Advanced Mechatronics Project Report

Smart Home Automation System Using Gesture Control

Sotomi Oluwadamilola - ots2014

El Tannir, Omar - oe2104

Ravi, Yukteshwar - yr2364

Ragupathy, Sai Nishit – sr6931

TABLE OF CONTENT

1.0.ACKNOWLEGEMENT

We would like to express our deep gratitude to Professor Vikram Kapila for his invaluable guidance throughout this project. His insight and encouragement helped us refine our ideas and navigate technical challenges. We also thank our peers, who generously shared their feedback and collaborated with us. Their constructive input provided a foundation for meaningful improvement. The successful completion of this project would not have been possible without everyone's combined effort.

2.0. ABSTRACT

This project develops a sophisticated gesture-controlled smart home system, leveraging advanced computer vision to enhance the interaction between users and their household environments. Utilizing a Raspberry Pi equipped with an Arducam camera, the system interprets specific hand gestures—where each finger raised corresponds to controlling a particular household device such as lights, fans, doors, and water pumps. OpenCV and MediaPipe technologies are employed to detect these gestures in real-time and translate them into actionable commands. When a finger is raised, it activates or deactivates the assigned device, making the system exceptionally intuitive and user-friendly. This setup not only facilitates a futuristic living experience but also significantly improves accessibility, especially for individuals with mobility impairments. The project showcases the potential of integrating gesture recognition with traditional home automation technologies to create a more adaptive and responsive smart home environment.

3.0. INTRODUCTION

The evolution of home automation has progressively aimed to integrate more seamless and intuitive control mechanisms that enhance user interaction and accessibility. Traditional systems, while functional, often rely on physical interactions or remote controls that can be less accessible for some individuals. Addressing this gap, our project introduces a gesture-based control system that simplifies the process by assigning specific household devices to individual finger gestures. Using a Raspberry Pi and an Arducam camera, the system captures and processes hand movements using the robust capabilities of OpenCV and MediaPipe for gesture recognition. Each finger's movement—whether an index finger, middle finger, ring finger, or little finger—is mapped to control a specific device, allowing for the activation or deactivation with just a simple gesture. This approach not only reduces the physical barriers associated with traditional control systems but also enriches the user experience by providing a quick, natural, and engaging way to interact with the smart home environment. Through detailed research and innovative engineering, this project endeavors to push the boundaries of what smart home systems can achieve, making them more accessible, efficient, and enjoyable for all users.

4.0. PARTS AND EXPLANATION

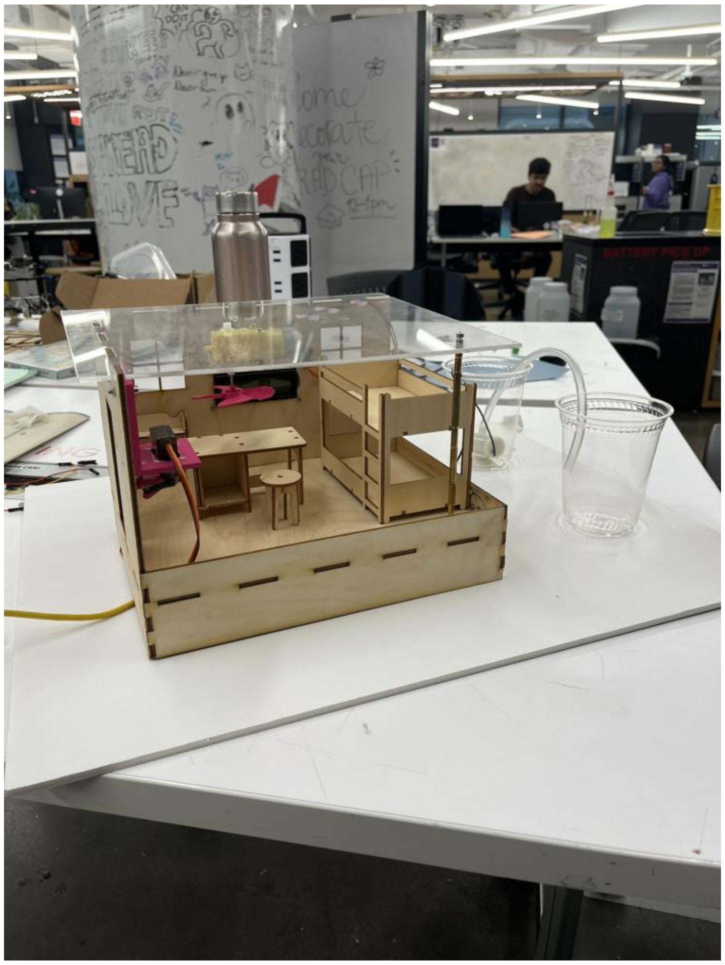


Fig 4.1. Smart Home Automation System

4.1 Raspberry Pi :

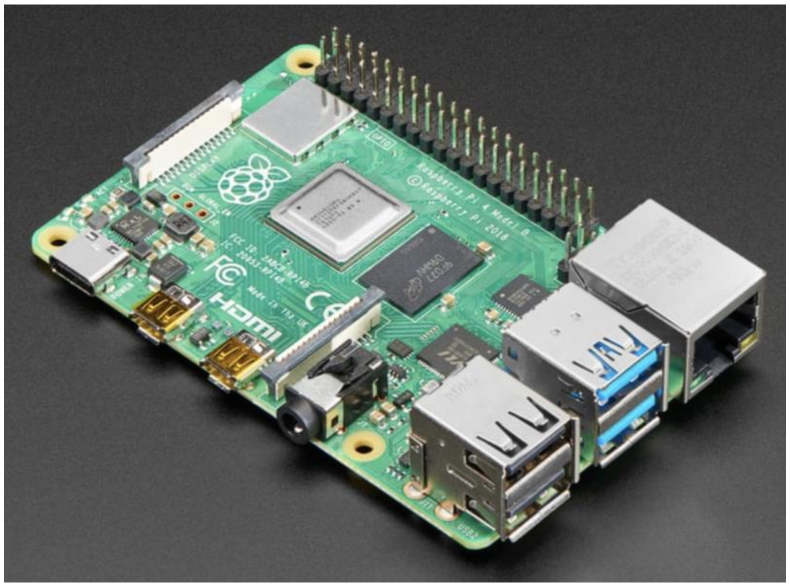


Fig.4.2 Raspberry Pi

The Raspberry Pi 4 serves as the brain of this gesture-based smart home control system. With a powerful quad-core Cortex-A72 CPU operating at 1.5 GHz, it provides the computational horsepower needed for real-time gesture recognition. It comes with 2GB to 8GB of RAM and supports HDMI output and wireless communication via Wi-Fi and Bluetooth. This microcomputer runs OpenCV and MediaPipe, two libraries used for hand gesture detection. The Raspberry Pi captures the live video feed from the camera, processes it to identify specific hand gestures, and maps these gestures to control commands. It then transmits the commands via a serial connection or a wireless protocol like Bluetooth or Wi-Fi to the Arduino, which triggers the appropriate devices.

4.2. Arducam :

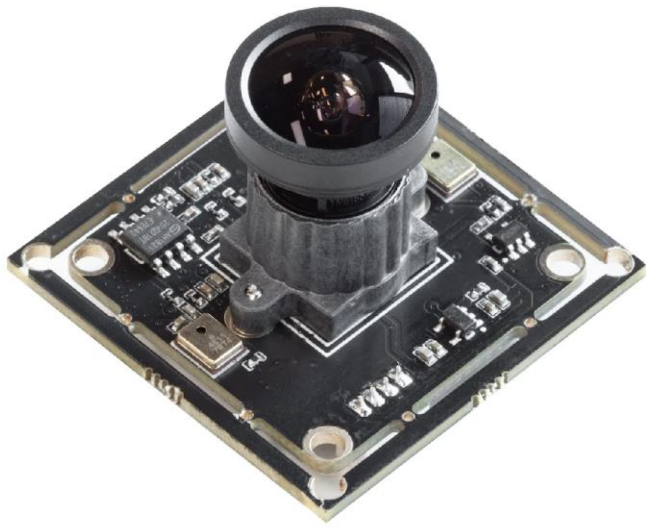


Fig 4.3. **Arducam**

A critical part of the gesture recognition system, the Arducam camera connects directly to the Raspberry Pi and captures a high-resolution video feed for accurate gesture recognition. This camera module is designed specifically for embedded applications and provides the Raspberry Pi with the raw video data needed to analyze hand movements effectively. It ensures that every gesture is captured clearly and quickly enough for real-time processing. This high-quality data stream is essential for the accurate detection and classification of different hand gestures.

4.3. Arduino UNO:

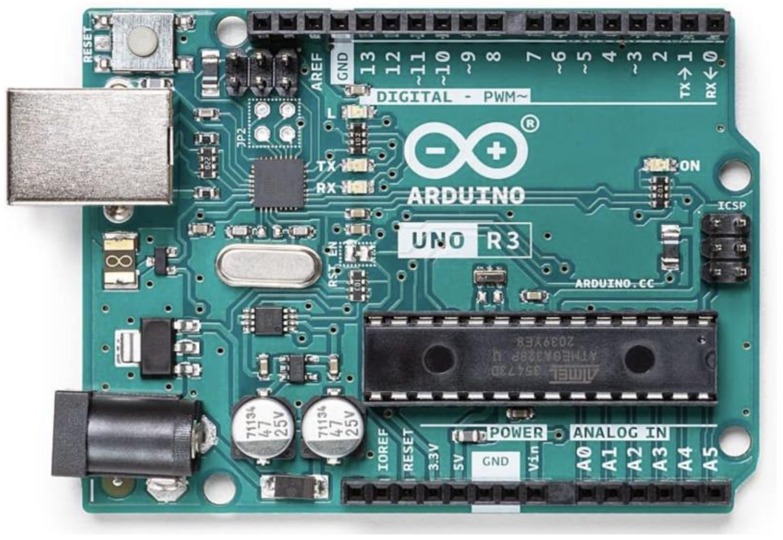


Fig 4.4. Arduino UNO

The Arduino Uno microcontroller acts as the system's executor, receiving control signals from the Raspberry Pi and operating the relevant household devices through connected relay modules. This microcontroller, built around the ATmega328P chip, offers 14 digital I/O pins and 6 analog inputs. It provides a USB interface for programming and communication. In this setup, the Arduino receives gesture commands from the Raspberry Pi via USB or wireless communication and activates or deactivates specific relays to control various home appliances like lights, fans, and locks.

4.4 DC Motor

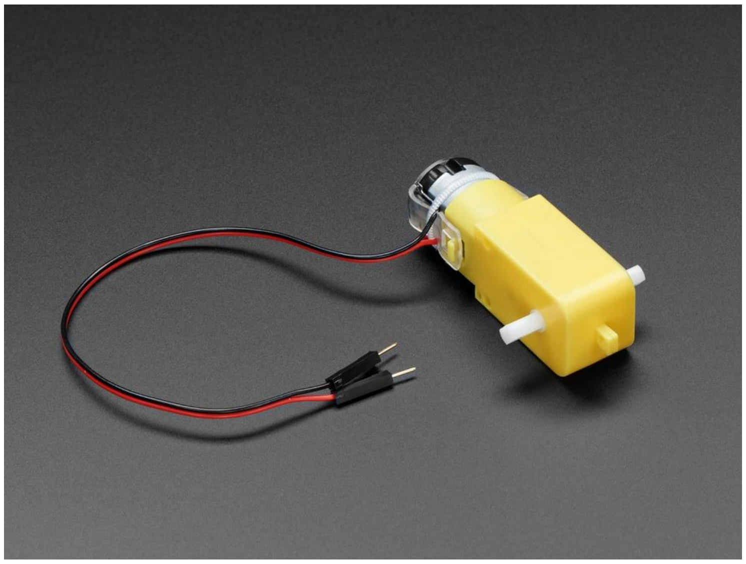


Fig 4.5 DC Motors

DC motors translate direct electrical energy into mechanical motion and are instrumental across a plethora of applications, from miniature models to substantial industrial equipment. Their operation is based on electromagnetic principles, with a design that offers high torque at low speeds, essential for robotics and automated machinery.

4.5 LED:

Fig.4.6. LED

Light-emitting diodes (LEDs) are semiconductor devices that emit light when an electric current passes through them. They are widely used in smart home projects for status indication, decorative lighting, and as part of a home’s main lighting system. LEDs are energy-efficient and offer bright, clear illumination with a long operational lifespan. They come in various colors and can be integrated into the system to indicate the status of devices or provide ambient lighting. Their fast switching ability makes them suitable for pulse signals or visual alerts in the smart home setup.

4.6 L298N H-Bridge Motor Driver:

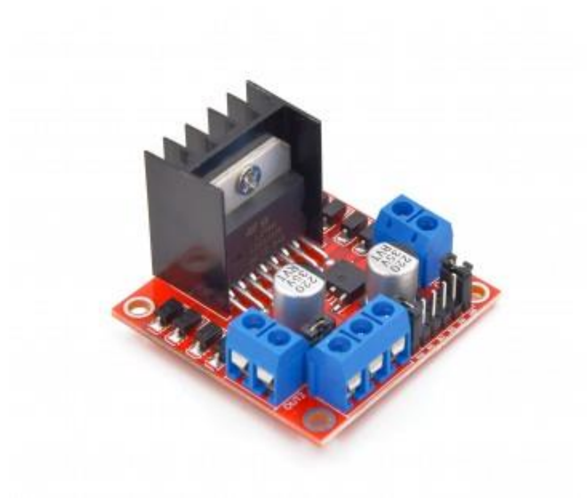


Fig 4.7. L298N **H-Bridge Motor Driver**

The L298N Dual H-Bridge Motor Driver module facilitates the independent operation of two DC motors, supporting both forward and backward movements. Its H-Bridge configuration is key for reversing motor supply voltage polarity, crucial for powering diverse automated and robotic applications.

4.7. Li-ion Battery:

Fig 4.8. Li-ion Battery

Characterized by their high energy density and longevity, Lithium-ion batteries are the project's power source, facilitating the movement of lithium ions between electrodes through an electrolyte. Their widespread use in portable electronics, electric vehicles, and energy storage solutions speaks to their efficiency and robustness.

4.8. Parallax LCD:

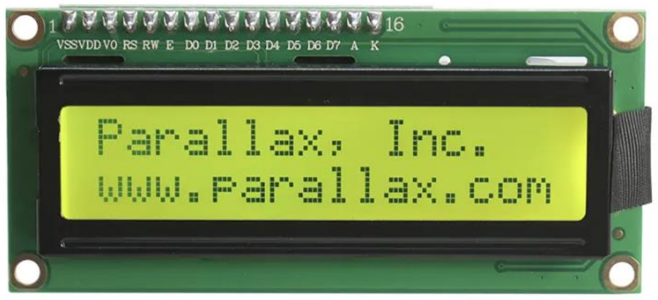


Fig 4.9. Parallax LCD

Parallax LCDs are liquid crystal displays designed for easy interface with microcontrollers. They provide a visual representation of the system status and device information. In smart home automation, these displays can be programmed to show the status of various household devices—such as lights, fans, or pumps—allowing users to quickly ascertain which appliances are active. They can also display alerts, system errors, or relevant information. These LCDs typically have a backlight for better readability and can be programmed to display custom characters or messages.

4.9. Servo Motor:

Fig.4.10. Servo Motor

A servo motor is a rotary actuator that enables precise control of angular position. It is composed of a motor, a feedback sensor, and control circuitry. Servo motors are used in various home automation applications like controlling window blinds, robotic arms, and automated door locks. They are preferred for their ability to accurately position an object to a specific angle or location. In this project, servo motors can be employed to control devices requiring a range of motion with a high degree of precision, like door hinges or camera gimbals.

4.10. **Water Pump:**

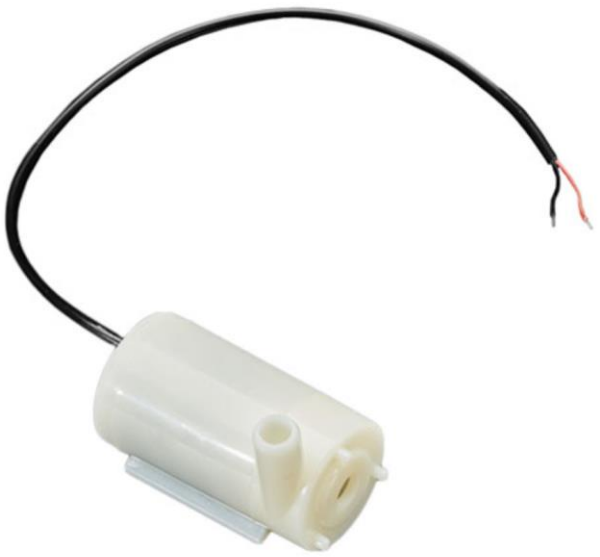
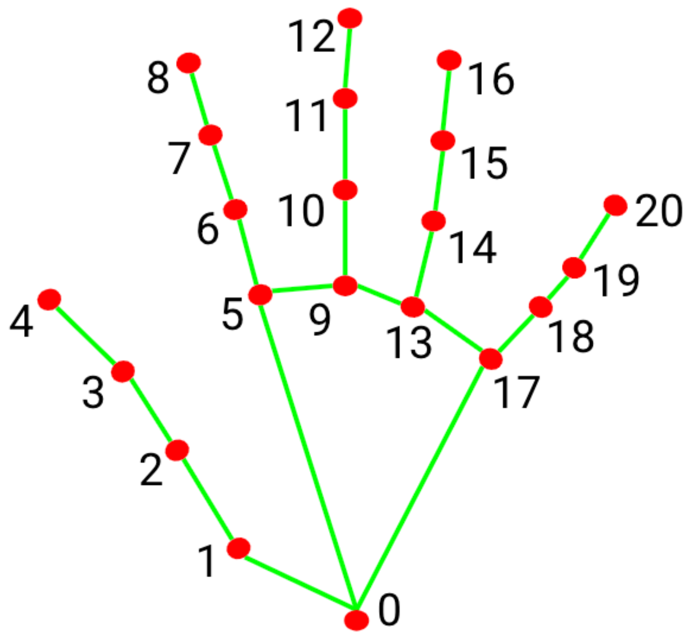


Fig.4.11.Water Pump

A water pump is a mechanical device used to move liquids, such as water. In smart home automation, water pumps find application in gardening systems, aquariums, and automated water supply systems. They can be controlled via relays connected to an Arduino to automate tasks like garden watering or draining water. Depending on the type of water pump used, the system can control varying flow rates and pressures to accommodate different applications. Integration with the smart home system ensures that the water pump operates based on a programmed schedule or in response to specific gestures.

5.0 OPEN CV AND MEDIAPIPE



5.1. Open CV:

OpenCV is a comprehensive open-source library, enabling real-time image processing. In this project, it plays a pivotal role in managing the live video stream from the Arducam camera, capturing frames, and enhancing them to identify shapes and gestures. OpenCV's ability to distinguish hand movements and patterns helps in interpreting individual finger gestures by pinpointing specific landmarks. This ensures that the detection of finger movements is consistent and accurate, a necessity for smooth gesture control, each frame is processed to detect whether a particular finger is raised, like an index finger for activating an LED or a little finger for opening a door. OpenCV’s real-time image manipulation allows for immediate feedback and ensures gestures are correctly interpreted even under challenging lighting conditions.

5.2. MediaPipe:

MediaPipe complements OpenCV with its specialized machine learning-based landmark detection. Its pre-trained models identify 21 landmarks per hand, providing data on finger and hand positions. For this project, MediaPipe processed video frames from OpenCV to recognize gestures and map them to specific commands. MediaPipe's architecture allows for efficient tracking, even when hands are partially obscured or moving quickly.

Each detected gesture corresponds to a predefined command, like turning on the water pump with a ring finger gesture. MediaPipe’s accurate hand landmark detection ensures that the intended gesture is accurately captured and translated into commands for the Arduino controller.

5.3. Combining MediaPipe and OpenCV:

By using OpenCV for real-time image capturing and MediaPipe for hand landmark detection, the system efficiently distinguishes gestures. OpenCV preprocesses the video stream, enhancing frames and extracting crucial details. MediaPipe then uses this information to analyze and identify the hand gestures.

For instance, OpenCV processes the Arducam camera feed, and MediaPipe interprets whether the hand movement corresponds to a fan activation command. The interpreted gesture triggers a signal to the Arduino, which activates the connected device. This synergy ensures seamless smart home automation with each finger assigned to specific devices, reducing errors and providing immediate responses.

6.0. CODE FLOW & EXPLANATION

6.1.Arduino Code:

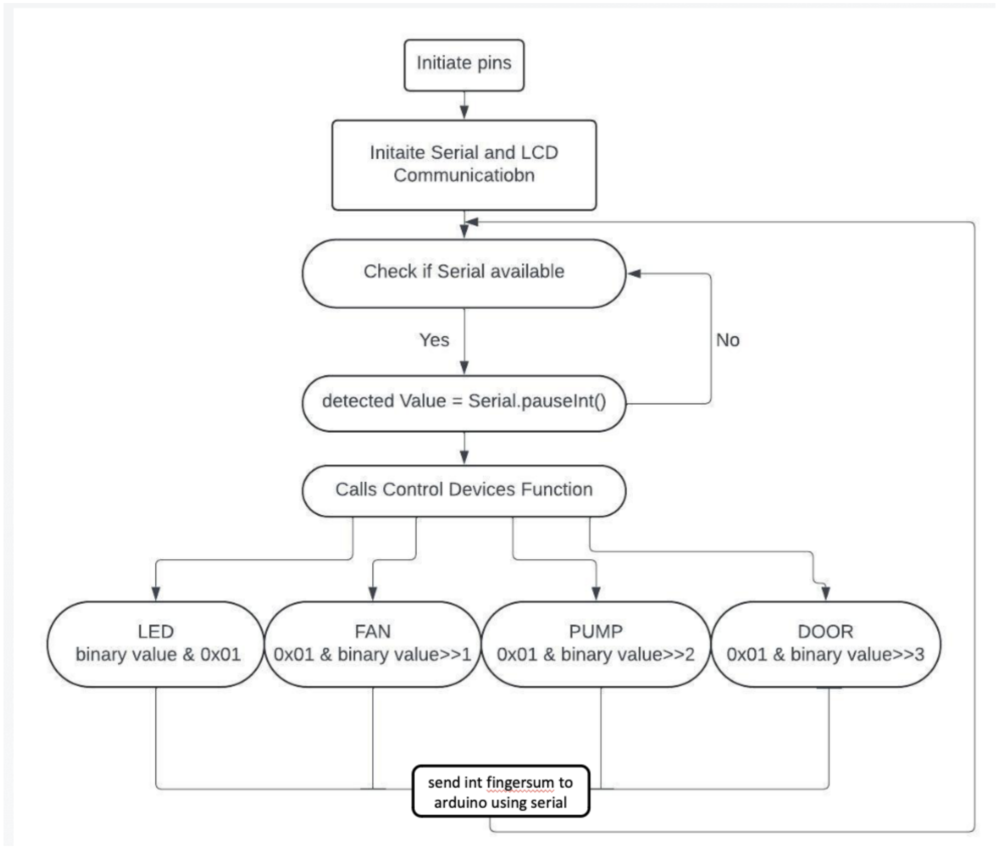

Initiate pins
Initaite Serial and LCD Communicatiobn
Check if Serial available
Yes
No
detected Value = Serial.pauseInt()
Calls Control Devices Function
LED binary value & 0x01
FAN 0x01 & binary value>>1
PUMP 0x01 & binary value>>2
DOOR 0x01 & binary value>>3
send int fingersum to arduino using serial

Fig.6.1.Arduino Code flow

This Arduino code is designed to control various devices based on binary signals received via serial communication from a Raspberry Pi. It starts by including the required libraries (Servo.h for servo motor control and SoftwareSerial.h for LCD communication) and defines pin assignments for each connected component. During setup, serial communication and LCD initialization occur, and all devices are switched off initially, while a welcome message appears on the LCD screen. The controlDevices() function processes binary signals to activate or deactivate the appropriate devices, including an LED, a DC motor, a water pump, and a servo motor. The binary signals are decoded to manage each device's state individually while updating their statuses on the LCD. In the main loop, incoming serial data is parsed into an integer value, which is then passed to the controlDevices() function, allowing each device to respond to the specific binary command it represents. The servo motor moves between two defined positions to control a door mechanism, the LED toggles on or off, the DC motor drives in one direction, and the water pump activates or deactivates. Together, these components form a comprehensive gesture-controlled automation system, where the Raspberry Pi captures gestures and relays them to the Arduino for direct device control.

6.2.Raspberry Pi Code:

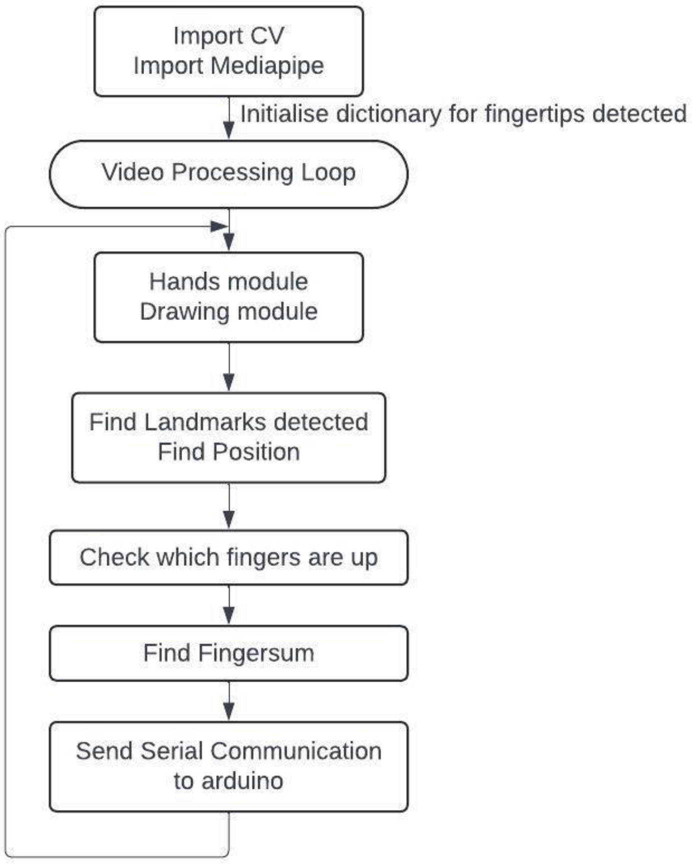


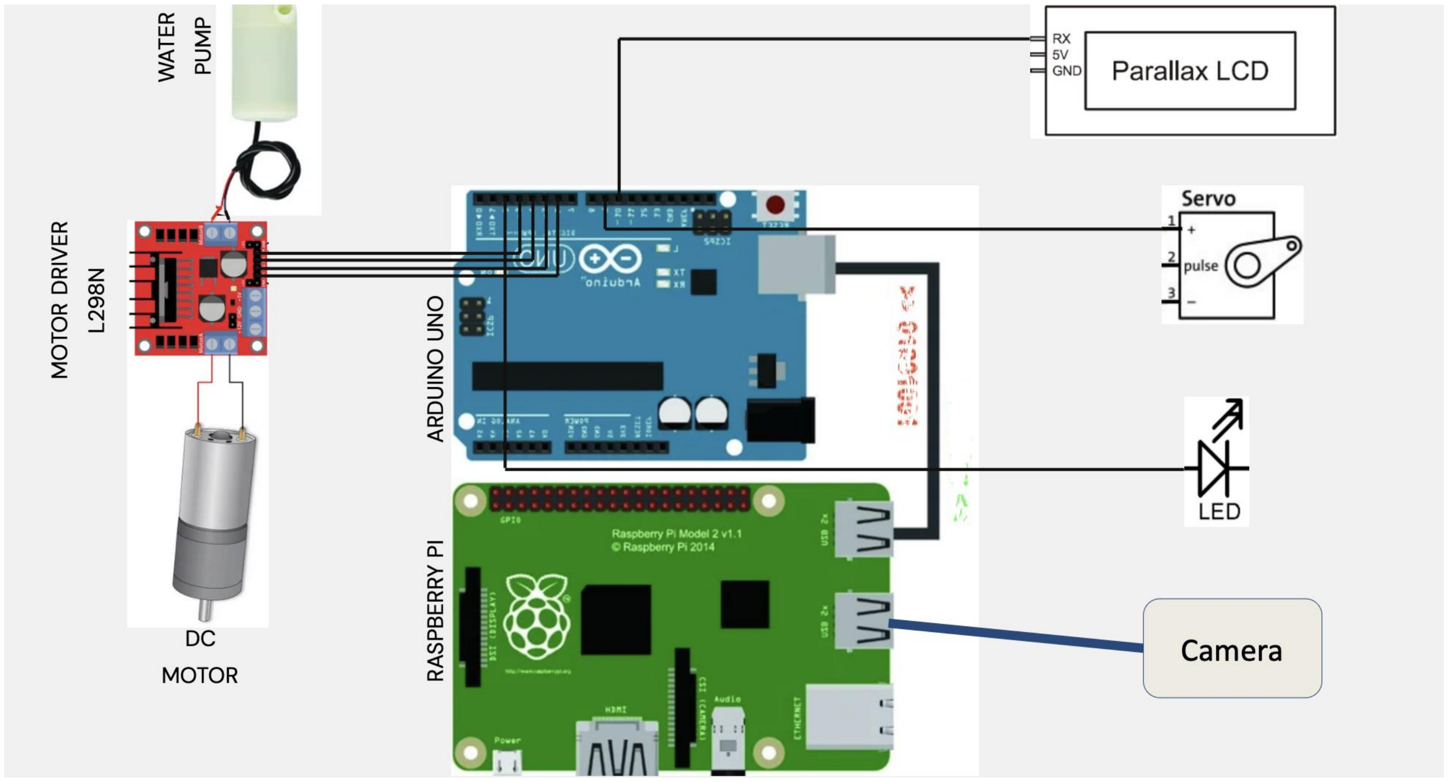
Fig.6.2.Raspberry Pi Code Flow

The first segment begins by importing necessary libraries, including cv2 for OpenCV and serial for Arduino communication. It sets up video capture and initializes serial communication with Arduino. The tip array stores fingertip landmarks for four fingers, each associated with a binary value in finger_binary_dict. The while loop captures video frames, processes hand positions with findpostion() and identifies fingertip landmarks with findnameoflandmark(). If landmarks are detected, a binary sum is calculated based on the raised fingers' positions and converted into an integer. This value is then sent to the Arduino via serial communication. The code also displays the video feed, breaking out of the loop when 's' is pressed.

The module segment sets up the hand landmark detection functionality using the mediapipe library. It initializes drawing and hands modules, while defining dimensions for processing. The findpostion() function identifies and returns the positions of all landmarks detected in a given frame. Meanwhile, the findnameoflandmark() function returns a list of all hand landmarks detected, providing names that can be used for gesture identification. Both functions leverage mediapipe's hands module to analyze the frames captured. These functions support the primary code, helping map detected landmarks for accurate gesture control.

OpenCV and MediaPipe are utilized togther to facilitate gesture-based control of various devices. Initially, the system uses OpenCV to process the video input into RGB format. This processed image is then analyzed by MediaPipe to detect hand landmarks. By comparing the y-coordinates of specific fingertip landmarks (8, 12, 16, 20) against the y-coordinates of landmarks located two indices prior (6, 10, 14, 18), the system assesses whether each finger is raised. The gestures activate different devices accordingly: the index finger controls an LED, the middle finger operates a DC motor, the ring finger manages a water pump, and the little finger actuates a servo motor that operates a door mechanism.

7.0. CIRCUIT SCHEMATIC



The circuit diagram outlines a gesture-controlled smart home automation system that integrates a Raspberry Pi and Arduino Uno to manage various devices. Utilizing a camera connected to the Raspberry Pi, gesture data processed via software like OpenCV and MediaPipe is transformed into commands sent to the Arduino. This Arduino then controls devices such as a DC motor, water pump, servo motor, LED, and an LCD display. The DC motor, connected through an L298N motor driver, and other devices like the water pump and servo motor, receive their operational directives from the Arduino, corresponding to specific gestures. This allows for intuitive gesture-based interactions with the system, enhancing user experience by enabling easy control over household devices.

8.0.CONCLUSION

In conclusion, the Gesture-Based Smart Home Control System project effectively demonstrates the integration of cutting-edge computer vision technology with traditional electronics to create a highly interactive and user-friendly smart home environment. By leveraging a Raspberry Pi and Arduino Uno, the system utilizes gesture recognition to control various devices such as lights, fans, and locks, thereby enhancing convenience and accessibility. The successful implementation of this project not only highlights the practical applications of gesture control in everyday life but also sets the stage for further innovation in smart home technology, promising more sophisticated, efficient, and adaptable home automation solutions in the future.

9.0 APPENDIX

Arduino Code:

```
#include <Arduino.h>
#include <Servo.h>
#include <SoftwareSerial.h>

// Setup servo object
Servo myServo;

// Define pin assignments
const int ledPin = 2;
const int motorDriverPin1 = 3;
const int motorDriverPin2 = 4;
const int motorDriverPin3 = 5;
const int motorDriverPin4 = 6;
const int servoPin = 9;
const int rxPin = 11;  // RX pin for LCD (not used)
const int txPin = 10;  // TX pin for LCD

// Initialize the SoftwareSerial for LCD
SoftwareSerial lcd(txPin, rxPin);

void setup() {
  // Initialize serial communication
  Serial.begin(9600);
  lcd.begin(9600); // Initialize LCD communication
  // Initialize pin modes
  pinMode(ledPin, OUTPUT);
  pinMode(motorDriverPin1, OUTPUT);
  pinMode(motorDriverPin2, OUTPUT);
  pinMode(motorDriverPin3, OUTPUT);
  pinMode(motorDriverPin4, OUTPUT);
  myServo.attach(servoPin);  // Attach servo motor to its control pin

  // Ensure all devices are off initially
  digitalWrite(ledPin, LOW);
  digitalWrite(motorDriverPin1, LOW);
  digitalWrite(motorDriverPin2, LOW);
  digitalWrite(motorDriverPin3, LOW);
  digitalWrite(motorDriverPin4, LOW);
  myServo.write(50);  // Set servo position to 0 degrees

  // Clear LCD and setup initial display
  lcd.write(17);
  lcd.write(12); // Clear screen command
  lcd.print("Welcome Home!");
  delay(3000);
  lcd.write(12);
  delay(10); // Short delay
  lcd.print("L    F   W    D "); // Assuming W is for Water Pump
}

void controlDevices(int binaryValue) {
  // Update LCD with the state of each device
  lcd.write(12); // Clear screen
  delay(10); // Short delay for LCD to process clear command
  lcd.print("L    F   W    D ");
  lcd.print((binaryValue & 0x01) ? "ON  " : "OFF ");
  lcd.print(((binaryValue >> 1) & 0x01) ? "ON   " : "OFF ");
  lcd.print(((binaryValue >> 2) & 0x01) ? "ON " : "OFF ");
  lcd.print(((binaryValue >> 3) & 0x01) ? "ON" : " OFF");

  // Control LED
  digitalWrite(ledPin, binaryValue & 0x01);

  // Control DC Motor
  digitalWrite(motorDriverPin1, (binaryValue >> 1) & 0x01);
  digitalWrite(motorDriverPin2, LOW);

  // Control Water Pump
  digitalWrite(motorDriverPin3, (binaryValue >> 2) & 0x01);
  digitalWrite(motorDriverPin4, LOW);

  // Control Servo Motor
  if ((binaryValue >> 3) & 0x01)
    myServo.write(170);  // Move servo to 180 degrees
  else
    myServo.write(50);    // Move servo back to 0 degrees
}

void loop() {
  if (Serial.available() > 0) {
    int detectedValue = Serial.parseInt();  // Read the incoming integer
    controlDevices(detectedValue);  // Control devices based on the binary input
  }
}
```

Raspberry pi code:

#module code

```
import cv2
import mediapipe as mp

#setup required mediapipe modules used
drawingModule = mp.solutions.drawing_utils
handsModule = mp.solutions.hands

mod=handsModule.Hands()

#adjust resolution
h=480
w=640



def findpostion(frame1):
    list=[]
    results = mod.process(cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB))
    if results.multi_hand_landmarks != None:
        #run thorugh the detected landmarks and draw the position onto the shown
image
       for handLandmarks in results.multi_hand_landmarks:
           drawingModule.draw_landmarks(frame1, handLandmarks,
handsModule.HAND_CONNECTIONS)

           list=[]
           for id, pt in enumerate (handLandmarks.landmark):
                x = int(pt.x * w)
                y = int(pt.y * h)
                list.append([id,x,y])

    return list



def findnameoflandmark(frame1):
     list=[]
     results = mod.process(cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB))
     if results.multi_hand_landmarks != None:
        for handLandmarks in results.multi_hand_landmarks:
            for point in handsModule.HandLandmark:
                 list.append(str(point).replace ("< ","").replace("HandLandmark.",
"").replace("_"," ").replace("[]",""))
     return list
```

#serial code

```
import cv2
import serial
from module import findnameoflandmark, findpostion

# Initialize serial communication with Arduino
ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)

# Set up the video capture
cap = cv2.VideoCapture(0)
tip = [8, 12, 16, 20]
tipname = [8, 12, 16, 20]
finger_binary_dict = {8: 1, 12: 10, 16: 100, 20: 1000}   #dictionary for detected
fingers

while True:
    ret, frame = cap.read()
    frame1 = cv2.resize(frame, (640, 480))
    #fill in list of position and landmarks
    a = findpostion(frame1)
    b = findnameoflandmark(frame1)

    if len(b and a) != 0:
        fingersum = 0
        for id in range(0, 4):
            if a[tip[id]][2:] < a[tip[id]-2][2:]:   #compare position of finger tip to
position of second preceding index
                fingersum += finger_binary_dict[tipname[id]]   #sum up to send
fingersum to arduino using dictionary values
```

```
        fingersum_int = int(str(fingersum), 2)

        print("Integer =", fingersum_int)

        # Send fingersum to Arduino
        ser.write((str(fingersum_int) + '\n').encode())

    cv2.imshow("Frame", frame1)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("s"):      #break out of loop by pressing s key on keyboard
        break

cap.release()
cv2.destroyAllWindows()
ser.close()
```