College of Engineering – Department of Electronics Engineering

Group 7:
Magboo, Ashton Leenard B.
Mendoza, Lex Angelo F.
Uy, Joshua Luigi U.

Spoil Alert: An Intelligent System for Monitoring Food Freshness

Introduction:

In today's world, food safety and preservation have become more critical than ever. With the growing global concern over food waste, health risks from spoiled food, and the demand for smarter kitchen solutions, there is a pressing need for innovative technologies that can address these issues effectively. One such innovation is the development of a smart food cover that is capable of detecting food spoilage in real-time. This kitchen gadget leverages the power of sensor technology and Internet of Things (IoT) integration to revolutionize the way we store and monitor food.

Traditionally, people have relied on visual inspection, smell, or expiration dates to determine whether food is still safe to consume. However, these methods are often unreliable, subjective, and unable to account for actual microbial or chemical changes in the food. In contrast, a smart food cover equipped with advanced sensors offers a more scientific and accurate approach. By detecting specific spoilage indicators—such as gas concentration levels like ammonia or methane and changes in humidity—the smart cover provides real-time insights into the freshness and safety of the food it protects. This information can then be communicated to the user through visual alerts or wireless notifications to a connected smartphone.

Beyond its core functionality, the smart food cover also promotes a more sustainable lifestyle. By helping users identify spoiled food before it becomes a health hazard, it minimizes the risk of foodborne illnesses. At the same time, it reduces unnecessary food disposal, thereby contributing to efforts against food waste—a major environmental and economic issue globally. The integration of this technology into everyday kitchen practices represents a significant step toward smarter, safer, and more efficient food management.

College of Engineering – Department of Electronics Engineering

This project aims to explore the design, development, and implementation of a smart food cover that not only detects spoilage but also seamlessly fits into modern smart kitchens. It combines disciplines such as sensor technology, embedded systems, and wireless communication to deliver a user-friendly, efficient, and highly functional solution to an age-old problem. As households and industries alike move toward automation and digitalization, the smart food cover emerges as a timely innovation with the potential to transform food storage practices for the better.

General Objectives:

- To develop a food spoilage detection system that uses smart sensors to precisely track and identify indicators of food deterioration and notify users in real time, ensuring food safety and cutting down on waste.

Specific Objectives:

- To promote food safety and reduce waste by providing timely alerts to users regarding the freshness of stored food items.
- To provide households with an affordable and effective solution for detecting food spoilage, preventing foodborne illnesses, and encouraging better food management.
- To demonstrate the benefits of microcontrollers and IoT in improving food safety, minimizing food waste, and promoting smart technology for the community.
- To enhance user-friendliness, ensuring that the system is intuitive and accessible for all users, regardless of technical expertise.

Project Components:

1. **Gas Sensors (MQ-4, MQ135)** – Detects gases emitted by spoiled food (ammonia, methane, hydrogen sulfide).
2. **DHT11 Sensor** - A low-cost digital sensor that measures temperature and humidity.
3. **Microcontroller (ESP32)** – Processes sensor data and triggers alerts.
4. **I2C LCD** – Indicates values gathered by the sensors..
5. **Food-Grade Plastic or Silicone Cover** – Ensures safety and compatibility with different food types.

6. **Power Source(3.7 lithium ion battery)** – Provides continuous operation.
7. **MT3608** - a DC-DC boost converter module that increases (or "boosts") a lower voltage input (like 2V–4.5V from a battery) to a higher output voltage (up to 28V, adjustable).
8. **Switch -** For on/off purposes
9. **Mobile App** (Blynk) – Displays real-time spoilage data and notifications.
10. **Resistors** - Limits and regulates current flow.

4C's:

- **Cost**

The SpoilAlert device's development will be moderately expensive, mostly because it requires food-safe materials and integrates several electronic components. The biggest costs will be associated with finding trustworthy sensors and microcontrollers. Prototyping, testing, house design, and possibly creating a companion mobile application might all result in extra expenses.

- **Constraints**

1. **Sensor Sensitivity -** Gas sensors may require calibration and may be sensitive to other household gases.
2. **Power Source Limitations -** Limited battery life if the cover is expected to work long-term without charging.
3. **Size & Flexibility -** Needs to fit over various container types while housing all electronic components securely.
4. **Hygiene & Safety -** Materials must be food-grade, heat-resistant, and easy to sanitize.

- **Considerations**

1. **User Interface:** Simple and intuitive status indicators or mobile app integration.
2. **Modular Design:** Sensor unit could be detachable to allow cleaning of the cover separately.
3. **Environmental Impact:** Use of eco-friendly, recyclable materials where possible.
4. **Customizable Sizes:** Offer multiple cover sizes or stretchable designs for different containers.

- **Concept**

SpoilAlert is an automated food cover that has sensors built into it that can identify early signs of spoiling. The device keeps an eye on conditions like gas emissions from food that is breaking down which are indicators on food spoilage. By using wireless notifications and LED indicators to notify users when food is no longer fresh, SpoilAlert helps to improve food safety and minimize food waste.

Block Diagram:

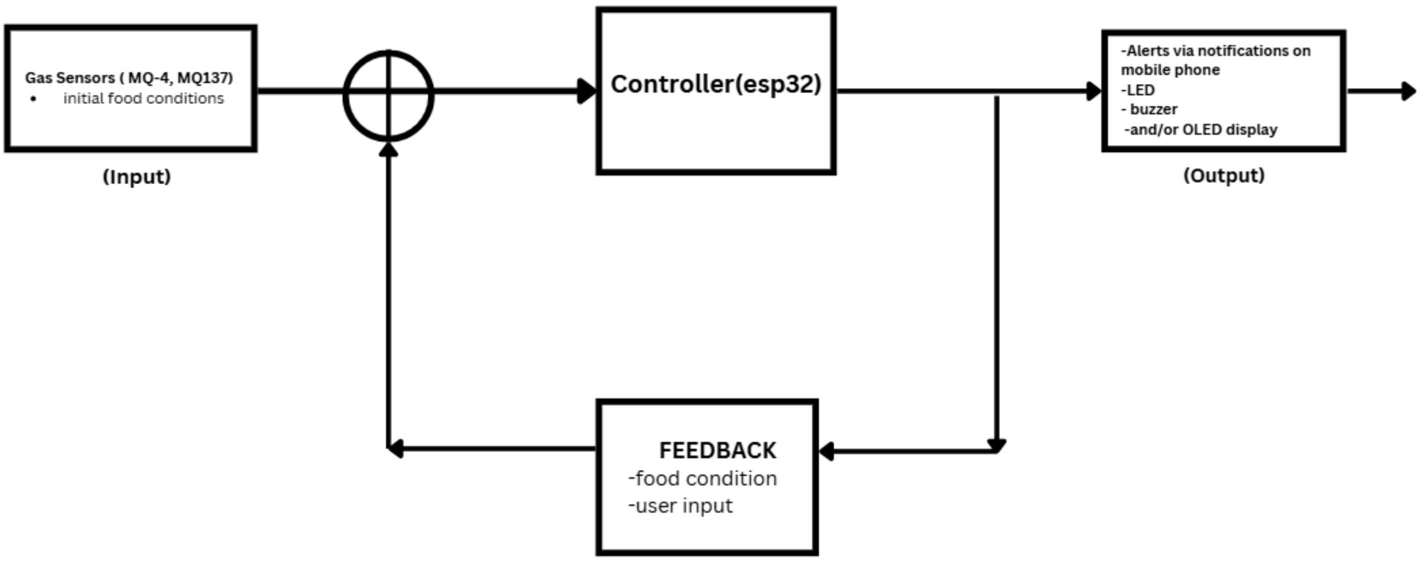


Figure 1: Block Diagram of Spoil alert

This block diagram shows a simple gas detection system designed to monitor food quality using sensors and alerts. The main parts include gas sensors (MQ-4 and MQ-135), an ESP32 microcontroller, and output devices like a mobile phone, LED, and OLED display. There's also a feedback loop, which helps the system improve over time.

First, the gas sensors (MQ-4 and MQ-135) detect harmful gases like methane and ammonia, which are often released when food starts to spoil. These sensors send signals to the ESP32 microcontroller, which acts like the brain of the system. The ESP32 checks the gas levels and decides whether they are too high.

If the gas levels are unsafe, the ESP32 triggers output alerts. These alerts can be notifications on a mobile phone, a blinking LED, or a message on an OLED display. This way, users are warned immediately if the food is going bad.

Finally, the feedback loop allows the system to learn and adjust. For example, it might recalibrate the sensors based on past data or let users confirm alerts to reduce false alarms.

Flow Chart:

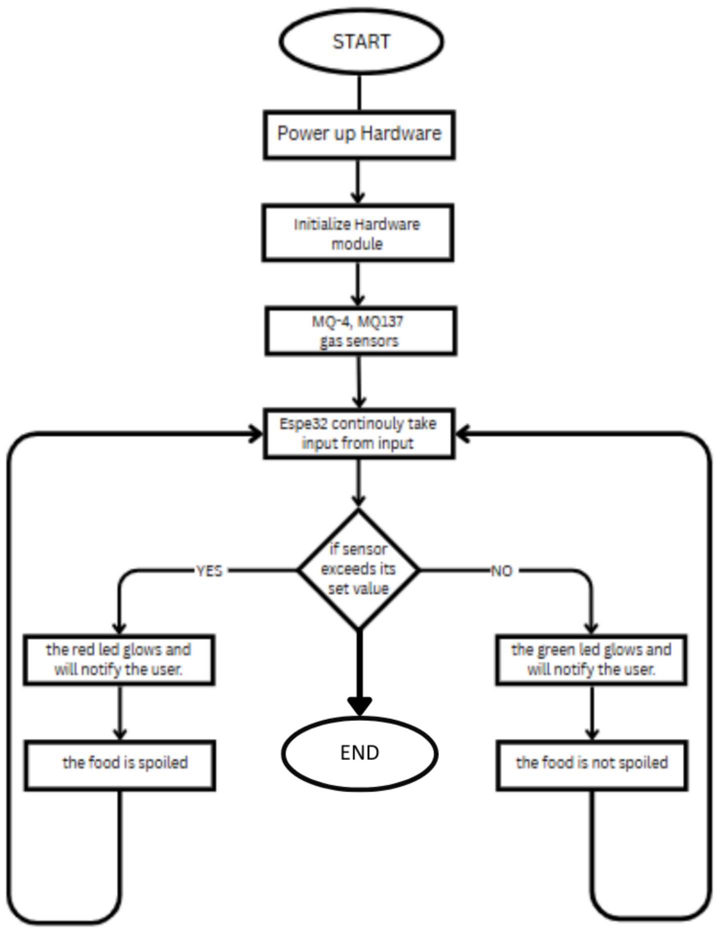


Figure 2: Flow chart of Spoil alert

The flowchart illustrates the operational process of a food spoilage detection system that utilizes gas sensors and an ESP32 microcontroller. The process begins when the system is powered on, activating all necessary hardware components. After powering up, the hardware module is initialized, which includes setting up the MQ-4 and MQ-135 gas sensors. These sensors are responsible for detecting gases such as methane and ammonia, which are commonly released by spoiled food.

Once the sensors are initialized, the ESP32 microcontroller continuously collects input data from them. It then evaluates the gas concentration levels by comparing them to a predefined threshold value. If the sensor readings exceed this threshold, the system interprets this as a sign of food spoilage. In this case, the red LED is activated to alert the user, indicating that the food is spoiled. On the other hand, if the sensor readings remain below the threshold, the system concludes that the food is still fresh. It then activates the green LED to notify the user that the food is not spoiled.

This process repeats in real time, ensuring continuous monitoring of food freshness. The system provides a straightforward and efficient method for detecting spoilage, helping users reduce food waste and maintain food safety.

Picture Diagram:

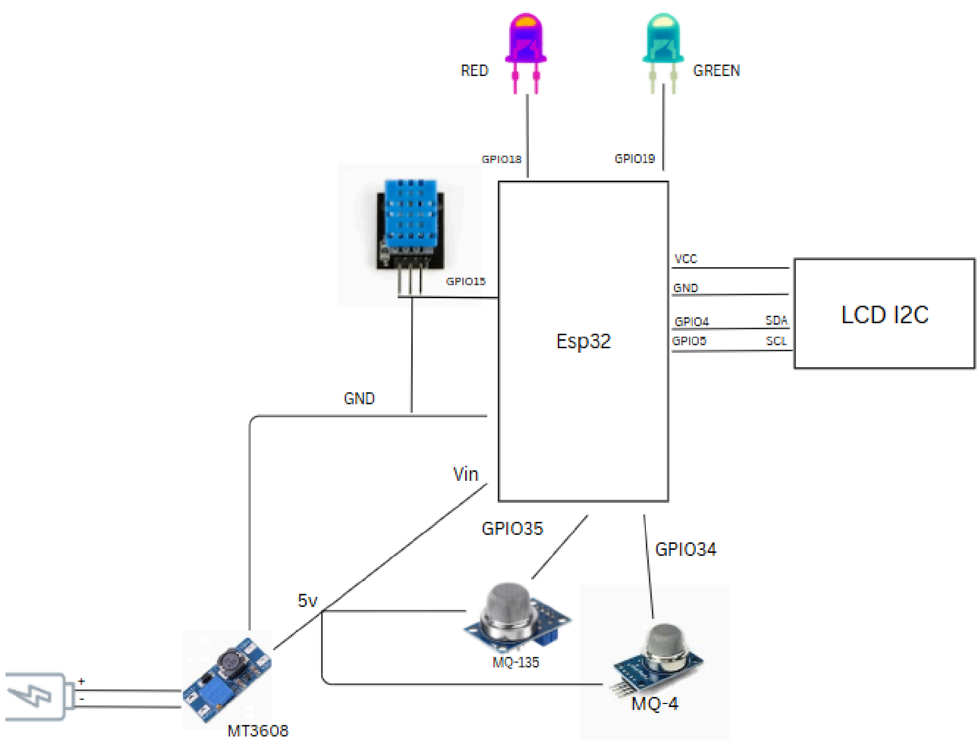


Figure 3: Picture Diagram circuit of Spoil alert

This picture diagram shows the connection of components in the Spoil Alert circuit. It includes an ESP32 microcontroller connected to various sensors and output devices. The DHT sensor is wired to GPIO15 to monitor temperature and humidity, while the MQ-135 and MQ-4 gas sensors are connected to GPIO35 and GPIO34 to detect harmful gases. A voltage booster module (MT3608) is used to supply a stable 5V to the circuit. Two LEDs are included: a red LED connected to GPIO18, which glows to notify the user of spoilage, and a green LED on GPIO19 for normal conditions. Additionally, an LCD display using the I2C protocol is connected via GPIO4 (SDA) and GPIO5 (SCL) to display sensor readings and alerts. The diagram provides a clear overview of how each component is integrated to form a functional spoil detection system.

Schematic Diagram:

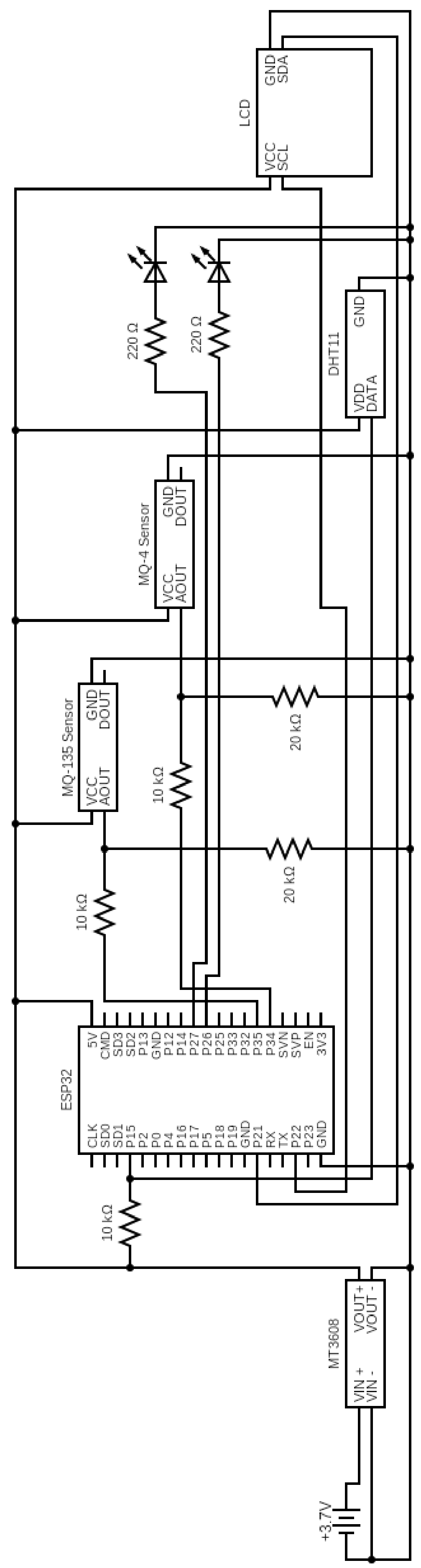


Figure 4: Schematic Diagram for SpoilAlert

College of Engineering – Department of Electronics Engineering

This schematic diagram represents a food freshness monitoring system designed to detect early signs of spoilage using various environmental sensors, an ESP32 microcontroller, and visual indicators. The system is powered by a 3.7V lithium-ion battery, which is connected to an MT3608 boost converter. This module steps up the voltage to 5V, supplying power to all components, including the sensors, LCD display, and ESP32.

At the core of the system is the ESP32, which processes input from multiple sensors. Two gas sensors, the MQ-135 and MQ-4, are used to detect harmful gases commonly released by spoiled food, such as ammonia, methane, and alcohol vapors. These sensors are connected to the ESP32 through analog output pins, each accompanied by a 10kΩ resistor to stabilize the signal. A DHT11 sensor is also included to monitor the surrounding temperature and humidity, which are important factors that influence the rate of food spoilage. The DHT11 connects to the ESP32 through a single digital data line.

For real-time status updates, the system uses an I2C LCD display module connected to the ESP32 via the SCL and SDA lines. This allows it to show gas levels, temperature, humidity, and possibly spoilage alerts. Additionally, two LEDs are connected to the ESP32 through 220Ω resistors, serving as visual indicators—likely to signify fresh (green LED) or spoiled (red LED) food conditions.

Voltage divider circuits composed of 10kΩ and 20kΩ resistors are used, potentially to scale analog signals to levels suitable for the ESP32's ADC inputs. Pull-up resistors are also in place to ensure reliable communication and signal integrity. Overall, the ESP32 gathers sensor data, evaluates it against defined thresholds, activates LEDs for visual alerts, and updates the display accordingly, making it an effective smart system for monitoring food freshness.

Wokwi simulation :

https://wokwi.com/projects/431264055343361025?fbclid=IwY2xjawKcqKRleHRuA2FlbQIxMQABHjrwEsiuOwP_3bdNTuWAMqrp9jfcWoZguFCLo8KLgINjpniuJb5_zRLuiBCR_aem_bHYU8PC5qN-NpHi8vHAj1A

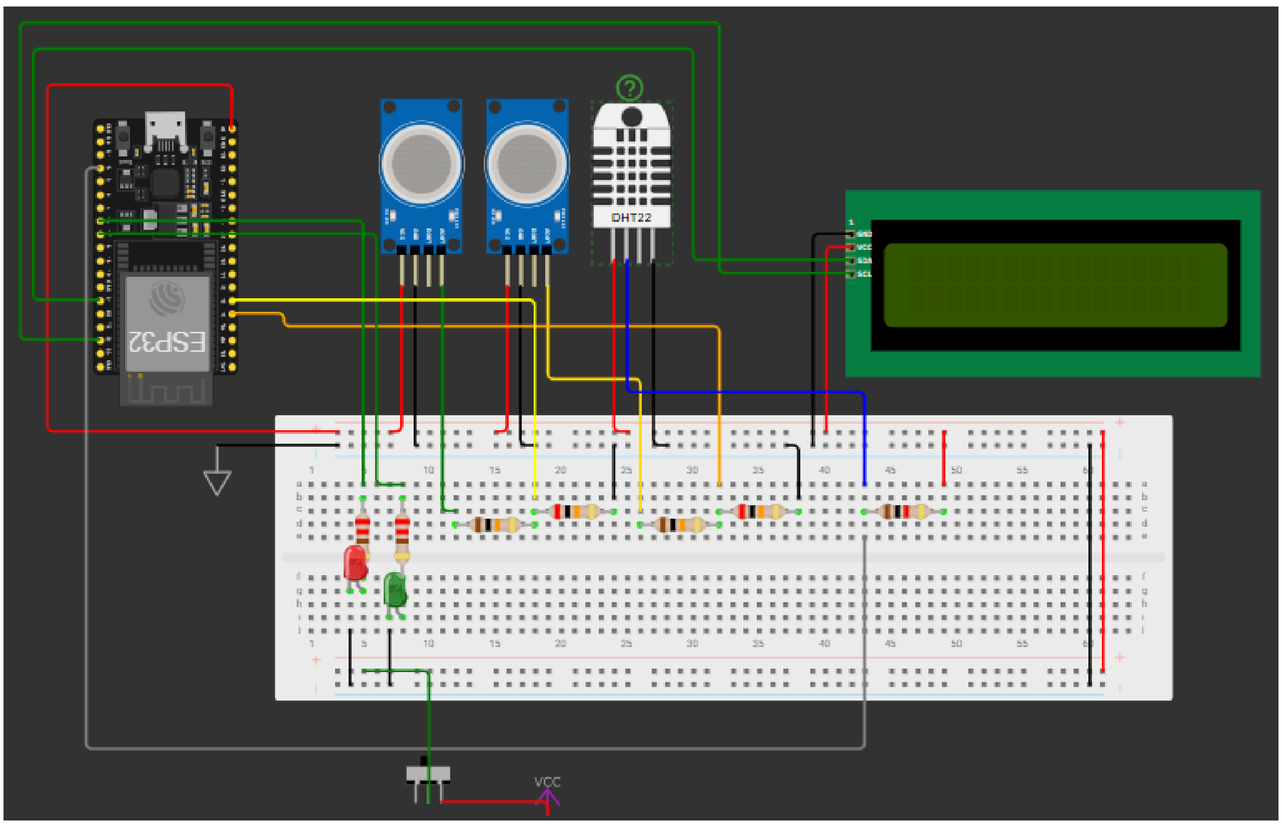
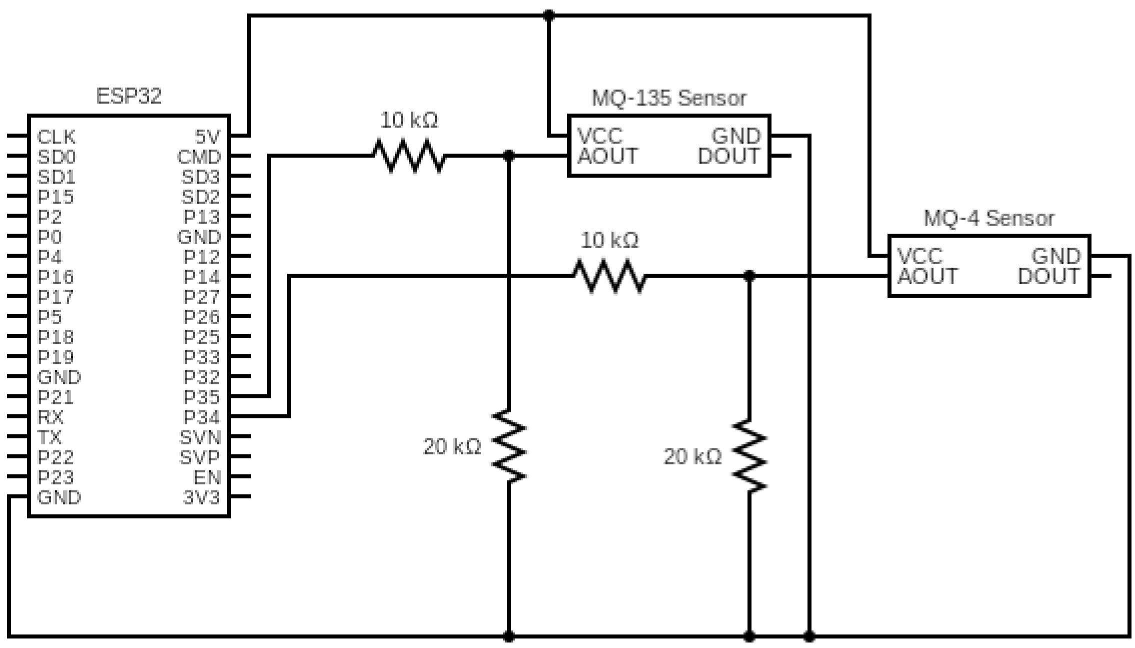


Figure 5: Picture Diagram for SpoilAlert

MQ sensor to esp32 Voltage divider:



In this schematic, the voltage divider is composed of 10kΩ and 20 kΩ resistors. It serves a key role in scaling down the sensor output voltage before it reaches the analog input pins of the ESP32.

The ESP32's analog-to-digital converter (ADC) typically reads voltages from 0V up to around 3.3V. However, many gas sensors like the MQ-135 and MQ-4 are powered by 5V and can output analog voltages up to 5V, which could potentially damage the ESP32 or give inaccurate readings.

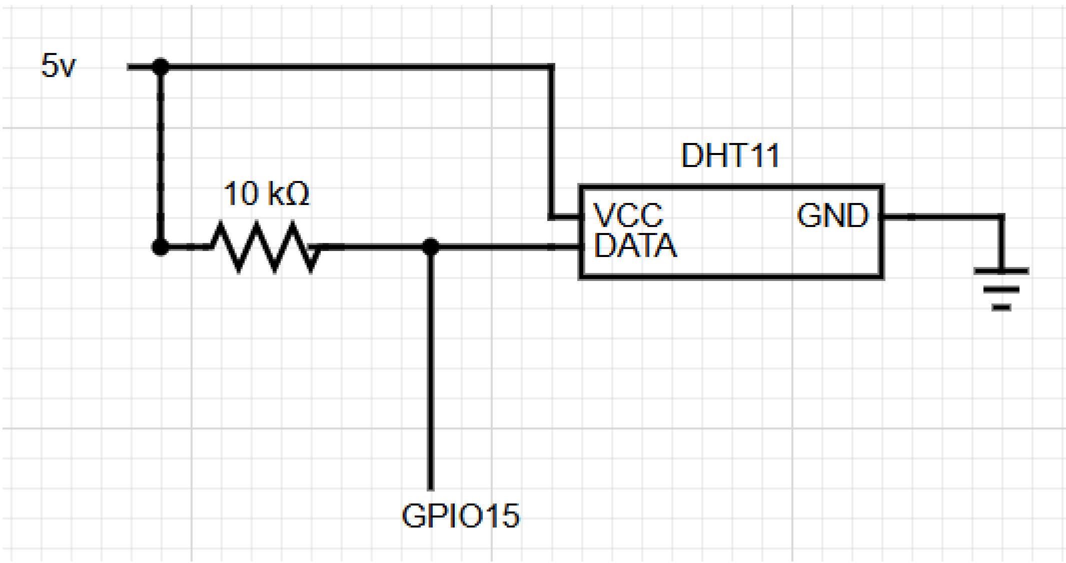
The voltage divider solves this by reducing the 5V signal to a safe level using the formula:

$$V_{out} = V_{in} \times \frac{R2}{R1 + R2}$$

$$V_{out} = 5V \times \frac{20k}{10k + 20k} = 5V \times \frac{2}{3} \approx 3.3V$$

In the circuit being discussed, the ESP32, which works at 3.3V levels, is safely connected with 5V analog signals from the gas sensors (MQ-135 and MQ-4), using a voltage divider. The circuit uses two resistors (10kΩ and 20kΩ) to provide a voltage divider because directly connecting a 5V output to an ESP32 GPIO pin could damage the microcontroller. The ESP32 can safely read the 5V analog output from the sensors thanks to this divider, which lowers it to about 3.3V. In particular, these resistor pairs are used to connect the analog outputs (AOUT) of the MQ-135 and MQ-4 sensors before they reach the analog input pins of the ESP32. Using the formula above, the resistors divide the voltage. Making sure that the voltage supplied to the ESP32 stays within its safe operating range. This technique allows accurate sensor data gathering while protecting the microcontroller. In circuits like this, a similar voltage divider configuration is generally required whenever a sensor or module outputs a 5V signal in order to guard against damage and guarantee dependable functioning.

Pull-up resistor for DHT11 Sensor:



The 10kΩ resistor used with the DHT11 sensor serves as a pull-up resistor on the data line. Its main function is to ensure a stable HIGH signal level when the sensor is not actively pulling the line LOW. This prevents the data line from floating, which can cause unreliable or inconsistent readings.

For digital communication to work reliably, especially with timing-sensitive sensors like the DHT11, the voltage transitions need to occur quickly enough. A 10kΩ resistor strikes a good balance—it is high enough to avoid excessive current draw, yet low enough to allow the signal to rise in time. For example, in a 5V system, a 10kΩ resistor would result in only 0.5 mA of current draw based on Ohm's Law (I = V/R). This makes it both safe and energy-efficient.

Documented Code:

```
// Blynk Template and Authentication Details
#define BLYNK_TEMPLATE_ID "TMPL6yZsq3MWN"  // Unique Template ID for Blynk project
#define BLYNK_TEMPLATE_NAME "esp32"        // Template name in Blynk app
#define BLYNK_AUTH_TOKEN "Dzp-i1Ef5FurNHz9rWUzCoSsfx7yTD3P"  // Authentication
token for Blynk


// Include necessary libraries
#include <Wire.h>                    // Wire library for I2C communication
#include <LiquidCrystal_I2C.h>        // LCD library for I2C LCD display
#include <DHT.h>                     // DHT sensor library for temperature and humidity
#include <WiFi.h>                    // Wi-Fi library for ESP32
#include <WiFiClient.h>               // Wi-Fi client library for ESP32
#include <BlynkSimpleEsp32.h>         // Blynk library for ESP32


// Wi-Fi credentials
char ssid[] = "Sm freeWIFI";          // Wi-Fi SSID
char pass[] = "152439298m";           // Wi-Fi password


// Sensor Pin Definitions
#define MQ4_PIN     34              // Analog pin for MQ4 gas sensor
#define MQ135_PIN   35              // Analog pin for MQ135 gas sensor
#define DHT_PIN     15              // Digital pin for DHT sensor
#define RED_LED     26              // Pin for red LED (spoiled food indicator)
#define GREEN_LED   27              // Pin for green LED (fresh food indicator)


// DHT Sensor Setup
#define DHTTYPE DHT11                 // Define DHT sensor type as DHT11
DHT dht(DHT_PIN, DHTTYPE);            // Initialize DHT sensor


// LCD Display Setup
LiquidCrystal_I2C lcd(0x27, 16, 2);    // Initialize LCD with I2C address 0x27 and 16x2 display


// Thresholds for gas sensor readings
const int MQ4_THRESHOLD = 500;        // Threshold for MQ4 gas sensor
const int MQ135_THRESHOLD = 500;      // Threshold for MQ135 gas sensor


// Blynk Timer for periodic tasks
BlynkTimer timer;
```

```
// Flags to prevent duplicate alerts
bool spoilageAlertSent = false;
bool dhtAlertSent = false;


// Function to read sensor data and send to Blynk
void sendToBlynk() {
  // Read temperature and humidity from DHT sensor
  float temp = dht.readTemperature();
  float humid = dht.readHumidity();


  // Read gas sensor values
  int mq4Value = analogRead(MQ4_PIN);
  int mq135Value = analogRead(MQ135_PIN);


  // Send sensor data to Blynk app
  Blynk.virtualWrite(V0, temp);        // Send temperature to virtual pin V0
  Blynk.virtualWrite(V1, humid);       // Send humidity to virtual pin V1
  Blynk.virtualWrite(V2, mq4Value);    // Send MQ4 sensor value to virtual pin V2
  Blynk.virtualWrite(V3, mq135Value);  // Send MQ135 sensor value to virtual pin V3


  // Check for food spoilage based on gas sensor readings
  if (mq4Value > MQ4_THRESHOLD || mq135Value > MQ135_THRESHOLD) {
    digitalWrite(RED_LED, HIGH);       // Turn on red LED
    digitalWrite(GREEN_LED, LOW);      // Turn off green LED
    Blynk.virtualWrite(V4, "⚠️ Spoiled"); // Update Blynk app with "Spoiled" status


    // Send spoilage alert if not already sent
    if (!spoilageAlertSent) {
      Blynk.logEvent("food_spoilage_detected", "⚠️ Food spoilage detected!");
      spoilageAlertSent = true;
    }


    Serial.println("⚠️ Food Spoilage Detected!");
  } else {
    digitalWrite(RED_LED, LOW);        // Turn off red LED
    digitalWrite(GREEN_LED, HIGH);     // Turn on green LED
    Blynk.virtualWrite(V4, "✅ Fresh"); // Update Blynk app with "Fresh" status


    // Reset spoilage alert flag
    if (spoilageAlertSent) {
      spoilageAlertSent = false;
    }


    Serial.println("✅ Food is Fresh.");
```

```
  }

  // Check for abnormal temperature or humidity readings
  if ((temp > 40 || humid > 80) && !dhtAlertSent) {
    Blynk.logEvent("dht11", "The surrounding can spoil your food");
    dhtAlertSent = true;
  }

  // Reset DHT alert flag if readings are normal
  if (temp <= 30 && humid <= 80) {
    dhtAlertSent = false;
  }

  // Display sensor data on LCD
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temp, 1);
  lcd.print("C H:");
  lcd.print(humid, 0);
  lcd.print("%  ");

  lcd.setCursor(0, 1);
  lcd.print("G1:");
  lcd.print(mq4Value);
  lcd.print(" G2:");
  lcd.print(mq135Value);
}

// Setup function to initialize hardware and connect to Blynk
void setup() {
  Serial.begin(115200);              // Start serial communication at 115200 baud rate
  dht.begin();                       // Initialize DHT sensor
  lcd.init();                        // Initialize LCD
  lcd.backlight();                   // Turn on LCD backlight

  // Set LED pins as output
  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);

  // Display initial message on LCD
  lcd.setCursor(0, 0);
  lcd.print(" Spoil Alert!");
  lcd.setCursor(0, 1);
  lcd.print(" Connecting...");
```

```
  // Connect to Wi-Fi and Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);


  // Clear LCD screen
  lcd.clear();


  // Setup timer to call sendToBlynk function every 2000 milliseconds for consistent and real
time analysis of data
  timer.setInterval(2000L, sendToBlynk);
}


// Main loop function
void loop() {
  Blynk.run();                    // Run Blynk tasks
  timer.run();                    // Run timer tasks
}
```

Threshold Approximation:

To estimate the threshold parts per million (ppm) that indicate the early stages of food spoilage, we couldn't find any existing studies that provided a clear or confirmed value. Because of this, we decided to use a simple method based on smell. We took food samples and left them at room temperature, without refrigeration or any other preservation. Over time, we smelled the food at regular intervals to notice any changes. We continued this process until a clear bad smell developed, which we used as a sign that the food had spoiled.

This method is not highly accurate, but it helped us get a basic idea of when spoilage begins, based on what a person can detect through smell. In this test, we used both egg and shrimp samples. For the egg, we let it spoil for five days to observe how bad the smell could get and to understand the upper limit of spoilage. For the shrimp, we focused more on identifying the first noticeable signs of spoilage using the smell test. The point at which we first noticed a change in smell was used as the estimated threshold value for the early stage of spoilage.

Sample	Time left	PPM Observed
Egg	5 days	2385 ppm

Sample	Time left	PPM Observed
Shrimp	5hrs	500 ppm

Test Circuit:

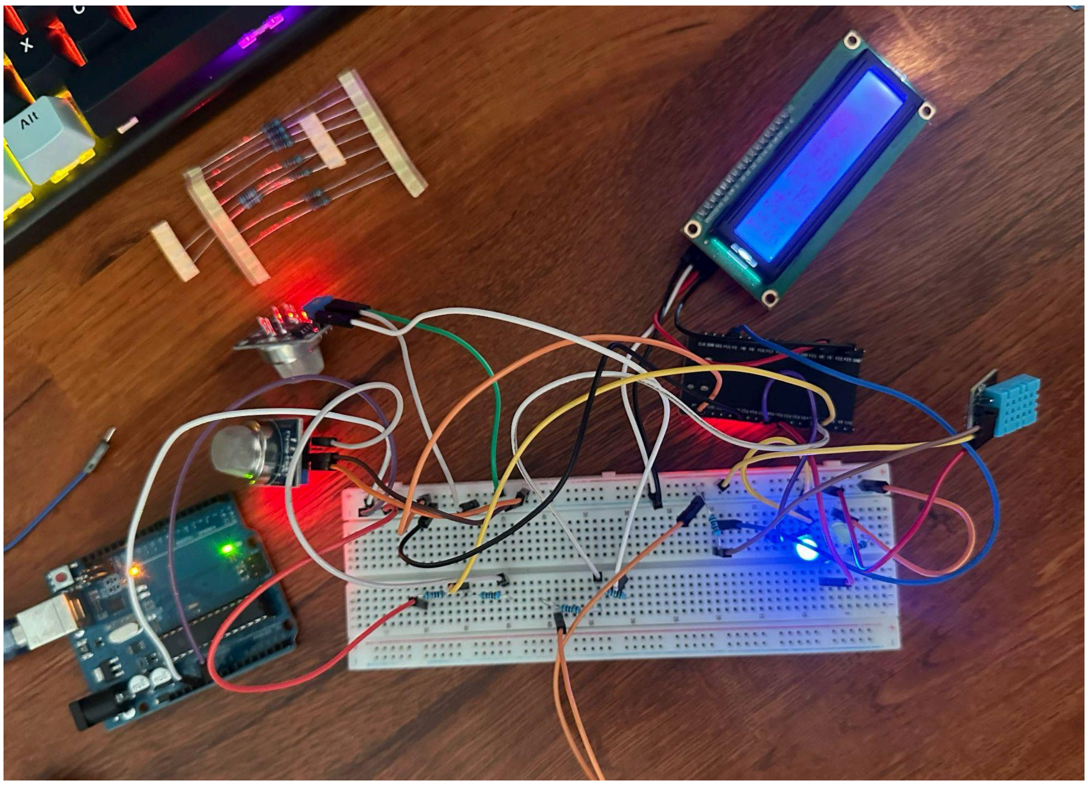


Figure 6.1: Breadboard Circuit Testing

In Figure 6.1, the components were assembled on a breadboard and connected to the Arduino Uno. Proper wiring practices were followed, and resistors were used where necessary to ensure voltage regulation and signal conditioning. The LCD display was connected via I2C for simplified communication and real-time display of sensor data.

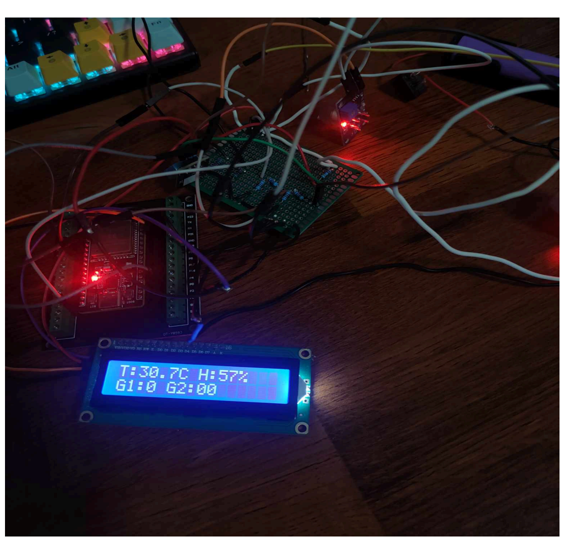


Figure 6.2: PCB Circuit Testing

The components were soldered onto a custom-designed Printed Circuit Board (PCB) for improved durability, stability, and reliability of connections. The PCB layout was carefully designed to minimize noise and ensure proper signal routing between components. Resistors and other passive components were placed strategically to support voltage regulation and signal conditioning. The MQ-4, MQ-135, and DHT11 sensors were connected through dedicated headers for secure and modular interfacing. The LCD display was integrated using the I2C protocol to reduce pin usage and streamline communication. The PCB-based setup provided a more robust and professional platform for testing, reducing the risk of loose connections and improving the overall performance of the system.

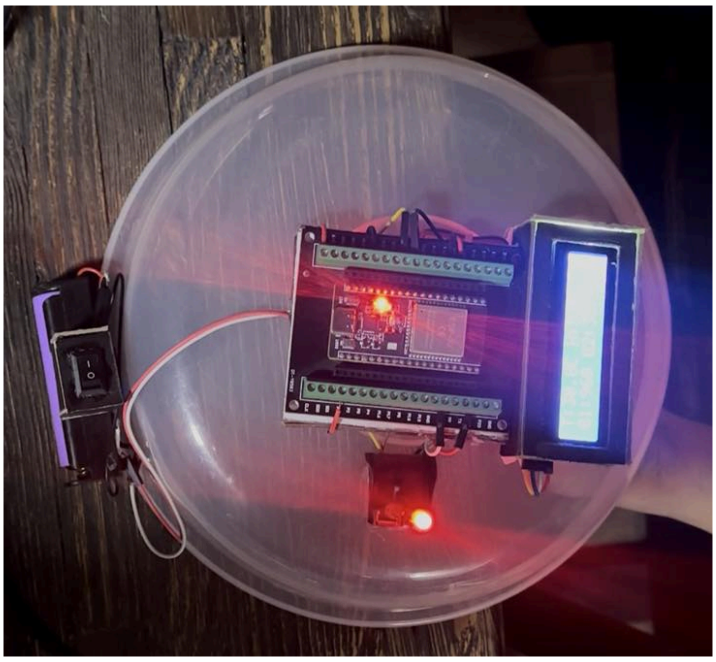


Figure 6.3: MQ4 Sensor Testing

In figure 6.3, the MQ-4 sensor, which detects methane and LPG gases, was tested by exposing it to lighter gas. Once exposed, the sensor's analog output fluctuated significantly, and the corresponding data was displayed on the LCD. This indicated that the sensor was successfully detecting gas and that the Arduino was processing and displaying the values correctly.

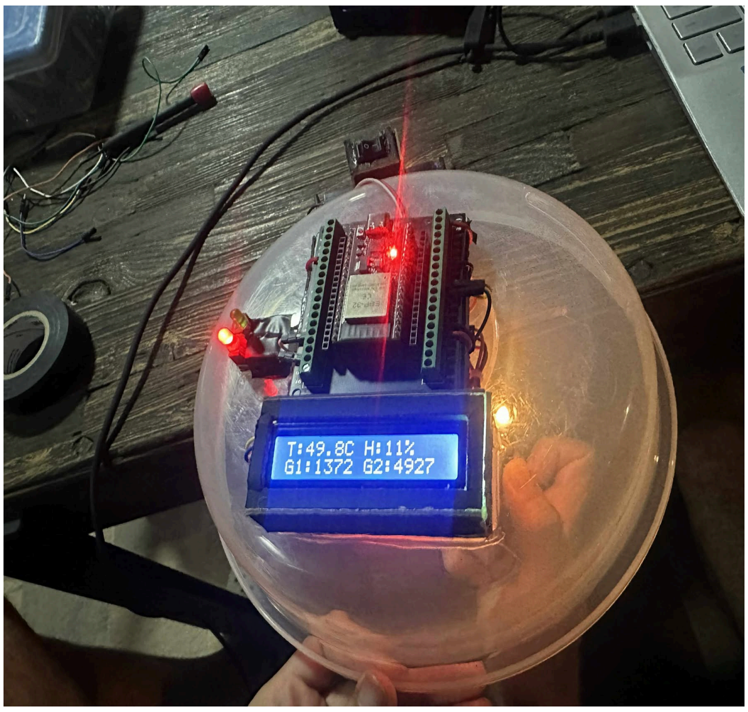


Figure 6.4: DHT11 Sensor Testing

In Figure 6.4, the DHT11 sensor was evaluated for temperature responsiveness by briefly exposing it to the heat from a lighter (without direct flame contact). The sensor detected a rise in temperature, and the data was promptly updated on the LCD. This confirmed the sensor's ability to accurately capture and send environmental data to the microcontroller.

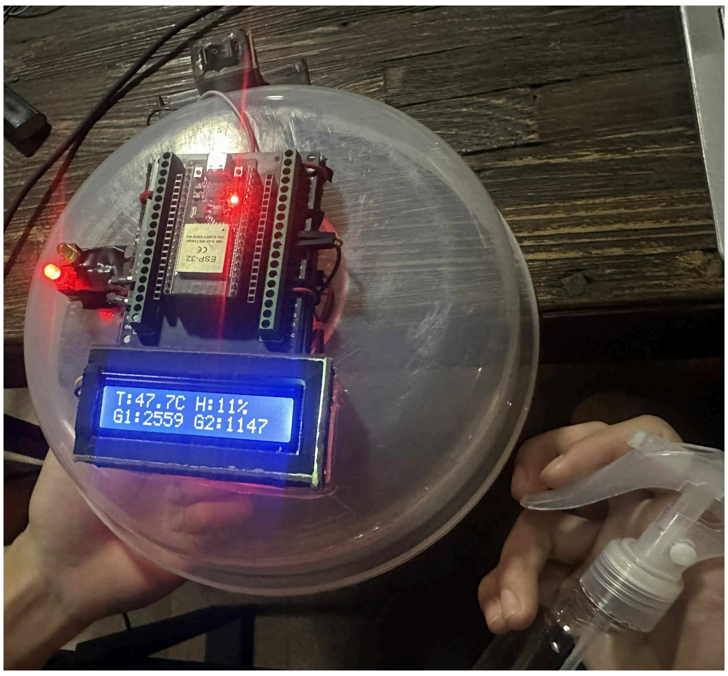


Figure 6.5: MQ135 Sensor Testing

In figure 6.5, or the MQ-135, we used alcohol vapors to simulate air contamination. Upon bringing an alcohol-soaked material near the sensor, a noticeable change in sensor readings was observed. This change confirmed the sensor's sensitivity to volatile organic compounds and its operational state. The change was also reflected on the LCD and transmitted via the IoT platform.

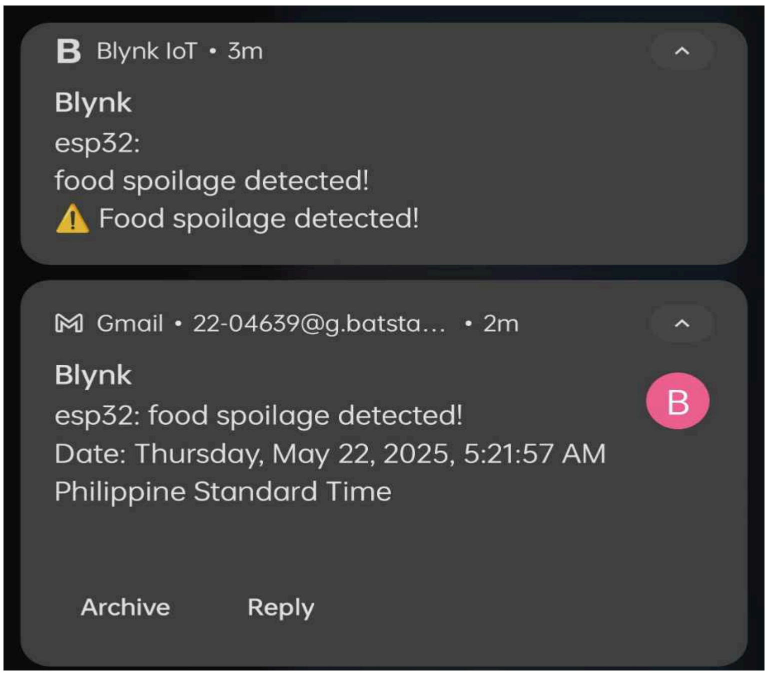


Figure 6.6: Blynk IoT Testing

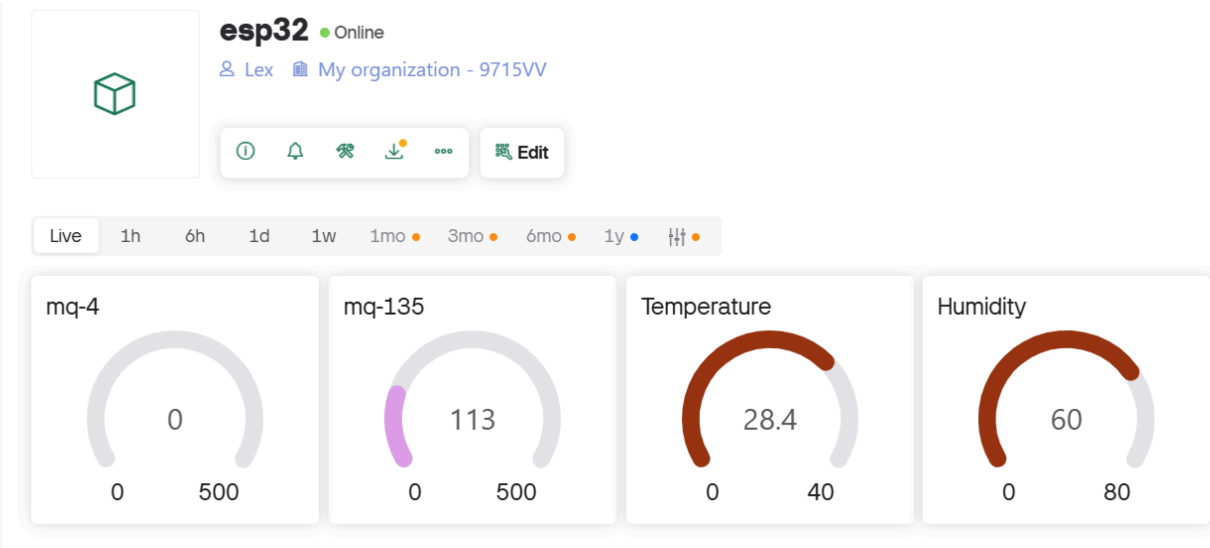


Figure 6.7: Blynk Dashboard

For Figure 6.6 and Figure 6.7, the ESP32 was programmed to transmit data to the Blynk app using Wi-Fi. The sensor readings were successfully displayed on the Blynk dashboard in real-time. Additionally, threshold conditions were set in the code to trigger email alerts via Blynk when gas levels or temperature values exceeded predefined limits. The system successfully sent email notifications during the tests, confirming the IoT integration's reliability.

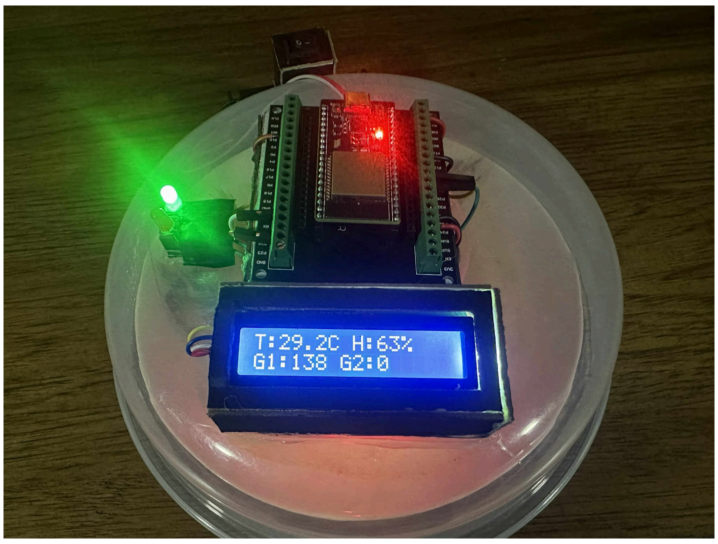


Figure 6.8: Actual Food Testing (30 mins old)

To simulate a real-world scenario, we tested the system's ability to detect gas emissions from actual food by placing a portion of shrimp on a table near the MQ-4 sensor. The shrimp was left exposed at room temperature for approximately 30 minutes. During this period, the MQ-4 sensor registered only a slight increase in methane levels, indicating the early stages of decomposition. The response was minimal, as expected, since the food had not yet undergone significant spoilage. This test demonstrated the sensor's sensitivity to low levels of methane emissions, which could be valuable for early detection in food freshness monitoring applications.

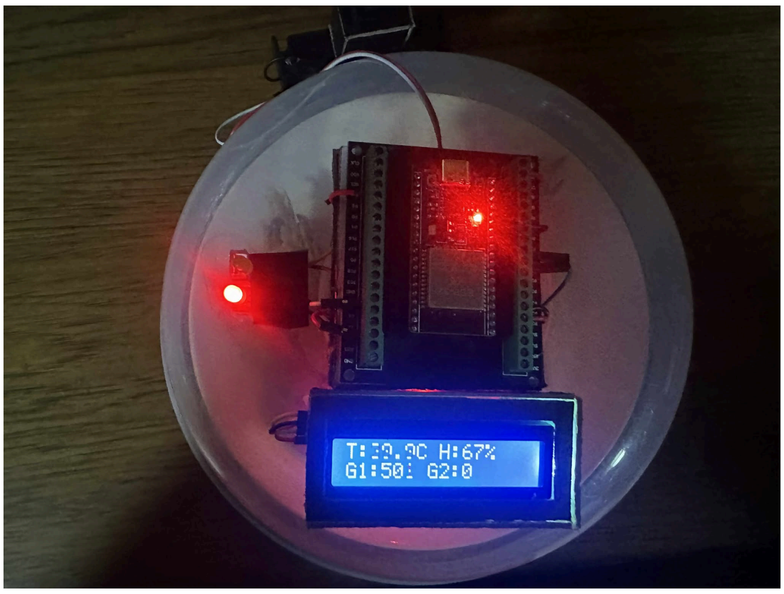


Figure 6.9: Actual Food Testing (5 hrs old)

In a follow-up test to assess the progression of spoilage, a portion of shrimp was left exposed at room temperature for approximately 5 hours before being placed near the MQ-4 sensor. This time, the sensor detected a significantly higher concentration of methane gas compared to the 30-minute test. The elevated readings indicated the onset of protein decomposition and microbial activity, marking the initial stage of spoilage. The MQ-4 sensor's strong response confirmed its capability to detect increased methane emissions associated with early food decay, reinforcing its potential application in real-time food freshness monitoring systems.

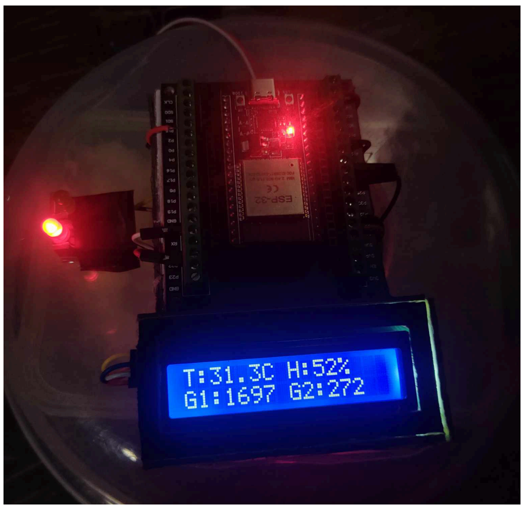


Figure 6.10: Actual Food Testing (5 days old)

To evaluate the sensor's performance in detecting advanced stages of spoilage, a sample of an egg that had been left at room temperature for five days was tested. Upon exposure to the MQ-4 sensor, a substantial spike in methane levels was recorded. The high concentration of gas indicated extensive decomposition, accompanied by the strong odor typically associated with rotting organic matter. This result confirmed that the food was in a late stage of spoilage, where microbial activity and protein breakdown were well underway.

Results:

MQ4 Sensor

Methane	Status	LED Status
138 ppm	Below threshold	Green
500 ppm	Threshold	Red

During the experimental testing of the MQ-4 sensor, methane levels were monitored to assess the food's freshness. When the methane concentration measured was 138 ppm, it remained below the defined spoilage threshold, indicating that the food was still safe to consume. Accordingly, the system activated the green LED to signal freshness. However, when the methane level increased to 500 ppm reaching the set threshold it signified the beginning of spoilage. In response, the system triggered the red LED, alerting users that the food was no longer fresh and should be inspected or discarded.

DHT11 Sensor

Temperature	Status	LED Status
32°C	Below threshold	Green
40°C	Above threshold	Red

Temperature readings from the DHT11 sensor played a critical role in assessing environmental conditions that influence food spoilage. At 32°C, the temperature remained below the system's defined risk threshold, and the green LED was activated to indicate safe storage conditions. However, when the temperature rose to 40°C exceeding the threshold the system identified this as a potential risk for accelerated spoilage. As a result, the red LED was triggered, warning users of unsafe temperature levels.

MQ135 Sensor

Alcohol	Status	LED Status
0	Below threshold	Green
40	Above threshold	Red

The MQ-135 sensor was utilized to detect the presence of alcohol vapors, which can serve as indicators of food spoilage and contamination. When the sensor reading was at 0, it signified the absence of detectable alcohol levels in the environment, and the green LED was illuminated to reflect safe conditions. However, when the alcohol concentration increased to a value of 40, the system identified this as a potential sign of spoilage or contamination. Consequently, the red LED was activated to alert users.

Actual Testing

Timeline	Methane	Status	LED Status
30 mins	138 ppm	Below threshold	Green
5 hrs	500 ppm	Threshold	Red
5 days	1697 ppm	Above Threshold	Red

The timeline-based testing of methane levels demonstrated the effectiveness of the MQ-4 sensor in detecting progressive stages of food spoilage. At the 30-minute mark, the methane level measured was 138 ppm, which remained below the spoilage threshold, and the green LED indicated the food was still fresh. After 5 hours, the methane concentration rose to 500 ppm right at the defined threshold triggering the red LED and signaling the initial stage of spoilage. By the fifth day, methane levels had spiked dramatically to 1697 ppm, far above the threshold, confirming advanced decomposition.

Conclusion:

The development and testing of the SpoilAlert smart food cover have demonstrated the feasibility and effectiveness of using sensor-based technology to monitor food freshness in real time. By integrating gas sensors (MQ-4 and MQ-135), a DHT11 sensor, and an ESP32 microcontroller with IoT capabilities, the system successfully detected varying levels of gases such as methane and alcohol common indicators of food spoilage. The gradual increase in methane levels during actual food testing from 30 minutes to 5 hours to 5 days validated the system's sensitivity and reliability across different stages of decomposition.

The PCB implementation enhanced circuit stability and minimized connection errors, making the device more robust for long-term use. Visual indicators (LEDs), real-time data display via I2C LCD, and wireless updates through the Blynk mobile app contributed to a user-friendly and accessible monitoring system. Furthermore, the system effectively issued alerts when gas or temperature readings crossed predefined thresholds, thereby helping to prevent the consumption of spoiled food and supporting timely intervention.

Overall, SpoilAlert has proven to be a practical, affordable, and scalable solution for promoting food safety, minimizing waste, and supporting smarter food storage practices. Its successful implementation highlights the potential of IoT and embedded systems in addressing real-world challenges in household and industrial settings alike.