Neil Gogte Institute of Technology
(Approved by AICTE & Affiliated to Osmania University), Kachavanisingaram, Uppal, Hyderabad - 500088, Telangana.

PROJECT REPORT

Gas Leakage Detection and Alert System using ESP32 with Firebase, Telegram Bot, and Servo-based Response

B.E – Ⅱ SEMESTER

IN

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted by

TEAM-4

B.CHARAN TEJA GOUD - 245324748009

CH.SURYA TEJA - 245324748016

D.SANTOSH KUMAR - 245324748024

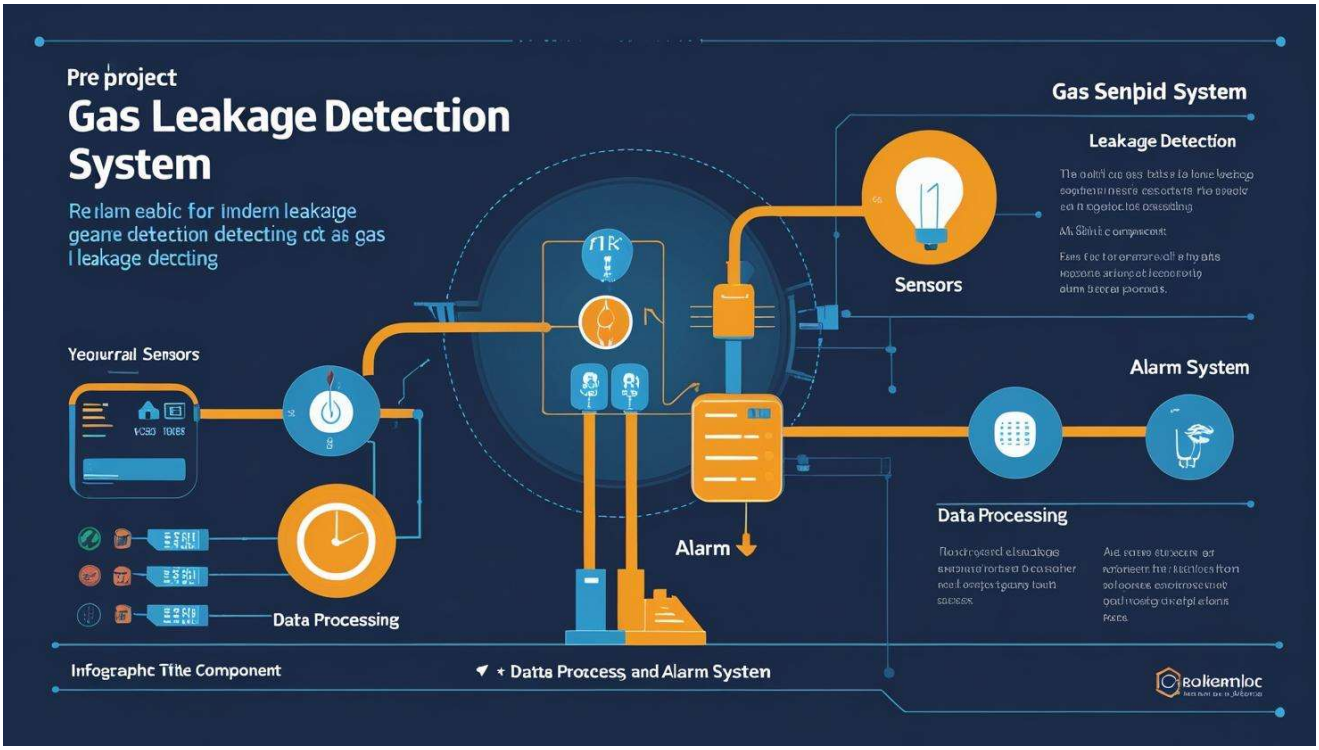
under esteemed guidance of

Guide Name : Mrs. GIRIJA SUTHOJU

Guide Designation : ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Table of Contents



Abstract / Project Summary

This project introduces a smart and reliable **Gas Leakage Detection and Alert System** aimed at enhancing safety in residential, industrial, and commercial environments. The system uses an **MQ-5 gas sensor** connected to an **ESP32 microcontroller**, which continuously monitors the presence of combustible gases in the environment. When the detected gas concentration exceeds a predefined threshold, the system immediately activates a **local buzzer** to alert nearby individuals.

To ensure timely remote alerts, the system integrates a **Telegram bot** that provides a **two-level alert mechanism**. The first alert is triggered when gas levels start to increase, serving as an early warning. If the gas concentration continues to rise and reaches a critical level, a second alert is sent. Simultaneously, a **servo motor is activated to trip a switch**, simulating an automatic safety response such as shutting off the gas supply or triggering an emergency power cutoff.

An **LCD display** is used to show real-time gas concentration data locally, allowing users to monitor the environment easily. Additionally, the system leverages **Firebase**, a cloud-based real-time database, to store and display alerts. However, to reduce unnecessary data traffic, the ESP32 is programmed to send only a **saved location link** to the Firebase console when the gas level crosses the danger threshold, rather than continuously uploading sensor data.

A **web-based interface** is also proposed for future development, which can serve as an alternative to Firebase for transmitting the location and system status. This modular and efficient system provides a proactive solution to gas leakage hazards by combining **real-time sensing, remote alerting, and automatic emergency response mechanisms**, ensuring both safety and scalability.

.

1.Introduction

Background and Context

Gas leaks are among the most silent yet devastating hazards in both domestic and industrial environments. Colorless and odorless gases like LPG, methane, and propane—commonly used for cooking and industrial heating—pose extreme risks when leaked, as they are highly combustible and toxic. The delay or failure in detecting such leaks can lead to explosions, fires, poisoning, property loss, and even fatalities.

Real-World Tragedies Due to Gas Leaks

Numerous incidents across the globe highlight the critical importance of early gas leak detection:

- **Vizag LG Polymers Gas Leak (India, 2020):** A massive gas leak from a chemical plant led to 12 deaths and over 1,000 people hospitalized, affecting multiple villages nearby.
- **Pune Gas Cylinder Explosion (India, 2023):** A domestic LPG leak caused an explosion in an apartment complex, resulting in the death of a family of four and leaving many injured.
- **Mexico City LPG Tanker Explosion (2013):** A gas tanker explosion on a highway killed 20 people and injured dozens, with shockwaves damaging homes and infrastructure.
- **Beirut Port Explosion (2020):** Although not a gas leak, this tragedy emphasized the dangers of improperly monitored chemical storage, highlighting parallels in monitoring volatile materials.

These incidents underline the pressing need for smart, automated, and real-time gas detection and response systems.

Problem Statement

Despite the prevalence of gas-powered appliances and industries, many existing gas leak detection systems are either:

- **Manual**, relying on human senses or basic detectors,
- **Not connected to a network**, thus unable to notify remote users,
- **Limited to sound alarms**, which may go unnoticed in the absence of occupants,
- **Lacking a physical response**, such as triggering emergency mechanisms to mitigate the leak.

As urban areas grow and automation becomes a necessity, there is a growing demand for intelligent systems that can detect gas leaks promptly, notify users remotely, and even take immediate physical action to reduce harm.

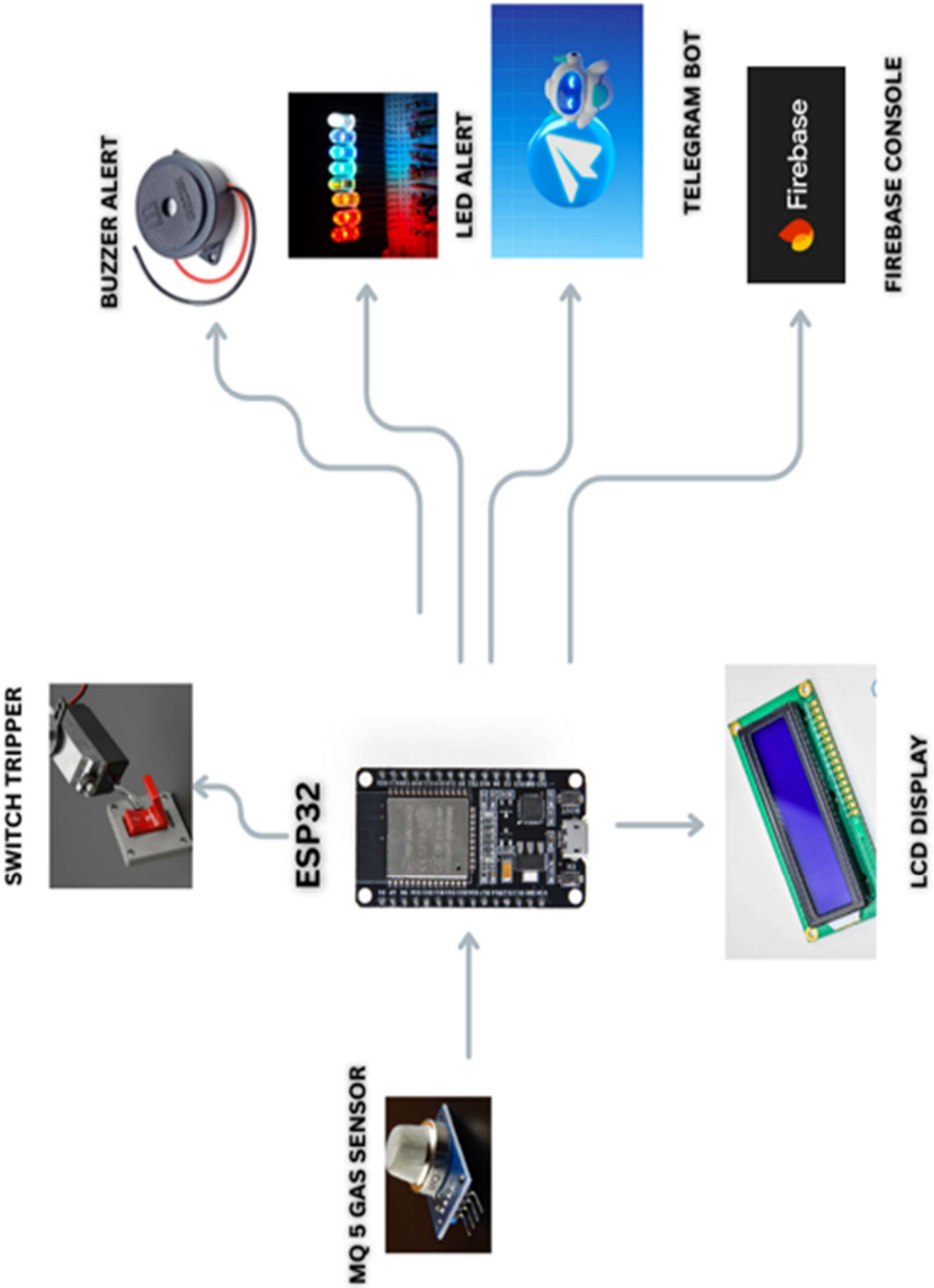
Proposed Solution

This project proposes a **Smart Gas Leakage Detection and Alert System** that not only senses gas levels using an **MQ-5 sensor** but also provides:

1. **Local Alerts** using a buzzer, LCD display, and LED indicators.
2. **Remote Notifications** through:
    - **Telegram Bot API** to send real-time alerts and respond to user queries.
    - **Firebase Realtime Database** to log alerts with geolocation for cloud-based monitoring.
3. **Physical Actuation** using a **servo motor** to simulate shutting off a valve or triggering a safety mechanism when a critical leak is detected.
4. **User Command Interface**, where a user can send /gaslevel to receive current gas levels remotely.

Built on the **ESP32 microcontroller**, which supports built-in WiFi and dual-core processing, the system ensures **reliability**, **low cost**, and **extensibility**. It merges IoT and embedded systems principles to create a fully autonomous safety tool suitable for households, restaurants, and industries.

2.Block Diagram



3. Method and Algorithm

Step-by-step Logic:

1. Initialize WiFi, LCD, and Servo Motor.
2. Continuously read analog values from MQ-5 gas sensor.
3. Classify gas levels into:
    - Safe (below WARNING_THRESHOLD)
    - Warning (between WARNING and ALERT thresholds)
    - Critical (above ALERT_THRESHOLD)
4. Display gas level on LCD.
5. If Warning:
    - Blink red LED.
    - Sound buzzer intermittently.
    - Send Telegram warning.
6. If Critical:
    - Send Firebase alert with location.
    - Activate Servo to simulate valve shutoff.
    - Flash red LED rapidly and continuous buzzer.
    - Send critical Telegram alert.
7. Telegram bot listens to /gaslevel command and responds with live gas readings.

4.Detailed Description

Hardware Components:

- ESP32 Dev Board
- MQ-5 Gas Sensor (Analog Output)
- I2C LCD Display (16x2)
- Buzzer
- Red and Green LEDs
- SG90 Servo Motor
- WiFi Router
- Power Supply (18650 Batteries or USB)

Software Platforms:

- Arduino IDE
- Firebase Real-Time Database
- Telegram Bot API
- ArduinoJson Library
- LiquidCrystal_PCF8574 for I2C LCD

Component Description:

ESP32: Dual-core microcontroller with built-in WiFi. Responsible for reading sensors, processing logic, and sending data.

MQ-5 Sensor: Detects combustible gases like LPG, CH4, CO.

LCD Display: Provides real-time gas level status.

Buzzer: Triggers alert in moderate and critical gas conditions.

LEDs:

- Green LED indicates safe status.
- Red LED indicates warning/alert status.

Servo Motor: Simulates automatic shutdown of gas valve.

Firebase: Stores gas leak alert with GPS location for cloud monitoring.

Telegram Bot: Sends alert and accepts /gaslevel command to respond with current gas levels.

Code Explanation:

WiFi Connection:

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) { delay(500); }
```

Sensor Reading and Classification:

```
int gasValue = analogRead(MQ5_SENSOR);
if (gasValue > ALERT_THRESHOLD) {...} else if (gasValue > WARNI
```

Firebase Posting:

```
StaticJsonDocument<200> doc;
doc["alert"] = "Gas Leak Detected";
doc["location"] = locationLink;
serializeJson(doc, requestBody);
http.POST(requestBody);
```

Telegram Integration:

```
bot.sendMessage(chatID, "🚨 ALERT: HIGH Gas Leak Detected! Take Immediate Action!", "");
```

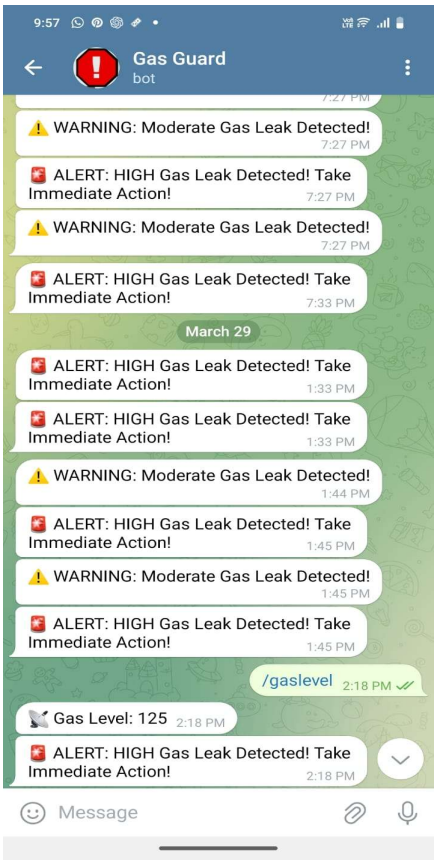
Servo Activation:

```
// ✅ Trigger Servo (if not already triggered)
if (!servoActivated) {
    tripServo.write(30); // Rotate to 90°
    delay(1000);         // Wait for servo to move
    tripServo.write(0);  // Optional: return to initial
    servoActivated = true;
}
```

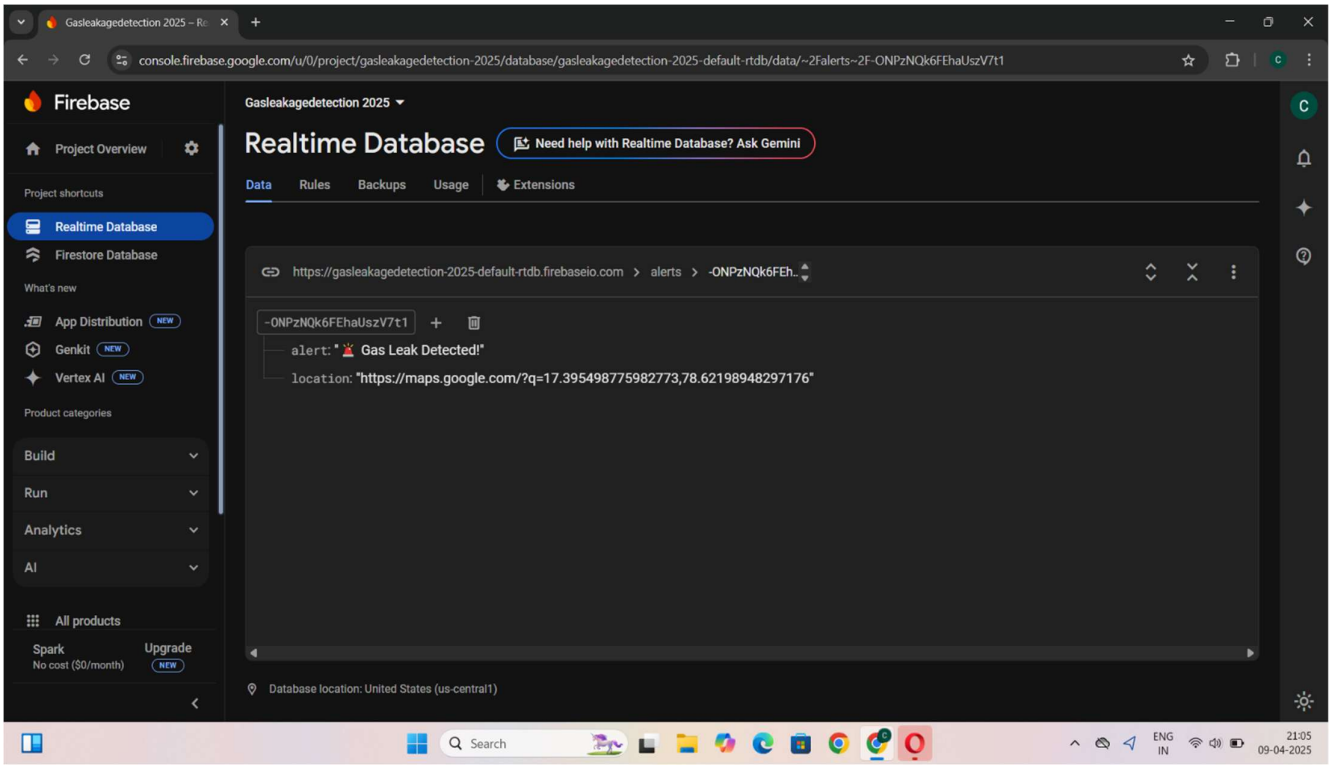
LCD Update:

```
lcd.setCursor(0, 0); lcd.print("Safe: No Gas Leak");
```

Telegram interface:



FireBase Console interface:



5.Project Analysis

5.1. Performance Evaluation

The system was evaluated in terms of detection speed, accuracy, communication latency, and overall responsiveness under simulated and controlled gas leak conditions. The results demonstrate that:

- Detection Latency: The MQ-5 sensor has a warm-up period but is capable of detecting gas concentration changes within 1–2 seconds after exposure.
- Response Time: The total system—from gas detection to buzzer activation and notification—is under 3 seconds for critical levels.
- Notification Speed:
    - Telegram alerts are received within ~2–5 seconds.
    - Firebase writes take approximately ~200–800ms depending on network bandwidth.

5.2. Real-Time Behavior

The system is optimized for real-time usage, and its loop executes within ~1 second cycles. The non-blocking design and use of timers (e.g., millis()) ensures the ESP32 can handle concurrent tasks such as:

- Continuous gas monitoring
- Display updates on LCD
- WiFi-based Firebase communication
- Telegram command listening
- Servo control

This architecture ensures non-interference between tasks, especially during alert conditions.

5.3. Safety and Reliability

- Redundancy: Alerts are sent via both Telegram and Firebase, providing both push and pull-based monitoring.
- Hardware Fail-safe:
    - In case of WiFi disconnection, the buzzer and servo still function locally.
    - LED indicators allow quick visual inspection.

- State Flags: Flags such as alertSent and servoActivated prevent system from re-triggering redundant actions during continuous high gas conditions.

5.4. Scalability

- The project can be expanded to multiple nodes in industrial zones by assigning unique paths in Firebase.
- Integration with other cloud platforms like ThingSpeak, Blynk, or Azure IoT Hub is possible without significant hardware changes.
- Telegram's bot API supports multiple commands and group broadcasts, making this scalable to building-level or institution-wide deployment.

5.5. Accuracy of MQ-5 Sensor

- Sensor Response: MQ-5 detects LPG, methane, and natural gases effectively.
- Analog Value Ranges:
    - Normal: < 300
    - Warning: 350–799
    - Alert: ≥ 800
- Environmental Sensitivity:
    - Sensor is influenced by humidity and temperature.
    - Dust and ventilation affect its readings.

To improve accuracy:

- Moving average filtering or Kalman filters could be implemented.
- Calibrate using reference gas concentrations in a controlled setting.

5.6. Servo Mechanism Analysis

The SG90 servo is used for simulated valve control. Key metrics:

- Trigger angle: 30 degrees rotation was found sufficient to mimic valve shut-off.
- Rotation speed: ~0.1 sec/60°
- Drawback: Cannot handle heavy mechanical loads; suitable only for demonstration or triggering relays.

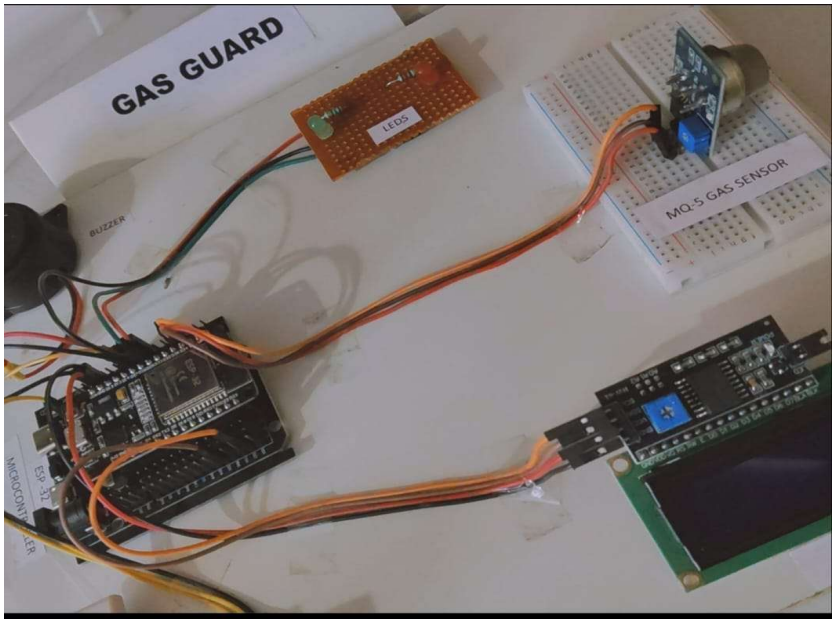
5.7. Firebase Integration Review

- Firebase Realtime Database stores timestamped alerts and location.

- The data path used: /alerts.json
- With minor enhancements, you can add:
    - Timestamp of alerts (millis() or ntp)
    - Gas level values with each entry
    - Alert ID and location name

5.8. Telegram Bot Integration Review

- The bot accepts /gaslevel command and replies with the real-time analog reading.
- Based on WiFiClientSecure and UniversalTelegramBot libraries.
- The bot enhances accessibility and user engagement, especially when:
    - The user is away from the premises
    - Multi-user access is needed (admin group chat)



5.9. User Interface and Experience

- LCD Display: Dual-line LCD provides clear real-time data.
    - Line 1: Gas leak status (Safe/Warning/Alert)
    - Line 2: Gas level (Analog value)
- LED Indicators:
    - Green: Safe

- Red: Warning/Alert

- Audio Feedback:
  - Short beeps: Warning
  - Continuous buzzer: Critical alert

This multimodal feedback ensures users are alerted through **sight, sound, and remote messages**.

5.10. Real-World Tragedy Connection

Numerous incidents underscore the importance of gas detection:

- **2014 GAIL Gas Pipeline Blast (Andhra Pradesh):** Killed over 20 people due to undetected gas accumulation.
- **2020 Vizag Gas Leak:** Over 10 people died, hundreds hospitalized.
- **Bhopal Gas Tragedy (1984):** Though not combustible gas, this event demonstrated the catastrophic results of gas leaks.

Such incidents could have been prevented or mitigated with real-time alerts and automated responses like those demonstrated in this project.

5.11. Data Flow Summary

Sensor → ESP32 → LCD + Alert System (Buzzer, LED, Servo) → WiFi → Firebase + Telegram Bot

6. Final Results

- Gas levels displayed correctly on LCD.
- Alerts successfully sent to Firebase and Telegram.
- Servo motor reliably triggers under critical alert.
- Telegram bot responds accurately to queries.

Challenges Faced:

- WiFi Fluctuations: Solved by using client.setInsecure().
- Telegram Rate Limiting: Managed with delay and interval checks.
- Servo Calibration: Required adjusting pulse width and position values.
- Sensor Fluctuation: Handled with threshold buffers.

Future Enhancements:

- Add GSM module as fallback for no-WiFi zones.
- Battery backup and voltage monitoring.
- Integrate voice alerts.
- Control servo remotely via Telegram.
- Upgrade to industrial-grade sensors.

Applications:

- Domestic kitchens
- Gas-powered industrial units
- Chemical plants
- Restaurants
- Mobile food trucks

7. Conclusion

This project presents a smart and reliable gas leakage detection system using ESP32. It combines modern IoT technologies including Firebase, Telegram Bot, and servo-based responses to create a secure and user-notified gas environment. Its modularity and open design make it ideal for real-world implementation and educational purposes.

8. References

- MQ5 Datasheet
- ESP32 Technical Reference Manual
- ArduinoJson Documentation
- Firebase Realtime Database Docs
- Telegram Bot API Docs
- LiquidCrystal_PCF8574 Library
- ESP32Servo GitHub Repository