

Python – RaspberryPi 3 - Air Quality Sensor

Table of Contents

Python – RaspberryPi 3 - Air Quality Sensor.....	1
Project Overview.....	2
Materials Required.....	2
Setting Up the Raspberry Pi.....	2
Connecting the Hardware.....	3
Sensor Integration.....	4
Connecting the BME680 Sensor to the Raspberry Pi.....	5
Checking the Connection and Enabling I2C.....	5
Software and Libraries Installation.....	6
Creating a Virtual Environment and Installing Libraries.....	6
Python Script Creation and Execution.....	7
Example Code and Further Customization.....	8



Project Overview

This project involves reading and displaying data from an air quality sensor accessed via a shared network using a Raspberry Pi, with the help of Python. The setup of materials and preparation of the necessary environment is also included, requiring approximately four hours of work. You can easily continue your projects, such as displaying results on the web or integrating them into applications, using this infrastructure.

Materials Required

- 1 - Raspberry Pi 3 Model-B V1.2
- 2 - BME680 Air Quality Sensor
- 3 - A screen for Raspberry Pi connection
- 4 - Mouse, Keyboard, HDMI cable
- 5 - Modem and internet cable if WiFi is not used
- 6 - A second computer connected to the shared network if you want to connect via SSH
- 7 - SD card

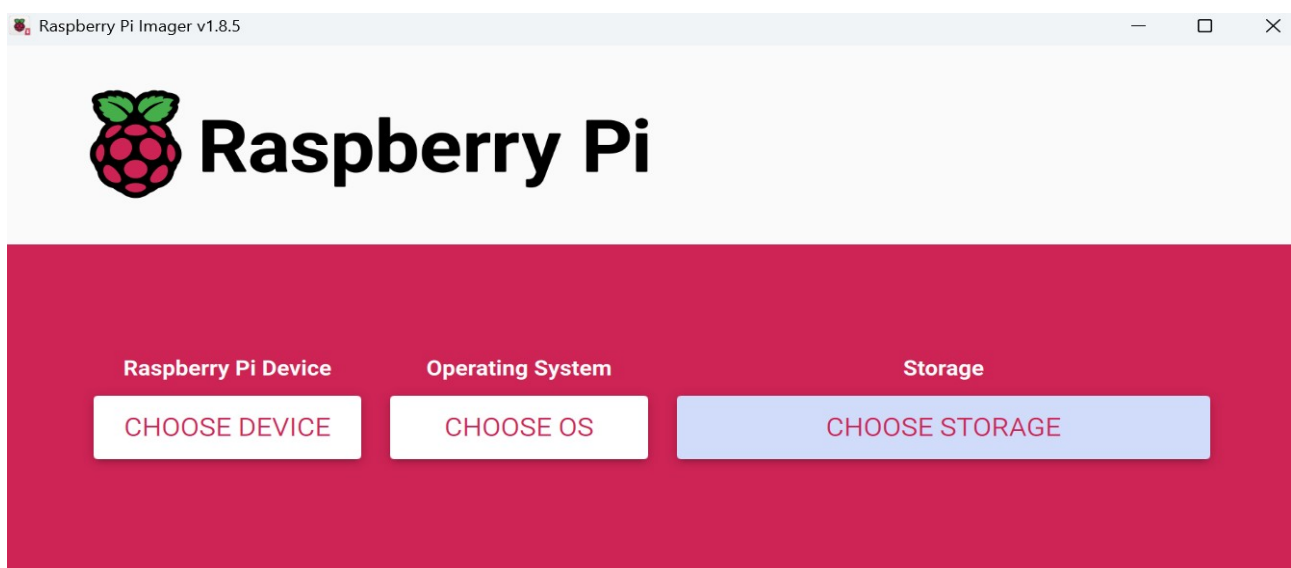
Setting Up the Raspberry Pi

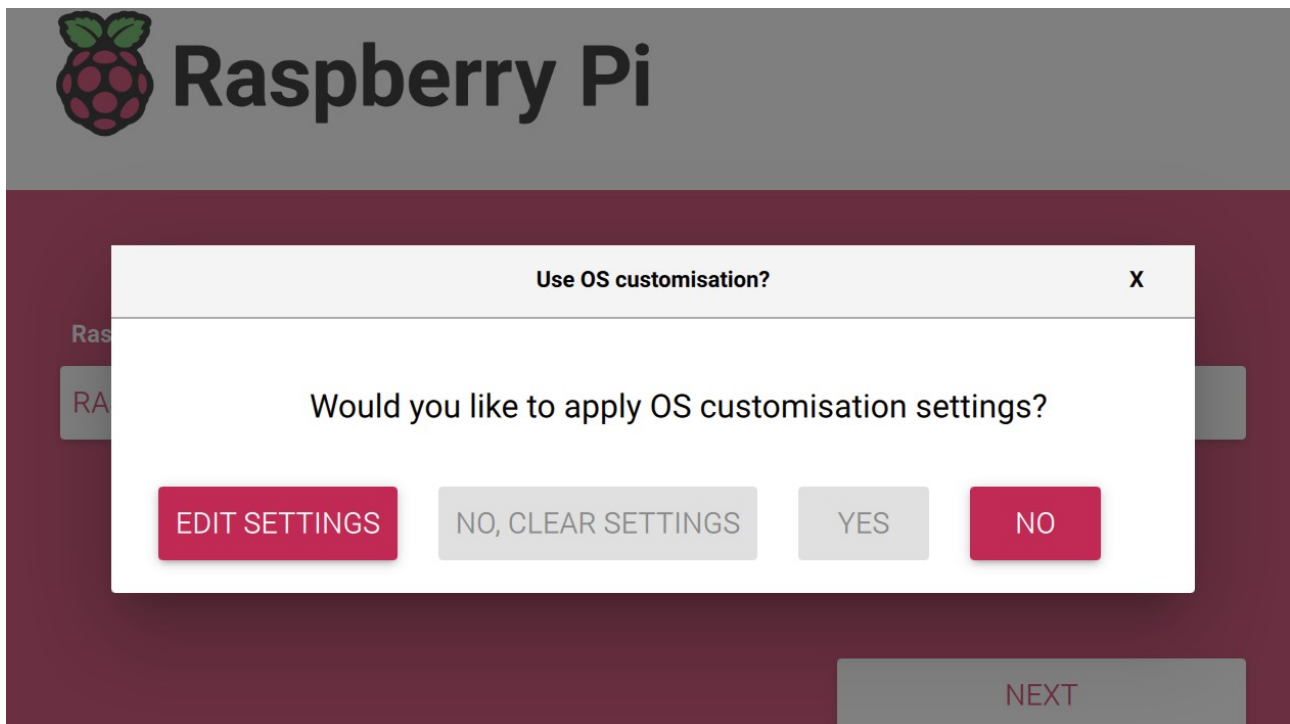
We begin by inserting our sd card into our computer for OS installation and necessary operations. Remember that the sd card must be empty at this stage, and all old data will be deleted after installation.

First, we start by downloading the image file for your operating system from the Raspberry Pi's own page: <https://www.raspberrypi.com/software/> or for Windows: type "sudo apt install rpi-imager" in a Terminal window.

After the download is complete, open the imager.exe to access the configuration screen.

Here, we select our device, the operating system, and the SD card we will install it on and proceed.





On this screen, you can decide whether to use the customization options for the operating system installation. If you want to preconfigure certain settings, you can click on the "EDIT SETTINGS" option and make the following customizations:

- Hostname: You can determine the name under which your Raspberry Pi will appear on the network.
- Enable SSH: You can activate the SSH service for remote access. This option is important for security when SSH access is needed.
- Configure wifi: If you will be using wireless internet connection, you can enter the Wi-Fi network name (SSID) and password here to enable your Raspberry Pi to connect to the internet at startup.
- Set locale settings: You can configure local settings such as language, time zone, and keyboard layout here. If you do not want to make these settings now or prefer to configure your Raspberry Pi manually later, you can continue by clicking the "NO" option. In this case, you will need to make these settings through a setup wizard when you first start your Raspberry Pi.

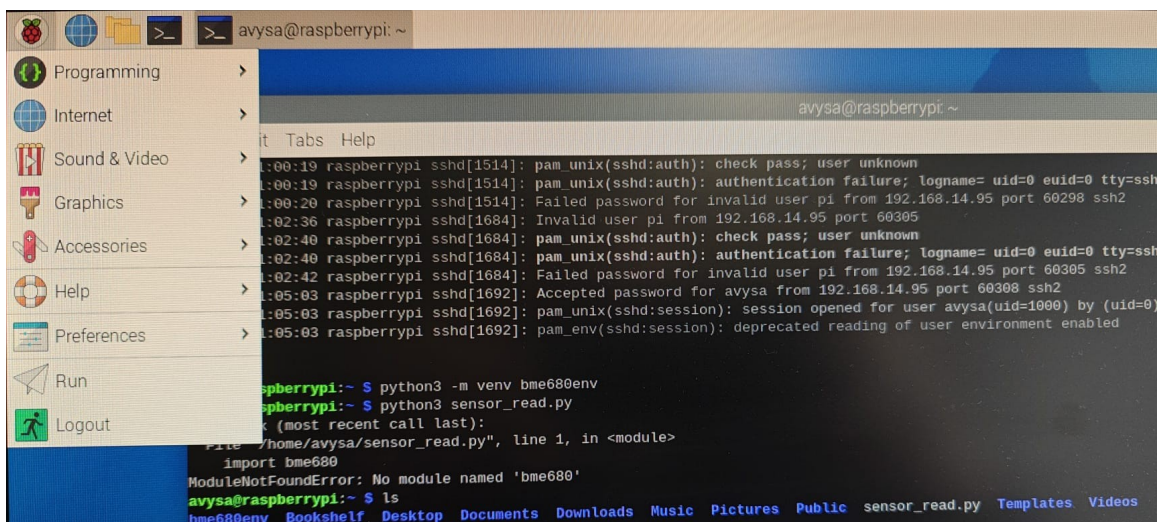
The Raspberry Pi Imager has a user-friendly interface, making the operating system selection and writing process quite simple. After the process is completed, as mentioned above, you can connect your Raspberry Pi to the hardware and start making the initial settings.

Connecting the Hardware

1. Insert the SD Card: Place the SD card you have written into the SD card slot of your Raspberry Pi.
2. Connect the Monitor: Connect your Raspberry Pi to a monitor using an HDMI cable.
3. Connect the Keyboard and Mouse: Connect a USB keyboard and mouse to your Raspberry Pi.
4. Connect the Power Supply: Finally, plug the power supply into your Raspberry Pi and turn on the device.

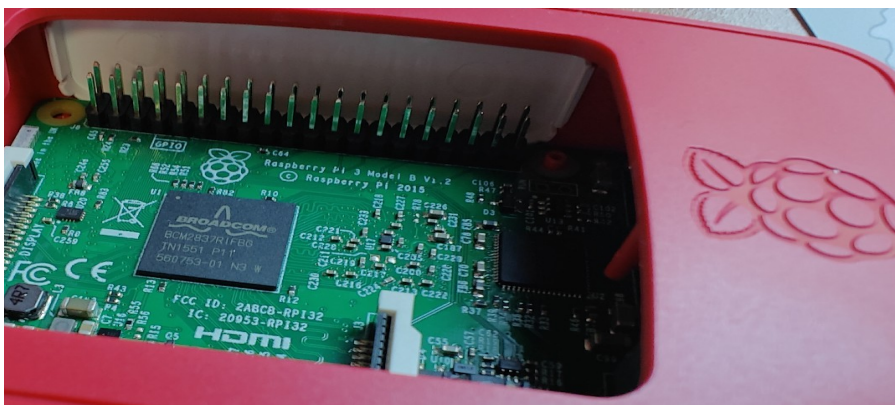
Initial Startup and System Setup When you start your Raspberry Pi for the first time, the operating system will ask you to perform some basic configurations:

1. Language and Region Settings: Choose the language and region you wish to use.
2. Network Connections: If you have not previously entered your Wi-Fi details, you can connect to a Wi-Fi network here and gain internet access.
3. User Account and Password: Set up a username and a strong password. These details will be used to log into the system.
4. System Updates: If your internet connection is active, check for system updates and install them. This can be done via the terminal with the commands "sudo apt update" and "sudo apt upgrade". After completing these steps, your Raspberry Pi will be ready to use, and you will see the Raspberry screen after the setup is complete.



Sensor Integration

Initially, we will face the question of how and where to connect the sensor's pins. In this case, let's proceed by paying attention to the order of operations and carefully placing the pins.



Connecting the BME680 Sensor to the Raspberry Pi

Let's start by connecting our BME680 sensor to the Raspberry Pi. The BME680 is typically connected via the I2C interface. This can be done using the GPIO pins of the Raspberry Pi. Below are the steps on how to connect the BME680 sensor to the Raspberry Pi:

Power Connections:

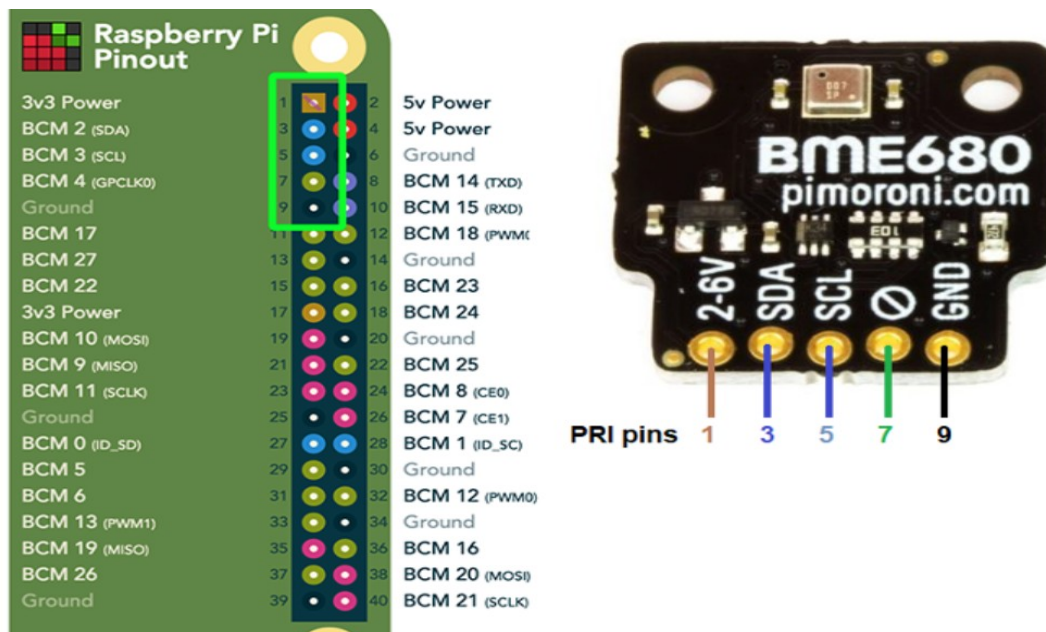
Connect the VCC pin of the BME680 to the 3.3V pin of the Raspberry Pi.

Connect the GND pin to one of the Raspberry Pi's GND pins.

I2C Connections:

Connect the SDA pin of the BME680 to the SDA pin of the Raspberry Pi (GPIO 2),

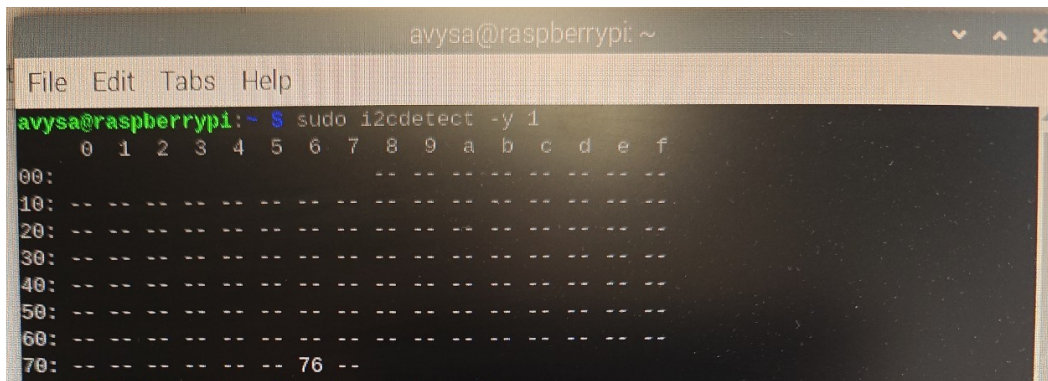
Connect the SCL pin to the SCL pin of the Raspberry Pi (GPIO 3).



Checking the Connection and Enabling I2C

After making the connections, the I2C interface on the Raspberry Pi needs to be enabled:

1. Open the terminal and enter the command «`sudo raspi-config`».
2. Go to "Interfacing Options".
3. Select the "I2C" option and enable it by selecting "Yes".
4. Restart the Raspberry Pi. To verify that the connections are made correctly, you can run the command «`sudo i2cdetect -y 1`» in the terminal. This command detects devices connected over the I2C interface and displays the address of our BME680 sensor (usually 76).



```
avysa@raspberrypi: ~  
File Edit Tabs Help  
avysa@raspberrypi:~$ sudo i2cdetect -y 1  
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
70: -- -- -- -- -- -- 76 --
```

Software and Libraries Installation

To read data from the BME680 using Python, we can use the library provided by Bosch. This library facilitates reading data from the sensor and obtaining various sensor values.

Python Libraries: Install the necessary Python libraries with the following commands in the terminal:

```
«sudo pip3 install RPi.GPIO»  
«sudo pip3 install bme680»
```

Enabling SSH on the Raspberry Pi

You can access and edit files and run codes on the Raspberry Pi from another computer on the same local network via SSH (Secure Shell). This is particularly useful when physical access to the Raspberry Pi is difficult. Here are the steps to do this:

Turn on the Raspberry Pi and log in to the desktop.

1. Open the terminal and enter «sudo raspi-config».
2. Select the "Interfacing Options" menu.
3. Find and enable the "SSH" option (select "Yes").
4. Select "Finish", exit, and restart the Raspberry Pi if necessary.

Finding the IP Address of the Raspberry Pi

To find the IP address of your Raspberry Pi, run the following command in the terminal:

```
«hostname -I»  
avysa@raspberrypi:~$ hostname -I  
192.168.14.24  
avysa@raspberrypi:~$
```

Creating a Virtual Environment and Installing Libraries

You can connect to your Raspberry Pi via SSH using another computer (Windows, Linux, or macOS).
Windows: You can use "Cmd" or "PowerShell" to SSH directly. Use the following command:

```
«ssh user_name@RASPBERRY_PI_IP_ADDRESS»
```

```
C:\Windows\System32>ssh avysa@192.168.14.24
avysa@192.168.14.24's password:
Linux raspberrypi 6.6.74+rpt-rpi-v7 #1 SMP Raspbian 1:6.6.74-1+rpt1 (2025-01-27) armv7l
```

Direct pip package installations in the system-wide Python environment can lead to errors. Instead, creating a virtual environment and specifically installing the bme680 library in it would be the best method. This way, you avoid conflicts with other packages in the system-wide environment. Below, I show you step-by-step how to create a Python virtual environment and install the bme680 package in it.

Continue Using the Existing Terminal Window on Your Raspberry Pi:

Keep the terminal open and follow the steps to create a virtual environment here.

Create and Activate a Virtual Environment:

Enter the following command in the terminal to create a virtual environment:

«python3 -m venv bme680env»

To activate the virtual environment you created:

«source bme680env/bin/activate»

In the virtual environment, install bme680 and other necessary libraries:

«pip install bme680»

```
avysa@raspberrypi:~ $ sudo apt install python3-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-venv is already the newest version (3.11.2-1).
The following packages were automatically installed and are no longer required:
  libcamera0.3 libdrm-nouveau2 libwlroots12 lxplug-network
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
avysa@raspberrypi:~ $ python3 -m venv bme680env
avysa@raspberrypi:~ $ source bme680env/bin/activate
(bme680env) avysa@raspberrypi:~ $ pip install bme680
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: bme680 in ./bme680env/lib/python3.11/site-packages (2.0.0)
Requirement already satisfied: smbus2 in ./bme680env/lib/python3.11/site-packages (from bme680) (0.5.0)
(bme680env) avysa@raspberrypi:~ $ python sensor_read.py
```

After this stage, we are now ready to receive and display data on the device using Python.

Python Script Creation and Execution

1. Open the Terminal:

- Open a terminal window on your Raspberry Pi.

2. Use a Text Editor:

- You can create a new Python file using Nano, a simple text editor. For example, to create a file named "your_filename.py", use the following command:

«nano your_file_name.py»

This command opens an existing file or creates a new one if it does not exist. After entering the file, you can write your Python code, save your work with Ctrl+O, press Enter, and then exit the file and the Nano editor with Ctrl+X.

Running the Python Script

- To run the Python script you created, use the following command in the terminal:

«python3 sensor_read.py»

When you type this command in the terminal, you will be able to see whether the Python codes you edited with Nano are working on your screen.

```
avysa@raspberrypi:~ $ sudo apt install python3-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-venv is already the newest version (3.11.2-1).
The following packages were automatically installed and are no longer required:
  libcamera0.3 libdrm-nouveau2 libwlroots12 lxplug-network
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
avysa@raspberrypi:~ $ python3 -m venv bme680env
avysa@raspberrypi:~ $ source bme680env/bin/activate
(bme680env) avysa@raspberrypi:~ $ pip install bme680
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: bme680 in ./bme680env/lib/python3.11/site-packages (2.0.0)
Requirement already satisfied: smbus2 in ./bme680env/lib/python3.11/site-packages (from bme680) (0.5.0)
(bme680env) avysa@raspberrypi:~ $ python sensor_read.py
^C
(bme680env) avysa@raspberrypi:~ $ ls
bme680env Bookshelf Desktop Documents Downloads Music Pictures Public sensor_read.py Templates Videos
(bme680env) avysa@raspberrypi:~ $ python3 sensor_read.py
Reading data from the BME680 sensor...
Temperature: 22.60 C, Humidity: 37.85 %RH, Pressure: 944.12 hPa, Air Quality: 9852941.73 ohms
Temperature: 22.60 C, Humidity: 37.85 %RH, Pressure: 944.12 hPa, Air Quality: 9843894.03 ohms
Temperature: 22.60 C, Humidity: 37.85 %RH, Pressure: 944.12 hPa, Air Quality: 9852941.73 ohms
Temperature: 22.60 C, Humidity: 37.85 %RH, Pressure: 944.12 hPa, Air Quality: 9852941.73 ohms
Temperature: 22.60 C, Humidity: 37.84 %RH, Pressure: 944.12 hPa, Air Quality: 9862006.07 ohms
Temperature: 22.60 C, Humidity: 37.84 %RH, Pressure: 944.12 hPa, Air Quality: 9852941.73 ohms
Temperature: 22.60 C, Humidity: 37.85 %RH, Pressure: 944.11 hPa, Air Quality: 9862006.07 ohms
Temperature: 22.60 C, Humidity: 37.85 %RH, Pressure: 944.11 hPa, Air Quality: 9843894.03 ohms
```

Example Code and Further Customization

```
(bme680env) avysa@raspberrypi:~ $ python3 sensor_read.py
[13:53] Temperature: 23.82 C, Humidity: 37.95 %RH, Pressure: 944.67 hPa, Air Quality: 5870756.31 ohms
[13:54] Temperature: 23.83 C, Humidity: 37.99 %RH, Pressure: 944.77 hPa, Air Quality: 5851528.95 ohms
[13:55] Temperature: 23.81 C, Humidity: 38.21 %RH, Pressure: 944.75 hPa, Air Quality: 5845147.78 ohms
[13:56] Temperature: 23.82 C, Humidity: 38.26 %RH, Pressure: 944.74 hPa, Air Quality: 5826087.52 ohms
```

Just as it is possible with Nano, you can also comfortably run codes from a computer connected via SSH using Flask with VSCode. At this stage, I am sharing my own Python code and HTML code, which display data and data timestamps once a minute.

It is possible to do many things by making additions to it.

If you have any questions, do not hesitate to contact me.

Thank you...