Experiment 3

Student Name: Sumit Kumar **UID:22BCS10048**
Branch: CSE **Section/Group:22KRG_IOT_1_A**
Semester: 6th **Date of Performance:30/01/25**
Subject Name: IOT LAB **Subject Code: 22CSP-367**

1. **Aim:**
   Monitor air quality using a gas sensor (MQ135) and display the data on ThingSpeak.
2. **Objective:**
   Monitor air quality using the MQ135 gas sensor and send the data to ThingSpeak for visualization and analysis.
3. **Hardware Used:**
   - Hardware Required:
   - MQ135 gas sensor
   - ESP8266/NodeMCU (or any microcontroller with Wi-Fi capability)
   - Breadboard and jumper wires
   - Power supply (5V for the sensor and microcontroller)
   - ThingSpeak account (free API key)
4. **Procedure:**

1. Connect the Hardware:
- MQ135 Pinout:
- VCC: Connect to 5V.
- GND: Connect to GND.
- AO (Analog Output): Connect to the analog pin of the ESP8266 (e.g., A0 on NodeMCU).

Wiring:
- MQ135 VCC → NodeMCU 3V3 or 5V (depending on module support)
- MQ135 GND → NodeMCU GND
- MQ135 A0 → NodeMCU A0

2. Set Up ThingSpeak:
- Go to ThingSpeak and create a free account.
- Create a new channel and add a Field (e.g., "Air Quality").

- Note down the Write API Key from the API Keys tab.

3. Install Required Libraries:

- Ensure the ESP8266 library is installed in your Arduino IDE:
- Go to Tools > Manage Libraries.
- Search for ESP8266 and install it.

5. Code:

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
ssid = "Your_SSID"; const char* password = "Your_PASSWORD";
const char* server = "http://api.thingspeak.com";
String apiKey = "YOUR_API_KEY"; //
MQ135 connected to A0 int
mq135Pin=A0;
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password); while (WiFi.status() !=
  WL_CONNECTED) { delay(1000);
  Serial.println("Connected to WiFi"); }
void loop() {
  int airQuality = analogRead(mq135Pin);
  Serial.println("Air Quality Value: " + String(airQuality));
   int httpCode = http.GET(); if
   (httpCode > 0) {
      Serial.println("Data sent to ThingSpeak successfully.");
   } else {
    Serial.println("Error sending data.");
   } http.end(); }
```

6. Output:

```
PS C:\Users\manik\downloads> python exp3.py
Received: MQ135 RZero: 52.72    Corrected RZero: 52.09  Resistance: 33.35    PPM: 1167.50   Corrected PPM: 1206.88ppm
Received: MQ135 RZero: 51.30    Corrected RZero: 50.69  Resistance: 32.45    PPM: 1259.32   Corrected PPM: 1301.79ppm
Received: MQ135 RZero: 50.47    Corrected RZero: 49.87  Resistance: 31.93    PPM: 1317.15   Corrected PPM: 1361.57ppm
Received: MQ135 RZero: 49.67    Corrected RZero: 49.08  Resistance: 31.42    PPM: 1356.89   Corrected PPM: 1402.65ppm
Received: MQ135 RZero: 49.14    Corrected RZero: 48.56  Resistance: 31.25    PPM: 1418.31   Corrected PPM: 1444.73ppm
Received: MQ135 RZero: 48.88    Corrected RZero: 48.30  Resistance: 30.92    PPM: 1439.29   Corrected PPM: 1487.83ppm
Received: MQ135 RZero: 48.62    Corrected RZero: 48.05  Resistance: 30.76    PPM: 1460.51   Corrected PPM: 1509.77ppm
Received: MQ135 RZero: 48.37    Corrected RZero: 47.79  Resistance: 30.60    PPM: 1481.99   Corrected PPM: 1531.97ppm
Received: MQ135 RZero: 48.11    Corrected RZero: 47.54  Resistance: 30.60    PPM: 1503.72   Corrected PPM: 1531.97ppm
Received: MQ135 RZero: 48.11    Corrected RZero: 47.54  Resistance: 30.43    PPM: 1503.72   Corrected PPM: 1554.44ppm
Received: MQ135 RZero: 47.86    Corrected RZero: 47.54  Resistance: 30.43    PPM: 1503.72   Corrected PPM: 1554.44ppm
Received: MQ135 RZero: 47.86    Corrected RZero: 47.29  Resistance: 30.28    PPM: 1525.72   Corrected PPM: 1577.18ppm
Received: MQ135 RZero: 47.86    Corrected RZero: 47.29  Resistance: 30.28    PPM: 1525.72   Corrected PPM: 1577.18ppm
Received: MQ135 RZero: 47.86    Corrected RZero: 47.29  Resistance: 30.28    PPM: 1525.72   Corrected PPM: 1577.18ppm
Received: MQ135 RZero: 47.61    Corrected RZero: 47.05  Resistance: 30.12    PPM: 1547.98   Corrected PPM: 1577.18ppm
```

Fig 1: Simulated Cloud Air Quality Variations

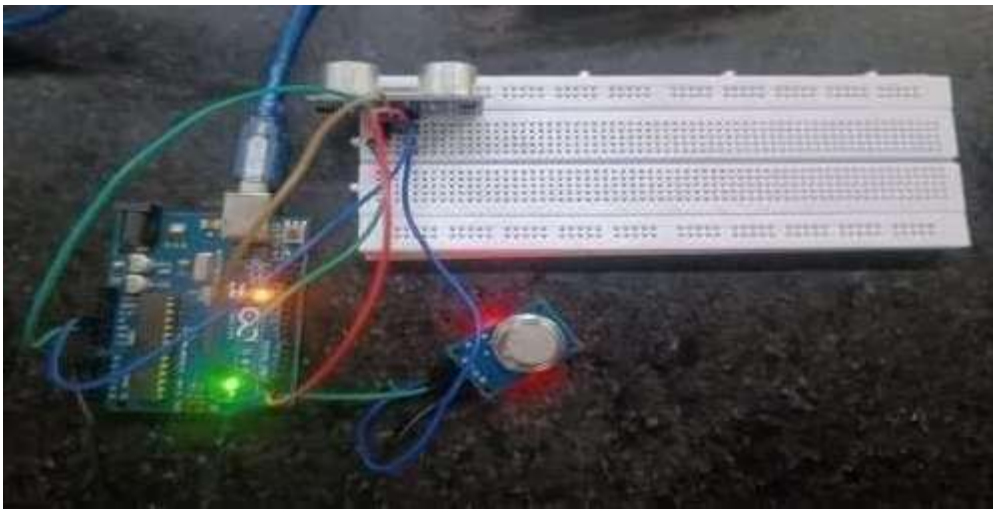

Fig 2: Hardware

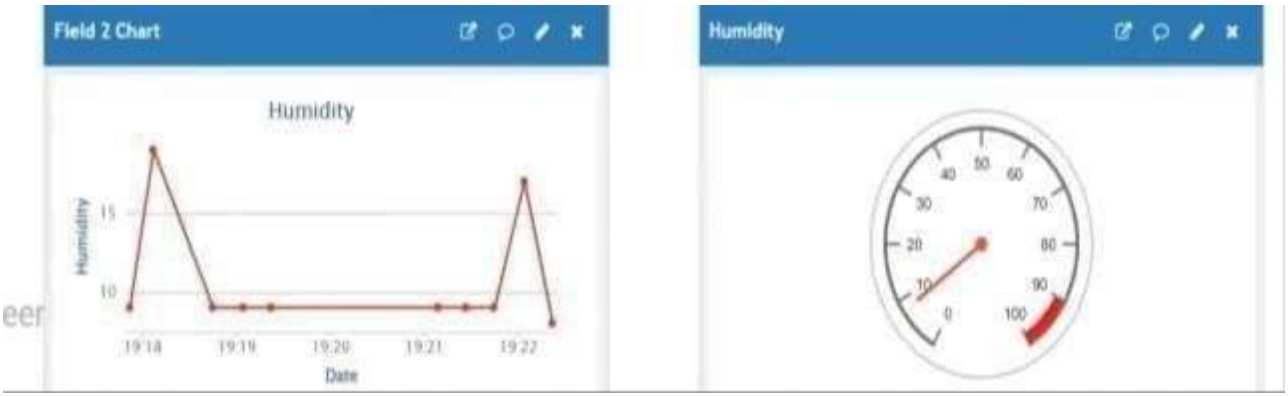

Fig 3: Think speak Visualization

7. Learning Outcome:

- Understanding how to interface and calibrate the MQ135 gas sensor with microcontrollers such as Arduino or ESP32.
- Collecting sensor data efficiently and reading analog values for air quality monitoring.
- Learning how to set up wireless communication protocols (Wi-Fi, MQTT) to connect with cloud platforms.