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 tables 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.

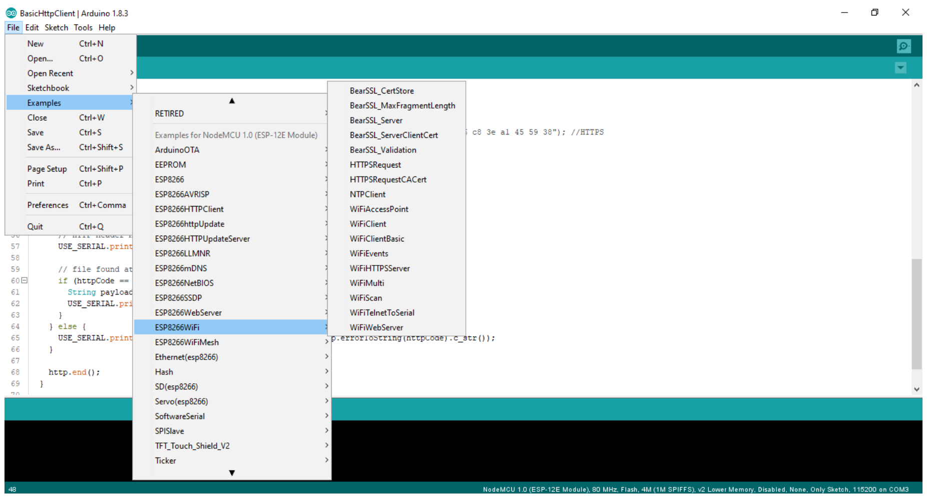


Figure 1: ESP8266 WiFi examples

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");
```

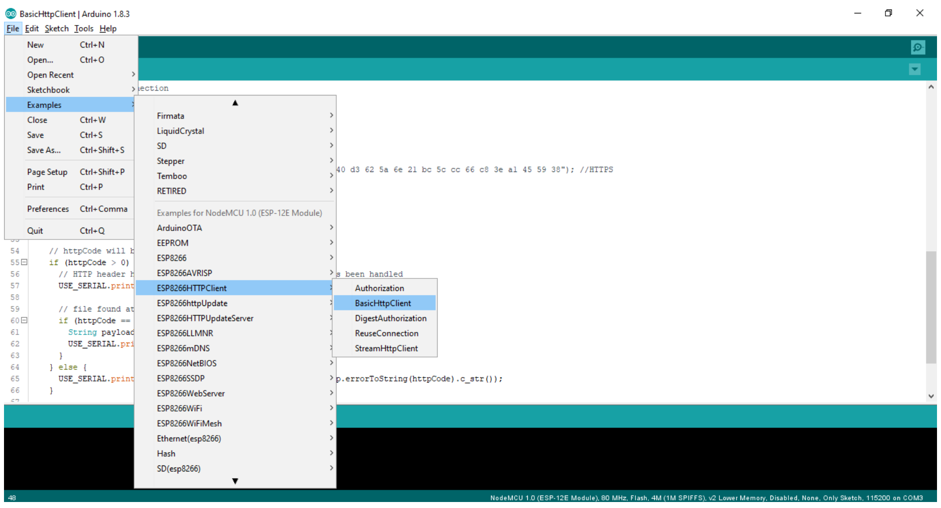
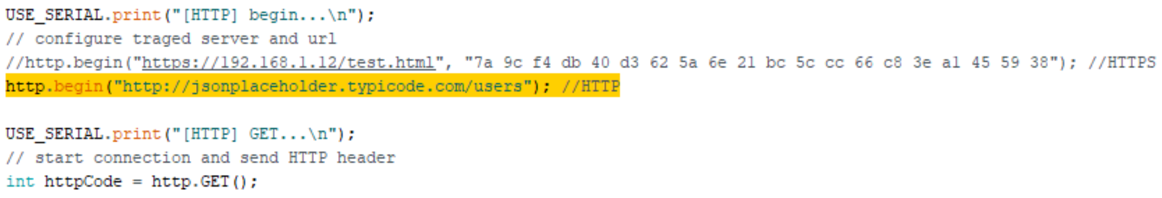


Figure 2: BasicHttpClient example



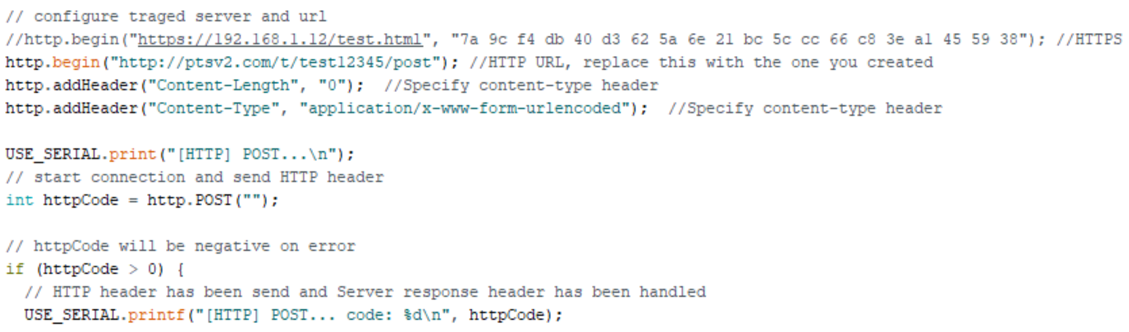
```
USE_SERIAL.print("[HTTP] begin...\n");
// configure traged server and url
//http.begin("https://192.168.1.12/test.html", "7a 9c f4 db 40 d3 62 5a 6e 21 bc 5c cc 66 c8 3e a1 45 59 38"); //HTTPS
http.begin("http://jsonplaceholder.typicode.com/users"); //HTTP

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.

```
case READCHAR:{
      if (Serial.available() > 0){
        c = Serial.read();

        if (c == '#'){
          state = SSIDPASS;
          read_level++;
        }
        else {
          s += c;
        }
      }
      break;
    }

    case SSIDPASS:{
      switch(read_level){
        case 0:{
          state = READCHAR; s = "";

          Serial.print("Enter SSID: ");
          break;
        }
        case 1:{
          Serial.println(s);
          s.toCharArray(ssid, s.length());
          ssid[s.length()] = 0;

          state = READCHAR; s = "";
          Serial.print("Enter Password: ");
          break;
        }
        case 2:{
          Serial.println(s);
          s.toCharArray(password, s.length());
          password[s.length()] = 0;

          state = HTMLPOST;
          WiFi.mode(WIFI_STA);
          WiFiMulti.addAP(ssid, password);
          break;
        }
      }
      break;
    }
```

2. Continue improving your program by uploading value(s) to ThingSpeak server at [https://thingspeak.com/.](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,

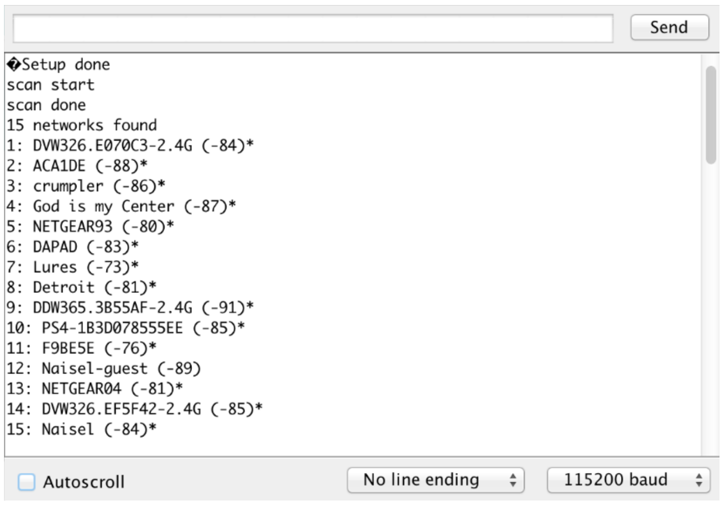
```
    case THINGSPEAK:{

            float h = (float)random(30, 90);
            float t = (float)random(20, 35);

            if (isnan(h) || isnan(t)){
             Serial.println("Failed to read from DHT sensor!");
             return;
            }

             if (client.connect(server,80)){  //  "184.106.153.149" or
api.thingspeak.com
            String postStr = apiKey;
            postStr +="&field1=";
            postStr += String(t);
```

```
                        postStr +="&field2=";
                        postStr += String(h);
                        postStr += "\r\n\r\n";

                        client.print("POST /update HTTP/1.1\n");
                        client.print("Host: api.thingspeak.com\n");
                        client.print("Connection: close\n");
                        client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
                                client.print("Content-Type:  application/x-www-form-
urlencoded\n");
                        client.print("Content-Length: ");
                        client.print(postStr.length());
                        client.print("\n\n");
                        client.print(postStr);

                        Serial.print("Temperature: ");
                        Serial.print(t);
                        Serial.print(" degrees Celcius, Humidity: ");
                        Serial.print(h);
                        Serial.println("%. Send to Thingspeak.");
                          }
                          client.stop();

                          Serial.println("Waiting...");

                         // thingspeak needs minimum 15 sec delay between updates,
i've set it to 30 seconds
                        delay(10000);
                        break;
                      }
```

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.



```
�Setup done
scan start
scan done
15 networks found
1: DVW326.E070C3-2.4G (-84)*
2: ACA1DE (-88)*
3: crumpler (-86)*
4: God is my Center (-87)*
5: NETGEAR93 (-80)*
6: DAPAD (-83)*
7: Lures (-73)*
8: Detroit (-81)*
9: DDW365.3B55AF-2.4G (-91)*
10: PS4-1B3D078555EE (-85)*
11: F9BE5E (-76)*
12: Naisel-guest (-89)
13: NETGEAR04 (-81)*
14: DVW326.EF5F42-2.4G (-85)*
15: Naisel (-84)*
```

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.

```
        case SCAN:{
              Serial.println("scan start");

              // WiFi.scanNetworks will return the number of networks found
              n = WiFi.scanNetworks();
              Serial.println("scan done");
              if (n == 0) {
                Serial.println("no networks found");
              } else {
                Serial.print(n);
                Serial.println(" networks found");
                for (int i = 0; i < n; ++i) {
                  // Print SSID and RSSI for each network found
                  Serial.print(i + 1);
                  Serial.print(": ");
                  Serial.print(WiFi.SSID(i));
                  Serial.print(" (");
                  Serial.print(WiFi.RSSI(i));
                  Serial.print(")");
                   Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ?
" " : "*");

                  delay(10);
                }
              }
              Serial.println("");

              state = SSIDPASS;
              break;
            }
```

2. 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.