ROBOTIC HAND CONTROLLED BASED ON FLEXIBLESENSOR BY GLOVE USING WIRELESS COMMUNICATION

DHILEN A/L MANIMARAN

BACHELOR OF ELECTRICAL
ENGINEERING(ELECTRONICS)
WITH HONOURS

UNIVERSITY MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : DHILEN A/L MANIMARAN

Date of Birth : 27 JANUARY 1997

Title : ROBOTIC HAND CONTROLLED BASED ON FLEXIBLE SENSOR BY GLOVE USING WIRELESS COMMUNICATION

Academic Session : 2022/2023

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserve the right as follows:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies for the purposeof research only.
3. The Library has the right to make copies of the thesis for academic exchange.

(Student's Signature)	(Supervisor's Signature)
970127-07-5497	DR. IZZELDIN IBRAHIM MOHAMED ABDELAZIZ
New IC/Passport Number Date: 10/02/2022	Name of Supervisor Date: 13/02/202

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter page 2 from the organization with the period and reasons for confidentiality or restriction.

SUPERVISOR’S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the Bachelor of Electrical Engineering (Electronics) with Honours

Signature :

Name of Supervisor : Sir Izzeldin Ibrahim Mohamed Abdelaziz

Position : Lecturer

Date : 13/02/202

Universiti
Malaysia
PAHANG
Engineering • Technology • Creativity

STUDENT’S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citation which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

Signature :

Name : Dhilen A/L Manimaran

ID Number : EA18016

Date : 10/02/2022

ROBOTIC HAND CONTROLLED BASED ON FLEXIBLE SENSOR BY GLOVE USING WIRELESS COMMUNICATION

DHILEN A/L MANIMARAN

Thesis submitted in fulfillment of the requirements
for the award of the
Bachelor of Electrical Engineering (Electronics) with Honours

College of Engineering
UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2022

ACKNOWLEDGEMENT

First and foremost, I would like to express my heartfelt appreciation to my supervisor, Dr. Izzeldin Ibrahim Mohamed Abdelaziz, for this support and encouragement over the course of this project. Many facets of this project have been investigated under supervision, and with his expertise, ideas, and assistance, this thesis will be presented in the time allotted. My gratitude also goes to my dad, who have been so patient and supportive over the years. Thank you for your motivation, affection, and emotional support. Thank you to my friends who assisted me, directly or indirectly in completing this project.

Finally, my heartfelt gratitude goes to my parents for their unwavering love and care.

ABSTRAK

Tangan robot kemanusian adalah komponen penting. Sukar untuk membuat aksi yang kelihatan seperti sisi manusia. Kajian ini mencadangkan tangan robotik yang sama dan ringan dan menyiasat kemungkinan kawalan berdasarkan sensor fleksibel dengan sarung tangan menggunakan komunikasi tanpa wayar. Tangan robot dibina untuk bergerak dan berfungsi seperti tangan manusia. Salah satu jari tangan robot mempunyai tiga darjah kebebasan, hampir sama dengan jari manusia. Apabila jari manipulator beralih, sarung tangan menggunakan komunikasi tanpa wayar menghantar isyarat maklum balas ke mikrokontroler. Setiap sektor jari tangan robot dikuasakan oleh pemacu rantai motor servo. Selanjutnya, kerana motor servo hanya dapat membekalkan tork yang rendah untuk jari, tangan robot hanya dapat menghasilkan daya yang sesuai untuk setiap jari. Arduino Mega memantau keseluruhan tingkah laku dan aktiviti robot. Tangan robot diuruskan secara langsung oleh sarung tangan menggunakan komunikasi tanpa wayar. Akibatnya, ia dapat melakukan tugas yang serupa dengan yang dilakukan oleh tangan manusia, seperti mencengkam objek. Tangan robot akan digunakan untuk melaksanakan robot humanoid dengan kajian dan pengembangan lebih lanjut.

ABSTRACT

A humanoid robot's hand is a vital component. It's tough to create an action that looks like a human side. This study suggested a similar-sized and light-weight robotic hand and investigated the possibility of controlling based on flexible sensor by glove using wireless communication. The robotic hand is built to manoeuvre and function like a human hand. One of the robot hand's fingers has three degrees of freedom, nearly identical to a human finger. When the fingers of the manipulator switch, the glove using wireless communication sends a feedback signal to the microcontroller. Each robot hand's finger sector is powered by a Servo Motor chain drive. Furthermore, since the servo motor can only supply a low torque for the finger, the robot hand can only produce a force that is appropriate for each finger. Arduino Mega monitor the robot's entire behaviour and activity. The robot hand is managed in real time by the glove using wireless communication. As a result, it may perform tasks similar to those performed by a human hand, such as gripping an object. The robot hand will be used to implement humanoid robots with further research and development.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VDC	Volts of Direct Current
LARM	Low Cost Easy Operation Robot Hand
SVM	Support Vector Machine
DOF	Degree of Freedom
UART	Universal Asynchronous Receiver/Transmitter
MCU	Microcontroller Unit
PWM	Pulse-Width Modulation
PID	Proportional integral derivative
TCP/IP	Transmission Control Protocol/Internet Protocol
IDE	Integrated Drive Electronics

CHAPTER 1

INTRODUCTION

1.1 Project Background

A humanoid robot's hand is a part of its limb, and it is a vital component. Welding, painting, ironing, assembling, pick and put, packing, and testing are some of the popular robot hand applications, many of which require a lot of agility, speed, and precision. In today's modern world, a robot hand can easily be found to perform hazardous and precise tasks in place of a human hand. For quality management and lifting large products, industrial robot hands are used in the manufacturing and transportation processes.

Robots could be used in strategic and aerospace operations even in a military base for research and demining. To avoid casualties, these robots use a robot hand in lieu of a human demolition bomb. To ensure the safety of the spacecraft, astronauts will allow the robot to inspect the surrounding areas. Furthermore, the robot hand is often used by doctors to assist patients undergoing surgical surgery. To prevent making an error, this necessitates extremely precise technology.

The majority of robot hands in the world are made for a specific function. As a result, the aim of this project is to research and build a robotic hand controlled by glove using wireless communication. Using the glove with wireless communication technology, the robot's hand can be manipulated. The robot hand's expression should be as similar to that of a human hand as possible.

1.2 Problem Statement

The difficult part is developing a robotic hand that can be controlled by a glove via wireless transmission in order to get a precise and accurate grasp. It is a close approximation of the human hand's flexibility and sensitivity. There are many different types of robotic hands and their applications available nowadays. The most crucial factors to evaluate are their stability, dependability, and financial viability. The main elements of a robot's hand are not the same as a human's. The cost of every robot hand mechanism is inextricably linked. The most difficult job is to simplify the robot mechanism while keeping it low-cost and human-like. As a result, our project will focus on the design and production of human hands, particularly for master-slave communication networks using RF Module.

1.3 Objective

The objectives of the project are as follows:

The aims of this work are to develop a robotic hand based on flexible sensor by glove using wireless communication that is able to:

i. Via wireless contact, create a robotic hand that can mimic the gestures of a user-worn controller glove as well as grip objects.

ii. Using wireless networking and a controller glove with five flex sensors, the hand should be able to grip surfaces of different sizes and weights.

1.4 Scope of Project

A mechanical architecture, electronic hardware, and software comprise the device. The project's scope of work is restricted to mechanical design, electronic control, and software creation.

The robot's mechanical components include a finger skeleton, engine, and caster mounting. Strings are attached to the servo motor's shaft and are used to pull each sector of the finger. This three-degree-of-freedom robot finger is designed to run. Using a future meter, the glove configuration incorporates a sensor in each finger joint.

Arduino Mega microcontroller is used in the electronic control system. The input is controlled by a potentiometer, and the output is also controlled by a potentiometer rotate sensor. When the signal from the main Arduino Mega microcontroller is received, the servo controller will produce a servo pulse that lasts between 0.5 and 2.5 milliseconds.

To run Arduino Mega, a compilation of C programming code must be coded into the computer using ARDUINO IDE software, and the programme must be modified depending on the requirements of the real-world robot hand attitude. Proteus is the programme that was used to build the schematic diagram.

CHAPTER 2

LITERATURE REVIEW

2.1 Robotic hand and Humanoid Robot

Due to the vast applications ground, such as pick-and-placed, upper limbs prosthesis, as well as modular automation of many manufacturing activities and automates assembly duties, there has been an increasing interest in robotic hands production in recent years. Current technological advances are mostly limited to special-purpose grippers and instruments, which are inadequate for dealing with items of any form or scale. Adopting the human hand architecture to accomplish the above applications seems to be an adequate approach, since the human hand is the most advanced and complex outer extremity on the human body and has evolved over millennia. As a result,[1] anthropomorphic robotic hands with a high degree of anthropomorphism are a desirable aim in final effectors design, since anthropomorphic hands enable robotic hands to be used as prosthetics, work in man-oriented settings, and be tele-operated by humans through interfaces that can replicate the operator hand conduct. Despite the extensive study that has been conducted and published in the literature, as well as advancements in architecture, control, and sensory, there are still problems to be resolved or improvements to be made on robotic hands. [2]The word "dexterity" is used to describe an index that measures the robotic hand's capability. A dexterous robotic hand has several features that make it a suitable design.

2.2 Actuators

The motors that move the robot are known as actuators. [3]The actuators of a device or robot are designed to imitate the muscles and joints of an animal or human body, so they work similarly to muscles and joints but with a different configuration. Robots usually use rotary actuators to accomplish the robot's impact motion. Electric, pneumatic, hydraulic, piezoelectric, or ultrasonic power are all typical actuators.

Figure 2.2 1 Types of Robotic Actuators

2.2.1 Servomotor

A servo motor is a term that refers to some kind of automatic control system. The Servo is an automated system that corrects the output of a function using error-sensing feedback. Only systems in which input or error-correction signals are used to monitor mechanical location or other parameters are properly referred to as feedback or error-correction systems. In practice, this translates to a system that you can set and forget, and that changes itself over time as a result of feedback.

There are several different types of servos, each with its own accuracy, speed, and weight. These servos have the same relation and are operated by three wires (see Figure

2.2.1.1) : negative, positive, and signal. The servo motor is driven by 6VDC, and the direction is indicated by a PWM pulse stream.

Figure 2.2.1 1 Servomotor and the Connectors

The servo would be set to its "neutral" spot, or 90°, with a 1.5 ms servo pulse. A 1.25 ms servo pulse, for example, could set the servo to 0°, while a 1.75 ms servo pulse could set the servo to 180°. The servo hardware's physical limits and timings vary by manufacturer and type, but a general servo's angular motion can move anywhere between 180° and 210°, and the neutral direction is about 1.5 ms.

2.2.2 Flex Sensor

Flex sensors usually come in two types. One is 2.2 inches long and the other is 4.5 inches long. Despite the differences in scale, the core function remains the same. They're also split up by opposition. There are three forms of resistance: LOW resistance, MEDIUM resistance, and HIGH resistance. Based on the situation, choose the right form. The FS-L-0055 2.2inch Flex sensor will be discussed in this article.

The flex sensor has two terminals. Unlike a diode, the Flex sensor does not have polarised terminals. As a result, there is no such thing as a positive or negative.

Pin Numbers	Description
P1	Connected to power source
P2	Connected to ground

Table 2.2.1 2 Shows pin configuration

As previously said, a FLEX SENSOR is essentially a VARIABLE RESISTOR with increased terminal resistance when the sensor is twisted. As a result, surface linearity affects sensor resistance. As a result, it's commonly used to detect variations in linearity.

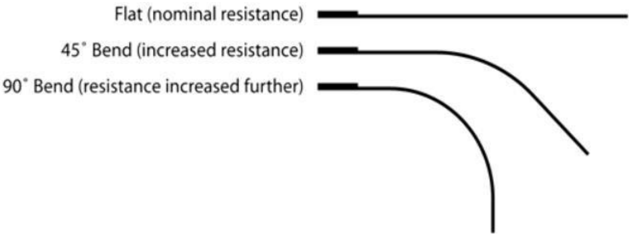


Figure 2.2.1 3 Shows the resistance of flex sensors

When the surface of the[4] FLEX SENSOR is totally linear, it will have its nominal resistance, as seen in the diagram above. The FLEX SENSOR resistance rises to double what it was before when bent at a 450 angle. When the bend is 90 degrees, the resistance can reach four times the nominal resistance. As a result, the resistance around the terminals increases linearly as the bent angle increases.

2.2.3 Communications

Contact between microcontrollers and microcontrollers with the machine is needed in my project, and the UART will be used. So, a universal asynchronous receiver/transmitter (UART)[5] is a kind of "asynchronous receiver/transmitter," which is a piece of computer hardware that converts data between parallel and serial types.

Serial transmission of digital data over a single wire or wireless medium is much more cost effective than simultaneous transmission over several wires by using UART. UART may be used to transmit information between sequential and parallel forms at either end of the chain. Each UART includes a shift register, which is the fundamental method of serial-to-parallel conversion. There are two types of communication that can be used: full duplex and half duplex. Half-duplex systems take turns sending and receiving, while full duplex devices send and receive at the same time).

UARTs are now widely used with RS-232 to communicate with embedded devices. It is beneficial to connect with microcontrollers as well as between microcontrollers and PCs. Many chips have UART features, and low-cost chips that transform logic level signals to RS-232 level signals, such as Maxim's MAX232, are widely available.

2.3 Characteristics of Human Hand

The [6]anatomical hand has 27 main bones (eight carpals, five metacarpals, and the phalanges), and at least 18 joint articulations of 26 degrees of freedom (DoF) powered by around forty muscles; grasps involving all hand fingers will exert up to 400N of power. Each finger has a proximal, central, and distal phalanx, as well as proximal and distal phalanges on the thumb. Only the proximal phalanges show abduction and adduction, while all of the finger joints can bend and extend. Flexion/extension, abduction/adduction, and rotating along the axis of the metacarpal joint on the metacarpal phalange are all shown by the palm, which has the most intricate structure. The musculoskeletal skeleton of the human hand and its gestures are depicted in figure 2.3.1

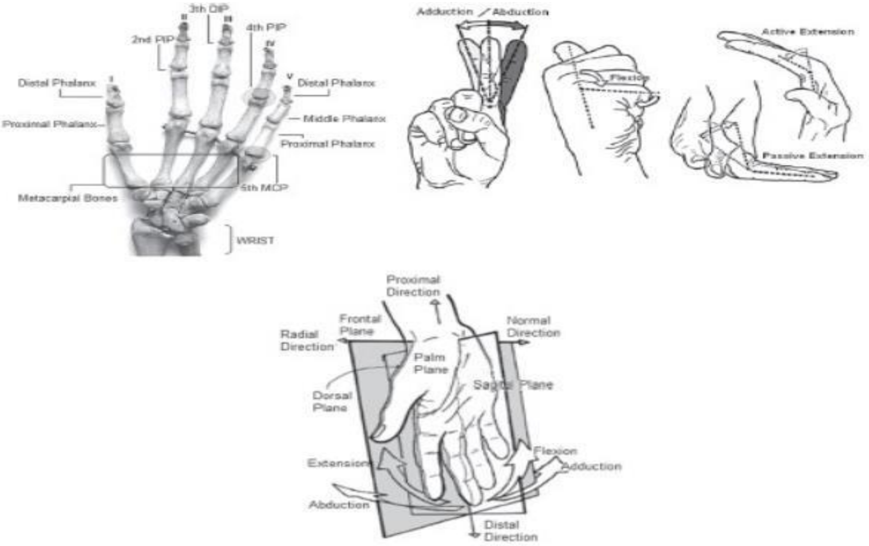


Figure 2.3 1 Shows the human hand and the musculoskeletal structure

2.4 Articles/Journals Review

2.4.1 LARM Hand with Human Size

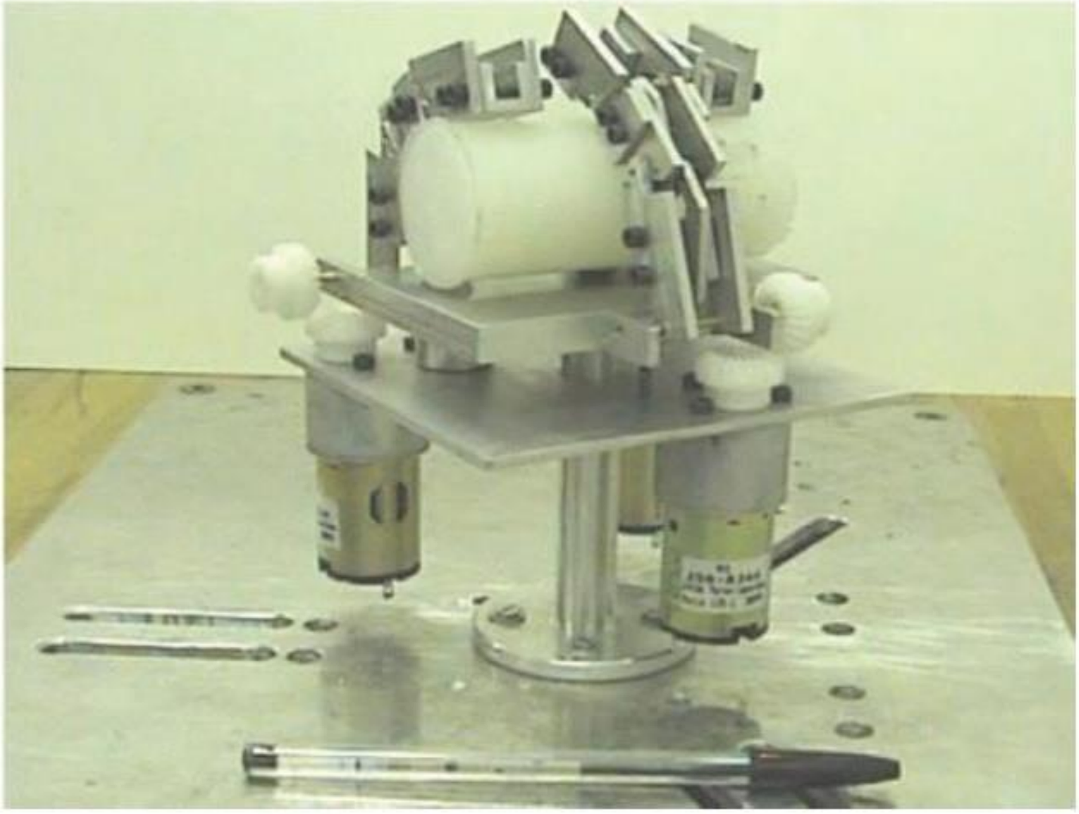


Figure 2.4 1 shows the LARM hand with human size

Jorge Eduardo Parada Puig, Nestor Eduardo Nava Rodriguez and Marco Ceccarelli has presented the development of LARM hand in Figure 2.4.1. By the use of the proposed technique, LARM Hand has been improved. [7]The LARM Hand was created with industrial manipulations and human prosthesis in mind. A prototype for grasping objects with normal geometries, such as cylinders, cubes, and spheres, has been developed. In human activity, grip force has also been calculated and analysed. A measuring device was mounted on a volunteer's human hand in order to investigate the grip forces that can be exerted during the grasp of objects with normal geometries experimentally. The grasping forces obtained by human hands will be used to verify the grasping forces obtained by robotic hands during testing. For calculating grasp forces, commercial force sensors were used, and a virtual instrument was created in the LabView setting to control the force sensors. LARM's mechanism of action. As a set of crossed-four bar linkages, the hand fingers are depicted. A robotic finger can execute motions that are similar to those of a human index finger using this articulated mechanism, but through a mechanical action of just one degree of freedom. The finger mechanism has been subjected to repeated kinematic analysis in order to achieve optimum measurements.

2.4.2 Real Time Control of Robotic Hand by Human Hand

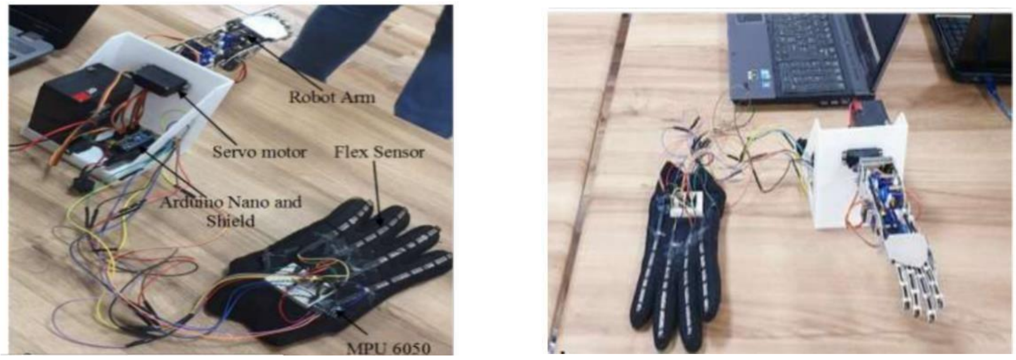


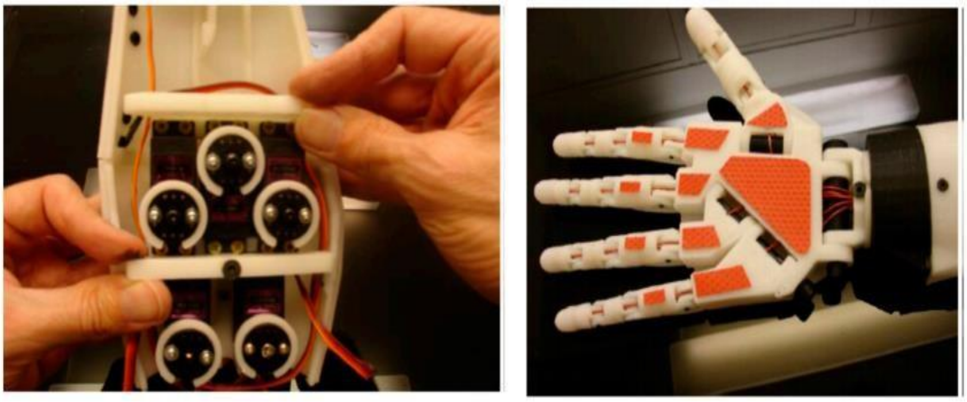
Figure 2.4 2 shows the Real time control of Robotic hand by human hand

[8]*Enas A. Khalid, Wisam T.Abbood, Oday I. Abdullah* Presented and developed this Real time control of robotic hand by human hand as shown in Figure 2.4.2 above. The proposed control is based on the Arduino controller and the Flexible sensor. It uses a glove to transfer motions, simulating the action of the human hand's five fingers with the robotic hand's five

fingers. The robotic hand's job is to grab some target while being controlled in real time. The Arduino controller and the Flexible sensor are used in the proposed regulation. It transfers gestures using a glove, simulating the human hand's five fingers with the robotic hand's five fingers. It is the duty of the robotic hand to catch a target while being operated in real time.

To pass portions of the finger to the fingertip, the friction force must be at its height at 90 degrees. At the end of gripping the material, the friction force decreased. For the experimental value of all joints in the finger to fingertip, the torque is calculated using the Jacobian matrix. The torque begins with a few values and gradually increases until the object is grasped. When measuring velocity, keep in mind that there is a positive relationship between velocity and angle. As you move the fingertip closer to the object, the velocity reduces in value as the angle increases. By grasping the target, the changes in values caused by the fingertip reach the object.

2.4.3 Anthropomorphic Robotic Hand



a) b)

Figure 2.4 3 shows the 3D printing of Anthropomorphic Robotic hand: a) interior b) exterior.

Refer to Figure 2.4.3, Rahul Raj Devaraja, Rytis Maskeliunas and Robertas Damasevicius describe a [9]Anthropomorphic Robotic hand which is to object grasping and shape recognition. an anthropomorphic multi-finger artificial hand that senses the form of the grasped object for a fine-scale object grasping task. The 3D printer was used to develop the robotic forearm, which has a servo bed for stand-alone finger movement. The Leap Motion unit is used to collect data on the robotic fingers' angular orientation, and a hybrid Support Vector Machine (SVM) classifier is used to identify object type. Using supervised learning techniques, it trained the designed robotic hand on a few monotonous convex-shaped things that are identical to commonplace objects (ball, cone, and rectangular box). Item form recognition accuracy is 94.4 percent on average.

The grasping task will be performed by the robotic hand on an unknown shape object, and the interaction with the object will be registered by the pressure sensors. The servo motors that regulate the motions of the hand tendons will then come to a halt, and the spatial locations of the finger joints will be registered. Fingers' spatial locations are converted to angles (in degrees) and saved as a dataset. Following data normalisation, the data are sent to the spot-checking process, which validates the data and eliminates measurements with corrupt (or unavailable) values. After that, a prediction of the shape of a grasped target is carried out. The thrilling action is replicated if there is insufficient evidence to allow an accurate forecast.

2.4.4 Five Fingered Robotic Hand

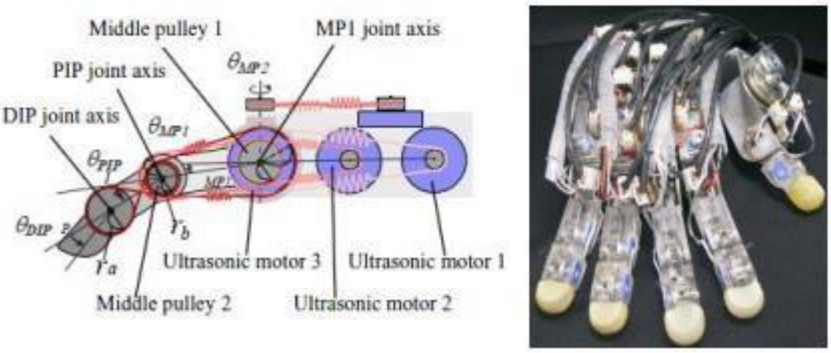


Figure 2.4 4 shows the developed five fingered robotic hand

Many robotics hand methods have been suggested in the literature. Almost all of them touch on prior research in the field of robotics and teleoperation. *Ikuo Yamano and Takashi Maeno* developed a [10]tele-operation five-fingered robot with about the same amount of degrees of freedom as a human hand. Ultrasonic motors and elastic components are used to move the robot hand in a novel way. The approach employs restoring force as a driving force in gripping objects, allowing the hand to execute steady and obedient grasping motions without the use of any additional strength. Ultrasonic motors provide high torque at low speeds and are used in multi-DOF mechanisms as a driving tool. A wire-driven mechanism overcomes a design constraint in the finger section. As a result, the robot hand has 20 degrees of freedom and resembles a human hand in shape. For force power, a Jacobian Matrix was used, and an Analog to Digital Converter was used as the control mechanism for this side.

2.4.5 HIT/DLR Hand with Data Glove and Cybergrasp

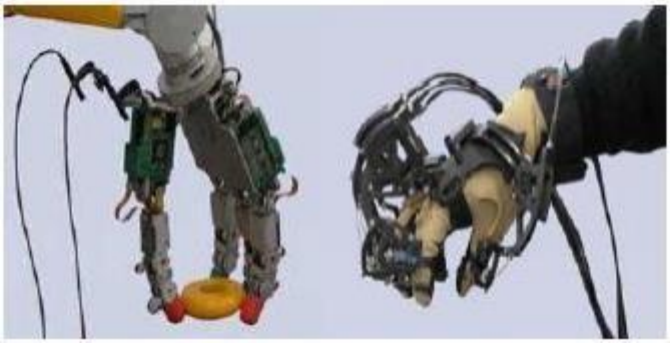


Figure 2.4 5 shows HIT/DLR hand with DataGlove and CyberGrasp

A master-slave tele-operation system is defined by [11]Haiying Hu, Jiawei Li, Zongwu Xie, Bin Wang, Hong Liu, and Gerd Hirzinger to assess the efficacy of tele-presence in tele-robotics applications. The user controls the dexterous hand with a data glove and a Spaceball. Visual telepresence devices gather remote operation scenes and show them to the operator through a stereo mask, and contact forces determined by finger sensors can be fed back to the operator. This robot arm developed a teleoperation system that combined high robot dexterity with deep human immersive power. Interface input devices such as the Space Mouse and Dataglove, as well as telepresence devices such as the CyberGrasp force feedback system and the helmet vision feedback device. An arm/hand robot system, bench, parallel hand-eye cameras system, and world cameras system are all used in the tele-robot system. The hand is an HIT/DLR dexterous hand, and the robot arm is a Staubli RX60 robot. Finally, the local network connectivity system links the human operation interface system and the tele-robot system using the TCP/IP protocol and the Sever/Client mode. The high-speed serial contact between the hand and PID position control systems can obey the commanded position trajectory, and impedance joint torque control is introduced for motion control in confined environments by monitoring a complex relation between the active force and impedance torque control.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explains the methods used and the factors that were taken into consideration for this initiative. It starts with a discussion of the project flow, then moves on to the device design process, as well as the methods and software used in this project. A cost-effective, appropriate, and good material selection is critical to a project's progress and perfection. In order to create well-functioning circuits, it is critical to choose the most suitable components with the right requirements. The concept can be applied to both hardware and software development.

3.2 Development Process

The aim of this project is to create and test the wireless communication using glove to control a robot hand. To do so, the most critical disciplined need to look at is the techniques and technological solutions suggested. As a result, a simpler phase-by-phase approach has been proposed. Problems can be observed early on with this approach, avoiding stressful failures. There is still an aim to achieve, whether it is short-term or long-term. It is more organized to complete the tasks one at a time, according to their stages. As a result, the robot's creation is split into three stages. Mechanical design, electronic system design, and software development are among them.

3.3 Mechanical Design

The dynamical model that describes the contact substantially changes with the type of surface involved. Although the surface of the object cannot being known a priori (can be both rigid and soft), the surface that covers the hand represents an important degree of freedom in the design. In general the surfaces of a robotic grippers can be rigid or soft. The choice of having a rigid surface is justified when the type of grasp expected by the application is a firm and power grip or if the expected objects to handle are soft. In other cases, providing at least zones of the fingers covered with soft layers, is preferable.

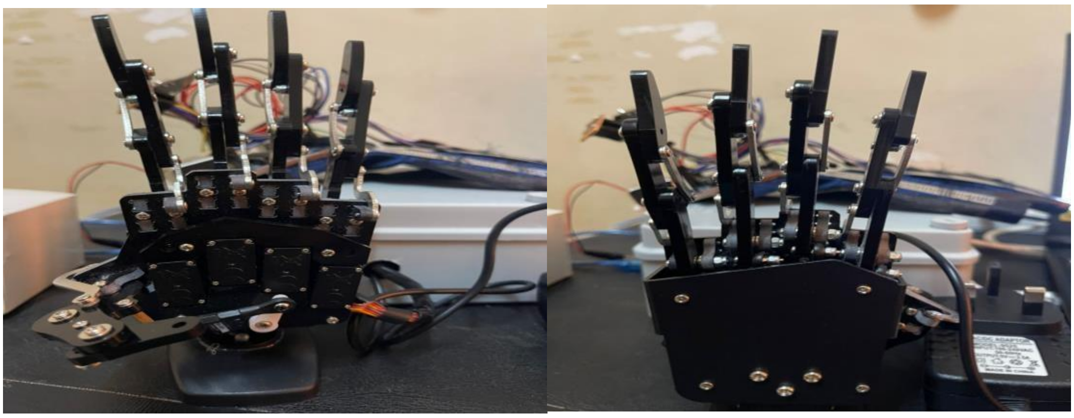


Figure 3.3 1 Shows the Robotic Hand

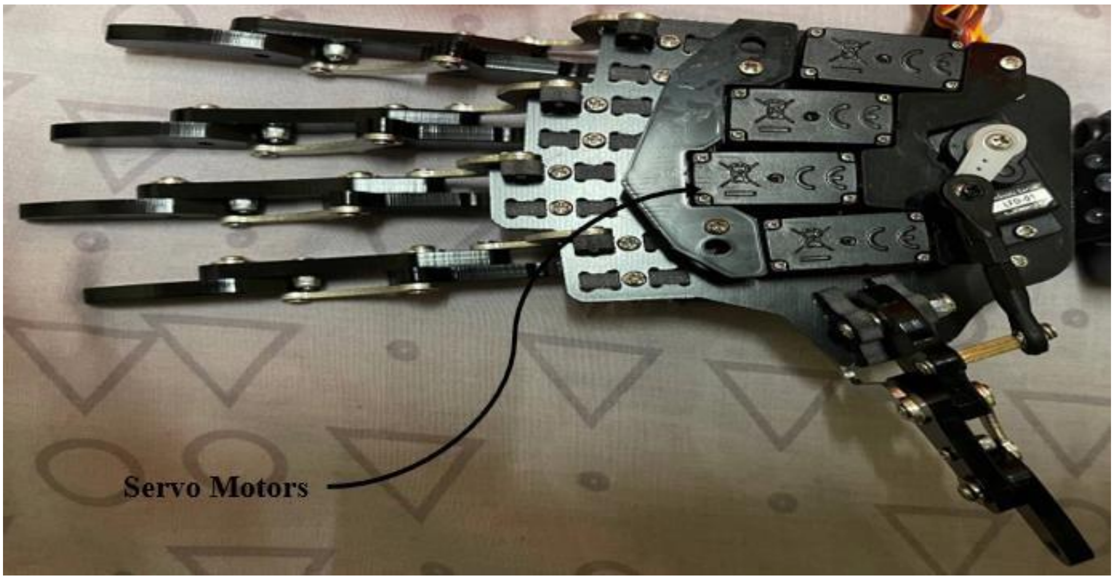


Figure 3.3 1 Show the attachment of motors in each finger and the robotic hand

3.3.1 The servomotor

A servo motor (servo) is an electromechanical device in which the position of a motor's armature is determined by an electrical input. It was chosen to power this robot hand due of its excellent torque and ability to calculate position via PWM. Figure 3.2.1.1 shows the wires of the servo motors.

Figure 3.3.1 1 shows the wires of servo motors

The specifications of the Servomotor were listed as below:

Parameter	**Specification**
supply	5V
speed	0.14s/60°
torque	4.50 kg.cm

Table 3.3.1 1 shows the servomotor specification

3.3.2 Master Glove Design

A flex sensor, sometimes known as a bend sensor, is a sensor that senses bending or deflection. The sensor is normally glued to the surface, and the sensor element's resistance is modified by bending the surface. It's called a goniometer or flexible potentiometer since the resistance is proportional to the amount of bend. Flex sensors potentiometers will be inserted at each finger of the master glove in this project to measure our hand's position. The potentiometers' output voltage was transformed to a 10-bit digital value and saved in the ADC register of the microcontroller. The flex sensor shown in the figure 3.2.2.1

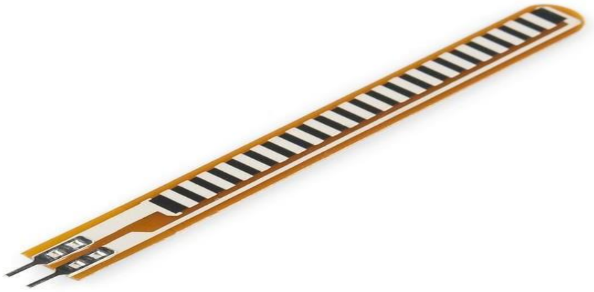


Figure 3.3.2 1 shows the flex sensors

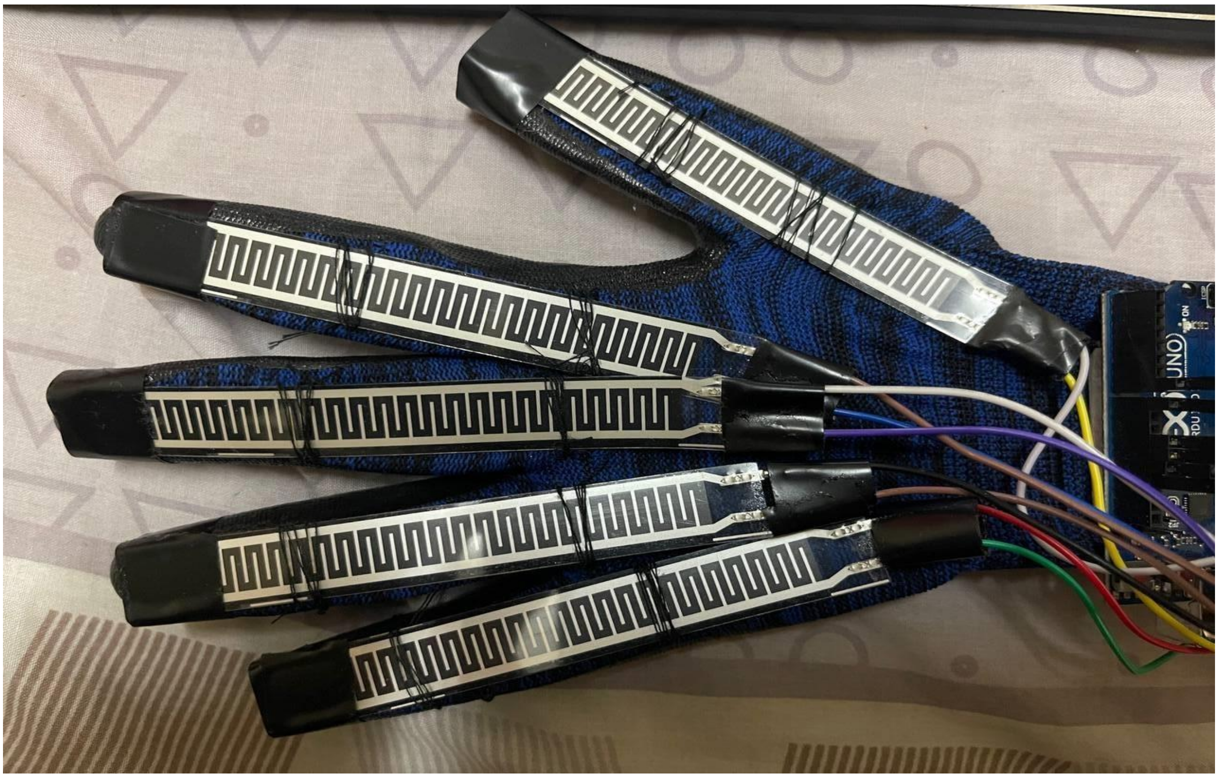


Figure 3.3.2 2 Shows the flex sensors at each of the finger

3.4 Electronic design process

Instead of using a wired system, the robotic hand was designed to operate wirelessly in remote locations without the need of an operator. For this purpose, an effective but simple system was designed through which these tasks were achieved. Arduino is used as microcontroller for controlling the robotic hand. The robotic hand contains all joints comparable as human hand which plays an imperative part in controlled movement of fingers that gives capacity to get a handle on hold of different shapes. The motion in robotic hand is performed with the help of servo motors. Five servos are used to control the robotic hand and one motor for each finger. Each servo motor provides an independent motion. Arduino uno (master) is interfaced with flex sensors and My transmitter transmitting module Arduino uno (master) takes information from the sensors and transmits wirelessly to the Arduino uno (slave). Arduino uno (slave) analyses the information and map the information on servo angles and hence control robotic hand.

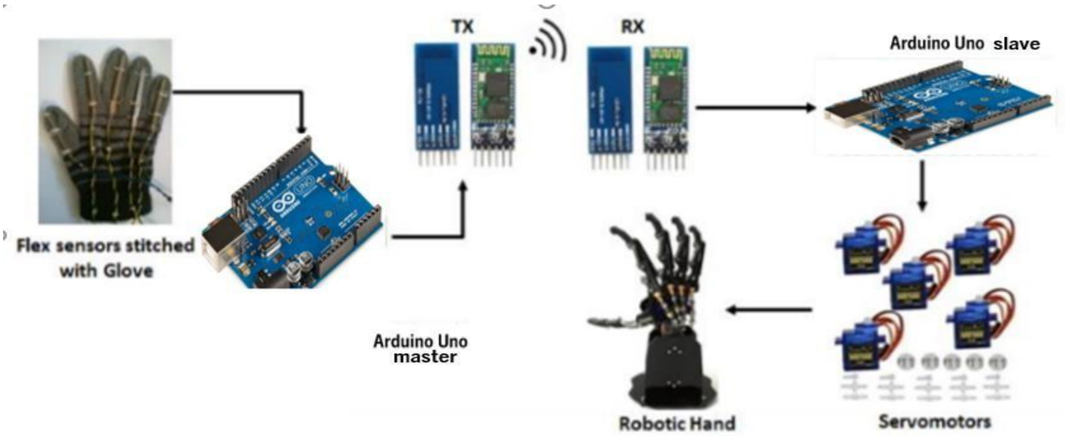


Figure 3.4 1 Shows the electronic system design of the project

Figure 3.4 2 Shows the electronic components fixed in the power box

3.5 Software development process

In order for the project to work, software must be developed in addition to the hardware. The Arduino microcontroller requires this software to conduct the right actions to create an output from the input it detects. C++ is used to create this software. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board. Active development of the Arduino software is hosted by

GitHub. See the instructions for building the code. Latest release source code archives are available here. The archives are PGP-signed so they can be verified using this gpg key.

[12]The Arduino code is written in C++, with a few extra methods and functions. C++ is a computer language that is easy to understand. A'sketch' (the term given to Arduino code files) is processed and compiled to machine language when you produce it. The main text editing application for Arduino programming is the Arduino Integrated Development Environment (IDE). It's here that we will write the code before uploading it to the board you're programming. Sketches are the name for Arduino code.

3.5.1 Arduino Programming

This part will show how to translate the robot hand programming flow chart to C++ utilising the debugger and a library that is already included in the Arduino software. The flow chart for the master glove hand programme is shown in Figure 3.4.1.1 The entire programme is included in the appendix.

The programme is manually controlled by the master glove. In order to make PWM, this software will receive data from an analogue input and convert it to a timer interrupt. If neither of the modes is selected, all of the fingers will return to their original position of straightness.

The transmitter programme and the receiver programme are the two sections of the programme. The transmitter software will constantly run on the microcontroller, and the data will be sent to the receiver programme. The robot hand will move after the receiver software has received the data from the transmitter programme.

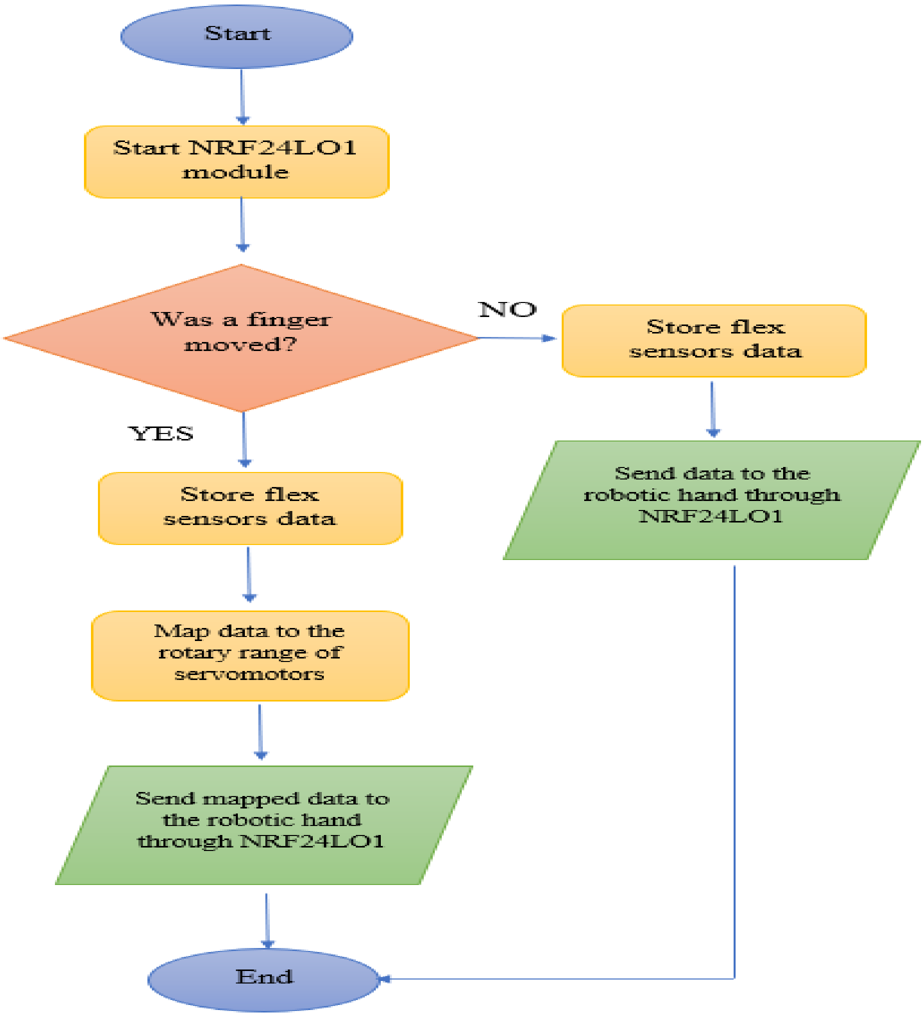


Figure 3.5.1 1 Shows the Master glove hand flowchart

3.5.2 NRF24L01 Module

NRF24L01 is a transceiver module.[13] It uses the 2.4 GHz frequency band and has baud rates ranging from 250 to 2 Mbps. If used in open space and at a lower baud rate, it has a range of up to 100 metres. This module uses just about 12mA during transmission, which is less than a single LED. The module's working voltage is 1.9 to 3.6V, but the other pins can accept 5V logic, thus we can connect it to an Arduino without using any logic level converters. In this example, two NRF24L01 modules are utilised. The master glove is used as a transmitter, while the other is employed as a receiver (slave robot hand). When the flex sensors began to move, the transmitter module collected the data and withheld it from the servomotor rotor.

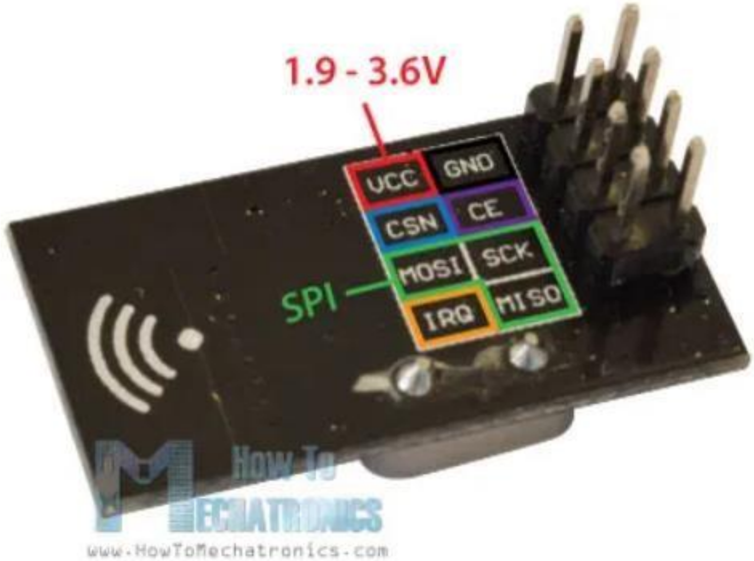


Figure 3.5.2 1 Shows the PINS that need to be connected

Three of these pins are used for [14]SPI communication and must be connected to the SPI pins on the Arduino. However, the SPI pins on each Arduino board change. The pins CSN and CE are used to switch between transmit and command mode and to put the module in standby or active mode. Any digital pin on the Arduino board may be used to connect them. The last pin is an interrupt pin that isn't used.

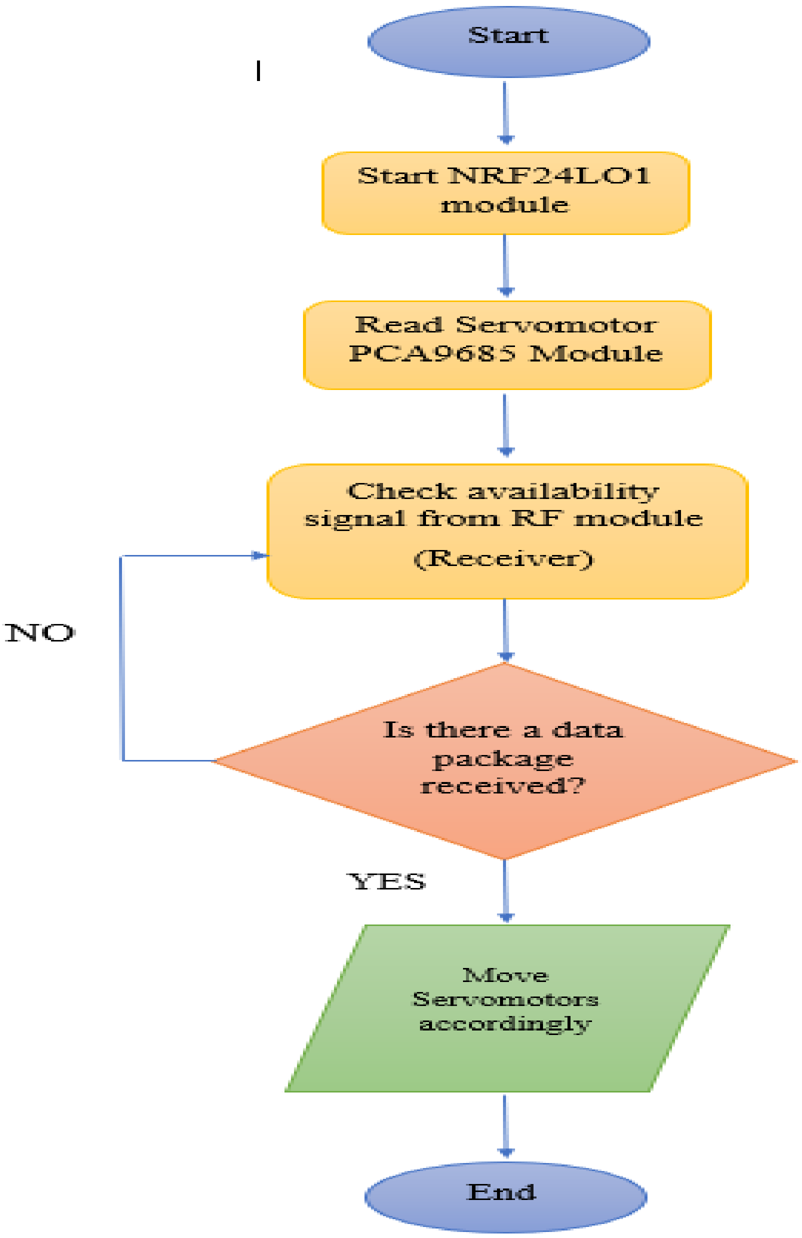


Figure 3.5.2 2 Shows the flowchart of robot hand

The open-source Arduino Software (IDE) software is used to programme the Arduino microcontroller, and it includes a serial class that allows you to output to the serial monitor, debugging comments, and variable values. This will be done with serial pins 0 and 1, which are linked to the USB port on most Arduino units. The C++ programming language is used to programme the Arduino microcontroller. This programme has the benefit of being able to build, debug, and optimise Arduino microcontroller-based applications.

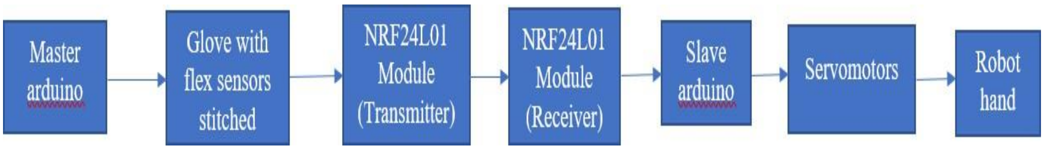


Figure 3.5.2 3 Shows the block diagram

3.6 Proteus design suite

Proteus is a simulation and electrical design development tool developed by Lab Center Electronics. It's a crucial tool since it ensures that the circuit design or firmware code is effective before you begin physically working on it. It has a wide collection of components that you may utilise to virtually build your circuit. Compiling and debugging your ideas is straightforward using Proteus' virtual metres (voltmeter and ammeter), oscilloscope, serial monitor, and other tools. I choose Proteus for Arduino project simulation because of its big library set. It is not limited to simulation, though; it may also be used to develop PCB designs. TinkerCAD is another tool that may be used for simulation.

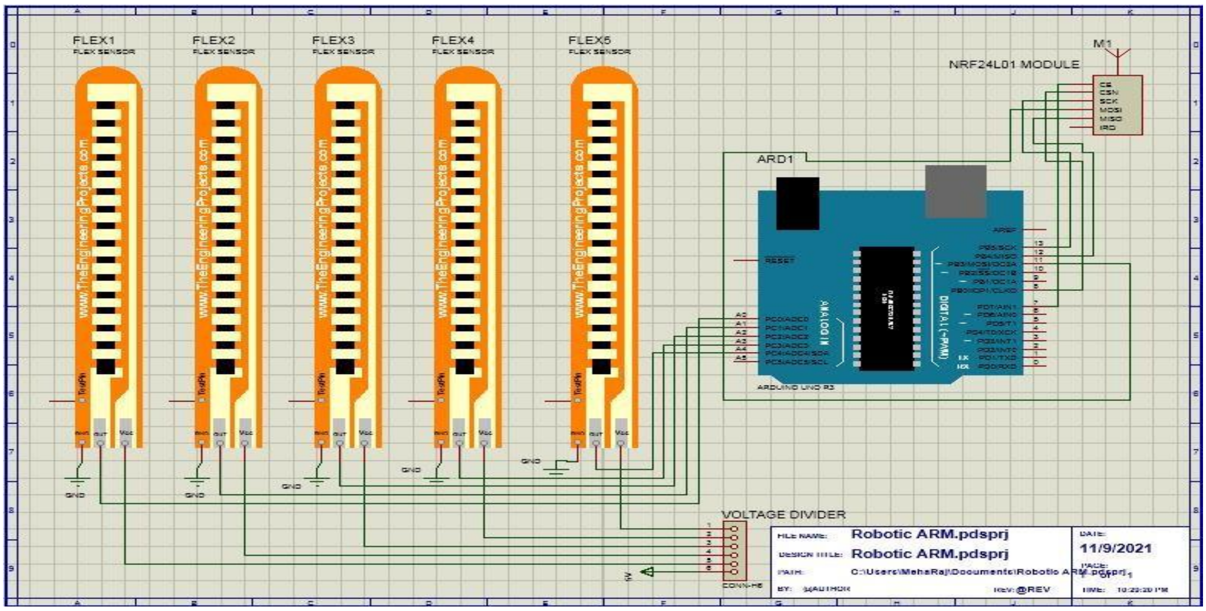


Figure 3.6 1 Shows the schematic drawing of Glove hand (Master) in proteus software

In begin, the flex sensors are connected with Arduino uno analogue pins along A0-A4 which means only 5 pins are used. Then the flex sensor VCC are connected to the voltage divider to make it imbalance with voltage supplies being used along with Arduino board. The flex sensors ground are connected to the GND. Furthermore, The NRF24L01 module (transmitter) has 5 major pins which is CS, CSN, SCK, MISO and MOSI are being used out of 6pins and it is connected to digital output of the Arduino uno board. So if these flex sensor starts to operate, the NRF24L01 module will received the data via Arduino uno board.

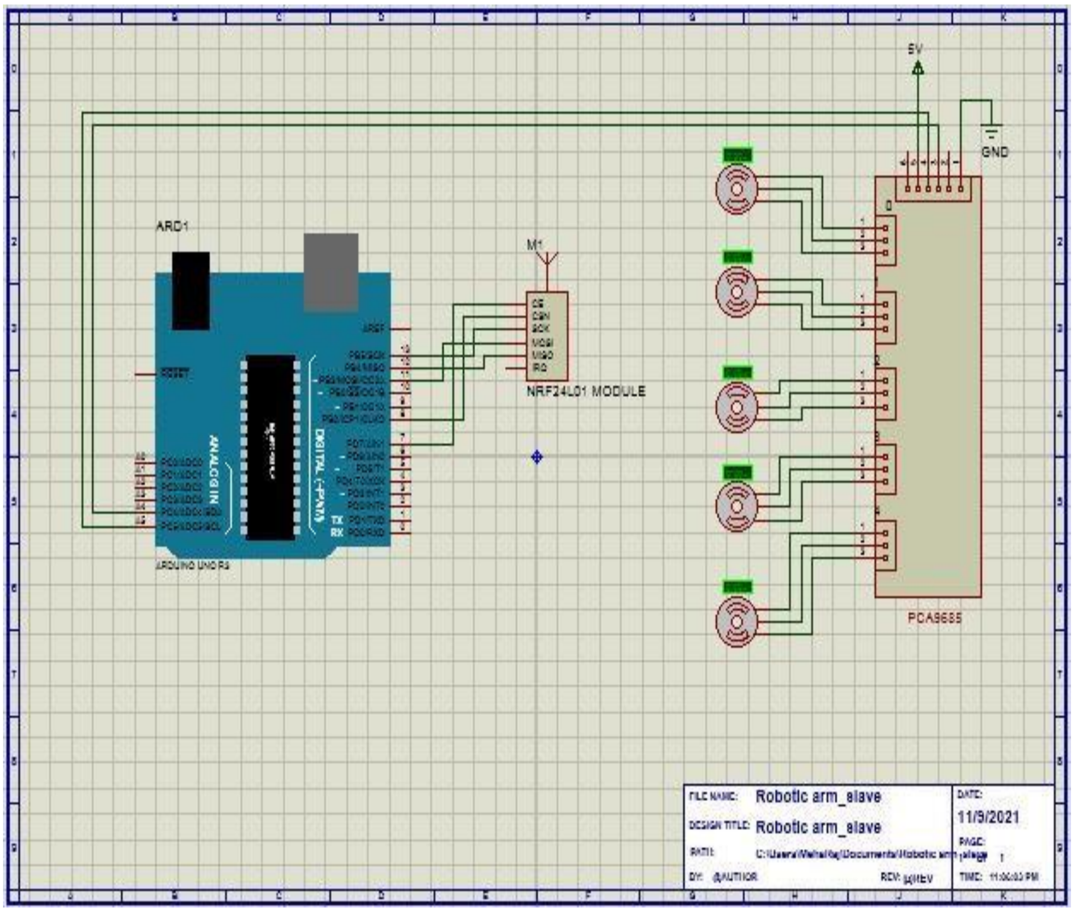


Figure 3.6 2 Shows the schematic drawing of Robot hand (slave) in Proteus software

In end, the PCA 9685 module has 5 slots with 3 pins each for the servomotor attachments and 1 slot with 6 pins. The analogue pins A4 and A5 is connected to the pin 3 and 4 in the PCA 9685 module and the pin 1 is for ground whereas the pin 4 for the 5V supply. The NRF24L01 module is connected to digital pin at Arduino uno board as well. The NRF24L01(Master) receiver module will get the mapped data from NRF24L01 (Slave) transmitter module and the PCA9685 module which holds 5 Servomotors for each finger start to move or rotate via Arduino uno board. Finally the schematic mechanism had been developed for both Master Glove and Slave Robot Hand.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

The conclusions of the project, which included analysis and experiments to evaluate the robot hand's functionality and performance, are discussed in this chapter. The final robot hand prototype is shown in Figure 4.1

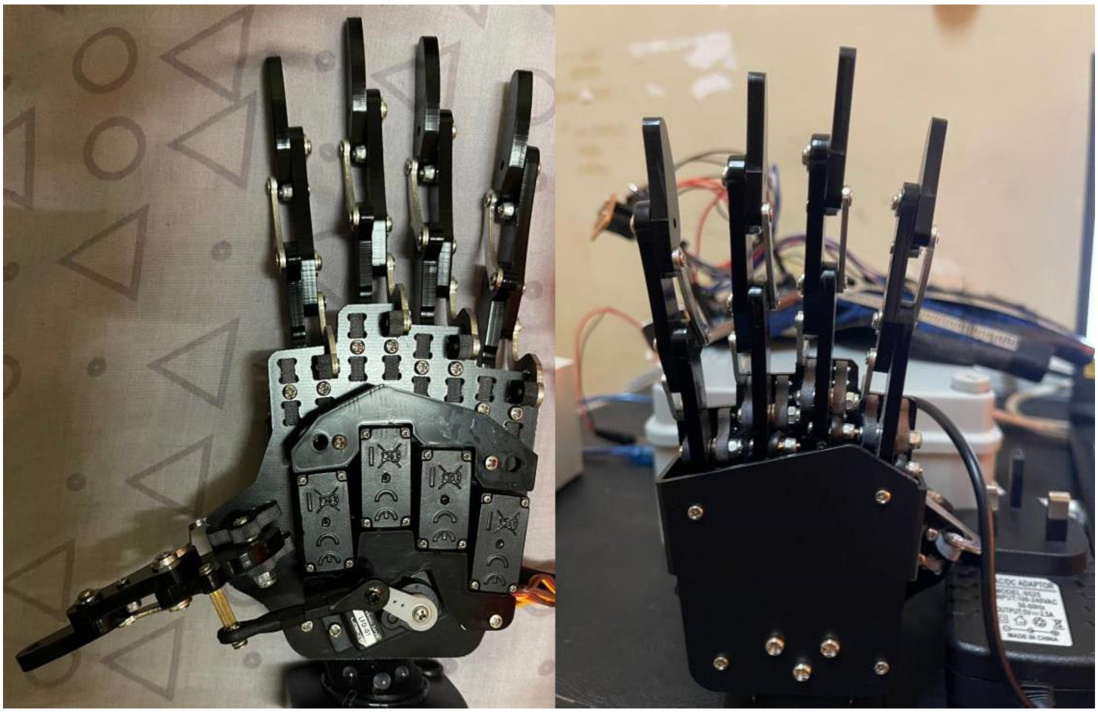


Figure 4 1 Shows the Prototype of the Robot Hand

4.2 Motor Torque and Speed

The speed of the servo motor was controlled by changes in the master glove's analogue input and was coupled to the voltage change of the potentiometer. On the other hand, the servo motor had two maximum speeds.[15] The maximum speed of the servomotor on this robot hand was 0.14 s/60° for this project because it was powered by a 5V battery, and the maximum speed of the fingers was also 0.14 s/60°. The shaft and finger lengths were measured, as well as a technical study of the robot hand's finger torque. The following formula can be used to predict torque:

Servomotor torque = 4.50 kg.cm

The Shaft's Radius is r = 1.20 cm.

Fs = 4.5/1.2

= 3.75 kg.cm/s2 is the force of the string.

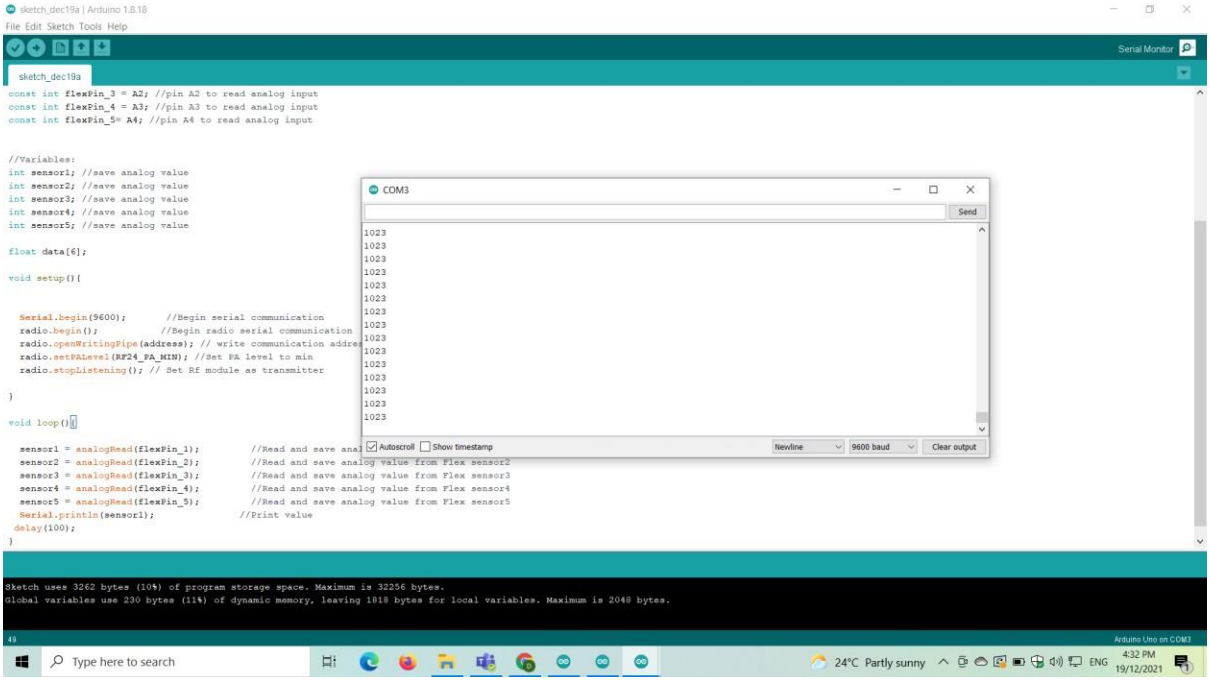
rj = 0.70 cm is the joint radius.

Fs x rj = 3.75 x 0.7 = 2.625 kg.cm Torque of the Fingers = Fs x rj

As a result, the torque was lowered to 2.625 kg.cm from 4.50 kg.cm.

4.3 Finger Force Analysis

The bending force of each finger is depend on the force exerts through the pressure of flex sensor that used in the master glove. So once we bend the our finger, the flex sensor will bend. So in this case, I have set the value for the opening and closing force of each finger in my programme. When the finger is open it will be at 1024 byte and if it close it will reduce or the value will fluctuate according to the flex sensor pressure.



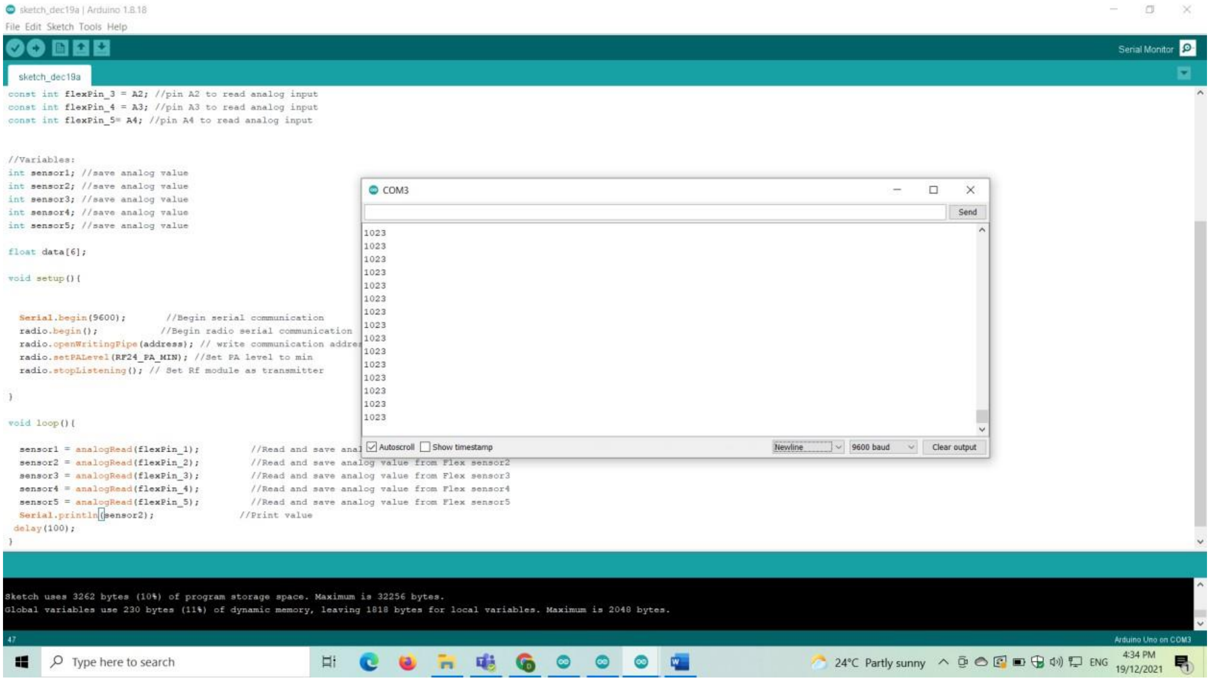
```
const int flexPin_3 = A2; //pin A2 to read analog input
const int flexPin_4 = A3; //pin A3 to read analog input
const int flexPin_5= A4; //pin A4 to read analog input


//Variables:
int sensor1; //save analog value
int sensor2; //save analog value
int sensor3; //save analog value
int sensor4; //save analog value
int sensor5; //save analog value

float data[6];

void setup(){


  Serial.begin(9600);        //Begin serial communication
  radio.begin();           //Begin radio serial communication
  radio.openWritingPipe(address); // write communication addres
  radio.setPALevel(RF24_PA_MIN); //Set PA level to min
  radio.stopListening(); // Set Rf module as transmitter

}

void loop(){

  sensor1 = analogRead(flexPin_1);         //Read and save anal
  sensor2 = analogRead(flexPin_2);         //Read and save analog value from Flex sensor2
  sensor3 = analogRead(flexPin_3);         //Read and save analog value from Flex sensor3
  sensor4 = analogRead(flexPin_4);         //Read and save analog value from Flex sensor4
  sensor5 = analogRead(flexPin_5);         //Read and save analog value from Flex sensor5
  Serial.println(sensor1);               //Print value
 delay(100);
}
```

```
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
```

```
Sketch uses 3262 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 230 bytes (11%) of dynamic memory, leaving 1818 bytes for local variables. Maximum is 2048 bytes.
```

```
49
```

Finger 1



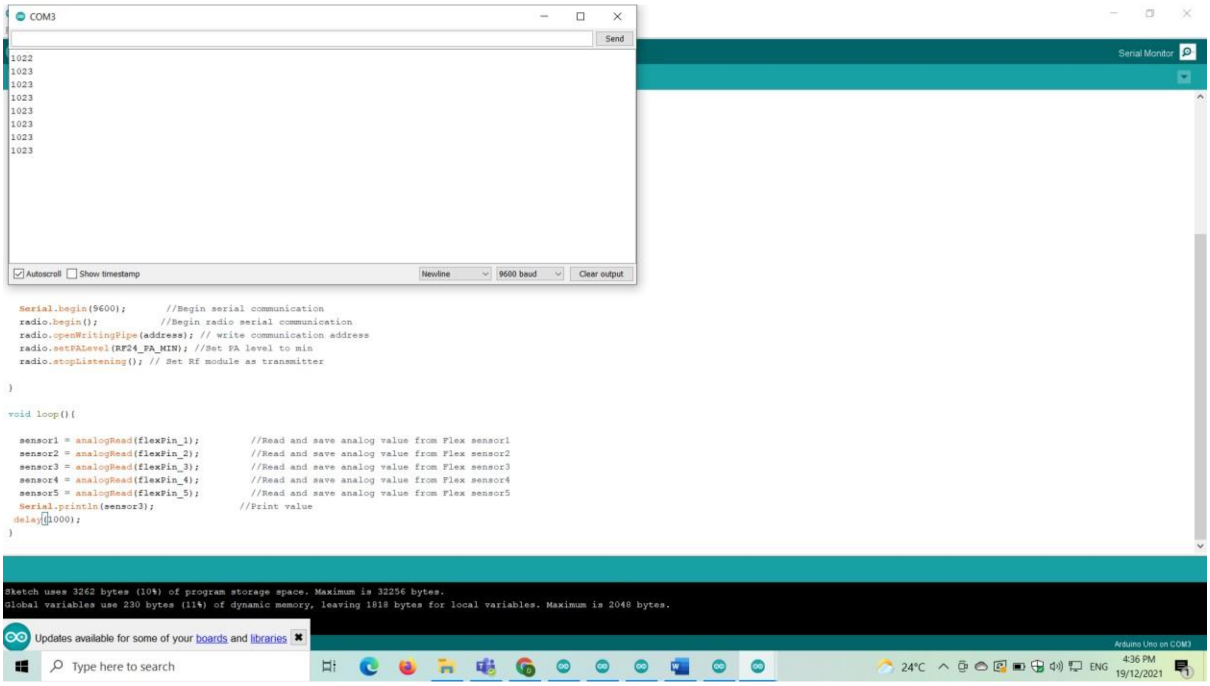
```
const int flexPin_3 = A2; //pin A2 to read analog input
const int flexPin_4 = A3; //pin A3 to read analog input
const int flexPin_5= A4; //pin A4 to read analog input


//Variables:
int sensor1; //save analog value
int sensor2; //save analog value
int sensor3; //save analog value
int sensor4; //save analog value
int sensor5; //save analog value

float data[6];

void setup(){


  Serial.begin(9600);        //Begin serial communication
  radio.begin();           //Begin radio serial communication
  radio.openWritingPipe(address); // write communication addres
  radio.setPALevel(RF24_PA_MIN); //Set PA level to min
  radio.stopListening(); // Set Rf module as transmitter

}

void loop(){

  sensor1 = analogRead(flexPin_1);         //Read and save anal
  sensor2 = analogRead(flexPin_2);         //Read and save analog value from Flex sensor2
  sensor3 = analogRead(flexPin_3);         //Read and save analog value from Flex sensor3
  sensor4 = analogRead(flexPin_4);         //Read and save analog value from Flex sensor4
  sensor5 = analogRead(flexPin_5);         //Read and save analog value from Flex sensor5
  Serial.println(sensor2);               //Print value
 delay(100);
}
```

```
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
```

```
Sketch uses 3262 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 230 bytes (11%) of dynamic memory, leaving 1818 bytes for local variables. Maximum is 2048 bytes.
```

```
47
```

Finger 2

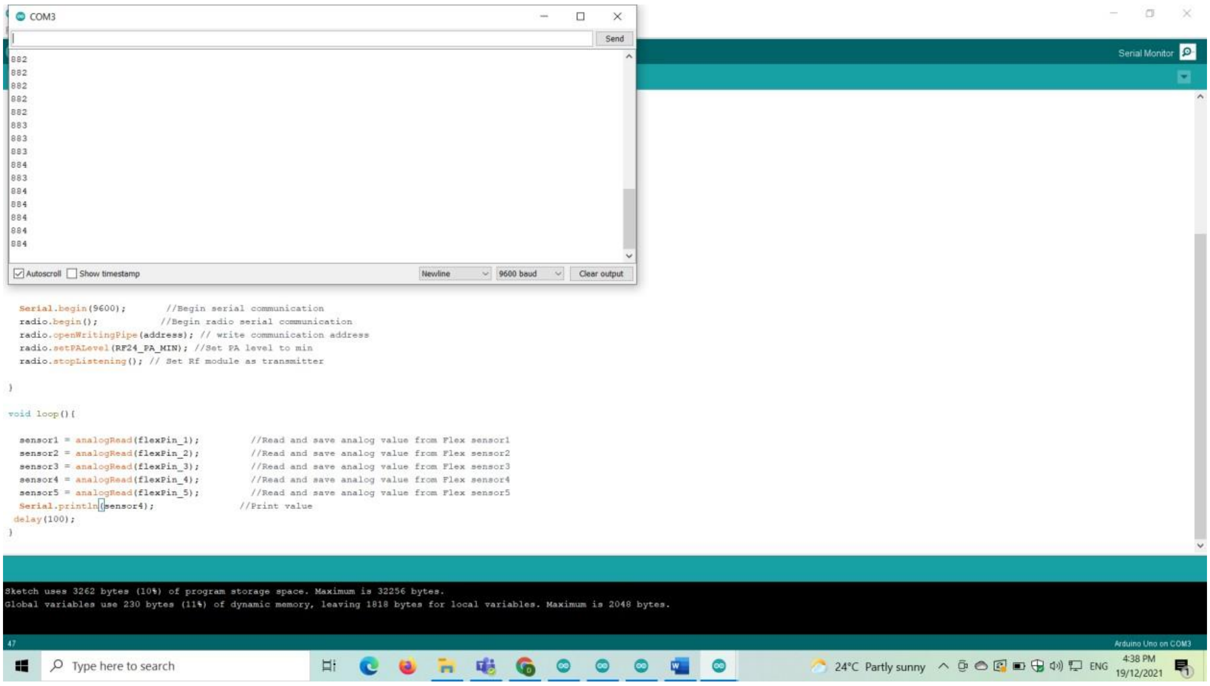


```
1022
1023
1023
1023
1023
1023
1023
1023
```

```
  Serial.begin(9600);        //Begin serial communication
  radio.begin();            //Begin radio serial communication
  radio.openWritingPipe(address); // write communication address
  radio.setPALevel(RF24_PA_MIN); //Set PA level to min
  radio.stopListening(); // Set Rf module as transmitter

}

void loop(){

  sensor1 = analogRead(flexPin_1);         //Read and save analog value from Flex sensor1
  sensor2 = analogRead(flexPin_2);         //Read and save analog value from Flex sensor2
  sensor3 = analogRead(flexPin_3);         //Read and save analog value from Flex sensor3
  sensor4 = analogRead(flexPin_4);         //Read and save analog value from Flex sensor4
  sensor5 = analogRead(flexPin_5);         //Read and save analog value from Flex sensor5
  Serial.println(sensor3);               //Print value
 delay(1000);
}
```

```
Sketch uses 3262 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 230 bytes (11%) of dynamic memory, leaving 1818 bytes for local variables. Maximum is 2048 bytes.
```

Finger 3

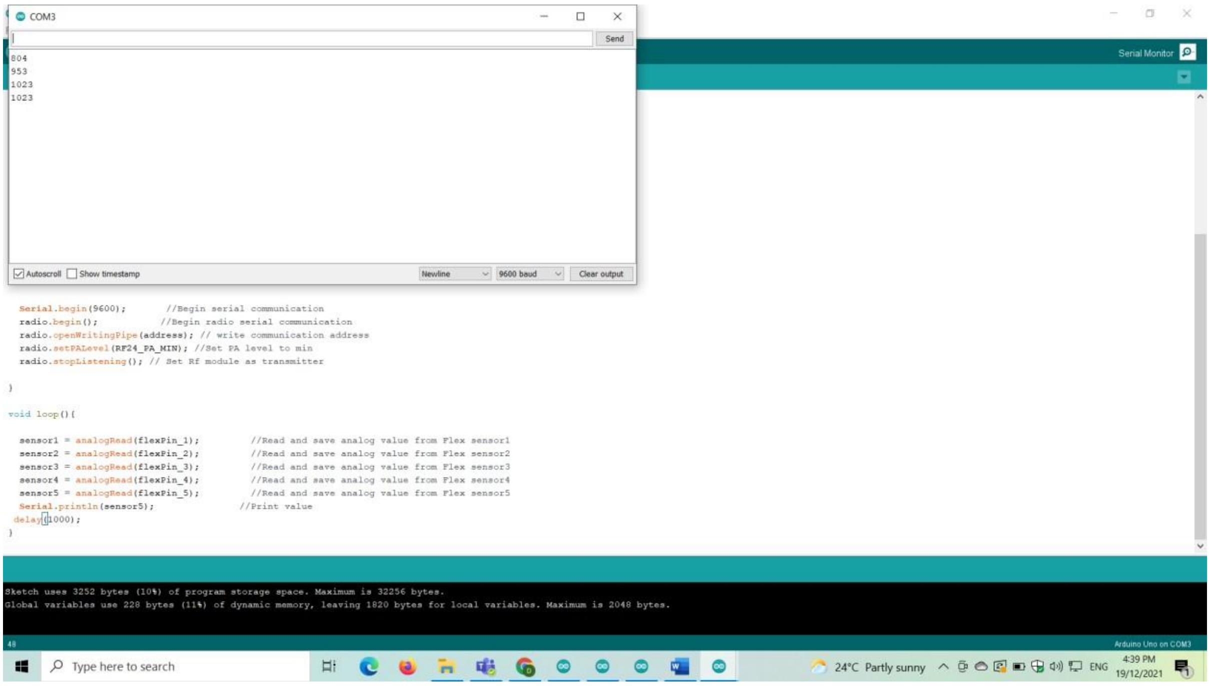


```
882
882
882
882
882
883
883
883
884
883
884
884
884
884
884
```

```
  Serial.begin(9600);        //Begin serial communication
  radio.begin();            //Begin radio serial communication
  radio.openWritingPipe(address); // write communication address
  radio.setPALevel(RF24_PA_MIN); //Set PA level to min
  radio.stopListening(); // Set Rf module as transmitter

}

void loop(){

  sensor1 = analogRead(flexPin_1);         //Read and save analog value from Flex sensor1
  sensor2 = analogRead(flexPin_2);         //Read and save analog value from Flex sensor2
  sensor3 = analogRead(flexPin_3);         //Read and save analog value from Flex sensor3
  sensor4 = analogRead(flexPin_4);         //Read and save analog value from Flex sensor4
  sensor5 = analogRead(flexPin_5);         //Read and save analog value from Flex sensor5
  Serial.println(sensor4);               //Print value
 delay(100);
}
```

```
Sketch uses 3262 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 230 bytes (11%) of dynamic memory, leaving 1818 bytes for local variables. Maximum is 2048 bytes.
```

Finger 4



```
804
953
1023
1023
```

```
  Serial.begin(9600);        //Begin serial communication
  radio.begin();            //Begin radio serial communication
  radio.openWritingPipe(address); // write communication address
  radio.setPALevel(RF24_PA_MIN); //Set PA level to min
  radio.stopListening(); // Set Rf module as transmitter

}

void loop(){

  sensor1 = analogRead(flexPin_1);         //Read and save analog value from Flex sensor1
  sensor2 = analogRead(flexPin_2);         //Read and save analog value from Flex sensor2
  sensor3 = analogRead(flexPin_3);         //Read and save analog value from Flex sensor3
  sensor4 = analogRead(flexPin_4);         //Read and save analog value from Flex sensor4
  sensor5 = analogRead(flexPin_5);         //Read and save analog value from Flex sensor5
  Serial.println(sensor5);               //Print value
 delay(1000);
}
```

```
Sketch uses 3252 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 228 bytes (11%) of dynamic memory, leaving 1820 bytes for local variables. Maximum is 2048 bytes.
```

Finger 5

Figure 4 3 Shows the bending result of each finger

As the result shown above, we can conclude that the force is fluctuating at each finger. This is because of the flex sensor pressure which couldn’t able to bend firmly when I bend the finger. At finger 1, I set the delay at 0.3 second for finger 1 bending and the reading shown from 800 to 1023. At finger 2, I set the delay at 0.5 second for finger 2 bending and the reading shown was 800 to 1023 as well. At finger 3, there was a 1 second delay and its able to capture the reading abit late at the serial monitor which is 1023. At finger 4, we can see the result of closing of the finger too late this is because of the delay I have set which is 3 second to 5 second. The result shown for finger 4 is from 882 to 884 and it will continue to 1023 at the end when we the finger in open position. At finger 5, the delay I have set is 8 to 10 second and we can the both closing and opening of the finger result in the same timing in the serial monitor. The result shown is from 804 to 1023.

4.4 Master Glove Controller

The master glove controller directs the movement of the robot hand. The master glove is equipped with 5 flex sensors that provide various output voltages to the Arduino microcontroller. The output is then used to compute the position of a human hand's fingers. The process of converting analogue input to PWM output took place in the Arduino microcontroller. This method allows the robot hand to move in real time like a human hand.

The master glove can control the robot hand to accomplish a few tasks. The robot hand is capable of emulating a human hand and gripping different type materials like as a irregular shape of material and regular shape material as well. The master glove controller is shown in figure 4.4(a), and figure 4.4(b) demonstrates how the master glove controller controls the robot hand.

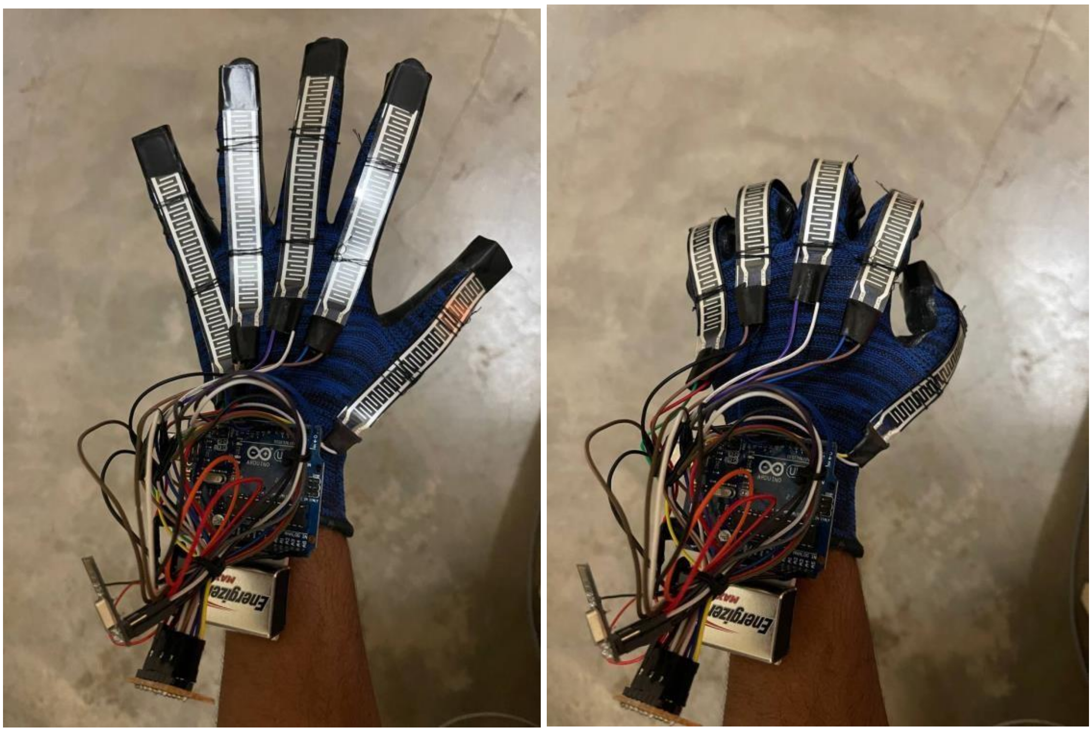
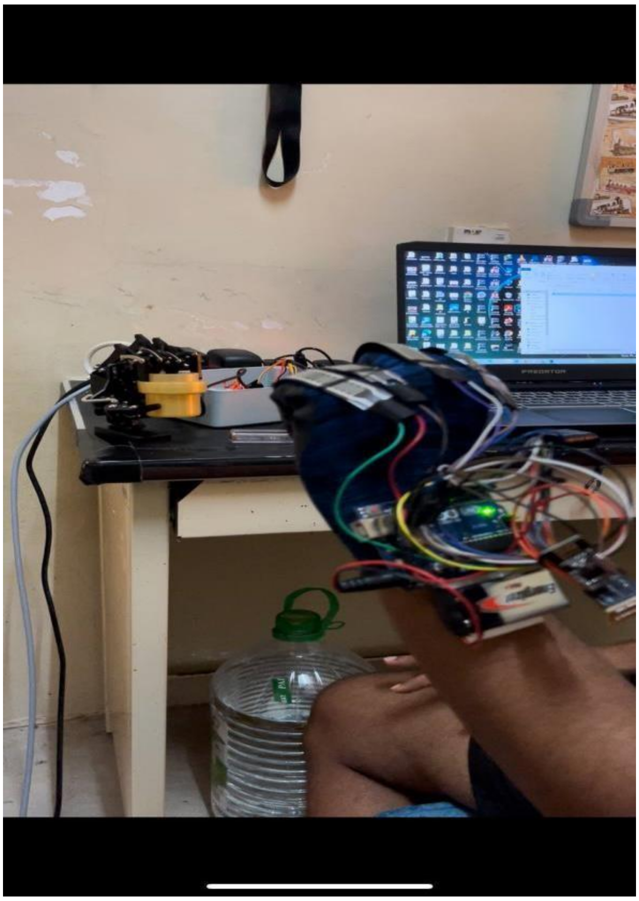
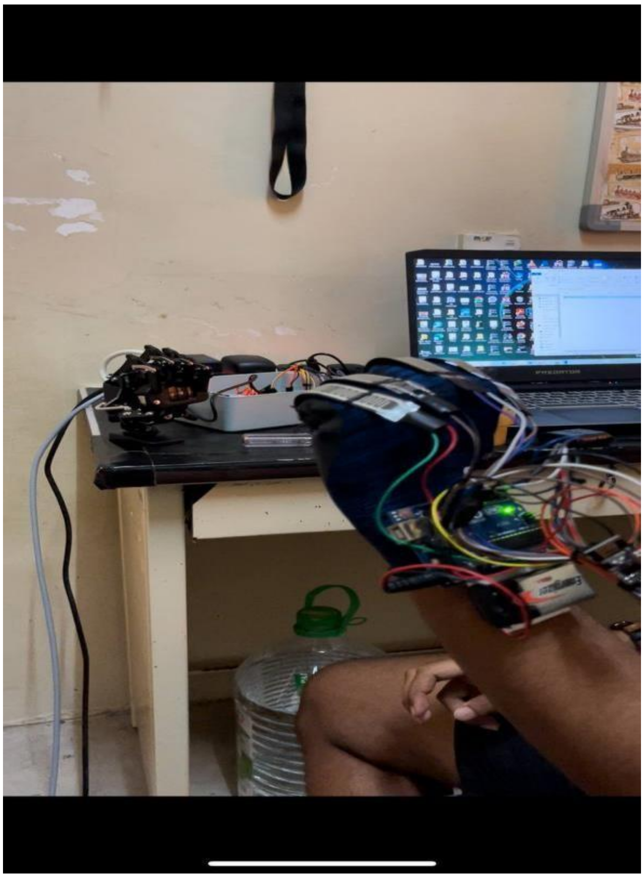


Figure 4.1 1 Shows the Master Glove Controller

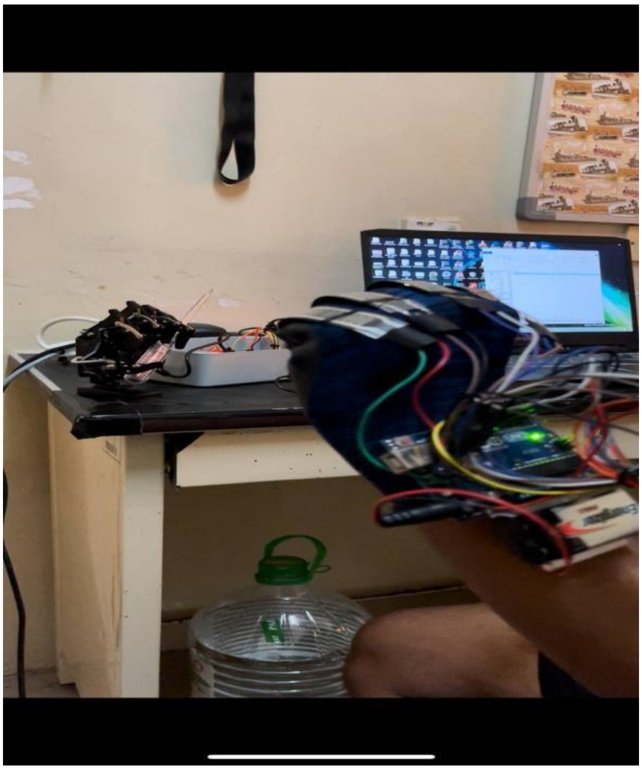


Grasping Circular shaped Object



Grasping a square shaped object

Grasping irregular shaped of object



Grasping a long shaped object

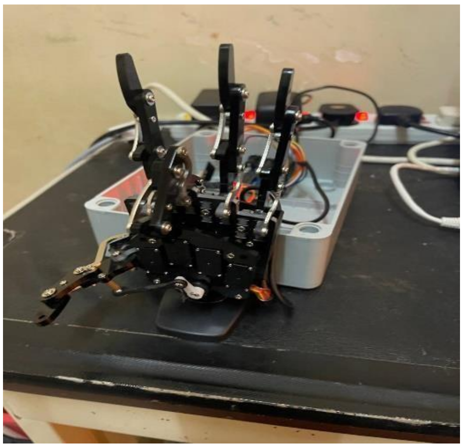
Figure 4.1 2 Shows Robot Hand Manually controlled by glove and grasping objects

4.5 Finger Movement

The robot hand will move follow a sequence of instruction given by the Master glove. So the robot hand will mimic the gesture of master glove and perform some actions which is bending finger and closing all the fingers. The figure 4.5 shows the movement of the robot hand.

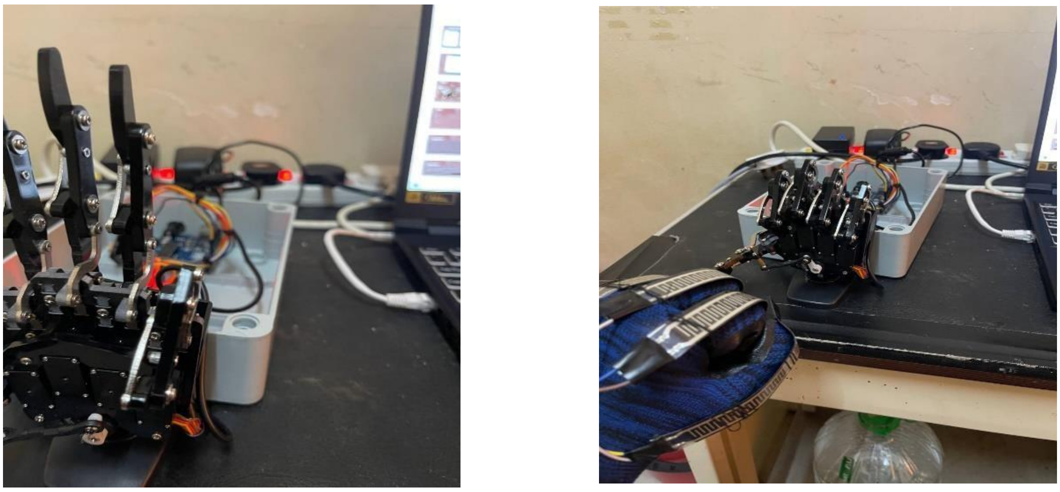
Thumb Bended

Index Finger Bended



Middle Finger Bended

Ring Finger Bended



Pinky Finger Bended Bending all fingers

Figure 4.5 1 shows finger movement of robot hand

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The project's goals had been accomplished. The robot hand was capable of grasping things and was designed to look like a human hand. It may also be controlled in real time by the master glove. To summarize, the robot hand platform has a huge impact on humanoid robotics and invalidity. The notion of a humanoid robot hand may be used in a variety of applications and research domains. NASA's Robonaut Program, Honda's Asimo Humanoid Robot, and military research are all examples of humanoid robot research. As a result, the robot hand project is a timely project with a wide range of potential applications in the fields of economics, scientific research, and design. This project's expertise and skills will provide a great deal of value and opportunity.

5.2 Recommendation

There was still a lot of potential for improvement and refining in this Robotic Hand Controlled by Glove Using Wireless Communication project. Humanoid robots require a lot of imagination, talent, and a dynamic mindset to fully use the technology, knowledge, and inspiration of nature. The master glove was constructed mechanically, with potentiometers serving as all of the sensors. The mechanical design made it difficult to move and wear, and each finger had just three degrees of flexibility. This might be enhanced by including a Flexible Bend Sensor into the master glove, which would make it more supple, comfortable to wear, and easy to use.

The servo motors used in this project were normal size, and there were five of them in the forearm. The size of the human forearm was set by the servo motors, but the torque was insufficient to sustain a heavy weight on the finger. A stronger air muscle should be applied with a metallic string to increase the robot hand's finger movement and motility. In addition, the finger on the robot hand has three degrees of freedom, but our human hand has four. One degree of freedom must be added to the robot hand to boost its agility and make it perfectly resemble a human hand.

REFERENCE

[1] I. Llop-Harillo, A. Pérez-González, J. Starke, and T. Asfour, “The Anthropomorphic Hand Assessment Protocol (AHAP),” *Rob. Auton. Syst.*, vol. 121, p. 103259, 2019, doi: 10.1016/j.robot.2019.103259.

[2] G. Gao, A. Dwivedi, N. Elangovan, Y. Cao, L. Young, and M. Liarokapis, “The New Dexterity Adaptive, Humanlike Robot Hand,” *IEEE Int. Conf. Robot. Autom.*, vol. 7, no. 2, p. 2046, 2019.

[3] “E lectroactive,” vol. 33, no. March 2008, pp. 173–181, 2016.

[4] A. Yudhana, J. Rahmawan, and C. U. P. Negara, “Flex sensors and MPU6050 sensors responses on smart glove for sign language translation,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 403, no. 1, 2018, doi: 10.1088/1757-899X/403/1/012032.

[5] Circuit Basics, “Basics of UART Communication,” *Electron. Hub*, 2017, [Online]. Available: https://www.electronicshub.org/basics-uart-communication/.

[6] J. Maw, K. Y. Wong, and P. Gillespie, “Hand anatomy,” *Br. J. Hosp. Med.*, vol. 77, no. 3, pp. C34–C40, 2016, doi: 10.12968/hmed.2016.77.3.C34.

[7] J. E. Parada Puig, N. E. N. Rodriguez, and M. Ceccarelli, “A methodology for the design of robotic hands with multiple fingers,” *Int. J. Adv. Robot. Syst.*, vol. 5, no. 2, pp. 177–184, 2008, doi: 10.5772/5600.

[8] E. A. Khalid, W. T. Abbood, and O. I. Abdullah, “Real-time control of robotic hand by human hand at low cost,” *J. Mech. Eng. Res. Dev.*, vol. 43, no. 3, pp. 455–467, 2020.

[9] E. N. Gama Melo, O. F. Avilés Sánchez, and D. Amaya Hurtado, “Anthropomorphic robotic hands: a review,” *Ing. Y Desarro.*, vol. 32, no. 2, pp. 279–313, 2014, doi: 10.14482/inde.32.2.4715.

[10] I. Yamano and T. Maeno, “Five-fingered robot hand using ultrasonic motors and elastic elements,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2005, no. April, pp. 2673–2678, 2005, doi: 10.1109/ROBOT.2005.1570517.

[11] H. Hu, J. Li, Z. Xie, B. Wang, H. Liu, and G. Hirzinger, “A robot arm/hand teleoperation system with telepresence and shared control,” *IEEE/ASME Int. Conf.*

Adv. Intell. Mechatronics, AIM, vol. 2, no. August, pp. 1312–1317, 2005, doi: 10.1109/aim.2005.1511192.

[12] R. D. D. Basics, *Programming With Arduino*. 2017.

[13] N. Simicondutor, “Preliminary Product Specification v1.0,” no. March, pp. 1–75, 2008, [Online]. Available: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf.

[14] A. Smith and J. Wu, “Basics of SPI : Serial Communications,” pp. 1–29.

[15] M. Hekim and Z. Dogan, “The Speed and Torque Control of Direct Current Servo Motors by using Cascade Fuzzy PI Controller Przeglad Elektrotechniczny (Electrical Review) Abstracted and Indexed : Science Citation Index . Starting from 1 January 2008 Przeglad Elektrotchniczny is indexed by Thomson Scientific Web of Science Philadephia (Philadelphia list).,” no. January, 2012.

APPENDIX

Source Code for Master Glove

```
///Library//////

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001"; ///Trasmitter address

const int ledPin = 3;  //pin 3 has PWM funtion

const int flexPin_1 = A0; //pin A0 to read analog input

const int flexPin_2 = A1; //pin A1 to read analog input

const int flexPin_3 = A2; //pin A2 to read analog input

const int flexPin_4 = A3; //pin A3 to read analog input

const int flexPin_5= A4; //pin A4 to read analog input

//Variables:

int sensor1; //save analog value

int sensor2; //save analog value

int sensor3; //save analog value

int sensor4; //save analog value

int sensor5; //save analog value

float data[6];

void setup(){

  Serial.begin(9600);       //Begin serial communication

  radio.begin();           //Begin radio serial communication
```

```
  radio.openWritingPipe(address); // write communication address
  radio.setPALevel(RF24_PA_MIN); //Set PA level to min
  radio.stopListening(); // Set Rf module as transmitter
}
void loop(){
  sensor1 = analogRead(flexPin_1); //Read and save analog value from Flex sensor1
  sensor2 = analogRead(flexPin_2); //Read and save analog value from Flex sensor2
  sensor3 = analogRead(flexPin_3); //Read and save analog value from Flex sensor3
  sensor4 = analogRead(flexPin_4); //Read and save analog value from Flex sensor4
  sensor5 = analogRead(flexPin_5);  //Read and save analog value from Flex sensor5
  //Serial.println(value);               //Print value
  delay(100);
//////////Finger1//////////////////////////////////////
  if(sensor1<=900){
    //Serial.println("90");
    data[0] = 434; // code to identify the transmitter.
    data[1] = 50;  //Angle for close position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
  if(sensor1>=1000){
    //Serial.println("90");
    data[0] = 434; // code to identify the transmitter.
    data[1] = 90;  //Angle for open position
```

```
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
////////////////////////////////////////////////////////
////////////Finger2///////////////////////////////////
  if(sensor2<=1000){
    //Serial.println("120");
    data[0] = 231; // code to identify the transmitter.
    data[2] = 150; //Angle for close position
    radio.write(&data, sizeof(data));//Transmit the angle data to receiver
    delay(300); //delay
  }
  if(sensor2>=1000){
   // Serial.println("20");
    data[0] = 231; // code to identify the transmitter.
    data[2] = 20; //Angle for open position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
///////////////////////////////////////////////////////
/////////Finger3//////////////////////////////////////
  if(sensor3<=1000){
    //Serial.println("120");
    data[0] = 346; // code to identify the transmitter.
```

```
    data[3] = 150; //Angle for close position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
  if(sensor3>=1000){
   // Serial.println("20");
    data[0] = 346; // code to identify the transmitter.
    data[3] = 20;  //Angle for open position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
//////////////////////////////////////////////////////
/////////Finger4/////////////////////////////////////
  if(sensor4<=1000){
    //Serial.println("120");
    data[0] = 561; // code to identify the transmitter.
    data[4] = 150;  //Angle for close position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
  if(sensor4>=1000){
   // Serial.println("20");
    data[0] = 561; // code to identify the transmitter.
    data[4] = 20;  //Angle for open position
```

```
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300);//delay
  }
//////////////////////////////////////////////////////
///////Finger5//////////////////////////////////////
  if(sensor5<=1000){
    //Serial.println("120");
    data[0] = 782; // code to identify the transmitter.
    data[5] = 150; //Angle for close position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
  if(sensor5>=1000){
   // Serial.println("20");
    data[0] = 782; // code to identify the transmitter.
    data[5] = 20;  //Angle for open position
    radio.write(&data, sizeof(data)); //Transmit the angle data to receiver
    delay(300); //delay
  }
/////////////////////////////////////////////////////////
}
```

Source Code For Slave Robot Hand

```
///LibraryRF2.4//////
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
///LibraryDriver//////
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
#define MIN_PULSE_WIDTH       650
#define MAX_PULSE_WIDTH       2350
#define DEFAULT_PULSE_WIDTH  1500
#define FREQUENCY             50
RF24 radio(7, 8); // CE, CSN
const byte address[6] = "00001";
float data[6]; // depending on the number of sensors used
float data1;
float data2;
float data3;
float data4;
float data5;
void setup() {
  Serial.begin(9600);
  pwm.begin();
  pwm.setPWMFreq(FREQUENCY);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
```

```
}
int pulseWidth(int angle)
{
  int pulse_wide, analog_value;
  pulse_wide = map(angle, 0, 180, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);
  analog_value = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096);
 // Serial.println(analog_value);
  return analog_value;
}
void loop() {
  if (radio.available()) {
    if (radio.available())
{
while (radio.available())              /* Loop until receiving valid data*/
{

radio.read(&data, sizeof(data));/* Read the received data and store in ' rx_data ' */
Serial.print("Received Data1 : ");
Serial.println(data1);          /* Print received value on Serial Monitor */
Serial.print("Received Data2 : ");
Serial.println(data2);          /* Print received value on Serial Monitor */
Serial.print("Received Data3 : ");
Serial.println(data3);          /* Print received value on Serial Monitor */
Serial.print("Received Data4 : ");
Serial.println(data4);          /* Print received value on Serial Monitor */
Serial.print("Received Data5 : ");
Serial.println(data5);          /* Print received value on Serial Monitor */
//Serial.println(data[4]);          /* Print received value on Serial Monitor */
```

```
if (data[0] == 434) // header
{
  data1 = data[1];
}
if (data[0] == 231) // header
{
  data2 = data[2];
}
if (data[0] == 346) // header
{
  data3 = data[3];
}
if (data[0] == 561) // header
{
  data4 = data[4];
}
if (data[0] == 782) // header
{
  data5 = data[5];
}
pwm.setPWM(0, 0, pulseWidth(data1));
pwm.setPWM(1, 0, pulseWidth(data2));
pwm.setPWM(2, 0, pulseWidth(data3));
pwm.setPWM(3, 0, pulseWidth(data4));
pwm.setPWM(4, 0, pulseWidth(data5));
}
}
else
{
```

```
Serial.println("Not Receiving !!!"); /* If not receiving valid data print " Not Receiving !!! " on Serial Monitor  */

  }

  }

}
```