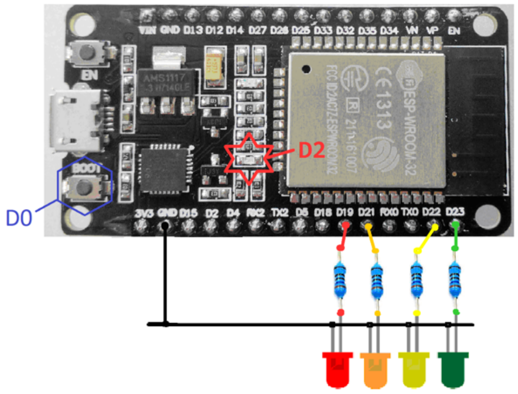
แนวทางการใช้งานอินเทอร์เน็ตของสรรพสิ่งในระบบการผลิต
IoT Approaches to Manufacturing System

ขื่อ-สกุล : B6304577 นายภานุพงศ์ แคนอินทร์

3/3. คำถามท้ายบทเพื่อทดสอบความเข้าใจ

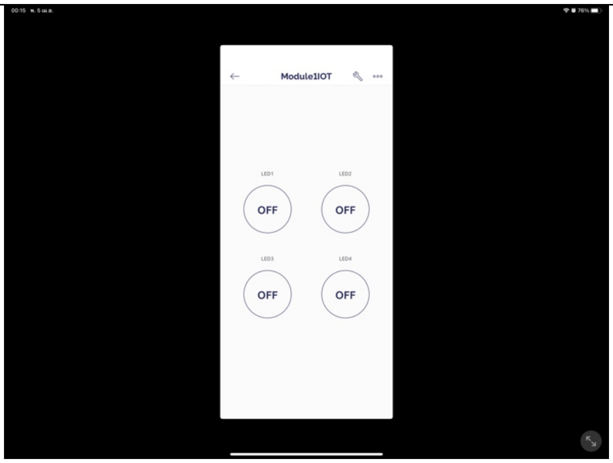
Quiz_301 – 4 External LED Control


VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
EN
BOOT
D2
D0
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
ESP-WROOM-32

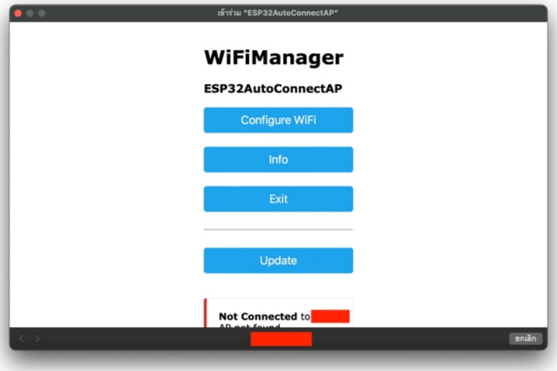
< Test Code >

```
#define BLYNK_PRINT Serial


/* Fill-in information from Blynk Device Info here */
#define BLYNK_TEMPLATE_ID "TMPxxxxxx"
#define BLYNK_TEMPLATE_NAME "Device"
#define BLYNK_AUTH_TOKEN "YourAuthToken"

#include <WiFiManager.h> // https://github.com/tzapu/WiFiManager
#include <BlynkSimpleEsp32.h>


char ssid[50] = ""; // increase the size of the character array to fit the maximum length of a WiFi
SSID
char pass[50] = ""; // increase the size of the character array to fit the maximum length of a WiFi
password


#define LED1 18
#define LED2 19
#define LED3 22
#define LED4 23
int ledState1 = LOW;
int ledState2 = LOW;
int ledState3 = LOW;
int ledState4 = LOW;

BLYNK_WRITE(V18) {
  ledState1 = param.asInt();
  digitalWrite(LED1, ledState1);
}
BLYNK_WRITE(V19) {
  ledState2 = param.asInt();
  digitalWrite(LED2, ledState2);
}
BLYNK_WRITE(V22) {
  ledState3 = param.asInt();
  digitalWrite(LED3, ledState3);
}
BLYNK_WRITE(V23) {
  ledState4 = param.asInt();
  digitalWrite(LED4, ledState4);
}

void setup()
{
  // Debug console
  Serial.begin(9600);

  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);
  pinMode(LED4,OUTPUT);

  ssid[0] = '\0'; // null-terminate the character array to avoid garbage values
  pass[0] = '\0'; // null-terminate the character array to avoid garbage values


  //WiFiManager, Local intialization. Once its business is done, there is no need to keep it around
  WiFiManager wm;
  bool res;
  // res = wm.autoConnect(); // auto generated AP name from chipid
  // res = wm.autoConnect("AutoConnectAP"); // anonymous ap
  res = wm.autoConnect("ESP32AutoConnectAP", "12345678"); // password protected ap

  if (!res) {
    Serial.println("Failed to connect");
    // ESP.restart();
  }
  else {
    //if you get here you have connected to the WiFi
    Serial.println("connected...yeey :)");
    Serial.println(res);
  }

  // use strcpy() to copy the String values into the character arrays
  strcpy(ssid, wm.getWiFiSSID().c_str());
  strcpy(pass, wm.getWiFiPass().c_str());

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
}

void loop()
{
  Blynk.run();
}
```

รูปหน้าจอ Blynk



รูปการต่อวงจร – 1

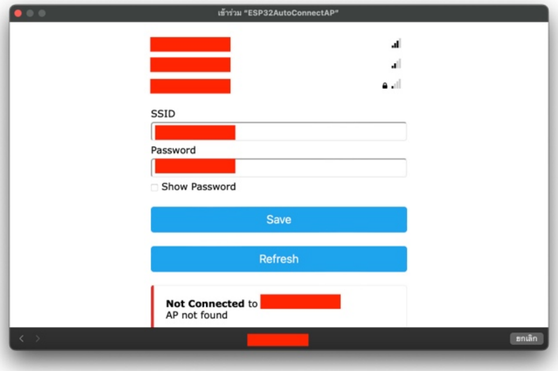


WiFiManager

ESP32AutoConnectAP

Not Connected to [redacted]

[redacted]



[redacted]

SSID [redacted]

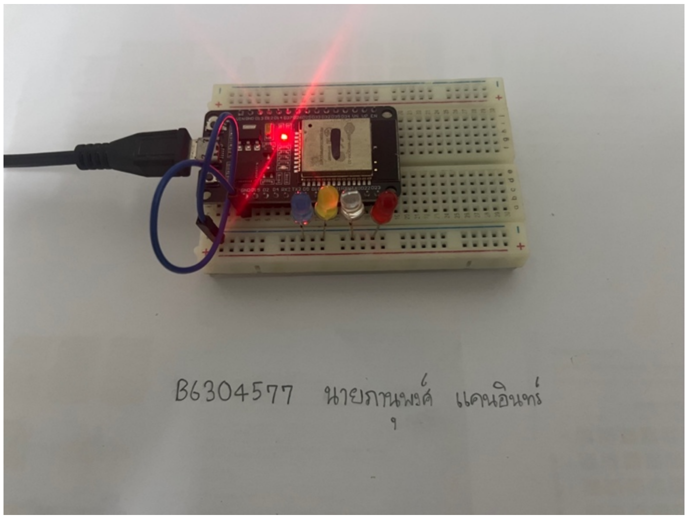
Password [redacted]

□ Show Password

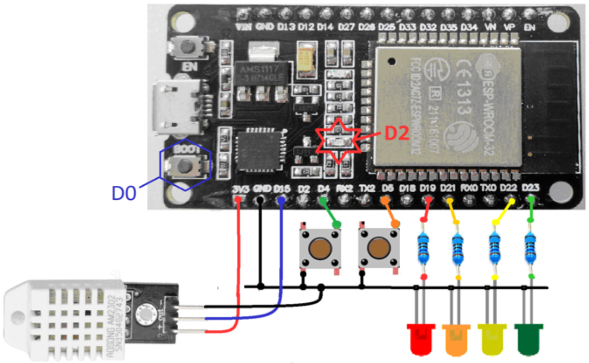
Not Connected to [redacted]
AP not found

[redacted]

รูปการต่อวงจร – 2



Quiz_302 – DHT22 + 4 LED + 2 Switch

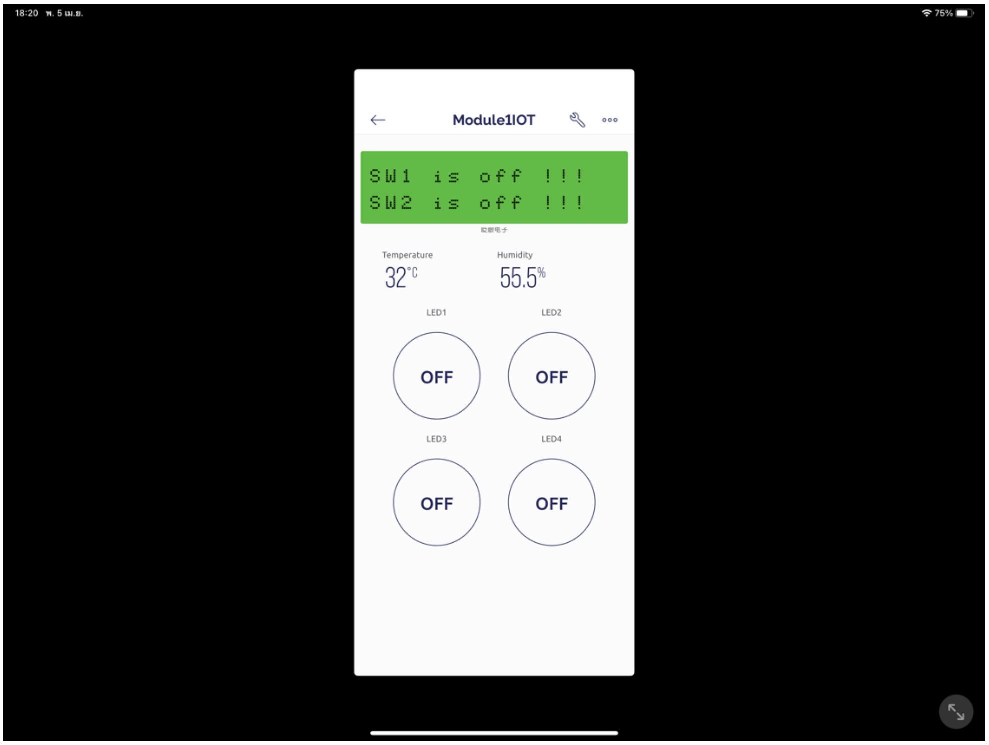


< Test Code >

```
#define BLYNK_PRINT Serial

#define BLYNK_TEMPLATE_ID "TMPL6kynEx4gG"
#define BLYNK_TEMPLATE_NAME "Module1IOT"
#define BLYNK_AUTH_TOKEN "1B5KC50t5IU5qdLvfG4NH1dBMMDG1sp_"

#include <WiFiManager.h> // https://github.com/tzapu/WiFiManager
#include <BlynkSimpleEsp32.h>
#include <DHT.h> //https://www.arduinolibraries.info/libraries/dht-sensor-library
#define DHT_SENSOR_PIN  15 // ESP32 pin GIOP21 connected to DHT22 sensor
#define DHT_SENSOR_TYPE DHT22

DHT dht_sensor(DHT_SENSOR_PIN, DHT_SENSOR_TYPE);

char ssid[50] = ""; // increase the size of the character array to fit the maximum length of a WiFi
SSID
char pass[50] = ""; // increase the size of the character array to fit the maximum length of a WiFi
password


#define LED1 18
#define LED2 19
#define LED3 22
#define LED4 23
int ledState1 = LOW;
int ledState2 = LOW;
int ledState3 = LOW;
int ledState4 = LOW;

#define SW1 4
#define SW2 5


int lastSteadyState_SW1 = LOW;       // the previous steady state from the input pin
int lastSteadyState_SW2 = LOW;       // the previous steady state from the input pin
int lastFlickerableState_SW1 = LOW;  // the previous flickerable state from the input pin
int lastFlickerableState_SW2 = LOW;  // the previous flickerable state from the input pin
int currentState_SW1;                // the current reading from the input pin
int currentState_SW2;                // the current reading from the input pin
int SW1_state = 0;     // the current state of SW
int SW2_state = 0;     // the current state of SW

int n = 1;

#define DEBOUNCE_TIME  50 // the debounce time in millisecond, increase this time if it still chatters
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
```

```
BlynkTimer timer;

BLYNK_WRITE(V18) {
  ledState1 = param.asInt();
  digitalWrite(LED1, ledState1);
}
BLYNK_WRITE(V19) {
  ledState2 = param.asInt();
  digitalWrite(LED2, ledState2);
}
BLYNK_WRITE(V22) {
  ledState3 = param.asInt();
  digitalWrite(LED3, ledState3);
}
BLYNK_WRITE(V23) {
  ledState4 = param.asInt();
  digitalWrite(LED4, ledState4);
}

void sendSensor()
{
  float h = dht_sensor.readHumidity();
  float t = dht_sensor.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
}

//void sendFromBTN()
//{
//
//  if (digitalRead(SW1) == 0) {
//    Blynk.virtualWrite(V7, "0");
//  } else {
//    Blynk.virtualWrite(V7, "1");
//  }
//    if (digitalRead(SW2) == 0) {
//      Blynk.virtualWrite(V8, "0");
//    } else {
//      Blynk.virtualWrite(V8, "1");
//    }
//}

void sendFromSW1() {

  // read the state of the switch/button:
  currentState_SW1 = digitalRead(SW1);

  // If the switch/button changed, due to noise or pressing:
  if (currentState_SW1 != lastFlickerableState_SW1) {
    // reset the debouncing timer
    lastDebounceTime = millis();
    // save the the last flickerable state
    lastFlickerableState_SW1 = currentState_SW1;
  }

  if ((millis() - lastDebounceTime) > DEBOUNCE_TIME) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (lastSteadyState_SW1 == HIGH && currentState_SW1 == LOW) {
      SW1_state = n - SW1_state;
      n = 1;
      Serial.println(SW1_state);
    }

    // save the the last steady state
    lastSteadyState_SW1 = currentState_SW1;
  }
  if (SW1_state == 1) {
    Blynk.virtualWrite(V7, "SW1 is on !!!");
  } else {
    Blynk.virtualWrite(V7, "SW1 is off !!!");
  }
}

void sendFromSW2() {

  // read the state of the switch/button:
  currentState_SW2 = digitalRead(SW2);

  if (currentState_SW2 != lastFlickerableState_SW2) {
    // reset the debouncing timer
    lastDebounceTime = millis();
    // save the the last flickerable state
    lastFlickerableState_SW2 = currentState_SW2;
  }

  if ((millis() - lastDebounceTime) > DEBOUNCE_TIME) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:
    // if the button state has changed:
    if (lastSteadyState_SW2 == HIGH && currentState_SW2 == LOW) {
      SW2_state = n - SW2_state;
      n = 1;
      Serial.println(SW2_state);
    }
    // save the the last steady state
    lastSteadyState_SW2 = currentState_SW2;
  }
  if (SW2_state == 1) {
    Blynk.virtualWrite(V8, "SW2 is on !!!");
  } else {
    Blynk.virtualWrite(V8, "SW2 is off !!!");
  }

}


void setup()
{
  // Debug console
  Serial.begin(9600);

  dht_sensor.begin(); // initialize the DHT sensor
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(SW1, INPUT_PULLUP);
  pinMode(SW2, INPUT_PULLUP);

  ssid[0] = '\0'; // null-terminate the character array to avoid garbage values
  pass[0] = '\0'; // null-terminate the character array to avoid garbage values


  //WiFiManager, Local intialization. Once its business is done, there is no need to keep it around
  WiFiManager wm;
  bool res;
  // res = wm.autoConnect(); // auto generated AP name from chipid
  // res = wm.autoConnect("AutoConnectAP"); // anonymous ap
  res = wm.autoConnect("ESP32AutoConnectAP", "12345678"); // password protected ap

  if (!res) {
    Serial.println("Failed to connect");
    // ESP.restart();
  }
  else {
    //if you get here you have connected to the WiFi
    Serial.println("connected...yeey :)");
    Serial.println(res);
  }

  // use strcpy() to copy the String values into the character arrays
  strcpy(ssid, wm.getWiFiSSID().c_str());
  strcpy(pass, wm.getWiFiPass().c_str());

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
  timer.setInterval(100L, sendFromSW1);
  timer.setInterval(100L, sendFromSW2);

}

void loop()
{
  Blynk.run();
  timer.run();
}
```

รูปหน้าจอ Blynk



18:20 พ. 5 เม.ย. 75%

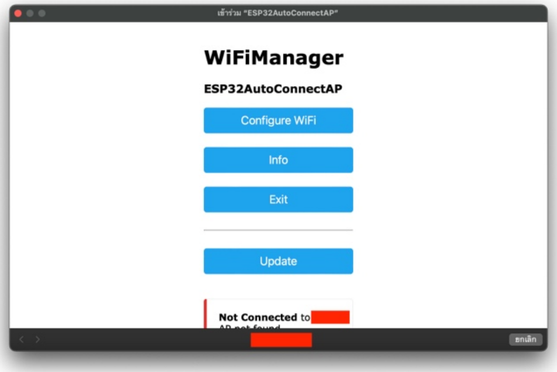
Module1IOT

```
SW1 is off !!!
SW2 is off !!!
```

Temperature 32°C

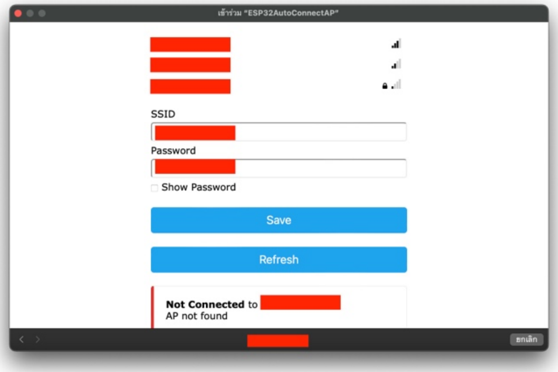
Humidity 55.5%

รูปการต่อวงจร – 1



Not Connected to [redacted]
AP not found

[redacted]



[redacted]

SSID [redacted]

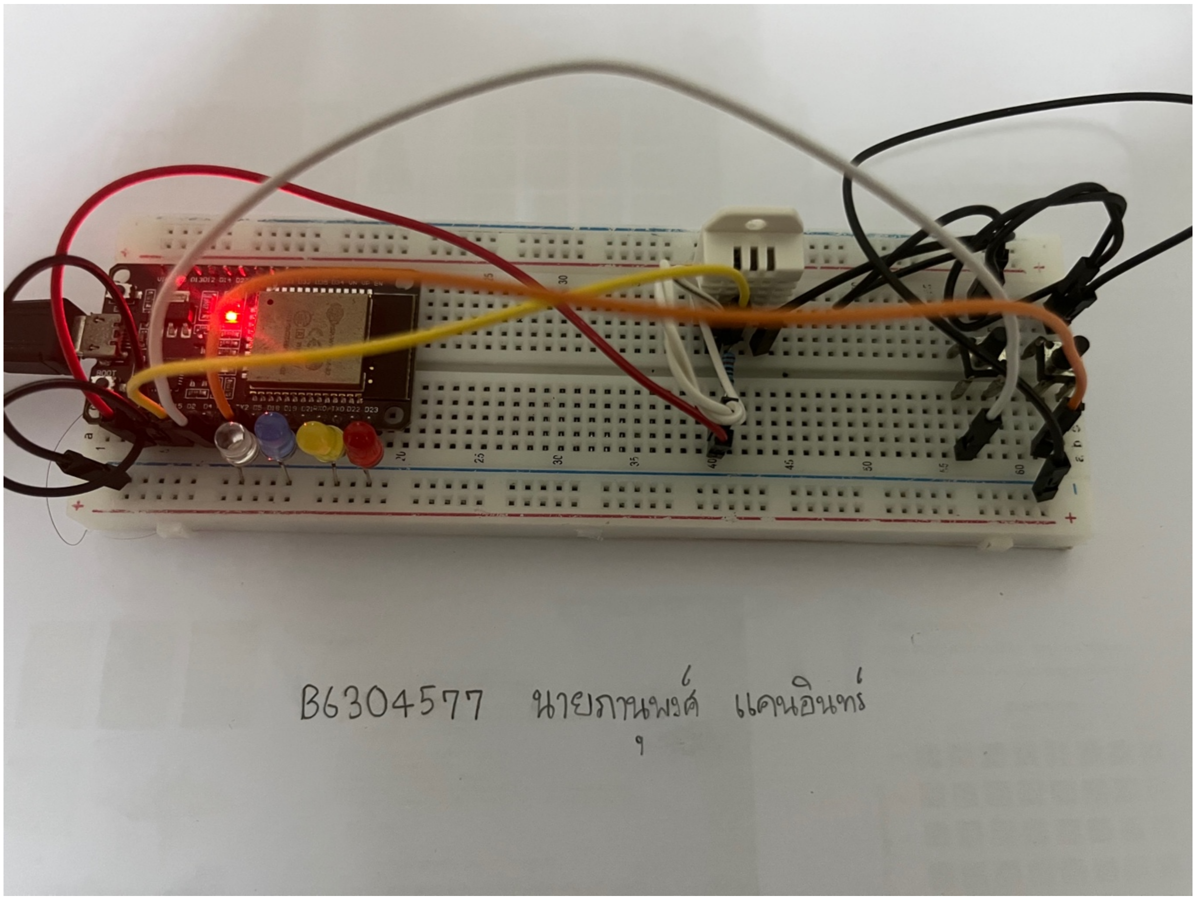
Password [redacted]

□ Show Password

Not Connected to [redacted]
AP not found

[redacted]

รูปการต่อวงจร – 2

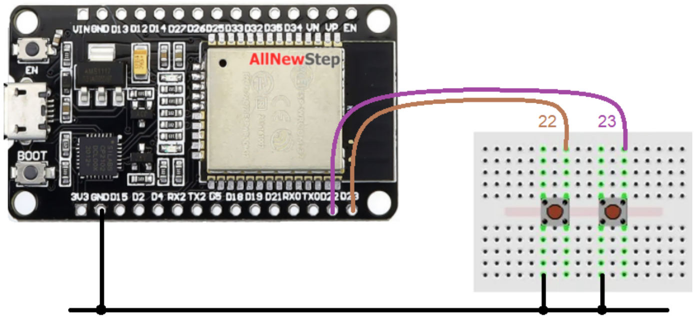


B6304577 นายภานุพงศ์ แคนอินทร์

Quiz_303 – Social Alert

ทดสอบการส่งข้อมูลไป ☐ LINE สำหรับสวิตซ์กด 3 ตัว

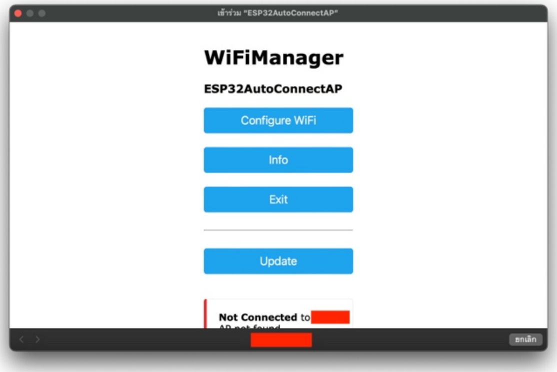
- กดปุ่ม B ที่ต่อกับ ESP32– ให้ส่งข้อความ “Door Open Alarm”
- กดปุ่ม C ที่ต่อกับ ESP32– ให้ส่งข้อความ “Intruders Alarm”


VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
EN
BOOT
AllNewStep
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
22
23

< Test Code >

```
#include <WiFiManager.h> // https://github.com/tzapu/WiFiManager
#include <HTTPClient.h>

#define WebHooksKey "xxxxxx" //Your Webhookskey
#define WebHooksEvent1 "xxxxxx"

#define sw1 4
#define sw2 5

#define LED_Green_Stastus 22
#define LED_Red_Stastus 23

void setup() {
  Serial.begin(115200);
  pinMode(LED_Green_Stastus, OUTPUT);
  pinMode(LED_Red_Stastus, OUTPUT);
  pinMode(sw1, INPUT_PULLUP);
  pinMode(sw2, INPUT_PULLUP);
  WiFiManager wm;
  bool res;
  // res = wm.autoConnect(); // auto generated AP name from chipid
  // res = wm.autoConnect("AutoConnectAP"); // anonymous ap
  res = wm.autoConnect("ESP32AutoConnectAP", "12345678"); // password protected ap

  if (!res) {
    digitalWrite(LED_Red_Stastus, HIGH);
    Serial.println("Failed to connect");
    // ESP.restart();
  }
  else {
    //if you get here you have connected to the WiFi
    Serial.println("connected...yeey :)");
    digitalWrite(LED_Green_Stastus, HIGH);
  }
}

void loop() {

  if (digitalRead(sw1) == LOW) {
    digitalWrite(LED_Green_Stastus, LOW);
    digitalWrite(LED_Red_Stastus, HIGH);
    String serverName = "http://maker.ifttt.com/trigger/" + String(WebHooksEvent1) + "/with/key/" +
String(WebHooksKey);
    String httpRequestData = "value1=" + String("Door Open Alarm!!!");
    Serial.println("Server Name :" + serverName);
    Serial.println("json httpRequestData :" + httpRequestData);
    if (WiFi.status() == WL_CONNECTED)
    { HTTPClient http;
      http.begin(serverName);
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      int httpResponseCode = http.POST(httpRequestData);
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
      if (httpResponseCode == 200)
        Serial.println("Successfully sent");
      else
        Serial.println("Failed!");
    }
    else
    {
      Serial.println("WiFi Disconnected");
    }
    Serial.print(" >> Wait for 5 Sec --> ");
    for (int i = 5; i > 0; i--)
    { Serial.print(i);
      delay(1000);
    }
    digitalWrite(LED_Green_Stastus, HIGH);
    digitalWrite(LED_Red_Stastus, LOW);
    Serial.println(" >> Ready");
  }

  if (digitalRead(sw2) == LOW)
  {
    digitalWrite(LED_Green_Stastus, LOW);
    digitalWrite(LED_Red_Stastus, HIGH);
    String serverName = "http://maker.ifttt.com/trigger/" + String(WebHooksEvent1) + "/with/key/" +
String(WebHooksKey);
    String httpRequestData = "value1=" +  String("Intruders Alarm!!!");
    Serial.println("Server Name :" + serverName);
    Serial.println("json httpRequestData :" + httpRequestData);
    if (WiFi.status() == WL_CONNECTED)
    { HTTPClient http;
      http.begin(serverName);
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      int httpResponseCode = http.POST(httpRequestData);
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
      if (httpResponseCode == 200) Serial.println("Successfully sent");
      else Serial.println("Failed!");
    }
    else
    {
      Serial.println("WiFi Disconnected");
    }
    Serial.print(" >> Wait for 5 Sec --> ");
    for (int i = 5; i > 0; i--)

    { Serial.print(i);
      delay(1000);
    }
    digitalWrite(LED_Green_Stastus, HIGH);
    digitalWrite(LED_Red_Stastus, LOW);
    Serial.println(" >> Ready");
  }
}
```

รูปการต่อวงจร – 1

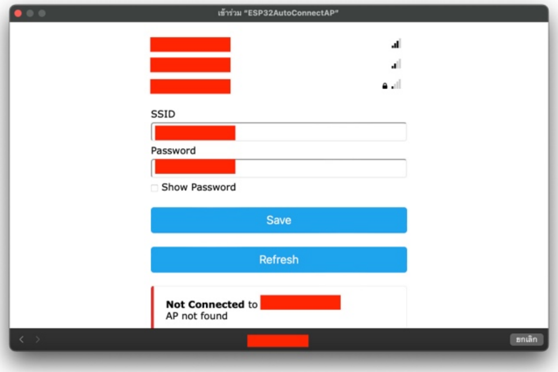


WiFiManager

ESP32AutoConnectAP

Not Connected to [redacted]
AP not found

[redacted]



[redacted]

SSID
[redacted]

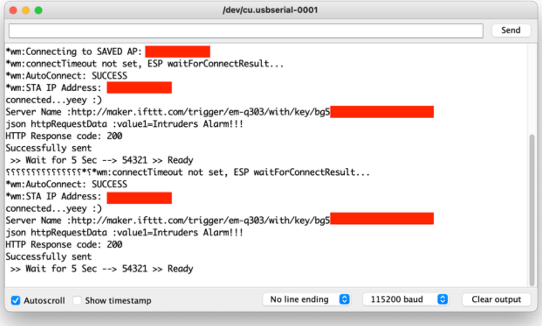
Password
[redacted]

□ Show Password

Not Connected to [redacted]
AP not found

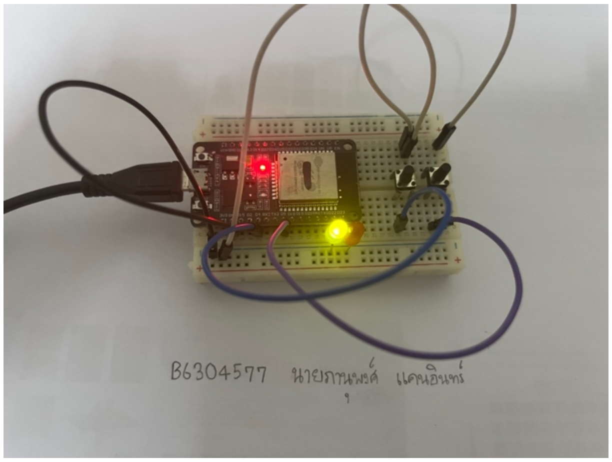
[redacted]

รูปการต่อวงจร – 2

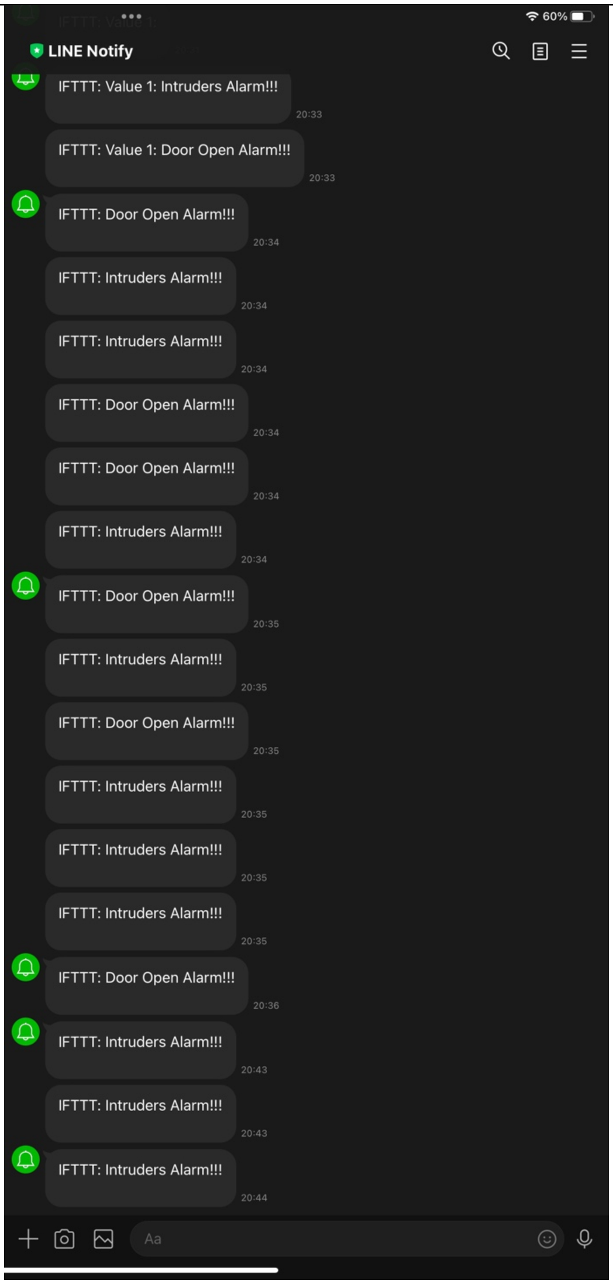


```
*wm:Connecting to SAVED AP: [redacted]
*wm:connectTimeout not set, ESP waitForConnectResult...
*wm:AutoConnect: SUCCESS
*wm:STA IP Address: [redacted]
connected...yeey :)
Server Name :http://maker.ifttt.com/trigger/em-q303/with/key/bg5[redacted]
json httpRequestData :value1=Intruders Alarm!!!
HTTP Response code: 200
Successfully sent
 >> Wait for 5 Sec --> 54321 >> Ready
⸮⸮⸮⸮⸮⸮⸮⸮⸮⸮⸮⸮⸮⸮⸮*⸮*wm:connectTimeout not set, ESP waitForConnectResult...
*wm:AutoConnect: SUCCESS
*wm:STA IP Address: [redacted]
connected...yeey :)
Server Name :http://maker.ifttt.com/trigger/em-q303/with/key/bg5[redacted]
json httpRequestData :value1=Intruders Alarm!!!
HTTP Response code: 200
Successfully sent
 >> Wait for 5 Sec --> 54321 >> Ready
```

รูปการต่อวงจร – 3



รูปหน้าจอ LINE ผลการทดสอบ



IFTTT: Value 1: Intruders Alarm!!! 20:33

IFTTT: Value 1: Door Open Alarm!!! 20:33

IFTTT: Door Open Alarm!!! 20:34

IFTTT: Intruders Alarm!!! 20:34

IFTTT: Intruders Alarm!!! 20:34

IFTTT: Door Open Alarm!!! 20:34

IFTTT: Door Open Alarm!!! 20:34

IFTTT: Intruders Alarm!!! 20:34

IFTTT: Door Open Alarm!!! 20:35

IFTTT: Intruders Alarm!!! 20:35

IFTTT: Door Open Alarm!!! 20:35

IFTTT: Intruders Alarm!!! 20:35

IFTTT: Intruders Alarm!!! 20:35

IFTTT: Intruders Alarm!!! 20:35

IFTTT: Door Open Alarm!!! 20:36

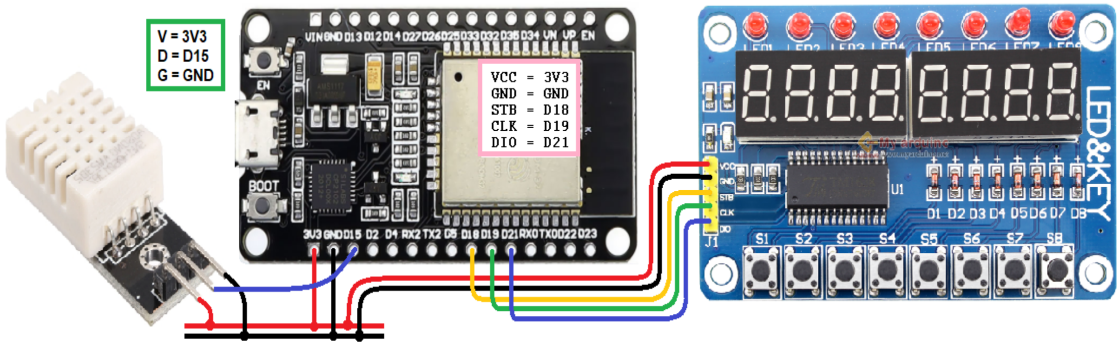
IFTTT: Intruders Alarm!!! 20:43

IFTTT: Intruders Alarm!!! 20:43

IFTTT: Intruders Alarm!!! 20:44

Quiz_304 – Data Logger and Social Alarm

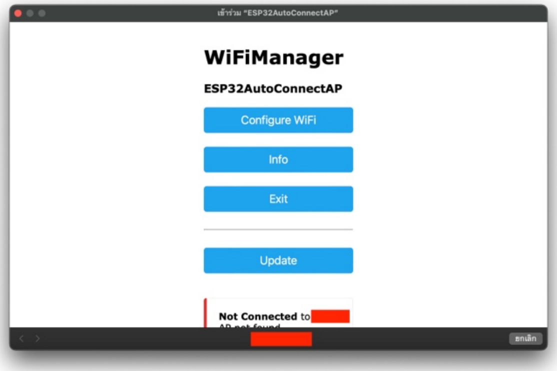
- ส่งข้อมูลอุณหภูมิไปยัง Google Spreadsheet (ทำแล้วในข้อ QB4)
- หากอุณหภูมิที่อ่านได้เกิน 28’C ให้แจ้งเตือนผ่าน ___ และบอกด้วยว่าอุณหภูมิเท่าใด
  □ SMS, □ FB Page, □ FB Massager, □ Twitter, ☑ LINE
- แสดงอุณหภูมิที่ 7_Segment Display TM1638 Board



< Test Code >

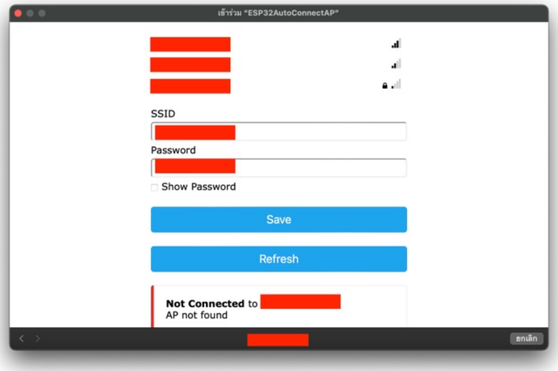
```
#include <WiFiManager.h> // https://github.com/tzapu/WiFiManager
#include <HTTPClient.h>
#include <DHT.h> //https://www.arduinolibraries.info/libraries/dht-sensor-library
#include <TM1638plus.h> [ver 1.9.1]

// Replace with your API endpoint
const char* serverName = "xxxxxx";

#define DHT_SENSOR_PIN  15 // ESP32 pin GIOP15 connected to DHT22 sensor
#define DHT_SENSOR_TYPE DHT22
DHT dht_sensor(DHT_SENSOR_PIN, DHT_SENSOR_TYPE);

#define WebHooksKey "xxxxxx" //Your Webhookskey
#define WebHooksEvent1 "xxxxxx"

#define Brd_STB 18 // strobe  = GPIO connected to str0be line of module
#define Brd_CLK 19 // clock = GPIO connected to clock line of module
#define Brd_DIO 21 // data = GPIO connected to data line of module
bool high_freq = true; // default , if using high freq CPU > 100 MHz set to true

TM1638plus tm(Brd_STB, Brd_CLK, Brd_DIO, high_freq);


void disPlay(float tempp, float humi) {
  if ( isnan(tempp) || isnan(humi)) {
    tm.displayHex(0, 0);
    tm.displayASCIIwDot(1, 0 + '0'); //turn on dot
    tm.displayHex(2, 0);
    tm.display7Seg(3, B01011000);

    tm.displayHex(4, 0);
    tm.displayASCIIwDot(5, 0 + '0'); //turn on dot
    tm.displayHex(6, 0);
    tm.display7Seg(7, B01110100);
  } else {
    tm.displayHex(0, int(tempp / 10));
    tm.displayASCIIwDot(1, int(int(tempp) % 10) + '0'); //turn on dot
    tm.displayHex(2, int(int(tempp * 10)) % 10);
    tm.display7Seg(3, B01011000);

    tm.displayHex(4, int(humi / 10));
    tm.displayASCIIwDot(5, int(int(humi) % 10) + '0'); //turn on dot
    tm.displayHex(6, int(int(humi * 10)) % 10);
    tm.display7Seg(7, B01110100);

  }
}

void sendData2Spreadsheet(float tempp, float humi) {
  // Create JSON object
  String jsonString = " {\"Test1\":\"" + String(tempp) + " ºC\"" + ",\"Test2\":\"" + String(humi) + "
%\"" + "}";
  Serial.print(jsonString);

  // Send HTTP POST request with JSON data
  HTTPClient http;
  http.begin(serverName);
  http.addHeader("Content-Type", "application/json");
  int httpResponseCode = http.POST(jsonString);

  // Check if POST request was successful
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    String response = http.getString();
    Serial.println(response);
  } else {
    Serial.print("Error on HTTP request: ");
    Serial.println(httpResponseCode);
  }

  http.end();
}



void setup() {

  dht_sensor.begin(); // initialize the DHT sensor
  tm.displayBegin(); // initialize the TM1638
  Serial.begin(115200);
  delay(1000);

  //WiFiManager, Local intialization. Once its business is done, there is no need to keep it around
  WiFiManager wm;
  bool res;
  // res = wm.autoConnect(); // auto generated AP name from chipid
  // res = wm.autoConnect("AutoConnectAP"); // anonymous ap
  res = wm.autoConnect("ESP32AutoConnectAP", "12345678"); // password protected ap

  if (!res) {
    Serial.println("Failed to connect");
    // ESP.restart();
  }
  else {
    //if you get here you have connected to the WiFi
    Serial.println("connected...yeey :)");
  }

  delay(1000);
}

void loop() {

  // read humidity
  float humi  = dht_sensor.readHumidity();
  // read temperature in Celsius
  float tempC = dht_sensor.readTemperature();
  // read temperature in Fahrenheit
  float tempF = dht_sensor.readTemperature(true);
  disPlay(tempC, humi);
  sendData2Spreadsheet(tempC, humi);


  if (tempC > 28.00) {
    String serverName = "http://maker.ifttt.com/trigger/" + String(WebHooksEvent1) + "/with/key/" +
String(WebHooksKey);
    String httpRequestData = "value1=" + String(tempC) + String(" ºC High Temperature");
    Serial.println("Server Name :" + serverName);
    Serial.println("json httpRequestData :" + httpRequestData);
    if (WiFi.status() == WL_CONNECTED)
    { HTTPClient http;
      http.begin(serverName);
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      int httpResponseCode = http.POST(httpRequestData);
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
      if (httpResponseCode == 200)
        Serial.println("Successfully sent");
      else
        Serial.println("Failed!");
    }
    else
    {
      Serial.println("WiFi Disconnected");
    }

    Serial.println(" >> Ready");
  }

  delay(5000);
}
```

รูปการต่อวงจร – 1



Not Connected to [redacted]
AP not found

[redacted]



[redacted]

SSID
[redacted]

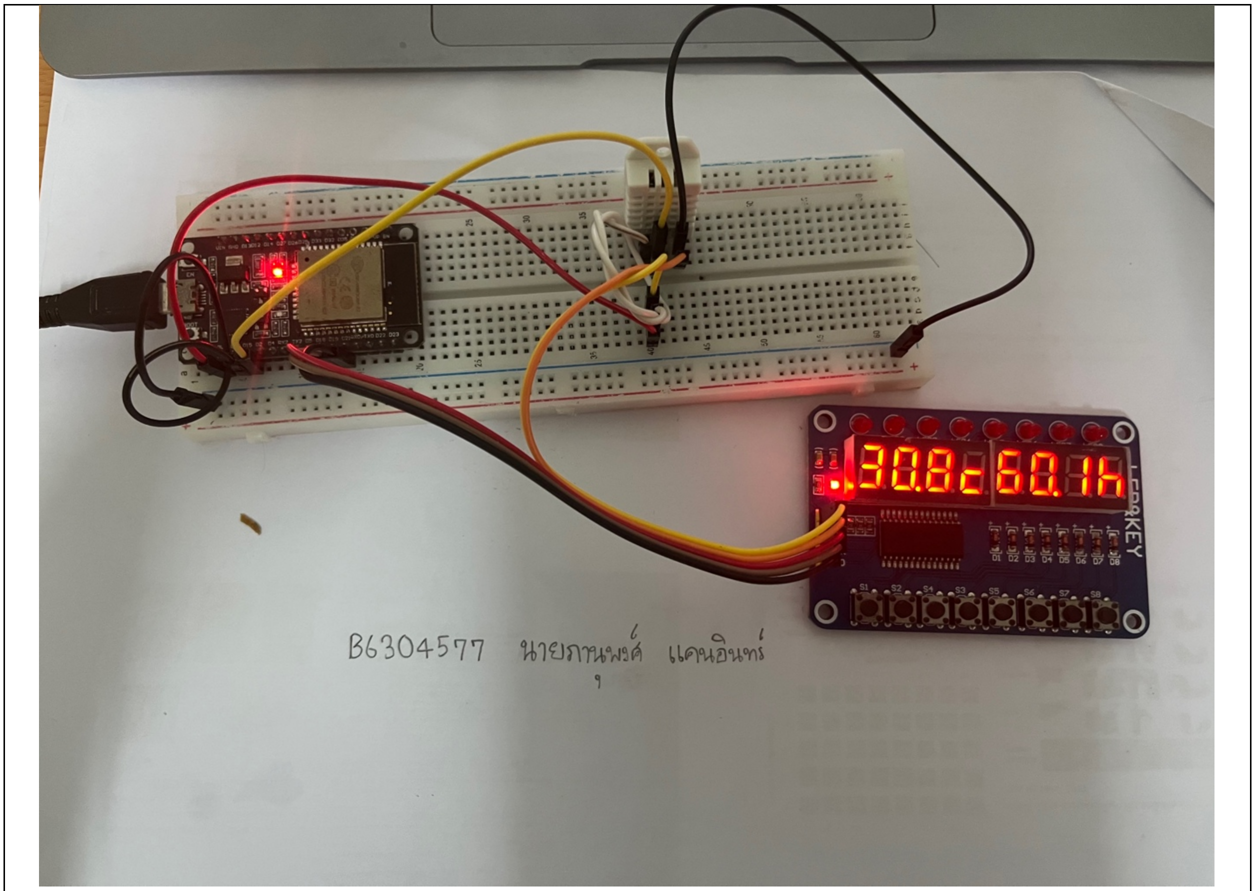
Password
[redacted]

□ Show Password

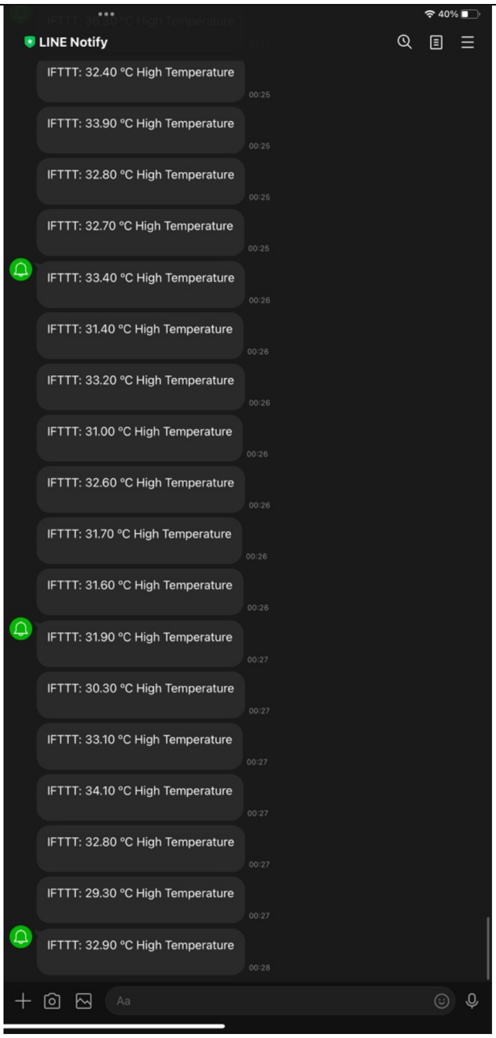
Not Connected to [redacted]
AP not found

[redacted]

รูปการต่อวงจร – 2



รูปหน้าจอ LINE ผลการทดสอบ



IFTTT: 32.40 °C High Temperature 00:25

IFTTT: 33.90 °C High Temperature 00:25

IFTTT: 32.80 °C High Temperature 00:25

IFTTT: 32.70 °C High Temperature 00:25

IFTTT: 33.40 °C High Temperature 00:26

IFTTT: 31.40 °C High Temperature 00:26

IFTTT: 33.20 °C High Temperature 00:26

IFTTT: 31.00 °C High Temperature 00:26

IFTTT: 32.60 °C High Temperature 00:26

IFTTT: 31.70 °C High Temperature 00:26

IFTTT: 31.60 °C High Temperature 00:26

IFTTT: 31.90 °C High Temperature 00:27

IFTTT: 30.30 °C High Temperature 00:27

IFTTT: 33.10 °C High Temperature 00:27

IFTTT: 34.10 °C High Temperature 00:27

IFTTT: 32.80 °C High Temperature 00:27

IFTTT: 29.30 °C High Temperature 00:27

IFTTT: 32.90 °C High Temperature 00:28