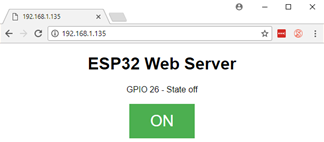
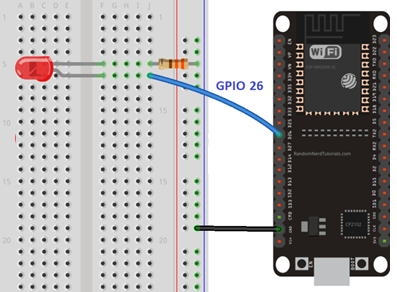
BOTON





#include <WiFi.h>

const char\* ssid = "CLARO\_608A";

const char\* password = "d2dACCDF";

WiFiServer server(80);

String header;

String ledState = "off";

const int led = 23; // GPIO 23

void setup() {

Serial.begin(115200);

pinMode(led, OUTPUT);

digitalWrite(led, LOW);

WiFi.begin(ssid, password);

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

delay(500);

}

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

WiFiClient client = server.available();

if (client) { // If a new client connects,

String currentLine = ""; // make a String to hold incoming data

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.write(c);

header += c;

if (c == '\n') {

if (currentLine.length() == 0) {

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

if (header.indexOf("GET /led/on") >= 0) {

ledState = "on";

digitalWrite(led, HIGH);

Serial.println("led on");

}

if (header.indexOf("GET /led/off") >= 0)

{

ledState = "off";

digitalWrite(led, LOW);

Serial.println("led off");

}

// Display the HTML web page

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

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");

// Display current led state

client.println("<p>State " + ledState + "</p>");

if (ledState=="off")

client.println("<p><a href=\"/led/on\"><button class=\"button\">ON</button></a></p>");

if (ledState=="on")

client.println("<p><a href=\"/led/off\"><button class=\"button button2\">OFF</button></a></p>");

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

client.println();

break;

} else {

currentLine = "";

}

} else if (c != '\r') {

currentLine += c;

}

}

}

header = "";

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

}

CON IP FIJA

#include <WiFi.h>

const char\* ssid = "CLARO\_60FA";

const char\* password = "d2dACF";

IPAddress local\_IP(192, 168, 1, 184);

IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 0, 0);

IPAddress primaryDNS(8, 8, 8, 8); //optional

IPAddress secondaryDNS(8, 8, 4, 4); //optional

WiFiServer server(80);

String header;

String ledState = "off";

const int led = 23;

void setup() {

Serial.begin(115200);

pinMode(led, OUTPUT);

digitalWrite(led, LOW);

Serial.print("Connecting");

// Configures static IP address

if (!WiFi.config(local\_IP, gateway, subnet, primaryDNS, secondaryDNS)) {

Serial.println("STA Failed to configure");

}

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.write(c);

header += c;

if (c == '\n') {

if (currentLine.length() == 0) {

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

if (header.indexOf("GET /led/on") >= 0) {

ledState = "on";

digitalWrite(led, HIGH);

Serial.println("led on");

}

if (header.indexOf("GET /led/off") >= 0)

{

ledState = "off";

digitalWrite(led, LOW);

Serial.println("led off");

}

// Display the HTML web page

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

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");

// Display current led state

client.println("<p>State " + ledState + "</p>");

if (ledState=="off")

client.println("<p><a href=\"/led/on\"><button class=\"button\">ON</button></a></p>");

if (ledState=="on")

client.println("<p><a href=\"/led/off\"><button class=\"button button2\">OFF</button></a></p>");

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

client.println();

break;

} else {

currentLine = "";

}

} else if (c != '\r') {

currentLine += c;

}

}

}

header = "";

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

}

//= = = = =

Incluyee Bluettoth para conocer la IP

#include <WiFi.h>

#include "BluetoothSerial.h"

BluetoothSerial BT;

const char\* ssid = "CLARO\_608FFA";

const char\* password = "d2dACCD34F";

WiFiServer server(80);

String header;

String ledState = "off";

const int led = 23;

void setup() {

Serial.begin(115200);

pinMode(led, OUTPUT);

digitalWrite(led, LOW);

BT.begin("ESP32"); //Name of your Bluetooth to pair

Serial.print("Connecting");

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

//Check if we receive anything from Bluetooth

if (BT.available())   
  {

int income = BT.read();

Serial.println(income); // 0-> 48 1-> 49

if (income ==49) BT.println(WiFi.localIP());

}

//WIFI

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.write(c);

header += c;

if (c == '\n') {

if (currentLine.length() == 0) {

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

if (header.indexOf("GET /led/on") >= 0) {

ledState = "on";

digitalWrite(led, HIGH);

Serial.println("led on");

}

if (header.indexOf("GET /led/off") >= 0)

{

ledState = "off";

digitalWrite(led, LOW);

Serial.println("led off");

}

// Display the HTML web page

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

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");

// Display current led state

client.println("<p>State " + ledState + "</p>");

if (ledState=="off")

client.println("<p><a href=\"/led/on\"><button class=\"button\">ON</button></a></p>");

if (ledState=="on")

client.println("<p><a href=\"/led/off\"><button class=\"button button2\">OFF</button></a></p>");

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

client.println();

break;

} else {

currentLine = "";

}

} else if (c != '\r') {

currentLine += c;

}

}

}

header = "";

client.stop();

Serial.println("Client disconnected.");

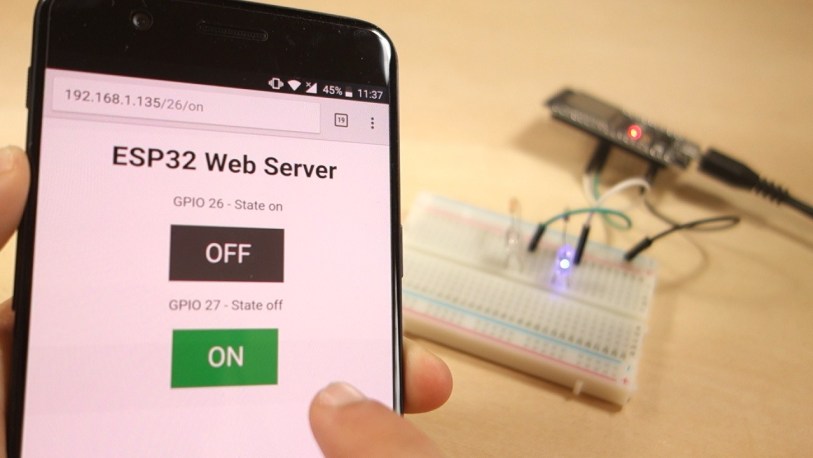
Serial.println("");

}

}

**ESP32 Web Server – Arduino IDE**

In this project you’ll create a standalone web server with an ESP32 that controls outputs (two LEDs) using the Arduino IDE programming environment. The web server is mobile responsive and can be accessed with any device that as a browser on the local network.



If you want to learn more about the ESP32, read [Getting Started Guide with ESP32](https://randomnerdtutorials.com/getting-started-with-esp32/).

Before going straight to the project, it is important to outline what our web server will do, so that it is easier to follow the steps later on.

* The web server you’ll build controls two LEDs connected to the ESP32 GPIO 26 and GPIO 27;
* You can access the ESP32 web server by typing the ESP32 IP address on a browser in the local network;
* By clicking the buttons on your web server you can instantly change the state of each LED.

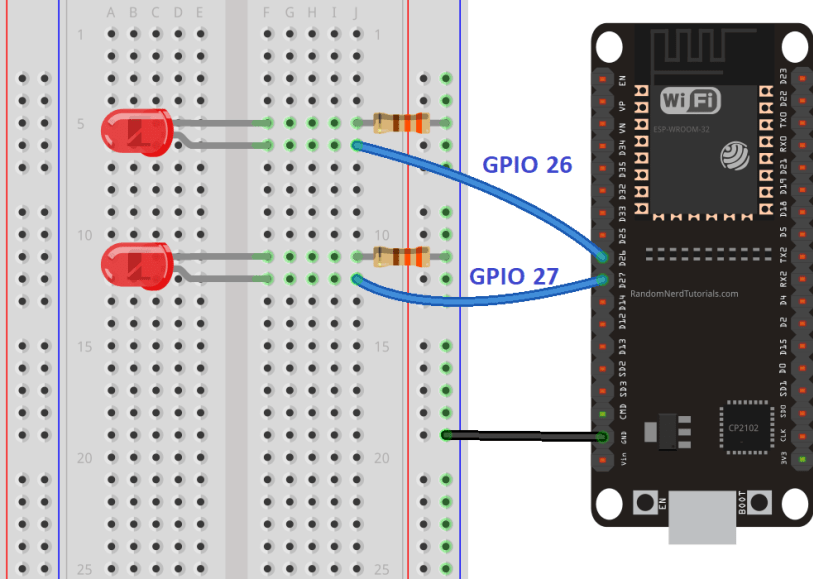
This is just a simple example to illustrate how to build a web server that controls outputs, the idea is to replace those LEDs with a [relay](https://makeradvisor.com/tools/5v-2-channel-relay-module-optocoupler/), or any other electronic components you want.

**Installing the ESP32 board in Arduino IDE**

There’s an add-on for the Arduino IDE that allows you to program the ESP32 using the Arduino IDE and its programming language. Follow one of the following tutorials to prepare your Arduino IDE:

* [**Windows instructions** – Installing the ESP32 Board in Arduino IDE](https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/)
* [**Mac and Linux instructions** – Installing the ESP32 Board in Arduino IDE](https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-mac-and-linux-instructions/)

For this tutorial you’ll need the following parts:



**ESP32 Web Server Code**

// Load Wi-Fi library

#include <WiFi.h>

// Replace with your network credentials

const char\* ssid = "";

const char\* password = "";

// Set web server port number to 80

WiFiServer server(80);

// Variable to store the HTTP request

String header;

// Auxiliar variables to store the current output state

String output26State = "off";

String output27State = "off";

// Assign output variables to GPIO pins

const int output26 = 26;

const int output27 = 27;

void setup() {

Serial.begin(115200);

// Initialize the output variables as outputs

pinMode(output26, OUTPUT);

pinMode(output27, OUTPUT);

// Set outputs to LOW

digitalWrite(output26, LOW);

digitalWrite(output27, LOW);

// Connect to Wi-Fi network with SSID and password

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

// Print local IP address and start web server

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data from the client

while (client.connected()) { // loop while the client's connected

if (client.available()) { // if there's bytes to read from the client,

char c = client.read(); // read a byte, then

Serial.write(c); // print it out the serial monitor

header += c;

if (c == '\n') { // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

// turns the GPIOs on and off

if (header.indexOf("GET /26/on") >= 0) {

Serial.println("GPIO 26 on");

output26State = "on";

digitalWrite(output26, HIGH);

} else if (header.indexOf("GET /26/off") >= 0) {

Serial.println("GPIO 26 off");

output26State = "off";

digitalWrite(output26, LOW);

} else if (header.indexOf("GET /27/on") >= 0) {

Serial.println("GPIO 27 on");

output27State = "on";

digitalWrite(output27, HIGH);

} else if (header.indexOf("GET /27/off") >= 0) {

Serial.println("GPIO 27 off");

output27State = "off";

digitalWrite(output27, LOW);

}

// Display the HTML web page

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

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

// CSS to style the on/off buttons

// Feel free to change the background-color and font-size attributes to fit your preferences

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");

// Display current state, and ON/OFF buttons for GPIO 26

client.println("<p>GPIO 26 - State " + output26State + "</p>");

// If the output26State is off, it displays the ON button

if (output26State=="off") {

client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/26/off\"><button class=\"button button2\">OFF</button></a></p>");

}

// Display current state, and ON/OFF buttons for GPIO 27

client.println("<p>GPIO 27 - State " + output27State + "</p>");

// If the output27State is off, it displays the ON button

if (output27State=="off") {

client.println("<p><a href=\"/27/on\"><button class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/27/off\"><button class=\"button button2\">OFF</button></a></p>");

}

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

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

break;

} else { // if you got a newline, then clear currentLine

currentLine = "";

}

} else if (c != '\r') { // if you got anything else but a carriage return character,

currentLine += c; // add it to the end of the currentLine

}

}

}

// Clear the header variable

header = "";

// Close the connection

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

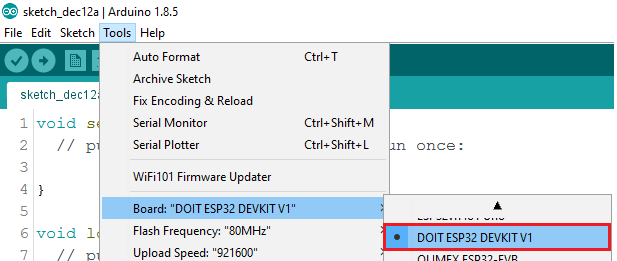
}

**Uploading the Code**

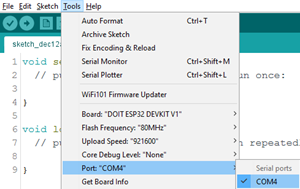
Now, you can upload the code and and the web server will work straight away. Follow the next steps to upload code to the ESP32:

**1)** Plug your ESP32 board in your computer;

**2)** In the Arduino IDE select your board in **Tools** > **Board** (in our case we’re using the ESP32 DEVKIT DOIT board);



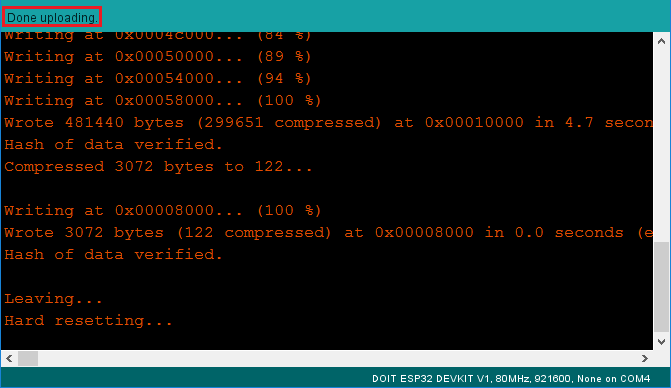
3) Select the COM port in **Tools** > **Port**.



4) Press the **Upload** button in the Arduino IDE

https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2016/12/arduino-ide-upload-button.png?resize=34%2C29&ssl=1

5) Wait for the “**Done uploading**” message.



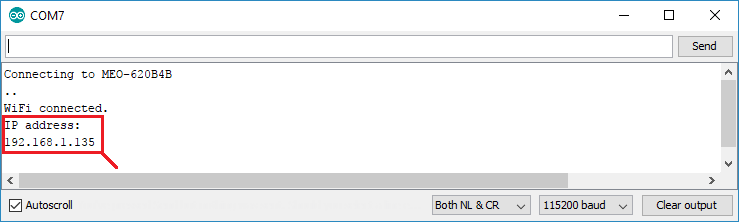
After uploading the code, open the Serial Monitor at a baud rate of 115200.



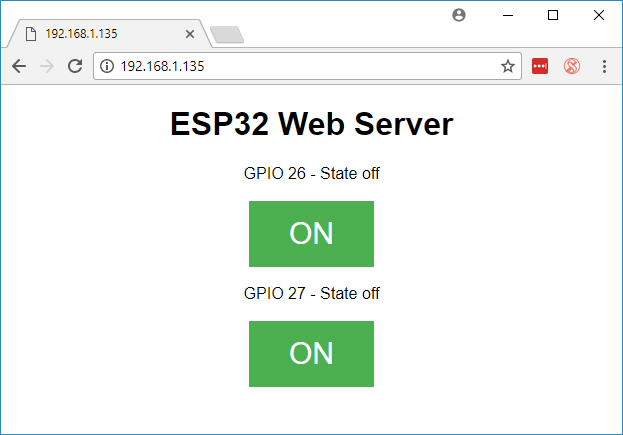
Your ESP8266 should have the new sketch running.

Press the “**ENABLE/RESET**” button to restart the ESP8266 and run the new uploaded sketch.

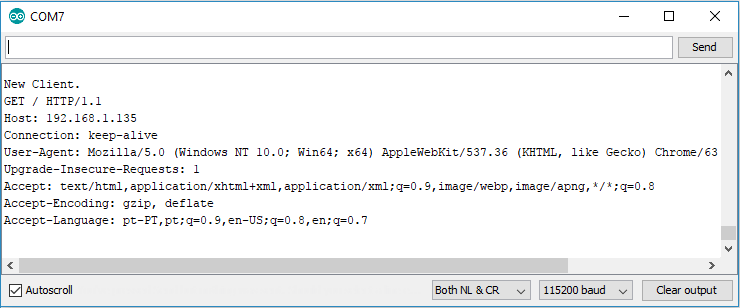
The ESP32 connects to Wi-Fi, and outputs the ESP IP address on the Serial Monitor. Copy that IP address, because you need it to access the ESP32 web server.



To access the web server, open your browser, paste the ESP32 IP address, and you’ll see the following page. In our case it is **192.168.1.135**.

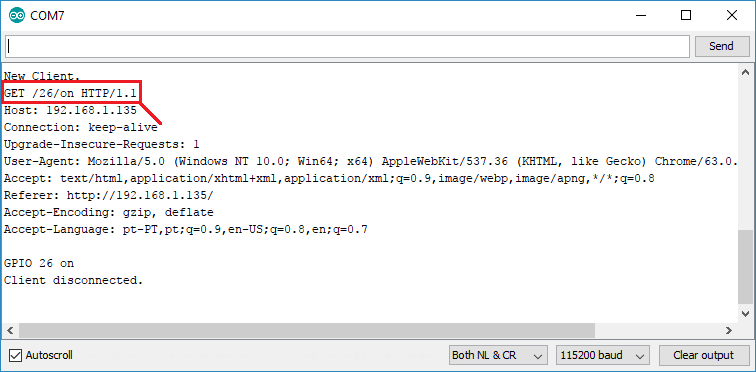


If you take a look at the Serial Monitor, you can see what’s happening on the background. The ESP receives an HTTP request from a new client (in this case, your browser).

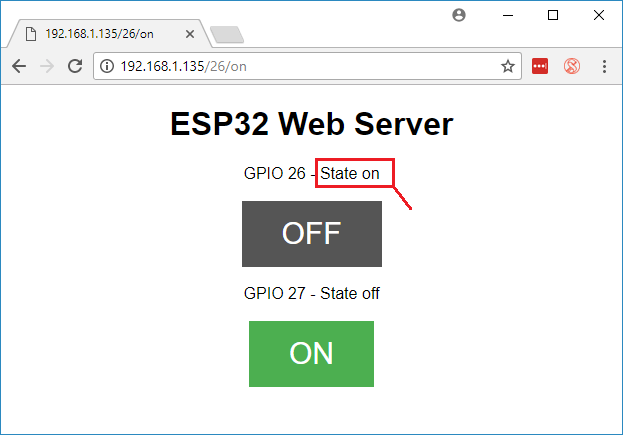


You can also see other information about the HTTP request.

At the same time, you can take a look at the Serial Monitor to see what’s going on in the background. For example, when you click the button to turn GPIO 26 ON, ESP32 receives a request on the **/26/on** URL.



When the ESP32 receives that request, it turns the LED attached to GPIO 26 ON and updates its state on the web page.



**Wrapping Up**

In this tutorial we’ve shown you how to build a web server with the ESP32. We’ve shown you a simple example that controls two LEDs, but the idea is to replace those LEDs with a relay, or any other output you want to control. For more projects with ESP32, check the following tutorials:

* [Build an All-in-One ESP32 Weather Station Shield](https://randomnerdtutorials.com/build-an-all-in-one-esp32-weather-station-shield/)
* [ESP32 Servo Motor Web Server](https://randomnerdtutorials.com/esp32-servo-motor-web-server-arduino-ide/)
* [Getting Started with ESP32 Bluetooth Low Energy (BLE)](https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/)
* [More ESP32 tutorials](https://randomnerdtutorials.com/category/esp32/)