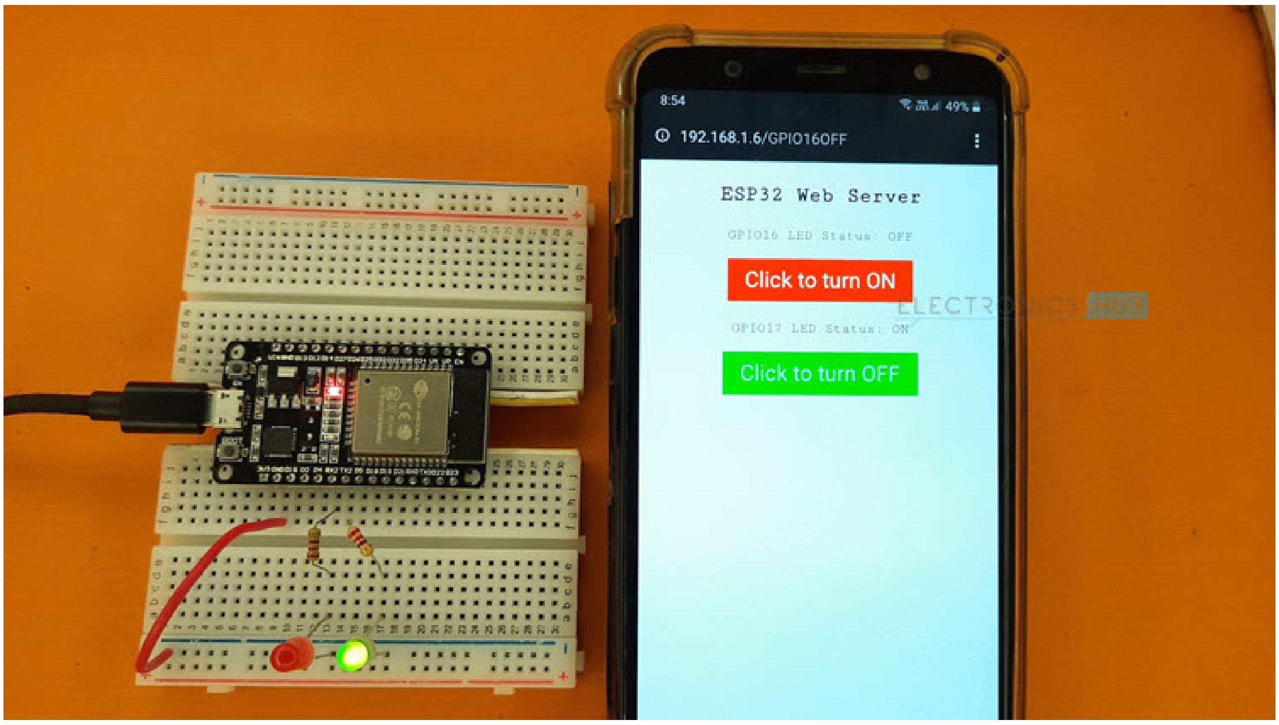
A Complete Beginner's Tutorial on How to Create ESP32 Web Server

electronicshub.org/esp32-web-server

In this tutorial, we will see how to build a simple ESP32 Web Server. This ESP32 Standalone Web Server can be accessed by mobile phone, computers, laptops and tablets which are connected to the same WiFi network as ESP32. As a demonstration of the web server, I created a simple web page which can be accesses by clients and control couple of LEDs connected to ESP32.

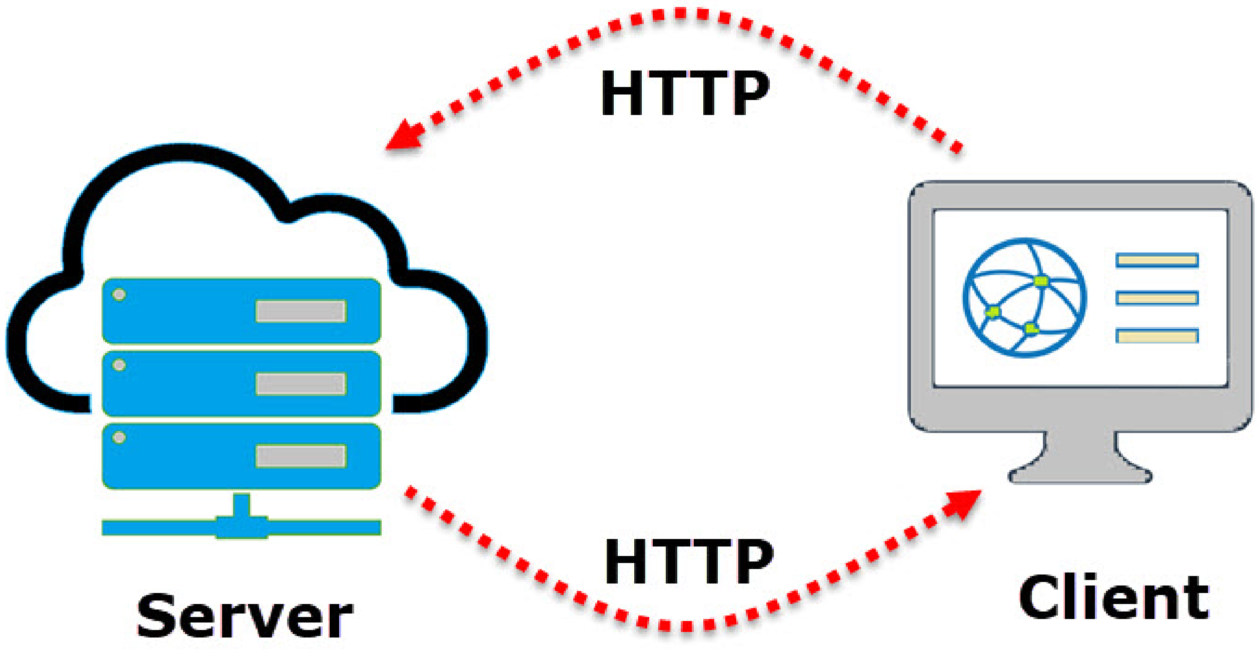


A Brief Note on Web Servers

I have already discussed about Web Servers in ESP8266 NodeMCU Web Server Project. But here is an overview once again.

A Web Server is combination of Hardware and Software which is responsible for maintaining, fetching and serving web pages to Web Clients. The information in the web pages can be in any format like Text in the form of HTML Documents, Images, Video, Applications etc.

Speaking of Web Clients, the Web Browsers in your laptops and mobile phones are one of the simplest types of web clients. The communication between a Web Server and a Web Client is often referred to as Client – Server Communication Model.

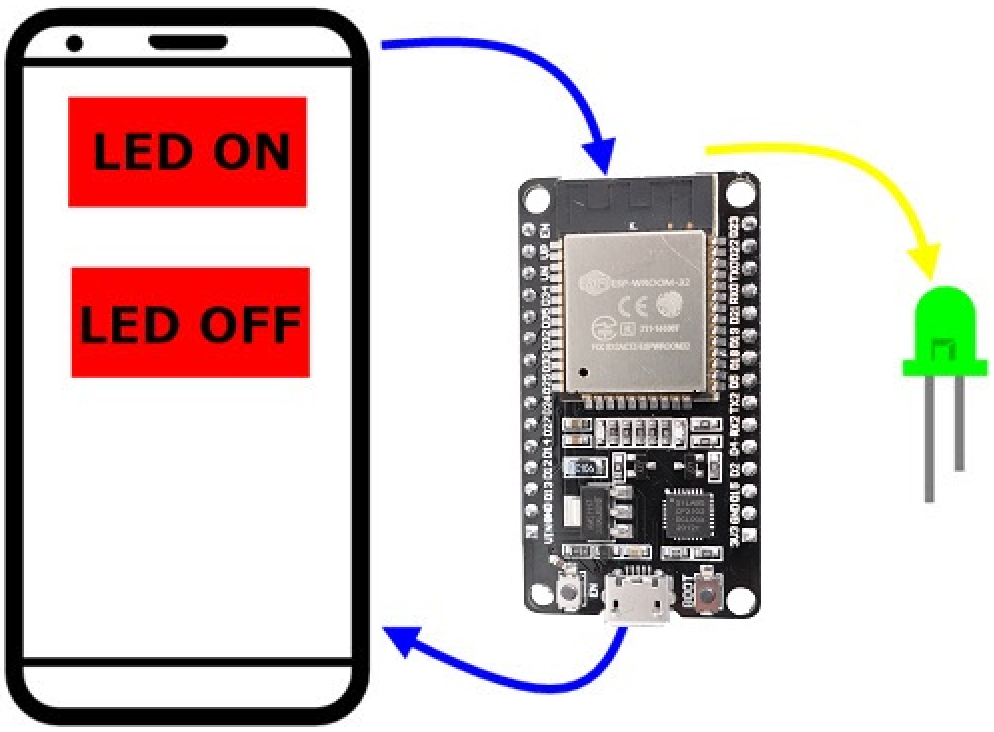


Hyper Text Transfer Protocol or simply HTTP is the protocol responsible for communication between Client and Server. In this type of communication, the Web Client makes a request for information from the server using HTTP. The Web Server, which is always waiting (listening) for a request, responds to the client's request with appropriate web page.

If the requested page is not found, then the server responds with HTTP 404 Error.

Requirements of ESP32 Web Server

With the brief introduction of Web Servers in general, we will now understand what are the requirement of a standalone ESP32 Web Server. A simple ESP32 Web Server must contain a web page in the form of HTML Text.



When a client, like a web browser in a mobile phone, sends a request for that web page over HTTP, the web server in ESP32 must respond with the web page. Additionally, when the client performs any operations, like clicking on a button, the server should respond with appropriate actions (like turning ON / OFF an LED).

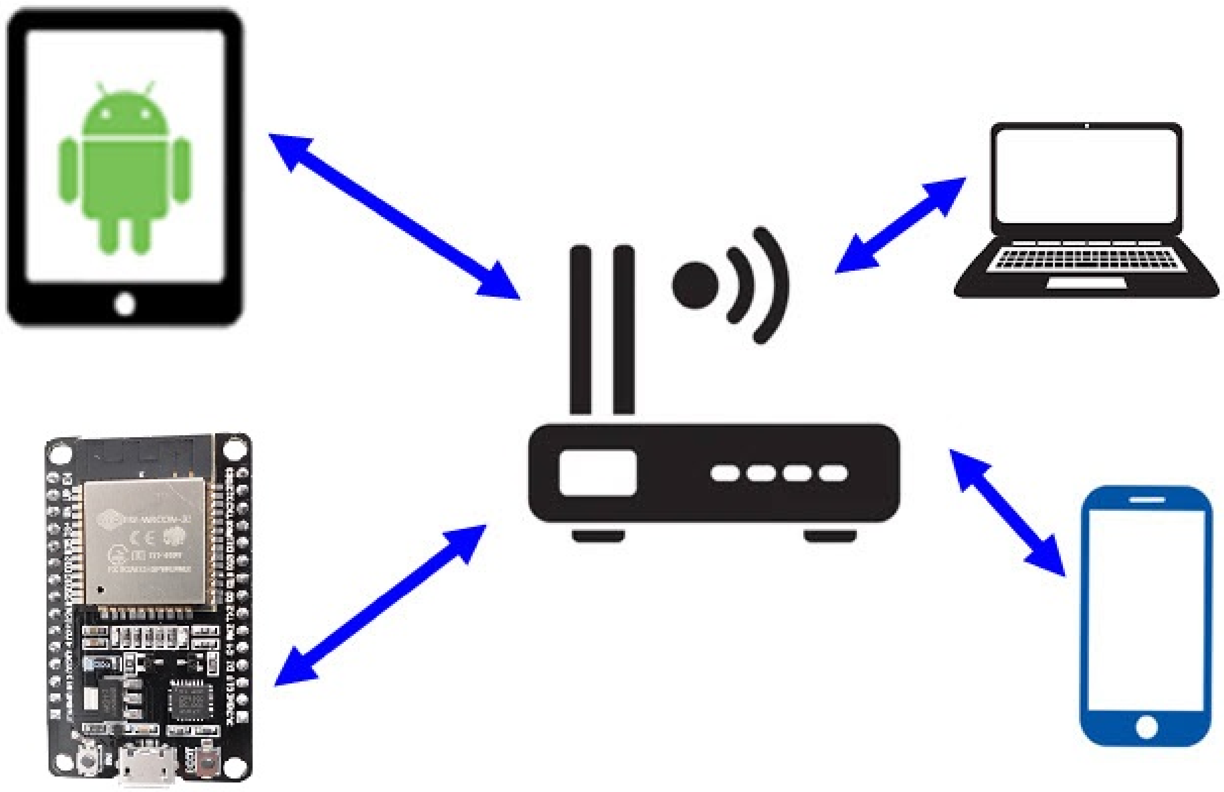
Wi-Fi Modes of Operation of ESP32

Before proceeding with creating a Web Server for ESP32, we will take a look at the different operating modes of Wi-Fi in ESP32. Basically, the ESP32 Wi-Fi Module operates in three WiFi operating modes. They are:

- Station Mode (STA)
- Soft Access Point Mode (AP)
- Station + Soft AP Mode

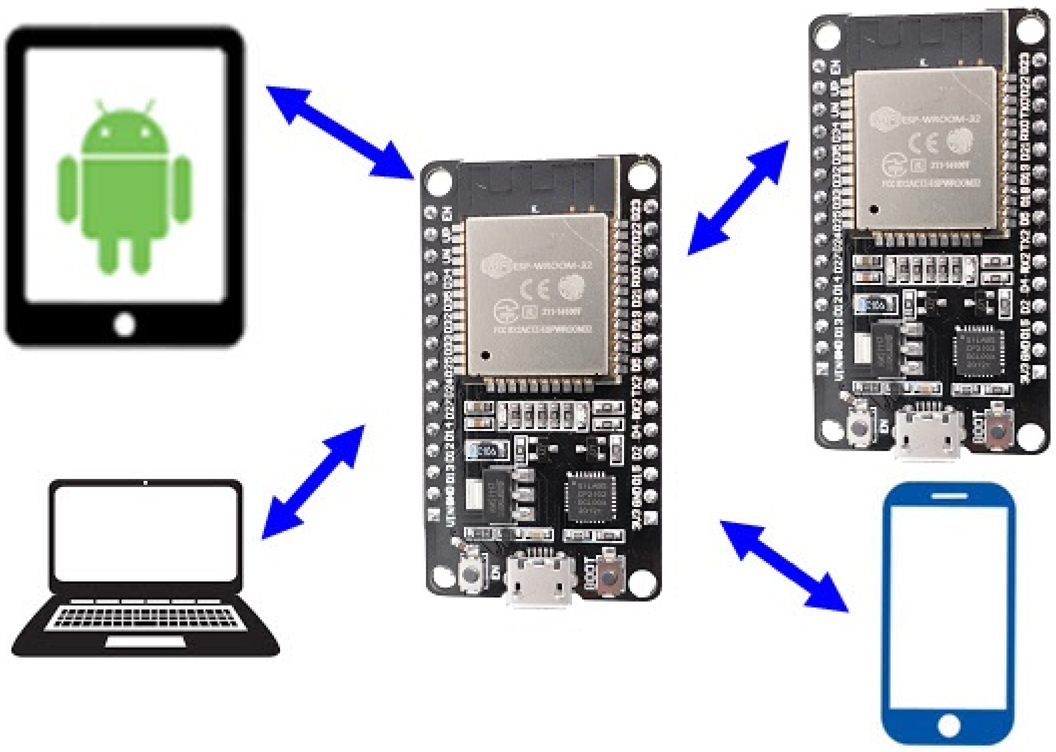
In station mode, the ESP32 Module connects to an existing WiFi Network, which is setup by a Wireless Router, just like our Mobile Phones and Laptops.

The ESP32 Wi-Fi Module connects to a Wi-Fi Network of Router using the router’s SSID and Password and the router assigns the local IP Address for ESP32.



Coming to Access Point Mode, the ESP32 Module creates its own WiFi Network like a Wireless Router, so that other stations like Mobile Phones, Laptops and even other ESP32 modules (in STA Mode) can connect to that network.

Since ESP32 doesn't have a Wired Ethernet connectivity to internet, this AP Mode is called as Soft AP Mode. While configuring ESP32 in AP Mode, you have to set the SSID and Password for the network, so that other devices can connect to that network using those credentials.



Station + Soft AP is a combination of Station Mode and Soft AP Mode. In this, the ESP32 acts as both the Station as well as an Access Point.

Which Mode to use for Creating Web Server?

You can configure ESP32 Wi-Fi Module either in Station Mode or in Access point Mode to create a web server. The difference is that in station mode, all the devices (Mobiles, laptops, ESP32, ESP8266, etc.) are connected to Wireless Router's WiFi Network and IP Address to all the devices (including the Web Server of ESP32) is assigned by the router.

Using this IP Address, clients can access the Web Page. Additionally, the clients do not lose internet connectivity from the Router.

But if we create Web Server for ESP32 in AP Mode, then clients must connect to the network provided by ESP32 using its own SSID and Password in order to access the Web Pages. Since it is a soft AP Mode, clients do not have internet connectivity.

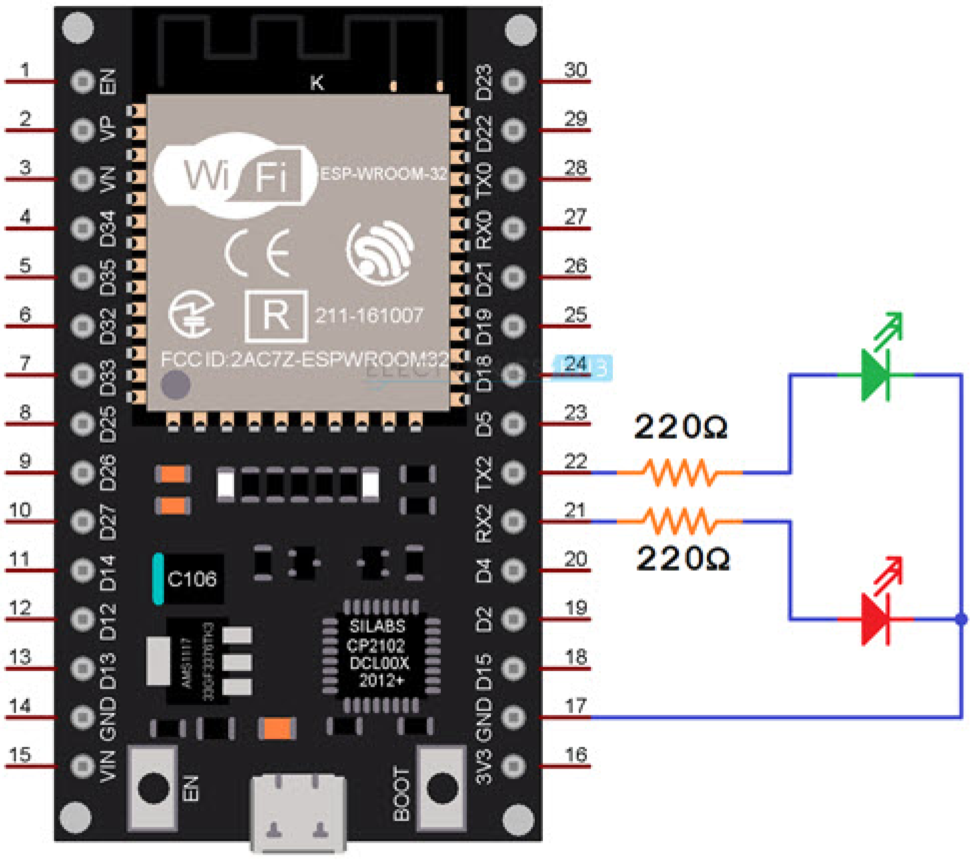
Creating ESP32 Web Server either in Station Mode or in Soft AP Mode is very similar except the configuration part of the ESP32.

In this tutorial, I will show you how to create a Web Server on ESP32 configured in Station Mode (STA).

ESP32 Web Server

Apart from creating the web server on ESP32 and accessing it on clients, we will also see how this web server responds to different requests for clients by controlling two LEDs connected to GPIO Pins of ESP32 Development Board.

To demonstrate this, I connected two 5mm LEDs to GPIO 16 and GPIO 17 of ESP32 through respective current limiting resistors (220Ω). GPIO 16 is labelled RX2 and GPIO 17 is labelled TX2 on ESP32 DevKit Development Board.



Code

Coming to the important and interesting stuff, the actual code for Web Server on ESP32. It is just an HTML Code with some text, couple of buttons and some stylization.

The following block shows the complete code for ESP32's Web Server. I commented the code in order to explain the code.

NOTE: This code is based on '[Arduino Web Server]' example.

```
#include <WiFi.h>

#define gpio16LEDPin 16 /* One LED connected to GPIO16 - RX2 */
#define gpio17LEDPin 17 /* One LED connected to GPIO17 - TX2 */

const char* ssid = "ESP32-WiFi"; /* Add your router's SSID */
const char* password = "12345678"; /*Add the password */

int gpio16Value;
int gpio17Value;

WiFiServer espServer(80); /* Instance of WiFiServer with port number 80 */
```

```
/* 80 is the Port Number for HTTP Web Server */

/* A String to capture the incoming HTTP GET Request */
String request;

void setup()
{
Serial.begin(115200); /* Begin Serial Communication with 115200 Baud Rate */
/* Configure GPIO16 and GPIO17 Pins as OUTPUTs */
pinMode(gpio16LEDPin, OUTPUT);
pinMode(gpio17LEDPin, OUTPUT);
/* Set the initial values of GPIO16 and GPIO17 as LOW*/
/* Both the LEDs are initially OFF */
digitalWrite(gpio16LEDPin, LOW);
digitalWrite(gpio17LEDPin, LOW);

Serial.print("\n");
Serial.print("Connecting to: ");
Serial.println(ssid);
WiFi.mode(WIFI_STA); /* Configure ESP32 in STA Mode */
WiFi.begin(ssid, password); /* Connect to Wi-Fi based on the above SSID and
Password */
while(WiFi.status() != WL_CONNECTED)
{
Serial.print("*");
delay(100);
}
Serial.print("\n");
Serial.print("Connected to Wi-Fi: ");
Serial.println(WiFi.SSID());
```

```
delay(100);
/* The next four lines of Code are used for assigning Static IP to ESP32 */
/* Do this only if you know what you are doing */
/* You have to check for free IP Addresses from your Router and */
/* assign it to ESP32 */
/* If you are comfortable with this step, */
/* please un-comment the next four lines and make necessary changes */
/* If not, leave it as it is and proceed */
//IPAddress ip(192,168,1,6);
//IPAddress gateway(192,168,1,1);
//IPAddress subnet(255,255,255,0);
//WiFi.config(ip, gateway, subnet);
delay(2000);
Serial.print("\n");
Serial.println("Starting ESP32 Web Server...");
espServer.begin(); /* Start the HTTP web Server */
Serial.println("ESP32 Web Server Started");
Serial.print("\n");
Serial.print("The URL of ESP32 Web Server is: ");
Serial.print("http://");
Serial.println(WiFi.localIP());
Serial.print("\n");
Serial.println("Use the above URL in your Browser to access ESP32 Web Server\n");
}

void loop()
{
WiFiClient client = espServer.available(); /* Check if a client is available */
if(!client)
```

```
{
return;
}

Serial.println("New Client!!!");
boolean currentLineIsBlank = true;
while (client.connected())
{
if (client.available())
{
char c = client.read();
request += c;
Serial.write(c);
/* if you've gotten to the end of the line (received a newline */
/* character) and the line is blank, the http request has ended, */
/* so you can send a reply */
if (c == '\n' && currentLineIsBlank)
{
/* Extract the URL of the request */
/* We have four URLs. If IP Address is 192.168.1.6 (for example),
* then URLs are:
* 192.168.1.6/GPIO16ON
* 192.168.1.6/GPIO16OFF
* 192.168.1.6/GPIO17ON
* 192.168.1.6/GPIO17OFF
*/
/* Based on the URL from the request, turn the LEDs ON or OFF */
if (request.indexOf("/GPIO16ON") != -1)
{
```

```
Serial.println("GPIO16 LED is ON");
digitalWrite(gpio16LEDPin, HIGH);
gpio16Value = HIGH;
}
if (request.indexOf("/GPIO16OFF") != -1)
{
Serial.println("GPIO16 LED is OFF");
digitalWrite(gpio16LEDPin, LOW);
gpio16Value = LOW;
}
if (request.indexOf("/GPIO17ON") != -1)
{
Serial.println("GPIO17 LED is ON");
digitalWrite(gpio17LEDPin, HIGH);
gpio17Value = HIGH;
}
if (request.indexOf("/GPIO17OFF") != -1)
{
Serial.println("GPIO17 LED is OFF");
digitalWrite(gpio17LEDPin, LOW);
gpio17Value = LOW;
}

/* HTTP Response in the form of HTML Web Page */
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connection: close");
client.println(); // IMPORTANT

client.println("<!DOCTYPE HTML>");
client.println("<html>");

client.println("<head>");
client.println("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("<style>");

client.println("html { font-family: Courier New; display: inline-block; margin: 0px auto; text-align: center;}");
client.println(".button {border: none; color: white; padding: 10px 20px; text-align: center;");
client.println("text-decoration: none; font-size: 25px; margin: 2px; cursor: pointer;}");
client.println(".button1 {background-color: #FF0000;}");
client.println(".button2 {background-color: #00FF00;}");

client.println("</style>");

client.println("</head>");

client.println("<body>");

client.println("<h2>ESP32 Web Server</h2>");

if(gpio16Value == LOW)
{
client.println("<p>GPIO16 LED Status: OFF</p>");
client.print("<p><a href=\"/GPIO16ON\"><button class=\"button button1\">Click to turn ON</button></a></p>");
}
else
{
```

```
client.println("<p>GPIO16 LED Status: ON</p>");
client.print("<p><a href=\"/GPIO16OFF\"><button class=\"button button2\">Click to
turn OFF</button></a></p>");
}

if(gpio17Value == LOW)
{
client.println("<p>GPIO17 LED Status: OFF</p>");
client.print("<p><a href=\"/GPIO17ON\"><button class=\"button button1\">Click to
turn ON</button></a></p>");
}
else
{
client.println("<p>GPIO17 LED Status: ON</p>");
client.print("<p><a href=\"/GPIO17OFF\"><button class=\"button button2\">Click to
turn OFF</button></a></p>");
}

client.println("</body>");

client.println("</html>");

break;

}
if(c == '\n')
{
currentLineIsBlank = true;
}
else if(c != '\r')
{
currentLineIsBlank = false;
```

```
    }
    //client.print("\n");

  }
  }

  delay(1);
  request = "";
  //client.flush();
  client.stop();
  Serial.println("Client disconnected");
  Serial.print("\n");
}
```

view raw ESP32-Web-Server.ino hosted with ❤ by GitHub

Modify and Upload Code

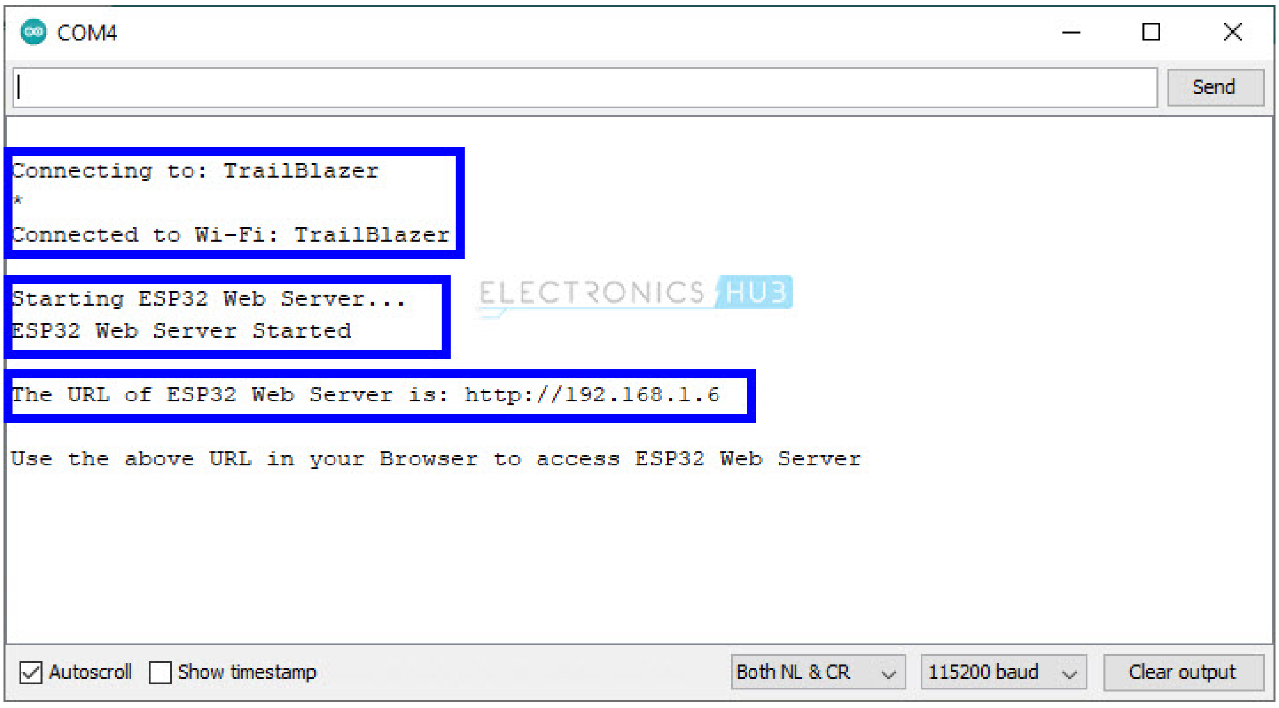
In lines 6 and 7 of above code, you have to make the modifications as per your Wi-Fi Network Settings. These are the SSID and Password of the Wi-Fi Network.

```
const char* ssid = "ESP32-WiFi"; /* Add your router's SSID */
const char* password = "12345678"; /*Add the password */
```

After making necessary modifications, make the necessary connections as per the circuit diagram, connect the ESP32 Board to the computer, select the ESP32 Board and also the right COM Port and upload the code.

If you are new to ESP32, then the Getting Started with ESP32 tutorial will help you in configuring Arduino IDE.

Open the Serial Monitor and ESP32 Module will print some important information like the progress of Wi-Fi Connection, IP Address and URL of Web Server (which is essentially the IP Address of the ESP32).



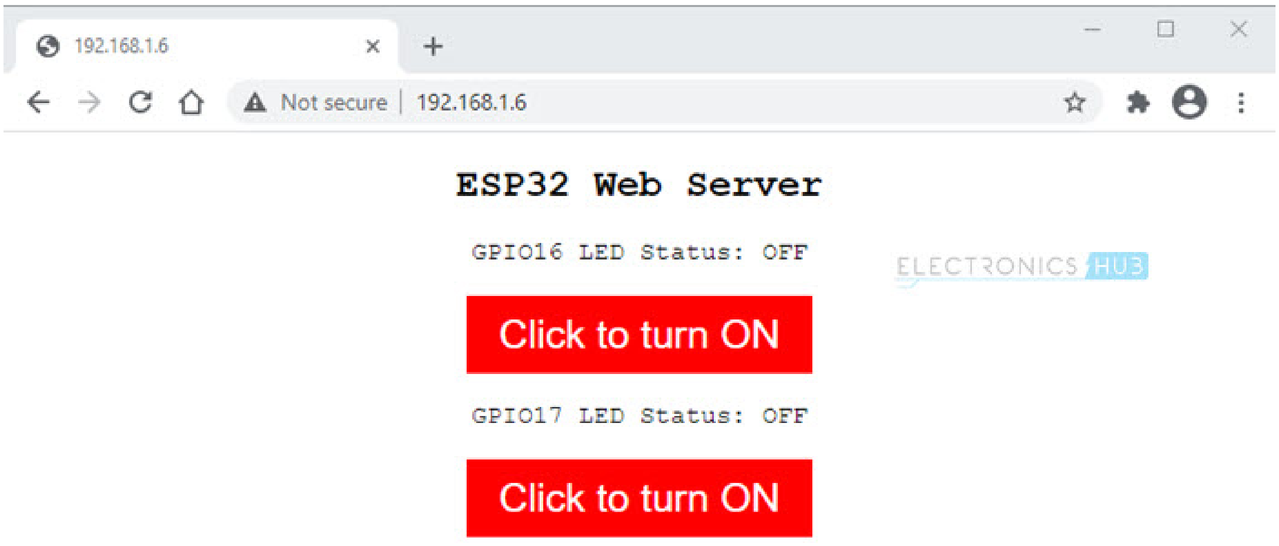
```
Connecting to: TrailBlazer
*
Connected to Wi-Fi: TrailBlazer

Starting ESP32 Web Server...
ESP32 Web Server Started

The URL of ESP32 Web Server is: http://192.168.1.6

Use the above URL in your Browser to access ESP32 Web Server
```

So, in my case the IP Address of ESP32 is 192.168.1.6.

Accessing the ESP32 Web Server from Clients

Open a Web Browser either in a laptop or mobile phone and type the IP Address of ESP32. This is the moment of truth. If everything goes well, then you should be able to see a simple web page hosted by the ESP32's Web Server.

The following is a screenshot of Chrome Web Browser on a laptop accessing the Web Server of ESP32.



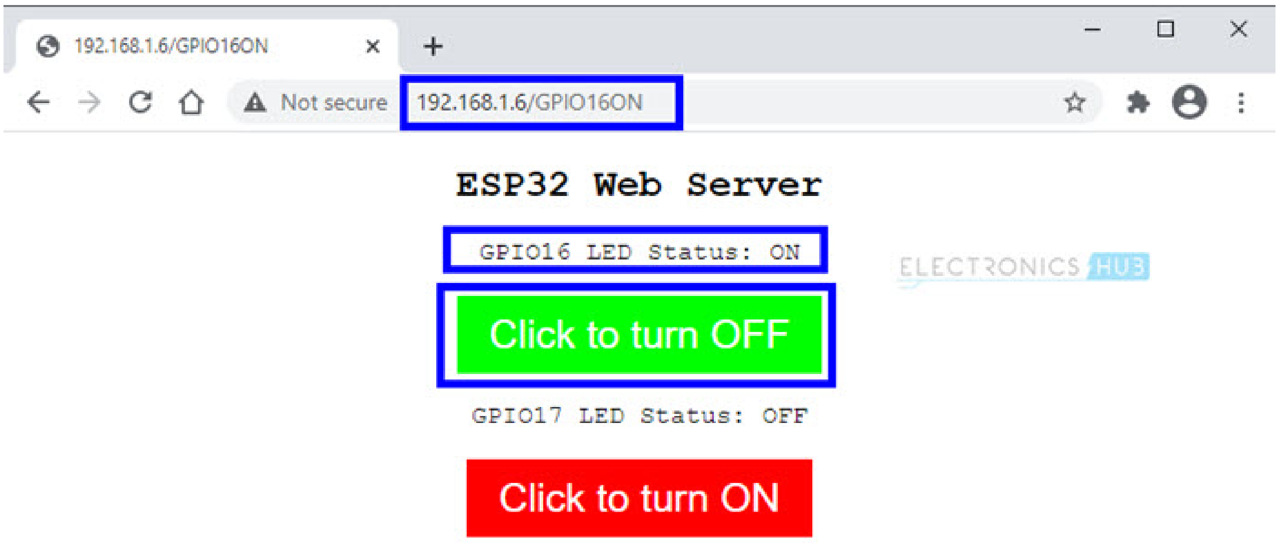
ESP32 Web Server

GPIO16 LED Status: OFF

GPIO17 LED Status: OFF

As you can see from the image, the web page displays a main header text, followed by status of the LED connected to GPIO 16. This is followed by a Button, which can be used to turn ON or OFF the LED. The same stuff again for GPIO 17 (status followed by Button).

Now, if I click on the first button, the LED connected to GPIO 16 will turn ON, the status is updated in the web page, the text and color of the Button is also changed.



ESP32 Web Server

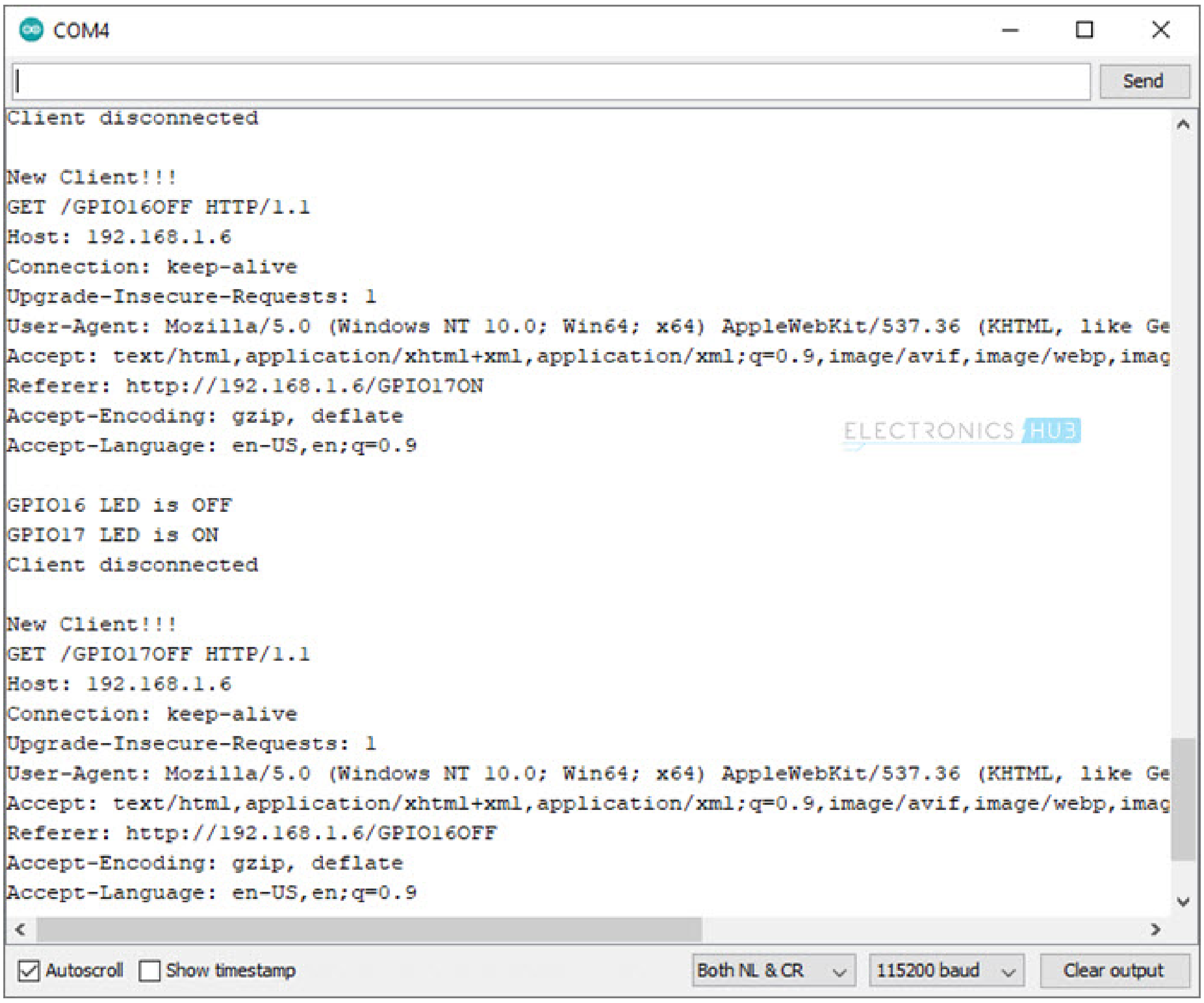
GPIO16 LED Status: ON

Click to turn OFF

GPIO17 LED Status: OFF

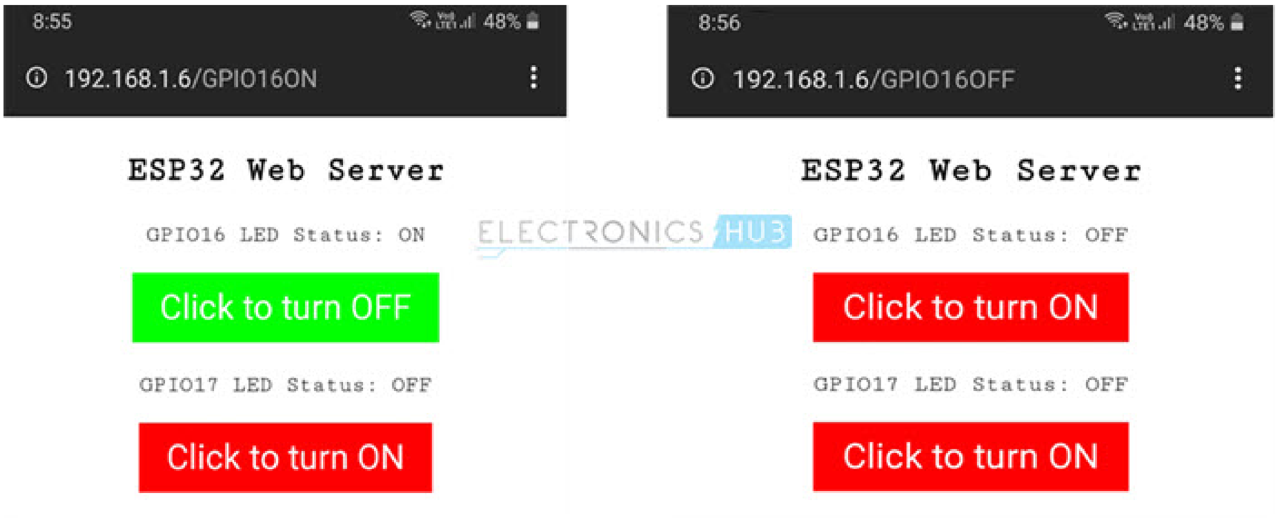
Click to turn ON

If you take a look at the Serial Monitor, every time a client tries to connect (or send a request), some key information is printed on the serial monitor. I will explain about this information (this is actually a part of request from client) in the next section.



```
Client disconnected

New Client!!!
GET /GPIO16OFF HTTP/1.1
Host: 192.168.1.6
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
Referer: http://192.168.1.6/GPIO17ON
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

GPIO16 LED is OFF
GPIO17 LED is ON
Client disconnected

New Client!!!
GET /GPIO17OFF HTTP/1.1
Host: 192.168.1.6
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
Referer: http://192.168.1.6/GPIO16OFF
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Next, I tried the same thing on a Mobile Phone. It works perfectly.



NOTE: All the clients i.e., mobiles, laptops, etc., must be connected to the same network as the ESP32 Module.

Conclusion

A complete step-by-step tutorial on how to create a Web Server using ESP32 DevKit Development Board. You learned some important basics about web servers, different modes of operation of ESP32 WiFi, how to build ESP32 Web Server and how to access this server from different clients.

Related Posts:

- In-depth tutorial on ESP32 Servo Control | Web...
- How to use BLE in ESP32? ESP32 BLE (Bluetooth Low...
- How to Create ESP8266 Web Server | Complete...
- ESP32 DS18B20 Tutorial | DS18B20 Temperature Sensor...
- ESP32 BMP180 Sensor Tutorial | How to Interface...
- ESP32 DHT11 Tutorial | DHT11 Humidity Temperature...

3 Responses

Leave a Reply

Your email address will not be published. Required fields are marked *