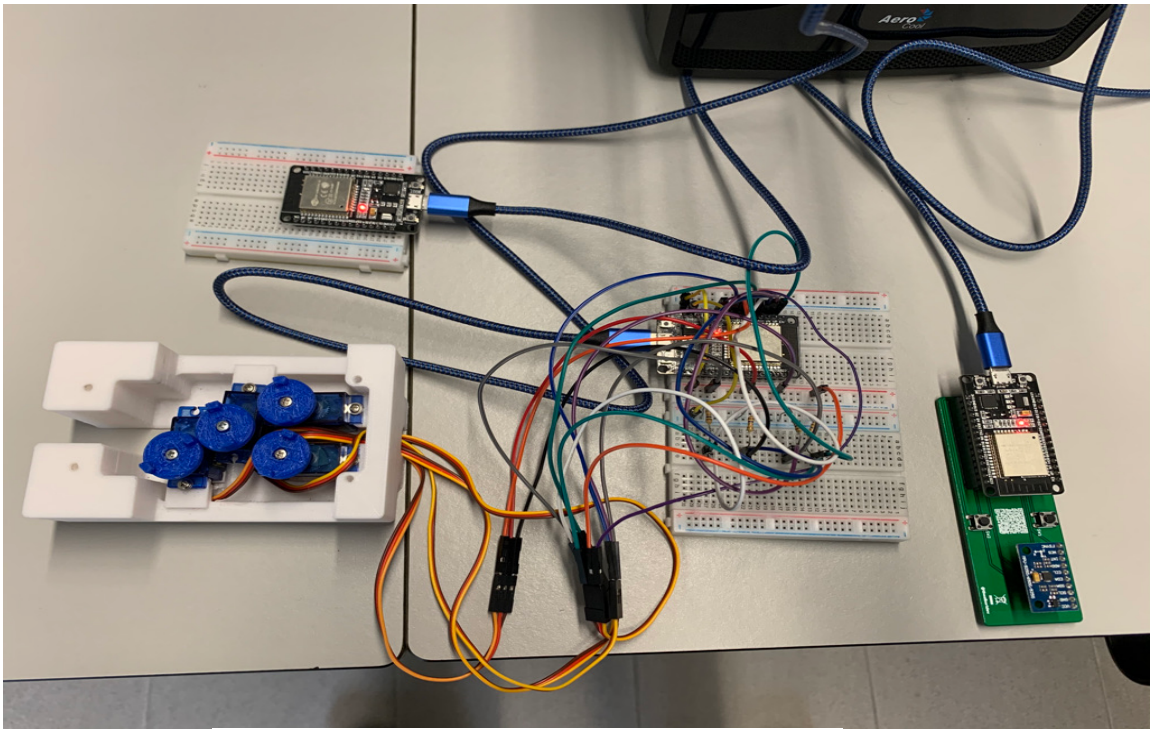


Figure 16. Prototype hardware.

5.1.4. Simulation in RoboDK

Before starting with the RoboDK, we have seen that the slave sends to the master the information about the torque, and the master send this last information, in addition to the RPY angles and the status of the buttons to the computer. Finally, the ESP32 connected to the computer has all this information. In RoboDK, we request this data to Arduino, storing the data and converting the variables to floats or integers.

The other section is to generate the robot UR5e in the environment, downloading it from the RoboDK library. The tool recreating the EndoWrist was provided to us. We needed to define both variables to be able to work with them.

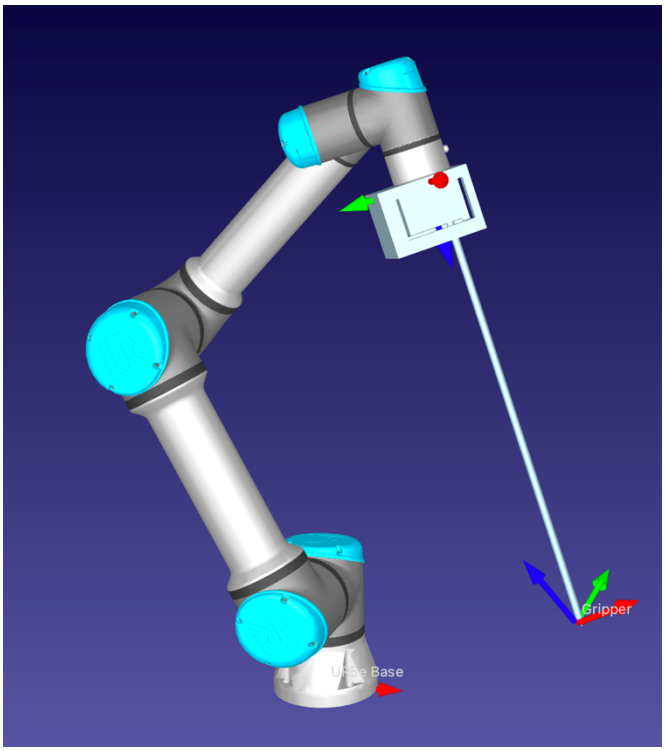


Figure 17. UR5e in RoboDK.

The idea is to recreate the movement of the tool of the EndoWrist. To achieve that we create a new frame named *'Gripper'* in the extreme of the EndoWrist. We will create a translation matrix achieving the only movement on this position, which will represent the movement of the gripper. After that, using the rotations matrixes, we will apply a *rotx, roty* and *rotz* using the roll, pitch and yaw values obtained from the IMU, respectively. A *rotx(pi)* is needed because if not the Z-axis is pointing down. With this matrix that defines the new position of the gripper, we implement the rotation with the function *.setPose()* which will move the gripper according the roll, pitch and yaw values.

Apart from that, depending on the status of the buttons, the robot will move forward or backwards. To achieve that we will get the position of the robot with the function *.Pose(),* which is a matrix 4x4, and we will multiply a translation just in the Z-axis of 1 mm. Depending on the button that we click, the value will be positive or negative affecting the direction of its movement.

5.2. Final model

As we have seen, the prototype model can't be implemented on the UR5e in real world and doesn't have any haptic technology. In this part we are going to

observe the different modifications that we have performed to the previous components, to achieve the defined objectives.

5.2.1. 3D pieces

One of the main inconvenient when carrying out the project in the UR5e, it is the lack of some anchoring pieces between the servomotors and the EndoWrist. We needed pieces that fixed in both parts and transmits the rotations so both parts were in agreement. Two types of pieces in question were designed. One of them considered the dimensions of the servomotor and the other, the dimensions of the EndoWrist.

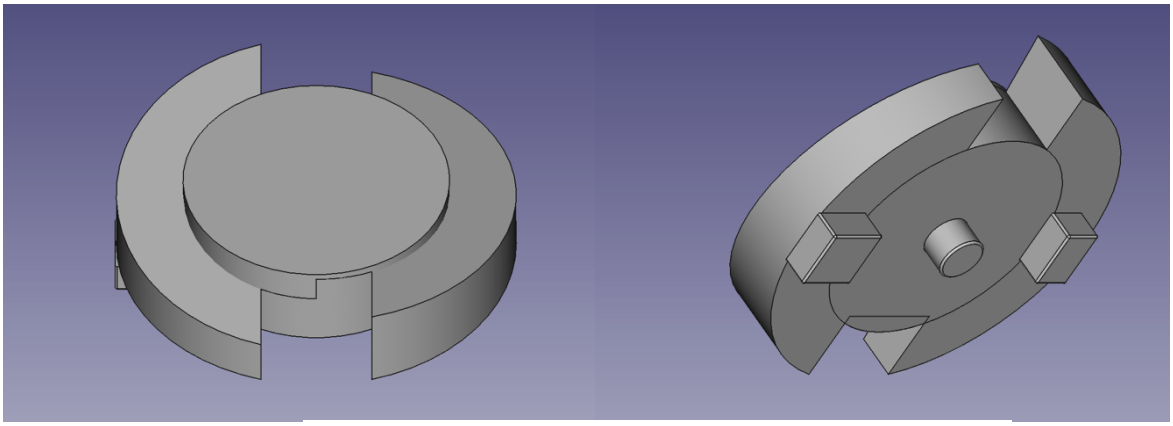


Figure 18. EndoWrist disks designed in FreeCAD.

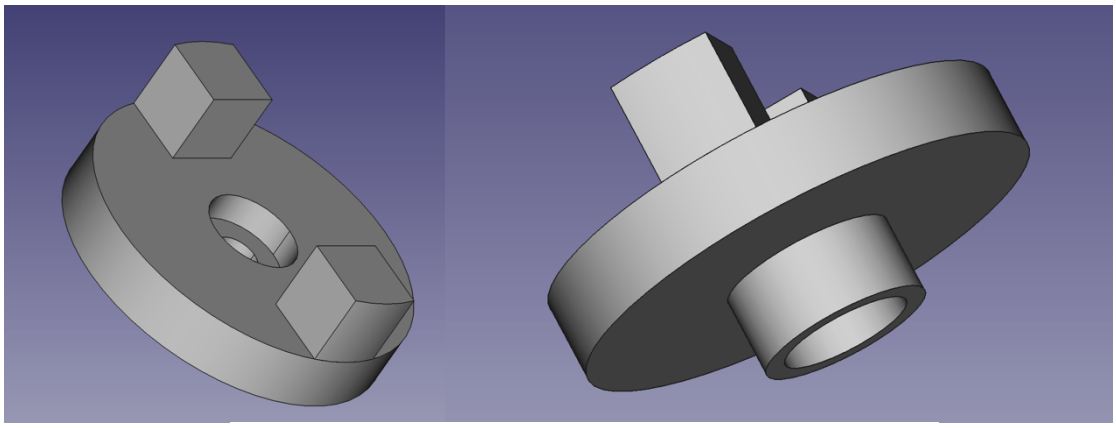


Figure 19. Servomotor disks designed in FreeCAD.

Both pieces were created in FreeCAD and then 3D printed. Due to the limitation of precision in the 3D printer, the two pieces were split in half to improve printing.

5.2.2. Slave on PCB board and torque limit

The servomotors were in a protoboard with a lot of cables which was difficult to put it in the UR5e. To solve this problem, we moved the 4 servomotors and the ESP32 into a PCB board which was performed by Estela. As the idea was to make it wireless, we have to avoid wiring and make it work by itself, that's why a battery was also included to be autonomous.

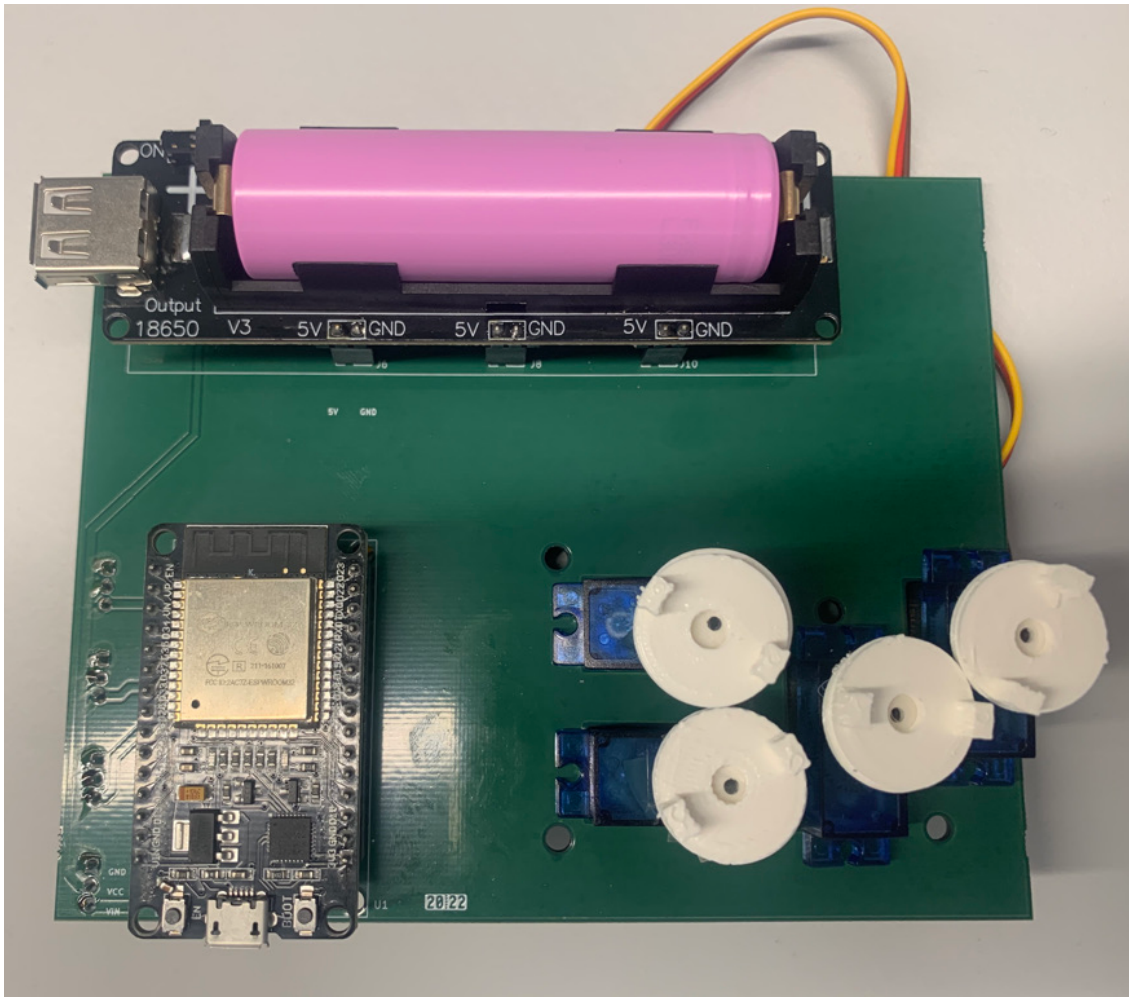


Figure 20. Slave PCB board composed by a battery, a ESP32 module and 4 servomotors.

5.2.2.1. Implementations slave software

The idea is to simulate the use of this device in a surgical environment, meaning that the movements of the hand of the surgeon needs to go very slow. However, we must consider any situation that could worsen the intervention, as for example, if the pencil falls or the surgeon shakes a lot its hand. In that case, we don't want the servomotors to rotate because it would make a very sharp rotation causing internal injuries to the patient.

To avoid this problem, we have limited the rotations. If the new rotation value is greater than 10 degrees with respect to the last value, the servomotor will not

rotate and will take the previous value as the measurement. In this way we ensure that the procedure is carried out smoothly, increasing the precision of the intervention and the safety of the patient.

Another aspect related to this topic, is the limitation presented by the servomotors used. These ones only rotate 180 degrees, nevertheless, the IMU offers 360 degrees. This inconvenient provides a tremendous rotation of 180 degrees when the servomotor gets a 0 degree and then a 360 degree. To solve that, we limit the field of movement of the servomotor between 0 and 180 degrees. If the degrees obtained are bigger than 180 degrees and smaller than 270 degrees, the servomotor will stay at 180 degrees. However, if the values obtained are between 270 and 360 degrees, the servomotor will remain at 0 degrees. In this way we avoid any unwanted rotation.

Finally, we have applied a modification to be able to open and close the clamps. It considers the position of the yaw because is the one that affects the gripper. Depending the value of the yaw, it will move one gripper while the other remains on the same position. This is done in this way because if the two grippers are closed and are in the 180-degree position, the one at the end will not be able to open because the servomotors only rotate 180 at most. For this reason, we will move the gripper that is not in the end 30 degrees, in this case at 150.

5.2.2.2. Threshold current

As we have seen in the section *Reading torque,* there are a lot of peaks and we are not sure if the value that is going to take is it at the peak, at the descent or at the rise of the curve. The solution carried out was to integrate all the current to accumulate all the values and avoid peaks.

Once we have the integration, what we really want is to observe if there has been a big change of current between a point and its previous. In other words, we want to know the slope of the function. Depending on the value of the slope, it means that we are applying a lot of force and consequently, the current increases, or the reverse situation.

So, we will derive the integration, obtaining which is the maximum force applied in each case. Also, the maximum of the function with hindrance and without hindrance are very different. We will use a value smaller than that of the function where an obstacle is applied, which we will use as a threshold to know when the pen should start to sound and vibrate, warning that the force being applied is excessive. These new values of the current are the ones that are going to be send to the master, to use it as the variable to active the haptic stimulus.

5.2.3. Master and haptic feedback

The master prototype didn't provide any sense of the force being applied and there was no way to decide what kind of movement we wanted the robot UR5e to perform.

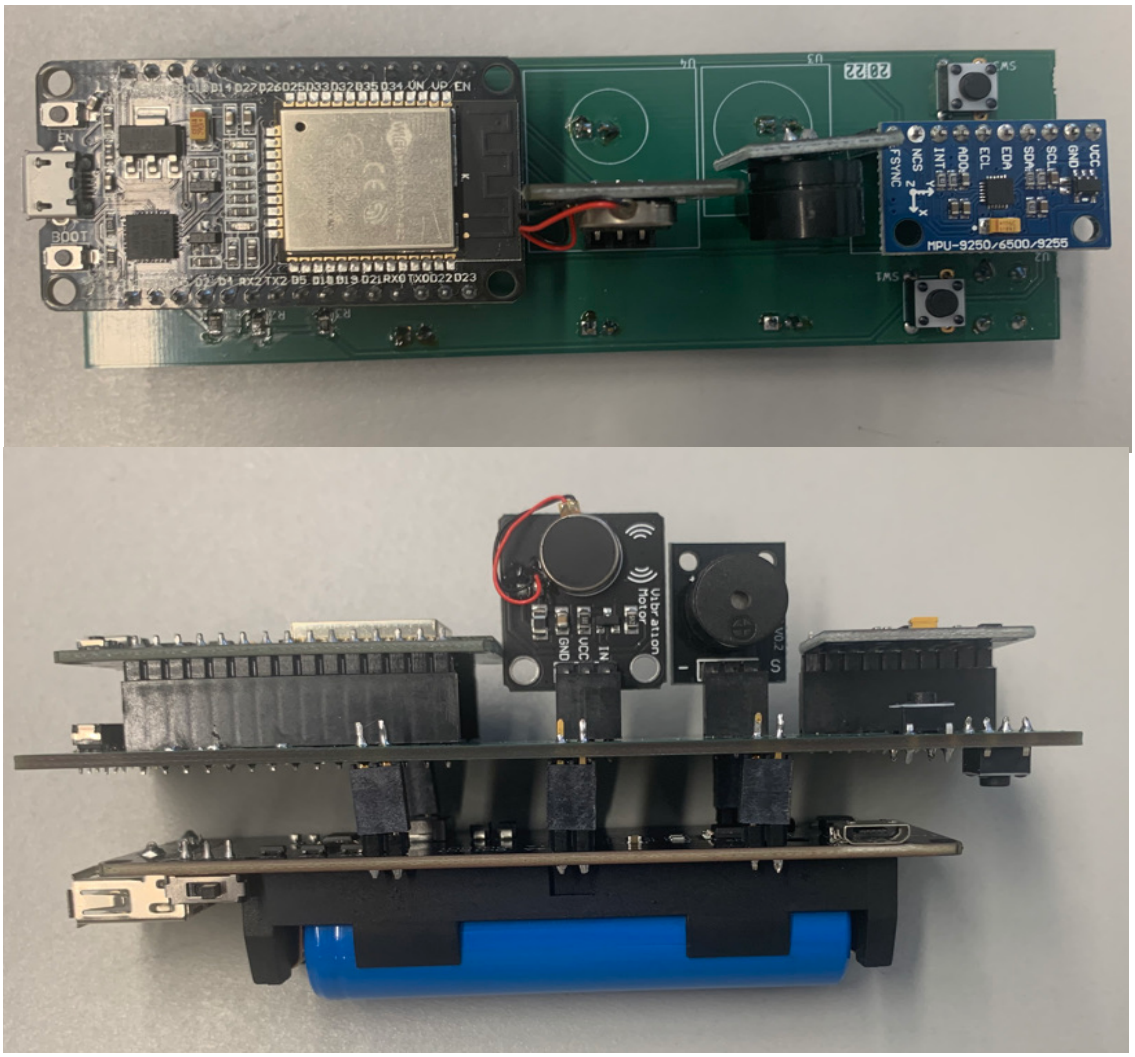


Figure 21. Pen MASTER with an IMU, three buttons, a battery, a buzzer, and a vibrator motor.

To achieve that, another button was included apart from the other two; a battery to allow the feeding of the entire circuit and that it is wireless; and a buzzer and vibrator motor to perform haptic feedback.

5.2.3.1. Haptic feedback

From the slave, we receive the derivative current explained in the last section. Considering this value, it will define the threshold from which a lot of force is being exerted.

The hardware in charge of the haptic stimulus is the buzzer and the vibrator motor. Once the value obtained from the derivative current is higher than the threshold, the haptic stimulus will jump. Both the sound of the buzzer and the

vibration of the vibrator motor will increase in relation to the value of the current. If the threshold has been slightly exceeded, it will not make as much noise as if it has been greatly exceeded. The frequency of both components will depend on the value of the current, meaning that if the current increase also does it the sound and vibration.

5.2.3.2. Modes of action

With the haptic pen, we need to perform all the movements that the Da Vinci robot can do. To achieve that, we are going to create different modes. Each mode will perform one movement and we can change modes thanks to the third button.

- Mode 0: it will perform the movement from the tip of the EndoWrist, the gripper. In that case, we need to rotate the servomotors according to the roll, pitch and yaw values from the IMU; and open and close the gripper while we push the first or second button.

```
if (modo == 0) {
  //if mode = 0 we move the gripper
  //we send the imu data (roll, pitch and yaw) and the real
  // state of the buttons to open or close to the servos
  // we send the mode to the computer and
  //the buttons don't matter to me because they won't do anything
  ledcWrite(ledchannel, 0);//esp32 led to visualize the change of mode

  dataToServos.roll = rpw[0];
  dataToServos.pitch = rpw[1];
  dataToServos.yaw = rpw[2];
  dataToServos.s1_pinzas = s1Status;
  dataToServos.s2_pinzas = s2Status;

  dataToComputer.roll = rpw[0];
  dataToComputer.pitch = rpw[1];
  dataToComputer.yaw = rpw[2];
  dataToComputer.s1_status = s1Status;
  dataToComputer.s2_status = s2Status;
  dataToComputer.modo = 0;
```

Figure 22. Code for the mode 0, movement of the gripper.

- Mode 1: it will simulate the entry and exit of the EndoWrist over the patient's orifice. In that case, we will move the EndoWrist along the Z-axis. However, when this mode is activated, we don't want that the EndoWrist gripper continues to move according to the IMU values. That's the reason why we store and send the last values of the IMU obtained in the mode 0 to the servomotors. Meaning, that they will remain in the position that they were in at the last moment of mode 0. Besides, the first and second buttons will be the ones that indicate the rise and fall of the tool. Therefore, we have to disable them for the servomotors

```
} else if (modo == 1) {
  //if mode = 1 we move the arm in z axis
  //we send the data of the previous imu to the servos
  //so that they remain still and the buttons as if they were not pressed
  // to the computer we send the mode and the real state of
  //the buttons to raise or lower the position
  ledcWrite(ledchannel,102);// esp32 led to display the mode change

  dataToServos.roll = dataToServos.roll;//we send the last stored data so that the servos do not move
  dataToServos.pitch = dataToServos.pitch;
  dataToServos.yaw = dataToServos.yaw;
  dataToServos.s1_pinzas = HIGH; //we send high to the gripper servos so that they do not open or close
  dataToServos.s2_pinzas = HIGH;

  dataToComputer.roll = dataToComputer.roll;
  dataToComputer.pitch = dataToComputer.pitch;
  dataToComputer.yaw = dataToComputer.yaw;
  dataToComputer.s1_status = s1Status;
  dataToComputer.s2_status = s2Status;
  dataToComputer.modo = 1;
```

Figure 23. Code for the mode 1, movement of the arm in the z-axis.

so that they do not open or close the clamps. The values send to the computer from the master will be the status of the two buttons, to raise or lower the tool with respect to the Z-axis.

```
} else if (modo == 2){
  //if mode = 2 we move the arm in all axes
  //we send the data of the previous imu to the servos so
  //that they remain still and the buttons as if they were not pressed
  // we send the mode and the state of the buttons to the computer we don't care
  ledcWrite(ledchannel,1024);

  dataToServos.roll = dataToServos.roll;
  //we send the last stored data so that the servos do not move
  dataToServos.pitch = dataToServos.pitch;
  dataToServos.yaw = dataToServos.yaw;
  dataToServos.s1_pinzas = HIGH;
  //we send high to the gripper servos so that they do not open or close
  dataToServos.s2_pinzas = HIGH;

  dataToComputer.roll = rpw[0];
  dataToComputer.pitch = rpw[1];
  dataToComputer.yaw = rpw[2];
  dataToComputer.s1_status = HIGH;
  dataToComputer.s2_status = HIGH;
  dataToComputer.modo = 2;
}
```

Figure 24. Code for the mode 2, movement of the arm to orientate the EndoWrist.

- Mode 2: Correspond the movement of the robot in all axes. It performs the movement to orient the tool in one direction or another, depending on where the orifice through which the patient is to be operated is located. The end-effector will remain fixed in one position, and taking the values from the IMU, the robot will be placed in the appropriate position to orient the EndoWrist. This is because during operations the EndoWrist must be oriented to enter through a different zone or shape in each case. With the servos it will happen the same as in mode 1 because we do not want them to move. In that case, we store and send the last values of the IMU obtained in the mode 0 to the servomotors, so they will be stopped. To the computer, we will send the values of the roll, pitch and yaw. The state of the two buttons is disabled because we won't be using them for this mode.

5.2.4. RoboDK simulation implementations

First of all, the simulation in the computer, to later be executed in real-world, its components must be as close as possible to what we have in real life. The EndoWrist we had in RoboDK did not have the same dimensions as the one we had. We created the EndoWrist tool and the two parts of the casing that hold it with FreeCAD, using the same dimensions as the real ones.

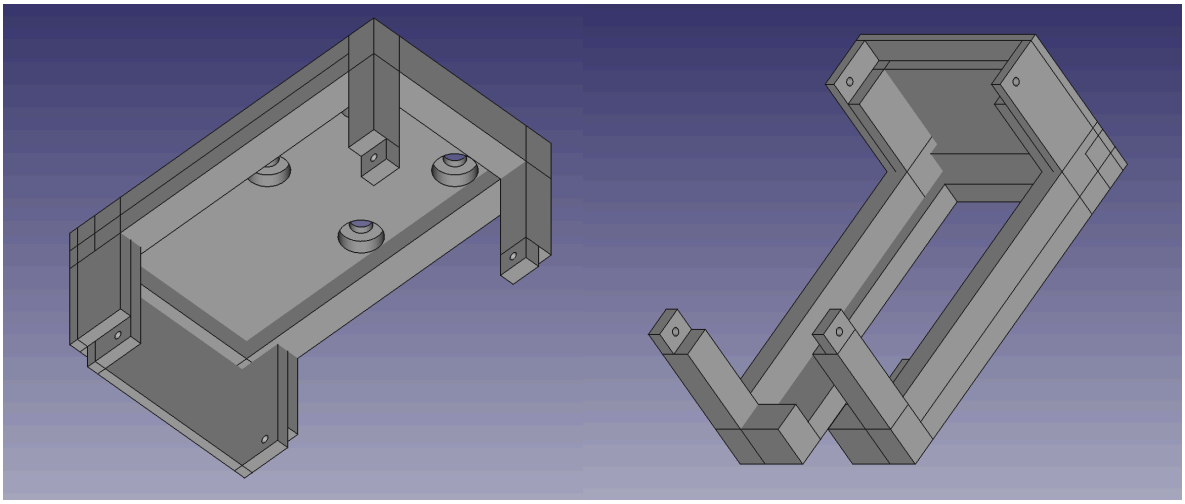


Figure 25. Casing that holds the EndoWrist created in FreeCAD.

These pieces were combined in RoboDK, using the function *Compound* that joins different figures. After that, they were defined as a tool and were placed at the end of joint 6 as an end-effector.

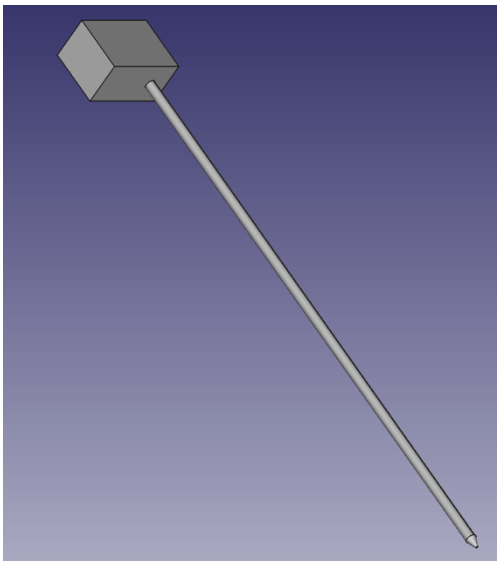


Figure 26. EndoWrist created in FreeCAD.

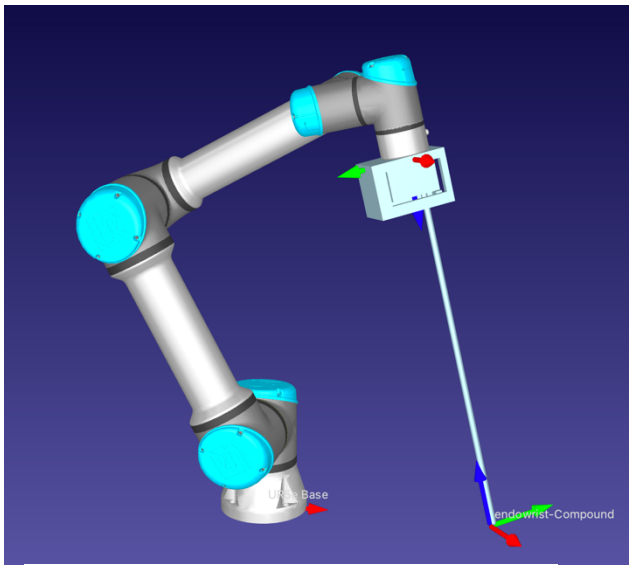


Figure 27. UR5e with the designed EndoWrist in RoboDK.

We must bear in mind that the robot will carry out one movement or another, depending on the mode that we have selected. This must be implemented so that the simulation is done correctly and can be performed in real life.

The mode 0 is the one that we created previously which will move only the gripper of the EndoWrist. Switching to the mode 1, it will go up or down following the Z-axis of the tool.

Activating the mode 2, the UR5e robot will move to change the orientation of the tool fixing the position of the end-effector. To achieve that we need to know the position of the robot, these values will be used in the translation matrix to move

```
if modo == 0: #moving the gripper
    gripper_pose =  transl(X,Y,Z) * rotx(pi) * rotz(W) * roty(P) * rotx(R) * roty(0)
    #tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix)
    gripper.setPose(gripper_pose)

elif modo == 1: #moving through Z axis the robot


    position=robot.Pose()
    xyzabc = Pose_2_KUKA(position)
    if not s1_status: #s1 button pressed, going up
        if ((float(xyzabc[2]) >50) & (float(xyzabc[2]) <500)): #upper limit
            approach = robot.Pose() * transl(0,0,1)
            robot.MoveL(approach, False)
        if (float(xyzabc[2]) <50): #lower limit
            (xyzabc[2])=51
            target=KUKA_2_Pose(xyzabc)
            robot.MoveL(target, False)
    elif not s2_status: #s2 button pressed, going down
        if ((float(xyzabc[2]) >50) & (float(xyzabc[2]) <500)): #upper limit
            approach = robot.Pose() * transl(0,0,-1)
            robot.MoveL(approach, False)
        if ((float(xyzabc[2]) >500)): #lower limit
            (xyzabc[2])=499
            target=KUKA_2_Pose(xyzabc)
            robot.MoveL(target, False)

elif modo == 2: #positioning the robot
    tcp_pose = transl(Xr,Yr,Zr) * rotz(W) * roty(P) * rotx (R)
    if robot.MoveL_Test(robot.Joints(),tcp_pose) == 0:
        robot.MoveL(tcp_pose)
    else:
        print('The robot cannot reach target because it is out of reach or causes collision.')
```

Figure 28. Program to perform all the movements in RoboDK.

the UR5e; and the values of the IMU, to rotate the UR5e changing the values of its position. To ensure that the position that the robot will take is safe, the function *.MoveL_Test()* is used. This function calculates if the position to which the robot will move with the new values is safe and there will be no collision between the links of the UR5e. If it sends 0 as a result, it means that the movement is secured; any other number means that there will be a collision between the bot's links, or that the bot cannot reach that position. To know the axes where the rotations will be made, a new fixed frame is placed at the tip.

Besides, the torque values generated during the EndoWrist movement process are stored and transferred to an Excel file in case it is necessary to carry out any type of analysis. What's more, a live plot is generated showing the actual torque values of each servomotor.

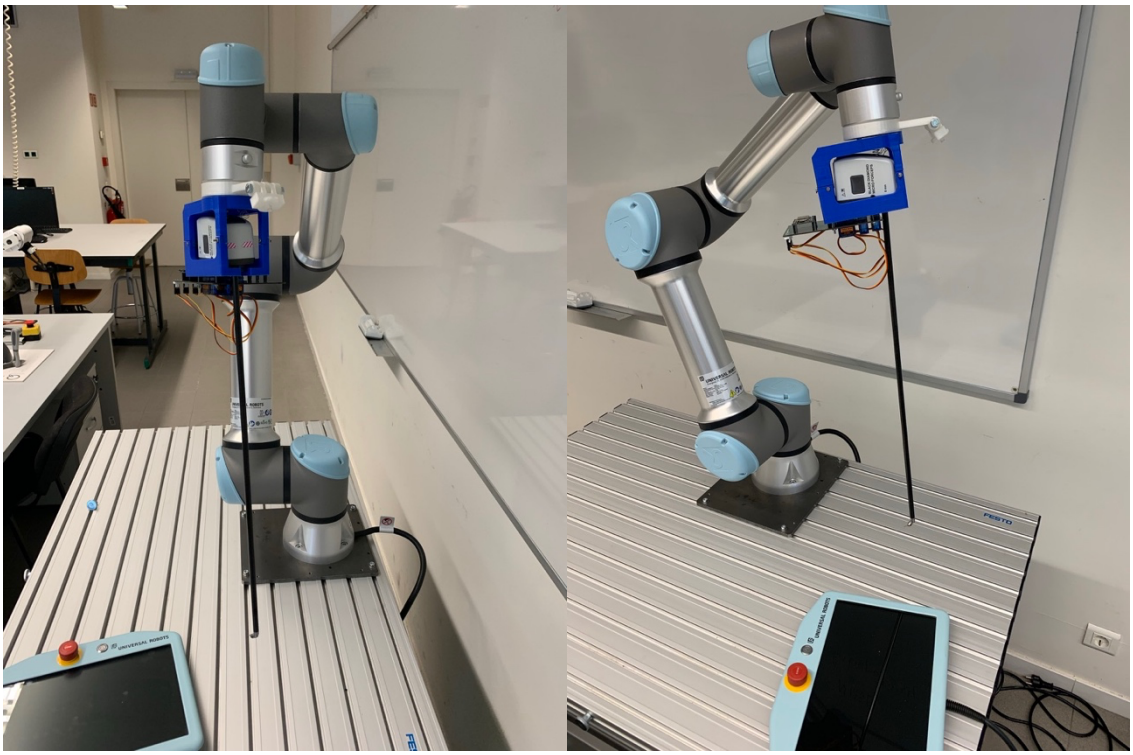


Figure 29. UR5e with the EndoWrist and the designed servomotor PCB.

6. EXPERIMENTAL VALIDATION

6.1. Instantaneous current measurement in servomotors

In the following graphs, we can observe a representation of the intensity going through the servomotor when we move them to a determined position. The servomotor tries to go to this position at a maximum velocity, that's why we have an initial spike. Then this value is decreasing until, at the end, it tries to correct its stationary error.

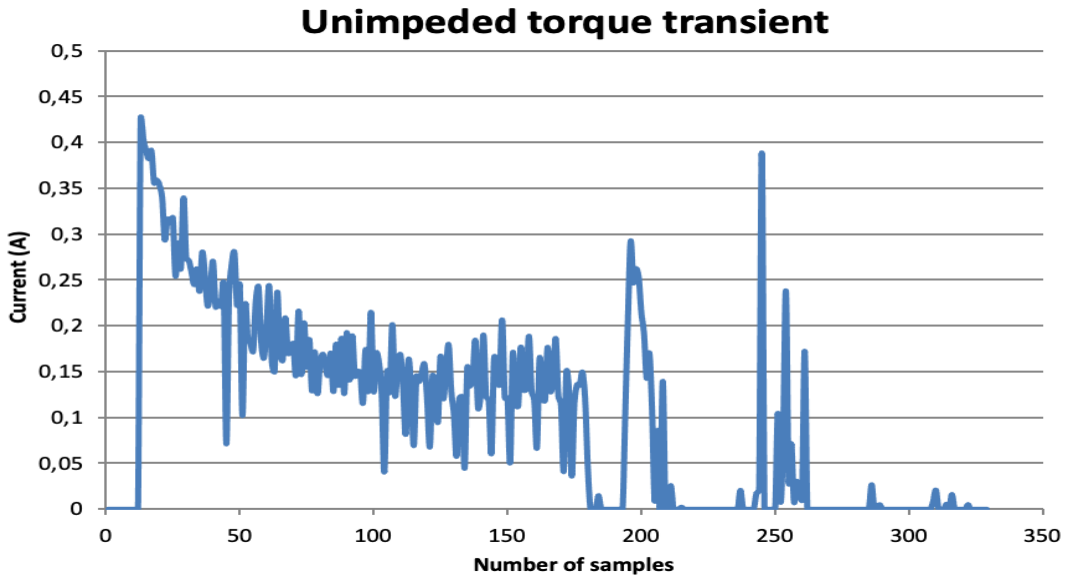


Figure 30. Current that goes through the servomotor when there is no obstacle.

When there is an obstacle, we can observe the current is a little bit higher because is trying to get into this position. What's more, there are pulses because it tries to reach the position, stops, tries again and so on periodically until it reaches the target.

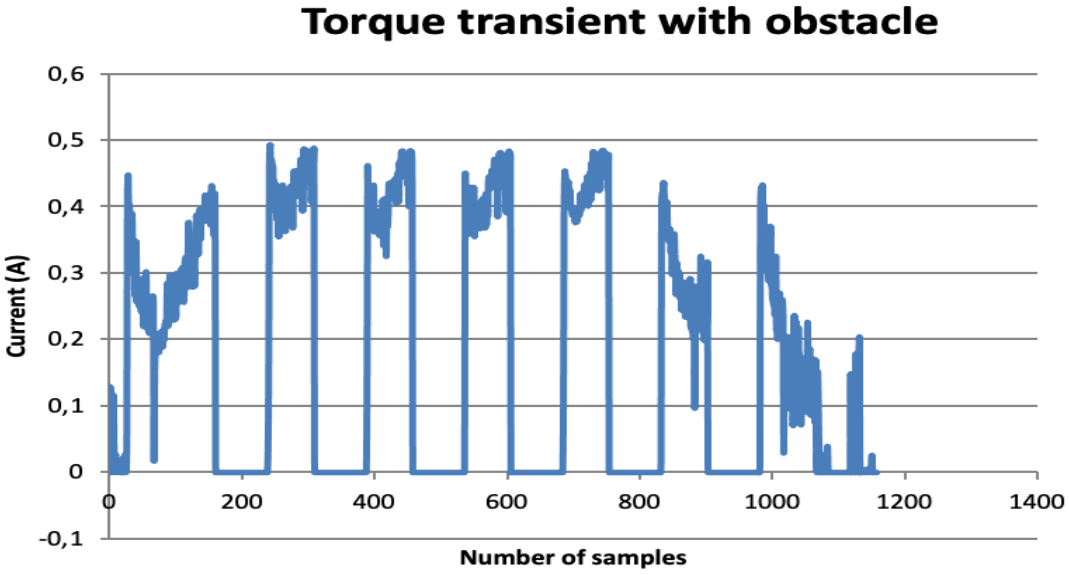


Figure 31. Current that goes through the servomotor when there is an obstacle.

First, the maximum value is practically the same when there is an obstacle or not, therefore it is difficult to differentiate both. It is quite noisy with many peaks; therefore, we will apply a filter integrating the current with respect to time. Because of this, we now get the electric charge, not the current.

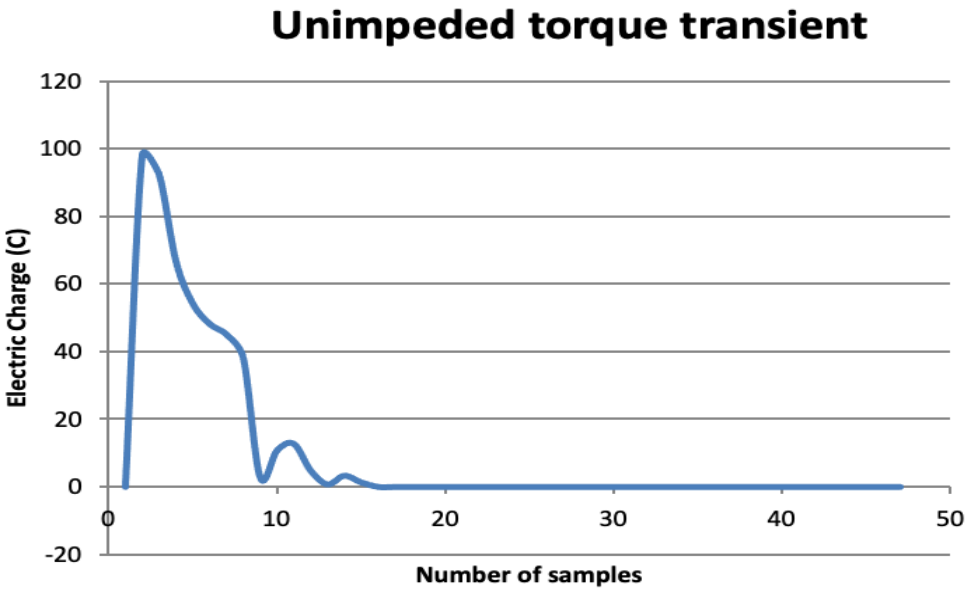


Figure 32. Filter of the current, without obstacle.

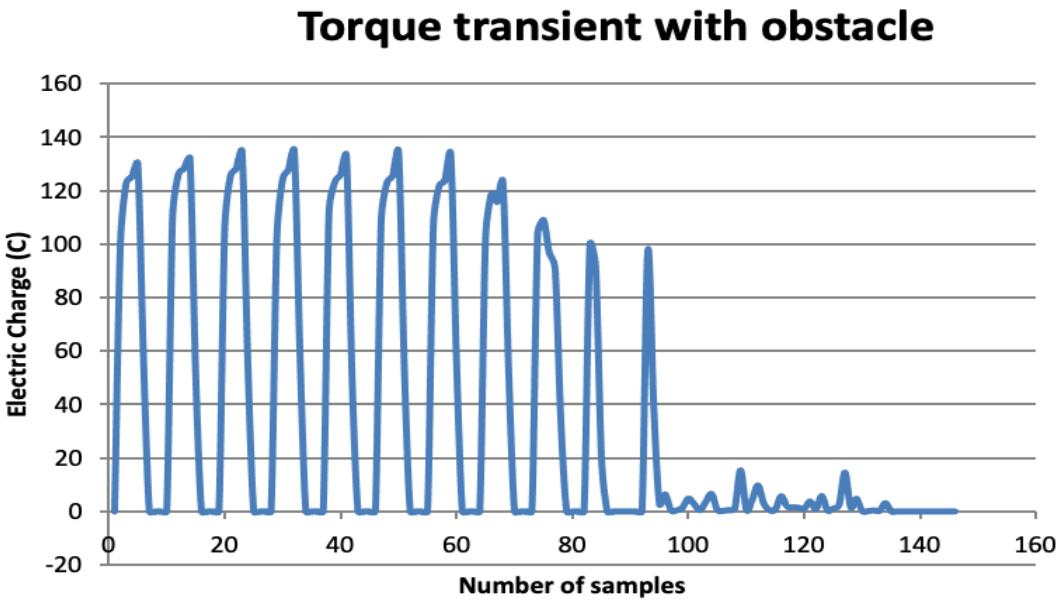


Figure 33. Electric Charge when there is an obstacle.

Thanks to the integration, the magnitude is significantly different when there is an obstacle or not. Helping us to differentiate each case.

6.2. Average current measurement

In this part we will properly measure the average torque, according to the previous experimental results, a possible method could be to integrate first the measured current and then derivate to extract the slope representing a proportional measure of the average torque.

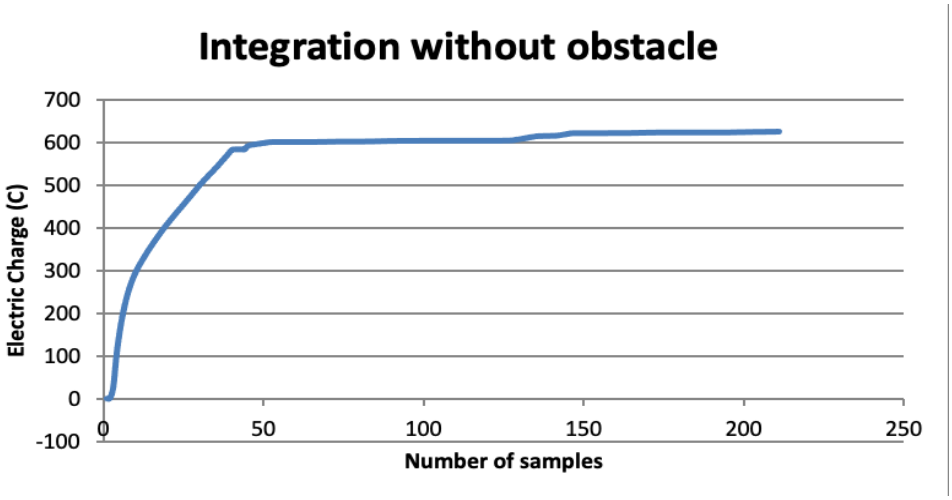


Figure 34. Integration of the current when there is no obstacle.

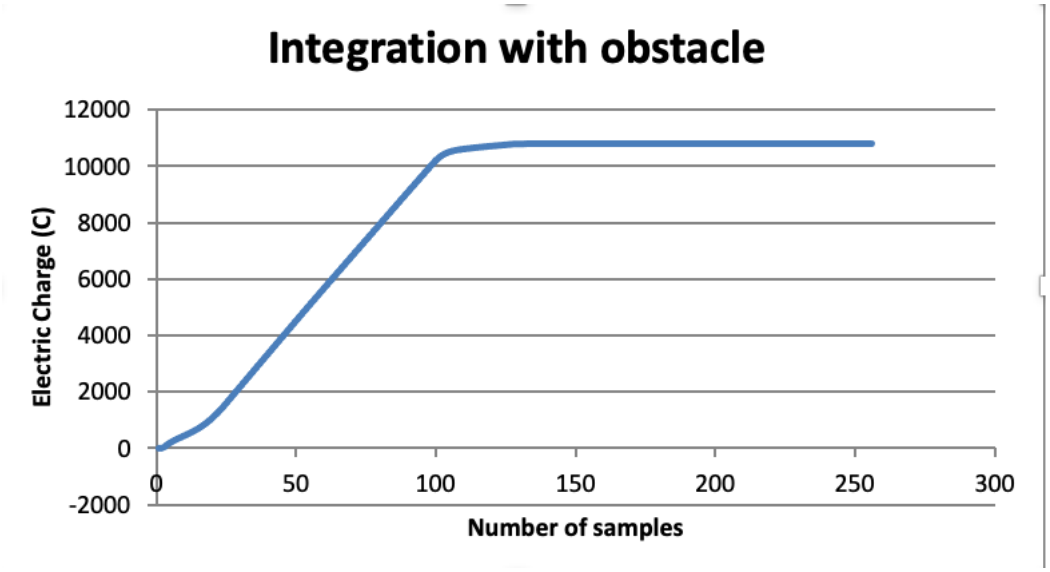


Figure 35. Integration of the current when there is an obstacle.

Comparing both graphs, when integrating the current, what we are doing is accumulating the values of the current. For this reason, as before, when there is an obstacle, its magnitude is greater because it needs more current to overcome the force it is facing. When there is no obstacle, it needs less current, only the necessary to reach the desired position at maximum speed.

After that, we perform the derivative to obtain which is the maximum force applied on the tissue or surface.

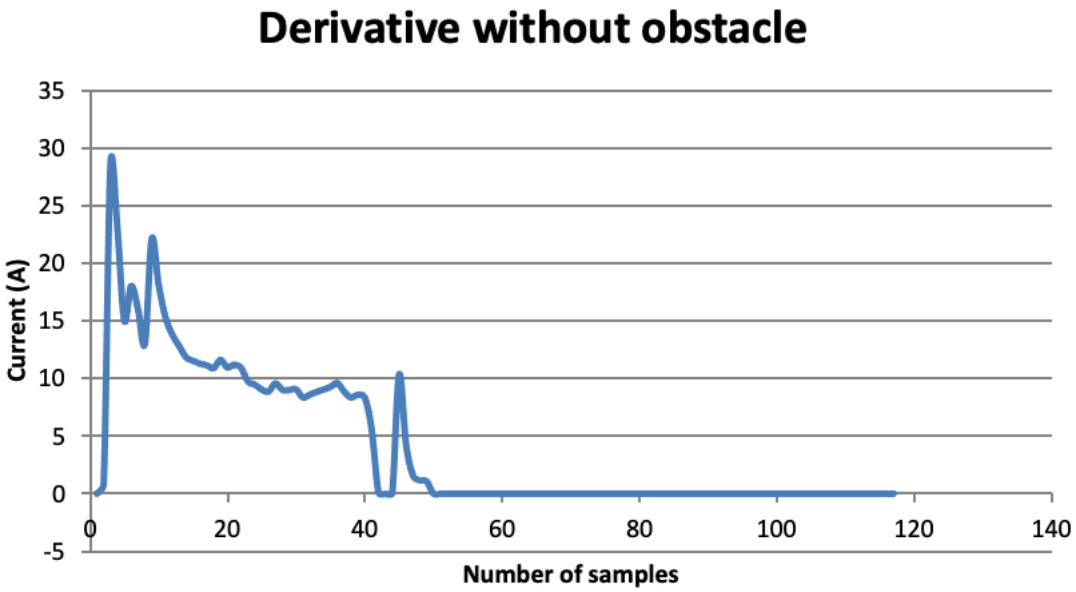


Figure 36. Derivative of the integrated current when there is no obstacle.

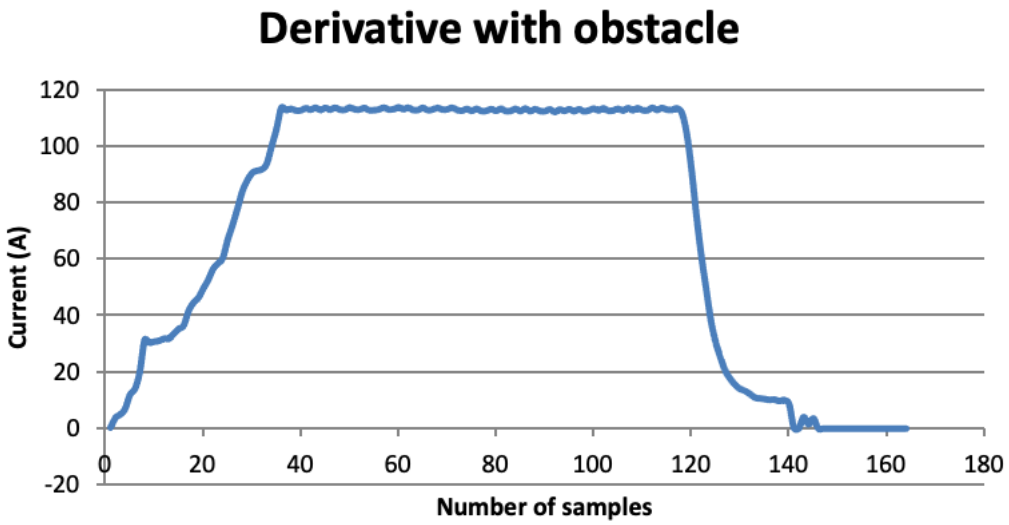


Figure 37. Derivative of the integrated current when there is an obstacle.

When there is an obstacle, we observe the value of the slope of the integrated function previously seen. When the value is increasing, it means that it's increasing its force to move the servomotor and arrive at the desired position. Comparing the graph of the obstacle without the obstacle, the difference is magnitude and plateau. As we have been mentioning, when there is an obstacle, it needs more current to overcome that obstacle. Increasing its value until it arrives to its maximum, the plateau, moment in which it is showing us that it is applying a lot of force on the tissue or obstacle in question. These values represent well the exerted force and will be used as threshold.

7. EXECUTION PLAN

In this section the phases and the time of each part are presented. Including the paths that we have followed to obtain the objectives of the project.

7.1. Work Breakdown Structure (WBS)

Work Breakdown Structure is a tool that divide the project in smaller tasks to obtain more productivity and manageable. This WBS is created to define the different steps and assignments to develop the project.

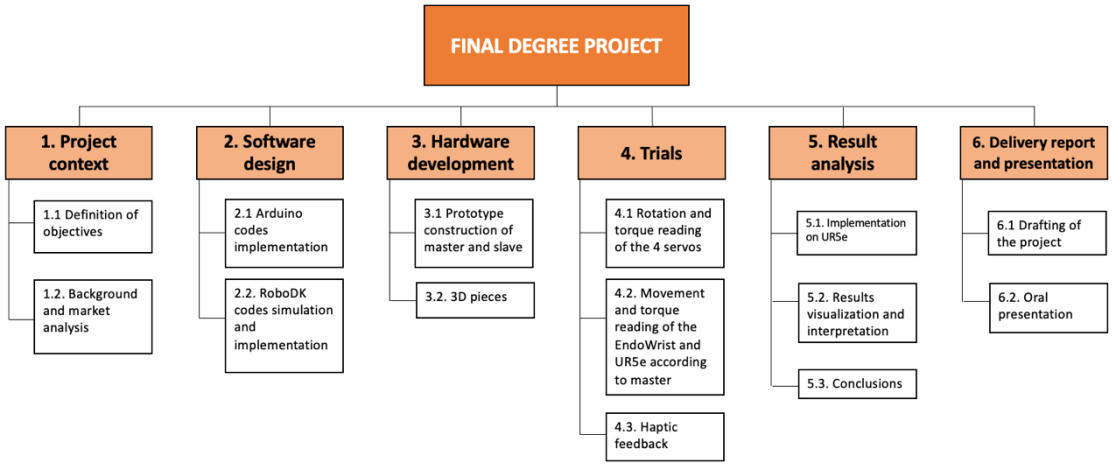


Figure 38. Work Breakdown Structure of the project.

7.2. Program Evaluation and Review Techniques (PERT)

The PART diagram is a representation of the tasks made in a project, and the different connection and dependencies between each one. The following table shows the different activities defined in WBS, its duration and connection between them.

Activity	*Identifier*	*Duration (weeks)*	*Previous activities*	*Following Activities*
1.1 Definition of objectives	A	1	-	B
1.2. Background and market analysis	B	2	A	C -D
2.1 Arduino codes implementation	C	2	B	E
2.2. RoboDK codes simulation and implementation	D	2	B	F
3.1 Prototype construction of master and slave	E	2	C	G
3.2. 3D pieces	F	3	D	G
4.1 Rotation and torque reading of the 4 servos	G	2	E - F	H - I

4.2. Movement and torque reading of the EndoWrist and UR5e according to master	H	3	G	K
4.3. Haptic feedback	I	3	G	J
5.1. Implementation on UR5e	J	2	I	K
5.2. Results visualization and interpretation	K	2	H - J	L
5.3. Conclusions	L	1	K	N
6.1 Drafting of the project	M	During all project	-	-
6.2. Oral presentation	N	2	L	-

Table 10. PERT diagram table.

Considering the information on the diagram, we perform the representation of the PERT. On each node we have on the left the *'Early Time',* which is the minimum time needed to reach a node, and on the right, the *'Late Time'* that is the maximum time it can take to reach a knot without delaying the project. The purple lines will determine the critical path, where there is the minimum time to perform the project.

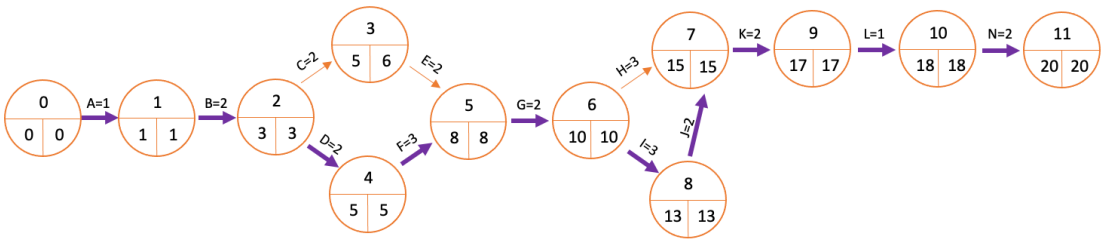


Figure 39. PERT diagram representation.

7.3. GANTT diagram

A GANTT chart is used in a project management to represent the activities carried out with respect to the time spent. On the first column it is shown the identifier of each activity, which are defined on the PERT. Each activity is displayed on a row and the bars in each row represent the duration of the activity.

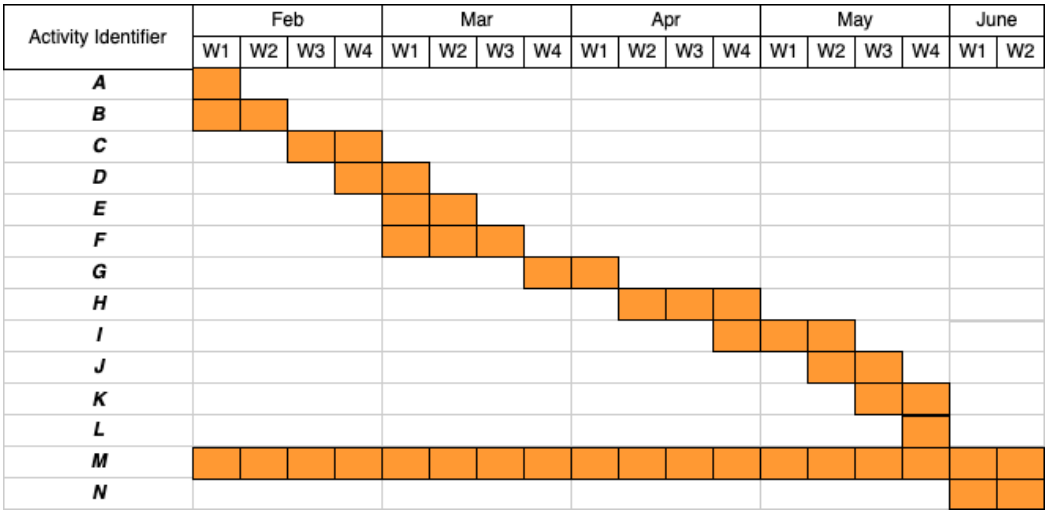


Figure 40. *GANTT project diagram.*

8. TECHNICAL VIABILITY

In this section, an analysis was performed to determine the technical feasibility of the project. To execute this analysis, we have done a SWOT, presenting the strengths, weaknesses, opportunities, and threats.

8.1. SWOT analysis

Performing and internal and external analysis of the project, the best advantages, opportunities and the project lacks and threats are presented on this strategic planning and management technique.

	Helpful to achieving the objective	**Harmful** to achieving the objective
Internal Factors	**Strengths** • Calculation of the force applied • Low-cost	**Weaknesses** • Small delay • Not knowing the direction where it is exceeding force
External Factors	**Opportunities** • Implementation on different robotic arms • Increase in minimally invasive surgeries	**Threats** • Solution search by different companies • Emerging technology

Table 11. SWOT table.

Strengths

The positives aspects of this project are that we have implemented an haptic feedback system with cheap components. The integral of the current and its subsequent derivation gives us a value that represents quite accurately the force that we are applying to it. What's more, the UR5e robot can be controlled just by a pen user interface that provides different types of movements.

Opportunities

This area is emerging because there is no real solution on the market for this problem. Meaning, that different companies and sectors are interested in obtaining a solution to this problem and would be willing to finance a real solution to this.

What's more, the minimally invasive surgery is being imposed on traditional surgery, which further increases the need to find a way to solve this problem.

Weaknesses

The transmission of data between the different components means that there is a small delay between the moment we execute the movement with the pen, and the subsequent execution in the UR5e.

Furthermore, the haptic feedback system can be improved. Although the way of measuring the force is the correct one, the different haptic stimulus on the pen doesn't provide a sense of the direction where we apply the force. The sound and vibration are proportional to the applied force, but if we closed our eyes we would not know in which area is where we are damaging the tissue.

Threats

We have commented that the need to find a solution is increasing. Although this is an opportunity, it also increases the risk of threat as different companies are also looking for ways to solve this problem. So, there is quite a bit of direct competition.

9. ECONOMIC FEASIBILITY

In this section we are going to calculate the total cost of the project. Considering the software programs, this has not involved any cost because many of them were free or we obtained a free license thanks to being university students.

However, an important part of the budget has been allocated to hardware components and the hours dedicated from the student and the director of the project. Regarding this topic, in the *'Boletín Oficial del Estado – Ministerio de Trabajo, migraciones y seguridad social'* [54], it is defined the minimum salary for a junior and senior biomedical engineer, 20€/h and 30€/h, respectively. As it is defined on the teaching level, the student must devote about 300 hours to the project, and about 100 hours have been considered for the director. What's more, PCB boards made by my partner are also considered in the budget.

Concept	Units	Unit Price	Total Cost
SOFTWARE			
Arduino	1	Free	0€
RoboDK	1	Free	0€
Python	1	Free	0€
FreeCAD	1	Free	0€
HARDWARE			
UR5e robot	1	30.851€	30.851€
EndoWrist	1	4500€	4.500€
IMU MPU-9250	1	12,22€	12,22€
ESP32 board	3	25€	75€
Servomotors Sunfounder SF 180M	4	13€	52€
Buzzer HW-508	1	3€	3€
Vibration Motor PWM Vibration Motor Module	1	5,9 €	5,9 €
Push Button COM 00097	3	0,32 €	0,96€
Computer	1	1000€	1000€
PCB SERVOMOTORS	1	12,01€	12,01€
PCB IMU	1	13,69€	13,69€
PROJECT DEVELOPMENT			
Student	1	20€/h	6.000€
Director	1	30€/h	3.000€

TOTAL	45.525,78€

Table 12. Final Degree Project Cost

In the table above, we can observe the total cost of the project. Around 75% of the budget is destined to the robotic arm and EndoWrist. For this reason, this robot is chosen since it is present in the laboratory and buying a new one would mean an extra cost.

10. REGULATIONS AND LEGAL ASPECTS

Robotics in healthcare presents an important number of potential benefits; however, there are several risks that needs to be considered by the manufacturer and supplier. Specific regulatory frameworks have been developed to ensure efficacy and safety for the patient. In the regulatory framework, robot will need to fulfill some standard to be approved by FDA and Conformité Européene.

FDA (Food and Drug Administration) regulates devices to provide reasonable assurance of their safety and effectiveness for their intended uses. It categorizes and classify the different medical products based on the invasiveness and risk to the patient. We must highlight that FDA neither provide accreditation nor training for physician. The implementation of training is responsibility of the own physician, manufacturer, and healthcare facilities. FDA also provides another way to categorize, regarding the software used in the medical device. If the software is integrated on the medical device, it would be approved in the same classification as the product. If not, it could be approved separately and have a different classification.

In the case of European regulations, all robots will go through Conformité Européene (CE) Mark certification under the European Medical Devices Directive. If the products are approved, the letters "*CE"* will appear; meaning that the product sold on the European Economic Area fulfill high safety, health, and environmental protection requirements.

Standardization is the process of implementing several technical bases on the consensus of different parties, that need to be fulfilled for the subsequent commercialization of the product. In this section we find ISO and IEC safety standards. ISO establish a management system to address controls on quality and security management. On the other hand, IEC are international standards grouping electrical, electronic, and related technologies. One of the regulations stablished by IEC is 80601-2-77:2019. It is the standard for medical electrical equipment that is applied to the basic safety and essential performance for robotically assisted surgical equipment and systems. It covers the issues related to robot movement, electrical and other functional connections; and related to the effect of unintended collision with other objects [55].

What's more, it is a Final Degree Project, which means that must follow the regulations described on "Normes generals reguladores dels treballs de fi de grau de la Universitat de Barcelona", specifically those referred to the degree of Biomedical Engineering.

10.1. Hardware of medical devices

Medical device is a product, as can be an instrument, machine, or implant, this is used for diagnosis, prevention, and treatment of diseases. According to this definition, we can include surgical robots on this group [56]. Some regulations regarding to the safety and quality are:

- ISO 13485:2016: Manage quality throughout the life cycle of a medical device. It involves the design, production, installation and servicing of medical devices and related services.
- ISO 14971:2019: It specifies the process for risk management of medical devices. This standard focuses on processes to identify and evaluate risks associated to the medical device and implement risk controls.
- ISO 11135:2014: Required for the development, validation, and routine control of sterilization process for medical devices. It specifies the symbols

that medical device manufacturers will use to show the information based on sterilization processes.

- ISO 10993-1:2018: Biological evaluation of medical devices. The standard is applied for devices that have direct or indirect contact with a patient and it defines the principles of the biological evaluation of the medical device in a range of risk management framework.

10.2. Software of medical devices

As we have mentioned, the medical devices incorporated a software which, indeed, also needs a regulation and be within a legal framework [57]. Some regulations applied to the software of the medical device are the following ones:

- IEC 62304:2015: it set downs a system for the processes and activities that can happen through the lifecycle of the medical device software. The standard defines the lifecycle for Software as a Medical Device (SaMD).
- IEC 80002-1:2009: it offers a guidance to correctly apply the conditions of ISO 19741. The technical report helps to understand the professionals which requirements are needed to complete ISO 14971.

11. CONCLUSIONS

The main objective of this project was created haptic feedback system using a UR5e robot and a real EndoWrist tool. For that we divided the project on 3 components, the pen user interface, EndoWrist maneuverability and Communication and supervision module. For the first two we have had to create different PCB boards with the necessary components to meet the objective. For this project, we have learned how the different robotics arms works and how to implement haptic feedback system.

Here we will discuss the fulfillment of the initial objectives proposed and some recommendations and improvements for future implementations.

11.1. Objectives fulfillment

We have achieved the transmission of the roll, pitch and yaw values to the servomotor and to the UR5e robot via WIFI. The wireless communication between the three main components has been executed correctly thanks to the ESP32.

What's more, we have performed the three desired movements. In the first case, we have rotated the gears of the EndoWrist moving the servomotors according to the motion of the surgeon-hand. However, the printed pieces were separated in two. To join them, it has been done with double-sided adhesive, which when a force is applied on the end-effector of the EndoWrist, makes it move a little. We have found that many times the pieces separated or did not stay together enough because the adhesive did not hold up as necessary. This has affected the movement of the gripper which is not executed perfectly imitating the movements of the surgeon.

The second one motion was the raising and lowering of the EndoWrist, mimicking the entry and exit of the orifice on the patient. First, it didn't move very smoothly and made very sudden stops. By lowering the range that had to move each time the button was pressed, we have achieved a very fluid movement that is performed perfectly.

The last movement, regarding to the motion of the UR5e change the orientation of the tool considering the end-effector of the EndoWrist as fixed position. At the beginning, we were unable to perform this move because the functions used were the wrong ones. Finally, making sure that there is no collision between the different links and knowing the reference axis on which to rotate, the movement is carried out correctly. However, we have to move the pen very smoothly, otherwise it will make a very sudden movement. Thanks to the collision limits, you will never get into a very awkward position.

Another achievement has been obtained the force that is exerted on the tissue by integrating and then deriving the current received from the servomotors. It is a parameter that shows how much tension we are applying on the tissue, and it increases or decreases depending on the force applied. In addition, the correct

design of the anchoring pieces, both EndoWrist and of the servomotor, have been necessary to be able to carry out this measurement. Also including the design of the EndoWrist and its casing, so that the simulation was as close as possible to the real world.

11.2. Future improvements

On the other hand, different implementations have not been fulfilled as expected. The transmission of data between the different components has caused a small delay of around 1 second when executing rotations and movements. It limits us a little the correct operation and avoids 100% comfort when executing the movements. However, in the future it could be reduced by changing the *delay* functions present in the code by *millis*.

Apart from that, the printed pieces were separated in two. To join them, it has been done with double-sided adhesive, which when a force is applied on the end-effector of the EndoWrist, makes it move a little. Ideally, each element would be created in 3D printing in one piece, without the need to use adhesive.

What's more, an easier anchorage between the UR5e robot, EndoWrist and servomotors than the current one. If one of the pieces separates, we have to remove and separate all the devices by removing the screws that hold them. An idea that would facilitate this would be the union of an iron in one of the ends and separate the components by an opening that will simulate the opening of a mouth. In this way we reduce the screws to remove and the time it takes.

Finally, the vibration exerted on the pen user interface is proportional to the applied force but does not offer a sense of the direction in which this tension is exerted.

11.3. Personal conclusions

Related to personal conclusions, this project has been a great opportunity for me to improve my skills in programming and 3D design, among other features. The chance to work with another student it helped me improve my group work and learn about its branch, in addition to knowing firsthand that there is a lot of interdisciplinarity in research groups.

12. REFERENCES

[1] *Minimally Invasive Surgery*. (2019, 25 noviembre). Yale Medicine. https://www.yalemedicine.org/conditions/minimally-invasive-surgery

https://www.yalemedicine.org/conditions/minimally-invasive-surgery

[2] Siddaiah-Subramanya, M., Tiang, K. W., & Nyandowe, M. (2017). A New Era of Minimally Invasive Surgery: Progress and Development of Major Technical Innovations in General Surgery Over the Last Decade. *Surgery journal (New York, N.Y.)*, *3*(4), e163–e166. https://doi.org/10.1055/s-0037-1608651.

[3] Mohiuddin K, Swanson SJ. Maximizing the benefit of minimally invasive surgery. J Surg Oncol. 2013 Oct;108(5):315-9. doi: 10.1002/jso.23398. Epub 2013 Aug 23. PMID: 24037974. https://pubmed.ncbi.nlm.nih.gov/24037974/

[4] Lanfranco, A. R., Castellanos, A. E., Desai, J. P., & Meyers, W. C. (2004). Robotic surgery: a current perspective. *Annals of surgery*, *239*(1), 14–21. https://doi.org/10.1097/01.sla.0000103020.19595.7d

[5] Ngu, J. C., Tsang, C. B., & Koh, D. C. (2017). The da Vinci Xi: a review of its capabilities, versatility, and potential role in robotic colorectal surgery. *Robotic surgery (Auckland)*, *4*, 77–85. https://doi.org/10.2147/RSRR.S119317

[6] Lane T. (2018). A short history of robotic surgery. *Annals of the Royal College of Surgeons of England*, *100*(6_sup), 5–7. https://doi.org/10.1308/rcsann.supp1.5

[7] C. Martínez-Ramos (2007). Cirugía robótica (I): origen y evolución. http://www.asecma.org/Documentos/Articulos/AE%201_3.pdf

[8] Valero, R. (2011). *Cirugía robótica: Historia e impacto en la enseñanza*. SciELO. https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0210-48062011000900006

[9] George, E. I., Brand, T. C., LaPorta, A., Marescaux, J., & Satava, R. M. (2018). Origins of Robotic Surgery: From Skepticism to Standard of Care. *JSLS : Journal of the Society of Laparoendoscopic Surgeons*, *22*(4), e2018.00039. https://doi.org/10.4293/JSLS.2018.00039

[10] INTUITIVE. Robotic-assisted surgery as a minimally invasive option. https://www.davincisurgery.com

[11] Michael E. Moran (2007). The da Vinci Robot. Epochs in Endourology—Sakti Das, M.D., Section Editor. https://doi.org/10.1089/end.2006.20.986

[12] Mahdi Azizian, May Liu, Iman Khalaji, and Simon DiMaio. “THE DA VINCI SURGICAL SYSTEM” in Intuitive Surgical, Inc. 1020 Kifer Road, Sunnyvale, CA, USA.
https://www.worldscientific.com/doi/pdf/10.1142/9789813232266_0001

[13] *Abex*. (2019). Abex. https://www.abexsl.es/en/da-vinci-robotic-system/da-vinci-xi

[14] Kapoor, S., Arora, P., Kapoor, V., Jayachandran, M., & Tiwari, M. (2014). Haptics - touchfeedback technology widening the horizon of medicine. *Journal of clinical and diagnostic research : JCDR*, *8*(3), 294–299. https://doi.org/10.7860/JCDR/2014/7814.4191

[15] Okamura A. M. (2009). Haptic feedback in robot-assisted minimally invasive surgery. *Current opinion in urology*, *19*(1), 102–107. https://doi.org/10.1097/MOU.0b013e32831a478c

[16] Ashrafian H, Clancy O, Grover V, Darzi A. The evolution of robotic surgery: surgical and anaesthetic aspects. Br J Anaesth. 2017 Dec 1;119(suppl_1):i72-i84. doi: 10.1093/bja/aex383. PMID: 29161400.

[17] C. Faria, W. Erlhagen, M. Rito, E. De Momi, G. Ferrigno and E. Bicho, "Review of Robotic Technology for Stereotactic Neurosurgery," in *IEEE Reviews in Biomedical Engineering*, vol. 8, pp. 125-137, 2015, doi: 10.1109/RBME.2015.2428305. https://pubmed.ncbi.nlm.nih.gov/25955851/

[18] Visconti TAC, Otoch JP, Artifon ELA. Robotic endoscopy. A review of the literature. *Acta Cir Bras*. 2020;35(2):e202000206. doi:10.1590/s0102-865020200020000006 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7184939/

[19] Kent, C. (2022, 3 marzo). *Robotic surgery: a race to the top*. Medical Device Network. https://www.medicaldevice-network.com/analysis/da-vinci-surgical-robot-competitors/

[20] Soler, E. (2020, 27 mayo). *Global R+D trends in Surgical Robotics*. I3PT Parc Taulí. https://www.tauli.cat/institut/actualitat/vigilancia-tecnologica/2020/05/glaboal-rd-trends-in-surgical-robotics/

[21] *Sistema de Cirugía Robótica Asistida Hugo™ | Medtronic (CL)*. (2020). Medtronic. https://www.medtronic.com/covidien/es-cl/robotic-assisted-surgery/hugo-ras-system.html

[22] *Minimal access. Maximum gain*. (2021). CMR Surgical. https://cmrsurgical.com/versius

[23] *Understanding the Benefits of the ROSA® Surgical Robot: Metro Orthopedics & Sports Therapy: Orthopedic Surgeons*. (2021). ROSA Surgical Robot. https://www.mostsportsmedicine.com/blog/understanding-the-benefits-of-the-rosa-surgical-robot

[24] M. (2020). *Spine & Orthopaedic Products - Mazor X Steath Edition*. Medtronic. https://www.medtronic.com/us-en/healthcare-professionals/products/spinal-orthopaedic/spine-robotics/mazor-x-stealth-edition.html

[25] *Robotic Endoscopy*. (2021). Auris Health. https://www.aurishealth.com

[26] Pietrabissa, A. (2013, 1 febrero). *Robotic Surgery: Current Controversies and Future Expectations | Cirugía Española (English Edition)*. ElSevier. https://www.elsevier.es/en-revista-cirugia-espanola-english-edition--436-articulo-robotic-surgery-current-controversies-future-S2173507713000379

[27] *Pricing and Licensing*. (2019). MATLAB & Simulink. https://es.mathworks.com/pricing-licensing.html

[28] *Seleccione su edición de LabVIEW*. (2020). NI. https://www.ni.com/es-es/shop/labview/select-edition.html

[29] *What is Arduino?* (2018). Arduino. https://www.arduino.cc/en/Guide/Introduction/

[30] *¿Qué es LabVIEW? Programación gráfica para pruebas y medidas*. (2018). NI. https://www.ni.com/es-es/shop/labview.html

[31] *ROS: Home*. (2018). ROS. https://www.ros.org

[32] *What Is MATLAB?* (2018). MATLAB & Simulink. https://es.mathworks.com/discovery/what-is-matlab.html

[33] *Welcome to*. (2022, 3 junio). Python.Org. https://www.python.org

[34] RoboDK. (2019). *Simulator for industrial robots and offline programming*. https://robodk.com

[35] *SolidWorks Packages*. (2019). GoEngineer. https://www.goengineer.com/error/405

[36] *AutoCAD*. (2020). AutoDesk. https://www.autodesk.com/products/autocad/overview

[37] *FreeCAD: Your own 3D parametric modeler*. (2019). FreeCAD. https://www.freecadweb.org

[38] *SolidWorks*. (2020). SolidWorks. https://www.solidworks.com

[39] Rosamond, Z. (2020, 28 octubre). *What Is SolidWorks? – Simply Explained*. All3DP. https://all3dp.com/2/what-is-solidworks-simply-explained/

[40] Pauliková A, Gyurák Babeľová Z, Ubárová M. Analysis of the Impact of Human-Cobot Collaborative Manufacturing Implementation on the Occupational Health and Safety and the Quality Requirements. *Int J Environ Res Public Health*. 2021;18(4):1927. Published 2021 Feb 17. doi:10.3390/ijerph18041927 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7922989/

[41] Escalette, G. (2020). *UR5 collaborative robot arm | Flexible and lightweight cobot*. UR5e. https://www.universal-robots.com/products/ur5-robot/

[42] WiredWorkers. (2022, 31 enero). *Universal Robots UR5e | Tough, flexible collaborative robot arm.* https://wiredworkers.io/universal-robots-ur5e/

[43] *Cobots – A Helping Hand to the Healthcare Industry*. (2020). Universal Robots. https://www.universal-robots.com/blog/cobots-a-helping-hand-to-the-healthcare-industry/

[44] Mover4Axis. Robot Arm Mover. https://cpr-robots.com/wp-content/uploads/2020/11/ProductSheet_Mover4_0510-06.pdf

[45] KUKA AG. (2021, 9 junio). *About KUKA*. https://www.kuka.com/en-de/company/about-kuka

[46] KUKA AG. (2020, 16 julio). *KR 360*. https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-360-fortec

[47] Intuitive (2020). Da Vinci Instruments. https://www.intuitive.com/en-gb/products-and-services/da-vinci/instruments

[48] Gastreich, W. (2022, 14 febrero). *What is a Servo Motor?' and How it Works | RealPars*. PLC Programming Courses for Beginners | RealPars. https://realpars.com/servo-motor/

[49] Pao, C. (2019, 13 septiembre). *The importance of IMU Motion Sensors - CEVA's Experts blog*. CEVAâs Experts Blog. https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/

[50] S. J. LEDERMAN. Haptic perception: A tutorial. *Attention, Perception, & Psychophysics 2009, 71 (7), 1439-1459.* *https://link.springer.com/content/pdf/10.3758/APP.71.7.1439.pdf*

[51] Yanghi Chen. Vibrator Motor (2013). https://www.egr.msu.edu/classes/ece480/capstone/spring13/group05/downloads/Application%20Note-yangyi.pdf

[52] G. (2020a, diciembre 22). *Introduction to Buzzers: Piezo and Magnetic buzzers*. Latest Open Tech From Seeed. https://www.seeedstudio.com/blog/2020/12/22/introduction-to-buzzers-piezo-and-magnetic-buzzers/

[53] *What Is a Microcontroller? The Defining Characteristics and Architecture of a Common Component - Technical Articles*. (2019, 4 abril). All about Circuits. https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/

[54] MINISTERIO DE TRABAJO, MIGRACIONES Y SEGURIDAD SOCIAL. Boletín Oficial del Estado. https://www.boe.es/boe/dias/2019/03/01/pdfs/BOE-A-2019-2956.pdf

[55] Kiyoyuki Chinzei (2019). Safety of Surgical Robots and IEC 80601-2-77: The First International Standard for Surgical Robots. http://acta.uni-obuda.hu/Chinzei_95.pdf

[56] *Ultimate List of ISO Standards for Medical Devices*. (2020). ISO List. https://www.greenlight.guru/blog/iso-standards

[57] ISO 13485 MEDICAL DEVICES (2020). International Organization for Standarization. https://www.iso.org/iso-13485-medical-devices.html

[58] Tholey, G., Desai, J. P., & Castellanos, A. E. (2005). Force feedback plays a significant role in minimally invasive surgery: results and analysis. *Annals of surgery*, *241*(1), 102–109. https://doi.org/10.1097/01.sla.0000149301.60553.1e

[59] Lehman, A. C., Rentschler, M. E., Farritor, S. M., & Oleynikov, D. (2007). The current state of miniature in vivo laparoscopic robotics. *Journal of robotic surgery*, *1*(1), 45–49. https://doi.org/10.1007/s11701-007-0019-9

[60] Li F, Leighton JA, Sharma VK. Capsule endoscopy: a comprehensive review. Minerva Gastroenterol Dietol. 2007 Sep;53(3):257-72. PMID: 17912188.

[61] Smithers T. Autonomy in robots and other agents. Brain Cogn. 1997 Jun;34(1):88-106. doi: 10.1006/brcg.1997.0908. PMID: 9209757.

13. APPENDIXES

13.1. Code of the pen user interface

The following code is the one performed on the pen user interface, where mainly the values of roll, pitch and yaw are obtained, and the haptic feedback is executed on it.

```
#include "src/RoboticsUB.h"
#include <esp_now.h>
#include <WiFi.h>

// MAC address of the slave that contains the servos
uint8_t servosMacAddress[] = {0x7c, 0x9e, 0xbd, 0x61, 0xa1, 0x34};

// MAC address of the slave that communicates with the Computer
uint8_t computerMacAddress[] = {0x7c, 0x9e, 0xbd, 0x66, 0xe2, 0x58};

//variables for the ESP32 led
#define LED 2
int brillo = 0;
const int ledfreq = 5000;
const int ledchannel = 0;
const int ledresolution = 10;

// variables for the buzzer
const int buzzfreq = 2000; //variable for frequency
const int buzzchannel = 1; //variable that determines the channel that generates the signal
const int buzzresolution = 8;//PWM resolution
const int BUZZER = 18;// pin connected to buzzer
//variables for the vibrator
const int vibfreq = 2000; //variable for frequency
const int vibchannel = 2; //variable that determines the channel that generates the signal
const int vibresolution = 8;// PWM resolution
const int VIBRATOR = 12;

//Declaration variables for IMU pins and buttons
const int PIN_IMU_INT = 35;
const int PIN_S1 = 15;
const int PIN_S2 = 4;
const int PIN_S3 = 5;

IMU imu;

// button state variables
int s1Status = HIGH;
int s2Status = HIGH;
int s3Status = HIGH;

//time variables for the current
```

```
const unsigned long periodMillis_roll = 10;
unsigned long currentMillis_roll = 0;
unsigned long previousMillis_roll = 0;
const unsigned long periodMillis_pitch = 10;
unsigned long currentMillis_pitch = 0;
unsigned long previousMillis_pitch = 0;
const unsigned long periodMillis_yaw1 = 10;
unsigned long currentMillis_yaw1 = 0;
unsigned long previousMillis_yaw1 = 0;
const unsigned long periodMillis_yaw2 = 10;
unsigned long currentMillis_yaw2 = 0;
unsigned long previousMillis_yaw2 = 0;

//initially we are in the mode of working with the tip of the endowrist
int modo = 0;

// variable to store roll, pitch and yaw data
float *rpw;

// data structure that the master will send to the servos
typedef struct {
  float roll;
  float pitch;
  float yaw;
  int s1_pinzas;
  int s2_pinzas;
} TxMessage;

// structure type variable for the output data to the servos
TxMessage dataToServos;

// data structure that the master will send to the computer
typedef struct {
  float roll;
  float pitch;
  float yaw;
  float current_roll;
  float current_pitch;
  float current_yaw1;
  float current_yaw2;
  int s1_status;
  int s2_status;
  int modo;
} Tx2Message;

// structure type variable for output data to the computer
Tx2Message dataToComputer;

// data structure that the master will receive from the servos
typedef struct {
```

```
  float current_roll;
  float current_pitch;
  float current_yaw1;
  float current_yaw2;
} RxMessage;

// structure type variable for the input data of the streams
RxMessage dataCurrent;


/*PROGRAM SETUP*/
void setup() {
  Serial.begin(115200);
  //configure the pins of the buttons and the imu as outputs
  pinMode(PIN_IMU_INT, INPUT_PULLUP);
  pinMode(PIN_S1, INPUT);
  pinMode(PIN_S2, INPUT);
  pinMode(PIN_S3, INPUT);

  //we configure the buzzer and the vibrator to assign them a PWM
  ledcSetup(vibchannel, vibfreq, vibresolution);
  ledcAttachPin(VIBRATOR, vibchannel);

  ledcSetup(buzzchannel, buzzfreq, buzzresolution);
  ledcAttachPin(BUZZER, buzzchannel);

  imu.Install();

  // We enable WiFi in station mode
  WiFi.mode(WIFI_STA);

  // check that the master is initialized correctly
  if (esp_now_init() != ESP_OK) {
    //Serial.println("Error initializing ESP-NOW (master)");
    return;
  }

  // We register the slave of the servos
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, servosMacAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  //we check that the slave of the servos has been added correctly
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    //Serial.println("Failed to add servo slave");
    return;
  }

  //We configure the function to use each time a message is received
```

```
  esp_now_register_recv_cb(OnDataRecv);

  // We register the slave computer
  esp_now_peer_info_t peerInfo_computer = {};
  memcpy(peerInfo_computer.peer_addr, computerMacAddress, 6);
  peerInfo_computer.channel = 1;
  peerInfo_computer.encrypt = false;

// check that the computer slave has been added correctly
  if (esp_now_add_peer(&peerInfo_computer) != ESP_OK) {
    //Serial.println("Failed to add computer slave");
    return;
  }

//initialize the led of ESP32
  pinMode(LED,OUTPUT);
  ledcSetup(ledchannel, ledfreq, ledresolution);
  ledcAttachPin(LED, ledchannel);
}

/*LOOP CONTINUOUS PLAY*/
void loop() {
  /*
   * READING IMU AND BUTTONS DATA
   */
  if (digitalRead(PIN_IMU_INT) == HIGH)
  {
    imu.ReadSensor();   //for each interruption of the imu we read the data it
provides us
    rpw = imu.GetRPW();  // store this data in our variable
  }

  //we read the states of the buttons (which we have initially forced to HIGH)
  s1Status = digitalRead(PIN_S1);
  s2Status = digitalRead(PIN_S2);
  s3Status = digitalRead(PIN_S3);
  delay(50);
  if (s3Status == LOW) {
                                    //initially we start moving the gripper (modo = 0)
    modo = modo + 1;   //this variable controls if we want to move the gripper, the
z axis of the arm (1) or the entire arm (2)
  }else if (modo > 2){
    modo = 0;
  }

  /*
   * STORAGE OF DATA TO THE SLAVES
   */

  // data from the imu and streams to the computer
```

```
  dataToComputer.current_roll = dataCurrent.current_roll;
  dataToComputer.current_pitch = dataCurrent.current_pitch;
  dataToComputer.current_yaw1 = dataCurrent.current_yaw1;
  dataToComputer.current_yaw2 = dataCurrent.current_yaw2;


//use of the buzzer and the vibrator according to the current that comes from the slave
//Each one has its threshold
  if (dataCurrent.current_pitch > 70) { // current threshold by which the haptic interface is activated
    currentMillis_pitch = millis();
    if (currentMillis_pitch - previousMillis_pitch >= 10) {
      ledcWriteTone(buzzchannel, 2000);
      ledcWriteTone(vibchannel, 125);
      //according to the measured current the vibrator and the buzzer will be activated with more or less force
       // a multiplying factor is used to be able to activate them at the hardware level
ledcWrite(buzzchannel,dataCurrent.current_pitch*4);
      ledcWrite(vibchannel,dataCurrent.current_pitch*5);
      previousMillis_pitch = currentMillis_pitch;
    }
    if (dataCurrent.current_roll > 70) { // current threshold by which the haptic interface is activated
    currentMillis_pitch = millis();
    if (currentMillis_roll - previousMillis_roll >= 10) {
      ledcWriteTone(buzzchannel, 2000);
      ledcWriteTone(vibchannel, 125);
      //according to the measured current the vibrator and the buzzer will be activated with more or less force
       // a multiplying factor is used to be able to activate them at the hardware level
      ledcWrite(buzzchannel,dataCurrent.current_roll*4);
      ledcWrite(vibchannel,dataCurrent.current_roll*100);
      previousMillis_roll = currentMillis_roll;
    }
  else if (dataCurrent.current_pitch < 70 && dataCurrent.current_roll < 6 && dataCurrent.current_yaw1 < 15 && dataCurrent.current_yaw2 < 15){
    ledcWriteTone(vibchannel,0);
    ledcWriteTone(buzzchannel,0);
  }
  haptic_interface(currentMillis_yaw1,                    previousMillis_yaw1, dataCurrent.current_yaw1);
  haptic_interface(currentMillis_yaw2,                    previousMillis_yaw2, dataCurrent.current_yaw2);

  Serial.print("Roll: ");
  Serial.println(dataCurrent.current_roll);
  Serial.print("Pitch: ");
  Serial.println(dataCurrent.current_pitch);
  Serial.print("Yaw: ");
```

```
  Serial.println(dataCurrent.current_yaw1);
  Serial.print("Yaw2 current: ");
  Serial.println(dataCurrent.current_yaw2);

  // work mode decision algorithm
  if (modo == 0) {
    //if mode = 0 we move the gripper
     //we send the imu data (roll, pitch and yaw) and the real state of the buttons to open or close to the servos
     // we send the mode to the computer and the buttons don't matter to me because they won't do anything
    ledcWrite(ledchannel, 0);// esp32 led to visualize the change of mode

    dataToServos.roll = rpw[0];
    dataToServos.pitch = rpw[1];
    dataToServos.yaw = rpw[2];
    dataToServos.s1_pinzas = s1Status;
    dataToServos.s2_pinzas = s2Status;

    dataToComputer.roll = rpw[0];
    dataToComputer.pitch = rpw[1];
    dataToComputer.yaw = rpw[2];
    dataToComputer.s1_status = s1Status;
    dataToComputer.s2_status = s2Status;
    dataToComputer.modo = 0;

  } else if (modo == 1) {
    //if modo = 1 we move the arm in z axis
     //we send the data of the previous imu to the servos so that they remain still and the buttons as if they were not pressed
     // to the computer we send the mode and the real state of the buttons to raise or lower the position
    ledcWrite(ledchannel,102);// esp32 led to visualize the change of mode

    dataToServos.roll = dataToServos.roll;// we send the last stored data so that the servos do not move
    dataToServos.pitch = dataToServos.pitch;
    dataToServos.yaw = dataToServos.yaw;
    dataToServos.s1_pinzas = HIGH; //we send high to the gripper servos so they don't open or close
    dataToServos.s2_pinzas = HIGH;

    dataToComputer.roll = dataToComputer.roll;
    dataToComputer.pitch = dataToComputer.pitch;
    dataToComputer.yaw = dataToComputer.yaw;
    dataToComputer.s1_status = s1Status;
    dataToComputer.s2_status = s2Status;
    dataToComputer.modo = 1;

  } else if (modo == 2){
```

```
    //if mode = 2 we move the arm in all axes
     //we send the data of the previous imu to the servos so that they remain still and the buttons as if they were not pressed
     // we send the mode and the state of the buttons to the computer we don't care
    ledcWrite(ledchannel,1024);

    dataToServos.roll = dataToServos.roll;// we send the last stored data so that the servos do not move
    dataToServos.pitch = dataToServos.pitch;
    dataToServos.yaw = dataToServos.yaw;
    dataToServos.s1_pinzas = HIGH; // we send high to the gripper servos so that they do not open or close
    dataToServos.s2_pinzas = HIGH;

    dataToComputer.roll = rpw[0];
    dataToComputer.pitch = rpw[1];
    dataToComputer.yaw = rpw[2];
    dataToComputer.s1_status = HIGH;
    dataToComputer.s2_status = HIGH;
    dataToComputer.modo = 2;
  }

  /*
   * SENDING DATA TO THE SLAVES
   */
  // we will send the data of the imu and the buttons to the slave of the servos
  esp_err_t data_servos = esp_now_send(servosMacAddress, (uint8_t *) &dataToServos, sizeof(dataToServos));
  delay(10);

  // We also send the data to the computer
  esp_err_t data_computer = esp_now_send(computerMacAddress, (uint8_t *) &dataToComputer, sizeof(dataToComputer));
  delay(10);

  /*Serial.print("Roll: ");
  Serial.println(dataToServos.roll);
  Serial.print("Roll current: ");
  Serial.println(dataCurrent.current_roll);
  Serial.print("Pitch: ");
  Serial.println(dataToServos.pitch);
  Serial.print("Pitch current: ");
  Serial.println(dataCurrent.current_pitch);
  Serial.print("Yaw: ");
  Serial.println(dataToServos.yaw);
  Serial.print("Yaw1 current: ");
  Serial.println(dataCurrent.current_yaw1);
  Serial.print("Yaw2 current: ");
  Serial.println(dataCurrent.current_yaw2);
```

```
  Serial.print("MODO: ");
  Serial.println(modo);
  delay (10);*/

}


/* DECLARATION OF FUNCTIONS WE WILL USE*/
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  /*
     We receive the data from the slave of the servos and we copy them in the
master
  */
  memcpy(&dataCurrent, incomingData, sizeof(dataCurrent));
  //memcpy(destination, source, size)
}


 void haptic_interface(unsigned long currentMillis, unsigned long previousMillis,
float current){
  if (current > 15) { // current threshold by which the haptic interface is activated
    currentMillis = millis();
    if (currentMillis - previousMillis >= 10) {
      ledcWriteTone(buzzchannel, 2000);
      ledcWriteTone(vibchannel, 125);
//according to the measured current the vibrator and the buzzer will be activated
with more or less force
       // a multiplying factor is used to be able to activate them at the hardware level
      ledcWrite(buzzchannel,current*4);
      ledcWrite(vibchannel,current*20);
      previousMillis = currentMillis;
    }
  }else if (dataCurrent.current_pitch < 15 && dataCurrent.current_roll < 15 &&
dataCurrent.current_yaw1 < 15 && dataCurrent.current_yaw2 < 15){
    ledcWriteTone(vibchannel,0);
    ledcWriteTone(buzzchannel,0);
  }
}
```

13.2. Code of the slave servomotors

The following code is the one performed on the EndoWrist maneuverability, where mainly the servomotors are rotated while the force applied is calculated.

```
  #include <esp_now.h>
#include <WiFi.h>
#include "src/RoboticsUB.h"
#include <ESP32Servo.h>

// MAC address of the master
```

```
uint8_t masterMacAddress[] = {0x7c, 0x9e, 0xbd, 0x66, 0x7e, 0x60};

// identification variables for servos
Servo servo_roll;
Servo servo_pitch;
Servo servo_yaw1;
Servo servo_yaw2;

// variable to store the read value
float NewValueRoll = 0;         // variable to store the read value
float OldValueRoll = 0;
float roll=0;

float NewValuePitch=0;
float OldValuePitch=0;
float pitch=0;

float NewValueYaw=0;
float OldValueYaw=0;
float yaw=0;

//pin identification variables
/*ROLL*/
const int PIN_ANALOG_ROLL =36;
const int PIN_SIGNAL_ROLL =32;
/*PITCH*/
const int PIN_ANALOG_PITCH =39;
const int PIN_SIGNAL_PITCH =33;
/*YAW*/
const int PIN_ANALOG_YAW1 = 34;
const int PIN_SIGNAL_YAW1 = 25;
/*GRIPPER*/
const int PIN_ANALOG_YAW2 = 35;
const int PIN_SIGNAL_YAW2 = 27;

// constants for shunt resistors (they are all the same)
const float Rshunt = 1.6;

// variables to store the read data corresponding to the torque of the servos
float derivativeCurrent_roll=0;
float derivativeCurrent_pitch=0;
float derivativeCurrent_yaw1=0;
float derivativeCurrent_yaw2=0;

float previousfinal_roll=0;
float previousfinal_pitch=0;
float previousfinal_yaw1=0;
float previousfinal_yaw2=0;

float final_torque_roll=0;
```

```
float final_torque_pitch=0;
float final_torque_yaw1=0;
float final_torque_yaw2=0;


// button state variables
int s1_status;
int s2_status;

// data structure to be sent by this slave
typedef struct {
  float current_roll;
  float current_pitch;
  float current_yaw1;
  float current_yaw2;
} TxMessage;

// structure type variable to send data
TxMessage dataServos;

// data structure that we will receive from the master
typedef struct {
  float roll;
  float pitch;
  float yaw;
  int s1_pinzas;
  int s2_pinzas;
} RxMessage;

// structure type variable to receive data
RxMessage dataPencil;

/*PROGRAM SETUP*/
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  // we check that our slave is initialized well
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW (slave)");
    return;
  }

  // We register the master
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, masterMacAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // we check if the master could be added
```

```
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add master");
    return;
  }

  esp_now_register_recv_cb(OnDataRecv);

  // we configure the PWM that feed the servos
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);

  // we designate a frequency to the servos
  servo_roll.setPeriodHertz(50);
  servo_pitch.setPeriodHertz(50);
  servo_yaw1.setPeriodHertz(50);
  servo_yaw2.setPeriodHertz(50);

  // we assign the pins to each servo
  servo_roll.attach(PIN_SIGNAL_ROLL);
  servo_pitch.attach(PIN_SIGNAL_PITCH);
  servo_yaw1.attach(PIN_SIGNAL_YAW1);
  servo_yaw2.attach(PIN_SIGNAL_YAW2);

  // configure the analog pins of the servos as inputs
  pinMode(PIN_ANALOG_ROLL, INPUT);
  pinMode(PIN_ANALOG_PITCH, INPUT);
  pinMode(PIN_ANALOG_YAW1, INPUT);
  pinMode(PIN_ANALOG_YAW2, INPUT);

  // we start the servos in the central position (0º)
  servo_roll.write(90);
  servo_pitch.write(90);
  servo_yaw1.write(90);
  servo_yaw2.write(90);
  delay(2000);

}

/* LOOP CONTINUOUS PLAY*/
void loop() {
  // we start by sending the current data of the servos
  esp_err_t result = esp_now_send(masterMacAddress, (uint8_t *) &dataServos,
sizeof(dataServos));

  // We check that the data has been sent correctly
  if (result == ESP_OK) {
    //Serial.println("Sent with success");
  }
```

```
  else {
    Serial.println("Error sending the data");
  }
  /*
   * we define the limits so that it does not move outside its range of movement
    * In addition, a protection is added to avoid sudden movements
    */
  NewValueRoll=dataPencil.roll;
  if (abs(NewValueRoll-OldValueRoll)>10 && abs(NewValueRoll-
OldValueRoll)<350) {
    roll=OldValueRoll;
  }else{
    roll=NewValueRoll;
  }
  OldValueRoll=roll;
  if (roll > 180 && roll < 270) {
    servo_roll.write(180);
  } else if (roll > 270 && roll < 360) {
    servo_roll.write(0);
  }else{
    servo_roll.write(roll);
  }

  NewValuePitch=dataPencil.pitch;
  if (abs(NewValuePitch-OldValuePitch)>10 && abs(NewValuePitch-
OldValuePitch)<350) {
    pitch=OldValuePitch;
  }else{
    pitch=NewValuePitch;
  }
  OldValuePitch=pitch;
  if (pitch > 0 && pitch < 90) {
    servo_pitch.write(pitch + 90);
  } else if (pitch > 270 && pitch < 360) {
    servo_pitch.write(pitch - 270);
  }else{
    servo_pitch.write(pitch);
  }

  NewValueYaw=dataPencil.yaw;
  if (abs(NewValueYaw-OldValueYaw)>10 && abs(NewValueYaw-
OldValueYaw)<350) {
    yaw=OldValueYaw;
  }else{
    yaw=NewValueYaw;
  }
  OldValueYaw=yaw;
  if (yaw > 180 && yaw < 270) {
    servo_yaw1.write(180);
    servo_yaw2.write(0);
```

```
  } else if (yaw > 270 && yaw < 360) {
    servo_yaw1.write(0);
    servo_yaw2.write(180);
  }else{
    servo_yaw1.write(yaw);
    servo_yaw2.write(180 - yaw);
  }
  /*Serial.print("Roll:");
  Serial.println(roll);
  Serial.println(dataPencil.roll);
    Serial.print("pitch:");
  Serial.println(pitch);
  Serial.println(dataPencil.pitch);
    Serial.print("yaw:");
  Serial.println(yaw);
  Serial.println(dataPencil.yaw);*/

  // open and close clamps
  if (dataPencil.s1_pinzas == LOW || dataPencil.s2_pinzas == LOW) {
    if(yaw < 90){
      servo_yaw2.write(180 - yaw-30);
      delay(500);
      servo_yaw2.write(180 - yaw);
    }
  }
  if (dataPencil.s1_pinzas == LOW || dataPencil.s1_pinzas == LOW) {
    if(yaw > 90){
      servo_yaw1.write(yaw-30);
      delay(500);
      servo_yaw1.write(yaw);
   }
  }

/*
    * It is calculated by means of the shunt resistance and the derivative of the
current consumed by each servo
    * this way you can have the idea of the force that is being done at the moment
    */
  derivativeCurrent_roll = getDerivative(final_torque_roll, PIN_ANALOG_ROLL,
previousfinal_roll);
  derivativeCurrent_pitch = getDerivative(final_torque_pitch,
PIN_ANALOG_PITCH, previousfinal_pitch);
  derivativeCurrent_yaw1 = getDerivative(final_torque_yaw1,
PIN_ANALOG_YAW1, previousfinal_yaw1);
  derivativeCurrent_yaw2 = getDerivative(final_torque_yaw2,
PIN_ANALOG_YAW2, previousfinal_yaw2);

  Serial.println(dataPencil.roll);
  Serial.println(roll);
  Serial.println(pitch);
```

```
  Serial.println(yaw);
  // we send the current data to the master
  dataServos.current_roll = derivativeCurrent_roll;
  dataServos.current_pitch = derivativeCurrent_pitch;
  dataServos.current_yaw1 = derivativeCurrent_yaw1;
  dataServos.current_yaw2 = derivativeCurrent_yaw2;


  //Serial.println(torque_roll,DEC);
  //Serial.println(final_torque_roll,DEC);
  //Serial.println(derivativeCurrent_roll,DEC);

  delay(10);
}

/*
*DECLARATION OF RECURSIVE FUNCTIONS
  */
void OnDataRecv(const uint8_t *mac, const uint8_t *incomingData, int len) {
  // We get the data from the master and we copy it to the slave
  memcpy(&dataPencil, incomingData, sizeof(dataPencil));
  //memcpy(destination, source, size)
}


float getDerivative(float final_torque, int analog, float previous){
  float torque;
  float derivative;
  torque = getCurrent(20, analog);
  final_torque = final_torque + torque;
  derivative = abs(final_torque - previous);
  previous = final_torque;

  return derivative;
}

float getCurrent(uint32_t integrationTimeMs, int servo) {
 // we read the ADC where the shunt is connected to obtain the current
  //once we have the reading, we convert the ADC data to be able to apply V/R
  uint32_t startTime = millis();
  float integratedCurrent = 0;

  // We are adding the current measurement during the fixed time ->
integrationTimeMs (in milliseconds)
  while (millis() < startTime + integrationTimeMs) {
    uint16_t adcValue = analogRead(servo);
    integratedCurrent = integratedCurrent + ((float)adcValue / 4095.0 * 3.3) /
Rshunt;
  }
  return integratedCurrent;
```

```
}
```

13.3. Code of the computer

The following code is that the take all the data generated in the master and servomotors, used in the simulation.

```
#include <esp_now.h>
#include <WiFi.h>
#include "src/RoboticsUB.h"
#include <ESP32Servo.h>

enum class Command : byte
{
  GET_RPW = 1
};

Command command = Command::GET_RPW;

// MAC address of the master
uint8_t masterMacAddress[] = {0x7c, 0x9e, 0xbd, 0x66, 0x7e, 0x60};

// data structure received from the master
typedef struct {
  float roll;
  float pitch;
  float yaw;
  float current_roll;
  float current_pitch;
  float current_yaw1;
  float current_yaw2;
  int s1_status;
  int s2_status;
  int modo;
} RxMessage;

// structure type variable to store what comes from the master
RxMessage dataFromMaster;

/* PROGRAM SETUP*/
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW (computer slave)");
    return;
```

```
  }

  // We register the master
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, masterMacAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;

  // we check that the master has been registered correctly
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add master");
    return;
  }

  esp_now_register_recv_cb(OnDataRecv);

}

/* LOOP CONTINUOUS PLAY*/
void loop() {
  if (Serial.available() > 0)
  {
    command = (Command)Serial.read();
    switch (command)
    {
      case Command::GET_RPW:
        Serial.println(dataFromMaster.roll, DEC);
        Serial.println(dataFromMaster.pitch, DEC);
        Serial.println(dataFromMaster.yaw, DEC);
        Serial.println(dataFromMaster.current_roll, DEC);
        Serial.println(dataFromMaster.current_pitch, DEC);
        Serial.println(dataFromMaster.current_yaw1, DEC);
        Serial.println(dataFromMaster.current_yaw2, DEC);
        Serial.println(dataFromMaster.s1_status, DEC);
        Serial.println(dataFromMaster.s2_status, DEC);
        Serial.println(dataFromMaster.modo, DEC);
        break;
    }
  }

delay(10);
}

/*
 * DECLARATION OF RECURSIVE USE VARIABLES
 */
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  /*
     We get the data from the master and we copy it to the slave
  */
```

```
  memcpy(&dataFromMaster, incomingData, sizeof(dataFromMaster));
  //memcpy(destination, source, size)
}
```

13.4. Code for the assembly

The following code is to set up the EndoWrist with the servomotors, so that they start and end in the same position, and thus the placement and assembly of these each time they want to be used is easier.

```
#include <esp_now.h>
#include <WiFi.h>
#include "src/RoboticsUB.h"
#include <ESP32Servo.h>

// identification variables for servos
Servo servo_roll;
Servo servo_pitch;
Servo servo_yaw;
Servo servo_gripper;

// pin identification variables

/*ROLL*/
const int PIN_ANALOG_ROLL = 36;
const int PIN_SIGNAL_ROLL = 32;
/*PITCH*/
const int PIN_ANALOG_PITCH = 39;
const int PIN_SIGNAL_PITCH = 33;
/*YAW*/
const int PIN_ANALOG_YAW = 34;
const int PIN_SIGNAL_YAW = 25;
/*GRIPPER*/
const int PIN_ANALOG_GRIPPER = 35;
const int PIN_SIGNAL_GRIPPER = 27;
/*SHUNT*/
float Rshunt = 1.6;

/* DECLARATION OF THE FUNCTIONS TO BE USED*/
float getCurrent(uint32_t integrationTimeMs, int servo) {
  /*
    * we read the ADC where the shunt is connected to obtain the current
    * once we have the reading, we convert the ADC data to be able to apply V/R
    */  uint32_t startTime = millis();
  float integratedCurrent = 0;

  // We are adding the current measurement during the fixed time ->
integrationTimeMs (in milliseconds)
  while(millis()< startTime + integrationTimeMs) {
    uint16_t adcValue = analogRead(servo);
    integratedCurrent = integratedCurrent + ((float)adcValue/4095.0 * 3.3)/Rshunt;
  }
```

```
  return integratedCurrent;
}


/*PROGRAM SETUP*/
void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  // we check that our slave is initialized well
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW (slave)");
    return;
  }

  // we configure the PWM that feed the servos
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);

  // we designate a frequency to the servos
  servo_roll.setPeriodHertz(50);
  servo_pitch.setPeriodHertz(50);
  servo_yaw.setPeriodHertz(50);
  servo_gripper.setPeriodHertz(50);

  // we assign the pins to each servo
  servo_roll.attach(PIN_SIGNAL_ROLL);
  servo_pitch.attach(PIN_SIGNAL_PITCH);
  servo_yaw.attach(PIN_SIGNAL_YAW);
  servo_gripper.attach(PIN_SIGNAL_GRIPPER);

  // we start the servos to a fixed position
  servo_roll.write(90);
  delay(500);
  servo_pitch.write(90);
  delay(500);
  servo_yaw.write(90);
  delay(500);
  servo_gripper.write(90);
  delay(1000);

}

/*BUCLE DE REPRODUCCION CONTINUA*/
void loop(){

}
```

13.5. Code for RoboDK

The following code is in charge of taking all the data that comes from the computer and simulating the movements. Apart from sending and executing these ones in the UR5e in real life.

```
import serial
import time
import math
from enum import Enum
import csv
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np

# RoboDK API: import the robolink library (bridge with RoboDK)
from robolink import *
# Robot toolbox: import the robodk library (robotics toolbox)
from robodk import *

# ----------------------------------------------------------------------------
# Connection
# ----------------------------------------------------------------------------

# Establish the connection on a specific port
arduino = serial.Serial('COM5', 115200, timeout=1)

# Lets bring some time to the system to stablish the connetction
time.sleep(2)

# Establish a link with the simulator
RDK = Robolink()

# ----------------------------------------------------------------------------
# Simulator setup
# ----------------------------------------------------------------------------

# Retrieve all items (object in the robodk tree)
# Define the "robot" variable with our robot (UR5e)
robot = RDK.Item('UR5e')

# Define the "tcp" variable with the TCP of Endowrist needle
tcp_tool = RDK.Item('endowrist-Compound')
gripper = RDK.Item('Gripper')

# Performs a quick check to validate items defined
if robot.Valid():
    print('Robot selected: ' + robot.Name())
if tcp_tool.Valid():
    print('Tool selected: ' + tcp_tool.Name())
```

```
# Robot Flange with respect to UR5e base Frame
print('Robot POSE is: ' + repr(robot.Pose()))
# Tool frame with respect to Robot Flange
print('Robot POSE is: ' + repr(robot.PoseTool()))
# Tool frame with respect to Tool frame
print('TCP pose is: ' + repr(tcp_tool.Pose()))


robot.setSpeed(5)
# Connection of the program with the UR5e
"""
RUN_ON_ROBOT = True

if RDK.RunMode() != RUNMODE_SIMULATE:
    RUN_ON_ROBOT = False
    print("Mode inici: " + str(RDK.RunMode()))

if RUN_ON_ROBOT:
    robot.setConnectionParams('192.168.1.5',30000,'/',  'anonymous','')#  Update
connection parameters if required:

    success = robot.Connect('192.168.1.5') # Try to connect once
    status, status_msg = robot.ConnectedState()
    if status != ROBOTCOM_READY: # Stop if the connection did not succeed
        print(status_msg)
        raise Exception("Failed to connect: " + status_msg)

    # This will set to run the API programs on the robot and the simulator (online
programming)
    RDK.setRunMode(RUNMODE_RUN_ROBOT)

else:

     RDK.setRunMode(RUNMODE_SIMULATE)  # This will run the API program
on the simulator (offline programming)

#print("Mode: " + str(RDK.RunMode()))

joints_ref = robot.Joints() #Get the current joint position of the robot (updates the
position on the robot simulator)
#print("Joints: " + str(joints_ref))
 """
# ----------------------------------------------------------------------------
# Reference frame is fixed to TCP
#
# Data comunication
# ----------------------------------------------------------------------------
#frame = RDK.ItemUserPick('Frame 2', ITEM_TYPE_FRAME)

class Command(Enum):
```

```
    GET_RPW = b'\x01'

data_torque_roll = []
data_torque_pitch = []
data_torque_yaw1 = []
data_torque_yaw2 = []
point = 0
fig, ax = plt.subplots()
plt.ion()
maxlen = 400
x_lista = []
y_roll=[]
y_pitch=[]
y_yaw1 = []
y_yaw2 = []
try:

    # Discard initial ESP32 message
    arduino.reset_input_buffer()

    while True:

        # Requesting data to Ardino
        arduino.write(Command.GET_RPW.value)
        # Storing received data
        roll_str = arduino.readline().strip()
        pitch_str = arduino.readline().strip()
        yaw_str = arduino.readline().strip()
        torque0_str = arduino.readline().strip()
        torque1_str = arduino.readline().strip()
        torque2_str = arduino.readline().strip()
        torque3_str = arduino.readline().strip()
        s1_str = arduino.readline().strip()
        s2_str = arduino.readline().strip()
        modo_str = arduino.readline().strip()
        # Convert variable values from string to float/itn
        roll = float(roll_str)
        pitch = float(pitch_str)
        yaw = float(yaw_str)
        torque_roll = float(torque0_str)
        torque_pitch = float(torque1_str)
        torque_yaw1 = float(torque2_str)
        torque_yaw2 = float(torque3_str)
        s1_status = bool(int(s1_str))
        s2_status = bool(int(s2_str))
        modo = int(modo_str)

        # printing data on screen
        print(roll, pitch, yaw, torque_roll, torque_pitch, torque_yaw1, torque_yaw2,
s1_status, s2_status, modo)
```

```
# Take the values of the tool position
tcp_tool_pose = tcp_tool.Pose()
x, y, z, r, p, w = Pose_2_TxyzRxyz(tcp_tool_pose)

# Take the values of the gripper position
gripper_pose = gripper.Pose()
xg, yg, zg, rg, pg, wg = Pose_2_TxyzRxyz(gripper_pose)

# Take the values of the robot position
robot_pose=robot.Pose()
xr, yr, zr, rr, pr, wr = Pose_2_TxyzRxyz(robot_pose)

# Convert from degrees to radians R,P,Y angles
R = math.radians(roll)
P = math.radians(pitch)
W = math.radians(yaw)
X = int(xg)
Y = int(yg)
Z = int(zg)
Xr=int(xr)
Yr=int(yr)
Zr=int(zr)

if modo == 0: #moving the gripper
    gripper_pose =  transl(X,Y,Z) * rotx(pi) * rotz(W) * roty(P) * rotx(R) * roty(0)
    #tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix)
    gripper.setPose(gripper_pose)

elif modo == 1: #moving through Z axis the robot
    #pose_matrix = transl(X,Y,Z) * rotz(-W) * roty(P) * rotx(-R)
    #tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix)

    position=robot.Pose()
    xyzabc = Pose_2_KUKA(position)
    if not s1_status: #s1 button pressed, going up
        if ((float(xyzabc[2]) >50) & (float(xyzabc[2]) <500)): #upper limit
            approach = robot.Pose() * transl(0,0,1)
            robot.MoveL(approach, False)
        if (float(xyzabc[2]) <50): #lower limit
            (xyzabc[2])=51
            target=KUKA_2_Pose(xyzabc)
            robot.MoveL(target, False)
    elif not s2_status: #s2 button pressed, going down
        if ((float(xyzabc[2]) >50) & (float(xyzabc[2]) <500)): #upper limit
            approach = robot.Pose() * transl(0,0,-1)
            robot.MoveL(approach, False)
        if ((float(xyzabc[2]) >500)): #lower limit
            (xyzabc[2])=499
            target=KUKA_2_Pose(xyzabc)
```

```
                robot.MoveL(target, False)

        elif modo == 2: #positioning the robot
            tcp_pose = transl(Xr,Yr,Zr) * rotz(W) * roty(P) * rotx (R)
            if robot.MoveL_Test(robot.Joints(),tcp_pose) == 0:
                robot.MoveL(tcp_pose)
            else:
                print('The robot cannot reach target because it is out of reach or causes
collision.')

        # exporting torques to an excel sheet
        data_r = [torque_roll]
        data_torque_roll = data_torque_roll + data_r
        data_p = [torque_pitch]
        data_torque_pitch = data_torque_pitch + data_p
        data_y1 = [torque_yaw1]
        data_torque_yaw1 = data_torque_yaw1 + data_y1
        data_y2 = [torque_yaw2]
        data_torque_yaw2 = data_torque_yaw2 + data_y2

        #Generating a real world plot of the torque values
        if torque_roll:
            data_roll = int(torque_roll)
            x_lista.append(point)
            y_roll.append(data_roll)
            data_pitch=int(torque_pitch)
            y_pitch.append(data_pitch)
            data_yaw1=int(torque_yaw1)
            y_yaw1.append(data_yaw1)
            data_yaw2=int(torque_yaw2)
            y_yaw2.append(data_yaw2)
            if len(x_lista) > maxlen:
                x_lista = x_lista[1:]
                y_roll = y_roll[1:]
                y_pitch = y_pitch[1:]
                y_yaw = y_yaw[1:]
            plt.plot(x_lista, y_roll, color='r')
            plt.plot(x_lista, y_pitch, color='b')
            plt.plot(x_lista, y_yaw1, color='g')
            plt.plot(x_lista, y_yaw2, color='y')
            point += 1
            plt.pause(0.05)

            ax.clear()
            fig.legend(["Roll",'Pitch','Yaw1','Yaw2'],loc = "upper left")
            plt.show()

        #Generate the excel with all the torque values generated
        with open('torque.csv', 'w') as csvfile:
            header = ['torque roll','torque pitch', 'torque yaw1', 'torque yaw2']
```

```
            writer = csv.DictWriter(csvfile,fieldnames=header)
            writer.writeheader()
            for i in range(0,len(data_torque_pitch)):
                writer.writerow({'torque roll': data_torque_roll[i],
                                 'torque pitch': data_torque_pitch[i],
                                 'torque yaw1': data_torque_yaw1[i],
                                 'torque yaw2': data_torque_yaw2[i]})

    except KeyboardInterrupt:
        print("Communication stopped.")
        pass

    # ------------------------------------------------------------------------
    # Disconnect Arduino
    # ------------------------------------------------------------------------
    print("Disconnecting Arduino...")
    arduino.close()
```