

## **IERG1000 Project 2 – Lab-4**

### **WiFi Thermometer**

#### **Important:**

**Your project task is Experiment-4.4**

#### **Objectives**

- To use WiFi ability and build web server on ESP32.
- To convert Lab-3 standalone thermometer with Wireless access.
- To start ESP32 as a client of your current WiFi network.
- To start ESP32 as a HotSpot.
- To start ESP32 as an Access Point.

#### **Equipment / Component / Software**

- Arduino IDE (ready for ESP32, SSD1306 and AHT10);
- ESP32 microprocess module (x1);
- OLED display module, SSD1306, size 128x64 (x1);
- Thermo sensor module, AHT10 (x1);
- WiFi environment (SSID, password);
- Smartphone or smart device with WiFi;
- Micro USB cable for ESP32 (x1);
- Breadboard (x1);
- A bunch of wires for breadboard;
- Tactile switch (x1);
- Thumb drive (x1, optional, if you don't have the right to write file to the computer)

#### **Introduction**

Wireless communication is essential today. Thus, some microprocessors have embedded WiFi and Bluetooth or other protocols. We will use ESP32 embedded with WiFi to build a web server on WiFi.

If WiFi is available in your environment, please have the WiFi SSID and password ready. In the second experiment, we will use your smartphone to set as an Access Point to provide WiFi through Hotspot function.

No extra wiring is needed in this Lab.

### Experiment 4.1: Web Server on ESP32

Arduino provided official example. So, it is very easy to setup web server on ESP32.

#### Procedures:

1. Open official example 'File' Example 'WebServer' 'HelloServer' (Figure-1.1).
2. Modify the lines 'ssid' and 'password', fill-in your WiFi ssid and password. (Figure-1.2)
3. Turn on Serial Monitor and set baud rate to 115200.
4. Compile and upload the project.
5. After uploading, some dots (Figure-1.3) will appear and increase for a few seconds, ESP32 is trying connect to your WiFi network. If the number of dots gets more than twenty, it means that ESP32 cannot join your WiFi network. It may be due to incorrect ssid or password or you have set the MAC filtering or no DHCP server in your WiFi router.
6. After ESP32 has joined your WiFi network successfully, it should be like Figure-1.3 and show the IP address of ESP32.
7. Use a web browser (PC/smartphone on the same network) to access the web page on ESP32 with its IP (e.g. 192.168.0.238 as Figure-1.3). The result should be like Figure-1.4.

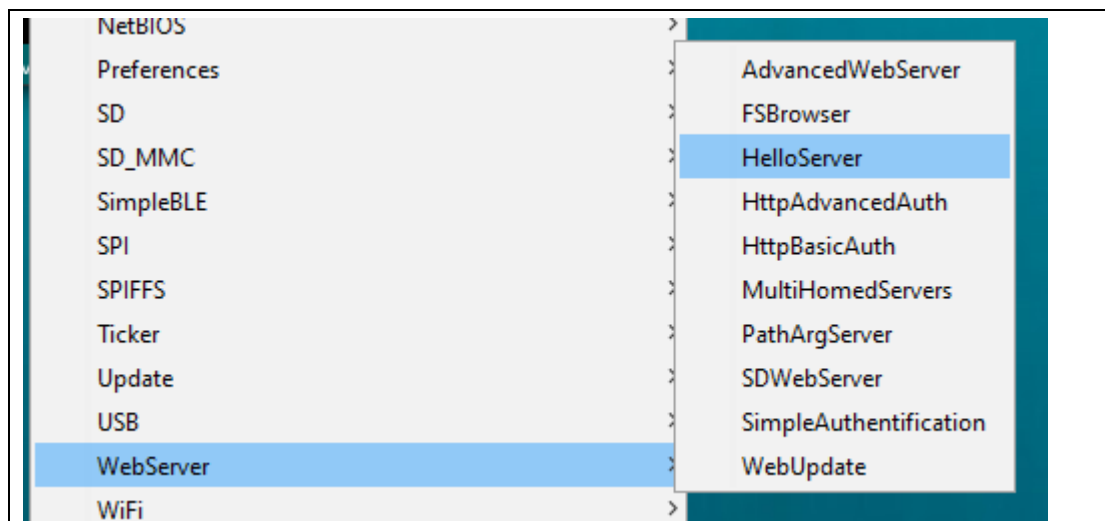


Figure-1.1 Open official example 'HelloServer'

```
const char* ssid = ".....";
const char* password = ".....";
```

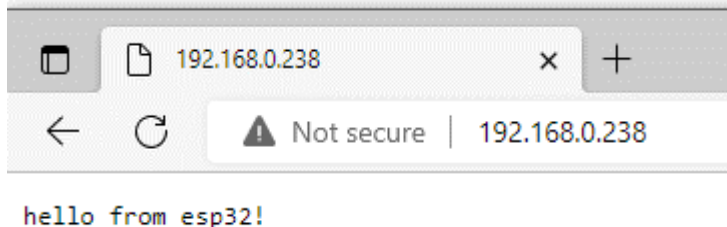
Figure-1.2 Fill-in WiFi ssid and password

```

.....
Connected to BD15B5
IP address: 192.168.0.238
MDNS responder started
HTTP server started

```

Figure-1.3 ESP32 join your WiFi success



```

hello from esp32!

```

Figure-1.4 Default page form ESP32 web server

8. In case you get error '**Brownout detector was triggered**' after uploading, as in Figure-1.5, it means that your computer USB port does not offer enough power. Try to plug to another USB port or connect ESP32 to your PC directly if you are using USB hub.

**Note:** If this problem cannot be solved, come to IE teaching laboratory and change another ESP32 module or use another computer.

```

Brownout detector was triggered
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13864
load:0x40080400,len:3608
entry 0x400805f0

```

Figure-1.5 Brownout detector was triggered

9. Understand the program
- Figure-1.6, comments are added. Change the on-board LED to pin-2.
  - Figure-1.7, `handleRoot( )` is the subroutine that generates page for Figure-1.4. We will use this part in our project.
  - Figure-1.7, `handleNotFound( )` generates page if the URL incorrect.
  - Figure-1.8, `setup( )`, comments are added. Try to understand.
  - Figure-1.9, `loop ( )`, Run forever, when web server is called, find the handler and send response content.

```

HelloServer$
#include <WiFi.h>           //-- important for WiFi
#include <WiFiClient.h>     //-- important for WiFi
#include <WebServer.h>      //-- important for WiFi
#include <ESPmDNS.h>        //-- important for WiFi

const char* ssid = ".....";    //-- your WiFi ssid
const char* password = "....."; //-- your WiFi password

WebServer server(80);          //-- web server port 80

//const int led = 13;    //-- comment out this line
const int led = 2;        //-- the LED on ESP32 is pin-2

```

Figure-1.6 before setup( )

```

void handleRoot() { ←
    digitalWrite(led, 1);
    server.send(200, "text/plain", "hello from esp32!");
    digitalWrite(led, 0);
}

void handleNotFound() { ←
    digitalWrite(led, 1);
    String message = "File Not Found\n\n";
    message += "URI: ";
    message += server.uri();

```

Figure-1.7 handleRoot( )

```

void setup(void) {
  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);    //-- setup WiFi
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {    //-- try to connect until success
    delay(500);
    Serial.print(".");    //-- print dots forever if still fail
  }
  Serial.println("");
  Serial.print("Connected to ");    //--when connect success
  Serial.println(ssid);    //--when connect success
  Serial.print("IP address: ");    //--when connect success
  Serial.println(WiFi.localIP());    //--when connect success

  if (MDNS.begin("esp32")) {    //--start DNS
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);    //--if URL="/" use handleRoot() <<< Figure-1.4

  server.on("/inline", []() {    //-- if URL='/inline'
    server.send(200, "text/plain", "this works as well");
  });

  server.onNotFound(handleNotFound);    //-- if URL incorrect

  server.begin();    //-- start Web Server
  Serial.println("HTTP server started");
}

```

Figure-1.8 setup( )

```

void loop(void) {
  server.handleClient();
  delay(2);    //--allow the cpu to switch to other tasks
}

```

Figure-1.9 loop( )

# 10. Modify the program

- a. Add a variable **centigrade** before setup(). (Figure-1.10)
- b. Show variable centigrade in handleRoot(). (Figure-1.11)
- c. Web server responses one line only with 'server.send( )'. You have to use '\n' in the string 'message' to send carriage return (start a new line).
- d. Increase variable centigrade in loop(). (Figure-1.12)

# 11. Compile and upload the program. Access the page **few times**. The result should be like Figure-1.13. Explain the reason why the displayed numbers are not same.

```

#include <WiFi.h>           //-- important for WiFi
#include <WiFiClient.h>     //-- important for WiFi
#include <WebServer.h>      //-- important for WiFi
#include <ESPmDNS.h>        //-- important for WiFi

const char* ssid = ".....";    //-- your WiFi ssid
const char* password = "....."; //-- your WiFi password

WebServer server(80);          //-- web server port 80

//const int led = 13;  //-- comment out this line
const int led = 2;  //-- the LED on ESP32 is pin-2
float centigrade;  //-- add variables

```

Figure-1.10 Add a variable 'centigrade'

```

void handleRoot() {
    digitalWrite(led, 1);
    String message = "Hello from ESP32!\n";  //-- refer handleNotFound
    message += centigrade;                    //-- to concatenate strings
    server.send(200, "text/plain", message); //-- show concatenated string
    digitalWrite(led, 0);
}

```

Figure-1.11 handleRoot( ) shows concatenated string

```

void loop(void) {
    server.handleClient();
    centigrade = centigrade + 0.1;
    delay(2); //allow the cpu to switch to other tasks
}

```

Figure-1.12 variable centigrade is increasing in loop( )

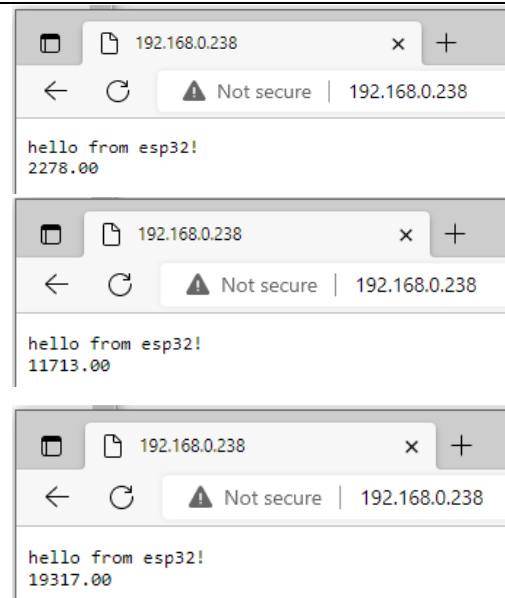


Figure-1.13 variable is changing

### **Experiment 4.2: (Optional) Use your smart phone as WiFi Access Point**

If you are working in the environment does not have WiFi. You can use your smartphone as a WiFi AP (network hub). It is like you share your phone data Internet access to your tablets.

#### **Procedures:**

1. Turn on your smartphone 'setting/connection.
2. Find out the function 'Hotspot' and enable it.
3. Kick the Hotspot and check its details. Find out the **name (SSID) and password** of the Hotspot.
4. Fill the name and password to ESP32 program (Figure-1.2).
5. Compile and upload your program to ESP32. Check the IP of ESP32 that connected to your smartphone.
6. Use a browser of your smartphone to access ESP32 web page. The results should be like Figure-1.3.

### **Experiment 4.3: (Optional) Use ESP32 as WiFi Access Point**

In case you have problem sharing WiFi with your smart phone (e.g. newer iOS), you can also set ESP32 module as WiFi Access Point and set your smart device to join its WiFi network.

**Note:** The SSID may not be the same as yours, it may be 'ESP\_xxxxxx'

#### **Procedures:**

1. Open official example 'File' Example 'WiFi' 'WiFiAccessPoint'' (Figure-3.1).
2. Modify the lines to set SSID and password for this ESP32 WiFi network. You should compose another SSID to prevent collision with your classmates' networks.
3. Save the project to a writable folder.
4. Compile the project and upload to ESP32.
5. When the server starts, you can see the IP address of ESP32 on Serial Monitor. (Figure-3.3)
6. Use a smart device to join this ESP32 WiFi network with SSID and password as you set. You can see the result as Figure-3.4. You can also click the links to turn ON and OFF the LED on ESP32.

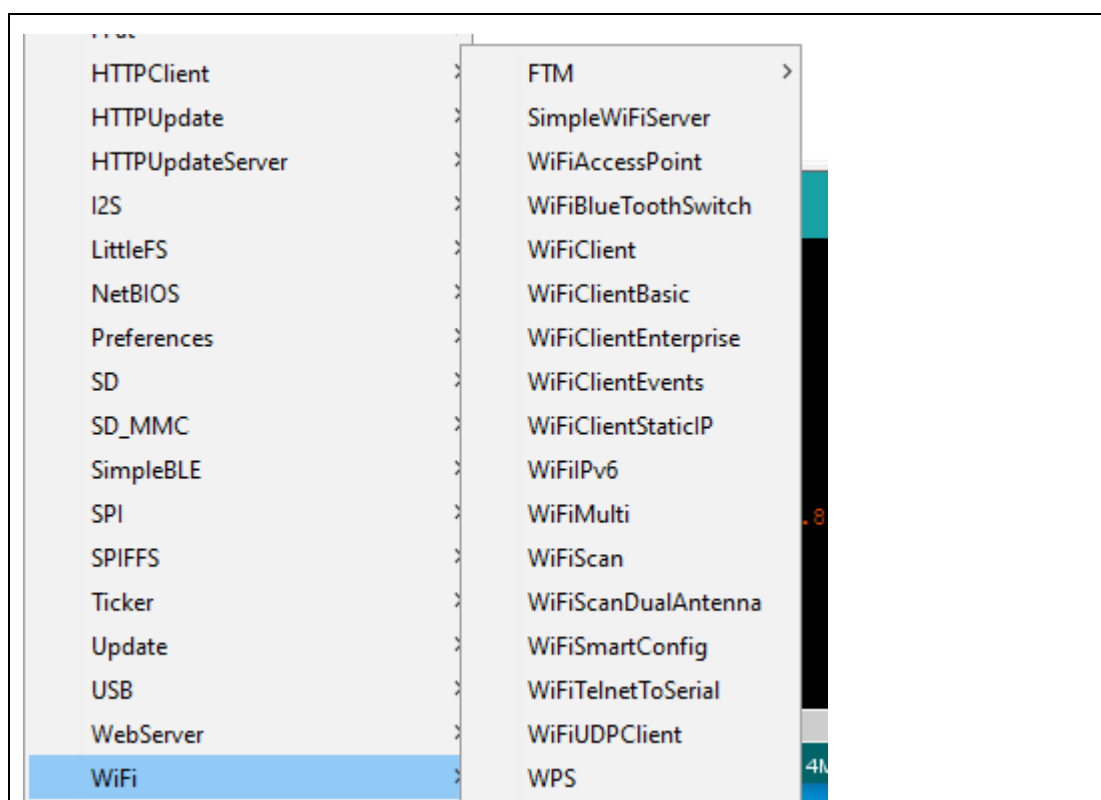


Figure-3.1 Example WiFiAccessPoint

```
// Set these to your desired credentials.
const char *ssid = "IERG1000";
const char *password = "welldone";

WiFiServer server(80);
```

Figure-3.2 Set SSID and password

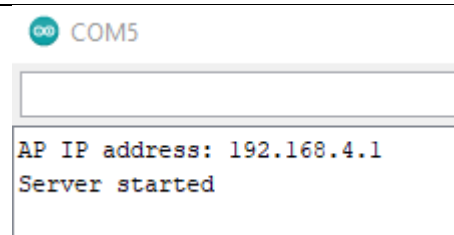


Figure-3.3 This is the IP address of ESP32

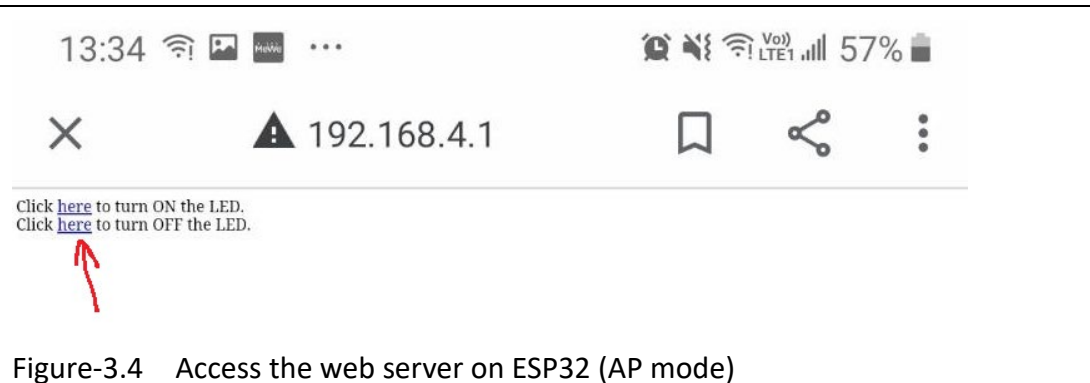


Figure-3.4 Access the web server on ESP32 (AP mode)



### Experiment 4.4: Make WiFi thermometer

Up to now, you can

- Get temperature from thermo-sensor module.
- Show text/ variables on OLED display.
- Show text/variables on ESP32 web page.

It is time to combine them, make a thermometer that can show temperature on OLED display, Serial Monitor and ESP32 web page.

#### Procedures:

1. Check with Lab-2 to Lab-4. Find out the essential programming statements and compose (copy and modify) a new program. There are three important portions in Arduino program.
  - a. Before setup( )
  - b. setup( )
  - c. loop( )
2. As a good software engineer, write down the comments on your program.
3. Show your result to tutor and answer questions about your design.

~ END ~