



UNIVERSITAT_{DE} BARCELONA

Final Degree Project

Biomedical Engineering Degree

***Da Vinci* robot at Hospital Clinic.
Manoeuvrability devices and
performance in robotic tech.**

Barcelona, 14th of June of 2021

Author: Júlia Meca Santamaria

Tutor: Manel Puig Vidal

Acknowledgments

This research was completed with unwavering support from Universitat de Barcelona and, specifically, the resources provided by the Electronics and Biomedical Engineering Department.

I would like to thank Professor Manel Puig Vidal for allowing me to take on an ambitious project such as the *Da Vinci System* and for being an inspirational mentor. I would like to thank him for his hands-on guidance through every step of the project. The knowledge I have gained in our work together, in both theoretical and practical matters, is immeasurably beneficial.

I would like to thank Meiling Chen for his many design contributions to the project.

I would like to thank Maria Mor for our mutual support during these year of practicum in the Department and for her contributions in the torque insight of this project, which have been of great importance in the development of haptic feedback improvements.

I would like to thank my family and friends for their support in handling the adversity and stress associated with completing this research.

Thank you.

Abstract

Robot-assisted surgical systems are becoming increasingly common in medical procedures as they embrace many of the benefits of minimally invasive surgery including less trauma, recovery time and financial costs associated to the treatment after surgery. These robotic systems allow the surgeons to navigate within confined spaces where an operator's human hand would normally be greatly limited. This dexterity is further strengthened through motion scaling, which translates large motions by the operator into diminutive actions of the robotic end effector. An example of this is the *Da Vinci System* which is coupled to the *EndoWrist* end effector tool.

Nevertheless, these systems also have some drawbacks such as the high cost of the surgery itself and the lack of tactile or haptic feedback. This means that as the surgeon is performing the procedures outside the patient's body, he/she can not feel the resistance of the human tissue's when cutting. Therefore, one can risk damaging healthy tissues if force is not controlled or, when sewing, one can exert an exaggerated force and break the thread.

In this project, a new system is created based on the UR5 robot (*Universal Robots*) and an *EndoWrist* needle to mimic the behaviour of the *Da Vinci System* and implement some improvements regarding the manoeuvrability and haptic feedback performance.

Index

List of figures	6
List of tables	7
1. Introduction.....	8
1.1. Objectives	8
1.2. Scope and span	9
1.3. Methodology	10
2. Background.....	11
2.1. State of the art of robotic arm assisted surgery	11
2.2. Da Vinci Surgical System research	12
2.3. Maneuverability performance	15
2.4. State of the situation and future development	15
3. Market analysis	17
3.1. Historical evolution of the market	17
3.2. Current situation of the market	17
3.3. Future perspective of the market	19
4. Regulatory and legal issues	20
4.1. Hardware	20
4.2. Software	20
5. Concept of engineering.....	21
5.1. Hardware solutions	21
5.1.1. Robotic arms	21
5.1.2. Servomotors	24
5.1.3. IMU sensors	24
5.1.4. Evaluation boards	25
5.1.5. Haptic feedback user interface	26
5.1.6. Vibration motors and buzzers	26
5.1.7. Push buttons	27
5.2. Software solutions	28
5.2.1. Programming Software	28
5.2.2. Designing Software	28
5.3. Proposed solution	29
6. Detailed Engineering.....	31
6.1. Hardware	31
6.2. Software	35
7. Experimental validation	39

7.1.	Assembly of the servomotors and the EndoWrist tool.....	39
7.2.	Final setup with the UR5 robot.....	39
8.	Technical feasibility.....	41
8.1.	SWOT analysis	41
9.	Execution schedule	43
9.1.	Work Breakdown Structure (WBS).....	43
9.1.1.	Dictionaries and duration of activities	43
9.2.	GANTT chart.....	45
10.	Economic viability.....	47
11.	Discussion and conclusions	49
11.1.	Achieved objectives and results.....	49
11.2.	Future opportunities	50
12.	Bibliography	51
13.	Appendixes.....	54
13.1.	Appendix 1: Transmission of the RPY angles from the pen to the servomotors (Arduino IDE).....	54
13.2.	Appendix 2: Transmission of the RPY angles from the pen to the servomotors (RoboDK).....	56
13.3.	Appendix 3: Acquisition and visualization of the RPY angles from the haptic pen (Arduino IDE)	59
13.4.	Appendix 4: Acquisition and visualization of the RPY angles from the haptic pen (RoboDK)	61
13.5.	Appendix 5: Transmission of the RPY angles from the PC to the servomotors (Arduino IDE).....	64
13.6.	Appendix 6: Transmission of the RPY angles from the PC to the servomotors (RoboDK).....	67
13.7.	Appendix 7: Transmission of the RPY angles from the PC to the servomotors -sliders (RoboDK)	69
13.8.	Appendix 8: Advancement of the TCP after pressing the pushbutton (Arduino IDE)	71
13.9.	Appendix 9: Advancement of the TCP after pressing the pushbutton (RoboDK)	74
13.10.	Appendix 10: Opening of the needle jaws after pressing the pushbutton (Arduino IDE).....	78

List of figures

Figure 1. Zeus System components. Zeus Robot Arms in the left and the console in the right.....	11
Figure 2. Da Vinci S set up components. In order: Surgeon's Console, the Surgical Cart and the Vision System.	13
Figure 3. The Da Vinci Single-Site platform [10].....	14
Figure 4. NAVIO Surgical System handheld robot representation. Real time anatomic characterization in the right interfaces.....	18
Figure 5. UR5 robot from Universal.....	21
Figure 6. UR6 robots in a hospital lab carrying out automatized tasks.....	22
Figure 7. Modus V, from Synaptive Medical, coupled to a robot arm from Universal Robots.....	22
Figure 8. Mover4 arm robot.....	22
Figure 9. Sawyer robot from Rethink Robotics.....	23
Figure 10. Servomotor prototypes under study A) Dynamixel AX-12A B) Longrunner ky66.....	24
Figure 11. Phantom Omni haptic device.....	26
Figure 12. Schematic diagram of the main modules composing the system.....	30
Figure 13. EndoWrist end effector needle.....	31
Figure 14. Working setup.....	31
Figure 15. Illustration of how roll, pitch and yaw are measured in the human hand.....	31
Figure 16. (Left) References of the disks. (Right) RPY movements of the tool. Photographs taken by Arturo Yscadar.....	32
Figure 17. Motion of the needle driver. A. Roll. B. Pitch. C. Yaw.....	32
Figure 18. Example of the needle driver opening.....	33
Figure 19. Overview of the entire circuit. Picture taken by the author.....	33
Figure 20. Electrical circuit of A. Servomotor. B. Pushbutton.....	34
Figure 21. Electrical design of the PCB with Kicad.....	34
Figure 22. Design of the haptic pen with the ESP32 and the PCB inside.....	34
Figure 23. Part of the Arduino code showing the transmission of the motion to the servomotors.....	35
Figure 24. Diagram of the RPY angles on the haptic pen and the IMU.....	36
Figure 25. Workflow of software A. Acquisition and visualization of the RPY angles of the pen haptic.....	36
Figure 26. RoboDK program showing the Tkinter interface of RPY acquisition and representation.....	36
Figure 27. Introduction of the RPY angles in the RoboDK interface A. from the terminal B. Through a slider (Tkinter)	37
Figure 28. RoboDK representation after a 45-degree rotation of each servomotor.....	37
Figure 29. RoboDK representation of the advancement in X-axis.....	38
Figure 30. Arduino code describing the opening of the needle when pushbutton is pressed.....	38
Figure 31. Mechanical solution to gear the servomotors with the EndoWrist disks.....	39
Figure 32. Plastic piece supporting the EndoWrist, servomotors and a camera.....	39
Figure 33. Pictures of the final setup with the UR5 robot.....	40
Figure 34. Work Breakdown Structure of the project.....	43
Figure 35. GANTT diagram.....	46

List of tables

Table 1. Prices list of 2018. This budget includes the robot arm mover and a complete set of accessories [28].	23
Table 2. Description and comparison of each servomotor features.....	24
Table 3. Description and comparison of the Inertial Measurement Units. The axis of motion depends on the components of the IMU, for example, the 9-axis IMU has a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer.	25
Table 4. Description and comparison of the evaluation boards.	25
Table 5. Buzzer comparison.	26
Table 6. Vibration motor comparison.....	27
Table 7. Push buttons comparison. The SparkFun buttons data was found in the commercial announcements, although no datasheets were found.	27
Table 8. Comparison between software environments.....	28
Table 9. Comparison between designing software's	28
Table 10. Proposed solution of hardware and software.....	30
Table 11. Descriptions of the tasks of the Final Degree Project.	45
Table 12. Project tasks with the corresponding duration and timings	46
Table 13. Estimation of the cost of the project.....	48

1. Introduction

1.1. Objectives

Robot-assisted surgery is a minimally invasive surgical technique that supposed a revolution in the medicine's field. However, its great complexity requires constant development of methods which improve and facilitate the procedures carried out during interventions. In the last years, the improvement of maneuverability and haptic feedback has become quite significant in the advancement of this instrumentation.

The world-leader surgical robot system is the *Da Vinci System* (Surgical Intuitive, Inc.). This modern groundbreaking technology enables a robotic arm to precisely translate the movements performed by a surgeon through three main elements: a Surgeon's Console, a Surgical Cart and a Vision System [1]. This project is focused on translating the movements from the Surgeon's Console to the end-effectors with maximum precision and with no delay.

The first objective of this project is to analyze the maneuverability performance of minimally invasive surgical robots, specifically the one implemented in the *Da Vinci* system: the *EndoWrist* end effector tool. This performance involves the speed at which the movement is transmitted from the motors to the end effector (the tweezers), the precision with which the robotic arm tries to mimic the surgeon's maneuvers, the stability of the movement and the tremor filtering. This study is made to comprehend the functioning of the *Da Vinci System* in order to be able to reproduce the maneuverability performance in our lab, by means of the UR5 (*Universal Robots*) and a set of end-effectors which will be designed by a Mechanical Engineer from Escola Universitária Salesiana de Sarrià (EUSS). Furthermore, this project is also carried out along with another Biomedical Engineering student in charge of analyzing the haptic feedback of the *Da Vinci System*.

To accomplish the main goal, one has to get acquainted with the functioning of the surgical robotic systems regarding the hardware elements that make possible the reproduction of the movement as well as the software interface that reads and transmits the information. With this purpose, several hours have been spent in the laboratory of the Faculty of Physics during the development of this Final Degree Project. However, due to the COVID-19 pandemic and the specific and controlled usage of the *Da Vinci* robot, students were not allowed to enter the clinical facilities. Therefore, one could not see the system in-person and properly analyze its properties and performance in-vivo.

An in-deep study of the different software frameworks or robotics middleware such as the *RoboDK*, *Arduino* or *Robotic Operating System (ROS)* will be examined. Furthermore, different robotic arms will be examined, specially UR5 (*Universal Robots*) which is the one available in the facilities. At the end, one will assess which is the most suitable configuration (of both software and hardware components) that allows a better performance in terms of maneuverability.

The final purpose of this project is to design a haptic pen tool (which will be fabricated by 3D printing) capable of mimicking the surgeon movements. This pen will be designed with some extra capabilities to improve both the maneuverability and the haptic feedback. In one hand, a pair of push buttons will be installed to manipulate the needle and the end effector tool's advancement. On the other hand, a buzzer and a vibration motor will be coupled

to the pen to provide the surgeon with a sensorial perception that an excessive force is being exerted. Taking all this into account, one will evaluate the capabilities of UR5 robot attached to the *EndoWrist* by a designed 3D printed piece and compare it to the motion capabilities accomplished by current surgical robots such as *Da Vinci Robotic System*. Finally, an economic study on the cost of the instrumentation needed will have to be fulfilled.

To sum up, the following list exemplifies the different aims this project aspires to accomplish:

- Familiarizing with the minimally invasive surgery robots.
- Learning about the performance and interface necessary for this purpose: *Robotic Operating System (ROS)*, *Arduino IDE* and *RoboDK*.
- Comparative study of the capabilities among different surgical robots currently used in the market.
- An exhaustive analysis of the hardware components and software environments that can be implemented in these technologies.
- Development of the programming code with *Arduino* and *RoboDK* that transmits the RPY angles from the computer to the servomotors (end effector tool) and vice versa.
- Development of the programming code with *Arduino* and *RoboDK* that controls the push buttons: advancement of the *EndoWrist* in the direction of motion and tweezer's opening and closing.
- Mechanically assembly of the arm and the end-effector with the 3D printed pieces from EUSS.
- Final validation and evaluation of its performance in the laboratory. If conditions are met, recording of the interlocking of the servomotors with the *EndoWrist* gears and overall performance of the system.

1.2. Scope and span

The extent of this project includes the accomplishment of all the previously described objectives. Nevertheless, as a Final Degree Project there are considerable restrictions in time, space, cost and knowledge of the author that should be considered.

With regards to the time limitation, as this project is less than a year length, it will mostly be based on the research study of the different solutions and the premature designs of the prototype. If favorable conditions of time and economic viability were met, its further development with finest materials would be executed in the future.

As the *Da Vinci* device is enormously high priced, one cannot risk damaging it with an unfortunate software or hardware modifications. Therefore, all the tests are made with *EndoWrist* tools in abeyance which are available in the Physics laboratory. Besides, one has not enough technical knowledge to manipulate the actual machine so the checking of the most suitable solutions and programs will have to be performed through simulations.

Although the uncertainty of the COVID-19 pandemic prevented the author from visiting the hospital, which supposed an important drawback or, at least, implied a slowdown in the progress of the project; one could benefit of all the facilities and resources provided by the Electronics and Biomedical Engineering Department of University of Barcelona's Physics' Faculty.

1.3. Methodology

With the objectives, scope and span clarified, the background will be introduced with information regarding to the state of the art of robot-assisted surgery specially the *Da Vinci* system performance. Next, we will analyze the market of surgical robots in order to familiarize with this field and with the technical features that characterize them. Then, the regulatory and legal issues will be approached.

Considering all the background information and the challenges that the prototype should overcome, one will study different robotic arms that could fulfil the final goals of this project. Once a solution is chosen, one will proceed to examine its strengths, weaknesses, opportunities and threats (SWOT analysis). Finally, the execution schedule will be described in order to have a proper planning on the project that optimized the time and effort of the people involved. The general budget required to carry out the project will be detailed at the end.

As can be seen in the Work Breakdown Structure (WBS), the methodology of the project is divided in four parts: an educational stage, a hardware and software stage, an experimental stage, and the final discussion stage. Each block has slightly different methodologies from the theoretical research of information regarding the robots, the learning of programming techniques through programming tutorials, the assembly of the mechanical components or the writing of the memory in the final part.

The practical part of the project will be carried out in the Electronic and Biomedical Engineering Department of the faculty of physics, where different approaches of the maneuverability and haptic technology will be analyzed, the UR5 robot performance will be tested, the 3D printed designs will be unified with the electronical circuits and after all, we will better understand its strengths and limitations. All materials needed for the project, belong to the laboratory of the faculty.

2. Background

2.1. State of the art of robotic arm assisted surgery

A procedure is considered to be surgical when it involves cutting a patient's tissues or closing a previously sustained wound. With the passage of time, many revolutions have occurred in this field such as the introduction of anesthesia in the 19th century, the first successful organ transplantations during the 20th century or the arrival of robotic surgery at the end of the 20th century as well. Although nowadays robotics is widely and routinely used, its entrance in the field of medicine has been slow and progressive.

Robotic surgery, or robot-assisted surgery, is a still emerging technology that allows minimally invasive procedures. This generates a great interest among health professionals because it means, fundamentally, shorter hospitalization and faster recovery of the patients [2]. On top of that, *there is* a reduced risk of infection, less blood loss and less scarring.

Authors often differ in the definition of the first robotic prototype as the way we know it. Nevertheless, for most of them, it is considered to be the *PUMA* (Programmable Universal Machine for Assembly) 560 robotic system, which was employed at 1985 in a neurosurgical biopsy. This served as a starting point for many companies and universities to develop robotic systems such as *PROBOT*, specialized in transurethral resection of the prostate; *ROBODOC* for hip replacement surgeries, the robotic arm *AESOP* (Automated Endoscopic System for Optimal Positioning) controlled by voice commands to manipulate the endoscopic camera and so on. Later modifications led to the development of two recognized and rival systems: *Da Vinci* and the *Zeus System*, which are similar in their capabilities but different in their approach to robotic surgery [3].

The *Zeus* (Computer Motion, Inc., Goleta, Ca) is a three-armed platform that makes use of the *AESOP* camera: one arm holds the voice-controlled camera and the other two (controlled by the surgeon) are used to hold the surgical instruments. It has two separate hubs: the patient side where the procedure is done and the surgeon side controlling the first. It received the FDA approval for limited use in 2001 [4]. In the *Zeus System* (see Figure 1), both the monitor and handles are ergonomically positioned to maximize dexterity and allow complete visualization. The system allows the articulation of the end-effector through 7 degrees of freedom (DOF). [5]



Figure 1. Zeus System components. Zeus Robot Arms in the left and the console in the right.

There are three main types of robotic systems currently in use in the surgical field: active, semi-active and master-

slave systems:

- **Active systems:** the robot essentially works autonomously, or undertakes pre-programmed tasks, under the supervision of the surgeon. These systems are able to recognize the changes in the environment and organize its duties accordingly. An example of an active system is the *ROBODOC*.
- **Semi-active systems:** the robot's total autonomy is combined with a surgeon-driven element. *Neuromate* is an image-guided robotic system used in stereotactic surgery [6]
- **Master-slave systems** lack of any pre-programmed or autonomous element. They allow the surgeon to directly telemanipulate the robot from a remotely placed command center. In this situation, the surgeon's hand movements are transmitted to the surgical end-effector instruments. *Zeus* and *Da Vinci* were the forerunners in the master-slave category.

Master-slave systems are also known as passive robots since it is the doctor who provides the motion inputs. The control of the system is achieved by using these inputs in its control algorithm during surgery. One of its most outstanding advantages is that it scales the motion received from the master system to increase the sensitivity of the slave system. In addition, it can have six or more DOF so the surgeon can enter inputs not only from his/her hands but also from fingers and elbow. Hence, flexibility increases. One of the most difficult challenge is to keep the hands steady during the entire surgical procedure. This problem can be improved by filtering the tremor with the master system and, although the surgeon's hand can shake, the robot remains steady inside the patient's cavities.

More about the advantages, as well as limitations, of these surgical robots will be addressed in the state of the situation section.

2.2. *Da Vinci Surgical System research*

The *Da Vinci* system is a sophisticated robotic platform designed to expand the surgeon's capabilities in minimally invasive option for major surgery. The first prototype was introduced in 2000 by *Intuitive Surgical* and it was approved by the FDA at the same year. During these two decades, medical institutions have been evaluating the clinical and economic benefits of the robot – there are now more than 21.000 reviewed published articles that support the safety, efficacy, and benefits of *Da Vinci* surgical systems. In fact, the single port *Da Vinci* platform is now in use in more than 40 centers [7].

The complexity of this kind of technology, both electrically and mechanically, requires an environment far from the traditional operating room (OR). As a matter of fact, all the personnel present in the OR (nurses, technicians, or surgeons) must be trained to manage the equipment. This way, problems that emerge during the surgery can be easily identified and solved by any member of the team.

From its introduction in the market, there has been many generations of this surgical system: the Standard *Da Vinci* (which was introduced at 2000 although its commercialization stopped seven years later) only had three robotic arms, whereas S model (introduced at 2006), Si model (2011), Xi model (presented in April of 2014), X model (approved on April of 2017) and Single Port (2018) all have four arms.

Da Vinci Systems allow the introduction of miniaturized wristed instruments and a high-definition 3D camera. The system cannot be programmed nor can make decisions on its own, it requires that every surgical maneuver is

performed with direct input from the surgeon. Even though every generation adds further improvements, all four-armed systems basically consist on [1]:

- *Surgeon's Console*: from this element the surgeon can manipulate the arms of the robot. The individual grabs two handles which position and orientation trigger highly sensitive motor sensors that transfer the information to the end-effector tool. In addition, some models incorporate foot pedals to control electrocautery, camera focus and instrument/camera arm clutches.
- *Surgical Cart* provides 3 degrees of freedom (pitch, yaw, insertion). Attached to the robot arm is the surgical instrument, the tip of which is a mechanical cable-driven wrist (*EndoWrist*) which adds 4 more degrees of freedom (internal pitch, internal yaw, rotation, and grip).

Intuitive Surgical patented *EndoWrist* instruments which are designed to provide natural dexterity through several accessories such as scissors, graspers, needle holders, monopolar cautery instruments, clip appliers, scalpels, etc. All these tools provide the total 7 DOF, 90° of articulation, intuitive motion, fingertip control, motion scaling and tremor reduction. The wrist-like movement, responsiveness and robotic control afforded by the *Da Vinci* and its exclusive *EndoWrist* instruments provide surgeons fluid ambidexterity and unparalleled precision.

The patient cart rolls on wheels and is moved and positioned over the patient. The robotic arms are designed like the human arm with a shoulder, an elbow and a wrist. The patient cart is connected through wires to the surgeon's console but before positioning the cart, It must be covered by an additional sterile coat to prevent coming into contact with non-sterile objects.

- A *Vision System* controls the whole network that resides in the surgeon's console. It is an image processing computer that generates a 3-dimensional image with depth of field. The 3D camera is attached to the 4th robotic arm, which magnifies the surgical site. The vision cart consists on a left eye camera control unit, a right eye camera control unit, a light source, video synchronizer and focus controller, assistant monitors...

We must consider that in the operating room there is the surgeon working from the computer console and the surgical team supervising the robot at the patient's bedside. In the following image (Figure 2), we can observe all the elements of the *Da Vinci* from left to the right in the order one has just explained:



Figure 2. *Da Vinci S* set up components. In order: Surgeon's Console, the Surgical Cart and the Vision System.

In this Final Degree Project, one will focus on the fourth generation of the *Da Vinci Systems* since it is the one Hospital Clinic adopted. The *Da Vinci X* surgical robot was designed to be more affordable while still providing most of the abilities of the principal model. It is something in between the earlier *Si* model and the *Da Vinci Xi*. The *Da Vinci Si* is able to roll in on a side cart, letting surgeons perform procedures more precisely with its array of mechanical arms. On the other hand, the *Da Vinci Xi* not only upgraded those arms with better movement, reach and dexterity, but moved them from a side cart to an overhead arrangement.

From that vantage point, *Intuitive Surgical* arguments the robot has better access to more parts of the body. The key of the price reduction lies on the fact that *Da Vinci X* takes the improved arms and instruments of the *Xi* model, into a cart like the *Si*. This modification which eases the finance of the robot, sacrifices the ability to perform procedures in several parts of the body at the same time. However, it differs from the *Si* in the voice and laser guidance systems or the lightweight endoscope.

The *Da Vinci SP* (Single Port) is described by *Intuitive* as a single arm that delivers three multi-jointed instruments and a fully wristed 3DHD camera for visibility and control in narrow surgical spaces. This novel set includes four cannulas (two curved and two straight) and an insufflation valve, which inflates the abdominal cavity with CO₂. The curved cannulas allow the controlled instruments to be positioned to achieve triangulation of the target anatomy (which is accomplished by crossing curved cannulas midway through the access port). Alternatively, one of the straight cannulas accommodates the endoscope whereas the other one serves as a bedside-assistant port. The second part of the platform is a set of semirigid, non-wristed instruments with standard *Da Vinci* instrument tips [8]. In the next illustration (Figure 3), the configuration of the multichannel access port can be exhaustively examined:

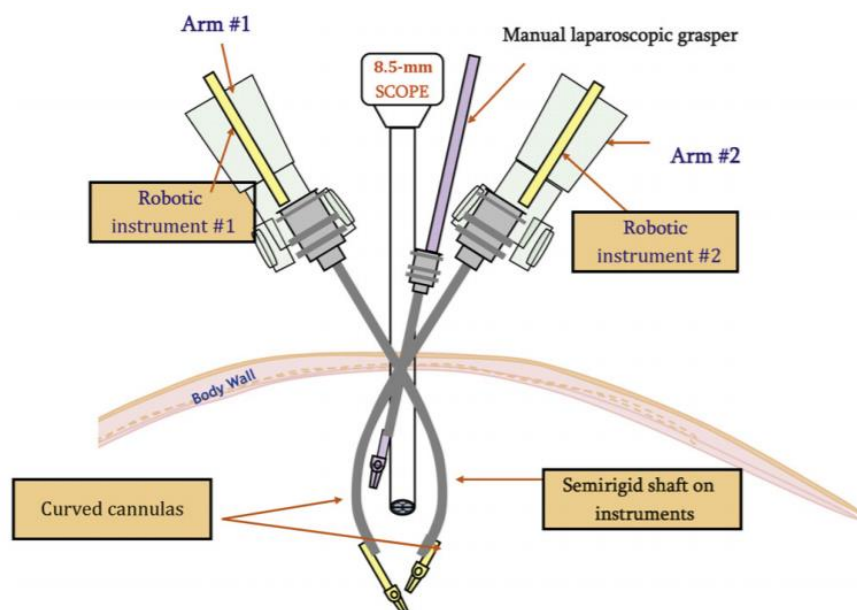


Figure 3. The *Da Vinci Single-Site* platform [10].

Currently urological procedures like prostatectomies, cystectomies and nephrectomies are performed with the *Da Vinci*. Therefore, one can affirm that urology has become a cutting-edge specialty in robotic surgery department. Moreover, the independently control of the camera meant a great improvement in this area since the urological cavities are extremely narrow.

2.3. Maneuverability performance

Robot-assisted surgery has several advantages as well as limitations. Regarding the shortcomings associated, one has already explained that it has low accessibility due to the high price of the installation and maintenance (reason why the cost-effectiveness of these devices is frequently questioned), the large space the robot occupies or the previous training of the surgeon and assistant personnel [9]. But there are more technical difficulties resisting the fully acceptance of the *Da Vinci System*. As one can imagine, the setup of the entire platform prolongs the time of operation, its large dimensions compromise the access to the surgical site, the lack of tactile/haptic feedback sensation creates mistrust among the users, the need to undock and re-dock for patient positioning and so on. Besides, the robot is a mechanical device that can malfunction any time.

However, the aim of this project is to attack another aspect of the device: its dexterity, haptic feedback and maneuverability performance. In surgical terms, maneuverability is defined as the ease of making error-free movements in all possible surgically intended planes, directions, and degrees of motion. This can be quantitatively measured by the index of maneuverability [10], which examines how close the actual responses are from the ideal one.

In previous sections, one has exposed the different classes of surgical robots (active, semi-active and passive). This classification can be understood as well based on the robot's design, maneuverability, and degree of autonomy [11]:

- Passive systems: the robotic arms are unactuated and lack of any autonomy. They possess the lowest degree of maneuverability.
- Semi-active systems: the power supply is cut-off during critically demanding tasks so as to limit their motion in certain restricted or delicate anatomical spaces within the patient. These manipulators are still limited by their lack of maneuverability and in terms of total number of DOF.
- Active systems: all joints are actuated and intrinsically capable of performing one or more parts of planned or assigned tasks. The manipulator is capable of bending along its entire length as mimicking biology. Thus, it possesses maximum degree of maneuverability with a much larger number of degrees of freedom.

Before robotics, there were conventional video endoscopic techniques that, although being revolutionary, were hampered by limited instrument maneuverability and 2D visualization. These shortcomings took away the wrist-like motion of the human hand and the depth perception of human eyes. Years later these capabilities were returned to the professionals with the introduction of surgical robots, concretely by virtue of wrist-like instrument maneuverability and 3D visualization. Likewise, robotic arms fitted with *EndoWrist* instruments offer seven degrees of freedom and allow an extended range of mobility, which enables a high degree of delicate maneuverability even within confined spaces.

2.4. State of the situation and future development

It is an undeniable fact that robotic surgery offers many benefits compared to open or traditional surgery. Some of them have already been mentioned but in this section, one will explore them in more detail.

The major benefits for the patient include a shorter hospitalization, a clear reduction in the pain and discomfort

after the procedure, faster recovery time and return to normal activities, smaller incisions resulting in a reduced risk of infection, minimal scarring and reduced blood loss and transfusions.

But not only does the robot improve the patient's experience but also the surgeon's performance. Among the advantages that the robot provides the surgeon *there is* greater visualization, enhanced dexterity, and more precision. It allows the surgeon to operate in very tight spaces in the body that would otherwise only be accessible through open (long incision) surgery. The surgeon is provided with better accuracy, flexibility and control thanks to the possibility of programming the device to aid in the positioning and manipulation of the instruments.

Nevertheless, we must not forget that robotic surgery is a relatively new technology that keeps improving and evolving. It is analogous to the first computers, enormous and slow, and current phones which are tremendously fast and portable. In the same way, a similar cost and technology curve is expected for robotics in general [12]. Is just a matter of time that smaller machines, decision-making algorithms, augmented reality vision systems, better optics and sensory feedback emerge and shakes up traditional medicine again? Surgeons will have to incorporate robotics into their training, so that they can take advantage of this future opportunity.

3. Market analysis

Robotics is the branch of engineering and science that deals with the design, manufacturing and use of mechanical virtual robots. In the last decades, it has been proved that this multidisciplinary field can be applied to the healthcare sector. The progress in robotics is re-shaping almost all fields of human activity by overcoming limitations in surgery, rehabilitation, assistance or facility management. In this section, one will provide a brief overview on the evolution of surgical robots in the market.

3.1. Historical evolution of the market

As one has exposed in the state of the art, many advances and models of robotic systems have led to the current situation. One of the key aspects in the development, progress and commercial success of the technology is the regulatory approval of the system. For example, it took *ISS* (Integrated Surgical Systems, Sacramento, CA, USA) 6 years to gain the FDA approval of *ROBODOC* because it was difficult to prove its clinical benefits. Showing that the longevity of implants implanted with *ROBODOC* increased naturally, implied a long timeframe needed to assess the veracity of the claim. A shortened summary on the chronology of R+D (Research and Development) surgical robots is shown below [13]:

- 1992: *ROBODOC* has the first surgical robot with FDA approval.
- 1994: *AESOP 3000* is FDA approved for laparoscopic surgery.
- 1998: Dr. Friedrich Wilhelm developed an endoscopic camera.
- 2000: *Da Vinci System* is FDA approved.
- 2001: *ZEUS* robotic was FDA approved.
- 2002: *SOCRATES* developed for remote telesurgery.
- 2003: *Merge of Da Vinci System and ZEUS*.
- 2007: *SENSEI* is FDA approved.
- 2008: *Mirosurge* is developed in Germany.
- 2010: *SOFIE* a robot with force feedback.

As any other technology sphere, surgical robots are constantly being renewed. For this reason, some of these systems have become outdated or displaced by improved versions of new emerging companies. However, *Intuitive*' success and dominance of the market, and first to market position, is leaving less space for competitors as the number of procedures addressed by *Da Vinci* increases.

3.2. Current situation of the market

Nowadays, many devices are out in the market with different surgical purposes. In this section, one will expose the current top surgical robots from different world leading companies in the health-tech sector.

The most outstanding system is *Da Vinci* (S, Si, X, Xi, and SP) by *Intuitive Surgical*, the main powerhouse within robotic surgical systems. Among its specialties, there is urology, laparoscopy, gynecology, thoracoscopy general surgery and cardiac surgery. However, taking into consideration all the opportunities this system offers, the capital and operating costs must be taken into account too [14]. The *Da Vinci* surgical system ranges in price from \$0.5 to \$2.5 million, depending on the model, configuration, and geographic location. To the basic cost of the device,

we must add the instrumental accessories (\$200.000), the disposables and consumables per procedure (\$2.500), the annual maintenance after first year warranty (\$175.000) and the training of surgeons (\$6.000 each), although the training of the first four surgeons is included in the purchase price of the robot. So the millions of dollars needed to afford the robotic system is not negligible and will be taken into account in the SWOT Analysis afterwards.

Another robot is the *Senhance Surgical System*, a digital laparoscopic platform from *Transenterix*. The system consists of a multi-port robotic system that attempts to address the perceived weaknesses of *Da Vinci* offering similar surgeon control while providing 3D-HD vision, haptic feedback and surgeon camera control via eye movements [15]. It is based on a console platform consisting of a remote-control unit, manipulator arms and a connection node. Unlike *Da Vinci* recent generations, *Senhance System* comprises three arms. Besides, it uses reusable re-sterilizable instruments with “unlimited” uses, which has a positive impact on its cost in comparison with the competition. [16]

NAVIO Surgical System from Smith & Nephew directly targets the orthopedic market, which is not covered by *Intuitive Surgical Systems*. As opposed to the already seen systems, this is a handheld robot (see Figure 4) that facilitates real-time characterization of bone and cartilage and allows accurate bone removal. This instrumentation is widely used in total or partial knee arthroplasties [17].

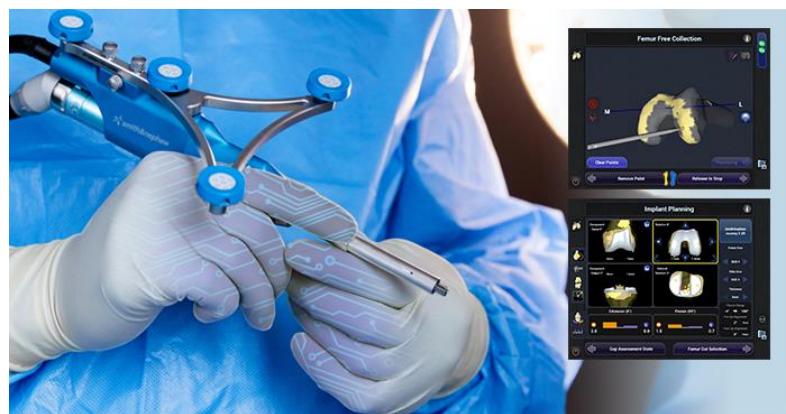


Figure 4. NAVIO Surgical System handheld robot representation. Real time anatomic characterization in the right interfaces.

Mazor X from *Medtronic* is a stealth robotic guidance system for spinal surgery. The power of the platform is the preoperative planning suite with 3D analytics and virtual tools that enables a predictable procedure with defined trajectories, preselected implants and no anatomical surprises. The software guides the surgical arm into position, translating the Surgical Plan to precision trajectory guidance thanks to the 6 DOF of the surgical arm [18]. *Mazor X* is indicated for precise positioning of spinal implants during general spinal and brain surgery; therefore it is neurosurgery oriented.

In a different surgical branch, there is the *Monarch system* or *ARES* (Auris Robotic Endoscopy System) which is specialized in bronchoscopy diagnosis and therapy such as lung cancer. It has a remote control similar to a videogame that allows physicians to navigate the flexible endoscope to the periphery of the lung [19]. The *Monarch* platform (the first FDA cancer-approved robot on the market) belonged to *Auris Health Inc* until *Johnson & Johnson* acquired the company in 2019, asserting its position in the steadily growing field of surgical robotics.

Medrobotics S-L developed *Flex Robotic System*, based on a flexible steerable scope that surgeons use to navigate around anatomical sites with an integrated 3D-HD vision system. Once it is in place, the scope can

become rigid to provide a stable platform through which flexible instruments can be deployed to perform procedures in a way that is not possible with line-of-sight approaches [20]. Initially approved for limited applications in otolaryngology, the system has received FDA approval for marketing in general surgery, gynecology and thoracic procedures. The differentiation value that this device offers is that it is carried out through natural body orifices, therefore it is even more minimally invasive.

Many other robots from different companies in this sector have been developed to overcome limitations of traditional surgery or other devices already in the market. For example, the *Bitrack* system was developed to be an alternative to the current laparoscopic surgical robot *Da Vinci* (in terms of efficiency and accessibility). It incorporates a flexible, modular and open robotic platform that improves the effectiveness of today's robotic surgery and makes it accessible to more hospitals around the world. It combines manual and robotic surgery, what is called Hybrid Minimally Invasive Surgery (HMIS) [21]. The new robot has already been technically validated in experimental models, used by surgeons from the Mayo Clinic (United States) and some Barcelona university hospitals: Clinic, Vall d'Hebron and Germans Trias. The project is being led by *Rob Surgical*, a spin-off created by IBEC and UPC in 2012.

3.3. Future perspective of the market

Attending the actual demands and growth of this field, it is clear that robotic surgery is here to stay. *Intuitive Surgical* dominates the industry despite its high initial and recurrent costs. This growth has prompted competitors to enter the field with new products that address some of the perceived weaknesses of the current offerings at lower costs.

We can identify three evolutionary processes that would lead to another revolutionary breakthrough. The first is the **level of invasiveness** of the procedure. Many companies are pushing themselves to minimize the impact and trauma of the surrounding tissue, reducing the risk of infection, enhancing a quicker recovery and reducing even more the hospitalization period. The problem is that to fulfill this goal, some other problems arise such as the need of smaller tools with fewer DOFs or more limited manipulability.

The second trend is associated to improve the **visualization capabilities**. Endoscopic cameras along with imaging modalities provide a view and representation of the anatomical structures. However, the physiology and function of the anatomy (neural activity, cardiac arrhythmia, etc.) cannot be represented yet. Including this in the robotic system would be a significant improvement for many surgical branches.

Finally, the third line of research is related to the **automation and control over the execution** of the surgery by the surgeon. This field can be attacked from two points of view: by improving the interface between the surgeon and the operating room or by enhancing the interface between the surgeon and the surgical site. The first approach refers to the possibility of making an entire OR fully automated so that there is no need of human presence, which it seems to be far from accomplishment.

4. Regulatory and legal issues

There are currently a large number of surgical robotic systems on the market, ranging from the *Da Vinci Surgical System* (used for a wide spectrum of surgical interventions, including urology and gynecology procedures), to Smith & Nephew's *NAVIO Surgical System* (used for orthopedic surgery).

Surgical robots can fit both in the product model or can be classified as a medical device. But, what about the software required to operate the robot? Is it a component of the product or is the software itself a medical device? The manufacturer of the robot will be the responsible for deciding whether the robot is a medical device and, if so, the classification of that device, which depends on the level of risk associated, the intended purpose, how long is intended to be used and if it is invasive.

Considering that this project is focused on robots as a medical tool for direct medical treatments, in the following sections one will describe the actual regulations of medical devices, which comprehend a wide range of technical requirements to be able to commercialize the product. In case a robot fulfils these standards, it will be able to obtain the FDA clearance (Food and Drug Administration of the United States of America) and/or the European Union certification (CE) to be commercialized in the respective continents.

4.1. Hardware

The International Organization for Standardization (ISO) is in charge of dictating the rules that regulate the market with the aim of normalizing it. Among the different areas that regulates, there is the medical field, in which it pretends to ensure the quality and safety of medical products. Some of the regulations already established are:

- **ISO 13485:2018** was designed to manage the quality of sanitary products, therefore it is related to Quality Management Systems. It specifies all the requirements about the design, the production, the installation and the service of this equipment [22]. This is a harmonized standard; hence it is recognized by the European Union and it must be complied in Spain.
- **ISO 10993:2018** document specifies the general principles governing the biological evaluation of medical devices within a risk management process. It assesses the biological safety of those devices that are expected to have direct or indirect contact with the patient's body or to the user's body (for the protection of the medical staff: gloves, masks...).

Other regulatory standards of the medical device industry include the **ISO 14971** (risk management), EU Medical Device Regulation (MDCG) and FDA regulations.

4.2. Software

The international standard **IEC 62304** – medical device software – is a standard which specifies the life cycle requirements for the development of software within medical devices (when the software itself is a medical device or the software is an embedded or integral part of the final prototype). This functional safety standard covers the design and maintenance of software and provides a set of processes, activities and tasks to ensure safety. It is also harmonized by the EU and the US.

5. Concept of engineering

For this draft, one will study different solutions regarding several hardware and software systems. Regarding the hardware solutions, one will analyze three robotic arms that could reproduce *Da Vinci's* maneuverability behavior with an *EndoWrist* needle, coupled to the robot with a 3D printed piece designed by the team. Besides the robot, other components must be studied for the final prototype such as the servomotors, the IMU sensor, an evaluation board (EVB) and so on. Finally, one will examine the different software options in order to choose the most suitable environment to create the programming code.

5.1. Hardware solutions

5.1.1. Robotic arms

A “cobot” or collaborative robot is intended for direct human robot interaction within a shared space or where humans and robots are in close proximity. These robots are characterized by lightweight construction materials, rounded edges and an inherent limitation of speed, force, sensors and/or software to ensure a safe behavior. Collaborative robots are generally dedicated to performing repetitive manual jobs so that it automatizes the process. Unlike industrial robots, cobots are designed to work (even interact) with people and can be easily programmed thanks to an intuitive interface. A cobot could be an interesting option to work in a hospital environment due to its safety parameters and its preparation to work around people. Therefore, although one is seeking for a robot to be applied in the surgical field, one should mention that the automobile industry is the one leader in the use of cobots.

In the following section, with the aim of selecting the most suitable robotic arm for our project, different collaborative robotic structures will be taken under study as well as some non-collaborative like Mover4. A considerable number of factors should be taken into consideration in order to properly select the best option. Not only referring to the economic viability, but also to the technical aspects of each solution.

5.1.1.1. UR5 robot arm

The UR5 from *Universal Robots* (UR) is a lightweight, aluminum robot with versatile a single arm. It possesses 6 degrees of freedom (six independent rotational joints), without a torso or an enclosing cage to keep it safely from humans [23], as can be seen in Figure 5. This robot complies point 5.10.5 of the standard EN ISO 10218-1:2006, which means that the robot may operate as a collaborative robot. It is not required to have safety guards between humans and the robot, which makes it possible to use the robot for medical applications.

The robot provides a range of challenging grasping and reaching tasks. As one has said before, it has inherent limitations to ensure safety. This built-in safety mechanism includes stopping when the robot joint torque deviates from the expected torque, a protective stop is also generated if joint velocity exceeds 3.2 rad/s or if the external force exceeds 150 N and, finally, it has an emergency stop button.

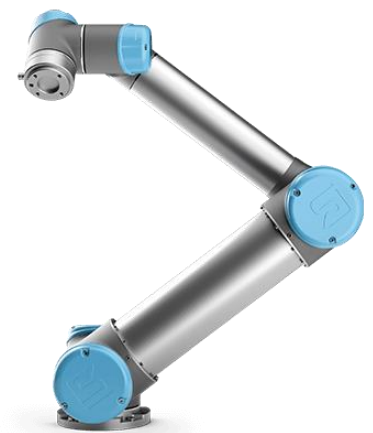


Figure 5. UR5 robot from Universal

The robot is lightweight (18 kg), compared to other industrial robots with similar features. For example, the *ABB robot IRB 1200* weighs 52 kg. Because it is lightweight, an impact with the robot would be less serious than for larger robots. The robot is connected to a computer, running the control algorithm using a direct Ethernet connection and a proprietary communication protocol.

As for the role of UR5 in the healthcare field, one of the major challenges this industry faces are inefficiency and being subject to high human error in non-automated environments. Hence, automatization of ITs (such as UR5 arm) can solve problems related to E- prescriptions, automated billing, electronic medical records and appointment reminders, advanced scheduling, clinic management, insurance claim automation and inventory management. In the next image (Figure 6), we can see two UR5 robots optimizing the handling and sorting of blood samples for analysis at the Copenhagen University Hospital in Gentofte [24]



Figure 6. UR5 robots in a hospital lab carrying out automatized tasks.

Another recent medical application would be robotic medical assistants monitoring patient's vital statistics and alerting nurses when there is a need for human presence. The robot automatically enters information into the patient electronic health record. In the next image (Figure 8), we observe the *Modus V* (developed by Synaptive Medical in Canada). It is used in robot-assisted neurosurgery with the most powerful optics available in the market with the aim of visualizing the patient anatomy and allow the surgeons to perform minimally invasive procedures with more precision. The solution incorporates a robot arm from *Universal Robots*. [25]



Figure 7. Modus V, from Synaptive Medical, coupled to a robot arm from Universal Robots.

5.1.1.2. Mover4 robot arm



Figure 8. Mover4 arm robot.

Mover4 is a robot arm from *Commonplace Robotics* (CPR) that allows to replay automation scenarios close to reality and can be used as a motion platform. With four degrees of freedom (4-axis robot), it can move free in space and turn the hand. Moreover, thanks to its middle size frame, this system can reach 550 mm of operating height and a payload of 500g. [26]

At the end of the arm, one can attach different tools to the flange. Therefore, it would be compatible to assemble our designed 3D piece, coupled to the *EndoWrist* TCP and carry out the project. In

addition, this product comes with its own software (as UR5). The control and programming environment allow to control the position of the robot in real time, to interact with it and to learn how to program it. Its 3D user interface is very intuitive and can communicate the data gathered to other programming environments such as *LabVIEW* or *ROS*, which are some of the software solutions that will be studied later on.

This company also offers two more models: the *Mover5* and the *Mover6*. In addition to the *Mover4*, the *Mover5* is able to rotate the gripper since it has 5 degrees of freedom. This allows to grip and release parts in the correct orientation. With the gripper, this prototype has a reach of 550 mm and a payload of 400g. In the other hand, *Mover6* robot arm allows to move the gripper with payload in all 6 dimensions. The robot has a reach of 600 mm including the gripper and a payload of 400g. [27]

Taking all this into consideration, this project will only consider the *Mover4* robot arm (among the options presented by CPR). The reason underneath is that its features are good enough for our project's framework and it is the prototype with lowest prize (which makes it truly competitive against UR5 robot), as we can see in Table 1.

Robotic arm	Price (€)
<i>Mover4</i>	3.135
<i>Mover5</i>	3.553
<i>Mover6</i>	4.714

Table 1. Prices list of 2018. This budget includes the robot arm mover and a complete set of accessories [28].

5.1.1.3. Sawyer robot arm

Sawyer is an industrial collaborative robot designed by *Rethink Robotics*. It is a flexible, easy to use, high-performing lightweight robot, which was developed specifically to take over precision tasks. It is controlled by its specially engineered software and operating system: *Intera*.

Sawyer BLACK Edition is an update of the robot which contributes to a quieter work environment and makes the cobot with a friendly face, as can be observed in Figure 9. The robot has 7 degrees of freedom, a payload of 4 kg and a range of 1.260 mm. *Sawyer* allows force sensing since it has sensitive torque sensors embedded into every joint. Hence, it allows constant force control that is used as a feedback in verification tasks. [29]

The *ClickSmart* gripper technology allows the robot to be deployed faster and easier in more tasks without time consuming customization. It can be trained by simply demonstrating the procedure moving its arm. It comes with an embedded *Cognex Vision System* in its arm that enables the Robot Positioning System (RPS) to provide for a dynamic reorientation and easy redeployment of the robot. The robot can maneuver in tight spaces much like a human arm. [30]



Figure 9. Sawyer robot from Rethink Robotics.

Finally, one must add that as a cobot, Sawyer is inherently safe and designed to work alongside people. Therefore, it is certified that meets ISO 10218-1:2011, like UR5 robot. Finally, it has an estimated cost of \$ 34.900 USD.

5.1.2. Servomotors

A servomotor (or servo motor) is a linear or rotatory actuator that allows for precise control of linear and/or angular position, acceleration and velocity. It uses a normal electric motor and combines it with a sensor in order to define its position, which is called position feedback. Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. The PWM sent to the motor determines position of the shaft and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired rotation.

For this project, two models were studied: *Dynamixel AX-12A* actuators (Figure 10-A) and *Longrunner ky66* (Figure 10-B). Some of the features that will be evaluated in these mechanical actuators are the operating voltage, the stall torque (the torque whose output rotational speed is zero), the rotation speed, the rotation angle and the angle resolution (which will define the accuracy of the system) among other parameters.



Figure 10. Servomotor prototypes under study A) Dynamixel AX-12A B) Longrunner ky66.

In the following table, some of these characteristics will be examined and compared in order to determine which one is the most suitable option for our final prototype:

Servomotor	Operating voltage	Stall torque (N·m)	Weight (g)	Rotation range (°)	Angle resolution (°)	Load feedback
Dynamixel AX-12A [30]	9 – 12 (V)	1.5	54.6	0 – 300	0.29	Yes
Longrunner ky66 [31]	4 – 7.2 (V)	0.2	9	0 - 180	-	Yes

Table 2. Description and comparison of each servomotor features.

5.1.3. IMU sensors

An Inertial Measurement Unit (IMU) is an electronic device composed by several accelerometers, gyroscopes and magnetometers that can report the specific gravity and angular rate of the object at which it is attached. Moreover, it measures the angular rate, acceleration, linear velocity and the magnetic field surrounding the object (to know its orientation). For this project's purpose, different IMUs have been studied. In the next table (Table 3), a comparison of the most important features of these component can be observed. Although IMUs have several parameters of interest, in this case, one will only consider the power supply of the sensor, axis of motion and size (since it must fit inside the haptic pen). All of them are compatible with *Arduino IDE*.




IMU	Power supply	Axis motion	Size	Model
MPU 9250	2.4 – 3.6 (V)	9-axis	3 x 3 x 1 mm	
MPU 6050	2.4 – 3.6 (V)	6-axis	4 x 4 x 0.9 mm	
BMI 055	2.4 – 3.6 (V)	6-axis	3 x 4.5 x 0.95 mm	

Table 3. Description and comparison of the Inertial Measurement Units. The axis of motion depends on the components of the IMU, for example, the 9-axis IMU has a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer.

5.1.4. Evaluation boards

To integrate and communicate the electronic part with the motorized instrument, one needs a microcontroller. This control hardware processes the information coming from the PC. In the next table, different Evaluation Boards will be examined depending on its cost, its power supply, its availability in the laboratory and the user experience using each board. All hardware's are compatible with *Arduino IDE*.




IMU	Input Voltage	Experience	Cost	Model
Arduino UNO [32]	3.3, 5, 7-12 (V)	No	19,55 €	
Arduino MEGA [33]	3.3, 5, 7-12 (V)	No	43,83 €	
STM32 Nucleo-64 [34]	3.3, 5, 7-12 (V)	Yes	13,46 €	

Table 4. Description and comparison of the evaluation boards.

5.1.5. Haptic feedback user interface

In the previous Final Degree Project, a Phantom Omni (Figure 11) was used for haptic control of the robot and is connected to the control computer as well. This device has 3 active DOF and 3 passive DOF [35]. In total, the cost of the cobot is approximately 35.000\$ although as it is already in the Faculty of Physics, its cost should be disregarded if it was considered as a final option [36].



Figure 11. Phantom Omni haptic device.

Although there are other haptic controllers in the market that one could study in this section, as this Final Degree Project is in collaboration with a Mechanical Engineering student, the idea is to design a haptic pen of our own that will be later on fabricated through 3D printing.

This would imply a reduction in the cost of the device although the materials and accuracy of the device would be reduced and the time needed to construct it, would be much higher. In addition, this self-designed pen would have some extra implementations such as a vibration motor, a buzzer and two push buttons for different purposes aimed to improve the haptic feedback and manoeuvrability of the system.

5.1.6. Vibration motors and buzzers

One of the limitations of the *Da Vinci* system is the lack of tactile perception when performing any surgical procedure. Therefore, a buzzer is installed in the pen so that it emits a sound once the surgeon overpasses a determined force one has established as a threshold. In the Table 5, different models are compared depending the frequency of the emitted sound, its intensity, its price and size (since it must be small enough to fit inside the pen). All of these buzzers can work with *Arduino IDE*.




Buzzer	Sound output	Frequency	Cost	Size (mm)	Model
Buzzer RS PRO [37]	95 dB	2.9Hz – 3.9kHz	2,78 €	14 x 6.7	
MCKPT-G1340-3917	80 dB	4kHz square wave	1,09 €	12.7 x 6.8	
VMA319 [38]	-	1.5 a 2.5 kHz	2,21 €	25 x 15 x 10	

Table 5. Buzzer comparison.

To test other methods of haptic feedback, a buzzer is also installed in the pen to emit a vibration once the force exerted by the user super passes the predefined threshold. In this case, the response must be controlled since an exaggerated vibration could affect the precision of the surgeon's movement and cause more damage than benefit

to the surgery. In the next table, we can observe the comparison between different buzzers and its main features.




V. Motor	Voltage	Cost	Size	Model
Seeed Studio Grove [39]	5 V	2,54 €	24 x 20 mm	
Parallax 28821 [40]	2.7 – 3.3 V	4,36 €	1 (diam) x 0.27 (thickness) cm	
SparkFun ROB-08449	3 V	2,20 €	10 mm (diameter)	

Table 6. Vibration motor comparison.

5.1.7. Push buttons

As commented before, another feature added to the haptic pen are the push buttons. One will have to choose two push buttons for two different purposes: advancement of the needle and opening/closing of the tweezers. For this purpose, one does not need very specific or expensive components, just a simple button that communicates to the computer and/or the tool, when the button of the pen has been pressed so that the corresponding action is performed. In the next table, a comparison of several buttons can be observed:




Button	Intensity	Cost	Company	Model
COM-00097	50 mA	0,36 €	SparkFun Electronics	
COM-10302	-	4,77 €	SparkFun Electronics	
B3F-1022 [41]	50 mA	0, 28 €	Omron Electronics	

Table 7. Push buttons comparison. The SparkFun buttons data was found in the commercial announcements, although no datasheets were found.

5.2. Software solutions

For the moment, one has only discussed and analyzed the hardware components of the project, but the software environment is important as well. In this section one will examine different software programs that are compatible with *Arduino IDE*, since this is the main SW one will use to communicate the robot with the computer and vice versa.

On the other hand, other software's are needed to design the pieces that will be printed (such as the haptic pen and the piece needed to couple the *EndoWrist* to the UR5 robot) in the EUSS. Although this part of the project belongs to Meiling, one has to consider this as part of the solution and budget of the total project.

5.2.1. Programming Software

In the table below, a comparison between the software environments that one has examined during the educational stage and studied during the Biomedical Engineering degree, can be observed:

Software	Cost	Experience	Arduino IDE
LabVIEW	433€ /year	2 years	Yes
ROS	Open source	1 year	Yes
RoboDK	Open Source	6 months	Yes
MATLAB	800€ /year	3 years	Yes
Python	Open source	3 years	Yes

Table 8. Comparison between software environments

5.2.2. Designing Software

Once we started collaborating with Meiling and the EUSS University and decided we wanted to design and created our own 3D pieces, one had to come up with different 3D designing software's capable of building three-dimensional structures that could be later printed. In the table below, one can see a description and comparison of the main characteristics of some 3D designing software's:

Software	Cost (€)	Operating systems	3D modeling
SolidWorks	6.600 – 10.950	Windows	Yes
FreeCAD	Open source	GNU/Linux, macOS, Unix, Windows	Yes
RhinoCeros	Open source	Windows, macOS	Yes

Table 9. Comparison between designing software's

Although it is not going to be compared and contrasted with other software's, one had to mention the *Kicad*

developer which has been used to design the electronic circuit of the PCB.

5.3. Proposed solution

In the previous section, several possible alternatives for the many hardware components and software environments have been presented and briefly compared. Now, the chosen prototype will be exposed as well as the arguments supporting this decision. One has taken into consideration the fact that as this project will be carried out in the University and economically financed with its resources, the selection of the proper hardware (and eventually, the software component too) cannot result to be very expensive.

Due to this reasoning, the proposed robotic arm should be the CPR robot, since it has the lowest price, and Sawyer's should be disregarded because of its high cost. However, the Electronics Department from the University of Barcelona has already acquired UR5 robot. Therefore, due to its already accomplished accessibility, one has no other option than to choose UR5 as the final solution. Choosing any other alternative would mean an extra expenditure since UR5 is being purchased anyway.

Furthermore, UR5 is the alternative with more experience in the surgical field. This argument is not determinant but provides confidence since we are not taking the risk of acquiring a robot that perhaps is not as efficient in the healthcare environment such as in industrial procedures. This way we will build upon firm and already explored ground. Another reason justifying this decision is the fact that one has already taken experience with this robot during the practical sessions of "Robotics and Control of Biomedical Systems" assignment.

Regarding the programming environment, UR5 robot arm provides a specially engineered software which is very easy to use. In addition, one trained before using it *in vivo*, through the *Universal Robots Academy's* online modules that provide core programming skills available to cobot users regardless of their robotics experience or backgrounds. These online formations include webinars, video tutorials, online training and in-class training. Once the online formation is concluded, one can get a certification (diploma) that verifies its knowledge about the functioning of several Universal Robot's devices. Besides the own interface of the robot, one can add or modify movements with external software environments such as *RoboDK* or *ROS* programs. This is very helpful since one has been training with this software in other stages of the project.

As for the smaller hardware components, most of them have imperceptible differences when it comes to power supplies, intensities, bandwidths, etc. Therefore, one accepted all the material and components that the teacher provided prioritizing its availability in the facilities. After having considered the advantages and disadvantages of the software's proposed, a brief summary on the chosen components and software's can be seen in the next table:

COMPONENT	MODEL
Evaluation Board	STM32 Nucleo-64
Push Button	COM-00097
Buzzer	VMA319
Vibration Motor	Seeed Studio Grove-Vibration Motor
Programming software	ROS, RoboDK, Arduino IDE
Designing Software	SolidWorks

IMU	MPU 9250
Servomotor	Longrunner ky66
Haptic feedback user interface	Designed with 3D software's

Table 10. Proposed solution of hardware and software

As there are a lot of components and modules to bear in mind, one has built an easy schematic to facilitate the overview of the project. As can be seen in the next picture (see Figure 12), all modules are controlled by the computer via *Arduino IDE* and *RoboDK*. Although, when working with UR5 robot, the interface needed is *ROS* instead.

From this central core, one can access to the actuator module: the UR5 robot and end effector tool (the *EndoWrist*) which will have the four servomotors attached to perform the desired movements (based on the RPY angles). Besides, the user interacts with the system with the 3D printed haptic pen (which will be seen in the *Detailed Engineering*) and contains the sensors needed required for force sensing and maneuverability goals. This means that the IMU, buzzer, vibration motor and push buttons must be fitted inside and connected to the STM board. In future studies or, in Meiling Chen's project, one can see a wireless connection to an Evaluation Board such as ESP32, which is a considerable improvement.

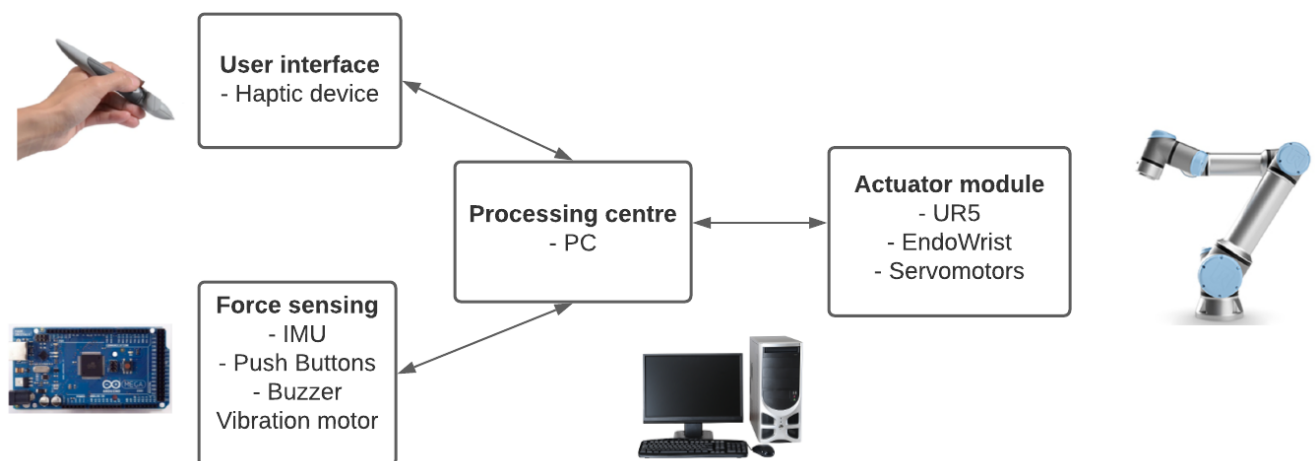


Figure 12. Schematic diagram of the main modules composing the system.

6. Detailed Engineering

This is the main section of the project in which one will exhaustively explain the development of the final prototype in the laboratory with the chosen hardware components and software environments that have been already discussed in the *Concept Engineering*. In addition, one will also show in further detail the development of the *Arduino* and *RoboDK* codes so that the reader can better understand the goals and functioning of the final prototype (regarding, mostly, the manoeuvrability improvements). Nevertheless, all the developed codes can be found in the annexes at the end of this document.

6.1. Hardware

To facilitate the understanding of the entire system, one will explain the develop of each component separately. First, an introduction to the inner working of the *EndoWrist* motors will be done since it is crucial for the programming code development and understanding. The tool one will be using is Intuitive 470006 *Da Vinci XI Surgical Large Needle Driver*, which can be seen in Figure 13.



Figure 13. EndoWrist end effector needle

One of the goals of the project is to create a programming code (with *Arduino* and *RoboDK*) that accurately sends the motion reproduced by the user with the haptic pen (which contains the IMU sensor, the vibration motor, the buzzer and two push buttons attached) to four servomotors that are geared to the *EndoWrist* system. Therefore, the needle will move as well. In Figure 14, a schematic diagram of this flow can be observed.

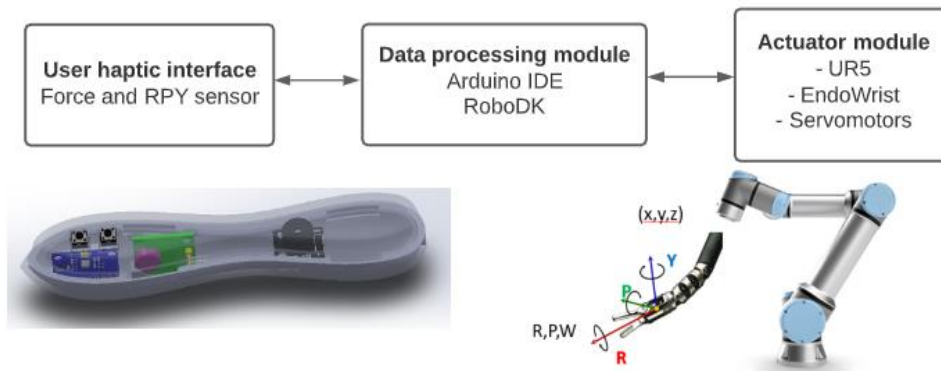


Figure 14. Working setup.

Although the transmission of the motion seems apparently simple, this mechanical module requires to be explained in detail since each movement of the joints (roll, pitch and yaw) that tries to mimic the human hand, is a result of a specific combination of rotations of the *EndoWrist* motors, which have to be triggered by the servomotors (controlled by the *Arduino IDE*).

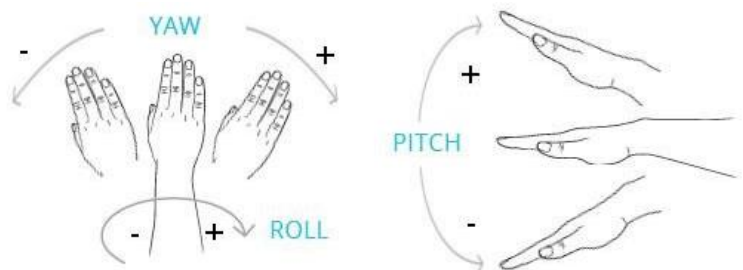


Figure 15. Illustration of how roll, pitch and yaw are measured in the human hand.

For this purpose, one will use a set of pictures of the previous project's author: Arturo Yscadar, who made a diagram of the joint motions associated to each motor and its rotation. In the Figure 16, one can see that the *EndoWrist* case has 5 motors although we are just going to use 4 of them to perform the three RPY movements: roll (R), pitch (P) and yaw (Y); and grasping, which can be done by moving one jaw or the other.

For the **roll motion**, only one disk needs to rotate (disk 1). This movement represents the rotation of the forearm/hand, also called pronation and supination of the hand extremity (see Figure 15). As we can see in the Figure 17, the rotations of the needle have the opposite direction than the one of the disk, for example: in the roll image one can see that the clockwise rotation of disk one creates a counter-clockwise rotation of the needle driver.

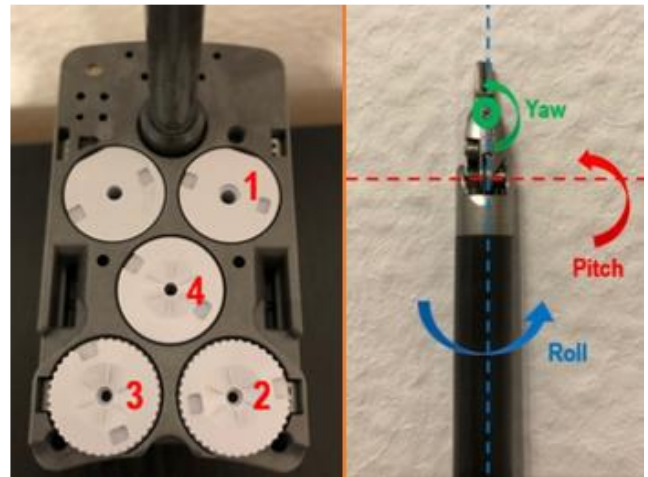


Figure 16. (Left) References of the disks. (Right) RPY movements of the tool. Photographs taken by Arturo Yscadar.

The **pitch movement** is a little bit more complex since it requires the motion of the disks two, three and four (see Figure 17-B). This motion represents the wrist flexion and, if we take a look at Figure 17-B, to rotate towards us the combination of rotations required is a counter-clockwise rotation of disk 4, a clockwise rotation of disk 3 and a counter-clockwise rotation of disk 2.

Regarding the **yaw movement**, which represents the lateral tilting of the human hand, only the lower disks are involved. Each one performs a rotation in the opposite direction: disk 2 needs to rotate clockwise while the disk 3 rotates counterclockwise the same amount of degrees. In the illustration below, all RPY movements can be observed, as well as the rotations involved:

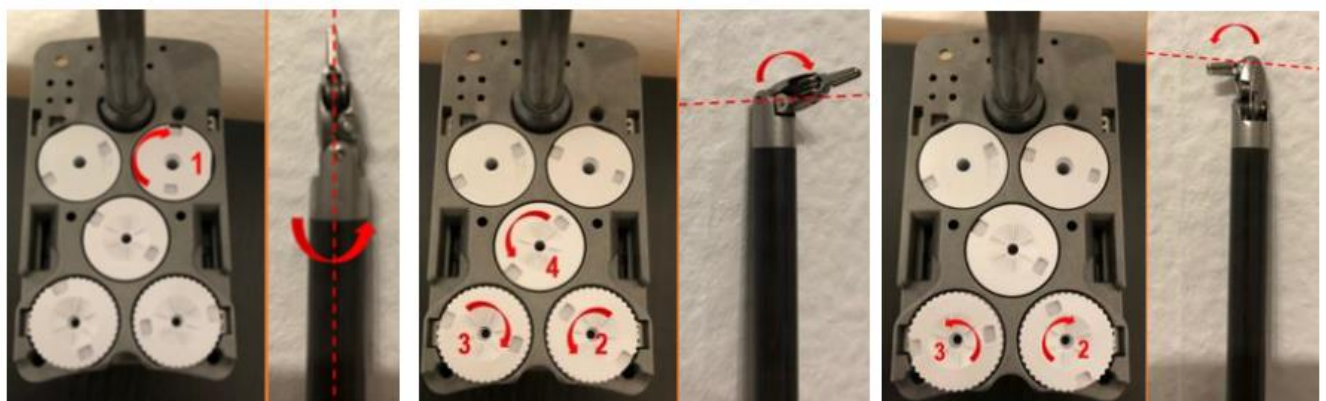


Figure 17. Motion of the needle driver. A. Roll. B. Pitch. C. Yaw.

Finally, for the opening and closing of the jaws one just needs to maintain one of the lower disks fixed and rotate the other between 90 and 100 degrees. With this, just one jaw would move so the tweezer would open. This movement is important when sewing during a surgical intervention since it is how the thread is subjected.



Figure 18. Example of the needle driver opening.

The Longrunner servomotors are interlocked with the *EndoWrist* disks. Then, after an *Arduino IDE* command is sent, the servomotors rotate, the motion is mechanically transmitted to the disks and, finally, to the *EndoWrist* needle. These servomotors are connected to a Protoboard and the STM32 EVB, which is at the same time connected to the computer via USB. Now, one will present the electrical circuitry of the project, which was based on the electrical connection of the hardware components to the protoboard and the evaluation board (see Figure 19). In this image one can observe three servomotors (due to limitations in power supply of the evaluation board, one did not want to risk adding another servomotor without an external battery), two pushbuttons, an IMU sensor, resistances ($1.6\ \Omega$) and the STM32.

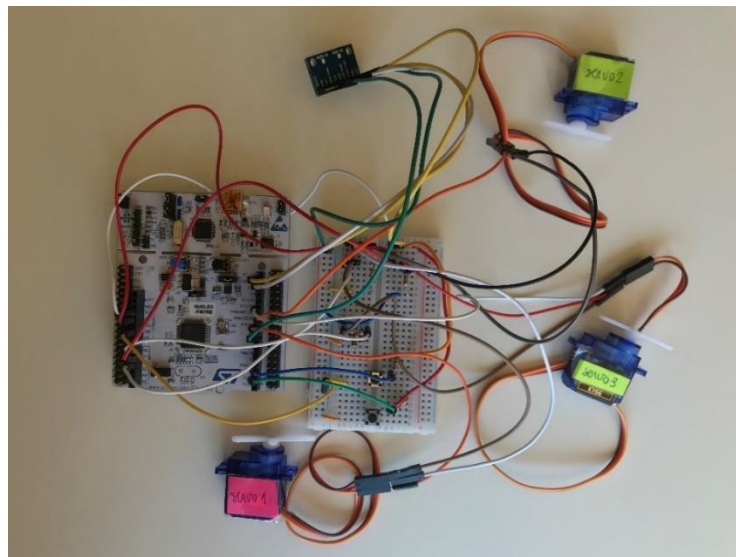


Figure 19. Overview of the entire circuit. Picture taken by the author.

Regarding the connectivity of the servomotors, they have three wires: a pulse width modulator (orange), Vcc of 5 V (red wire) and GND (brown). Therefore, one connected the Vcc wire to power supply of 5V (white wire of the protoboard is directly connected to the 5V of the STM32), the GND was connected to the evaluation board as well as to the resistance, and the PWM to the digital pins of the STM32. Finally, we add an Analog connection to the circuit (yellow wire).

Moreover, in the Figure 20-B, one can observe the connections of a pushbutton, which is very simple. In the left part of the image, it is connected to GND while in the right part, it is connected to 3.3V through the red wire, a small resistance and a digital output (yellow and white wires) that will tell us whether the button has been pressed or not.

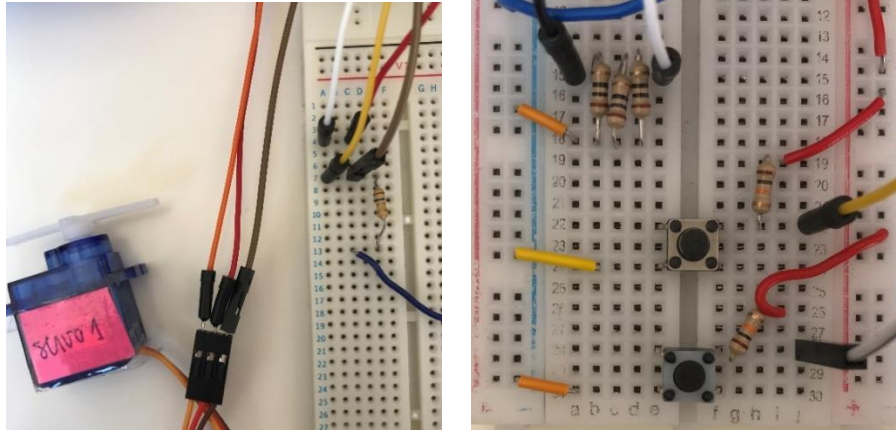


Figure 20. Electrical circuit of A. Servomotor. B. Pushbutton.

As have been observed, many wires are required, which increased the noise of the signal and its complexity. This is why Meiling Chen end up designing a PCB board with *Kicad* software (Figure 21) that had all the hardware components integrated in it and, with this, the pen could be used wirelessly. In addition, with the implementation of the ESP32 board, no USB port is required. In the first diagram we can see the PCB is electrically designed and in Figure 22 we can observe how it can be inserted inside the pen.

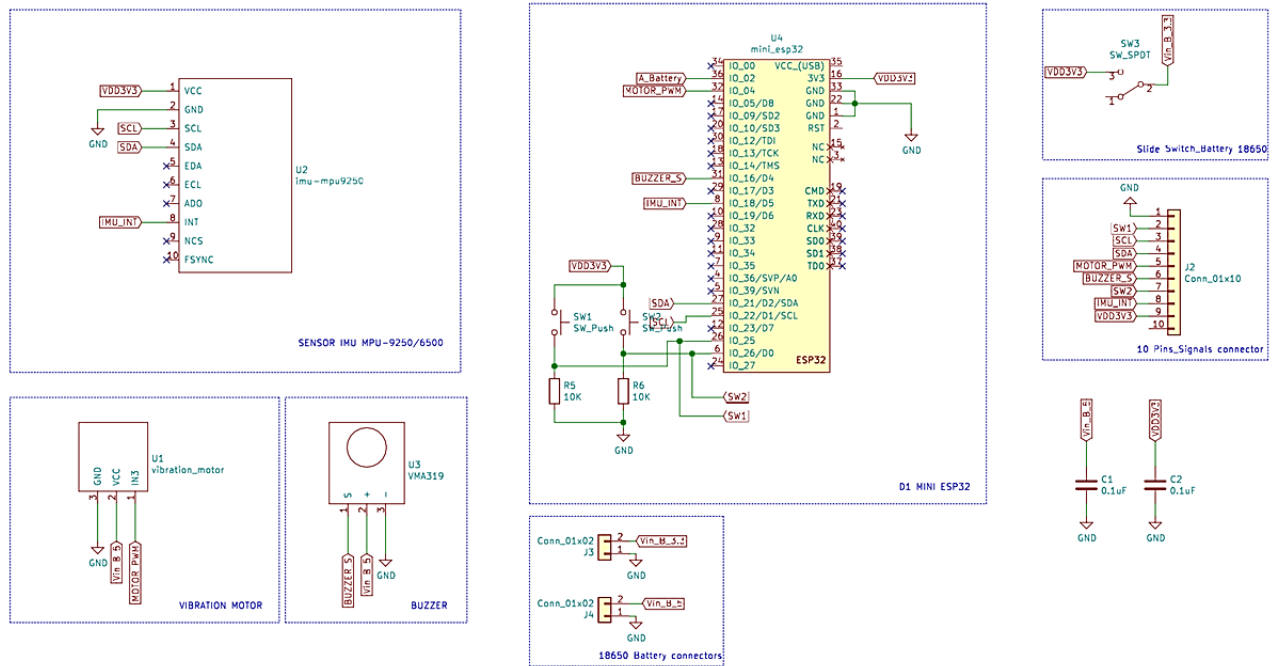


Figure 21. Electrical design of the PCB with Kicad.

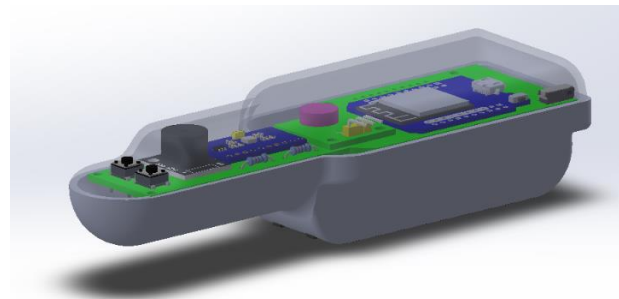


Figure 22. Design of the haptic pen with the ESP32 and the PCB inside.

With this prototype, no wires were needed and a reduction of space and therefore, of the size of the haptic pen, was accomplished.

6.2. Software

As one has already commented in the previous sections, *Arduino IDE* and *RoboDK* have been the two programming environments used to develop the code. In this part of the *Detailed Engineering* one will review the results obtained after the simulations and tests were executed.

A. Transmission of the RPY angles from the pen haptic to the servomotors

The term manoeuvrability arises from the transmission of the motion from the surgeons to the end effector tool. Therefore, good manoeuvrability should be instantaneous (no delay between the surgeon's movement and the *EndoWrist* motion), precise, noise- and tremor-filtered.

For all these reasons, to start the testing of the system, one created a program that transmitted the position and orientation (POSE) of the IMU sensor to a rotation of each servomotor as a result of the RPY angles. Concretely, the rotation around the X-axis (roll) is translated as the rotation of the second servomotor, the rotation around the Y-axis (pitch) represents the rotation of the third servomotor and finally, the Z-axis rotation (yaw) is translated as the rotation of the first servo. In the next code, one can see that the movement in the servomotor is limited to 180 degrees and that the command needed to rotate the motor is `servoX.write()`.

The final purpose of this program is that each servomotor is interlocked with the *EndoWrist* disks and transmit the motion. For example: the servomotor 2 must be geared to the first disk so that when it rotates, roll movement is performed.

```
void loop()
{
    current_milis = millis();
    if (digitalRead(PIN_IMU_INT) == HIGH) {
        imu.ReadSensor();
        rpw = imu.GetRPY();
    }

    // Angle range from 0 to 180 degrees
    if (rpw[0] <= 180 && rpw[0] >= 0)
    {
        motor_angle_X = rpw[0];
    }

    servo2.write(motor_angle_X);

    // Angle range from 0 to 180 degrees
    if (rpw[1] <= 180 && rpw[1] >= 0)
    {
        motor_angle_Y = rpw[1];
    }

    servo3.write(motor_angle_Y);
}
```

Figure 23. Part of the Arduino code showing the transmission of the motion to the servomotors.

B. Acquisition and visualization of the RPY angles from the pen haptic

The Inertial Measurement Unit (IMU) sensor moves along the x, y and z axis and, depending over which one of them we rotate, we'll perform a roll, pitch or yaw movement (see Figure 24).

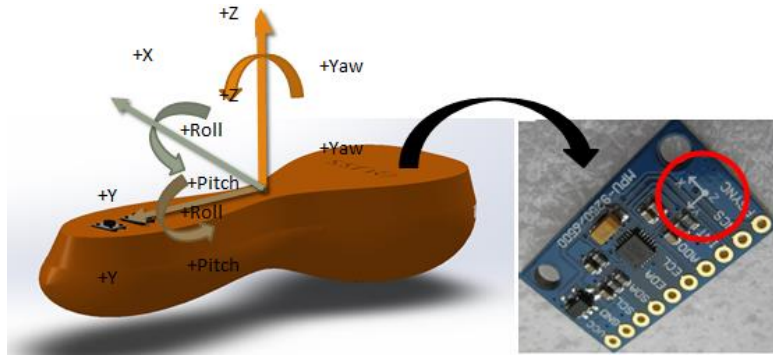


Figure 24. Diagram of the RPY angles on the haptic pen and the IMU.

In this program, the goal is not only to move the motors but to visualize the RPY angles in our computer, concretely in a *RoboDK* graphic interface called *TKinter*. This flow is represented in Figure 25 for a better understanding of the process. Besides the `write()` function, one has to read the angles thanks to the `Serial.read()` *Arduino* command. In this code, the pushbuttons are included although its function is not implemented yet.

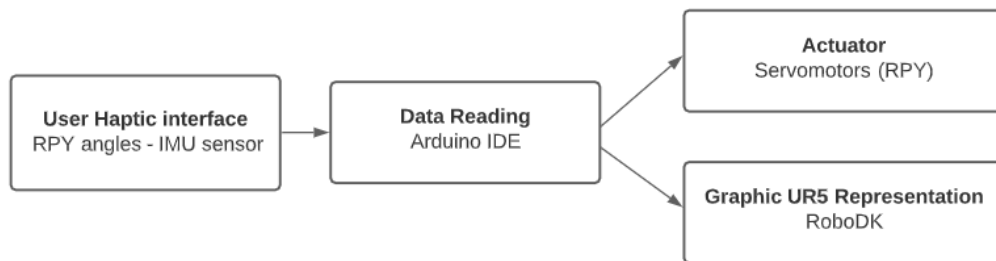


Figure 25. Workflow of software A. Acquisition and visualization of the RPY angles of the pen haptic.

As for the *RoboDK*, the *TKinter* interface is used to obtain the RPY angles orientation. First of all, a start page is opened and asks the user to push a button in order to obtain the angles and, once this is done, the right window appears and the roll, pitch and yaw values are constantly changing as long as the IMU sensor is moving too. As can be observed, both pushbuttons get a value of 1. This is because they are not being pressed, once they are, the value will change to 0.

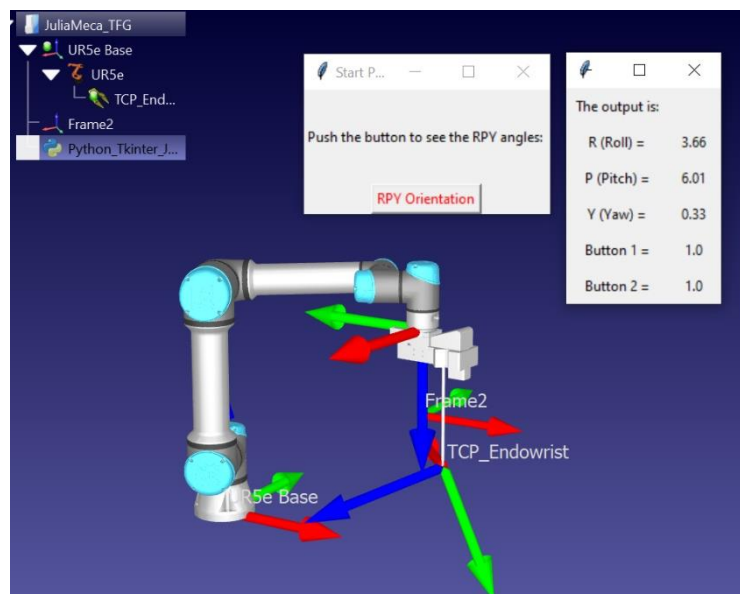


Figure 26. RoboDK program showing the *TKinter* interface of RPY acquisition and representation.

C. Transmission of the RPY angles from the computer to the servomotors

In this part, one is focused on sending a specific angle from the *RoboDK* environment to the servomotor so that the *EndoWrist* moves precisely. This was done by writing the angles in the output terminal (as can be seen in Figure 27-A) or through a *TKinter* graphic interface that allowed to change the RPY angles progressively with a slider.

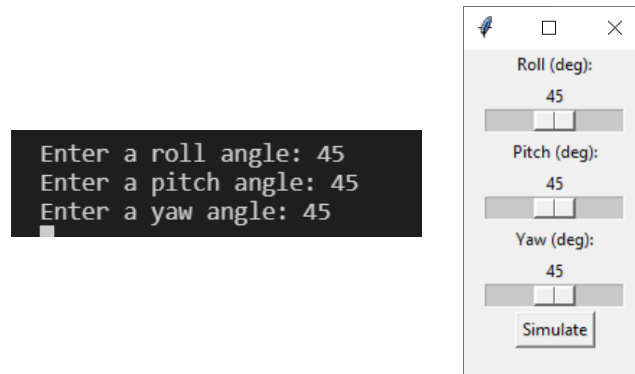


Figure 27. Introduction of the RPY angles in the RoboDK interface A. from the terminal B. Through a slider (Tkinter)

Once the program is run, the servomotors rotate 45 degrees each one (in this example) and the UR5 graphic tool of the *RoboDK* interface does it too. This can be observed in the illustration below, in which the *TCP_EndoWrist* that represents the end effector, has rotated 45 degrees around each axis (x, y, z).

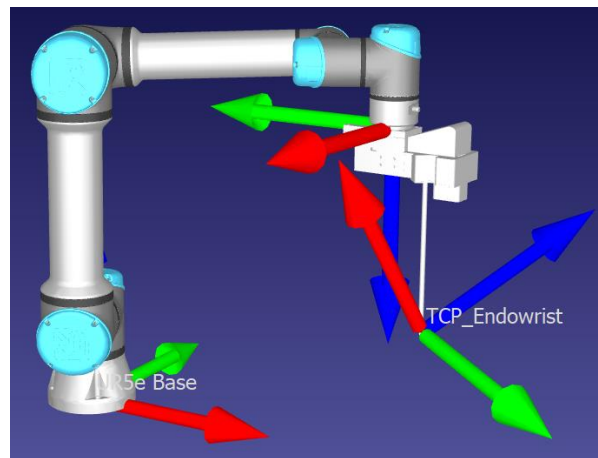


Figure 28. RoboDK representation after a 45-degree rotation of each servomotor.

D. Advancement of the TCP after pressing the pushbutton

In previous section, one has mentioned the purpose of the pushbuttons. In this case, our goal is to move the end-effector tool in the X-axis direction while the first button is being pressed (in other words, while its value is equal to zero). In the next picture (Figure 29) one can see how the robot was intended to go from the first frame *TCP_EndoWrist* to the second *TCP_Moving* which is constantly changing in function of the IMU sensor motion and orientation. As can be implied by looking at the picture, the program has not been fully perfected since only the frame was moving instead of the robot itself. However, in future studies this can be easily fixed with some assistance and more time.

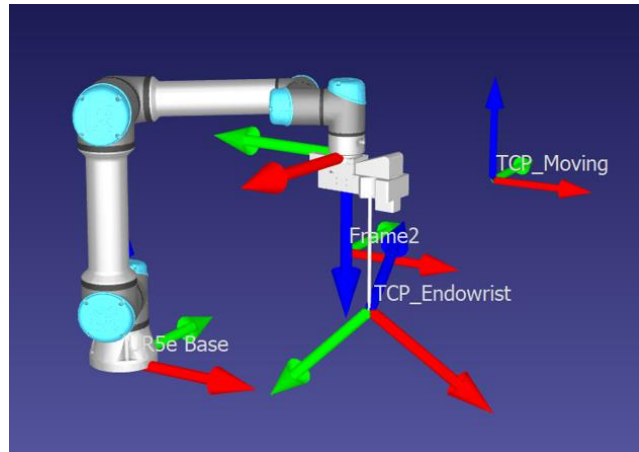


Figure 29. RoboDK representation of the advancement in X-axis.

E. Opening of the needle jaws after pressing the pushbutton

Finally, this goal consists on opening the needle driver to subject the thread while suturing as long as the second pushbutton is pressed. An option could be to open both jaws at the same time but to simplify the code and be more efficient, one has chosen to only open one of the jaws while fixing the other one. This has been also seen in the Figure 18 and its corresponding description and disk rotation.

In the next image (Figure 30), one can observe the programming code that fulfills this purpose. The buttons must be read as digital inputs in which "0" represents that the button is being pressed and "1" that it is not. Finally, with the `servo.write()` command, the servos rotate the desired amount of degrees. Then this motion is transmitted to the *EndoWrist* disks and the needle moves.

```
pinMode(pushButtonA, INPUT);
pinMode(pushButtonB, INPUT);
}

void loop()
{
    imu.ReadSensor();
    rpw = imu.GetRPY();
    int buttonStateA = digitalRead(pushButtonA);
    int buttonStateB = digitalRead(pushButtonB);

    if (buttonStateB == 0)
    {
        // Open the jaw 90 degrees
        motor_angle_Y = 90; // we open a needle
        motor_angle_Z = 0; // but fix the other
    }
    servo3.write(motor_angle_Y);
    servo1.write(motor_angle_Z);

    if (Serial.available() > 0)
    {
        Serial.print("Output data: ");
        Serial.println(String(motor_angle_Y, 2));
        Serial.println(String(motor_angle_Z, 2));
        Serial.println(String(buttonStateB, 2));
        delay(1);
    }
}
```

Figure 30. Arduino code describing the opening of the needle when pushbutton is pressed.

7. Experimental validation

7.1. Assembly of the servomotors and the EndoWrist tool

The *EndoWrist* disks have a central hole and two cracks in each side of the circle (see Figure 31). This is helpful to interlock the disks with the motors and fixed them all, although this process is not as easy as it seems. By now, a program that automatically gears all the servos with each disk independently and autonomously, has not been developed yet. A more rudimentary method was implemented thanks to Meiling, who designed four plastic pieces with certain slope so that when the servos rotated, they involuntarily get fixed in the cracks. These 3D printed pieces can be seen in the right illustration.



Figure 31. Mechanical solution to gear the servomotors with the *EndoWrist* disks.

7.2. Final setup with the UR5 robot

Once the servomotors can be coupled to the *EndoWrist*, the tool is placed inside a plastic structure (also fabricated by 3D printing and designed with SolidWorks) which allows us to attach the *EndoWrist* to the UR5 robot. This structure can be seen in the next image (Figure 32) in blue, supporting the end-effector tool, with a white case encapsulating the servomotors.

In addition, we can observe a camera on the top of the system, in charge of recording the action that is taking place near the TCP, just as the *Da Vinci System* has a high-definition 3D camera (or as happens in MIS where a trocar with an incorporated vision system is introduced inside the cavity). However, this camera can be better seen in the illustrations below.

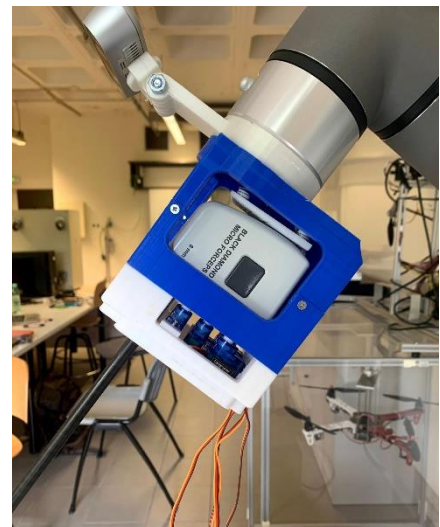


Figure 32. Plastic piece supporting the *EndoWrist*, servomotors and a camera.

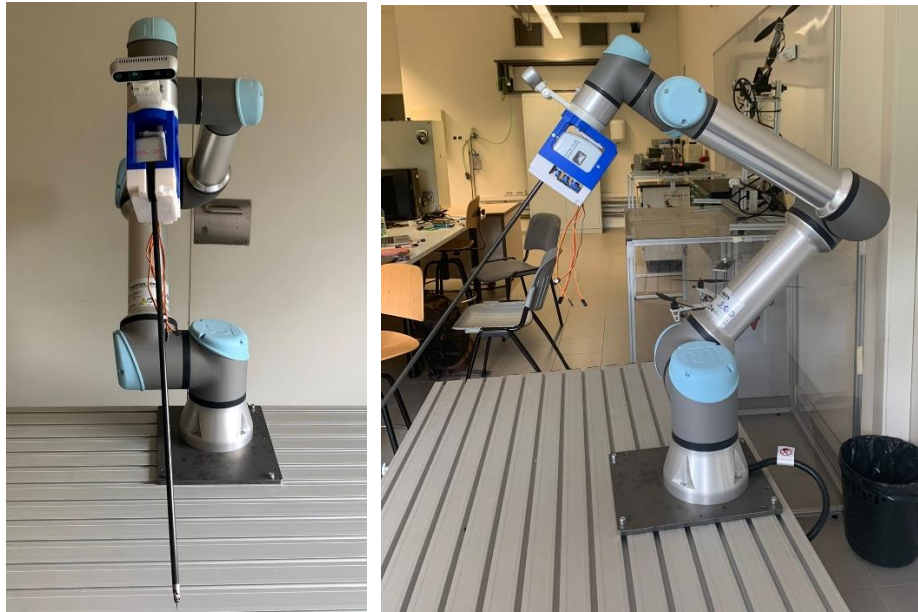


Figure 33. Pictures of the final setup with the UR5 robot.

As one can imagine, this prototype is not finished. Here we should connect the evaluation board with the STM32, and all the circuitry attached. The problem is that the position of the robot was so high, and one could not place the materials on the table.

Moreover, another option could be to use the haptic pen which is wireless and works with the ESP32 board. Unfortunately, this setup will have to be prepared in the future prospects of the project.

8. Technical feasibility

In this section, one has analyzed the corresponding strengths, weaknesses, opportunities and threats of the proposed solution. The SWOT analysis represent the positive and negative aspects coming both externally and internally from the project.

In one hand, one has encountered for technical incompatibilities, errors, and other problems when mechanically building a prototype or programming the code with the different software. On the other hand, some limitations are related to the lack skills of the user as a student when creating these pieces and programs.

Moreover, one has to take into account the global scenario in which this project has been developed (a global pandemic) and the limitations it raised. In the next table, one can see a brief summary on the main internal and external strategic factors of the project:

8.1. SWOT analysis

STRENGTHS

- UR5 has a strong software development, very complete and easy to use.
- Good systems integration capability since it can be used with other programming environments such as *ROS* and *RoboDK*.
- Safety of the product is ensured thanks to several mechanisms that stop the device if certain conditions are met.
- Certification of ISO standard ensuring its quality and safety.
- Most economic option taking into account that the University was already acquiring the UR5 robot and the haptic pen is being designed by the team.

WEAKNESSES

- High budget: although being the most economical option, it is still expensive to develop since UR5 supposed a high investment.
- As it is not possible to manipulate the real *Da Vinci* system, all tests and performance must be performed in the designed robotic arm prototype. Therefore, the real functioning and impact of the maneuverability and haptic feedback when operating with the *Da Vinci* robot cannot be directly proved.
- A lack of experience in robotics and robot assembly have led to approximate assumptions and comparisons based on information found on the internet. This might cause technical errors in further stages of the project.
- Lots and strong healthcare regulations required for having the FDA or CE approved.

OPPORTUNITIES

- It can maintain and expand in the market with its online educational platform and personnel training.
- Proximity of the project to Hospital Clinic, to do the internship and observe the setup and performance of the *Da Vinci* system, and to the Electronics and Biomedical Engineering Department, where the

mechanically assembly and evaluation will be carried out.

- This project mainly intends mimic the nearly excellent maneuverability performance offered by the *Da Vinci* system in order to solve related problems in other robotic-assisted surgery systems. Other factors such as the haptic feedback, the tremor filtering, the accuracy, the degrees of freedom... could be also taken under study.
- Personally, it has been an opportunity to learn about robots and electronics, since one had very limited knowledge about it before starting this project.

THREATS

- COVID-19 pandemic restrictions prevents us to do the internship in the surgical service of Urology Department at the Hospital Clinic, which would have been useful to gather information on the manoeuvrability and performance of the *Da Vinci System*.
- Similar projects have been and are being carried out in order to commercialize their devices. Although the introduction of the prototype in the market is not among this project's goals; the competitiveness of these groups, its resources and knowledge might leave our project out-of-step.
- Medical devices are subjected to continuously changing regulations and legislations.

9. Execution schedule

Many techniques are used to plan the project. First, we'll analyze the WBS and the description of every activity and task included. Afterwards, one will elaborate the PERT diagram and the GANTT based on the breakdown of activities, its duration and precedence.

9.1. Work Breakdown Structure (WBS)

In this section, one is going to define all the activities required to develop this final degree project. To build the Work Breakdown Structure is very important in the project's planning because it describes the phases and tasks as well as the responsible (in case of teamwork) and deadline of each one of them.

In the following representation (Figure 11), we can see the simplified structure of this project's WBS. In darker colors, we can see the major functional deliverables that correspond to the general phases of the project. Finally, in a brighter tone, *there is* the decomposition of these packages into a list of tasks or "to-dos" that produce specific units of work.

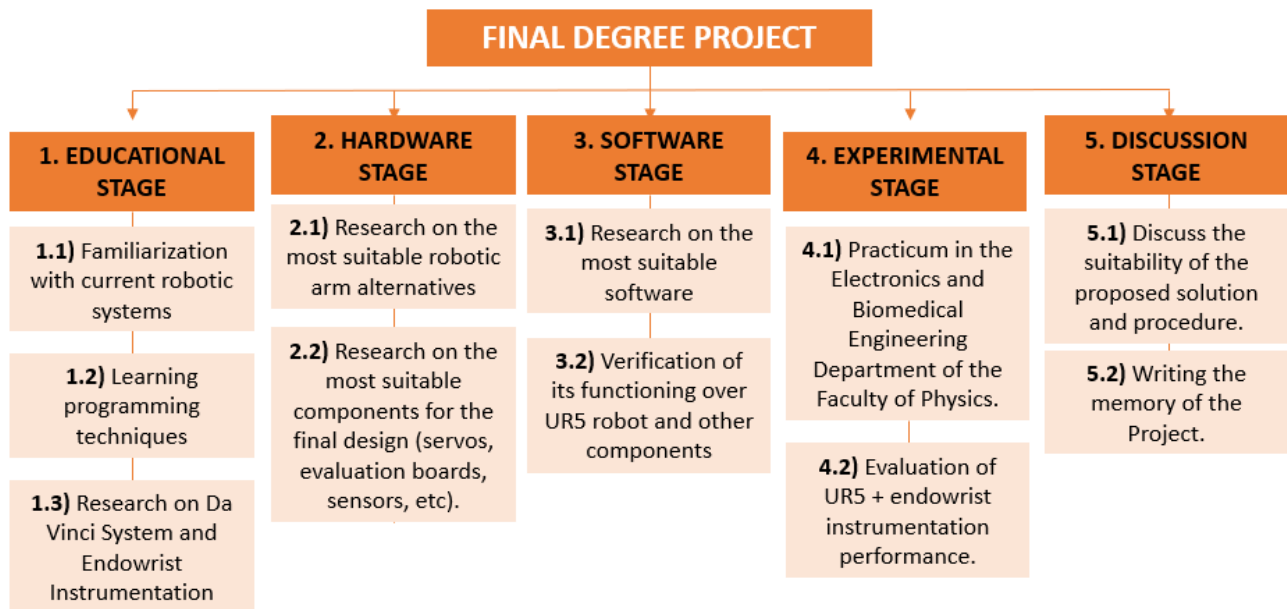


Figure 34. Work Breakdown Structure of the project

9.1.1. Dictionaries and duration of activities

In the following table (Table 12), one will expose the WBS dictionary with the detailed information about the deliverables and tasks, explaining on what they consist on. One will assume that the duration of a stage corresponds to the sum of its corresponding tasks, which will be further taken into account in the development of the GANTT chart. The responsible of every activity is the author of the project.

Nº	Name of the task	Description
1	Educational stage	During the whole process of educational stage, it has been performed a bibliographic research in order to familiarize, characterize and broaden the knowledge related to robotic assisted interventions, current systems in the market, <i>Da Vinci</i> Surgical System, <i>EndoWrist</i> and UR5 systems. In addition, this stage includes the tutorials used to learn this programming techniques.
1.1	Familiarization with robotic systems	A bibliographic research will be carried out on the world leading robotic systems such as <i>Senhance</i> , <i>NAVIO</i> , <i>FLEX</i> , etc. Its most important differentiating features will be analyzed as well as its presence in the market.
1.2	Learning programming techniques	Different tutorials (based on videos or exercises) have been completed with the aim of learning its strengths and weaknesses and being able to assess which is the most suitable environment. In the case of <i>ROS</i> , one has completed a 10-hour tutorial
1.3	<i>Da Vinci</i> Research	Explanatory document explaining the features, limitations, and performance of <i>Da Vinci</i> surgical system and <i>EndoWrist</i> instrumentation both patented by <i>Intuitive Surgical</i> . This task corresponds to a bibliographic study of this system and it does not include the learning on the device during the sessions in the Service of Urology.
2	Hardware stage	Study of the possible robotic arms that could fulfil our goal of reproducing the <i>Da Vinci System's</i> manoeuvrability. We'll focus on the performance of UR5 robot since it is the one that the University will acquire
2.1	Robotic arm alternatives	Bibliographic research on the different robotic arms that successfully works nowadays in the surgical field. Among them, one will study more in deep the technical features of UR5 due its availability in the lab.
2.2	Research on components	Bibliographic research on the different components such as the evaluation boards, the push buttons, the buzzers, vibration motors, etc.
3	Software stage	Analysis of the different software that can be used to program the movements and reading out the information (feedback). In addition, one will have to verify if the optimal software is fully compatible with the robotic environment, we want to apply it on.

3.1	Research on software	This task consists on the search of information of the most used <i>software</i> and programming tools that are used in surgical robots. Examples of this are <i>Arduino</i> , <i>LabVIEW</i> , <i>ROS</i> , <i>RoboDK</i> , <i>Universal Robots'</i> interface, etc.
3.3	Verification	Once the software has been chosen due to its features, learning curve, and reading velocity; one has to test in the lab if this software is also the most suitable one to work along with UR5 robot.
4	Experimental stage	Assembly of the different elements in order to build the prototype and evaluate it in terms of manoeuvrability and haptic feedback. In addition, one will be able to do an internship on the Hospital Clinic in relation to the <i>Da Vinci System</i> .
4.1	Practicum	200 h of practises in the Electronics and Biomedical Engineering Department of the Faculty of Physics.
4.2	Final validation	One must ensure a proper physical coupling to the robotic arm prototype. The eventual coupling in the lab and the evaluation of different movement transmission systems will be carried out.
5	Discussion stage	Selection of the more appropriate software and hardware, evaluation of the suitability of the designed prototype and elaboration of the final degree document (memory of the project).
5.1	Final prototype discussion	A final study has been performed in order to discuss the obtained results and to be able to state a consistent conclusion (selecting the more appropriate configuration) coherent with the maneuverability and haptic feedback provided.
5.2	Memory	Finally, the memory has been elaborate following all the sections previously mentioned.

Table 11. Descriptions of the tasks of the Final Degree Project.

9.2. GANTT chart

It has also been represented the same information (task and timing) but this time as a GANTT diagram, where it can be observed the task flow of the project (see Figure 35). To build this chart, a planning of the beginning and end of each activity is required. This information is attached in the next Table 12. It is worth mentioning that this time schedule does not consider the associated limitation of the OR availability and surgeon's schedule.

Number	Name	Duration (days)	Start	End
1.1.	Familiarization with robotic systems	150	15-Feb-20	1-Jun-20
1.2.	Learning programming techniques	420	15-Feb-20	1-May-21
1.3.	<i>Da Vinci</i> Research	120	15-Feb-20	1-May-20
2.1	Robotic arm alternatives	150	15-Feb-20	1-May-20
2.2.	Research on components	120	1-Mar-21	28-Jun-21
3.1.	Research on software	30	28-Mar-21	28-Apr-21
3.2.	Verification	90	1-Apr-21	2-Jun-21
4.1.	Practicum	150	2-Feb-21	10-Jun-21
4.2.	Final validation	90	1-Apr-20	14-Jun-21
5.1.	Final prototype discussion	60	1-May-21	14-Jun-21
5.2.	Memory	120	15-Mar-21	13-Jun-21

Table 12. Project tasks with the corresponding duration and timings

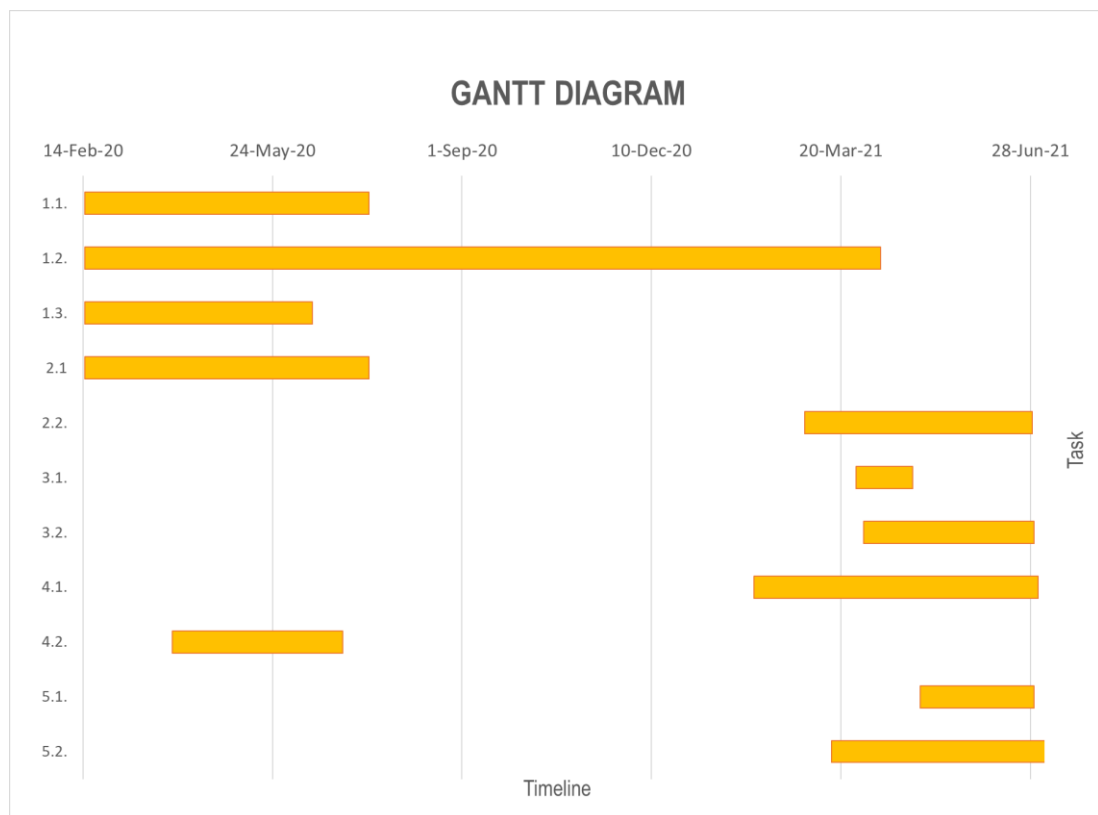


Figure 35. GANTT diagram

10. Economic viability

In this section, one will consider the price of fungible products (such as the robot), the time invested as professionals and the software licenses required for the performance of the entire project.

Although the project has implied the work of three students, one will only consider the working hours of the author supervised by a professor in charge of the project. It is considered that a student should have a salary of approximately 15 € / hour. Considering that a Final Degree Project should take, at least, 300 hours, the personal expenses (disregarding the professor since he has its own salary), would reach approximately 4.500€.

With respect to the software used, one has trained with different programming environments such as *LabVIEW* (which license is financed by the UB), *Arduino*, *RoboDK* and *ROS* (which are all free). For the bibliographic research, one has read several articles which access has been allowed thanks to the PubMed License.

Finally, one will take into account the costs that rise from the purchase of UR5 robot and the Hardware components. The breakdown of the project costs is shown in the next table (Table 13). Some prices which has been exposed in dollars (USD) are now converted into Euros (€) so that they can be added.

Components	Cost per unit	Units required	Total cost (€)
SOFTWARE			
Arduino IDE	Free	1	-
RoboDK	Free	1	-
ROS	Free	1	-
PubMed License	Financed by UB	1	-
SolidWorks License	Financed by UB	1	-
HARDWARE			
UR5 robot	30.996 €	1	31.000 €
EndoWrist TCP	Recycled	1	-
Computer	1.500 €	1	1.500 €
STM32 Nucleo-64	13,46 €	1	13,46 €
COM-00097	0,36 €	2	0,72 €
VMA319	2,21 €	1	2,21 €
Protoboard	8 €	1	8 €
Sseed Studio Grove-Vibration Motor	2,54 €	1	2,54 €
WORKING HOURS			

Biomedical Engineering student	15€ / hour	300 hours	4.500 €
TOTAL			37.027 €

Table 13. Estimation of the cost of the project.

Afterall, the total price of the project would be of 37.027 €. However, one must mention that this estimation of the components might change eventually since the intention of the author is to use a base shield and battery supplies that provide more power. Moreover, it seems that *Arduino* Mega evaluation boards can provide better functionalities than STM32 boards, which would vary the total budget of the project as well. Hence, one estimates that this price will rise at the end of the project.

11. Discussion and conclusions

In addition to the resulting simulations and whether the final prototype has been assembled or not during the time the team has been in the Biomedical Engineering Department, additional criteria has been taken into account to formulate the conclusions of this project at the end of the chapter.

In this section, one will analyse the objectives that were mentioned at the beginning of the project to examine if they were accomplished or not and the reasons for each scenario. In the last section, some recommendations will be made for the future development of the project in terms of new materials and components as well as new approaches of the research.

11.1. *Achieved objectives and results*

During these months in the Electronics and Biomedical Engineering Department, one has get acquainted with many software environments, hardware and electronical components which were unknown at the beginning of this practicum. Therefore, one can solidly affirm that this experience has been very rewarding and educational since a lot of knowledge in electronics and robotics has been acquainted, as well as the current situation of the world leader companies in health-tech.

The most relevant achievement of this project is the reproduction of a surgical robot such as the *Da Vinci System* in the laboratory facilities with UR5 robot. With the use of very simple hardware components (components used in the daily assignments of a Biomedical Engineering student), we have been able to create a system that reproduces the mobility of a haptic pen through electrical circuitry but also wirelessly, to the end effector tool (*EndoWrist*) attached to the robot. Moreover, a new design (different from the one of *Intuitive Surgical*) for the coupling structure between the robot and the tool was created which, once it is more tested, might allow a faster and easier change of instruments.

In addition, different components have been added to improve the manoeuvrability performance such as the pushbuttons which have shown good results in the tests of advancement and opening of the needle. These applications have a lot of future in the sewing area since it is easier to press a button and that the *EndoWrist* moves ahead on its own, instead of the hand of the surgeon. Analogously, it is more comfortable to control the tweezers with a button rather than with the tips of our fingers.

Although the studies on the haptic feedback and the torque have not been seen in this Final Degree Project, the buzzer and the vibration motor have successfully fulfilled its purpose of warning the surgeon when arriving to the force threshold. Nevertheless, one has to be careful with the vibration of the haptic pen since we cannot allow large movements of the pen that could damage the patient.

An objective that one has not accomplished due to lack of time is the filtering of the tremor. The goal was to reduce the difference between an RPY angle (of the IMU sensor) and the previous one. Based on this, a large variation in the degrees between two consecutive angles could only mean a dangerous tremor of the hand or that the pen has fallen down. In any case, the angle should not variate any longer to maintain safety conditions for the patient.

Moreover, the final assembly of all the components and the evaluation of its performance could not be

accomplished since the 3D printed pieces were completely finished late and some limitation regarding the power supply and the evaluation board were found. In addition, new material that could have fixed some of these limitations, arrived during the last weeks of the Final Degree Project but there was no time to implement these modifications in the project.

However, alongside with Meiling and Maria, we tried to fulfil all the objectives as far as we could so that the next team that takes over the project can easily take on our work and solve these issues. In the next section, one will mention which are these materials that can be useful (and already available) for future studies.

11.2. *Future opportunities*

This project's contribution is the continuation of an encouraging improvement for a research environment which is prepared for minimally invasive surgeries. Down below, some outstanding tasks focused to achieve a better performance of the current system are proposed:

- The implementation of a wireless evaluation board such as ESP32 so that the pen has not to be connected to the computer.
- To manufacture the instrument support in a rigid and resistant material such as metal. A better fixing capacity would avoid movements in the encapsulating system, which are translated into vibration at the end of the end-effector tool. If this was not possible, one could improve the 3D impression since the current prototypes easily break and shatters.
- To include robust and easy method to calibrate the IMU sensor each time it is used. Low-cost embedded IMU sensors are affected by systematic errors given by imprecise scaling factors and axes misalignments that decrease accuracy in the position and attitudes estimation. Therefore, some program or application should be developed to automatically calibrate the device so that the obtained data was more reliable and precise.
- An interesting purpose for future projects would be to design a real prototype useful in clinical environment, since nowadays the UR5 is not the most competitive robot to be used in hospitals although it has very fine properties.

12. Bibliography

- [1] Minimally Invasive Thoracic and Cardiac Surgery: Textbook and Atlas. (2013). *Pneumology*, 67(09), 512–512
- [2] Uclahealth.org. 2020. About Robotic Surgery: What Is Robotic Surgery? | UCLA Health. [online] Available at: <<https://www.uclahealth.org/robotic-surgery/what-is-robotic-surgery#whatis>> [Accessed 9 May 2020].
- [3] Lanfranco, A. R., Castellanos, A. E., Desai, J. P., & Meyers, W. C. (2004, January). Robotic Surgery: A Current Perspective. *Annals of Surgery*.
- [4] Shah, J., Vyas, A., & Vyas, D. (2015). The History of Robotics in Surgical Specialties. *American Journal of Robotic Surgery*, 1(1), 12–20.
- [5] Mancisidor, A., Zubizarreta, A., Cabanes, I., Bengoa, P., & Jung, J. H. (2018). New Trends in Medical and Service Robots. (M. Husty & M. Hofbaur, Eds.), *New Trends in Medical and Service Robots Design, Analysis and Control* (Vol. 48, pp. 117–130). Springer International Publishing.
- [6] John, H., & Wiklund, P. (2008). Robotic urology. *Robotic Urology* (pp. 1–267). Springer Berlin Heidelberg.
- [7] Uclahealth.org. 2020. About Robotic Surgery: What Is Robotic Surgery? | UCLA Health. [online] Available at: <<https://www.uclahealth.org/robotic-surgery/what-is-robotic-surgery#whatis>> [Accessed 9 May 2020].
- [8] Autorino, R., Kaouk, J. H., Stolzenburg, J. U., Gill, I. S., Motttrie, A., Tewari, A., & Cadeddu, J. A. (2013, February). Current status and future directions of robotic single-site surgery: A systematic review. *European Urology*.
- [9] Roterman-Konieczna, I. (Eds.). (2020). *Simulations in Medicine*. Berlin, Boston: De Gruyter.
- [10] Yokokohji Y, Yoshikawa T. Bilateral control of master-slave manipulators for ideal kinesthetic coupling – formulation and experiment. *IEEE Trans Robot Automat* 1994;10(5):605–20.
- [11] Iqbal Singh (2011) Robotics in urological surgery: Review of current status and maneuverability, and comparison of robot-assisted and traditional laparoscopy, *Computer Aided Surgery*, 16:1, 38-45,
- [12] Wei, F. C., & Mardini, S. (2009). *Flaps and Reconstructive Surgery*. Flaps and Reconstructive Surgery. Elsevier Inc.
- [13] Surgical robotics: Reviewing the past, analysing the present, imagining the future. Paula Gomes. 2011.
- [14] Ho, C., Tsakonas, E., Tran, K., Severn, M., Mierzwinski-Urban, M., Corcos, J., & Pautler, S. (2011). Robot-Assisted Surgery Compared with Open Surgery and Laparoscopic Surgery: Clinical Effectiveness and Economic Analyses. *Canadian Agency for Drugs and Technologies in Health* (pp. i–286). Retrieved from http://www.cadth.ca/media/pdf/H0496_Surgical_robotics_e.pdf
- [15] TransEnterix. Annual Report. Morrisville: Transenterix, Inc, 2018.
- [16] Samalavicius, N. E., Janusonis, V., Siauly, R., Jasėnas, M., Deduchovas, O., Venckus, R., ... Klimaviciute, G. (2020). Robotic surgery using Senhance® robotic platform: single center experience with first 100 cases. *Journal of Robotic Surgery*, 14(2), 371–376.
- [17] Smith-nephew.com. 2020. NAVIO Surgical System - Robotics-Assisted Knee Replacement | Smith & Nephew. [online] Available at: <<https://www.smith-nephew.com/professional/microsites/navio/>> [Accessed 20 May 2020].
- [18] Medtronic.com. 2020. Mazor X Robotic Guidance System - Technical Specifications. [online] Available at: <<https://www.medtronic.com/us-en/healthcare-professionals/products/neurological/spine-robotics/mazorx/technical-specifications.html>> [Accessed 20 May 2020].
- [19] Auris Health. 2020. Monarch Platform - Endoscopy Transformed - Auris Health. [online] Available at: <<https://www.aurishealth.com/monarch-platform>> [Accessed 21 May 2020].
- [20] Medrobotics.com. 2020. Flex® Robotic System Technology: How It Works | Medrobotics. [online] Available at: <<https://medrobotics.com/gateway/technology-int/>> [Accessed 21 May 2020].
- [21] Robsurgical.com. 2020. Rob Surgical. [online] Available at: <<https://www.robsurgical.com/bittrack/>> [Accessed

21 May 2020].

- [22] Coronato, A. (2018). ISO 13485: medical devices - quality management systems - requirements for regulatory purposes. In *Engineering High Quality Medical Software: Regulations, standards, methodologies and tools for certification* (pp. 51–67). Institution of Engineering and Technology.
- [23] Kyrkjebø, E., Johan Laastad, M., & Stavadahl, O. (2018). Feasibility of the UR5 Industrial Robot for Robotic Rehabilitation of the Upper Limbs After Stroke. In *IEEE International Conference on Intelligent Robots and Systems* (pp. 6124–6129). Institute of Electrical and Electronics Engineers Inc.
- [24] Blog.universal-robots.com. 2020. Cobots – A Helping Hand To The Healthcare Industry. [online] Available at: <<https://blog.universal-robots.com/cobots-conquering-the-healthcare-frontier>> [Accessed 2 June 2020].
- [25] CommonPlace Robotics (2019). *Mover4* Product Sheet. [online] Cpr-robots.com. Available at: https://cpr-robots.com/wp-content/uploads/2018/06/ProductSheet_Mover4_0510-06.pdf
- [26] Cpr-robots.com. 2020. Education | Commonplace Robotics. [online] Available at: <<https://cpr-robots.com/education>> [Accessed 1 June 2020].
- [27] <https://cpr-robots.com/>. 2020. Price List 2018. [online] Available at: <https://cpr-robots.com/wp-content/uploads/2018/06/Preisliste-CPR_EN.pdf> [Accessed 2 June 2020].
- [28] www.stevenengineering.com. 2017. Sawyer Technical Data Sheet. [online] Available at: <https://stevenengineering.com/Tech_Support/PDFs/35_SAWYER.pdf> [Accessed 2 June 2020].
- [29] Rethinkrobotics.com. 2020. Sawyer Collaborative Robots For Industrial Automation. [online] Available at: <<https://www.rethinkrobotics.com/Sawyer>> [Accessed 2 June 2020].
- [30] ROBOTIS. (2010). AX-12/ AX-12+/ AX 12+/ AX 12+/ AX-12A (ROBOTIS e-Manual v1.10.00). https://www.pishrobot.com/files/products/datasheets/dynamixel_ax-12a.pdf
- [31] Servo Motor SG-90 Basics, Pinout, Wire Description, Datasheet. (s. f.). Components101. Accessed 13 March 2021, de <https://components101.com/motors/servo-motor-basics-pinout-datasheet>
- [32] Farnell. (2015). *Arduino Uno*. <https://www.farnell.com/datasheets/1682209.pdf>
- [33] RobotShop. (2004). *Arduino Mega 2560* Datasheet. Everlight Electronics <http://eprints.polsri.ac.id/4598/8/File%20VIII%20%28Lampiran%29.pdf>
- [34] UM1724 User manual. STM32 Nucleo-64 boards (MB1136). (2020, July). STMicroelectronics. https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf
- [35] 3D Systems (2019). Touch User Guide. [online] 3dsystems.com. Available at: <https://www.3dsystems.com/sites/default/files/2017-12/3DSystems-Touch-UserGuide.pdf>
- [36] IEEE Spectrum: Technology, Engineering, and Science News. 2020. *Universal Robots UR3 Arm Is Small and Nimble, Helps To Build Copies Of Itself*. [online] Available at: <<https://spectrum.ieee.org/autaton/robotics/industrial-robots/universal-robots-ur3-robotic-arm>> [Accessed 2 June 2020].
- [37] VMA319. ARDUINO® COMPATIBLE ACTIVE BUZZER MODULE (2 PCS). (2020). Velleman. https://www.velleman.eu/downloads/29/vma319_a4v01.pdf
- [38] Piezo Buzzer MCKPT-G1340-3917. (2020, January). multcomp pro. <http://www.farnell.com/datasheets/2891582.pdf>
- [39] Zuo, B. (n.d.). Grove - Vibration Motor - Seeed Wiki. Seeed Studio. Retrieved April 2, 2021, from https://wiki.seeedstudio.com/Grove-Vibration_Motor/
- [40] Coin Vibration Motor (C1026B002F). (2009). Vibramotor. <https://www.mouser.es/datasheet/2/321/28821-Flat-Coin-Vibration-Motor-Documentation-369707.pdf>
- [41] B3F-1022 Tactile Switch. (2020). Omron Corporation.

https://omronfs.omron.com/en_US/ecb/products/pdf/en-b3f.pdf

[42] Seleccione su edición de *LabVIEW*. (n.d.). NI. Retrieved June 6, 2021, from <https://www.ni.com/es-es/shop/LabVIEW/select-edition.html>

[43] Pricing and Licensing. (n.d.). MATLAB & Simulink. Retrieved June 13, 2021, from <https://es.mathworks.com/pricing-licensing.html>

[44] FreeCAD: Your own 3D parametric modeler. (n.d.). FreeCAD. Retrieved June 13, 2021, from <https://www.freecadweb.org/>

[45] Robert McNeel & Associates. (n.d.). Features. [Www.Rhino3d.Com](http://www.Rhino3d.Com). Retrieved June 13, 2021, from <https://www.rhino3d.com/features/>

13. Appendixes

13.1. *Appendix 1: Transmission of the RPY angles from the pen to the servomotors (Arduino IDE)*

```
// digital pins 2 and 4 have pushbuttons attached
int pushButtonA = 2;
int pushButtonB = 4;

#include "src/RoboticsUB.h"
#include <Servo.h>

IMU imu;
Servo servol;
Servo servo2;
Servo servo3;

//int PIN_IMU_VCC = 4;
//int PIN_IMU_INT = 5;

float *rpw;           // Pointer to read RPW
char instruction = 0; // For incoming serial data

int Pin_R1 = A0;       // Analogic pin used by R1 (Servo1)
int Pin_R2 = A1;       // Analogic pin used by R2 (Servo2)
int Pin_R3 = A2;       // Analogic pin used by R3 (Servo3)

float R1 = 1.6;        // Resistance value R1
float R2 = 1.6;        // Resistance value R2 (Servo2)
float R3 = 3.3;        // Resistance value R3 (Servo3)

float motor_angle_X = 0; // Motor angle
float motor_angle_Y = 0;
float motor_angle_Z = 0;

void setup()
{
    Serial.begin(115200);

    imu.Install();
    servol.attach(9);
    servo2.attach(10);
    servo3.attach(11);

    pinMode(pushButtonA, INPUT);
    pinMode(pushButtonB, INPUT);
}

void loop()
{
    imu.ReadSensor();
    rpw = imu.GetRPY();

    // Angle range from 0 to 180 degrees
    if (rpw[2] <= 180 && rpw[2] >= 0)
    {
        motor_angle_X = rpw[0];
    }
}
```

```

}

servo2.write(motor_angle_X);

// Angle range from 0 to 180 degrees
if (rpw[1] <= 180 && rpw[1] >= 0)
{
    motor_angle_Y = rpw[1];
}

servo3.write(motor_angle_Y);

// Angle range from 0 to 180 degrees
if (rpw[2] <= 180 && rpw[2] >= 0)
{
    motor_angle_Z = rpw[2];
}

servo1.write(motor_angle_Z);

if (Serial.available() > 0)
{
    // read the incoming byte:
    instruction = Serial.read();

    switch (instruction)
    {
        case 'A':
            Serial.print("RPW: ");
            Serial.println(String(rpw[0], 4));
            Serial.println(String(rpw[1], 4));
            Serial.println(String(rpw[2], 4));

            break;

        default:
            break;
    }

    instruction = NULL;
}

////////// BUTTON ///////////
// read the input pin:
//int buttonStateA = digitalRead(pushButtonA);
//int buttonStateB = digitalRead(pushButtonB);
// print out the state of the button:
//Serial.println(buttonStateA);
//Serial.println(buttonStateB);
//delay(1); // delay in between reads for stability
//Serial.println(torque1);
}

```

13.2. Appendix 2: Transmission of the RPY angles from the pen to the servomotors (RoboDK)

```
import serial
import time
import math
import TKinter as tk
import numpy as np
from robolink import *
from RoboDK import *

# Variables definition
# TCP end-effector respect the Flange
X=0
Y=-60
Z=320

# Lets bring some time to the system to stablsh 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('TCP_EndoWrist')
pose_tcp=tcp_tool.Pose()

# 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(pose_tcp))

# -----
# Establish the connection on a specific port
arduino = serial.Serial("COM3", 115200, timeout=1)

# PROVES SENSE ARDUINO

condicio=0
iteration=0

def change_condition():
    global condicio
    if condicio == True:
        condicio = False
```



```

        else:
            condicio = True

# RPY page
l1 = 0
l2 = 0
l3 = 0
l4 = 0
l5 = 0
ChangeButton2 = 0
page = 0
ChangeButton = 0
page2 = 0

def RPY_page():
    global l1
    global l2
    global l3
    global l4
    global ChangeButton
    global page2
    global condicio
    page2=tk.Tk()
    page2.title("RPY angles and Torque")
    title=tk.Label(page2, text="The output is: ").grid(row=1, column=1,padx=5,
pady=5)
    l1t = tk.Label(page2, text = "R (Roll)= ").grid(row=2, column=1,padx=5,
pady=5)
    l1 = tk.Label(page2, text = " ")
    l1.grid(row=2, column=2,padx=5, pady=5)
    l2t = tk.Label(page2, text = "P (Pitch)= ").grid(row=3, column=1,padx=5,
pady=5)
    l2 = tk.Label(page2, text = " ")
    l2.grid(row=3, column=2,padx=5, pady=5)
    l3t = tk.Label(page2, text = "Y (Yaw)= ").grid(row=4, column=1,padx=5,
pady=5)
    l3 = tk.Label(page2, text = " ")
    l3.grid(row=4, column=2,padx=5, pady=5)
    StartPage.quit()
    condicio = True

# Close window
def close():
    StartPage.destroy()
    quit(0)

# Start Page configuration
StartPage=tk.Tk()
StartPage.title("Start Page")
text=tk.Label(StartPage, text="Select the output's convention: ", height=5)
text.pack()
RPY_button=tk.Button(StartPage,text="RPY Orientation", fg="red", command =
RPY_page)
RPY_button.pack(side="left")
StartPage.mainloop()
imagen = PhotoImage(file="UR5e.gif")
Label(StartPage, image=imagen, bd=30).pack()
StartPage.protocol("WM_DELETE_WINDOW", close) # Delete the window when we close
it, therefore it won't keep running

try:
    while True:

```

```

if condicio==True:
    # Requesting data to Ardino (command A)
    arduino.write(b'A')

    # RPY
    roll_str = arduino.readline().strip()
    pitch_str = arduino.readline().strip()
    yaw_str = arduino.readline().strip()

    # As we can not try it with Arduino, we define the RPY angles and
torque
    # roll_str = 1
    # pitch_str = 2
    # yaw_str = 3

    # Convert variable values from string to float
    roll = float(roll_str)
    pitch = float(pitch_str)
    yaw = float(yaw_str)

    # Convert from degrees to radians R,P,Y angles
    R = math.radians(roll)
    P = math.radians(pitch)
    W = math.radians(yaw)

    # Calculate the POSE matrix (UR)
    pose_matrix_rpy = transl([X, Y, Z])*rotx(pi)*rotx(-R)*roty(-P)*rotz(-
W)

    tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix_rpy)
    print("The POSE matrix with RPY: "+ repr(pose_matrix_rpy))
    l1.config(text = np.around(R, decimals = 2))
    l2.config(text = np.around(P, decimals = 2))
    l3.config(text = np.around(W, decimals = 2))
    page2.update()

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

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

```

13.3. *Appendix 3: Acquisition and visualization of the RPY angles from the haptic pen (Arduino IDE)*

```

/*****
Reading IMU Sensor
*****/

#include "src/RoboticsUB.h"
#include <Servo.h>

IMU imu;
Servo servo1;
Servo servo2;
Servo servo3;

int PIN_IMU_VCC = 4;
int PIN_IMU_INT = 5;
float *rpw;           // Pointer to read RPW
float *q;             // Pointer to quaternion
char instruction = 0; // For incoming serial data
float torque = 0;

int Pin_R1 = A0;      // Analogic pin used by R1 (Servo1)
int Pin_R2 = A1;      // Analogic pin used by R2 (Servo2)
int Pin_R3 = A2;      // Analogic pin used by R3 (Servo3)

float R = 1.6;        // Resistance value R1

const unsigned long period_milis = 200; //Time for torque output
unsigned long current_milis = 0;
unsigned long previous_milis = 0;

float motor_angle_X = 0; // Motor angle
float motor_angle_Y = 0; // Motor angle
float motor_angle_Z = 0; // Motor angle

void setup()
{
    Serial.begin(115200);
    imu.Install();

    // Power the IMU from pin to reset
    pinMode(PIN_IMU_VCC, OUTPUT);
    digitalWrite(PIN_IMU_VCC, LOW);
    delay(100);
    digitalWrite(PIN_IMU_VCC, HIGH);
    delay(100);

    servo1.attach(9);
    servo2.attach(10); // poso aquests perquè són PWM, com D9
    servo3.attach(11);
}

void loop()
{
    current_milis = millis();
    if (digitalRead(PIN_IMU_INT) == HIGH) {
        imu.ReadSensor();
        rpw = imu.GetRPY();
    }
}
```

```

}

// Angle range from 0 to 180 degrees
if (rpw[0] <= 180 && rpw[0] >= 0)
{
    motor_angle_X = rpw[0];
}

servo2.write(motor_angle_X);

// Angle range from 0 to 180 degrees
if (rpw[1] <= 180 && rpw[1] >= 0)
{
    motor_angle_Y = rpw[1];
}

servo3.write(motor_angle_Y);

// Angle range from 0 to 180 degrees
if (rpw[2] <= 180 && rpw[2] >= 0)
{
    motor_angle_Z = rpw[2];
}

servo1.write(motor_angle_Z);

if (Serial.available() > 0)
{
    instruction = Serial.read();

    switch (instruction)
    {
        case 'A':
            //Serial.print("RPW angles are: ");
            Serial.println(String(rpw[0], 2));
            Serial.println(String(rpw[1], 2));
            Serial.println(String(rpw[2], 2));
            delay(20);

            break;

        case 'B':

            Serial.println(String(q[0], 4));
            Serial.println(String(q[1], 4));
            Serial.println(String(q[2], 4));
            Serial.println(String(q[3], 4));
            Serial.println(String(torque, 4));

            break;

        default:
            break;
    }

    instruction = NULL;
}
}

```

13.4. Appendix 4: Acquisition and visualization of the RPY angles from the haptic pen (RoboDK)

```
import serial
import time
import math
import TKinter as tk
import numpy as np
from robolink import *
from RoboDK import *

# Variables definition
# TCP end-effector respect the Flange
X=0
Y=-60
Z=320

# 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('TCP_EndoWrist')
pose_tcp=tcp_tool.Pose()

# 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(pose_tcp))

# -----
# Establish the connection on a specific port
arduino = serial.Serial("COM3", 115200, timeout=1)

# PROVES SENSE ARDUINO

condicio=0
iteration=0

def change_condition():
    global condicio
    if condicio == True:
        condicio = False
```

```

        else:
            condicio = True

# RPY page
l1 = 0
l2 = 0
l3 = 0
l4 = 0
l5 = 0
page = 0
page2 = 0

def RPY_page():
    global l1
    global l2
    global l3
    global l4
    global l5
    global page2
    global condicio
    page2=tk.Tk()
    page2.title("RPY angles")
    title=tk.Label(page2, text="The output is: ").grid(row=1, column=1,padx=5,
pady=5)
    l1t =tk.Label(page2, text = "R (Roll) = ").grid(row=2, column=1,padx=5,
pady=5)
    l1 = tk.Label(page2, text = " ")
    l1.grid(row=2, column=2,padx=5, pady=5)
    l2t = tk.Label(page2, text = "P (Pitch) = ").grid(row=3, column=1,padx=5,
pady=5)
    l2 = tk.Label(page2, text = " ")
    l2.grid(row=3, column=2,padx=5, pady=5)
    l3t = tk.Label(page2, text = "Y (Yaw) = ").grid(row=4, column=1,padx=5,
pady=5)
    StartPage.quit()

    condicio = True

# Close window
def close():
    StartPage.destroy()
    quit(0)

# Start Page configuration
StartPage=tk.Tk()
StartPage.title("Start Page")
text=tk.Label(StartPage, text="Push the button to see the RPY angles: ",
height=5)
text.pack()
RPY_button=tk.Button(StartPage,text="RPY Orientation", fg="red", command =
RPY_page)
RPY_button.pack(side="bottom")
StartPage.mainloop()

StartPage.protocol("WM_DELETE_WINDOW", close) # Delete the window when we close
it, therefore it won't keep running

try:
    while True:

        if condicio==True:

```

```

# Requesting data to Ardino (command A)
arduino.write(b'A')

# RPY
roll_str = arduino.readline().strip()
pitch_str = arduino.readline().strip()
yaw_str = arduino.readline().strip()

# As we can not try it with Ardino, we define the RPY angles and
torque

# Convert variable values from string to float
roll = float(roll_str)
pitch = float(pitch_str)
yaw = float(yaw_str)

# Convert from degrees to radians R,P,Y angles
R = math.radians(roll)
P = math.radians(pitch)
W = math.radians(yaw)

# Calculate the POSE matrix (UR)
pose_matrix_rpy = transl([X, Y, Z])*rotx(pi)*rotx(-R)*roty(-P)*rotz(-
W)

tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix_rpy)
print("The POSE matrix with RPY: "+ repr(pose_matrix_rpy))
l1.config(text = np.around(R, decimals = 2))
l2.config(text = np.around(P, decimals = 2))
l3.config(text = np.around(W, decimals = 2))
page2.update()

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

# -----
# Disconnect Ardino
# -----
print("Disconnecting Ardino...")
arduino.close()

```

13.5. *Appendix 5: Transmission of the RPY angles from the PC to the servomotors (Arduino IDE)*

```
// Aquest codi controla la placa hardware de l'ur5e.
// Rebrà la orientació i posició de la IMU (que es troba al pen) mitjançant wifi,
// i farà girar els servos
// D'altra banda, llegirà el torque, que enviarà al haptic pen.

#include "src/RoboticsUB.h"
#include <Servo.h>

Servo servo1;
Servo servo2;
Servo servo3;

int Pin_R1 = A0;      // Analogic pin used by R1 (Servo1)
int Pin_R2 = A1;      // Analogic pin used by R2 (Servo2)
int Pin_R3 = A2;      // Analogic pin used by R3 (Servo3)

float R = 1.6;        // Resistance value
float torque1 = 0;    // Indicated as current (ampere)
float torque_int1 = 0;
float torque2 = 0;    // Indicated as current (ampere)
float torque_int2 = 0;
float torque3 = 0;    // Indicated as current (ampere)
float torque_int3 = 0;

const unsigned long period_milis = 200; //Time for torque output
unsigned long current_milis1 = 0;
unsigned long previous_milis1 = 0;

unsigned long current_milis2 = 0;
unsigned long previous_milis2 = 0;

unsigned long current_milis3 = 0;
unsigned long previous_milis3 = 0;

//unsigned long current_milis4 = 0;
//unsigned long previous_milis4 = 0;

float motor_angle_X = 0; // Motor angle
float motor_angle_Y = 0; // Motor angle
float motor_angle_Z = 0; // Motor angle

bool condicio = HIGH;

float rpw_1; // valor absolut, no incremental
float rpw_2;
float rpw_3;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    //Serial.setTimeout(1);
    servo1.attach(9);
    servo2.attach(10);
    servo3.attach(11);
}

void loop() {
    // put your main code here, to run repeatedly:
```



```

if (condicio == HIGH){

    while (!Serial.available());

    rpw_1 = Serial.readString().toInt();
    rpw_2 = Serial.readString().toInt();
    rpw_3 = Serial.readString().toInt();

}

current_milis1 = millis();
current_milis2 = millis();
current_milis3 = millis();

if (rpw_1 <= 180 && rpw_1 >= 0){
    motor_angle_X = rpw_1;
}

servo1.write(motor_angle_X);

if (current_milis1-previous_milis1>=period_milis){
    torque1=torque_int1;
    torque_int1=0;
    previous_milis1=current_milis1;
}

else{
    torque_int1 += analogRead(Pin_R1) * (3.3 / 1023.0) / R;
}

if (rpw_2 <=180 && rpw_2 >= 0)
{
    motor_angle_Y = rpw_2;
}
servo2.write(motor_angle_Y);

if (current_milis2-previous_milis2>=period_milis){
    torque2=torque_int2;
    torque_int2=0;
    previous_milis2=current_milis2;
}

else{
    torque_int2 += analogRead(Pin_R2) * (3.3 / 1023.0) / R;
}

if (rpw_3 <= 180 && rpw_3 >= 0)
{
    motor_angle_Z = rpw_3;
}
servo3.write(motor_angle_Z);

if (current_milis3-previous_milis3>=period_milis){
    torque3=torque_int3;
    torque_int3=0;
    previous_milis3=current_milis3;
}

else{
    torque_int3 += analogRead(Pin_R3) * (3.3 / 1023.0) / R;
}

```

```
    }  
  
    if (Serial.available() > 0) {  
        //Serial.print("The torque values are: ");  
        Serial.println(String(torque1, 2));  
        Serial.println(String(torque2, 2));  
        Serial.println(String(torque3, 2));  
    }  
}
```

13.6. Appendix 6: Transmission of the RPY angles from the PC to the servomotors (RoboDK)

```
"""
*****
TFG juls
*****
"""
import serial
import time
import math
import TKinter as tk
import numpy as np
from robolink import *
from RoboDK import *

# Variables definition
# TCP end-effector respect the Flange
X=0
Y=-60
Z=320

# Lets bring some time to the system to stablsh 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('TCP_EndoWrist')
pose_tcp=tcp_tool.Pose()

# 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(pose_tcp))

# -----
# Establish the connection on a specific port
arduino = serial.Serial("COM3", 115200, timeout=1)
```

```

def write_read_R(rpw_1):
    arduino.write(bytes(rpw_1, 'utf-8'))
    time.sleep(0.05)
    roll = arduino.readline()
    return roll

def write_read_P(rpw_2):
    arduino.write(bytes(rpw_2, 'utf-8'))
    time.sleep(0.05)
    pitch = arduino.readline()
    return pitch

def write_read_Y(rpw_3):
    arduino.write(bytes(rpw_3, 'utf-8'))
    time.sleep(0.05)
    yaw = arduino.readline()
    return yaw

while True:
    roll_ = str(input("Enter a roll angle: "))
    pitch_ = str(input("Enter a pitch angle: "))
    yaw_ = str(input("Enter a yaw angle: "))

    roll_value = write_read_R(roll_)
    pitch_value = write_read_P(pitch_)
    yaw_value = write_read_Y(yaw_)

    torque1 = arduino.readline().strip()
    torque2 = arduino.readline().strip()
    torque3 = arduino.readline().strip()

    print(roll_value, pitch_value, yaw_value)
    print()
    print(torque1, torque2, torque3)

    # Convert from degrees to radians R,P,Y angles
    R = math.radians(int(roll_))
    P = math.radians(int(pitch_))
    W = math.radians(int(yaw_))

    # Calculate the POSE matrix (UR)
    pose_matrix_rpy = transl([X, Y, Z])*rotx(pi)*rotx(-R)*roty(-P)*rotz(-W)
    tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix_rpy)
    print("The POSE matrix with RPY: "+ repr(pose_matrix_rpy))

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

```

13.7. Appendix 7: Transmission of the RPY angles from the PC to the servomotors - sliders (RoboDK)

```
import serial
import time
import math
import TKinter as tk
import numpy as np
from robolink import *
from RoboDK import *

# Set up default parameters
# Variables definition
# TCP end-effector respect the Flang
ROLL=90
PITCH=90
YAW=90

# Main program
def RunProgram():
    # Use default global variables
    global ROLL
    global PITCH
    global YAW

    ROLL=math.radians(entry_roll.get())
    PITCH=math.radians(entry_pitch.get())
    YAW=math.radians(entry_yaw.get())

    #Any interaction with RoboDK must be done through RDK:
    RDK = Robolink()

    # get the home target and the welding targets:
    home = RDK.Item('Home')
    target = RDK.Item('Target 1')

    # get the robot as an item:
    robot = RDK.Item('', ITEM_TYPE_ROBOT)

    # get the pose of the reference target (4x4 matrix representing position and
    orientation):
    poseref = target.Pose()

    # move the robot to home, then to an approach position
    robot.MoveJ(home)
    robot.MoveJ(transl(0,0,APPROACH)*poseref)

# Use TKinter to display GUI menus
from TKinter import *

# Generate the main window
root = tk.Tk()
root.title("Program settings")

# Use variables linked to the global variables

entry_roll = IntVar()
entry_roll.set(ROLL)

entry_pitch = IntVar()
```

```

entry_pitch.set(PITCH)

entry_yaw = IntVar()
entry_yaw.set(YAW)

# Define a label and entry text for the different parameters
Label(root, text="Roll (deg):").pack()
entry_roll=Scale(root, from_=0, to=90, orient=HORIZONTAL).pack()
#entry_roll.pack()
Label(root, text="Pitch (deg):").pack()
entry_roll=Scale(root, from_=0, to=90, orient=HORIZONTAL).pack()
#entry_pitch.pack()
Label(root, text="Yaw (deg):").pack()
entry_yaw=Scale(root, from_=0, to=90, orient=HORIZONTAL).pack()
#entry_yaw.pack()

def Execute():
    # Run the main program once all the global variables have been set
    RunProgram()

Button(root, text='Simulate', command=Execute).pack()
Label(root, text="").pack() # Separador

# Important to display the graphical user interface
root.mainloop()

```

13.8. *Appendix 8: Advancement of the TCP after pressing the pushbutton (Arduino IDE)*

```
#include "src/RoboticsUB.h"
#include <Servo.h>

IMU imu;
Servo servo1;
Servo servo2;
Servo servo3;

// digital pins 3 and 2 have pushbuttons attached
const int pushButtonA = 2;
const int pushButtonB = 3;

int PIN_IMU_VCC = 4;
int PIN_IMU_INT = 5;
float *rpw;          // Pointer to read RPW
float *q;             // Pointer to quaternion
char instruction = 0; // For incoming serial data
float torque = 0;

int Pin_R1 = A0;      // Analogic pin used by R1 (Servo1)
int Pin_R2 = A1;      // Analogic pin used by R2 (Servo2)
int Pin_R3 = A2;      // Analogic pin used by R3 (Servo3)

float R = 1.6;        // Resistance value R1

const unsigned long period_milis = 200; //Time for torque output
unsigned long current_milis = 0;
unsigned long previous_milis = 0;

float motor_angle_X = 0; // Motor angle
float motor_angle_Y = 0; // Motor angle
float motor_angle_Z = 0; // Motor angle

void setup()
{
    Serial.begin(115200);
    imu.Install();

    // Power the IMU from pin to reset
    pinMode(PIN_IMU_VCC, OUTPUT);
    digitalWrite(PIN_IMU_VCC, LOW);
    delay(100);
    digitalWrite(PIN_IMU_VCC, HIGH);
    delay(100);

    servo1.attach(9);
    servo2.attach(10); // poso aquests perquè són PWM, com D9
    servo3.attach(11);

    pinMode(pushButtonA, INPUT);
    pinMode(pushButtonB, INPUT);
}

void loop()
{
    current_milis = millis();
    if (digitalRead(PIN_IMU_INT) == HIGH) {
```

```

    imu.ReadSensor();
    rpw = imu.GetRPY();
}

// Angle range from 0 to 180 degrees
if (rpw[0] <= 180 && rpw[0] >= 0)
{
    motor_angle_X = rpw[0];
}

servo2.write(motor_angle_X);

// Angle range from 0 to 180 degrees
if (rpw[1] <= 180 && rpw[1] >= 0)
{
    motor_angle_Y = rpw[1];
}

servo3.write(motor_angle_Y);

// Angle range from 0 to 180 degrees
if (rpw[2] <= 180 && rpw[2] >= 0)
{
    motor_angle_Z = rpw[2];
}

servo1.write(motor_angle_Z);

int buttonStateA = digitalRead(pushButtonA);
int buttonStateB = digitalRead(pushButtonB);

if (Serial.available() > 0)
{
    instruction = Serial.read();

    switch (instruction)
    {
    case 'A':
        //Serial.print("RPW angles are:  ");
        Serial.println(String(rpw[0], 4));
        Serial.println(String(rpw[1], 4));
        Serial.println(String(rpw[2], 4));
        Serial.println(String(buttonStateA, 4));
        Serial.println(String(buttonStateB, 4));
        delay(20);

        break;

    case 'B':

        Serial.println(String(q[0], 4));
        Serial.println(String(q[1], 4));
        Serial.println(String(q[2], 4));
        Serial.println(String(q[3], 4));
        Serial.println(String(torque, 4));

        break;

    default:
        break;
    }
}

```



```

    }

    instruction = NULL;

}

////////// BUTTON //////////
// read the input pin:
//int buttonStateA = digitalRead(pushButtonA);
//int buttonStateB = digitalRead(pushButtonB);
// print out the state of the button:
//Serial.print("Botón Inferior D2: ");
//Serial.println(buttonStateA);
//Serial.print("Botón Superior D3: ");
//Serial.println(buttonStateB);
//delay(20); // delay in between reads for stability
// 1: sense apretar
// 0: quan apretes
}

```

13.9. Appendix 9: Advancement of the TCP after pressing the pushbutton (RoboDK)

```
import serial
import time
import math
import TKinter as tk
import numpy as np
from robolink import *
from RoboDK import *

# Variables definition
# TCP end-effector respect the Flange
X=0
Y=-60
Z=320

# 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('TCP_EndoWrist')
pose_tcp=tcp_tool.Pose()

# 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(pose_tcp))

# -----
# Establish the connection on a specific port
arduino = serial.Serial("COM3", 115200, timeout=1)

# PROVES SENSE ARDUINO

condicio=0
iteration=0

def change_condition():
    global condicio
    if condicio == True:
        condicio = False
    else:
```

```

        condicio = True

# RPY page
l1 = 0
l2 = 0
l3 = 0
l4 = 0
l5 = 0
page = 0
page2 = 0

def RPY_page():
    global l1
    global l2
    global l3
    global l4
    global l5
    global page2
    global condicio
    page2=tk.Tk()
    page2.title("RPY angles")
    title=tk.Label(page2, text="The output is: ").grid(row=1, column=1,padx=5,
pady=5)
    l1t = tk.Label(page2, text = "R (Roll) = ").grid(row=2, column=1,padx=5,
pady=5)
    l1 = tk.Label(page2, text = " ")
    l1.grid(row=2, column=2,padx=5, pady=5)
    l2t = tk.Label(page2, text = "P (Pitch) = ").grid(row=3, column=1,padx=5,
pady=5)
    l2 = tk.Label(page2, text = " ")
    l2.grid(row=3, column=2,padx=5, pady=5)
    l3t = tk.Label(page2, text = "Y (Yaw) = ").grid(row=4, column=1,padx=5,
pady=5)
    l3 = tk.Label(page2, text = " ")
    l3.grid(row=4, column=2,padx=5, pady=5)
    l4t = tk.Label(page2, text = "Button 1 = ").grid(row=5, column=1,padx=5,
pady=5)
    l4 = tk.Label(page2, text = " ")
    l4.grid(row=5, column=2,padx=5, pady=5)
    l5t = tk.Label(page2, text = "Button 2 = ").grid(row=6, column=1,padx=5,
pady=5)
    l5 = tk.Label(page2, text = " ")
    l5.grid(row=6, column=2,padx=5, pady=5)
    StartPage.quit()

        condicio = True

# Close window
def close():
    StartPage.destroy()
    quit(0)

# Start Page configuration
StartPage=tk.Tk()
StartPage.title("Start Page")
text=tk.Label(StartPage, text="Push the button to see the RPY angles: ",
height=5)
text.pack()
RPY_button=tk.Button(StartPage,text="RPY Orientation", fg="red", command =
RPY_page)
RPY_button.pack(side="bottom")

```

```

StartPage.mainloop()

StartPage.protocol("WM_DELETE_WINDOW", close) # Delete the window when we close
it, therefore it won't keep running

try:
    while True:

        if condicio==True:
            # Requesting data to Ardino (command A)
            arduino.write(b'A')

            # RPY
            roll_str = arduino.readline().strip()
            pitch_str = arduino.readline().strip()
            yaw_str = arduino.readline().strip()
            b1_str = arduino.readline().strip()
            b2_str = arduino.readline().strip()

            # As we can not try it with Arduino, we define the RPY angles and
torque

            # Convert variable values from string to float
            roll = float(roll_str)
            pitch = float(pitch_str)
            yaw = float(yaw_str)
            b1 = float(b1_str)
            b2 = float(b2_str)

            # Convert from degrees to radians R,P,Y angles
            R = math.radians(roll)
            P = math.radians(pitch)
            W = math.radians(yaw)

            # Calculate the POSE matrix (UR)
            pose_matrix_rpy = transl([X, Y, Z])*rotx(pi)*rotx(-R)*roty(-P)*rotz(-
W)

            tcp_tool_pose = tcp_tool.setPoseTool(pose_matrix_rpy)
            print("The POSE matrix with RPY: "+ repr(pose_matrix_rpy))
            l1.config(text = np.around(R, decimals = 2))
            l2.config(text = np.around(P, decimals = 2))
            l3.config(text = np.around(W, decimals = 2))
            l4.config(text = np.around(b1, decimals = 2))
            l5.config(text = np.around(b2, decimals = 2))
            page2.update()

            ## BUTTONS -----

            if b1 == 0:
                # Define the motion of the TCP
                RDK.AddFrame("TCP_Moving")
                tcp_motion = RDK.Item('TCP_Moving')

                # Next position

                tcp_motion_pose = tcp_motion*transl(10,0,0) # translation in x
axis

                next_pose.setPose(tcp_motion_pose)
                next_tcp_pose=tcp_motion.Pose()

```

```

        print("POSE of the next POSE after the movement:
"+repr(next_tcp_pose))

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

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

```

13.10. *Appendix 10: Opening of the needle jaws after pressing the pushbutton (Arduino IDE)*

```
/*
*****
TFG
*****
*/

// digital pins 2 and 3 have pushbuttons attached
int pushButtonA = 2;
int pushButtonB = 4;

#include "src/RoboticsUB.h"
#include <Servo.h>

// X-axis (roll) is translated as the rotation of the 2nd servomotor
// Therefore, we need servos 1 and 3

IMU imu;
Servo servo1;
Servo servo3;

//int PIN_IMU_VCC = 4;
//int PIN_IMU_INT = 5;

float *rpw;          // Pointer to read RPW
char instruction = 0; // For incoming serial data

int Pin_R2 = A1;      // Analogic pin used by R2 (Servo2)
int Pin_R3 = A2;      // Analogic pin used by R3 (Servo3)

float R1 = 1.6;        // Resistance value R2 (Servo2)
float R3 = 3.3;        // Resistance value R3 (Servo3)

float motor_angle_X = 0; // Motor angle
float motor_angle_Y = 0;
float motor_angle_Z = 0;

void setup()
{
    Serial.begin(115200);

    imu.Install();
    servo1.attach(9);
    servo3.attach(11);

    pinMode(pushButtonA, INPUT);
    pinMode(pushButtonB, INPUT);
}

void loop()
{
    imu.ReadSensor();
    rpw = imu.GetRPY();
    int buttonStateA = digitalRead(pushButtonA);
    int buttonStateB = digitalRead(pushButtonB);

    if (buttonStateB == 0)
    {
```

```

    // Open the jaw 90 degrees
    motor_angle_Y = 90; // we open a needle
    motor_angle_Z = 0; // but fix the other
}
servo3.write(motor_angle_Y);
servo1.write(motor_angle_Z);

if (Serial.available() > 0)
{
    Serial.print("Output data: ");
    Serial.println(String(motor_angle_Y, 2));
    Serial.println(String(motor_angle_Z, 2));
    Serial.println(String(buttonStateB, 2));
    delay(1);

    // PROBLEMA? ARA ES MOUEN TOTS IGUAL.

    break;

default:
    break;
}

}

}

```