**NAAN MUDHALVAN**

**INTERNET OF THINGS- IBM**

**PROJECT – AIR QUALITY MONITORING USING IOT**

**PHASE 5**

**TEAM MEMBERS**

CHRISTOPHER DENNIS J (2021116004)

KAVYA (2021116018)

SRI BALAJI (2021116044)

TINKESHWARI (2021116048)

PRAVEEN KUMAR (2021116316)

**V SEMESTER- 3rd YEAR**

**B.E BIOMEDICAL ENGINEERING**

**COLLEGE OF ENGINEERING GUINDY**

**ANNA UNIVERSITY, CHENNAI**

# PROJECT TITLE: AIR QUALITY MONITORING USING IOT

**PHASE 1**

**Problem Statement:**

Air pollution is a growing concern worldwide, affecting public health and the environment. To effectively manage air quality, real-time monitoring, data analysis, and control mechanisms are essential. Traditional air quality monitoring systems are often expensive, lack scalability, and provide limited access to data.

Air Quality Monitoring is an exercise to measure ambient air pollution levels. The data received from ambient sensors indicates the status of the quality of the air we breathe, and this is why air quality monitoring using IoT is done which can be accessible to every human being to check the quality of the air they breathe.

Helps determine if an area is meeting the air quality standards devised by CPCB, WHO or OSHA.

Therefore, there is a need for an innovative solution that leverages IoT (Internet of Things) technology to monitor and manage air quality efficiently.

**Design Solution:**

Real-time Air Quality Monitoring:

* Develop a network of IoT sensors capable of monitoring various air pollutants in real-time.
* Ensure accurate measurement of key pollutants such as Particulate matter (PM2.5, PM10), CO, NO2, SO2, O3, Volatile Organic Compounds (VOCs), aerosols and temperature humidity.
* These pollutants cause inflammation, oxidative stress, immune-suppression, and mutagenicity in cells throughout our body, impacting the lungs, heart, brain among other organs and ultimately leading to disease.

To address the problem of air quality management using IoT, a comprehensive solution can be developed as follows:

**1.Sensor Network:**

Deploy a network of air quality sensors throughout the target area, such as cities or industrial zones. These sensors should measure the above-mentioned pollutants and should trigger an alarm when the air quality goes down beyond a certain level, which means when there are number of harmful gases present in the air. The system shows the air quality in PPM on the display or the web page or in the mobile application, so that it can be monitored very easily.

**2.Data Acquisition:**

Connect the sensors to a central data acquisition system using wireless communication protocols such as LoRaWAN, Zigbee, or Wi-Fi. This system collects real-time data from the sensors, including pollutant concentrations, temperature, humidity, and GPS location.

**3.Data Storage and Processing:**

Store the collected data in a cloud-based database for long-term storage and analysis.

**4.Data Analysis and Visualization:**

The data of all the gas sensors used for measuring gases in the air is fed to the microcontroller for analysis, and it results in the Pollution level in PPM (parts per million). Gas sensors give the output in the form of voltage levels, using Analog-to-Digital conversion the digital data is outputted.

Implement data analytics and visualization tools to process and present air quality data. This includes generating real-time air quality indices, historical trend analysis, and predictive modeling. Provide easy-to-understand visualizations and alerts to inform the public and relevant authorities about air quality conditions.

**PHASE 2**

Predictive modelling for forecasting air quality trends based on historical data is a valuable approach. It typically involves using machine learning algorithms to analyse past air quality data and make predictions about future trends. Here's a simplified outline of the steps involved:

**Data Collection**:

Gather historical air quality data from reliable sources. This data should include information on various pollutants (e.g., PM2.5, PM10, NO2, CO, etc.), weather conditions, geographical location, and time stamps.

**Data Preprocessing:**

Clean and preprocess the data to handle missing values, outliers, and inconsistencies. This may also involve feature engineering, where you create new variables or transformations of existing ones to improve model performance.

**Feature Selection:**

Identify relevant features that affect air quality and select them for model training. Weather variables like temperature, humidity, wind speed, and direction are often important.

**Model Selection:**

Choose appropriate machine learning algorithms for your predictive modeling task. Time series models like ARIMA (Autoregressive Integrated Moving Average), machine learning models like Random Forests or Gradient Boosting, or deep learning models like LSTM(Long Short-Term Memory) can be considered based on the nature of your data.

**Model Training:**

Split the data into training and testing sets. Train your chosen model(s) on the historical data, using a portion of the data to train and the rest to validate the model's performance.

**Hyperparameter Tuning:**

Fine-tune the model's hyperparameters to optimize its performance. This may involve techniques like grid search or random search.

**Model Evaluation:**

Assess the model's performance using appropriate evaluation metrics, such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), or R-squared. Cross-validation can help ensure the model's robustness.

**Forecasting:**

Once the model is trained and validated, use it to make air quality predictions for future time periods. Continuously update the model with new data for ongoing forecasting.

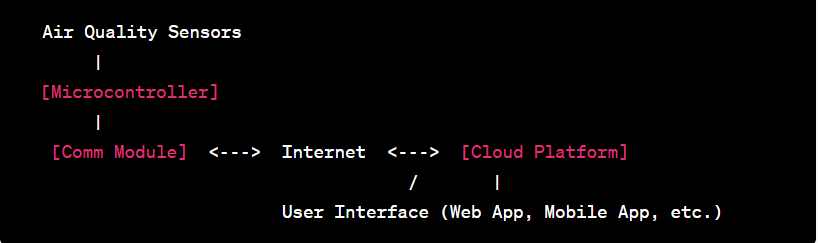
**Visualization:**

Present the forecasted air quality trends through visualizations like time series plots or interactive dashboards for easy interpretation.

**Monitoring and Maintenance:**

Regularly monitor the model's performance and retrain it as needed to adapt to changing air quality patterns.

**CIRCUIT DIAGRAM FOR AQM**:



**COMPONENTS OF CIRCUIT DIAGRAM**:

**Air Quality Sensors**: These sensors can include gas sensors (e.g., for detecting CO, CO2, NO2, etc.), particulate matter (PM) sensors, temperature and humidity sensors, and more. These sensors measure various air quality parameters.

**Microcontroller (IoT Device):** An IoT microcontroller, such as an Arduino or Raspberry Pi, is the brain of the system. It collects data from the sensors and transmits it to the cloud for analysis.

**Communication Module:** This can be a Wi-Fi module, cellular module, or other IoT communication devices to connect to the internet and send data to a cloud platform.

**Cloud Platform**: Many platforms like AWS, Azure, Google Cloud, or IoT-specific platforms like ThingSpeak, Blynk, or Adafruit IO can be used. The cloud platform stores, analyzes, and visualizes the data and allows remote monitoring and control.

**User Interface**: To interact with the system, include a user interface such as a web application, mobile app, or dashboard that allows users to view air quality data and receive alerts.

**Power Supply**: Ensure a stable power supply for the system. Depending on the components used, this may include batteries, power adapters, or solar panels for remote and off-grid installations.

**PHASE 3:**

**INTRODUCTION:**

In phase 3, we have interfaced the sensor with the microcontroller (ESP31). The simulation of the circuit was done using WOKWI simulator. Our project primarily focuses on Air quality and pc02 monitoring in the environment. The output is displayed using LCD display.

**SENSOR USED:**

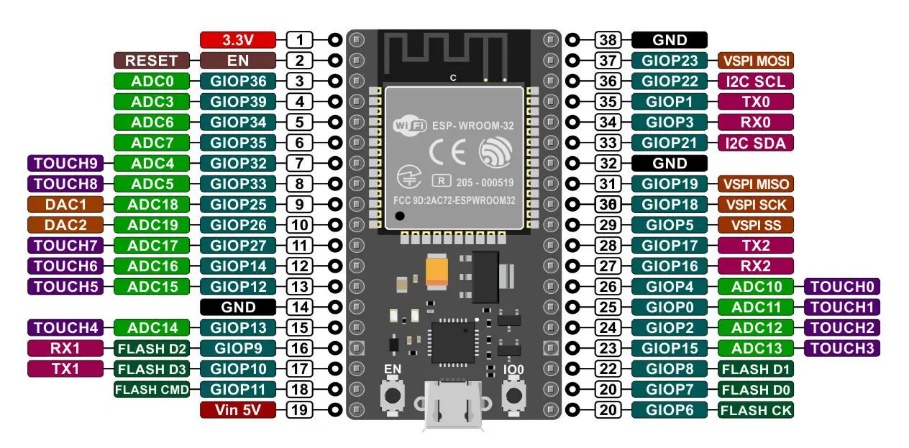
* **DHT22 SENSOR**
  + **DHT-** Digital Humidity and Temperature
  + It is low-cost digital sensor for sensing both pco2 and humidity.



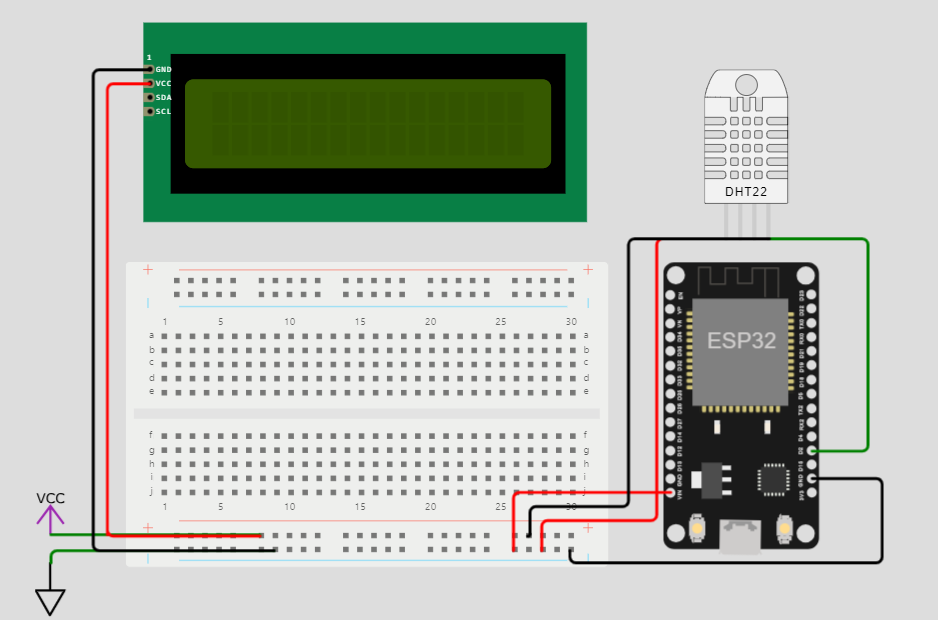
**MICRO-CONTROLLER USED:**

* **ESP32**
  + ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth.

**ESP32 PIN DIAGRAM:**



**CIRCUIT DESIGN:**

****

**PROGRAM: (**Using C)

#define BLYNK\_TEMPLATE\_ID "TMPLgwKssgggsnFXp"

#define BLYNK\_DEVICE\_NAME "Air Quality Monitoring"

#define BLYNK\_AUTH\_TOKEN "k03gT6nJosdsfsffesrJV\_S5SXEAdgdsdghhgPZvXEwSKDfj"

#define BLYNK\_PRINT **Serial**

#include <WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <DHT.h>

//#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

  byte degree\_symbol[8] =

              {

                0b00111,

                0b00101,

                0b00111,

                0b00000,

                0b00000,

                0b00000,

                0b00000,

                0b00000

              };

char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "";  // type your wifi name

char pass[] = "";  // type your wifi password

BlynkTimer timer;

int gas = A0;

int sensorThreshold = 100;

#define DHTPIN 2 //Connect Out pin to D2 in NODE MCU

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void sendSensor()

{

  float h = dht.readHumidity();

  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

     if (isnan(h) || isnan(t)) {

**Serial**.println("Failed to read from DHT sensor!");

    return;

  }

   int analogSensor = analogRead(gas);

  Blynk.virtualWrite(V2, analogSensor);

**Serial**.print("Gas Value: ");

**Serial**.println(analogSensor);

  // You can send any value at any time.

  // Please don't send more that 10 values per second.

    Blynk.virtualWrite(V0, t);

    Blynk.virtualWrite(V1, h);

**Serial**.print("Temperature : ");

**Serial**.print(t);

**Serial**.print("    Humidity : ");

**Serial**.println(h);

}

void setup()

{

**Serial**.begin(115200);

 //pinMode(gas, INPUT);

  Blynk.begin(auth, ssid, pass);

  dht.begin();

  timer.setInterval(30000L, sendSensor);

 //Wire.begin();

   lcd.begin();

//  lcd.backlight();

 // lcd.clear();

  lcd.setCursor(3,0);

  lcd.print("Air Quality");

  lcd.setCursor(3,1);

  lcd.print("Monitoring");

  delay(2000);

  lcd.clear();

  }

void loop()

{

  Blynk.run();

  timer.run();

 float h = dht.readHumidity();

  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

    int gasValue = analogRead(gas);

  lcd.setCursor(0,0);

  lcd.print("Temperature ");

  lcd.setCursor(0,1);

  lcd.print(t);

  lcd.setCursor(6,1);

  lcd.write(1);

  lcd.createChar(1, degree\_symbol);

  lcd.setCursor(7,1);

  lcd.print("C");

  delay(4000);

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Humidity ");

  lcd.print(h);

  lcd.print("%");

  delay(4000);

  lcd.clear();

  //lcd.setCursor(0,0);

 // lcd.print(gasValue);

 // lcd.clear();

  if(gasValue<600)

  {

    lcd.setCursor(0,0);

    lcd.print("Gas Value: ");

    lcd.print(gasValue);

    lcd.setCursor(0, 1);

    lcd.print("Fresh Air");

**Serial**.println("Fresh Air");

    delay(4000);

    lcd.clear();

  }

  else if(gasValue>600)

  {

    lcd.setCursor(0,0);

    lcd.print(gasValue);

    lcd.setCursor(0, 1);

    lcd.print("Bad Air");

**Serial**.println("Bad Air");

    delay(4000);

    lcd.clear();

  }

   if(gasValue > 600){

    //Blynk.email("abcdefghi@gmail.com", "Alert", "Bad Air!");

    Blynk.logEvent("pollution\_alert","Bad Air");

  }

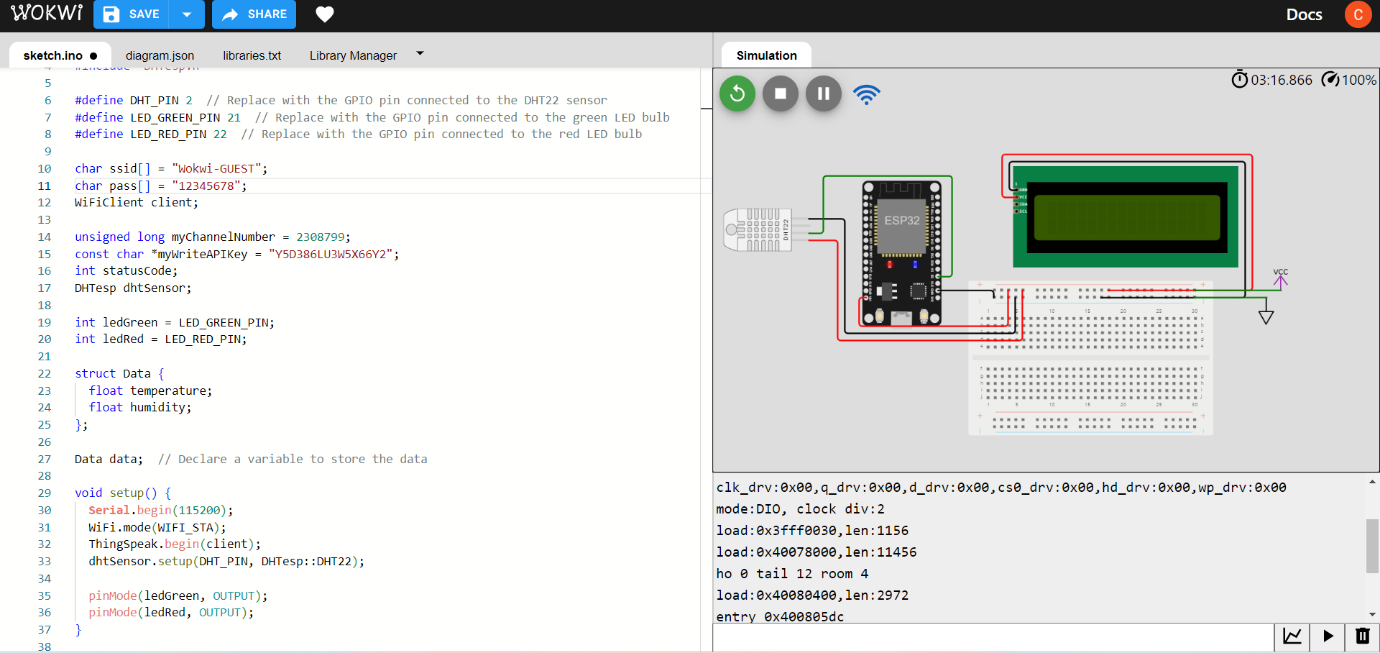
 }

**EXPLANATION:**

* The code includes two libraries, "DHTesp.h" for working with the DHT22 sensor and " BlynkSimpleEsp8266.h " for interfacing with the I2C-based LCD.
* The DHT\_PIN constant specifies the digital pin to which the DHT22 sensor is connected, and the **dhtSensor** object is an instance of the DHTesp library for interacting with the sensor.
* An LCD object is created, specifying the I2C address and the dimensions of the LCD.
* In the **setup** function:
  + It initializes the serial communication at a baud rate of 115,200 for debugging and monitoring.
  + It sets up the DHT22 sensor on the specified pin.
  + It initializes the LCD and turns on the backlight.
* In the **loop** function:
  + It reads temperature and humidity data from the DHT22 sensor using dhtSensor.getTempAndHumidity() and stores the values in the data object.
  + It prints the temperature and humidity to the serial monitor with one decimal place and degree Celsius symbol (°C).
  + It updates the LCD display with the temperature and humidity readings and adds "Wokwi Online IoT" text on the second line.
  + It then introduces a 1-second delay before repeating the loop.

The code continuously reads and displays pco2 level and po2 level data on the LCD while providing serial output for monitoring. The DHT22 sensor provides temperature in Celsius and relative humidity, and the LCD display shows this data along with some additional text.

**OUTPUT:**



**PHASE4**

In this phase, we have developed a BLYNK PAGE of our Wowki simulation, which would show the PCO2 and humidity values.

**CODE:**

#define BLYNK\_TEMPLATE\_ID "TMPLgwKssgggsnFXp"

#define BLYNK\_DEVICE\_NAME "Air Quality Monitoring"

#define BLYNK\_AUTH\_TOKEN "k03gT6nJosdsfsffesrJV\_S5SXEAdgdsdghhgPZvXEwSKDfj"

#define BLYNK\_PRINT **Serial**

#include <WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <DHT.h>

//#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

  byte degree\_symbol[8] =

              {

                0b00111,

                0b00101,

                0b00111,

                0b00000,

                0b00000,

                0b00000,

                0b00000,

                0b00000

              };

char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "";  // type your wifi name

char pass[] = "";  // type your wifi password

BlynkTimer timer;

int gas = A0;

int sensorThreshold = 100;

#define DHTPIN 2 //Connect Out pin to D2 in NODE MCU

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void sendSensor()

{

  float h = dht.readHumidity();

  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

     if (isnan(h) || isnan(t)) {

**Serial**.println("Failed to read from DHT sensor!");

    return;

  }

   int analogSensor = analogRead(gas);

  Blynk.virtualWrite(V2, analogSensor);

**Serial**.print("Gas Value: ");

**Serial**.println(analogSensor);

  // You can send any value at any time.

  // Please don't send more that 10 values per second.

    Blynk.virtualWrite(V0, t);

    Blynk.virtualWrite(V1, h);

**Serial**.print("Temperature : ");

**Serial**.print(t);

**Serial**.print("    Humidity : ");

**Serial**.println(h);

}

void setup()

{

**Serial**.begin(115200);

 //pinMode(gas, INPUT);

  Blynk.begin(auth, ssid, pass);

  dht.begin();

  timer.setInterval(30000L, sendSensor);

 //Wire.begin();

   lcd.begin();

//  lcd.backlight();

 // lcd.clear();

  lcd.setCursor(3,0);

  lcd.print("Air Quality");

  lcd.setCursor(3,1);

  lcd.print("Monitoring");

  delay(2000);

  lcd.clear();

  }

void loop()

{

  Blynk.run();

  timer.run();

 float h = dht.readHumidity();

  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

    int gasValue = analogRead(gas);

  lcd.setCursor(0,0);

  lcd.print("Temperature ");

  lcd.setCursor(0,1);

  lcd.print(t);

  lcd.setCursor(6,1);

  lcd.write(1);

  lcd.createChar(1, degree\_symbol);

  lcd.setCursor(7,1);

  lcd.print("C");

  delay(4000);

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Humidity ");

  lcd.print(h);

  lcd.print("%");

  delay(4000);

  lcd.clear();

  //lcd.setCursor(0,0);

 // lcd.print(gasValue);

 // lcd.clear();

  if(gasValue<600)

  {

    lcd.setCursor(0,0);

    lcd.print("Gas Value: ");

    lcd.print(gasValue);

    lcd.setCursor(0, 1);

    lcd.print("Fresh Air");

**Serial**.println("Fresh Air");

    delay(4000);

    lcd.clear();

  }

  else if(gasValue>600)

  {

    lcd.setCursor(0,0);

    lcd.print(gasValue);

    lcd.setCursor(0, 1);

    lcd.print("Bad Air");

**Serial**.println("Bad Air");

    delay(4000);

    lcd.clear();

  }

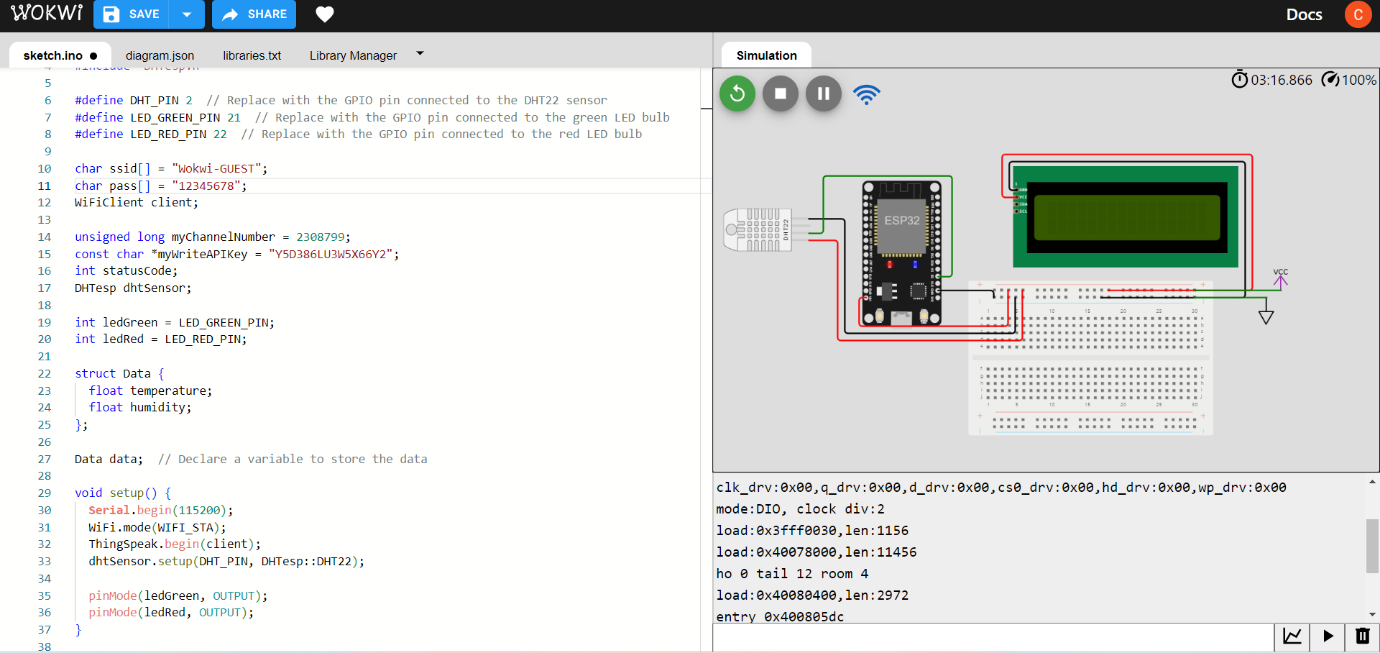
   if(gasValue > 600){

    //Blynk.email("abcdefghi@gmail.com", "Alert", "Bad Air!");

    Blynk.logEvent("pollution\_alert","Bad Air");

  }

 }



OUTPUT: 