PHASE-3

Development part-3

Air quality monitoring using a Raspberry Pi is a practical and cost-effective way to measure and analyze air pollutants and environmental conditions in a specific area. This project typically involves sensors, data collection, and data visualization. Here's a step-by-step working explanation for an air quality monitoring system using a Raspberry Pi:

Materials Required:

- Raspberry Pi (with power supply and SD card)
- Air quality sensors (e.g., particulate matter sensor, gas sensor)
- Breadboard and jumper wires
- Internet connection (Wi-Fi or Ethernet)
- Monitor and keyboard (for initial setup)

Step 1: Setting Up the Raspberry Pi

1.1. Install the Raspberry Pi OS (Raspbian or Raspberry Pi OS Lite) on your SD card. You can use the official Raspberry Pi Imager to do this.

1.2. Connect the Raspberry Pi to a monitor, keyboard, and internet source (Wi-Fi or Ethernet).

1.3. Boot up the Raspberry Pi and complete the initial setup, including updating the OS and setting up your Wi-Fi network if needed.

Step 2: Connect Air Quality Sensors

2.1. Identify the appropriate pins on your Raspberry Pi for connecting the air quality sensors. This may vary depending on the specific sensors you're using.

2.2. Connect the sensors to the Raspberry Pi using jumper wires and a breadboard. Make sure to follow the wiring instructions provided by the sensor manufacturer.

Step 3: Install Sensor Libraries

3.1. Install any necessary libraries or drivers for your air quality sensors. Typically, this involves running terminal commands or using Python packages to interface with the sensors.

Step 4: Collect Air Quality Data

4.1. Create a Python script to read data from the sensors at regular intervals. This script should read sensor values (e.g., particulate matter concentration, gas levels) and store them in variables.

4.2. You can use libraries like **`Adafruit_IO`** or **`MQTT`** to send this data to a cloud platform or a remote server for storage and analysis. Alternatively, you can store the data locally on the Raspberry Pi.

Step 5: Data Logging and Analysis

5.1. Implement data logging to save historical air quality data. You can use a database or plain text files for this purpose.

5.2. To analyze the data, you can create data visualization tools using Python libraries like Matplotlib or Plotly. This allows you to create charts and graphs that show trends in air quality over time.

Step 6: Remote Monitoring (Optional)

6.1. If you want to monitor air quality remotely, set up a web interface or mobile app that displays real-time and historical air quality data. You can use Flask, Django, or other web frameworks for this.

Step 7: Alerts and Notifications (Optional)

7.1. Implement alerting mechanisms to notify you when air quality levels exceed predefined thresholds. You can use email notifications, SMS, or other communication methods.

Step 8: Calibration and Maintenance

8.1. Regularly calibrate and maintain your air quality sensors to ensure accurate readings. Calibration depends on the sensor type and manufacturer's recommendations.

Step 9: Power Management

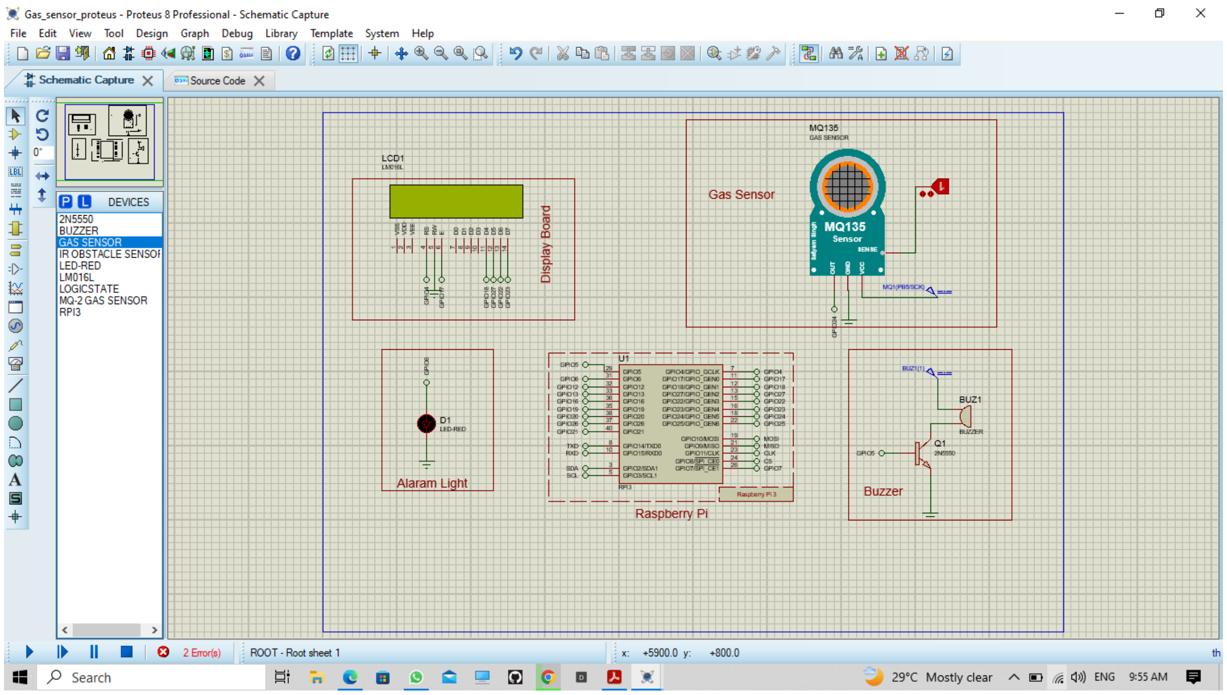
9.1. To ensure continuous monitoring, consider a suitable power management solution for the Raspberry Pi, such as an uninterruptible power supply (UPS) or power bank.

Step 10: Deployment

10.1. Mount the Raspberry Pi and sensors in the location where you want to monitor air quality, ensuring it is protected from the elements and positioned appropriately.

This project enables you to monitor and analyze air quality in real-time or over extended periods, helping you make informed decisions about environmental conditions and potential health risks.

CIRCUIT DIAGRAM



PROGRAM

```
#!/usr/bin/python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)



'''
define pin for lcd
'''
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1
```

```
buzzer=37

GPIO.setup(buzzer, GPIO.OUT)


# Define GPIO to LCD mapping

LCD_RS = 7

LCD_E  = 11

LCD_D4 = 12

LCD_D5 = 13

LCD_D6 = 15

LCD_D7 = 16

gas_Sensor = 18

red_light = 31


Buzzer= 29

GPIO.setup(LCD_E, GPIO.OUT)  # E

GPIO.setup(LCD_RS, GPIO.OUT) # RS

GPIO.setup(LCD_D4, GPIO.OUT) # DB4

GPIO.setup(LCD_D5, GPIO.OUT) # DB5

GPIO.setup(LCD_D6, GPIO.OUT) # DB6


GPIO.setup(LCD_D7, GPIO.OUT) # DB7

GPIO.setup(gas_Sensor, GPIO.IN) # DB7

GPIO.setup(red_light, GPIO.OUT)

GPIO.setup(Buzzer, GPIO.OUT)

# Define some device constants

LCD_WIDTH = 16    # Maximum characters per line

LCD_CHR = True

LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
```

```
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line


'''
Function Name :lcd_init()
Function Description : this function is used to initialized lcd by sending the different commands
'''
def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)
'''
Function Name :lcd_byte(bits ,mode)
Fuction Name :the main purpose of this function to convert the byte data into bit and send to lcd port
'''
def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command

  GPIO.output(LCD_RS, mode) # RS

  # High bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

  # Low bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
    GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
    GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
    GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
    GPIO.output(LCD_D7, True)
  # Toggle 'Enable' pin
  lcd_toggle_enable()
'''
Function Name : lcd_toggle_enable()
Function Description:basically this is used to toggle Enable pin
'''
def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
'''
Function Name :lcd_string(message,line)
Function  Description :print the data on lcd
'''
def lcd_string(message,line):
  # Send string to display

  message = message.ljust(LCD_WIDTH," ")

  lcd_byte(line, LCD_CMD)

  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)

lcd_init()
lcd_string("welcome ",LCD_LINE_1)
```

```
time.sleep(1)

# Define delay between readings

delay = 5


while 1:

 # Print out results

 if GPIO.input(gas_Sensor):

  lcd_string("AIR QUALITY IS HIGH",LCD_LINE_1)

  GPIO.output(Buzzer, True)

  GPIO.output(red_light, True)

 else:

   lcd_string("AIR QUALITY IS NORMAL",LCD_LINE_1)

   GPIO.output(Buzzer, False)

   GPIO.output(red_light, False)
```

S.NO:	NAME	NAAN MUDHALVAN ID	E-MAIL ID
1	R.Nivesh	au820321106029	nivesheceece@gmail.com
2	R.Raghulgandh	au820321106030	raghulgandh4@gmail.com
3	M.Shyam vasiharan	au820321106035	shyamvasiharan@gmail.com
4	S.Vishnuprakash	au820321106040	alpvishnu00@gmail.com