<table>
<tr><td colspan="2" align="center">การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร
M2M - Intelligence Machine Control</td></tr>
</table>

ชื่อ-สกุล : วราสิริ ลิ้มประเสริฐ B6214005
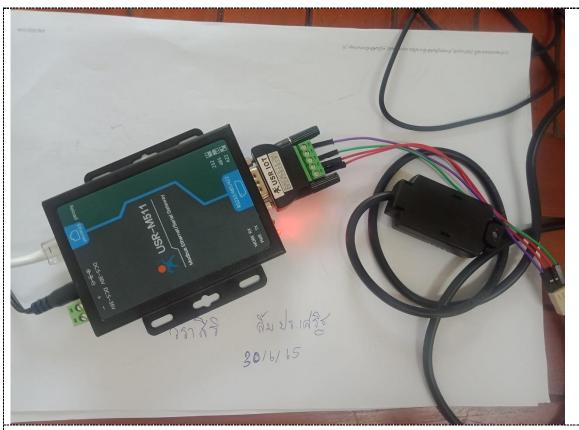
5/5: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_301 – Start SCADA

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

Quiz_302 – Modbus TCP Read/Write
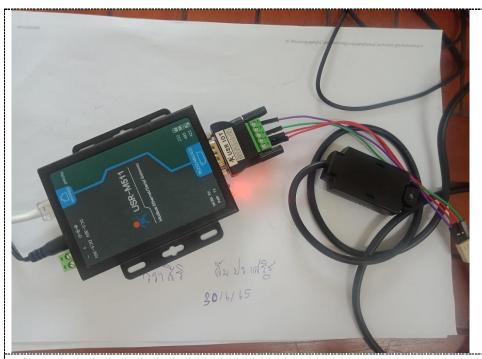
รายยละเอียดการทดสอบ

< 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 ⌄   Clear output

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 ⌄ Clear output