

KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COE 381 MICROPROCESSORS



LPG LEAK DETECTION SYSTEM

GROUP 4

GROUP MEMBERS:

1. TANDOH ELIZABETH MAAME AMA	1828522
2. NANA KWASI OPARE LARBI	1823822
3. ISAAC QUARTEY	1827422
4. ADJAPONG YEBOAH CLEMENT	1813722
5. EDWARD MENSAH OWUSU	1826822
6. ASIAMAH OMARI YAW	1818322
7. ISHOLA FAAZELE ADEBIYI	1853722
8. ASIEDU ENOCH OFORI KWASI	1818522
9. DEFIAT SELORM ZION	1821422

Contents

Project Overview.....	3
Features	3
Schematic and Connections.....	3
Components Used.....	3
APPROACH.....	4
Wiring Instructions:.....	4
Software & Tools.....	7
Installation & Setup.....	7
File Structure	8
Usage.....	8
Conclusion	8
Challenges	9
Future Improvements.....	9
Credits and Reference	9

Project Overview

This project is an LPG (Liquefied Petroleum Gas) leak detection system developed as part of a microprocessors course. The system continuously monitors the LPG concentration in the air using a gas sensor. When the LPG level surpasses a predefined threshold, the system:

- Sounds an alarm using a buzzer.
- Blinks an LED bar proportional to the LPG level.
- Displays the LPG concentration on an LCD screen.
- Sends real-time data to a Blynk dashboard, visualizing the LPG level using a gauge and chart.
- Triggers an alarm notification on the Blynk dashboard.
- Allows users to adjust the LPG threshold for detection via the Blynk dashboard.

Features

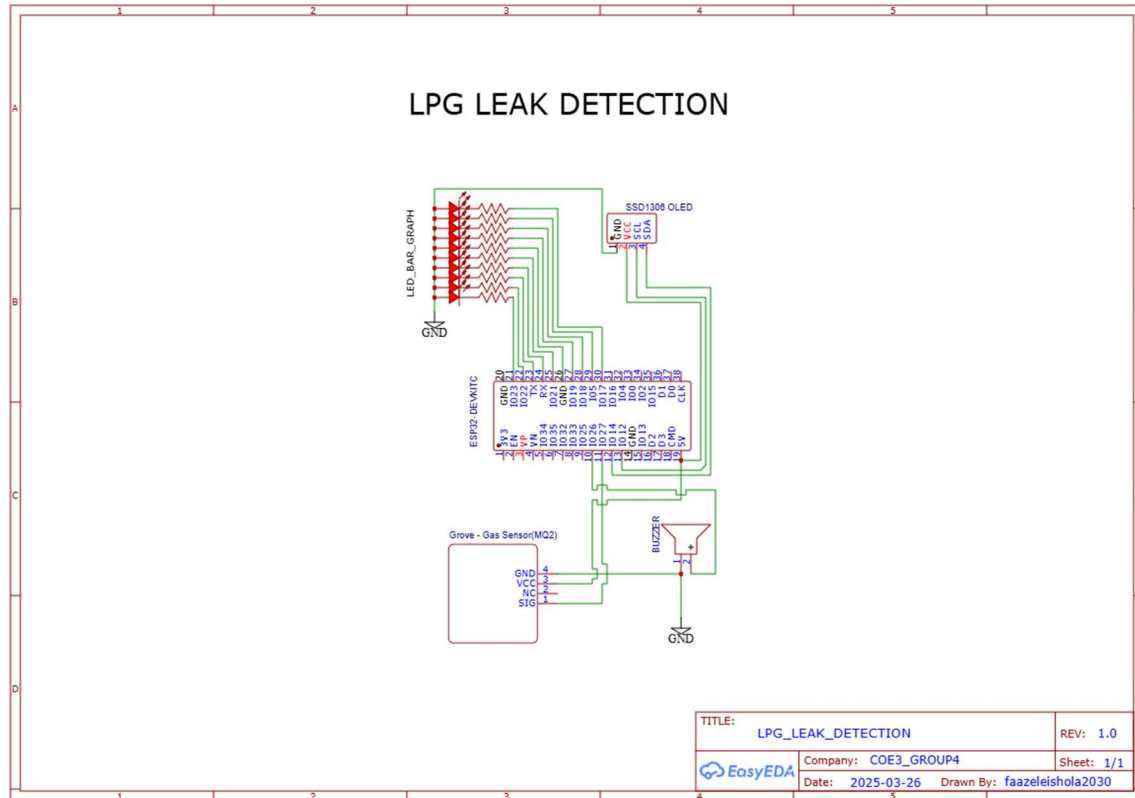
- **Real-time LPG Monitoring:** Continuously reads LPG levels using a gas sensor.
- **Alarm System:** Triggers a buzzer when the LPG level crosses the set threshold.
- **Visual Indicators:** Blinking LED bar and OLED display for real-time monitoring.
- **Remote Monitoring:** Integration with Blynk dashboard for remote visualization.
- **Configurable Threshold:** Users can adjust the detection threshold via the Blynk app.
- **Platform Support:** Developed using PlatformIO in VS Code with Wokwi simulation support.

Schematic and Connections

Components Used

1. ESP32 Microcontroller
2. MQ-2 Gas Sensor (LPG detection)
3. OLED Display (SSD1306) (LPG level visualization)
4. Buzzer (Alarm system)
5. LED Bar (Indicates LPG level)
6. Ten 1k ohms resistors

7. Blynk IoT Dashboard (Remote monitoring and control)



APPROACH

The LPG leak detection system uses an ESP32 microcontroller to control various components, including the MQ2 gas sensor, an OLED display, a buzzer, and an LED bar graph. The system continually monitors the gas levels and activates an alarm if the concentration surpasses a pre-set threshold. The data is also sent to Blynk, an IoT platform that allows for remote monitoring of the system.

Wiring Instructions:

1. MQ-2 Gas Sensor (LPG Detection)

Pin	Connection
VCC	Connect to 3.3V
GND	Connect to GND
SIG	Connect to a viable ESP32 analog pin

2. OLED Display (SSD1306)

Pin	Connection
VCC	Connect to 3.3V
GND	Connect to GND
SDA	Connect to ESP32 Pin 21
SCL	Connect to ESP32 Pin 22









3. Buzzer (Alarm System)

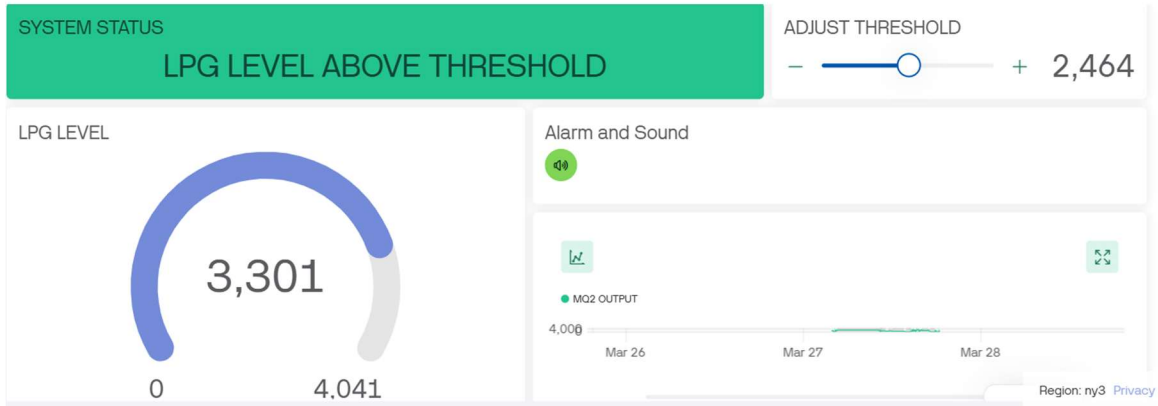
Pin	Connection
+ (Positive)	Connect to a viable ESP32 digital output pin
- (Negative)	Connect to GND

4. LED Bar Graph (Indicates LPG Level)

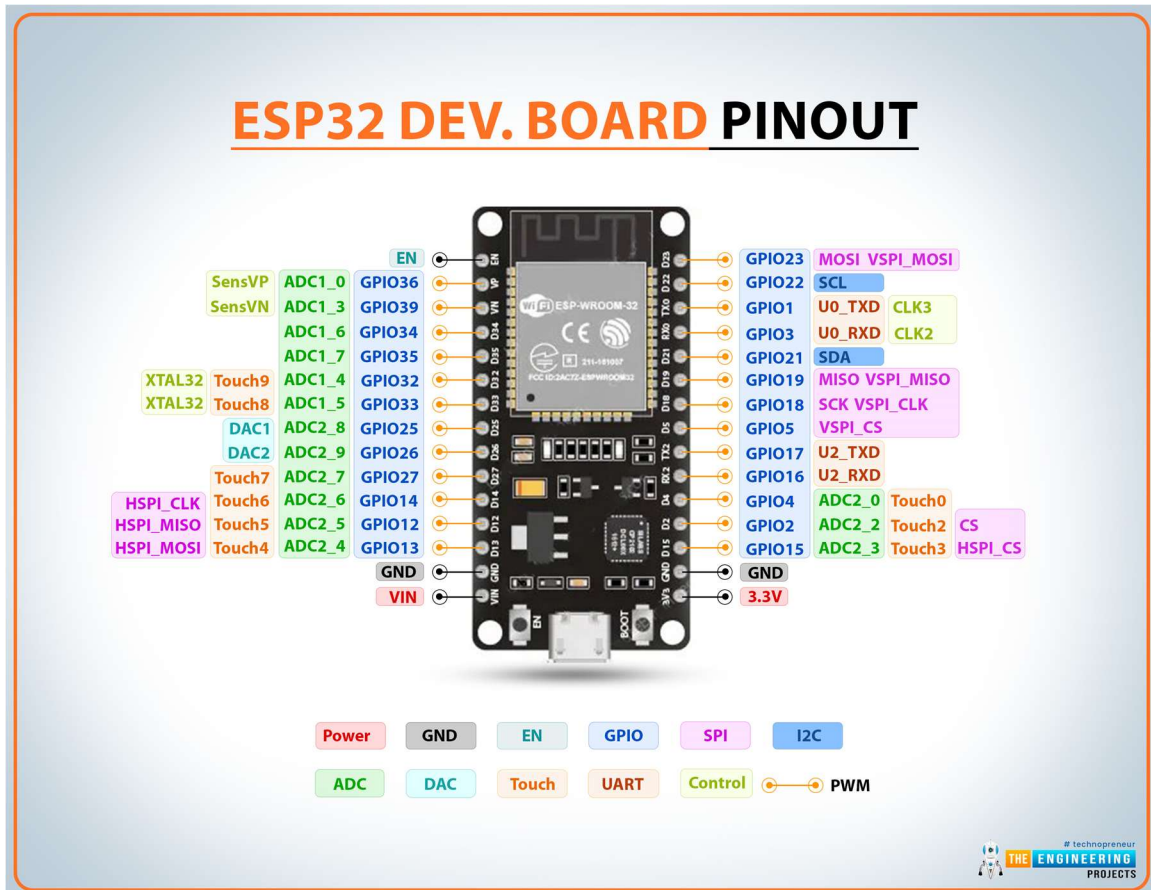
Pin	Connection
Cathode (-)	Connect to GND
Anode (+)	Connect to resistors and then to multiple viable ESP32 digital output pins

5. BLYNK SETUP

<input type="checkbox"/>	Id	Name	Pin	Color	Data Type	Units	Is Raw	Min	Max	Actions
	1	MQ2 OUTPUT	V0		Integer		false	0	4041	
	2	ALARM CONTROL	V1		Integer		false	0	1	
	3	SYSTEM_STATYS	V2		String		false			
	4	THRESHOLD	V3		Integer		false	844	4040	



6. ESP32 Pin Functions



Software & Tools

PlatformIO: (Development environment)

Visual Studio Code: (Code editor)

Wokwi Extension: (Simulation and testing)

Blynk IoT Dashboard: (Cloud-based monitoring)

Installation & Setup

1. Run an online simulation here:

<https://wokwi.com/projects/426166087854034945>

2. Or to run a local simulation, install Vscode and follow the steps below:

<https://code.visualstudio.com/download>

3. Clone the Repository

- `git clone https://github.com/ishola-faazele/LPG_DETECTION_MICROP.git`

4. Install PlatformIO: Install the PlatformIO extension in VS Code.

5. Open the Project in PlatformIO

- Open the project folder in VS Code and ensure PlatformIO is installed.

6. Connect Blynk

- Set up a new Blynk project.

- Obtain the authentication token.

- Update the `main.cpp` file with your Blynk credentials.

7. Compile and Upload Code

- `pio run --target upload`

8. Monitor Serial Output

- `pio device monitor`

File Structure

- `src/main.cpp`: Contains the core logic for sensor reading, alarm activation, and Blynk integration.
- `diagram.json`: Wokwi simulation setup file.
- `.pio\build\esp32dev\firmware.bin`: [generated after building] Contains compiled binary
- `.pio\build\esp32dev\firmware.hex`: [generated after building] Contains compiled firmware
- `platformio.ini`: Contains project configurations.
- `wokwi.toml`: Contains wokwi configurations.
- `schematic.png`: Schematic diagram of the project.

Usage

1. Run the simulation
2. Observe the LPG level on the OLED display.
3. If the gas level exceeds the threshold, the buzzer will sound, and the LED bar will blink.
4. Open the Blynk app to view LPG levels and adjust the threshold remotely.

Conclusion

The **LPG leak detection** system successfully addresses the problem of detecting and alerting individuals to the presence of dangerous gas leaks.

Key Conclusions:

1. **Effectiveness:** The system accurately detects gas leaks and alerts both ****locally**** (via LED/buzzer) and **remotely** (via Blynk).
2. **Real-Time Monitoring:** With the integration of **Blynk**, users can monitor gas levels remotely, adding an extra layer of safety.

3. **Reliability:** The combination of **MQ2 sensor, ESP32, OLCD** and **Blynk** works seamlessly, providing a robust solution for detecting and responding to gas leaks.

Challenges

1. Ensuring the **accuracy** of the **MQ2 sensor readings**, as environmental factors can influence sensor output.
2. Fine-tuning the **threshold value** to minimize false alarms while ensuring safety.

Future Improvements

1. Implement Wi-Fi notifications (email/SMS alerts).
2. Improve sensor calibration for higher accuracy.
3. Add a mobile app for more advanced remote control.

Credits and Reference

Developed as part of a microprocessors course using PlatformIO, VS Code, and Wokwi simulation.

- <http://theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html>
- <https://examples.blynk.cc/>
- <https://docs.espressif.com/projects/esp-idf/en/v5.1/esp32/hw-reference/esp32/get-started-devkitc.html>
- <https://docs.arduino.cc/>
- <https://docs.wokwi.com/guides/esp32>
-