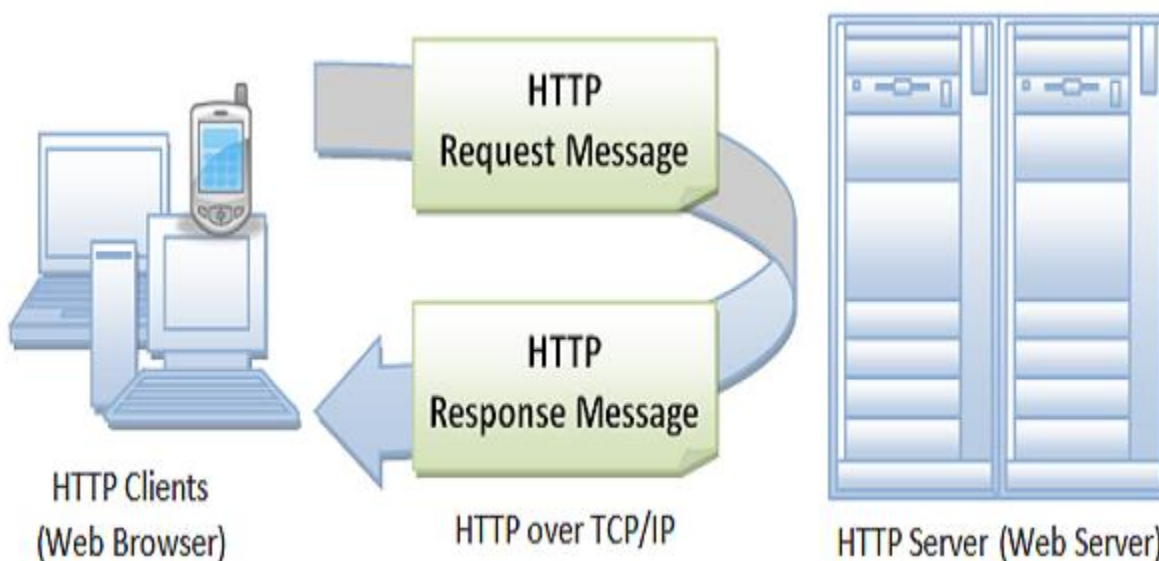


# [IoT] Bài 4: Giới thiệu ngôn ngữ HTML và mô hình http request/response

17 Tháng Ba, 2020 Đào Nguyễn IoT tutorial, WIFI-ESP8266 3



**HTTP** là giao thức truyền tải siêu văn bản được sử dụng trong www dùng để truyền tải dữ liệu giữa Web server đến các trình duyệt Web và ngược lại. Web server ở đây là máy chủ còn trình duyệt web là máy khách.

Có thể hiểu 1 cách đơn giản, khi bạn gõ 1 địa chỉ vào trình duyệt WEB, trình duyệt web sẽ gửi 1 bản tin yêu cầu (request) tới địa chỉ (server) đó qua giao thức http, server sẽ tiếp nhận và trả lời kết quả lại cho trình duyệt WEB

Để liên hệ tới server, chúng ta cần 2 thông tin cơ bản là địa chỉ IP của server và cổng hoạt động.

- Địa chỉ IP là 1 dãy số khó nhớ, ví dụ IP của **blog của mình** là 103.82.32.32 nó là 1 còn số khá khó nhớ nên mình đã mua thêm tên miền **<http://iot47.com/>**, khi các bạn gõ tên website này vào, trình duyệt sẽ truy cập tới **DNS Server** và hỏi xem tên miền này ứng với IP nào, sau khi lấy đc IP trình duyệt sẽ truy cập vào blog của mình bằng IP đó. Các bạn cũng có thể gõ IP trên vào thanh trình duyệt để vào trực tiếp blog của mình luôn. Nhưng mình tắt chế độ này nên các bạn chỉ vô được bằng tên miền thôi
- Cổng (Port), một địa chỉ IP như 1 ngôi nhà lớn vậy, muốn vào nhà phải đi qua cổng, ghi truy cập vào server cũng phải báo cần vào cổng nào (ví dụ **[iot47.com:80](http://iot47.com:80)**). Khi bạn gõ địa chỉ vào trình duyệt web mà không gõ kèm cổng, trình duyệt sẽ mặc định ngầm hiểu đó là cổng 80

## Bản tin truy vấn HTTP

Chúng ta sẽ tìm hiểu cụ thể nội dung của truy vấn http

1 request gồm các dòng, mỗi dòng có 2 thành phần là header và value. Đây là ví dụ request của 1 trình duyệt web

Request Header	Value
(Request-Line)	GET / HTTP/1.1
Host	iot47.com
User-Agent	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding	gzip, deflate
Cookie	_ga=GA1.2.529740874.1426574792; PHPSESSID=0e61i2eb0mff46hejl7r0kfms2; _gat=1; ci_session=a%3...
Connection	keep-alive

Dòng 1 thông tin kiểu truyền dữ liệu (GET/POST), tiếp đến là phiên bản HTTP

Dòng 2 là host cần truy vấn tới

Các dòng dưới là 1 số thông tin của máy khách gửi kèm để server biết rõ hơn về khách của mình (ví thời gian truy cập, máy tính truy cập hãng gì, hệ điều hành gì, xài ngôn ngữ gì, có hỗ trợ cookie không ....v....v...)

Mình sẽ chỉ giới thiệu qua vậy thôi nhé, các bạn muốn tìm hiểu kĩ hơn thì tham khảo thêm tài liệu trên google

## Bản tin trả lời

Sau khi nhận được truy vấn, server sẽ phản hồi lại nội dung phù hợp với truy vấn đó cho khách.

Response Header	Value
(Status-Line)	HTTP/1.1 304 Not Modified
Date	Tue, 17 Mar 2015 14:15:52 GMT
Server	Apache/2
Connection	Keep-Alive
Keep-Alive	timeout=1, max=100
Etag	"b8c1a-2275-5115f173d1e40"
Vary	Accept-Encoding,User-Agent

Dòng 1 trả về trạng thái kèm 1 mã lỗi, được phân loại như sau:

- 200 OK: request thành công.
- 202 Accepted: request đã được nhận, nhưng không có kết quả nào trả về, thông báo cho client tiếp tục chờ đợi.
- 204 No Content: request đã được xử lý nhưng không có thành phần nào được trả về.
- 205 Reset: giống như 204 nhưng mã này còn yêu cầu client reset lại document view.
- 206 Partial Content: server chỉ gửi về một phần dữ liệu, phụ thuộc vào giá trị range header của client đã gửi.

- 301 Moved Permanently: tài nguyên đã được chuyển hoàn toàn tới địa chỉ Location trong HTTP response.
- 303 See other: tài nguyên đã được chuyển tạm thời tới địa chỉ Location trong HTTP response.
- 304 Not Modified: tài nguyên không thay đổi từ lần cuối client request, nên client có thể sử dụng đã lưu trong cache.
- 400 Bad Request: request không đúng dạng, cú pháp.
- 401 Unauthorized: client chưa xác thực.
- 403 Forbidden: client không có quyền truy cập.
- 404 Not Found: không tìm thấy tài nguyên.
- 405 Method Not Allowed: phương thức không được server hỗ trợ.
- 500 Internal Server Error: có lỗi trong quá trình xử lý của server.
- 501 Not Implemented: server không hỗ trợ chức năng client yêu cầu.
- 503: Service Unavailable: Server bị quá tải, hoặc bị lỗi xử lý.

## Phương thức GET và POST

Khi truy vấn lên server, máy khách sẽ phải gửi các thông tin cần thiết, 2 cách để gửi cơ bản gồm có GET và POST.

- GET request có thể được cached, bookmark và lưu trong lịch sử của trình duyệt.
- GET request bị giới hạn về chiều dài, do chiều dài của URL là có hạn.
- GET request không nên dùng với dữ liệu quan trọng, chỉ dùng để nhận dữ liệu.

Với GET, câu truy vấn sẽ được đính kèm vào đường dẫn của HTTP request. Ví dụ: `/?username="abc"&password="def"`

Ngược lại, với POST thì câu truy vấn sẽ được gửi trong phần message body của HTTP request

- POST không thể, cached, bookmark hay lưu trong lịch sử trình duyệt.
- POST không bị giới hạn về độ dài.

Nói chung, với các dữ liệu quan trọng cần bảo mật thì ta xài POST, còn không quan trọng thì xài GET

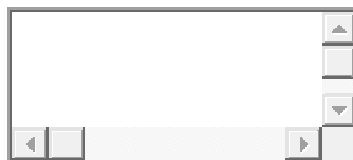
## Ngôn ngữ HTML

Sau khi server phản hồi các header, nó sẽ trả về nội dung html của trang web đó, trình duyệt sẽ phân tích mã html rồi render thành màn hình đồ họa cho người dùng

Để xem mã html của 1 trang web, các bạn ấn Ctrl+U



## 1 chương trình html cơ bản có dạng như sau



```
1<!DOCTYPE html>
```

```
2<html>
```

```
3 <head>
```

```
4 <title>This is a title</title>
```

```
5 </head>
```

```
6 <body>
```

```
7 <p>Hello world!</p>
```

```
8 </body>
```

```
9</html>
```

Trên google có rất nhiều blog và tài liệu rồi, các bạn tự tìm hiểu thêm nhé !

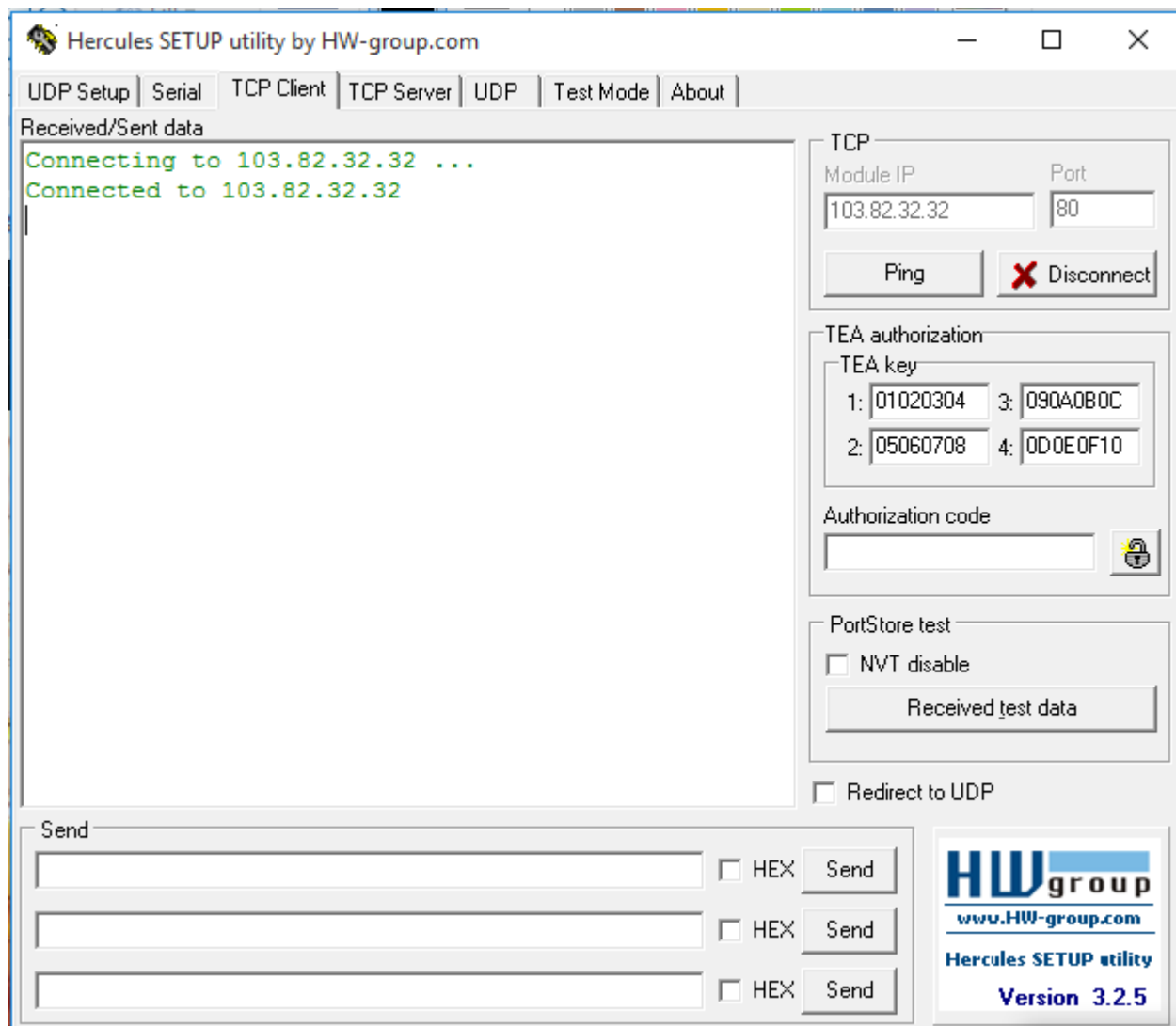
## DEMO gửi thử truy vấn HTTP bằng tay

Mình đã tạo 1 website giám sát hành trình của xe máy trên trang <http://giamsathanhtrinh.iot47.com/>

Bây giờ chúng ta thử gửi tín hiệu tọa độ của xem máy lên trang này nhé

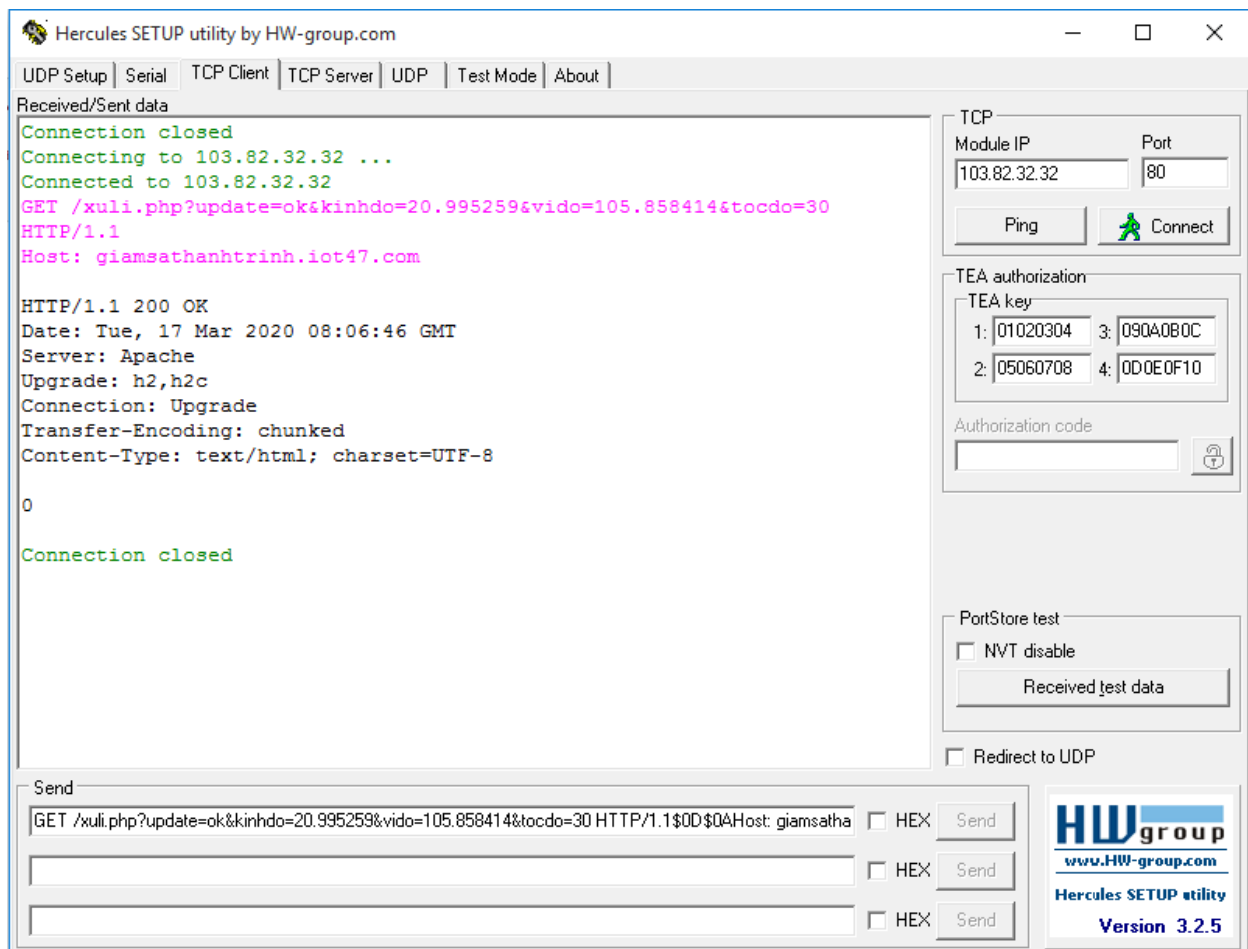
Các bạn tải phần mềm **Hercules**

Vào tab TCP Client, mục Module IP gõ IP của blog mình là 103.82.32.32, cổng 80 và ấn connect



Kết nối đã thành công, giờ hãy gõ request vào, nó có dạng như này:

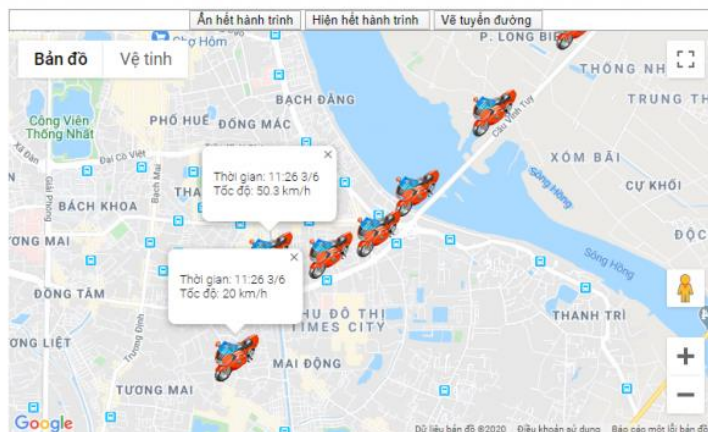
GET /xuli.php?update=ok&kinhdo=20.995259&vido=105.858414&tocdo=30 HTTP/1.1\$0D\$0AHost: giamsathanhtrinh.iot47.com\$0D\$0A\$0D\$0A



Sau khi ấn Send thì server đã trả về mã 200 tức request thành công và chủ động ngắt kết nối do mình không yêu cầu nó giữ kết nối lại. Bây giờ truy cập vào <http://giamathsanhtrinh.iot47.com/> để xem tọa độ được cập nhật lên nhé !

## Đồ án giám sát hành trình và tốc độ của xe máy qua GPS

Time	Tọa độ	Tốc độ		
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:27 3/6	20.989318,105.858427	0.27	Xem	ADD
11:26 3/6	20.989318,105.858427	20	Xem	ADD
11:26 3/6	20.995034,105.857151	40	Xem	ADD
11:26 3/6	20.995189,105.858107	51	Xem	ADD
11:26 3/6	20.998053,105.861487	50.3	Xem	ADD
11:25 3/6	20.998053,105.866997	45.12	Xem	ADD
11:24 3/6	20.999919,105.871390	40.17	Xem	ADD
11:24 3/6	21.003343,105.874876	31.2	Xem	ADD
11:24 3/6	21.010147,105.881949	30.2	Xem	ADD

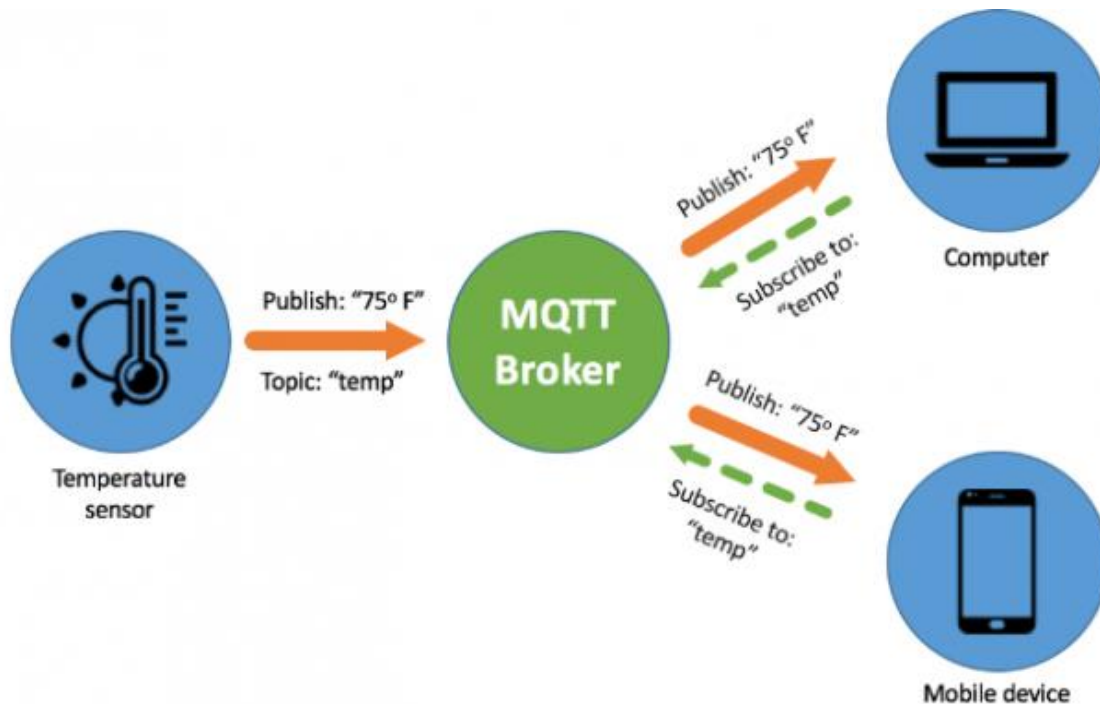


Trong bài sau mình sẽ giới thiệu về ngôn ngữ lập trình server PHP và hệ quản trị cơ sở dữ liệu MySQL



# [IoT] Bài 7: ESP8266 – arduino ide và giao thức MQTT

24 Tháng Ba, 2020 Đào Nguyên IoT tutorial, WIFI-ESP8266 1



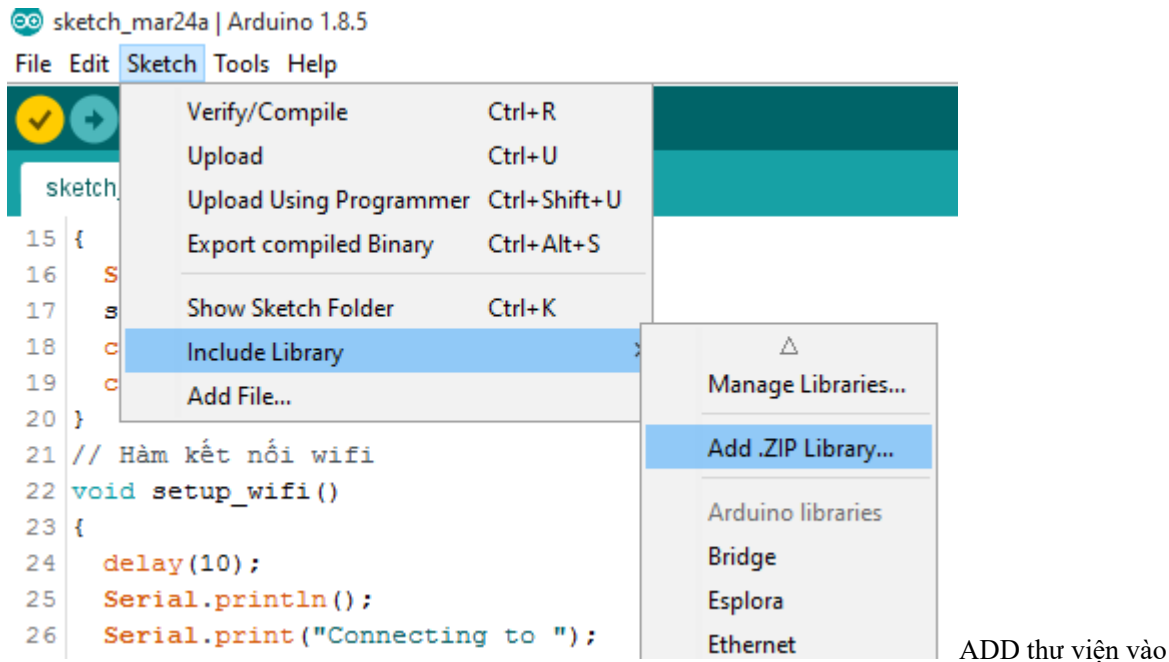
Giao thức MQTT phù hợp nhất cho các dự án IoT thương mại, nó đáp ứng tốc độ tốt, băng thông ít, độ tin cậy cao. Tài liệu về giao thức MQTT thì các bạn tham khảo ở các trên mạng hoặc 1 số bài sau :

- <https://smartfactoryvn.com/technology/internet-of-things/giao-thuc-mqtt-la-gi-nhung-ung-dung-cua-mqtt-nhu-the-nao/>
- <https://esp8266.vn/nonos-sdk/mqtt/what-is-mqtt/>

Mình sẽ không nhắc lại phần lý thuyết nữa vì trên mạng có rất nhiều rồi. Chúng ta sẽ đi vào thực hành làm thử 1 project với giao thức MQTT luôn

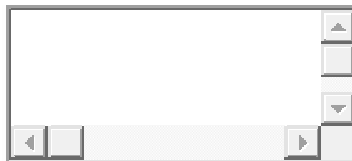
Đầu tiên, phía esp8266 các bạn tải thư viện [Pubsubclient](#)





Giao thức MQTT cần có 1 server ( gọi là broker) để làm trung tâm của mọi luồng dữ liệu, trong các bài viết sau mình sẽ hướng dẫn các bạn tự build server, còn trong bài này mình sẽ sử dụng server miễn phí không bảo mật là [broker.hivemq.com](https://broker.hivemq.com) để demo

Các bạn copy chương trình cho esp8266



```

1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3
4
5 // Thông tin về wifi
6 #define ssid "dieukhien"
7 #define password "12345678"
8 #define mqtt_server "broker.hivemq.com"
9 const uint16_t mqtt_port = 1883; //Port của CloudMQTT TCP
10
11 WiFiClient espClient;
12 PubSubClient client(espClient);
13
14 void setup()
15 {
16   Serial.begin(115200);
17   setup_wifi();
18   client.setServer(mqtt_server, mqtt_port);
19   client.setCallback(callback);
20 }
21 // Hàm kết nối wifi

```

```

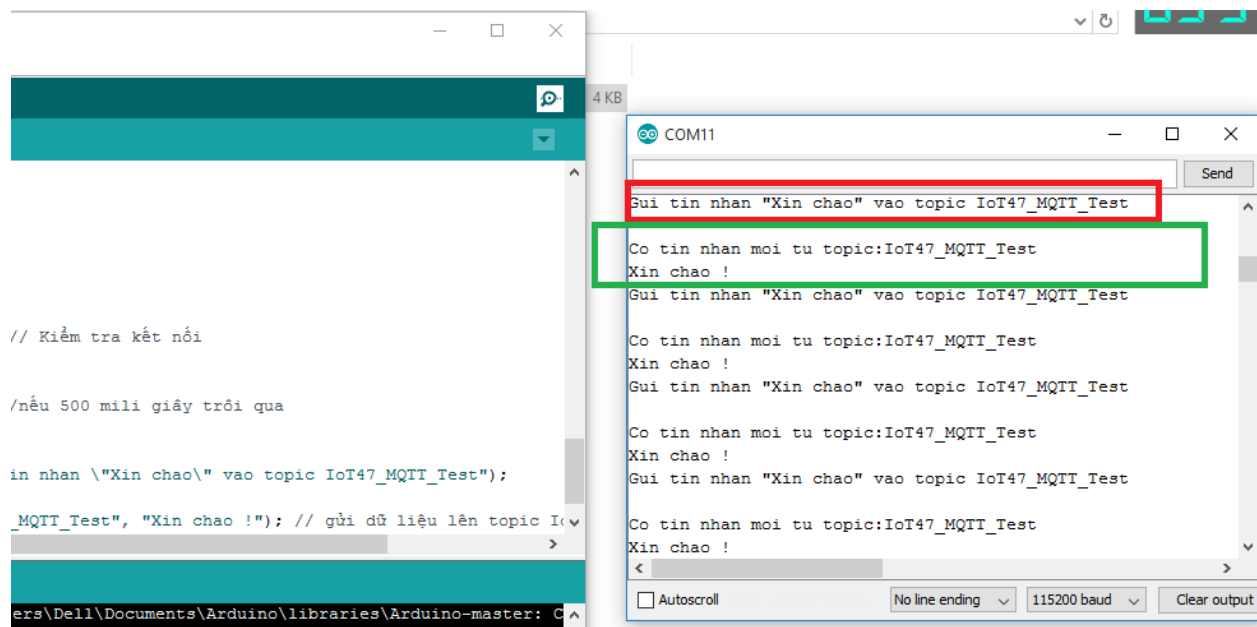
22 void setup_wifi()
23 {
24   delay(10);
25   Serial.println();
26   Serial.print("Connecting to ");
27   Serial.println(ssid);
28   WiFi.begin(ssid, password);
29   while (WiFi.status() != WL_CONNECTED) {
30     delay(500);
31     Serial.print(".");
32   }
33   Serial.println("");
34   Serial.println("WiFi connected");
35   Serial.println("IP address: ");
36   Serial.println(WiFi.localIP());
37 }
38 // Hàm call back để nhận dữ liệu
39 void callback(char* topic, byte* payload, unsigned int length)
40 {
41   Serial.print("Co tin nhan moi tu topic:");
42   Serial.println(topic);
43   for (int i = 0; i < length; i++)
44     Serial.print((char)payload[i]);
45   Serial.println();
46 }
47 // Hàm reconnect thực hiện kết nối lại khi mất kết nối với MQTT Broker
48 void reconnect()
49 {
50   while (!client.connected()) // Chờ tới khi kết nối
51   {
52     // Thực hiện kết nối với mqtt user và pass
53     if (client.connect("ESP8266_id1", "ESP_offline", 0, 0, "ESP8266_id1_offline")) //kết nối vào broker
54     {
55       Serial.println("Đã kết nối:");
56       client.subscribe("IoT47_MQTT_Test"); //đăng kí nhận dữ liệu từ topic IoT47_MQTT_Test
57     }
58     else
59     {
60       Serial.print("Lỗi:, rc=");
61       Serial.print(client.state());
62       Serial.println(" try again in 5 seconds");
63       // Đợi 5s
64       delay(5000);
65     }
66   }
67 }
68 unsigned long t;
69 void loop()
70 {
71   if (!client.connected()) // Kiểm tra kết nối
72     reconnect();
73   client.loop();
74   if (millis() - t > 500) //nếu 500 mili giây trôi qua
75   {
76     t = millis();
77     Serial.print("Gui tin nhan \"Xin chao\" vao topic IoT47_MQTT_Test");
78     client.publish("IoT47_MQTT_Test", "Xin chao !"); // gửi dữ liệu lên topic IoT47_MQTT_Test

```

```
79 }  
80 }
```

- Ở dòng 5 và 6 các bạn đổi thành wifi của mình.
- Hàm callback là hàm gọi lại khi có dữ liệu gửi đến topic mà chúng ta đăng kí
- Hàm client.subscribe dùng để đăng kí 1 topic
- Hàm client.publish dùng để gửi dữ liệu lên 1 topic
- Hàm client.connect để kết nối vào MQTT Broker, với các tham số  
ESP8266\_id1 : Id của thiết bị đăng kí vào (có thể chỉnh sửa bất thành bất kì)  
ESP\_offline : khi thiết bị (esp8266) mất mạng (offline) thì broker sẽ xuất bản 1 tin nhắn vào topic này  
ESP8266\_offline : Nội dung của tin nhắn offline

Sau khi kết nối thành công ở dòng 53 mình đăng kí topic **IoT47\_MQTT\_Test** và trong hàm loop xuất bản tin nhắn "Xin chào" vào chính topic **IoT47\_MQTT\_Test**. Như vậy chúng ta sẽ nhận lại được chính tin nhắn mà ta đã xuất bản !

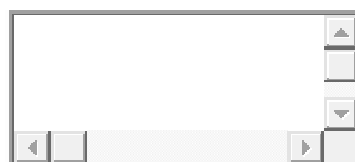


Kết quả: Mình đã nhận được chính tin nhắn mà mình xuất bản lên

## Thiết kế giao diện web gửi tin nhắn cho ESP8266

Để có thể kết nối tới MQTT broker, mình sẽ sử dụng ngôn ngữ JavaScript và thư viện PahoMQTT

Các bạn tạo 1 file tên là index.html và thêm mã code web



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Demo MQTT</title>
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <meta charset="utf-8">
7     <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
8     <script type="text/javascript" language="javascript">
9         var max_at_OK;
10        function makeid()
11        {
12            var text = "";
13            var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
14
15            for (var i = 0; i < 5; i++)
16                text += possible.charAt(Math.floor(Math.random() * possible.length));
17
18            return text;
19        }
20        // Create a client instance
21        var client = new Paho.MQTT.Client("broker.hivemq.com", 8000, makeid());
22
23        // set callback handlers
24        client.onConnectionLost = onConnectionLost;
25        client.onMessageArrived = onMessageArrived;
26
27        var options = {
28            useSSL: false,
29            userName: "",
30            password: "",
31            onSuccess: onConnect,
32            onFailure: doFail
33        }
34
35
36        console.log("Connect to broker.hivemq.com:8000");
37        // connect the client
38        client.connect(options);
39
40        function doFail(e){
41            console.log(e);
42        }
43
44        function onConnect() //sự kiện kết nối thành công
45        {
46            console.log("Connect OK");
47            client.subscribe("IoT47_MQTT_Test"); //đăng kí kênh
48        }
49
50        // called when the client loses its connection
51        function onConnectionLost(responseObject)
52        {
53            if (responseObject.errorCode !== 0)
54            {
55                console.log(responseObject.errorMessage);
56            }
57        }
```

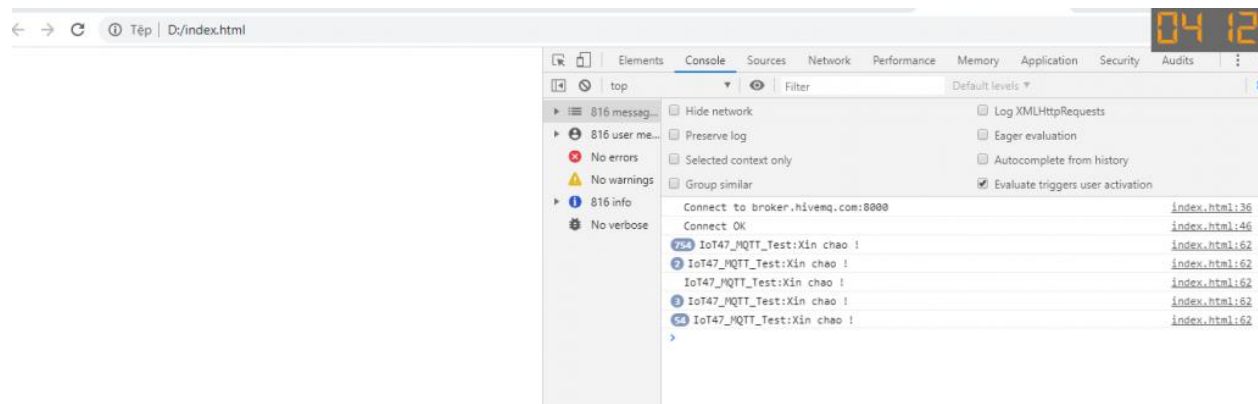
```

58
59         // called when a message arrives
60         function onMessageArrived(message)
61         {
62             console.log(message.destinationName + ":" + message.payloadString);
63         }
64         function public (topic,data)
65         {
66             message = new Paho.MQTT.Message(data);
67             message.destinationName = topic;
68             client.send(message);
69         }
70     </script>
71 </head>
72 <body>
73
74 </body>
75 </html>

```

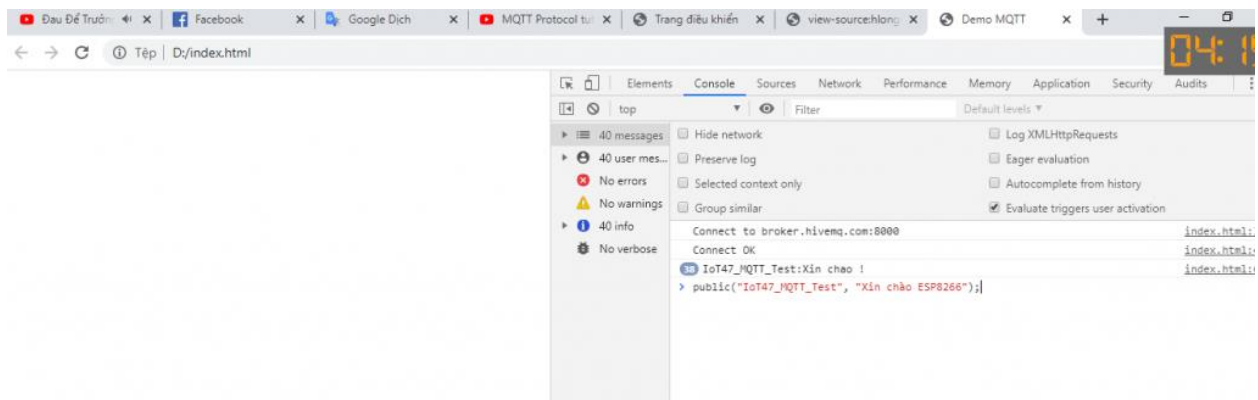
- Hàm makeID có nhiệm vụ tạo ra 1 id ngẫu nhiên để web kết nối tới MQTT broker tránh bị trùng id
- Mình sẽ kết nối vào **broker.hivemq.com** qua cổng 8000 vì cổng 8000 là cổng dành cho các kết nối thông qua Socket, trong khi cổng 1883 dành cho các kết nối qua TCP
- Hàm **public** dùng để xuất bản 1 tin nhắn tới 1 topic nào đó
- Hàm **client.subscribe** dùng để đăng kí nhận tin nhắn từ 1 topic nào đó
- Hàm **onMessageArrived** là hàm callback khi có 1 tin nhắn từ 1 topic đã đăng kí gửi tới

Mình sẽ lưu lại và mở file này bằng trình duyệt, sau đó ấn F12 để xem dữ liệu debug in ra màn hình console

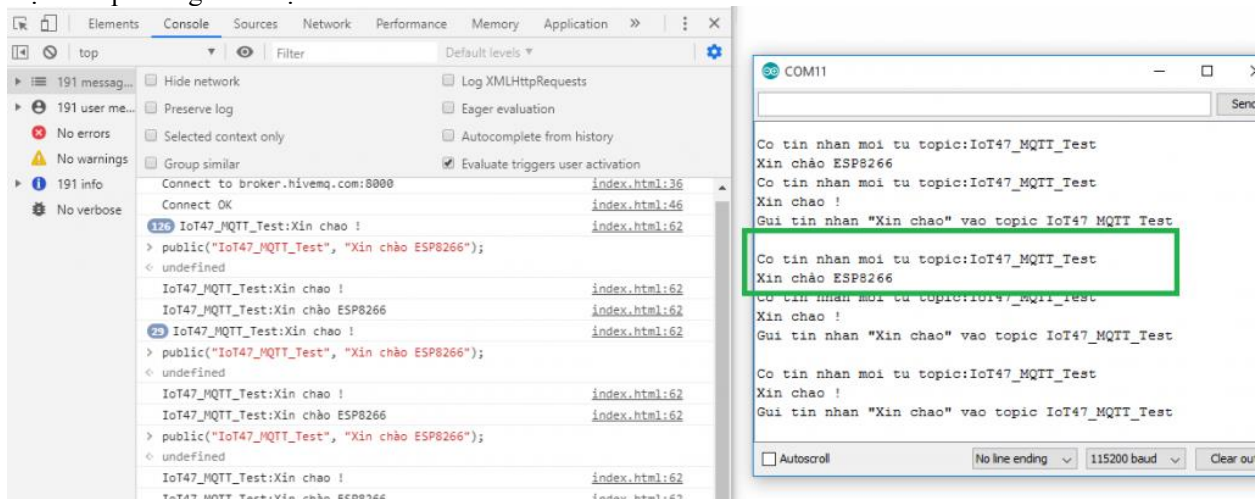


Các bạn có thể thấy sau khi thông báo Connect OK thì chúng ta đã nhận được tin nhắn “Xin chao !” mà esp8266 liên tục gửi lên mỗi 500 mili giây

Cũng trong màn hình debug, mình gõ hàm public để gửi thử dữ liệu đến esp8266 nhé !



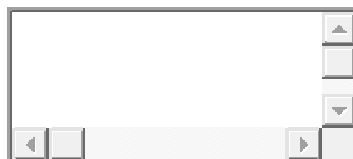
Gọi hàm public gửi dữ liệu



ESP8266 đã nhận được tin nhắn xin chào

## Thiết kế giao diện nút nhấn điều khiển thiết bị cho giao thức mqtt

Chúng ta sẽ tận dụng lại giao diện đã xài ở [bài 5](#) nhé



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Demo MQTT</title>
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <meta charset="utf-8">
7     <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
8     <script type="text/javascript" language="javascript">
9         var max,at_OK;
10        function makeid()
11        {

```

```

12         var text = "";
13         var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
14
15         for (var i = 0; i < 5; i++)
16             text += possible.charAt(Math.floor(Math.random() * possible.length));
17
18         return text;
19     }
20     // Create a client instance
21     var client = new Paho.MQTT.Client("broker.hivemq.com", 8000, makeid());
22
23     // set callback handlers
24     client.onConnectionLost = onConnectionLost;
25     client.onMessageArrived = onMessageArrived;
26
27     var options = {
28         useSSL: false,
29         userName: "",
30         password: "",
31         onSuccess:onConnect,
32         onFailure:doFail
33     }
34
35
36     console.log("Connect to broker.hivemq.com:8000");
37     // connect the client
38     client.connect(options);
39
40     function doFail(e){
41         console.log(e);
42     }
43
44     function onConnect() //sự kiện kết nối thành công
45     {
46         console.log("Connect OK");
47         client.subscribe("ESP8266_sent_data"); //đăng kí kênh
48     }
49
50     // called when the client loses its connection
51     function onConnectionLost(responseObject)
52     {
53         if (responseObject.errorCode !== 0)
54         {
55             console.log(responseObject.errorMessage);
56         }
57     }
58
59     // called when a message arrives
60     function onMessageArrived(message)
61     {
62         console.log(message.destinationName + ":" + message.payloadString);
63         document.getElementById("tinhanh").innerHTML = "Tin nhắn từ esp8266: " + message.payloadString;
64     }
65
66     function public (topic,data)
67     {
68         message = new Paho.MQTT.Message(data);

```



```

69         message.destinationName = topic;
70         client.send(message);
71     }
72 </script>
73 <style>
74         .b{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius:
75 10px;}
76         .t{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius:
77 10px;}
78     </style>
79 </head>
80 <body>
81 <div style="width: 330px;height: auto;margin: 0 auto;margin-top: 70px">
82 <h1 align="center">Điều khiển thiết bị qua WIFI - MQTT</h1>
83 <p align="center" id="tinnhan">Tin nhắn từ esp8266: ... </p>
84     <table align="center">
85         <tr>
86             <td><button class='b' onclick="public('ESP8266_read_data','Bật 1')">Bật 1</button><td>
87             <td><button class='t' onclick="public('ESP8266_read_data','Tắt 1')">Tắt 1</button><td>
88         <tr>
89         <tr>
90             <td><button class='b' onclick="public('ESP8266_read_data','Bật 2')">Bật 2</button><td>
91             <td><button class='t' onclick="public('ESP8266_read_data','Tắt 2')">Tắt 2</button><td>
92         <tr>
93         <tr>
94             <td><button class='b' onclick="public('ESP8266_read_data','Bật 3')">Bật 3</button><td>
95             <td><button class='t' onclick="public('ESP8266_read_data','Tắt 3')">Tắt 3</button><td>
96         <tr>
97         <tr>
98             <td><button class='b' onclick="public('ESP8266_read_data','Bật 4')">Bật 4</button><td>
99             <td><button class='t' onclick="public('ESP8266_read_data','Tắt 4')">Tắt 4</button><td>
100        <tr>
101        </table>
102</div>
</body>
</html>

```

## Điều khiển thiết bị qua WIFI - MQTT

Tin nhắn từ esp8266: ...

Bật 1		Tắt 1	
Bật 2		Tắt 2	
Bật 3		Tắt 3	
Bật 4		Tắt 4	

Khi ấn nút ( sự kiện OnClick) mình sẽ cho xuất bản tin nhắn tương ứng tới ESP8266 qua topic **ESP8266\_read\_data**. Do vậy ở code esp8266 mình sẽ sửa lại code cho nó đăng kí vào topic **ESP8266\_read\_data** và cho web đăng kí vào kênh **ESP8266\_sent\_data** để nhận tin nhắn esp8266 gửi lên

```

while (!client.connected()) // Chờ tới khi kết nối
{
    // Thực hiện kết nối với mqtt user và pass
    if (client.connect("ESP8266_id1","ESP_offline",0,0,"ESP8266_id1_offline")) //kết nối vào broker
    {
        Serial.println("Đã kết nối:");
        client.subscribe("ESP8266_read_data"); //đăng kí nhận dữ liệu từ topic ESP8266_read_data
    }
    else
    {
        Serial.print("Lỗi:, rc=");
        Serial.print(client.connect());
    }
}

```

Sửa lại topic đăng kí của esp8266 thành ESP8266\_read\_data

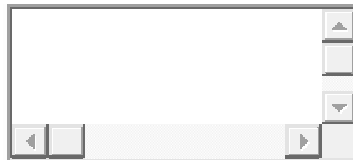
Mình cũng thêm đoạn mã đọc data từ cổng Serial để gửi lên cho web

```

if(Serial.available() > 0)
{
    delay(30);
    char inputString[30]="";
    int i=0;
    while(Serial.available() > 0)
    {
        char inChar = (char)Serial.read();
        inputString[i++] = inChar;
    }
    client.publish("ESP8266_sent_data", inputString); // gửi dữ liệu lên topic ESP8266_sent_data
}

```

Full code cho esp8266



```

1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3
4
5 // Thông tin về wifi
6 #define ssid "dieukhien"
7 #define password "12345678"
8 #define mqtt_server "broker.hivemq.com"
9 const uint16_t mqtt_port = 1883; //Port của CloudMQTT TCP
10
11 WiFiClient espClient;
12 PubSubClient client(espClient);
13
14 void setup()
15 {
16   Serial.begin(115200);
17   setup_wifi();
18   client.setServer(mqtt_server, mqtt_port);
19   client.setCallback(callback);
20 }
21 // Hàm kết nối wifi

```

```

22 void setup_wifi()
23 {
24     delay(10);
25     Serial.println();
26     Serial.print("Connecting to ");
27     Serial.println(ssid);
28     WiFi.begin(ssid, password);
29     while (WiFi.status() != WL_CONNECTED) {
30         delay(500);
31         Serial.print(".");
32     }
33     Serial.println("");
34     Serial.println("WiFi connected");
35     Serial.println("IP address: ");
36     Serial.println(WiFi.localIP());
37 }
38 // Hàm call back để nhận dữ liệu
39 void callback(char* topic, byte* payload, unsigned int length)
40 {
41     Serial.print("Co tin nhan moi tu topic:");
42     Serial.println(topic);
43     for (int i = 0; i < length; i++)
44         Serial.print((char)payload[i]);
45     Serial.println();
46 }
47 // Hàm reconnect thực hiện kết nối lại khi mất kết nối với MQTT Broker
48 void reconnect()
49 {
50     while (!client.connected()) // Chờ tới khi kết nối
51     {
52         // Thực hiện kết nối với mqtt user và pass
53         if (client.connect("ESP8266_id1", "ESP_offline", 0, 0, "ESP8266_id1_offline")) //kết nối vào broker
54         {
55             Serial.println("Đã kết nối:");
56             client.subscribe("ESP8266_read_data"); //đăng kí nhận dữ liệu từ topic ESP8266_read_data
57         }
58         else
59         {
60             Serial.print("Lỗi:, rc=");
61             Serial.print(client.state());
62             Serial.println(" try again in 5 seconds");
63             // Đợi 5s
64             delay(5000);
65         }
66     }
67 }
68 void loop()
69 {
70     if (!client.connected()) // Kiểm tra kết nối
71         reconnect();
72     client.loop();
73     if (Serial.available() > 0)
74     {
75         delay(30);
76         char inputString[30] = "";
77         int i = 0;
78         while (Serial.available() > 0)

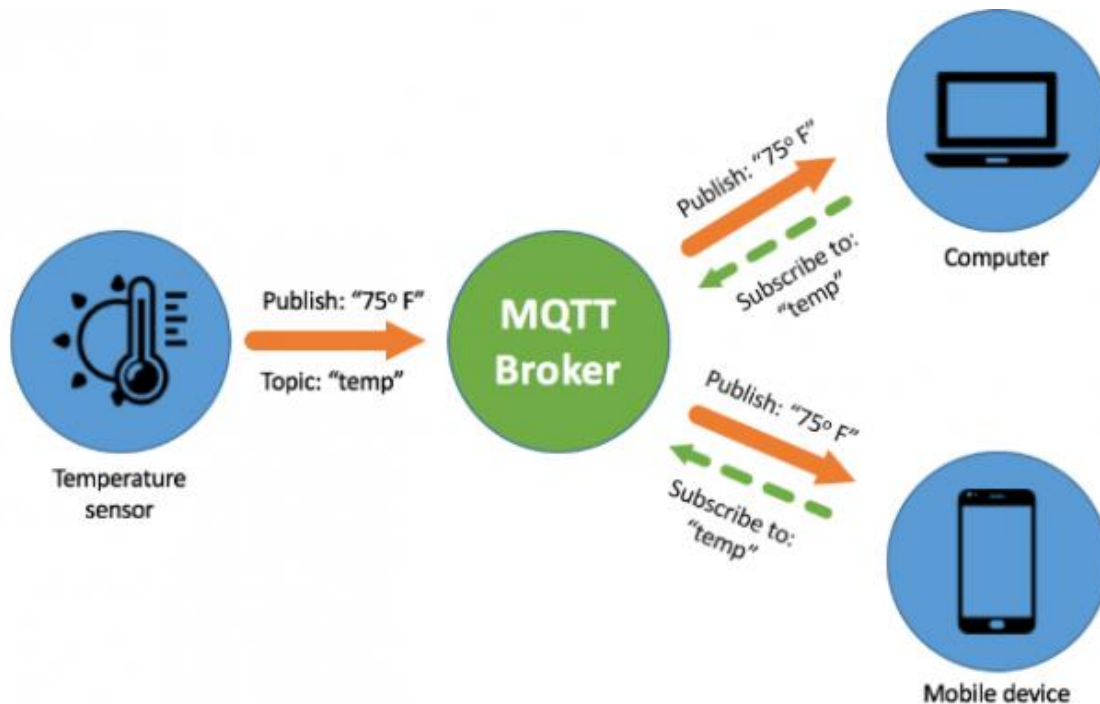
```

```
79 {  
80   char inChar = (char)Serial.read();  
81   inputString[i++] = inChar;  
82 }  
83 client.publish("ESP8266_sent_data", inputString); // gửi dữ liệu lên topic ESP8266_sent_data  
84 }  
85 }
```

Demo kết quả

# [IoT] Bài 8: Giao thức MQTT với tập lệnh ATcommand – esp8266

28 Tháng Ba, 2020 Đào Nguyễn IoT tutorial, WIFI-ESP8266 1



Trong bài trước, mình đã giới thiệu về giao thức MQTT và chúng ta đã nhanh chóng test qua về MQTT trên esp8266 với arduino. Tiếp tục phần này, mình sẽ giới thiệu về giao tiếp MQTT qua tập lệnh AT

Các bạn cần đọc **bài 3** và **bài 5** trước thì mới hiểu bài này nhé !

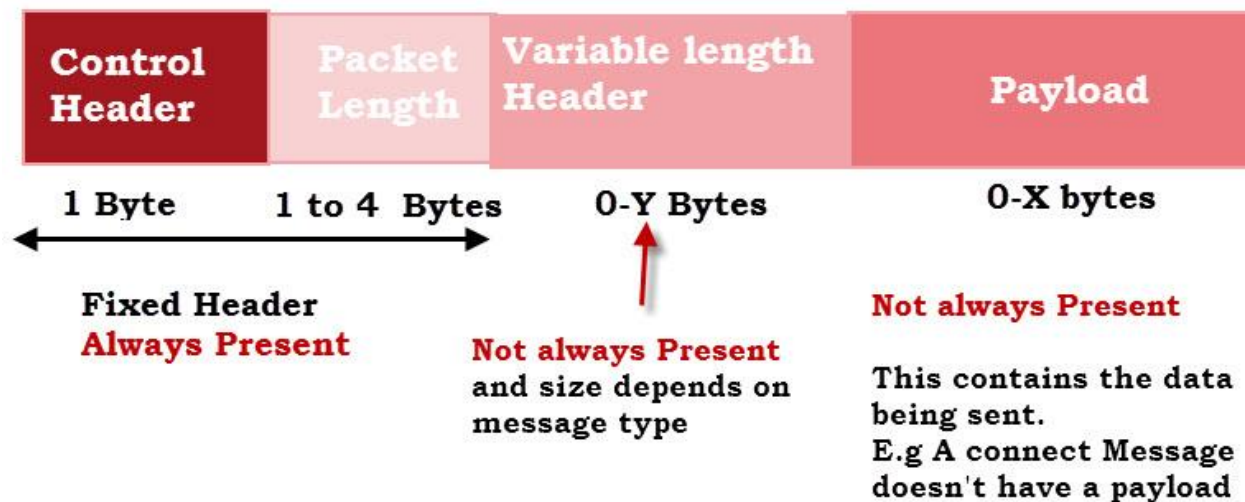
MQTT hoạt động trên nền TCP hoặc Socket, trong bài này mình sẽ chỉ demo qua trên nền tảng TCP, còn socket chúng ta sẽ tìm hiểu ở các bài sau nhé

## Cấu trúc của các gói tin MQTT

Các gói tin liên quan đến MQTT thì nhiều lắm, mình chỉ giới thiệu qua 3 gói tin cơ bản

- Gói tin connect
- Gói tin publish
- Gói tin subscriber

1, Connect – Client gửi 1 thông điệp kết nối đến broker



## MQTT Standard Packet Structure

**Control Header** : 1 byte

4 bit cao nhất mô tả MQTT **Control Packet type** và 4 bit thấp chứa **Control flags**.

Dưới đây là bảng mô tả của **Control Header**

CONNECT	0x10
CONNACK	0x20
PUBLISH	0x30
PUBACK	0x40
PUBREC	0x50
PUBREL	0x60
PUBCOMP	0x70
SUBSCRIBE	0x80
SUBACK	0x90

UNSUBSCRIBE	0xA0
UNSUBACK	0xB0
PINGREQ	0xC0
PINGRESP	0xD0
DISCONNECT	0xE0

**Packet Length** (1 byte đến 4 byte)

**Packet Length** mô tả phía sau nó còn bao nhiêu byte nữa (không tính bản thân nó)

Trường này không cố định số lượng byte, tối đa 4 và tối thiểu 1. Bit cao nhất của byte sẽ xác định xem byte phía sau nó có thuộc **Packet Length** không ! Do đó chỉ có 7bit dùng để mã hóa dữ liệu

Ví dụ 1: Gói tin phía sau còn 31 byte thì ta chỉ cần 1 byte cho **Packet Length**  
=> Packet Length = 0x1F

Ví dụ 2: Gói tin phía sau còn 321 byte thì ta sẽ cần 2 byte cho **Packet Length** với byte1 là byte thấp và byte2 là byte cao. Cứ mỗi giá trị của byte cao sẽ = 128 lần byte thấp. Ta tách  $321 = 65 + 2 \times 128$

=> Packet Length = 0x41 0x02

Bit cao nhất của byte1 phải được set lên 1 để báo vẫn còn 1 byte nữa cho **Packet Length** nên phải sửa thành  
Packet Length = 0xC1 0x02

Với 4 byte **Packet Length**, ta sẽ mã hóa được tối đa 268,435,455 byte dữ liệu

From	To
0 (0x00)	127 (0x7F)
28 (0x80, 0x01)	16 383 (0xFF, 0x7F)
16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

Tiếp tục tới trường thứ 3 là **Variable length Header**, nó gồm 4 trường nhỏ sau: Protocol Name, Protocol Level, Connect Flags, and Keep Alive

- Protocol Name: (6byte) 0x00 0x04 0x4D 0x51 0x54 0x54
- Protocol Level: (1byte) 0x04



- Connect Flags: (1 byte) 0x??
- Keep Alive: (2 byte) 0x?? 0x??

Các bạn chỉ cần quan tâm tới Connect Flags  
Cấu trúc của nó như sau:

Bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
byte 8	X	X	X	X	X	X	X	0

Bit 1 chúng ta sẽ mặc định để bằng 1

Các bit khác các bạn xài cái nào thì set nó lên. Ví dụ mình không cần user, password thì chỉ cần 0x02 là ok, nếu set cái nào lên thì phải mô tả nó trong **payload** nhé

**Payload** là trường sẽ chứa các thông tin để phục vụ việc connect như ID\_Client, User, Password. Thông tin trong trường này phải tuân thủ thứ tự sau:

Client ID -> Will Topic -> Will Message -> User Name -> Password

Chú ý: Ở trên byte Connect Flags cái nào không set thì bỏ qua nó trong **Payload** nhé !

Còn Keep Alive là 2 byte chứa thời gian được tính bằng giây. Đó là khoảng thời gian tối đa được phép trôi qua giữa điểm mà Client hoàn thành việc truyền một Control Packet và điểm mà nó bắt đầu gửi tiếp theo. Nói chung cứ để từ 10 đến 60s tùy các bạn

Dưới đây là 1 ví dụ về 1 gói tin connect vào broker với Client id là **IOT47** và không sử dụng tên user lẫn password

0x10 0x11 0x00 0x04 0x4D 0x51 0x54 0x54 0x04 0x02 0x00 0x3C 0x00 0x05 0x49 0x4F 0x54 0x34 0x37

Trong đó:

0x10 là Control Header

0x11 = 17 là trường **Packet Length** báo phía sau nó có 17 byte

0x00 0x04 0x4D 0x51 0x54 0x54 là Protocol Name thuộc trường **Variable length Header**

0x04 là Protocol Level

0x02 là Connect Flag

0x00 0x3C là Keep Alive mình để 60 giây

0x00 0x05 0x49 0x4F 0x54 0x34 0x37 là gói Packet, với 0x05 ám chỉ phía sau nó có 5byte data. Do Connect Flag mình để 0x02 ( tức không xài user name, password hay gì hết) nên thông tin ở packet chỉ cần id client là được (với id client là IOT47 = 0x49 0x4F 0x54 0x34 0x37 ) ( id client là bắt buộc phải có nha, nó là gì cũng được tùy các bạn)

Publish – gửi 1 tin nhắn đến 1 topic

Cấu trúc như sau

1byte **Control Header** = 0x30  
1 đến 4 byte **Packet Length**  
1 byte 0x00  
1byte chứa độ dài của topic  
còn lại là nội dung của tin nhắn

Ví dụ: mình sẽ publish tin nhắn tới topic ESP8266\_sent\_data và nội dung là xinchao

0x30 0x1A 0x00 0x11 0x45 0x53 0x50 0x38 0x32 0x36 0x36 0x5F 0x73 0x65 0x6E 0x74 0x5F 0x64 0x61  
0x74 0x61 0x78 0x69 0x6E 0x63 0x68 0x61 0x6F

Trong đó:

0x30 là Control Header  
0x1A = 26 là số lượng byte phía sau  
0x11 = 17 là độ dài của topic  
0x45 0x53 0x50 0x38 0x32 0x36 0x36 0x5F 0x73 0x65 0x6E 0x74 0x5F 0x64 0x61 0x74 0x61 =  
ESP8266\_sent\_data là tên của topic  
0x78 0x69 0x6E 0x63 0x68 0x61 0x6F = xinchao là nội dung tin nhắn

Subscriber – đăng kí nhận tin nhắn từ 1 topic

Cấu trúc như sau

1byte **Control Header** = 0x82  
1 đến 4 byte **Packet Length**  
1 byte 0x00 và 1 byte 0x01 (**Variable header**)  
1 byte 0x00  
1byte chứa độ dài của topic  
1 byte 0x00

Ví dụ, mình sẽ đăng kí topic ESP8266\_read\_data

0x82 0x16 0x00 0x01 0x00 0x11 0x45 0x53 0x50 0x38 0x32 0x36 0x36 0x5F 0x72 0x65 0x61 0x64 0x5F  
0x64 0x61 0x74 0x61 0x00

Trong đó:

1byte **Control Header** = 0x82  
0x16 = 22 là số lượng byte phía sau  
0x11 = 17 là độ dài của topic  
0x45 0x53 0x50 0x38 0x32 0x36 0x36 0x5F 0x72 0x65 0x61 0x64 0x5F 0x64 0x61 0x74 0x61 =  
ESP8266\_read\_data là tên của topic

Mình đã giới thiệu qua 1 vài cấu trúc cơ bản để làm việc với MQTT, các bạn tự tìm hiểu thêm ở đây nhé !  
<https://docs.solace.com/MQTT-311-Prtl-Conformance-Spec/MQTT%20Control%20Packet%20format.htm>

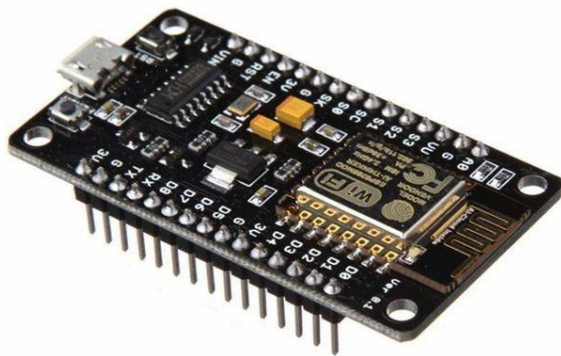
## DEMO giao tiếp MQTT với tập lệnh AT Command

Trước tiên chúng ta sẽ test bằng cách gửi thủ công bằng tay qua phần mềm Hercules luôn nhé

Các bạn chuẩn bị

- Mạch chuyển đổi **USB-UART PL2303**
- Laptop/PC đã cài **Driver cho PL2303**
- **Phần mềm nạp firmware cho ESP8266**

Hoặc sử dụng module esp8266 node-MCU cho nó tiện, chỉ việc cắm dây usb vào là xong, bao nhanh bao phê



Nhắc lại 1 vài lệnh cơ bản ở bài 2

```
AT+CWJAP="IOT47","12345678"<CR><LF> //kết nối vào wifi nhà bạn
AT+CWMODE=1<CR><LF> // yêu cầu module hoạt động ở chế độ Station/Client
AT+CIPMUX=0<CR><LF>
ATE0<CR><LF> //tắt chế độ phản hồi ngửa mắt
```

Các bạn gọi các lệnh cơ bản phía trên để khởi tạo module trước nhé ! <CR><LF> là 2 byte 0x0D 0x0A hay \r\n đấy nhé ! Trên phần mềm **Hercules** thì là \$0D\$0A

Còn đây là lệnh cho giao thức TCP mà chúng ta sẽ dùng để phục cho các kết nối MQTT  
`AT+CIPSTART="TCP","yourserver.com",80<CR><LF>` // khởi động 1 kết nối TCP đến server nào đó ở 1 port nào đó (ví dụ ở đây là cổng 80)  
`AT+CIPSEND=X<CR><LF>` // bắt đầu gửi 1 gói tin TCP với độ dài X

Ở demo này, mình kết nối tới broker MQTT là broker.hivemq.com và cổng sử dụng là 1883 – đây là cổng chuyên dụng cho các kết nối TCP. Do broker.hivemq.com là broker free không bảo mật nên chúng ta không cần user và password gì hết 😊

Các bạn có thể thử với mã html mình đã viết và hướng dẫn ở **bài 7** hoặc mình sẽ show giao diện web đó trực tiếp ở đây cho các bạn test luôn

## Điều khiển thiết bị qua WIFI - MQTT

Tin nhắn từ esp8266: ...

Bat 1		Tat 1	
Bat 2		Tat 2	
Bat 3		Tat 3	
Bat 4		Tat 4	

Note pad của mình đây nhé:

```
//kết nối tới broker mqtt
```

```
AT+CIPSTART="TCP","broker.hivemq.com",1883$0D$0A
```

```
// kết nối tới broker
```

```
AT+CIPSEND=19$0D$0A
```

```
$10$11$00$04$4D$51$54$54$04$02$00$3C$00$05$49$4F$54$34$37
```

```
//gửi tin nhắn xin chào tới topic ESP8266_sent_data
```

```
AT+CIPSEND=28$0D$0A
```

```
$30$1A$00$11$45$53$50$38$32$36$36$5F$73$65$6E$74$5F$64$61$74$61$78$69$6E$63$68$61$6F
```

```
//đăng kí nhận tin nhắn từ topic ESP8266_read_data
```

```
AT+CIPSEND=24$0D$0A
```

```
$82$16$00$01$00$11$45$53$50$38$32$36$36$5F$72$65$61$64$5F$64$61$74$61$00
```

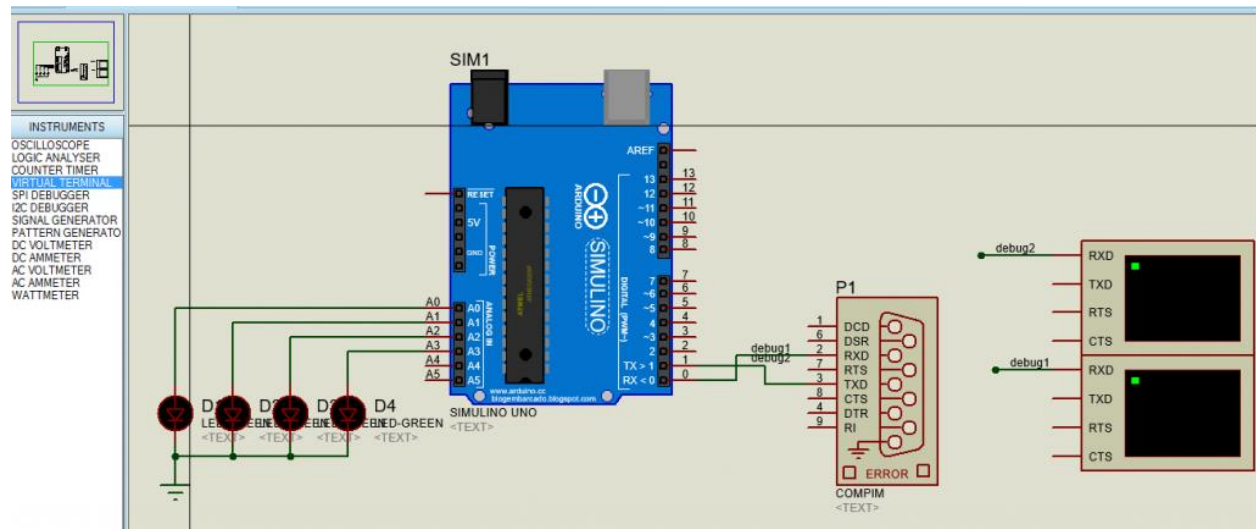
## Lập trình giao tiếp Arduino với esp8266 qua tập lệnh AT - giao thức MQTT điều khiển 4 thiết bị

Kết nối

Arduino	ESP8266
3.3V	3.3V
GND	GND

RX	TX
TX	RX

Giống như **bài 3**, mình sẽ sử dụng phần mềm mô phỏng proteus để giao tiếp arduino ảo với esp8266 thật thông qua cổng COMPI (nghèo quá không có tiền mua arduino 😊)

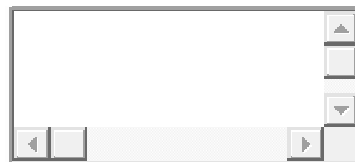


Sơ đồ kết nối

Mình thêm 2 cái **virtul terminal** để debug dữ liệu vào và ra trên cổng UART

Lập trình

Hàm **ESP8266\_SendCommand** có nhiệm vụ gửi 1 AT command tới cho esp8266 và chờ cho tới khi trả về value, mình cũng cho thêm 1 cái timeout để thoát ra khi hết thời gian chờ



```

1 //hàm này có nhiệm vụ gửi 1 lệnh AT đến cho esp8266 và chờ xem nó có phản hồi đúng data về không
2 bool ESP8266_SendCommand(String cmd,String value,int timeout)
3 {
4   String rx_data="";
5   while(Serial.available())Serial.read(); //xóa sạch bộ đệm RX
6
7   Serial.print(cmd);    // gửi lệnh AT đi

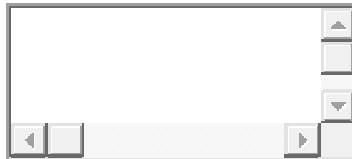
```

```

8 unsigned long t=millis();
9 while(1)
10 {
11     if((millis() - t) > timeout)return false; //nếu tới time out thì thoát
12     if (Serial.available() > 0)
13     {
14         char inChar = (char)Serial.read();
15         rx_data+=inChar;
16         if(rx_data.indexOf(value) != -1)return true;
17     }
18 }
19 }

```

Hàm **ESP8266\_init** sẽ khởi tạo module esp8266 và trả về err\_code nếu gặp phải lỗi



```

1 int ESP8266_init()
2 {
3     int i=0;
4
5     while(1)//GỬI LỆNH AT
6     {
7         if(ESP8266_SendCommand("AT\r\n","OK",500) == true)break; // liên tục gửi lệnh AT xem esp8266 có hoạt động không
8         i++; if(i==10)return 1; //trả về false báo init thất bại
9     }
10
11     //1 số setup cơ bản
12     delay(100);Serial.print("ATE0\r\n"); //tắt phản hồi
13     delay(100);Serial.print("AT+CWMODE=1\r\n"); //chế độ station
14     delay(100);Serial.print("AT+CIPMUX=0\r\n"); //chế độ đơn kênh
15
16     delay(100);

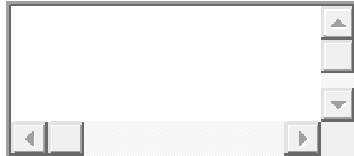
```

```

17 String data_connect = "AT+CWJAP=\"" + ssid + "\",\"\" + pass + "\"\r\n";
18 if(ESP8266_SendCommand(data_connect,"OK",10000) == false)return 2; //kết nối vào wifi, timeout 10 giây
19 return 0;
20}

```

Tiếp tục là hàm khởi tạo giao thức MQTT



```

1 void MQTT_init()
2 {
3   Connect_packet[13] = ID_Client.length(); //byte 14 ghi độ dài của id
4   Connect_packet[1] = Connect_packet[13] + 12; //byte 1 ghi Packet Length
5   Connect_Packet_Length = Connect_packet[1] + 2;
6 }
7 int MQTT_broker_connect()
8 {
9   MQTT_init();
10  int i=10;
11  while(i--)
12  {
13    if(ESP8266_SendCommand("AT+CIPSTART=\"TCP\", \"broker.hivemq.com\", 1883\r\n", "CONNECT", 3000) == true) //kết
14    nối tcp tới broker
15    {
16      String CIPSEN = "AT+CIPSEND=" + String(Connect_Packet_Length) + "\r\n";
17      Serial.print(CIPSEN);
18      delay(50);
19      //gửi gói tin connect đi
20      for(int i=0;i<14;i++)Serial.write(Connect_packet[i]);
21      Serial.print(ID_Client);
22      ESP8266_SendCommand("", "SEND OK", 3000); //chờ phản hồi SEND OK
23      return 0;
24    }

```

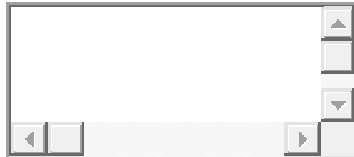


```

25 }
26 return 1; //không thể kết nối tới server
    }

```

FULL code arduino, các bạn tự ngâm cứu nhé



```

1 String ssid ="Tang5";
2 String pass ="99999999";
3 String ID_Client="IOT47";
4
5 unsigned char Connect_packet[14]={0x10, 0xFF, 0x00, 0x04, 0x4D, 0x51, 0x54, 0x54, 0x04, 0x02, 0x00, 0x3C, 0x00,
6 0xFF};
7 int Connect_Packet_Length;
8
9 //hàm này có nhiệm vụ gửi 1 lệnh AT đến cho esp8266 và chờ xem nó có phản hồi đúng data về không
10 bool ESP8266_SendCommand(String cmd,String value,int timeout)
11 {
12     String rx_data="";
13     while(Serial.available())Serial.read(); //xóa sạch bộ đệm RX
14
15     Serial.print(cmd);    // gửi lệnh AT đi
16     unsigned long t=millis();
17     while(1)
18     {
19         if((millis() - t) > timeout)return false; //nếu tới time out thì thoát
20         if (Serial.available() > 0)
21         {
22             char inChar = (char)Serial.read();
23             rx_data+=inChar;
24             if(rx_data.indexOf(value) != -1)return true;
25         }
26     }
27 }

```

```

26 }
27 }
28 int ESP8266_init()
29 {
30     int i=0;
31
32     while(1)//GỬI LỆNH AT
33     {
34         if(ESP8266_SendCommand("AT\r\n","OK",500) == true)break; // liên tục gửi lệnh AT xem esp8266 có hoạt động không
35         i++; if(i==10)return 1; //trả về false báo init thất bại
36     }
37
38     //1 số setup cơ bản
39     delay(100);Serial.print("ATE0\r\n"); //tắt phản hồi
40     delay(100);Serial.print("AT+CWMODE=1\r\n"); //chế độ station
41     delay(100);Serial.print("AT+CIPMUX=0\r\n"); //chế độ đơn kênh
42
43     delay(100);
44     String data_connect = "AT+CWLAP=\"" + ssid + "\",\"\" + pass + "\"\r\n";
45     if(ESP8266_SendCommand(data_connect,"OK",10000) == false)return 2; //kết nối vào wifi, timeout 10 giây
46     return 0;
47 }
48 void ESP8266_pub(String topic,String mess) //max topic length + mess =127
49 {
50
51 }
52 void ESP8266_sub(String topic) //max topic length =255
53 {
54     //size      size
55     unsigned char Sub_packet[6]={0x82, 0xFF, 0x00, 0x01, 0x00, 0xFF};
56     //tính độ dài gói packet
57     Sub_packet[5] = topic.length();
58     Sub_packet[1] = 4 + Sub_packet[5] + 1;
59

```

```

59 int Sub_packet_length = Sub_packet[1] + 2;
60 String CIPSEN = "AT+CIPSEND=" + String(Sub_packet_length) + "\r\n";
61 Serial.print(CIPSEN);
62 delay(50);
63
64 //gửi gói tin sub đi
65 for(int i=0;i<6;i++)Serial.write(Sub_packet[i]);
66 Serial.print(topic);
67 Serial.write(0x00);
68 ESP8266_SendCommand("", "SEND OK",3000); //chờ phản hồi SEND OK
69 }
70 void MQTT_init()
71 {
72   Connect_packet[13] = ID_Client.length(); //byte 14 ghi độ dài của id
73   Connect_packet[1] = Connect_packet[13] + 12; //byte 1 ghi Packet Length
74   Connect_Packet_Length = Connect_packet[1] + 2;
75 }
76 int MQTT_broker_connect()
77 {
78   MQTT_init();
79   int i=10;
80   while(i--)
81   {
82     if(ESP8266_SendCommand("AT+CIPSTART=\\"TCP\\",\\"broker.hivemq.com\\",1883\r\n","CONNECT",3000) == true)
83       //kết nối tcp tới broker
84       {
85         String CIPSEN = "AT+CIPSEND=" + String(Connect_Packet_Length) + "\r\n";
86         Serial.print(CIPSEN);
87         delay(50);
88         //gửi gói tin connect đi
89         for(int i=0;i<14;i++)Serial.write(Connect_packet[i]);
90         Serial.print(ID_Client);
91         ESP8266_SendCommand("", "SEND OK",3000); //chờ phản hồi SEND OK
92         return 0;

```

```
92 }
93 }
94 return 1; //không thể kết nối tới server
95 }
96 void MQTT_reconnect()
97 {
98
99 }
100 void setup()
101 {
102 pinMode(A0,OUTPUT);digitalWrite(A0,LOW);
103 pinMode(A1,OUTPUT);digitalWrite(A1,LOW);
104 pinMode(A2,OUTPUT);digitalWrite(A2,LOW);
105 pinMode(A3,OUTPUT);digitalWrite(A3,LOW);
106 delay(500);
107 Serial.begin(9600);
108
109 int err_code = 0;
110 err_code=ESP8266_init();
111 if(err_code != 0)
112 {
113 /*
114  * err_code=1 lỗi kết nối
115  * err_code=2 không thể truy cập vào wifi
116  */
117 Serial.print("Error code=");Serial.println(err_code);
118 while(1);
119 }
120 int err_code_connect=MQTT_broker_connect(); // khởi tạo giao thức MQTT
121 if(err_code != 0)
122 {
123 /*
124  * err_code_connect=1 //không thể kết nối tới server
```

```

125  *
126  */
127  Serial.print("err_code_connect=");Serial.println(err_code_connect);
128  while(1);
129  }
130  ESP8266_sub("ESP8266_read_data"); //đăng kí nhận mess từ topic
131}
132
133String inputString="";
134void loop()
135{
136  if (Serial.available() > 0) //nhận tin nhắn từ topic
137  {
138    delay(100);
139    while(Serial.available() > 0) //data bắt đầu từ byte 0x00 , byte tiếp theo chưa độ dài của topic
140    {
141      char inChar = (char)Serial.read();
142      if(inChar == 0) break;
143    }
144    inputString="";
145    while(Serial.available() > 0)
146    {
147      char inChar = (char)Serial.read();
148      inputString+=inChar;
149    }
150    int topic_length = inputString[0];
151    String topic = inputString.substring(1,1+topic_length); //lấy được tên topic
152    String mess = inputString.substring(1+topic_length); //lấy được tên mess
153
154    if(mess == "Bat 1")digitalWrite(A0,HIGH);
155    if(mess == "Tat 1")digitalWrite(A0,LOW);
156    if(mess == "Bat 2")digitalWrite(A1,HIGH);
157    if(mess == "Tat 2")digitalWrite(A1,LOW);

```

```
158  if(mess == "Bat 3")digitalWrite(A2,HIGH);
159  if(mess == "Tat 3")digitalWrite(A2,LOW);
160  if(mess == "Bat 4")digitalWrite(A3,HIGH);
161  if(mess == "Tat 4")digitalWrite(A3,LOW);
162  }
}
```

Demo mô phỏng trên proteus

*Chú ý: Các bạn tự viết thêm hàm publish tương tự nhé, và phải thường xuyên gửi gói tin keep alive để giữ kết nối với broker*

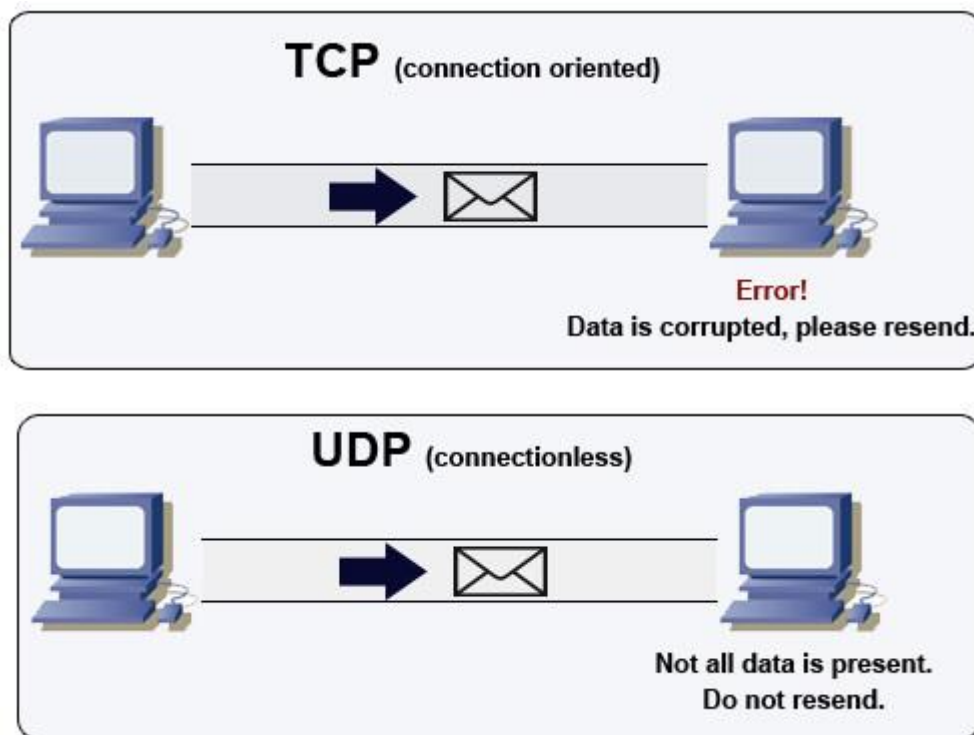
## Download

Các bạn có thể tải về project **tại đây**

**<https://drive.google.com/open?id=1Pi-MqD1HbaRWmgQhB6Kbmwe4dpg6SDjf>**

# [IoT] Bài 9: Tìm hiểu giao thức TCP và UDP

30 Tháng Ba, 2020 Đào Nguyễn IoT tutorial, WIFI-ESP8266 1



Từ bài 1 đến giờ mình đã hướng dẫn và thực hành việc truyền nhận dữ liệu qua TCP rất nhiều, bây giờ chúng ta sẽ tìm hiểu kĩ hơn về TCP và **UDP** !

## TCP

Tham khảo: <https://vi.wikipedia.org/wiki/TCP>

Giao thức TCP là gì?

Nó là giao thức điều khiển truyền vận (Transmission Control Protocol) thuộc giao thức cốt lõi của bộ giao thức TCP/IP. Thông qua TCP, các ứng dụng trên các máy chủ được nối mạng có thể liên lạc với nhau, qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nhận một cách đáng tin cậy và đúng thứ tự. Hơn nữa, TCP có chức năng phân biệt giữa dữ liệu của nhiều ứng dụng (như dịch vụ Web và dịch vụ Email) đồng thời chạy trên cùng một máy chủ.

- TCP là giao thức hướng kết nối (connection-oriented), có nghĩa là buộc phải thiết lập kết nối trước sau đó mới đến tiến trình truyền dữ liệu.
- Cung cấp cơ chế đánh số thứ tự gói tin (sequencing): sử dụng để ráp các gói tin chính xác ở điểm nhận, loại bỏ gói tin trùng lặp.
- TCP có khả năng truyền và nhận dữ liệu cùng một lúc — song công (full-duplex).



- Cơ chế báo nhận (Acknowledgement): tức là khi A gửi gói tin cho B, nếu B nhận được thì sẽ gửi thông báo cho A, trường hợp A không nhận được thông báo thì sẽ gửi lại gói tin tới khi nào B báo nhận thì thôi.
- Tính năng phục hồi dữ liệu bị mất trên đường truyền.

### Đặc điểm

Một số đặc điểm cơ bản của TCP để phân biệt với UDP:

- Truyền dữ liệu không lỗi (do có cơ chế sửa lỗi/truyền lại)
- Truyền các gói dữ liệu theo đúng thứ tự
- Truyền lại các gói dữ liệu mất trên đường truyền
- Loại bỏ các gói dữ liệu trùng lặp
- Cơ chế hạn chế tắc nghẽn đường truyền

### Cấu trúc gói tin

Một gói tin TCP bao gồm 2 phần

- header (có độ dài 20 bytes)
- dữ liệu

Phần header có 11 trường trong đó 10 trường bắt buộc. Trường thứ 11 là tùy chọn (trong bảng minh họa có màu nền đỏ) có tên là: options

+	Bít 0 - 3	4 - 9	10 - 15	16 - 31
0	Source Port			Destination Port
32	Sequence Number			
64	Acknowledgement Number			
96	Data Offset	Reserved	Flags	Window
128	Checksum			Urgent Pointer
160	Options (optional)			
160/192+	Data			

**Source port** : Số hiệu của cổng tại máy tính gửi.

**Destination port** : Số hiệu của cổng tại máy tính nhận

**Sequence number** : Trường này có 2 nhiệm vụ. Nếu cờ SYN bật thì nó là số thứ tự gói ban đầu và byte đầu tiên được gửi có số thứ tự này cộng thêm 1. Nếu không có cờ SYN thì đây là số thứ tự của byte đầu tiên.

**Acknowledgement number** : Nếu cờ ACK bật thì giá trị của trường chính là số thứ tự gói tin tiếp theo mà bên nhận cần

**Data offset** : Trường có độ dài 4 bit quy định độ dài của phần header (tính theo đơn vị từ 32 bit). Phần header có độ dài tối thiểu là 5 từ (160 bit) và tối đa là 15 từ (480 bit) **Reserved** : Dành cho tương lai và có giá trị là 0

**Flags (hay Control bits)** : Bao gồm 6 cờ: URG ACK PSH RST SYN FIN

**Window** : Số byte có thể nhận bắt đầu từ giá trị của trường báo nhận (ACK)

**Checksum** : 16 bit kiểm tra cho cả phần header và dữ liệu. Phương pháp sử dụng được mô tả trong **RFC 793**:

**Urgent pointer** : Nếu cờ URG bật thì giá trị trường này chính là số từ 16 bit mà số thứ tự gói tin (*sequence number*) cần dịch trái.

**Options** : Đây là trường tùy chọn. Nếu có thì độ dài là bội số của 32 bit.

Hết phần **Header** là **Dữ liệu**

*Thực ra nếu các bạn chỉ làm việc với esp8266 thôi thì cũng không cần quan tâm quá sâu vào cấu trúc **Header** của giao thức TCP đâu, vì thư viện wifi hay tập lệnh AT đã lo phần này cho chúng ta rồi. Nhưng do sau này mình sẽ hướng dẫn các bạn làm việc với mạng ethernet nữa ( module ENC28J60) nên mình nói rõ ở bài này luôn. ( khi làm việc với module ethernet ENC28J60 chúng ta sẽ tự thân vận động viết thư viện giao tiếp để hiểu cho rõ )*

## UDP

Tham khảo: <https://vi.wikipedia.org/wiki/UDP>

**UDP (User Datagram Protocol)** là một trong những giao thức cốt lõi của **giao thức TCP/IP**. Dùng UDP, chương trình trên **mạng máy tính** có thể gửi những dữ liệu ngắn được gọi là **datagram** tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà **TCP** làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu

## Cấu trúc gói tin

UDP không đảm bảo cho các tầng phía trên thông điệp đã được gửi đi và người gửi cũng không có trạng thái thông điệp UDP một khi đã được gửi

+	Bits 0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Phần header của UDP chỉ chứa 4 trường dữ liệu, trong đó có 2 trường là tùy chọn (ô nền đỏ trong bảng)

**Source port** : Trường này xác định cổng của người gửi thông tin và có ý nghĩa nếu muốn nhận thông tin phản hồi từ người nhận. Nếu không dùng đến thì đặt nó bằng 0

**Destination port** : Trường xác định cổng nhận thông tin, và trường này là cần thiết.

**Length** : Trường có độ dài 16 bit xác định **chiều dài** của toàn bộ datagram: phần header và dữ liệu. **Chiều dài** tối thiểu là 8 byte khi gói tin không có dữ liệu, chỉ có header.

**Checksum** : Trường **checksum** 16 bit dùng cho việc kiểm tra lỗi của phần header và dữ liệu. Phương pháp tính checksum được định nghĩa trong **RFC 768**.

Do thiếu tính tin cậy, các ứng dụng UDP nói chung phải chấp nhận mất mát, lỗi hoặc trùng dữ liệu.

## Thực hành truyền nhận dữ liệu qua UDP với ESP8266

Do chúng ta đã làm việc với TCP nhiều nên phần này mình chỉ demo với UDP

Các lệnh AT thường dùng cho UDP

Tham khảo : [https://www.espressif.com/sites/default/files/4b-esp8266\\_at\\_command\\_examples\\_en\\_v1.3.pdf](https://www.espressif.com/sites/default/files/4b-esp8266_at_command_examples_en_v1.3.pdf)

AT+CIPSTART="UDP","192.168.1.14",8000,23,0\$0D\$0A

Trong đó:

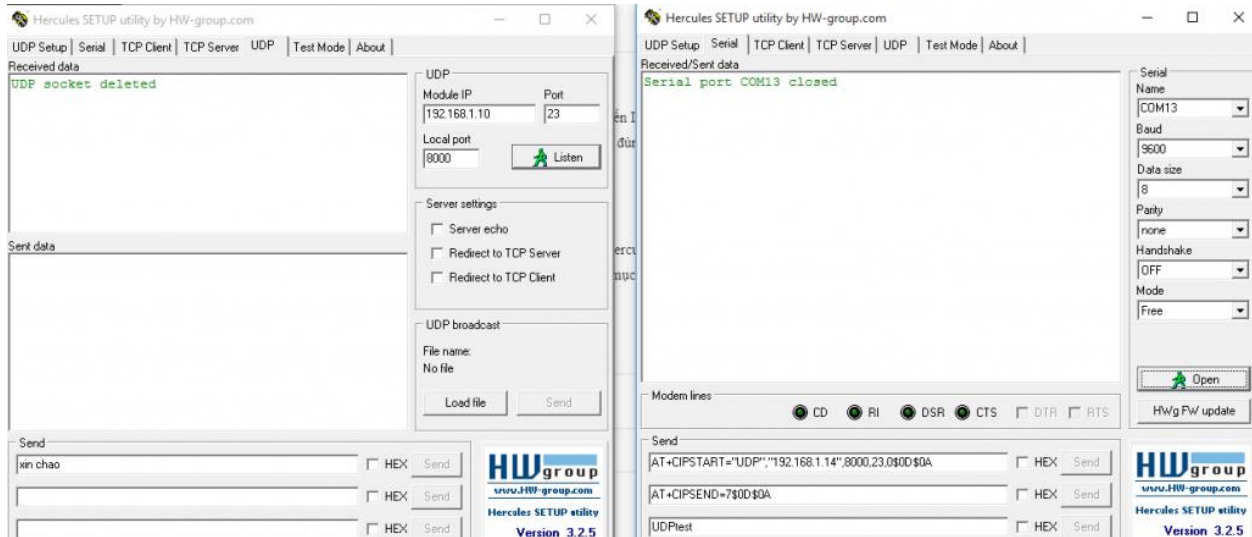
- UDP là tên giao thức ta muốn xài
- 192.168.1.14 là địa chỉ ip của đối phương
- 8000 là cổng dùng để nhận data của đối phương
- 23 là cổng dùng để nhận data của esp8266
- 0 thì các bạn để mặc định ( hoặc tìm hiểu thêm trong file pdf tham khảo phía trên)

AT+CIPSEND=X\$0D\$0A

Lệnh này để gửi 1 lượng dữ liệu có độ dài là X đến IP đã kết nối ở phía trên. Sau lệnh này esp8266 trả về ">" và chúng ta gửi dữ liệu đi với đúng độ dài X là được

Test

Mình sẽ truyền nhận thủ công bằng phần mềm Hercules. Các bạn mở 2 app Hercules lên, 1 app để giao tiếp Serial với esp8266, 1 app mở mục UDP



Ở app mở tab UDP, mình điền:

**Module IP** chính là ip của esp8266 ( dùng lệnh AT+CIFSR để lấy IP của esp8266)

**Port** là cổng nhận dữ liệu của esp8266 (23)

**Listen** là cổng nhận dữ liệu của máy tính (8000)

Sau đó ấn Listen để khởi tạo kết nối UDP cho máy tính

Với ESP8266

Bước 1 : khởi tạo kết nối UDP với máy tính bằng lệnh

AT+CIPSTART="UDP","192.168.1.14",8000,23,0\$0D\$0A

Với 192.168.1.14 là IP của máy tính (nếu không biết thì vào cmd -> gõ ipconfig để lấy ip của máy tính

Bước 2: Đo xem chuỗi data gửi đi dài bao nhiêu, ví dụ mình gửi nội dung là ESP8266\_hello\_PC thì chuỗi này dài 16

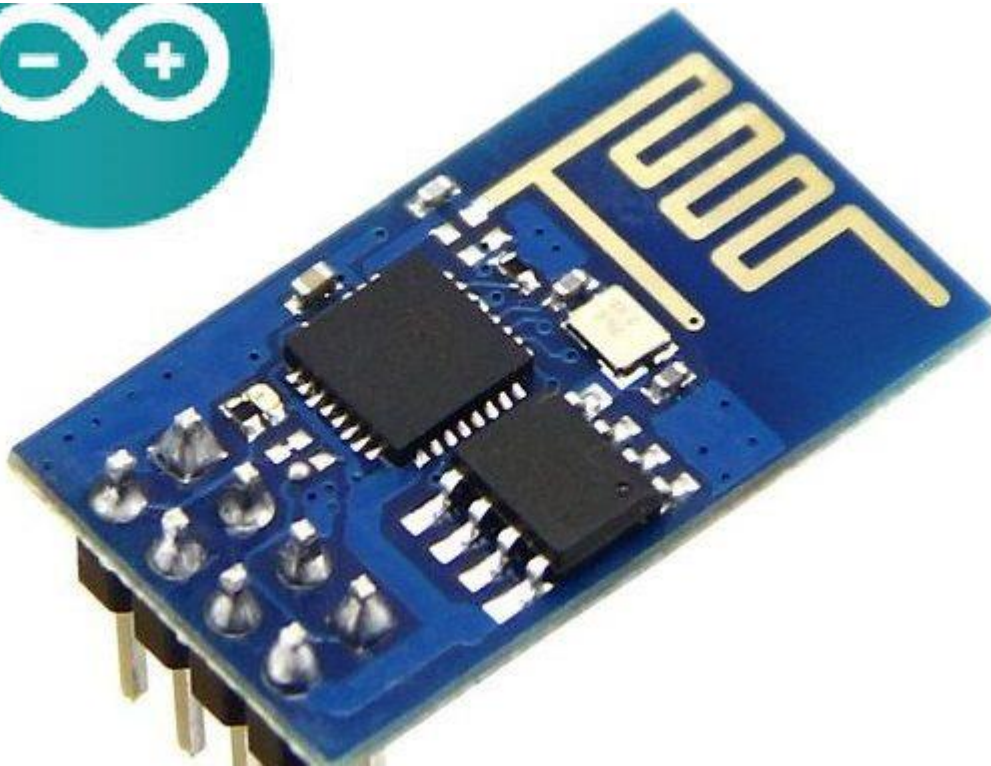
Bước 3: Gửi AT+CIPSEND=16\$0D\$0A

Bước 4: Gửi ESP8266\_hello\_PC

Trăm nghe không bằng 1 thấy, dưới đây là video test

# [IoT] Bài 5: Tạo WebServer với ESP8266 và lập trình cho esp8266 bằng arduino IDE

22 Tháng Ba, 2020 Đào Nguyễn IoT tutorial, WIFI-ESP8266 8



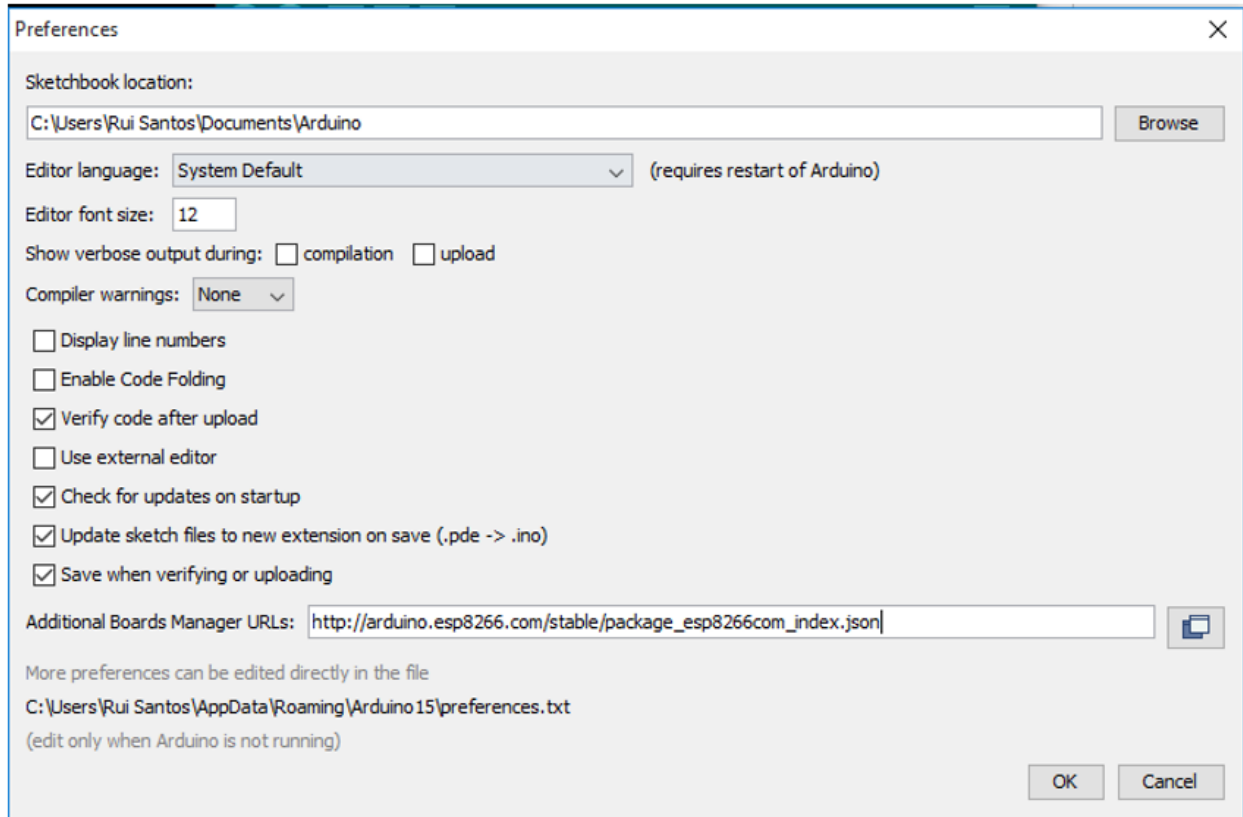
Trong bài này, chúng ta sẽ tạo webserver với esp8266 và biến esp8266 thành một máy chủ web để điện thoại, máy tính truy cập tới ! Chúng ta đã quen với việc điều khiển esp8266 qua các lệnh AT command, tuy nhiên các lệnh AT command không phát huy được hết hoàn toàn khả năng của esp8266. Bản thân esp8266 hỗ trợ lập trình trực tiếp như 1 con vi điều khiển, trong bài này, mình sẽ hướng dẫn các bạn tạo web server cho esp8266

Arduino IDE là một môi trường lập trình với hệ thống thư viện đồ sộ, cộng đồng cực kì lớn mạnh ! Và đương nhiên Arduino IDE cũng hỗ trợ chúng ta lập trình cho esp8266, ngoài arduino ide còn rất nhiều môi trường khác nữa để lập trình cho esp8266, tuy nhiên cộng đồng và thư viện của nó không được mạnh. Do đó, mình sẽ chọn Arduino IDE để viết code

Về việc cài đặt arduino IDE, các bạn tham khảo trên google nhé vì nó quá cơ bản rồi. Sau khi cài xong, chúng ta tích hợp thư viện esp8266 để vào arduino ide như sau: (phần này mình tham khảo [tại đây](#))

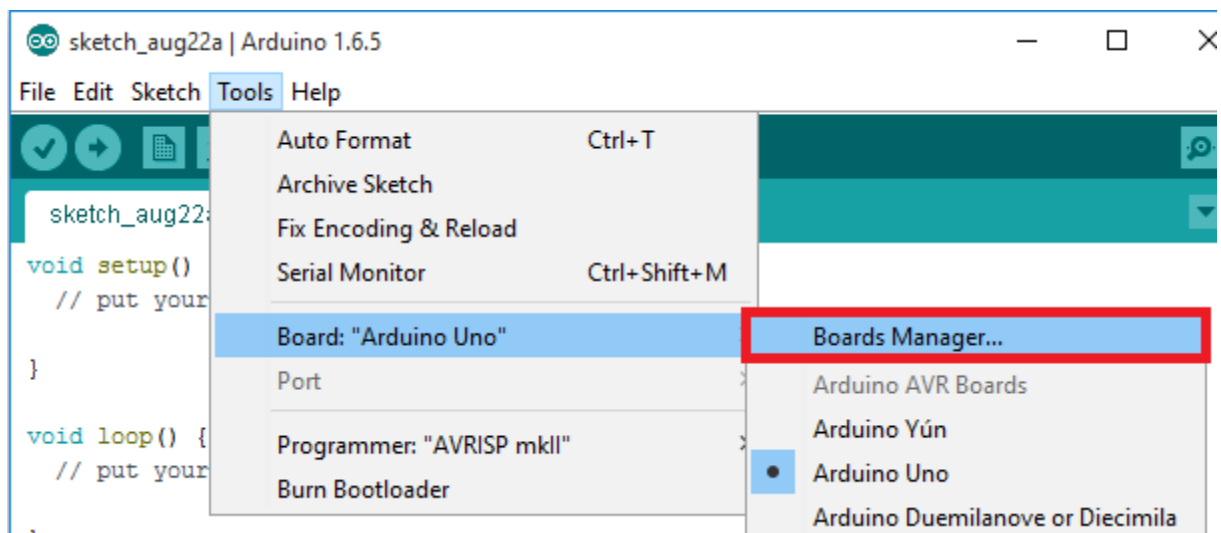
Vào **File**→ **Preferences**, vào textbox **Additional Board Manager URLs** thêm đường link này vào:

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

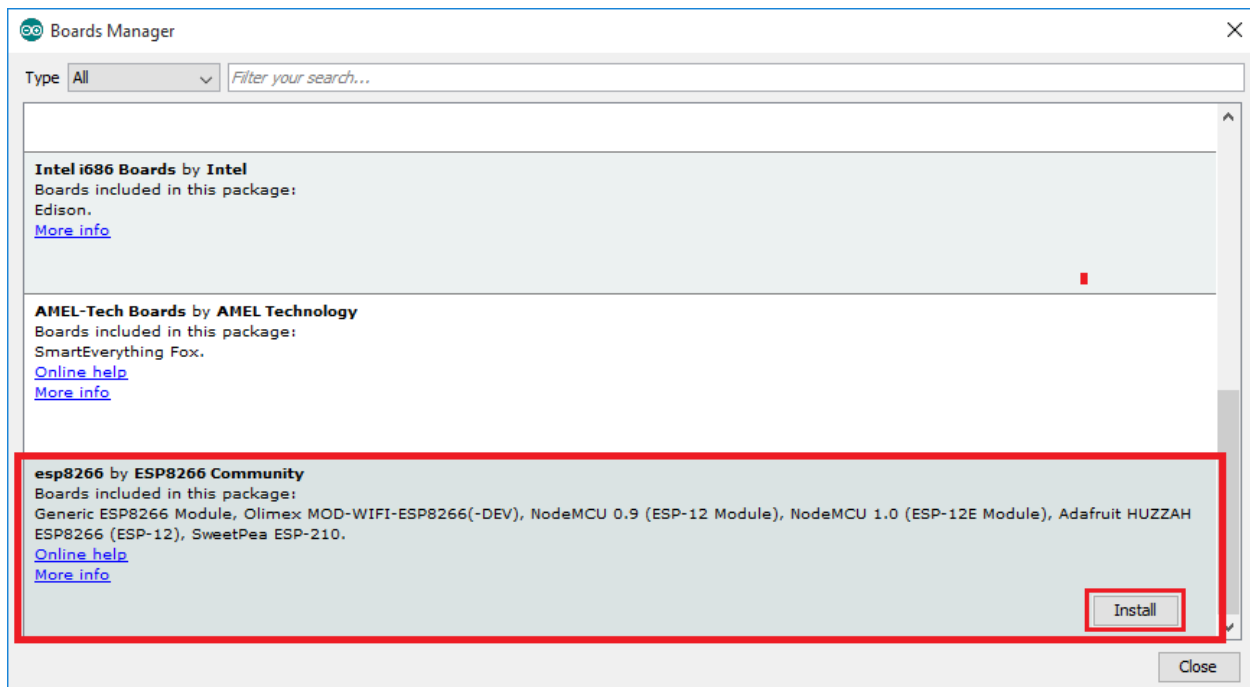


Click **OK** để chấp nhận

Tiếp theo vào **Tool**→**Board**→**Boards Manager**

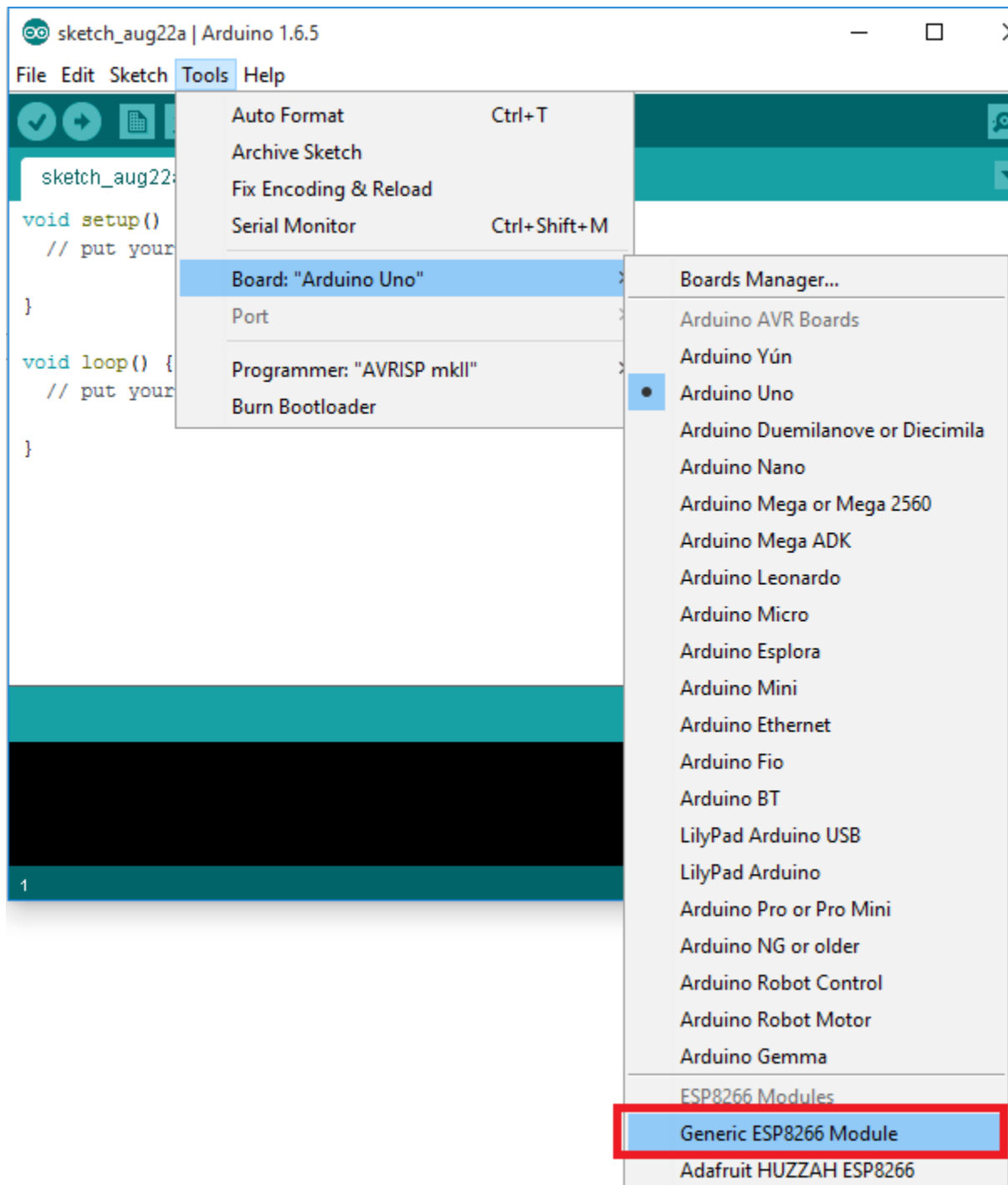


Đợi một lát để chương trình tìm kiếm. Ta kéo xuống và click vào **ESP8266 by ESP8266 Community**, click vào **Install**. Chờ phần mềm tự động download và cài đặt.



Chọn Board để lập trình cho ESP8266:

Kết nối module USB-to-UART vào máy tính. Vào **Tool**→**Board**→**Generic ESP8266 Module**, chọn cổng COM tương ứng với module USB-to-UART tương ứng.



Chọn chế độ nạp **Arduino as ISP**. Vậy là ta đã có môi trường lập trình cho esp8266 rất thân thiện.

## Thiết kế webserver



## Ngôn ngữ HTML

**HTML** là chữ viết tắt của **Hypertext Markup Language**. Nó giúp người dùng tạo và cấu trúc các thành phần trong trang web hoặc ứng dụng, phân chia các đoạn văn, heading, links, blockquotes ...

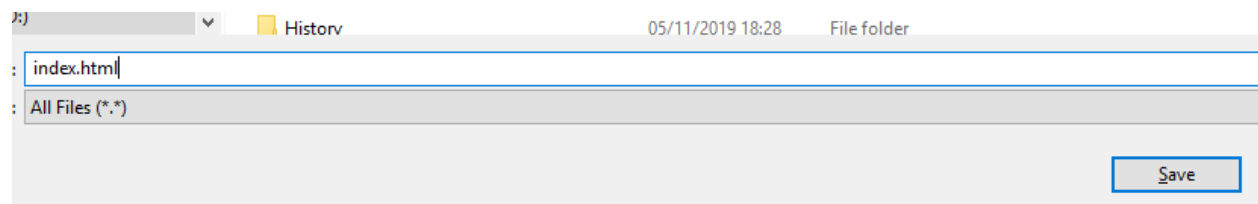
Khi làm việc với **HTML**, chúng ta sẽ sử dụng cấu trúc code đơn giản (tags và attributes) để đánh dấu lên trang web. Ví dụ, chúng ta có thể tạo một đoạn văn bằng cách đặt văn bản vào trong cặp tag mở và đóng văn bản `<p>` và `</p>`

Các bạn tự tìm hiểu thêm nhé,

- <https://thachpham.com/web-development/html-css/html-la-gi-va-vi-sao-no-quan-trong.html>
- <https://www.hostinger.vn/huong-dan/html-la-gi/>
- <https://vi.wikipedia.org/wiki/HTML>
- <https://wiki.matbao.net/kb/html-la-gi-nen-tang-lap-trinh-web-cho-nguoi-moi-bat-dau/>

Trước khi đi vào lập trình, chúng ta sẽ viết mã **html** cho server web bằng máy tính trước, giới thiệu với các bạn phần mềm **Sublime Text** đây là 1 công cụ text editor mà mình rất thích, giao diện đẹp, chuyên nghiệp, hệ thống nhắc lệnh rất ok ! Mình sẽ viết demo mã **html** trên phần mềm này !

Các bạn vào **File -> New** tạo 1 file mới và lưu với đuôi `.html`, ở đây mình lưu file với tên `index.html`



Tiến hành viết mã code html

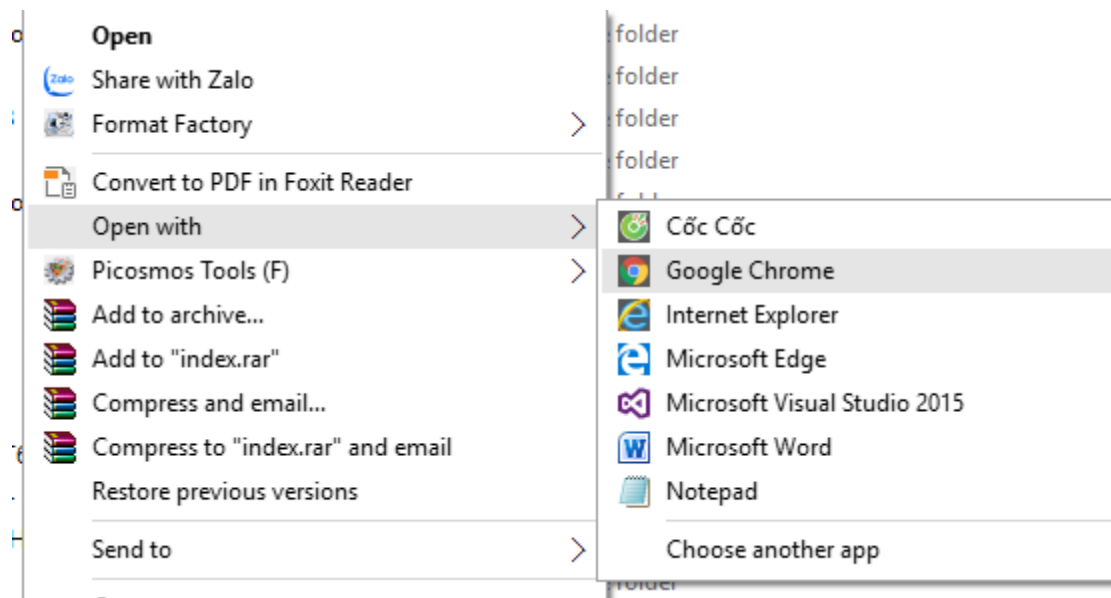


```

10         .t{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}
11     </style>
12</head>
13<body>
14<div style="width: 330px;height: auto;margin: 0 auto;margin-top: 70px">
15<h1 align="center">Điều khiển thiết bị qua WIFI</h1>
16     <table align="center">
17         <tr>
18             <td><a href="/bat1"><button class="b">Bật 1</button></a><td>
19             <td><a href="/tat1"><button class="t">Tắt 1</button></a><td>
20         <tr>
21         <tr>
22             <td><a href="/bat2"><button class="b">Bật 2</button></a><td>
23             <td><a href="/tat2"><button class="t">Tắt 2</button></a><td>
24         <tr>
25         <tr>
26             <td><a href="/bat3"><button class="b">Bật 3</button></a><td>
27             <td><a href="/tat3"><button class="t">Tắt 3</button></a><td>
28         <tr>
29         <tr>
30             <td><a href="/bat4"><button class="b">Bật 4</button></a><td>
31             <td><a href="/tat4"><button class="t">Tắt 4</button></a><td>
32         <tr>
33     </table>
34</div>
35</body>
</html>

```

Sau đó lưu lại và mở file đó lên bằng trình duyệt WEB nhé, ở đây mình dùng Chrome

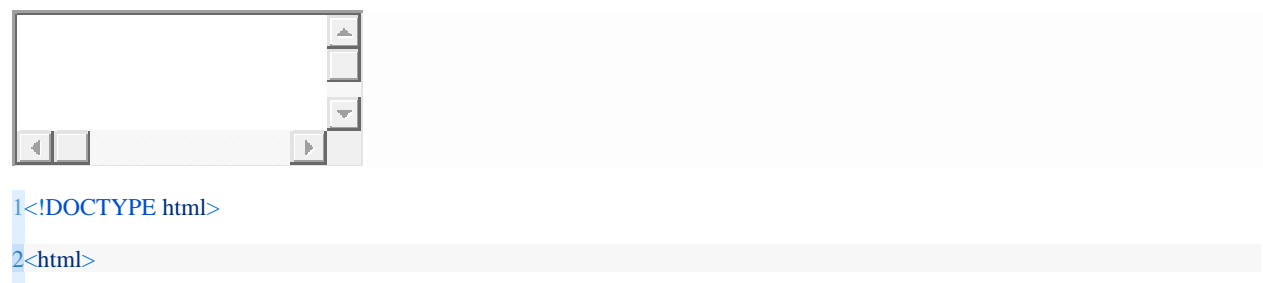


Và đây là kết quả:



Giải thích:

Cấu trúc của 1 file html như sau:



```
3<head>
```

```
4    <title></title>
```

```
5</head>
```

```
6<body>
```

```
7
```

```
8</body>
```

```
9</html>
```

`<!DOCTYPE html>` là tiêu đề bắt buộc mở đầu file

Cặp thẻ `<html> </html>` cũng là cặp thẻ bắt buộc. Tiếp đến là cặp thẻ `<head> </head>` nơi chứa nhưng khai báo mở đầu của trang, ví dụ thêm thư viện, ...

Thẻ `<title> </title>` chính là phân nội dung in trên tab của trình duyệt web (cạnh favicon)

Và cặp thẻ `<body> </body>` nơi chứa nội dung của trang web

Tiếp đến

`<meta http-equiv="Content-Type" content="text/html; charset=utf-8">` khai báo định dạng text – kiểu utf-8

`<title>Điều khiển thiết bị</title>` Khai báo phần title trên tab của trình duyệt web

`<meta name="viewport" content="width=device-width, initial-scale=1">` đây là đoạn mã để trình duyệt web tự động căn chỉnh trang cho vừa nhìn với màn hình (để vừa đẹp với cả điện thoại lẫn máy tính)

`<style>`

`.b{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius: 10px;}`

`.t{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}`

`</style>`

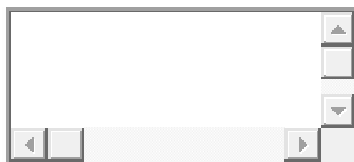
Trong cặp thẻ style mình khai báo 2 class “b” và “t” và viết các thuộc tính như chiều dài, chiều rộng, màu sắc cho nút nhấn

Và ở phần thẻ body chính là nội dung chính của trang web với các button

Sau khi đã biên tập file html xong, chúng ta tiến hành viết code trên arduino và nhúng mã html vào esp8266

Server web ở chế độ Access Point

Ở chế độ Access Point, mình sẽ cho wifi tự thân phát ra wifi để các thiết bị khác kết nối tới



```
1 #include <ESP8266WiFi.h>
```

```
2 #include <ESP8266WebServer.h>
```

```
3 #include <ESP8266mDNS.h>
```

```
4
```

```
5 ESP8266WebServer server(80);
6
7 String webPage =
8 {
9  "<!DOCTYPE html>"
10 "<html>"
11 "<head>"
12 "  <meta http-equiv='Content-Type' content='text/html; charset=utf-8>"
13 "  <title>Điều khiển thiết bị</title>"
14 "  <meta name='viewport' content='width=device-width, initial-scale=1>"
15 "  <style>"
16 "    .b{ width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius: 10px;}"
17 "    .t{ width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}"
18 "  </style>"
19 "</head>"
20 "<body>"
21 "<div style='width: 330px;height: auto;margin: 0 auto;margin-top: 70px>"
22 "<h1 align='center'>Điều khiển thiết bị qua WIFI</h1>"
23 "  <table align='center'>"
24 "    <tr>"
25 "      <td><a href='/bat1'><button class='b'>Bật 1</button></a><td>"
26 "      <td><a href='/tat1'><button class='t'>Tắt 1</button></a><td>"
27 "    <tr>"
28 "      <tr>"
29 "      <td><a href='/bat2'><button class='b'>Bật 2</button></a><td>"
30 "      <td><a href='/tat2'><button class='t'>Tắt 2</button></a><td>"
31 "    <tr>"
32 "      <tr>"
33 "      <td><a href='/bat3'><button class='b'>Bật 3</button></a><td>"
34 "      <td><a href='/tat3'><button class='t'>Tắt 3</button></a><td>"
35 "    <tr>"
36 "      <tr>"
37 "      <td><a href='/bat4'><button class='b'>Bật 4</button></a><td>"
```

```

38 " <td><a href='/tat4'><button class='t'>Tắt 4</button></a><td>"
39 " <tr>"
40 " </table>"
41 "</div>"
42 "</body>"
43 "</html>"
44};
45void TrangChu()
46{
47 server.send(200, "text/html", webPage);
48}
49void setup()
50{
51 Serial.begin(9600);
52 while (WiFi.softAP("ESP8266 WiFi", "12345678") == false)
53 {
54 Serial.print(".");
55 delay(300);
56 }
57 IPAddress myIP = WiFi.softAPIP();
58 server.on("/", TrangChu);
59 server.begin();
60}
61void loop()
62{
63 server.handleClient();
64}

```

Sau khi include các file thư viện cần thiết, ở dòng số 3, mình khởi tạo 1 đối tượng server hoạt động ở cổng 80. Tiếp đến nhúng toàn bộ đoạn code web đã viết vào biến **String webPage**

*Lưu ý: Ở đầu và cuối mỗi dòng chúng ta nhé thêm 1 dấu nháy đôi “ và tất cả các dấu nháy đôi phía trong phải sửa thành dấu nháy đơn ‘*

*Bạn cũng có thể thêm dấu xỏ \ trước nháy đôi thay vì sửa thành nháy đơn*

Trong hàm setup chúng ta khởi tạo cổng serial với tốc độ baud 9600 và cho esp8266 phát ra wifi với tên **ESP8266 WiFi** và password **12345678**

Hàm **server.on()**, yêu cầu esp tạo ra 1 callback tới hàm tên là **TrangChu** khi có thiết bị truy cập tới, và ở hàm **TrangChu**, mình sẽ cho esp8266 trả lời lại mã html với code reponse là 200 (http code 200 báo truy vấn thành công – mình đã nói ở **bài 4**)

## Nạp chương trình:

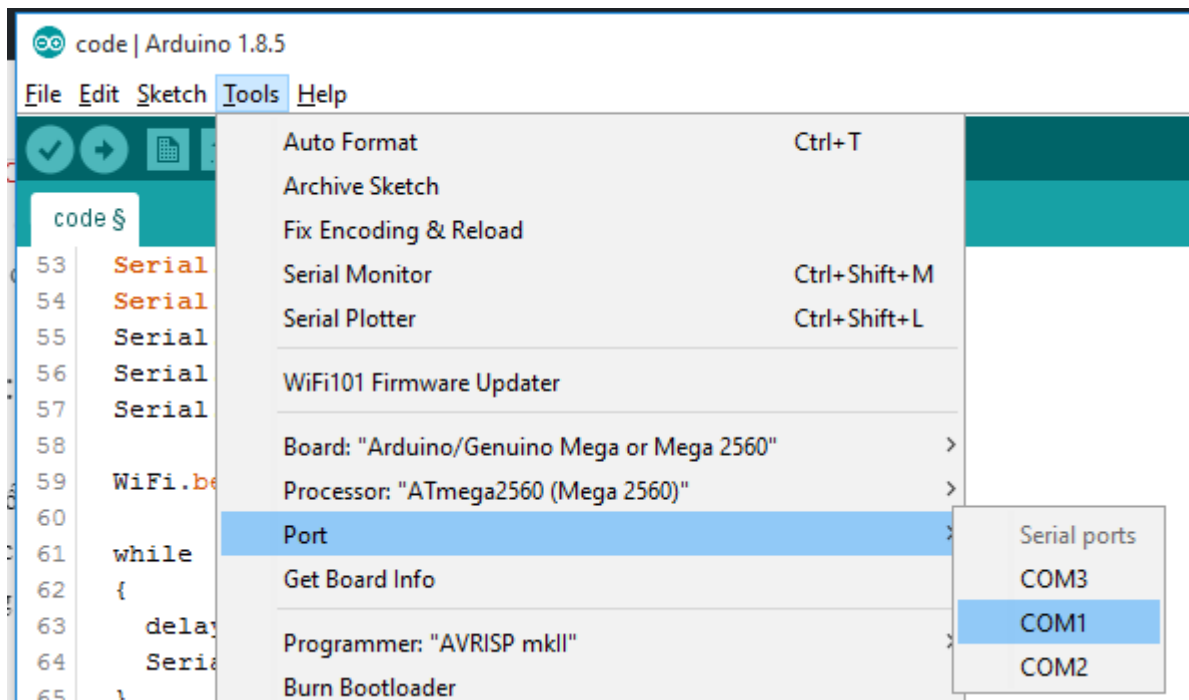
Cách nạp chương trình giống như nạp firmware mà mình đã hướng dẫn ở **bài 2**. Tuy nhiên chúng ta sẽ nạp trực tiếp bằng phần mềm arduino luôn !

Các bạn chuẩn bị 1 module UART-USB như pl2303, hc340 rồi kết nối như sau:

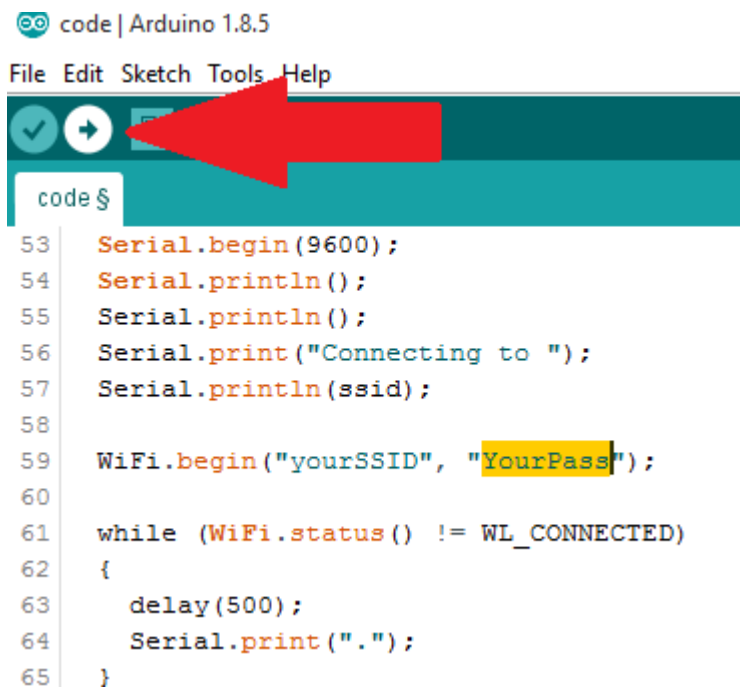
ESP8266	PL2303 (hoặc module uart bất kì)
3.3V	3.3V
GND	GND
RX	TX
TX	RX
CH_PD	3.3V
GPIO0	GND

Sau khi kết nối như trên xong thì chúng ta mới cắm module uart vào máy tính

Các bạn vào Tool -> Port để chọn cổng COM tương ứng:



Sau đó ấn nút nạp xuống:

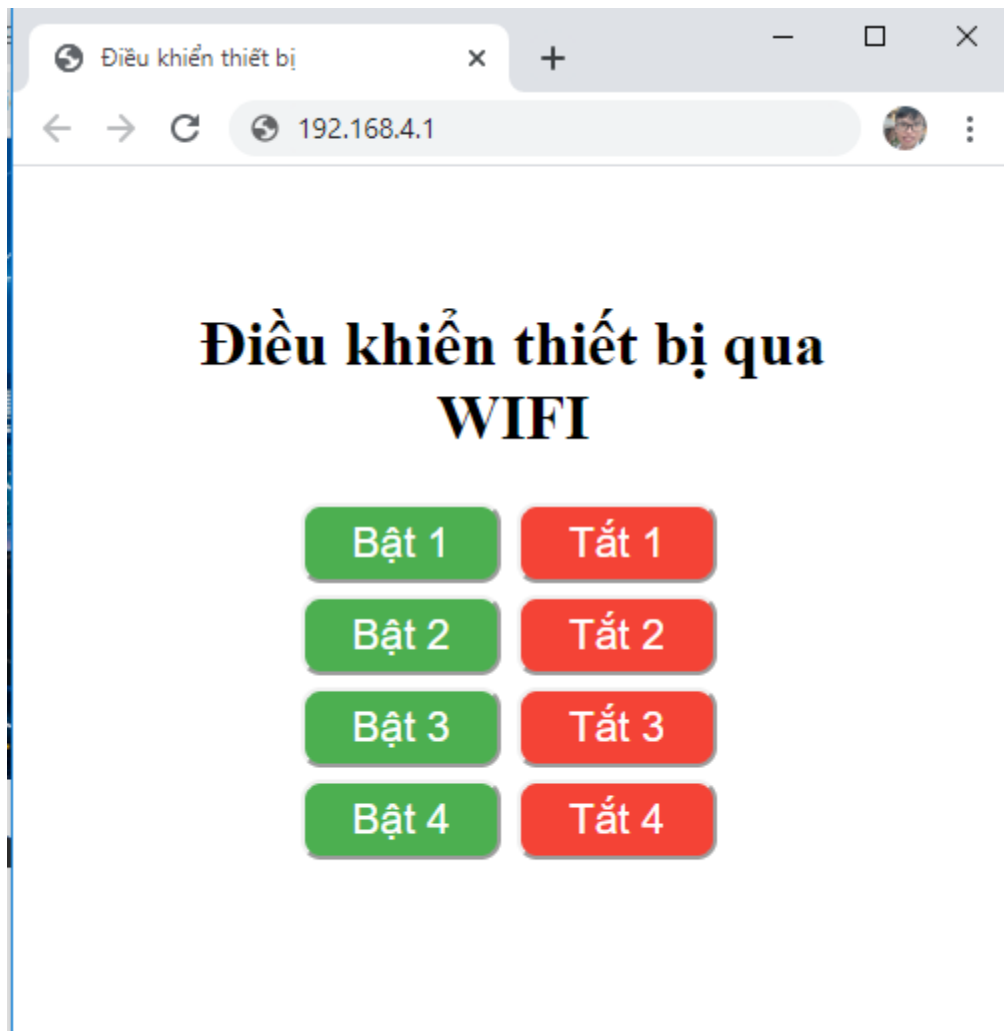


Sau khi nạp code xong thì gỡ GPIO0 ra khỏi mass và reset esp8266 (cấp lại nguồn hoặc kích GND vào chân reset)

Bây giờ lấy máy tính ra, cho nó kết nối vào wifi **ESP8266 WiFi** với mật khẩu **12345678**  
 Sau đó mở trình duyệt gõ địa chỉ mặc định của server là **192.168.4.1**



Chúng ta có kết quả



Server web ở chế độ Station

Ở chế độ station, esp8266 sẽ không phát wifi nữa mà truy cập vào wifi ở nhà



```
1 #include <ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <ESP8266WebServer.h>
4 #include <ESP8266mDNS.h>
5
```

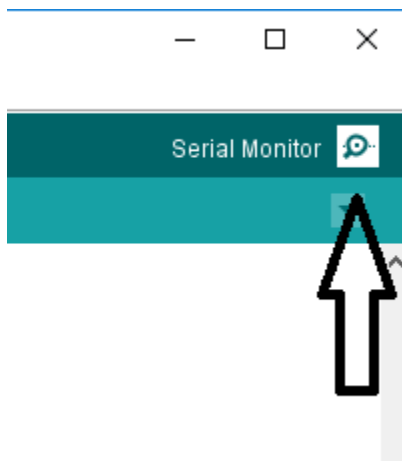
```
6 MDNSResponder mdns;
7 ESP8266WebServer server(80);
8
9 String webPage =
10 {
11     "<!DOCTYPE html>"
12     "<html>"
13     "<head>"
14     "  <meta http-equiv='Content-Type' content='text/html; charset=utf-8'>"
15     "  <title>Điều khiển thiết bị</title>"
16     "  <meta name='viewport' content='width=device-width, initial-scale=1'>"
17     "  <style>"
18     "    .b{ width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius: 10px;}"
19     "    .t{ width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}"
20     "  </style>"
21     "</head>"
22     "<body>"
23     "<div style='width: 330px;height: auto;margin: 0 auto;margin-top: 70px'>"
24     "<h1 align='center'>Điều khiển thiết bị qua WIFI</h1>"
25     "  <table align='center'>"
26     "    <tr>"
27     "      <td><a href='/bat1'><button class='b'>Bật 1</button></a><td>"
28     "      <td><a href='/tat1'><button class='t'>Tắt 1</button></a><td>"
29     "    </tr>"
30     "    <tr>"
31     "      <td><a href='/bat2'><button class='b'>Bật 2</button></a><td>"
32     "      <td><a href='/tat2'><button class='t'>Tắt 2</button></a><td>"
33     "    </tr>"
34     "    <tr>"
35     "      <td><a href='/bat3'><button class='b'>Bật 3</button></a><td>"
36     "      <td><a href='/tat3'><button class='t'>Tắt 3</button></a><td>"
37     "    </tr>"
38     "  </table>"
```

```
39 " <td><a href='/bat4'><button class='b'>Bật 4</button></a><td>"
40 " <td><a href='/tat4'><button class='t'>Tắt 4</button></a><td>"
41 " <tr>"
42 " </table>"
43 "</div>"
44 "</body>"
45 "</html>"
46};
47void TrangChu()
48{
49 server.send(200, "text/html", webPage);
50}
51void setup()
52{
53 Serial.begin(9600);
54 Serial.println();
55 Serial.println();
56 Serial.print("Connecting to ");
57 WiFi.begin("Tang5", "12345678");
58
59 while (WiFi.status() != WL_CONNECTED)
60 {
61 delay(500);
62 Serial.print(".");
63 }
64 Serial.println("");
65 Serial.println("WiFi connected");
66 Serial.println("IP address: ");
67 Serial.println(WiFi.localIP());
68 if (mdns.begin("esp8266", WiFi.localIP()))
69 Serial.println("MDNS responder started");
70
71 server.on("/", TrangChu);
```

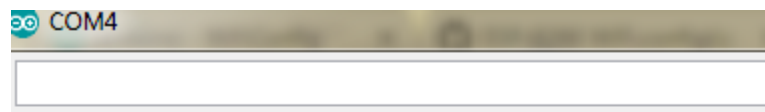
```
72 server.on("/bat1", bat1);
73 server.on("/tat1", tat1);
74 server.on("/bat2", bat2);
75 server.on("/tat2", tat2);
76 server.begin();
77}
78void loop()
79{
80 server.handleClient();
81}
```

Trong đó **yourSSID** và **YourPass** là tài khoản và mật khẩu wifi nhà bạn !

OK bây giờ esp8266 sẽ tự kết nối vào wifi nhà bạn và in ra địa chỉ IP có dạng 192.168.1.xxx Đây là địa chỉ IP mà modem mạng của bạn cấp phát cho nó, bạn chỉ việc gõ địa chỉ ip vào trình duyệt web là trang web như trên sẽ hiện ra

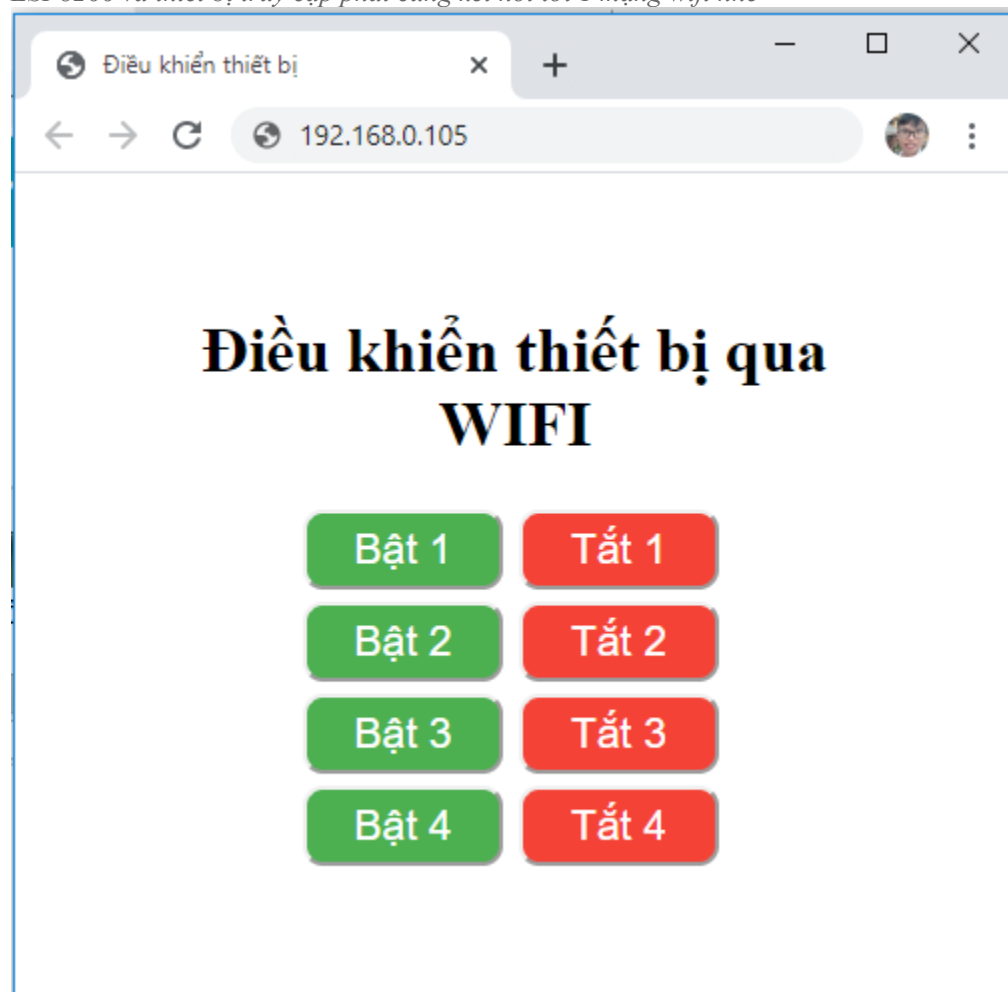


Bật công Serial Monitor lên để xem Log nhé



```
Connecting to daonguyenWF
.....
WiFi connected
IP address:
192.168.0.105
MDNS responder started
HTTP server started
```

*ESP8266 và thiết bị truy cập phải cùng kết nối tới 1 mạng wifi nhé*



Kết quả

# Code chức năng bật tắt LED

Tiếp tục, chúng ta sẽ code thêm chức năng bật tắt led, do ở đây mình dùng esp8266v1 nó chỉ có 2 chân GPIO nên mình sẽ chỉ code chức năng bật tắt cho kênh 1,2. Kênh 3,4 tạm để trạng trí vậy 😊



```
1 #include <ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <ESP8266WebServer.h>
4 #include <ESP8266mDNS.h>
5
6 MDNSResponder mdns;
7 ESP8266WebServer server(80);
8
9 String webPage =
10 {
11   "<!DOCTYPE html>"
12   "<html>"
13   "<head>"
14   "  <meta http-equiv='Content-Type' content='text/html; charset=utf-8'>"
15   "  <title>Điều khiển thiết bị</title>"
16   "  <meta name='viewport' content='width=device-width, initial-scale=1'>"
17   "  <style>"
18   "    .b{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius: 10px;}"
19   "    .t{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}"
20   "  </style>"
21   "</head>"
22   "<body>"
23   "<div style='width: 330px;height: auto;margin: 0 auto;margin-top: 70px>"
24   "<h1 align='center'>Điều khiển thiết bị qua WIFI</h1>"
25   "  <table align='center'>"
26   "    <tr>"
```

```
27 " <td><a href='/bat1'><button class='b'>Bật 1</button></a><td>"
28 " <td><a href='/tat1'><button class='t'>Tắt 1</button></a><td>"
29 " <tr>"
30 " <tr>"
31 " <td><a href='/bat2'><button class='b'>Bật 2</button></a><td>"
32 " <td><a href='/tat2'><button class='t'>Tắt 2</button></a><td>"
33 " <tr>"
34 " <tr>"
35 " <td><a href='/bat3'><button class='b'>Bật 3</button></a><td>"
36 " <td><a href='/tat3'><button class='t'>Tắt 3</button></a><td>"
37 " <tr>"
38 " <tr>"
39 " <td><a href='/bat4'><button class='b'>Bật 4</button></a><td>"
40 " <td><a href='/tat4'><button class='t'>Tắt 4</button></a><td>"
41 " <tr>"
42 " </table>"
43 "</div>"
44 "</body>"
45 "</html>"
46 };
47 void TrangChu()
48 {
49   server.send(200, "text/html", webPage);
50 }
51 void bat1()
52 {
53   digitalWrite(0, HIGH);
54   Serial.println("Bật LED 1");
55   server.send(200, "text/html", webPage);
56 }
57 void tat1()
58 {
59   digitalWrite(0, LOW);
```

```
60 Serial.println("Tắt LED 1");
61 server.send(200, "text/html", webPage);
62 }
63 void bat2()
64 {
65   digitalWrite(2, HIGH);
66   Serial.println("Bật LED 2");
67   server.send(200, "text/html", webPage);
68 }
69 void tat2()
70 {
71   digitalWrite(2, LOW);
72   Serial.println("Tắt LED 2");
73   server.send(200, "text/html", webPage);
74 }
75 void setup()
76 {
77   pinMode(0, OUTPUT);
78   pinMode(2, OUTPUT);
79
80   Serial.begin(9600);
81   Serial.println();
82   Serial.println();
83   Serial.print("Connecting to ");
84   WiFi.begin("Tang5", "12345678");
85
86   while (WiFi.status() != WL_CONNECTED)
87   {
88     delay(500);
89     Serial.print(".");
90   }
91   Serial.println("");
92   Serial.println("WiFi connected");
```



```

93 Serial.println("IP address: ");
94 Serial.println(WiFi.localIP());
95 if (mdns.begin("esp8266", WiFi.localIP()))
96 Serial.println("MDNS responder started");
97
98 server.on("/", TrangChu);
99 server.on("/bat1", bat1);
100 server.on("/tat1", tat1);
101 server.on("/bat2", bat2);
102 server.on("/tat2", tat2);
103 server.begin();
104}
105void loop()
106{
107 server.handleClient();
108}

```

Code trên mình thêm dòng cấu hình ngõ ra cho GPIO0 và GPIO2, sau đó thêm hàm callback nhận lệnh bật tắt từ trình duyệt và xuất tín hiệu HIGH LOW ra chân GPIO và in log ra màn hình Serial

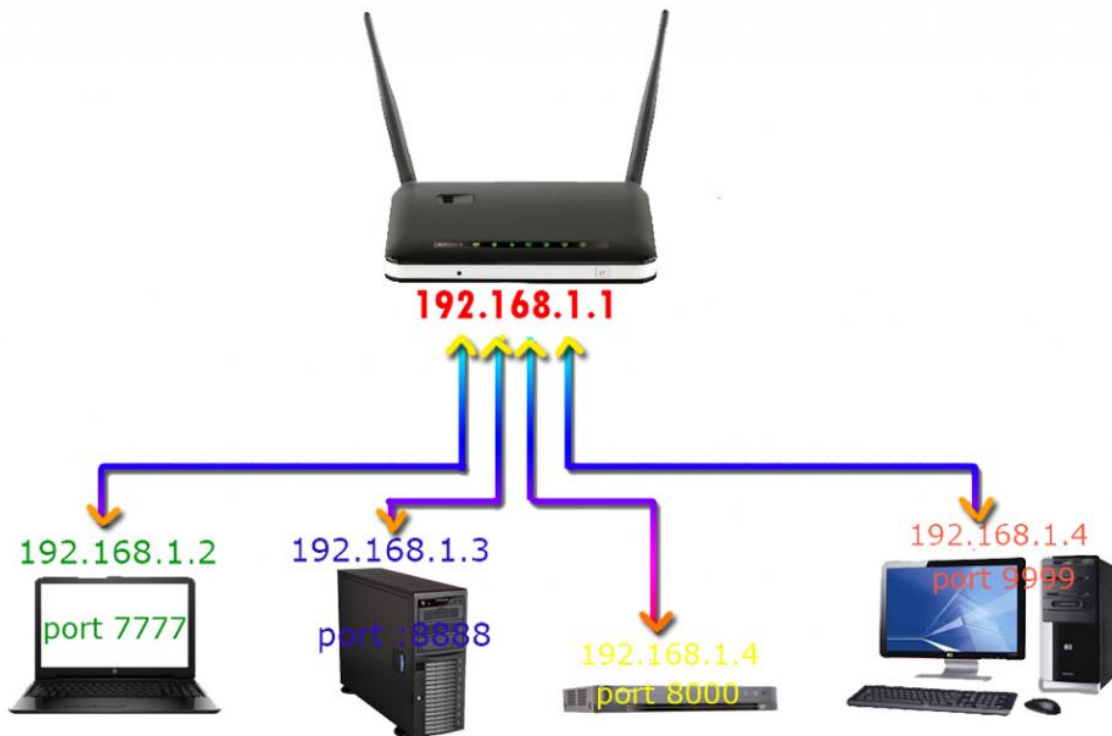
*Mẹo: Có thể tận dụng 2 chân Serial làm GPIO luôn, như thế con espv1 sẽ điều khiển được 4 thiết bị*

## NAT PORT

Chúng ta đã có thể truy cập vào server của esp8266 thông qua mạng LAN trong gia đình, tuy nhiên, nó chỉ là 1 cái server hoạt động cục bộ trong gia đình, nếu muốn đi ra ngoài mà điều khiển được thì cần phải đưa nó ra ngoài thế giới internet rộng lớn.

### IP cục bộ và IP công cộng


Các bạn có thể thấy ở video demo trên, modem mạng nhà mình đã cấp phát cho esp8266 địa chỉ 192.168.1.11, đây chính là IP cục bộ mà modem đã cấp cho esp8266 để nó phân biệt với các thiết bị khác cũng chung mạng wifi ( điện thoại, laptop, smart tivi cũng đều dc cấp 1 cái ip riêng, không thằng nào dùng chung với thằng nào)



Bản thân modem wifi thông thường cũng sẽ lấy địa chỉ 192.168.1.1 hoặc 192.168.0.1 là IP cục bộ mặc định ! Nếu nó được đăng kí internet với nhà mạng (viettel, vina ... ) – tức có sợi dây cáp quang gắn vào đấy – thì nhà mạng sẽ cung cấp cho modem 1 IP công cộng, đây là địa chỉ IP duy nhất để phân biệt modem nhà bạn với hàng tỉ thiết bị internet trên thế giới. Như vậy tất cả thiết bị trong LAN muốn giao tiếp với bên ngoài phải thông qua modem.

Để biết modem của bạn có IP công cộng (public IP) là gì, bạn hãy ấn **vào đây**

[MY IP](#) [IP LOOKUP](#) [HIDE MY IP](#) [VPNS ▾](#) [TOOLS ▾](#) [LEARN ▾](#)



### IP Lookup

Know the IP address of another computer? You can find where in the world it is—and more.



### Trace Email

Track down the geographical location and origin of an email you received.




### Hide My IP

Learn how to use a high-tech "middleman" to shield your real IP address on the Internet.



### VPN Comparison

Compare top rated VPN service providers that meet your needs and budget.



### Blacklist Check

Have you been blacklisted because of the IP address you use? Check to see here.



## My IP Address Is:

IPv4: **117.1.165.234**  
IPv6: Not detected

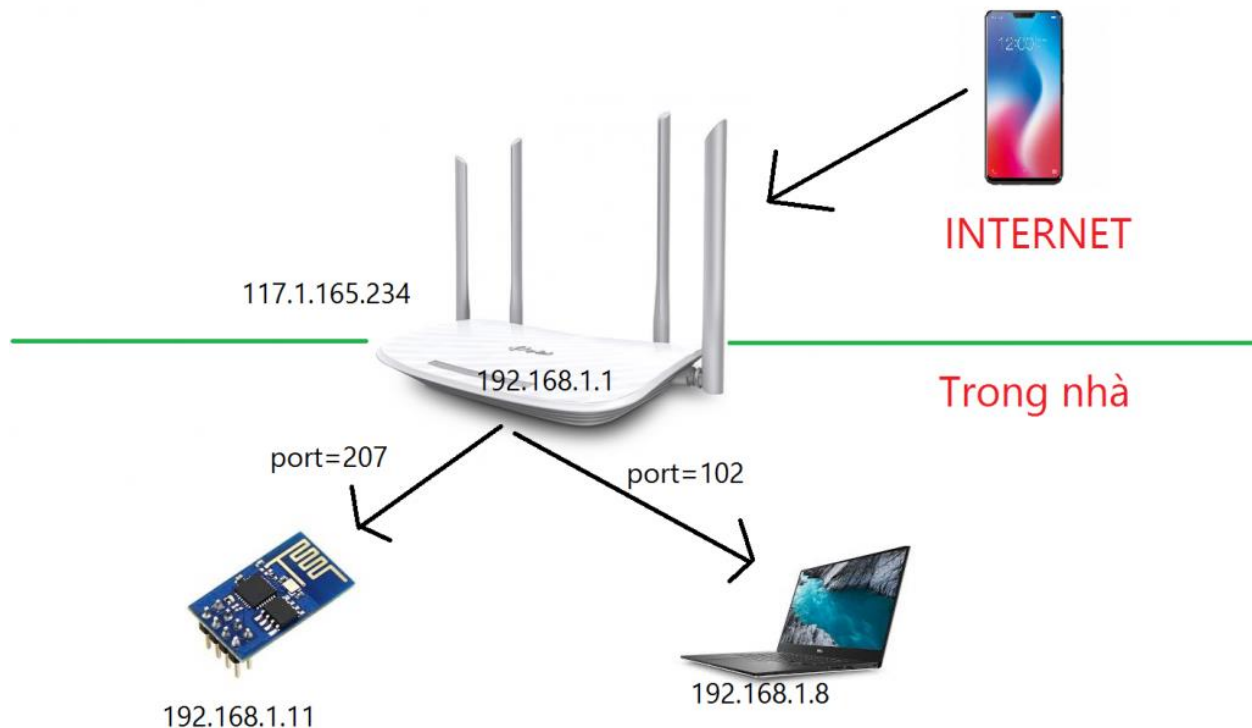
### My IP Information:

ISP: Viettel Group  
City: Hanoi  
Region: Hanoi  
Country: Vietnam

**Hide My IP Address**  
[Click Here](#)



Như ở trên 117.1.165.234 chính là IP của modem wifi nhà mình, bây giờ 1 thiết bị bên ngoài muốn truy cập vào web server của esp8266, nó phải truy cập vào IP này trước, sau đó, nó sẽ thông báo cho modem biết cụ thể là nó muốn truy cập vào esp8266 bằng cách thông báo cổng



Ở bức ảnh trên, laptop và esp8266 đều ở trong LAN và có IP riêng, esp muốn cái điện thoại bên ngoài liên lạc được với nó, nó sẽ nói với modem rằng: “Ê modem, tao đăng kí cổng số 207”, tương tự nếu laptop cũng đưa đòi tạo web server và muốn điện thoại truy cập vào, nó cũng báo với modem cần đăng kí cổng số 102

Bây giờ, khi điện thoại bên ngoài gõ vào trình duyệt **117.1.165.234:207** nó sẽ truy cập được vào esp8266, còn nếu muốn truy cập vào laptop thì nó sẽ vào **117.1.165.234:102**

Hành động khai báo cổng cho modem chính là **NAT PORT**

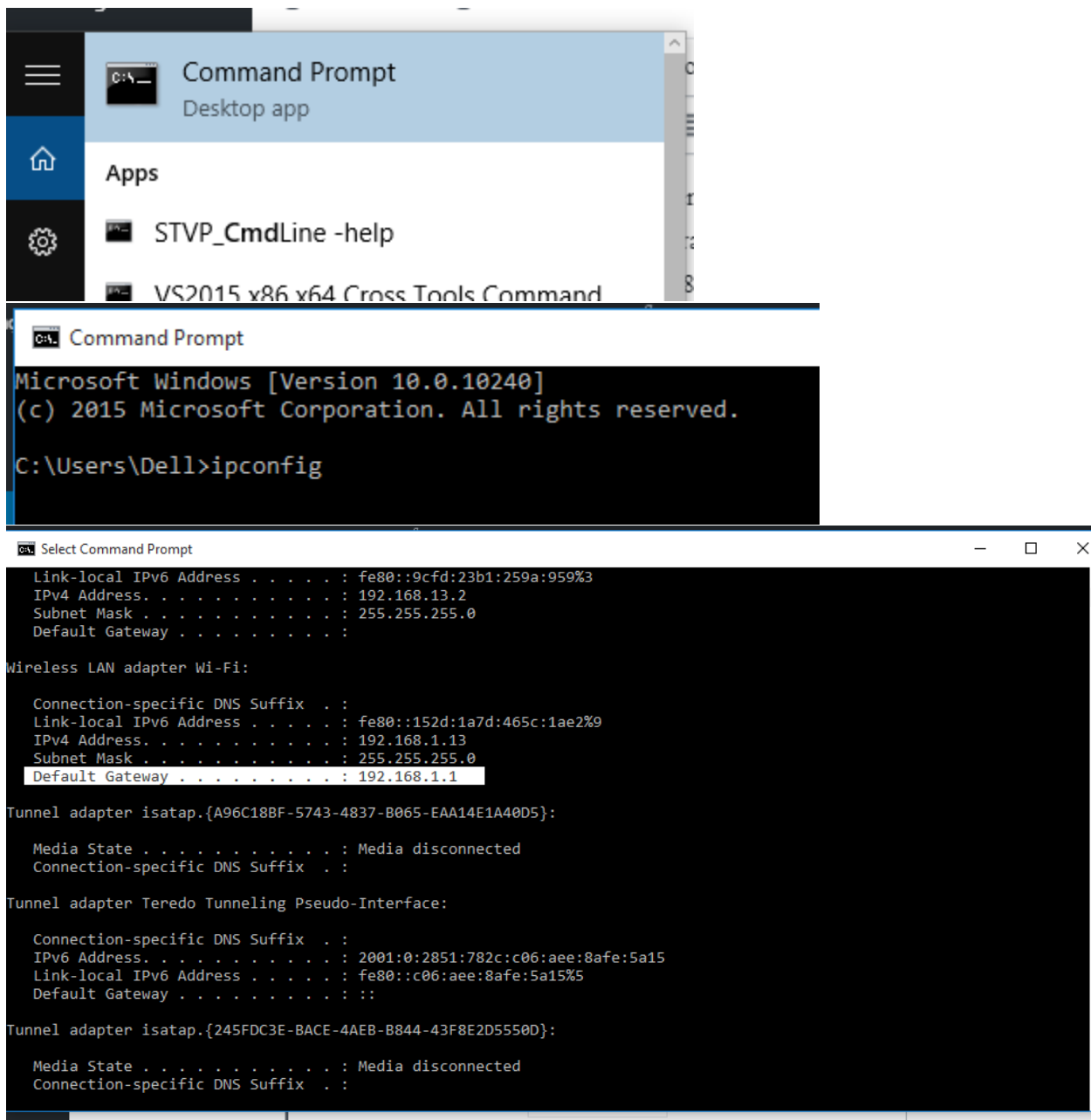
Để **NAT PORT**, có 2 cách chính:

- Port Forwarding, đây là cách mình khuyên các bạn nên xài và mình sẽ hướng dẫn các bạn Port Forwarding
- DMZ, cái này các tự tìm hiểu thêm

## Hướng dẫn NAT PORT – tạo webserver với esp8266

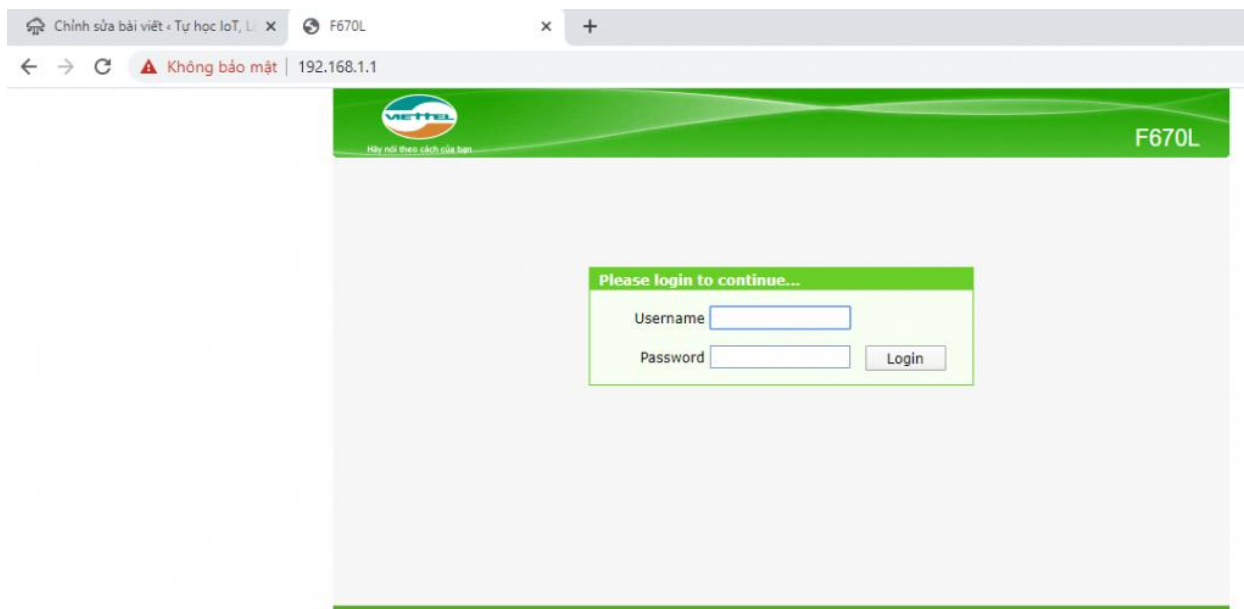
Tùy vào modem mạng của bạn thuộc hãng nào, phiên bản nào mà sẽ có các cách NAT khác nhau, nhưng nhìn chung đều tương tự nhau và mò tí là ra thôi.

Đầu tiên, phải biết được IP cục bộ của modem mạng để tin hành vào trang cài đặt ( chính là cái trang các bạn hay vào để đổi tên, mật khẩu wifi đó), thông thường nó sẽ là 192.168.1.1 hoặc 192.168.0.1 các bạn cứ gõ thử vào trình duyệt để xem cái nào vào được. Nếu cả 2 đều không vào được thì lấy laptop ra, mở cmd lên gõ **ipconfig** rồi tìm đến dòng **Default gateway** sẽ thấy ip của modem



Như ở trên ip modem của mình 192.168.1.1


Sau đó gõ IP đó vào trình duyệt và bạn sẽ được đưa đến trang đăng nhập ( Nguồn tham khảo <https://lapcamera247.com/huong-dan-nat-port-va-cau-hinh-ten-mien-modem-viettel-f608-f600w.html> )



Thông thường tài khoản là **admin** và mật khẩu là **admin** hoặc **123456**. Nếu thử không được thì lật mặt dưới modem wifi lên xem sẽ thấy ! Nếu vẫn không được thì ấn giữ nút reset modem để khôi phục mật khẩu về mặc định



Tài khoản mật khẩu được ghi ở phía sau

  
Hãy nói theo cách của bạn

F600W

+Status

+Network

**-Security**

Firewall

IP Filter

MAC Filter

URL Filter

Service Control

ALG

+Application

+Administration

+Help

?

Path:Security-Firewall

[Logout](#)

Enable Anti-Hacking Protection ☐

Firewall Level

1

2

☒ Off

☐ Low

☐ Middle

☐ High

Submit

Cancel

Tắt tường lửa

VETTEL F600W

Hãy nói theo cách của bạn

Path: Application-Port Forwarding Logout

**+Status**

**+Network**

**+Security**

**-Application**

DDNS

DMZ Host

UPnP

UPnP Port Mapping

**Port Forwarding**

+DNS Service

SNTp

+MultiCast

BPDu

USB Storage

DMS

FTP Application

Port Trigger

Port Forwarding ( Application List )

Application List

Samba Service

USB print server

**+Administration**

**+Help**

Enable ☒

Name camip2p

Protocol TCP AND UDP

WAN Host Start IP Address

WAN Host End IP Address

WAN Connection omci\_ipv4\_pppoe\_1

WAN Start Port 80 (1 ~ 65535)

WAN End Port 80 (1 ~ 65535)

Enable MAC Mapping ☐

LAN Host IP Address 192.168.1.19

LAN Host Start Port 80 (1 ~ 65535)

LAN Host End Port 80 (1 ~ 65535)

Modify Cancel

Enable	Name	WAN Host Start IP Address	WAN Start Port	LAN Host Start Port	WAN Connection	Modify	Delete
<input checked="" type="checkbox"/>	camip2p		80	80	omci_ipv4_pppoe_1		
<input checked="" type="checkbox"/>	TCP AND UDP		80	80	192.168.1.19		
<input checked="" type="checkbox"/>	camip		34567	34567	omci_ipv4_pppoe_1		
<input checked="" type="checkbox"/>	TCP AND UDP		34567	34567	192.168.1.19		

- Tích vào enable
- Name: Đặt tên gì cũng được
- WAN Connection: mặc định
- 4 ô có thông số PORT: điền PORT mà bạn muốn
- LAN Host IP: Chính là IP của esp8266 mà modem đã cấp phát cho

Sau đó save lại

Kiểm tra xem đã thông được ra internet chưa

Vào <https://canyouseeme.org/> gõ IP công cộng của modem và port mà lúc này bạn đã mở, nếu báo OK là ok rồi đó



**Success** I can see your service on **117.1.165.234** on port **(207)**

Reason: Connection refused

Your IP:

Port to Check:

Từ giờ mình có thể đi ra Úc, hay Tây Ban Nha rồi gõ **117.1.165.234:207** là có thể ung dung vào điều khiển các thiết bị trong nhà

## Khắc phục lỗi NAT mãi không được – tạo webserver với esp8266

Chủ yếu có 2 nguyên nhân:

Nguyên nhân 1, là mạng đang dùng là mạng đa lớp



Như hình trên, có thể thấy esp8266 của chúng ta không kết nối trực tiếp với modem wifi gốc mà qua 1 roter tp-link nữa. Vậy nên bạn phải NAT cả cho roter tp-link luôn ( NAT 2 lần). Nếu bạn đang ở phòng trọ thì đến 99% là dùng qua mạng nhiều lớp !

*Cái modem mà của vina, vietel, FPT mà có cái dây cáp quang cắm vào ý, cái ấy mới là modem gốc*

**Nguyên nhân 2 là nhà mạng chưa cập nhật đúng IP công cộng của modem nhà bạn**

Khắc phục rất đơn giản, alo cho tổng đài báo nó reset lại modem wifi (nó reset ở trên tổng chứ không phải cái nút reset ở modem đâu :v )

Bài đã dài, có thể bạn nghĩ tới đây đã hết, nhưng chưa đâu, vào 1 ngày đẹp trời cái web server lại đ\*o vào được >< , Nguyên nhân là do khi modem wifi bị mất điện hay khởi động lại nó sẽ được nhà mạng phát ip mới, bạn lên trang <https://whatismyipaddress.com/> sẽ thấy cái ip mới hết >< Để khắc phục vấn đề này chúng ta cần tới **DDNS**

## Hướng dẫn cấu hình DDNS – tạo webserver với esp8266

Hiểu nôm na DDNS là 1 tên miền để bạn gõ vào trình duyệt thay vì gõ mấy cái số ip khô khan. Lúc nhà mạng phát ip công cộng mới cho modem ta vẫn vào được.

Vào trang <https://www.noip.com/> đăng kí tài khoản và đăng kí 1 tên miền có dạng yourname.ddns.com, mà ngày trước nó free giờ nó đòi phí hay sao ấy, để rảnh mình kiểm cái trang free khác up lên sau

Sau khi đăng kí xong và đã có tên miền riêng thì quay trở lại rang quản lí của modem wifi, tìm đến phần DDNS setting và điền thông tin tên miền + user name + password mà bạn vừa đăng kí vào

Path: Application-DDNS	Logout
<div><div><div>+Status</div><div>+Network</div><div>+Security</div><div>-Application</div><div>DDNS</div><div>DMZ Host</div><div>UPnP</div><div>UPnP Port Mapping</div><div>Port Forwarding</div><div>+DNS Service</div><div>SNTP</div><div>+MultiCast</div><div>BPDU</div><div>CATV</div><div>Port Trigger</div><div>Port Forwarding (</div></div></div> <div><div>Enable <input checked="" type="checkbox"/></div><div>Service Type <div>No-IP</div></div><div>Server <div>http://www.no-ip.com</div></div><div>Username <div>cctv01</div></div><div>Password <div>*****</div></div><div>WAN Connection <div>omci_ipv4_pppoe_1</div></div><div>Hostname <div></div></div></div>	

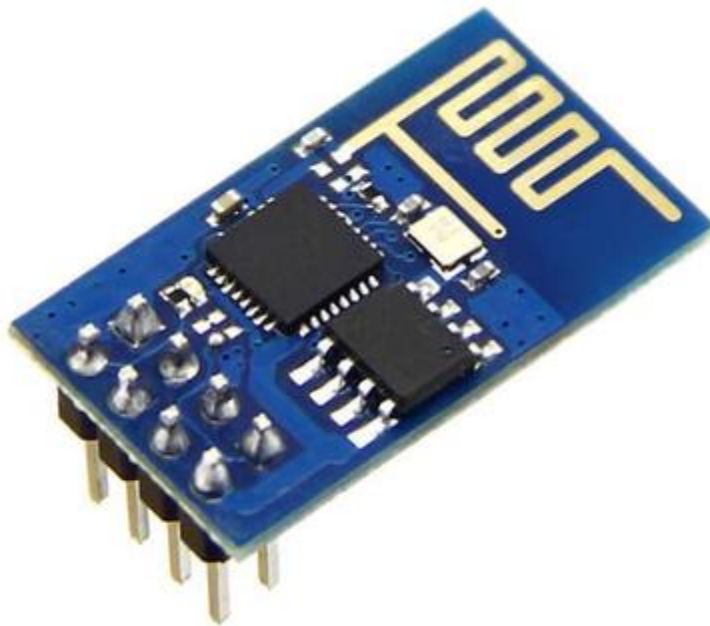
Vậy là xong, giờ chúng ta có thể an tâm gõ tên miền kèm theo đuôi port phía sau là có thể điều khiển ngôi nhà của mình rồi !

Ví dụ đây là địa chỉ esp8266 mà mình đã nat và cấu hình ddns: **iot47-test.ddns.net:80**



# [IoT] Bài 3: ESP8266 Demo ứng dụng điều khiển LED từ xa qua internet bằng tập lệnh AT esp8266

9 Tháng Tám, 2019 Đào Nguyễn IoT tutorial, WIFI-ESP8266 19



Bài này chúng ta sẽ xây dựng 1 ứng dụng demo nho nhỏ để điều khiển led qua internet bằng tập lệnh AT – mode Station. Nguyên lý hoạt động khá đơn giản. ESP8266 sẽ gửi 1 truy vấn (request) tới 1 trang web, trang web này sẽ trả về reponse có chứa từ khóa mô tả trạng thái ON hay OFF của đèn LED.

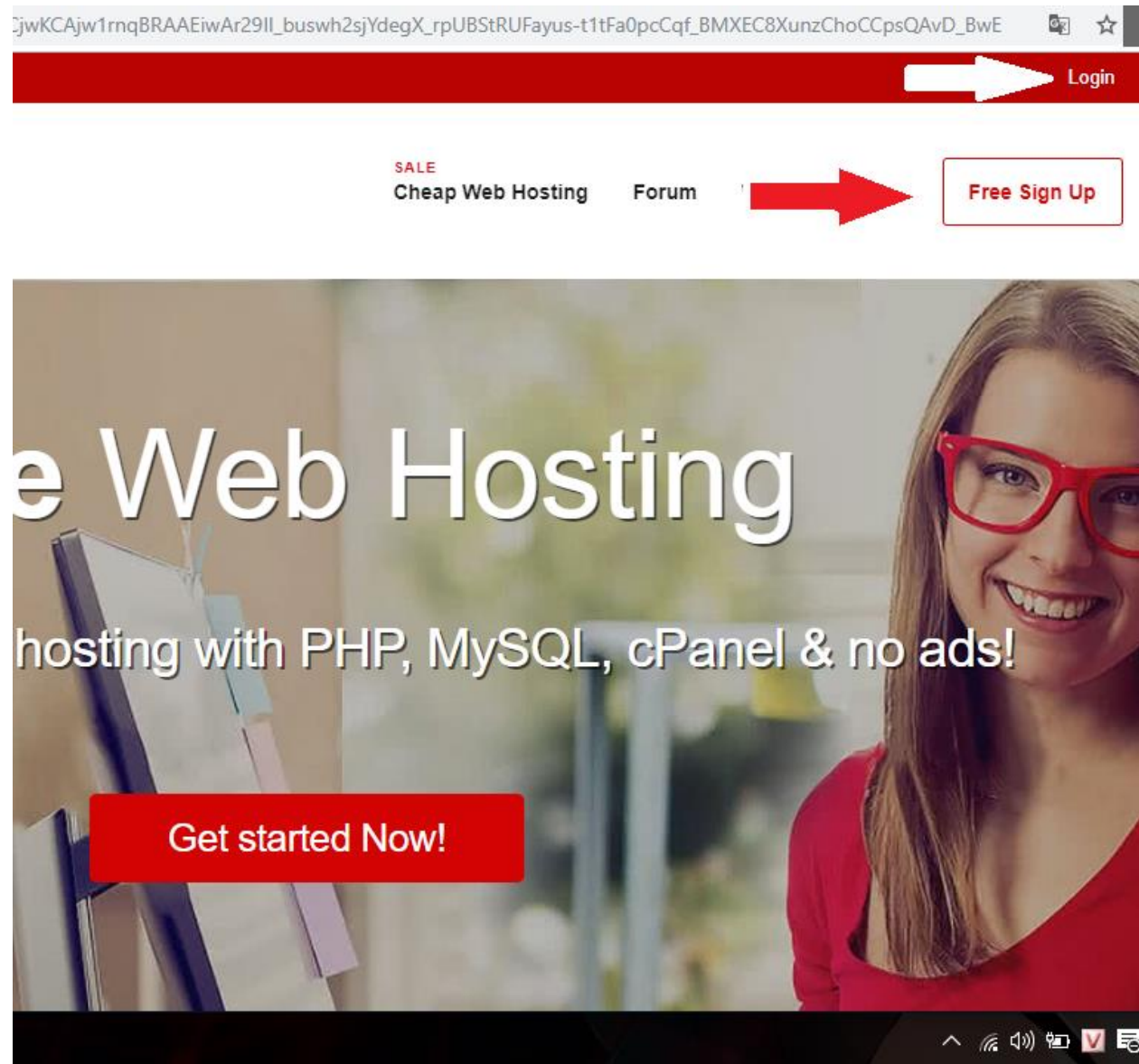
Các bạn đọc kĩ hơn về tập lệnh at esp8266 tại **bài trước**

Hmm... thế trang web này chúng ta sẽ lấy ở đâu ra ? Bình thường se phải đi thuê từ các nhà cung cấp dịch vụ như Mắt Bão, Supper host, Zcom.... như blog này của mình cũng đi thuê nên mình có thể dùng được luôn. Nhưng mình sẽ hướng dẫn các bạn sử dụng dịch vụ cung cấp website free để làm demo này. Có 2 trang web cung cấp web free khá phổ biến mà mình biết đó là:

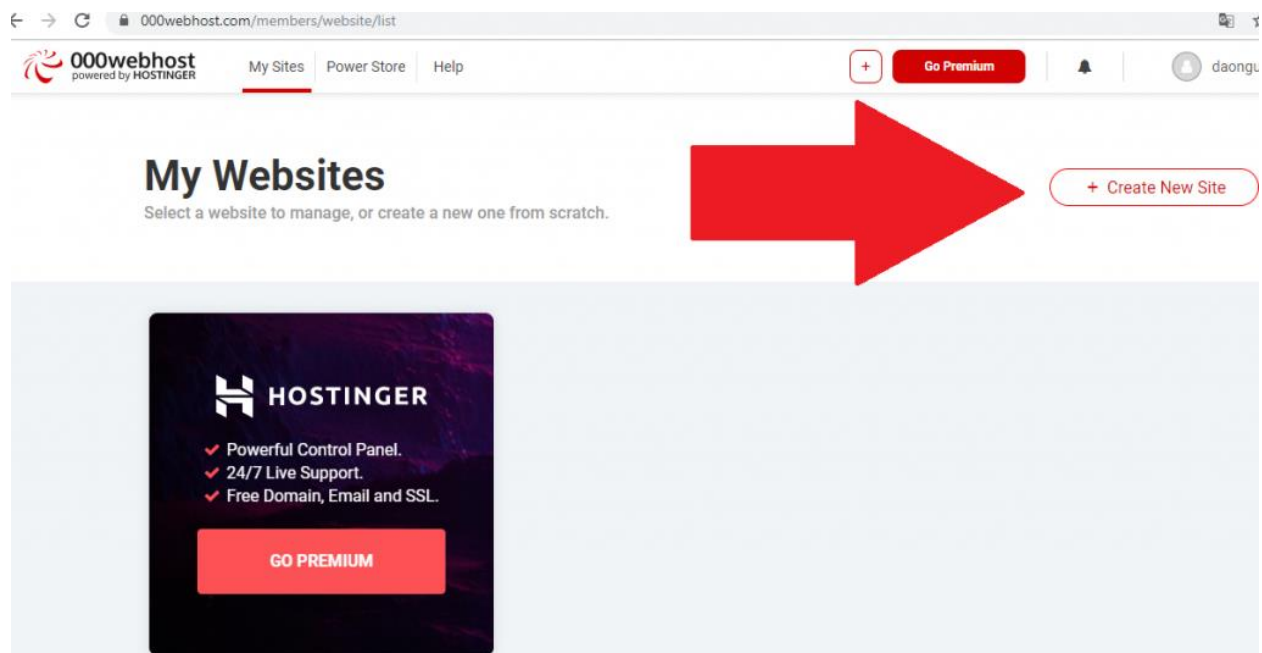
- <https://www.000webhost.com>
- <https://infinityfree.net/>

Mình sẽ sử dụng dịch vụ của **000WEB** cho project này này !

## Đăng kí tài khoản và đăng nhập



Đăng nhập hoặc đăng kí tài khoản nếu chưa có



Các bạn Click vô đây để tạo 1 Web site mới nhé


The image shows a modal window titled 'Website Mới' (New Website). It contains two input fields: 'Tên website (tùy chọn)' (Website Name) with the value 'iot47' and 'Mật khẩu' (Password) which is masked with dots. Below the password field is a checkbox labeled 'Hiện mật khẩu' (Show password) and a red link 'TẠO MẬT KHẨU KHÁC' (Create different password). A red 'Tạo' (Create) button is located at the bottom right of the modal.

Nhập tên web và password của bạn để vào trang admin của web

## Website của tôi

Chọn website để quản lý hay tạo mới từ đầu






iot47

Trạng thái đang chạy

<https://iot47.000webhostapp.com/>

Nâng cấp



Website của mình vừa được tạo

Sau đó click CREATE để tiến hành tạo website. Như thế là xong. Mình đã có 1 web tên là <https://iot47.000webhostapp.com/> cho riêng mình. Các bạn có thể gõ vào trình duyệt để test thử !



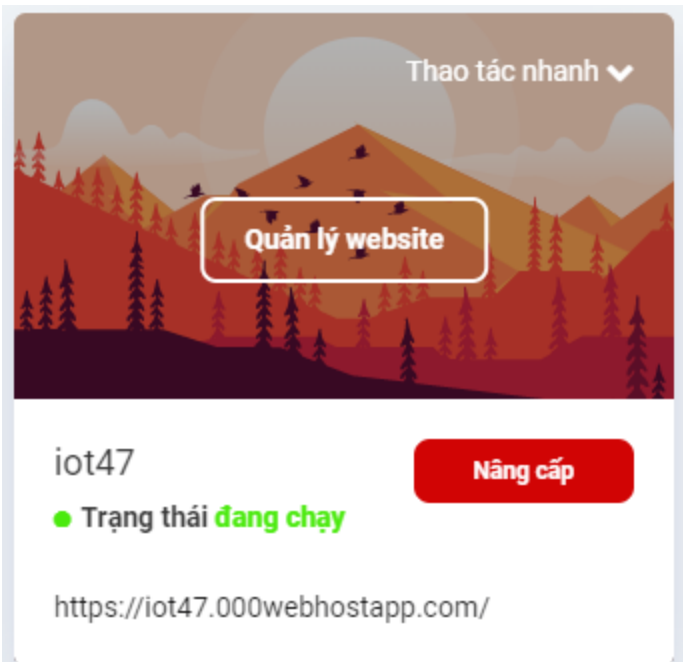
Hooray, your free website has been started!

[iot47.000webhostapp.com](https://iot47.000webhostapp.com)

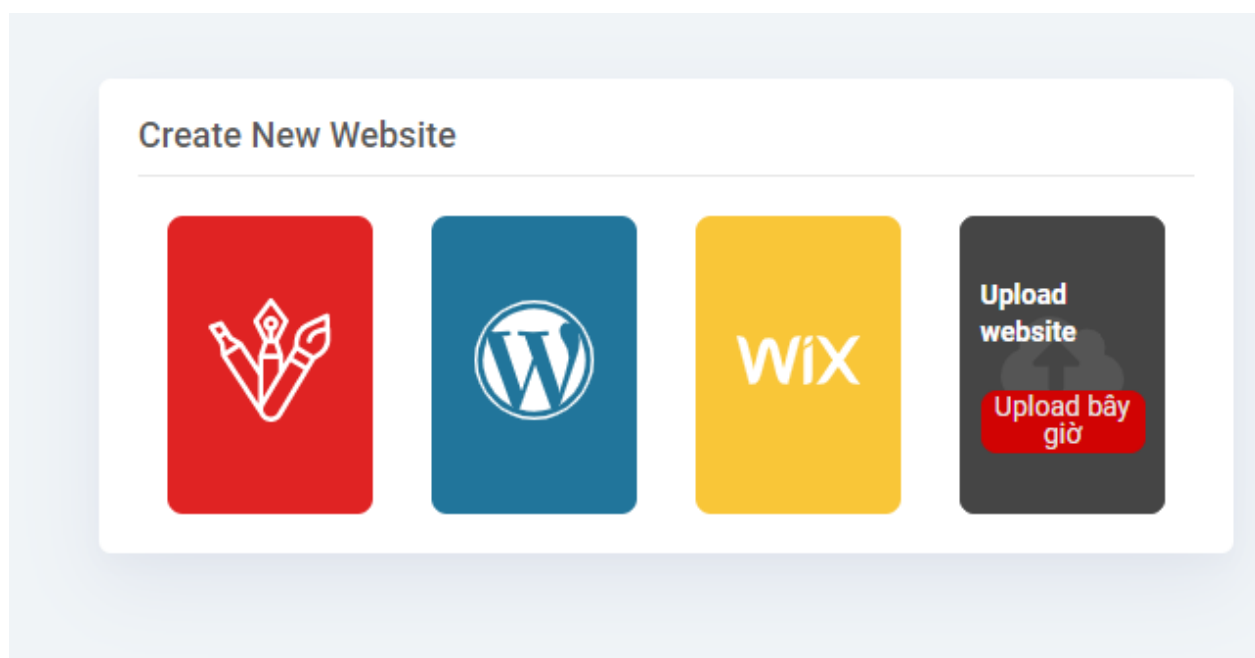
You see this page because your website doesn't have "index.php" or "index.html" file in public\_html folder.  
[Create index file](#)

Khi vừa mới tạo chưa code gì thì vào nó sẽ như này

## Thiết kế giao diện



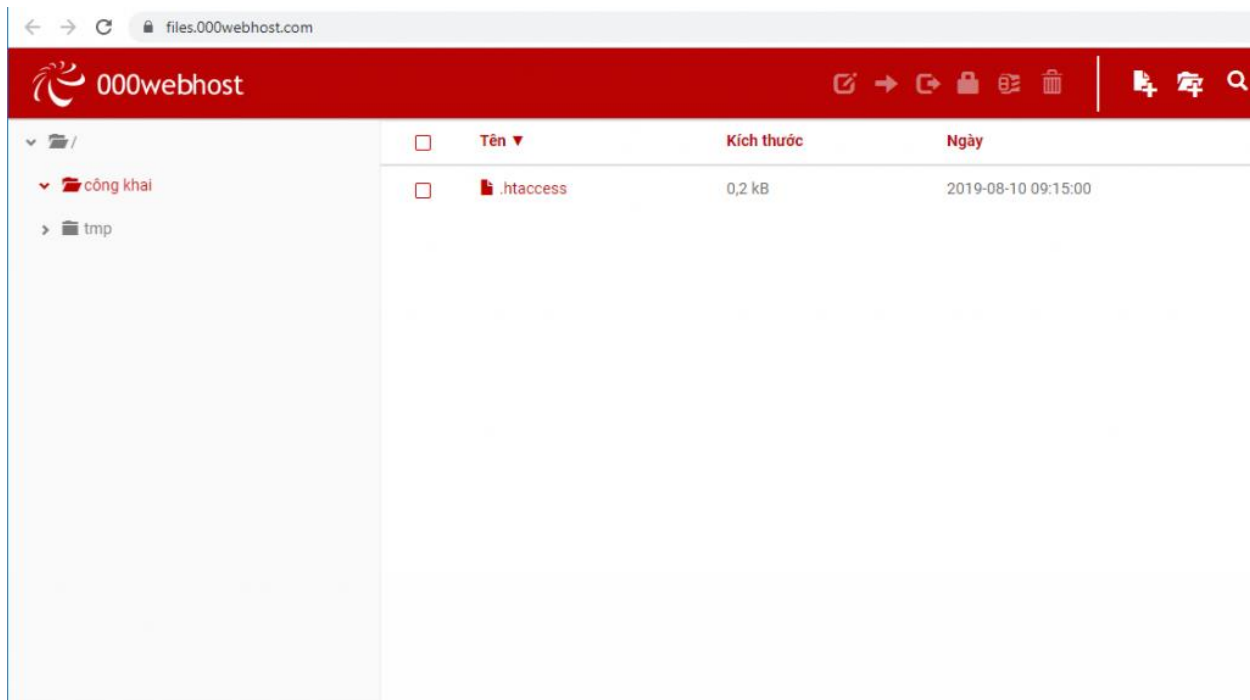
Click vào Quản lý web để vào trang admin



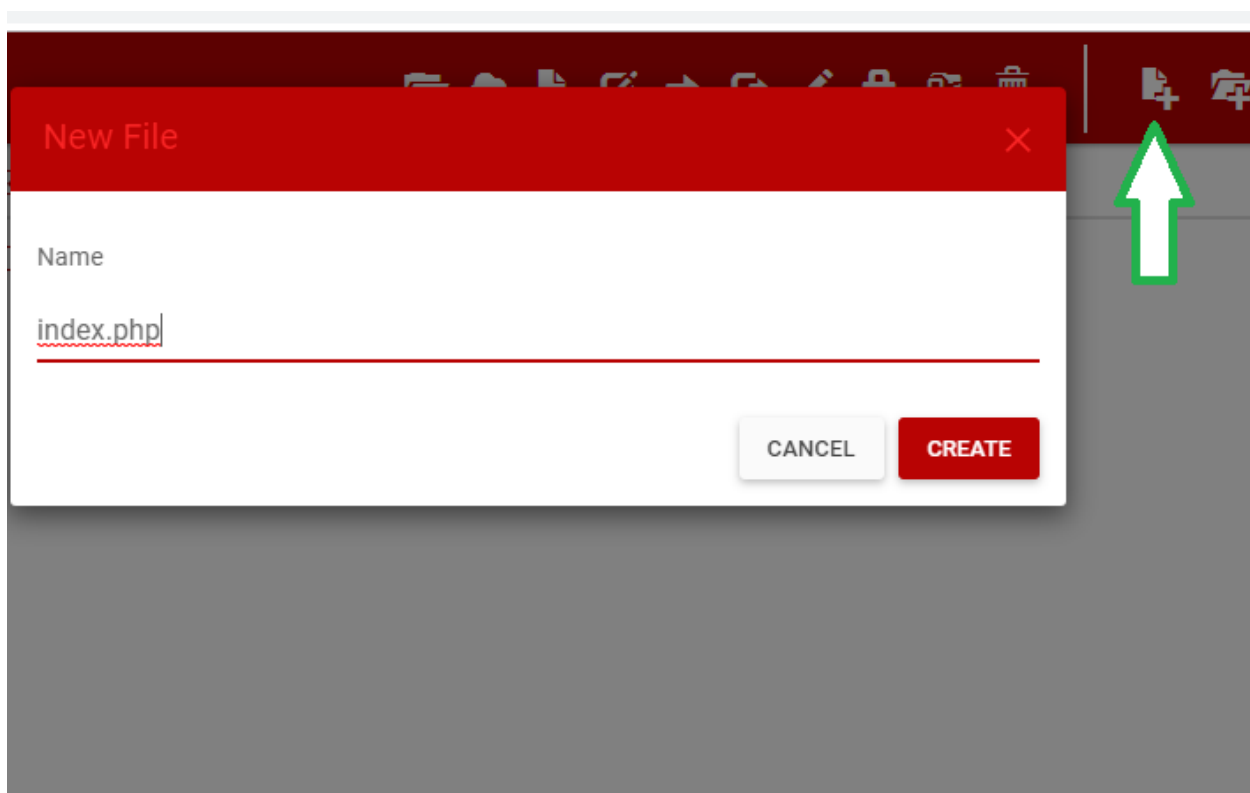
Có 4 tùy chọn để dựng web, do chúng ta sẽ code tay nên chọn vào cái Upload Website nhé

Click chuột phải và tạo 1 file mới có tên là index.php





OK. Đã vào được trang quản lí thư mục



Click New file và tạo file index.php nhé ! Nếu có bất đăng nhập thì các bạn điền tên web và pass admin lúc này đã tạo nhé

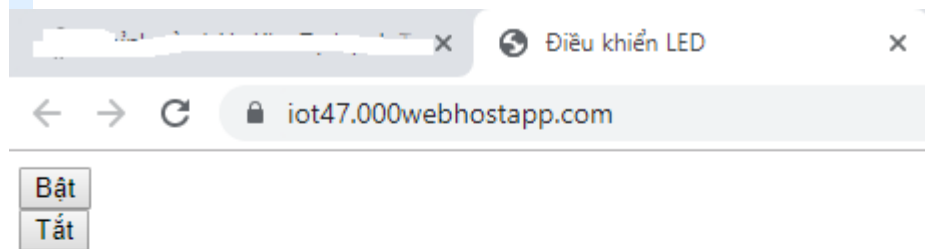
index.php chính là file đầu tiên mà trình duyệt sẽ chạy vào để load mã code khi người dùng vào web chúng ta. Nó cũng chính là file để chúng tạo code giao diện web, xử lý các sự kiện ấn nút của người dùng

Click chuột phải và Open file index.php lên. Copy đoạn mã HTML này vào và truy cập vào WEB thử nhé !



```
1 <?php
2 if (isset($_GET["bat"]))
3 {
4     $myfile = fopen("trangthaiLED.txt", "w");
5     fwrite($myfile,"bat");
6     fclose($myfile);
7     echo "Đã bật LED";
8 }
9 else if (isset($_GET["tat"]))
10 {
11     $myfile = fopen("trangthaiLED.txt", "w");
12     fwrite($myfile,"tat");
13     fclose($myfile);
14     echo "Đã tắt LED";
15 }
16?>
17
18<!DOCTYPE html>
19<html>
20<head>
21    <meta charset="UTF-8">
22    <title>Điều khiển LED</title>
23</head>
24<body>
25    <form method="GET" action="index.php"><input type="submit" name="bat" value="Bật"></form>
26    <form method="GET" action="index.php"><input type="submit" name="tat" value="Tắt"></form>
27
28</body>
```

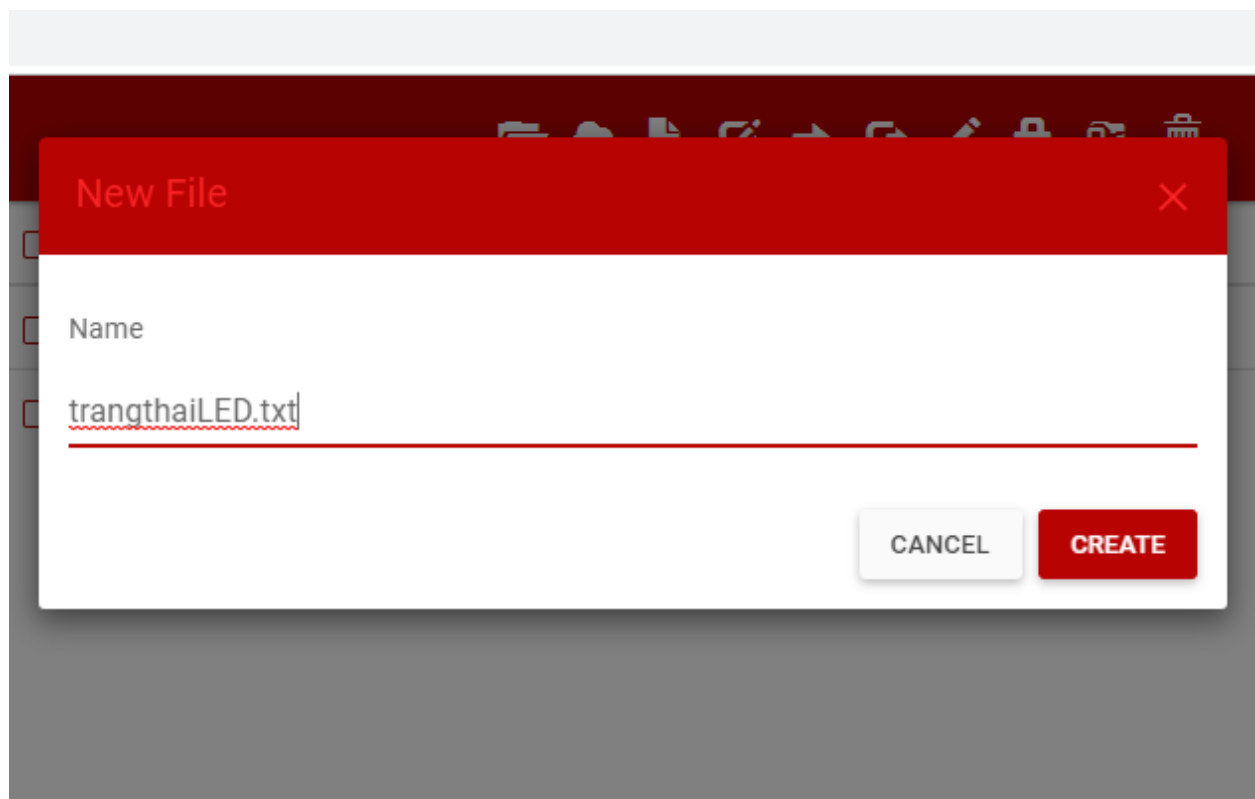
29</html>



thứ nào

Truy cập vào web của mình

Đã có giao diện điều khiển. Bây giờ mình sẽ tạo tiếp 1 file trangthaiLED.txt



OK. Bây nguyên tắc hoạt động của web chúng ta sẽ như sau:

Khi truy cập vào web sẽ không có dữ liệu nào được nạp nên màn hình chỉ hiển ra nội dung của mã html

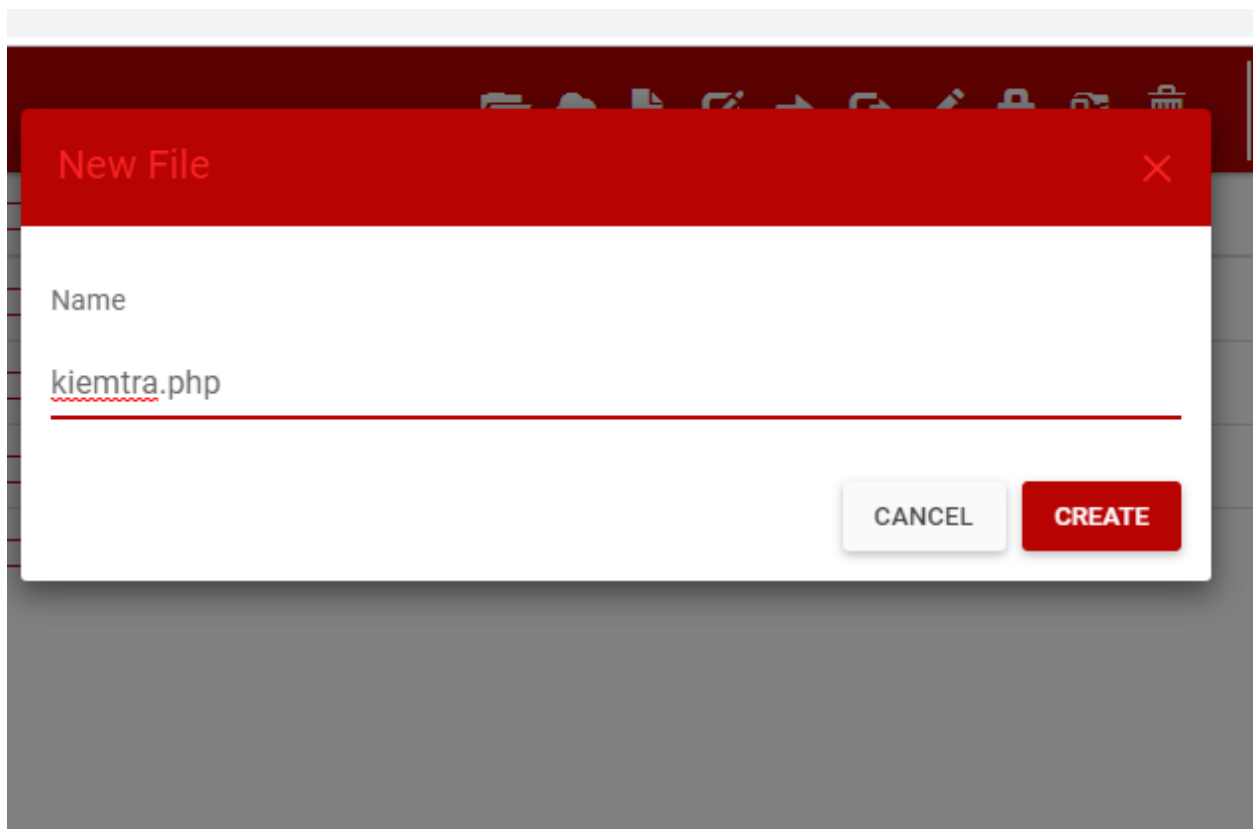
Khi ấn nút tắt trên WEB sẽ gọi tới file index.php và truyền thêm biến tắt để sửa nội dung file trangthaiLED.txt thành tắt

Ngược lại: Khi ấn nút Bật trên WEB sẽ gọi tới file index.php để sửa nội dung file txt thành bật

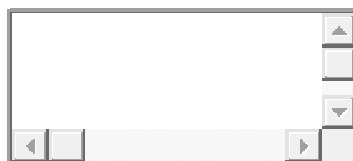
Giờ hãy thử ấn nút Bật và Tắt sau đó kiểm tra nội dung file trangthaiLED.txt xem có giống như mình nói không nhé. Chúng ta có thể thấy file trangthaiLED.txt giống như 1 con IC eeprom vậy

OK, bây giờ còn 1 nhiệm vụ nữa trước khi kết thúc phần thiết kế server. Đó là ta phải tạo ra 1 đường dẫn để truy cập vào và đọc thông tin có trong file text

Hãy tạo 1 file tên là kiểmtra.php



Nội dung của file kiểmtra.php như sau

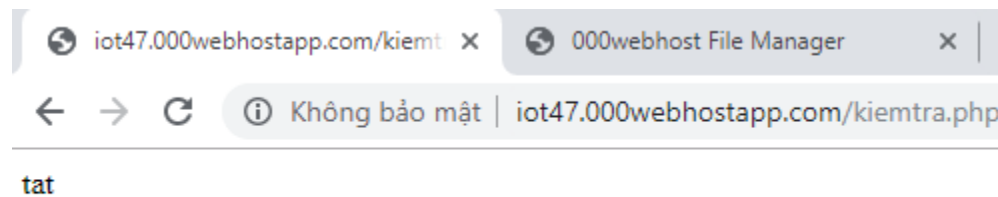


```
1<?php
```

```
2 $read = file('trangthaiLED.txt');
```

```
3 foreach ($read as $line)
4 {
5     echo $line;
6 }
7?>
```

OK. Giờ hãy gõ đường link có dạng **<http://iot47.000webhostapp.com/kiemtra.php>**  
Lưu ý: iot47 là tên miền của mình, bạn phải thay bằng tên miền của bạn nhé



Và đây là kết quả. Nội dung của file txt đã hiện ra màn hình. Hãy thử vào trang chính ấn nút Tắt và load lại trang kiểm tra xem nội dung có thay đổi không nhé !

Vậy là xong ! Bạn đã có 1 server mọi lúc mọi nơi và FREE cho riêng mình rồi đó !  
Giờ tới phần ESP cho nó truy cập vào địa chỉ **[iot47.000webhostapp.com/kiemtra.php](http://iot47.000webhostapp.com/kiemtra.php)** để lấy nội dung trạng thái bat tat về.

## ESP8266 kiểm tra trạng thái LED và bật tắt LED

### 1. Test thử bằng máy tính với Hecurles và module chuyển đổi USB-UART

Trước khi làm việc với vi điều khiển, mình sẽ lại test thử bằng cách gửi lệnh bằng tay trước nhé



Mình đã chuẩn bị module PL2303 và một pcb ra chân cho esp8266 để kết nối cho dễ. Các bạn cứ kết nối theo sơ đồ này là được.

ESP8266	PL2303
VCC + CH_PD	3.3V
GND	GND
TX	RX
RX	TX

Bây giờ cắm PL2303 vào máy tính, bật Hercules lên và gửi lệnh thử nào

Chú ý, sau mỗi truy vấn các bạn gửi CIPCLOSE để đóng kết nối thì k bị lỗi nhé như mình ở trên video nhé !  
 Khi gửi lệnh kết nối tới server thì phải chờ nó trả về CONNECT OK đã rồi mới send truy vấn đi !  
 Sau mỗi truy vấn, chuỗi reponse sẽ trả về các header(thứ mà ta không quan tâm lắm) và dưới cùng chính là nội dung của file txt ( **bat** or **tat**). Chúng ta sẽ bắt chuỗi này để đưa ra trạng thái điều khiển LED tương ứng !

## Demo giao tiếp với vi điều khiển PIC16F877A điều khiển LED

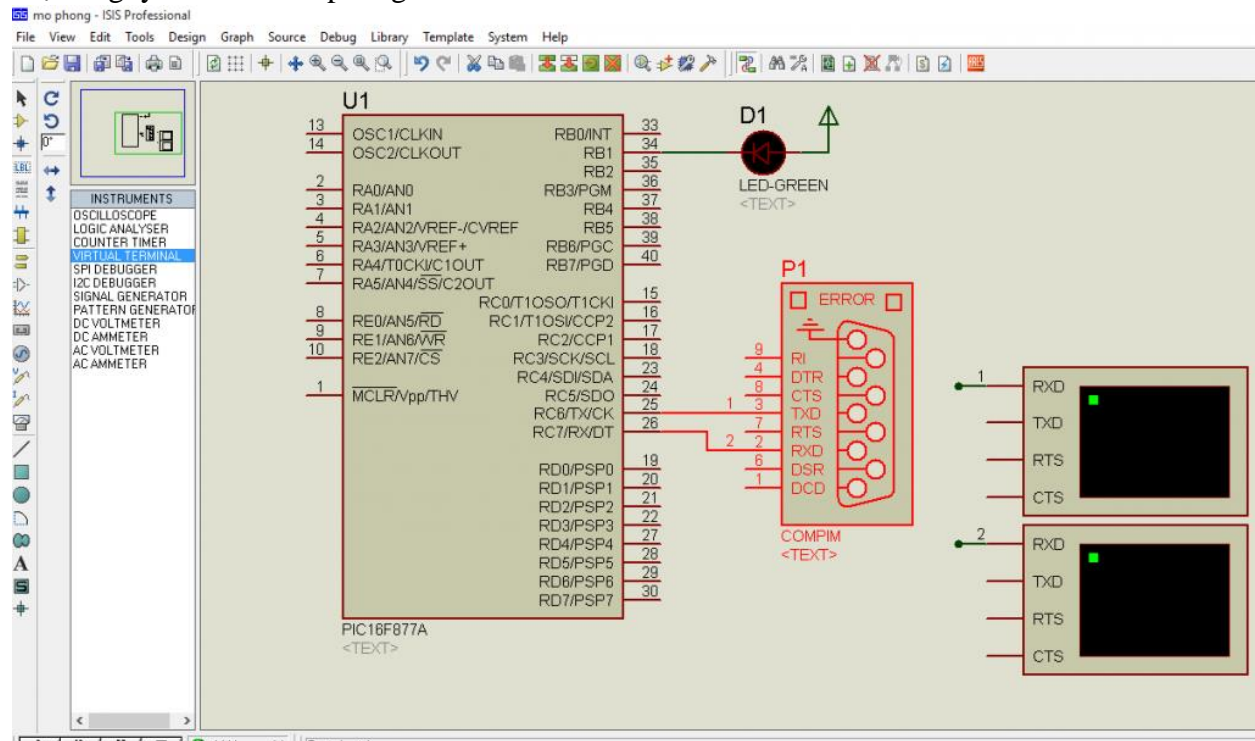
Chúng ta đã test gửi thử với lệnh AT bằng máy tính xong. Giờ mình sẽ giao tiếp bằng video khiển. Do hơi ngại làm mạch thật nên mình sẽ sử dụng phần mềm mô phỏng Proteus, đương nhiên trong proteus không có module wifi cho chúng ta mô phỏng nên con PIC ảo sẽ giao tiếp với ESP8266 thật của cổng COMPIM của proteus

**Các bạn chuẩn bị**

- Phần mềm mô phỏng mạch điện tử Proteus
- Phần mềm lập trình CCS PIC
- Module pl2303 và esp8266 (hàng thật :v)

Chú ý, proteus 7.3 mình dùng thì thấy nó không hỗ trợ tốc độ baud 115200 nên tốt nhất hãy chuyển tốc độ baud của ESP8266 về 9600 cho chắc chắn nhé. Sử dụng phần mềm hecurles và gửi lệnh **AT+UART\_DEF=9600,8,1,0,0,\$0D\$0A** để chuyển baudrate về 9600

## Mạch nguyên lí trên mô phỏng



Mạch mô phỏng đơn giản như thế thôi nhé, các bạn nhớ click chuột phải vào cổng COMPIM để cài tốc độ baud 115200 và chọn cổng COM của PL2303 (nếu không biết COM mấy thì vào Device Manager ở trong (chuột phải) My Computer nhé !

Mình có thêm 2 cái terminal để DEBUG dữ liệu gửi nhận UART

**Edit Component**

Component Reference:  Hidden: ☐

Component Value:  Hidden: ☐

VSM Model:

Physical port:

Physical Baud Rate:

Physical Data Bits:

Physical Parity:

Virtual Baud Rate:

Virtual Data Bits:

Virtual Parity:

Advanced Properties:

Physical Stop Bits:

Other Properties:

☐ Exclude from Simulation ☐ Attach hierarchy module

☐ Exclude from PCB Layout ☐ Hide common pins

☐ Edit all properties as text

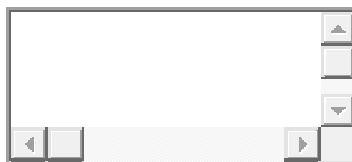
Click

phải vào COMPIM để cài baud

Chương trình điều khiển cho PIC trên phần mềm CCS qua tập lệnh at esp8266

Các bạn khởi tạo project với thạch anh 20Mhz

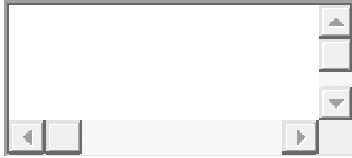
Cấu hình UART như sau



```
#use rs232(uart1, baud=9600,ERRORS)
```

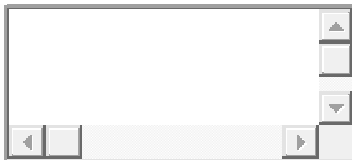
Cấu hình các thông số của bạn nhé





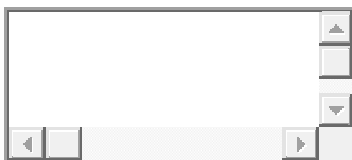
```
1 char *SSID="Quoc Khanh";           //your wifi
2 char *PASS="daoxuankhanh";         //your wifi
3 char *WEBSITE="iot47.000webhostapp.com"; //your web
```

Hàm khởi tạo ESP8266



```
1 void ESP8266_init()
2 {
3   printf("AT\r\n");delay_ms(500);
4   printf("ATE0\r\n");delay_ms(500);
5   printf("AT+CWMODE=1\r\n");delay_ms(500);
6   printf("AT+CWJAP=\"%s\",\"%s\"\r\n",SSID,PASS);delay_ms(4000); // thay mat khau va tai khoan tuong ung
7   printf("AT+CIPMUX=0\r\n");delay_ms(500);
8 }
```

Hàm bắt chuỗi **bat** hoặc **tat** để điều khiển LED tương ứng



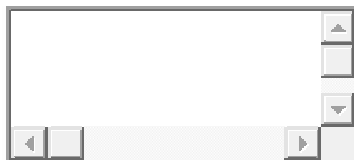
```
1 #int_RDA
2 void RDA_isr(void)//Ngat du lieu khi truyen nhan
3 {
4   unsigned char c;
5   c=getc();//Gan ki tu vua nhan duoc vao bien tam
6   //-----bat ki tu bat-----//
7   if(c=='b')
```

```

8  {
9      c=getc();
10     if(c=='a')
11     {
12         c=getc();
13         if(c=='t')output_low(PIN_B1); //ON
14     }
15 }
16 else if(c=='t')
17 {
18     c=getc();
19     if(c=='a')
20     {
21         c=getc();
22         if(c=='t')output_high(PIN_B1); //OFF
23     }
24 }
25 //-----bat ki tu tat-----//
26}

```

Tiếp theo xây dựng hàm để gửi request liên tục lên server của bạn và reponse có kèm trạng thái LED về. Mình sẽ đặt hàm này vào trong **while(1)** của main



```

1 void GET_StatusLED()
2 {
3     int length;
4     char data[90];
5     sprintf(data,"GET /kiemtra.php HTTP/1.1\r\nHost: iot47.000webhostapp.com\r\n\r\n");
6     printf("AT+CIPSTART=\"%TCP\\\", \"%s\\\",80\r\n",WEBSITE);delay_ms(2000); //wait connect
7     length=strlen(data);
8     printf("AT+CIPSEND=%i\r\n",length);delay_ms(500);

```

```
9 printf(data);delay_ms(1000);  
10 printf("AT+CIPCLOSE\r\n");delay_ms(200);  
11 }
```

Và TEST nào !

## Kết luận

Với phương pháp này và sử dụng tập lệnh at esp8266, chúng ta có ưu điểm là thực hiện nhanh chóng, dễ dàng, điều khiển mọi lúc mọi nơi miễn là có internet. Tuy nhiên, tốc độ đáp ứng rất chậm do phải thiết lập các kết nối liên tục, ESP8266 sẽ nhanh nóng, cũng như hao tổn tài nguyên WEB của bạn nếu như có quá nhiều esp8266 truy vấn vào.

Các bạn có thời gian có thể làm mạch thực để test nhé. Nhưng chỉ demo hay làm đồ án thôi chứ không nên xài vì cách này không tối ưu lắm và cái server FREE chả biết khi nào nó ngỏm . Hehe

Ở các bài tiếp theo mình sẽ hướng dẫn các bạn biến ESP8266 thành 1 cái server WEB thay vì phải đi thuê web hay xài web free như trên !

## DOWNLOAD

Toàn bộ file code, mô phỏng các bạn có thể tải về **TẠI ĐÂY**

Đọc thêm các bài về tập lệnh at esp8266