2024 10th International Conference on Control, Decision and Information Technologies (CoDIT) | 979-8-3503-7397-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/CoDIT62066.2024.10708583

Prototyping and Control of an Educational Manipulator Robot*

João Coelho[1], Laiany Brancalião[1,3], Mariano Alvarez[1,4], Paulo Costa[4] and José Gonçalves[1,2]

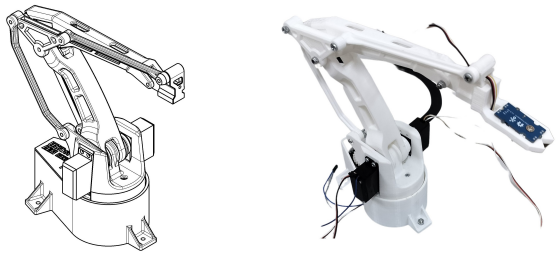
***Abstract*— This article presents the prototyping of an educational manipulator robot, based on the "EEZYbotARM Mk2" robot, tailored for first-year master's students in the field of robotics. The project encompasses the assembly of the robot arm, computation of both forward and inverse kinematics, and analysis of two path-planning movement algorithms. These features are consolidated into an Arduino library to streamline the process for students to generate instructions for the robot. The "EEZYbotARM Mk2" features a three-degree-of-freedom revolute arm with a gripper that remains parallel to its base at all times, enhancing its suitability for educational applications such as pick-and-place tasks. The article provides detailed descriptions of the materials and methods employed, along with proposed challenges for student engagement.**

I. INTRODUCTION

This research project, focused on robotics education, instrumentation, and control, involves the prototyping and control of an educational manipulator robot. The project requires the assembly and adaptation of the "EEZYbotARM Mk2" manipulator robot, originally developed by Carlo Franciscone [1]. The "EEZYbotARM Mk2" is an open-source design featuring a three-degree-of-freedom revolute robot arm with a distinctive characteristic: its gripper remains parallel to the base of the robot. This design attribute enhances its suitability for pick-and-place applications. Figure 1 illustrates the CAD design of the robot arm implemented and the final version of the prototyped robot.

The robot is powered by three 5 V servo motors, each one handling one of the three degrees of freedom within the robot's structure. These servos have a range of 180 degrees. Central to its operation is a Wemos D1 R32 board, equipped with the ESP32 processor, serving as the controller of the robot. Additionally, the structural components of the robot were printed using PLA.

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 choice of the Wemos D1 R32 board with the ESP32 processor 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 [2]. Section II details the prototyping process of the robot, while Section III discusses the open-source library developed for calculating kinematics and movement.



(a) CAD rendering of the "EEZYbotARM Mk2" robot arm.

(b) Assembled "EEZYbotARM Mk2" robot.

Fig. 1: Technical rendering and the assembled prototype of the "EEZYbotARM MK2" robot.

The school laboratories are equipped with the "ABB IRB 1400" industrial robot, as shown in Figure 2. To prepare the students to handle the programming of this industrial robot, the "EEZYbotARM Mk2" is used as a stepping stone to accommodate the students in dealing with such manipulators. The control library, developed for this robot, resembles the naming conventions and parameterization of functions in RAPID language, the programming language used for the "ABB IRB 1400" industrial robot. Following their familiarization and testing with the "EEZYbotARM Mk2", students will be prepared to program the industrial robot.

To evaluate student comprehension, two tasks, the Tower of Hanoi and Snake Challenges, have been proposed. These challenges, outlined in Section V, aim to evaluate students' understanding and application of robotics concepts through hands-on tasks.

Lastly, Section VI concludes this article by summarizing the project's findings, outcomes, and future work.

*This work was not supported by any organization

[1]Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, Bragança, 5300-253, Portugal joaocoelho@ipb.pt, laiany@ipb.pt, marianoalvarez@ipb.pt, goncalves@ipb.pt

[2]Laboratory for Sustainability and Technology in Mountain Regions (SusTEC), Instituto Politécnico de Bragança, Campus de Santa Apolónia, Bragança, 5300-253, Portugal

[3]University of Léon, Campus de Vegazana s/n, León, 24071, Spain

[4]Institute of Engineering Systems and Computers, Technology and Science (INESC TEC), University of Porto, Porto, 4200-465, Portugal paco@fe.up.pt

Fig. 2: Representation of the "ABB IRB 1400" industrial robot manipulator in the school laboratories.

II. PROTOTYPING OF THE ROBOT

The "EEZYbotARM Mk2" prototype integrates specialized components for enhanced functionality. Customized grippers and servo motors were implemented to achieve the desired performance. The servo motors were adapted to provide position feedback via potentiometers connected to the analog ports of the microcontroller board. Control signals for the servo motors, operating at 3.3 V, were generated with PWM-capable digital ports [3]. A gripper design was created to incorporate an electromagnet for handling iron pieces, in pick-and-place operations. The electromagnet's control signal was also ordered by a PWM port. All components, including servo motors, electromagnet, and microcontroller board, were powered by a 5 V power supply. Refer to Figure 3 for a visual representation of the system's connections. Yellow wires denote the control signals, purple wires indicate the potentiometer connections of the servo motors, red wires illustrate the 5 V power supply and black wires represent the ground connection.

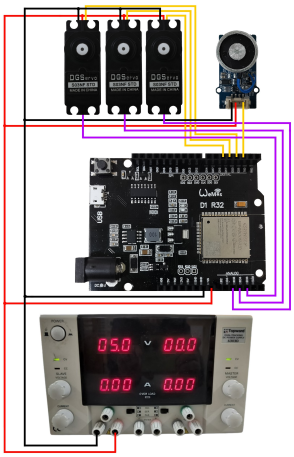


Fig. 3: Connection diagram of the robot arm.

As previously mentioned, modifications were made to the servo motors to enable position monitorization. Specifically, a wire was soldered to the middle pin of the potentiometer to access the analog signal corresponding to its position. Figure 4 provides a visual depiction of a servo motor with the potentiometer's wiper soldered to a wire.

Fig. 4: Soldered wire connected to the potentiometer's wiper on the servo motor.

III. MATERIALS AND METHODS

This section details the materials and methodologies employed in prototyping the system outlined in Section I. Specifically, it covers the assembly of the robot, hardware integration, determination of the kinematics of the robot, and the firmware responsible for the robot control. The robot's kinematics was initially tested in MATLAB using Peter Corke's "Robotics Toolbox" [4], and then it was implemented into an Arduino library.

A. Kinematics

Kinematics is a branch of physics that focuses on analyzing the motion of bodies and joints without factoring in forces. In robotics, kinematics plays a pivotal role in controlling robot movement. This branch is divided into two main components: forward kinematics and inverse kinematics. Forward kinematics determines the position of the robot's end effector based on joint angles, while inverse kinematics computes joint angles based on a desired end effector position.

The subsequent sections elaborate on forward and inverse kinematics as applied to the three-degree-of-freedom revolute robot, "EEZYbotARM Mk2".

1) Forward Kinematics: To define movements for the robot, forward kinematics is necessary to extract the position of the robot based on the angles of its axis. As such, the standard Denavit-Hartenberg method was applied to extract the transformation matrix of the end effector about the base of the robot [5]. Figure 5 illustrates the kinematics diagram of the implemented robot.

Observing the Figure 5, the robot features two distinct types of revolute joints. The initial joint, referred to as the hip joint, promotes yaw movement (rotation about the 'z' axis). Subsequently, the second joint, identified as the shoulder joint, and the third joint, known as the elbow joint, enable roll movement (rotation about the 'y' axis). To calculate the transformation matrix, the first step is to inscribe a frame to each joint. Each frame will have a direct relation with rotation and translation to the frame before, allowing, therefore, the computation of the relation of the frame between the end effector and the base. Figure 6 represents the indexation of the frames in the diagram.

With the frame indexing established for each joint, construction of the Denavit-Hartenberg table becomes feasible,

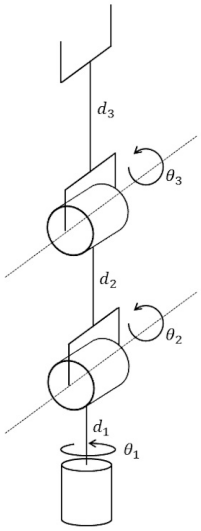


Fig. 5: Kinematics diagram of the three-degree-of-freedom revolute robot.

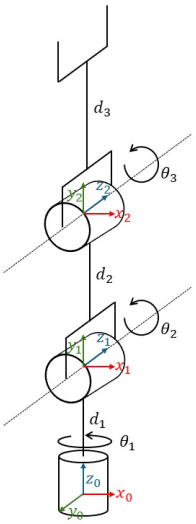


Fig. 6: Indexation of the frames for each joint.

incorporating parameters for each joint as per the standard Denavit-Hartenberg method, as illustrated on Table I.

TABLE I: Parameterization of the robot according to the standard Denavit-Hartenberg method.

i	α	**a**	**d**	θ
1	90º	0	d_1	θ_1
2	0º	d_2	0	θ_2
3	0º	d_3	0	θ_3

The computation of the transformation matrices for the relationship between each frame becomes possible. Each transformation matrix is a 4x4 matrix comprising a 3x3 rotation matrix around the x and z axes, along with a 3x1 translation matrix along the same axes. Following the determined parameters, the general transformation matrix calculation proceeds:

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

$$\mathbf{T_i^{i-1}} = \left[\begin{array}{ccc|c} & R & & T \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{2}$$

$$\mathbf{T_i^{i-1}} = \left[\begin{array}{ccc|c} cos(\theta) & -cos(\alpha)sin(\theta) & sin(\alpha)sin(\theta) & acos(\theta) \\ sin(\theta) & cos(\alpha)cos(\theta) & -sin(\alpha)cos(\theta) & asin(\theta) \\ 0 & sin(\alpha) & cos(\alpha) & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3}$$

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 following equation:

$$\mathbf{T_3^0} = \mathbf{T_1^0} \times \mathbf{T_2^1} \times \mathbf{T_3^2} \tag{4}$$

To extract the position of the end effector from the transformation matrix, the translation matrix, positioned between the first row and third row of the fourth column, embedded in the transformation matrix was extracted.

2) Inverse Kinematics: After computing the forward kinematics of the robot, the inverse kinematics was determined. 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 multiple or no solutions. Singularities denote critical points where the robot's manipulator loses one or more degrees of freedom, 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 numerically, employing iterative methods for approximate solutions. Both approaches have their merits and limitations, requiring thoughtful consideration based on the specific requirements and complexity of the robotic system [7].

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 angles of the shoulder and elbow; 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 aid in visualizing the geometric calculations for this angle, refer to Figure 7. This illustration provides a top-down view of the robot's configuration.

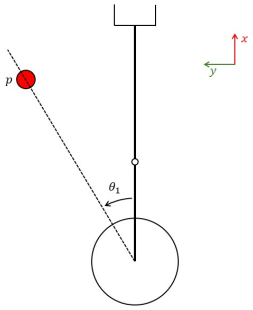


Fig. 7: Top-down schematic of the robot configuration.

As shown in Figure 7, the idea is to align the shoulder and elbow with the same 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 follows:

$$\theta_1 = \text{atan2}(y,x) \tag{5}$$

After calculating the base angle, the elbow angle was determined next. To visualize the calculation of this angle, Figure 8 illustrates the robot reaching a given point in three-dimensional space.

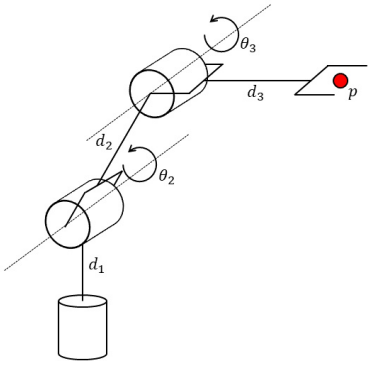


Fig. 8: Side-view schematic of the robot configuration.

As can be seen in Figure 9, 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.

Thus, utilizing the triangle formation and the cosine rule, the elbow angle, expressed as θ_3, can be determined by:

$$x^2 + y^2 + (z-d_1)^2 = d_2^2 + d_3^2 - 2d_2d_3\cos(-180+\theta_3) \tag{6}$$

$$\theta_3 = \pm\text{acos}\left(\frac{x^2+y^2+(z-d_1)^2 - d_2^2 - d_3^2}{2d_2d_3}\right) \tag{7}$$

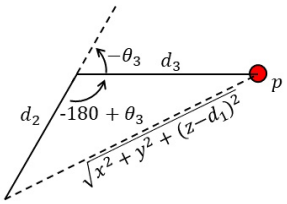


Fig. 9: Triangle formed by the relation of the shoulder and elbow links.

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. From these two triangles, two new angles, α and β, can be extracted. Figure 10 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:

$$\alpha = \text{atan2}\left(\frac{z-d_1}{\sqrt{x^2+y^2}}\right) \tag{8}$$

$$\beta = \text{atan2}\left(\frac{d_3\sin(-\theta_3)}{d_2+d_3\cos(-\theta_3)}\right) \tag{9}$$

$$\theta_2 = \alpha+\beta = \text{atan2}\left(\frac{z-d_1}{\sqrt{x^2+y^2}}\right) + \text{atan2}\left(\frac{d_3\sin(-\theta_3)}{d_2+d_3\cos(-\theta_3)}\right) \tag{10}$$

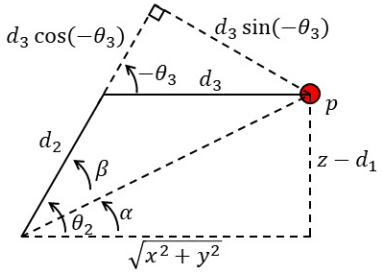


Fig. 10: Right triangles formed to calculate the shoulder angle.

B. Robot Movements

Path planning is a crucial aspect when creating routines for the robot to follow. Consequently, the library implements two main movements: linear movement and joint movement. These movements are standard in the industry for controlling robot manipulators[8]. Linear movements, as indicated by their name, involve linear trajectories in three-dimensional space. They are essential for operations requiring precise movements when approaching a desired point. In contrast, joint movement offers a less precise approach to reaching a

given point in space, without consideration for the specific path taken. This implementation is typically chosen when the robot's path is not critical to the routine it is executing.

1) Linear Movement: To execute this movement, two primary steps are employed: first, linearly interpolating between the current position of the end effector and the desired position; second, computing the inverse kinematics for each interpolated position and updating the servo angles accordingly. The pseudo-code in Algorithm 1 shows the implementation details of the linear movement function.

Algorithm 1 Linear Movement Function

Require: x, y, z, $velocity$
1: $steps \leftarrow$ distance/resolution
2: $time\_step \leftarrow$ (distance/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**

2) Joint Movement: Implementing joint movement for path planning is relatively simpler compared to linear movement. The process involves two key steps: first, calculate the current and desired angles of the servos; second, increment the angles of each servo to reach its desired position. The pseudo-code provided in Algorithm 2 outlines the necessary code for this movement.

Algorithm 2 Joint Movement Function

Require: x, y, z, $velocity$
1: $steps \leftarrow$ distance/resolution
2: $time\_step \leftarrow$ (distance/velocity)/steps
3: $step\_factors \leftarrow$ (wanted_angle − current_angle)/steps
4: **for** $i \leftarrow 0$ **to** $steps$ **do**
5: $angles \leftarrow$ current_angle + step_factors $\times (i+1)$
6: **set servo angles** to angles
7: **delay** for time step
8: **end for**

IV. RESULTS

As outlined in Section I, the initial stage involved implementing the robot's kinematics using MATLAB and a robotics toolbox developed by Peter Corke [4]. By inputting the derived equations into the kinematic functions, the robot's movements were accurately represented. To validate the accuracy and reliability of these functions, a cycle was constructed to iterate through the three angles with a step of 1.8º, serving as inputs for the forward kinematics function. The resulting positions were then used as input for the inverse kinematics function to compute the angles for each frame of the robot. To confirm the position reached by the end effector, a sphere was represented at each position used as input for the forward kinematics. Figure 11 illustrates the robot's representation in MATLAB following a trajectory of points.

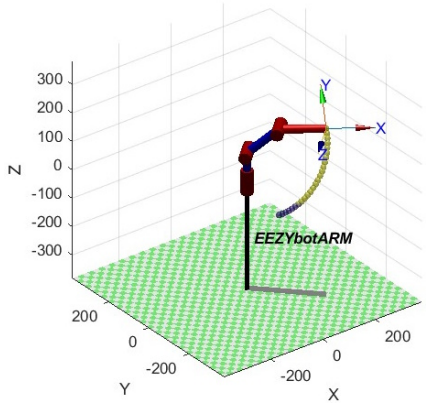
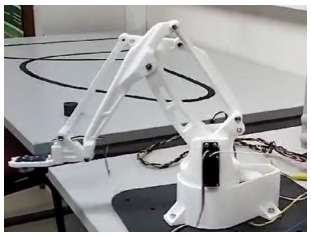


Fig. 11: Visualization of the robot's kinematics functions output, depicting the achieved positions of the robot and representing them as spheres in MATLAB.

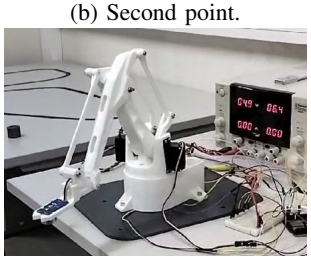
After validating the robot's kinematics, a library was developed for the ESP32 microcontroller. This enabled the integration between the hardware setup and the C-based software. Subsequently, the robot underwent testing to evaluate its ability to follow a programmed path. Navigation through the specified positions was successful, as depicted in Figure 12, where the robot reaches the four programmed points during the test.

(a) First point.



(b) Second point.

(c) Third point.



(d) Fourth point.

Fig. 12: Robot reaching different points during a programmed task.

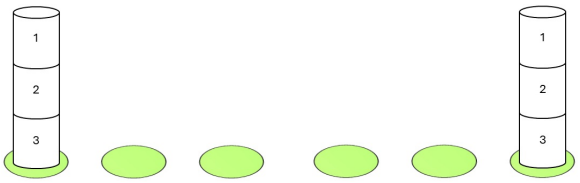
V. PROPOSED CHALLENGES

As this robot is intended for use in an academic context, particularly for master's level students, it will serve as a teaching aid platform for practical learning and assessment.

To facilitate this, two challenges were proposed, each one assigned to a group of students randomly. These tasks are the Tower of Hanoi and Snake challenges, designed to test the student's understanding and application of robotics concepts. The following sections provide detailed explanations of each challenge.

A. Challenge 1 - Tower of Hanoi

The Tower of Hanoi presents a mathematical challenge comprising three rods and a set of disks with varying diameters. These disks are arranged initially in decreasing size, with the smallest disk atop and the largest at the bottom, on the first rod. The objective of the challenge is to transfer the entire stack of disks from the first rod to the third, adhering to the rule that a larger disk cannot be placed atop a smaller one. In this adaptation, the challenge transforms into an advanced pick-and-place system, utilizing three cylinders labeled from '1' to '3' for differentiation. Corresponding to the Tower of Hanoi setup, the cylinder labeled with the lowest number represents the disk with the smallest diameter, and vice versa. Figure 13 provides a visual representation of the challenge posed to the students.



(a) Initial configuration of the Tower of Hanoi challenge.

(b) Final configuration of the Tower of Hanoi challenge.

Fig. 13: Illustration of the Tower of Hanoi Challenge.

B. Challenge 2 - Snake

The snake challenge entails moving three cylinders arranged side by side, without any gaps, from a starting point to an ending point. During the transportation process, the cylinders must remain contiguous, without any separation between them, thus mimicking the movement pattern of a snake, as shown in Figure 14.

(a) Initial configuration of the snake challenge.

(b) Final configuration of the snake challenge.

Fig. 14: Illustration of the snake Challenge.

VI. CONCLUSIONS AND FUTURE WORK

This work aims to aid students' comprehension of mechanical and kinematic principles in robotics through hands-on experience and programming of a manipulator robot. The Arduino library developed for this purpose simplifies the generation of robot instructions. The project aims to elucidate the foundational aspects of robot programming, particularly how robot programs are designed in industrial settings. By aligning with the RAPID language used in industrial settings, students are better prepared for real-world applications in robotics programming.

Moreover, the project offers practical tasks like the Tower of Hanoi and Snake challenges, which effectively allow students to apply and evaluate their skills in a final laboratory exercise. These challenges not only reinforce theoretical knowledge but also promote problem-solving and critical thinking skills essential in the robotics field.

Looking ahead, future work will focus on the development of hardware in the loop, enabling offline programming of the "EEZYbotARM Mk2" robot. This advancement will empower students to test their code efficiently before implementing it on the physical robot, further enhancing their learning experience and proficiency in robotics programming. By continually refining and expanding upon these educational tools and methodologies, this project aims to foster well-equipped students able to navigate modern robotics in industrial settings.

ACKNOWLEDGMENT

The authors are grateful to the Foundation for Science and Technology (FCT, Portugal) for financial support through national funds FCT/MCTES (PIDDAC) to CeDRI (UIDB/05757/2020 and UIDP/05757/2020) and SusTEC (LA/P/0007/2021). Laiany Suganuma Brancaliao and Mariano Alvarez thank FCT for the PhD Grant 2023.01441.BD and 2023.01113.BDANA.

REFERENCES

[1] C. Franciscone. (2018, February 20). EEZYbotARM Mk2. Retrieved from http://www.eezyrobots.it/eba_mk2.html

[2] Handson Technology. WeMOS D1 R32 ESP32 Wi-Fi and Bluetooth Board. Retrieved from https://handsontec.com/dataspecs/module/ESP/WeMos%20D1%20R32.pdf

[3] Jamie C, University of Auckland. (2021, 27 August). D1 R32 – ESP32. Retrieved from https://designtech.blogs.auckland.ac.nz/d1-r32-esp32/

[4] P. I. Corke, Robotics, Vision & Control: Fundamental Algorithms in MATLAB, 2nd ed. Springer, 2017. ISBN 978-3-319-54413-7.

[5] P. I. Corke, A Simple and Systematic Approach to Assigning Denavit–Hartenberg Parameters, in IEEE Transactions on Robotics, vol. 23, no. 3, pp. 590-594, June 2007, doi: 10.1109/TRO.2007.896765.

[6] C. s. g. Lee and M. Ziegler, Geometric Approach in Solving Inverse Kinematics of PUMA Robots, in IEEE Transactions on Aerospace and Electronic Systems, vol. AES-20, no. 6, pp. 695-706, Nov. 1984, doi: 10.1109/TAES.1984.310452.

[7] Kucuk, Serdar & Bingul, Z.. (2006). Robot Kinematics: Forward and Inverse Kinematics. 10.5772/5015.

[8] S. Dietrich. (2022, 29 Decemver). Robot Motion Command Types: Understanding Linear, Joint, and Arc Movement. Retrieved from https://control.com/technical-articles/robot-motion-command-typesexplained/