ESP8266 Mesh Networks

Topology
- ให้ ESP ตัวหนึ่งเชื่อมต่อกับ PC เรียกตัวนี้ว่า BASE และมี ssid AP เป็น _MeshNode
- ESP ที่ยังไม่มี ID จะมองหา Parent ขณะนี้มีตัวเดียวคือ _MeshNode เมื่อเชื่อมต่อได้และร้องขอ ID เมื่อได้ ID แล้ว ก็นำมาตั้งเป็น ssid ของตัวเอง ในที่นี้จะได้ _MeshNode1
- ขณะนี้ถ้ามี ESP อีกตัวต้องการ ID อาจจะเชื่อมต่อกับ _MeshNode หรือ _MeshNode1 ขึ้นอยู่กับว่าจับสัญญานตัวไหนได้ก่อน ถ้าจับ _MeshNode ได้ก่อน ก็จะได้ ID เป็น _MeshNode2 แต่ถ้าจับได้ _MeshNode1 ก็จะได้ ID เป็น _MeshNode11
- สมมุติว่า ESP อีกตัวจับ _MeshNode1 ได้ ตัวนี้จะได้ ID เป็น _MeshNode21
- สังเกตุการกำหนด ID นะครับ
- เรียกตัวที่มาจับว่า client และตัวที่ถูกจับเรียกว่า parent
- ข้อจำกัดตอนนี้ของ ESP คือมี client ได้มากสุด 4 client
- ส่วนสำคัญคือ parent จะปลด client ออก เมื่อทำการส่งข้อมูลไปแล้วไม่มีการตอบรับจาก client เกินกว่าจำนวนครั้งที่กำหนด
- เช่นกัน client จะตัดตัวเองออกจาก parent เมื่อไม่ได้การตอบรับ ตามจำนวนครั้งที่กำหนด
- การปลดตัวเองออกจาก parent จะทำให้ client สามารถเชื่อมต่อกับ parent ตัวอื่นที่ทำงานอยู่ และทำให้ client อยู่ในระบบได้ตลอดเวลา
- ส่วนของ parent เมื่อปลด client ออก ก็จะเกิดห้องว่างให้ ESP ตัวอื่นสามารถเข้ามาเชื่อมต่อได้
- การกระจายข้อมูลเพื่อนำคำสั่งไปที่ ESP ตัวที่ต้องการ จะเริ่มต้นที่ PC ผ่าน BASE
- BASE จะส่งต่อข้อมูล (packet) ไปยัง client ทุกตัวที่เชื่อมต่ออยู่ (1,2,3,4) และ client ทุกตัวที่มี client เชื่อมต่ออยู่ก็จะส่งข้อมูล ต่อๆกันไป
- ข้อมูลใน packet จะมีคำสั่งติดไปด้วย โดยระบุว่าจะให้ ESP ตัวไหนทำงาน ซึ่ง ESP จะถูกกำหนดชื่อไว้แล้ว เมื่ออ่านแล้วชื่อในคำสั่งเป็นของตัวเอง ก็จะทำคำสั่งและตอบข้อมูลกลับไปที่ BASE ผ่าน parent และ parent ก็จะส่งไปให้กับ parent ของตัวเอง จนถึง BASE และข้อมูลนั้นก็จะไปปรากฏที่ PC
- จะเห็นว่าการเชื่อมต่อ ปลดตัวเองออก ทำโดยอัตโนมัติ ทำให้ระบบสามารถทำงานได้ตลอดเวลา แม้จะมี ESP บางตัวเกิดความเสียหาย
- ESP จะทำตัวเป็น AP ตลอดเวลา และจะทำตัวเป็น STATION ก็ต่อเมื่อต้องการส่งผ่านข้อมูลไปยัง parent หรือ client เท่านั้น
- การเชื่อมต่อกับ internet จะอยู่ที่ PC หรืออาจเป็น ESP ตัวหนึ่ง ทำให้ใช้ IP Address จาก router เพียงตัวเดียว
- ข้อด้อยที่เห็นและพยายามปรับปรุง คือการใช้เวลาในการส่งข้อมูลที่ผ่านไปแต่ละ hop จะอยู่ที่ 2-3 วินาที เวลาที่ใช้มากสุดคือระหว่างที่ client พยายามในเชื่อมต่อกับ parent

- อาจถามว่าทำไมไม่เชื่อมต่อกันตลอดเวลา คำตอบอยู่ในบรรทัดบนๆแล้ว เนื่องจากทุกตัวจะต้องทำตัวเป็น AP ตลอดเวลานั่นเอง

Mesh network ตอนที่ 1 ทำความรู้จักไลบรารี่

#include <ESP8266WiFi.h>

#include <ArduinoJson.h>

#include <EEPROM.h>

มีทั้งหมด 3 ไลบรารี่

1. ESP8266WiFi

ทำหน้าที่ทั้งหมดในการเชื่อมต่อ ESP แบบ WiFi

ส่วนที่นำมาใช้ได้แก่

1.1 WiFiServer

สร้างตัวแปร server เพื่อรับและตอบกลับกับ client

1.2 WiFiClient

ช่องทางในการร้องขอและรับข้อมูลจาก server

1.3 WiFi

กำหนดบทบาทให้เป็น AP หรือ station

ตั้งชื่อ AP ตั้ง Password

ควบคุมและตรวจสอบการทำงานของ WiFi

2. ArduinoJson

JSON เป็นรูปแบบของการจัดวางข้อมูล ลักษณะ key:value เป็นที่นิยมใช้กับโปรแกรมที่ไม่ต้องการใช้ระบบฐานข้อมูล โดยทำการ encode ที่ต้นทาง และ decode ที่ปลายทาง

สำหรับ Arduino ได้สร้างไลบรารี่มาช่วยให้เราทำงานได้ง่ายขึ้น โดยละเอียดจะอธิบายในเนื้อหาโค๊ด

3. EEPROM

สำหรับ EEPROM เองบอกความหมายในตัวเองแล้วว่า ใช้บันทึกข้อมูลในหน่วยความจำ ที่ไม่สูญหายไปแม้จะไม่มีพลังงานป้อนอยู่ ในที่นี้จะใช้เก็บหมายเลข node ซึ่งเมื่อป้อนไฟให้ก็จะถูกนำเอาไปตั้งชื่อของ device

แบ่งเป็นตอนๆเล็กๆครับ ตอนหน้าจะเอาไลบรารี่ไปวางในโค๊ด ทีละส่วน

>>>>>>>>>>>>>>>>>>>>>>>>

> หมายเหตุ >

>>>>>>>>>>>>>>>>>>>>>>>>

เนื้อหาที่เขียนจะยาวมากหลายตอน ก็เลยขอให้ผู้ติดตามที่เข้ามาจากเพจอื่น ช่วยไลค์เพจนี้ จะได้ติดตามได้สะดวกขึ้น ทุกครั้งเมื่อโพสต์แล้วจะทำการแชร์จากเพจอื่นๆที่ผมเป็นสมาชิกอยู่ เกรงว่าถ้าหลงลืมไม่ได้แชร์ จะติดตามกันไม่สะดวก

แต่หลักๆจะยังแชร์ไปที่ ESP8266 Thailand และ Node-RED Siam อยู่ครับ

>>
>>>>>>>>>>>>>>>>>>>
Mesh network ตอนที่ 2 JSON
JavaScript Object Notation เป็นการส่งข้อมูลระหว่าง server กับ client ปัจจุบันไม่ได้ใช้เพียงกับ JavaScript เท่านั้น
วิธีการแยกข้อมูล

 ตัวข้อมูลมีรูปแบบที่เป็นตัวอักษร เมื่อนำมาดูจะเข้าใจได้ง่ายๆ ตัวอย่างของข้อมูลรูปแบบ
{firstname:Supot,lastname:Sae-Ea}
จะเห็นได้ว่าเราสามารถตีความได้เลย คือ
 ชื่อตัวแปร(key)คือ firstname มีค่า(value)คือ Supot
 ชื่อตัวแปร(key)คือ lastname มีค่า(value)คือ Sae-Ea
มีเครื่องหมาย : คั่นระหว่าง key กับ value
 มีเครื่องหมาย , คั่นระหว่างชุด key-value
 ใช้เครื่องหมาย { } จุดเริ่มต้นและสิ้นสุดของชุด key-value
การซ้อนชุด key-value ตัวอย่าง
{firatname:Supot,lastname:Sae-Ea,info:{sex:male, hobby:blogger}}
info เป็น key มี value เป็น sex:male, hobby:blogger
จะแนะนำไว้เท่านี้ครับ จริงแล้วจะมีการสร้าง array อีกด้วย แต่ในเนื้อหาไม่ได้ใช้
การใช้งานไลบรารี่ ArduinoJson
การนำค่าออกมาใช้จาก key:value

```
1 String json = "{firstname:Supot,lastname:Sae-Ea};
2 StaticJsonBuffer<500> jsonBuffer;
 3 JsonObject& root = jsonBuffer(json);
4 String firstname = root["firstname"].asString();
 5 String lastname = root["lastname"].asString();
```

บรรทัดที่ 1 เป็นตัวอย่างข้อมูล
 บรรทัดที่ 2 สร้างตัวแปรเพื่อเก็บข้อมูล <500> คือจำนวน byte
 บรรทัดที่ 3 สร้าง root เป็นตัวแปรแบบ JsonObject และทำการตีความ json
 บรรทัดที่ 4 และ 5 ดึงค่าออกจาก root โดยการระบุชื่อ key
วิธีการสร้าง json

ต้องการสร้าง {firstname:Supot, lastname:Sae-Ea}
 เขียนโค๊ดตามนี้

```
1 StaticJsonBuffer jsonBuffer;
 2 JsonObject& root = jsonBuffer.createObject();
3 root["firstname"] = "Supot";
 4 root["lastname"] = "Sae-Ea";
```

5 String msg = "";
6 root.printTo(msg);
บรรทัดที่ 2 สังเกตุว่าใช้ createObject()
บรรทัดที่ 3 - 4 นำ String ไปเข้ารูปแบบ
บรรทัดที่ 6 ใช้คำสั่ง printTo ทำการนำข้อมูล json ไปเก็บไว้ที่ตัวแปร msg
ถ้าทดลองเขียนโค๊ดตามนี้ และทดลองพิมพ์ msg ออกที่ Serial monitor ดูครับ

สำหรับ JSON จบเท่านี้ครับ

>>>

Mesh network ตอนที่ 3

กำหนดค่าคงที่ ตัวแปร ฟังค์ชั่น และ setup
บรรทัดที่ 1-3 ไลบรารี่ที่ใช้
บรรทัดที่ 5 ชื่อนำหน้าสำหรับตั้งชื่อ Access point ของ device แต่ละตัว โดยจะมีหมายเลขของ device จำนวน 3 หลักต่อท้าย เช่น _MyMesh_001
บรรทัดที่ 6 เป็นตำแหน่งใน EEPROM สำหรับเก็บหมายเลขของ device จำนวน 1 byte ทำให้ระบุได้ 256 หมายเลข อาจเพิ่มตำแหน่งเก็บได้ตามต้องการ ซึ่งต้องมีปรับโค๊ดเล็กน้อย
บรรทัดที่ 8 กำหนดตำแหน่งเก็บข้อมูล IO ในโค๊ดยังไม่ได้นำไปใช้งาน
บรรทัดที่ 9 เป็น GPIO ที่เชื่อมต่อกับ LED เพื่อแสดงผลการทำงาน สำหรับ ESP-01 จะตัดลายเส้นของ LED ที่ใช้แสดงว่ามีการป้อนไฟออก แล้วเชื่อมสายไปที่ขา GPIO-2
บรรทัดที่ 11 เรียกใช้งาน server โดยกำหนดพอร์ทใช้งานที่หมายเลข 4011 (ปกติจะใช้หมายเลข 80) เพื่อรับและตอบสนองการเชื่อมต่อจาก client
บรรทัดที่ 12 เรียกใช้งาน client เพื่อทำการส่งคำร้องและรับผลการตอบสนองจาก servver
บรรทัดที่ 14 สร้างตัวแปรสำหรับเก็บหมายเลขของ device เช่น 001, 002, 123
บรรทัดที่ 15 สร้างตัวแปรสำหรับเก็บชื่อ Access point ของ device เช่น _MyMesh_001, _MyMesh_002, _MyMesh_123
บรรทัดที่ 18 - 22 เป็นฟังค์ชั่นต่างๆที่ใช้ จะแยกอธิบายเป็นส่วนๆ
บรรทัดที่ 24 - 30 กำหนดค่าต่างๆให้พร้อมทำงาน
บรทัดที่ 25 ให้ EEPROM เริ่มทำงาน กำหนดไว้ 100 ตำแหน่ง จริงแล้วใช้ไม่กี่ตำแหน่ง
บรรทัดที่ 26 เริ่มการติดต่อกับพอร์ท Serial
บรรทัดที่ 27 ให้ GPIO-2 เป็นแบบ OUTPUT
บรรทัดที่ 28 ให้ myId เก็บหมายเลข 999
บรรทัดที่ 29 เข้าสู่การตั้งค่า
ถัดจากนี้จะนำโค๊ดของแต่ละฟังค์ชั่นมาอธิบาย

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <EEPROM.h>

#define SSID_PREFIX "_MyMesh_"
#define NODE 0

#define IO      1
#define LEDPIN  2

WiFiServer _server(4011);
WiFiClient _client;

String myId = "";
String ssid = "";


void setupAP();
void acceptRequest();
void acceptCommand();
void forwardMessage();
void forwardToNode(String AP, String message);

void setup() {
  EEPROM.begin(100);
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);
  myId = "999";
  setupAP();
}
```

>>>

Mesh network ตอนที่ 4 setupAP ในส่วนของ void setup()

กำหนดชื่อ AP และเริ่มการทำงานของ server

บรรทัดที่ 47 อ่านหมายเลข device ด้วยคำสั่ง EEPROM.read(NODE); โดยที่ NODE = 0 เป็นตำแหน่งที่เก็บค่าไว้ แล้วแปลงเป็นสตริงจ์เก็บไว้ที่ตัวแปร myId

บรรทัดที่ 48 ทำการเติมเลข 0 ด้านหน้า ทำให้เป็นตัวเลข 3 หลัก เช่น ถ้า myId = "1" ก็จะเติม 0 ด้านหน้าไป 2 ตัว myId จะเป็น 001

บรรทัดที่ 50 นำ myId มารวมกับชื่อของกลุ่ม ในที่นี้คือ _MyMesh_ จะได้ ssid เป็น _MyMesh_001 ที่จะปรากฏในรายการ AP ที่ device อื่นๆมองเห็น

บรรทัดที่ 52 - 61 จะแสดงชื่อ ssid และคำสั่งที่ใช้งานได้ จะมีอยู่ 2 คำสั่งคือ NODE=XXX โดยที่ XXX เป็นหมายเลข device ที่ต้องการ เช่น NODE=5 จะทำให้ device ตัวนี้มีหมายเลข 5 ซึ่งจะแสดงในรายการของ AP เป็น _MyMesh_005

ระมัดระวังในการตั้งค่า ไม่ให้ซ้ำกัน

บรรทัดที่ 62 เป็นการกำหนดความแรงของสัญญาณของ device

บรรทัดที่ 63 กำหนดโหมดของ WiFi เป็น WIFI_AP_STA

บรรทัดที่ 64 ให้ AP เริ่มทำงาน สำหรับ .c_str() ต่อท้ายตัวแปร ssid ทำหน้าที่แปลงตัวแปรสตริงจ์ให้เป็นแบบ character array ถ้าใส่ ssid จะคอมไพล์ไม่ผ่าน

บรรทัดที่ 66 เริ่มการทำงานของ server รอรับคำร้องจาก device ตัวอื่น

สำหรับสองสามตอนที่ผ่านมาจะนำเอาโค๊ด มาเรียงร้อยให้ดู จะยังไม่ถึงจุดสำคัญที่เป็นหัวใจของการทำงาน ค่อยๆติดตามไปครับ อยากให้ค่อยๆพิจารณาตามไป ดูการทำโค๊ด สงสัยจุดไหนในโค๊ด สามารถสอบถามเข้ามาได้ครับ

```
44
45  void setupAP(){
46
47    myId = String(EEPROM.read(NODE));
48    myId = String("000").substring(0,3-myId.length()) + myId;
49
50    ssid = SSID_PREFIX + myId;
51
52    Serial.println(); Serial.println();
53    Serial.println("   ****************************");
54    Serial.println("   *                          *");
55    Serial.println("   * Setting up mesh node...  *");
56    Serial.println("   * ssid = " + ssid + "       *");
57    Serial.println("   *                          *");
58    Serial.println("   ****************************");
59    Serial.println("   Command:");
60    Serial.println("   NODE=XXX  assign node nunmber");
61    Serial.println("   CONFIG    Setup and Show this dialog");
62    WiFi.setOutputPower(0);
63    WiFi.mode(WIFI_AP_STA);
64    WiFi.softAP(ssid.c_str());
65
66    _server.begin();
67
68  }
69
70  String temp = "";
71
```

Mesh network ตอนที่ 5 ฟังค์ชั่นใน void loop()

ใน loop() จะวนทำงานฟังค์ชั่นไปเรื่อยๆ ไม่มีที่สิ้นสุด จนกว่าจะหยุดป้อนไฟ หรือเมื่อเกิดบั๊กขึ้นในโค๊ด

บรรทัดที่ 32 ตัวแปรสตริงค์ command สำหรับเก็บตัวอักษรที่ป้อนเข้ามาจาก Serial monitor หรือผ่าน Serial port จากแอปอื่นๆ

บรรทัดที่ 33 ตัวแปรสตริงจ์ attempMsg จะนำเอา command มาตรวสอบว่าเป็นคำสั่งหรือเป็นข้อมูลแบบ JSON ถ้าเป็นคำสั่งก็จะปฏิบัติแล้วก็รอรับตัวอักษร ถ้าไม่ใช่คำสั่งก็จะทำการตรวจสอบว่าเป็น JSON ที่มีรูปแบบถูกต้องหรือไม่ และจะแยก key:value ออกมา

บรรทัดที่ 37 acceptCommand เป็นฟังค์ชั่นรับตัวอักษรจาก Serial port แปลความหมาย ปฏิบัติ
บรรทัดที่ 39 forwardMessage เป็นการนำข้อมูล attempMsg ซึ่งเป็นรูปแบบ JSON ส่งผ่านไปให้กับ device อื่นๆ หรือไม่ส่งต่อ ถ้าคำสังใน attempMsg เป็นคำสั่งที่ตัวเองต้องปฏิบัติ แล้วทำการส่งการตอบสนองกลับไปให้กับ device ตัวที่เริ่มสร้างคำสั่ง
บรรทัดที่ 41 acceptRequest เป็นฟังชั่นที่ทำการตรวจสอบว่ามี device ส่งคำร้องเข้ามาให้ server หรือไม่ ซึ่งก็คือการส่งออกมาจากฟังค์ชั่น forwardMessage จาก device อื่นนั่นเอง ฟังชั่นจะตรวจสอบคำสั่ง ปฏิบัติคำสั่ง หรือ ส่งผ่านไปยัง device ที่ถูกระบุว่าเป็นปลายทาง กรณีไม่พบก็จะส่งไปให้กับ device อื่นที่ตรวจพบ เป็นการส่งผ่านไปเรื่อยๆ จนพบ device ปลายทาง ตอนต่อไปจะให้ดูโค๊ดของแต่ละฟังชั่น

```
31
32  String command = "";
33  String attempMsg = "";
34
35  void loop() {
36
37    acceptCommand();
38
39    forwardMessage();
40
41    acceptRequest();
42
43  }
44
```

Mesh network ตอนที่ 6 ฟังค์ชั่นใน acceptCommand()

ทำหน้าที่รับตัวอักษรจาก Serial port รวบรวมดูว่าเป็นคำสั่งอะไรที่กำหนดไว้หรือไม่
บรรทัดที่ 74 - 82 เป็นการวนรับตัวอักษรที่เข้ามาที่ Serial port ด้วยการดูว่ามีตัวอักษรที่รับมาหรือไม่
บรรทัดที่ 74 ดูว่า Serial.available() มีจำนวนตัวอักษรมากกว่า 0 จะเข้าไปในลูป while
บรรทัดที่ 75 ทำการอ่านตัวอักษรมาทีละตัว ด้วยคำสั่ง Serial.read() นำค่าที่อ่านได้เก็บไว้ที่ตัวแปร c ที่เป็นตัวแปรแบบ char
บรรทัดที่ 76 ตรวจสอบ c ว่าเป็นตัวอักษร \n (new line) หรือไม่
บรรทัดที่ 77 ถ้า c เท่ากับ \n ก็จะสำเนาค่าที่ได้รวบรวมไว้ที่ตัวแปร temp มาเก็บไว้ที่ตัวแปร command
บรรทัดที่ 78 ล้างค่าตัวแปร temp

บรรทัดที่ 79 ถ้า c ไม่เท่ากับ \n ก็จะนำค่าของ c ไปต่อท้ายตัวแปร temp ในบรรทัดที่ 80
บรรทัดที่ 84 ดูว่า command ได้รับสำเนาคำสั่ง นั่นคือไม่ได้เป็นสตรงจ์ว่าง ก็จะนำไปทำคำสั่งต่อไป
บรรทัดที่ 85 นำตัวแปร command ผ่านค่าไปยังฟังค์ชั่น attempCommand (command) เพื่อดูว่าเป็นคำสั่งที่ถูกกำหนดไว้หรือไม่ ในที่นี้คือ NODE=XXX หรือ CONFIG
โดยฟังค์ชั่น attempCommand(command) จะส่งค่ากลับมาเป็นแบบ bool ได้แก่
true หมายถึง command เป็นคำสั่ง
หรือ false หมายถึง command ไม่ใช่คำสั่งแต่อาจเป็นข้อมูลที่ต้องการส่งไปยัง device อื่น
บรรทัดที่ 86 ถ้า attempCommand(command) ตอบกลับมาว่าไม่ใช่คำสั่ง ซึ่งก็อาจเป็นข้อมูลที่ต้องการส่งผ่านไปให้ device อื่นๆในรูปแบบ JSON ก็จะดำเนินการจัดเก็บสำเนา command ไว้ที่ตัวแปร attempMsg
บรรทัดที่ 88 เมื่อค่ากำหนดได้รับการเปล่ยนแปลง เช่น ถ้าคำสั่งเป็น NODE=3 คือต้องการให้ device มีหมายเลขกำกับเป็น 3 ซึ่งต้องการปรับระบบให้เป็นไปตามคำสั่ง จึงเรียกฟังค์ชั่น setupAP() เพื่อปรับระบบอีกครั้ง
บรรทัดที่ 90 ทำการล้างค่า command ป้องกันการทำงานซ้ำ
ในตอนนี้เพื่อให้ทราบถึงการรับข้อมูลจาก Serial port รวบรวมตัวอักษร เมื่อพบว่ามีตัวอักษรที่กำหนดว่าเป็นการจบคำสั่ง ก็จะนำค่าที่ได้ไปตีความ
ตอนต่อไปจะดูว่าฟังค์ชั่น attempCommand ทำอะไร และถ้าไม่ยาวมากนักก็จะพูดถึง attempMsg ด้วย
เพิ่มเติมอักขระพิเศษ อาจเป็น
\n = new line
\r = return
\t = tab

เมื่อทำการทดสอบโปรแกรม ต้องทำการตั้งค่าให้ Serial monitor ให้เพิ่ม \n ต่อท้ายให้กับเรามี่อกดปุ่ม send หรือเคาะแป้น Enter ด้วยการเลือก Newline ด้านล่างขวาของหน้าจอ Serial monitor

https://www.facebook.com/permalink.php?story_fbid=1071749996194165&id=1043638695671962

```
70  String temp = "";
71
72  void acceptCommand(){
73
74    while(Serial.available()>0){
75      char c = Serial.read();
76      if(c == '\n'){
77        command = temp;
78        temp = "";
79      } else {
80        temp += c;
81      }
82    }
83
84    if(command != ""){
85      if(!attempCommand(command)){
86        attempMsg = command;
87      }else{
88        setupAP();
89      }
90      command = "";
91    }
92
93  }
94
```

Mesh network ตอนที่ 7 ฟังค์ชั่นใน attempCommand()

ในฟังค์ชั่น acceptCommand()

บรรทัดที่ 86 จะนำ command ไปตีความ ถ้าเป็นคำสั่งก็จะปฏิบัติและส่งค่ากลับมาเป็น true

บรรทัดที่ 89 จะทำฟังค์ชั่น setupAP เพื่อเปลี่ยนค่าต่างๆ อาจเป็นหมายเลข device

บรรทัดที่ 87 ถ้า attempCommand() ส่งค่ากลับเป็น false ก็จะเก็บสตริงค์ command ไว้ที่ attempMsg เพื่อนำไปใช้ในฟังค์ชั่น forwardMessage()

บรรทัดที่ 91 จะล้างสตริงจ์ command ป้องกันไม่ให้ทำงานซ้ำ

ในฟังค์ชั่น attempCommand(String command)

เป็นการตีความ command ที่ป้อนมาเทียบกับคำสั่งที่กำหนดไว้ ในที่นี้เรามีสองคำสั่งคือ NODE และ CONFIG

บรรทัดที่ 97 - 103 ทำการเปลี่ยนหมายเลข device ที่ต่อท้ายคำสั่ง NODE=

บรรทัดที่ 97 เป็นการตรวสอบว่าสตริงจ์ที่ป้อนมาประกอบด้วยคำว่า NODE= หรือไม่ ด้วยการใช้คำสั่ง indexOf ผลของการตรวจสอบจะบอกตำแหน่งเริ่มต้นที่พบ ถ้าไม่พบจะส่งค่า -1 กลับมา
บรรทัดที่ 99 ทำการตัดคำว่า NODE= ออกจากสตริงจ์ command จะได้ตัวเลขที่ส่งเข้ามา เช่น ถ้า command คือ NODE=3 เมื่อทำคำสั่ง replace("NODE=","") จะทำให้ command เหลือเพียง 3 เนื่องจาก NODE= ถูกแทนที่ด้วยสตริงจ์ว่าง
บรรทัดที่ 100 แปลงค่าตัวอักษรเป็นตัวเลข เพื่อเก็บค่าหมายเลข device ใน EEPROM
บรรทัดที่ 101 บันทึกค่า node ที่ตำแหน่ง NODE หรือตำแหน่ง 0
บรรทัดที่ 102 EEPROM.commit(); เป็นการยืนยันการบันทึก ถ้าไม่ทำคำสั่งนี้ คำสั่ง EEPROM.write จะไม่มีผลใดๆ
บรรทัดที่ 103 ส่งค่าคืนเป็น true เนื่องจากเป็นคำสั่ง
บรรทัดที่ 105 - 107 เป็นคำสั่งที่ให้แสดงหน้าจอหลัก เนื่องจากเป็นคำสั่ง จึงส่งค่า true กลับ
บรรทัดที่ 111 เป็นการส่งค่า false กลับ เนื่องจาก command ไม่เป็นหนึ่งในคำสั่ง ซึ่งอาจเป็นข้อมูลที่ต้องการส่งไปที่ device อื่นๆ

คำสั่งที่แสดงสองคำสั่งเป็นตัวอย่าง การเพิ่มคำสั่งอื่น ก็ให้กำหนดคำสั่ง แล้วทำการตรวจสอบด้วย indexOf คิดว่าสะดวก และถ้ามีพารามิเตอร์ด้วยเหมือนกับคำสั่ง NODE=3 ก็ใช้วิธี replace ตัดคำสั่งออกก็จะเหลือค่าที่จะนำไปใช้งาน

```
73  void acceptCommand(){
74
75    while(Serial.available()>0){
76      char c = Serial.read();
77      if(c == '\n'){
78        command = temp;
79        temp = "";
80      } else {
81        temp += c;
82      }
83    }
84
85    if(command != ""){
86      if(!attempCommand(command)){
87        attempMsg = command;
88      }else{
89        setupAP();
90      }
91      command = "";
92    }
93
94  }
```

```
96  bool attempCommand(String command){
97    if(command.indexOf("NODE=")>-1){
98
99      command.replace("NODE=","");
100     int node = command.toInt();
101     EEPROM.write(NODE, node);
102     EEPROM.commit();
103     return true;
104
105   }else if(command.indexOf("CONFIG")>-1){
106
107     return true;
108
109   }else{
110
111     return false;
112   }
113 }
114
```