

UNIVERSITAT_{DE} BARCELONA

Final Degree Project

Biomedical Engineering Degree

Integration of Haptic Feedback in Robotic Surgery: Enhancing the Performance of the UR5e Robotic Arm

Barcelona, 5 de juny de 2024

Author: Jan Sampol Ramirez

Tutor: Dr. Manel Puig i Vidal

Abstract

In recent years, there has been a remarkable technological leap that has allowed the progressive integration of cutting-edge innovations in the medical field. In this context, robotic surgery has emerged as a fundamental and pioneering tool that fuses computer science, electronics, and medicine. Its major expansion has been significantly driven by the Da Vinci Surgical System. Although it has been positively received by the medical community, it has certain limitations that prevent it from reaching its full potential. Among these limitations, the lack of haptic feedback and the high acquisition cost stand out.

In the light of these considerations, the project has been initiated with the objective of developing new solutions to the problems and limitations currently encountered by this technology. To this end, the study design adopts a comprehensive approach to tackle the identified shortcomings, structured into two distinct phases. The first stage involves the development of software and hardware to enhance the performance and functionality of the surgical system, which comprises a UR5e robotic arm and a series of surgeon-operated controllers. In the subsequent stage, haptic technology is integrated into the robotic arm, followed by comprehensive testing to verify its functionality.

The results indicate the development of an economical system with an acceptable level of robustness and fluid communication, allowing for comfortable and intuitive operation by the surgeon. In addition, it has been possible to transmit with precision and in real time the force exerted by the surgical tool on the tissue to the surgeon, via a vibratory actuator and a graphic interface that offers a visual and audible representation of the force exerted. These two implementations enhance the surgeon's performance and increase the safety of the procedure.

Keywords: Robotic surgery, Haptic feedback, UR5e, EndoWrist, Da Vinci Surgical System, Low-cost surgical system

Acknowledgements

ENGLISH:

I would first like to express my deepest gratitude to my tutor, Dr. Manel Puig i Vidal, for his support and guidance throughout the development of this project. His dedication and involvement have enabled me to explore a previously unfamiliar subject in great depth, sparking a newfound passion and interest in it. Without his encouragement and expertise, this project would not have been possible.

On the other hand, I would like to thank Aitor, an Electronic Engineering student, for the help he has offered me in designing the boards and choosing the appropriate electronic components, as well as making the time spent in the lab much more enjoyable.

On a personal note, I would like to make a special mention of my friend Jordi Penalva. I greatly appreciate his unwavering support, companionship, and assistance over the years. Without his help, this journey would have been significantly more challenging.

Likewise, I don't want to forget my colleagues and friends, who have provided me with moments of joy and relaxation amidst the long hours of intense work.

Last but not least, my heartfelt thanks go to my family. To my parents, Miriam and Toni, and my sister Laura, your unwavering support and encouragement have been crucial throughout this journey. Your unwavering belief in me and constant presence have empowered me to overcome every obstacle and have provided me with the strength to persevere. I am deeply grateful for everything you have done.

Thank you all.

CATALÀ:

En primer lloc, voldria expressar el meu més profund agraïment al meu tutor, Dr. Manel Puig i Vidal, pel seu suport i la seva guia durant el desenvolupament d'aquest projecte. La seva dedicació i implicació m'han permès explorar en profunditat un tema fins ara desconegut per a mi, despertant una nova passió i interès en aquest. Sense el seu ànim i experiència, aquest projecte no hauria estat possible.

D'altra banda, vull agrair a l'Aitor, estudiant d'Enginyeria Electrònica, per l'ajuda que m'ha ofert en el disseny de les plaques i l'elecció dels components electrònics adequats, així com per fer que el temps passat al laboratori fos molt més agradable.

En l'àmbit personal, m'agradaria fer una menció especial al meu amic Jordi Penalva. Valoro enormement el seu suport constant, la seva companyia i l'ajuda que m'ha brindat durant tots els anys de la carrera. Sense la seva ajuda, aquesta experiència hauria estat significativament més difícil.

Així mateix, no vull oblidar els meus companys i amics, els quals m'han proporcionat moments d'alegria i relax entre tantes hores de treball intens.

Finalment, però no menys important, vull expressar el meu més sincer agraïment a tota la meva família. Als meus pares, Miriam i Toni, i a la meva germana Laura, el vostre suport incondicional i ajuda constant han estat crucials al llarg d'aquest camí ple de canvis i noves experiències. La vostra creença en mi i la vostra presència m'han donat la força per superar cada obstacle i m'han proporcionat la determinació per seguir endavant. Estic profundament agraït per tot el que heu fet.

Gràcies a tots.

Table of Contents

Abstract	i
Acknowledgements	iii
Glossary	xi
1 Introduction	1
1.1 Robotic surgery	1
1.2 Laparoscopic surgical robots	1
1.3 Benefits and complications of robotic surgery	2
1.4 Economic aspects of robotic surgery	2
1.5 Minimal invasive surgery procedure	3
1.6 Objectives	3
1.7 Limitations of the project	4
1.8 Characteristics of the project development	4
2 Background	5
2.1 First surgical robots	5
2.2 Widespread surgical robots	5
2.2.1 AESOP	6
2.2.2 Endoassist	6
2.2.3 ZEUS	6
2.2.4 Da Vinci Surgical System [®]	7
2.3 Endowrist instrument	9
2.4 Haptic technology	9
2.5 Project context	10
3 Market analysis	11
3.1 Market evolution	11
3.2 Current market	12
3.3 Surgical robotic systems	13
3.4 Patents	16
3.5 Future prospects	17
4 Conceptual engineering	19
4.1 Software	20
4.1.1 Programming software	20
4.1.2 PCB designing Software	22
4.2 Hardware	22
4.2.1 Robotic arm	22
4.2.1.1 UR5e	23

4.2.1.2	Kuka 210 robotic arm	23
4.2.1.3	SWIFTITM CRB 1300 robotic arm	24
4.2.2	Quirurgic Gripper	24
4.2.2.1	Virtual Incision by Vicarious Surgical	24
4.2.2.2	Homis Surgical System by Memic	25
4.2.2.3	EndoWrist tool by Intuitive Surgical	25
4.2.3	Servomotor	26
4.2.4	IMU sensor	26
4.2.5	Microprocessor	26
4.2.6	Vibration motor	27
4.3	Testing material	27
4.4	Final election	27
5	Detail engineering	29
5.1	Hardware development	29
5.1.1	Master's PCB	30
5.1.1.1	Position and Orientation reading	30
5.1.1.2	Pen User Interface design	31
5.1.1.3	Forearm Master design	31
5.1.2	EndoWrist Motion Panel PCB	32
5.1.2.1	Torque reading technology	32
5.1.2.2	EndoWrist Motion Panel board design	33
5.2	Assembling system	34
5.2.1	EndoWrist and Servomotors coupling	34
5.2.2	Masters subjection system	35
5.3	Software developement	36
5.3.1	Arduino code	36
5.3.1.1	Pen User Interface Code	36
5.3.1.2	Forearm Master Code	37
5.3.1.3	EndoWrist Motion Panel Code	37
5.3.1.4	Control and Supervision Module Code	38
5.3.2	Python code	38
5.3.2.1	TKinter root	39
5.3.2.2	Suture process	40
5.4	Haptic feedback system implementation	42
6	Experimental validation	43
6.1	Haptic feedback system performance	43
6.2	Robot maneuverability	46
7	Execution schedule	47
7.1	WBS (Work Breakdown Structure)	47
7.2	WBS Dictionary	47
7.3	PERT (Program Evaluation and Review Technique)	51
7.4	GANTT diagram	52
8	Thecnical viability	53
8.1	Threats	53
8.2	Weakness	53
8.3	Opportunities	54
8.4	Strengths	54

9	Economical viability	55
9.1	Hardware and Software Analysis	55
9.2	Conclusions on System Viability	56
10	Legal aspects	57
10.1	Legal protection of robotic surgery	57
10.2	Regulations and institutions	58
11	Conclusions	59
11.1	Objectives and results	59
11.2	Future work	60
	Bibliography	61
	Appendices	69
.1	Main Python code	69
.2	TKinter Python code	77
.3	Pen User Interface Arduino code	83
.4	Forearm Master Arduino code	87
.5	EndoWrist Motion Panel Arduino code	91
.6	Control and Supervision Module Arduino code	98
.7	Pen User Interface PCB design	101
.8	EndoWrist Motion Panel PCB design	102

List of Tables

4.1	Comparison between programs	21
4.2	Comparison between PCB designing programs	22
4.3	Comparison between possible Servomotors	26
4.4	Comparison between possible IMUs	26
4.5	Comparison between possible Microprocessors	26
4.6	Comparison between possible Vibration Motors	27
4.7	Final selection of the components	28
5.1	Pen User Interface PCB connections	31
5.2	EndoWrist Motion Panel PCB connections	33
7.1	"Objectives Definition" entry in the WBS Dictionary	48
7.2	"Background and market analysis" entry in the WBS Dictionary	48
7.3	"Preparation of the report" entry in the WBS Dictionary	48
7.4	"Project revision" entry in the WBS Dictionary	48
7.5	"Arduino code development" entry in the WBS Dictionary	48
7.6	"RoboDK code simulation" entry in the WBS Dictionary	49
7.7	"Sockets implementation" entry in the WBS Dictionary	49
7.8	"Construction of Master-slave prototype" entry in the WBS Dictionary	49
7.9	"PCB board design" entry in the WBS Dictionary	49
7.10	"Servomotors and EndoWrist coupling" entry in the WBS Dictionary	49
7.11	"Project familiarization" entry in the WBS Dictionary	50
7.12	"Master-slave prototype testing" entry in the WBS Dictionary	50
7.13	"UR5e and Endowrist testing" entry in the WBS Dictionary	50
7.14	"Haptic feedback testing" entry in the WBS Dictionary	50
7.15	"Results and analysis" entry in the WBS Dictionary	50
7.16	"Conclusion" entry in the WBS Dictionary	51
7.17	Project timeline	51
8.1	SWOT of the project	53
9.1	Cost breakdown of the TFG project	55

List of Figures

2.1	AESOP sugical system [1]	6
2.2	Endoassist sugical system [2]	6
2.3	ZEUS sugical system [3]	7
2.4	Da Vinci's Patients-side cart, vision tower and surgeons console [4]	9
2.5	Assorted models of EndoWrist instruments [5]	9
3.1	Surgical robotic systems	16
4.1	Connection between the different components of the prototype	19
4.2	Pen user interface	19
4.3	EndoWrist motion panel	20
4.4	Control and supervision module	20
4.5	Connexion between modules	20
4.6	UR5e robot [6]	23
4.7	Kuka 210 robotic arm [7]	24
4.8	SWIFTITM CRB 1300 robotic arm [8]	24
4.9	Virtual Incision gripper [9]	25
4.10	Hominis gripper [10]	25
4.11	EndoWrist rotation mechanism	25
4.12	Tools movement [11]	25
4.13	Testing system suture material	27
4.14	Outer skin material	27
5.1	Prototype Connection Diagram	29
5.2	Roll, Pitch and Yaw angles	30
5.3	IMU sensor	30
5.4	Pen User Interface PCB design	31
5.5	Pen User Interface final board	31
5.6	Forearm Master PCB design	32
5.7	Forearm Master final board	32
5.8	EndoWrist Motion Panel PCB design	33
5.9	EndoWrist Motion Panel final board	33
5.10	Servomotor's components	34
5.11	EndoWrist Gears	34
5.12	3D pice [12]	34
5.13	3D parts anchored to the EndoWrist	35
5.14	3D parts anchored to the servomotors	35
5.15	Coupled final design	35
5.16	Bracelet with Velcro section	35
5.17	Master fixation final design	35
5.18	Data flow through the modules	36
5.19	Graphical interface	39
5.20	RoboDK simulation	40

5.21	Graphical representation of both IMU and UR5e frames	41
6.1	Pulse intensity without EndoWrist	43
6.2	Pulse intensity with the free EndoWrist	43
6.3	Pulse intensity without obstacle	44
6.4	Pulse intensity with obstacle	44
6.5	Electrical charge without obstacle	44
6.6	Electrical charge with obstacle	44
6.7	Analog intensity without obstacle	45
6.8	Analog intensity with obstacle	45
6.9	Torque representation in the graphical interface	45
6.10	Final design device	46
7.1	WBS diagram of the project	47
7.2	PERT diagram	52
7.3	GANTT diagram	52
1	Pen User Interface Schematic	101
2	EndoWrist Motion Panel schematic	102

Glossary

3D Three-dimensional 5, 8, 11, 14, 15, 18, 22–24, 30, 34

ADC Analog to Digital Converter 32

AESOP Automated Endoscopic System for Optimal Positioning 6

CAGR Compound Annual Growth Rate 12, 13

CE Conformité Européenne 58

CPU Central Processing Unit 26

CT Computerised Tomography 15

DMP Digital Motion Processor 28

DOF Degrees of Freedom 6, 26

EAR Endoscopic-Assisted Robotic 58

FDA Food and Drug Administration 5–7, 13, 25, 58

GSPR Safety and Performance Requirements 58

HD High Definition 8

HMD Head-Mounted Display 5

IEC International Electrotechnical Commission 58

IMU Inertial Measurement Unit 26, 28, 30, 31, 37, 41

IoT Internet of Things 26, 28

ISO International Organization for Standardization 58

LESS Laparoendoscopic Single-Port Surgery 3

LESS-PR Laparoendoscopic Single-Port Surgery Radical Prostatectomy 3

MIRS Minimally Invasive Robotic Surgery System 15

MIS Minimal Invasive Surgery 11, 12, 15, 27

MRI Magnetic Resonance Imaging 18

NOTES Natural Orifice Translumenal Endoscopic Surgery 12

PCB Printed Circuit Board 22, 28, 30–34

PERT Program Evaluation and Review Technique 51

POSE Position and Orientation 40

- PROBOT** Prostate Robot 5, 11
- PUMA** Programmable Universal Manipulation Arm 5, 11, 17
- PWM** Pulse Width Modulation 31, 32, 42–44
- RAM** Random Access Memory 26
- RAS** Robotic Surgical Assist System 14, 15
- ROM** Read-Only Memory 26
- RPY** Roll, Pitch and Yaw 28, 30, 36, 37, 40, 41
- SAGES** Society of American Gastrointestinal and Endoscopic Surgeons 5
- SARP** Surgeon-Assistant Robot for Prostatectomy 5
- SCARA** Selective Compliant Assembly Robot Arm 11
- SPL** Single Port Laparotomy 12
- TCP** Tool Center Point 40
- TORS** Trans Oral Robotic Surgery 15
- WBS** Work Breakdown Structure 47

Introduction

1.1 Robotic surgery

Medical technology has undergone a great evolution throughout the last few years, leading to the progressive implementation of the latest advances in the different fields of medicine. This informatics revolution has contributed to the generation of new medical disciplines, such as nanotechnology [13], robotic surgery [14] and artificial intelligence-based techniques [15], in addition to improve the already existing diagnostic, treatment and surgical modalities.

In this context, robotic surgery emerged in the 2000s with the combination of computer science, electronics and medicine [16] to become a fundamental and pioneering tool in the surgical field. Evidence of its importance are the operation rates and the rise of this type of surgeries to the detriment of traditional ones. In the Iberian Peninsula, a total of 100 surgical robots have been in operation in different hospitals and health centres, performing a total of 60,000 operations since the installation of the first device [17]. The eagerness of the healthcare sector to invest and improve this facet of medicine has led to a year-on-year growth in the number of operations (2021 closed with an increase of 37% over the previous year). Moreover, in centres such as the Hospital Clínic, the percentage of robotic surgeries in certain fields has risen to 11% of total operations [18].

Therefore, robotic surgery refers to surgical technology that places a computer-assisted electromechanical device in the path between the surgeon and the patient. "Telepresence remote manipulators" could be a term used to refer to this concept, since this type of technology does not usually operate without the explicit control of a human operator [19]. Hence, the introduction of surgical robots aims to improve surgical activity from several standpoints. On the one hand, it intends to overcome errors and traditional difficulties [13, 20] such as tremor, instrument limitations or non-ergonomic positions [21]; meanwhile, the manual labour strengths are enhanced, increasing the accuracy and precision of movements, and improving the surgeon's adaptability to the working environment [20].

The fact that this new surgical technique has been received so positively by the medical community has facilitated the specialisation of this technology according to the type of surgery. Therefore, surgical robots can be divided into several categories, from which we can highlight laparoscopic surgical robots, neurosurgery robots, orthopaedic surgical robots and microsurgery robots [20].

1.2 Laparoscopic surgical robots

Laparoscopic surgery has been constantly evolving since its appearance in 1987 [20, 22], as minimally invasive surgery is crucial in modern medicine. The inclusion of three-dimensional cameras in the procedure and the use of robotic devices has led to improved surgical results and a decrease in postoperative complications [13].

Currently, the Da Vinci[®] robotic surgical system from Intuitive Surgical, Inc. is the most prevalent robotic surgical system in operating rooms worldwide [22], generally used in laparoscopic minimally invasive surgery, urology, gynaecology and general surgery [20]. It is a slave robot, as it has no autonomous movement capability and is absolutely dependent on the surgeon's knowledge and skills.

Its structure is specifically designed to mimic the movements of the doctor's joints, and is therefore intended to resemble the anatomy of human arms. Da Vinci® consists of a number of elements, of which the master console, the slave robot, the instrumentation and the imaging system are the most important.

The master console serves as communication between the robot and surgeon, who is placed on the control table and observes the operating field through the binoculars included in the console. By its side, the slave robot usually consists of three arms, one of which contains the manipulator for the camera and the others the articulated instrument manipulators that reproduce the surgeon's hand movements performed from the master console [23]. These robotic arms offer 7 degrees of freedom of movement in the three Cartesian planes. This allows a total of 117649 movements, which represents 0.019% of the capacity of the human arm. This is close to the percentage used by a surgeon during an operation and represents a considerable improvement over the 0.00012% that traditional laparoscopic surgery provided [23].

The Da Vinci® system provides the surgeon with intuitive control, a wide range of motion and 3D visualisation. However, the major drawback of this surgical method is the lack of haptic feedback [20].

1.3 Benefits and complications of robotic surgery

As is evident, the implementation of this emerging technology has generated a series of improvements, which facilitate the surgeon's work, improve the preceadures results and contribute to the safety of the patient [20]. On the one hand, when compared to open surgery, robotic surgery means a reduction in operating time and less hematic loss, as well as greater precision on the part of the surgeon. On the other hand, in relation to laparoscopic surgery, a reduction in the rate of conversion to open surgery is indicated and greater preservation of nerves and delicate structures is achieved [13].

Nevertheless, this technology has great room for improvement and is not without complications and drawbacks. The large dimensions of the robot require larger operating rooms, which prevents institutions with limited space from having them available. Moreover, in certain types of surgery, such as bariatric surgery, the use of surgical robots has led to an increase in operating time and infection rates. However, one of the main disadvantages, specifically in the Da Vinci® device, is the absence of haptic sensation. This is defined as tactile feedback from the console to the surgeon's hand. This lack could lead to technical errors, such as excessive use of force when manipulating tissues, or increased operating times [21].

In addition, the failure of the equipment and software during surgeries can change the surgeon's plan and compromise the safety of the surgery. Considering results from several test studies [24], robot malfunctions have been reported between 0.4% and 4.6% of all robotic surgeries performed.

1.4 Economic aspects of robotic surgery

Despite all the advantages that this surgical system offers, its widespread adoption has been limited by the high costs associated with it. The acquisition cost of the da Vinci system is approximately €1.7 million, plus around €125,000 for annual maintenance. These initial and recurring costs represent a significant barrier for many hospitals and health centres, especially in regions with limited financial resources [25].

However, compared to other surgical techniques, robotic surgery offers some cost savings, although it also incurs additional costs. For example, robotic-assisted prostatectomy has been shown to save \$487 compared to laparoscopic prostatectomy and \$1,726 compared to open radical prostatectomy, not including the cost of purchasing the robot. The reduction in surgical time and hospital stay, as well as the reduced need for medication and transfusions, are factors that may contribute to these savings, but amortisation of the cost of the robot and consumable instruments remain important costs [25].

Hence, it can be seen that the introduction of robotic surgery could lead to large savings for healthcare institutions in the long term, provided that the purchase price of robotic systems decreases significantly. The entry of new competitors into the market could help to reduce these costs, making robotic surgery more accessible.

1.5 Minimal invasive surgery procedure

In recent decades, there has been significant advancement in the field of minimally invasive surgery. This method emerged as a way of diagnosing and treating procedures involving the pleura, lungs and mediastinum. However, the constant appearance of innovative techniques has made it possible to extend this form of surgery to other structures of the thorax [26].

Currently, the medical community shows a clear preference for a series of procedures to perform minimally invasive surgery. These are single-port laparoscopy and the aforementioned robotic surgery. Both have a similar surgical structure, but the instruments and techniques used during the procedure vary. This is performed through a small incision called port, which is used to introduce the laparoscope and the necessary surgical instruments [27]. In order to make this cut, the surgeon uses trocars, an instrument that allows perforation of the musculature and fascia of the abdominal wall with a sharp edge [22].

Single-port surgery can be performed in many different procedures, such as laparoscopic single-port surgery (LESS) and radical prostatectomy (LESS-PR). Traditional single-site surgery has a learning curve and longer procedure times due to limited space for laparoscopic instruments to move. The transition to single site robotic surgery with the da Vinci Surgical Robot[®] has reduced that learning curve in addition to providing the advantages discussed above [22].

1.6 Objectives

Having introduced the topic of robotic surgery and, more specifically, the functionalities of the Da Vinci surgical device, we can identify some of the weaknesses of this technology on which we can make some improvements. In this way, future surgeries will be able to offer better results and greater safety for the patient.

Among the shortcomings of this surgical methodology lies the high acquisition cost of the technology involved, which prevents many low-income hospitals from being able to perform this type of surgery. In addition, we also find a lack of haptic sensitivity in the current systems. Therefore, a system will be developed to bring robotic surgery closer to all hospitals, with adequate manoeuvrability and agility, and with the implementation of haptic technology. The main objectives are presented below:

Main objectives:

- Development of a robust, low-cost surgical system.
- Modification of the robotic arm control system in order to improve manoeuvrability and communication between modules.

- Improve the software to facilitate the surgeon’s work, making the movements as similar as possible to those that would be performed in a conventional operation.
- Enhance the assembly system between the EndoWrist and the UR5e robotic arm to streamline the process of changing tools during the surgical procedure.
- Provide the system with haptic sensitivity, introducing actuators in the surgeon’s control that inform about the force that is being applied to the tissue.
- Design a graphical interface to facilitate the surgeon’s work, showing the orientation of the arm and the gripper, as well as a real-time indicator of the force exerted.
- Deepen in the subject of robotic surgery and learn about its evolution and future improvements, with the aim of carrying out implantations that allow for the advancement of technology.

1.7 Limitations of the project

This project has certain limitations partly related with the context in which it is developed. This is due to the fact that it will be carried out in a short period of time, only five months, and that the student’s experience in the field of robotics is quite scarce.

So, the limitations can be summarised as follows:

- Limited time, which reduces the chances of developing a system as robust as desired.
- Little experience in the robotics sector.
- The Universitat de Barcelona only owns one UR5e robotic arm, so our surgical system will only have one point of action, when the ideal would be to be able to translate the movement of both hands of the surgeon into two different robotic arms.

1.8 Characteristics of the project development

This project will be carried out at the Faculties of Physics and Medicine of the University of Barcelona, over a period of five months, starting in mid-January and ending in early June. On the one hand, the Physics Faculty will develop the entire electronic and hardware system, using its own facilities. In addition, it will also be possible to carry out the test tasks of the project, assembling the system to the UR5e robot.

On the other hand, at the Faculty of Medicine, thanks to one of the subjects of the fourth course of the Biomedical Engineering degree (Medical Applications to Engineering) we will have the opportunity to see the Da Vinci surgical device of the Hospital Clínic during an operation. Therefore this experience will allow us to become more familiar with the system and to be able to detect some more limitations of this surgical method.

The development of the project will be carried out with the help of Dr. Manel Puig i Vidal, who will offer guidance and follow-up throughout the course.

Background

2.1 First surgical robots

Throughout history, humans have sought ways to improve the outcome of surgical activities by constantly implementing novel techniques. With the improvement of technology, surgeons have been provided with increasingly sophisticated and precise tools that have led to better surgical results. One of the most significant advances in this field has been the development of surgical robots, which have revolutionised the way in which surgical interventions are performed.

The first industrial robot, called Unimate, appeared on the market in 1961 by General Motors. This technology served as a benchmark for the development and production of robotic devices capable of facilitating and improving human performance [28]. However, it was not until 1985 that the first robotic surgical procedure was performed, in which the Programmable Universal Manipulation Arm (PUMA) device was used to carry out a stereotactic brain biopsy. This robot, designed by Victor Scheinman in 1978 [1], consisted on an adapted industrial robotic arm that was used to position the biopsy needle, eliminating tremor and improving precision [28]. The successful results achieved in that first surgery led to its implementation in different surgeries at the Imperial College in London during 1988.

This technology evolved into different devices that were finally approved by the Food and Drug Administration (FDA). The prostate robot (PROBOT), the surgeon-assistant robot for prostatectomy (SARP) and the UROBOT were some of its successors. The first two performed prostate surgeries whereas the last one was mainly used in urological procedures [1]. In addition to the aforementioned devices, ROBODOC appeared as a tool to perform hip replacement surgeries, allowing precise casting of the femoral head to allow prosthesis placement [28].

All these devices were designed specifically for a particular procedure and specialty, and their main objective was to achieve the highest possible accuracy. However, the most widely used robotic surgery system today has its origins in a different concept, telepresence [28]. Robotic telesurgery appeared as a solution to the need to perform long-distance surgery on astronauts. It developed a stereoscopic display unit with 3D vision called Head-Mounted Display (HMD), which was combined with data gloves that allowed the user to interact with the virtual world [29]. Despite the technical infeasibility of implementing this technology in space operations, it served to establish the foundations of telepresence surgery. On that basis, the HMD was replaced by monitors and the data gloves by surgeon-controlled handles, in order to develop a functional device. After generating high expectations in the Society of American Gastrointestinal and Endoscopic Surgeons (SAGES), the first prototype of a surgical robot with telepresence was designed and assembled [1].

2.2 Widespread surgical robots

Currently, there are three types of robots in the surgical field according to their level of autonomy: active, semi-active and master-slave systems. Active systems perform a series of pre-programmed functions autonomously under the supervision of a surgeon. The PROBOT and ROBODOC devices discussed above are examples. Semi-active systems,

on the other hand, combine and complement the preprogrammed actions of the robot with those of the surgeon. As for the master-slave systems, they lack preprogrammed elements and depend entirely on the surgeon's activity. The AESOP surgical system, Endoassist, ZEUS and Da Vinci[®] robots, the most widespread devices in their respective era, incorporate this type of system [30].

2.2.1 AESOP

The Automated Endoscope Optimal Positioning System (AESOP) was developed by Computer Motion Inc. founded by Yulin Wang in 1990. AESOP was the first surgical robotic system approved by the FDA in 1994. It was a robot with a master-slave system designed to grasp and guide a laparoscope under the surgeon's control [28]. For this purpose, AESOP used a voice-controlled 4 Degrees of Freedom (DOF) robotic arm [3], eliminating thus the need of an assistant to perform this task [1].

One of the main advantages of AESOP was the improvement in accuracy, as eliminating the need for an assistant to hold the camera provided more stable and precise control of the laparoscope, potentially improving surgical outcomes. In addition, the use of a robotic arm to hold the camera reduced physical strain and fatigue for the assistant [1].

However, its functionality was mainly limited to controlling the endoscopic camera and did not assist in other surgical tasks. Moreover, surgeons had to adapt to the new system, which could initially slow down procedures while they were getting used to the voice controls and the robotic arm [28]. A representation of the surgical system can be found in 2.1.

2.2.2 Endoassist

It is a device very similar to the AESOP for holding and positioning the camera. It is controlled by a sensor that detects the surgeon's head movements, and by pressing a foot pedal, the camera is positioned where the surgeon is looking. Although it was as good an instrument as the AESOP, it was not as widely used. [31].



Figure 2.1: AESOP surgical system [1]



Figure 2.2: Endoassist surgical system [2]

2.2.3 ZEUS

It was in 1996 when ZEUS was introduced, a robotic surgery system that included AESOP and 2 laparoscopic instruments with 7 degrees of freedom, which increases the dexterity and improves the surgeon's feel. The introduction of the AESOP system made it possible to dispense with the surgical assistant holding the camera and changing instruments, thus reducing errors and possible contamination of the camera by contact with the tissue. On the other hand, the voice control of the arm enabled by the system gave the surgeon back the ability to dominate the operating field as in an open operation [29].

The ZEUS surgical system introduced a number of improvements and conveniences that translated into better operating results. First, the ergonomic design of the remote workstation eliminated awkward postures. Secondly, the 15-fold magnification of the image by the system's camera allowed for more accurate operation. Finally, it allowed motion scaling to be adjusted and jitter to be reduced. Motion scaling reduced the surgeon's movement at the console to finer movements inside the patient, which increased the stability of the operation and reduced operation errors [29].

Although the adaptation and learning time for this system was longer than for traditional laparoscopic surgery, the precision, comfort and results were notably improved. However, despite all these advantages, two years after receiving FDA approval, the acquisition of Computer Motion, Inc. by Intuitive Surgical Inc. resulted in a halt to the development of ZEUS [1].



Figure 2.3: ZEUS surgical system [3]

2.2.4 Da Vinci Surgical System[®]

Intuitive Surgical Inc. developed an initial prototype of the current Da Vinci Surgical System[®] during the early stages of the company's formation. This was named Lenny, and included 3 robotic arms, one of which was used for the laparoscope and the others incorporated non-interchangeable instruments with 7 degrees of freedom. The second prototype, Mona, incorporated interchangeable instruments, but lacked an arm to hold the camera, requiring a bedside assistant to hold the laparoscope. After solving the main problems with the latter prototype, the first model of the Da Vinci Surgical System[®] was released in 1999 [28].

Currently, the Da Vinci Surgical System[®] is the most widely used robotic device in the surgical field. It is a master-slave device initially intended for cardiac surgery. However, the first results did not meet expectations, and the da Vinci[®] ended up being used generally in gynaecology and urology surgery, achieving very satisfactory results. It was not until 2000 when the FDA approved the device in human abdominal surgery. The technical improvements presented by this novel device, such as high magnification three-dimensional vision or precisely controlled EndoWrist instruments with seven degrees of freedom, made the robotic platform more attractive for surgeons to use da Vinci in a wide range of surgical procedures during the first decade of the millennium. In addition, this technology offered stable imaging of the operative field, image magnification up to 10 $\times$, physiological tremor filtering, and motion scaling up to 5:1 [1].

Due to the acquisition of Computer Motions by Intuitive Surgical, the technology present in the ZEUS device could be used to improve the next generations of systems. This facilitated the emergence of several generations of the platform: DaVinciS, Si, SiHD, XandXi, SPandIon [28]. Throughout this evolution, it developed from a three-arm system to a four-arm system. The first device to incorporate the four arms was the da Vinci S,

with improved vision technology (including 3D HD) and easier setup. This was followed by the da Vinci Si with further visual enhancements and upgradeable architecture, and the da Vinci Xi with improved vision and laser guidance [30]. Despite the considerable improvements of the new models compared to the previous ones, the concept and structure of the robot have remained unchanged in the five generations marketed to date. It consists of three distinct sub-systems: the patient-side cart, the surgeon's console and the vision cart [32].

On its side, the surgeon's console is operated by the doctor, from which he has total control of the wristed instruments on each of four arms located at the patient side and observes video images of the operating area in a 3D HD quality [33]. This connection with the surgical field is made by means of the principle of teleoperation. In this context, the surgeon's master interface and the surgical instruments are connected by means of a control system. Since this is an electronic link, the control system software can modify the signals to filter out the surgeon's physiological tremor or reduce the surgeon's movements to improve accuracy. Moreover, the comfortable seating position and the arm rest provides the surgeon ample support allowing him to comfortably manipulate the master slave system [34]. So, the surgeon's control is composed of four different elements. The master control, from which the surgeon controls the endoscope and the instruments placed inside the patient; the serious viewer, which provides 3D imaging to the Surgeon Console operator; the Left and Right Side Pods that give user interface functions for system configuration; and the Footswitch Panel, that provides the user interface for instrument arm repositioning, camera and focus control, and energy use during surgery [35].

Meanwhile, the patient-side manipulators are assembled on the patient-side cart by means of a configuration structure. Each of these manipulators is capable of holding a laparoscopic camera or a surgical instrument, which can be changed during the surgical procedure by an assistant positioned next to the cart. The da Vinci trocars have a remote centre feature that keeps the fulcrum of the trocar centred in the wall of the body to minimise torque or excessive trauma of the surrounding tissue. This fact is especially important in the chest whereby the remote centre keeps the trocar from rubbing on the surrounding ribs [36]. Being in direct contact with the patient, the instruments work in the sterile field, while the surgeon and his console can remain outside the sterile field [32].

The Endowrist Stapler and Vessel Sealer are some of the innovations that have been introduced in recent generations. The Endowrist Stapler is a linear, endoscopic, articulating mechanical stapler. It has a wide mobility articulation (108° on the horizontal axis and 54° on the vertical axis), which allows reaching deep anatomical locations without sacrificing precision and safety. The Vessel Sealer is an electrosurgical instrument capable of sealing and cutting tissue structures. It is also extremely versatile and articulable and facilitates safe and precise access to the surgical field [37]. These instruments are supported by the Instrument arms, which have a wide range of motion in order to help simplify port setup and provide greater reach into the patient anatomy [35].

Finally, we have the vision cart. This contains a central image processing unit for real 3D vision [34], which is composed of a 24-inch touchscreen monitor, an ERBE VIO dV electrosurgical unit and a series of shelves for storing auxiliary surgical equipment [37]. To this unit belongs the endoscope, which can be a 12 mm 3D endoscope or a 5 mm 2D endoscope. The light from the illuminator is sent through the shaft of the endoscope by fibre optic cables and is projected onto the surgical site, and the heat transported by the fibre optic cables reduces fogging on the lens [35].



Figure 2.4: Da Vinci's Patients-side cart, vision tower and surgeons console [4]

2.3 Endowrist instrument

Intuitive surgical's Endowrist instruments are designed to provide surgeons with natural dexterity and the full range of motion for precise operation through tiny incisions. Modeled after the human wrist, Endowrist instruments can offer an even greater range of motion than the human hand. They allow the da Vinci[®] system to take surgical precision and technique beyond the limits of the human hand. Endowrist instrument's internal cables provide maximum responsiveness, allowing rapid and precise suturing, dissection, and tissue manipulation. This tool provides 90° of articulation, motion scaling and tremor reduction, intuitive motion and finger-tip control and, when coupled to the Da Vinci[®], provides 7 degrees of freedom.

The EndoWrist instruments are available in a wide selection of specialized tip designs to enable a broad range of da Vinci[®] procedures. As da Vinci[®] surgery is adopted in new specialties, EndoWrist models are being designed in order to address new clinical needs. Among the wide variety of tools currently available, we find the energy instruments, which are used to provide coagulation, cutting, and dissection of tissues.

On the other hand, we find the grasping instruments, that allow the manipulation of a wide range of tissues, from fine, thin tissues such as peritoneum to dense, fibrous tissues such as the uterus. For its part, needle drivers provide the ability to suture with the finest of needles used in cardiovascular surgery as well as the thickest needles used in the repair of uterine defects [38].

Intuitive Surgical's catalogue includes two different dimensions of the tool, in order to adapt to the surgeon's needs. On the one hand, the 8 mm are intended for conventional procedures, while a selection of 5 mm EndoWrist instruments provides the surgeon with the ability to use smaller access ports [39].



Figure 2.5: Assorted models of EndoWrist instruments [5]

2.4 Haptic technology

The introduction of minimally invasive robotic surgery was intended to overcome some of the shortcomings of traditional instruments. However, this resulted in a complete loss of haptic feedback from the surgeon, depriving him of any cutaneous or kinesthetic sensation [40]. This factor forced surgeons to rely solely on visual cues and their experience

to perform the operative movements, which lead to an increase in operating time and an increase in the number of errors. Therefore, an adequate tactile response allows precise localization of anatomical structures, in addition to providing the necessary information about the force exerted, which plays a key role in reducing injury to delicate neighbouring structures [41].

In the 1990s, haptics research expanded significantly with the availability of high-fidelity force feedback systems. Nowadays, much of the force feedback research is focused on developing practical systems for application in several fields, one of which is medicine. Although researchers have been studying tactile feedback for many years, there is currently no commercial tactile display system that provides distributed information to the skin in a compact package feasible for applications. Haptic feedback systems used in robotic surgery consist of two key components: sensors and actuators. While sensors detect the forces applied to the tissue by the surgical instruments, actuators transmit the signal detected by the sensors to the surgeon's fingertips. However, current actuators are not as accurate as they should be, as they only provide simple information such as vibration and static pressure [40].

When we talk about force feedback, we are referring to the measurement of the forces applied to the patient by a surgical instrument and the provision of that force to the hand through an actuator device. Currently, commercially available sensors are very effective in many teleoperative operations, however, their characteristics such as size, cost or biocompatibility represent severe restrictions in the surgical environment. In addition, the fact that robotic instruments have not been designed to detect the force performed has made it difficult to incorporate these sensors. One approach to solving this problem could be to redesign the instruments to accommodate the integration of this technology. Direct force feedback to the surgeon's hands can be achieved through the master manipulator motors, which are programmed to recreate the forces sensed by the patient-side robot. However, the robotic arm features seven degrees of freedom of which, some of them, are not actuated on the master. This makes it impossible for the system to provide force feedback in some directions, resulting in potentially detrimental effects [40].

On the other hand, tactile feedback is also a necessary resource throughout the surgical procedure, since palpation is a particularly relevant task to learn about the different mechanical properties of the tissue. In addition to cost and biocompatibility limitations, many sensors require some deformation to measure distributed information, requiring flexible coatings, which also eliminate detailed local information [40]. Therefore, this is a technology that currently has many limitations and complications but whose implementation is indispensable to greatly improve robotic surgery.

2.5 Project context

Once all the advantages and disadvantages of the Da Vinci Surgical System[®] are known, we can conclude that its implementation has led to a great leap in the quality of surgery. However, we have seen that this surgical system has a high acquisition cost, so that many hospitals are deprived of using this new technology. In this context, our project is designed to develop a surgical system with a lower cost than those currently available on the market, but at the same time with a high level of reliability and comfort for the surgeon.

Once the system we have developed is as reliable as desirable, it will be provided with an haptic sensitivity, one of the major disadvantages of robotic surgery today. In this way, the surgeon would have real-time tactile feedback of the force he is exerting on the patient's tissues.

Market analysis

3.1 Market evolution

Over the years, the introduction of new features in robots has allowed for their classification into five distinct generations. Each generation comprises a set of devices that share similar advantages, limitations, and technological advancements. These generations are outlined below [42]:

- **First generation (stereotaxic robots):** The beginnings of robotic surgery date back to 1985, with the appearance of the first PUMA surgical robot. This device was used to perform a stereotactic brain biopsy and acted as a precursor to a modified brain tumour excision device known as Neuromate. Similarly, ROBODOC, AcroBot14 and SCARA also belong to this first generation. These were introduced in the field of orthopaedics to perform procedures that had a clear mathematical and mechanical strategy.

The different components of this group had their origin in either industrial or non-industrial environments. Moreover, they were all master-slave robots, in which the robot acted as an extension of the surgeon and never performed tasks autonomously. This was due to the fact that, at that time, a human surgical review was required at the end of each step of the robotic surgery. The demand for these robots was driven by the need for greater precision in delicate procedures where a small error could have serious consequences. However, the limitations offered by the systems at the time caused surgeons to be wary of the future of robotic surgery, and accused an improvement in dexterity and flexibility in the operative tissue targets.

- **Second generation (endoscopic robots):** It has been with the introduction of the second generation that the major expansion of robotic surgery to date has occurred. The first device that can be considered from this group is the PROBOT at Imperial College London, as it introduced soft tissue surgical capability. With the emergence of this new capability, potential basic advantages of robotic surgery over traditional MIS surgery were realised. These include improved visualisation through the use of 3D consoles, increased tactile feedback or reduced difficulty of access to different tissues and organ systems.

Two of the best known systems, Computer Motion’s ZEUS and Intuitive Surgical’s Da Vinci, belong to this second generation. Despite the introduction of new endoscopic platforms in the market, the Da Vinci system continues to maintain its dominance in the market. This has been achieved thanks to its unique market for so many years and the exclusive features that make it the most advanced device to date. This technology is primarily applied to the specialties of urology, general surgery and gynaecology. At the end of 2018, there were more than 4,600 Da Vinci robots worldwide, 3,010 of which were in the United States. In Europe this number was 800, while in Asia it exceeded 600 [43]. This large expansion of robots belonging to this generation was due to their ability to perform minimally invasive procedures with greater precision and less trauma to the patient. The possibility to perform complex tasks in hard-to-reach areas also drove adoption.

As these second-generation devices are the most widespread and established in clinical use today, many of the latest technological innovations are being incorporated to provide them with new capabilities to enhance the surgical experience.

- **Third generation (bioinspired robots):** The third generation of surgical robots, most of them designed during the last decade, adopted the advances in biomimetics, bionics and autobionics that had occurred since the 1950s. One of these was the NOTES platform, which consisted of inserting surgical instruments through a natural orifice in order to reduce the number of scars. By eliminating external incisions, NOTES procedures could reduce patient trauma, the likelihood of infection, and recovery time [44]. In addition, the evolution of the multiport MIS method into Single Port Laparotomy (SPL) is noteworthy, which made it possible to reduce the impact of surgical exposure. All this set of innovations combined allowed the use of MIS instruments with articulated tips facilitating the use of these devices for the treatment of multiple surgical pathologies.

This generation is made up of robots such as the i-SNAKE and the CardioArm from Imperial College, both included in the flexible tendon-driven systems, or the Amigo from Catheter Robotics Inc. and the Magellan from Hansen Medical Inc. belonging to the catheter navigated systems. These new devices are being widely accepted by the medical community, which sees a need to perform complex surgeries with fewer incisions and less impact on the patient's body.

- **Fourth generation (microbots):** This generation consists of these microscopic-sized robots, which could be introduced into the body with a minimal surgical footprint to visualise and treat not only conventional surgical diseases, but also non-surgical diseases such as infectious and immune processes that could be managed at the cellular level. With the incorporation of advances in biomimicry, they will be able to provide a higher resolution micrometre image in real time of the patient's diseases and anatomy.

Currently, capsule endoscopes, such as the PillCam[®] WCEs-Wireless Capsule Endoscopes, are under development. These generally function as passive mobile capsules carried by the peristaltic motility of the gastrointestinal system while imaging the gastrointestinal tract. The next generation of these microrobots would include new advances in each of their components, from vision, locomotion, localization, power, diagnostics and tissue manipulation [42]. On the other hand, we can find other examples such as responsive microgrippers, which can perform tissue biopsies without the need for external connections, magnetically actuated microrobots, used in eye surgeries, and ultrasound-driven microbullets, for deep tissue ablations [45].

- **Fifth generation (autonomous systems):** This latest generation is still conceptual, as it has not yet succeeded in developing a fully autonomous system to perform individual pre-programmed tasks. The possibilities of this new system are immense, as they can range from a humanoid-like cyborg platform to a swarm-like system with comparable swarm intelligence.

3.2 Current market

During 2019, the global robotic surgical procedures market was valued at \$ 5.32 billion. Studies have predicted market growth that may reach \$19.03 billion by 2027, exhibiting a CAGR (compound annual growth rate) of 21.8 %. However, the COVID-19 pandemic had a negative impact on the robotic surgery market, even causing a -14.3 % drop in 2020,

due to the cancellation or postponement of robotic surgical procedures. According to the CovidSurg Collaborative research study, a total of 28 million elective surgeries worldwide were cancelled or postponed over a 12-week period due to the coronavirus outbreak. In addition, the sale of surgical robots was severely affected. For example Intuitive Surgical reported a 6 % decline in its first half 2020 revenue from the previous year [46].

Despite the hurdles this posed, the robotic surgery market has maintained similar growth to that prior to the pandemic over the past two years. This is due to factors including high demand for automation, increasing incidence of chronic diseases, growing geriatric population, complexity of surgical procedures, and increasing demand for non-invasive surgeries with more precision and flexibility [47].

Currently, there have been numerous mergers and acquisitions among medical robot manufacturers. As a result, there has been an increase in the number of robotic surgical procedures, faster market growth and the introduction of new technologies and applications. In addition, several market players have developed strong distribution networks through agreements and partnerships with distributors and hospitals in emerging and developed countries in order to establish their position in the global market. All these ongoing and future initiatives will accelerate the adoption of surgical robots globally and subsequently drive the growth of the robotic surgical procedures market. An example of such initiatives could be Intuitive Surgical and Stryker, both of which are constantly focused on innovation and updating their portfolios with new and advanced product offerings. The former announced in 2018 that the FDA had approved its Da Vinci SP system for urology [46].

The United States is currently the region with the highest dominance in the robotic surgical devices market. This is due to the higher prevalence of chronic diseases, increasing adoption of advanced devices, and the presence of favourable reimbursement policies for robotic surgical procedures. As far as Europe is concerned, CAGR is expected to grow due to the number of patients with urological and orthopaedic disorders, as well as due to the launch and approval of new surgical systems [46]. On the other hand, Asia Pacific will witness a huge growth in the market, due to the increasing demand in countries such as India, Japan, China, combined with the growth of robotic surgery in that region. In China, in particular, several national versions have appeared, which has led to a drop in prices. In addition, the large investments being made in Asian countries make it easier to pay for the costly procedures [47].

Against this backdrop of expansion, a number of companies have emerged that aim to expand their customer bases across foreign countries and take advantage of strategic collaboration, merger and acquisition initiatives to increase their market shares and profitability [47]. However, in terms of income, the market is still dominated by Intuitive Surgical due to its diversified product offerings, strong brand image, and extensive distribution networks in developed and emerging countries. Some of the companies making space in this market could be Stryker Corporation, Accuracy Incorporated, CMR Surgical, Renishaw PLC or Catalunya's Rob Surgical [46].

3.3 Surgical robotic systems

As discussed in the previous sections, the market has been dominated by Intuitive Surgical since the launch of its Da Vinci device in 1999. However, the expiration of some patents in 2019 caused an opening of the market, which different companies have taken advantage of to market several robotic surgery devices that could end Intuitive Surgical's current monopoly.

Despite the rumours that have appeared about the new robots from companies as important as Google or Johnson and Johnson, they have not yet obtained the necessary approval to bring them to the market.

On the other hand, many companies have developed surgical robots to take advantage of the new situation. These new robots bring other advantages, such as artificial intelligence, better prices (although this remains to be demonstrated), 5G technology, more compact models, or material that is not exclusive to the brand [48].

Today, in addition to the Da Vinci model, the most evolved and approved robots are the following:

- **Hugo RAS (HugoTM Robotic Surgical Assist System):** This is a device designed by Medtronic, a major company focused on the sale of medical devices. It was designed to assist the surgeon in approaching minimally invasive surgeries, specifically those within the field of urology, gynaecology and oncology [49]. It is a system consisting of a console, a laparoscopic tower and four independent modules that support the arms. The latter modules differ from the original Da Vinci design, which consists of a single structure supporting the four arms. The console is open with a three-dimensional viewing screen that requires glasses and has several foot pedals similar to the da Vinci. As for the controls, these are similar to the small ligasure handle, which provides a good grip. Also, the angle of movement of the instrument is somewhat larger than the competition. However, there is a slight delay in the transmission of the coagulation command. Overall, this device establishes a non-inferiority relationship with the Da Vinci model, which is a great achievement, since it is the first model to be marketed by Medtronic [48].
- **Versius CMR Surgical:** The Versius surgical robotic system, developed by CMR Surgical, is designed to facilitate minimally invasive surgery. It features three mobile instrument units that can connect to a variety of articulated surgical instruments. Additionally, the system includes a console controller with a three-dimensional screen, offering an interface similar to modern gaming consoles. The most outstanding feature of this device, which makes it different from the Da Vinci surgical system, is the renunciation of foot pedals to allow the surgeon to perform all tasks with the hands. One of the advantages of this technology is the lightness and good design of the arm, which barely reaches 100 kg in total, including both the arm and the base. It is fixed directly to the instrument, without the need to fix the trocar, which reduces wall trauma, in addition to leaving room on the surgical table for surgeons and assisting instrumentalists. However, there are some drawbacks, such as the difficulty of handling the joysticks of the camera, which compromises the comfort and intuition of the surgeon [50].
- **Bitrack System:** This system has been put on the market by the Barcelona-based company Rob Surgery with the intention of improving the efficiency of current robots through the use of unique technology, cost reduction and improved usability. It is a device designed to perform a wide variety of abdominal and pelvic surgical procedures with sub-millimeter pressure. In addition, the dynamic fulcrum and new automatic functions reduce forces at the incision site and prevent possible unexpected movements. Unlike the Da Vinci Surgical System, this one only consists of a robot and a console. The former consists of 4 arms, which have 7 degrees of freedom and incorporate generic trocars and a vision system. As for the console, it consists of a 3D screen and a large, ergonomic workspace.

With this system, the company has offered a new approach to robotic surgery, hybrid surgery. This is a technique which allows combining MIS and MIRS (Minimally Invasive Robotic Surgery System) in the same procedure by means of a quick and agile changeover. The fact that the high precision provided by RAS is not necessary during the whole process makes this technique a breakthrough that facilitates the surgeon's work and saves time [51].

- **Flex Robotic System:** The Flex Robotic System is a device design by the company Medrobotics Inc. intended for robot-assisted visualisation and surgical site access to the oropharynx, hypopharynx, and larynx in adults, and has been developed to broaden the scope of TORS offered by the da Vinci Si HD System. This is a system that combines the flexibility of an endoscope to access surgical sites, as well as the ability to harden itself to perform the operation. In the same way as the Da Vinci system, this is also a master-slave robot, where all actions are performed by the surgeon. The flexibility offers 102 degrees of freedom in angular movement, which makes it possible to operate in regions that are difficult to access. One of its strong points is the presence of tactile feedback, which allows the surgeon to have full control of the tool. In addition, the surgical site is visualised using a 3D camera incorporated into the distal end of the flexible endoscopic arm. However, the difficulty of system configuration and preparation can affect operational efficiency in a busy surgical environment [52].
- **DLR MIRO:** The DLR MIRO is a versatile robotic arm designed for medical applications, first introduced in 2008 by the DLR Institute of Robotics and Mechatronics. It stands out for its compact size, with a weight of approximately 10 kg and a maximum length of 1.1 meters, and for its seven degrees of freedom. Unlike current Da Vinci systems, this one incorporates modern haptic technology consisting of two haptic devices that allow the surgeon to feel the force feedback. Among its advantages is its fully torque-based control capability, which enables sensitive and precise movements, as well as its versatility when combined with specialized instruments such as the DLR MICA for various surgical procedures. However, a disadvantage is that its technical complexity requires a basic understanding of the system by the user, which can demand a considerable learning curve [53].
- **Mako system:** Already outside the Da Vinci field, we can find the Mako surgical system, designed by Mako Surgical Corporation and later acquired by Stryker Corporation, which was launched to address the surgeon's need for technical imperfections and enable consistent surgical outcomes in the field of orthopaedic surgery. This is an image-based system where a CT scan is used during surgical planning to help determine component sizing, placement and bony resection. In addition, the robot provides haptic feedback to the surgeon to avoid bony resection outside of the executed plan. With all these advantages, it has been possible to address technical errors related to unicompartmental failures, such as misalignment or malposition of the joint line due to excessive or insufficient bony resection. Studies have shown that, compared with manual techniques, the Mako system has greater accuracy in recreating posterior tibial tilt [54, 55].

Figure 3.1 shows an image of each of the surgical systems discussed above.

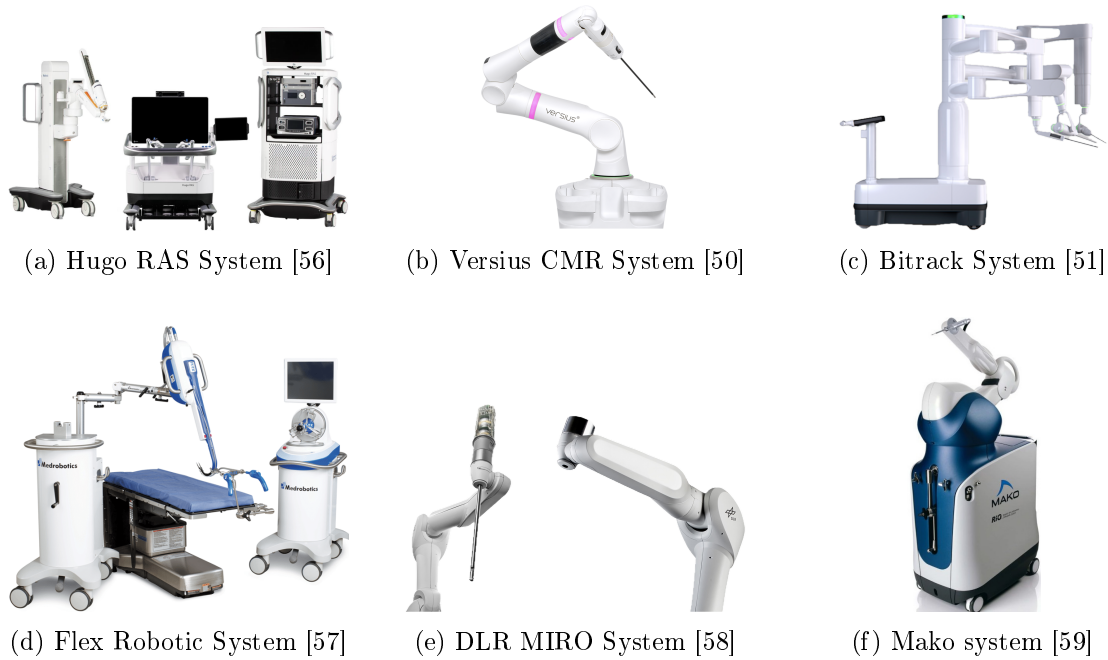


Figure 3.1: Surgical robotic systems

3.4 Patents

As indicated in the section 1.6 Objectives, one of the goals is to provide haptic sensitivity to the UR5e robotic arm assembled with an EndoWrist tool. However, as this is a topic of so much current interest, the number of patents that could hinder, slow down or increase the costs of the research has increased considerably. There are currently a total of 3012 patents related to the Da Vinci surgical system, published by Intuitive Surgical Inc. on the specialized patent portal Google Patents. However, if we narrow down the search, limiting it to patents concerning haptic sensitivity, the number decreases to 904.

Among all the published patents, we find a large number that focus on developing a system that allows the transmission of the reaction force exerted by the tissue on the Gripper to the surgeon. Some examples can be seen below:

- Surgical Manipulator with Haptic Feedback (JP2020039898A): describes a surgical manipulator that incorporates advanced haptic feedback. This system uses advanced sensors to detect interaction forces between surgical instruments and patient tissues. Haptic actuators generate forces on the surgeon’s hand controller, mimicking tactile sensations. This improves surgical precision, reduces the risk of damaging sensitive tissues, and provides a better sense of control and responsiveness during surgery. In addition, the system is designed to integrate with existing surgical equipment, such as the da Vinci system, enabling improvements without the need to replace equipment [60].
- Application of force feedback on an input device to urge its operator to command an articulated instrument to a preferred pose (EP2775950B1): focuses on a force feedback system for robotic medical devices. This system applies force feedback to the surgeon’s input device, guiding it to move the instrument to a preferred pose. It operates in two modes: normal, to maintain the optimal pose during surgery, and retraction, to facilitate the transition to a safe pose. Key components include an input guide, an instrument controller and a stereo camera for visualization [61].

- Master control device with finger grip sensing (WO2019217882A1): describes a master control device with finger grip sensing. This device includes an elongated control body with a surface sensitive to the force applied by the user’s fingers, equipped with force sensors that detect pressure and generate signals based on the amount of force detected. It also has position and orientation sensors for the control body [62].

As we can see, the published patents focus on a technology that is outside the scope of the project. For this reason, the ones concerning haptic feedback will not cause us any setback. On the other hand, patents related to other fields of robotic surgery, such as robotic arm control or the integration of image-based processes, should also be analyzed. This will allow us to orient our project to establish a research path for possible future improvements.

- Method and system for moving plural articulated instruments in tandem back towards entry guide (JP2017153998A): describes a method and system for moving several articulated instruments backwards together towards the infeed rail. This is done prior to changing the orientation of the infeed guide and instruments, providing tactile indications and speed limits to assist the operator. Once the instruments have been retracted, the entry guide can be pivoted without risk of damaging the patient’s anatomy. [63]
- Preventing instrument/tissue collisions (US10695136B2): describes methods to avoid collisions between minimally invasive surgical instruments and patient tissue. It uses a real-time image of the surgical site superimposed on a previously recorded image further distal. Surgical instruments are visible in the real-time image, and representations of the instruments are shown as they would appear in the recorded image, allowing the user to view them outside the field of view of the real-time imaging system [63].

However, the fact that patents expire 20 years after publication and that many of these were launched together with the surgical device in 2001, has favoured the emergence of many other systems that have taken advantage of the opportunity.

Currently, the characteristics of our project, such as the use of the UR5e robotic arm, programmed using Python, the generation of haptic sensitivity through the use of a vibrating motor or the non-incorporation of image processing methods, exempt us from any patent on the market. Therefore, we do not find any limitations in this aspect.

3.5 Future prospects

It is clear, therefore, that robotic surgery has undergone constant improvements over the last three decades. Since the first industrial devices appeared in 1985, such as PUMA or ROBODOC, there have been advances both in the size and anatomy of the robots and in their intelligence and precision. However, current designs are far from flawless, forcing researchers to find ways to improve their performance. Several fronts are currently open, addressing this problem from different angles. Mainly, the aim is to improve the ergonomics and design of the robot, reduce costs, improve the software and implement advanced technology.

On the one hand, the desire to reduce invasiveness requires redesigning the current models to achieve a more compact and resistant structure. In the case of single-port interventions, it requires a new intracorporeal performer to overcome the current limitations [64]. In addition, in general operations, advances are intended to allow automatic

instrument exchanges and the elimination of power cables [28]. New robotic surgical systems are currently being developed and tested with unprecedented technological resources, such as the Surgeon Eindhoven Operative Force Feedback Interface (SOFIE, Eindhoven University of Technology, The Netherlands), with haptic feedback [1].

Another highlight in the future improvements of these devices is the use of the data collected during all operations to achieve automatic learning by incorporating artificial intelligence. In this way, the system could autonomously recognize the relevant anatomy and perform surgical tasks appropriately [64]. This could offer a host of benefits, such as improved quality, the possibility of early warning to prevent complications, or automation [28]. Although some such technology has already been developed, a great deal of research is still needed in terms of design, biocompatibility, autonomy, monitoring and control [65].

On the other hand, another key aspect for the correct functioning of the device is the interaction between the robotic instruments and the surgical staff. The implementation of an adequate human-robot interface and specific control strategies is fundamental for the safety and optimal outcome of the surgery. Recent studies aim to achieve a robotic system that enhances cooperation with the surgeon by providing not only superhuman dexterity, but also real-time guidance, sensing and integrated decision making of intraoperative images. To this end, MRI-compatible surgical robots will enable the use of real-time functional 3D imaging in the operating room to provide an unprecedented level of surgical navigation, dexterity, and comfort [65].

Finally, it will be necessary to reduce the cost of robotic equipment to allow faster adoption [1]. Because of the additional materials, the cost of this type of surgery is very high. In certain operations, such as total laryngectomy, a 90% increase over traditional surgery has been reported (€3,581 vs. €6,767). While in robotic surgery the costs come from the instruments themselves, in conventional surgery they are related to the personnel. With these values, it has been calculated that about 300 surgical operations per year are necessary to achieve cost equivalence between the two techniques [66].

Conceptual engineering

In this section, an analysis of the different alternatives to build the final device is going to be carried out, taking into account both their advantages and disadvantages. This will help to choose the best options, either software or hardware, and to acquire an overall view of the project. As for the software, factors such as price, student experience and ease of use will be taken into account. On the other hand, as far as hardware is concerned, the different components that can be used and their different characteristics will be numbered. In addition, the impossibility of using the Da Vinci surgical robot, due to its high cost and low availability, a robotic arm will be chosen from among those available on the market.

In order to make the selection, it is first necessary to know how the device will work. It will be made of four modules, which will be wirelessly connected to each other and will provide the robot with controlled movement. Figure 4.1 shows the relationship and connection between all the components of the prototype.

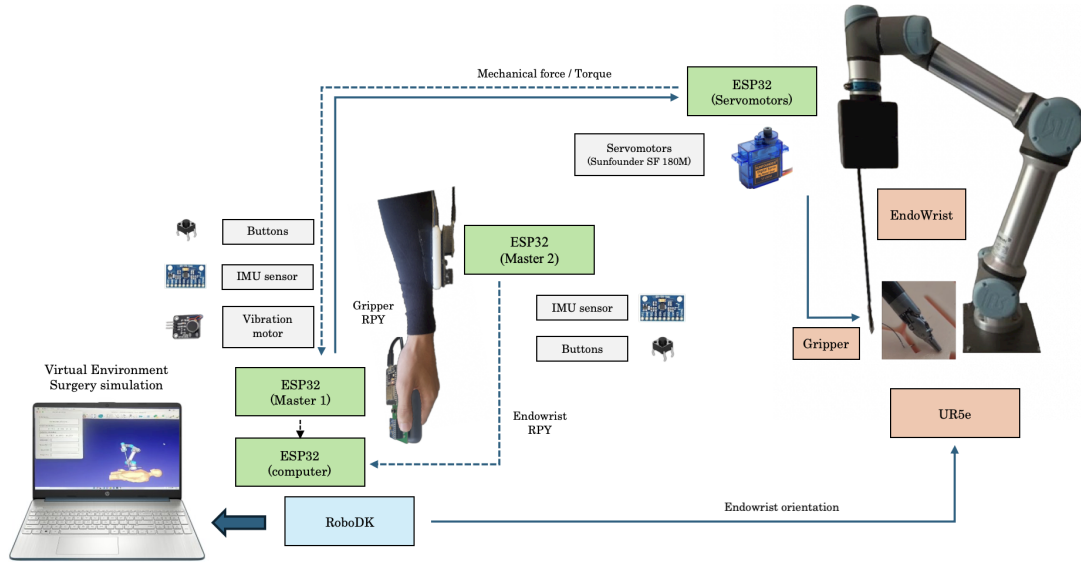


Figure 4.1: Connection between the different components of the prototype

Besides, it is necessary to know what are the functionalities of the four main modules:

- Pen User Interface (Master 1) and Forearm Master (Master 2): This prototype is designed to have two Master devices. These will be the ones that the surgeon will manipulate in order to control the surgical arm as well as the EndoWrist gripper. The Pen User Interface will be controlled by the hand, whose movement will be translated into the gripper, while the remaining one, the Forearm Master, will be located on the forearm, controlling the movement of the robotic arm.

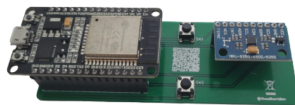


Figure 4.2: Pen user interface

- EndoWrist Motion Panel (Servomotors): This element will be the one that will be connected to the robotic arm and the one that achieves the movement of the EndoWrist tool. This movement will be performed by means of the activation of four servomotors, which will operate according to the commands sent from the pen user interface.

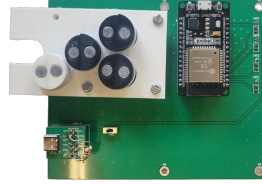


Figure 4.3: EndoWrist motion panel

- Control and Supervision Module (Computer): This last element allows the connection of all the modules with the computer, and also allows the action of the two previous ones to be visualised in the computer's simulator programs.



Figure 4.4: Control and supervision module

As mentioned above, a connection is established between the four modules so that all the objectives set out in the Introduction section can be achieved. These can be summarised in Figure 4.5.

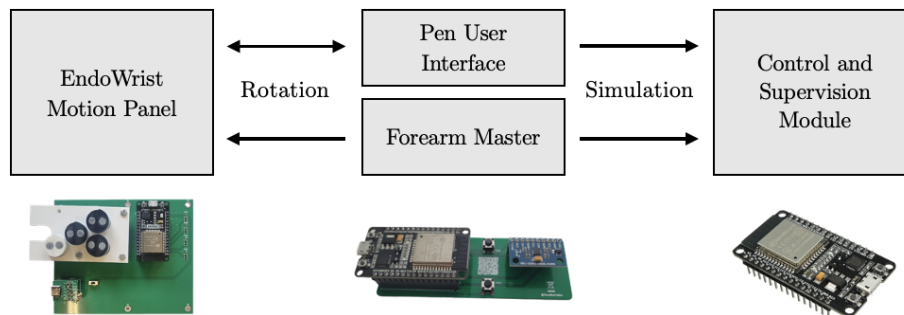


Figure 4.5: Connexion between modules

4.1 Software

Software development is a fundamental aspect for the success of our project. Therefore, it is necessary to make the right choice of language and platform. In addition, it must be distinguish between the software used for the development and creation of code that allows the modules to be correctly connected and the desired functions to be executed, and the design software that enables the creation of PCB.

4.1.1 Programming software

In order to choose the best program, different parameters have to be taken into account: cost, type of language used and the student's previous experience using that language.

- Arduino: It is an electronic prototyping platform that allows users to easily create interactive projects and physical devices. Its most important pros include its accessibility, as it is easy to learn and use, as well as being a free tool. It also offers a wide range of compatible boards and modules, and an extensive open source library to build on existing projects and components. However, some cons of Arduino include its limitation in terms of processing power and memory, which can restrict the complexity and scope of projects. In addition, programming is done in a proprietary C/C++-based language, which may require some level of programming knowledge [67].
- Python: It is a high-level programming language recognized for its clear and readable syntax. Some of its most notable features include an extensive standard library, versatility in diverse applications, a large developer community and ease of learning. Python is highly productive and offers agile programming. However, some cons are that it may require more memory and processing resources compared to low-level languages, and certain applications that require high speed may be better suited for other languages [68].
- Matlab: It is a programming environment and development platform used primarily in the field of mathematics and engineering. Its pros include its wide range of specialised tools and libraries for numerical computations and data analysis, making it a powerful choice for research and technical development. MATLAB offers a syntax that is easy to learn and use, which speeds up the programming process. However, its commercial licence limits its accessibility for some users, and its performance may be inferior compared to other general-purpose programming languages [69].
- LabVIEW: It is a visual programming and system development environment that is widely used in control, acquisition and automation applications. It includes its visual, block-based approach, which makes it easy to create programs by interconnecting graphical elements. LabVIEW offers a wide variety of tools and libraries for data acquisition, signal processing and instrumentation control, making it a popular choice in engineering and research lab environments. However, some cons of LabVIEW are its initial learning curve, as it requires familiarity with the visual approach and programming paradigm. In addition, LabVIEW is a proprietary tool, which entails licensing costs and vendor lock-in [70].
- RoboDK: It is a robot simulation and programming software used in the field of industrial robotics. Its pros include its ability to simulate and program a wide range of robots from different manufacturers, allowing users to visualise and optimise automation processes before implementing them in the real world, as well as offering an intuitive and easy-to-use interface. However, this program features a commercial licence that increases the cost of the project [71].

Program	Cost (€/year)	Language	Experience
Arduino	Open source	C#	6 month
Python	Open source	Python	3 years
MATLAB	860	MATLAB	2 years
LabVIEW	536	LabVIEW	1 month
RoboDK	Open source	—	3 month

Table 4.1: Comparison between programs

4.1.2 PCB designing Software

The same criteria will be used to choose the designing software, but the programming language section will be omitted, and the difficulty of use will be added.

- KiKad: is an open-source software tool for creating electronic diagrams and designing printed circuit boards. The benefits of KiCad include its open-source nature, which allows users to access and compile the code on their own computers, its freeware nature, which makes it accessible to hobbyists, and its open-ended nature, with no restrictions on size or available features. One of the disadvantages of KiCad is its somewhat steep learning curve. In addition, some 3D component designs available online may be corrupted, making them difficult to import [72].
- Proteus: is an application for the execution of electronic equipment construction projects in all their stages: electronic schematic design, software programming, printed circuit board construction, simulation of the whole assembly, debugging, documentation and construction. Among the advantages, we can highlight its simulation capacity, which avoids the construction of new prototypes for testing, thus saving time and costs. In addition, its modular structure allows the acquisition of functionalities according to development needs, maximising investments in the tool. On the other hand, it is a programme with a steeper learning curve than most. [73].
- Eagle: is an electronic design automation software that allows PCB designers to seamlessly connect schematic diagrams, component placement, PCB routing and content from an extensive library. As advantages, it offers a neat interface, making it easy to create and edit PCB layouts. In addition, it has a wide variety of libraries available, allowing users to access components and symbols needed for their projects. In addition, it offers a free, full version, which means that users do not need to invest in a professional version to access all the features and functionality of the software. In contrast, simulation is done through Spice, which requires an external program to carry out the process, as well as 3D visualisation, which is more laborious than other programs [74].

Program	Cost (€/year)	Difficulty of use	Experience
KiKad	Open source	Medium	No experience
Proteus	200	Medium	No experience
Eagle	Open source	High	No experience

Table 4.2: Comparison between PCB designing programs

4.2 Hardware

As discussed above, the hardware analysis is based on the comparison of robotic arms and the choice of suitable electronics for the fabrication of the designed device.

4.2.1 Robotic arm

A robotic arm is a programmable device whose main functions and behaviour resemble those of a human arm. The operation of the robotic arm is almost that of a human arm and can be understood as a complete mechanism or part of a more complex robotic system. The parts of these mechanisms are interconnected by means of joints that can have a rotary motion or a linear displacement. The robotic arm consists of a mechanical

structure, transmitters, actuators, sensors, terminal elements and controller. The physical constitution of most industrial robots is similar to the parts and upper extremities of the human body [75]. In addition, it is important to introduce a robotic arm modality called *cobot*, which is a robot created to physically interact with humans in a collaborative work environment.

In the case of the latest models of the Da Vinci Surgical System, they feature four robotic arms with seven degrees of freedom each. Since we cannot count on one of these devices, we will have to opt for a single individualised arm from those available on the market. The best options are presented below.

4.2.1.1 UR5e

The UR5e is the medium-sized collaborative robot in the Universal Robots range. It features a payload of 5 kg and an action radius of 850 mm, which, together with its flexibility, makes it a perfect device for automating lightweight processing tasks. The overall purposes of the development of this collaborative robot are versatility and adaptability. The *cobot* UR5e is designed to integrate seamlessly into a wide range of applications.

Universal Robots' UR5e *cobot* is easy to program and configure, collaborative and safe, and represents the perfect balance between size and power. This easy-to-access technology allows users with no programming experience to quickly set up and operate the *cobots* using an intuitive interface with 3D visualisation [76]. On the other hand, if the desired functionalities exceed the possibilities of the software offered by the company, this system offers the option of controlling the robotic arm remotely. By means of an Ethernet connection, we can control it using previously created programs.

As for the robot's technical features, it is important to mention the six rotating joints that give it 6 degrees of freedom, as well as its high rotational speed (180°/s) and its low weight of only 20.6 kg. Regarding the price, it is close to 30.000€, which could be a disadvantage if the University of Barcelona did not have one and it had to be acquired.

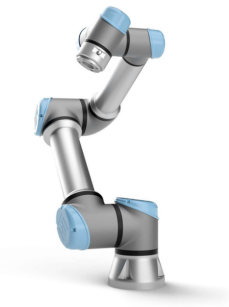


Figure 4.6: UR5e robot [6]

4.2.1.2 Kuka 210 robotic arm

The KUKA industrial robot is a robotic arm known for its load-carrying capacity and its flexibility to add additional axes. It has 6 joints which, like the UR5e robot, give it 6 degrees of freedom [77]. Currently there are a large number of models which are suitable depending on the function that needs to be performed. The KR 3 Agilus model from Kuka Arm, for example, is particularly suitable for moving small components and products in a minimum of space. It has a payload of 3 kg and a maximum reach of 541 mm [7]. This would be a good option to carry out the project, considering its cost of 20.000 €.



Figure 4.7: Kuka 210 robotic arm [7]

4.2.1.3 SWIFTITM CRB 1300 robotic arm

The SWIFTITM CRB 1300 robotic arm is a high-speed industrial cobot that combines industrial-grade performance and innovative safety, enabling a combination of collaborative safety, high performance and precision. In addition, it offers a maximum tool centre point speed of more than 6 metres per second, and boasts high accuracy. Besides these features, it has payload options from 0.5 kg to 11 kg, a weight of 75 kg and 6 degrees of freedom [78]. This robotic device has a total cost of 37,500 € [79].



Figure 4.8: SWIFTITM CRB 1300 robotic arm [8]

4.2.2 Quirurgic Gripper

To check the correct functioning of the software as well as the hardware components, we will connect the EndoWrist motor panel to a minimally invasive surgical forceps. The main idea is to use the original EndoWrist to maximise compatibility with the Da Vinci surgical device. However, other tools will also be analysed for better performance.

4.2.2.1 Virtual Incision by Vicarious Surgical

The Vicarious surgical system is a nimble, small tool that allows incisions as small as 1.2 cm, when most robotic forceps make them about 1.8 cm. In addition, it has great mobility, allowing movements in all directions, even backwards. The pivot centred within the abdomen is designed to allow the surgeon to work from any direction through any entry point. It also features immersive 3D visualisation and provides advanced stereoscopic images of the patient's anatomy and full responsiveness to the surgeon's movements. This system could be compatible with the Da Vinci [80].



Figure 4.9: Virtual Incision gripper [9]

4.2.2.2 Homis Surgical System by Memic

The Hominis Surgical System presents a concept unlike anything seen before, as it is the first and only FDA-cleared surgical robot to feature miniature humanoid-shaped arms with shoulder, elbow and wrist joints that allow for high dexterity and articulation. Multiple instruments can be inserted into the body through a single portal and the articulation offers optimal access and working angles. In addition, it requires a very small footprint and costs significantly less than conventional robotic systems [81].



Figure 4.10: Hominis gripper [10]

4.2.2.3 EndoWrist tool by Intuitive Surgical

As for the EndoWrist instruments, when these are assembled with the Da Vinci arm (or the UR5e in our case) they provide 7 degrees of freedom. These are: In out, Rotation, General Yaw, General Pitch, Roll 1 and Roll 2, and Pitch and Yaw.

The movements corresponding to the entry/exit of the abdomen (In out), the rotation of the tool (Rotation), and the rotation around the fulcrum point (General Pitch and General Yaw) are performed by the movement of the robotic arm. On the other hand, by controlling the tool tip, we find four additional movements, the opening/closing of the gripper (Roll 1 and Roll 2) and the exclusive turns of the tool tip (Pitch and Yaw). These degrees of freedom are controlled by a mechanism that consists of a part that is directly coupled to the da Vinci robotic arms and interfaces with the EndoWrist tool's rotation mechanisms, as seen in Figure 4.11. It will be the servomotors that, controlled by the user pen, will cause the gears of the system to rotate and move the tool tip [82].

Using this system, the EndoWrist features 180-degree swing and 360-degree rotation capabilities. Moreover, the EndoWrist line of instruments features a variety of specialised tip designs, including forceps, needles and scissors; monopolar and bipolar electrocautery instruments; scalpels and more. EndoWrist instruments are available in 5 and 8 mm diameters to meet surgeons' needs [83].

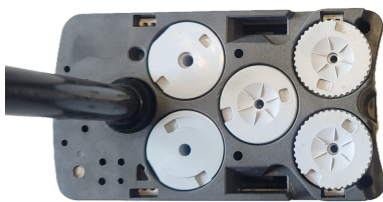


Figure 4.11: EndoWrist rotation mechanism

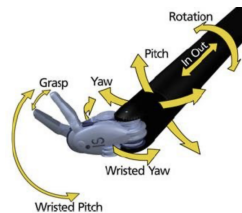


Figure 4.12: Tools movement [11]

4.2.3 Servomotor

The servomotor is one of the essential components of the project, as it is used to move the Endowrist gears. Since the gripper has four degrees of freedom (pitch, yaw, roll 1 and roll 2) and taking into account the structure of the EndoWrist, a total of four servomotors will be necessary. These will generate torque and speed depending on the current and voltage supplied. The torque applied must be sufficient for the grippers to have enough force to carry out the different actions. The possible options are presented below:

Servo Motor	Unit Cost (€)	Weight (g)	Torque (Kg/cm)	Rotation Range
TowerPro MG996R	10,22	55	10	180
Sunfounder SF180M	6	9	2,5	180
SG90	3	9	1,8	180

Table 4.3: Comparison between possible Servomotors

4.2.4 IMU sensor

The inertial measurement unit (IMU) is a very important component in inertial guidance systems. An IMU works by detecting motion, including motion type, speed and direction, using a combination of accelerometers and gyroscopes. The IMU can be selected based on its degrees of freedom as developed by the manufacturer, with 3,5,6 and 9 DOF models available [84]. This device will be in charge of registering the surgeon's hand movement in order to transfer it to the servomotors. Although there is a wide variety of these on the market, we have selected a list of those that we have found most interesting.

IMU	Unit cost (€)	Weight (g)	Power (V)	Degrees of Freedom
ICM-20948	13,2	0,7	1,71–3,6	9
MPU 9250	12,99	2,72	2,4–3,6	9
MPU 6050	5	2,1	2,4–3,6	6

Table 4.4: Comparison between possible IMUs

4.2.5 Microprocessor

Since the three elements of the project must be connected to each other wirelessly, it will be necessary to use a microprocessor, which is a programmable digital integrated circuit that has a wide range of functionalities. It is composed of a central processing unit (CPU), memories (ROM and RAM) and input and output lines (peripherals). Of the large number of microprocessors on the market, we will focus on those that offer Internet of Things (IoT) features, are small and have low power consumption.

Microprocessor	Unit cost (€)	Memory (kB)	Bluetooth	Operating voltage (V)
ESP32	11	520	Yes	2.2 - 3.6
STM32F4 Series	23	384	Yes	1.8–3.6
Atmega328P	13	2	Yes	1.8–5.5

Table 4.5: Comparison between possible Microprocessors

4.2.6 Vibration motor

To achieve haptic feedback, we need to select devices that let the surgeon feel the force applied to the patient's tissue. In robotics, there are two main feedback mechanisms: force feedback and vibrotactile feedback. As discussed in the section 6.1 Haptic Technology, the robot's design makes force feedback impractical, so we opt for a vibrotactile system using a buzzer and vibration motor.

As it is intended to monitor the force exerted during the operation, and not to activate the vibration device due to excessive force, which the computer's speaker already indicates, the motor will vibrate more with increased force on the tissue. Excessive vibration should be avoided to prevent hindering surgical activity and affecting the doctor's precision.

Vibration Motor	Unit cost (€)	Weight (g)	Voltage (V)
Sseed Studio 105020003	3	8.8	3–5
PWM Vibration Motor Module	6	16	3–5.3
WE-DA-159	1.2	2	3–5.3

Table 4.6: Comparison between possible Vibration Motors

4.3 Testing material

Once the design has been finalised, it will be necessary to test it to see if it is performing well. Therefore, a material that resembles the characteristics of the skin and perform a suturing procedure has to be selected. For this purpose, we have chosen to use three-layer suture pads made of silicone. This kid also offers Trusilk 75 cm needles, which are ideal for suturing. In addition, a material simulating the patient's outer skin will also be needed, into which a hole will be drilled to simulate a MIS port through which the EndoWrist will be inserted.

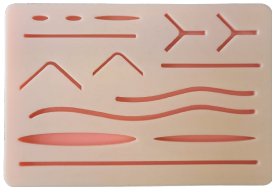


Figure 4.13: Testing system suture material



Figure 4.14: Outer skin material

4.4 Final election

Taking all of the above into account, a pre-selection of the materials to be used in the project can be made, which can change throughout the project if the situation requires it. In some cases, the characteristics of each element will be set aside, and the one with greater access will be chosen, as it is provided by the university. A good example of this could be the UR5e robotic arm, which is located in the Robotics Laboratory. The different elements selected are in Table 4.7, and an extended explanation of the different reasons for choosing them are presented below.

In terms of programming software, the Python, Arduino and RoboDK programs have been selected. The choice of Python is due to the wide experience available in this programme compared to the others. It is also compatible with RoboDK, which will make

our work much easier, using Virtual Studio Code as a working environment. As far as Arduino is concerned, it is one of the most widely used environments for the development of software for microprocessors, using the Arduino IDE platform. It contains a wide variety of libraries and a comfortable and accessible interface that will allow us to program the ESP32 easily.

KiKad has been chosen due to the experience and predisposition of the tutor Dr. Manel Puig i Vidal, since the student has no experience in this type of programme. After comparing the characteristics of the different environments proposed, it was decided to use Kikad as the PCB design software.

As for the robotic arm, as mentioned above, the University of Barcelona has a UR5e arm in the Robotics laboratory at the Faculty of Physics. Therefore, taking into account that its performance is considerably good, the use of this system has been chosen to carry out the project.

The chosen surgical gripper is the EndoWrist, more specifically the needle driver model. This is due, on the one hand, to the fact that the relationship with the Hospital Cl nic gives us the opportunity to obtain some units of this tool and, on the other hand, to the great variety of tools that Intuitive Surgical has developed. This provides great versatility to the project, since once the system is finished, it can be used with a multitude of tools according to the surgeon's needs.

The Sunfounder SF180M has been chosen among the different servomotor models due to its low cost and size and the torque it is capable of applying. It is powerful enough to generate the rotation of the EndoWrist's gears. Despite the initial idea of using the SG90 9G model, it does not have enough torque to rotate the EndoWrist gears, so a more powerful one has been chosen.

The IMU selected was the MPU 9250, as only this model directly provides the RPY angles thanks to its Digital Motion Processor (DMP), which can perform sensor fusion. The other models only provide rotational speeds and accelerations, and it is necessary to calculate the RPY angles from these values. In addition, it has a small size and weight and a great precision in angle reading.

Regarding the microprocessor, the ESP32 is an excellent choice for an IoT projects due to its combination of affordability, performance, and built-in Wi-Fi capabilities. Its integration of Wi-Fi enables seamless communication between modules, facilitating data transfer and remote control functionalities. Additionally, its low power consumption allows us to keep the controllers connected to an external battery with a very high autonomy.

Component	Item selected
Software	
Programming Software	Arduino, RoboDK, Python
Designing Software	FreeCAD
PCB Designing Software	KiKad
Hardware	
Robotic arm	UR5e
Quirurgic Gripper	EndoWrist
Servomotor	Sunfounder SF 180M
IMU sensor	MPU 9250
Microcontroller	ESP32
Vibration Motor	WE-DA-159

Table 4.7: Final selection of the components

Detail engineering

Having contextualised the proposal, presented the objectives and analysed what could be the best options, this section will describe the process that has been carried out throughout the implementation of the project.

In order to understand all the tasks we have carried out, these will be divided into four main blocks. On the one hand, the design of the Hardware that will form the Master-slave device will be presented. On the other hand, the development of the software responsible for improving the maneuverability of the surgical system will be addressed. Additionally, the improvements made in the assembly and mounting framework between the different components enhance the effectiveness and performance of the prototype will be reviewed. Finally, the development of the system that will allow the haptic feedback integration will be explained.

5.1 Hardware development

Before proceeding with the explanation of how this section has been realised, Figure 5.1 presents the diagram showing the connections between all the elements that make up the prototype.

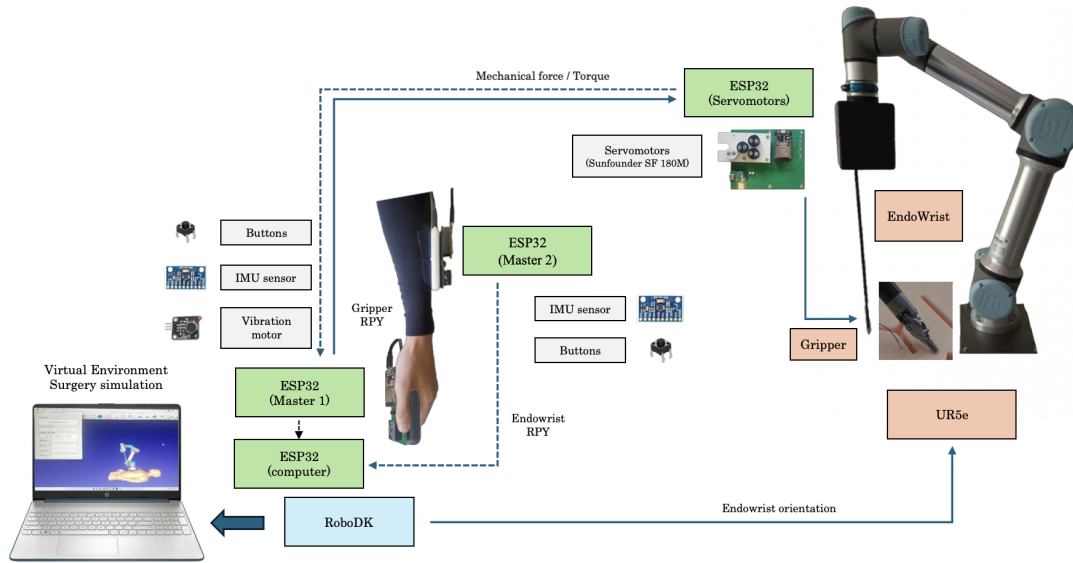


Figure 5.1: Prototype Connection Diagram

As shown in the diagram above, the system has four electronic modules whose ESP32 communicate with each other via Wifi, providing a fast and efficient data exchange. The MAC address of each of the microprocessors allows us to unequivocally identify each component, which guarantees secure transmission between the different modules.

In order to meet the objectives set out in the section 1.6 Objectives and to be able to test the correct operation of the device once the project has been completed, all the electronic modules are needed in order to offer all the desired functionalities. When the project was taken up, the prototype consisted of a single Master that managed the orientation of both the gripper and the Endowrist. Moreover, it did not incorporate any actuator that

could inform the surgeon of the force being exerted on the tissue. On the other hand, the EndoWrist Motion Panel had to be upgraded to improve its performance. Therefore, both PCBs had to be redesigned in order to incorporate all the elements that would allow us to obtain the desired functionalities. In addition, a new Master (Forearm Master) had to be added to control only the EndoWrist position, reserving the Gripper position for the Pen User Interface (Master 1).

5.1.1 Master's PCB

First of all, it is essential to clearly understand the function of each of the masters.

Pen User Interface, also known as Master 1, is the component that the surgeon will hold, simulating the tool used in the surgical field. It will have two main functions. The first will be to transmit the position and orientation of the surgeon's hand to both the EndoWrist Motion Panel and the Control and Supervision Module. In this way, we will be able to translate the position of the surgeon's wrist into a rotation of the four servomotors that will move the gears of the Endowrist, obtaining the same orientation in the gripper. In addition, the gripper will be opened and closed by controlling the buttons on the remote control. Furthermore, by sending the information to the computer, we will also be able to carry out the simulation, as well as generate a graphic interface with useful data for doctors. On the other hand, it will also transmit information received from the EndoWrist Motion Panel about the force the surgeon is applying to the tissue. This will be done by means of a vibration motor.

Forearm Master, also called Master 2, will be placed on the surgeon's forearm. It will transmit the position and orientation of the device to the computer, so that commands can be sent to the robotic arm. On the other hand, since the orientation of the gripper depends on the difference between the orientations of the forearm and the hand, it must also be sent to the servomotors so that they can take its orientation into account. In addition, by using buttons, the tool can be raised and lowered according to the requirements of the operation.

5.1.1.1 Position and Orientation reading

As reading the orientation of the Master is its main function, we need to know the process by which the values that characterise this position are obtained, which can be sent to the different devices of the prototype. For this purpose, the IMU is employed, which uses the Roll, Pitch and Yaw (RPY) angles to describe all possible orientations in three-dimensional space. Roll is the rotation around the X-axis, Pitch is the rotation around the Y-axis, and Yaw is the rotation around the Z-axis. The IMU measures linear acceleration and angular velocity in these three axes using accelerometers and gyroscopes, providing an accurate and stable estimate of the 3D object orientation using the RPY angles [85].

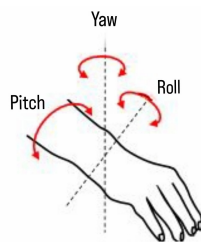


Figure 5.2: Roll, Pitch and Yaw angles



Figure 5.3: IMU sensor

Therefore, by holding the Master in such a way that the XYZ axes of the IMU, represented in Figure 5.3, coincide with those of the surgeon's hand, seen in Figure 5.2, the orientation of the hand can be represented very accurately.

5.1.1.2 Pen User Interface design

Once the functionalities this module should have are known, it is necessary to create a PCB that allows us to connect all the necessary elements. The Figures 5.4 and 5.5 show the PCB design and the final board.

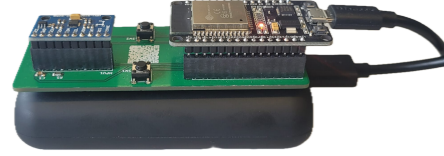
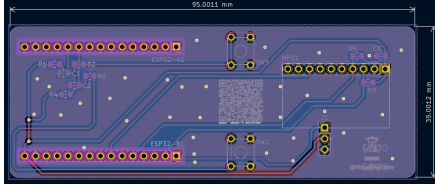


Figure 5.4: Pen User Interface PCB design

Figure 5.5: Pen User Interface final board

In order to be able to operate the device in a wireless and versatile way, the ESP32 will be powered using a Power Supply. In this way, the entire circuit will receive the necessary current and voltage to work correctly without the need to be connected to a computer or plug.

The module consists of two push buttons, an IMU sensor, a vibration motor and the ESP32. To provide a clearer understanding, the Table 5.1 lists all the connections established on the board. Since it is needed to process the data coming from the sensors (IMU) and buttons, as well as send a PWM to the actuator (vibrator motor), all components are connected to the ESP32.

Signal	Pin	Usage
IMU SCL	29	I2C serial clock
IMU SDA	26	I2C serial data
IMU INT	24	Interrupt
Vibration IN	30	PWM
Vibration VCC	16	Power Supply
SW1	10	GPIO
SW2	11	GPIO
GND	14, 17	Ground

Table 5.1: Pen Unser Interface PCB connections

In addition, for a more detailed analysis of the circuit, a schematic of the circuit can be found in the Appendix .7.

5.1.1.3 Forearm Master design

As for the Forearm Master, since it has very similar functionalities and requirements to the Pen User Interface, it presents a very similar design, although a little simpler as it does not require the inclusion of haptic technology (vibration motor). Figure 5.6 and 5.7 shows the PCB design and the final board.

In the same way, this device is connected to a Power Supply that offers autonomy and comfort. Moreover, the connections of this module are the same as those of Pen User Interface (Table 5.1), except for the vibration motor, which is not included in Forearm Master.

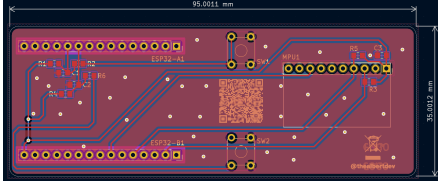


Figure 5.6: Forearm Master PCB design

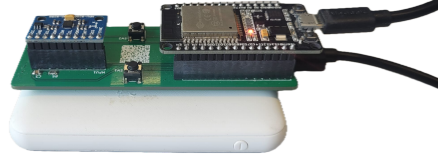


Figure 5.7: Forearm Master final board

5.1.2 EndoWrist Motion Panel PCB

As in the case of the Masters, it is also necessary to know what will be the functions of the servomotors in order to be able to design the PCB correctly. To begin with, they have to receive indications from the two masters in order to be able to adjust the rotation angle. We also need to send Pen User Interface the torque values of each of the servomotors, which will give us an idea of the force exerted on the tissue.

5.1.2.1 Torque reading technology

In order to know the force that is being exerted on the tissue or suture thread, we will calculate a value that will be proportional to the torque of the servomotor. This parameter will be the current, which we will read using Shunt resistors. These must have a small value (1.5Ω) so that the current can flow through it without a problem. The Shunt resistor generates a voltage proportional to the current flowing through it according to Ohm's Law:

$$V_{\text{shunt}} = I \cdot R_{\text{shunt}} \quad (5.1)$$

To measure the V_{shunt} , we will use the analogue-to-digital converter (ADC). The digitised value provided by the ADC, ADC_{value} , is converted into voltage by means of Equation 5.2. This equation allows us to convert a digitised value to an analogue voltage, which enables us to apply Ohm's Law to obtain the current.

$$V_{\text{adc}} = \frac{ADC_{\text{value}}}{2^n - 1} \times V_{\text{ref}} \quad (5.2)$$

Where 2^n is the resolution of the ADC (2^{12} in our case) and V_{ref} is the ADC reference voltage (3.3V in our case).

With this, we can obtain the current through the Shunt resistor by isolating the Ohms Equation.

$$I_{\text{pulsatile}} = \left(\frac{ADC_{\text{value}}}{2^n - 1} \times V_{\text{ref}} \right) \times \frac{1}{R_{\text{shunt}}} \quad (5.3)$$

However, the waveform of this current is pulsed, taking values between 0 and I_{max} due to the nature of the PWM signals used to control the servomotors. When the PWM signal is high, the current is I_{max} , and when the PWM signal is low, the current is 0. This results in a pulsating current waveform.

If we want to obtain an analogue value, we must integrate the signal over a period of time and then derive it again. This can be seen in Equation 5.4:

$$I_{\text{analogue}} = \frac{d}{dt} \left(\int_0^t I_{\text{pulsatile}}(\tau) d\tau \right) \quad (5.4)$$

As stated, the torque applied by the servomotor is proportional to the current passing through it. Then, knowing the value of the analogue current, we have a parameter that

allows us to quantify the torque that the servomotor is exerting. If we wanted to obtain the torque value, we would have to obtain the calibration factor.

5.1.2.2 EndoWrist Motion Panel board design

Once we know what functions it has to perform, how to calculate the torque and what components we need, we can proceed to the design of the board. To do this, we will need four servomotors, four $1.5\ \Omega$ resistors, an ESP32, a switch and a C-type PCB port. While the switch will allow us to decide when to power the circuit without the need to unplug the power cable, the type C port will let us connect a type C cable and obtain a 5V and 3A output. However, to achieve this purpose, two $5k\Omega$ resistors had to be placed between GND and both pins CC1 and CC2.

With this, the servo PCB design (Figure 5.8) and the final board (Figure 5.9) can be seen.

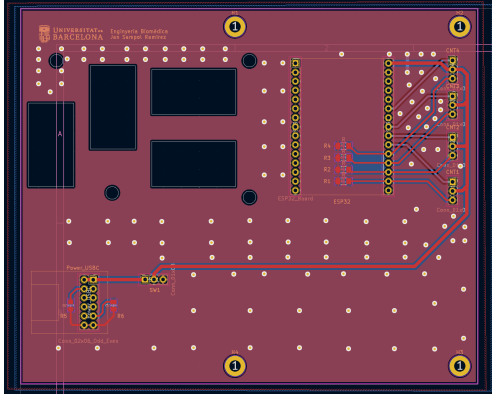


Figure 5.8: EndoWrist Motion Panel PCB design

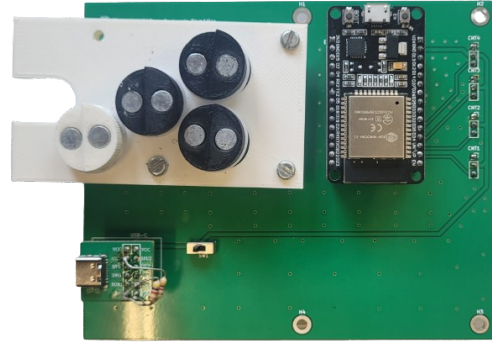


Figure 5.9: EndoWrist Motion Panel final board

As has been done with the connections on the Master board, all the connections that have been made on this board are presented below.

Signal	Pin	Usage
Servomotor Signal 1	32	PWM input
Servomotor Signal 2	33	PWM input
Servomotor Signal 3	25	PWM input
Servomotor Signal 4	27	PWM input
Servomotor GND 1	36	Analog Read
Servomotor GND 2	39	Analog Read
Servomotor GND 3	34	Analog Read
Servomotor GND 4	35	Analog Read
5V	15	Power Supply
GND	14,17	Ground

Table 5.2: EndoWrist Motion Panel PCB connections

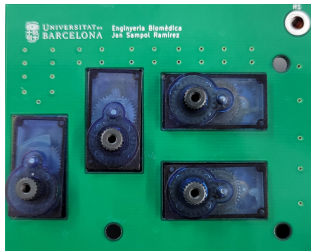
With these selected elements and the circuit in Figure 5.2, we have achieved a module capable of receiving the necessary data and translating it into a rotational force strong enough to cause the EndoWrist gears to rotate. As in the case of the Pen User Interface, the schematic can be found in the Appendix .8.

5.2 Assembling system

Having designed and manufactured the three PCB needed to carry out the project, it is now important to devise some structures that allows the assemblage of the four servomotors with the EndoWrist gears, and another one that permits the correct locations and manipulation of both Masters.

5.2.1 EndoWrist and Servomotors coupling

To understand the problem at hand, it is necessary to analyze the two structures being coupled. On the one hand, we find the four servomotors arranged as shown in Figure 5.10a, to whose gears the circle shaped piece in Figure 5.10b can be attached by means of a screw. The servomotors have been positioned in this way so that they could fit to the gears of the EndoWrist.



(a) Placement of the servomotors



(b) Part assembled to the servomotor

Figure 5.10: Servomotor's components

On the other hand, the components of the EndoWrist are the gears in Figure 5.11. These, when rotated, will allow the gripper to move. The two gears on the right will control Roll1 and Roll2, which will open and close the clamp. The middle gear drives the Pitch, while the top left gear controls the Yaw.

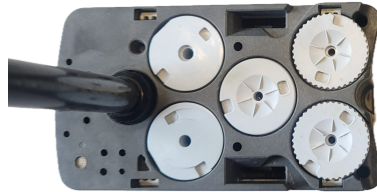


Figure 5.11: EndoWrist Gears

As evident, a structure is needed to link the servomotors to the EndoWrist. For this, some parts created with a 3D printer and designed by Daniel Alcaraz in [12] will be used. These can be seen in Figure 5.12.

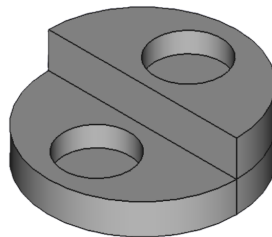


Figure 5.12: 3D pice [12]

Once the pieces have been printed, a disc shaped magnet has been placed in each of the holes that can be seen. One piece will be attached to each of the gears and to the discs of the four servomotors. However, these parts must be positioned in a specific manner. The gripper must be aligned in such a way that it is straight and, once in that position, hook the parts as shown in Figure 5.13. In the same way, the parts of the servomotors must be engaged with all servomotors with a 90° rotation, as shown in Figure 5.14.



Figure 5.13: 3D parts anchored to the EndoWrist



Figure 5.14: 3D parts anchored to the servomotors

This system is designed in such a way that when the two elements are put together, the pieces fit and the magnets hold them together. Its shape allows us to rotate the servomotor part and, without displacement, generate the same rotation on the EndoWrist part. The final system can be seen in Figure 5.15.

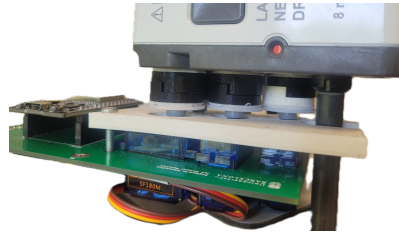


Figure 5.15: Coupled final design

5.2.2 Masters subjection system

Since surgery is a procedure that can last for a long period of time, a system that provides the greatest possible comfort for the surgeon must be developed. As for the positioning of the Pen User Interface, it is not necessary to design any structure to hold it, as it will be the surgeon himself who will hold it by hand, as a suturing tool.

However, the Forearm Master's grip will not be as straightforward. It has to remain fixed at a given position of the forearm so that it can reliably transmit the orientation of the arm. For this purpose, a mesh armband has been designed which contains a Velcro section to attach the Master to. In this way the surgeon can put on the comfortable cuff and then insert the Master into the indicated section. Figure 5.16 and 5.17 shows the final design of the fixture model.



Figure 5.16: Bracelet with Velcro section



Figure 5.17: Master fixation final design

5.3 Software development

Once we have both the devices that make up the Master-slave prototype and, moreover, the assembly structure, we can go on to develop the software that will allow us to make all the hardware we have designed to meet all the requirements that we had set.

In this case, there will be two different sections. Firstly, the software uploaded to each of the prototype modules will be found, which will be programmed with the Arduino IDE. Secondly, we are going to create a Python program that, on the one hand, will allow us to run a simulation in RoboDK and, moreover, will send the necessary commands to the UR5e robot.

5.3.1 Arduino code

Although the previous sections have shown how the modules communicate with each other, the following is a flow of data that allows all devices to use the data they receive and update it to send it to the other devices. This will help a lot in understanding the code.

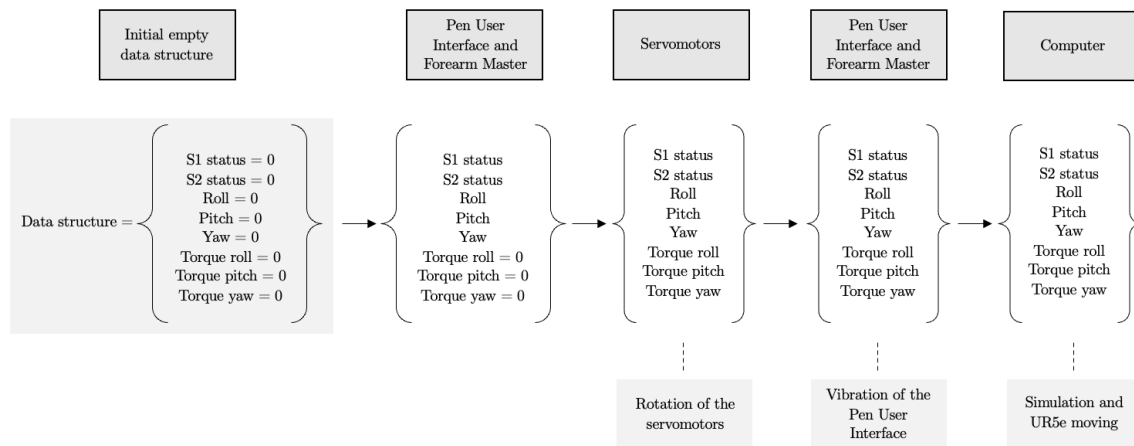


Figure 5.18: Data flow through the modules

As can be seen, we start from an empty data structure with the variables present in Figure 6.9. To begin with, both Masters send the RPY values to the EndoWrist Motion Panel, and the Pen User Interface also sends the S1 and S2 statuses. Once this data reaches the EndoWrist Motion Panel, the RPY values will be used to rotate each of the servomotors as will be explained later, and the Boolean S1 and S2 will be used to open and close the gripper. Then, with the torques of the four servomotors read, the data structure will be sent back to the Pen User Interface, which will use the torques to generate a vibration using the vibration motor. Finally, all the data is sent to the to carry out the simulation and move the UR5e robot.

Although it may seem a slow process with many steps, the Wifi communication used by the ESP32 is very fast and secure, so we can obtain a fluid communication and with real time variables updating.

5.3.1.1 Pen User Interface Code

The Pen User Interface is one of the most important elements of the prototype, as it not only reads the orientation of the surgeon's wrist, but also provides haptic sensitivity to the user.

Since robotic surgery is a high-precision procedure, we must avoid large, jerky movements at all costs. To do this, we must generate a programme that allows us to filter out these movements and stop the gripper if they are detected. To do this, it has been developed a code in which, if a difference of more than 10° is detected between two consecutive readings in any of the angles, the angle that has undergone the abrupt movement remains fixed at the value before it. To regain mobility on this axis, the master must be positioned at the angle at which it has been blocked, with the help of the graphic interface that indicates which angle this is. Below is a short code fragment that performs the action explained in the Roll angle. The complete commented code can be found the Appendix .3.

```

1  NewValueRoll=rpw[0];
2  valRoll=abs(NewValueRoll-OldValueRoll);
3  if (valRoll>10 && valRoll<350) {
4      roll=OldValueRoll;}
5  else {
6      roll=NewValueRoll;}
7  OldValueRoll=roll;
```

Listing 5.1: Angle protection code

As for the part concerning the vibration motor, it will be explained in the section 5.4 Haptic feedback system implementation.

5.3.1.2 Forearm Master Code

The Forearm Master code mirrors the structure of the Pen User Interface. It sends data to the EndoWrist Motion Panel and the Control and Supervision Module but doesn't receive torque values due to lacking a haptic system. It directly sends S1 and S2 values to the computer for controlling the EndoWrist's vertical movements with the UR5e arm. Moreover, this code also incorporates a safety programme against sudden movements, in order to prevent the robotic arm from injuring the patient. The complete code can be found in Appendix .4.

5.3.1.3 EndoWrist Motion Panel Code

The EndoWrist Motion Panel code can be divided into two sections. On one side we find the control of the rotation of the servomotors and, on the other, the calculation of the torque of each one of them.

When we talk about rotation control, the safety loop that is executed once the programme is started has to be mention. After having positioned the four servomotors at 90° , they remain static until the Pen User Interface has not been aligned with $RPY = 0$. In this way, any kind of undesired movement before starting the procedure are avoided. Once aligned, the servomotors start to rotate and the surgeon can control the gripper.

Since the RPY readings of the IMU range from 0° to 360° , and it is desired to have a movements in the range $[-90, 90]$, the conversion shown in Listing 5.2 has to be performed. This allows us to obtain positive angles up to 180 and negative angles up to -180 of both Master 1 and 2 readings.

```

1  if (angle > 180 && angle <= 360) { // Re-escape Master1 values
2      angle = angle - 360;}
3
4  if (angle2 > 180 && angle2 <= 360) { // Re-escape Master2 values
5      angle2 = angle2 - 360;}
6
```

Listing 5.2: RPY angles re-escaping code

Once the angles of both two Masters are on the scale it is needed, the difference between them has to be computed. This is because the movement of the gripper does not depend exclusively on the orientation of the Pen User Interface, but on the relative orientation of the Pen User Interface with respect to the Forearm Master. If this were not taken into account, a rotation of, for example, 45° of the forearm in Yaw, keeping the wrist aligned with it, would generate a total rotation of 90° of the gripper. This is because, being anchored to the EndoWrist, and the EndoWrist to the UR5e, the gripper would rotate the same 45° of the robotic arm and, in addition, another 45° reading the orientation of the Pen User Interface, generating a total rotation of 90° . When applying the angle difference, this same movement, on detecting a 0° difference between both masters, would only generate the 45° rotation induced by the robot, keeping the servomotors and, therefore, the gripper aligned with the EndoWrist.

To this difference, 90° must be added, which will allow the servomotor to remain in the centre when the reading is 0. Furthermore, in order to limit the movement to a range of 180° , avoiding sudden and unexpected rotations, the movement of the servomotors will be limited to an interval between -90 and 90° of the angle difference. The rotation code can be seen in Listing 5.3.

```

1  angle_rel = angle2 - angle; // Compute the angle difference
2  angle_gir = angle_rel + 90; // Add 90 degrees to center the servo
3
4  if ( -90 < angle_rel < 90){ // Bounding the movement
5      servo_angle.write(angle_gir);} // Rotate the servomotor

```

Listing 5.3: Servomotor rotation code

This, applied to the four servomotors, allows for a three-dimensional movement of the gripper. However, it is necessary to control the state of the gripper (open or closed) by using the Pen User Interface buttons S1 and S2. To do this, it will be establish that while S1 is held down, the Roll1 will increase the value of its reading by 30 degrees. In this way, by keeping Roll2 unchanged, the gripper will be opened. When the button is released, both readings are equalised and the gripper closes again. On the other hand, when S2 is pressed, 15 degrees are subtracted from the Roll1 reading, causing one gripper to collide with the other. This allows greater force to be exerted on the object being clamped, making the suturing process safer and more reliable.

As for the second section of the code, the calculation of the torque, the steps explained in the subsection 5.1.2.1 Torque Reading Technology has to be followed. In this way, the analogue value of the current passing through the shunt resistor is obtained, which is proportional to the torque exerted by the motor. The complete code can be found in Appendix .5.

5.3.1.4 Control and Supervision Module Code

The Control and Supervision Module receives the data from the Pen User Interface and the Forearm Master, so that the laptop can read them via the port and carry out both the simulation in RoboDK and the movement of the UR5e robot.

5.3.2 Python code

The Python code will be the one that, reading the data from the port sent by the Computer module, executes the simulation in RoboDK and transmits the commands to the robot. The code is divided into several sections, which makes it much more convenient to handle and easier to understand. First, there is a block where a connection is established with the

UR5e robot, provided that it is available, as well as configuring all the necessary parameters to run the simulation.

This programme is characterised by the presence of several processes running in parallel. This allows us to carry out several tasks at the same time without them interfering with each other, thus obtaining a fluid programme with no downtime. For this, the Python's threading functionality is going to be used. This allows the concurrent execution of multiple threads within a single process. Each thread represents an independent sequence of execution that can run in parallel with other threads. In addition, threads are especially useful for I/O-bound operations, which are characterised by data input and output, and are the ones that will be used in the project.

Hence, we find the reading of data from the serial port, which allows us to update them in real time independently of the delays that may occur in the other processes. On the other hand, there is the generation of the graphic interface, for which the TKinter library has been used. Finally, the last process to be carried out is the suture process itself.

5.3.2.1 TKinter root

The graphic interface is an instrumental tool for the surgeon to perform the procedure in a user-friendly manner. It consists of two screens that display the orientation of both the Pen User interface and the Forearm Master, allowing the surgeon to know when the safety mechanism has been triggered by one of the angles remaining fixed at a value. In addition, it also shows the state in which the robot is displayed, whether it is opening or closing the gripper, in a motion mode, or moving up or down with respect to the Z-axis of the EndoWrist.

Moreover, we can find four horizontal bars in which the torque exerted by each of the servomotors is represented in real time. When any of the servomotors exerts excessive force on the tissue, the bar turns red and an audible alarm sounds saying "Threshold reached in Pitch", depending on the angle at which it is reached. In Figure 5.19 the TKinter root is shown.

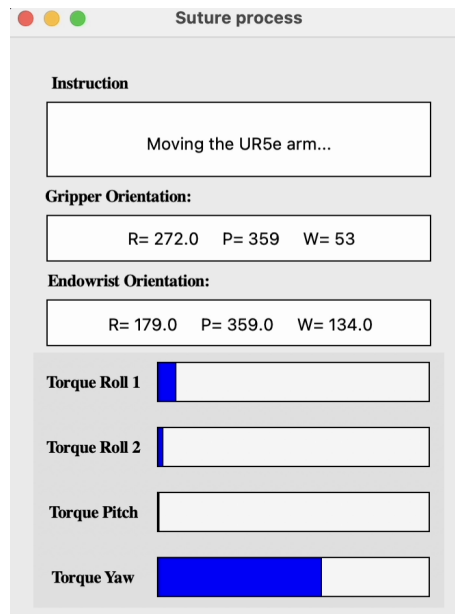


Figure 5.19: Graphical interface

5.3.2.2 Suture process

This threading is where both the sending of commands to the UR5e and the simulation in RoboDK takes place. As with the movement of the servomotors, in this case there is also a safety loop to avoid starting the movement before the Forearm Master is properly aligned with the initial position.

On the one hand, we find the simulation in RoboDK. As long as no button is pressed, the program will read the RPY values of both masters and, after reading the Tool Center Point (TCP) position of both the EndoWrist and the gripper in the RoboDK, it will translate them into POSE through the multiplications of Listing 5.4.

TCP is defined as the reference point used for precise positioning of the robot in applications that require defined targets in Cartesian space. The TCP is set as a transformation from the robot mounting flange and is crucial to ensure accuracy and efficiency in robot programming and operation. In our case, since we want the EndoWrist to rotate around the fulcrum point, where the tool will contact the skin, we will set the TCP coordinates to (8, 2.5, 60) cm, due to the geometry of the EndoWrist.

```

1 endowrist_pose = transl(Xr, Yr, Zr)*rotz(W2)*roty(P2)*rotx(R2)
2 gripper_pose = transl(Xg, Yg, Zg)*rotx(pi/2)*rotz(W_gripper_rel)
   *roty(P_gripper_rel)*rotx(R_gripper_rel)

```

Listing 5.4: EndoWrist and Gripper POSE code

Where X_r , Y_r , Z_r are the EndoWrist TCP coordinates, X_g , Y_g , Z_g are the gripper frame coordinates and $R_gripper_rel$, $Y_gripper_rel$, $W_gripper_rel$ are the difference between the angles received from the Pen User Interface and the Forearm Master.

The fact that, in both cases, they are a fixed frames, means that the order of multiplication of the rotations along the axes is performed in reverse order of the graphic rotations. In addition, in the case of the gripper frame, we first apply a 90° rotation along the X-axis to correct the 90° that we have rotated the EndoWrist TCP in the initial program parameters. This must be done since the EndoWrist TCP is configured as the parent of the gripper frame.

Moreover, the S1 and S2 buttons on the Forearm Master allow the EndoWrist to be raised or lowered, by using a linear movement, a translation. Meanwhile, the S1 and S2 buttons of the Pen User Interface nest and un-nest the needle item to the Gripper frame, simulating the opening and closing of the tool. In Figure 5.20 we can see the simulation of the UR5e arm in the RoboDK program.

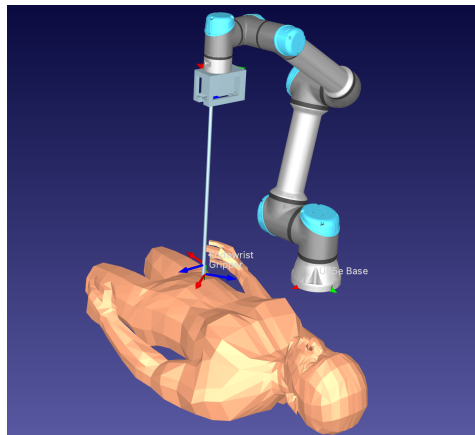


Figure 5.20: RoboDK simulation

In the context of sending commands to the UR5e robot, we have used sockets as a means of communication. Sockets are an interface that facilitates communication between processes, both on the same computer and on different devices over a network. Using this technology, we establish a connection between the computer from which we issue the commands and the UR5e robotic arm, connected via Ethernet, which allows us to send and receive data bidirectionally, efficiently and in real time. Data is sent every 0.4 seconds, thus allowing a constant update of the arm orientation, and a smooth movement of it.

However, this type of communication works in a different way from the one explained above. In this case, we must transform the RPY angles read from Forearm Master to Angle Vector. In order to do this conversion in a simple way we must use the Spatialmath Python library. The code can be consulted in Appendices.

Nevertheless, if the RPY angles are converted to Angle Vector and then sent to the robot, the movements performed with the Forearm Master would not correspond to those carried out by the robotic arm. This is because, in order to achieve a similarity between the movements, we must make a base change to match the frames of the IMU and the robot flange, as exposed in Equation 5.5.

$$\begin{pmatrix} \text{roll}_{\text{UR5e}} \\ \text{pitch}_{\text{UR5e}} \\ \text{yaw}_{\text{UR5e}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \text{roll}_{\text{IMU}} \\ \text{pitch}_{\text{IMU}} \\ \text{yaw}_{\text{IMU}} \end{pmatrix} \quad (5.5)$$

When performing matrix multiplication, we obtain the equality of the Equation 5.6. This allows us to express the orientation of the Forearm Master in the coordinates of the robotic arm flange frame, thus generating exactly the same movement.

$$\begin{pmatrix} \text{roll}_{\text{UR5e}} \\ \text{pitch}_{\text{UR5e}} \\ \text{yaw}_{\text{UR5e}} \end{pmatrix} = \begin{pmatrix} \text{pitch}_{\text{IMU}} \\ -\text{roll}_{\text{IMU}} \\ \text{yaw}_{\text{IMU}} \end{pmatrix} \quad (5.6)$$

The spatial orientation of each of the frames is shown in Figure 5.21. In this way, the correlation between the RPY axes of each of the frames can be seen graphically.

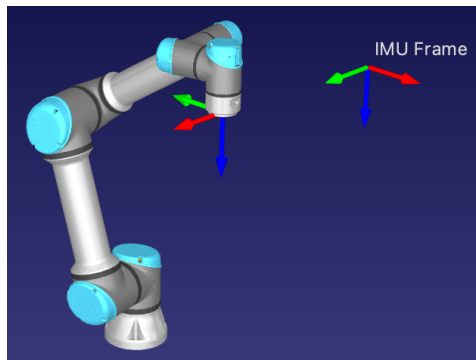


Figure 5.21: Graphical representation of both IMU and UR5e frames

Then, by making this base change and then transforming the RPY angles into Angle Vector, we can send a command to the robot to move to the desired position and orientation. The complete code of both the suture process and the tkinter sections can be found in Appendix .1 and .2 respectively.

5.4 Haptic feedback system implementation

As for the haptic technology we have introduced in our prototype, it can be found in both the EndoWrist motion panel and the Pen User interface. Moreover, the Python code has been programmed to display a graphical representation of this in real time. As we have seen in the subsection 5.1.2.1 Torque Reading Technology, we have developed a system to obtain a value proportional to the torque that the servomotor is exerting. This gives us an idea of the force being applied to the tissue. These values are sent to the Pen User Interface, so that the module generates a tactile sensation to the surgeon of the magnitude of force being performed.

To achieve this, it is necessary to convert the current values to a range of 0 to 255. This rescaling process is crucial to transform the data into a PWM signal. The PWM signal is used to control the vibrating motor, modulating its vibration depending on the intensity of the applied force. The higher the force, the higher the value of the signal within the range of 0 to 255, which in turn increases the pulse width of the PWM signal and causes the motor to vibrate more intensely. In this way, the motor is made to vibrate proportionally to the detected force, allowing an appropriate tactile response.

Experimental validation

Once we have designed, manufactured, programmed and assembled the entire robotic surgery system, we can analyze its operation. We will check, on the one hand, the performance of the haptic feedback technology and, on the other hand, the maneuverability of the robotic arm.

6.1 Haptic feedback system performance

Once the haptic feedback system has been installed, we must check that it provides accurate information according to the force applied to the tissue with the gripper. To do this, we will analyze the results by steps. First, we will verify and check the readings of the current intensity passing through the shunt resistor. And, subsequently, we can proceed to study both the PWM values received by the vibrating motor and the information displayed on the graphical interface.

To do the first analysis, we will verify the operation of the Pitch servomotor, since the variance between the four servomotors is negligible. A rotation of the servomotor will be performed under two conditions. In one, the EndoWrist tool will be assembled and allowed to move freely as a reaction to the movement of the motors. In the second case, the movement of the tool will be subjected to a force when colliding with the tissue. The differences between the two cases should then be analysed and whether they are significant enough for this technology to be considered useful in transmitting the force being applied.

However, even if we refer to the first case as no obstacle, the EndoWrist itself is already a resistance that hinders the rotation of the servomotor. To get an idea of the amount of additional current that the servomotors must receive to reach the desired position by rotating the EndoWrist gears, Figures 6.1 and 6.2 should be studied.

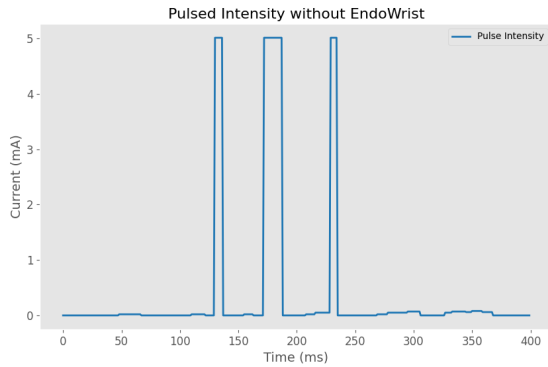


Figure 6.1: Pulse intensity without EndoWrist

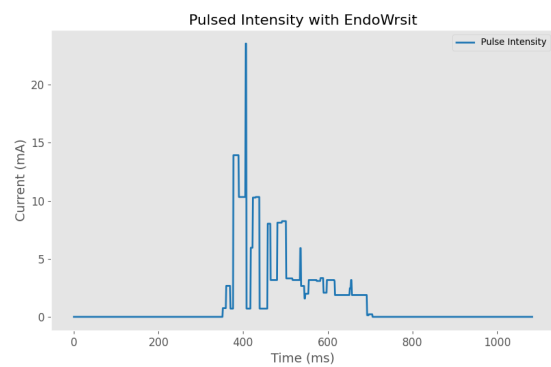


Figure 6.2: Pulse intensity with the free EndoWrist

As can be seen, when the servomotors can move freely, they need only three current pulses to reach the desired position. In contrast, when the tool is inserted, a larger amount of current is required to carry out the same movement. On the one hand, the number of pulses increases and, on the other, the intensity of these pulses also varies. After an initial peak of increased intensity, as the servomotor attempts to reach the position at its maximum speed, a series of peaks appear which decrease over time.

After a brief introduction of how the current received by the servomotors varies with the introduction of loads, the performance of the haptic system can be analysed. As discussed in the section 6.1 Haptic feedback implementation, the first measurement to be made is the pulsating current that is sent to the servomotors via a PWM signal. The representation of this can be seen in Figures 6.3 and 6.4.

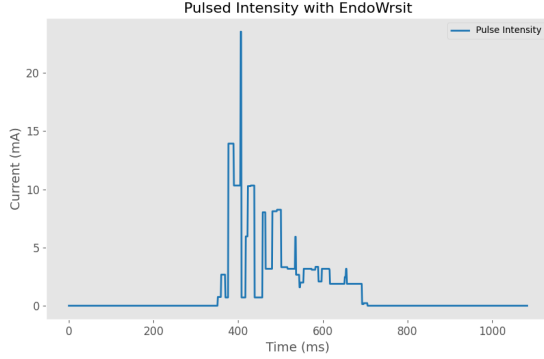


Figure 6.3: Pulse intensity without obstacle

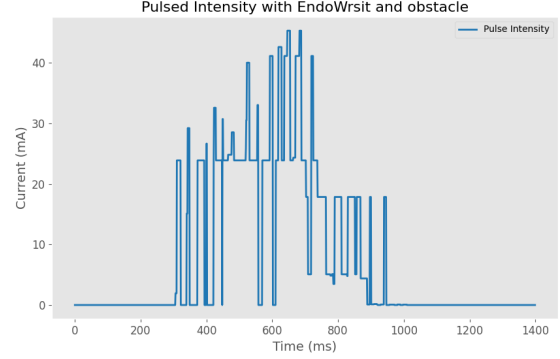


Figure 6.4: Pulse intensity with obstacle

As can be seen in the graphs above, the servomotors receive a pulsed current that is difficult to quantify. Furthermore, there is a notable difference between the number of peaks that the servomotor receives and the shape of these, depending on the load that it supports during the movement. In the case of the free EndoWrsit, the servomotors need less current to carry out the movement, so the number of pulses that reach it are fewer than in the case of the EndoWrist that collides with the tissue. When talking about the EndoWrist under load, it is worth noting the large number of current peaks that occur during movement. These are caused by the servomotor's inability to reach the desired position.

However, we need a magnitude that allows us to quantify more accurately the force being exerted. To do this, we are going to integrate the current reaching the servomotor in a given period of time, thus obtaining the electrical charge. The two cases can be seen in Figures 6.5 and 6.6.

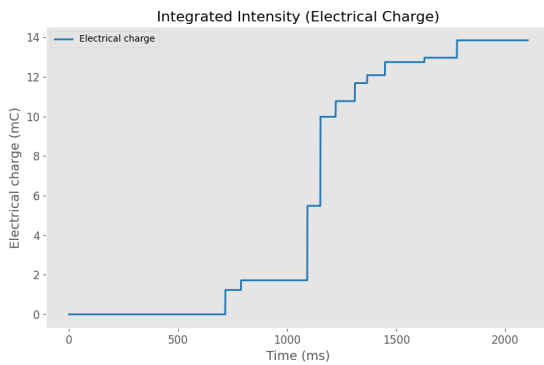


Figure 6.5: Electrical charge without obstacle

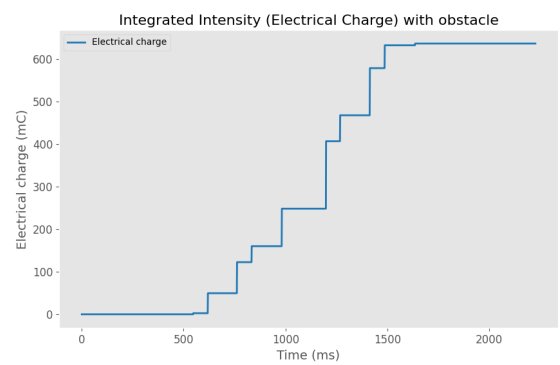


Figure 6.6: Electrical charge with obstacle

By studying the electrical charge, very significant differences are found that allow us to identify the force being used. As can be seen, the magnitude is multiplied by 40 when an obstacle hinders the rotation of the servomotor. Nevertheless, this does not allow us to know the real time force being exerted, since it represents the sum of the intensity that is provided to the servomotor during a given time. Therefore, a stationary region on the graph indicates that the servomotor is not exerting any force during that period.

To solve this problem, the signal obtained must be derivated, thus generating a representation proportional to the force being applied. Figures 6.7 and 6.8 show this new magnitude.

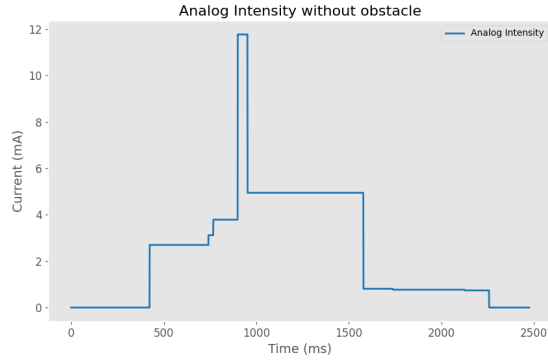


Figure 6.7: Analog intensity without obstacle

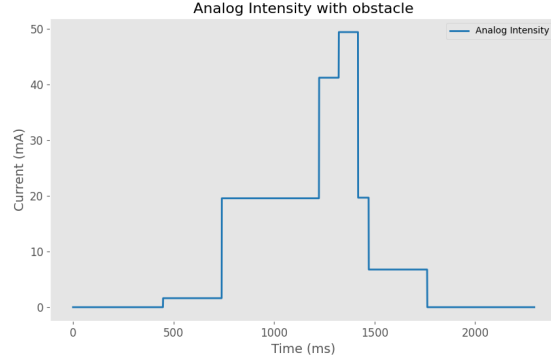


Figure 6.8: Analog intensity with obstacle

As can be seen, both graphs have a similar trend. When the servomotor starts rotating, it exerts a torque that allows the EndoWrist gear to move, with or without resistance, which can be reflected in a rise in magnitude. On the other hand, when the movement stops, the value returns to 0. In this case, the force applied by the servomotors is reflected in real time. In addition, there is a noticeable difference between when the EndoWrist is moving freely and when it is resisted. The intensity is multiplied by 4 in this case. If the force exerted were greater, the difference would be even greater, which makes it possible to differentiate quite accurately the intensity of the current and thus the force being exerted.

Once we have verified that the haptic technology is working correctly, we can proceed to verify the correct functioning of the graphic interface, which shows us in real time a value proportional to the torque. Since these values change depending on the force that the servomotors have to carry out, a video has been inserted where the movement of the EndoWrist is shown together with the graphic interface representing the force that it carries out on the tissue.

Click [here](#) to be directed to the video showing the performance of the graphical interface.

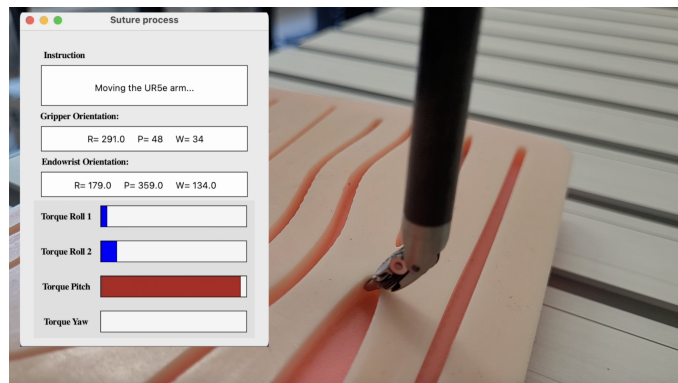


Figure 6.9: Torque representation in the graphical interface

We can see how the technology we have developed detects the force exerted by the EndoWrist on the tissue and, in addition, provides both a visual and audible alarm to the surgeon when the tissue safety threshold is exceeded.

6.2 Robot maneuverability

After carrying out several attempts of the suturing process using the surgical system developed throughout the project, we have been able to obtain a good performance of the system. Using both Masters, we have been able to control the orientation of both the EndoWrist tool, by means of the movements of the UR5e arm and the gripper. Below is a link in which you can see the operation of the final prototype.

Click [here](#) to be directed to the video showing the performance of the system.

In addition, Figure 6.10 shows the final device.



Figure 6.10: Final design device

Analysing the movements of both the robotic arm and the gripper in relation to the movements carried out with the Forearm Master and the Pen Unser Interface, it is possible to perceive a high precision in transferring the orientation of the surgeon's arm to the surgical system.

In addition, intermittent movement of the robotic arm can be identified. This is due to the communication established between the computer and the UR5e, via sockets. As discussed in section 5.3.2.2 Suture process, a command containing both the orientation and position of the Forearm Master is sent every 0.4 seconds, so the arm moves in 0.4 seconds to the desired position and then resumes movement.

Execution schedule

Once the project has been contextualised and the various activities to be carried out have been presented, it is necessary to organise and plan these activities in order to complete the project in the shortest possible time and with as few unforeseen events as possible.

7.1 WBS (Work Breakdown Structure)

First of all, we must hierarchically decompose the project in order to facilitate the identification of the different elements and deliverables of the work, thus making it easier to organise and coordinate throughout the project.

Below is a breakdown of the project activities, which are grouped into six different sections according to the nature of each of them.

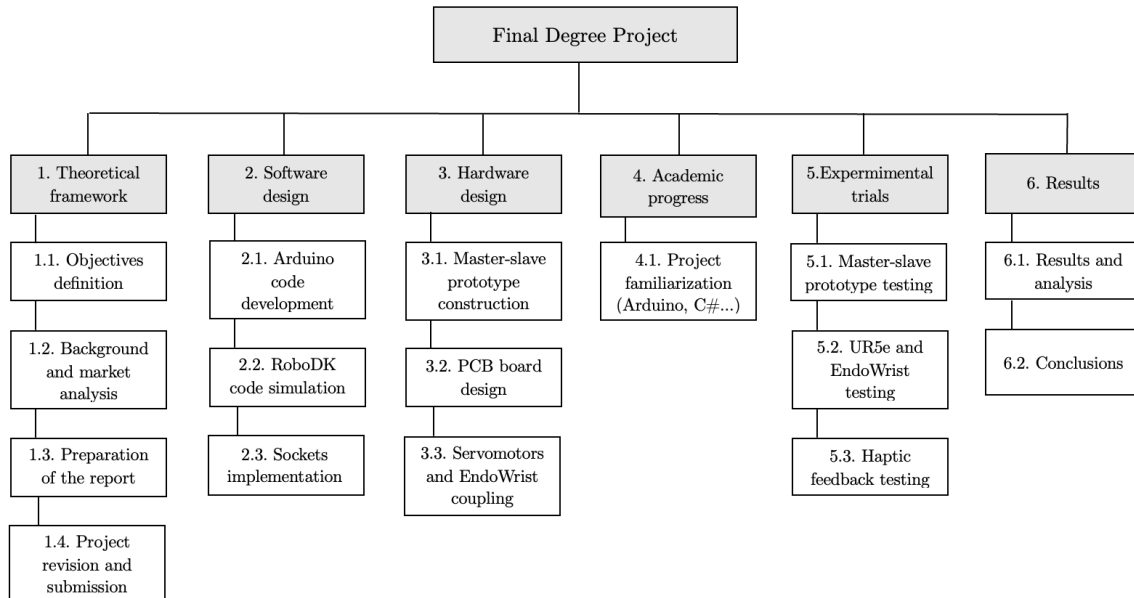


Figure 7.1: WBS diagram of the project

7.2 WBS Dictionary

In the following section we present all the necessary information on the various activities that make up the WBS. The WBS code is presented, as well as the name of the activity itself and a brief description of it. On the other hand, the acceptance criteria, the expected deliverable and the resources that have been used are also shown. Finally, we introduce the estimated cost, together with the time needed to carry out the activity.

WBS code	1.1
Name	Objectives definition
Description	Establish the objectives of the project, both the main ones and the desirable ones.
Estimated cost (€)	140 (student) + 60 (tutor)
Estimated duration (h)	7 (student) + 2 (tutor)
Deadline	17/01/2024

Table 7.1: "Objectives Definition" entry in the WBS Dictionary

WBS code	1.2
Name	Background and market analysis
Description	In-depth study on the evolution of robotic surgery from its origins to the present day, in addition to the analysis of future trends and the current market situation.
Estimated cost (€)	600 (student) + 120 (tutor)
Estimated duration (h)	30 (student) + 4 (tutor)
Deadline	07/02/2024

Table 7.2: "Background and market analysis" entry in the WBS Dictionary

WBS code	1.3
Name	Preparation of the report
Description	This task will be transversal throughout the entire work. It will consist of the preparation of the report to be delivered at the end of the project.
Estimated cost (€)	1300 (student) + 330 (tutor)
Estimated duration (h)	65 (student) + 11 (tutor)
Deadline	22/05/2024

Table 7.3: "Preparation of the report" entry in the WBS Dictionary

WBS code	1.4
Name	Project revision and submission
Description	Check the structure, grammar, spelling, correspondence of references, results and conclusions of the work to submit the final version.
Estimated cost (€)	180 (student) + 300 (tutor)
Estimated duration (h)	9 (student) + 10 (tutor)
Deadline	05/06/2024

Table 7.4: "Project revision" entry in the WBS Dictionary

WBS code	2.1
Name	Arduino code development
Description	Modification of the Servomotor's code and development of the code for both Master 1 and Master 2, to allow communication with the Computer and the servomotors, as well as the correct reading of the RPY angles.
Estimated cost (€)	500 (student) + 300 (tutor)
Estimated duration (h)	25 (student) + 10 (tutor)
Deadline	28/02/2024

Table 7.5: "Arduino code development" entry in the WBS Dictionary

WBS code	2.2
Name	RoboDK code simulation
Description	Transfer of the Arduino code to the RoboDK Software to check the correct operation of this, and the relevant modifications of the Python code to improve the functionalities of the programme.
Estimated cost (€)	500 (student) + 330 (tutor)
Estimated duration (h)	25 (student) + 11 (tutor)
Deadline	27/3/2024

Table 7.6: "RoboDK code simulation" entry in the WBS Dictionary

WBS code	2.3
Name	Sockets implementation
Description	Translated the functional python program of the RoboDK to the Sockets communication system in order to improve the communication between the Master devices, the Computer and the UR5e robot.
Estimated cost (€)	300 (student) + 180 (tutor)
Estimated duration (h)	15 (student) + 6 (tutor)
Deadline	10/04/2024

Table 7.7: "Sockets implementation" entry in the WBS Dictionary

WBS code	3.1
Name	Construction of Master-slave prototype
Description	Design of a prototype that facilitates the work and usability of the controls, in the same way that allows us to obtain better results.
Estimated cost (€)	300 (student) + 150 (tutor)
Estimated duration (h)	15 (student) + 5 (tutor)
Deadline	20/03/2024

Table 7.8: "Construction of Master-slave prototype" entry in the WBS Dictionary

WBS code	3.2
Name	PCB board design
Description	Design of a board for the control of the servomotors, which has an ESP32 microprocessor that allows communication with the other modules.
Estimated cost (€)	240 (student) + 180 (tutor)
Estimated duration (h)	12 (student) + 6 (tutor)
Deadline	25/02/2024

Table 7.9: "PCB board design" entry in the WBS Dictionary

WBS code	3.3
Name	Servomotors and EndoWrist coupling
Description	Refinement of the current coupling system between the servomotor plate and the EndoWrist to improve the motion transmission between the two.
Estimated cost (€)	240 (student) + 120 (tutor)
Estimated duration (h)	12 (student) + 4 (tutor)
Deadline	17/04/2024

Table 7.10: "Servomotors and EndoWrist coupling" entry in the WBS Dictionary

WBS code	4.1
Name	Project familiarization
Description	To understand the operation of the different programs that will be used throughout the project, as well as the different hardware components.
Estimated cost (€)	300 (student) + 120 (tutor)
Estimated duration (h)	15 (student) + 4 (tutor)
Deadline	31/01/2024

Table 7.11: "Project familiarization" entry in the WBS Dictionary

WBS code	5.1
Name	Master-slave prototype testing
Description	Check if the communication between the Masters, the computer and the servo board is as desired.
Estimated cost (€)	360 (student) + 180 (tutor)
Estimated duration (h)	18 (student) + 6 (tutor)
Deadline	03/04/2024

Table 7.12: "Master-slave prototype testing" entry in the WBS Dictionary

WBS code	5.2
Name	UR5e and Endowrist testing
Description	Checking the correct functioning and mobility of the UR5e arm, as well as the Endowrist gripper once assembled with the robot.
Estimated cost (€)	400 (student) + 150 (tutor)
Estimated duration (h)	20 (student) + 5 (tutor)
Deadline	24/04/2024

Table 7.13: "UR5e and Endowrist testing" entry in the WBS Dictionary

WBS code	5.3
Name	Haptic feedback testing
Description	An analysis of the measured torque values, as well as the study of both pulsatile and integrated current measurements. Besides, a revision of the performance of the haptic actuator and the accuracy of the graphical interpreter of the torque.
Estimated cost (€)	300 (student) + 180 (tutor)
Estimated duration (h)	15 (student) + 6 (tutor)
Deadline	08/02/2024

Table 7.14: "Haptic feedback testing" entry in the WBS Dictionary

WBS code	6.1
Name	Results and analysis
Description	Analysis of the results obtained in the different simulations and calculation of different parameters of interest.
Estimated cost (€)	300 (student) + 150 (tutor)
Estimated duration (h)	15 (student) + 5 (tutor)
Deadline	22/05/2024

Table 7.15: "Results and analysis" entry in the WBS Dictionary

WBS code	6.2
Name	Conclusion
Description	From the analysis of the results, a series of conclusions will be drawn to analyze the success or failure of the project.
Estimated cost (€)	240 (student) + 150 (tutor)
Estimated duration (h)	12 (student) + 5 (tutor)
Deadline	29/05/2024

Table 7.16: "Conclusion" entry in the WBS Dictionary

7.3 PERT (Program Evaluation and Review Technique)

With the different activities classified by blocks, it is much easier to relate them to each other in order to carry out the PERT. The PERT diagram is a tool that facilitates the scheduling, organisation and planning of project tasks in detail. It provides a visual representation of the schedule and breaks down the individual tasks. Below are the correlations between the activities entered in the WBS diagram.

Definition	ID	Previous activities	Duration (weeks)	Early start	Early finish
Objectives definition	A	—	1	0	1
Background and market analysis	B	A	3	1	4
Project familiarization	C	A	2	1	3
Master's code development	D	C	4	3	7
RoboDK codes simulation	E	D	4	7	11
Servomotors board design	F	C	2	3	5
Master-slave prototype construction	G	D, F	3	5	10
Master-slave prototype testing	H	G	2	10	12
Sockets implementation	I	E	3	11	14
Servomotors/EndoWrist coupling	J	H	1	12	13
UR5e and EndoWrist testing	K	I, J	1	14	15
Haptic feedback testing	L	K	2	15	17
Results and analysis	M	L	2	17	19
Conclusion	N	M	1	19	20
Project revision and submission	O	N, P	1	20	21
Preparation of the report	P	B	15	4	16

Table 7.17: Project timeline

With this information, we can now make the PERT diagram. For this, we will have to calculate the last time and the early time. These will allow us to know which activities make up the critical path and which ones have slack.

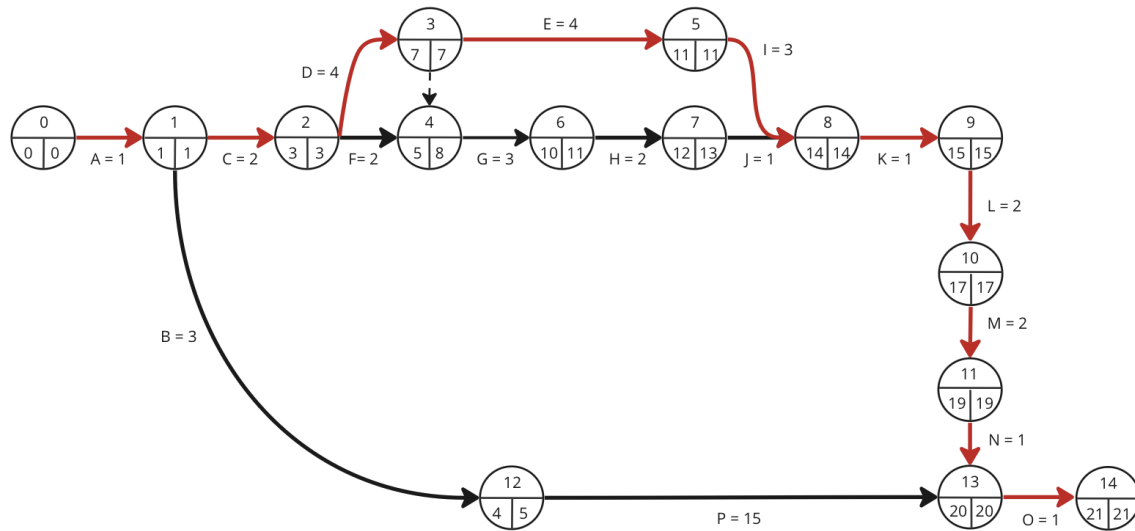


Figure 7.2: PERT diagram

In Figure 7.2 we can identify the critical path, which follows the red arrows. This will allow us to know which activities to pay attention to and prioritise so that delays do not occur. In addition, we have been able to calculate the total time the project would take if everything goes according to plan. This time would be 21 weeks.

7.4 GANTT diagram

Similarly, with the duration of each of the activities and the relationship between them, we can create a GANTT diagram. In this, each bar represents a specific task and its length indicates the expected duration of that task. The GANTT diagram shows critical activities in red, while those with slack are shown in blue, with the slack time represented in grey.

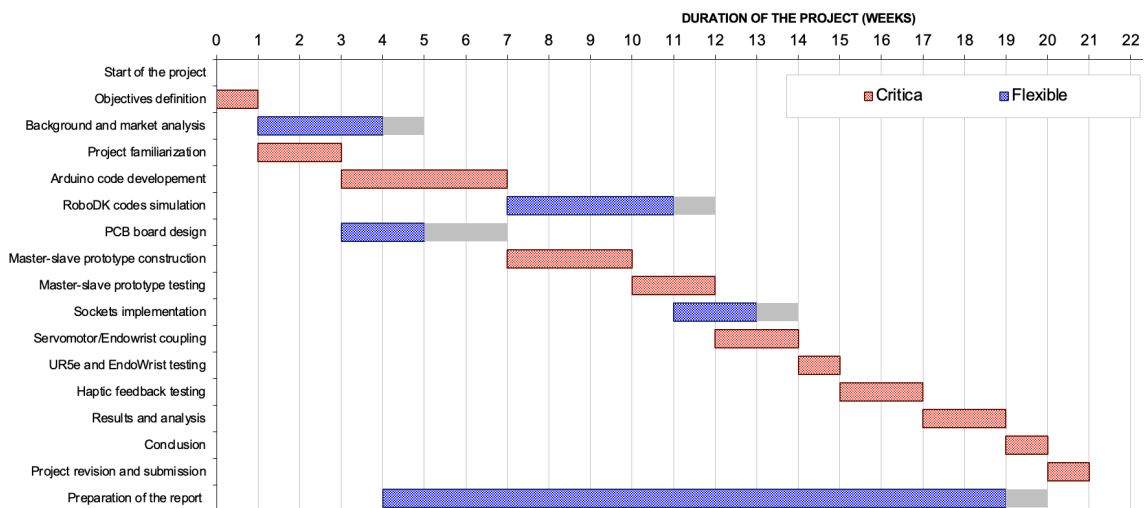


Figure 7.3: GANTT diagram

As we can see in Figure 7.3, the critical activities are the following: Objectives definition, Project familiarization, Master's code development, RoboDK code simulation, Sockets implementation, UR5e and EndoWrist testing, Results and analysis, Conclusions and Project revision and submission.

Thecnical viability

After having discussed the different aspects of our project, in this section we will address the technical feasibility of the project by means of a SWOT analysis. This practice will allow us to evaluate and recognize the different internal and external aspects that may affect the development of the project from a technical perspective. To do so, we will identify the internal strengths and weaknesses, as well as the external opportunities and threats that could determine the final result. In Table 7.1, strengths are represented as positive internal aspects, weaknesses as negative internal aspects, opportunities as positive external aspects and threats as negative external aspects. With this we will be able to acquire a global vision of the context in which it is included and of the aspects that we will have to take into account to achieve the success of the project.

	Positive Aspects	Negative Aspects
Internal Factors	- Compatibility - Low cost - Continuous development	- Hardware limitations - Time and experience - Economic constraints
External Factors	- Expanding market - Academic training - Investment	- Competitive market - Regulations and patents - Acceptance

Table 8.1: SWOT of the project

8.1 Threats

- Competitive market: Over the last few years, many companies and research centres are tackling this field of robotic surgery. This is resulting in a very competitive market where ultimately only those devices that are more precise will be the ones that succeed.
- Regulations and patents: The clinical market is one of those with the most regulations and restrictions, in order to ensure the proper functioning of medical devices and the health of patients. Compliance with all these requirements can mean an increase in both project cost and time.
- Acceptance: The introduction of a new surgical system on the market may lead doctors to opt for an already known and proven mechanism, such as the Da Vinci.

8.2 Weakness

- Hardware limitations: The fact that we have to adapt the hardware we are developing to the preset configuration of the Endowrist tools makes compatibility difficult and may compromise the transmission of motion or communication.
- Time and experience: The time we have to carry out the project and the little experience in programming and robotics of the student limits our possibilities to create a device that meets all the objectives and desirables exposed in the Objectives section.

- Economic constraints: The lack of budget hinders the development of the project and prevents us from acquiring sensors with the best sensitivity on the market, which would result in a much more precise arm movement.

8.3 Opportunities

- Expanding market: The market growth that has been taking place in recent years makes it easier to obtain information on the subject and to acquire electronic material and components.
- Academic training: The fact that it is a low-cost system and that robotic surgery techniques are currently being taught among medical students, makes it easier for some institutions to become interested in our device.
- Investment: The growing demand for surgical technologies could drive investment in research and development in this field, which could lead to significant advances and new applications.

8.4 Strengths

- Compatibility: The usage of the original Da Vinci EndoWrist tool allows our surgical robotic system to be used with a large number of tools that the company has developed. This gives us great versatility and flexibility when performing procedures.
- Low cost: The system is designed to have a much lower production cost than current surgical systems on the market. With this, we could bring robotic surgery to hospitals with fewer resources.
- Continuous development: Possibility to further enhance the system to upgrade the performance and introduce improvements as required.

Economical viability

The economic aspect is fundamental to evaluate the financial and economic viability of the project, as well as to know where the product can be positioned in the market. In this section an analysis of the costs will be made, including the hardware and software used, as well as the student's and director's salary. In accordance with the teaching plan for the Final Degree Project of the Biomedical Engineering degree, a minimum of 300h is required from the student. In addition, it will be assumed that the director will end up investing a total of 100h in the project.

Below is a table with the total cost of the project broken down according to the different fields of the project.

Concept	Unit Price (€)	Quantity	Total Cost (€)
Software			
Python	0	1	0
Arduino	0	1	0
KiKad	0	1	0
RoboDK	0	1	0
Hardware			
UR5e	30.851	1	30.851
EndoWrist	4500	1	4500
IMU MPU-9250	12.22	2	24.44
ESP32 board	11	4	44
Servomotors Sunfounder SF 180M	6	4	24
Resistors	0.09	6	0.54
Switch	0.04	1	0.04
C-type port	6	1	6
Vibration Motor	1.2	1	1.2
Push Button	0.32	4	1.28
Power Supply	20	2	40
Suture practice pad	15	1	15
Elastic bracelet	6	1	6
PCB fabrication	20	2	40
Computer	1000	1	1000
Salary			
Student	20€/h	300h	6000
Director	30€/h	100h	3000
Estimated total cost of the TFG project			45553.5 €

Table 9.1: Cost breakdown of the TFG project

9.1 Hardware and Software Analysis

As far as the software is concerned, the fact that we have the licences provided by the university has allowed us to significantly reduce the final cost of the project, as we can use the different programmes free of charge. On the other hand, 77.6% of the total cost of the project corresponds to the price of the UR5e robotic arm and the EndoWrist tool. As

the robot was available at the robotics laboratory of the Universitat de Barcelona and the EndoWrist was loaned by the Hospital Cl nic, a total of 35,351 euros were saved.

In addition, a significant part of the project cost, amounting to 9,000 euros, goes to the salaries of both the student and the tutor.

Therefore, if we focus exclusively on the cost of the development and manufacture of the hardware, we obtain a value of 202.5 euros, considering that a computer was already available.

9.2 Conclusions on System Viability

As for the complete robotic system, it has been manufactured for a total of 36553.5 . However, although the proposed solution is technically feasible and the price of manufacturing and development is affordable, it is crucial to analyse its economic viability in comparison with established robotic surgical systems such as the da Vinci. For its part, the da Vinci system, despite its high acquisition and maintenance cost, has proven to be an efficient and reliable tool in a wide variety of surgical procedures. Moreover, its dominant position in the market and the support of multiple clinical studies and certifications make it a reference system in robotic surgery.

After analysing the performance results of the developed system, we can conclude that it works remarkably well. Nevertheless, in order to be competitive, it needs to demonstrate robustness and superior performance in real clinical conditions. This would mean focusing not only on manoeuvrability and haptic technology, but also on other aspects such as increased precision in tasks like suturing or the ability to withstand intensive and continuous use in diverse surgical environments. System durability and reliability are critical aspects that must be rigorously validated through extensive clinical studies and certification processes.

Hence, for the system to be considered viable, the solution must be integrated into a complete, multifunctional system that can perform a wide range of surgical procedures with efficiency similar to that of existing systems. Nonetheless, the presented prototype is a good start towards such a final device.

Legal aspects

10.1 Legal protection of robotic surgery

Surgical robotics has undergone a great evolution over the last few years, establishing new treatment models and new ways of practising and performing surgery. This change has caused the capacity of adaptation of the current legal institutions to overflow, which are being modernised to avoid being an obstacle to the development of this new technology. The current regulation of intellectual property that has shielded general robotics is not managing to fill the gaps that robotic surgery is generating, which creates certain issues of discussion that had already been raised in general robotics. From questioning the patentability of the robot or the procedure it performs, to considering whether a protection mechanism can stifle advances in this field.

Intellectual property offers multiple protections for surgical robotics, due to the fact that robots are composed of hardware and software and each of these elements can be protected by different mechanisms. This plurality of protections has given rise to multiple discussions, due to clashes between the legislations present. An example would be the dilemma of the possibility of patenting or not the procedure for which the robot is programmed when it is in an exception of patentability.

The first protection mechanism for the surgical robot is the protection of the software as a literary work by means of copyright. The TRIPS Agreement establishes in its Article 10 that the legal protection of software shall be made by this means, expressing that "computer programs, whether source programs or object programs, shall be protected as literary works". However, copyright protects only the source code and object code of a certain device, so that any idea or development resulting from the application or modification thereof is not protected. Indeed, Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs, in point 11 states the following:

"For the avoidance of doubt, it should be clearly stated that only the expression of the computer program is protected and that the ideas and principles implicit in the elements of the program, including those of its interfaces, are not eligible for copyright protection under this Directive. In accordance with this copyright principle, to the extent that logic, algorithms and programming languages embrace ideas and principles, the latter are not protected under this Directive."

Another important protection mechanism is the establishment of patents. When talking about patents for surgical robotics, there are several protection scenarios, such as software patents, patents for the robot as a whole or procedure patents.

Regarding the software patent, article 15 of the Andean Decision 486 of 2000 states that computer programs (software) are not patentable, since it affirms that computer programs or software, as such, shall not be considered inventions. In support of this statement, it has been pointed out that software is reduced to a mathematical expression. However, the jurisprudence is trying to make interpretations so that there is the possibility of patenting certain inventions developed by software.

Regarding the patenting of the robot as a whole, by conceiving the robot as the invention itself without abstracting from its parts, it is possible to grant a patent. This means

changing the method of protection for software, since it is not the literal expression that is to be patented, but a complete machine.

When it is a matter of patenting the procedure performed by the robot surgeon, although there is no pronouncement on the matter, it does not seem possible to apply for a patent for the procedure, precisely because it is a therapeutic or surgical method [86].

10.2 Regulations and institutions

As discussed along the project, robotic-assisted surgery has a number of advantages over traditional laparoscopic surgery, however, its safety and efficacy can only be guaranteed when used properly and with appropriate training. To this end, the FDA regulates devices to provide reasonable assurance of their safety and efficacy for their intended uses. For EAR devices, this includes ensuring that manufacturers implement appropriate training programs for both new and experienced users [87].

On the other hand, if the device is to be marketed in Europe, it is essential that it obtains Conformité Européenne (CE) approval. Once the CE marking has been obtained, the company guarantees that the medical device complies with the General Safety and Performance Requirements (GSPR) of all relevant European Medical Device Regulations, which is a legal requirement for marketing a product in the European Union [88].

A medical device has been defined by the Royal Decree 1591/2009 as any instrument, device, equipment, computer program, material or other article intended by the manufacturer to be used on human beings for the diagnosis, prevention, control, treatment or alleviation of a disease. By this definition, surgical robots are considered medical devices and are therefore regulated by the same principles, standards and laws. One of the most recent regulations, the Medical Devices Regulation (EU) 2017/745, was approved with the aim of safeguarding the safety and health of patients and medical staff alike, in the face of the launch of new medical devices.

Another fundamental process in the implementation and launch of new devices is standardisation. In this, a series of global standards are established to improve quality, safety and efficiency in the production of products and services. These standards are ISO and IEC, which are developed by experts from around the world to contribute to the global commitment to sustainability. These regulations provide a homogeneous language for communication between businesses and even nations, as well as helping to promote the free and fair exchange of goods and services between countries. This allows consumers to receive safe, high-quality products, which reduces the cost associated with defective or low-quality products [89].

Conclusions

Once the prototype has been assembled and tested, and the economic and technical feasibility study has been carried out, it is possible to analyse whether the objectives set at the start of the project have been satisfactorily achieved. The two main objectives that had the most importance in the project were, on the one hand, the introduction of a haptic sensitivity system that would allow the surgeon to feel the force that is being applied and, on the other hand, to improve the robustness of the transmission system of the surgeon's arm orientation to the unit formed by the UR5e and the EndoWrist.

Consequently, an analysis of the outcomes achieved at the conclusion of the project and the inferences that can be drawn from them within the context of the current technological landscape and market conditions will be presented.

11.1 Objectives and results

To begin with, fluid communication has been established between all the modules that make up the prototype, thanks to the use of ESP32 microprocessors. In addition, the data flow developed has allowed the exchange of information between modules to be reduced to a minimum, which has helped to optimise the programme.

On the other hand, as can be seen in the video attached in the section 6.2 Robot maneuverability, a good translation of the orientations of both the Pen User Interface and the Forearm Master has been achieved. After applying the necessary transformations and rotations, explained in the section 5.3.2.2 Suture process, the movement carried out by the surgeon is accurately translated into the surgical system, which provides comfort and is intuitive for the user.

However, there are some limitations regarding the transmission of information from the computer to the UR5e robotic arm. Although the socket system allows rapid, real-time communication between these two elements, it also introduces a delay so that the command can be sent correctly and the robot can reach the established position. When this delay is less than the time given to the robot to carry out the movement, the arm stops and resumes the movement every time the delay time is reached, which causes a vibration in the system that, firstly, can end up damaging it and, secondly, reduces the precision and stability of the suturing process. In the end, a time of 0.4 seconds was chosen in an attempt to balance a fast system response on the one hand, and an adequate speed of movement on the other hand, to allow a smooth and safe procedure to be carried out.

With regard to the transmission of motion between the servomotors and the gears of the EndoWrist, the results achieved can be considered satisfactory. Following the final assembly, a good fit of the designed parts could be observed. However, on some occasions, there is a slight displacement between the parts, which causes a small difference in the degrees of rotation between the servomotor and the EndoWrist. The issue is currently being addressed manually, with the user required to make a slight movement to compensate for the discrepancy.

This leads us to analyse the comfort and performance of the suturing process, both in terms of gripper movement and needle clamping. On the one hand, as the rotation is correctly transmitted to the EndoWrist, the gripper movement is correct, without delays

and faithful to the movement carried out by the Pen User Interface. On the other hand, the needle is well gripped and can be inserted into the tissue without any problems and without slipping out of the gripper. For this, it has been very important to modify the code of the EndoWrist Motion Panel, since the addition of 30 degrees to the Roll1 allows us to hold the needle much more firmly, preventing it from moving or dislodging when it comes into contact with the tissue.

As far as haptic technology is concerned, after analysing the results pertaining to the section 6.1 Haptic feedback system performance, it can be affirmed that it is capable of transmitting the force exerted on the tissue by the gripper. Furthermore, both the actuator introduced in the Pen User Interface and the graphic interface provide very useful and complementary information to the surgeon. While the vibration of the control and the display of the bars on the interface provide an idea of the amount of force being applied, the audible alarm after passing the threshold provides a safety measure to avoid excessive forces.

Therefore, if we add to all these conclusions that the system has a total cost of 36555.5 euros, we can consider that the main objective of the project has been fulfilled, by developing a low-cost system with a certain robustness. Although it is true that certain updates are still pending to facilitate the surgeon's task and to provide extra safety, avoiding the most abrupt and unexpected movements of the robotic arm, the system's performance and manoeuvrability have been notably improved.

11.2 Future work

Although the objectives have been met, the technology developed has a great deal of room for improvement, which could be addressed in future work.

As for the haptic technology, it currently provides an estimation of the applied force. However, the actuator does not provide information on which axis the force is being exerted, requiring the surgeon to look away from the surgical field to know exactly what force is being exerted on each axis. To improve this, a system could be developed to reflect the force on all three axes so that the user can more easily control the force being exerted.

On the other hand, it would be necessary to carry out a study to calculate the force thresholds allowed by each of the tissues of the human body, as well as that of the suture thread. In this way, the safety of the system would increase notably, since it could be programmed to reduce the force exerted if it is exceeded at any time during the operation.

As for the anchoring system between the servomotors and the EndoWrist, although it works quite well, it should be improved to avoid the small displacement that sometimes occurs. A possible solution could be the introduction of small magnets in the vertical section of the parts, thus preventing them from moving apart.

To conclude, the manoeuvrability of the final system should be improved by reducing the delay produced by communication via sockets, by changing parameters such as speed and movement time. In this way, the delay could be reduced to 0.1 seconds, offering near real-time communication.

Bibliography

- [1] T. Leal Ghezzi and O. Campos Corleta, “30 years of robotic surgery,” *World J Surg*, vol. 40, no. 10, pp. 2550–2557, 2016. Available from: <http://dx.doi.org/10.1007/s00268-016-3543-9>.
- [2] A. C. R. A. Kommu SS, Rimington P, “Initial experience with the endoassist camera-holding robot in laparoscopic urological surgery.” Available from: https://www.researchgate.net/figure/The-arrow-shows-the-camera-driver-of-the-EndoAssist_fig1_227007521 [accessed 31 May, 2024], 2024. Scientific Figure on ResearchGate.
- [3] J. Romero Otero, P. Paparel, D. Atreya, K. Touijer, and B. Guillonneau, “Antecedentes, desarrollo y situación actual de la robótica en cirugía,” *Archivos Espanoles de Urologia*, vol. 60, no. 4, pp. 335–341, 2007.
- [4] E. Oliver, “Los millonarios robots para tratar el cancer se expanden en españa,” *Crónica Global*, enero 11 2020. Available from: https://www.google.com/amp/s/cronicaglobal.elespanol.com/creacion/vida-tecky/20200111/los-millonarios-robots-tratar-cancer-expanden-espana/458954148_0.amp.html.
- [5] C. Technology, “Minogue medical inc. endowrist instruments.” <http://www.minogue-med.com/core-technology/>.
- [6] Cobotica, “UR5e - Cobotica - Robots colaborativos,” may 12 2022. Available from: <https://cobotica.es/producto/robots/universal-robots/ur5e/>.
- [7] Riders, “What is kuka arm?.” Available from: <https://riders.ai/en-blog/what-is-kuka-arm>.
- [8] “SWIFTITM CRB 1100.” Available from: <https://webshop.robotics.abb.com/es/catalog/product/view/id/234/s/swifti-crb-1100-assembly/category/3/>.
- [9] E. Demaitre, “Vicarious surgical ceo discusses novel approach to surgical robotics, spac and nyse validation,” *Robotics 24/7*. Available from: https://www.robotics247.com/article/vicarious_surgical_ceo_discusses_novel_surgical_robotics_approach_spac_nyse_validation.
- [10] H. e. Espanyol, “Robot quirurgico con brazos en forma de humanoide replica los movimientos del cirujano,” *HospitMedica En Español*, junio 29 2022. Available from: https://mobile.hospimedica.es/?option=com_mobile_article&Itemid=294793675.
- [11] P. Kumar and B. Ravi, “A comparative study of robots in laparoscopic surgeries,” in *Proceedings of the Advances in Robotics 2019*, 2019. Available from: <https://dl.acm.org/doi/10.1145/3352593.3352608>.
- [12] D. Alcaraz, “Optimized endowrist tool for ur5e robot arm in surgical robotics applications,” 2023.
- [13] R. Bravo and A. Lacy, “Medicina y robótica,” *Med Clin (Barc)*, vol. 145, no. 11, pp. 493–495, 2015. Available from: <https://pubmed.ncbi.nlm.nih.gov/26004280/>.

- [14] G. Antoniou, C. Riga, E. Mayer, N. Cheshire, and C. Bicknell, "Clinical applications of robotic technology in vascular and endovascular surgery," *J Vasc Surg*, vol. 53, no. 2, pp. 493–499, 2011. Available from: <https://pubmed.ncbi.nlm.nih.gov/20801611/>.
- [15] S. Kulkarni, N. Seneviratne, M. Baig, and A. Khan, "Artificial intelligence in medicine: Where are we now?," *Acad Radiol*, vol. 27, no. 1, pp. 62–70, 2020. Available from: <https://pubmed.ncbi.nlm.nih.gov/31636002/>.
- [16] D. Pavard, A. Decock, and F. Bruyère, "La néphrectomie pour don vivant de rein par laparoscopie robot assistée," *Prog Urol*, vol. 25, no. 14, pp. 892–899, 2015. Available from: <https://www.sciencedirect.com/science/article/pii/S1166708715006314>.
- [17] R. Médica, "100th da vinci robotic system installed on the iberian peninsula," 2023.
- [18] Barnaclinic+, "The hospital clinic group, pioneer and leader in robotic surgery," 2021. Available from: <https://www.barnaclinic.com/blog/en/blog/robotic-surgery/>.
- [19] SAGES, "Sages/mira consensus document on robotic surgery," *SAGES*, 2007. Available from: <https://www.sages.org/publications/guidelines/consensus-document-robotic-surgery/>.
- [20] W. Zhang, H. Li, L. Cui, H. Li, X. Zhang, S. Fang, *et al.*, "Research progress and development trend of surgical robot and surgical instrument arm," *Int J Med Robot*, vol. 17, no. 5, p. e2309, 2021. Available from: <http://dx.doi.org/10.1002/rcs.2309>.
- [21] R. Jara, A. Guerrón, and D. Portenier, "Complications of robotic surgery," *Surg Clin North Am*, vol. 100, no. 2, pp. 461–468, 2020. Available from: <https://www.sciencedirect.com/science/article/pii/S0039610919301665>.
- [22] T. Williamson and S.-E. Song, "Robotic surgery techniques to improve traditional laparoscopy," *JSLs*, vol. 26, no. 2, p. e2022.00002, 2022. Available from: <http://dx.doi.org/10.4293/JSLs.2022.00002>.
- [23] C. Martínez Ramos, "Robótica y cirugía laparoscópica," *Cir Esp*, vol. 80, no. 4, pp. 189–194, 2006.
- [24] L. J. Borden, P. Kozlowski, C. Porter, and J. Corman, "Mechanical failure rate of da vinci robotic system," *Can J Urol*, vol. 14, no. 2, pp. 3499–3501, 2007.
- [25] S. Maeso Martínez, D. Callejo Velasco, M. Guerra Rodríguez, and J. A. Blasco Amaro, *Revisión sistemática de las evaluaciones económicas de la cirugía mediante equipo quirúrgico da Vinci*. Informes, estudios e investigación. Serie: Informes de Evaluación de Tecnologías Sanitarias. UETS 2010/03, Madrid: Plan de Calidad para el SNS del MSSSI. Unidad de Evaluación de Tecnologías Sanitarias, Agencia Laín Entralgo, 2011. ISBN: 978-84-451-3423-8, NIPOS: 725-12-023-4; 680-12-019-0, Depósito legal: BI. 1.070-2012.
- [26] R. Cirocchi, F. Cesare Campanile, S. Di Saverio, G. Popivanov, L. Carlini, D. Pironi, *et al.*, "Laparoscopic versus open colectomy for obstructing right colon cancer: A systematic review and meta-analysis," *J Visc Surg*, vol. 154, no. 6, pp. 387–399, 2017. Available from: <https://www.sciencedirect.com/science/article/pii/S1878788617301145>.

- [27] “Nci cancer dictionary,” 2011. Available from: <https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/cirugia-minimamente-invasiva>.
- [28] D. Ranev and J. Teixeira, “History of computer-assisted surgery,” *Surg Clin North Am*, vol. 100, no. 2, pp. 209–218, 2020. Available from: <https://pubmed.ncbi.nlm.nih.gov/32169176/>.
- [29] H.-X. Zhou, Y.-H. Guo, X.-F. Yu, S.-Y. Bao, J.-L. Liu, Y. Zhang, *et al.*, “Clinical characteristics of remote zeus robot-assisted laparoscopic cholecystectomy: a report of 40 cases,” *World J Gastroenterol*, vol. 12, no. 16, pp. 2606–2609, 2006. Available from: <http://dx.doi.org/10.3748/wjg.v12.i16.2606>.
- [30] H. Ashrafian, O. Clancy, V. Grover, and A. Darzi, “The evolution of robotic surgery: surgical and anaesthetic aspects,” *Br J Anaesth*, vol. 119, no. suppl_1, pp. i72–i84, 2017. Available from: <https://pubmed.ncbi.nlm.nih.gov/29161400/>.
- [31] S. Aiono, J. M. Gilbert, B. Soin, *et al.*, “Controlled trial of the introduction of a robotic camera assistant (endoassist) for laparoscopic cholecystectomy,” *Surgical Endoscopy*, vol. 16, p. 1267, 2002.
- [32] M. Azizian, M. Liu, I. Khalaji, and DiMaio’S, “The da vinci surgical system,” *The Encyclopedia of Medical Robotics*, pp. 3–28, 2018.
- [33] Intuitive Surgical Inc., “Da Vinci Surgical System. Intuitive Surgical main page.” Available from: <https://www.intuitive.com/en-us/products-and-services/da-vinci/systems>.
- [34] M. Ramírez, “El sistema robótico da vinci: la última evolución de la cirugía mínimamente invasiva,” 2022.
- [35] D. R. Mishra, “System components minimal access surgery, robotic surgery.” Available from: <https://www.laparoscopyhospital.com/Book/Ch-03.pdf>.
- [36] R. studio, “Davinci surgeon console.” Available from: <http://www.rsalinas.com/davinci-si-1>.
- [37] Abex Excelencia Robórica, “Sistema robotico da vinci.” Available from: <https://www.abexsl.es/es/sistema-robotico-da-vinci/nuevas-tecnologias>.
- [38] R. K. Mishra, “Instruments used in robotic surgery.” <https://www.laparoscopyhospital.com/instrument-used-in-robotic-surgery.php>.
- [39] I. Surgical, “Endowrist instrument and accessory catalog.” https://www.unthsc.edu/patient-care/wp-content/uploads/sites/27/Intuitive_Surgical_ins_cat_1_.pdf.
- [40] A. Okamura, “Haptic feedback in robot-assisted minimally invasive surgery,” *Curr Opin Urol*, vol. 19, no. 1, pp. 102–107, 2009. Available from: <http://dx.doi.org/10.1097/MOU.0b013e32831a478c>.
- [41] Q. Ouyang, J. Wu, S. Sun, J. Pensa, A. Abiri, E. Dutson, *et al.*, “Bio-inspired haptic feedback for artificial palpation in robotic surgery,” *IEEE Trans Biomed Eng*, vol. 68, no. 10, pp. 3184–3193, 2021. Available from: <https://pubmed.ncbi.nlm.nih.gov/33905321/>.

- [42] R. Ciranov, J. Clobet, V. Grover, and A. Darzi, "The evolution of robotic surgery: surgical and anaesthetic aspects," *Br J Anaesth*, vol. 119, no. suppl_1, pp. i72–i84, 2017. Available from: <https://pubmed.ncbi.nlm.nih.gov/29161400/>.
- [43] A. Cadanet, "Objetivo, 63 millones: el gigante español de robots da vinci preve crecer un 40 en 2019," 2019.
- [44] K. B. Azizi, E. Wilson, R. Monfaredi, C. Peters, G. Kronreif, and K. Cleary, "Robotic natural orifice transluminal endoscopic surgery (r-notes): literature review and prototype system," *Minim Invasive Ther Allied Technol*, vol. 24, no. 1, pp. 18–23, 2015. Available from: <http://dx.doi.org/10.3109/13645706.2014.992907>.
- [45] T. G. Leong, D. H. Gracias, P. A. Monge, P. G. Flexman, J. T. Kavalier, C. J. Randall, and D. H. Gracias, "Microgrippers for minimally invasive tissue sampling and analysis," *Nature Biomedical Engineering*, vol. 3, pp. 346–351, 2019.
- [46] "Robotic surgical procedures market size, share and growth." Available from: <https://www.fortunebusinessinsights.com/industry-reports/robotic-surgical-procedures-market-100124>.
- [47] "Informe del mercado de procedimientos quirurgicos robotica pronostico 2028." Available from: <https://www.mordorintelligence.com/industry-reports/robotic-surgical-procedures-market>.
- [48] Uromadrid.es, "Oferta de equipos roboticos disponibles en el mercado. presente y futuro, ventajas e inconvenientes de cada uno."
- [49] ROC Clinic, "Robot hugo," 2022. Available from: <https://rocclinic.com/robot-hugo/>.
- [50] CMR Surgical, "Meet versius." Available from: <https://cmrsurgical.com/>.
- [51] Rob Surgical, "Bitrack system," 2022. Available from: <https://www.robsurgical.com/es/bitrack-system/>.
- [52] G. Riva, E. Cravero, M. Briguglio, P. Capaccio, and G. Pecorari, "The flex robotic system in head and neck surgery: A review," *Cancers (Basel)*, vol. 14, no. 22, p. 5541, 2022. Available from: <https://www.mdpi.com/2072-6694/14/22/5541>.
- [53] U. Hagn, M. Nickl, S. Jörg, G. Passig, T. Bahls, A. Nothhelfer, F. Hacker, L. Le-Tien, A. Albu-Schäffer, R. Konietschke, M. Grebenstein, R. Warpup, R. Haslinger, M. Frommberger, and G. Hirzinger, "The dlr miro - a versatile lightweight robot for surgical applications." https://elib.dlr.de/76858/1/J_Industrial_Robots_MIR02008_Hagn.pdf, n.d. Retrieved May 30, 2024.
- [54] M. Roche, "The mako robotic-arm knee arthroplasty system," *Arch Orthop Trauma Surg*, vol. 141, no. 12, p. 2043,2047, 2021. Available from: <http://dx.doi.org/10.1007/s00402-021-04208-0>.
- [55] D. J. Jacofsky and M. Allen, "Robotics in arthroplasty: A comprehensive review," *J Arthroplasty*, vol. 31, no. 10, p. 2353,2363, 2016. Available from: <https://pubmed.ncbi.nlm.nih.gov/27325369/>.
- [56] German Aerospace Center, "Hugotm ras-system." <https://www.dlr.de/en/images/2021/4/hugo-ras-system-en>.

- [57] S. Ross, "Flex robotic system and flex colorectal drive." <https://www.sages.org/publications/tavac/flex-robotic-system-and-flex-colorectal-drive/>, April 2 2018. Accessed: 2024-05-31.
- [58] Institute of Robotics and Mechatronics, "Miro." <https://www.dlr.de/en/rm/research/robotic-systems/arms/miro>.
- [59] Uintah Basin Healthcare, "Surgery: Makotm robotic – arm assisted surgery." <https://ubh.org/surgery-mako-robotic-arm-assisted-surgery/>. Accessed: 2024-05-31.
- [60] G. G. S. Richmond, Gregory F., "Grip force control for robotic surgical instrument end effector," 2020.
- [61] A. MacDougall, "Application of force feedback on an input device to urge its operator to command an articulated instrument to a preferred pose," 2012.
- [62] A. C. THOMPSON, "Master control device with finger grip sensing and methods therefor," 2018.
- [63] D. Q. Larkin, "Preventing instrument/tissue collisions," 2018.
- [64] Y. Chen, S. Zhang, Z. Wu, B. Yang, Q. Luo, and K. Xu, "Review of surgical robotic systems for keyhole and endoscopic procedures: state of the art and perspectives," *Front Med*, vol. 14, p. 382,403, 2020.
- [65] J. Troccaz, G. Dagnino, and G.-Z. Yang, "Frontiers of medical robotics: From concept to systems to clinical translation," *Annu Rev Biomed Eng*, vol. 21, no. 1, p. 193,218, 2019. Available from: <https://pubmed.ncbi.nlm.nih.gov/30822100/>.
- [66] P. Schuler, "Robotic surgery - who is the boss?," *Laryngorhinootologie*, vol. 97, no. S 01, p. S231,S278, 2018. Available from: <http://dx.doi.org/10.1055/s-0043-121791>.
- [67] Pensamiento Computacional, "Acamica: Ventajas y desventajas de arduino," 2016. Available from: <https://edgardosilvi.wordpress.com/2016/02/29/acamica-ventajas-y-desventajas-de-arduino/>.
- [68] KeepCoding Bootcamps, "Ventajas y desventajas de python," 2021. Available from: <https://keepcoding.io/blog/ventajas-y-desventajas-de-python/>.
- [69] Jorge Marez, "J370: Ventajas y desventajas de la simulacion," 2011. Available from: <https://jmirez.wordpress.com/2011/11/26/j370-ventajas-y-desventajas-de-la-simulacion/>.
- [70] "Los beneficios de la programacion grafica en ni labview," June 14 2013. Available from: <https://www.ni.com/es-es/shop/labview/benefits-of-programming-graphically-in-ni-labview.html>.
- [71] B. McNeel, "Noticias de rhino, etc.," n.d. Available from: <https://blog.es.rhino3d.com/2018/07/robodk-programacion-de-robots-en-rhino.html?m=1>.
- [72] "Kikad reference manual." <https://docs.kicad.org/4.0/es/kicad/kicad.html>.
- [73] "¿que es proteus?." <https://www.hubor-proteus.com/proteus-pcb/proteus-pcb/2-proteus.html>.
- [74] Comparativa de Software para diseño de PCB - Tutoriales HeTPro, "Hetpro tutoriales," April 15 2014. <https://hetpro-store.com/TUTORIALES/disenio-de-pcb/>.

- [75] C. Esneca, "What is a robotic arm and what is it for?," November 7 2022. Available from: <https://www.esneca.lat/blog/brazo-robotico-aplicaciones-funciones/>.
- [76] "UR5e - Cobotica - Collaborative Robots," May 12 2022. Available from: <https://cobotica.es/producto/robots/universal-robots/ur5e/>.
- [77] "KUKA Robots For Sale," July 1 2021. Available from: <https://www.robot-store.co.uk/kuka-robots-for-sale>.
- [78] "El nuevo cobot industrial swiftitm de abb ofrece velocidad, precision y seguridad lideres en su clase."
- [79] A. S. S. Ex, "Swiftitm crb 1300." Available from: <https://webshop.robotics.abb.com/es/swiftitm-crb-1300-part-handling-set.html>.
- [80] "Vicarious surgical," September 24 2022. Available from: <https://www.vicarioussurgical.com>.
- [81] H. e. Espanyol, "Robot quirurgico con brazos en forma de humanoide replica los movimientos del cirujano," June 29 2022.
- [82] C. A. Pomalia Neyra, I. Rivas Blanco, and M. B. Estabanez Campos, "Design and control of a coupling system of an endowrist instrument for an UR3 robotic arm," 2020.
- [83] U. F. O. Themes, "The da vinci surgical system and endowrist instruments," March 16 2017.
- [84] D. Hazry, M. Sofian, and A. Zul Azfar, "Study of inertial measurement unit sensor."
- [85] S. Systems, "IMU - Unidad de medición inercial," 2018. <https://www.sbg-systems.com/es/unidad-de-medicion-inercial-sensor-imu/>.
- [86] L. Paola, "Robtica quirurgica y su proteccion juridica desde la propiedad intelectual," Available from: <https://revistas.uexternado.edu.co/index.php/propin/article/view/7023/9740>.
- [87] "Computer-assisted surgical systems," n.d. Available from: <https://www.fda.gov/medical-devices/surgery-devices/computer-assisted-surgical-systems>.
- [88] "Mercado ce para productos sanitarios en europa." Available from: <https://www.bsigroup.com/es-ES/Productos-Sanitarios/Servicios-para-Productos-Sanitarios/Mercado-CE/>.
- [89] "Iso: ¿que es y para que sirven estas normas de estandarizacion?," 2023. Available from: <https://www.gotelgest.net/iso-que-es-y-para-que-sirven-estas-normas-de-estandarizacion/>.

Appendices

Appendices

.1 Main Python code

```
1 # -----
2 # Import of libraries
3 # -----
4
5 import socket
6 import time
7 from math import radians
8 import serial
9 from enum import Enum
10 from spatialmath.base import *
11 from numpy import array
12 from robodk.robomath import *
13 import threading
14 from pynput import keyboard as kb
15 from robolink import *
16 from robodk import *
17 import subprocess
18
19 # -----
20 # UR5e connection checking
21 # -----
22
23 # Define the robot's IP address and port
24 robot_ip = "192.168.1.5"
25 robot_port = 30002 # Default port for UR robots
26
27 def check_robot_port(robot_ip, robot_port):
28
29     global robot_socket
30
31     try:
32         robot_socket = socket.socket(socket.AF_INET,
33 socket.SOCK_STREAM)
34         robot_socket.settimeout(1) # Timeout of 1 second
35         robot_socket.connect((robot_ip, robot_port))
36         return True
37     except (socket.timeout, ConnectionRefusedError):
38         return False
39
40 robot_is_connected = check_robot_port(robot_ip, robot_port)
41
42 # -----
43 # Connection with RoboDK
44 # -----
45
46 # Establish a link with the simulator RoboDK
47 RDK = robolink.Robolink()
48
49 robot = RDK.Item('UR5e')
50 base = RDK.Item('UR5e Base')
```

```

51 # Define the "endowrist" variable with the TCP of Endowrist needle
52 endowrist = RDK.Item('Endowrist')
53 set_TCP = "set_tcp(p[0.08,0.025,.6,0,0,0])" # Set hardware TCP tool
54 gripper = RDK.Item('Gripper')
55 needle = RDK.Item('Needle')
56 Init_target = RDK.Item('Init') # Defined in the RBK programme
57
58 # Move robot to Init POSE
59 endowrist_init= TxyzRxyz_2_Pose([0, -10, 500, pi/2, 0, 0])
60 endowrist.setParent(robot)
61 endowrist.setPose(endowrist_init)
62 gripper_init = TxyzRxyz_2_Pose([0, 109, 0, pi/2, 0, 0])
63 gripper.setParent(endowrist)
64 gripper.setPose(gripper_init)
65 needle_init = TxyzRxyz_2_Pose([0, 0, 0, 0, -pi/2, 0])
66 needle.setParent(gripper) # First time important setParent for
67                             # absolute POSE
68 needle.setPose(needle_init)
69
70 # Set the robot position
71 print("POSE Init: ")
72 robot.MoveL(Init_target)
73 print("Pose obtained")
74 time.sleep(1)
75 robot.setSpeed(5)
76
77 # -----
78 # Serial connection
79 # -----
80
81 serial_port = '/dev/cu.usbserial-0001'
82 baud_rate = 115200
83
84 # Socket configuration
85 host = 'localhost'
86 port = 9999
87
88 # Start the serial port
89 try:
90     arduino = serial.Serial(serial_port, baud_rate, timeout=1)
91     print(f"Connected to serial port: {serial_port} at {baud_rate}
92         baud.")
93 except serial.SerialException as e:
94     print(f"Error opening the serial port: {e}")
95     exit(1)
96
97 # Start the server socket
98 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
99 server_socket.bind((host, port))
100 server_socket.listen(5)
101 print(f"Server listening on {host}: {port}")
102
103 # List of connected clients
104 clients = []
105
106 time.sleep(2)
107 # -----

```

```

108 # Client  initialisation
109 # -----
110
111 # Client initialization
112 def run_script(script):
113     subprocess.Popen(["/usr/local/bin/python3", script])
114
115 if __name__ == "__main__":
116     scripts = ["TKinter_root_final.py"]
117     for script in scripts:
118         run_script(script)
119
120 # -----
121 # RoboDK reading
122 # -----
123
124 def read_RDK():
125
126     #Read the position of the robot
127     robot_pose=robot.Pose()
128     Xr, Yr, Zr,_,_,_ = Pose_2_TxyzRxyz(robot_pose)
129
130     #Read the position of the gripper
131     gripper_pose=grripper.Pose()
132     Xg, Yg, Zg,_,_,_ = Pose_2_TxyzRxyz(gripper_pose)
133
134     return Xg, Yg, Zg, Xr, Yr, Zr
135
136
137 # -----
138 # Data reading
139 # -----
140
141 class Command(Enum):
142     GET_RPW = b'\x01'
143
144 # Discard initial ESP32 message
145 arduino.reset_input_buffer()
146
147 # Initialize variables to be called later
148 s1_1, s2_1, roll1, pitch1, yaw1, s1_2, s2_2, roll2, pitch2, yaw2,
149 torque_roll1, torque_roll2, torque_pitch, torque_yaw, R1, R2,
150 P1, P2, W1, W2= [0]*20
151
152 def data_reading_sending(client_socket):
153
154     global s1_1, s2_1, roll1, pitch1, yaw1, s1_2, s2_2, roll2,
155     pitch2, yaw2, torque_roll1, torque_roll2, torque_pitch,
156     torque_yaw, R1, R2, P1, P2, W1, W2
157
158     while True:
159         try:
160             arduino.write(Command.GET_RPW.value)
161             # Read all data lines from Arduino
162             data = b""
163             for _ in range(14):
164                 data += arduino.readline()
165             data = data.strip()

```

```

162
163         if data:
164             # Save received values
165             float_values = [float(val) for val in data.split()]
166             s1_1, s2_1, roll1, pitch1, yaw1, s1_2, s2_2,
roll2, pitch2, yaw2, torque_roll1, torque_roll2, torque_pitch,
torque_yaw = float_values
167
168             # Convert roll, pitch and yaw angles to radians
169             R1, P1, W1, R2, P2, W2 = map(math.radians, [roll1,
pitch1, yaw1, roll2, pitch2, yaw2])
170             roll2, pitch2, yaw2 =
rotation_function(roll2, pitch2, yaw2)
171
172             # Send all data lines to the client
173             for client in clients:
174                 try:
175                     client.sendall(data + b"\n")
176                 except socket.error as e:
177                     print(f"Error sending data to client: {e}")
178                     clients.remove(client)
179                     client.close()
180             else:
181                 print("Data not received")
182
183         except Exception as e:
184             print(f"Error: {e}")
185             if client_socket in clients:
186                 clients.remove(client_socket)
187                 client_socket.close()
188             break
189
190 def accept_clients():
191     while True:
192         client_socket, addr = server_socket.accept()
193         print(f"Client connected: {addr}")
194         clients.append(client_socket)
195         client_thread =
threading.Thread(target=data_reading_sending,
args=(client_socket,))
196         client_thread.start()
197
198 accept_thread = threading.Thread(target=accept_clients)
199 accept_thread.start()
200
201
202 # -----
203 # Command sending
204 # -----
205
206 # Send commands to the robot
207 def send_ur_script(command):
208     robot_socket.send(("{}\n".format(command)).encode())
209
210 def receive_response(t):
211     try:
212         print("Waiting time: " + str(t))
213         time.sleep(t)

```

```

214
215     except socket.error as e:
216         print(f"Error receiving data from the robot: {e}")
217         exit(1) #Exit code not zero to indicate the error
218
219 # -----
220 # Key press detection
221 # -----
222
223 # We set the default value None for the pressed key.
224 pressed_key = None
225
226 # We define a series of functions that allow us to read if there
227 # is any key pressed and which one it is.
228 def which_key(key):
229     return str(key) if key else None
230
231 def key_status():
232     def on_press(key):
233         global pressed_key
234         pressed_key = which_key(key)
235
236     def on_release(key):
237         global pressed_key
238         pressed_key = None
239
240     # Start the listener to detect key presses
241     with kb.Listener(on_press=on_press, on_release=on_release) as
listener:
242         listener.join()
243
244 # Function to run the listener in a separate thread
245 def start_listener():
246     suture_thread = threading.Thread(target=key_status)
247     suture_thread.start()
248
249 # Start the listener process in a separate thread
250 start_listener()
251
252 # -----
253 # Angle vector conversion
254 # -----
255
256 def angle_vector_conversion(R2, P2, W2):
257
258     rpy = [R2, P2, W2]
259     poseB = xyzrpw_2_pose([0, 0, 0, rpy[0], rpy[1], rpy[2]])
260
261     # Conversion
262     Orientation = poseB.Rot33() # This function stores
263                                # the array transposed
264     R = array(Orientation).T # I have to transpose again to
265                             # obtain the correct array
266     angvec = tr2angvec(R, unit="deg")
267
268     roll = radians(angvec[0] * angvec[1][0])
269     pitch = radians(angvec[0] * angvec[1][1])
270     yaw = radians(angvec[0] * angvec[1][2])

```

```

271 |
272 |     return roll, pitch, yaw
273 |
274 | # -----
275 | # Rotation function
276 | # -----
277 |
278 | def rotation_function(old_roll2, old_pitch2, old_yaw2):
279 |
280 |     x_angle = 90
281 |     y_angle = 180
282 |     z_angle = 90
283 |
284 |     roll2 = (old_pitch2 + y_angle) % 360
285 |     pitch2 = -(old_roll2 + x_angle) % 360
286 |     yaw2 = (old_yaw2 + z_angle) % 360
287 |
288 |     return roll2, pitch2, yaw2
289 |
290 | # ----- #
291 | # ----- #
292 | # ----- #
293 | #             MAIN PROGRAM             #
294 | # ----- #
295 | # ----- #
296 | # ----- #
297 |
298 |
299 | # -----
300 | # Variable definition
301 | # -----
302 |
303 | p_x = 0.35
304 | p_y = -0.11
305 | p_z = 0.119
306 |
307 | rx = radians(180)
308 | ry = radians(0)
309 | rz = radians(0)
310 |
311 | time_init = 6
312 | timel = 6
313 |
314 | # -----
315 | # Movement definition
316 | # -----
317 |
318 | move_to_init = f"move1(p[{p_x},{p_y},{p_z},{rx},{ry},{rz}], a=1.2,
319 |     v=0.25, t={time_init}, r=0)"
320 |
321 | # -----
322 | # Suture process
323 | # -----
324 |
325 | def suture_process():
326 |
327 |     global s1_1, s2_1, roll1, pitch1, yaw1, s1_2, s2_2, roll2,
328 |     pitch2, yaw2, torque_roll1, torque_roll2, torque_pitch,

```

```

torque_yaw, R1, R2, P1, P2, W1, W2

327
328     if robot_is_connected:
329
330         # Set the TCP tool
331         print("TCP")
332         send_ur_script(set_TCP)
333         receive_response(1)
334
335         # Send the move_to_init command
336         print("Move to Init RoboDK")
337         send_ur_script(move_to_init)
338         receive_response(time_init)
339
340         p_x = 0.35
341         p_y = -0.11
342         p_z = 0.119
343
344         # Security variable
345         time.sleep(5)
346         security = True
347
348         # Up-down variable
349         ap_up_count = False
350         ap_down_count = False
351
352         while True:
353
354             # We define the RPY angles for the RoboDK simulation
355             R_gripper_rel = R1 - (R2 + pi / 2)
356             P_gripper_rel = P1 - P2
357             W_gripper_rel = W1 - W2
358
359             while (3 <= (roll2 - 180) <= 357 or 3 <= pitch2 <= 357 or
360 3 <= yaw2 <= 357): # Security loop, to avoid abrupt movements
361
362                 if security == True: # The first time we enter into
363                     # the security function
364                     time.sleep(0.1)
365                 else:
366                     break # Avoid the loop every time the function
367                     # is called except the first one
368
369             if tecla_presionada == None: # If there is no pressed key
370
371                 if s1_1 and not s2_1: # Close gripper
372                     needle.setParentStatic(gripper) # Static to
373                     # maintain pose
374
375                 elif not s1_1 and s2_1: # Open gripper
376                     needle.setParentStatic(base)
377
378                 elif not s1_2 and s2_2: # Move Up
379                     if ap_up_count == False:
380                         approach_up = robot.Pose()
381                     else:
382                         approach_up = robot.Pose() * transl(0, -10, 0)

```

```

383
384     # Definition of the new tool xyz after the
385     # movement, according to the robot simulation
386     # position
387     p_x = (approach_up[0, 3]) / 1000
388     p_y = (approach_up[1, 3]) / 1000
389     p_z = (approach_up[2, 3]) / 1000
390
391     robot.MoveL(approach_up, False)
392     ap_up_count = True
393     ap_down_count = False
394     time.sleep(0.1)
395
396     elif s1_2 and not s2_2:          # Move down
397
398         if ap_down_count == False:
399             approach_down = robot.Pose()
400
401         else:
402             approach_down = robot.Pose() * transl(0, 10, 0)
403
404         # Definition of the new tool xyz after the
405         # movement, according to the robot simulation
406         # position
407         p_x = (approach_down[0, 3]) / 1000
408         p_y = (approach_down[1, 3]) / 1000
409         p_z = (approach_down[2, 3]) / 1000
410
411         robot.MoveL(approach_down, False)
412         ap_up_count = False
413         ap_down_count = True
414         time.sleep(0.1)
415
416         # Conversion of the RPY angles into Angle Vector
417         roll, pitch, yaw = angle_vector_conversion(roll2, pitch2,
418 yaw2)
419
420         Xg, Yg, Zg, Xr, Yr, Zr = read_RDK()
421
422         # Move the simulation robot
423         endowrist = transl(Xr, Yr, Zr) * rotz(W2) * roty(P2) *
424 rotx(R2)
425         gripper_pose = transl(Xg, Yg, Zg) * rotx(pi / 2) *
426 rotz(W_gripper_rel) * roty(P_gripper_rel) * rotx(R_gripper_rel)
427
428         gripper.setPose(gripper_pose)
429
430         if robot.MoveL_Test(robot.Joints(), endowrist) == 0:
431             robot.MoveL(endowrist, True)
432
433         if robot_is_connected:
434             # POSE definition
435             move_to_rotation =
436 f"move1(p[{p_x},{p_y},{p_z},{roll},{pitch},{yaw}], a=1.2,
437 v=0.25, t={timel}, r=0)"
438
439             # Move the UR5e robot
440             print("Move to Rotation position")

```



```

436         send_ur_script(move_to_rotation)
437         receive_response(timel)
438
439         # Set the security variable False in order to
440         # avoid the security loop
441         security = False
442
443         # Break the loop if desired
444         if tecla_presionada == "'q'":
445             break
446
447         if robot_is_connected:
448             # Close the socket connection when done
449             robot_socket.close()
450
451 # Start printing the received values in a separate thread
452 print_thread = threading.Thread(target=suture_process)
453 print_thread.daemon = True
454 print_thread.start()
455
456 # Keep the program running
457 try:
458     while True:
459         pass
460 except KeyboardInterrupt:
461     arduino.close()
462     server_socket.close()
463     print("Server closed.")

```

Listing 1: Main Python code

.2 TKinter Python code

```

1  # -----
2  # Import of libraries
3  # -----
4
5  import socket
6  import tkinter as tk
7  import threading
8  from pynput import keyboard as kb
9  from robolink import *
10 from robodk import *
11 from gtts import gTTS
12 import tempfile
13 import pygame
14
15 # -----
16 # Connection
17 # -----
18
19 # Client socket configuration
20 host = 'localhost'
21 port = 9999
22
23 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
24

```

```

25 # Global variables to store received values
26 s1_1, s2_1, s1_2, s2_2, roll1, pitch1, yaw1, roll2, pitch2, yaw2,
    torque_roll1, torque_roll2, torque_pitch, torque_yaw = [0] * 14
27
28 def connect_to_server():
29     try:
30         client_socket.connect((host, port))
31         print("Connected to the server")
32     except socket.error as e:
33         print(f"Error connecting to the server: {e}")
34         exit(1)
35
36 # Connect to the server
37 connect_to_server()
38
39 # -----
40 # Voice settings
41 # -----
42
43 pygame.mixer.init()
44 def create_audio(i):
45     torque_names = ["Roll 1", "Roll 2", "Pitch", "Yaw"]
46     text = "Threshold reached in {}".format(torque_names[i])
47     filename = 'audio.mp3'
48     tts = gTTS(text)
49
50     temp_file = tempfile.NamedTemporaryFile(suffix=".mp3",
51 delete=False)
52     filename = temp_file.name
53     tts.save(filename)
54
55     # Wait until the alarm sound ends
56     while pygame.mixer.music.get_busy():
57         continue
58
59     # Then play the spoken text
60     pygame.mixer.music.load(filename)
61     pygame.mixer.music.play()
62
63     root.after(1000, lambda: temp_file.close())
64
65 # -----
66 # TKinter root
67 # -----
68
69 # Create the main Tk window
70 root = tk.Tk()
71 root.title("Suture process")
72
73 def create_window():
74
75     global window
76     window = tk.Canvas(root, borderwidth=0, bg="#ECECEC",
77 cursor="arrow", height=448, width=350)
78     window.pack()
79     return window

```

```

80
81 # Create a function to create and update the vertical bars
82 def updateBars(canvas, values):
83     canvas.delete("bars")
84     for i, value in enumerate(values):
85         x = 115 + value * 10
86         y = 250 + i * 50
87
88         if x <= 250:
89             canvas.create_rectangle(115, y, x, y+30, fill="blue",
90 tags="bars")
91         elif x < 315:
92             canvas.create_rectangle(115, y, x, y+30,
93 fill="firebrick", tags="bars")
94             create_audio(i)
95         elif x >= 334:
96             canvas.create_rectangle(115, y, 315, y+30,
97 fill="firebrick", tags="bars")
98             create_audio(i)
99
100     time.sleep(0.1)
101
102 def window_command(window):
103
104     "This function creates the main window"
105
106     window.create_text(64, 35, text="Instruction", fill="black",
107 font=("times 13 bold"))
108     window.create_rectangle(30, 50, 325, 107, fill="white",
109 outline="black")
110     window.create_text(85, 122, text="Gripper Orientation:",
111 fill="black", font=("times 13 bold"))
112     window.create_rectangle(30, 137, 325, 172, fill="white",
113 outline="black")
114     window.create_text(93, 187, text="Endowrist Orientation:",
115 fill="black", font=("times 13 bold"))
116     window.create_rectangle(30, 202, 325, 237, fill="white",
117 outline="black")
118     window.create_rectangle(20, 243, 335, 438, fill="grey88",
119 outline="grey88")
120
121     torques = ["Torque Roll 1", "Torque Roll 2", "Torque Pitch",
122 "Torque Yaw"]
123     for i in range(4):
124
125         x = 115 + 209
126         y = 250 + i * 50
127         window.create_rectangle(115, y, x, y+30, fill="grey97")
128         window.create_text(66, y+15, text=torques[i], font=("times
129 13 bold"))
130
131     # Initial values for the bars (you can change these)
132     values = [0, 0, 0, 0]
133     updateBars(window, values) # Draw initial bars
134
135 window_command(window)
136
137

```

```

126 # Create a label to display the text
127 text_label = tk.Label(window, text="Waiting...", wraplength=300,
    bg="white") # Adjust wraplength as needed
128 text_label.place(relx=0.5, rely=0.178, anchor="center")
129 text_label_pose_gripper = tk.Label(window, text="Waiting...",
    wraplength=300, bg="white") # Adjust wraplength as needed
130 text_label_pose_gripper.place(relx=0.5, rely=0.34, anchor="center")
131 text_label_pose_endo = tk.Label(window, text="Waiting...",
    wraplength=300, bg="white") # Adjust wraplength as needed
132 text_label_pose_endo.place(relx=0.5, rely=0.485, anchor="center")
133
134 # -----
135 # Rotation function
136 # -----
137
138 def rotation_function(old_roll2, old_pitch2, old_yaw2):
139
140     x_angle = 90
141     y_angle = 180
142     z_angle = 90
143
144     roll2 = (old_pitch2 + y_angle) % 360
145     pitch2 = -(old_roll2 + x_angle) % 360
146     yaw2 = (old_yaw2 + z_angle) % 360
147
148     return roll2, pitch2, yaw2
149
150 # -----
151 # Data receiving
152 # -----
153
154 def receive_data():
155     global s1_1, s2_1, s1_2, s2_2, roll1, pitch1, yaw1, roll2,
    pitch2, yaw2, torque_roll1, torque_roll2, torque_pitch,
    torque_yaw
156
157     while True:
158         try:
159             data = client_socket.recv(1024).decode('utf-8').strip()
160             if data:
161                 # Divide the message into data lines
162                 lines = data.split('\n')
163                 float_values = [float(line.strip()) for line in
    lines]
164
165                 # Data assignment
166                 s1_1 = float_values[0]
167                 s2_1 = float_values[1]
168                 roll1 = float_values[2]
169                 pitch1 = float_values[3]
170                 yaw1 = float_values[4]
171                 s1_2 = float_values[5]
172                 s2_2 = float_values[6]
173                 roll2 = float_values[7]
174                 pitch2 = float_values[8]
175                 yaw2 = float_values[9]
176                 torque_roll1 = float_values[10]
177                 torque_roll2 = float_values[11]

```

```

178         torque_pitch = float_values[12]
179         torque_yaw = float_values[13]
180
181
182         roll2, pitch2, yaw2 =
rotation_function(roll2, pitch2, yaw2)
183
184     except socket.error as e:
185         print(f"Error recibiendo datos: {e}")
186         client_socket.close()
187         break
188
189
190 # Recieving data in separate threadings
191 data_thread = threading.Thread(target=receive_data)
192 data_thread.daemon = True
193 data_thread.start()
194
195 # -----
196 # Pressed key detection
197 # -----
198
199 # We set as default the value None for the key pressed.
200 tecla_presionada = None
201
202 # We define a series of functions that allow us to read if there
203 # is any key pressed, and what it is.
204 def which_key(tecla):
205
206     return str(tecla) if tecla else None
207
208 def key_status():
209
210     def on_press(key):
211
212         global tecla_presionada
213         tecla_presionada = which_key(key)
214
215     def on_release(key):
216         global tecla_presionada
217         tecla_presionada = None
218
219     # Starting the listener to detect keystroke
220     with kb.Listener(on_press=on_press, on_release=on_release) as
escuchador:
221         escuchador.join()
222
223 # Function to run the listener in a separate thread
224 def start_listener():
225
226     suture_thread = threading.Thread(target=key_status)
227     suture_thread.start()
228
229 # Start listener process in a separate thread
230 start_listener()
231
232 # -----
233 # Suture process

```

```

234 # -----
235
236 def suture_process():
237
238     time.sleep(8)
239     # Security variable
240     security=True
241
242     while True:
243
244         while (3 <= (roll2-180) <= 357 or 3 <= pitch2 <= 357 or 3
245 <= yaw2 <= 357): # Security loop, to avoid abrupt movements
246
247             if security==True: # The first time we enter into
248                 # the security function
249
250                 text_label.config(text="Please, align the RPY with
251 the (0,0,0)")
252                 text_label_pose_gripper.config(text=" R=
253 "+str(round((roll1)%360,0))+ " P= "+str(round(pitch1))+
254 W= "+str(round(yaw1)))
255                 text_label_pose_endo.config(text=" R=
256 "+str(round((roll2)%360,0))+ " P= "+str(round(pitch2))+
257 W= "+str(round(yaw2)))
258                 time.sleep(0.1)
259
260             else:
261
262                 break # Avoid the loop every time the
263                 # function is called except the first one
264
265             if tecla_presionada==None: # If there is no pressed key
266
267                 if s1_1 and not s2_1: # Close gripper
268                     text_label.config(text="Close Gripper")
269
270                 elif not s1_1 and s2_1: # Open gripper
271                     text_label.config(text="Open Gripper")
272
273                 elif not s1_2 and s2_2: # Move Up
274                     text_label.config(text="Gripper move up")
275
276                 elif s1_2 and not s2_2: # Move down
277                     text_label.config(text="Gripper move down")
278
279             # Set the labels
280             text_label_pose_gripper.config(text=" R=
281 "+str(round(roll1,0))+ " P= "+str(round(pitch1))+ " W=
282 "+str(round(yaw1)))
283             text_label_pose_endo.config(text=" R=
284 "+str(round(roll2,0))+ " P= "+str(round(pitch2,0))+ " W=
285 "+str(round(yaw2,0)))
286             text_label.config(text="Moving the UR5e arm...")
287
288             torque_values =
289 [torque_roll1,torque_roll2,torque_pitch,torque_yaw]
290             actualizar_barras(finestra, torque_values)
291
292

```

```

281         # Set the security variable False in order to
282         # avoid the security loop
283         security = False
284
285         # Break the loop if desired
286         if tecla_presionada=="'q'":
287             text_label.config(text="Surgery completed!")
288             text_label_pose_gripper.config(text="Have a nice day
289 :)")
290             text_label_pose_endo.config(text="Take a rest, you
291 deserve it!")
292             break
293 keyboard_thread = threading.Thread(target=suture_process)
294 keyboard_thread.daemon = True
295 keyboard_thread.start()
296 # Start the GUI event loop
297 root.mainloop()

```

Listing 2: TKinter Python code

.3 Pen User Interface Arduino code

```

1  #include <esp_now.h>
2  #include <WiFi.h>
3  #include "src/RoboticsUB.h"
4
5  const int PIN_IMU_INT = 18;
6  const int PIN_S1 = 14;
7  const int PIN_S2 = 27;
8  const int pinVibrationMotor = 23;
9
10 float NewValueRoll = 0;
11 float OldValueRoll = 0;
12 float roll=0;
13 float valRoll=0;
14
15 float NewValuePitch=0;
16 float OldValuePitch=0;
17 float pitch=0;
18 float valPitch=0;
19
20 float NewValueYaw=0;
21 float OldValueYaw=0;
22 float yaw=0;
23 float valYaw=0;
24 float zero_yaw=90;
25
26 int int_roll;
27 int int_pitch;
28 int int_yaw;
29
30 float torque_roll1 = 0;
31 float torque_roll2 = 0;
32 float torque_pitch = 0;
33 float torque_yaw = 0;

```

```

34 float torque_vibr_roll1 = 0;
35 float torque_vibr_roll2 = 0;
36 float torque_vibr_pitch = 0;
37 float torque_vibr_yaw = 0;
38 float max_torque = 0;
39
40 int s1Status = HIGH;
41 int s2Status = HIGH;
42
43 enum class Command : byte
44 {
45     GET_RPW = 1
46 };
47
48 Command command = Command::GET_RPW;
49
50 IMU imu;
51
52 float *rpw;
53
54 //Servo's MAC Address
55 uint8_t ServoAddress[] = {0x24, 0xdc, 0xc3, 0xc6, 0xbd, 0x14};
56
57 //Computer's MAC Address
58 uint8_t ComputerAddress[] = {0x94, 0xb9, 0x7e, 0xe5, 0x9c, 0xc8};
59
60 // Must match the receiver structure
61 typedef struct struct_message {
62     int id; // must be unique for each sender board
63     int s1Status;
64     int s2Status;
65     int roll;
66     int pitch;
67     int yaw;
68     float torque_yaw;
69     float torque_pitch;
70     float torque_roll1;
71     float torque_roll2;
72 } struct_message;
73
74
75 // Create a struct_message called myData
76 struct_message myData;
77
78 // Create a structure to hold the readings from each board
79
80 struct_message Servomotors;
81
82 // Create peer interface for the servos
83 esp_now_peer_info_t peerInfo;
84
85 // Create peer interface for the computer
86 esp_now_peer_info_t peerInfo_computer;
87
88 // Create an array
89 struct_message boardsStruct[1] = {Servomotors};
90
91 // callback function that will be executed when data is received

```



```

92 void OnDataRecv(const uint8_t * mac_addr, const uint8_t
    *incomingData, int len) {
93
94     char macStr[18];
95     snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
96              mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3],
97              mac_addr[4], mac_addr[5]);
98
99     memcpy(&myData, incomingData, sizeof(myData));
100
101     boardsStruct[0].torque_yaw = myData.torque_yaw;
102     boardsStruct[0].torque_pitch = myData.torque_pitch;
103     boardsStruct[0].torque_roll1 = myData.torque_roll1;
104     boardsStruct[0].torque_roll2 = myData.torque_roll2;
105 }
106 // callback when data is sent
107 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t
    status) {
108
109 }
110
111 void setup() {
112     // Init Serial Monitor
113     Serial.begin(115200);
114
115     pinMode(PIN_IMU_INT, INPUT_PULLUP);
116     pinMode(PIN_S1, INPUT);
117     pinMode(PIN_S2, INPUT);
118
119     imu.Install();
120
121     // Set device as a Wi-Fi Station
122     WiFi.mode(WIFI_STA);
123
124     // Init ESP-NOW
125     if (esp_now_init() != ESP_OK) {
126         Serial.println("Error initializing ESP-NOW");
127         return;
128     }
129
130     // Once ESPNow is successfully Init, we will register for Send
131     // CB to get the status of Trasnmitted packet
132     esp_now_register_send_cb(OnDataSent);
133
134     // Register peer for the servo
135     memcpy(peerInfo.peer_addr, ServoAddress, 6);
136     peerInfo.channel = 0;
137     peerInfo.encrypt = false;
138
139     // Add peer
140     if (esp_now_add_peer(&peerInfo) != ESP_OK){
141         Serial.println("Failed to add peer");
142         return;
143     }
144
145     // Register peer for the computer
146     memcpy(peerInfo_computer.peer_addr, ComputerAddress, 6);

```

```

147  peerInfo.channel = 0;
148  peerInfo.encrypt = false;
149
150  // Add peer
151  if (esp_now_add_peer(&peerInfo_computer) != ESP_OK){
152      Serial.println("Failed to add peer");
153      return;
154  }
155
156  // Once ESPNow is successfully Init, we will register for recv
157  // CB to get recv packer info
158  esp_now_register_recv_cb(OnDataRecv);
159
160 }
161
162 void loop() {
163
164     if (digitalRead(PIN_IMU_INT) == HIGH)
165     {
166         imu.ReadSensor();
167         rpw = imu.GetRPW();
168     }
169
170     s1Status = digitalRead(PIN_S1);
171     s2Status = digitalRead(PIN_S2);
172
173     //Angle protection
174
175
176     NewValueRoll=rpw[0];
177     valRoll=abs(NewValueRoll-OldValueRoll);
178     if (valRoll>10 && valRoll<350) {
179         roll=OldValueRoll;
180     }
181     else {
182         roll=NewValueRoll;
183     }
184     OldValueRoll=roll;
185
186     int_roll=int(roll);
187
188     NewValuePitch=rpw[1];
189     valPitch=abs(NewValuePitch-OldValuePitch);
190     if (valPitch>10 && valPitch<350) {
191         pitch=OldValuePitch;
192     }
193     else {
194         pitch=NewValuePitch;
195     }
196     OldValuePitch=pitch;
197     int_pitch=int(pitch);
198
199     NewValueYaw=rpw[2];
200     valYaw=abs(NewValueYaw-OldValueYaw);
201     if (valYaw>10 && valYaw<350) {
202         yaw=OldValueYaw;
203     }
204     else {

```

```

205     yaw=NewValueYaw;
206 }
207 OldValueYaw=yaw;
208 int_yaw=int(yaw);
209
210
211 torque_roll1 = boardsStruct[0].torque_roll1;
212 torque_roll2 = boardsStruct[0].torque_roll2;
213 torque_pitch = boardsStruct[0].torque_pitch;
214 torque_yaw = boardsStruct[0].torque_yaw;
215
216 torque_vibr_roll1 = map(torque_roll1,0,20,0,255);
217 torque_vibr_roll2 = map(torque_roll2,0,20,0,255);
218 torque_vibr_pitch = map(torque_pitch,0,20,0,255);
219 torque_vibr_yaw = map(torque_yaw,0,20,0,255);
220
221 max_torque = max(max(torque_vibr_roll1, torque_vibr_roll2),
222     max(torque_vibr_pitch, torque_vibr_yaw));
223
224 if (max_torque > 20) {
225     analogWrite(pinVibrationMotor, max_torque); // Set the PWM
226 }
227
228 // Set values to send
229 myData.id = 1;
230 myData.s1Status = s1Status;
231 myData.s2Status = s2Status;
232 myData.roll = int_roll;
233 myData.pitch=int_pitch;
234 myData.yaw=int_yaw;
235 myData.torque_roll1 = torque_roll1;
236 myData.torque_roll2 = torque_roll2;
237 myData.torque_pitch = torque_pitch;
238 myData.torque_yaw = torque_yaw;
239
240 Serial.println(torque_pitch);
241
242 // Send message via ESP-NOW
243 esp_err_t result = esp_now_send(ServoAddress, (uint8_t *)
244     &myData, sizeof(myData));
245 esp_err_t result_computer = esp_now_send(ComputerAddress,
246     (uint8_t *) &myData, sizeof(myData));
247
248 delay(10);
249 }

```

Listing 3: Pen User Interface Arduino Code

.4 Forearm Master Arduino code

```

1 #include "src/RoboticsUB.h"
2 #include <esp_now.h>
3 #include <WiFi.h>
4
5 const int PIN_IMU_INT = 18;
6 const int PIN_S1 = 14;

```

```

7  const int PIN_S2 = 27;
8
9  float NewValueRoll = 0;
10 float OldValueRoll = 0;
11 float roll=0;
12 float valRoll=0;
13
14 float NewValuePitch=0;
15 float OldValuePitch=0;
16 float pitch=0;
17 float valPitch=0;
18
19 float NewValueYaw=0;
20 float OldValueYaw=0;
21 float yaw=0;
22 float valYaw=0;
23 float zero_yaw=90;
24 int int_roll;
25 int int_pitch;
26 int int_yaw;
27
28
29 enum class Command : byte
30 {
31     GET_RPW = 1
32 };
33
34 Command command = Command::GET_RPW;
35
36 IMU imu;
37
38 float *rpw;
39 int s1Status = HIGH;
40 int s2Status = HIGH;
41
42 // Servo's MAC Adress
43
44 uint8_t ServoAddress[] = {0x24, 0xdc, 0xc3, 0xc6, 0xbd, 0x14};
45
46 //Computer's MAC Address
47 uint8_t ComputerAddress[] = {0x94, 0xb9, 0x7e, 0xe5, 0x9c, 0xc8};
48
49 // Must match the receiver structure
50 typedef struct struct_message {
51     int id; // must be unique for each sender board
52     int s1Status;
53     int s2Status;
54     int roll;
55     int pitch;
56     int yaw;
57     float torque_yaw;
58     float torque_pitch;
59     float torque_roll1;
60     float torque_roll2;
61 } struct_message;
62
63
64 // Create a struct_message called myData

```

```

65 struct_message myData;
66
67 // Create peer interface for the servos
68 esp_now_peer_info_t peerInfo;
69
70 // Create peer interface for the computer
71 esp_now_peer_info_t peerInfo_computer;
72
73 // callback when data is sent
74 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t
    status) {
75     Serial.print("\r\nLast Packet Send Status:\t");
76     Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery
        Success" : "Delivery Fail");
77 }
78
79 void setup() {
80     // Init Serial Monitor
81     Serial.begin(115200);
82
83     pinMode(PIN_IMU_INT, INPUT_PULLUP);
84     pinMode(PIN_S1, INPUT);
85     pinMode(PIN_S2, INPUT);
86
87     imu.Install();
88
89     // Set device as a Wi-Fi Station
90     WiFi.mode(WIFI_STA);
91
92     // Init ESP-NOW
93     if (esp_now_init() != ESP_OK) {
94         Serial.println("Error initializing ESP-NOW");
95         return;
96     }
97
98     // Once ESPNow is successfully Init, we will register for Send
99     // CB to get the status of Trasnmitted packet
100     esp_now_register_send_cb(OnDataSent);
101
102     // Register peer for the servo
103     memcpy(peerInfo.peer_addr, ServoAddress, 6);
104     peerInfo.channel = 0;
105     peerInfo.encrypt = false;
106
107     // Add peer
108     if (esp_now_add_peer(&peerInfo) != ESP_OK){
109         Serial.println("Failed to add peer");
110         return;
111     }
112
113     // Register peer for the computer
114     memcpy(peerInfo_computer.peer_addr, ComputerAddress, 6);
115     peerInfo_computer.channel = 0;
116     peerInfo_computer.encrypt = false;
117
118     // Add peer
119     if (esp_now_add_peer(&peerInfo_computer) != ESP_OK){
120         Serial.println("Failed to add peer");

```

```

121     return;
122 }
123
124 }
125
126 void loop() {
127
128     if (digitalRead(PIN_IMU_INT) == HIGH)
129     {
130         imu.ReadSensor();
131         rpw = imu.GetRPW();
132     }
133
134     s1Status = digitalRead(PIN_S1);
135     s2Status = digitalRead(PIN_S2);
136
137     //Angle protection
138     NewValueRoll=rpw[0];
139     valRoll=abs(NewValueRoll-OldValueRoll);
140     if (valRoll>10 && valRoll<350) {
141         roll=OldValueRoll;
142     }
143     else {
144         roll=NewValueRoll;
145     }
146     OldValueRoll=roll;
147
148     int_roll=int(roll);
149
150     NewValuePitch=rpw[1];
151     valPitch=abs(NewValuePitch-OldValuePitch);
152     if (valPitch>10 && valPitch<350) {
153         pitch=OldValuePitch;
154     }
155     else {
156         pitch=NewValuePitch;
157     }
158     OldValuePitch=pitch;
159     int_pitch=int(pitch);
160
161     NewValueYaw=rpw[2];
162     valYaw=abs(NewValueYaw-OldValueYaw);
163     if (valYaw>10 && valYaw<350) {
164         yaw=OldValueYaw;
165     }
166     else {
167         yaw=NewValueYaw;
168     }
169     OldValueYaw=yaw;
170     int_yaw=int(yaw);
171
172     // Set values to send
173     myData.id = 2;
174     myData.s1Status = s1Status;
175     myData.s2Status = s2Status;
176     myData.roll = int_roll;
177     myData.pitch=int_pitch;
178     myData.yaw=int_yaw;

```

```

179 // Send message via ESP-NOW
180 esp_err_t result = esp_now_send(ServoAddress, (uint8_t *)
181   &myData, sizeof(myData));
182
183 esp_err_t result_computer = esp_now_send(ComputerAddress,
184   (uint8_t *) &myData, sizeof(myData));
185
186 if (result == ESP_OK) {
187   Serial.println("Sent with success");
188 }
189 else {
190   Serial.println("Error sending the data");
191 }
192
193 if (result_computer == ESP_OK) {
194   Serial.println("Sent with success ");
195 }
196 else {
197   Serial.println("Error sending the data");
198 }
199 delay(10);
200 }

```

Listing 4: Forearm Master Arduino code

.5 EndoWrist Motion Panel Arduino code

```

1 // Include the necessary libraries
2 #ifdef ESP32
3   #include <WiFi.h>
4 #else
5   #include <ESP8266WiFi.h>
6 #endif
7
8 #include <esp_now.h>
9 #include <WiFi.h>
10 #include <ESP32Servo.h>
11
12 // Master's MAC Address
13 uint8_t masterMacAddress[] = {0x24, 0xdc, 0xc3, 0xc6, 0xbc, 0x28};
14
15 //variables de identificacion para los servos
16 Servo servo_yaw;
17 Servo servo_pitch;
18 Servo servo_roll1;
19 Servo servo_roll2;
20
21 //variable para almacenar el torque
22 float torque_yaw=0;
23 float torque_pitch=0;
24 float torque_roll1=0;
25 float torque_roll2=0;
26
27 // variables para el angle protection
28 float roll = 0;

```

```

29 float roll2 = 0;
30 float roll_gir = 0;
31 float roll_gir2 = 0;
32 float roll_rel = 0;
33
34 float pitch = 0;
35 float pitch2 = 0;
36 float pitch_gir = 0;
37 float pitch_rel = 0;
38
39 float yaw = 0;
40 float yaw2 = 0;
41 float yaw_gir = 0;
42 float yaw_rel = 0;
43
44 int s1Status = HIGH;
45 int s2Status = HIGH;
46
47 bool gripper_open= false;
48 bool security = true;
49
50 // variables to store the read data corresponding to the torque
51 // of the servos
52 float derivativeCurrent_yaw=0;
53 float derivativeCurrent_pitch=0;
54 float derivativeCurrent_roll1=0;
55 float derivativeCurrent_roll2=0;
56
57 float previousfinal_yaw=0;
58 float previousfinal_pitch=0;
59 float previousfinal_roll1=0;
60 float previousfinal_roll2=0;
61
62 float final_torque_yaw=0;
63 float final_torque_pitch=0;
64 float final_torque_roll1=0;
65 float final_torque_roll2=0;
66
67 float servoPosition = 0;
68
69 //pin identification variables
70 /*YAW*/
71 const int PIN_ANALOG_YAW = 36;
72 const int PIN_SIGNAL_YAW = 32;
73 /*PITCH*/
74 const int PIN_ANALOG_PITCH = 34;
75 const int PIN_SIGNAL_PITCH = 25;
76 /*ROLL*/
77 const int PIN_ANALOG_ROLL1 = 39;
78 const int PIN_SIGNAL_ROLL1 = 33;
79 /*ROLL*/
80 const int PIN_ANALOG_ROLL2 = 35;
81 const int PIN_SIGNAL_ROLL2 = 27;
82
83 //constants for shunt resistors (they are all the same)
84 const float Rshunt = 1.6;
85
86 enum class Command : byte

```



```

87 {
88     GET_RPW = 1
89 };
90
91 Command command = Command::GET_RPW;
92
93 // We define the data structure that we will receive
94 // from the master
95 typedef struct struct_message {
96     int id;
97     int s1Status;
98     int s2Status;
99     int roll;
100    int pitch;
101    int yaw;
102    float torque_yaw;
103    float torque_pitch;
104    float torque_roll1;
105    float torque_roll2;
106
107 }struct_message;
108
109 // Create a struct_message called myData
110 struct_message myData;
111
112 // Create a structure to hold the readings from each board
113 struct_message master1;
114 struct_message master2;
115
116 // Create an array
117 struct_message boardsStruct[2] = {master1, master2};
118
119 // callback function that will be executed when data is received
120 void OnDataRecv(const uint8_t * mac_addr, const uint8_t
    *incomingData, int len) {
121
122     char macStr[18];
123     snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
124             mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3],
125             mac_addr[4], mac_addr[5]);
126
127     memcpy(&myData, incomingData, sizeof(myData));
128
129     // Update the structures with the new incoming data
130     boardsStruct[myData.id-1].s1Status = myData.s1Status;
131     boardsStruct[myData.id-1].s2Status = myData.s2Status;
132     boardsStruct[myData.id-1].roll = myData.roll;
133     boardsStruct[myData.id-1].pitch = myData.pitch;
134     boardsStruct[myData.id-1].yaw = myData.yaw;
135 }
136
137 // callback when data is sent
138 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t
    status) {
139
140 }
141

```

```

142 void setup(){
143
144     Serial.begin(115200);
145     Serial.println();
146     Serial.print("ESP Board MAC Address:  ");
147     Serial.println(WiFi.macAddress());
148
149     //Set device as a Wi-Fi Station
150     WiFi.mode(WIFI_STA);
151
152     //Init ESP-NOW
153     if (esp_now_init() != ESP_OK) {
154         Serial.println("Error initializing ESP-NOW");
155         return;
156     }
157
158     // Once ESPNow is successfully Init, we will register for Send
159     // CB to get the status of Trasnmited packet
160     esp_now_register_send_cb(OnDataSent);
161
162     // Register the master
163     esp_now_peer_info_t peerInfo = {};
164     memcpy(peerInfo.peer_addr, masterMacAddress, 6);
165     peerInfo.channel = 0;
166     peerInfo.encrypt = false;
167
168     // check if the master could be added
169     if (esp_now_add_peer(&peerInfo) != ESP_OK) {
170         Serial.println("Failed to add master");
171         return;
172     }
173
174     // Once ESPNow is successfully Init, we will register for recv
175     // CB to get recv packer info
176     esp_now_register_recv_cb(OnDataRecv);
177
178     // We set all the variables related to the servo
179
180     //configure the PWMs that power the servos
181     ESP32PWM::allocateTimer(0);
182     ESP32PWM::allocateTimer(1);
183     ESP32PWM::allocateTimer(2);
184     ESP32PWM::allocateTimer(3);
185
186     //we designate a frequency to the servos
187     servo_yaw.setPeriodHertz(50);
188     servo_pitch.setPeriodHertz(50);
189     servo_roll1.setPeriodHertz(50);
190     servo_roll2.setPeriodHertz(50);
191
192     //we assign the pins to each servo
193     servo_yaw.attach(PIN_SIGNAL_YAW);
194     servo_pitch.attach(PIN_SIGNAL_PITCH);
195     servo_roll1.attach(PIN_SIGNAL_ROLL1);
196     servo_roll2.attach(PIN_SIGNAL_ROLL2);
197
198     //we configure the analog pins of the servos as inputs
199     pinMode(PIN_ANALOG_YAW, INPUT);

```

```

200 pinMode(PIN_ANALOG_PITCH, INPUT);
201 pinMode(PIN_ANALOG_ROLL1, INPUT);
202 pinMode(PIN_ANALOG_ROLL2, INPUT);
203
204 //we start the servos in the central position
205 servo_yaw.write(90);
206 servo_pitch.write(90);
207 servo_roll1.write(90);
208 servo_roll2.write(90);
209 delay(2000);
210 }
211
212 void loop(){
213
214 //we start by sending the servo current data
215 esp_err_t result = esp_now_send(masterMacAddress, (uint8_t *)
    &myData, sizeof(myData));
216
217 //we check that the data have been sent correctly
218 if (result == ESP_OK) {
219     //Serial.println("Sent with success");
220 }
221 else {
222     Serial.println("Error sending the data");
223 }
224
225 // We define the button variables
226 s1Status = boardsStruct[0].s1Status;
227 s2Status = boardsStruct[0].s2Status;
228
229 // Security function
230 if (security == true){
231     roll = boardsStruct[0].roll;
232     pitch = boardsStruct[0].pitch;
233     yaw = boardsStruct[0].yaw;
234
235     if (2 <= roll <= 358 or 2 <= pitch <= 358 or 2 <= yaw <= 358){
236         Serial.print("Roll: ");
237         Serial.print(roll);
238         Serial.print(" Pitch: ");
239         Serial.print(pitch);
240         Serial.print(" Yaw: ");
241         Serial.print(yaw);
242         Serial.println(" Please, align the RPY with the (0,0,0)
angle");
243     }
244
245     if (roll == 0 && pitch ==0 && yaw == 0) {
246
247         security = false;
248     }
249 } else{
250
251 //Protection angle of each servomotor
252
253 // Roll

```

```

256 roll = boardsStruct[0].roll; // We get the value of the master1
257 roll2 = boardsStruct[1].roll; // We get the value of the master2
258
259
260 if (roll > 180 && roll <= 360) { // Re-escape the values
261     roll = roll - 360;           // of the master1
262 }
263 if (roll2 > 180 && roll2 <= 360) { // Re-escape the values
264     roll2 = roll2 - 360;         // of the master2
265 }
266
267 roll_rel = roll - roll2;
268 roll_gir = roll_rel + 90;
269 roll_gir2 = -roll_rel + 90;
270
271 if (s1Status == LOW && s2Status == HIGH){
272     gripper_open = true; // Turn the variable into True, in order
273                         // to maintain the gripper open
274     roll_gir += 30;
275 }
276
277 else if (s2Status == LOW && s1Status == HIGH){
278     gripper_open = false; // Turn the variable into False, in order
279                          // to maintain the gripper closed
280     roll_gir -= 15;
281 }
282
283
284 if ( -90 < roll_rel < 90){
285     servo_roll1.write(roll_gir-10);
286     servo_roll2.write(roll_gir2-10);
287 }
288
289 // Pitch
290 pitch=boardsStruct[0].pitch; // We get the value of master1
291 pitch2=boardsStruct[1].pitch; // We get the value of master2
292
293 if (pitch > 180 && pitch <= 360) { // Re-escape the values
294     pitch = pitch - 360;           // of the master1
295 }
296 if (pitch2 > 180 && pitch2 <= 360) { // Re-escape the values
297     pitch2 = pitch2 - 360;         // of the master2
298 }
299
300 pitch_rel = pitch2 - pitch;
301 pitch_gir = pitch_rel + 90;
302
303 if ( -90 < pitch_rel < 90){
304     servo_pitch.write(pitch_gir);
305 }
306
307 // Yaw
308 yaw = boardsStruct[0].yaw; // We get the value of master1
309 yaw2 = boardsStruct[1].yaw; // We get the yaw value of master2
310
311 if (yaw > 180 && yaw <= 360) { // Re-escape the values
312     yaw = yaw - 360;           // of the master1
313 }

```

```

314   if (yaw2 > 180 && yaw2 <= 360) { // Re-scale the values
315       yaw2 = yaw2 - 360;           // of the master2
316   }
317
318   yaw_rel = yaw2 - yaw;
319   yaw_gir = yaw_rel + 90;
320
321   if ( -90 < yaw_rel < 90){
322       servo_yaw.write(yaw_gir);
323   }
324
325
326   // the current is calculated using the shunt resistance and we
327   // integrate the current to find out if there is an obstacle or
328   // not by observing the magnitude
329
330
331   torque_yaw = getDerivative(final_torque_yaw, PIN_ANALOG_YAW,
332                               previousfinal_yaw);
333   torque_pitch = getDerivative(final_torque_pitch,
334                                 PIN_ANALOG_PITCH, previousfinal_pitch);
335   torque_roll1 = getDerivative(final_torque_roll1,
336                                 PIN_ANALOG_ROLL1, previousfinal_roll1);
337   torque_roll2 = getDerivative(final_torque_roll2,
338                                 PIN_ANALOG_ROLL2, previousfinal_roll2);
339
340   // We send the torque values to the Master
341   myData.torque_yaw=torque_yaw;
342   myData.torque_pitch=torque_pitch;
343   myData.torque_roll1=torque_roll1;
344   myData.torque_roll2=torque_roll2;
345 }
346
347 delay(10);
348
349 }
350
351 float getDerivative(float final_torque, int analog, float
352                     previous){
353     float torque;
354     float derivative;
355     torque = getCurrent(20, analog);
356     final_torque = final_torque + torque;
357     derivative = abs(final_torque - previous);
358     previous = final_torque;
359
360     return derivative;
361 }
362
363 float getCurrent(uint32_t integrationTimeMs, int servo) {
364     // we read the ADC where the shunt is connected to
365     // obtain the current. once we have the reading,
366     // we convert the ADC data to be able to apply V/R
367     uint32_t startTime = millis();
368     float integratedCurrent = 0;
369
370     // We add up the current measurement for the set
371     // time -> integrationTimeMs (in milliseconds)

```

```

367 while (millis() < startTime + integrationTimeMs) {
368     uint16_t adcValue = analogRead(servo);
369     integratedCurrent = integratedCurrent + ((float)adcValue /
        4095.0 * 3.3) / Rshunt;
370 }
371 return integratedCurrent;
372 }

```

Listing 5: EndoWrist Motion Panel Arduino code

.6 Control and Supervision Module Arduino code

```

1  #ifdef ESP32
2      #include <WiFi.h>
3  #else
4      #include <ESP8266WiFi.h>
5  #endif
6
7  #include <esp_now.h>
8  #include <WiFi.h>
9
10 enum class Command : byte
11 {
12     GET_RPW = 1
13 };
14
15 Command command = Command::GET_RPW;
16
17 // Structure example to receive data
18 typedef struct struct_message {
19     int id; // must be unique for each sender board
20     int s1Status;
21     int s2Status;
22     int roll;
23     int pitch;
24     int yaw;
25     float torque_yaw;
26     float torque_pitch;
27     float torque_roll1;
28     float torque_roll2;
29 }struct_message;
30
31 // Create a struct_message called myData
32 struct_message myData;
33
34 // Create a structure to hold the readings from each board
35 struct_message master1;
36 struct_message master2;
37
38 // Create an array with all the structures
39 struct_message boardsStruct[2] = {master1, master2};
40
41 // callback function that will be executed when data is received
42 void OnDataRecv(const uint8_t * mac_addr, const uint8_t
43     *incomingData, int len) {
44

```

```

45 char macStr[18];
46 //Serial.print("Packet received from: ");
47 snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
48          mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3],
49          mac_addr[4], mac_addr[5]);
50 //Serial.println(macStr);
51 memcpy(&myData, incomingData, sizeof(myData));
52 //Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
53 // Update the structures with the new incoming data
54 boardsStruct[myData.id-1].s1Status = myData.s1Status;
55 boardsStruct[myData.id-1].s2Status = myData.s2Status;
56 boardsStruct[myData.id-1].roll = myData.roll;
57 boardsStruct[myData.id-1].pitch = myData.pitch;
58 boardsStruct[myData.id-1].yaw = myData.yaw;
59 boardsStruct[myData.id-1].torque_roll1 = myData.torque_roll1;
60 boardsStruct[myData.id-1].torque_roll2 = myData.torque_roll2;
61 boardsStruct[myData.id-1].torque_pitch = myData.torque_pitch;
62 boardsStruct[myData.id-1].torque_yaw = myData.torque_yaw;
63 }
64
65 void setup(){
66   Serial.begin(115200);
67   Serial.println();
68   Serial.print("ESP Board MAC Address: ");
69   Serial.println(WiFi.macAddress());
70
71   //Set device as a Wi-Fi Station
72   WiFi.mode(WIFI_STA);
73
74   //Init ESP-NOW
75   if (esp_now_init() != ESP_OK) {
76     Serial.println("Error initializing ESP-NOW");
77     return;
78   }
79
80   // Once ESPNow is successfully Init, we will register for recv
81   //CB to get recv packer info
82   esp_now_register_recv_cb(OnDataRecv);
83 }
84
85 void loop(){
86
87   if (Serial.available() > 0)
88   {
89     command = (Command)Serial.read();
90
91     switch (command)
92     {
93     case Command::GET_RPW:
94       Serial.println(boardsStruct[0].s1Status, DEC);
95       Serial.println(boardsStruct[0].s2Status, DEC);
96       Serial.println(boardsStruct[0].roll, DEC);
97       Serial.println(boardsStruct[0].pitch, DEC);
98       Serial.println(boardsStruct[0].yaw, DEC);
99       Serial.println(boardsStruct[1].s1Status, DEC);
100      Serial.println(boardsStruct[1].s2Status, DEC);
101      Serial.println(boardsStruct[1].roll, DEC);

```

```
102     Serial.println(boardsStruct[1].pitch,DEC);
103     Serial.println(boardsStruct[1].yaw,DEC);
104     Serial.println(boardsStruct[0].torque_roll1,DEC);
105     Serial.println(boardsStruct[0].torque_roll2,DEC);
106     Serial.println(boardsStruct[0].torque_pitch,DEC);
107     Serial.println(boardsStruct[0].torque_yaw,DEC);
108     break;
109
110   }
111 }
112
113 delay(10);
114
115 }
```

Listing 6: Control and Supervision Module Arduino code

.7 Pen User Interface PCB design

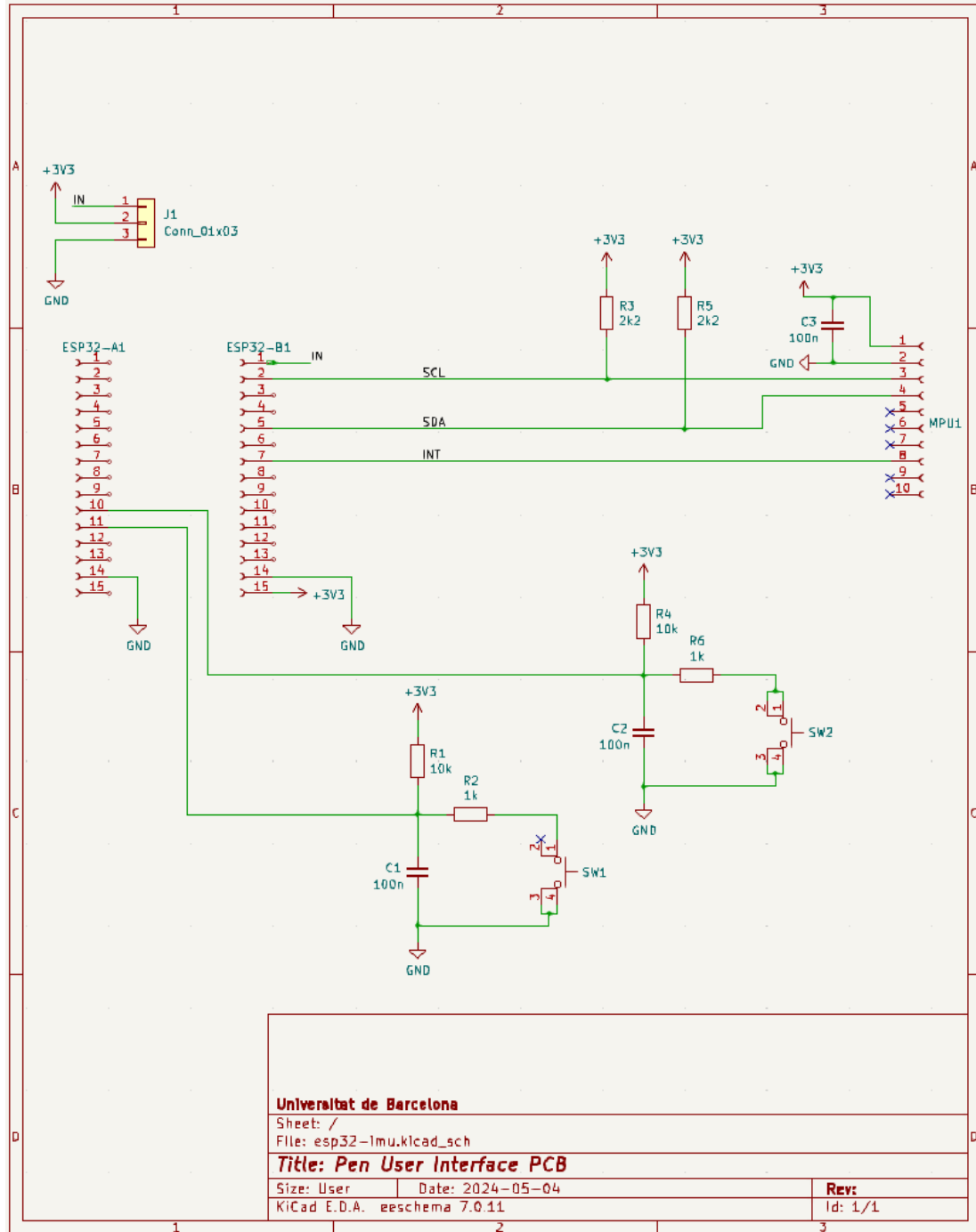


Figure 1: Pen User Interface Schematic

