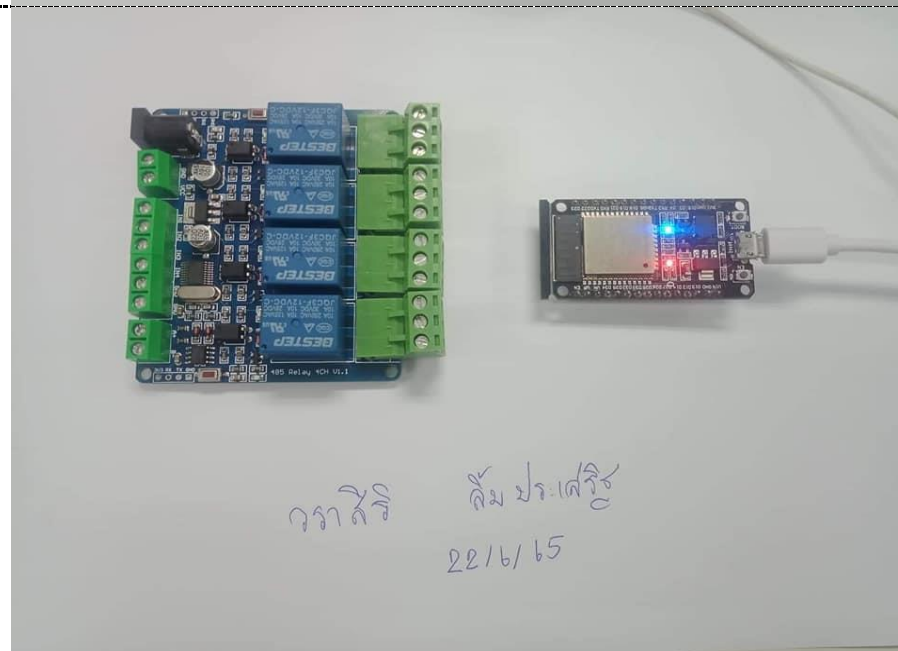
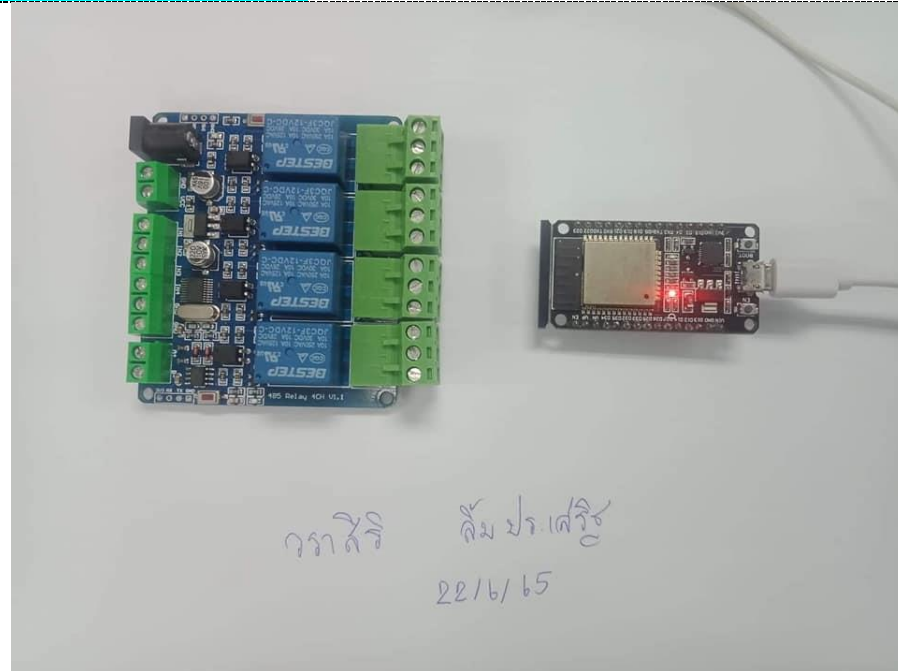


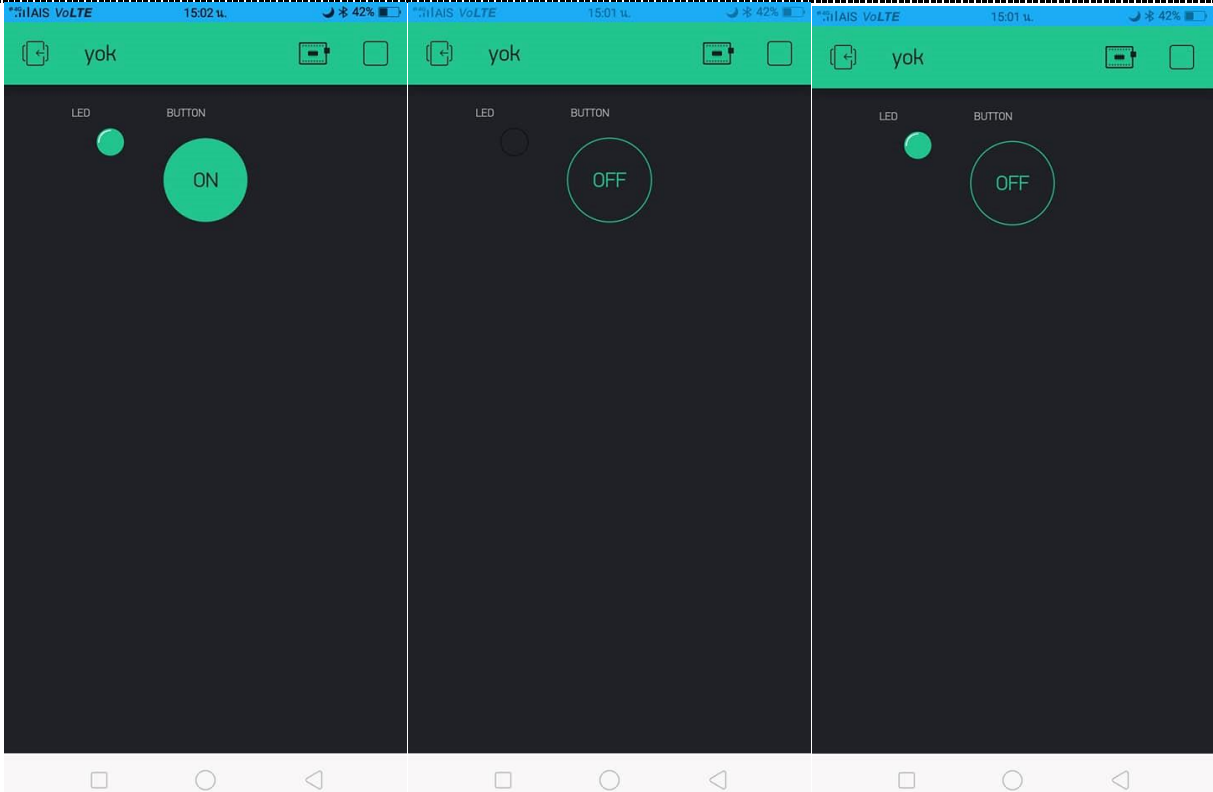
การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร
M2M - Intelligence Machine Control

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--;
}

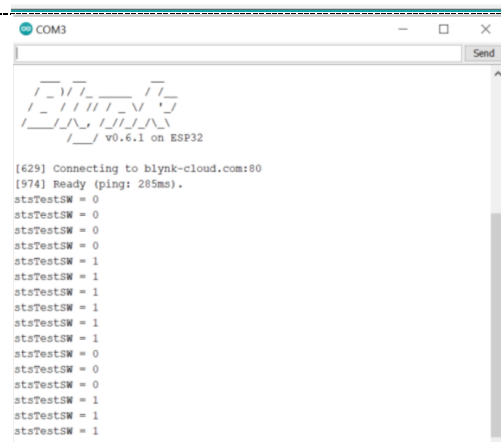
```



```

sketch_jun30a $
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = "WPa2mtFT_6qi9RGfh3Nxc8AXs-y1fnoe";
char ssid[] = "v2036";
char pass[] = "fnafehica";
#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--;
}

```

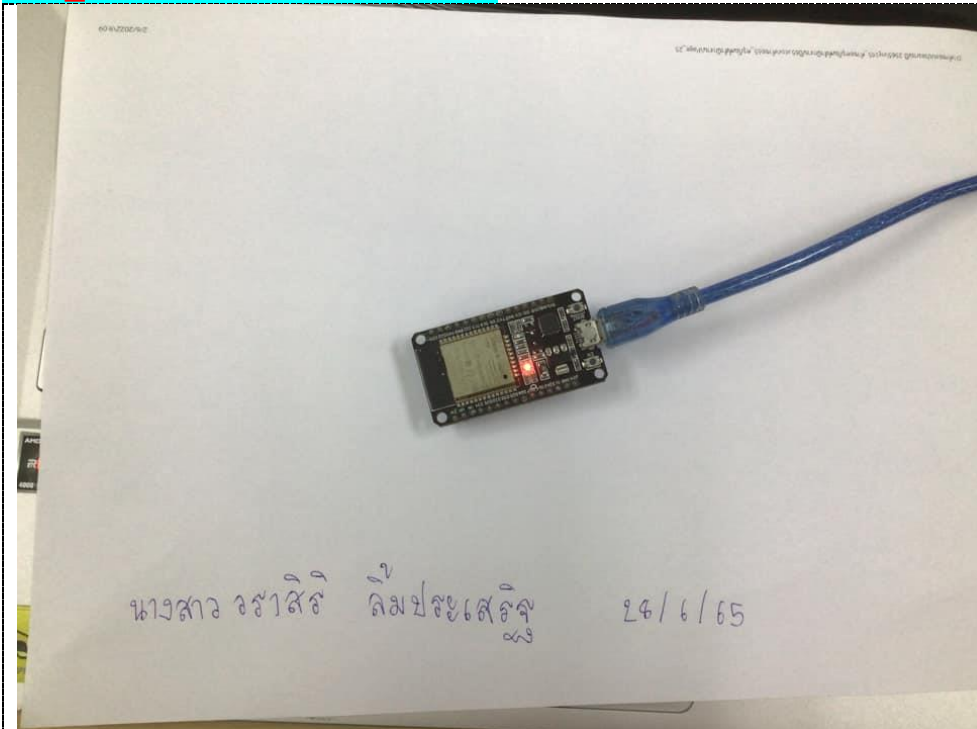


```

COM3
[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 = 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 xTemp = random(2000, 4000) / 100.0;
    float xHumid = random(6000, 8000) / 100.0;
  }
}

```

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

```

sketch_jun30a $
#include <WiFi.h>
#include <PubSubClient.h>
const char *My_SSID = "V2036";
const char *My_Pass = "fnafehica";
const char *MQTT_Server = "things.ubidots.com";
const char *MQTT_User = "BBFF-YgcQcuqoAiO9g46ehWh12TxVwyTjCx";
const char *MQTT_Pass = "BBFF-YgcQcuqoAiO9g46ehWh12TxVwyTjCx";
const char *FTopic1 = "/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 xTemp = random(2000, 4000) / 100.0;
    float xHumid = random(6000, 8000) / 100.0;
  }
}

```

```

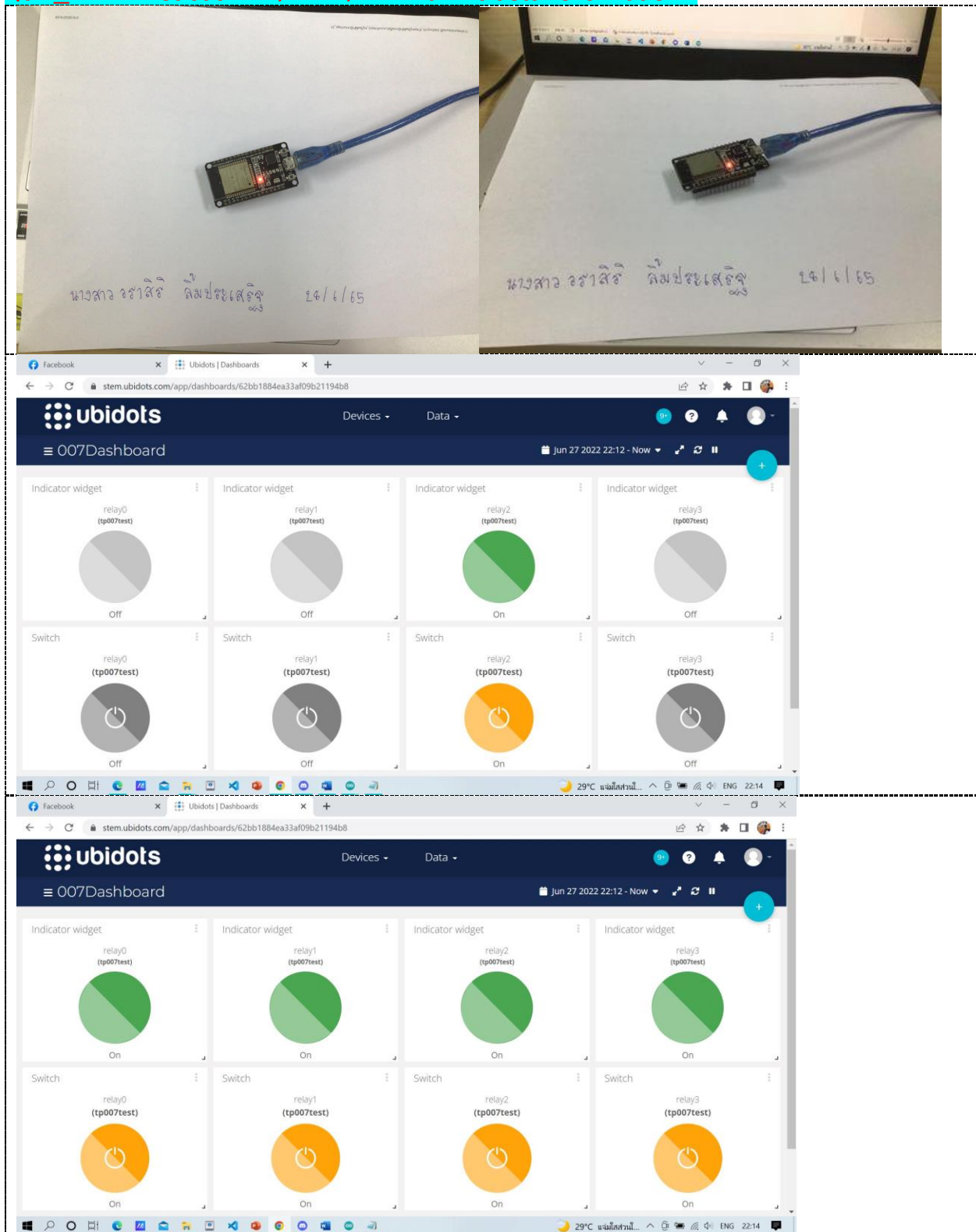
    snprintf (msg, 75, "[ \"humid\" : %5.2f, \"temp\" : %5.2f]", xHumid, xTemp);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(PTopic1, msg);
  }
}

```

< ผลการทดสอบ >

The screenshot displays the Ubidots web application interface. On the left, a sidebar for device 'yk007test' shows its ID (62bb063ef7fcb704a857f3d0), token, and various management options. The main area features a world map and two orange variable cards: 'humid' with a value of 78.24 and 'temp' with a value of 21.03, both updated 7 minutes ago. A 'SET LOCATION' button is visible in the top right. The bottom of the screen shows a Windows taskbar with system information: 30°C, 30% battery, and ENG 21:45.

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



```

sketch_jun30a $
#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 = "fnafehica";
const char *MQTT_Server = "things.ubidots.com";
const char *MQTT_User = "BBFF-dZWJxFOz4nw2XpLm18D4tsNLUi3M1G";
const char *MQTT_Pass = "BBFF-dZWJxFOz4nw2XpLm18D4tsNLUi3M1G";
const char *STopic1 = "/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);
}

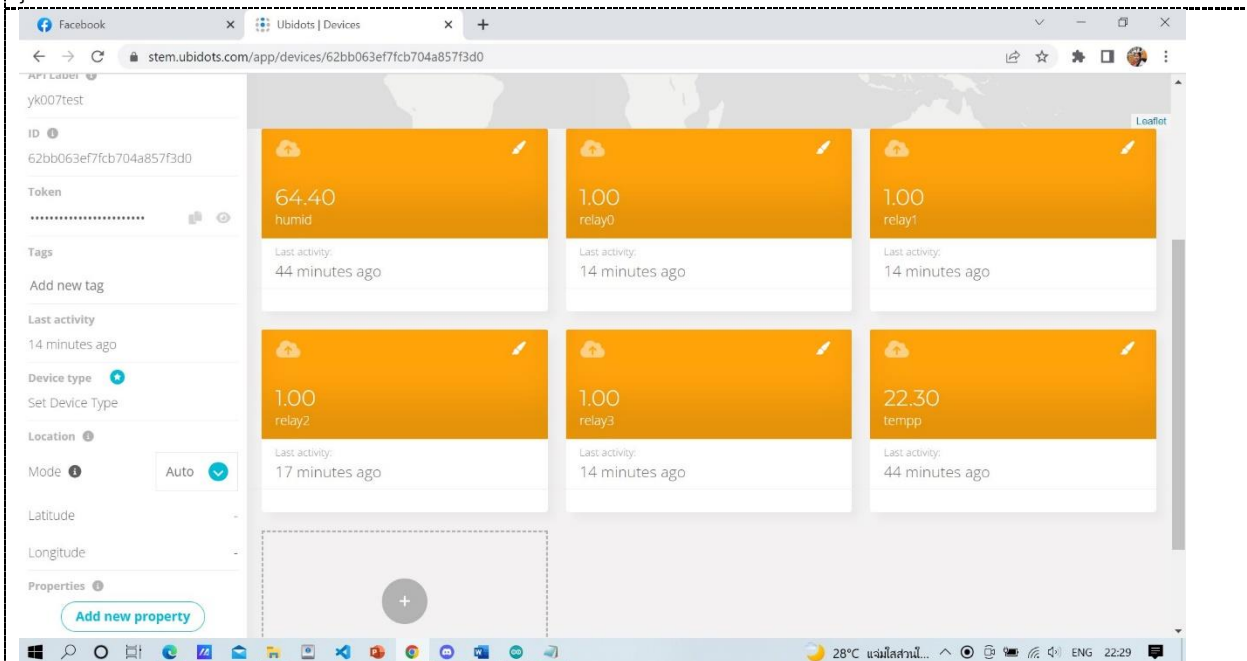
```

```

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;
  }
}
}

```



Quiz_404 – Application

จากทั้งสามข้อที่ผ่านมา เราสามารถใช้หลักการนี้ไปทำ IoT กับอุปกรณ์ต่างๆที่ Subscribe และสามารถที่จะสั่งการจากทางไกลได้ ซึ่งแพลตฟอร์ม Ubidot และ Blynk ถือว่าเป็น IoT ที่แนะนำสำหรับผู้ศึกษา IoT ใหม่ๆ เพราะนอกจากมี UI ที่ใช้งานง่ายยังสามารถใช้ได้ฟรีอีกด้วย