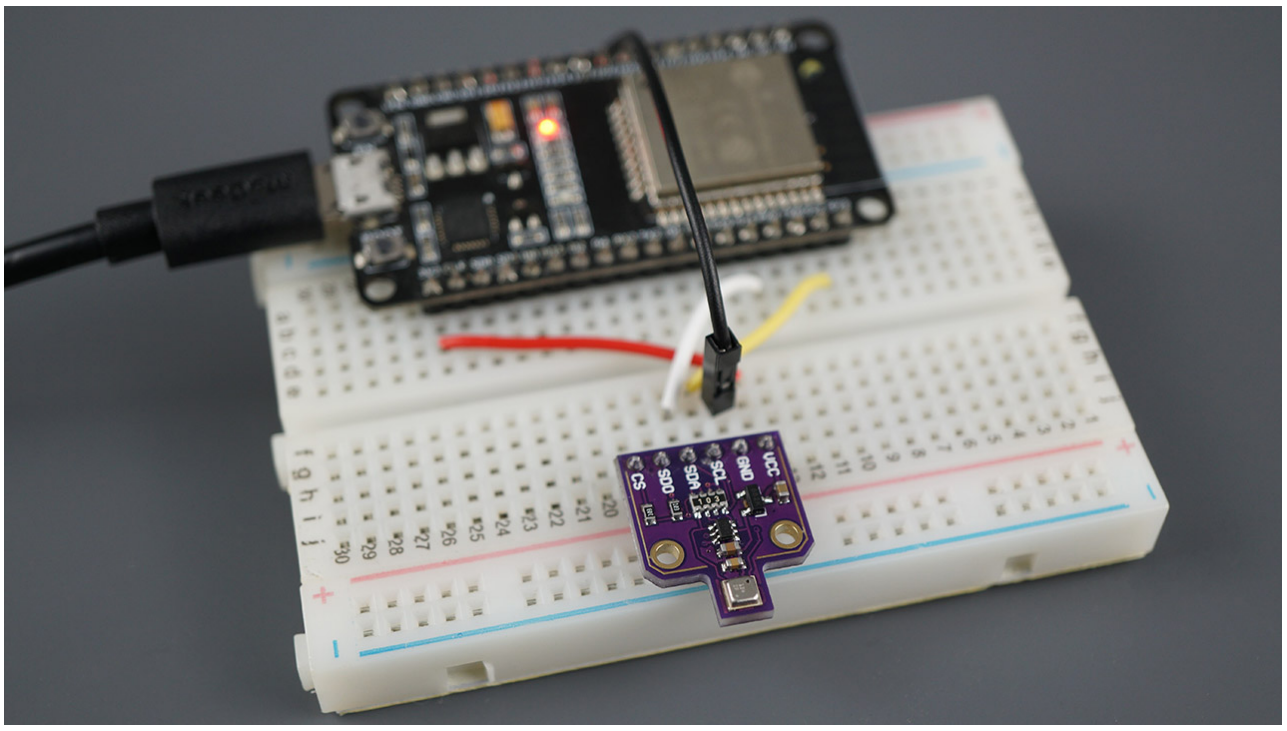
ESP32: BME680 Environmental Sensor using Arduino IDE (Gas, Pressure, Humidity, Temperature)

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 ESP32 board using Arduino IDE. The sensor communicates with a microcontroller using I2C or SPI communication protocols.

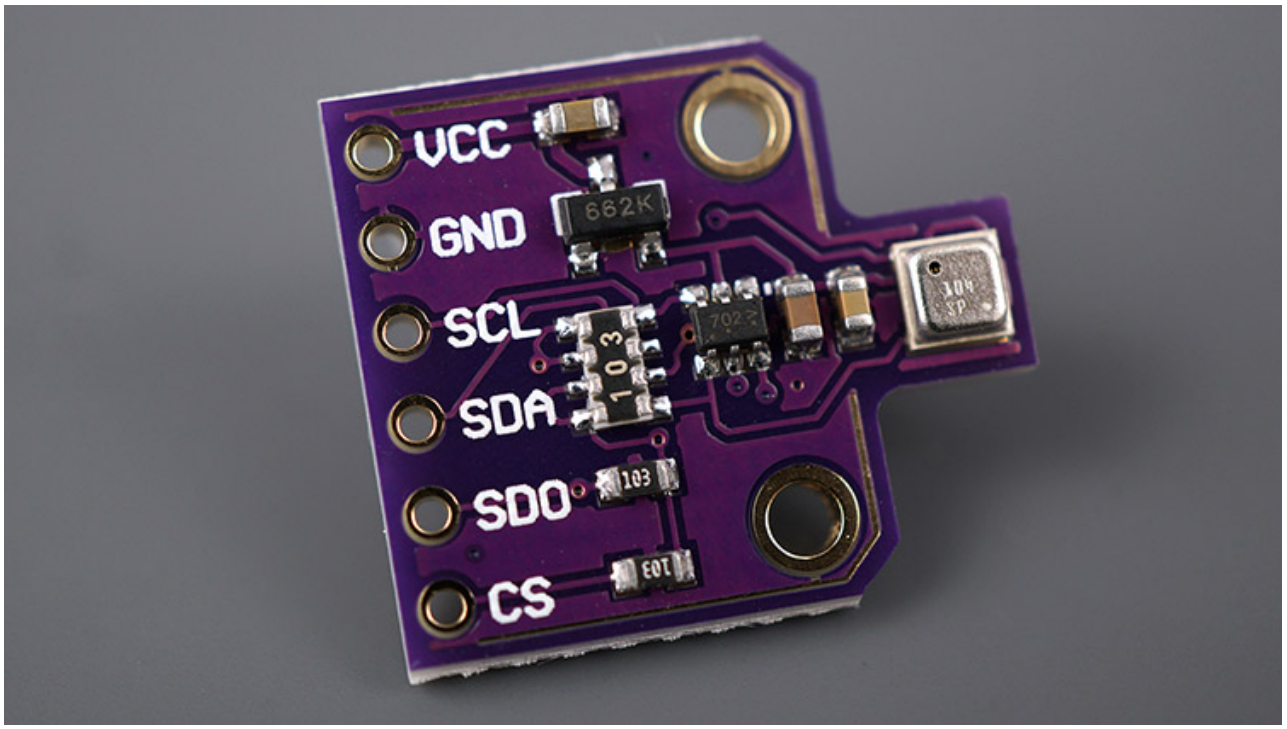


You'll learn how to wire the sensor to the ESP32 board, install the required libraries, use a simple sketch to display the sensor readings in the Serial Monitor and build a web server to monitor your sensor remotely.

Introducing BME680 Environmental Sensor Module

The BME680 is an environmental sensor that combines gas, pressure, humidity and temperature sensors. The gas sensor can detect a broad range 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

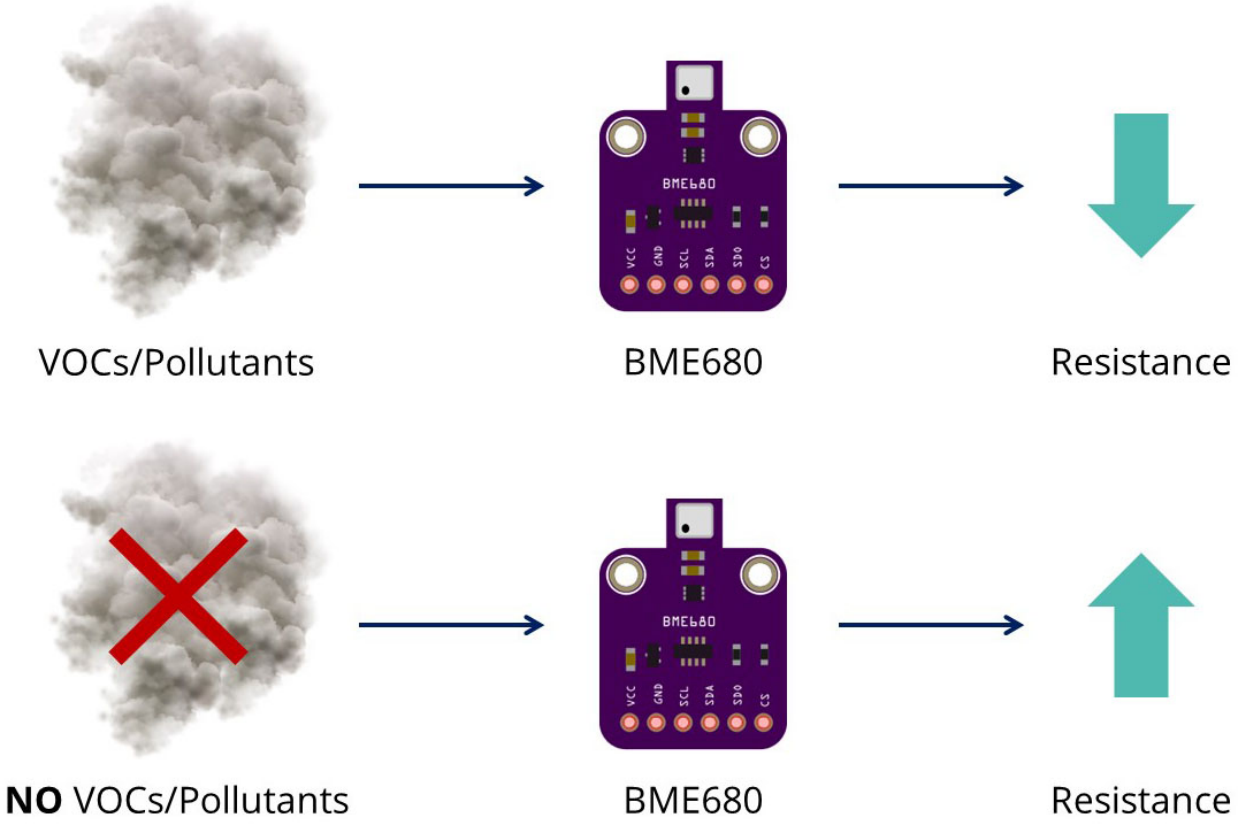
Gas Sensor

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 measure changes in conductivity, and a heater. It detects VOCs by adsorption of oxygen molecules on its sensitive layer. The BME680 reacts to most VOCs 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 increasing or decreasing. To get precise measurements, you need to calibrate the sensor against knows sources and build a calibration curve.

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

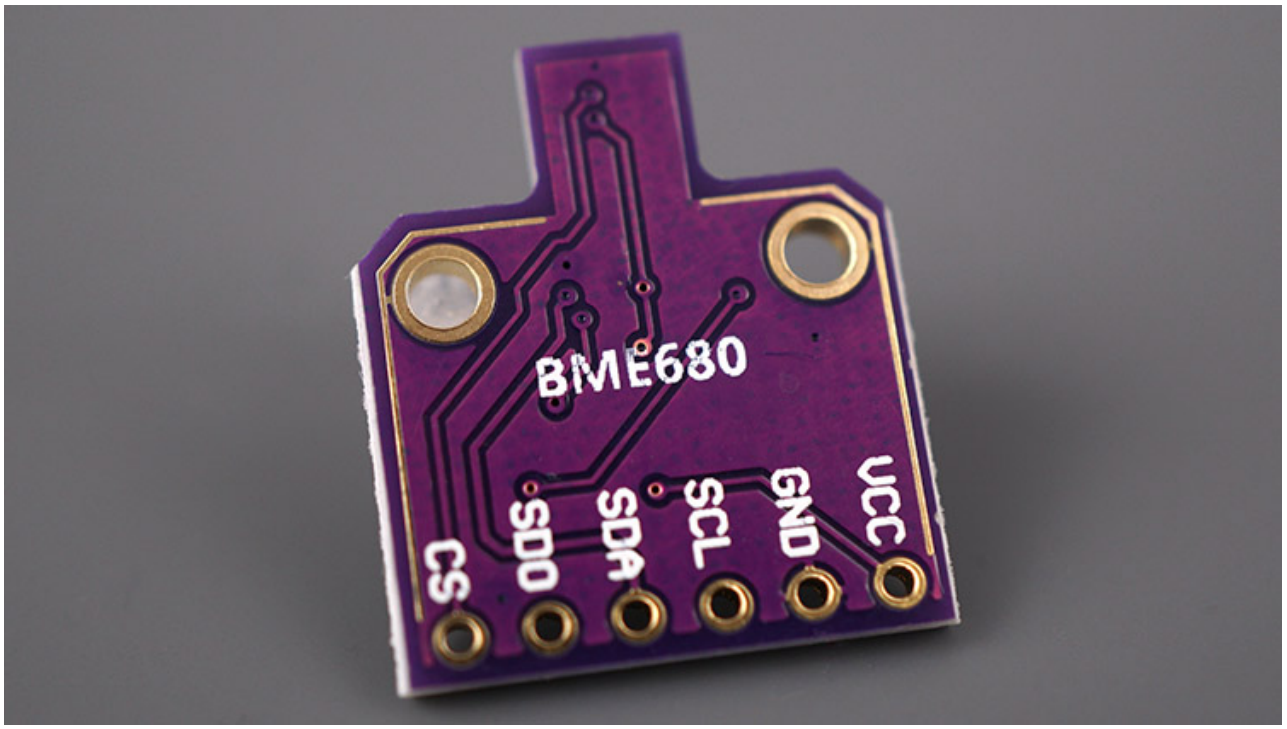
Here's the BME680 Pinout:

VCC	Powers the sensor
GND	Common GND
SCL	SCL pin for I2C communication SCK pin for SPI communication
SDA	SDA pin for I2C communication SDI (MOSI) pin for SPI communication

SDO	SDO (MISO) 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:

BME680	ESP32
SCL	GPIO22
SDA	GPIO 21

GPIO 22 (SCL) and GPIO 21 (SDA) are the default ESP32 I2C pins. You can use other pins as long as you set them properly on code.

Recommended reading: ESP32 I2C Communication: Set Pins, Multiple Bus

Interfaces and Peripherals (Arduino IDE)

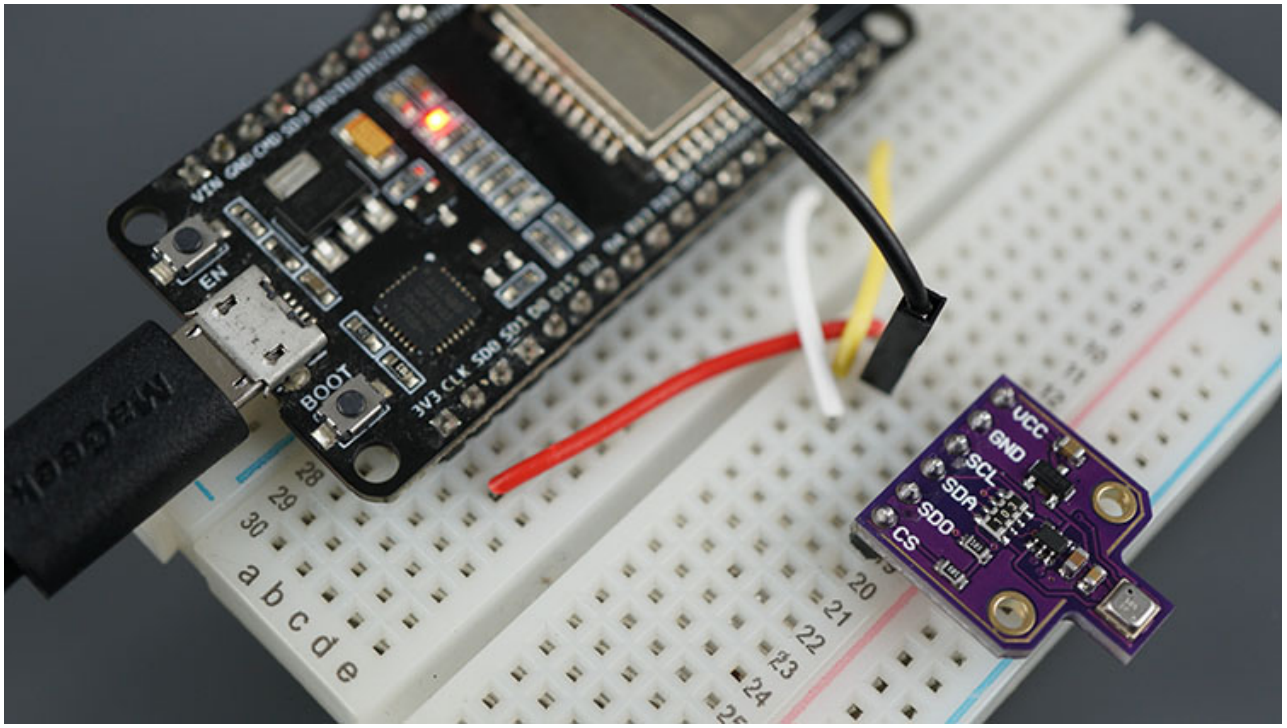
BME680 SPI

To use SPI communication protocol, use the following pins:

BME680	ESP32
SCL (SCK SPI Clock)	GPIO 18
SDA (SDI MOSI)	GPIO 23
SDO (MISO)	GPIO 19
CS (Chip Select)	GPIO 5

These are the default ESP32 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
- ESP32 (read Best ESP32 development boards)
- Breadboard
- Jumper wires

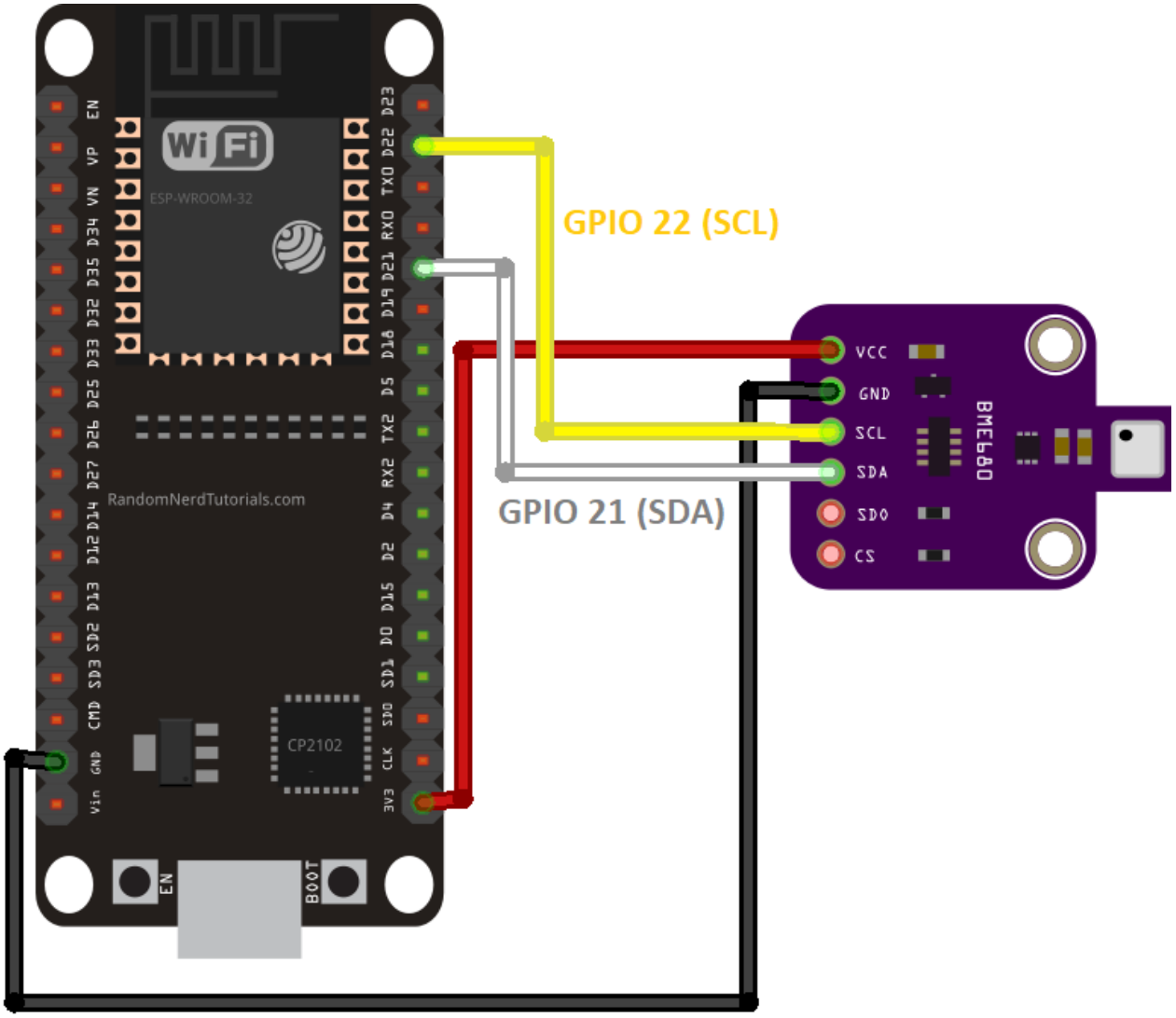
You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Schematic – ESP32 with BME680

The BME680 can communicate using I2C or SPI communication protocols.

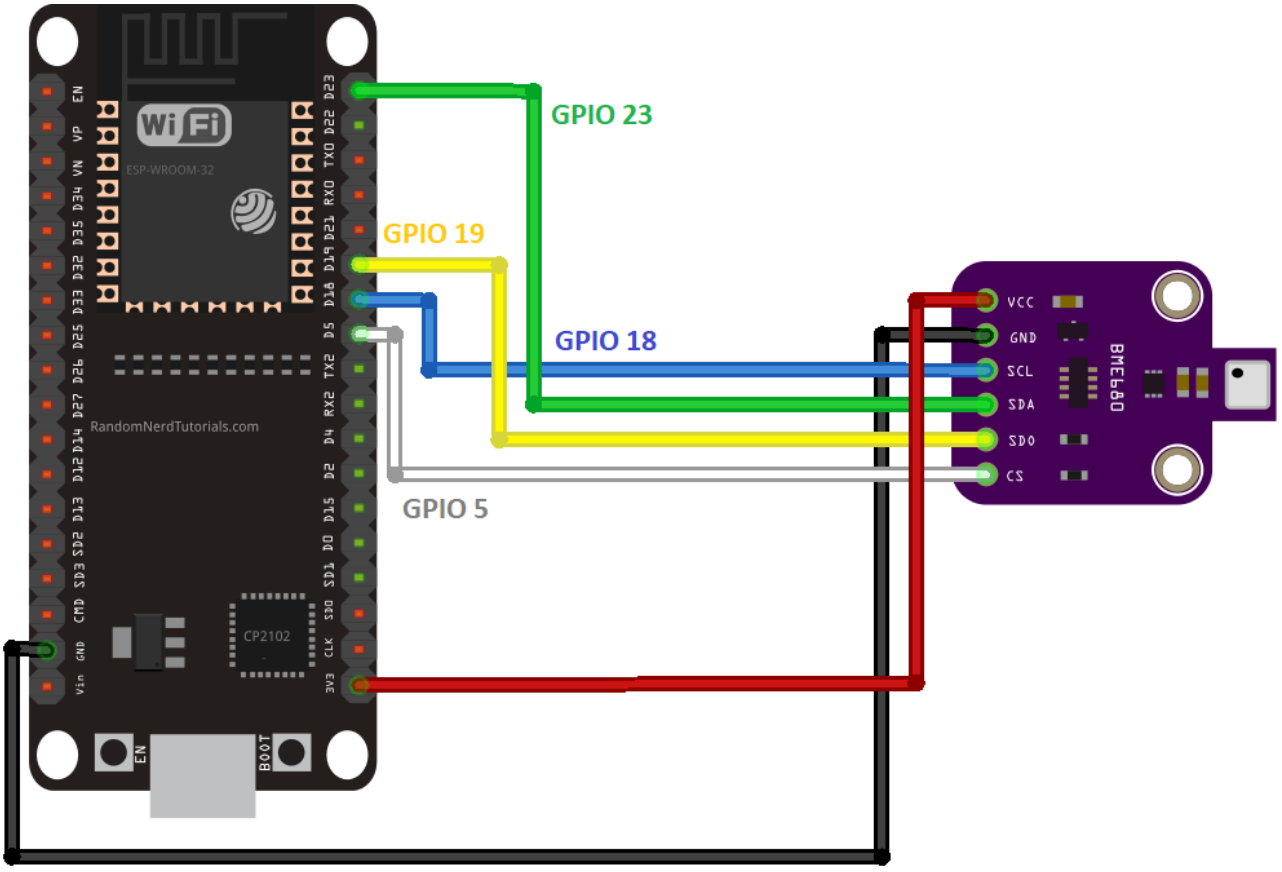
ESP32 with BME680 using I2C

Follow the next schematic diagram to wire the BME680 to the ESP32 using the default I2C pins.



ESP32 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 ESP32 using the default SPI pins.



Recommended reading: ESP32 Pinout Reference: Which GPIO pins should you use?

Preparing Arduino IDE

We'll program the ESP32 board using Arduino IDE. So, make sure you have the ESP32 add-on installed. Follow the next tutorial:

- Install the ESP32 Board in Arduino IDE

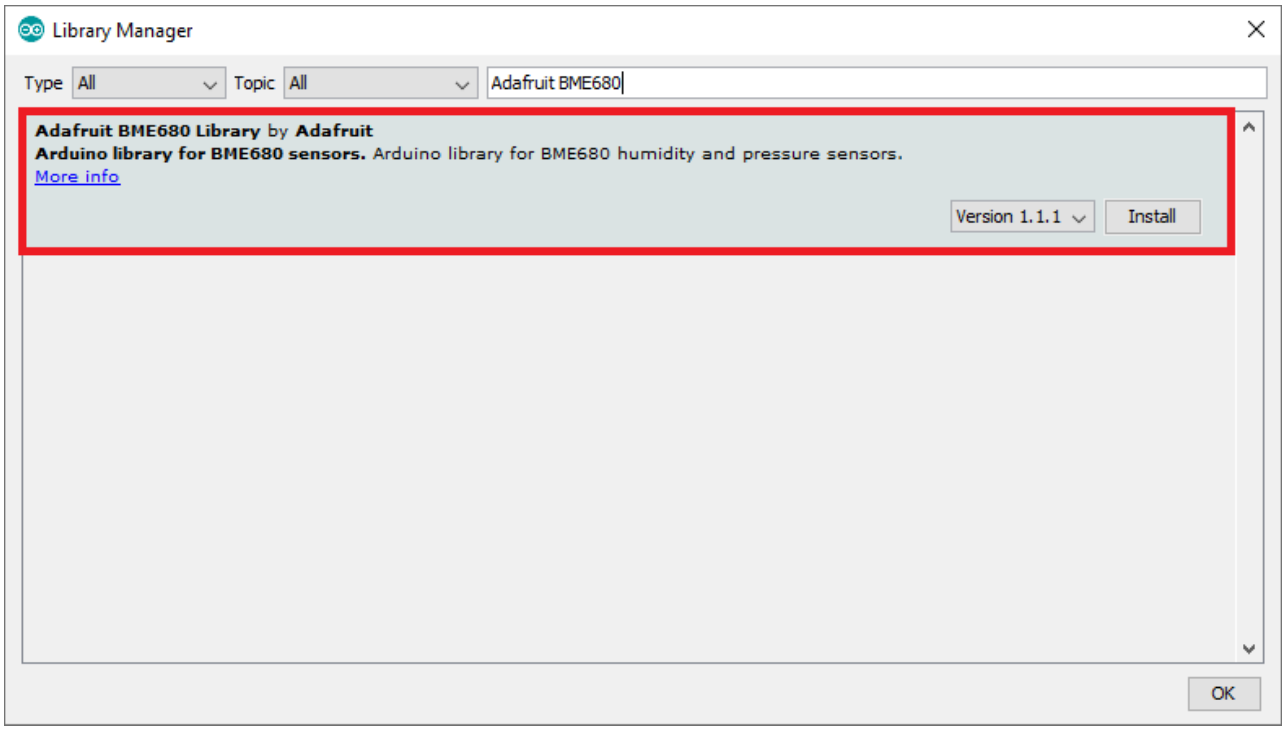
You also need to install the Adafruit BME680 library and the Adafruit Unified Sensor library.

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 Libraries**. The Library Manager should open.

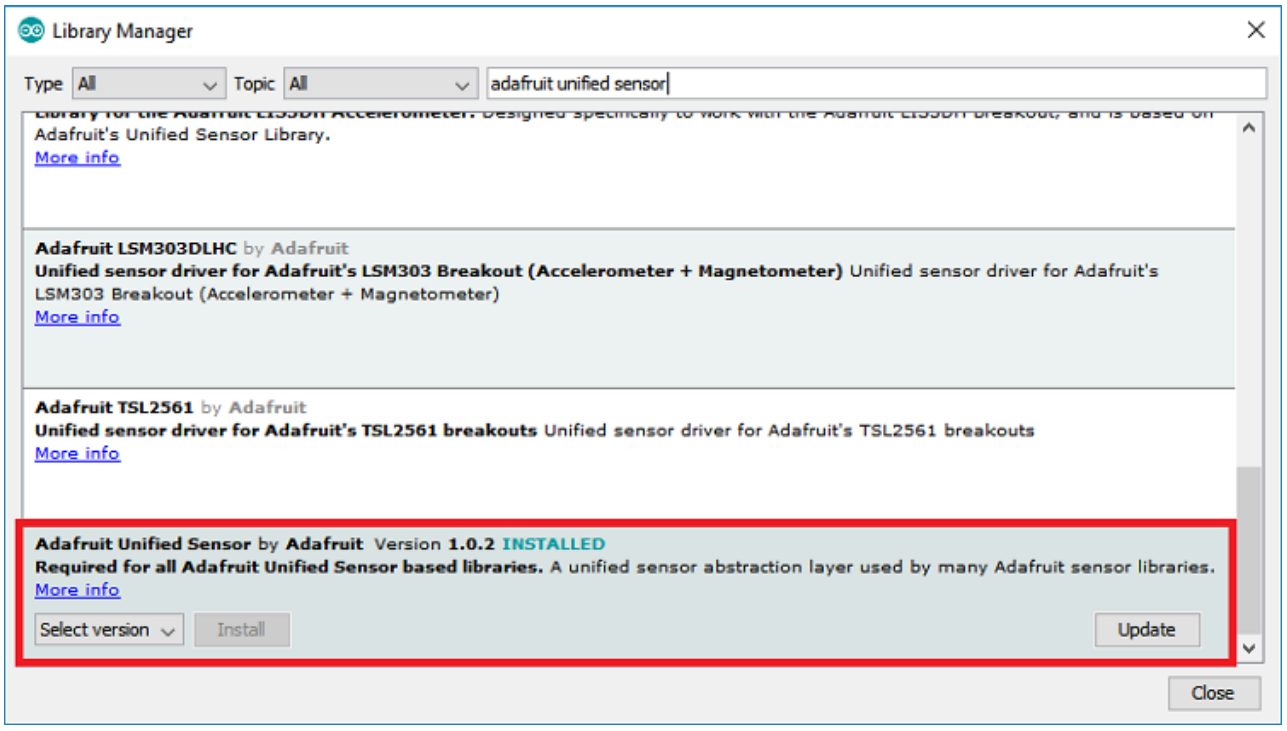
Search for “**adafruit bme680** ” on the Search box and install the library.



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/esp32-bme6
  Designed specifically to work with the Adafruit BME680 Breakout ---
***/

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

/*#define BME_SCK 18
#define BME_MISO 19
#define BME_MOSI 23
#define BME_CS 5*/

#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"));
```

```
  if (!bme.begin()) {
```

View raw code

We've made a few changes to the sketch to make it fully compatible with the ESP32.

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 <Wire.h>
#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 18
#define BME_MISO 19
#define BME_MOSI 23
#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 ESP32 I2C pins: GPIO 22 (SCL), GPIO 21 (SDA).

```
Adafruit_BME680 bme; // I2C
```

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); // softwa
```

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 wirin
  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);
bme.setGasHeater(320, 150); // 320*C for 150 ms
```

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:

- the heater temperature (in degrees Centigrade)
- the time the heater should be on (in milliseconds)

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.temperature` : returns temperature reading
- `bme.pressure` : returns pressure reading
- `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));
```

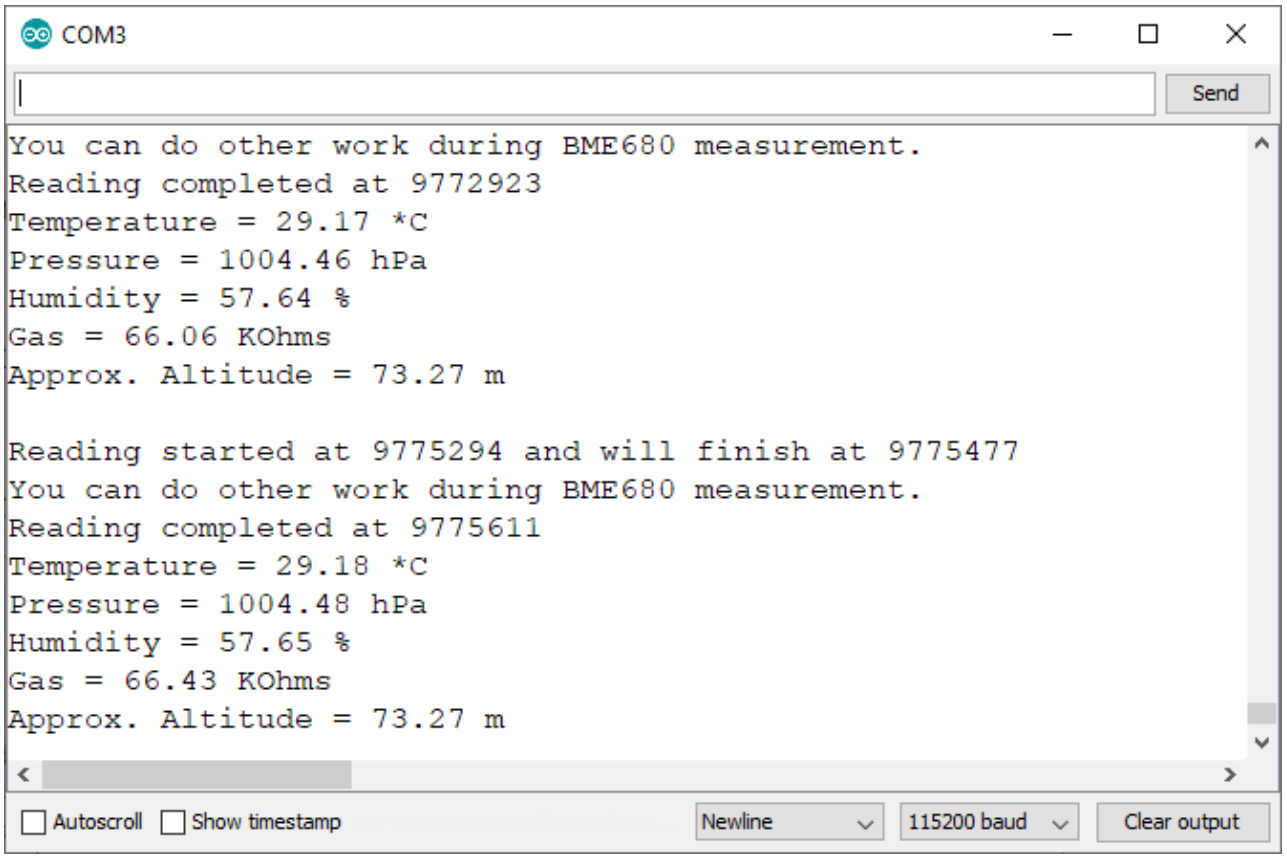
```
  Serial.println(F(" m"));
```

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

Demonstration

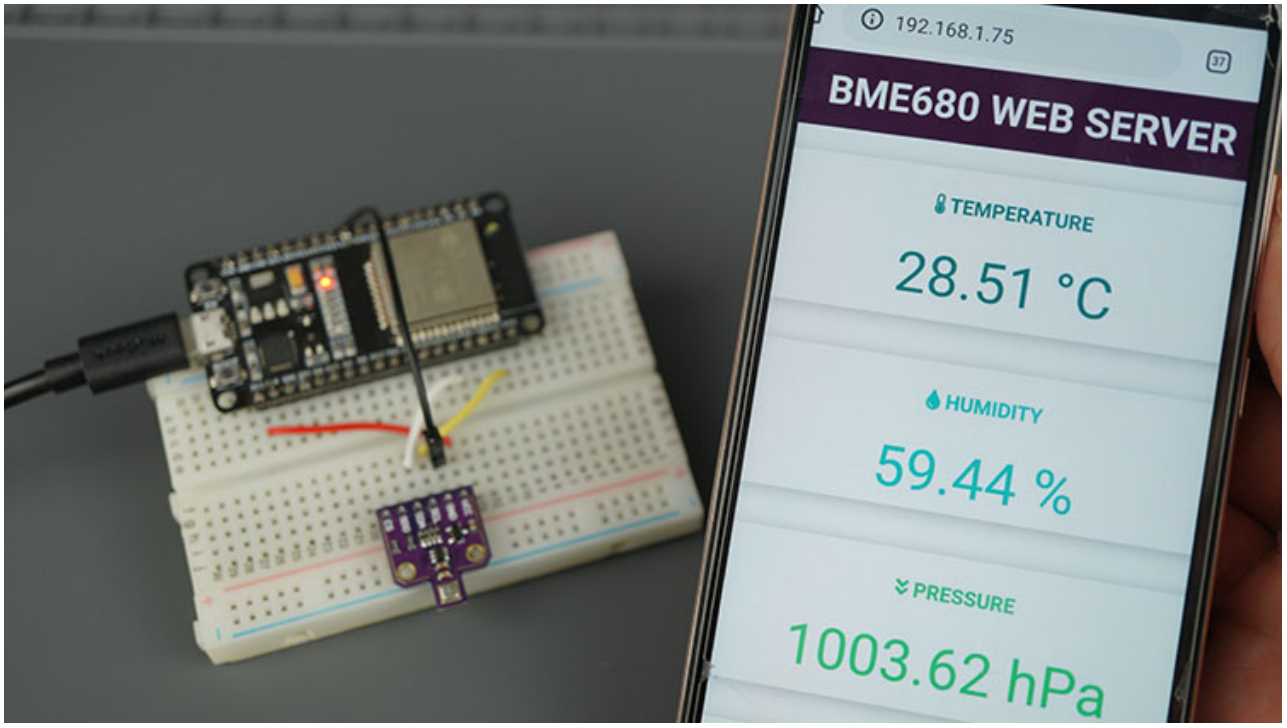
Upload the code to your ESP32 board. Go to **Tools** > **Board** and select the ESP32 board you're using. Go to **Tools** > **Port** and select the port your board is connected to. Then, click the upload button.

Open the Serial Monitor at a baud rate of 115200, press the on-board RST button. The sensor measurements will be displayed.



Code – ESP32 Web Server with BME680

In this section, we provide an example of web server that you can build with the ESP32 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
- AsyncTCP

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/esp32-b

  Permission is hereby granted, free of charge, to any person obtaini
  of this software and associated documentation files.
```

```
  The above copyright notice and this permission notice shall be incl
  copies or substantial portions of the Software.
*********/

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <WiFi.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 18
#define BME_MISO 19
#define BME_MOSI 23
```

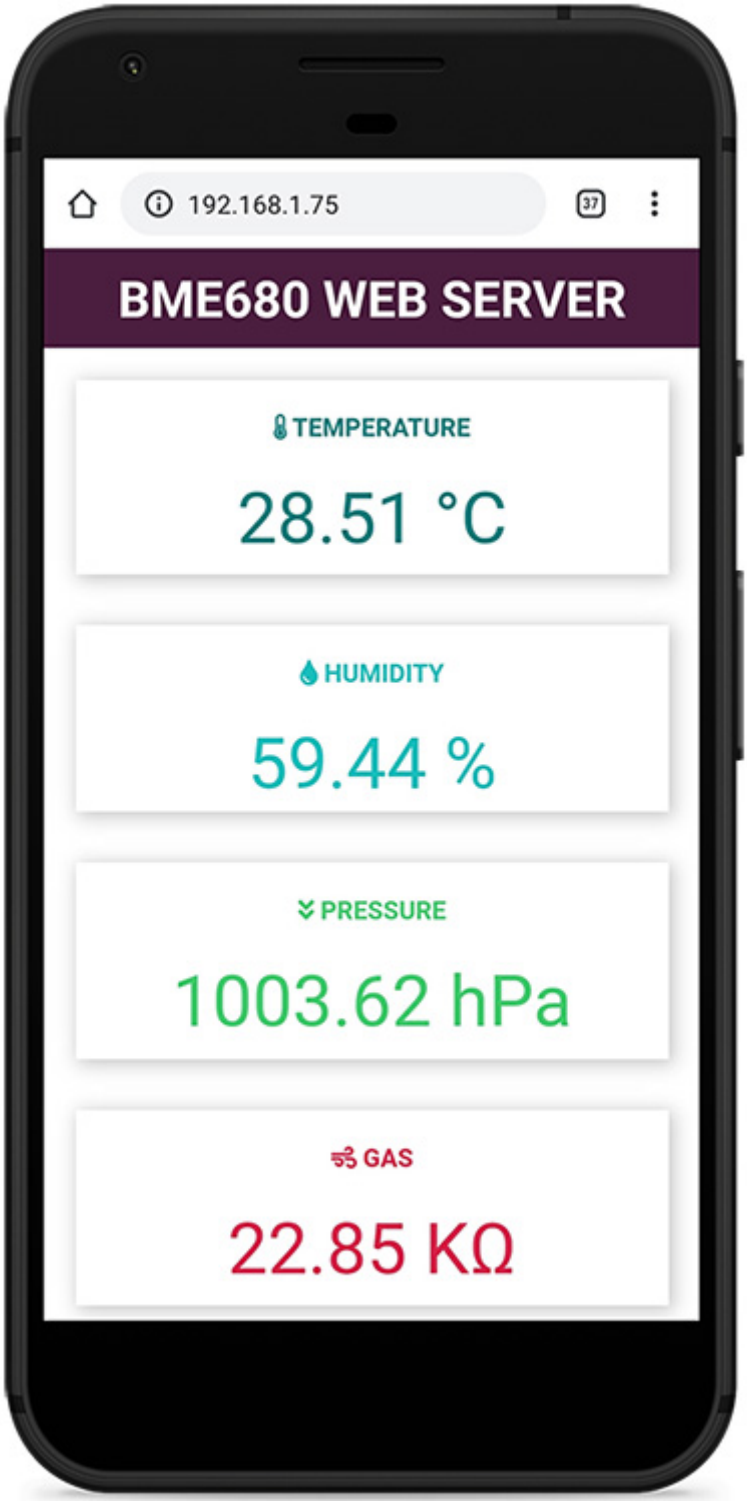
View raw code

Demonstration

After uploading, open the Serial Monitor at a baud rate of 115200 to get the ESP32 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.

```
// Tell BME
```



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 ESP32 board.

Wrapping Up

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

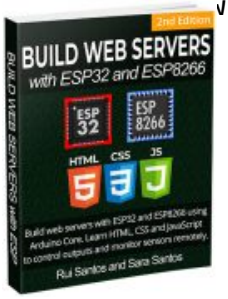
We hope you've found this getting started guide useful. We have guides for other popular sensors:

- ESP32 with **DHT11/DHT22 Temperature and Humidity Sensor** using Arduino IDE
- ESP32 with **BME280** using Arduino IDE (**Pressure, Temperature, Humidity**)
- ESP32 **DS18B20 Temperature Sensor** with Arduino IDE (Single, Multiple, Web Server)
- ESP32 with **BMP180 Barometric Sensor** (Temperature and Pressure)

Learn more about the ESP32 with our resources:

- **Learn ESP32 with Arduino IDE (eBook + video course)**
- More ESP32 Projects and Tutorials …

Thanks for reading.

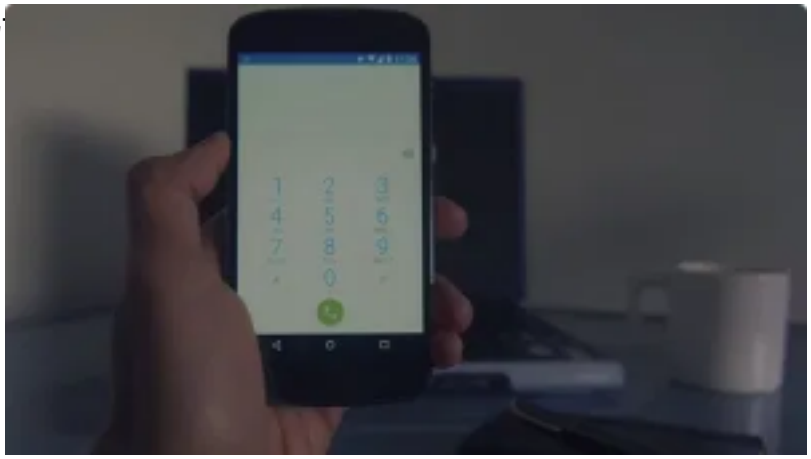


[eBook] Build Web Servers with ESP32 and ESP8266 (2nd Edition)

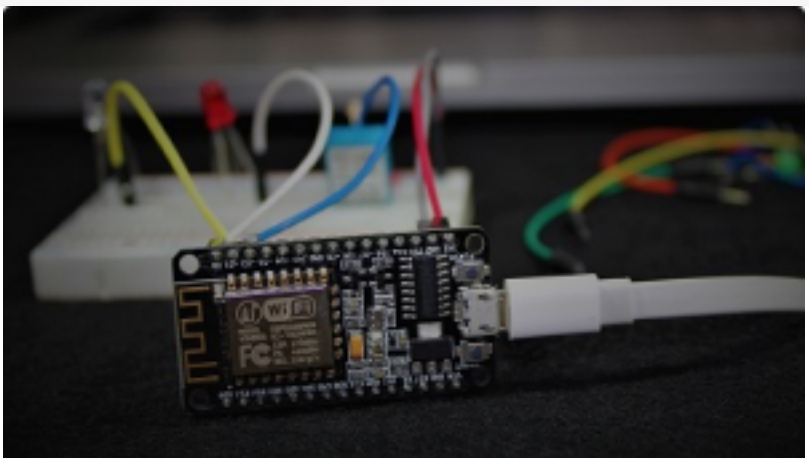
Build Web Server projects with the ESP32 and ESP8266 boards to control outputs and monitor sensors remotely. Learn HTML, CSS, JavaScript and client-server communication protocols **DOWNLOAD »**

Recommended Resources

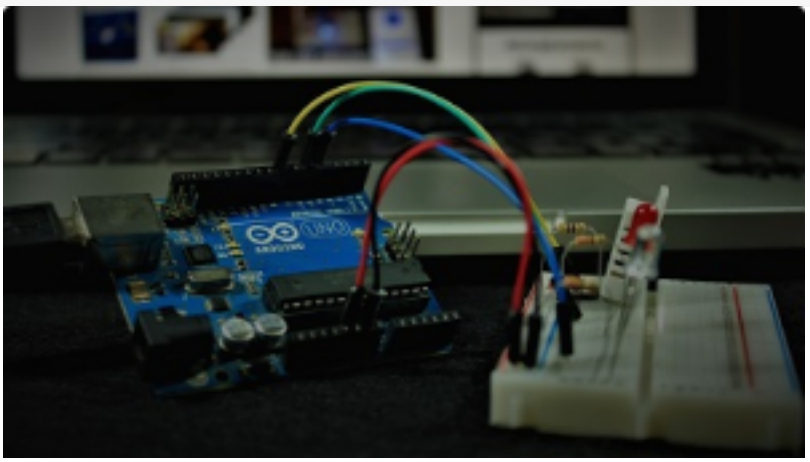
```
</html>)raw
```



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 Weather Station Interface PCB Shield (Temperature, Humidity, Pressure, Date and Time)

ESP32 with Multiple DS18B20 Temperature Sensors

8 thoughts on “ESP32: BME680 Environmental Sensor using Arduino IDE (Gas, Pressure, Humidity, Temperature)”

Ulrich

July 28, 2020 at 5:17 am

Hi,
it seems that there is a conflict according description of SDA and SDO respectively MISO and MOSI in the table “BME680 Pinout”
I think SDA must be MOSI and SDO must be MISO.

Reply

Sara Santos

July 28, 2020 at 9:36 am

Hi.
Thanks for noticing.
It’s fixed now.
Regards,
Sara

Reply

Ian
July 28, 2020 at 5:06 pm

Great tutorial, would cool if you could add sending the values over MQTT

Reply

Ray Bright
August 6, 2020 at 2:45 am

Hint For Others… I've been proposing to use a BMP280 to measure Supply Air Temperature and Duct (Air) Pressure in a ceiling mounted split air conditioning unit. Worked great on the bench BUT after using 6 meters of screened cable to the BMP280 & using I2C it would not work. Further research revealed a limit of maybe less than 1 meter of cable else I2C has lots of problems with SCA / SCL crosstalk. Fortunately I installed at the same time DS18B20 to measure air conditioning temperatures into and out of system and that works fine.
I'm using this ESP8266-E12 as a sender for ESP-NOW and this works fine including a Bluetooth module to my smart phone etc… ENJOY

Reply

Cristian
August 11, 2020 at 4:14 pm

Hi guys,

I'm using the AsyncEventSource to update automatically the readings form 12 sensors, but looks like only 8 of those values are updated automatically. Do you know if there's any limit in the number of events that the handler can manage?

Thanks!

Reply

Rui Santos
September 16, 2020 at 10:18 am

I haven't tested the max amount, but it should be able to use more than 12 sensors. Make sure you have the right ID and updating the right HTML elements

Reply

paulo lima

September 7, 2020 at 9:27 pm

Hi Sara.
I made this project everything works but comparing whith several weather stations including my father weather station and all values are lower in esp32, i only cant get values for gas is there any way to change, there is a part of your explication where you speak abought altitude but not in code can this be a problem ? I made two projects of BME680 one for me other for my father and both with difeferent lectures and both low values.
Thanks for your great projects and for your usual disponibility.

Reply

SAJAL BHARGAVA

May 3, 2022 at 6:32 am

do we need it to power it for 48 hours before running the code ?

Reply

Leave a Comment

Name *

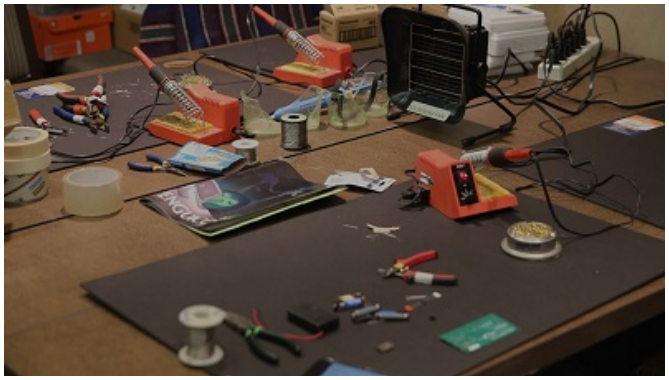
Email *

Website

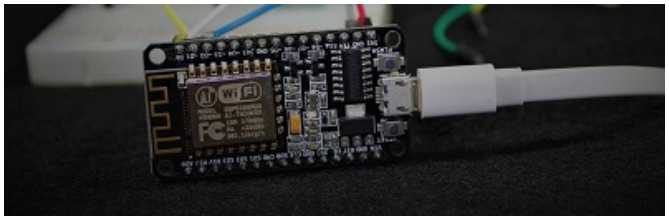
Notify me of follow-up comments by email.

Notify me of new posts by email.

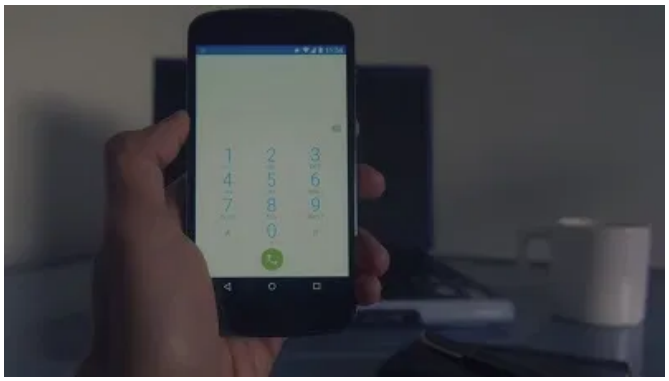
Post Comment



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 Web Servers with ESP32 and ESP8266 » boards to control outputs and monitor sensors remotely.