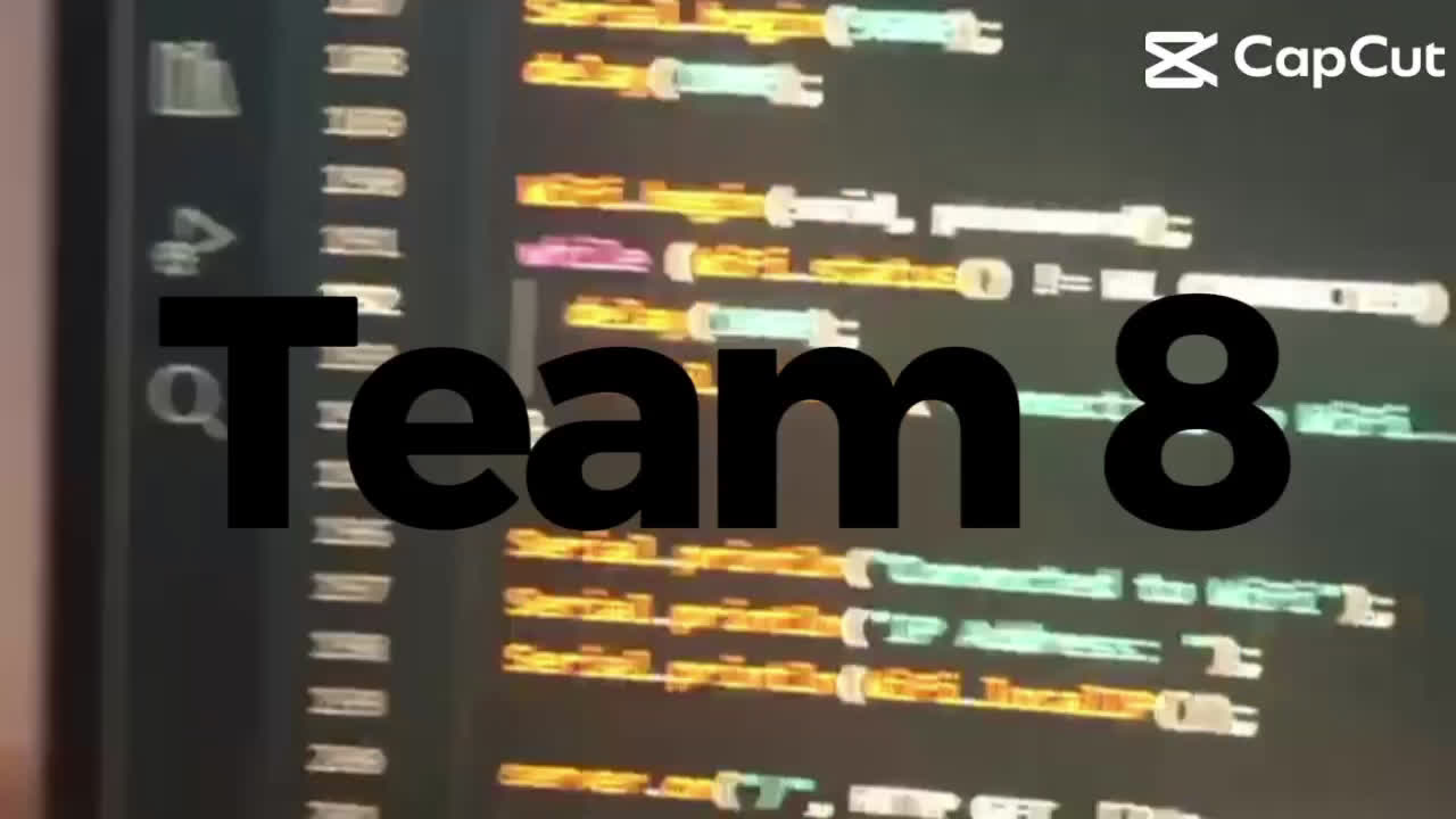

Team 8
CapCut

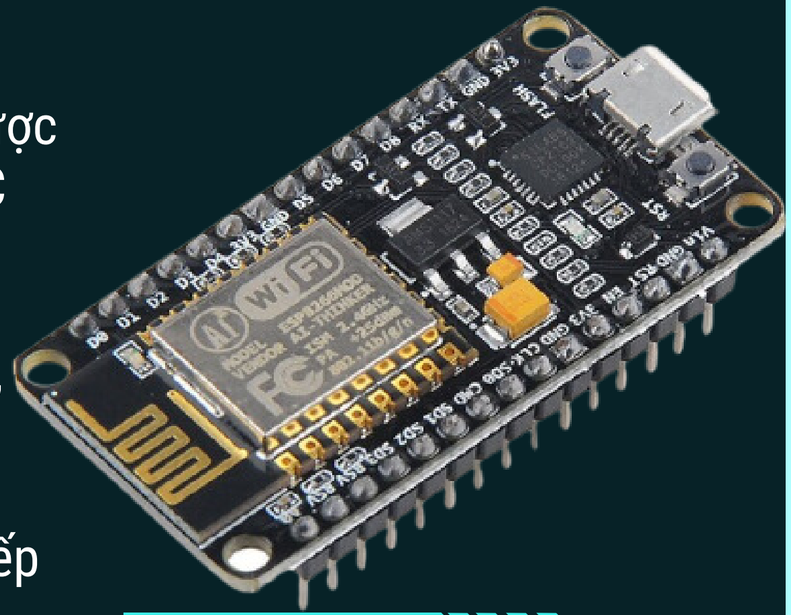
Member task

	Nội dung	**Slide**	Code
Trần Văn Thanh Nhật	x	x	x
Phạm Nguyễn Đạt		x	
Nguyễn Đình Tiến Đạt	x	x	
Nguyễn Quang Thạc	x	x	x
Phạm Minh Chiến	x	x	

I. GIỚI THIỆU

ESP8266 NodeMCU

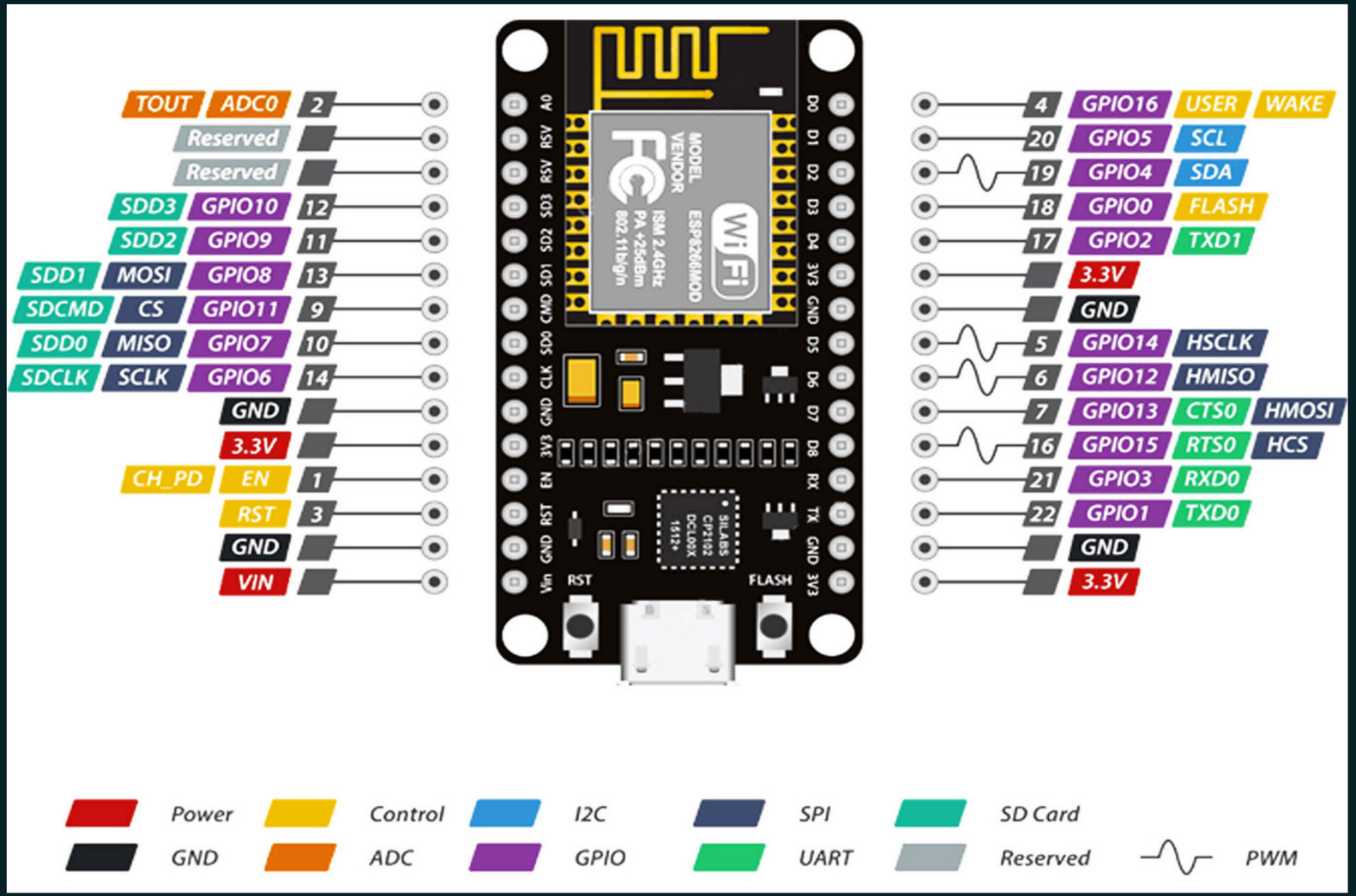
- ESP8266 NodeMCU là một nền tảng IoT mã nguồn mở, được xây dựng trên ESP8266. Đây là một vi điều khiển Wi-Fi SoC (System on a Chip) được sản xuất bởi Espressif Systems.
- NodeMCU cung cấp một bộ SDK để lập trình cho ESP8266 bằng ngôn ngữ Lua hoặc C++. Với các tính năng như Wi-Fi, GPIO, ADC, I2C, SPI, PWM và một số tính năng khác, NodeMCU ESP 8266 được sử dụng rộng rãi trong các ứng dụng IoT như kiểm soát thiết bị, thu thập dữ liệu và giao tiếp với các thiết bị khác.



THÔNG SỐ KĨ THUẬT

- Microcontroller : ESP8266EX
- Điện áp hoạt động: 3.3V DC
- Số chân I/O: 17 chân GPIO
- Kết nối mạng: WiFi 802.11 b/g/n
- Giao diện mạng: TCP/IP
- Đồng hồ thời gian thực (RTC): không tích hợp
- Bộ nhớ trong: 4MB
- RAM: 80KB
- Cổng nạp: Micro-USB
- Hỗ trợ các giao thức: MQTT, CoAP, HTTP/HTTPS
- Kích thước: 49 x 24.5 x 13mm

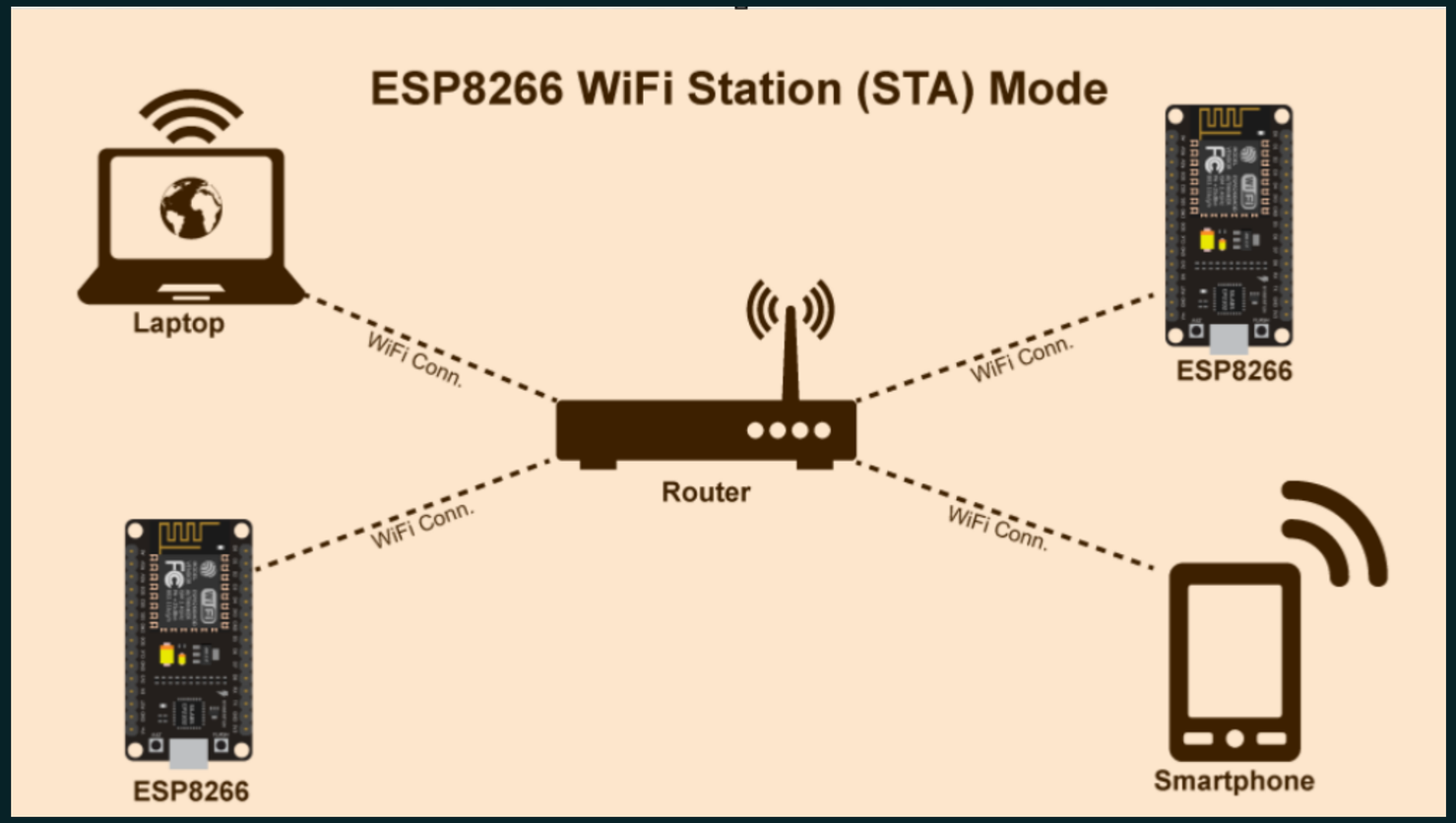
II. SƠ ĐỒ CHÂN



- 17 chân GPIO
- 1 kênh ADC
- 2 giao tiếp UART
- 4 đầu ra PWM
- 2 giao tiếp SPI và 1 giao tiếp I2C
- Nguồn 3.3V và GND

III. NGUYÊN LÍ HOẠT ĐỘNG

- **Station (STA) Mode**

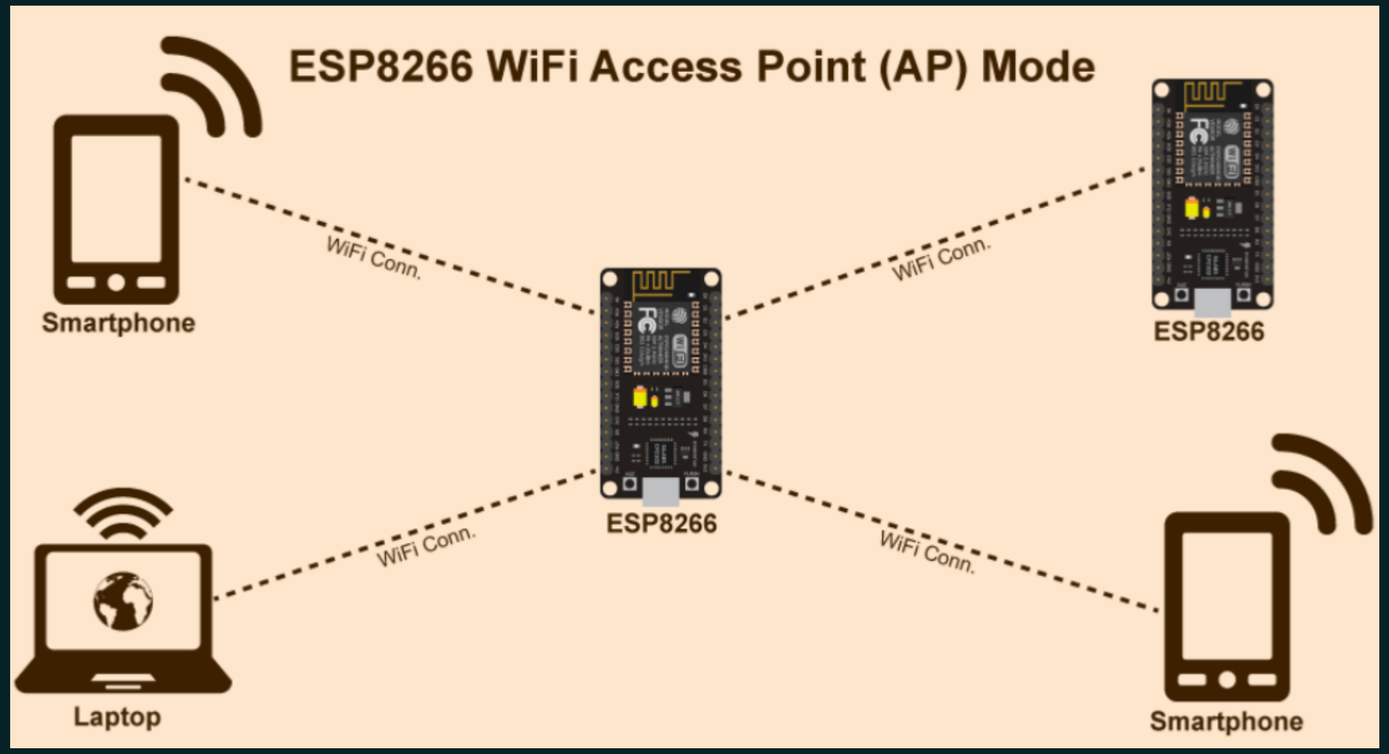


Ở chế độ trạm, ESP8266 sẽ hoạt động giống như điện thoại thông minh hoặc máy tính xách tay của bạn. Nó sẽ kết nối với kênh WiFi hiện có hoặc trong hầu hết các trường hợp, WiFi được bộ định tuyến của bạn quảng cáo.

Khi bạn đã lập trình ESP8266 ở chế độ STA và kết nối thành công với WiFi ổn định, bạn có thể truy cập bất kỳ trang web nào trên internet.

III. NGUYÊN LÍ HOẠT ĐỘNG

- **Access Point (AP) Mode**

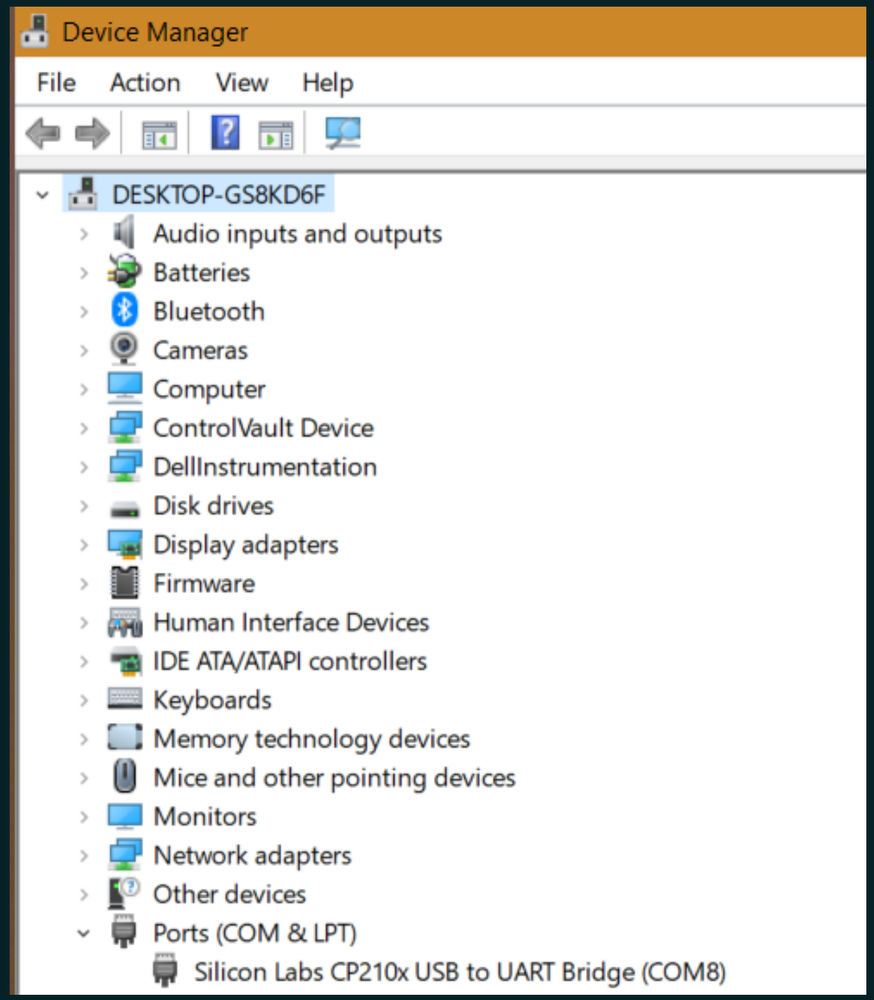


Ở chế độ này, ESP8266 sẽ được coi như một điểm phát sóng WiFi của mình bằng SSID và Mật khẩu tùy chỉnh. Các thiết bị thông minh khác sẽ có thể kết nối và do đó thiết lập liên lạc với mô-đun WiFi ESP8266.

Trong trường hợp này, ESP8266 sẽ không thể truy cập Internet vì nó chỉ quảng cáo một điểm phát sóng WiFi. Ngoài ra, các thiết bị kết nối với WiFi do ESP8266 quảng cáo sẽ chỉ có thể giao tiếp với ESP8266 chứ không thể giao tiếp với bất kỳ máy chủ web nào.

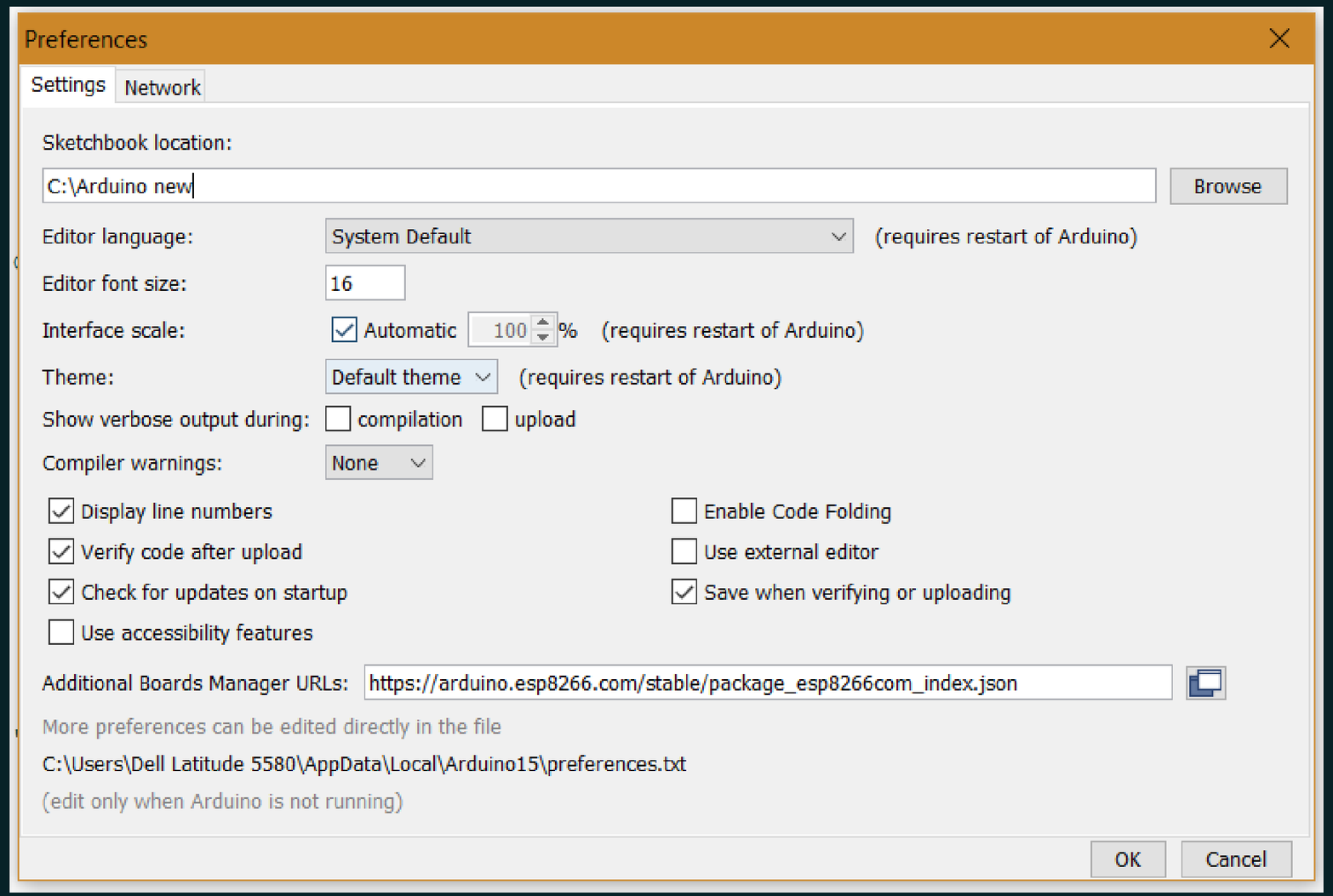
CÀI ĐẶT DRIVER CHO MCU ESP8266

Truy cập :
https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
Tải “CP210x_Windows_Drivers” và cài đặt như bình thường

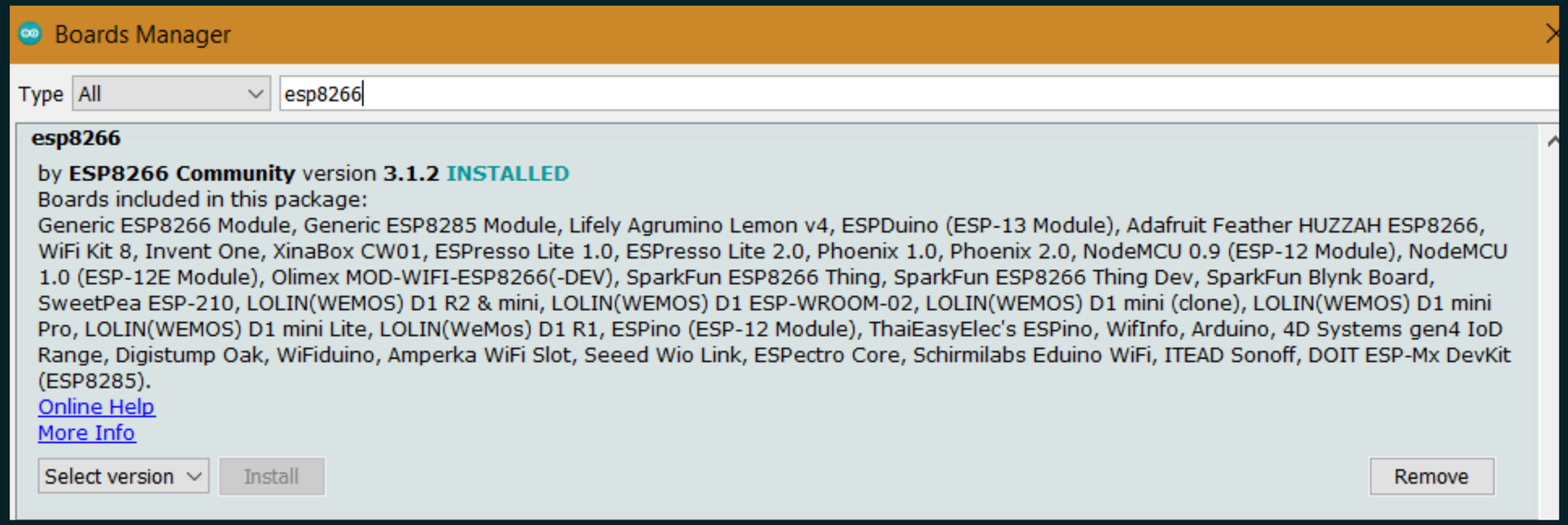


THIẾT LẬP THƯ VIỆN CHO NODE MCU ESP8266

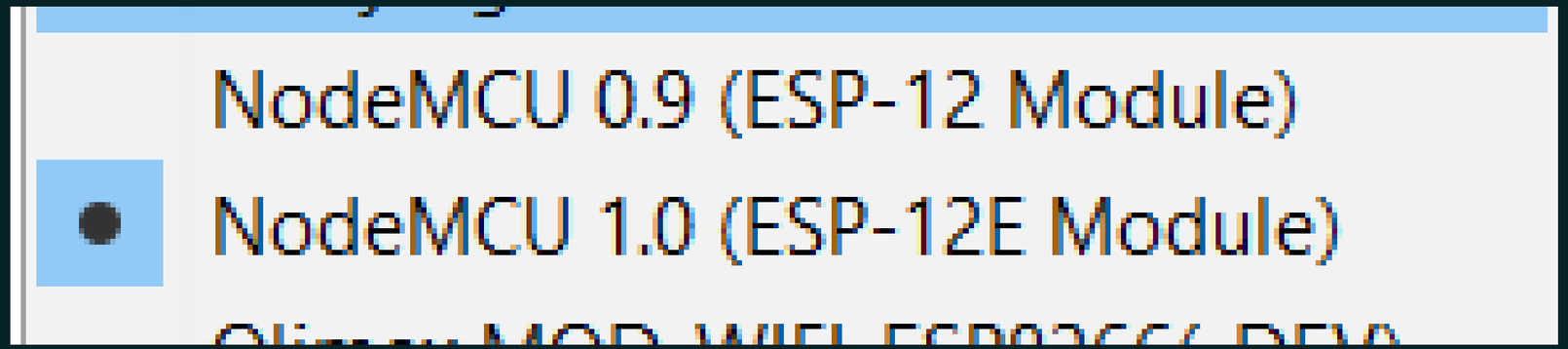
File -> Preferences -> Additional Boards Manager
https://arduino.esp8266.com/stable/package_esp8266com_index.json



Tools -> Board -> Boards Manager
Search ESP8266



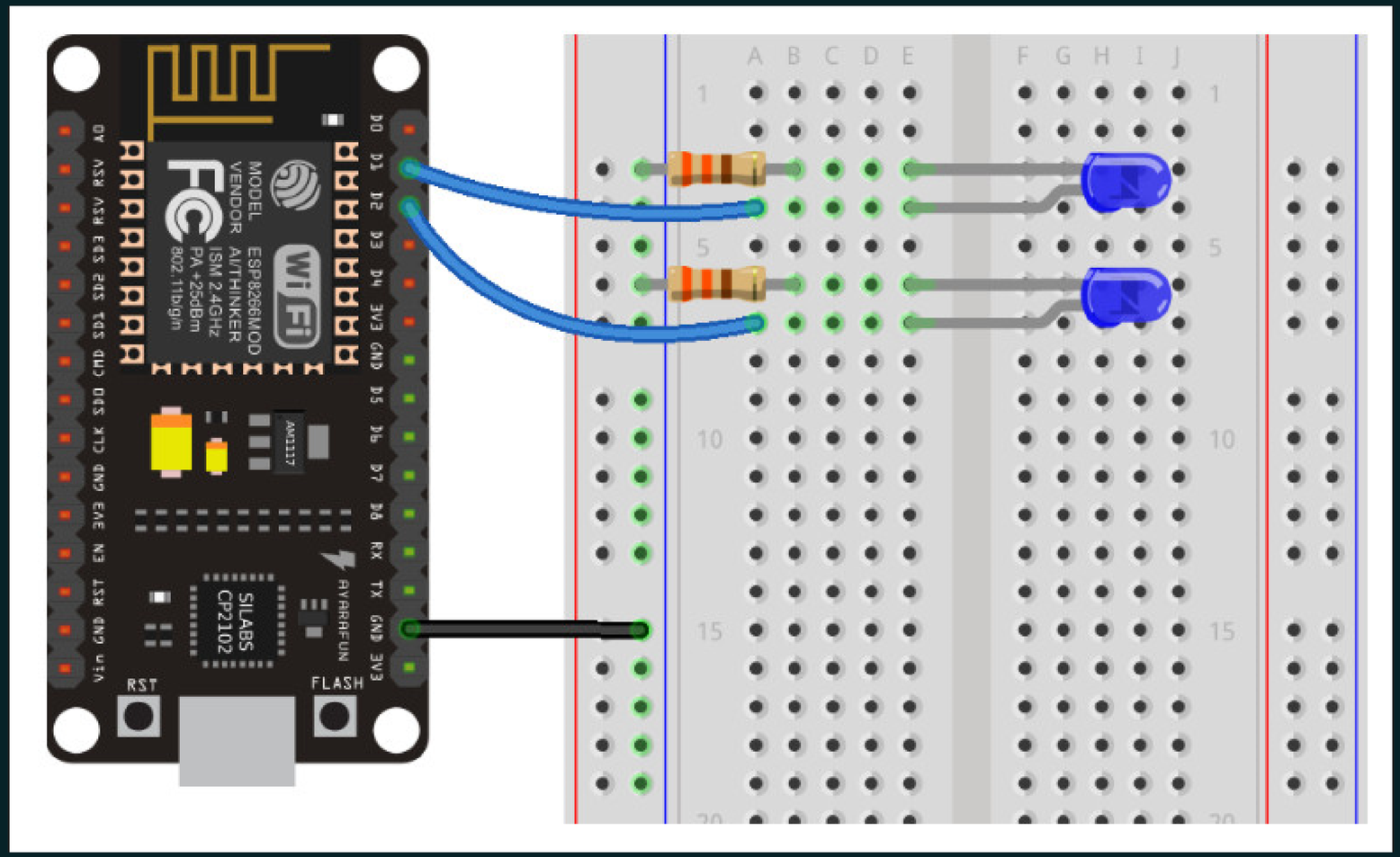
Vào Board chọn như hình bên dưới



IV. CODING

1. Tạo webserver để điều khiển bật tắt đèn

Simulation



```cpp
// Load Wi-Fi library
#include <ESP8266WiFi.h>

// Replace with your network credentials
const char* ssid     = "Thanh Nhat";
const char* password = "666666666";

// Set web server port number to 80
WiFiServer server(80); // 80 co the thay doi duoc , 80 clients truy cập

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String LED1State = "off";
String LED2State = "off";

// Assign output variables to GPIO pins
const int LED1 = 5;
const int LED2 = 4;

// These variables are used to track time for client connections and implement a timeout mechanism for handling HTTP requests.
// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
```

```
  // Set outputs to LOW
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
```

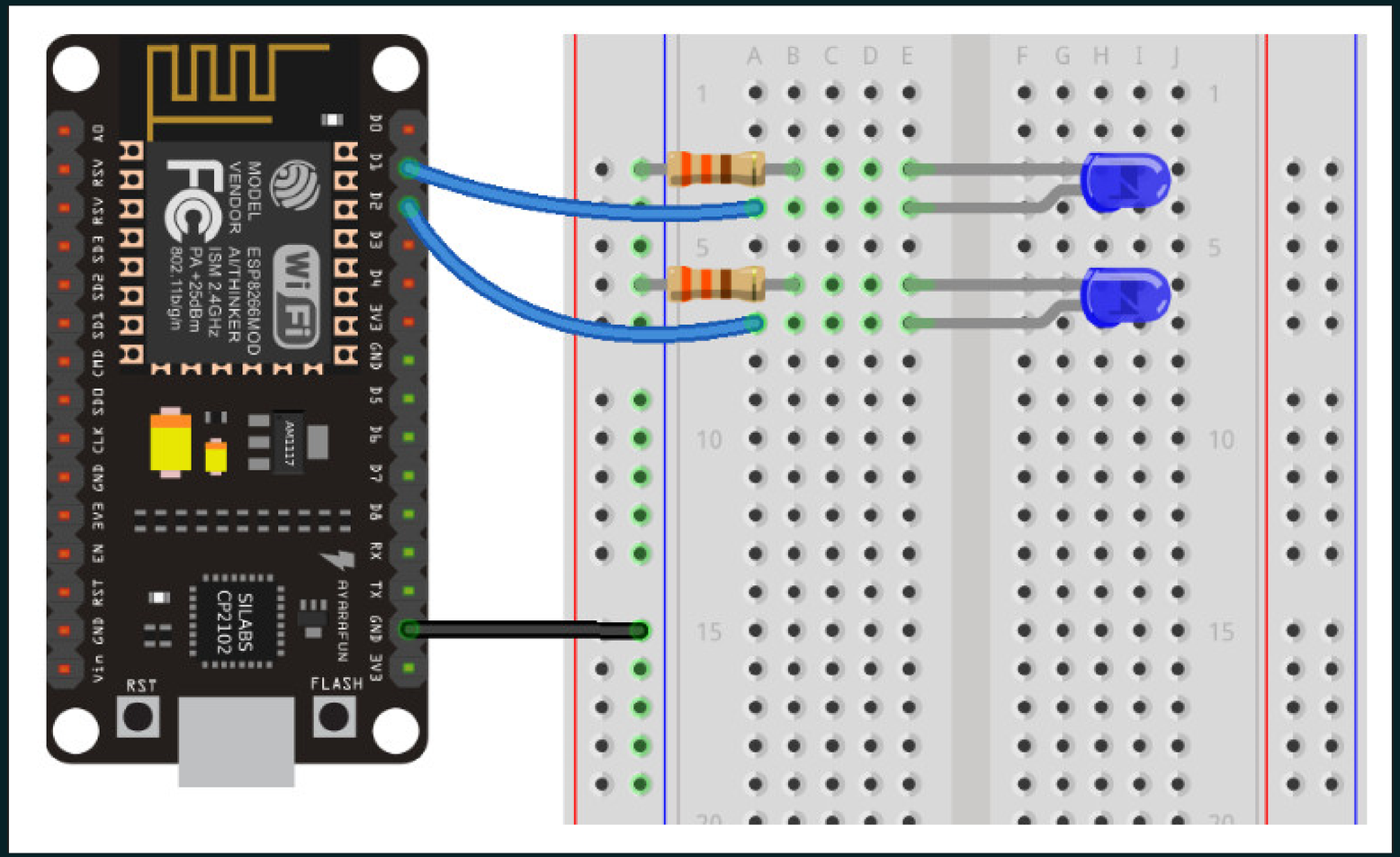
```
70          // if the current line is blank, you got two newline characters in a row.
71          // that's the end of the client HTTP request, so send a response:
72          if (currentLine.length() == 0) {
73            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
74            // and a content-type so the client knows what's coming, then a blank line:
75            client.println("HTTP/1.1 200 OK");
76            client.println("Content-type:text/html");
77            client.println("Connection: close");
78            client.println();
79
80            // turns the GPIOs on and off
81            if (header.indexOf("GET /1/on") >= 0) {
82              Serial.println("LED 1 on");
83              LED1State = "on";
84              digitalWrite(LED1, HIGH);
85            } else if (header.indexOf("GET /1/off") >= 0) {
86              Serial.println("LED 1 off");
87              LED1State = "off";
88              digitalWrite(LED1, LOW);
89            } else if (header.indexOf("GET /2/on") >= 0) {
90              Serial.println("LED 2 on");
91              LED2State = "on";
92              digitalWrite(LED2, HIGH);
93            } else if (header.indexOf("GET /2/off") >= 0) {
94              Serial.println("LED 2 off");
95              LED2State = "off";
96              digitalWrite(LED2, LOW);
97            }
```

```
99            // Display the HTML web page
100           client.println("<!DOCTYPE html><html>");
101           client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
102           client.println("<link rel=\"icon\" href=\"data:,\">");
103           // CSS to style the on/off buttons
104           // Feel free to change the background-color and font-size attributes to fit your preferences
105           client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
106           client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px 40px;");
107           client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
108           client.println(".button2 {background-color: #77878A;}</style></head>");
109
110           // Web Page Heading
111           client.println("<body><h1>ESP8266 Web Server</h1>");
112
113           // Display current state, and ON/OFF buttons for LED 1
114           client.println("<p>LED 1 - State " + LED1State + "</p>");
115           // If the LED1State is off, it displays the ON button
116           if (LED1State=="off") {
117             client.println("<p><a href=\"/1/on\"><button class=\"button\">ON</button></a></p>");
118           } else {
119             client.println("<p><a href=\"/1/off\"><button class=\"button button2\">OFF</button></a></p>");
120           }
121
122           // Display current state, and ON/OFF buttons for LED 2
123           client.println("<p>LED 2 - State " + LED2State + "</p>");
124           // If the LED2State is off, it displays the ON button
125           if (LED2State=="off") {
126             client.println("<p><a href=\"/2/on\"><button class=\"button\">ON</button></a></p>");
127           } else {
128             client.println("<p><a href=\"/2/off\"><button class=\"button button2\">OFF</button></a></p>");
129           }
130           client.println("</body></html>");
```

```
132                     // The HTTP response ends with another blank line
133                     client.println();
134                     // Break out of the while loop
135                     break;
136                   } else { // if you got a newline, then clear currentLine
137                     currentLine = "";
138                   }
139                 } else if (c != '\r') {  // if you got anything else but a carriage return character,
140                   currentLine += c;      // add it to the end of the currentLine
141                 }
142               }
143             }
144             // Clear the header variable
145             header = "";
146             // Close the connection
147             client.stop();
148             Serial.println("Client disconnected.");
149             Serial.println("");
150           }
151         }
```

IV. CODING

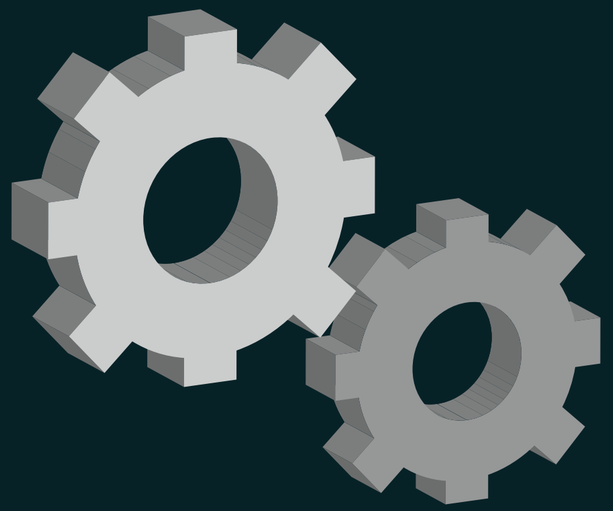
2.Sử dụng ESP8266 tạo webserver , đọc giá trị của cảm biến DHT11 và điều khiển động cơ servo GS90



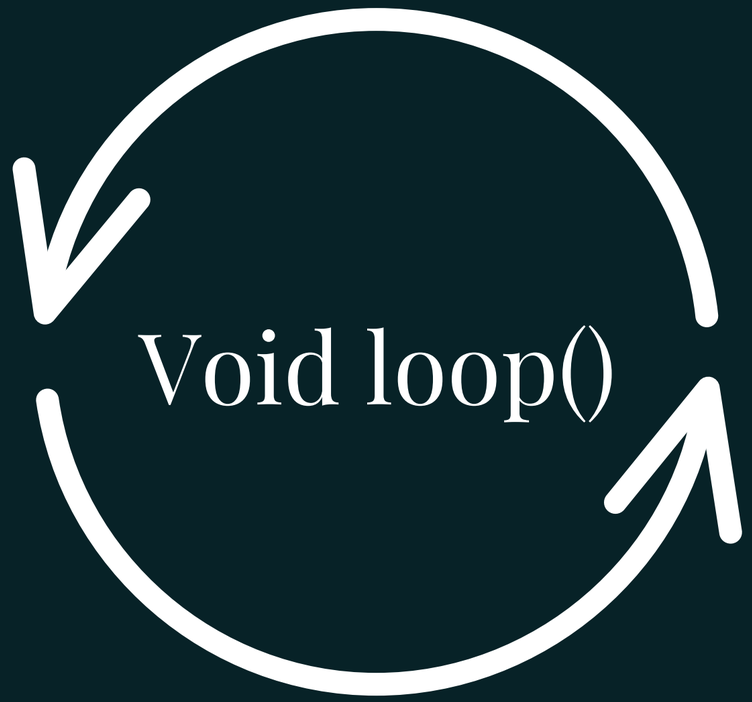
```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Servo.h>

const char* ssid = "ten wifi";
const char* password = "password";

#define DHTPIN 5     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11     // DHT 11
#define ServoPort D4


DHT dht(DHTPIN, DHTTYPE);
float temperature = 0.0;
float humidity = 0.0;

AsyncWebServer server(80);
Servo myservo;

```

Void setup()

```
186  void setup() {
187    Serial.begin(9600);
188    delay(1000);
189
190    WiFi.begin(ssid, password);
191    while (WiFi.status() != WL_CONNECTED) {
192      delay(1000);
193      Serial.println("Connecting to WiFi...");D
194    }
195
196    Serial.println("Connected to WiFi");
197    Serial.println("IP Address: ");
198    Serial.println(WiFi.localIP());
199
200    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
201      request->send_P(200, "text/html", index_html);
202    });
203
204    server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request) {
205      request->send_P(200, "text/plain", String(temperature).c_str());
206    });
207
208    server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request) {
209      request->send_P(200, "text/plain", String(humidity).c_str());
210    });
```



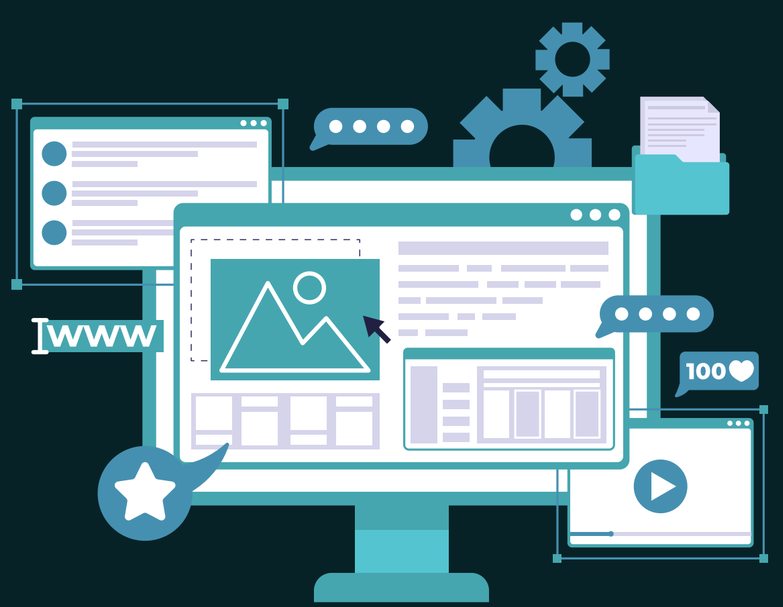
```
211
212    server.on("/setPOS", HTTP_GET, [](AsyncWebServerRequest *request) {
213      if (request->hasParam("servoPOS")) {
214        String posStr = request->getParam("servoPOS")->value();
215        int pos = posStr.toInt();
216        myservo.write(pos);
217        request->send(200, "text/plain", "Servo position set to: " + posStr);
218      } else {
219        request->send(400, "text/plain", "No servoPOS parameter");
220      }
221    });
222
223    server.begin();
224
225    dht.begin();
226    myservo.attach(D4,600, 2300); // Servo signal pin
227  }
```



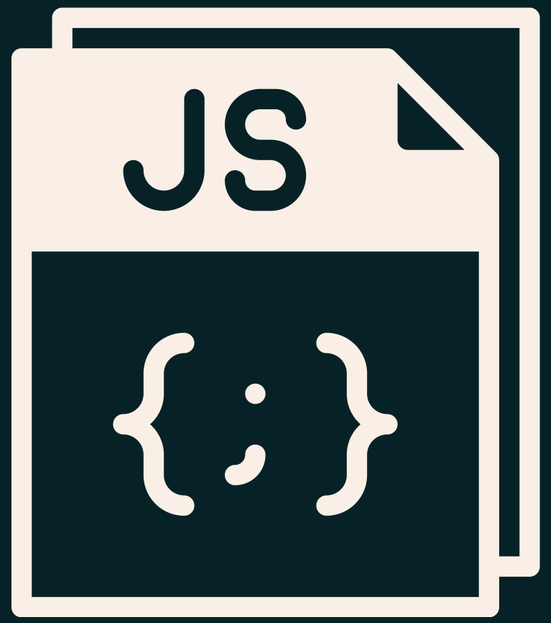

```
229  void loop() {
230    delay(2000);  // Delay between sensor readings
231
232    float newTemperature = dht.readTemperature();
233    float newHumidity = dht.readHumidity();
234
235    if (isnan(newTemperature) || isnan(newHumidity)) {
236      Serial.println("Failed to read from DHT sensor!");
237    } else {
238      temperature = newTemperature;
239      humidity = newHumidity;
240      Serial.print("Temperature: ");
241      Serial.println(temperature);
242      Serial.print("Humidity: ");
243      Serial.println(humidity);
244    }
245  }
```


web design
25 const char index_html[] PROGMEM = R"rawliteral(
26 <!DOCTYPE html>
27 <html lang="en">···
181 </html>
182
183)rawliteral";
184

```
<style>
  /*chung*/
  html {
    font-family: Arial;
    display: inline-block;
    margin: 0px auto;
    text-align: center;
  }

  /*DHT11 sensor*/

  h2 {
    font-size: 3rem;
  }
  .dht {
    font-size: 3rem;
  }
  .units {
    font-size: 1.2rem;
  }
  .dht-labels {
    font-size: 1.5rem;
    vertical-align: middle;
    padding-bottom: 15px;
  }
```

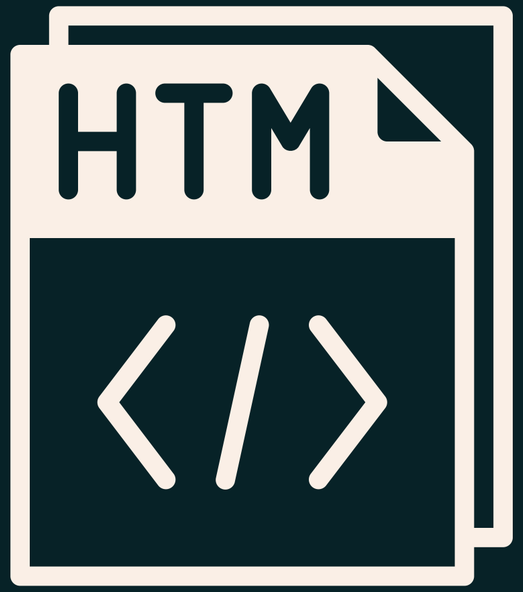



```
<body>
  <!--DHT11 sensor-->

  <h2>ESP8266 Receive Value From DHT Server</h2>
  <p class="dht">
    <i class="fas fa-thermometer-half" style="color: #059e8a"></i>
    <span class="dht-labels">Temperature</span>
    <span id="temperature">%TEMPERATURE%</span>
    <sup class="units">&deg;C</sup>
  </p>
  <p class="dht">
    <i class="fas fa-tint" style="color: #00add6"></i>
    <span class="dht-labels">Humidity</span>
    <span id="humidity">%HUMIDITY%</span>
    <sup class="units">%</sup>
  </p>
```




```
<script>
  // DHT11 sensor

  setInterval(function () {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("temperature").innerHTML = this.responseText;
      }
    };
    xhttp.open("GET", "/temperature", true);
    xhttp.send();
  }, 10000);

  setInterval(function () {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("humidity").innerHTML = this.responseText;
      }
    };
    xhttp.open("GET", "/humidity", true);
    xhttp.send();
  }, 10000);

```

```css
/*Servo Sg90*/

.slidecontainer {
  width: 100%;
}

.slider {
  -webkit-appearance: none;
  width: 50%;
  height: 15px;
  border-radius: 5px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: 0.2s;
  transition: opacity 0.2s;
}

.slider:hover {
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #4caf50;
  cursor: pointer;
}

.slider::-moz-range-thumb {
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #4caf50;
  cursor: pointer;
}
</style>
```




```html
<!--servo sg90-->

<h2>ESP8266 Control Servo SG90</h2>
<br /><br />
<div class="slidecontainer">
  <input
    type="range"
    min="0"
    max="180"
    value="50"
    class="slider"
    id="myRange"
  />
  <p>Value : <span id="demo"></span></p>
</div>
</body>
```

```
135     // Servo Sg90
136
137     function sendData(pos) {
138       var xhttp = new XMLHttpRequest();
139       xhttp.onreadystatechange = function () {
140         if (this.readyState == 4 && this.status == 200) {
141           console.log(this.responseText);
142         }
143       };
144       xhttp.open("GET", "setPOS?servoPOS=" + pos, true);
145       xhttp.send();
146     }
147     var slider = document.getElementById("myRange");
148     var output = document.getElementById("demo");
149     output.innerHTML = slider.value;
150
151     slider.oninput = function () {
152       output.innerHTML = this.value;
153       sendData(output.innerHTML);
154     };
155   </script>
```

THANK YOU

Mong cô và các bạn hoan hỉ

A di da phat🙏🙏🙏


Cáo Từ !!