TECHNICKÁ UNIVERZITAV LIBERCI

Fakulta mechatroniky, informatiky
a mezioborovýchstudií

Human Hand-Gesture Controlled Robotic Arm by Image Processing

Semester Thesis by

Anoop Rode

Under the guidance of

Ing. Miroslav Holada Ph.D.

Liberec 2023

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

FACULTY OF MECHATRONICS,
INFORMATICS AND INTERDISCIPLINARY
STUDIES TUL

Master Thesis

Human Hand-Gesture Controlled Robotic Arm by Image Processing

Study programme: Masters
Study branch: M.Sc. Mechatronics
Author: **ANOOP RODE**
Supervisor: Mr. Miroslav Holada
{Department}

Liberec { October 20, 2023 }

FACULTY OF MECHATRONICS, INFORMATICS AND INTERDISCIPLINARY STUDIES TUL

Master Thesis Assignment Form

Human Hand-Gesture Controlled Robotic Arm by Image Processing

Name and surname:	**ANOOP RODE**
Identification number:	98012924AR
Study programme:	Masters
Study branch (specialization):	Mechatronics
Assigning department:	{MTI}
Academic year:	2023-24/2024

Rules of Elaboration:

1. Find out the current state of development of gesture-assisted robotic arms. Learn about the 3D printing capabilities of robotic arm parts. Map the current state of the art in computer image processing, especially the real-time gesture recognition capabilities.
2. Design and 3D print robotic arm components. Install appropriate actuators in the printed robotic arm to enable precise and dexterous movements. Connect the servo motors to a suitable control unit. Create an interface between the image recognition program (hand gesture recognition) running on the desktop
3. Implement an algorithm for hand gesture recognition, which then sets the recognized gesture on the robotic arm.
4. . Demonstrate the feasibility and functionality of a robotic arm controlled by a human hand. Design and execute a series of repeatable tests to document the functionality achieved.
5. Consider possible improvements and limitations of a given implementation. Suggest possible applications.

List of professional literature:

[1] SZELISKI, Richard. Computer Vision. Online. Texts in Computer Science. Cham: Springer International Publishing, 2022. ISBN 978-3-030-34371-2. Available from: https://doi.org/10.1007/978-3-030-34372-9. [Feeling. 2023-10-30].

[2] ŠONKA, Milan; HLAVÁČ, Václav and BOYLE, Roger. Image Processing, analysis, and machine vision. 4th ed. Stamford: Cengage Learning, 2015. ISBN 978-1133593607

[3] KOS, Anton and Anton UMEK. Biomechanical biofeedback systems and applications. New York, NY: Springer Berlin Heidelberg, 2018. ISBN 978-331-9913-483.

In Liberec on.....

.....

Ing. Miroslav Holada Ph.D.

Scope of Graphic Work:	by appropriate documentation
Scope of Report:	40–50 pages
Thesis Form:	printed/electronic
Thesis Language:	English

List of Specialized Literature:

Thesis Supervisor:	Mr. Miroslav Holada {Institute of Information Technology and Electronics}
Date of Thesis Assignment:	October 20, 2023
Date of Thesis Submission:	May 14, 2024

S.S.

{Dean of faculty}	{Head of institute}
Dean	Head of the institute

Liberec October 20, 2023

Declaration

I hereby certify, I, myself, have written my master's thesis as an original and primary work using the literature listed below and consulting it with my thesis supervisor and my thesis counselor.

I acknowledge that my master thesis is fully governed by Act No. 121/2000 Coll., the Copyright Act, in particular Article 60 – School Work.

I acknowledge that the Technical University of Liberec does not infringe on my copyrights by using my master thesis for internal purposes of the Technical University of Liberec.

I am aware of my obligation to inform the Technical University of Liberec of having used or granted a license to use the results of my master thesis; in such a case the Technical University of Liberec may require reimbursement of the costs incurred for creating the result up to their actual amount.

At the same time, I honestly declare that the text of the printed version of my master thesis is identical to the text of the electronic version uploaded into the IS STAG.

I acknowledge that the Technical University of Liberec will make my master thesis public in accordance with paragraph 47b of Act No. 111/1998 Coll., on Higher Education Institutions and on Amendment to Other Acts (the Higher Education Act), as amended.

I am aware of the consequences which may under the Higher Education Act result from a breach of this declaration.

May 14, 2024 ANOOP RODE

Human Hand-Gesture Controlled Robotic Arm by Image Processing

Abstract

This research presents an interdisciplinary methodology for the design, implementation and development of the robotic gesture control system integrating with computer vision, mechanical principles, and electronics circuitry. The interdisciplinary approach of the project involves the collaboration of mechanical, computer and electrical electronics domains.

The circuit design architecture has generic components such as ESP32 WROOM32 Module and MG996R Servo Motor. This system architecture has been developed to obtain the precise angular control of servo motors with the help of PWM signals generated by the ESP32 module. The interfacing and configuration step of the ESP32 WROOM Module involves firmware development using Arduino IDE and leading communication with peripheral devices via UART protocol. Serial communication has been established between C++ running on ESP32 and Python script on Laptop.

Computer Vision Algorithm allows us to detect and track the objects desired objects in footage. The Libraries used for gesture recognition by using MediaPipe & OpenCV Libraries. Over the pre-processed footage with the help of libraries, I have implemented a custom post-processing landmark detection technique for angular calculation via vector approach.

The mechanical structure design and fabrication involves Modelling and 3D printing of the parts. After 3D printing involves the assembly of parts and calibrating with servo angular motion for operating precisely.

The demonstrated methodology connects engineering principles with robotic control strategy to obtain a robust control algorithm using computer vision to operate finger movement precisely.

This project can be the promising for the applications such as prostherics Hands and HMI applications on unmanned fields.

Keywords

Computer Vision, ESP32 Wroom, 3D Printing, Hand Gesture Recognition, Bionic Hand Motion and Motion Control, HMI

Acknowledgments

I extend my deepest gratitude to **Ing. Miroslav Holada, Ph.D**., for his invaluable guidance, unwavering support, and profound expertise throughout the duration of this research project. His insightful supervision, encouragement, and constructive feedback have been instrumental in shaping this thesis and enhancing its quality.

I am immensely thankful to him for providing me with the opportunity to explore the fascinating realm of human hand-gesture-controlled robotic arms by image processing. His passion for research and dedication to fostering academic excellence have been truly inspiring.

I am also grateful to the faculty members of the **Technical University of Liberec (TUL)** for their encouragement and assistance during the course of my study. Their expertise and scholarly insights have enriched my learning experience and contributed significantly to the development of this thesis.

Furthermore, I extend my heartfelt appreciation to my family and friends for their unwavering support, understanding, and encouragement throughout this academic journey. In particular, I would like to thank my parents, **Mr. Mohan Sudhakar Rode** and **Mrs. Neeta Mohan Rode**, for their unconditional love, sacrifices, and belief in my abilities. Their guidance and encouragement have been the driving force behind my accomplishments.

Lastly, I would like to express my gratitude to all those who have directly or indirectly contributed to the completion of this thesis. Your support and assistance are deeply appreciated.

Table of Contents

List of images

1. Introduction

In today's world robotic industry is a boom for our lives to excel in every field. Robotics is a technology that we can use to reduce human efforts and perform the work with accuracy.

To execute the tasks with robotics we have to program the robot for the desired task with the help of various input devices such as joystick, buttons, sensors, etc. But nowadays as the computer industry is excelling from the technological point of view computer vision is a robust algorithm for object detection and tracking.

This Project is an extension of the customized computer vision technique for autonomous fetching of desired parameters from the object while tracking its trajectory. Human-computer interaction is an emerging field that allows us to communicate with machines to give input for executing the desired task with the help of machines.

This Project emphasizes gesture control robotic hands which unlock the feature to fetch the required parameters from objects detected in the camera frame and use those parameters in work execution by a mathematical approach to execute in perfection. So robotic movement would be fluid and full of dexterity like human anatomy.

1.1 Objectives of the Project

1. The main objective of this project is to design and fabricate a robotic bionic hand similar to human anatomy to execute the same gestures in real time by mimicking a human hand with good fluidity and dexterity.

2. Design and building of the robot hand will include CAD for modeling and fabrication of the robotic hand for a precise DOF and task operation.

3. Installation of servo motors MG996R-0-180 for controlling each finger with precise continuous motion from 0 to 180 as per the real-time finger movement with good dexterity and fluidity like human hands.

4. For interaction with ESP32 WROOM Module we used serial communication protocol to run and compile C++ program to receive commands from python code to run respective servos to specific angle.

5. Implementation of image processing algorithm. I developed computer vision code by using OpenCV and MediaPipe libraries to execute detection and tracking algorithms for basic vision algorithms for implementing angle detection calculation with vector calculation and speed of finger movement.

6. Gesture recognition focuses on finger movements to detect the gestures of the human finger to extract meaningful data with desired parameters such as the angle of the finger.

1.2 Significance of the Research

The significance of the research is the Human-Computer Interaction of this project's computer vision algorithm to implement a communication channel between robots and humans to execute a particular task through a computer vision algorithm[7].

This system would enable users to control robots by giving natural gestures and movements. These advancements give a boom solution for the fields, including manufacturing, healthcare, rehabilitation, and assistive technology[9]. Where exactly no human can enter the field but human-controlled robots will perform the task with good dexterity and fluidity in the movements of robotic hands.

2. Literature Review

1. Human-mobile robot interaction using hand gesture-based leap:

This article describes the integration of Leap Motion sensors for intuitive gesture control of mobile robots[3]. This highlights the potential of cyber-physical systems to enhance human-robot interaction by providing a more natural and efficient means of communication between humans and robots [3]. This research demonstrates how accurate motion recognition can be achieved using Leap Motion sensors and advances the development of control interfaces that could revolutionize interaction with robotic systems in a variety of applications [3].

2. Human-robot interaction based on gesture and motion recognition:

In this study, present a sophisticated gesture and action recognition model using a 3D Single Shot MultiBox Detector (SSD) combined with Dynamic Time Warping (DTW). This method aims to provide a more flexible and intuitive way to control robotic arms by reducing the barrier between human commands and robot actions[3]. Using 3D SSD for spatial perception and DTW for temporal analysis, this paper provides a powerful framework for improving the ability of robots to understand and perform complex human gestures [2].

3. Robotic hand controlled by gestures using OpenCV:

To create a cost-effective and accessible solution, this paper details the development of a 3D-printed robotic arm controlled by real-time gesture recognition[7]. Using the open-source computer vision library (OpenCV) and a standard USB camera, this study demonstrates how gestures can be accurately translated into robot movements without the need for expensive hardware[7]. This methodology includes comprehensive steps of gesture detection, feature extraction, and command translation, highlighting the practicality of OpenCV in bridging the gap between human gestures and robot execution [7].

4. Real-time control of a robotic arm using hand gestures with multiple end-effectors.

By studying the control of a robotic arm with increased flexibility, this paper evaluates the effectiveness of the Leap Motion controller in controlling a robot equipped with various end effectors[4]. Specifically, we compare the capabilities of a standard electric gripper with an AR10 robotic arm to analyze its performance in a real-time control scenario [4].

5. A Robotic Hand: Controlled With a Vision-Based Hand Gesture Recognition System.

A workaround is to use cameras to identify hand movements. This method eliminates the need for any other gadgets. Rather, your hand movements are captured by a camera, which the robot interprets as commands[5]. This type of technology makes engaging with robots easier, which is helpful for patients, the elderly, and those with impairments. It is also advantageous in areas where human travel is hazardous or unfeasible[5]. In this regard, a system that recognizes hand motions and controls a robotic hand using a camera was developed. The idea is to make it easier for everyone to engage with robots by developing a straightforward and efficient method that only requires your hands[5].

6. Efficient and Feasible Gesture Controlled Robotic Arm

Remote controllers became immensely popular in the 1990s because they simplified user experience. Everywhere you looked, people were using remote controllers for everything from TVs to toy cars[12]. However as technology developed, people's needs for convenience and flexibility increased. This resulted in the creation of robotic arms, which are currently employed in numerous industries. Robotic arms have a lot of advantages over remote controls. Applications in aviation, medical technology, and even the military have adopted them[12]. When it comes to efficiency and functionality, they constitute a significant improvement. We can now improve upon the current state of robotic arms[12]. The most recent technology enables us to operate robotic arms with gestures rather than remote controls by utilizing computer vision and different hardware components including sensors and controllers. This paper discusses how this approach can significantly expand the use of robotic arms and increase their efficiency in many applications[6].

3. Methodology

3.1 Electronic Circuit Designing and Component Selection

This project includes interdisciplinary work from Mechanical Engineering, Computer Engineering, and Electrical & Electronics Engineering. Which includes interdisciplinary skills such as Modelling, Fabrication, Programming, and interfacing of actuators and sensors.

3.1.1. List of Components Electronic Components:

1. ESP32 WROOM Module
2. MG996R Servo Motor
3. Servo Power Distribution Board
4. Bread Board
5. Soldering Machine
6. Jumper Wires/Soldering Cables
7. Adhesive Agents
8. Laptop (Python Program Execution via Webcam)

3.1.2. Design and interfacing of system architecture:

Development of the system architecture to drive the servo motors MG996r which are capable of rotating from 0-180 and 180-0 as per the command received from ESP32 WROOM Module. The interfacing of the controller and other peripheral devices are driven with an active power DC source along with a servo driver to drive the servo motor with stable PWM (50Hz) input under the controlled signal to control the servo within its operating voltage & ampere rating between 4.2v-6v at 500mV.

The Main control strategy for controlling servo movement by generating compatible PWM out put via ESP32 WROOM module to rotate a specific angle received from python script running parallelly on the laptop and detecting realtime angles of the finger. The precised angular data ranging from 0-180 is being transferring from python to C++ via serial communication UART Protocol running at 9600 boud rate.

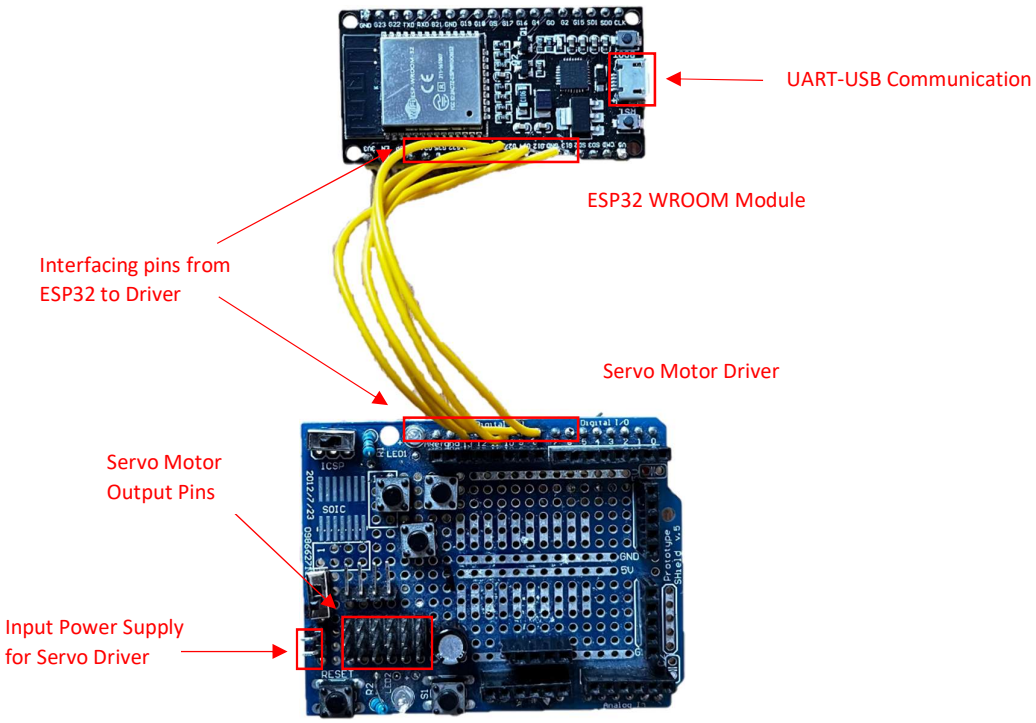

Image 1: System Architecture and Interfacing *[source: own]*

3.1.3. Installation & Pin Configuration of Servo Motors:

Mechanical integration: This project has used MG996r servo motor with 0-180 rotation. Which has feedback loop enabled architecture to measure the precise agnle shaft rotation. Need to make sure that motor has been perfectly aligned in to the servo brackets in robotic hand stand.

Electrical connection: Servo motor model MG996r has pin configuration as Vcc, GND & Signal Pin. Which has to be connected to the dedicated Port number on ESP32 WROOM to control the movement of servo motor as per the signals sent from ESP32 WROOM to respective servo Motor.

Calibration Process: Calibration of the servo is essential step for ensuring that Servo motor will rotate in precise motion and smooth rotation of servomotor. In practical application of servo motor we provided customized operating dead angle zones where servo must not operates. That allows servo motor to operates from range (10-170).

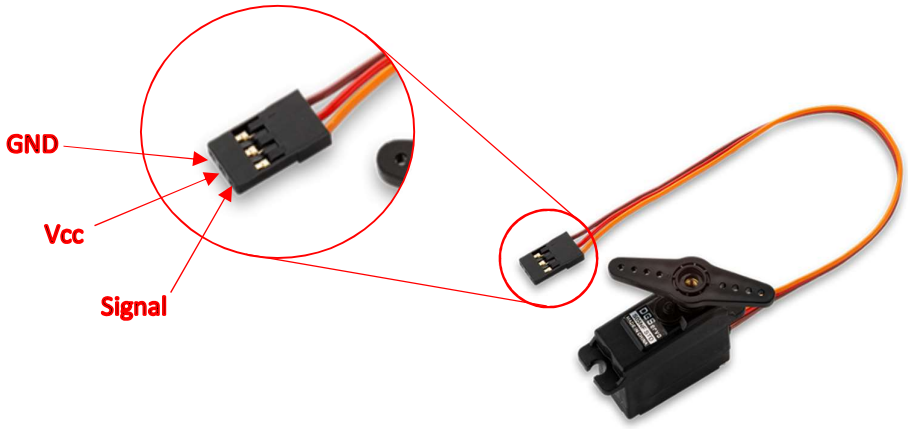

Image 2: Servo Motor Pin Configuration [source: 30]

Specifications

- Weight: 55g
- Dimension: 40.7mm X 19.7mm X 42.9mm
- Stall Torque: 9.4 kg-cm (4.8V); 11 kg-cm (6V)
- Operating Speed: 0.23sec/60degree (4.8V); 0.2sec/60degree (6.0V)
- Operating Voltage: 4.8V ~ 6.6V
- Gear Type: Metal gear
- Temperature Range: 0°C - 55°C
- Dead Band Width: 1us
- Servo Wire Length: 32cm
- Current Draw at idle: 10mA
- No Load Operating Current: 170mA
- Stall Current: 1.4A
- Servo Arms and Screws included

Image 3: Servo Motor Specification Sheet [source:30]

3.1.4. Interfacing & Configuring ESP32 WROOM Module:

Firmware development and IDE Support: The development of firmware for ESP32 WROOM module supported by Arduino IDE which helps to program the firmware and implement the control algorithms for external peripherals for communication protocol and control algorithms.

Image 4: ESP32 Pin Configuration [source:30]

The communication protocol and data transfer channel are set to be serial UART communication for this project. The communication protocol is set up in between Python script running on a laptop and ESP32 WROOM module via a micro USB module. The protocol is configured for this project's UART communication is COM port 5 with a baud rate of 9600.

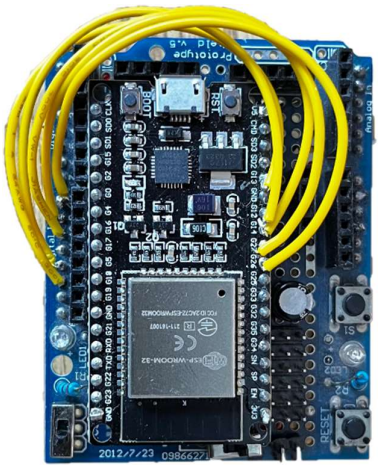

Image 5: ESP32 Servo Driver Interface [source: own]

The ESP32 WROOM module is interfaced with a DC power source to get power up and configured and interfaced pins for servo operation are [G11, G12, G13, G25, and G27].

3.1.5. Interfacing & Configuring Servo Driver:

The servo motor driver is an essential component to drive the servo with the desired PWM frequency and constant controlled input signal to achieve the desired angle of the servo motor.

As soon as the servo motor receives a PWM signal from the ESP32 WROOM module signals have been sent to the servo drive for signal amplification and stable servo signals. Once signals are refined in the servo driver those signals are sent to the servo motor. So that, the servomotor can accept the signals and ask the motor to rotate in continuous rotation along with that potentiometer meter starts to measure the angle rotated by the motor, and as soon as the motor rotates to the desired angle potentiometer starts sending a signal to the servo driver to stop the input signal for drive the motor[2].

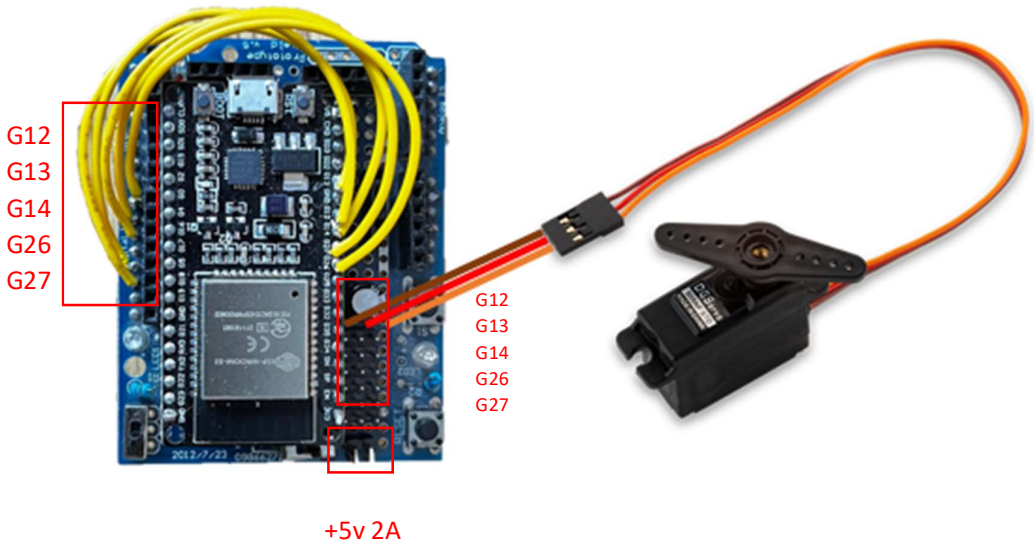

Image 6: Servo Driver Motor Interface [source: own, 30]

3.1.6. Power Source configuration and specification:

This system architecture has two power inputs separately to power up the ESP32WROOM module and Servo Driver. The input power specification for these two individual modules is defined as 4.2V for the ESP32 WROOM Module and 4.2-6.0V required to power up the servo driver with 5 servo motors interfaced with the driver. The range of 4.2-6V for servo motor drivers mainly depends upon the torque and speed needed for servo.

3.1.7. System architecture and component interfacing:

All the components used in this project such as the Microcontroller ESP32 WROOM module, Servo Driver to drive the servo Motor, Power Unit, Servo motor along with all other wiring and communication channels, etc. The main crucial module of this architecture is the ESP3 WROOM microcontroller which is responsible for receiving commands from the laptop via the URAT COM5 Port.

Once signals are received from the laptop to the ESP32 module C++ will take over the signal processing and assign and generate individual signals for the servo motor to rotate by specific angles for each Servo motor.

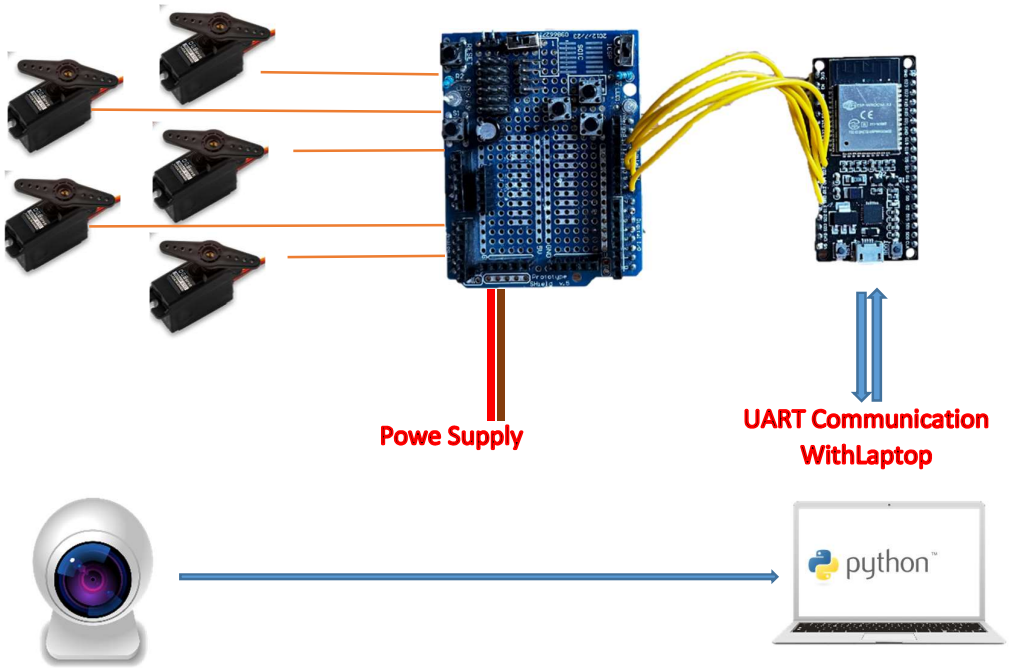

Image 7: System architecture [source: 30]

3.2. Designing and building strategic approach for Computer Vision in Python Programming.

3.2.1. Evolution & Implementation of a computer program.

Computers are a very crucial part. Where the world is progressing with the advancement of computers and automation. When the word automation word comes into the picture, the computer plays a vital role in controlling all the tasks and functionalities of automation.

Industry 4.0 is an autonomous factory that deals with fast-paced production, manufacturing, and Delivery of finished products.

Computer networks and computer intercommunication lead the automation process with a set of factory instructions. Autonomous sets of instructions play a vital role in using smart vision and Data Analysis techniques to carry out tasks[11].

Computer vision is one of the computer programs that enables humans to implement Image processing, Object Detection, And Object tracking capabilities. These capabilities would be an asset to make machines understand the task to execute by visual gestures.

3.2.2. Design and Development of Computer Vision Algorithm

The computer vision algorithm has four basic fundamental pillars to execute. The designing of the algorithm has been started with the declaration of required libraries. Library used are OpenCV, MediaPipe, Serial, Numpy, Matplotlib, etc[13].

The next segment of the code is dedicated to initializing the computer camera to fetch every frame that the camera can detect in the loop. After detection, every frame will be taken into consideration for preprocessing.

In the Preprocessing step, the camera frame will be analyzed fully to detect the entities with two parameters such as detection confidence and Tracking detection.

In the next segment of code, preprocessed frames of the camera will be processed under an object detection algorithm and object tracking capabilities with specific confidence values.

After detection and tracking implementation on the frame, relevant information is fetched from each frame in a loop.

Based on desired values gathered from each frame, we process the data or modify the data with some relevant mathematical computational techniques[1].

Finally, we bundle the data in appropriate data blocks and send it to the relevant communication to other peers in the network.

3.2.3. Strategic planning for implementing Computer Vision

The computer vision algorithm used for this project is based on the detection of the hand gestures along with some parameters of the hand such as the angles of each finger and speed of rotation of each finger[15].

The process of fetching angles and speed of the finger is only possible after projecting skeleton landmarks on the hand. These landmarks are used to extract useful information from human hands.

I have used the Hand Detection model from the MediaPipe library which allows us to superimpose skeleton data points on hand. I used the same data to process further angle calculations using the vector approach.

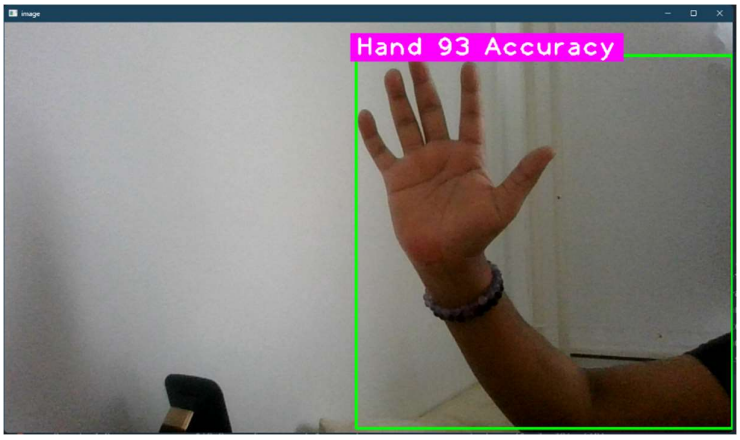

Image 8: Hand Detected by Computer Vision [source: own]

The skeleton hand detection approach uses a total of 21 landmark points on a detected human hand to generate a skeleton to superimpose on the real camera frames fetched by the camera. Based on desired values gathered from each frame we process the data and modify the data with relevant mathematical vector computational techniques. Finally, we bundle the data in appropriate data blocks and send it to the relevant communication to other peers in the network.

Computer vision is one of the computer programs that allow people to realize image processing, object detection, and tracking functions. These features are useful when a machine uses visual gestures to understand what to do. The next part of the algorithm is used for the interaction of the model with the preprocessed video frames. The preprocessed video frames are passed to the next image-processing algorithm controlled by the MediaPipe library[2].

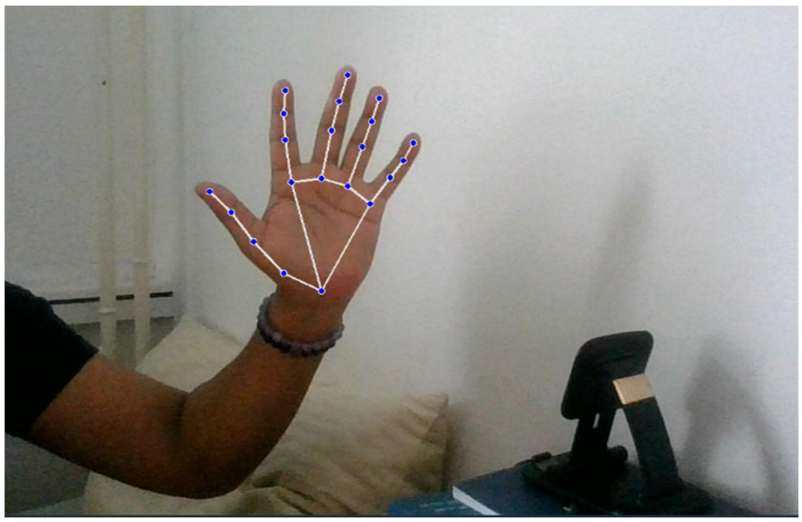

Image 9: Post Processing Hand Detected by Computer Vision [source: own]

3.2.4. Fundamental Skeleton of Python Program

1. Initializing & Declaration of Libraries.
2. Initialize MediaPipe Solution.
3. Initializing Serial Communication.
4. Defining Utility Function.
5. Main Program Loop.
6. Release Resources.

3.2.5. Libraries and Functions

1. MediaPipe: Used for hand Tracing & Landmark Detection.
2. OpenCV (cv2): Used for webcam access & Processing webcam feed.
3. Numpy: Handling numerical operation.
4. Serial: Establishing serial communication channel between Python & C++.

3.2.6. Architecture for Image Detection & Tracking

The architecture of image detection and object tracking is typically implemented with the first step which is data acquisition received from the webcam. The camera footage is first processed with the OpenCV library to convert it into video in multiple image frames.

```
cap = cv2.VideoCapture(0)
cap.set(3, 1080)
cap.set(4, 1920)
```

Code Snippet [source: own]

This input undergoes preprocessing for quality enhancement and conversion into a suitable format. The detection algorithm in code is usually carried out by locating the predefined objects that are being asked in the data library of code to find in the real-time footage. While tracking the object the same predefined object detected in the virtual frame or a boundary has been made to locate the object into the frame and each frame goes into preprocessing to update the detection frame for the same object. If we continue the same process in a loop until the object lies within the frame then it declares as object tracking[1].

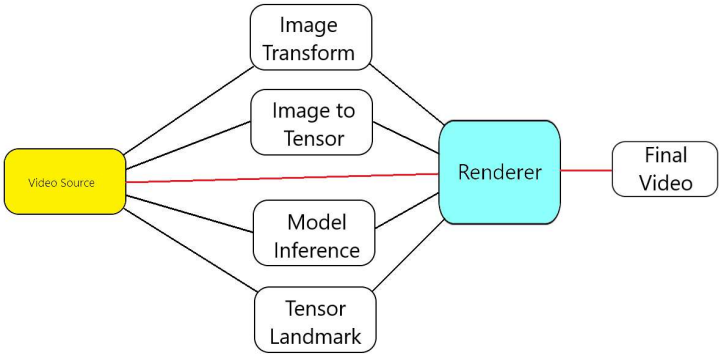

Image 10: OpenCV & MediaPipe Architecture for Video Rendering. [souce: own]

Computer vision is one of the computer programs that enables humans to implement Image processing, Object Detection, And Object tracking capabilities. These capabilities would be an asset to make machines understand the task to execute by visual gestures.

The next Segment of the algorithm deals with model interference with preprocessed video frames. The preprocessed video frames will delivered to the next image processing algorithm which is governed by MediaPipe Library.

3.2.7. Superimposition of landmarks on detected hand

The OpenCV library uses the Hand Detection Module to gain access to the webcam and fetch the video footage. This enables Python code to operate the data received from the webcam of the system and access each video frame with a suitable resolution. After accessing each video frame separately OpenCV processes the frame to enhance the quality and color gradient scheme to a desired level which would make it easy to render the same video frame footage with a computer vision model to execute over the same.

The next Segment of the algorithm deals with model interference with preprocessed video frames. The preprocessed video frames will delivered to the next image processing algorithm which is governed by MediaPipe Library[3].

The MediaPipe is meant for plotting desired landmarks on the detected object such as Hand, Face, Leg, Human Body, etc. This would help to extract important useful information from the detected object. This information again ahead we can use it for net post-processing to find useful results.

In this project, we have used Hand Detection Landmark which is rendered with real-time footage that got preprocessed with the OpenCV library and then passed to this MediaPipe module.

The final rendered footage of the video would have overlaid landmarks on the screen with skeleton behavior connected to each node worth a linear link between them. Object tracking finds objects in frames by generating identical predefined objects detected at virtual frames or boundaries, and each frame goes into preprocessing to update detection frames for the same objects. We continue the same process in a loop until the object gets inside the object and it is declared as a tracking object.

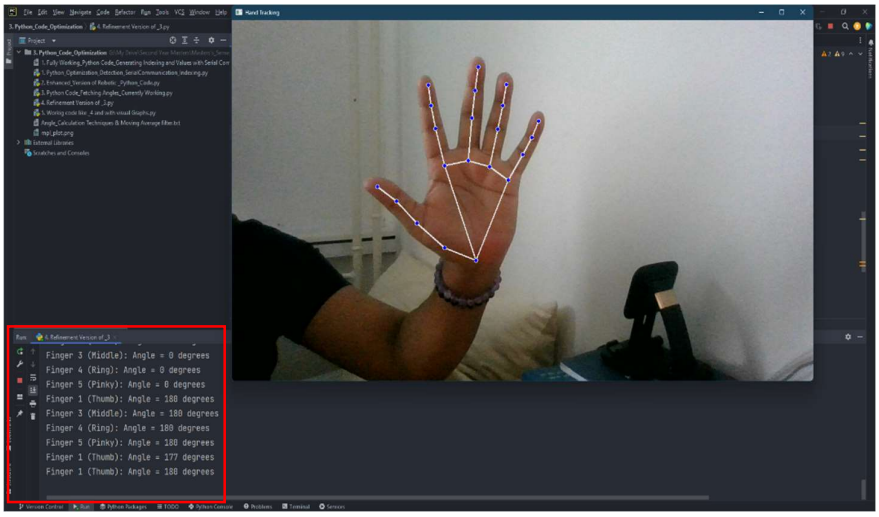

Image 11: Skeleton Landmark plot rendered real-time footage with angle. [source: own]

3.2.8. Implementation of Hand Gesture Recognition

The implementation of the hand recognition algorithms works on the basic principle of scanning and locating landmarks in specific shapes. In this project, we are using a hand detection module by MediaPipe which helps to identify the gestures of hand fingers and map the predefined actions to the gestures.

$ 0 0 0 1 0					
Gesture	**Pinky**	**Ring**	**Middle**	**Index**	**Thumb**
Zero	0	0	0	0	0
One	0	0	0	1	0
Two	0	0	1	1	0
Three	0	1	1	1	0
Four	1	1	1	1	0
Five	1	1	1	1	1

Image 12: Communication Array with State & Servo Number [source: own]

Initially, with the starting of the algorithm, we designed and implemented a binary value mapping technique where we mapped every finger state with binary values 0s & 1s. Where 0 denotes with finger state closed finger and 1 indicates the finger up state these values we saved and bundled in the form of an array with an array length of 5.

Each finger number denotes the index number of the array starting from 0 to 4 as mentioned size of 5 and the value contained at a specific index number will be denoted as the state of the finger, which would vary from 0 to 1 as open and closed.

3.2.9. Implementation of fetching hand attributes

This project aims to fetch the necessary attributes from gestures made by human hands. These attributes would later be very useful to consider as features for training algorithms and performing specific tasks.

In this project, we are using features such as angle as the degree of each finger and the speed of the finger moving at a velocity V. These parameters will later be used for further calculation for performing tasks.

3.2.10. Processing and calculating desired angles

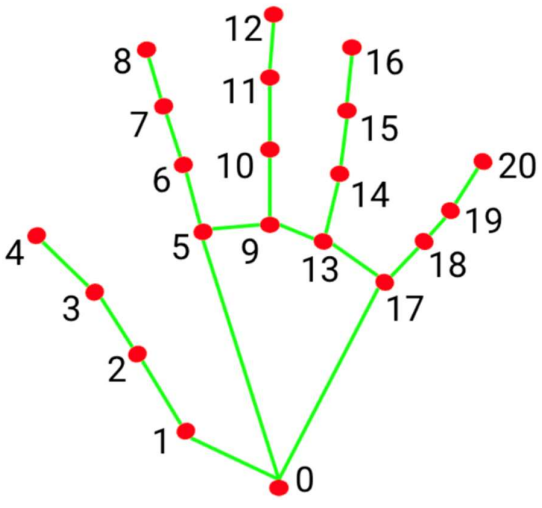

Image 13: Ploted Landmarks on hand skeleton on Human Hand. [source: 31]

We used OpenCV and media pipe libraries to access the webcam footage and then considered each frame for processing each frame to draw skeletons ver the human hand and plot landmarks on a human hand.

The total number of landmarks I am dealing with is 21 points spread over the hand and with the help of these landmarks, we performed the calculation by using the vector approach.

In the Python program, I have defined the landmarks that we have to take into consideration and perform further calculations

First, we consider landmarks which help to calculate the angle of fingers efficiently. For the selection of landmarks we used 3 consecutive landmarks and the angle can be found at the middle landmark.

Thumb	2	3	4
Index	5	6	7
Middle	9	10	11
Ring	13	14	15
Pinky	17	18	19

Image 14: Ploted Landmarks on hand skeleton on Human Hand. [source: own]

```
def draw_finger_angles(image, hand_landmarks, joint_list):
    global previous_angles
    finger_names = ['Thumb', 'Index', 'Middle', 'Ring', 'Pinky']
    for idx, (joint, name) in enumerate(zip(joint_list, finger_names)):
        a = np.array([hand_landmarks.landmark[joint[0]].x,
hand_landmarks.landmark[joint[0]].y])
        b = np.array([hand_landmarks.landmark[joint[1]].x,
hand_landmarks.landmark[joint[1]].y])
        c = np.array([hand_landmarks.landmark[joint[2]].x,
hand_landmarks.landmark[joint[2]].y])
        angle = np.abs(np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1],
a[0]-b[0])) * 180 / np.pi
        angle = 360 - angle if angle > 180 else angle
        normalized_angle = normalize_angle(angle, 10, 170)

joint_list = [[2, 3, 4], [5, 6, 7], [9, 10, 11], [13, 14, 15], [17, 18, 19]]
```

Code Snippet [source: own]

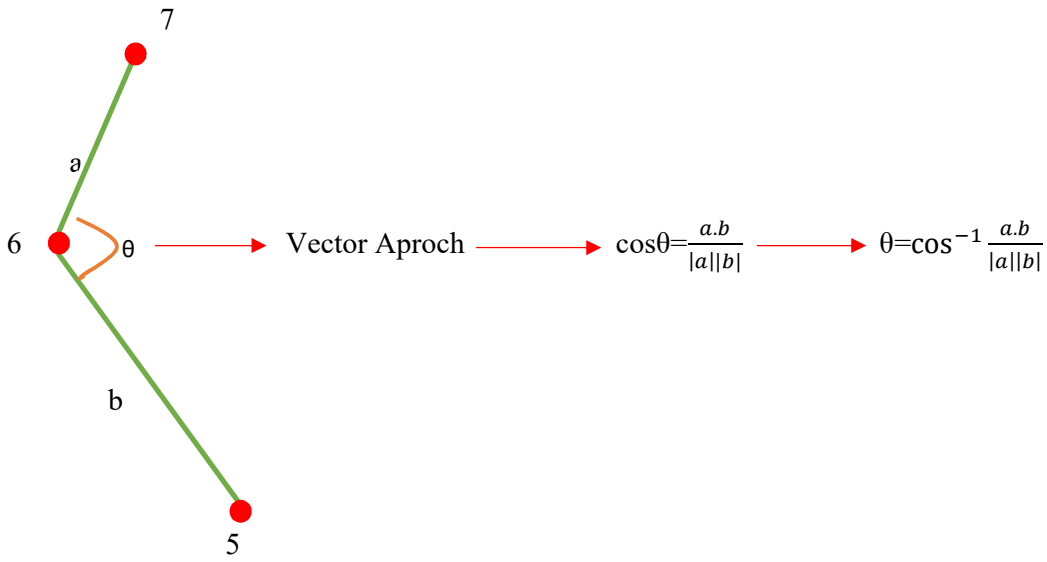

Image 15: Vector Calculation from detected gecture of bonic skeleton. [source: own]

3.2.11. Formation of arrays with desired angles

The above shown vector approach for angle calculation would be used for angle determination and rate of change of angle per unit time.

These angles will saved in an array of size 5 starting from 0-4. Every index represents each finger and the data stored at every index will be in the range of 0-180 degrees. This data will then trnsfered to the c++ code via serial communication with a predefined baud rate of 9600 via COM5.

Index	0	1	2	3	4
Angle	**30**	**175**	**56**	**48**	**126**

Image 16: Arrays data Packet formation from detected gesture of bonic skeleton. [source: own]

Finally, we bundle the data in appropriate data blocks and send it to the relevant communication to other peers in the network.

3.2.12. Establishing Serial Communication with ESP32

Inially we declare libraries which support for the serial communication called as “serial’. After declaration we initialize the serial library with declaration COM5 port and baudrate needed that is 9600 for sending data packets via serial communication.

To make the C++ code robust to handle errors and issues during serial communication, code algorithm has implemented If statement in code to through an error if COM5 is disconnected or the data packets got collided with each other and makes the code freeze.

```
try:
    ser = serial.Serial('COM5', 9600)  # Adjust the
port and baud rate as needed
except serial.SerialException as e:
    print(f"Error: {e}")
    exit(1)
```

Code Snippet [source: own]

3.2.13. Data Encryption & Transmission Strategy

Once the seial communication protocol declared. Now it allows us to send via COM5 with a predefined baud rate of 9600.

```
def send_serial_data(servo_num, angle):
    try:
        ser.write(f"{servo_num} {angle}\n".encode())
    except serial.SerialException as e:
        print(f"Error: {e}")


if abs(previous_angles[name] - normalized_angle) >=
3:
    send_serial_data(idx+1, normalized_angle)
    previous_angles[name] = normalized_angle

ser.close()
```

Code Snippet [source: own]

After the sending operation done with the serial communication command we can close serial communication properly. This is a crucial for releasing resources and avoiding issues with subsequent connection.

3.2.14. Development of a C++ code for the ESP32 Module

In machines, C++ is a widely used low-level programming language. Because of its time complexity and ease of compiling at high speed. This programming language is very useful for designing and programming microcontrollers to perform specific tasks.

Here in this project, we are using the ESP32 WROOM module to drive the servo motors. This microcontroller is powered by the ESP32 processor combines a CPU with 2 Tensilica LX6 cores, clocked at up to 240 MHz, and 512 kilobytes of SRAM in a single microcontroller chip. This enables the module to hand multiple sensors and actuators to work in a multitasking protocol with 512kb RAM management[15].

For this C++ code for programming, we have used Arduino IDE for programming the ESP32 WROOM module. Where we have two sections in a code.

3.2.15. Establishing C++ Serial Data Communication

In C++ Code, defined a Serial communication protocol for receiving data packets from COM5 with a suitable baud rate of 9600. The data packets are in the form of an array with array size 5 starting from 0 to 4. Each index number denotes the finger number and values contained at each index number will denote the angle of the respective finger.

These array data packets are kept on updating with a certain frequency to operate the servo motors. To operate the servo motor C++ program has to generate certain PWM signals to control the position precision and speed of the servo motor.

3.2.16. Data Decryption Strategy after receiving the data

The C++ code defines a serial communication protocol for receiving data packets from COM5 with a reasonable baud rate of 9600. Atr data packets are an array of array size 5 starting from

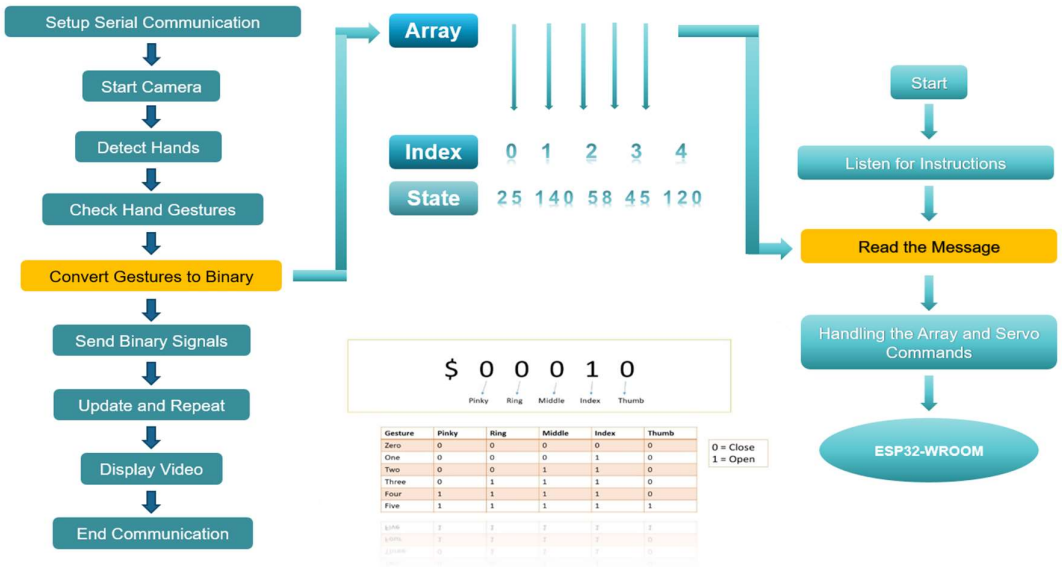

Image 17: Communication Strategy between Python and C++.[source: own]

0 to 4. Each index number represents a finger number, and the value contained in each index number represents the angle of that finger. These array data packets are continuously updated at a specific frequency to control the servo motor. To operate a servo motor, a C program must generate specific PWM signals to control the position accuracy and speed of the servo motor.

3.3. Mechanical Design and Fabrication

3.3.1. Modelling and Designing of Mechanical Structure of Hand

I have designed and customized a predesigned model of a robotic hand for implementing continuous robotic hand movements with fluidity and detexitrity of the robotic hand. There are a total of nine parts that have been modeled and designed for the robotic hand. These robotic hand parts are shown below.

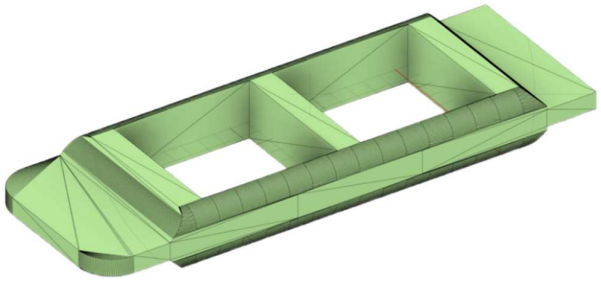

Image 18: Servo Motor Mount. [source: own]

This is a Servo motor mount shown in *Image 18*: which is specifically designed to hold 5 servo motors to operate the movements of finger open and closed actions of fingers. The connection between the servo motor and the finger would be connected via a nonelastic transperant string which takes care to pull action for closing the fingers.

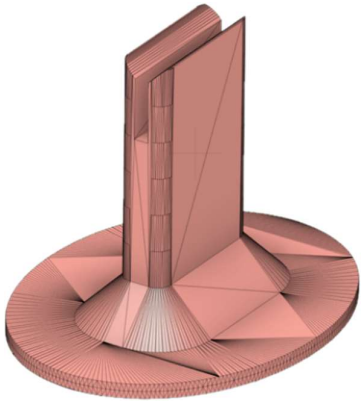

Image 19: Supporting Stand for Robo Hand. [source: own]

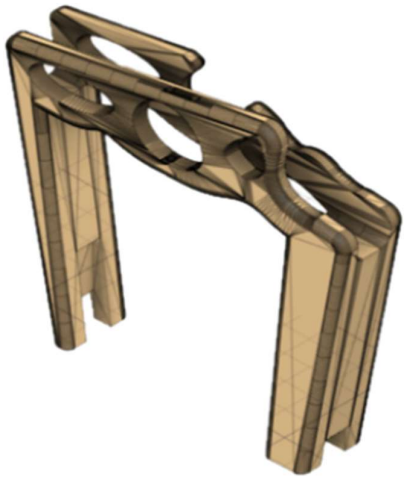

Image 20: Robotic Palm Support. [source: own]

This Robotic hand palm support shown in *Image 20* is specially designed to support the robotic hand palm over the foundation of a robotic hand that is shown in *Image 21*.

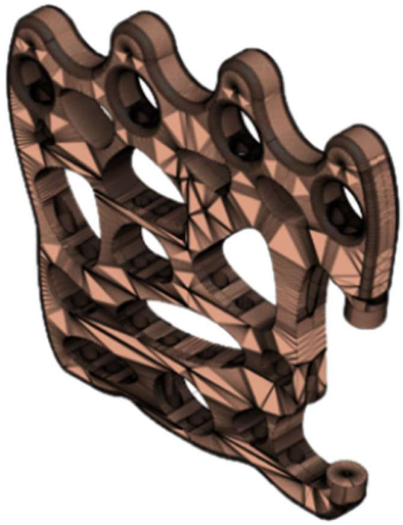

Image 21: Robotic Hand Palm and Finger Mount. [source: own]

This is a palm of robotic hand which has been designed to replicate the human hand anatomy to hold five fingers and also give a rigid support got the human hand anatomy. The aim is to designed in a such way that, the connecting thread guide ways are alredy been created while

modeling in the software which hand handle the pull and controls the action of the servo motor for fingers to operate continuously open and close movement.

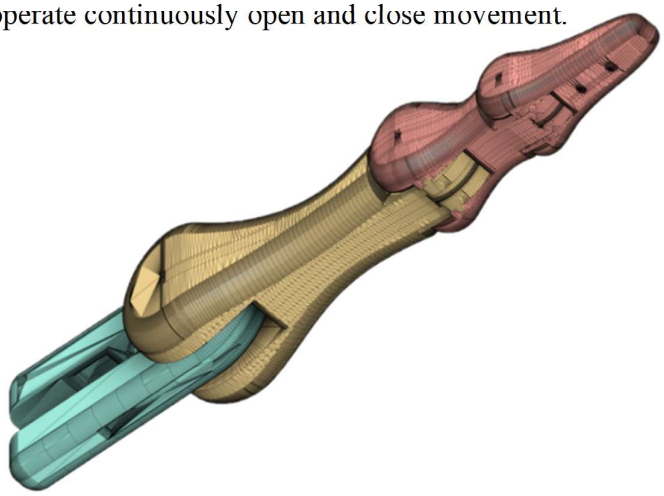

Image 22: Index Finger. [source: own]

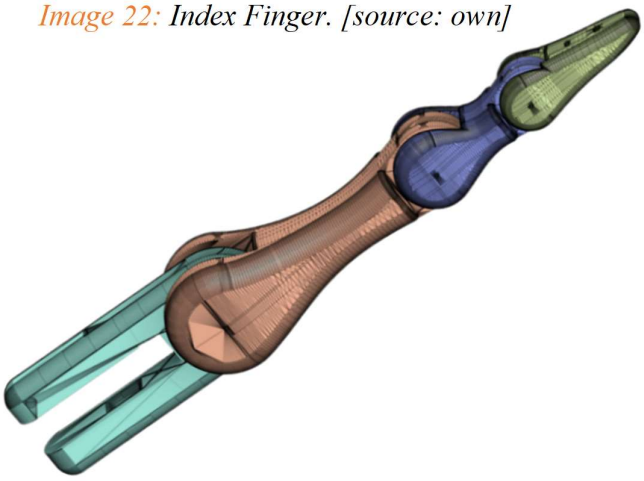

Image 23: Little Finger [source: own]

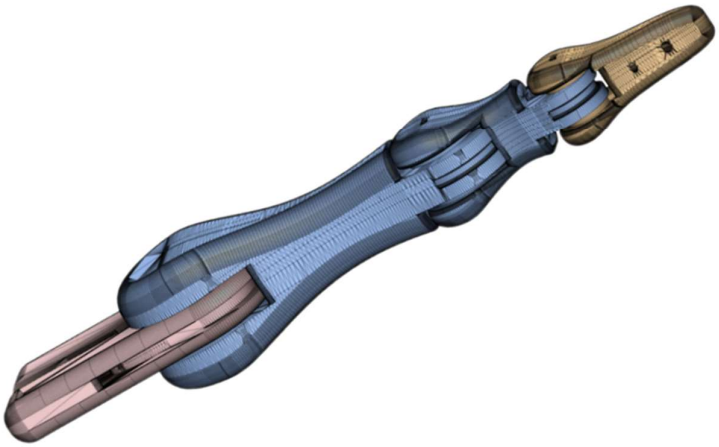

Image 24: Middle Finger [source: own]

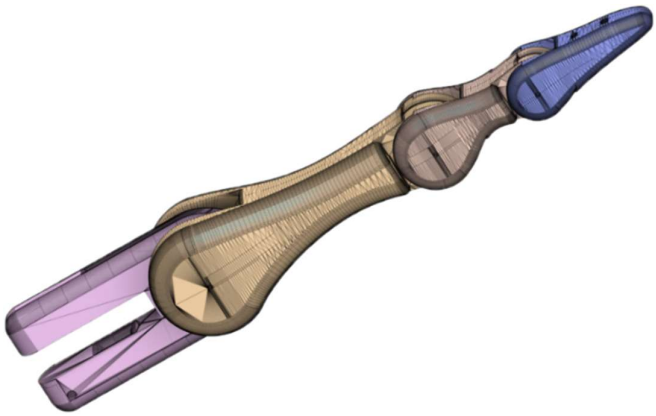

Image 25: Ring Finger [source: own]

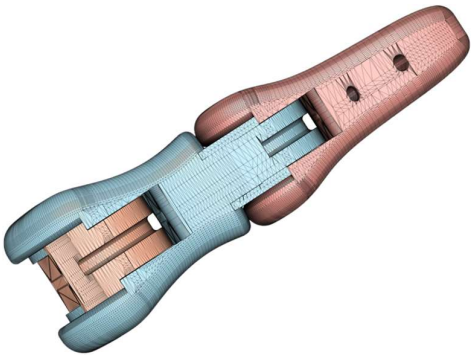

Image 26: Thumb Finger [source: own]

These are the STL versions of the 3D parts of the hand. Which has been designed and modeled on Fusion 360 and printed on a 3D printer with a PLA material Type and 0.4mm Nozzle diameter with a printing precision of 0.1-0.3mm.

3.3.2. 3D Printing of Robotic Hand Parts.

Designing of the robotic hand parts as shown in previous images and I have printed with 3D printer as shown in below images. The image exactly shown *Image 31: Actual 3D printed Palm* is hand palm to hold all fingers and servo mount.

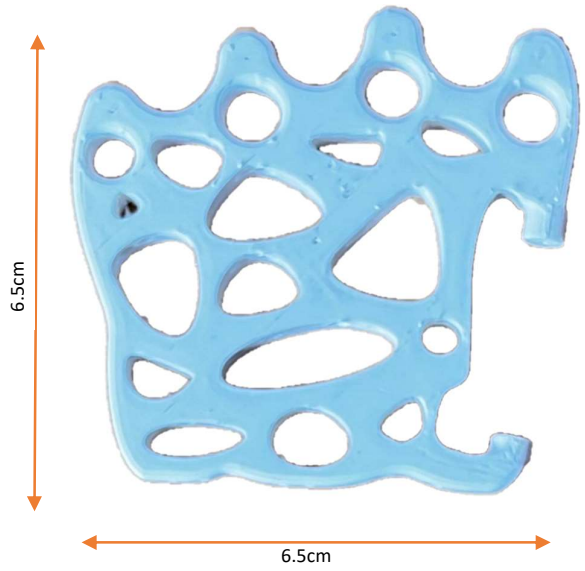

Image 27: Actual 3D printed Palm [source: own]

The image is shown below in *Image 27: Palm Connector* responsible for the palm mount which will be holding hand structure and servo mount.

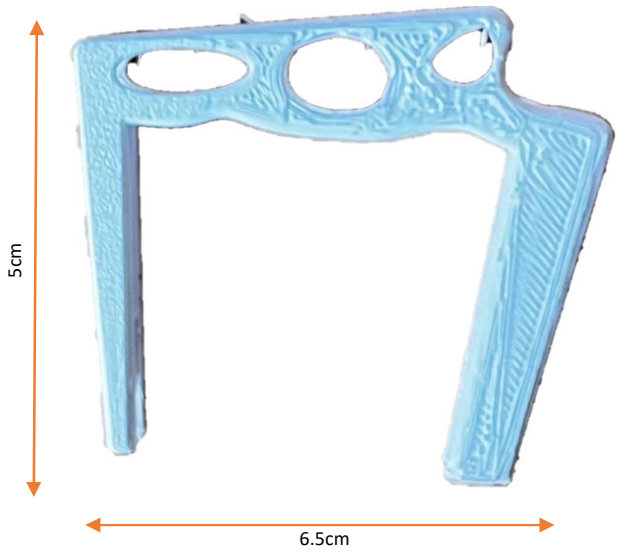

Image 28: Palm Connector [source: own]

The part shown below in *Image 28: Hand Stand this a structure stand that will help to give rigid support for the structure. This stand will be responsible for the holding servo mount and the palm of the robotic hand to hold 5 fingers.*

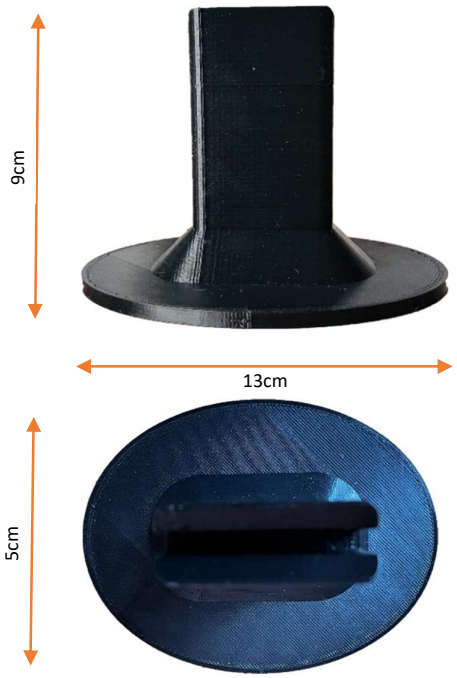

Image 29: Hand Stand [source: own]

This part is a servo mount shown in *Image 29: Servo Mount Which is capable and designed to hold 5 servos with rigid fitting to control fingers which optimizes pull torque.*

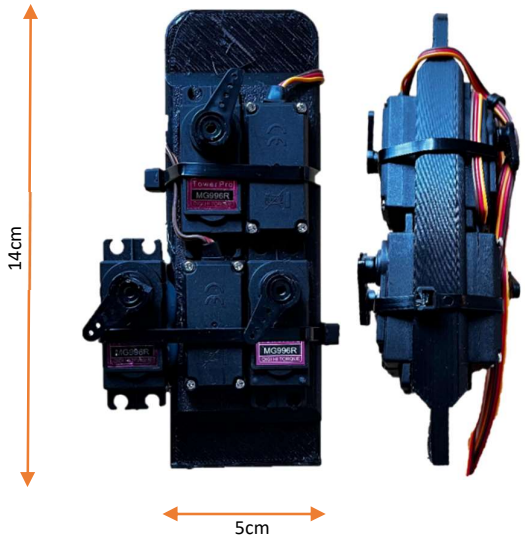

Image 30: Servo Mount [source: own]

3.3.3. Assembly and fabrication of parts.

The complete assembly of the robotic hand is is as shown below in the *Image 31: Final Assembly of Robotic Hand.* This includes all the 3D printer parts that are interconnected with each other for better linkage control between the servo lever and finger. The fingers attached to this robotic hand are 3 fold joint except the thumb which ensures the closing and opening of the finger exactly as per the human finger anatomy. The thumb is designed with 2 fold joint for better controllable human dexterity.

Further, I have modified the servo motor lever with wooden sticks attached to the levers for more control length of the finger when the finger operates between an open and closed state.

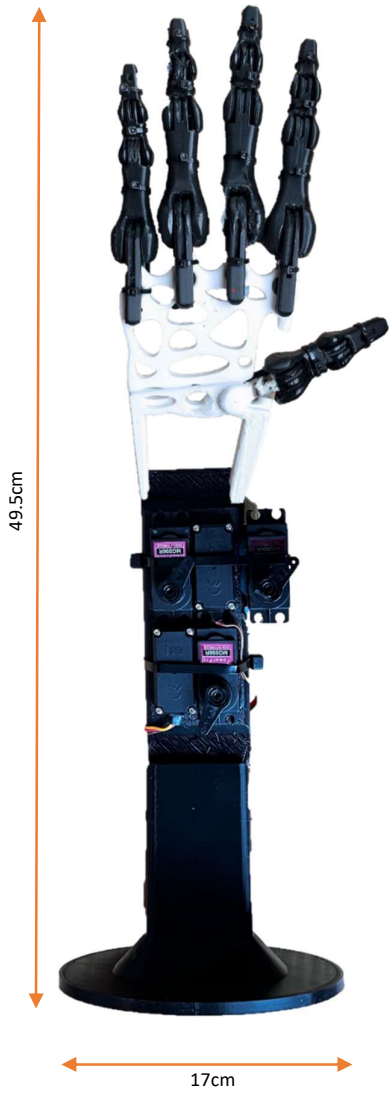

Image 31: Final Assembly of Robotic Hand [source: own]

3.3.4. Mechanical joint connections & tuning of Servo Drive.

This project consists of 3D-printed parts along with some other electric drives. I am using servo motors as an electric drive to operate the movement of fingers precisely. To operate the finger with precision, I have used Digital filters to reduce fluctuations and noise in signals. to operate mechanically between the connection of the servo motor and finger, I am using a nonelastic string that provides active control for pull action for fingers.

In this robotic arm, I am using 5 servo motors for 5 separate fingers. These servo motors are enabled to rotate in 0-180 and 180-0. To get an enhanced precision performance, I have enabled end restriction for servo movement from 10-170 degrees.

3D printed hand has five fingers, each finger has 3 joints. Every joint allows for rotation of the finger with 30 degrees at the top joint, 60 degrees with the lower joint, and 90 degrees with the bottom joint. This angle of rotation allows the finger to total 180 degrees. The precise angle of rotation is governed by a string connected along with all three joints and with the servo motor lever which is 2 cm in length.

As the signals from the servo driver are received by the servo motor then the servo motor rotates with the desired angle and the thread connected between these finger joints will rotate in with a desired curve of a finger. Once this operation is completed the finger would go back to its original position with an elastic strip installed at the back side of each finger which helps the finger to regain its initial position at 180 degrees.

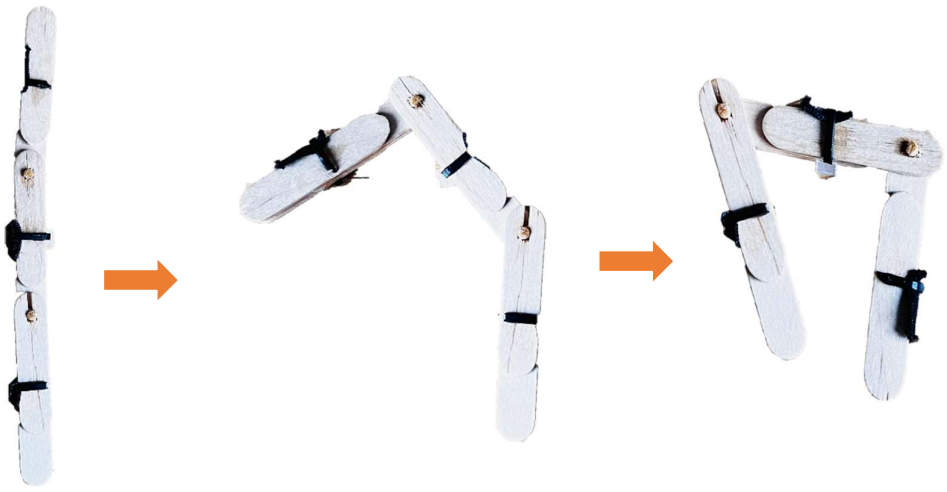

Image 32: Finger Folding Mechanism Tested on Wooden Prototype.
In 3 stages of folding. [source: own]

4. Results

4.1. Detection Accuracy and Tracking Confidence

The detection of objects is a challenging task to detect the object with the highest accuracy like human beings. This detection process needs high computational power and a good preprocessing approach in order to find and consider relevant features from the object. In this project, implemented an object detection algorithm with the help of OpenCV which allows us a capable environment where we can program an object detection algorithm and tacking of the same with requires preprocessing of the video frames captured from the webcam.

Here in this project, implemented an object detection with a threshold value of 0.8 confidence which allows us to detect the object with an accuracy up to 80% and above. If any object is detected with a low accuracy confidence eg. Less than 80% it won't be labeled with a desired category of the objects from the code.

Once the object is detected and continuously tracked with tracking threshold confidence we could conduct statistical analysis on the object and post-processing of the object. The post-processing techniques taking care of the desired result are plotting exoskeleton landmarks on the human hand detected in the camera. This skeleton accuracy depends on the number of landmarks plotted on the hand. Here in this project, used 21 landmark points to plot over the hand to conduct vector calculations to find angles. These angles I have plotted these on a bar chart to a representation of the angle of each finger in real-time and accuracy achieved around 80% with 0.8 confidence in tracking the frame.

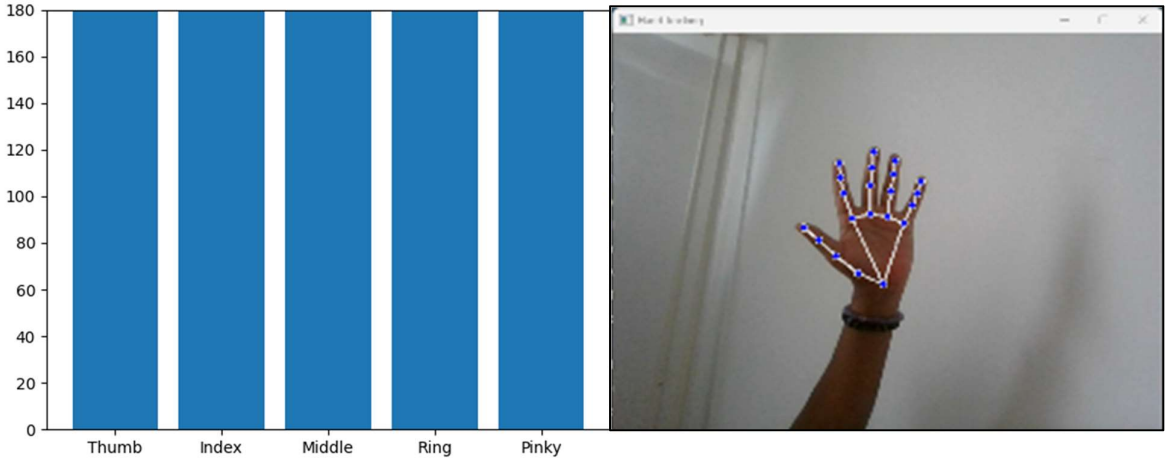

Image 33:Angle Representation of fingers. [source: own]

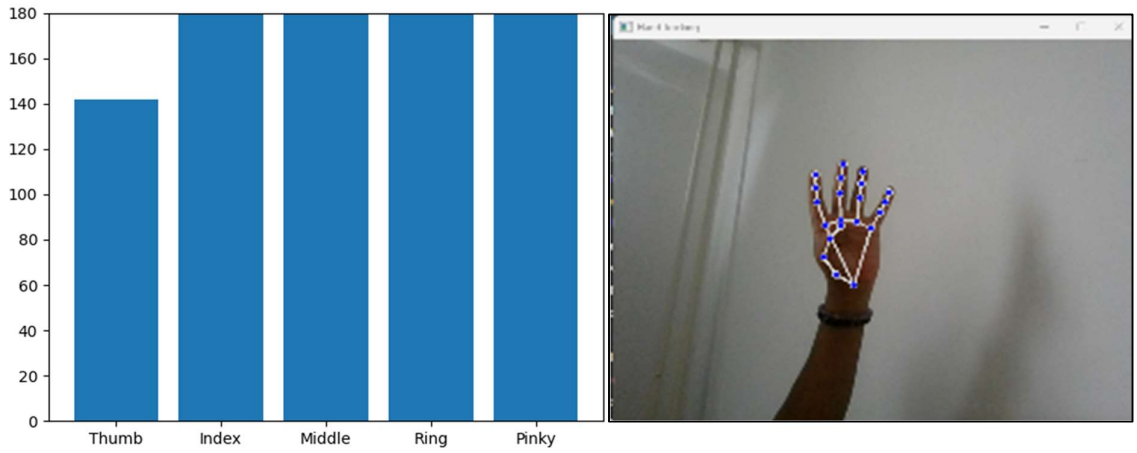

Image 34:Angle Representation of Thumb Finger. [source: own]

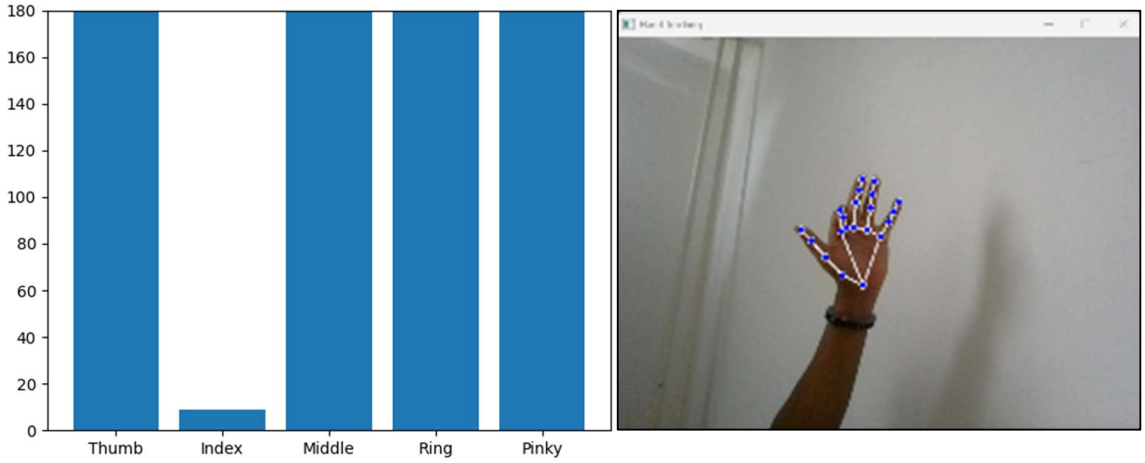

Image 35:Angle Representation of Index fingers. [source: own]

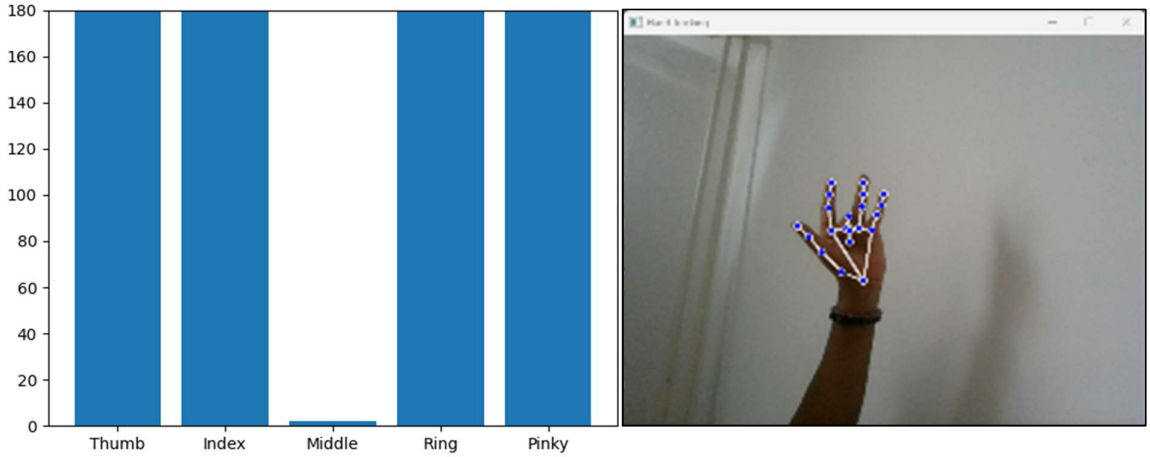

Image 36:Angle Representation of Middle fingers. [source: own]

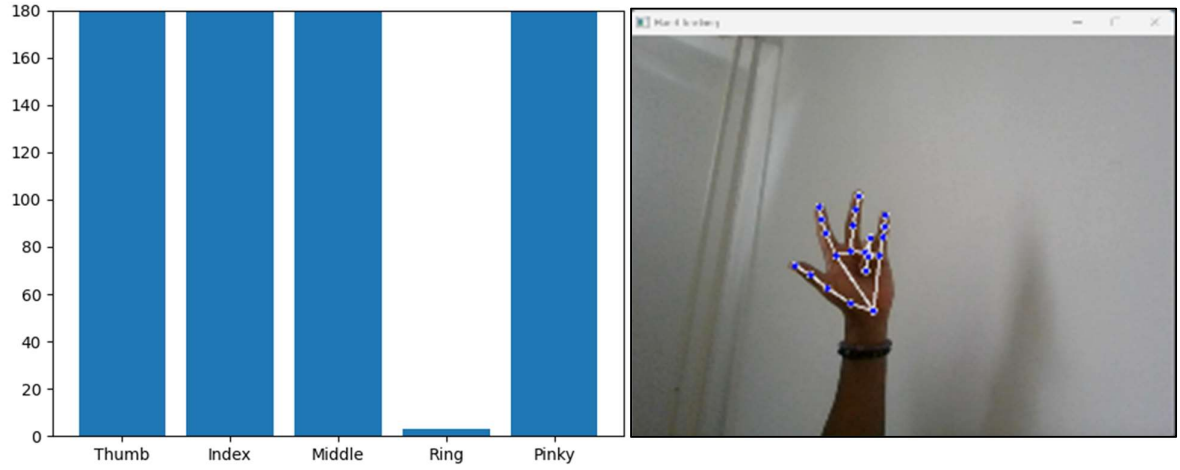

Image 37:Angle Representation of Ring fingers. [source: own]

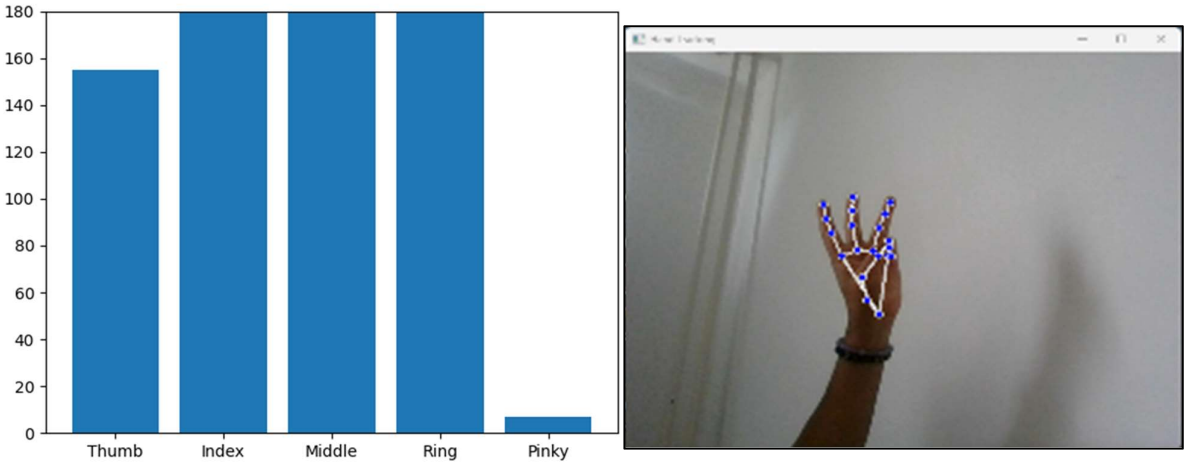

Image 38:Angle Representation of Pinky fingers. [source: own]

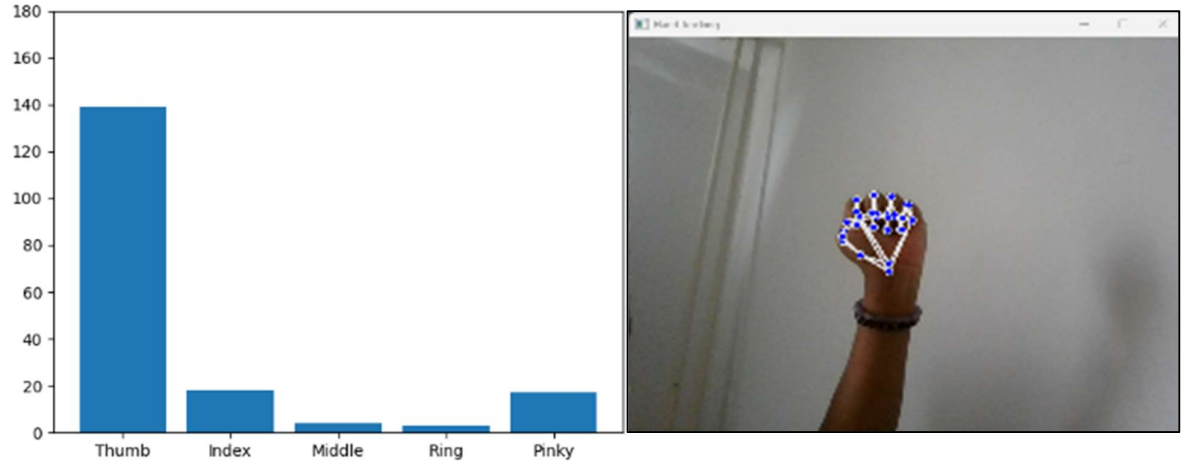

Image 39:Angle Representation of fingers. [source: own]

4.2. Angle – Frequncy chart to demonstrate finger movement.

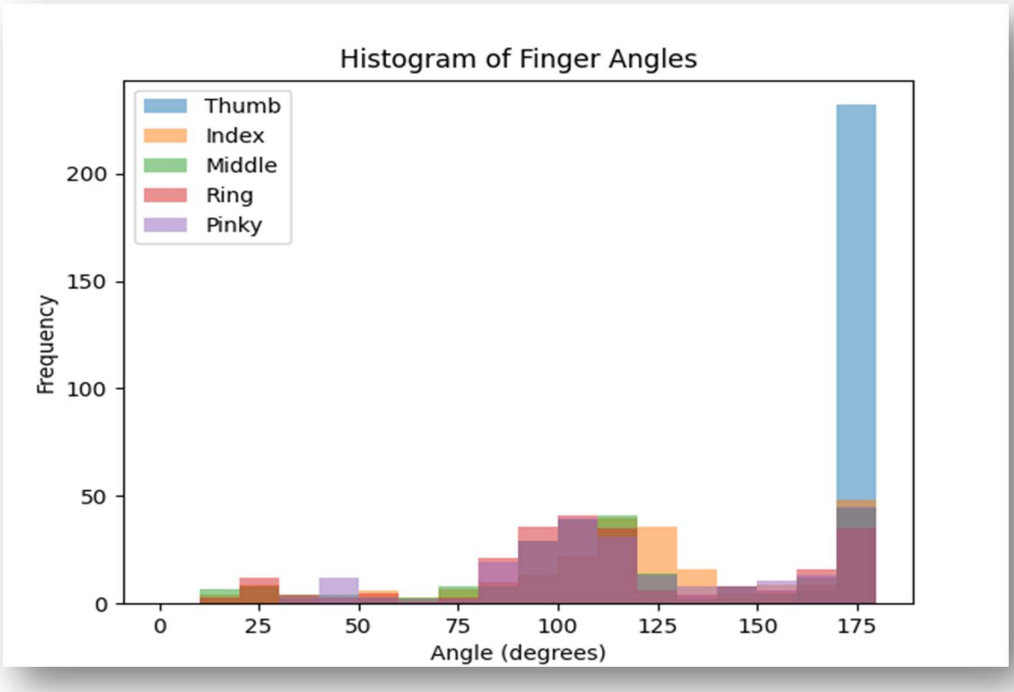

Image 40:Angle Frequency Chart. [source: own]

This particular histogram represents the angular frequency of a particular finger at a specific angle. As you can see in *Image 40: Angle Frequency Chart.* X The axis represents an angle in degrees and Y-axis represents frequency and the color of the histogram represents the type of fingers. There are a total of five colors represented in the above histogram which shows five fingers.

At the time when this image was captured the states of the graph represent that code started with all fingers detected at 180 degrees and later on most of the finger operations were made within the range of 75 degrees to 150 degrees.

As you can see in the above chart there are fewer agular finger movements detected at the lower range of the finger from 15 degrees to 75 degrees and no finger has been achieved 0 degrees till the above plot is captured.

This specific plot is very useful when we want to track the motion of fingers or any object with respective to specific mathematical parameters as here in my case, I am tracking the motion of the finger at every angle and calculating the frequency of finger angle achieved that same angle which will beneficial to track the motion of finger in particular angular range. And operation angular report.

5. Discussion

5.1 Interpretation of Results and Implications:

Once the robotic arm was assembled, the finger movements worked perfectly. Researched and performed extensive testing and analysis of signals generated from computer vision algorithms and serial communication servo motor signal receivers. The robotic arm control operation worked flawlessly with smooth communication and minimal latency for real-time operation. I have achieved novelty in a project that involves continuous control of robotic finger movements. This has not been done before today's literature in this area. During the initial testing phase, we encountered some issues, such as a lack of signal filtering. This resulted in increased variation in the signal generated by the servo motor, resulting in the effects of the servo motor jitter. Another problem I face is that the robot's fingers are not smooth enough. We also need to configure and set parameters to control the closed and open states of the fingers, which are displayed in the range 0 to 180 depending on the servo angle.

5.2 Analysis of Challenges and Proposed Solutions:

The robotic arm control operation worked flawlessly, providing seamless communication and minimal latency for real-time operations. I achieved novelty in the project. It is to control the movement of robotic fingers in continuous mode. Work that had not yet been done before today's literature in this field. During the initial testing phase, we encountered several challenges, including a lack of signal filtering, which increased the variation in the signals produced by the servo motors and created jitter effects on the servo motors. Another problem I faced is that the robot's fingers are not smooth enough. We also need to configure and set parameters to control the closed and open states of the fingers, which range from 0 to 180 depending on the angle of the servo.

Challenges encountered:

1. Establishing a Real-time Communication Channel between Python and C++.

I started this project with real-time implementation of communication between Python and C++ by passing string/Char from C++ - to Python or vice versa. Ahead in time, I succeeded in communication. I established communication to control servo one motor by passing string input. Then tried multiple servos with binary control protocol. 0 denoted 0 degreed state of the servo motor and 1 denoted 180 degrees of the servo motor. Finally, I was able to pass random values from 0 to 180 in the form of an array and was able to control the servo motor.

2. Designing and implementing Object Detection and Tracking Strategy.

Initially build object detection algorithm with the predefined library by open CV to know the working principle of computer vision detection algorithm. Once I was able to establish the detected approach I built a customized code to plot skeleton landmarks on the detected hand which will be overlying over the original hand footage to perform further computation requirements.

In the first attempt artificial skeleton landmark was not exactly overlying on the real footage of the hand which resulted in difficulties in performing further calculations.

So I adjusted a few color and How parameters by converting it to greyscale and analyzed histogram output for better detection accuracy of hand. Finally resolved that the exoskeleton structure was able to plot on original hand footage.

3. Angle Calculation from gesture detected.

Designed and performed a vector approach to calculate the angular calculation for finger states. In this computation step, I took the help of landmarks plotted on original footage and found out the distance between 3 nodes and performed angular calculation but it was lacking in the angular calculation as the skeleton landmark was continuously moving and hence values of angle kept on fluctuating rigorously which related in servo motor jitter effect.

To optimize the detected angle value and less fluctuation in servo angle values I used a moving average filter with the size of 12. Which filtered out the fluctuation in the calculation and resulted in smooth operation in servo movement.

4. Post-3D printed part refinement for smooth movements of parts.

I printed all the 3D printed parts, But the problem I encountered was that there a lot of mesh lines on the 3D printed parts which resulted in friction in the operating stage. Hence this resulted in less fluidity and dexterity of fingers and also heating of servo motor and lack of accuracy of angle.

In order to resolve this issue needed to refine each part join by making its surface smooth so that it won't interfere with the smooth moving operation of fingers when fingers are under servomotor influence. Also, autolock strips to each finger for better rigidity in the finger when fingers are in tangential stress influence.

5. Assembly of all components and connection between finger and servo motor.

I did the assembly of all 3D-printed parts. After assembly, I encountered imperfections and very close dimension clashes. so to get rid of that, I needed to clean and sharpen the surfaces of 3D printed parts for perfect fitting and smooth kinematic movements between links and joints of the robotic hand.

One finger got installed to the palm. I connected the string linkage between the servo motor lever and the 3-fold joint finger. Initially, I found that the servo turning angle was greater than the 3-fold joint finger which resulted in the breakage of the finger and link of the finger.

I solved this proposed change by increasing the elastic linkage which will take care of over-torque operations from the servo which won't result in breakage.

6. Conclusion

6.1 Key Findings:

Fabricated successfully robotic bionic hand similar to the human anatomy which ensures a high degree of fluidity and dexterity of motion in real-time operation with servo actions and capable of executing similar gestures in real-time with minimal latency like the real human hand.

Assembled and installed servo motor into 3D printed servo mounting as shown in image: 30. Calibrated the servo angular rotation from 0-180 degrees precisely with 3-fold finger mechanisms to operate the finger from 0-180 degrees between open and close states similar to human hand.

Successfully established the UART channel communication which can able to communicate between Python to C++ code to receive commands from Python code to drive servo motors at specific angles with precise degrees of motion.

Developed an OpenCV algorithm that will fetch the angle parameters from the detected Human hand by overlying an exoskeleton plot over the original hand. This method has been executed with an object detection algorithm to detect the hand in the camera footage and plot exoskeleton structural landmarks to implement mathematical calculations with the help of those landmarks. A total of 21 landmarks have been plotted and we have used 5 sets of landmark groups each group containing 3 landmarks to calculate the angle of five fingers via vector approach.

The key finding and novelty of this project mainly has the continuous movement of the servo motor which allows the controlling of fingers to rotate from 0-180 by feeding appropriate signals to the servo for precise rotation of the servo shaft to achieve fluidity movement of the robotic hand fingers and dexterity similar to human anatomy. Which makes this finding superior to other literature work done till now.

6.2 Summary of Objectives and Significance

The goal and objective of this prototype research is to bridge the gap between human-computer communication interfaces via a gesture input to operate the robots with precise and accurate movements of robotic actuators. This research surely would be an asset for the robotic industry to operate the robots in fields where man can not enter the territory such as Industrial Production lines, Military applications, Space Robots, the Medical Industry, etc.

This prototype research allows us to operate robots with a precise and fluid motion similar to human anatomy to control the real-time environment with active actions of human gestures. With some more advancement and future work proposed for the extension of this prototype research would unlock tremendous opportunities for robotic application in Human-Computer-Interaction (HMI).

7 Recommendations for Future Work:

1. The 3D printing step is complete. The use of 3D printing to create robotic arm components remains a significant advance. Future work should prioritize completing this step to begin the integration and demonstration of the robotic arm.

2. Enhancement in image processing algorithms. Further improvements and optimizations of computer vision algorithms are recommended to improve accuracy and practical application by using stereo camera mechanisms.

3. rigorous inspection and testing are required to ensure smooth integration and functionality operation components. Testing in different environments and scenarios helps identify and eliminate potential issues and limitations.

4. Similar to the bionic hand can be applicable to fabricate and automate biped robotic leg and other parts of robot which will be an intelligent and superior key to establish a natural connection with humanoid robots and getting tasks done on the unmanned fields.

8 References

[1] S. Ahmed, V. Popov, A. Topalov, N. Shakev, "Hand Gesture based Concept of Human - Mobile Robot Interaction with Leap," in IFAC PapersOnLine, vol. 52-25, pp. 321–326, 2019. DOI: 10.1016/j.ifacol.2019.12.543

[2] X. Li, "Human-robot interaction based on gesture and movement recognition," Signal Processing: Image Communication, vol. 81, 115686, 2020. DOI: 10.1016/j.image.2019.115686

[3] J. Paterson, A. Aldabbagh, "Gesture-Controlled Robotic Arm Utilizing OpenCV," in European Journal of Science and Technology, June 2021. DOI: 10.1109/HORA52670.2021.9461389

[4] S. Devine, K. Rafferty, S. Ferguson, "Real-time robotic arm control using hand gestures with multiple end effectors," in 2016 UKACC 11th International Conference on Control (CONTROL), Belfast, UK, August 31 - September 2, 2016. DOI: 10.1109/CONTROL.2016.7737545

[5] J. Hossain Gourob, S. Raxit and A. Hasan, "A Robotic Hand: Controlled With Vision Based Hand Gesture Recognition System," 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 2021, pp. 1-4, doi: 10.1109/ACMI53878.2021.9528192.

[6] P. Atre, S. Bhagat, N. Pooniwala and P. Shah, "Efficient and Feasible Gesture Controlled Robotic Arm," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 1-6, doi: 10.1109/ICCONS.2018.8662943

[7] R. M. Gurav and P. K. Kadbe, "Real-time finger tracking and contour detection for gesture recognition using OpenCV," 2015 International Conference on Industrial Instrumentation and Control (ICIC), Pune, India, 2015, pp. 974-977, doi: 10.1109/IIC.2015.7150886.

[8] Pedro Neto, J. Norberto Pires, A. Paulo Moreira, “Accelerometer-Based Control of an Industrial Robotic Arm”

[9] Dr. R. V. Dharaskar, S. A. Chhabria, Sandeep Ganorkar, “Robotic Arm Control Using Gesture and Voice”, In International Journal of Computer, Information Technology & Bioinformatics (IJCITB), Vol. 1, Issue 1, pp. 41-46

[10] S. Waldherr, R. Romero and S. Thrun, 2000, “A gesture based interface for human-robot interaction”, In Autonomous Robots in Springer, vol. 9, Issue 2, pp. 151- 173

[11] K. Brahmani, K. S. Roy, Mahaboob Ali, April 2013, “Arm 7 Based Robotic Arm Control by Electronic Gesture Recognition Unit Using Mems”, International Journal of Engineering Trends and Technology, Vol. 4 Issue 4 Available at: http://www.ijettjournal.org/volume-4/issue4/IJETT-V4I4P347.pdf

[12] S. Perrin, A. Cassinelli and M. Ishikawa, May 2004, “Gesture Recognition Using Laser-Based Tracing System”, In Automated Face and Gesture Recognition. Proceeding, Sixth IEEE Conference, pp. 541-546

[13] Y. Song, S. Shin, S. Kim, D. Lee, and K. H. Lee, “Speed estimation from a tri-axial accelerometer using neural networks, ” in 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS 2007, pp. 3224-3227, 2007

[14] J. Yang, W. Bang, E. Choi, S. Cho, J. Oh, J. Cho, S. Kim, E. Ki and D. Kim, 2006, “A 3D Hand drawn Gesture Input Device using Fuzzy ARTMAP-based Recognizer”, In Journal of Systemic, Cybernetics and Informatics, Vol. 4 Issue 3, pp. 1-7.

[15] K. Murakami and H. Taguchi, 1991, “Gesture Recognition using Recurrent Neural Networks”, In Proceedings of ACM CHI’91 Conference on Human Factors in Computing Systems, New Orleans, USA, pp. 237-242.

[16] P. Atre, S. Bhagat, N. Pooniwala and P. Shah, "Efficient and Feasible Gesture Controlled Robotic Arm," *2018 Second International Conference on Intelligent Computing and*

Control Systems (ICICCS), Madurai, India, 2018, pp. 1-6, doi: 10.1109/ICCONS.2018.8662943.

[17] J. Paterson and A. Aldabbagh, "Gesture-Controlled Robotic Arm Utilizing OpenCV," *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey, 2021, pp. 1-6, doi: 10.1109/HORA52670.2021.9461389

[18] S. Bularka, R. Szabo, M. Otesteanu and M. Babaita, "Robotic Arm Control with Hand Movement Gestures," *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, Athens, Greece, 2018, pp. 1-5, doi: 10.1109/TSP.2018.8441341.

[19] R. V. Dharaskar. et.al., Robotic Arm Control Using Gesture and Voice. Int. Journal of Computer, Inf. Technology & Bioinformatics (IJCITB). ISSN:2278-7593, Vol. 1, Issue-1. 2012.

[20] D. ZheKang. et.al. A fuzzy-based parametric fault diagnosis approach for multiple circuits. 11th Int. Conference on Control. DOI: 10.1109/CONTROL. 2016.7737525. Publisher: IEEE. Belfast, UK. 2016.

[21] Abidhusain, et.al. "Flex Sensor Based Robotic Arm Controller Using Micro Controller" Journal of Software Eng. and Applications, Vol.5 No.5, 2012

[22] A. Rashmi, et.al. "Robotic Hand Controlling Using Flex Sensors and Arduino Uno". Int. Research Journal of Eng. and Technology (IRJET) e-ISSN: 2395-0056. Vol. 06 Issue: 04, Apr 2019.

[23] U. D. Meshram and R. Harkare, "FPGA Based Five-Axis Robot Arm Controller," Int. Journal of Electronics Engineering, Vol. 2, No. 1, 2010, pp. 209-211.

[24] M. Bhusa, et.al. "Controlling Robot by Fingers Using Flex Sensors" Int. Journal of Computer Trends and Tech.67.8 (2019): 13-17.

[25] C. P. Shinde," Design of Myoelectric Arm", Int. Journal of Advanced Science, Engineering, and Technology. ISSN 2319-5924, V. 1, I. 1, pp 21-25. 2012

[26] R. Singh, et.al, "Design and Development of a Data Glove for the Assistance of the Physically Challenged". Int. Journal of Elect. and Communication Eng. and Tech. (IJECET).4(4), pp. 36–41. 2013.

[27] L. Shrimanth Sudheer, Immanuel J., P. Bhaskar, and Parvathi C. S. ARM 7 Microcontroller Based Fuzzy Logic Controller for Liquid Level Control System. International Journal of Electronics and Communication Eng.& Tech. (IJECET).4(2), pp. 217–224. 2013

[28] A. Turnip, et.al, An application of modified filter algorithm fetal electrocardiogram signals with various subjects, International Journal of Artificial Intelligence, vol. 18, no. 1, pp. 207-217, 2020.

[29] R. Deepan, Santhana Vikrama Rajavarman, and K. Narasimhan., "Hand Gesture-Based Control of Robotic Hand using Raspberry Pi Processor". Asian Journal of Scientific Research. Vol. 8, (I). 3. pp. 392-402. 2015.

[30] ESP32 WROOM 32E — SunFounder ESP32 Starter Kit documentation, [no date]. . Online. Available from: https://docs.sunfounder.com/projects/esp32-starter-kit/en/latest/components/component_esp32_extension.html [Accessed 14 May 2024].

[31] SHAKHADRI, Syed Abdul Gaffar, 2021. Building a Hand Tracking System using OpenCV. Analytics Vidhya. Online. 8 July 2021. Available from: https://www.analyticsvidhya.com/blog/2021/07/building-a-hand-tracking-system-using-opencv/.

9. Annexure

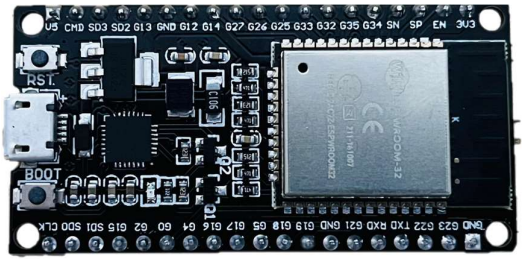

Image 41: ESP-WROOM-32 Module [source: own]

Image 42: Servo Drive (MG996R)-(5 Nos) [source: own]

Image 43: Connecting Wires [source: own]

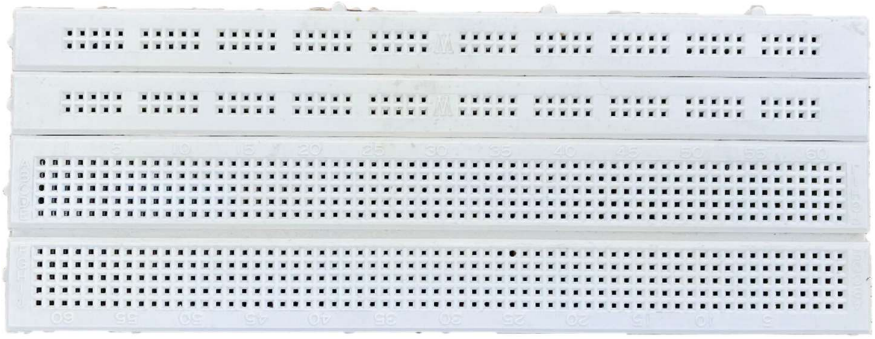

Image 44: Breadboard [source: own]

Image 45: Soldering Machine [source: own]