Ministry of Higher Education & Scientific Research
Maysan University

College of Engineering
Department of Electrical Engineering

A graduation project report submitted in accordance with the requirements of Maysan University for the degree of bachelor of Electrical Engineering

Robotic Arm Controlled by Hand Gestures

Students:

Kuwthar jassim
Nawras Majed
Narjes Heriz
Sajeda Hussan

Supervisor:

Sarah Kareem

1446 A.H 2025 A.C

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

{ وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ}

صَدَقَ اللهُ العَلِيُّ العَظِيمْ

الإهداء

نهدي هذا المشروع إلى اهالينا الاعزاء ، الذين كانوا ولا زالوا مصدر قوتنا بدعمهم الثابت، وصبرهم، ومحبتهم خلال مسيرتنا الدراسية.

كما نتوجّه بخالص امتناننا إلى مشرفنا، [ست ساره كريم]، على توجيهاتها القيمة، وخبرتها، وتشجيعها المستمر، والتي كان لها الدور الأكبر في إنجاح هذا العمل.

ونتقدم أيضًا بجزيل الشكر لجميع أعضاء الهيئة التدريسية في القسم على دعمهم المتواصل والمعرفة القيّمة التي نقلوها إلينا خلال سنوات دراستنا الجامعية.

كما نخص بالشكر الدكتور محمد خلف، رئيس قسم الهندسة الكهربائية، على تعاونه وتفانيه، وتوفيره البيئة الأكاديمية اللازمة.

ولا ننسى أصدقائنا وزملائنا الذين رافقونا في هذه الرحلة، فشكرًا على كل من له سبب على وصولنا لهذه المرحلة.

وأخيرًا، إلى كل من ساهم بكلمة، أو فكرة، أو يدٍ معطاءة — نشكركم من أعماق قلوبنا.

Abstract

This project aims to control a robotic arm using human hand gestures in real-time. A glove equipped with flex sensors and MPU6050 modules captures hand and finger movements, which are wirelessly transmitted via Bluetooth to an Arduino-controlled robotic arm. The system translates gestures into physical movements such as rotation, bending, and gripping. Designed for interactive and assistive use cases, the robotic arm responds within one second, providing smooth and accurate motion. This gesture-based approach improves usability and offers potential applications in education, rehabilitation, and industrial automation.

List of Contents

Index of Figures

Index of Tables

CHAPTER ONE

General Introduction and Project Framework

1.1 Introduction

The development of robot technology, also known as robotics, has made the workplace safer and more efficient. The use of robots in hazardous areas, such as those involving harmful chemicals or dangerous operations, reduces the likelihood of damage to goods or the occurrence of industrial accidents. Robots are also used in repetitive and productive tasks more than relying on human workers. In some applications, robots are capable of performing repetitive tasks faster, at a lower cost, and with higher accuracy than humans. The word “robot” is used to refer to the device that takes the place of a human in performing work, especially when it is difficult or impractical for humans to carry it out, and to reduce the potential risks in certain operations [1]. The current challenge in using computers does not lie in collecting or transmitting visual data, but in perceiving visual data in order to extract useful information from it [2]. Among these applications is the creation of a smart car capable of operating without a driver. One of the applications of robotics is in the field of navigation and surveillance, due to the high and fast accuracy it provides, as mobile and independent means [3]. Many autonomous and non-autonomous robots are available today in various areas, such as commercial robots, in research, and military applications. An autonomous robot is a mechanically and electronically complex system. Designing an autonomous robot requires knowledge of many engineering-related fields necessary before designing it technically, including mechanical and electronic control, and computer engineering. Mechanical engineers usually study the dynamics of the system. System modeling is a difficult task and can be infinitely complex, and designing a specific model in a simple and concise way largely depends on basic and sequential steps carried out by electrical and electronic engineers to design the system and its control interfaces. First, the mathematical model of the system is created, then control systems and techniques are designed as desired, taking into account the physical laws, and finally, the economic cost of the design. The economic costs include sensors and control devices, as obtaining these devices to perform a

specific task may be feasible in local markets. Therefore, designers make efforts to find alternatives, whether for the electronic components required for the design or the programming capabilities of the designer and the availability of sensor parts and microcontrollers [4]. Finally, self-controlled robots require knowledge from computer engineering and computer science, in addition to extensive experience to complete the software that will control the movement of the robot. The computer engineer must understand the system in which it operates and translate its movements into simple code instructions in order to obtain sensor data or other data required from the robot. That is, the robot can be programmed to perform specific and fixed movements, or to make it sensitive so that it processes the data it receives and then performs the required movements based on the commands it has received and processed [5].

1.2 Background and Evolution of Robotics

A robot is a mechanical tool capable of performing pre-programmed activities. The robot accomplishes these activities either by direct instruction and control from a human or by instruction from computer programs. The activities programmed for the robot to perform are usually exhausting or dangerous, such as mine detection, outer space exploration, and cleaning the waste produced in nuclear reactors [6]. Many studies and predictions have been made about this robot that later failed. However, after many good designs and serious attention to many details and precise matters, engineers succeeded in presenting various robotic systems for many industries expected in the near future. Today, due to the tremendous development of computers, artificial intelligence, technologies, and the obsession with developing space programs, we are on the verge of another major achievement in the field of robot design sciences. The robot is a reprogrammable manipulator that can perform multiple tasks and is dedicated to moving materials, parts, tools, or specific machines through various programmed movements to perform a number of tasks. There is an ongoing debate among scientists and linguists alike regarding the precise definition of the robot. Some argue that this term applies to every machine that can be controlled and moved remotely by humans, while others do not agree with this [7]. Their argument is that such machines, like a remote-controlled car or plane, cannot be considered robots because they do not have the ability to think and make decisions on their own. They

provide an example that if such a machine can act according to a pre-programmed set of instructions by stepping back two steps from an obstacle, turning right or left, and continuing forward, then it becomes possible to label it a true robot [8]. It becomes clear from this that the fundamental idea held by those with this opinion is that a true robot, according to some, must possess artificial intelligence and have the ability to recognize patterns, understand systems, reason, and make inferences. In the field of robotics, the first successful applications of the robot were in the American automotive industry. At the Ford Motor Company, specifically in the year 1940, a new word was born called "automation." After much time and effort, the robot began to perform many tasks in this field such as spot welding, machine loading, and many other applications. In 1995, around 25,000 robots were introduced into the automotive industry in America alone, and in the rest of the world, the number was not less than that, as 1,000,000 robots were used in various industrial fields [9]. The most important factor in the development of robotics technology, which helped the robot break into the industrial field relentlessly, was based on the discovery of microprocessors (zero-level processors), whose controllers were able to produce programs capable of executing coordinated movements for several degrees of freedom. The emergence of the industrial robot also came in multiple types prepared for manual tasks. Among these types is the robotic arm, which can be defined as a type of manipulator robot that is usually programmed to perform the same functions as the human arm, to a degree that may be equivalent to the human arm’s capability in terms of freedom of movement.

1.3 Laws of Robotics

The laws of robotics were established by Isaac Asimov and are as follows:

First Law: A robot may not cause harm to a human being or, through inaction, allow a human being to come to harm.

Second Law: A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.

Third Law: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law [10].

1.4 Basic Functions of the Robot

The robot has three main tasks or basic functions:

1. Sensing the environment around it through sensors.
2. Processing the received information through sensors.
3. Acting appropriately based on the previously made decision.

These basic functions are what make the robot an intelligent machine capable of operating without any human supervision or direct intervention in its work.

1.5 Defining the Project Problem

Recent advancements in human-machine interaction have highlighted the need for more natural and intuitive methods of communication with robotic systems. Among these, gesture-based control has emerged as a promising alternative to traditional interfaces that rely on keyboards, joysticks, or complex programming. Despite the progress, most robotic arms still depend on intricate control schemes, which can be impractical for individuals with limited mobility or technical background. This project addresses the challenge of creating a cost-effective, user-friendly system that translates real-time human hand gestures into robotic arm movements without requiring prior expertise or physical strain. By bridging the gap between human intention and robotic execution, the proposed system aims to make robotic manipulation more accessible and responsive.

1.6 Project Importance

The importance of the project lies in the significant role of robots in production lines and in carrying out various tasks in the modern era, due to their precision, speed, and ability to reach places that are difficult for humans to access. Robots have become an integral part of industrial production and daily life in developed countries. This project specifically stands out by offering a gesture-based control method, making interaction with the robotic arm more natural and efficient. It opens new possibilities for future applications in areas such as assistive technologies for people with disabilities, remote operations, and smart industrial processes.

1.7 Aims and Objectives

1. To design a multi-movement robotic arm controlled by hand gestures using flex sensors and motion sensors, aiming to accurately mimic human arm movements.
2. To assist humans in performing certain daily life tasks.
3. To reduce reliance on traditional control tools (such as buttons or mechanical joysticks) and replace them with a more intuitive and natural gesture-based interaction interface.

1.8 Chapters Layout

- **Chapter 1:** General Introduction and Project Framework
- **Chapter 2:** Literature Review
- **Chapter 3:** Design and Implementation Methodology for Gesture-Based Robotic Arm
- **Chapter 4:** Coding and Programming
- **Chapter 5:** Conclusion and Future Work

CHAPTER TWO

Literature Review

2.1 Related Works

This chapter reviews the literature (previous studies) on Arm Controlled by Hand Gestures and related topics concerning gesture-based robotic control systems through low-cost and user-friendly projects. It highlights the integration of these systems with various sensors and examines their practical applications. The fundamental concepts of these studies will be understood and analyzed, and the existing gaps will be identified and developed. The goal is to provide more efficient projects with high performance, simple systems, and low costs, and to further develop them in the future.

1. In 2016, The authors [**11**] proposed **MEMS Controlled Robotic Arm with Gestures**. This Paper addresses the design and implementation of a "MEMS Controlled Robotic Arm with Gesture". The system design was divided into three parts, which are: the accelerometer part, the robotic arm, and the platform. The system relies on a low-cost 3-axis (Degrees of Freedom) accelerometer, through which the robotic arm was controlled wirelessly using Zigbee signals. The robotic arm was mounted on a movable platform, and both were controlled wirelessly using the same accelerometer in a different switching mode. An accelerometer was fixed on the human hand to capture its behavior (gestures and postures), and as a result, the robotic arm moved accordingly, and in the other mode, the same accelerometer was used to capture the gestures and postures, so the platform moved accordingly. In summary, the robotic arm and platform were synchronized with the gestures and postures of the user's or operator’s hand, respectively. The movements executed by the robotic arm include: pick, drop, raise, and lower objects. As for the platform's movements, they are: forward, backward, right, and left. The system includes an IP-based camera that allows real-time video streaming wirelessly to any internet-connected device such as a mobile phone or laptop.

2. In 2017, The authors [**12**] proposed **Gesture Controlled Robotic Arm using Leap Motion.** This paper, Gesture control commence to be one of the simplest and easy way by with a complex robot could be controlled easily, with the help of different sensors and among them is

the leap motion sensor. It helps to build the inactive way of communication between the human hand and robotic arm with the help of few mathematical equations. This system can be utilized in the field of robotics where reduction of lot of effort and manual control is needed. The main object to study this system is to keep in mind the end goal to empower the embodiment of a mechanical system in to the home surroundings, to improve the self-governance and freedom of individuals with rigorous mobility disabilities and to concede at the same the tracking and aversion of eccentric disorders. The initial design used for the hand was very effective but due to the limitation used in the system, also limited the efficiencies of the system. The experimental results conclude that the system can detect hand gesture efficiently in real time and execute accordingly. As for future development it is considered that leap motion technology will definitely benefit and enable new ways to human-machine interaction in the field of ADLs and AAL due to its size and efficiency. This innovation enables more perceptive five DOF control with an end effector.

3. In 2018, The authors [**13**] proposed **Hand Gesture Controlled Robot Using Arduino.** This paper describes regarding how the conventional hand gestures can control a robot and perform our desired tasks. The transmitter will transmit the signal in line with the position of accelerometer and your hand gesture and therefore the receiver will receive the signal and make the robot move in respective direction. In this paper, an automated robot has been developed which works according to your hand gesture. The robot moves wirelessly according to palm gesture. The RF module is working on the frequency of 433 MHz and has a range of 50-80 meters. This robot can be upgraded to detect human life in earthquake and landslide by implementing the sensor accordingly. It can also be upgraded to bomb detecting robot by adding robotic arm which can also lift the bomb as well as in general terms, a robotic arm can be added which can be used in our day to day activities making human life easy.

4. In 2020, The authors [**14**] proposed **Hand Gesture Controlled Robot Arm.** The outcome of this work is to control a robotic arm using flex sensors pinned with a hand glove. The sensors are employed for remote control that will enable forward, backward, left and right control movements and pick and drop depends on the hand movements. The hardware setup was designed which results in the robotic arm formation. The software section enables

movements processing wherein the hand gestures were analyzed to extract the actual direction. The identified direction was transferred to the robotic arm through Zigbee. The command for the robotic arm is to direct specifically in the surroundings which is enabled by hand gestures technique adopted by the user. The use of external hardware support for gesture input not necessary like specified existing system. This working model enables user to control a robotic arm from his software station. The robotic arm delivers the programmed movement and the proposed model have widespread application for people working in hazardous areas.

5. In 2022, The authors [**15**] proposed **Gesture Controlled Robotics Hand**. This paper deals with the gesture recognition for controlling the movements of the robotic hand through wireless control using servo control, flex sensor, Arduino Nano, (receive rand transmitter) transceiver. Each finger managed to contract individually, however some fingers performed better than others. An example of a digit with worse performance was the thumb on the robotic hand. As for the hand signs the robotic hand was able to replicate the hand signs carried out by the operator with the glove The performed hand motions were also successfully imitated by the robotic hand, with little to no significant delays being observed.

6. In 2023, The authors [**16**] proposed **A Study on Hand Motion Controlled Robotic Arm.** This study has sought to address the challenges of intuitive control, precision, and usability in robotic arm technology, with the overarching goal of enhancing the potential applications across industries. The "A Study on Hand Motion Controlled Robotic Arm" project has made significant strides in advancing the field of human-robot interaction. The developed HMCR system represents a leap forward in intuitive and precise robotic control, with the potential to revolutionize industries and improve the quality of life for individuals with limited mobility. As we continue to refine and expand this technology, we move closer to a future where human-robot collaboration is seamless, efficient, and accessible to all.

7. In March 2024, The authors [**17**] proposed **Design and Development of Robotic Arm Control by Human Hand.** This research paper presents a comprehensive study on a robotic arm controlled by hand gestures and developed using 3D printing technology. The system's intuitive nature, coupled with its accuracy, versatility, and importance in industry, positions it as a transformative solution for optimizing human-robot collaboration and driving innovation in

industrial automation with an intuitive and user-friendly control interface, enhancing human-robot interaction in various applications such as manufacturing, healthcare, and education. The paper discusses the Design, implementation, experimental results, and potential applications of this innovative control System. Future work may focus on optimizing gesture recognition algorithms, expanding the range of supported gestures, and exploring advanced control strategies for more complex tasks.

8. In May 2024, The authors [**18**] proposed **Design and Development of Gesture Controlled Robotic Arm.** This paper introduces a groundbreaking approach to wirelessly control robotic hand movements via gesture recognition. The integration of flex sensors and the ESP32 module in a Gesture controlled robotic hand marks a notable advancement in human-robot interaction. This innovative system adeptly interprets hand gestures, enabling precise robotic movements with potential applications in prosthetics, industrial automation, and assistive technologies. The project's success highlights the feasibility and promise of leveraging accessible technologies to elevate user experience and propel robotics research forward.

9. In 2025, The authors [**19**] proposed **Arduino-Based Gesture-Controlled Robot.** The primary objective of this project—controlling a robot using hand gestures—was successfully achieved without any major obstacles. The robot accurately responds to hand movements, ensuring smooth operation. To enable remote control, a Holtek encoder-decoder pair (HT12E and HT12D) was implemented in conjunction with a 433MHz transmitter-receiver module. HT12E and HT12D are CMOS integrated circuits (ICs) that operate within a voltage range of 2.4V to 12V. The HT12E encoder consists of eight address lines and four additional address/data lines. When the transmit-enable (TE) pin is set to low, the encoded data from these lines is transmitted serially. The DOUT pin outputs the data in a repeated sequence, using positive-going pulses of varying durations to represent binary ‘1’ and ‘0’—with the pulse width for ‘0’ being twice that of ‘1’. The frequency of these pulses ranges from 1.5 kHz to 7 kHz, depending on the resistance value connected between the OSC1 and OSC2 pins.

2.2 Identified Research Gaps

Although gesture-controlled robotic systems have attracted increasing interest, many previous studies remain limited in key aspects such as real-world validation, user diversity, and integration with modern technologies. .This section highlights the main research gaps found in each reviewed study, aiming to guide future work toward more practical and advanced solutions.

1. In [2016] MEMS Controlled Robotic Arm with Gestures the gaps observed:

The system could not accommodate irregularities in hand size or gesture shape and this resulted in circumstances where the accuracy of the measurement was impacted due to environmental aspects or hand tremors of the respective users. The system did not accommodate automatically correcting for hand tremors and/or environmental factors such as lighting that caused measurement inaccuracies.

2. In [2017] Gesture Controlled Robotic Arm using Leap Motion the gaps observed:

Using the Leap Motion allowed for improved accuracy for the system; however, the Leap Motion capability did not provide adaptive calibration consideration for differing hand size or speeds of velocity, and this situation can negatively impact performance consistency for users.

3. In [2018] Hand Gesture Controlled Robot Using Arduino the gaps observed:

The system only accepted relatively simple directional gestures and posed difficulty in accommodating dynamic variations in gestures, or scale and could not accommodate complex robotic arm operations for users completing real world task.

4. In [2020] Hand Gesture Controlled Robot Arm the gaps observed:

The system did not provide examples of handling errors associated with detection error that can occur related to inaccuracies and delays associated with signals; this concern leads to decreased stability and responsiveness that is useful in time-critical applications.

5. In [2022] Gesture Controlled Robotics Hand the gaps observed:

The stability of the glove dimensions, along with the sensor placements had definitive fixed sizes, placements, motion tracking and gesture detection and these concerns limit flexibility and

comfort for these actions, moreover, long-term usability considerations are valid due to variations in users' hand shapes or required shapes and any needs of the users.

6. In [2023] A Study on Hand Motion Controlled Robotic Arm the gaps observed:

The potential for good accuracy existed for the system, however the system consistently lacked adaptive ability to account for and recognize different gesture patterns or hand shapes without needing retraining to recognize different gesture patterns and/or hand shapes; this concern influences the breathability of use of the system to a greater diversity of users.

7. In [2024, March] Design and Development of Robotic Arm Control by Human Hand the gaps observed:

The fixed placement of the potentiometers could also limit flexibility when the users' hand size differs or when range of motions vary since these concepts are vital to performance precision and comfort during extended periods of wear.

8. In [2024, May] Design and Development of Gesture Controlled Robotic Arm the gaps observed:

The system provided reliable gesture recognition performance, but by not providing any mechanism for adaptively correcting for sensor drift or exploration of user-specific calibration, could restrict long-term reliability and personalization.

9. In [2025] Arduino-Based Gesture-Controlled Robot the gaps observed:

The system offers useful gesture-based control, but can only filter noise, not adaptively to correct for sensor drift over long periods of use which could impact accuracy and reliability.

10. Common Gaps Summary:

There was generally no automatic calibration for differences in hand size and user movement. This was a glaring gap in most studies. While most systems didn't differ significantly in this area, other researchers didn't account for sensor noise and evolving drift over time. These gaps can be addressed to enhance the system’s credibility in this field, resulting in a more accurate, user-friendly, highly efficient, and cost-effective system.

2.3 Technical Comparison of Gesture-Based Robotic Arm Designs

This section provides a comparative overview of the technical designs used in the nine selected studies. It highlights the sensor types, system interfaces, performance feedback, communication methods, and key features to offer a clearer understanding of the evolution and diversity in gesture-controlled robotic arm technologies. As shown in **Table 2.1**, a technical comparison has been conducted across all reviewed studies.

Table 2.1 Technical Comparison of Gesture-Based Robotic Arm Designs

Title	Sensor Type	Display	Performance	Connectivity	Features
MEMS Controlled Robotic Arm with Gestures [11].	3-Axis Accelerometer	Wireless Camera/Not Specified	Pick, place, direction	Zigbee	Dual accelerometer sync for arm + base
Gesture Controlled Robotic Arm using Leap Motion [12].	Leap Motion	Software Visualization	**5** DOF with end effector	USB	Inverse kinematics, real-time tracking
Hand Gesture Controlled Robot Using Arduino [13].	Accelerometer	Not Specified	Basic movement	RF	Simple directional motion
Hand Gesture Controlled Robot Arm [14].	Flex Sensor	Mobile App / LEDs	Pick and place	Bluetooth	RF + app dual input
Gesture Controlled Robotics Hand [15].	Flex + Transceiver	Not Specified	Wireless glove control	RF	Custom glove with servos
A Study on Hand Motion Controlled Robotic Arm [16].	Vision + Sensor Fusion	Graphical Interface	High accuracy	Local	Motion tracking + gesture mapping
Design and Development of Robotic Arm Control by Human Hand [17].	Potentiometers	Not Specified	Smooth mimic motion	Arduino USB	3D printed arm, simple interface
Design and Development of Gesture Controlled Robotic Arm [18].	Flex Sensors	Not Specified	Functional control	433 MHz RF	Wireless pick-and-place
Arduino-Based Gesture-Controlled Robot [19].	MPU6050 (Accelerometer + Gyro)	None / LEDs	Directional control	RF	Accelerometer-based mobile navigation

2.4 Evaluation Outcomes of Previous Gesture-Controlled Robotic Prototypes

This section summarizes the experimental results and evaluation outcomes for each of the reviewed prototypes. It covers key aspects such as accuracy, validation methods, and practical performance, providing insights into system reliability and effectiveness. The summarized evaluation results are presented in **Table 2.2**.

Table 2.2 Evaluation Results of Reviewed Gesture-Controlled Robotic Systems

Study	Accuracy /Error	Validation Method	Remarks
MEMS Controlled Robotic Arm with Gestures [11].	Good real-time performance	Accelerometer sync test	Dual part control tested
Gesture Controlled Robotic Arm using Leap Motion [12].	Highly responsive	Leap 3D coordinate mapping	5 DOF, fine control
Hand Gesture Controlled Robot Using Arduino [13].	Moderate	Movement-based response	Directional only
Hand Gesture Controlled Robot Arm [14].	Effective	App command test	Basic grasping task
Gesture Controlled Robotics Hand [15].	Decent	Wireless motion trials	Servo precision measured
A Study on Hand Motion Controlled Robotic Arm [16].	High	Vision-mapping validation	Reliable tracking
Design and Development of Robotic Arm Control by Human Hand [17].	Accurate	Angle-to-PWM mapping	Smooth servo motion
Design and Development of Gesture Controlled Robotic Arm	Functional	RF gesture demo	Controlled motion achieved
Arduino-Based Gesture-Controlled Robot [19].	Basic	Gesture-to-motor response	Simple four-directional control

CHAPTER THREE

Design and Implementation Methodology for Gesture-Based Robotic Arm

3.1 System Architecture and Overview

This chapter presents the hardware and software design methodology of the proposed system. It also includes experimental results. The idea behind this project started from a simple yet practical goal “controlling a robotic arm using human hand gestures in real time”. To make this possible, a sensor glove was used to capture finger bending and hand orientation. These values are then translated into movement commands that are sent to a robotic arm, making it perform corresponding motions. The entire system is divided into three main parts: the mechanical part, which includes the 3D-printed arm; the electrical part, which includes the sensors, Arduino boards, motors, and drivers; and the software part, which controls the logic and communication between the glove and the arm. The mechanical structure of the arm was built using a 3D printer, allowing for fast prototyping and customization. The electrical components include three flex sensors and an MPU6050 sensor fixed on a glove, connected to an Arduino Nano. The captured data is sent via Bluetooth using the HC-05 module to another Arduino Uno on the arm side, which controls six servo motors using the PCA9685 driver. These motors move the joints of the robotic arm according to the received commands. The software was programmed using Arduino IDE. It reads the sensors’ input, processes them, and maps the values to motor positions. This helps simulate the same hand gestures in the robotic arm, providing a responsive and intuitive experience. The system is powered using a 7.4V Li-ion battery, providing stable energy to both the control boards and servo motors. All the mechanical parts of the arm were printed using a Creality Ender 3 printer, which helped reduce cost and allowed for a high level of mechanical precision.

3.2 Hardware Components and Functions

This section outlines the key hardware components used in the system, including sensors, microcontrollers, drivers, and actuators. Each component plays a specific role in capturing hand gestures, transmitting data, and executing precise robotic arm movements.

3.2.1 Input Subsystem

This subsystem is responsible for sensing the hand gestures and transmitting the interpreted data to the robotic arm.

3.2.1.1 Flex Sensors

The versatility of flex sensors makes them valuable for creating interactive and responsive systems that can adapt to physical changes in their environment. When the sensor is bent, the resistance of the conductive material changes, and this change in resistance. Flex sensors are commonly used in various applications, including robotics, wearable devices, and medical equipment [8].

Specification:

- Operating voltage of this sensor ranges from 0V to 5V
- It can function on low-voltages.Length 2.2 inch.
- Power rating is 1 Watt for peak & 0.5Watt for continuous.
- Operating temperature ranges from -45ºC to +80ºC
- Flat resistance is 25K Ω
- The tolerance of resistance will be ±30%
- The range of bend resistance will range from45K -125K Ohms.

Figure (3.1) Flex Sensor

3.2.1.2 MPU6050 Accelerometers

The MPU6050 Accelerometer and Gyroscope is a 6-axis motion tracking sensor that integrates a 3-axis accelerometer and a 3-axis gyroscope into a single compact module. . It provides orientation data (pitch, roll, yaw) which helps in detecting hand tilt and rotation. This module plays a crucial role in identifying directional movements of the hand. The fusion of gyroscope and accelerometer data enables more stable and accurate gesture recognition.It is widely used in motion-based applications, robotics due to its high precision and ease of integration with microcontrollers like Arduino. [20].

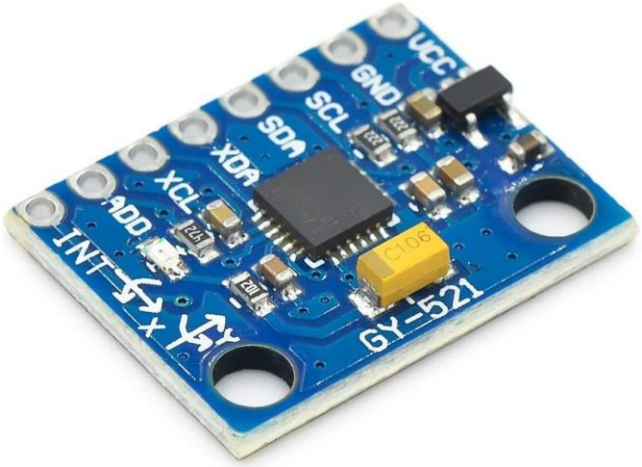


Figure (3.2) MPU6050 Accelerometers

3.2.1.3 Arduino Nano

The Arduino Nano is equipped with 30 male I/O headers, in a DIP30-like configuration, which can be programmed using the Arduino Software integrated development environment (IDE), which is common to all Arduino boards and running both online and offline. The board can be powered through a type-B mini-USB cable or from a 9 V battery.Micro-controller: Microchip ATmega328P [15].

Specification:

- Operating voltage: 5 volts
- Input voltage: 6 to 20 volts
- Digital I/O pins: 14 (6 optional PWM outputs)

- Analog input pins: 8
- DC per I/O pin: 40 mA
- DC for 3.3 V pin: 50 mA
- Flash memory: 32 KB, of which 0.5 KB is used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock speed: 16 MHz
- Length: 45 mm
- Width: 18 mm
- Mass: 7 g
- USB: Mini-USB Type-B [5]
- ICSP Header: Yes
- DC Power Jack: No [15].

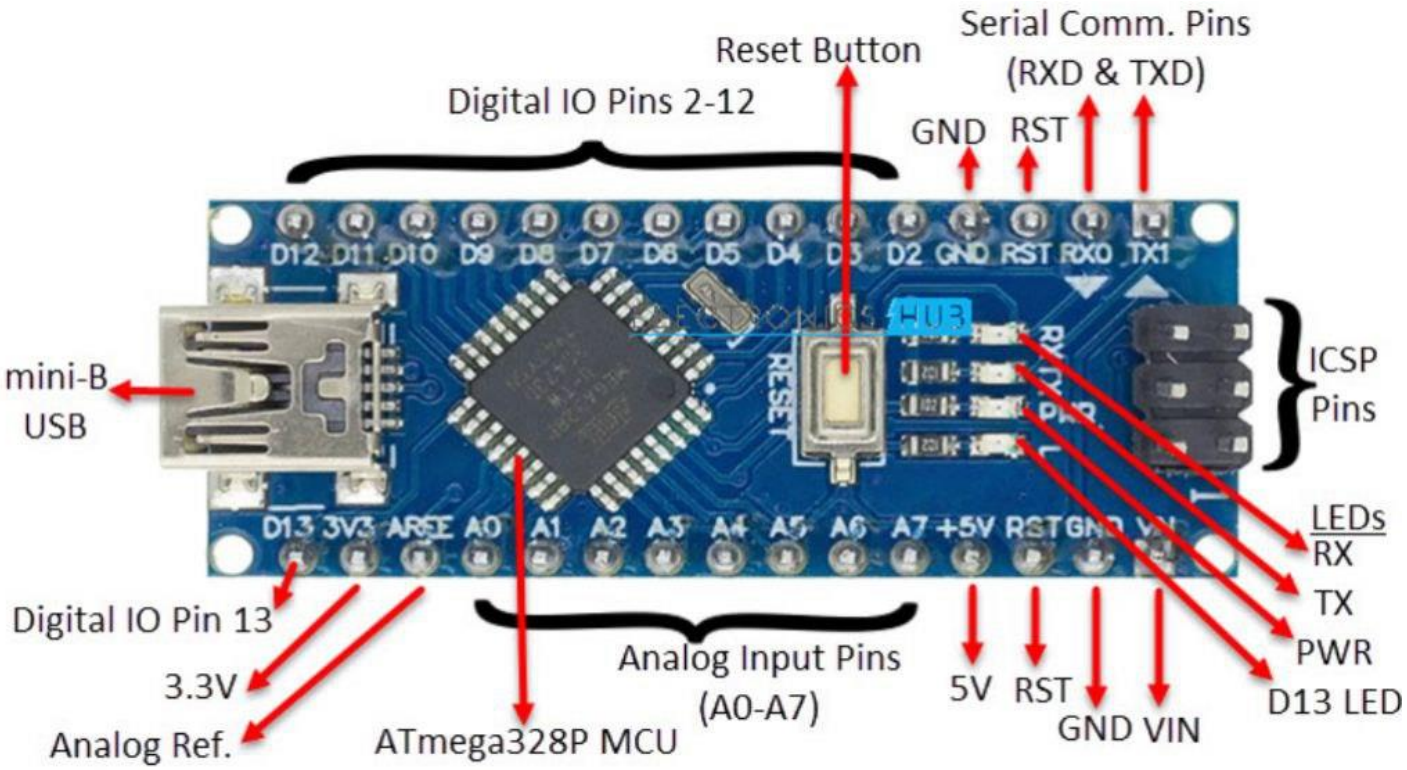


Figure (3.3) Parts of the Arduino Nano

3.2.1.4 Bluetooth Module HC05

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. It has the footprint as small as 12.7mmx27mm. It is connected to android mobile using its inbuilt Bluetooth and receives signal from the android app and delivers characters to microcontroller accordingly [21].

Features:

- Up to +4dBm RF transmit power
- Low Power 1.8V Operation,1.8 to 3.6V /O
- Default Baud rate: 38400, Data bits:8, Stop bit:1, Parity: No parity, Data control has supported baud rate:9600,19200,3
- 8400,57600, 115200,230400,460800.
- Auto-connect to the last device on power as default.

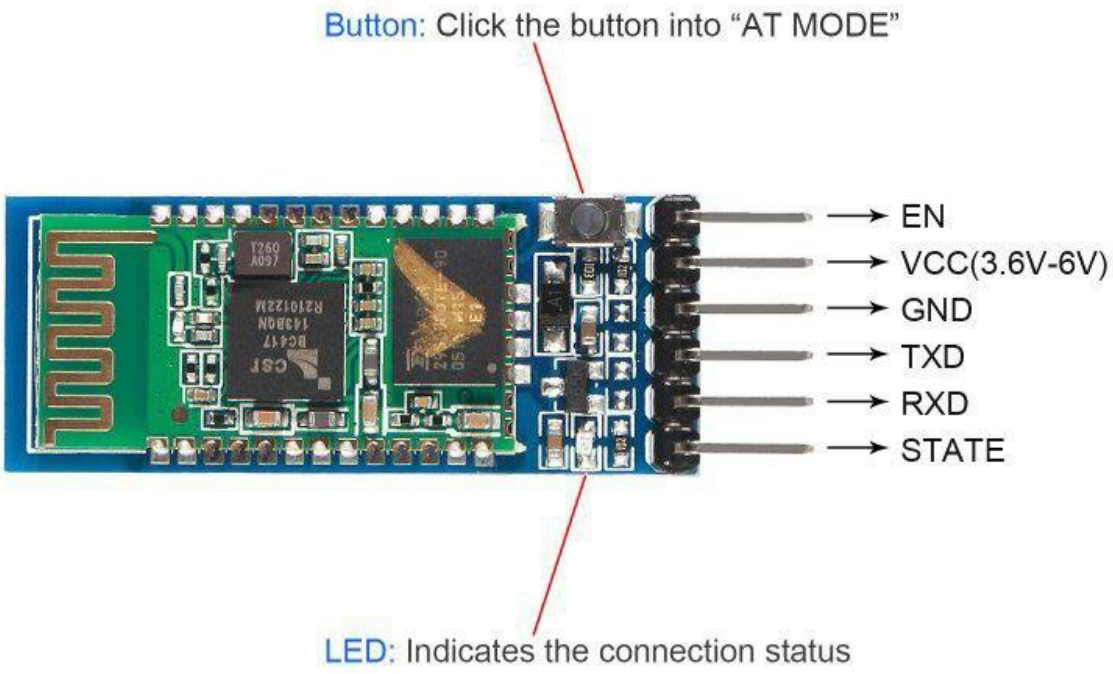


Figure (3.4) Bluetooth Module HC05

is considered part of both the input (glove) and output (arm) subsystems, as it facilitates wireless communication between them.

3.2.2 Output Subsystem (Robotic Arm)

This subsystem interprets the received commands and translates them into mechanical motion through actuators.

3.2.2.1 MG966R Servo Motors

Tower Pro MG996 Metal Gear Servo Motor The MG996R is a metal gear servo motor with a maximum stall torque of 11 kg/cm. Like other RC servos the motor rotates from 0 to 180 degree based on the duty cycle of the PWM wave supplied to its signal pin. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwidth and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec. This high-torque standard servo can rotate approximately 120 degrees (60 in each direction) [22].

Specifications [22]:

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kg·cm (4.8 V), 11 kg·cm (6 V)
- Operating speed: 0.17 s/60o (4.8 V), 0.14 s/60 o (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Running Current: 500 mA – 900 mA (6V)
- Stall Current: 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0-55 ℃

Figure (3.5) MG966R Servo Motors

3.2.2.2 NEMA 17 Stepper Motor

A stepper motor used with 3D printers, CNC machines, robotics, automation systems, precision instrumentation, and medical equipment.

Specifications :

- Operating Mode: Bipolar Mode
- Detent Torque: 1.6 to 2.6 cNm
- Holding Torque: 19.5 to 55 cNm
- Step Angle: 1.8 degrees (200 steps per revolution)
- Number of Leads: 4
- Rated Voltage: 12 to 24 V
- Rated Current: 0.33 to 1.30 Amps
- Compatible with external and internal micro-stepping drivers for enhanced resolution and smoother operation.

Features :

1. High Precision: The 1.8-degree step angle ensures precise control with 200 steps per revolution.

2. Versatile Voltage Range: Operates effectively between 12 to 24 V, accommodating various power supply options.

3. Adjustable Current: Rated current range of 0.33 to 1.30 Amps allows for flexibility in torque and speed settings.

4. Strong Holding Torque: With a holding torque ranging from 19.5 to 55 cNm, it is suitable for applications requiring strong positional stability.

5. Micro-Stepping Support: Compatible with both internal and external micro-stepping drivers to achieve finer resolution and smoother motion [23].

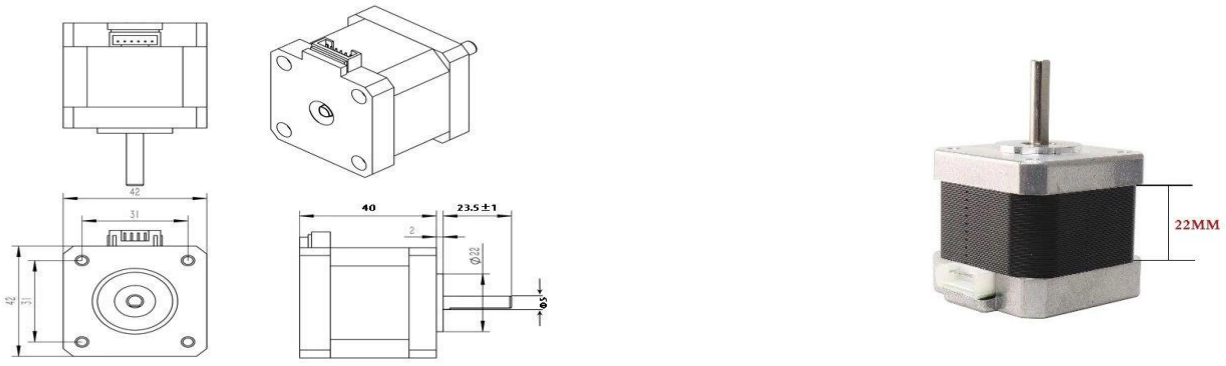


Figure (3.6) Internal and External View of the NEMA 17 Stepper Motor [23].

3.2.2.3 A4988 Stepper Motor Driver

Drives the NEMA 17 stepper motor by controlling the current and providing micro-stepping features. It enhances motion smoothness and efficiency in stepper motor control [24].

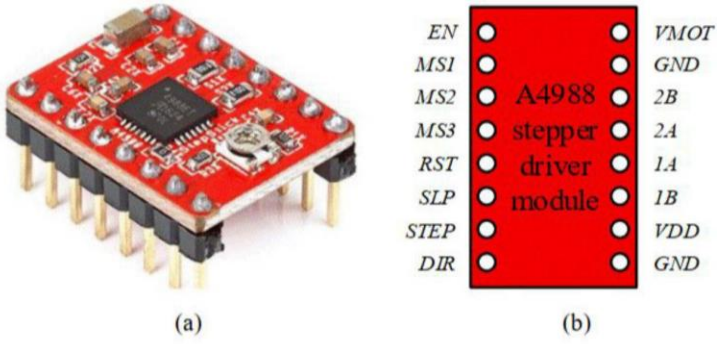


Figure (3.7) (a) A4988 Stepper motor driver module, and (b) its pinout diagram.

3.2.2.4 PCA9685 Servo Driver

The PCA9685 is a 16-channel I2C-bus controlled LED controller optimized for Red/Green/Blue/Amber (RGBA) color backlighting applications. Each LED output has individual 12-bit resolution (4096 steps) PWM controller with a fixed frequency. The controller operates at a programmable frequency from a typical 24 Hz to 1526 Hz with a duty cycle that is adjustable from 0% to 100% so the LED can be set to output a specific brightness. All outputs are set to the same PWM frequency. With the PCA9685 as the master chip, the 16-channel 12-bit PWM Servo Driver only needs 2 pins to control 16 servos, thus greatly reducing the occupant I/Os. Moreover, it can be connected to 62 driver boards at most in a cascade way, which means it will be able to control 992 servos in total.

Technical Specification:

- **Driver IC:** PCA9685
- **Size:** 62mm*25mm*15mm (L*W*H)

Features:

Contains an I2C-controlled PWM driver with a built-in clock. It means, unlike the TLC5940 family. 5V compliant, which means you can control it from a 3.3V microcontroller, which is good when you want to control white or blue LEDs with a 3.4V+ forward voltage Supports using only two pins to control 16 free-running PWM outputs – can even chain up 62 breakouts to control up to 992 PWM outputs. 3 pin connectors in 4 groups, so plug in 16 servos at one time (Servo plugs are slightly wider than 0.1" so you can only stack 4 adjacent ones on 0.1"-hole female headers 12-bit resolution for each output - for servos, that means about 4us resolution at an update rate of 60Hz

Applications:

- RGB or RGBA LED drivers
- LED status information
- LED displays
- LCD backlights
- Keypad backlights for cellular phones or handheld devices

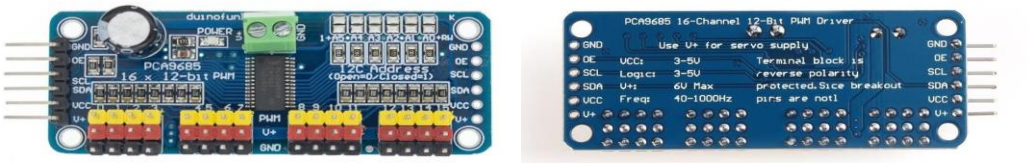


Figure (3.8) PCA9685 Servo Driver

3.2.2.5 Arduino Uno

The central controller of the robotic arm. It receives Bluetooth data from the glove via the HC-05 module and generates the necessary PWM signals to control the servos and stepper motor. It is chosen for its compatibility with a wide range of motor control libraries and its ease of programming.

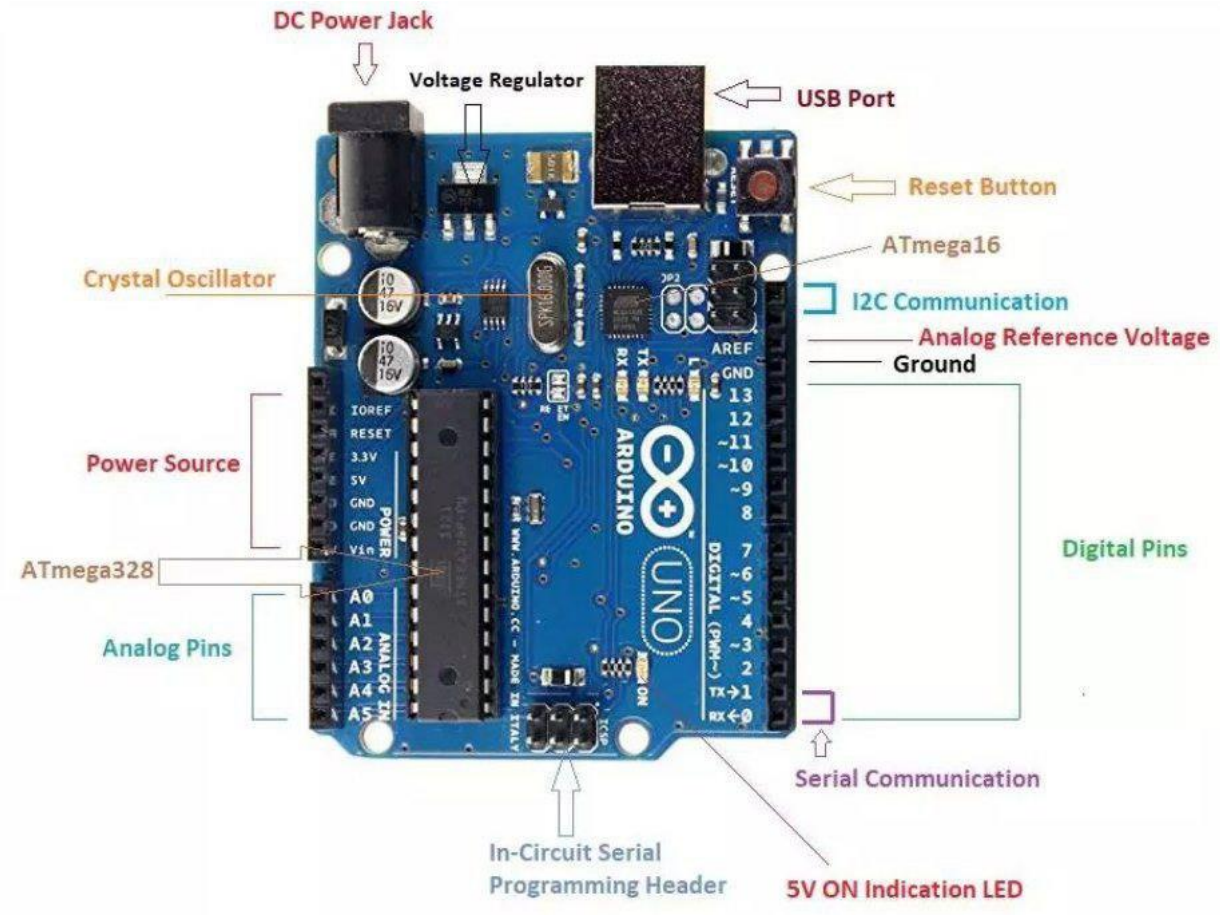


Figure (3.9) Components and Parts of the Arduino Uno

3.2.3 Components of the Project (with Quantity)

This section presents a detailed breakdown of all hardware components used in the development of the hand-gesture controlled robotic arm. Each item is listed with its exact quantity to provide a clear overview of the system's physical structure and assembly needs.

Table 3.1 List of Project Components with Quantities

No.	Component	Quantity	Description/Note
1	MG966R Servo Motor	6	For joint movements of the robotic arm
2	Servo Driver PCA9685	1	Controls all servo motors simultaneously
3	Arduino Uno	1	Main controller for the robotic arm
4	Arduino Nano	1	Microcontroller for the glove unit
5	Flex Sensor	3	Captures finger bending gestures
6	MPU6050 Accelerometer/Gyro	2	Measures hand orientation
7	Bluetooth Module HC-05	1	Wireless communication between glove and arm
8	NEMA 17 Stepper Motor	1	Linear movement for base/extension
9	A4988 Stepper Motor Driver	1	Drives the NEMA 17 motor
10	Battery (5V, 2200mAh)	1	Power source for the system
11	Breadboard	1	For circuit prototyping
12	Jumper Wires	Multiple	Connections between components
13	Resistors (10kΩ + 220Ω)	Several	Used with sensors and LEDs
14	Capacitors (100nF)	3	Noise reduction and signal stability
15	Builders Glove	1	Platform for sensor attachment
16	9V Battery + Clip	1	Powers the glove module
17	Braided Cable Sleeve	1	Organizes and protects wires

3.3 Circuit Wiring and Integration

The complete wiring layout of the project, titled Robotic Arm Controlled by Hand Gestures, was diagrammed to clearly illustrate the electrical connections among all major components, including the Arduino Uno & Arduino Nano, PCA9685 servo driver, A4988 stepper motor driver, HC-05 Bluetooth module, Li-Po battery pack, and actuators, as depicted in the following figure.:

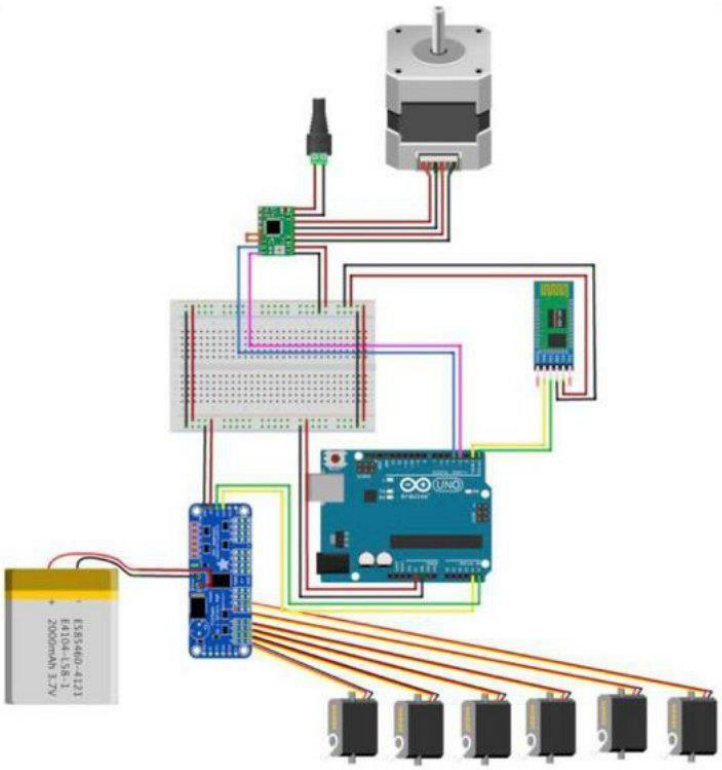
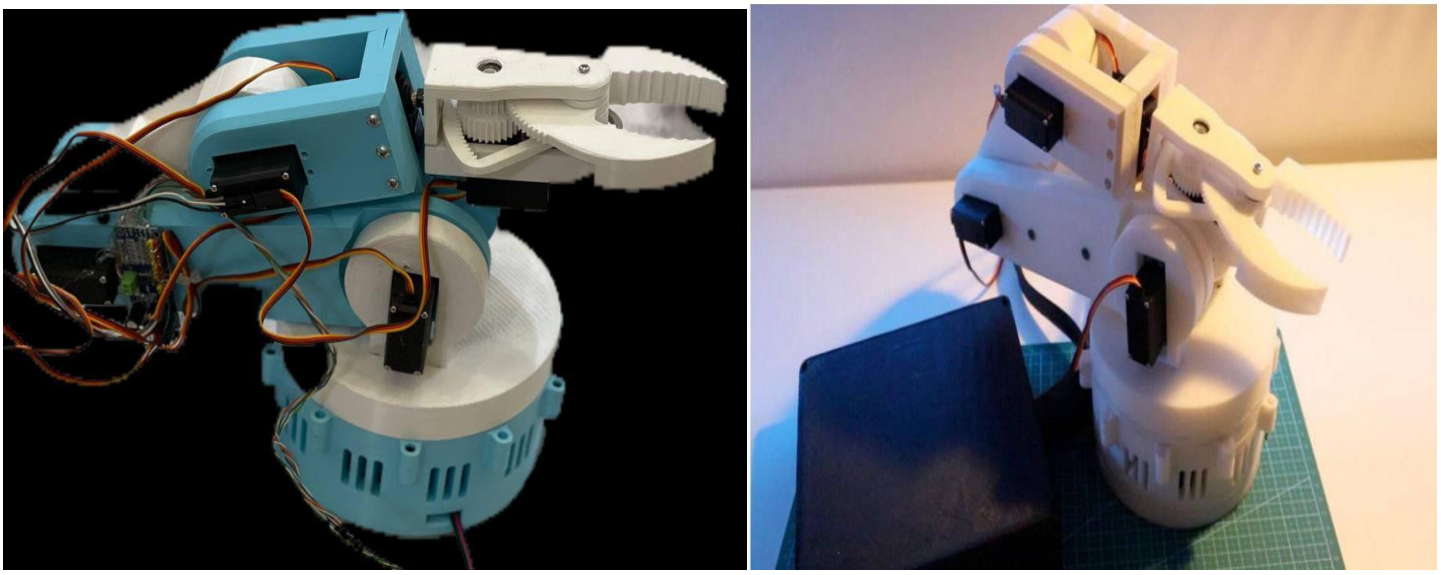


Figure (3.10) Circuit wiring diagram for the Robotic Arm Controlled by Hand Gestures project

Figure(3.10) Show the interconnection of input and output subsystems through the Arduino Uno and associated modules. Where the circuit wiring of the robotic arm system was carefully designed to ensure synchronized movement based on hand gestures and reliable communication between the glove and the robotic arm. The integration process began by placing all electronic components on a breadboard for testing and alignment. The Flex sensors, embedded in a glove, were connected to the analog input pins of the Arduino Nano to capture finger bending. Each sensor's VCC and GND pins were connected to the 3.3V and ground rails respectively, while

their output signal pins were routed to analog pins A0 to A4, enabling the Arduino Nano to read varying resistance values caused by finger movement. To facilitate wireless communication, the HC-05 Bluetooth module was integrated and connected to the TX and RX pins of the Arduino Nano, allowing data transmission from the glove to the robotic arm controller. On the receiving end, an Arduino Uno was configured to control the motors. A PCA9685 servo driver was employed to drive six MG966R servo motors, providing sufficient PWM signals with stable timing. The SCL and SDA pins of the PCA9685 module were connected to A5 and A4 respectively, which are the default I2C pins on the Arduino Uno. The driver's power and ground were connected to a 5V battery pack and a common GND to ensure consistent motor operation. Additionally, a NEMA 17 stepper motor was driven via an A4988 stepper driver, with DIR and STEP pins connected to digital I/O pins on the Arduino Uno. Power was supplied to the system through either USB during testing or a 9V battery connected to the Vin and GND pins for standalone operation. Great care was taken to shorten wire lengths, ensure tight connections, and prevent noise or instability. This organized circuit structure enabled accurate gesture recognition, real-time actuation of the robotic arm, and stable wireless communication, making the system both portable and efficient.



Figure(3.11) Final shape of the robot arm after connection

3.4 Software Architecture and Programming Logic

1. PCA9685 Servo Driver Library and Variable Definitions

```
#include "HCPCA9685.h"
#define  I2CAdd 0x40
HCPCA9685 HCPCA9685(I2CAdd);

const int servo_joint_L_parking_pos = 60;
int servo_joint_L_pos_increment = 20;
int servo_joint_L_parking_pos_i =
servo_joint_L_parking_pos;
int servo_joint_L_min_pos = 10;
int servo_joint_L_max_pos = 180;
```

Note: These initial values define the starting position, motion sensitivity, and angular limits for each servo.

2. Initializing Serial Communication for Bluetooth

```
Serial.begin(4800); // Initialise default communication rate of the Bluetooth module
```

Note: The 4800 baud rate is suitable for stable communication with the HC-05 module.

3. Receiving Bluetooth Data and Storing It

```
if (Serial.available() > 0) {
  state = Serial.read();
  Serial.print(state);
}
```

Note: The received value is used to determine the motor behavior.

4. Executing Motion Based on Received Character

```
if (state == 'S') {
  baseRotateLeft();
  delay(response_time);
}
```

Note: A short delay is added to prevent repeated triggering of the same command.

5. Rotating the Base Using a Stepper Motor

```
void baseRotateLeft() {
  digitalWrite(stepPin, LOW);
  delayMicroseconds(stepDelay);
}
```

6. Moving a Servo Motor Based on Gesture

```
if (state == 'f') {
  if (servo_joint_3_parking_pos_i < servo_joint_3_max_pos) {
    HCPCA9685.Servo(4, servo_joint_3_parking_pos_i);
    delay(response_time);
    Serial.println(servo_joint_3_parking_pos_i);
    servo_joint_3_parking_pos_i += servo_joint_3_pos_increment;
  }
}
```

Note: A hard stop prevents the motor from exceeding its maximum rotation (typically 180°), which could damage the servo gears.

7. Defining Flex Sensors and MPU6050 Addresses

```
int pinkie_Data = A1;
int finger_Data = A2;
int thumb_Data = A3;

const int MPU2 = 0x69, MPU1 = 0x68;

int response_time = 1000;
```

8. Reading Values from the Flex Sensors

```
pinkie = analogRead(pinkie_Data);
finger = analogRead(finger_Data);
thumb = analogRead(thumb_Data);
```

9. Reading Orientation Data from MPU6050

```
GetMpuValue1(MPU1);
GetMpuValue2(MPU2);
```

10. Sensor Calibration Routine on Reset

```
if (bool_caliberate == false ) {
  delay(1000);
  thumb_high = (thumb * 1.15);
  thumb_low = (thumb * 0.9);
  finger_high = (finger * 1.03);
  finger_low = (finger * 0.8);
  pinkie_high = (pinkie * 1.06);
  pinkie_low = (pinkie * 0.8);
  bool_caliberate = true;
}
```

Note: The calibration is triggered automatically upon pressing the reset button on the Arduino Nano.

11. Sending Gesture Command Based on Sensor Thresholds

```
if (finger >= finger_high) {
  Serial.print("F");
  delay(response_time);
}
if (finger <= finger_low) {
  Serial.print("f");
  delay(response_time);
}
```

Note: Each character corresponds to a specific servo action, such as opening or closing the claw.

3.5 Control Unit

The control system is structured around two Arduino boards: one acts as a peripheral unit and the other as a central controller. This design improves modularity, separates signal interpretation from actuation, and enhances overall performance.

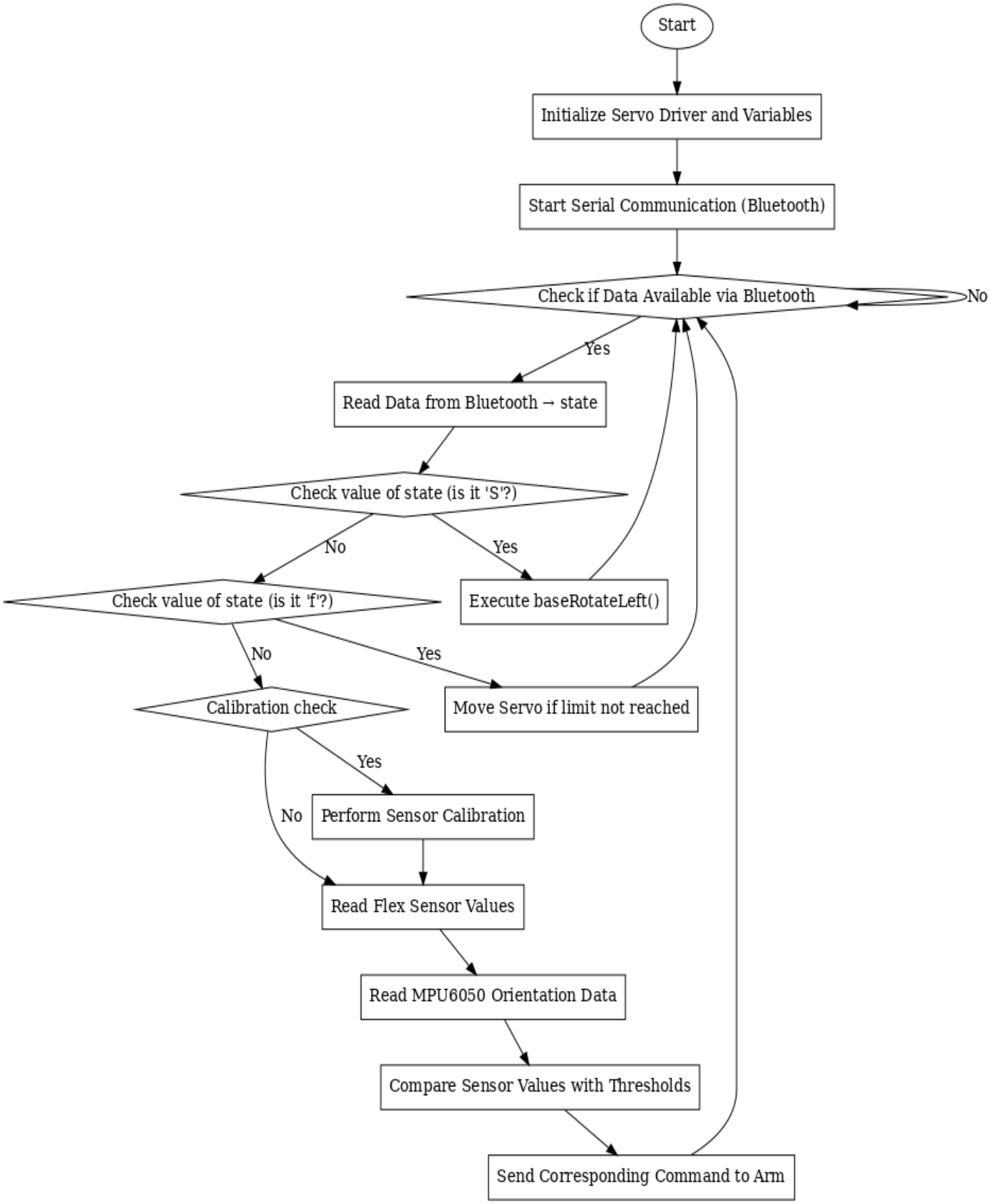
1. Peripheral Control Unit – Arduino Nano

Mounted on the glove, the Arduino Nano reads values from flex sensors and MPU6050 IMUs. It processes these inputs to detect gestures and sends symbolic commands via Bluetooth to the Arduino Uno. This reduces the computational load on the central controller.

2. Central Control Unit – Arduino Uno

The Arduino Uno receives gesture codes and translates them into motor actions. It controls servo motors through a PCA9685 driver and a stepper motor using an A4988 driver. Safety checks like hard stops and signal validation ensure proper operation.

3.6 Flow Chart



This flowchart shows a (fully integrated) control program of the robotic arm with two different types of control: servo motor and sensor-based gesture through which logical order to run is integrated between them. The system is live in a sense that it relies on bluetooth to receive commands and with the Arduino it translates these to mechanical movements through a blend of programmed conditions and sensor return values. program starts, where servo driver (PCA9685) was initialized and Motor position variables declared. Next it sets up serial communication via Bluetooth handling data transfer from glove interface to robot arm. A loop which checks whether there is any data coming through the serial port. After sensing data, the variable state reads and stores the detected data. This part then looks for what the value of state is versus a list of predefined commands (”s" for base turn, ‘f’ claw grip) and runs the motor with it using conditionals. All motor movement is accompanied by validator bounds checks to make sure they do not exceed the ranges of the servos. Alternatively, if the received command does not match with some pre-defined characters the system will get into gesture based controlling. Then it checks for button_calibrate (bool_calibrate == true). If not, system calibrates it by measuring with a dynamic manner high and low of thumb, finger, pinkie sensors. This process will be executed only once in each session. Post calibration, the flex sensors and (MPU6050) orientation modules values are read live. They are then compared to the thresholds of that user's calibrations to see what movement they wanted. From the comparison, the system commands (open claw, rotate base for example) and transmits the command via Bluetooth to the robotic arm. This loop runs for an infinite time causing the robotic arm to keep responding to either the direct serial commands or gestures. Combining these two control strategies provides a bidentate of precision and user-friendly interaction, the system is ideal for educational and experimental robotics.

3.7 Results and Performance Analysis

The robotic arm used in this project features six degrees of freedom (6-DOF), comprising multiple articulated joints: the Base Joint, Shoulder Joint, Elbow Joint, two Wrist Joints (Wrist Joint 1 & Wrist Joint 2), and the Gripper Joint. Each joint is actuated by a servo motor controlled via Bluetooth communication from the main processing unit. The image below provides a labeled diagram identifying the precise locations and names of each joint.

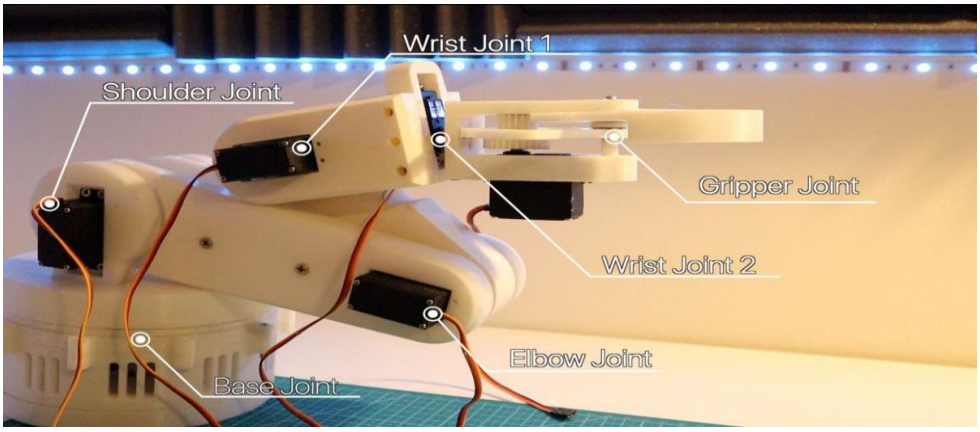


Figure 3.12 Labeled diagram of the robotic arm joints (Base, Shoulder, Elbow, Wrist, and Gripper)

The system relies on a smart glove to detect hand movements and translate them into robotic commands. The glove integrates Flex sensors and an MPU6050 accelerometer to monitor finger bending and wrist orientation. When the user wears the glove and performs gestures, the sensor data is captured and processed by an Arduino Nano, which transmits the commands wirelessly to the robotic arm. This integration enables real-time responsiveness—typically within one second—making the system highly suitable for interactive tasks.

Figure 3.13 The robotic arm responding to real-time hand gestures captured via the smart glove

The implemented robotic arm has proven its capability to execute real-time tasks by responding immediately to hand gestures transmitted via the glove. In this demonstration, the system successfully converted a physical hand motion into precise servo actuation within approximately one second. This level of responsiveness reflects a well-calibrated communication loop between the glove sensors and the arm's microcontroller, emphasizing the system's viability for interactive and assistive applications.

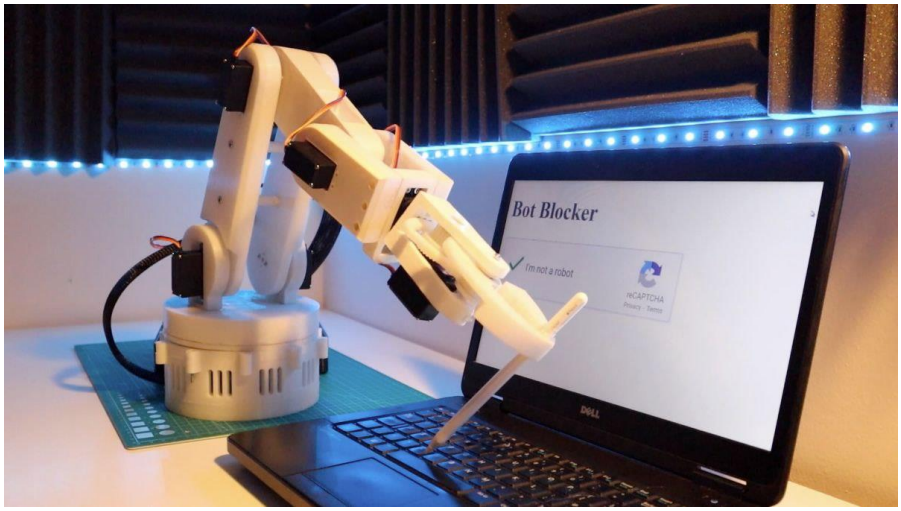


Figure 3.14 Real-time response of the robotic arm to user hand gesture for precision interaction task.

The robotic arm was also able to recognize and respond to gesture inputs to perform object manipulation tasks, such as targeting and preparing to pick up a cylindrical item. In this trial, the system correctly aligned the gripper with the object using only data from the glove's flex and motion sensors, demonstrating its spatial accuracy and ability to interpret user intention in three-dimensional space.

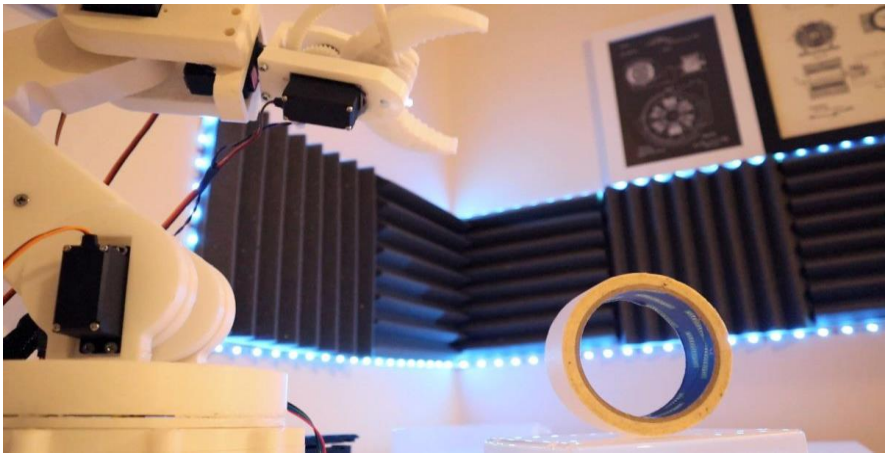


Figure 3.15 Robotic arm aligning with a target object (tape roll) for a grip action using hand gesture input.

3.7.1 Joint Mapping of Glove Inputs to Robotic Arm Movements

To enable intuitive control of the robotic arm, each servo motor is mapped to a specific gesture or hand orientation using data from flex sensors and accelerometers embedded in the glove. The following table provides the complete mapping between glove inputs and robotic joint responses.

Table 3.2 Mapping of Robotic Arm Joints to Glove Gestures and Sensor Values

Number	Servo Name	Robotic Arm Action	Glove Sensor Type	Glove Symbol
1	Base Joint	Rotate Base Left or Right	Accelerometer_2	T, t
2 & 3	Shoulder Joint	Move Shoulder Closer or Further	Pinkie Finger Flex	P, p
3	Elbow Joint	Rotate Elbow Joint	Accelerometer_2	C, c
4	Wrist Joint 1	Move Wrist Up or Down	Accelerometer_1	U, D
5	Wrist Joint 2	Rotate Wrist (CW or CCW)	Accelerometer_1	L, R
6	Gripper Joint	Open or Close Claw Grip	Index Finger Flex	F, f

Note:

- T/t – Rotate Base Left/Right
- P/p – Move Shoulder Forward/Backward (Pinkie bend)
- C/c – Rotate Elbow Joint
- U/D – Move Wrist Up or Down
- L/R – Rotate Wrist Left or Right
- F/f – Close/Open Gripper (Index bend)

CHAPTER FOUR

Coding and Programming

4.1 Glove Program (Transmitter Side)

```
#include<Wire.h>
//Create thumb Sensors
int pinkie = 0; //Pinkie thumb
int finger = 0; //finger thumb
int thumb = 0; //Index thumb
int pinkie_Data = A1;
int finger_Data = A2;
int thumb_Data = A3;
//const int MPU_addr = 0x68;
const int MPU2 = 0x69, MPU1 = 0x68;
//First MPU6050
int16_t AcX1, AcY1, AcZ1, Tmp1, GyX1, GyY1, GyZ1;
int minVal = 265;
int maxVal = 402;
double x;
double y;
double z;
//Second MPU6050
int16_t AcX2, AcY2, AcZ2, Tmp2, GyX2, GyY2, GyZ2;
int minVal2 = 265;
int maxVal2 = 402;
double x2;
double y2;
double z2;
/*Autotune flex parameter
  For Debug Mode. Check the upper and lowe limit of the flex sensors
  3 Flex sensors used. Thumb, Middle, Pinkie*/
int thumb_high = 0;
int thumb_low = 0;
int finger_high = 0;
int finger_low = 0;
int pinkie_high = 0;
int pinkie_low = 0;
//Stop Caliberating the Flex Sensor when complete
bool bool_caliberate = false;
//How often to send values to the Robotic Arm
int response_time = 100;
void setup() {
  pinMode(3, OUTPUT);
  Wire.begin();
  Wire.beginTransmission(MPU1);
  Wire.write(0x6B);// PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true); Wire.begin();
  Wire.beginTransmission(MPU2);
  Wire.write(0x6B);// PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
```

```
  Serial.begin(4800);
  delay(1000);
}
void loop() {
  pinMode(3, HIGH); //Use basic LED as visual indicator if value being sent
  GetMpuValue1(MPU1);
  delay(10);
  //get values for second mpu having address of 0x69
  GetMpuValue2(MPU2);
  delay(10);
  //Print out a value, based on the change of the XYZ co-ordinates of 1st or 2nd MPU
  //Move Left
  if ( x > 15 && x < 55 && y < 30) {
    Serial.print("L");
    delay(response_time);
  }
  //Move Right
  if ( x < 310 && x > 270) {
    Serial.print("R");
    delay(response_time);
  }
  //Claw Up
  if ( y > 60 && y < 80) {
    Serial.print("G");
    delay(response_time);
  }
  //  //Claw Down
  if ( y < 310 && y > 270) {
    Serial.print("U");
    delay(response_time);
  }
  //  //Move right
  if ( y2 > 50 && y2 < 85) {
    Serial.print("C");
    delay(response_time);
  }
  //  //Move left --- Right Hand
  if ( y2 < 160 && y2 > 120) {
    Serial.print("c");
    delay(response_time);
  }
  // read the values from Flex Sensors to Arduino
  pinkie = analogRead(pinkie_Data);
  finger = analogRead(finger_Data);
  thumb = analogRead(thumb_Data);
  //Calibrate to find upper and lower limit of the Flex Sensor
  if (bool_caliberate == false ) {
    delay(1000);
    thumb_high = (thumb * 1.15);
    thumb_low = (thumb * 0.9)
    finger_high = (finger * 1.03);
    finger_low = (finger * 0.8);
    pinkie_high = (pinkie * 1.06);
    pinkie_low = (pinkie * 0.8);
    bool_caliberate = true;
  }
```

```
  delay(response_time);
  // Pinkie
  if (pinkie >= pinkie_high) {
    Serial.print("P");
    delay(response_time);
  }
  if (pinkie <= pinkie_low ) {
    Serial.print("p");
    delay(response_time);
  }
  // thumb 1 - thumb (Base Rotation)
  if (thumb >= thumb_high) {
    Serial.print("T");
    delay(response_time);
  }
  if (thumb <= thumb_low) {
    Serial.print("t");
    delay(response_time);
  }
  // finger 1 - Claw Bend/Open
  if (finger >= finger_high) {
    Serial.print("F");
    delay(response_time);
  }
  if (finger <= finger_low) {
    Serial.print("f");
    delay(response_time);
  }
  else {
    delay(5);
  }
}
 void GetMpuValue1(const int MPU) {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 14, true); // request a total of 14 registers
  AcX1 = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY1 = Wire.read() << 8 |  Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ1 = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp1 = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
 int xAng = map(AcX1, minVal, maxVal, -90, 90);
  int yAng = map(AcY1, minVal, maxVal, -90, 90);
  int zAng = map(AcZ1, minVal, maxVal, -90, 90);
  GyX1 = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY1 = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ1 = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  x = RAD_TO_DEG * (atan2(-yAng, -zAng) + PI) + 4; //offset by 4 degrees to get back to zero
  y = RAD_TO_DEG * (atan2(-xAng, -zAng) + PI);
  z = RAD_TO_DEG * (atan2(-yAng, -xAng) + PI);
}
 void GetMpuValue2(const int MPU) {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 14, true); // request a total of 14 registers
```

```
  AcX2 = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY2 = Wire.read() << 8 |  Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ2 = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp2 = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  int xAng2 = map(AcX2, minVal2, maxVal2, -90, 90);
  int yAng2 = map(AcY2, minVal2, maxVal2, -90, 90);
  int zAng2 = map(AcZ2, minVal2, maxVal2, -90, 90);
  GyX2 = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY2 = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ2 = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  x2 = RAD_TO_DEG * (atan2(-yAng2, -zAng2) + PI) + 4; //offset by 4 degrees to get back to zero
  y2 = RAD_TO_DEG * (atan2(-xAng2, -zAng2) + PI);
  z2 = RAD_TO_DEG * (atan2(-yAng2, -xAng2) + PI);
}
void debug_flex() {
  //Sends value as a serial monitor to port
  //thumb (Claw open / close)
  Serial.print("Thumb: ");
  Serial.print(thumb);
  Serial.print("\t");
  //  //thumb Params
  Serial.print("thumb High: ");
  Serial.print(thumb_high);
  Serial.print("\t");
  Serial.print("T Low: ");
  Serial.print(thumb_low);
  Serial.print("\t");
  //finger (Claw Further)
  Serial.print("finger: ");
  Serial.print(finger);
  Serial.print("\t");
  //  finger Params
  Serial.print("finger High: ");
  Serial.print(finger_high);
  Serial.print("\t");
  Serial.print("finger Low: ");
  Serial.print(finger_low);
  Serial.print("\t");
  //Pinkie (Claw Further)
  Serial.print("Pinkie: ");
  Serial.print(pinkie);
  Serial.print("\t");
  //  //Pinkie Params
  Serial.print("Pinkie High: ");
  Serial.print(pinkie_high);
  Serial.print("\t");
  Serial.print("Pinkie Low: ");
  Serial.print(pinkie_low);
  Serial.print("\t");
  Serial.println();
}
```

4.2 Robotic Arm Program (Receiver Side)

```
/* Include the HCPCA9685 library */
#include "HCPCA9685.h"
/* I2C slave address for the device/module. For the HCMODU0097 the default I2C address
   is 0x40 */
#define  I2CAdd 0x40
/* Create an instance of the library */
HCPCA9685 HCPCA9685(I2CAdd);
//initial parking position of the motor
const int servo_joint_L_parking_pos = 60;
const int servo_joint_R_parking_pos = 60;
const int servo_joint_1_parking_pos = 70;
const int servo_joint_2_parking_pos = 47;
const int servo_joint_3_parking_pos = 63;
const int servo_joint_4_parking_pos = 63;
//Degree of robot servo sensitivity - Intervals
int servo_joint_L_pos_increment = 20;
int servo_joint_R_pos_increment = 20;
int servo_joint_1_pos_increment = 20;
int servo_joint_2_pos_increment = 50;
int servo_joint_3_pos_increment = 60;
int servo_joint_4_pos_increment = 40;
//Keep track of the current value of the motor positions
int servo_joint_L_parking_pos_i = servo_joint_L_parking_pos;
int servo_joint_R_parking_pos_i = servo_joint_R_parking_pos;
int servo_joint_1_parking_pos_i = servo_joint_1_parking_pos;
int servo_joint_2_parking_pos_i = servo_joint_2_parking_pos;
int servo_joint_3_parking_pos_i = servo_joint_3_parking_pos;
int servo_joint_4_parking_pos_i = servo_joint_4_parking_pos;
//Minimum and maximum angle of servo motor
int servo_joint_L_min_pos = 10;
int servo_joint_L_max_pos = 180;
int servo_joint_R_min_pos = 10;
int servo_joint_R_max_pos = 180;
int servo_joint_1_min_pos = 10;
int servo_joint_1_max_pos = 400;
int servo_joint_2_min_pos = 10;
int servo_joint_2_max_pos = 380;
int servo_joint_3_min_pos = 10;
int servo_joint_3_max_pos = 380;
int servo_joint_4_min_pos = 10;
int servo_joint_4_max_pos = 120;

int servo_L_pos = 0;
int servo_R_pos = 0;
int servo_joint_1_pos = 0;
int servo_joint_2_pos = 0;
int servo_joint_3_pos = 0;
int servo_joint_4_pos = 0;
char state = 0; // Changes value from ASCII to char
int response_time = 5;
int response_time_4 = 2;
int loop_check = 0;
int response_time_fast = 20;
```

```
int action_delay = 600;
//Posiion of motor for example demos
unsigned int Pos;
// Define pin connections & motor's steps per revolution
const int dirPin = 4;
const int stepPin = 5;
const int stepsPerRevolution = 120;
int stepDelay = 4500;
const int stepsPerRevolutionSmall = 60;
int stepDelaySmall = 9500;
void setup()
{
  // Declare pins as Outputs
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
  /* Initialise the library and set it to 'servo mode' */
  HCPCA9685.Init(SERVO_MODE);
  /* Wake the device up */
  HCPCA9685.Sleep(false);
  Serial.begin(4800); // Initialise default communication rate of the Bluetooth module
  delay(3000);
}
void loop() {
  if (Serial.available() > 0) { // Checks whether data is coming from the serial port
    state = Serial.read(); // Reads the data from the serial port
    Serial.print(state); // Prints out the value sent
    //For the naming of the motors, refer to the article / tutorial
    //Move (Base Rotation) Stepper Motor Left
    if (state == 'S') {
      baseRotateLeft();
      delay(response_time);
    }
    //Move (Base Rotation) Stepper Motor Right
    if (state == 'O') {
      baseRotateRight();
      delay(response_time);
    }
    //Move Shoulder Down
    if (state == 'c') {
      shoulderServoForward();
      delay(response_time);
    }
    //Move Shoulder Up
    if (state == 'C') {
      shoulderServoBackward();
      delay(response_time);
    }
    //Move Elbow Down
    if (state == 'p') {
      elbowServoForward();
      delay(response_time);
    }
    //Move Elbow Up
    if (state == 'P') {
      elbowServoBackward();
      delay(response_time);
```

```
    }
    //Move Wrist 1 UP
    if (state == 'U') {
wristServo1Backward();
      delay(response_time);
    }
    //Move Move Wrist 1 Down
    if (state == 'G') {
      wristServo1Forward();
      delay(response_time);
    }
    //Move Wrist 2 Clockwise
    if (state == 'R') {
      wristServoCW();
      delay(response_time);
    }
    //Move Wrist 2 Counter-CW
    if (state == 'L') {
      wristServoCCW();
      delay(response_time);
    }
    //Open Claw Grip
    if (state == 'F') {
      gripperServoBackward();
      delay(response_time);
    }
    //Close Claw Grip
    if (state == 'f') {
      gripperServoForward();
      delay(response_time);
    }
  }
}
//Boiler plate function - These functions move the servo motors in a specific direction for a duration.
void gripperServoForward() {
  if (servo_joint_4_parking_pos_i > servo_joint_4_min_pos) {
    HCPCA9685.Servo(5, servo_joint_4_parking_pos_i);
    delay(response_time); //Delay the time takee to turn the servo by the given increment
    Serial.println(servo_joint_4_parking_pos_i);
    servo_joint_4_parking_pos_i = servo_joint_4_parking_pos_i - servo_joint_4_pos_increment;
  }
}
void gripperServoBackward() {
  if (servo_joint_4_parking_pos_i < servo_joint_4_max_pos) {
    HCPCA9685.Servo(5, servo_joint_4_parking_pos_i);
    delay(response_time);
    Serial.println(servo_joint_4_parking_pos_i);
    servo_joint_4_parking_pos_i = servo_joint_4_parking_pos_i + servo_joint_4_pos_increment;
  }
}
void wristServoCW() {
  if (servo_joint_3_parking_pos_i > servo_joint_3_min_pos) {
    HCPCA9685.Servo(4, servo_joint_3_parking_pos_i);
    delay(response_time_4);
    Serial.println(servo_joint_3_parking_pos_i);
    servo_joint_3_parking_pos_i = servo_joint_3_parking_pos_i - servo_joint_3_pos_increment;
```

```
  }
}
void wristServoCCW() {
  if (servo_joint_3_parking_pos_i < servo_joint_3_max_pos) {
    HCPCA9685.Servo(4, servo_joint_3_parking_pos_i);
    delay(response_time_4);
    Serial.println(servo_joint_3_parking_pos_i);
    servo_joint_3_parking_pos_i = servo_joint_3_parking_pos_i + servo_joint_3_pos_increment;
  }
}
void wristServo1Forward() {
  if (servo_joint_2_parking_pos_i < servo_joint_2_max_pos) {
    HCPCA9685.Servo(3, servo_joint_2_parking_pos_i);
    delay(response_time);
    Serial.println(servo_joint_2_parking_pos_i);
    servo_joint_2_parking_pos_i = servo_joint_2_parking_pos_i + servo_joint_2_pos_increment;
  }
}
void wristServo1Backward() {
  if (servo_joint_2_parking_pos_i > servo_joint_2_min_pos) {
    HCPCA9685.Servo(3, servo_joint_2_parking_pos_i);
    delay(response_time);
    Serial.println(servo_joint_2_parking_pos_i);
    servo_joint_2_parking_pos_i = servo_joint_2_parking_pos_i - servo_joint_2_pos_increment;
  }
}
void elbowServoForward() {
  if (servo_joint_L_parking_pos_i < servo_joint_L_max_pos) {
    HCPCA9685.Servo(0, servo_joint_L_parking_pos_i);
    HCPCA9685.Servo(1, (servo_joint_L_max_pos - servo_joint_L_parking_pos_i));
    delay(response_time);
    Serial.println(servo_joint_L_parking_pos_i);

    servo_joint_L_parking_pos_i = servo_joint_L_parking_pos_i + servo_joint_L_pos_increment;
    servo_joint_R_parking_pos_i = servo_joint_L_max_pos - servo_joint_L_parking_pos_i;
  }
}
void elbowServoBackward() {
  if (servo_joint_L_parking_pos_i > servo_joint_L_min_pos) {
    HCPCA9685.Servo(0, servo_joint_L_parking_pos_i);
    HCPCA9685.Servo(1, (servo_joint_L_max_pos - servo_joint_L_parking_pos_i));
    delay(response_time);
    Serial.println(servo_joint_L_parking_pos_i);
    servo_joint_L_parking_pos_i = servo_joint_L_parking_pos_i - servo_joint_L_pos_increment;
    servo_joint_R_parking_pos_i = servo_joint_L_max_pos - servo_joint_L_parking_pos_i;
  }
}
void shoulderServoForward() {
  if (servo_joint_1_parking_pos_i < servo_joint_1_max_pos) {
    HCPCA9685.Servo(2, servo_joint_1_parking_pos_i);
    delay(response_time);
    Serial.println(servo_joint_1_parking_pos_i);
    servo_joint_1_parking_pos_i = servo_joint_1_parking_pos_i + servo_joint_1_pos_increment;
  }
}
void shoulderServoBackward() {
```

```
if (servo_joint_1_parking_pos_i > servo_joint_1_min_pos) {
    HCPCA9685.Servo(2, servo_joint_1_parking_pos_i);
    delay(response_time);
    Serial.println(servo_joint_1_parking_pos_i);
    servo_joint_1_parking_pos_i = servo_joint_1_parking_pos_i - servo_joint_1_pos_increment;
  }
}
void baseRotateLeft() {
  //clockwise
  digitalWrite(dirPin, HIGH);
  // Spin motor
  for (int x = 0; x < stepsPerRevolution; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(stepDelay);
  }
  delay(response_time); // Wait a second
}
void baseRotateRight() {
  //counterclockwise
  digitalWrite(dirPin, LOW);
  // Spin motor
  for (int x = 0; x < stepsPerRevolution; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(stepDelay);
  }
  delay(response_time); // Wait a second
}
void wakeUp() {
  //Pre-Program Function - Wake Up Robot on Start
  if (loop_check == 0) {
    //    //Shoulder Raise
    for (Pos = 0; Pos < 10; Pos++)
    {
      HCPCA9685.Servo(1, Pos);
      delay(response_time_fast);
    }
    //  //Move Elbow Backwards
    for (Pos = 400; Pos > 390; Pos--)
    {
      HCPCA9685.Servo(2, Pos);
      delay(response_time_fast);
    }
    //Move Wrist 1 Forward
    for (Pos = 10; Pos < 20; Pos++)
    {
      HCPCA9685.Servo(3, Pos);
      delay(response_time);
    }
    //Move Wrist 2 Backwards
    for (Pos = 380; Pos > 50; Pos--)
```

```
    {
      HCPCA9685.Servo(4, Pos);
      delay(response_time);
    }
    //Move Wrist 2 Backwards
    for (Pos = 50; Pos < 150; Pos++)
    {
      HCPCA9685.Servo(4, Pos);
      delay(response_time);
    }
    //Move Wrist 1 Forward
    for (Pos = 19; Pos < 100; Pos++)
    {
      HCPCA9685.Servo(3, Pos);
      delay(response_time);
    }
    loop_check = 0;
  }
}
void flexMotors() {
  //Example Demo Pre-program Function to Make Robot Wake Up (Motor by Motor)
  if (loop_check == 0) {
    delay(action_delay);
    //Move Wrist 1 Forward
    for (Pos = 100; Pos > 10; Pos--)
    {
      HCPCA9685.Servo(3, Pos);
      delay(10);
    }
    delay(action_delay);
    //Move Wrist 1 Forward
    for (Pos = 10; Pos < 70; Pos++)
    {
      HCPCA9685.Servo(3, Pos);
      delay(10);
    }
    delay(action_delay);
    baseRotateLeft();
    delay(action_delay);
    //Move Wrist 2 Backwards
    for (Pos = 200; Pos < 380; Pos++)
    {
      HCPCA9685.Servo(4, Pos);
      delay(10);
    }
    delay(action_delay);
    loop_check = 1;
  }
}
```

CHAPTER FIVE

Conclusion and Future Work

5.1 Conclusion

This project presented the successful design and implementation of a gesture-controlled robotic arm using low-cost components such as Arduino boards, servo motors, flex sensors, and MPU6050 modules. The system was able to interpret human hand gestures in real time and execute corresponding robotic actions with a response time of less than one second. Throughout testing, the arm demonstrated precise control in movements like gripping, rotating, and reaching. The communication between the glove and robotic arm, handled via Bluetooth, proved stable and reliable under typical operating conditions. Overall, the system validated the feasibility of using wearable gesture input to control robotic hardware in practical scenarios. This establishes a strong foundation for future enhancements in interactive robotics.

5.2 Future Work

Several directions can be pursued to enhance the capabilities of the system. One major improvement would be integrating a feedback mechanism, such as position sensors, to enable closed-loop control and better accuracy. Additionally, replacing Bluetooth with Wi-Fi or IoT modules would allow for remote and cloud-based control features. The gesture recognition process could also be improved by incorporating AI algorithms, enabling the system to learn and adapt to a wider range of hand gestures. Mechanically, the design could be enhanced by using stronger servos or lighter materials to improve payload capacity.

Reference

[1] Book, Schilling J.. "Fundamentals Robotics Analysis and Control", Prentice Hall, 1996.

[2]Harris, Tom. "How Robots Work". How Stuff Works.

http://science.howstuffworks.com/robot.htm

[3] http://www.preterhuman.net

[4] "Robot (technology)". Encyclopedia Britannica Online. Internet Survey:

http://www.britannica.com/EBchecked/topic/505818/robot.

[5- Internet Survey: http://www.avinashgandi.com/projects-amp- papers.html

[6] Al-Mawrid Dictionary, Baalbaki, Beirut, Lebanon

[7] Polk Igor (2005). "RoboNexus 2005 Robonexus Exhibition 2005 robot exhibition virtual our".

[8]Arrick Robotics: Building your first robot.

[9] Robot Oppression: Unethicality of the Three Laws

[10] http://www.dhadh.com/page.php?id=9568

[11] Devi, V. G., & Vijetha, T. (2016, June). MEMS controlled robotic arm with gestures. International Journal of Scientific Development and Research (IJSDR), 1(6), 82–84.

[12] Hameed, S., Khan, M. A., Kumar, B., Arain, Z., & Hasan, M. U. (2017, December). Gesture controlled robotic arm using Leap Motion. Indian Journal of Science and Technology, 10(45). https://doi.org/10.17485/ijst/2017/v10i45/120630

[13] Vala, S. T. (2018, November). Hand gesture controlled robot using Arduino. International Journal of Research in Engineering, Science and Management (IJRESM), 1(11), 194–196.

[14] Elam Cheren, S., Madhubala, S., Pradeepa, C., Poovitha, M., & Madumitha, B. (2020, August). Hand gesture controlled robot arm. International Journal of Engineering and Advanced Technology (IJEAT), 9(6), 405–407. https://doi.org/10.35940/ijeat.E1019.089620

[15] Rahman, M. Z., Khatun, M. A., & Ahmed, T. (2022, May). Gesture controlled robotics hand. International Journal of Computer Applications Technology and Research (IJCATR), 13(5), 8–11. https://doi.org/10.7753/IJCATR1305.1002

[16] Kesharwani, A., Chaudhary, A. P., Singh, B. P., Kumar, V., & Manjunath, T. C. (2023). A study on hand motion controlled robotic arm. Journal of Propulsion Technology, 44(3), 812–814.

[17] Lohakare, H., Chainde, T., Jadhav, P., Rahangdale, K., Wanjari, D., & Ikhar, A. (2024, March). Design and development of robotic arm control by human hand. International Journal of Research Publication and Reviews, 5(3), 4247–4252. https://doi.org/10.55248/gengpi.5.0324.07104

[18] Prathibha, K., Jagadeesha, S., Sonu, S., Priyanka, N., Shahid, M., & Kaleem, S. S. (2024, May). Design and development of gesture controlled robotic arm. International Journal of Computer Applications Technology and Research (IJCATR), 13(5), 8–11. https://doi.org/10.7753/IJCATR1305.1002

[19] Anonymous. (2025). Arduino-based gesture-controlled robot. Unpublished undergraduate project report.

[20]Zhou, H., & Hu, H. (2008). Human motion tracking for rehabilitation—A survey. Biomedical Signal Processing and Control, 3(1), 1–18. https://doi.org/10.1016/j.bspc.2007.09.001

[21] Satghare, N. S. (2019). Gesture controlled robotic arm. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 7(9), 401–407. https://doi.org/10.22214/ijraset.2019.9056

[22] https://www.uruktech.com/product/mg996/

[23] https://mechtex.com/products/hybrid-stepper-motors/nema17

[24] Texas Instruments. (2014). DRV8825 Stepper Motor Controller IC datasheet (Rev. F).