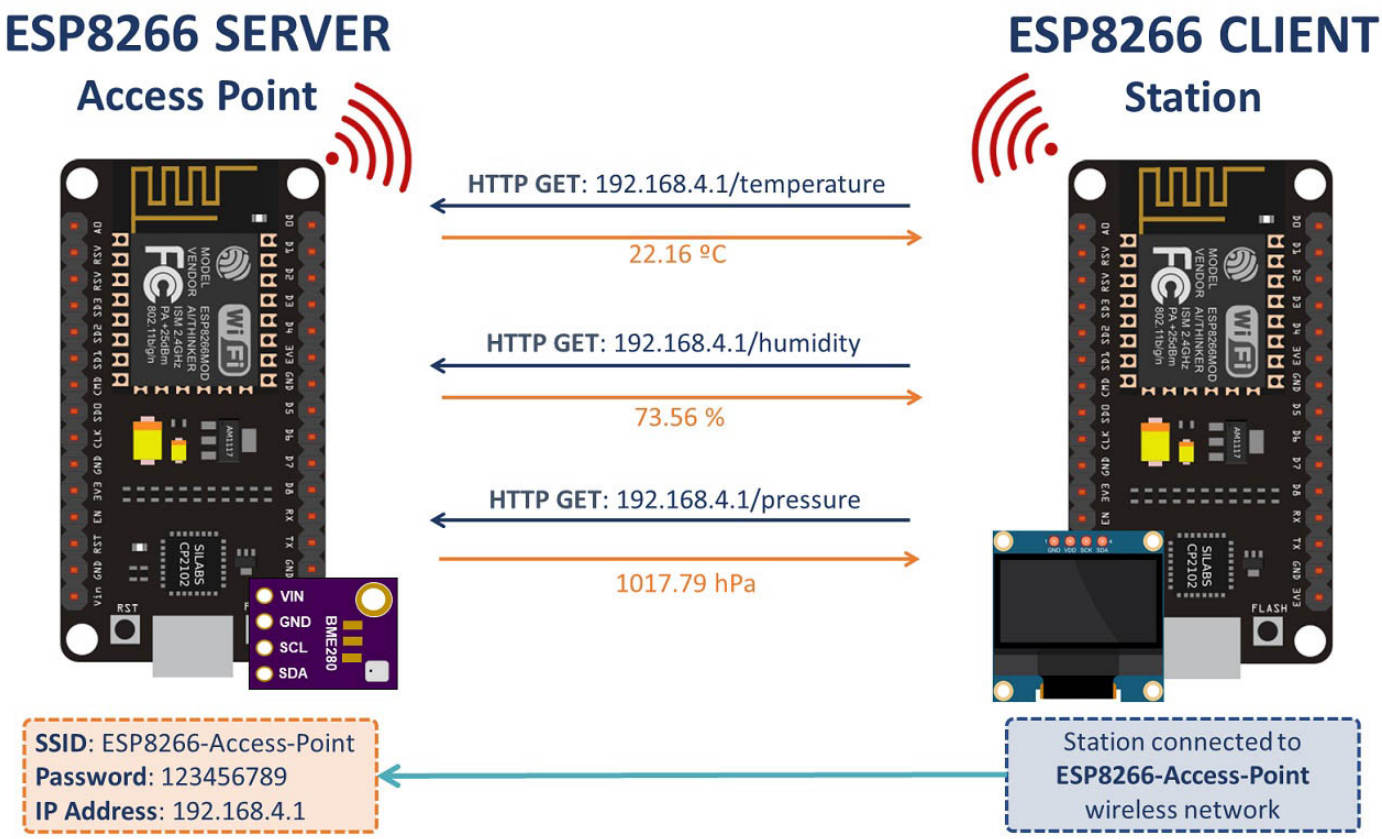
Project Overview

To better understand how everything works, take a look at the following diagram.

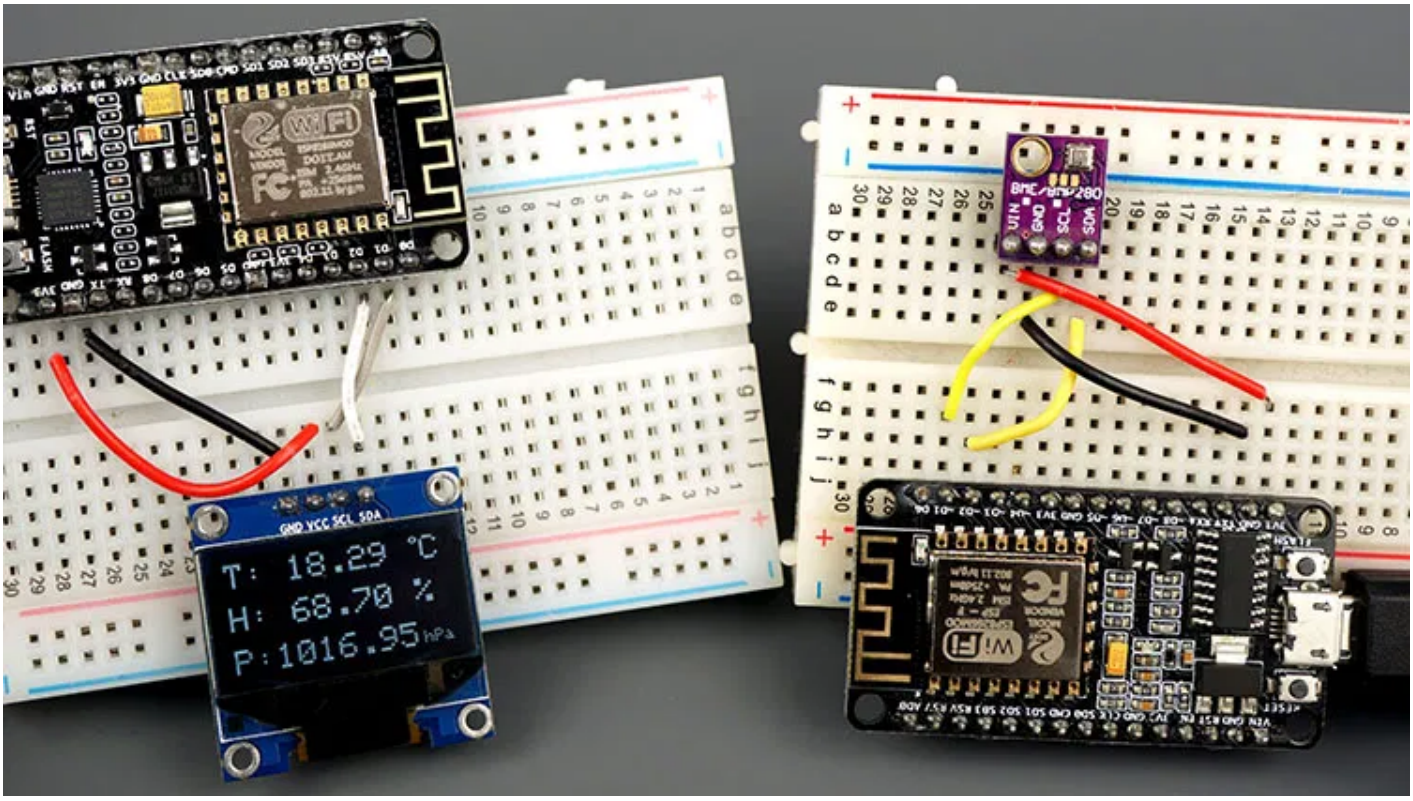


- The ESP8266 server creates its own wireless network (ESP8266 Soft-Access Point). So, other Wi-Fi devices can connect to that network (**SSID:** ESP8266-Access-Point, **Password:** 123456789).
- The ESP8266 client is set as a station. So, it can connect to the ESP8266 server wireless network.
- The client can make HTTP GET requests to the server to request sensor data or any other information. It just needs to use the IP address of the server to make a request on a certain route: `/temperature`, `/humidity` or `/pressure`.
- The server listens for incoming requests and sends an appropriate response with the readings.
- The client receives the readings and displays them on the OLED display.

As an example, the ESP8266 client requests temperature, humidity and pressure to the server by making requests on the server IP address followed by `/temperature`, `/humidity` and `/pressure`, respectively (HTTP GET).

The ESP8266 server is listening on those routes and when a request is made, it sends the corresponding sensor readings via HTTP response.

Parts Required



For this tutorial, you need the following parts:

- 2x ESP8266 Development boards – read Best ESP8266 Boards Comparison
- BME280 sensor
- I2C SSD1306 OLED display
- Jumper Wires
- Breaboard

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!

Installing Libraries

For this tutorial you need to install the following libraries:

Asynchronous Web Server Libraries

We'll use the following libraries to handle HTTP request:

- ESPAsyncWebServer library (download ESPAsyncWebServer library)
- ESPAsync TCP library (download ESPAsyncTCP library)

These libraries are not available to install through the Library Manager. So, you need to unzip the libraries and move them to the Arduino IDE installation libraries folder.

Alternatively, you can go to **Sketch** > **Include Library** > **Add .ZIP library…** and select the libraries you've just downloaded.

You may also like: DHT11/DHT22 Asynchronous Web Server with the ESP8266

BME280 Libraries

The following libraries can be installed through the Arduino Library Manager. Go to **Sketch** > **Include Library**> **Manage Libraries** and search for the library name.

- Adafruit_BME280 library
- Adafruit unified sensor library

You may also like: Guide for BME280 with ESP8266

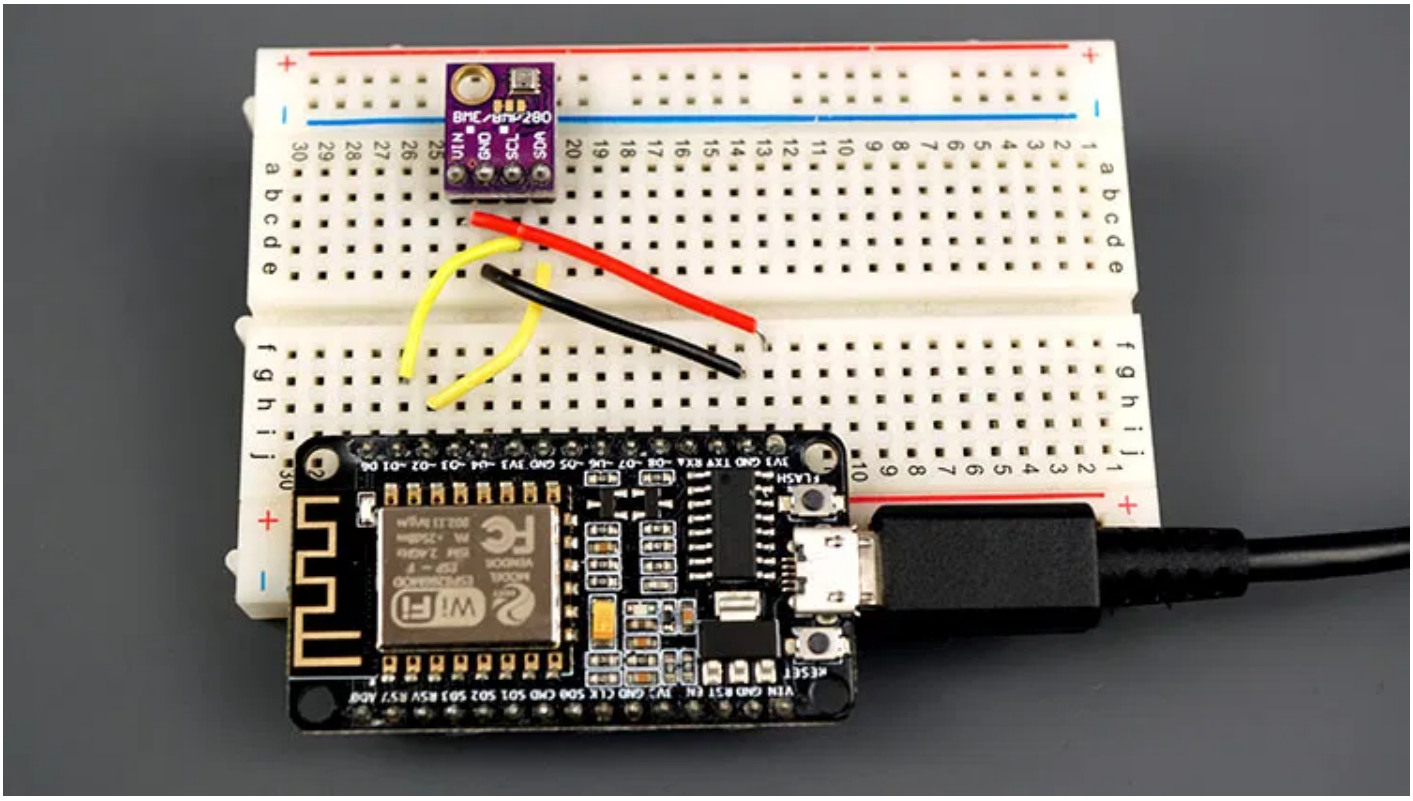
I2C SSD1306 OLED Libraries

To interface with the OLED display you need the following libraries. These can be installed through the Arduino Library Manager. Go to **Sketch** > **Include Library**> **Manage Libraries** and search for the library name.

- Adafruit SSD1306
- Adafruit GFX Library

You may also like: Complete Guide for SSD1306 OLED Display with ESP8266

#1 ESP8266 Server (Access Point)

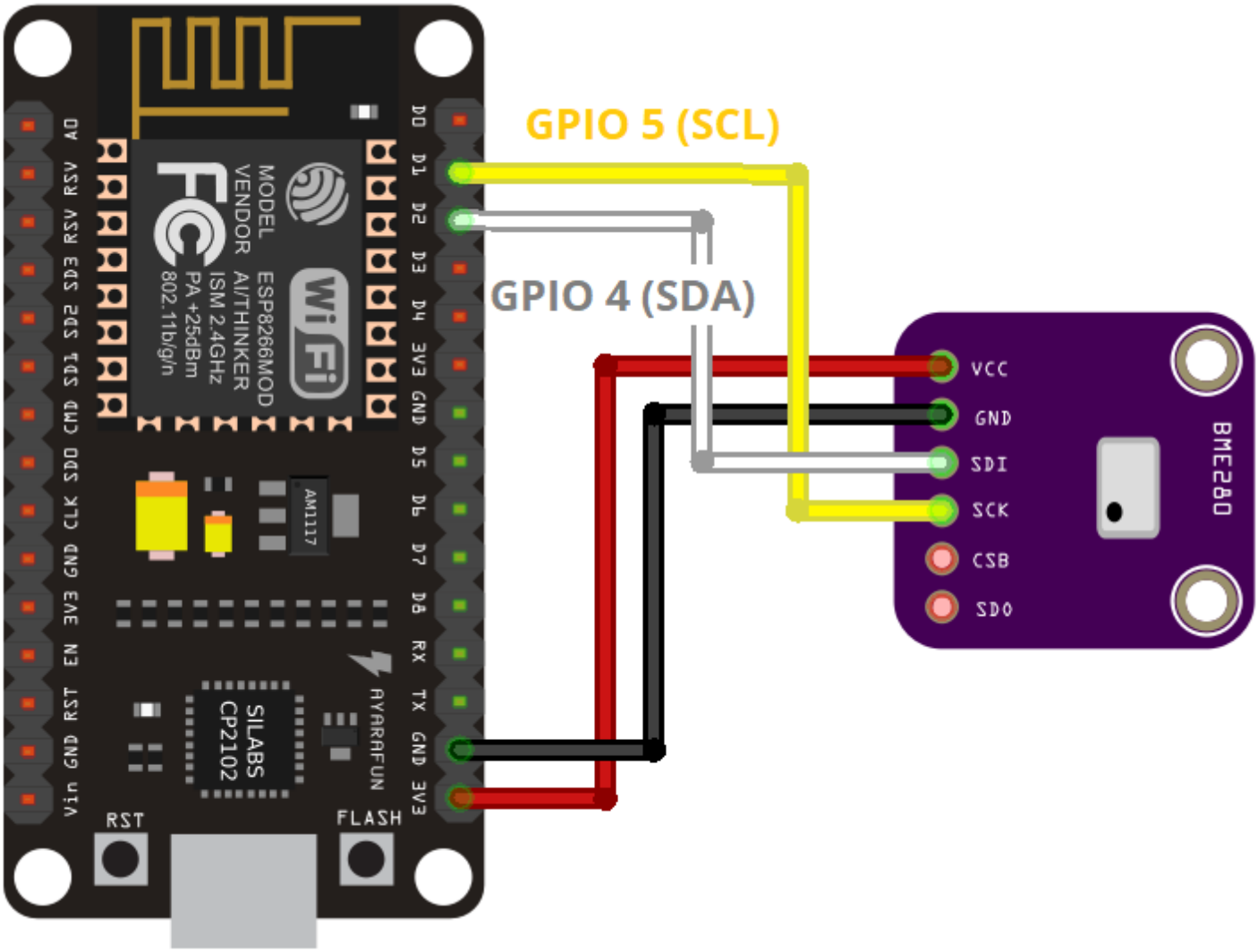


The ESP8266 server is an Access Point (AP), that listens for requests on the `/temperature`, `/humidity` and `/pressure` URLs. When it gets requests on those URLs, it sends the latest BME280 sensor readings.

For testing, we’re using a BME280 sensor, but you can use any other sensor by modifying a few lines of code (for example: DHT11/DHT22 or DS18B20).

Schematic Diagram

Wire the ESP8266 to the BME280 sensor as shown in the following schematic diagram.



BME280	ESP8266
VIN/VCC	3.3V
GND	GND
SCL	GPIO 5 (D1)
SDA	GPIO 4 (D2)

Arduino Sketch for #1 ESP8266 Server

Upload the following code to your board.

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp826

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

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

// Import required libraries

#include <ESP8266WiFi.h>
#include "ESPAsyncWebServer.h"

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

// Set your access point network credentials
const char* ssid = "ESP8266-Access-Point";
const char* password = "123456789";

/*#include <SPI.h>
#define BME_SCK 18
#define BME_MISO 19
```

View raw code

How the code works

Start by including the necessary libraries. Include the `ESP8266WiFi.h` library and the `ESPAsyncWebServer.h` library to handle incoming HTTP requests.

```
#include <ESP8266WiFi.h>
#include "ESPAsyncWebServer.h"
```

Include the following libraries to interface with the BME280 sensor.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

In the following variables, define your access point network credentials:

```
const char* ssid = "ESP8266-Access-Point";
const char* password = "123456789";
```

We're setting the SSID to `ESP8266-Access-Point`, but you can give it any other name. You can also change the password. By default, its set to `123456789`.

Create an instance for the BME280 sensor called `bme`.

```
Adafruit_BME280 bme;
```

Create an asynchronous web server on port 80.

```
AsyncWebServer server(80);
```

Then, create three functions that return the temperature, humidity, and pressure as String variables.

```
String readTemp() {
  return String(bme.readTemperature());
  //return String(1.8 * bme.readTemperature() + 32);
}

String readHumi() {
  return String(bme.readHumidity());
}

String readPres() {
  return String(bme.readPressure() / 100.0F);
}
```

In the `setup()`, initialize the Serial Monitor for demonstration purposes.

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

Set your ESP8266 as an access point with the SSID name and password defined earlier.

```
WiFi.softAP(ssid, password);
```

Then, handle the routes where the ESP8266 will be listening for incoming requests.

For example, when the ESP8266 server receives a request on the `/temperature` URL, it sends the temperature returned by the `readTemp()` function as a char (that's why we use the `c_str()` method.

```
server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request
  request->send_P(200, "text/plain", readTemp().c_str());
});
```

The same happens when the ESP receives a request on the `/humidity` and `/pressure` URLs.

```
server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", readHumi().c_str());
});
server.on("/pressure", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", readPres().c_str());
});
```

The following lines initialize the BME280 sensor.

```
bool status;

// default settings
// (you can also pass in a Wire library object like &Wire2)
status = bme.begin(0x76);
if (!status) {
  Serial.println("Could not find a valid BME280 sensor, check wiring!
```

```
  while (1);
}
```

Finally, start the server.

```
server.begin();
```

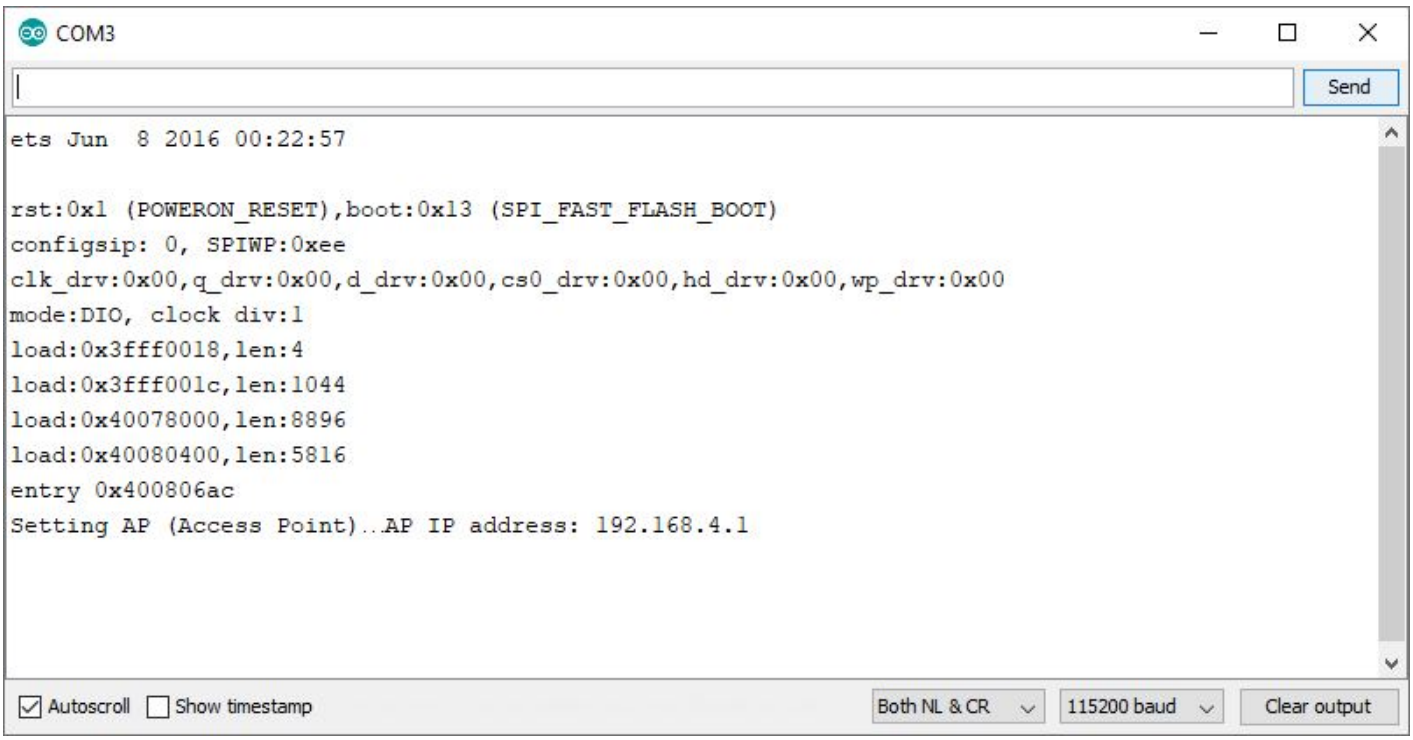
Because this is an asynchronous web server, there's nothing in the `loop()`.

```
void loop(){

}
```

Testing the ESP8266 Server

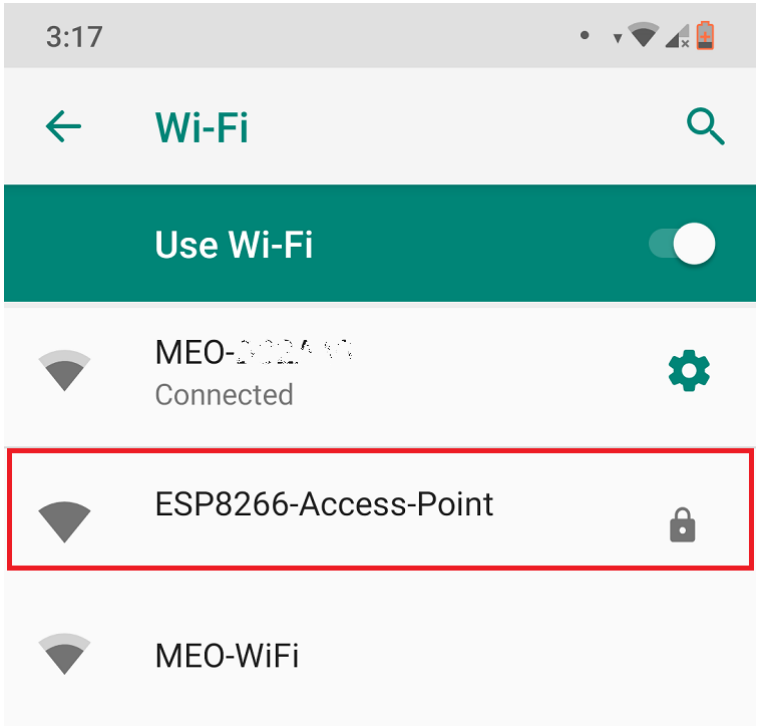
Upload the code to your board and open the Serial Monitor. You should get something as follows:



This means that the access point was set successfully.

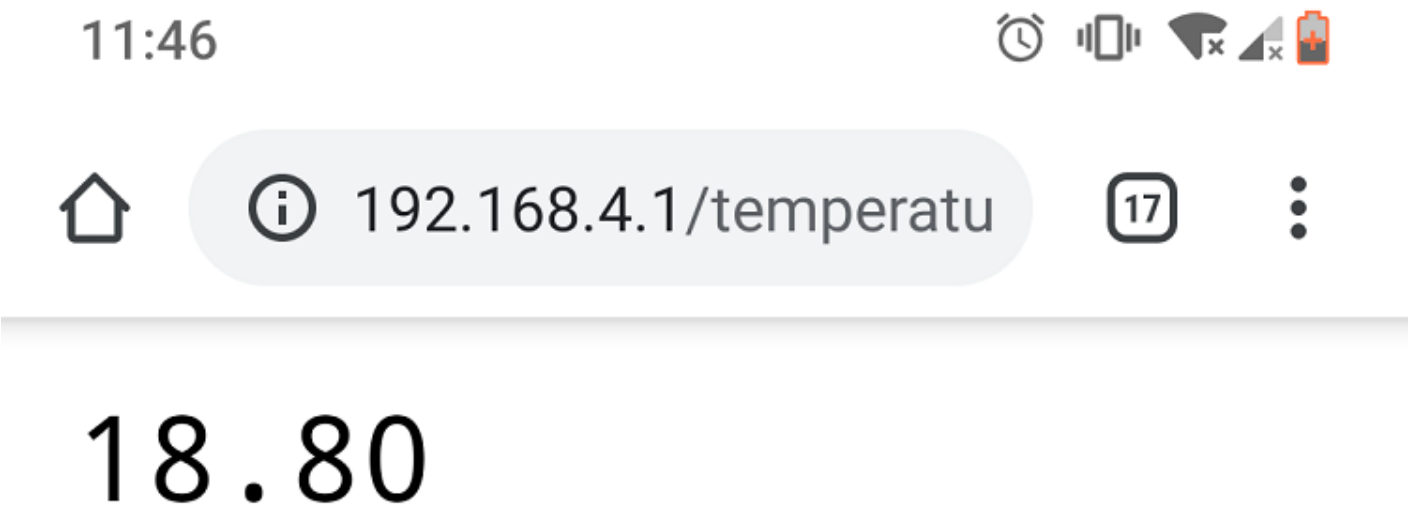
Now, to make sure it is listening for temperature, humidity and pressure requests, you need to connect to its network.

In your smartphone, go to the Wi-Fi settings and connect to the **ESP8266-Access-Point**. The password is **123456789**.

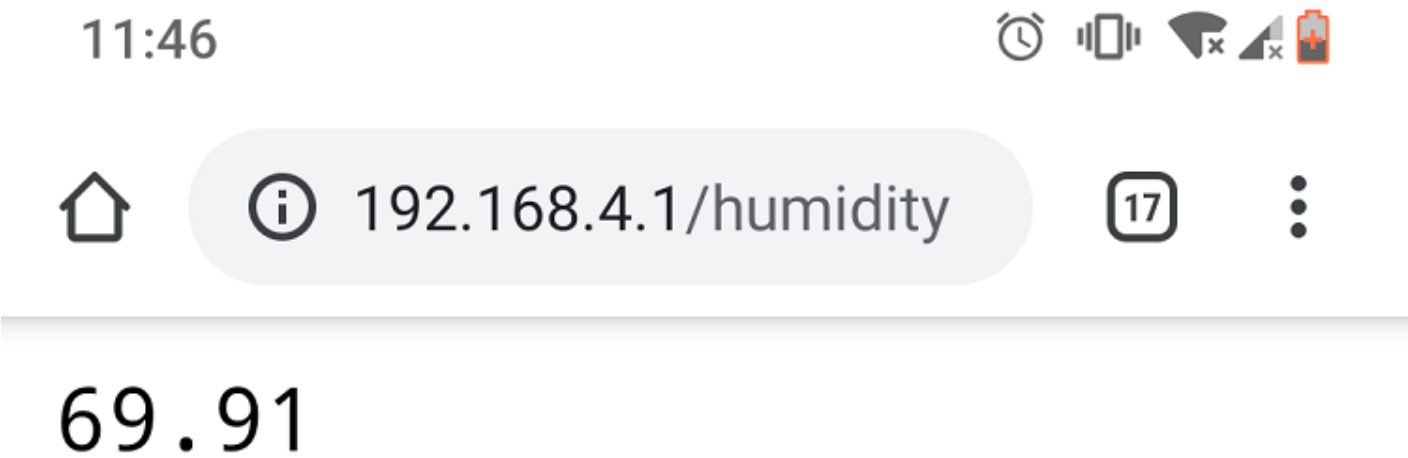


While connected to the access point, open your browser and type `192.168.4.1/temperature`

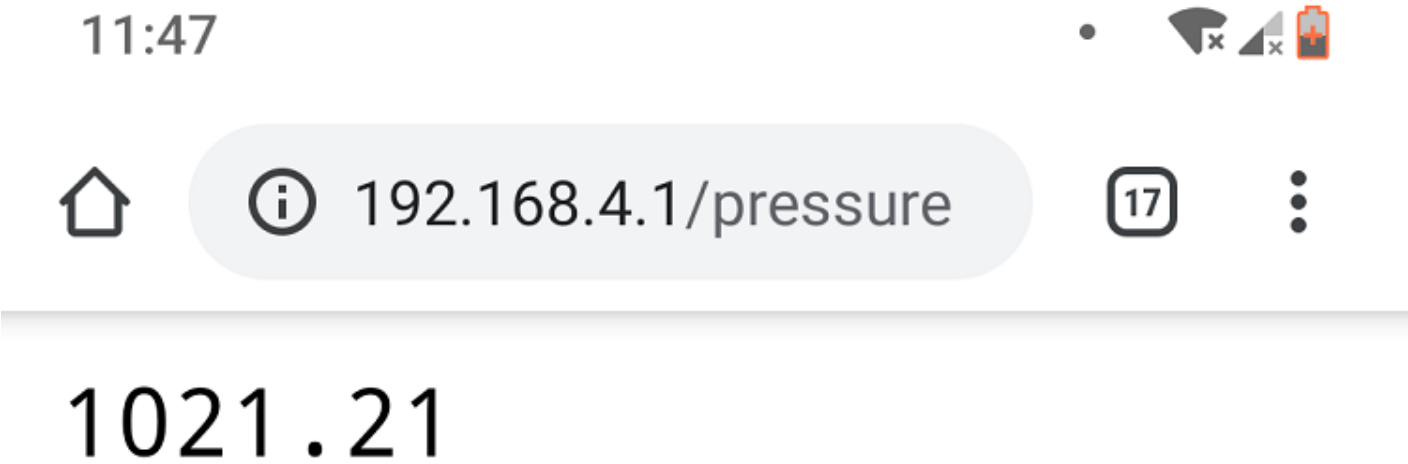
You should get the temperature value in your browser:



Try this URL path for the humidity `192.168.4.1/humidity`:



Finally, go to `192.168.4.1/pressure` URL:



If you're getting valid readings, it means that everything is working properly. Now, you need to prepare the other ESP8266 board (client) to make those requests for you and display them on the OLED display.