

DATA COMMUNICATION

LAB 5: NODE MCU - CLIENT MODE

I. Introduction to Working Mode in ESP8266 Series

The ESP8266 or almost every Wifi modules usually have three working modes: Station, AP (Access Point) and Dual mode. The Station mode is similar to the computers, cell phones or tablets while the AP mode is playing a role of the routers in our network. In a combination, the dual mode allows the Wifi module to work in both Station and AP mode. The ESP8266, ESP8285 and ESP32 module fully support three modes mentioned above.

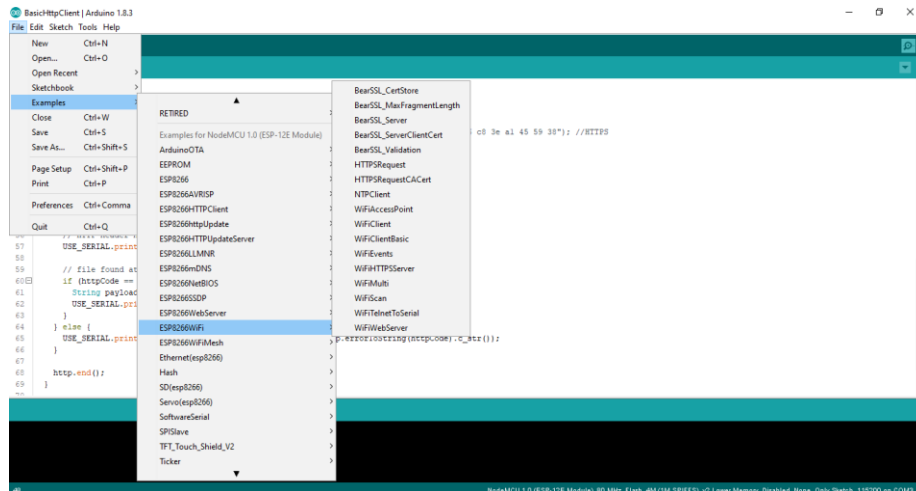


Figure 1: ESP8266 WiFi examples

There are many examples located in **File, Examples**. As presented in Figure 1, many free source codes concerning ESP8266 Wifi module are supported in Arduino IDE.

II. Getting Started with Examples

HTTP GET Request

In the first step, go to menu **File, Examples, ESP8266HTTPClient**, and open the project **BasicHttpClient** (see Figure 2), replace the string `"http://192.168.1.12/test.html"` with the new one `"http://jsonplaceholder.typicode.com/users"` (see Figure 3) and then, upload the program to the NodeMCU board. In fact, the old URL (<http://192.168.1.12/test.html>) is a local link, which needs to be implemented in a local PC to act like a server. However, with a new URL (<http://jsonplaceholder.typicode.com/users>), we are sending a request to an Internet server. You can copy this URL, paste in the browser (IE, Chrome or Firefox) and compare the response between your MCUNode and the browser.

In the second step, change the URL to `"http://jsonplaceholder.typicode.com/users/1"`, upload the program and then, check the results. The first two examples are some basic steps to use HTTP GET request.

For your more understanding about GET request, some headers can be added in the request. To do that, after the initialization of HTTP GET (the `http.begin("your url here")`), following statement can be used:

```
http.begin("your url here");  
  
http.addHeader("Content-Type", "text/plain");
```

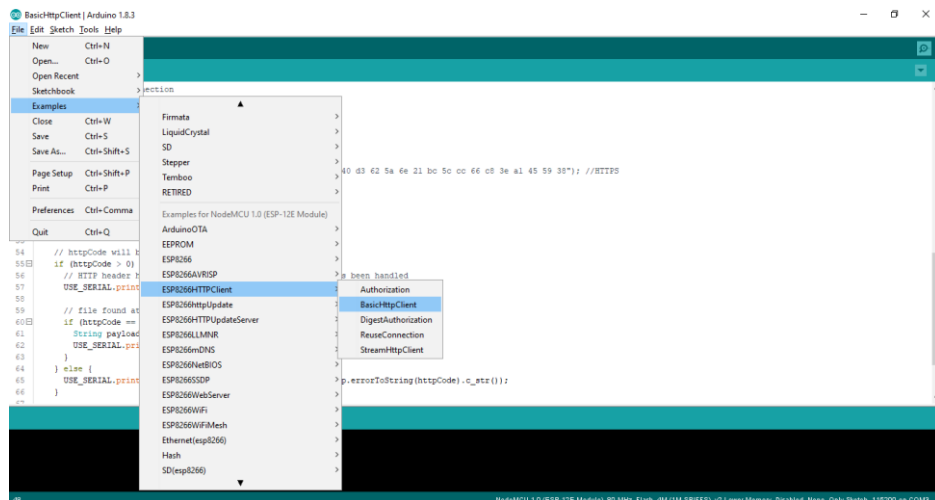


Figure 2: BasicHttpClient example

```
USE_SERIAL.print("[HTTP] begin...\n");
// configure traged server and url
//http.begin("https://192.168.1.12/test.html", "7a 9c f4 db 40 d3 62 5a 6e 21 bc 5c cc 66 c8 3e a1 45 59 38"); //HTTPS
http.begin("http://jsonplaceholder.typicode.com/users"); //HTTP

USE_SERIAL.print("[HTTP] GET...\n");
// start connection and send HTTP header
int httpCode = http.GET();
```

Figure 3 Replace the old URL by a new one

HTTP POST Request

POST request has some similarities to GET requests and is also supported in Arduino Library (There are many documents about the differences between POST and GET, you can find it yourself on the internet). This time, you can create your own POST request/response by using this online server: <http://ptsv2.com/> and this tool for testing POST/GET request: <https://apitester.com/> it will takes you some time (may be 10 to 15 minutes to get used to with them)

Once you have created and tested your server, do some modification for the BasicHttpClient example like the figure below (remember to read the code comment):

```
// configure traged server and url
//http.begin("https://192.168.1.12/test.html", "7a 9c f4 db 40 d3 62 5a 6e 21 bc 5c cc 66 c8 3e a1 45 59 38"); //HTTPS
http.begin("http://ptsv2.com/t/test12345/post"); //HTTP URL, replace this with the one you created
http.addHeader("Content-Length", "0"); //Specify content-type header
http.addHeader("Content-Type", "application/x-www-form-urlencoded"); //Specify content-type header

USE_SERIAL.print("[HTTP] POST...\n");
// start connection and send HTTP header
int httpCode = http.POST("");

// httpCode will be negative on error
if (httpCode > 0) {
    // HTTP header has been send and Server response header has been handled
    USE_SERIAL.printf("[HTTP] POST... code: %d\n", httpCode);
```

How you should change your code

Then you can upload your code and test it.

III. Exercise

1. Improve the programs in the manual that allows the user to enter WiFi SSID and password then connect to that WiFi network. In your examples, these data are hard-coded in the program. Please use the Serial Monitor on Arduino IDE to provide SSID and password.

- Khởi tạo 2 mảng ssid[50] và pass[50] để lưu kí tự từ Serial

```
1. char ssid[50];  
2. char pass[50];
```

- Sử dụng hàm Serial.ReadBytesUntil() để đọc các kí tự từ Serial. Số 10 truyền vào là kí tự “/n” trong char. Khi nhập từ Serial chú ý chọn chế độ NEWLINE.

```
1. while (Serial.available() == 0) {  
2.     // wait  
3. }  
4. Serial.readBytesUntil(10, ssid, 50);  
5.  
6.  
7. while (Serial.available() == 0) {  
8.     // wait  
9. }  
10. Serial.readBytesUntil(10, pass, 50);
```

- Thiết lập kết nối Wifi và thông báo “Connected”, nếu chưa thì hiện thị “...”

```
1. WiFi.begin(ssid, pass);  
2. while (WiFi.status() != WL_CONNECTED) {  
3.     delay(200);  
4.     Serial.print(".");  
5. }  
6. Serial.println("");  
7. Serial.println("WiFi connected");  
8. Serial.println("IP address: ");  
9. Serial.println(WiFi.localIP());
```

- Kết quả:

```
Enter your WiFi credentials.  
  
SSID: P436  
PASS: 436.436.436.123456  
.  
WiFi connected  
IP address:  
192.168.100.6
```

2. Continue improving your program by uploading value(s) to ThingSpeak server at <https://thingspeak.com/>. You can upload a counter value (increased by one every upload), or even some sensor values (e.g. DTH11). There are many manuals available on Internet that you can search. In ThingSpeak a POST request is required to upload your data. An example for

your reference can be found at <https://roboindia.com/tutorials/nodeMCU-dht11-thingspeak-data-upload>,

- Thiết lập chế độ Wifi Station, Kết nối Wifi

```
1. WiFi.mode(WIFI_STA);           //This line hides the viewing of ESP as wifi hotspot
2. WiFi.begin(ssid, password);    //Connect to your WiFi router
3.
4. Serial.print("Connecting");
5.
6. while (WiFi.status() != WL_CONNECTED) {
7.     delay(500);
8.     Serial.print(".");
9. }
10.
11. Serial.print("Connected to ");
12. Serial.println(ssid);
13. Serial.print("IP address: ");
14. Serial.println(WiFi.localIP());
```

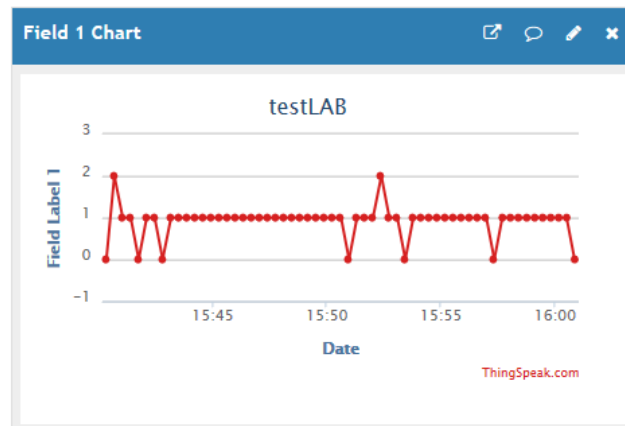
- Kết nối với host ThingSpeak

```
1. WiFiClient client;
2. const char *host = "api.thingspeak.com";
3. const int httpPort = 80;
4. //-----
5.
6. if(!client.connect(host, httpPort)){
7.     Serial.println("Connection Failed");
8.     delay(300);
9.     return;
10. }
```

- Tạo GET request với host theo format của HTTP

```
1. String ADCData;
2. int adcvalue=analogRead(A0);
3. ADCData = String(adcvalue);
4. String Link="GET /update?api_key="+apiKey+"&field1=";
5. Link = Link + ADCData;
6. Link = Link + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection: close\r\n\r\n";
7. client.print(Link);
8. delay(100);
```

- Để xem kết quả của data đọc được, truy cập <https://thingspeak.com/> và vào phần PRIVATE VIEW để xem dạng biểu đồ của data.



IV. Extra Exercise

1. Integrate a small program that allows the user to scan for WiFi networks and list them with an index at the beginning. The user selects a Wifi network by providing an index (e.g. 1, 2 or 3) and then, enter a password. The Network signal strength is expected to be displayed also. The results will be something like Figure 4. Your program should wait for an index from user and then, ask for a password in order to connect to a network.



Figure 4: Network wifi scanning

A manual for your reference can be found at <https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiScan/WiFiScan.ino>. However, you can search on Google and please add your reference in the report.

- Thiết lập chế độ Wifi Station, tắt kết nối với wifi lần trước, thiết lập Serial 115200.

```

1. Serial.begin(115200);
2. WiFi.mode(WIFI_STA);
3. WiFi.disconnect();
4. delay(100);
5. Serial.println("Setup done");
  
```

- Tiến hành quét các nguồn Wifi và hiển thị ra Serial

```

1. int n = WiFi.scanNetworks();
2.   Serial.println("scan done");
3.   if (n == 0) {
4.     Serial.println("no networks found");
5.   } else {
6.     Serial.print(n);
7.     Serial.println(" networks found");
8.     for (int i = 0; i < n; ++i) {
9.       Serial.print(i + 1);
10.      Serial.print(": ");
11.      Serial.print(WiFi.SSID(i));
12.      Serial.print(" (");
13.      Serial.print(WiFi.RSSI(i));
14.      Serial.print(")");
15.      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " : "*");
16.      delay(10);
17.    }
18.  }

```

- Yêu cầu người dùng nhập vào số thứ tự của Wifi muốn chọn và in kết quả ra Serial

```

1. char choice[10];
2. Serial.print("Enter your choice: ");
3. while (Serial.available() == 0) {
4.   // wait
5. }
6.
7. Serial.readBytesUntil(10, choice, 10);
8. Serial.println(choice);

```

- Hiển thị ra tên SSID của Wifi mà người dùng lựa chọn

```

1. for(int i=0;i<n;++i){
2.   if(myChoice==i){
3.     Serial.println("Your choice: " + WiFi.SSID(i-1));
4.     int myLength=WiFi.SSID(i-1).length();
5.     WiFi.SSID(i-1).toCharArray(ssid,myLength+1);
6.   }
7. }

```

- Yêu cầu nhập Password và kết nối

```

1. Serial.println("Enter the password: ");
2. while (Serial.available() == 0) {
3.   // wait
4. }
5. Serial.readBytesUntil(10, pass, 50);
6.
7. WiFi.begin(ssid, pass);

```

- MQTT is widely used in IoT, this is a kind of protocol. Check the instruction from <https://arduino.esp8266.vn/network/mqtt.html> then write a program that uploads the sensor values to <https://www.cloudmqtt.com/> (this is the MQTT broker). There are also 3 LEDs for the user to control through the internet.

- Kết nối với Wifi

```

1. void setup_wifi() {
2.   Serial.print("Connecting to ");
3.   Serial.println(ssid);
4.   WiFi.begin(ssid, password);
5.   while (WiFi.status() != WL_CONNECTED) {
6.     delay(500);
7.     Serial.print(".");
8.   }
9.   Serial.println("WiFi connected");
10.  Serial.println("IP address: ");
11.  Serial.println(WiFi.localIP());
12. }

```

- Kết nối với MQTT Cloud. Truy cập <https://www.cloudmqtt.com> để tạo một Instance mới

Server	m15.cloudmqtt.com	
User	tnsotidt	<button>Restart</button>
Password	U2maRry45asS	<button>Rotate</button>
Port	14286	
SSL Port	24286	
Websockets Port (TLS only)	34286	
Connection limit	5	

Sau đó thiết lập theo các thông tin được cung cấp để kết nối với MQTT Cloud

```

1. WiFiClient espClient;
2. PubSubClient client(espClient);
3. client.setServer(mqtt_server, mqtt_port);

```

- Hiện thực hàm callback() để nhận dữ liệu gửi về NodeMCU, với topic là “demo”, nếu nhận được “1” có nghĩa là bật LED, ngược lại đối với “0”.

```

1. void callback(char* topic, byte* payload, unsigned int length) {
2.   Serial.print("Message arrived [");
3.   Serial.print(topic);
4.   Serial.print("] ");
5.   for (int i = 0; i < length; i++) {
6.     char receivedChar = (char)payload[i];
7.     Serial.print(receivedChar);
8.     if (receivedChar == '1')
9.       // Kiểm tra nếu tin nhận được là 1 thì bật LED và ngược lại
10.      digitalWrite(ledPin, LOW); //Node MCU LOW = Turn On
11.     if (receivedChar == '0')
12.      digitalWrite(ledPin, HIGH); //Node MCU LOW = Turn Off
13.   }
14.   Serial.println();
15. }

```

- Hiện thực hàm reconnect() để kết nối lại với MQTT Cloud mỗi khi mất kết nối


```

1. void reconnect() {
2.     // Chờ tới khi kết nối
3.     while (!client.connected()) {
4.         Serial.print("Attempting MQTT connection...");
5.         // Thực hiện kết nối với mqtt user và pass
6.         if (client.connect("ESP8266Client",mqtt_user, mqtt_pwd)) {
7.             Serial.println("connected");
8.             // Khi kết nối sẽ publish thông báo
9.             client.publish(mqtt_topic_pub, "ESP_reconnected");
10.            // ... và nhận lại thông tin này
11.            client.subscribe(mqtt_topic_sub);
12.        } else {
13.            Serial.print("failed ");
14.            Serial.println(" try again in 5 seconds");
15.            // Đợi 5s
16.            delay(5000);
17.        }
18.    }

```

- Kết quả truyền dữ liệu về Node MCU từ server MQTT Cloud

Websocket

Send message

Topic

Message

Send

Received messages

Topic	Message
demo	0
demo	1