Tutorial

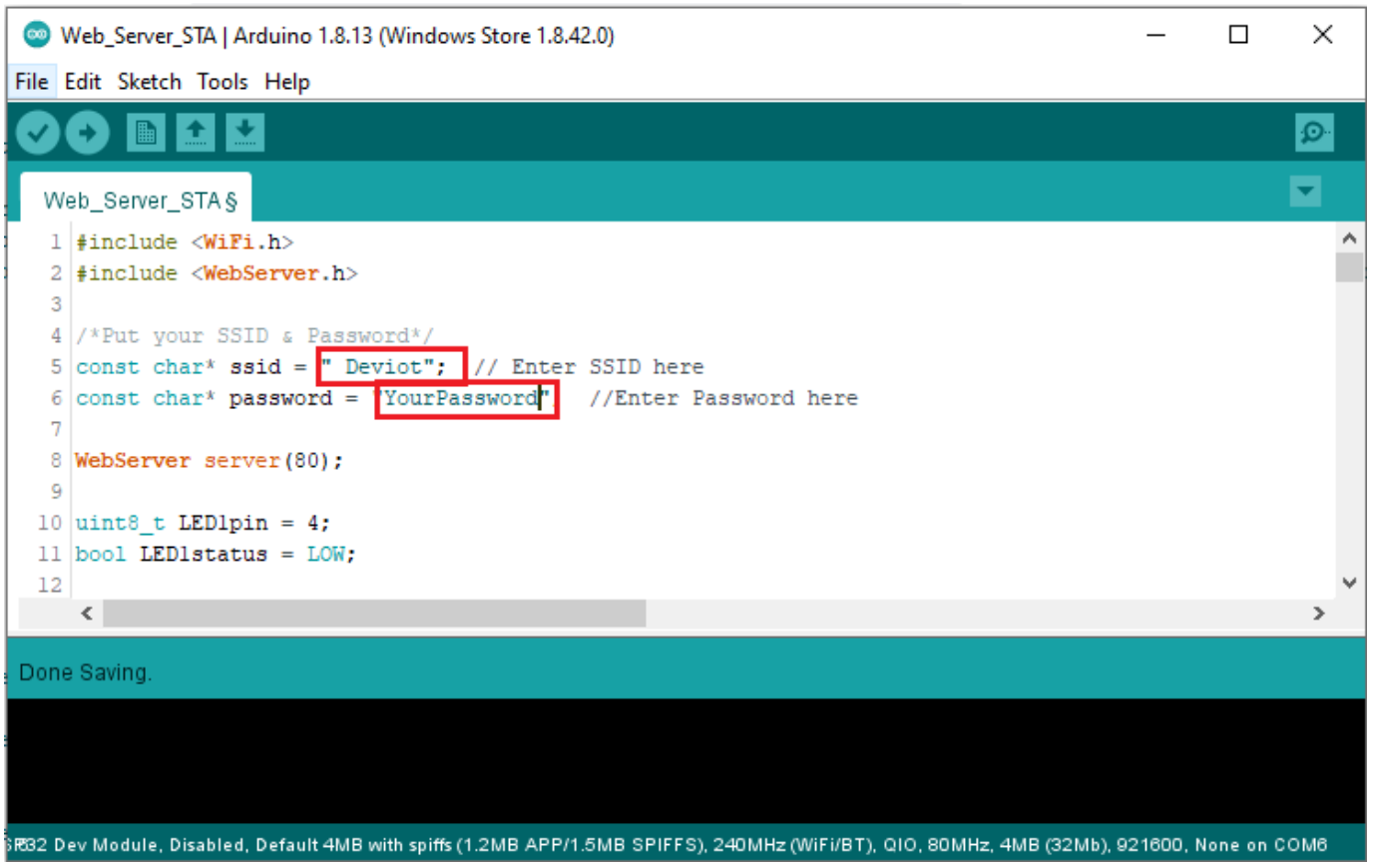
Series chia sẻ kiến thức lập trình

ESP32 WEB SERVER (Phần 4) Tuan Doan

ESP32 làm HTTP Server sử dụng chế độ Station (STA) điều khiển ngoại vi

Bây giờ chúng ta hãy chuyển sang ví dụ tiếp theo bằng cách dùng ESP32 ở chế độ Station (STA) và cung cấp các trang web cho bất kỳ ứng dụng khách nào được kết nối trong mạng hiện có.

Trước khi bắt đầu nạp code, bạn cần thực hiện một số thay đổi để làm cho nó phù hợp ví dụ này. Bạn cần sửa đổi hai biến *ssid* và *password* với thông tin đăng nhập mạng của mình để ESP32 có thể thiết lập kết nối với mạng hiện có.

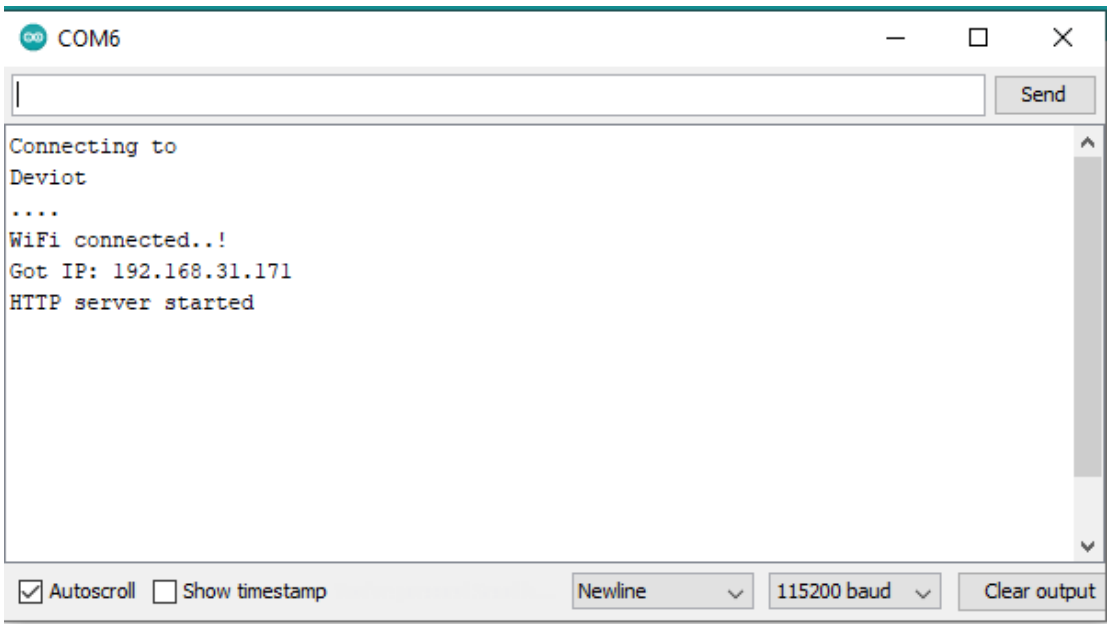


Kết nối ngoại vi với ESP32

Bây giờ chúng ta đã biết những điều cơ bản về cách máy chủ web hoạt động và ESP32 có thể tạo web server ở chế độ nào, đã đến lúc kết nối một số đèn LED với ESP32 mà chúng ta muốn kiểm soát qua WiFi.

Bắt đầu bằng cách đặt ESP32 vào breadboard của bạn, đảm bảo mỗi mặt của bảng nằm ở một phía riêng biệt của breadboard. Tiếp theo, kết nối hai đèn LED với GPIO 4 và 5 thông qua một điện trở 220Ω.

Mạch của chúng ta sẽ trông giống như hình bên dưới.

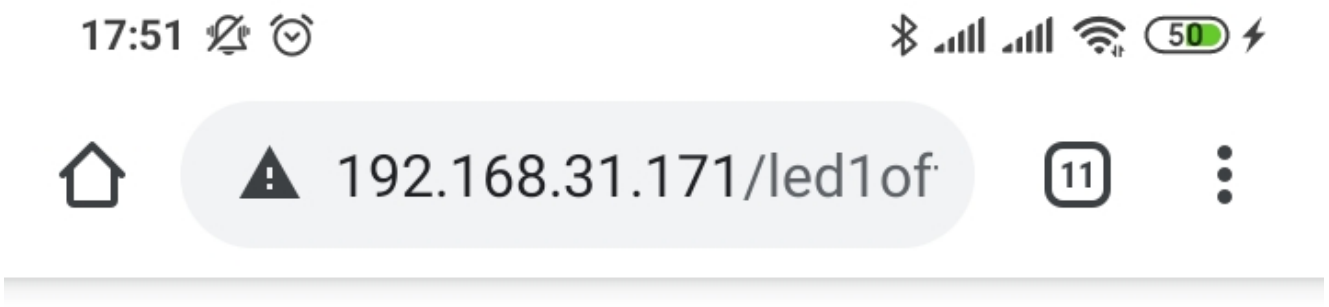


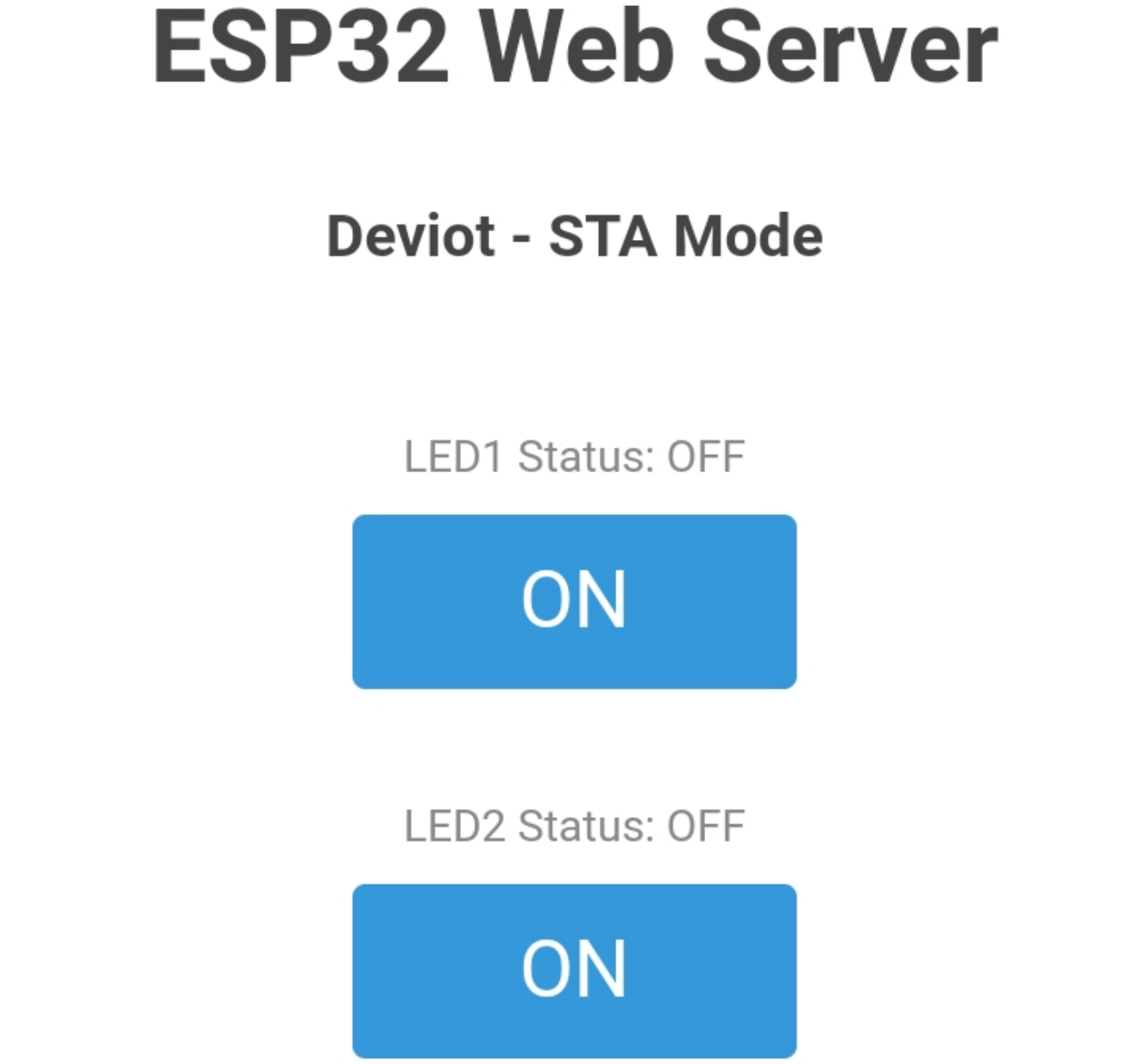
Code với Arduino IDE

```
/*Put your SSID & Password*/
const char* ssid = " Deviot"; // Your SSID here
const char* password = "12345678"; // Your Password here
WebServer server(80);
uint8_t LED1pin = 4;
bool LED1status = LOW;
uint8_t LED2pin = 5;
bool LED2status = LOW;

void setup() {
 Serial.begin(115200);
 delay(100);
 pinMode(LED1pin, OUTPUT);
 pinMode(LED2pin, OUTPUT);
 Serial.println("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.print(".");
 }
 Serial.println("WiFi connected..!");
 Serial.print("Got IP: "); Serial.println(WiFi.localIP());
 server.on("/", handle_OnConnect);
 server.on("/led1on", handle_led1on);
 server.on("/led1off", handle_led1off);
 server.on("/led2on", handle_led2on);
 server.on("/led2off", handle_led2off);
 server.onNotFound(handle_NotFound);
 server.begin();
 Serial.println("HTTP server started");
}
void loop() {
 server.handleClient();
 if(LED1status)  digitalWrite(LED1pin, HIGH);
 else            digitalWrite(LED1pin, LOW);
 if(LED2status)  digitalWrite(LED2pin, HIGH);
 else            digitalWrite(LED2pin, LOW);
}

void handle_OnConnect() {
 LED1status = LOW;
 LED2status = LOW;
 Serial.println("GPIO4 Status: OFF | GPIO5 Status: OFF");
 server.send(200, "text/html", SendHTML(LED1status,LED2status));
}

void handle_led1on() {
 LED1status = HIGH;
 Serial.println("GPIO4 Status: ON");
 server.send(200, "text/html", SendHTML(true,LED2status));
}
void handle_led1off() {
 LED1status = LOW;
 Serial.println("GPIO4 Status: OFF");
 server.send(200, "text/html", SendHTML(false,LED2status));
}
void handle_led2on() {
 LED2status = HIGH;
 Serial.println("GPIO5 Status: ON");
 server.send(200, "text/html", SendHTML(LED1status,true));
}
void handle_led2off() {
 LED2status = LOW;
 Serial.println("GPIO5 Status: OFF");
 server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
 server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led1stat,uint8_t led2stat){
 String ptr = "<!DOCTYPE html> <html>\n";
 ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
 ptr +="<title>LED Control</title>\n";
 ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
 ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom:
50px;}\n";
 ptr +=".button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 13px
30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
 ptr +=".button-on {background-color: #3498db;}\n";
 ptr +=".button-on:active {background-color: #2980b9;}\n";
 ptr +=".button-off {background-color: #34495e;}\n";
 ptr +=".button-off:active {background-color: #2c3e50;}\n";
 ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
 ptr +="</style>\n";
 ptr +="</head>\n";
 ptr +="<body>\n";
 ptr +="<h1>ESP32 Web Server</h1>\n";
 ptr +="<h3>Using Station(STA) Mode</h3>\n";

 if(led1stat)  ptr +="<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">OFF</a>\n";
```

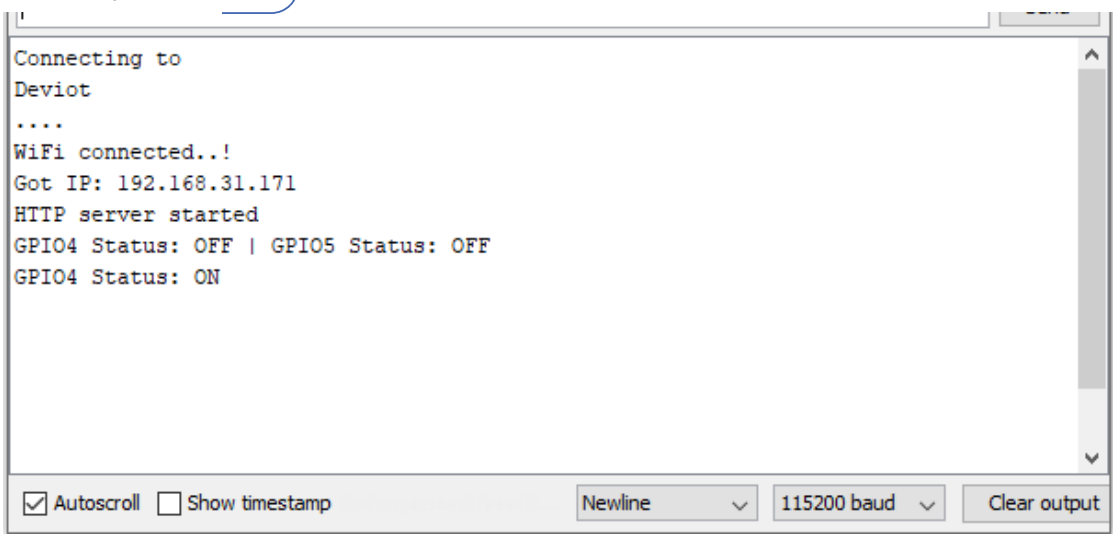
```
  else          ptr +="<p>LED2 Status: OFF</p><a class=\"button button-on\" href=\"/led2on\">ON</a>\n";

  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

Sau khi nạp code, hãy mở Serial Monitor với baudrate là 115200 baud. Và nhấn nút RESET trên ESP32. Serial sẽ in địa chỉ IP động mà router cấp cho ESP32 của bạn và hiển thị thông báo HTTP server started.



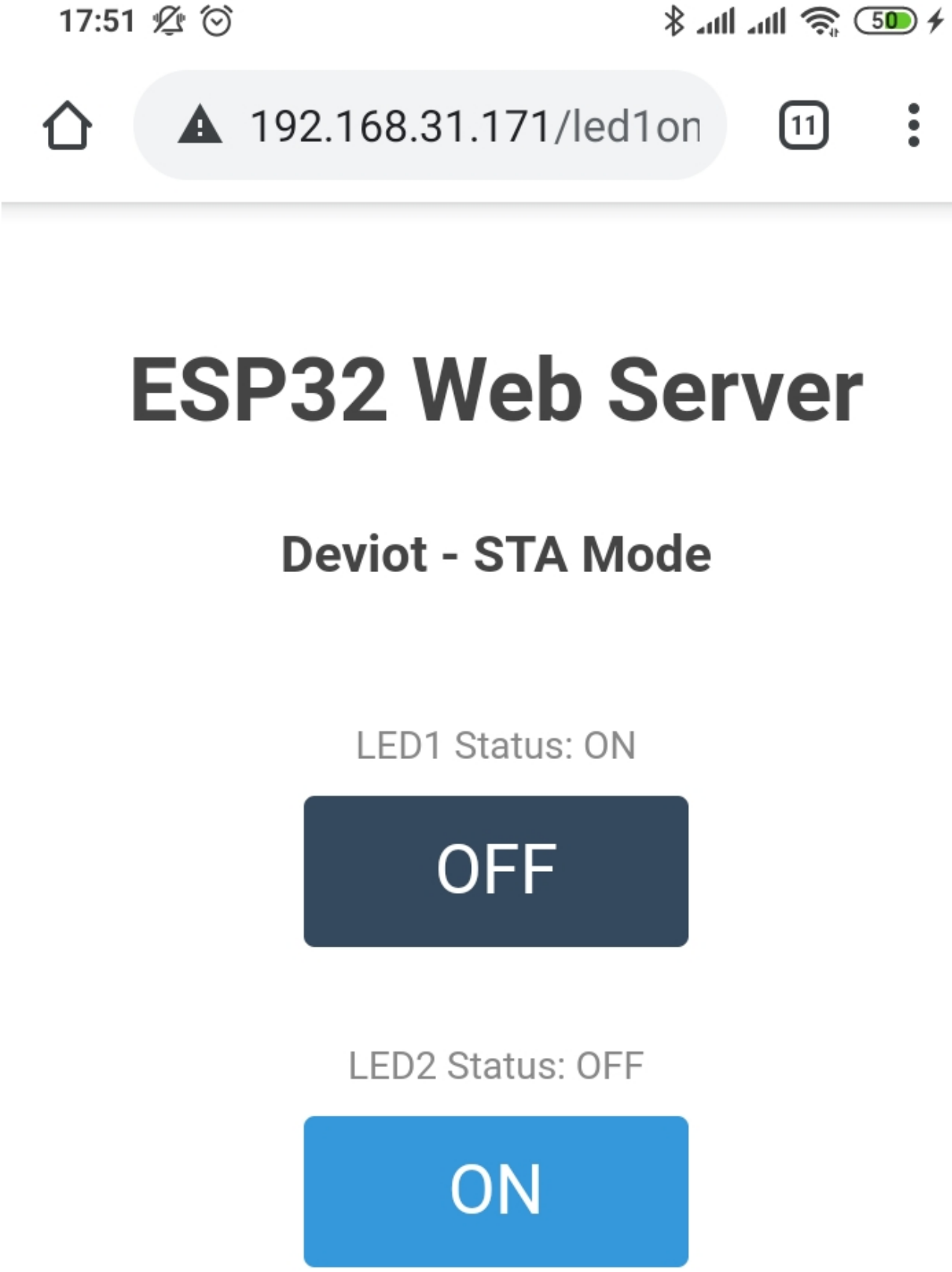


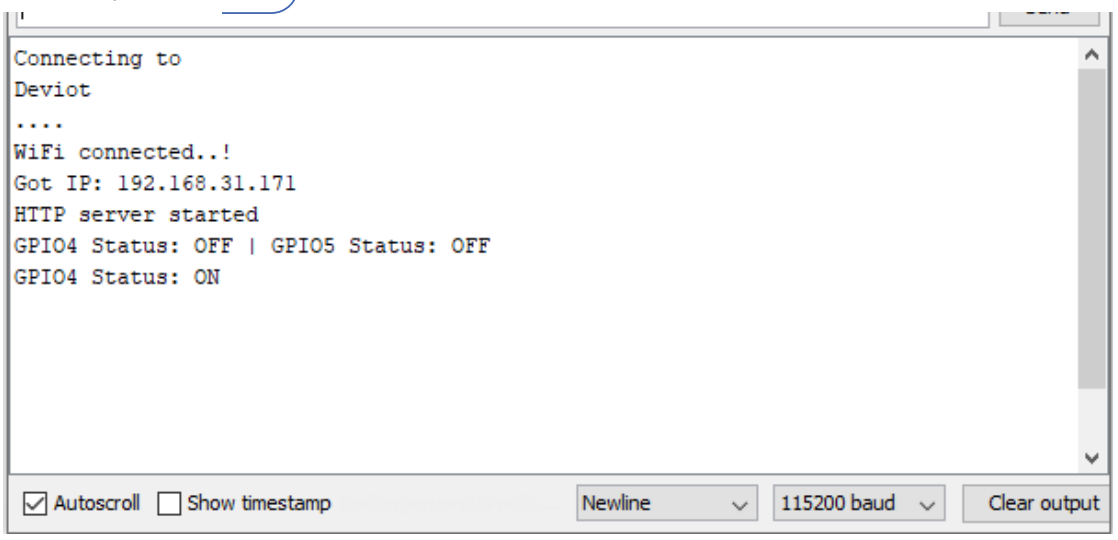
Tiếp theo, mở duyệt lên và trỏ nó đến địa chỉ IP hiển thị trên Serial Monitor vừa in ra. ESP32 sẽ cung cấp một trang web hiển thị trạng thái hiện tại của đèn LED và hai nút để điều khiển chúng. Nếu đồng thời nhìn vào Serial Monitor, bạn có thể thấy trạng thái của các chân GPIO của ESP32.



192.168.31.171 ở đây là IP mà Điểm truy cập (Access Point) cấp cho ESP32.

Bây giờ, hãy nhấn vào nút để BẬT LED1 trong khi theo dõi URL. Khi bạn nhấn vào nút ON/OFF, ESP32 nhận được yêu cầu cho URL/led1on . Sau đó, nó BẬT LED1 và cung cấp một trang web với trạng thái của đèn LED được cập nhật. Nó cũng in trạng thái của chân GPIO trên Serial Monitor.





Bạn có thể kiểm tra LED2 và kiểm tra xem nó có hoạt động theo cách tương tự.

Giải thích

Nếu bạn quan sát mã này với mã trước, sự khác biệt duy nhất là mình không thiết lập Điểm truy cập, Thay vào đó để ESP32 kết nối tới mạng hiện có bằng hàm WiFi.begin(). Tham số cần truyền vào là tên và mật khẩu Wi-Fi.

```
//connect to your local wi-fi network
WiFi.begin(ssid, password);
```

Trong khi ESP32 cố gắng kết nối với mạng, chúng ta có thể kiểm tra trạng thái kết nối bằng hàm *WiFi.status()*.

```
//check wi-fi is connected to wi-fi network
while (WiFi.status() != WL_CONNECTED)
{
 delay(1000);
 Serial.print(".");
}
```

Hàm này trả về các trạng thái sau:

- *WL_CONNECTED* : đã kết nối thành công Wi-Fi
- *WL_CONNECT_FAILED* : kết nối thất bại.
- *WL_CONNECTION_LOST* : mất kết nối.
- *WL_DISCONNECTED* : ngắt kết nối khỏi mạng.

Khi ESP32 được kết nối với mạng, ta sẽ in địa chỉ IP được gán cho ESP32 bằng cáchgọi hàm WiFi.localIP() và in giá trị này trên Serial Monitor.

```
Serial.println("");
Serial.println("WiFi connected..!");
Serial.print("Got IP: "); Serial.println(WiFi.localIP());
```

Sự khác biệt duy nhất giữa chế độ AP & STA là một chế độ tạo mạng và chế độ khác tham gia mạng hiện có. Vì vậy, phần còn lại của mã để xử lý các yêu cầu HTTP và phục vụ trang web ở chế độ STA giống như mã của chế độ AP được giải thích ở trên. Điều nay bao gồm:

- Khai báo các chân GPIO của ESP32 mà đèn LED được kết nối
- Định nghĩa phương thức server.on() để xử lý các HTTP request gửi đến
- Định nghĩa phương thức server.onNotFound() để xử lý mã lỗi HTTP 404
- Tạo các hàm tùy chỉnh được thực thi khi URL cụ thể được truy cập
- Tạo giao diện HTML
- Tạo kiểu cho trang web
- Tạo các nút nhấn và hiển thị trạng thái của chúng

Nếu không có LED và dây cắm các bạn có thể sử dụng GPIO 2 trên Board như Video dưới.

Chúc các bạn thành công!

Blog Nổi Bật

RTOS Phần 4: Queue

Series tutorial con trỏ trong lập trình C/C++ (cực hay)

RTOS Phần 3: Interrupt

RTOS Phần 8: Event Groups

Khóa Học Nổi Bật

Khóa học lập trình C

Lập trình STM32F103 cơ bản

Lập trình STM32 nâng cao

Lập trình IoT chip ESP32

Số 101C, ngõ Xã Đàn 2, Đống Đa, Hà Nội

0969666522

deviotcore@gmail.com

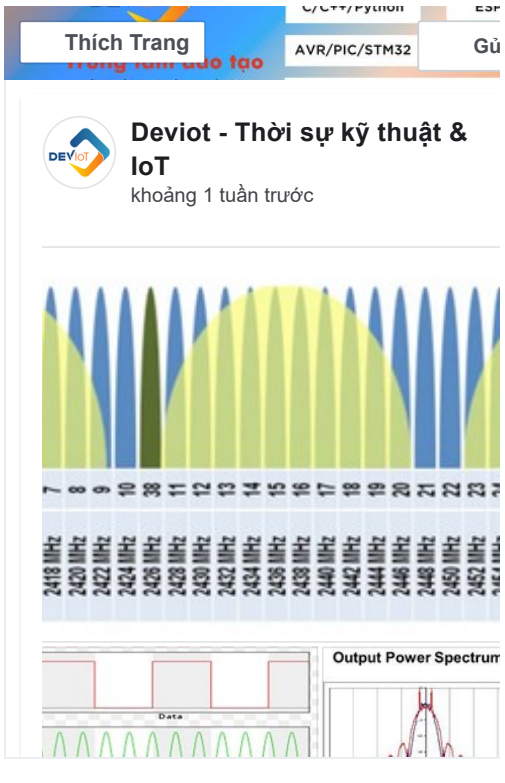
Chúng tôi là ai?

- DEVIOT là một trung tâm đào tạo về kỹ thuật được xây dựng bởi những kĩ sư giàu kinh nghiệm đến từ các công ty tập đoàn công nghệ nổi tiếng trong nước
- Chúng tôi mang trong mình sứ mệnh đem những kiến thức thực tế tích lũy được trong quá trình làm việc đến với các bạn học sinh, sinh viên trên cả nước

Chuyên ngành đào tạo

- Lập trình nhúng và IoT
- Thiết kế mạch điện tử
- Ngôn ngữ lập trình
- Xử lý ảnh và AI

Fanpage Facebook


Thích Trang
AVR/PIC/STM32
Deviot - Thời sự kỹ thuật & IoT
khoảng 1 tuần trước
Output Power Spectrum