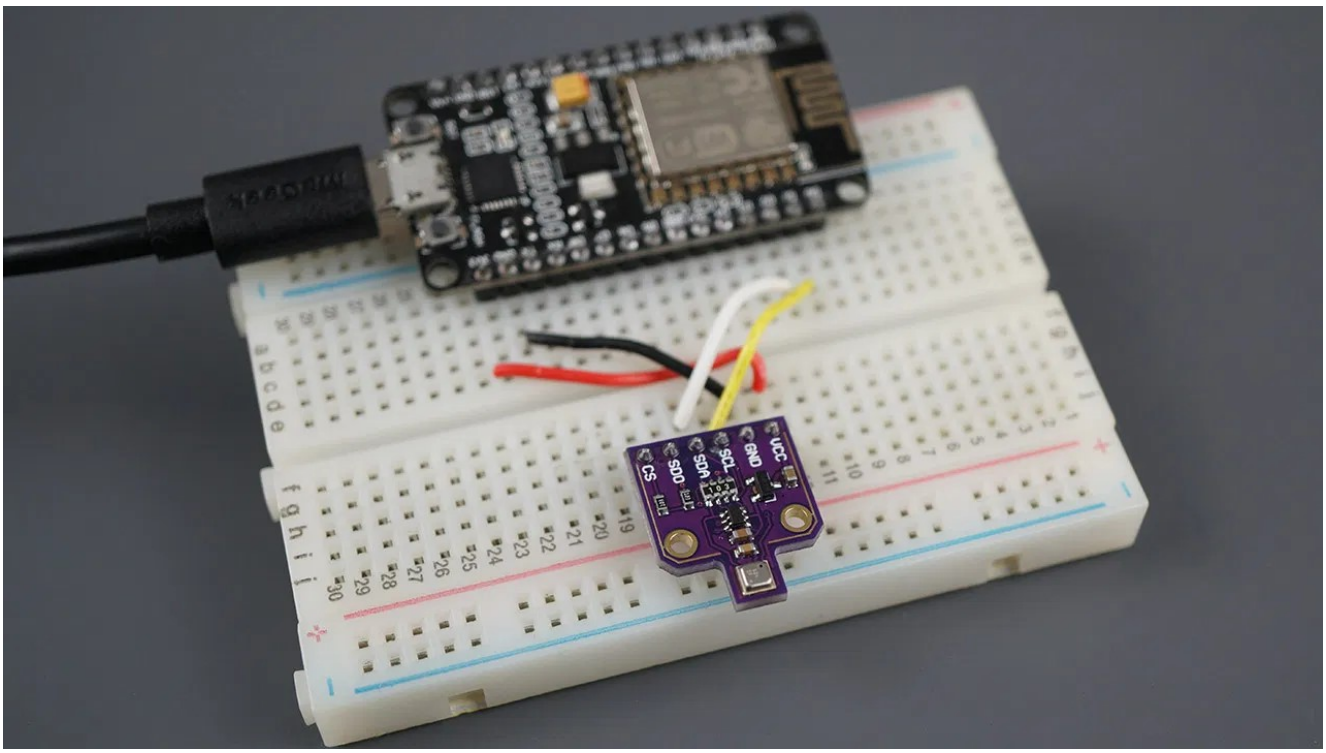
ESP8266 NodeMCU: BME680 Environmental Sensor using Arduino IDE (Gas, Pressure, Humidity, Temperature)

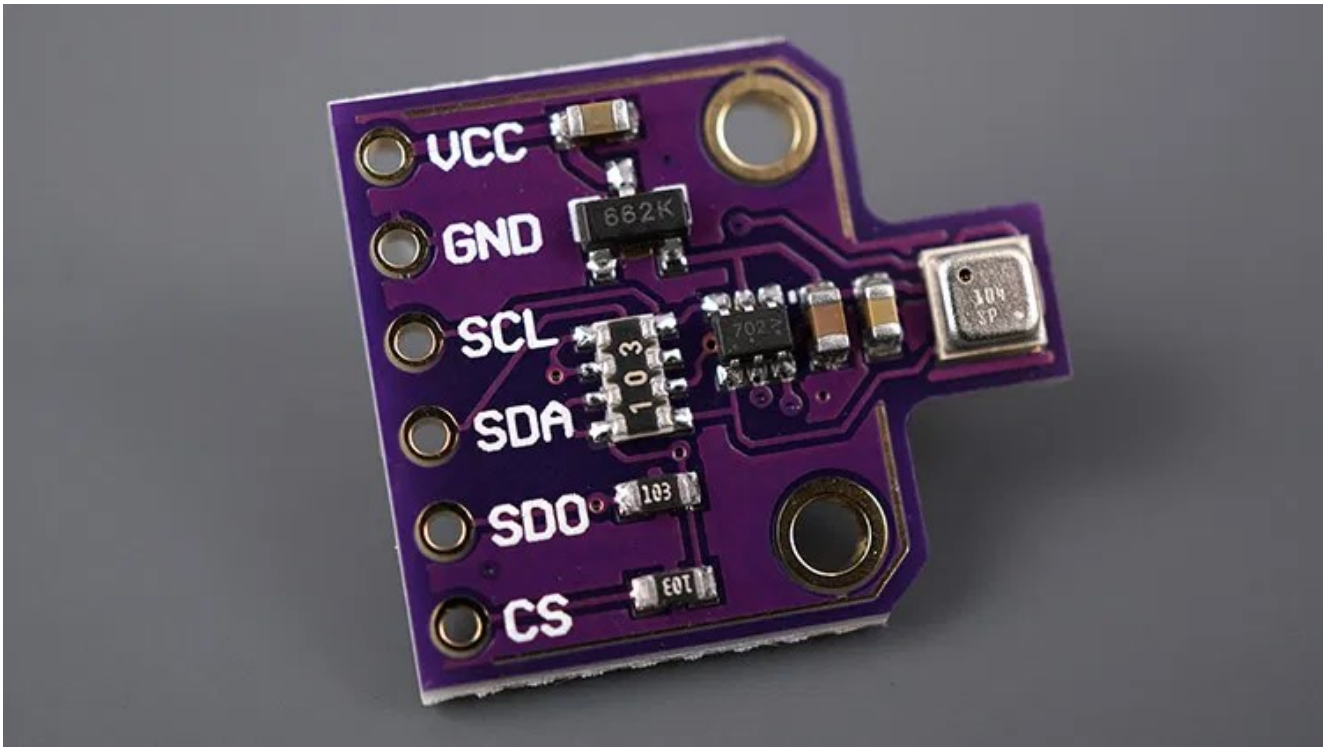
The BME680 is an environmental digital sensor that measures gas, pressure, humidity and temperature. In this guide you'll learn how to use the BME680 sensor module with the ESP8266 NodeMCU board using Arduino IDE. The sensor communicates with a microcontroller using I2C or SPI communication protocols.



You'll learn how to wire the sensor to the ESP8266 NodeMCU board, install the required libraries, use a simple sketch to display the sensor readings in the Serial Monitor and build a web server to monitor your sensor remotely.

Introducing BME680 Environmental Sensor Module

of gases like volatile organic compounds (VOC). For this reason, the BME680 can be used in indoor air quality control.



BME680 Measurements

The BME680 is a 4-in-1 digital sensor that measures:

- Temperature
- Humidity
- Barometric pressure
- Gas: Volatile Organic Compounds (VOC) like ethanol and carbon monoxide

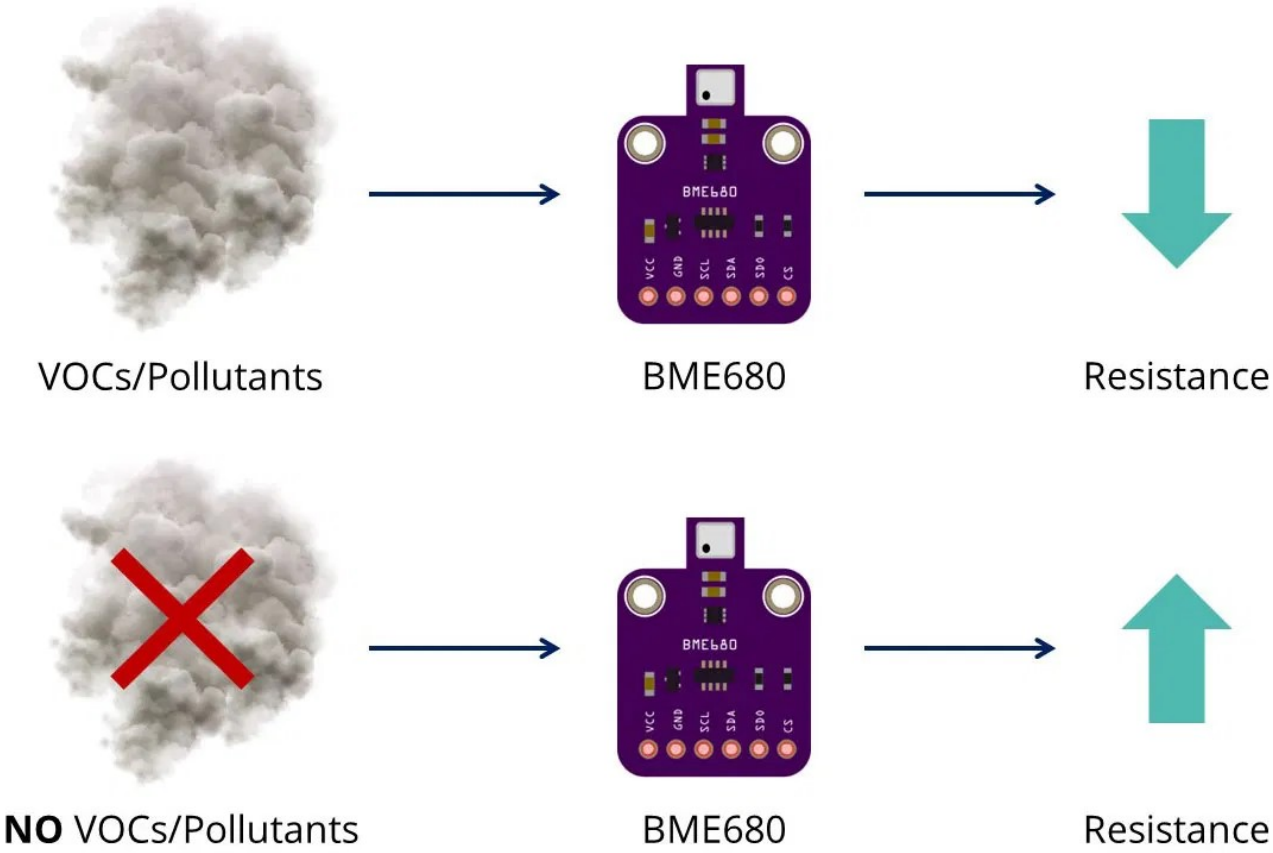
Gas Sensor

The BME680 contains a MOX (Metal-oxide) sensor that detects VOCs in the air. This sensor gives you a qualitative idea of the **sum of VOCs/contaminants** in the surrounding air – **it is not specific** for a specific gas molecule.

MOX sensors are composed of a metal-oxide surface, a sensing chip to

polluting indoor air (except CO2).

When the sensor comes into contact with the reducing gases, the oxygen molecules react and increase the conductivity across the surface. As a raw signal, the BME680 outputs resistance values. These values change due to variations in VOC concentrations:



- **Higher** concentration of VOCs » **Lower** resistance
- **Lower** concentration of VOCs » **Higher** resistance

The reactions that occur on the sensor surface (thus, the resistance) are influenced by parameters other than VOC concentration like temperature and humidity.

Relevant Information Regarding Gas Sensor

The gas sensor gives you a qualitative idea of VOCs gasses in the surrounding air. So, you can get trends, compare your results and see if the air quality is

When you first get the sensor, it is recommended to run it for 48 hours after start collecting “real” data. After that, it is also recommend to run the sensor for 30 minutes before getting a gas reading.

BME680 Accuracy

Here’s the accuracy of the temperature, humidity and pressure sensors of the BME680:

Sensor	**Accuracy**
Temperature	+/- 1.0ºC
Humidity	+/- 3%
Pressure	+/- 1 hPa

BME680 Operation Range

The following table shows the operation range for the temperature, humidity and pressure sensors for the BME680.

Sensor	**Operation** Range
Temperature	-40 to 85 ºC
Humidity	0 to 100 %
Pressure	300 to 1100 hPa

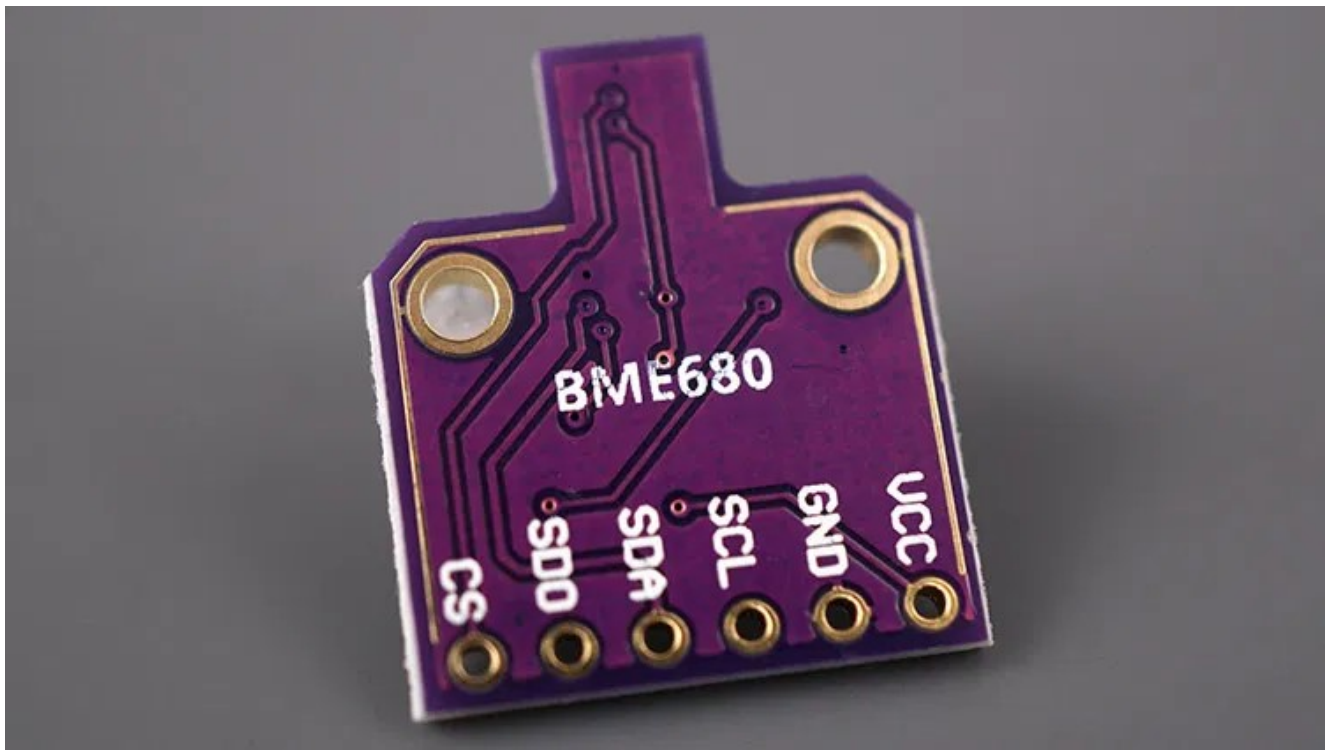
BME680 Pinout

Here’s the BME680 Pinout:

GND	Common GND
SCL	SCL pin for I2C communication SCK pin for SPI communication
SDA	SDA pin for I2C communication SDI (MISO) pin for SPI communication
SDO	SDO (MOSI) pin for SPI communication
CS	Chip select pin for SPI communication

BME680 Interface

The BME680 supports I2C and SPI Interfaces.



BME680 I2C

To use I2C communication protocol, use the following pins:

SCL	GPIO 5 (D1)
SDA	GPIO 4 (D2)

GPIO 5 (SCL) and GPIO 4 (SDA) are the default ESP8266 I2C pins. You can use other pins as long as you set them properly on code.

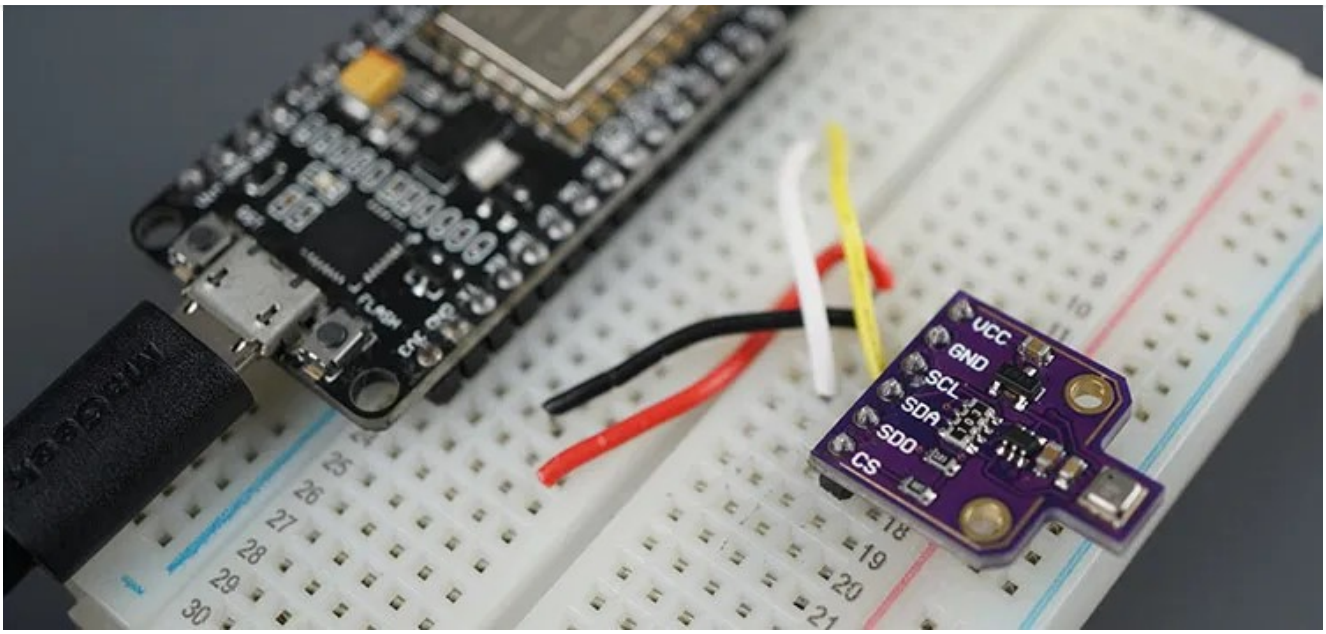
BME680 SPI

To use SPI communication protocol, use the following pins:

BME680	**ESP8266**
SCL (SCK SPI Clock)	GPIO 14 (D5)
SDA (SDI MOSI)	GPIO 13 (D7)
SDO (MISO)	GPIO 12 (D6)
CS (Chip Select)	GPIO 15 (D8)

These are the default ESP8266 SPI pins. You can use other pins as long as you set them properly in the code.

Parts Required



To complete this tutorial you need the following parts:

- BME680 sensor module
- ESP8266 (read Best ESP8266 development boards)
- Breadboard
- Jumper wires

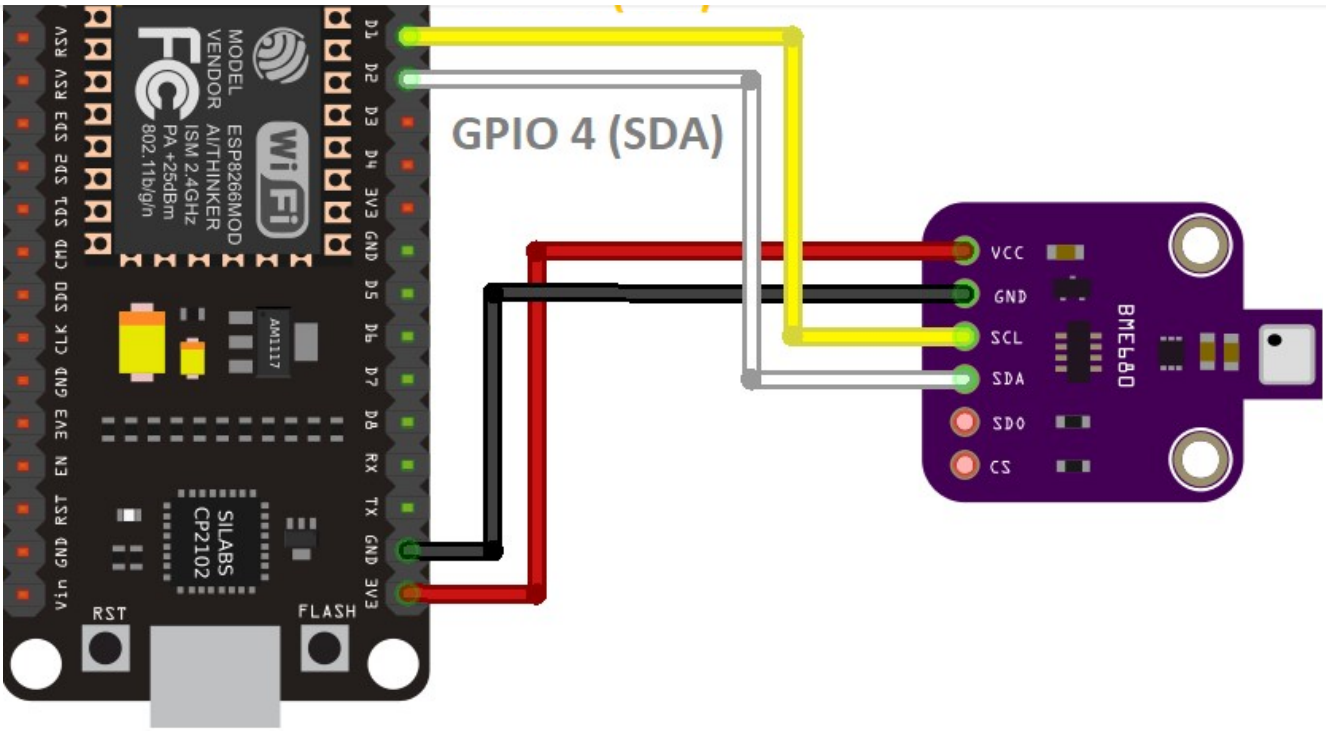
You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Schematic – ESP8266 NodeMCU with BME680

The BME680 can communicate using I2C or SPI communication protocols.

ESP8266 with BME680 using I2C

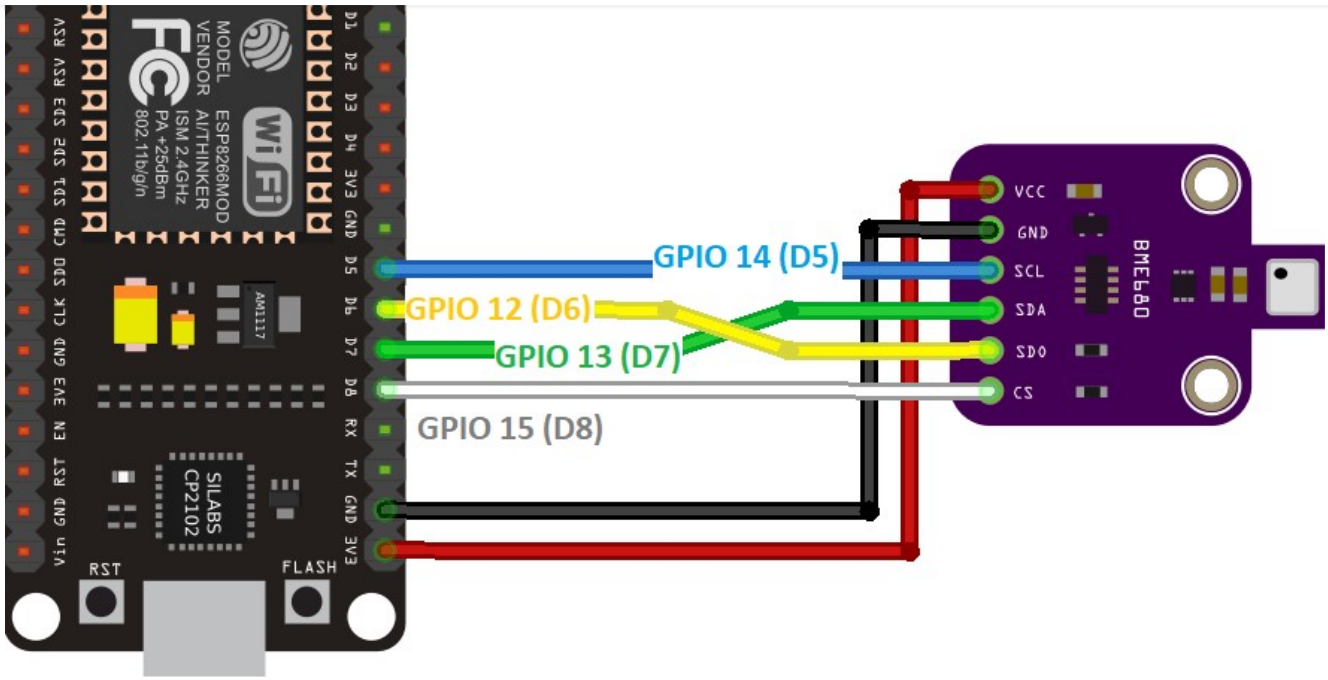
Follow the next schematic diagram to wire the BME680 to the ESP8266 using the default I2C pins.



ESP8266 with BME680 using SPI

Alternatively, you may want to use SPI communication protocol instead. In that case, follow the next schematic diagram to wire the BME680 to the ESP8266 using the default SPI pins.

Note: if you use SPI communication protocol, wire the sensor only after uploading the code.



Recommended reading: ESP8266 Pinout Reference: Which GPIO pins should you use?

Preparing Arduino IDE

We'll program the ESP8266 board using Arduino IDE. So, make sure you have the ESP8266 add-on installed. Follow the next tutorial:

- Install the ESP8266 Board in Arduino IDE

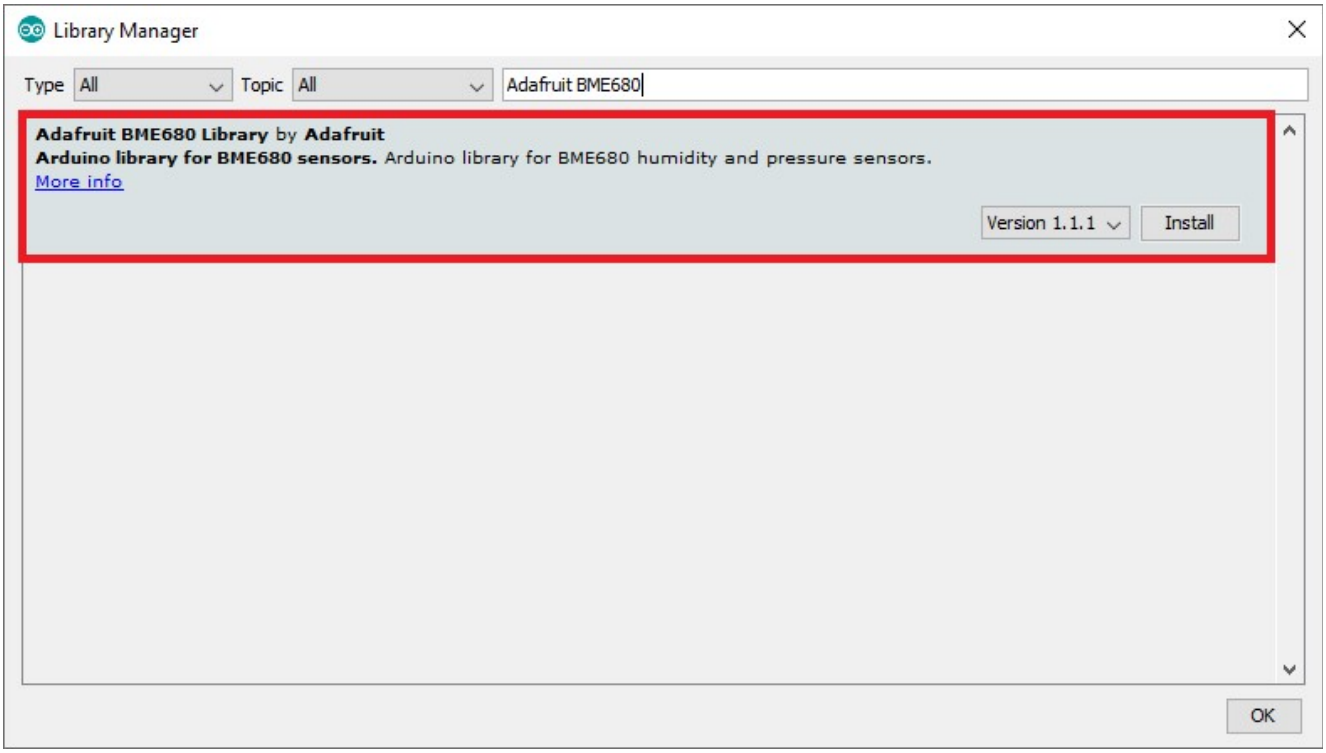
You also need to install the Adafruit BME680 library and the Adafruit Unified Sensor library.

Installing the BME680 Library

To get readings from the BME680 sensor module we'll use the Adafruit_BME680 library. Follow the next steps to install the library in your Arduino IDE:

Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage**

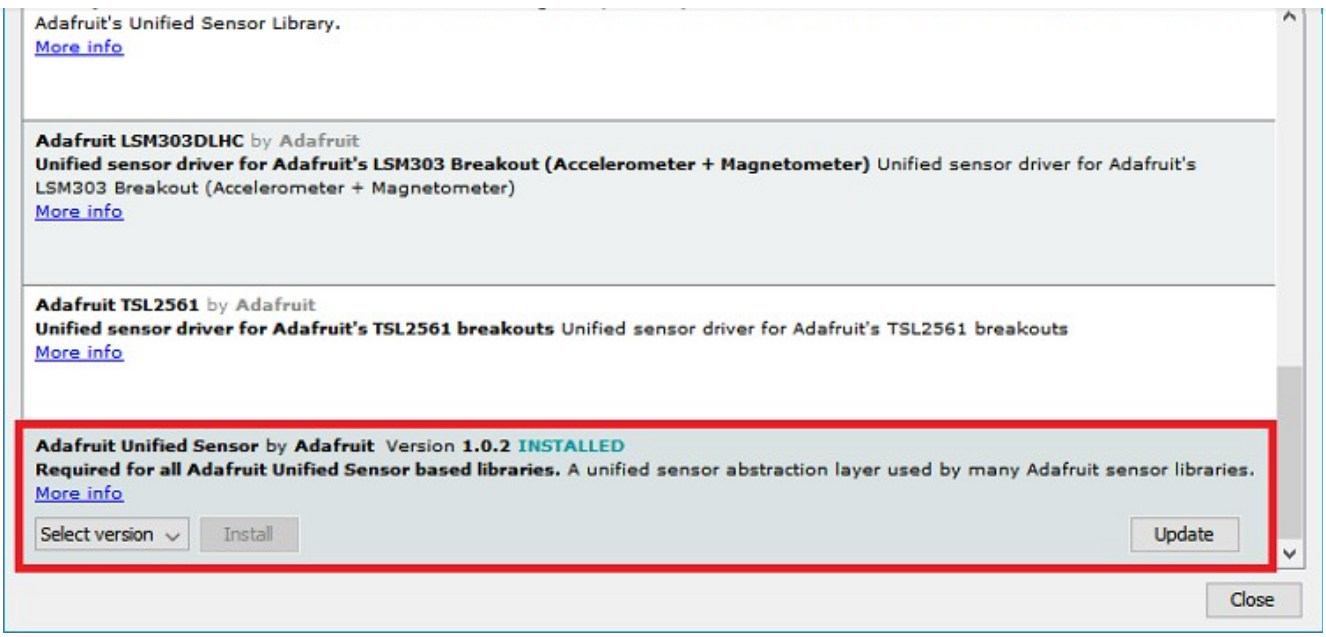
Search for "**adafruit bme680** " on the Search box and install the library.



Installing the Adafruit_Sensor Library

To use the BME680 library, you also need to install the Adafruit_Sensor library. Follow the next steps to install the library in your Arduino IDE:

Go to **Sketch** > **Include Library** > **Manage Libraries** and type "**Adafruit Unified Sensor**" in the search box. Scroll all the way down to find the library and install it.



After installing the libraries, restart your Arduino IDE.

Code – Reading BME680 Gas, Pressure, Humidity and Temperature

To read gas, pressure, temperature, and humidity we'll use a sketch example from the library.

After installing the BME680 library, and the Adafruit_Sensor library, open the Arduino IDE and, go to **File** > **Examples** > **Adafruit BME680 Library** > **bme680async**.

```
/***
  Read Our Complete Guide: https://RandomNerdTutorials.com/esp8266-nodemcu-bme
  Designed specifically to work with the Adafruit BME680 Breakout ----> http:/
***/

#include <Wire.h>
#include <SPI.h>
```

```
/*#define BME_SCK 14
#define BME_MISO 12
#define BME_MOSI 13
#define BME_CS 15*/

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME680 bme; // I2C
//Adafruit_BME680 bme(BME_CS); // hardware SPI
//Adafruit_BME680 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);

void setup() {
  Serial.begin(115200);
  while (!Serial);
  Serial.println(F("BME680 async test"));
```

View raw code

We've made a few changes to the sketch to make it fully compatible with the ESP8266.

How the Code Works

Continue reading this section to learn how the code works, or skip to the Demonstration section.

Libraries

The code starts by including the needed libraries: the `wire` library to use I2C, the `SPI` library (if you want to use SPI instead of I2C), the `Adafruit_Sensor` and `Adafruit_BME680` libraries to interface with the BME680 sensor.

```
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
```

SPI communication

We prefer to use I2C communication protocol with the sensor. However, the code is prepared if you want to use SPI. You just need to uncomment the following lines of code that define the SPI pins.

```
/*#define BME_SCK 14
#define BME_MISO 12
#define BME_MOSI 13
#define BME_CS 15*/
```

Sea level pressure

A variable called `SEALEVELPRESSURE_HPA` is created.

```
#define SEALEVELPRESSURE_HPA (1013.25)
```

This variable saves the pressure at the sea level in hectopascal (is equivalent to milibar). This variable is used to estimate the altitude for a given pressure by comparing it with the sea level pressure. This example uses the default value, but for accurate results, replace the value with the current sea level pressure at your location.

I2C

This example uses I2C communication protocol by default. The following line creates an `Adafruit_BME680` object called `bme` on the default ESP8266 I2C pins: GPIO 5 (SCL), GPIO 4 (SDA).

To use SPI, you need to comment this previous line and uncomment the following line.

```
//Adafruit_BME680 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI
```

setup()

In the `setup()` start a serial communication.

```
Serial.begin(115200);
```

Init BME680 Sensor

Initialize the BME680 sensor:

```
if (!bme.begin()) {
  Serial.println(F("Could not find a valid BME680 sensor, check wiring!"));
  while (1);
}
```

Set up the following parameters (oversampling, filter and gas heater) for the sensor.

```
// Set up oversampling and filter initialization
bme.setTemperatureOversampling(BME680_OS_8X);
bme.setHumidityOversampling(BME680_OS_2X);
bme.setPressureOversampling(BME680_OS_4X);
bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
```

To increase the resolution of the raw sensor data, it supports oversampling. We'll use the default oversampling parameters, but you can change them.

- `setTemperatureOversampling()` : set temperature oversampling.
- `setHumidityOversampling()` : set humidity oversampling.
- `setPressureOversampling()` : set pressure oversampling.

These methods can accepts one of the following parameters:

- `BME680_OS_NONE` : turn off reading;
- `BME680_OS_1X`
- `BME680_OS_2X`
- `BME680_OS_4X`
- `BME680_OS_8X`
- `BME680_OS_16X`

The BME680 sensor integrates an internal IIR filter to reduce short-term changes in sensor output values caused by external disturbances. The `setIIRFilterSize()` method sets the IIR filter. It accepts the filter size as a parameter:

- `BME680_FILTER_SIZE_0` (no filtering)
- `BME680_FILTER_SIZE_1`
- `BME680_FILTER_SIZE_3`
- `BME680_FILTER_SIZE_7`
- `BME680_FILTER_SIZE_15`
- `BME680_FILTER_SIZE_31`
- `BME680_FILTER_SIZE_63`
- `BME680_FILTER_SIZE_127`

The gas sensor integrates a heater. Set the heater profile using the `setGasHeater()` method that accepts as arguments:

We'll use the default settings: 320 ºC for 150 ms.

loop()

In the `loop()`, we'll get measurements from the BME680 sensor.

First, tell the sensor to start an asynchronous reading with `bme.beginReading()`. This returns the time when the reading would be ready.

```
// Tell BME680 to begin measurement.
unsigned long endTime = bme.beginReading();
if (endTime == 0) {
  Serial.println(F("Failed to begin reading :("));
  return;
}
Serial.print(F("Reading started at "));
Serial.print(millis());
Serial.print(F(" and will finish at "));
Serial.println(endTime);
```

Then, call the `endReading()` method to end an asynchronous reading. If the asynchronous reading is still in progress, block until it ends.

```
if (!bme.endReading()) {
  Serial.println(F("Failed to complete reading :("));
  return;
}
```

After this, we can get the readings as follows:

- `bme.humidity` : returns humidity reading
- `bme.gas_resistance` : returns gas resistance

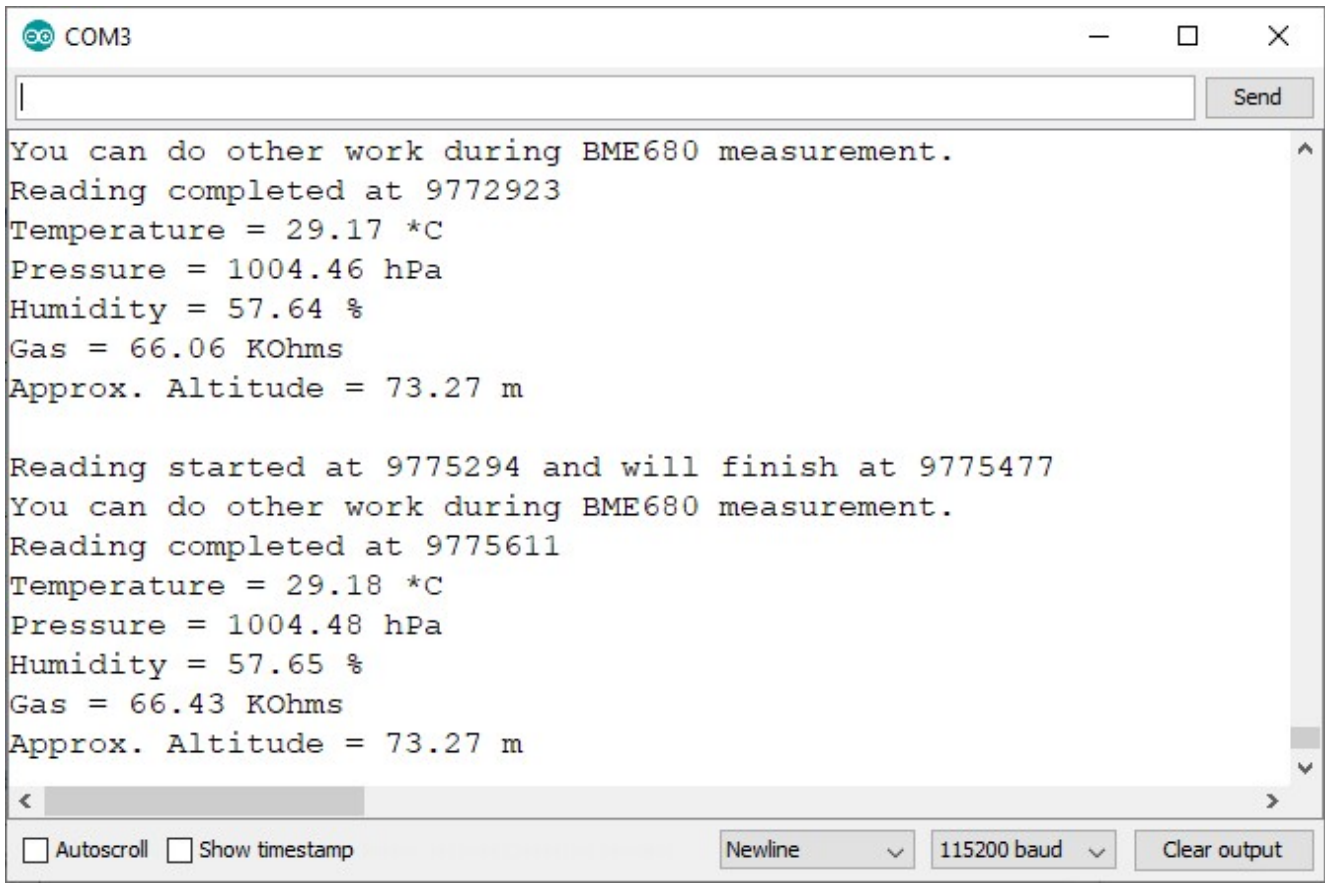
```
Serial.print(F("Temperature = "));
Serial.print(bme.temperature);
Serial.println(F(" *C"));

Serial.print(F("Pressure = "));
Serial.print(bme.pressure / 100.0);
Serial.println(F(" hPa"));

Serial.print(F("Humidity = "));
Serial.print(bme.humidity);
Serial.println(F(" %"));

Serial.print(F("Gas = "));
Serial.print(bme.gas_resistance / 1000.0);
Serial.println(F(" KOhms"));

Serial.print(F("Approx. Altitude = "));
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(F(" m"));
```

For more information about the library methods, take a look at the Adafruit_BME680 Class Reference.

Demonstration

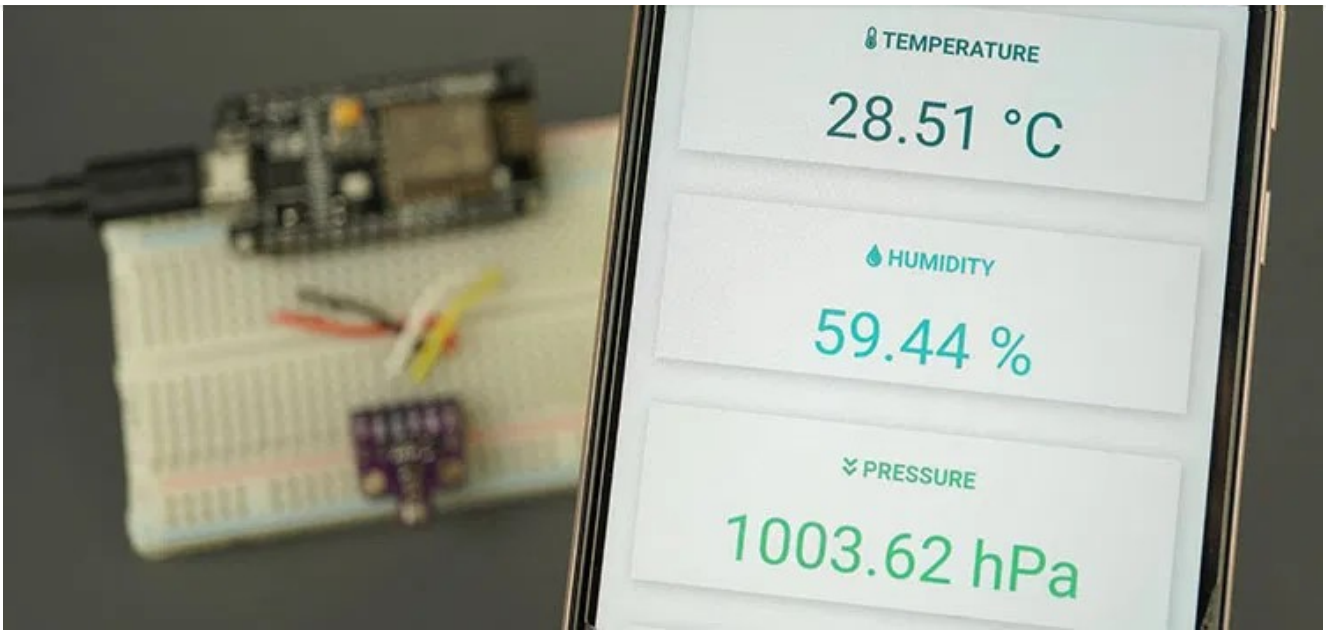
Upload the code to your ESP8266 board. Go to **Tools** > **Board** and select the ESP8266 board you're using. Go to **Tools** > **Port** and select the port your board is connected to. Then, click the upload button.

Note: if you'r using SPI communication, wire the circuit only after pressing the RST button.



Code – ESP8266 NodeMCU Web Server with BME680

In this section, we provide an example of web server that you can build with the ESP8266 to display BME680 readings.



Installing Libraries – Async Web Server

To build the web server you need to install the following libraries. Click the links below to download the libraries.

- ESPAsyncWebServer
- ESPAsyncTCP

These libraries aren't available to install through the Arduino Library Manager, so you need to copy the library files to the Arduino Installation Libraries folder. Alternatively, in your Arduino IDE, you can go to **Sketch** > **Include Library** > **Add .zip Library** and select the libraries you've just downloaded.

Code

Then, upload the following code to your board (type your SSID and password).

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp8266-nodemcu-
```

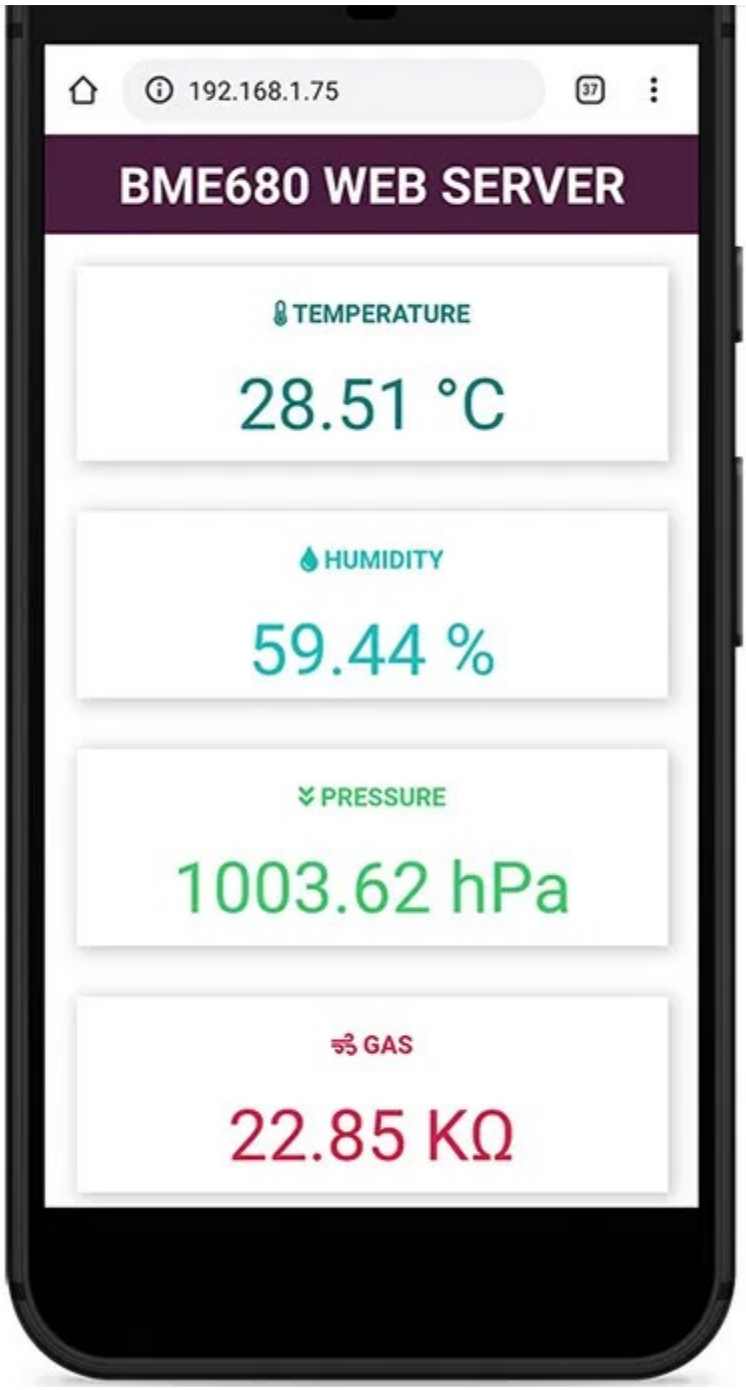
```
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in a
  copies or substantial portions of the Software.
*********/

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <ESP8266WiFi.h>
#include "ESPAsyncWebServer.h"

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

//Uncomment if using SPI
/*#define BME_SCK 14
#define BME_MISO 12
#define BME_MOSI 13
```

View raw code

Demonstration

After uploading, open the Serial Monitor at a baud rate of 115200 to get the ESP8266 IP address.

Open a browser and type the IP address. You should get access to the web server with the latest sensor readings. You can access the web server on your computer, tablet or smartphone in your local network.



The readings are updated automatically on the web server using Server-Sent Events.

We won't explain how the web server works in this tutorial. We wrote this guide dedicated to the BME680 web server with the ESP8266 board.

The BME680 sensor module is a 4-in-1 digital sensor that combines gas, pressure, temperature and humidity sensors. The BME680 contains a MOX sensor that senses the presence of most VOC gases. This sensor gives you a qualitative idea of the sum of VOCs/contaminants in the surrounding air. For this reason, the BME680 can be used to monitor indoor air quality.

If you're using an ESP32, read ESP32: BME680 Environmental Sensor using Arduino IDE (Gas, Pressure, Humidity, Temperature).

We hope you've found this getting started guide useful. We have guides for other popular sensors:

- ESP8266 **DHT11/DHT22 Temperature and Humidity** with Arduino IDE
- ESP8266 with **BME280** using Arduino IDE (**Pressure, Temperature, Humidity**)
- ESP8266 **DS18B20 Temperature Sensor** with Arduino IDE (Single, Multiple, Web Server)

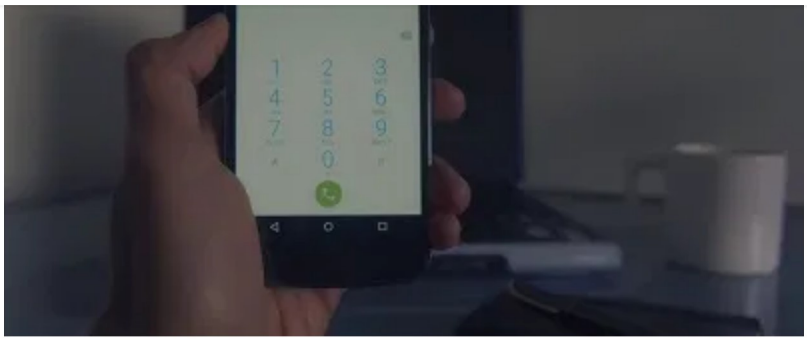
Learn more about the ESP8266 with our resources:

- **Home Automation using ESP8266 (eBook)**
- More ESP8266 NodeMCU Projects and Tutorials ...

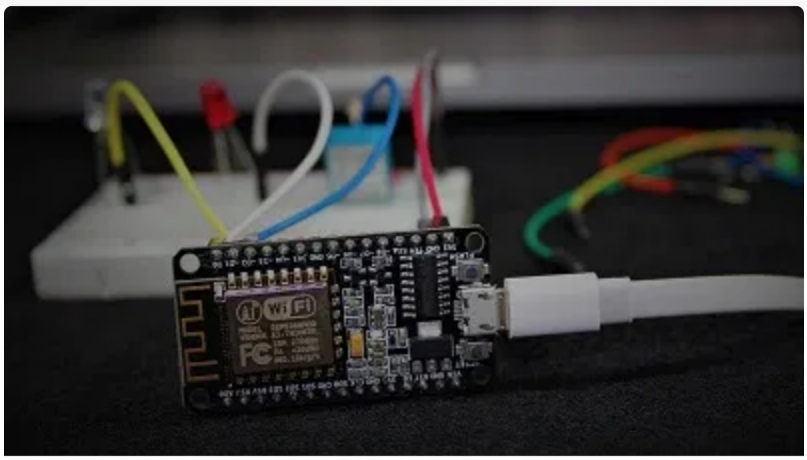
Thanks for reading.

Learn how to program and build projects with the ESP32 and ESP8266 using MicroPython firmware **DOWNLOAD »**

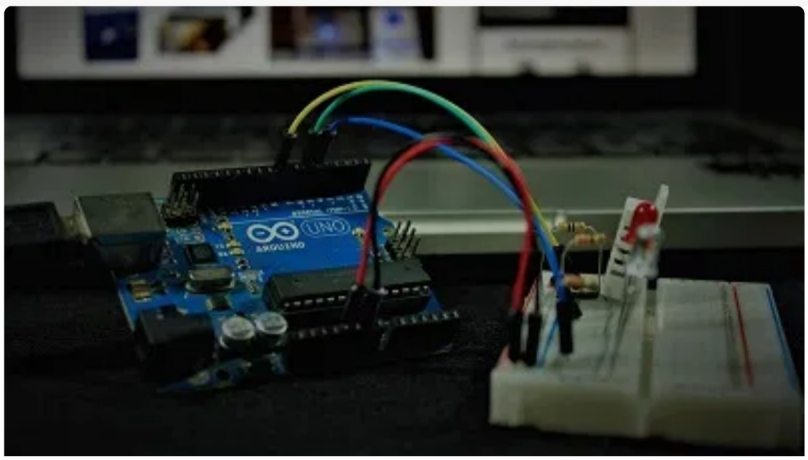
Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Arduino Step-by-Step Projects » Build 25 Arduino projects with our course, even with no prior experience!

What to Read Next...

ESP32 Web Server with BME680 – Weather Station (Arduino IDE)

ESP8266 NodeMCU Web Server with BME680 – Weather Station (Arduino IDE)

ESP8266 0.96 inch OLED Display with Arduino IDE

ESP32 Troubleshooting Guide

MicroPython: MQTT – Publish DS18B20 Temperature Readings (ESP32/ESP8266)

Enjoyed this project? Stay updated by subscribing our weekly newsletter!

Your Email Address

SUBSCRIBE

11 thoughts on “ESP8266 NodeMCU: BME680 Environmental Sensor using Arduino IDE (Gas,

WILLIAM E WEBB
September 7, 2020 at 12:58 am

How would I display on the web page, sensor values rounded to one decimal point (ie. 80,33 to 80.3)?

Reply

Moises Guergolet
September 17, 2020 at 4:35 pm

Hi.
I keep getting the message: “Could not find a valid BME680 sensor, check wiring”

The wiring seems to be OK.

How do I know if the sensor it self is running?

Reply

Sara Santos
September 19, 2020 at 11:34 am

Hi.
Please check the I2C address of your sensor.

Tutorials/master/Projects/LCD_I2C/I2C_Scanner.ino
Then, change the following line to include your I2C address.
if (!bme.begin(YOUR I2C ADDRESS)) {
I hope this helps.
Regards,
Sara

Reply

David
September 30, 2020 at 8:00 am

I had a delivery of 680's recently. Only half a dozen but one can't be found by my code or indeed by the i2c scanner. They're a bit pricey so it'll go back to @Bay for a replacement.
So, it does happen.

Reply

David
September 29, 2020 at 2:38 pm

Hi,
thank you for the useful tutorial.
What is a "safe" gas resistance reading?
Is there a conversion to ppm contaminants?
Many thanks

Reply

Pentti
October 5, 2020 at 8:01 am

Fine tutorial bus like David I would like to convert Gas resistance to PPM. Is there any code or formulas available?

Reply

WILLIAM E WEBB
October 5, 2020 at 3:39 pm

I found this helpful.

https://github.com/G6EJD/BME680-Example

Reply

Pentti
October 5, 2020 at 7:34 pm

Thanks a lot William! Seems to be usefull and I will study this.

WILLIAM E WEBB
October 5, 2020 at 7:36 pm

Using the code in the G6EJD link, my server looks like this:

http://orangecaweather.duckdns.org:8002/

Reply

Pentti
October 6, 2020 at 3:55 am

Looks fine! Thanks

OH7QT

Steve Spence
January 5, 2021 at 7:46 am

if (!bme.begin()) {
to
if (!bme.begin(BME680_I2C_ADDR_PRIMARY)) { //0x76

use BME680_I2C_ADDR_SECONDARY for alternate address 0x77

Reply

Leave a Comment

Name *

Email *

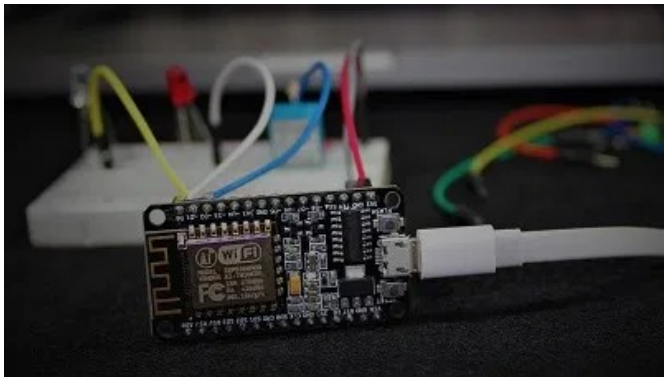
Website

□ Notify me of follow-up comments by email.

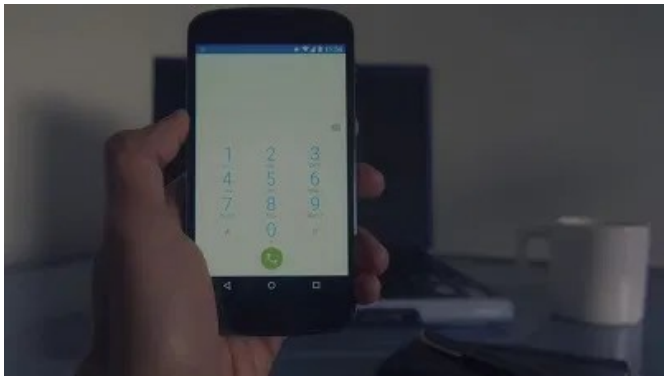
□ Notify me of new posts by email.

Post Comment

Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.