LAUTECH Journal of Engineering and Technology 19 (1) 2025: 154-163
10.36108/laujet/5202.91.0141

Development of IoT-based Smart Gas Leakage Detection with Alert

[1*]Shobowale Y. I., [2]Lawal A. I., [3]Olasehinde O. O., [4]Sobowale O. O. and [5]Oluwaranti A. I.

[1]*Department of Information and Communication Engineering, Elizade University, Ilara-Mokin, Nigeria.*
[2,3,5]*Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.*
[4]*Department of Computer Engineering, Federal University, Oye-Ekiti, Nigeria.*

[1]*yusuf.shobowale@elizadeuniversity.edu.ng*,[2]*arlawal@oauife.edu.ng*, [3]*olasehinde03@gmail.com*
[4]*sobowaleadedayo@gmail.com*, [5]*nirano@gmail.com*

Article Info

Article history:

Received: Jan. 08, 2025
Revised: Feb. 06, 2025
Accepted: Feb. 15, 2025

Keywords:

IoT,
Gas Leakage,
Alert System

Corresponding Author:
yusuf.shobowale@elizadeuniversity.edu
+2348131657658

ABSTRACT

Gas leaks present significant and persistent danger, continually threatening lives and properties. Traditional gas leak detection methods often depend on manual inspections and assessments, and they are inherently vulnerable to various shortcomings. This work addressed the challenge of gas leaks by creating an IoT-based gas leakage detection and smart alerting system. This automation, remote and real-time solution, provides significant benefits compared to traditional gas detection methods, greatly enhancing safety and reducing the likelihood of gas-related accidents. The sensor integration phase focuses on connecting the MQ-135 gas sensor to the Arduino Uno microcontroller and calibrating the sensor to ensure accurate readings. Code is implemented on the Arduino Uno to read sensor data from the MQ-135 sensor and convert it into gas concentration values. This approach significantly enhances environmental safety by identifying hazardous gas leaks. The system presented triggers alarm as well as displaying gas leak results on the think speak at a level of detection above 200 being the threshold value. The availability of such a system provided automation and eliminated the need for manual sensing of gas leaks via nasal perception or guesswork

INTRODUCTION

Gas detection has a rich historical context that spans ancient civilizations to modern technological advancements. In ancient times, people recognized the presence of hazardous gases through their distinct odours or effects on living organisms, such as the suffocating properties of carbon dioxide in confined spaces. Pioneers like John Tyndall introduced the first practical gas detector, which utilized the principle of light absorption to detect gases like methane (Kodali *et al.*, 2018). Subsequent milestones in gas detection technology emerged in the late 19th and early 20th centuries. Electrochemical sensors, invented in the 1920s, revolutionized the detection of toxic gases such as carbon monoxide and hydrogen sulfide These sensors marked a significant leap forward in accuracy and reliability. Catalytic bead sensors and infrared absorption spectroscopy further refined gas detection capabilities (Kodali *et al.*, 2018).

These technologies allowed for more precise measurements and expanded the range of detectable gases. Traditional sensors may not be sensitive enough to detect certain gases, leading to delayed or inaccurate readings that can put workers' safety at risk (Adenubi *et al.*, 2023). Another significant drawback of conventional gas detection systems is their reliance on periodic manual monitoring (Khan, 2020). The emergence of the Internet of Things (IoT) has ushered in a new era in gas detection, offering novel solutions that address many of the challenges of traditional systems. IoT characterized by interconnected devices and sensors capable of collecting and exchanging data, has revolutionized various industries, including gas detection (Balaji *et al.*, 2019). IoT-enabled gas detection systems leverage the power of connectivity to provide real-time monitoring and remote management capabilities (Hasibuan and Idris, 2019). Moreover, IoT-based gas detection systems offer advantages in terms of data collection and analysis (Goyal *et al.*, 2021). The wealth of data generated by connected sensors can be aggregated and analyzed to gain valuable insights into gas concentrations, trends, and patterns over time. This data-driven approach facilitates proactive maintenance, decision-making, and risk assessment, leading to more efficient and effective gas detection strategies.

IoT technology enables continuous monitoring of gas levels, allowing for real-time detection of hazardous gases (Rath *et al.*, 2024). The emergence of IoT in gas detection represents a paradigm shift in how safety and risk management are approached (Balaji *et al.*, 2019). By harnessing the power of connectivity and data analytics, IoT-based solutions offer unparalleled capabilities in detecting and mitigating the risks associated with hazardous gases, ultimately contributing to safer and more secure environments (Hasibuan and Idris, 2019). One of the primary benefits of IoT-enabled gas detection systems is the ability to provide real-time monitoring and response (Goyal *et al.*, 2021). By deploying connected sensors throughout facilities or environments at risk of gas leaks, organizations can continuously monitor gas levels and receive immediate alerts in the event of abnormal readings or hazardous conditions (Balaji *et al.*, 2019). This real-time monitoring capability enables swift response actions, such as evacuations or shutdown procedures, to minimize the potential impact of gas-related incidents. Semiconductor sensors, operating on the change in conductivity of semiconductor materials, present a cost-effective and compact solution (Adekitan *et al.*, 2018).

Electrochemical sensors are commonly used for detecting toxic gases like carbon monoxide and hydrogen sulfide (Khan *et al.*, 2019). Catalytic sensors are, however, used for detecting flammable gases like methane and propane (Baranov and Osipova, 2022). Infrared sensors, on the other hand, are designed to detect gases based on their absorption of infrared radiation.

Furthermore, IoT technology enhances the effectiveness of gas detection systems through advanced data analytics and predictive capabilities (Goyal *et al.*, 2021). The vast amounts of data collected by connected sensors can be analyzed to identify trends, patterns, and anomalies in gas concentrations over time (Hasibuan and Idris, 2019). This data-driven approach enables proactive maintenance, early detection of potential issues, and predictive modeling of future scenarios, empowering organizations to implement preventive measures and optimize safety protocols.

Overall, the significance of IoT in enhancing gas safety lies in its ability to provide real-time monitoring, advanced analytics, and seamless integration with existing systems (Balaji *et al.*, 2019). By harnessing the transformative power of IoT technology, gas leak detection and prevention systems can overcome the limitations of existing approaches. There is a critical need for smarter and more extensive gas leak detection that offers continuous real-time monitoring, high-sensitivity detection, automated multi-tiered alerts, remote access and control, and easy scalability to adapt to various environments and evolving needs (Jumaa *et al.*, 2022; Varma *et al.*, 2017). Seamless integration of sensors, data analytics, and cloud-based platforms provides a more effective and comprehensive solution for gas leak detection, enabling proactive intervention and mitigating risks before they escalate (Debnath *et al.*, 2020).

MATERIALS AND METHODS

The IoT-based gas leakage detection system was designed with a modular and interconnected architecture to ensure seamless integration and efficient functionality. At its core, the system comprises several key modules, including the sensor module, microcontroller module, communication module, alerting module, and power supply module as depicted in Figure 1.

System Design

The core of the system revolves around the Arduino UNO microcontroller, which acts as the central processing unit and interface for units of the system. The system modules in the architecture were designed to ensure seamless data flow and effective communication between hardware and software components. The sensor module incorporates the MQ-135 gas sensor, responsible for detecting hazardous gases such as methane, carbon monoxide, and Volatile Organic Compounds (VOCs) in the environment. This sensor interfaces with the microcontroller module, which utilizes the Arduino Uno microcontroller for data processing, control, and interfacing with other system components.

Facilitating wireless communication is the communication module, which integrates the ESP8266 Wi-Fi module. This module establishes a wireless connection to the ThingSpeak platform, enabling remote monitoring and management of gas levels. The alerting module consists of LED indicators and a buzzer, providing visual and audible alerts in case of gas leakages. Powering the system is achieved through the power supply module, which provides the necessary electrical power to operate the system.

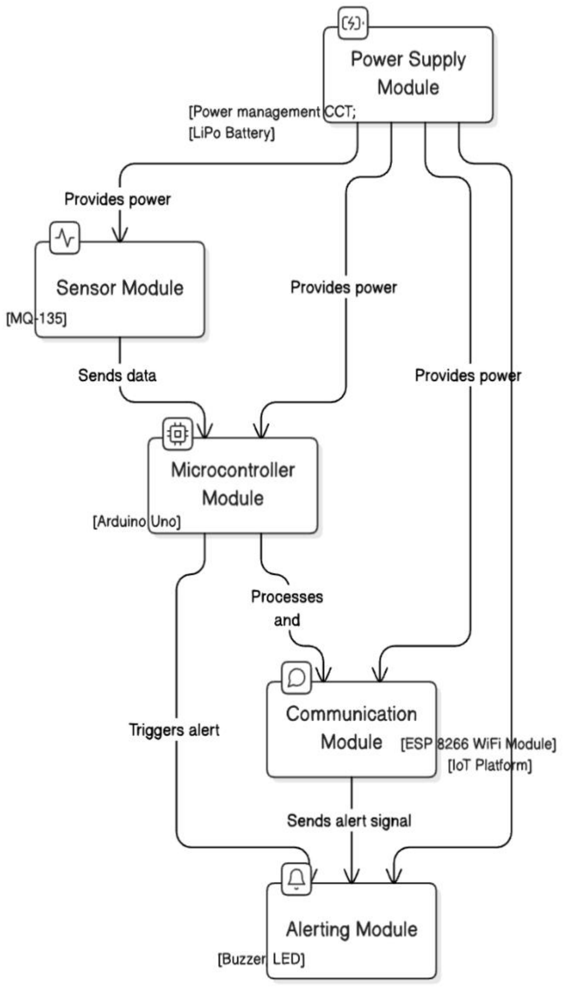


Figure 1: System Architecture and Modules

The Arduino Uno operates at a voltage of 5V and draws a current of approximately 50mA, while the MQ-135 gas sensor operates within a voltage range of 5V-24V and draws a current of around 150mA during operation. The ESP8266 Wi-Fi module operates at a voltage of 3.3V and draws a current of approximately 70mA during data transmission. LED indicators and the buzzer operate at a voltage of 5V and draw currents of approximately 20mA and 30mA, respectively.

The system employed programming logic and algorithms implemented on the Arduino Uno microcontroller. This included data acquisition from the gas sensor, data processing to detect gas leakages, alert generation through LED indicators and the buzzer, wireless communication with the

ThingSpeak platform, and error-handling mechanisms to ensure system reliability.

To ensure the durability and protection of internal components, the system was housed within a protective enclosure. The enclosure design allows for easy access to components for maintenance and servicing while providing adequate ventilation for heat dissipation. Overall, the system design incorporates hardware and software components working in concert to achieve reliable gas monitoring and alerting capabilities, ensuring ease of use and safety for users. Figure 2. Shows the flowchart of the system in effectively detecting gas leaks and notifying users in real-time while the circuit diagram is shown in Figure 3.

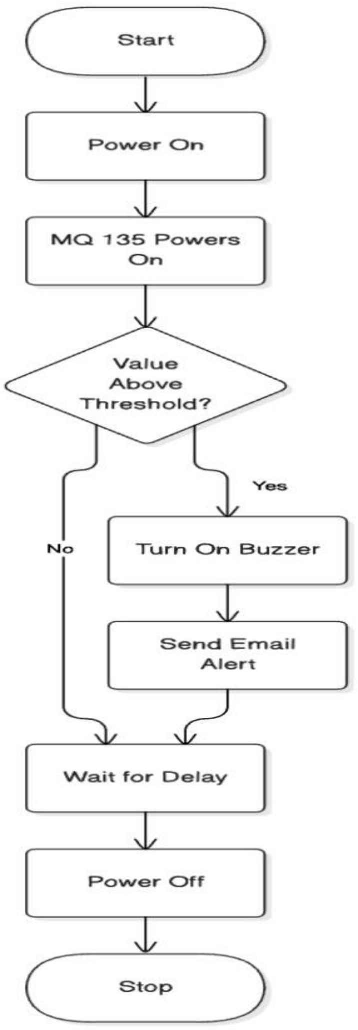


Figure 2: Flowchart of the IoT-Based Gas Leakage Detection System

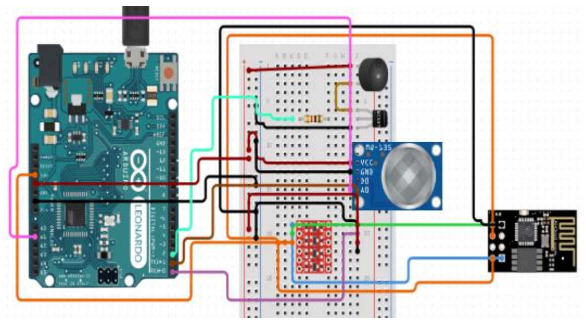


Figure 3 Circuit Diagram of the System

System Implementation

The implementation of the IoT-based Gas Leakage Detection System involves assembling, configuring, and testing the hardware and software components as depicted in Figure 4.

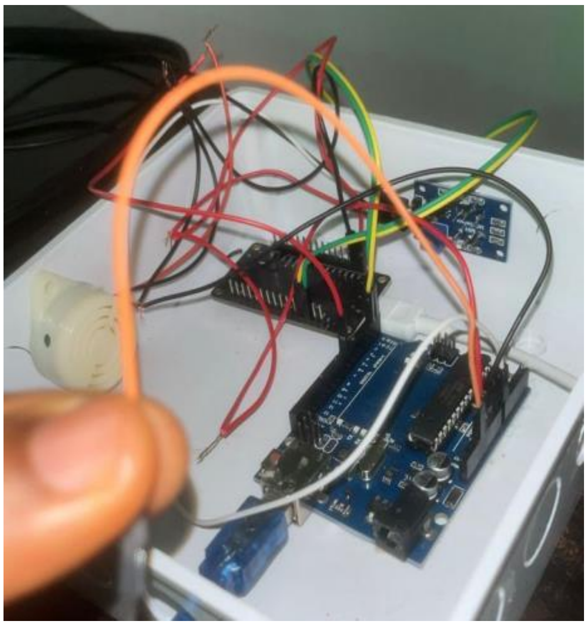
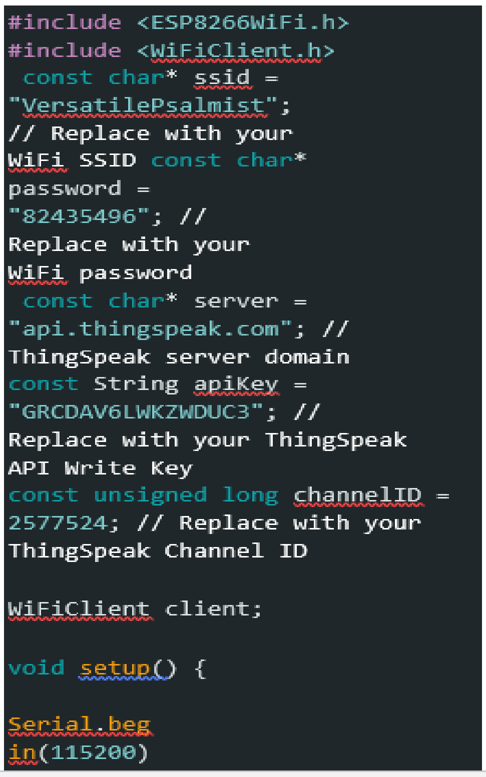


Figure 4: Assembling of components.

The alert mechanism is implemented by connecting LED indicators and a buzzer to the Arduino Uno microcontroller and implementing code to activate these components upon detecting a gas leakage based on predefined thresholds. The communication setup involves the connection of the ESP8266 Wi-Fi module to the Arduino Uno microcontroller and configuring the module to establish a Wi-Fi connection to the local network and access the ThingSpeak platform. Code is then implemented on the Arduino Uno to transmit sensor data to the

ThingSpeak platform using the ESP8266 module. The Arduino Uno firmware is configured to transmit sensor data to the ThingSpeak channel using appropriate API endpoints. Figure 5 represents a typical section of codes used for data receiver and transmission to the ThingSpeak.



```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
 const char* ssid =
"VersatilePsalmist";
// Replace with your
WiFi SSID const char*
password =
"82435496"; //
Replace with your
WiFi password
 const char* server =
"api.thingspeak.com"; //
ThingSpeak server domain
const String apiKey =
"GRCDAV6LWKZWDUC3"; //
Replace with your ThingSpeak
API Write Key
const unsigned long channelID =
2577524; // Replace with your
ThingSpeak Channel ID

WiFiClient client;

void setup() {

Serial.beg
in(115200)
```

Figure 5: ESP Code for Data Receival and Transmittal to ThingSpeak

Email notification setup is implemented by coding the Arduino Uno to send email notifications to the user's designated email address in case of gas leakage. Configuration of email notification settings, including SMTP server details and authentication credentials, is completed, followed by testing to ensure prompt and accurate alerts. System testing and validation involve comprehensive testing of the entire system to verify functionality, reliability, and accuracy.

RESULTS

The testing phase of the IoT-based Gas Leakage Monitoring System was conducted to validate the functionality, accuracy, and reliability of the components involved. This phase comprised three primary tests: Gas Concentration Simulation, Wi-Fi Connectivity, and Alert Mechanism Verification. Each test aimed to ensure that the system performs as expected in real-world scenarios, effectively detecting gas levels and transmitting data to users.

Gas Concentration Simulation Testing

The gas concentration simulation testing was conducted to evaluate the accuracy and responsiveness of the MQ-135 gas sensor in detecting various levels of gas concentration, specifically using cooking gas as the test substance. The test aimed at ensuring the reliable measurement of air quality and providing accurate data to the Arduino for further processing and communication with the ESP8266 module.

In the initial setup, the MQ-135 gas sensor was connected to the Arduino, with the analog output (AO) pin of the sensor linked to the A0 pin on the Arduino. The Arduino was connected to a computer running the Arduino IDE, allowing for real-time monitoring of sensor readings via the serial monitor. To establish a baseline reading, the sensor was exposed to a clean-air environment. This involved allowing the sensor to warm up and stabilize, ensuring that the initial readings represented normal air quality conditions.

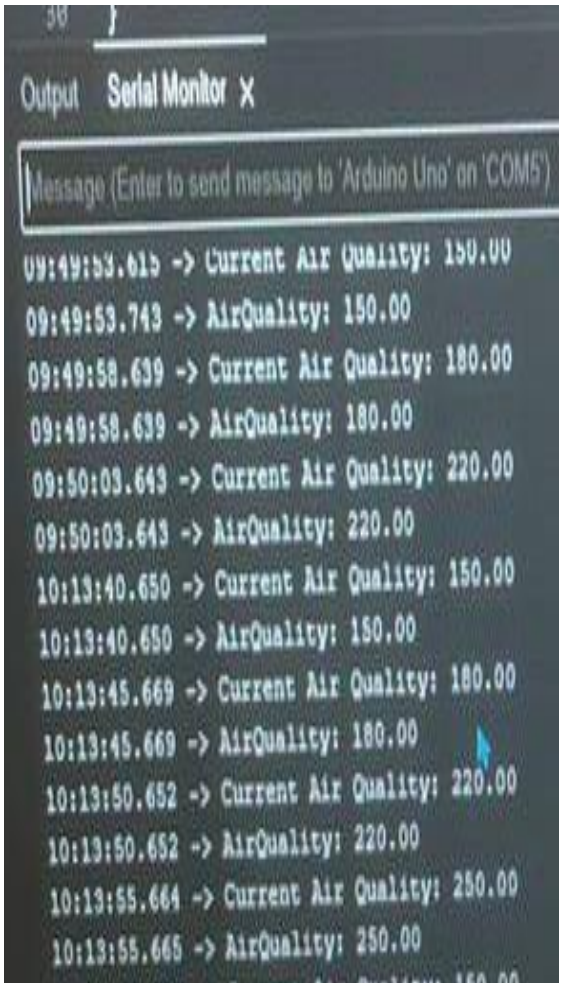
Each concentration level was maintained for a specific duration to observe the sensor's response. The analog readings from the MQ-135 sensor were continuously monitored on the Arduino IDE's serial monitor. The Arduino code processed these readings, converting them into voltage and subsequently into air quality values. These values were displayed on the serial monitor, providing real-time feedback on the gas concentration levels detected by the sensor.

The sensor's responsiveness to changes in gas concentration was assessed by observing the fluctuations in the air quality readings on the serial monitor. The system's ability to return to baseline readings after the removal of the gas source was also evaluated. The MQ-135 sensor successfully detected various levels of gas concentration, with the corresponding air quality values accurately reflected on the serial monitor. The sensor demonstrated a high degree of sensitivity, with noticeable changes in readings correlating to the presence and removal of gas sources. The system's ability to return to baseline readings after the gas source was removed indicated that the sensor could reliably reset and provide accurate measurements in fluctuating environmental conditions.

The gas concentration simulation test results, as shown in Figure 5, confirmed that the MQ-135 gas sensor is capable of accurately detecting changes in gas concentration levels. The real-time data displayed on the Arduino IDE's serial monitor provided clear evidence of the sensor's responsiveness and precision. This test validated the sensor's functionality as a crucial component of the "IoT Based Gas Leakage Monitoring System," ensuring reliable air quality monitoring and data processing for further transmission to the ESP8266 module.

WiFi Connectivity

The Wi-Fi connectivity testing was carried out to ensure that the ESP8266 module could establish a stable connection to the local Wi-Fi network and reliably transmit air quality data to the ThingSpeak platform. This test was crucial to validate the system's ability to send real-time gas concentration readings to the cloud for remote monitoring and data analysis. In this test, the ESP8266 module was configured to connect to the local Wi-Fi network using the specified Service Set Identifier (SSID) and password.



```
09:49:53.615 -> Current Air Quality: 150.00
09:49:53.743 -> AirQuality: 150.00
09:49:58.639 -> Current Air Quality: 180.00
09:49:58.639 -> AirQuality: 180.00
09:50:03.643 -> Current Air Quality: 220.00
09:50:03.643 -> AirQuality: 220.00
10:13:40.650 -> Current Air Quality: 150.00
10:13:40.650 -> AirQuality: 150.00
10:13:45.669 -> Current Air Quality: 180.00
10:13:45.669 -> AirQuality: 180.00
10:13:50.652 -> Current Air Quality: 220.00
10:13:50.652 -> AirQuality: 220.00
10:13:55.664 -> Current Air Quality: 250.00
10:13:55.665 -> AirQuality: 250.00
```

Figure 5 Gas Simulation Test Results

The Arduino was programmed to read the air quality values from the MQ-135 sensor, process the data, and send it to the ESP8266 via serial communication. The ESP8266 then sent these values to the ThingSpeak platform using HTTP POST requests. Initially, the Wi-Fi connection was verified by checking the status of the ESP8266 module. The module was powered up and connected to the Arduino, with its TX and RX pins properly interfaced through voltage dividers to ensure safe communication levels. The serial monitor on the Arduino IDE displayed the connection status, including attempts to connect to the specified Wi-Fi network. The successful connection was confirmed when the module obtained an IP address from the router, which was also printed on the serial monitor.

The Arduino continuously read the air quality values from the MQ-135 sensor and sent the readings to the ESP8266 over serial communication. The ESP8266, in turn, formatted the data into an HTTP POST request and sent it to the ThingSpeak platform. Each successful data transmission was acknowledged by a response from ThingSpeak, which was also printed on the serial monitor.

To further validate the data transmission, the ThingSpeak channel associated with the project was monitored. The real-time air quality values sent by the ESP8266 module were displayed on the ThingSpeak dashboard, providing a visual representation of the gas concentration levels detected by the MQ-135 sensor. Figure 6 shows the graphical representation of the different gas values with a threshold being 200.

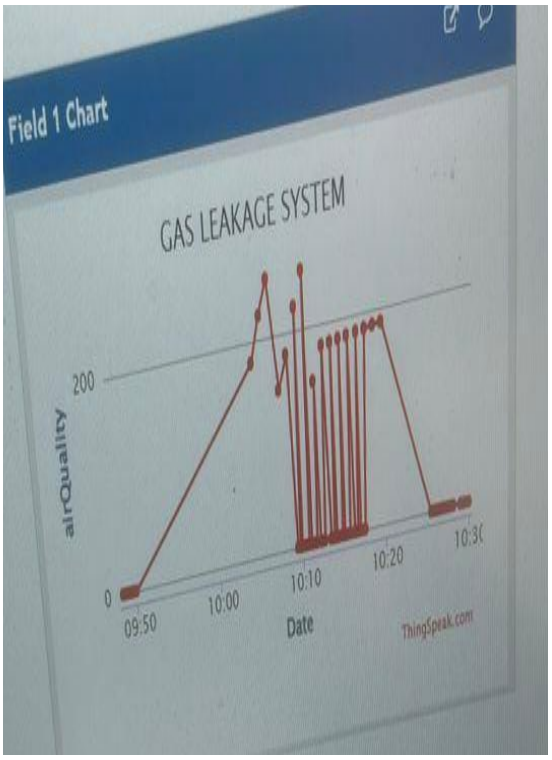


Figure 6: Wi-Fi Connectivity Test Results

The stability of the Wi-Fi connection was tested by continuously running the system for an extended period. The ESP8266 maintained a stable connection to the Wi-Fi network, consistently sending data to ThingSpeak without interruption. The system's ability to recover from temporary Wi-Fi outages was also tested by manually disconnecting and reconnecting the Wi-Fi network. The ESP8266 successfully re-established the connection and resumed data transmission, demonstrating its reliability in varying network conditions.

The Wi-Fi connectivity testing confirmed the reliable connection of the ESP8266 module to the local Wi-Fi network and transmitted air quality data to the ThingSpeak platform. The successful real-time data display on the ThingSpeak dashboard validated the system's functionality for remote monitoring and data analysis. This test ensured that the "IoT Based Gas Leakage Monitoring System" could effectively communicate critical gas concentration readings to the cloud for real-time monitoring and alerts.

Alert Mechanism

The Alert Mechanism Testing was conducted to ensure that the system could effectively notify users when gas concentration levels exceeded a predefined threshold. This test aimed to validate the functionality of the buzzer for immediate audible alerts and the ability of the ESP8266 module to send email notifications via the ThingSpeak platform using the SendGrid API.

Buzzer Activation

The first part of the alert mechanism involved testing the buzzer connected to the Arduino. The Arduino was programmed to monitor the air quality values read from the MQ-135 sensor continuously. If the air quality value exceeds a specified threshold (in this case, 200), the Arduino would trigger the buzzer to sound an alert. To simulate a high gas concentration scenario, cooking gas was released near the MQ-135 sensor. As the gas concentration increased, the sensor's analog output changed, and the Arduino detected the higher values. Once the air quality value surpassed the threshold, the Arduino set the digital pin connected to the buzzer to HIGH, activating the buzzer and providing an immediate audible warning.

Email Notification

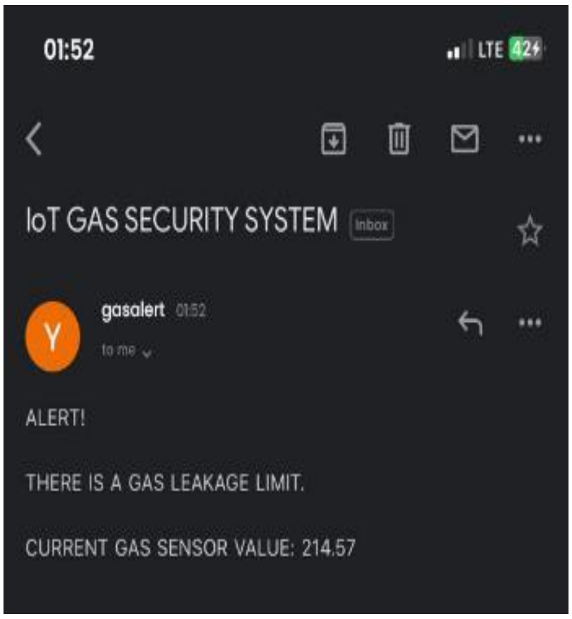
The second part of the alert mechanism involved testing the system's capability to send email notifications when hazardous gas levels were detected. The ESP8266 module was programmed to send data to the ThingSpeak platform, which was configured to trigger email alerts using the SendGrid API.

When the air quality value exceeded the threshold, the ESP8266 sent the data to the ThingSpeak channel. A MATLAB analysis script running on the ThingSpeak platform continuously monitored the channel data. When the gas concentration values were higher than the threshold, the script triggered an email notification via the SendGrid API.

Upon crossing the threshold, the MATLAB script on ThingSpeak was executed, and an email alert was sent to the designated recipient as shown in Figure 7.

The email contained relevant information, including the gas concentration value and a timestamp, providing immediate notification of the hazardous condition. This real-time alert mechanism ensured that users were promptly informed of potential gas leaks, allowing them to take necessary actions.

The alert mechanism testing demonstrated the system's capability to provide immediate and effective alerts in the event of hazardous gas concentration levels. The buzzer provided a direct and audible warning, while the email notifications ensured remote users were informed in real time.


01:52

IoT GAS SECURITY SYSTEM Inbox

gasalert 01:52
to me

ALERT!

THERE IS A GAS LEAKAGE LIMIT.

CURRENT GAS SENSOR VALUE: 214.57

Figure 7 ESP Code for Data Receival and Transmittal to ThingSpeak

The successful activation of the buzzer and the receipt of email alerts confirmed the reliability and effectiveness of the alert mechanisms implemented in the "IoT Based Gas Leakage Monitoring System." This comprehensive alert system enhances safety by providing timely warnings to prevent potential gas-related accidents.

DISCUSSION

System Evaluation

The IoT-based Gas Leakage Monitoring System was evaluated to ensure its functionality, reliability, and effectiveness in detecting gas leaks and providing timely alerts. The evaluation criteria used in the system are as presented:

Air Quality Calculation: The voltage is used to estimate the air quality level through the formula float airQuality = voltage * 100;. This scaling factor may need adjustment based on calibration and specific application requirements.

Serial Output: The air quality value is sent to the serial monitor for real-time monitoring using

Serial.println (airQuality); This allows users to observe the air quality readings directly.

Buzzer Activation: The code checks if the air quality exceeds a predefined threshold (200) using the conditional statement if (airQuality > 200). If this condition is met, the buzzer is activated with digitalWrite(buzzerPin, HIGH);, providing an audible alert. If the air quality is within safe levels, the buzzer is turned off using digitalWrite(buzzerPin, LOW);.

Delay: The loop concludes with a delay (5000); command, which introduces a pause of 5 seconds before the next sensor reading is taken. This delay helps manage resource usage and avoids rapid fluctuations in readings.

Accuracy and Sensitivity

The accuracy and sensitivity of the MQ-135 gas sensor were evaluated by exposing it to varying concentrations of cooking gas. The sensor's analog output was monitored and compared with known gas concentration levels to verify its response. The sensor demonstrated high sensitivity to changes in gas concentration, with a clear and consistent increase in output voltage corresponding to higher gas levels. This confirmed the sensor's ability to detect even small changes in gas concentration, making it suitable for early detection of gas leaks.

CONCLUSION

The IoT-based gas leakage monitoring system successfully demonstrated the integration of IoT principles in addressing the critical issue of gas safety. By combining the MQ-135 gas sensor, Arduino Uno, and ESP8266 Wi-Fi module, the system provides real-time monitoring and detection of hazardous gas concentrations, ensuring prompt alerts and notifications for users. The testing conducted confirmed the system's reliability and accuracy, showcasing its capability to operate effectively in various scenarios. With the successful simulation of gas concentration, verification of Wi-Fi connectivity, and demonstration of the alert mechanism, the system proved to be a valuable solution for enhancing safety in both residential and industrial environments. As gas leaks can lead to severe consequences, the development of this monitoring system highlights the importance of proactive measures in hazard prevention. The findings emphasized the potential of IoT applications in improving safety standards and protecting lives. Future improvements may include expanding the system's capabilities to include additional sensors or integrating machine learning algorithms for predictive analysis, further enhancing its effectiveness.

REFERENCES

Adekitan, A. I., Matthews, V. O., and Olasunkanmi, O. (2018). A microcontroller-based gas leakage detection and evacuation system. In *IOP Conference Series: Materials Science and Engineering* 413(1), IOP Publishing.

Adenubi, S., Appah, D., Okafor, E., and Aimikhe, V. (2023). A review of leak detection systems for natural gas pipelines and facilities. *Journal of Natural Gas Science and Engineering*.

Balaji, S., Nathani, K., and Santhakumar, R. (2019). IoT technology, applications and challenges: A contemporary survey. *Wireless Personal Communications*, *108*(1), 363-388.

Baranov, A. M., and Osipova, T. V. (2022). Latest progress in sensors for pre-explosive detection of flammable gases: A review. *Sensors and Materials, 34*(3). 76

Debnath, S., Ahmed, S., Das, S., Nahid, A. A., and Bairagi, A. K. (2020). IoT based low-cost gas leakage, fire, and temperature detection system with call facilities. In *2020 2nd International Conference on Advanced Information and*

Communication Technology (ICAICT) (pp. 11-16). IEEE.

Goyal, S., Sharma, N., Bhushan, B., Shankar, A., and Sagayam, M. (2021). IoT enabled technology in secured healthcare: applications, challenges and future directions. In *Cognitive Internet of Medical Things for Smart Healthcare: Services and Applications* (pp. 25-48).

Hasibuan, M. S., and Idris, I. (2019). Intelligent LPG gas leak detection tool with SMS notification. In *Journal of Physics: Conference Series* (1424(1), p. 012020). IOP Publishing.

Jumaa, N. K., Abdulkhaleq, Y. M., Nadhim, M. A., and Abbas, T. A. (2022). IoT-based gas leakage detection and alarming system using Blynk platforms. *Iraqi Journal of Electrical and Electronic Engineering*, *18*(1), 64-70.

Khan, M. A. H., Rao, M. V., and Li, Q. (2019). Recent advances in electrochemical sensors for detecting toxic gases: NO2, SO2 and H2S. *Sensors*, *19*(4), 905.

Kodali, R. K., and Rajanarayanan, S. C. (2019). IoT-based automatic LPG gas booking and leakage detection system. In *2019 11th International Conference on Advanced Computing (ICoAC)* (pp. 338-341). IEEE.

Rath, K. C., Khang, A., and Roy, D. (2024). The role of internet of things (IoT) technology in Industry 4.0 economy. In *Advanced IoT Technologies and Applications in the Industry 4.0 Digital Economy* (pp. 1-20). Springer.

Varma, A., Prabhakar, S., and Jayavel, K. (2017). Gas leakage detection and smart alerting and prediction using IoT. In *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)* (pp. 1-5). IEEE.