Real Time Environmental Monitoring & Air Quality Sensing

TEAM MEMBERS

- **ALDRICH MARIAN A**
- **BARATH M**
- **ASHISH C**
- **AKASH SAI**

Aim:

To design and develop a real-time environmental monitoring and air quality sensing system using Raspberry Pi Pico W and MQ-135 sensor to detect harmful gas concentrations and provide immediate alerts through a buzzer.

Components Required:

1. Raspberry Pi Pico W
2. MQ-135 Gas Sensor
3. Buzzer
4. Breadboard
5. OLED
6. Jumper wire

Raspberry Pi Pico W:

A compact microcontroller board developed by Raspberry Pi with built-in Wi-Fi support. It is used to control sensors, process data in real time, and can also send information wirelessly if required.

MQ-135 Gas Sensor:

A sensor module used to measure air quality by detecting gases like CO_2, NH_3, NOx, benzene, and smoke. It provides an analog output that varies depending on the concentration of gases in the environment.

Buzzer:

A simple sound-producing device that generates beeps or tones when given electrical power. In this project, it acts as an alarm to alert users when harmful gas levels are detected.

Breadboard:

A reusable prototyping board that allows electronic components to be connected without soldering. It is mainly used for testing, learning, and quick modification of circuits.

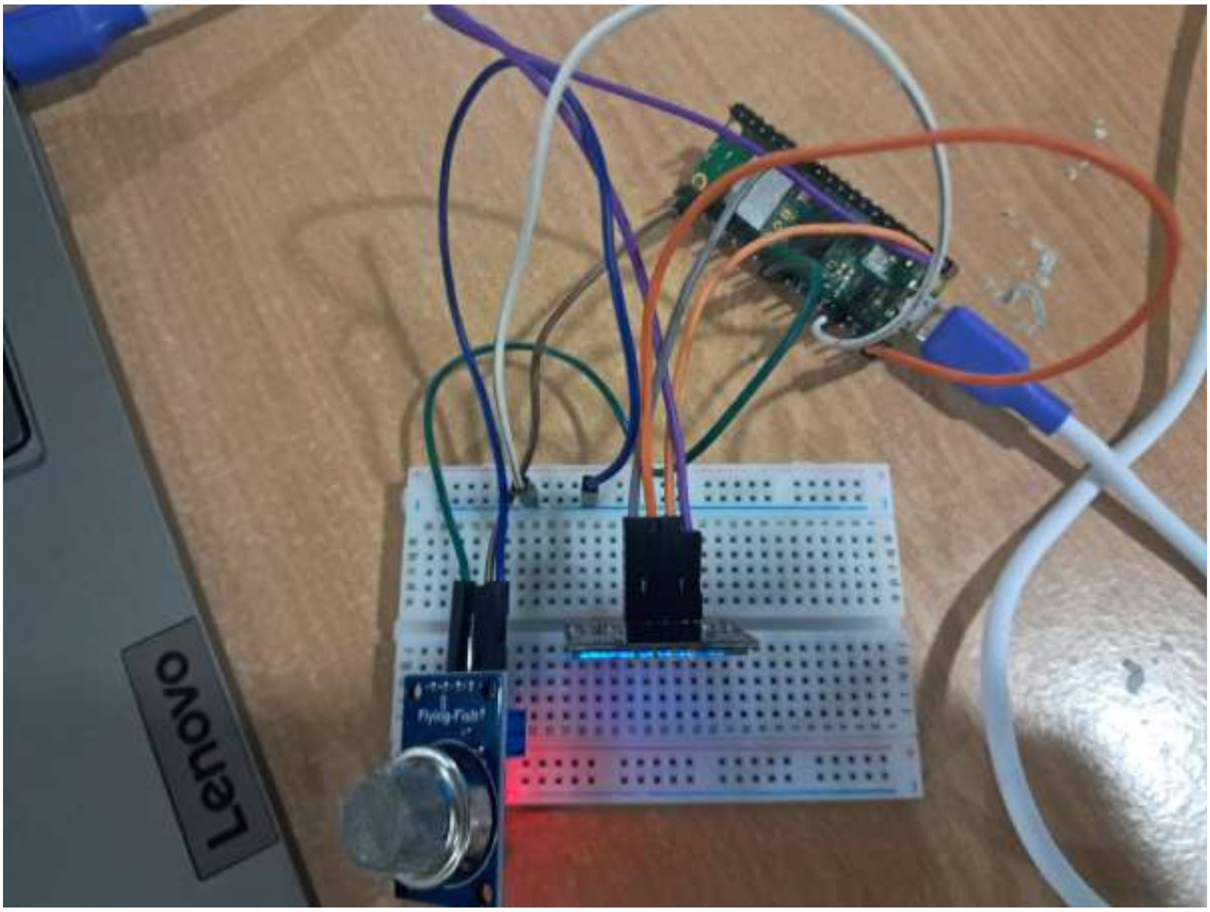
OLED:

A small, energy-efficient display module that uses organic compounds to emit light when an electric current passes through. It’s used to visually display sensor data, readings, or alerts in real time with sharp contrast and low power consumption.

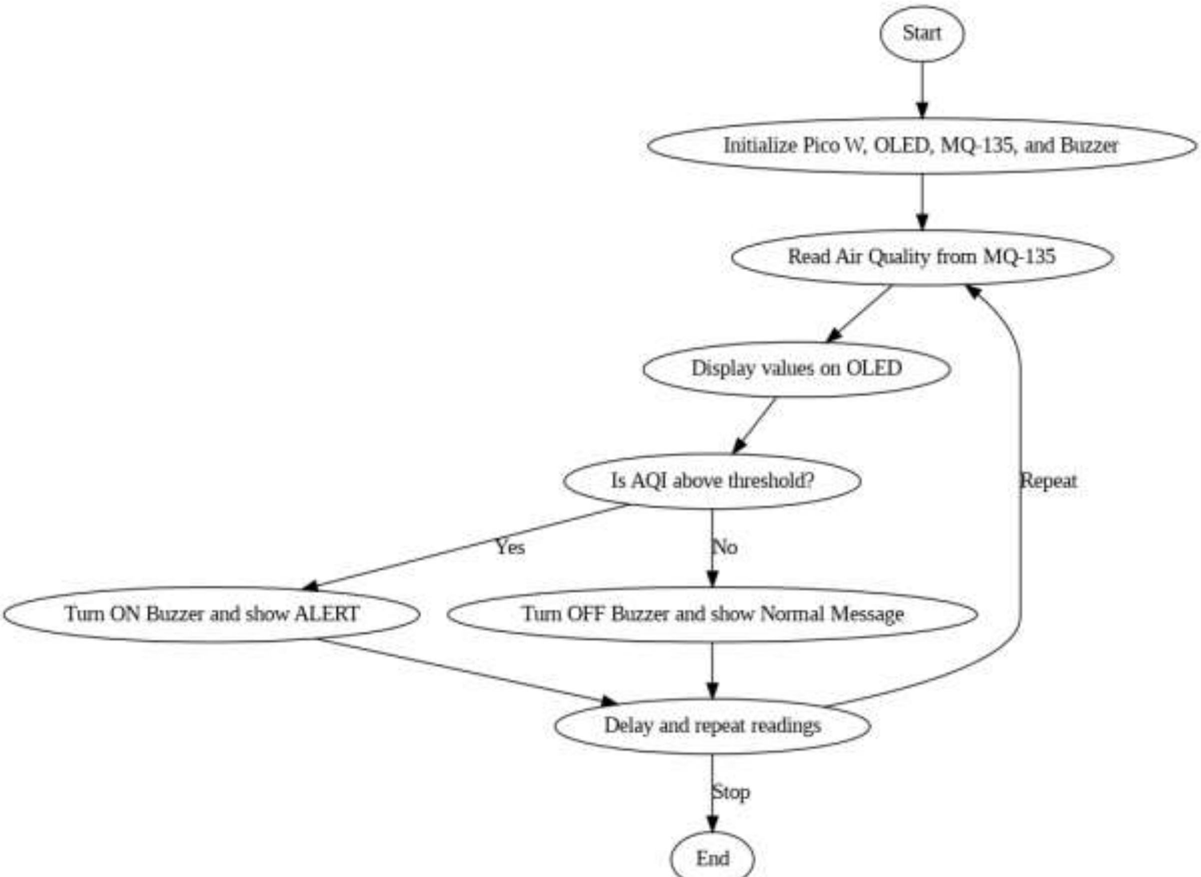
Pin Table:

Component	**GND**	**Raspberry Pi Pico W Pin**	**Description**
MQ-135 VCC	GND	3.3V	Power supply to the gas sensor
MQ-135 GND	GND	GND	Ground connection
MQ-135 A0	Analog output	GP26 (ADC0)	Reads air quality as an analog value
OLED VCC	VCC	3.3V	Power supply to OLED display
OLED GND	GND	GND	Ground connection
OLED SDA	Data line	GP8	I2C data communication
OLED SCL	Clock line	GP9	I2C clock communication
Buzzer (+)	Positive lead	GP15	Digital output to control buzzer
Buzzer (–)	Negative lead	GND	Ground connection for buzzer

Connection Diagram:



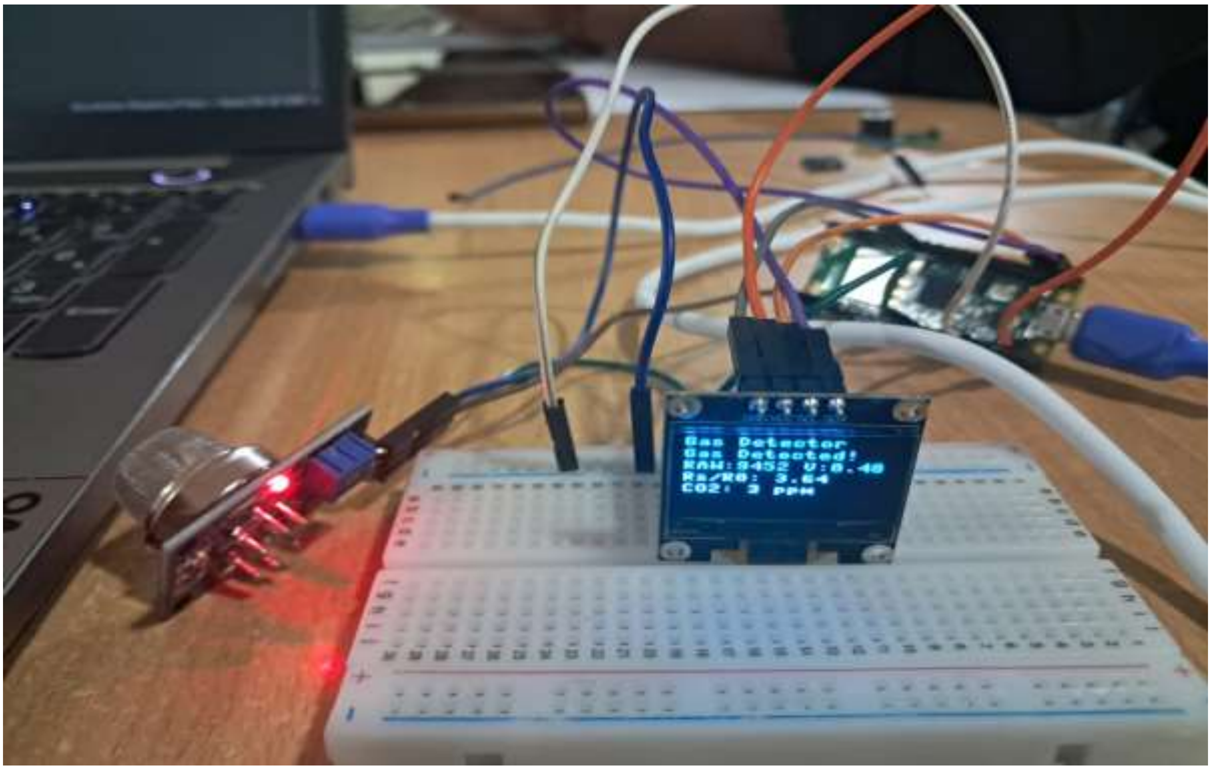
Flowchart:


Start
Initialize Pico W, OLED, MQ-135, and Buzzer
Read Air Quality from MQ-135
Display values on OLED
Is AQI above threshold?
Repeat
Yes
No
Turn ON Buzzer and show ALERT
Turn OFF Buzzer and show Normal Message
Delay and repeat readings
Stop
End

Code:

```
from machine import Pin, ADC, I2C
import ssd1306
import time

# MQ135 on ADC0 (GP26)
mq135 = ADC(Pin(26))

# Buzzer on GP15
buzzer = Pin(15, Pin.OUT)

# I2C for OLED (SDA=GP0, SCL=GP1)
i2c = I2C(0, scl=Pin(1), sda=Pin(0))
oled = ssd1306.SSD1306_I2C(128, 64, i2c)

# Threshold for air quality (adjust after calibration)
THRESHOLD = 300

def get_ppm(value):
    """
    Convert raw ADC (0–65535) to a simulated PPM scale (0–1000).
    NOTE: For real accuracy, you need MQ135 calibration curves.
    """
    return int((value / 65535) * 1000)

while True:
    raw_value = mq135.read_u16()
    ppm = get_ppm(raw_value)

    # Clear display
    oled.fill(0)
    oled.text("Air Quality", 0, 0)
    oled.text("PPM: {}".format(ppm), 0, 20)

    if ppm > THRESHOLD:
        oled.text("Status: BAD", 0, 40)
        buzzer.value(1)  # Turn buzzer ON
    else:
        oled.text("Status: GOOD", 0, 40)
        buzzer.value(0)  # Turn buzzer OFF

    oled.show()
    print("Raw:", raw_value, " PPM:", ppm)
    time.sleep(1)
```

Execution:



```
Gas Detector
Gas Detected!
RAW:9452 V:0.48
Rs/R0: 3.64
CO2: 3 PPM
```