TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Project Report

On

Virtual Hand Simulation Of Hand Gesture Using Hall Effect Sensor

Submitted By:

Aayush Pathak (33552)

Bal Krishna Shah (33558)

Sabin Acharya (33583)

Safal Karki (33584)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Project Report

On

Virtual Hand Simulation Of Hand Gesture Using Hall Effect Sensor

Submitted By:

Aayush Pathak (THA077BEI002)

Bal Krishna Shah (THA077BEI010)

Sabin Acharya (THA077BEI035)

Safal Karki (THA077BEI036)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics Communication and Information Engineering.

Under the Supervision of:

Er. Saroj Shakya

March, 2024

DECLARATION

We hereby declare that the report of the project entitled **Virtual Hand Simulation Of Hand Gesture Using Hall Effect Sensor** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Electronics, Communication and Information Engineering, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Aayush Pathak (THA077BEI002) _____

Bal Krishna Shah (THA077BEI010) _____

Sabin Acharya (THA077BEI035) _____

Safal Karki (THA077BEI036) _____

Date: March, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled **Virtual Hand Simulation Of Hand Gesture Using Hall Effect Sensor** submitted by **Aayush Pathak, Bal Krishna Shah, Sabin Acharya** and **Safal Karki** in partial fulfillment for the award of Bachelor's Degree in Electronics and Communication Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics and Communication Engineering.

_____

Project Supervisor
Er. Saroj Shakya
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner
Er. Shankar Gangaju
Civil Aviation Authority of Nepal, Babar Mahal, Kathmandu

_____

Project Co-Ordinator
Mr. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Mr. Kiran Chandra Dahal
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We are grateful towards the Department of Electronics and Computer Engineering of Thapathali Engineering Campus for scheduling this project in an efficient manner. Our sincere thanks also go to all the faculty members of the department for providing us useful information required for this project.

We are highly indebted to our supervisor **Er. Saroj Shakya** for his constant guidance and support throughout this project.

Last but not the least, we would like to thank our friends, families, seniors and juniors for their kind support and encouragement.

Aayush Pathak (THA077BEI002)

Bal Krishna Shah (THA077BEI010)

Sabin Acharya (THA077BEI035)

Safal Karki (THA077BEI036)

ABSTRACT

Virtual simulation imitates real-world entities in a virtual environment. For an immersive experience, the simulation needs to be in real-time, i.e. the real-world entity needs to be rendered as soon as the entity changes its state. In order to achieve this immersive experience, this project consists of a sensorized glove that uses a combination of hall effect sensors and magnets to detect the amount of bend at each joint of finger phalanges and a gyroscope/accelerometer sensor to determine the orientation of the hand. With the help of this data, a virtual hand designed in blender is rendered in an Unity 3D application and imitate the gestures of the hand wearing the gloves in real time.

Keywords: Hall Effect Sensor, Human-Computer Interaction, Unity3D, Virtual Reality, Wearable Design, WebSocket

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABD	Abduction
ADC	Analog to Digital Converter
ADD	Adduction
API	Application Programming Interface
AR	Augmented Reality
DIP	Distal Interphalangeal
DHCP	Dynamic Host Configuration Protocol
DOF	Degree of Freedom
HTTP	Hyper Text Transfer Protocol
I2C	Inter-Integrated Circuit
IOT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
MCP	Metacarpophalangeal
MEMS	Mico-Electromechanical System
MSB	Most Significant Bit
PIP	Proximal Interphalangeal
PLA	Polylactic Acid
SAR	Successive Approximation Resistor
SCL	Serial Clock Line
SDA	Serial Data Line
SSID	Service Set Identifier
UDP	User Datagram Protocol
VR	Virtual Reality
WI-FI	Wireless Fidelity

1. INTRODUCTION

With the use of hand gesture recognition technologies, computers are now able to recognize and comprehend finger movements. These movements can be tracked using various sensors. The perceptual data of hand movement can be used to create a real-time simulation of the hand in a virtual environment, opening the doorway to Virtual Reality and Augmented Reality. Virtual Reality (VR) refers to creating a computer-simulated environment where users can explore and interact with the digital world and digital assets. The term "Augmented Reality" refers to an interactive experience in which computer-generated perceptual data is used to enhance real-world items.

1.1 Background

Gesture recognition using IOTs and sensors was started in the early 1950s. Several attempts were made by many researchers in this field. The first commercially successful product was developed by Thomas Zimmerman in the early 1980's. With the advancement of sensor technology and computing power, more sophisticated devices have been created. These devices can be integrated with the Virtual and Immersive reality (VR) technologies that can find many applications that go far beyond gaming, in areas such as real estate promotion, education and medical training, museum exhibitions, showrooms, simulation of accident scenarios, police training, social training, etc.

Augmented Reality (AR) and Virtual Reality (VR) are rapidly evolving fields with the potential to transform daily life experiences. These technologies aim to create a sense of being in a completely different environment. Gesture recognition and AR VR technologies provide a way to interact with the virtual world. For example, a user can cast spells in a VR game with a finger pinch, or we can have a virtual tour of a real estate property and interact with the objects and furniture.

1.1.1 Finger Joints Study

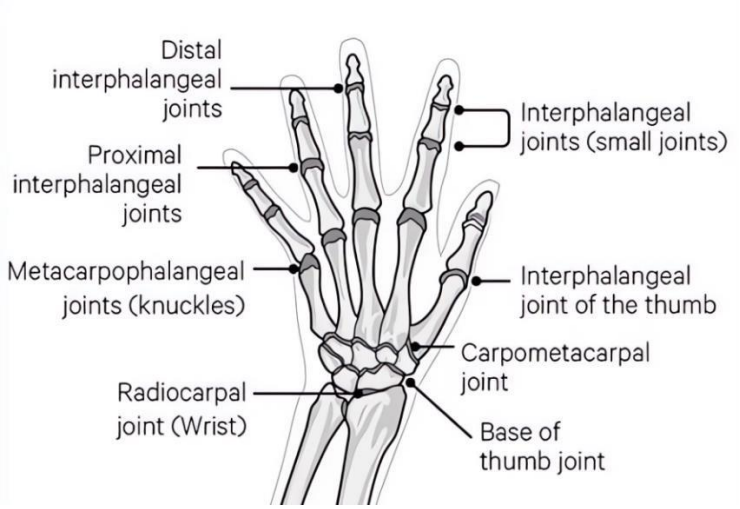


Figure 1-1: Different joints of human hand

Glove is designed considering the biological structure of a human hand. The anatomical structure of a human consists of multiple joints: Finger Joints, Wrist Joints, Elbow Joints and Shoulder Joints. This project only needs finger joints and wrist joints. Finger joints have further multiple joints for each finger: 2 for the thumb and 3 for the other four fingers. All these joints are called Interphalangeal and Phalangeal joints. Similarly, the wrist joint consists of Radiocarpal and Ulnocarpal joints among which only the Radiocarpal joint is needed. All these joints are shown in Figure 1-1.

Index, Middle, Ring, Pinky Finger: DIP (Distal Interphalangeal), PIP (Proximal Interphalangeal) and MCP(Metacarpophalangeal) Joints

Thumb: Interphalangeal and Metacarpophalangeal Joints

Wrist: Radiocarpal Joint

1.2 Motivation

In today's world, everything is first tested and developed as a simulation before being implemented in the real world. Even training sessions like pilot training are also conducted in a simulated environment, which mitigates the risk of plane crashes during actual training. The gaming industry has already integrated the simulation in various games like using car's steering wheel, accelerator, brakes, clutch and gears equipment

to simulate them in a car driving game. All these things aim to break the barrier between the physical and digital world and create an immersive experience.

In simpler words, we can consider these technologies as the future of the entertainment industry, learning and training industry and many more. All the tech giants of this time are competing to develop and launch their products in this category. Some of them are as follows:

1.2.1 Apple Vision Pro

Apple Vision Pro is Apple's augmented and virtual reality headset. This device marks Apple's first entrance into a new product category of virtual reality and augmented reality.

Apple vision pro is specifically a mixed reality headset that displays augmented reality content overlaid on the world around us. The R1 chip incorporated in the device processes input from the infrared cameras and motion sensors to actively track eye movements and hand gestures.

1.2.2 SteamVR

SteamVR is a virtual reality platform developed by Valve Corporation. It is a component of the digital platform. SteamVR is used to control and support VR controllers and headsets. SteamVR uses SteamVR Tracking which uses Base Stations that sweep the room with multiple sync pulses and laser lines reaching out to about 5 meters and tracking the timing between pulses and sweeps, the tracking system uses trigonometry to find the location of each sensor to within a fraction of millimeter. The tracking elements are usually located in the front of the VR headsets.

1.2.3 VRChat

VRChat is a well-known virtual reality platform for virtual interaction between user created avatars. Moreover, users can create and upload their own world and avatars using a software development kit of Unity game engine.

Furthermore, Meta, the company that owns popular social media platforms like Facebook, Instagram and WhatsApp, is also working on a virtual world of social media of its own.

The original concept of a virtual world is known as Metaverse, which is basically a virtual embodiment of the entire internet. From digital cryptocurrencies to social media platforms, everything will be a part of the metaverse in a virtual reality form. For example, social media profiles will be analogous to virtual avatars in the metaverse. Such an immersive experience is the goal of all the larger businesses today.

Hence, a small step towards an immersive experience in a virtual world by simulating human hand gestures is the core motivation of this project.

1.3 Problem Definition

The problem with virtual simulation is that it often lacks the capability to simulate the gestures in real time. This reduces the immersive experience for the user. So, this project aims to solve this issue by simulating the gestures in real time using WebSocket connection between client and server. Moreover, the commercial devices for capturing data for hand gestures and simulate them are pretty expensive. This narrows the accessibility of this technology to a limited audience. The hall effect sensors are extremely cheap as compared to other commonly used sensors like flex sensors. Engineering a device with similar capabilities as commercial devices for capturing gestures using hall effect sensors makes it accessible and affordable to a wider audience.

1.4 Objective

- To realize hand gestures in a virtual environment of Unity3D using hall effect sensors and a gyroscope sensor.

1.5 Application

The applications of this project are:

1.5.1 Sign Language

Sign language uses different hand gestures to communicate with deaf and hard-of-hearing individuals. Different hand gestures can be translated into words which helps them communicate with the one who does not understand sign language.

1.5.2 Motor Function Rehabilitation

Rehabilitation training is found to be the most effective for motor impairments in stroke patients as it reduces the possibility of the impairment getting worse and restores the upper limb motor functions. The rehabilitation using sensorized gloves utilizes technology to aid in the recovery of motor skills (hands and fingers) by diagnostic method. Gesture recognition gloves using sensors like Hall Effect Sensors can capture hand movements and gestures, allowing real-time monitoring and analysis [1].

1.5.3 Entertainment

The Sensorized glove can be worn and played around as a simulation for virtual games. Different environments can be rendered in the client application for the player's immersive experience.

1.5.4 Virtual tour

User can take a virtual tour of the real estate property using the VR devices, where they can visit different sections of house and interact with the objects, furniture, or open or close the door using the door handle with their simulated hand.

1.6 Scope

The scope of this project is listed below.

1.6.1 Capabilities of the Project

- The bend of the fingers can be simulated to its full degree of freedom.
- Wrist bend around X, Y and Z axes can be simulated.

1.6.2 Limitations of the Project

- Rotation of the hand beyond the range of the gyroscope sensor is not possible.
- Only one hand is simulated.
- Imitation of gestures involving overlapping of one finger over another is not possible.
- Positional changes of hand are not simulated.
- Proximity between fingers cannot be simulated.
- Glove may not fit for people with small hands.

1.7 Report Organization

This report is organized into nine chapters. The Introduction chapter is the formal opening of the report, which provides an overview of the project. Commentary on previously published similar projects is done in the literature review chapter. The analysis of instruments required for this project along with their specific usage is specified in requirement chapter, which is divided into two parts: Hardware Requirement and Software Requirement. Analysis of sensor data is also included at the end of this chapter.

The interconnection of all the system components, as well as the theoretical working principle and flow of the project is explained in system architecture and methodology chapter through the usage of various diagrams. Moreover, the Implementation Details chapter provides the details of how the project is engineered. The technical aspects of the project are deeply explored in this part.

The output of the system is documented and their behavior is analyzed in result and analysis chapter. Improvements on current version of the project as well as possibilities to take it further ahead is listed in Future Enhancement chapter. The project is summed up and report is formally concluded in the Conclusion chapter. Appendix chapter contains a list of miscellaneous figures such as circuit design, glove design, photograph of 3D printed case and code snippets.

2. LITERATURE REVIEW

Exos designed the Dextrous HandMaster which measures bend of fingers using hall effect sensors as potentiometers. Originally used as a master controller for MIT Dextrous Hand Robot, it has been redesigned by Exos, as an exoskeleton-like device worn on fingers and hands. The sensors used in the glove transmit analog signals to an A/D board at about 200 samples per second [2]. Although the glove is quite convenient to put on and the hall effect sensors measure the bend pretty accurately, it becomes unstable when the hand is moved rapidly.

Furthermore, Humanware created the Humanglove, a sensorized elastic fabric glove with 20 Hall effect sensors spaced evenly over the fingers. Different degrees of freedom (DOFs) associated with hand movements are measured by the sensors [3]. The glove uses Graphical Virtual Hand (GVH) software for data acquisition and display when connected to a computer via RS-232. The thumb, index finger, middle finger, ring finger, and little finger movements are included in the DOFs. Six subjects were used in the experiment, and grip repeatability was tested using custom plaster molds. Tests evaluating grip and hand positioning repeatability were conducted both with and without the glove. Further testing examined force and wrist flexion/extension as possible error sources. Segmentation, the extraction of important parameters, and statistical analysis were all part of the data processing for the repeatability evaluation [4].

Later, James Karmer developed the Cyber Glove at Stanford University using five flex Sensors and a wi-fi module ESP8266-07 embedded microcontroller. All of the glove's data is sent in UDP/IP packets by the ESP8266's Wi-Fi module to the Unity program, which runs on the host computer and handles data processing and graphic rendering. A mathematical function to map the reading obtained from flex sensor into an angle in a range 0 to 180 which is then fed to Unity’s Hinge Joint component that couples two rigid GameObjects. This enables a rotation along one of the common axes, reproducing hand gestures [5]. The main drawback of this cyber glove is that it uses flex sensors to detect the hand gesture. Flex sensors are sensitive to direction of bending and the probability of getting flex sensor increases as the device is used frequently. That is why, we tried to use hall effect sensors to increase the life span of the glove.

Regarding flex sensors, W. Industries designed the virtual simulation glove Space Gloves. The glove uses a soft molded plastic that fits over the back of the hand. Rings around the fingers and straps around the wrist hold the glove in place. One flex angle for each joint of four fingers and two flex angles for the thumb are measured using a sensor with 12-bit analog-to-digital converter. The stiffness of the plastic makes the glove hard to get the rings over the finger joints when putting on or taking off the glove. Due to the plastic structure used, finger movement was affected.

Overcoming the plastic structure, Tom DeFanti and Daniel J. Sandin developed Sayre glove in University of Illinois at Chicago [6]. They used a flexible tube with a light source at one end and a photocell at other end. Tubes were mounted along each fingers of the glove. As the tube was bent the amount of light passing sensor and photoelectric cell decreased evenly. Voltage from each photocell was correlated with the finger bending. Abduction and adduction of a finger cannot be calculated using this technology.

In 2019, Suramya Sharma Dahal, Adarsh Ghimire, Aakriti Basnet, Anushma Shrestha, and Bhupendra Kadayat developed a smart glove at Tribhuvan University [6]. They used six flex sensors, which were attached to the thumb, index finger, middle finger, pinky finger, and palm of the glove, to measure the bend of a finger and the clench of the hand. An accelerometer and gyroscope sensor were placed in the back of the hand to determine the position and movement of the hand. Those sensors were interfaced with the Arduino mega 2560. Data from those sensors was then fed to a machine-learning algorithm, which translated the hand gesture into text. Though the accuracy of the model was near about 80 percent use of flex sensors increased the cost of the project. The durability of flex sensors decreases when frequently used.

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirement

3.1.1 Hall effect Sensors (49E)

The hall effect sensor measures the magnetic field around it. This capability can be utilized to map the change in magnetic field occurred at finger joints (as a result of the placement of the sensor and the magnet around the joint) to the phalanges bend.

3.1.2 MPU6050

MPU6050 is a three axes accelerometer and gyroscope sensor. Its acceleration data can be used to calculate the pitch and roll angles of the wrist bend whereas its gyroscope data can be used for calculating the yaw angle.

3.1.3 ESP32

ESP32 is a low-cost, low-power microcontroller with integrated Wi-Fi module and dual mode Bluetooth. It is required for interfacing the hardware components and sensors, collect the data from them, and send the data to the server.

3.1.4 AMS1117

AMS stands for Advanced Monolithic System. It is a linear voltage regulator used to provide a stable voltage to electronic circuits. When 4.8V is supplied to its input, it generates a stable output voltage of 3.3V. To power the hall effect sensors with 3.3V, AMS117 is required in this project.

3.1.5 LM7805

LM7805 is a linear voltage regulator device. It produces a stable output voltage of 5V when voltage between 7 to 20V is supplied. LM7805 is required to supply 5V power up ESP32.

3.1.6 Analog Multiplexer (CD74HC4067)

Analog multiplexer is a device that chooses one input signal from multiple input sources and routes it to a single output channel. Selection of an input signal is done by four selection lines. In this project, 14 hall effect sensors are to be interfaced to ADC. The ESP32 supports Analog-to-digital conversion on 15 pins only. When the Wi-Fi module of the ESP32 is used, the ADC pins relying on a second SAR-ADC cannot perform the conversion which limits us to use only 5 ADC pins of ESP32 for analog to digital conversion. So, using a 16-channel analog multiplexer makes it possible to read the data from 14 different sensors by time multiplexing to get the inputs from multiple sensors.

3.2 Software Requirement

3.2.1 NodeJS

NodeJS is a JavaScript runtime environment which is used for executing JavaScript code outside of the browser. It provides functionalities for server-side scripting. Its fast server processing with non-blocking event loop makes it suitable for real time transmission of data between ESP32 and Unity Application. It is compatible with socket programming, so that a seamless and continuous transmission of data is ensured.

3.2.2 Blender

Blender is a software which is extensively used for modeling 3D objects. In order to correctly map the data obtained from hall effect sensors at finger joints and the gyroscope sensor, the model was rigged, or in simpler words the hand model was provided with bones using Blender.

3.2.3 Unity Engine

Unity is a widely used platform for developing games. Unity provides the feature for importing the hand model and rendering it. The hand model can be controlled through the use of scripts. When a script is attached in an object (also known as GameObject) in Unity, the object's behavior can be programmed. Unity compiles and builds all the scripts and GameObjects and produces binaries, which are specific to a platform.

3.2.4 . NET

.NET (dotnet) is an open-source software framework for developing applications on Windows, Linux and MacOS platforms. It compiles and provides a runtime environment for applications developed in Unity.

3.3 Sensor Data Analysis

3.3.1 Hall Effect Sensor Data Analysis for Bending of Finger-Joint

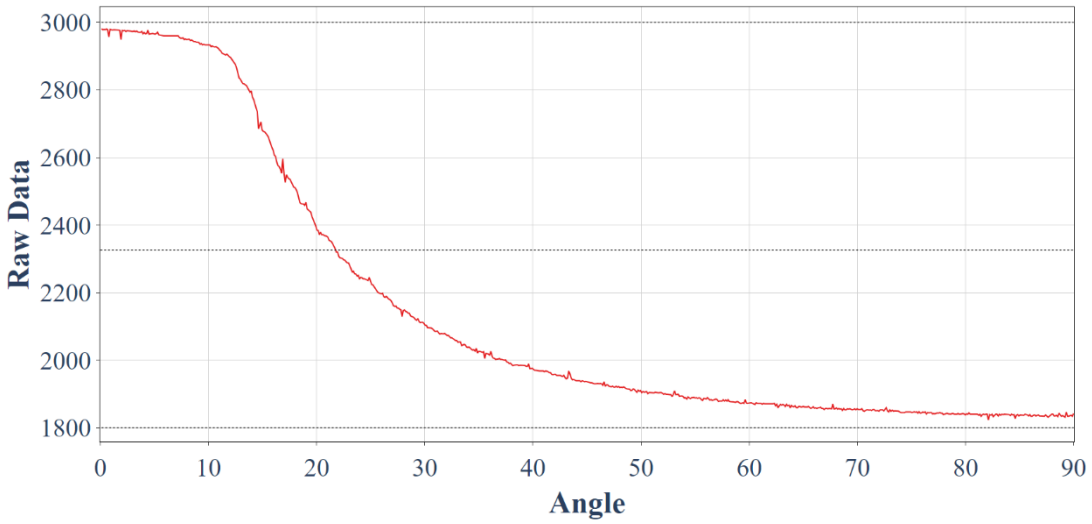


Figure 3-1: Raw Data vs Angle

The approximate curve of the raw data from different Hall Effect sensors vs the bending angle of finger is shown in Figure 3-1. To obtain this data, a separate setup was established as shown in Figure 9-7 of Appendix E. The hall effect sensors were attached to the shaft of a stepper motor and magnet is placed in front of the sensor. Then the shaft is rotated to move the sensor away from the magnet. As stepper motor can precisely rotate the shaft, the shaft was rotated by around 0.115 degrees in each step and the reading from hall effect sensor was logged in a CSV format.

The data from the ESP32 is written into a file by using the PuTTY client software. To log the data, the PuTTY client is run and the connection type was set to Serial and the COM port in which the ESP32 was running was selected along with the appropriate baud rate. i.e. 115200 baud/sec. The data was then mapped and plotted using a python script.

The data clearly shows that there is no linear relationship between the bending angle of the joint and the data from the hall effect sensor. Form this it can be concluded that a simple linear curve is not able to accurately map the raw data into the angle data for the motion that involves both the translational and rotational motion of either magnet or the Hall Effect sensor. To counter this problem, Piecewise Linear Mapping method is utilized.

Although the whole curve doesn't follow the linear pattern, the parts of the curve tend to show a linear pattern in multiple pieces. So, Piecewise Linear Mapping method is suitable to map the raw data to angle.

3.3.2 Output Data Analysis of Multiple Hall Effect Sensors

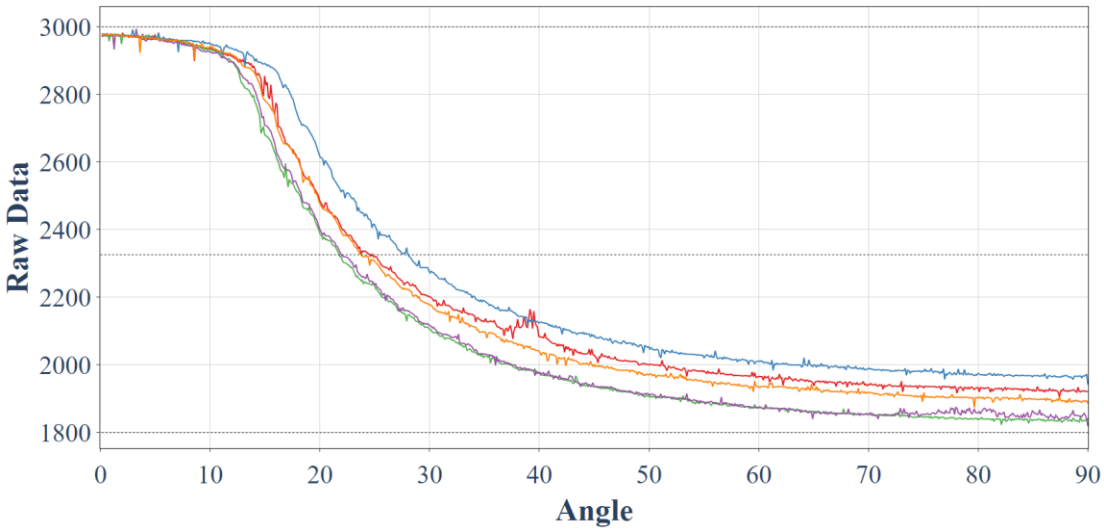


Figure 3-2: Plot of Raw Data vs Time of Five Different Hall Effect Sensors

Figure 3-2 shows the base value and highest value comparison between the data from five different Hall Effect sensors. The plot from the different hall effect sensors is not exactly same which is due to the manufacturing anomaly but the slope of the plot at same time interval are nearly same. With small amplitude shifting and scaling, the plot of all the sensors can be made nearly identical. The scaling and shifting factor for each hall effect sensor can be obtained manually by studying the plot of the sensors individually. After these transformations of the signals, the same piecewise linear mapping function is applied to map the raw data from all the hall effect sensor into the angle data.

The shifting and scaling applied dynamically to the different sensors readings using python script is shown in Figure 3-3. The figure shows that scaling and shifting the signals in amplitude can improve the data accuracy to some extent.

3.3.3 Scaling Raw Data

As every Hall Effect Sensors have different maximum and minimum values, the raw data obtained from the sensors need to be scaled to a common maximum and minimum values.

To obtain this scaling factor, first of all a maximum and minimum value is chosen at which all the sensors are to be fitted. Maximum value is taken as 3000 and minimum value is taken as 1800. The difference between this maximum value and the current maximum value of sensor is added to all the values of the sensor. Similarly, the difference between the minimum value to be fitted to and the current minimum value is taken and halved. Then, this value is reduced from all the values of the sensor data. Similarly, the average of the fitting maximum value and minimum value is reduced from the sensor data.

Finally, the maximum value is reduced from the average value and divided by the new maximum value of sensor data obtained from above calculation. This gives the scaling factor and it is multiplied with every data from sensor. Finally, to reposition the data, the values are added with the average value of maximum and minimum.

This approach is utilized in python script to obtain a graph showcasing the new scaled values.

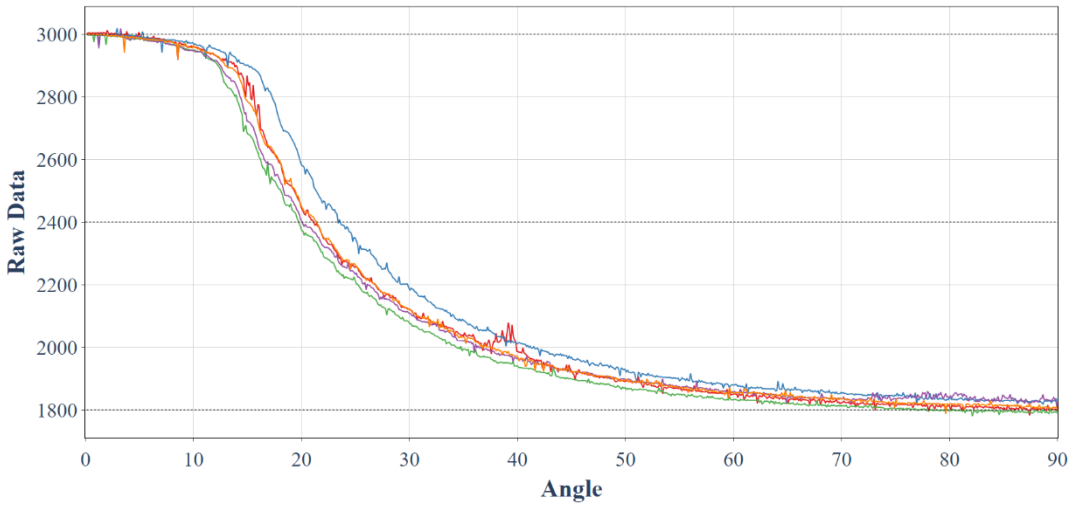


Figure 3-3: Scaled Plot

3.3.4 Theoretical Analysis of Behavior of Hall Effect Sensor on 3.3V Supply

The datasheet for 49E Hall Effect sensor provides the plot of output voltage vs magnetic field when supply voltage is 5V as shown in Figure 9-12. Similarly, to get the idea of output voltage of the sensor according to the supply voltage, the manufacturer has provided a graph of output voltage and supply voltage at 0 Gauss as shown in Figure 9-11.

But since ESP 32 has an operating voltage of 3.3 Volts, the hall effect sensors will have to work with the supply voltage of 3.3 Volts. So, we need to deduce how the output of the sensor will be changed when magnetic field changes when the supply voltage is 3.3V. Using these plots and mathematical equations, we can develop the plot for 3.3 Volts.

From the plot of Figure 3-4, we can see that there is a linear relationship between the output voltage and supply voltage. So, a simple equation of line can define this relationship.

Taking two approximate reference points in the graph: (At 0GS)

(Vcc1, Vout1) = (4V, 2.1V) and (Vcc2, Vout2) = (5V, 2.6V), we can derive the equation of the straight line by using two-point form:

$$(\text{Vout} - 2.1) = \frac{(2-2.1)}{(5-4)} * (\text{Vcc} - 4)$$

$$Or, Vout - 2.1 = 0.5 * Vcc - 2$$

$$Or, Vout = 0.5 * Vc + 1$$

Now defining two equations:

For Vcc1 = 5 V and Vout = Vout1,

$$\mathbf{Vout1 = 0.5\,Vcc1 + 0.1} \quad \textbf{(3-1)}$$

For Vcc2 = 3.3V and Vout = Vout2,

$$\mathbf{Vout2 = 0.5\,Vcc2 + 0.1} \quad \textbf{(3-2)}$$

Subtracting equation 3-1 from equation 3-2,

$$Vout2 - Vout1 = 0.5(Vcc2 - Vcc1)$$

$$\text{Vout2} = \text{Vout1} - 0.85$$

$$Vout1 = 1.6 * 10^{-6} * B + 1.65$$

From above calculation, it is clear that we will have an output voltage of hall effect sensor that will be in the range of 0 to 3.3 Volts. So, we can easily work with the hall effect sensor with ESP 32 operating voltage range without any voltage conversions.

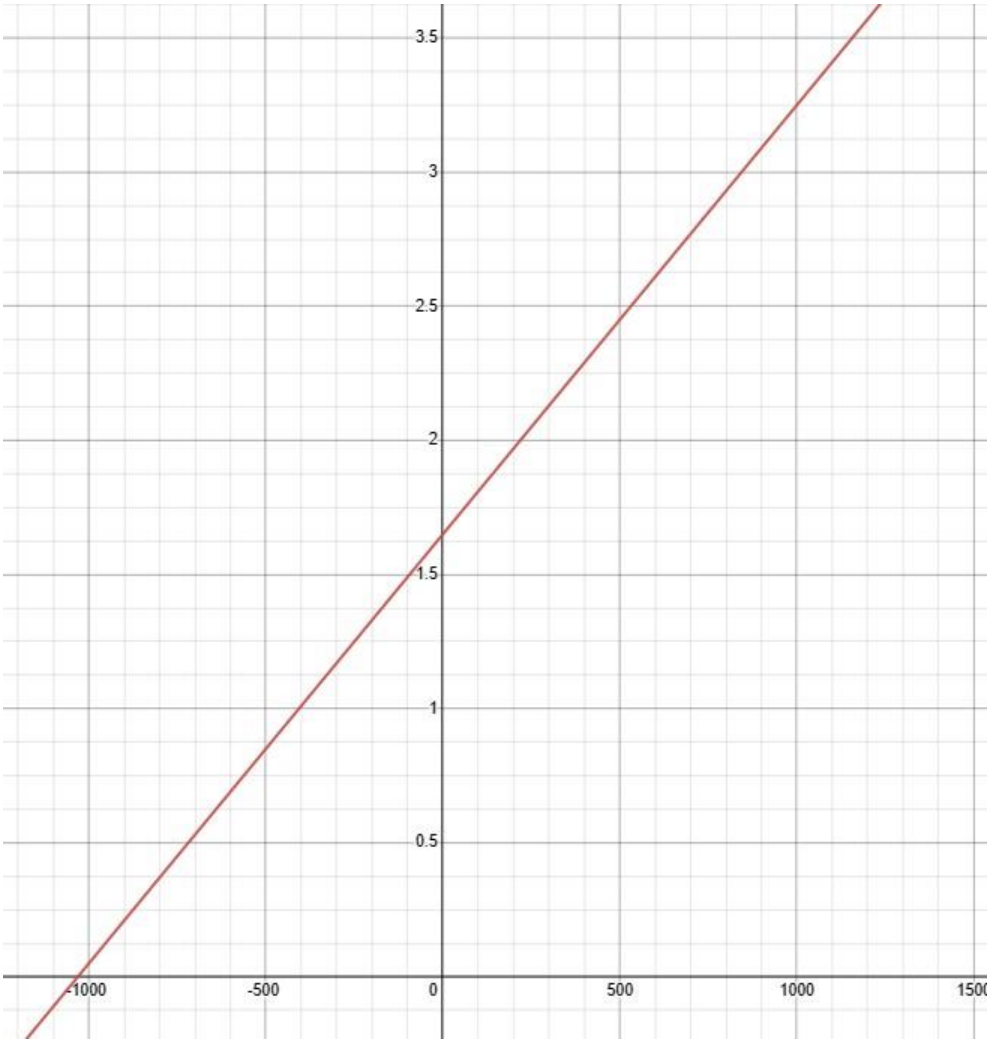


Figure 3-4: Vout vs Magnetic Field when Vin = 3.3V

3.4 Feasibility Study

The Hall Effect sensors can measure the magnetic field accurately between -1000 Gs to 1000 Gs and produce linear voltage output with change in magnetic field. This range is suitable to measure the magnetic field produced by the 10mm diameter N35 Neodymium magnet of 5mm thickness used in this project.

The ADC of ESP32 used to measure the analog voltage produced by the Hall Effect sensors is a 12-bit ADC that can measure the voltage range between 0V to 3.3V and map them linearly to the range of integer values between 0 and 4095. When 3.3V supply is provided to the Hall Effect sensors, they also produce the voltage output in the range of 3.3V. So, the ADC is feasible to measure the sensors' output. Moreover, the output range of ADC of ESP32 is also suitable to map the angle data from 0 to 90 degrees.

The MPU6050 sensor can also accurately measure the acceleration data of all three axes up to sixteen times the value of acceleration due to gravity i.e. 16*g, which is more than enough for this project. These data can be used to calculate the angle of rotation along x and y axes. The acceleration data cannot be used to measure the angle of rotation along z axes. So, the gyroscope data can be used to calculate this angle. Thus, MPU6050 is feasible to detect the wrist bending.

Furthermore, the Hall Effect sensors and Neodymium magnets are attached on placement attachments and glued on a glove. The ESP32 microcontroller and Gyroscope sensor are placed on the Matrix board on the dorsal side of the glove. This design is feasible and has a simple circuitry.

Moreover, the blazing fast speed of NodeJS server minimizes the server-side processing time. This ensures the data is continuously taken and fed to the application software. The Unity3D engine provides a high level and interactive environment for application development. Exporting the project to the targeted platform is just a work of a few clicks as well. In addition to that, object-oriented high-level programming with .NET framework makes it an excellent choice for scripting the behavior of the virtual hand object.

Furthermore, the cost of the instruments does not exceed the budget range, so it is economically feasible as well. Hence, the project is feasible.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 System Block Diagram

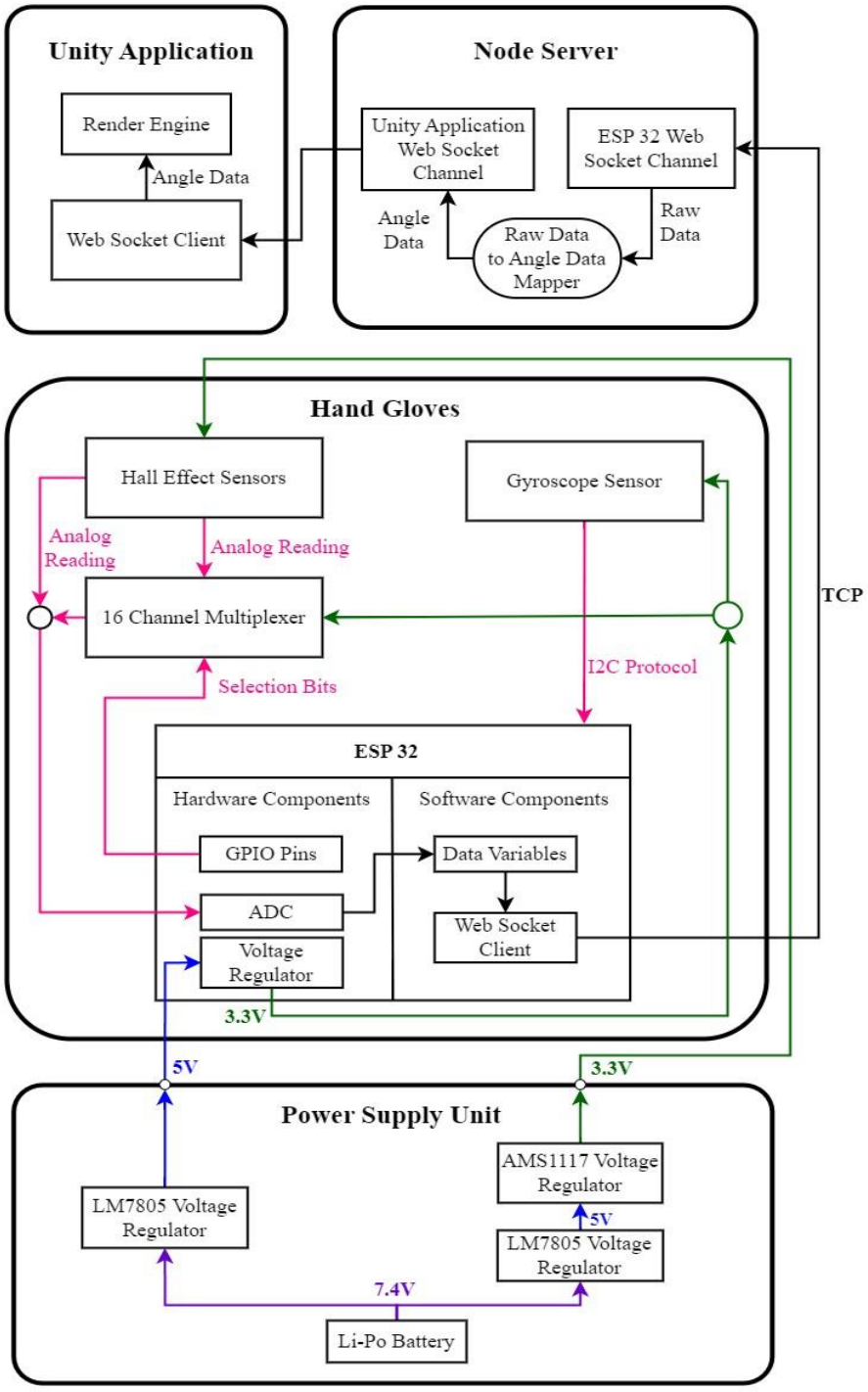


Figure 4-1: Proposed System Block Diagram

4.1.1 Hand Gloves

The hand glove is the part that holds all the sensors and circuit components of the project. It consists of the glove model with the magnets and hall effect sensors placement. The data from Hall Effect sensor and MPU6050 are read and stored in ESP32. To cope up with the shortage of analog pins a 16-channel analog multiplexer is used. The collected data is then sent to the Node Server for processing using WebSockets.

4.1.2 Power Supply Unit

This unit solely functions to power the circuit and components. The 3.3V and 5V supply is generated by this unit which are used to power the ESP32 and the other sensors. This unit provides enough current to run the circuitry without any current outage.

4.1.3 Node Server

This unit is the connecting block between the hardware and the simulation software. It provides WebSocket API for both ESP32 and Unity Application to connect with it. It receives the sensor data from the ESP32, applies proper mapping techniques to transform raw data to useful data and then sends the data in real-time to the Unity Application.

4.1.4 Unity Application

The Unity Application displays the final simulation output. It receives the angle data and maps them to different parts of the hand model and then moves those parts according to the data creating the real-time simulation of the virtual hand.

4.2 Working Principles

4.2.1 Hall Effect Sensor (49E)

49E is the linear hall effect sensor. It is a small integrated sensor that converts the magnetic field into analog voltage. It operates based on the Hall effect, a physical phenomenon where a current-carrying conductor experiences a magnetic force when

the magnetic field is applied perpendicular to it. The magnetic force experienced by the charged particle of the current-carrying conductor is given by

$$\boldsymbol{F = q(\vec{V}\, X\, \vec{B})} \qquad \textbf{(4-1)}$$

Where, **F** = Magnetic Force

q = Charge of electron

V = Velocity of electron

The direction of the magnetic force is given by Fleming's right-hand rule. Due to this force, charged particles of different polarity move apart until they balance the electric force given by

$$\boldsymbol{F_e = q * E} \qquad \textbf{(4-2)}$$

Where, $\mathbf{F_e}$ = Electric force

q = charge of electron

E = Electric Field Intensity

On equating equations (4-1) and (4-2) and solving them,

$$\boldsymbol{V_h = \frac{I*B}{n*e*t}} \qquad \textbf{(4-3)}$$

Where, $\mathbf{V_h}$ = Hall Voltage

I = Current through conductor

n = number of charged particle per unit volume

e = charge of an electron (1.6×10^{-19} C)

t = thickness of conductor

This amount of hall voltage generated by the hall element is given by the above equation 4-3. Hall voltage is usually in the range of a few millivolts, so the amplifier present inside the IC amplifies it.

As from the above expression, it is clear that hall effect sensors can detect the presence of a magnetic field near to them. So, to detect the movement of the finger joints a

magnet is placed at one end and the hall effect sensor at the other end. When the proximity between the sensor and the magnet changes, the output of the sensor changes.

4.2.2 MPU6050 Sensor

MPU6050 is a motion-sensing device that uses a 3-axis accelerometer and a 3-axis gyroscope for measuring motion and orientation. Accelerometers are based on MicroElectro-Mechanical-System (MEMS) fabrication technology. It is a micro-machined structure built in a silicon wafer, as shown in the figure below.

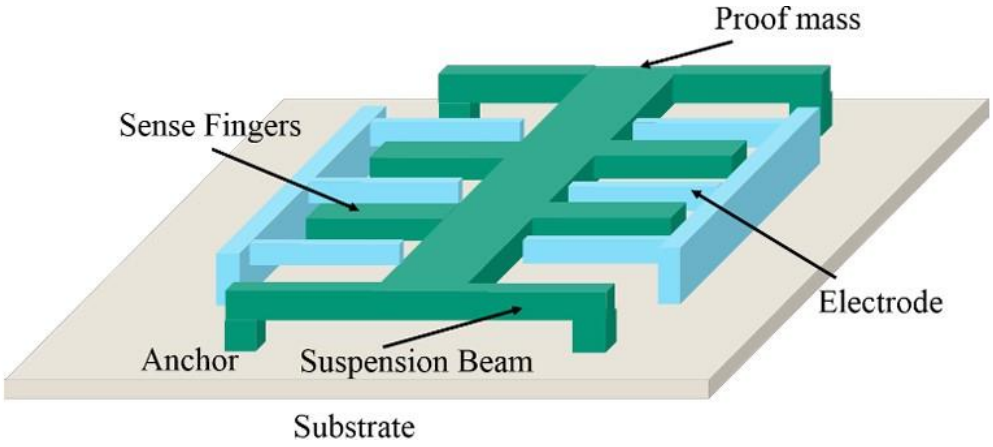


Figure 4-2: Structure of MPU6050 [8]

The Silicon wafer contains a polysilicon spring/suspension bridge, which allows the proof mass to deflect when accelerated along the X, Y, and/or Z axes.

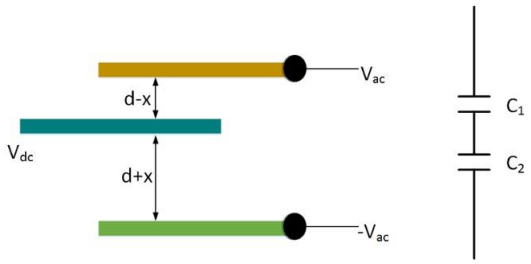


Figure 4-3: Structure of Wafer [8]

As a result of the deflection of the proof mass, the capacitance between fixed electrodes and plates attached to the suspended structure changes as shown in the figure. This change in capacitance is proportional to the acceleration along that axis. The sensor

processes this change in capacitance and converts it into an analog output voltage. In this project to find roll angle (angle of rotation along x-axis) and pitch angle (angle of rotation along y-axis) we decided to use the accelerometer sensor. The formula that gives the relation between acceleration and angle is shown below

$$\boldsymbol{\theta = tan^{-1}\left(-\frac{A_x}{\sqrt{A_x{}^2 + A_z{}^2}}\right)} \quad \textbf{(4-4)}$$

$$\boldsymbol{\Phi = tan^{-1}\left(\frac{A_y}{\sqrt{A_y{}^2 + A_z{}^2}}\right)} \quad \textbf{(4-5)}$$

Where, Θ = pitch angle

ϕ = roll angle

A_y = acceleration along Y-axis

A_x = acceleration along X-axis

A_z = acceleration along Z-axis

Gyroscope sensors are used to measure angular velocity. The angle around the X axis is found by integrating the gyroscopic data $\dot{\theta}$x over a small amount of time δt such as

$$\boldsymbol{\theta_x(t + \delta t) \cong \theta_x(t) + \theta_x \dot{} \, \delta t} \quad \textbf{(4-6)}$$

The same equations hold for the Y and Z axis.

Gyroscope sensors are also based on Micro-Electro-Mechanical-System (MEMS). It consists of a proof mass consisting of four parts M1, M2, M3, and M4 as shown in the figure.

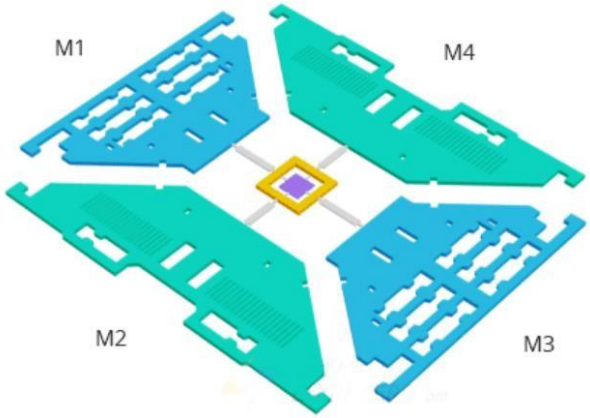


Figure 4-4: Micro Electro Mechanical System

Those proof masses are maintained in a continuous oscillating movement so that they can respond to the Coriolis effect. They simultaneously move inward and outward in the horizontal plane. When the structure starts to rotate, the Coriolis force starts to act on the proof masses, which causes the vibration to change from horizontal direction to vertical direction. Based on the axis along which the angular rotation is applied, the proof mass rotates in different directions. When the angular rotation is applied along the X-axis, masses M1 and M3 will move up and down out of the horizontal plane due to the coriolis effect. When the angular rotation is applied along the Y-axis, masses M2 and M4 will move up and down out of the horizontal plane due to the Coriolis effect. When the angular rotation is applied along the Z-axis, masses M2 and M4 will move horizontally in opposite directions due to the Coriolis effect. Due to the change in motion, Capacitance will be changed. This change in capacitance is proportional to the angular velocity applied. Sensor processes this change in capacitance and converts it into analog output voltage. As Yaw refers to the angle of rotation along the Z-axis, when the sensor is rotated along the Z-axis the magnitude of the force acting along the Z-axis doesn't change, so the accelerometer cannot measure the yaw angle. To measure the yaw angle, a gyroscope sensor is used.

The MPU 6050 gyroscope sensor uses the I2C communication protocol which is a serial communication protocol that communicates using 2 lines, namely SCL and SDA. It works on a voltage range of 2.375V - 3.46V which is compatible with the operating voltage of ESP 32. SCL (Serial Clock Line) is controlled by the master and

determines the timing of the data bits being communicated. The SDA (Serial Data Line) carries the actual data between the devices, the information on this line is only allowed to change when the signal on SCL is low.

4.2.3 SAR-ADC

SAR-ADC stands for Successive Approximation Register Analog to Digital Converter.

Circuit diagram for 8-bit SAR-ADC is as shown in the Figure 4-5.

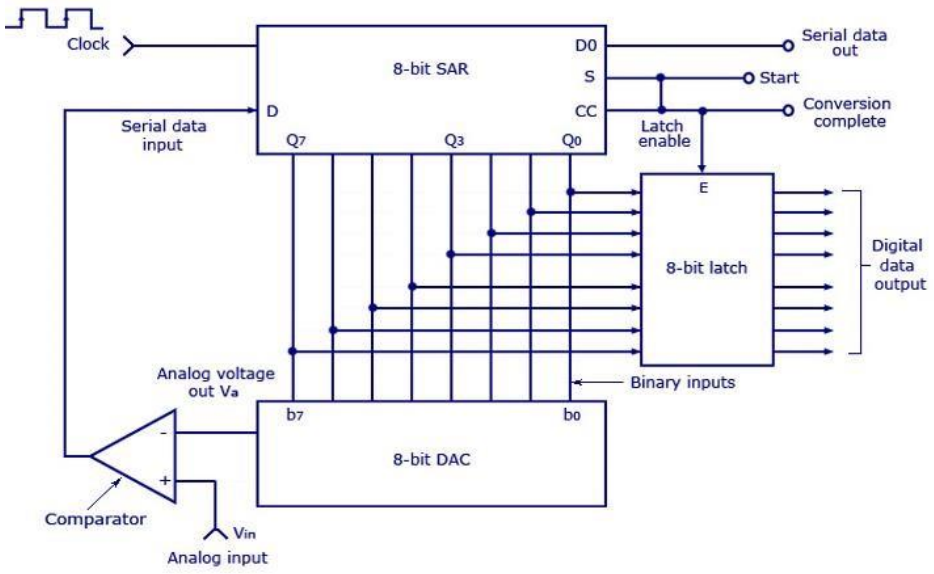


Figure 4-5: 8-bit SAR-ADC

The circuit consists of a comparator, a digital-to-analog converter, and an 8-bit latch. SAR-ADC implements the binary search algorithm for its operation. It performs a binary search through all bits. For each bit, they have three possibilities: it can be set to 1, or it can be reset to 0, or maintain its value. At the start of conversion cycle, on positive edge of clock pulse Most Significant Bit (MSB) is set to '1' and other bits are reset to '0'. The digital-to-analog converter generates the equivalent analog voltage. The comparator circuit compares the output of DAC and the input analog voltage. If the analog output voltage is lower than DAC output then the comparator gives the low value and the MSB is set. If the analog voltage is higher than the DAC output then the comparator gives the high value and the MSB bit is retained and on the next positive edge the bit next to the MSB is set to '1' and the same procedure is repeated until the best digital value for the input signal is found. For a 12-bit SAR-

ADC, a total of 12 comparisons are made to find the value for 12 bits for any input voltage level. So, the time taken to convert the data is constant for any value of input voltage level.

4.2.4 16-channel Analog Multiplexer

An analog multiplexer is a combinational circuit that selects information from one of many lines and directs it to a single output line. The selection of particular input lines is controlled by a set of selection lines. For 2^n input lines there are n selection lines. Bit combination of those n selection lines selects the input line. In CD74HC4067 analog multiplexer, 4 selection lines are used to select the 2^4 i.e. 16 input lines. Those selection lines can be high or low at a time. For a particular bit combination, specific channel is selected as shown in the table.

Table 4-1: Truth Table of 16-channel Analog Multiplexer

S0	S1	S3	S4	E'	Selected Channel
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

4.2.5 Voltage Regulators

Voltage regulator is a circuit that produces a fixed-stable output voltage when input is supplied. Two different voltage regulators i.e. LM7805 & AMS1117 are used in this project.

LM7805 is a voltage regulator that belongs to the 74xx family, where "xx" in their name represents their output value. LM7805 is made up of different blocks as shown in Figure 4-6.

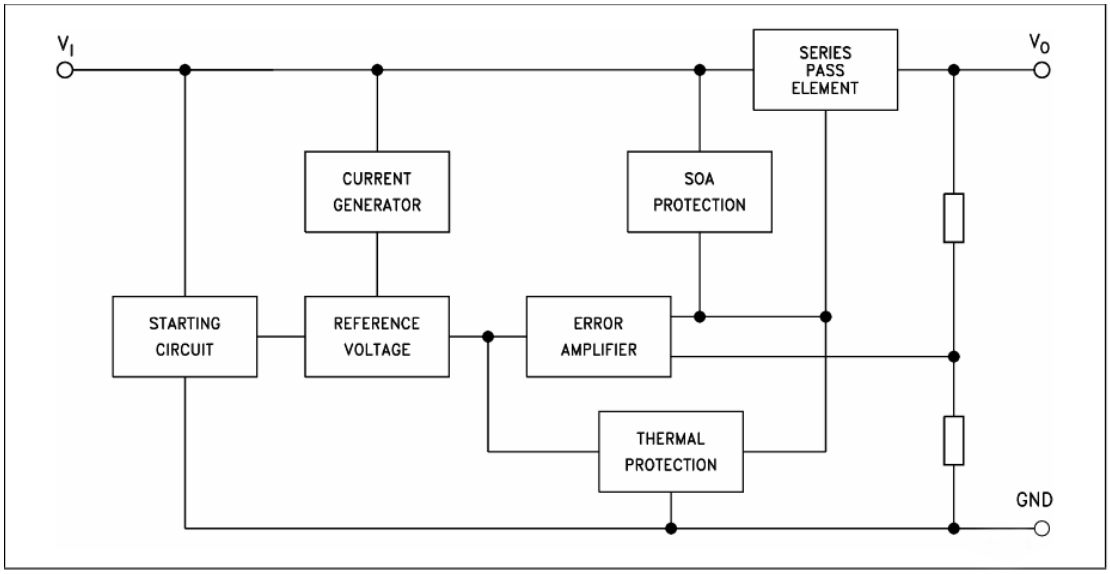


Figure 4-6: Block Diagram of LM7805

When an input voltage in the range of 7-20 V is supplied to the input terminal, starting circuit block operates the circuit and ensures the smooth startup by managing the necessary initial conditions. The current generator block generates constant current that flows throughout the circuit. The reference voltage block generates a fixed output voltage for comparison purpose. Error amplifier amplifies any difference between the reference voltage and the feedback voltage to adjust the output voltage accordingly. Safe Operation Area (SOA) ensures that the components are working in specified limits. If any component exceeds its limits, then the thermal protection block shuts down or reduce power to prevent from damage. Series Pass element is basically an internal transistor that provides a steady voltage at output end.

AMS1117 is a voltage regulator that produces a fixed–stable 3.3V at output end when an output of 4.8V is supplied. Internal circuit diagram of AMS1117 is shown in Figure 4-7.

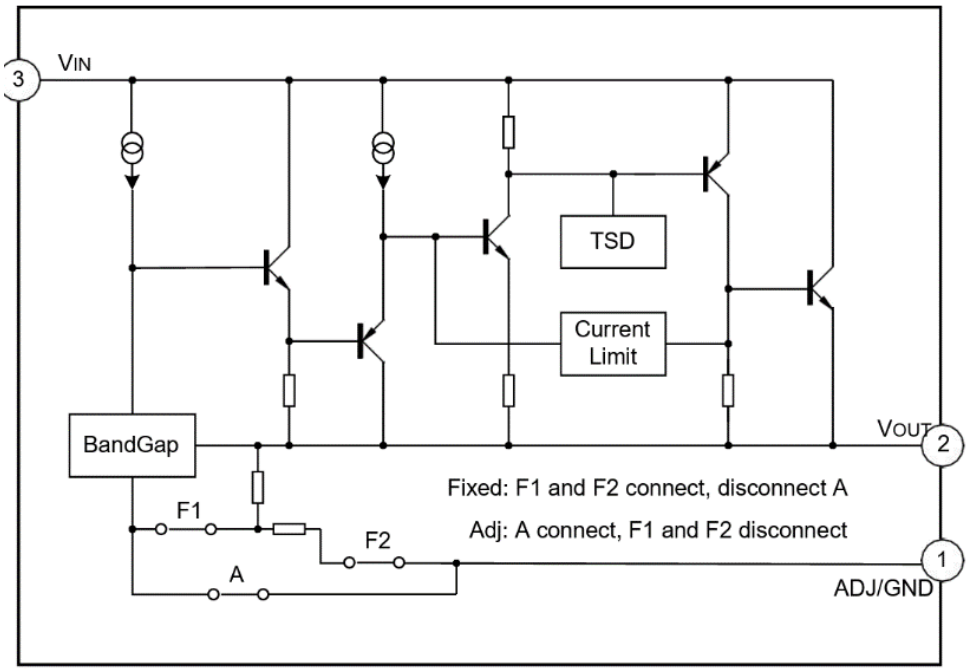


Figure 4-7: Circuit Diagram of AMS1117

When an unregulated input voltage is supplied at the input terminal of AMS1117. BandGap block generates a stable voltage which is used to compare with the output voltage. The current limit block ensures that output current does not exceed the maximum limit. If the temperature of the circuit exceeds the maximum limit, then Thermal Shutdown (TSD) block shuts down the circuit to prevent from overheating.

4.2.6 I2C Communication Protocol

The I2C communication protocol is a serial communication protocol. In this protocol, two lines are involved for data communication: SCL and SDA. The SCL pin involves clock pulses for synchronizing tasks. On the other hand, SDA is used for data transportation.

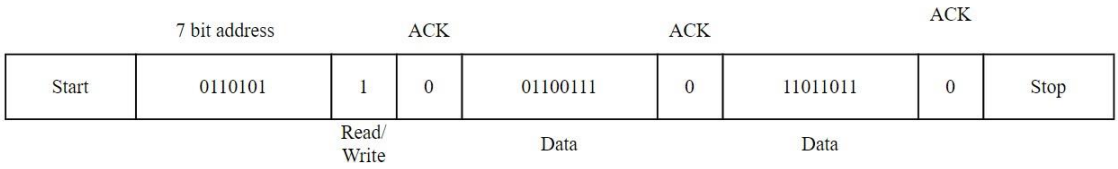


Figure 4-8: I2C Message Format

At first, a start byte is sent to initiate the communication. Then, a 7-bit address message is sent through the SDA line to locate the communicating device. Since MPU6050 has both read and write mode, it needs to be specified on the next bit. 0 is for write and 1 is for read mode.

Each byte of message needs to be acknowledged in I2C communication to tackle any discrepancy. The controller can proceed to the next group of bits after the receiver notifies it that the previous information has been acknowledged.

The next step is to send a byte of data. This data is again acknowledged, another data pack is sent and again this data is acknowledged. Finally, a stop sequence is sent to signal the completion of communication.

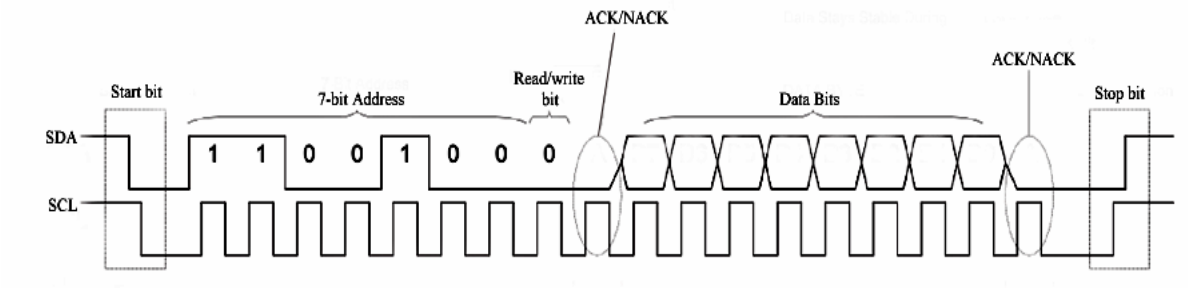


Figure 4-9: Timing Diagram of Data Communication Using I2C Protocol

4.3 Work Flow

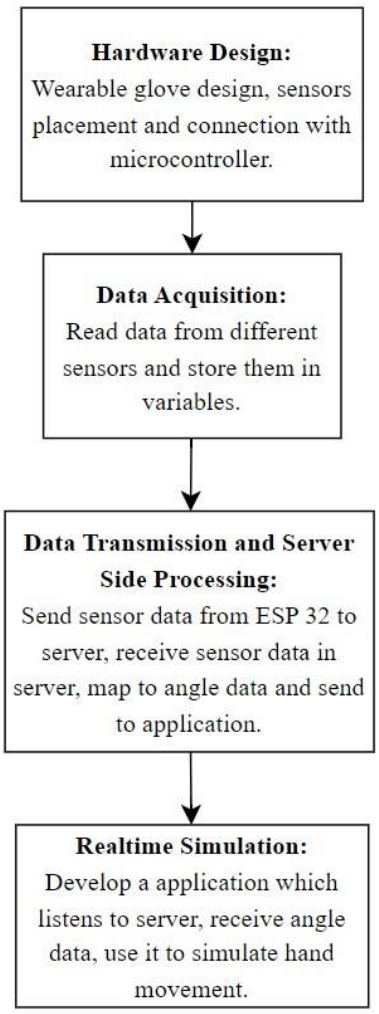


Figure 4-10: Project Work Flow

This project has the following work flow:

4.3.1 Hardware Design

Neodymium Magnets and Hall Effect sensors are placed on each finger joint. So, we need to place 14 magnets and sensors for this purpose. The magnets and sensors are carefully placed on a glove, which is worn by the user.

The attachment is designed in such a way that in normal state the magnet sits adjacent to the Hall Effect Sensor and when the joint moves, the magnet moves near to and away from the sensor relative to the joint motion, which gets detected by the sensor.

The attachments are fixed permanently on the glove. For easier mapping and consistent data acquisition, complete glove is designed based on this attachment.

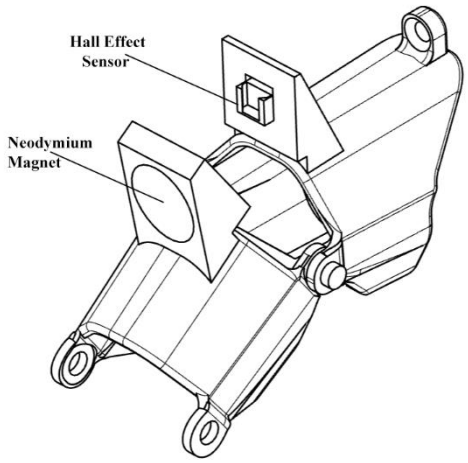


Figure 4-11: Magnet and Sensor Attachment for Single Joint

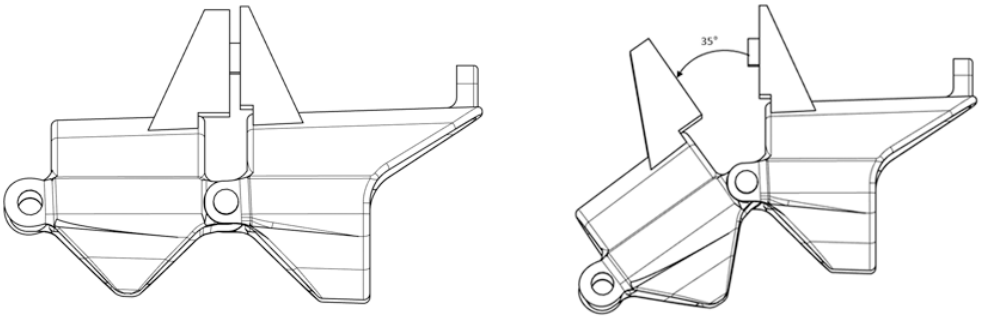


Figure 4-12: Working of Attachment

The design in Figure 4-12 shows the working of sensor and magnet together attached over the joint of the finger. As seen from the figure, when the finger is bent to a certain degree, the distance between the sensor and magnet changes. As the magnetic flux density of the magnet changes with distance from it, the sensor detects that change. Along with translational motion of the magnet, the data from the sensor is also affected by the rotational motion of it with respect to the sensor. So, the sensor data changes due to both translational and rotational motion of the magnet. This data changes can be further processed to calculate the angle of rotation of the finger joints.

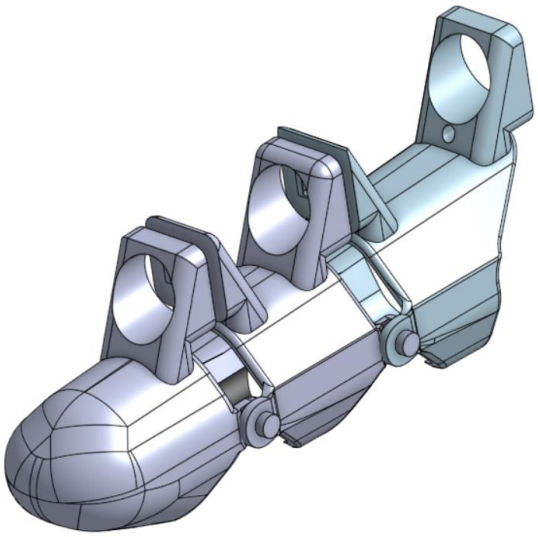


Figure 4-13: 3D Model of Glove with Sensors and Magnets

As this project depends heavily on the accuracy of the magnet and sensor placement, a glove is designed that fits around the finger and fixes the fulcrum of each joint to obtain consistent and accurate sensor readings. The glove fits fingers precisely and contains sensors and magnets while maintaining the essential functionality of the aforementioned basic attachment design as shown in Figure 4-13. To keep the sensor value consistent across all Hall Effect Sensors, both the Magnet and Sensor are placed at the same position on each joint. This also limits the bend of each joint to exactly 90° to aid in sensor data mapping. In this design, the global average of each finger's measurement was taken to design the glove for the particular hand.

4.3.2 Data Acquisition

The data required for simulation comes from two types of sensors:

- Hall Effect Sensor
- Gyroscope Sensor

Taking Data from the Hall Effect Sensors

All total of 14 Hall Effect sensors will be used as described in the Hardware Design section. Those 14 sensors will be used to determine the bend of fingers.

ESP32 consists of two 12-bit SAR ADC which produces value within range of 0 to 4095 mapped linearly with voltage range of 0 to 3.3 volts. The voltage produced from the Hall Effect sensors are read via the analog input pins of the ESP32. ESP32 WROOM32 DevKit only offers 15 pins that support analog input. Among those 15 pins, 5 pins are connected to ADC1 and remaining 10 pins are connected to ADC2 within ESP32. Furthermore, when Wi-Fi functionality of the ESP32 is used, the ADC2 of ESP32 is not functional. This removes the functionality of analog to digital conversion from 10 pins. Since the project requires the Wi-Fi functionality of ESP32 for data transmission to server, only 5 pins supporting ADC are functional. But 18 Hall Effect sensors are to be interfaced which induces the shortage of analog input pins. To solve this issue, a 16 channel analog multiplexer module (CD74HC4067) is used and time multiplexing is implemented to take 16 analog inputs from the single analog pin of the ESP32. The multiplexing circuit in turn uses 4 digital pins of ESP32 to specify selection bits. Since ESP 32 has an 80-240 MHZ clock, any kind of noticeable delays between time multiplexed readings will not be encountered.

Each of these raw data values are then stored in 14 separate variables which will be used later during data transmission.

Taking Data from the Gyroscope Sensor.

The MPU 6050 gyroscope sensor uses the I2C communication protocol which is a serial communication protocol that communicates using 2 lines, namely SCL and SDA. ESP 32 also supports the I2C communication protocol via SCL and SDA pins.

The MPU6050_light.h library is readily available for Arduino IDE which allows easy, light and fast communication between ESP32 and MPU6050. The library retrieves accelerometer and gyroscope measurements from the sensor which is processed using a complementary filter to provide an estimation of angles in X-axis and Y-axis with respect to the horizontal frame. The library accesses acceleration, gyroscopic and temperature measurements for this purpose. This library integrates angular velocity to calculate angles. But in MPU6050, any little bias in angular velocity is integrated which causes severe drift in angles which is tackled by complementary filter that merges both accelerometer and gyroscope measurements.

In this library, the gyroscope data is weighted with a factor of 0.98, while the accelerometer data is subject to a complementary factor of 0.02 which can be fine-tuned by the user. The accurate angle estimation is ensured by maintaining a short loop delay between successive calls to the "*update()*" function. This delay is important for preserving the approximation used in the numerical integration of angular velocity. This library works perfectly for relatively small linear accelerations, with gravity being predominant force. The X, Y and Z angles are obtained in degree from this library.

The acceleration data obtained can be used to calculate the pitch and roll angles using equations 4-4 and 4-5 respectively and Yaw angle is obtained using equation 4-6

4.3.3 Data Transmission and Server-side Processing

Since the ESP32 works on a synchronous processing loop, one event taking more time to execute blocks the other events. The ESP 32 already has the time cost of time multiplexing while reading analog input from hall effect sensors. So, processing the code inside the ESP32 increases the time delay between the two subsequent readings even more. So, in order to avoid such delays, all the burden of processing the data is shifted to the server. The ESP32 only collects the data and sends the data to the server.

ESP 32 has an inbuilt Wi-Fi module which will allows us to communicate to the server wirelessly via Wi-Fi. To communicate with the server easily, we can use the HTTP requests or WebSocket. Since the simulation happens in real time, the delays need to be reduced as much as possible. So, a protocol with minimum time delay is to be selected.

HTTP requests and responses create connections, perform 3-way handshaking between the server and client and then terminate the connection for each request. For normal web applications this approach works perfectly, but for real time applications where time is crucial, these delays induce a high cost in performance. In this scenario, web-socket helps to remove these time delays. Web sockets create the connection between the server and client once and do not terminate the connection until one of them wants to terminate the connection. Between the connection and termination, a full duplex communication channel is created that allows fast data sharing in real time.

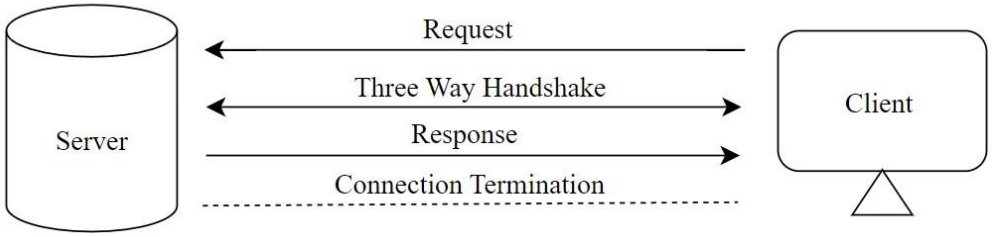


Figure 4-14: HTTP Connection

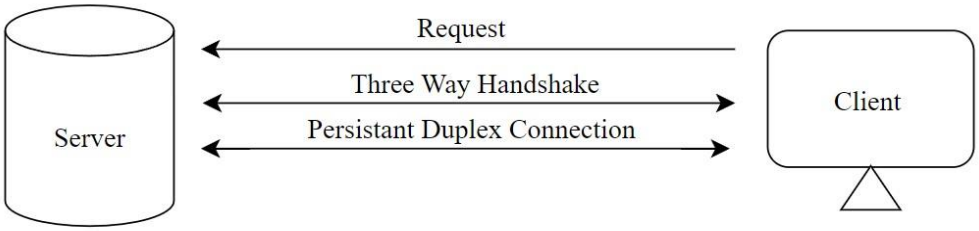


Figure 4-15: WebSocket Connection

This step completes in four different stages. They are:

Creation of WebSocket Server

The WebSocket Server is developed using Node.js, which is a cross platform JavaScript Runtime Environment that allows the JavaScript code to run outside of the web browser.

It uses the chrome's V8 engine with additional capabilities like accessing file-system, etc. to run JavaScript outside of the browser in a highly performant manner. Although the Node application runs on a single process thread, it extensively utilizes asynchronous programming to achieve non-blocking code and can handle numerous clients simultaneously.

To create the WebSocket server, the "*ws*" library of Node.js is used. It is an external node library that provides a low-level API for fast and efficient WebSocket Communication. It is a thoroughly tested library that allows to create web sockets server and clients. Creating server with this library provides a WebSocket server object with a bunch of methods that registers callback functions for handling different events like

connection events, message event from clients, etc. and also to easily recognize and send the message to specific clients.

Creation of WebSocket Client in ESP 32 and Data transmission

To transmit data to the server first, the ESP 32 Wi-Fi development board needs to be connected to a Wi-Fi network. For this purpose, the "Wifi.h" library available for Arduino IDE is used. This library provides a function to find the available Wi-Fi networks and connect to the Wi-Fi network.

After that, the WebSocket client needs to be set-up. For this purpose, the "WebSocketsClients.h" library by Markus Sattler is used. This library provides a direct WebSocket client object which registers an event handler to handle different web socket events like connection, disconnection, ping, pong, text message received event, binary message received event, error events and many more. The library also provides functionality to reconnect to the network if the connection with the server is disrupted by any means. The client object also provides a function to send the message to the server at any point of time after set-up.

After the Wi-Fi and WebSocket client is set-up, the data from the sensors are formatted in a valid JSON format. C++, which is the language used to write code in Arduino IDE, natively has no support for JSON data type, to work with JSON objects i.e. creating, serializing and deserializing the JSON object, a library is required. 'ArduinoJSON' by Benoit Blanchon is a very simple and efficient library that allows to achieve all of the above-mentioned functionality. This library is used to create and serialize JSON object. The data from all the sensors are acquired and stored in variables as described in the Data Acquisition section. These data are now used to create key value pair in a JSON object where key specifies the data label and value specifies the actual data value.

The JSON formatted data looks like this:


```
{
  "indexTop": "1023,",
  "indexMid": "1034",
```

```
  "indexBottom": "1103",
  "middleTop": "1203",
  "middleMid": "1224",
  "middleBottom": "1913",
  "ringTop": "1205",
  "ringMid": "1224",
  "ringBottom": "1203",
  "pinkyTop": "1031",
  "pinkyMid": "1033",
  "pinkyBottom": "1092",
  "thumbTop": "1501",
  "thumbBottom": "1008",
  "thumbIndex": "1033",
  "indexMiddle": "1092",
  "middleRing": "1501",
  "ringPinkey": "1008",
  "xAngle": "37",
  "yAngle": "63",
  "zAngle": "12"
}
```


Processing of Raw data from ESP 32 in Server

The data values from ESP 32 are just the analog readings from ADC of ESP 32. These data need to be processed to form the angle data that describe the bend of each finger joint. This mapping of the reading from ADC to angle data can be performed in various ways.

For this project, two simple methods are chosen.

1. Linear Mapping
2. Piecewise Linear Mapping

Linear Mapping

This type of mapping is the simplest of all the mapping techniques. In this technique, the magnetic field of the magnet is assumed to be changing linearly with the bending angle of the finger. This method ignores the rotational motion of the magnet. So, we just define the minimum and maximum reading when the bend of the bend is at 0 degree and 90 degrees. Then the equation of the straight line is calculated by using a two-point form of straight line.

The mapping function thus developed is:

$$Angle = \frac{90 - 0}{NDR - ZDR} * (RD - ZDR)$$

Where,
NDR = Raw data reading at ninety degrees
ZDR = Raw data reading at zero degree
RD = Raw data to be mapped
Angle = Bending angle to be calculated.

This method is accurate if the output voltage is varying linearly with respect to the magnetic field during bending of the finger. In case of mapping the separation between the adjacent fingers, there is almost linear movement between magnet and sensor. So, this method is appropriate for that case.

In the attachment for this project, when there is a bend there are both rotational and translational motions of the magnet. So, the actual curve is also not linear, as shown in Figure 3-1. Thus, this mapping technique encounters heavy error for calculating the bending angle of the finger.

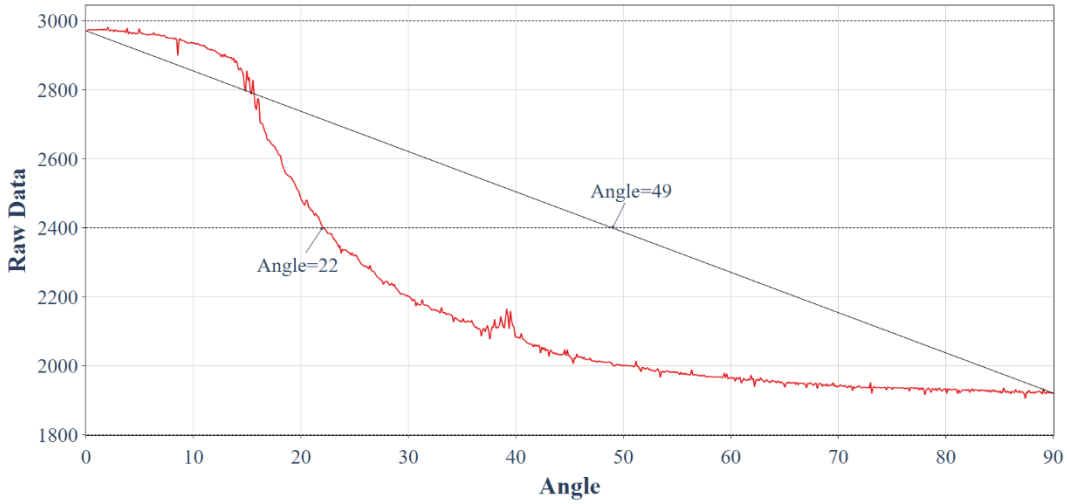


Figure 4-16: Raw Data vs Angle (Linear Mapping)

Figure 4-16 shows the linear mapped plot for the approximate curve of raw data vs bending angle data. When a projection from a value of 2400 of raw data is made into the actual plot and the Linear Mapping plot, the two angles obtained are 22 and 49 degrees respectively as shown in figure. This clearly shows an error of 27 degrees between the actual reading and calculated angle data. So, this method is less feasible for accurate mapping of the bending angle of the phalanges.

Despite the above facts, this type of mapping can be used for cases like rapid simulation like game animation where the angle data accuracy is less concerning but the fast processing of data is of concern.

Piecewise Linear Mapping

This mapping is an upgrade to the linear mapping method and thus is resource consuming than the linear mapping method but it provides a very high level of accuracy compared to that of linear mapping. This method makes use of the fact that although the whole curve doesn't follow the linear pattern, the parts of the curve tend to show a linear pattern in multiple pieces. So, these parts can be individually mapped linearly to get a nearest approximation to the actual curve.

A lookup table is established by recording the angles with their reading from the sensors manually where the curve tends to have a very different slope as shown in Table 5-1. Then, the raw data is searched in the table and the two data points having just greater

and just smaller raw data (from the table) than the raw data to be mapped are selected. And from those two points a linear equation is created like the linear Mapping method. And the angle value is thus calculated from that equation.

Table 4-2: Look Up Table

Raw Data	Angle Data
2970	0
2850	14
2326	23
2059	42
1961	62
1961	90

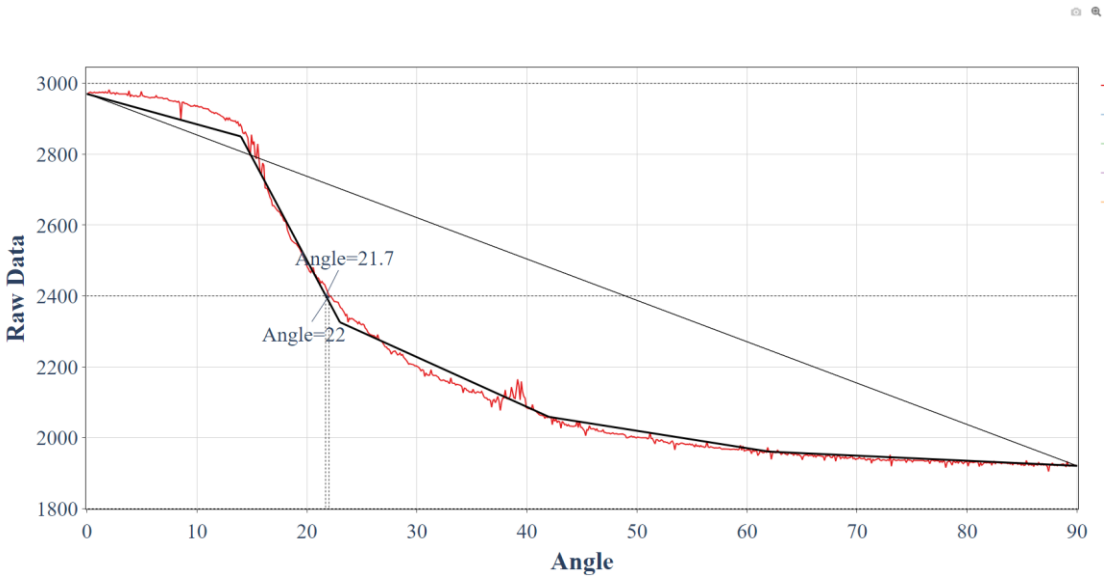


Figure 4-17: Raw Data vs Angle (Piecewise Linear Mapping)

Table 4-2 shows the Piecewise Linear Mapping plot for the approximate curve of raw data vs bending angle data. The line pieces are created from each of two adjacent points shown in the lookup table (Table 5-1). When a projection from a value of around 2326 of raw data into the actual plot and the linear plot, the two angles obtained are 23 and 23.74 degrees respectively as shown in figure. This clearly shows an error of 0.74

degrees between the actual reading and angle data which is a miniscule error. Furthermore, the accuracy of this method can be further improved by increasing the number of line segments/pieces of the mapping plot.

Transmission of Processed Data to Application

When the server establishes connection with the unity application, the web socket instance is created for the application in the server. Then, when the server finishes processing the data, it directly sends the data in JSON format with actual bend angles of fingers and rotational data of hand. These data will have the same format as that of the data from ESP32 but the only difference is that these data are processed and ready to use by the application.

4.3.4 Real Time Simulation

The real time simulation of virtual hand is rendered in Unity engine, which uses C# programming language for scripting the movement of hand.

Model generated by blender is of ".blend" format. Unity engine automatically imports the '.blend' file as '.fbx' file, since it only supports the '.fbx' format for 3D models [8]. Then it is placed in the scene to be rendered.

Moving the finger involves rotating the fingers of the hand model along a coordinate axis. Unity engine library for C# provides a rotation function for rotating the object in the scene.

The data sent from the Node.js server is received by the client application.

For this purpose, WebSocket is used for listening for incoming data from the server in real time. Websocket-sharp library is suitable for this project [7]. A WebSocket object is built using this library. The URL of the server is passed on to this object and a connection is established.

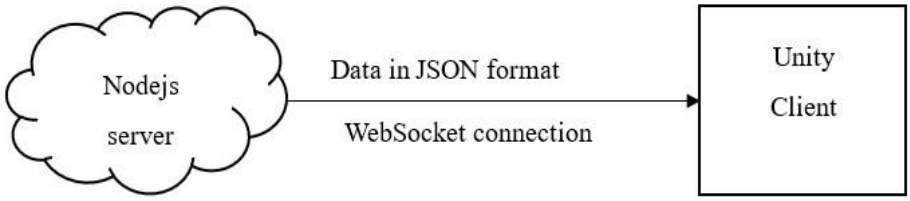


Figure 4-18: Server Client Connection

The message sent from the server triggers an event called *WebSocket.OnMessage*. The received data is then be parsed and mapped into the hand model for movement.

5. IMPLEMENTATION DETAILS

5.1 Hardware Design

The Hardware design section of the project is further divided into three parts: Glove Model Design, Circuit Design and Circuit Case Design. The Model Design includes designing and implementation of glove with sensors and magnets. Whereas, the circuit design includes all the circuits that have been designed and their implementation. The Circuit Case Design encapsulates the circuit which makes the design modular.

5.1.1 Glove Design

A modified glove design of GloveOne by Brian Cera was constructed first for a single finger which was Index finger and the same design and construction was repeated for all other four fingers based on the attachment for single joint.

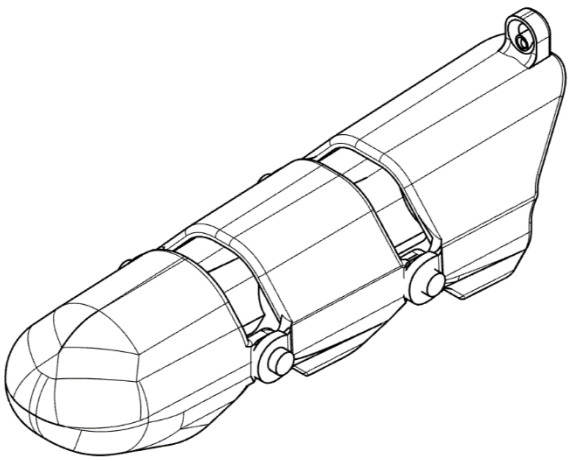


Figure 5-1: Sketch of Glove Design (Index Finger)

The attachment houses both the 49E Hall Effect Sensor and N35 Neodymium Magnet and defines the basic functionality of each joint of the glove. This provides the base for the glove design. Both the sensor and magnet are placed close to each other so that when the finger is straight the Sensor touches the Magnet and when the finger is completely bent, Sensor moves away from the magnet. In the attachment shown in Figure 4-11, the point of rotation is fixed which doesn't allow the position of magnet and sensor to shift which affects the sensor data consistency.

This design fixes the point of rotation for each joint and also limits the rotation of each joint to 90°. This design is divided into 3 sections for 3 joints of the finger and a fulcrum is defined for DIP, PIP and MCP as shown in Figure 5-2.

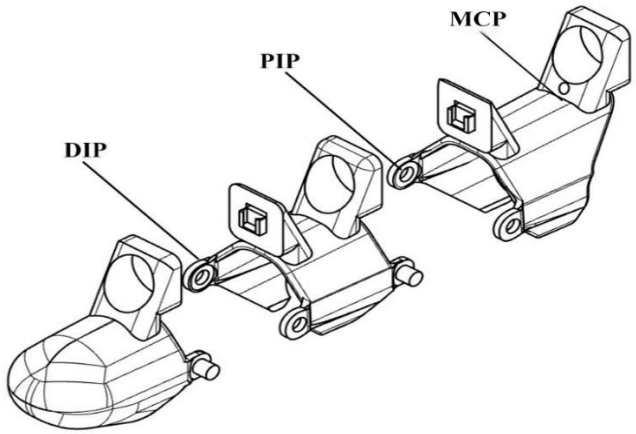


Figure 5-2: Glove Segment Detached (Index Finger)

To design a proper functional glove and attach all designed finger sleeves, a back plate was needed which didn't interfere with the knuckle joint or metacarpophalangeal joints of the hand and houses sensors for these joints. So, the back plate is designed using the design in [ref to glove one] and making the required modifications to it to properly house the sensors and easily attach all fingers with flat plane over the back of the hand to house the circuits.

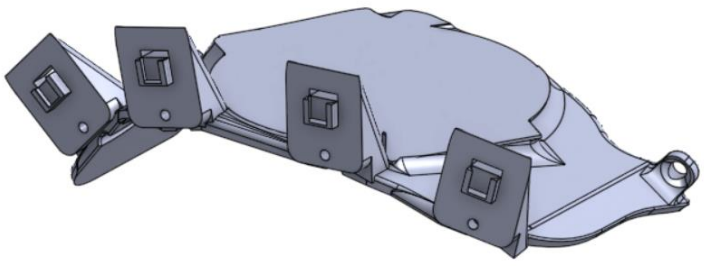


Figure 5-3: Backplate - Front with Sensor Attachments for Four Fingers

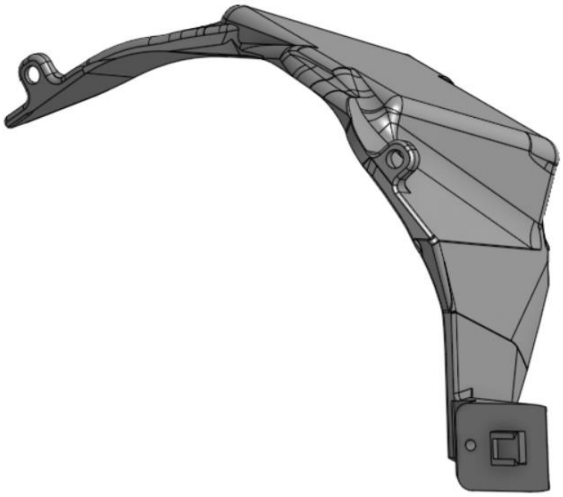


Figure 5-4: Backplate - Back with Sensor Attachment for Thumb

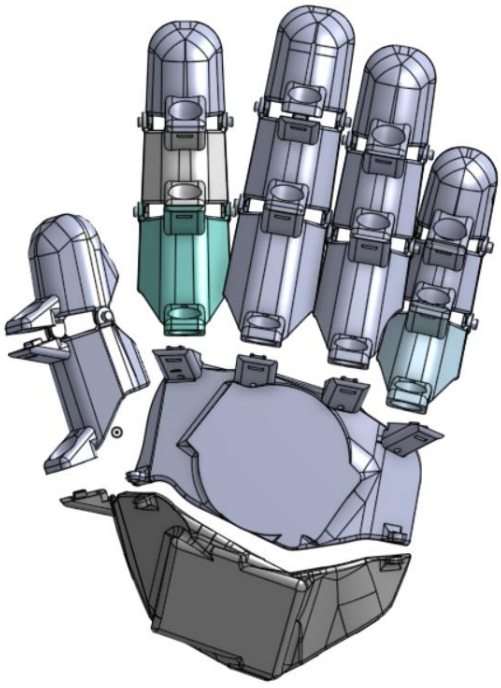


Figure 5-5: Final Glove Design

The designed glove is 3D printed and assembled considering complex design, precision and sturdy construction. The sleeve for each finger is designed with a global average of finger measurements to fit the glove to as many people as possible and properly define the joints. In this design both the sensor and magnet are placed close to each other to minimize the noise from other magnets. The N35 Neodymium magnet used here is of 10 mm diameter for enough magnetic field.

5.1.2 Circuit Design

Power Supply Unit Design

Power supply unit consists of a 7.4V Lithium Polymer (Li-Po) battery, 1 AMS117 & 2 LM7805 voltage regulators.

The 7.4V supply from the battery is supplied to the input pins of both LM7805 voltage regulators which provide us with 5V regulated output. The output from one of the regulators is fed to the 'V_{in}' pin of the ESP32 to power it. The output of the other regulator is fed to the input pin of the AMS1117 Voltage regulator which thus provides 3.3V regulated output.

To power the 14 Hall Effect sensors, mpu6050 sensor and analog multiplexer the output of 3.3V from the AMS1117 regulator is used.

The circuit diagram of the power supply is shown in Figure 9-2.

Overall Circuit Connections

Pins of the analog multiplexer are connected to the output pins of the sensors. SIG pin of analog multiplexer is connected to D34 (which supports analog-to-digital conversion). Selection pins S3, S2, S1, and S0 of the multiplexer are connected to D17, D16, D4, and D2 respectively. The V_{cc} of the multiplexer is connected to output of AMS1117 to provide power to the multiplexer. Enable pin of the multiplexer is connected to the GND pin because it is an active low pin.

V_{cc} of the hall effect sensor is connected to 3.3V output of AMS1117. GND pins of hall effect is connected to GND. The Vout pins of 12 hall effect sensors are connected to the pins C4 to C15 of analog multiplexer whereas the 2 pins are connected to the pins 32 and 33 of ESP32.

SDA and SCL pins of MPU6050 are connected to the D21 and D22 pin of ESP32 respectively which are the SDA and SCL pins of ESP32. V_{cc} of MPU 6050 is connected to 3.3V and GND pin is connected to GND of AMS1117.

The RGB LED being in common cathode configuration, its common cathode is connected to the GND of ESP32 and the three anodes for red, green and blue are connected to the D13, D12 and D14 pins of ESP32 respectively.

LED is powered via 3.3V supply of ESP32 and to power ESP32 the output of 5V from LM7805 is connected to the Vin pin of the ESP32 with common ground.

The overall circuit diagram is shown in Figure 9-1.

5.1.3 Circuit Case Design

A case was required to house all the circuitry to contain it in a managed way. In order to make the whole circuit less messy and easier to manage, the whole circuit was divided into two parts: Top and Bottom part on PCB to stack it together in the case. Top board containing ESP32, indicator LED and a switch and Bottom board containing multiplexer, MPU6050 and voltage regulator.

After taking precise measurements of each part of the circuit; Top and Bottom board a case was designed in such a way that Top board sits right above the bottom board with enough solder clearance below the circuit and enough height clearance as well.

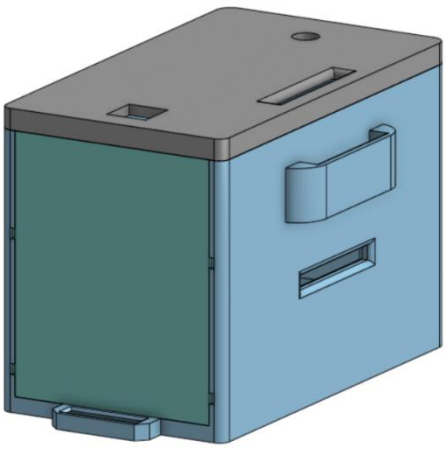


Figure 5-6: Circuit Case Design

Whole Case is divided in to three parts: Circuit Holder, Sliding Door and Cap. The Circuit Holder holds the circuit and contains all the wires, while the Sliding Door closes the box from all four sides and the box is capped off by a cap. The Circuit Holder has

wide grooves to slide the circuit board, passthrough holes for wires, 3wire management passthrough on the back and strap holes to tie it around the hand. The design details can be seen in Appendix D.

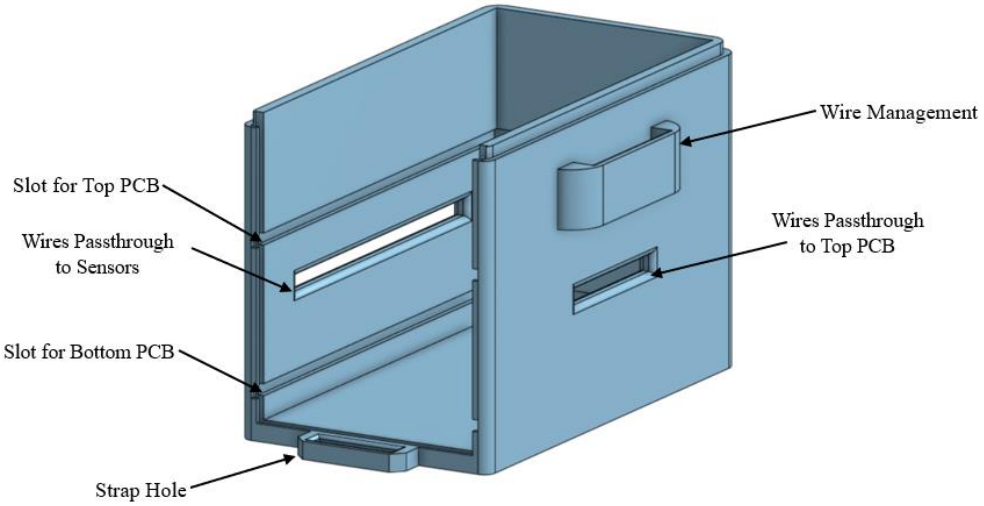


Figure 5-7: Circuit Holder

To facilitate the modular design and rigidity of the case, a sliding door was designed to close off the side from which the circuit boards can slide in and out. For the sliding door, a groove was cut out on all three sides of the Circuit Holding part to fit it perfectly.

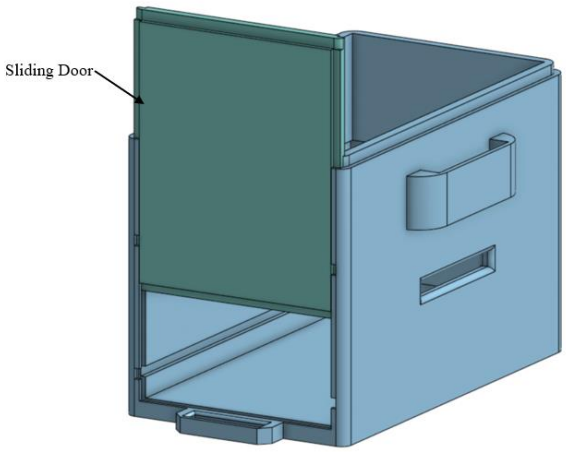


Figure 5-8: Sliding Door Mechanism

After the box that houses the circuit board, the cap was designed to close it with a place for a power switch and an indicator LED. For this, a cut out was made to place the LED and the switch in a convenient place.

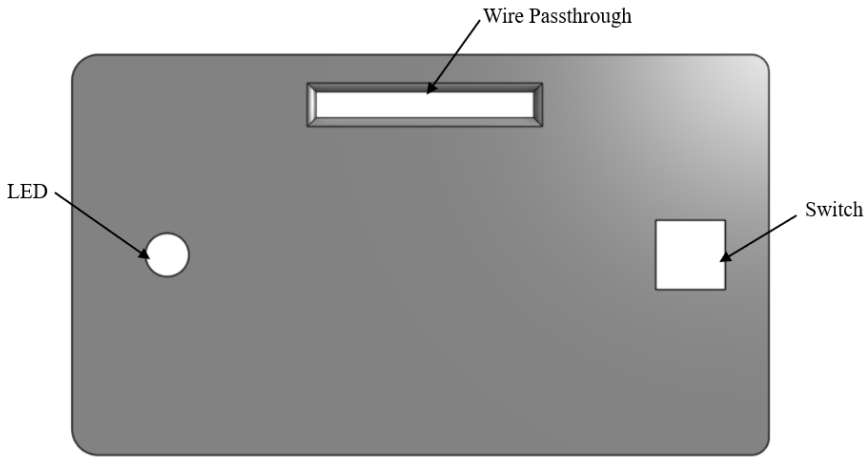


Figure 5-9: Circuit Case Cap

All the parts were combined for complete modular circuit case and then 3D printed using PLA (Polylactic Acid) plastic filament.

5.2 Data Acquisition

5.2.1 Taking Data from Hall Effect Sensor

Data from the hall effect sensors are read through the analog multiplexer. To set the multiplexer pins '*setMultiplexerPin*' function is defined and used. Four arguments are passed to this function which represent the 4-bit binary data representing the pin number from 0 to 15. Selection lines S0, S1, S2, S3 are set high or low depending on value '0' or '1' passed as an argument.

For example, if four 0's are passed as an argument then the analog data from multiplexer pin C0 is selected. Similarly, 1010 selects the C10 pin of multiplexer. To read the data from Hall Effect Sensors, '*readHallEffecrSensorData*' function is defined. This function calls '*setMultiplexerPin*' to select particular channel then '*analogRead*' function from Arduino is called which returns the sensor value from the hall effect sensors. The data from all the sensors are read sequentially one after another by selecting the appropriate pin of multiplexer in which the hall effect sensors are connected. The data from two hall effect sensors are directly read by using *'analogRead*' function on pin D32 and D33 of ESP32.

5.2.2 Taking Data from MPU6050

To calibrate the MPU6050 '*clacoffsets'* function from "MPU6050_light" library is used. 'update' function from the library is used to read the acceleration and angle of rotation from the sensor. '*getAngleX'*, '*getAngleY'*, '*getAngleZ'* are the functions from the library used to calculate roll, pitch and yaw angle respectively from the data obtained from MPU6050. Since gyroscope returns angular velocity, so to calculate angle of rotation from angular velocity exact time is required. To manage the timing to calculating angle, a variable named timer is used.

5.3 Data Transmission and Server-Side Processing

5.3.1 Creation of Web Socket Server

The server is set up to run locally. The server is hosted on the port '3000' of the host operating system. The server contains only one end point for the WebSocket connection on the root path '/'.

When the server is started, the server listens to the WebSocket connection events on the specified port. The server can identify the clients, i.e. if it is ESP32 or the Unity Application based on the API key received as a query parameter on the connection request. When a connection request reaches the server, the server extracts the query parameter and checks if the API key sent by the client matches with the known API key.

If the API key doesn't match the known API key, the server instantly disconnects to the client sending an error message to the client. If a known API key is received, then the server provides a name to the client based on the API key and persists the connection with that client. To identify and name the clients two different API keys are used. Client with API key 'a1b2-c3d4-e5f6-g7h8-i9' is identified as ESP32 and the client with API key 'A1B2-C3D4-E5F6-G7H8-I9' is identified as the Unity application. The server stores each instance of the web socket connection separately.

5.3.2 Creation of Web Socket Client in ESP32 and Data transmission

Configuring the RGB Led as status indicator

To get the status of the connections in ESP32, a RGB led is used whose color indicates the different status of the ESP32. "RGBLed.h" library is used to control the color of the LED easily. A led instance is created by providing the pin numbers of the ESP32 pins in which the LED anodes are connected and the configuration of the LED i.e. "Common Cathode" configuration. Then to set the color of the LED, the '*setColor*' method of the object is used. When the ESP32 boots and is not connected to WI-FI, the RGB Led is set to red color. After connecting to the Wi-Fi, the color of LED is set to yellow indicating that the ESP32 now has WI-FI connection. If MPU6050 sensor is not, the LED is set to magenta color. After MPU6050 is found and calibration is done, the color of LED is set to blue indicating that there is no connection with the server.

WI-FI Connection Set Up

The ESP32 stores the Wi-Fi SSID and password in its program code and uses those credentials to log in to the Wi-Fi using the "Wi-Fi" object from the "Wifi.h" library.

Connection with Server

The web socket client for ESP32 is created using "WebSocketsClient.h" by instantiating a WebSocket client object from the class "WebSocketsClient". The client then uses its '*begin*' method to set up the server IP address, port and path of the endpoint that the client will use to connect to the server.

Then it registers an event handler function via its '*onEvent*' method. This method accepts a void callback function with three parameters: event-type, payload and length. Based on the type of event that occurs, different tasks functionalities are defined. For this project, only three events are crucial. They are:

a. Connection Event

This event is triggered when the connection is established with the server. When this event is triggered, the ESP32 prints the connected message to the serial monitor and sets the '*clientConnected*' Boolean to true. And then the LED light is set to green color.

b. Disconnection Event

This event is triggered when the connection with the server is terminated by any cause. When this event is triggered, the ESP32 prints the disconnected message to the serial monitor and sets the '*clientConnected*' Boolean to false. And then the LED light is set to blue color.

C. MESSAGE EVENT

This event is triggered when any message is received from the server. When this event is triggered, the ESP32 prints the received message to the serial monitor. The code snippet for this event handler is shown in Appendix F-2.

After the event handler is registered, then the reconnection interval of 5 seconds is set using the client's '*setReconnectInterval*' method.

Since the client needs to listen and respond continuously, the loop method of the client is called inside the loop method of the code.

Creating and Serializing JSON object

An object of type '*JsonDocument*' named '*rawJson*' is instantiated. When the data from all sensors are collected, the values in object are filled using the syntax as:

rawJson["indexTop"] = indexTopData;

Then the raw data is serialized using the '*serializeJson*' function from the library to convert it into a string type as shown in Appendix F-3.

The serialized JSON data is then sent to the server via the client's '*sendTXT*' method.

5.3.3 Processing of Raw Data from ESP 32 in Server

The server receives the JSON object of raw data from the ESP32 client and then maps the raw data from the hall effect sensors into the angle data using Linear Mapping technique. The mapped data is then sent to the Unity Application client. Since the server instantly maps the raw data from the ESP32 client and sends the data to the Unity

Application client, the data received by the Unity Application is directly dependent on the rate of data sent by the ESP32 client.

Since the raw data from the hall effect sensors are the output of the 12-bit ADC of ESP32, the data are in the range of 0 to 4096 which represent the voltage levels. To convert this data to angle data, linear mapping is used. When the finger is not bent, i.e. the angle is 0 degree, the sensor is closest to the magnet. So, we took data from the hall effect sensor by bringing the magnet to touch the sensor. The data is stored as '*zeroDegreeReading*'. Similarly, when the finger is bent to 90 degrees, the sensor is farthest to the magnet so that the magnetic field of the magnet is barely felt by the sensor. The data is stored as 'ninetyDegreeReading'.

Then a mapping equation was developed using the two point form of straight line.
i.e.

$$y - y1 = \frac{y2 - y1}{x2 - x2} * (x - x1)$$

where, y = mappled angle

x = raw data

From this equation, each raw value is mapped to meaningful angle data as seen from Appendix I. And thus, obtained data is packed into JSON format and then sent to the Unity Application via it's WebSocket channel instance's '*send*' method.

The linear mapping method is implemented in server code as shown in Appendix F-4.

Similarly, the mapping table is defined in code as shown in Appendix F-5. The piecewise linear mapping is done as shown in Appendix F-6.

5.4 Realtime Simulation

5.4.1 Rigging Model in Blender

Blender provides a wide range of tools for tempering a 3D model. In order for the hand model to be simulated properly like a real-world hand, it needs to be rigged with bones.

Rigging the hand model is like providing a skeleton to flesh. For this purpose, blender uses “armature”. Armature has an origin, position and a scale factor. The base element of armature is called bone. Each bone is posed on another forming a parent-child structure.

Figure 5-10: Rigged Hand Model in Blender

5.4.2 Development of Scene in Unity

The major components of scene for the simulation are Hand Model, UI and Cameras. Hand Model is imported from blender, whereas UI and Camera are built in Unity editor. There are multiple cameras fitted throughout the scene.

The scene is divided into two segments: Main Menu and Gameplay. One of the cameras, called the Canvas Camera is pointed towards the Main Menu when the application opens, whereas other cameras: First Person View Camera, Side View Camera and Top View Camera are pointed towards the hand model.

For changing the state of main menu, there are three buttons available: Play, About and Quit. The play button proceeds to game scene, the About button opens up a webpage containing details of the makers of the project whereas the Quit button will exit the application. The script containing the logic for these buttons is attached to the Main Menu component of the “*Canvas*” in the scene. From there, the button objects are

referenced and are associated with the methods, which contain the logic for these buttons.

Behind the scene, the main menu also attempts to connect to the server asynchronously and displays the state of connection to the server on frontend. The play button is hidden as long as the connection to the server is not established.

Figure 5-11: Main Menu When Server Online

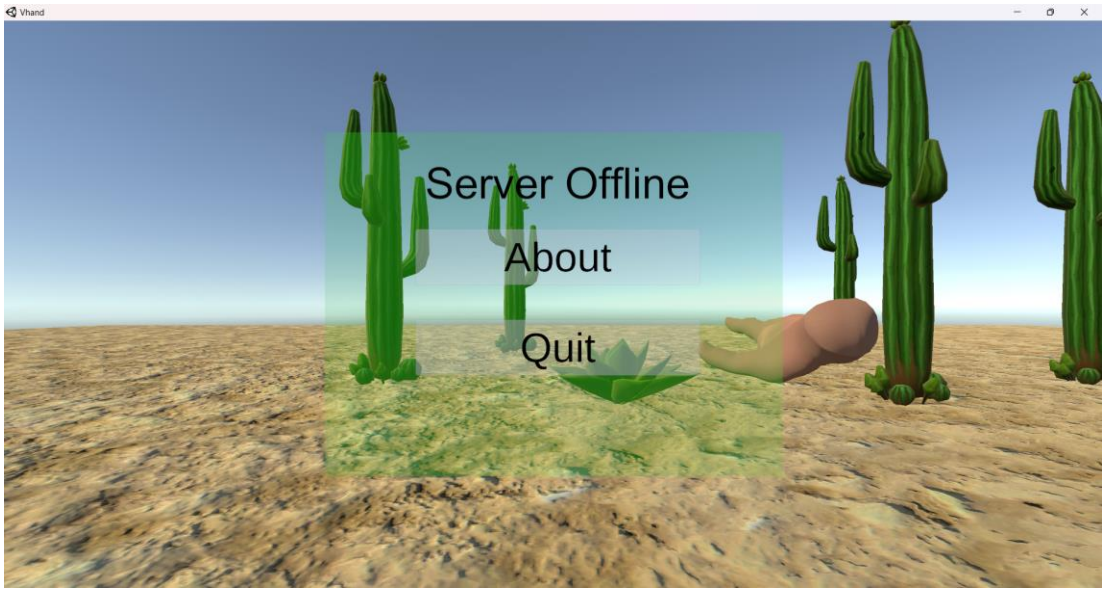


Figure 5-12: Main Menu when Server Offline

The connection to the server is checked on every frame, i.e. on each call to the *Update* function. Whenever the server is online, the WebSocket connection is established and the play button is available to be pressed.

In the actual gameplay, the data obtained from the server is mapped to the local rotation of the hand model. Unity uses a coordinate system involving Quaternions. However, using the methods provided by Unity, Euler Angles, which are much simpler than Quaternions and as the server itself sends the bend angles in Euler angles, can be used instead of Quaternions. Unity conveniently converts the Euler Angles to Quaternions.

The model is developed in such a way that each finger joint is a parent of the previous finger joint and the root parent is the wrist. So, the angles through which the finger joints are rotated are relative to the rotation of their parent finger joint, and all the finger joints are rotated relative to the wrist. Hence, the method "*transform.localEulerAngles*" is used for transforming the rotation of all the objects. The Euler Angles of each joint as well as the wrist is stored in a Vector3 type object. The x-component of this vector3 object is the one which is updated in every frame as the fingers are rotated along x-axis.

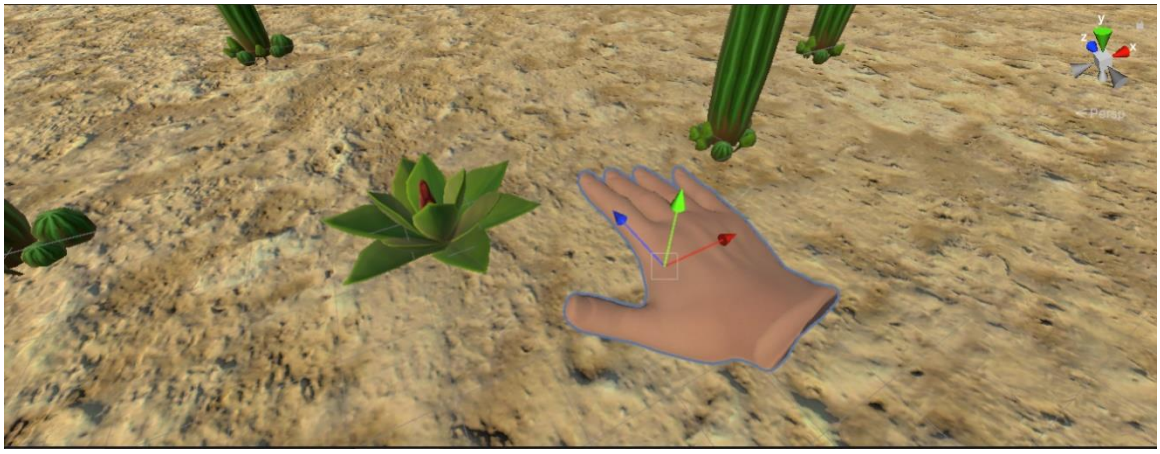


Figure 5-13: Hand Model in Unity scene

The highlighted object in Figure 6-5 is named as "Player", which is a "*gameobject*" type. The hand model and the logic associated with it are the children of the Player object. The scripts for movement of hand along with other functionalities are added as a component to the Player object.

5.4.3 Parsing of Data from Server

The server sends the data of bend angles of finger joints and wrist in JSON format. This data comes through a WebSocket connection subscribed by the application whenever the application requests data by sending a message to the server. Whenever the server sends JSON data, it is received by the application in serialized form, which means the key and value pairs are in string. This data needs to be deserialized for their assignment to *gameobjects*. For simplicity, a class of the hand data, resembling the keys of the JSON data, is defined which is then passed on to the deserialization method of the JSON.net library as a '*type*' Now, the deserialized data can be used as a part of the '*hand data*' type object.

This deserialized data, which is the bend in degrees, is then assigned to the respective location vectors of finger joints and wrist and the transform method of Unity is called to transform the rotation of the specified object.

5.4.4 Building the Game

Setting the targeted platform as windows and the architecture as x86_64, unity builds both of the scenes and generates binaries. The compilation of scripts, packages and assets is handled automatically by Unity.

An executable named "Vhand.exe" is generated, which is run to start the application. Running the executable requires no additional permissions except access to the firewall for connection to the server.

6. RESULTS AND ANALYSIS

The project has its 4 distinctive parts. So, the result can be properly analyzed separately in 4 parts and a final integration part.

6.1 Result of Glove Design

The glove was printed using a 3D printer as shown in Figure 9-3 of Appendix-D and tested on 2 hand subjects. The designed glove was a perfect fit for medium to large hands without any issues. As seen from the Figure 9-4 of Appendix D, the matrix boards were placed on the top inside a 3D printed compartment.

Due to the improper placement of attachment in the metacarpophalangeal joint of the glove, the orientation of the thumb attachment is off, which hinders the movement of the thumb. For other fingers, the glove provided a decent degree of freedom for each joint.

In the knuckles joint of 4 fingers other than thumb, there is not much change in distance between the sensor and magnet even when there is full 90 degrees bending of the joint at that place. This is due to the difficulty in placing a proper rotating fulcrum in that part. If a fixed fulcrum was placed it would make that joint rigid restricting the fingers to move horizontally which would bring difficulty for the person wearing the glove. So, a rubber band was used to attach the fingers to the backplate ensuring the free movement of fingers.

Apart from the knuckles, there was proper separation between the sensor and magnet when the phalanges were bent.

6.2 Result of Circuit Design

The circuit was connected in a matrix board as designed in Figure 9-4.

The circuit was functioning as expected. The ESP32 was able to connect to the Wi-Fi, detect the MPU6050 sensor, connect to the server, read data from the Hall Effect

sensors and MPU6050 sensor, format the data into JSON format and then transmit the data to the server as shown in Figure 6-1.

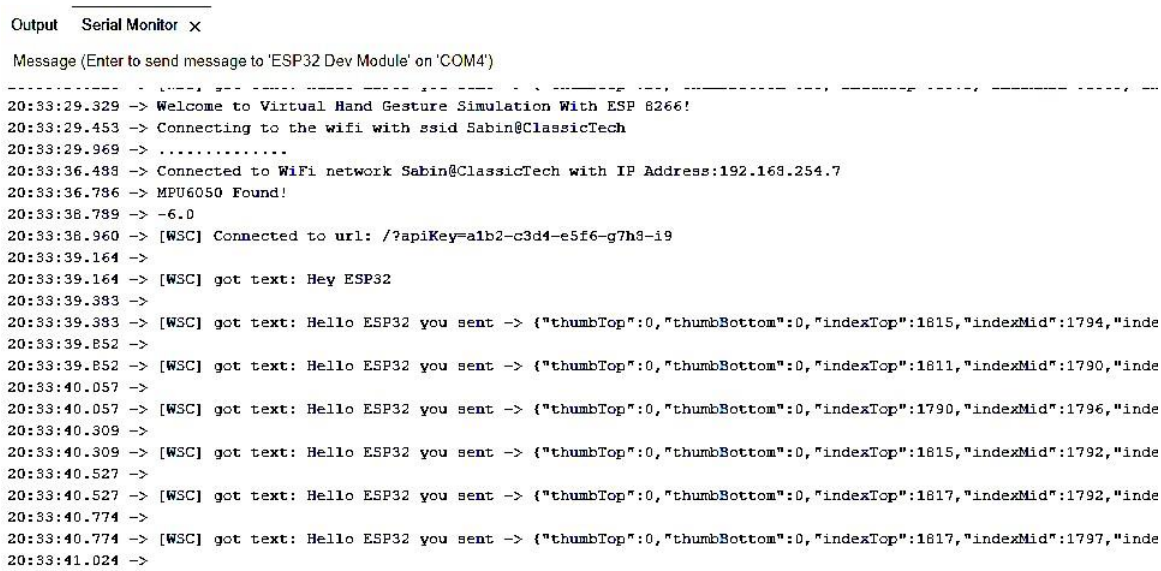


Figure 6-1: ESP32 Connection and Communication with Server

The MPU6050 sensor was able to detect the acceleration data of all three axes which was then used in the formula to calculate the angle of rotation of the hand correctly.

The reading from the two hall effect sensors were taken via the analog multiplexer which also showed the correct values when the magnet was placed near them. So, the multiplexer also worked as expected.

Finally, the data RGB led was used as indicator for different connection status of the ESP32 also indicated the status correctly i.e. red color at startup, yellow color after Wi-Fi connection, magenta color if MPU6050 is not found, blue color after MPU 6050 is found and calibrated and green color if the connection with the server is established. The blue color and green color were changing dynamically when the connection with the server was disconnected and again reconnected in the middle of data transmission too. The ESP32 also reconnected to the server by itself when the connection with the server was disrupted due to server restarting.

These results confirmed the expected working of the designed circuit and its components. The data from Hall Effect sensors were a bit fluctuating which could be

because of the noise and loose connection to the circuit. Apart from these minor issues, the circuit diagram provided expected outputs.

6.3 Result of Data Transmission and Server-side Processing.

To test the functionality of the server, a chromium browser extension called Simple WebSocket Client was used which was used to test and confirm that the server was correctly functioning.

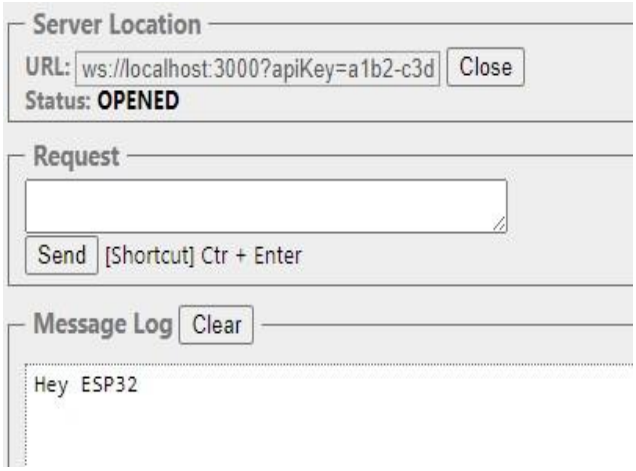


Figure 6-2: Sending Connection Request to Server by ESP32

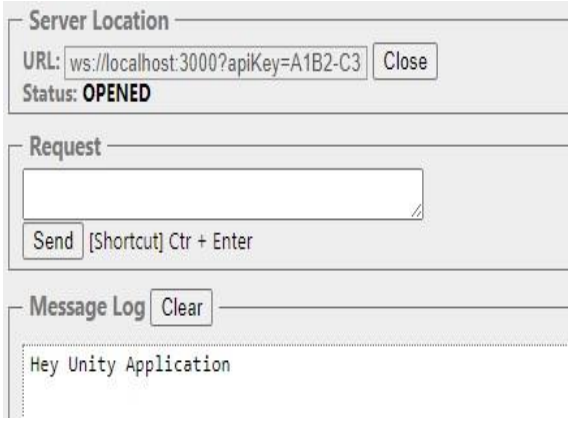


Figure 6-3: Sending connection request to Server by Unity Application

The server was successfully able to connect to both of its ESP32 and Unity Application clients and also identify them based on the API key sent by the clients in the connection request as shown in Figure 6-4.

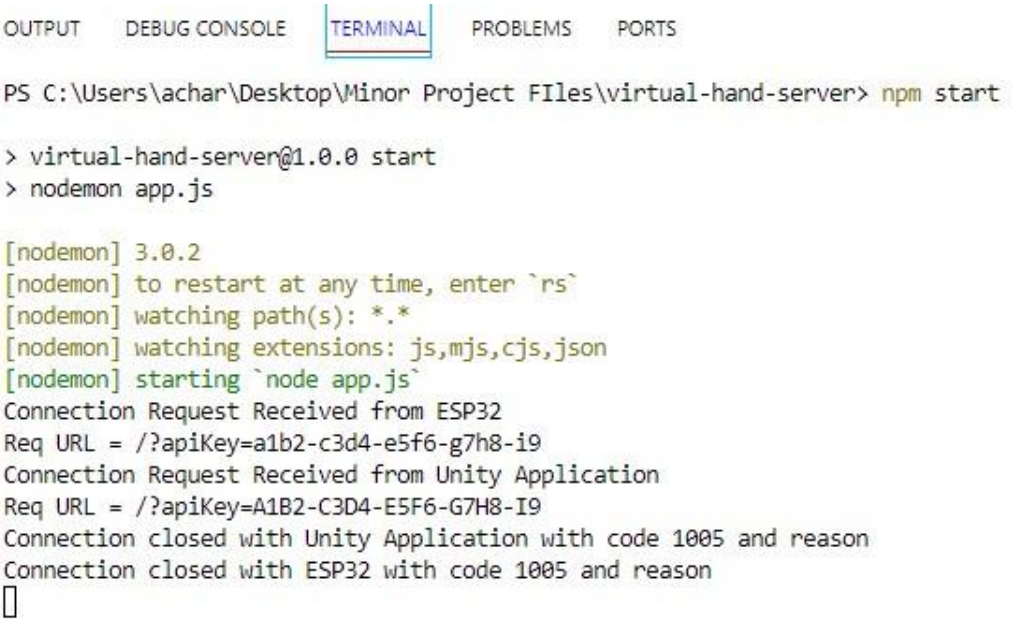


Figure 6-4: Connection Establishment and Disconnection Functionality

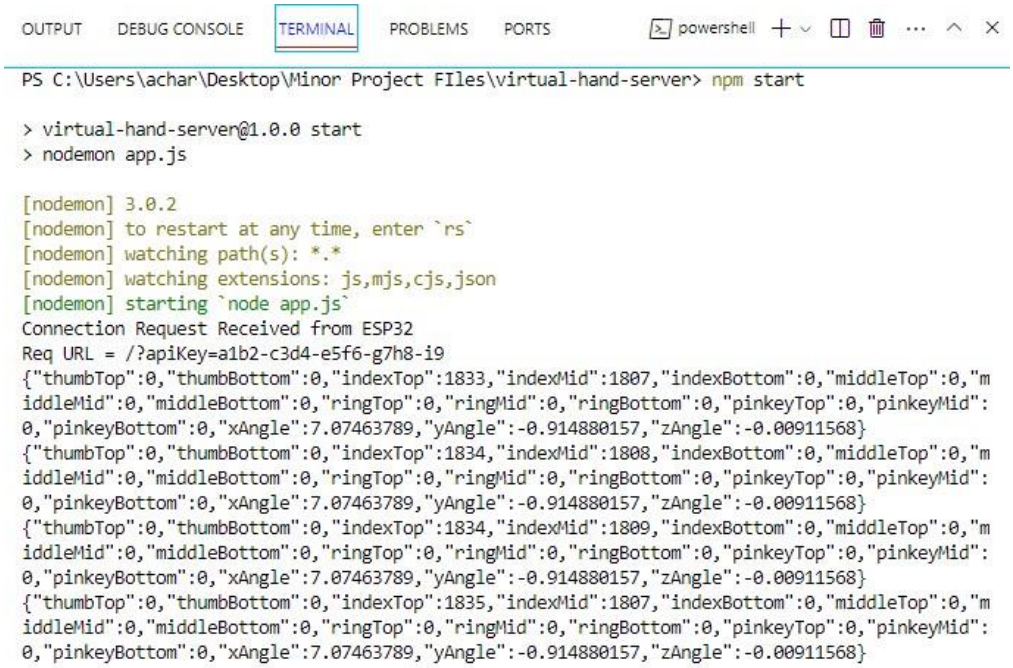


Figure 6-5: Receiving of Data from ESP32 by Server

The server was also able to receive data from the ESP32 in real time as shown in Figure 6-5.

The server was also able to provide the mapped angle data for simulation in real-time to the unity application which can be seen in the results of the whole integrated system.

6.4 Result of Unity Application Development.

The index finger joints and X-axis rotation of the wrist is mapped and rendered as shown in Figure 6-6. The testing was performed by using dummy variables in the range of 0 to 90 degrees. The connection to the server was successfully established and the status of the server was displayed accurately as well. In the *gameplay*, the hand model was rendered, and the index finger was moving along with the wrist. The finger joints and the wrist were bent from 0 to 90 degrees as per the data received from the server. They were being bent in relative to their parent object, just like the gesture of a real human hand.

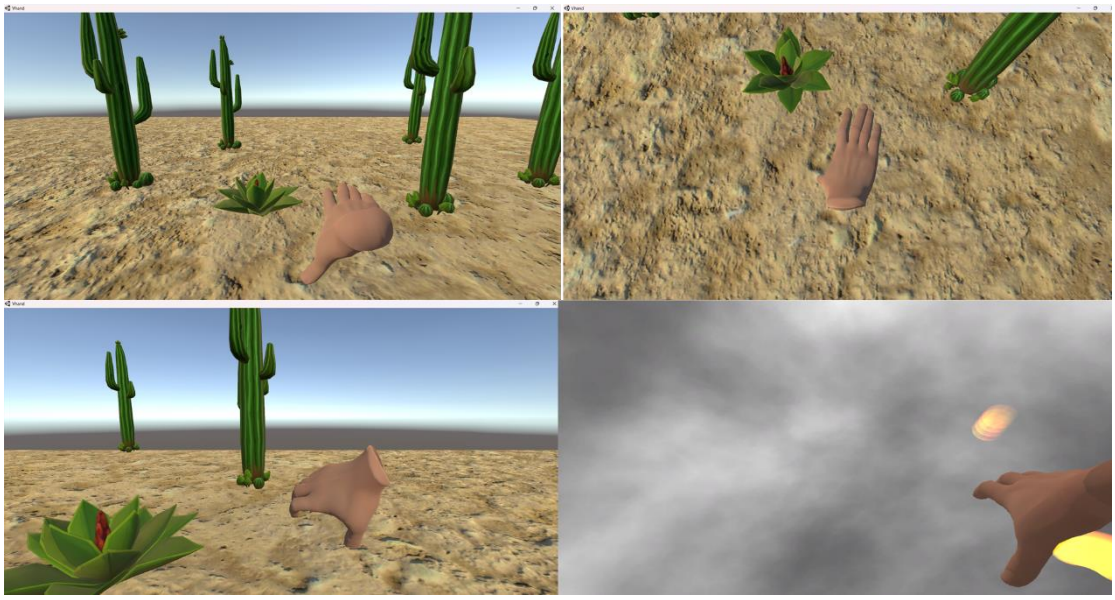


Figure 6-6: Different Views of Gameplay

Pressing 1, 2 or 3 on the keyboard changes the camera angle to First Person View, Side View and Top View as shown in Figure respectively.

6.5 Result of Integrating All the Parts

After each of the parts were tested successfully on the dummy data and testing software, all the components were integrated. First the Server was started which actively listened to the connection request. Then the ESP32 was powered up. The ESP32 performed all its functionality that takes place before connecting to the server as mentioned in the result of circuit design. Then, the ESP32 sent the connection request to server. A connection channel was successfully created and the ESP32

started reading the data from sensor and sending the JSON object containing those data to the server.

Then Unity application was opened and it also sent connection request to server and connected to the server. The server then mapped the data from the ESP32 and sent to the Unity Application. The hand model moved as per the data from the server.

The wrist bend and rotation along X, Y and Z axes were simulated accurately. A very small amount of drift was observed on the rotation along Z-Axis (i.e. yaw angle) which is due to the error in integration approach of gyroscope sensor to find the rotating angle data. The rotation around X and Y axes had no issues at all.

The simulation of fingers bent was improved by using the piecewise linear mapping but it introduced another issue. When the sensor is close to the magnet i.e. when the bend angle is in range of 0-50 degrees, the simulation is stable and has decent accuracy but when the magnet keeps moving farther to the sensor i.e. when the bend angle is in the range of 50-90 degrees, the bend was observed to the fluctuating highly. The cause of this fluctuation is explained by the nature of the sensor data as shown in Figure 3-1. From this figure, it is clearly seen that from 0 to 50 degree, the slope is very high so the disturbance occurred due to the analog multiplexer's sampling limitation, it couldn't provide a noticeable deflection in the angle data leading stable simulation. But the slope of data is extremely low from 50 degrees to 90 degrees. So, a small deflection in raw data can cause the mapped angle to be changed drastically. Since the analog multiplexer causes the data from its pin to fluctuate in absence of proper delay, the raw data fluctuates by 50-80 points. This fluctuation is unavoidable while using the analog multiplexer to get real-time data. This fluctuation in raw data by 50-80 points brings drastic change in mapped angle output in the region with low slope causing the fluctuation more prominent in that region.

Due to lack of proper separation between the sensor and magnet in the knuckle joint and lack of fixed fulcrum due to glove design limitation, the angle data in that part was highly fluctuating and inaccurate. Also, some fluctuations were seen due to the interference caused due to the magnets of adjacent fingers

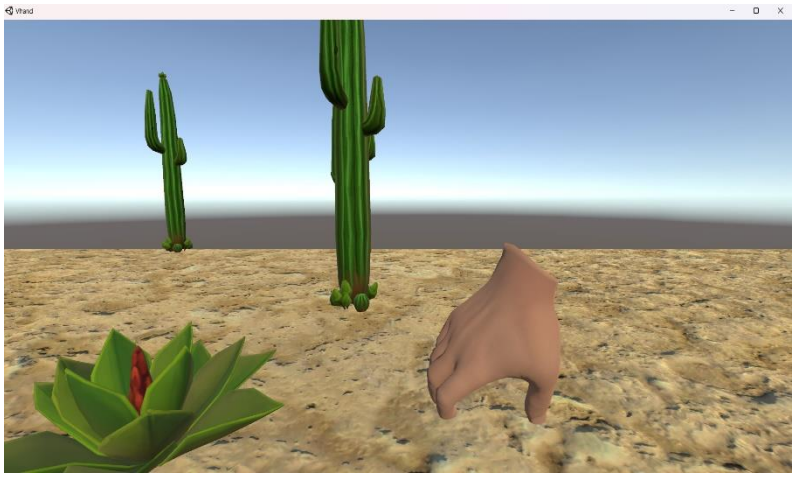


Figure 6-7: Finger Bend and Wrist Bend after Integration

A little bit of wiggly animation was seen due to fluctuating input reading from the hall effect sensors.

7. FUTURE ENHANCEMENTS

- The data acquired from the Hall effect sensor is not stable. Either a hardware solution like including a low pass filter or a software solution to this problem can be used.
- Abduction and adduction angles can be mapped for greater immersive experience.

- Instead of using these fixed mathematical equations to map the raw data to angle data, machine learning models can be implemented which can be trained on a vast amount of data to get the accurate mapping of raw data. The interference would also be greatly reduced with this approach.

- As most of the data is ruined due to improper movement of parts of gloves, a more rigid and accurate glove can be designed for stable data reading. Doing this would greatly improve the performance of the simulation of the joint in the knuckles.

- The data is fluctuating due to the limitation of the sampling of analog multiplexer, a development board with 14 pins fully supporting the ADC can be used to avoid the use of the analog multiplexer. This improves the raw data quality and the overall simulation will be stable based on the mapping.

- The design can also be made more modular and circuitry can be made more compact with dedicated chip design

8. CONCLUSION

It can be seen from the results that the objective of the project, simulation of gestures of hand using Hall Effect sensors and Gyroscope, was fulfilled.

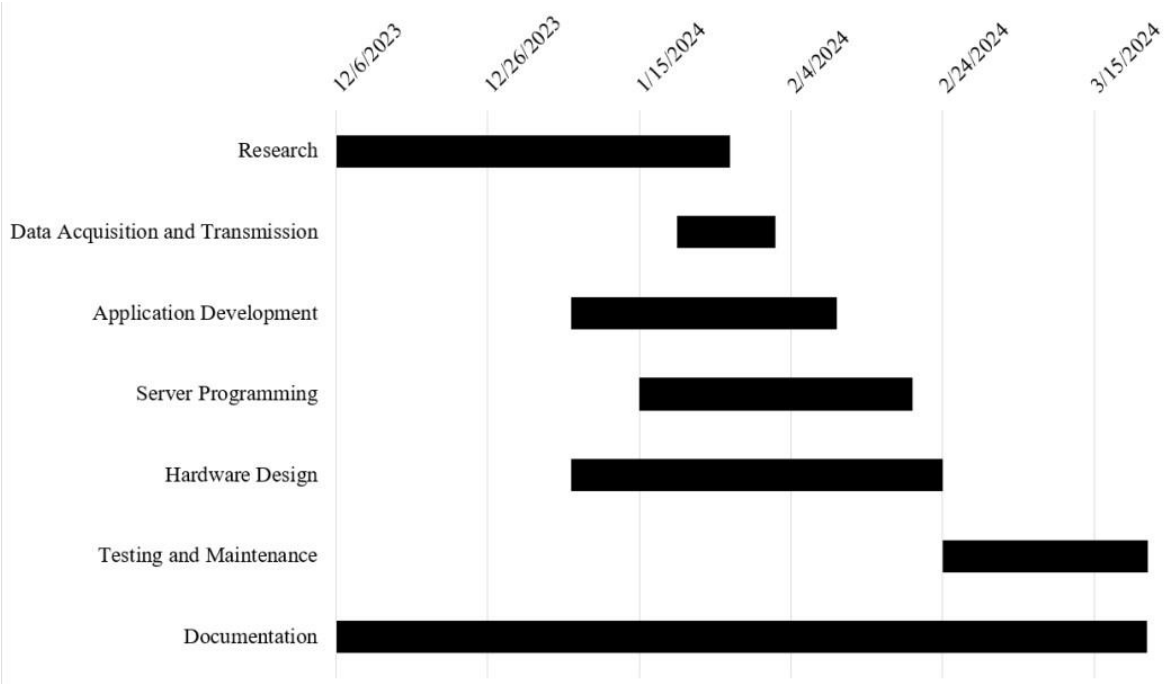
The hall effect sensor provides a very stable output based on the magnetic field near it. Hence, if proper ADC is used and proper mapping methods are used along with a more stable glove design, virtual simulation of hand gestures with high accuracy can be performed.

This project is modular and uses minimum resources. The simulation of a real-world human hand is a step towards achieving the greater objective of mapping human locomotive abilities to a limitless virtual environment.

9. APPENDICES

APPENDIX A: PROJECT SCHEDULE

Table 9-1: Gantt Chart



APPENDIX B: PROJECT BUDGET

Table 9-2: Budget

S.N.	Components	Quantity	Price per quantity (Rs.)	Total Price (Rs.)
1	ESP32	1	900	900
2	Hall Effect Sensor	20	30	600
3	Neodymium Magnet	20	50	1000
4	Gyroscope	1	350	350
5	16-channel Multiplexer	1	300	300
6	LM7805	1	50	50
7	AMS 1117	1	15	15
8	Battery	1	2000	2000
9	Gloves	1	150	150
10	Miscellaneous	-	-	575
Total				**5800**

APPENDIX C: CIRCUIT DIAGRAM

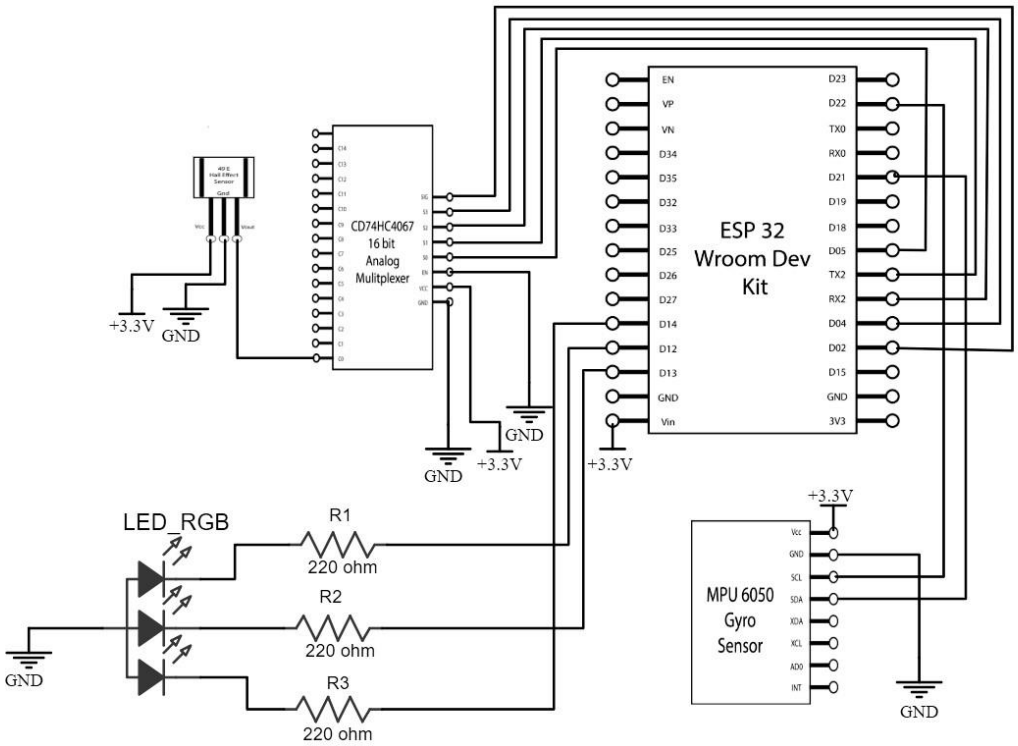


Figure 9-1: Circuit Diagram

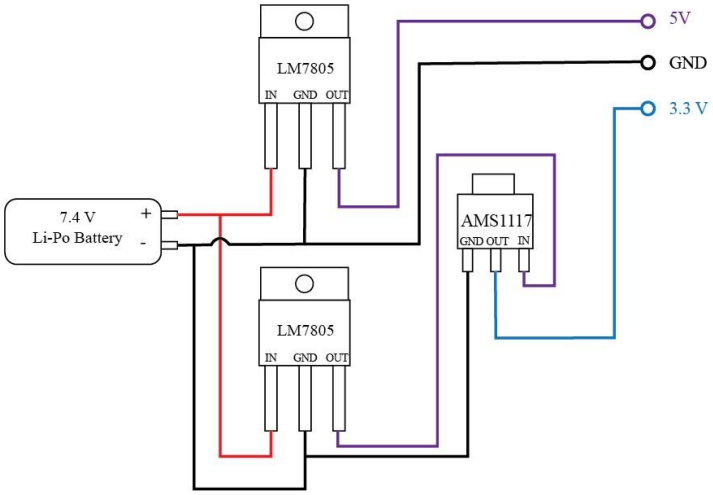


Figure 9-2: Power Supply Unit Circuit Diagram

APPENDIX D: HARDWARE DESIGN

Figure 9-3: Glove Design

Figure 9-4: Glove without Circuit Case

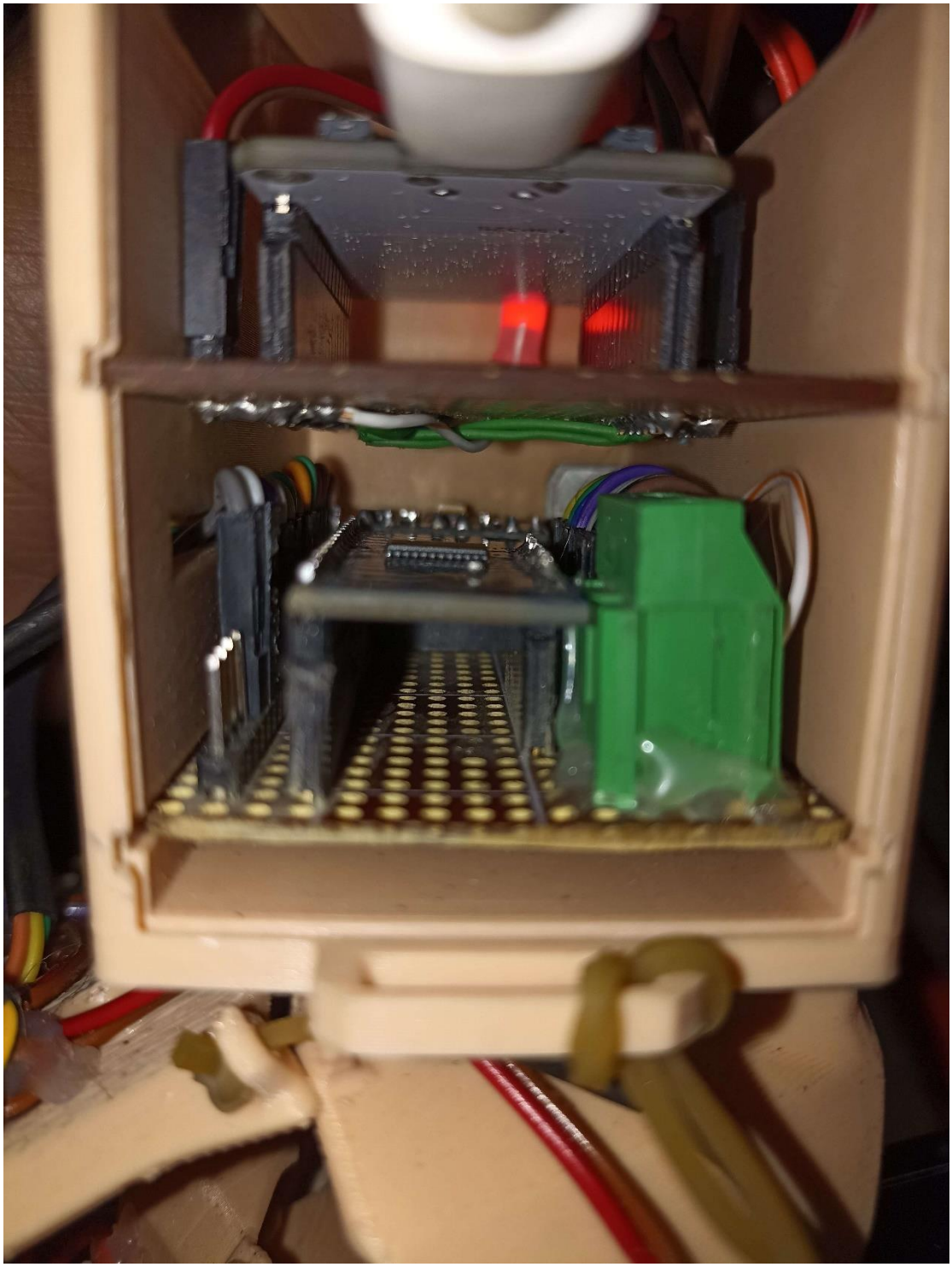


Figure 9-5: Circuit Inside Case

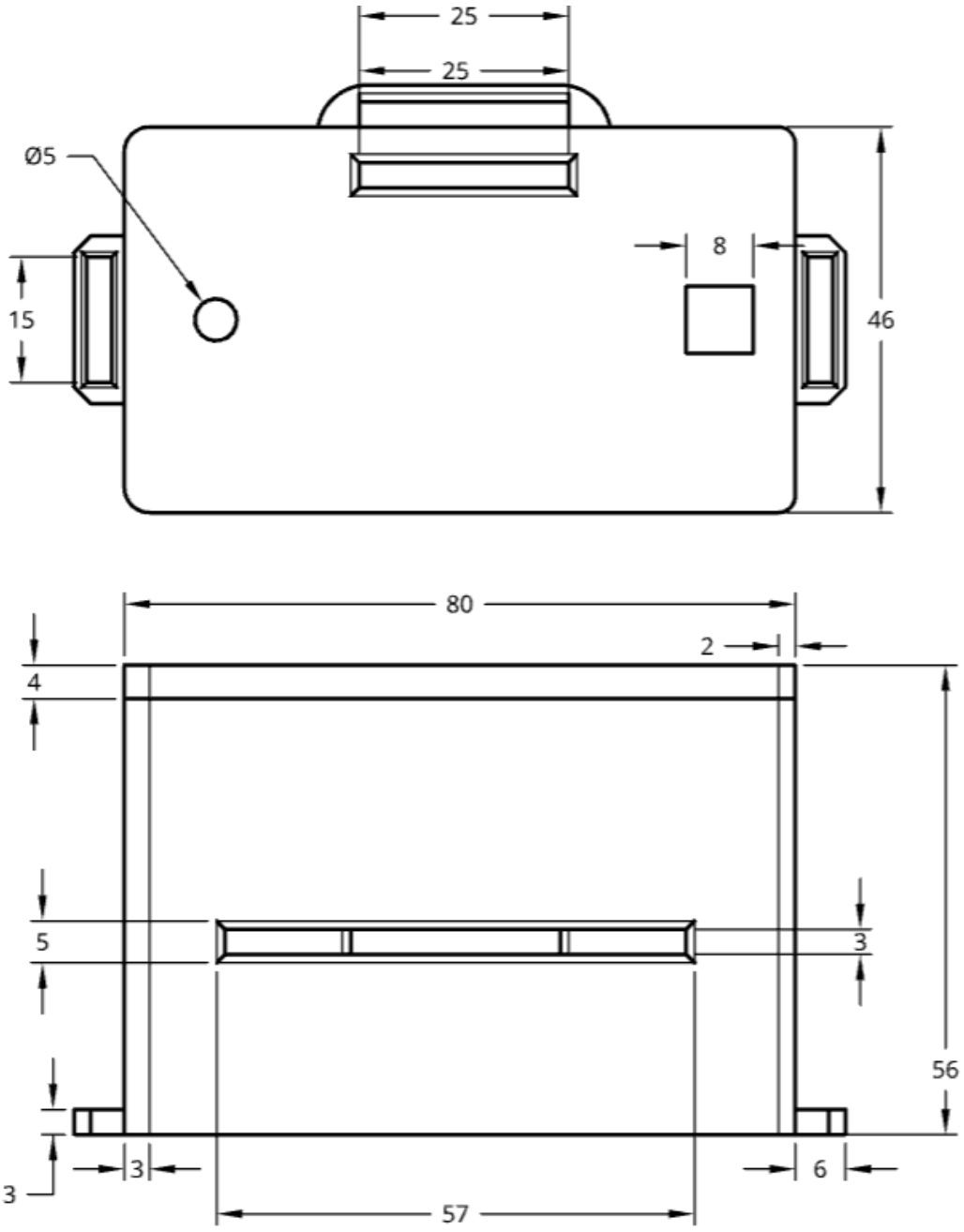


Figure 9-6: Top and Front View of Circuit Case

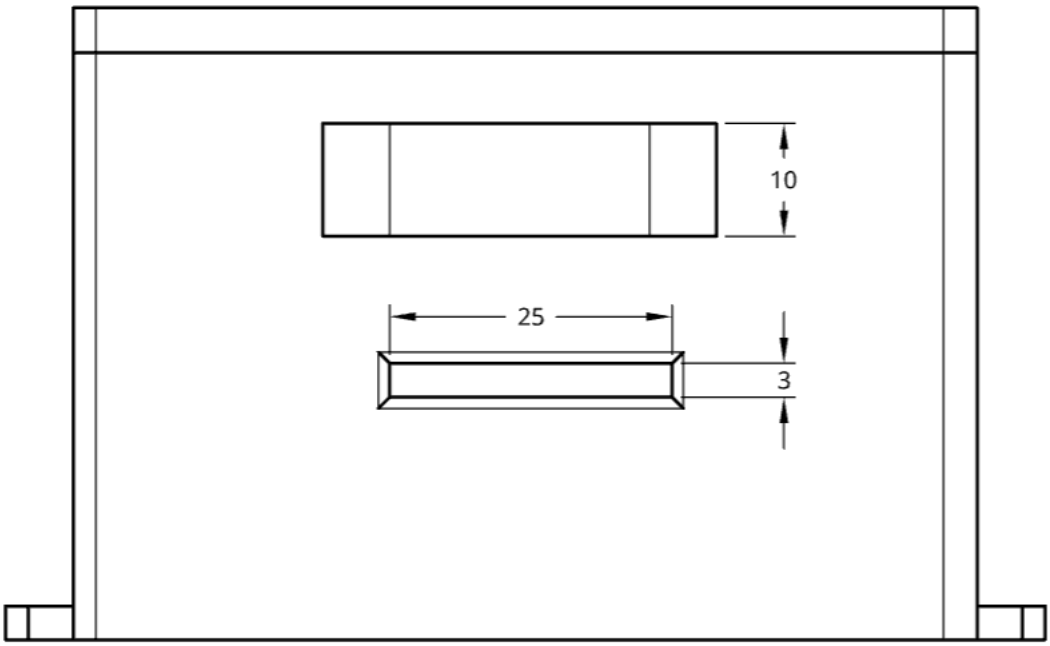


Figure 9-7: Back View of Circuit Case

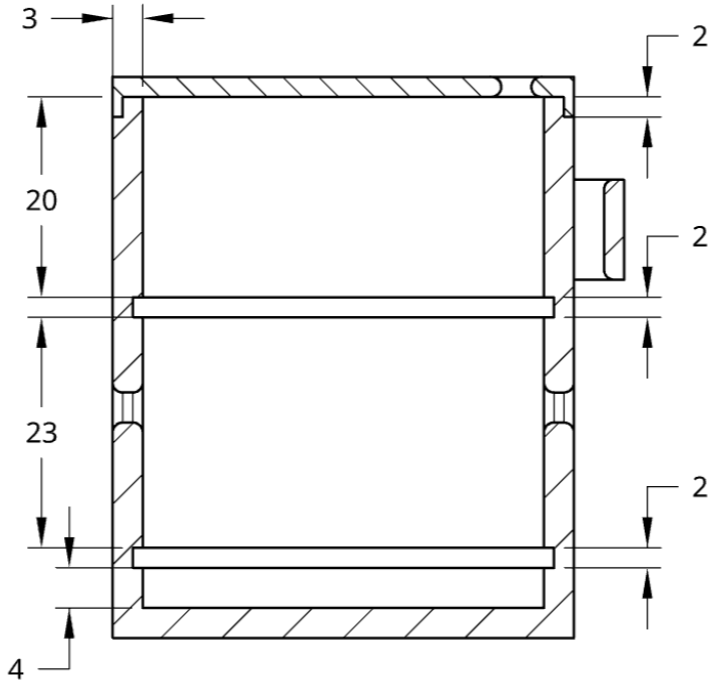


Figure 9-8: Right Section of Circuit Case

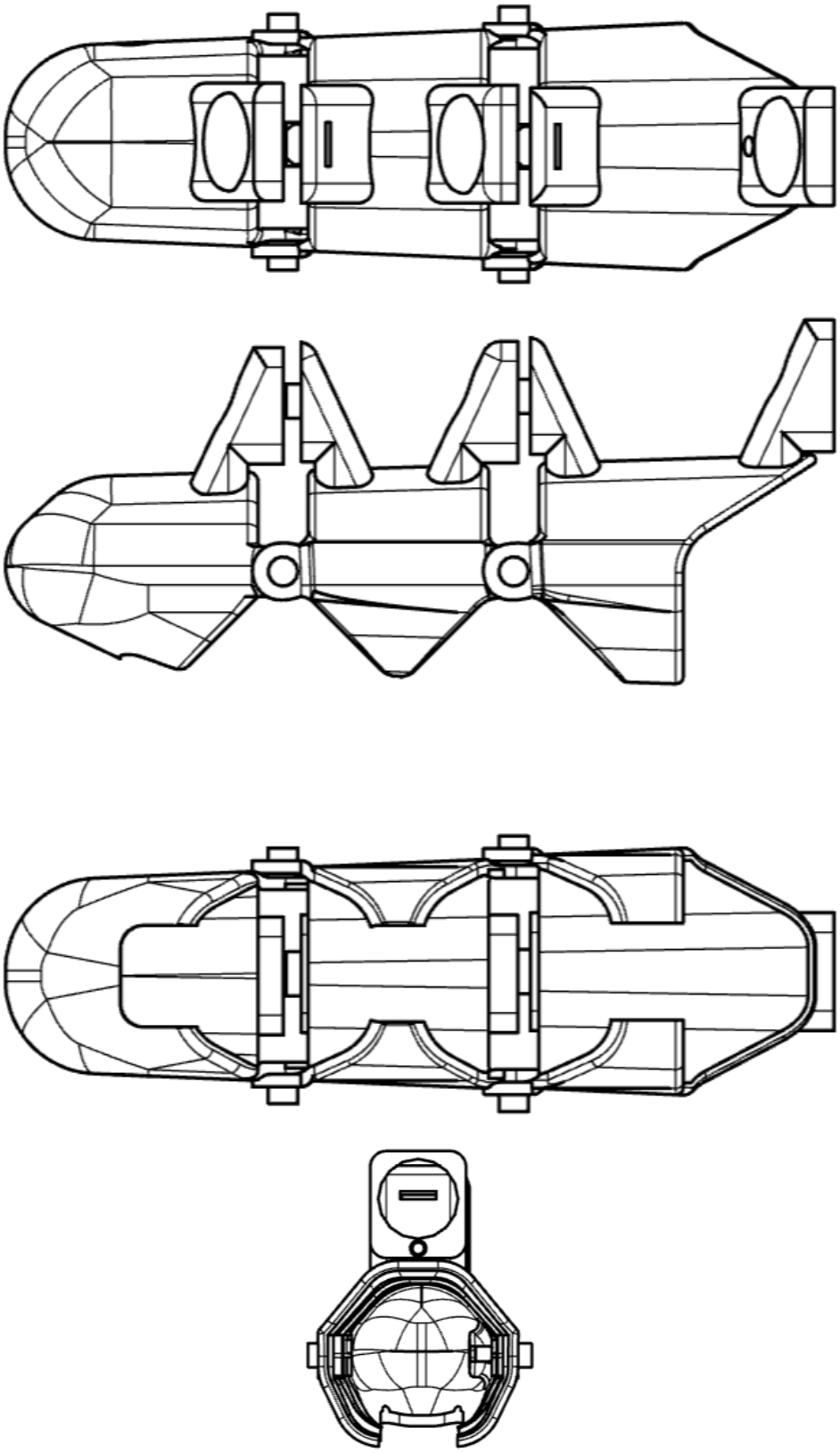


Figure 9-9: Glove Design Sketch

APPENDIX E: BEHAVIOR OF HALL EFFECT SENSOR

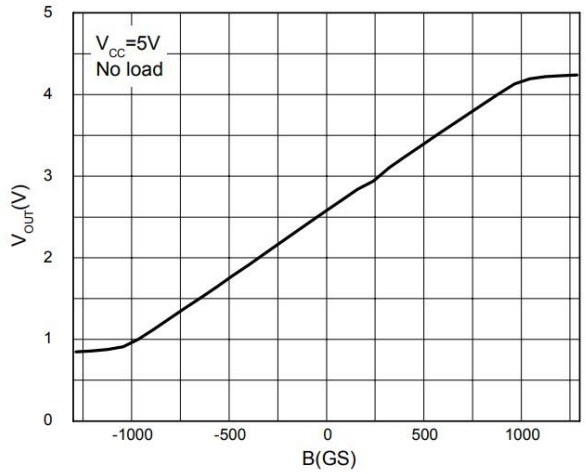


Figure 9-10: Vout vs Magnetic Field for Vcc = 5V

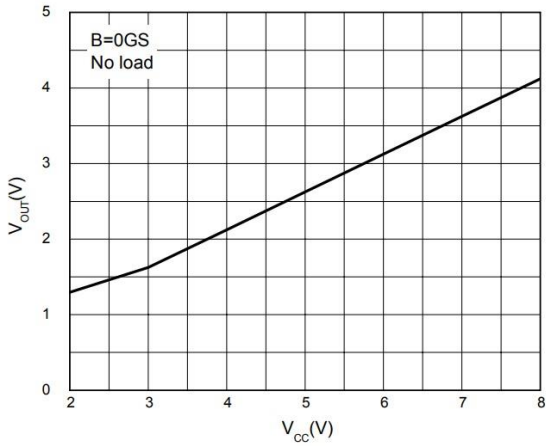


Figure 9-11: Vout vs Vcc at Constant Magnetic Flux Density

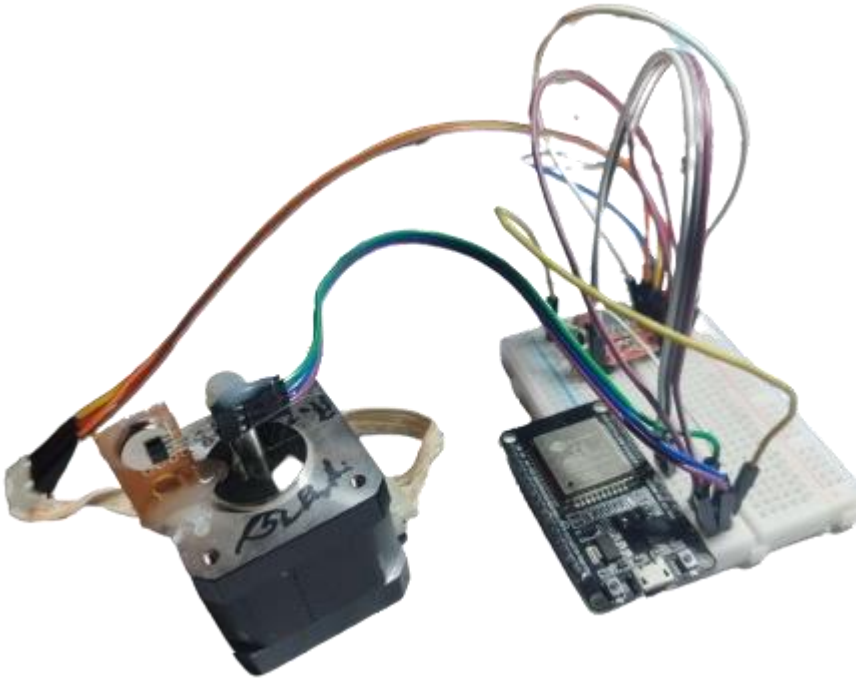


Figure 9-12: Analysis of Hall Effect Sensor using Stepper Motor

APPENDIX F: CODE SNIPPETS

F-1: Python Script for Scaling Data

```
def scale_func(y_values, min, max):
    mid_val = (max + min)/2
    length = len(y_values)
    length = length - 1
    high_diff = max - y_values.iloc[0]
    y_values += high_diff

    low_diff = y_values.iloc[length] - min
    y_values -= ((low_diff)/2)

    y_values -= mid_val

    new_max_value = y_values.iloc[0]
    scaling_fac = (max - mid_val)/new_max_value
```

```
    y_values *= scaling_fac
    y_values += mid_val

    return y_values
```

F-2: Event Handler in ESP32

```
void webSocketEvent(WStype_t type, uint8_t* payload, size_t length)
{
  switch(type)
  {
    case WStype_DISCONNECTED:
      Serial.printf("\n[WSC] Disconnected");
      clientConnected = false;
      break;
    case WStype_CONNECTED:
      Serial.printf("\n[WSC] Connected to url: %s\n", payload);
      clientConnected = true;
      break;
    case WStype_TEXT:
      Serial.printf("\n[WSC] got text: %s\n", payload);
      //if any operation need to be performed with the received data, then do it here.
      break;
    case WStype_BIN:
      Serial.printf("\n[WSC] got binary of length: %u\n", length);
      //if any operation need to be performed with the received data, then do it here.
      break;
    case WStype_PING:
      Serial.printf("\n[WSC] get ping\n");
      break;
    case WStype_PONG:
```

```
      Serial.printf("\n[WSC] get pong\n");
      break;
  }
}
```

F-3: Serialization of JSON object

```
   rawJson["thumbTop"] = thumbTop;
   rawJson["thumbBottom"] = thumbBottom;
   rawJson["indexTop"] = indexTop;
   rawJson["indexMid"] = indexMid;
   rawJson["indexBottom"] = indexBottom;
   rawJson["middleTop"] = middleTop;
   rawJson["middleMid"] = middleMid;
   rawJson["middleBottom"] = middleBottom;
   rawJson["ringTop"] = ringTop;
   rawJson["ringMid"] = ringMid;
   rawJson["ringBottom"] = ringBottom;
   rawJson["pinkeyTop"] = pinkeyTop;
   rawJson["pinkeyMid"] = pinkeyMid;
   rawJson["pinkeyBottom"] = pinkeyBottom;
   rawJson["xAngle"] = xAngle;
   rawJson["yAngle"] = yAngle;
   rawJson["zAngle"] = zAngle;
   serializeJson(rawJson, serializedData);
   webSocket.sendTXT(serializedData);
```

F-4: Linear Mapper

```
const linearMapper = function (raw) {
   //Linear Mapping using the simple formula of straight line y = mx+c
   // y-y1 = ((y2-y1)/(x2-x1))*(x-x1); y = angle, x = raw data, x2 = ninety degree reading, x1 = zero degreee reading, y2 = 90, y1 = 0,
```

```
    const angle = ((90 - 0) / (ninetyDegreeReading - zeroDegreeReading)) * (raw - zeroDegreeReading);
    return Math.round(angle);
}
```

F-5: Mapping Table

```
const mappingTable = {
    2970 : 0,
    2850 : 14,
    2326 : 23,
    2059 : 42,
    1961 : 62,
    1921 : 90,
}
```

F-6: Piecewise Linear Mapper

```
const pieceWiseLinearMapper = function(raw)
{
    console.log(keysList);
    let angle;
    if(raw > keysList[0])
    {
        angle = 0;
    }
    else if(raw<keysList[keysList.length-1])
    {
        angle = 90;
    }
    else{
        const lesser = keysList.filter(element => element > raw).sort((a,b)=> a-b)[0];
        const greater = keysList.filter(element => element < raw).sort((a,b)=>b-a)[0];
```

```
        const lesserAngle = mappingTable[lesser];
        const greaterAngle = mappingTable[greater];
        angle = Math.round(((greaterAngle - lesserAngle) / (greater - lesser)) * (raw - lesser) + lesserAngle);
    }
    console.log(angle);
    return angle;
}
```

F-7: Update Loop to Check Server Status

```
void Update()
    {
        if (!ws.IsAlive)
        {
            //Debug.Log("Not Connected");
            if (playb.activeSelf)
            {
                playb.SetActive(false);
                serverstat.SetActive(true);
                //Debug.Log("Server Offline");
            }
            StartCoroutine(checkStatus());
        }
        else
        {
            if (!playb.activeSelf && serverstat.activeSelf)
            {
                playb.SetActive(true);
                serverstat.SetActive(false);
            }
```

```
    }
  }
```

F-8: Try establishing connection

```
IEnumerator checkStatus()
  {
      ws.ConnectAsync();
      yield return new WaitForSeconds(2);
  }
```

F-9: Transformation of coordinates

```
      fingerbend_top_ring.x = ringTop;
      ring_top_R.transform.localEulerAngles = -fingerbend_top_ring;

      fingerbend_mid_ring.x = ringMid;
      ring_mid_R.transform.localEulerAngles = -fingerbend_mid_ring;

      fingerbend_bottom_ring.x = ringBottom;
      ring_bottom_R.transform.localEulerAngles = -fingerbend_bottom_ring;
```

F-10: Class of angle data

```
public class FingerJointData
{
  public int ThumbTop { get; set; }
  public int ThumbBottom { get; set; }
  public int IndexTop { get; set; }
  public int IndexMid { get; set; }
  public int IndexBottom { get; set; }
  public int MiddleTop { get; set; }
  public int MiddleMid { get; set; }
  public int MiddleBottom { get; set; }
```

```
    public int RingTop { get; set; }
    public int RingMid { get; set; }
    public int RingBottom { get; set; }
    public int PinkyTop { get; set; }
    public int PinkyMid { get; set; }
    public int PinkyBottom { get; set; }
    public int XAngle { get; set; }
    public int YAngle { get; set; }
    public int ZAngle { get; set; }
}
```

APPENDIX G: SUMMARY OF PLAGIARISM REPORT

Virtual Hand Simulation Of Hand Gesture Using Hall Effect Sensor

ORIGINALITY REPORT

13%
SIMILARITY INDEX

PRIMARY SOURCES

1	www.researchgate.net Internet	293 words — 2%
2	lastminuteengineers.com Internet	177 words — 1%
3	it.overleaf.com Internet	161 words — 1%
4	hdl.handle.net Internet	122 words — 1%
5	upcommons.upc.edu Internet	116 words — 1%
6	www.angelfire.com Internet	96 words — 1%
7	elibrary.tucl.edu.np Internet	85 words — < 1%
8	Ayash Ashraf, Shazia Ashraf, Navaid Zafar Rizvi, Shakeel Ahmad Dar. "Low power design of asynchronous SAR ADC", 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016	77 words — < 1%

9	www.coursehero.com Internet	73 words	< 1%
10	mrtmrcn.com Internet	64 words	< 1%
11	ro.uow.edu.au Internet	58 words	< 1%
12	"Advances in Machinery, Materials Science and Engineering Application", IOS Press, 2022 Crossref	45 words	< 1%
13	Yu, Ho. "Design and control of a compact 6-degree-of-freedom precision positioner with Linux-based real-time control", Proquest, 20111108 ProQuest	42 words	< 1%
14	es.scribd.com Internet	42 words	< 1%
15	Laura Dipietro. "A Survey of Glove-Based Systems and Their Applications", IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews), 07/2008 Crossref	38 words	< 1%
16	lup.lub.lu.se Internet	38 words	< 1%
17	iotmakervn.github.io Internet	33 words	< 1%
18	www.macrumors.com Internet	33 words	< 1%

19	Ashish Kumar Dutta, J.Simon Gnanesh Paul, S. Siddharth, K.G. Nitish, J.S.Shyam Sundar, P.S. Manoharan. "Hand Gesture Controlled Car using Bluetooth Modules and Accelerometer Sensor", 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), 2023 Crossref	29 words — < 1%
20	escholarship.mcgill.ca Internet	29 words — < 1%
21	dspace.aiub.edu Internet	28 words — < 1%
22	eprints.utar.edu.my Internet	27 words — < 1%
23	archive.org Internet	26 words — < 1%
24	doczz.net Internet	26 words — < 1%
25	forum.arduino.cc Internet	21 words — < 1%
26	fabacademy.org Internet	18 words — < 1%
27	www.geeksforgeeks.org Internet	18 words — < 1%
28	Hadi Pahlavanzadeh, Mohammad Azim Karami. "A low settling time switching scheme for SAR ADCs with reset-free regenerative comparator", International Journal of Circuit Theory and Applications, 2023 Crossref	17 words — < 1%

29	www.devteam.space Internet	17 words — < 1%
30	www.encora.com Internet	17 words — < 1%
31	"Data Science", Springer Science and Business Media LLC, 2019 Crossref	16 words — < 1%
32	www.pm.lth.se Internet	16 words — < 1%
33	dspace.uii.ac.id Internet	15 words — < 1%
34	johnandsharepoint.ca Internet	15 words — < 1%
35	Winn, Joshua D.. "Integration of Massive Multiplayer Online Role-Playing Games Client-Server Architectures with Collaborative Multi-User Engineering CAx Tools.", Brigham Young University, 2020 ProQuest	14 words — < 1%
36	shop.ottobock.us Internet	13 words — < 1%
37	www.trademarkelite.com Internet	13 words — < 1%
38	Ling Bai, Yinguo Li, Thia Kirubarajan, Xinbo Gao. "Quadruple tripatch-wise modular architecture-based real-time structure from motion", Neurocomputing, 2022 Crossref	12 words — < 1%
	prism.ucalgary.ca	

39	Internet	12 words — < 1%
40	sc.uobaghdad.edu.iq Internet	12 words — < 1%
41	clok.uclan.ac.uk Internet	11 words — < 1%
42	etd.lib.metu.edu.tr Internet	11 words — < 1%
43	lib.buet.ac.bd:8080 Internet	11 words — < 1%
44	pdfs.semanticscholar.org Internet	11 words — < 1%
45	www.dkes-scs.com Internet	11 words — < 1%
46	www.dynisco.com Internet	11 words — < 1%
47	C.-S. Fahn, H. Sun. "Development of a Data Glove With Reducing Sensors Based on Magnetic Induction", IEEE Transactions on Industrial Electronics, 2005 Crossref	10 words — < 1%
48	ia600202.us.archive.org Internet	10 words — < 1%
49	patents.google.com Internet	10 words — < 1%
50	www.manchestereveningnews.co.uk Internet	10 words — < 1%

51	www.toppr.com Internet	10 words — < 1%
52	Marcin Bujko, Marta Bocheńska, Piotr Srokosz, Ireneusz Dyka. "Modernized Resonant Column and Torsional Shearing Apparatus With Multipoint Contactless Displacement Detection System", Studia Geotechnica et Mechanica, 2023 Crossref	9 words — < 1%
53	Timofeev, Vladimir P., and Dmitriy O. Nikolsky. "The Role of the Fast Motion of the Spin Label in the Interpretation of EPR Spectra for Spin-Labeled Macromolecules", Journal of Biomolecular Structure and Dynamics, 2003. Crossref	9 words — < 1%
54	Tomas CUZANAUSKAS. "INVESTIGATION OF MEDIA ACCESS CONTROL IN WIRELESS NETWORKS", Vilnius Gediminas Technical University, 2018 Crossref	9 words — < 1%
55	arcanerobotics.wordpress.com Internet	9 words — < 1%
56	dl.icdst.org Internet	9 words — < 1%
57	dokumen.pub Internet	9 words — < 1%
58	mafiadoc.com Internet	9 words — < 1%
59	pdfcoffee.com Internet	9 words — < 1%

60	rocop.sourceforge.net Internet	9 words — < 1%
61	"Intelligent Sustainable Systems", Springer Science and Business Media LLC, 2022 Crossref	8 words — < 1%
62	Bonafede, Nicholas J., Jr.. "Low-Cost Reaction Wheel Design for Cubesat Applications.", California Polytechnic State University, 2023 ProQuest	8 words — < 1%
63	J. Neel, S. Srikanteswara, J.H. Reed, P.M. Athanas. "A comparative study of the suitability of a custom computing machine and a VLIW dsp for use in 3G applications", IEEE Workshop onSignal Processing Systems, 2004. SIPS 2004., 2004 Crossref	8 words — < 1%
64	Nur Zazmera Mustafa Kamal, Nabihah Ahmad, Siti Hawa Ruslan, Hasmayadi Abdul Majid et al. "Design voltage comparator 14-bit for successive approximation analog-to-digital converter", Journal of Physics: Conference Series, 2020 Crossref	8 words — < 1%
65	auto.sfgate.com Internet	8 words — < 1%
66	digitalcommons.fiu.edu Internet	8 words — < 1%
67	docplayer.net Internet	8 words — < 1%
68	ece.anits.edu.in Internet	8 words — < 1%

69	eprints.ucm.es Internet	8 words — < 1%
70	smallbizclub.com Internet	8 words — < 1%
71	www-dweb-cors.dev.archive.org Internet	8 words — < 1%
72	"Nanomagnetism: Applications and Perspectives", Wiley, 2017 Crossref	6 words — < 1%
73	Ajit Kumar, Ajoy Kanti Ghosh. "ANFIS-Delta method for aerodynamic parameter estimation using flight data", Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 2018 Crossref	6 words — < 1%
74	David A. Duce. "Chapter 12 Theory and practice in interactionally rich distributed systems", Springer Nature, 1997 Crossref	6 words — < 1%
75	Sulove Bhattarai, Sudip Bhujel, Santosh Adhikari, Shanta Maharjan. "Design and Implementation of a Portable ECG Device", Journal of Innovations in Engineering Education, 2020 Crossref	6 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF

APPENDIX H: STUDENT SUPERVISOR CONSULTATION FORM

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING, THAPATHALI CAMPUS
Department of Electronics and Computer Engineering
Student & Supervisor Consultation Form
(BEI Minor Project, Batch 2077)

Project Title	Virtual Simulation of Hand Gestures using Hall Effect Sensors
Group Members (Name & Roll Number)	Aayush Pathak (THA077BEI002) Bal Krishna Shah (THA077BEI010) Sabin Acharya (THA077BEI035) Sagal Kanki (THA077BEI036)
Name of Supervisor	Er. Sagar Shakya

S.N.	Brief Summary of Discussion Agenda	Date	Supervisor Signature
1	Proposal Discussion and Guidance	2080/09/19	
2	Presentation Guidance	2080/09/20	
3	Project Workflow Discussion	2080/09/16	
4	Presentation Verification and Correction	2080/09/21	
5	Report Feedback and Design Guidelines	2080/09/27	
6	Discussion on prototype and Wiring Issues	2080/10/07	
7	Progress Demonstration and Target Setting for Mid term	2080/10/12	
8	Discussion on Making Report, Demo Video and Presentation	2080/10/17	
9	Report Verification and Correction For Mid-Term Defence.	2080/10/22	
10	3D Printing Consultation	2080/10/28	

Jan-07
Date of Approval

Signature of Supervisor

At least TWO consultation is required before Final Proposal Defense
At least FIVE consultations are required BEFORE Midterm and TEN consultations for FINAL

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING, THAPATHALI CAMPUS
Department of Electronics and Computer Engineering
Student & Supervisor Consultation Form
(BEI Minor Project, Batch 2077)

Project Title	Virtual Simulation of Hand Gestures using Hall Effect Sensors
Group Members (Name & Roll Number)	Aayush Pathak (THA077BEI002) Bal Krishna Shah (THA077BEI010) Sabin Acharya (THA077BEI035) Safal Karki (THA077BEI036)
Name of Supervisor	Er. Saroj Shakya

S.N.	Brief Summary of Discussion Agenda	Date	Supervisor Signature
1	Index Finger Assembly Demonstration	2080-11-14	
2	Sensors and Wiring Placement Demo.	2080-11-15	
3	Circuit Casing error Correction.	2080-11-03	
4	Full hand Assembly demonstration.	2080-11-16	
5	Final Report Discussion.	2080-11-17	
6	Final Output Demonstration and Problems Discussion.	2080-11-18	
7			
8			
9			
10			

Falgun-21, 2080 B.S.
Date of Approval

Signature of Supervisor

At least TWO consultation is required before Final Proposal Defense
At least FIVE consultations are required BEFORE Midterm and TEN consultations for FINAL

References

[1] R. Adnan and H. Osman, "Wearable technologies for hand joints monitoring for rehabilitation: A survey," *Microelectronics Journal,* vol. 88, no. 0026-2692, pp. 173183, 2019.

[2] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *IEEE Computer Graphics and Applications,* vol. 14, no. 1, pp. 30-39, 1994.

[3] Humanware, "HumanGlove," Humanware, November 2023. [Online]. Available: https://www.hmw.it/en/research-and-development/humanglove/. [Accessed December 2023].

[4] P. Laura Dipietro, P. Angelo M. Sabatini and P. Paolo Dario, "Evaluation of an instrumented glove for hand-movement," *Journal of Rehabilitation Research and Development ,* vol. 40, no. 2, pp. 179-190, 2003.

[5] L. Almeida, E. R. Lopes, B. Yalçinkaya and R. Martins, "Towards natural interaction in immersive reality with a cyber-glove," in *2019 IEEE International Conference on Systems, Man, and Cybernetics*, Bari, Italy, 2019.

[6] L. Dipietro, A. Sabatini and P. Dario, "A Survey of Glove-Based Systems and Their Applications," *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews),* vol. 4, no. 38, pp. 461-482, 2008.

[7] S. S. Dahal, A. Ghimire, A. Basnet, A. Shrestha and B. Kadayat, "Smart Glove," in *Office of KEC Research and Publication (OKRP)*, Kathmandu, 2019.

[8] BCD Semiconductor Manufacturing Limited, "AH49E," August 2010. [Online]. Available: https://www.diodes.com/assets/Datasheets/AH49E.pdf. [Accessed December 2023].

[9] MDN Web Docs, "The WebSocket API (WebSockets)," Mozilla, 22 November 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Accessed 27 December 2023].

[10] Unity Documentation, "Model file formats," Unity, 3 January 2024. [Online]. Available: https://docs.unity3d.com/Manual/3D-formats.html. [Accessed 4 January 2024].