**PHASE-4**

**DEVELOPMENT PART –2**

**Data Sharing Platform:**

We devoloping a Web platform for display the air quality and creating awareness to the public. WiFi module is used to send the data from the device to Web platform.e air quality and creating awareness to the public. WiFi module is used to send the data from the device to Web platform.

**Alerts and actions :**

Setup alert for different air quality levels and define action to be taken when certain threshold are crossed ,for example ,if the model predict poor air quality for the next day the system can triggered action like incase ventilation , air purification, or issuing health advisories.

**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:

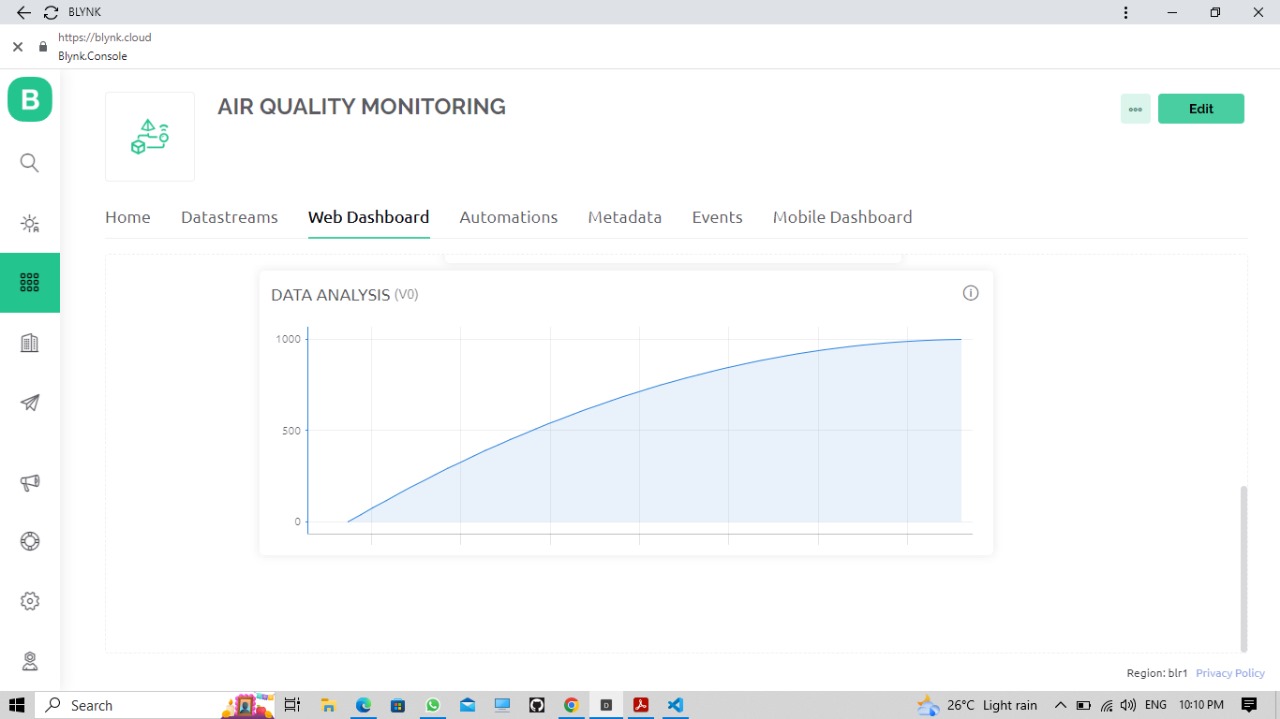
lcd\_string("AIR IS GOOD ",LCD\_LINE\_1)

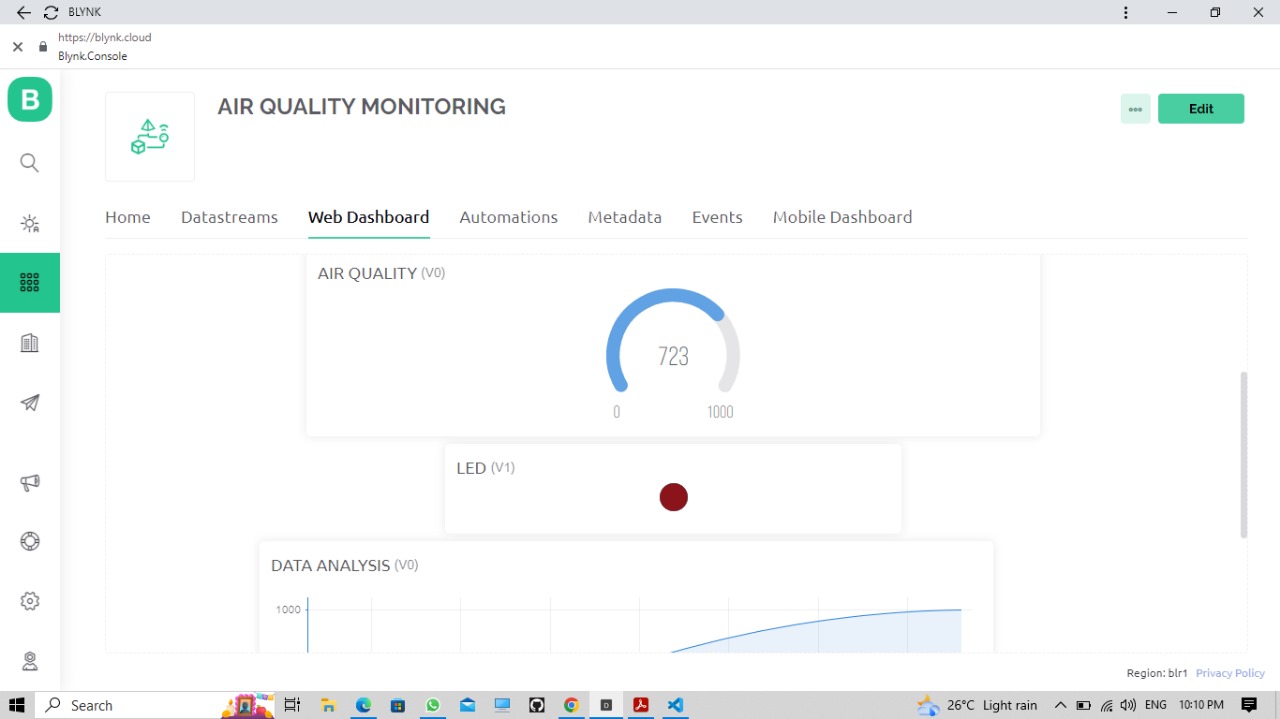
GPIO.output(Buzzer, False)

GPIO.output(red\_light, False)

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

**Data sharing platform :**





|  |  |  |  |
| --- | --- | --- | --- |
| **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** |