**Báo cáo tuần 12**

# **Công việc đã làm trong tuần qua**

Tuần qua, em đã cài đặt web server trên ESP8266 để có thể cài đặt thông tin WiFi mà thiết bị sẽ kết nối đến, áp dụng khi chuyển đổi mạng Wifi hoặc lần sử dụng đầu tiên thay vì thiết lập cứng trong mã nguồn.

Toàn bộ mã nguồn và báo cáo em cập nhật hàng tuần trên github: <https://github.com/huydinh3010/DoAnHeNhung>

1. **Ý tưởng**

Để có thể kết nối đến cơ sở dữ liệu lưu trên Firebase, thiết bị cần phải kết nối Internet thông qua Wifi, thông tin về mạng Wifi (ssid, password) sẽ được lưu trong EEPROM để có thể thay đổi khi cần, tránh việc thiết lập cứng trong mã nguồn.

Mỗi khi thiết bị khởi động lại sẽ đọc giá trị byte đầu tiên trong EEPROM, byte này sẽ cho biết đã có thông tin về mạng Wifi hay chưa. Nếu có sẽ đọc tiếp các byte tiếp theo để lấy ra ssid và password sau đó kết nối đến mạng wifi và hoạt động ở chế độ chính.

Nếu chưa có thông tin về wifi thì thiết bị sẽ chuyển sang chế độ thứ 2, sử dụng web server để lấy thông tin mạng wifi được nhập từ phía người dùng. Lúc này thiết bị sẽ đóng vai trò là một Access point có ssid là “ESP8266”, khởi tạo và chạy một local web server hoạt động trong mạng LAN. Người dùng sẽ kết nối vào Access point đó và truy cập từ trình duyệt vào địa chỉ “192.168.1.1” sau đó nhập thông tin và submit, thiết bị sẽ lưu thông tin vào EEPROM.

1. **Cài đặt**

Chế độ Access Point với địa chỉ IP tự thiết lập, đặt trong hàm setup():

WiFi.mode(WIFI\_AP); // chế độ Access point

WiFi.softAP(“ESP8266”,”12345678”); // ssid và password

// thiết lập ip

IPAddress ip(192,168,1,1);

IPAddress n\_mask(255,255,255,0);

WiFi.softAPConfig(ip, ip, n\_mask); // ip, defaul gateway, netmask

Khởi tạo và chạy web server, sử dụng thư viện “ESP8266WebServer.h”:

* Phương thức *on(String uri, function handler)*: đăng ký hàm callback mỗi khi có sự kiện từ client trong phạm vi đường dẫn uri.
* Phương thức *begin()*: khởi chạy đối tượng web server
* Phương thức *handleClient()*: đặt trong hàm loop để kiểm tra và kích hoạt hàm callback sự kiện tương ứng.
* Phương thức *send(int code, String content\_type, String content)*: gửi đến client các chuỗi, văn bản html.
* Phương thức *hasArg(String arg)* và *arg(String arg)*: kiểm tra và lấy ra các tham số của HTTP request (GET, POST,…) từ phía client. Sử dụng để lấy ssid và password sau khi người dùng submit từ trình duyệt.

ESP8266WebServer server(80); // khởi tạo server trên port 80 (HTTP)

void handleWSRoot(){

// kiểm tra request từ client

// server.hasArg(“arg”), server.arg(“arg”);

...

// trả về các chuỗi html tương ứng...

String html = “...”;

Server.send(200, “text/html”, html);

}

void setup(){

...

server.on(“/”, handleWSRoot);

server.begin();

...

}

void loop(){

...

server.handleClient();

...

}

Mã html và giao diện trang web:

<html>

<head>

<meta charset='UTF-8'>

<h2>Enter wifi information to connect the device</h2>

</head>

<body>

<form method='post'>

WiFiName: <input type='text' name='name' value='' pattern='.{1,}' required title='1 characters minimum'/> <br/>

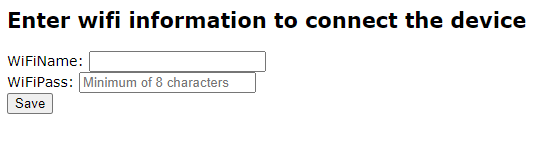
WiFiPass: <input type='text' name='pass' placeholder='Minimum of 8 characters' value='' pattern='.{8,}' required title='8 characters minimum'/> <br />

<input class='button' type='submit' value='Save'/>

</form>

</body>

</html>



*Giao diện trang web*

Lưu trữ, đọc và xóa thông tin trong EEPROM:

* Lưu thông tin wifi vào EEPROM khi nhận được HTTP POST request gồm 2 tham số “name” – ssid và “password” – password từ client. Lưu theo định dạng: *“<1><ssid><0><password><0*>”, trong đó byte đầu tiên bằng 1 để báo hiệu đã có thông tin wifi.

EEPROM.begin(512); // bắt đầu với 512bytes EEPROM

EEPROM.write(0, 1); // ghi tại địa chỉ 0, byte 1

delay(5); // đợi 5ms sau mỗi lần ghi 1 byte

int e\_in = 1;

for(int i = 0; i < wf\_name.length(); i++){ // ghi ssid

EEPROM.write(e\_in++, (int)wf\_name[i]);

delay(5);

}

EEPROM.write(e\_in++, 0);

delay(5);

for(int i = 0; i < wf\_pass.length(); i++){ // ghi password

EEPROM.write(e\_in++, (int)wf\_pass[i]);

delay(5);

}

EEPROM.write(e\_in++, 0);

delay(100);

EEPROM.end();

* Đọc thông tin từ EEPROM, kiểm tra byte đầu tiên nếu khác 0 thì đọc các byte tiếp theo để lấy ra ssid và password.

EEPROM.begin(512);

int w\_a = EEPROM.read(0); // đọc byte địa chỉ 0 – byte đầu tiên

if(w\_a){ // nếu có thông tin (w\_a != 0)

int i = 1;

int \_b = -1;

WIFI\_SSID = "";

WIFI\_PASSWORD = "";

while(\_b != 0 && i < 512){ // đọc ssid

\_b = EEPROM.read(i++);

if(\_b != 0) WIFI\_SSID += (char)\_b;

}

\_b = -1;

while(\_b != 0 && i < 512){ // đọc password

\_b = EEPROM.read(i++);

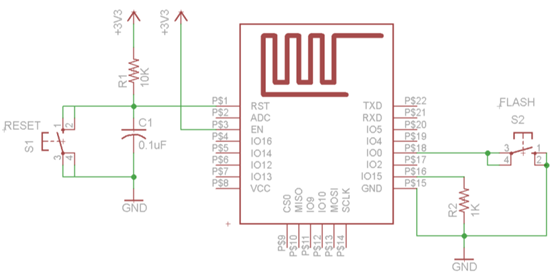
if(\_b != 0) WIFI\_PASSWORD += (char)\_b;

}

EEPROM.end();

}

* Xóa thông tin trong EEPROM, mục đích là để cập nhật lại thông tin wifi khi cần thiết. Sử dụng nút Flash trên board, nút này được nối sẵn với chân GPIO-0 theo sơ đồ:



*Sơ đồ nút Flash trên broad*

Khi được đặt chế độ chân 0 là INPUT\_PULLUP thì chân số 0 này sẽ có giá trị mức cao và kéo xuống mức thấp khi nút Flash được nhấn. Bắt sự kiện nút nhấn và thực hiện đoạn mã để xóa byte đầu tiên trong EEPROM về 0.

EEPROM.begin(512);

int w\_a = EEPROM.read(0); // kiểm tra giá trị byte đầu tiên

delay(200);

if(w\_a){

EEPROM.write(0, 0); // xóa byte đầu tiên về 0

delay(5);

}

EEPROM.end();

# **Dự kiến các tuần còn lại**

Ngoài việc tinh chỉnh giao diện ứng dụng, tìm và sửa các lỗi phát sinh, dưới đây là tính năng em dự kiến bổ sung trong các tuần cuối nếu đủ thời gian:

* Tìm kiếm và thêm một số mã hồng ngoại của các hãng thiết bị phổ biến.