### REAL-TIME ENVIRONMENTAL MONITORING AND AIR QUALITY SENSING

### TEAM MEMBERS

* VISHWA G
* STEVE CHRISTIAN SAMUEL CS
* SUNDARESH S
* SANGEETH KUMAR LM

### Aim

To design and build a real-time environmental monitoring system using an ESP8266 microcontroller. The system will collect data on ambient temperature, humidity, and air quality using MQ-2 sensors. The collected data will be processed and displayed continuously on an OLED screen, providing an immediate and localized reading of environmental conditions.

### Apparatus Required

* ESP8266 (Microcontroller)
* MQ-2 Air Quality and Gas Sensor
* SSD1306 I2C OLED Display
* Breadboard
* Jumper Wires

### COMPONENT DESCRIPTIONS

### ESP8266

The ESP8266 is a low-cost Wi-Fi-enabled microcontroller. Its powerful processing capabilities and onboard Wi-Fi make it an excellent choice for Internet of Things (IoT) projects. The development board provides easy access to its GPIO pins, including an analog-to-digital converter (ADC) which is essential for reading data from analog sensors like the MQ-2

### MQ-2 Air Quality Sensor

The MQ-135 is a semiconductor sensor designed to detect a wide range of gases, including smoke, ammonia, benzene, alcohol, and carbon dioxide. It is highly sensitive to pollutants in the air. The sensor provides an analog voltage output that corresponds to the concentration of these gases, making it ideal for creating air quality monitoring systems and alarms.

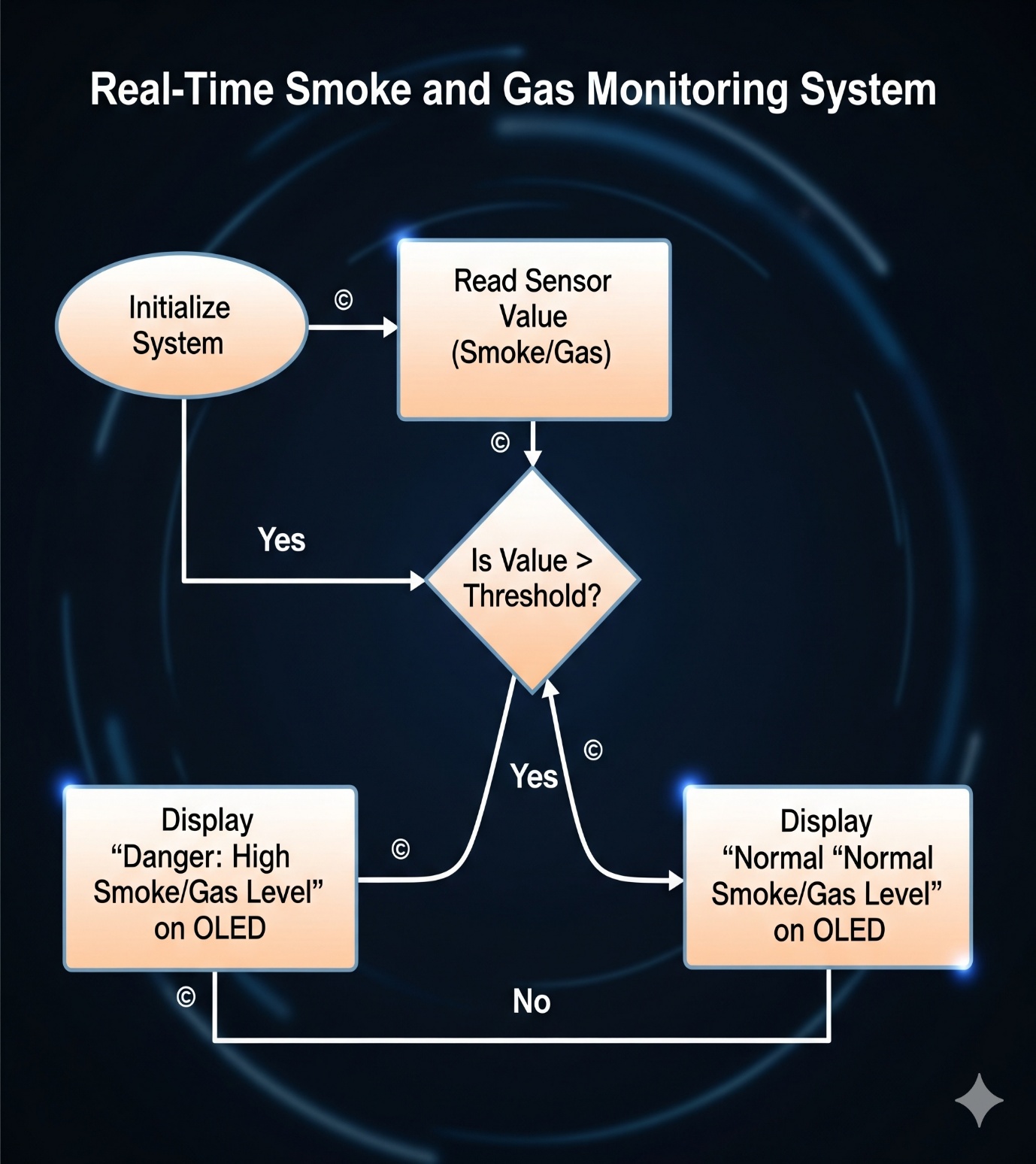
### OLED Display (SSD1306)

An Organic Light Emitting Diode (OLED) display is a screen technology that does not require a backlight, resulting in high contrast and energy efficiency. The SSD1306 is a common driver for these displays. This module uses the I2C communication protocol, which simplifies wiring by requiring only two data lines (SDA and SCL) to interface with the microcontroller for displaying text and graphics.

### PIN TABLE

| Component Pin | ESP8266 (NodeMCU) Pin |
| --- | --- |
| **MQ-2 Sensor** VCC | 3.3V |
| **MQ-2 Sensor** GND | GND |
| **MQ-2 Sensor** AOUT | A0 |
| OLED Display VCC | 3.3V |
| OLED Display GND | GND |
| OLED Display SDA | D2 (GPIO4) |
| OLED Display SCL | D1 (GPIO5) |

### FLOW CHART



### PROGRAM

from machine import Pin, SoftI2C

import utime

from pico\_i2c\_lcd import I2cLcd

# Define the GPIO pins for the HC-SR04 ultrasonic sensor

trigger = Pin(3, Pin.OUT)

echo = Pin(2, Pin.IN)

# Define the built-in LED pin for proximity alerts

led = Pin(25, Pin.OUT) # Onboard LED on Raspberry Pi Pico W

# --- LCD Configuration ---

I2C\_ADDR = 0x27 # I2C address of the LCD

I2C\_NUM\_ROWS = 2 # Number of rows on the LCD

I2C\_NUM\_COLS = 16 # Number of columns on the LCD

# Initialize I2C communication for the LCD

i2c = SoftI2C(sda=Pin(4), scl=Pin(5), freq=400000)

lcd = I2cLcd(i2c, I2C\_ADDR, I2C\_NUM\_ROWS, I2C\_NUM\_COLS)

def measure\_distance():

"""

Triggers the ultrasonic sensor and measures the distance to an object.

Returns the distance in centimeters.

"""

# Send a short pulse to trigger the sensor

trigger.low()

utime.sleep\_us(2)

trigger.high()

utime.sleep\_us(5) # A 10us pulse is recommended, but 5us works well

trigger.low()

# Wait for the echo pin to go high, marking the start of the echo pulse

while echo.value() == 0:

pulse\_start = utime.ticks\_us()

# Wait for the echo pin to go low, marking the end of the echo pulse

while echo.value() == 1:

pulse\_end = utime.ticks\_us()

# Calculate the duration of the pulse

pulse\_duration = pulse\_end - pulse\_start

# Calculate distance using the speed of sound (343 m/s or 0.0343 cm/µs)

# The duration is divided by 2 because the pulse travels to the object and back.

distance\_cm = (pulse\_duration \* 0.0343) / 2

return distance\_cm

# --- Main Program Execution ---

# Display an initial message on the LCD

lcd.putstr("Measuring...")

utime.sleep(2)

lcd.clear()

try:

# Main loop to continuously measure and display distance

while True:

distance = measure\_distance()

# Clear the LCD and display the new distance reading

lcd.clear()

lcd.putstr("Distance:\n{:.2f} cm".format(distance))

# Check if an object is within the 10 cm threshold

if distance < 10:

led.value(1) # Turn on the LED for alert

else:

led.value(0) # Turn off the LED

# Wait for 1 second before the next measurement

utime.sleep(1)

except KeyboardInterrupt:

# Clean up resources if the program is stopped manually (Ctrl+C)

lcd.backlight\_off()

lcd.display\_off()

led.value(0) # Ensure LED is off on exit

print("Program stopped.")

# EXECUTION

