AUTOMATED IRRIGATION CONTROL SYSTEM BASED ON ENVIRONMENTAL SENSING

TEAM MEMBERS:

Joseph Manuel Thomas R

Ram Prasanth S

Manoj N

Revanth Kumar M G

Aim:

To design and implement a real-time environmental monitoring and air quality sensing system using the Raspberry Pi Pico microcontroller with an MQ135 gas sensor for air pollution detection, OLED display for visualization, and buzzer alarm for poor air quality alerts.

Tools / Hardware Required:

• Raspberry Pi Pico (Microcontroller)
• MQ135 Air Quality Sensor (detects CO_2, NH_3, benzene, alcohol, smoke, etc.)
• SSD1306 OLED Display (128x64, I^2C interface)
• Buzzer module
• Breadboard
• Jumper wires
• USB cable & 5V power supply

Theory:

The Air Quality Monitoring System works by detecting harmful gases and pollutants in the environment and displaying real-time readings:

1. Raspberry Pi Pico – Main controller running MicroPython for real-time monitoring.
2. MQ135 Gas Sensor – Detects gases like CO_2, NH_3, benzene, alcohol, and smoke. The analog output is read by the Pico's ADC pin.
3. Buzzer Module – Gives an audible alert when pollution levels cross a threshold.
4. SSD1306 OLED Display – Shows live PPM values and air quality status ("GOOD" or "BAD").

This is the Calibration Logic in which Raw analog sensor values are converted to an approximate PPM scale (0–1000). Threshold values are set for safe and unsafe ranges. This system helps in early detection of poor air quality conditions and can be expanded for IoT-based remote monitoring.

Pin Connections:

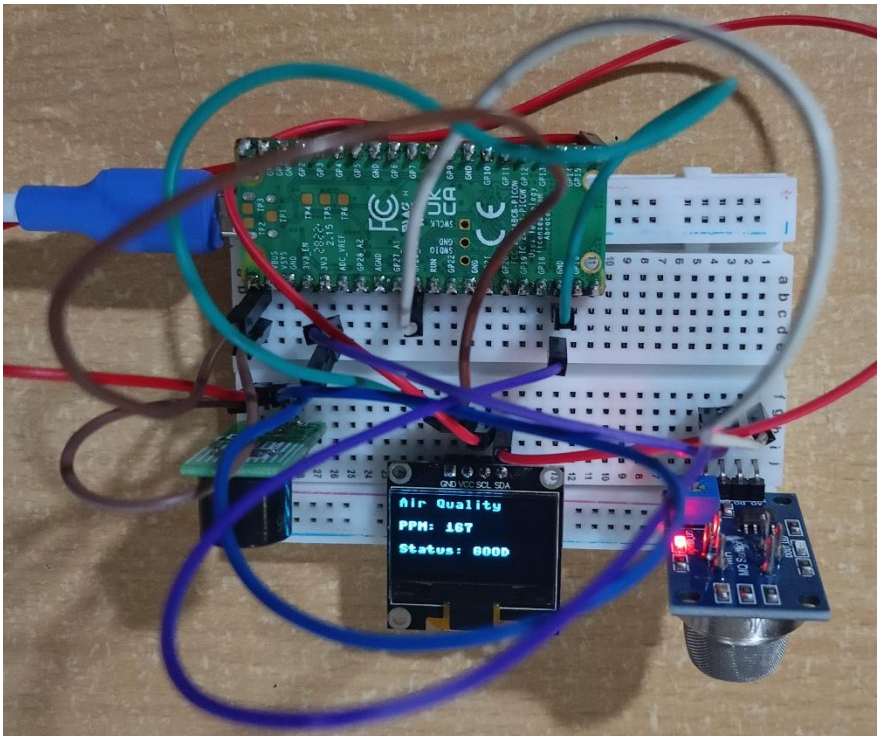
MODULES PIN	GPIO PIN
MQ135 Analog Out	GP26 (ADC0)
Buzzer	GP15
OLED SDA	GP4

MODULES PIN	GPIO PIN
OLED SCL	GP5
VCC	3.3V
GND	GND

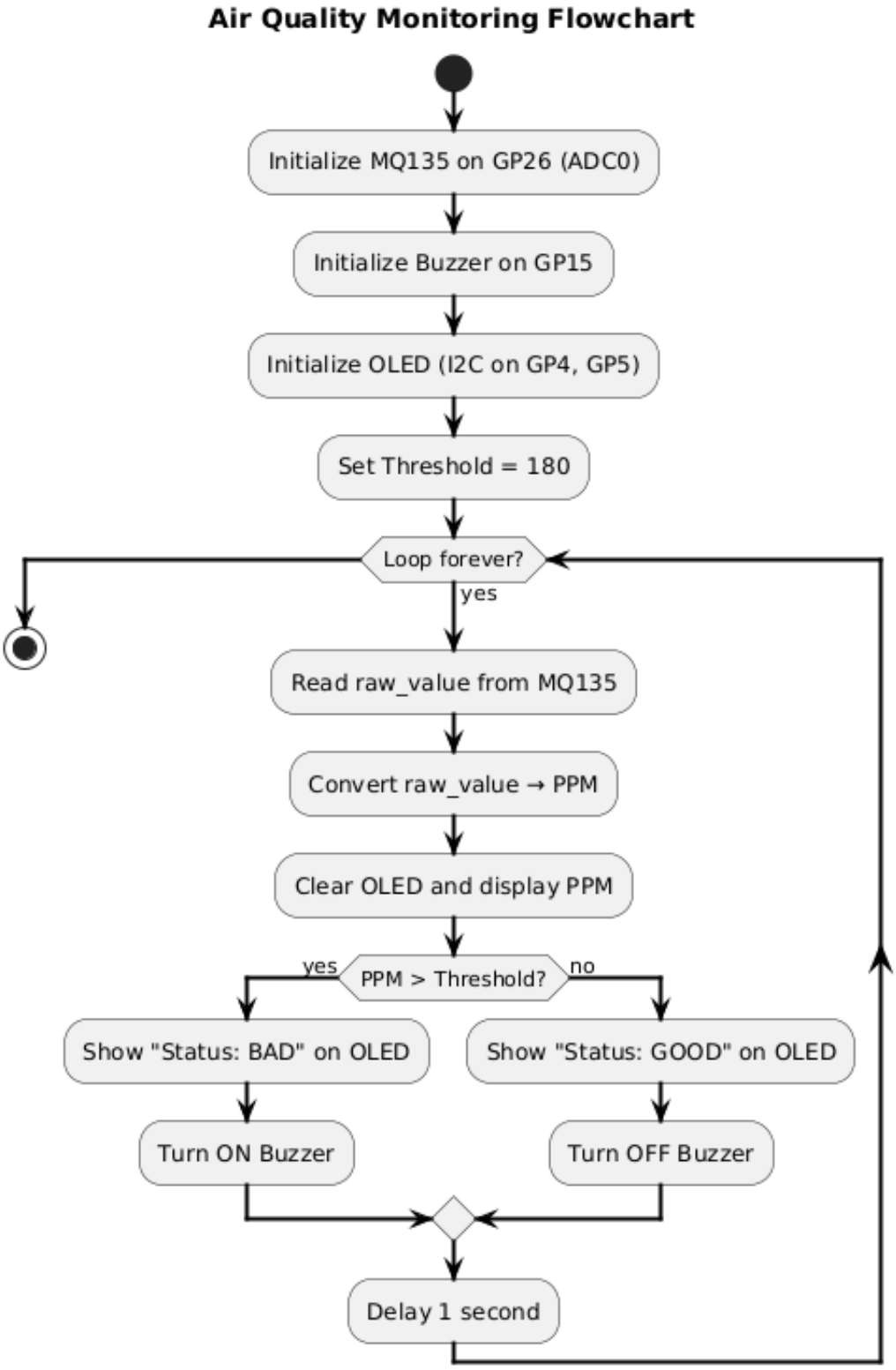
Procedure:

1. Connect the MQ135 sensor's analog output to GP26 (ADC0) of Pico.
2. Connect the OLED display via I²C interface to GP4 (SDA) and GP5 (SCL).
3. Connect the buzzer to GP15.
4. Upload the MicroPython code to Pico using Thonny IDE.
5. Calibrate the MQ135 sensor (allow warm-up time).
6. The OLED will display real-time PPM values.
7. If pollutant levels exceed the set threshold, the buzzer activates and OLED shows "BAD".

Circuit Diagram:



Flowchart:


Air Quality Monitoring Flowchart
Initialize MQ135 on GP26 (ADC0)
Initialize Buzzer on GP15
Initialize OLED (I2C on GP4, GP5)
Set Threshold = 180
Loop forever?
yes
Read raw_value from MQ135
Convert raw_value → PPM
Clear OLED and display PPM
PPM > Threshold?
yes
no
Show "Status: BAD" on OLED
Show "Status: GOOD" on OLED
Turn ON Buzzer
Turn OFF Buzzer
Delay 1 second

Program:

```
from machine import Pin, ADC, I2C
import ssd1306
import time
mq135 = ADC(Pin(26))
buzzer = Pin(15, Pin.OUT)
i2c = I2C(0, scl=Pin(5), sda=Pin(4), freq=400000)
oled = ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3C)
THRESHOLD = 180
def get_ppm(value):
    return int((value / 65535) * 1000)

while True:
    raw_value = mq135.read_u16()
    ppm = get_ppm(raw_value)
    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)
    else:
        oled.text("Status: GOOD", 0, 40)
        buzzer.value(0)
    oled.show()
    print("Raw:", raw_value, " PPM:", ppm)
    time.sleep(1)
```

Result:

The Real-Time Environmental Monitoring and Air Quality Sensing system was successfully developed and tested. The MQ135 gas sensor detected variations in pollutant levels, and the Raspberry Pi Pico processed these values into approximate PPM readings. Results were displayed on the OLED in real time. When the air quality dropped below the safe threshold, the buzzer alarm was triggered, and the display indicated “BAD”. This system enables effective detection of poor air quality conditions and can be extended to IoT platforms for remote monitoring and environmental research.