INDEX

Sr. No.	Title	Page Number
1.	Introduction	4
2.	Proposed Methodology	5
3.	Flow Chart and Algorithm	7
4.	Code and Output	9
5.	Conclusion	14

1. Introduction

Our mini project is all about controlling motor speed using hand gestures. The main objective of this experiment is to demonstrate effective communication between Python code, Arduino, and a servo motor. We've utilized a range of powerful libraries for this task, including pyserial (for serial communication), opencv-python (for image processing), cvzone (to bridge Python with Arduino), mediapipe (for hand tracking), and numpy (for numerical operations).

This experiment employs a webcam to capture hand gestures, detects hand landmarks using Mediapipe, and calculates the distance between specific points (the thumb and index finger) to map and send motor control signals to the Arduino. The result is a responsive system that adjusts motor speed based on hand gestures.

2. Proposed Methodology

a. Hardware Setup:

- Connect the Arduino to the computer and ensure it is properly interfaced with the servo motor.
- Attach the servo motor's signal wire to the 9th digital pin on the Arduino.
- Connect the power wires: positive to the 5V pin and negative to GND on the Arduino.

b. Software Environment:

- Use Python to write and execute the primary code for gesture detection and data transmission.
- Ensure pyserial, opencv-python, cvzone, mediapipe, and numpy are installed for seamless operation.

c. Hand Gesture Detection:

- Capture the live video feed using OpenCV.
- Convert video frames to RGB and use Mediapipe to detect and track hand landmarks.
- Extract coordinates for the thumb and index finger tips.

d. Distance Calculation and Mapping:

- Calculate the Euclidean distance between the detected points of the thumb and index finger.
- Map this distance to motor control values corresponding to different speeds and directions:
  - ✓ **Inward and Fast:** Higher speed and clockwise rotation.
  - ✓ **Outward and Fast:** Higher speed and anticlockwise rotation.
  - ✓ **Inward and Slow:** Lower speed and clockwise rotation.
  - ✓ **Outward and Slow:** Lower speed and anticlockwise rotation.

e. Data Transmission to Arduino:

- Send the calculated control value from the Python script to the Arduino using the cvzone.SerialModule or pyserial library.
- Ensure the format is appropriate for the Arduino to interpret the data.

f. Motor Control Logic on Arduino:

- Receive the speed and direction data from the Python script.
- Translate the received values to control the servo motor's position and speed using PWM signals on pin 9.

g. Feedback Loop and Visualization:

- Visualize the detected hand landmarks and the line between the thumb and index finger in real-time using Open CV.
- Display debug information such as the mapped speed and direction values to aid in troubleshooting and fine-tuning.

h. Testing and Optimization:

- Perform initial tests to verify gesture detection and servo response.
- Refine the mapping function for smoother motor control and response accuracy.

3. Flow Chart and Algorithm

a. Algorithm:

1) **Start**

2) **Initialize System:**
   - Set up the webcam.
   - Initialize the servo motor on pin 9.

3) **Run the Arduino IDE code.**

4) **Start Video Capture:**
   - Begin capturing video from the webcam.

5) **Process Each Frame:**
   - For each frame captured from the webcam:
     a. Convert the frame to grayscale.
     b. Apply gesture detection algorithm (e.g., using OpenCV for contour detection).

6) **Identify Hand Gestures:**
   - If a hand gesture is detected:

a. **Check Gesture Type**:
   i. **Inward and Fast**: Move servo to position for inward fast.
   ii. **Outward and Fast**: Move servo to position for outward fast.
   iii. **Inward and Slow**: Move servo to position for inward slow.
   iv. **Outward and Slow**: Move servo to position for outward slow.

b. If no gesture is detected, keep the servo at the last position.

7) **Control Servo Motor:**
   - Based on the identified gesture, send the corresponding command to the servo motor.
   - Adjust the position of the servo motor based on the gesture type.

8) **Repeat Process:**
   - Repeat steps 3-5 until the program is terminated.

9) Cleanup:

- Release the webcam and any resources used when exiting.

10) End

4. Code and Output

a. Arduino IDE Code:

```
#include <Servo.h>

Servo myServo;
int servoPin = 9;  // Connect your servo to pin 9
int angle = 0;

void setup() {
    Serial.begin(9600);
    myServo.attach(servoPin);
    myServo.write(90);  // Start at 90 degrees
}

void loop() {
    if (Serial.available() > 0) {
        angle = Serial.parseInt();  // Read the angle sent from Python
        angle = constrain(angle, 0, 180);  // Constrain the angle to avoid overdriving
        myServo.write(angle);
        Serial.println(angle);  // For debugging
    }
}
```

b. Pycharm Python Code:

```
import cv2
import mediapipe as mp
from cvzone.SerialModule import SerialObject

# Variables for the motor control
arduino = SerialObject("COM6")  # Replace with your correct COM port
x1 = y1 = x2 = y2 = 0

# Initialize camera and Mediapipe hands detector
webcam = cv2.VideoCapture(0)
my_hands = mp.solutions.hands.Hands()
drawing_utils = mp.solutions.drawing_utils

while True:
    _, image = webcam.read()
    frame_height, frame_width, _ = image.shape

    # Convert image to RGB for hand detection
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    output = my_hands.process(rgb_image)
    hands = output.multi_hand_landmarks

    # Check if any hands are detected
    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)
            landmarks = hand.landmark

            # Iterate through hand landmarks to get thumb and index
finger positions
            for id, landmark in enumerate(landmarks):
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)

                if id == 8:  # Index finger tip
                    cv2.circle(img=image, center=(x, y), radius=8,
color=(0, 255, 255), thickness=3)
                    x1 = x
                    y1 = y
                if id == 4:  # Thumb tip
                    cv2.circle(img=image, center=(x, y), radius=8,
color=(0, 0, 255), thickness=3)
                    x2 = x
                    y2 = y

                    # Ensure both points are detected before calculating
the distance
                    if x1 and y1 and x2 and y2:
                        # Calculate the distance between the thumb and
index finger
                        dist = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
                        cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0),
5)
```

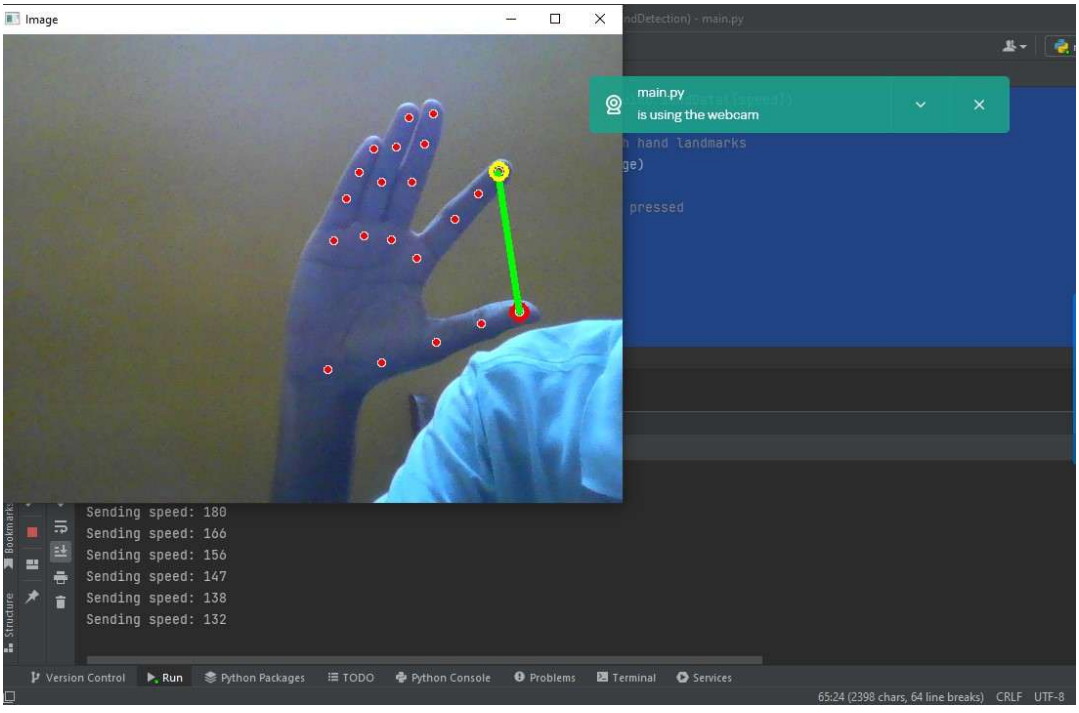
```
                        # Map distance to speed or angle value
                        # For example, map dist (0-200) to angle (0-180)
                        speed = int(min(max((dist / 200) * 180, 0),
180))
                        print(f"Sending speed: {speed}")  # Print the
speed for debugging
                        arduino.sendData([speed])

    # Display the image with hand landmarks
    cv2.imshow("Image", image)

    # Exit loop if 'ESC' is pressed
    key = cv2.waitKey(10)
    if key == 27:
        break

# Release resources
webcam.release()
cv2.destroyAllWindows()
```

OUTPUT:



Serial Monitor in pycharm:

```
C:\Users\Admin\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\Admin\PycharmProjects\pythonProject3\main.py
Attempt 1 of 5 to connect...
Serial Device Connected
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
W0000 00:00:1730637333.271257   15256
inference_feedback_manager.cc:114] Feedback manager requires a model
with a single signature inference. Disabling support for feedback tensors.
W0000 00:00:1730637333.319877   15256
inference_feedback_manager.cc:114] Feedback manager requires a model
with a single signature inference. Disabling support for feedback tensors.
C:\Users\Admin\PycharmProjects\pythonProject3\venv\Lib\site-
packages\google\protobuf\symbol_database.py:55: UserWarning:
SymbolDatabase.GetPrototype() is deprecated. Please use
message_factory.GetMessageClass() instead.
SymbolDatabase.GetPrototype() will be removed soon.
  warnings.warn('SymbolDatabase.GetPrototype() is deprecated. Please '
Sending speed: 67
Sending speed: 69
Sending speed: 69
Sending speed: 74
Sending speed: 105
Sending speed: 127
Sending speed: 127
Sending speed: 139
Sending speed: 141
Sending speed: 137
Sending speed: 143
Sending speed: 145
Sending speed: 141
Sending speed: 148
Sending speed: 151
Sending speed: 153
Sending speed: 144
```