**RTOS-Based LPG Gas Leak Detection and Alerting System**

Alinda Arnold, Kayong David, Wu Junjie.  
School of Electrical Engineering, Malaysia University of Science and Technology

# Abstract

This report presents the design and implementation of an RTOS-based LPG gas leak detection and alerting system using the STM32F446 microcontroller and Mbed OS. The system employs an MQ2 gas sensor to monitor LPG levels and triggers alerts through an LED, buzzer, and motor when dangerous gas concentrations are detected. The implementation leverages the real-time capabilities of Mbed OS to manage concurrent tasks, ensuring timely detection and response. Key functionalities include sensor data acquisition, threshold-based alerting, and synchronized task execution using semaphores.

# 1. Introduction

The presence of LPG (liquefied petroleum gas) in the environment poses significant risks due to its highly flammable nature. Early detection of gas leaks is crucial to prevent potential hazards such as explosions and fires. This project aims to develop an efficient and reliable LPG gas leak detection and alerting system using the STM32F446 microcontroller and the Mbed OS real-time operating system. The system is designed to continuously monitor LPG levels using an MQ2 gas sensor and promptly activate alert mechanisms including an LED, buzzer, and motor when dangerous gas concentrations are detected. The use of Mbed OS allows for the effective management of multiple tasks running concurrently, ensuring that the system can respond in real-time. Key components of the system include sensor data acquisition, threshold-based decision-making, and synchronized task execution to maintain system integrity and performance.

# 2. Software/Tools Setup

## 2.1 Integrated Development Environment (IDE)

This project utilizes Mbed OS Studio, which we installed and took advantage of its built-in RTOS libraries. During the setup, we ensured to select Mbed OS version 6 to leverage its latest features and capabilities

## 2.2 Toolchains in Mbed OS Studio Version 6

Mbed OS Studio primarily uses the Arm Compiler 6 for compiling projects. This compiler is optimized for ARM Cortex-M processors, providing advanced code optimizations and compliance with the latest C and C++ standards. It ensures efficient performance and compact code, making it ideal for embedded systems development. The integration of Arm Compiler 6 within Mbed OS Studio simplifies the development process, offering robust support for ARM architecture and comprehensive debugging capabilities.

## 2.4 Drivers and Libraries

In our RTOS-based LPG gas detection program, we utilized essential libraries and drivers provided by Mbed OS. The key libraries included mbed and rtos. The mbed library provided fundamental functionalities such as handling the ADC for reading analog values from the MQ2 gas sensor and managing GPIO operations for controlling the LED, buzzer, and motor driver. The rtos library enabled real-time operating system capabilities, allowing us to effectively manage multiple concurrent tasks. Additionally, we used drivers specific to the STM32F446 microcontroller to interface with its peripherals, ensuring smooth communication and control of the hardware components. These libraries and drivers were crucial for building a reliable and efficient embedded system.

# 3. Configuration Steps

## 3.1 Creating a Project with MbedOs Studio

1. Open MbedOs studio and create a new project.  
2. Select the MbedOS version.  
3. Name the project and then click enter.

## 3.2 Peripheral Configuration

1. Enable and Configure ADC Interface:

We configured the ADC interface for the MQ2 gas sensor. This involved selecting the ADC pin (A0) and setting up the ADC to read analog values from the sensor, allowing us to measure the LPG concentration.

1. Enable and Configure GPIO Pins:

We configured the GPIO pins for controlling the LED, buzzer, and motor driver:

1. LED: Configured on GPIO pin D2.
2. Buzzer: Configured on GPIO pin D3.
3. Motor Driver: Configured on GPIO pins D4 (IN1) and D5 (IN2).

## 3.3 Generating Code

Once the project settings are configured, click the "Save" button to apply the changes. Then, use the "Compile" button (represented by a hammer icon) to generate the project code. This process compiles the code and builds the project, providing a structured framework that includes initialization functions, peripheral configurations, and the main application logic. The compilation output, including any errors or warnings, is displayed in the console window, allowing issues to be addressed promptly. Once the build is successful, the resulting binary file can be flashed to the microcontroller.

# 4. Firmware Development Steps

## 4.1 Initializing Functions and Threads

**4.1.1 Global Initializations**

We performed global initializations for threads, sensor variables, and the threshold value:

Variables:

volatile float lpg\_value = 0; - Stores the current LPG sensor reading.

Semaphore sem\_sensor (1); - Ensures synchronized access to lpg\_value.

#define LPG\_THRESHOLD 100 - Sets the threshold for detecting dangerous LPG levels.

## 4.1.2 Thread Initializations

* Sensor Task (read\_mq2\_sensor): High-priority thread that reads the LPG sensor value via the ADC pin (A0).
* Motor Task (handle\_motor): High-priority thread that controls the motor using GPIO pins (D4 and D5).
* LED Task (handle\_led): Normal priority thread that controls the LED using GPIO pin (D2).
* Buzzer Task (handle\_buzzer): Normal priority thread that controls the buzzer using GPIO pin (D3).

## 4.1.3 Function Prototypes

We defined the following function prototypes to ensure readiness for execution:

void read\_mq2\_sensor ();

void handle\_led ();

void handle\_buzzer ();

void handle\_motor ();

## 4.1.4 Main Function Initialization

Implement the logic to process the acquired temperature data, ensuring real-time data processing to maintain system accuracy.

Main Function Initialization:

In the main () function, the RTOS threads are initialized and started. This setup includes:

Starting the sensor reading thread (thread\_sensor. start(read\_mq2\_sensor) ;).

Starting the motor control thread (thread\_motor. start(handle\_motor) ;).

Starting the LED control thread (thread\_led. start(handle\_led) ;).

Starting the buzzer control thread (thread buzzer. start(handle\_buzzer) ;).

The main loop is kept simple to allow the RTOS to manage task execution:

The while (true) loop contains a ThisThread::sleep\_for(1000ms); call, which effectively makes the main thread idle while allowing the RTOS to schedule and run the initialized tasks based on their priorities and synchronization mechanisms.

## 4.1.5 Logic Integration and Interaction

The RTOS-based LPG gas detection system operates by concurrently running multiple tasks using Mbed OS. The sensor task, running with high priority, reads the LPG sensor value via ADC every 500ms and updates the shared lpg\_value variable. The motor task, also a high priority, checks this value every 500ms and activates the motor if the LPG concentration exceeds a set threshold (100). Similarly, the LED and buzzer tasks, both with normal priority, turn on the LED and buzzer respectively based on the same threshold check. A semaphore ensures synchronized access to the lpg\_value variable, preventing race conditions. The main function initializes these threads and enters an idle loop, allowing the RTOS to handle task scheduling and execution, ensuring timely detection and response to LPG gas levels.

**5.0. Steps for Hardware Development**

The developed system interfaces several modules with a 32-bit STM32F446RE ARM microcontroller. These modules include an MQ-2 Gas Sensor for LPG leak detection, a motor driver control module for driving a 3.3V fan, and a buzzer module for audio alerts.

The MQ-2 Gas Sensor detects Liquefied Petroleum Gas (LPG) leaks and triggers the fan activation. The fan, implemented using a 3.3V motor, connects to the microcontroller through a dedicated driver control circuit. This motor driver control module, sourced from Texas Instruments, receives its power from a separate 9V external battery. While the modules have separate power supplies, they all share a common ground connection.

The overall system schematic was designed using EasyEDA software, seamlessly integrating all the modules. The provided circuit diagram illustrates the communication pathways between these components.

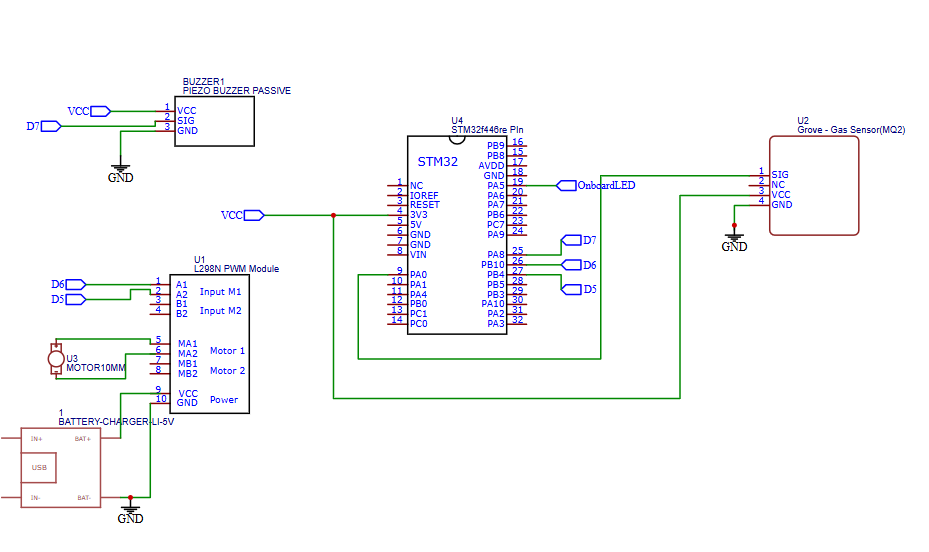


Figure 1 Schematic Diagram of a Gas Leakage Detection System

**5.1. Components and their Specification**

**5.1.1. Microcontroller (STM32F446RE)**

The STM32F446RE is a high-performance 32-bit ARM microcontroller from STMicroelectronics belonging to the STM32F4 family. The following is a brief description of its key features and specifications (STMicroelectronics., n.d.):

1. **Core and Memory:** ARM Cortex-M4 Core: This 32-bit core operates at a frequency of up to 180 MHz, offering good processing power for various embedded applications.
2. **Flash Memory:** Up to 512 Kbytes of embedded Flash memory provide ample space for storing program code and data.
3. **SRAM:** Up to 128 Kbytes of SRAM offer fast access to frequently used data.

**Key Features:**

**DMA (Direct Memory Access):** This feature allows for efficient data transfer between memory and peripherals without CPU intervention, improving overall performance.

Multiple Interfaces: The microcontroller boasts a rich set of interfaces for communication and peripheral control, including:

**UART** (Universal Asynchronous Receiver/Transmitter) for serial communication

**SPI** (Serial Peripheral Interface) for high-speed data transfer

**I2C** (Inter-Integrated Circuit) for low-speed communication with sensors and other devices

**USB OTG** (On-The-Go) for connecting to a host or acting as a host itself

**Low-Power Operation:** The STM32F446RE features various power-saving modes to optimize battery life in portable applications.

**Digital Signal Processing (DSP) Instructions:** The ARM Cortex-M4 core includes DSP instructions, making it suitable for signal processing tasks.

**5.1.2 MQ2 Gas Sensor Module**

The SEED STUDIO Grove – Gas Sensor (MQ2) module is a metal oxide semiconductor (MOS) sensor designed for gas leakage detection in both home and industrial environments. It exhibits sensitivity to various gases like Hydrogen (H2), Liquefied Petroleum Gas (LPG), Methane (CH4), Carbon Monoxide (CO), Alcohol, Smoke, Propane (Seeed-Studio-Wik, n.d.).

**Features:**

1. High sensitivity and fast response time for gas detection.
2. Easy to use with Grove Base Shield or Arduino boards.
3. Wide operating voltage range (5 V DC).

**5.1.3. DC Motor (MOT1N)**

The VELLEMAN DC Motor 3V (MOT1N) is a small, brushed DC motor. It operates on a voltage range of 1.5 V to 3 V DC and offers a balance of speed and torque (Velleman., n.d.).

**Specifications:**

**Voltage:** Operating Range: 1.5 V to 3 V DC (nominal voltage: 3 V)

**Current**: No Load Current: 0.3 A. Current at Maximum Efficiency: 1.33 A

**Speed:** No Load Speed: 14200 RPM. Speed at Maximum Efficiency: 11780 RPM

**Torque:** Stall Torque: 115.0 g⋅cm**.** Maximum Torque: 19.6 g⋅cm

**Power Rating**: 2.37 W

**Motor Size**: Diameter: 23.8 mm, External Length: 27 mm

**5.1.4. Motor Driver Controller (L298N)**

The L298N is an integrated circuit (IC) designed to drive two DC motors bi-directionally. It allows you to control the speed and direction of each motor independently (STMicroelectronics., L298 Dual Full Bridge Motor Driver - Datasheet., n.d.).

**Specifications:**

**Operating Voltage:** Driver Supply Voltage (VS): +5 V to +46 V DC (Typical: +12 V)

**Logic Supply Voltage (Vss):** +5 V to +7 V DC (Can be powered by onboard regulator if VS > 7 V)

**Motor Drive Current:** Up to 2 A per channel (4 A total)

**Logic Control Inputs:** Four digital input pins for controlling motor direction and enabling/disabling (IN1, IN2, IN3, IN4)

**Current Sense Outputs:** Two analog outputs (CSA, CSB) for monitoring motor current (requires external resistors for conversion)

**Over-temperature Protection:** Built-in protection to shut down the driver in case of overheating

**5.1.5. Piezoelectric Passive Buzzer.**

A passive buzzer module is a simple electronic device that generates sound when an electrical signal is applied. It typically consists of a piezoelectric element that vibrates when an alternating current (AC) signal is applied across its terminals.

**Specifications (element14, n.d.):**

**Operating Voltage**: 3 V - 12 V DC

**Frequency Response:** 2 kHz - 4 kHz (approx.)

**Sound Output:** Single tone

**Current Consumption**: 20 mA (typical)

**Physical Dimensions**: Diameter: 12 mm, Height: 10 mm.

**Appendix**

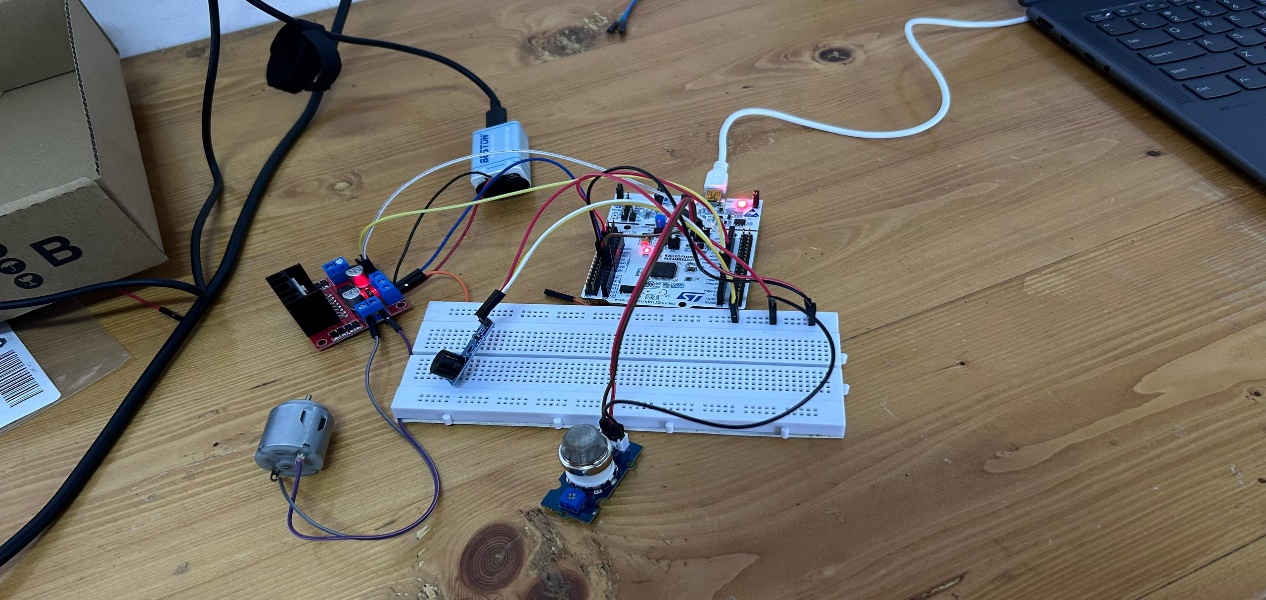


Figure 2 Image of system prototype

|  |  |
| --- | --- |
|  |  |

Figure 3 showing the output at the terminal of the Mbed Os

# References

element14. (n.d.). *multicomp-pro*. Retrieved from https://my.element14.com/multicomp-pro

Seeed-Studio-Wik. (n.d.). *Grove - Gas Sensor(MQ2).* Retrieved from https://wiki.seeedstudio.com/Grove-Gas\_Sensor/

STMicroelectronics. (n.d.). *L298 Dual Full Bridge Motor Driver - Datasheet.* Retrieved from https://www.st.com/resource/en/datasheet/l298.pdf

STMicroelectronics. (n.d.). *STM32F446RE: High-performance foundation line, Arm Cortex-M4 core with DSP and FPU, 512 Kbytes of Flash memory, 180 MHz, ART Accelerator, Dual QSPI*. Retrieved from https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html

Velleman. (n.d.). *DC Motor 3Vdc 300mA 14200rpm (1.5-3Vdc)*. Retrieved from https://www.velleman.eu/products/view?id=375872&lang=de&country=be