KL University Hyderabad

Electronic System Design
24SDEC01
Project Based Learning
Review 2

Gas leakage detection, Prediction and Alert

18th August 2025

OVERVIEW

Gas leaks pose serious threats to both residential and industrial safety, with the potential to cause fires, explosions, and health hazards. Traditional gas detectors are often limited to standalone alarms that provide no remote monitoring or automated control, making them ineffective in many real-world situations.

This project proposes a smart gas leakage detection system that combines embedded hardware and IoT communication protocols. Using an MQ-2 gas sensor, a Raspberry Pi, an exhaust fan, and MQTT-based alerting via HiveMQ Cloud, the system is capable of detecting hazardous gas concentrations, activating ventilation systems automatically, and sending real-time alerts to users.

GOALS

1. Detect the presence of hazardous gases (e.g., LPG, methane) in real-time using the MQ-2 gas sensor
2. Automatically activate an exhaust fan upon gas detection to ventilate the environment
3. Send instant cloud-based alerts using the MQTT protocol via HiveMQ
4. Design and implement a compact, reliable PCB for hardware integration
5. Ensure the system operates autonomously with minimal human intervention

SPECIFICATIONS

🔌 Hardware Specifications

- **Controller**: Raspberry Pi 2 Model B
- **Gas Sensor**: MQ-2 (detects LPG, methane, smoke)
- **Actuator**: 12V DC exhaust fan
- **Switching Device**: NPN transistor (e.g., BC547) or relay module
- **Power Supply**:
    - 5V/2A Micro-USB for Raspberry Pi 2
    - 12V for fan (with voltage regulation if needed)
- **Other Components**: Resistors, diode, LED, terminal blocks

🌐 Communication

- **Protocol**: MQTT via `paho-mqtt` (Python library)
- **Broker**: HiveMQ Cloud (or local Mosquitto for offline testing)
- **Connection**: External USB Wi-Fi dongle (or Ethernet)

💻 Software Environment

- **OS**: Raspbian Lite or Raspberry Pi OS
- **Language**: Python 3.x
- **Libraries**: `paho-mqtt`, `RPi.GPIO`, `time`, `json`
- **Tools**: Thonny IDE / VS Code + SSH access

📏 Performance Parameters

- **Detection Time**: < 2 seconds
- **Fan Activation**: Instant on threshold exceed
- **Alert Delivery (MQTT)**: < 1 second on Wi-Fi

LITERATURE REVIEW:

Gas leakage is a critical safety concern in households, laboratories, and industries where gases like LPG, methane, or propane are used. Traditional systems rely on manual observation or standalone alarms, which are insufficient in environments where real-time alerts and automated response are essential. Recent advancements in embedded systems and IoT technologies have led to the development of smarter gas detection systems. These solutions integrate gas sensors with microcontrollers or single-board computers, using cloud-based protocols like MQTT for remote monitoring and control.

One of the most widely used sensors in gas detection systems is the MQ series gas sensors, specially the MQ-2 gas sensor, used to “sniff out” widely known harmful gases such as LPG, methane, hydrogen, smoke,etc. It is also cost effective and can be embedded easily. The sensor's output voltage varies proportionally with the concentration of gas in the surrounding environment, due to a change in its internal resistance. Each gas affects the resistance differently, but the MQ-2 is not selective. Projects and studies (e.g., Sharma et al., 2019) have shown that the MQ-2, when calibrated, can detect gas concentrations above 300 ppm reliably.

Now that we know which sensor we are using and why, we need to interface it with something more generalized so that it can become a system instead of just a component. This is where Raspberry Pi 2 comes into the picture. It is not just a microcontroller, it is a whole mini computer- perfect for IoT systems that need communication and intelligence. Compared to microcontrollers, RPi can handle multiple tasks like gas sensing, fan control, and MQTT communication simultaneously. Raspberry Pi 2 enables real-time data processing and cloud communication in embedded applications.

Once the system detects the presence of gas, it must immediately trigger a physical response to reduce risk. This is achieved by activating a 12V exhaust fan, which helps ventilate the environment and disperse the leaked gas. However, a RPi alone cannot power the DC fan since it produces a current that is 16 mA or less. To overcome this limitation, we need to use a transistor switch or a relay module. This operation is more complicated than the switch powering the fan on. Once the RPi GPIO is set to HIGH, the transistor base gets activated which allows current to flow, this current powers the fan ON. Think of the transistor like a valve. The Pi opens the valve slightly, and a big flow of current goes to the fan.

Detecting the gas is just one part of the system – the other critical component is ensuring that the alert reaches the user in real time. This is where MQTT (Message Queuing Telemetry Transport) plays a vital role. MQTT is a lightweight and efficient messaging protocol designed specifically for IoT systems, allowing devices to communicate with one another over a network using a publish/subscribe model. In this project, the Raspberry Pi 2 acts as a publisher, sending gas alerts to a cloud-based MQTT broker called HiveMQ, which then forwards the message to all subscribed clients instantly. This system flow can be integrated into applications like MyGate or smart safety dashboards, offering a range of real-world use cases in residential and industrial safety automation.

Several gas leakage detection systems have been implemented using microcontrollers like Arduino, often combined with buzzers or GSM modules for alerting. While functional, these systems lack scalability and real-time cloud integration. More recent designs using Raspberry Pi and MQTT have improved communication, but still rely on text-based data formats like JSON. To overcome these limitations, we tried to include a more scalable service. To overcome these limitations and make the system more scalable in the future, this project looks at using Google Protocol Buffers (Protobuf) — a method of sending data in a smaller and faster format compared to JSON and even MQTT.

Gas detection systems have evolved significantly over the past decade. Earlier systems were limited to basic setups using gas sensors connected to buzzers or GSM modules, offering only local or delayed alerts. With the advancement of embedded systems and IoT protocols, modern designs now support real-time cloud communication, automated actuation, and remote monitoring. This project represents the present stage of development, where Raspberry Pi, MQTT, and a custom PCB work together to detect, respond, and alert instantly. Looking forward, the system has been designed to support future upgrades like Protocol Buffers and mobile dashboard integration, allowing for faster communication, predictive analytics, and smarter control systems suitable for both domestic and industrial deployment.

Common designs focus solely on local alerts or use GSM modules that lack scalability and flexibility. Few systems integrate cloud-based communication, real-time actuation, and reliable PCB-level design in a single package. Additionally, structured and efficient data transmission formats like Protocol Buffers are rarely explored at the undergraduate level. This project aims to bridge these gaps by

combining real-time detection, automated response, and scalable communication, all integrated into a compact and modular embedded system that can be deployed in real-world safety-critical environments.

In summary, the literature reveals a strong foundation in gas detection using embedded systems and highlights the growing integration of IoT protocols like MQTT. However, there remains a clear gap in combining real-time actuation, remote alerting, efficient data communication, and compact PCB design into a single, deployable system. This project addresses that gap by connecting reliable hardware, cloud-based communication, and future-ready protocols like Protobuf (emerging technology) . The proposed system not only aligns with current trends in embedded safety design but also pushes beyond traditional student-level implementations by focusing on modularity, scalability, and real-world applicability. It lays the groundwork for further research into predictive gas monitoring, edge analytics, and AI-based safety automation.