Project Report

# G-Sentinel IoT - ARGUS (Smart Gas Leakage Detection & Alert System)

Prepared by: **Omar Abu Abbass**

## 1. SYSTEM DESCRIPTION

The G-Sentinel IoT (ARGUS) is a smart safety system designed to monitor gas leakage levels in real-time. ARGUS uses the power of the Internet of Things thanks to the integration of the ESP32 microcontroller. The gas concentration is constantly tracked by the use of an MQ-6 gas sensor, and the results are relayed to a cloud server by Wi-Fi connectivity. The safety statuses can be viewed, as well as instant “High Alert” notifications, whenever the gas concentration levels go beyond the set safety limit, by end users through a mobile app. The system has local safety devices, such as a buzzer and LED display, to offer immediate safety warnings on site. **Figure 1** illustrates the high-level system architecture and data flow.

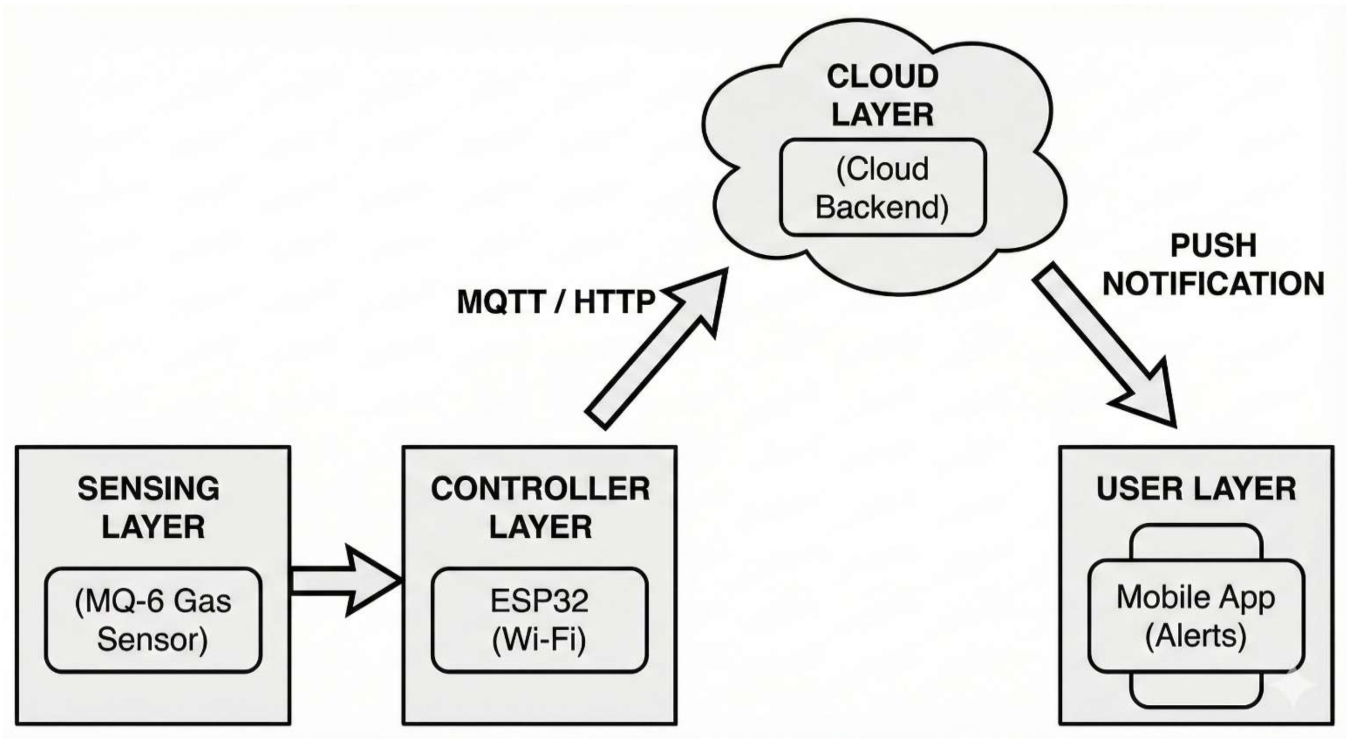

Figure 1 System Architecture of G-Sentinel IoT (ARGUS) showing data flow from sensing layer to user application.

## 2. TEST SCOPE

The test process emphasizes the reliability of the sensing, processing, and transmission layers.

- In-Scope (Included):
  - Functionality of the MQ-6 Gas Sensor and analog-to-digital mapping.
  - ESP32 Wi-Fi connectivity and transmission of data to Cloud.
  - Responsiveness of the Mobile Application (receiving alerts).
  - Local Alarms (The buzzer alarm and the LED Matrix alarm).
- Out-of-Scope (Excluded):
  - Testing the chemical properties of the gas itself.
  - Stability of the external Internet Service Provider (ISP).
  - Durability of the casing (Physical stress tests).

## 3. TEST OBJECTIVES

The major goals of this test plan are given below:

1. *Accuracy*: To confirm whether the readings of the gas sensor are correctly translated to the 0-10 scale, as well as whether the warning is given at the right point.
2. *Connectivity*: For the system to reliably connect to the Wi-Fi network and reconnect automatically in the event of signal loss.
3. *Latency*: To ensure that the time difference between the detection of the gas leakage and the mobile notification does not exceed 3 seconds.
4. *Integrity*: The aim of this is to ensure that the data viewed in the Mobile App is accurate and reflects the actual sensor values.

## 4. TEST STRATEGY

For the proposed research, a multi-layer testing method is employed to validate the reliability of the G-Sentinel IoT system. The following matrix (Table 1) highlights the system components for verification, divided into hardware, software, network, and security testing:

Table 1. Test Strategy Matrix for G-Sentinel IoT System.

| **Test Type** | **Targeted Component** | **Description** |
|---|---|---|
| **Hardware** | MQ-6 Sensor & Wiring | This stage involves the checking of correct connections between power and grounds (VCC and GND) as well as the warming characteristics of the MQ-6 sensor to ensure it works as expected without overheating problems.. |
| **Software** | Decision Logic & App | There should be unit tests for the software logic (for example, evaluation of conditions like the concentration of gas exceeding the set limit). Additionally, the stability of the Mobile App should be checked to prevent its unexpected termination while frequently updating the data. |
| **Network** | Wi-Fi Stability | Assess the stability of data transmissions under different levels of Wi-Fi signal strength and ensure that the system responds to packet loss in a graceful manner. |
| **Security** | Authentication & Data | Ensure that the Mobile App has user authentication (login) and that cloud data transmission is encrypted (using HTTPS/MQTT Security). |

To provide further detail on the Software Testing component, the following flowchart illustrates the decision-making logic programmed into the ESP32 controller:

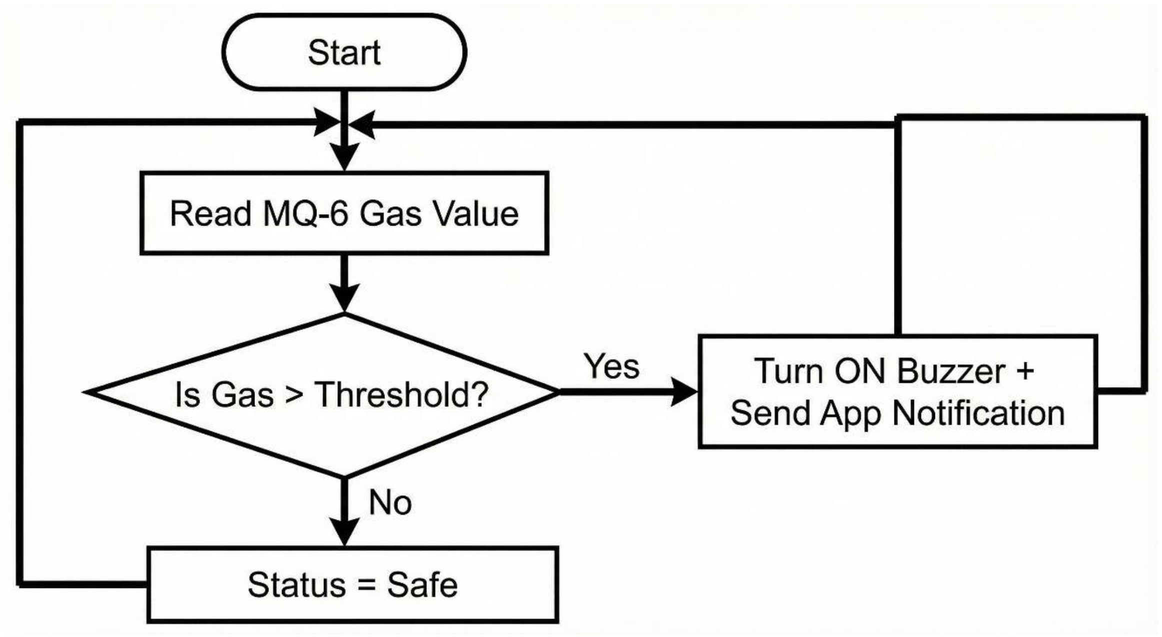

Figure 2 Software Logic Flowchart illustrating the decision-making process for gas detection and alerts.

## 5. TEST ENVIRONMENT

The testing process focuses on the reliability of the sensing, processing, and transmission layers.

- Hardware: ESP32 Development Board, MQ-6 Gas Sensor, Buzzer, LED Matrix, Android Smartphone, Wi-Fi Router (2.4 GHz).
- Software: Arduino IDE (for firmware testing), Serial Monitor, Cloud Dashboard (e.g., Blynk/ThingsBoard), Mobile Application.

The experimental setup configured to validate the system functionality is depicted in Figure 3 below. It consists of the development station, the microcontroller unit, and the monitoring device:

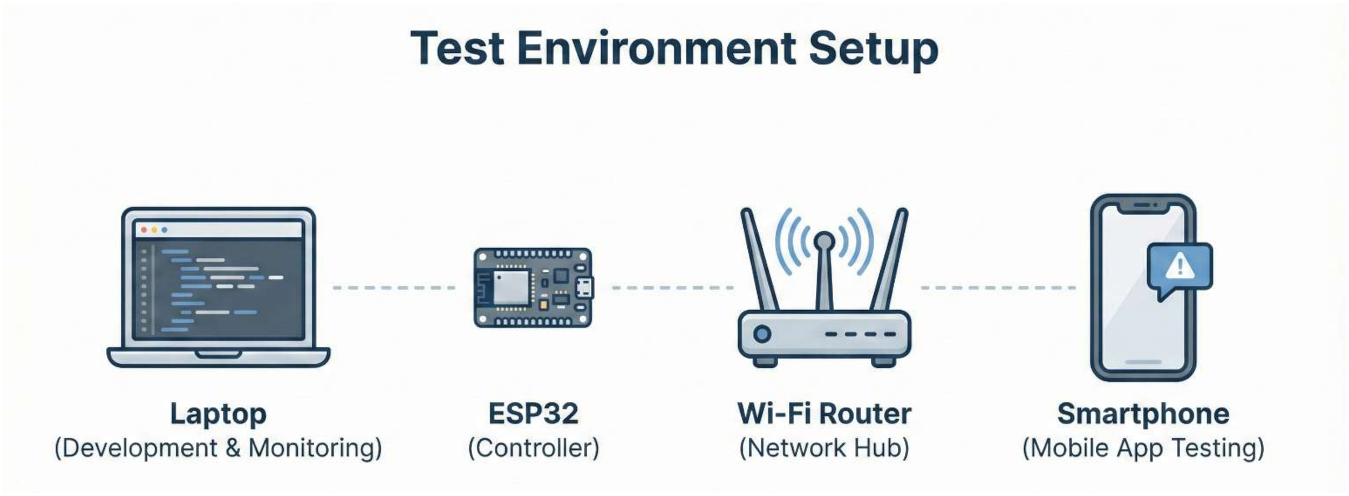

Figure 3 Test Environment Setup showing the interaction between the ESP32 controller, sensors, and debugging tools.

## 6. COMMON RISKS

The following risks have been identified along with mitigation strategies:

- Risk: Sensor Noise/Fluctuation: The gas sensor could potentially be providing false positives because of air drift and/or temperature variations.
  - Mitigation: Incorporate software filtering (average of last 10 readings) in the code.
- Risk: Network Latency: Notifications can be delayed because of a poor internet connection.
  - Mitigation: Implement lightweight protocols such as MQTT. Ensure that local notifications (Buzzer) are Wi-Fi independent.
- Risk: Power Failure: The device won't work if there is no power.
  - Mitigation: It is designed with an external Battery Backup.