بسم الله الرحمن الرحيم

An-Najah National University
Faculty of Engineering and Information Technology
Computer Engineering Department

Graduation Project 2

Programmable Robotic Arms

Done by:
Naser Shaer, Kareem Yaqoup

Supervisor: **Dr. Loai Malhis**

Submitted in partial fulfillment of the requirements for the
Bachelor degree in Computer Engineering

2024

Acknowledgment

We would first like to thank the Almighty from the bottom of our hearts. Without His blessings, our path would not have taken us to the successful completion of this project at every turn.

We really appreciate our parents and other family members. Your unwavering support and encouragement have kept us motivated and inspired throughout our journey. Your warm and encouraging words have given us strength and stability and laid a strong foundation.

We also thank Dr. Loai Malhis, our supervisor, for all of your help and thoughtful recommendations. Your accurate observations, perceptive comments, thought-provoking discussions, and helpful criticism have significantly increased the potential of our initiative.

We are especially grateful to the Computer Engineering Department of An-Najah National University for their ongoing assistance.

Disclaimer

The two members of the team, Naser Shaer and Kareem Yaqoub, are An-Najah University students studying computer engineering. worked together to finish this paper. Any inaccuracies found within are fully the responsibility of the writers. The thoughts, recommendations, conclusions, and views presented in this work are entirely the authors' own and do not represent An-Najah National University.

Table of Contents

Table of Figures:

Abstract

The Programmable Robotic Arm is a smart, 6 Degree of Freedom (DOF) robotic arm that can be programmed using two interfaces. It features a special custom compiler with simple set instructions, allowing for sequential coding, control statements, loops, and a variety of built-in functions which enables on-fly programming/testing. Additionally, it supports a mobile application that enables of the robotic arm through a simple interface, with camera as it enabling quality checking and live video streaming, also it offers object detection that can be selected from the mobile so that arm can automatically grips it then give back the control to the user. The robotic arm is equipped with prebuilt sensors and supports sensor expansion via digital and analog external fixed inputs. This flexibility allows for the creation of diverse sequences based on different sensors and inputs. This prototype follows industrial robotic arms version 4 which allows controlling the robotic arm wirelessly through a computer or mobile, the arm also has intelligence which allow it to adapt to object size, weight, and also it can detect objects that are out of range.

1 Introduction

1.1 Problem Statement

In modern assembly lines, adjusting or reprogramming robotic arms for new tasks or products often requires a specialist to be program it on-site. This will lead to costly downtime as the entire assembly line have to be paused during programming or configuration updates. Moreover, getting new robotic arms that can adapt to varying products or installing a new assembly line for new product is expensive and will need additional devices for quality check. These inefficiencies increase production costs and limit flexibility in rapidly changing manufacturing environments.

1.2 Problem Objective

Creating a programmable robotic arm with advanced adaptability and a user-friendly interface to reduce the costs and the need of on-site specialist. By enabling remote control and viewing through a mobile application, the robotic arm can reflect these adjustment on-fly without needed to reboot nor to pause the assembly line. Furthermore, by integrating automatic object detection and sensor expansion capability, the robotic arm will be flexible enough to adapt to different product sizes, classes. In addition, the custom compiler allows the integration with variety of other softwares or technologies, and also it allows for full control over the low layers of the robotic arms and if provides easy to use interface and instructions which eliminates the need of a specialist. All of those will provide a cost-effective solution for modern assembly lines.

1.3 Scope of work

The scope of this project is to design and develop a 6 Degree of Freedom (DOF) programmable robotic arm that offers high flexibility and cost-effective automation across various industries. While the primarily target of the robotic arm is for assembly line operations, the robotic arm is flexible enough for use in general-purpose automation including warehouse management, logistics, healthcare and agriculture. It will eliminate the need for on-site specialists and minimize downtime by providing fast and easy adapting to new tasks and products. Additionally, the arm can automate repetitive tasks such as object sorting, picking, placing, and quality checking making it a valuable tool for any environment that requires human work automation, improving efficiency, safety, and productivity while reducing costs.

2 Constraints, Standards and Codes and Earlier Work

2.1 Constraints

1. Limited resources for implementing 3D inverse kinematics.
2. Short time available for prototyping and testing.
3. Project must be completed within a 2-month timeframe.
4. Restricted access to the university due to ongoing war.
5. Need to quickly learn new concepts and various microcontrollers (ESP32, Arduino, Raspberry Pi)
6. Limited availability of components in local stores

2.2 Standards / Codes

- **Programming Environment**: Arduino IDE was used for writing and deploying code on the Arduino Mega 2560 and ESP32, using Arduino C.

- **Programming Languages**: Arduino C was used for both Arduino Mega 2560 and ESP32. Python was used for programming the Camera Server, the backend of the mobile application. React/React Native was used for developing the mobile application. Python was utilized for building the custom compiler.

- **Communication Protocols**: Headless SSH communication was employed for programming and controlling the Raspberry Pi 5, following standard secure shell protocols (SSH) for remote access and secure data transmission. Serial Communication was utilized for communicating between the microcontrollers.

- **WIFI Connectivity**: The Raspberry was configured using IEEE 802.11 wireless communication standards to establish Wi-Fi connection for mobile application and camera feedback/control.

- **Motor Control**: The 6 servo motors were controlled via PWM signals.

- **LCD Integration**: External sensors were interfaced using standard I2C and SPI communication protocols.

- **Sensor Integration**: External sensors were interfaced using standard analog/digital input/output.

2.3 Earlier work:

Previous efforts made in the field of robotic arms for assembly lines have primarily focused on specialized and high-cost industrial robots with specific tasks. Giving companies example like ABB and KUKA that make these robots, these robots are programmed using proprietary software and require skilled technicians for on-site setup and programming. Other research has explored using microcontrollers like Arduino and Raspberry Pi to develop lower-cost, open-source alternatives, but these often lack the flexibility and ease of use required for rapid reprogramming and remote control.

In earlier work, projects like the "6DOF Robotic Arm" utilized Arduino platforms for basic motion control using angles of each motor, but they were limited in functionality, particularly in handling complex tasks such as controlling, programming it required knowledge in these microcontrollers.

This project builds on these foundations by incorporating real-time object detection, wireless control via both mobile and computer and adapt to object size and weight, aiming to address the limitations of previous designs in both cost and ease of use.

3 literature Review

The development of robotic arms has been significant recently, back since the early automata created by innovators like Leonardo da Vinci, and extending to modern surgical systems. This advance shows that the technological steps that have transformed applications in both industry and medicine [1]. Over the past years, robotic arms have seen a significant improvement and now is used in many industrial and general fields. Robotic arms operate on several principles like forward and inverse kinematics which study the motion without taking the caused force into account, also degrees of freedom which refers to the number of independent moves, also the concepts of accuracy, repeatability and programmability [2]. The development and integration of robotic arms plays an important role in Industry 4.0. They are increasingly used in digital manufacturing to perform 3D printing, milling and assembly tasks, demonstrating their contribution to enhancing flexibility, precision and efficiency within modern production processes [3].

Inverse kinematic was a significant problem in robotics research, that are many existing methods are usually designed manually and limited to applications with known track movements [4]. Although iterative methods may require massive computational resources and may not always converge, closed-form solutions are precise and efficient [5].

4 Methodology

4.1 Technologies – Components:

4.1.1 IR sensor:

Infrared Sensor was used detect obstacles on both sides (left and right), the arm will pause if an obstacle is present and resume once the obstacle is cleared. Also we used special type of IR sensors for small range detection for ensuring that the grip doesn't hit the object when trying to catch it.

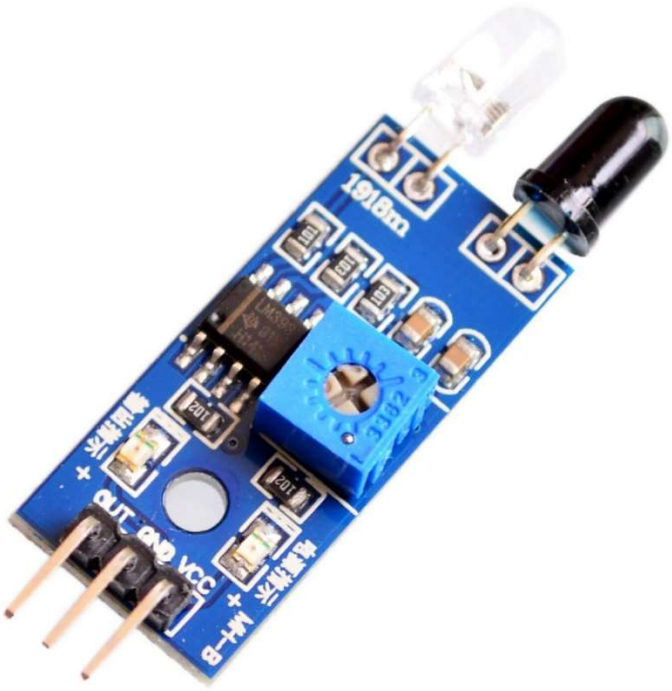


Figure 1: IR sensor

4.1.2 Arduino Mega 2560

The Arduino Mega is a microcontroller board. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 4 UARTs (hardware serial ports), 16 analog inputs, 16 MHz crystal oscillator, a USB connection and a power jack. We used the Arduino for direct control of the servo motors and the sensors (built-in and the expansion ones), it features 54 input/output pins, which makes it suitable for controlling the 6 servo motors and for continuous reading of sensors.

Figure 2: Arduino Mega 2560

4.1.3 Servo Motor MG996R

A servo motor is a rotary actuator that allows precise control of angular position, speed, and acceleration Used it as arm joints, with 180 degree angle, and with approximately up to 10KG max weight to hold. We used the servo motors for high precision movement and we combine both the high speed and high torque motors for best utilization. PWM was used to control the servo motors at different speeds to make synchronization among the 6 motors.

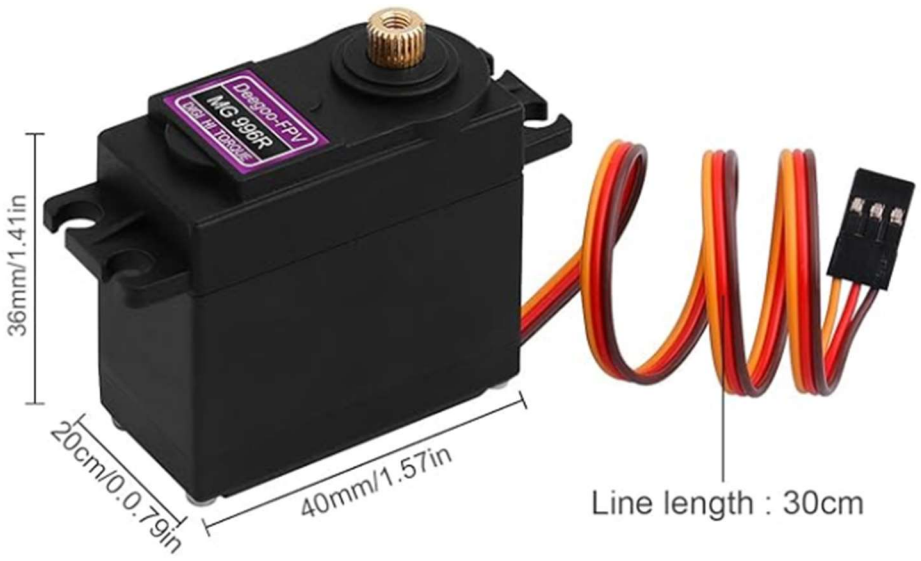


Figure 3: Servo Motor (MG996R)

4.1.4 LCD

LCD (Liquid Crystal Display) is flat panel display which commonly used in screens for many devices and projects. It works by using liquid crystals that modulate light to display images, characters, or data. We used it to display arm current state (Idle, Compiler Mode, Moving, Holding, Camera Mode, Arrows Mode), current position of the robotic arm related to the 3D space, target position to move to, and feedback whether there was an obstacle during the movement or not.

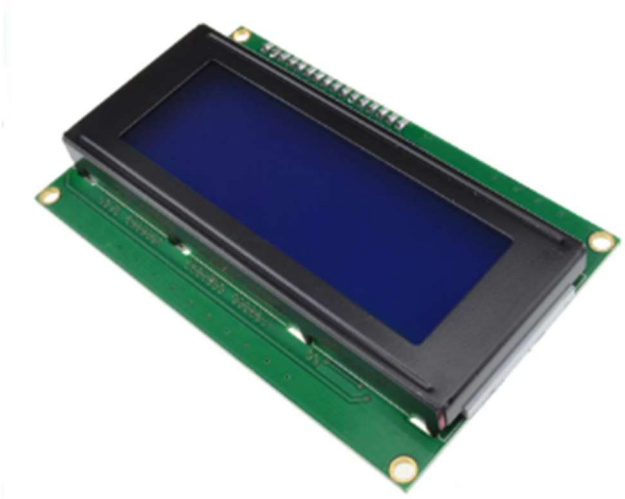


Figure 4: 20x4 LCD

4.1.5 Ultrasonic Sensor

We used the ultrasonic sensor for enhancing the gripping of objects as it gets as close as needed so it can grip the object tightly. Also it was used for detecting obstacles on the right and the left of the robotic arm as if obstacle is detect the arm will stop and send feedback waiting the obstacle to be moved or removed thus, it can continue the execution of the process.

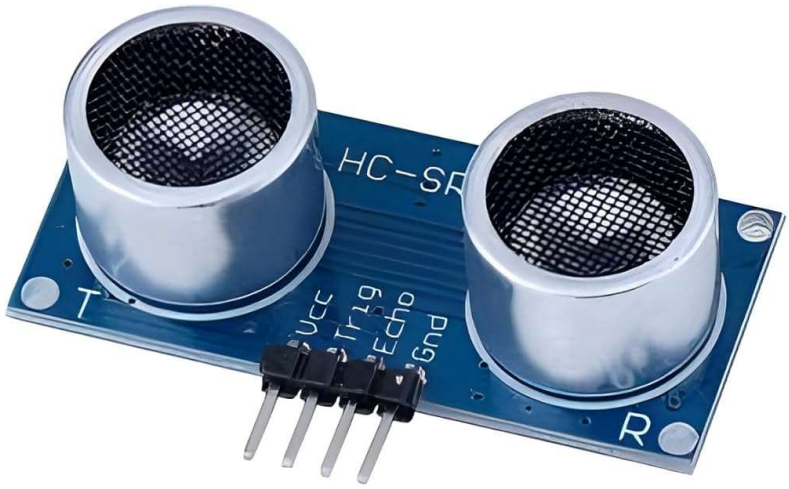


Figure 5: Ultrasonic Sensor

4.1.6 Current Sensor

A sensor that measures the flow of electric current in a circuit. It converts the current into a corresponding voltage or digital signal. We used the current sensor for implementing adaptive and safety gripper for ensuring both things: the target object is not affected by the force of the gripper and the servo motor in the gripper will not over strain so it doesn't get damaged over the time.

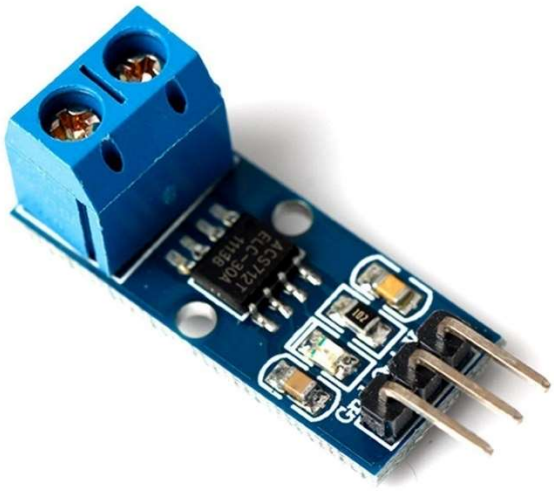


Figure 6: Current Sensor

4.1.7 Logic Level Shifter

A module that converts signals between different voltage levels, allowing devices with different operating voltages to communicate, it can safely interface a 5V microcontroller with a 3.3V sensor. we used it to facilitate communication between an ESP32 and an Arduino without damaging the ESP32, as the operating voltage for Arduino is 5V, while for the ESP32 it is 3.3V, and the pins on ESP32 does not tolerate 5V that coming from Arduino.

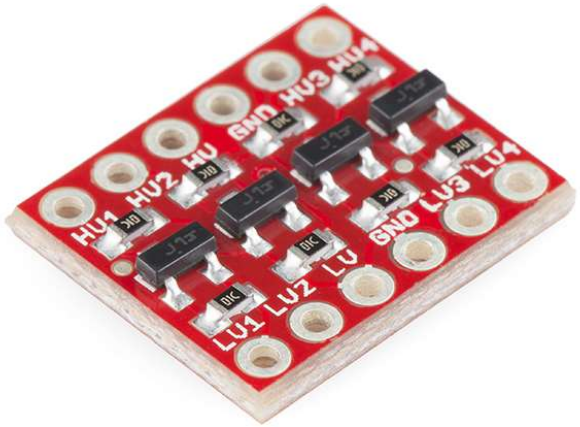


Figure 7: Level Shifter

4.1.8 ESP32

The ESP32 is a powerful microcontroller with integrated Wi-Fi and Bluetooth capabilities. It features dual-core processors, three UARTS, making it suitable for tasks that require high performance and connectivity like compiling and communication. We used it as arm central control by handling compiler output and sending commands to the Arduino, also it handles the communication with the raspberry pi. ESP32 was needed for high processor performance so that it enables on-fly code compilation and communication with Arduino.

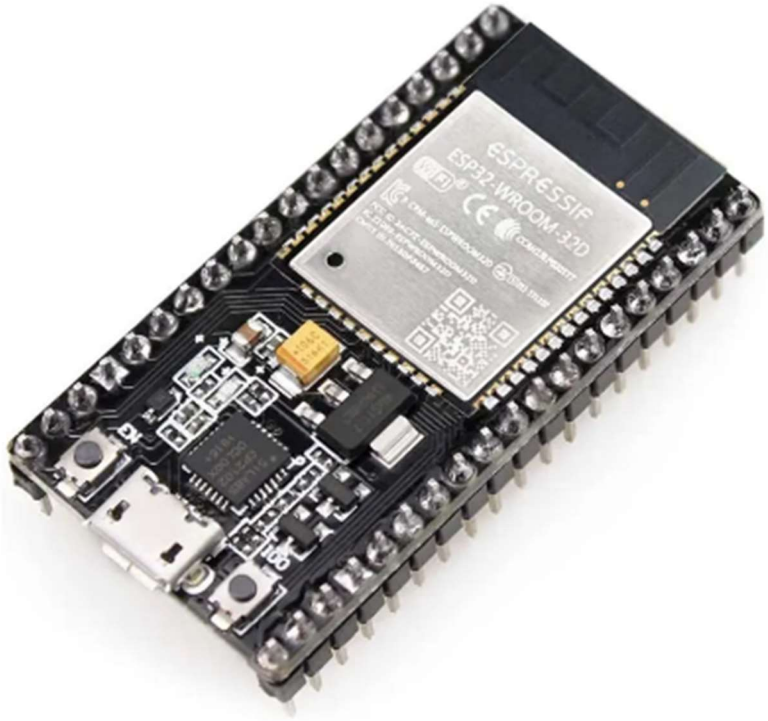


Figure 8: ESP32

4.1.9 Raspberry bi 5:

The Raspberry Pi is a small, affordable single-board computer that runs a full custom Linux operating system (PiOS). It is equipped with GPIO (General-Purpose Input/Output) pins, allowing it to interface with various electronic components and sensors. We used it to handle OAK-D camera image processing, Object detection and live streaming. It communicates with the mobile application through the Wi-Fi using a Flask Server, and communicate with the Arduino directly using serial communication (UART).

Figure 9: Raspberry bi 5

4.1.10 OAK Camera:

The OAK-D is a camera used for robotic vision that combining high-resolution colored camera and two stereo depth camera and with, equipped with Neural Network inferencing and Computer Vision capabilities. It also offers a Spatial AI which allows to get real life coordinates of an detected object. We used it as add-on feature to detect objects around the arm and allow the user to choose object to make the arm lift it. Additionally, can be used for live streaming of robotic arm for more visual feedback.

Figure 10: OAK-D AI Camera

4.1.11 Power Supply:

Providing the 6 servo motors with 12 volt and up to 10A power. Each servo motor at most need around 900mA so 10 Ambers will be more than enough to power up the robotic arm motors.

Figure 11: Power Supply

4.1.12 Springs:

We used springs to support the robotic arm shoulder and elbow, making it more precise and enabling it to have more rotational force

Figure 12: Metal Springs

4.1.13 Input Headers:

Input Headers were used to allow the sensor expansion, which provides external inputs for other sensor so that it can be used inside the custom compiler for running different sets of instructions based on the output of these sensors.

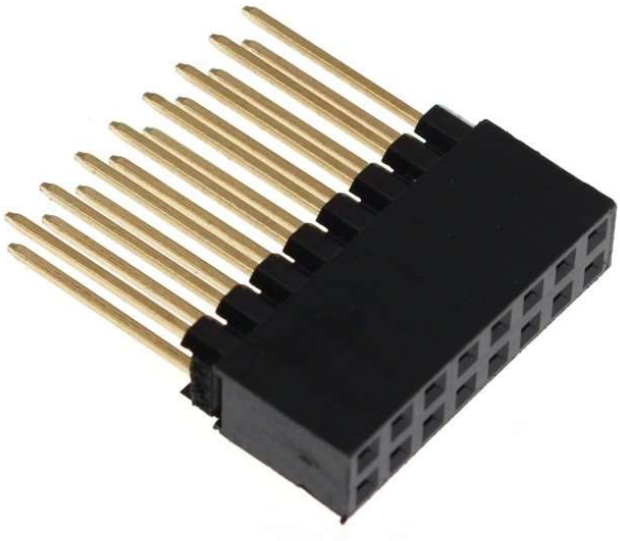


Figure 13: External Input Headers

4.1.14 Wires:

We used different type of wires to adapt to the type of connectivity among components.

Figure 14: male to male wires

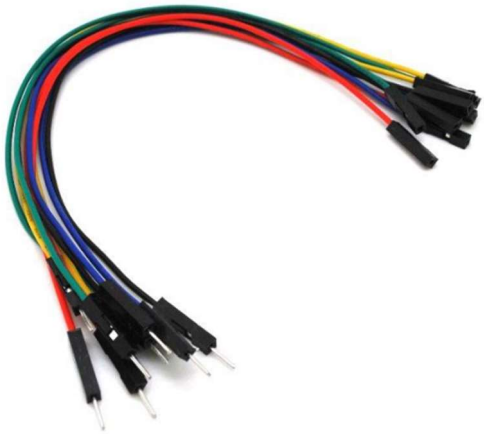


Figure 15: male to female wires

Figure 16: female to female wires

4.2 Mechanical Part:

The mechanical structure of the robotic arm is designed to provide six degrees of freedom (DOF) enabling it to perform a wide range of motions for variety range of general tasks. The arm is composed of lightweight and durable materials like aluminum alloy, which will provide the required strength while maintaining ease of movement.

The joints of the arm are powered by combination of high-torque and high-speed servo motors, high-torque motors were used for ensuring precise control of movements along all axis. These motors are capable of handling a range of objects, making the arm flexible for different object sizes and weights. And the high-speed motors were used for faster operations.

The base of the arm is mounted on a stable wood platform which providing a solid base that ensures stability during operation. And the arm gripper is equipped with a flexible, adjustable mechanism consist of gears that allows for secure handling of various objects. In addition, it is equipped with current sensor, ultrasonic and IR sensors for better adapting to different shapes and sizes of objects.

For object detection and quality checking tasks, the arm is equipped with the OAK-D camera and additional sensors that are added into the mechanical design. These sensors enable real-time feedback, allowing the arm to avoid obstacles and protect motors from getting damaged.

This mechanical design allows manual and automated control making the arm suitable for applications such as assembly line operations, warehouse management and other industrial automation tasks.

The software control for the robotic arm provides inverse kinematics, which allows to control the arm through X,Y and Z axis coordinates instead of controlling each motor degree.

4.3 Mechanical Design:

These following pictures demonstrate the 3D design of the robotic arm and the assembled design of the robotic arm.

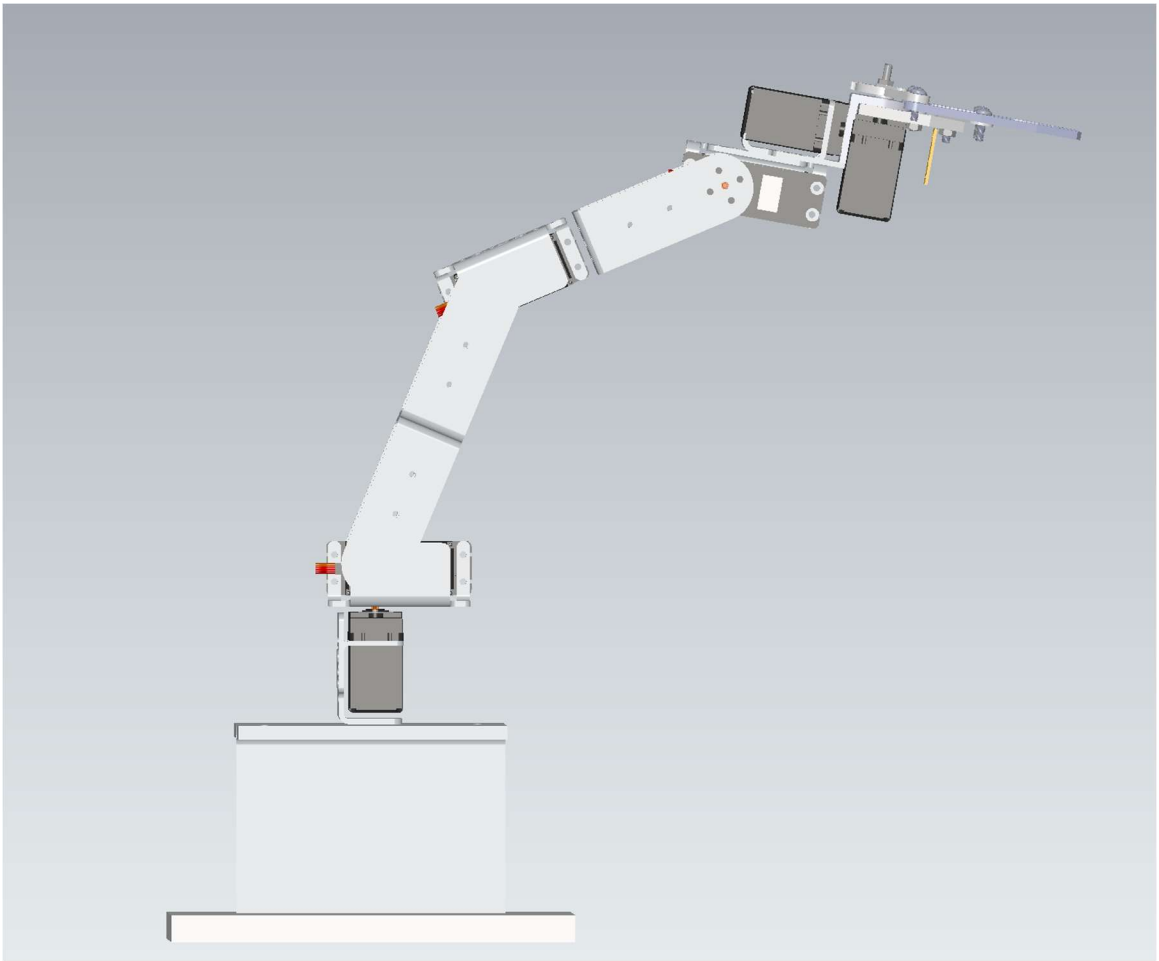


Figure 17: Side view of the robotic arm design

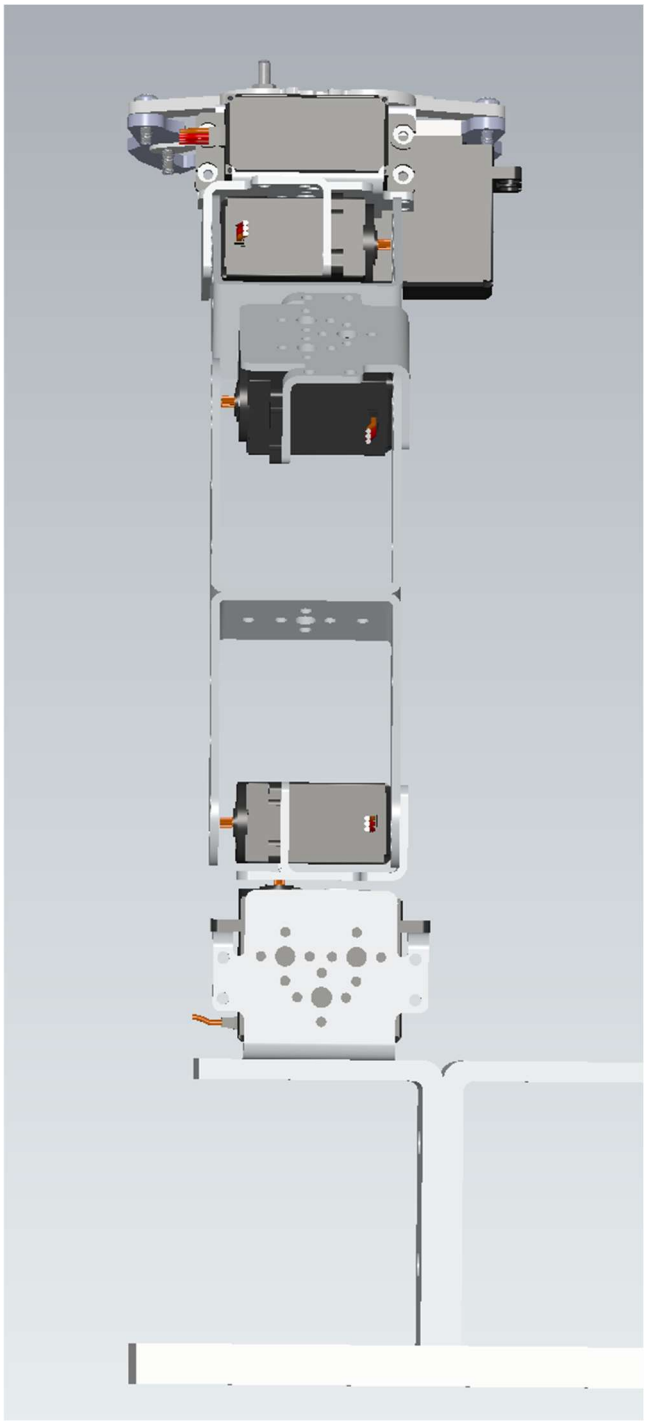


Figure 18: Back view of the robotic arm design

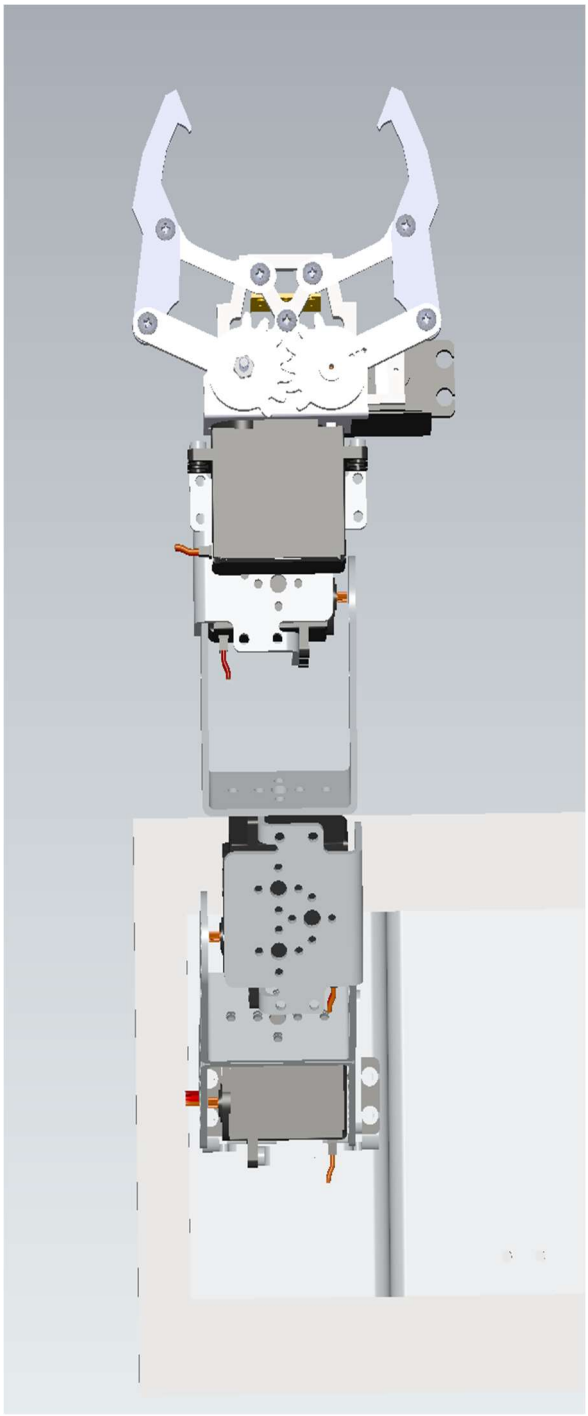


Figure 19: Front view of the robotic arm design

Figure 20: The assembled design

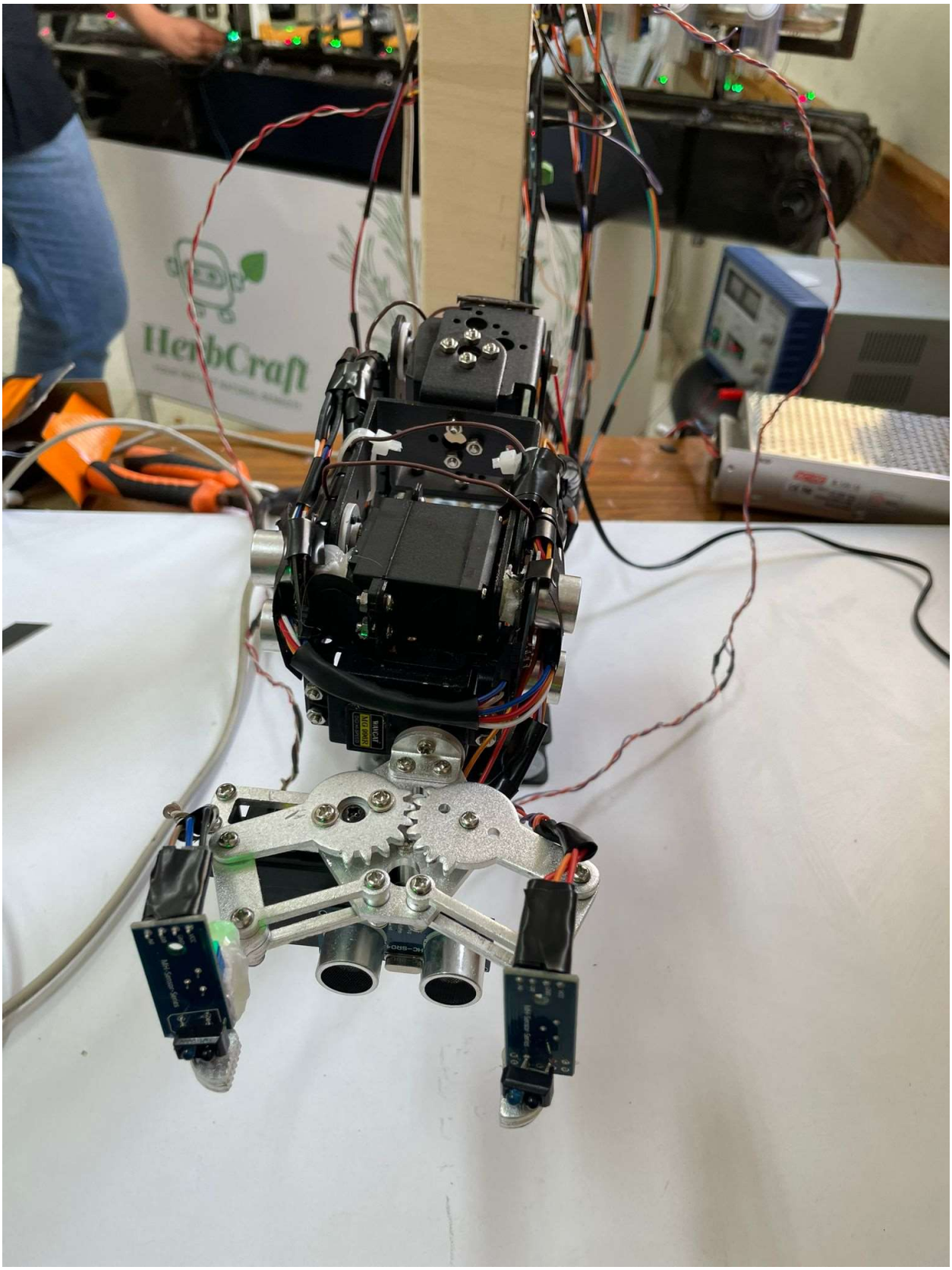


Figure 21: The final assembled robotic arm

Figure 22: Side view of the final design

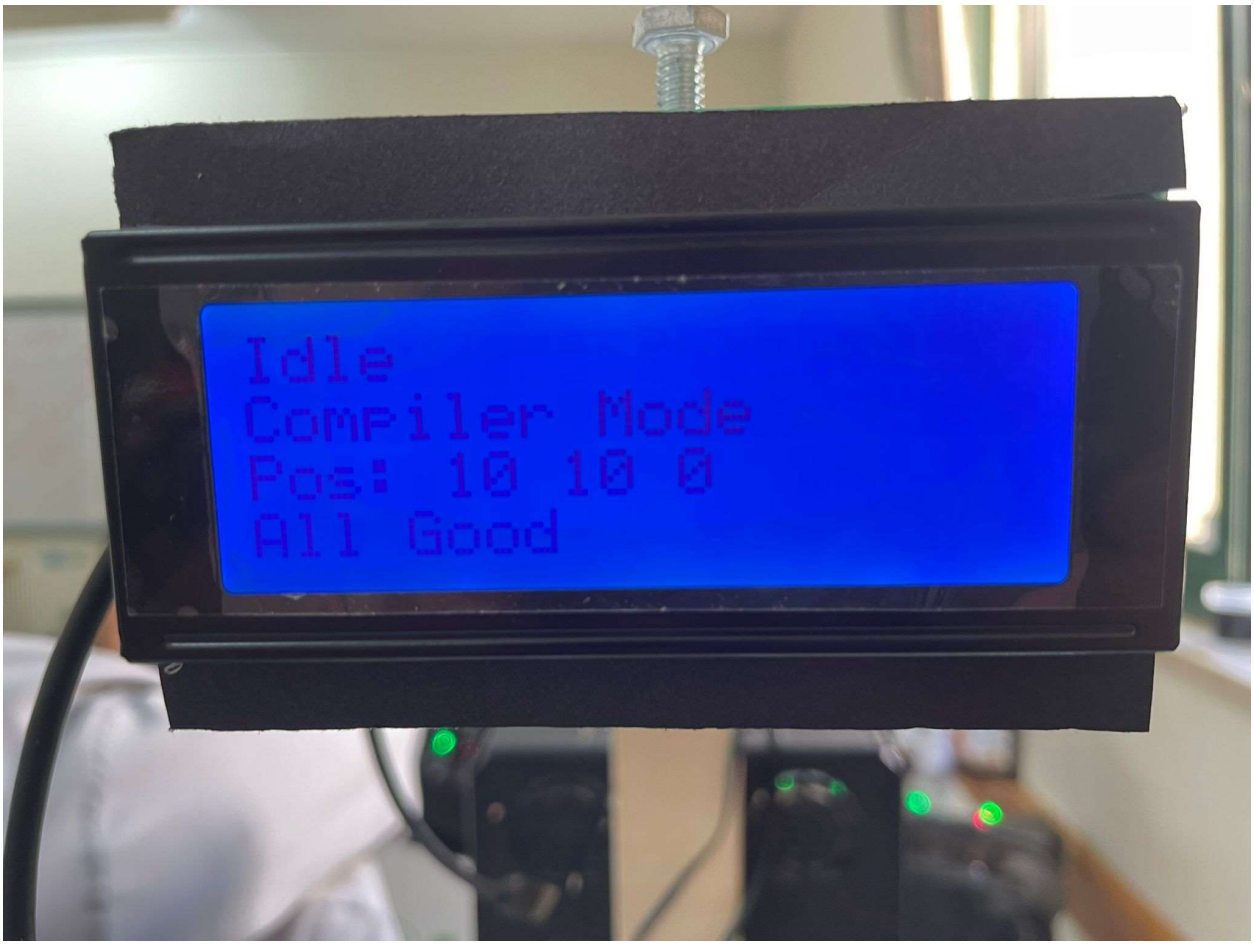


Figure 23: LCD Status display

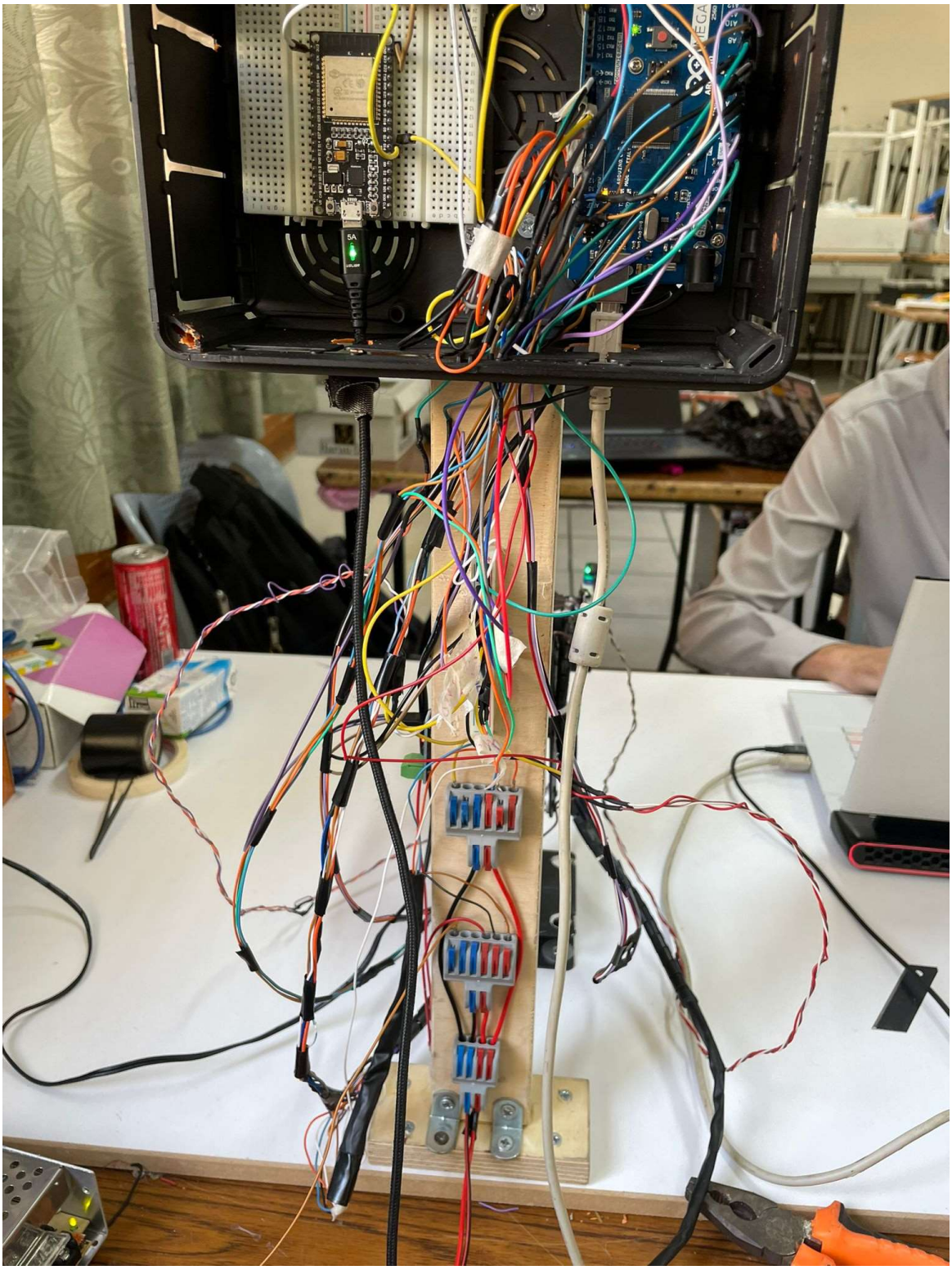


Figure 24: The wire connections for the project

4.4 Features:

4.4.1 Inverse Kinematics

Inverse kinematics in a 6 DOF robotic arm is the process of calculating the joint angles needed for the arm to reach a specific position and orientation in 3D space. It involves determining how each joint should move to the desired location accurately. This is essential for ensuring precise and accurate movement, especially in tasks like object manipulation or assembly line.

Now we need inverse kinematics to make x y z system for the arm. X is the distance from arm base center to the target, Y is the height from arm base and Z is for arm direction. Now for X and Y we need pioneer 2 inverse kinematics for shoulder (Θ2) and elbow (Θ3)

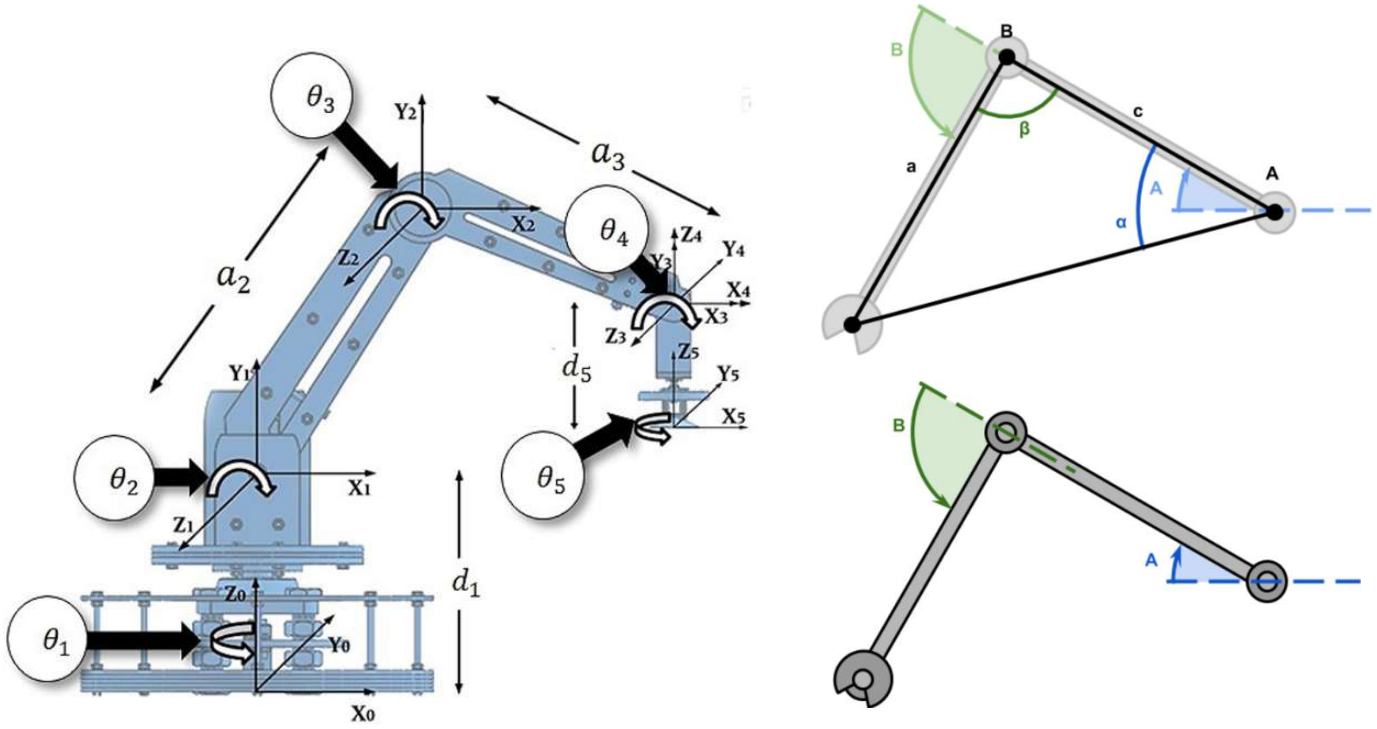


Figure 25: Inverse Kinematics Angles and Lengths

Now let's imagine robotic arm with 2 joints, and its end effector that we need to control, we actually don't have control on end effector, we can only change position of servo motors. The problem of inverse kinematics, is that finding the best way to change the motors position to move the end effector to the desired location.

A robotic arm with two joints can be modelled as a triangle, which is one of the most studied geometrical figures in geometry.

Now we have 2 joints, A and B in black, can be rotated by A (blue) and B (green) degrees, respectively. This causes the end effector to move to a position C.

Now we have triangle A, B and C, with internal angles β, α and γ like figure below.

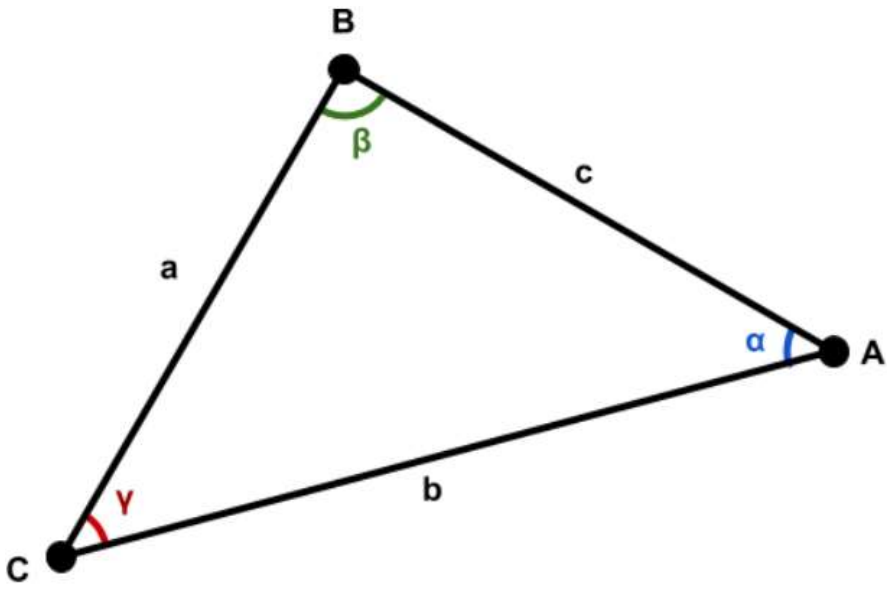


Figure 26Triangle concept for Inverse Kinematics

While these three angles are unknown, we know the length of edges:

- Segment **CB** Represents the humerus
- Segment **BA** Represents the forearm
- Segment **AC** Represents the distance between shoulder and the end effector

Knowing these three segments length is enough to find all of the angles.

We need to calculate β, α and γ to control the end effector, Let's start with α:

$$a^2 = b^2 + c^2 - 2bc\ cos\ \alpha$$

After refactor the formula to get α:

$$\cos \alpha = \frac{b^2 + c^2 - a^2}{2bc}$$

$$\alpha = \cos^{-1} \frac{b^2 + c^2 - a^2}{2bc}$$

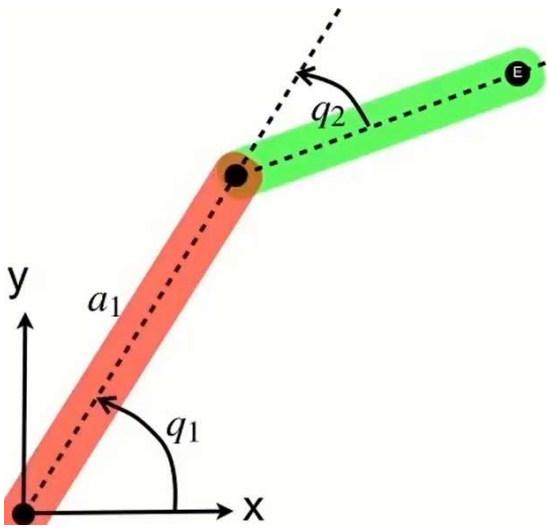
And same to β:

$$b^2 = a^2 + c^2 - 2ac\ cos\ \beta$$

$$\cos \beta = \frac{a^2 + c^2 - b^2}{2ac}$$

$$\beta = \cos^{-1} \frac{a^2 + c^2 - b^2}{2ac}$$

Now for γ we can get it from β angle by this procedure:



Now like we said previously we have two main segments a1(a) and a2 (c), and two main angles q1 (γ) and q2 (-β). Based on this information we can get q1 by this formula:

$$q1 = \tan^{-1} \frac{y}{x} + \tan^{-1} \frac{a2 \sin q2}{a1 + a2 \cos q2}$$

The result after combining all of the mentioned concepts and equations is shown bellow on DESMOS software. It shows the two anchor points of the robotic arm which is associated with these equations to control the movement based on coordinates rather than angles.

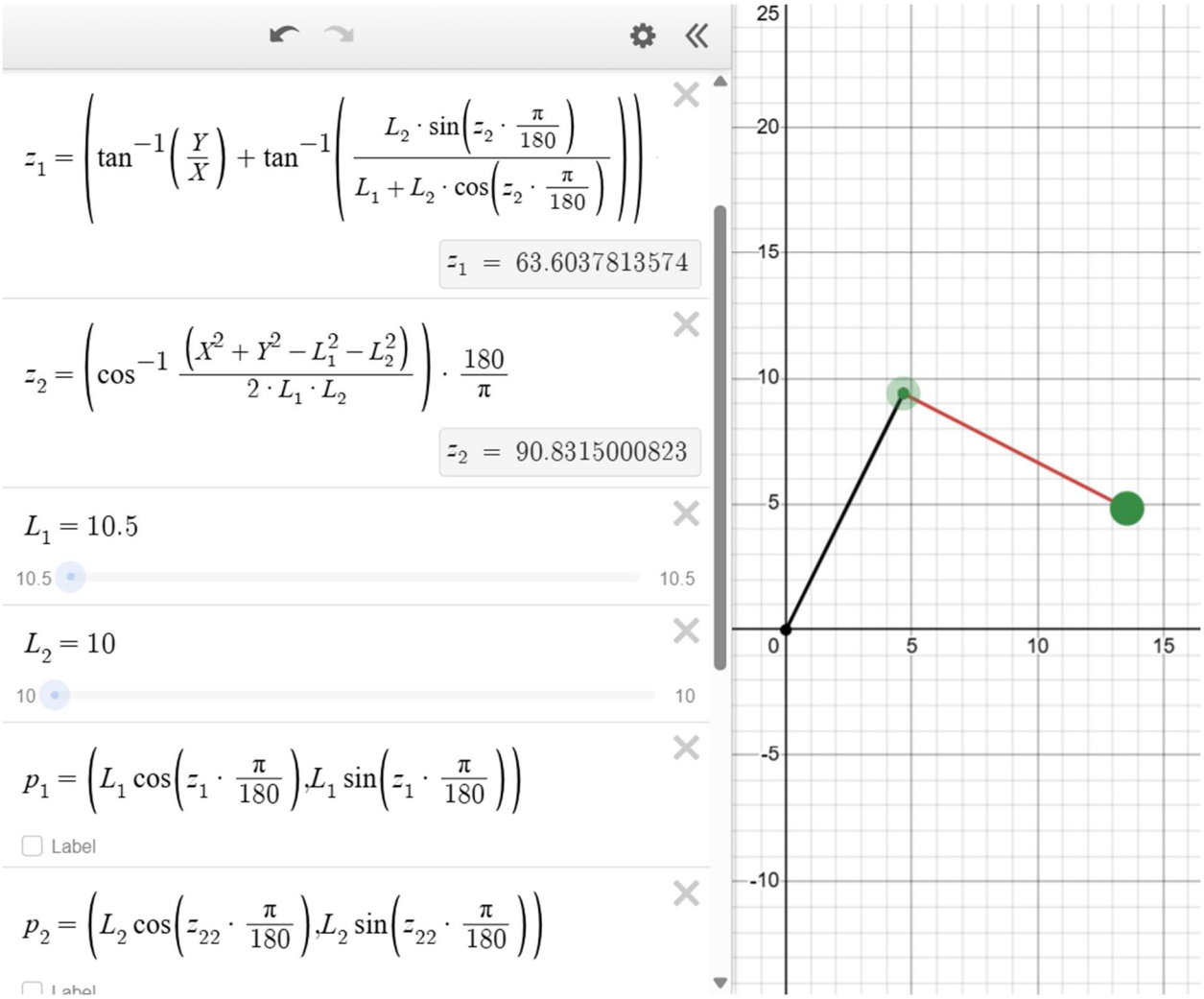


Figure 27: Inverse kinematics formulas

4.4.2 Custom Compiler Design

There are too many standard languages that made for robotics field but we built our own language for two main reasons. The first reason is that we need to simplify the programming process for the client as much as possible, as the known languages are very difficult to learn and require a specialist, and the second reason is that the final code (assembly code) in the known languages is difficult to parse on microcontrollers in general and on esp32 in particular, due to the small memory size and weak processor compared to computer processors, and requires to reboot it to run the code, while our final code just needs a few seconds to run it without reboot.

We made the compiler on python, and made our low-level standard code to parse it on esp32 without filling the memory with unnecessary strings and loops, and we used esp32 rather than Arduino because it is affordable alternative with larger memory capacity and superior processing speed.

We built Lexer to generate lexemes and Parser to parse these lexemes to AST, and code generator that takes the AST and convert it into JSON format, after that we made local web application to send the final code (JSON) to esp32 in few seconds to parse it recursively and run it.

The code is saved permanently, so if the ESP32 is turned off and then turned on again, the code remains saved.

Figure 28: Compiling mechanism

After the ESP32 done the parsing of the final code, it will start sending the commands to Arduino that directly controls the robotic arm, by using serial communication through UART.

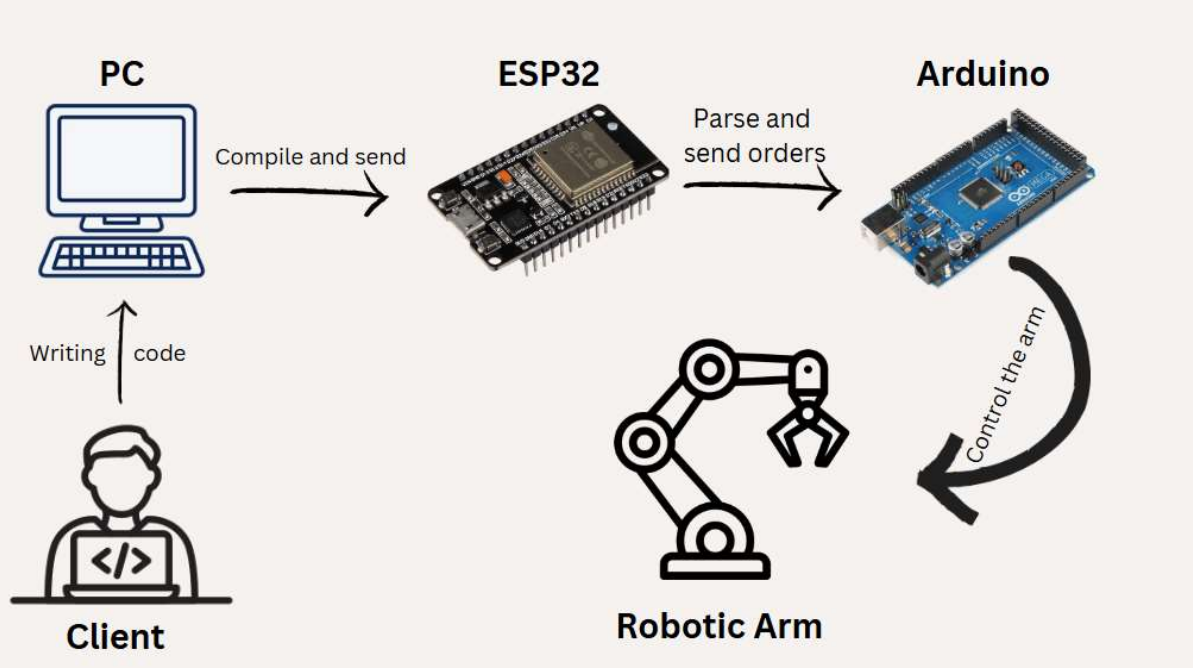


Figure 29: Compiler flow

In the client side, a web page was made to handle the compiler input, which offers syntax highlighting, auto-indentation, auto-closing brackets, some semantic checks, status and feedback and it shows the compiled version of the input for debugging purpose.

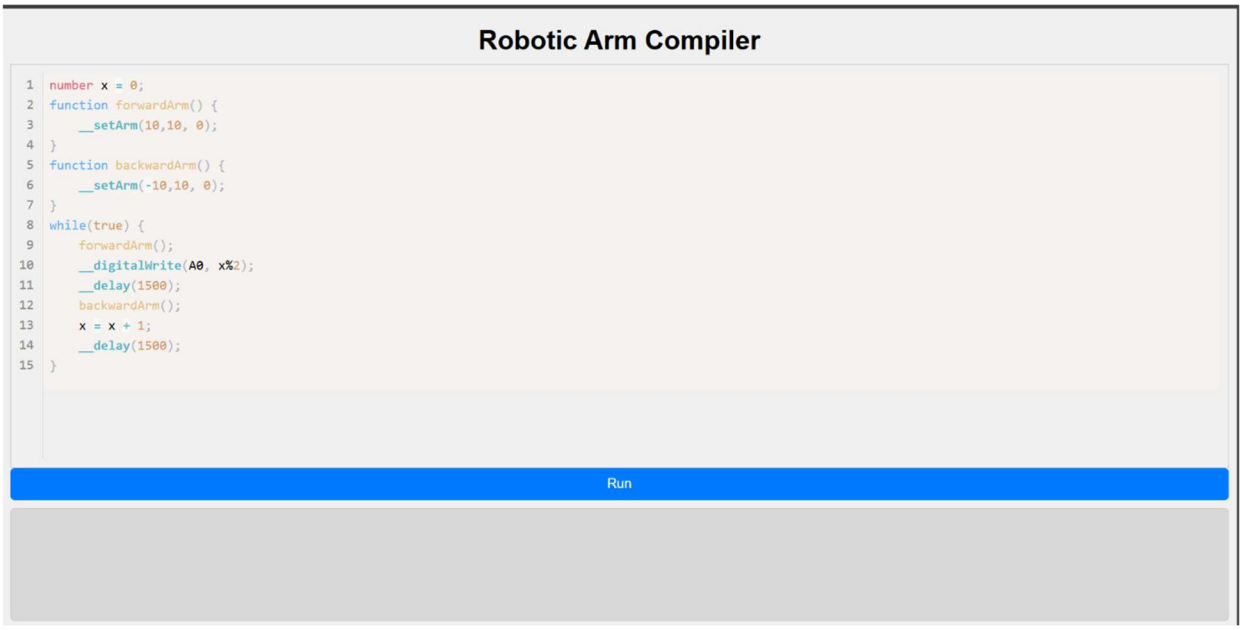


Figure 30: Compiler IDE interface

4.4.3 Programming Language Documentation

1. **digitalWrite(pin, state)**
   Description: Sets the specified digital pin to either HIGH or LOW.
   Parameters:
   - *pin*: The pin number to which you want to write (e.g., 2, 3, 13).
   - *state*: The state to set the pin to; use true for HIGH and false for LOW.

   Example Usage:

```
digitalWrite(13, true);  // Set pin 13 to HIGH
digitalWrite(13, false); // Set pin 13 to LOW
```

2. **digitalRead(pin)**
   Description: Reads the current state of a specified digital pin.
   Parameters:
   - *pin*: The pin number to read (e.g., 2, 3, 13).

   Returns: true if the pin is HIGH, false if the pin is LOW.
   Example Usage:

```
bool state = digitalRead(13); // Read state of pin 13
```

3. **analogWrite(pin, value)**
   Description: Writes an analog value (PWM wave) to a specified pin.
   Parameters:
   - *pin*: The pin number to which you want to write.
   - *value*: The duty cycle of the PWM signal (0-255).

   Example Usage:

```
analogWrite(9, 128);  // Set PWM value of pin 9 to 128
```

4. **analogRead(pin)**
   Description: Reads the value from the specified analog pin.
   Parameters:
   - *pin*: The analog pin number to read.

   Returns: A number between 0 and 1023.
   Example Usage:

```
int sensorValue = analogRead(A0); // Read analog value from pin A0
```

5. **delay(milliseconds)**
   Description: Pauses the program for the specified number of milliseconds.
   Parameters:
   - *milliseconds*: The number of milliseconds to delay.

   Example Usage:

```
delay(1000);  // Pause for 1 second
```

6. **print(number)**
   Description: Prints a number to the serial monitor (string output is not supported).
   Parameters:
   - *number*: The number to print.

   Example Usage:

```
print(42);  // Prints the number 42 to the serial monitor
```

7. **setArm(x, y, z)**
   Description: Sets the robotic arm's position in 3D space.
   Parameters:
   - *x*: X-coordinate.
   - *y*: Y-coordinate.
   - *z*: Z-coordinate.

   Example Usage:

```
setArm(10, 20, 30);  // Move arm to (10, 20, 30)
```

8. **setArmSpeed(speed)**
   Description: Changes the arm speed (0-255).
   Parameters:
   - *speed*: Speed (0-255).

   Example Usage:

```
setArmSpeed(100);  // Change arm speed to 100
```

9. **setWrist(angle[, speed])**
   Description: Sets the wrist's angle and speed.
   Parameters:
   - *angle*: The angle to set the wrist.
   - *speed*: Wrist speed (0-255), optional.

   Example Usage:

```
setWrist(45, 10);  // Set wrist to 45 degrees at speed 10
```

10. **autoWrist()**
    Description: Automatically adjusts the wrist angle relative to the shoulder and elbow positions.
    Example Usage:

```
autoWrist();  // Adjust wrist based on shoulder and elbow
```

11. **rotateWrist(angle[, speed])**
    Description: Rotates the wrist to the specified angle.
    Parameters:
    - *angle*: The angle to rotate the wrist to.
    - *speed*: Rotating speed (0-255), optional.

    Example Usage:

```
rotateWrist(90);  // Rotate wrist to 90 degrees
```

12. **openGripper()**
    Description: Opens the robotic gripper.
    Example Usage:

```
openGripper();  // Open the gripper
```

13. **closeGripper()**
    Description: Closes the robotic gripper.
    Example Usage:

```
closeGripper();  // Close the gripper
```

14. **forceCloseGripper(speed)**
    Description: Closes the robotic gripper without considering the gripper strength.
    Parameters:
    - *speed*: Closing speed (0-255).

    Example Usage:

```
forceCloseGripper(100);  // Close the gripper with speed 100
```

15. **forceOpenGripper(speed)**
    Description: Opens the robotic gripper without considering the gripper strength.
    Parameters:
    - *speed*: Opening speed (0-255).

    Example Usage:

```
forceOpenGripper(100);  // Open the gripper with speed 100
```

16. **wait()**
    Description: Waits until the arm stops moving.
    Example Usage:

```
wait();  // Wait until the arm stops
```

4.4.4 OAK-D Camera and Image Processing

The OAK-D is the ultimate camera for robotic vision that perceives the world like a human by combining stereo depth camera and high-resolution color camera with an Neural Network inferencing and Computer Vision capabilities. The primary usage of the camera was the Spatial AI that it offers, which can be used to calculate the real position of objects.

At first phase, we have tried to use Yolo image classifier but it didn't handle the job well, as this model face difficulties in detecting objects from the top view, and it is limited to small set of detectable objects. So, we had to make our own object detection by pross the colored image that comes from the RGB camera in the OAK-D camera. The graph below shows the flow of objects detection.

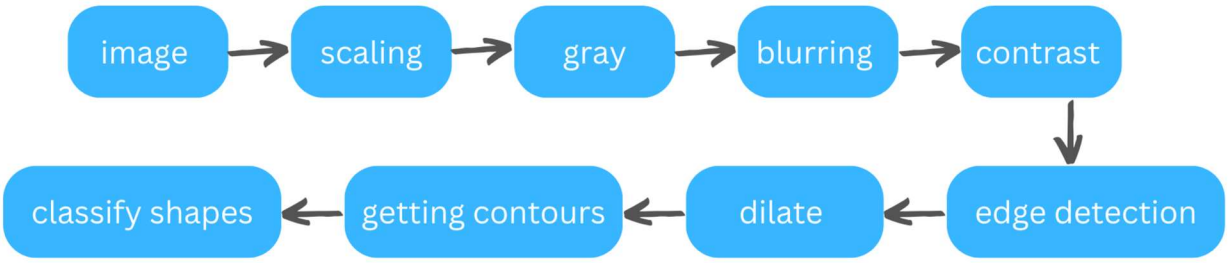


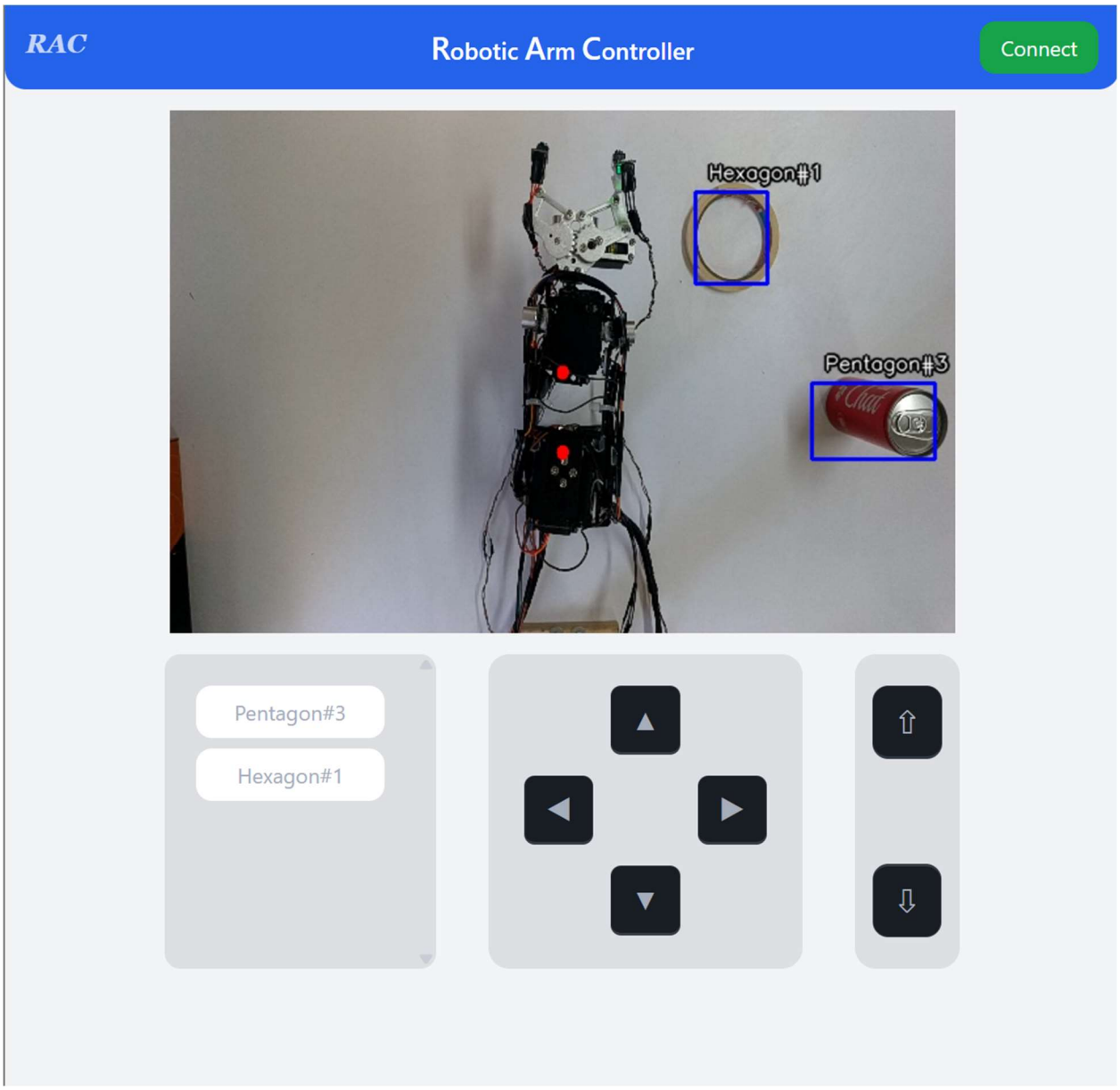
Figure 31: Image processing flow

Second phase was to use the Spatial AI on the device, but we faced another challenge that is, the spatial calculator on the camera can't handle high resolution images. Therefore, we had to calculate the spatial coordinates on the host instead (which is Raspberry Pi 5), by giving it the ROI (Region-Of-Interest, bounding box of an object) and setting the lower and upper threshold of a depth can be. Fortunately, it did well and better than sending depth/ROI back to the device.

For the third phase, we have obtained the spatial coordinates, but we need to make it relative to the base of the robotic arm rather than the center of the camera, which required a lot of time for tuning.

Flask server was used to handle the API of camera connectivity, image processing and object detections. Which runs on the Raspberry pi 5 and it communicate with Arduino serially through UART.

For the manual control, arrows with live camera feedback is provided through a web and a mobile application.



The camera shows the object inside a frame, also detect the object so that user can either pick objects from the camera view, or by the list of object names provided. Also, the user can control the movement by simple arrows for testing and debugging.

5 Conclusion

The development of the 6 Degree of Freedom (DOF) programmable robotic arm has shown its potential in solving key challenges in assembly line operations. By integrating flexibility, wireless control, and adaptability to different object sizes and weights, also enabling object detection and selection, thus, this robotic arm reduces the need for manual on-site programming and minimizes costly shutdowns for changes and adjustments. And the use of Arduino, ESP32 and Raspberry Pi 5 for control and communication has allowed for a robust system that can be easily programmed and controlled through a mobile interface or coding it through simple and easy to use custom compiler. Also, the addition of object detection and quality assurance enhances its flexibility making it suitable for a wide range of industrial applications.

6 Future work

While the current prototype is fully functional, several enhancements can be made to improve its usability and performance:

1. **Support for Integrating Standard Programming Languages**: By expanding the system to support integration of widely used languages such as C, C++, and MATLAB would enable more users to program the robotic arm, and enable to build more complex system.

2. **Recording Human Arm Motion**: Implementing a feature to record and replicate human arm movements would allow the robotic arm to learn from human demonstrations, enhancing its ability to perform complex, human-like tasks.

3. **Cooperation with Other Robotic Arms**: Integrating communication protocols for cooperation between multiple robotic arms would enable collaborative tasks in manufacturing and assembly, which will lead for performing more complex tasks.

4. **Enhanced Object Recognition and AI**: Future versions of the arm could have advanced AI algorithms to improve object recognition and task automation, reducing the need for human intervention and checking.

5. **Integration with IoT and Cloud**: Connecting the robotic arm to IoT networks and cloud services for remote monitoring and data analysis would offer real-time diagnostics and performance optimization which would increase its industrial value and performance.

These future developments and features would expand the capabilities of the robotic arm, making it more flexible, efficient and applicable across different sectors beyond assembly lines, such as warehouse management and human task automation even surgical ones.

7 References

[1] M. E. Moran, "Evolution of robotic arms," *J. Robotic Surgery*, vol. 1, no. 2, pp. 103–111, Jul. 2007, doi: 10.1007/s11701-006-0002-x.

[2] K. Singh, "Working of Robotic Arm in Industries," unpublished.

[3] W. S. Barbosa, M. M. Gioia, V. G. Natividade, R. F. F. Wanderley, M. R. Chaves, F. C. Gouvea, and F. M. Gonçalves, "Industry 4.0: examples of the use of the robotic arm for digital manufacturing processes," *Int. J. Interactive Design and Manufacturing (IJIDeM)*, vol. 14, no. 4, pp. 1569-1575, Dec. 2020, doi: 10.1007/s12008-020-00714-4.

[4] J. Q. Gan, E. Oyama, E. M. Rosales, and H. Hu, "A complete analytical solution to the inverse kinematics of the Pioneer 2 robotic arm," *Robotica*, vol. 23, no. 1, pp. 123–129, 2005, doi: 10.1017/S0263574704000529.

[5] M. J. Hayawi, "Analytical inverse kinematics algorithm of a 5-DOF robot arm," *J. Educ. College*, vol. 1, no. 4, 2011.