



KHUÊ NGUYỄN CREATOR

Chia sẻ đam mê



## LẬP TRÌNH ESP32

# Bài 1: Lập trình ESP32 Webserver chế độ Wifi Station bật tắt Led

POSTED ON 09/07/2021 BY KHUÊ NGUYỄN

09  
Th7

## ESP32 và Platform IO



Khuê Nguyễn Creator



## Bài 1 WIFI: Lập trình ESP32 Webserver chế độ Station

Lập trình ESP32 Webserver là một bài cơ bản nhất để chúng ta có thể điều khiển thiết bị thông qua sóng wifi. Các thiết bị kết nối wifi như thế nào, cách làm việc với Web browser và Webserver như thế nào? Tất cả sẽ có trong bài viết ngày hôm nay.

Bài 1 Networking trong serie [Học Lập Trình ESP32 từ A tới Z](#)



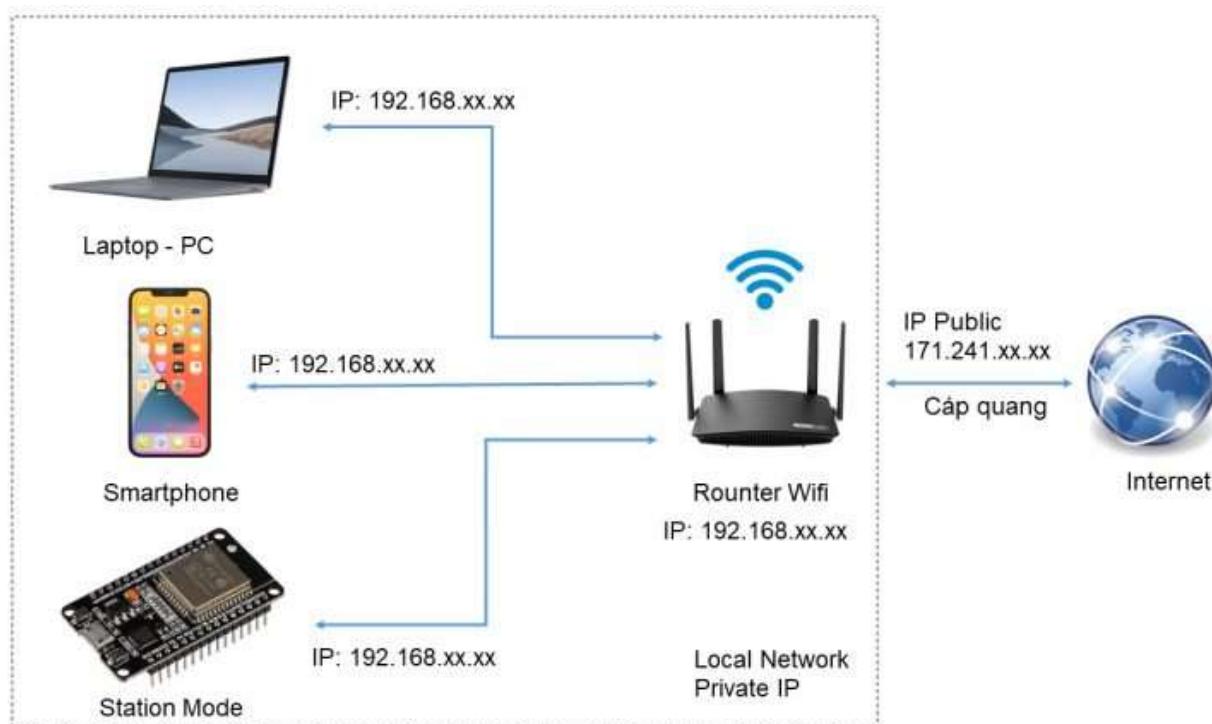
# Mục Lục

1. Chế độ Station trong kết nối wifi
2. Webserver là gì?
3. Lập trình ESP32 Webserver chế độ Station
  - 3.1. Sơ đồ nguyên lý
  - 3.2. Code và giải thích code ESP32 Webserver
    - 3.2.1. Trong Setup
    - 3.2.2. Trong loop()
    - 3.2.3. Trang trí trang web bằng CSS
    - 3.2.4. Tạo thân trang web với H1 và H2
    - 3.2.5. Thay đổi trạng thái nút nhấn trên web
  - 3.3. Nạp Code ESP32 Webserver và kiểm tra kết quả
  - 3.4. Kết Quả
4. Các hàm thường được sử dụng trong ESP32 Webserver chế độ Wifi Station
  - 4.1. Các hàm quản lý kết nối
    - 4.1.1. reconnect
    - 4.1.2. disconnect
    - 4.1.3. isConnected
    - 4.1.4. setAutoConnect
    - 4.1.5. getAutoConnect
    - 4.1.6. setAutoReconnect
    - 4.1.7. waitForConnectResult
  - 4.2. Các hàm cấu hình
    - 4.2.1. macAddress
    - 4.2.2. localIP
    - 4.2.3. subnetMask
    - 4.2.4. gatewayIP
    - 4.2.5. hostname
    - 4.2.6. status
    - 4.2.7. SSID
    - 4.2.8. psk – Mật khẩu
    - 4.2.9. BSSID
    - 4.2.10. RSSI
5. Kết
  - 5.1. Related posts:

# Chế độ Station trong kết nối wifi

Để kết nối các thiết bị wireless (như điện thoại thông minh, máy tính xách tay) hoặc có dây (như máy tính để bàn) vào trong mạng nội bộ (LAN) ta dùng một thiết bị gọi là Access Point (điểm truy cập).

Những thiết bị kết nối với Access Point được gọi là Station (trạm).



*ESP32 Webserver chế độ Station*

Trong gia đình các thiết bị router/modem có chức năng phát WiFi chính là một Access Point để các Station như điện thoại, máy tính kết nối vào. Sau đó các Router/Modem đó kết nối với internet thông qua mạng cáp quang.

Mỗi một Access Point có một SSID (Service Set Identifier) cụ thể hay chúng ta thường gọi là Tên wifi. Những Station kết nối với AP theo mô hình TCP/IP vậy nên, để phân biệt các Station và AP trong một mạng chúng ta sử dụng IP Address.

## Websvver là gì?

Webserver hay máy chủ web là một máy tính cài đặt các chương trình phục vụ các ứng dụng web. Webserver có khả năng tiếp nhận request từ các trình duyệt web và gửi phản hồi đến client thông qua giao thức HTTP hoặc các giao thức khác.

Bất cứ khi nào bạn xem một trang web trên internet, có nghĩa là bạn đang yêu cầu trang đó từ một web server.



Khi bạn nhập URL trên trình duyệt (web browser) như chrome, firefox thực chất nó sẽ làm các bước sau:

- Trình duyệt sẽ yêu cầu thông tin từ một hoặc nhiều máy chủ DNS (qua internet). Máy chủ DNS sẽ cho trình duyệt biết địa chỉ IP nào tên miền sẽ trả đến cũng là nơi đặt trang web.
- Máy chủ DNS sẽ chuyển hướng trình duyệt sang địa chỉ IP của máy chủ ứng với tên miền đó
- Máy chủ phản hồi cho trình duyệt những thông tin được yêu cầu: Thông thường đó là một trang HTML
- Trình duyệt render các file html, css, js... mà máy chủ truyền về thành một website hiển thị lên máy tính cho người dùng sử dụng

# Lập trình ESP32 Webserver chế độ Station

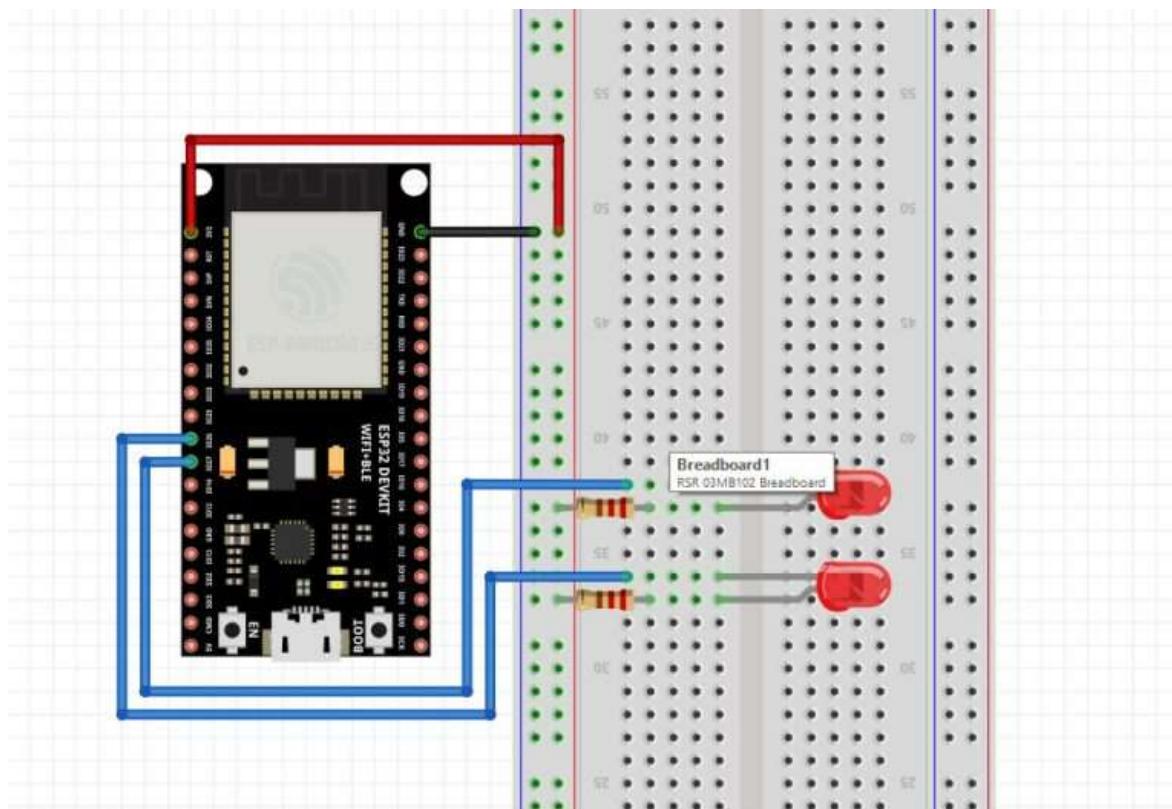
Trong bài này chúng ta sử dụng ESP32 tạo ra một webserver, lưu trữ file html. Trong đó có các nút nhấn để chúng ta điều khiển led bằng trình duyệt máy tính

Chuẩn bị:

- ESP32 development board
- 2x 5mm LED
- 2x 330 Ohm trở
- Breadboard
- Dây cắm

## Sơ đồ nguyên lý

Chúng ta kết nối 2 Led tới 2 chân 26 và 27. Hoặc bất kì chân GPIO nào khác có thể đặt làm OUTPUT. Chi tiết về các chân linh kiện có thể tham khảo bài viết: [Tổng quan về EPS32 DEV KIT](#)



*esp32 webserver chế độ wifi station*

## Code và giải thích code ESP32 Webserver

### Full Code

```
002 #include <Arduino.h>
003 //khai báo thư viện để kết nối wifi cho esp32 webserver
004 #include <WiFi.h>
005 //khai báo chân sử dụng led
006 const int led1 = 26;
007 const int led2 = 27;
008
009 const char* ssid = "TEN_WIFI";
010 const char* password = "MAT_KHAU";
011 //Tạo một web server tại cổng 80 - cổng mặc định cho esp32
012 WiFiServer webServer(80);
013
014 String led1Status = "OFF";
015 String led2Status = "OFF";
016
017 String header;
018
019 unsigned long currentTime = millis();
020 // Previous time
021 unsigned long previousTime = 0;
022 // Define timeout time in milliseconds (example: 2000ms = 2s)
023 const long timeoutTime = 2000;
024
025 void setup() {
026     Serial.begin(115200);
027
028     pinMode(led1, OUTPUT);
029     pinMode(led2, OUTPUT);
030     // Set outputs to LOW
031     digitalWrite(led1, LOW);
032     digitalWrite(led2, LOW);
033
034     Serial.print("Connecting to wifi");
035     Serial.println(ssid);
036     WiFi.begin(ssid, password);
037     while (WiFi.status() != WL_CONNECTED)
038     {
039         delay(500);
040         Serial.print(".");
041     }
042     Serial.println("");
043     Serial.println("WiFi connected.");
044     Serial.println("IP address: ");
```

```

045 Serial.println(WiFi.localIP());
046
047 //khởi tạo webserver
048 webServer.begin();
049 }
050
051 void loop() {
052 WiFiClient webClient = webServer.available(); // nếu có c:
053
054 if(webClient)
055 {
056 //khoi tao gia tri ban dau cho time
057 currentTime = millis();
058 previousTime = currentTime;
059 Serial.println("New web Client");
060 //biến lưu giá trị response
061 String currentLine = "";
062 //nếu có client connect và không quá thời gian time out
063 while(webClient.connected() && currentTime - previousTir
064 {
065 //đọc giá trị timer tại thời điểm hiện tại
066 currentTime = millis();
067 //nếu client còn kết nối
068 if(webClient.available())
069 {
070 //đọc giá trị truyền từ client theo từng byte kiểu
071 char c = webClient.read();
072 Serial.write(c);
073 header += c; // lưu giá trị vào Header
074 if(c == '\n') //Nếu đọc được kí tự xuống dòng (hết c
075 {
076 if (currentLine.length() == 0)
077 {
078 //esp32 webserver response
079 // HTTP headers luôn luôn bắt đầu với code HTTP
080 webClient.println("HTTP/1.1 200 OK");
081 webClient.println("Content-type:text/html"); //
082 webClient.println("Connection: close"); // kiểu
083 webClient.println();
084
085 // nếu trong file header có giá trị
086 if (header.indexOf("GET /led1/on") >= 0)
087 {
088 Serial.println("Led1 on");
089 led1Status = "on";
090 digitalWrite(led1, HIGH);
091 }
092 else if (header.indexOf("GET /led1/off") >= 0)
093 {
094 Serial.println("Led1 off");

```

```

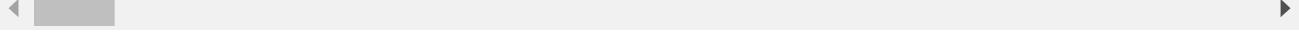
095
096
097
098
099
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
    led1Status = "off";
    digitalWrite(led1, LOW);
}
else if (header.indexOf("GET /led2/on") >= 0)
{
    Serial.println("Led2 on");
    led2Status = "on";
    digitalWrite(led2, HIGH);
}
else if (header.indexOf("GET /led2/off") >= 0)
{
    Serial.println("Led2 off");
    led2Status = "off";
    digitalWrite(led2, LOW);
}
// Response trang HTML của esp32 webserver
webClient.println("<!DOCTYPE html><html>");
webClient.println("<head><meta name=\"viewport\""
//thêm font-awesome
webClient.println("<link rel=\"stylesheet\" href="
// code CSS cho web
//css cho toàn bộ trang
    webClient.println("<img src=\"data:image/gif;ba"
// Web Page Heading H1 with CSS
webClient.println("<body><h1 style=\"color:Tomat"
// Web Page Heading H2
webClient.println("<h2 style=\"color:#077a39;\">"
webClient.println("<i class=\"fa fa-home\" aria-"
// Display current state, and ON/OFF buttons for
webClient.println("<p>Led1 - State " + led1Statu
// If the Led1Status is off, it displays the ON
if (led1Status=="off")
{
    //khởi tạo một nút nhấn có đường dẫn đích là ,
    webClient.println("<p><a href=\"/led1/on\"><b"
}
else
{
    //khởi tạo một nút nhấn có đường dẫn đích là ,
    webClient.println("<p><a href=\"/led1/off\"><b"
}
// Display current state, and ON/OFF buttons for
webClient.println("<p>Led2 - State " + led2Statu
// If the led2 is off, it displays the ON button
if (led2Status=="off")
{

```

```

145 //khởi tạo một nút nhấn có đường dẫn đích là ,
146 webClient.println("<p><a href=\"/led2/on\"><br>");
147 }
148 else
149 {
150 //khởi tạo một nút nhấn có đường dẫn đích là ,
151 webClient.println("<p><a href=\"/led2/off\"><br>");
152 }
153 webClient.println("</body></html>");

155 // The HTTP response ends with another blank line
156 webClient.println();
157 // Break out of the while loop
158 break;
159 }
160 else
161 {
162 currentLine = "";
163 }
164 }
165 else if (c != '\r') //nếu giá trị gửi tới khác xuống dòng
166 {
167 currentLine += c; //lưu giá trị vào biến
168 }
169 }
170 }
171 // Xoá header để sử dụng cho lần tới
172 header = "";
173 // ngắt kết nối với client trên esp32 webserver
174 webClient.stop();
175 Serial.println("Client disconnected.");
176 Serial.println("");
177
178 }
179 }
```



Trong chế độ station chúng ta cần kết nối nó với một mạng WIFI.

**Lưu ý: Để điều khiển được trong mạng Local thì máy tính (điện thoại) dùng để điều khiển ESP32 cũng cần phải kết nối vào cùng một mạng**

Thêm thư viện WiFi cho ESP32 `#include <WiFi.h>`

## Tạo 2 chuỗi lưu tên (SSID) và mật khẩu WIFI

```
//khai báo chân sử dụng led
const int led1 = 26;
const int led2 = 27;

const char* ssid = "TEN_WIFI";
const char* password = "MAT_KHAU";
```

Khởi tạo Webserver tại Port 80 (port cho webserver) `WiFiServer webServer(80);`

Khởi tạo biến lưu giá trị Client gửi lên: `String header;`

Khởi tạo các biến lưu trữ giá trị Timer:

```
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
```

## Trong Setup

Khởi tạo Serial và giá trị ban đầu cho Led

```
Serial.begin(115200);

pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
// Set outputs to LOW
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
```

Kết nối với mạng wifi của AP ( Router tại nhà)

```

Serial.print("Connecting to wifi");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

//khởi tạo webserver
webServer.begin();

```

## Trong loop()

Chúng ta lập trình esp32 webserver sẽ làm gì mỗi khi một client (trình duyệt của chúng ta) kết nối với nó,

ESP32 luôn luôn lắng nghe sự kết nối từ Client:

```
WiFiClient webClient = webServer.available();
```

Nếu Client vẫn connect chúng ta sẽ lưu giá trị mà Client yêu cầu vào biến header

Khi nhận được toàn bộ yêu cầu, Server sẽ phản hồi code 200 (OK)

```

//nếu có client connect và không quá thời gian time out
while(webClient.connected() && currentTime - previousTime <= timeoutTime)
{
    //đọc giá trị timer tại thời điểm hiện tại
    currentTime = millis();
    //nếu client còn kết nối
    if(webClient.available())
    {
        //đọc giá trị truyền từ client theo từng byte kiểu char
        char c = webClient.read();

```

```

Serial.write(c);
header += c; // lưu giá trị vào Header
if(c == '\n') //Nếu đọc được kí tự xuống dòng (hết chuỗi truyền tới)
{
    if (currentLine.length() == 0)
    {
        // HTTP headers luôn luôn bắt đầu với code HTTP (ví d HTTP/1.1 200 OK)
        webClient.println("HTTP/1.1 200 OK");
        webClient.println("Content-type:text/html"); // sau đó là kiểu nội dung
        webClient.println("Connection: close"); // kiểu kết nối ở đây là close.
        webClient.println();
    }
}

```

Tiếp đến chúng ta kiểm tra trong chuỗi nhận được có chuỗi GET + đường dẫn (tương ứng với button được nhấn) và thay đổi trạng thái led theo chuỗi đó

```

// nếu trong file header có giá trị
if (header.indexOf("GET /led1/on") >= 0)
{
    Serial.println("Led1 on");
    led1Status = "on";
    digitalWrite(led1, HIGH);
}
else if (header.indexOf("GET /led1/off") >= 0)
{
    Serial.println("Led1 off");
    led1Status = "off";
    digitalWrite(led1, LOW);
}
else if (header.indexOf("GET /led2/on") >= 0)
{
    Serial.println("Led2 on");
    led2Status = "on";
    digitalWrite(led2, HIGH);
}
else if (header.indexOf("GET /led2/off") >= 0)
{
    Serial.println("Led2 off");
    led2Status = "off";
}

```

```

        digitalWrite(led2, LOW);
    }

```

Điều tiếp theo bạn cần làm là tạo trang web. **ESP32** sẽ gửi phản hồi đến trình duyệt của bạn với một số mã HTML để xây dựng trang web. Trang web được gửi đến ứng dụng khách bằng cách sử dụng hàm `client.println()` này. Với tham số truyền vào là một chuỗi.

Điều đầu tiên chúng ta nên gửi luôn là dòng sau, điều đó cho biết rằng chúng ta đang gửi HTML.

```
webClient.println("<!DOCTYPE html><html>");
```

Sau đó, dòng sau làm cho trang web phản hồi trong bất kỳ trình duyệt web nào (Resize kích thước)

```
webClient.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
```

Thêm thư viện để chúng ta sử dụng ICON

```
webClient.println(
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">");
```

## Trang trí trang web bằng CSS

Thêm nút nhấn và style cho nút nhấn

Định nghĩa màu sắc, kiểu chữ cho các loại Text trên Web

```

webClient.println("<style>html { font-family: Helvetica; display: inline-block;
//css cho nut nhan
webClient.println(".button { background-color: #4CAF50; border: none; color: white;
webClient.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;
webClient.println(".button2 {background-color: #5e0101;}</style></head>");
```

## Tạo thân trang web với H1 và H2

```
// Web Page Heading H1 with CSS
webClient.println("<body><h1 style=\"color:Tomato;\">"ESP32 Station Web Server</h1>");

// Web Page Heading H2
webClient.println("<h2 style=\"color:#077a39;\">"<a href=\"https://khuenguyencrea
webClient.println("<i class=\"fa fa-home\" aria-hidden=\"true\"></i>");
```



## Thay đổi trạng thái nút nhấn trên web

Trong dòng này chúng ta sẽ điều hướng web tới 1 trang đích bằng href = . Trong trang đích đó. Thay đổi kiểu nút nhấn Button thành Button 2. Và thay đổi Text ghi trạng thái nút

```
// Display current state, and ON/OFF buttons for Led1
webClient.println("<p>Led1 - State " + led1Status + "</p>");
// If the Led1Status is off, it displays the ON button
if (led1Status=="off")
{
    //khởi tạo một nút nhấn có đường dẫn đích là /led1/on
    webClient.println("<p><a href=\"/led1/on\"><button class=\"button\">ON</button></a>" + "</p>");
}
else
{
    //khởi tạo một nút nhấn có đường dẫn đích là /led1/off
    webClient.println("<p><a href=\"/led1/off\"><button class=\"button button2\">OFF</button></a>" + "</p>");
}

// Display current state, and ON/OFF buttons for Led2
webClient.println("<p>Led2 - State " + led2Status + "</p>");
// If the led2 is off, it displays the ON button
if (led2Status=="off")
{
    //khởi tạo một nút nhấn có đường dẫn đích là /led2/on
    webClient.println("<p><a href=\"/led2/on\"><button class=\"button\">ON</button></a>" + "</p>");
}
else
{
    //khởi tạo một nút nhấn có đường dẫn đích là /led2/on
    webClient.println("<p><a href=\"/led2/on\"><button class=\"button button2\">OFF</button></a>" + "</p>");
}
```

```

webClient.println("<p><a href=\"/led2/off\"><button class=\"button button2\">0
}
webClient.println("</body></html>");
```

Cuối cùng đóng kết nối tới Client và xóa Header chuẩn bị cho các lần kết nối tiếp theo.

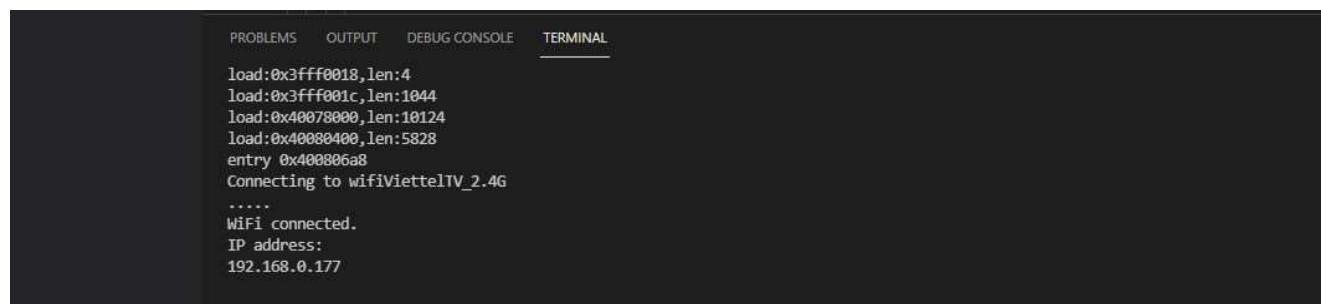
```

// Xoá header để sử dụng cho lần tới
header = "";
// ngắt kết nối
webClient.stop();
Serial.println("Client disconnected.");
Serial.println("");
```

## Nạp Code ESP32 Webserver và kiểm tra kết quả

Nhấn Build và Upload code esp32 webserver

Mở serial để xem IP mà ESP32 được cấp phát trong mạng.



The screenshot shows the VS Code interface with the terminal tab selected. The output in the terminal is as follows:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8
Connecting to wifiViettelTV_2.4G
.....
WiFi connected.
IP address:
192.168.0.177
```

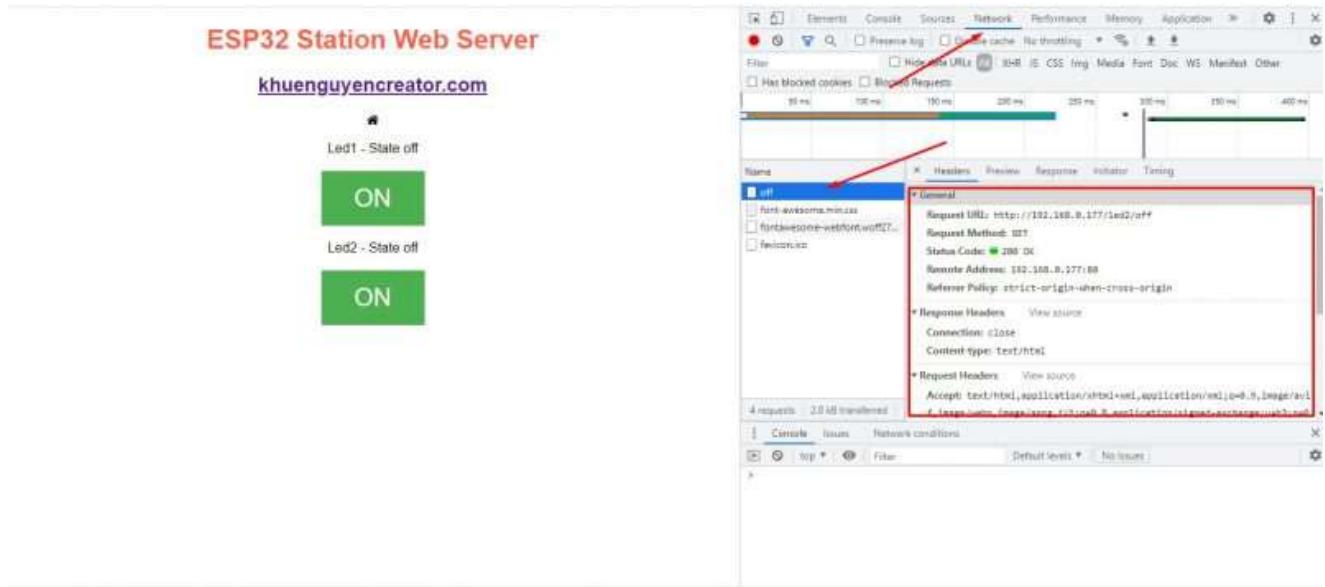
Dùng trình duyệt kết nối tới ESP32 thông qua ip đó.

Nhấn nút nhấn để thay đổi trạng thái led



Ta thấy rằng, mỗi khi nhấn button, led sẽ sáng tắt theo mong muốn. Thực chất mỗi khi bấm button, trình duyệt sẽ lại 1 lần GET dữ liệu từ ESP32 webserver, với đường dẫn thay đổi theo trạng thái button.

Nhấn **F12** trong trình duyệt Chrome để vào mode nhà phát triển. Chuyển qua tab Network. Refresh trang và nhấn nút nhấn. Chúng ta sẽ đọc được những gì trình duyệt gửi lên esp32 webserver và những gì esp32 webserver truyền xuống



## Kết Quả



Facebook Watch

## Các hàm thường được sử dụng trong ESP32 Webserver chế độ Wifi Station

### Các hàm quản lý kết nối

#### reconnect

Điều này được thực hiện bằng cách ngắt kết nối sau đó thiết lập kết nối lại đến cùng một điểm truy cập.

```
WiFi.reconnect()
```

Ví dụ:

```
WiFi.reconnect();
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
```

#### disconnect

Thiết lập cấu hình ssid và password thành null và thực hiện ngắt kết nối đến điểm truy cập.

```
WiFi.disconnect(wifioff)
```

Đối số wifioff là tham số tùy chọn kiểu boolean, nếu là true thì chế độ trạm (station mode) sẽ bị tắt.

## isConnected

Trả về true nếu Station kết nối với một điểm truy cập hoặc false nếu không.

```
WiFi.isConnected()
```

## setAutoConnect

Định cấu hình module để tự động kết nối khi cấp nguồn đến điểm truy cập cuối cùng được sử dụng.

```
WiFi.setAutoConnect(autoConnect)
```

autoConnect đây là tham số tùy chọn. Nếu đặt là false thì chức năng kết nối tự động sẽ bị tắt, nếu là true hoặc bỏ qua thì kết nối tự động sẽ được kích hoạt.

## getAutoConnect

Đây là chức năng đi đôi với setAutoConnect(). Nó trả về true nếu module được cấu hình để tự động kết nối với điểm truy cập được sử dụng lần cuối khi bật nguồn.

```
WiFi.getAutoConnect()
```

Trả về false nếu chức năng tự động kết nối bị vô hiệu.

## setAutoReconnect

Thiết đặt cho module tự động kết nối lại với một điểm truy cập trong trường hợp nó bị ngắt kết nối.

```
WiFi.setAutoReconnect(autoReconnect)
```

Nếu tham số autoReconnect được đặt thành true, thì module sẽ cố gắng thiết lập lại kết nối bị mất với AP. Nếu thiết lập để false module sẽ không thực hiện kết nối lại.

Lưu ý:

Chạy setAutoReconnect(true) khi module đã bị ngắt kết nối sẽ không kết nối lại với điểm truy cập. Thay vào đó reconnect() nên sử dụng.

## waitForConnectResult

Chờ cho đến khi module kết nối với điểm truy cập. Chức năng này dành cho các module được cấu hình trong chế độ STA hoặc STA + AP

```
WiFi.waitForConnectResult()
```

Chức năng trả về một trong các trạng thái kết nối sau đây:

- WL\_CONNECTED – Sau khi kết nối thành công được thiết lập
- WL\_NO\_SSID\_AVAIL – Trong trường hợp cấu hình SSID không thể đạt được
- WL\_CONNECT\_FAILED – Nếu mật khẩu không chính xác
- WL\_IDLE\_STATUS – Khi WiFi đang trong quá trình thay đổi giữa các trạng thái
- WL\_DISCONNECTED – Nếu module không được cấu hình trong chế độ station

## Các hàm cấu hình

### macAddress

Lấy địa chỉ MAC của ESP station

```
WiFi.macAddress(mac)
```

Với mac đó là một con trỏ đến vị trí bộ nhớ (một mảng uint8\_t có 6 phần tử) để lưu địa chỉ mac. Cùng một giá trị con trỏ được trả về bởi chính hàm đó.

Ví dụ:

```
if (WiFi.status() == WL_CONNECTED)
{
    uint8_t macAddr[6];
    WiFi.macAddress(macAddr);
    Serial.printf("Connected, mac address: %02x:%02x:%02x:%02x:%02x:%02x\n", macAd
}
```

Nếu bạn không muốn sử dụng con trỏ, bạn có thể dùng lệnh dưới, nó trả về một định dạng String chứa địa chỉ mac:

```
WiFi.macAddress() Ví Dụ:
```

```
if (WiFi.status() == WL_CONNECTED)
{
    Serial.printf("Connected, mac address: %s\n", WiFi.macAddress().c_str());
```

### localIP

Chức năng dùng để lấy địa chỉ IP của ESP station

```
WiFi.localIP()
```

Kiểu trả về là đại chỉ IP của module ESP32

```
if (WiFi.status() == WL_CONNECTED)
{
    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}
```

OutPut:

```
Connected, IP address: 192.168.1.10
```

## subnetMask

Trả về subnet mask của ESP station

```
WiFi.subnetMask()
```

Module nên được kết nối với điểm truy cập. Nếu không sẽ trả về 0.0.0.0

```
Serial.print("Subnet mask: ");
Serial.println(WiFi.subnetMask());
```

Output:

```
Subnet mask: 255.255.255.0
```

## gatewayIP

## Lấy địa chỉ IP của gateway

```
WiFi.gatewayIP()
```

VD:

```
Serial.printf("Gataway IP: %s\n", WiFi.gatewayIP().toString().c_str());
```

Output:

```
Gateway IP: 192.168.1.9
```

## hostname

Lấy DHCP hostname được gán cho ESP station.

```
WiFi.hostname()
```

Trả về kiểu String. Tên máy chủ mặc định ở định dạng ESP\_24xMAC với 24xMAC là 24 bit cuối cùng của địa chỉ MAC của module.

Tên máy chủ có thể được thay đổi bằng cách sử dụng chức năng sau:

```
WiFi.hostname(aHostname)
```

Tham số đầu vào aHostname có thể là một kiểu char\*, const char\* hoặc String. Chiều dài tối đa của tên máy chủ được chỉ định là 32 ký tự. Chức năng trả về true hoặc false phụ thuộc vào kết quả.

Ví dụ, nếu giới hạn 32 ký tự vượt quá, chức năng sẽ trả lại false mà không gán tên máy chủ mới.

Ví dụ:

```
Serial.printf("Default hostname: %s\n", WiFi.hostname().c_str());  
WiFi.hostname("Station_Tester_02");  
Serial.printf("New hostname: %s\n", WiFi.hostname().c_str());
```

output:

```
Default hostname: ESP_081117  
New hostname: Station_Tester_02
```

## status

Trả về trạng thái kết nối Wi-Fi.

```
WiFi.status()
```

Chức năng trả về một trong các trạng thái kết nối sau đây:

- WL\_CONNECTED – Sau khi kết nối thành công được thiết lập
- WL\_NO\_SSID\_AVAIL – Trong trường hợp cấu hình SSID không thể đạt được
- WL\_CONNECT\_FAILED – Nếu mật khẩu không chính xác
- WL\_IDLE\_STATUS – Khi Wi-Fi đang trong quá trình thay đổi giữa các trạng thái
- WL\_DISCONNECTED – Nếu module không được cấu hình trong chế độ trạm

Giá trị trả lại kiểu wl\_status\_t được định nghĩa trong wl\_definitions.h

Ví dụ:

```

void setup(void)
{
    Serial.begin(115200);
    Serial.printf("Connection status: %d\n", WiFi.status());
    Serial.printf("Connecting to %s\n", ssid);
    WiFi.begin(ssid, password);
    Serial.printf("Connection status: %d\n", WiFi.status());
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.printf("\nConnection status: %d\n", WiFi.status());
    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {}

```

Các trạng thái kết nối đặc biệt 6 và 3 có thể được xem xét trong wl\_definitions.h như sau:

```

3 - WL_CONNECTED
6 - WL_DISCONNECTED

```

Dựa trên ví dụ này, khi chạy trên mã, mô-đun ban đầu bị ngắt kết nối khỏi mạng và trả về trạng thái kết nối 6 – WL\_DISCONNECTED. Nó cũng bị ngắt kết nối ngay sau khi chạy WiFi.begin(ssid, password). Sau đó, sau khoảng 3 giây (dựa trên số dấu chấm được hiển thị mỗi 500ms), cuối cùng nó sẽ được kết nối trở lại trạng thái 3 – WL\_CONNECTED.

## SSID

Trả lại tên của mạng Wi-Fi đã kết nối.

```
WiFi.SSID()
```

## Kiểu trả về String

```
Serial.printf("SSID: %s\n", WiFi.SSID().c_str());
```

output:

```
SSID: sensor-net
```

## psk – Mật khẩu

Trả lại mật khẩu hiện tại của WiFi mà module đã kết nối tới:

```
WiFi.psk()
```

## Kiểu trả về String

Ví dụ:

```
void setup()
{
    Serial.begin(115200);
    Serial.println();
    Serial.printf("pass: %s ", WiFi.psk().c_str() );
}
void loop() {}
```

output:

```
pass: khuenguyencreator.com
```

## BSSID

Trả lại địa chỉ mac điểm truy cập mà ESP kết nối đến.

```
WiFi.BSSID()
```

Trả về một con trỏ đến vị trí nhớ (một mảng uint8\_t với có kích thước là 6), nơi BSSID được lưu.

Hàm dưới đây có chức năng tương tự, nhưng trả lại BSSID là một kiểu String.

```
WiFi.BSSIDstr()
```

```
Serial.printf("BSSID: %s\n", WiFi.BSSIDstr().c_str());
```

output:

```
BSSID: 00:1A:70:DE:C1:68
```

## RSSI

Trả lại cường độ tín hiệu của mạng Wi-Fi.

```
WiFi.RSSI()
```

Giá trị cường độ tín hiệu được cung cấp trong dBm. Kiểu trả về giá trị là int32\_t.

```
Serial.printf("RSSI: %d dBm\n", WiFi.RSSI());
```

output:

```
RSSI: -68 dBm
```

# Kết

ESP32 Webserver được sử dụng chủ yếu để thu thập dữ liệu từ các client khác. Hoặc sử dụng trong các ứng dụng nhỏ, để có thể điều khiển trực tiếp thiết bị trong mạng local bằng Web Browser.

Nếu bạn thấy bài viết này có ích hãy bình luận và đừng quên ra nhập **Hội Anh Em Nghiện Lập trình** nhé.

5/5 - (1 bình chọn)

## Related Posts:

1. [Bài 2: Lập trình ESP32 Webserver chế độ Access Point \(WIFI AP Mode\)](#)
2. [Bài 6: Lập trình ESP32 Timer Millis và ngắt Timer](#)
3. [Bài 5: Lập trình ESP32 ngắt ngoài EXTI](#)
4. [Bài 2: Lập trình ESP32 Analog Input đọc tín hiệu tương tự \(ADC\)](#)
5. [Tổng quan về sơ đồ chân ESP32 và ngoại vi](#)
6. [Lập trình ESP32 từ A tới Z](#)



**KHUÊ NGUYỄN**

Chỉ là người đam mê điện tử và lập trình. Làm được gì thì viết cho anh em xem thôi. :D

# 7 THOUGHTS ON “BÀI 1: LẬP TRÌNH ESP32 WEB SERVER CHẾ ĐỘ WIFI STATION BẬT TẮT LED”

*Tuan Anh* says:

a làm thêm về con esp32 đi a <3

10/07/2021 AT 3:47 CHIỀU

TRẢ LỜI

*Khuê Nguyễn* says:

Anh đang làm thêm nhé

10/07/2021 AT 11:23 CHIỀU

TRẢ LỜI

*Tiến đẹp trai* says:

khởi tạo WiFiSever khác gì cách e hay làm là WebSever

12/07/2021 AT 5:06 CHIỀU

TRẢ LỜI

*Khải* says:

Anh có thể làm 1 post về esp32-s2 được không anh!

13/07/2021 AT 4:26 CHIỀU

TRẢ LỜI

*Khuê Nguyễn* says:

A chưa dùng con S2 này bao giờ, nó khác gì vậy?

13/07/2021 AT 10:10 CHIỀU

TRẢ LỜI

*xuân* says:

Anh cho em hỏi mình có thể kết nối lora và wifi cùng 1 lúc được không ạ

12/08/2021 AT 9:44 SÁNG

TRẢ LỜI

*Hải* says:

webSever.begin(); hàm này em không gọi ra được ạ. Em đã add thư viện Wifi.h và WebSever.h rồi ạ

## Trả lời

Email của bạn sẽ không được hiển thị công khai. Các trường bắt buộc được đánh dấu \*

**Bình luận \***

**Tên \***

**Email \***

**Trang web**

**PHẢN HỒI**

**Fanpage**

 Khuê Nguyễn Creator - Họ...  
2.754 lượt thích

**Đã thích** **Chia sẻ**

**Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển**  
khoảng một tháng trước

Lý do thời gian gần đây mình không viết bài và làm thêm gì cả là đây 😊) Chính thức ra mắt sản phẩm định vị thông minh vTag.

Đây là một sản phẩm định vị đa năng với 3 công nghệ định vị WIFI, GPS, LBS kết hợp với sóng NB-IOT dành riêng cho các sản phẩm IOT.

Chỉ với 990.000đ chúng ta đã có thể có sản phẩm để:

- Định vị trẻ em, con cái... [Xem thêm](#)



## Bài viết khác

### Lập trình 8051 - AT89S52



**Bài 1: Tổng quan về 8051 và chip AT89S51 - 52**



**Khuê Nguyễn Creator**



## Tổng quan về 8051

8051 là một dòng chip nhập môn cho lập trình viên nhúng, chúng được sử...

[ĐỌC THÊM](#)

## Lập trình STM32 và CubeMX



Khuê Nguyễn Creator

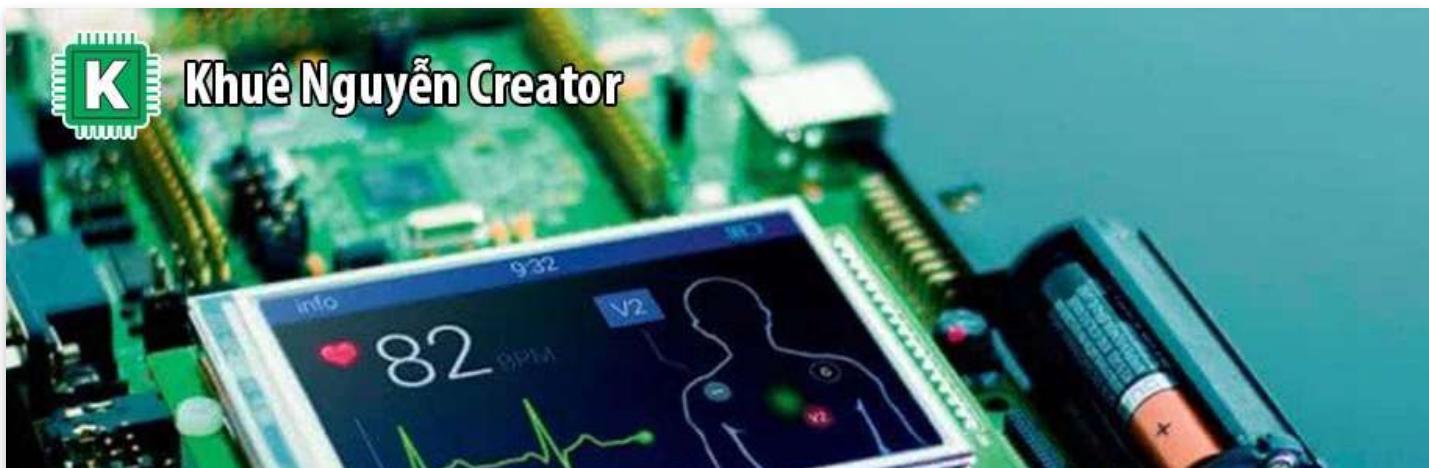


## Lập trình STM32 HID Host giao tiếp với chuột và bàn phím

Lập trình STM32 USB HID Host giao tiếp với chuột và bàn phím máy tính

Trong bài này chúng ta sẽ cùng học STM32 HID Host, biến STM32 giống như...

[ĐỌC THÊM](#)





## Lộ trình học lập trình nhúng từ A tới Z

Lập trình nhúng là một ngành có cơ hội nhưng cũng đòi hỏi nhiều kiến...

3 COMMENTS

[ĐỌC THÊM](#)

### Lập trình STM32 và CubeMX





### Khuê Nguyễn Creator




### Lập trình STM32F407 SDIO đọc dữ liệu thẻ nhớ

[Lập trình STM32 SDIO đọc ghi dữ liệu vào thẻ nhớ SD card](#)

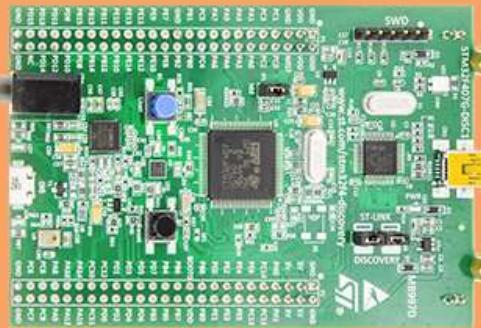
Trong bài này chúng ta cùng học cách lập trình STM32 SDIO, một chuẩn giao...

[ĐỌC THÊM](#)

### Lập trình STM32 và CubeMX



### Khuê Nguyễn Creator



## Lập trình STM32F407 DAC chuyển đổi số sang tương tự

Lập trình STM32 DAC tạo sóng hình Sin trên KIT STM32F407 Discovery

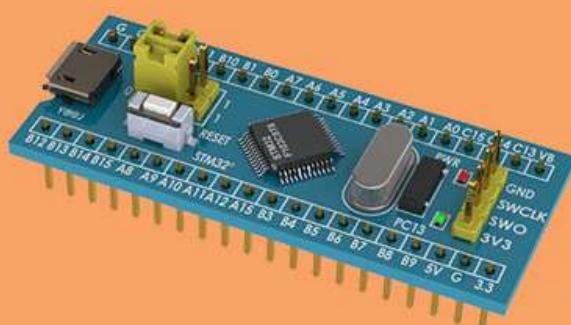
Trong bài này chúng ta sẽ cùng nhau tìm hiểu STM32 DAC với KIT STM32F407VE...

[ĐỌC THÊM](#)

## Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



## Sử dụng hàm printf để in Log khi Debug trên STM32

Hướng dẫn sử dụng printf với STM32 Uart để in Log trên Keil C

Trong bài này chúng ta sẽ học cách retarget hàm printf của thư viện stdio...

3 COMMENTS

[ĐỌC THÊM](#)

## ESP32 và Platform IO



Khuê Nguyễn Creator



## Bài 9 WIFI: Lập trình ESP32 OTA nạp firmware trên Internet

Lập trình ESP32 FOTA nạp firmware qua mạng Internet với OTA Drive

Trong bài này chúng ta sẽ học cách sử dụng ESP32 FOTA (Firmware Over The...

4 COMMENTS

[ĐỌC THÊM](#)

## Lập trình Nuvoton



Khuê Nguyễn Creator



# Cài đặt SDC Complier và Code:Blocks IDE

## Hướng dẫn cài đặt SDCC và Code::Blocks lập trình Nuvoton

Ở bài này chúng ta sẽ cài đặt các công cụ cần thiết cho việc...

**ĐỌC THÊM**



Blog này làm ra để lưu trữ tất cả những kiến thức, những câu chuyện của mình. Đôi khi là những ý tưởng nhất thời, đôi khi là các dự án tự mình làm. Chia sẻ cho người khác cũng là niềm vui của mình, kiến thức mỗi người là khác nhau, không hẳn quá cao siêu nhưng sẽ có lúc hữu dụng.

## Liên Kết

Nhóm: Nghiên Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

## Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn