SMART ENVIRONMENTAL MONITORING

The primary objective of the code we are about to unveil is to establish a robust connection between our ESP32 microcontroller and the state-of-the-art BME680 sensor, which will serve as our sentinel for monitoring key indoor air parameters. This code will orchestrate the collection of data related to temperature, humidity, pressure, gas resistance, altitude, and the Dew Point. Subsequently, it will transmit this vital information to the cloud for comprehensive analysis and visualization.

Key Functions of the Code:

**Data Collection**: The code is responsible for interfacing with the BME680 sensor and gathering data on the environmental conditions within the monitored indoor space.

**Data Transmission**: Once collected, the code facilitates the transmission of this data to the cloud via the Blynk platform, ensuring that the information is readily accessible and can be analyzed in real-time.

**IAQ Assessment**: An integral part of the code involves assessing the Indoor Air Quality (IAQ) based on the gas resistance measurements. It categorizes IAQ into distinct levels, providing valuable insights into the air's health.

**Dew Point Calculation**: The code also calculates the Dew Point, which is a crucial metric for understanding humidity and potential condensation.

As we delve into the intricacies of the code, we will meticulously dissect its various components and functionalities. We aim not only to ensure its efficacy but also to make it adaptable for future enhancements. Once the code is complete, we will seamlessly transition into running a simulation on the Wokwi platform, where we can validate its performance in a controlled environment.

The following sections will provide a detailed breakdown of the code, explaining its structure, functions, and the steps needed to set up the simulation.

STEPS TO TAKE BEFORE RUNNING THE CODE:

To run the Python code for monitoring indoor air quality using an ESP32 and the BME680 sensor, follow these steps:

1. Install Python:

* If you don't have Python installed on your computer, download and install the latest version of Python from the official Python website:

https://www.python.org/downloads/.

2. Install Required Python Libraries:

* Open a terminal or command prompt and use pip (Python's package manager) to install the necessary libraries. Run the following commands:

pip install BlynkLib

pip install adafruit-circuitpython-bme680

3. Hardware Setup:

* Connect your ESP32 to your computer.
* Connect the BME680 sensor to the ESP32 using the appropriate pins.
* Make sure you have a stable Wi-Fi connection for the ESP32 to connect to Blynk.

4. Blynk Setup:

* Install the Blynk mobile app on your smartphone.
* Create a new project on the Blynk app.
* Obtain the authentication token for your Blynk project.

5. Code Modification:

* Replace `"xxx-xxxx-xxx"` in the Python code with your Blynk authentication token.
* Ensure that the hardware connections match your setup.

6. Run the Python Code:

* Save the Python code in a file with a `.py` extension (e.g., `air\_monitor.py`).
* Open a terminal or command prompt and navigate to the directory where the Python script is located.
* Run the script by entering the following command:

python air\_monitor.py

This will start the Python script, which will connect to your ESP32 and BME680 sensor, and it will start sending data to your Blynk project.

7. Monitor the Data:

* Open the Blynk app on your smartphone.
* In your Blynk project, you will see widgets corresponding to the data sent from the ESP32.
* You can monitor the temperature, humidity, pressure, gas resistance, altitude, IAQ, and Dew Point through the app.

SOURCE CODE:

import BlynkLib

from Adafruit\_BME680 import Adafruit\_BME680

# **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.

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()

OUTPUT:

