# **[Demo 12: How to turn the Arduino ESP32 into a Web Server](http://www.iotsharing.com/2017/05/how-to-turn-esp32-into-web-server.html)**

**1. Introduction**

A **Web Server**will be responsible for returning the web page to the Client when client connect to it through HTTP/HTTPs protocol (HTTP/HTTPs is a TCP server at default port 80/443 with more constraints for HTTP/HTTPS protocol). For example: when you use your web browser (Firefox, Chrome client) to access "google.com", first the DNS system will resolve "google.com" to IP address then using this IP address to connect to the Google Web Server, the server will return the Search page to you.  
In this demonstration we will make ESP32 a HTTP (port 80) Web Server. Because we are in local network so we do not have DNS system so we will use mDNS to resolve the IP address of ESP32 . When user using web browser to access "**esp32.local**", ESP32 Web Server will return the "Hello world" string on the screen of web browser.

We also have another concept, called **HTTP status code** to express the status of processing client request. Example: if request is successful then return status code **200**, in case request is failed, we can return **503**. You can refer: <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>  
**Note**: for Firefox user, you can monitor what ESP32 Web Server responding by go to **Menu - Developer - Web Console – Network** tab.

**2. Hardware**

You do not need any extra hardware.

**3. Software**

There are 2 ways to implement this:

**3.1 Create a HTTP response manually**

We create a WiFi Server as in [Demo 8: How to use TCP/IP with ESP32](http://www.iotsharing.com/2017/05/tcp-udp-ip-with-esp32.html" \t "http://www.iotsharing.com/2017/05/_blank). This server listen on port 80. When browser access it, we return the data according to HTTP protocol. That is: **"HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n".**

In this response:

 -  **"HTTP/1.1 200 OK"**, it mean server accept the connection and respond **HTTP status code is 200 OK.**Following by "\r\n", it is constraint of HTTP protocol.

**- "Content-Type: text/html\r\n\r\n"**, it means the response content type is html format**,**following by "\r\n"

**- "\r\nHello from ESP32 at**

", this is the content of response, it is a well-known html string.

For more information about HTTP, you can refer: <https://tools.ietf.org/html/rfc2616>

**3.2 Using the Web Server library which I modified from ESP8266**

You can download it here: <https://github.com/nhatuan84/esp32-webserver>

Then unzipping downloaded file and copy it in to Arduino/libraries folder.

It has some interfaces:

**- ESP32WebServer server(port)**: create an instance of ESP32WebServer and the port which it will listen on.

**- server.on("/", handleRoot)**: register the callback function that will be invoked when client request "/"**.**Here client request root folder "/" and the call back handleRoot (prototype: void handleRoot()) will be invokedand reteurn the HTTP status code **200** by invoking **server.send(200, "text/plain", "hello from ESP32!")** with the content type is **"text/plain"** and response content is **"hello from ESP32!"**

**- server.onNotFound(handleNotFound)**: in case we cannot process the client request then the callback function handleNotFound() will be invoked and return the HTTP status code **404** by invoking **server.send(404, "text/plain", "Not found")**with the content type is **"text/plain"** and response content is **"Not found".**  
**- server.streamFile(file, content-type):**this function stream the content of file back to client along with the content type of file (E.g: text/html, application/x-gzip, ...).  
**- server.begin():** start web server

**\* Let 's make 2 versions: one for 3.1 (esp32webserver1) and one for 3.2(esp32webserver2).**We will apply: [Demo 8: How to use TCP/IP with ESP32](http://www.iotsharing.com/2017/05/tcp-udp-ip-with-esp32.html" \t "http://www.iotsharing.com/2017/05/_blank) and [Demo 9: How to use mDNS to resolve host names to ESP32 IP addresses](http://www.iotsharing.com/2017/05/how-to-use-mdns-to-resolve-hostname-esp32-ipaddress.html" \t "http://www.iotsharing.com/2017/05/_blank)

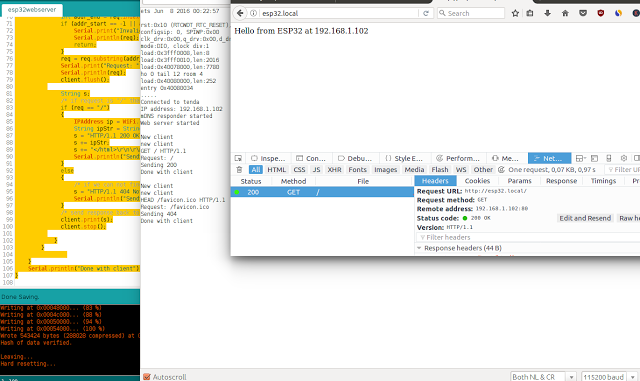
- **esp32webserver1**: Create an Arduino project and Save as **esp32webserver1** with code:

|  |
| --- |
| #include <WiFi.h>  #include <ESPmDNS.h>  #include <WiFiClient.h>  const char\* ssid = "tenda";  const char\* password = "phong707";  /\* TCP server at port 80 will respond to HTTP requests \*/  WiFiServer server(80);  void setup(void)  {  Serial.begin(115200);  /\* Connect to WiFi network \*/  WiFi.begin(ssid, password);  /\* Waiting for connection \*/  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  /\* Set up mDNS \*/  if (!MDNS.begin("esp32")) {  Serial.println("Error setting up MDNS responder!");  while(1) {  delay(1000);  }  }  Serial.println("mDNS responder started");  /\* Start Web Server server \*/  server.begin();  Serial.println("Web server started");  /\* Add HTTP service to MDNS-SD \*/  MDNS.addService("http", "tcp", 80);  }  void loop(void)  {  /\* Check if a client has connected \*/  WiFiClient client = server.available();  if (!client) {  return;  }  Serial.println("");  Serial.println("New client");  if (client) {  Serial.println("new client");  /\* check client is connected \*/  while (client.connected()) {  /\* client send request? \*/  if (client.available()) {  /\* request end with '\r' -> this is HTTP protocol format \*/  String req = client.readStringUntil('\r');  /\* First line of HTTP request is "GET / HTTP/1.1"  here "GET /" is a request to get the first page at root "/"  "HTTP/1.1" is HTTP version 1.1  \*/  /\* now we parse the request to see which page the client want \*/  int addr\_start = req.indexOf(' ');  int addr\_end = req.indexOf(' ', addr\_start + 1);  if (addr\_start == -1 || addr\_end == -1) {  Serial.print("Invalid request: ");  Serial.println(req);  return;  }  req = req.substring(addr\_start + 1, addr\_end);  Serial.print("Request: ");  Serial.println(req);  client.flush();    String s;  /\* if request is "/" then client request the first page at root "/" -> we process this by return "Hello world"\*/  if (req == "/")  {  IPAddress ip = WiFi.localIP();  String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' + String(ip[3]);  s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>Hello from ESP32 at ";  s += ipStr;  s += "</html>\r\n\r\n";  Serial.println("Sending 200");  }  else  {  /\* if we can not find the page that client request then we return 404 File not found \*/  s = "HTTP/1.1 404 Not Found\r\n\r\n";  Serial.println("Sending 404");  }  /\* send response back to client and then close connect since HTTP do not keep connection\*/  client.print(s);  client.stop();    }  }  }  Serial.println("Done with client");  } |

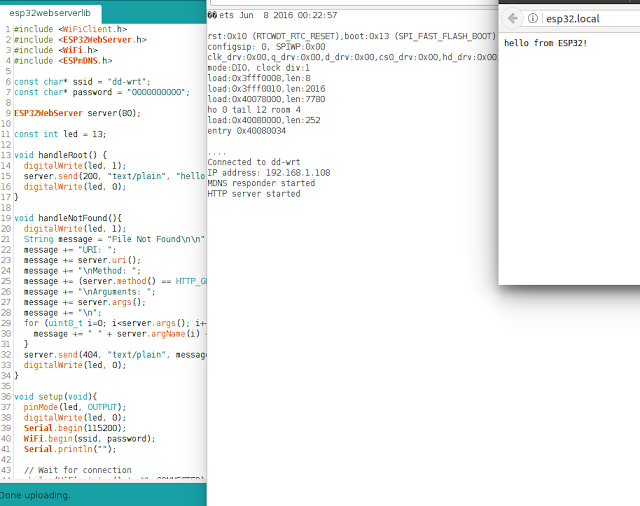
- **esp32webserver2**: Create an Arduino project and Save as **esp32webserver2** with code:

|  |
| --- |
| #include <WiFiClient.h>  #include <ESP32WebServer.h>  #include <WiFi.h>  #include <ESPmDNS.h>  /\*fill your ssid and password here \*/  const char\* ssid = "mieu mieu 01";  const char\* password = "09471919479";  ESP32WebServer server(80);  /\* this callback will be invoked when user request "/" \*/  void handleRoot() {  /\* server respond 200 with content "hello from ESP32!" \*/  server.send(200, "text/plain", "hello from ESP32!");  }  void handleNotFound(){  String message = "File Not Found\n\n";  server.send(404, "text/plain", message);  }  void setup(void){  Serial.begin(115200);  WiFi.begin(ssid, password);  Serial.println("");  /\* Wait for connection \*/  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  /\* we use mDNS here http://esp32.local \*/  if (MDNS.begin("esp32")) {  Serial.println("MDNS responder started");  }  /\* register callback function when user request root "/" \*/  server.on("/", handleRoot);  server.onNotFound(handleNotFound);  /\* start web server \*/  server.begin();  Serial.println("HTTP server started");  }  void loop(void){  server.handleClient();  } |

1. **Result**

[](https://3.bp.blogspot.com/-iZELOyx0_7c/WSG9TtHX_RI/AAAAAAAAD5s/WYUKyVlJW1gHrFAUDNETK4abqpIOY5VUwCEw/s1600/webserver1.png)

**Figure: esp32webserver1, Firefox client access "http://esp32.local/" and get response**

**[](https://4.bp.blogspot.com/-HsMFRV839i4/WSG_ACuHKFI/AAAAAAAAD54/4anG_5otMdgHoJt9CKFSUldM4ZOkypw7gCEw/s1600/webserver2.png)**

**Figure: esp32webserver2, Firefox client access “http://esp32.local/” and get response**