PHASE 5

PROJECT DOCUMENTATION AND SUMBISION

PROJECT OBJECTIVE

The main objective of the project is monitoring the air quality parameters using the IOT technology and display the real time air quality parameters to the public for awareness about the impact on their human health and air quality of our environment

Our project is ,to measure a AIR QUALITY PARAMETERS of our environment using a IOT technology.Air quality sensor(MQ-135) is used to measure the parameters in air like., NO2, CO, O3, and SO2 and create a web platform that provides a real-time air quality information to the public. The collected datas raising awareness to the public about the air quality and its impact on public health . Display real-time air quality information on the web platform, providing updates at regular intervals (e.g., every 15 minutes) to ensure users have monitoring the the current data. Implement an alert system to notify users of significant air quality changes or events, such as high pollution levels or health advisories. Allow users to download raw air quality data for research and analysis purposes. Implement an alert system to notify users of significant air quality changes or events, such as high pollution levels or health advisories.

IOT DEVICE SET UP

APPARATUS 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

import BlynkLib

import smbus

#import serial

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

# Initialize Blynk

blynk = BlynkLib.Blynk('<YOUR\_AUTH\_TOKEN>')

# Initialize I2C bus

bus = smbus.SMBus(1) # Use '0' instead of '1' for older Raspberry Pi models

'''

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

# Read air quality data from the sensor

def read\_air\_quality():

try:

data = bus.read\_i2c\_block\_data(gas\_Sensor )

air\_quality = data[0] \* 256 + data[1]

return air\_quality

except Exception as e:

print("Error reading air quality:", str(e))

return None

while 1:

blynk.run()

# Print out results

if GPIO.input(gas\_Sensor):

lcd\_string("AIR IS CONTAMINATED ",LCD\_LINE\_1)

GPIO.output(Buzzer, True)

GPIO.output(red\_light, True)

else:

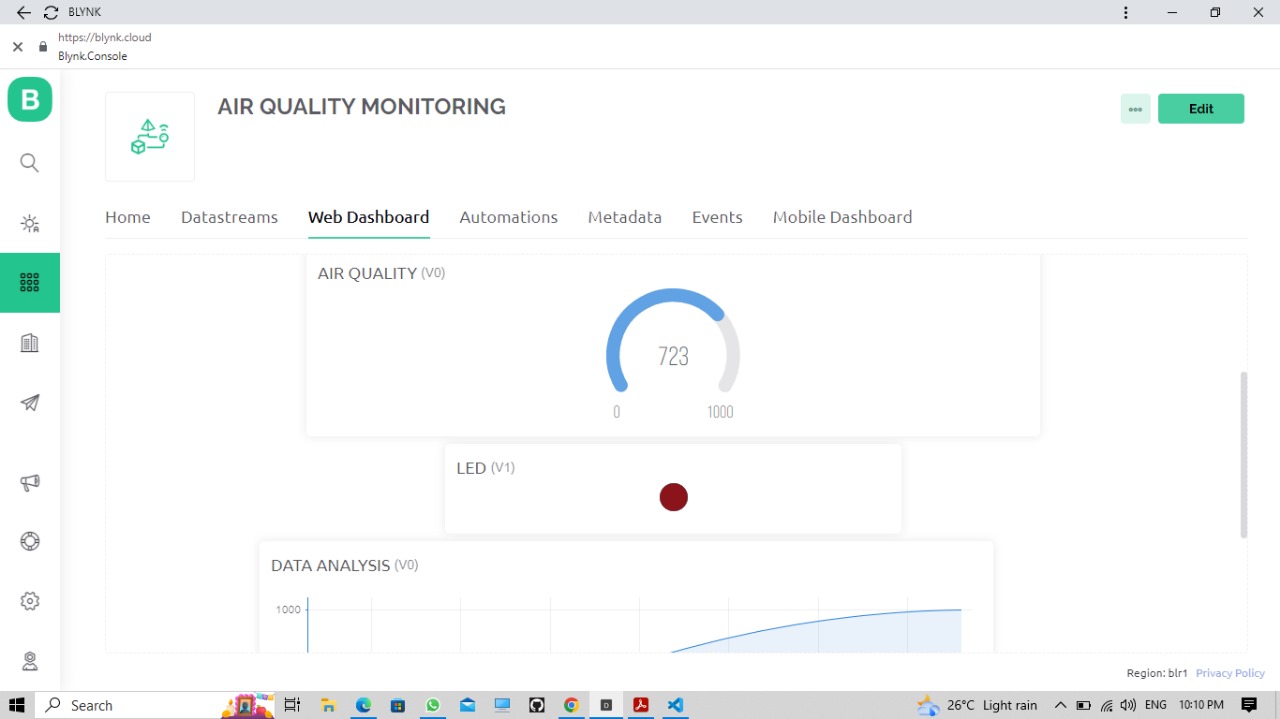
lcd\_string("AIR IS GOOD ",LCD\_LINE\_1)

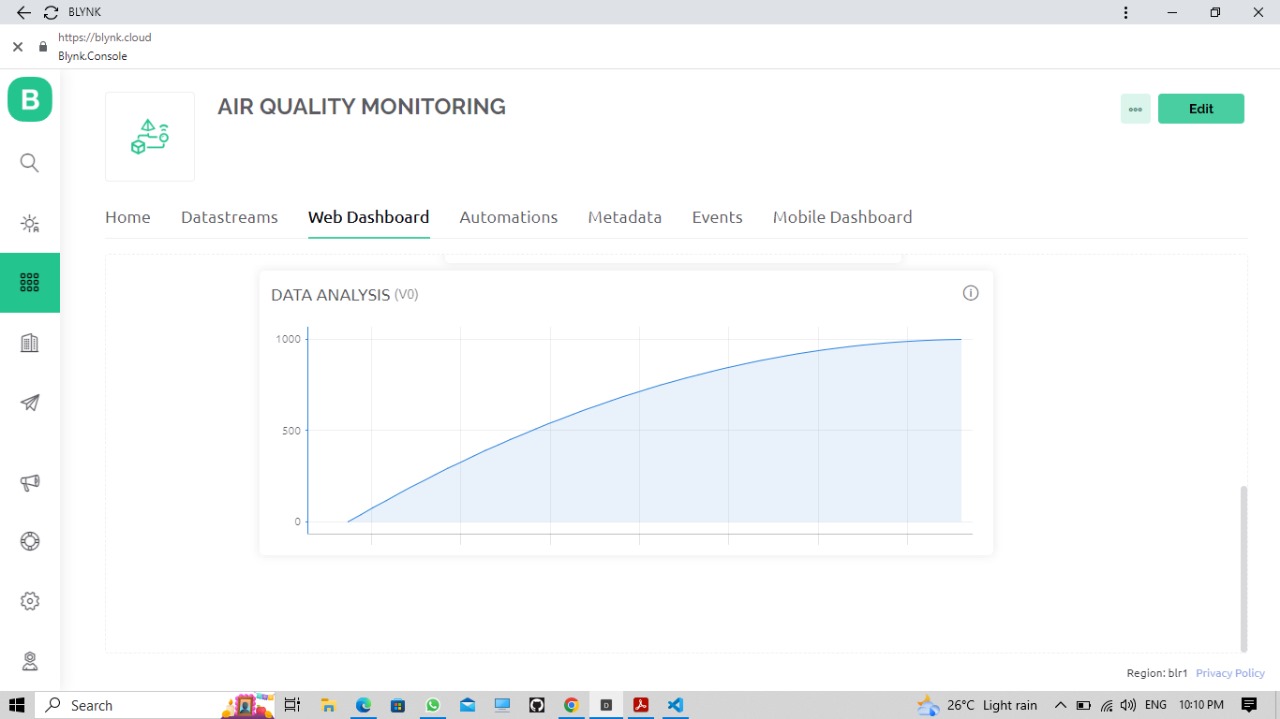
GPIO.output(Buzzer, False)

GPIO.output(red\_light, False)

blynk.virtual\_write(0, air\_quality) # Send data to Virtual Pin V0 in the Blynk app

DATA ANALYSIS





CONCLUSION:

Allow users to download raw air quality data for research and analysis purposes. Implement an alert system to notify users of significant air quality changes or events, such as high pollution levels or health advisories. Allow users to download raw air quality data for research and analysis purposes. The Public Air Quality Monitoring Web platform aims to empower the public with real-time information about air quality, enabling individuals, communities to make informed decisions to safeguard their health and the environment. By fostering transparency and awareness, the project contributes to improving air quality and public health.

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