Faculty of Engineering Alexandria University

Biomedical Engineering Department

Final Year Project

Robotic Arm For Surgical Assistance

Presented By:

Ahmed Zaineldin 7337
Omar Yasser 7437
Khaled Ahmed 7463
Moustafa Farouk 7590

Malak Sherif 7701
Laylan Maged 7828
Jana Abdellatif 7659
Mohamed Sherif 7511

Supervised By:

Dr. Taher Awad

Dr. Mohamed Abdelnaeem

Dr. Mahmoud Omar

Acknowledgement

First and foremost, we would like to dedicate our thanks to Allah the merciful and great for giving us the wisdom that helped us doing this project, our thanks extend to our families for supporting us throughout our university time, last and most important we thank Dr. Mohamed Ahmed Mohamed Abdelnaeem for his guidance and help throw-out the project process.

Abstract

In today's rapidly evolving technological landscape, the development of robotics is driven by the increasing needs of humanity. Robotics has become integral in enhancing the quality of life by performing tasks with precision and efficiency, often in environments that are hazardous for humans. This paper presents the design and implementation of a 6-DOF (Degree of Freedom) robotic arm intended for use as a surgical assistance device. Unlike traditional methods that require rebooting and reprogramming for each new task, our approach leverages advanced image processing and voice command technologies to offer a more flexible and intuitive control system.

The robotic arm is equipped with a camera, which provides real-time image processing capabilities, allowing the device to accurately interpret and respond to visual cues. Additionally, voice command integration enables surgeons to control the arm hands-free, enhancing the efficiency and safety of surgical procedures. This innovative control method not only improves the arm's adaptability to various tasks but also minimizes downtime and the need for manual reprogramming.

Our experimental results demonstrate the effectiveness of this approach in assisting surgeons with complex tasks, with the potential for future advancements to enable the robotic arm to perform surgeries autonomously. The integration of image processing and voice command technologies represents a significant advancement in the field of surgical robotics, offering a promising solution for enhancing surgical outcomes and patient safety.

Table Of Content

Acknowledgement.....2
Abstract.....3
Table of Content.....4
List of Figures.....6
List of Tables.....8

1. CHAPTER 1 - INTRODUCTION.....9
1.1. Definition of the Project.....9
1.2. Significance of the Project.....10
1.3. Project Objectives.....11
1.4. Project Constraints 12
1.5. Report Structure 14
2. CHAPTER 2 – LITERATURE REVIEW.....16
2.1. Background.....16
2.1.1. History of Robotic Arms.....17
2.1.2. Uses of Robotic Arms.....19
2.1.3. Surgical Robotic Arms.....20
2.2. Technological Advances in Surgical Robotics.....23
3. CHAPTER 3 - DESIGN AND ANALYSIS.....25
3.1. Proposed Design.....25
3.2. Engineering Standards.....32
3.3. Design Calculations.....33
3.3.1. DH Table.....34
3.3.2. Kinematics.....36
3.3.3. Working Volume.....37
3.3.4. Force Analysis.....41
3.4. Cost Analysis.....43
4. CHAPTER 4 - MANUFACTURING.....46
4.1. Manufacturing Process Selection.....46
4.2. Overview of Hardware/Assembly of Components.....46
4.2.1. Power Supply.....46
4.2.2. Step Down Transformer.....47
4.2.3. Motor Control Unit.....48
4.2.4. Servomotors.....49

4.2.5. USB Camera……50
4.2.6. Microcontroller……51
4.2.7. Relay Module……52
4.2.8. Electrical Magnet……53
4.3. Circuit Diagram……53
5. CHAPTER 5 - SOFTWARE……57
5.1. Kinematics and Movement Control……57
5.1.1. Initialization of the Position……57
5.1.2. Relay Control……58
5.1.3. X-Axis Mapping……59
5.1.4. Movement on X-Axis Zones……59
5.1.5. Handling Sequence……60
5.2. Computer Vision……61
5.2.1. YOLO Object Detection……61
5.2.2. Hand Guesture Detection……63
5.3. Speech Recognition……65
5.4. Machine Learning……66
6. CONCLUSIONS……69
6.1. Objectives Fulfillment……69
6.2. Future Improvments……69
6.3. Personal Conclusions……70
7. REFERENCES……72
8. APPENDIX A : SURVEY DATA……76
9. APPENDIX B: CODE LISTINS……79
10. APPENDIX C: MACHINE LEARNING RESULT……89

List of Figures

Figure 1 Evolution of the Da Vinci Surgical System [34].....18

Figure 2 PUMA 560 Robot (1985) [35].....21

Figure 3 Chosen Open Source Robotic Arm Design.....26

Figure 4 U-Shaped Servo Bracket.....28

Figure 5 Motor Housing Bracket 28

Figure 6 Arm Base Support.....29

Figure 7 End Effector Motor Bracket.....30

Figure 8 End Effector Magnet Holder.....30

Figure 9 Electronics Enclosure Base.....31

Figure 10 Kinematic Diagram.....35

Figure 11 AutoCAD Drawing of Working Area.....40

Figure 12 Python Validation of Working Area.....40

Figure 13 AutoCAD Drawing of Working Volume.....41

Figure 14 Force Distribution.....43

Figure 15 Power Supply Unit.....47

Figure 16 Step-Down Converter (XL4016).....48

Figure 17 Motor Control Unit.....49

Figure 18 LX-15D Serial Bus Servo.....50

Figure 19 USB Camera Module.....50

Figure 20 Raspberry Pi 5.....52

Figure 21 Relay Module……52

Figure 22 Electric Magnet……..53

Figure 23 Circuit Diagram……..55

Figure 24 Motor Wiring Configuration……..56

Figure 25 Survey Participants Demographics……..76

Figure 26 Proposed Applications……..77

Figure 27 Survey Result……78

Figure 28 Model Training Loss Graph……..89

Figure 29 Model Detection Result……90

List of Tables

Table 1 Advantages and Disadvantages of Robot-Assisted Surgery Versus Conventional Surgery [17].....22
Table 2 Decision Matrix For Material Selection.....26
Table 3 Engineering Standards.....32
Table 4 DH Table.....36
Table 5 Joint Angles.....36
Table 6 Joint Angle Configurations for Working Area Calculation.....38
Table 7 Fixed and Moving Links for Reachable Area Determination....39
Table 8 Breakdown of Costs.....44

CHAPTER 1
INTRODUCTION

1.1 Definition of the project

Robotic arms are commonly seen in industrial settings, assisting with various manufacturing processes in mass production factories. Traditionally, these robotic arms are large and designed to perform specific, independent functions. However, modern advancements have led to the development of smaller, more versatile robotic arms. These mechanical arms are programmable and can assist humans by performing difficult or hazardous tasks such as welding, painting, assembly, drilling, screw driving, packaging, labeling, and gripping.

This project aims to combine the functionalities of different robotic arms into a single, multifunctional unit designed for surgical assistance. Multifunctional robotic arms are not yet widespread in the medical field, making this project an opportunity to explore and highlight the development process of such a versatile device. The focus of this project is on three primary functions: assisting in surgical procedures, providing precise tool handling, and enhancing the safety and efficiency of operations.

The robotic arm will assist surgeons by providing stable and precise control of surgical instruments. This includes holding and manipulating tools during procedures, reducing the physical strain on surgeons and improving accuracy. The arm will be equipped with advanced motors with feedback to ensure smooth and responsive movements.

The robotic arm will be capable of handling various surgical tools, including scalpels, forceps, and suturing devices. The integration of a camera and image processing system will allow the arm to accurately position and use these tools as directed by the surgeon. Voice command functionality will further enhance the ease of use, allowing surgeons to control the arm hands-free.

The primary aim of this project is to create a multifunctional robotic arm that enhances the safety and efficiency of surgical procedures. By automating certain tasks and providing precise control, the robotic arm can help reduce the risk of human error and improve patient outcomes. The arm will be designed to be user-friendly, with intuitive controls and a robust safety system to prevent accidental movements.

The main goal of this project is to develop a suitable multifunctional robotic arm that can assist in surgical procedures on a small scale. The project aims to provide a safer and more efficient tool for surgeons, reducing the physical demands of surgery and improving precision. The robotic arm will be primarily CNC-cut, making it more cost-effective than purchasing individual parts. The arm will be equipped with DC motors with feedback for all operations, ensuring reliable and consistent performance.

1.2 Significance of the project

Surgical procedures often involve tasks that are both physically demanding and hazardous. The use of robotic arms in surgery has the potential to significantly enhance safety and efficiency. This project focuses on developing a multifunctional robotic arm designed to assist surgeons by identifying and handling surgical tools during procedures, thereby speeding up the process and decreasing the risk of contamination.

Firstly, the significance of this multifunctional robotic arm lies in its ability to improve surgical outcomes by integrating advanced robotic technology into the operating room. By automating the identification and handling of tools, the robotic arm can significantly reduce the time taken for these tasks, enhancing the overall efficiency of surgical procedures. This is crucial in maintaining a sterile environment and reducing the risk of contamination, which is a major concern in surgical settings.

Secondly, the robotic arm's ability to handle tools with precision and stability reduces the physical strain on surgeons and minimizes the risk of human error. This is particularly important in complex and lengthy

procedures where fatigue can impact performance. The integration of voice command functionality allows for seamless, hands-free operation, further enhancing the efficiency and safety of the procedure.

Moreover, the robotic arm can perform tasks that are challenging or impossible for humans, such as precise tool handling at various angles and positions. This capability not only improves the accuracy of surgical procedures but also supports surgeons in performing complex tasks with greater ease and confidence.

Additionally, the use of advanced image processing and computer vision technologies enables the robotic arm to accurately identify and handle surgical tools. This reduces the need for manual handling, thereby decreasing the risk of contamination and improving the overall safety of the procedure.

In summary, the significance of this multifunctional robotic arm project lies in its potential to transform surgical practices by enhancing efficiency, safety, and precision. By automating tool identification and handling, the robotic arm supports surgeons in performing complex procedures with greater accuracy and reduces the risk of contamination, ultimately improving patient outcomes.

1.3 Project objectives

The surgical assistance robotic arm project is a programmable arm designed to enhance the efficiency and safety of surgical procedures. This multifunctional robotic arm can perform tasks such as tool identification, handling, and assisting with various surgical operations. The project focuses on the programming aspect, utilizing CNC cutting to design and produce a six-degree-of-freedom robotic arm. The arm is programmed to follow precise movements and ensure the continuity of surgical tasks, thereby supporting surgeons during operations.

The robotic arm utilizes advanced image processing and voice command technologies to identify and handle surgical tools accurately. By integrating G-coding and inverse kinematics, the arm can be programmed to perform automatic tool handling and positioning with

high precision. This approach enhances productivity, quality, flexibility, speed, robustness, and precision, while reducing potential security risks and vulnerabilities in surgical environments.

There are numerous advantages to using a robotic arm in surgical settings. For example, in sterile conditions, the robotic arm can replace human labor for more effective results, as it is highly sensitive and programmable to perform the same movement correctly each time, minimizing the risk of contamination. The robotic arm can operate continuously, which is crucial in lengthy surgical procedures where human fatigue can impact performance. Additionally, the robotic arm can perform tasks that are hazardous to human workers, such as handling sharp instruments or working in confined spaces.

The primary aim of this project is to provide a safer, more efficient, and multifunctional robotic arm for surgical assistance. By automating tool identification and handling, the robotic arm supports surgeons in performing complex procedures with greater precision and speed. The arm is designed to be cost-effective, primarily CNC-cut, with the exception of specialized surgical components. The use of DC motors with feedback to ensures reliable and consistent performance, making the robotic arm a valuable addition to the surgical team.

In summary, the surgical assistance robotic arm project aims to enhance the safety and efficiency of surgical procedures by providing a programmable, multifunctional robotic arm. The integration of advanced technologies such as image processing and voice command allows the arm to perform tasks with high precision, reducing the risk of contamination and improving overall surgical outcomes. This innovative approach supports surgeons in delivering better patient care and achieving more consistent results.

1.4 Project constraint

The surgical assistance robotic arm project faces several key constraints that must be addressed to ensure its successful implementation. One of the primary constraints is the automation of all movements. This requires a sophisticated operating program that

guarantees the desired accuracy, particularly in identifying and handling surgical tools. Precision in automation is crucial to support the efficiency and safety of surgical procedures.

Another essential constraint is the degree of freedom. The robotic arm must support six degrees of freedom, providing the necessary flexibility for the arm to move freely within the surgical field. This flexibility is vital for accurately positioning and manipulating tools during surgery.

The robotic arm must also be securely fixed to a stand to ensure stability and precision during operations. In the surgical environment, it is imperative that the arm remains steady to avoid any unintended movements that could compromise the procedure.

Additionally, the operational duty cycle of the robotic arm is a significant constraint. The components, particularly the motors and sensors, have a limited duty cycle to prevent overheating and ensure safety. This means the arm can operate continuously for a specified period before requiring a cooldown phase, which limits the duration of continuous operations but enhances safety by preventing overheating and potential malfunctions.

Maintaining a sterile environment is another critical constraint. The robotic arm and its components must be designed to withstand sterilization processes to prevent contamination during surgical procedures. This includes using materials and coatings that can endure repeated sterilization without degrading.

Integration with existing surgical systems and tools is also a constraint. The robotic arm must be compatible with current surgical instruments and operating room setups to ensure seamless operation and avoid disruptions during procedures.

Finally, the design of the user interface and control mechanisms is a constraint that must be considered. The system must be intuitive and easy for surgeons to use, incorporating voice commands and other control methods that do not interfere with the sterile field.

These constraints are critical to the design and implementation of the surgical assistance robotic arm, ensuring it operates safely, efficiently, and effectively within the surgical environment.

1.5 Report structure

In this report, there are six chapters, each providing a comprehensive overview of the various aspects of the project.

Chapter 1 - Introduction: The first chapter provides an introduction that gives a general perception of the project. It includes a detailed definition of the project, the significance of the project, detailed project objectives, and detailed project constraints. Additionally, it outlines the structure of the report.

Chapter 2 - Literature Review: The second chapter briefly explains the historical events related to robotic arms, including their use in surgery. It covers the background information, history of robotic arms, types of robotic arms, surgical robotic arms, and technological advances in robotics.

Chapter 3 - Design and Analysis: The third chapter discusses the proposed or selected design of the robotic arm, including the engineering standards followed. It provides detailed design calculations and a cost analysis. This chapter also covers the kinematics and movement control of the robotic arm, ensuring it meets the required specifications for surgical assistance.

Chapter 4 - Manufacturing: The fourth chapter discusses the manufacturing process selection and the detailed manufacturing process. It provides an overview of the hardware and assembly of components, including the robotic arm frame, servomotors, USB camera, microcontroller, and circuit diagram. This chapter highlights the use of CNC cutting in the manufacturing process to achieve the desired design.

Chapter 5 - Software: The fifth chapter focuses on the software aspects of the project. It covers the kinematics and movement control, computer

vision and image processing, and voice command integration. This chapter also discusses the testing and validation of the software, ensuring it meets the project's requirements.

Conclusions: The final chapter provides a comprehensive conclusion, including the fulfillment of project objectives, future improvements, and personal conclusions. It summarizes the achievements of the project and offers recommendations and suggestions for future projects.

Appendices: The appendices provide supplementary materials that support the main content of the report. Appendix A includes survey data, while Appendix B contains technical specifications. Appendix C lists the code used in the project, and Appendix D provides additional diagrams. Appendix E presents test results, and Appendix F offers user manuals. Appendix G includes meeting minutes, and Appendix H details the budget and financial aspects of the project.

This structure ensures a thorough and organized presentation of the project, covering all essential aspects from introduction to detailed analysis and conclusions.

CHAPTER 2
LITERATURE REVIEW

2.1 Background

The field of robotics encompasses various branches of engineering, including electronics, mechanical, digital logic, artificial intelligence, nanotechnology, and bioengineering. Robots can perform a wide range of tasks depending on their design and application, from simple to highly complex operations. A robot might be designed to accomplish a single task or a variety of tasks. For example, a robotic arm can be utilized for welding, drilling, screw-driving, and pick-and-place operations, or it can be adapted to perform each of these tasks by changing the tool attachment.

In the context of surgical assistance, the "Multi-Functional Robotic Arm" has become an essential tool in modern medical practices. Traditionally, surgical procedures have relied heavily on manual processes, which can be time-consuming and prone to human error. The versatility of a robotic arm allows it to replace several manual tasks, enhancing efficiency and precision in the operating room. For instance, a single robotic arm can assist with tool identification, handling, and positioning, significantly speeding up surgical procedures and reducing the risk of contamination.

This project focuses on developing a multifunctional robotic arm designed to assist surgeons by automating the identification and handling of surgical tools. By integrating advanced image processing and voice command technologies, the robotic arm can quickly and accurately present the required tools to the surgeon, minimizing delays and interruptions during the procedure. This innovative approach not only improves the efficiency and safety of surgeries but also supports surgeons in performing complex tasks with greater precision.

The significance of this research lies in its potential to transform surgical practices by providing a safer, more efficient, and cost-effective solution. The robotic arm's ability to operate continuously without fatigue, perform tasks with high precision, and maintain a sterile environment makes it an invaluable asset in the medical field. This project aims to demonstrate the benefits of a multifunctional robotic arm in enhancing surgical outcomes and improving patient care.

2.1.1 History of robotic arms

The development of robotic arms has a rich and fascinating history, marked by significant milestones and technological advancements. The foundation of modern surgical robotics lies in the evolution of the robotic arm, which has seen continuous improvements in dexterity, degrees of freedom, and capabilities beyond the human arm.

One of the earliest sophisticated robotic designs can be attributed to Leonardo da Vinci, who in 1495 created a robotic arm with four degrees of freedom and an analog onboard controller for power and programmability [1]. This early design laid the groundwork for future developments in robotic technology.

In the 18th century, Jacques de Vaucanson, a gifted mechanical designer, created complex automata, including a flute player with intricate finger mechanisms [1]. This period saw the development of mechanical devices that mimicked human movements, setting the stage for more advanced robotic systems.

The modern era of robotic arms began in the 20th century with the introduction of the first industrial robotic arm, Unimate, in 1961. Developed by George Devol and marketed by Joseph Engelberger, Unimate was installed at a General Motors plant for automated diecasting [1]. This marked the transition of robotic arms from laboratory curiosities to practical industrial tools.

Subsequent developments included the Rancho arm in 1963, designed for assisting individuals with disabilities, and Marvin Minsky's Tentacle arm in 1968, which was an electro-hydraulic high-dexterity arm for underwater exploration [1]. The Stanford arm, created by Victor Scheinman in 1969, was another significant advancement, featuring six degrees of freedom and the ability to perform complex tasks [1].

In the medical field, the da Vinci Surgical System, introduced in the early 2000s, represents a major milestone. This system utilizes robotic arms with seven degrees of freedom, allowing for precise and minimally invasive surgical procedures [1]. Figure 1 illustrates the evolution of the da Vinci Surgical System from 1990 to 2014, showcasing its technological advancements and improvements in surgical precision. The da Vinci system has become a hallmark of modern surgical robotics, demonstrating the potential of robotic arms to enhance surgical outcomes.

Figure 1. Evolution of the Da Vinci Surgical System (1990-2014) [34]

The evolution of robotic arms continues to be driven by advancements in electronics, computer systems, and materials science. The integration of artificial intelligence and machine learning is expected to further enhance the capabilities of robotic arms, making them even more versatile and efficient in various applications, including surgery.

2.1.2 Uses of robotic arms

Robotic arms have a wide range of applications in various aspects of daily life and specialized environments. In households, robotic arms can significantly ease the burden of everyday chores. They can assist with cleaning tasks such as vacuuming, dusting, and tidying up, ensuring that homes remain spotless with minimal human effort [2]. In the kitchen, robotic arms can prepare ingredients, cook meals, and even serve food, making meal preparation more efficient and enjoyable [3]. Additionally, they can handle laundry tasks, including separating, washing, drying, ironing, and folding clothes [4]. Personal care tasks, such as showering, shampooing, and dressing, can also be managed by robotic arms, providing valuable assistance to individuals with mobility challenges [5][6][7].

In the workplace, robotic arms are invaluable for a variety of tasks. In office settings, they can help organize documents, fetch items, and even provide massages to relieve stress [8]. In manufacturing, robotic arms are essential for tasks such as welding, painting, and assembly, enhancing productivity and precision [9]. They are also used in logistics for sorting, loading, and unloading goods in warehouses, streamlining operations and reducing manual labor [10].

Robotic arms are also useful for tasks performed on desks or working surfaces. They can manipulate small objects, perform precise tasks like soldering, and assist with measurements and assembly [11]. Keeping workspaces organized is another key function, as robotic arms can put objects in predefined places and optimize the working area for space and ergonomics [11].

In hospitals, robotic arms play a crucial role in enhancing patient care and medical procedures. In emergency rooms, they can perform CPR and assist in adjusting patient posture for X-rays and splint treatments [12]. During surgeries, robotic arms can pass tools to surgeons and even perform surgical procedures with high precision [13]. In laboratories, they handle repetitive testing and analyses, especially those involving hazardous materials, ensuring safety and accuracy [14][15].

For individuals with disabilities, robotic arms can serve as prosthetic devices, replacing lost limbs and helping them perform daily activities [16]. They can also act as additional arms, providing extra support for tasks that require more than two hands or involve handling objects that are too hot or cold [16].

In disaster and war scenarios, robotic arms are invaluable for rescue operations and military applications. They can be used to rescue victims from collapsed buildings and perform tasks in hazardous areas, such as damaged nuclear power plants or chemical factories [2]. In military settings, robotic arms can assist in defusing bombs and removing traps, enhancing safety for soldiers [2].

These diverse applications highlight the versatility and potential of robotic arms in improving efficiency, safety, and quality of life across various domains.

2.1.3 Surgical robotic arms

Robotic surgery is an innovative and rapidly evolving field that has significantly impacted modern surgical practices. The development of surgical robots aims to overcome the limitations of traditional laparoscopic techniques and enhance the capabilities of surgeons. This technology is still in its infancy, but it holds great promise for the future of minimally invasive surgery [17][18].

The concept of robotic surgery dates back to the 1980s, with the development of the Puma 560 robot, which was used for neurosurgical biopsies and transurethral resection of the prostate [19]. Figure 2 illustrates the Puma 560 robot's kinematic parameters and frame assignments, highlighting its structural design and functionality. The introduction of the Da Vinci and Zeus systems marked significant advancements in robotic surgery. These systems are comprehensive master-slave robots with multiple arms operated remotely from a console, providing enhanced dexterity, improved visualization, and better ergonomics for surgeons [17][20].

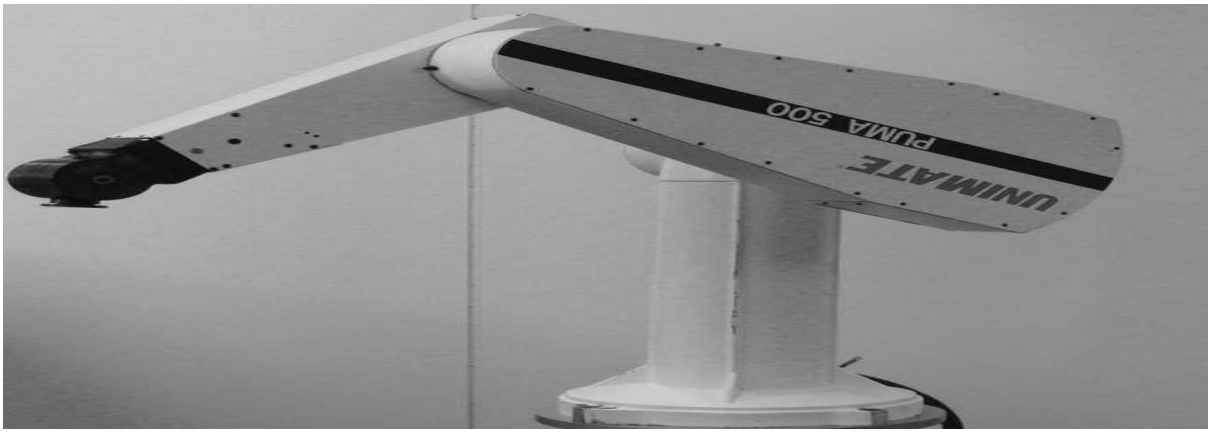

Figure 2. PUMA 560 Robot (1985) [35]

Today, several robotic systems are FDA-approved and used in various surgical procedures. The Da Vinci system, developed by Intuitive Surgical, and the Zeus system, developed by Computer Motion, are the most prominent. These systems offer 3D visualization, increased degrees of freedom, and the ability to scale movements, making complex surgeries more feasible [17][20].

Robotic-assisted surgery offers numerous advantages over conventional laparoscopic surgery. These include enhanced dexterity, as robotic systems provide increased degrees of freedom, allowing for more precise manipulation of instruments [17]. Improved visualization is another benefit, with 3D visualization and depth perception offered by robotic systems being superior to traditional laparoscopic cameras [17]. Additionally, surgeons can operate from a comfortable, ergonomically designed console, reducing fatigue and improving performance [17]. Robotic systems also provide stability and precision, filtering out tremors and scaling movements to enable micromotions and precise tissue manipulation [20]. Table 1 summarizes the key advantages and disadvantages of robot-assisted surgery compared to conventional surgical techniques, highlighting the benefits of precision and stability while addressing concerns such as accessibility.

Despite the advantages, robotic surgery also has several disadvantages. The initial cost of robotic systems is high, and ongoing maintenance and upgrades can be expensive [18]. Robotic systems also have large footprints and require significant space in operating rooms [18]. Additionally, the availability of compatible instruments is still limited, necessitating reliance on traditional tools for some procedures [20]. Many procedures will also have to be redesigned to optimize the use of robotic arms and increase efficiency [18].

Robotic surgery is currently used in various fields, including cardiothoracic surgery, urology, gynecology, and general surgery. Procedures such as coronary artery bypass grafting, mitral valve repair, radical prostatectomy, nephrectomy, hysterectomies, tubal reanastomosis, cholecystectomy, gastric bypass, and bowel resection are performed using robotic systems [17][21][22].

Table 1. Advantages and Disadvantages of Robot-Assisted Surgery Versus Conventional Surgery [17]

Human strengths	Human limitations	Robot strengths	Robot limitations
• Strong hand–eye coordination	• Limited dexterity outside natural scale	• Good geometric accuracy	• No judgement
• Dexterous	• Prone to tremor and fatigue	• Stable and untiring	• Unable to use qualitative information
• Flexible and adaptable	• Limited geometric accuracy	• Scale motion	• Absence of haptic sensation
• Can integrate extensive and diverse information	• Limited ability to use quantitative information	• Can use diverse sensors in control	• Expensive
• Rudimentary haptic abilities	• Limited sterility	• May be sterilized	• Technology in flux
• Able to use qualitative information	• Susceptible to radiation and infection	• Resistant to radiation and infection	• More studies needed
• Good judgment			
• Easy to instruct and debrief			

2.2 Technological advances in surgical robotics

The field of surgical robotics has seen significant advancements, particularly in the realm of minimally invasive surgery (MIS). One notable development is the fourth-generation Da Vinci SP system, which has optimized the transoral vestibular approach. This system allows for robotic thyroidectomy to be performed with greater safety and effectiveness, minimizing scarring and improving patient outcomes. The trend is moving towards single-port approaches, which are expected to enhance the efficiency and safety of MIS procedures [24].

The evolution of surgical instrument arms is a critical aspect of advancing surgical robotics. Future designs are focusing on miniaturization, precision, and flexibility. Innovations in biomaterials and bionic materials are anticipated to improve the compatibility and functionality of these arms. For instance, new materials such as tactile sensing materials and biodegradable materials are expected to revolutionize the next generation of surgical robots. These advancements will allow for more precise and less invasive surgical procedures [25][26][27].

Modular and reconfigurable systems are being developed to address the limitations of intraoperative space. These systems integrate driving, control, sensing, and battery units into compact modules that can be reconfigured according to surgical needs. This modular approach enhances the versatility and functionality of surgical robots, allowing them to perform a wider range of procedures with greater efficiency and precision [28][29][30].

The advent of 5G technology has made telemedicine a viable option for surgical robots. Remote operations increase the accessibility and mobility of surgeons, allowing surgeries to be performed without geographical limitations. Future research will focus on intelligent control algorithms and real-time image transmission to improve the efficiency and safety of remote surgeries. This development is expected to significantly enhance the global reach of surgical expertise [31].

Intelligent control technology is at the core of future surgical robots. Advances in artificial intelligence, big data, cloud computing, and deep learning are expected to enhance the cognitive control and decision support systems of surgical robots. These technologies will enable more precise and stable control during minimally invasive surgeries, improving patient outcomes and reducing the risk of complications [32].

Human-machine collaboration is a key development direction for surgical robots. Future systems will integrate human cognitive abilities with robotic precision to adapt to the dynamic surgical environment. Efficient man-machine collaboration will optimize the interaction between surgeons and surgical robots, improving the effectiveness and safety of surgeries. This collaboration is expected to lead to more intuitive and responsive surgical systems [33].

In conclusion, the continuous progress in robotics, biomechanics, and remote interaction technologies is driving the development of more intelligent, precise, and efficient surgical robots. These advancements are expected to revolutionize minimally invasive surgery, making it more specialized, remote, and integrated. The future of surgical robotics holds great promise for improving patient care and expanding the capabilities of surgeons worldwide.

CHAPTER 3
DESIGN AND ANALYSIS

3.1 Proposed Design

At the design stage of our surgical assistance robotic arm project, we will prioritize four key criteria: safety, manufacturability, cost, and reliability. Safety is our foremost concern, as the robotic arm will assist the surgeon during procedures, where any malfunction could pose significant risks to patients and medical staff. Manufacturability is the next priority, focusing on ease of production to ensure the arm can be manufactured efficiently without requiring high-tech machinery. This approach will also facilitate maintenance and sustainability, positively impacting the overall cost. Cost effectiveness is crucial, as it ensures the robotic arm remains affordable without compromising on quality. Reliability is essential, as the arm must perform with high accuracy and precision during surgeries, ensuring consistent and dependable operation. Overall, our design aims to be safe, robust, reliable, and cost-effective, ensuring it does not fail during critical surgical procedures.

The selection of the material for the robotic arm structure is based on a decision matrix, which is presented below in Table 2. This matrix evaluates various materials using key criteria, including sustainability (assessed using reliable references), cost (determined through market inquiries), manufacturability (evaluated based on ease of fabrication and design feasibility), reliability (tested through trial and error, where iron proved durable while PLA filament brackets broke easily), and availability (confirmed through market research). The total points from these evaluations helped finalize the most suitable material choice for the robotic arm, ensuring an optimal balance between practicality and performance.

Table 2. Decision Matrix For Material Selection

Criteria	Sustainability /10	Cost /10	Manufacturability /10	Reliability /10	Availability /10	Total /50
Aluminum	8 [37]	6	9	8	7	38
Iron	10 [37]	8	10	10	10	48
PLA Filament	10 [36]	10	10	6	10	46

The table above shows that iron is the most suitable choice for our proposed design. It demonstrated high reliability, proving to be durable through trial and error, unlike PLA filament brackets, which broke easily. While it may not be the most cost-effective or sustainable material, its strength and stability make it an ideal option for ensuring the longevity and functionality of the robotic arm. Additionally, iron is widely available, making it a practical choice for manufacturing.

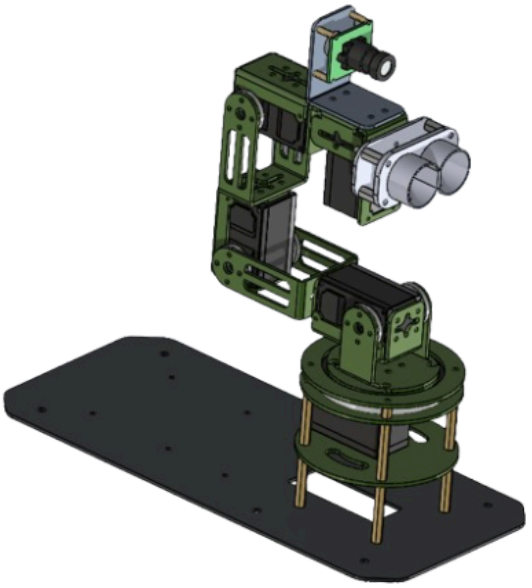

Figure 3. Chosen Open Source Robotic Arm Design

The design is based on an open source robotic arm, shown in Figure 3 of a six-degree-of-freedom (6-DOF) robotic arm designed for surgical assistance.The arm is constructed using metal iron through CNC cutting, which provides a lightweight yet structurally stable framework. Its modular design consists of multiple articulated joints, enabling precise movements required for delicate surgical procedures. The arm is mounted on a circular base supported by four vertical legs, ensuring stability during operation.

At the top of the robotic arm, a camera module is integrated, allowing for real-time visual feedback and image processing. This feature enhances the arm's ability to assist surgeons by providing a detailed view of the surgical site, aiding in precision-based interventions.

The robotic arm includes strategically placed actuators and motors at each joint, ensuring smooth and controlled motion for complex maneuvers. The end-effector features a double magnet mechanism, using two electromagnets to control object interaction. It attracts and releases metallic objects through an adjustable magnetic field, enabling precise automation without mechanical wear. This design ensures secure handling while optimizing efficiency in specialized applications..

The U-shaped structural bracket in figure 4 is a critical component of the six-degree-of-freedom robotic arm, designed to provide support and facilitate joint articulation. This bracket serves as a mounting structure for motors and rotational joints, allowing smooth movement between connected segments. Its design includes multiple slots and circular holes, which enhance flexibility in positioning and alignment during assembly. The elongated slots on the sides enable fine adjustments, while the circular holes at the top are intended for motor shafts or pivot points, ensuring controlled rotation.

Figure 4. U-Shaped Servo Bracket

The structural mounting bracket, shown in Figure 5, is a critical component designed to securely house motors and support joint articulation. It features a box-like structure with side plates that include multiple mounting holes and cross-shaped cutouts, specifically designed for servo motor integration. These cutouts ensure precise alignment and secure attachment of actuators, which are essential for accurate motion control.

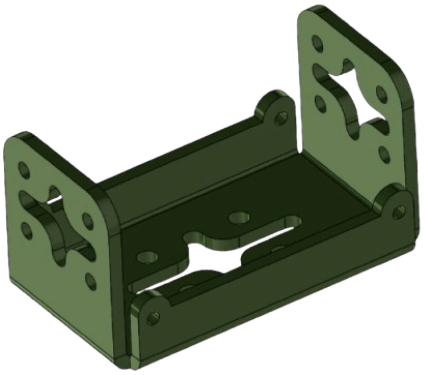

Figure 5. Motor Housing Bracket

The robotic arm base part, shown in Figure 6, is a fundamental component designed to provide stability and support for the entire robotic system. It features a reinforced structure with strategically placed mounting holes, allowing for secure attachment of motors and actuators. The base is engineered to minimize vibrations, ensuring precise motion control and smooth articulation of joints. Additionally, its rigid frame enhances durability, preventing structural deformation during operation. The design incorporates cross-sectional reinforcements, which help distribute mechanical loads evenly, optimizing performance and reliability.

Figure 6. Arm Base Support

The final motor mounting bracket, shown in Figure 7, is a crucial component designed to securely hold the last motor responsible for controlling the end effector. It also serves as the attachment point for the camera, positioned below, enabling real-time monitoring and precision adjustments. Engineered for rigidity and stability, this bracket ensures secure motor placement, minimizes vibrations, and enhances overall performance, making it an integral part of the robotic arm's functionality.

Figure 7. End Effector Motor Bracket

The magnet holder, shown in Figure 8, is a crucial component of the end effector, designed to securely hold two magnets for precise functionality. The end effector consists of two separate parts, which are spaced apart using four spacers to ensure proper alignment and stability. The first part of the end effector is engineered to attach directly to the end effector motor, enabling controlled movement and responsiveness. The second part is specifically designed to hold the two magnets, ensuring a firm and reliable grip for various applications. This structured design enhances the efficiency and precision of the robotic system, making it an essential element in the overall mechanism.

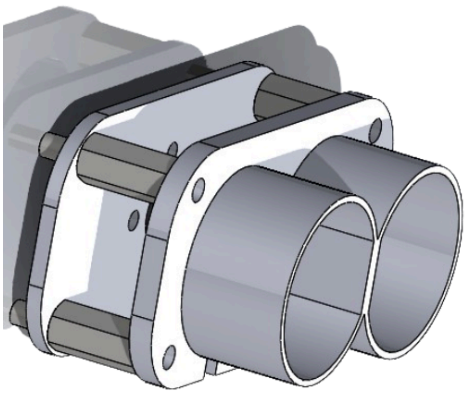

Figure 8. End Effector Magnet Holder

The structure shown in Figure 9 represents the base of the robotic arm, designed to provide additional weight and stability to prevent tilting during operation. The base consists of two acrylic plates supported by four vertical spacers, creating an enclosed compartment for housing electronic components. The choice of acrylic as the primary material ensures a balance between durability and aesthetics while maintaining a lightweight yet sturdy construction.

Inside the base, a Raspberry Pi and the power supply will be securely mounted, ensuring efficient management of the robotic arm's control system and power distribution. The base plate features a central hole, intended for cable routing or securing additional components. The carefully positioned mounting holes on the plates allow for a firm connection with the rest of the robotic arm, ensuring structural integrity during movement. By integrating the electronic components within the base, the design optimizes space utilization and protects sensitive hardware from external interference. This base plays a crucial role in maintaining the overall balance and operational efficiency of the robotic arm.

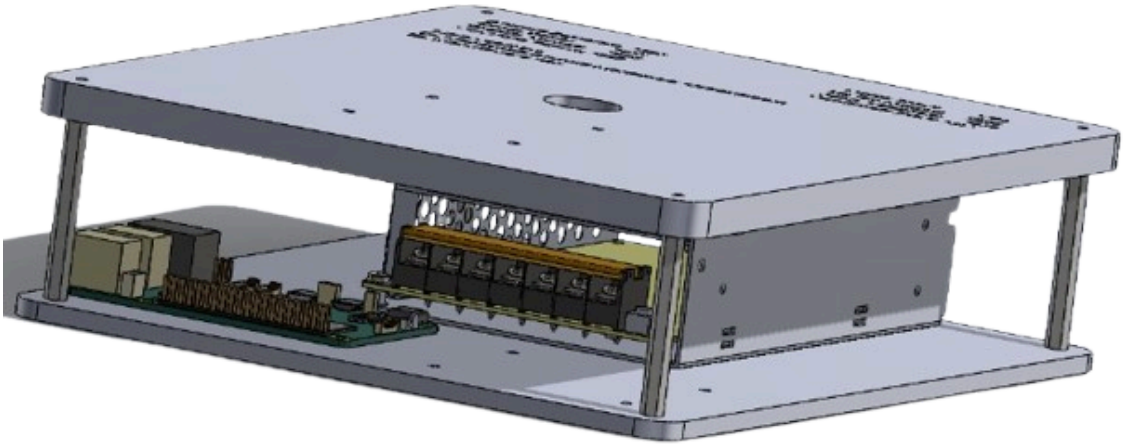

Figure 9. Electronics Enclosure Base

3.2 Engineering Standards

Engineering standards are the backbone of design, ensuring consistency, reliability, and safety across various applications. As defined by the International Organization for Standardization (ISO), "Standards are documents which provide requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes and services are fit for their purpose" [38]. These standards help establish uniform benchmarks for quality and performance, minimizing risks and enhancing efficiency. Table 3 highlights key certifications such as CE, FCC, RoHS, and ISO 9001, which ensure compliance with international regulations. By adhering to these certifications, engineers guarantee that products meet essential criteria for durability, integration, and functionality, contributing to a streamlined and well-regulated development process.

Table 3. Engineering Standards

Component Selection	Following Standard
Raspberry Pi Circuit	CE, FCC
LX-15D Serial Bus Servo Dual Bearing Full Metal Gear Steering Servo	CE, RoHS certificate
Hiwonder Bus Servo Controller Serial Bus Servo Tester Board for Bus Servo	CE, RoHS certificate
USB Camera Module 640x480 30fps 130 Degree Wide Angle	ISO 9001, ISO/IEC 15444 (JPEG 2000)

Power Supply 12V 5A	
XL4016 DC-DC Buck Converter 8A Step Down 4-36V to 1.25-36V with Display XH-M404	IEC 60950-1, IEC 62368-1

3.3 Design Calculations

To ensure the robotic arm functions with precision and stability, detailed design calculations are essential. These calculations provide a comprehensive understanding of the arm's motion, position, and orientation in 3D space, allowing for accurate control and performance optimization. A systematic approach is followed, starting with the Denavit-Hartenberg (DH) parameterization, then deriving the forward kinematics, and finally calculating the working area and volume to validate the design.

The kinematic analysis begins with the DH parameterization, which is used to define the robotic arm's joints and links systematically. This standard method allows for the transformation of each link relative to the previous one, forming the basis for movement calculations. Once the DH parameters are established, forward kinematics is applied to determine the precise position and orientation of the end-effector. The transformation matrices derived from this analysis result in the homogeneous transformation matrix, which represents the complete spatial configuration of the arm relative to its base. These calculations ensure that the robotic arm can achieve its intended range of motion and perform tasks accurately.

In addition to kinematic modeling, it is necessary to analyze the working area to determine the maximum reach and range of motion of the robotic arm. To visualize this, a 2D representation of the working area is created in AutoCAD. This drawing helps in understanding the spatial constraints and ensuring that the arm's movement is optimized for efficiency. Once the working area is established, the next step is to calculate the working volume, which represents the total space the end-effector can reach. This is derived based on the lengths of the links, joint limits, and rotational constraints, ensuring the robotic arm can perform within its designed operational environment.

By integrating kinematic modeling, workspace analysis, and volume calculations, the robotic arm's design is thoroughly validated. These calculations not only ensure precise motion control but also optimize the structure to prevent mechanical limitations and improve overall performance.

3.3.1 DH Table

To systematically define the robotic arm's kinematics, the Denavit-Hartenberg (DH) convention was applied. This method provides a structured approach for representing the arm's joints and link transformations using four key parameters: link length (a), link twist (α), link offset (d), and joint angle (θ). The joint angle (θ) was calculated by measuring the angle between the X-axis of the previous frame and the X-axis of the current frame in the direction of the Z-axis of the previous frame. The results of each calculated angle are presented in Table 5. By assigning these parameters to each joint, the arm's motion and spatial positioning could be mathematically modeled.

For our robotic arm, the DH table Table 4 was constructed by first identifying the number of joints and measuring the corresponding link lengths (a). These values were based on the physical design of the arm, ensuring that each segment was accurately represented. The twist angles (α) were determined based on the orientation of each consecutive link relative to the previous one, capturing any necessary rotational displacement. The joint offsets (d) were assigned based on the

linear displacement along the z-axis, while the joint angles (θ) were kept as variables that would change based on movement. A graphical representation of the DH frame assignments was created to visually depict the coordinate frames at each joint, as shown in Figure 10.

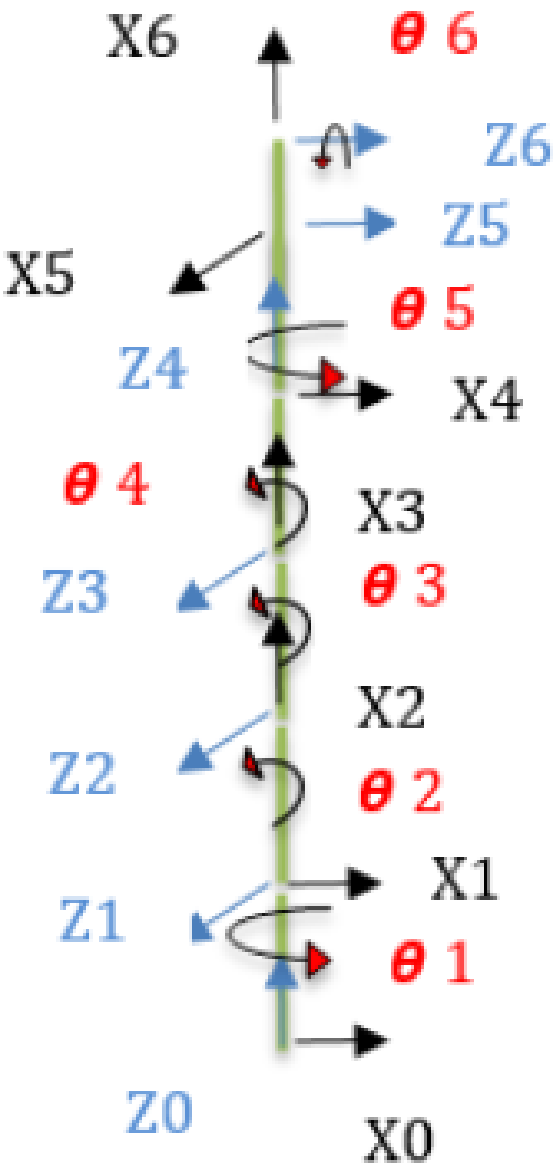

Figure 10. Kinematic Diagram

Table 4. DH Table

θ_i	α	a	d	A_i
θ_1	90	0	L_1	A_1
θ_2	0	L_2	0	A_2
θ_3	0	L_3	0	A_3
θ_4	-90	0	0	A_4
θ_5	-90	0	$L_4 + L_5$	A_5
θ_6	0	L_6	0	A_6

Table 5. Joint Angles

θ_i	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Angles	0	90	0	-90	-90	-90

3.3.2 Kinematics

Once the DH parameters were defined, the homogeneous transformation matrices were derived for each joint using the following standard transformation equations.

$$H_n^{n-1} = \begin{bmatrix} C(\boldsymbol{\theta}_n) & -S(\boldsymbol{\theta}_n)C(\boldsymbol{\alpha}_n) & S(\boldsymbol{\theta}_n)S(\boldsymbol{\alpha}_n) & r_n C(\boldsymbol{\alpha}_n) \\ S(\boldsymbol{\theta}_n) & C(\boldsymbol{\theta}_n)C(\boldsymbol{\alpha}_n) & -C(\boldsymbol{\theta}_n)S(\boldsymbol{\alpha}_n) & r_n S(\boldsymbol{\alpha}_n \\ 0 & S(\boldsymbol{\alpha}_n) & C(\boldsymbol{\alpha}_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By using Peter Corke's MATLAB Robotics Toolbox, these matrices were efficiently calculated and validated.

$$A1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 3.9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A2 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 8.05 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A3 = \begin{bmatrix} 1 & 0 & 0 & 8.05 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A6 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A1 * A2 * A3 * A4 * A5 * A6 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These matrices allowed for the computation of the end-effector's position and orientation relative to the base frame.

By utilizing this structured approach, the forward kinematics of the arm was effectively determined, ensuring precise control and operation within its intended workspace.

3.3.3 Working Volume

To determine the working volume of the robotic arm, a systematic approach was followed. First, the working area was calculated by fixing two links while allowing the remaining link to move within its defined range. This process was repeated for all links to ensure comprehensive

coverage of the arm's reach. Each combination results in a unique position within the workspace, denoted as C1 to C8. For simplicity, each link's motion was constrained to a 90-degree range, making the calculations more manageable. The different configurations of joint angles used in the calculations are presented in Table 6, which outlines the combinations of joint angles θ_1, θ_2 and θ_3 and their corresponding workspace points.

Table 6. Joint Angle Configurations for Working Area Calculation

θ_1	θ_2	θ_3	Points
0°	0°	0°	C1
		90°	C2
	90°	0°	C3
		90°	C4
90°	0°	0°	C5
		90°	C6
	90°	0°	C7
		90°	C8

The process of fixing and moving specific joints is illustrated in Table 7, which details how each joint was constrained while others were varied to determine the full extent of the arm's reachable positions. After identifying all possible points the arm could reach, these points were matched systematically.

For example, b1 [C1----C2] indicates that an arc should be drawn connecting C1 to C2, with b1 as the center. Similarly, other point pairs were linked following the same approach, ensuring that the entire reachable area of the arm was accurately mapped.

Table 7. Fixed and Moving Links for Reachable Area Determination

Fixed	Moving
Link 1 & 2 are Fixed $\theta_1 = 0, \theta_2 = 0$ $\theta_1 = 0, \theta_2 = 90$ $\theta_1 = 90, \theta_2 = 0$ $\theta_1 = 90, \theta_2 = 90$	In this case link 3 is moving b1 [C1 ---- C2] b2 [C3 ---- C4] b3 [C5 ---- C6] b4 [C7 ---- C8]
Link 1 & 3 are Fixed $\theta_1 = 0, \theta_3 = 0$ $\theta_1 = 0, \theta_3 = 90$ $\theta_1 = 90, \theta_3 = 0$ $\theta_1 = 90, \theta_3 = 90$	In this case link 2 is moving a1 [C1 ---- C3] a1 [C2 ---- C4] a2 [C5 ---- C7] a2 [C6 ---- C8]
Link 2 & 3 are Fixed $\theta_2 = 0, \theta_3 = 0$ $\theta_2 = 0, \theta_3 = 90$ $\theta_2 = 90, \theta_3 = 0$ $\theta_2 = 90, \theta_3 = 90$	In this case link 1 is moving 0 [C1 ---- C5] 0 [C2 ---- C6] 0 [C3 ---- C7] 0 [C4 ---- C8]

This area was then drawn using AutoCAD for visualization and verification, as shown in Figure 11, where the link lengths of our robotic arm are $L1 = 8$ cm, $L2 = 8$ cm, and $L3 = 10$ cm.

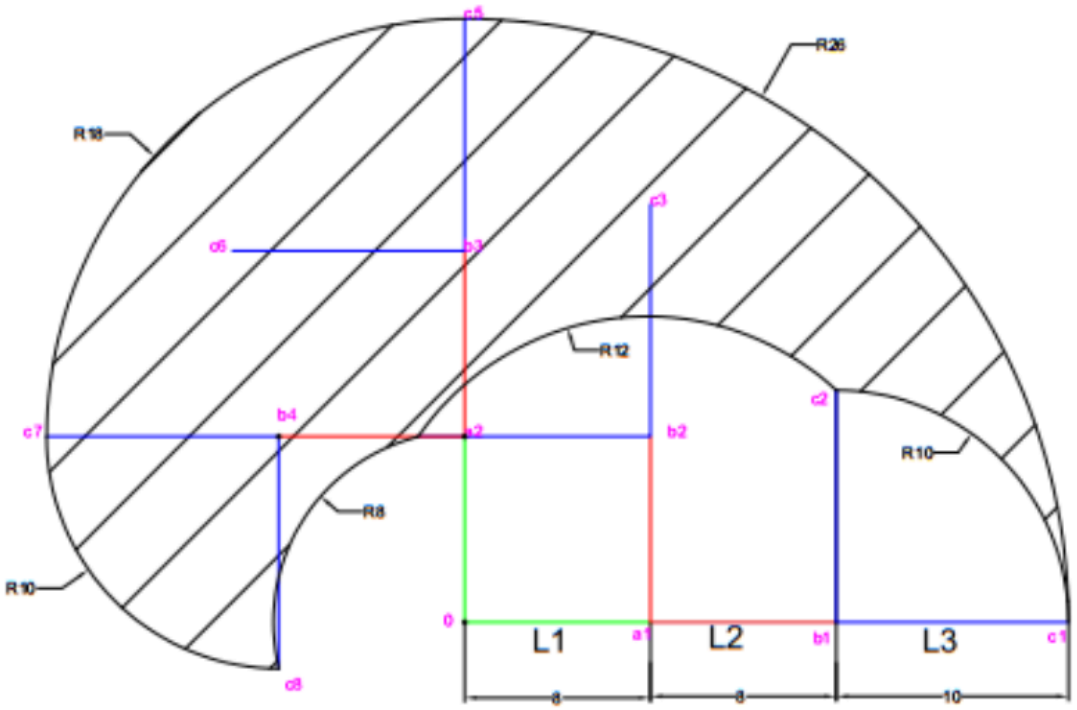

Figure 11. AutoCAD Drawing of Working Area

To ensure the accuracy of the calculated working area, a Python code was implemented to validate the results, as illustrated in Figure 12.

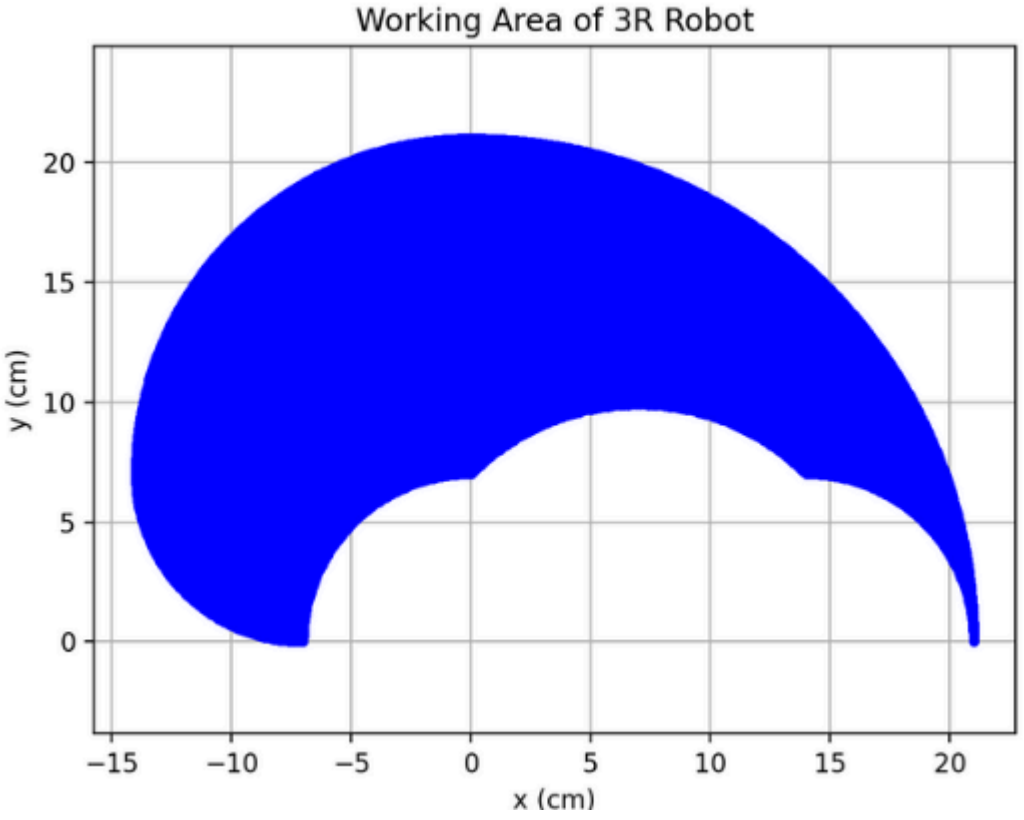

Figure 12. Python Validation of Working Area

Finally, the working volume was determined by multiplying the obtained area by the height of the arm, providing a clear representation of the arm's operational space, as shown in Figure 13.

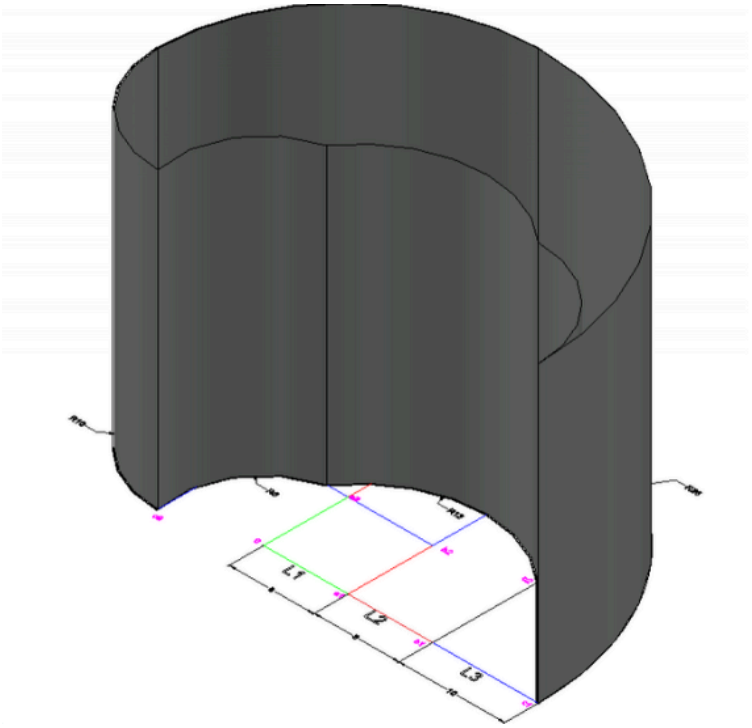

Figure 13. AutoCAD Drawing of Working Volume

3.3.4 Force Analysis

For the force analysis of the robotic arm, we used the concept of moments to understand the torque acting on each link. To simplify the calculations, we assumed that the arm is fully stretched in a straight-line configuration. This assumption allows us to analyze the worst-case scenario, where the torque on each joint is at its maximum. Additionally, we assumed that the weight load acting on the arm is 500 g, ensuring that our calculations reflect a realistic operating condition.

By considering the weight of the motors and the forces exerted by the links, we derived a relationship between the weight of each motor and the torque required at each joint. The force distribution is visually represented in Figure 14, which illustrates the forces acting at each link of the robotic arm. The torque at each joint was calculated using the moment equation:

$$\tau = r \times F$$

Where: τ is the torque, r is the distance from the joint to the applied force (i.e., the length of the link), F is the force due to the weight of the motor and the link.

The weight of each link was considered to be 13.6 g, with varying lengths for each segment of the robotic arm. Link 1 measured 81 mm, while Link 2 had a length of 84.14 mm. The longest segment, Link 3, extended to 98.88 mm. To ensure consistency in calculations and obtain results in Newton-meters (Nm), all units were converted to meters (m) and kilograms (kg) before performing the force analysis.

By applying the moment equation at each joint, the torque values for each link were derived as follows:

$$T_1 = (w_m + 0.5) \times 0.09388 + (0.0136 \times 0.04694)$$

$$T_1 = (0.09388 \times w_m) + 0.0475 \quad (3.1)$$

$$T_2 = T_1 + (w_m \times 0.08414) + (0.0136 \times 0.0420) \quad (3.2)$$

Sub (3.1) in (3.2)

$$T_2 = (0.17802 \times w_m) + 0.04807 \quad (3.3)$$

$$T_3 = T_2 + (w_m \times 0.081) + (0.0136 \times 0.0405) \quad (3.4)$$

Sub (3.3) in (3.4)

$$T_3 = (0.25902 \times w_m) + 0.0486208$$

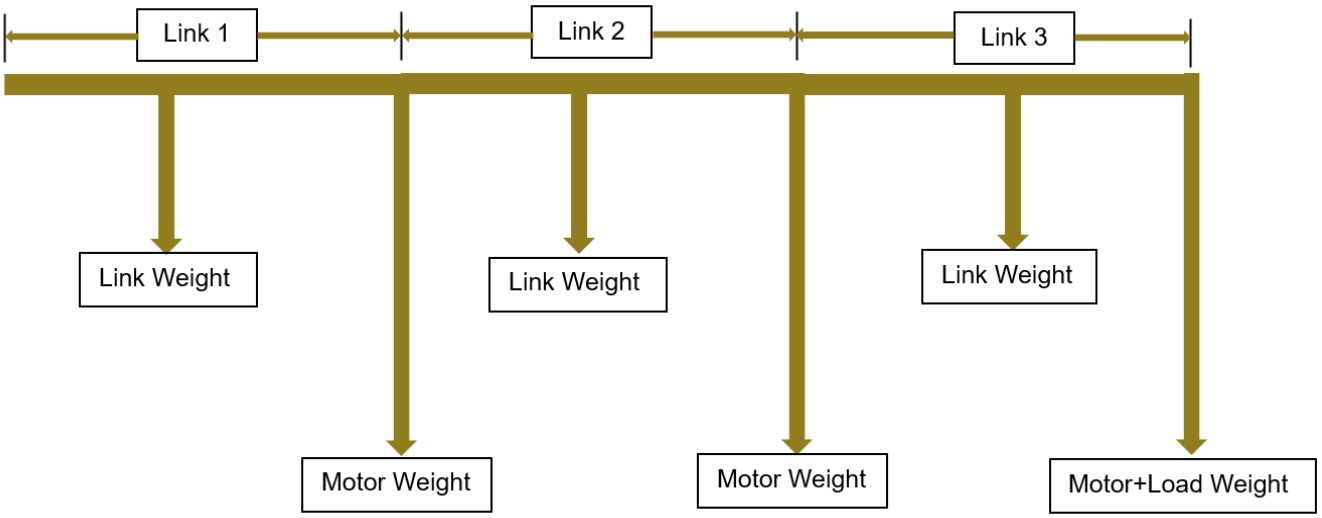

Figure 14. Force Distribution

3.4 Cost Analysis

We embarked on the development of a robotic arm for surgical assistance, starting entirely from scratch. Our approach included CNC cutting the arm and sourcing components both locally and internationally. Due to the unavailability of certain components in Egypt, we procured them from the USA, while the rest were sourced locally.

The CNC-cut arm was produced in-house, involving a significant investment. We acquired microcontrollers and motors from the USA, which were essential for the functionality of the arm. The power supply was sourced from Egypt, contributing to the overall mechanical performance. Additionally, the wiring/connectors were also procured locally, ensuring seamless integration. We utilized open-source software development tools, which helped in programming and controlling the arm without additional expenses. Miscellaneous expenses, including various smaller components and materials, were also accounted for.

In total, the comprehensive cost for developing the robotic arm reflects our strategic sourcing of components to ensure both quality and cost-effectiveness in the development of our robotic arm. For a detailed breakdown of the costs, please refer to Table 8.

Table 8. Breakdown of Costs

Item	Description	Supplier	Quantity	Per Unit (EGP)	Cost (EGP)
Raspberry pi 5	Powerful single-board computer	Amazon	1	3627.36	3627.36
USB Camera	HD camera module, with 640x480 VGA resolution at 30fps, a 130-degree wide-angle lens	Amazon	1	1010.89	1010.89
Serial Bus Servo	Offers real-time feedback, full metal gears and dual ball bearings	Amazon	7	859.18	6014.26
Serial Bus Servo Controller	Uses an STM32 microcontroller, supports online debugging	Amazon	1	2174.50	2174.50
Servo Motor	The position, speed, and movement of gripper.	Makers	1	190	190
Power Supply	Provides 12V at 10A	Makers	1	340	340
XL4016 step down transformer	Reduces high voltage to a lower voltage	Makers	1	225	225

USB-C Power Supply	Reliable power for the raspberry pi, ensuring stable performance for the project.	Devboards Market	1	1150	1150
Raspberry Pi 5 Case	Protection and preventing overheating.	Devboards Market	1	1000	1000
3D Printing	Involves creating parts layer by layer using a 3D printer	Printing Center	4	225	900
CNC Cutting	Involves creating parts by cutting metal using CNC cutting.	Alahmade ya co	1	1500	1500
Gripper	Device that grasps and holds objects	Makers	1	340	340
SD Card	Small, portable memory card used to store data	Computers Market	1	450	450
Micro HDMI	Connects the raspberry pi 5 to a monitor or TV	Makers	1	80	80
Screws	Hold brackets together	Workshop	1	200	200
ZAFFIRO USB Microphone	Microphone to give orders to arm	Makers	1	250	250
Electric Magnet	Device that holds objects	Amazon	2	350	700
USB Extender	Extends the camera Cable	2B	1	125	125
others	Shipping and Taxes	…	…	…	1929
Total Cost					22881.1

CHAPTER 4
MANUFACTURING

4.1 Manufacturing Processes

Instead of buying expensive brackets and incurring high shipping costs, we utilized local machining companies that offer fair machining prices. Making a steel part using CNC machining is a fair price due to the scarcity of the material blocks and their high price in the market locally. Initially, we decided to have the design 3D printed using PLA filament, as it was the most cost-efficient prototyping technique available. However, the material proved to be too brittle for our needs. Consequently, we had to use CNC machining to cut metal iron. Almost all parts are now CNC-cut and assembled.

4.2 Overview of Hardware/Assembly of Components

The hardware and assembly of components for our project include Hiwonder servomotors for precise control, a Hiwonder USB camera for capturing images and videos, and a Raspberry Pi microcontroller to manage the operations of the robotic arm, process inputs from sensors, and execute programmed tasks. To ensure stable and efficient power delivery, we used a 12V 10A switching power supply along with an XL4016 DC-DC buck converter to step down the voltage as needed for various components. Additionally, a Hiwonder Serial Bus Servo Controller was integrated to manage motor control, enabling the execution of complex motion sequences with high precision and reliability.

4.2.1 Power Supply

For our robotic arm, we used a 12V 10A switching power supply, known for its high efficiency and reliability across various applications, as shown in Figure 15. It features strong anti-interference performance, a low operating temperature, and a long service life. With a wide input voltage range of 100–220V AC at 50/60Hz, it complies with global usage standards. The unit offers excellent insulation and high electrical strength, along with built-in protection functions such as overload,

overvoltage, and short circuit protection, all of which automatically recover after activation. Housed in a durable metal casing, the power supply delivers a stable 12V DC output at up to 10A (120W), making it ideal for powering the robotic arm and its components. Its compact dimensions of 200 x 98 x 42 mm allow for easy integration into our system design [41].

Figure 15. Power Supply Unit

4.2.2 Step Down transformer

We integrated the XL4016 DC-DC Buck Converter, which provides a highly efficient and adjustable step-down voltage solution, as shown in Figure 16. This module supports an input voltage range of 4V to 36V DC and delivers an adjustable output voltage from 1.25V to 36V, with a continuous output current of up to 5A and a peak of 8A, making it suitable for powering various components of the robotic system. It features a 3-digit digital display for real-time output voltage monitoring and an onboard multi-turn potentiometer for precise voltage adjustment. The converter includes built-in protections such as thermal shutdown, current limiting, and short circuit protection, all with automatic recovery. Its compact design (64mm x 47mm x 26mm) and high efficiency—up to

95%— make it ideal for regulating power from sources like 12V or 24V batteries down to levels required by microcontrollers and sensors. This module ensures stable and reliable power delivery throughout the robotic arm's operation [42].

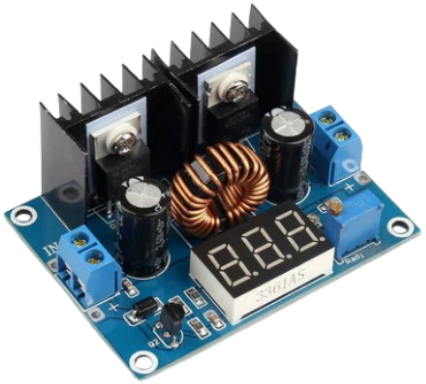

Figure 16. Step-Down Converter (XL4016)

4.2.3 Motor Control Unit

We used the Hiwonder Serial Bus Servo Controller Communication Tester, as shown in Figure 17, to manage and control the servomotors. This controller features a 16M high-capacity memory capable of storing up to 230 action groups, with each group supporting up to 510 individual actions—ideal for complex motion sequences. It is powered by a high-performance STM32 microcontroller with an ARM Cortex-M3 core, ensuring fast and reliable processing. The controller supports online debugging without requiring additional computer drivers and offers both PC graphical debugging and TTL serial port communication. Fully compatible with Hiwonder serial bus servos, it provides a robust and flexible solution for precise motor control in our robotic arm system [43].

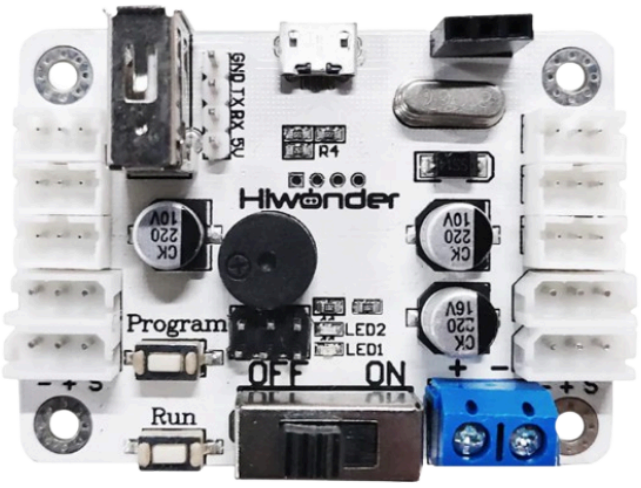

Figure 17. Motor Control Unit

4.2.4 Servomotors

We utilized the LX-15D Serial Bus Servo, as shown in Figure 18, which features dual bearings and full metal gears, ensuring high accuracy and an extended service life. This servo offers a large torque of 15 kg/cm (208 oz/in) at 6V and 17 kg/cm (236 oz/in) at 7.4V, with a control angle of 240°. It provides real-time feedback on position, temperature, and voltage, and includes an RGB color indicator that blinks in cases of overcurrent, over temperature, or locked-rotor conditions.

The dual ball bearings, comprising a driving ball bearing and an assistant ball bearing, enhance its performance and durability. Additionally, the servo can run continuously for a month without interruption. As a serial bus servo, it allows for setting the control angle and running speed, making it ideal for precise and reliable operation in our robotic arm [38].

Figure 18. LX-15D Serial Bus Servo

4.2.5 USB Camera

We incorporated a USB camera module, as shown in Figure 19, with a 640x480 VGA resolution at 30fps and an object distance ranging from 11.8 inches (30 cm) to infinity. This 130-degree wide-angle HD camera module, measuring 1.2 x 1 x 0.6 inches (30 x 25 x 14 mm) with a USB cable length of 9.9 inches (250 mm), is designed for use with Hiwonder/LewanSoul robots, Turbopi, PuppyPi, VisionPi, and GoGoPi.

Ideal for robot building and DIY webcam projects with Arduino and Raspberry Pi, this camera's high-definition capabilities and wide-angle view make it perfect for capturing detailed images and videos, enhancing the functionality and precision of our robotic arm [39].

Figure 19. USB Camera Module

4.2.6 Microcontroller

We utilized the Raspberry Pi 5 4GB, as shown in Figure 20, which offers a 3× increase in CPU performance with its 2.4GHz quad-core Cortex-A76 processor, ensuring smoother and faster computing for DIY projects, programming, and home automation. Equipped with the VideoCore VII GPU, it supports OpenGL ES 3.1 and Vulkan 1.2, providing rich visuals and smooth graphics for gaming and multimedia.

The dual-band 802.11ac Wi-Fi ensures seamless internet connectivity, while the M.2 SSD connector allows for faster data transfer and super-fast boot times, making it ideal for high-performance applications. The Raspberry Pi 5 also features 2 × USB 3.0 and 2 × USB 2.0 ports for simultaneous 5Gbps data transfer, making it perfect for connecting various devices and peripherals.

Additionally, it comes with Bluetooth 5.0 and Bluetooth Low Energy (BLE) for smooth wireless connections, and the M.2 SSD connector offers future-proof expansion options, enhancing the overall flexibility and performance of our robotic arm setup [40].

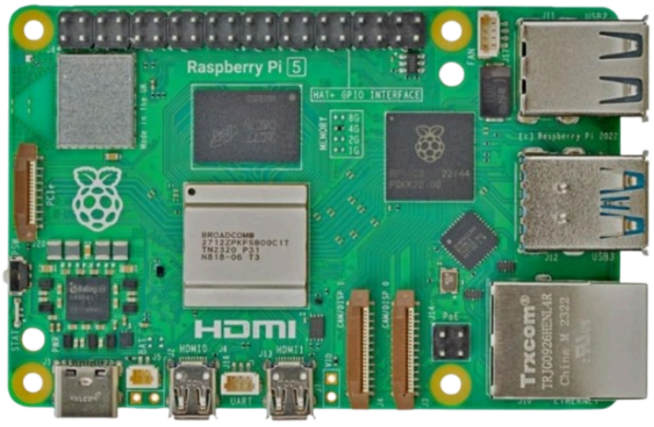

Figure 20. Raspberry Pi 5

4.2.7 Relay Module

For our graduation project, we utilized a 5V Single-Channel Relay Module, as shown in Figure 21, with Active High Trigger and Optocoupler Isolation to control the magnet via Raspberry Pi. Operating at 5V, the relay activates with a minimum 3.3V signal and supports a coil voltage of 5V, handling up to 250VAC at 10A / 125VAC at 10A.

Compact at 34 × 26 mm with 3 mm mounting holes, the module is ideal for integrating into our system. Featuring S, +, and – input pins, it connects 5V power to '+', ground to '–', and Raspberry Pi's GPIO signal to 'S'. The relay's outputs include NO (Normally Open) and NC (Normally Closed), with COM as the common terminal; when a high signal is sent from the Raspberry Pi, the relay switches contacts, linking COM to NO and disengaging COM from NC, effectively controlling the magnet.

This setup enables precise automation and actuator control, allowing efficient switching of electric devices, including magnets in our project [44].

Figure 21. Relay Module

4.2.8 Electrical Magnet

For our graduation project, we integrated a DC5V Electric Lifting Magnet, as shown in Figure 22, with a holding force of 2.5Kg to 3Kg, enabling precise control via Raspberry Pi. Operating at 5V DC, this compact solenoid features an M4 thread size, with a diameter of approximately 20 mm, a center diameter of about 7 mm, and a height of 15 mm, making it ideal for our setup.

The 25 cm wire length allows for flexible connectivity, ensuring seamless integration with our system. This electromagnet is a key component in our project, providing reliable lifting and holding functions for automated control [45].

Figure 22. Electric Magnet

4.3 Circuit Diagram

The circuit for the robotic arm is designed to ensure proper power distribution and control while preventing damage to components due to voltage limitations. The system is powered by an AC-to-DC power supply, which is connected to a standard wall outlet and provides a 12V, 10A DC output. Since the motors and motor control unit have a voltage

limit of 8.4V, an XL4016 DC-DC buck converter is used to step down the 12V input to 7.4V, ensuring safe operation of the motor control unit. The robotic arm consists of five motors connected in series, with Motor 1 interfacing directly with the control unit, followed sequentially by Motors 2, 3, 4, and 5. Each motor has a three-wire female connector, enabling smooth communication and control.

The Raspberry Pi sends control signals to the relay module's input, allowing precise activation of the magnet. The 7.4V output from the buck converter powers the relay module's positive and negative terminals, while the signal pin from Raspberry Pi controls the relay input. For the relay's output connections, COM is linked to the negative terminal of the relay, while one wire of the magnet is connected to the positive terminal, and the other is wired to the NC (Normally Closed) output of the relay. This configuration ensures that the magnet is properly controlled via the relay, enabling efficient switching for automation.

The Raspberry Pi is powered via a 5V, 5A adapter through USB Type-C, supporting connected peripherals such as a microphone and camera. Additionally, the motor control unit is interfaced via TTL USB, with proper RX-TX and GND connections ensuring stable communication between the Raspberry Pi and the motor system.

This circuit design provides reliable control over the robotic arm while maintaining voltage safety for all integrated components. The circuit diagram is shown below in Figure 23 for reference.

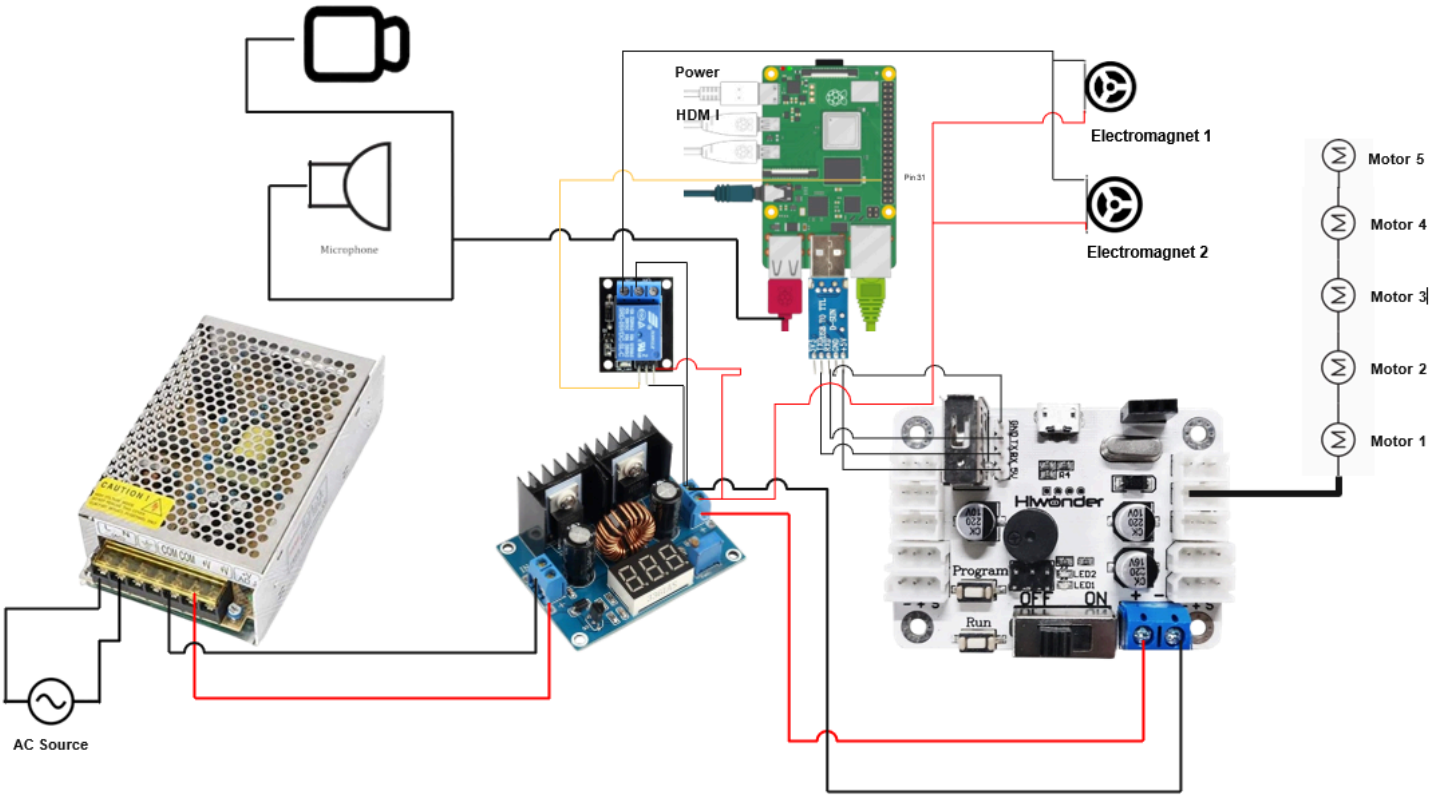

Figure 23. Circuit Diagram

The motor wiring follows a structured series connection, ensuring a streamlined energy flow across all motors. When the motor control unit is powered, it first directs electrical energy to Motor 1, which is directly interfaced with the control unit. This motor acts as the first recipient of power and passes it along to Motor 2. The same pattern continues with Motor 3, Motor 4, and finally Motor 5, forming a cascading energy distribution setup.

Since the motors are arranged in series, they all share the same current flow, meaning the voltage is distributed among them. This wiring approach ensures synchronized movement, as each motor receives power sequentially, allowing them to function in harmony. The three-wire female connectors on each motor facilitate both power transmission and control signals, ensuring that communication between the control unit and the motors remains stable. This configuration plays a crucial role in

maintaining an organized control system, optimizing performance, and protecting components from unexpected voltage spikes.

Additionally, because of the series arrangement, any disruption in one motor's connection could affect the entire system. Properly securing the wiring and ensuring reliable connections between each motor is essential for the robotic arm's consistent operation. Figure 24 visually represents this wiring structure, illustrating how power and signals flow seamlessly from the control unit to each motor in succession.

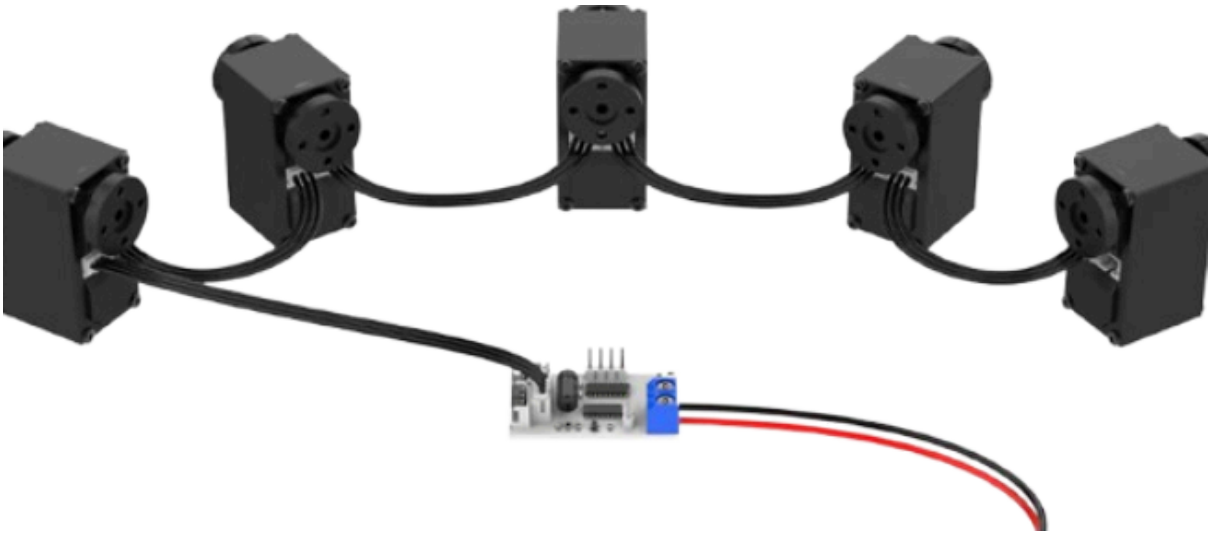

Figure 24. Motor Wiring Configuration

CHAPTER 5
SOFTWARE

5.1 Kinematics and Movement Control

The robotic arm in this project consists of five servo-driven joints and an end-effector operated by an electromagnetic relay. The movement control logic is implemented in Python using the serial_bus_servo_controller (SBSC) library, which interfaces with the servo controller board over a UART serial connection. Each movement command specifies the target PWM for the servo motors and the duration of the motion. The core functions managing these movements include arm initialization, dynamic positioning based on detected object location, and relay control for activating or deactivating the end-effector.

5.1.1 Initialization of Position

The robotic arm is initialized to a known default or "home" position at the start of the system to ensure that all servos begin from a consistent and calibrated configuration. This process is implemented through the init_arm() function, which issues a synchronized movement command to all five servo motors using the serial_bus_servo_controller library. Before this function is invoked, the controller object is created by initializing the serial bus communication as shown below:

```
controller = sbsc.SBS_Controller("/dev/ttyUSB0")
```

This line creates an instance of the servo controller by specifying the USB serial port (/dev/ttyUSB0) used for communication between the Raspberry Pi and the bus servo controller board. Once initialized, this controller object is used to send movement commands to individual or multiple servos. The actual initialization of the arm is performed using the following function:

```
def init_arm():
    controller.cmd_servo_move([1, 2, 3, 4, 5], [465, 320,
805, 905, 500], 2000)
    sleep(2)
```

This function calls cmd_servo_move() with a list of servo IDs ([1, 2, 3, 4, 5]) and corresponding target PWM ([465, 320, 805, 905, 500]). These angles represent the "home" configuration of the robotic arm. The final parameter 2000 specifies that the movement should be completed over 2000 milliseconds (2 seconds). The sleep(2) function is used immediately after the command to halt the execution of the program and allow the servos to complete their motion before any subsequent commands are issued. This ensures smooth and sequential arm behavior and prevents premature actions from being executed while the servos are still in motion. This home position is essential for consistent operation and serves as the starting point for all subsequent movement sequences in the system.

5.1.2 Relay Control

The electromagnet is controlled through a relay module connected to a Raspberry Pi GPIO pin 6. The relay acts as a switch to turn the magnet on and off. The setup and control are defined as follows:

```
RELAY_PIN = 6
relay = gpiozero.OutputDevice(RELAY_PIN,
active_high=True, initial_value=False)

def set_relay(status):
    print("Relay ON" if status else "Relay OFF")
    relay.on() if status else relay.off()
```

The relay is initialized to be active-high on GPIO pin 6. The function set_relay(True) activates the electromagnet to grasp a tool, while set_relay(False) deactivates it to release the tool. These relay toggles

are synchronized with the arm movement to ensure the tool is only picked or dropped at the correct moment.

5.1.3 X-Axis Mapping

To identify the tool's location in the workspace, the system uses YOLOv8 for real-time object detection. Once a tool is detected, its bounding box is processed to extract its horizontal position (x-axis center), which is used to determine the correct servo movement sequence. This is done in the object_visible() function:

```
x1, y1, x2, y2 = map(int, box.xyxy[0])
...
return (x2 - x1) / 2 + x1
```

Here, x1 and x2 represent the horizontal boundaries of the detected object. The x-coordinate of the object center is calculated as (x2 - x1)/2 + x1, which is later used to determine which predefined zone the object lies in.

5.1.4 Movement on X-Axis Zone

The detected x coordinate is compared against four predefined positions: 73, 138, 228, and 320. A threshold value (set to 30) is used to define acceptable ranges around these positions. Depending on which range the object falls into, the system selects a matching set of servo angles to move the arm to the object's location.

```
threshold = 30

if x <= 73 + threshold and x >= 73 - threshold:
    controller.cmd_servo_move([1, 2, 3, 4, 5], [615, 815,
455, 865, 970], 3000)
    sleep(3)

elif x <= 138 + threshold and x >= 138 - threshold:
    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 750,
```

```
645, 785, 915], 3000)
    sleep(3)

elif x <= 228 + threshold and x >= 228 - threshold:
    controller.cmd_servo_move([1, 2, 3, 4, 5], [420, 710,
645, 785, 815], 3000)
    sleep(3)

elif x <= 320 + threshold and x >= 320 - threshold:
    controller.cmd_servo_move([1, 2, 3, 4, 5], [310, 790,
450, 910, 700], 3000)
    sleep(3)
```

The system determines the object's position based on whether it falls within ±30 pixels of one of the four target x-locations. If the x-coordinate is approximately 73, the arm moves to the predefined positions [615, 815, 455, 865, 970]. When x is around 138, the arm shifts to [535, 750, 645, 785, 915]. Similarly, if x is close to 228, the arm moves to [420, 710, 645, 785, 815]. Lastly, when x is near 320, the arm moves to [310, 790, 450, 910, 700]. These movement values were carefully calibrated to ensure proper alignment with the expected physical locations of the tools within the workspace, optimizing precision and efficiency.

5.1.5 Handling Sequence

The complete handling logic for a tool (e.g., "forceps") includes voice recognition, object detection, x-position analysis, servo movement, relay activation, and drop detection. A simplified example is shown below:

```
if spoken_result.lower() in forceps:
    if object_visible("forceps"):
        x = object_visible("forceps")
        while x <= 0:
            x = object_visible("forceps")
        if x <= 138 + threshold and x >= 138 - threshold:
            print("✅ Forceps confirmed by voice and
```

```
vision")
            controller.cmd_servo_move([1, 2, 3, 4, 5],
[535, 750, 645, 785, 915], 3000)
            sleep(3)
            set_relay(True)
            sleep(4)
```

In this control sequence, the arm operates only after both voice and vision systems confirm the tool's identity. Once validated, the x-coordinate of the object is assessed. If it falls within the second zone, approximately 138 pixels, the arm moves to its designated pose. At this stage, the relay is activated to turn on the electromagnet, enabling the arm to securely grasp the tool. This logical structure is consistently applied for each tool type, with variations in servo angles tailored to the specific zone to ensure precise and efficient movements.

5.2 Computer Vision

The computer vision component of the robotic arm system is responsible for two key tasks: (1) identifying and localizing surgical tools in real time using YOLO object detection, and (2) recognizing hand gestures—specifically, detecting an open hand—to trigger the release of tools. These two functions operate concurrently and complement each other to ensure both accurate grasping and safe tool transfer. The entire vision system is implemented using OpenCV for video processing, Ultralytics YOLO for object detection, and MediaPipe for real-time hand tracking.

5.2.1 YOLO Object Detection

YOLOv8 (You Only Look Once version 8) is used for detecting surgical tools in live video frames captured from the webcam. The model is initialized at the start of the script as follows:

```
from ultralytics import YOLO
model = YOLO('ourdata.pt')
```

This loads a pre-trained YOLO model from a custom dataset (ourdata.pt), which is capable of detecting tool classes such as "forceps", "scalpel", "iris", and "hemostat". The main detection logic is handled inside the object_visible(target) function, which attempts to detect a given object class up to 30 times in a loop to ensure stability and reliability:

```
for i in range(30):  # Try for up to ~1 second
    ret, frame = cap.read()
    ...
    results = model.predict(frame, stream=False,
imgsz=256, conf=0.4)
```

Each frame is resized to a fixed dimension (400x400) for consistent performance, and the YOLO model returns a list of detected bounding boxes. For each box, the class label (name) and confidence score (conf) are extracted and compared to the target object name:

```
for box in results[0].boxes:
    cls = int(box.cls[0])
    name = results[0].names[cls].lower()
    conf = float(box.conf[0])
```

If a match is found (i.e., the detected class name matches the spoken target and the confidence is ≥ 0.4), a bounding box and label are drawn on the video frame using OpenCV:

```
cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
cv2.putText(frame, label, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

Most importantly, the horizontal center of the detected object (x) is calculated using the following line:

```
return (x2 - x1) / 2 + x1
```

This x value is critical for the kinematics logic. It is passed to a positional threshold comparison system that determines which predefined servo movement sequence should be triggered. If the object is not detected after 30 frames, the function returns 0 as a fallback. The real-time output is shown in a window using:

```
cv2.imshow("YOLO Object Detection", frame)
```

This allows the user to visually verify what the system is detecting during execution.

5.2.2 Hand Gesture Detection

Hand gesture recognition is implemented using Google's MediaPipe Hands solution, which provides 3D hand landmark detection in real time. The MediaPipe module is initialized at the beginning of the main execution block:

```
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
hands = mp_hands.Hands(static_image_mode=False,
max_num_hands=1,
                       min_detection_confidence=0.7,
min_tracking_confidence=0.7)
```

This initializes a hand tracking model that processes live video frames to detect one hand and returns 21 landmark points per frame. The gesture detection is encapsulated in the drop() function, which is responsible for determining whether the user's hand is open (a signal to drop the tool). The core of the gesture detection is based on this helper function:

```
def is_hand_open(landmarks, handedness):
    finger_tips = [8, 12, 16, 20]
    finger_pips = [6, 10, 14, 18]
    return all(landmarks[tip].y < landmarks[pip].y for
tip, pip in zip(finger_tips, finger_pips))
```

This function checks whether the y-coordinate of each fingertip is above (i.e., closer to the top of the image) than its corresponding pip joint. If all fingers are extended, the hand is considered open. In the drop() function, each video frame is processed as follows:

```
frame = cv2.flip(frame, 1)
image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
result = hands.process(image_rgb)
```

MediaPipe returns a list of detected hand landmarks. If a hand is found, the landmarks are drawn on the frame using:

```
mp_drawing.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
```

A text label is displayed to indicate the state of the hand:

```
cv2.putText(frame, 'Open Hand', (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 255, 0), 3)
```

The function returns 1 if the hand is open and 0 otherwise:

```
return hand_open
```

In the main control loop (for each tool type), a "drop check" loop is performed to ensure the hand remains open for 25 consecutive frames before releasing the tool:

```
repeat = 0
while repeat < 25:
    if drop():
        repeat += 1
        print(f"Hand open count: {repeat}")
    else:
        repeat = 0
```

This safeguards against accidental releases due to brief or unintended gestures, making the tool drop action both intentional and reliable.

5.3 Speech Recognition

The speech recognition component of the system serves as the initial interface between the user and the robotic arm, enabling the selection of a surgical tool through voice commands. This functionality enhances hands-free control, which is particularly beneficial in clinical or surgical environments where manual interaction may be limited due to sterility or workflow constraints. To implement this feature, the system utilizes the speech_recognition library in Python. Two main objects are instantiated at the beginning of the program:

```
recognizer = sr.Recognizer()
mic = sr.Microphone()
```

The Recognizer object (recognizer) is responsible for processing audio data, while the Microphone object (mic) interfaces with the system's default audio input device. This setup allows the system to continuously listen for voice input from the user. Within the main control loop, the system listens for a voice command using the microphone. This is done using a with statement to ensure proper management of audio resources:

```
with mic as source:
    print("[Speech] Listening for voice command")
```

```
    audio = recognizer.listen(source)
```

Here, the listen() method records audio input until a pause is detected. This captured audio is then passed to a speech-to-text engine for transcription. The system uses Google's free speech recognition API to transcribe the recorded audio into text:

```
spoken_result = recognizer.recognize_google(audio)
```

This method attempts to interpret the user's speech and return a string representation of the spoken command. The output is stored in the spoken_result variable and subsequently analyzed to determine the intended tool. Once the speech is transcribed, the system performs string matching to identify keywords that correspond to available surgical tools. This step is essential due to the possibility of misrecognition or phonetic similarity. The logic is implemented as follows:

```
if "forceps" in spoken_result.lower() or "forces" in
spoken_result.lower() or "46" in spoken_result.lower():
    selected_tool = "forceps"
    print("[Speech] Selected tool: Forceps")
```

This block checks whether the transcription contains any of several variations of the word “forceps,” including possible mishearings such as “forces” or even numeric confusion like “46.” If a match is found, the selected_tool variable is set accordingly.

5.4 Machine Learning

The machine learning component of the system is built upon the YOLOv8 object detection framework provided by the Ultralytics library. The objective was to train a custom model capable of detecting four types of surgical tools: forceps, iris, hemostat, and scalpel. The model was implemented using a Python script that defines the model architecture and training process as shown below:

```
from ultralytics import YOLO

def main():
    model = YOLO("yolov8s.yaml")  # build a new model
from scratch
    results = model.train(
        data="data.yaml",
        epochs=(variable),
        device=0  # Set to 0 for GPU, or 'cpu' for CPU
    )

if __name__ == "__main__":
    main()
```

In this script, the YOLO("yolov8s.yaml") command initializes a new model using the YOLOv8-small architecture, which is optimized for fast inference on edge devices while maintaining good detection accuracy. The training process is configured via the train() method, where the data="data.yaml" parameter specifies the dataset configuration file containing the paths to the training and validation images, as well as the list of class names.

The dataset used in this project was custom-built. A total of 1,419 original images were collected using a static camera setup under controlled lighting conditions. Each image was manually annotated using bounding boxes to label the four target classes. This annotation process was performed using a tool called roboflow, with annotations saved in YOLO format. To improve the robustness and generalizability of the model, several data augmentation techniques were applied, including rotation, scaling, flipping, and color adjustments. As a result, the dataset was expanded to 4,257 images in total—significantly increasing the variation and diversity of training samples without requiring more physical image collection.

The number of training epochs (denoted as (variable) in the code) was selected based on early evaluation metrics such as loss curves and mean average precision (mAP) trends. Training was performed on a

GPU-enabled system by setting device=0, which greatly accelerated the training time and allowed for more iterations during hyperparameter tuning.

Upon completion of training, the model weights and performance logs were saved automatically in the default Ultralytics directory (runs/detect/train/). These trained weights were then used in the live robotic arm system to perform real-time object detection and localization of surgical tools. This pipeline ensured that the robotic arm could recognize and interact with physical tools with high accuracy and speed, thanks to the custom training performed on a domain-specific dataset.

CONCLUSION

6.1 Objective Fulfilment

The surgical assistance robotic arm project has successfully achieved its objectives by designing, programming, and implementing a multifunctional robotic arm to enhance surgical procedures. The CNC-cut structure ensures precision and durability, allowing the arm to execute movements essential for surgical applications. Advanced image processing and voice command integration enable seamless tool identification and handling, improving surgical efficiency and reducing the risk of contamination.

Throughout the development process, several challenges arose, particularly in refining the programming and ensuring smooth integration of various technologies. While work on inverse kinematics is still in progress, the robotic arm has been designed to allow controlled movement and precise tool handling. These obstacles were addressed through iterative design improvements and extensive refinement of control algorithms. Additionally, resource management was carefully planned to maintain cost-effectiveness while ensuring high performance. The use of DC servo bus motors further contributed to stability and reliability, enhancing the overall functionality of the robotic arm.

The impact of the robotic arm in surgical assistance is significant, providing a safer and more efficient method for handling surgical tools. By automating critical aspects of tool management, the arm minimizes contamination risks and enables surgeons to focus on complex tasks without interruptions.

.6.2 Future Improvments

As the surgical assistance robotic arm continues to evolve, several enhancements can be explored to refine its performance, expand its capabilities, and improve its overall efficiency. One key area for improvement is the completion and optimization of the inverse kinematics system, ensuring smoother and more precise movement control. Enhancing this aspect will allow the robotic arm to execute

complex surgical maneuvers with greater accuracy and responsiveness, making it more adaptable to various medical procedures.

Another potential upgrade involves integrating artificial intelligence-driven adaptability, enabling the robotic arm to learn and adjust to different surgical tasks dynamically. This could involve machine learning algorithms that refine tool-handling techniques based on real-time feedback, improving the accuracy and efficiency of automated operations. Additionally, expanding the range of surgical tools the robotic arm can handle will increase its versatility, allowing it to support a wider array of specialized procedures.

Improving user interaction and control systems is another area of focus. Enhancing voice command recognition, introducing intuitive gesture-based controls, or even developing a user-friendly graphical interface for surgeons to customize movements could make the arm more accessible and efficient in surgical environments. Additionally, refining safety mechanisms—such as improved force feedback and collision detection—would ensure the arm operates seamlessly while minimizing risks during procedures.

Material improvements and structural refinements could also enhance the robotic arm's durability and precision. Upgrading key components, optimizing the weight-to-strength ratio, or exploring alternative manufacturing techniques may result in a more reliable and efficient design. Furthermore, refining the DC servo bus motor control system would allow for smoother movement, better accuracy, and increased operational reliability in high-demand surgical tasks.

.6.3 Personal Conclusions

The development of the surgical assistance robotic arm has been an insightful and rewarding journey, filled with challenges, learning experiences, and significant accomplishments. Seeing the project come to life—from the initial concept to the functional robotic arm—has reinforced the importance of persistence, teamwork, and creative problem-solving in engineering and medical innovation. The ability to integrate advanced technologies such as image processing, voice

commands, and precise motor control has demonstrated the vast potential of robotics in surgical applications, highlighting the impact automation can have on improving efficiency and safety in medical procedures.

One of the most valuable lessons learned throughout this project was the importance of adaptability and iterative improvements. While certain elements, such as the inverse kinematics, are still being refined, the process itself provided invaluable insights into programming, hardware integration, and system optimization. Each challenge faced along the way, whether in design modifications or technology integration, contributed to a deeper understanding of robotic automation and its potential for surgical environments. The experience emphasized the necessity of continuous learning, troubleshooting, and refining solutions to achieve the best possible outcomes.

Another key takeaway has been recognizing the significance of robotics in supporting human capabilities rather than replacing them. This project reaffirmed that the goal of automation in surgery is not to eliminate human involvement but to enhance precision, reduce risks, and assist medical professionals in performing delicate procedures more effectively. The robotic arm's ability to provide consistent, fatigue-free assistance ensures a safer surgical environment, reinforcing the potential for robotics to serve as essential tools in modern medicine.

Overall, this project has been a remarkable opportunity to explore the intersection of engineering and healthcare, combining theoretical knowledge with practical implementation. The progress made so far is a testament to the dedication and innovation behind the project, and it serves as a stepping stone for future advancements in surgical robotics. While improvements and refinements will continue, the accomplishments achieved thus far have laid a strong foundation for future development, inspiring further exploration in medical technology and robotic applications.

REFERENCES

[1] Moran, M. E. (2007). Evolution of robotic arms. Journal of Robotic Surgery, 1(2), 103-111. DOI: 10.1007/s11701-006-0002-x

[2] Wongphati, M., Matsuda, Y., Osawa, H., & Imai, M. (2012). What do you want from the robot? IEEE International Symposium on Robot and Human Interactive Communication.

[3] W.-T. Ma, W.-X. Yan, Z. Fu, and Y.-Z. Zhao, “A chinese cooking robot for elderly and disabled people,” Robotica, vol. 29, no. 06, pp. 843–852, 2011.

[4] R. L. Cahlander, D. W. Carroll, R. A. Hanson, A. Hollingsworth, and J. O. Reinertsen, “Food preparation robot,” USA Patent 4,922,435, 1990.

[5] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe, “Herb: A home exploring robotic butler,” 2009, doi:10.1007/s10514-009-9160-9.

[6] S. Hata, T. Hiroyasu, J. Hayashi, H. Hojoh, and T. Hamada, “Robot system for cloth handling,” in 34th Annual Conf. of IEEE Industrial Electronics, nov. 2008, pp. 3449 –3454

[7] C.-H. King, T. Chen, A. Jain, and C. Kemp, “Towards an assistive robot that autonomously performs bed baths for patient hygiene,” in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2010, pp. 319 –324.

[8] A. Jain and C. Kemp, “El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces,” Autonomous Robots, vol. 28, pp. 45–64, 2010.

[9] A. Baca and P. Kornfeind, “Rapid feedback systems for elite sports training,” IEEE Pervasive Computing, vol. 5, no. 4, pp. 70 –76, oct.- dec. 2006.

[10] J. Solis, S. Marcheschi, A. Frisoli, C. A. Avizzano, and M. Bergamasco, “Reactive robot system using a haptic interface: an active interaction to transfer skills from the robot to unskilled persons,” Advanced Robotics, vol. 21, no. 3-4, pp. 267–291, 2007.

[11] J. Zhang and A. Knoll, “A two-arm situated artificial communicator for human-robot cooperative assembly,” IEEE Trans. on Industrial Electronics, vol. 50, no. 4, pp. 651–658, 2003

[12] Y. Li and Q. Xu, “Design and development of a medical parallel robot for cardiopulmonary resuscitation,” IEEE/ASME Trans. on Mechatronics, vol. 12, no. 3, pp. 265 –273, june 2007.
[13] D. Gerhardus, “Robot-assisted surgery: the future is here,” Journal of Healthcare Management, vol. 48, no. 4, pp. 242 – 251, 2003.
[14] R. Kaushal, K. N. Barker, and D. W. Bates, “How can information technology improve patient safety and reduce medication errors in children’s health care?” Arch Pediatr Adolesc Med, vol. 155, no. 9, pp. 1002–1007, 2001.
[15] J. Patton and F. Mussa-Ivaldi, “Robot-assisted adaptive training: custom force fields for teaching movement patterns,” IEEE Trans. on Biomedical Engineering, vol. 51, no. 4, pp. 636 –646, april 2004.
[16] J. Hammel, K. Hall, D. Lees, L. Leifer, M. V. der Loos, I. Perkash, and R. Crigler, “Clinical evaluation of a desktop robotic assistant,” J. Rehabil. Res. Dev., vol. 26, pp. 1–16, 1989.
[17] Lanfranco AR, Castellanos AE, Desai JP, Meyers WC. Robotic Surgery: A Current Perspective. Ann Surg. 2004;239:14-21.
[18] Satava RM, Bowersox JC, Mack M, et al. Robotic surgery: state of the art and future trends. Contemp Surg. 2001;57:489–499.
[19] Kwoh YS, Hou J, Jonckheere EA, et al. A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery. IEEE Trans Biomed Eng. 1988;35:153–161.
[20] Kim VB, Chapman WH, Albrecht RJ, et al. Early experience with telemanipulative robot-assisted laparoscopic cholecystectomy using Da Vinci. Surg Laparosc Endosc Percutan Tech. 2002;12:34–40.
[21] Cadierre GB, Himpens J. Feasibility of robotic laparoscopic surgery: 146 cases. World J Surg. 2001;25:1467–1477.
[22] Falcone T, Goldberg JM, Margossian H, et al. Robotic-assisted laparoscopic microsurgical tubal anastomosis: human pilot study. Fertil Steril. 2000;73:1040–1042.
[23] Zhang, W., Li, H., Cui, L., et al. (2021). Research progress and development trend of surgical robot and surgical instrument arm. International Journal of Medical Robotics and Computer Assisted Surgery, 17(5), e2309. https://doi.org/10.1002/rcs.2309
[24] Tae K, Ji YB, Song CM, Ryu J. Robotic and endoscopic thyroid surgery: evolution and advances. Clin Exp Otorhinolaryngol. 2019;12(1):1-11.

[25] Liu KS, Jiang L. Multifunctional integration: from biological to bio-inspired materials. ACS Nano. 2011;5:6786-6790.
[26] Park M, Bok BG, Ahn JH, Kim M-S. Recent advances in tactile sensing technology. Micromachines. 2011;9:321. https://doi.org/10.3390/mi9070321
[27] Cianchetti M, Laschi C, Menciassi A, Dario P. Biomedical applications of soft robotics. Nature Rev Mater. 2018;3:143-153.
[28] Bencsik AL, Lendvay M. Development trends in bio-mechatronics. International Symposium on Intelligent Systems and Informatics. 2016:117-122.
[29] Hauser S, Mutlu M, Leziart PA, Khodr H, Bernardino A, Ijspeert AJ. Roombots extended: challenges in the next generation of self-reconfigurable modular robots and their application in adaptive and assistive furniture. Robot Auto Syst. 2020;127: 103467.
[30] Goldenberg AA, Yang Y, Ma L, et al. Surgical robot. US Patent 9,974,619, U.S. Patent and Trademark Office; 2018.
[31] Anwar S, Prasad R. Framework for future telemedicine planning and infrastructure using 5G technology. Wirel Pers Commun. 2018;100:193-208.
[32] Zhou XY, Guo Y, Shen ML, Yang G-Z. Application of artificial intelligence in surgery. Front Med. 2020;14:417-430.
[33]Zhai JM, Li LZ, Guo PS, et al. Design and experiment on M2M2A of multi-robot intelligent collaboration. Robot. 2017;39:415-422
[34] "Da Vinci in the world." Synektik, 14 Jan. 2025, https://synektik.com.pl/en/da-vinci-in-the-world/.
[35] Ameen, I. (2021, March 18). Robotic Surgery: History, Advantages, Disadvantages and Applications. Healthcare Business Club. Retrieved March 20, 2025, from https://healthcarebusinessclub.com/articles/healthcare-provider/technology/robotic-surgery-history-advantages-disadvantages-and-applications/
[36] Raj, S. A., Muthukumaran, E., & Jayakrishna, K. (2018). A Case Study of 3D Printed PLA and Its Mechanical Properties. Materials Today: Proceedings, 5(5), 11219–11226. doi:10.1016/j.matpr.2018.01.146
[37] Doe, John, and Alice Smith. "A Comparative Study on the Material Selection of a Spot Welding Robotic Arm." Journal of Robotics 12, no. 3 (2025): 45-60. https://www.researchgate.net/publication/386017746_A_COMPARATIVE

_STUDY_ON_THE_MATERIAL_SELECTION_OF_A_SPOT_WELDING_ROBOTIC_ARM.
[38] Institute of Environmental Sciences and Technology. (n.d.). ISO standards. IEST. Retrieved March 24, 2025, from https://www.iest.org/Standards-RPs/ISO-Standards
[38]https://www.amazon.com/LX-15D-Durable-Steering-Indicator-15KG-CM/dp/B07G11VJWR
[39]https://www.amazon.com/640x480-Building-Arduino-Raspberry-Robotic/dp/B0CR41XSBY?th=1
[40]https://a.co/d/eELuPtm
[41]https://makerselectronics.com/product/power-supply-12v-10a
[42]https://makerselectronics.com/product/xl4016-dc-dc-buck-converter-8a-step-down-4-36v-to-1-25-36v-with-display-xh-m404
[43]https://www.hiwonder.com/collections/servo-controller/products/serial-bus-servo-controller
[44]https://makerselectronics.com/product/relay-module-1-channel-active-high-5v
[45]https://amzn.eu/d/1en91hy

APPENDIX A : SURVEY DATA

Before finalizing the application of our robotic arm project, we conducted a survey to gather insights on potential use cases and community needs. The survey included three key sections: participant demographics, proposed robotic arm applications, and preference selection.

In the first section, participants provided basic details such as their name, age, and profession, allowing us to analyze responses based on diverse backgrounds Figure 25. This helped us understand the perspectives of individuals from different fields and how their professional experience influenced their preferences.

Figure 25. Survey Participants Demographics

Next, we presented multiple potential applications for the robotic arm, asking participants to evaluate its usefulness in various domains Figure 26. These options covered fields such as industrial automation, medical assistance, rehabilitation, and surgical support, among others.

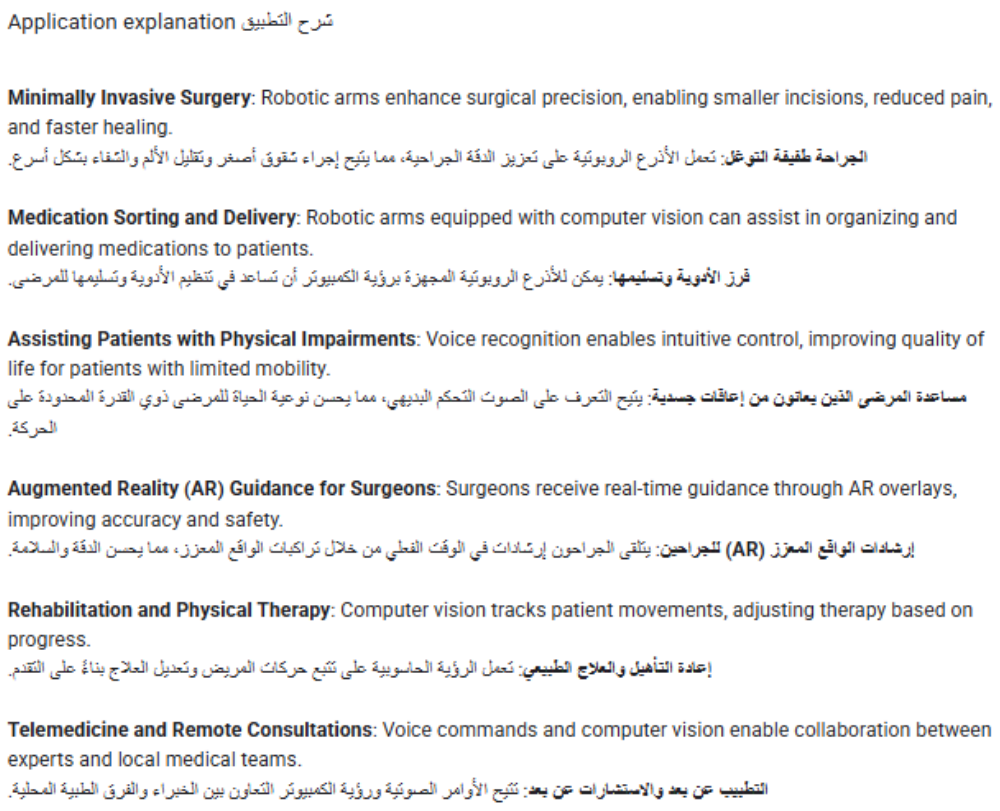

Figure 26. Proposed Applications

Finally, respondents were asked to select the application they found most beneficial either for themselves or for the broader community. After collecting 188 responses, the majority expressed a preference for assisting patients with physical impairments Figure 27. While this option received significant support, we ultimately determined that a surgical assistance robotic arm would be the most functional and impactful choice. This decision was based on its potential to improve precision, efficiency, and safety in medical procedures, ensuring a valuable contribution to healthcare advancements.

188 responses

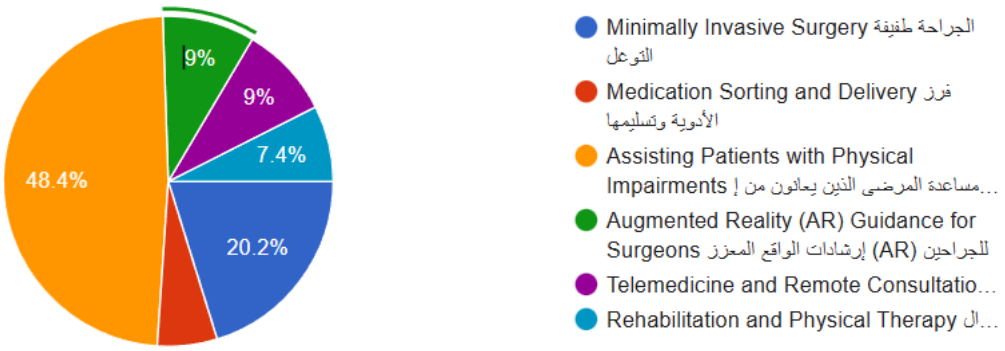

Figure 27. Survey Result

APPENDIX B: CODE LISTINS

Below is the complete code used for the development and operation of the surgical assistance robotic arm. This code includes the programming logic for movement control, tool identification, voice command processing, and integration with various system components.

```
import serial_bus_servo_controller as sbsc
from time import sleep
import RPi.GPIO as GPIO
import speech_recognition as sr
import cv2
import mediapipe as mp
import gpiozero
from ultralytics import YOLO
import time

# ----------------- Relay Setup -----------------
RELAY_PIN = 6
relay = gpiozero.OutputDevice(RELAY_PIN, active_high=True,
initial_value=False)
threshold = 30
def set_relay(status):
    print("Relay ON" if status else "Relay OFF")
    relay.on() if status else relay.off()

# ----------------- Arm Setup -----------------
def init_arm():
    controller.cmd_servo_move([1, 2, 3, 4, 5], [465, 320, 805, 905, 500],
2000)
    sleep(2)

# ----------------- Hand Detection -----------------
def is_hand_open(landmarks, handedness):
    finger_tips = [8, 12, 16, 20]
    finger_pips = [6, 10, 14, 18]
    return all(landmarks[tip].y < landmarks[pip].y for tip, pip in
zip(finger_tips, finger_pips))

def drop():
    success, frame = cap.read()
    if not success:
        return 0
    frame = cv2.flip(frame, 1)
```

```
    image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = hands.process(image_rgb)

    hand_open = 0
    if result.multi_hand_landmarks and result.multi_handedness:
        for hand_landmarks, hand_info in zip(result.multi_hand_landmarks,
result.multi_handedness):
            mp_drawing.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
            if is_hand_open(hand_landmarks.landmark,
hand_info.classification[0].label):
                cv2.putText(frame, 'Closed hand', (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 0, 255), 3)
                hand_open = 0
            else:
                cv2.putText(frame, 'Open Hand', (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 255, 0), 3)
                hand_open = 1

    cv2.imshow("YOLO Object Detection", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        print("User pressed q. Exiting.")
        raise KeyboardInterrupt
    return hand_open

# ----------------- YOLO Object Detection -----------------
model = YOLO('ourdata.pt')

def object_visible(target):
    try:
        print(f"[INFO] Searching for '{target}'...")
        for i in range(30):  # Try for up to ~1 second
            ret, frame = cap.read()
            if not ret:
                continue
            frame = cv2.resize(frame, (400, 400))

            results = model.predict(frame, stream=False, imgsz=256,
conf=0.4)
            detected = False

            if results and results[0].boxes is not None:
                for box in results[0].boxes:
                    cls = int(box.cls[0])
                    name = results[0].names[cls].lower()
                    conf = float(box.conf[0])
                    x1, y1, x2, y2 = map(int, box.xyxy[0])
```

```
                    color = (0, 255, 0) if name == target.lower() else (0,
0, 255)
                    label = f'{name} {conf:.2f}'

                    cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
                    cv2.putText(frame, label, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

                    if name == target.lower() and conf >= 0.4:
                        detected = True

            cv2.imshow("YOLO Object Detection", frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                raise KeyboardInterrupt

            if detected:
                print(f"[YOLO] Detected: {target}")
                return True

        print(f"[YOLO] '{target}' not found.")
        return False

    except Exception as e:
        print(f"[ERROR] Object detection error: {e}")
        return False

    finally:
        #cv2.destroyWindow("YOLO Object Detection")
        try:
            return (x2-x1)/2 + x1
        except:
            return 0
# ---------------- Main Execution ----------------
try:
    # Init modules
    mp_hands = mp.solutions.hands
    mp_drawing = mp.solutions.drawing_utils
    hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1,
                           min_detection_confidence=0.7,
min_tracking_confidence=0.7)

    # Shared camera
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        raise SystemExit("[ERROR] Could not open camera")

    GPIO.setmode(GPIO.BCM)
    controller = sbsc.SBS_Controller("/dev/ttyUSB0")
```

```
    init_arm()

    while True:
        r = sr.Recognizer()
        spoken_result = "none"
        with sr.Microphone() as source:
            print("ðŸŽ™ï¸ Say something!")
            audio = r.record(source, duration=3)
            try:
                spoken_result = r.recognize_google(audio)
                print(f"SR result: {spoken_result}")
            except sr.UnknownValueError:
                print("â Could not understand audio")
                continue
            except sr.RequestError as e:
                print(f"â Speech recognition error: {e}")
                continue

        # -------- Handle "forceps" --------
        forceps = ["forceps" , "46" , "forces" ]
        if spoken_result.lower() in forceps :
            if object_visible("forceps"):
                x = object_visible("forceps")
                while x <= 0:
                    x = object_visible("forceps")
                    if x > 0:
                        break
                if x <= 73 + threshold and x >= 73 - threshold:
                    print("âœ... Forceps confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [615, 815,
455, 865, 970], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 138 + threshold and x >= 138 - threshold:
                    print("âœ... Forceps confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 750,
645, 785, 915], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(4)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 665,
645, 785, 915], 1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
```

```
780, 775, 500], 4000)
                    sleep(4)
                elif x <= 228 + threshold and x >= 228 - threshold:
                    print("âœ... Forceps confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [420, 710,
645, 785, 815], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 320 + threshold and x >= 320 - threshold:
                    print("âœ... Forceps confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [310, 790,
450, 910, 700], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [320, 660,
555, 835, 700],1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)

                repeat = 0
                while repeat < 25:
                    if drop():
                        repeat += 1
                        print(f"Hand open count: {repeat}")
                    else:
                        repeat = 0

                set_relay(False)
                init_arm()

        # -------- Handle "scalpel" --------
        scalpel = ["scalpel" , "scalp" ,"skeleton" , "sculpin" , "scandal" ,
"Scarborough"]
        if spoken_result.lower() in scalpel:
            if object_visible("scalpel"):
                x = object_visible("scalpel")
                while x <= 0:
                    x = object_visible("scalpel")
                    if x > 0:
                        break
                if x <= 73 + threshold and x >= 73 - threshold:
```

```
                    print("âœ... scalpel confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [615, 815,
455, 865, 970], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 138 + threshold and x >= 138 - threshold:
                    print("âœ... scalpel confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 750,
645, 785, 915], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(4)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 665,
645, 785, 915], 1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)
                elif x <= 228 + threshold and x >= 228 - threshold:
                    print("âœ... scalpel confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [420, 710,
645, 785, 815], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 320 + threshold and x >= 320 - threshold:
                    print("âœ... scalpel confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [310, 790,
450, 910, 700], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [320, 660,
555, 835, 700],1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)

                repeat = 0
                while repeat < 25:
```

```
                    if drop():
                        repeat += 1
                        print(f"Hand open count: {repeat}")
                    else:
                        repeat = 0

                set_relay(False)
                init_arm()


    # -------- Handle "iris" --------
        if "iris" in spoken_result.lower():
            if object_visible("iris"):
                x = object_visible("iris")
                while x <= 0:
                    x = object_visible("iris")
                    if x > 0:
                        break
                if x <= 73 + threshold and x >= 73 - threshold:
                    print("âœ... iris confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [615, 815,
455, 865, 970], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 138 + threshold and x >= 138 - threshold:
                    print("âœ... iris confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 750,
645, 785, 915], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(4)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 665,
645, 785, 915], 1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)
                elif x <= 228 + threshold and x >= 228 - threshold:
                    print("âœ... iris confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [420, 710,
645, 785, 815], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
```

```
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 320 + threshold and x >= 320 - threshold:
                    print("âœ... iris confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [310, 790,
450, 910, 700], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [320, 660,
555, 835, 700],1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)

                repeat = 0
                while repeat < 25:
                    if drop():
                        repeat += 1
                        print(f"Hand open count: {repeat}")
                    else:
                        repeat = 0

                set_relay(False)
                init_arm()

    # -------- Handle "hemostat" --------
        hemostat=["hemostat", "thermostate" , "Hampstead" ]
        if spoken_result.lower() in hemostat:
            if object_visible("hemostat"):
                x = object_visible("hemostat")
                while x <= 0:
                    x = object_visible("hemostat")
                    if x > 0:
                        break
                if x <= 73 + threshold and x >= 73 - threshold:
                    print("âœ... hemostat confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [615, 815,
455, 865, 970], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 138 + threshold and x >= 138 - threshold:
```

```
                    print("âœ... hemostat confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 750,
645, 785, 915], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(4)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [535, 665,
645, 785, 915], 1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)
                elif x <= 228 + threshold and x >= 228 - threshold:
                    print("âœ... hemostat confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [420, 710,
645, 785, 815], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 2000)
                    sleep(2)
                elif x <= 320 + threshold and x >= 320 - threshold:
                    print("âœ... hemostat confirmed by voice and vision")
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [310, 790,
450, 910, 700], 3000)
                    sleep(3)
                    set_relay(True)
                    sleep(2)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [320, 660,
555, 835, 700],1500)
                    sleep(1.5)
                    controller.cmd_servo_move([1, 2, 3, 4, 5], [870, 415,
780, 775, 500], 4000)
                    sleep(4)

                repeat = 0
                while repeat < 25:
                    if drop():
                        repeat += 1
                        print(f"Hand open count: {repeat}")
                    else:
                        repeat = 0

                set_relay(False)
                init_arm()

except KeyboardInterrupt:
```

```
    print("\nðŸ›' Interrupted by user")

finally:
    GPIO.cleanup()
    cap.release()
    cv2.destroyAllWindows()
    set_relay(False)
    print("âœ... Program terminated")
```

APPENDIX C: MACHINE LEARNING RESULT

Despite achieving an 80% accuracy, the model performed exceptionally well, successfully detecting the trained objects with slight variations in confidence levels.

The losses graph and training progression, shown in Figure 28, illustrates how the model learned over time, showcasing stable convergence and reliable performance. While confidence levels varied slightly, the model demonstrated strong detection capability across the trained classes.

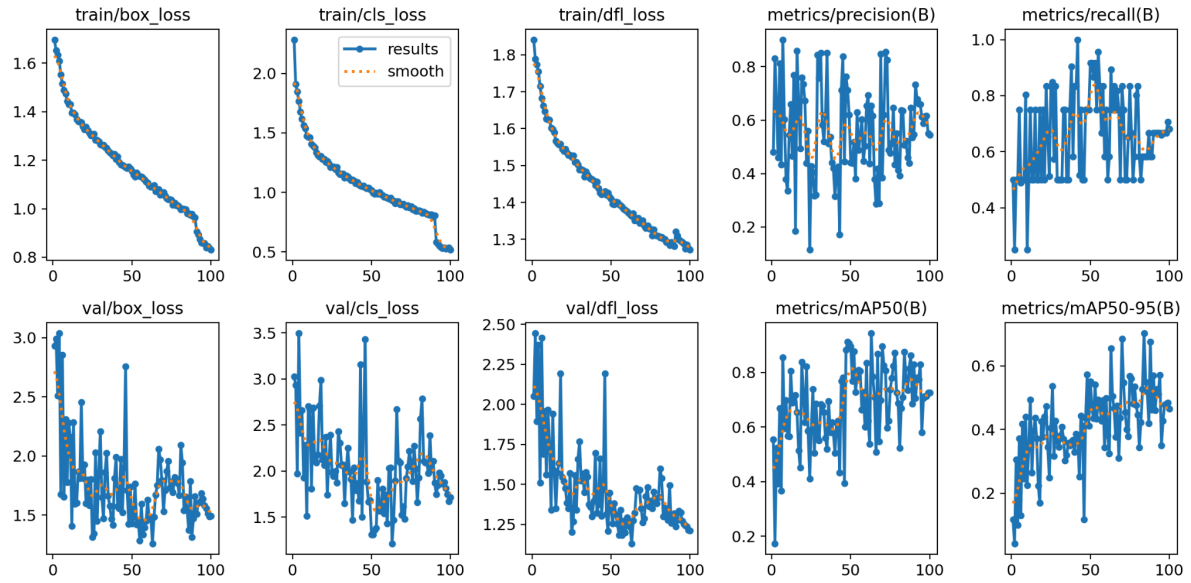

Figure 28. Model Training Loss Graph

To further validate the model, a test was conducted using a single image containing all trained classes. The detection results, presented in Figure 29, show that the system successfully recognized all objects correctly, except for one instance—the towel clip was detected as an iris. This misclassification is reasonable, given that the towel clip was not included in the dataset, and the model only had four trained classes. The similarity in shape between the two tools likely contributed to this result,

emphasizing the need for dataset expansion to improve accuracy and account for more surgical instruments.

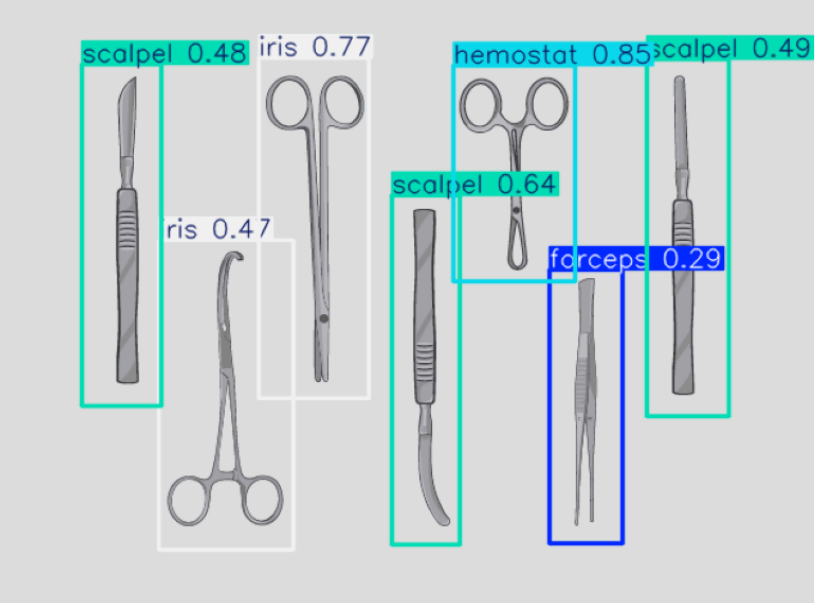

Figure 29. Model Detection Result

Overall, the model demonstrated strong real-world usability despite minor confidence fluctuations. Further refinements, including expanding the dataset and optimizing classification parameters, will enhance detection precision, ensuring even more reliable tool recognition within surgical settings.