<u>TEAM:GRUHRAKSHAK</u>

Project Name : Home Security

Team members: -

1. Harshal Patil (FY23F130)
2. Akshay Dhere (FY23F141)
3. Tanmay Kanade (FY23F170)
4. Harshal More (FY23F173)

Abstract Idea of the Project

Home security has become very important now a days as both individuals in family are working and children are often alone in house. The basic idea of our project is to make completely self dependant home security system. The system will be able to detect events such leakage of flammable and harmful gases, smoke along with raise in temperature. This project presents a comprehensive approach to home security using a unique combination of software and hardware. The system is able to integrate various smart devices such as cameras, alarm systems along with real time data vitalization and analysis in order to create a robust security framework. System will be able to work efficiently even in case of power failure.

Introduction

The system will comprise a central unit and multiple smaller units which are placed across house for comprehensive coverage. Each unit is equipped with different sensors to detect hazards such as gas or fire breakout. Whenever a treat is detected small unit immediately communicates signal to the central unit. According to fixed threshold which were set by user central unit analyses the information and initiates an appropriate response protocol. Moreover our anti burglary system utilizes detection to identify intruders. It will also consist of a motion sensor to capture and detect motion outside the house. Upon any movement detection and unrecognized detection small unit relays signal to central unit which will immediately turn on surveillances cameras to capture and store live feed of intruder. This feed can be viewed though anywhere in world by simply accessing a provided URL.

Components Used

To execute our project we used following components :-

- Raspberry pi Zero w: -

  A Raspberry Pi is a single-board computer (SBC) that's the size of a credit card and can be used to turn a monitor, TV, mouse, or keyboard into a PC.

- ESP 32: -

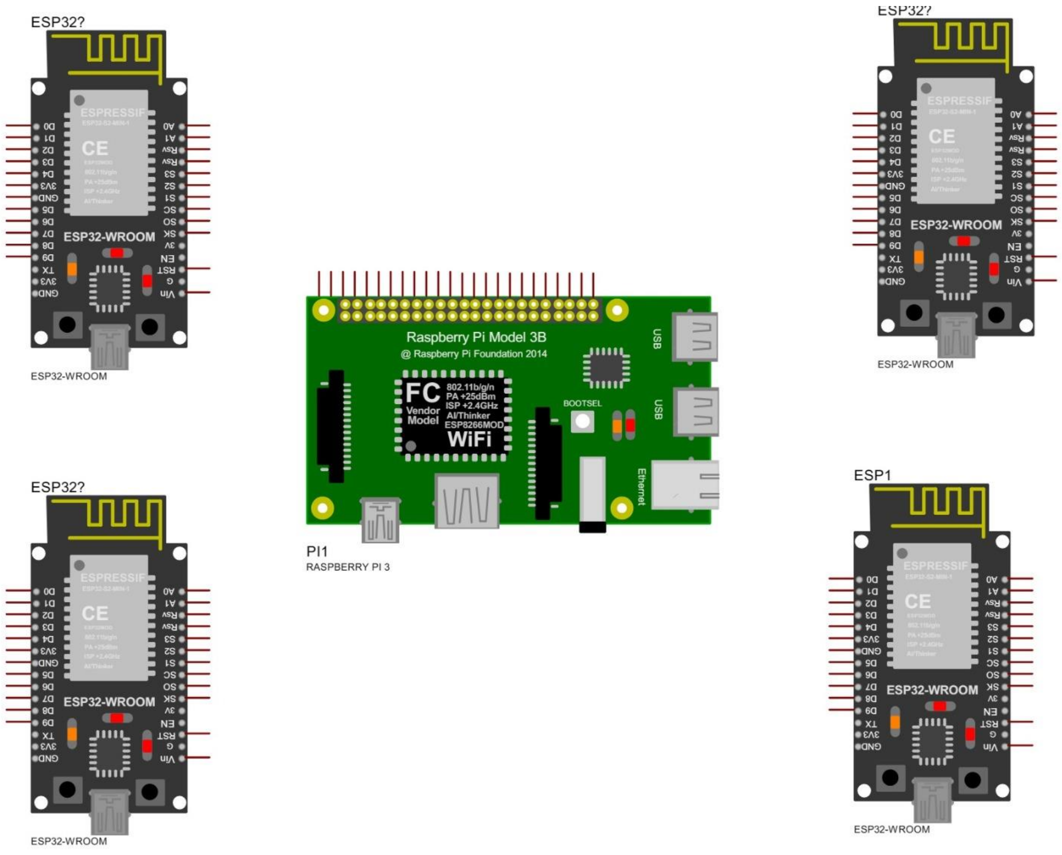
  ESP32 is a microcontroller and system-on-chip (SoC) that provides Wi-Fi and Bluetooth connectivity for embedded devices, or IoT devices. It's a low-cost, low-power chip from Espressif Systems.

- AHT – 21 Temperature and Humidity Sensor: -

  The AHT21 is a high-precision temperature and humidity sensor module that can be used with Arduino or microcontrollers. It has an I2C interface, so it's easy to communicate with and Arduino libraries are available.

Working of the model

- A Raspberry Pi board will act a main server.
- Multiple ESP32 modules can be placed at various part of the house. Raspberry and ESP32 will communicate with each other using HTTP protocol.



- Each ESP32 module will be equipped with a ATH-21 temperature sensor.

- ATH-21 will keep taking readings after a interval of 5 sec. Data received from AHT-21 will be plotted in form of a graph in real time.

- This data will be sent to a access point that can be accessed via browser.

- A camera module able to stream real time video and take photos. The camera output will be sent to another access point that can be accessed via browser

AHT 21B sensor

Code for the prototype

- Code for ESP 32 as a client: -

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_AHTX0.h>
#include <ArduinoJson.h>

const char *ssid = "ssid";
const char *password = "password";
const char *serverIP = "Raspberry pi IP address";
const int serverPort = 5000; // Change this to your Flask server port
const char *endpoint = "/temperature_humidity_data";

Adafruit_AHTX0 aht;

void setup() {
  Serial.begin(115200);
  delay(100);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");

  if (!aht.begin()) {
    Serial.println("Could not find AHT? Check wiring");
    while (1) delay(10);
  }
  Serial.println("AHT21 found");
}

void loop() {
  float temperature, humidity;

  // Read temperature and humidity from AHT21 sensor
  if (readAHT21Data(&temperature, &humidity)) {
    sendTemperatureHumidityData(temperature, humidity);
  }
  delay(5000); // Delay before next reading
}

bool readAHT21Data(float *temperature, float *humidity) {
  sensors_event_t humidity_event, temperature_event;

  if (!aht.getEvent(&humidity_event, &temperature_event)) {
    Serial.println("Failed to read from AHT21 sensor!");
    return false;
  }

  *temperature = temperature_event.temperature;
  *humidity = humidity_event.relative_humidity;

  if (isnan(*temperature) || isnan(*humidity)) {
    Serial.println("Invalid data from AHT21 sensor!");
    return false;
  }

  return true;
}

void sendTemperatureHumidityData(float t, float h) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    String url = "http://" + String(serverIP) + ":" + String(serverPort) +
endpoint;

    http.begin(url);
    http.addHeader("Content-Type", "application/json");

    // Create JSON payload
    StaticJsonDocument<200> jsonPayload;
    jsonPayload["temperature"] = t;
    jsonPayload["humidity"] = h;

    String jsonString;
    serializeJson(jsonPayload, jsonString);

    // Send HTTP POST request
    int httpResponseCode = http.POST(jsonString);

    if (httpResponseCode > 0) {
      Serial.print("Temperature: ");
      Serial.print(t);
      Serial.print("°C, Humidity: ");
      Serial.print(h);
      Serial.println("%");
      Serial.print("Data sent successfully. Response code: ");
      Serial.println(httpResponseCode);
    } else {
      Serial.print("Error sending data. Error code: ");
      Serial.println(httpResponseCode);
      Serial.println(http.errorToString(httpResponseCode).c_str()); // Print
HTTP error
    }

    http.end();
  } else {
    Serial.println("WiFi not connected");
  }
}
```

After entering you ssid that is you WiFi name, WiFi password and correct IP of your raspberry pi the ESP 32 will be ready to send and receive data to the main server.

The readAHT21Data() function reads temperature and humidity data from the AHT21 sensor and returns true if the data is valid.

The sendTemperatureHumidityData() function constructs a JSON payload containing temperature and humidity data, sends an HTTP POST request to the Flask server with the payload, and prints the response from the server.

- Code for Raspberry Pi as a sever

```
from flask import Flask, render_template, Response, request, send_file,
jsonify
import io
import picamera
import seaborn as sns
import matplotlib.pyplot as plt
import base64
import time

app = Flask(__name__)

temperature_data = []
humidity_data = []
plot_data = None

def generate_frames():
    with picamera.PiCamera() as camera:
        camera.resolution = (640, 480)
        camera.framerate = 24
        stream = io.BytesIO()

        for _ in camera.capture_continuous(stream, 'jpeg',
use_video_port=True):
            stream.seek(0)
            yield b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' +
stream.read() + b'\r\n'
            stream.seek(0)
            stream.truncate()

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

@app.route('/temperature_humidity_data', methods=['POST'])
def receive_temp_humidity_data():
    data = request.json
    timestamp = time.strftime('%Y-%m-%d %H:%M:%S')

    temperature_data.append({'timestamp': timestamp, 'temperature':
data['temperature']})
```

```
    humidity_data.append({'timestamp': timestamp, 'humidity':
data['humidity']})

    print(f"Received Temperature: {data['temperature']}°C, Humidity:
{data['humidity']}%, Timestamp: {timestamp}")

    return "Temperature and humidity data received successfully"

@app.route('/plot_data')
def plot_data():
    global plot_data
    # Generate the plot
    if not plot_data:
        sns.set(style="whitegrid")
        plt.figure(figsize=(8, 6))
        plt.plot([1, 2, 3, 4], [10, 20, 25, 30])  # Example plot data
        plt.xlabel('X-axis')
        plt.ylabel('Y-axis')
        plt.title('Example Plot')
        plt.tight_layout()

        # Save the plot to a BytesIO object
        img_bytesio = io.BytesIO()
        plt.savefig(img_bytesio, format='png')
        plt.close()

        # Encode the plot image to base64
        img_base64 = base64.b64encode(img_bytesio.getvalue()).decode('utf-8')

        plot_data = {'image': img_base64}

    return jsonify(plot_data)

@app.route('/')
def index():
    return """
    <!DOCTYPE html>
    <html>
    <head>
        <title>Real-Time Plot Viewer</title>
        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    </head>
    <body>
        <h1>Real-Time Plot Viewer</h1>
        <div id="plot-container">
            <!-- Plot image will be loaded here -->
        </div>
        <script>
            function updatePlot() {
                $.get("/plot_data", function(data) {
                    // Update the plot container with the new plot image
                    $("#plot-container").html('<img
src="data:image/png;base64,' + data.image + '">');
                });
            }

            // Update the plot every 5 seconds
            setInterval(updatePlot, 5000);

            // Initial plot update
            updatePlot();
        </script>

        <h1>Live Video Feed</h1>
        <img src="/video_feed" width="640" height="480">
    </body>
    </html>
    """

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Once this code is uploaded and runed on the raspberry software your raspberry pi will start acting as a flask server and get connected to ESP 32 once it is connected to the same WiFi as ESP 32

generate_frames() thisfunction is a generator that continuously captures frames from a Raspberry Pi camera using Picamera. It yields each frame as a multipart response, allowing for live video streaming.It is used by the /video_feed route to provide the live video feed.

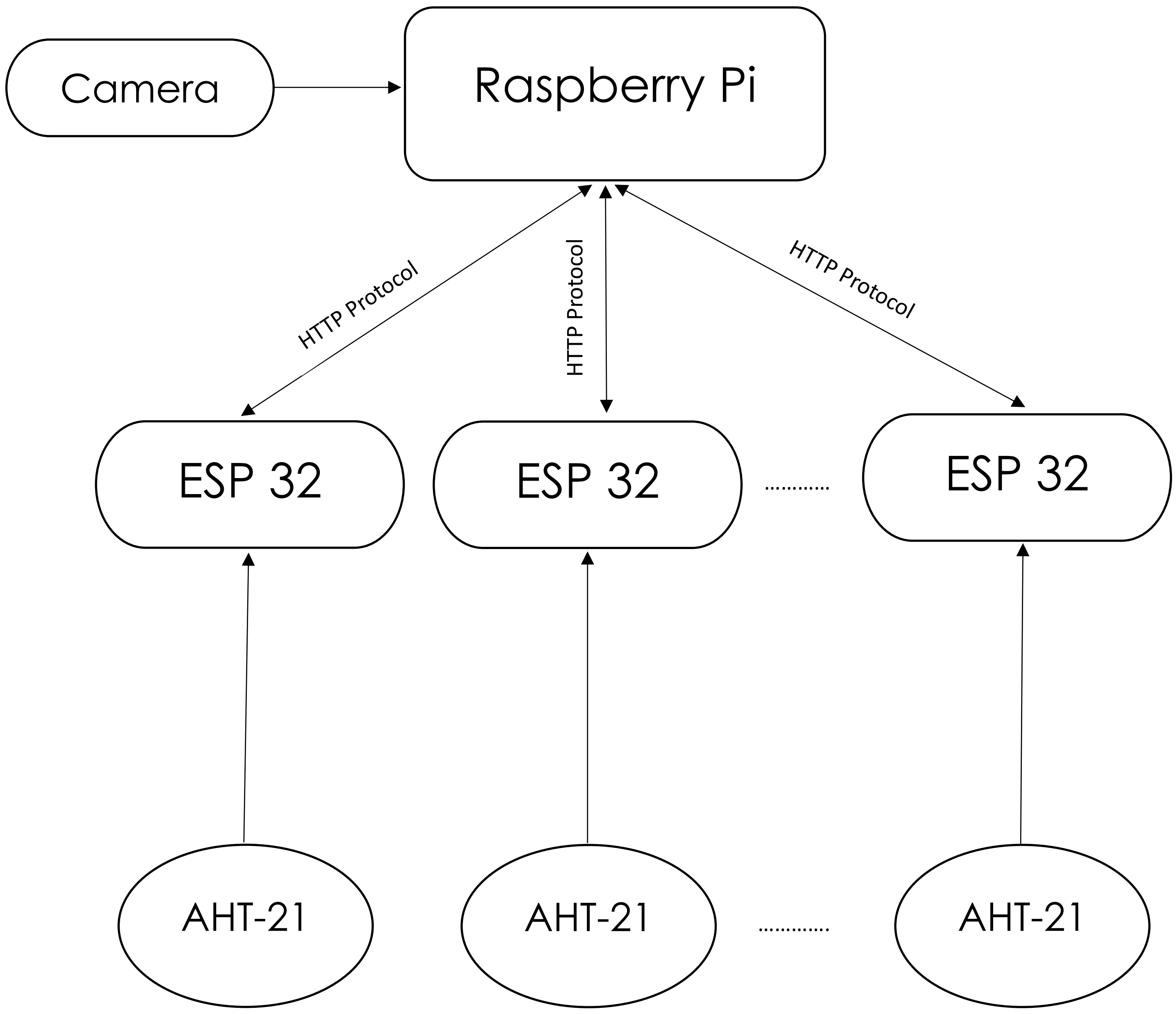
video_feed() this route function returns the response generated by the generate_frames() function.It sets the response MIME type to multipart/x-mixed-replace, indicating that multiple images will be sent in the response with each image replacing the previous one.

receive_temp_humidity_data() thisroute function receives temperature and humidity data from a client device via HTTP POST requests.It extracts the data from the JSON payload in the request.It adds a timestamp to the data and appends it to global lists temperature_data and humidity_data.It prints the received data and a success message to the console. It returns a simple acknowledgment message as the response.

plot_data() this route function generates a plot using Seaborn and Matplotlib libraries. If plot data is not available (i.e., the first request), it creates an example plot. It encodes the plot image to base64 format and returns it as JSON. It caches the generated plot to avoid regenerating it for subsequent requests.

index() this route function serves the HTML page for the application. It includes JavaScript code to periodically update the plot image and load the live video feed. It sends the initial plot image and loads the video feed when the page is loaded. It combines the plot container and video feed in a single HTML page for user interaction.

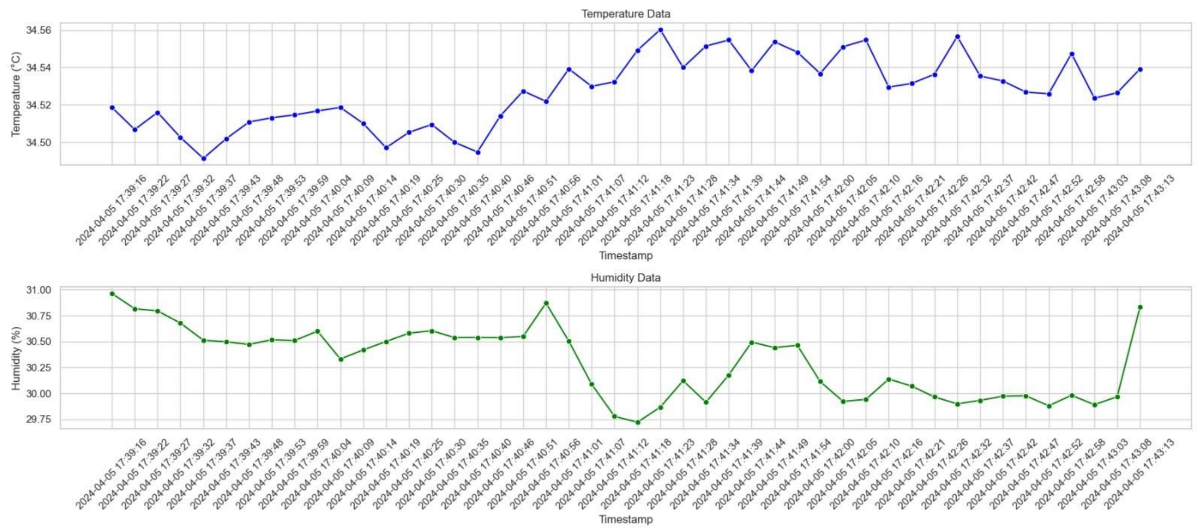
Flowchart



Camera
Raspberry Pi
HTTP Protocol
HTTP Protocol
HTTP Protocol
ESP 32
ESP 32
....
ESP 32
AHT-21
AHT-21
....
AHT-21

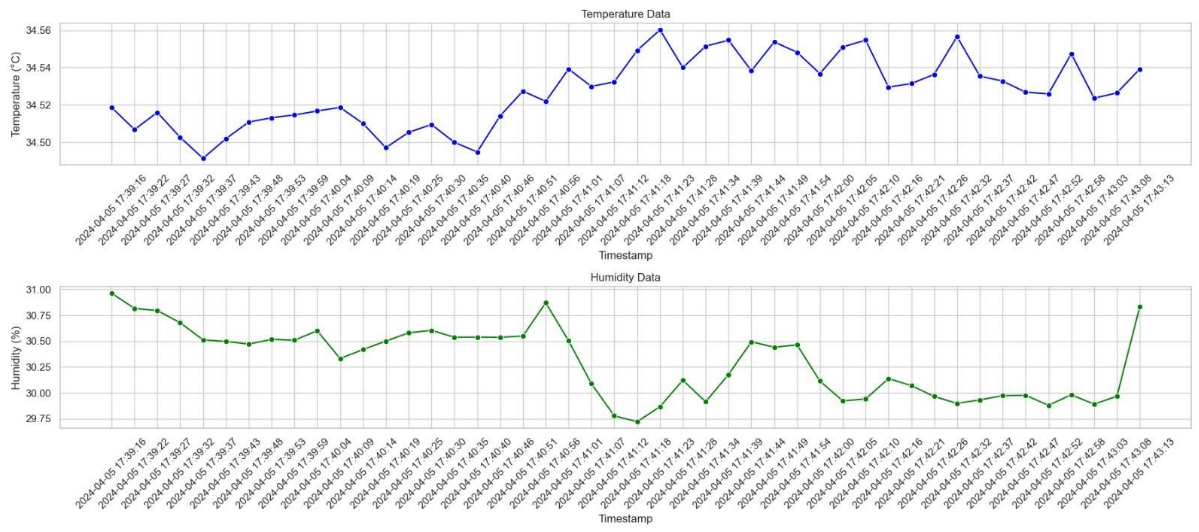
Output

- AHT-21 data



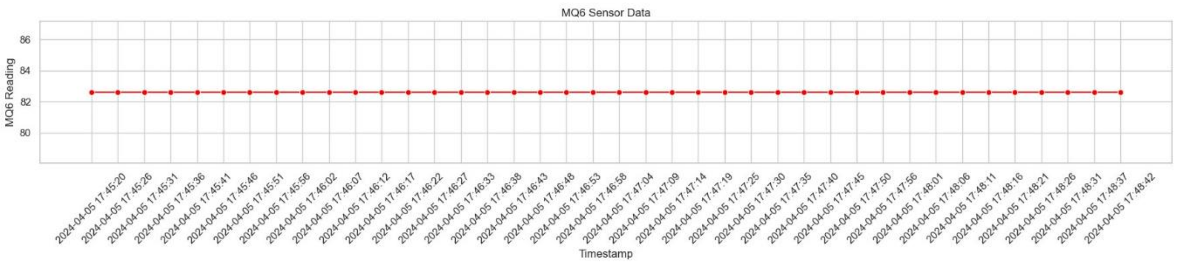
In the picture above you can see that the AHT-21 takes readings after every 5 sec, this delay can be adjusted as desired time also not only it can take readings it plots graph of the readings, and the range of the sensor is in a very small range showing the accuracy of the sensor.

- MQ6 Gas sensor data

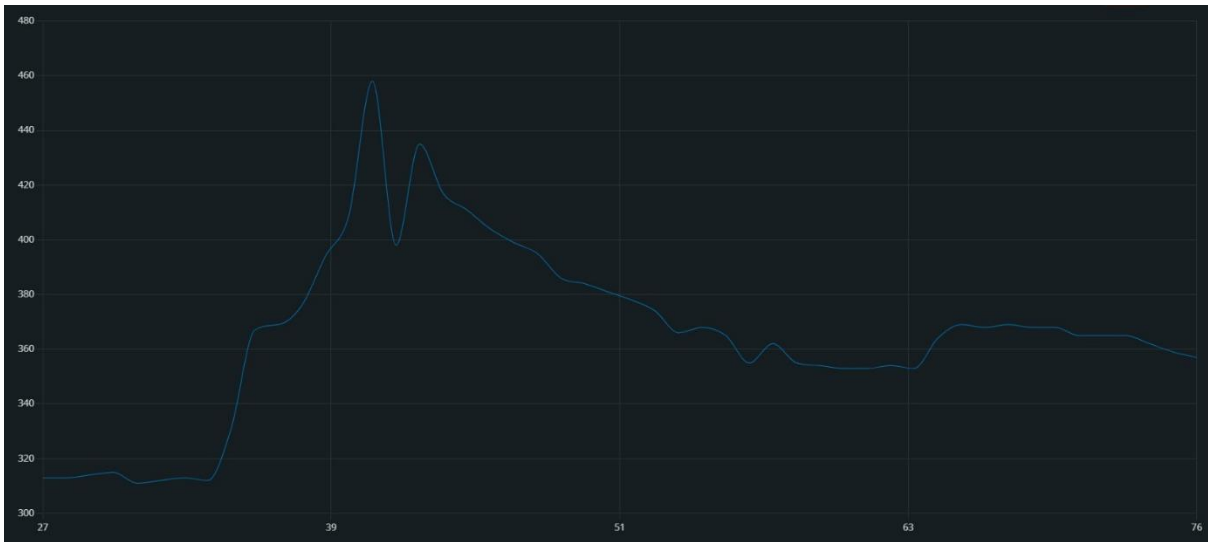


In the picture below you can see that the MQ6 Gas sensor takes readings after every 5 sec, this delay can be adjusted as desired time also not only it can take readings it plots graph of also as stated is showing the accuracy of the sensor.
 During the demonstration as there was no fire or any gas leak condition the readings taken were constant.

This constant reading ensures that during normal working environment and no hazardous gas leakage there won't be any false alarms raised for gas leak thus depicting the accurate readings.



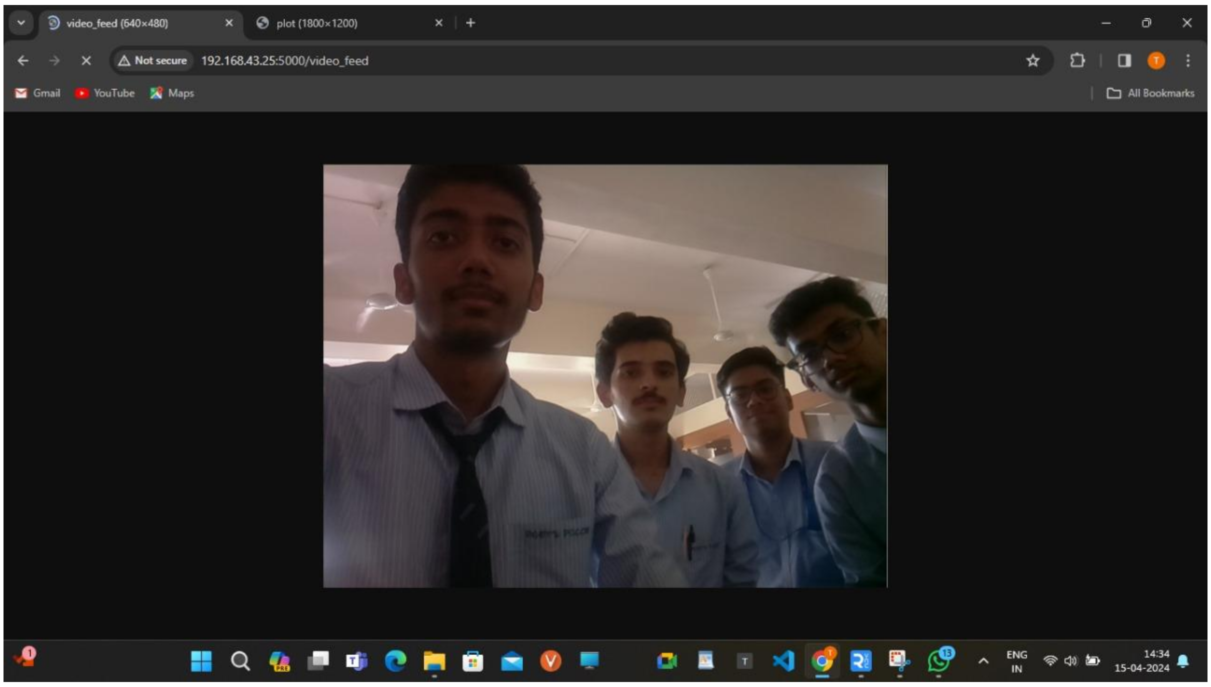
A plot of LPG gas leak sensing as an output is given below :-
Here the sensor was held near to the gas stove and was left open to detect the gas leak using the sensor. The peaks of width are of the duration for which the gas was leaking



- Raspberry Pi zero Camera streaming

For the home security a live video feed is an essential and inseparable part of any integrated system aimed for security.
Hence we used a 5 MP camera and wrote the code for wireless transmission of live video to devices and also on the http flask server.

Here is the image clicked by our group on a different device connected to the same network.



Applications & Future aspects

- <u>Monitoring Health and comfort :</u>

  With integration of the smart devices such as watches we can even track health and comfort of the user. Along with that we can also collect information about indoor air quality, thermal comfort, etc.

- <u>AI & Machine learning integration :</u>

  Combining temperature sensors with AI and machine learning could enable predictive analytics and proactive maintenance. System might predict equipment failures and warns users about it.

- <u>Remote monitoring and control :</u>

  With more components of iot lead to personalized control. By this user can access the devices using given api and able to check home and surroundings across the globe.

- <u>Development of User interface :</u>

  Better user interface along with the api help user to easily access the information also to take further actions such as to call emergency authorities.

- <u>Interoperability and Standards:</u>

  Future home automation systems should focus on interoperability and standards to ensure they work well with other smart home devices and platforms. This would let users create comprehensive smart home setups with different functions.

- <u>Network Security and Privacy :</u>

  As these all devices are connected over a single WiFi network security becomes a concern. In future we will thrive to use different protocol such as ESP-NOW which will use built in capabilities of modules to develop a successful communication network along with user id and password which will provide robust security to the whole home automation system.