EEC493 - Introduction to Robotics

Final Project Report

Revolute-Prismatic Robotic Manipulator Controlled by Mobile Application

AbdElrhman Mamdouh Khalil	19015935
Amr Ramadan Agamy	19016110
Ali Emad Abdo	19016028
Ahmed Elgohary	19015328
Mariam Ahmed Fathy	19016627
Mostafa Sayed Ahmed Taha	19016660
Marwan Mohammed Ahmed	19016615
Mohamed Rizq Amin	19016365
Nada Attia Eid Attia	19016780
Ahmed Roshdi	18010212

May 20, 2024(Spring)

1 Introduction

Robotic manipulators play a crucial role in various fields, from industrial automation to delicate surgical procedures. Their ability to perform tasks with precision and repeatability makes them a vital tool in modern technology. This report delves into the design and analysis of a simple yet effective robotic manipulator – the Revolute-Prismatic (RP) manipulator.
The RP manipulator utilizes two fundamental joint types: revolute and prismatic. A revolute joint allows for rotational movement about a single axis, while a prismatic joint facilitates linear motion along a designated axis. By combining these joint types, the RP manipulator achieves a balance between simplicity and functionality.
This report explores the design considerations, kinematic analysis, and potential applications of this basic RP manipulator. We will examine the manipulator's workspace, dexterity, and limitations. The analysis will provide valuable insights into the capabilities of this manipulator and its suitability for various tasks.

2 Objective and Applications

The design of this RP manipulator prioritizes both simplicity and functionality for industrial applications. By utilizing only revolute and prismatic joints, we aim to achieve a manipulator that's easy to maintain and operate. However, this simplicity shouldn't come at the cost of limited movement. The manipulator should offer precise and controllable motion within a well-defined workspace. Additionally, the end effector needs to be adaptable to handle various industrial objects and materials. These design objectives translate into several potential applications. The RP manipulator's precise movements and controlled reach make it suitable for pick-and-place tasks in assembly lines, and efficiently transferring components and parts.

3 List of Components and budget

The used components:

- ESP32 WROOM - DA (500 EGP)
- MG995 Servo Motor x2 (400 EGP)
- Electromagnet (350 EGP)
- Wires (1 EGP)
- Breadboard (40 EGP)
- Relay Module (90 EGP)
- Mechanical Fabrication (150 EGP)
- USB cable (50 EGP)

Total Cost = 1590 EGP

4 Mechanical Design

This section explores the mechanical design aspects of the RP robotic arm constructed primarily from wood and PVC pipes, detailing the materials, construction methods, and mechanical considerations involved in the design process.

4.1 Materials and Components

The robotic arm is designed using readily available and inexpensive materials:

- **Wood:** Used for the base and structural components, providing rigidity and support.
- **PVC Pipes:** Utilized for the arm segments due to their lightweight, ease of manipulation, and cost-effectiveness.
- **Acrylic fixation:** fixes the PVC pipe to the servo rotor.
- **Fasteners:** Screws, zip ties, nuts, and bolts for assembling wooden and PVC parts.
- **Servomotors:** Provide actuation and control of the arm's movements.

4.2 Design Considerations

The mechanical design of the robotic arm encompasses several critical factors to ensure functionality, stability, and ease of construction.

4.2.1 Structural Integrity

- **Base**: A sturdy wooden base ensures stability. A rectangular is preferred for better balance.
- **Arm Segments**: PVC pipes are chosen for their balance between strength and weight. The pipes are cut to appropriate lengths for the internal, and external segments.

4.2.2 Joints and Mobility

- **Rotational Joint**: The arm features a rotational joint at the shoulder.
- **Prismatic Joint**: The arm features a prismatic joint at the end effector.
- **Degrees of Freedom**: Typically, the arm is designed with two degrees of freedom (DOF).

4.2.3 Actuation

- **Servomotors**: Chosen for their precise control and ease of integration. Each joint is equipped with a servomotor, selected based on the required torque and load-bearing capacity.
- **Mounting**: Motors are mounted on wooden brackets attached to the joints, ensuring they are secure and correctly aligned.

4.3 Construction Process

The construction process involves several steps to ensure precise assembly and alignment:

1. **Cutting Materials**: Wood and PVC pipes are measured and cut to specified dimensions.
2. **Assembly of the Base and Supports**: The wooden base and vertical supports are assembled first, ensuring a stable foundation.
3. **Joint Assembly**: Bearings are fitted into the joint supports, and PVC arm segments are connected using fasteners.
4. **Motor Installation**: Servomotors are mounted, and their arms are attached to the joints.

4.4 Testing and Calibration

Post-construction, the arm undergoes rigorous testing to calibrate motor movements and ensure smooth operation:

- **Range of Motion**: Each joint's range of motion is tested and adjusted.
- **Load Testing**: The arm's ability to lift and manipulate objects is evaluated.
- **Control Accuracy**: The precision of the microcontroller's commands to the servomotors is fine-tuned.

4.5 Modifications and Improvements

While the initial design is geared towards simplicity and educational use, several improvements can enhance functionality:

- **Material Upgrades**: Incorporating metal reinforcements or higher-grade bearings.
- **Additional Degrees of Freedom**: Adding more joints for complex tasks.
- **Advanced Control Systems**: Integrating sensors for feedback and more sophisticated control algorithms.

5 Electrical Components

Electrical components are crucial in the design and operation of robotic manipulators, with servo motors and microcontrollers being key elements. Servo motors, like the MG995, are essential for providing precise control over joint movements due to their ability to accurately position their output shaft. These servos are typically controlled via Pulse Width Modulation (PWM) signals. The Arduino IDE (Integrated Development Environment) facilitates the programming of the ESP32 microcontroller which is often used to generate the necessary PWM signals for servo control.

5.1 ESP32 Connection

The ESP32 is a series of microcontroller chips produced by Espressif Systems in Shanghai. It is available in a number of low-cost modules. Its chip comes with 48 pins with multiple functions. Not all pins are exposed in all ESP32 development boards and some pins cannot be used.

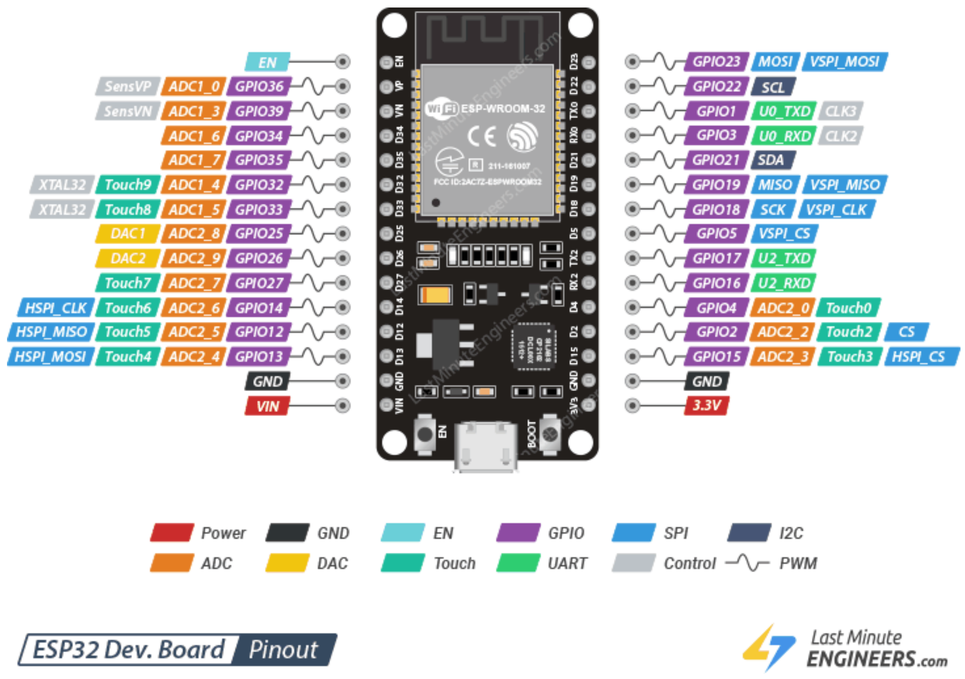


Figure 1: ESP32 WROOM Pinout.

The ESP32 microcontroller acts as the heart of the RP manipulator, controlling its movements and user interaction. Here's how it tackles these tasks:

- **Controlling the Show:** The ESP32's dual-core processor and high clock speed provide the muscle for real-time control. This translates to smooth and precise movements of the manipulator's joints based on user commands.

- **Mobile App Maestro:** Leveraging the ESP32's built-in WiFi module, a mobile app can be developed to wirelessly control the manipulator. This app, running on a smartphone or tablet, acts as the user interface, sending control signals to the ESP32 via WiFi.

- **Powering the Duo:** The ESP32 can directly power the two servo motors controlling the revolute joints of the manipulator. Using libraries readily available in the ESP32 development environment, the ESP32 can generate the necessary PWM signals to control the position of each servo motor.

5.2 Servo Motors

Servo motors are widely used in robotic applications due to their precise control of angular position, velocity, and acceleration. The MG995 servo motor, in particular, is a popular choice in robotics projects because of its robust performance and affordability. It features a metal gear train, high torque, and good speed, making it suitable for various manipulator configurations.

Figure 2: MG995 Servo Motor

The RP robotic manipulator combines a revolute joint and a prismatic joint to achieve two degrees of freedom. The revolute joint allows rotation around a fixed axis, while the prismatic joint provides linear motion along an axis. This configuration is versatile, enabling a wide range of movements ideal for tasks requiring both rotational and linear positioning.

Application of MG995 Servo Motors
In the context of a revolute-prismatic robotic manipulator, two MG995 servo motors can be employed to actuate the revolute and prismatic joints:

1. **Revolute Joint Actuation:**
    - **Motor Placement:** One MG995 servo motor is mounted at the base of the manipulator to control the revolute joint.
    - **Function:** This servo motor provides the necessary torque to rotate the arm around a fixed axis, allowing the end effector (Elctromagnet) to sweep through an arc.
    - **Control:** By varying the pulse width of the control signal sent to the MG995, the angular position of the revolute joint can be precisely controlled. This enables the manipulator to reach different angular positions as required by the task.
2. **Prismatic Joint Actuation:**
    - **Motor Placement:** The second MG995 servo motor is coupled with a rack and pinion mechanism to drive the prismatic joint.
    - **Function:** This configuration translates the rotational motion of the servo into linear motion, enabling the end effector to extend or retract along a straight line.
    - **Control:** Similar to the revolute joint, the linear position of the prismatic joint is controlled by adjusting the pulse width modulation (PWM) signal sent to the servo. This

allows for precise control of the linear displacement, essential for tasks that require specific positioning.

5.3 ESP32 WiFi Server Setup with Arduino IDE

There are several dependencies to program the circuit:

- ESP32 Board jason file, it is downloaded from ESP32 package
- ServoESP32 library

After installing the required board and libraries, It is the code time, The ESP32 can be programmed using many different development environments. Code can be written in C++ (like the Arduino) or in MicroPython, where the Arduino is used to program it.

As a communication system the mobile app is considered the client and the ESP is the server. For a client to be able to reach the HTTP server, the router is not needed but rather to the WiFi network hosted by the ESP32, it works as the access point .

The possibility of setting an HTTP server on the ESP32 working as a soft AP is very useful since in real application scenarios, an IoT device may be deployed in a WiFi network to which the credentials are not known in code compile time.

The ESP32 operates as a soft AP when connected for the first time, starting a HTTP server that serves as an configuration HTML webpage for the user to input the name and password of the WiFi network to which the device should connect to be able to reach the Internet and operate.

ESP32 WiFi Server Setup

- `WiFiServer server(80);` This class creates a server that listens for incoming connections on the HTTP port 80.
- `WiFi.softAP(ssid, password);` To start the soft AP, we simply need to call the `softAP` method of the `WiFi` external variable (this is the same variable we use to connect the ESP32 to a WiFi network). This method receives as first input the name of the WiFi network we want to set and as second input its password. Setting a password is not mandatory, and we could omit it if we wanted our Access Point to be open.
- `Serial.begin(115200);` Moving on to the `setup` function, start it by opening a serial connection needed to print the IP of the ESP32 for the client to be able to reach it.

Finally, we will need to know the ESP32 IP in order for the client connected to its network to be able to send requests to it. We can obtain the IP by calling the `softAPIP` method on the same `WiFi` variable.
In the `void setup` function, the function will try to connect to WiFi. This process executes in a loop, which means it runs until there is a connection to WiFi.

In the `void loop`, check if a client has connected. Wait until the client sends some data and performs tasks according to the input. Then the connection is disabled after it is performed. The tasks are the requests received from the client (mobile app). For example, the client can command the robot to reach an object to pick it up, then move back to the place to release it, and finally return to the home position.

6 Manipulator Schematic

The design and simulation of a revolute-prismatic robotic manipulator can be efficiently performed using MATLAB Simulink and the Robotics Toolbox. This setup allows for a comprehensive schematic of the manipulator, integrating the dynamic modeling and control aspects within a graphical environment. By utilizing the Robotics Toolbox, various components such as revolute and prismatic joints can be accurately modeled and connected. The simulation enables the real-time visualization of the manipulator's motion and performance.
The outputs, such as joint angles, velocities, and end-effector position, can be monitored using Simulink's Scope block. This real-time feedback is essential for validating the manipulator's behavior and tuning control parameters. Notably, the position of the end effector is exactly similar to the position generated from the Trapezoidal Velocity Profile Trajectory block, ensuring that the robot operates as intended. The integration of these tools provides a robust platform for developing, testing, and refining robotic systems before physical implementation.

6.1 Simulation

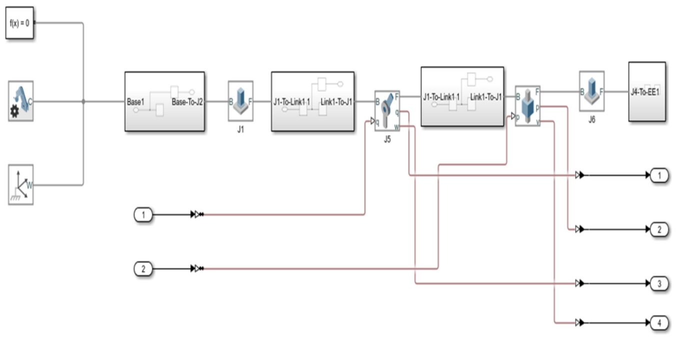


Figure 3: Rigid Body Diagram

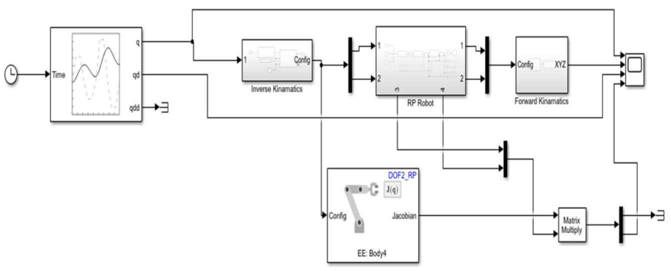


Figure 4: Robot System

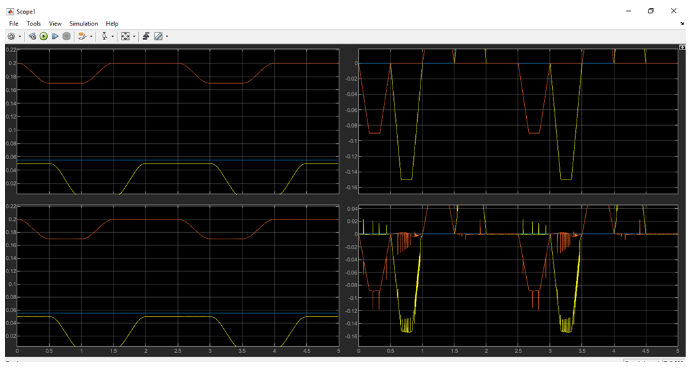


Figure 5: Scope Output

Comment : The position of the end effector is exactly similar to the position generated from the Trapezoidal velocity profile trajectory block so the robot works regularly.

6.2 3D Robot Model

The Mechanism consists of a base which is the yellow block and one revolute joint which is the green cylinder connected to the prismatic joint that moves along the red link, the end effector which is the blue cube is fixed at a distance equal to half of the red link and move along this link from half it's length to the end.

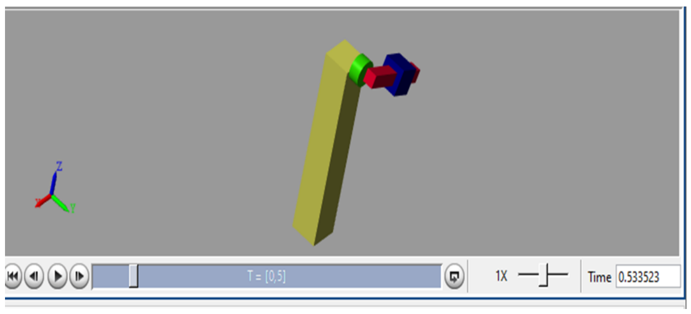


Figure 6: Robot Model

7 Forward and inverse Kinematics

7.1 DH Parameter

Forward kinematics involves calculating the position and orientation of the end-effector of a robotic manipulator based on the known joint parameters. The Denavit-Hartenberg (DH) parameters provide a standardized method to represent the spatial relationships between adjacent links of a manipulator. The DH parameters consist of four parameters for each link i:

- Θ_1 : The angle of rotation of joint 1
- D_2 : the distance from the reference frame to the frame of prismatic
- L_2: the distance from frame of prismatic to the End Effector.

Link i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	90	d_2	0
3	0	0	L_2	0

Using these parameters, we can define the transformation matrix for each link i as follows:

$$^{0}T_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -\mathrm{d}_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{L}_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{0}T_2 = {}^{0}T_1 * {}^{1}T_2 = \begin{bmatrix} C1 & 0 & S1 & \mathrm{d}_2\ 1 \\ S1 & 0 & -C1 & -\mathrm{d}_2\ \mathrm{C1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The overall transformation matrix $^{0}T_3$ from the base frame to the end-effector frame is obtained by multiplying the individual transformation matrices of each link:

$^{0}T_3 = {}^{1}T_2 * {}^{2}T_3 =$

$$\begin{bmatrix} C1 & 0 & S1 & \text{d}_2\text{ S1} + L_2\text{ S1} \\ S1 & 0 & -C1 & -\text{d}_2\text{ C1 - }L_2\text{ C1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C1 & 0 & S1 & (\text{d}_2 + L_2\text{)S1} \\ S1 & 0 & -C1 & -(\text{d}_2 + L_2)\text{ C1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.2 Inverse Kinematics

Inverse kinematics involves determining the joint parameters that provide a desired position and orientation of the end-effector. Unlike forward kinematics, which has a straightforward calculation, inverse kinematics is often more complex and may have multiple solutions or require numerical methods to solve.

The general approach to inverse kinematics involves solving the nonlinear equations derived from the forward kinematics. For a manipulator with 2 joints, we set the desired position and orientation of the end-effector as ${}^0T_3$. The goal is to find the joint parameters θ_1, d_2
such that:

$$\text{X}= (d_2 + L_2\text{)S1}$$
$$\text{Y}= \text{-}(d_2 + L_2)\text{ C1}$$
$$\tan\theta_1 = \text{ - }\frac{x}{y}$$
$$\theta_1 = \arctan\frac{-x}{y}$$
$$\text{X2}= (d_2 + L_2\text{ })^2 S1^2,$$
$$\text{Y2}= (d_2 + L_2\text{ })^2 C1^2$$
$$X^2 + Y^2 = (d_2 + L_2\text{ })^2,$$
$$d_2 = \sqrt{\text{x}^2 + y^2}\text{ - }L_2$$

7.3 Jacobian

The Jacobian matrix is a fundamental concept in robotics that relates the velocities of the joints to the linear and angular velocity of the end-effector. For a manipulator with 2 joints, the Jacobian is defined as:

$$J_\omega = \begin{bmatrix} {}^{o}R_0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & {}^{o}R_1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$J_v = \begin{bmatrix} \frac{dx}{d\Theta 1} & \frac{dx}{d(d2)} \\ \frac{dy}{d\Theta 1} & \frac{dy}{d(d2)} \\ \frac{dz}{d\Theta 1} & \frac{dz}{d(d2)} \end{bmatrix} =$$

$$\begin{bmatrix} (\text{d}_2 + L_2\text{) C1} & S1 \\ (\text{d}_2 + L_2\text{) S1} & -C1 \\ 1 & 0 \end{bmatrix}$$

$$\text{J} = \begin{bmatrix} \text{J}_v \\ \text{J}_\omega \end{bmatrix} =$$

$$\begin{bmatrix} (\mathrm{d}_2 + \mathrm{L2}\ )\ \mathrm{C1} & S1 \\ (\mathrm{d}_2 + \mathrm{L2}\ )\ \mathrm{S1} & -C1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

8 Mobile Application

Here we present our app that allows users to control the movement of the robotic arm to three predefined positions - home position, target one position, and target two position. Additionally, it enables users to manipulate the end effector to collect and release metal objects at the designated positions.
App Functionality: The primary functionality of the app includes:

1. **Robotic Arm Control:** Users can control the movement of the robotic arm to three predefined positions:
   - **Home Position:** Initial position of the robotic arm.
   - **Target One Position:** Position designated for collecting metal objects.
   - **Target Two Position:** Position designated for releasing collected metal objects.
2. **End Effector Manipulation:** The app allows users to control the end effector attached to the robotic arm. The end effector is an electric magnet used for collecting metal objects. Users can:
   - **Collect:** Activate the electric magnet to attract and collect metal objects at the target one position.
   - **Release:** Deactivate the electric magnet to release the collected metal objects at the target two positions.
3. **Connection via Wi-Fi:** The app facilitates communication with the robotic arm through Wi-Fi connectivity. It sends HTTP requests containing the target name when the user pushes the button corresponding to the desired target position. This allows for seamless control of the robotic arm's movement and end effector manipulation remotely.

Development Process: The development of the app was carried out using MIT App Inventor, a web-based platform for creating Android applications. The following steps were involved in the development process:

- **User Interface Design:** The user interface (UI) of the app was designed to be intuitive and user-friendly. It includes buttons for controlling the movement of the robotic arm to different positions (home, target one, and target two), as well as buttons for activating the collect and release functionalities of the end effector.

- **Block-Based Programming:** MIT App Inventor utilizes a block-based programming interface, making it accessible for both novice and experienced developers. The app's functionality was implemented using blocks to define the behavior of each UI component and handle user interactions.

- **Integration with Robotic Arm:** The app was integrated with the robotic arm using Wi-Fi communication. HTTP requests are sent from the app to the robotic arm's control system to trigger movement to the desired positions and activate the end effector for collecting and releasing metal objects.

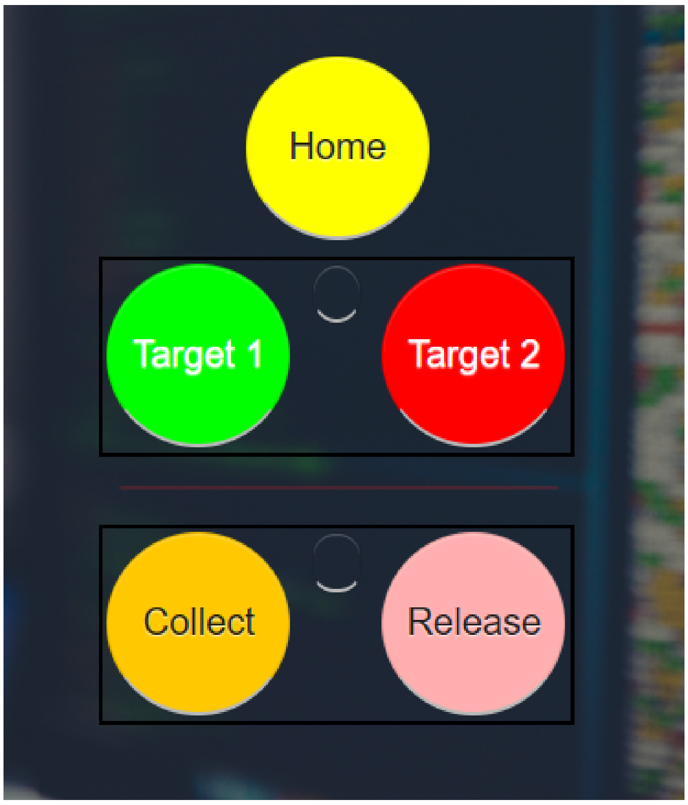
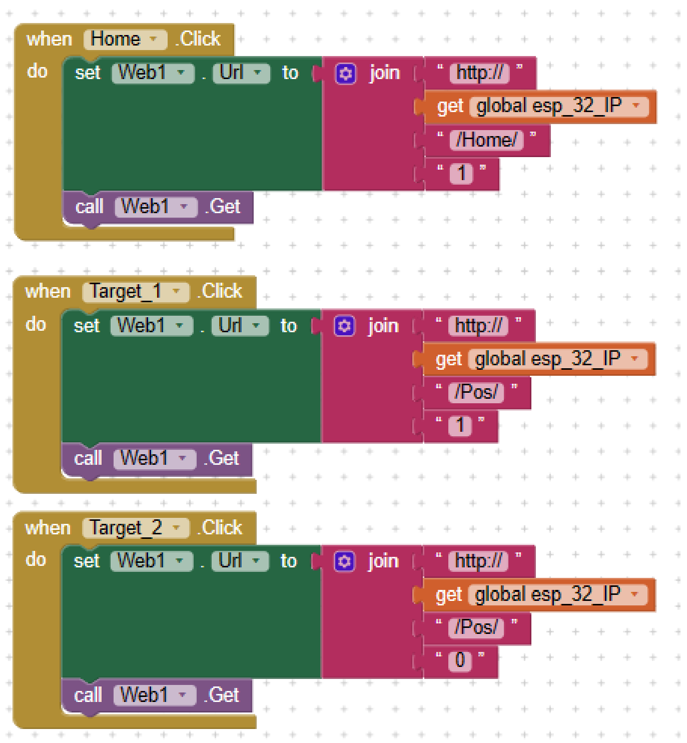


Figure 7: App Interface

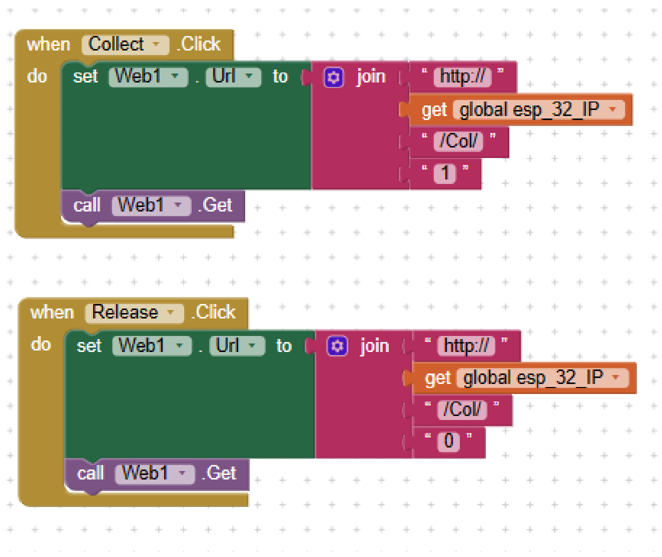


```
when Home .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Home/ "
                             " 1 "
    call Web1 .Get

when Target_1 .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Pos/ "
                             " 1 "
    call Web1 .Get

when Target_2 .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Pos/ "
                             " 0 "
    call Web1 .Get
```

Figure 8: Position Control



```
when Collect .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Col/ "
                             " 1 "
    call Web1 .Get

when Release .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Col/ "
                             " 0 "
    call Web1 .Get
```

Figure 9: Collection control

9 Risks and Challenges

9.1 Mechanical Risks

- Fragility: As the arm is made of light materials.
- Wear, and corrosion.
- Performance deterioration: As the joints tend to drift over time.

9.2 Electromagnet Power

One of the challenges in the project is that the electromagnet operates at 5V, while the output pins of the ESP32 microcontroller provide only 3.3V. This voltage mismatch means the ESP32 cannot directly power the electromagnet, as the lower voltage would not be sufficient to activate it. To address this issue, a relay module is used as an intermediary to control the electromagnet.

10 References

1. Electromagnet Datasheet: `https://www.mouser.com/datasheet/2/744/Seeed_101020073-1217554.pdf`
2. Relay Module (2 Channel) 5V with High/Low Level Trigger Selector: `https://makerselectronics.com/product/relay-module-2-channel-5v-with-high-low-level-trigger-selector`
3. Lecture Slides
4. ESP32 Datasheet: `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`
5. MIT App Inventor Tutorial: `https://appinventor.mit.edu/explore/ai2/tutorials.html`
6. MG995 Servo Datasheet: `https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf`