Smart Gesture Control using Arduino and Python (Computer Vision)

A Mini project Project Report submitted
In the Partial of Fulfillment of the Requirements for
ECE IOT – III Semester PTI Lab

By

1. BT22ECI002 Jjateen Gundesha
2. BT22ECI004 Vedant Shrivastava
3. BT22ECI005 Ayush R. Ambatkar

Under the Guidance of

Dr. Snehal Shinde

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

Indian Institute of Information Technology, Nagpur

CERTIFICATE

This is to certify that the Project Based Learning-I work titled "**Smart Gesture Control using Arduino and Python (Computer Vision)**" that is being submitted by **Jjateen Gundesha (BT22ECI002), Vedant Shrivastava (BT22ECI004), Ayush R. Ambatkar (BT22ECI005)** is in partial fulfillment of the requirements for **B. Tech. (ECE-IoT) – III Semester**, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Snehal Bankatrao Shinde

Guide

The Report is satisfactory / unsatisfactory

Approved by

Mini-Project coordinator

Dr. Snehal B. Shinde

ABSTRACT

This project addresses the challenge of creating a gesture-based control system that enables **Human-Computer Interaction** in a tangible and intuitive manner. With the growing interest in gesture recognition technology, we seek to provide a practical solution that combines computer vision and microcontroller capabilities.

The project offers a unique approach by employing MediaPipe Hands for real-time hand landmark detection. It provides two distinct modes of interaction: one for controlling a servo motor's angle through left-hand gestures and another for controlling the illumination of LEDs through right-hand gestures. This system's potential applications span a wide range of domains, from home automation to interactive exhibitions, gaming, and assistive technologies.

By implementing this project, we contribute to the ongoing exploration of gesture-based control systems and their practical applications. We aim to empower developers and enthusiasts with an example of how machine vision and hardware integration can facilitate user-friendly and immersive experiences. This project serves as a foundation for further developments in the field, promoting innovative solutions to real-world problems and fostering user engagement through natural interactions.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

In the realm of human-computer interaction, the quest for more intuitive and natural ways to control and communicate with technology has been a driving force. Gesture recognition technology has emerged as a captivating solution that promises to bridge the gap between humans and machines by enabling direct and physical interaction. It holds immense potential across various domains, including robotics, gaming, augmented reality, and assistive technologies, to name a few. This project embarks on an exploration of gesture-based control systems, offering a comprehensive solution that brings together computer vision and microcontroller technology.

The primary objective of this project is to create an interactive system that recognizes and responds to hand gestures in real time. To achieve this, we harness the power of MediaPipe Hands, a state-of-the-art library for hand landmark detection. The project comprises two key components: a Flask web application and a standalone Python script. The web application serves as a user interface, offering a visually engaging way to interact with the system, while the Python script handles the underlying logic of gesture recognition and control.

The system enables users to control an Arduino board, which is equipped with both a servo motor and a set of LEDs, through hand gestures. Left-hand gestures manipulate the angle of the servo motor, allowing for precise and dynamic control. Right-hand gestures govern the illumination of LEDs, offering a visual feedback mechanism. This two-pronged approach demonstrates the versatility of gesture-based control and its potential to be employed in diverse applications.

This report delves into the technical details and the underlying mechanisms of the system, providing insights into the implementation and functionality. It outlines the steps taken to integrate MediaPipe Hands, configure the Arduino board, and create an interactive user interface. Moreover, it discusses the potential applications of such a system in the real world, highlighting the implications for home automation, entertainment, and accessibility.

The project represents an exciting fusion of technology and creativity, demonstrating how machine vision can be harnessed to enable natural and tangible interactions with physical devices. It serves as a foundation for further exploration in the realm of gesture-based control systems and encourages innovation in human-computer interaction. This report outlines the methodology, technical aspects, and potential impact of the project, paving the way for a deeper understanding of the possibilities and challenges in this emerging field.

1.1 Objectives

The below mentioned are the objectives of this project:

- Develop a gesture-based control system that leverages MediaPipe Hands for real-time hand landmark detection.
- Implement two modes of interaction: left-hand gestures to control a servo motor's angle and right-hand gestures to manipulate the illumination of LEDs.
- Create a user-friendly web interface using Flask for intuitive interaction with the system.
- Showcase the potential of gesture-based control for applications in home automation, entertainment, gaming, and accessibility.
- Explore the technical intricacies of integrating computer vision and microcontroller technology.
- Promote innovation in human-computer interaction by merging machine vision with tangible, natural interaction methods.

1.2 Literature Survey

AUTHOR	TITLE	YEAR And PUBLISHER	METHODOLOGY	ACCURACY	OBSERVATIONS
Camillo Lugaresi Jiuqiang Tang Hadon Nash Chris McClanahan Esha Uboweja	MediaPipe: A Framework for Perceiving and Processing Reality	2019, Google	Device Integration, Resource Management, Parallel Processing and Pipelining, Time-Series Data Synchronization, Performance Measurement, Use of MediaPipe Framework, Reproducibility	MediaPipe's accuracy varies based on specific tasks and models, aiming for high accuracy in computer vision but subject to change with updates; for	MediaPipe simplifies perceptual input processing by providing efficient resource management, time-series data synchronization, and cross-platform development, enabling

			across Devices and Platforms.	precise figures, refer to official documentation and research papers.	developers to focus on algorithm and model development.
B Sundar, T Bagyammal	American Sign Language Recognition for Alphabets Using MediaPipe and LSTM	2022, Procedia Computer Science	Vision-Based ASL Alphabet Recognition, MediaPipe for Hand Landmarks, Custom Dataset Creation, LSTM for Gesture Recognition, 99% Accuracy Achieved, Application for Converting Hand Gestures to Text	The system achieved a 99% accuracy in recognizing ASL alphabet signs using vision-based techniques and LSTM.	Significance of Gesture Recognition, Applications in Deaf-Mute Communication, HCI, Robotics, Home Automation, and Medical Fields, Vision-Based ASL Alphabet Recognition for Static and Dynamic Gestures, Utilization of Google's MediaPipe Framework, Creation of a Custom Dataset for Experimental Study, Usage of LSTM for Hand Gesture Recognition, High Accuracy Rate of 99% for Recognizing 26 ASL Alphabet Signs, Potential for Converting

					Hand Gestures into Text.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the System Architecture and its Details
- Chapter 3 describes implementation of the project.
- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report and gives idea of future scope.
- Chapter 6 consists of code of our project.
- Chapter 7 gives references.

CHAPTER 2

Smart Gesture Control Using Arduino and Python (Computer Vision)

2.1 System Details

This section describes the software and hardware details of the system:

2.1.1 Software Details

Python, Legacy Arduino IDE, OpenCV, MediaPipe and Pyfirmata are used.

i) Python

Python is a versatile, high-level programming language known for its simplicity and readability. It is widely used in various domains, including web development, data analysis, artificial intelligence, and scientific computing. Python's strengths lie in its clean and easy-to-understand syntax, extensive standard library, and dynamic typing, making it suitable for both beginners and experienced programmers. With a large open-source community, Python supports rapid development and is a popular choice for a wide range of applications, offering flexibility and ease of use.

BENEFITS OF PYTHON

• Presence of Third-Party Modules

• Extensive Support Libraries

• Open Source and Community Development

• Learning Ease and Support Available

• User-friendly Data Structures

• Productivity and Speed

• Highly Extensible and Easily Readable Language.

ii) Arduino IDE

The Arduino Integrated Development Environment (Arduino IDE) is an open-source software platform designed for programming Arduino microcontroller boards. It provides a user-friendly code editor, library support, and a streamlined upload process. With cross-platform compatibility and a strong community, it simplifies Arduino development for a wide range of users, from beginners to experienced developers.

iii) Pyfirmata

PyFirmata is a Python library that facilitates communication with Arduino boards using the Firmata protocol. It allows Python programs to control and interact with Arduino's hardware components, such as sensors, motors, and LEDs, in real-time. PyFirmata simplifies the process of sending and receiving data between Python and Arduino, making it an invaluable tool for projects involving Arduino integration. This library helps bridge the gap between software and hardware, enabling the development of interactive and responsive systems.

iv) OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source software library designed for computer vision and machine learning tasks. It provides essential functions for image and video processing, making it a valuable tool for applications like image analysis, object detection, and facial recognition. With broad platform support and an active community, OpenCV is widely used in fields such as robotics, artificial intelligence, and computer vision.

v) MediaPipe

MediaPipe is a popular open-source framework developed by Google for building real-time, cross-platform AI solutions. It provides pre-trained machine learning models and tools for a range of tasks, including hand tracking, face detection, pose estimation, and more. MediaPipe simplifies complex computer vision tasks, making it a valuable resource for developers and researchers in the field of machine learning and computer vision.

2.4.2 Hardware Details

Arduino UNO, LED and Servo motor are used.

i) Arduino UNO

The Arduino Uno is a widely-used microcontroller board based on the ATmega328P. It's known for its simplicity, 14 digital I/O pins, 6 analog inputs, and USB programming interface. Operating at 5V, it's versatile for a range of projects and supported by the user-friendly Arduino IDE.

ii) Servo motor

A servo motor is a rotary actuator that provides precise control over angular position. It uses feedback mechanisms to maintain the desired position and is commonly employed in applications requiring accurate and controlled movement, such as robotics and industrial automation.

CHAPTER 3
IMPLEMENTATION

3.1 Importing required libraries

1. Flask:

- Flask is a micro web framework for building web applications.
- Used for creating a user-friendly web interface for the gesture-based control system.
- Allows for serving video streams and user interactions.
- Simplifies the development of the user interface and web server components.

2. OpenCV (Open Source Computer Vision Library):

- OpenCV is a powerful computer vision library for image and video analysis.
- Used for capturing, processing, and displaying video frames from a camera.
- Enables image manipulation, feature detection, and drawing on frames.
- Essential for real-time video stream handling.

3. MediaPipe:

- MediaPipe is a framework developed by Google for building real-time, cross-platform AI solutions.
- Utilized for hand landmark detection in real-time video frames.
- Provides pre-trained models and tools for object recognition and tracking.
- Simplifies the implementation of complex computer vision tasks.

4. math (Python's Built-in Math Library):

- Used for mathematical calculations within the code.
- Particularly employed for calculating angles and performing trigonometric operations.
- Enables precise angle calculations for servo motor control.
- Essential for mathematical operations required in the gesture recognition process.

5. pyFirmata:

- pyFirmata is a Python library for communicating with Arduino boards using the Firmata protocol.
- Employed for controlling the Arduino board and its connected hardware components.

- Facilitates communication between the Python script and the Arduino.
- Allows for sending signals to the servo motor and LEDs.

6. Python's Built-in Libraries:

- Python's built-in libraries such as os are used for general-purpose functionality.
- os may be used for tasks like managing file paths and configurations.
- Provides platform-independent operations and access to the operating system.
- Enhances the overall functionality and versatility of the code by enabling system-level interactions and configurations.

3.2 Code

Flask Application

```python
from flask import Flask, render_template, Response
import cv2
import mediapipe as mp
import math
import pyfirmata

app = Flask(__name__)

# Initialize MediaPipe Hands
mp_hands = mp.solutions.hands
hands = mp_hands.Hands()
mp_drawing = mp.solutions.drawing_utils

# Initialize pyFirmata to communicate with the Arduino
board = pyfirmata.Arduino('COM7')  # Replace with your Arduino port

# Define servo and LED pins
servo_pin = 9  # Change this to the appropriate pin on your Arduino
led_pins = [13, 12, 11, 10]  # Define the pins for your LEDs

# Initialize the servo
servo = board.get_pin(f'd:{servo_pin}:s')

# Initialize the LEDs
leds = [board.get_pin(f'd:{pin}:o') for pin in led_pins]

# Set the screen resolution to 1920x1080
cap = cv2.VideoCapture(0)
cap.set(3, 1920)  # Width
cap.set(4, 1080)  # Height

# Define finger tip IDs and LED count
```

```
tipIds = [8, 12, 16, 20]  # Finger tip landmarks
led_count = 0

# Set the hand distinction threshold
threshold = 0.2  # Adjust as needed

@app.route('/')
def index():
    return render_template('index.html')

def generate_frames():
    while True:
        ret, frame = cap.read()
        if not ret:
            continue

        # Increase the size of the output window
        frame = cv2.resize(frame, (1080, 720))  # Adjust the size as needed

        # Convert the BGR image to RGB
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Process the frame with MediaPipe Hands
        results = hands.process(rgb_frame)

        if results.multi_hand_landmarks:
            for landmarks in results.multi_hand_landmarks:
                if landmarks.landmark[8].x < landmarks.landmark[20].x -
threshold:
                    for i, led in enumerate(leds):
                        led.write(0)

                    middle_finger_landmarks = [landmarks.landmark[i] for i in
range(12, 17)]
                    base_x, base_y = middle_finger_landmarks[0].x,
middle_finger_landmarks[0].y
                    tip_x, tip_y = middle_finger_landmarks[4].x,
middle_finger_landmarks[4].y
                    angle = math.degrees(math.atan2(tip_y - base_y, tip_x -
base_x))
                    servo_angle = max(0, min(180, int(90 + angle)))
                    servo.write(servo_angle)

                    mp_drawing.draw_landmarks(frame, landmarks,
mp_hands.HAND_CONNECTIONS)
                    angle_text = f'Middle Finger Angle: {angle:.2f}'
                    text_size, _ = cv2.getTextSize(angle_text,
cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
                    text_x = frame.shape[1] - text_size[0] - 10
                    cv2.putText(frame, angle_text, (text_x, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                else:
                    fingers = []
                    for id in tipIds:
                        if landmarks.landmark[id - 1].y < landmarks.landmark[id -
2].y:
                            fingers.append(1)
                        else:
                            fingers.append(0)
                    led_count = fingers.count(1)

                    cv2.putText(frame, f'LED Count: {led_count}', (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

                    for i, led in enumerate(leds):
                        if i < led_count:
                            led.write(1)
                        else:
                            led.write(0)

                    mp_drawing.draw_landmarks(frame, landmarks,
mp_hands.HAND_CONNECTIONS)

        ret, buffer = cv2.imencode('.jpg', frame)
        if not ret:
            continue

        frame = buffer.tobytes()

        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video')
def video():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')

if __name__ == '__main__':
    app.run(debug=True)
```

HTML and CSS for Flask Application

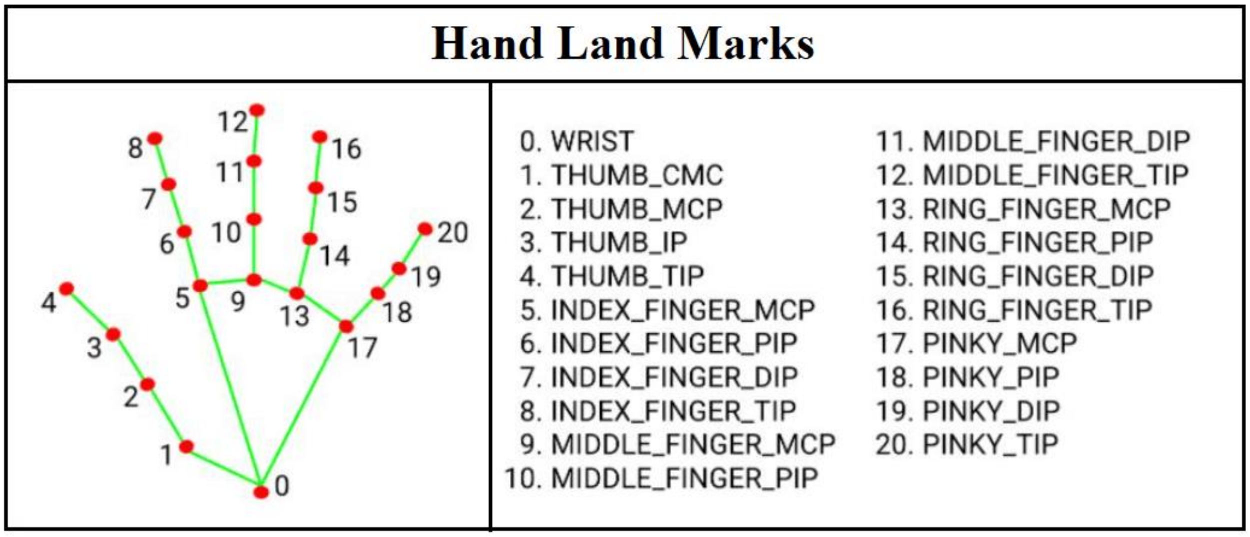
```
<!DOCTYPE html>
<html>
<head>
    <title>Hand Control</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body>
    <header>
        <h1>LED and Motor Control using Hand Movement</h1>
        <h2>[Project for Programming Techniques for IoT]</h2>
    </header>
    <section class="content">
        <div class="video-container">
            <img id="video_feed" src="{{ url_for('video') }}" alt="Video Feed">
        </div>
    </section>
</body>
</html>
```

```
@import
url('https://fonts.googleapis.com/css2?family=Fira+Sans&family=Open+Sans:wght@300
&display=swap');

body {
    font-family: 'Open Sans', Arial, sans-serif;
     background: linear-gradient(45deg, #FCCF31, #F55555);
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    text-align: center;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    transition: background-color 0.3s ease;
}

header {
    margin-top: 16px;
    font-family: 'Fira Sans', sans-serif;
    font-size: 12px;
    font-weight: 600;
    text-transform: uppercase;
    color: transparent;
    background: white;
    -webkit-background-clip: text;
    background-clip: text;
    transition: color 0.3s, font-size 0.3s;


    &:hover {

        font-size: 18px;
    }
}

.video-container {
    margin-top: 20px;
    border: 1px solid #ccc;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
    border-radius: 16px;
    overflow: hidden;
    transition: box-shadow 0.3s ease; /* Smooth shadow transition */
}

#video_feed {
    max-width: 100%;
    height: auto;
    transition: transform 0.3s ease; /* Smooth image scale transition */
}

/* Hover effect for the video container */
.video-container:hover {
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.4);
    transform: scale(1.08); /* Slight zoom-in effect on hover */
    transition: box-shadow 0.3s ease, transform 0.3s ease; /* Smooth hover
transitions */
}

header:hover {
    color: #ff0000;
}


.video-container:hover {
    background-color: #fff;
    transition: background-color 0.3s ease;
}
```

3.3 Working

1. Hand Gesture Detection with MediaPipe Hands:

- The core functionality of the system relies on MediaPipe Hands, which accurately detects 21 hand landmarks. It processes each frame from the camera and identifies key points on the user's hand in real-time.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

2. Dual Gesture Modes:

- The system implements two gesture modes:

  - Left-Hand Gestures: The system calculates the angle between specific hand landmarks to control a servo motor. As the user's hand gestures change, the servo motor's angle adjusts accordingly.

  - Right-Hand Gestures: In this mode, the system counts the number of extended fingers to control the state of a set of LEDs. Users can turn LEDs on and off by changing the number of extended fingers.

3. Flask Web Interface:

- A Flask web application serves as the user interface, offering a visual platform for interaction. Users can access the system through a web page that displays the video stream from the camera.

4. Real-Time Video Stream Processing:

- The Python script continuously captures video frames from the camera using OpenCV. These frames are then converted to the RGB format to be processed by MediaPipe Hands.

5. Gesture Recognition Logic:

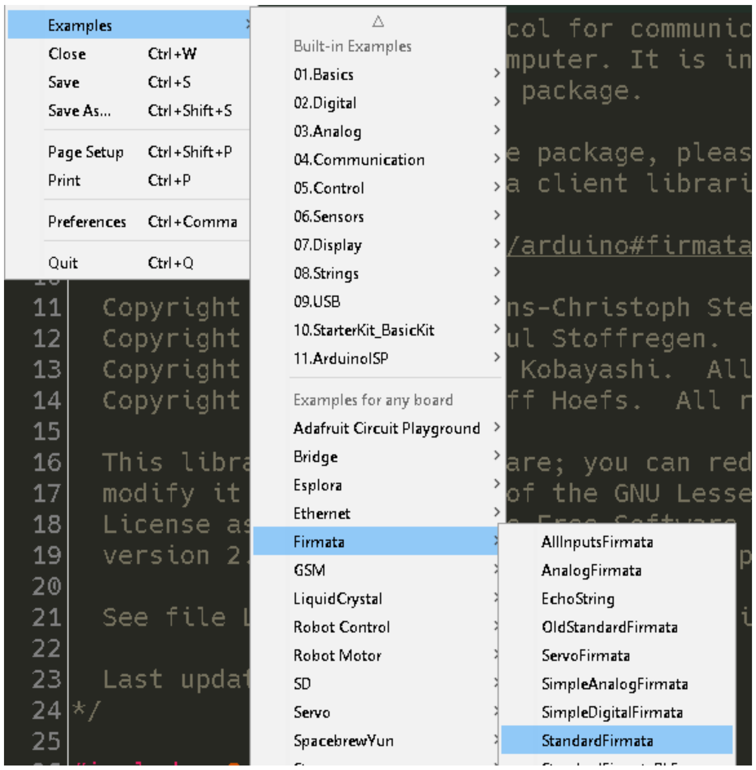
- The script includes specific logic for recognizing the defined gestures. For left-hand gestures, the system calculates the angle between two key points on the hand and maps this angle to the servo motor's angle. For right-hand gestures, it counts the number of extended fingers to determine LED control.

6. Visual Feedback Mechanisms:

- To enhance the user experience, the system provides visual feedback. For example, when users manipulate the servo motor, the web interface displays the real-time angle measurement, allowing users to see the immediate effect of their hand gestures. Similarly, the LED count is displayed when controlling the LEDs.

7. Integration with Arduino:

- The system communicates with an Arduino board using the pyFirmata library. This enables control over the servo motor and LEDs connected to the Arduino. The Arduino board translates the commands received from the Python script into physical actions.



8. Extensibility and Adaptability:

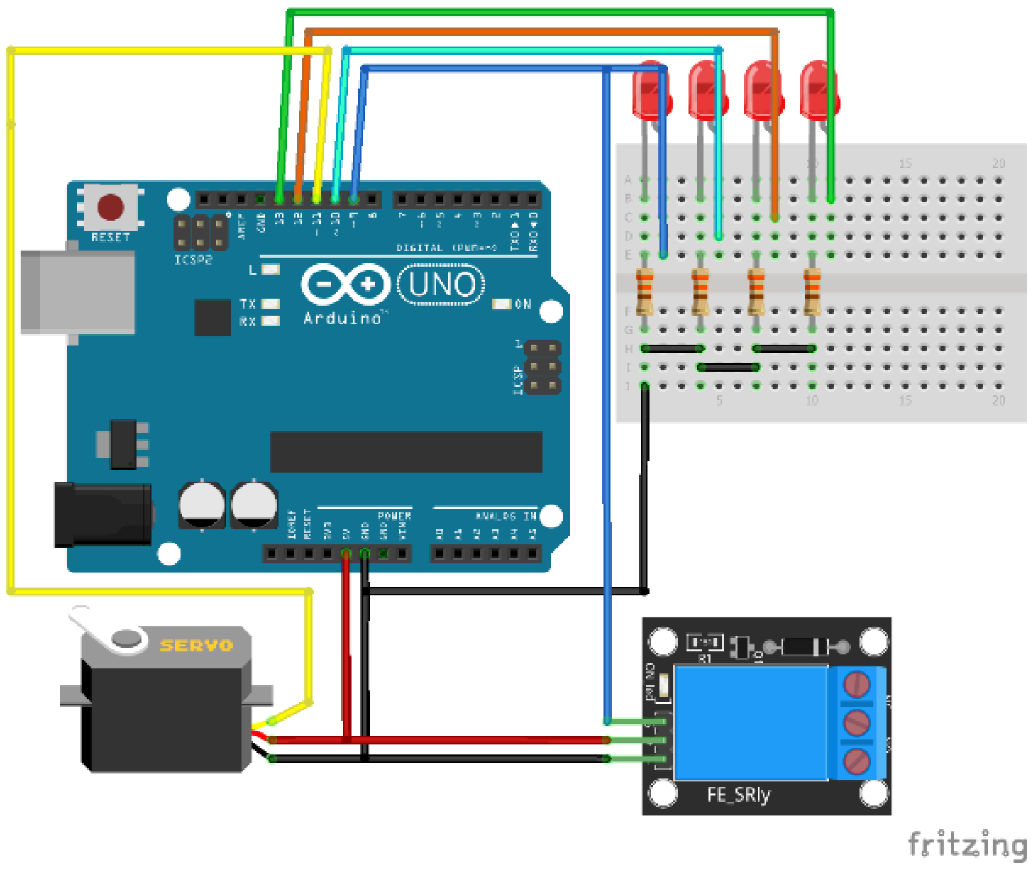
- The code is designed to be adaptable and easily configurable. Users can customize the Arduino board's pins and adjust gesture recognition logic to suit their specific hardware and application requirements.

9. Real-World Applications:

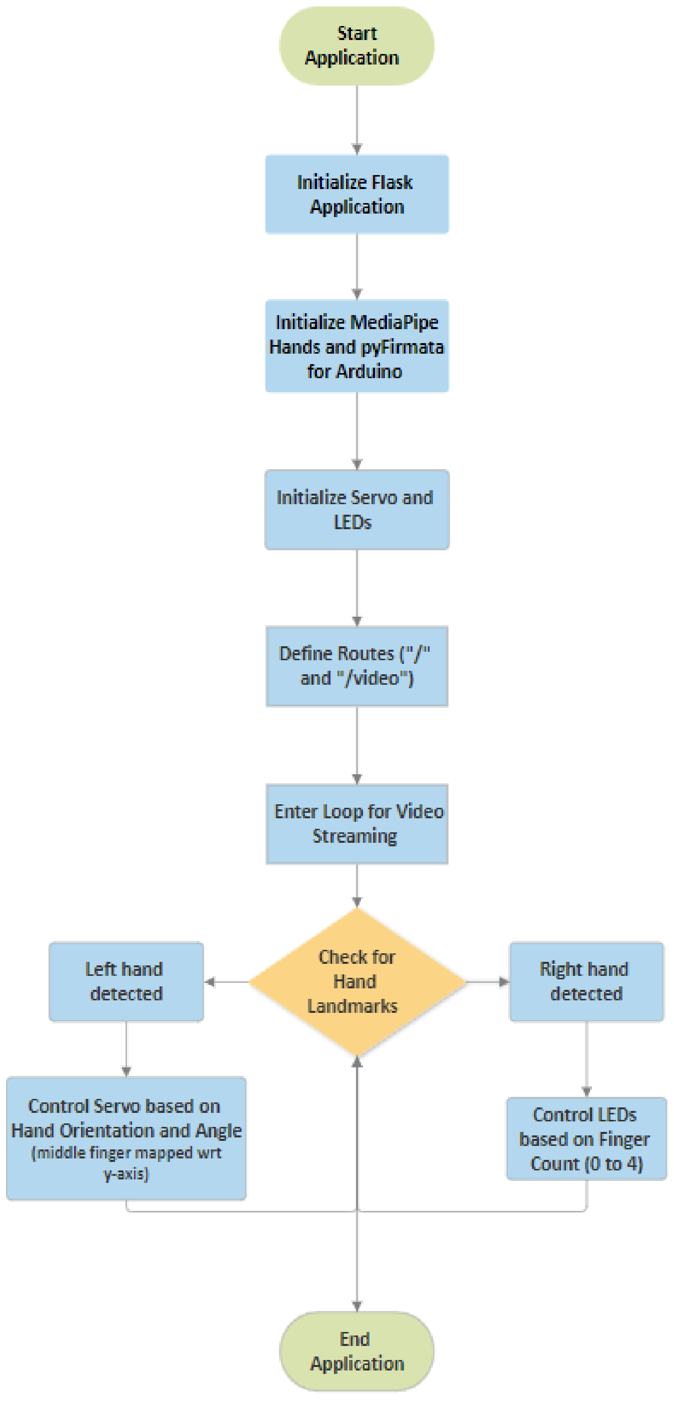
- The project demonstrates the potential for practical applications in various domains, including home automation, entertainment, gaming, and accessibility. It highlights the versatility of gesture-based control systems and their relevance in diverse contexts.

By effectively combining computer vision, hardware control, and web-based interfaces, the system provides an intuitive and engaging platform for users to interact with technology. This division of the working part into subtopics illustrates the various aspects and components that collectively make the project functional and versatile.

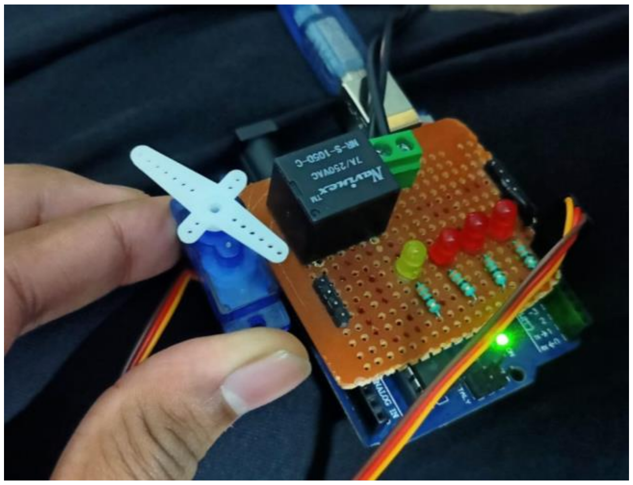
3.4 Circuit Diagram

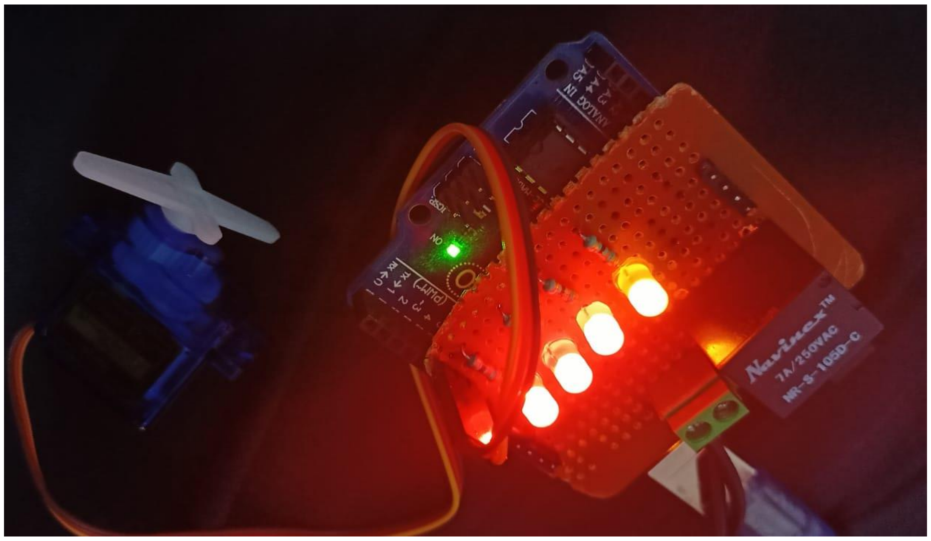


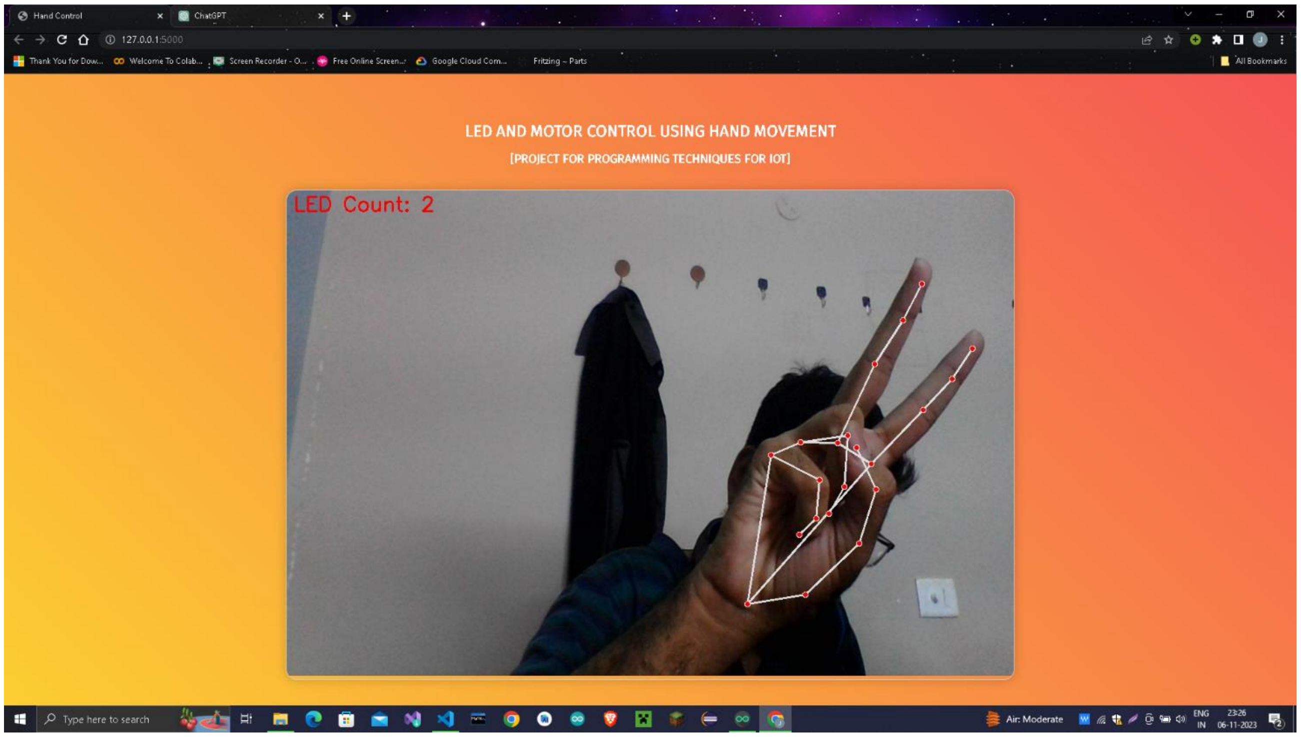
3.5 Flowchart


Start Application
Initialize Flask Application
Initialize MediaPipe Hands and pyFirmata for Arduino
Initialize Servo and LEDs
Define Routes ("/" and "/video")
Enter Loop for Video Streaming
Check for Hand Landmarks
Left hand detected
Right hand detected
Control Servo based on Hand Orientation and Angle (middle finger mapped wrt y-axis)
Control LEDs based on Finger Count (0 to 4)
End Application

CHAPTER 4
RESULTS AND DISCUSSIONS



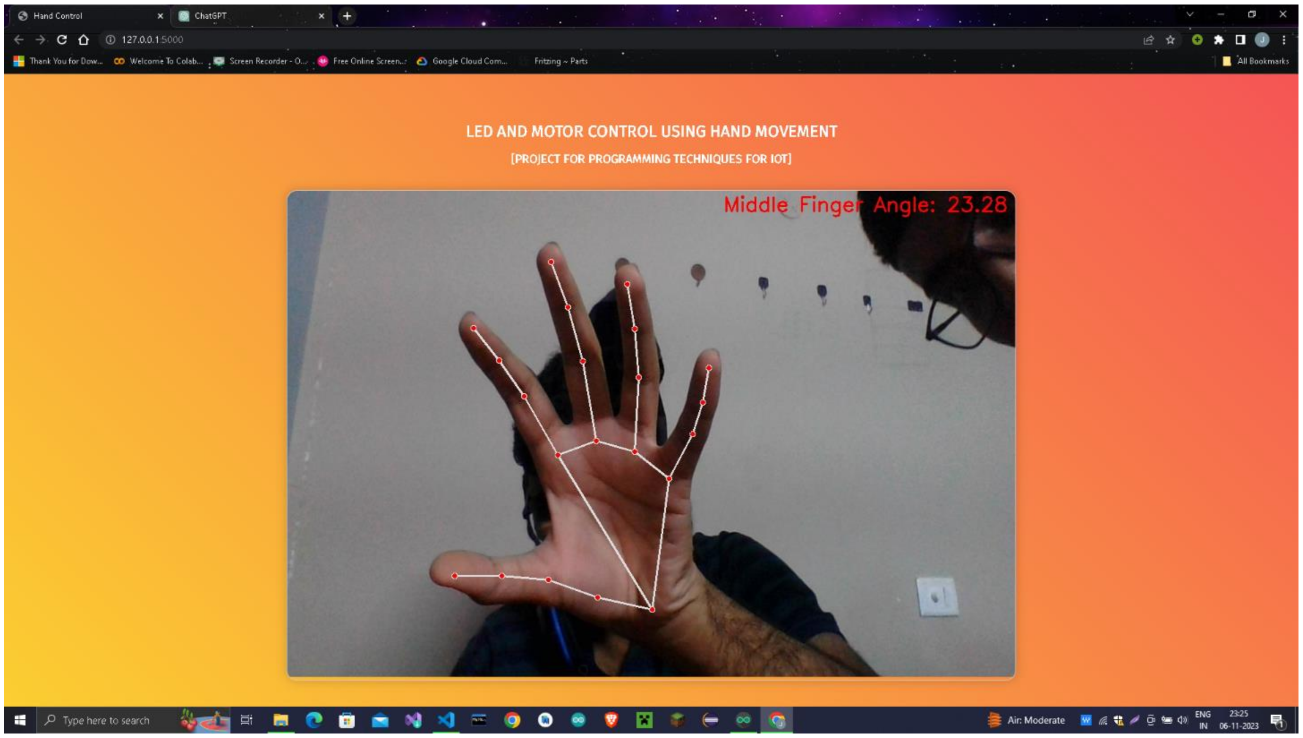




LED AND MOTOR CONTROL USING HAND MOVEMENT

[PROJECT FOR PROGRAMMING TECHNIQUES FOR IOT]

LED Count: 2



LED AND MOTOR CONTROL USING HAND MOVEMENT

[PROJECT FOR PROGRAMMING TECHNIQUES FOR IOT]

Middle Finger Angle: 23.28

\

CHAPTER 5

CONCLUSION AND FUTURE WORK

In conclusion, the project's successful implementation of a gesture-based control system using MediaPipe, Arduino, and Python libraries underscores the potential of merging computer vision and microcontroller technology. The ability to interact with hardware through natural hand gestures opens up exciting possibilities in various fields, including home automation, entertainment, gaming, and accessibility.

This project highlights the adaptability and extensibility of the code, allowing users to customize and configure their own hardware components, providing a foundation for innovation in human-computer interaction.

As technology continues to evolve, gesture-based control systems exemplify the convergence of human-friendly interfaces and powerful computing capabilities. This project serves as a testament to the exciting possibilities that lie at the intersection of software and hardware, offering both educational insights and practical applications.

CHAPTER 6

APPENDIX

https://github.com/Jjateen/PTI-mediapipe

REFERENCES

- https://www.iieta.org/journals/isi/paper/10.18280/isi.280311

1. https://youtu.be/EgjwKM3KzGU?si=S6gMfJm8SOADD7bR (AI Hand Pose Estimation with mediapipe)
2. https://research.google/pubs/pub48292/ (mediapipe)
3. https://www.irjmets.com/uploadedfiles/paper/issue_6_june_2022/27011/final/fin_irjmets1656344520.pdf