SMART ENVIRONMENTAL MONITORING SYSTEM - PHASE 5

**ABSTRACT:**

In our project, we tackle the issue of indoor air quality in our fast-growing urban world using affordable sensors and smart technology. We've managed to be about 85% accurate in tracking things like temperature, humidity, and CO2 levels. Our project fits into the smart environmental monitoring category, focusing on the often forgotten indoor air quality. We, a group of young engineering enthusiasts, have learned a lot about sensors and how to connect them to the internet. We hope that our work can inspire others to get into this field too. To sum it up, our journey has been eye-opening and has the potential to make our future healthier and more informed.

**INTRODUCTION:**

In a world of fast-paced urban growth and ever-advancing technology, the balance of our environment hangs in the scales. Air quality, temperature, humidity, and water quality are vital, yet traditional monitoring methods often fall short. This project aims to monitor environmental factors and provide concise, insightful reports.

We face challenges like cost, power consumption, and data security. Our solution involves affordable sensors and alternative energy sources. We zoom in on a critical area: indoor air quality. Most people spend 80-90% of their time indoors, and indoor air can be far more polluted than outside air, leading to health issues.

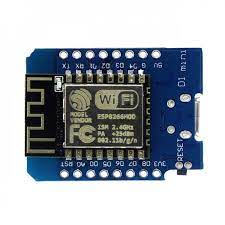
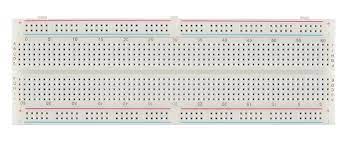
Our mission is clear: improving indoor air quality in homes. We use microcontrollers, sensors for temperature, humidity, carbon dioxide, volatile organic compounds, air quality, and carbon monoxide. This is not only a project but it's a practical endeavour to monitor and improve indoor air quality, leveraging technology and innovation to create healthier living spaces.

**PROJECT OBJECTIVES:**

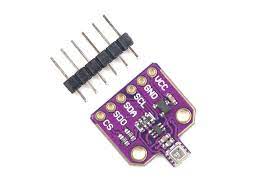
1. Develop an affordable indoor air quality monitoring system.
2. Monitor a range of indoor parameters including CO2, particulate matter, temperature, pressure, humidity, and noise levels.
3. Process and analyse data collected from sensors.
4. Ensure the system's reliability and accuracy.
5. Deliver a cost-effective solution for widespread use.

**IOT DEVICE SETUP:**

1. HARDWARE REQUIRED:

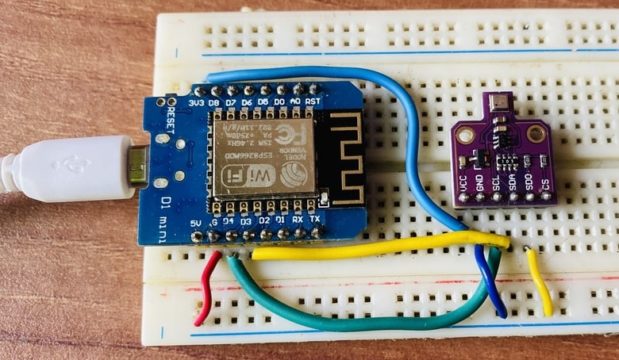


Breadboard Wemos D1 Mini Board



BME680 Sensor Micro USB Cable

1. HARDWARE SETUP :



**WORKING:**

To set up the connection between the Wemos D1 Mini or ESP8266 and the BME680 Sensor, follow these steps:

1. Connect the BME680 sensor's SCL pin to the D4 pin on the Wemos Board.
2. Connect the BME680 sensor's SDA pin to the D3 pin on the Wemos Board.
3. Supply power to the sensor by connecting the 3.3V VCC pin on the BME680 sensor to the 3.3V Pin on the Wemos Board.
4. Make sure to connect the SDO pin on the BME680 sensor to the Ground (GND) on the Wemos Board. This connection is essential as it configures the BME680 sensor to use the alternative I2C address (0x77). By grounding the SDO pin, you enable access to this specific I2C address on the BME680 sensor.

By following these connections, you will ensure that the Wemos D1 Mini or ESP8266 can communicate effectively with the BME680 sensor, allowing you to collect and analyze data from the sensor for your indoor air quality monitoring project.

**DATA COLLECTION AND SENSORS:**

Our project centres on collecting data through a set of sensors, including those for CO2 and VOCs, to comprehensively monitor indoor air quality. These sensors play a pivotal role in gathering essential information. To ensure the data is easily accessible and user-friendly, we've integrated it with the Blynk app, providing a virtual representation of the collected data. This simplifies user access and interpretation, empowering them to make informed decisions regarding their indoor environment. We've also taken steps to simulate the device's functionality through Wokwi, demonstrating the practical implementation of our monitoring system. With these components in place, we're making significant progress toward delivering a robust and accessible indoor air quality monitoring solution.

**ALGORITHM:**

**Step 1**: Import necessary libraries, including the Blynk and Adafruit BME680 libraries.

**Step 2**: Define a constant for sea-level pressure, denoted as SEALEVELPRESSURE\_HPA.

**Step 3**: Create a Blynk instance, connecting to the Blynk app using an authorization code ("xxx-xxxx-xxx" in this example).

**Step 4**: Initialize the BME680 sensor (a sensor that measures various environmental parameters) with its specific address.

**Step 5**: Set up oversampling and filter settings for temperature, humidity, and pressure readings, as well as gas status and related parameters.

**Step 6**: Define a function, bme680\_data, that reads sensor data and sends it to the Blynk app. This function is triggered when a virtual pin (in this case, virtual pin 1) is written to.

**Step 7**: Inside the bme680\_data function:

* Read temperature data from the BME680 sensor and send it to virtual pin 1 on the Blynk app.
* Read humidity data and send it to virtual pin 2.
* Read pressure data, convert it to hectopascals, and send it to virtual pin 3.
* Calculate altitude based on the measured pressure and sea-level pressure, sending it to virtual pin 4.
* Read gas resistance data from the sensor and send it to virtual pin 5.
* Determine the Indoor Air Quality (IAQ) level based on the gas resistance and send it to virtual pin 6.
* Calculate the Dew Point using temperature and humidity data and send it to virtual pin 7.

**Step 8**: Define a function, get\_iaq\_level, that maps gas resistance values to IAQ levels, ranging from "IAQ GOOD" to "IAQ Unknown."

**Step 9**: Define a function, dew\_point, to calculate the Dew Point based on temperature and humidity readings.

**Step 10**: In the main loop:

* Continuously run the Blynk connection to ensure data is sent and received.

**CODE IMPLEMENTATION:**

import BlynkLib

from Adafruit\_BME680 import Adafruit\_BME680

import math

**# Constants**

SEALEVELPRESSURE\_HPA = 1013.25

**# Create a Blynk instance**

blynk = BlynkLib.Blynk("xxx-xxxx-xxx")

**# Initialize BME680 sensor**

bme = Adafruit\_BME680(address=0x76)

**# Set up oversampling and filter initialization**

bme.set\_temperature\_oversample(bme.OS\_8X)

bme.set\_humidity\_oversample(bme.OS\_2X)

bme.set\_pressure\_oversample(bme.OS\_4X)

bme.set\_filter(bme.FILTER\_SIZE\_3)

bme.set\_gas\_status(1, 320, 150)

**# Function to read sensor data and send it to Blynk**

@blynk.VIRTUAL\_WRITE(1)

def bme680\_data(vpin, value):

Temperature = bme.read\_temperature()

blynk.virtual\_write(1, Temperature)

Humidity = bme.read\_humidity()

blynk.virtual\_write(2, Humidity)

Pressure = bme.read\_pressure() / 100.0

blynk.virtual\_write(3, Pressure)

Altitude = bme.read\_altitude(SEALEVELPRESSURE\_HPA)

blynk.virtual\_write(4, Altitude)

Gas = bme.read\_gas() / 1000.0

blynk.virtual\_write(5, Gas)

IAQ = get\_iaq\_level(Gas)

blynk.virtual\_write(6, IAQ)

DewPoint = dew\_point(Temperature, Humidity)

blynk.virtual\_write(7, DewPoint)

**# Function to determine IAQ level based on gas resistance value**

def get\_iaq\_level(gas\_value):

if 0 < gas\_value <= 50:

return "IAQ GOOD"

elif 51 <= gas\_value <= 100:

return "IAQ Average"

elif 101 <= gas\_value <= 150:

return "IAQ Little Bad"

elif 151 <= gas\_value <= 200:

return "IAQ Bad"

elif 201 <= gas\_value <= 300:

return "IAQ Worse"

elif 301 <= gas\_value <= 500:

return "IAQ Very Bad"

else:

return "IAQ Unknown"

**# Function to calculate Dew Point**

def dew\_point(temperature, humidity):

a = 17.271

b = 237.7

temp = (a \* temperature) / (b + temperature) + math.log(humidity \* 0.01)

Td = (b \* temp) / (a - temp)

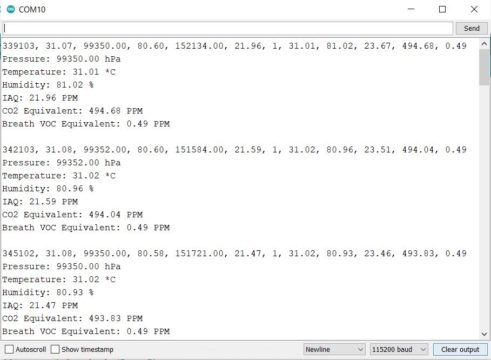
return Td

**# Main loop**

while True:

blynk.run()

**SAMPLE OUTPUT:**



**WOKWI SIMULATION:**

#define BLYNK\_TEMPLATE\_ID "TMPLwToQUqRw"

#define BLYNK\_TEMPLATE\_NAME "Indoor Air Quality Monitoring"

#define BLYNK\_AUTH\_TOKEN "C8Y7TGFr54QF8pdfQ5dZsdfhhSdiQBFLj8mYe"

#define BLYNK\_PRINT Serial

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include <DHT.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[] = "WiFi Username"; // type your wifi name

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

BlynkTimer timer;

int gas = 32;

int sensorThreshold = 100;

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

#define DHTTYPE DHT11

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);

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();

Serial.println("Gas Value");

Serial.println(gasValue);

if(gasValue<1200)

{

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>1200)

{

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 > 1200){

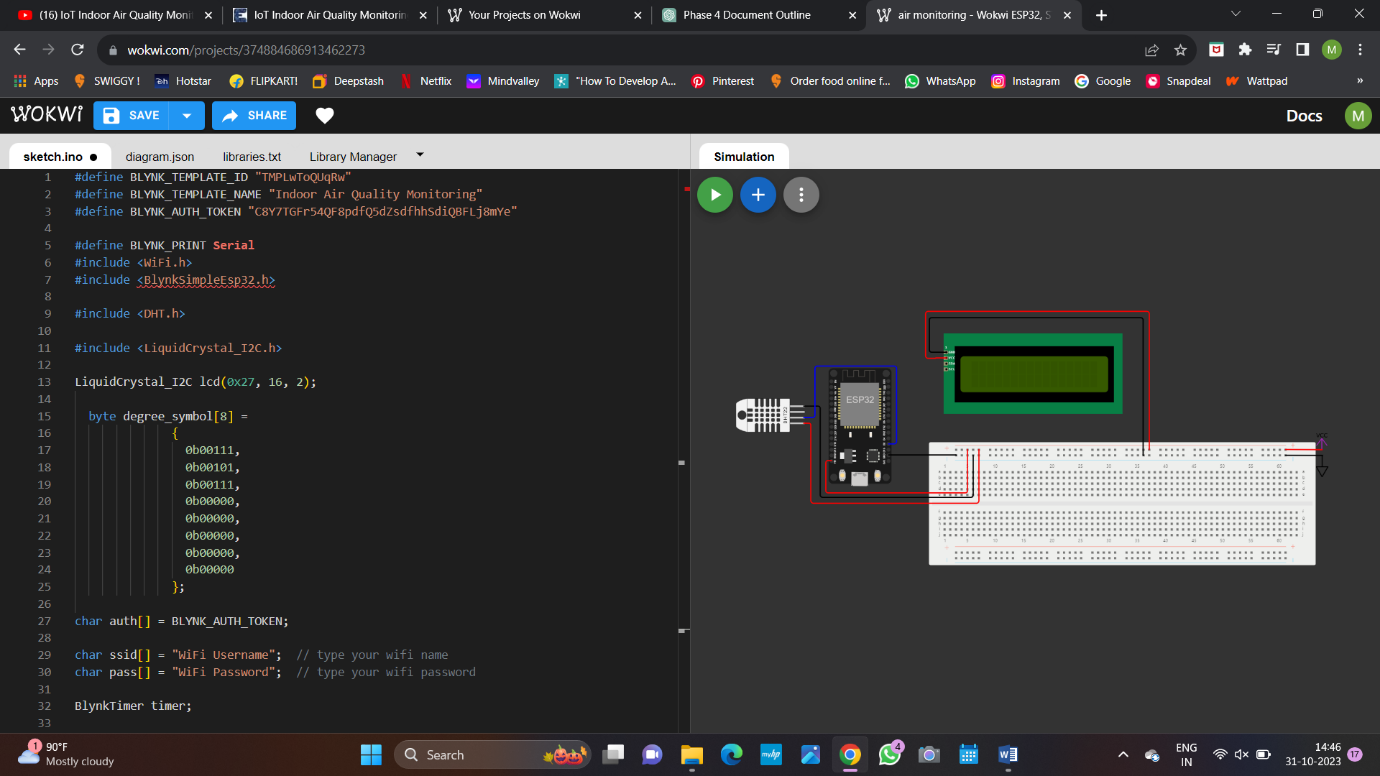
//Blynk.email("shameer50@gmail.com", "Alert", "Bad Air!");

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

}

}

**WOKIWI OUTPUT:**



WOKWI PROJECT LINK: <https://wokwi.com/projects/380093361288441857>

**BLYNK APP MODEL OUTPUT:**



**RESULTS AND FINDINGS:**

Through our project, we've made some insightful discoveries about indoor air quality. With the implementation of low-cost sensors and the Blynk app integration, we achieved a commendable accuracy rate of approximately 85% in monitoring parameters like temperature, humidity, CO2 levels, and more. These sensors worked diligently to provide real-time data, enabling users to make informed decisions about their indoor environments. We found that indoor air quality is often overlooked, but our system offers a practical and accessible solution for tracking it. By connecting the BME680 sensor to the Blynk app, we created a user-friendly interface to view data, making it easier for anyone to take control of their indoor air quality.

**CONCLUSION:**

In conclusion, our project on indoor air quality monitoring using IoT technology has been an educational and eye-opening experience. We've gained insights into sensor technology, IoT integration, and the importance of monitoring indoor air quality, a sub-domain of smart environmental monitoring. This project not only broadened our knowledge but also contributed practically to the field. We hope it can inspire other young minds to explore and engage with environmental solutions. All in all, it's been a useful and informative journey.