

รายงาน  
เรื่อง Web Embedded สำหรับ Setup/Configuration

จัดทำโดย  
นาย โสพัส พันธุสุวรรณ รหัส 61042504

เสนอ  
ดร. ผิน ฉัตรแก้วมณี

คณะ เทคโนโลยีสารสนเทศ  
สาขา วิศวกรรมคอมพิวเตอร์  
วิชา CPE 405  
มหาวิทยาลัยศรีปทุม  
คู่มือติดตั้ง Web Embedded

- มีการเสียบจ่ายไฟให้กับอุปกรณ์
- ได้มีการที่เราได้เชื่อมต่อกับอุปกรณ์ผ่าน Wi-Fi ใน Smartphone หรือ Tablet ที่มี Internet

## Cellular Personal Hotspot

Personal Hotspot on your iPhone can provide Internet access to other devices signed into your iCloud account without requiring you to enter the password.

Allow Others to Join



Wi-Fi Password

fAiRxRyX2907 >

Allow other users or devices not signed into iCloud to look for your shared network "MEEBWOW\_Wi-Fi" when you are in Personal Hotspot settings or when you turn it on in Control Center.



### TO CONNECT USING WI-FI

- 1 Choose "MEEBWOW\_Wi-Fi" from the Wi-Fi settings on your computer or other device.
- 2 Enter the password when prompted.



### TO CONNECT USING BLUETOOTH

- 1 Pair iPhone with your computer.
- 2 On iPhone, tap Pair or enter the code displayed on your computer.
- 3 Connect to iPhone from computer.



### TO CONNECT USING USB


- 1 Plug iPhone into your computer.

- ส่วนหน้าเข้าไปใน IP address /apSetup

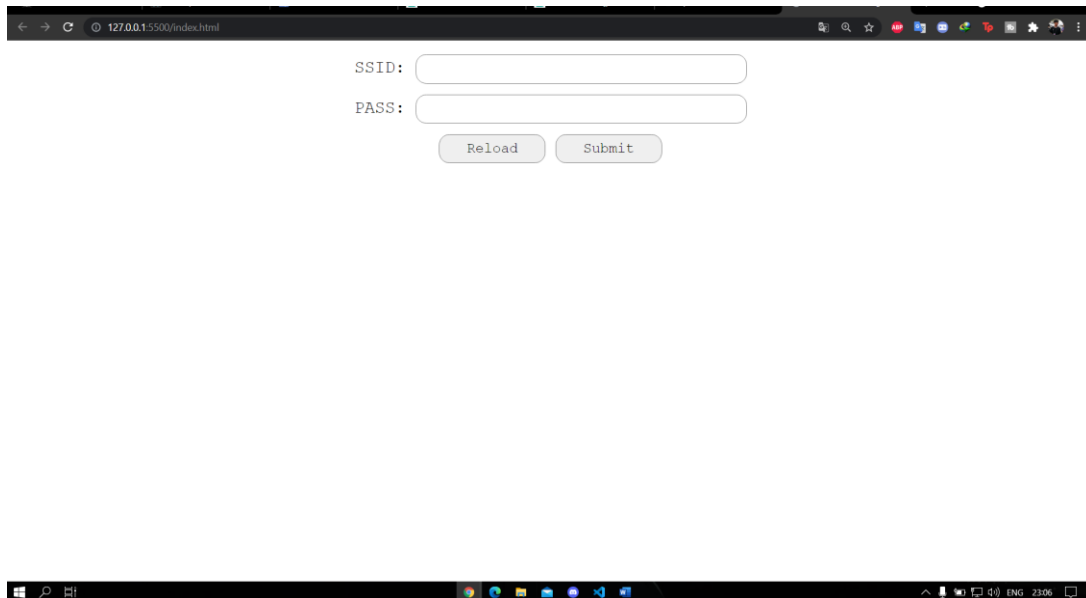
ในส่วนหน้านี้ผมยังเข้าไปไม่ได้เพราะผมไม่มีบอร์ดในการทดสอบถ้าเข้าได้มันจะประมาณนี้ครับ  
ตย.

- เข้า IP address 192.168.4.1/apsetup

```
.....  
Fail To Connect..  
AccessPoint ssid: SPU_2G-[A4F544BF713C]  
IP address (AccessPoint Mode) : 192.168.4.1  
HTTP server started
```



- อันนี้จะเป็นในส่วนหน้า SSID และ PASS ของ Wi-Fi ที่เราจะใช้

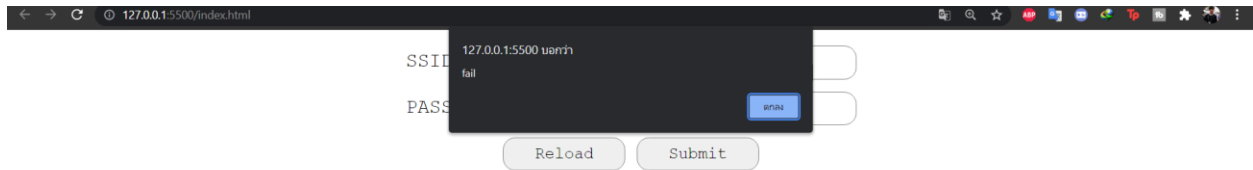


SSID:

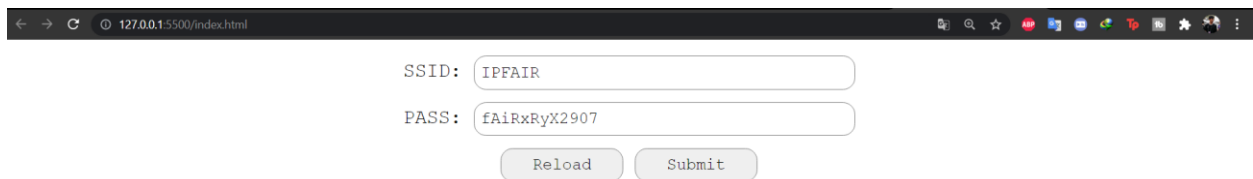
PASS:

- ถ้าเวลาเราจะกด Submit มันจะแจ้งเตือนว่า Sending Success

แต่ตอนนี้ของผมนั้นยังขึ้น Fail ยังไม่ได้ทำการเชื่อมต่อกันในทางบอร์ด ESP32



- ในการ Setup ตัว Wi-Fi

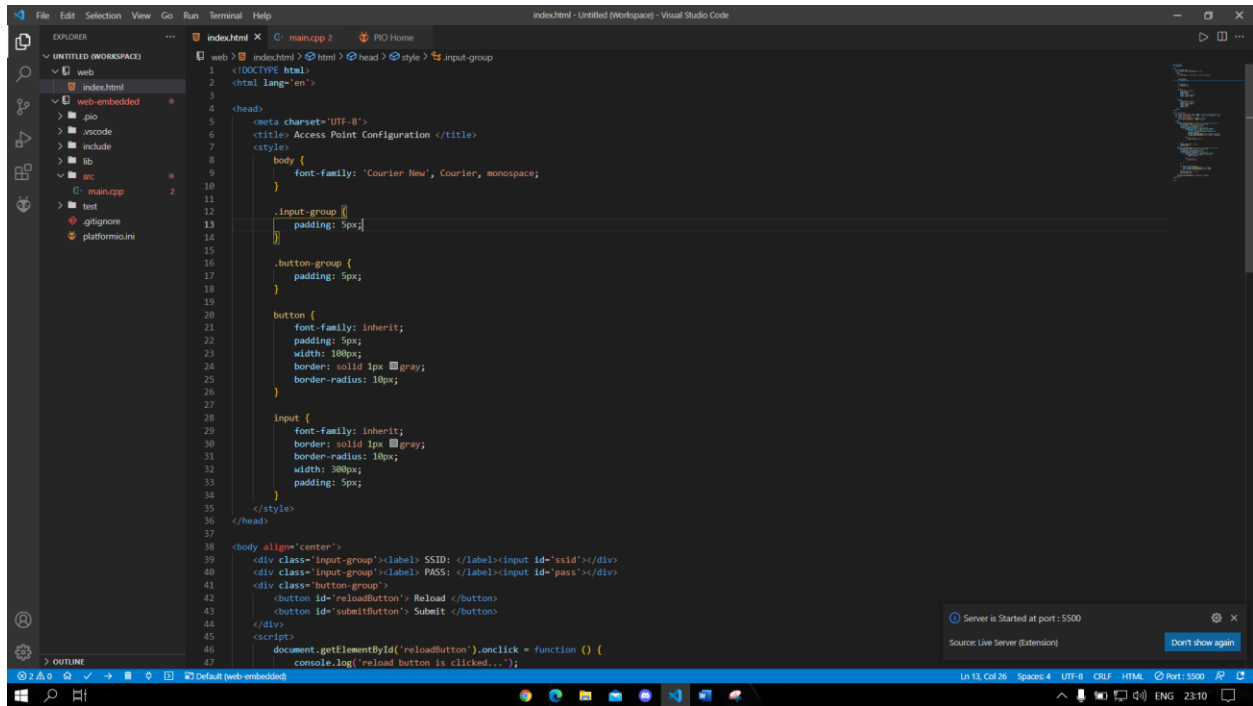


- จะมีการ reboots board 1 ครั้ง

ตย.

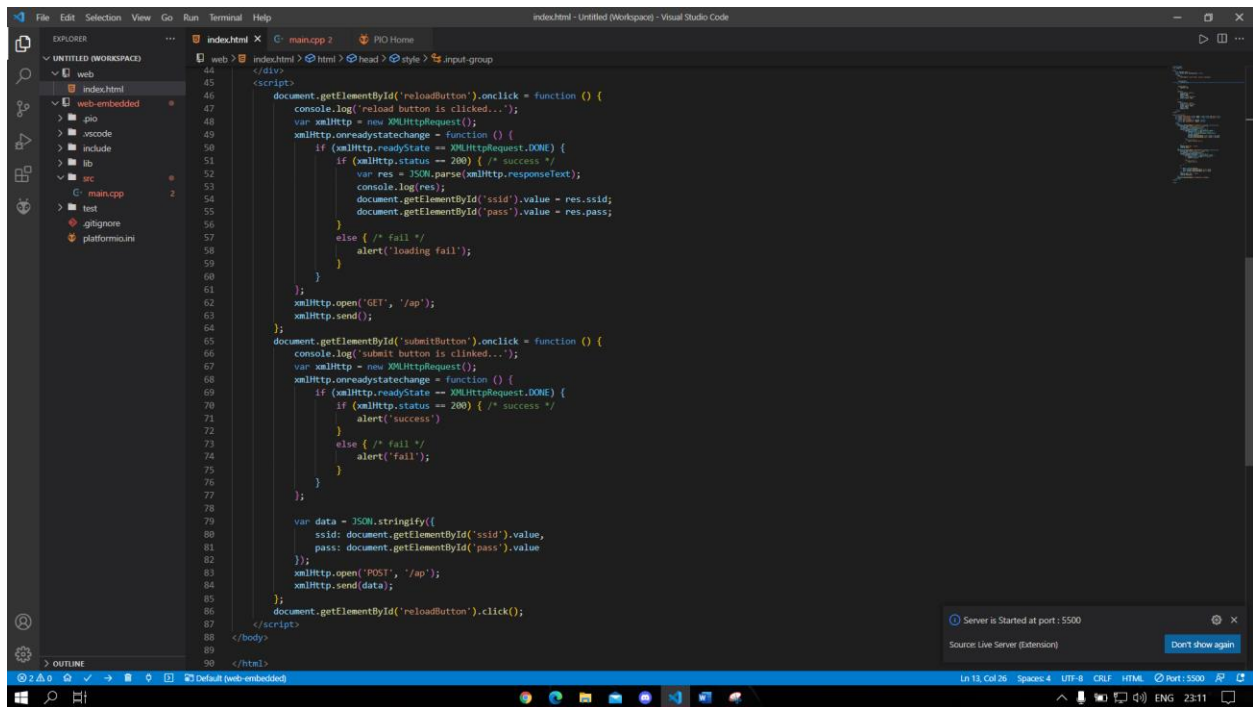
```
Connection Success..
IP address(STA mode) : 192.168.1.51
MDNS responder started
AccessPoint ssid: SPU_2G-[A4F544BF713C]
IP address(AccessPoint Mode) : 192.168.4.1
HTTP server started
```

# Code HTML



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with folders like 'web', 'web-embedded', 'pio', 'include', 'lib', 'src', 'test', 'platformio.ini', and 'main.cpp'. The code editor displays the initial HTML code for 'index.html'.

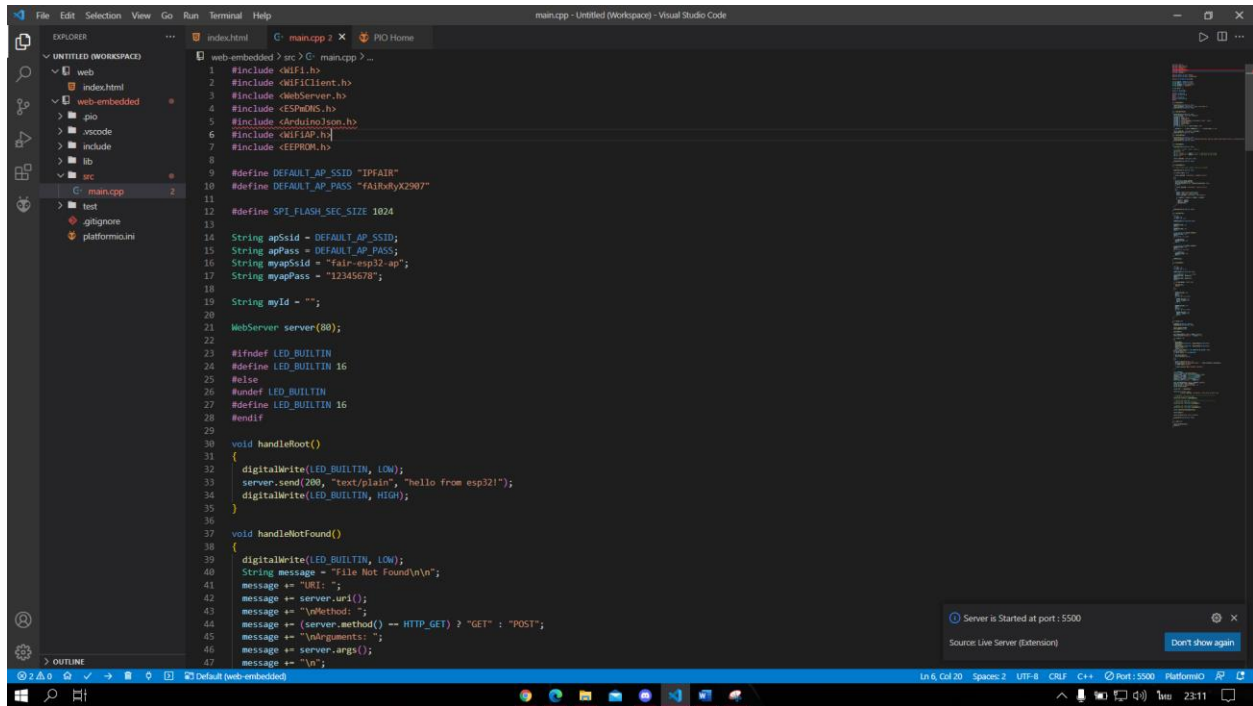
```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <title> Access Point Configuration </title>
7   <style>
8     body {
9       font-family: 'Courier New', Courier, monospace;
10    }
11
12    .input-group {
13      padding: 5px;
14    }
15
16    .button-group {
17      padding: 5px;
18    }
19
20    button {
21      font-family: inherit;
22      padding: 5px;
23      width: 100px;
24      border: solid 1px #gray;
25      border-radius: 10px;
26    }
27
28    input {
29      font-family: inherit;
30      border: solid 1px #gray;
31      border-radius: 10px;
32      width: 300px;
33      padding: 5px;
34    }
35  </style>
36 </head>
37
38 <body align="center">
39   <div class="input-group"><label> SSID: </label><input id="ssid"></div>
40   <div class="input-group"><label> PASS: </label><input id="pass"></div>
41   <div class="button-group">
42     <button id="reloadButton"> Reload </button>
43     <button id="submitButton"> Submit </button>
44   </div>
45   <script>
46     document.getElementById('reloadButton').onclick = function () {
47       console.log('reload button is clicked...');
```



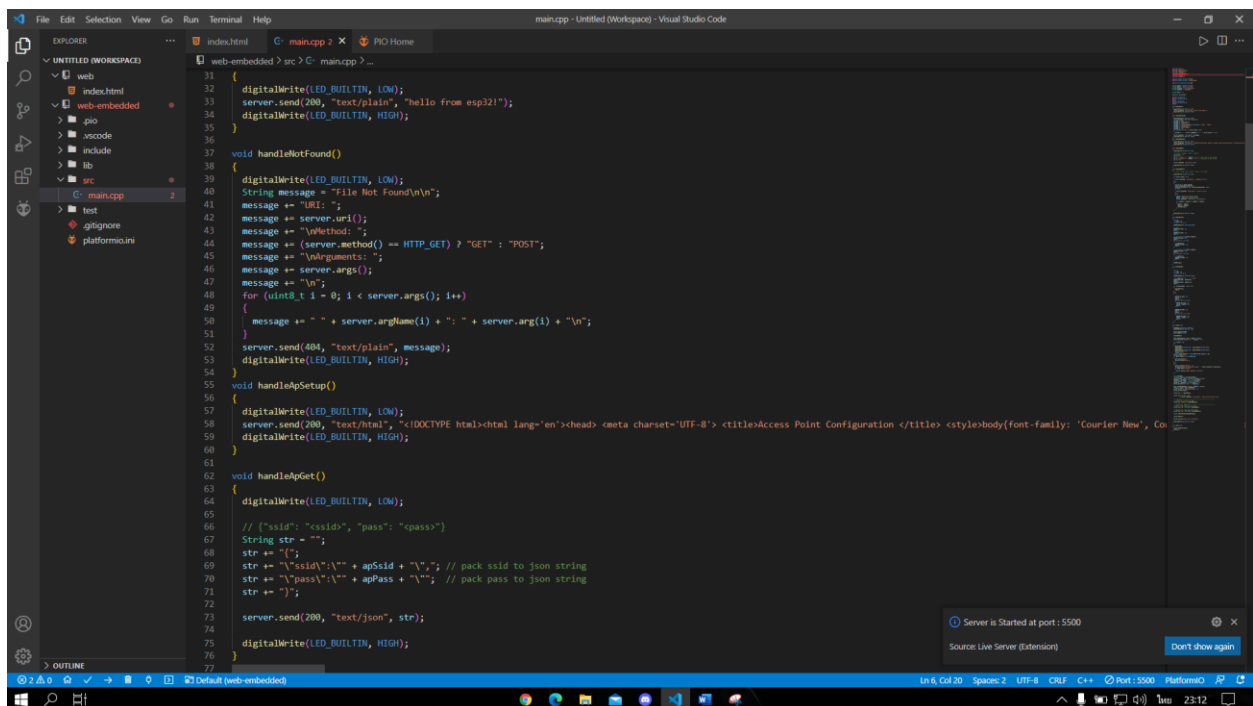
The screenshot shows the Visual Studio Code editor with the same file explorer on the left. The code editor displays the JavaScript code for 'index.html'.

```
44 </div>
45 </script>
46 document.getElementById('reloadButton').onclick = function () {
47   console.log('reload button is clicked...');
48   var xmlhttp = new XMLHttpRequest();
49   xmlhttp.onreadystatechange = function () {
50     if (xmlhttp.readyState == XMLHttpRequest.DONE) {
51       if (xmlhttp.status == 200) { /* success */
52         var res = JSON.parse(xmlhttp.responseText);
53         console.log(res);
54         document.getElementById('ssid').value = res.ssid;
55         document.getElementById('pass').value = res.pass;
56       }
57       else { /* fail */
58         alert('loading fail');
59       }
60     }
61   };
62   xmlhttp.open('GET', '/ap');
63   xmlhttp.send();
64 }
65
66 document.getElementById('submitButton').onclick = function () {
67   console.log('submit button is clicked...');
68   var xmlhttp = new XMLHttpRequest();
69   xmlhttp.onreadystatechange = function () {
70     if (xmlhttp.readyState == XMLHttpRequest.DONE) {
71       if (xmlhttp.status == 200) { /* success */
72         alert('success');
73       }
74       else { /* fail */
75         alert('fail');
76       }
77     }
78   };
79   var data = JSON.stringify({
80     ssid: document.getElementById('ssid').value,
81     pass: document.getElementById('pass').value
82   });
83   xmlhttp.open('POST', '/ap');
84   xmlhttp.send(data);
85 }
86 document.getElementById('reloadButton').click();
87 </script>
88 </body>
89 </html>
```

## Code Main ในส่วน IO



```
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 #include <WebServer.h>
4 #include <ESPmDNS.h>
5 #include <ArduinoJson.h>
6 #include <HTTP.h>
7 #include <EEPROM.h>
8
9 #define DEFAULT_AP_SSID "IPFAIR"
10 #define DEFAULT_AP_PASS "fAIRbHyX2907"
11
12 #define SPI_FLASH_SEC_SIZE 1024
13
14 String apSsid = DEFAULT_AP_SSID;
15 String apPass = DEFAULT_AP_PASS;
16 String myapSsid = "fair-esp32-ap";
17 String myapPass = "12345678";
18
19 String myId = "";
20
21 WebServer server(80);
22
23 #ifnndef LED_BUILTIN
24 #define LED_BUILTIN 16
25 #else
26 #undef LED_BUILTIN
27 #define LED_BUILTIN 16
28 #endif
29
30 void handleRoot()
31 {
32   digitalWrite(LED_BUILTIN, LOW);
33   server.send(200, "text/plain", "hello from esp32!");
34   digitalWrite(LED_BUILTIN, HIGH);
35 }
36
37 void handleNotFound()
38 {
39   digitalWrite(LED_BUILTIN, LOW);
40   String message = "File Not Found\n\n";
41   message += "URI: ";
42   message += server.uri();
43   message += "\nMethod: ";
44   message += (server.method() == HTTP_GET) ? "GET" : "POST";
45   message += "\nArguments: ";
46   message += server.args();
47   message += "\n";
48 }
```



```
31 {
32   digitalWrite(LED_BUILTIN, LOW);
33   server.send(200, "text/plain", "hello from esp32!");
34   digitalWrite(LED_BUILTIN, HIGH);
35 }
36
37 void handleNotFound()
38 {
39   digitalWrite(LED_BUILTIN, LOW);
40   String message = "File Not Found\n\n";
41   message += "URI: ";
42   message += server.uri();
43   message += "\nMethod: ";
44   message += (server.method() == HTTP_GET) ? "GET" : "POST";
45   message += "\nArguments: ";
46   message += server.args();
47   message += "\n";
48   for (uint8_t i = 0; i < server.args(); i++)
49   {
50     message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
51   }
52   server.send(404, "text/plain", message);
53   digitalWrite(LED_BUILTIN, HIGH);
54 }
55
56 void handleSetup()
57 {
58   digitalWrite(LED_BUILTIN, LOW);
59   server.send(200, "text/html", "<!DOCTYPE html><html lang='en'><head> <meta charset='UTF-8'><title>Access Point Configuration</title><style>body{font-family: 'Courier New', Co
60 }
61
62 void handleGet()
63 {
64   digitalWrite(LED_BUILTIN, LOW);
65   // ["ssid": "ssid", "pass": "pass"]
66   String str = "";
67   str += "[";
68   str += "\"ssid\": \"" + apSsid + "\", "; // pack ssid to json string
69   str += "\"pass\": \"" + apPass + "\", "; // pack pass to json string
70   str += "]";
71   server.send(200, "text/json", str);
72   digitalWrite(LED_BUILTIN, HIGH);
73 }
```

```
1 // web-embedded > src > C: main.cpp > ...
2 // digitalWrite(LED_BUILTIN, LOW);
3 // server.send(200, "text/html", "<DOCTYPE html><html lang='en'><head> <meta charset='UTF-8'> <title>Access Point Configuration </title> <style>body{font-family: 'Courier New', Co
4 // }
5
6 void handleHttpGet()
7 {
8     digitalWrite(LED_BUILTIN, LOW);
9
10    // ("ssid": "ssid", "pass": "pass")
11    String str = "";
12    str += "{";
13    str += "\"ssid\":\"" + apSsid + "\", // pack ssid to json string
14    str += "\"pass\":\"" + apPass + "\", // pack pass to json string
15    str += "}";
16
17    server.send(200, "text/json", str);
18
19    digitalWrite(LED_BUILTIN, HIGH);
20 }
21
22 void handleHttpPost()
23 {
24    // (ssid: <ssid>, pass: <pass>), args is 1 at arg(0)
25    digitalWrite(LED_BUILTIN, LOW);
26
27    if (server.args() != 1)
28    {
29        server.send(400, "text/plain", "argument error");
30    }
31    else
32    {
33        String str = server.arg(0);
34        StaticJsonDocument<100> doc;
35        DeserializationError err = deserializeJson(doc, str);
36        if (err)
37        {
38            server.send(500, "text/plain", "server error");
39        }
40        else
41        {
42            apSsid = doc["ssid"].asString();
43            apPass = doc["pass"].asString();
44            server.send(200, "text/plain", "OK Success");
45            if (_apSsid != apSsid || _apPass != apPass)
46            {
47                digitalWrite(LED_BUILTIN, HIGH);
48            }
49        }
50    }
51 }
```

```
88 // }
89 // else
90 // {
91 //     String str = server.arg(0);
92 //     StaticJsonDocument<100> doc;
93 //     DeserializationError err = deserializeJson(doc, str);
94 //     if (err)
95 //     {
96 //         server.send(500, "text/plain", "server error");
97 //     }
98 //     else
99 //     {
100 //         apSsid = doc["ssid"].asString();
101 //         apPass = doc["pass"].asString();
102 //         server.send(200, "text/plain", "OK Success");
103 //         if (_apSsid != apSsid || _apPass != apPass)
104 //         {
105 //             apSsid = _apSsid;
106 //             apPass = _apPass;
107 //             EEPROM.write();
108 //         }
109 //     }
110 // }
111
112 digitalWrite(LED_BUILTIN, HIGH);
113 }
114
115 void EEPROMWrite()
116 {
117     char c;
118     int addr = 0;
119     unsigned char s, i;
120
121     EEPROM.begin(SPI_FLASH_SEC_SIZE);
122
123     c = '0';
124     EEPROM.put(addr, c);
125     addr++;
126     c = '1';
127     EEPROM.put(addr, c);
128     addr++;
129
130     s = (unsigned char)apSsid.length();
131     EEPROM.put(addr, s);
132     addr++;
133     for (i = 0; i < s; i++)
```

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure with folders like 'web', 'web-embedded', 'pio', 'src', and 'test'. The main editor window shows the 'main.cpp' file with the following code:

```
131     EEPROM.put(addr, s);
132     addr++;
133     for (i = 0; i < s; i++)
134     {
135         c = apSsid[i];
136         EEPROM.put(addr, c);
137         addr++;
138     }
139
140     s = (unsigned char)apPass.length();
141     EEPROM.put(addr, s);
142     addr++;
143     for (i = 0; i < s; i++)
144     {
145         c = apPass[i];
146         EEPROM.put(addr, c);
147         addr++;
148     }
149
150     EEPROM.end();
151 }
152
153 void eepromRead()
154 {
155
156     char c;
157     int addr = 0;
158     unsigned char s, i;
159
160     EEPROM.begin(SPI_FLASH_SEC_SIZE);
161
162     char header[3] = {' ', ' ', '\0'};
163     EEPROM.get(addr, header[0]);
164     addr++;
165     EEPROM.get(addr, header[1]);
166     addr++;
167
168     if (strcmp(header, "g5") != 0)
169     {
170         eepromWrite();
171         return;
172     }
173     else
174     {
175         EEPROM.get(addr, s);
176     }
177 }
```

A notification at the bottom right states: "Server is Started at port : 5500" with a "Don't show again" button.

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure with folders like 'web', 'web-embedded', 'pio', 'src', and 'test'. The main editor window shows the 'main.cpp' file with the following code:

```
177     addr++;
178     apSsid = "";
179     for (i = 0; i < s; i++)
180     {
181         EEPROM.get(addr, c);
182         apSsid = apSsid + c;
183         addr++;
184     }
185
186     EEPROM.get(addr, s);
187     addr++;
188     apPass = "";
189     for (i = 0; i < s; i++)
190     {
191         EEPROM.get(addr, c);
192         apPass = apPass + c;
193         addr++;
194     }
195 }
196
197
198 void setup(void)
199 {
200     pinMode(LED_BUILTIN, OUTPUT);
201     digitalWrite(LED_BUILTIN, LOW);
202
203     Serial.begin(115200);
204     WiFi.mode(WIFI_STA);
205
206     eepromRead();
207
208     WiFi.begin(apSsid.c_str(), apPass.c_str());
209     Serial.println("Connect to " + apSsid + "");
210
211     int counter = 0;
212     do
213     {
214         delay(500);
215         digitalWrite(LED_BUILTIN, digitalRead(LED_BUILTIN));
216         delay(500);
217         digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
218         Serial.print(".");
219         counter += 1;
220     } while (WiFi.status() != WL_CONNECTED && counter < 10);
221     Serial.println("");
222     if (WiFi.status() != WL_CONNECTED)
223     {
224     }
```


The status bar at the bottom indicates: "Ln 6, Col 20 Spaces: 2 UTF-8 CRLF C++ Port: 5500 PlatformIO".



```
197 void setup(void)
198 {
199   pinMode(LED_BUILTIN, OUTPUT);
200   digitalWrite(LED_BUILTIN, LOW);
201
202   Serial.begin(115200);
203   WiFi.mode(WIFI_STA);
204
205   EEPROM.read();
206
207   WiFi.begin(apSsid.c_str(), apPass.c_str());
208   Serial.println("Connect to " + apSsid + "");
209
210   int counter = 0;
211   do
212   {
213     delay(500);
214     digitalWrite(LED_BUILTIN, digitalRead(LED_BUILTIN));
215     delay(500);
216     digitalWrite(LED_BUILTIN, digitalRead(LED_BUILTIN));
217     Serial.print("-");
218     counter++;
219   } while (WiFi.status() != WL_CONNECTED && counter < 10);
220   Serial.println("");
221   if (WiFi.status() != WL_CONNECTED)
222   {
223     WiFi.disconnect();
224     Serial.println("Fail");
225   }
226   else
227   {
228     Serial.println("Success...");
229     Serial.println("IP address(STA mode): " + WiFi.localIP().toString());
230     if (MDNS.begin("esp32"))
231     {
232       Serial.println("MDNS responder started");
233     }
234   }
235
236   char temp[10];
237   uint64_t chipid = ESP.getFuseMac();
238   sprintf(temp, "M0X", (uint16_t)(chipid >> 32));
239   myapSsid = myapSsid + "-" + String(temp);
240   sprintf(temp, "M0X", (uint32_t)chipid);
241   myapSsid = myapSsid + String(temp) + "-";
242   Serial.println("Ap ssid: " + myapSsid);
243 }
```

```
230   Serial.println("IP address(STA mode): " + WiFi.localIP().toString());
231   if (MDNS.begin("esp32"))
232   {
233     Serial.println("MDNS responder started");
234   }
235
236   char temp[10];
237   uint64_t chipid = ESP.getFuseMac();
238   sprintf(temp, "M0X", (uint16_t)(chipid >> 32));
239   myapSsid = myapSsid + "-" + String(temp);
240   sprintf(temp, "M0X", (uint32_t)chipid);
241   myapSsid = myapSsid + String(temp) + "-";
242   Serial.println("Ap ssid: " + myapSsid);
243
244   WiFi.softAP(myapSsid.c_str(), myapPass.c_str());
245   IPAddress myIP = WiFi.softAPIP();
246   Serial.print("IP address(AP Mode): ");
247   Serial.println(myIP);
248
249   server.on("/", handleRoot);
250
251   server.on("/inline", []()
252   {
253     { server.send(200, "text/plain", "this works as well!"); });
254   });
255
256   // front-end -----
257   server.on("/apSetup", handleApSetup);
258
259   // webservices (back-end) -----
260   // return ssid, pass in json format
261   server.on("/ap", HTTP_GET, handleApGet);
262
263   // update ssid, pass from web browser
264   server.on("/ap", HTTP_POST, handleApPost);
265
266   server.onNotFound(handleNotFound);
267
268   server.begin();
269
270   Serial.println("HTTP server started");
271
272   digitalWrite(LED_BUILTIN, HIGH);
273 }
274
275 void loop(void)
276 {
277 }
```


ส่วนเสริมที่ต้องใช้คร่าวๆที่ผมได้ลง



**C/C++**  
C/C++ IntelliSense, debugging, and code browsing.  
Microsoft

⌚ 471ms


⚙️



**HTML CSS Support**  
CSS Intellisense for HTML  
ecmel


⌚ 49ms

⚙️




**HTML Snippets**  
Full HTML tags including HTML5 Snippets  
Mohamed Abusaid

⚙️



**HTML Snippets**  
Full HTML tags including HTML5 Snippets  
Mohamed Abusaid


⚙️



**Live Server**  
Launch a development local Server with live reload feature for st...  
Ritwick Dey

⌚ 5668ms

⚙️



**PlatformIO IDE**  
Professional development environment for Embedded, IoT, Ardui...  
PlatformIO

⌚ 4293ms

⚙️