ESP32 BASED GAS LEAK DETECTION SYSTEM

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
for the course Design Thinking and Innovation- SCSBDPROJ

by

HARSHIDA A (Reg.No -43110286)

`

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
CATEGORY - 1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC I Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

APRIL - 2025

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
CATEGORY - 1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC I Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that Design Thinking and Innovation- SCSBDPROJ is the bonafide work of **HARSHIDA A(43110286),HARIPPRIYA(43110284),J.SUREKHA(43110297),HARSHINI SUBRAMANI(43110287),HEJASVINI M(43110288),JAISHA J(43110294)** who carried out the Design Product entitled **"ESP32 BASED GAS LEAK DETECTION SYSTEM**" as a team under my supervision from January 2025 to April 2025.

Design Supervisor
Dr. R. RAHIN BATCHA., M.Tech., Ph.D.M

Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

I **HARSHIDA A(43110286) ,** hereby declare that the Design Product Report entitled **"ESP32 BASED GAS LEAK DETECTION SYSTEM"** done by me under the guidance of **Dr. R. RAHIN BATCHA, M.Tech., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE:

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph. D**, **Dean**, School of Computing, and **Dr. L. LAKSHMANAN M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R. RAHIN BATCHA, M.Tech., Ph.D** for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my Design Product.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the Design Product work.

ABSTRACT

This project introduces an ESP32-based gas leak detection system designed to provide timely and reliable warnings in the event of a gas leak. Gas leaks pose significant dangers, including fire hazards, explosions, and health risks, highlighting the critical need for effective detection systems. Current gas detection solutions often suffer from limitations such as high costs, complex installation procedures, delayed response times, lack of portability, and high power consumption. To address these shortcomings, the proposed system leverages the ESP32 microcontroller and the MQ-2 gas sensor to offer a cost-effective, compact, and efficient alternative. The system's key features include accurate gas detection, real-time alert generation via a buzzer, and low power consumption. The MQ-2 sensor measures gas concentration and sends an analog voltage to the ESP32. The ESP32 processes this data, compares it to a predefined threshold, and activates the buzzer to emit an audible alarm if the threshold is exceeded. This system offers several advantages, including cost-effectiveness, simplicity in design and deployment, fast response time, reliability, and ease of understanding. Potential applications span across home safety (kitchens, basements), industrial settings (factories, warehouses), gas storage facilities, chemical laboratories, and even recreational vehicles and boats. Future work may include adding wireless connectivity for remote monitoring, incorporating data logging capabilities, integrating with smart home systems, and implementing self-calibration features for the sensor.

.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In recent years, the need for smart safety systems has significantly increased, especially in environments where the risk of gas leakage poses a serious threat to human life and property. The ESP32 Gas Leak Detection System is an Internet of Things (IoT)-based solution designed to detect the presence of flammable or toxic gases such as LPG (Liquefied Petroleum Gas), methane, propane, or carbon monoxide. This system not only detects gas leakage but also provides real-time alerts to prevent accidents and ensure safety.

At the core of this system is the ESP32 microcontroller, a powerful, energy-efficient module that offers both Wi-Fi and Bluetooth connectivity. Its dual-core processor, multiple GPIO pins, and built-in analog-to-digital converter (ADC) make it ideal for sensor integration and real-time data processing. The system uses gas sensors (commonly MQ-series such as MQ-2, MQ-5, or MQ-135) to sense the concentration of gases in the environment. When the gas concentration exceeds a predefined threshold, the ESP32 triggers alarms, notifies users through mobile apps or web interfaces, and can even activate safety measures such as turning off gas valves or enabling exhaust fans.

1.1 The key features of this system include:

- Real-time monitoring of gas concentration levels.
- Wi-Fi-based remote notifications through platforms like Blynk, Telegram, or Firebase.
- On-device alerts such as buzzers, LEDs, or display messages.
- Low-cost and customizable hardware setup, suitable for home, industrial, and laboratory applications.
- Compact and energy-efficient design, ideal for long-term deployments.

1.2 Objectives:

1.2.1 Detect Hazardous Gas Leaks in Real-Time:

To monitor the surrounding environment continuously and identify the presence of harmful gases (e.g., LPG, methane, carbon monoxide) using suitable gas sensors (such as MQ-series).

1.2.2 Ensure Timely Alerts and Notifications:

To provide instant alerts via buzzer, LED, or notifications (SMS, email, or mobile app) when gas levels exceed safe thresholds, helping prevent accidents.

1.2.3 Implement Wireless Communication with ESP32:

To utilize the ESP32 microcontroller's built-in Wi-Fi and/or Bluetooth features for sending alerts to users remotely through cloud services or mobile devices.

1.2.4 Log Sensor Data for Analysis:

To record gas concentration levels periodically and store them on a cloud platform or local storage for later review, analysis, and pattern tracking.

1.2.5 Incorporate Fail-Safe Mechanisms:

To implement safety features such as automatic shutdown of appliances or ventilation system activation when a gas leak is detected.

This project demonstrates how embedded systems and IoT technologies can be effectively used to develop intelligent safety applications. With increasing concerns about safety and automation, the ESP32 Gas Leak Detection System stands as a reliable, scalable, and user-friendly solution for gas leak prevention and early warning mechanisms.

CHAPTER 2

PROBLEM STATEMENT

In many households, industries, and commercial kitchens, the leakage of combustible gases like LPG (Liquefied Petroleum Gas) poses a serious threat to safety, potentially leading to fire hazards, explosions, or health complications. Despite the availability of gas sensors and safety measures, incidents still occur due to delayed detection or lack of real-time alerts.

To address this issue, there is a need for a low-cost, efficient, and real-time gas leak detection system that can alert users immediately when a leak is detected. The system should be capable of remote monitoring and should provide multi-channel alerts (buzzer, SMS, mobile notifications, etc.) to ensure rapid response and preventive action.

This project aims to develop a Gas Leak Detection and Alert System using the ESP32 microcontroller, which utilizes gas sensors (like MQ-2 or MQ-5), and integrates with Wi-Fi to send alerts via IoT platforms or mobile apps. The system will enhance safety by enabling early detection and fast notification, thereby reducing the risk of accidents caused by gas leaks.

CHAPTER 3

EXISTING MODEL

3.1 Observations During Testing

Throughout the testing process, the following key observations were noted:

3.1.1 Sensor Warm-Up Time

- The MQ-2 sensor required an initial *preheat time of 1 to 2 minutes* after power-on to stabilize its readings.
- Readings taken before this warm-up period were inconsistent or abnormally low.

3.1.2 Varying Sensor Readings

- The sensor output fluctuated slightly even when exposed to a constant gas source.
- Fluctuation range: ~±2–5% due to natural electronic noise or minor air movements.
- Repeated readings helped average out noise and improve reliability.

3.1.3 Gas Detection

- The sensor responded positively when exposed to various gases like:
- Butane from a cigarette lighter
- LPG (from a gas cylinder connection)
- Smoke from a burning paper
- The *mapped output (0–100%) increased sharply*, especially when gas was closer to the sensor.

3.1.4 Threshold Sensitivity

- A threshold of *50%* was found to be a good starting point for alerting.
- Increasing or decreasing the threshold changed how quickly the buzzer activated:
    - Too low → false alarms
    - Too high → delayed response

3.1.5 Environmental Impact

- High humidity or strong wind near the sensor affected readings.
- Best results were observed under stable room temperature and indoor testing conditions.

3.2 Limitations :

3.2.1 Sensor Accuracy and Drift

- MQ-series gas sensors (e.g., MQ-2, MQ-135) are not highly accurate.
- Readings can vary due to temperature, humidity, or dust.
- Over time, the sensors may drift or degrade, affecting reliability.

3.2.2 Calibration Required

- Gas sensors need manual calibration for accurate detection.
- Calibration must be done in a controlled environment, which may not be feasible for all users..

3.2.3 False Alarms

- Environmental factors like smoke from cooking or strong odors can trigger false alerts.
- May lead to user desensitization, reducing the system's effectiveness over time.

3.2.4 Dependency on Internet Connection

- Remote monitoring and alerts require a stable Wi-Fi connection.
- In case of Wi-Fi failure, cloud-based alerts and data logging will not work.

3.2.5 Power Supply Limitations

- The ESP32 requires a continuous power supply.
- Power failure can disable the system unless a battery backup is added.

3.2.6 Security Risks

- If not properly secured, the Wi-Fi-enabled system could be vulnerable to hacking.
- Data transmitted to the cloud can be intercepted without encryption.

3.2.7 Limited Range of Detection

- The sensor only detects gas in its immediate surroundings.
- May not be effective in larger areas unless multiple sensors are installed.

3.3 Market comparison

The ESP32-based gas detector is a cost-effective solution, typically priced between ₹600-₹1500 ($8 - $20), offering Wi-Fi/Bluetooth connectivity for remote monitoring. However, it requires technical knowledge for assembly, programming, and calibration, and its accuracy depends on the sensor used (e.g., MQ-2, MQ-135). Calibration and environmental factors can affect performance, making it more suited for DIY projects or home use rather than professional applications.

On the other hand, commercial gas detectors are more expensive, ranging from ₹5000-₹20000 ($70 - $300). They offer high accuracy, often factory-calibrated for specific gases, and are easy to use, typically providing plug-and-play functionality. These devices are ideal for professional or industrial use, where reliability and certification (e.g., ATEX or UL) are crucial. In summary, the ESP32-based system is a budget-friendly, flexible option for enthusiasts and small-scale applications, while commercial detectors are more accurate, user-friendly, and better suited for critical environments.

CHAPTER 4

PROPOSED MODEL

4.1 System Description:

The ESP32-based Gas Leak Detection System is a simple yet effective embedded system designed to monitor the presence of harmful gases in the environment. Its primary goal is to ensure safety by detecting gas leaks early and alerting users immediately.

The system uses an MQ-2 gas sensor, which is capable of detecting gases like LPG, methane, propane, and carbon monoxide. When the system is powered on, the sensor continuously monitors the air quality and generates an analog signal that corresponds to the concentration of gas in the environment.

This signal is sent to the ESP32 microcontroller, which reads the data through its built-in analog-to-digital converter (ADC). The ESP32 processes this data and checks whether the gas concentration has crossed a predefined threshold.

- If the gas level is within the safe limit, the system continues to monitor without triggering any alert.

- If the gas concentration exceeds the safe limit, the ESP32 activates a buzzer connected to it, producing a loud alarm sound to warn people nearby.

Additionally, since the ESP32 has built-in Wi-Fi capabilities, the system can also be programmed to send real-time alerts (via apps like Blynk or email) or upload data to the cloud, enabling remote monitoring and data logging.

This system is ideal for use in homes, kitchens, laboratories, or any place where gas leakage could pose a danger. It provides a low-cost, real-time safety mechanism that helps prevent accidents and ensures peace of mind.

4.2 Block Diagram:

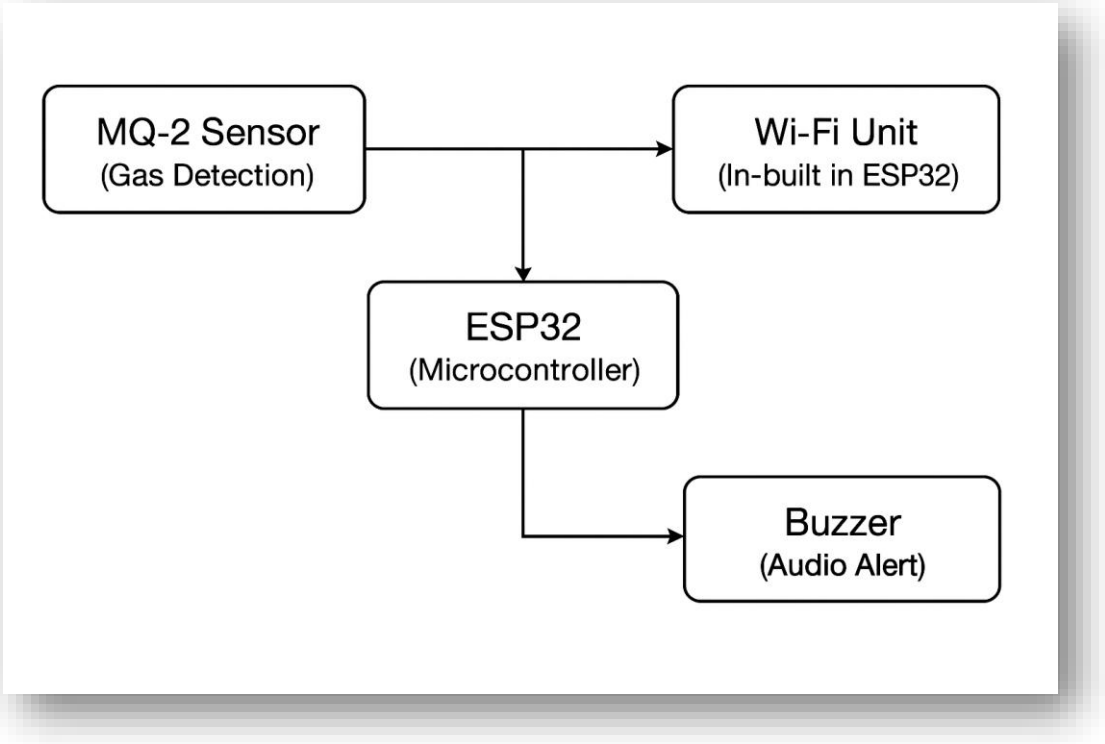

Fig 4.1 block diagram

Explanation of Components:

- ***MQ-2 Gas Sensor:***
  .1 Detects gases like LPG, methane, CO, smoke.
  .2 Analog output voltage increases with gas concentration.
  .3 Sends this analog signal to ESP32.
- ***ESP32 DevKit V1:***
  .1 Central unit controlling the system.
  .2 Reads analog value from the MQ-2 sensor.
  .3 Compares it against a predefined threshold.
  .4 If the gas level exceeds the limit, it activates the buzzer and optionally sends alerts via Wi-Fi.
  .5 Has in-built Wi-Fi support for IoT applications.

- ***5V Buzzer:***
  .1 Connected to a digital output pin of ESP32.
  .2 Emits a sound when gas concentration crosses the threshold, alerting nearby users.
- ***Wi-Fi (in-built ESP32):***
  .1 Can send alerts to users via platforms like Blynk, IFTTT, or cloud databases (e.g., Firebase, ThingSpeak).
  .2 Enables remote monitoring.

***Flow of Operation*:**

1. Power On System via USB or 5V external source.
2. MQ-2 detects gas in the environment and outputs an analog voltage.
3. ESP32 reads sensor data through an analog pin and processes it.
4. If the gas level is above threshold:
   - Activates the buzzer.
   - Sends real-time notification (optional Wi-Fi/cloud integration).
5. Continues to monitor environment in real time.

4.3 Circuit Design

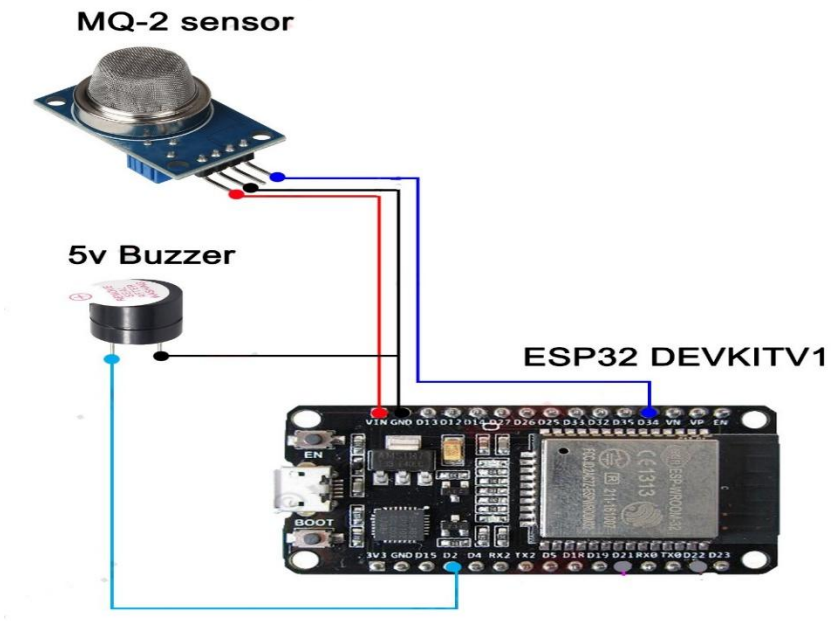

Fig 4.2 Circuit Diagram

This circuit diagram illustrates the connection of an MQ-2 gas sensor and a 5V buzzer to an ESP32 DEVKIT V1 board. The objective is to detect harmful gas levels and trigger an audible alarm when gas concentration crosses a safe threshold.

Components and Pin Connections

4.3.1 MQ-2 Gas Sensor

The MQ-2 sensor detects gases such as LPG, methane, butane, smoke, etc.

- VCC (Power Pin)– Connected to Vin on ESP32
    - The Vin pin of the ESP32 provides around 5V, which is needed for the sensor to heat up its internal coil.
- GND (Ground) – Connected to GND on ESP32
    - Provides the necessary return path for the current.
- A0 (Analog Output) – Connected to GPIO 34 on ESP32
    - Sends analog gas concentration values to the ESP32. GPIO 34 is one of the ADC (Analog to Digital Converter) pins on the ESP32.

4.3.2 5V Buzzer

Used to produce a sound alert when gas levels are high.

- (Positive Terminal) – Connected to GPIO 2 on ESP32
    - GPIO 2 is a digital output pin that sends HIGH (5V) to trigger the buzzer.
- (Negative Terminal)– Connected to GNDon ESP32
    - Completes the buzzer circuit by providing ground.

4.3.3 ESP32 DEVKIT V1

This is the central microcontroller that processes sensor readings and controls output devices.

- It reads the analog gas levels from GPIO 34.
- Based on a threshold value (like 50%), it sends a HIGH signal to GPIO 2 to turn the buzzer ON.
- Powered via USB (5V), which also supplies power to the sensor and buzzer through its Vin pin.

4.4 Working Principle :

This project is designed to detect the leakage of combustible gases such as LPG, propane, methane, and hydrogen using the MQ-2 gas sensor. It ensures safety by providing an immediate alert through a buzzer when a gas leak is detected. The system mainly comprises the ESP32 microcontroller, MQ-2 sensor, a buzzer, and is programmed using the Arduino IDE.

4.4.1 MQ-2 Gas Sensor Operation

The MQ-2 gas sensor detects combustible gases using a SnO_2 layer whose resistance drops in the presence of gas, changing the output voltage. It provides both analog and digital outputs. In this system, the analog output (A0) is connected to the ESP32 (e.g., GPIO34) to measure real-time gas concentration accurately..

4.4.2 ESP32 Microcontroller Role

The ESP32 acts as the brain of the system. It continuously reads the analog signal from the MQ-2 sensor using its ADC (Analog-to-Digital Converter) pin. Based on the analog value received, the ESP32 compares it with a predefined threshold (set in the code). If the gas concentration crosses this threshold, the ESP32 interprets it as a gas leak.

When this condition is met:

- The ESP32 sends a signal to the buzzer, activating an audible alarm to warn people nearby.
- In advanced versions, the ESP32 can also be configured to send notifications over Wi-Fi, thanks to its built-in wireless connectivity features (optional).

4.4.3 Buzzer Function

The buzzer is connected to one of the digital output pins of the ESP32. When the ESP32 detects a gas leak, it outputs a HIGH signal to this pin, triggering the buzzer. The buzzer will remain ON until the gas level falls below the threshold again or until the system is manually reset.

4.4.4 Power Supply

The system is powered using either:

- A 5V USB power supply connected to the ESP32 board.
- A battery pack or external regulated 5V supply that powers both the ESP32 and the MQ-2 sensor.

Proper voltage regulation is necessary to ensure safe operation of the components.

4.4.5 Programming via Arduino IDE

The ESP32 is programmed using the Arduino IDE, a widely used open-source platform for microcontroller programming. The required ESP32 board definitions and libraries are installed via the Board Manager in Arduino IDE.

The code involves:

- Initializing the input (sensor) and output (buzzer) pins.
- Reading analog values from the MQ-2 sensor.
- Comparing the value with the threshold.
- Activating the buzzer if the threshold is crossed.

This program runs in a continuous loop to constantly monitor the environment for gas leaks.

Summary of Operation Flow

1. System powers on → ESP32 and MQ-2 initialize.
2. MQ-2 detects gas levels → Sends analog voltage to ESP32.
3. ESP32 reads the value → Compares with threshold.
4. If gas level is safe→ System stays idle.
5. If gas level is dangerous→ Buzzer is activated for alert.

.

CHAPTER 5

FEASABILITY STUDY

5.1 Technical Feasibility

- ***Availability of Components***: The required hardware—ESP32 Devkit V1, MQ-2 gas sensor, and buzzer—are easily available and well-documented.
- ***Technology Suitability***: ESP32 supports Wi-Fi and sensor integration, making it suitable for real-time gas detection and alerting.
- ***Skill Requirement***: The project requires basic knowledge of embedded systems, IoT, and Arduino programming, which is manageable for students or hobbyists.
- ***Scalability***: Can be scaled by adding cloud integration, multiple sensors, or IoT dashboards in the future.

5.2 Economic Feasibility

- ***Low-Cost Implementation***: Estimated budget is minimal (under ₹1000–₹1500 depending on parts), making it feasible for educational, household, or small-scale industrial use.
- ***Cost-Effectiveness***: Offers an affordable solution compared to commercial gas detectors with wireless capabilities.

5.3 Operational Feasibility

- ***Ease of Use***: Once installed, the system operates automatically and alerts users without manual intervention.
- ***Maintenance***: Basic maintenance required—sensor recalibration and occasional cleaning.
- ***User Alerts***: Buzzer provides immediate local warning; remote alert options (if added) improve responsiveness.

5.4 Legal and Environmental Feasibility

- ***Safety Compliance***: The project doesn’t involve high voltage or harmful emissions, making it safe to use in homes.
- ***Gas Sensor Limitations***: While MQ-2 can detect gases like LPG and methane, it's not highly specific—safety-critical applications may require certified industrial-grade sensors.

5.5 Schedule Feasibility

- ***Time Requirements***: The project can be completed in 1–2 weeks including:
  - Circuit design & wiring – 1–2 days
  - Code development & testing – 3–4 days
  - Documentation & enhancements – 2–3 days
- ***Learning Curve***: Ideal for academic timelines or quick prototyping.

CHAPTER 6
COMPONENTS AND HARDWARE REQUIREMENT

6.1 ESP32 Microcontroller

Features:

- Dual-core Tensilica LX6 processor with clock speeds up to 240 MHz.
- Wi-Fi (802.11 b/g/n) and Bluetooth v4.2 (BLE + Classic) connectivity.
- Multiple I/O pins: Includes 34 programmable GPIOs (depending on the variant).
- ADC & DAC support: 12-bit ADC (up to 18 channels), 2 × 8-bit DAC.
- PWM, I2C, SPI, UART communication protocols supported.
- Integrated Hall sensor and temperature sensor.
- Ultra-low-power co-processor for low-energy applications.

Advantages:

- **Low Power Consumption**: Ideal for battery-powered or always-on IoT applications.
- **High Performance**: Dual-core processor ensures fast response and parallel processing capabilities.
- **Integrated Connectivity**: No need for external Wi-Fi or Bluetooth modules.
- **Cost-effective**: Offers advanced features at a very affordable price point.
- **Wide Community Support**: Rich documentation and active online forums aid in quick development.

Why It Was Chosen:

The ESP32 was selected due to its versatility and built-in Wi-Fi capabilities, which are essential for real-time gas monitoring and remote alerts. Its sufficient GPIOs allow easy interfacing with sensors and actuators like the MQ-2 sensor and buzzer. Furthermore, its low power operation and compact size make it perfect for embedded system applications.

6.2 MQ-2 Gas Sensor

***Working Principle*:**

The MQ-2 is a metal oxide semiconductor (MOS) sensor, also known as a chemiresistor. It contains a sensing layer made of tin dioxide (SnO_2) which has lower conductivity in clean air. When target gases are present, the conductivity increases due to a redox reaction on the sensor's surface. The change in resistance is measured and converted into an analog signal.

Gases It Can Detect:

- LPG (Liquefied Petroleum Gas)
- Methane (CH_4)
- Butane
- Propane
- Hydrogen (H_2)
- Smoke
- Alcohol

Features:

- Fast response time (~10s).
- High sensitivity to flammable gases and smoke.
- Analog output for concentration (can be calibrated).
- Long life and low cost.

Why It Was Chosen:

The MQ-2 sensor was chosen due to its ability to detect a wide range of combustible gases and smoke, making it ideal for safety and hazard detection systems. Its analog output simplifies integration with the ESP32, and its low cost and ready availability make it practical for prototyping and deployment.

6.3 Buzzer

Role in Alerting the User:

The buzzer acts as an audio indicator that alerts users immediately when harmful gas levels exceed a predefined threshold. It is connected to a GPIO pin on the ESP32 and is triggered via a digital signal.

Features:

- Compact and lightweight
- Low power consumption
- High-pitched sound output (~2-4 kHz)
- Can be continuously or intermittently powered for tone variations

Advantages:

- Provides a real-time audible warning, which is critical in emergency scenarios.
- Simple interfacing with microcontrollers.
- Instant feedback, no need for visual monitoring.

Why It Was Chosen:

An audio alert is one of the most effective ways to draw immediate attention to dangerous conditions. The buzzer ensures that users are warned even if they are not actively monitoring the system.

6.4 Power Supply

Power Management and Considerations:

The system is powered through a regulated 5V DC power supply, which ensures stable voltage for both the ESP32 and the MQ-2 sensor. A USB connection or a battery (e.g., Li-ion 18650) with a voltage regulator circuit (like an AMS1117) can be used.

Considerations:

- ***Current Requirements***: The ESP32 and MQ-2 both require significant startup current; thus, the power supply must provide at least 1A of current.
- ***Voltage Regulation***: The ESP32 works at 3.3V logic, so any input above this is stepped down using a low-dropout regulator.
- ***Safety***: Protection circuits like fuses or TVS diodes may be added for surge protection.

Why It Was Chosen:

A reliable power supply ensures the stable and uninterrupted operation of the system. Since gas monitoring is a critical application, power fluctuations must be avoided to prevent sensor errors or microcontroller resets.

6.5 Cost Estimation Table :

Component	Quantity	Unit Cost (INR)	Total Cost (INR)
ESP32 Microcontroller (with adapter)	1	₹700	₹700
MQ-2 Gas Sensor	1	₹200	₹200
Buzzer Pack (3–5 pcs)	1	₹50	₹50
Jumper Wires	1	₹60	₹60
Breadboard Set	1	₹120	₹120
Power Adapter	1	₹100	₹100
Batteries (with holder)	1	₹70	₹70
Total Estimated Cost			₹1300

CHAPTER 7

SOFTWARE IMPLEMENTATION

The software implementation of the ESP32 Gas Leak Detection System is done using the Arduino IDE, an open-source development environment used for writing and uploading code to the ESP32 microcontroller. This section outlines the steps involved in setting up the software, writing the program, and uploading it to the ESP32 board.

7.1 Step-by-Step Implementation:

7.1.1 Installing the Arduino IDE

- Download the Arduino IDE from the official website: https://www.arduino.cc/en/software
- Install it on your computer (Windows/Mac/Linux)

7.1.2 Installing the ESP32 Board Package

- Open Arduino IDE → Go to File → Preferences
- In the “Additional Boards Manager URLs” field, add: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp3_index.json
- Then go to *Tools → Board → Boards Manager
- Search for *ESP32* and click *Install

7.1.3 Selecting the Board and Port

- Go to Tools → Board → ESP32 Dev Module (or select your specific ESP32 board)
- Connect the ESP32 to your PC via USB
- Go to Tools → Port and select the correct COM port for your ESP32

7.1.4 Writing the Code

- The code is written in the Arduino C/C++ language
- It includes:
  - Pin configuration
  - Sensor reading using analogRead()
  - Conditional statements to activate the buzzer
  - Serial monitor for debugging and gas value display

7.1.5 Uploading the Code

- Click the Upload button (→)
- Wait for the code to compile and upload
- After successful upload, the ESP32 starts running the gas detection logic immediately

7.1.6 Serial Monitoring

- Open the Serial Monitor from Tools → Serial Monitor (or press Ctrl + Shift + M)
- Set baud rate to 115200
- View real-time gas values being detected by the MQ-2 sensor

programming in ESP32

```
#define MQ_SENSOR_PIN 34 // Analog pin connected to MQ-2 sensor
#define BUZZER_PIN 2     // Digital pin connected to the buzzer
void setup() {
   Serial.begin(115200);
   pinMode(BUZZER_PIN, OUTPUT);
   digitalWrite(BUZZER_PIN, LOW); // Ensure buzzer is off initially
}
void loop() {
   int sensorValue = analogRead(MQ_SENSOR_PIN);
   sensorValue = map(sensorValue, 0, 4095, 0, 100); // Convert to percentage (0-100)
   Serial.print("Gas Level: ");
```

```
    Serial.print(sensorValue);
    Serial.println("%");
    if (sensorValue >= 50) { // Threshold for high gas level
      digitalWrite(BUZZER_PIN, HIGH);
      Serial.println("⚠ ALERT: High Gas Level Detected!");
    } else {
      digitalWrite(BUZZER_PIN, LOW);
      Serial.println("Gas Level Normal");
    }
    delay(1000); // Wait 1 second before next reading
}
```

7.2 Step-by-Step Logic:

1. Start the program
2. Define hardware connections
    - Set MQ_SENSOR_PIN to analog pin 34 (connected to MQ-2 sensor)
    - Set BUZZER_PIN to digital pin 2 (connected to buzzer)
3. Initialize Serial Communication at 115200 baud rate
4. Configure BUZZER_PIN as OUTPUT
5. Set the buzzer to LOW (OFF initially)
6. Enter the main loop (loop()):
    - Read gas sensor value from MQ_SENSOR_PIN using analogRead()
    - Map the sensor value from the range (0–4095) to percentage (0–100)
7. Display the gas level percentage on the Serial Monitor
8. Check if gas level is above or equal to threshold (50%)
    - If true:
        - Turn ON the buzzer using digitalWrite(HIGH)
        - Print a high gas level warning message
    - Else:
        - Turn OFF the buzzer using digitalWrite(LOW)
        - Print "Gas Level Normal"
9. Wait for 1 second using delay(1000)
10. Repeat from Step 6 continuously

CHAPTER 8

TESTING CALIBRATION

Proper testing and calibration are critical to ensure the accuracy, sensitivity, and reliability of the ESP32 Gas Leak Detection System. The MQ-2 gas sensor must be tested with various gases and environmental conditions to calibrate it effectively for real-world usage.

8.1 Testing with Different Gases

The MQ-2 sensor is capable of detecting various combustible gases such as:

- Liquefied Petroleum Gas (LPG)
- Methane (CH_4)
- Propane (C_3H_8)
- Hydrogen (H_2)
- Alcohol vapors
- Smoke

Procedure:

- Expose the MQ-2 sensor to small controlled quantities of each gas.
- Observe the analog output readings on the serial monitor.
- Ensure proper ventilation and safety measures during this test.
- Record sensor readings for each gas and note the percentage mapped value.

Objective:

Verify that the sensor correctly detects each type of gas and responds with a proportional output.

8.2 Response Time Measurement

Response time refers to how quickly the sensor detects gas after exposure and how fast it reacts to falling gas levels.

Procedure:

- Use a stopwatch to measure the time between the moment gas is introduced and the buzzer is activated.
- Similarly, measure the time taken for the buzzer to turn off after gas is removed.
- Conduct multiple trials to get an average.

Typical Response Time:

- MQ-2 sensors generally have a response time of *10 to 60 seconds* depending on gas concentration.

Objective:

- Determine the responsiveness of the sensor for real-time alerting.

8.3 Accuracy Test

Accuracy defines how close the sensor's readings are to the actual gas concentration.

Procedure:

- Compare the MQ-2 sensor's output against a commercial gas detector or gas leak analyzer.
- Test under stable environmental conditions (room temperature, no wind, no dust).
- Take multiple readings and compare the average.

Observation:

- The MQ-2 may not give exact PPM (parts per million) values without advanced calibration but shows reliable percentage trends.

Objective:

- Ensure that the system detects and warns at appropriate gas levels even if the exact concentration is unknown.

8.4 Sensitivity Test

Sensitivity is the sensor’s ability to detect even small amounts of gas.

Procedure:

- Gradually release a small amount of gas from a lighter (butane) or LPG valve from a safe distance.
- Measure how small a quantity the sensor can detect before triggering the buzzer.
- Adjust the threshold in the code if needed to increase or decrease sensitivity.

Adjustment Tip:

Lowering the threshold value (e.g., from 50 to 40) makes the system more sensitive.

Objective: Fine-tune the sensor threshold to balance between false alarms and early warnings.

CHAPTER 9

APPLICATION

The ESP32-based Gas Leak Detection System has a wide range of practical applications across various environments due to its real-time monitoring and alert capabilities. Some of the key applications include:

9.1 Home Safety

- Kitchens: Detects LPG or cooking gas leaks, helping prevent household fires or explosions.
- Basements: Monitors buildup of gases like methane which can be hazardous in enclosed spaces.

9.2 Industrial Safety

- Factories: Ensures worker safety by monitoring for combustible or toxic gases in industrial environments.
- Warehouses: Especially where chemicals or fuel-based goods are stored, early gas detection can prevent large-scale hazards.

9.3 Gas Storage and Chemical Laboratories

- Used to monitor gas leakage from cylinders or containers storing flammable or toxic substances.
- Prevents accidents in research labs and chemical processing areas.

9.4 Mobile Environments

- Boats and Ships: Detects gas buildup in engine rooms or cabins, enhancing marine safety.
- Recreational Vehicles (RVs): Provides real-time monitoring in mobile homes where gas is used for cooking or heating.

CHAPTER 10

CONCLUSION

The ESP32 Gas Leak Detection System presents a modern, efficient, and cost-effective solution for ensuring safety against hazardous gas leaks in residential, commercial, and industrial environments. By leveraging the powerful features of the ESP32 microcontroller, such as built-in Wi-Fi and GPIO support, along with reliable gas sensors like the MQ series, the system enables real-time gas level monitoring and instant alert notifications

This project demonstrates how IoT technology can be effectively integrated into safety applications. With the ability to send alerts via Wi-Fi to smartphones or cloud platforms, users can respond quickly to dangerous situations, potentially preventing accidents, fires, or explosions. The system is also highly customizable and scalable, making it adaptable to a wide range of use cases.

However, the system is not without limitations. It depends on stable internet connectivity, accurate sensor calibration, and continuous power supply. Despite these challenges, the core concept provides a solid foundation for developing more advanced and intelligent safety systems.

Looking forward, the ESP32-based gas leak detection system has the potential to evolve into a fully automated smart safety solution, with features like automatic gas shut-off, multi-gas sensing, cloud analytics, and AI-based prediction models. Such advancements would not only increase efficiency but also enhance overall safety and reliability.

In conclusion, this system highlights the importance and practicality of incorporating IoT-based solutions in day-to-day safety applications, and it lays the groundwork for future enhancements toward smarter and safer living and working environments.

CHAPTER 11

REFERENCE

Here are some **recommended reference books** for working with **ESP32-based gas leak detection systems**—covering ESP32 programming, sensor interfacing (like MQ-series gas sensors), and real-time IoT applications:

1. "Internet of Things with ESP32" by Neil Cameron

- Focuses on ESP32 development using Arduino IDE and MicroPython.
- Includes sensor interfacing examples useful for gas sensors.

2. "Internet of Things Projects with ESP32" by Agus Kurniawan

- Real-world projects using ESP32 for smart monitoring and control.
- Covers sensor data reading, Wi-Fi, cloud integration, etc.

3. "Getting Started with the Internet of Things" by Cuno Pfister

- Though not ESP32-specific, this book introduces IoT basics and sensor networks.
- Helpful for understanding how gas leak detection fits in IoT architecture.

4. "Mastering ESP32" by Dogan Ibrahim

- A more in-depth, advanced guide to ESP32 programming and interfacing.
- Includes ADC, I2C, UART — all of which are useful when using gas sensors like MQ-2.

5. "Practical ESP32 Development" by Warren Gay

- Great for developers using ESP-IDF (the official ESP32 SDK).
- Good for low-level programming, real-time data handling.

These references should provide a solid foundation for understanding the current landscape and advancements in automated transcriptive systems for medical records.

The article **https://srituhobby.com/how-to-make-a-gas-level-monitoring-system-with-esp32-board/** provides a step-by-step guide to building a gas leak detection system using an ESP32 and an MQ gas sensor, complete with circuit diagrams and code examples.

APPENDIX

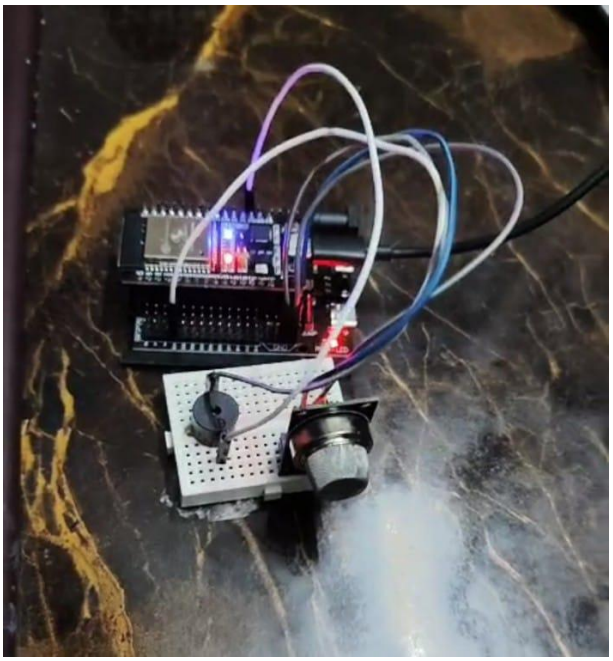

Fig 12.1 PICTURE OF THE SYSTEM

PROGRAM

```
#define MQ_SENSOR_PIN 34 // Analog pin connected to MQ-2 sensor
#define BUZZER_PIN 2     // Digital pin connected to the buzzer
void setup() {
   Serial.begin(115200);
   pinMode(BUZZER_PIN, OUTPUT);
   digitalWrite(BUZZER_PIN, LOW); // Ensure buzzer is off initially
}
void loop() {
   int sensorValue = analogRead(MQ_SENSOR_PIN);
   sensorValue = map(sensorValue, 0, 4095, 0, 100); // Convert to percentage
(0-100)
   Serial.print("Gas Level: ");
   Serial.print(sensorValue);
   Serial.println("%");
   if (sensorValue >= 50) { // Threshold for high gas level
```

```
        digitalWrite(BUZZER_PIN, HIGH);
        Serial.println("⚠ ALERT: High Gas Level Detected!");
    } else {
        digitalWrite(BUZZER_PIN, LOW);
        Serial.println("Gas Level Normal");
    }
    delay(1000); // Wait 1 second before next reading
}
```

SAMPLE GAS SENSOR READING :

Time (HH:MM:SS)	Gas Sensor Reading (0-100%)	Gas Level Status	Buzzer Status
10:00:00	20	Safe	OFF
10:00:05	25	Safe	OFF
10:00:10	30	Safe	OFF
10:00:15	55	Danger	ON
10:00:20	60	Danger	ON
10:00:25	40	Safe	OFF