Prototyping and Simulation of an Educational Manipulator Robot: An EEZYbotARM MK2 Revamping Case Study

João Alexandre Batista Coelho - a41195

Dissertation submitted to the School of Technology and Management of Bragança to obtain the Master Degree in Electrical and Computer Engineering

Scientific supervision:
Prof. Dr. José Alexandre de Carvalho Gonçalves
Prof. Dr. Paulo José Cerqueira Gomes da Costa

Bragança
2024

Prototyping and Simulation of an Educational Manipulator Robot: An EEZYbotARM MK2 Revamping Case Study

João Alexandre Batista Coelho - a41195

Dissertation submitted to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering

Scientific supervision:
Prof. Dr. José Alexandre de Carvalho Gonçalves
Prof. Dr. Paulo José Cerqueira Gomes da Costa

Bragança
2024

Dedication

(. 9o3x";6'3x& 5.r=o"9=3x !x"7-<r(.;+a

key: $\oplus$

Acknowledgment

The successful completion of this work would not have been possible without the invaluable support and contributions of numerous individuals. I wish to express my heartfelt gratitude to my family, professors, and colleagues for their guidance, availability, and patience throughout this journey.

I would like to extend my sincere appreciation to Professor José Gonçalves for his generosity in accepting me as a student, as well as for the technical and scientific guidance he provided, which has played a critical role in shaping my academic career. Additionally, I am deeply grateful to Professor Paulo Costa for his unwavering support during the development of the robot, his clarity in elucidating the theoretical aspects of control, and his amicable demeanor throughout my educational experience.

I would also like to acknowledge my fellow research colleagues who have provided invaluable support and warmth throughout my academic journey. Laiany Brancalião demonstrated remarkable patience and dedication, contributing to the hardware implementation of various components of the robot. Her thoughtful insights and encouragement greatly influenced the direction of this work and made the process more enjoyable. Similarly, Mariano Alvarez played a crucial role in my experience, offering both kindness and patience in the research office. His welcoming demeanor and assistance in integrating the robot structure into the SimTwo physics simulator were instrumental in my progress. I am truly grateful for their heartfelt support and their academic insights, which have greatly enriched my academic trajectory.

Finally, I would like to acknowledge my entire family—my parents, sister, and grandparents—who have provided steadfast support throughout my years of academic training. Their encouragement has been essential, and without it, none of this would have been possible.

Abstract

This dissertation delineates the prototyping, development, and integration of an educational robotic manipulator, specifically aimed at mitigating the financial and practical barriers that hinder hands-on robotics education. Robotics encompasses a multidisciplinary field requiring mechatronics, programming, control systems, and physics proficiency. Therefore, direct access to robotic hardware is critical for effective learning. Unfortunately, the prohibitive cost of industrial-grade robots restricts many educational institutions from offering students comprehensive, real-world training experiences.

To address this challenge, the work is focused on creating a cost-effective robotic manipulator platform for the Polytechnic Institute of Bragança, School of Technology and Management. This platform enables students to engage with the practical aspects of robotics without relying on high-cost industrial robots.

The project's objectives included the design of a custom robotic gripper, the implementation of kinematics and path-planning algorithms for the manipulator, the modeling and control of servomotors, and the integration of the robot within a physics simulation environment. Each component was developed, focusing on accessibility and adaptability, providing a realistic yet affordable alternative to industrial systems. This integrated robotic platform offers students valuable hands-on learning experiences in kinematic chains, motion control, and time restrain problem-solving, thereby effectively bridging the gap between theoretical knowledge and practical application in robotics.

Keywords: cost-effective robotics; educational robotics; hands-on learning; hardware-in-the-loop; kinematics; mechatronics education; motor control; path planning; physics simulation; robotic manipulator; SimTwo.

Resumo

Esta dissertação delineia a prototipagem, desenvolvimento e integração de um manipulador robótico educacional, direcionado especificamente para mitigar as barreiras financeiras e práticas que dificultam a educação prática em robótica. A robótica abrange uma área multidisciplinar que exige proficiência em mecatrónica, programação, sistemas de controlo e física, tornando o acesso direto ao hardware robótico essencial para uma aprendizagem eficaz. No entanto, o elevado custo dos robôs de nível industrial impede muitas instituições de ensino de oferecerem aos estudantes experiências de formação completas e baseadas em cenários reais.

Para enfrentar este desafio, o projeto concentrou-se na criação de uma plataforma de manipulador robótico económica para o Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão. Esta plataforma permite que os estudantes explorem os aspetos práticos da robótica, sem a necessidade de recorrer a robôs industriais de alto custo.

Os objetivos do projeto incluíram o design de uma garra robótica personalizada, a implementação de algoritmos de cinemática e planeamento de trajetórias para o manipulador, a modelação e controlo de servomotores e a integração do robô num ambiente de simulação física. Cada componente foi desenvolvido com foco na acessibilidade e adaptabilidade, proporcionando uma alternativa realista, mas acessível, aos sistemas industriais. Esta plataforma robótica integrada oferece aos estudantes experiências práticas valiosas em áreas como cadeias cinemáticas, controlo de movimento e resolução de problemas com restrição de tempo, aproximando de forma eficaz o conhecimento teórico da aplicação prática na área de robótica.

Palavras-chave: aprendizagem prática; cinemática; controlo de motores; educação em mecatrónica; hardware-in-the-loop; motor de física; planeamento de trajetórias; robótica custo-eficaz; robótica educacional; robôs manipuladores; SimTwo.

Contents

List of Tables

List of Figures

List of Acronyms

DC direct current.

DoF degree of freedom.

FPM first-principles model.

HIL hardware-in-the-loop.

ODE Open Dynamics Engine.

PRBS pseudorandom binary sequence.

ROS Robot Operating System.

UR Universal Robots.

WoS Web of Science.

Chapter 1

Introduction

This chapter provides insight into the terminology surrounding the study documented in this dissertation. The chapter is structured into a main section presenting the contextual background relevant to the developed study, the project's objectives to solve the problem, and the document's structure.

1.1 Problem Statement

Robotics is an interdisciplinary field encompassing mechatronics, signal processing, control, programming, and other physics and mathematical-related domains. These disciplines are commonly integrated within the electrical engineering course, and a standalone robotics class can be part of the course in some educational institutions, such as the Polytechnic Institute of Bragança, School of Technology and Management, where this research takes place. Integrating robots in classrooms has become a powerful tool for motivating and supporting students' learning, facilitating engaging activities, and sparking curiosity [1], [2]. Researchers and educators have widely applied this approach to enhance learning, captivate students' attention, and promote skills development, ultimately leading to academic success [2], [3]. However, integrating robotics into school curricula faces financial barriers. Industrial-grade robots predominantly characterize the market, which, despite their high level of sophistication, are prohibitively expensive for most educational institutions. Even robots designed for educational purposes often carry high price tags, posing challenges for academic schools to acquire an adequate number of units for classroom deployment.

Universal Robots (UR) has made significant efforts to bring robotics education to a broader audience through its UR Academy. The UR Academy offers free online training modules designed to teach basic programming skills and applications specific to their collaborative robots (cobots). These courses cover robot setup, safety protocols, and simple programming tasks. However, the UR Academy emphasizes online content without working directly with physical robots. While UR also offers hardware bundles aimed at educational institutions, the cost of acquiring a single UR robot may exceed the budget of schools and universities [4].

Similarly, KUKA, one of the leading industrial robot manufacturers, provides educational support through its KUKA College, which offers a wide range of e-learning courses and certification programs. KUKA also provides virtual robotic simulators, such as KUKA Sim, allowing students to visualize and simulate robotic applications without needing a physical robot. These tools are valuable for learning the theory in the robotics field and enable students to design, simulate, and optimize robotic systems in a virtual environment. However, as with Universal Robots, the cost of KUKA's industrial robotic arms can be prohibitive for institutions. As a result, universities generally may not provide students with direct access to KUKA robots for hands-on learning [5].

Another possible solution for academic learning applications is the Dobot "Magician", a small-scale 4-axis desktop manipulator robot. Dobot, a company specialized in robotic solutions for industrial, commercial, and educational applications, designed the "Magician" specifically for educational use. Capable of performing tasks such as 3D printing, laser engraving, and even calligraphy, this robot is aimed at students to explore various topics, including robotic systems, robot movement control, programming, and other related subjects. With a reference price of €1500, although more affordable than many industrial robots, it still represents a somewhat costly investment for robotics classes. Figure 1.1 illustrates the Dobot "Magician" robot [6].

The lack of physical access to robots in robotic learning curricula increases the gap between theoretical learning and real-world applications. Robotics hands-on experience is not merely supplementary—it is essential. Students learn to integrate sensors, actuators, and controllers in real-world settings and gain problem-solving skills that are difficult to replicate in purely virtual environments [7]. For instance, troubleshooting sensor malfunctions, calibrating robotic arms, and dealing with real-time delays or inaccuracies in motion are experiences that require direct interaction with robotic hardware. As noted

Figure 1.1: Picture of the Dobot "Magician" robot.

in various studies on engineering education, hands-on learning significantly enhances students' understanding of complex systems, fostering the development of critical skills in problem-solving and innovation. While companies like Universal Robots and KUKA have made strides in providing educational resources, the high cost of physical robots remains a significant hurdle for universities, particularly those with limited budgets. The price of a single industrial robot arm can range from €25,000 to over €100,000, depending on its capabilities and configurations. Adding the costs of maintenance, safety equipment, and ancillary hardware further drives up the expense. For many institutions, this is simply out of reach, forcing them to rely on virtual learning environments, which, while helpful, cannot fully replicate the experience of working with an actual robot in a laboratory setup. The gap lies in this disconnect between affordable educational and expensive industrial robots. As a result, institutions must weigh the balance between cost-effective virtual training and the crucial benefits of hands-on experience. This financial barrier makes it difficult for educational programmes to provide comprehensive robotics training, especially when compared to more affordable but less capable systems, like LEGO Mindstorms or VEX Robotics, which serve entry-level purposes but need more sophistication for advanced engineering education.

During the second semester of the 2023-2024 academic year at the Polytechnic Institute of Bragança, School of Technology and Management, a robotics course featured a practical project focused on programming a mobile robot. In preparation for this course, a group of researchers from the Research Centre in Digitalization and Intelligent Robotics

(CeDRI), of which the author of this dissertation was also part, manufactured four identical mobile robots, culminating in the publication of two research papers in the CoDIT 2024 and TEEM 2024 conferences [8], [9].

The robot prototypes have differential kinematics and were specifically designed for line following and obstacle avoidance. Figure 1.2 illustrates the robots developed and utilized in the robotics course. The robots were assigned to student teams to be programmed to track a designated path delineated by a black line and effectively navigate obstacles. As part of the project assessment, students competed to develop the most efficient program capable of completing an unspecified trajectory in the shortest possible time.

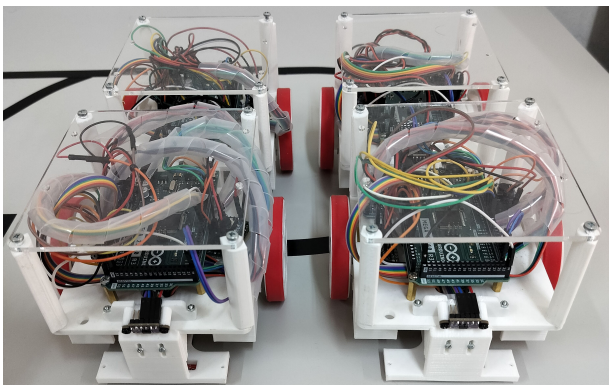


Figure 1.2: Assembled mobile robots built for the robotics class.

The successful integration of the aforementioned mobile robots in the robotics class has further strengthened the focus of this project, which aims to develop a manipulator robot also for educational purposes.

Following what was initially mentioned, the prohibitive cost of acquiring industrial robots is not viable for their use in education. However, students must be exposed to challenges that involve their programming and allow recognition of the challenges associated with controlling and modeling different kinematic chains. In this context, this work describes the work done to develop an integrated platform, which involves not only hardware but also a computational physics engine, to align the learning experience with what could be experienced, in practice, using industrial manipulators. In line with this motivation, the following work objectives were outlined:

- Design and development of a custom robotic gripper;
- Implementation of the robot's kinematics;

- Development of path planning algorithms;
- Modeling of the servo motors;
- Design and implementation of a custom controller for the servo motors;
- Integration of the robot's model within a physics simulator.

The structure of this document is as follows: Chapter 2 presents the theoretical framework supporting the current work. Chapter 3 details the robot prototyping process, including its architecture, structure, the determination of the direct kinematics of the robot, and the mathematical modeling of the direct current (DC) motors used in the servo system. Chapter 4 focuses on the robot's control system, covering the design of the closed-loop DC motor controller, developing the inverse kinematics, and implementing path planning algorithms. Chapter 5 discusses the simulation of the robot, explicitly integrating the robot's body model into the SimTwo physics simulator. Finally, Chapter 6 concludes with a summary of the findings and future directions for research.

Chapter 2

Theoretical Framework

This chapter is devoted to analyzing the state of the art regarding the development of robotic manipulators for educational purposes. The research was carried out using the "Web of Science" database using a set of keywords, detailed below, which are considered representative of the scope of this work. At the end of this chapter, the main conclusions are presented regarding academic works published in the literature and other resources found on the GitHub platform.

2.1 Bibliographic Review

One of the initial phases of this work, and one of the most relevant, consisted of investigating the scientific activity that has been taking place in the field of robotics in education. In particular, the research directions have been taken to bring the experience of using robots in an educational context closer to their use in an industrial environment. To this aim, a search was conducted in the Web of Science database (WoS), where a collection of refereed scientific articles aligned with this dissertation topic was made. The query process was filtered through the parametrization of the search engine by defining a time span that includes the last five years (from 2019 to 2024) and considering "robotic manipulator" and "education" as keywords.

The search results returned thirty-seven publications, of which fifteen were open-access, and six were review articles. The graph in Figure 2.1 illustrates the distribution of these articles by the scientific area.

The acronyms shown in the graph correspond to the following scientific areas:

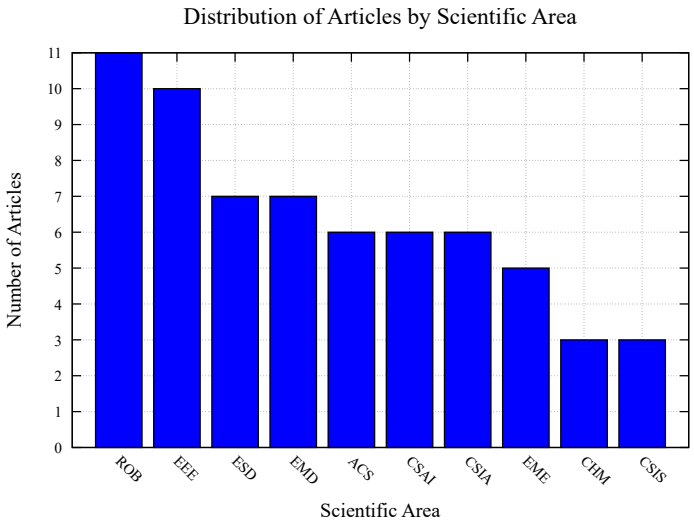


Figure 2.1: Distribution of the WoS search results in articles by scientific areas.

- **ROB**: Robotics;
- **EEE**: Engineering Electrical and Electronic;
- **ESD**: Education Scientific Disciplines;
- **EMD**: Engineering Multidisciplinary;
- **ACS**: Automation Control Systems;
- **CSAI**: Computer Science Artificial Intelligence;
- **CSIA**: Computer Science Interdisciplinary Applications;
- **EME**: Engineering Mechanical;
- **CHM**: Chemistry Multidisciplinary;
- **CSIS**: Computer Science Information Systems.

Certain scientific areas identified by the WoS platform, such as "Environmental Studies," were deemed irrelevant to the present study; hence, those papers were excluded from the review process. This current state-of-the-art review concentrates on five primary areas: "Robotics," "Electrical and Electronic Engineering," "Educational Scientific Disciplines," "Multidisciplinary Engineering," and "Automation Control Systems."

From these designated areas, a total of eight papers were identified as directly related to the current research endeavor.

In this reference frame, [10] describes a 6 degree of freedom (DoF) anthropomorphic robot designed and built using additive manufacturing processes such as 3D printing. The objective was to use the robot for the undergraduate level to promote the teaching of mathematical modeling and Robot Operating System (ROS) programming. According to the authors, this approach positively impacted the technical performance of the participants by stimulating learning and understanding.

A DeltaZ, open-source and open-hardware type of robot was proposed by [11] to promote an accessible robot platform that could be used for research and education. The device was designed to be 3D printed, assembled, and employed to perform translational movements and tasks. The authors highlight the repeatability of their solution and suggest that their platform be used as a benchmark tool in the future.

Still, in the context of using low-cost robots to promote teaching, [12] describes a project-based teaching approach where students are motivated to develop their robotic manipulator. The challenge is posed in a competition where a set of tests exists that must be overcome.

In [13], a robotics education kit is described. This kit includes a set of 3D parts to be assembled and used as a tool for an online/hybrid course in teleoperated robots. Once again, the teaching methodology is based on a challenge that students must overcome. The students are divided into teams, and the team designs and fabricates (3D printing and assembly) and implements a control strategy for a gripper to be coupled to the educational kit.

Trajectory planning and validation of an open-source delta robot was also described by [14]. Besides the hardware, the presented solution includes a software package incorporating a velocity profile planner and an inverse and forward kinematics solver. The 3D printed parts were obtained from an open-source "Thingverse" project. According to the authors, the proposed hardware and motion planning software can be utilized in different contexts, such as education.

In [15] took a more complex approach where a 7-DoF robotic arm tracking system was devised. As stated by the authors, the robot can be employed in education and research activities, and all the resources, such as CAD files and software, are publicly available online and free to use.

Also, for STEM education, the work of [16] describes a robotic manipulator arm with real-time control algorithms and a human-robot interface. The work also explores

using the robotic manipulator as a teaching tool for robotics and STEM subjects in developing country schools.

Finally, in [17], the authors describe the implementation strategy for teaching Cyber-Physical Systems to graduate high-level students. The approach included theoretical lectures and hands-on practices using off-the-shelf mobile robots and ROS as the software platform. The paper highlights that students were expected to work in teams and that this approach promoted research skills development and improved programming abilities and understanding of manipulator arm robotics.

2.2 Other Public Resources

This section focuses on the open-source robotic manipulators that are available online, open-source, and suitable for educational purposes. The search was done in the "GitHub" platform, and without following any particular order, the first repository was the one of Alex Thiel's. The project is designated by "uArm", and the author provides a set of Python files that enables computer vision to interact with robot arms. Moreover, the plans for a 4-DoF robotic arm, supported by an Arduino platform, are also provided. This repository can be found by following the URL `https://github.com/uArm-Developer/uArmCreatorStudio`

Another robot manipulator can be found at `https://github.com/ROBOTIS-GIT/open_manipulator`. This project, coined by the name of "OpenManipulator", is a 6-DoF manipulator that is ROS-compatible. Besides a set of libraries and CAD files, the authors provide documentation and a wiki page.

Like "uArm", "Dobot" is also a 4-DoF robotic arm with ROS and Arduino support. This project can be found at `https://github.com/Dobot-Arm/`, and it is a trendy robot used in both research and education.

Despite being a commercial product, the "MeArm" project can also be freely downloaded from a GitHub repository. Once again, we are discussing a 4-DoF arm supported by an Arduino platform. Firmware and laser cutting files are available, and many other resources can be found from `https://github.com/phenoptix/MeArm`.

As a 6-DoF robot, "Thor" is a 3D-printable open-source solution that includes complete CAD files and features ROS integration. From `https://github.com/AngelLM/Thor`, the user can download the STL files for 3D printing. Those files, in addition to

gerber and schematics, can be found at http://thor.angel-lm.com/.

Another 6-DoF robot arm is the "Faze4". According to the author, "Faze4" is a compact, fully 3D-printable, open-source 6-axis robotic arm. It has a resemblance to industrial robotic arms, and it is intended for use in research and educational applications. All the files and assembly instructions can be found at https://github.com/PCrnjak/Faze4-Robotic-arm

In the GitHub repository of Alexander Kock, it is possible to find another 4-DoF robotic manipulator that is entirely open-source. The files, available from https://github.com/AlexanderKoch-Koch/low_cost_robot/tree/main, include the STL files and a set of Python scripts. The motion of the arm does not require a microcontroller and is handled by a serial bus servo driver board.

"PAROL6" is also a 3D-printed desktop robotic arm designed to emulate industrial robots in mechanical design, control software, and usability. Its control software, GUI, and STL files are fully open-source and available at https://github.com/PCrnjak/PAROL6-Desktop-robot-arm/tree/main.

Finally, in the same line, the "Koch v1.1" is advertised as a low-cost robotic arm, and all the required files are made available at https://github.com/jess-moss/koch-v1-1.

2.3 Conclusion

In this section, a review of open-source solutions, both hardware and software, aimed at building robotic manipulators for educational purposes was made. The research was carried out along two distinct axis: on the one hand, work carried out within an academic context and, on the other, projects made available by their authors through the GitHub platform. Regarding the former, a search was conducted on the WoS search engine for scientific works. From a set of keywords and considering a time window concerning the last five years, a total of eight of the most relevant articles were selected. From the analysis of those papers, it can be concluded that, in general, additive manufacturing and 3D printing techniques were used to create the mechanical components of the robotic platform. Many of these approaches involved encouraging students to learn by integrating robots into playful activities and challenges, such as competitions. Some of the reviewed works also include some type of high-level software to interact with the robot.

The search carried out in the GitHub platform was, by no means, exhaustive. Even if many other examples can be found, the ones enumerated in the previous section are the most relevant and highly cited. The common denominator in all the nine examples cited previously relies on providing a set of files that allow the user to replicate a given robotic manipulator in terms of hardware and control software. However, none of the solutions presented an integrated co-simulation proposal between the physical device and its digital twin. In the current work, this line of research was followed by developing a prototype that could operate in a broader context and where closed-loop control would be formalized in two distinct domains: in the physical, through a printed manipulator using 3D technology and the entire actuation and sensing infrastructure, and digitally, through its representation using an open source physics engine. The following chapters present all details of the tasks carried out to achieve this objective.

Chapter 3

Robotic Manipulator Prototyping

This chapter provides an in-depth examination of the architecture and structural design of the robot that was developed. The robot has 3-DoF revolute joints, and it incorporates a design that ensures that the gripper remains parallel to the robot's base. Such a design serves as a good introductory model for students engaging with manipulator robots, as it exemplifies a conventional pick-and-place manipulator without the additional complexities associated with a higher number of DoF.

Moreover, this chapter explores the calculations associated with direct kinematics, which are critical for determining the position of the end effector based on the joint angles. These calculations are integral to the effectiveness of path planning algorithms. This study is presented in a recently published paper presented at the CoDIT conference [18].

This chapter also addresses the mathematical modeling of the DC motors applied in the robot's servo motors. Due to the inherent characteristics of these servo motors, enhancements to their control mechanisms were imperative. The initial controllers exhibited deficiencies in effective velocity management, necessitating the implementation of alternative strategies. Specifically, time pauses between designated points in the planned trajectory were employed to sustain the desired velocity. Consequently, to facilitate the implementation of a more robust controller, as well as to establish a simulation of the robot that possesses analogous characteristics to the physical model, the mathematical modeling of the motors was deemed essential.

3.1 Architecture and Structure

The robot described in this dissertation was built upon the open-source design of Carlo Franciscone's "EEZYbotARM MK2" [19]. This design incorporates a 3-DoF revolute robot arm characterized by a significant feature: the gripper maintains a parallel orientation to the robot's base. This feature is achieved through the mechanical configuration, which allows for the gripper to rotate freely in accordance with the angles of the elbow and shoulder joints. As a result, this design enhances the robot's effectiveness in pick-and-place applications without necessitating extra DoF for the gripper to remain parallel to the base. The choice of a 3-DoF design simplifies control by focusing on two axles plus the yaw rotation, which is sufficient for basic pick-and-place tasks. While a fourth DoF could provide added flexibility, this design minimizes computational and mechanical complexity, making it suitable for educational purposes.

Carlo Franciscone developed the design of his robot based on the ABB IRB 460 model. This robot features a kinematic linkage similar to that of the ABB IRB 460, scaled down by a factor of 1:7. A notable distinction between the ABB robot and Franciscone's design is the inclusion of a fourth DoF in the gripper, which includes a yaw motion allowing for the rotation of objects. According to ABB, the IRB 460 is recognized as the leading palletizing robot globally, ideal for end-of-line and bag palletizing applications [20]. This 4-axis robot not only ranks as the fastest in its category but also possesses a compact footprint, enabling seamless integration into existing packing lines. Figure 3.1 illustrates the ABB IRB 460 model.

To construct the prototype, 3D printing technology was employed. This approach allowed for a rapid assembly process, enabling quick identification and correction of design flaws. Reprinting and replacing individual parts made it easy to iterate on the design, allowing the cycle of testing, refining, and reprinting to continue efficiently until the final result met the desired specifications. This process made replicating and improving the manipulators both simple and cost-effective. Figure 3.2 visually captures the step-by-step assembly stages of the robot body.

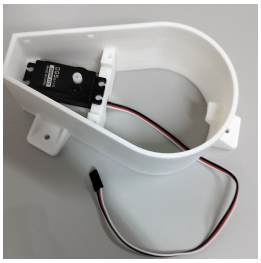
Due to the specific requirements of the prototyped robot, a custom gripper was developed to house an electromagnet. The objective of this robot is to facilitate the picking and placement of ferromagnetic components using the electromagnet. To accomplish this, a new gripper was designed with a hollow compartment to accommodate the electromagnet. Figure 3.3 illustrates the printed gripper, while Figure 3.4 presents the CAD

Figure 3.1: The ABB IRB 460 high-speed palletizing robot, known for its compact design and efficiency in end-of-line applications [21].

design of the implemented robot arm coupled with the customized gripper.

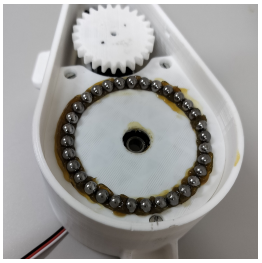
Central to the robot's operation is a Wemos D1 R32 board, which incorporates the ESP32 processor and serves as the primary controller for the robot. The Wemos D1 R32 microcontroller was selected given that the main interest of this robot is to emulate the functioning of an industrial robot with the aim for students to understand, at a low level, the operation of manipulator industrial robots, the processing of the microcontroller of this robot must handle precise and fast calculations; hence the microcontroller is adequate for this application. Firstly, its processing enables the rapid calculation of kinematics and movements required for the robot, ensuring swift transmission of control signals to the servo motors. Additionally, being a 32-bit processor, it provides suitable accuracy for trigonometric functions, which are crucial for precise kinematic calculations. Moreover, the board's low power consumption requirements simplify its power supply, requiring only a 5 V source for operation [22].

Furthermore, the robot is equipped with a "Grove Switch Electromagnet v1.1" electromagnet, enabling it to function as an actuator for pick-and-place operations. The microcontroller receives power from an external power supply, which allows the robot to operate independently of a micro USB connection. Additionally, a custom shield was

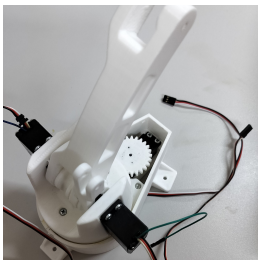


(a) Attaching the servo motor to the base support, enabling base rotation.

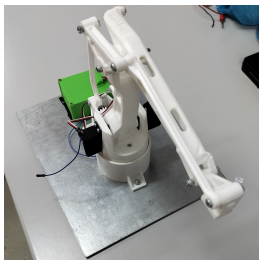
(b) Installing the main gear interacting with the base motor to drive rotation.



(c) Adding the lower gear, spheres, and pulley to enable smooth base rotation.

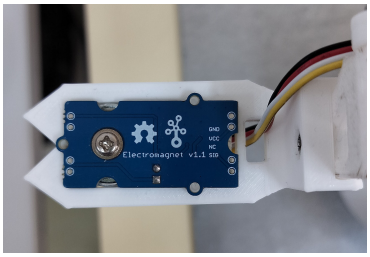


(d) Mounting the shoulder and elbow servo motors, along with the shoulder arm.

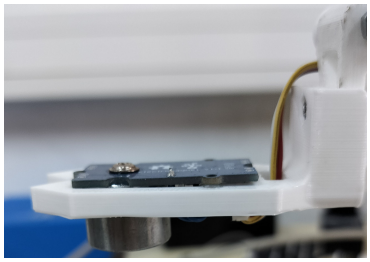


(e) Final body assembly, completing the elbow arm.

Figure 3.2: Step-by-step assembly stages of the body of the manipulator robot.



(a) Top view of the gripper.

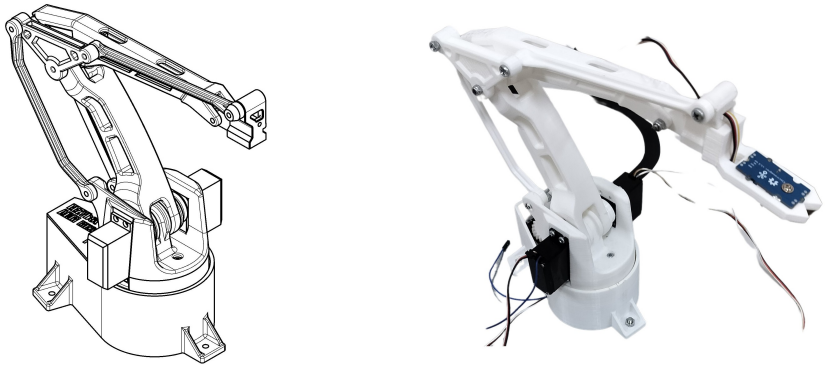


(b) Side view of the gripper.

Figure 3.3: Design of the customized gripper.

developed to streamline the connection of the power supply to the microcontroller, along with facilitating the attachment of the servo motors and electromagnet to the microcontroller ports, thus eliminating the problem of de-organized wiring with headers. Figure 3.5 illustrates the schematic of the shield electric circuit designed in EasyEDA.

The robot is also powered by three 5 V servo motors, each one managing one of the DoFs integral to the design. These servos possess a range of 180 degrees. Control signals for the servo motors, operating at 3.3 V, were generated with PWM-capable digital ports [23]. A PWM port also ordered the electromagnet's control signal. Refer to



(a) 3D model. (b) Physical prototype.

Figure 3.4: Technical 3D rendering and physical implementation of the "EEZYbotARM MK2" manipulator robot.

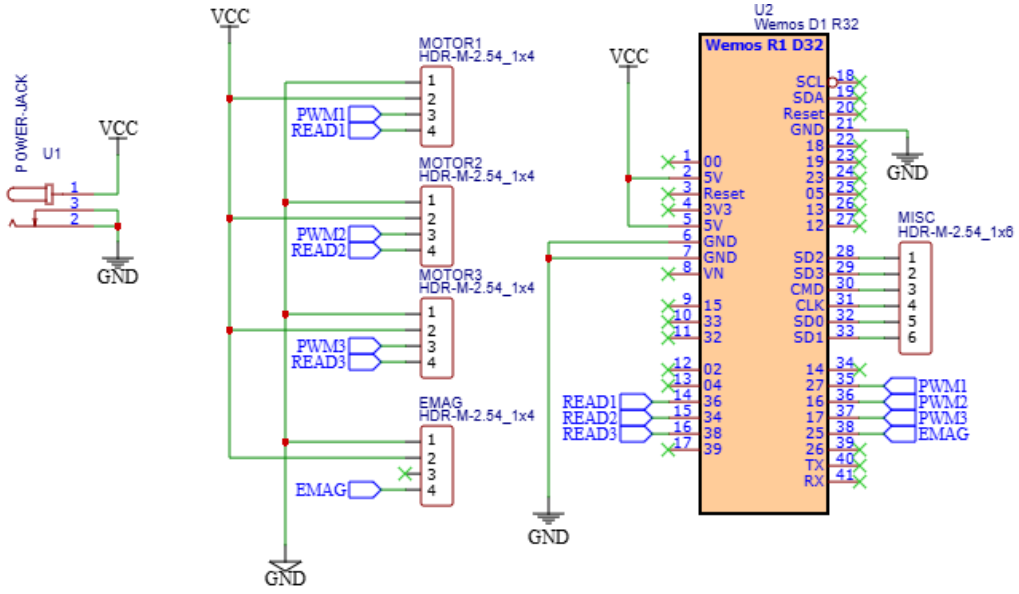


Figure 3.5: Shield schematic for the Wemos D1 R32 microcontroller that allows easy connection to the servo motors, electromagnet, and power supply.

Figure 3.6 for a visual representation of the system's connections without using the shield. Yellow wires denote the control signals; red wires indicate the 5 V power supply; and black wires represent the ground connection.

The initial version of the servo motor proved useful for the first working implementation of the robot; however, a more responsive and specialized servo motor was required to enhance the controller for path planning capabilities and to facilitate the implementation of hardware-in-the-loop (HIL) functionality, thereby allowing users to control the robot

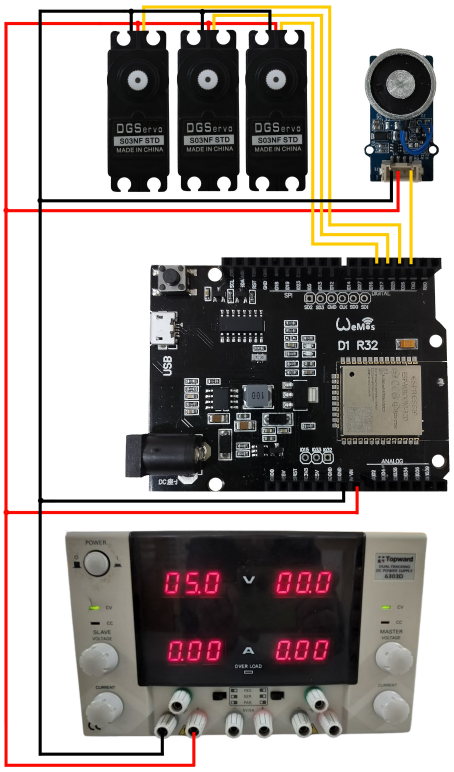


Figure 3.6: Connection diagram of the robot arm.

via a simulator. The initial controller faced challenges in maintaining a consistent velocity, leading to jerky movements and inefficiencies. Modeling the motor dynamics allowed for finer control over speed adjustments, resulting in smoother operation. Consequently, the initial step in the development of the motor controller, as well as the integration of the motor into the simulator, involved the modeling of the DC motor applied in the servo. Further details on this topic can be found in Section 3.3.

3.2 Robot's Direct Kinematics

As mentioned, the robot developed is designed with three revolute joints, which can move and be controlled along three axes: x, y, and z. This arrangement allows for versatile movement and precise positioning in a three-dimensional space. For a comprehensive understanding of its design and the configuration of its joints, please refer to the detailed illustration presented in Figure 3.7. This figure highlights the robot's structural components and how each joint contributes to its overall functionality.

Upon reviewing Figure 3.7, it is evident that the robot is equipped with two distinct

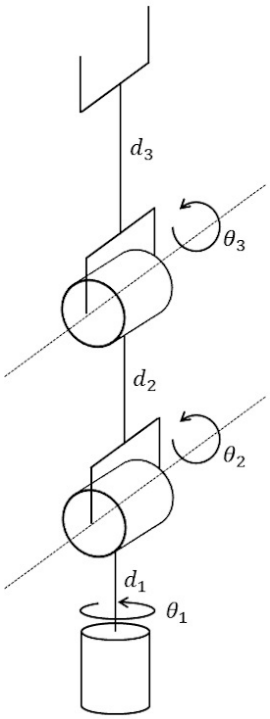


Figure 3.7: Kinematics diagram of the 3-DoF revolute robot.

types of revolute joints. The initial joint, referred to as the hip joint, facilitates yaw movement, which involves rotation around the z axis. In addition, the second joint, known as the shoulder joint, along with the third joint, termed the elbow joint, enables roll movement or rotation around the y axis.

A transformation matrix is employed to achieve a comprehensive understanding of the robot's kinematic structure. This matrix presents a mathematical representation of the robot's movements and the interrelationships among its various components. The initial step in deriving the transformation matrix involves assigning a reference frame to each robot joint. A reference frame serves as a coordinate system that defines the position and orientation of a joint in relation to its predecessor. By establishing these frames, it becomes possible to accurately map the impact of each joint's movement on those following it. Each reference frame is meticulously connected to the preceding frame through rotation and translation parameters, which are fundamental to the calculations. This relationship is essential, as it permits the computation of how the end effector—responsible for executing tasks—relates to the robot's base, which provides overall stability and support. To facilitate the visualization of this process, Figure 3.8 illustrates the indexing of these reference frames within the robot's kinematic diagram, effectively emphasizing how

each frame is interconnected and contributes to the overall motion of the robotic system.

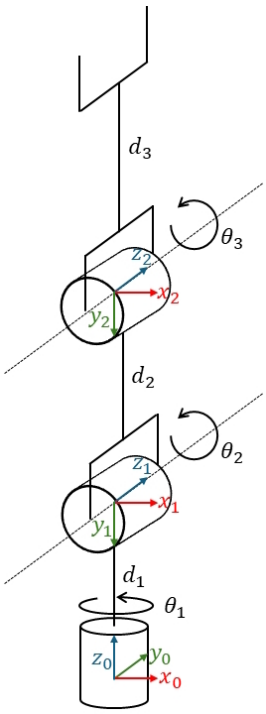


Figure 3.8: Indexation of the frames for each joint.

With the frame indexing established for each joint, construction of the Denavit-Hartenberg table becomes feasible, incorporating parameters for each joint as per the standard Denavit-Hartenberg method, as illustrated on Table 3.1.

Table 3.1: Parameterization of the robot according to the standard Denavit-Hartenberg method.

i	$\boldsymbol{\alpha}$	**a**	**d**	$\boldsymbol{\theta}$
1	90^{o}	0	d_1	θ_1
2	0^{o}	d_2	0	θ_2
3	0^{o}	d_3	0	θ_3

The computation of the transformation matrices for the relationship between each frame becomes possible. Each transformation matrix is a 4×4 matrix comprising a 3×3 rotation matrix around the x and z axes, along with a 3×1 translation matrix along the same axes. Following the determined parameters, the general transformation matrix calculation proceeds in equations (3.1, 3.2, 3.3).

$$\mathbf{T_i^{i-1}} = \mathbf{Rot_z} \times \mathbf{Trans_z} \times \mathbf{Trans_x} \times \mathbf{Rot_x} \tag{3.1}$$

$$\mathbf{T_i^{i-1}} = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2}$$

$$\mathbf{T_i^{i-1}} = \left[\begin{array}{ccc|c} cos(\theta) & -cos(\alpha)sin(\theta) & sin(\alpha)sin(\theta) & acos(\theta) \\ sin(\theta) & cos(\alpha)cos(\theta) & -sin(\alpha)cos(\theta) & asin(\theta) \\ 0 & sin(\alpha) & cos(\alpha) & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.3}$$

The calculation of the transformation matrix was applied sequentially to establish the relationship between each frame. Following these individual calculations, the transformation matrix between the end effector and the robot's base was computed according to the equation (3.4).

$$\mathbf{T_3^0} = \mathbf{T_1^0} \times \mathbf{T_2^1} \times \mathbf{T_3^2} \tag{3.4}$$

3.3 Mathematical Modeling of the Actuator

The development of a mathematical model to accurately represent the DC motor within the servo system was identified as essential for multiple reasons. Primarily, one of the foremost motivations for modeling the DC motor was to facilitate the design of an enhanced control system, surpassing the capabilities of the existing controller. The previous control system exhibited significant limitations, particularly regarding its inability to regulate the motor's velocity. This lack of velocity control is critical for effective path planning and trajectory execution in robotic applications, where precision in movement is paramount. Users are often required to specify a customized velocity for certain actions within the robot's operational routines, thereby making this feature a crucial component of all robotic systems.

Furthermore, a substantial portion of this project concentrated on integrating a robot model into a physics-based simulator. This integration allows for the offline programming of the robot, enabling students to test and refine their algorithms without the risks associated with direct interaction with a physical robotic system. The method,

referred to as HIL control, is particularly advantageous for educational purposes, as it provides students the opportunity to learn how to operate robots safely and efficiently, thereby minimizing the potential for physical damage, resulting from untested programs. For an effective HIL implementation, it is imperative that the simulator accurately replicates the behavior of the DC motor—further underscoring the necessity of developing a comprehensive mathematical model of the motor and extracting the requisite parameters for the simulator.

During the data acquisition phase for the DC motor, a diverse array of sensors was employed to measure both angular velocity and angle. Among these was an infrared sensor combined with a custom-engineered indented wheel. This configuration was applied to detect the dynamic velocity measurement each time the infrared sensor detected an object near the marked strip on the wheel. The operation of this sensor was designed to emulate that of a conventional encoder, with its effectiveness assessed to obtain the necessary parameters accurately. However, a significant challenge emerged: the sensor was unable to capture the transient response of the motor. This transient behavior is vital for understanding the motor's response during rapid changes in input, and as such, this sensor was not viable for the data acquisition of the motor. Figure 3.9 illustrates the implemented acquisition structure utilizing the infrared sensor.

Figure 3.9: Custom-designed 3D-printed support structure for mounting the infrared sensor and servo motor.

In addition to the infrared sensor, the integrated potentiometer within the servo motor was employed as an alternative method for measuring both motor velocity and angle. To access the analog signal corresponding to the potentiometer's angular position, a wire was soldered to its wiper pin. This modification enabled direct measurement of the

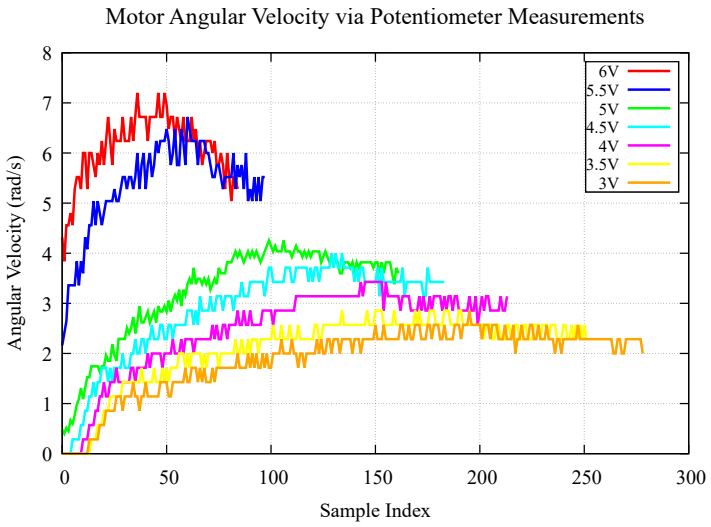
potentiometer's output. Figure 3.10 illustrates a servo motor with the potentiometer's wiper connected via a soldered wire.

Figure 3.10: Adapted servo motor for absolute position sensing using the potentiometer.

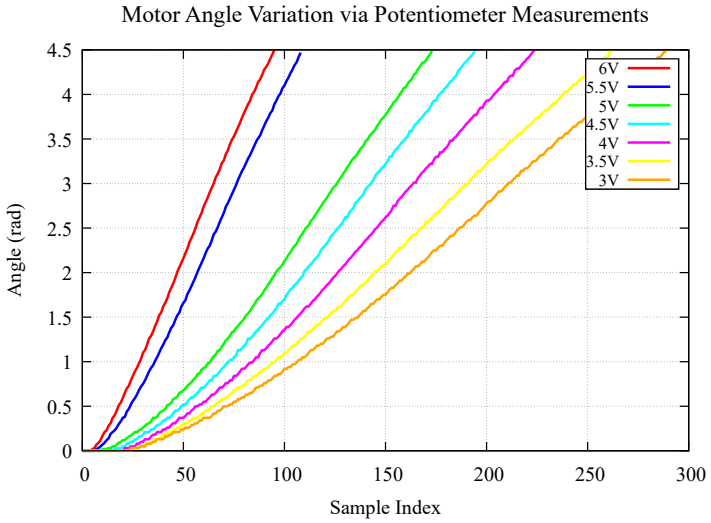
Throughout the parameter acquisition process, it became evident that the data collected from the potentiometer often exhibited considerable noise. This noise posed challenges, necessitating substantial post-processing to cleanse the resultant signals. This noisy reading also would result in the controller facing difficulties in processing this noisy data in real-time to accurately ascertain the motor's velocity and angle. Figure 3.11 illustrates the acquisition data utilizing the potentiometer.

One potential solution to mitigate this issue was the implementation of an exponential filter. Although this approach aided in reducing noise, it inadvertently introduced a delay in the motor's dynamic behavior, which was absent in the actual motor operation. This delay resulted in an inaccurate representation of the motor's real-time performance and complicated the overall findings. Figure 3.12 illustrates a block diagram representative of the filter introduced in the z-domain as well as the bode diagram of the filter.

After further experimentation, an encoder "E6A2-CW5C" was finally used to measure the velocity and angle of the motor. In preparation for the experiments, the servo motor underwent dismantling, with the subsequent removal of its controller and potentiometer. The motor's gears remained unaltered as the encoder was connected to the output shaft. For the connection of the encoder to the motor, a custom-designed support was created using 3D printing technology. This support holds the servo motor and the encoder in a fixed position, safeguarding them against external forces and minimizing potential disturbances during the data acquisition process. In addition to securing both the encoder and the motor, a fitting was devised to connect the motor's gear output shaft to the encoder shaft. This component, made of PLA and incorporating rubber within its structure, provides the necessary flexibility to compensate for any misalignment between the shafts and, for this reason, reduce friction. Figure 3.13 depicts the configuration of the data acquisition structure used to capture the encoder's output.



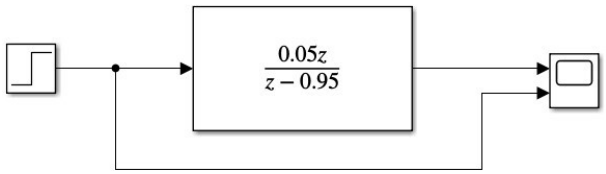
(a) Angular velocity of the motor after applying a moving average filter.



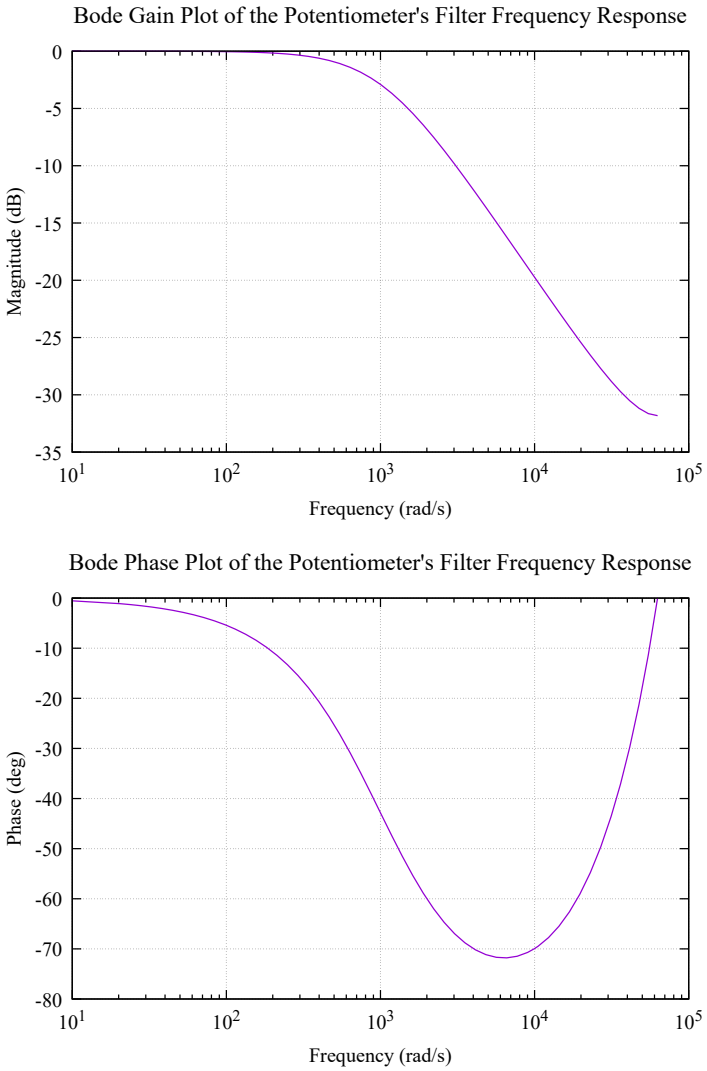
(b) Angle of the motor.

Figure 3.11: Graph representative of the acquisition of the angular velocity (utilizing a moving average filter) and the angle of the motor by utilizing a potentiometer as the measuring sensor.

A voltage-controlling driver was employed to analyze the motor's operational characteristics within both transient and steady-state conditions. In addition to that, an "INA219" current sensor was utilized to measure the current fed to the motor. This current measurement is essential for calculating key parameters in the motor's mathematical model, providing empirical data to ensure system identification. Finally, an



(a) Simulink block diagram of the exponential filter.



(b) Bode diagram showing the filter's frequency response.

Figure 3.12: Illustration of the filter implementation: (a) Simulink diagram of the exponential filter block used to minimize potentiometer noise on the microcontroller, and (b) Bode diagram representing the filter's frequency response, highlighting its effectiveness in noise reduction.

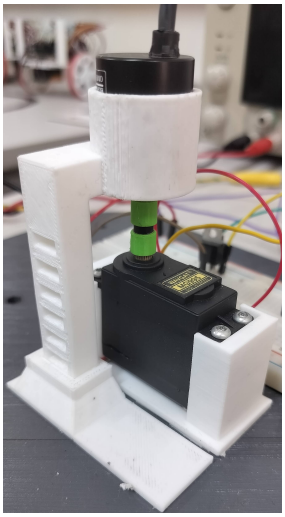


Figure 3.13: Custom-designed 3D-printed support structure for mounting the encoder and servo motor.

Arduino Mega 2560 was the microcontroller that captured sensor data and handled the driver. This methodology involves using external interrupt-driven procedures concerning the timestamp of the preceding interrupt and a counter that progresses in alignment with the signal response from both operational phases. This framework facilitates the determination of the velocity's magnitude and rotation direction.

In order to calculate the velocity, the dual-phase signals of the encoder (A and B) are applied to determine the direction of the rotation. Specifically, a change in signal A leads to an adjustment in the counter, either upward or downward, depending on the status of signal B, and vice versa. The phase lead or lag between signals A and B is used to compute the rotation direction.

The time interval (Δt), in microseconds, between consecutive state changes for both phases, was determined by calculating the difference between the last recorded external interruption time and the last sampled time. This time difference was then converted to seconds. The motor's rotational velocity (ω_r), in radians per second, was derived from the encoder counts and the number of encoder pulses per revolution. The encoder resolution (N) was adjusted to 400 due to the implementation of two phases, with the counter (C) updated for both rising and falling edges. The velocity was computed using the provided equation (3.5).

$$\omega_r = \begin{cases} \frac{(C \times 2\pi)}{\mathrm{N} \times \Delta t} & \text{if } \Delta t > 0 \\ 0 & \text{if } \Delta t = 0 \end{cases} \tag{3.5}$$

Mathematical models are essential for simulation and controller design since they provide a precise and systematic representation of complex systems that can be used to analyze and predict system behavior under various conditions. They must be able to describe the dynamic relationship between the system's input and output signals according to some internal state space [24]. Moreover, having a mathematical model of the system can lead to a more systematic controller design. In practice, mathematical models can be obtained by many different methods. Some, derived from the first principle, are based on the physical laws that govern the underlying system's dynamic behavior. Other, derived only from observed data, lead to black-box type of models. The latter ignores any internal mechanism, and both model structure and parameters are derived from data analysis methods.

In this work, a mathematical model must be derived to include the DC motor dynamics in the SimTwo software. Moreover, due to the format required by the software, the model must be based on physical principles. In particular, the electrical motor must be described by the electromechanical lumped parameter model presented in Figure 3.14.

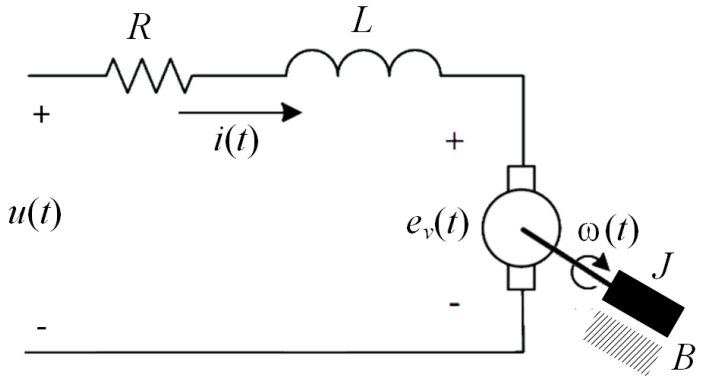


Figure 3.14: Lumped parameter model for a permanent magnet DC motor.

As can be seen, the electrical behavior is captured by the series of three lumped elements. The first is a resistance with value R, the second is an inductor with inductance L, and finally, the third is a voltage source that represents the counter-electromotive force

generated by the movement of the rotor. On the mechanical side, the diagram includes the rotor's inertia moment, J, and a friction coefficient, B.

From Kirchhoff's voltage law, the relation between the input voltage, $u(t)$, and electrical current, $i(t)$, is given by the differential equation (3.6).

$$u(t) = Ri(t) + L\frac{di(t)}{dt} + e_V(t) \tag{3.6}$$

Within a given operating point, the counter-electromotive force is linearly proportional to the rotor speed, $\omega(t)$. Assuming K_e as the proportionality constant, then the counter-electromotive force can be derived according to the equation (3.7).

$$e_V(t) = K_e\omega(t) \tag{3.7}$$

On the other hand, mechanical behavior can be derived from Newton's second law as stated by equation (3.8).

$$T_M - T_B - T_Q = J\frac{d\omega(t)}{dt} \tag{3.8}$$

where the motor torque, T_M, is a linear function of the electrical current, $i(t)$, according to $T_M = K_M i(t)$ for any positive constant K_M. The braking torque, T_B, depends on the friction coefficient, B, and is proportional to the rotator's speed, ω, by $T_B = B\omega(t)$. A constant torque, T_Q, due to other unaccounted factors, is also considered. In this reference frame, equation (3.9) is deduced.

$$K_M i(t) - B\omega(t) - T_Q = J\frac{d\omega(t)}{dt} \tag{3.9}$$

Combining (3.6) and (3.9) leads to the second order differential equation stated in (3.10).

$$\frac{LJ}{K_M}\frac{d^2\omega(t)}{dt^2} + \left(\frac{RJ}{K_M} + \frac{LB}{K_M}\right)\frac{d\omega(t)}{dt} + \left(\frac{RB}{K_M} + K_e\right)\omega(t) + \frac{RT_Q}{K_M} = u(t) \tag{3.10}$$

Now, neglecting the effect of the inductance and assuming that $K_e = K_M = K$, the previous expression results in the first-order differential equation (3.11).

$$\frac{RJ}{K}\frac{d\omega(t)}{dt} + \left(\frac{RB}{K} + K\right)\omega(t) + \frac{RT_Q}{K} = u(t) \tag{3.11}$$

To describe this dynamic behavior in SimTwo, it is necessary to determine the parameters R, B, K, and T_Q. For this purpose, a series of tests were conducted in which the electric current and the rotational speed of the motor at steady state were measured for different voltage values. The results obtained are presented in Table 3.2.

Table 3.2: Steady-state current and angular velocity for different motor voltages.

Voltage (V)	**Current (A)**	**Angular Velocity (rad/s)**
1	0.03914	0.55665
1.5	0.0429	0.98467
2	0.0497	1.38156
2.5	0.05502	1.7766
3	0.05765	2.20709
3.5	0.06337	2.61674
4	0.06616	2.98489
4.5	0.06962	3.29374
5	0.07476	3.70856

For the steady-state behavior, the system of equations is derived in (3.12).

$$\begin{cases} u(t) = Ri(t) + K\omega(t) \\ 0 = Ki(t) - B\omega(t) - Tq \end{cases} \tag{3.12}$$

Based on the results from Table 3.2, the parameters R and K of equation (3.12) are obtained, in the least squares sense, by solving the equation (3.13).

$$\boldsymbol{\theta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{u} \tag{3.13}$$

where $\mathbf{H}$ is the matrix of regressor vectors, $\mathbf{u}$ is the vector associated with the voltage values, and $\theta = \begin{bmatrix} R & K \end{bmatrix}^T$. The results obtained yielded $R = 7.29\ \Omega$ and $K = 1.2$. The same procedure was carried out for the second equation represented in (3.12), resulting in $B = 0.0133$ and $T_Q = 0.0396$. The value of the inertia coefficient, J, was determined from the DC gain of the system using equation (3.14).

$$J = \frac{1}{9.374}\frac{K}{R} = 0.0176 \tag{3.14}$$

Although the motor's transfer function can be directly calculated from its mathematical model, another approach was utilized for comparison and validation. Specifically, a pseudorandom binary sequence (PRBS) was employed to excite the system, and the resultant output was analyzed to determine the transfer function. This methodology was selected to provide an independent means of identifying the system's dynamics, thereby facilitating a comparison of the results with the transfer function derived from the first-principles model (FPM). This experimental approach aims to evaluate the accuracy and reliability of the mathematical model using an alternative system identification technique.

Experimental tests were conducted to assess its dynamic behavior when considering the system as a first-order system. A series of measurements were taken by applying a voltage input to the motor, which was driven by the PRBS signal illustrated in Figure 3.15. Applying a PRBS signal is particularly advantageous, as it encompasses a diverse set of frequency components that enables systems identifications. This methodology allows for determining the system's transfer function without necessitating an in-depth modeling of the motor's internal dynamics, thus providing a robust means of validating the theoretical model.

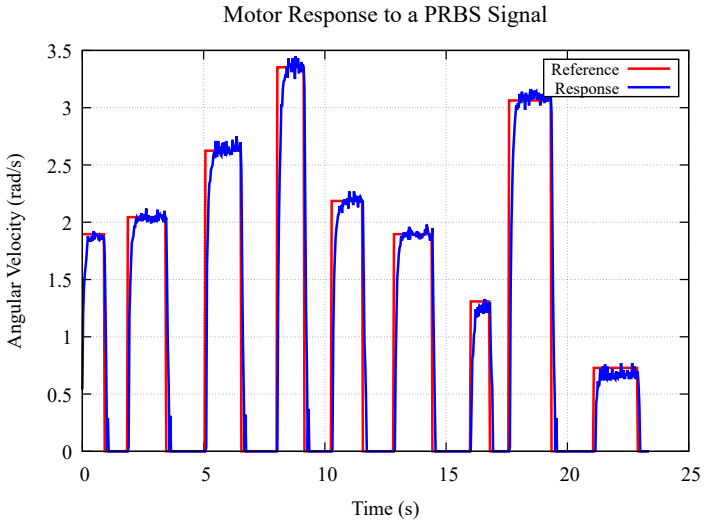


Figure 3.15: DC motor response to a PRBS signal.

An iterative system identification method was employed to derive the model parameters of a canonical first-order transfer function. This approach has led to the equation (3.15).

$$H(s) = \frac{9.374}{s + 12.7} \tag{3.15}$$

The transfer function is compared to the one derived from the FPM, as depicted in the subsequent figure. This comparison verifies the accuracy of the model obtained through system identification and confirms the validity of the assumptions incorporated within the theoretical model. As illustrated in Figure 3.16, both transfer functions demonstrate similarities, leading to a satisfactory result.

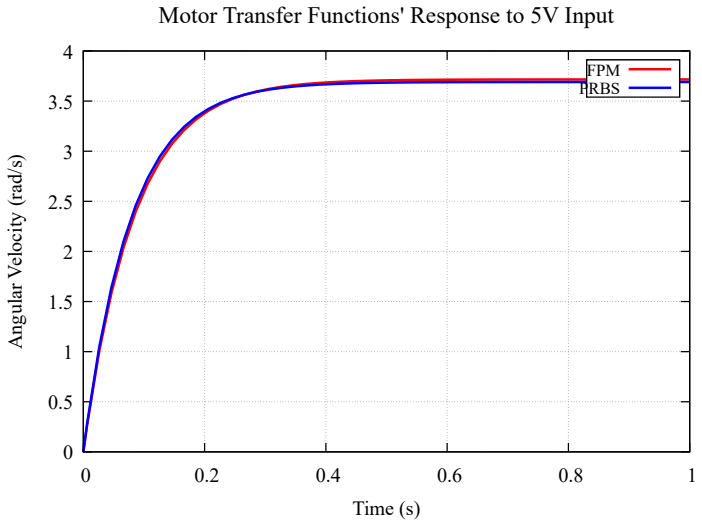


Figure 3.16: Comparison between the transfer function derived from the FPM and the transfer function based on the system identification method.

Chapter 4

Control of the Robot

This chapter presents a comprehensive examination of the control mechanisms employed in the robotic system, highlighting three principal areas: closed-loop control of the robot's DC motor, inverse kinematics for precise positioning, and path planning algorithms for efficient movement.

Initially, the chapter outlines the design and implementation of a proportional-integral (PI) controller aimed at regulating the motor's velocity. This discussion includes the tuning process, the integration of a feed-forward component to enhance response times, and the methodologies for testing carried out in MATLAB. The motor modeling approach, developed as part of the TEEM conference paper, provides foundational insights into the motor's behavior and performance under varying control parameters [25].

Subsequently, the chapter investigates the calculations associated with inverse kinematics, which are vital for determining the required joint angles to achieve specified target positions in three-dimensional space. This aspect of the study, as presented in the CoDIT paper [18], includes optimization of joint movements for efficient and precise end-effector positioning.

Lastly, the chapter analyzes path-planning strategies that encompass both linear and joint movement techniques, illustrating how the robot effectively navigates its environment. These strategies, also documented in the CoDIT paper [18], demonstrate the robot's capability to follow complex trajectories while minimizing error and maximizing smoothness in its movement. Collectively, these sections provide a thorough overview of the robot's control system, facilitating accurate and responsive operational capabilities.

4.1 Closed-loop DC Motor Controller

To regulate motor velocity, a proportional-integral (PI) controller was designed using the transfer function derived in the previous chapter [26], [27]. The PI controller's structure in the Laplace domain is defined by equation (4.1).

$$PI = K_c \frac{T_i s + 1}{T_i s} \tag{4.1}$$

In this equation, K_c is the inverse gain relating angular velocity to counter electromotive force and motor torque gain, while T_i denotes the coefficient associated with the open-loop response time, set to achieve 60% of its steady-state value. To improve response time, a feed-forward component with a gain of 35% of K_c was added. This combined controller configuration was tested in MATLAB, with results summarized in Figure 4.1. Additionally, a saturation block was included in the output to limit motor voltage within the operational range of -5 V to 5 V.

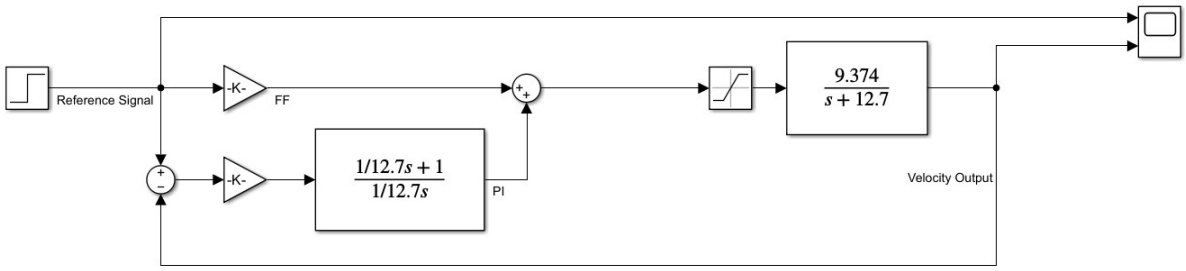


Figure 4.1: PI controller model implemented in Simulink, comprising the PI controller block and feed-forward component.

The closed-loop simulation in Figure 4.1 showcases the interaction between the controller and motor in response to a step input for a reference velocity of 2 rad/s. This input drives the controller to adjust the motor voltage to match the desired speed. After the controller's output, a saturation block limits voltage values within acceptable bounds. The resulting voltage is fed to the motor's transfer function model, producing the motor's velocity, which is then looped back to the controller. This feedback mechanism enables continuous error correction, adjusting the voltage based on the discrepancy between the target and actual velocity.

The simulation (Figure 4.2) confirmed the controller's efficacy, achieving a response time of approximately 200 milliseconds with minimal overshoot in the transient response.

Given these satisfactory results, the controller was adapted for discrete-time operation, enabling its implementation in microcontroller hardware. The simplified continuous-time transfer function of the controller, inclusive of the feed-forward constant, is expressed by equation (4.2).

$$G(s) = \frac{0.1067s + 1.355}{0.07874s} + 0.47418 \tag{4.2}$$

To obtain a discrete-time equation from the previous expression, the Tustin approximation method was employed [28]. Considering a 30 Hz sampling frequency, the controller transfer function in the z-domain takes the form in equation (4.3).

$$G(z) = \frac{1.639z - 1.071}{z - 1} + 0.47418 \tag{4.3}$$

The discrete-time controller was implemented in Simulink, and a one-second simulation was performed. The performance between the continuous and discrete domain controllers was then compared. Further details on the obtained results are in the following section.

The FPM and the system-identified model using PRBS sampling. The comparison between the transfer functions and the model in the time domain, as outlined in equation (3.11), was simulated, yielding the results presented in Figure 3.16. This figure demonstrates the close similarity between both transfer functions.

Figure 4.2 depicts the comparison results between the reference velocity and the actual motor output for evaluating the controller's performance. The figure demonstrates a satisfactory response time and also highlights that the discrete-time controller closely matches the performance of its continuous counterpart, albeit with some deviations attributable to the selected sample frequency and phase lag due to the sample and hold.

4.2 Robot's Inverse Kinematics

Unlike forward kinematics, which establishes the end effector's position based on joint angles, inverse kinematics ascertains the joint angles needed to reach a specified position in three-dimensional space. This task is notably more intricate than forward kinematics due to several factors, including the presence of singularities and the possibility of

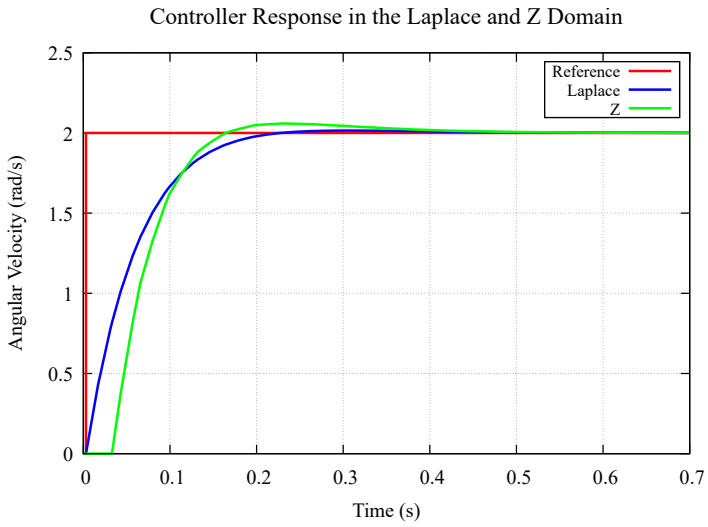


Figure 4.2: Comparison between the controller's response in the continuous-time and the discrete-time domain to a step reference signal.

multiple or no solutions. Singularities denote critical points where the robot's manipulator loses one or more DoF, effectively blocking the end effector's movement. Furthermore, inverse kinematics problems may yield multiple solutions, necessitating careful selection based on the task at hand. Conversely, some scenarios may present no feasible solution, especially when attempting to reach positions outside the robot's reachable workspace. To address these challenges, inverse kinematics solutions can be pursued analytically, using mathematical equations for precise solutions or employing iterative methods for approximate solutions numerically. Both approaches have their merits and limitations, requiring thoughtful consideration based on the specific requirements and complexity of the robotic system, given that analytic methods are more suitable for less DoFs [29].

Given the simplicity of this robot, geometric methods were employed to calculate its inverse kinematics, offering a faster and more optimized approach. The calculation process involved two main steps: determining the base angle to project the target point onto a single plane, similar to the shoulder and elbow angles, and establishing the relationship between the shoulder and elbow angles to reach the desired point within their planes. The base angle, defined by θ_1, was the first computed angle. To visualize the geometric calculations for this angle, refer to Figure 4.3. This illustration provides a top-down view of the robot's configuration.

As shown in Figure 4.3, the idea is to align the shoulder and elbow with the same

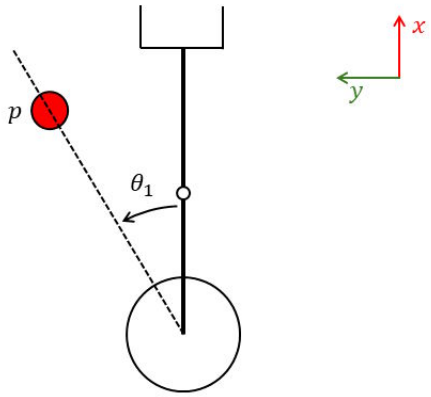


Figure 4.3: Top-down schematic of the robot configuration.

plane as the desired point, depicted as a dashed line. Consequently, a right triangle can be formed by drawing a line from the point to the plane of the robot's current rotation perpendicular to the plane. Given the known position of the desired point, the base angle can be calculated as specified in equation 4.4.

$$\theta_1 = \text{atan2}(y, x) \tag{4.4}$$

After calculating the base angle, the elbow angle was determined next. Figure 4.4 illustrates the robot reaching a given point in three-dimensional space to visualize the calculation of this angle.

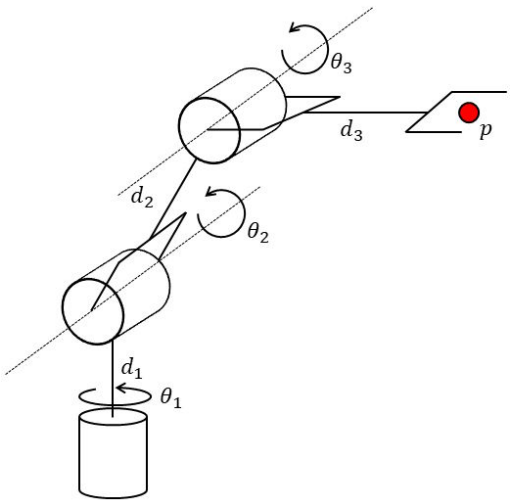


Figure 4.4: Side-view schematic of the robot configuration.

As shown in Figure 4.5, a triangle can be formed by the shoulder and elbow links. With the desired position known, the distance between the shoulder joint and the point can be calculated using the Euclidean distance, accounting for the offset of the base height.

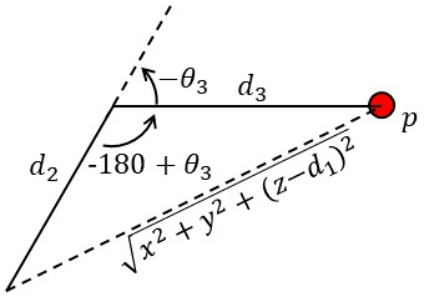


Figure 4.5: Triangle formed by the relation of the shoulder and elbow links.

Thus, utilizing the triangle formation and the cosine rule, the elbow angle, expressed as θ_3, can be determined by (4.5, 4.6).

$$x^2 + y^2 + (z - d_1)^2 = d_2^2 + d_3^2 - 2d_2d_3\cos(-180 + \theta_3) \tag{4.5}$$

$$\theta_3 = \pm\text{acos}\left(\frac{x^2 + y^2 + (z - d_1)^2 - d_2^2 - d_3^2}{2d_2d_3}\right) \tag{4.6}$$

Once the elbow angle is found, it becomes possible to calculate the shoulder angle. A right triangle is drawn from the intersection between the continuous projection of the shoulder link and a ninety-degree angle between that projection and the end effector. Another right triangle is formed between the height and distance of the shoulder link. Two new angles, α and β, can be extracted from these two triangles. Figure 4.6 illustrates the representation of the triangles used to calculate the shoulder angle. Given that all the distances of the triangles are known, the shoulder angle, characterized by θ_2, can be determined by (4.7, 4.8, 4.9).

$$\alpha = \text{atan2}\left(\frac{z - d_1}{\sqrt{x^2 + y^2}}\right) \tag{4.7}$$

$$\beta = \text{atan2}\left(\frac{d_3\sin(-\theta_3)}{d_2 + d_3\cos(-\theta_3)}\right) \tag{4.8}$$

$$\theta_2 = \alpha + \beta = \text{atan2}\left(\frac{z - d_1}{\sqrt{x^2 + y^2}}\right) + \text{atan2}\left(\frac{d_3 \sin(-\theta_3)}{d_2 + d_3 \cos(-\theta_3)}\right) \tag{4.9}$$

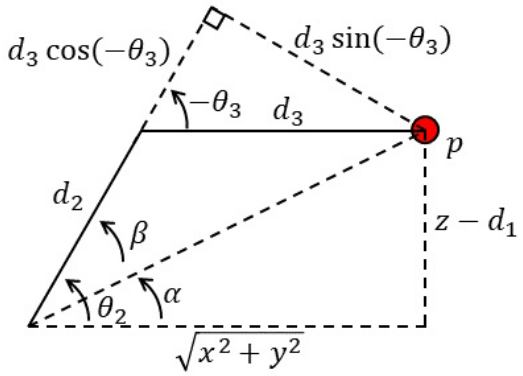


Figure 4.6: Right triangles formed to calculate the shoulder angle.

4.3 Path Planning Algorithms

Path planning is fundamental to robot control, as it dictates how a robot navigates through its environment to reach specific goals. The library developed for the robot implemented two primary types of movements for robot manipulators: linear and joint movement. These methods are widely recognized in the robotics industry for effectively controlling the motion of robotic arms and end effectors [30]. It is paramount to add that the path planning algorithms written were based on the first iteration of the servo motors where the original controller was used, leaving the need for the emulation of velocity control by controlling the dead time duration between points in the path.

Linear movements, true to their name, follow direct trajectories in three-dimensional space, which proves critical for applications that require utmost precision, particularly when an end effector must approach a target location accurately. For instance, in tasks like assembly or surgical applications, where even the slightest deviation can lead to errors, linear movements ensure that the robot adheres closely to the intended path. On the other hand, joint movement entails adjusting each joint separately to reach a desired point in space without drawing attention to the specific trajectory taken. This method is generally applied when the exact route is less important, such as when speed is prioritized over precision.

4.3.1 Linear Movement

Executing linear movement involves a systematic approach composed of two essential steps. First, a linear interpolation must be performed between the current position of the end effector and the target position. This involves dividing the trajectory into multiple segments based on a defined resolution. Second, the inverse kinematics for each interpolated position must be computed to determine the necessary servo angles for the robot's joints, ensuring that the end effector successfully reaches the desired location.

The implementation details of this linear movement function are encapsulated in the pseudo-code illustrated in Algorithm 1. The algorithm calculates the number of steps required to achieve the desired motion based on the distance and resolution. It determines the time step for each iteration by dividing the distance by the velocity, ensuring that the movement occurs at a constant speed.

Algorithm 1 Linear Movement Function

Require: x, y, z, *velocity*
1: $steps \leftarrow$ distance/resolution
2: $time\_step \leftarrow \frac{\text{distance}}{\text{velocity}}$/steps
3: $interpolated\_positions \leftarrow$ linear_interpolation(current_position, wanted_position)
4: **for** $i \leftarrow 0$ **to** *steps* **do**
5: $interpolated\_angles \leftarrow$ calculate_inverse_kinematics(interpolated_positions)
6: **set servo angles** to interpolated angles
7: **delay** for time step
8: **end for**

4.3.2 Joint Movement

In contrast to linear movement, implementing joint movement for path planning is comparatively more straightforward. This method consists of two main steps focusing primarily on the robot's joints. The initial step involves calculating each servo joint's current and desired angles. Following this, it incrementally adjusts each joint's angles to reach the target positions steadily.

The essential operations for this movement strategy are outlined in the pseudo-code provided in Algorithm 2. Beginning similarly to the linear movement, the algorithm first calculates the number of steps based on the joint movement's distance and resolution. It then computes the difference between the wanted and current angles to determine the

adjustment required for each step, ensuring that the servos increment toward their targets in a smooth and controlled manner.

Algorithm 2 Joint Movement Function

Require: x, y, z, $velocity$

$steps \leftarrow$ distance/resolution
$time\_step \leftarrow \frac{\text{distance}}{\text{velocity}}$/steps
$step\_factors \leftarrow \frac{\text{wanted\_angle} - \text{current\_angle}}{\text{steps}}$
for $i \leftarrow 0$ **to** $steps$ **do**
    $angles \leftarrow \text{current\_angle} + \text{step\_factors} \times (i + 1)$
    set servo angles to angles
    delay for time step
end for

Chapter 5

3D Robot Simulation

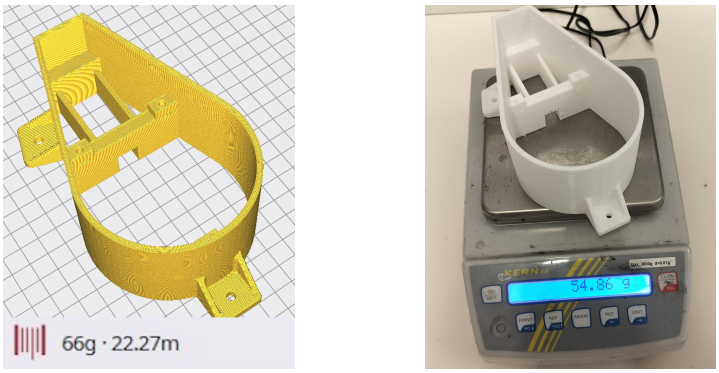
A physics-based simulator was employed to simulate the manipulator robot and accurately represent its rigid body dynamics. The chosen simulator for this purpose was SimTwo, an advanced platform capable of simulating various robotic configurations. It supports robots characterized as combinations of rigid bodies interconnected by optionally powered hinges or prismatic joints. The simulator accommodates multiple mobility mechanisms, including classical wheels, omnidirectional wheels, and propellers.

SimTwo achieves a realistic simulation environment by decomposing the robot into a system of rigid bodies with hinge or prismatic joints, which electric motors can actuate. Specific joint types, including hinge, universal, and prismatic, can be integrated with an actuation system comprising a drive, a motor, and an encoder. Various control strategies can be utilized, including PID controllers for position or velocity regulation, as well as state feedback controllers.

The DC motor model includes several nonlinear elements that account for voltage saturation, current limitations, and Coulomb friction. Beyond the low-level control framework, SimTwo can provide reference signals to these controllers from a user-implemented higher-level controller. This higher-level controller can be developed utilizing a scripting language that is both editable and compilable within the SimTwo environment or a remote application capable of communication via UDP packets or through the serial port.

To effectively replicate the physical robot, several critical factors required thorough consideration. Foremost among these was the determination of the weight of each component. This information was essential for conducting accurate calculations within the

simulated environment. Additionally, the parameters for the DC motor were crucial, having been previously established through mathematical modeling and experimental data related to the servo, as detailed in Chapter 3. It was necessary to weigh the components using a balance instead of relying solely on the weight estimation provided by the Creality Slicer software. Experimental results showed a significant discrepancy between the weight estimated by the slicer and the actual weight of the printed parts. As shown in Figure 5.1, the slicer estimated the weight of the robot's base to be 66 g, whereas the actual measured weight was 54.86 g. This amounts to an approximate 11 g difference, which is a substantial deviation when precision is required for accurate simulations.



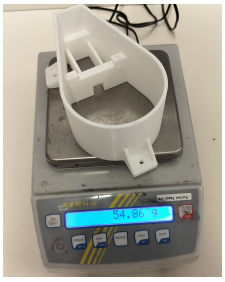
(a) Weight estimation of the base of the robot using Creality Slicer.

(b) Weighting of the base of the robot using a balance.

Figure 5.1: Comparison between the weight estimation of the robot's base provided by Creality Slicer and the actual measured weight using a balance.

Therefore, each 3D-printed component made from PLA material was weighed using the balance to ensure the integrity of measurements. Figure 5.2 depicts some robot components placed on a scale.

Upon weighing all requisite robot components, the specifications and weight parameters for the DC motor were integrated into the simulator's XML file. This file serves as a configuration blueprint that delineates the essential parameters required for simulating operations. Furthermore, the construction of the robot within the simulated environment necessitated the use of simple geometric shapes, such as cylinders and cuboids. This design choice is inherently linked to the capabilities and limitations of the physics engine, referred to as Open Dynamics Engine (ODE), which SimTwo employs for collision detection and rigid body dynamics calculations. ODE is optimized to manage collisions and dynamics involving simple geometric primitives efficiently. The geometric shapes were



(a) Base of the robot, serving as the foundational structure that supports all other components.

(b) Hip of the robot.

(c) Plate that, coupled with the bearings, ensures a smooth hip rotation.

(d) Big gear component used to amplify the torque generated by the robot's motor.

(e) Gripper of the robot.

(f) Small gear component used to amplify the torque generated by the robot's motor.

(g) Tri-link, an essential linkage that connects three parts of the robot.

(h) Shoulder arm of the robot.

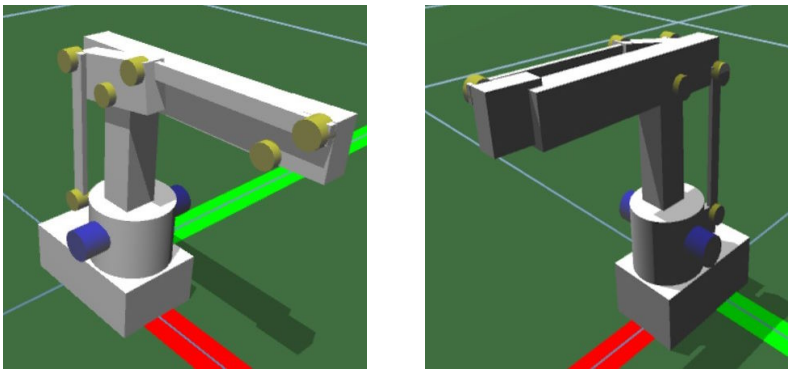
(i) Variable arm mechanism, serving as a link for the hip and elbow.

Figure 5.2: Weighing of various robot components.

required to be mechanically connected using the same joints and hinges utilized in the robot to ensure movement and interaction accuracy. Appendix 6 provides the parameters defined within the XML file, which are vital for establishing the robot's characteristics and behavior within the simulation environment.

Utilizing cuboids, spheres, and cylinders enables the simulation to conduct collision calculations, contact resolution, and other physics-related computations with significantly less processing overhead. In contrast, complex three-dimensional meshes can impose a substantial computational burden due to their intricate geometry, resulting in increased processing times and potential instability in real-time simulations. These constraints guided the modeling process, as SimTwo's scene editor is specifically configured to support these simpler geometric shapes. While the simplified geometry may not visually reflect the complexities of the actual robot, it retains essential features, including mass distribution, joint placements, and dynamic properties, which are critical for accurate physical simulation.

A 3D representation of the physical robot was achieved within the simulation environment after successfully implementing all parameters into the XML file. Figure 5.3 illustrates the 3D rendering of the robot as it appears within the simulator.



(a) Left side view of the robot implemented inside the simulator.

(b) Right side view of the robot implemented inside the simulator.

Figure 5.3: 3D rendering of the robot constructed inside the physics simulator.

Chapter 6

Conclusions and Future Work

By developing and integrating a manipulator robot designed for educational purposes, this work has introduced a cost-effective solution to facilitate hands-on training in robotics, addressing fundamental challenges academic institutions face globally. This project delved into the intricate aspects of designing and implementing a robotic manipulator and underscored the necessity of bridging the divide between theoretical knowledge and practical, real-world applications. Through the creation of a custom robotic gripper, the implementation of kinematic and path-planning algorithms, modeling of servomotors, and the development of a custom controller, this project provides a comprehensive and engaging robotics learning experience in a financially considerate manner.

A primary motivation for this initiative was the prohibitive cost associated with acquiring industrial robots, which restricts many institutions from offering comprehensive robotics courses. This project has demonstrated a viable approach to developing a robotic manipulator that, while more economical than industrial models, still provides a stimulating learning environment for students. By enabling direct interaction with robotic hardware and control systems in a laboratory setup, this manipulator effectively narrows the gap between theoretical understanding and practical skill development. The system was designed to focus on modularity and user-friendliness, ensuring adaptability for various educational objectives. As evidenced by the project's goals and successful outcomes, students can now engage with intricate concepts such as kinematic chains, dynamic modeling, and controller design within a realistic yet accessible framework.

Throughout this project, the kinematic structure of the robot, including the custom-designed gripper, was successfully implemented and tested, establishing a functional platform for imparting knowledge in robotics. The development and effective integration of path-planning algorithms provided foundational experience in maneuvering robotic manipulators through predefined trajectories. The modeling of servo motors and controller design guaranteed that the manipulator could execute movements with precision, stability, and responsiveness, thereby enhancing its effectiveness as an educational tool. Additionally, incorporating the robot model into a physics simulation engine allowed students to visualize and analyze robotic movements within a controlled, virtual environment, enriching the overall educational experience.

The significance of such a robotic platform lies in its capacity to offer practical insights into robotics while remaining affordable and scalable. This project illustrates that cost-effective robotic systems can serve as viable alternatives to high-cost industrial models within educational contexts, thus enabling institutions with limited budgets to cultivate robust robotics programmes. Such solutions expose students to technical challenges, foster problem-solving abilities, and develop analytical thinking essential in engineering education, particularly for those aspiring to careers in automation and robotics.

Regarding future work, several avenues exist to enhance and expand this initiative. Implementing a HIL control is a promising next step, facilitating seamless integration between the virtual simulation and the physical robot. HIL systems allow more intricate interactions by enabling physical components to respond to inputs from simulated environments and vice versa. Such implementation would enrich students' comprehension of real-world applications, allowing them to test their algorithms and control strategies in a semi-virtual environment before transferring them to actual hardware. Moreover, incorporating HIL would further refine students' practical experience with robotics and provide a controlled setup for testing advanced algorithms without jeopardizing physical components.

Another prospective direction is replicating the manipulator robot, enabling larger student cohorts to engage concurrently in programming and controlling robots. By creating multiple robot units, educational institutions could scale their robotics programs to accommodate a greater number of students and promote collaborative learning. This approach also allows students to partake in team-based projects that simulate industrial environments, thereby enhancing the realism and applicability of their training experiences.

Bibliography

[1] S. Evripidou, K. Georgiou, L. Doitsidis, A. Amanatiadis, Z. Zinonos, and S. Chatzichristofis, "Educational robotics: Platforms, competitions and expected learning outcomes," *IEEE Access*, vol. 8, pp. 219 534–219 562, Jan. 2020. DOI: 10.1109/ACCESS.2020.3042555.

[2] D. Bazylev, A. Margun, K. Zimenko, A. Kremlev, and E. Rukujzha, "Participation in robotics competition as motivation for learning1," *Procedia - Social and Behavioral Sciences*, vol. 152, Oct. 2014. DOI: 10.1016/j.sbspro.2014.09.330.

[3] C.-H. Chen, C.-K. Yang, K. Huang, and K.-C. Yao, "Augmented reality and competition in robotics education: Effects on 21st century competencies, group collaboration and learning motivation," *Journal of Computer Assisted Learning*, vol. 36, Jul. 2020. DOI: 10.1111/jcal.12469.

[4] M. Vahl, *Cobots in the classroom: How the ur academy upskills students and industry professionals*, Universal Robots, 2021. [Online]. Available: https://www.universal-robots.com.

[5] K. AG, *Kuka college – your path to the future of automation*, KUKA, 2021. [Online]. Available: https://www.kuka.com.

[6] Dobot, *The dobot magician – desktop 4-axis robotic arm for education*, Dobot, 2020. [Online]. Available: https://www.dobot.cc.

[7] A. Suarez, D. García-Costa, J. Perez, E. López-Iñesta, F. Grimaldo, and J. Torres, "Hands-on learning: Assessing the impact of a mobile robot platform in engineering learning environments," *Sustainability*, vol. 15, no. 18, 2023, ISSN: 2071-1050. DOI: 10.3390/su151813717. [Online]. Available: https://www.mdpi.com/2071-1050/15/18/13717.

[8] L. Brancalião, M. Alvarez, J. Coelho, M. Conde, P. Costa, and J. Gonçalves, "Programming mobile robots in an educational context: A hardware-in-the-loop approach," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2024, pp. 1820–1824. DOI: 10.1109/CoDIT62066.2024.10708336.

[9] L. Brancalião, M. Alvarez, J. Coelho, M. Conde, P. Costa, and J. Gonçalves, "Learning mobile robotics: An approach based on a classroom competition," in *12th edition of the Technological Ecosystems for Enhancing Multiculturality (TEEM) conference*, 2024.

[10] J. Galarza, L. Escobar, and D. Loza, "6 dof anthropomorphic robot as a platform for teaching robotics," in *2020 IEEE/ASME INTERNATIONAL CONFERENCE ON ADVANCED INTELLIGENT MECHATRONICS (AIM)*, ser. IEEE ASME International Conference on Advanced Intelligent Mechatronics, IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), ELECTR NETWORK, JUL 06-09, 2020, IEEE; ASME, 2020, pp. 631–636, ISBN: 978-1-7281-6794-7.

[11] S. Patil, S. C. Alvares, P. Mannam, O. Kroemer, and F. Z. Temel, "Deltaz: An accessible compliant delta robot manipulator for research and education," in *2022 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS)*, ser. IEEE International Conference on Intelligent Robots and Systems, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, JAPAN, OCT 23-27, 2022, IEEE; Royal Soc Japan; IEEE Robot & Automat Soc; IES; SICE; New Technol Fdn, 2022, pp. 13 213–13 219, ISBN: 978-1-6654-7927-1. DOI: 10.1109/IROS47612.2022.9981257.

[12] A. C. Passos, F. Luiz Junior, and H. H. de Arruda, "Project-based learning activity with robotics: A low-cost case study," in *2022 LATIN AMERICAN ROBOTICS SYMPOSIUM (LARS), 2022 BRAZILIAN SYMPOSIUM ON ROBOTICS (SBR), AND 2022 WORKSHOP ON ROBOTICS IN EDUCATION (WRE)*, T. Homem, R. Bianchi, B. DaSilva, C. Curvelo, and M. Pinto, Eds., Latin American Robotics Symposium (LARS) / Brazilian Symposium on Robotics (SBR) / Workshop on Robotics in Education (WRE), Sao Bernardo do Campo, BRAZIL, OCT 18-21, 2022, Inst Elect & Elect Engineers; Coordenacao Aperfeicoamento Pessoal Nivel Super; Conselho Nacl Desenvolvimento Cienfico & Tecnologico; Soc Brasileira Computacao; Centro Univ FEI, 2022, pp. 360–365, ISBN: 978-1-6654-6280-8. DOI: 10.1109/LARS/SBR/WRE56824.2022.9995849.

[13] M. M. Marinho, H.-C. Lin, and J. Zhao, "Umirobot: An open-{software, hardware} low-cost robotic manipulator for education," in *2023 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, IROS*, ser. IEEE International Conference on Intelligent Robots and Systems, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, OCT 01-05, 2023, IEEE; RSJ, 2023, pp. 4464–4471, ISBN: 978-1-6654-9190-7. DOI: 10.1109/IROS55552.2023.10341347.

[14] A.-A. Ayazbay, G. Balbaye, S. Orazaliyeva, K. Gromaszek, and A. Zhauyt, "Trajectory planning, kinematics, and experimental validation of a 3d-printed delta robot manipulator," *INTERNATIONAL JOURNAL OF MECHANICAL ENGINEERING AND ROBOTICS RESEARCH*, vol. 13, no. 1, 2024. DOI: 10.18178/ijmerr.13.1.113-125.

[15] A. Shintemirov, T. Taunyazov, B. Omarali, *et al.*, "An open-source 7-dof wireless human arm motion-tracking system for use in robotics research," *SENSORS*, vol. 20, no. 11, Jun. 2020. DOI: 10.3390/s20113082.

[16] M. S. Benyeogor, T. L. Igbigbi, O. Lawal I, *et al.*, "Development of microcontroller-based manipulator arm: A trilemma problem-solving framework for robotics and stem education in nigerian schools," in *2024 XVI CONGRESO DE TECNOLOGIA, APRENDIZAJE Y ENSENANZA DE LA ELECTRONICA, TAEE 2024*, ser. Technologies Applied to Electronics Teaching Conference, 16th Congress on Technology, Learning, and Teaching of Electronics (TAEE), Sch Telecommunicat Engn, Malaga, SPAIN, JUN 26-28, 2024, 2024, ISBN: 979-8-3503-4868-2; 979-8-3503-4867-5. DOI: 10.1109/TAEE59541.2024.10604968.

[17] C. Rodriguez-Padilla, M. Martinez Guerrero, A. Stancu, *et al.*, "A study on teaching cyber-physical systems with a customized branded mobile robot for industry 4.0," in *2024 IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE, EDUCON 2024*, ser. IEEE Global Engineering Education Conference, 2024, ISBN: 979-8-3503-9402-3; 979-8-3503-9403-0. DOI: 10.1109/EDUCON60312.2024.10578683.

[18] J. A. B. Coelho, L. Brancalião, M. Alvarez, P. Costa, and J. Gonçalves, "Prototyping and control of an educational manipulator robot," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2024, pp. 1814–1819. DOI: 10.1109/CoDIT62066.2024.10708583.

[19] C. Franciscone, *Eezybotarm mk2*, Feb. 2018. [Online]. Available: http://www.eezyrobots.it/eba_mk2.html.

[20] ABB, *Irb 460*. [Online]. Available: https://new.abb.com/products/robotics/robots/articulated-robots/irb-460.

[21] Eurobots, *Irb 460*. [Online]. Available: https://www.eurobots.com.br/irb-460-pt.html.

[22] H. Technology, *Wemos d1 r32 esp32 wi-fi and bluetooth board*. [Online]. Available: https://handsontec.com/dataspecs/module/ESP/WeMos%20D1%20R32.pdf.

[23] C. Jamie, *D1 r32 – esp32*, University of Auckland, Aug. 2021. [Online]. Available: https://designtech.blogs.auckland.ac.nz/d1-r32-esp32/.

[24] J. Gonçalves, J. Lima, and P. Costa, "Dc motors modeling resorting to a simple setup and estimation procedure," in *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*, ser. Lecture Notes in Electrical Engineering, vol. 321, Springer, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-10380-8_42.

[25] J. A. B. Coelho, L. Brancalião, M. Alvarez, P. Costa, and J. Gonçalves, "Modeling and control of an educational manipulator robot joint," in *12th edition of the Technological Ecosystems for Enhancing Multiculturality (TEEM) conference*, 2024.

[26] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems-Third Edition*. Nov. 2022, ISBN: 0-9791226-3-5; 978-0-9791226-3-7.

[27] M. Ibrahim, A. Nathim, and B. Salih, "Pi controller for dc motor speed realized with simulink and practical measurements," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 11, p. 119, 2020. DOI: 10.11591/ijpeds.v11.i1.pp119-126.

[28] J. Malinen, "Tustin's method for final state approximation of conservative dynamical systems," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, 2011. DOI: 10.3182/20110828-6-IT-1002.01445.

[29] S. Kucuk and Z. Bingul, "Robot kinematics: Forward and inverse kinematics," in Dec. 2006, ISBN: 3-86611-285-8. DOI: 10.5772/5015.

[30] S. Dietrich, *Robot motion command types: Understanding linear, joint, and arc movement*, Dec. 2022. [Online]. Available: https://control.com/technical-articles/robot-motion-command-typesexplained/.

Annexes

Annex A: XML Parameters and Structure for SimTwo

The following XML code defines the structure and parameters of the robotic manipulator used in this research. It includes details on the robot's components, joint configurations, and other critical parameters necessary for simulation and implementation.

```
<?xml version="1.0"?>

<robot>
    <defines>
        <!-- motor parameters -->
        <const name='joint_height' value='0.050' />
        <const name='joint_radius' value='0.015' />
    </defines>

    <solids>
        <cuboid>
            <ID value = 'base' />
            <mass value = '0.0874' /> <!-- 87.4 g -->
            <size x = '0.142' y = '0.088' z = '0.055' />
            <pos x = '0.027' y = '0' z = '0.0275' />
            <rot_deg x = '0' y = '0' z='0' />
            <color_rgb r = '255' g = '255' b = '255' />
        </cuboid>

        <cylinder>
            <ID value= 'hip' />
            <mass value= '0.04142' /> <!-- 41.42 g -->
```

```
    <size radius= '0.041' z= '0.070' />
    <pos x= '0' y= '0' z= '0.084' />
    <rot_deg x= '0' y= '0' z= '0' />
    <color_rgb r='255' g='255' b='255' />
</cylinder>

<cuboid>
    <ID value = 'varm' />
    <mass value = '0.00567' /> <!-- 5.67 g -->
    <size x = '0.075' y = '0.010' z = '0.030' />
    <pos x = '0.02138' y = '-0.017' z = '0.08932' />
    <rot_deg x = '0' y = '6.83710716' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'link147' />
    <mass value = '0.00436' /> <!-- 4.36 g -->
    <size x = '0.013' y = '0.0055' z = '0.148' />
    <pos x = '0.05473' y = '-0.020' z = '0.1595' />
    <rot_deg x = '0' y = '1.10787615' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'shoulder' />
    <mass value = '0.02745' /> <!-- 27.45 g -->
    <size x = '0.032' y = '0.033' z = '0.161' />
    <pos x = '-0.00095' y = '0.005' z = '0.1565' />
    <rot_deg x = '0' y = '0' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'elbow' />
    <mass value = '0.024' /> <!-- 24 g -->
    <size x = '0.215' y = '0.028' z = '0.039' />
```

```
    <pos x = '-0.04545' y = '-0.00605' z = '0.2375' />
    <rot_deg x = '0' y = '0' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'link135_angled' />
    <mass value = '0.00453' /> <!-- 4.53 g -->
    <size x = '0.013' y = '0.0055' z = '0.148' />
    <pos x = '0.03731' y = '0.03175' z = '0.1795' />
    <rot_deg x = '0' y = '0.22897185' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'trialink' />
    <mass value = '0.00774' /> <!-- 7.74 g -->
    <size x = '0.084' y = '0.020' z = '0.040' />
    <pos x = '0.000025' y = '0.01872' z = '0.2379' />
    <rot_deg x = '0' y = '7.72847118' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'link135' />
    <mass value = '0.00414' /> <!-- 4.14 g -->
    <size x = '0.1605' y = '0.0055' z = '0.013' />
    <pos x = '-0.1066' y = '0.016' z = '0.25584' />
    <rot_deg x = '0' y = '0.22476793' z='0' />
    <color_rgb r = '255' g = '255' b = '255' />
</cuboid>

<cuboid>
    <ID value = 'trialink_front' />
    <mass value = '0.00651' /> <!-- 6.51 g -->
    <size x = '0.046' y = '0.022' z = '0.0415' />
    <pos x = '-0.164' y = '0.00145' z = '0.24156' />
```

```
        <rot_deg x = '0' y = '1.10955061' z='0' />
        <color_rgb r = '255' g = '255' b = '255' />
    </cuboid>
</solids>

<articulations>
    <default>
        <draw radius='0.015' height='0.050' rgb24='8F0000' />
        <motor ri='0.5' ki='0.3' vmax='24' imax='20' active='1' />
        <gear ratio='1' />
        <friction bv='0.05' fc='0.1' />
        <encoder ppr='1000' mean='0' stdev='0' />
        <controller mode='pidposition' kp='10' ki='200' kd='0.0' kf='0.0'
        ↪  active='1' period='2' />
        <spring k='0' zeropos='0' />
    </default>

    <joint>
        <ID value='hip_joint' />
        <pos x='0' y='0' z='0.055' />
        <axis x='0' y='0' z='1' wrap='1' />
        <connect B1='hip' B2='base' />
        <type value='Hinge' />
    </joint>

    <joint>
        <ID value='elbow_joint' />
        <pos x='0' y='-0.04083' z='0.092' />
        <axis x='0' y='1' z='0' wrap='1' />
        <connect B1='varm' B2='hip' />
        <type value='Hinge' />
    </joint>

    <joint>
        <ID value='link147_connector' />
        <draw radius='0.01' height='0.015' rgb24='008F8F' />
        <motor ri='0.5' ki='0.3' vmax='24' imax='20' active='0'/>
```

```
    <pos x='0.05438' y='-0.02275' z='0.09225' />
    <axis x='0' y='1' z='0' wrap='1' />
    <connect B1='link147' B2='varm' />
    <type value='Hinge' />
</joint>

<joint>
    <ID value='shoulder_joint' />
    <pos x='0' y='0' z='0.092' />
    <axis x='1' y='0' z='0' wrap='1' />
    <connect B1='shoulder' B2='hip' />
    <type value='Hinge' />
</joint>

<joint>
    <ID value='link135_connector' />
    <pos x='0.02734' y='0.02159' z='0.1968' />
    <axis x='0' y='1' z='0' wrap='1' />
    <connect B1='link135' B2='elbow' />
    <type value='Hinge' />
</joint>

<joint>
    <ID value='trialink_joint' />
    <pos x='0.04715' y='0.03063' z='0.1968' />
    <axis x='1' y='0' z='0' wrap='1' />
    <connect B1='trialink' B2='link135' />
    <type value='Hinge' />
</joint>

<joint>
    <ID value='trialink_front_joint' />
    <pos x='-0.11406' y='0.00689' z='0.2617' />
    <axis x='1' y='0' z='0' wrap='1' />
    <connect B1='trialink_front' B2='trialink' />
    <type value='Hinge' />
</joint>
```

```
    </articulations>
</robot>
```