การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร
M2M - Intelligence Machine Control

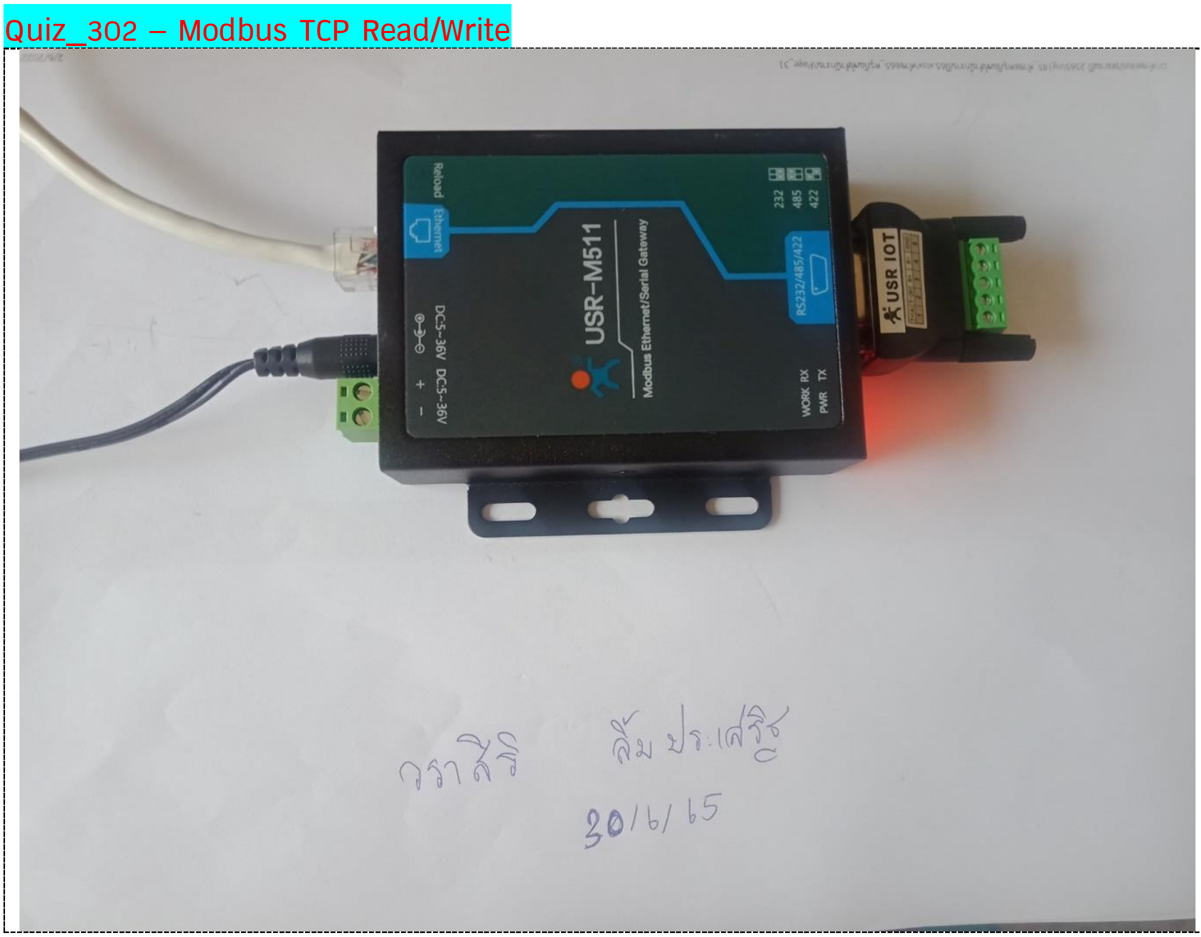
ชื่อ-สกุล : วราสิริ ลิ้มประเสริฐ B6214005

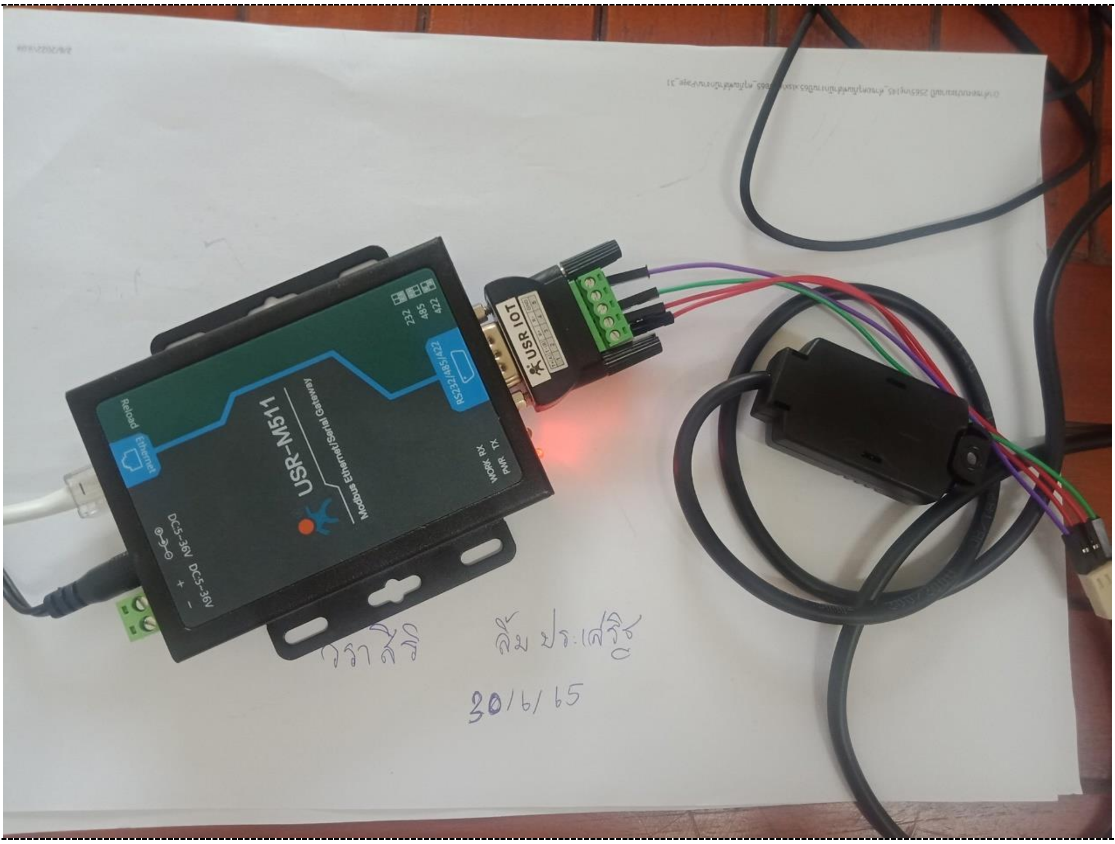
5/5: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_301 – Start SCADA

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
รายยละเอียดการทดสอบ
< โปรแกรมทดสอบ >
< ผลการทดสอบ >

Quiz_302 – Modbus TCP Read/Write



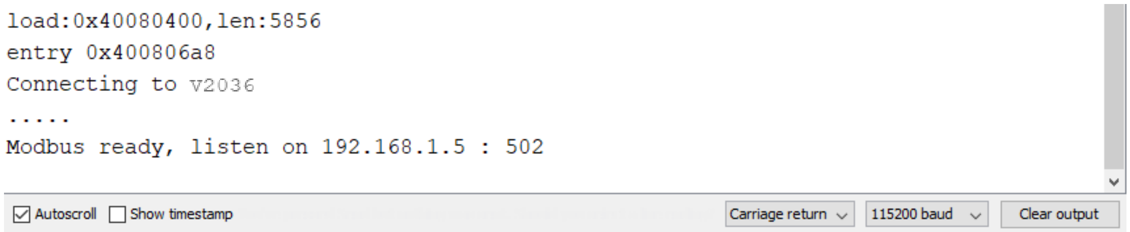


รายยละเอียดการทดสอบ

< โปรแกรมทดสอบ >

```
// https://github.com/yaacov/ArduinoModbusSlave
#include <WiFi.h>
#include <ModbusSlaveTCP.h>
const char* ssid = "V2036";
const char* pass = "fnafchica";
#define SLAVE_ID 3
ModbusTCP slave(SLAVE_ID);
void setup() {
  Serial.begin(115200);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  slave.cbVector[CB_WRITE_COIL] = writeDigitlOut;
  slave.cbVector[CB_READ_COILS] = readDigitalIn;
  slave.cbVector[CB_READ_REGISTERS] = readAnalogIn;
  slave.begin();
  Serial.println("");
  Serial.print("Modbus ready, listen on ");
  Serial.print(WiFi.localIP());
  Serial.println(" : 502");
}
void loop() {
  slave.poll();
}
/**
  Handel Force Single Coil (FC=05)
  set digital output pins (coils) on and off
*/
void writeDigitlOut(uint8_t fc, uint16_t address, uint16_t status) {
  pinMode(address, OUTPUT);
  digitalWrite(address, status);
  Serial.println("digitalWrite(" + String(address) + "," + String(status) + ")");
}
/**
  Handel Read Input Status (FC=02/01)
  write back the values from digital in pins (input status).
  handler functions must return void and take:
  uint8_t fc - function code
  uint16_t address - first register/coil address
  uint16_t length/status - length of data / coil status
*/
void readDigitalIn(uint8_t fc, uint16_t address, uint16_t length) {
  // read digital input
  for (int i = 0; i < length; i++) {
    pinMode(address + i, INPUT_PULLUP);
    int dValue = digitalRead(address + i);
    slave.writeCoilToBuffer(i, dValue);
    Serial.println("digitalRead(" + String(address + i) + ") = " + String(dValue));
  }
}
/**
  Handel Read Input Registers (FC=04/03)
  write back the values from analog in pins (input registers).
*/
void readAnalogIn(uint8_t fc, uint16_t address, uint16_t length) {
  // read analog input
  for (int i = 0; i < length; i++) {
    //int aValue = analogRead(address + i);
    int aValue = (address + i) * 1000 + random(111, 999);
    Serial.println("analogRead(" + String(address + i) + ") = " + String(aValue));
    slave.writeRegisterToBuffer(i, aValue);
  }
}
```

< ผลการทดสอบ >

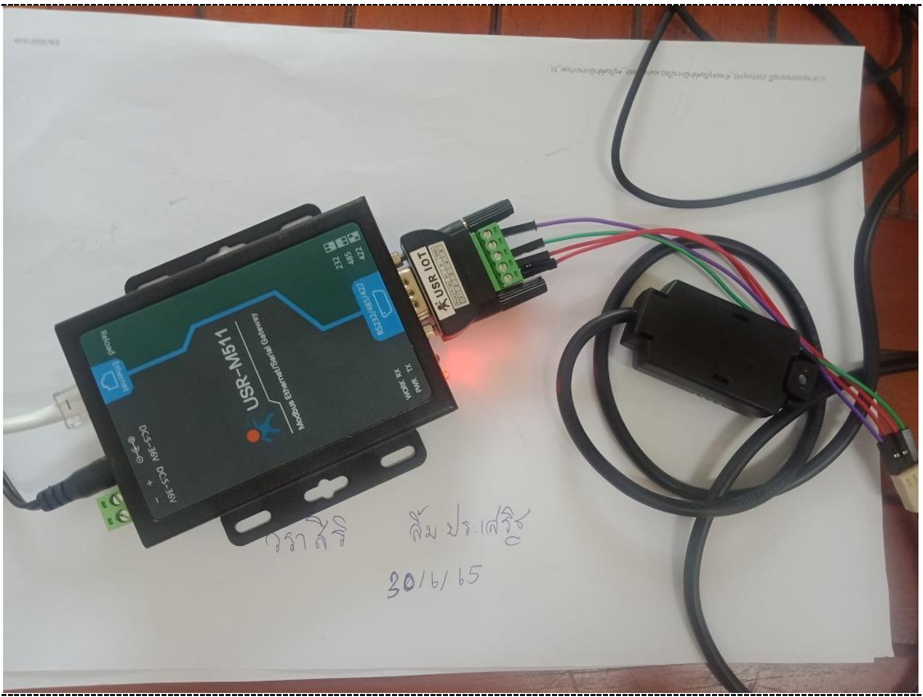


```
load:0x40080400,len:5856
entry 0x400806a8
Connecting to v2036
.....
Modbus ready, listen on 192.168.1.5 : 502
```

☑ Autoscroll ☐ Show timestamp

Carriage return | 115200 baud

Quiz_303 – Modbus RTU/ASCII/TCP with IoTs

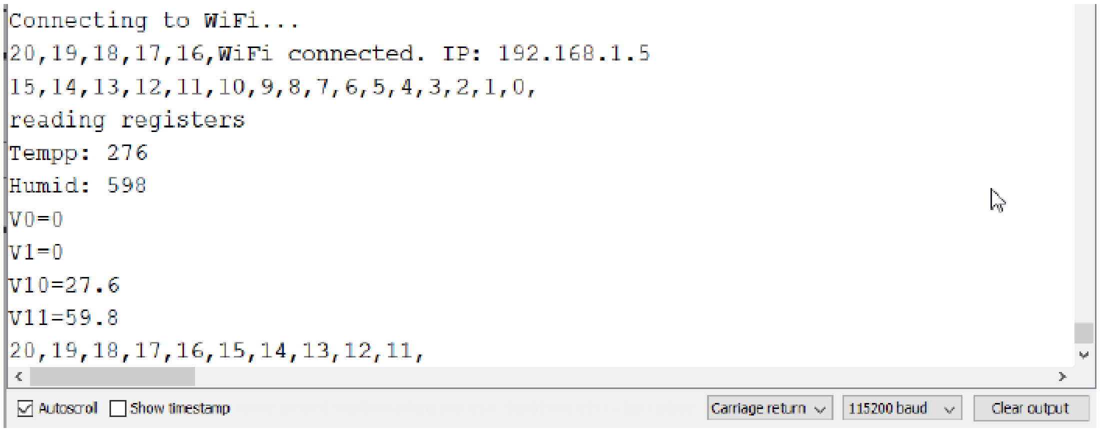


< โปรแกรมทดสอบ >

```
// esp32ModbusTCP >> https://github.com/bertmelis/esp32ModbusTCP
// AsyncTCP.h >> https://github.com/me-no-dev/AsyncTCP
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Arduino.h>
#include <esp32ModbusTCP.h>
char ssid[] = "Mue.Home";
char pass[] = "pk1212312121";
char auth[] = "YD3FmnLEk5vdhs-BeQlWwrACl8gXNgXK";
bool WiFiConnected = false;
int Value_V0, Value_V1;
esp32ModbusTCP sunnyboy(1, {192, 168, 1, 4}, 502);
enum smaType {
  ENUM, // enumeration
  UFIX0, // unsigned 2 Byte, no decimals
  SFIX0, // signed 4 Byte, no decimals
};
struct smaData {
  const char* name;
  uint16_t address;
  uint16_t length;
  smaType type;
  uint16_t packetId;
};
smaData smaRegisters[] = {
  "Tempp", 0, 1, UFIX0, 0,
  "Humid", 1, 1, UFIX0, 0
};
uint8_t numberSmaRegisters = sizeof(smaRegisters) / sizeof(smaRegisters[0]);
uint8_t currentSmaRegister = 0;
```

```
uint16_t ResultData[3];
BLYNK_WRITE(V0) {
  int temp = param.asInt();
  if (temp != Value_V0) {
    Value_V0 = temp;
    RelayControl(801 + temp * 10);
  }
}
BLYNK_WRITE(V1) {
  int temp = param.asInt();
  if (temp != Value_V1) {
    Value_V1 = temp;
    RelayControl(802 + temp * 10);
  }
}
void RelayControl(int Code) {
  Serial.println("Code is = " + String(Code));
}
void setup() {
  Serial.begin(115200);
  WiFi.disconnect(true); // delete old config
  sunnyboy.onData([](uint16_t packet, uint8_t slave, esp32Modbus::FunctionCode fc , uint8_t* data , uint16_t len) {
    for (uint8_t i = 0; i < numberSmaRegisters; ++i) {
      if (smaRegisters[i].packetId == packet) {
        smaRegisters[i].packetId = 0;
        switch (smaRegisters[i].type) {
          case ENUM:
          case UFIX0: {
              uint32_t value = 0; // 2-Byte Data
              value = (data[0] << 8) | (data[1]); // 2-Byte Data
              Serial.printf("%s: %u\n", smaRegisters[i].name, value);
              ResultData[i] = value;
              break;
            }
          case SFIX0: {
              int32_t value = 0;
              value = (data[0] << 24) | (data[1] << 16) | (data[2] << 8) | (data[3]);
              Serial.printf("%s: %i\n", smaRegisters[i].name, value);
              break;
            }
        }
        return;
      }
    }
  });
  sunnyboy.onError([](uint16_t packet, esp32Modbus::Error e) {
    Serial.printf("Error packet %u: %02x\n", packet, e);
  });
  delay(1000);
  WiFi.onEvent([](WiFiEvent_t event, WiFiEventInfo_t info) {
    Serial.print("WiFi connected. IP: ");
    Serial.println(IPAddress(info.got_ip.ip_info.ip.addr));
    WiFiConnected = true;
  }, WiFiEvent_t::SYSTEM_EVENT_STA_GOT_IP);
  WiFi.onEvent([](WiFiEvent_t event, WiFiEventInfo_t info) {
    Serial.print("WiFi lost connection. Reason: ");
    Serial.println(info.disconnected.reason);
    WiFi.disconnect();
    WiFiConnected = false;
  }, WiFiEvent_t::SYSTEM_EVENT_STA_DISCONNECTED);
  WiFi.begin(ssid, pass);
  Serial.println();
  Serial.println("Connecting to WiFi... ");
}
int loopCont = 20;
void loop() {
  if (loopCont < 0 && WiFiConnected) {
    loopCont = 20;
    Serial.print("\nreading registers\n");
    for (uint8_t i = 0; i < numberSmaRegisters; ++i) {
      uint16_t packetId = sunnyboy.readHoldingRegisters(smaRegisters[i].address, smaRegisters[i].length);
      if (packetId > 0) {
        smaRegisters[i].packetId = packetId;
      } else {
        Serial.print("reading error\n");
      }
    }
    delay(5000);
    //Blynk.config(auth);
    float CTempp = ResultData[0] / 10.0;
    float Hudmid = ResultData[1] / 10.0;
    Blynk.virtualWrite(V10, CTempp);
    Blynk.virtualWrite(V11, Hudmid);
    Serial.println("V0=" + String(Value_V0));
    Serial.println("V1=" + String(Value_V1));
    Serial.println("V10=" + String(CTempp, 1));
    Serial.println("V11=" + String(Hudmid, 1));
  }
  Serial.print(String(loopCont--) + ",");
  //Blynk.run();
  delay(500);
}
```

< ผลการทดสอบ >



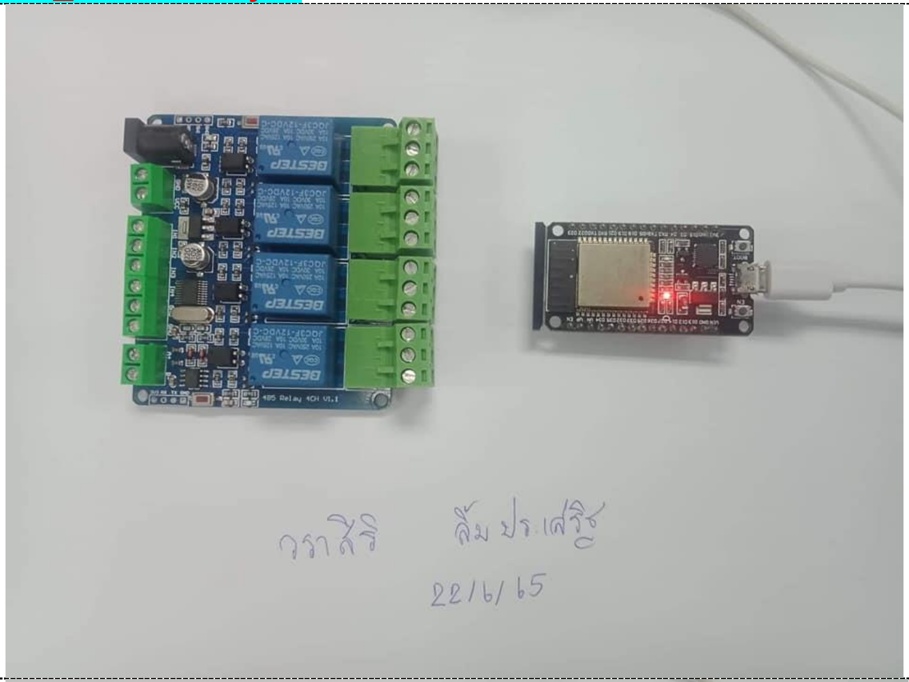
```
Connecting to WiFi...
20,19,18,17,16,WiFi connected. IP: 192.168.1.5
15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,
reading registers
Tempp: 276
Humid: 598
V0=0
V1=0
V10=27.6
V11=59.8
20,19,18,17,16,15,14,13,12,11,
```

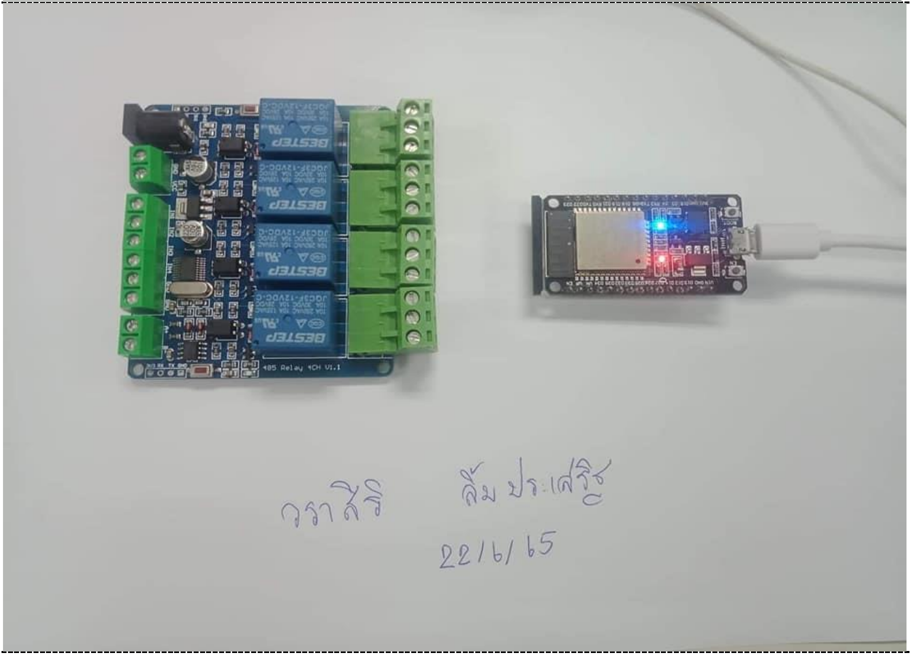
☑ Autoscroll □ Show timestamp Carriage return 115200 baud

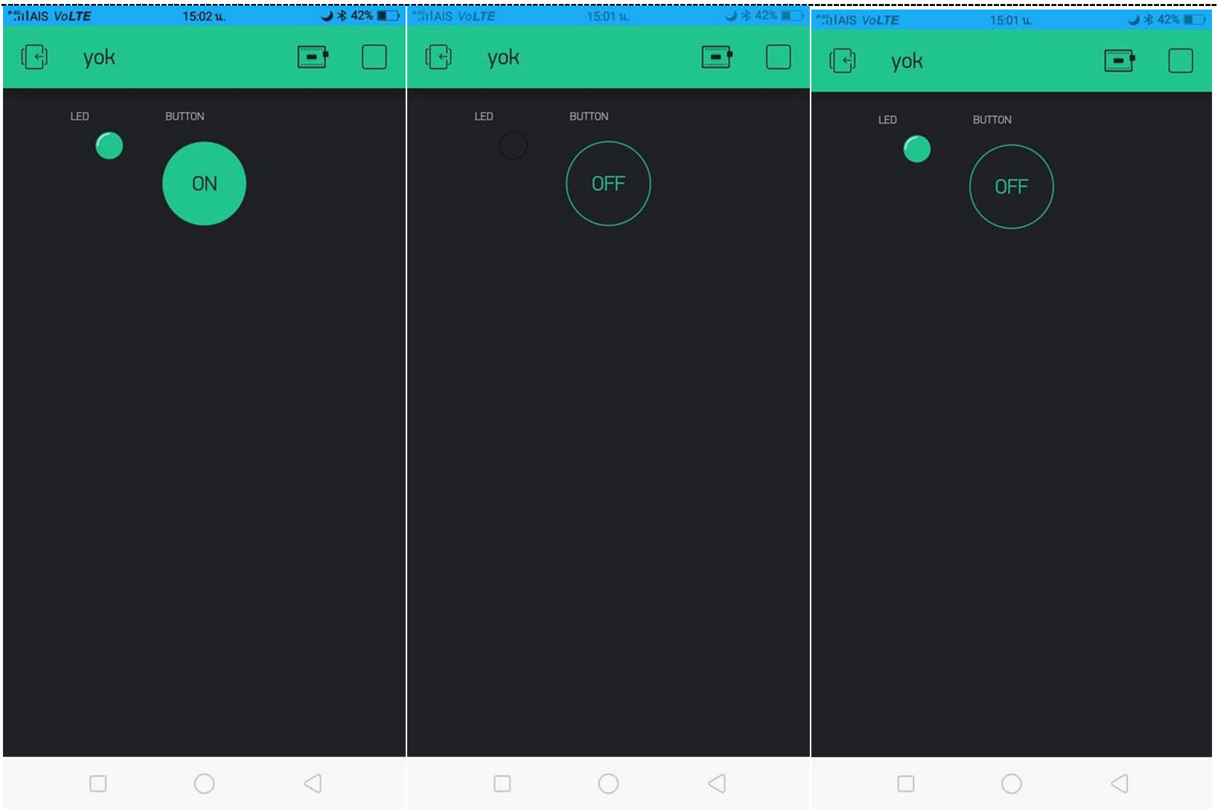
การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร M2M - Intelligence Machine Control
ชื่อ-สกุล : วราสิริ ลิ้มประเสริฐ B6214005

4/4: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

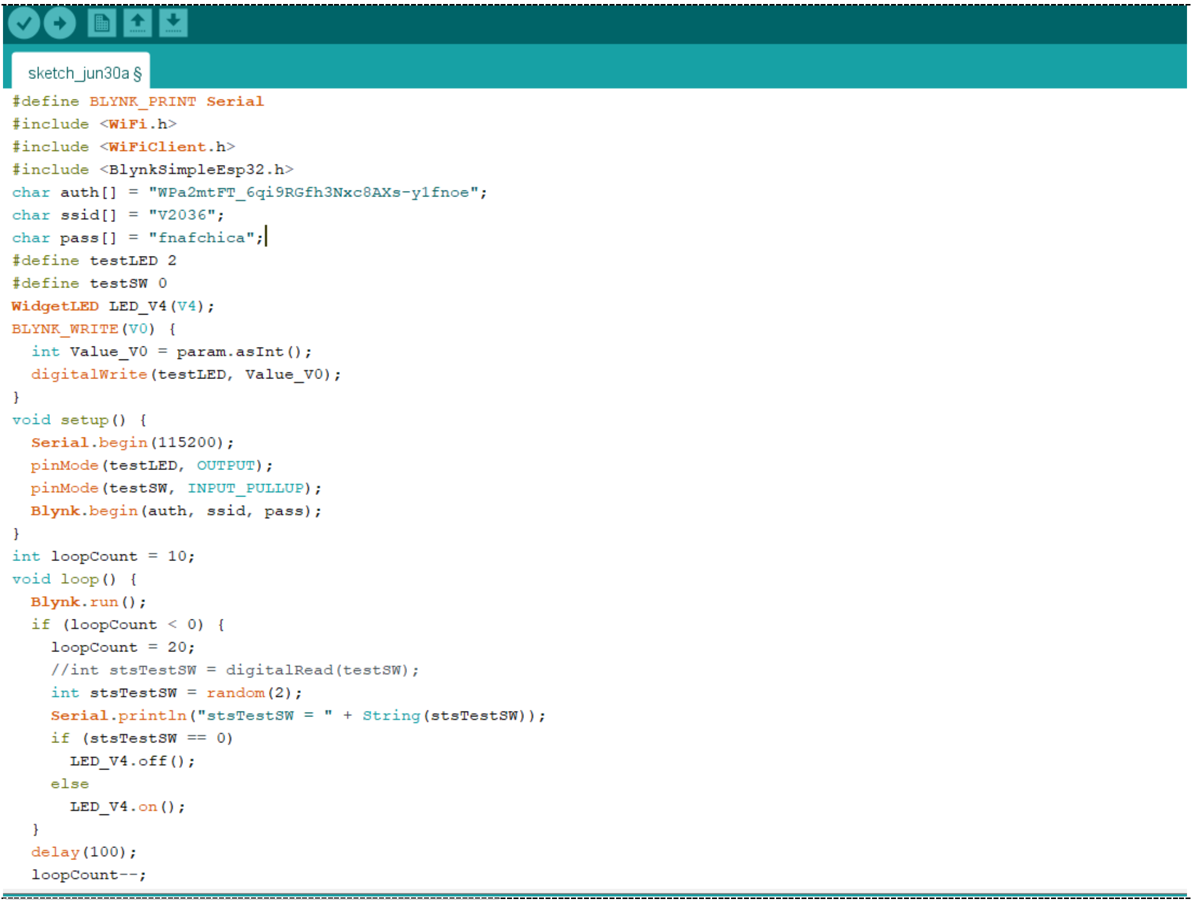
Quiz_401 – test Blynk



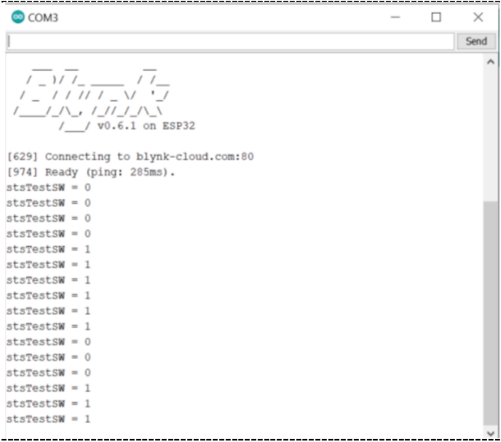




```
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = "WPa2mtFT_6qi9RGfh3Nxc8AXs-y1fnoe";
char ssid[] = "V2036";
char pass[] = "fnafchica";
#define testLED 2
#define testSW 0
WidgetLED LED_V4(V4);
BLYNK_WRITE(V0) {
  int Value_V0 = param.asInt();
  digitalWrite(testLED, Value_V0);
}
void setup() {
  Serial.begin(115200);
  pinMode(testLED, OUTPUT);
  pinMode(testSW, INPUT_PULLUP);
  Blynk.begin(auth, ssid, pass);
}
int loopCount = 10;
void loop() {
  Blynk.run();
  if (loopCount < 0) {
    loopCount = 20;
    //int stsTestSW = digitalRead(testSW);
    int stsTestSW = random(2);
    Serial.println("stsTestSW = " + String(stsTestSW));
    if (stsTestSW == 0)
      LED_V4.off();
    else
      LED_V4.on();
  }
  delay(100);
  loopCount–;
}
```

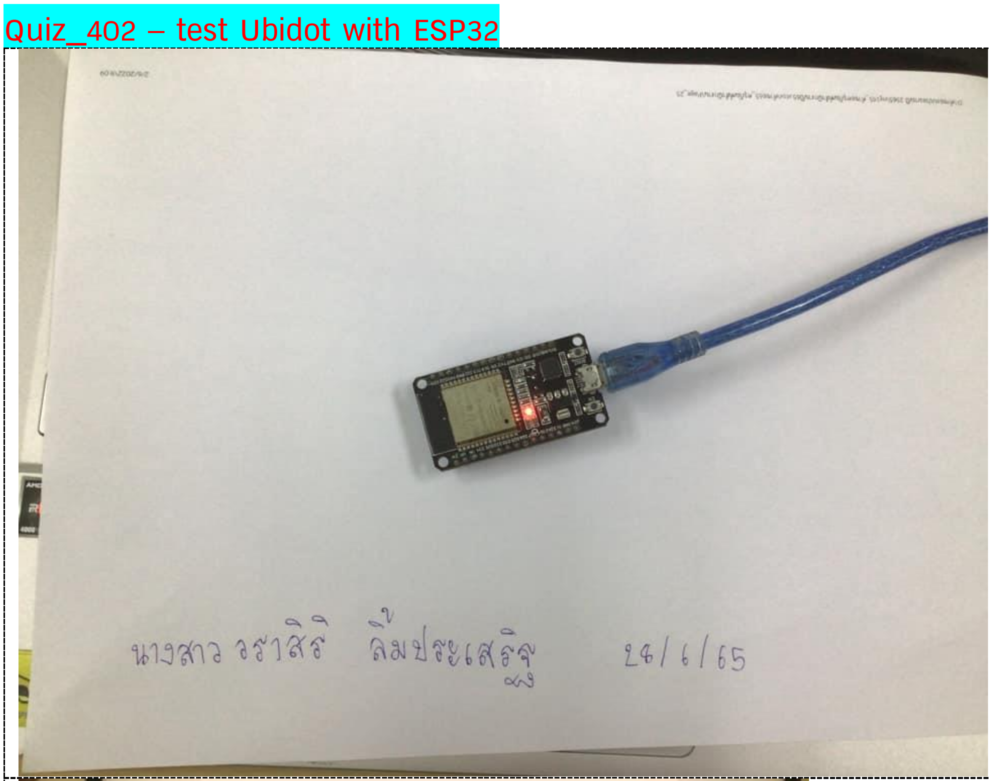


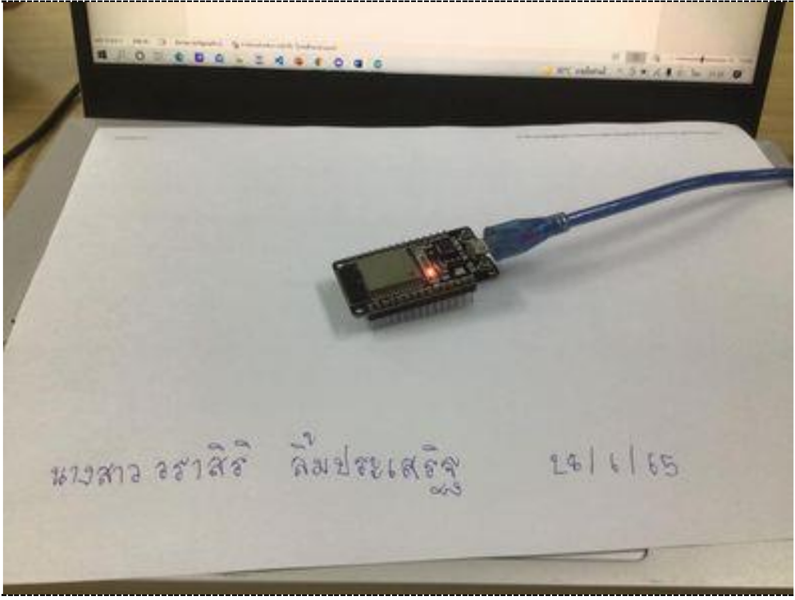
```
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = "WPa2mtFT_6qi9RGfh3Nxc8AXs-y1fnoe";
char ssid[] = "V2036";
char pass[] = "fnafchica";
#define testLED 2
#define testSW 0
WidgetLED LED_V4(V4);
BLYNK_WRITE(V0) {
  int Value_V0 = param.asInt();
  digitalWrite(testLED, Value_V0);
}
void setup() {
  Serial.begin(115200);
  pinMode(testLED, OUTPUT);
  pinMode(testSW, INPUT_PULLUP);
  Blynk.begin(auth, ssid, pass);
}
int loopCount = 10;
void loop() {
  Blynk.run();
  if (loopCount < 0) {
    loopCount = 20;
    //int stsTestSW = digitalRead(testSW);
    int stsTestSW = random(2);
    Serial.println("stsTestSW = " + String(stsTestSW));
    if (stsTestSW == 0)
      LED_V4.off();
    else
      LED_V4.on();
  }
  delay(100);
  loopCount--;
```



```
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v0.6.1 on ESP32

[629] Connecting to blynk-cloud.com:80
[974] Ready (ping: 285ms).
stsTestSW = 0
stsTestSW = 0
stsTestSW = 0
stsTestSW = 0
stsTestSW = 1
stsTestSW = 1
stsTestSW = 1
stsTestSW = 1
stsTestSW = 1
stsTestSW = 1
stsTestSW = 0
stsTestSW = 0
stsTestSW = 0
stsTestSW = 1
stsTestSW = 1
stsTestSW = 1
```

Quiz_402 – test Ubidot with ESP32

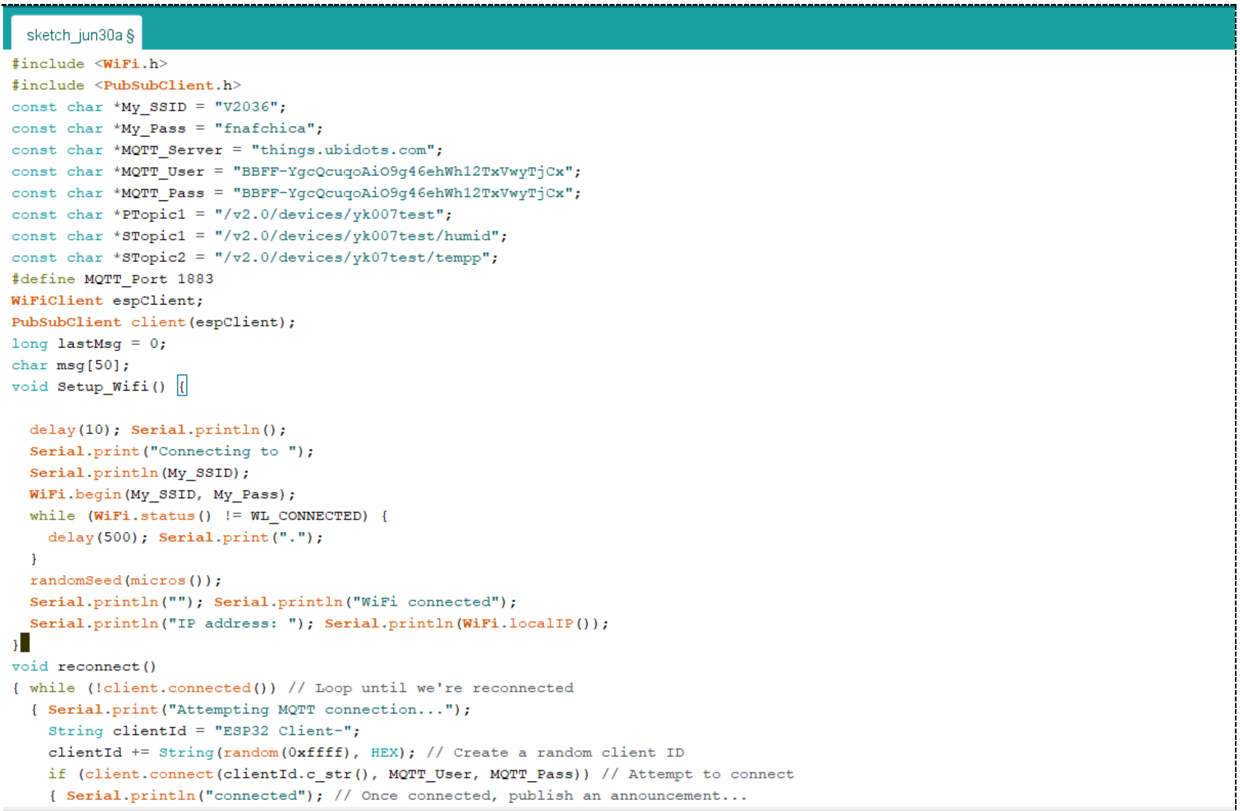




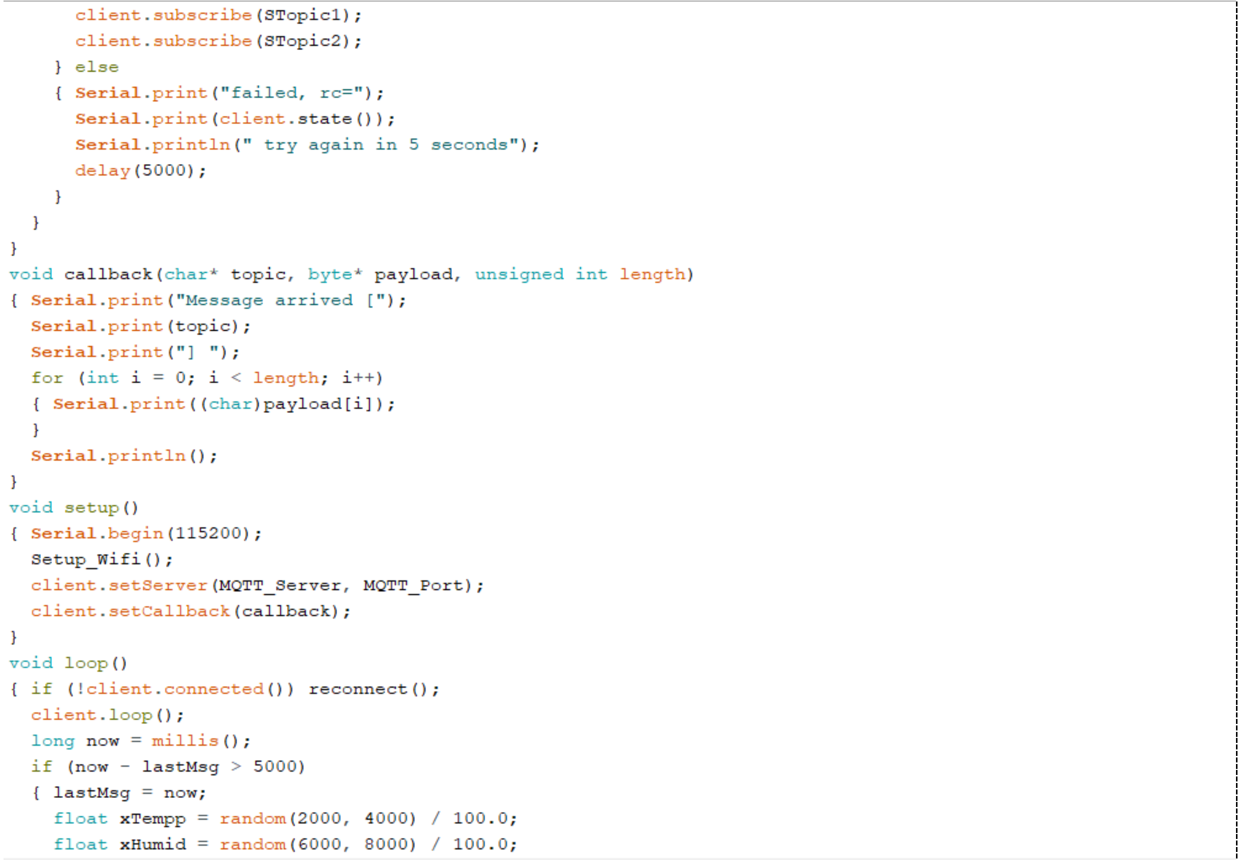
```
#include <WiFi.h>
#include <PubSubClient.h>
const char *My_SSID = "V2036";
const char *My_Pass = "fnafchica";
const char *MQTT_Server = "things.ubidots.com";
const char *MQTT_User = "BBFF-YgcQcuqoAiO9g46ehWh12TxVwyTjCx";
const char *MQTT_Pass = "BBFF-YgcQcuqoAiO9g46ehWh12TxVwyTjCx";
const char *PTopic1 = "/v2.0/devices/yk007test";
const char *STopic1 = "/v2.0/devices/yk007test/humid";
const char *STopic2 = "/v2.0/devices/yk007test/tempp";
#define MQTT_Port 1883
```

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

  delay(10); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(My_SSID);
  WiFi.begin(My_SSID, My_Pass);
  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(), MQTT_User, MQTT_Pass)) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
      client.subscribe(STopic1);
      client.subscribe(STopic2);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
void callback(char* topic, byte* payload, unsigned int length)
{ Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
  }
  Serial.println();
}
void setup()
{ Serial.begin(115200);
  Setup_Wifi();
  client.setServer(MQTT_Server, MQTT_Port);
  client.setCallback(callback);
}
void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 5000)
  { lastMsg = now;
    float xTempp = random(2000, 4000) / 100.0;
    float xHumid = random(6000, 8000) / 100.0;
```

```
   snprintf (msg, 75, "{ \"humid\" : %5.2f, \"tempp\": %5.2f}", xHumid, xTempp);
   Serial.print("Publish message: ");
   Serial.println(msg);
   client.publish(PTopic1, msg);
 }
```

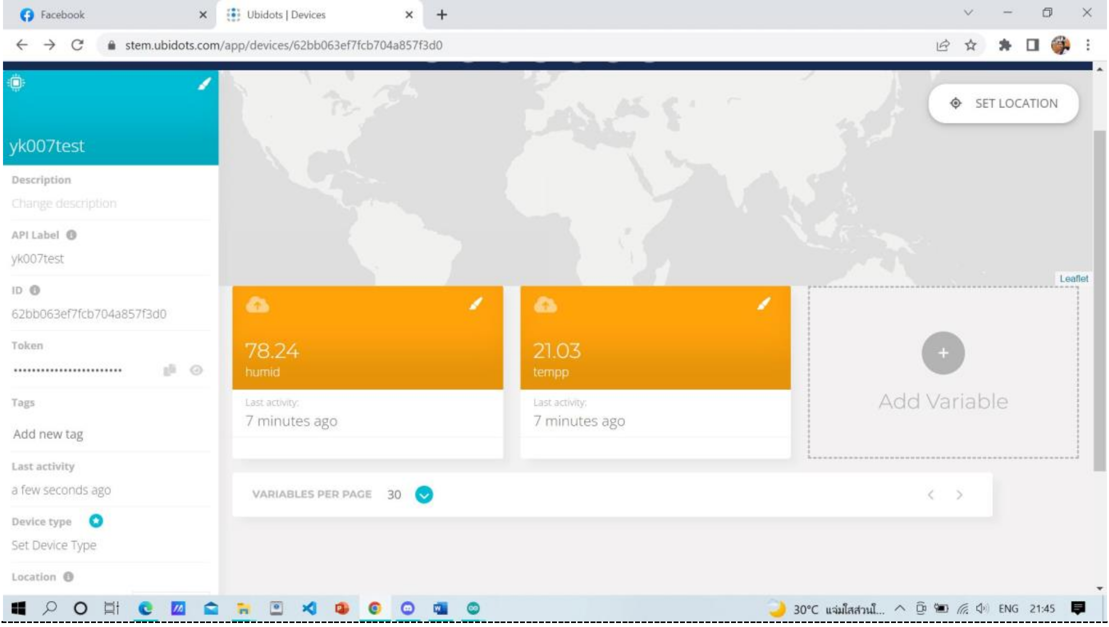


```
#include <WiFi.h>
#include <PubSubClient.h>
const char *My_SSID = "V2036";
const char *My_Pass = "fnafchica";
const char *MQTT_Server = "things.ubidots.com";
const char *MQTT_User = "BBFF-YgcQcuqoAiO9g46ehWh12TxVwyTjCx";
const char *MQTT_Pass = "BBFF-YgcQcuqoAiO9g46ehWh12TxVwyTjCx";
const char *PTopic1 = "/v2.0/devices/yk007test";
const char *STopic1 = "/v2.0/devices/yk007test/humid";
const char *STopic2 = "/v2.0/devices/yk07test/tempp";
#define MQTT_Port 1883
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
void Setup_Wifi() {

  delay(10); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(My_SSID);
  WiFi.begin(My_SSID, My_Pass);
  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(), MQTT_User, MQTT_Pass)) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
```



```
      client.subscribe(STopic1);
      client.subscribe(STopic2);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
void callback(char* topic, byte* payload, unsigned int length)
{ Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
  }
  Serial.println();
}
void setup()
{ Serial.begin(115200);
  Setup_Wifi();
  client.setServer(MQTT_Server, MQTT_Port);
  client.setCallback(callback);
}
void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 5000)
  { lastMsg = now;
    float xTempp = random(2000, 4000) / 100.0;
    float xHumid = random(6000, 8000) / 100.0;
    snprintf (msg, 75, "{ \"humid\" : %5.2f, \"tempp\": %5.2f}", xHumid, xTempp);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(PTopic1, msg);
  }
}
```

< ผลการทดสอบ >



yk007test

Description
Change description

API Label
yk007test

ID
62bb063ef7fcb704a857f3d0

Token
••••••••••••••••••••••••

Tags
Add new tag

Last activity
a few seconds ago

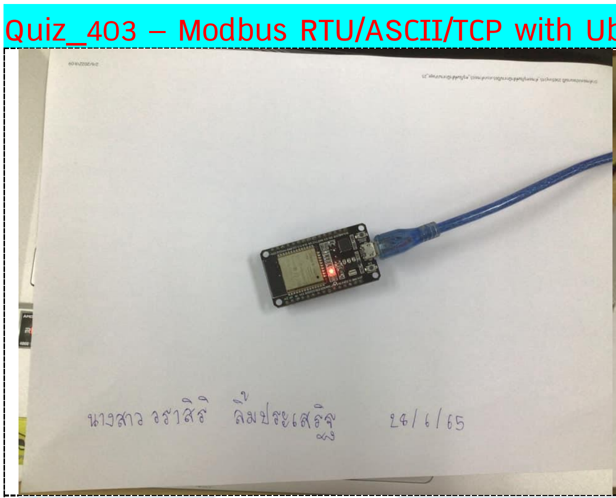
Device type
Set Device Type

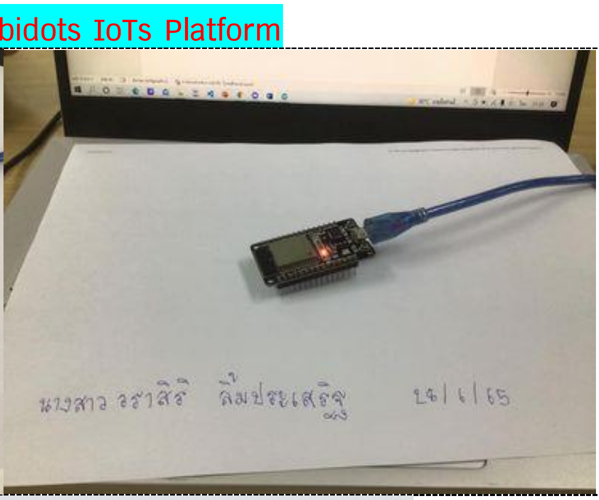
Location

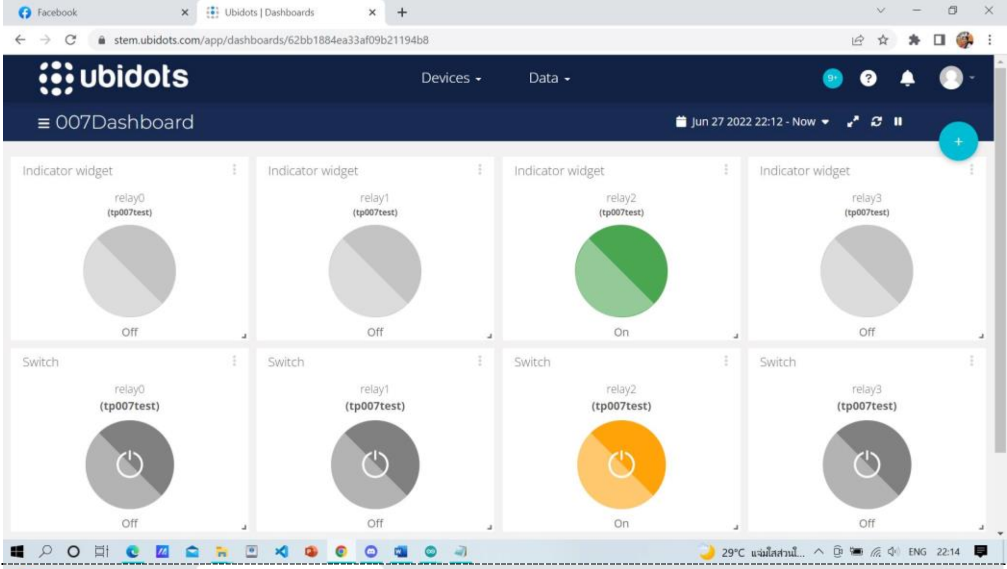
78.24
humid
Last activity:
7 minutes ago

21.03
tempp
Last activity:
7 minutes ago

Quiz_403 – Modbus RTU/ASCII/TCP with Ubidots IoTs Platform






Facebook | Ubidots | Dashboards

stem.ubidots.com/app/dashboards/62bb1884ea33af09b21194b8

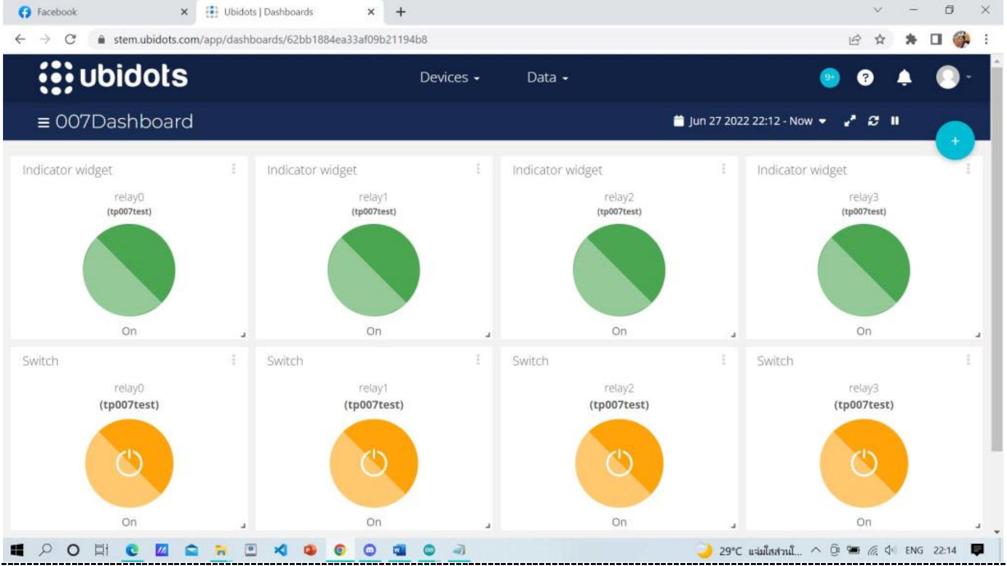
ubidots — Devices — Data

≡ 007Dashboard — Jun 27 2022 22:12 - Now

Indicator widget	Indicator widget	Indicator widget	Indicator widget
relay0 (tp007test)	relay1 (tp007test)	relay2 (tp007test)	relay3 (tp007test)
Off	Off	On	Off

Switch	Switch	Switch	Switch
relay0 (tp007test)	relay1 (tp007test)	relay2 (tp007test)	relay3 (tp007test)
Off	Off	On	Off

29°C ENG 22:14


Facebook | Ubidots | Dashboards

stem.ubidots.com/app/dashboards/62bb1884ea33af09b21194b8

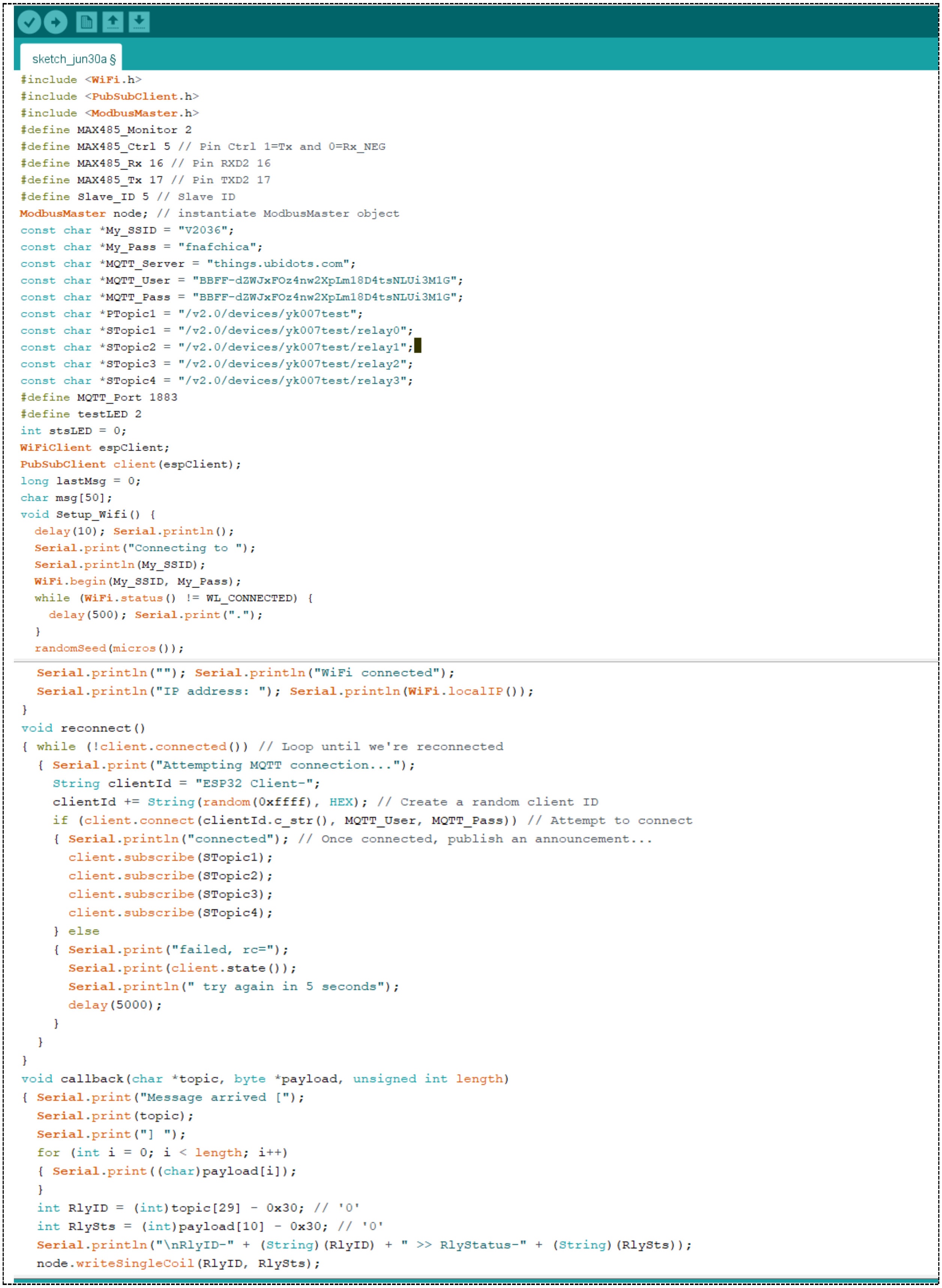
ubidots — Devices — Data

≡ 007Dashboard — Jun 27 2022 22:12 - Now

Indicator widget	Indicator widget	Indicator widget	Indicator widget
relay0 (tp007test)	relay1 (tp007test)	relay2 (tp007test)	relay3 (tp007test)
On	On	On	On

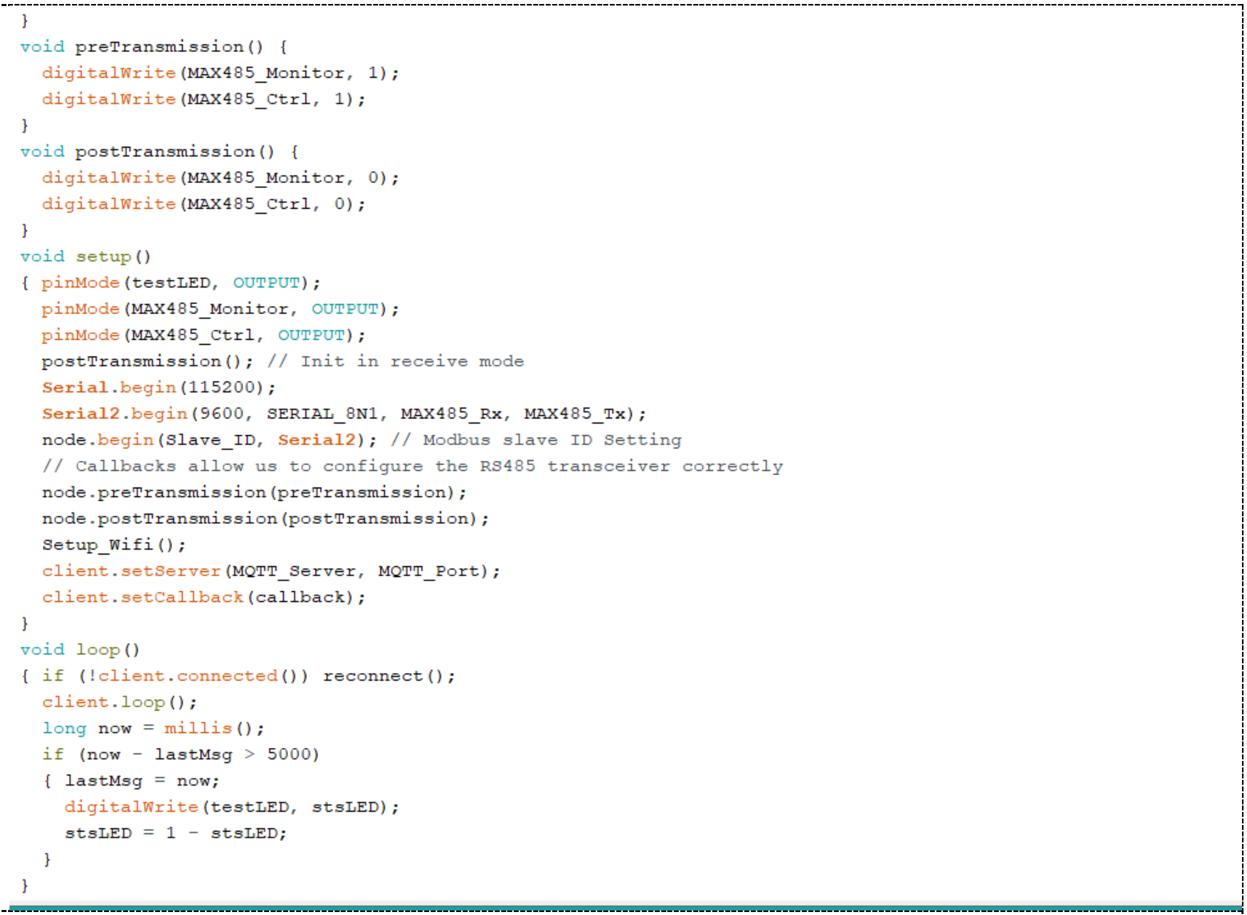
Switch	Switch	Switch	Switch
relay0 (tp007test)	relay1 (tp007test)	relay2 (tp007test)	relay3 (tp007test)
On	On	On	On

29°C ENG 22:14



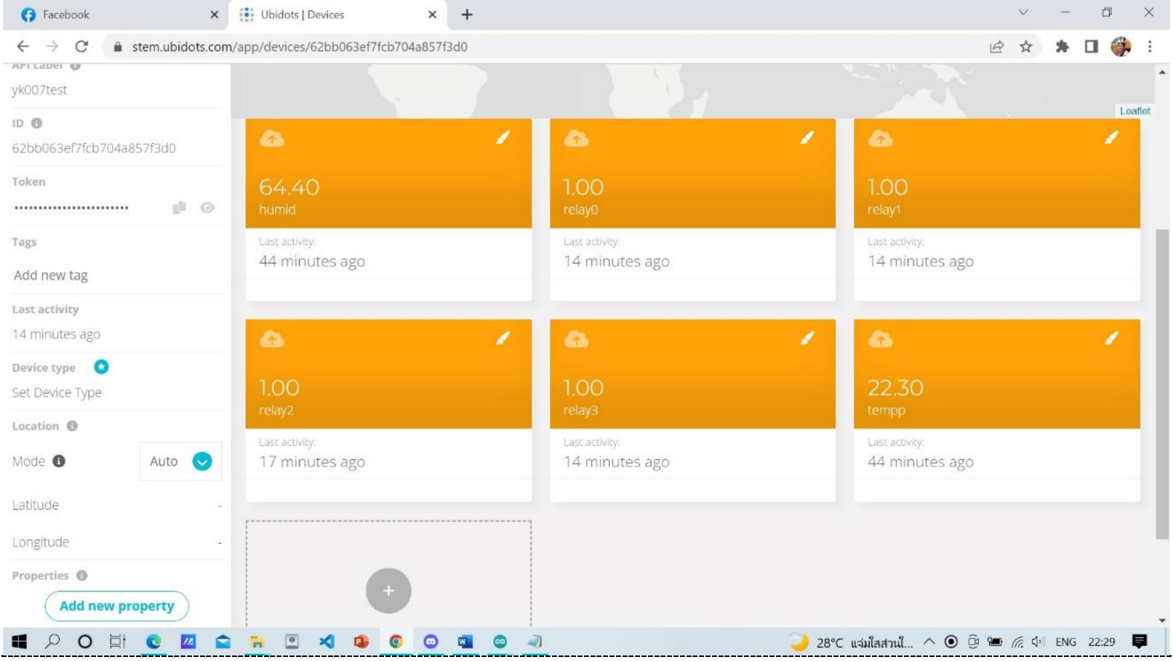
```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ModbusMaster.h>
#define MAX485_Monitor 2
#define MAX485_Ctrl 5 // Pin Ctrl 1=Tx and 0=Rx_NEG
#define MAX485_Rx 16 // Pin RXD2 16
#define MAX485_Tx 17 // Pin TXD2 17
#define Slave_ID 5 // Slave ID
ModbusMaster node; // instantiate ModbusMaster object
const char *My_SSID = "V2036";
const char *My_Pass = "fnafchica";
const char *MQTT_Server = "things.ubidots.com";
const char *MQTT_User = "BBFF-dZWJxFOz4nw2XpLm18D4tsNLUi3M1G";
const char *MQTT_Pass = "BBFF-dZWJxFOz4nw2XpLm18D4tsNLUi3M1G";
const char *PTopic1 = "/v2.0/devices/yk007test";
const char *STopic1 = "/v2.0/devices/yk007test/relay0";
const char *STopic2 = "/v2.0/devices/yk007test/relay1";
const char *STopic3 = "/v2.0/devices/yk007test/relay2";
const char *STopic4 = "/v2.0/devices/yk007test/relay3";
#define MQTT_Port 1883
#define testLED 2
int stsLED = 0;
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
void Setup_Wifi() {
  delay(10); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(My_SSID);
  WiFi.begin(My_SSID, My_Pass);
  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(), MQTT_User, MQTT_Pass)) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
      client.subscribe(STopic1);
      client.subscribe(STopic2);
      client.subscribe(STopic3);
      client.subscribe(STopic4);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
void callback(char *topic, byte *payload, unsigned int length)
{ Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
  }
  int RlyID = (int)topic[29] - 0x30; // '0'
  int RlySts = (int)payload[10] - 0x30; // '0'
  Serial.println("\nRlyID-" + (String)(RlyID) + " >> RlyStatus-" + (String)(RlySts));
  node.writeSingleCoil(RlyID, RlySts);
```



```
}
void preTransmission() {
  digitalWrite(MAX485_Monitor, 1);
  digitalWrite(MAX485_Ctrl, 1);
}
void postTransmission() {
  digitalWrite(MAX485_Monitor, 0);
  digitalWrite(MAX485_Ctrl, 0);
}
void setup()
{ pinMode(testLED, OUTPUT);
  pinMode(MAX485_Monitor, OUTPUT);
  pinMode(MAX485_Ctrl, OUTPUT);
  postTransmission(); // Init in receive mode
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, MAX485_Rx, MAX485_Tx);
  node.begin(Slave_ID, Serial2); // Modbus slave ID Setting
  // Callbacks allow us to configure the RS485 transceiver correctly
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
  Setup_Wifi();
  client.setServer(MQTT_Server, MQTT_Port);
  client.setCallback(callback);
}
void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 5000)
  { lastMsg = now;
    digitalWrite(testLED, stsLED);
    stsLED = 1 - stsLED;
  }
}
```

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ModbusMaster.h>
#define MAX485_Monitor 2
#define MAX485_Ctrl 5 // Pin Ctrl 1=Tx and 0=Rx_NEG
#define MAX485_Rx 16 // Pin RXD2 16
#define MAX485_Tx 17 // Pin TXD2 17
#define Slave_ID 5 // Slave ID
ModbusMaster node; // instantiate ModbusMaster object
const char *My_SSID = "V2036";
const char *My_Pass = "fnafchica";
const char *MQTT_Server = "things.ubidots.com";
const char *MQTT_User = "BBFF-dZWJxFOz4nw2XpLm18D4tsNLUi3M1G";
const char *MQTT_Pass = "BBFF-dZWJxFOz4nw2XpLm18D4tsNLUi3M1G";
const char *PTopic1 = "/v2.0/devices/yk007test";
const char *STopic1 = "/v2.0/devices/yk007test/relay0";
const char *STopic2 = "/v2.0/devices/yk007test/relay1";
const char *STopic3 = "/v2.0/devices/yk007test/relay2";
const char *STopic4 = "/v2.0/devices/yk007test/relay3";
#define MQTT_Port 1883
#define testLED 2
int stsLED = 0;
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
void Setup_Wifi() {
  delay(10); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(My_SSID);
  WiFi.begin(My_SSID, My_Pass);
  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(), MQTT_User, MQTT_Pass)) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
      client.subscribe(STopic1);
      client.subscribe(STopic2);
      client.subscribe(STopic3);
      client.subscribe(STopic4);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
void callback(char *topic, byte *payload, unsigned int length)
{ Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
  }
  int RlyID = (int)topic[29] - 0x30; // '0'
  int RlySts = (int)payload[10] - 0x30; // '0'
  Serial.println("\nRlyID-" + (String)(RlyID) + " >> RlyStatus-" + (String)(RlySts));
  node.writeSingleCoil(RlyID, RlySts);
}
void preTransmission() {
  digitalWrite(MAX485_Monitor, 1);
  digitalWrite(MAX485_Ctrl, 1);
}
void postTransmission() {
  digitalWrite(MAX485_Monitor, 0);
  digitalWrite(MAX485_Ctrl, 0);
}
void setup()
{ pinMode(testLED, OUTPUT);
  pinMode(MAX485_Monitor, OUTPUT);
  pinMode(MAX485_Ctrl, OUTPUT);
  postTransmission(); // Init in receive mode
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, MAX485_Rx, MAX485_Tx);
  node.begin(Slave_ID, Serial2); // Modbus slave ID Setting
  // Callbacks allow us to configure the RS485 transceiver correctly
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
  Setup_Wifi();
  client.setServer(MQTT_Server, MQTT_Port);
  client.setCallback(callback);
}
void loop()
{ if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 5000)
  { lastMsg = now;
    digitalWrite(testLED, stsLED);
    stsLED = 1 - stsLED;
  }
}
```



API Label
yk007test

ID
62bb063ef7fcb704a857f3d0

Token
••••••••••••••••••••••••

Tags
Add new tag

Last activity
14 minutes ago

Device type
Set Device Type

Location

Mode Auto

Latitude -

Longitude -

Properties

64.40
humid
Last activity:
44 minutes ago

1.00
relay0
Last activity:
14 minutes ago

1.00
relay1
Last activity:
14 minutes ago

1.00
relay2
Last activity:
17 minutes ago

1.00
relay3
Last activity:
14 minutes ago

22.30
tempp
Last activity:
44 minutes ago

Quiz_404 – Application

จากทั้งสามข้อที่ผ่านมา เราสามารถใช้หลักการนี้ไปทำ IoT กับอุปกรณ์ต่างๆที่ Subscripe และสามารถที่จะสั่งการจากทางไกลได้ ซึ่งแพลตฟอร์ม Ubidot และ Blynk ถือว่าเป็น IoT ที่แนะนำสำหรับผู้ที่ศึกษา IoT ใหม่ๆ เพราะนอกจากมี UI ที่ใช้งานง่ายยังสามารถใช้ได้ฟรีอีกด้วย