Department of Electronic and Computer Engineering

Electronic Design Project 3A (EDPA301)

Assessment Number 5A

Final Report (Individual)

Project Title: Remote Controlled Robotic Arm that Measures Distance

By

Group Number: A		
Student Surname	Initials	Student Number
Thakurdin	T Y	22139454

PLAGIARISM DECLARATION

1. I know and understand that plagiarism is using another person's work and pretending it is one's own, which is wrong.

2. This report is my own work.

3. I have appropriately referenced the work of other people I have used.

4. I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his/her own work.

Thakurdin T Y	22139454	[illegible]
Surname and Initials	Student Number	Signature

1 July 2023

Date

Abstract

This report discusses the design, construction, and testing of the remote-controlled robotic arm that can measure distance project, specifically the joystick and servo motor subsystem.

The remote-controlled robotic arm was controlled by two joystick modules which were used to actuate three servo motors which in turn enabled the movement of the robotic arm. The system incorporates two ultrasonic distance sensors for vertical and horizontal distance measurements as well as two nRF24L01+ modules to facilitate bi-directional wireless communication between the control centre and the robotic arm microcontrollers. The final distance measurements were displayed on an I2C LCD screen at the control centre. Indicator LEDs provided visual feedback for button presses and successful data transfers.

Thus, this project offers a user-friendly and efficient solution for controlling and monitoring the robotic arm's functionalities by combining the joystick control of servo motors, ultrasonic sensors for distance measurement, radio frequency modules for wireless communication, and an LCD screen and indicator LEDs for real time process monitoring.

Table of Contents

List of Figures

List of Tables

Table of Abbreviations

Abbreviations	**Meaning**
LED	Light Emitting Diode
LCD	Liquid Crystal Display
I2C	Inter-Integrated Circuit
Hz	Hertz
GND	Ground
VCC	Voltage Common Collector
RF	Radio Frequency
PS2	PlayStation 2
DOF	Degrees of Freedom
I/O	Input and Output
ADC	Analogue to Digital Converter
DC	Direct Current

Chapter 1: INTRODUCTION

1.1 Project Overview

A robotic arm is a machine that mimics the movement and motions of a human arm via the use of linkages and joints which in turn enable the arm to manoeuvre within its defined degrees of freedom and execute designated tasks [1-3].

Robotic arms are extensively used in medical surgeries, manufacturing, warehouses, assembly lines, research laboratories and space exploration [4, 5]. They excel in challenging environments, offering increased precision, repeatability, and efficiency.

In certain scenarios, remote control becomes necessary for robotic arms, particularly in military applications like remote bomb defusal. This project aimed to undertake the research, design, and construction of a remote-controlled robotic arm equipped with distance measurement capabilities which holds potential real-world applications, such as assisting surgeons in intricate procedures where precision cutting on a minute scale was required [4, 5].

This project utilized joysticks to control the precise actuation of servo motors to facilitate various horizontal and vertical movements of the robotic arm and once held in a new position, distance measurement commenced, and the results were displayed on an LCD.

The robotic arm consisted of interconnected joints, specifically the base, shoulder, and elbow joints, each equipped with their individual servo motors used to facilitate the movement of the robotic arm. User control was achieved at the control centre by utilizing two dual-axes joystick modules, the physical X axis and Y axis movements of the joysticks served as analogue inputs for the control centre microcontroller. This positional data was sent to the robotic arm system where they were converted into PWM signals. These signals were used to control the rotation of the servo motors, enabling the operator to manipulate the arm's movement and position.

To measure distance, ultrasonic sensors were implemented. These sensors emit sound waves, and the distances were calculated based on the time it takes for sound waves to return.

For visual feedback, an LCD screen with an I2C interface was integrated into the system. The control centre microcontroller was connected to the LCD display. The microcontroller received the transmitted distance measurement data from the robotic arm system and displayed the final values on the LCD.

The two systems operated on separate DC power supplies which powered the microcontrollers which in turn powered and controlled their respective components. This enables operators to remotely control the robotic arm's movements using the joysticks while receiving real-time distance measurement information on the LCD.

1.2 Aims, Objectives and Specifications

1.2.1 Aim

The aim of the project was to design, construct and test a remote-controlled robotic arm that was capable of measuring distance. The project involved the development of two separate systems, namely the control centre and robotic arm, each equipped with their own dedicated external DC power supply.

1.2.2 Objectives

- Design and develop the control centre and robotic arm systems.
- Implement wireless communication between the control centre and the robotic arm using two nRF24L01+ transceiver modules.
- Integrate an LCD into the control centre to display distance measurements from the robotic arm.
- Incorporate two PS2 joystick modules as the primary input devices for the control centre.
- Configure the robotic arm system to actuate three servo motors based on the input commands received from the control centre.
- Utilize the Arduino microcontrollers to control and coordinate the functionalities of the control centre and the robotic arm system.
- Establish bidirectional communication between the control centre and the robotic arm system to enable the control centre to receive input commands from the joystick module and transmit them wirelessly to the robotic arm system to actuate servo motors and transmit the calculated distance measurements back to the control centre.
- Collaborate effectively with other group members to ensure a cohesive and coordinated project outcome.

1.2.3 Hardware and Performance Specifications

- Arduino Uno Rev 3 microcontroller: The Arduino Uno Rev 3 microcontroller was used to control the operations and functions of the control centre.
- Arduino ATmega328P microcontroller: The Arduino ATmega328P microcontroller was used to control the operations and functions of the robotic arm system.

- Servo Motors: The project utilized three servo motors as the main output devices for controlling the movements of the robotic arm.
- PS2 Dual-Axes joysticks: The joysticks served as the primary input devices for the control centre and were used to control the movement of the robotic arm.
- I2C LCD: The final distance measurements were displayed at the control centre using the I2C LCD.
- NRF24L01+ transceiver modules: The NRF24L01+ transceiver modules facilitated wireless communication between the control centre and the robotic arm.
- Accuracy: The robotic arm's movement and distance measurements must be accurate based on the operator's control.
- Responsiveness: The project must exhibit minimal delay and provide real-time response to operator inputs.
- Reliability: The wireless communication between the control centre and the robotic arm must maintain a consistent connection without significant signal loss.
- Reporting and Documentation: The project must be documented using IEEE referencing and include technical specifications, wiring diagrams, code documentation, and project management details.

1.2.4 Background and Significance

A robotic arm imitates human arm movements using linkages and joints. It operates within specific degrees of freedom and performs various tasks in medical surgeries, manufacturing plants, warehouses, assembly lines, research laboratories and space exploration [4, 5]. They are used in challenging environments, offering precision, repeatability, and efficiency that a human could not.

Figure 1.1 Robotic arm used to position a satellite [5].

A remote-controlled robotic arm with distance measurement capabilities offers valuable technical benefits in various industries. It enhances operational safety by enabling operators to control the arm from a secure distance, ensuring precise object detection. Furthermore, the integration of the distance measurement functionality ensures accurate and controlled movements, crucial for tasks that demand precision [6].

By minimizing manual intervention, the remote-control feature streamlines operations, translating to improved efficiency and safety [6]. Additionally, the arm's remote accessibility allows for exploration and maintenance in challenging or remote environments.

1.2.5 Project Benefits

- Precise control over the movement and positioning of the robotic arm.
- Accurate distance measurement in two dimensions.
- Real time visual feedback of process operations due to indicator LEDs and LCD.
- Increased safety by performing tasks deemed too dangerous for humans.
- Increase efficiency due to

1.2.6 Project Limitations

- The servo motors have a limited rangeability between 0° - 180°.
- The robotic arm's movement was limited to three DOF.
- The joysticks will degrade overtime and cause inaccuracies.
- The ultrasonic sensors have a limited measurement range from 2 – 19 cm.
- The transceiver module has a transmission rate limit of 250 Kbps to 2Mbps.
- The project faces power constraints due to the microcontrollers' voltage limits.

Chapter 2: DESIGN

2.1 Project Block Diagram

Figure 2.1 illustrates the systems block diagram. There are two main systems with three distinct stages. The control centre system and robotic arm system each have an input stage, a process stage, and an output stage.

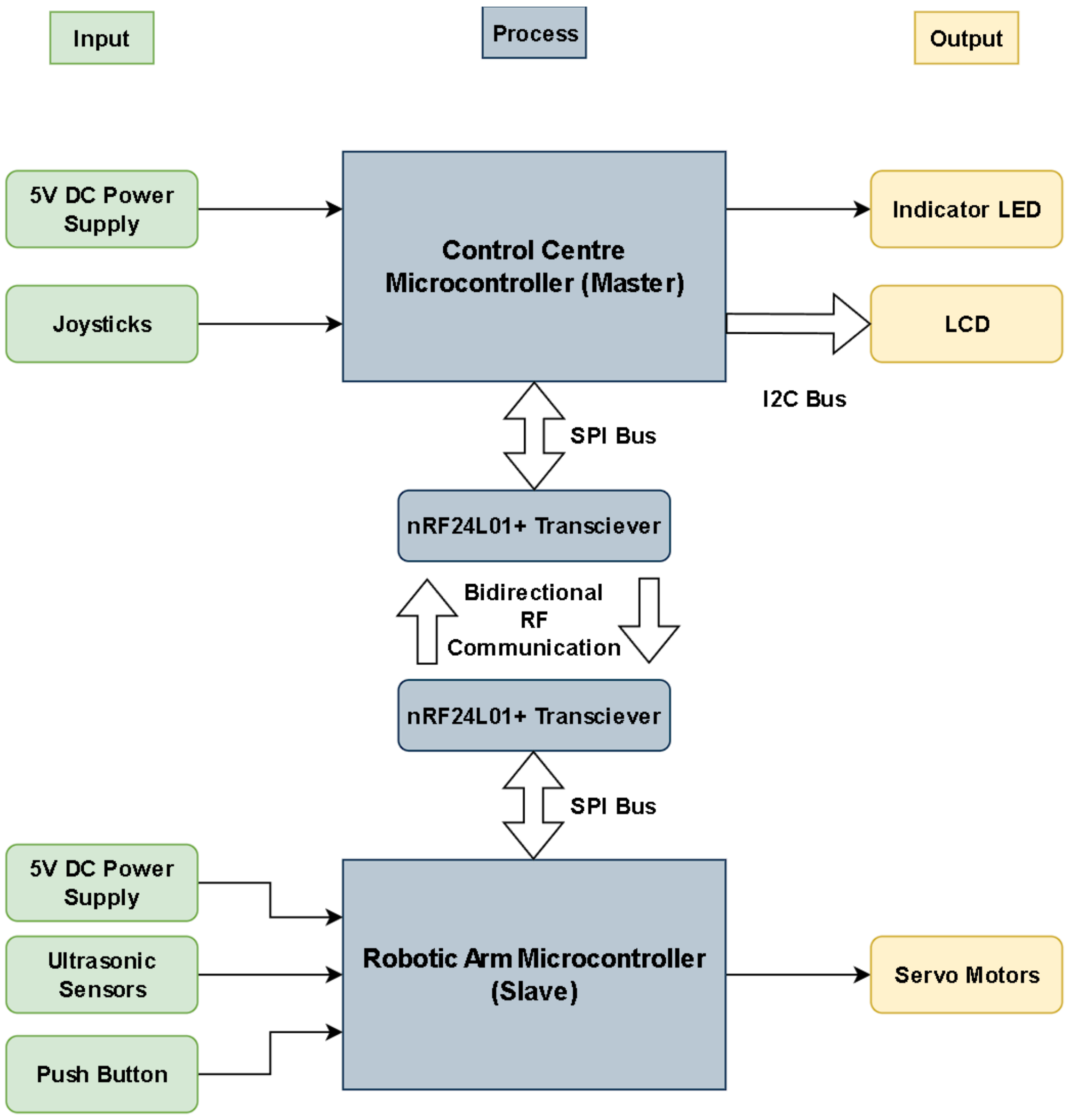


Figure 2. 1 Project Flowchart

2.1.1 Input Stage

With respect to the control centre's input stage, the microcontroller was powered by a 5V DC power source, which it used to distribute power to the other control centre components. Furthermore, the microcontroller read the changing X-axis and Y-axis positions of the joystick as analogue inputs.

Within the input stage of the robotic arm system, the microcontroller was powered by a 5V DC power source, which it used to distribute power to the other robotic arm system components. Such as the ultrasonic sensors which functioned as the main input devices of this system furthermore, when the push button was pressed once, the vertical distance was measured however if the button was pressed twice, the horizontal distance was measured.

2.1.2 Process Stage

Within the process stage, the input data was converted and transmitted or received between the two microcontrollers. The transmission of data was bidirectional and was facilitated using two nRF24L01+ transceiver modules.

The control centre stored the current position of the joystick and transmitted it to the robotic arm microcontroller where it was then mapped to an angle between 0° and 180°. This angle was then used to generate a PWM signal.

The robotic arm microcontroller converted the input from ultrasonic sensors into distance measurements, in centimetres, which were transmitted to the control centre microcontroller.

2.1.3 Output Stage

The robotic arm microcontroller used the previously generated PWM signals to control the actuation of the servo motors, in addition to this, the rotation of the servo motors caused the movement of the robotic arm. By changing the position of the arm, various distance measurements could be taken.

The control centre microcontroller used the data which it received from the robotic arm system to display the final distance measurement on the I2C LCD.

2.2 Project Circuit Diagram

As previously discussed, the project consists of two systems that communicate wirelessly., namely the control centre system and robotic arm system which are shown in figures 2.2 and 2.3.

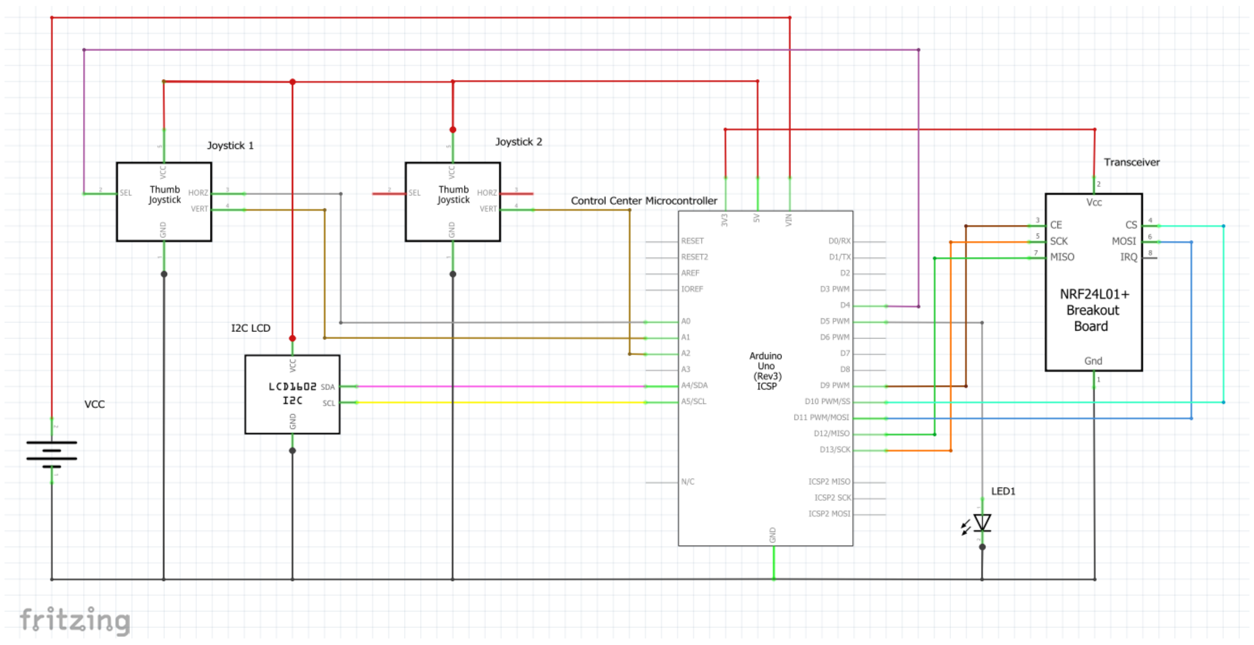


Figure 2. 2 Control Centre Circuit Schematic

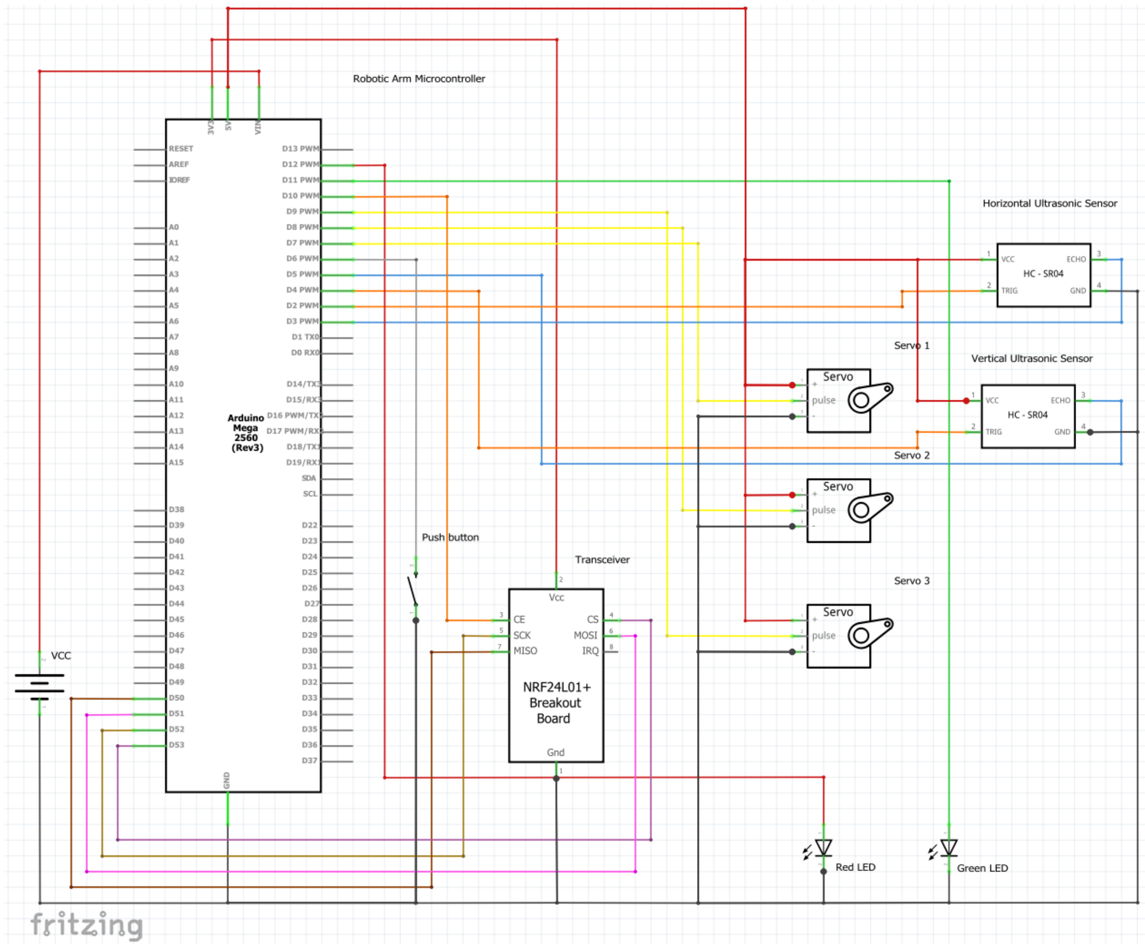


Figure 2. 3 Robotic Arm Circuit Schematic

2.2.1 Movement Stage:

The control centre system as the name suggests was where the operator remotely controlled the project. This system contained the two dual axis joysticks which functioned as the main input devices of the control centre that the operator interacted with to control the movement of the robotic arm. This was accomplished by reading the changing X-axis and Y-axis positions of the joysticks as analogue inputs. The joysticks were analogue inputs due to their ability to provide a continuous range of values, which allowed for smooth and precise control over the arm's movements, which was not achievable with discrete on/off digital inputs [7, 8].

The joysticks' positional data underwent serialization, where it was structured and formatted into a data packet which was wirelessly transmitted to the robotic arm microcontroller using the radio frequency transceiver modules.

The robotic arm microcontroller received the data packet and extracted the joystick positions. It then mapped the positional data to an angle between 0° and 180° to make use of the servo motors' full range of motion. The mapped angle was used as an argument for the servo.write() function to internally generate a PWM signal with the desired duty cycle and pulse width.

The PWM signal was generated on the digital I/O PWM pin connected to the signal wire of the servo motor. The servo motor interpreted the pulse width of the signal, using its built-in control circuit, to determine the desired degree of rotation, it then adjusted its shaft position based on the received pulse width, aligning itself to the specified angle [9, 10]. The three servo motors, the base, shoulder, and elbow servo motors were mounted on the robotic arm. The base servo motor facilitated horizontal movement while the shoulder and elbow servo motors were responsible for vertical movement.

2.2.2 Distance measurements and display stage

To measure the distance covered by the robotic arm, the project included two HC-SR04 ultrasonic sensors. These sensors emitted sound waves that travelled through the air and reflected off obstacles encountered along the path.

For vertical distance measurement, the ground surface was detected as an obstacle, while a solid box served as an obstacle for horizontal distance measurement. By calculating the time duration of the echo pin's high state and utilizing the speed of sound, the distance could be accurately By using formula 2-1 where D is the distance measured in cm, t is the time duration of the echo pin's high state, 0.34 is a constant representing the speed of sound in air which is 0.034cm/µs and the division constant 2 represent the path of the sound wave being emitted to an obstacle its path back.

$$D(cm) = \frac{t*0.34}{2} \tag{2-1}$$

This distance measurement was transmitted to the control centre microcontroller where it was displayed on a 16×2 LCD which was interfaced with the control centre microcontroller through the I2C interface.

2.2.3 Wireless communications stage

Two nRF24L01+ transceiver modules facilitated the wireless communications between the control centre and robotic arms systems using radio frequency signals furthermore, the two systems communicate bidirectionally.

Prior to the transmission stage, the input data was obtained and converted into a data packet. By checking if the transmission channel was open, the microcontroller determined whether it should transmit or receive data. If the transmission channel was available, the microcontroller transmitted the data packet to the other system. Similarly, if the transmission channel was not available, the microcontroller received the transmitted data packet and extracted the data from the other system. This process was identical for both the systems. Should neither system have transmitted nor received data, this would have indicated that either one or both systems were inoperative.

2.3 Factors Influencing Component Selection

2.3.1 Servo Motors

Precise positional control: Servo motors have built-in feedback mechanisms and control circuits enabling accurate control of the motor's position. This precision is crucial for robotic arms, which require precise and repeatable movements.

Compact size: Servo motors are available in compact sizes, making them suitable for robotic arm designs where space is limited. They were easily integrated into the arm structure without adding excessive bulk.

Ease of use and compatibility: Servo motors are easy to use and interface with and control due to the Arduino h.Servo library which simplified their integration and programming.

Cost-effectiveness: Servo motors provided a good balance between performance and cost. They are more affordable than other high-performance motor options, such as stepper motors or linear actuators, while still offering the necessary capabilities for robotic arm applications.

2.3.2 Joysticks

Intuitive and ergonomic control: joysticks are designed with ergonomics in mind, providing a comfortable grip and intuitive hand placement. This ensures that users can control the robotic arm with ease and minimal fatigue over extended periods.

Two-dimensional control: Dual-axis joysticks provide control in two dimensions. This enabled precise control over the horizontal and vertical movement of the robotic arm.

Simultaneous control: Dual-axis joysticks allow for simultaneous control of both axes.

Compact and lightweight: Dual-axis joysticks are compact and lightweight allowing them to be easily integrated into the handheld control centre device.

Analogue input: Joysticks were used to provide a continuous range of values, which allowed for smooth and precise control over movements, which is not achievable with discrete inputs such as a keypad.

Real-time control: Dual-axis joysticks provided real-time control, allowing users to make immediate adjustments to the arm's position and movement.

2.3.3 Ultrasonic Sensors

Non-contact sensing: Ultrasonic sensors operate on a non-contact principle, meaning they do not require physical contact with objects to measure distance. This feature was advantageous for the robotic arm as it allows for safe and efficient object detection without the risk of collision or damage.

Fast response time: Ultrasonic sensors provide fast response times due to operating at the speed of sound.

Cost-effective: Ultrasonic sensors are cost-effective compared to other sensing technologies like lidar and vision systems. It offered a good balance between performance and cost.

2.3.4 nRF24L01+ Transceivers

Wireless communication: The transceivers provide wireless communication capabilities through radio frequencies; wireless communications was one of the objectives of this project.

Reliable and robust connectivity: The transceivers offered dependable and connectivity, ensuring stable and consistent communication between the robotic arm and control system.

Low power consumption: The transceivers are designed with low power consumption in mind. The low power consumption of the transceivers helps prolong the battery life of the robotic arm.

Long-range communication: The transceivers can provide long-range communication capabilities, allowing robotic arms to be controlled or monitored from a considerable distance.

Cost-effective: The transceivers provide a cost-effective solution for wireless communication and offered a good balance between performance and cost.

2.3.5 I2C LCD

Simplified wiring: I2C LCD modules utilize the I2C protocol for communication. Compared to traditional parallel LCDs, I2C LCD modules require fewer wires for connection.

Cost-effective: I2C LCD modules are cost-effective compared to other display technologies, such as graphical LCDs or OLED displays. It provided a good balance between functionality and cost.

2.4 Software Design Method:

The following paragraphs explain the method and process of designing and implementing the code and algorithms used for this project.

2.4.1 Requirements Analysis

The basic functionality and capabilities of the project were determined such as the range of motion, number of degrees of freedom, and specific tasks the project had to perform such as distance measurement and displaying the result.

2.4.2 System Architecture

The major components of the project were identified, such as the servo motors, joysticks, ultrasonic sensors, transceiver modules and LCD. Flowcharts were then developed to visualize how these components would work together to achieve the desired movements and functionality.

2.4.3 Algorithm Design

Algorithms for each subsystem were developed such as the joystick and servo motor control algorithm.

2.4.4 Code Organization

The individual algorithms were combined and organized into logical sections based on the different components and systems to simplify troubleshooting and subsystem integration.

2.4.5 Component and sensor Integration

The final project was constructed, and the code was uploaded to assess if the project functioned and met the specified objectives.

2.4.6 Testing and Iteration

The project was evaluated to ensure that the arm moved correctly and performed the desired tasks. The code was enhanced through an iterative method to refine and functionality and address issues.

2.5 Software Resources

2.5.1 Fritzing

The Fritzing software was used to design and draw out the circuit diagrams.

2.5.2 Arduino IDE

This software was used to develop, run, and upload the code to the microcontrollers to translate the change in position of the joystick into the actuation of the servo motors.

2.5.3 Diagrams.net

This software was utilized to design the system block diagram and flowchart for the servo motor and joystick subsystem.

2.5.4 Tinkercad

This software was used to evaluate and visualize the range of motion of the servo motors through a sweeping program.

Chapter 3: CONSTRUCTION

3.1 Preliminary information on joystick and servo motor construction

The joysticks and servo motors were included in the 4DOF robotic arm kit and were interfaced with their respective microcontrollers according to the circuit diagrams and the wiring configurations tables 3.2 and 3.3.

3.1.1 Joystick Specifications

The following are the joystick specifications from the manufacturer [7]:

- Dual-Axis, X and Y axis.
- Dimensions: 4 X 2.6 X 3.2cm.
- Two potentiometers for two axes.
- Connector Pins: +5Vcc, GND, VRx, VRy, SW.

The joysticks position was based on values between 0 and 1023 for both the X and Y axis, this is clearly depicted in figure 3.1.

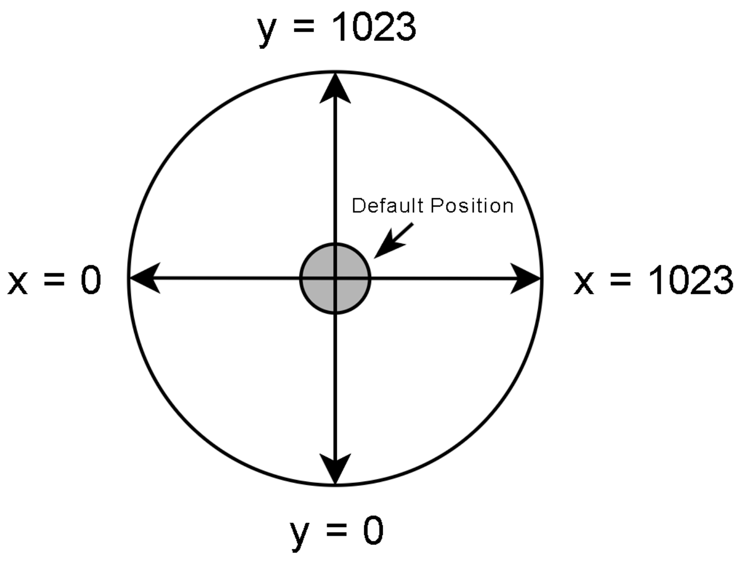


Figure 3. 1 Joystick Position Detection

The joystick values were represented as numbers from 0 to 1023 because they are measured using the Arduino Uno’s 10-bit ADC [8]. This means the joystick's position was divided into 1024 steps, starting from 0. The minimum position was represented by 0, and the maximum position was represented by 1023. These values were used to accurately track the joysticks’ movement.

3.1.2 Servo Motor

As shown in figure 3.2, the servomotors consist of a regular DC motor which was controlled by a built-in control circuit [9]. The servo's torque and speed are set by the gears and lastly the position potentiometer provides feedback on the angle of rotation of the servo motor [9, 10].

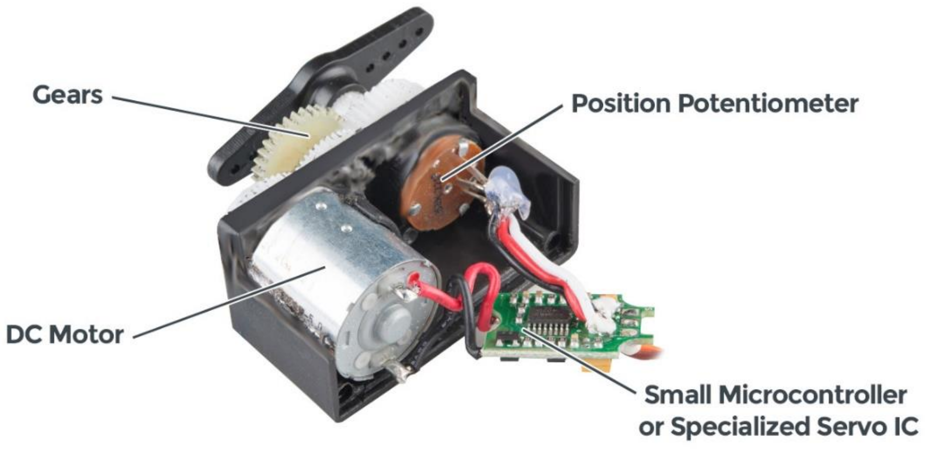


Figure 3. 2 Servo Motor Deconstructed [9]

The microcontroller maps the position of the joystick to an angle between 0° and 180°. This angle was then used to generate a PWM signal which dictated the angle of rotation of the servo motor [9, 10].

The following are servo motor specifications from the manufacturer [11]:

- Operating voltage: DC 4.8V～6V
- Angle range: 180°
- Pulse width range: 500→2500µsec
- No-load speed: 0.12±0.01 sec/60 （DC4.8V）；0.1±0.01 sec/60 （DC6V）
- No-load current: 200±20mA （DC4.8V）；220±20mA （DC6V）
- Stop torque: 1.3±0.01kg/cm (DC4.8V）；1.5±0.1kg/cm (DC6V)
- Stop current: ≦850mA （DC4.8V）；≦1000mA （DC6V）
- Standby current: 3±1mA （DC4.8V）；4±1mA （DC6V）
- Operation temperature: -10℃～50℃
- Motor wire length: 250 ± 5 mm
- Dimensions: 22.9mm*12.2mm*30mm
- Weight: 9± 1 g (without servo mounts)

3.2 Individual role and tasks in construction

The role of the electromechanical servo motor engineer for this project required interfacing the servo motors and joysticks with their respective microcontrollers as well, adjusting the screw tensions to ensure the robotic arm moved smoothly and to ensure that the housings did not disrupt the movement of the joysticks or servo motors.

Furthermore, a prototype for the joystick and servo motor subsystem was developed where the servo motors and joysticks were connected to a single microcontroller to assess the functionality and interactions as show in figure 3.3.

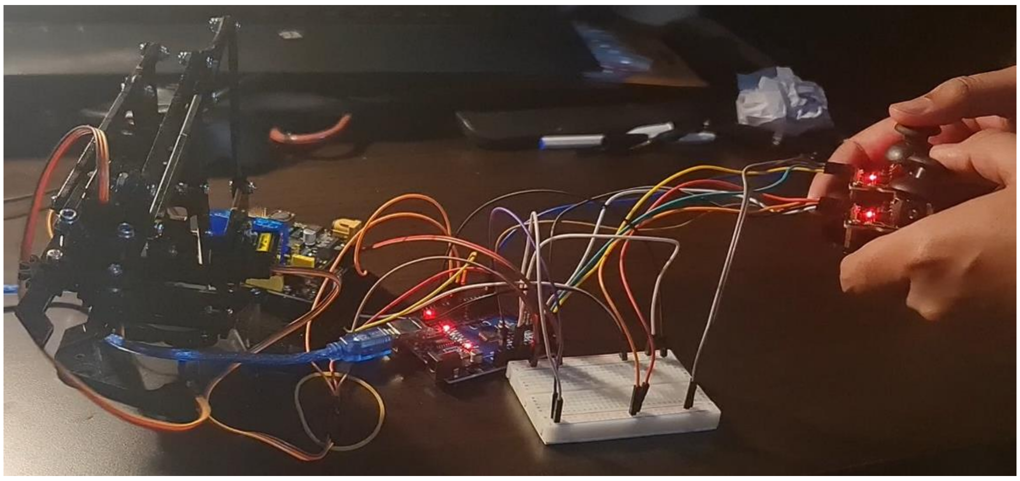


Figure 3. 3 Constructed Prototype for Servo Motor and Joystick Subsystem

3.3 Construction Overview

Preliminary circuit designs for subsystems and prototype were created using the Fritzing software. Group members constructed subsystems according to their designated roles to ensure proper working of the components, demonstrate an understanding of their assigned tasks and limit errors incurred by the final project artefact.

Following the successful testing of each subsystem, a prototype was constructed by combining the individual subsystems into either the control centre system or the robotic arm system.

The construction process involved assembling the robotic arm structure according to the manual provided with the 4DOF Robotic Arm kit as well as interfacing the electronic and mechanical components together and lastly uploading the codes in annexures C and D which were developed using the flowcharts in annexures E and F.

Furthermore, the servo motors were mounted onto the robotic arm structure and functioned as joints. The base servo motor rotated the base of the robotic arm allowing for horizontal movement while the shoulder and elbow servo motors raised and lowered the arm's linkages allowing for vertical movement.

The vertical distance measuring ultrasonic sensor was mounted on the robotic arm structure in place of the claw, while the horizontal distance measuring sensor was mounted at the base on the robotic arm.

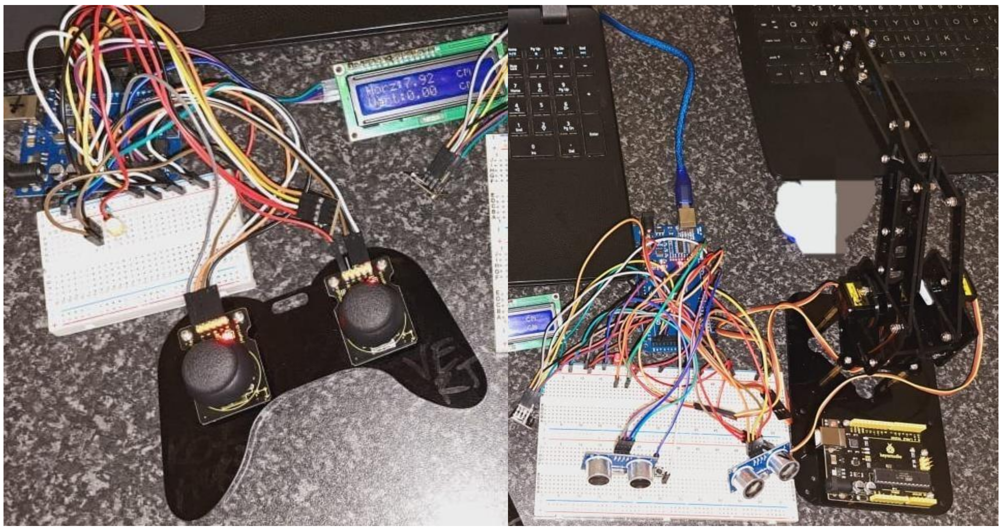


Figure 3. 4 Constructed Prototype for Control Centre and Robotic Arm

The project was assembled according to the designed circuit schematics and the wiring configurations in tables 3.1 to 3.3.

Table 3. 1 Transceiver Module Pinout

Transceiver pin	Arduino ATmega328P pin number	Arduino Uno Rev 3 pin number
VCC	3.3V	3.3V
Ground	GND	GND
CE	7	9
CSN	8	10
SCK	52	13
MOSI	51	11
MISO	50	12

Table 3. 2 Robotic Arm Wiring Configuration

Component pin	Arduino Uno pin number
VCC	5V
Ground	GND
Ultrasonic sensor 1 VCC	5V
Ultrasonic sensor1 GND	VCC
Ultrasonic sensor1 VCC	GND
Ultrasonic sensor1 TRIG	2
Ultrasonic sensor1 ECHO	3
Ultrasonic sensor2 VCC	VCC
Ultrasonic sensor2 GND	GND
Ultrasonic sensor2 TRIG	4
Ultrasonic sensor2 ECHO	5
Red LED Anode	11
Red LED Cathode	GND
Green LED Positive Pin	12
Green LED Anode	GND
Push Button Pin	6
Push Button GND Pin	GND
Base servo	7
Shoulder servo	9
Elbow servo	8

Table 3. 3 Control Centre Wiring Configuration

Component pin	Arduino UNO pin number
VCC	5V
Ground	GND
I2C Module VCC	5V
I2C Module GND	GND
I2C Module SCL	A5
I2C Module SDA	A4
Yellow LED Positive Pin	5
Yellow LED Negative Pin	220Ω Resistor to GND
Joystick1 VCC Pin	5V
Joystick1 GND Pin	GND
Joystick1 X Pin	A0
Joystick1 Y Pin	A1
Joystick2 VCC Pin	5V
Joystick2 GND Pin	GND
Joystick2 Y Pin	A2
Joystick1 Button	4

Lastly, housing for both the robotic arm and control centre system were constructed to provide a layer of protection to the components. Lastly through an iterative method, the prototype was refined into the final artefact as can be seen in figure 3.5.

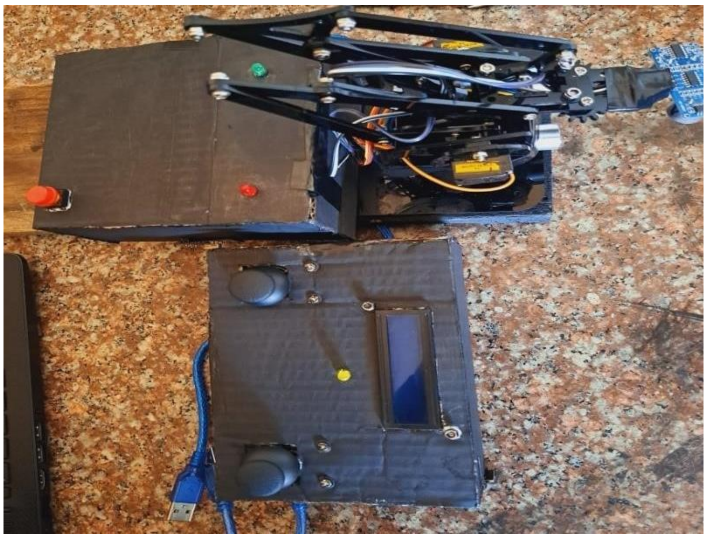


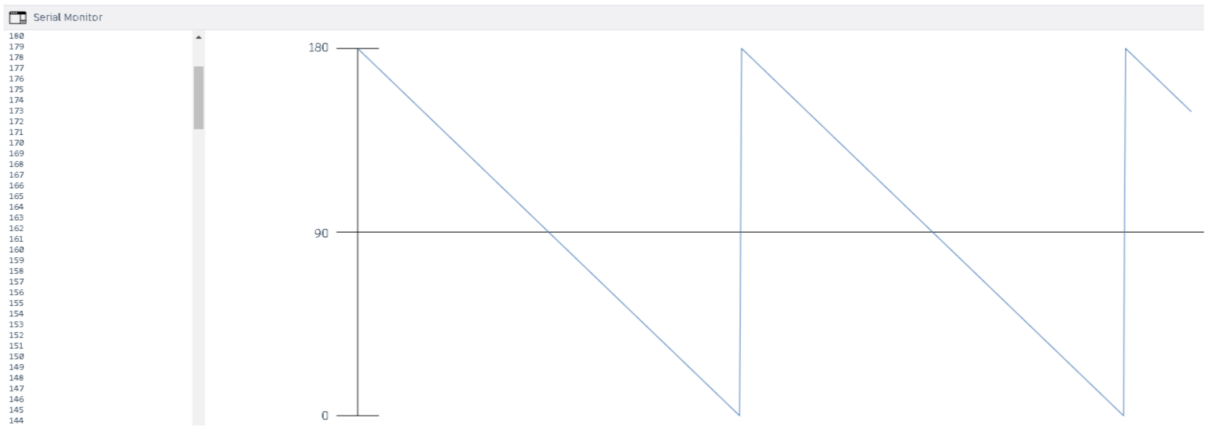
Figure 3. 5 Constructed Final Prototype for Control Centre and Robotic Arm

Chapter 4: TESTING

4.1 Servo Motor Range of Motion Testing

The code used in annexure A was used to assess the full range of motion of the servo motors using a sweep function that would increment and change the angle of the servo motor incrementally. Subsequently, the serial monitor and plotter in figure 4.1 depict the results of this test. As can be seen, the servo motor rotated from 0° to 180° and then from 180° to 0° repeatedly.

The sawtooth waveform occurs due to the delay where the servo motor stops for fifteen microseconds and then flips the direction it rotates in, the serial plotter registered this as the servo motor starting at 180° again.



```
180
179
178
177
176
175
174
173
172
171
170
169
168
167
166
165
164
163
162
161
160
159
158
157
156
155
154
153
152
151
150
149
148
147
146
145
144
```

Figure 4. 1 Serial Monitor and Plotter for Servo Sweep Code

Furthermore, the codes in annexure I and J were used to calibrate the servo motors minimum and maximum angles.

4.2 Subsystem Movement testing

An initial prototype circuit was constructed as shown in figures 4.2 and 4.3. The circuit consisted of the robotic arm, servo motors, joysticks, and a single microcontroller. This test circuit was built to isolate the servo motors and joysticks from the other subsystems within this project. This was done to evaluate the aforementioned components and ensure that this subsystem was operational to lower the risk of causing or incurring errors when the final project was constructed. Furthermore, each group member performed a similar task with their respective subsystems where applicable.

Annexures G and K depict the subsystem flowchart and code, respectively. The code functions by reading the changing position of the joystick as an analogue input and converting it to a PWM signal used to control the rotation of the servo motors. By pressing the left joystick button, the servo motors enter hold mode where the position of the robotic arm does not change regardless of joystick input. To exit hold mode, the joystick button simply had to be pressed again.

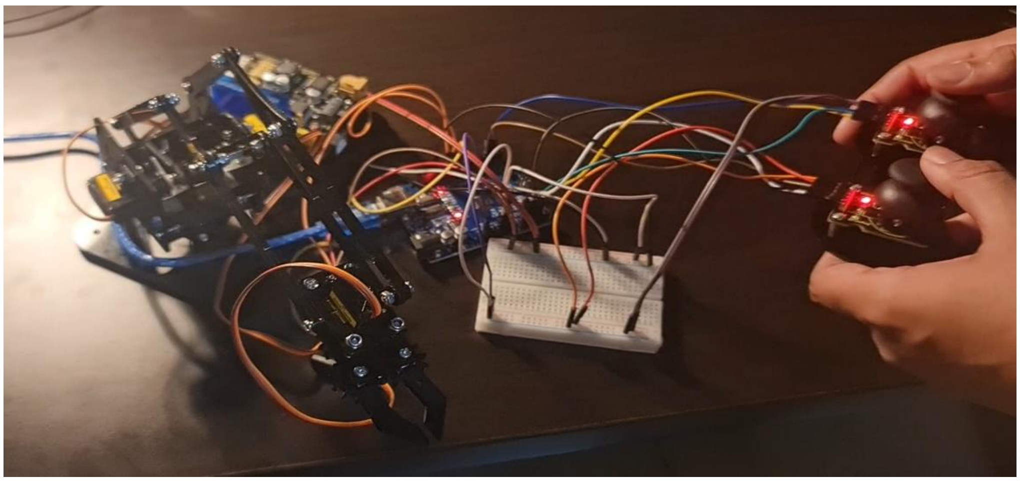


Figure 4. 2 Servo Motor and Joystick Subsystem Prototype

4.3 Project Testing

Once all the subsystems were interfaced the projects control centre and robotic arm systems were powered on. The joysticks were used to control the movement of the robotic arm. Once the arm was in the desired position, a push button was pressed once to read the vertical distance measurements using the ultrasonic sensor 1 and when pushed twice ultrasonic sensor 2 was used to take the horizontal distance measurements.

For vertical distance measurements, the arm's vertical position was changed relative to the surface the robotic arm rested on as this surface was the obstacle that ultrasonic sensor 1 detected. However, for the horizontal distance measurement, an obstacle was introduced for ultrasonic sensor 2 to measure the distance. The arm would move into the desired position and the relevant distance measurements were taken as shown in figures 4.4 and 4.5.

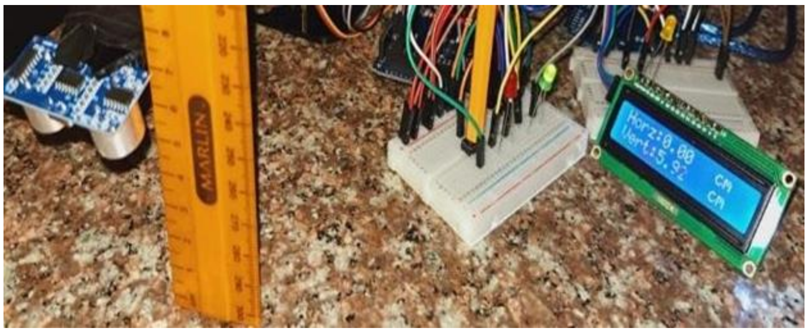


Figure 4. 3 Vertical Distance Measurement Test

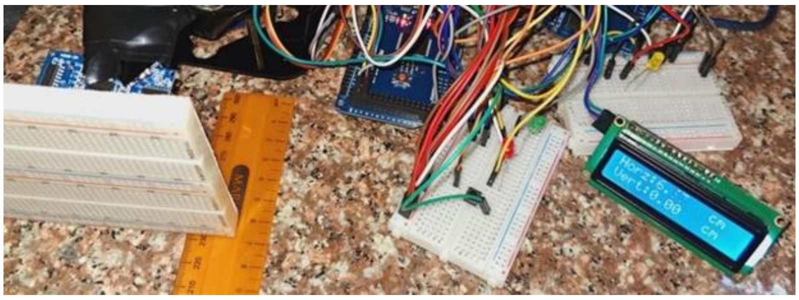


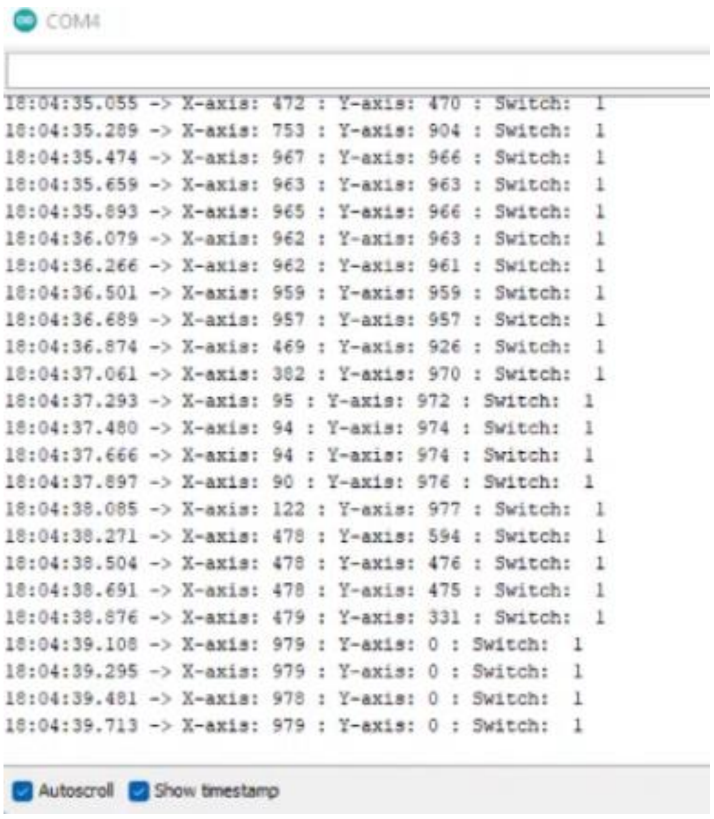
Figure 4. 4 Horizontal Distance Measurement Test

To verify the systems were functional, the LCD was used to display the distance measurements and LED indicators depicted when a distance measurement was being done as well as when successful data transfers had occurred between the control centre and the robotic arm systems.

Chapter 5: RESULTS

5.1 Servo Motor and Joystick results

The servo motors actuated the joysticks precisely and accurately as intended. Figure 5.1 depicts the serial monitor used to capture the changing position of the joysticks. The X and Y values correspond to the joystick position as depicted in figure 3.1.



```
18:04:35.055 -> X-axis: 472 : Y-axis: 470 : Switch:  1
18:04:35.289 -> X-axis: 753 : Y-axis: 904 : Switch:  1
18:04:35.474 -> X-axis: 967 : Y-axis: 966 : Switch:  1
18:04:35.659 -> X-axis: 963 : Y-axis: 963 : Switch:  1
18:04:35.893 -> X-axis: 965 : Y-axis: 966 : Switch:  1
18:04:36.079 -> X-axis: 962 : Y-axis: 963 : Switch:  1
18:04:36.266 -> X-axis: 962 : Y-axis: 961 : Switch:  1
18:04:36.501 -> X-axis: 959 : Y-axis: 959 : Switch:  1
18:04:36.689 -> X-axis: 957 : Y-axis: 957 : Switch:  1
18:04:36.874 -> X-axis: 469 : Y-axis: 926 : Switch:  1
18:04:37.061 -> X-axis: 382 : Y-axis: 970 : Switch:  1
18:04:37.293 -> X-axis: 95 : Y-axis: 972 : Switch:  1
18:04:37.480 -> X-axis: 94 : Y-axis: 974 : Switch:  1
18:04:37.666 -> X-axis: 94 : Y-axis: 974 : Switch:  1
18:04:37.897 -> X-axis: 90 : Y-axis: 976 : Switch:  1
18:04:38.085 -> X-axis: 122 : Y-axis: 977 : Switch:  1
18:04:38.271 -> X-axis: 478 : Y-axis: 594 : Switch:  1
18:04:38.504 -> X-axis: 478 : Y-axis: 476 : Switch:  1
18:04:38.691 -> X-axis: 478 : Y-axis: 475 : Switch:  1
18:04:38.876 -> X-axis: 479 : Y-axis: 331 : Switch:  1
18:04:39.108 -> X-axis: 979 : Y-axis: 0 : Switch:  1
18:04:39.295 -> X-axis: 979 : Y-axis: 0 : Switch:  1
18:04:39.481 -> X-axis: 978 : Y-axis: 0 : Switch:  1
18:04:39.713 -> X-axis: 979 : Y-axis: 0 : Switch:  1
```

Figure 5. 1 Joystick Position Results

These values were mapped to an angle between 0° and 180° to actuate the servo motors. This angle was used to generate a PWM signal to actuate the servo motors to the desired angles. Figure 3.2 depicts the relationship between angle of the servo motor’s rotation and the pulse width of the signal sent to the servo motor.

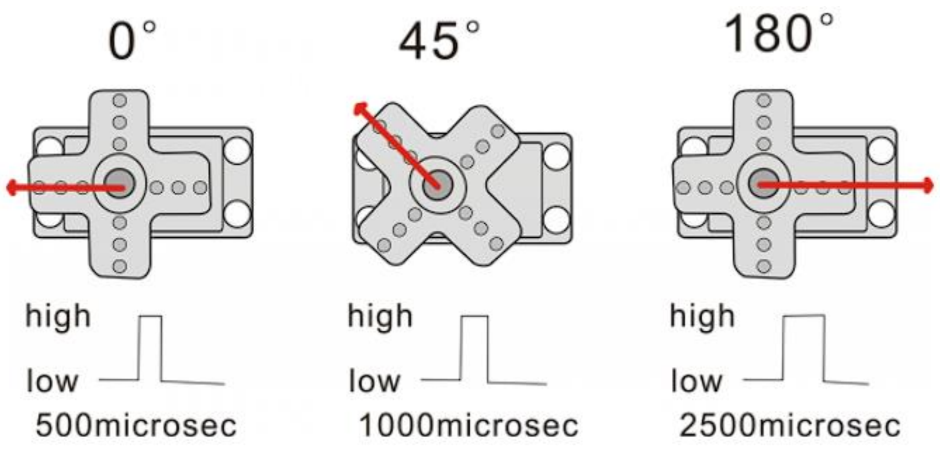


Figure 5. 2 Relationship Between Pulse Width and Servo Rotation [11]

5.2 Project Results

The measurement results obtained were accurate as expected however, there were discrepancies caused by servo jitter, which are erratic movements of the servo motors caused by their high operating speeds. Insufficient screw tension also inhibited the desired motion of the robotic arm however once these issues were addressed, the robotic arm's movement was smoother and more controlled.

Further discrepancies were caused by the ultrasonic sensor not being perfectly parallel to the measured object as well as possible electrical noise as well as radio frequency interference from the transceiver modules.

Table 5.1 shows the largest and the least variations that were measured during the testing phase. Furthermore, equation 5-1 explains how the percentage error was calculated.

$$\%E = \frac{Ultrasonic\ Sensor\ Distance - Ruler\ Distance}{Ultrasonic\ Sensor\ Distance} \times 100\% \qquad [5\text{-}1]$$

Table 5. 1 Final Project Distance Measurement Results

Measurement	Ultrasonic sensor	Ruler	Difference in measurement	% error
Vertical distance	5.92 cm	5.4 cm	0.52	8.8%
	9.06 cm	9 cm	0.06	0.66%
Horizontal distance	6.24 cm	6 cm	0.24	3.8%
	9.17 cm	8.8 cm	0.37	4%

Figures 5.3 and 5.4 show the testing procedure for horizontal and vertical distance measurement, respectively. The green LED indicates horizontal distance measurement mode while the red LED indicates vertical distance measurement mode.

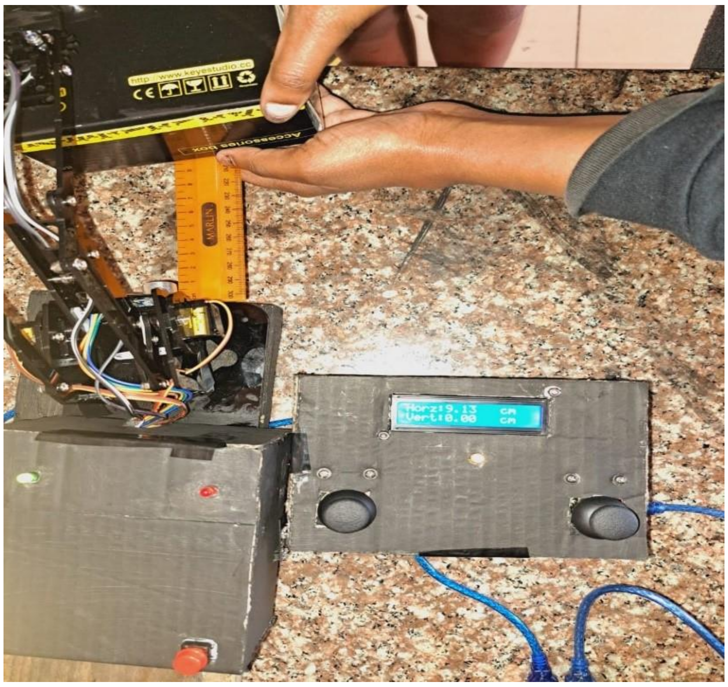


Figure 5. 3 Final Project Horizontal Distance Measurement

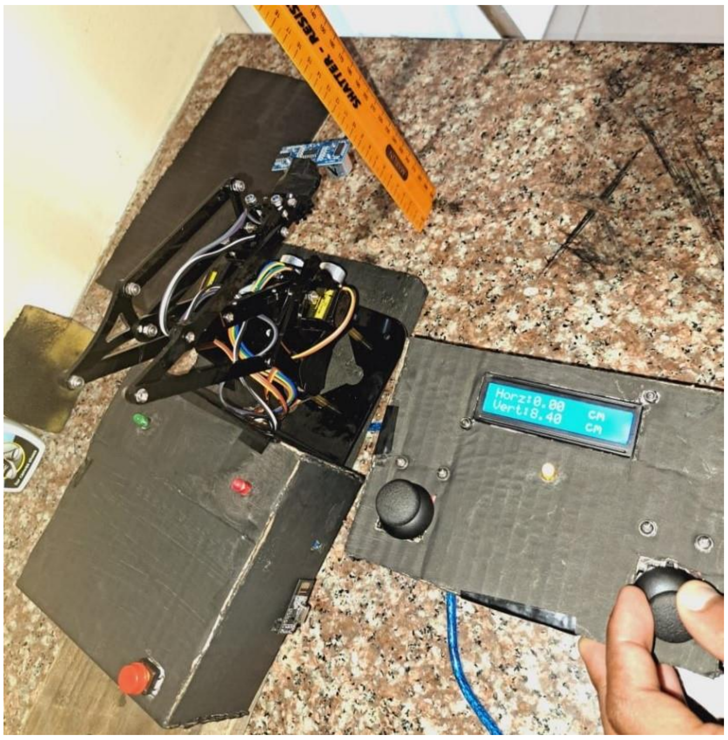


Figure 5. 4 Final Project Horizontal Distance Measurement

Chapter 6: PROBLEMS AND SOLUTIONS

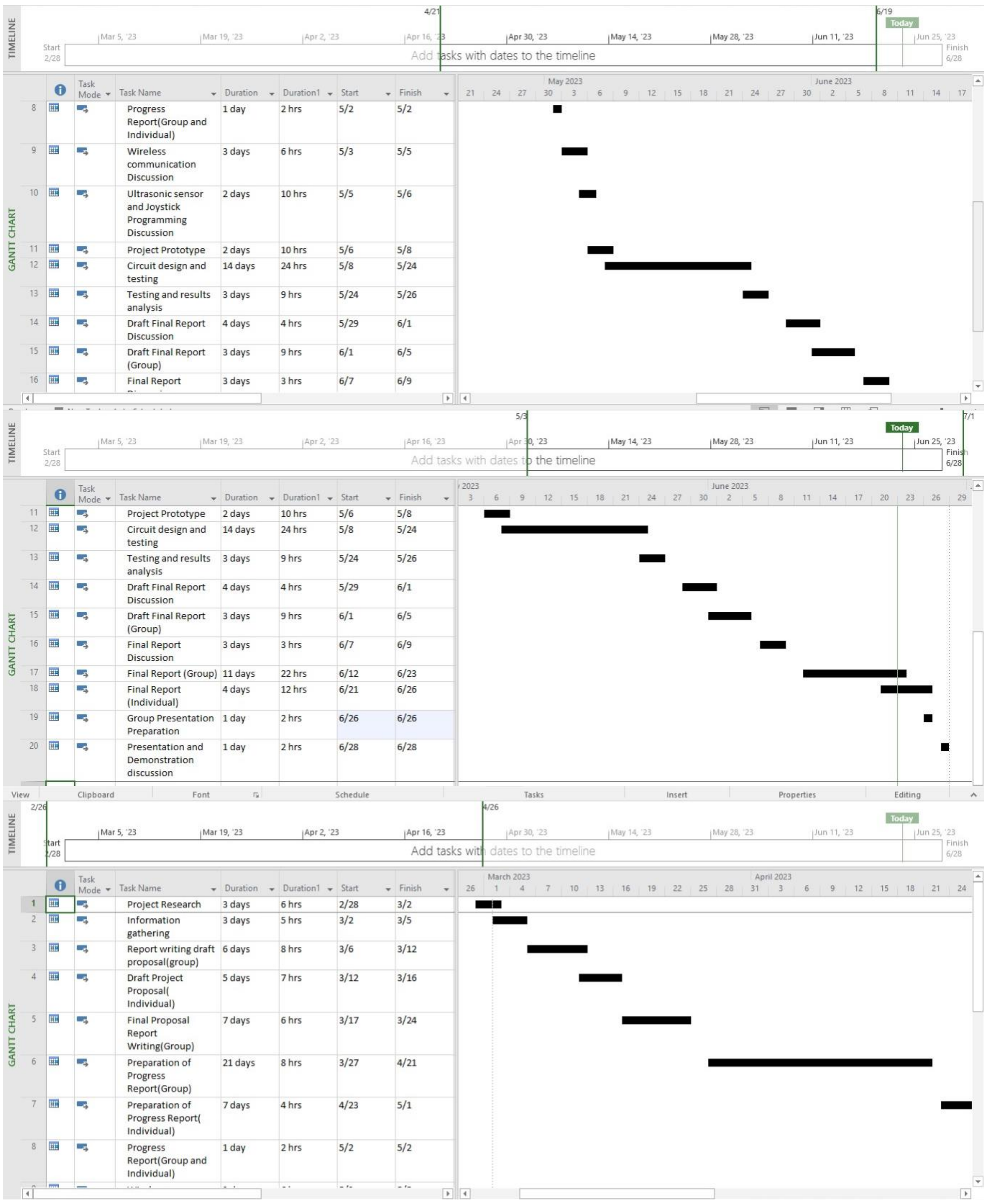
Table 6. 1 Problems and Solutions regarding servo motors and joysticks

Problem	**Solution**
The servo motors experienced jitter and moved rapidly during operation.	A 50ms delay was added to the servo motor operation to enable smoother movement of the robotic arm.
The robotic arm would not retain the desired position.	A hold function was implemented within the code to allow measurements to be taken accurately and consistently.
The Arduino Uno microcontroller had an insufficient number of digital pins for the robotic arm system due to the inclusion of additional upgrades to the base project such as an additional ultrasonic sensor and indicator LEDs.	To accommodate the servo motors as well as the additional components, the Arduino Uno was substituted with an Arduino Mega 2560 to facilitate the additional upgrades and leave room for further expansion should the need arise.
Poor movement rangeability of the initial prototype.	Code was rewritten to facilitate the full range of motion of the servo motors to allow for a greater movement rangeability of the robotic arm.
The robotic arm movement was erratic and inconsistent upon first construction and testing due to inadequate screw tightness.	The screws of the robotic arm were readjusted to allow for smoother movement.
Transportation limitations were faced.	The group conducted online meetings when applicable and communicated regularly on WhatsApp.

Chapter 7: PROJECT MANAGEMENT

7.1 Group Schedule

The group schedule is a culmination of all individual and group tasks such as project research, subsystem development, report writing, prototyping, and presenting.



	Task Name	Duration	Duration1	Start	Finish
1	Project Research	3 days	6 hrs	2/28	3/2
2	Information gathering	3 days	5 hrs	3/2	3/5
3	Report writing draft proposal(group)	6 days	8 hrs	3/6	3/12
4	Draft Project Proposal(Individual)	5 days	7 hrs	3/12	3/16
5	Final Proposal Report Writing(Group)	7 days	6 hrs	3/17	3/24
6	Preparation of Progress Report(Group)	21 days	8 hrs	3/27	4/21
7	Preparation of Progress Report(Individual)	7 days	4 hrs	4/23	5/1
8	Progress Report(Group and Individual)	1 day	2 hrs	5/2	5/2
9	Wireless communication Discussion	3 days	6 hrs	5/3	5/5
10	Ultrasonic sensor and Joystick Programming Discussion	2 days	10 hrs	5/5	5/6
11	Project Prototype	2 days	10 hrs	5/6	5/8
12	Circuit design and testing	14 days	24 hrs	5/8	5/24
13	Testing and results analysis	3 days	9 hrs	5/24	5/26
14	Draft Final Report Discussion	4 days	4 hrs	5/29	6/1
15	Draft Final Report (Group)	3 days	9 hrs	6/1	6/5
16	Final Report Discussion	3 days	3 hrs	6/7	6/9
17	Final Report (Group)	11 days	22 hrs	6/12	6/23
18	Final Report (Individual)	4 days	12 hrs	6/21	6/26
19	Group Presentation Preparation	1 day	2 hrs	6/26	6/26
20	Presentation and Demonstration discussion	1 day	2 hrs	6/28	6/28

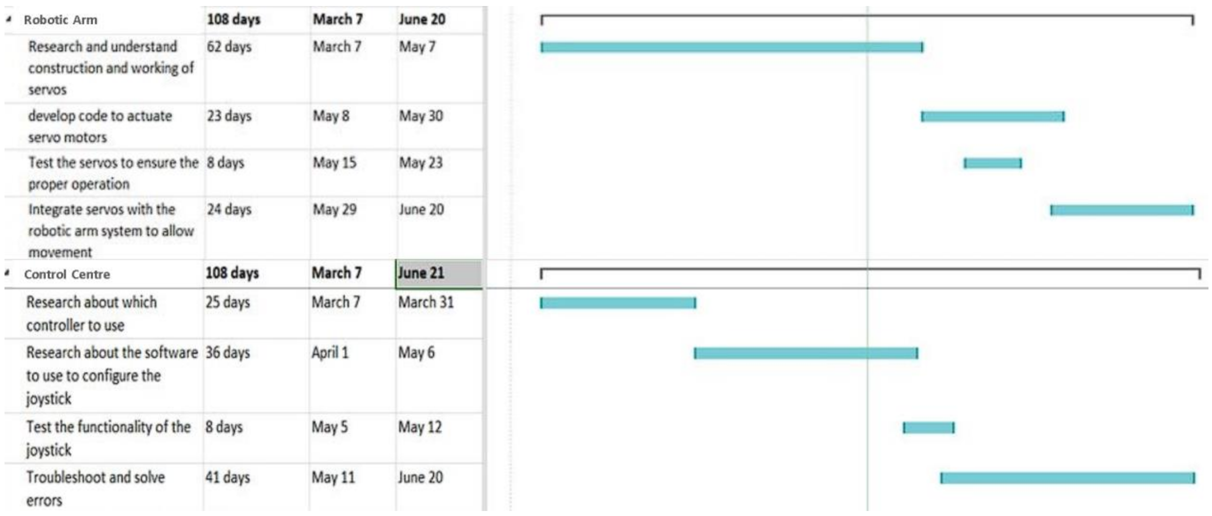
Figure 7. 1 Group Gantt Chart

Overall, the project deadlines were met in due time due to the time management skills of Group A's members. In general, each group member performed their required tasks within their designated time frame with negligible discrepancies. This allowed the group to functioning smoothly with minimal disruptions.

7.2 Individual Schedule

The individual tasks of the Electro-Mechanical Servo Motor Engineer were distributed between the two main systems of the project since the servo motors were output devices for the robotic arm system while the joysticks were input devices for the control centre.

Thus, initial research on the inner machinations and implementations of the joysticks and servo motors were necessary to design circuits and flowcharts as well as to understand in detail the processes and operations that the servo motors and joysticks would undergo. Following this, Arduino code was developed to assess and integrate the aforementioned components into their respective systems. Lastly troubleshooting was performed and errors that were incurred were appropriately dealt with as discussed in table 6.1.



Robotic Arm	**108 days**	**March 7**	**June 20**
Research and understand construction and working of servos	62 days	March 7	May 7
develop code to actuate servo motors	23 days	May 8	May 30
Test the servos to ensure the proper operation	8 days	May 15	May 23
Integrate servos with the robotic arm system to allow movement	24 days	May 29	June 20
Control Centre	**108 days**	**March 7**	**June 21**
Research about which controller to use	25 days	March 7	March 31
Research about the software to use to configure the joystick	36 days	April 1	May 6
Test the functionality of the joystick	8 days	May 5	May 12
Troubleshoot and solve errors	41 days	May 11	June 20

Figure 7. 2 Individual Role Gantt Chart

7.3 Cost and Quantity

The following tables display the proposed costs and additional project costs, respectively. Thus, the project did expend more capital than intended due to the oversight of the costs of construction of the housings as well as damaged or insufficient components. However, the additional capital was used to ensure proper working of the project and to enhance the overall functionalities of the project and provide expansion possibilities for future iterations of the project should the need arise.

Table 7. 1 Proposed Project Costs

Component	Cost per Unit	Quantity	Total	Supplier
Arduino Mega 2560	R429,95	1	R429,95	DIY Electronics
NRF24L01+ WIRELESS MODULE	R42,21	2	R84,42	Mantech
DIY 4DOF Robotic Mechanical Arm Kit for Arduino	R1 533,86	1	R1 533,86	Mantech
LCD Display Module 1602 With I2C Interface	R143,91	1	R143,91	Mantech
HC-SR04 Ultrasonic Reflective Sensor	R52,03	1	R52,03	Mantech
M/F Jumper Wires Pack	R28,59	1	R28,59	Mantech
M/M Jumper Wires Pack	R31,07	1	R31,07	Mantech
85x55mm Breadboard	R55,72	2	R111,44	Mantech
Total Proposed Project Cost			R2 415,27	

Table 7. 2 Final Project Costs

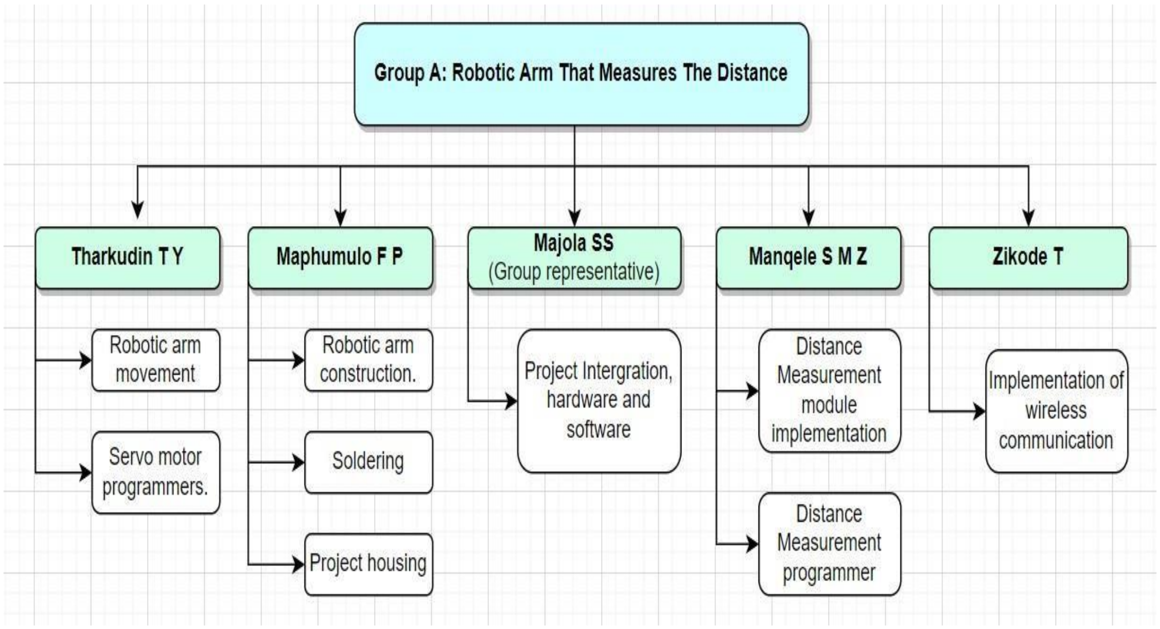
Components and materials	Cost (R)	Quantity
Arduino Mega	549	1
Black paint	45	1
Tape	21	2
Soldering iron	150	1
nRF24L01+	25	1

The 4DOF Robotic Arm Kit was by far the highest priced item, however included in the kit were the servo motors, joysticks, a microcontroller and shield and various construction equipment and linkages as can be seen in annexure J.

7.4 Organogram and roles

7.4.1 Group Roles

Each member had a designated task, based on the organogram in figure 7.1, that they were required to complete in a specific time frame according to the group Gantt chart.



Group A: Robotic Arm That Measures The Distance
Tharkudin T Y
Maphumulo F P
Majola SS (Group representative)
Manqele S M Z
Zikode T
Robotic arm movement
Servo motor programmers.
Robotic arm construction.
Soldering
Project housing
Project Intergration, hardware and software
Distance Measurement module implementation
Distance Measurement programmer
Implementation of wireless communication

Figure 7. 3 Group Organogram

Table 7. 3 Group Members and Their Designated Tasks

Group Member	**Designated Engineering Tasks**
Thakurdin T Y	Electro-Mechanical Servo Motor Engineer and Arduino programmer
Maphumulo F P	Electro-Mechanical Assembly Technician
Maloja S S	Integration Specialist and Arduino Programmer
Manqele S M Z	Ultrasonic Distance Measurement Specialist and Arduino Programmer
Zikode T	Wireless Communication Specialist and Arduino Programmer

7.4.2 Individual Role

The role of an Electro-Mechanical Servo Motor Engineer and Arduino programmer within this project was to interface the servo motors and joysticks with their respective systems, develop code to translate the changing X and Y positions of the joystick into the actuation of the servo motors and lastly to troubleshoot errors incurred by this subsystem.

Chapter 8: CONCLUSION

8.1 Overall Conclusion

In conclusion, by harnessing the precision and ergonomics of joysticks as input devices and with the precise positioning capabilities of servo motors as output devices, operators gain the ability to exert intuitive and efficient control over the movement and positioning of the robotic arm which allowed for smooth and accurate motion.

One key advantage of this remote-control system was the enhanced manoeuvrability it offers because operators can achieve precise movements without worry of being in the path of the arm's movement. Moreover, the implementation of the remote-control functionality contributes to improved safety. By eliminating the need for direct physical contact with the machinery, operators can safely control the robotic arm in hazardous environments or hard-to-reach locations. This remote operation capability enhances overall safety by minimizing potential risks associated with an operator being in close proximity to moving parts.

In summary, the combination of servo motors with joysticks provided a user-friendly, precise, and reliable solution for the control of the project's robotic arm. Furthermore, by facilitating the movement of the robotic arm, the joysticks and servo motors allowed the ultrasonic sensors to take various distance measurements in numerous positions. Thus, the objectives that were set for this project were successfully met which resulted in the final project meeting the design requirements and functioning as intended.

8.2 Decisions Made

An additional ultrasonic sensor was integrated into the robotic arm system, allowing accurate distance measurement in both vertical and horizontal dimensions.

To accommodate the additional components, including the second ultrasonic sensor, the Arduino Uno was replaced with the Arduino ATmega328P due to its increased digital pin capacity. This enabled seamless integration of enhancements and potential future expansion.
Indicator LEDs were added to visually indicate data transmission and sensor readings, simplifying monitoring of operations.

To enhance the overall functionality and aesthetics of the control centre and robotic arm, a housing was designed and constructed. The housing reduced wire clutter and provided protection to delicate components as well as presented a neat and organized appearance.

8.3 Observations

- The intuitive nature of joysticks as input devices enabled operators to swiftly assimilate to their functionality, which allowed the operators to quickly grasp how to use them effectively.
- The servo motors operate at high speeds which can cause jitter however a small delay should be used to counteract this.
- Screw tension of the robotic arm can restrict the motion of the robotic arm if too tight or diminish control of the movements if too loose.

8.4 Outcomes

- The project was successfully completed within the allocated time frame.
- Project aim and objectives were achieved.
- Resources, including time, budget, and labour were used effectively to complete the project.
- Through an iterative method the project's quality and efficiency was improved.
- Intuitive operator control was achieved.
- Valuable insights on the importance of teamwork and co-operation when solving complex engineering problems were gained.

8.5 Recommendations

Should the robotic arm be required to lift an external load, the claw servo motor should be implemented. In addition to this, servo motors with a higher torque should be used depending on the expected operating loads.

Collision prevention should be implemented. The arm should use its sensors to detect obstacles in its path and the user input from the joysticks should be overridden, following this, the servos should either avoid the obstacle or stop before contacting it.

The elbow and servo motor should actuate between 0° and 90° to prevent the arm from unintentionally contacting the ground and damaging the sensors.

REFERENCES

[1] J. Flynt, “What are robotic arms and how do they work?,” *3D Insider*, 2 July 2019. [Online]. Available: https://3dinsider.com/what-are-robotic-arms. [Accessed 25 June 2023].

[2] P. Shepley, “What is a robotic arm?,” *EasyTechJunkie*, 21 February 2023. [Online]. Available: https://www.easytechjunkie.com/what-is-a-robotic-arm.htm. [Accessed 25 June 2023].

[3] S. Daley, "Robotics: What Are Robots? Robotics Definition & Uses," *Built In*, 18 August 2022. [Online]. Available: https://builtin.com/robotics [Accessed 25 June 2023].

[4] Y. Stolworthy, “Robots Can Make Great Surgical Assistants,” *Medical Daily*, 2 September 2020. [Online]. Available: https://www.medicaldaily.com/robotic-surgery-helps-surgeons-perform-more-precise-surgery-455742. [Accessed 25 June 2023].

[5] A. Jones, “On China's new space station, a robotic arm test paves way for future construction,” *Space.com*, 11 January 2022. [Online]. Available : https://www.space.com/china-space-station-robotic-arm-construction-test [Accessed 25 June 2023].

[6] M. Stevens, “Pros and cons of using industrial robots in your manufacturing operation,” *WIPFLI*, 7 September 2021. [Online]. Available: https://www.wipfli.com/insights/articles/mad-pros-and-cons-of-using-industrial-robots-in-manufacturing [Accessed 25 June 2023].

[7] "KY-023 Dual-axis Joystick Module." *Keyestudio* [Online]. Available: https://www.mantech.co.za/datasheets/products/KY-023-220909A.pdf [Accessed 29 June 2023].

[8] S. R. Karthiga, “How To Connect Arduino Joystick In Arduino Uno,” *C# Corner* [Online]. https://www.c-sharpcorner.com/article/how-to-connect-arduino-joystick-in-arduino-uno/ [Accessed 29 June 2023].

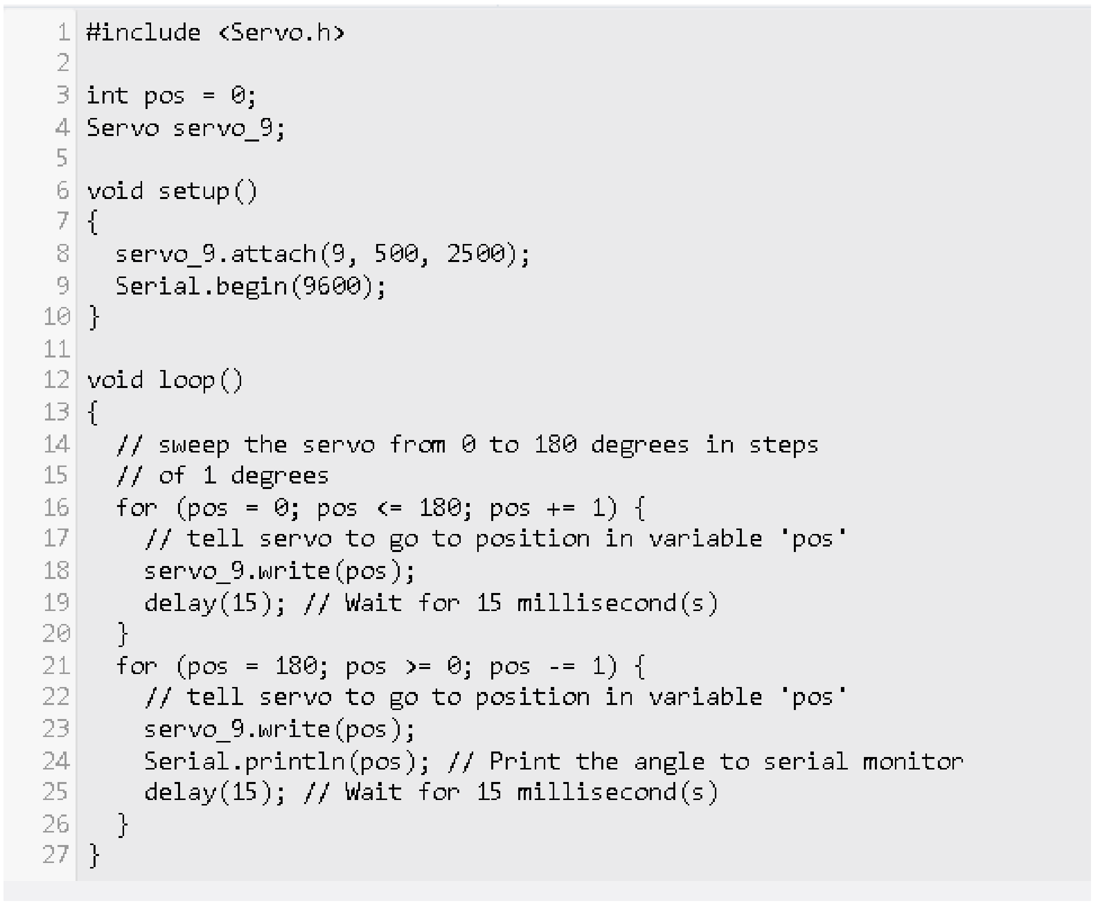
[9] “Servos Explained,” *SparkFun Electronics* [Online]. Available: https://www.sparkfun.com/servos [Accessed 29 June 2023].

[10] P. Evans, “Servo Motor Explained,” *The Engineering Mindset*, 23 January 2022. [Online]. Available: https://theengineeringmindset.com/servo-motor-xplained/. [Accessed 29 June 2023].

[11] “KS0194 Micro Servo.” *Keyestudio* [Online]. Available: https://wiki.keyestudio.com/Ks0194_keyestudio_Micro_Servo [Accessed 29 June 2023].

ANNEXURES

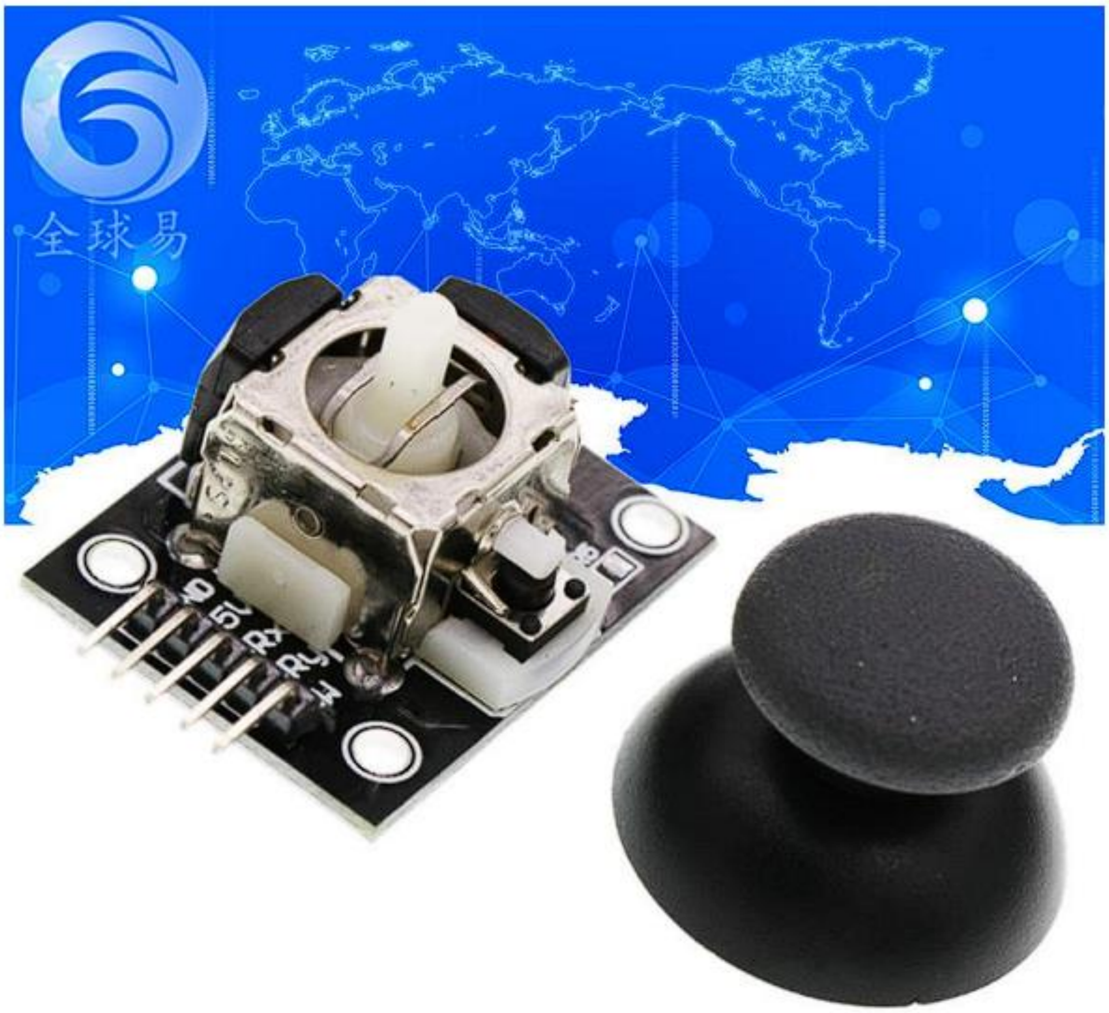
Annexure A: Servo motor sweep code for range of motion evaluation



```
#include <Servo.h>

int pos = 0;
Servo servo_9;

void setup()
{
  servo_9.attach(9, 500, 2500);
  Serial.begin(9600);
}

void loop()
{
  // sweep the servo from 0 to 180 degrees in steps
  // of 1 degrees
  for (pos = 0; pos <= 180; pos += 1) {
    // tell servo to go to position in variable 'pos'
    servo_9.write(pos);
    delay(15); // Wait for 15 millisecond(s)
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    // tell servo to go to position in variable 'pos'
    servo_9.write(pos);
    Serial.println(pos); // Print the angle to serial monitor
    delay(15); // Wait for 15 millisecond(s)
  }
}
```

Annexure B: Joystick Datasheet

KY-023 Dual-axis Joystick Module



Product Description

- Dual-axis XY Joystick Module
- Dimensions:
- Size: 4 X 2.6 X 3.2CM
- 2 potentiometers for 2 axes
- 1 switch
- Connector: +5Vcc - GND - VRx - VRy - SW

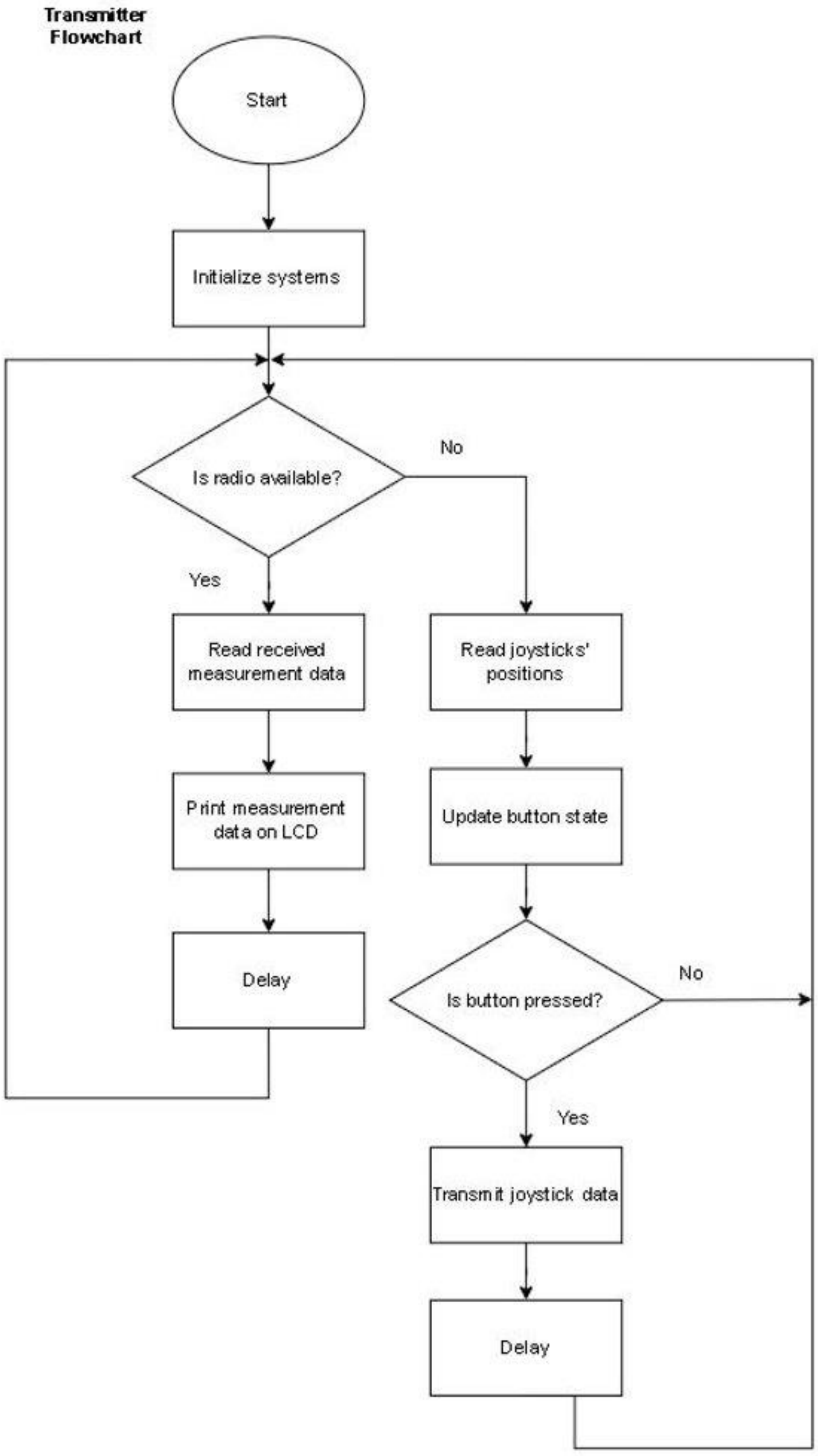
Annexure C: Robotic Arm code

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>

RF24 radio(10, 53); // CE, CSN pins
const byte address[6] = "00001"; // Address for communication

Servo servo1;
Servo servo2;
Servo servo3;

const int trigPinHorizontal = 2; // Trig pin for horizontal ultrasonic sensor
const int echoPinHorizontal = 3; // Echo pin for horizontal ultrasonic sensor
const int trigPinVertical = 4; // Trig pin for vertical ultrasonic sensor
const int echoPinVertical = 5; // Echo pin for vertical ultrasonic sensor

const int buttonPin = 6; // Push button pin
bool isButtonPressed = false; // Button state
bool buttonClickedOnce = false; // Flag to track single click
bool buttonClickedTwice = false; // Flag to track double click
unsigned long buttonClickTime = 0; // Variable to store button click time
const unsigned long doubleClickInterval = 300; // Maximum interval between clicks for a double click

const int redLedPin = 11; // Red LED pin
const int greenLedPin = 12; // Green LED pin

struct DataPacket {
  float joystickX_1;
  float joystickY_1;
  float joystickY_2;
  float horizontalDistance;
  float verticalDistance;
};

DataPacket data;
float finalDistance = 0.0;

void setup() {
  Serial.begin(9600);

  servo1.attach(7); // Servo1 pin
  servo2.attach(8); // Servo2 pin
  servo3.attach(9); // Servo3 pin

  pinMode(trigPinHorizontal, OUTPUT);
  pinMode(echoPinHorizontal, INPUT);
  pinMode(trigPinVertical, OUTPUT);
  pinMode(echoPinVertical, INPUT);

  pinMode(buttonPin, INPUT_PULLUP); // Set button pin as input with internal pull-up resistor

  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);

  radio.begin();
  radio.openReadingPipe(1, address);
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_LOW);
  radio.startListening();
}

void loop() {
  if (radio.available()) {
    radio.read(&data, sizeof(data));

    // Check if button is clicked once or twice
    if (buttonClickedOnce) {
      data.horizontalDistance = measureDistance(trigPinHorizontal, echoPinHorizontal
      data.verticalDistance = 0.0; // Set vertical distance to zero

      digitalWrite(greenLedPin, HIGH); // Turn on green LED
      digitalWrite(redLedPin, LOW); // Turn off red LED
    } else if (buttonClickedTwice) {
      data.verticalDistance = measureDistance(trigPinVertical, echoPinVertical);
      data.horizontalDistance = 0.0; // Set horizontal distance to zero

      digitalWrite(greenLedPin, LOW); // Turn off green LED
      digitalWrite(redLedPin, HIGH); // Turn on red LED
    } else {
      data.horizontalDistance = 0.0; // Set horizontal distance to zero
      data.verticalDistance = 0.0; // Set vertical distance to zero

      digitalWrite(greenLedPin, LOW); // Turn off green LED
      digitalWrite(redLedPin, LOW); // Turn off red LED
    }

    radio.stopListening();
    radio.write(&data, sizeof(data));
    radio.startListening();

    moveServos(data.joystickX_1, data.joystickY_1, data.joystickY_2);

    // Update final distance based on button state
    if (buttonClickedOnce) {
      finalDistance = data.horizontalDistance;
    } else if (buttonClickedTwice) {
      finalDistance = data.verticalDistance;
    } else {
      finalDistance = 0.0; // No distance measured
    }

    // Display final distance
    Serial.print("Final Distance: ");
    Serial.println(finalDistance);

    Serial.print("Received Joystick X: ");
    Serial.println(data.joystickX_1);
    Serial.print("Received Joystick Y: ");
    Serial.println(data.joystickY_1);
    Serial.print("Received Joystick Y: ");
    Serial.println(data.joystickY_2);
    Serial.print("Horizontal Distance: ");
    Serial.println(data.horizontalDistance);
    Serial.print("Vertical Distance: ");
    Serial.println(data.verticalDistance);
  }
```

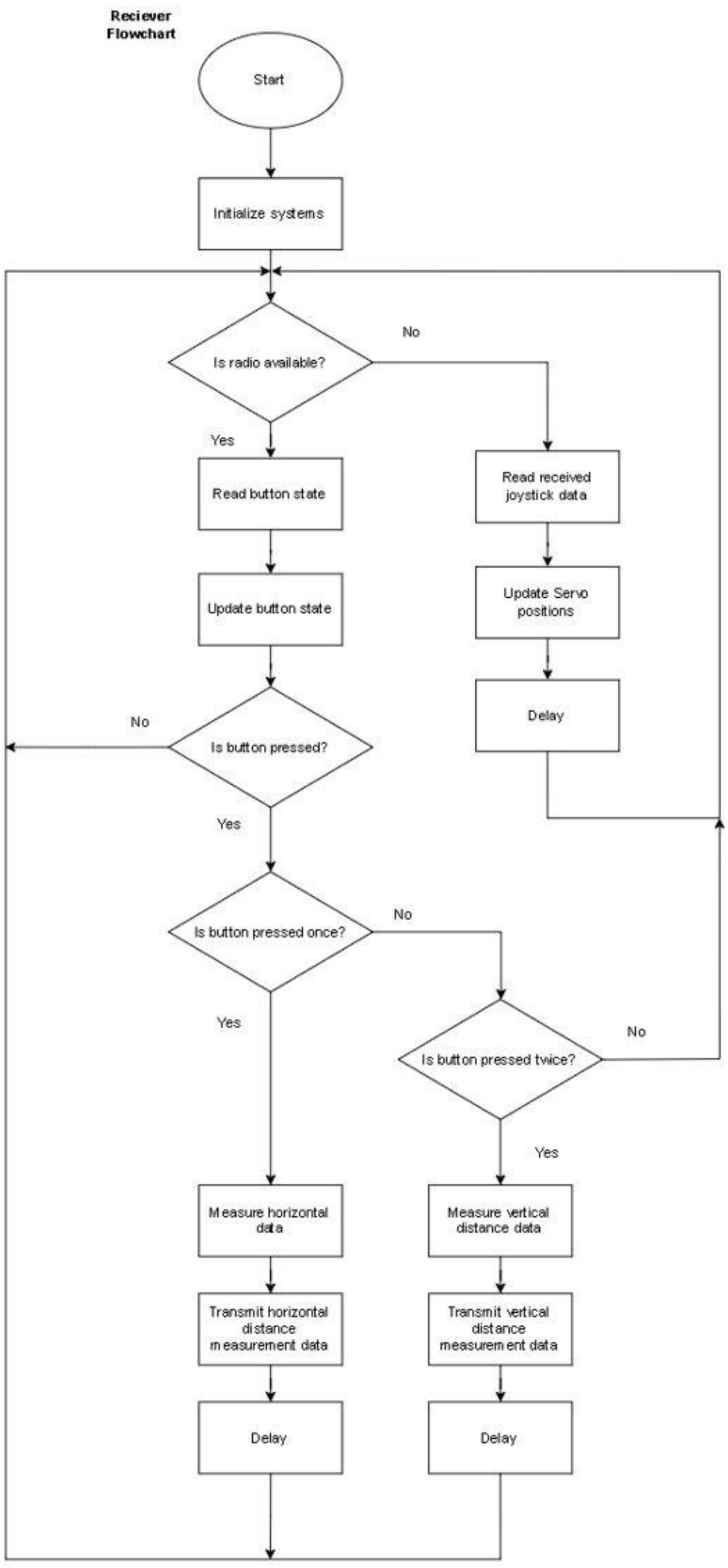
Annexure D: Control Centre code

```cpp
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

RF24 radio(9, 10); // CE, CSN pins
const byte address[6] = "00001"; // Address for communication

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address, number of columns, number of rows

struct DataPacket {
  float joystickX_1;
  float joystickY_1;
  float joystickY_2;
  float horizontalDistance;
  float verticalDistance;
};

DataPacket data;

const int transmitLedPin = 5; // Pin for the transmit LED
const int buttonPin = 2; // Pin for the button (replace with the actual pin number)

bool lastButtonState = HIGH; // Variable to store the last button state
bool currentButtonState; // Variable to store the current button state
bool armPositionChanged = false; // Flag to indicate if the arm position has changed

void setup() {
  Serial.begin(9600);

  pinMode(transmitLedPin, OUTPUT); // Set the transmit LED pin as an output

  lcd.begin(16, 2);
  lcd.backlight();
  lcd.clear();
  lcd.print("Horz:");
  lcd.setCursor(0, 1);
  lcd.print("Vert:");

  radio.begin();
  radio.openWritingPipe(address);
  radio.openReadingPipe(1, address);
  radio.setPALevel(RF24_PA_LOW);

  digitalWrite(transmitLedPin, LOW); // Initially turn off the transmit LED
  pinMode(buttonPin, INPUT); // Set the button pin as input
}

void loop() {
  data.joystickX_1 = analogRead(A0); // Read joystick X-axis value
  data.joystickY_1 = analogRead(A1); // Read joystick Y-axis value
  data.joystickY_2 = analogRead(A2); // Read joystick Y-axis value

  // Check the button state
  currentButtonState = digitalRead(buttonPin);

  // Compare the current button state with the last button state
  if (currentButtonState != lastButtonState) {
    // If the button state has changed, update the last position flag
    armPositionChanged = true;
  }

  lastButtonState = currentButtonState; // Update the last button state

  radio.stopListening();
  bool transmissionStatus = radio.write(&data, sizeof(data)); // Store the transmission status
  radio.startListening();

  if (transmissionStatus) {
    digitalWrite(transmitLedPin, HIGH); // Turn on the transmit LED if data is transmitted
  } else {
    digitalWrite(transmitLedPin, LOW); // Turn off the transmit LED if data is not transmitted
  }

  delay(500); // Delay for 500 milliseconds (adjust as needed)

  if (radio.available()) {
    radio.read(&data, sizeof(data));

    lcd.setCursor(5, 0);
    lcd.print("       cm");
    lcd.setCursor(5, 0);
    lcd.print(data.horizontalDistance);

    lcd.setCursor(5, 1);
    lcd.print("       cm");
    lcd.setCursor(5, 1);
    lcd.print(data.verticalDistance);

    Serial.print("Received Horizontal Distance: ");
    Serial.println(data.horizontalDistance);
    Serial.print("Received Vertical Distance: ");
    Serial.println(data.verticalDistance);
  }

  delay(1500); // Delay for 1500 milliseconds (adjust as needed)
```

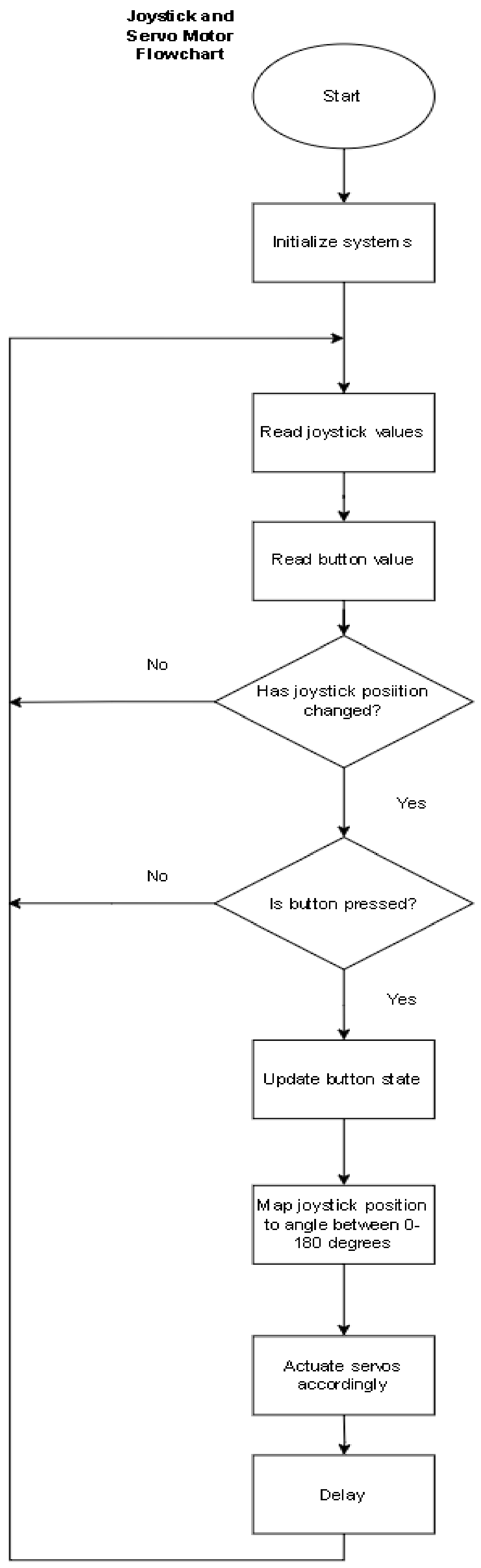
Annexure E: Control Centre flowchart



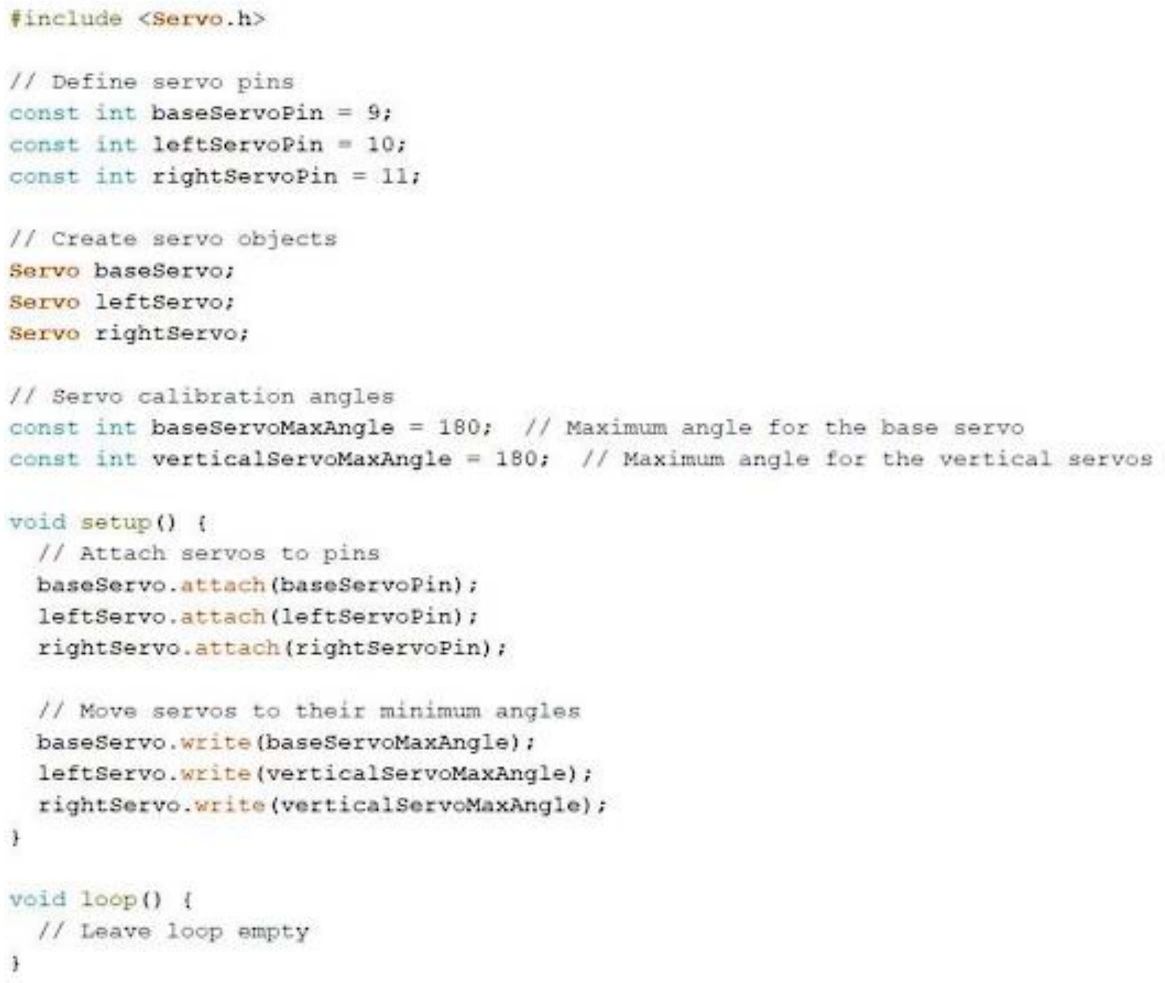
Annexure F: Robotic Arm flowchart



Annexure G: Joystick and servo motor subsystem flowchart

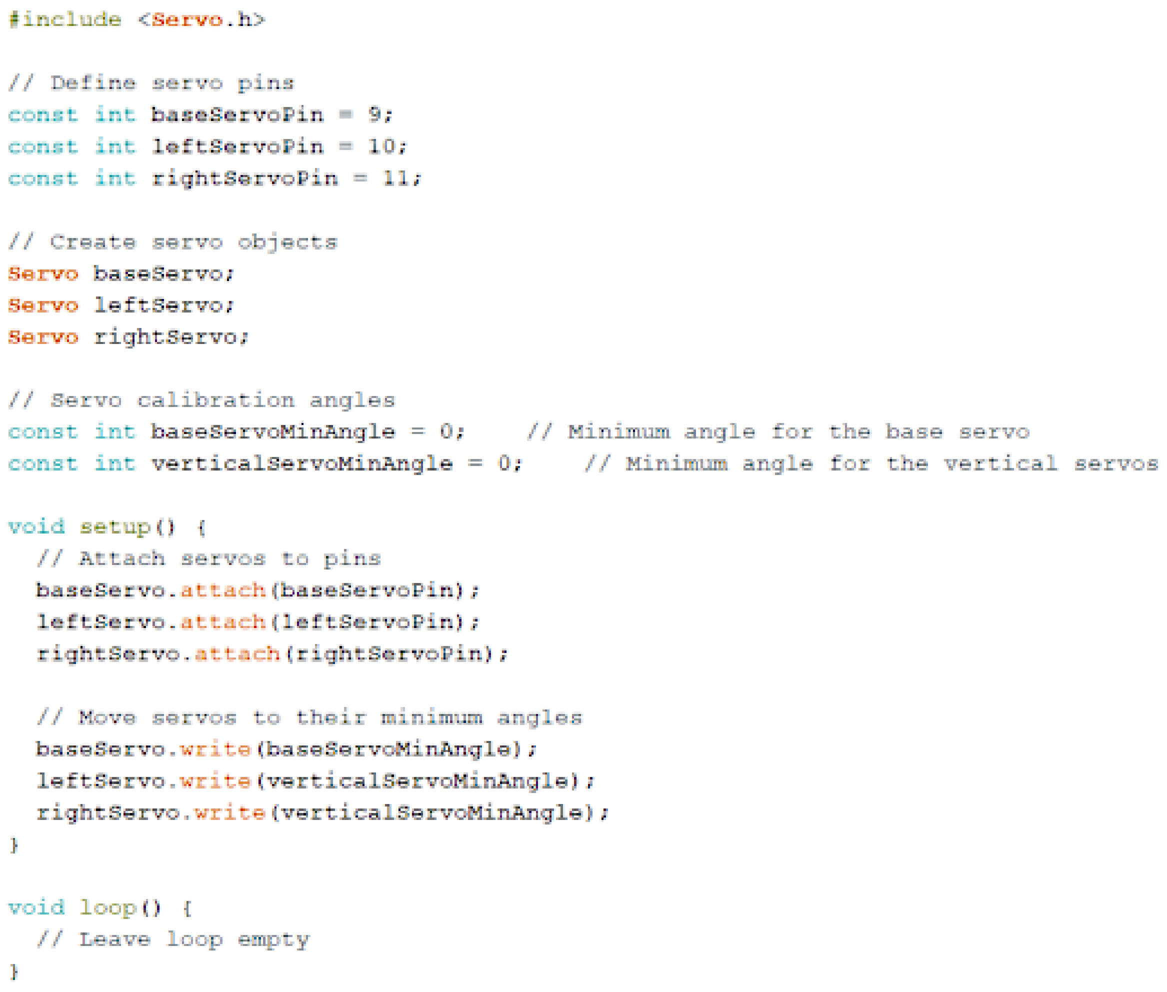

Joystick and Servo Motor Flowchart
Start
Initialize systems
Read joystick values
Read button value
No
Has joystick posiition changed?
Yes
No
Is button pressed?
Yes
Update button state
Map joystick position to angle between 0-180 degrees
Actuate servos accordingly
Delay

Annexure H: Maximum Angle Servo Motor Calibration



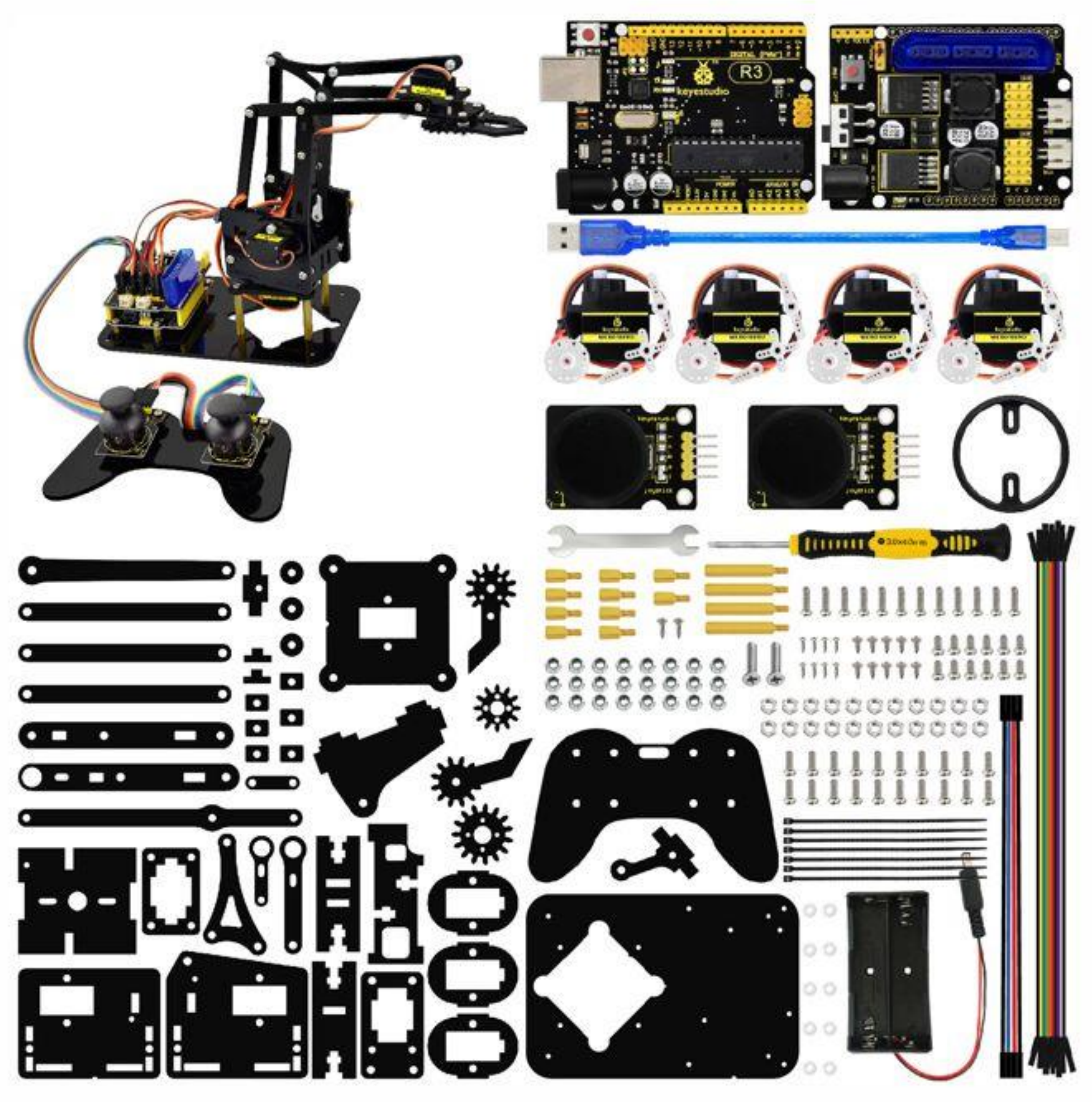
```
#include <Servo.h>

// Define servo pins
const int baseServoPin = 9;
const int leftServoPin = 10;
const int rightServoPin = 11;

// Create servo objects
Servo baseServo;
Servo leftServo;
Servo rightServo;

// Servo calibration angles
const int baseServoMaxAngle = 180;  // Maximum angle for the base servo
const int verticalServoMaxAngle = 180;  // Maximum angle for the vertical servos

void setup() {
  // Attach servos to pins
  baseServo.attach(baseServoPin);
  leftServo.attach(leftServoPin);
  rightServo.attach(rightServoPin);

  // Move servos to their minimum angles
  baseServo.write(baseServoMaxAngle);
  leftServo.write(verticalServoMaxAngle);
  rightServo.write(verticalServoMaxAngle);
}

void loop() {
  // Leave loop empty
}
```

Annexure I: Minimum Angle Servo Motor Calibration

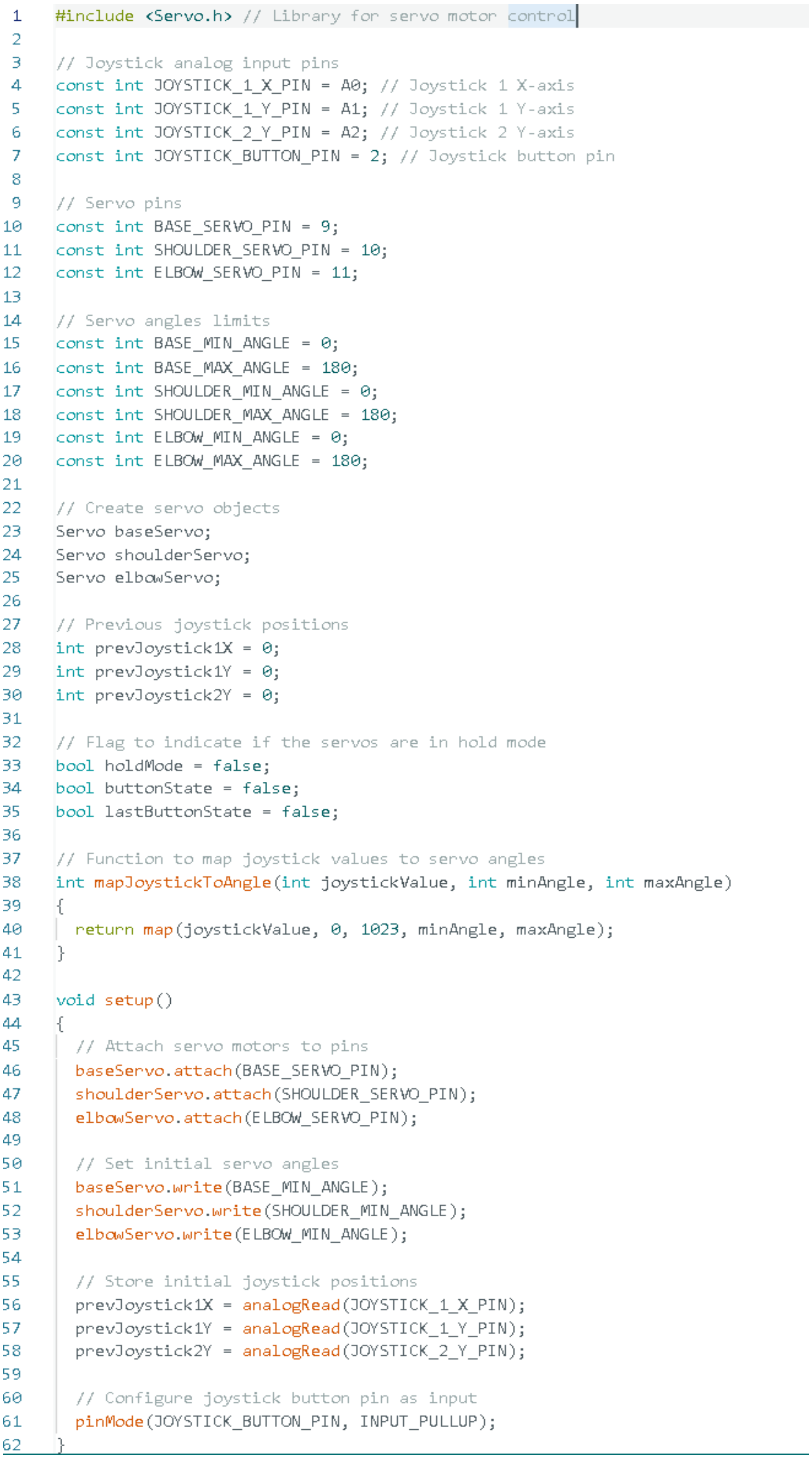


```
#include <Servo.h>

// Define servo pins
const int baseServoPin = 9;
const int leftServoPin = 10;
const int rightServoPin = 11;

// Create servo objects
Servo baseServo;
Servo leftServo;
Servo rightServo;

// Servo calibration angles
const int baseServoMinAngle = 0;     // Minimum angle for the base servo
const int verticalServoMinAngle = 0;     // Minimum angle for the vertical servos

void setup() {
  // Attach servos to pins
  baseServo.attach(baseServoPin);
  leftServo.attach(leftServoPin);
  rightServo.attach(rightServoPin);

  // Move servos to their minimum angles
  baseServo.write(baseServoMinAngle);
  leftServo.write(verticalServoMinAngle);
  rightServo.write(verticalServoMinAngle);
}

void loop() {
  // Leave loop empty
}
```

Annexure J: 4DOF Robotic Arm Kit



Annexure K: Subsystem Code



```
#include <Servo.h> // Library for servo motor control

// Joystick analog input pins
const int JOYSTICK_1_X_PIN = A0; // Joystick 1 X-axis
const int JOYSTICK_1_Y_PIN = A1; // Joystick 1 Y-axis
const int JOYSTICK_2_Y_PIN = A2; // Joystick 2 Y-axis
const int JOYSTICK_BUTTON_PIN = 2; // Joystick button pin

// Servo pins
const int BASE_SERVO_PIN = 9;
const int SHOULDER_SERVO_PIN = 10;
const int ELBOW_SERVO_PIN = 11;

// Servo angles limits
const int BASE_MIN_ANGLE = 0;
const int BASE_MAX_ANGLE = 180;
const int SHOULDER_MIN_ANGLE = 0;
const int SHOULDER_MAX_ANGLE = 180;
const int ELBOW_MIN_ANGLE = 0;
const int ELBOW_MAX_ANGLE = 180;

// Create servo objects
Servo baseServo;
Servo shoulderServo;
Servo elbowServo;

// Previous joystick positions
int prevJoystick1X = 0;
int prevJoystick1Y = 0;
int prevJoystick2Y = 0;

// Flag to indicate if the servos are in hold mode
bool holdMode = false;
bool buttonState = false;
bool lastButtonState = false;

// Function to map joystick values to servo angles
int mapJoystickToAngle(int joystickValue, int minAngle, int maxAngle)
{
  return map(joystickValue, 0, 1023, minAngle, maxAngle);
}

void setup()
{
  // Attach servo motors to pins
  baseServo.attach(BASE_SERVO_PIN);
  shoulderServo.attach(SHOULDER_SERVO_PIN);
  elbowServo.attach(ELBOW_SERVO_PIN);

  // Set initial servo angles
  baseServo.write(BASE_MIN_ANGLE);
  shoulderServo.write(SHOULDER_MIN_ANGLE);
  elbowServo.write(ELBOW_MIN_ANGLE);

  // Store initial joystick positions
  prevJoystick1X = analogRead(JOYSTICK_1_X_PIN);
  prevJoystick1Y = analogRead(JOYSTICK_1_Y_PIN);
  prevJoystick2Y = analogRead(JOYSTICK_2_Y_PIN);

  // Configure joystick button pin as input
  pinMode(JOYSTICK_BUTTON_PIN, INPUT_PULLUP);
}
```

```
void loop()
{
  // Read joystick values
  int joystick1X = analogRead(JOYSTICK_1_X_PIN);
  int joystick1Y = analogRead(JOYSTICK_1_Y_PIN);
  int joystick2Y = analogRead(JOYSTICK_2_Y_PIN);

  // Check if joystick positions are within the dead zone threshold
  const int deadZoneThreshold = 10; // Adjust this value if needed

  // Read the current state of the button
  buttonState = digitalRead(JOYSTICK_BUTTON_PIN);

  if (!holdMode) {
    // Servo control when not in hold mode
    if (abs(joystick1X - prevJoystick1X) > deadZoneThreshold) {
      int baseAngle = mapJoystickToAngle(joystick1X, BASE_MIN_ANGLE, BASE_MAX_ANGLE);
      baseServo.write(baseAngle);
      prevJoystick1X = joystick1X;
    }
    if (abs(joystick1Y - prevJoystick1Y) > deadZoneThreshold) {
      int shoulderAngle = mapJoystickToAngle(joystick1Y, SHOULDER_MIN_ANGLE, SHOULDER_MAX_ANGLE);
      shoulderServo.write(shoulderAngle);
      prevJoystick1Y = joystick1Y;
    }
    if (abs(joystick2Y - prevJoystick2Y) > deadZoneThreshold) {
      int elbowAngle = mapJoystickToAngle(joystick2Y, ELBOW_MIN_ANGLE, ELBOW_MAX_ANGLE);
      elbowServo.write(elbowAngle);
      prevJoystick2Y = joystick2Y;
    }

    // Check if the button state has changed
    if (buttonState != lastButtonState && buttonState == LOW) {
      holdMode = true;
    }
  } else {
    // Servo hold mode
    if (buttonState != lastButtonState && buttonState == LOW) {
      holdMode = false;
    }
  }

  lastButtonState = buttonState;

  delay(50); // Delay for smooth control (adjust if needed)
}
```