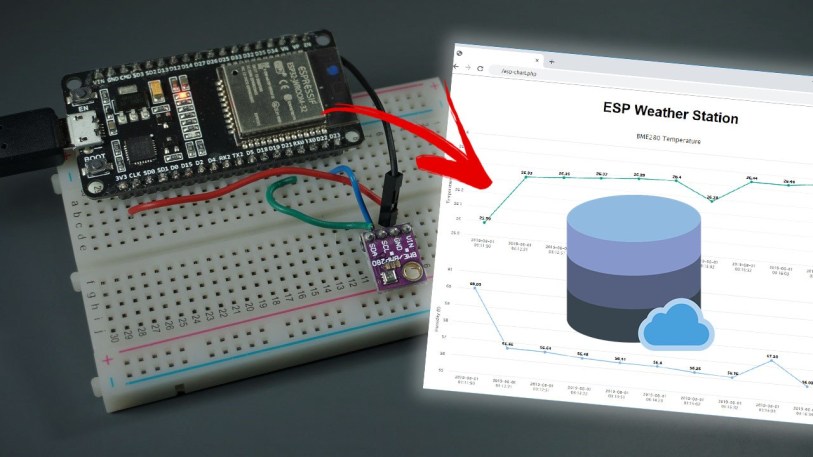
**Visualize Your Sensor Readings from Anywhere in the World (ESP32/ESP8266 + MySQL + PHP)**

In this project, you’ll create a web page that displays sensor readings in a plot that you can access from anywhere in the world.

In summary, you’ll build an ESP32 or ESP8266 client that makes a request to a PHP script to publish sensor readings in a MySQL database.



We’ll be using a BME280 sensor connected to an ESP board. To create this project, you’ll use these technologies:

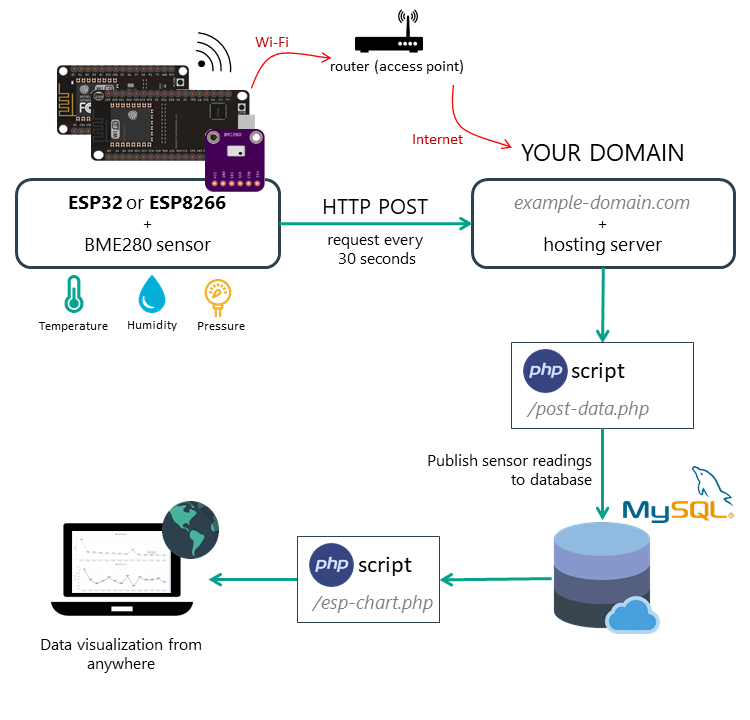
* ESP32 or ESP8266 programmed with Arduino IDE
* Hosting server and domain name
* PHP script to insert data into MySQL database and display it on a web page
* MySQL database to store readings
* PHP script to plot data from database in charts

You might also find helpful reading these projects:

* [ESP32/ESP8266 Insert Data into MySQL Database using PHP and Arduino IDE](https://randomnerdtutorials.com/esp32-esp8266-mysql-database-php/)
* [ESP32/ESP8266 Plot Sensor Readings in Real Time Charts – Web Server](https://randomnerdtutorials.com/esp32-esp8266-plot-chart-web-server/)

**1. Hosting Your PHP Application and MySQL Database**

The goal of this project is to have your own domain name and hosting account that allows you to store sensor readings from the ESP32 or ESP8266.

[](https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/08/ESP32-MySQL-Charts-Project-Overview.png?ssl=1)

I recommend using one of the following hosting services that can handle all the project requirements:

* [Bluehost (user-friendly with cPanel)](https://randomnerdtutorials.com/bluehost): free domain name when you sign up for the 3-year plan. I recommend choosing the unlimited websites option;
* [Digital Ocean](https://randomnerdtutorials.com/digitalocean): Linux server that you manage through a command line. I only recommended this option for advanced users.

[Get Hosting and Domain Name with Bluehost »](https://randomnerdtutorials.com/bluehost)

When buying a hosting account, you’ll also have to purchase a domain name. This is what makes this project interesting: you’ll be able to go your domain name (http://example.com) and see your ESP readings.

If you like our projects, you might consider signing up to one of the recommended hosting services, because you’ll be supporting our work.

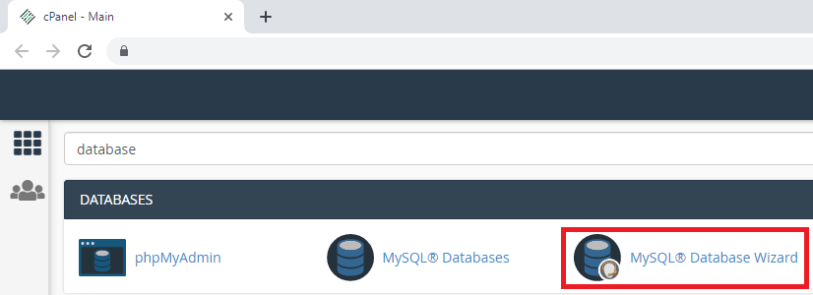
**Note:**you can also run PHP and MySQL on a Raspberry Pi, but the purpose of this tutorial is to publish readings in your own domain name that you can access from anywhere in the world.

**2. Preparing Your MySQL Database**

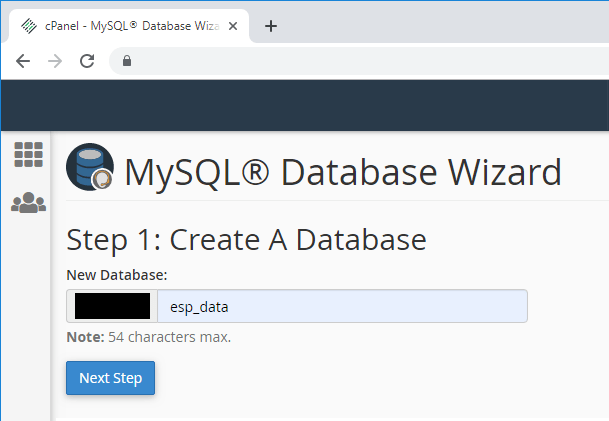
After signing up for a [hosting account and setting up a domain name](https://randomnerdtutorials.com/bluehost), you can login to your cPanel or similar dashboard. After that, follow the next steps to create your database, username, password and SQL table.

**Creating a database and user**

**1.** Type “database” in the search bar and select “MySQL Database Wizard”.

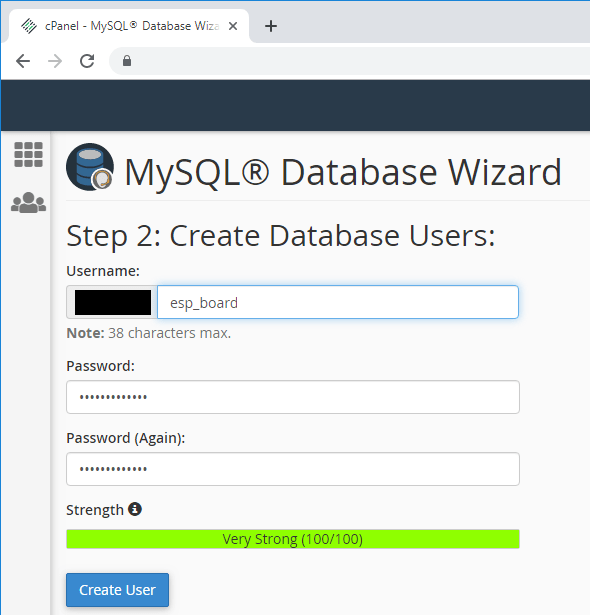


**2.** Enter your desired Database name. In my case, the database name is esp\_data. Then, press the “Next Step” button:



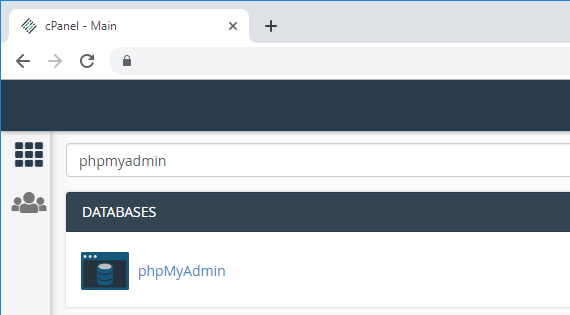
**Note:**later you’ll have to use the database name with the prefix that your host gives you (my database prefix in the screenshot above is blurred). I’ll refer to it as  example\_esp\_data from now on.

**3.** Type your Database username and set a password.

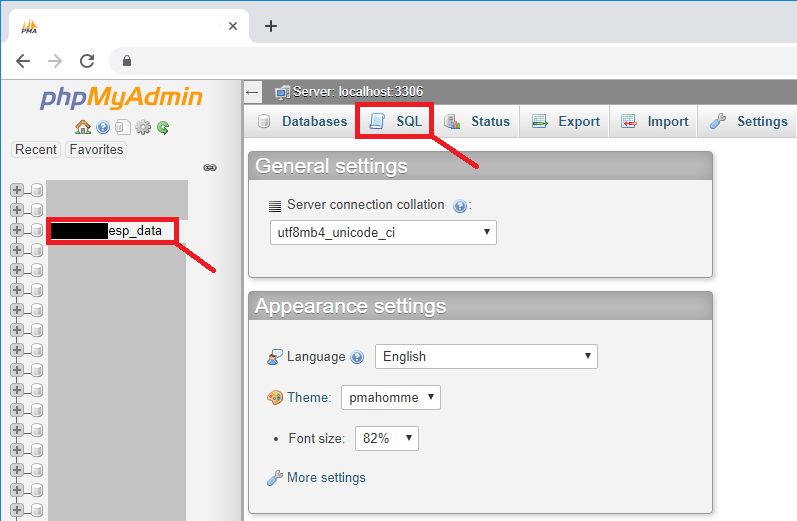


**Creating a SQL table**

After creating your database and user, go back to cPanel dashboard and search for “phpMyAdmin”.



In the left sidebar, select your database name example\_esp\_data and open the “SQL” tab.



**Important:**make sure you’ve opened the example\_esp\_data database. Then, click the SQL tab.

Copy the SQL query in the following snippet:

CREATE TABLE Sensor (

id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

value1 VARCHAR(10),

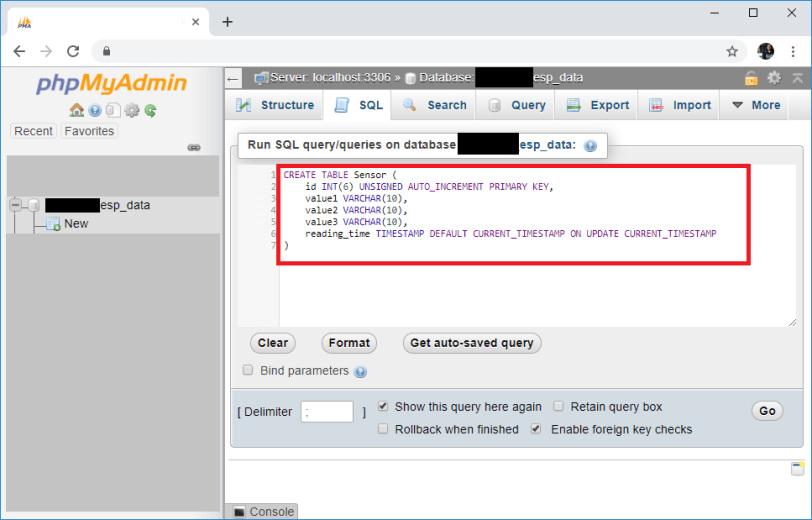
value2 VARCHAR(10),

value3 VARCHAR(10),

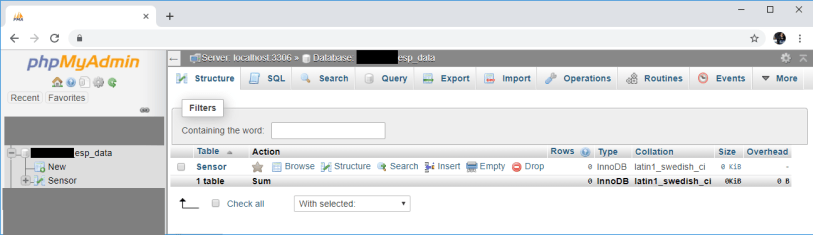
reading\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

)

Paste it in the SQL query field (highlighted with a red rectangle) and press the “Go” button to create your table:



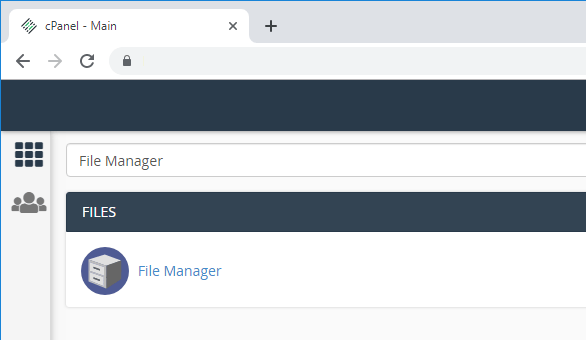
After that, you should see your newly created table called Sensor in the example\_esp\_data database as shown in the figure below:

[](https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/06/ESP32-ESP8266-PHPMyAdmin-View-SQL-Database.png?ssl=1)

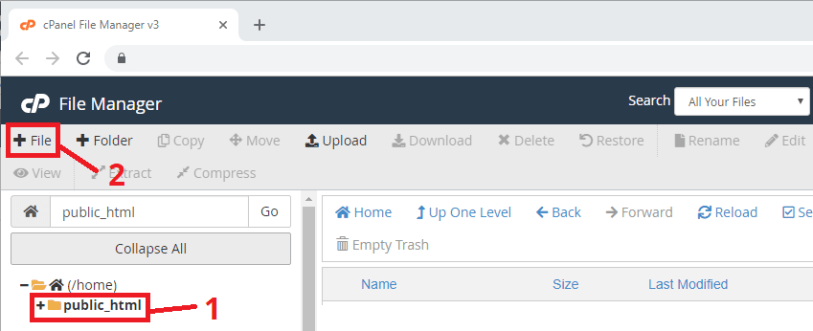
**3. PHP Script HTTP POST – Insert Data in MySQL Database**

In this section, we’re going to create a PHP script that receives incoming requests from the ESP32 or ESP8266 and inserts the data into a MySQL database.

If you’re using a hosting provider with cPanel, you can search for “File Manager”:

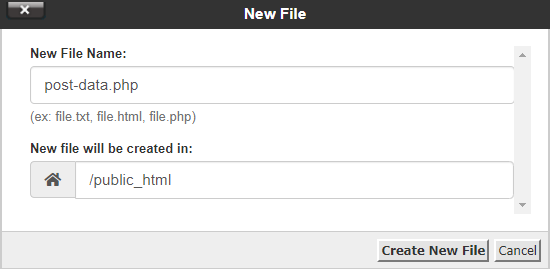


Then, select the **public\_html** option and press the “+ File” button to create a new *.php* file.



**Note:**if you’re following this tutorial and you’re not familiar with PHP or MySQL, I recommend creating these exact files. Otherwise, you’ll need to modify the ESP sketch provided with different URL paths.

Create a new file in **/public\_html** with this exact name and extension: *post-data.php*



Edit the newly created file (*post-data.php*) and copy the following snippet:

<?php

$servername = "localhost";

$dbname = "example\_esp\_data";

$username = "example\_esp\_board";

$password = "YOUR\_USER\_PASSWORD";

// Keep this API Key value to be compatible with the ESP32 code provided in the project page. If you change this value, the ESP32 sketch needs to match

$api\_key\_value = "tPmAT5Ab3j7F9";

$api\_key = $value1 = $value2 = $value3 = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$api\_key = test\_input($\_POST["api\_key"]);

if($api\_key == $api\_key\_value) {

$value1 = test\_input($\_POST["value1"]);

$value2 = test\_input($\_POST["value2"]);

$value3 = test\_input($\_POST["value3"]);

$sql = "INSERT INTO Sensor (value1, value2, value3)

VALUES ('" . $value1 . "', '" . $value2 . "', '" . $value3 . "')";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

}

else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

}

else {

echo "Wrong API Key provided.";

}

}

else {

echo "No data posted with HTTP POST.";

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

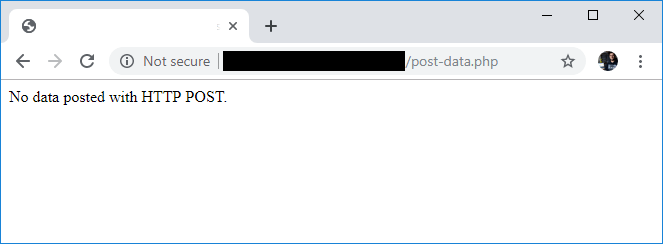
$data = htmlspecialchars($data);

return $data;

}

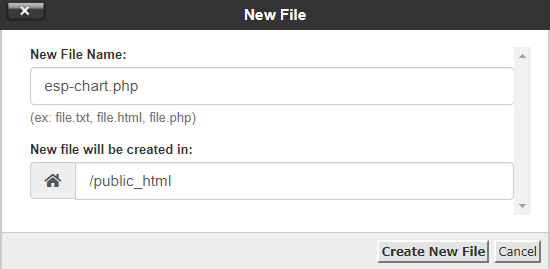
If you try to access your domain name in the next URL path, you’ll see the message:

http://example.com/post-data.php



**4. PHP Script – Visualize Database Content in a Chart**

Create another PHP file in the **/public\_html** directory that will plot the database content in a chart on a web page. Name your new file: *esp-chart.php*



Edit the newly created file (*esp-chart.php*) and copy the following code:

<?php

$servername = "localhost";

$dbname = "example\_esp\_data";

$username = "example\_esp\_board";

$password = "YOUR\_USER\_PASSWORD";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "SELECT id, value1, value2, value3, reading\_time FROM Sensor order by reading\_time desc limit 40";

$result = $conn->query($sql);

while ($data = $result->fetch\_assoc()){

$sensor\_data[] = $data;

}

$readings\_time = array\_column($sensor\_data, 'reading\_time');

// \*\*\*\*\*\*\* Uncomment to convert readings time array to your timezone \*\*\*\*\*\*\*\*

/\*$i = 0;

foreach ($readings\_time as $reading){

// Uncomment to set timezone to - 1 hour (you can change 1 to any number)

$readings\_time[$i] = date("Y-m-d H:i:s", strtotime("$reading - 1 hours"));

// Uncomment to set timezone to + 4 hours (you can change 4 to any number)

//$readings\_time[$i] = date("Y-m-d H:i:s", strtotime("$reading + 4 hours"));

$i += 1;

}\*/

$value1 = json\_encode(array\_reverse(array\_column($sensor\_data, 'value1')), JSON\_NUMERIC\_CHECK);

$value2 = json\_encode(array\_reverse(array\_column($sensor\_data, 'value2')), JSON\_NUMERIC\_CHECK);

$value3 = json\_encode(array\_reverse(array\_column($sensor\_data, 'value3')), JSON\_NUMERIC\_CHECK);

$reading\_time = json\_encode(array\_reverse($readings\_time), JSON\_NUMERIC\_CHECK);

/\*echo $value1;

echo $value2;

echo $value3;

echo $reading\_time;\*/

$result->free();

$conn->close();

?>

<!DOCTYPE html>

<html>

<meta name="viewport" content="width=device-width, initial-scale=1">

<script src="https://code.highcharts.com/highcharts.js"></script>

<style>

body {

min-width: 310px;

max-width: 1280px;

height: 500px;

margin: 0 auto;

}

h2 {

font-family: Arial;

font-size: 2.5rem;

text-align: center;

}

</style>

<body>

<h2>ESP Weather Station</h2>

<div id="chart-temperature" class="container"></div>

<div id="chart-humidity" class="container"></div>

<div id="chart-pressure" class="container"></div>

<script>

var value1 = <?php echo $value1; ?>;

var value2 = <?php echo $value2; ?>;

var value3 = <?php echo $value3; ?>;

var reading\_time = <?php echo $reading\_time; ?>;

var chartT = new Highcharts.Chart({

chart:{ renderTo : 'chart-temperature' },

title: { text: 'BME280 Temperature' },

series: [{

showInLegend: false,

data: value1

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#059e8a' }

},

xAxis: {

type: 'datetime',

categories: reading\_time

},

yAxis: {

title: { text: 'Temperature (Celsius)' }

//title: { text: 'Temperature (Fahrenheit)' }

},

credits: { enabled: false }

});

var chartH = new Highcharts.Chart({

chart:{ renderTo:'chart-humidity' },

title: { text: 'BME280 Humidity' },

series: [{

showInLegend: false,

data: value2

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

}

},

xAxis: {

type: 'datetime',

//dateTimeLabelFormats: { second: '%H:%M:%S' },

categories: reading\_time

},

yAxis: {

title: { text: 'Humidity (%)' }

},

credits: { enabled: false }

});

var chartP = new Highcharts.Chart({

chart:{ renderTo:'chart-pressure' },

title: { text: 'BME280 Pressure' },

series: [{

showInLegend: false,

data: value3

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#18009c' }

},

xAxis: {

type: 'datetime',

categories: reading\_time

},

yAxis: {

title: { text: 'Pressure (hPa)' }

},

credits: { enabled: false }

});

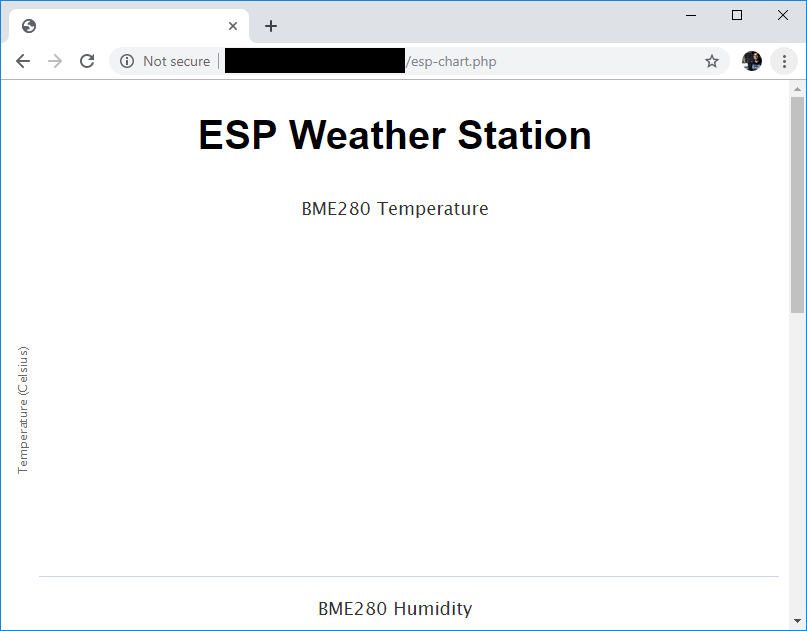
</script>

</body>

</html>

If you try to access your domain name in the following URL path, you’ll see the following:

<http://example.com/esp-chart.php>



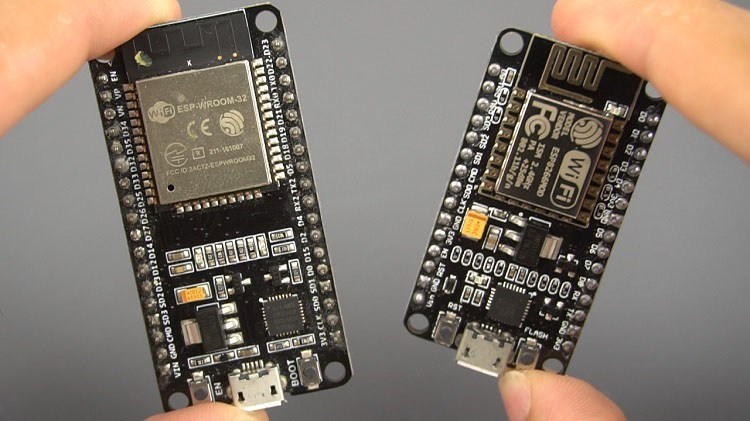
That’s it! If you see three empty charts in your browser, it means that everything is ready. In the next section, you’ll learn how to publish your ESP32 or ESP8266 sensor readings.

To build the charts, we’ll use the [Highcharts library](https://www.highcharts.com/docs/" \t "_blank). We’ll create three charts: temperature, humidity and pressure over time.

The charts display a maximum of 40 data points, and a new reading is added every 30 seconds, but you change these values in your code.

**5. Preparing Your ESP32 or ESP8266**

This project is compatible with both the ESP32 and ESP8266 boards. You just need to assemble a simple circuit and upload the sketch provided to insert temperature, humidity, pressure and more into your database every 30 seconds.



**Parts Required**

For this example we’ll get sensor readings from the **BME280** sensor.

Here’s a list of parts you need to build the circuit for this project:

* [ESP32 board](https://makeradvisor.com/tools/esp32-dev-board-wi-fi-bluetooth/)  (read [Best ESP32 dev boards](https://makeradvisor.com/esp32-development-boards-review-comparison/))
* Alternative – [ESP8266](https://makeradvisor.com/tools/esp8266-esp-12e-nodemcu-wi-fi-development-board/) board (read [Best ESP8266 dev boards](https://makeradvisor.com/best-esp8266-wi-fi-development-board/))
* [BME280 sensor](https://makeradvisor.com/tools/bme280-sensor-module/)
* [Jumper wires](https://makeradvisor.com/tools/jumper-wires-kit-120-pieces/)
* [Breadboard](https://makeradvisor.com/tools/mb-102-solderless-breadboard-830-points/)

You can use the preceding links or go directly to [MakerAdvisor.com/tools](https://makeradvisor.com/tools/?utm_source=rnt&utm_medium=post&utm_campaign=post) to find all the parts for your projects at the best price!

**Schematics**

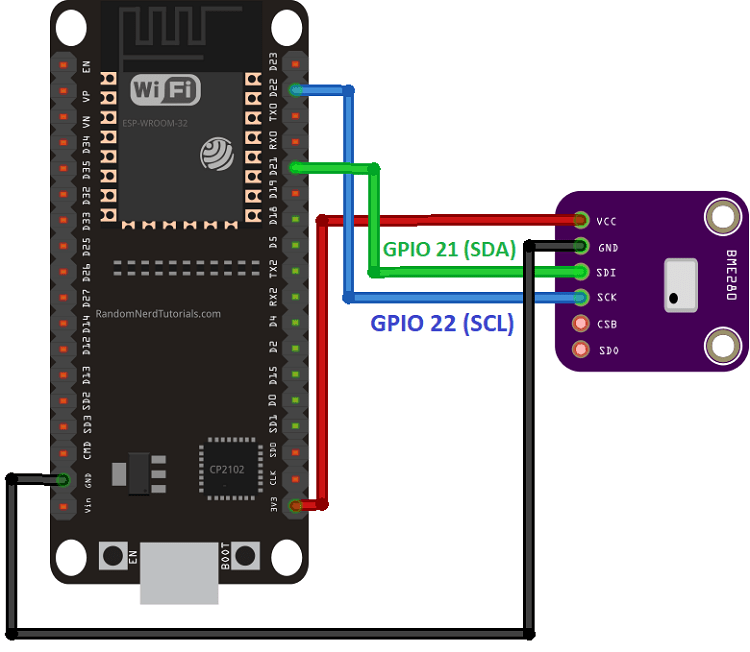
The **BME280** sensor module we’re using communicates via I2C communication protocol, so you need to connect it to the ESP32 or ESP8266 I2C pins.

**BME280 wiring to ESP32**

The ESP32 I2C pins are:

* **GPIO 22:** SCL (SCK)
* **GPIO 21:** SDA (SDI)

So, assemble your circuit as shown in the next schematic diagram ([read complete Guide for ESP32 with BME280](https://randomnerdtutorials.com/esp32-bme280-arduino-ide-pressure-temperature-humidity/)).



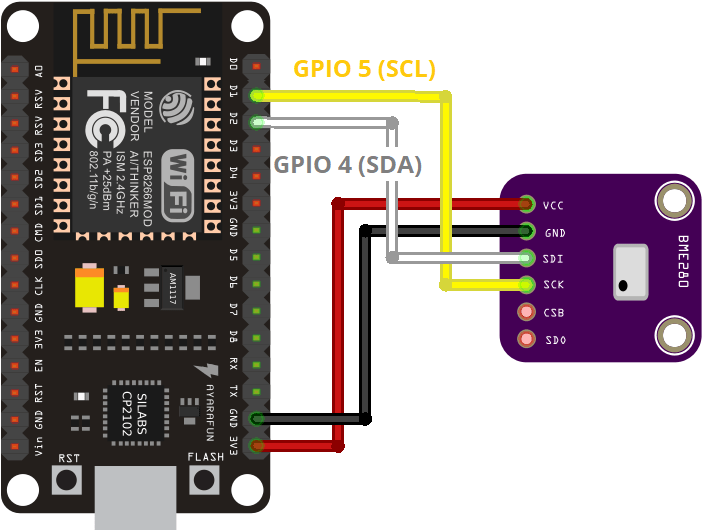
**Recommended reading:** [ESP32 Pinout Reference Guide](https://randomnerdtutorials.com/esp32-pinout-reference-gpios/)

**BME280 wiring to ESP8266**

The ESP8266 I2C pins are:

* **GPIO 5**(D1): SCL (SCK)
* **GPIO 4**(D2): SDA (SDI)

Assemble your circuit as in the next schematic diagram if you’re using an ESP8266 board ([read complete Guide for ESP8266 with BME280](https://randomnerdtutorials.com/esp8266-bme280-arduino-ide/)).



**Recommended reading:** [ESP8266 Pinout Reference Guide](https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/)

**ESP32/ESP8266 Code**

We’ll program the ESP32/ESP8266 using Arduino IDE, so you must have the ESP32/ESP8266 add-on installed in your Arduino IDE. Follow one of the next tutorials depending on the board you’re using:

* [Install the ESP32 Board in Arduino IDE](https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/) – you also need to [install the BME280 Library and Adafruit\_Sensor library](https://randomnerdtutorials.com/esp32-bme280-arduino-ide-pressure-temperature-humidity/)
* [Install the ESP8266 Board in Arduino IDE](https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/) – you also need to [install the BME280 Library and Adafruit\_Sensor library](https://randomnerdtutorials.com/esp8266-bme280-arduino-ide/)

After installing the necessary board add-ons, copy the following code to your Arduino IDE, but don’t upload it yet. You need to make some changes to make it work for you.

#ifdef ESP32

#include <WiFi.h>

#include <HTTPClient.h>

#else

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#endif

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

const char\* ssid = "REPLACE\_WITH\_YOUR\_SSID";

const char\* password = "REPLACE\_WITH\_YOUR\_PASSWORD";

const char\* serverName = "http://example.com/post-data.php";

// Keep this API Key value to be compatible with the PHP code provided

String apiKeyValue = "tPmAT5Ab3j7F9";

/\*#include <SPI.h>

#define BME\_SCK 18

#define BME\_MISO 19

#define BME\_MOSI 23

#define BME\_CS 5\*/

Adafruit\_BME280 bme; // I2C

//Adafruit\_BME280 bme(BME\_CS); // hardware SPI

//Adafruit\_BME280 bme(BME\_CS, BME\_MOSI, BME\_MISO, BME\_SCK); // software SPI

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.println("Connecting");

while(WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("Connected to WiFi network with IP Address: ");

Serial.println(WiFi.localIP());

// (you can also pass in a Wire library object like &Wire2)

bool status = bme.begin(0x76);

if (!status) {

Serial.println("Could not find BME280 sensor, check wiring or change I2C address!");

while (1);

}

}

void loop() {

//Check WiFi connection status

if(WiFi.status()== WL\_CONNECTED){

HTTPClient http;

// Your Domain name with URL path or IP address with path

http.begin(serverName);

// Specify content-type header

http.addHeader("Content-Type", "application/x-www-form-urlencoded");

// Prepare your HTTP POST request data

String httpRequestData = "api\_key=" + apiKeyValue + "&value1=" +

String(bme.readTemperature()) + "&value2=" +

String(bme.readHumidity()) + "&value3=" +

String(bme.readPressure()/100.0F) + "";

Serial.println(httpRequestData);

//For test

//String httpRequestData = "api\_key=tPmAT5Ab3j7F9&value1=24.75&value2=49.54&value3=1005.14";

// Send HTTP POST request

int httpResponseCode = http.POST(httpRequestData);

// If you need an HTTP request with a content type: text/plain

//http.addHeader("Content-Type", "text/plain");

//int httpResponseCode = http.POST("Hello, World!");

// If you need an HTTP request with a content type: application/json

//http.addHeader("Content-Type", "application/json");

//int httpResponseCode = http.POST("{\"value1\":\"19\",\"value2\":\"67\",\"value3\":\"78\"}");

if (httpResponseCode>0) {

Serial.print("HTTP Response code: ");

Serial.println(httpResponseCode);

}

else {

Serial.print("Error code: ");

Serial.println(httpResponseCode);

}

// Free resources

http.end();

}

else {

Serial.println("WiFi Disconnected");

}

//Send an HTTP POST request every 30 seconds

delay(30000);

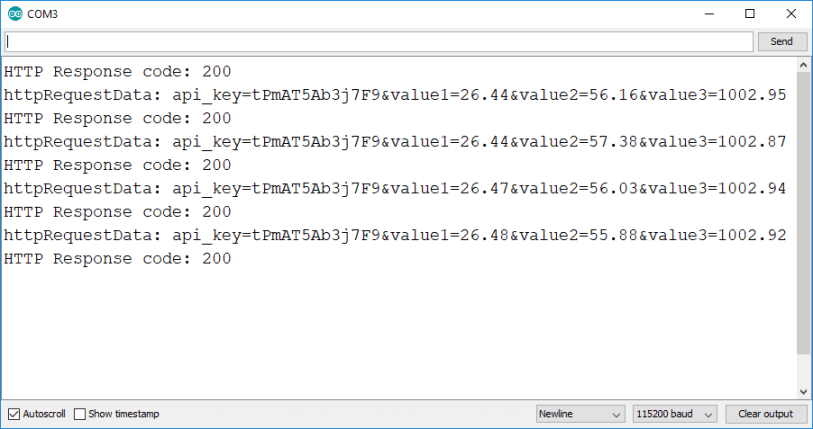
}

* Import all the libraries to make it work (it will import either the ESP32 or ESP8266 libraries based on the selected board in your Arduino IDE)
* The apiKeyValue is just a random string that you can modify. It’s used for security reasons, so only anyone that knows your API key can publish data to your database

You can comment the httpRequestData variable and use

String httpRequestData = "api\_key=tPmAT5Ab3j7F9&value1=24.75&value2=49.54&value3=1005.14";

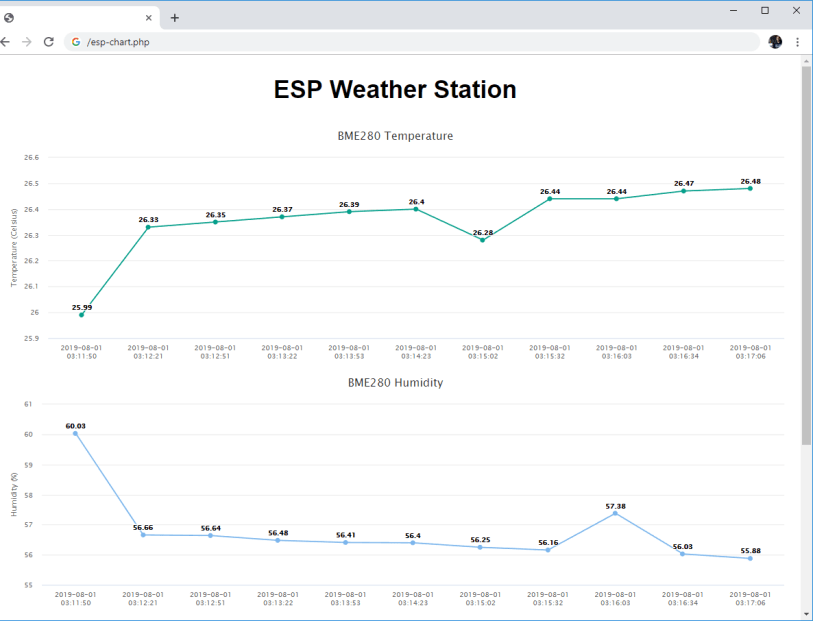
If everything is correct, this is what you should see in your Arduino IDE Serial Monitor:



If you open your domain name in this URL path:

http://example.com/esp-chart.php

You should see the all the readings stored in your database. Refresh the web page to see the latest readings:

[](https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/08/esp32-esp8266-publishing-readings-to-MySQL-database-open-visualize-plot-charts.png?ssl=1)

You can also go to phpMyAdmin to manage the data stored in your Sensor table. You can delete it, edit, etc…

