https://electropeak.com/learn/create-a-web-server-w-esp32/

MODULE 2b

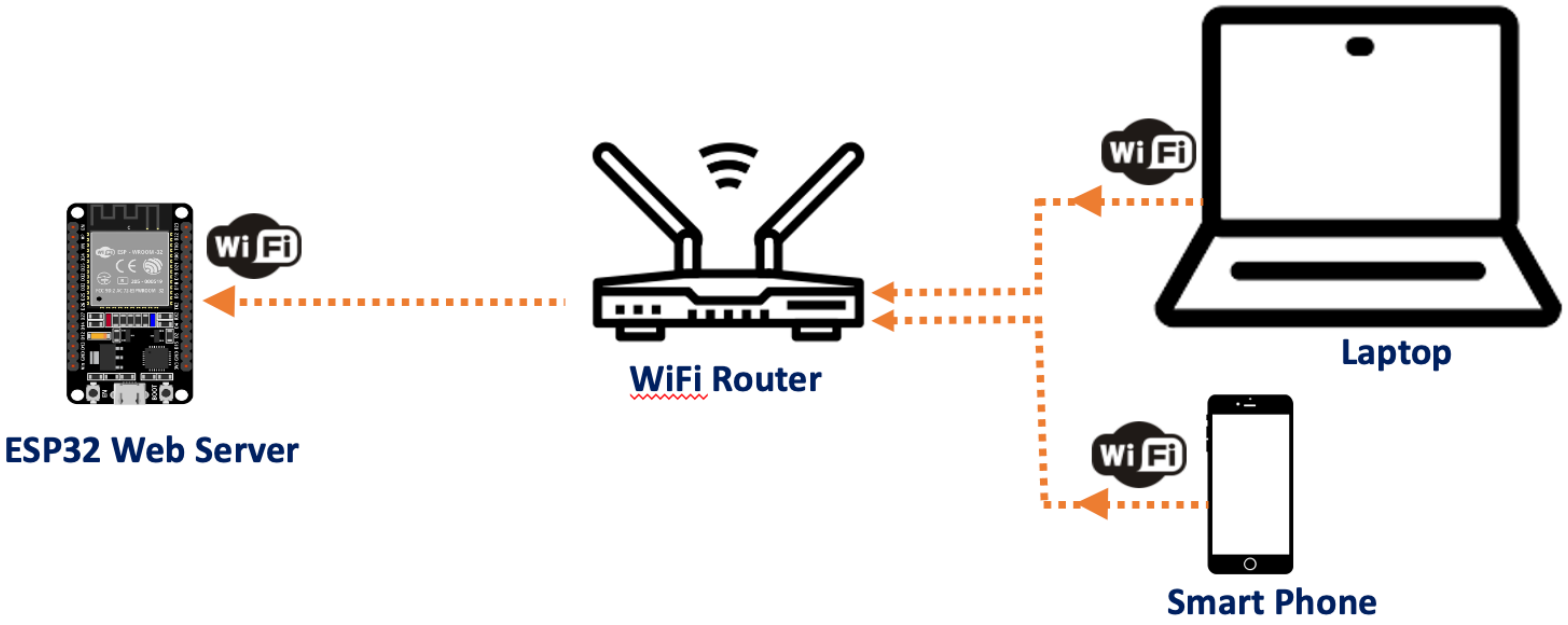
Create a Web Server with ESP32 and Arduino IDE

safyzan salim
019 622 0575

Project Overview

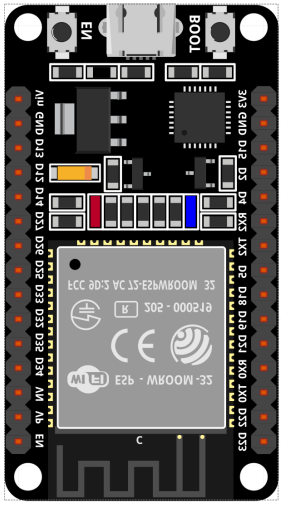
Before we start, it is important to outline what our web server will do:

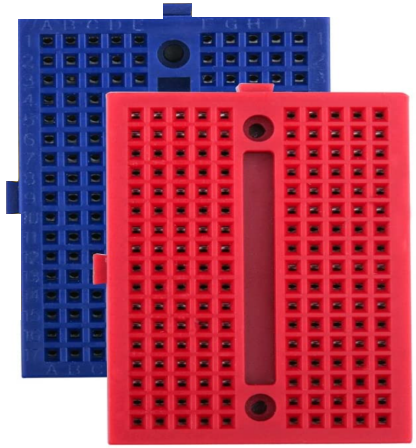
- → Setting up ESP32 as web server (Station/STA).
- → In this mode, ESP32 gets IP from wireless router to which it is connected.
- → Can locally access the ESP32 web server by typing either ***IP address*** or ***esp32.local*** at browser.
- → Access ESP32 from various devices.



Components

- → 1 x ESP-32 Wifi+Bluetooth 2-In-1 Development Board for Arduino (30 pin) + Cable.
- → 2 x Mini Breadboard.





Working File

→ Download the file working file from http://bit.ly/3aB3WBL by click at 

.

→ Unzip and find ***sketch-iii_esp32-webserver.ino***. Double click to open the file with Arduino IDE.

→ Click ***OK*** when Arduino pop-up window appears. This action will create a folder that carry same name with sketch (***sketchiii_esp32-webserver.ino***) and move the sketch into it. Click ***Cancel*** will not open the sketch at all.

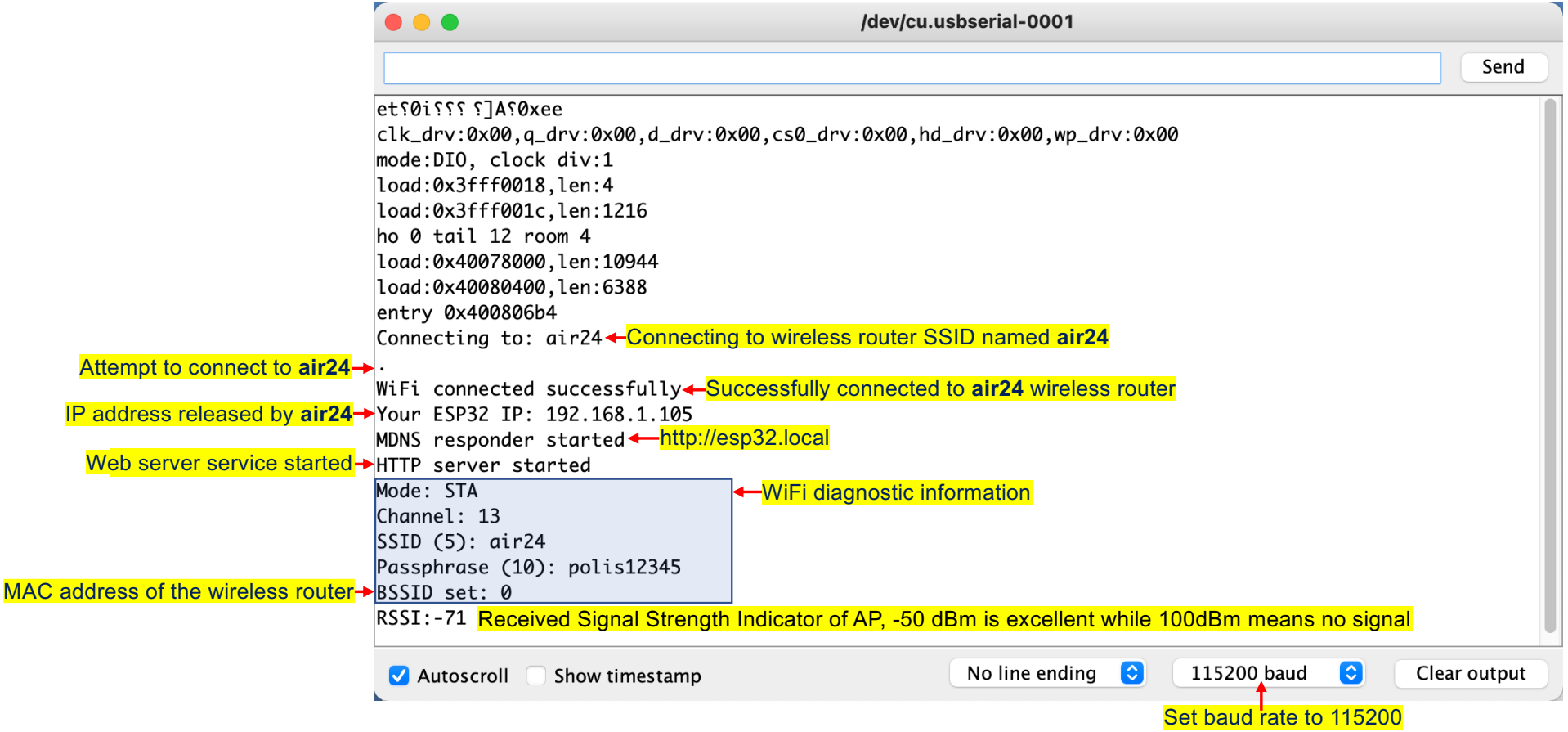
→ Before upload the sketch to the board, modify the following two variables and suit with your network credentials. This will allow ESP32 establish connection with your network.

```
16 // enter your network credentials
17 const char* ssid = "SSID";
18 const char* password = "PASSWORD";
```

→ Upload the sketch to your ESP32 board. Ensure that you have selected the right board and set the current port. Don't forget to press **BOOT** button. Refer to **Troubleshoot** section if you have problem.

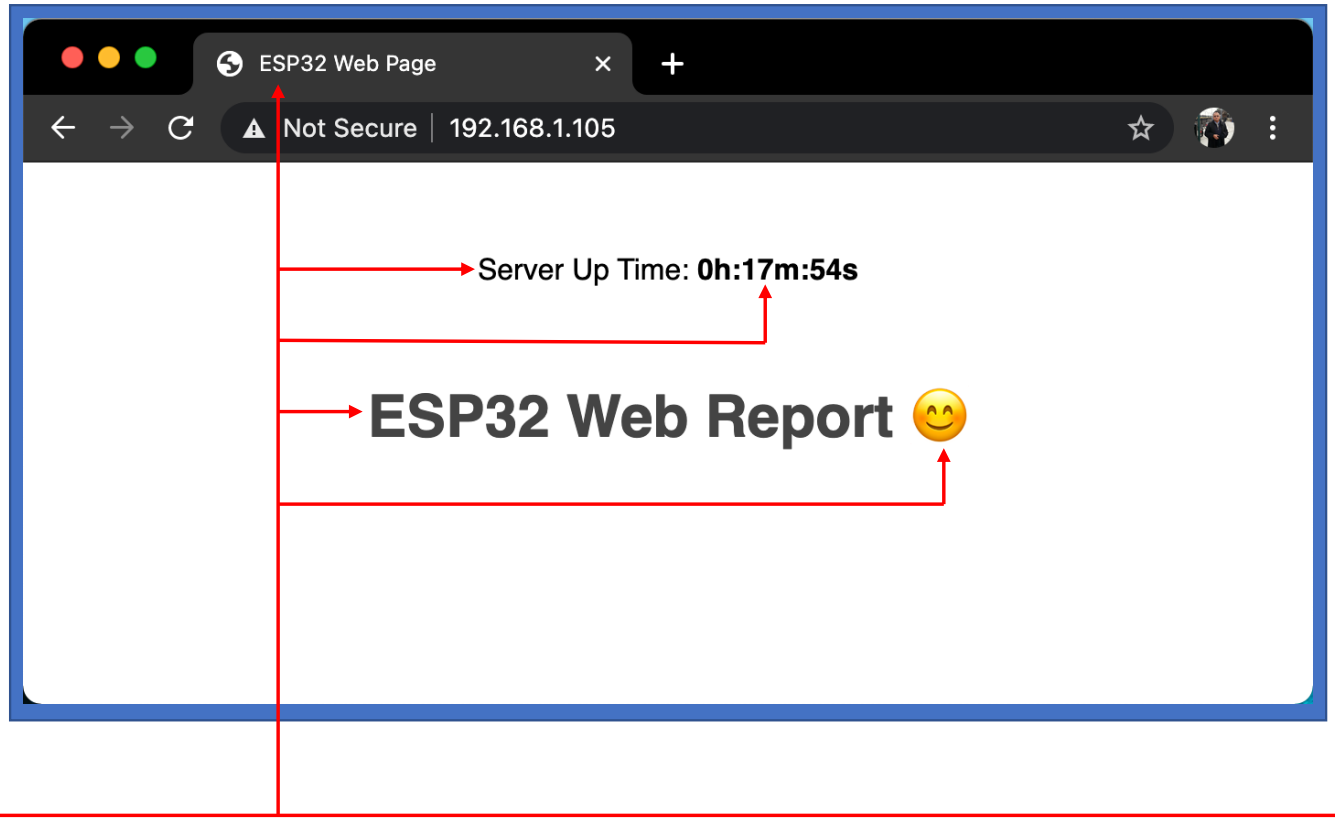
The Output

→ Click the Serial Monitor icon to display the serial data receive & transmit by ESP32.



The Output

→ Open a browser and type the ESP32 IP address, ***192.168.1.105***

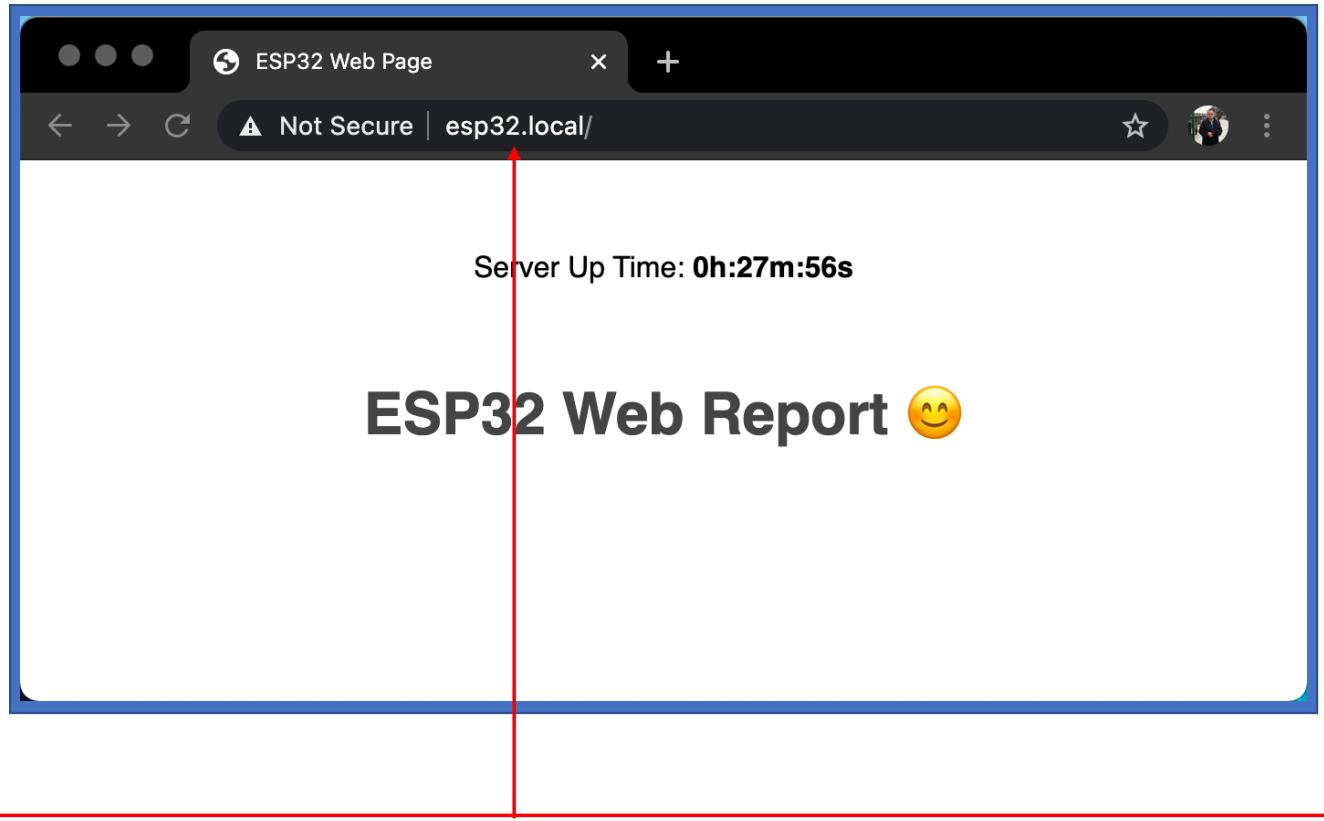


Compare the output with the sketch & indicate which line produces them.

What is the refresh rate? Trace the syntax.

The Output

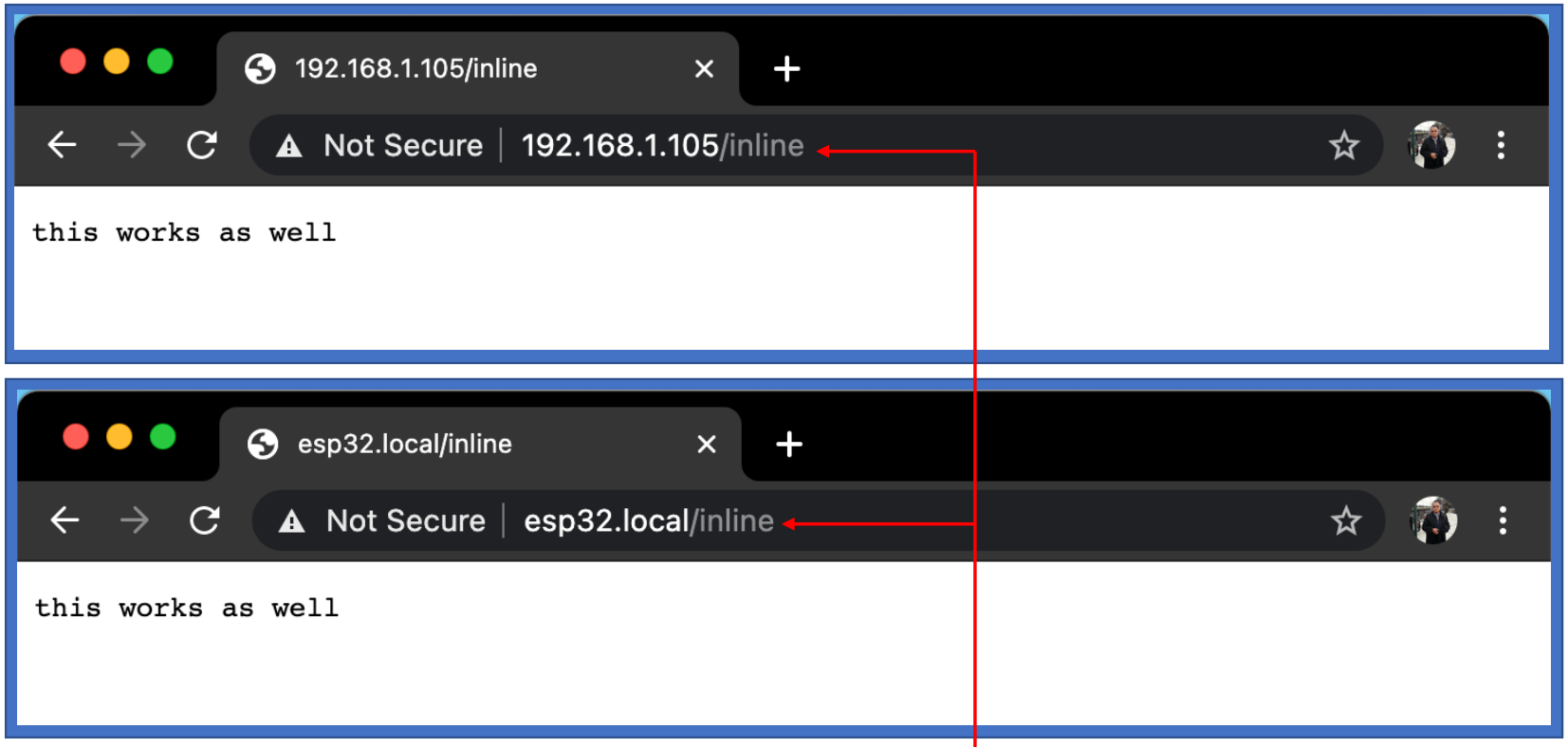
→ You can also type ***esp32.local*** instead of ***192.168.1.105***



Compare the output with the sketch & indicate which line produce this.

The Output

→ Add ***inline*** at the end of ***192.168.1.100/*** and ***esp32.local/***.



Trace which syntax produce this output.

Code Explanation

- Filename: ***sketch-iii_esp32-webserver.ino***
- This sketch consists of 4 functions:

  void setup( ), void loop( ), String SendHTML( ), void handle_root( )

- Start the sketch with the library includes:

  Wifi.h is to allow ESP32 to join WiFi network

  WebServer.h is to enable HTTP web server service to run on EPS32

  ESPmDNS.h is to resolves hostname to ESP32 IP address

```
 8 #include <WiFi.h>
 9 #include <WebServer.h>
10 #include <ESPmDNS.h>
```

- Add the network credential to which wireless router the ESP32 will be connected:

```
16 // enter your network credentials
17 const char* ssid = "SSID";
18 const char* password = "PASSWORD";
```

Code Explanation

→ Create a Web Server object that listens for HTTP request in port 80.

```
20 // port 80 is default website port
21 WebServer server(80);
```

Line #27 - #78 took place in void setup( ) function

→ Opens serial port & set data rate at 115200. This will allow us to see text & values (send & receive) activities in ESP32 on your computer/laptop thru Serial Monitor by using ***Serial.print( )*** command. The 9,600 baud rate is approximately 1,000 characters per second.

```
27 void setup() {
28   Serial.begin(115200);
29   Serial.print("Connecting to: ");
30   Serial.println(ssid);
```

→ Set ESP32 as web server mode. Other modes available: **WIFI_OFF** (turn off WiFi), **WIFI_STA** (Station mode), **WIFI_AP** (Access Point mode), **WIFI_AP_STA** (both Station and Access Point mode). In **WIFI_STA** mode, the ESP32 will gets the IP address from wireless router that connects to it.

```
32   // set ESP32 mode
33   // WiFi.mode(mode);
```

Code Explanation

→ The following function is to make ESP32 a WiFi client that connect to a network with provided ssid and password. If we use this function can replace **WiFi.mode(WIFI_STA)** function.

```
35  // ESP32 will connect to your WiFi modem with SSID & PASSWORD
36  WiFi.begin(ssid, password);
```

→ Print dotted lines until ESP32 connected to wireless router.

```
38  // wait until ESP32 connect to WiFi
39  while (WiFi.status() != WL_CONNECTED) {
40  delay(1000);
41  Serial.print(".");
42  }
```

→ The following information will be display at Serial Monitor including EPS32 IP address.

```
44  Serial.println("");
45  Serial.println("WiFi connected successfully");
46  Serial.print("Your ESP32 IP: ");
47
48  // display "ESP32 IP Address" at Serial Monitor
49  Serial.println(WiFi.localIP());
```

Code Explanation

→ ESP32 will multicast a message including its IP address to members of network and they will update their mDNS cahces. Domain System resolves host name to IP address. To access, just use http://esp32.local instead of http://IPaddress.

```
51  // set up mDNS responder:
52  // "http://esp32.local" instead of "http://IPaddress"
53  if (MDNS.begin("esp32")) {
54    Serial.println("MDNS responder started");
55  }
```

→ ***server.on*** will specify what ESP32 to do upon client request. When requested, handle_root function is called and display the web page.

```
57  // attach handles
58  server.on("/", handle_root);
```

→ When client request IPaddress/inline or esp32/inline, a plain text website with the 'this works as well' will appears.

```
60  // "http://ipaddress/inline" OR "htpp://esp32.local/inline"
61  server.on("/inline", []() {
62    server.send(200, "text/plain", "this works as well");
63  });
```

Code Explanation

→ Start listening for incoming connections. ESP32 will display WiFi diagnostic information and received signal strength over Serial Monitor.

```
65   // start web server
66   server.begin();
67   Serial.println("HTTP server started");
68   delay(100);
69
70   // print key WiFi diagnostic information
71   WiFi.printDiag(Serial);
72
73  // print the received signal strength:
74   long rssi = WiFi.RSSI();
75   Serial.print("RSSI:");
76   Serial.println(rssi);
77
78 } // end of void setup( ) function
```

Code Explanation

→ In the loop function, ESP32 will check if any client has sent an HTTP request or not. If client request is available, the ESP32 will deliver the requested HTML page by calling the ***handle_root( )*** function.

```
84 void loop() {
85   // monitors the presence of a client
86   // and delivers the requested HTML page
87   server.handleClient();
88 } //end of loop function
```

→ The ***handle_root( )*** function gets called when the web page URL is set to root like esp32.local. The function simply call ***SendHTML( )*** to display at browser.

```
94 // Handle root url (/)
95 void handle_root() {
96   server.send(200, "text/html", SendHTML());
97 }
```

Code Explanation

Line #101 to #132 took place in String SendHTML ( ) function

→ You may add lines that reads sensor values, control motors, servos, leds, timer etc within ***SendHTML( )*** function which bind with HTML syntax. We are using ***millis( )*** function as time measurement within the sketch. The time itself will become zero whenever power supply is removed or reset the board.

```
 99 // HTML & CSS contents which display on web server
100 // park your GPIO activities in "SendHtml( )" function
101 String SendHTML( ){
102   // returns the number of milliseconds that
103   // your ESP32 board since powered up,
104   // & become zero when reset or unplugged
105   int sec = millis() / 1000;
106   int min = sec / 60;
107   int hr = min / 60;
```

Code Explanation

→ Basically, the HTML used in ESP32 is similar with other HTML for web, except need to add certain lines in order to allow the board ability to translate the language that its understand. FYI, the server will refresh every 5 second – refer to Line #112.

```
109  String ptr = "<!DOCTYPE html> <html>\n";
110  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
111  ptr +="<title>ESP32 Web Page</title>\n";
112  ptr +="<meta http-equiv=refresh content=5>\n";
113  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
114  ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
115  ptr +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
116  ptr +="</style>\n";
117  ptr +="</head>\n";
```

Code Explanation

→ Go through every line & notice that the syntax without double quote ***"foo"*** are the C language while within ***"foo"*** are HTML syntax. Different application will have different sensors, motors & web layout.

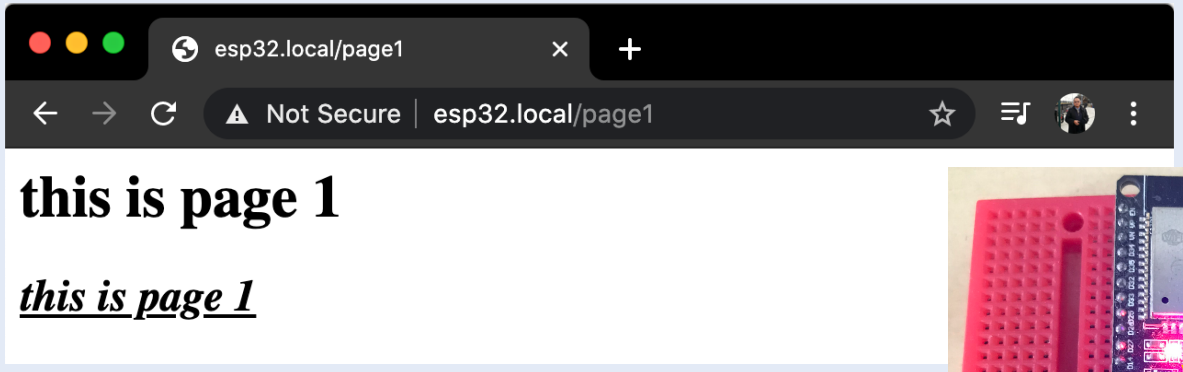
```
118   ptr +="<body>\n";
119   ptr +="<div id=\"webpage\">\n";
120   ptr += "Server Up Time: <b>";
121   ptr += hr;
122   ptr += "h:";
123   ptr += min % 60;
124   ptr += "m:";
125   ptr +=  sec % 60;
126   ptr += "s </b>\n";
127   ptr +="<h1>My First Web Server with ESP32 &#128522;</h1>\n";
128   ptr +="</div>\n";
129   ptr +="</body>\n";
130   ptr +="</html>\n";
131   return ptr;
132 }
```

QUESTIONS?

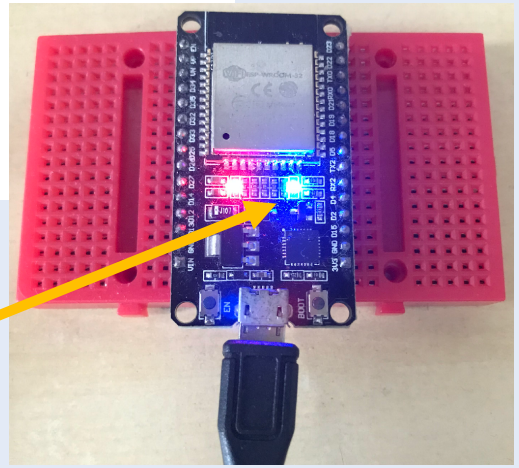
Questions?

EXERCISE

Add a few lines in the sketch & produce output as follows.
[10 minutes activity]



Turn on built-in led when client browsing the ***esp32.local/page1*** website.



ANSWER

```
...
server.on("/page1", [ ]( ) {
   digitalWrite(led,1);
   server.send(200, "text/html",
   "<h1>this is page 1</h1>"
   "<h2><i><u>this is page 1</u></i></h2>"
   )
...
```

TROUBLESHOOT GUIDE

Failed to upload the sketch to ESP32? Follow these steps:

→ Press REBOOT button until the IDE approaches **Connecting**… segment;

→ **OR**, check whether you have chosen the correct **Board** and set the correct **PORT**;

* Choose the right board by going to **Tools > Boards > ESP32 Dev Module**

* Select the correct port at **Tools > Port > choose the appropriate serial port**

→ **OR**, unplugging the board from USB and plugging back;

→ **OR**, **Verify** to confirm that your sketch is error free;

→ **OR**, Swap to other PC / Laptops;

→ **OR**, Use different board;

→ You are using micro-USB power cable instead of power & data cable.