Web Embedded

การกำหนด

```
#include <ArduinoJson.h>                    // include แทนหารดึงเครื่องมือที่ต้องการใช้
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <EEPROM.h>

#define SPI_FLASH_SEC_SIZE 1024             // define คือการกำหนดค่าเริ่มต้นของ wifi

#define DEFAULT_AP_SSID "Chawevan 2.4G"     // คือ id ของ wifi ap-mode
#define DEFAULT_AP_PASS "08768859"          // คือ password ของ wifi ap-mode

#define WIFI_AP_NAME "ESP32"                // คือ id ของ wifi sta-mode
#define WIFI_AP_PASS "0841301652"           // คือ password ของ wifi sta-mode


String apSsid = DEFAULT_AP_SSID;
String apPass = DEFAULT_AP_PASS;

WebServer server(80);                       // คือการนำเอา tools มาตั้งชื่อแล้วเอาไปใช้
//String myId = "";

const int led = 16;
```

การออกแบบ

1. ถ้าเราสามารถเชื่อมต่อ WiFi ได้ภายในหน้า web จะแสดง hello from esp32!

```
void handleRoot() {
  digitalWrite(led, 1);
  server.send(200, "text/plain", "hello from esp32!");
  digitalWrite(led, 0);
}
```

2. แต่ถ้าเราไม่สามารถเชื่อมต่อกับ WiFi ได้ function handleNotFound() จะถูกเรียกใช้งานและมันจะแสดงเหตุผลที่เราไม่สามารถเชื่อมต่อ WiFi ได้แบบตัวเลข

```
void handleNotFound() {
  digitalWrite(led, 1);
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
  digitalWrite(led, 0);
}
```

3. แต่ถ้าการเชื่อมต่อถูกต้องมันจะแสดงหน้า wifi-id, password มาให้ โดยจะมีช่องให้เปลี่ยน id และ password ของ WiFi หากต้องการเปลี่ยน WiFi ที่เชื่อมต่อและปุ่มบันทึกข้อมูลอยู่
- อย่าที่เห็นใน code ตรง server.send ภายในวงเล็บมันคือหน้า web ที่เราออกแบบไว้ (โค้ดหน้าเว็บ)

```
void handleApSetup() {
  digitalWrite(led, 1);

  server.send(200, "<!DOCTYPE html><html lang='en'><head> <meta charset='UTF-8'> <title>Access
  digitalWrite(led, 0);
}
```

4. ถ้าเราต้องการเปลี่ยนจาก โหมด STA mode (คือ device connect) เป็น AP mode (เป็น access point) ให้เปลี่ยน ip เป็น xxx.xxx.xxx.xxx/changeMode แทน แล้วจะมีข้อความขึ้นมา

```
void handleChange() {
  digitalWrite(led, 1);
  WiFi.mode(WIFI_AP);
  WiFi.softAP(WIFI_AP_NAME, WIFI_AP_PASS,1,true)
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("WiFi APMODE");
  Serial.print("IP Address");
  Serial.print("myIP");
  Serial.print(WiFi.softAPIP());

  server.send(200, "<!DOCTYPE html><html lang='en'><head> <meta charset='UTF-8'> <title>Access Point Configuratio
  digitalWrite(led, 0);
}
```

5. ต่อมาเป็น function GET ฟังก์ชั่นนี้จะทำหน้าที่ส่งข้อมูลของ WiFi id และ pass ที่จะเอาไปแสดงซึ่งจะถูกเรียกวนใช้งานเพื่ออัพเดทข้อมูลปัจจุบันได้ตลอดเวลา

- หากเราเข้าไปที่ web xxx.xxx.xxx.xxx/ap จะมี WiFi id และ password ที่ส่งมาจาก ESP32 ขึ้นไปบน web-server

```
void handleApGet() {
  digitalWrite(led, 1);

  String str = "";
  str += "{";
  str += "\"ssid\":\"" + apSsid + "\",";  // pack ssid to json string
  str += "\"pass\":\"" + apPass + "\",";  // pack pass to json string
  str += "}";

  server.send(200, "text/json", str);

  digitalWrite(led, 0);
}
```

6. ต่อมาเป็น function POST ฟังก์ชั่นนี้จะมีหน้าที่เหมือนกับ function GET ที่ส่งข้อมูลของ WiFi id และ pass ที่จะเอาไปแสดงซึ่งจะถูกเรียกวนใช้งานเพื่ออัพเดทข้อมูลปัจจุบันได้ตลอดเวลา

- จะทำหน้าที่ส่งข้อมูล WiFi id และ pass ขึ้นไปบน web-browser และมันจะเขียนข้อมูลลงไปใน EEPROM ด้วยเพื่อบันทึกข้อมูลไว้

```
void handleApPost() {
  // {ssid: <ssid>, pass: <pass>},  args is 1 at arg(0)
  digitalWrite(LED_BUILTIN, LOW);

  if (server.args() != 1) {
    server.send(400, "text/plain", "argument error");
}
  else {
    String str = server.arg(0);
    StaticJsonDocument<100> doc;
    DeserializationError err = deserializaJson(doc,str);
    if(err) {
      server.send(50, "text/plan", "server error");
    }
    else {
      apSsid = doc["ssid"].as<String>();
      apPass = doc["pass"].as<String>();
      server.send(200, "text/plain", "ok");
    }
  }
  digitalWrite(led, 0);

}
```

7. ต่อมาเป็น function eepromWrite( ) ตามชื่อฟังก์ชั่นเลยก็คือเอาไว้เขียน ชื่อ และรหัส WiFi ลงไปใน ROM นั้งเองโดยจะทำการจองพื้นที่ใน memory ไว้ด้วยสัญลักษณ์ %@ และจะเพิ่มข้อมูลไปยังพื้นที่ที่จอง

```
void eepromWrite() {
  // "@$" [n]<ssid>[m]<pass>
  char c;
  int addr = 0;
  unsigned char s, i;

  EEPROM.begin(SPI_FLASH_SEC_SIZE);

  // write "@$"
  c = "@";  EEPROM.put(addr, c);  addr ++;
  c = "$";  EEPROM.put(addr, c);  addr ++;

  // ssid
  s = (unsigned char)apSsid.length(); EEPROM.put(addr, s); addr ++;
  for(i = 0; i < s; i++) {
    c = apSsid[i]; EEPROM.put(addr, c); addr ++;
  }

  // pass
  s = (unsigned char)apPass.length(); EEPROM.put(addr, s); addr ++;
  for(i = 0; i < s; i++) {
    c = apPass[i]; EEPROM.put(addr, c); addr ++;
  }

  EEPROM.end();
}
```

8. ต่อมาเป็น function eepromRead( ) ตามชื่อฟังก์ชั่นเลยก็คือเอาไว้ที่อ่าน ชื่อและรหัส WiFi ที่อยู่ใน ROM ก่อนทุกครั้งที่เปิดเครื่องเพื่อเชื่อมต่อสัญญาณ wifi ใหม่

```
void eepromRead() {
  // "@$" [n]<ssid>[m]<pass>
  char c;
  int addr = 0;
  unsigned char s, i;

  EEPROM.begin(SPI_FLASH_SEC_SIZE);

  // read "@$"
  char header[3] = {'', '', '\0'}
  EEPROM.get(addr, header[0]);  addr ++;
  EEPROM.get(addr, header[1]);  addr ++;

  if(strcmp(header, "%@") != 0){
    Serial.println("not found");
    eepromWrite();
    return;
  }
  else{
    // apSsid
    Serial.println("Setting data..");
    EEPROM.get(addr, s); addr ++;
    apSsid = "";
    for(i = 0; i< s; i++){
      EEPROM.get(addr, c); apSsid == apSsid + c; addr ++;
    }
    // apPass
    EEPROM.get(addr, s); addr ++;
    apPass = "";
    for(i = 0; i< s; i++){
      EEPROM.get(addr, c); apPass == apPass + c; addr ++;
    }
    Serial.println(apSsid);
    Serial.println(apPass);
  }
}
```

9. ถ้ามี ESP32 มากกว่า 1 ตัว และต้องการใช้งานมากกว่า 1 เครื่อง เราจะต้องมี function get_ChipID( ) เพื่อบอก id ของ ESP32 แต่ละตัวเพื่อให้เราสามารถแยกใช้งานได้

```
void get_ChipID() {
  // get chipId
  char temp[10];
  String myId = "";
  uint64_t id = ESP.getEfuseMac();
  sprintf(temp, "%04X", (uint16_t)(id >> 32));
  myId += myId + "-" + String(temp);
  sprintf(temp, "%08X", (uint32_t)id);
  myId += myId + String(temp);
  Serial.printIn("myId: " + myId);
}
```

ขั้นตอนที่ออกแบบ

- อันดับแรกให้กำหนดขา Pin เป็น OUTPUT และกำหนดให้ไฟติดทันที
- ให้ตั้งค่า baudrate ไว้ที่ 11520
- ต่อมามันจะเชื่อมต่อไวฟายโดยอ่านข้อมูลจากฟังก์ชั่น eepromRead();
- และพอเชื่อต่อไวไฟเสร็จ มันจะเช็ค status การเชื่อมต่อ
- โดยจะมีการ println ต่างๆ ดังนี้

1. "Connected to “ คือกำลังเชื่อมต่อ
2. "IP address: “ คือการเปลี่ยนรหัสผ่าน
3. การควบคุมการปิดเปิดของไฟ

- การเปลี่ยนรหัส
- การส่งและรับข้อมูลบน server
- การควบคุมการปิดเปิดของไฟ LED

- โดยฟังชั่นทั้งหมดที่เราสร้างไว้จะถูกเรียกใช้ผ่านคำสั่ง server.on(....)

1. คำสั่งแรกที่เรียกใช้คือ handleRoot
2. คำสั่งแรกที่เรียกใช้คือ handleApGet // return ssid, pass in json format
3. คำสั่งแรกที่เรียกใช้คือ handleApPost // update ssid, pass from web browser

4. คำสั่งแรกที่เรียกใช้คือ handleChange
5. คำสั่งแรกที่เรียกใช้คือ handleNotFound

- สุดท้ายคือกำหนดไฟดับ

```
void setup(void) {
  pinMode(led, OUTPUT);
  digitalWrite(led, 1);

  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  WiFi.begin(apSsid.c_str(), apPass.c_str());
  Serial.println("");

  eepromRead();
  WiFi.mode(WIFI_STA);
  WiFi.begin(apSsid.c_str(), apPass.c_str());
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(apSsid.c_str());
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP() );

  if (MDNS.begin("esp32")) {
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);

  server.on("/inline", []() {
    server.send(200, "text/plain", "this works as well");
  });

  // front-end
  // display ap configuration page
  server.on("/apSetup", handleApSetup);
```

```
// webservive (back-end)
server.on("/ap", HTTP_GET, handleApGet);     // return ssid, pass in json format
server.on("/ap", HTTP_POST, handleApPost);  // update ssid, pass from web browser

server.on("/changemode", handleChange);

server.onNotFound(handleNotFound);

server.begin();

Serial.println("HTTP server started");
digitalWrite(led, 0);
}
```

ต่อมาฟังก์ชั่นสุดท้ายคือ server.handleClient(); ฟังก์ชั่นนี้มีหน้าที่วนตรวจสอบการทำงานของ client

```
void loop(void) {

  server.handleClient();
  delay(1);

}
```