

แนวทางการใช้งานอินเทอร์เน็ตของสรรพสิ่งในระบบการผลิต

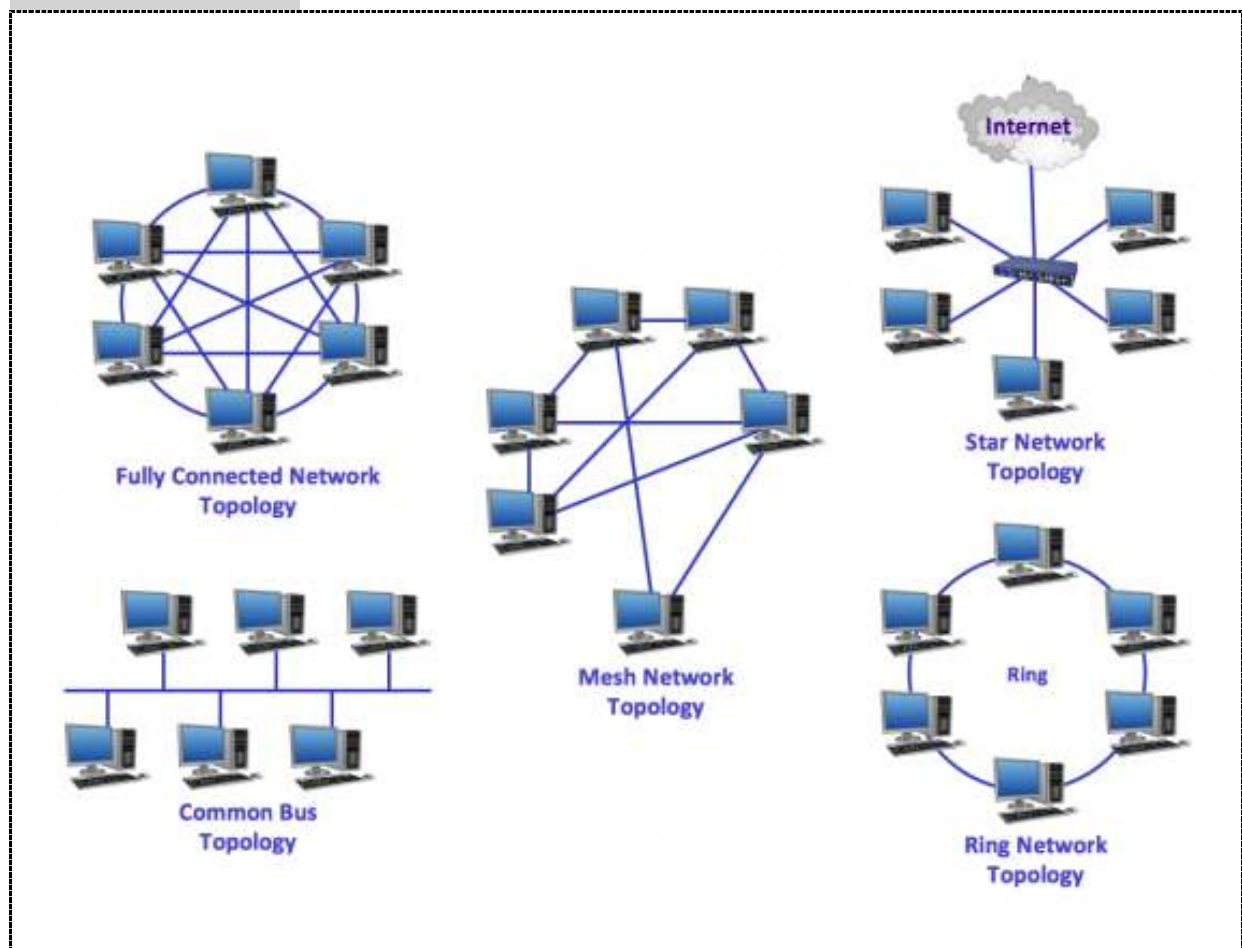
IoT Approaches to Manufacturing System

2/4 - Industrial 4.0 กับ IoTs และ IIoTs, ก้าวสู่ยุคของ IoB

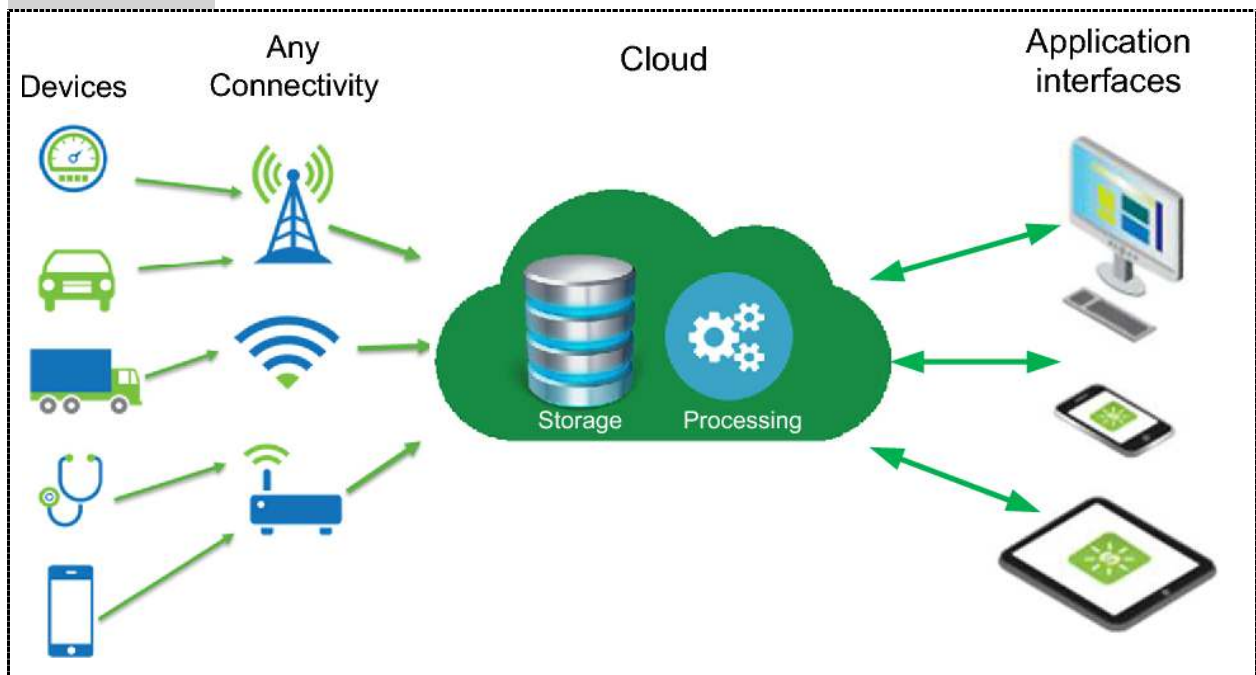
- บทนำ
- การโปรแกรมใช้งานแบบ All Over IP
- การโปรแกรมใช้งานแบบ Internet of Things
- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

1/4 บทนำ

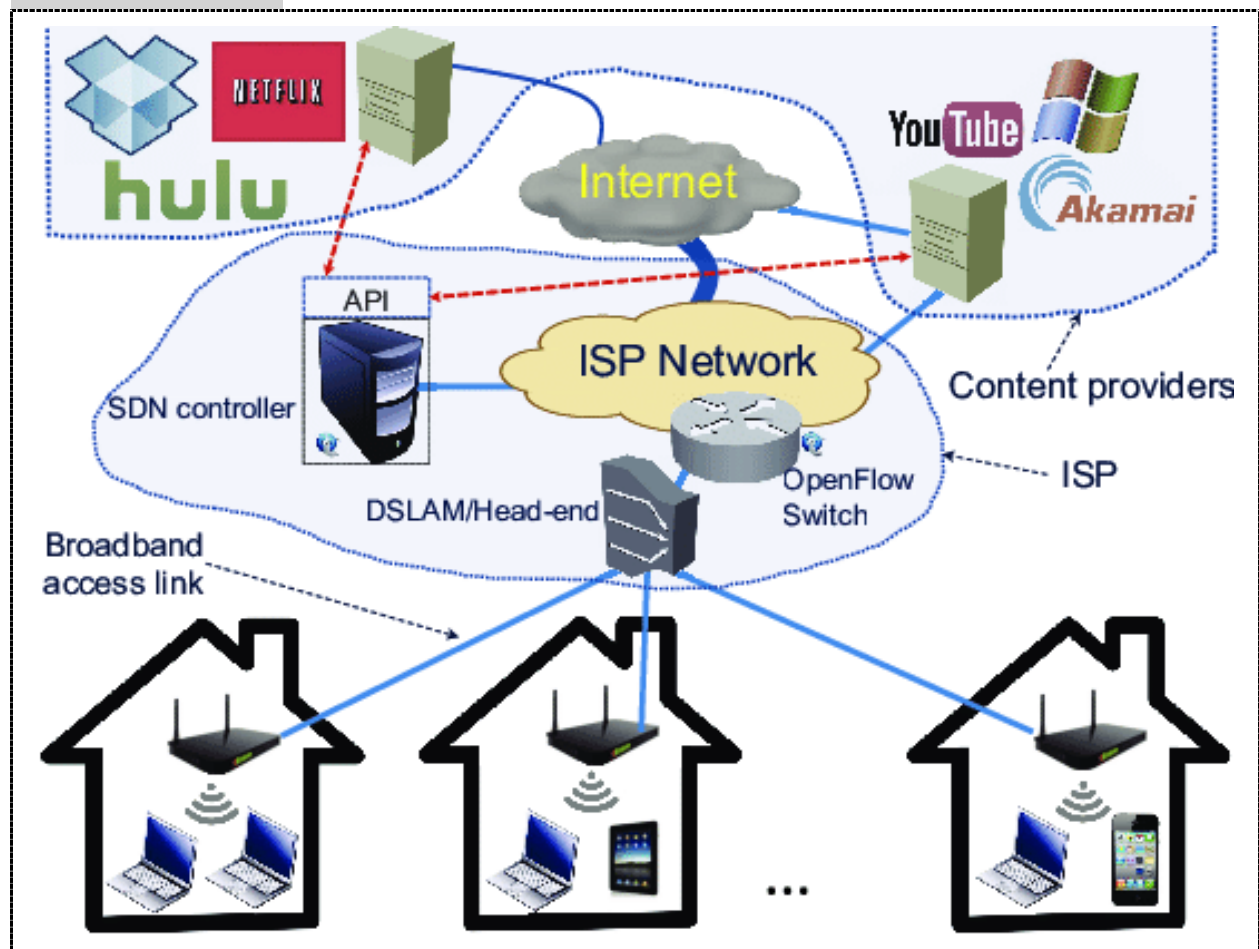
1.1 Network Topology



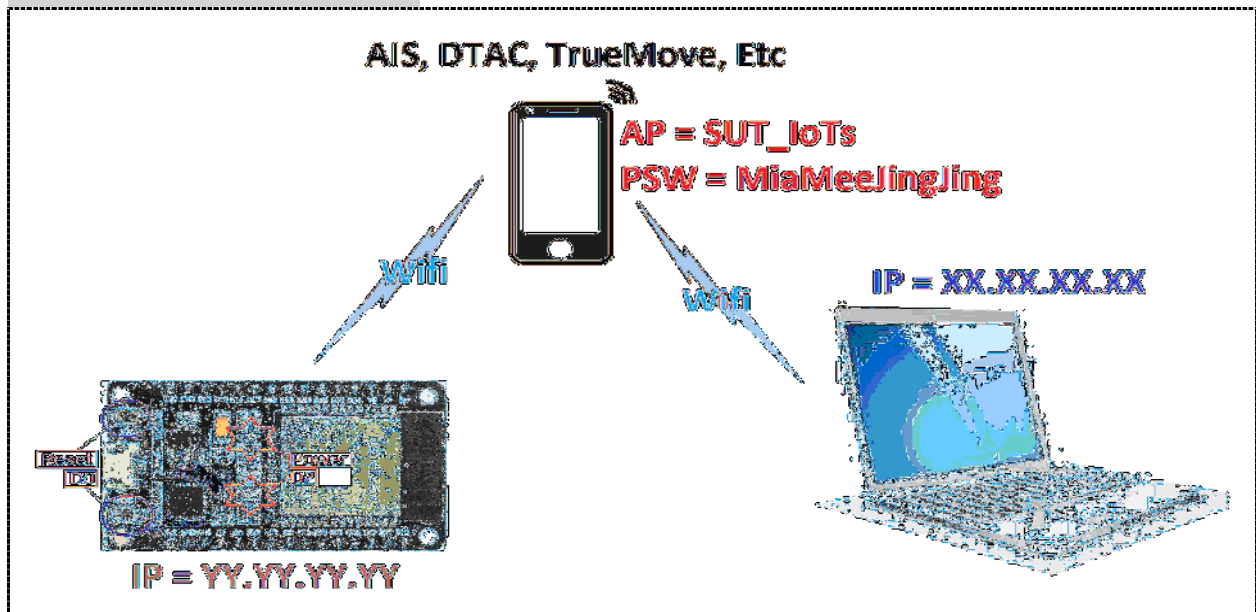
1.2 IoT Network



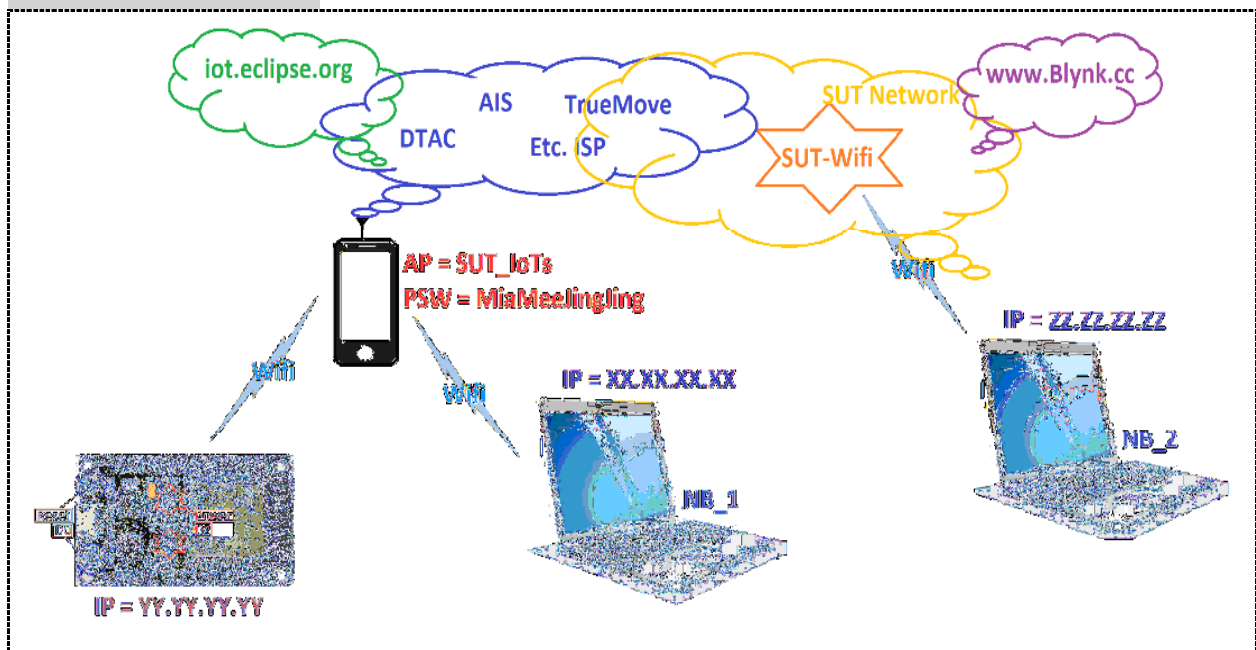
1.3 Internet Network



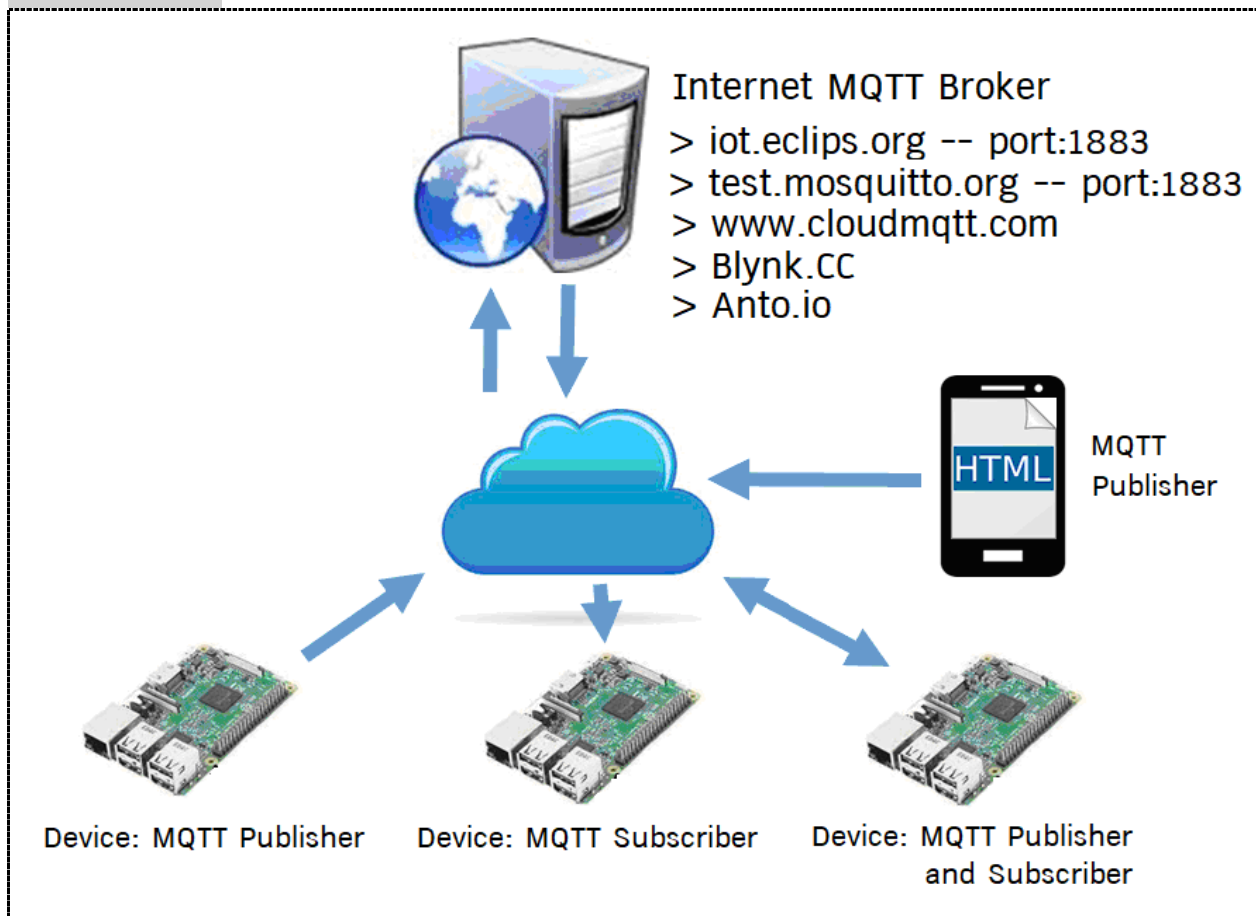
1.4 Test Network1 – All Over IP



1.5 Test Network2 – IoT

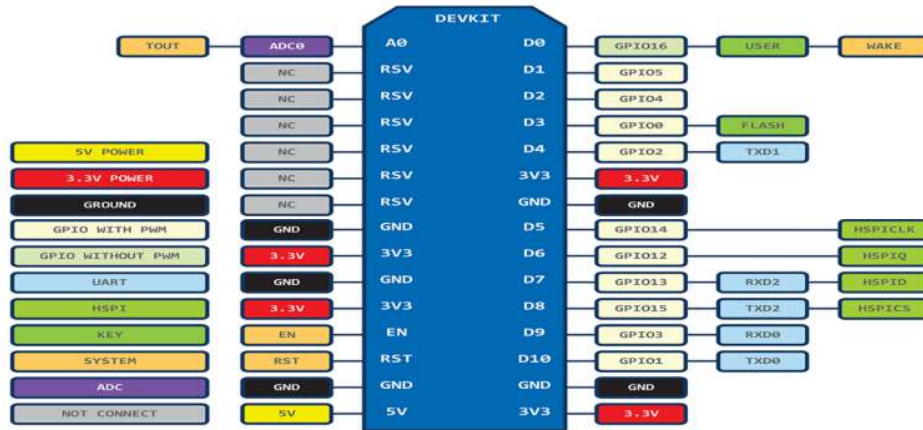


1.6 IoT Concept



1.7 ESP-Board

1.7.1 NodeMCU - ESP8266



NodeMCU V0.9

ESP-12 Module

Arduino IDE = Node0.9



- USB-SERIAL CH340
- ใช้ Serial LED ที่ GPIO1 ได้ แต่ต้องไม่ใช้พร้อม Serial Communication
- มี LED Buid in ที่ GPIO16“BUILTIN_LED”

NodeMCU V1.0

ESP-12E Module

Arduino IDE = Node1.0



- Silicon Labs CP210x USB to UART Bridge
- ใช้ Serial LED ที่ GPIO2 ได้ แต่ต้องไม่ใช้พร้อม Serial Communication
- มี LED Buid in ที่ GPIO16“BUILTIN_LED”

NodeMCU V3.0

ESP-12E Module

Arduino IDE = Node1.0

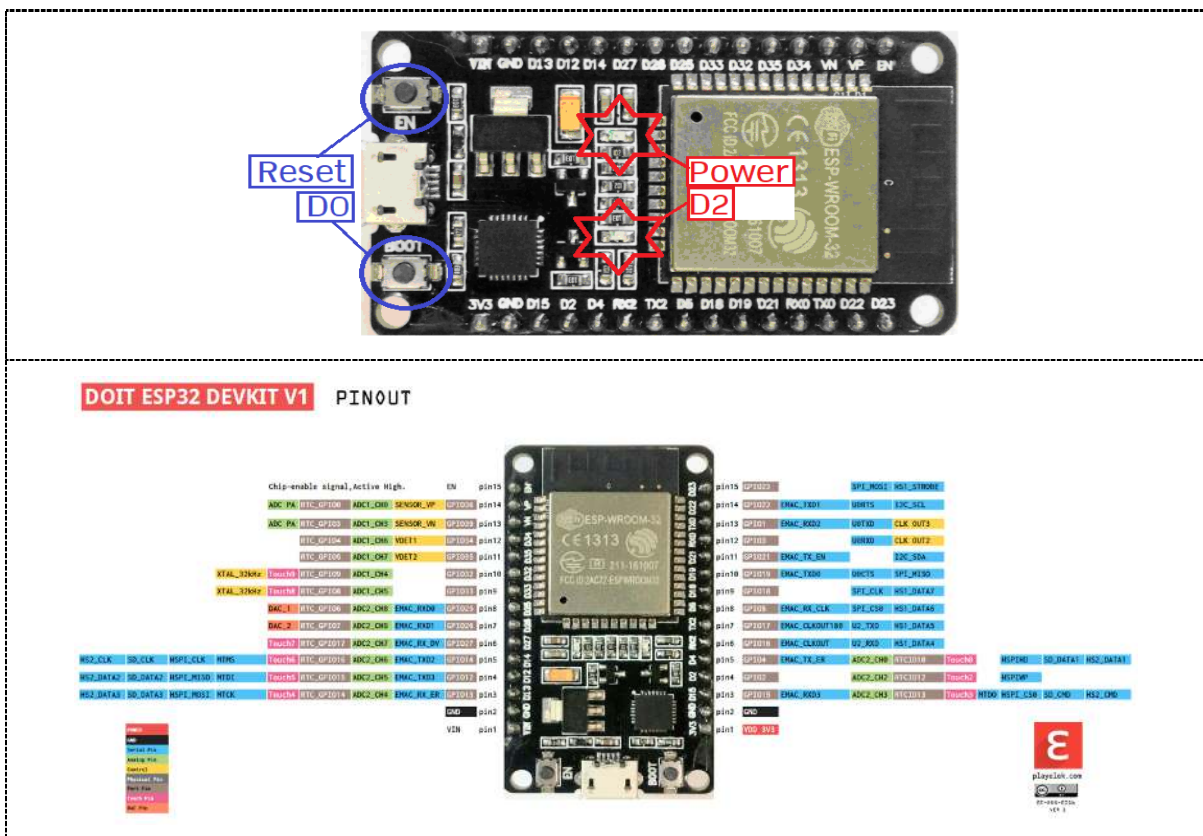


- USB-SERIAL CH340
- ใช้ Serial LED ที่ GPIO2 ได้ แต่ต้องไม่ใช้พร้อม Serial Communication

1.7.2 ESP-32 Dev Kit V1 Board

ก่อนหน้านี้มีการใช้งาน NodeMCU V2 ซึ่งเป็น ESP8266 อย่างแพร่หลาย แต่ด้วยเสียงลือเสียงเล่าอ้างเรื่องความสามารถของ ESP32 ที่พัฒนาความสามารถเพิ่มมาแก้จุดด้อยของ ESP8266 ทั้งรองรับการเชื่อมต่อแบบ Hybrid ทั้ง WiFi และ Bluetooth มีพอร์ตรองรับ I/O ได้เพิ่มขึ้น รองรับ touch sensor มี hardware เข้ารหัสสำหรับ HTTPS และอีกมากมาย ด้วยเหตุผลที่ว่าไปแล้วและราคาที่ไม่แพงรอบนี้เลยได้ ESP32 Development Board ที่มีชื่อเต็มคือ DOIT ESP32 DevKit V1 ใช้โมดูล ESP-WROOM-32 มาทำการทดสอบ

รายละเอียดเพิ่มเติมของ DOIT ESP32 DevKit V1 ลองเข้าไปดูใน SmartArduino (<https://github.com/SmartArduino/SZDOITWiki/wiki/ESP8266---ESP32>) หน้าตาคล้าย ESP32 DevKit C V2 ของ Espressif และ Development Board ตระกูลเดียวกันกับเจ้าอื่นๆเลย มีเจาะรูสี่มุมมาด้วย แต่ pinout ไม่เหมือนกัน



DOIT ESP32 DEVKIT V1 PINOUT

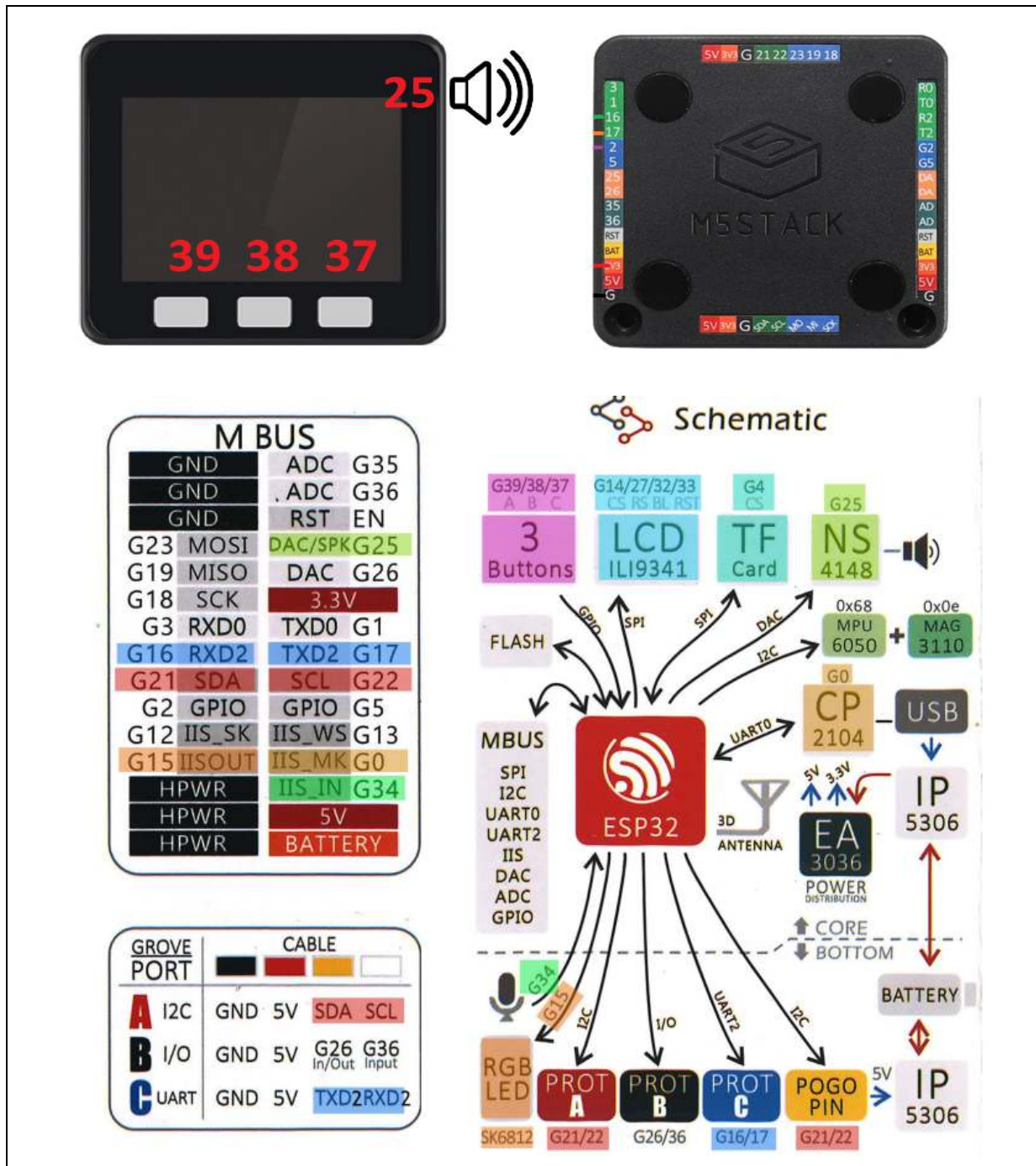
Chip-enable signal, Active High.				EN	pin15
ADC_PA	RTC_GPIO0	ADC1_CH0	SENSOR_VP	GPI036	pin14
ADC_PA	RTC_GPIO3	ADC1_CH3	SENSOR_VN	GPI039	pin13
	RTC_GPIO4	ADC1_CH6	VDET1	GPI034	pin12
	RTC_GPIO5	ADC1_CH7	VDET2	GPI035	pin11
XTAL_32kHz	Touch9	RTC_GPIO9	ADC1_CH4	GPI032	pin10
XTAL_32kHz	Touch8	RTC_GPIO8	ADC1_CH5	GPI033	pin9
DAC_1	RTC_GPIO6	ADC2_CH8	EMAC_RXD0	GPI025	pin8
DAC_2	RTC_GPIO7	ADC2_CH9	EMAC_RXD1	GPI026	pin7
Touch7	RTC_GPIO17	ADC2_CH7	EMAC_RX_DV	GPI027	pin6
HS2_CLK	SD_CLK	HSPI_CLK	MTMS	Touch6	RTC_GPIO16
HS2_DATA2	SD_DATA2	HSPI_MISO	MTDI	Touch5	RTC_GPIO15
HS2_DATA3	SD_DATA3	HSPI_MOSI	MTCK	Touch4	RTC_GPIO14
					pin5
					pin4
					pin3
					pin2
					pin1



pin15	GPI023	SPI_MOSI	HS1_STROBE
pin14	GPI022	EMAC_TXD1	UART5
pin13	GPI01	EMAC_RXD2	UART4
pin12	GPI03	UART3	CLK_OUT3
pin11	GPI021	EMAC_TX_EN	CLK_OUT2
pin10	GPI019	EMAC_TXD0	I2C_SDA
pin9	GPI018	EMAC_TXD0	I2C_SCL
pin8	GPI05	EMAC_RX_CLK	SPI_MISO
pin7	GPI017	EMAC_CLKOUT180	SPI_CS0
pin6	GPI016	EMAC_CLKOUT	HS1_DATA7
pin5	GPI04	EMAC_TX_ER	HS1_DATA6
pin4	GPI02	ADC2_CH0	HS1_DATA5
pin3	GPI015	EMAC_RXD3	HS1_DATA4
pin2	GND	ADC2_CH2	U2_TXD
pin1	VDD 3V3	ADC2_CH3	U2_RXD



1.7.3 M5-Stack

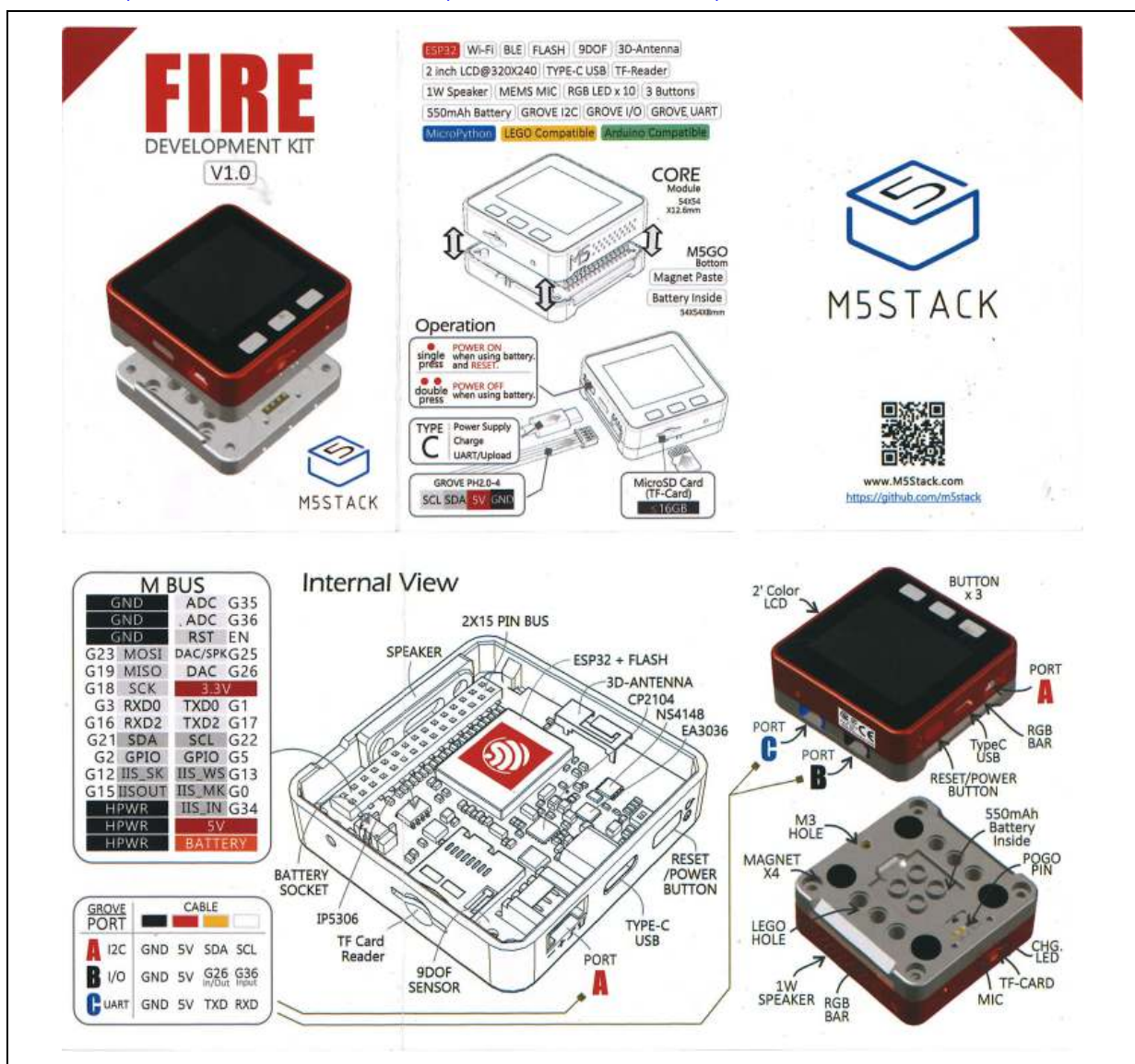


5Stack ESP32 คอมพิวเตอร์จิ๋วที่ใช้สร้างงานต้นแบบที่มาพร้อมกับหน้าจอ 2 นิ้ว ความละเอียด 320x240 pixel และมีหน้ากากให้เปลี่ยนแป้นพิมพ์ได้ 3 แบบ ตามการใช้งานที่ออกแบบ Keyboard panel, Gameboy panel และ Number Panel พร้อมกับฐานชาร์ตและแบตเตอรี่ LiPo ขนาด 650mAh

M5Stack คืออุปกรณ์ที่ใช้สร้างตัวต้นแบบของอุปกรณ์ WiFi+Bluetooth โดยใช้ชิป ESP32 ของ Espressif โปรแกรมได้ทั้ง Arduino, Micro-Python หรือ Web-IDE ตัวบอร์ดหลัก M5 Core จะอยู่ด้านบนติดกับจอ LCD จะมีเสาอากาศแบบ 3D มาให้ (3D Antenna), Grove connector สำหรับ I2C, microSD card socket, JST battery socket, สวิตช์ เปิด/ปิด/reset, 3ปุ่มกดบนหน้าจอ

บอร์ดด้านหลังจะเป็น M5 Faces ซึ่งทำหน้าที่ต่อเข้ากับแป้นพิมพ์ Panel ต่างๆ และยังมีแบตเตอรี่ LiPo ขนาด 650mAh ที่สามารถชาร์จผ่าน charging Base ได้

- <https://github.com/m5stack/M5Stack>
- <http://forum.m5stack.com/topic/360/m5stack-fire-pinout-leaflet>



1.8 อ่านเพิ่มเติม

- NodeMCU GitHub: [https:// github.com/nodemcu](https://github.com/nodemcu)
- NodeMCU Driver: [https://
www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx](https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx)
- ThaiEasyElect: [http:// www.thaieasyelec.com/products/internet-of-things/nodemcu-development-kit-v2-detail.html?gclid=Cj0KEQjwl-e4BRCwqeWkv8TWqOoBEiQAMocbPytjm40atWOSYlaQI7Vo0Op-7asSWryeJ9tCQJNxnpoaAk2-8P8HAQ](http://www.thaieasyelec.com/products/internet-of-things/nodemcu-development-kit-v2-detail.html?gclid=Cj0KEQjwl-e4BRCwqeWkv8TWqOoBEiQAMocbPytjm40atWOSYlaQI7Vo0Op-7asSWryeJ9tCQJNxnpoaAk2-8P8HAQ)
- AiyaraFun: [http:// www.ayarafun.com/2015/08/introduction-arduino-esp8266-nodemcu/](http://www.ayarafun.com/2015/08/introduction-arduino-esp8266-nodemcu/)
- Firmware Build and Example: [http:// nodemcu-build.com/](http://nodemcu-build.com/)
- Read This <https://playelek.com/doit-esp32-devkit-v1/>
- Read This <http://esp32.net/>
- Read This <https://www.arduitronics.com/product/1329/doit-esp32-development-board-esp-wroom-32-wifiblueetooth-esp-32s>
- Read This <https://www.mcucity.com/product/1144/doit-esp32-wifiblueetooth-ultra-low-power-consumption-dual-core-esp-32-esp-32s-esp-32-similar-esp8266>

2/4 การโปรแกรมใช้งานแบบ All Over IP

การทดลอง 1: Start with Arduino IDE in Hello World

1. Install Arduino IDE and Add ESP32 Board
2. Test Ex00_Blink

```
// ESP-32  
  
char DispBuff[] = {1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0};  
  
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  int nloop = sizeof(DispBuff);  
  for (int i = 0; i < nloop; i++)  
  { digitalWrite(LED_BUILTIN, DispBuff[i]); delay(120);  
    digitalWrite(LED_BUILTIN, LOW); delay(120);  
  }  
}
```

การทดลอง 2: My MAC Address

3. โหลดโปรแกรม My MAC = _____

```
uint64_t chipid;

void setup() {
  Serial.begin(115200);
}

void loop() {
  chipid = ESP.getEfuseMac(); //The chip ID is essentially its MAC address(length: 6 bytes).
  Serial.printf("ESP32 Chip ID = ");
  Serial.printf("%02X:", (uint8_t)(chipid >> 0)); //print 1 bytes
  Serial.printf("%02X:", (uint8_t)(chipid >> 8)); //print 1 bytes.
  Serial.printf("%02X:", (uint8_t)(chipid >> 16)); //print 1 bytes.
  Serial.printf("%02X:", (uint8_t)(chipid >> 24)); //print 1 bytes.
  Serial.printf("%02X:", (uint8_t)(chipid >> 32)); //print 1 bytes.
  Serial.printf("%02X", (uint8_t)(chipid >> 40)); //print 1 bytes.
  Serial.println();
  delay(3000);
}
```

4. ทดสอบ File → Eample → WiFi → WifiScan หรือโปรแกรมต่อไปนี้

```
#include "WiFi.h"

void setup()
{
  Serial.begin(115200);

  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  Serial.println("Setup done");
}

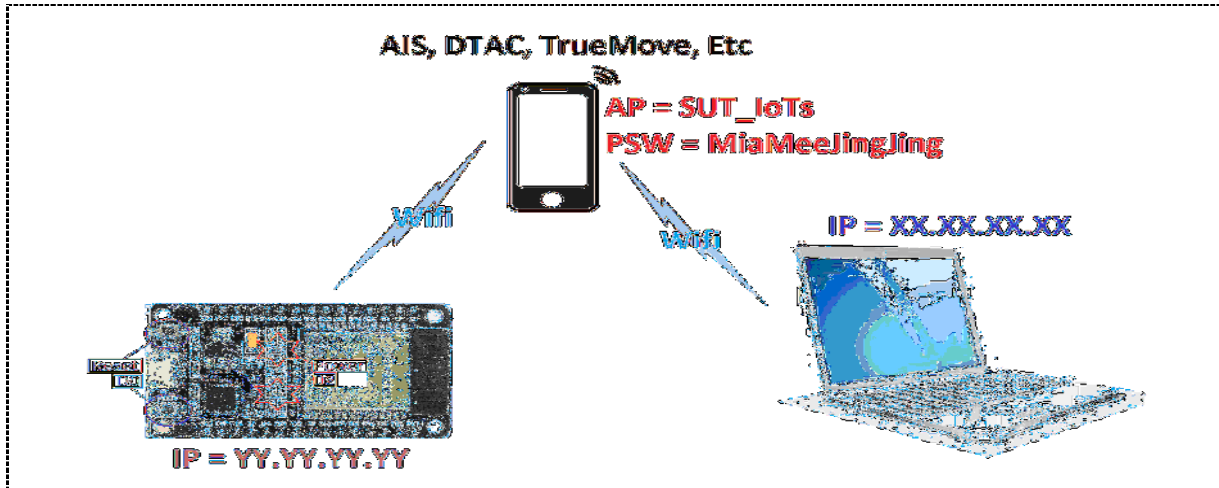
void loop()
{
  Serial.println("scan start");

  // WiFi.scanNetworks will return the number of networks found
  int 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) == WIFI_AUTH_OPEN)? " ":"*");
      delay(10);
    }
  }
  Serial.println("");

  // Wait a bit before scanning again
  delay(5000);
}
```


การทดลอง 3: Connect to Network

5. โหลดโปรแกรมและ อย่าลืมแก้เป็นชื่อ SSID และ Password เป็นของตัวเอง



6. ทดสอบการเชื่อมต่อด้วยคำสั่ง Ping ในหน้าต่าง command
7. เมื่อโหลดโปรแกรมแล้วให้ตรวจสอบว่า ESP32 ต่อ Access Point ได้ หากได้จะแสดง IP เช่น 192.168.43.87
8. เปิดคอมมานด้วย Search → command
9. พิมพ์คำสั่งทดสอบการต่อกับ www.google.com ด้วยคำสั่ง **ping 8.8.8.8**
10. ทดสอบการต่อกับ ESP 32 บอร์ดด้วยคำสั่ง ping IP **ping 192.168.43.87**

```
#include <WiFi.h>
const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
void setup() {
  Serial.begin(115200);          delay(10);
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  { delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {}
```

The screenshot shows a serial monitor window titled 'COM3'. The output text is as follows:

```
load:0x3fff0018,len:4
load:0x3fff001c,len:808
load:0x40078000,len:6084
load:0x40080000,len:6696
entry 0x400802e4

Connecting to SUT_IoT
..
WiFi connected
IP address:
192.168.43.87
```

At the bottom of the window, there are settings: 'Autoscroll' is checked, 'No line ending' is selected, '115200 baud' is selected, and a 'Clear output' button is present.

The screenshot shows a Windows Command Prompt window titled 'Select Command Prompt'. The output of the command 'ping 192.168.43.87' is as follows:

```
Microsoft Windows [Version 10.0.16299.547]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Pk007.Bit32>ping 192.168.43.87

Pinging 192.168.43.87 with 32 bytes of data:
Reply from 192.168.43.87: bytes=32 time=780ms TTL=255
Reply from 192.168.43.87: bytes=32 time=377ms TTL=255
Reply from 192.168.43.87: bytes=32 time=386ms TTL=255
Reply from 192.168.43.87: bytes=32 time=395ms TTL=255

Ping statistics for 192.168.43.87:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 377ms, Maximum = 780ms, Average = 484ms

C:\Users\Pk007.Bit32>
```

การทดลอง 4: Web Server-Command

11. Test Ex20_WebServer_Cmd

```

#include <WiFi.h>

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
int pinTest = 2;

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  pinMode(pinTest, OUTPUT);          // set the LED pin mode
  delay(10);
  Serial.print("\n\nConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();
}

int value = 0;
bool LED_Status = LOW;

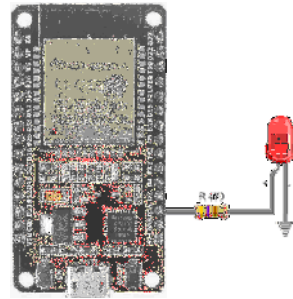
void loop() {
  digitalWrite(pinTest, LED_Status);
  WiFiClient client = server.available();   // listen for incoming clients

  if (client) {                             // if you get a client,
    Serial.println("New Client.");           // print a message out the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected()) {            // loop while the client's connected
      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
        if (c == '\n') {                    // if the byte is a newline character
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.println("<html>");
            client.println("<body>");
            client.println("<h1>LED Status</h1>");
            client.println("<p>");
            if (LED_Status == HIGH)
              client.println("LED On");
            else
              client.println("LED Off");
            client.println("<p>");
            client.println("<body>");
            client.println("<html>");
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
        if (currentLine.endsWith("GET /ledon")) LED_Status = HIGH;
        if (currentLine.endsWith("GET /ledoff")) LED_Status = LOW;
      }
    }
    client.stop(); // close the connection:
    Serial.println("Client Disconnected.");
  }
}

```

การทดลองนี้เป็นการนำเอา ESP-32 มาสร้างเป็น Web Server โดยเมื่อมีการร้องขอหน้าเว็บไซต์มาเป็นตัวกำหนดให้หลอด LED ติดดับ

ต่อวงจร โหลดโปรแกรมและ อย่าลืมแก้เป็นชื่อ SSID และ Password เป็นของตัวเอง



เปิด Web Browser แล้วกำหนด url ไปที่ IP ของ ESP-32 → **YY.YY.YY.YY**



เรียกหน้าเว็บไปที่ **YY.YY.YY.YY/ledon** สังเกต >> หลอด LED จะติด



เรียกหน้าเว็บไปที่ **YY.YY.YY.YY/ledoff** สังเกต >> หลอด LED จะดับ



การทดลอง 5: Web Server-Button

12. Test Ex21_WebServer_Button

```

#include <WiFi.h>

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
int pinTest = 2;

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  pinMode(pinTest, OUTPUT); // set the LED pin mode
  delay(10);
  Serial.print("\n\nConnecting to "); Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected."); Serial.println("IP address: ");
  Serial.println(WiFi.localIP()); server.begin();
}

int value = 0;
bool LED_Status = LOW;

void loop() {
  digitalWrite(pinTest, LED_Status);
  WiFiClient client = server.available(); // listen for incoming clients

  if (client) { // if you get a client,
    Serial.println("New Client."); // print a message out the serial port
    String currentLine = ""; // make a String to hold incoming data from the client
    while (client.connected()) { // loop while the client's connected
      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
        if (c == '\n') { // if the byte is a newline character
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.println("<html>");
            client.println("<body>");
            client.println("<h1>LED Status</h1>");

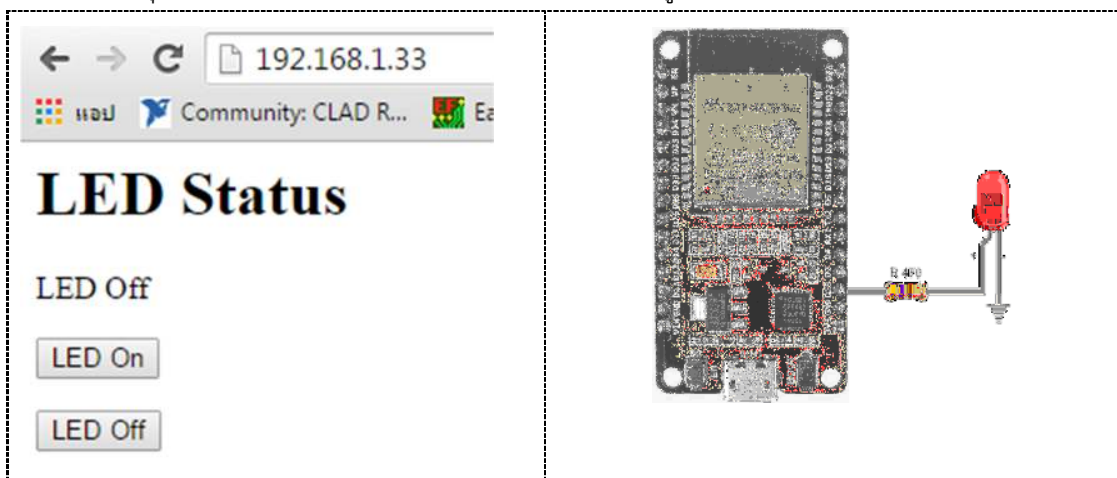
            client.println("<p>");
            if (LED_Status == HIGH)
              client.println("LED On");
            else
              client.println("LED Off");

            client.println("<p>");
            client.println("<a href='\"/ledon\"'><button>LED On</button></a>");
            // client.println("<a href='\"/ledon\"'><button style = \"background-color: #f44336;\">LED On</button></a>");
            client.println("</p>");
            client.println("<a href='\"/ledoff\"'><button>LED Off</button></a>");
            // client.println("<a href='\"/ledoff\"'><button style = \"background-color: #008CBA;\">LED Off</button></a>");

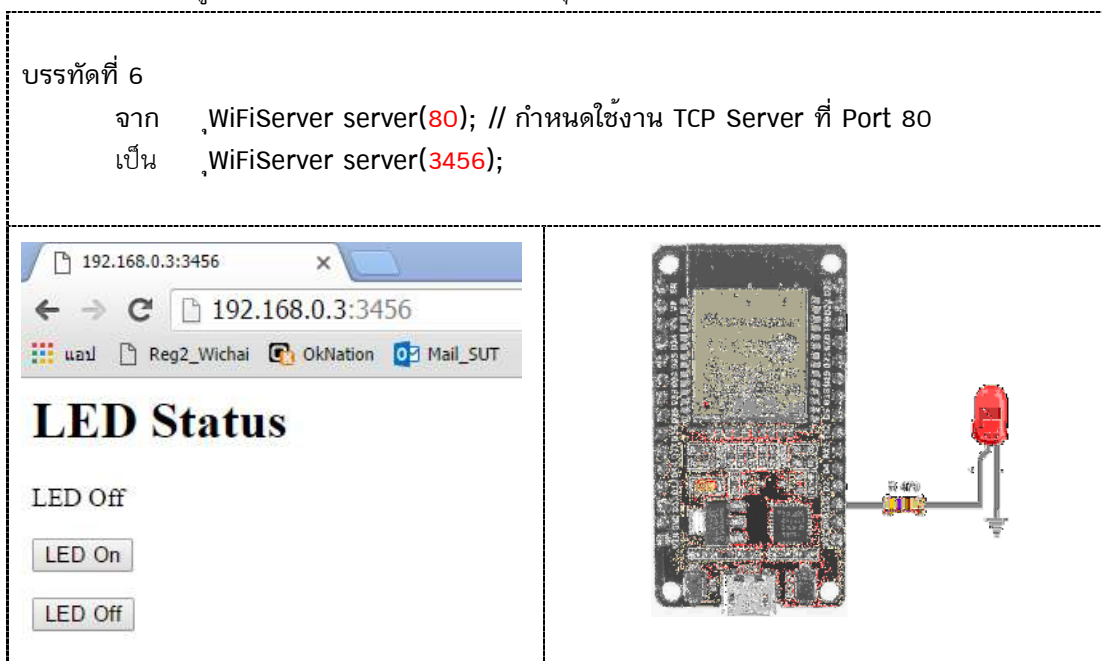
            client.println("<body>");
            client.println("<html>");
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
      }
      if (currentLine.endsWith("GET /ledon")) LED_Status = HIGH;
      if (currentLine.endsWith("GET /ledoff")) LED_Status = LOW;
    }
    client.stop(); // close the connection:
    Serial.println("Client Disconnected.");
  }
}

```

- ควบคุมการทำงานของ LED D1 โดยต่อวงจรตามรูป



- ลองปรับแก้จากพอร์ต 80 เป็นพอร์ตอื่น เช่น 3456
- การเรียกดูผ่าน Web Browser ก็ต้องระบุพอร์ตด้วย เช่น **192.168.0.3:3456**



Quiz_201 – Web Control 2 LED

- อยากได้ปุ่มสำหรับคุมปิด-เปิด หลอดไฟ LED 2 ดวง
- https://www.colorhexa.com/008cba?fbclid=IwAR3dIZ_gRgDWmREmnzucknLbMxV3pOHY4YIPuLEz8-ZzTOX2VhWxcH2QjLGk



การทดลอง 6: Web Server-Digital Read

13. Test Ex30_WebServer_DigitalRead

```

#include <WiFi.h>
#define pinTest 2
#define SW_Test 4

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  pinMode(SW_Test, INPUT_PULLUP);
  pinMode(pinTest, OUTPUT);
  delay(10);
  Serial.print("\n\nConnecting to "); Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected."); Serial.println("IP address: ");
  Serial.println(WiFi.localIP()); server.begin();
}

int value = 0;
bool LED_Status = LOW;

void loop() {
  digitalWrite(pinTest, LED_Status);
  WiFiClient client = server.available(); // listen for incoming clients

  if (client) { // if you get a client,
    Serial.println("New Client."); // print a message out the serial port
    String currentLine = ""; // make a String to hold incoming data from the client
    while (client.connected()) { // loop while the client's connected
      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
        if (c == '\n') { // if the byte is a newline character
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.println("<html>");
            client.println("<body>");
            client.println("<h1>LED Status</h1>");

            client.println("<p>");
            if (LED_Status == HIGH)
              client.println("LED On");
            else
              client.println("LED Off");

            client.println("<p>");
            client.println("<a href='\"/ledon\"'><button>LED On</button></a>");
            client.println("<a href='\"/ledoff\"'><button>LED Off</button></a>");
            client.println("</p>");

            client.println("<h1>Read Switch</h1>");
            client.println("<style>");
            client.println(".circle-green,.circle-red");
            client.println("{width: 100px; height: 100px; border-radius: 50%;}");
            client.println(".circle-green {background-color: green;}");
            client.println(".circle-red {background-color: red;}");
            client.println("</style>");

            client.println("<meta http-equiv='\"refresh\"' content='\"1\"'>");
            client.println("<p>");
            if (digitalRead(SW_Test) == HIGH)
              { client.println("<div class='\"circle-red\"'></div>");
                client.println("<p>SW = 1</p>");
              }
            else
              { client.println("<div class='\"circle-green\"'></div>");
                client.println("<p>SW = 0</p>");
              }
            client.println("</p>");
            client.println("<body>");
            client.println("<html>");
            break;

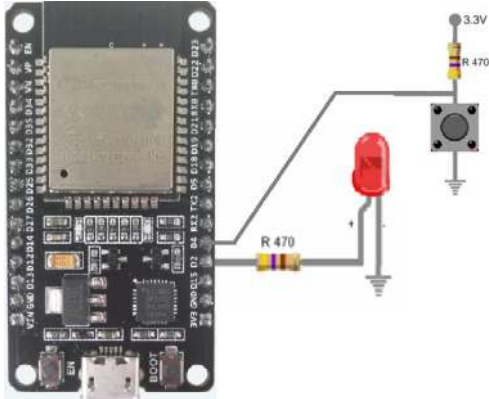

```

```



    } else {
        currentLine = "";
    }
    } else if (c != '\r') {
        currentLine += c;
    }
    if (currentLine.endsWith("GET /ledon")) LED_Status = HIGH;
    if (currentLine.endsWith("GET /ledoff")) LED_Status = LOW;
    }
}
client.stop(); // close the connection:
Serial.println("Client Disconnected.");
}
}

```

- วงจรตามรูป

	<h2>LED Status</h2> <p>LED On</p> <div>LED On LED Off</div> <h2>Read Switch</h2>  <p>SW = 1</p>
--	---

- เปิด Web Browser แล้วเรียกหน้า Page ไปยัง IP ของ Node MCU

<ul style="list-style-type: none"> • เมื่อไม่กด Switch <h2>Read Switch</h2>  <p>SW = 1</p>	<ul style="list-style-type: none"> • ทดลองกด Switch <h2>Read Switch</h2>  <p>SW = 0</p>
--	--

การทดลอง 7: Web Server-Sensor

14. Test Ex40_DHT22 Sensor ทดสอบโปรแกรมนี้

- ให้แน่ใจว่าใช้ DHT-22 library ของ [beegee_tokyo Ver 1.17.0](#)

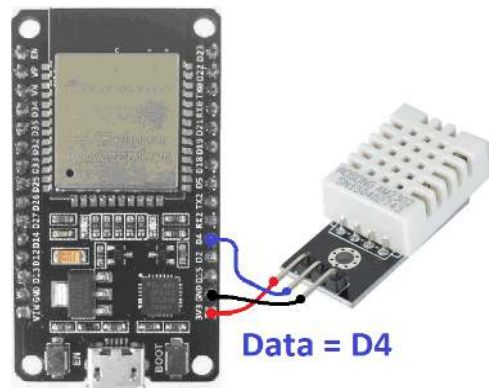
DHT sensor library for ESPx
 by beegee_tokyo Version 1.17.0 **INSTALLED**
 Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors Optimized lib
 changes: Reduce CPU usage and add decimal part for DHT11
[More info](#)

- DHT-22 Test Code

```
#include "DHTesp.h"
DHTesp dht;

void setup()
{ Serial.begin(115200);
  Serial.println();
  Serial.println("Status\tTemperature
(C)\tHumidity (%)");
  dht.setup(4, DHTesp::DHT22); // DHT_Pin D4,
  DHT22
}

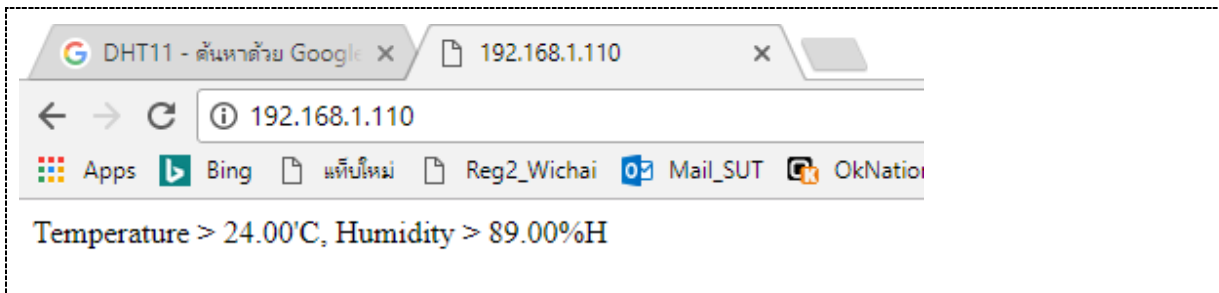
void loop()
{ delay(2000);
  float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();
  Serial.println();
  Serial.print(dht.getStatusString());
  Serial.print("\t");
  Serial.print(temperature, 1);
  Serial.print("\t\t");
  Serial.print(humidity, 1);
}
```



Status	Temperature (C)	Humidity (%)
OK	27.6	46.2
OK	27.6	46.3
OK	27.6	46.3
OK	27.5	45.9
OK	27.5	45.4
OK	27.5	45.1
OK	27.5	45.2
OK	27.5	45.4

15. Test Ex41_WebServer_DHT22 Sensor

- ผลของการทดสอบการทำงาน



- WebServer Test Code

```
#include <WiFi.h>
#include "DHTesp.h"
#define DHT_Pin 4
const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";

DHTesp dht;
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  Serial.print("\n\nConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  { delay(500); Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());
  dht.setup(DHT_Pin, DHTesp::DHT22); // DHT_Pin D4, DHT22
}

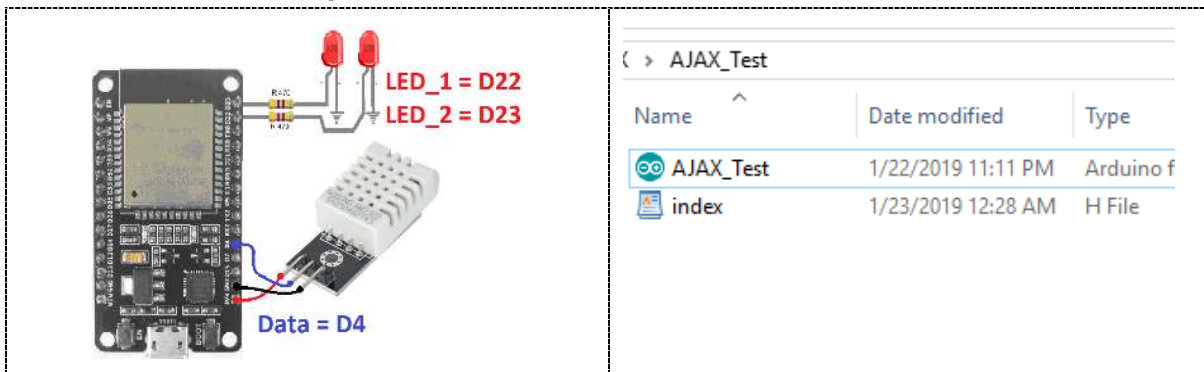
void loop() {
  WiFiClient client = server.available(); // wait Client request
  if (client) {
    Serial.println("new client"); // an http request ends with a blank line
    Serial.println("Requesting temperatures...");
    Serial.print("Temperature is: ");
    float Humid = dht.getHumidity();
    float cTemp = dht.getTemperature();
    Serial.print(cTemp, 2); Serial.print("C, ");
    Serial.print(Humid, 2); Serial.println("%H");
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) // send a standard http response header
        { client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close"); // the connection will be closed after completion of the response
          client.println("Refresh: 5"); // refresh the page automatically every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          client.print("Temperature > "); client.print(cTemp, 2);
          client.print("C, Humidity > "); client.print(Humid, 2);
          client.print("%H");
          client.println("<br />");
          client.println("</html>");
          break;
        }
      }
      if (c == '\n') // you're starting a new line
      { currentLineIsBlank = true;
      }
      else if (c != '\r') // you've gotten a character on the current line
      { currentLineIsBlank = false;
      }
    }
  }
  delay(1); // give the web browser time to receive the data
  client.stop(); // close the connection:
  Serial.println("client disconnected");
}
}
```

การทดลอง 8: Web Server-AJAX

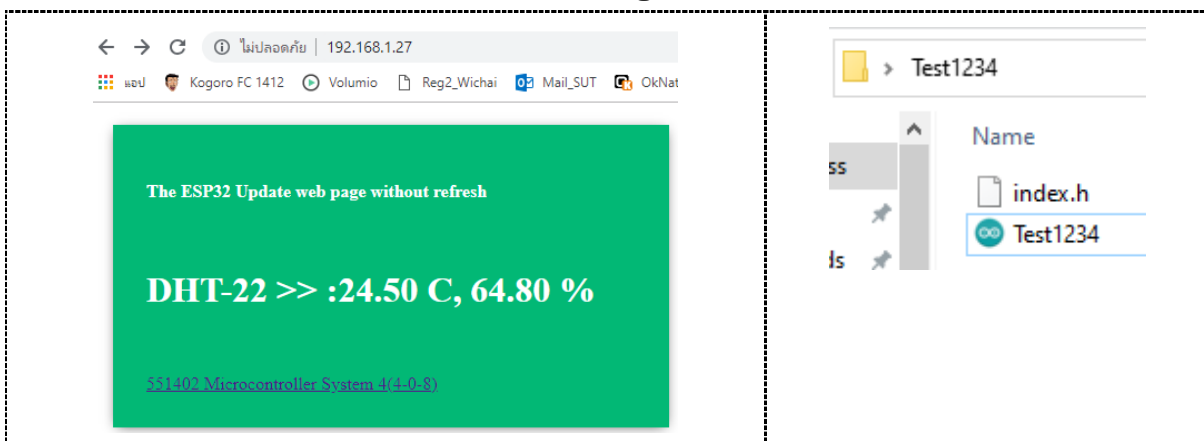
16. Test Ex50_WebServer AJAX_Monitor by AJAX

ในตัวอย่างก่อนหน้านี้ การ Update สถานะของการกด Button จะใช้การ refresh หน้า web ทั้งหมดทุกๆ 1 วินาที ทำให้ทั้งหน้ากระพริบ และ เป็นการรับ/ส่ง Data ที่ค่อนข้างสิ้นเปลือง เนื่องจากในบางส่วนไม่ได้มีการเปลี่ยนแปลงแต่เราต้อง update ทั้งหมด ในการทดลองนี้เราได้นำเอา Ajax เข้ามาช่วยให้สามารถ update ข้อมูลมาแสดงเฉพาะส่วนที่มีการเปลี่ยนแปลง ทำให้ไม่ต้อง refresh ทั้งหมด และลดภาระของ Web server ให้ทำงานน้อยลง

- Read More <https://circuits4you.com/2018/11/20/web-server-on-esp32-how-to-update-and-display-sensor-values/>
- Read More <https://circuits4you.com/2018/02/04/esp8266-ajax-update-part-of-web-page-without-refreshing/>
- ตัวอย่างดังรูป ดังนี้



- Create Program “Test_AJAX_01.ino”
- Open Notepad, Create “index.h” file and save to Test_AJAX_01 folder
- Upload “Test_AJAX_01.ino” to ESP-32
- เปิด Web Browser แล้วเรียกหน้า Page ไปยัง IP ของ ESP-32



- Code “Test_AJAX_01.ino” – WebServer AJAX Monitor by AJAX

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include "index.h" //Web page header file

#include "DHTesp.h"
#define DHT_Pin 4 // DHT-11 Pin D4

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";

DHTesp dht;
WebServer server(80);

void handleRoot() {
  String s = MAIN_page; //Read HTML contents
  server.send(200, "text/html", s); //Send web page
}

void handleADC() {
  float h = dht.getHumidity();
  float t = dht.getTemperature();
  String Value = String(t) + " C, ";
  Value += String(h) + " %";
  Serial.print("DHT-22 >> ");
  Serial.println(Value);
  server.send(200, "text/plain", Value);
}

void setup(void) {
  Serial.begin(115200);
  Serial.println("\nBooting Sketch...");
  WiFi.mode(WIFI_STA); //Connectto your wifi
  WiFi.begin(ssid, password);
  Serial.println("Connecting to ");
  Serial.print(ssid);

  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.print(".");
    delay(10);
  }

  Serial.print("\nConnected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP()); //IP address assigned to your ESP

  server.on("/", handleRoot); //This is display page
  server.on("/readADC", handleADC); //To get update of ADC Value only
  server.begin(); //Start server
  Serial.println("HTTP server started");
  dht.setup(DHT_Pin, DHTesp::DHT22); // DHT_Pin D4, DHT22
}

void loop(void) {
  server.handleClient();
  delay(1);
}
```

- Test_AJAX_01 = "index.h"

```
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<style>
.card{
  max-width: 500px;
  min-height: 250px;
  background: #02b875;
  padding: 30px;
  box-sizing: border-box;
  color: #FFF;
  margin:20px;
  box-shadow: 0px 2px 18px -4px rgba(0,0,0,0.75);
}
</style>
<body>

<div class="card">
  <h4>The ESP32 Update web page without refresh</h4><br>
  <h1>DHT-22 >> :<span id="ADCValue">0</span></h1><br>
  <br><a href="https://www.facebook.com/groups/311747285898180/">551402 Microcontroller System 4(4-0-8)</a>
</div>
<script>

setInterval(function() {
  // Call a function repetatively with 2 Second interval
  getData();
}, 2000); //2000mSeconds update rate

function getData() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("ADCValue").innerHTML =
        this.responseText;
    }
  };
  xhttp.open("GET", "readADC", true);
  xhttp.send();
}
</script>
</body>
</html>
)=====";
```

17. Test Ex51_WebServer AJAX_Control Monitor by AJAX

- Create Program “Test_AJAX_02.ino”
- Open Notepad, Create “index.h” file and save to Test_AJAX_02 folder
- Upload “Test_AJAX_02.ino” to ESP-32
- เปิด Web Browser แล้วเรียกหน้า Page ไปยัง IP ของ ESP-32



- Code “Test AJAX_02.ino” – WebServer AJAX Monitor by AJAX

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include "DHTesp.h"

#include "index.h" //Our HTML webpage contents with javascripts
#define DHT_Pin 4
#define testLED1 22
#define testLED2 23

//SSID and Password of your WiFi router
const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";

WebServer server(80); //Server on port 80
DHTesp dht;

String ledState1 = "NA";
String ledState2 = "NA";

//=====
// This routine is executed when you open its IP in browser
//=====
void handleRoot() {
  String s = MAIN_page; //Read HTML contents
  server.send(200, "text/html", s); //Send web page
}

void handleADC() {
  float h = dht.getHumidity();
  float t = dht.getTemperature();
  String tmpValue = "Temp = ";
  tmpValue += String(t) + " C, Humidity = ";
  tmpValue += String(h) + " %";
  server.send(200, "text/plain", tmpValue); //Send value to client ajax request
}

void handleLED() {
  String t_state = server.arg("LEDstate"); //Refer xhttp.open("GET", "setLED?LEDstate="+led, true);
  Serial.println(t_state);
  if (t_state == "11") { digitalWrite(testLED1, HIGH); ledState1 = "ON"; } //Feedback parameter
  if (t_state == "10") { digitalWrite(testLED1, LOW); ledState1 = "OFF"; } //Feedback parameter
  if (t_state == "21") { digitalWrite(testLED2, HIGH); ledState2 = "ON"; } //Feedback parameter
  if (t_state == "20") { digitalWrite(testLED2, LOW); ledState2 = "OFF"; } //Feedback parameter
  server.send(200, "text/plain", ledState1+" "+ledState2); //Send web page
}

void setup(void) {
  Serial.begin(115200);
  dht.setup(DHT_Pin, DHTesp::DHT22); // DHT_Pin D4, DHT22
  pinMode(testLED1, OUTPUT);
  pinMode(testLED2, OUTPUT);
  Serial.print("\n\nConnect to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.print("\nConnected "); Serial.println(ssid);
  Serial.print("IP address: "); Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.on("/setLED", handleLED);
  server.on("/readADC", handleADC);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient(); //Handle client requests
}
```

- Test AJAX_02 = "index.h"

```
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h1>The ESP-32 Update web page without refresh</h1>
<button type="button" onclick="sendData(11)" style="background: rgb(202, 60, 60);">LED1 ON</button>
<button type="button" onclick="sendData(10)" style="background: rgb(100,116,255);">LED1 OFF</button><br><br>
<button type="button" onclick="sendData(21)" style="background: rgb(202, 60, 60);">LED2 ON</button>
<button type="button" onclick="sendData(20)" style="background: rgb(100,116,255);">LED2 OFF</button><br><br>
State of [LED1, LED2] is >> <span id="LEDState">NA</span><br>
</div>

<div>
<br>DHT-22 sensor : <span id="ADCValue">0</span><br>
</div>

<script>
function sendData(led) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("LEDState").innerHTML =
        this.responseText;
    }
  };
  xhttp.open("GET", "setLED?LEDstate="+led, true);
  xhttp.send();
}

setInterval(function() {
  // Call a function repetatively with 2 Second interval
  getData();
}, 2000); // 2000mSeconds update rate

function getData() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("ADCValue").innerHTML =
        this.responseText;
    }
  };
  xhttp.open("GET", "readADC", true);
  xhttp.send();
}
</script>

<br><a href="https://www.facebook.com/groups/557681561647621/">551402 Microcontroller System 4(4-0-8)</a>

</body>
</html>
)=====";
```

Quiz_202 – Web Control 4 LED and Monitor Humid/Temperature

- เพิ่มเติมจาก Q202 อยากได้ปุ่มสำหรับคุมปิด-เปิด หลอดไฟ LED 4 ดวง
- อยากมีกด Link ไปที่หน้า FB ของตัวเอง
- https://www.colorhexa.com/008cba?fbclid=IwAR3dIZ_gRgDWmREmnzknLbMxV3pOHY4YIPuLEz8-ZzTOX2VhWxcH2QjLGk

← → ↻

Not secure | 192.168.43.237

The ESP-32 Update web page without refresh

LED1 ON

LED2 ON

LED3 ON

LED4 ON

LED1 OFF

LED2 OFF

LED3 OFF

LED4 OFF

State of [LED1, LED2, LED3, LED4] is >> ON, OFF, OFF, ON

DHT-22 sensor : Temp = 28.10 C, Humidity = 43.90 %

By [Wichai Srisuruk](#)

```

58   ledState2 = "OFF";
59   }
60   if (t_state == "31") {
61       digitalWrite(testLED1, HIGH); //Feedback parameter
62       ledState3 = "ON";
63   }
64   if (t_state == "30") {
65       digitalWrite(testLED1, LOW); //Feedback parameter
66       ledState3 = "OFF";
67   }
68   if (t_state == "41") {
69       digitalWrite(testLED1, HIGH); //Feedback parameter
70       ledState4 = "ON";
71   }
72   if (t_state == "40") {
73       digitalWrite(testLED1, LOW); //Feedback parameter
74       ledState4 = "OFF";
75   }
76   server.send(200, "text/plain", ledState1 + ", " + ledState2 + ", " + ledState3 + ", " + ledState4); //Send web page
77   }
78   }

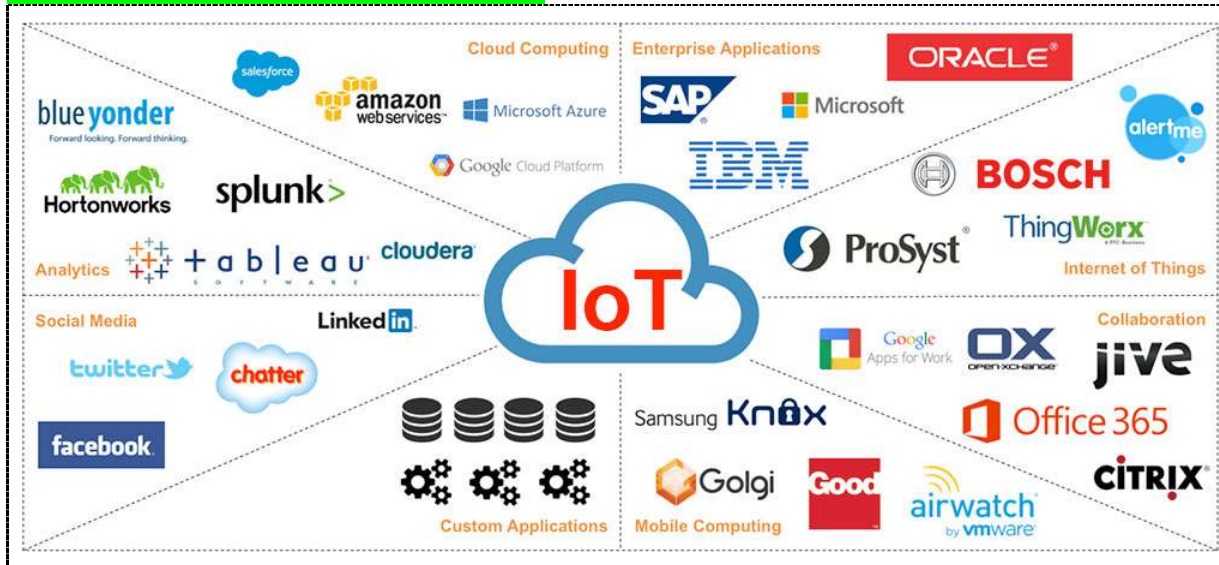
```

```

<div id="demo">
<h1>The ESP-32 Update web page without refresh</h1>
<button type="button" onclick="sendData(11)" style="background: rgb(202, 60, 60); height: 40px; width: 100px">LED1 ON</button>
<button type="button" onclick="sendData(21)" style="background: rgb(202, 60, 60); height: 40px; width: 100px">LED2 ON</button>
<button type="button" onclick="sendData(31)" style="background: rgb(202, 60, 60); height: 40px; width: 100px">LED3 ON</button>
<button type="button" onclick="sendData(41)" style="background: rgb(202, 60, 60); height: 40px; width: 100px">LED4 ON</button><br><br>
<button type="button" onclick="sendData(10)" style="background: rgb(100,116,255); height: 40px; width: 100px">LED1 OFF</button>
<button type="button" onclick="sendData(20)" style="background: rgb(100,116,255); height: 40px; width: 100px">LED2 OFF</button>
<button type="button" onclick="sendData(30)" style="background: rgb(100,116,255); height: 40px; width: 100px">LED3 OFF</button>
<button type="button" onclick="sendData(40)" style="background: rgb(100,116,255); height: 40px; width: 100px">LED4 OFF</button><br><br>
State of [LED1, LED2, LED3, LED4] is >> <span id="LEDState">NA</span><br>
</div>

```

3/4 การโปรแกรมใช้งานแบบ Internet of Things

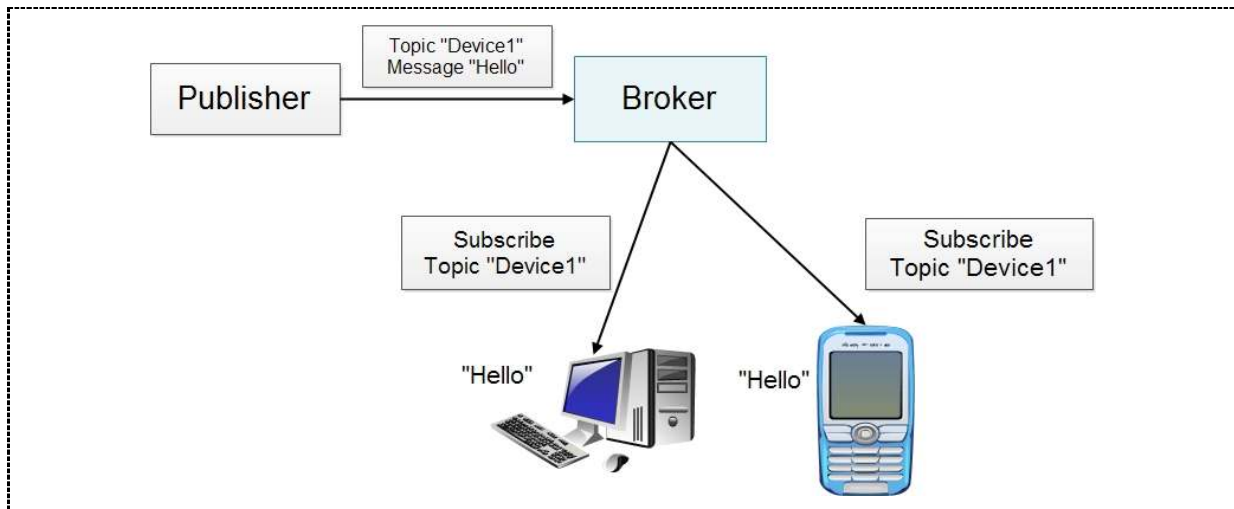


3.1 IoT Concept

ปัจจุบันเทคโนโลยีที่กำลังมาแรงสำหรับนักพัฒนาด้าน Embedded (ไม่ได้แค่เฉพาะ Embedded อย่างเดียวครับ) เป็นเทคโนโลยีที่กำลังมาแรงมากคงจะหนีไม่พ้น IoT ซึ่งเป็นเทคโนโลยีใหม่ในยุคนี้เลยก็ว่าได้ แรงขนาดที่ว่า Microsoft เอง ก็ยังพอร์ต Windows 10 มาวิ่งเล่นบน Raspberry Pi แถมยังใจดีติด IoT มาให้ด้วย ซึ่งผมยังไม่ได้ตามลงไปดูว่าใช้ Broker ตัวไหน และมี Library ให้ใช้งานมาด้วยหรือไม่? หรือผมก็เข้าใจผิดเกี่ยวกับมันครับถ้าผิดพลาดก็ขออภัยมานะที่นี้ด้วยครับ.

IoT มันคืออะไร พอค้นดูมีหลายลิงค์อธิบายไว้มากมาย เช่น [Internet of Things เมื่อคอมพิวเตอร์เริ่มคุยกันเองได้](#), [โลกแห่ง IoT มาถึงแล้ว](#), [IoT เทคโนโลยีที่ธุรกิจต้องรู้](#). ลองนึกภาพดูครับว่าถ้าหากอุปกรณ์สามารถส่งงานไปมาหากันได้ผ่าน www ไม่ว่าจะเป็น PC, Smart Phone หรือแม้แต่อุปกรณ์ขนาดเล็กพวก Micro-Controller, PLC, HUB, Switch หรืออะไรก็แล้วแต่ที่มันสามารถต่อระบบ Network ไม่ว่ามันจะอยู่ที่บ้าน ที่โรงงาน ไร่ นา ฟาร์มโรงเรือน โรงงานอุตสาหกรรมหรือที่อื่นๆ ที่มีระบบเน็ตเวิร์กที่เข้าถึง www ได้เราจะสามารถควบคุมมันได้ทั้งหมดที่ไหนก็ได้ในโลกใบนี้

องค์ประกอบหลักของ IoT จะมี 3 ส่วนคือ Broker, Publisher และ Subscriber. ซึ่งการรับและส่งข้อมูลนั้นมันจะส่งข้อมูลไปมาหากันนั้นจะส่งผ่านตัวกลางนั้นก็คือ Broker Server โดยตัวส่งนี้จะเรียกว่า Publisher ส่งข้อมูลขึ้นไปยัง Broker พร้อมระบุหัวข้อ (Topic) ที่ต้องการส่งข้อมูลออกไป จากนั้นตัวรับซึ่งเรียกว่า Subscriber ถ้าหากตัวรับต้องการรับข้อมูลจากตัวส่งจะต้องทำการ Subscribe หัวข้อ Topic ของ Publisher นั้นๆ ผ่าน Broker เช่นกัน



จากรูปภาพด้านบนจะมีตัว Publisher ทำการ Publish ข้อความ “Hello” ใน Topic Device1 เมื่อและถ้าหากมีคอมพิวเตอร์ หรืออุปกรณ์อื่นๆทำการ Subscribe หัวข้อ Topic Device1 เมื่อ Publisher ทำการส่งข้อมูลไปยัง Topic อุปกรณ์ Subscribe จะได้ข้อความ “Hello” เช่นเดียวกัน. ก็คล้ายๆกับที่ใช้งานไลน์ที่คุยกันเป็นกลุ่มนั้นเลยครับ. ซึ่งจะเห็นข้อความ “Hello” ในเวลาเดียวกันนั้นหมายความว่าอุปกรณ์ใดๆที่ทำการ Subscribe Topic เดียวกันก็จะได้รับข้อความเดียวกันครับ

3.2 MQTT-Message Queue Telemetry Transport

โปรโตคอลที่ใช้สำหรับรับและส่งข้อมูลนั้นคือ MQTT ปัจจุบันถึง Version 3.1 ในที่นี้จะมาทำการทดลองส่งข้อมูลกันตัว Server จะมีอยู่ด้วยกันหลายค่ายครับสำหรับที่ลิสมาด้านล่างนี้ครับ

Open Source MQTT Broker Server

- Mosquitto
- RSMB
- ActiveMQ
- Apollo
- Moquette
- Mosca
- RabbitMQ

Client

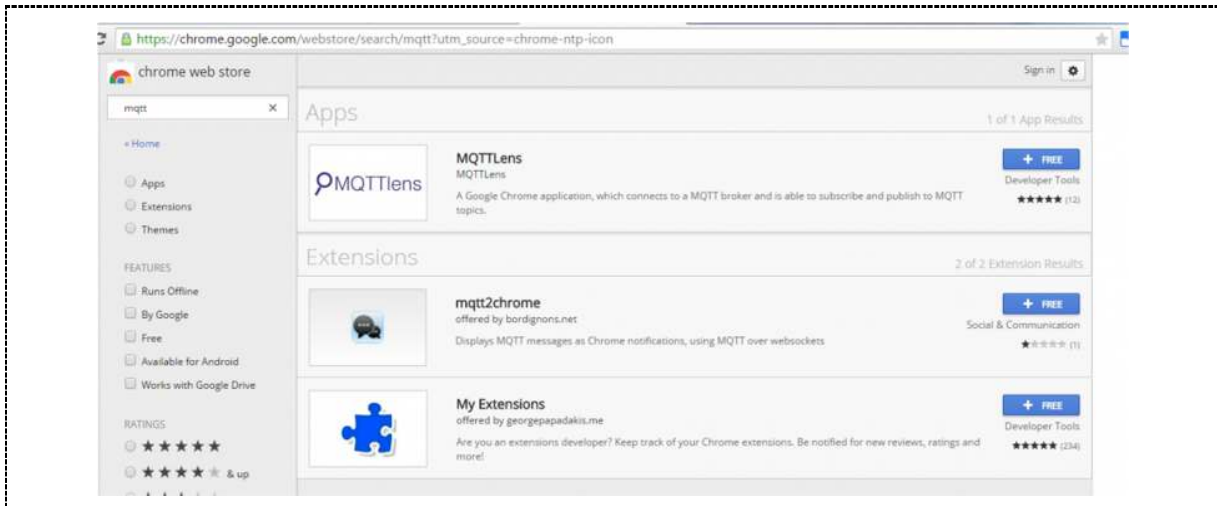
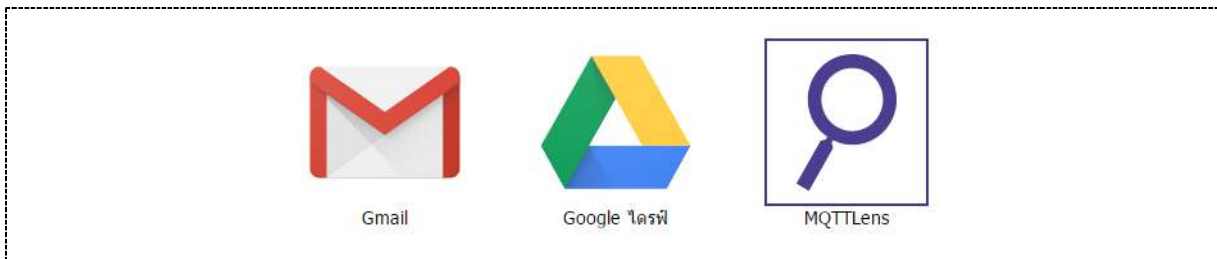
- Paho
- Xenqtt
- mqtt.js
- node_mqtt_client
- Ascoltatori
- Arduino MQTT Client

สำหรับ MQTT Broker Server ฟรีที่ผมพอค้นได้ก็มีดังนี้ครับ

- test.mosquitto.org
- mqtt.eclipse.org
- broker.mqttdashboard.com

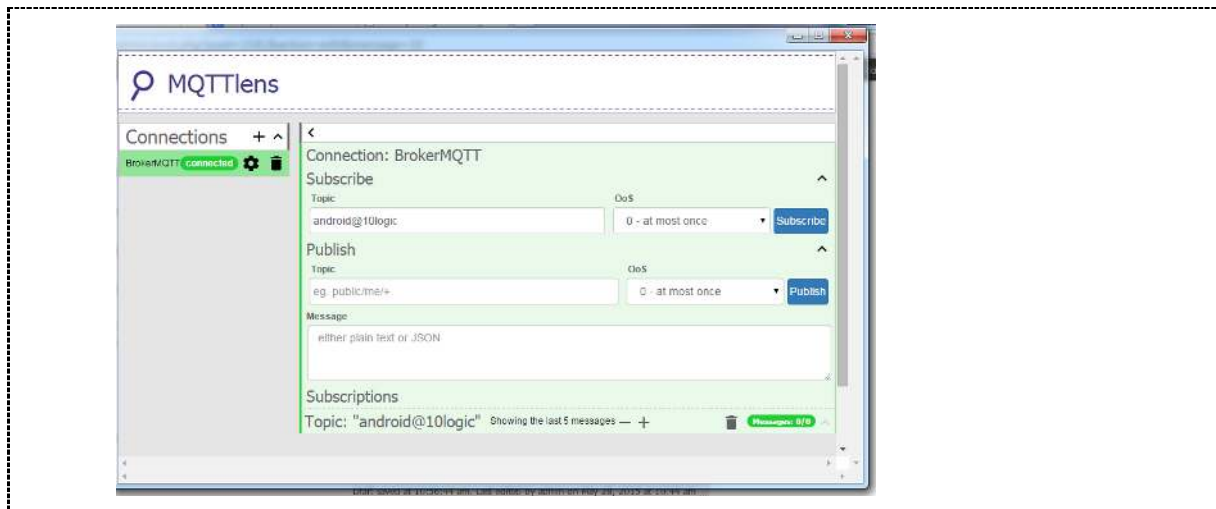
Step1a/3 กำหนดตัว Subscribe

สำหรับเครื่องมีสำหรับทดสอบที่จะทำการส่งข้อมูล(pub) และรับข้อมูล(sub) ก็มียู่ด้วยกันหลายตัวครับเช่น แต่จะเลือกมาใช้งานสักตัวหนึ่ง ในที่นี้ผมเลือกเป็น plugin สำหรับ chrome คือ MQTTLens

*Mqttlens**mqttlens*

เปิด MQTTLens ขึ้นมาจากนั้นป้อนรายละเอียด เมื่อป้อนรายละเอียดครบให้คลิกที่ CREATE CONNECTION

- Connection Name: test_MQTT ← อะไรก็ได้
- Hostname: test.mosquitto.org
- Port: 1883 ← default = 1883
- Client ID: RXL77Nb ← ตามที่ MQTTLens ให้มา



mqttlens

- Subscribe: android@10logic

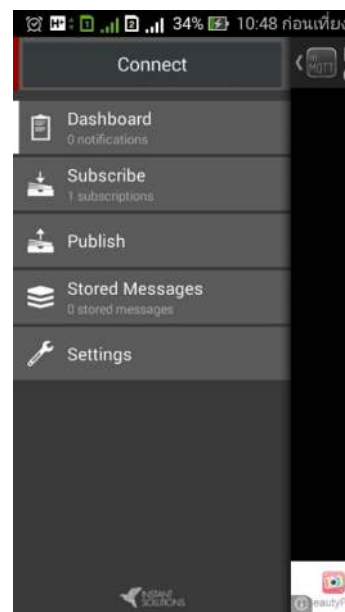
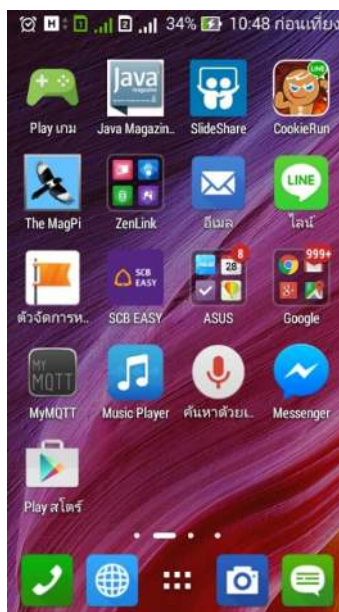
ในที่นี้ผมจะทำการ Subscriber ที่ Topic ชื่อว่า android@10logic

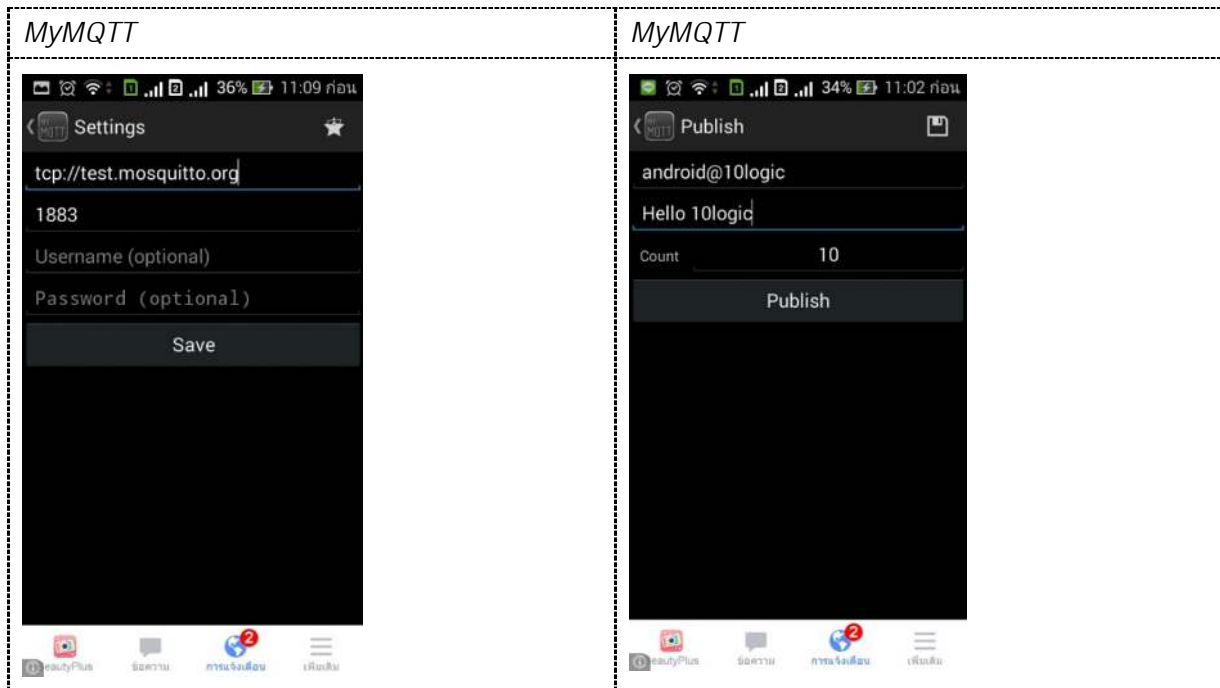
Step2a/3: กำหนดตัว Publisher

Publisher ซึ่งเป็น App สำหรับ Android ทำการ Public ข้อความ Hello 10logic ไปยัง Topic android@10logic เข้าไปใน play store และค้นคำว่า MyMQTT แล้วติดตั้งลงบน Smart Phone ของเรารับ

MyMQTT

เปิด MyMQTT และเข้าไปยังเมนู Settings

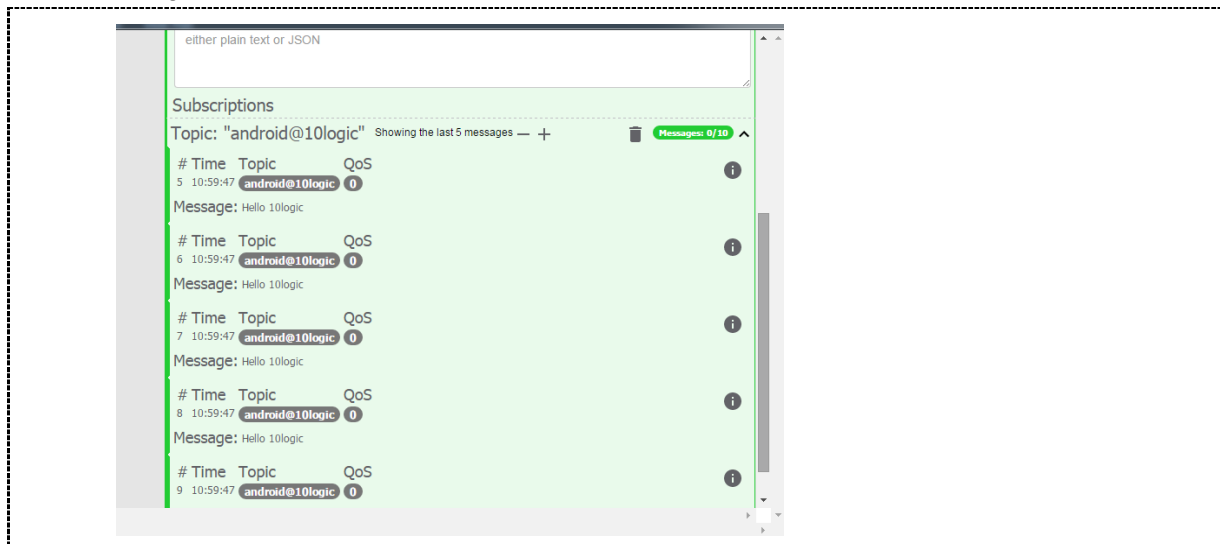




Step3a/3: ทดสอบการทำงาน

MyMQTT

เมื่อผมใช้ Smart Phone ที่มียูเอชการ Public ข้อความ “Hello 10logic” ผ่าน MyMQTT แสดงผลตามรูป



mqttlens

จะเห็นว่าสามารถรับข้อความ Hello 10logic ได้ตามตัวอย่างดังภาพ เห็นภาพกันแล้วใช่ไหมครับ
 ที่นี้เมื่อนักพัฒนาต้องการส่งข้อมูลจากอุปกรณ์ embedded สามารถส่งข้อมูลขึ้นมาได้เช่นกัน

3.3 IOT และ MQTT คือ อะไร?

MQTT ย่อมาจาก Message Queue Telemetry Transport เป็นโพรโตคอลประยุกต์ที่ใช้โพรโตคอล TCP เป็นรากฐาน ออกแบบมาสำหรับงานที่ต้องการ ๗ สื่อสารแบบเรียลไทม์แบบไม่จำกัดแพลตฟอร์ม หมายถึงอุปกรณ์ทุกชิ้นสามารถสื่อสารกันได้ผ่าน MQTT

MQTT จะแบ่งเป็น 2 ฝั่ง คือฝั่งเซิร์ฟเวอร์มักจะเรียกว่า MQTT Broker ส่วนฝั่งผู้ใช้งานจะเรียกว่า MQTT Client ในการใช้งานด้าน IoT จะเกี่ยวข้องกับ MQTT Client เป็นหลัก โดยจะมี MQTT Broker ทั้งแบบฟรี และเสียเงินไว้รองรับอยู่แล้ว ทำให้การสื่อสารข้อมูลผ่าน MQTT จะใช้เซิร์ฟเวอร์ฟรี หรือ MQTT Broker ฟรี เหล่านั้นเป็นตัวกลาง

ลักษณะการใช้งาน MQTT อาจจะเปรียบเสมือนได้กับการใช้งานห้องแชท Line สำหรับอุปกรณ์ โดยอุปกรณ์แต่ละตัวจะมีชื่อเป็นของตนเอง มี Username Password เป็นของตัวเอง และอาจจะมีการลับเฉพาะของตนเอง ดังนั้นการใช้งาน MQTT ผู้เขียนจึงจะขอยกตัวอย่างของ MQTT เทียบกับห้องแชทได้ดังนี้

กลุ่มผู้ใช้ (User)

ใน MQTT จะแบ่งกลุ่มของผู้ใช้งานออกเป็น 2 ระดับ คือ

- ระดับสูงสุด – สามารถที่จะรับ-ส่งข้อมูลกับอุปกรณ์ หรือช่องทางใด ๆ ก็ได้ในระบบ หรือเปรียบได้กับแอดมินที่สามารถเข้าไปดูข้อความได้ทุกห้องแม้จะเป็นห้องลับก็ตาม
- ระดับทั่วไป – สามารถรับ-ส่งข้อมูลกับอุปกรณ์หรือช่องทางที่กำหนดไว้เฉพาะเท่านั้นเปรียบได้กับผู้ใช้งาน Line ที่สามารถแชทในห้องที่ตัวเองสร้างได้ หรือเป็นสมาชิกในห้อง แต่ไม่สามารถเข้าไปแชทในห้องที่ไม่ได้เป็นสมาชิก

ในการใช้งานจริง ในอุปกรณ์ต่าง ๆ ควรจะใช้งานในระดับทั่วไป เพื่อความปลอดภัยกรณีอุปกรณ์เหล่านั้นถูกแฮกแล้วไม่สร้างความเสียหายไปยังอุปกรณ์อื่น ๆ ที่อยู่ในช่องทางเฉพาะของแต่ละอุปกรณ์

เส้นทาง (Topic)

เส้นทาง เปรียบเหมือนกับหัวข้อ หรือห้องแชทที่ต้องการจะคุย และการคุยกันจะมีเฉพาะอุปกรณ์ที่อยู่ในห้องนั้น ๆ (Subscribe) ถึงจะสามารถได้รับข้อมูลที่มีการส่งไปในห้องนั้น ๆ ที่ถูกเรียกว่าเส้นทาง เนื่องจากการใช้งานส่งข้อมูลและรับข้อมูลจะเหมือนกับเส้นทางในระบบไฟล์ เช่น /Room1/LED ซึ่งระบบเส้นทางนี้นอกจากอุปกรณ์จะสามารถรอการสนทนาในห้องตามเส้นทาง /Room1/LED ได้แล้ว ยังสามารถรอสนทนาเส้นทาง /Room1 ได้ด้วย หากเป็นการรอฟังในเส้นทาง (Subscribe) /Room1 จะหมายถึงการส่งข้อมูลใด ๆ ที่นำหน้าด้วย /Room1 เช่น /Room1/LED , /Room1/Value ผู้ที่รอฟัง (Subscribe) /Room1 อยู่จะได้รับข้อมูลเหล่านั้นด้วย

คุณภาพข้อมูล (QoS)

แบ่งออกเป็น 3 ระดับดังนี้

- QoS0 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่
- QoS1 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่ แต่ให้จำค่าที่ส่งล่าสุดไว้ เมื่อมีการเชื่อมต่อใหม่จะได้รับข้อมูลครั้งล่าสุดอีกครั้ง
- QoS2 – ส่งข้อมูลหลาย ๆ ครั้งจนกว่าปลายทางจะได้รับข้อมูล มีข้อเสียที่สามารถทำงานได้ช้ากว่า QoS0 และ QoS1

การส่งข้อมูล (Publish)

การส่งข้อมูลในแต่ละครั้งจะต้องประกอบไปด้วยเส้นทาง (Topic) ข้อมูล และคุณภาพข้อมูล ซึ่งการส่งข้อมูลจะเรียกว่า Publish

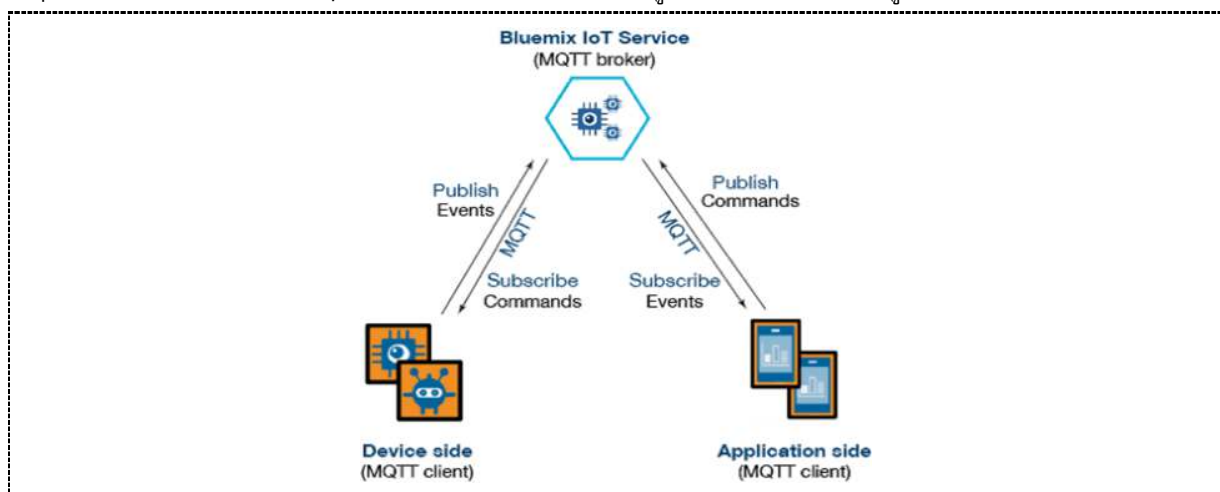
การรับข้อมูล (Subscribe)

การรับข้อมูลในระบบ MQTT จะรับข้อมูลได้เฉพาะเมื่อมีการเรียกใช้การ Subscribe ไปยัง Topic ที่กำหนด อาจเปรียบได้กับการ Subscribe คือการเข้าไปนั่งรอเพื่อนในกลุ่ม Line ส่งแชทมาหา เมื่อมีการส่งข้อมูลเข้ามาจะเกิดสิ่งที่เรียกว่าเหตุการณ์ (Event) ให้เราเข้าไปดูข้อความที่เพื่อน ๆ ส่งเข้ามา

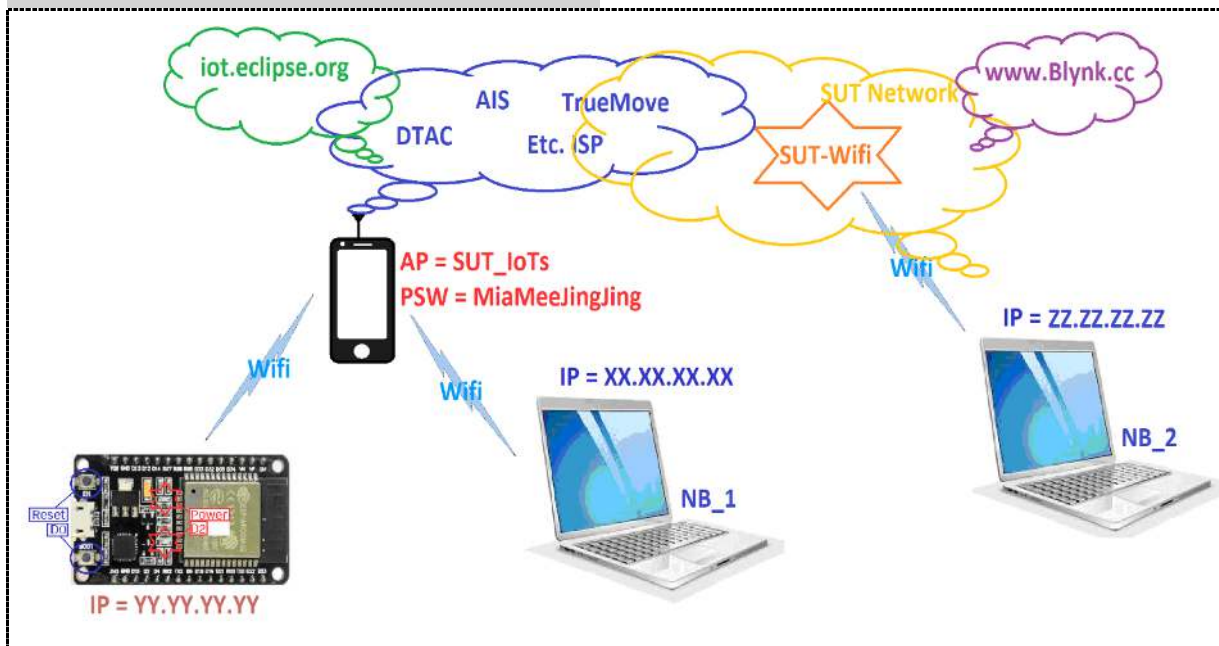
จะเห็นได้ว่า MQTT ก็เปลี่ยนเสมือนห้องแชทของอุปกรณ์ที่จะสนทนาแลกเปลี่ยนข้อมูลกันแบบเรียลไทม์ผ่านเครือข่ายอินเทอร์เน็ต

3.4 IoT มีวิธีการทำงานอย่างไร?

องค์ประกอบหลักของ IoT จะมี 3 ส่วนคือ **Broker**, **Publisher** และ **Subscriber**. ซึ่งการรับและส่งข้อมูลนั้นมันจะส่งข้อมูลไปมาหากันนั้นจะส่งผ่านตัวกลางนั้นก็คือ Broker Server โดยตัวส่งนี้จะเรียกว่า Publisher ส่งข้อมูลขึ้นไปยัง Broker พร้อมระบุหัวข้อ (Topic) ที่ต้องการส่งข้อมูลออกไป จากนั้นตัวรับซึ่งเรียกว่า Subscriber ถ้าหากตัวรับต้องการรับข้อมูลจากตัวส่งจะต้องทำการ Subscribe หัวข้อ Topic ของ Publisher นั้นๆ ผ่าน Broker เช่นกัน ลองดูความสัมพันธ์ ตามรูป



3.5 ขอแตกต่างระหว่าง IoT กับ Over Internet



การทดลองก่อนหน้านี้เป็นการควบคุมผ่านอินเทอร์เน็ต จำเป็นต้องรู้ IP ของอุปกรณ์ปลายทาง และระบบต้องอยู่ในวงเครือข่ายเดียวกัน เช่น จากรูปเราไม่สามารถใช้ NB_2 เข้ามาควบคุม ESP32 ได้โดยตรงเพราะ IP=ZZ.ZZ.ZZ.ZZ และ IP=YY.YY.YY.YY อยู่คนละเครือข่าย หากต้องการสามารถกำหนดเส้นทางจาก NB_2 ผ่านเครือข่ายของมหาวิทยาลัย ไปยังผู้ให้บริการมือถือ วนมาที่มือถือ เข้ามายัง ESP32 การควบคุมสั่งการแบบนี้จำเป็นต้องรู้เลขปลายทางซึ่งเป็นไอพีของอุปกรณ์

กรณีของ IoTs กระบวนการข้างต้นจะปรับเปลี่ยน คือ ไม่จำเป็นต้องรู้เลขไอพีของอุปกรณ์ปลายทางแต่ให้อุปกรณ์วิ่งไปรับคำสั่งที่ตัวกลาง (Broker) แทน จากรูป NB_2 จะส่งคำสั่ง (Publish) ไปยังตัวกลาง ตัวอุปกรณ์ปลายทางต้องแจ้งรับข้อความ (Subscribe) จากตัวกลาง เมื่อมีคำสั่งเข้ามา และตัวอุปกรณ์ปลายทางเข้ามารับข้อมูลอุปกรณ์ปลายทางคอยทำงานตามคำสั่งที่ได้รับ

เห็นได้ว่าแบบแรกจำเป็นต้องเข้าให้ถึง ESP32 แต่แบบหลังใช้วิธีนี้รับข้อความที่ตัวกลางที่ทั้งสองฝั่งตกลงกันได้

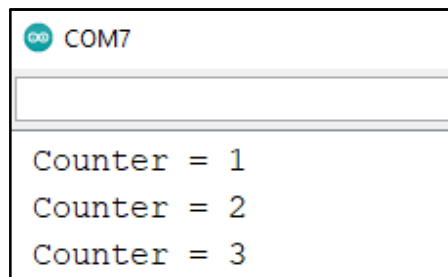
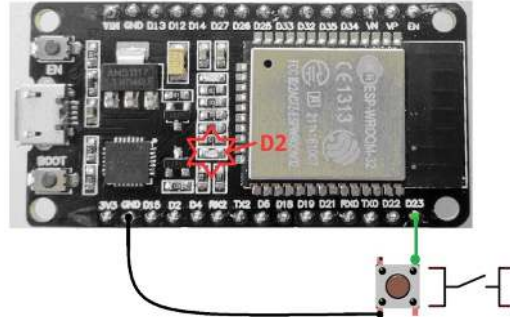
การทดลองที่ 9 – Publish

1. ทดสอบโปรแกรมนับจำนวนครั้งการกดสวิตช์ แสดงผลที่ Serial Monitor ด้วย baud rate 115200

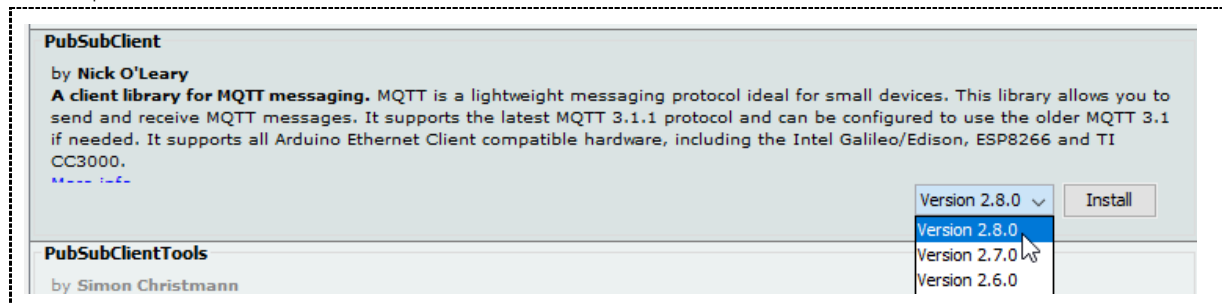
```
int pushButton = 23;
int LED_Monitor = 2;
int Counter = 0;

void setup() {
  Serial.begin(115200);
  pinMode(LED_Monitor, OUTPUT);
  pinMode(pushButton, INPUT_PULLUP);
  Serial.print(" Counter = ");
  Serial.println(Counter);
  digitalWrite(LED_Monitor, Counter % 2);
}

void loop() {
  if (digitalRead(pushButton) == 0)
  { Counter++;
    Serial.print(" Counter = ");
    Serial.println(Counter);
    digitalWrite(LED_Monitor, Counter % 2);
    while (digitalRead(pushButton) == 0);
    delay(100);
  }
}
```



2. Add Library → การใช้งาน MQTT บน ESP32 จะใช้งานผ่านไลบรารี PubSubClient.h จะต้องติดตั้งเพิ่มเติมโดยใช้ Library Manager ค้นหาว่า PubSubClient เลือก **PubSubClient Version 2.8.0** แล้วสามารถกดปุ่ม Install เพื่อติดตั้ง



3. โปรแกรมจะ Publish ไปยัง **test.mosquitto.org** ในหัวข้อ **myHome1234**

- Test Broker = **test.mosquitto.org**
- Port = **1883**
- Topic, Subscribe, Publish = **myHome1234**



MQTTLens

<https://mntolia.com/10-free-public-private-mqtt-brokers-for-testing-prototyping/>

<https://www.hivemq.com/blog/mqtt-toolbox-mqtt-lens/>

4. คำแนะนำการติดตั้งใช้งานตามนี้ <http://www.steves-internet-guide.com/using-mqtt-lens/>
5. เปิด MQTT Lens ตั้งค่าโบรคเกอร์เป็น → Test Broker = **test.mosquitto.org**, Port = **1883**,

Connection Details

Connection name

Connection color scheme

Hostname

tcp://

Port

6. กำหนด Topic → Topic, Subscribe, Publish = **myHome1234**

Connections + ^

Pk007

Connection: Pk007

Subscribe

0 - at most once

SUBSCRIBE

Publish

0 - at most once

☐ Retained

PUBLISH

Message

hello

Subscriptions

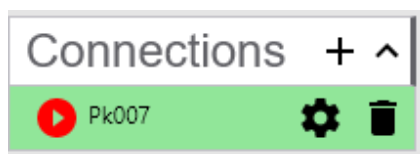
Topic: "myHome1234" Showing the last 5 messages — +

Time Topic QoS

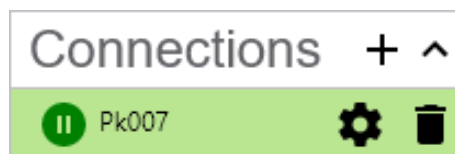
0 7:00:19 myHome1234 0

Message: hello

สีแดง - ยังไม่พร้อมใช้งาน



สีเขียว - พร้อมใช้งาน



7. โปรแกรมทดสอบ Publish ไปยัง **test.mosquitto.org** ในหัวข้อ **myHome1234**

```

#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "test.mosquitto.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

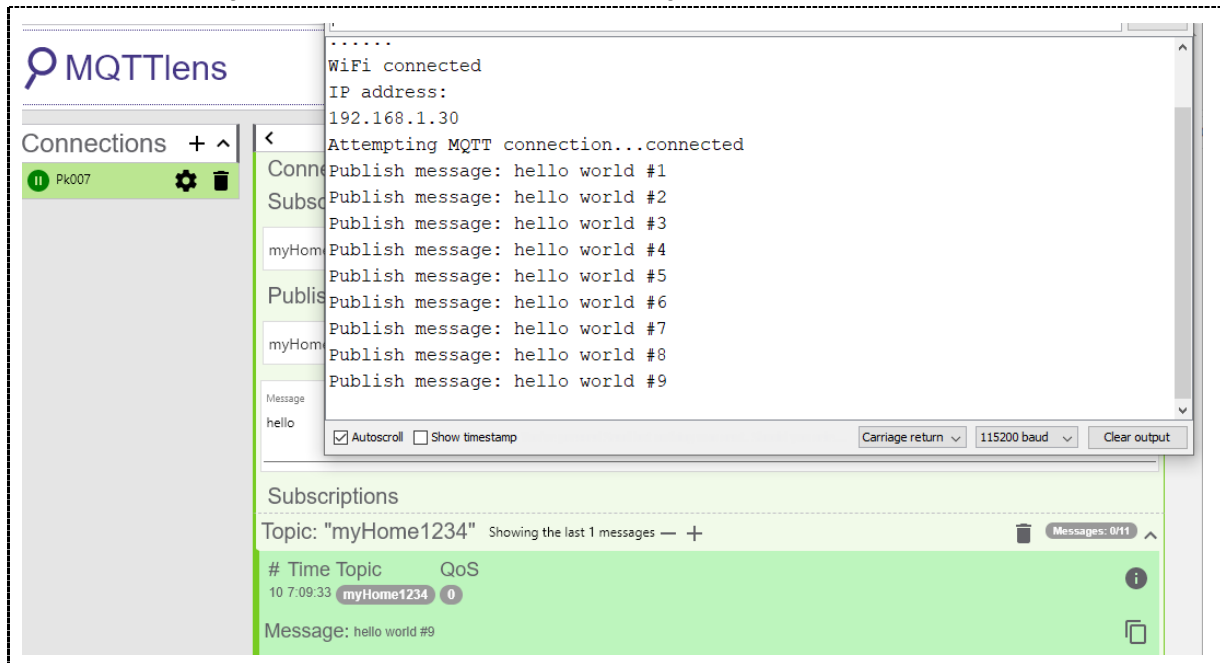
void reconnect()
{ while (!client.connected())          // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32 Client-";
    clientId += String(random(0xffff), HEX); // Create a random client ID
    if (client.connect(clientId.c_str())) // Attempt to connect
    { Serial.println("connected");        // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007"); // ... and resubscribe
      client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

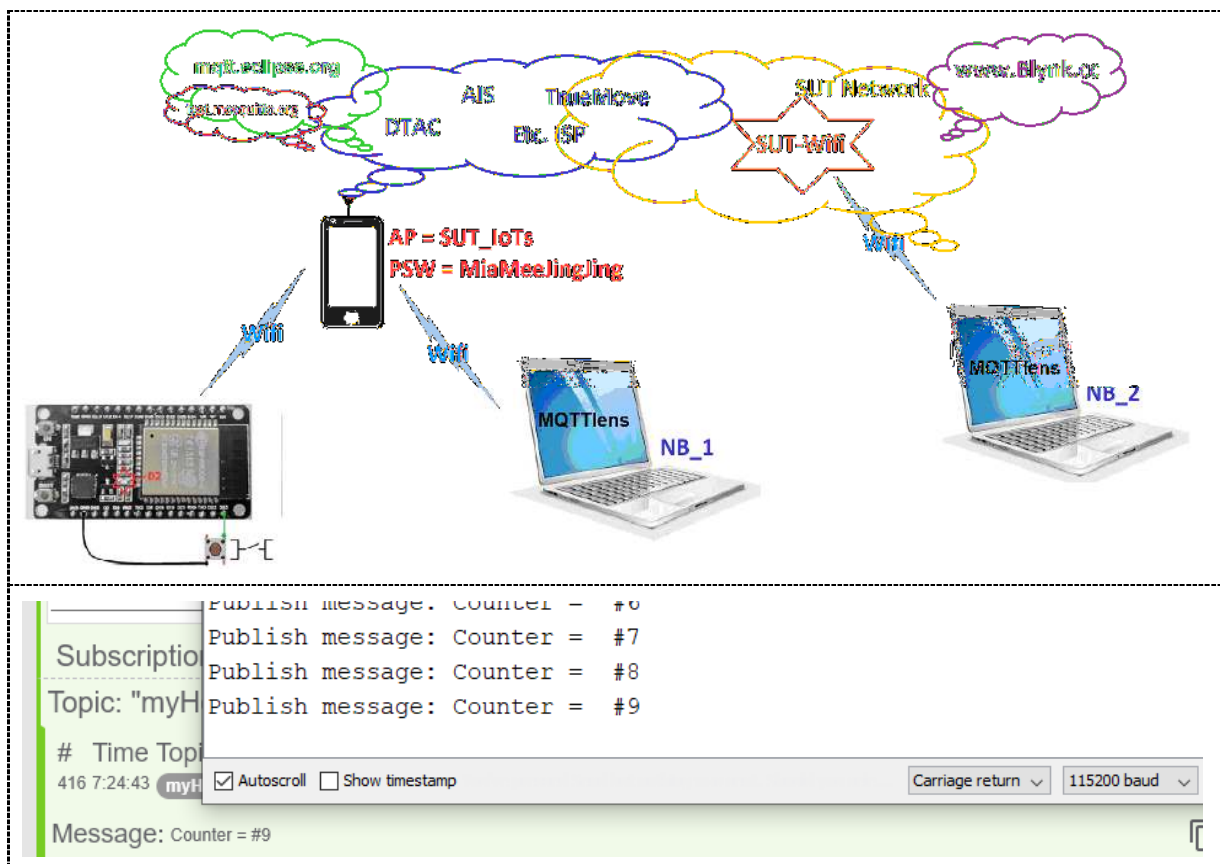
void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000)
  { lastMsg = now;
    ++value;
    snprintf (msg, 75, "hello world #%ld", value);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
  }
}

```


8. ผลการทำงานดูที่ Serial Monitor ว่าส่งอะไรออกไป และดูที่ MQTTLens ว่าได้รับอะไรมาบ้าง



9. ทดสอบโปรแกรมนับจำนวนครั้งการกดสวิตช์ ส่งค่าไปยัง MQTT Broker



10. โปรแกรมนับจำนวนครั้งการกดสวิตช์ ส่งค่าไปยัง MQTT Broker

```

#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "test.mosquitto.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];

int pushButton = 23;
int LED_Monitor = 2;
int Counter = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to "); Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  randomSeed(micros());
  Serial.println(""); Serial.println("WiFi connected");
  Serial.println("IP address: "); Serial.println(WiFi.localIP());
}

void reconnect()
{ while (!client.connected()) // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32 Client-";
    clientId += String(random(0xffff), HEX); // Create a random client ID
    if (client.connect(clientId.c_str())) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007"); // ... and resubscribe
      client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  pinMode(LED_Monitor, OUTPUT);
  pinMode(pushButton, INPUT_PULLUP);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

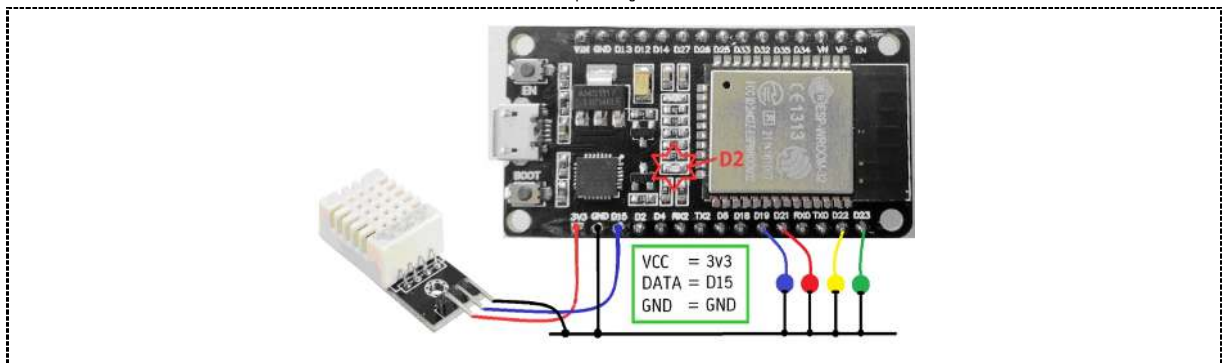
void loop()
{ if (digitalRead(pushButton) == 0)
  { Counter++;
    if (!client.connected()) reconnect();
    client.loop();
    snprintf(msg, 75, "Counter = %d", Counter);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
    digitalWrite(LED_Monitor, Counter % 2);
    while (digitalRead(pushButton) == 0);
    delay(100);
  }
}

```

Quiz_203 – Publish

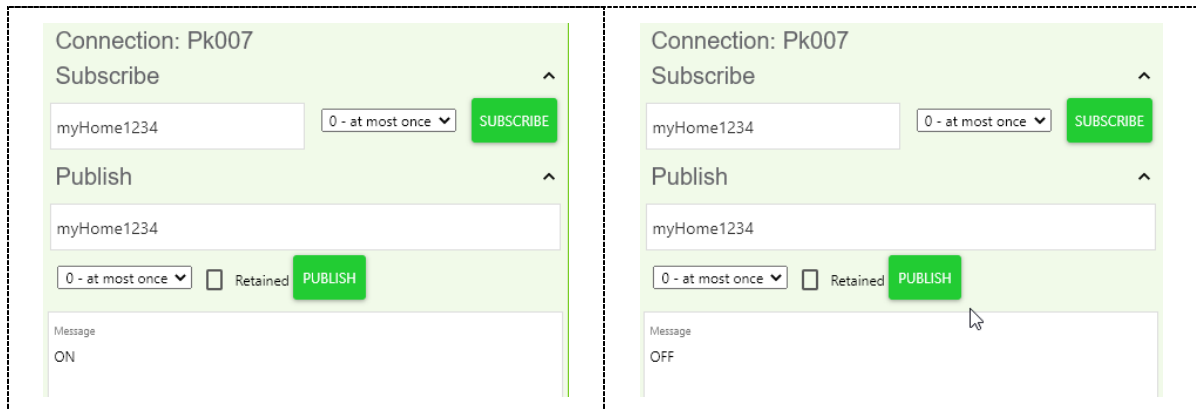
- อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- ควบคุมการแสดงผลให้ 4 LED แสดงผลตามข้อกำหนดดังนี้

*○○○(Blink)	หากการอ่านค่าแล้วเป็น null, หรือไม่มีเซ็นเซอร์
●○○○	ช่วงของอุณหภูมิ ($-\infty$, 24)
●●○○	ช่วงของอุณหภูมิ [24,26)
●●●○	ช่วงของอุณหภูมิ [26,28)
●●●●	ช่วงของอุณหภูมิ [28,30)
****(Blink)	ช่วงของอุณหภูมิ [30, ∞)



การทดลองที่ 10 – Publish and Subscribe

11. เมื่อโหลดโปรแกรมแล้วทำการทดสอบ publish “OFF” → Off LED และ “ON” → On LED



12. โปรแกรมการ Publish และ Subscribe หน่วงเวลารั้งละ 5 วินาที

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "test.mosquitto.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];

int TestLED = 2;
int value = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  pinMode(TestLED, OUTPUT);
}

void callback(char* topic, byte* payload, unsigned int length)
{
  char myPayload[50];
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
    myPayload[i] = payload[i];
    myPayload[i + 1] = '\0'; // End of String
  }
  Serial.print("\n ---> "); Serial.println(myPayload);
  myPayload[4] = '\0'; // String less than 4 characters
  if ((String)myPayload == "ON") digitalWrite(TestLED, HIGH);
  if ((String)myPayload == "OFF") digitalWrite(TestLED, LOW);
}

void reconnect()
{
  while (!client.connected()) // Loop until we're reconnected
  {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX); // Create a random client ID
    if (client.connect(clientId.c_str())) // Attempt to connect
    {
      Serial.println("connected"); // Once connected, publish an announcement...
    }
  }
}
```

```

        client.publish(topic1, "Hello World Pk007"); // ... and resubscribe
        client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(TestLED, OUTPUT);
}

void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000)
  { lastMsg = now;
    ++value;
    sprintf (msg, "hello world #%ld", value);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
  }
}

```

13. ทดสอบโปรแกรมนับจำนวนครั้งการกดสวิตช์ ส่งค่าไปยัง MQTT Broker และควบคุมการปิด-เปิด LED

```

// DOIT ESP-32

#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeiJingJing";
const char* mqtt_server = "test.mosquitto.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];

int TestLED = 2;
int pushButton = 23;
int Counter = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length)
{ char myPayload[50];
  Serial.print("Message arrived [");
  Serial.print(topic1);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
    myPayload[i] = payload[i];
    myPayload[i + 1] = '\0'; // End of String
  }
  Serial.print("\n ---> "); Serial.println(myPayload);
}

```

```

myPayload[4] = '\0'; // String less than 4 characters
if ((String)myPayload == "ON") digitalWrite(TestLED, HIGH);
if ((String)myPayload == "OFF") digitalWrite(TestLED, LOW);
}

void reconnect()
{ while (!client.connected()) // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32 Client-";
    clientId += String(random(0xffff), HEX); // Create a random client ID
    if (client.connect(clientId.c_str())) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007"); // ... and resubscribe
      client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

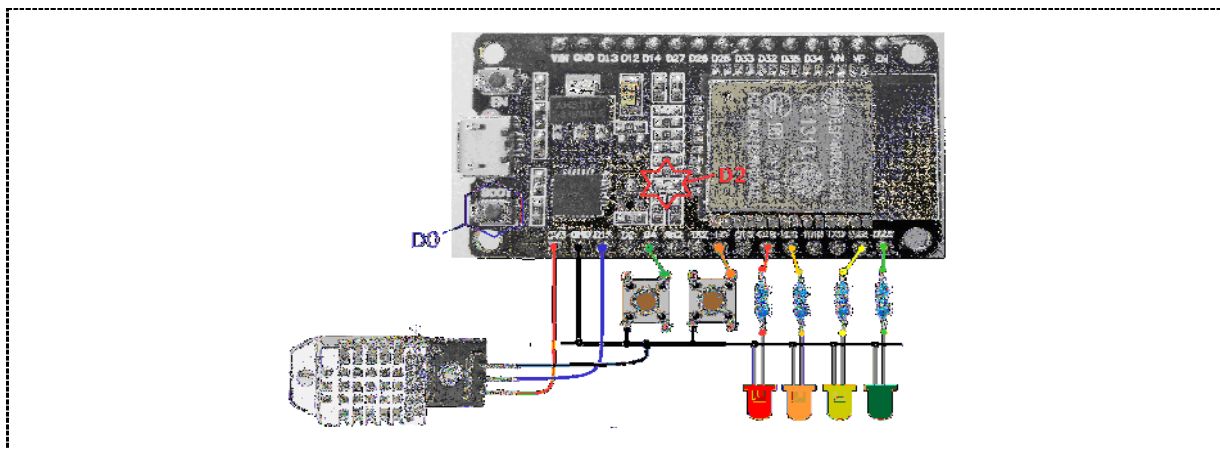
void setup()
{ Serial.begin(115200);
  pinMode(pushButton, INPUT_PULLUP);
  pinMode(TestLED, OUTPUT);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  if (digitalRead(pushButton) == 0)
  { Counter++;
    client.loop();
    sprintf (msg, "Count = %d", Counter);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
    while (digitalRead(pushButton) == 0);
    delay(100);
  }
}

```

Quiz_204 – Publish and Subscribe

- อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- ควบคุมการปิดเปิด 4 LED
- รับคำสั่งที่กำหนด SW1 แจ้ง Overheat Alarm, SW2 แจ้ง Intruders Alarm




แนวทางการใช้งานอินเทอร์เน็ตของสรรพสิ่งในระบบการผลิต
IoT Approaches to Manufacturing System

ชื่อ-สกุล :

4/4. คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_201 – Web Control 2 LED

- อยากได้ปุ่มสำหรับคุมปิด-เปิด หลอดไฟ LED 2 ดวง
- https://www.colorhexa.com/008cba?fbclid=IwAR3dIZ_gRgDWmREmnzuhnLbMxV3pOHY4YIPuLEz8-ZzTOX2VhWxcH2QjLGk


< Test Code >
รูปการต่อวงจร – 1
รูปการต่อวงจร – 2
หน้าจอ Web Control

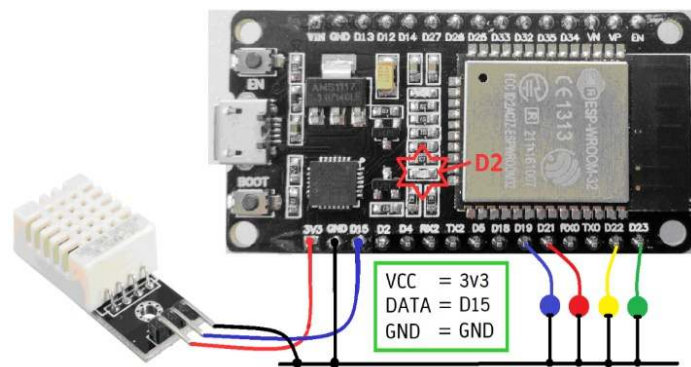
Quiz_202 – Web Control 4 LED and Monitor Humid/Temperature

- เพิ่มเติมจาก Q202 อยากได้ปุ่มสำหรับคุมปิด-เปิด หลอดไฟ LED 4 ดวง
- อยากมีกิด Link ไปที่หน้า FB ของตัวเอง

<p>← → ↻ ⓘ Not secure 192.168.43.237</p> <h2>The ESP-32 Update web page without refresh</h2> <div> <div>LED1 ON</div> <div>LED2 ON</div> <div>LED3 ON</div> <div>LED4 ON</div> </div> <div> <div>LED1 OFF</div> <div>LED2 OFF</div> <div>LED3 OFF</div> <div>LED4 OFF</div> </div> <p>State of [LED1, LED2, LED3, LED4] is >> ON, OFF, OFF, ON</p> <p>DHT-22 sensor : Temp = 28.10 C, Humidity = 43.90 %</p> <p>By Wichai Srisuruk</p>	
< Test Code >	รูปการต่อวงจร – 1
รูปการต่อวงจร – 2	หน้าจอ Web Control

Quiz_203 – Publish

- อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- ควบคุมการแสดงผลให้ 4 LED แสดงผลตามข้อกำหนดดังนี้
 - *○○○(Blink) หากการอ่านค่าแล้วเป็น null, หรือไม่มีเซ็นเซอร์
 - ช่วงของอุณหภูมิ ($-\infty$, 24)
 - ช่วงของอุณหภูมิ [24,26)
 - ช่วงของอุณหภูมิ [26,28)
 - ช่วงของอุณหภูมิ [28,30)
 - ****(Blink) ช่วงของอุณหภูมิ [30, ∞)



< Test Code >

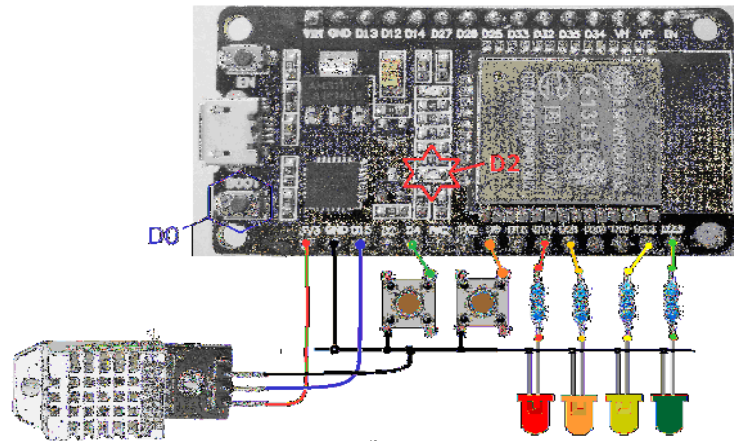
รูปการต่อวงจร – 1

รูปการต่อวงจร – 2

หน้าจอ MQTT Lens

Quiz_204 – Publish and Subscribe

- อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- ควบคุมการปิดเปิด 4 LED
- รับคำสั่งรหัสกำหนด SW1 แจ้ง Overheat Alarm, SW2 แจ้ง Intruders Alarm



< Test Code >

รูปการต่อวงจร – 1

รูปการต่อวงจร – 2

หน้าจอ MQTT Lens