INTERNET OF THINGS

19I512

GESTURE CONTROLLED HOME AUTOMATION SYSTEM

Batch 2

Anand Narayanan N	**21i205**
Chandru M	**21i209**
Kishoar S	**21i224**
Nithish T	**21i232**
Kavin S	**22i435**

TABLE OF CONTENTS

Introduction:

The Internet of Things (IoT) Mini Project undertaken by our team focuses on Gesture Controlled Home Automation. In this report, we will discuss the concept of gesture control and how we integrated it with common household appliances like temperature sensor, lights and speaker volume Control using open-cv, mediapipe and Arduino Uno.

Objective :

The primary objective of this project is to enhance accessibility in home automation for individuals with limited or no speech capabilities. By integrating gesture-based control, we aim to provide an alternative and user-friendly interface that empowers people with communication challenges to independently manage their living environments. This project not only promotes inclusivity but also fosters increased convenience and efficiency in home automation, ultimately improving the quality of life for individuals with diverse abilities.

What is Gesture Control? :

Gesture control is a technology that enables the control of electronic devices through hand or body movements without the need for physical contact. It relies on sensors, cameras, or other motion-sensing technology to interpret gestures as commands. Gesture control is becoming increasingly popular in consumer electronics, gaming, and home automation due to its intuitive and hands-free nature. It enhances user interaction and convenience, making it a promising area of innovation in IoT.

Gesture Analysis and Home Automation :

In our project, we combined gesture control with common household appliances, such as fans and lights, using Arduino. We utilized sensors, like Temperature and Humidity Sensors to analyze the environmental conditions and they can be controlled using specific hand movements and gestures. When a user performs a predefined gesture, the Arduino microcontroller interprets it as a command to control the appliances. For example, a simple wave of the hand might turn on the lights, while another hand gesture that resembles the count of one might adjust the speaker volume. This integration not only adds a futuristic touch to home automation but also increases energy efficiency and convenience. This is also equipped with Google Text To Speech Module.

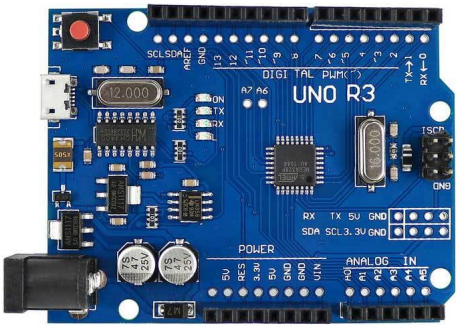
Components Required :

Hardware Components :

1.Breadboard

A breadboard, solderless breadboard, or protoboard is a construction base used to build semi-permanent prototypes of electronic circuits.

2. Arduino Uno



The Arduino UNO is the best board to get started with electronics and coding. If this is your first experience tinkering with the platform, the UNO is the most robust board you can start playing with. The UNO is the most used and documented board of the whole Arduino family

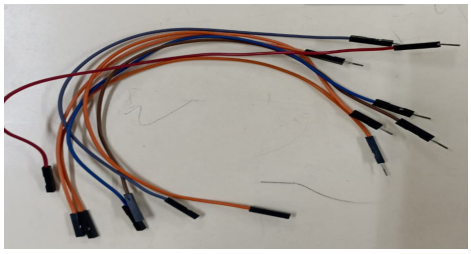
3. Temperature Sensor(DHT11)

The **DHT11** is a commonly used **Temperature and humidity sensor that** comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

4. LED

A Light Emitting Diode (LED) is a semiconductor device, which can emit light when an electric current passes through it.

5. Jumper Wires



Jumper wires are widely used in IoT (Internet of Things) projects and electronics prototyping due to their versatility and ease of use. These wires, typically made of flexible, insulated conductive material, serve several important purposes in IoT development and experimentation.

6.Arduino Uno Cable

Arduino Uno cables are commonly used in IoT (Internet of Things) applications for various purposes due to their versatility and widespread compatibility. In IoT, these cables serve several crucial purposes such as power supply, data transfer, serial communication, sensor configurations, gateway connections and firmware updates.

Software Components:

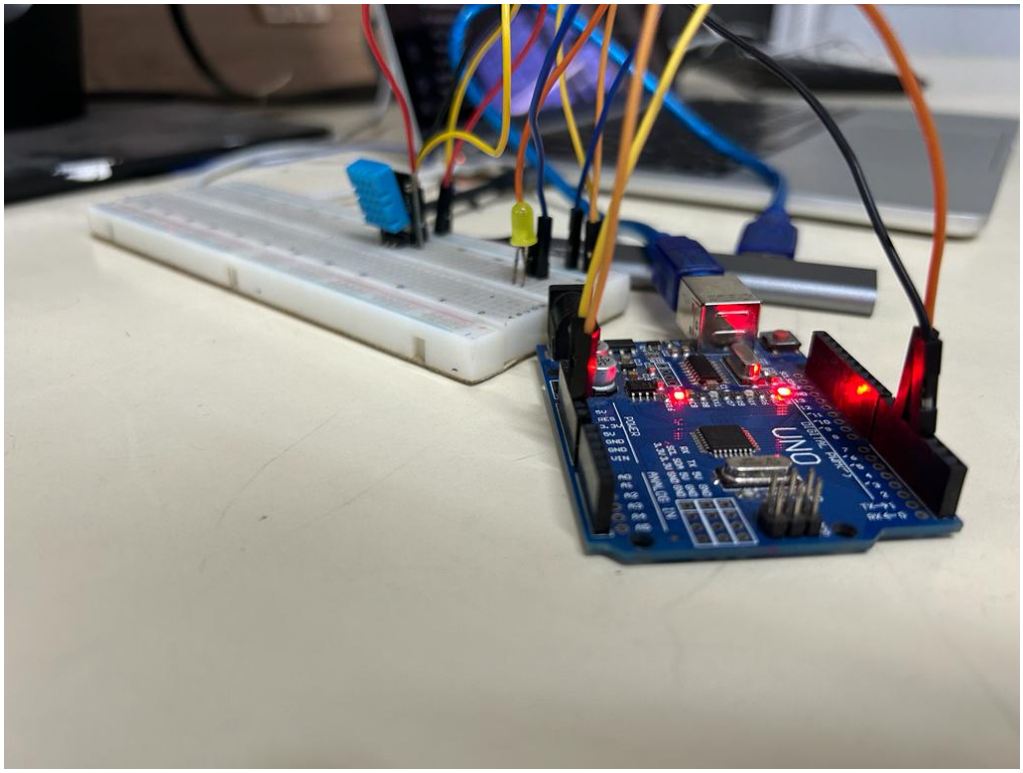
- Thingspeak platform
- OpenCV
- Google TTS

Thingspeak:

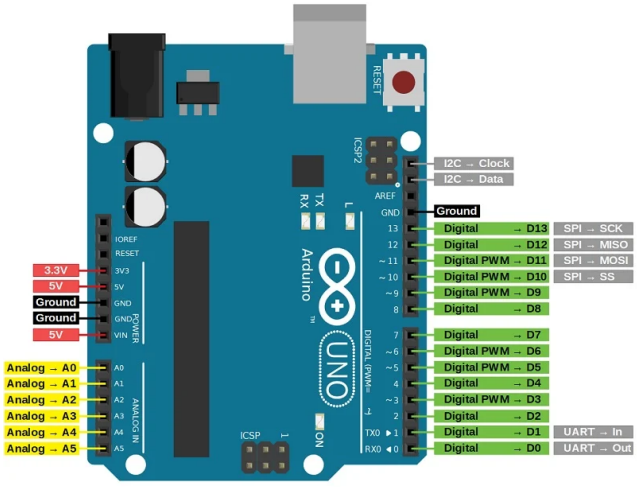
ThingSpeak is an open-source Internet of Things (IoT) platform developed by MathWorks. It provides a cloud-based environment for collecting, processing, analyzing, and visualizing data from IoT devices, sensors, and applications. ThingSpeak allows users to collect data from a wide range of IoT devices and sensors, such as temperature sensors, humidity sensors, GPS modules, and more. It supports multiple data formats, including numerical values, text, and timestamps. The platform offers data storage in the form of channels. Users can create multiple channels to organize and store different types of data. Each channel can store millions of data points, making it suitable for long-term data collection. ThingSpeak includes a built-in MATLAB analysis and visualization engine. Users can apply custom MATLAB code to preprocess and analyze the data as it is ingested. The platform provides various visualization options, such as line charts, bar graphs, heatmaps, and gauges, to help users create interactive and real-time visual representations of their data. These visualizations can be embedded in websites or shared with others. The platform supports various IoT protocols, including MQTT, HTTP, and RESTful API, making it easy to integrate with a wide range of IoT devices and

services. It provides a user-friendly interface for IoT enthusiasts, developers, and engineers to collect, analyze, and share data from their connected devices and sensors.

Circuit Diagram:



Pinout Configuration:



Hardware Setup:

1. Connect the VCC pin of the DHT11 sensor to the 5V pin on the Arduino.
2. Connect the GND pin of the DHT11 sensor to the GND pin on the Arduino.
3. Connect the DATA pin of the DHT11 sensor to digital pin 2 on the Arduino.
4. Connect the LED to digital pin 13 on the Arduino.
5. Connect the LED's positive lead (usually longer) to digital pin 13 and its negative lead to ground

Working of the Project:

1. The gesture sensor continuously monitors hand movements.
2. When a specific gesture is detected, the microcontroller triggers the corresponding action. For example, if gesture 2 is detected, the lights turn on, and the microcontroller sends a command to the TTS module to say, "Lights turned on."
3. For gesture 3 (temperature), the microcontroller reads data from the DHT sensor, uploads it to ThingSpeak, and sends the temperature reading to the TTS module to voice it out.
4. For gesture 4 (face recognition), the microcontroller activates the camera, captures an image, processes it to identify faces, and provides voice feedback if faces are recognized.
5. The voice feedback module communicates with the TTS module to convert text messages into voice output.
6. The system repeats the process continuously, waiting for gestures, and providing voice-based feedback as necessary.
7. Users can control various home automation functions through gestures and receive audible feedback for each action.

Source code/Program:

```
import cv2
import time
import os
import HandTrackingModule as htm
import VolumeControl as vc
import Arduino as ard
from FaceRecognition import AttendanceSystem as ats
import sensor as se
from gtts import gTTS
import serial

ser = serial.Serial('/dev/cu.usbserial-1140', 9600, timeout=1)

def speak(text) :
speech = gTTS(text)
speech_file = 'speech.mp3'
speech.save(speech_file)
os.system('afplay ' + speech_file)


wCam, hCam = 1080, 720
cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)

folderPath = "HandTracker/Photos"
myList = sorted(os.listdir(folderPath))
print(myList)
overlayList = []
for imPath in myList :
image = cv2.imread(f'{folderPath}/{imPath}')
overlayList.append(image)
pTime = 0
```

```
detector = htm.handDetector(detectionCon=0.75)
tipIds = [4, 8, 12, 16, 20]
prevLs = list()

while True :
success, img = cap.read()
img = detector.findHands(img, draw=False)
lmList = detector.findPosition(img, draw=False)

if len(lmList) != 0 :
fingers = []

# Thumb
if lmList[tipIds[0]][1] > lmList[tipIds[0]-1][1] :
fingers.append(1)
else :
fingers.append(0)

# Four Fingers
for id in range(1, 5) :
if lmList[tipIds[id]][2] < lmList[tipIds[id]-3][2] :
fingers.append(1)
else :
fingers.append(0)
totalFingers = fingers.count(1)
print(totalFingers)
prevLs.append(totalFingers)
if prevLs[-1:max(-len(prevLs), -11):-1].count(0) >= 10 and
totalFingers == 0 :
# Off Command
print('Off')
speak('Switching Off Devices')
se.off(ser)
elif prevLs[-1:max(-len(prevLs), -11):-1].count(1) >= 10 and
totalFingers == 1 :
cv2.waitKey(10)
```

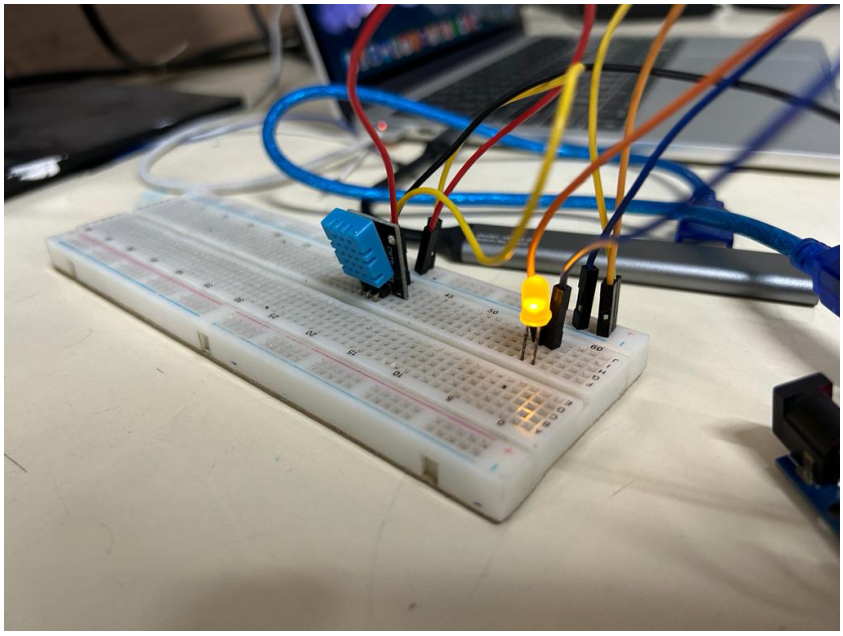
```
print('Controlling Volumne')
speak('Controlling Volume')
vc.volumeControl()
elif prevLs[-1:max(-len(prevLs), -11):-1].count(2) >= 10 and
totalFingers == 2 :
cv2.waitKey(10)
print('Switching On LED')
speak('Switching On LED')
se.led(ser)
elif prevLs[-1:max(-len(prevLs), -11):-1].count(3) >= 10 and
totalFingers == 3 :
cv2.waitKey(10)
print('Switching On Buzzer')
speak('Switching On Buzzer')
se.buzzer(ser)
elif prevLs[-1:max(-len(prevLs), -11):-1].count(3) >= 10 and
totalFingers == 3 :
cv2.waitKey(10)
print('Checking Temperature')
speak('Checking Temperature')
val = se.tempCheck(ser)
if val is not None :
speak(f'Temperature is currently {val} degree celsius')
else :
speak(f'Could not get the temperature')
elif prevLs[-1:max(-len(prevLs), -11):-1].count(4) >= 10 and
totalFingers == 4 :
cv2.waitKey(10)
speak('Recognizing Faces')
speak(ats.faceReg(5))
prev = totalFingers
cv2.rectangle(img, (20, 225), (170, 425), (0, 255, 0), cv2.FILLED)
cv2.putText(img, str(totalFingers), (45, 375),
cv2.FONT_HERSHEY_PLAIN, 3, (255, 0, 0), 3)
cTime = time.time()
fps = 1/(cTime - pTime)
```

```
pTime = cTime
cv2.putText(img, f'FPS : {int(fps)}', (400,70),
cv2.FONT_HERSHEY_PLAIN, 3, (0, 255, 0), 3)
cv2.imshow("Image", img)
cv2.waitKey(10)
```

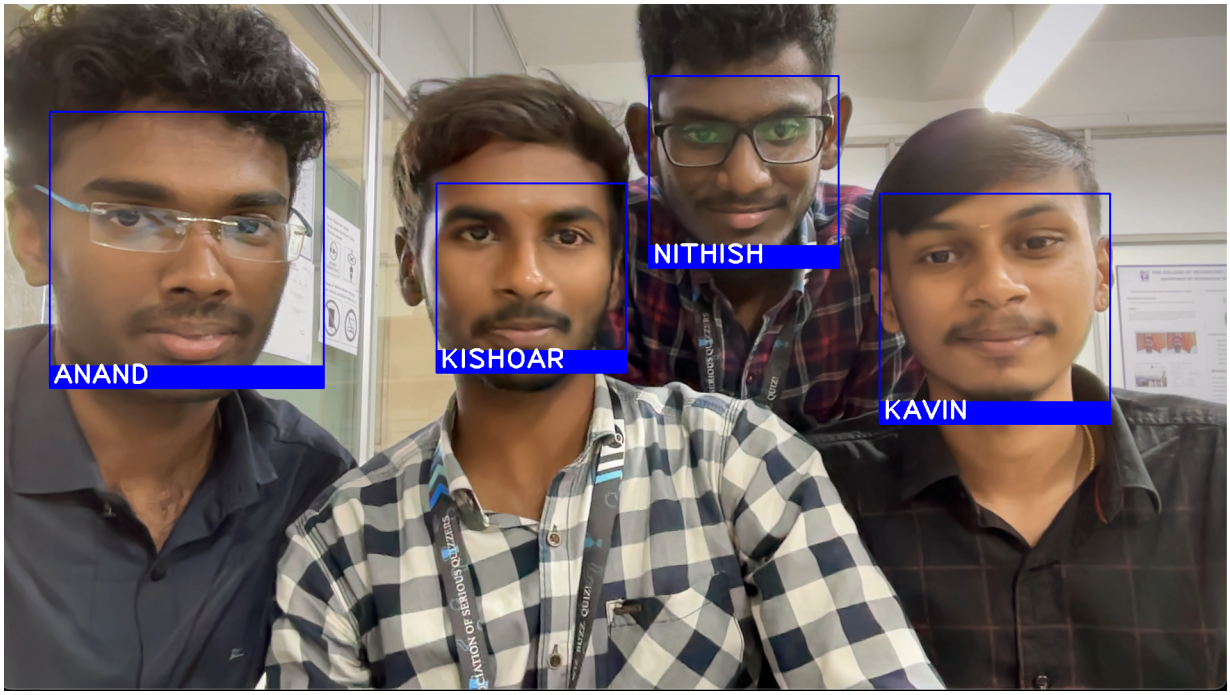
Testing and Results:

- **Gesture 1 Speaker Volume Control :**

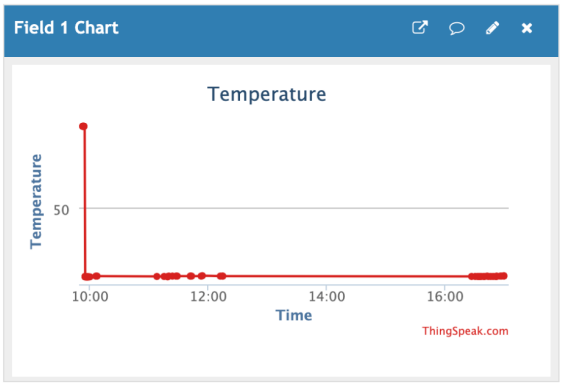
- **Gesture 2 Lights Control :**

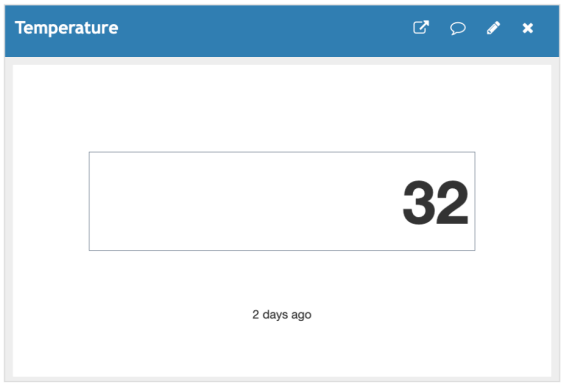


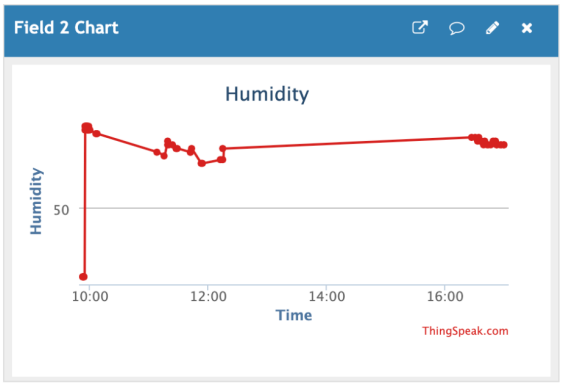
- **Gesture 3 Face Recognition :**

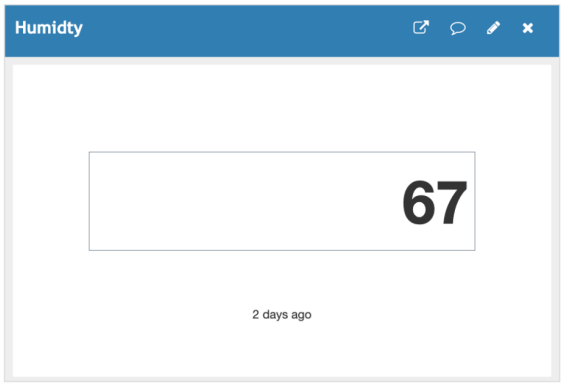


Temperature and Humidity Values and Visualizations :









Future Work :

1. Enhance gesture recognition with a larger library.
2. Integrate voice assistants and improve face recognition.
3. Develop a mobile app for remote control.
4. Prioritize energy efficiency and user-friendly interfaces.

Conclusion:

In our gesture-controlled home automation project, we've successfully combined gesture recognition technology with IoT components for intuitive and user-friendly device control. Through gesture sensors, a microcontroller, and cloud connectivity, we enable users to seamlessly manage their home environment. The integration of voice feedback further enhances the user experience. This project demonstrates the potential of IoT in home automation, offering customization options and potential energy efficiency improvements. It showcases the intersection of hardware, software, and user-centric design, making it a compelling example of innovation in the field of smart home technology.