NAMIBIA UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ELECTRONIC DESIGN PROJECT 415 (ECD811S)

DESIGN 2 REPORT: IOT ENABLED GAS DETECTION AND TEMPERATURE MONITORING SYSTEM

Name: NAMASIKU VERONICA MUSHAUKWA

Student ID: 221053530

Prepared for: DR ZACCHAEUS OYEDOKUN

23/05/24

ACKNOWLEGMENT

I'm incredibly grateful to the University of Science and Technology, especially the Department of Electrical and Computer Engineering. This course ignited a passion for creativity and innovation within me, and the learning opportunities were invaluable.

My deepest thanks go to Dr. Oyedokun, whose guidance was instrumental in completing this module. His insightful advice and enthusiasm helped me optimize my project design, ensuring it met all the requirements and allowing me to finish on time.

Finally, a heartfelt thank you to my family and friends. Their unwavering support, especially during challenging moments, kept me motivated and reminded me that perseverance and hard work are key to success. I'm also grateful to everyone who offered constructive criticism and guidance – these contributions were essential to this journey.

LIST OF ACRONYMS

IoT	Internet of Things
LPG	liquefied petroleum gas
Wi-Fi	Wireless Fidelity
OS	Operating System
MCU	Microcontroller Unit

TERMS OF REFERNCE

It is the request from the ECD811s Coordinator and lecturer Dr. Oyedokun that a final report for the IoT based workstation be done. It is also a necessity that all projects done should have a technical report compiled. The information given in this report should show a step-by-step information of how the project was carried out and this should allow other technicians to replicate the project elsewhere without too much difficulty since all key specifications and requirements will be fully outlined.

Table of Contents

INTRODUCTION

In the wake of rising concerns about gas-related accidents, traditional methods of gas leak detection, which often involve manual inspections, are proving to be insufficient. This project presents a novel approach – an IoT based gas leakage and temperature monitoring system. This innovative system utilizes strategically placed sensors to continuously monitor for the presence of combustible gases and any concerning fluctuations in ambient temperature. The sensor data is then transmitted wirelessly to a central hub or cloud platform, enabling real-time monitoring and analysis. This approach offers a multitude of advantages over conventional methods. Firstly, the system provides continuous monitoring, eliminating the human error and time constraints associated with manual checks. Secondly, remote access to data allows for constant vigilance, even when the premises are unoccupied. Most importantly, the system can be programmed to trigger automated responses in critical situations. Upon detecting a gas leak or an abnormal temperature rise, the system can automatically trigger alarms, shut off gas supply valves, and send emergency notifications to designated personnel. This real-time response capability significantly enhances safety by providing immediate warnings and enabling prompt action to mitigate potential hazards. By leveraging the power of IoT technology, this project aims to create a safer and more secure environment for everyone.

OBJEVTIVES

- Design and develop an IoT-based gas leakage detection & temperature monitoring system.
- Integrate alerting mechanisms for timely notification.
- Enable remote monitoring and control of the system.

LITERATURE REVIEW

Gas leaks pose a significant threat to life and property. Traditional methods for gas leak detection, often reliant on manual inspections, are proving to be inadequate due to factors like human error and time constraints. This necessitates the exploration of more advanced solutions. The IoT offers a promising approach for developing intelligent gas leak and temperature monitoring systems.

Several research studies have explored the potential of IoT in this domain. A study by Sharma et al. (2023) proposes an IoT-based system that utilizes gas sensors for leak detection and transmits data to a cloud platform for real-time monitoring. This approach enables remote access and facilitates the triggering of automated responses, such as alarm activation and gas valve shut-off. Similarly, Pandley et al. (2017) investigates the use of MQ-sensor technology within an IoT framework for gas leak detection and alerting. Their research highlights the potential for improved security in industrial settings through multi-sensor combinations.

Furthermore, Rachana et al. (2022) emphasizes the role of IoT in not only enhancing safety but also in streamlining processes like automatic gas booking. This highlights the multifaceted benefits of IoT integration. Additionally, Sharma et al. (2023) provides a comprehensive review of various sensor-based and non-sensor-based IoT systems for gas leak detection. This study underscores the importance of cost-effectiveness, scalability, and continuous improvement in the development of such systems.

While existing research has laid a strong foundation, there's significant potential for further development in IoT-based gas leakage and temperature monitoring systems. Future efforts should focus on refining sensor technology to improve gas detection accuracy and minimize false alarms. Advanced data analytics can be harnessed for real-time anomaly detection and even predictive maintenance, identifying potential issues before they escalate. Additionally, seamless integration with existing building management systems can create a more holistic approach to safety and efficiency. Finally, establishing standardized protocols and communication interfaces would enable broader interoperability between different systems, fostering a more flexible and scalable approach to system design and implementation. By addressing these future directions, IoT-based gas leakage and temperature monitoring systems can become even more robust and widely applicable, ultimately creating a safer and more secure environment.

LIMIT OF STUDY

The types of gases detected will be limited to methane ($CH4$), carbon monoxide (CO), and liquefied petroleum gas (LPG), while other gases will not be considered. The temperature monitoring component will function within a range of -20°C to 80°C, and measurements outside this range will be excluded. Geographically, the study will focus on urban residential settings or industrial zones within a defined area, without extending to remote or extreme environments. Standard environmental conditions will be considered, excluding extreme weather events. Technologically, the study will use commercially available sensors with predefined sensitivity and accuracy levels, avoiding custom or experimental designs. It will explore common IoT communication protocols such as Wi-Fi, excluding less established or emerging protocols. Scalability assessments will be limited to small to medium-scale deployments, excluding large-scale or nationwide feasibility. Performance metrics will focus on the accuracy and precision of gas detection and temperature readings under controlled conditions, without extensively studying long-term wear and environmental impacts on sensor performance.

PROCEDURES

1. Selected and gathered components.
2. Tested individual components for functionality.
3. Designed and assembled the circuit in fritzing software and then on a breadboard.
4. Developed the codes in the Thonny software on the raspberry OS.
5. Debugged issues, such as interference from jumper wires, to ensure successful code uploads.
6. Configured the webapp using html and embedded it in the code.
7. Integrated hardware with software for a complete prototype.

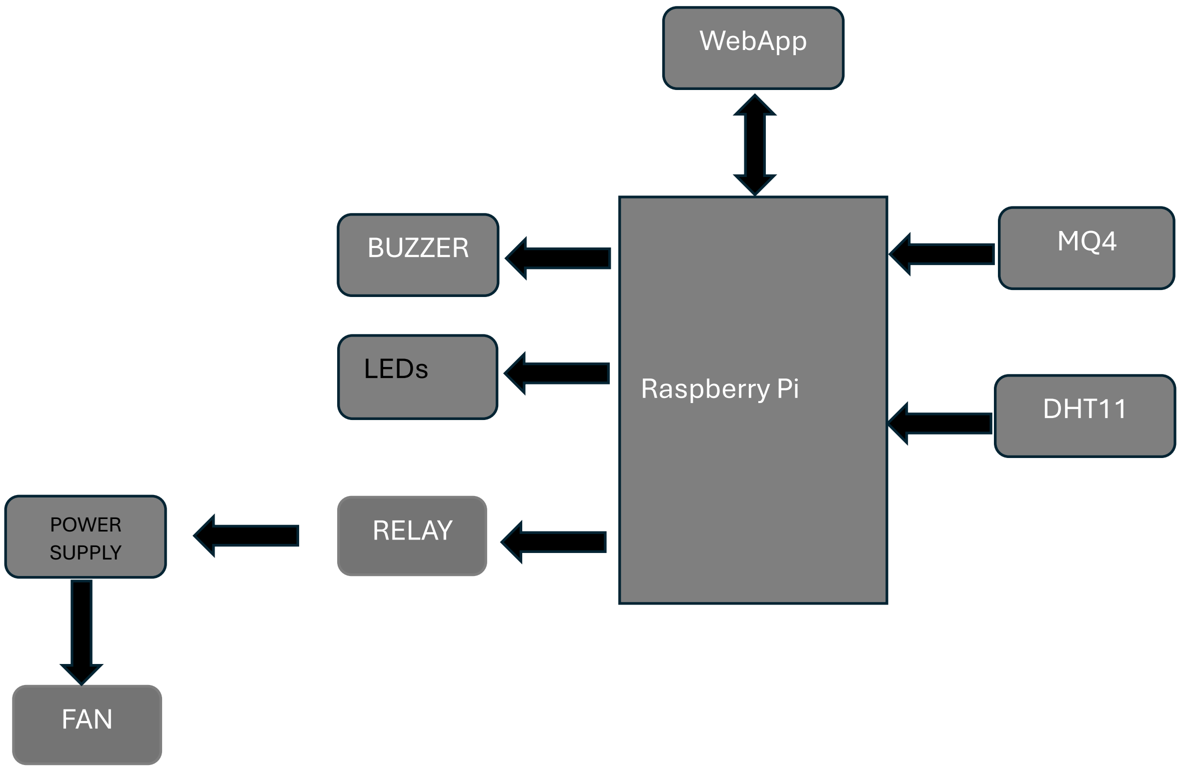
8. Conducted tests for temperature monitoring and smoke detection, relay operation, and real time webapp monitoring.
9. Documented project steps, configurations, and findings.

DESIGN AND IMPLEMENTATION

Components:

- ❖ Raspberry Pi3: Acts as the MCU and the connectivity hub.
- ❖ Web Application: IoT platform for remote monitoring and data analytics
- ❖ MQ4: This is the gas detecting sensor. Supplied with 5volts from the raspberry pi.
- ❖ DHT11: This is the temperature measuring sensor. Supplied with 3.3volts from the raspberry pi.
- ❖ Buzzer: Actuator that produces audible alert in response to gas detection.
- ❖ Fan: Actuator that responds to high temperatures and gas detections.
- ❖ Relay: Relays power from the power supply to the 12V fan.
- ❖ LEDs: Actuators that turn on based on the state of the temperature.
- ❖ Jumper wires
- ❖ Breadboard

Block Diagram:



Schematic from fritzing:

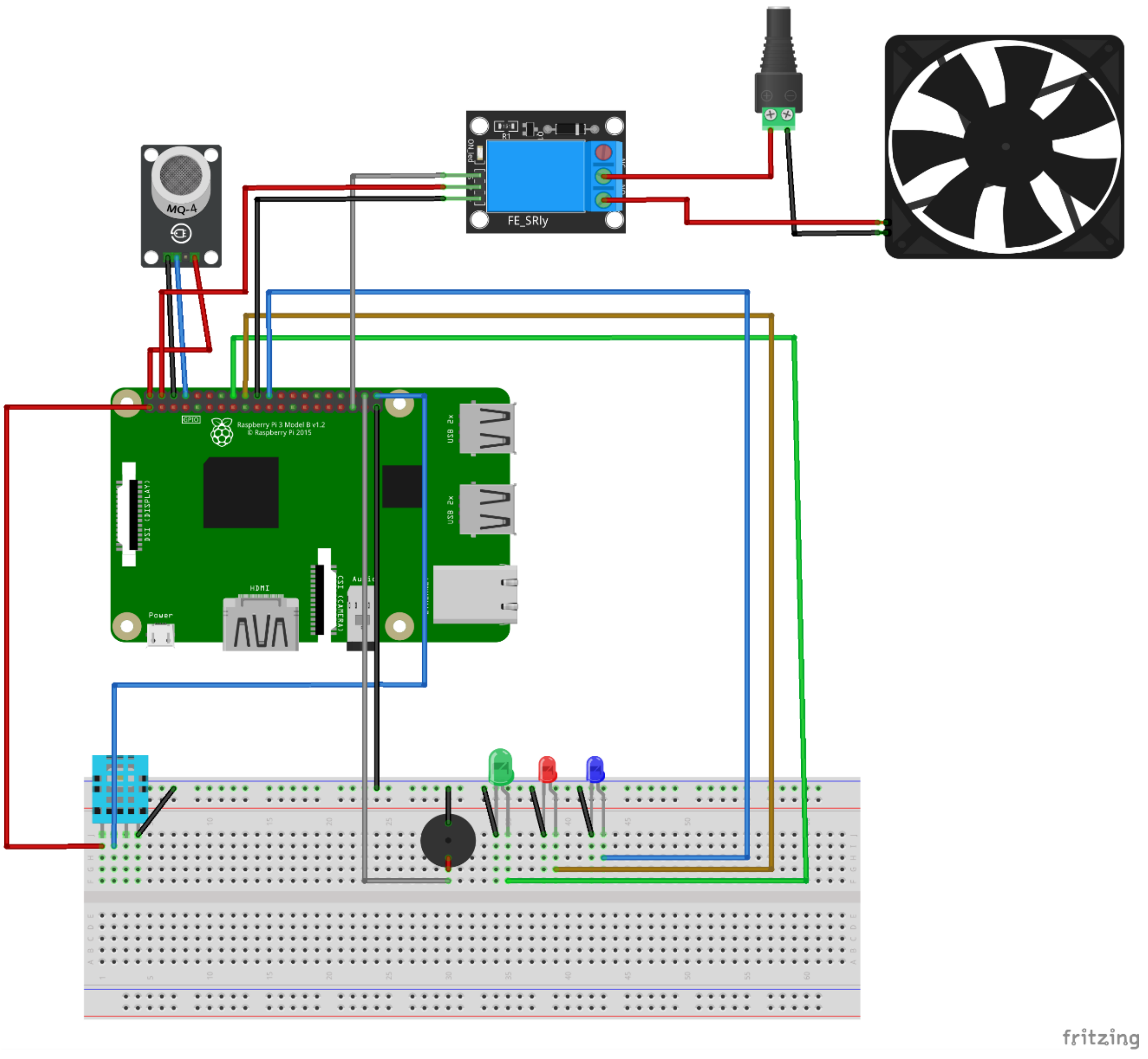


Figure 1

System architecture:

Device Layer:

- ✓ Microcontroller Unit: Processes sensor data and controls communication.
- ✓ Gas Sensors: Detect target gases.
- ✓ Temperature Sensor: Measures ambient temperature.
- ✓ Communication Module: Enables data transmission.
- ✓ Power Supply: Provides stable voltage.
- ✓ Actuators: Alert system (buzzer, LEDs).

Network Layer:

- ✓ Communication Protocol: Defines how data is transmitted between devices.
- ✓ Network Infrastructure: Provides connectivity between devices and the cloud.

Application Layer:

- ✓ Web App: Provides a user interface for real-time and historical data visualization (gas detection, temperature) as well as remote control of functionalities.

RESULTS AND DISSCUSSION

The system set up:

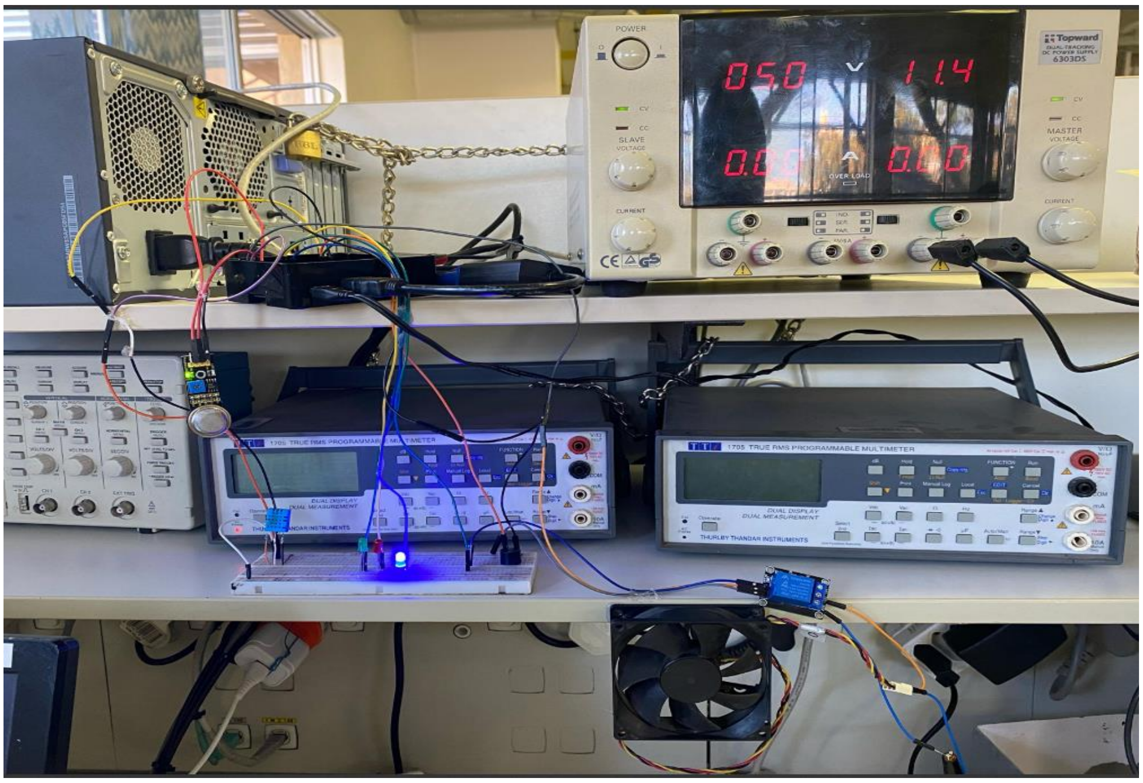


Figure 2

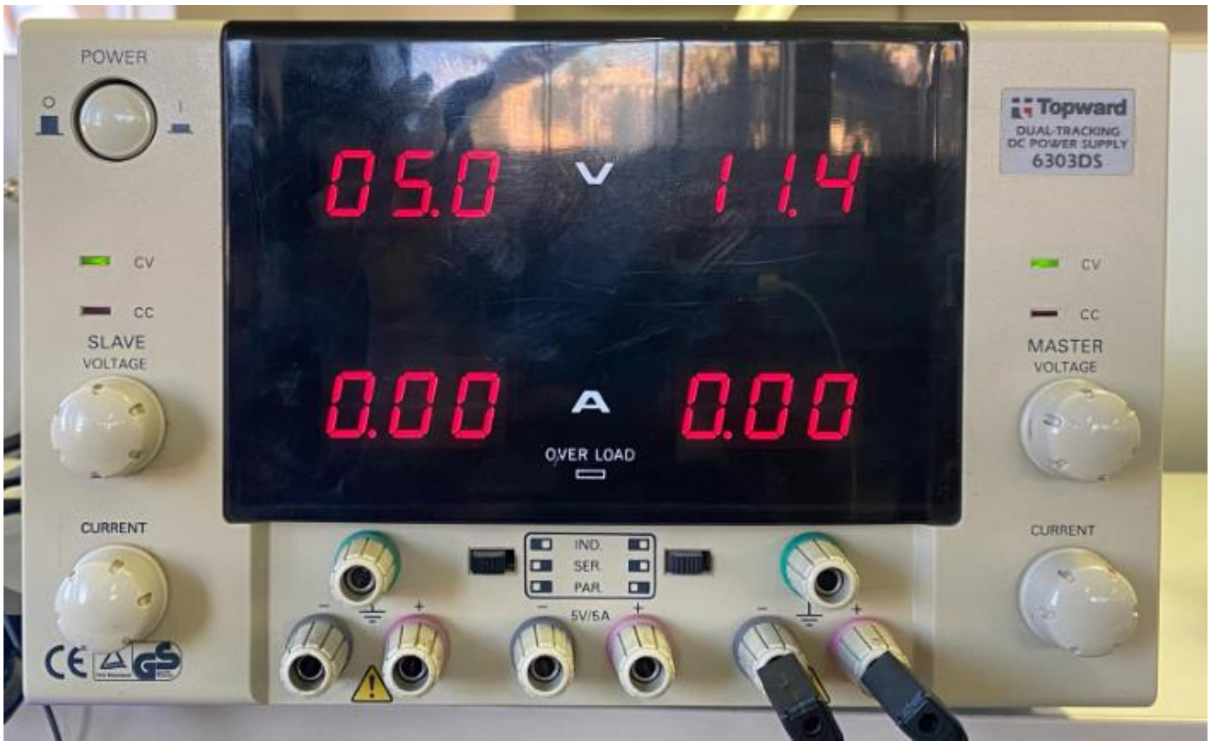


Figure 3

Figure 3 is the power supply used to supply 12V to the fan since the raspberry pi cannot supply 12 volts to the fan.

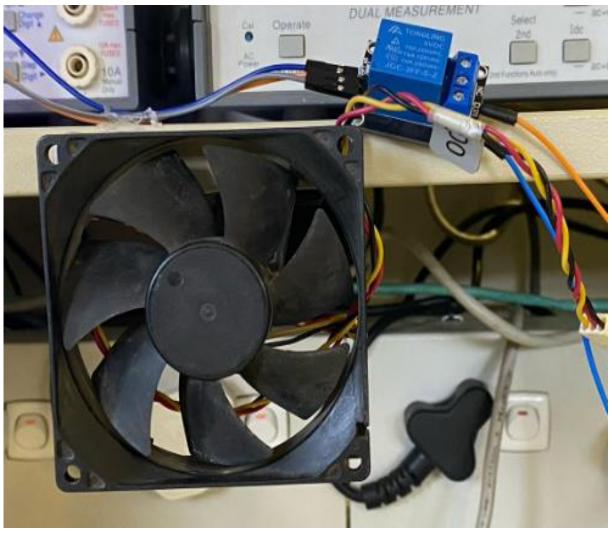


Figure 4

The 12V fan used to extract gas incase of a gas leak

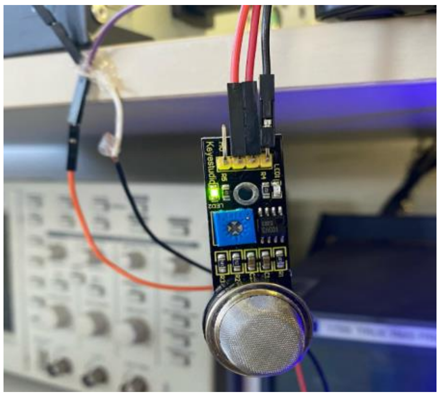
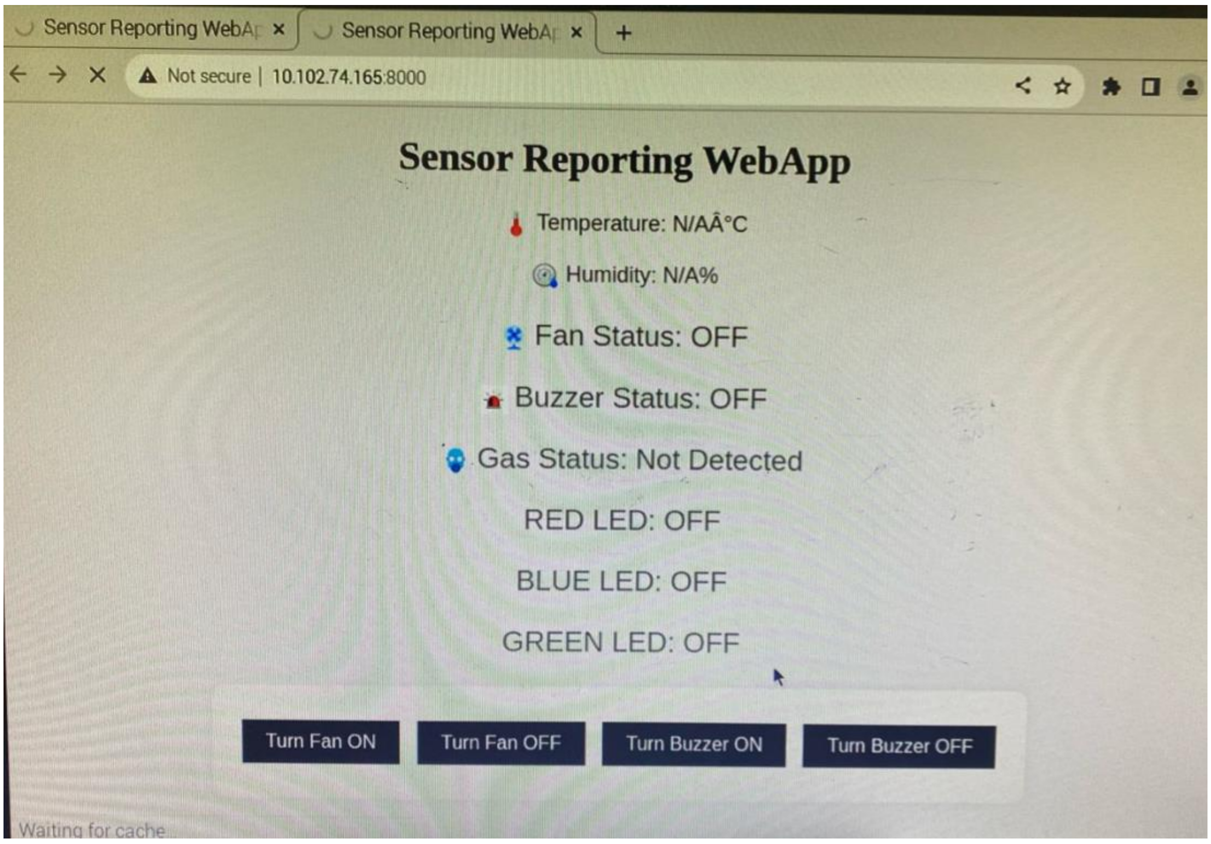


Figure 5(Gas sensor)

Web app with no data:



Sensor Reporting WebAp × | Sensor Reporting WebAp × | +

Not secure | 10.102.74.165:8000

Sensor Reporting WebApp

Temperature: N/AÂ°C

Humidity: N/A%

Fan Status: OFF

Buzzer Status: OFF

Gas Status: Not Detected

RED LED: OFF

BLUE LED: OFF

GREEN LED: OFF

Figure 6

Scenario 1:

Moderate temperatures ranging from 25 to 30 degrees.

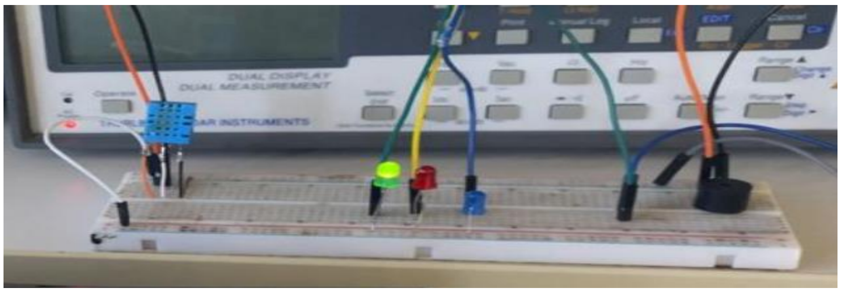


Figure 7

Green LED is expected to turn on as visible in figure 7.

Scenario 2:

Low temperatures, temperatures below 25 degrees

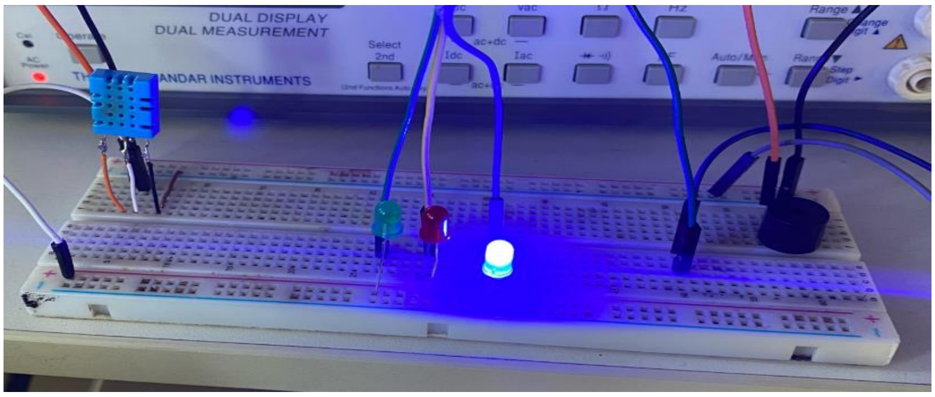
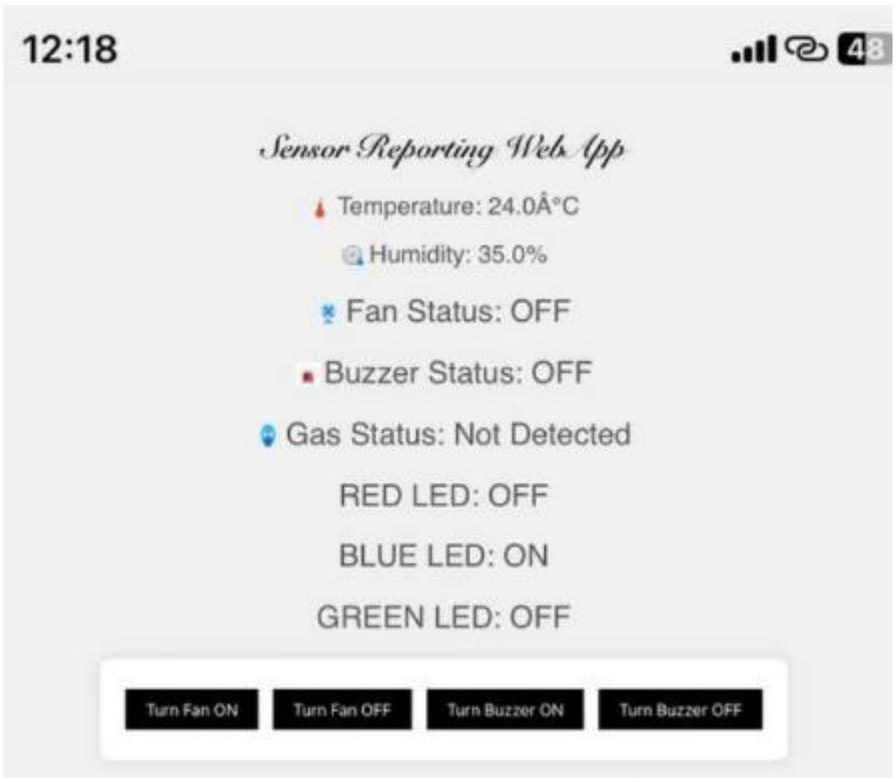


Figure 8


12:18

Sensor Reporting Web App

Temperature: 24.0Â°C

Humidity: 35.0%

Fan Status: OFF

Buzzer Status: OFF

Gas Status: Not Detected

RED LED: OFF

BLUE LED: ON

GREEN LED: OFF

Figure 9

Blue LED is expected to turn on as indicated in figure 8 and figure 9.

Scenario 3:

High temperatures, temperatures above 30 degrees or gas is detected.

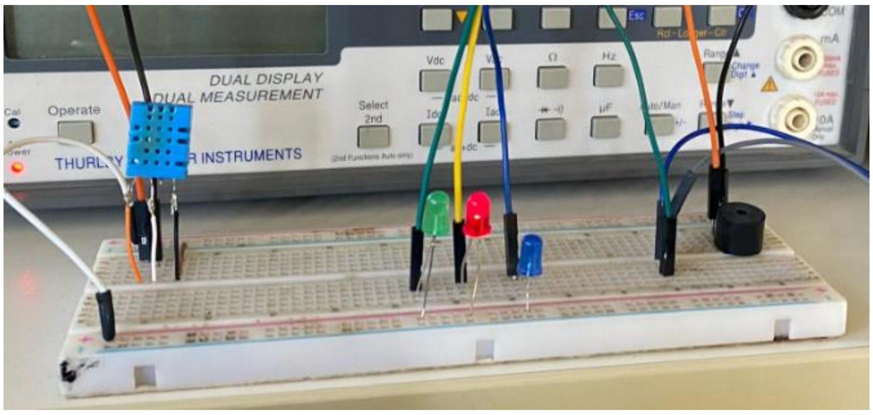
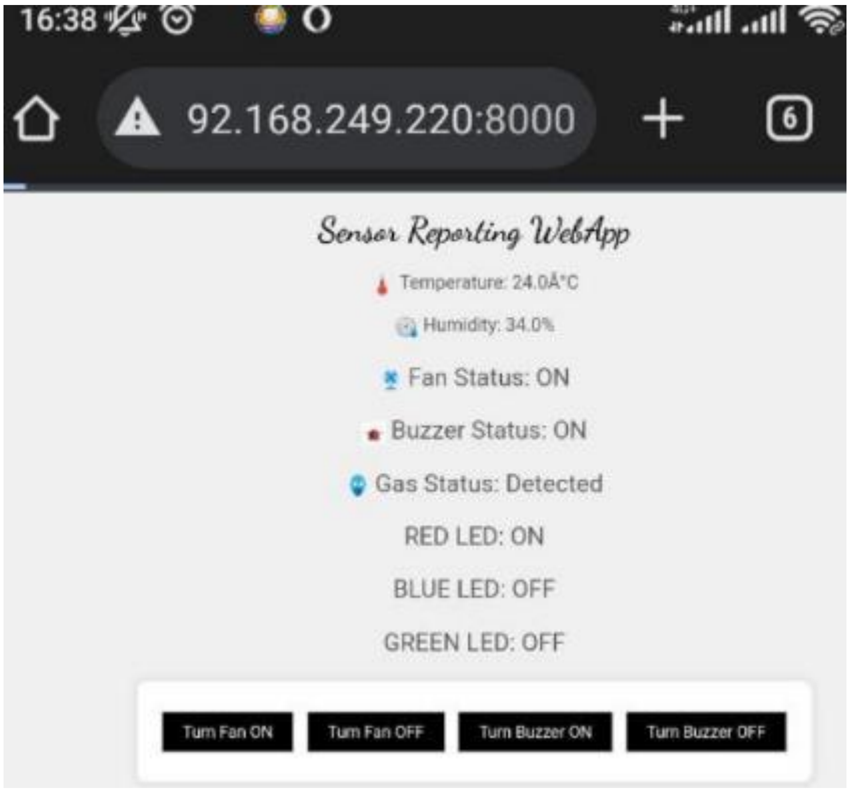


Figure 10


16:38

92.168.249.220:8000

Sensor Reporting WebApp

Temperature: 24.0Â°C

Humidity: 34.0%

Fan Status: ON

Buzzer Status: ON

Gas Status: Detected

RED LED: ON

BLUE LED: OFF

GREEN LED: OFF

Figure 11

Red Led, fan and buzzer are all expected to turn on.

The fan serves to extract the gas while the buzzer serves as an alert.

CHALLENGES FACED

Getting data from sensors DHT11 and transmitting temperature data to the web app posed to be an issue. This was because the OS of the raspberry pi was outdated and did not have libraries such as Adafruit to implement the sensor. This was resolved by simply updating the software and installing new libraries. The second challenge came from the gas sensor itself. It took time to respond to gas changes, delaying leak detection. The sensor needed calibration to ensure it was reading gas concentrations accurately. Uncalibrated sensors or delays in gas detection can be safety hazards, so these were important issues to address.

FUTURE IMPROVEMENTS

Moving forward, several enhancements can be integrated into the IoT-enabled gas detection and temperature detection system to enhance the system even more. Integration with other systems, automated shutoff valves and AI-powered analysis could lead to faster response times and even prevent emergencies. Improved communication protocols, user interfaces, and cloud storage will enhance data accessibility and usability. Standardization and robust security measures will ensure system reliability and data integrity. Finally, a focus on low-power operation will make these systems even more versatile and user-friendly. By incorporating these improvements, these systems will become even more effective in safeguarding people, property, and the environment.

CONCLUSION

An IoT enabled gas detection and temperature monitoring system was successfully developed and designed that offered alerting mechanisms and enabled remote real time monitoring of gas detection and fluctuating temperatures. Looking ahead, advancements in sensor technology, machine learning, and integration capabilities promise even greater functionalities. These improvements can lead to predictive maintenance, self-healing systems, and seamless integration with building management systems, creating a more intelligent and responsive environment.

In conclusion, IoT-enabled gas detection and temperature systems are valuable tools for various applications, from industrial settings to residential environments. As technology continues to evolve, these systems will become even more sophisticated, offering unparalleled levels of safety, efficiency, and peace of mind.

REFERENCES

Pandey, R. C., Verma, M., & Sahu, L. K. (2017, May). *Internet of Things (IOT) Based Gas Leakage Monitoring and Alerting System with MQ-2 Sensor.* From Researchgate: https://www.researchgate.net/publication/357768388_Internet_of_Things_IOT_Based_Gas_Leakage_Monitoring_and_Alerting_System_with_MQ-2_Sensor

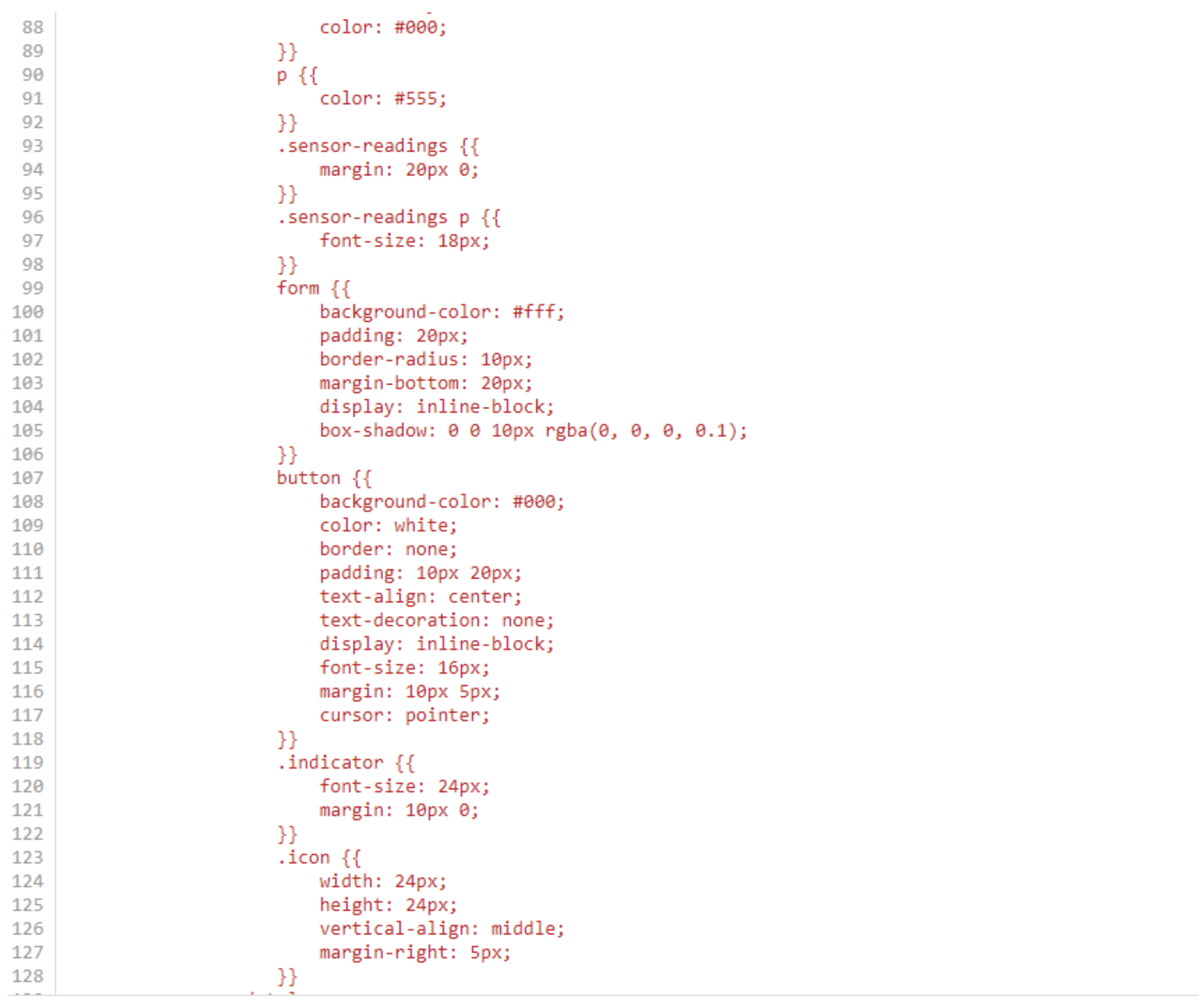
R, R. S., Reshma, M. R., Lakshmi, S. R., Vaishnavi , P., & Pavithra. (2022, June 6). *Survey on LPG Gas Monitoring System using IoT.* From IJCRT: https://www.ijcrt.org/

Sharma, V. P., Dugyala, R., Padmavathi, V., & Gurram, V. R. (2023, June 5). *Gas Leakage Detection System Using IoT And.* From esciences: https://www.e3s-conferences.org/articles/e3sconf/abs/2023/28/e3sconf_icmed-icmpc2023_01063/e3sconf_icmed-icmpc2023_01063.html

APPENDIX

Snippets of the code:

```python


import RPi.GPIO as GPIO
from http.server import BaseHTTPRequestHandler, HTTPServer
import time
from datetime import datetime
import Adafruit_DHT

host_name = '192.168.234.220'  # Replace with your Raspberry Pi's IP address
host_port = 8000

# Define GPIO pins
BUZZER_PIN = 20
RED_LED_PIN = 24
ORANGE_LED_PIN = 25
GREEN_LED_PIN = 23
FAN_PIN = 19
GAS_SENSOR_PIN = 14
DHT_SENSOR_PIN = 21

sensor = Adafruit_DHT.DHT11

def setupGPIO():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(BUZZER_PIN, GPIO.OUT)
    GPIO.setup(RED_LED_PIN, GPIO.OUT)
    GPIO.setup(ORANGE_LED_PIN, GPIO.OUT)
    GPIO.setup(GREEN_LED_PIN, GPIO.OUT)
    GPIO.setup(FAN_PIN, GPIO.OUT)
    GPIO.setup(GAS_SENSOR_PIN, GPIO.IN)

def read_sensor():
    humidity, temperature = Adafruit_DHT.read_retry(sensor, DHT_SENSOR_PIN)
    return temperature, humidity

class MyServer(BaseHTTPRequestHandler):

    def do_HEAD(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
    def _redirect(self, path):
        self.send_response(303)
        self.send_header('Content-type', 'text/html')
        self.send_header('Location', path)
        self.end_headers()

    def do_GET(self):
        gas_detected = GPIO.input(GAS_SENSOR_PIN) == GPIO.LOW
        temperature, humidity = read_sensor()

        # Control LEDs and fan based on temperature
        if gas_detected or (temperature is not None and temperature > 30):
            GPIO.output(FAN_PIN, GPIO.HIGH)  # Fan on
            GPIO.output(BUZZER_PIN, GPIO.HIGH)
            GPIO.output(RED_LED_PIN, GPIO.HIGH)  # Red LED on
            GPIO.output(ORANGE_LED_PIN, GPIO.LOW)  # Orange LED off
            GPIO.output(GREEN_LED_PIN, GPIO.LOW)  # Green LED off
        elif temperature is not None and temperature < 25:
            GPIO.output(FAN_PIN, GPIO.LOW)  # Fan off
            GPIO.output(BUZZER_PIN, GPIO.LOW)
            GPIO.output(RED_LED_PIN, GPIO.LOW)  # Red LED off
            GPIO.output(ORANGE_LED_PIN, GPIO.HIGH)  # Orange LED on
            GPIO.output(GREEN_LED_PIN, GPIO.LOW)  # Green LED off
        elif temperature is not None and 25 <= temperature <= 29:
            GPIO.output(FAN_PIN, GPIO.LOW)  # Fan off
            GPIO.output(BUZZER_PIN, GPIO.LOW)
            GPIO.output(RED_LED_PIN, GPIO.LOW)  # Red LED off
            GPIO.output(ORANGE_LED_PIN, GPIO.LOW)  # Orange LED off
            GPIO.output(GREEN_LED_PIN, GPIO.HIGH)  # Green LED on

        html = f'''
            <!DOCTYPE html>
            <html>
            <head>
                <title>Sensor Reporting WebApp</title>
                <style>
                    body {{
                        font-family: 'Open Sans', sans-serif;
                        text-align: center;
                        background-color: #f0f0f0;
                        color: #333;
                    }}
                    h1 {{
                        font-family: 'Lobster', cursive;
                        color: #000;
```



```
                        color: #000;
                    }}
                    p {{
                        color: #555;
                    }}
                    .sensor-readings {{
                        margin: 20px 0;
                    }}
                    .sensor-readings p {{
                        font-size: 18px;
                    }}
                    form {{
                        background-color: #fff;
                        padding: 20px;
                        border-radius: 10px;
                        margin-bottom: 20px;
                        display: inline-block;
                        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
                    }}
                    button {{
                        background-color: #000;
                        color: white;
                        border: none;
                        padding: 10px 20px;
                        text-align: center;
                        text-decoration: none;
                        display: inline-block;
                        font-size: 16px;
                        margin: 10px 5px;
                        cursor: pointer;
                    }}
                    .indicator {{
                        font-size: 24px;
                        margin: 10px 0;
                    }}
                    .icon {{
                        width: 24px;
                        height: 24px;
                        vertical-align: middle;
                        margin-right: 5px;
                    }}
```

```
                </style>
                <meta http-equiv="refresh" content="2">
            </head>
            <body>
            <h1>Sensor Reporting WebApp</h1>
            <div class="sensor-readings">
                <p><img src="https://img.icons8.com/color/48/000000/temperature.png" class="icon">Temperature: {temperature if temperatur
is not None else 'N/A'}°C</p>
                <p><img src="https://img.icons8.com/fluency/48/000000/humidity.png" class="icon">Humidity: {humidity if humidity is not
None else 'N/A'}%</p>
            </div>
            <div class="indicator">
                <p><img src="https://img.icons8.com/color/48/000000/fan.png" class="icon">Fan Status: {'ON' if GPIO.input(FAN_PIN) else
'OFF'}</p>
            </div>
            <div class="indicator">
                <p> <img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ftse1.mm.bing.net%2Fth%3Fid%3DOIP.SEmjkhbiyHb8tp
7imtx1AHaHa%26pid%3DApi&f=1&ipt=2e1a969065c9520b9e1b9b8865a3d3843043d97ca61d1154716f60bc4d4510e9&ipo=images" class="icon">Buzzer Status:
{'ON' if GPIO.input(BUZZER_PIN) else 'OFF'}</p>
            </div>

            <div class="indicator">
                <p><img src="https://img.icons8.com/fluency/48/000000/gas-mask.png" class="icon">Gas Status: {'Detected' if gas_detected
else 'Not Detected'}</p>
            </div>
            <div class="indicator">
                <p>RED LED: {'ON' if GPIO.input(RED_LED_PIN) else 'OFF'}</p>
                <p>BLUE LED: {'ON' if GPIO.input(ORANGE_LED_PIN) else 'OFF'}</p>
                <p>GREEN LED: {'ON' if GPIO.input(GREEN_LED_PIN) else 'OFF'}</p>
            </div>
            <form action="/" method="POST">
                <button name="control" value="fan_on">Turn Fan ON</button>
                <button name="control" value="fan_off">Turn Fan OFF</button>
                <button name="control" value="buzzer_on">Turn Buzzer ON</button>
                <button name="control" value="buzzer_off">Turn Buzzer OFF</button>
            </form>
            </body>
            </html>'''

        self.do_HEAD()
        self.wfile.write(html.encode("utf-8"))

    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        post_data = self.rfile.read(content_length).decode("utf-8")
        post_data = post_data.split("=")[1]

        if post_data == 'fan_on':
            GPIO.output(FAN_PIN, GPIO.HIGH)  # Turn on the Fan
        elif post_data == 'fan_off':
            GPIO.output(FAN_PIN, GPIO.LOW)  # Turn off the Fan
        elif post_data == 'buzzer_on':
            GPIO.output(BUZZER_PIN, GPIO.HIGH)  # Turn on the Buzzer
        elif post_data == 'buzzer_off':
            GPIO.output(BUZZER_PIN, GPIO.LOW)  # Turn off the Buzzer

        # Redirect back to the main page after handling the POST request
        self._redirect('/')

def run(server_class=HTTPServer, handler_class=MyServer, port=host_port):
    setupGPIO()
    server_address = (host_name, port)
    httpd = server_class(server_address, handler_class)
    print(f"Starting server on http://{host_name}:{port}")
    httpd.serve_forever()

if __name__ == '__main__':
    setupGPIO()
    run()
```