

การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร M2M - Intelligence Machine Control
2/4 – Industrial Protocol, PLC and SCADA System <ul style="list-style-type: none"> • OSI Model and Industrial Protocol • การโปรแกรมเพื่อสื่อสารข้อมูลผ่าน Modbus RTU/ASCII Protocol • การโปรแกรม PLC แบบ Single Controller • คำถามท้ายบทเพื่อทดสอบความเข้าใจ

1/4: -- OSI Model and Industrial Protocol

< เอกสารแนบ “M2M-D21 -- OSI Model and Industrial Communication Protocol.pdf “ >

2/4: -- การโปรแกรมเพื่อสื่อสารข้อมูลผ่าน Modbus RTU/ASCII Protocol

Test 1/4. MODBUS RTU SENSOR H/T

http://www.etteam.com/productSensor/MODBUS%20RTU%20SENSOR%20H_T/คู่มือ%20Modbus%20RTU%20Sensor%20HT.pdf

Modbus RTU Sensor H/T เป็นชุด Sensor สำหรับวัดอุณหภูมิ และความชื้น โดยใช้การเชื่อมต่อสื่อสารผ่านสัญญาณ RS485 แบบ Half Duplex ด้วย Protocol การสื่อสารแบบ Modbus RTU รองรับการเชื่อมต่อสื่อสารระยะไกลแบบ Multi-drop ตามมาตรฐาน RS485 สามารถเชื่อมต่ออุปกรณ์ร่วมกันในระบบบัสเดียวกันได้มากถึง 255 อุปกรณ์ โดยสามารถกำหนดค่า Address ได้อิสระสามารถตั้งกำหนดค่าเพื่อชดเชยค่าการวัดของเซ็นเซอร์ที่อ่านได้ ให้ผลลัพธ์ค่าอุณหภูมิและความชื้นแบบ Signed ผ่าน Protocol Modbus RTU

ข้อกำหนดในการสื่อสาร

ข้อกำหนดในการติดต่อสื่อสาร จะใช้การสื่อสารแบบอนุกรม ผ่านระบบสัญญาณ RS485 แบบ Half Duplex ด้วย Protocol แบบ Modbus RTU มาตรฐาน โดยมีข้อกำหนดของค่าพารามิเตอร์ของการสื่อสารเป็นดังนี้

- Baudrate 9600BPS
- Data 8Bit
- None Parity
- 1 Stop Bit

การกำหนด Address อุปกรณ์

Modbus RTU Sensor H/T สามารถกำหนดหมายเลข Device Address ได้ 256 ตำแหน่งระหว่าง 0-255 แต่สำหรับในการสื่อสารของ Modbus RTU นั้นยอมให้ Device มีค่าตำแหน่งระหว่าง 1-255 เท่านั้น ค่าตำแหน่ง 0 สงวนไว้สำหรับอุปกรณ์ที่ทำหน้าที่เป็น Master โดยค่ามาตรฐานจะกำหนดค่าหมายเลข Device Address เป็น หมายเลข 1 ไว้ ซึ่งผู้ใช้สามารถกำหนดค่าตำแหน่ง Address ได้ใหม่ โดยในการกำหนดตำแหน่งนั้นต้องทำการเชื่อมต่ออุปกรณ์ในบัสแบบ RS485 เพียง 1 ตัว ในระบบเท่านั้น ถ้ามีการเชื่อมต่ออุปกรณ์มากกว่า 1 ตัวในบัส อุปกรณ์ทุกตัวที่เชื่อมต่ออยู่จะถูกกำหนดค่าเหมือนกันทั้งหมดทุกตัว ในการสั่งกำหนดค่าตำแหน่ง Device Address นั้น Master จะส่งคำสั่งแบบ Modbus RTU ผ่านฟังก์ชัน 06 เพื่อสั่งกำหนดค่าตำแหน่ง Device Address ใหม่ไปยังอุปกรณ์หลังจากอุปกรณ์ได้รับคำสั่งถูกต้องแล้วจะตอบรับด้วยชุดข้อมูลเหมือนที่ได้รับกลับคืนมายังบัส ดังตัวอย่าง

คำสั่ง	Addr	Func	Start Address		Modify Data		CRC	
Master ส่งคำสั่ง	0x00	0x06	0x00	0x00	0x00	0x01	0x49	0xDB
Device ตอบรับ	0x00	0x06	0x00	0x00	0x00	0x01	0x49	0xDB

คำสั่ง	Addr	Func	Start Address		Modify Data		CRC	
Master ส่งคำสั่ง	0x00	0x06	0x00	0x00	0x00	0x02	0x09	0xDA
Device ตอบรับ	0x00	0x06	0x00	0x00	0x00	0x02	0x09	0xDA

ตัวอย่าง การส่งคำสั่งเปลี่ยนค่าตำแหน่ง Device Address เป็น 0x01 และ 0x02

Test 0a/0 – ใช้ Modbus Poll ในการกำหนด Address Device

- ต้องต่ออุปกรณ์เพียงตัวเดียวในสายสื่อสาร มิฉะนั้นจะกลายเป็นว่าอุปกรณ์ทั้งหมดมี ID เดียวกัน

The image displays two screenshots of the Modbus Poll software interface. The top screenshot shows the 'Modbus Poll - Mbpoll1' window with the 'Connection' menu open, highlighting 'Connect... F3'. The 'Connection Setup' dialog box is also visible, showing settings for 'Serial Port', 'Port 3', 'Mode' (RTU selected), '9600 Baud', '8 Data bits', 'None Parity', '1 Stop Bit', 'Response Timeout' (1000 ms), 'Delay Between Polls' (10 ms), 'Remote Server IP Address' (0.0.0.0), 'Port' (502), and 'Connect Timeout' (3000 ms). The bottom screenshot shows the 'Functions' menu with '06: Write Single Register... Alt+F6' selected. The 'Write Single Register' dialog box is also visible, showing 'Slave ID' (0), 'Address' (0), and 'Value' (0). The 'Result' section shows 'Response ok' and the 'Use Function' section has '06: Write single register' selected.

การอ่านค่าเซ็นเซอร์

ในการส่งอ่านค่าเซ็นเซอร์จะใช้ฟังก์ชัน 0x03 สำหรับส่งอ่านค่า โดย Master จะส่งข้อมูลเป็นลำดับ คือ Device Address, Command(0x03), Start Address(0x0000), Read Size(0x0002 = 2Value)และตามด้วยค่ารหัส CRC 16บิต ตามลำดับไปยังอุปกรณ์ เมื่ออุปกรณ์ได้รับคำสั่งถูกต้องก็จะตอบรับกลับมาด้วยลำดับข้อมูลจำนวน 0 ไบต์ คือ Device Address, Command(0x03), Valid Data Size(0x04 = 4Byte), data1, data2, data3, data4,

CRC16 โดย data1,data2 จะเป็นค่าของอุณหภูมิ ส่วน data3,data4 จะเป็นค่าของความชื้น โดยเป็นค่าแบบ Signed ดังตัวอย่าง(ค่าในตารางเป็นเลขฐานWT)

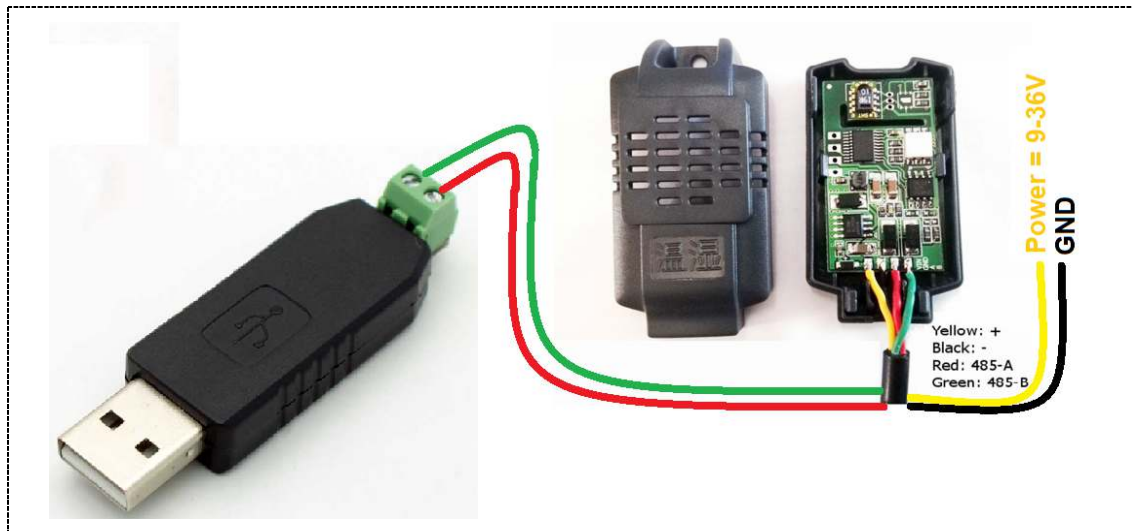
Command	Addr	Func	Start Addr		Read Size		CRC	
Master Send	0x01	0x03	0x00	0x00	0x00	0x02	0xC4	0x0B

Command	Addr	Func	Size	Temperature		Humidity		CRC	
Device Echo	0x01	0x03	0x04	0x01	0x16	0x02	0xC5	0xDB	0x38

- Temperature Value : 0x01,0x16 = 0x0116 = 278(decimal) = 27.8 'C
- Humidity Value : 0x02,0xC5 = 0x02C5 = 709(decimal) = 70.9 %RH

Test 1a/4 -- ทดสอบโดย Modbus Poll

1. ต่อดวงจร ใช้แหล่งจ่าย 9 – 36V



1. ตั้งค่า Modbus Poll

Connection Setup

Connection: ☒ Serial Port ☐ TCP/IP ☐ UDP/IP

Port 21

9600 Baud

8 Data bits

None Parity

1 Stop Bit

Mode: ☒ RTU ☐ ASCII

Response Timeout: 1000 [ms]

Delay Between Polls: 10 [ms]

Advanced...

Remote Server IP Address: 0.0.0.0 Port: 502 Connect Timeout: 3000 [ms]

Read/Write Definition

Slave ID: 1

Function: 03 Read Holding Registers (4x)

Address: 0

Quantity: 2

Scan Rate: 1000 ms

☒ Read/Write Enabled

Read/Write Once

View

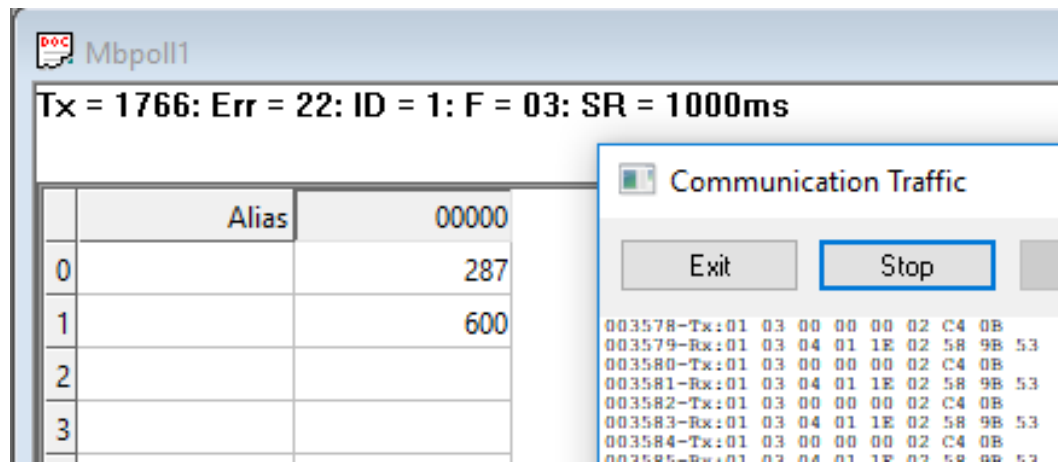
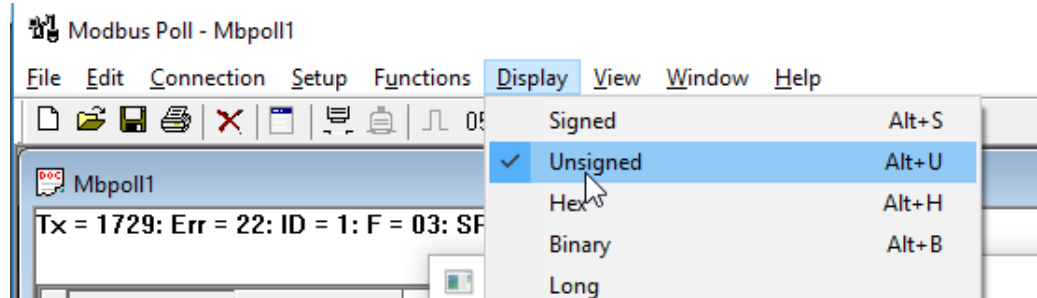
Rows: ☒ 10 ☐ 20 ☐ 50 ☐ 100

☐ Hide Alias Columns

☐ Address in Cell

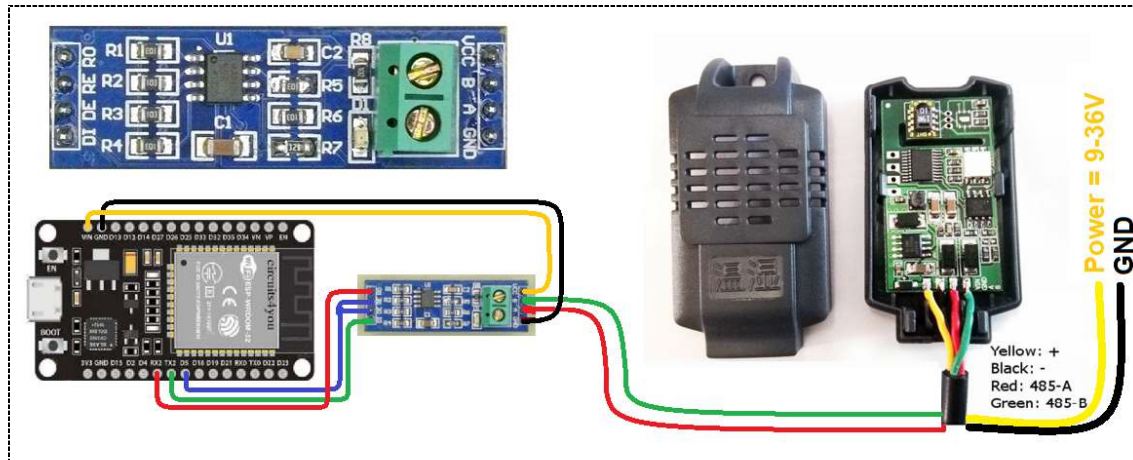
☐ PLC Addresses (Base 1)

Display: Unsigned



Test 1b/4 -- ทดสอบโดย Arduino

1. ตัวอย่าง ใช้แหล่งจ่าย 9 – 36V



2. ทดสอบโปรแกรม – version 1.0

```
#define RS485Transmit  HIGH
#define RS485Receive  LOW
#define RS485Control  5 //RS485 Direction control
#define Pin_LEDMonitor 2

byte byteSend;
int StepConut = 0;

void setup() {
  pinMode(RS485Control, OUTPUT);
  pinMode(Pin_LEDMonitor, OUTPUT);
  Serial.begin(9600);
  Serial2.begin(9600);
  digitalWrite(RS485Control, RS485Receive);
  Serial.println("Start Test MODBUS RTU");
}

byte Request[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x02, 0xC4, 0x0B};

void loop() {
  Serial.print("\nTest");
  Serial.print(++StepConut);
  Serial.print(" > ");
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit);
  delay(10);
  for (int i = 0; i < sizeof(Request); i++)
    Serial2.write(Request[i]);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);

  for (long int i = 0; i < 40000; i++) {
    if (Serial2.available()) {
      byteSend = Serial2.read();
      if (byteSend < 0x10) Serial.print("0");
      Serial.print(byteSend, HEX);
      Serial.print(" ");
    }
    delayMicroseconds(100);
  }
}
```

3. ทดสอบโปรแกรม – version 1.1

```

#define RS485Transmit  HIGH
#define RS485Receive  LOW
#define RS485Control  5 //RS485 Direction control
#define Pin_LEDMonitor 2

int Wr_Index, StepConut = 0;
byte Request[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x02, 0xC4, 0x0B};
byte Echo[20];

void setup() {
  pinMode(Pin_LEDMonitor, OUTPUT);
  pinMode(RS485Control, OUTPUT);
  Serial.begin(9600);
  Serial2.begin(9600);
  digitalWrite(RS485Control, RS485Receive);
  Serial.println("Start Test MODBUS RTU");
}

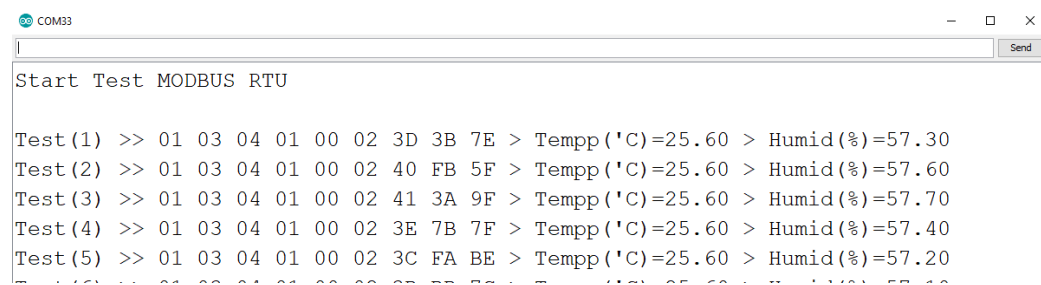
void loop() {
  Serial.print("\nTest(");
  Serial.print(++StepConut);
  Serial.print(") >>");
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit);
  delay(10);
  for (int i = 0; i < sizeof(Request); i++)
    Serial2.write(Request[i]);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);
  Wr_Index = -1;
  for (long int i = 0; i < 300000; i++) {
    if (Serial2.available())
      Echo[Wr_Index++] = Serial2.read();
    if (Wr_Index > 12) i = 999999;
    delayMicroseconds(10);
  }

  for (int i = 0; i < Wr_Index - 1; i++) {
    Serial.print(" ");
    if (Echo[i] < 0x10) Serial.print("0");
    Serial.print(Echo[i], HEX);
  }

  float Tempp = (Echo[3] * 256 + Echo[4]) / 10.0;
  Serial.print(" > Tempp('C)="); Serial.print(Tempp, 2);
  float Humid = (Echo[5] * 256 + Echo[6]) / 10.0;
  Serial.print(" > Humid(%)="); Serial.print(Humid, 2);

  delay(5000);
}

```



```

COM3
Start Test MODBUS RTU

Test(1) >> 01 03 04 01 00 02 3D 3B 7E > Tempp('C)=25.60 > Humid(%)=57.30
Test(2) >> 01 03 04 01 00 02 40 FB 5F > Tempp('C)=25.60 > Humid(%)=57.60
Test(3) >> 01 03 04 01 00 02 41 3A 9F > Tempp('C)=25.60 > Humid(%)=57.70
Test(4) >> 01 03 04 01 00 02 3E 7B 7F > Tempp('C)=25.60 > Humid(%)=57.40
Test(5) >> 01 03 04 01 00 02 3C FA BE > Tempp('C)=25.60 > Humid(%)=57.20

```

Test 1c/4 -- ทดสอบโดย ModbusMaster Library on Arduino

5. ทดสอบสอโปรแกรม – version 2.0 Using Modbus Library
6. Add **ModbusMaster by Doc Walker V2.0.1** and test this code

<https://github.com/4-20ma/ModbusMaster>

ModbusMaster

by Doc Walker 4-20ma@wvfans.net> Version 2.0.1 **INSTALLED**

Enlighten your Arduino to be a Modbus master. Enables communication with Modbus slaves over RS232/485 (via RTU protocol). Requires an RS232/485 transceiver.

[More info](#)

Select version ▾

Install

```
#include "ModbusMaster.h" //https://github.com/4-20ma/ModbusMaster
```

```
#define Slave_ID 1
#define MAX485_RE_NEG 4
#define RX_PIN 16
#define TX_PIN 17
```

```
ModbusMaster modbus;
```

```
void preTransmission() {
  digitalWrite(MAX485_RE_NEG, HIGH); //Switch to transmit data
}
```

```
void postTransmission() {
  digitalWrite(MAX485_RE_NEG, LOW); //Switch to receive data
}
```

```
void setup() {
  pinMode(MAX485_RE_NEG, OUTPUT);
  digitalWrite(MAX485_RE_NEG, LOW);
  Serial.begin(115200, SERIAL_8N1);
  Serial2.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  modbus.begin(Slave_ID, Serial2);
  modbus.preTransmission(preTransmission);
  modbus.postTransmission(postTransmission);
}
```

```
long lastMillis = 0;
void loop() {
  long currentMillis = millis();
  if (currentMillis - lastMillis > 1000) {
    uint8_t result = modbus.readHoldingRegisters(0, 2);
    if (getResultMsg(&modbus, result)) {
      Serial.println();
      double res_dbl = modbus.getResponseBuffer(0) / 10;
      String res = "Temperature: " + String(res_dbl) + " C\r\n";
      res_dbl = modbus.getResponseBuffer(1) / 10;
      res += "Humidity: " + String(res_dbl) + " %";
      Serial.println(res);
    }
    lastMillis = currentMillis;
  }
}
```

```
bool getResultMsg(ModbusMaster *node, uint8_t result) {
  String tmpstr2 = "\r\n";
  switch (result) {
    case node->ku8MBSuccess:
      return true;
      break;
    case node->ku8MBIllegalFunction:
      tmpstr2 += "Illegal Function";
  }
}
```

```
break;
case node->ku8MBIllegalDataAddress:
    tmpstr2 += "Illegal Data Address";
    break;
case node->ku8MBIllegalDataValue:
    tmpstr2 += "Illegal Data Value";
    break;
case node->ku8MBSlaveDeviceFailure:
    tmpstr2 += "Slave Device Failure";
    break;
case node->ku8MBInvalidSlaveID:
    tmpstr2 += "Invalid Slave ID";
    break;
case node->ku8MBInvalidFunction:
    tmpstr2 += "Invalid Function";
    break;
case node->ku8MBResponseTimedOut:
    tmpstr2 += "Response Timed Out";
    break;
case node->ku8MBInvalidCRC:
    tmpstr2 += "Invalid CRC";
    break;
default:
    tmpstr2 += "Unknown error: " + String(result);
    break;
}
Serial.println(tmpstr2);
return false;
}
```

Temperature: 30.00 C

Humidity: 78.00 %

Temperature: 30.00 C

Humidity: 78.00 %

Temperature: 30.00 C

Humidity: 78.00 %

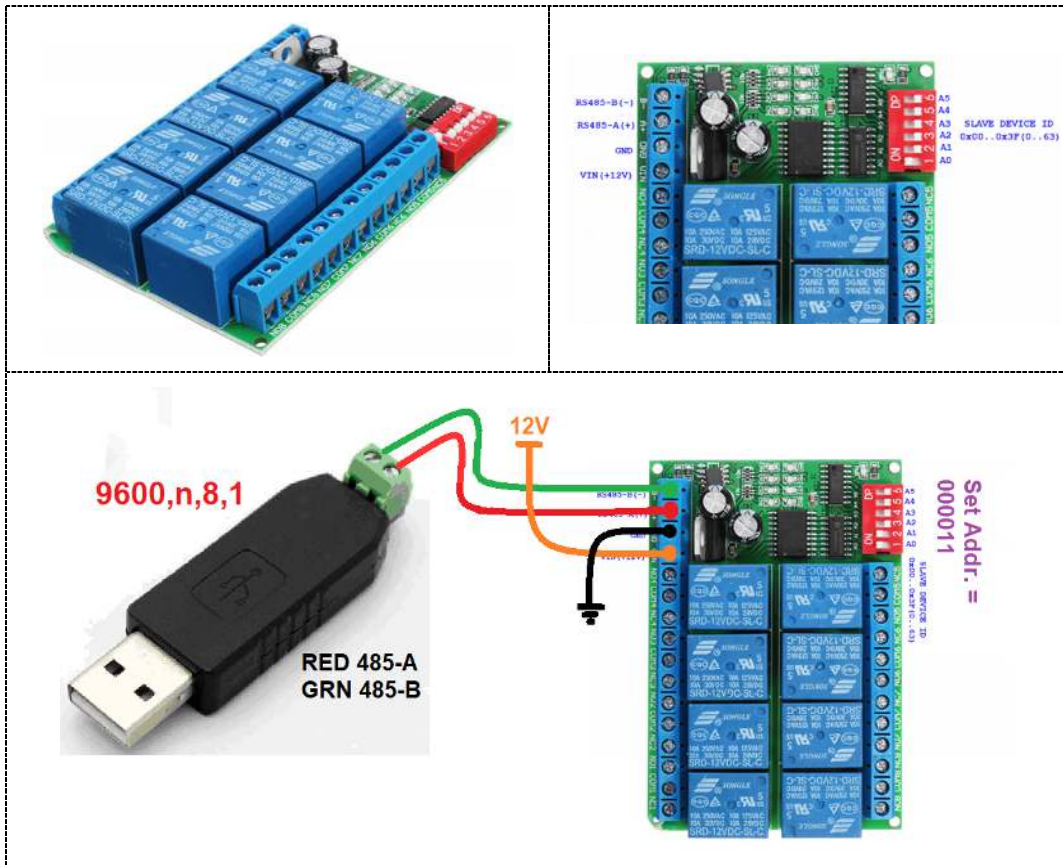
Temperature: 30.00 C

Humidity: 78.00 %

Test 2/4. Ctrl Relay - Modbus RTU Relay8

Test 2a/4 -- ทดสอบโดย Modbus Poll

7. ศึกษาการทำงานของ MODBUS RTU Relay8 และต่อวงจร



8. การสั่ง ปิด-เปิดรีเลย์

Byte No.	1	2	3	4	5	6	7	8
Modbus	Slave ID	Function	Address		Data		CRC Check	
Function	Device Addr	Function	Ch Number		Command	Delay	CRC Check	
Open Ch1	0x00...0x3F	0x06	0x0001		0x01	0x00	2 Byte CRC	

Close Ch1	0x00...0x3F	0x06	0x0001	0x02	0x00	2 Byte CRC		
Close Ch2	0x00 0x3F	0x06	0x0002	0x02	0x00	2 Byte CRC		

On CH-1 >> 0x03,0x06,0x00,0x01,0x01,0x00,H-CRC,L-CRC

0x03,0x06,0x00,0x01,0x01,0x00,0xD8,0x78

Off CH-1 >> 0x03,0x06,0x00,0x01,0x02,0x00, H-CRC, L-CRC

0x03,0x06,0x00,0x01,0x02,0x00, 0xD8,0x88

9. การสั่ง ปิด-เปิดรีเลย์ ด้วย Modbus Poll ใช้ Baudrate = 9600,n,8,1

The image displays two screenshots of the Modbus Poll - Mbpoll1 software interface, showing the 'Write Single Register' dialog box and the 'Communication Traffic' window.

Top Screenshot:

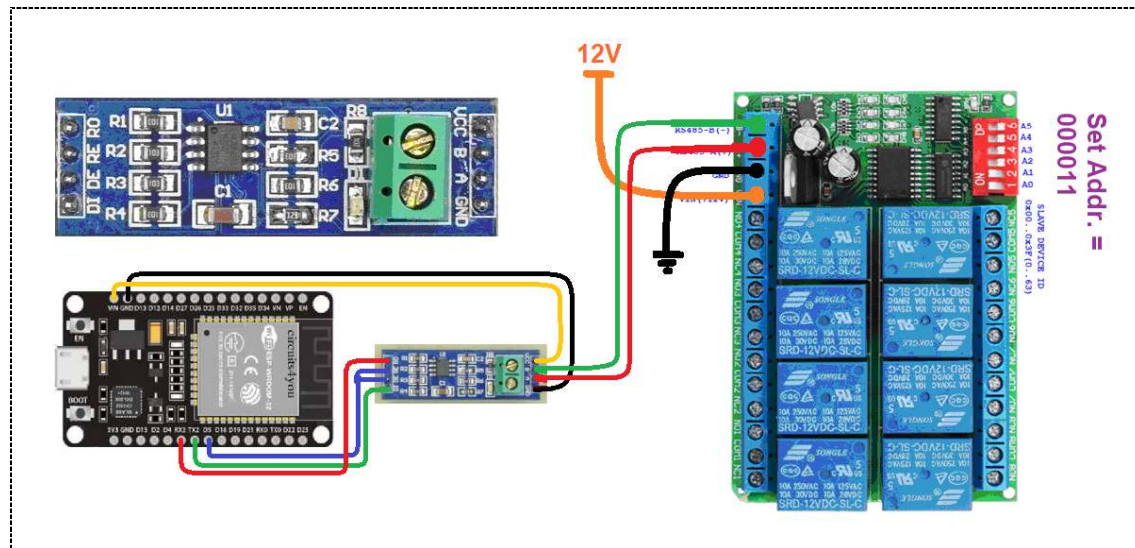
- Write Single Register Dialog:** Slave ID: 3, Address: 1, Value: 256. The 'Result' section shows 'Timeout Error' and 'Close dialog on "Response ok"'. The 'Use Function' section has '06: Write single register' selected.
- Communication Traffic:** A list of transactions is shown. The transaction 002914-Tx:03 06 00 01 01 00 D8 78 is highlighted with a red circle, indicating a successful response.
- Hex Data:** 03 06 00 01 01 00 D8 78

Bottom Screenshot:

- Write Single Register Dialog:** Slave ID: 3, Address: 1, Value: 512. The 'Result' section shows 'Timeout Error' and 'Close dialog on "Response ok"'. The 'Use Function' section has '06: Write single register' selected.
- Communication Traffic:** A list of transactions is shown. The transaction 003177-Tx:03 06 00 01 02 00 D8 88 is highlighted with a red circle, indicating a successful response.
- Hex Data:** 03 06 00 01 02 00 D8 88

Test 2b/4 – ทดสอบโดย Arduino – Direct Command

11. ตัวอย่าง



12. การสั่ง ปิด-เปิดรีเลย์ ด้วย Arduino V1.0

```

#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 5 //RS485 Direction control
#define Pin_LEDMonitor 2

int Wr_Index, StepConut = 0;
byte Echo[20];
byte cmd_On[] = {0x03, 0x06, 0x00, 0x01, 0x01, 0x00, 0xD8, 0x78};
byte cmd_Off[] = {0x03, 0x06, 0x00, 0x01, 0x02, 0x00, 0xD8, 0x88};

void setup() {
  pinMode(Pin_LEDMonitor, OUTPUT);
  pinMode(RS485Control, OUTPUT);
  Serial.begin(9600);
  Serial2.begin(9600);
  digitalWrite(RS485Control, RS485Receive);
  Serial.println("Start Test MODBUS RTU");
}

void loop() {
  Serial.print("\nTest");
  Serial.print(++StepConut);
  Serial.print(">>");
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit);
  delay(10);
  if ((StepConut % 2) == 0)
    for (int i = 0; i < sizeof(cmd_Off); i++)
      Serial2.write(cmd_Off[i]);
  else
    for (int i = 0; i < sizeof(cmd_On); i++)
      Serial2.write(cmd_On[i]);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);
  delay(5000);
}

```

Test 2c/4 – ทดสอบโดย Arduino – On/Off Relay Long Coding

14. การสั่ง ปิด-เปิดรีเลย์ ด้วย Arduino V2.0

```

#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 5 //RS485 Direction control
#define Pin_LEDMonitor 2

byte Board_ID = 0x03;
byte Mdbb_Cmd = 0x06;
byte H_RelayID = 0x00;
byte L_RelayID = 0x03;
byte Relay_On = 0x01;
byte Relay_Off = 0x02;
byte OnOff_Dly = 0x00;
byte HByte_CRC = 00;
byte LByte_CRC = 00;

int StepConut = 0;
byte Echo[20];

void setup() {
  pinMode(Pin_LEDMonitor, OUTPUT);
  pinMode(RS485Control, OUTPUT);
  Serial.begin(9600);
  Serial2.begin(9600);
  digitalWrite(RS485Control, RS485Receive);
  Serial.println("Start Test MODBUS RTU");
}

uint16_t CRC16_Update(uint16_t tempCRC, uint8_t inData) {
  tempCRC ^= inData;
  for (int i = 0; i < 8; ++i)
    if (tempCRC & 1) tempCRC = (tempCRC >> 1) ^ 0xA001;
    else tempCRC = (tempCRC >> 1);
  return tempCRC;
}

uint16_t SendByte_CRCUpdate(uint16_t tempCRC, uint8_t inData) {
  Serial2.write(inData);
  if (inData < 0x10) Serial.print("0");
  Serial.print(inData, HEX);
  Serial.print(" ");
  tempCRC = CRC16_Update(tempCRC, inData);
  return tempCRC;
}

void RTU_RelayCtrl(int rly_ID, byte rly_Cmd) {
  uint16_t Calc_CRC = 0xffff; // the initial value
  H_RelayID = highByte(rly_ID);
  L_RelayID = lowByte(rly_ID);
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit); delay(10);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Board_ID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Mdbb_Cmd);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, H_RelayID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, L_RelayID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, rly_Cmd);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, OnOff_Dly);
  HByte_CRC = highByte(Calc_CRC);
  LByte_CRC = lowByte(Calc_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, LByte_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, HByte_CRC);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);
  Serial.println();
}

void loop() {
  for (int relay = 1; relay <= 8; relay++) {
    RTU_RelayCtrl(relay, Relay_On);
    delay(3000);
  }
  for (int relay = 1; relay <= 8; relay++) {
    RTU_RelayCtrl(relay, Relay_Off);
    delay(3000);
  }
}

```

Test 2d/4 – ทดสอบโดย Arduino Library

```
#include "ModbusMaster.h" //https://github.com/4-20ma/ModbusMaster
```

```
#define Slave_ID 3
#define MAX485_RE_NEG 4
#define RX_PIN 16
#define TX_PIN 17
```

```
ModbusMaster modbus;
```

```
void preTransmission() {
  digitalWrite(MAX485_RE_NEG, HIGH); //Switch to transmit data
}
```

```
void postTransmission() {
  digitalWrite(MAX485_RE_NEG, LOW); //Switch to receive data
}
```

```
void setup() {
  pinMode(MAX485_RE_NEG, OUTPUT);
  digitalWrite(MAX485_RE_NEG, LOW);
  Serial.begin(115200, SERIAL_8N1);
  Serial2.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  modbus.begin(Slave_ID, Serial2);
  modbus.preTransmission(preTransmission);
  modbus.postTransmission(postTransmission);
}
```

```
long lastMillis = 0;
void loop() {
  uint8_t result;
  result = modbus.writeSingleRegister(1, 0x0100); // Relay1 On
  getResultMsg(&modbus, result);
  delay(5000);

  result = modbus.writeSingleRegister(1, 0x0200); // Relay1 Off
  getResultMsg(&modbus, result);
  delay(5000);
}
```

```
bool getResultMsg(ModbusMaster *node, uint8_t result) {
  String tmpstr2 = "\r\n";
  switch (result) {
    case node->ku8MBSuccess:
      tmpstr2 += "Compleat";
      Serial.println(tmpstr2);
      return true;
      break;
    case node->ku8MBIllegalFunction:
      tmpstr2 += "Illegal Function";
      break;
    case node->ku8MBIllegalDataAddress:
      tmpstr2 += "Illegal Data Address";
      break;
    case node->ku8MBIllegalDataValue:
      tmpstr2 += "Illegal Data Value";
      break;
    case node->ku8MBSlaveDeviceFailure:
      tmpstr2 += "Slave Device Failure";
      break;
    case node->ku8MBInvalidSlaveID:
      tmpstr2 += "Invalid Slave ID";
      break;
    case node->ku8MBInvalidFunction:
      tmpstr2 += "Invalid Function";
      break;
    case node->ku8MBResponseTimedOut:
      tmpstr2 += "Response Timed Out";
      break;
  }
}
```

```
case node->ku8MBInvalidCRC:  
  tmpstr2 += "Invalid CRC";  
  break;  
default:  
  tmpstr2 += "Unknown error: " + String(result);  
  break;  
}  
Serial.println(tmpstr2);  
return false;  
}
```

COM3

Compleat

Compleat

Invalid Slave ID

Compleat

Compleat

Compleat

Compleat

Compleat

Test 2e/4 – ทดสอบโดย Arduino – Read Status Relay

การอ่านค่า Status และการตอบรับ

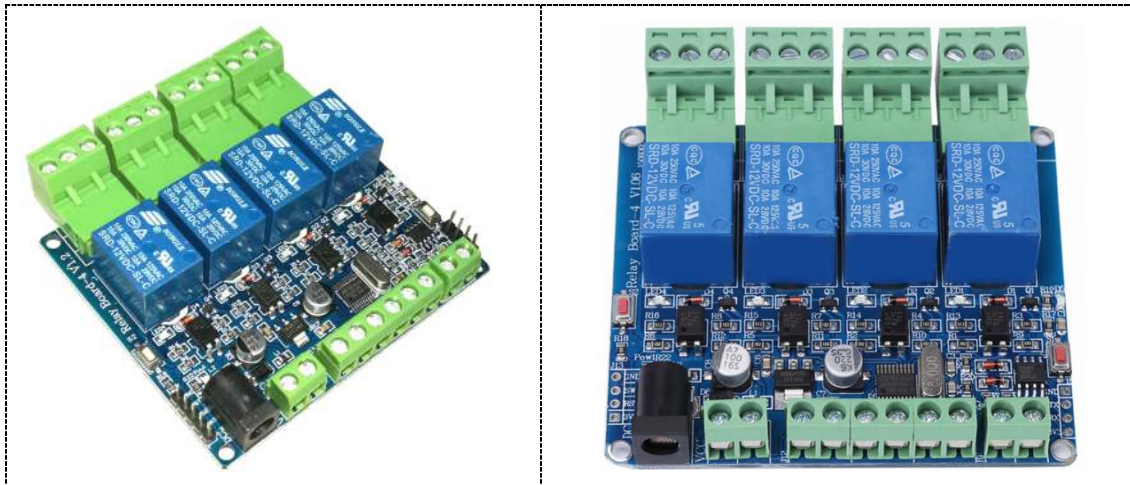
การอ่านค่า Status และ การตอบรับคำสั่งของบอร์ด Modbus RTU Relay8 จะใช้ฟังก์ชัน 0x03 ใน Modbus RTU Protocol สำหรับอ่านค่าและตอบรับโดยมีรูปแบบดังนี้

Byte No.	1	2	3	4	5	6	7	8
Modbus	Slave ID	Function	Address		Data		CRC Check	
Function	Device Addr	Function	Start Reg Addr		Register Length		CRC Check	
Read Ch1	0x00...0x3F	0x03	0x0001		0x0001		2 Byte CRC	
Read Ch2	0x00...0x3F	0x03	0x0002		0x0001		2 Byte CRC	
Read Ch3	0x00...0x3F	0x03	0x0003		0x0001		2 Byte CRC	
Read Ch4	0x00...0x3F	0x03	0x0004		0x0001		2 Byte CRC	
Read Ch5	0x00...0x3F	0x03	0x0005		0x0001		2 Byte CRC	
Read Ch6	0x00...0x3F	0x03	0x0006		0x0001		2 Byte CRC	
Read Ch7	0x00...0x3F	0x03	0x0007		0x0001		2 Byte CRC	
Read Ch8	0x00...0x3F	0x03	0x0008		0x0001		2 Byte CRC	
Read Ch1..Ch8	0x00...0x3F	0x03	0x0001		0x0008		2 Byte CRC	

15. คำถาม ให้ปรับเปลี่ยนโปรแกรม Arduino ESP32 เพื่ออ่านสถานะปัจจุบันของ Relay

Test 3/4. Ctrl Relay and Read Switch - Modbus RTU Relay4/In4

1. Introduction



MODBUS RTU RELAY4/IN4 (C-YA-A-00288) [590.00.- ยังไม่รวม Vat]

MODBUS RTU RELAY4/IN4 ...เป็นบอร์ดที่มี RELAY ON/OFF จำนวน 4 ตัว และ INPUT TTL (3.3V) จำนวน 4 INPUT รับคำสั่งการทำงานผ่านทางวิธีการสื่อสาร RS485 แบบ HALF DUPLEX มี PROTOCOL ในการสั่งงานแบบ MODBUS RTU ต่อใช้งานได้ในระยะไกลถึง 1.2 กิโลเมตร (แยกแหล่งจ่ายไฟ)

2. คุณสมบัติ

- : 4 INPUT RELAY แบบ 2 CONTACT (NO, NC, COM)
 - DC CONTACT RATING 10A/30VDC
 - AC CONTACT RATEING 10A/220VAC
- : 4 INPUT LOGIC TTL 3.3V (INPUT ต่อตรงกับ MCU ของบอร์ด ห้ามเกิน 3.3 V)
- : สื่อสารสั่งงานทาง RS485 ในแบบคำสั่ง MODBUS RTU
- : กำหนดค่า ADDRESS ของบอร์ดได้ 32 ตำแหน่ง จากคำสั่งในการตั้งค่า
- : BAUDRATE 9600 bps, DATA 8 BIT, NONE PARITY, 1 STOP BIT I
- : มีคำสั่งรูปแบบ WRITE ON/OFF, READ อ่านสถานะ INPUT
- : ใช้กับแหล่งจ่ายไฟ 12VDC ขั้วแบบ SCREW TERMINAL 2 PIN และ ขั้ว DC JACK
- : ขนาด 7.7 x 6.7 cm.

3. การกำหนด Address

ตามปรกติแล้วค่า Slave Device Address จะถูกกำหนดค่ามาตรฐานเป็น 0x01 ไว้เมื่อต้องการใช้งาน บอร์ดร่วมกันมากกว่า 1 บอร์ดจึงมีความจำเป็นต้องกำหนดค่า Slave Address ให้โดยใช้คำสั่งของฟังก์ชัน 0x06 ในการกำหนดและต้องกระทำในขณะที่ทำการเชื่อมต่อบอร์ดไวน์บัส RS485 เพียง 1 บอร์ดเท่านั้น ไม่เช่นนั้นแล้วบอร์ดทุกบอร์ดจะถูกกำหนดให้มีค่าเหมือนกัน

ตัวอย่างการกำหนดตำแหน่ง Slave Device Address เป็น 0x01

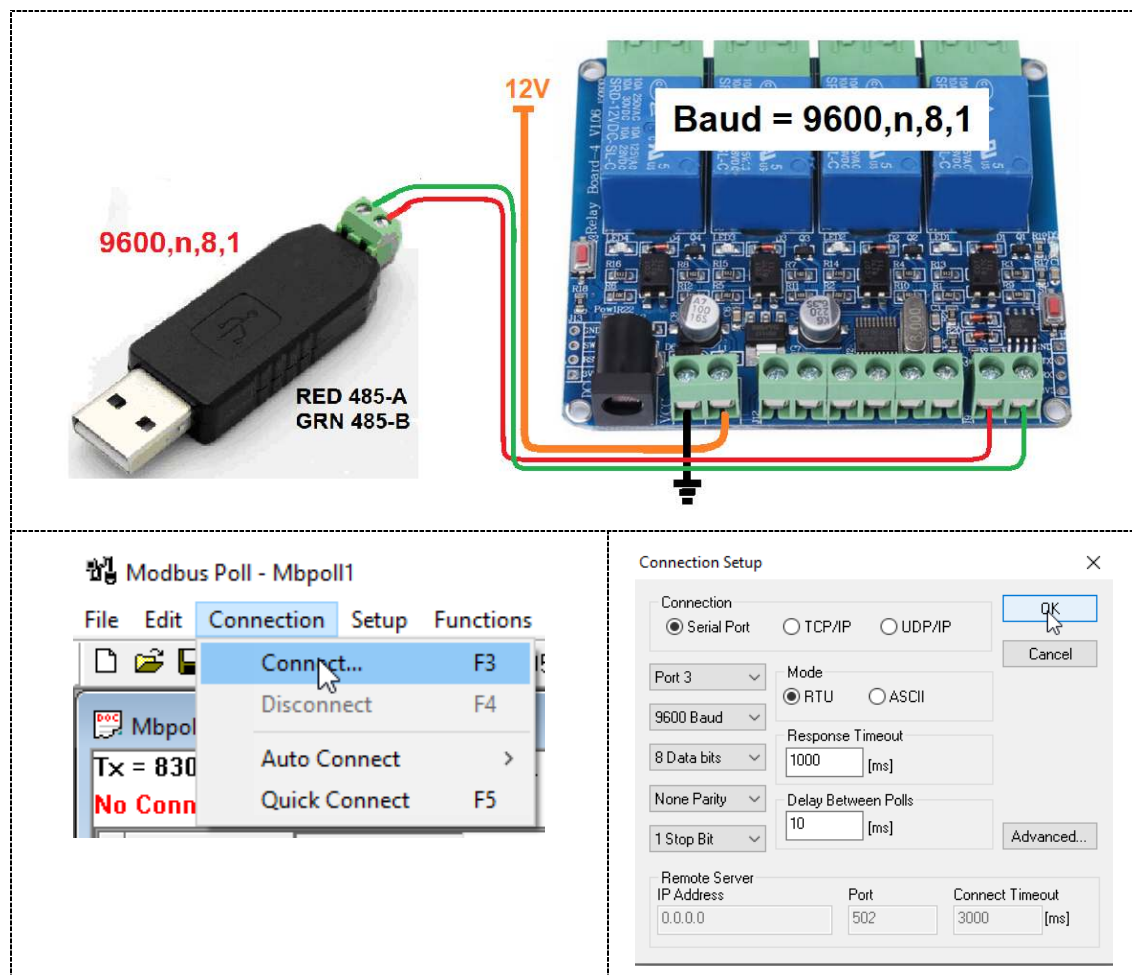
Tx : 00 06 40 00 00 01 5C 1B
Rx : 01 06 00 00 00 01 48 0A

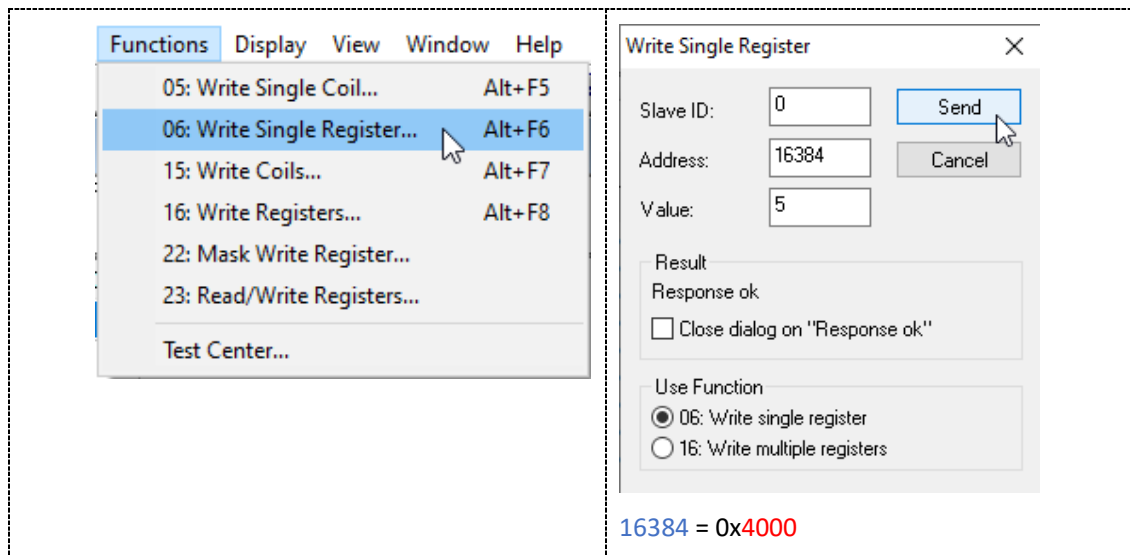
ตัวอย่างการกำหนดตำแหน่ง Slave Device Address เป็น 0x02

Tx : 00 06 40 00 00 02 1C 1A
Rx : 02 06 00 00 00 02 08 38

Test 0a/4 – ใช้ Modbus Poll ในการกำหนด Address Device

- ต้องต่ออุปกรณ์เพียงตัวเดียวในสายสื่อสาร มิฉะนั้นจะกลายเป็นว่าอุปกรณ์ทั้งหมดมี ID เดียวกัน
- เป็นการกำหนด Address เป็น 5





4. การควบคุม RELAY

การควบคุมการทำงานของ Relay ในบอร์ด Modbus RTU Relay4 / In4 จะใช้คำสั่งของ ฟังก์ชัน Write Single Coil (0x05) ในการสั่งงาน โดยเฟรมคำสั่งจะประกอบไปด้วยข้อมูลจำนวน n ไบต์ ซึ่งมีรูปแบบดังนี้

Byte No	1	2	3	4	5	6	7	8
Modbus	Slave ID	Function	Address		Data		CRC Check	
Function	Device ID	Function	Relay(0..3)		Command ON/OFF		CRC Check	
ON RELAY1	00-FF	05	00	00	01/FF	00	2 Byte CRC	
ON RELAY2	00-FF	05	00	01	01/FF	00	2 Byte CRC	
ON RELAY3	00-FF	05	00	02	01/FF	00	2 Byte CRC	
ON RELAY4	00-FF	05	00	03	01/FF	00	2 Byte CRC	
ON All RELAY	00-FF	05	00	FF	FF	00	2 Byte CRC	
OFF RELAY1	00-FF	05	00	00	00	00	2 Byte CRC	
OFF RELAY2	00-FF	05	00	01	00	00	2 Byte CRC	
OFF RELAY3	00-FF	05	00	02	00	00	2 Byte CRC	
OFF RELAY4	00-FF	05	00	03	00	00	2 Byte CRC	
OFF All RELAY	00-FF	05	00	FF	00	00	2 Byte CRC	

ตัวอย่างการสั่งงาน Relay Device ID = 01

ON RELAY1	Tx : 01 05 00 00 FF 00 8C 3A	Rx : 01 05 00 00 FF 00 8C 3A
ON RELAY2	Tx : 01 05 00 01 FF 00 DD FA	Rx : 01 05 00 01 FF 00 DD FA
ON RELAY3	Tx : 01 05 00 02 FF 00 2D FA	Rx : 01 05 00 02 FF 00 2D FA
ON RELAY4	Tx : 01 05 00 03 FF 00 7C 3A	Rx : 01 05 00 03 FF 00 7C 3A
OFF RELAY1	Tx : 01 05 00 00 00 00 CD CA	Rx : 01 05 00 00 00 00 CD CA
OFF RELAY2	Tx : 01 05 00 01 00 00 9C 0A	Rx : 01 05 00 01 00 00 9C 0A
OFF RELAY3	Tx : 01 05 00 02 00 00 6C 0A	Rx : 01 05 00 02 00 00 6C 0A
OFF RELAY4	Tx : 01 05 00 03 00 00 3D CA	Rx : 01 05 00 03 00 00 3D CA

ID=5, On Relay 0	ID=5, Off Relay 0

5. การสั่งอ่านค่า Input

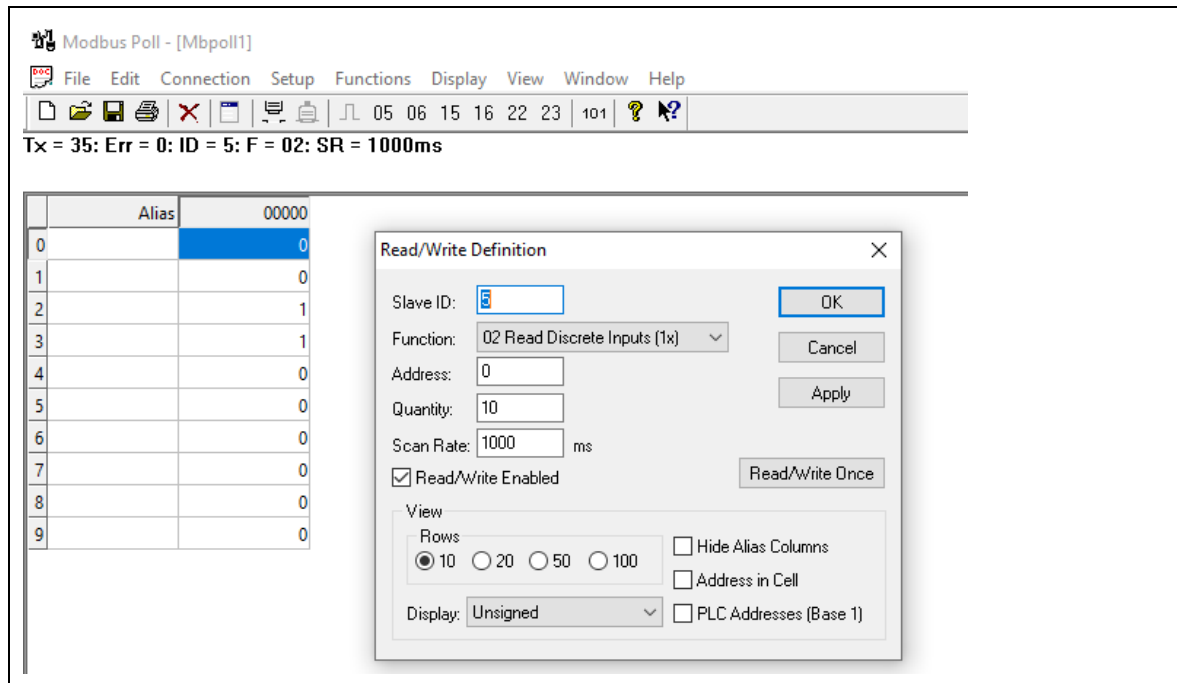
ในการอ่านค่า Input ทั้ง 4 ช่อง นั้นจะใช้ฟังก์ชัน 0x02 ในการอ่านค่า โดยสัญญาณ Input ทั้ง 2 ช่อง สามารถรับสัญญาณภายนอกแบบ TTL Logic ขนาด 3.3V เท่านั้น หรือใช้การ ต่อ Input ผ่านหน้าสัมผัสสวิตช์ หรือหน้าสัมผัสสรีเลย์ลง GND ที่บอร์ดเท่านั้น ถ้าจะต่อ Input ที่มีระดับสัญญาณสูงกว่าต้องทำการแปลงสัญญาณให้เป็น 3.3V TTL เสียก่อน สำหรับการสั่ง อ่านค่า Input ทำได้ ดังตัวอย่าง ตัวอย่างการสั่งอ่านค่า Input จาก Device ID = 01

TX : 01 02 00 00 00 04 79 C9

- **01 Slave-01**
- **02 Function Code : Read Discrete Input**
- **00 00 : Start Address 00 00**
- **00 04 : 4 Address Read**
- **79 C9 : CRC**

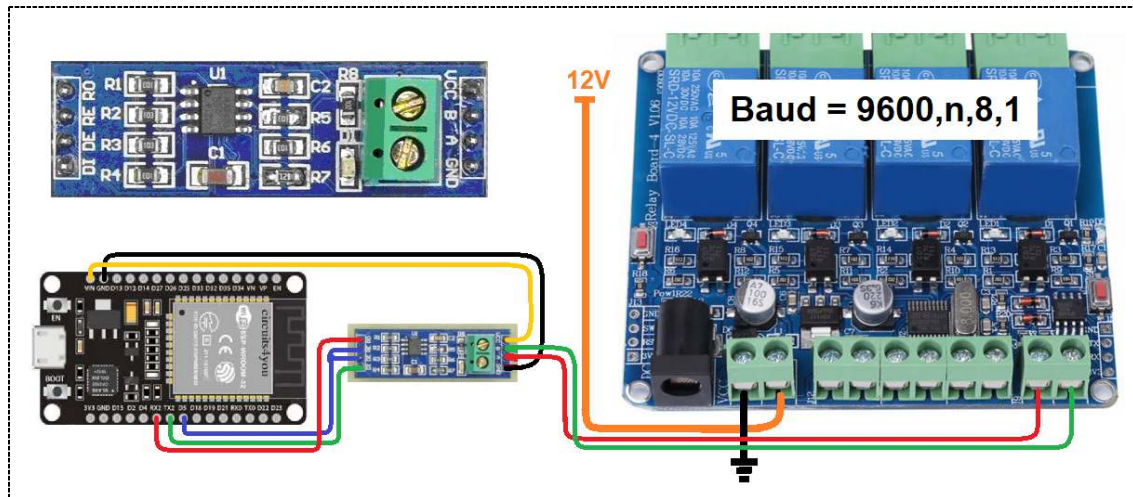
RX : 01 02 01 00 A1 88

- **01 : Slave 01**
- **02 : Function Code : Read Discrete Input**
- **01 : 1 Byte Result**
- **00 : Result(D7-D6-D5-D4-D3-D2-D1-D0 : 0000????)**
 - **00: All Input OFF**
 - **01: Input1 ON**
 - **02: Input2 ON**
 - **04: Input3 ON**
 - **08: Input4 ON**
 - **0F: Input1-4 ON**
- **A1 A8 : CRC**



Test 3a/4 – ทดสอบโดย Arduino – Relay Control

6. Circuit and Wiring Relay Control



7. Test Relay Control V1.0

```
#define RS485Transmit  HIGH
#define RS485Receive  LOW
#define RS485Control  4  //RS485 Direction control
#define Pin_LEDMonitor 2

byte Board_ID = 0x05; // ID = 5
byte Mdbbs_Cmd = 0x05; // Command 05
byte H_RelayID = 0x00;
byte L_RelayID = 0x00;
byte Relay_On  = 0x01; // On = 0100
byte Relay_Off = 0x00; // Off = 0000
byte OnOff_Dly = 0x00;
byte HByte_CRC = 00;
byte LByte_CRC = 00;

int StepConut = 0;
byte Echo[20];

void setup() {
  pinMode(Pin_LEDMonitor, OUTPUT);
  pinMode(RS485Control, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  digitalWrite(RS485Control, RS485Receive);
  Serial.println("Start Test MODBUS RTU");
}

uint16_t CRC16_Update(uint16_t tempCRC, uint8_t inData) {
  tempCRC ^= inData;
  for (int i = 0; i < 8; ++i)
    if (tempCRC & 1) tempCRC = (tempCRC >> 1) ^ 0xA001;
    else tempCRC = (tempCRC >> 1);
  return tempCRC;
}

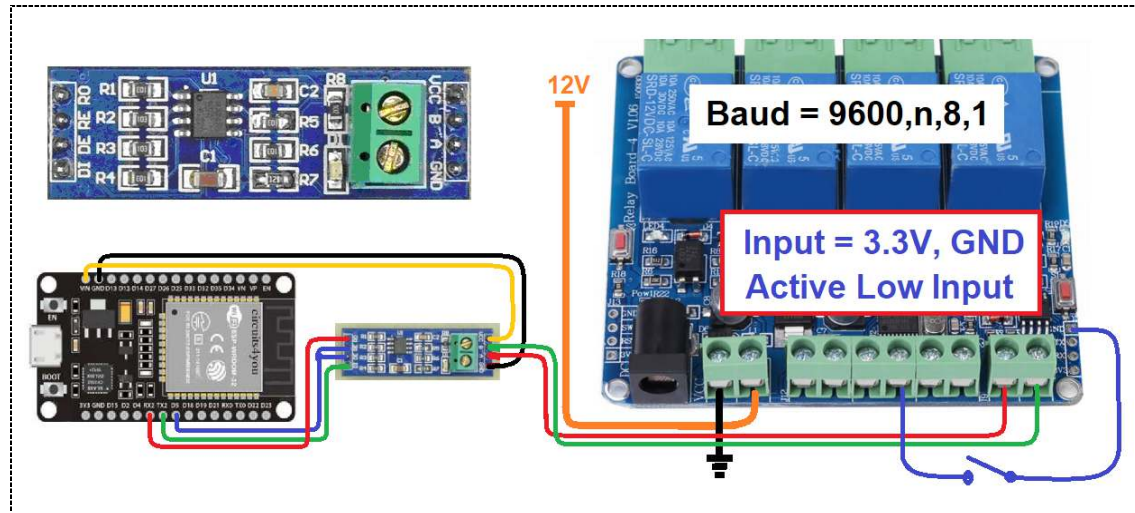
uint16_t SendByte_CRCUpdate(uint16_t tempCRC, uint8_t inData) {
  Serial2.write(inData);
  if (inData < 0x10) Serial.print("0");
  Serial.print(inData, HEX);
  Serial.print(" ");
  tempCRC = CRC16_Update(tempCRC, inData);
  return tempCRC;
}

void RTU_RelayCtrl(int rly_ID, byte rly_Cmd) {
  uint16_t Calc_CRC = 0xffff; // the initial value
  H_RelayID = highByte(rly_ID);
  L_RelayID = lowByte(rly_ID);
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit); delay(10);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Board_ID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Mdbbs_Cmd);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, H_RelayID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, L_RelayID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, rly_Cmd);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, OnOff_Dly);
  HByte_CRC = highByte(Calc_CRC);
  LByte_CRC = lowByte(Calc_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, LByte_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, HByte_CRC);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);
  Serial.println();
}

void loop() {
  RTU_RelayCtrl(0, Relay_On); delay(3000);
  RTU_RelayCtrl(1, Relay_On); delay(3000);
  RTU_RelayCtrl(2, Relay_On); delay(3000);
  RTU_RelayCtrl(3, Relay_On); delay(3000);
  RTU_RelayCtrl(0, Relay_Off); delay(3000);
  RTU_RelayCtrl(1, Relay_Off); delay(3000);
  RTU_RelayCtrl(2, Relay_Off); delay(3000);
  RTU_RelayCtrl(3, Relay_Off); delay(3000);
}
```

Test 3b/4 – ทดสอบโดย Arduino – Read Switch

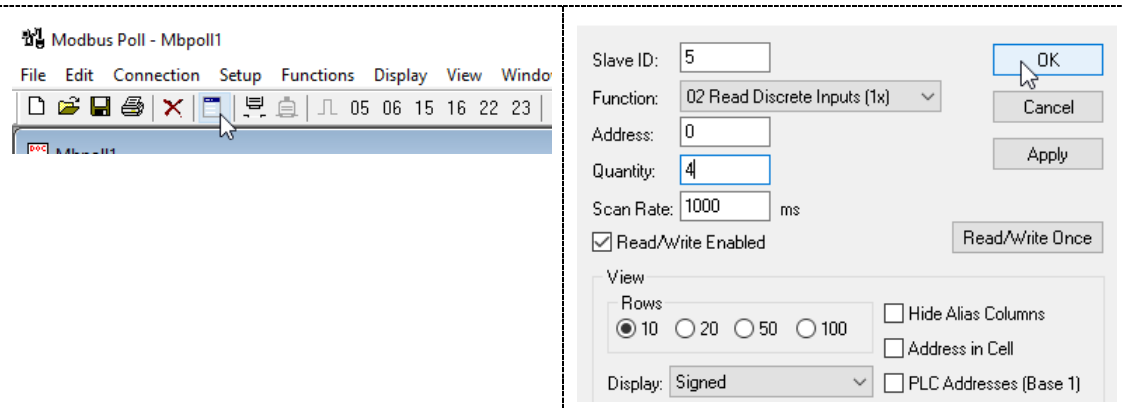
8. Circuit and Wiring Input Test



9. Read from Modbus Poll

Function 02 – Read Discreat Input

Quantity = 4



	Alias	00000
0		0
1		1
2		1
3		0

TX		05	02	00	00	00	04	78	4D	>> Read Start 0000 > 4 Byte
RX		05	02	01	06	20	BA			>> Ask 1 Byte 06 =
		0000	0110							

10. Test Relay Control V1.0

```

#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 4 //RS485 Direction control
#define Pin_LEDMonitor 2

byte Board_ID = 0x05; // ID
byte Mdbcs_Cmd = 0x05; // Command 06
byte H_RelayID = 0x00;
byte L_RelayID = 0x00;
byte Relay_On = 0x01; // On = 0100
byte Relay_Off = 0x00; // Off = 0000
byte OnOff_Dly = 0x00;
byte HByte_CRC = 00;
byte LByte_CRC = 00;

int Wr_Index, StepConut = 0;
byte Echo[20];

void setup() {
  pinMode(Pin_LEDMonitor, OUTPUT);
  pinMode(RS485Control, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  digitalWrite(RS485Control, RS485Receive);
  Serial.println("Start Test MODBUS RTU");
}

uint16_t CRC16_Update(uint16_t tempCRC, uint8_t inData) {
  tempCRC ^= inData;
  for (int i = 0; i < 8; ++i)
    if (tempCRC & 1) tempCRC = (tempCRC >> 1) ^ 0xA001;
    else tempCRC = (tempCRC >> 1);
  return tempCRC;
}

uint16_t SendByte_CRCUpdate(uint16_t tempCRC, uint8_t inData) {
  Serial2.write(inData);
  if (inData < 0x10) Serial.print("0");
  Serial.print(inData, HEX);
  Serial.print(" ");
  tempCRC = CRC16_Update(tempCRC, inData);
  return tempCRC;
}

void RTU_RelayCtrl(int rly_ID, byte rly_Cmd) {
  uint16_t Calc_CRC = 0xffff; // the initial value
  H_RelayID = highByte(rly_ID);
  L_RelayID = lowByte(rly_ID);
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit); delay(10);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Board_ID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Mdbcs_Cmd);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, H_RelayID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, L_RelayID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, rly_Cmd);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, OnOff_Dly);
  HByte_CRC = highByte(Calc_CRC);
  LByte_CRC = lowByte(Calc_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, LByte_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, HByte_CRC);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);
  Serial.println();
}

void RTU_ReadBoard(void) {
  uint16_t Calc_CRC = 0xffff; // the initial value
  digitalWrite(Pin_LEDMonitor, HIGH);
  digitalWrite(RS485Control, RS485Transmit); delay(10);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, Board_ID);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, 0x02);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, 0x00);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, 0x00);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, 0x00);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, 0x00);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, 0x04);
  HByte_CRC = highByte(Calc_CRC);
  LByte_CRC = lowByte(Calc_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, LByte_CRC);
  Calc_CRC = SendByte_CRCUpdate(Calc_CRC, HByte_CRC);
  delay(10);
  digitalWrite(RS485Control, RS485Receive);
  digitalWrite(Pin_LEDMonitor, LOW);

  Wr_Index = 0;
  for (long int i = 0; i < 600000; i++) {
    if (Serial2.available() > 0) {
      Echo[Wr_Index] = Serial2.read();
    }
  }
}

```



```

    if (Wr_Index > 8) i = 999999;
    Wr_Index++;
  }
  delayMicroseconds(5);
}

Serial.print(" >> ");
for (int i = 0; i < 10; i++) {
  if (Echo[i] < 0x10) Serial.print("0");
  Serial.print(Echo[i], HEX);
  Serial.print(" ");
}
Serial.println();
}

void loop() {
  RTU_RelayCtrl(0, Relay_On); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(0, Relay_Off); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(1, Relay_On); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(1, Relay_Off); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(2, Relay_On); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(2, Relay_Off); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(3, Relay_On); delay(1500); RTU_ReadBoard(); delay(1500);
  RTU_RelayCtrl(3, Relay_Off); delay(1500); RTU_ReadBoard(); delay(1500);
}

```

```

01 02 00 00 00 04 79 C9 >> 01 02 01 00 A1 88 FF 00 00 00
01 05 00 00 00 00 CD CA
01 02 00 00 00 04 79 C9 >> 01 02 01 02 20 49 FF 00 00 00
01 05 00 01 00 00 9C 0A
01 02 00 00 00 04 79 C9 >> 01 02 01 08 A0 4E FF 00 00 00
01 05 00 02 00 00 6C 0A

```

Test 3c/4 – ทดสอบโดย Arduino + Library → Read Switch. Control Relay

11. Test Control Relay 0-3 – with Arduino Library

```

#include "ModbusMaster.h" //https://github.com/4-20ma/ModbusMaster

#define Slave_ID 5
#define MAX485_RE_NEG 4
#define RX_PIN 16
#define TX_PIN 17

ModbusMaster modbus;

void preTransmission() {
  digitalWrite(MAX485_RE_NEG, HIGH); //Switch to transmit data
}

void postTransmission() {
  digitalWrite(MAX485_RE_NEG, LOW); //Switch to receive data
}

void setup() {
  pinMode(MAX485_RE_NEG, OUTPUT);
  digitalWrite(MAX485_RE_NEG, LOW);
  Serial.begin(115200, SERIAL_8N1);
  Serial2.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  modbus.begin(Slave_ID, Serial2);
  modbus.preTransmission(preTransmission);
  modbus.postTransmission(postTransmission);
}

void loop() {
  uint8_t result;
  result = modbus.writeSingleRegister(0, 0x0100); getResultMsg(&modbus, result); delay(1000); // On Relay0
  result = modbus.writeSingleRegister(0, 0x0000); getResultMsg(&modbus, result); delay(1000); // Off Relay0
  result = modbus.writeSingleRegister(1, 0x0100); getResultMsg(&modbus, result); delay(1000); // On Relay1
  result = modbus.writeSingleRegister(1, 0x0000); getResultMsg(&modbus, result); delay(1000); // Off Relay1
  result = modbus.writeSingleRegister(2, 0x0100); getResultMsg(&modbus, result); delay(1000); // On Relay2
  result = modbus.writeSingleRegister(2, 0x0000); getResultMsg(&modbus, result); delay(1000); // Off Relay2
  result = modbus.writeSingleRegister(3, 0x0100); getResultMsg(&modbus, result); delay(1000); // On Relay3
  result = modbus.writeSingleRegister(3, 0x0000); getResultMsg(&modbus, result); delay(1000); // Off Relay3
}

bool getResultMsg(ModbusMaster *node, uint8_t result) {

```



```

String tmpstr2 = "\r\n";
switch (result) {
  case node->ku8MBSuccess:
    tmpstr2 += "Compleat";
    Serial.println(tmpstr2);
    return true;
    break;
  case node->ku8MBIllegalFunction:
    tmpstr2 += "Illegal Function";
    break;
  case node->ku8MBIllegalDataAddress:
    tmpstr2 += "Illegal Data Address";
    break;
  case node->ku8MBIllegalDataValue:
    tmpstr2 += "Illegal Data Value";
    break;
  case node->ku8MBSlaveDeviceFailure:
    tmpstr2 += "Slave Device Failure";
    break;
  case node->ku8MBInvalidSlaveID:
    tmpstr2 += "Invalid Slave ID";
    break;
  case node->ku8MBInvalidFunction:
    tmpstr2 += "Invalid Function";
    break;
  case node->ku8MBResponseTimedOut:
    tmpstr2 += "Response Timed Out";
    break;
  case node->ku8MBInvalidCRC:
    tmpstr2 += "Invalid CRC";
    break;
  default:
    tmpstr2 += "Unknown error: " + String(result);
    break;
}
Serial.println(tmpstr2);
return false;
}

```

12. Test Control Relay 0-3 and Monitor – with Arduino Library

```

#include <ModbusMaster.h>
#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 4 //RS485 Direction control
#define Pin_LEDMonitor 2

#define Slave_ID 5
ModbusMaster node;

void preTransmission() {
  digitalWrite(RS485Control, RS485Transmit);
}

void postTransmission() {
  digitalWrite(RS485Control, RS485Receive);
}

void setup() {
  pinMode(RS485Control, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  postTransmission();
  node.begin(Slave_ID, Serial2); // Set Modbus ID, Communication
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
}

int Read_Relay4In4(void)
{ uint8_t result, xValue = 0xff;
  uint16_t data[6];
  result = node.readDiscreteInputs(0, 4); // Start=0, nByte=4
  if (result == node.ku8MBSuccess) {
    xValue = node.getResponseBuffer(0); // Read return from 0_Byte
  }
}

```

COM3

```

1000
0000
1100
1111
1110
1110
0110
0110

```

```
    return xValue;
}

void BinDisplay(int DataIn) {
    if (DataIn == 0xff)
        Serial.println("Read Error");
    else {
        Serial.print(DataIn >> 3 & 1);
        Serial.print(DataIn >> 2 & 1);
        Serial.print(DataIn >> 1 & 1);
        Serial.print(DataIn >> 0 & 1);
        Serial.println();
    }
}

void loop() {
    node.writeSingleRegister(0, 0x0100); delay(1000); // On Relay0
    BinDisplay(Read_Relay4In4());
    node.writeSingleRegister(0, 0x0000); delay(1000); // Off Relay0

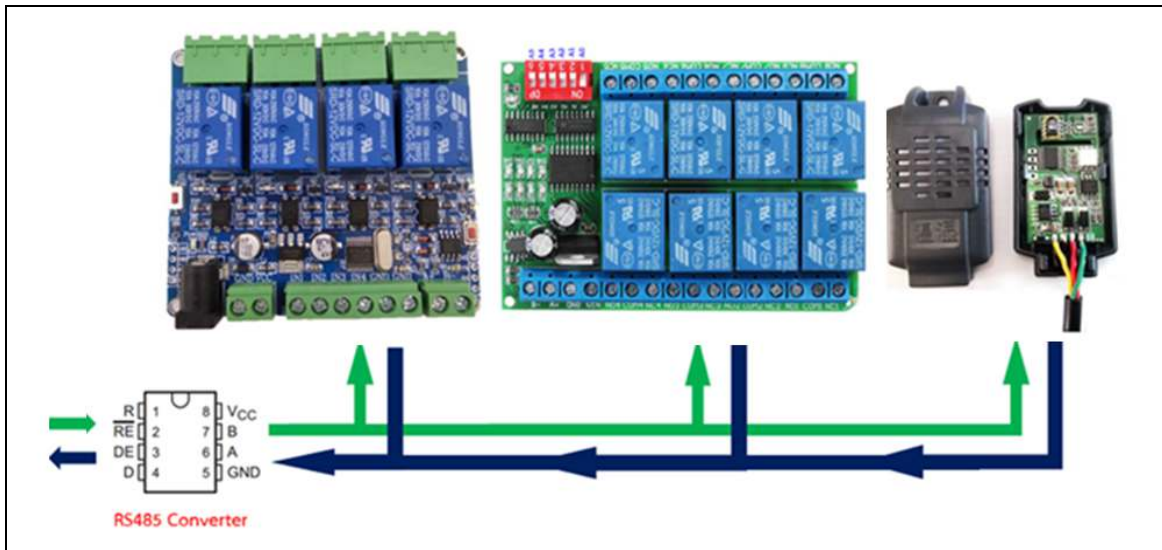
    node.writeSingleRegister(1, 0x0100); delay(1000); // On Relay1
    BinDisplay(Read_Relay4In4());
    node.writeSingleRegister(1, 0x0000); delay(1000); // Off Relay1

    node.writeSingleRegister(2, 0x0100); delay(1000); // On Relay2
    BinDisplay(Read_Relay4In4());
    node.writeSingleRegister(2, 0x0000); delay(1000); // Off Relay2

    node.writeSingleRegister(3, 0x0100); delay(1000); // On Relay3
    BinDisplay(Read_Relay4In4());
    node.writeSingleRegister(3, 0x0000); delay(1000); // Off Relay3
}
```

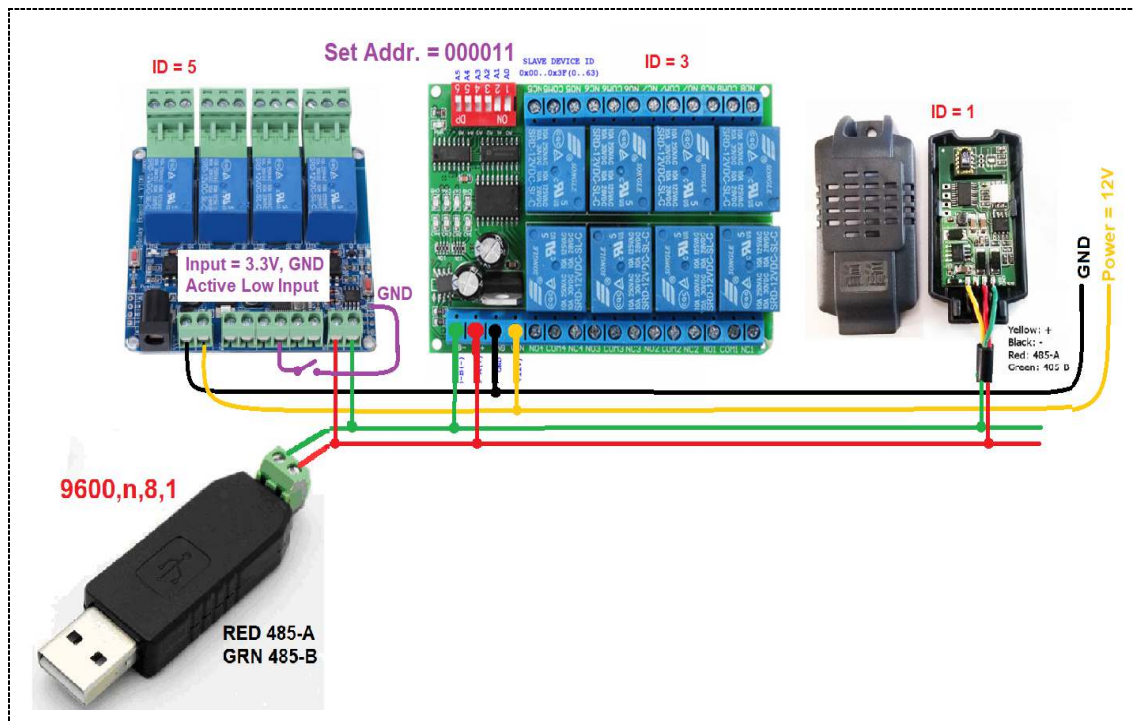
Test 4/4. Read and Write 3 Device

1. ภาพวงจรโดยรวม



Test 4a/4 – ทดสอบโดย Modbus Poll

2. ภาพการต่อวงจร



3. Arduino Code and Result

Read ID#1

Read/Write Definition

Slave ID: 1

Function: 03 Read Holding Registers (4x)

Address: 0

Quantity: 2

Scan Rate: 1000 ms

☒ Read/Write Enabled

View

Rows: ☒ 10 ☐ 20 ☐ 50 ☐ 100

Display: Unsigned

☐ Hide Alias Columns

☐ Address in Cell

☐ PLC Addresses (Base 1)

OK

Cancel

Apply

Read/Write Once

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions Display View Window Help

Mbpoll1

Tx = 84: Err = 5: ID = 1: F = 03: SR = 1

	Alias	
		00000
0		268
1		742

Signed Alt+S

☒ Unsigned Alt+U

Hex Alt+H

Binary Alt+B

Long

Long Inverse

Float

Float Inverse

Read ID#5

Read/Write Definition [X]

Slave ID:

Function: 02 Read Discrete Inputs (1x) ▾

Address:

Quantity:

Scan Rate: ms

☒ Read/Write Enabled

View

Rows
☒ 10 ☐ 20 ☐ 50 ☐ 100

☐ Hide Alias Columns
☐ Address in Cell
☐ PLC Addresses (Base 1)

Display: ▾

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions **Display** View Window Help

Mbpoll1

Tx = 7: Err = 0: ID = 5: F = 02: SR = 10

	Alias	
0		0
1		1
2		1
3		0

Signed Alt+S
✓ Unsigned Alt+U
Hex Alt+H
Binary Alt+B
Long
Long Inverse
Float
Float Inverse
Double
Double Inverse

Write ID#3

On Relay 1 from [1:8] → 256 = 0x0100

Off Relay 1 from [1:8] → 512 → 0x0200

Write Single Register ✕

Slave ID:

Address:

Value:

Result
Response ok
☐ Close dialog on "Response ok"

Use Function
☒ 06: Write single register
☐ 16: Write multiple registers

Write Single Register ✕

Slave ID:

Address:

Value:

Result
Response ok
☐ Close dialog on "Response ok"

Use Function
☒ 06: Write single register
☐ 16: Write multiple registers

Write ID#5

On Relay 0 from [0:3]

Off Relay 0 from [0:3]

Write Single Coil ✕

Slave ID:

Address:

Value
☒ On ☐ Off

Result
Response ok
☐ Close dialog on "Response ok"

Use Function
☒ 05: Write single coil
☐ 15: Write multiple coils

Write Single Coil ✕

Slave ID:

Address:

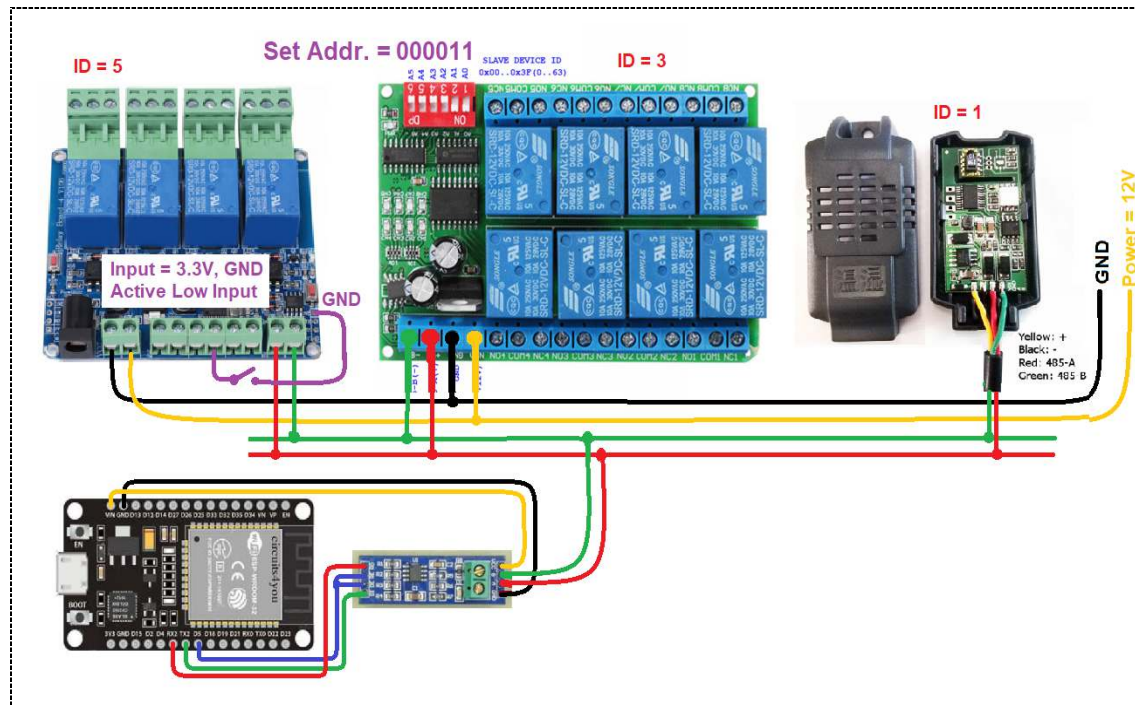
Value
☐ On ☒ Off

Result
Response ok
☐ Close dialog on "Response ok"

Use Function
☒ 05: Write single coil
☐ 15: Write multiple coils

Test 4b/4 – ทดสอบโดย Arduino with Library

4. ภาพการต่อวงจร



5. Arduino Code and Result – Read All Input

```
#include <ModbusMaster.h>
#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 4 //RS485 Direction control
#define Pin_LEDMonitor 2
#define Slave_Sensor_ID 1
#define Slave_Relay8_ID 3
#define Slave_Ry4In4_ID 5

int state = 0;
float CTemp, Hudmid;
bool DgInput0, DgInput1, DgInput2, DgInput3;

ModbusMaster node_Sensor;
ModbusMaster node_Relay8;
ModbusMaster node_Ry4In4;

void preTransmission() {
  digitalWrite(RS485Control, RS485Transmit);
}

void postTransmission() {
  digitalWrite(RS485Control, RS485Receive);
}

void setup() {
  pinMode(RS485Control, OUTPUT);
  pinMode(Pin_LEDMonitor, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  postTransmission();
  node_Sensor.begin(Slave_Sensor_ID, Serial2); // Modbus slave ID=1
  node_Sensor.preTransmission(preTransmission);
  node_Sensor.postTransmission(postTransmission);
  node_Relay8.begin(Slave_Relay8_ID, Serial2); // Modbus slave ID=3
  node_Relay8.preTransmission(preTransmission);
  node_Relay8.postTransmission(postTransmission);
  node_Ry4In4.begin(Slave_Ry4In4_ID, Serial2); // Modbus slave ID=5
  node_Ry4In4.preTransmission(preTransmission);
}
```

```

node_Ry4In4.postTransmission(postTransmission);
}

void ReadTemperature(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Sensor.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 2 registers starting at 0x0000
  result = node_Sensor.readInputRegisters(0x0000, 2); // From=0, nByte=2
  if (result == node_Sensor.ku8MBSuccess) {
    CTempp = node_Sensor.getResponseBuffer(0x00) / 10.0f;
    Hudmid = node_Sensor.getResponseBuffer(0x01) / 10.0f;
  }
}

void ReadDigitalInput(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Ry4In4.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 4 registers starting at 0x0000
  result = node_Ry4In4.readDiscretInputs(0, 4); // Start=0, nByte=4
  if (result == node_Ry4In4.ku8MBSuccess) {
    int DgTemp = node_Ry4In4.getResponseBuffer(0x00);
    DgInput3 = (DgTemp >> 3) & 1;
    DgInput2 = (DgTemp >> 2) & 1;
    DgInput1 = (DgTemp >> 1) & 1;
    DgInput0 = (DgTemp >> 0) & 1;
  }
}

void loop() {
  ReadTemperature();
  ReadDigitalInput();
  Serial.print("\n Temp('C): "); Serial.print(CTempp, 2);
  Serial.print(", Humid(%): "); Serial.print(Hudmid, 2);
  Serial.print(", Sensor[0:3]: "); Serial.print(DgInput3);
  Serial.print("-"); Serial.print(DgInput2);
  Serial.print("-"); Serial.print(DgInput1);
  Serial.print("-"); Serial.print(DgInput0);
  delay(2000);
}

```

COM7

Send

```

Temp('C): 27.20, Humid(%): 72.30, Sensor[0:3]: 0-1-1-1
Temp('C): 27.20, Humid(%): 72.20, Sensor[0:3]: 0-1-1-1
Temp('C): 27.20, Humid(%): 72.20, Sensor[0:3]: 0-1-1-1
Temp('C): 27.20, Humid(%): 72.10, Sensor[0:3]: 0-1-1-1
Temp('C): 27.20, Humid(%): 72.00, Sensor[0:3]: 0-1-1-1
Temp('C): 27.20, Humid(%): 71.90, Sensor[0:3]: 0-1-1-1
Temp('C): 27.20, Humid(%): 71.80, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 71.80, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 71.90, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.00, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.10, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.20, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.30, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.30, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.40, Sensor[0:3]: 0-1-1-1
Temp('C): 27.30, Humid(%): 72.40, Sensor[0:3]: 0-1-1-1

```

☒ Autoscroll ☐ Show timestamp

Carriage return 9600 baud Clear output

6. Arduino Code and Result – Read/Write All Board

```

#include <ModbusMaster.h>
#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 4 //RS485 Direction control
#define Pin_LEDMonitor 2
#define Slave_Sensor_ID 1
#define Slave_Relay8_ID 3
#define Slave_Ry4In4_ID 5

int state = 0;
float CTemp, Hudmid;
bool DgInput0, DgInput1, DgInput2, DgInput3;

ModbusMaster node_Sensor;
ModbusMaster node_Relay8;
ModbusMaster node_Ry4In4;

void preTransmission() {
  digitalWrite(RS485Control, RS485Transmit);
}

void postTransmission() {
  digitalWrite(RS485Control, RS485Receive);
}

void setup() {
  pinMode(RS485Control, OUTPUT);
  pinMode(Pin_LEDMonitor, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  postTransmission();
  node_Sensor.begin(Slave_Sensor_ID, Serial2); // Modbus slave ID=1
  node_Sensor.preTransmission(preTransmission);
  node_Sensor.postTransmission(postTransmission);
  node_Relay8.begin(Slave_Relay8_ID, Serial2); // Modbus slave ID=3
  node_Relay8.preTransmission(preTransmission);
  node_Relay8.postTransmission(postTransmission);
  node_Ry4In4.begin(Slave_Ry4In4_ID, Serial2); // Modbus slave ID=5
  node_Ry4In4.preTransmission(preTransmission);
  node_Ry4In4.postTransmission(postTransmission);
}

void ReadTemperature(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Sensor.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 2 registers starting at 0x0000
  result = node_Sensor.readInputRegisters(0x0000, 2); // From=0, nByte=2
  if (result == node_Sensor.ku8MBSuccess) {
    CTemp = node_Sensor.getResponseBuffer(0x00) / 10.0f;
    Hudmid = node_Sensor.getResponseBuffer(0x01) / 10.0f;
  }
}

void ReadDigitalInput(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Ry4In4.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 4 registers starting at 0x0000
  result = node_Ry4In4.readDiscreteInputs(0, 4); // Start=0, nByte=4
  if (result == node_Ry4In4.ku8MBSuccess) {
    int DgTemp = node_Ry4In4.getResponseBuffer(0x00);
    DgInput3 = (DgTemp >> 3) & 1;
    DgInput2 = (DgTemp >> 2) & 1;
    DgInput1 = (DgTemp >> 1) & 1;
    DgInput0 = (DgTemp >> 0) & 1;
  }
}

void RelayControl(int inputCase) {
  int rnMode = inputCase / 10;
  int nRelay = inputCase % 10;
  if (rnMode == 81) node_Relay8.writeSingleRegister(nRelay, 0x0100); // On RelayX
  if (rnMode == 80) node_Relay8.writeSingleRegister(nRelay, 0x0200); // Off RelayX
}

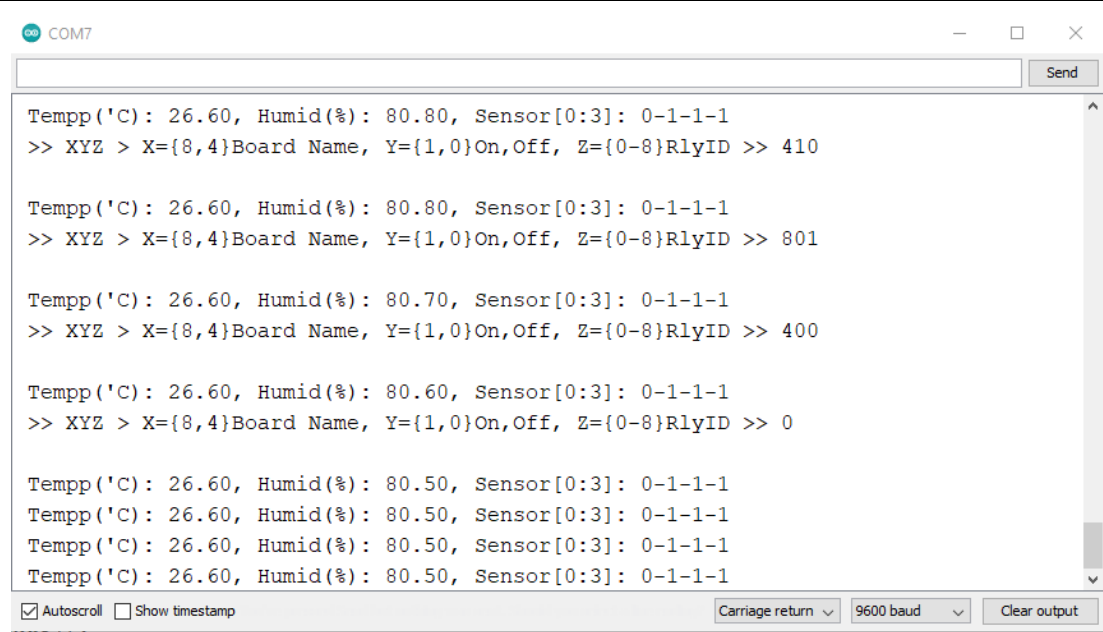
```

```

if (rnMode == 41) node_Ry4In4.writeSingleRegister(nRelay, 0x0100); // On RelayX
if (rnMode == 40) node_Ry4In4.writeSingleRegister(nRelay, 0x0000); // Off RelayX
}

void loop() {
  ReadTemperature();
  ReadDigitalInput();
  Serial.print("\n Tempp('C): "); Serial.print(CTempp, 2);
  Serial.print(", Humid(%): "); Serial.print(Hudmid, 2);
  Serial.print(", Sensor[0:3]: "); Serial.print(DgInput3);
  Serial.print("-"); Serial.print(DgInput2);
  Serial.print("-"); Serial.print(DgInput1);
  Serial.print("-"); Serial.print(DgInput0);
  if (Serial.available() > 0) {
    int DataInput = Serial.parseInt();
    Serial.print("\n >> XYZ > X={8,4}Board Name, Y={1,0}On,Off, Z={0-8}RlyID >> ");
    Serial.println(DataInput);
    RelayControl(DataInput);
  }
  delay(2000);
}

```



Command > XYZ	400	812
• X={8,4}Board Name	Relay4/In4	Relay8
• Y={1,0}On,Off	Off	On
• Z={0-8}RlyID	Relay0	Relay2

3/4: -- การโปรแกรมใช้งาน PLC แบบ Single Controller

3.1 Schneider - Modicon M221 Logic Controller



Modicon M221 ออลอินวันคอนโทรลเลอร์รุ่นใหม่ล่าสุดจากชไนเดอร์ อิเล็คทริค มาพร้อมประสิทธิภาพที่ดีที่สุด Modicon™ M221 จัดเป็นคอนโทรลเลอร์ขนาดเล็กที่ทำงานได้รวดเร็วที่สุดในบรรดาผลิตภัณฑ์ชนิดเดียวกัน ซึ่งสามารถทำงานได้เร็วถึง 200 นาโนวินาทีต่อหนึ่งชุดคำสั่ง ทำให้เหมาะอย่างยิ่งกับการใช้งานร่วมกับแอปพลิเคชันต่างๆ ที่ไม่สามารถจะหาได้ในผลิตภัณฑ์ประเภทนี้

Modicon M221 มาพร้อมซีพียูประสิทธิภาพสูงตอบสนองทุกความต้องการ มีพอร์ตเชื่อมต่อเอสดีการ์ด ยูเอสบี แอนาล็อก และช่องเชื่อมต่อสัญญาณอีก 2 ช่อง รองรับต่อขยายได้ตามความต้องการ และการเติบโตของธุรกิจ ติดตั้งได้อย่างรวดเร็วและง่ายดาย สามารถควบคุมเซอร์โวมอเตอร์ได้พร้อมกัน 2 ตัว ด้วยพอร์ต PTO 2 แชนแนล ที่ความละเอียดสูงถึง 100 กิโลเฮิร์ตซ์ รวมถึงฟังก์ชันเอสเคิร์ฟเพื่อควบคุมการทำงานของมอเตอร์ได้อย่างราบรื่น

Modicon M221 มาพร้อมซอฟต์แวร์ให้คุณใช้งานได้ครบถ้วน So Machine Basic ซึ่งใช้งานง่าย ไม่ต้องเรียนรู้เพิ่มเติม ผู้ใช้งานสามารถดาวน์โหลดและอัปเดตฟรี ตลอดอายุการใช้งาน

ผู้ใช้งานสามารถเชื่อมต่อและเรียกดูข้อมูลเครื่องจักรผ่านเครือข่ายอีเธอร์เน็ต เพิ่มประสิทธิภาพการทำงาน โดยใช้งานผ่านโมบายแอป บนมือถือหรือแท็บเล็ต จากทุกที่ ทุกเวลา รวมถึงเรียกดูข้อมูลเทคนิคอย่างง่ายดายผ่านคิวอาร์โค้ด

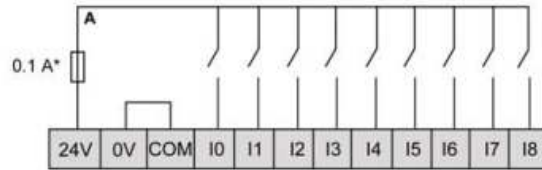
Note:

- ตัวแปร %I คือ ตัวแปรที่รับค่ามาจาก port input จาก ตัว PLC
- ตัวแปร %Q คือ ตัวแปรที่ส่งค่าไปออกที่ port output ที่ตัว PLC
- ตัวแปร %M คือ ตัวแปร memory ภายใน เพื่อส่งค่าไปให้ ตัวแปรภายใน

Input Wiring

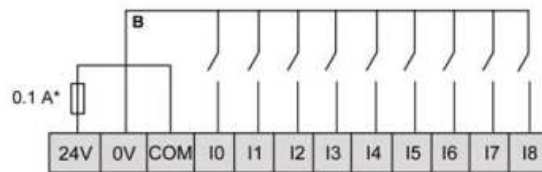
TM221C16R / TM221CE16R Wiring Diagrams

The following figure shows the sink wiring diagram (positive logic) of the inputs to the sensors for TM221C16R and TM221CE16R:



* Type T fuse

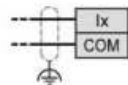
The following figure shows the source wiring diagram (negative logic) of the inputs to the sensors for TM221C16R and TM221CE16R:



* Type T fuse

NOTE: The TM221C Logic Controller provides a 24 Vdc supply to the inputs.

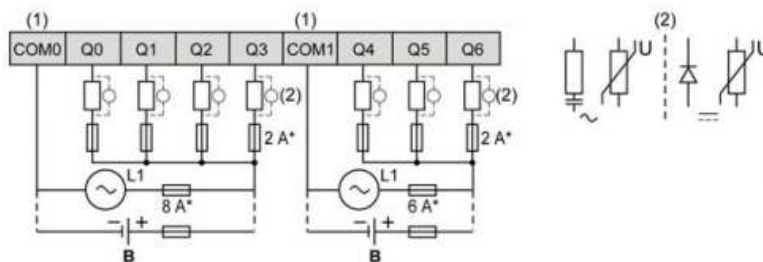
The following figure shows the connection of the fast inputs:



Output Wiring

Relay Outputs Wiring Diagrams - Positive Logic (Sink)

The following figure shows the sink wiring diagram (positive logic) of the outputs of the to the load for the TM221C16R / TM221CE16R:



* Type T fuse

(1) The COM1 and COM2 terminals are **not** connected internally.

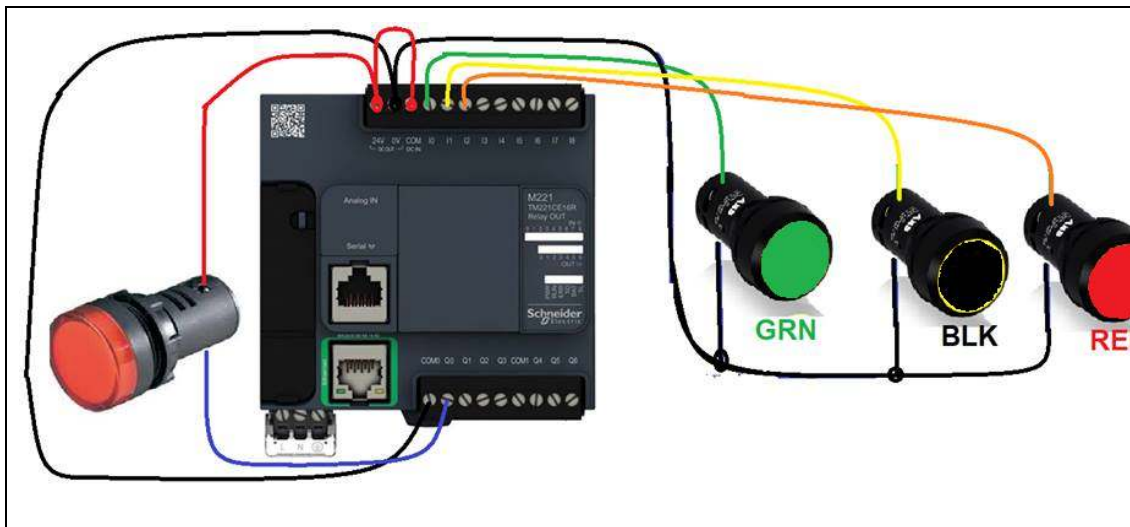
(2) To improve the life time of the contacts, and to protect from potential inductive load damage, you must connect a free wheeling diode in parallel to each inductive DC load or an RC snubber in parallel of each inductive AC load

3.2 ทดสอบการเขียนโปรแกรมบน PLC M221CE16R

- M221CE16R



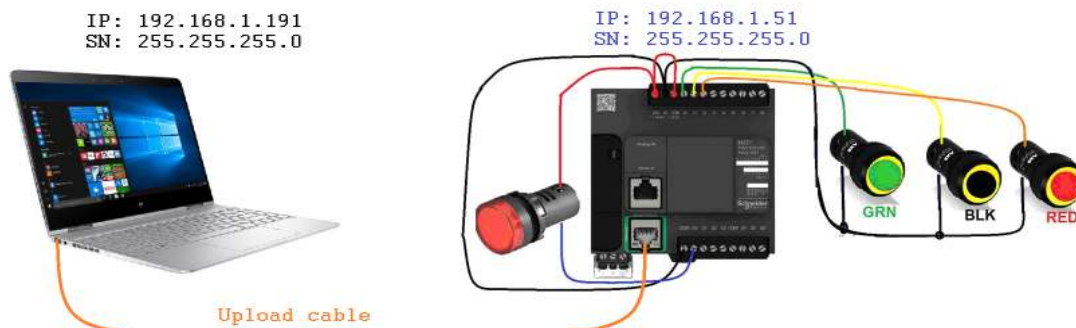
- Lab Wiring: 1 24V Lamp + 2 NO Switch + 1 NC Switch



PLC Test 1/3. Basic Input / Output

1. Install SoMachine and Vejio Citec → Read “Exp1.PLC - Install.pdf”
2. Setup System → Read “Exp2.PLC - Setup_System.pdf”
3. Start New Project → Read “Exp3.PLC - PLC_Start.pdf”
4. Switch to RUN Mode
5. ทดสอบความเข้าใจด้วยคำถามชุด A

System



Model >> TM221CE16R

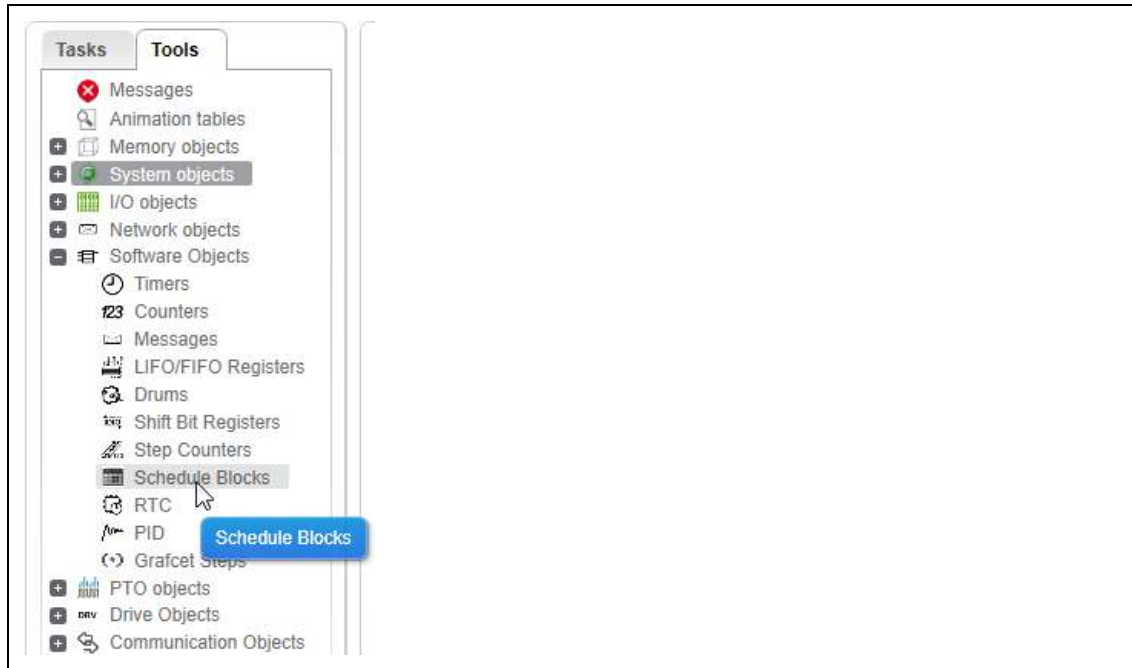
Configuration	Programming	Display																																				
<table border="1"> <thead> <tr> <th colspan="4">M221 Logic Controllers</th> </tr> <tr> <th>Reference</th> <th>Power supply</th> <th>Comm. Ports</th> <th>Digital in</th> </tr> </thead> <tbody> <tr> <td>TM221C40T</td> <td>24 Vdc</td> <td>1 SL</td> <td>24</td> </tr> <tr> <td>TM221C40U</td> <td>24 Vdc</td> <td>1 SL</td> <td>24</td> </tr> <tr> <td>TM221CE16R</td> <td>100...240 Vac</td> <td>1 SL + 1 ETH</td> <td>9</td> </tr> <tr> <td>TM221CE16T</td> <td>24 Vdc</td> <td>1 SL + 1 ETH</td> <td>9</td> </tr> <tr> <td>TM221CE16U</td> <td>24 Vdc</td> <td>1 SL + 1 ETH</td> <td>9</td> </tr> <tr> <td>TM221CE24R</td> <td>100...240 Vac</td> <td>1 SL + 1 ETH</td> <td>14</td> </tr> <tr> <td>TM221CE24T</td> <td>24 Vdc</td> <td>1 SL + 1 ETH</td> <td>14</td> </tr> </tbody> </table>			M221 Logic Controllers				Reference	Power supply	Comm. Ports	Digital in	TM221C40T	24 Vdc	1 SL	24	TM221C40U	24 Vdc	1 SL	24	TM221CE16R	100...240 Vac	1 SL + 1 ETH	9	TM221CE16T	24 Vdc	1 SL + 1 ETH	9	TM221CE16U	24 Vdc	1 SL + 1 ETH	9	TM221CE24R	100...240 Vac	1 SL + 1 ETH	14	TM221CE24T	24 Vdc	1 SL + 1 ETH	14
M221 Logic Controllers																																						
Reference	Power supply	Comm. Ports	Digital in																																			
TM221C40T	24 Vdc	1 SL	24																																			
TM221C40U	24 Vdc	1 SL	24																																			
TM221CE16R	100...240 Vac	1 SL + 1 ETH	9																																			
TM221CE16T	24 Vdc	1 SL + 1 ETH	9																																			
TM221CE16U	24 Vdc	1 SL + 1 ETH	9																																			
TM221CE24R	100...240 Vac	1 SL + 1 ETH	14																																			
TM221CE24T	24 Vdc	1 SL + 1 ETH	14																																			

Ladder Diagram

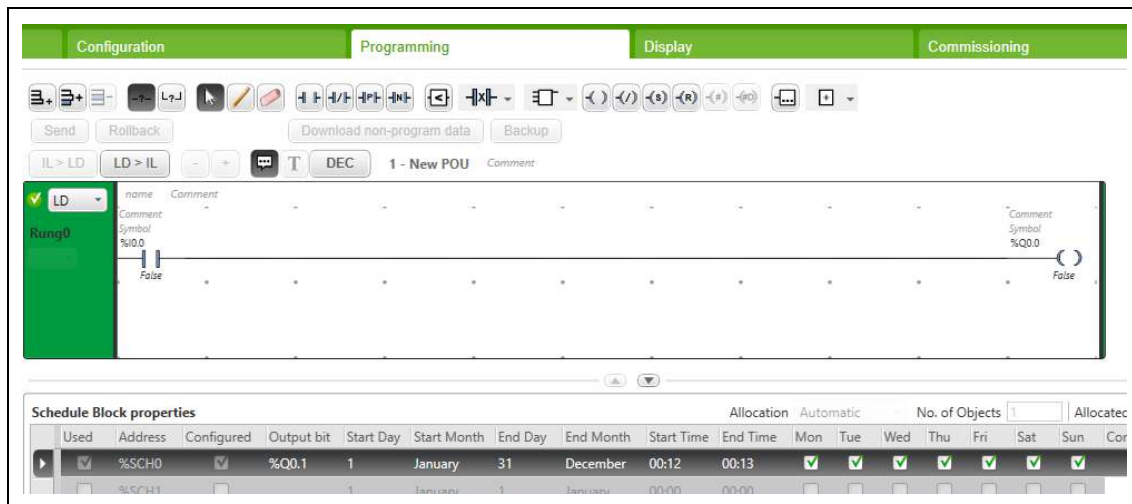
Configuration	Programming	Display	Commissioning

PLC Test 1/3. RTC Control

6. Using RTC on M221 PLC, Stop System and Logout
7. Create New project, Select Model >> TM221CE16R
8. Programming → Tools → Software Object → Schedule Block



9. Configure Schedule ➡ Scho Control Q0.1 Start Date>1Jan to 31Dec, Time>00.00-00.15, Every Day



PLC Test 1/3. Counter

10. Using Timer on M221 PLC, Stop System and Logout
11. Create New project, Select Model >> TM221CE16R
12. Programming → Tools → Software Object → Timer
13. Configure Timer → %TMx , Time Base = 1S, Present=1
14. ทดสอบโปรแกรมตาม ladder

The screenshot shows the SIMATIC Manager interface. The top part displays a ladder logic program with a single rung (Rung0) containing a normally open contact labeled 'IN' (Symbol: %I0.0) connected to a coil labeled 'Q' (Symbol: %Q0.0). The coil is configured as a timer (TON) with a time base of 1 s and a preset value of 2. Below the ladder logic, the 'Timer properties' dialog box is open, showing a table of timer configurations.

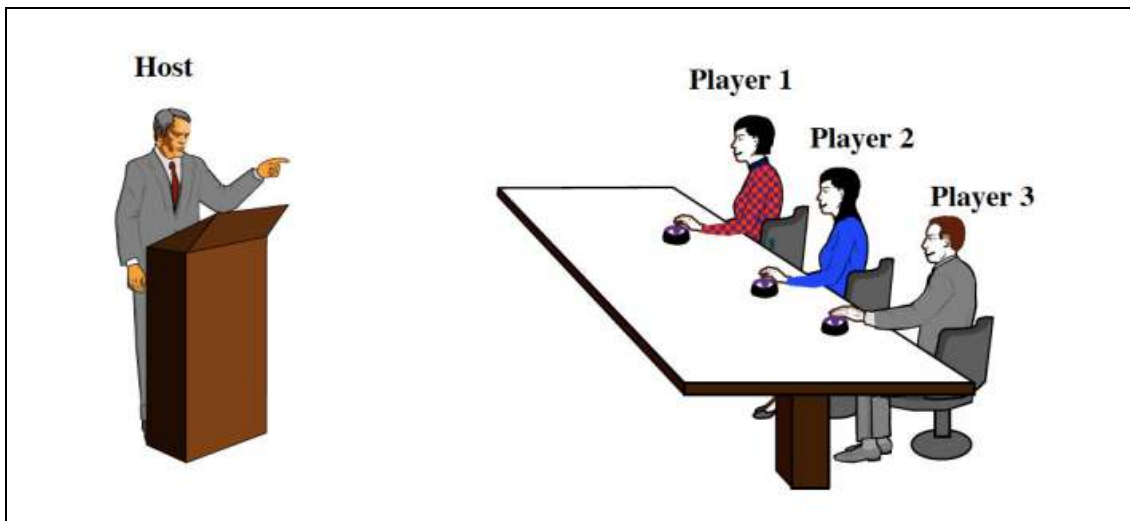
Used	Address	Symbol	Type	Retentive	Time Base	Preset	Comment
<input checked="" type="checkbox"/>	%TM0		TON	<input type="checkbox"/>	1 s	2	
<input type="checkbox"/>	%TM1		TON	<input type="checkbox"/>	1 min	9999	
<input type="checkbox"/>	%TM2		TON	<input type="checkbox"/>	1 min	9999	
<input type="checkbox"/>	%TM3		TON	<input type="checkbox"/>	1 min	9999	
<input type="checkbox"/>	%TM4		TON	<input type="checkbox"/>	1 min	9999	

At the bottom right of the dialog box are 'Apply' and 'Cancel' buttons.

15. ทดสอบความเข้าใจด้วยคำถามชุด B

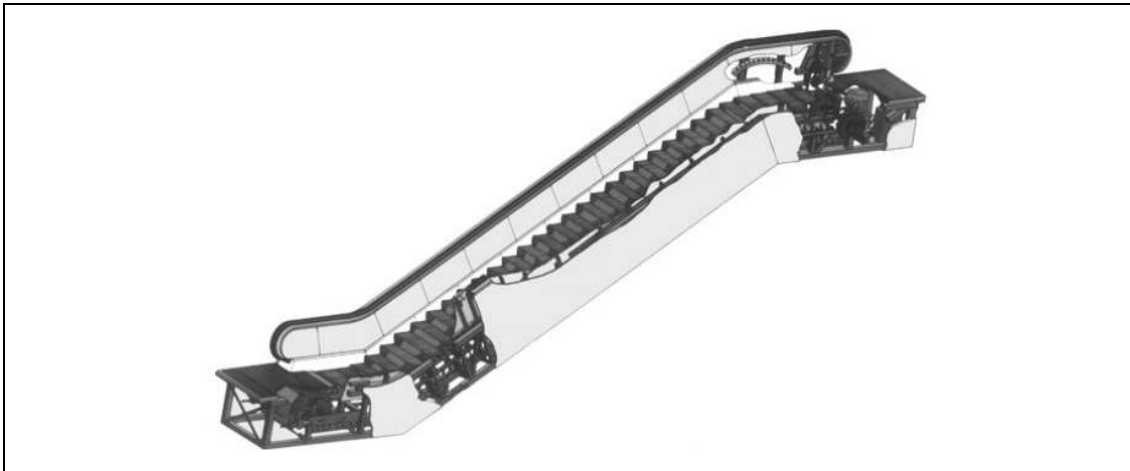
3.3 Question A Set

1. จงเขียน Ladder Diagram เพื่อควบคุมการทำงานของอุปกรณ์ที่ควบคุมการทำงานด้วยการเชื่อมต่อสวิตช์ input แบบ AND 2 อันและมี output 1 อัน เขียน Ladder Diagram บน computer แล้ว upload ไปสู่เครื่อง PLC
2. จงเขียน Ladder Diagram เพื่อควบคุมการทำงานของอุปกรณ์ที่ควบคุมการทำงานด้วยการเชื่อมต่อสวิตช์ input แบบ OR 2 อันและมี output 1 อัน เขียน Ladder Diagram บน computer แล้ว upload ไปสู่เครื่อง PLC
3. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟแบบกดติดกดดับพร้อมทั้งทดสอบการทำงานที่ควบคุมด้วยPLC
4. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟบันไดคือผู้ใช้ไม่ว่าจะสลับสวิชไฟไปในทิศทางใดก็สามารถที่จะเปิดหรือปิดไฟตามที่เรต้องการพร้อมทั้งทดสอบการทำงานที่ควบคุมด้วยPLC
5. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟถ้ามี push button เป็น input อยู่ 2 เมื่อกดปุ่มแรกหลอดไฟที่ output จะติดและติดต่อไปแม้ว่าเราจะปล่อยสวิตช์ไปแล้วก็ตามไฟจะดับก็ต่อเมื่อเรากด push button อันที่สอง
6. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟของผู้เล่นเกมสโว์โดยมีพิธีกร 1 คนและผู้เข้าแข่งขันอีก 3 คนเมื่อกรรมการเปิดไฟของตน ผู้เข้าแข่งขันจึงจะสามารถแย่งการกดไฟได้ การกดก่อนไฟกรรมการติดจะไม่มีผล และเมื่อผู้เข้าแข่งขันผู้ใดกดไฟได้ก่อนไฟตนเองจะติดและผู้เข้าแข่งขันที่เหลืออีกสองคนจะไม่สามารถกดไฟให้ติดได้จนกระทั่งกรรมการปิดไฟตนเองวงจรจึงจะกลับเข้าสู่สภาวะเริ่มต้นอีกครั้งหนึ่ง



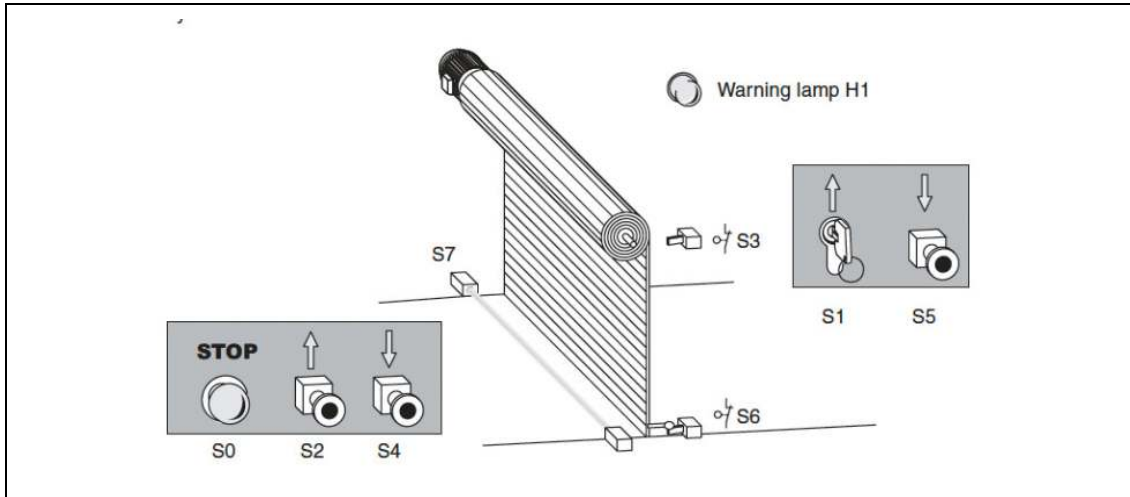
3.3 Question B Set

7. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟเริ่มจากไฟดับอยู่และเมื่อเราเปิดmaster switch ไฟดวงหนึ่งกระพริบติดดับสลับกันทุก 1 วินาทีไปเรื่อยๆ และหลอดไฟนี้จะหยุดกระพริบเมื่อเราสั่ง off master switch
8. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟ 3 ดวงเริ่มจากทุกดวงดับหมดเมื่อเปิดลิทซ์หลักจะเริ่มต้นการทำงานคือไฟดวงที่หนึ่งติดหนึ่งวินาทีขณะที่ดวงอื่นดับจากนั้นเปลี่ยนเป็นดวงที่สองติดหนึ่งวินาทีขณะที่ดวงอื่นดับแล้วเปลี่ยนเป็นดวงที่สามติดหนึ่งวินาทีขณะที่ดวงอื่นดับต่อด้วยดวงที่สองติดหนึ่งวินาทีขณะที่ดวงอื่นดับจากนั้นจะย้อนกลับมาเป็นดวงที่หนึ่งติดดวงเดียวหนึ่งวินาทีแล้ววนไปเรื่อยๆ จนกว่าจะสลับลิทซ์หลัก
9. ออกแบบโปรแกรมควบคุมการทำงานแบบประหยัพลังงานของบันไดเลื่อนที่เชิงบันไดจะมีเซนเซอร์ตรวจจับว่ามีคนเดินผ่านหรือไม่ถ้าไม่มีคนเดินผ่านเป็นระยะเวลา30วินาทีPLCจะสั่งให้มอเตอร์บันไดเลื่อนหยุดหมุนและจะหมุนก็ต่อเมื่อมีคนมาที่เชิงบันไดอีก

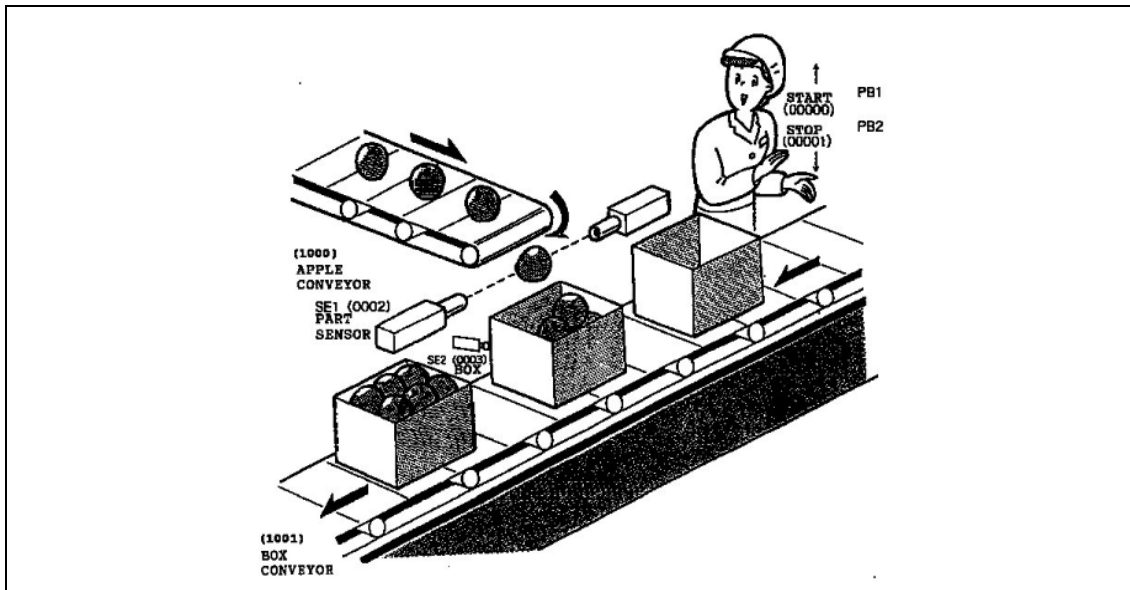


10. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดไฟ 3 หลอดโดยเริ่มต้นทั้งสามหลอดจะดับทั้งหมดถ้า กด push button A หนึ่งครั้งหลอดไฟจะติด 1 หลอดเมื่อกดครั้งที่สองหลอดไฟจะติดเพิ่มขึ้นอีกหนึ่งหลอดเมื่อกดครั้งที่สามหลอดไฟจะติดเพิ่มขึ้นอีกหนึ่งหลอดแต่เมื่อกดครั้งที่สี่หลอดไฟทุกดวงจะดับหมด และหากเราเริ่มกดใหม่วัฏจักรก็จะเริ่มวนต่อไป

11. จงเขียน Ladder Diagram เพื่อควบคุมการเปิดปิดประตูโรงรถโดยทางเข้าจะมีกุญแจ S1เพื่อให้ประตูเปิด ส่วน push button S5 ด้านนอกจะสั่งให้ประตูปิดลงสำหรับด้านในจะมีpush button 2 อันสำหรับ S2 จะให้ประตูเลื่อนขึ้นและ S4 จะทำให้ประตูเลื่อนลงนอกจากนั้นที่ประตูจะมี Limit Switch S3 ตรวจสอบว่าประตูขึ้นสุดและ S4 เพื่อตรวจสอบว่าประตูเลื่อนปิดสุดนอกเหนือจากนั้นเพื่อความปลอดภัยป้องกันไม่ให้ประตูปิดระหว่างที่มีรถจอดขวางอยู่จะมี switch S7 เป็น safety switch อยู่และระหว่างประตูเลื่อนเปิดหรือปิดไฟ H1 จะติดและลองทดสอบการทำงานบน PLC ด้วยไฟแสดงผลด้วย



12. เมื่อกดปุ่มPB1 (Start Push Button) กล้องที่อยู่จะเคลื่อนที่โดย conveyor ของกล่องและเมื่อกล่องเข้าประจำที่เรียบร้อยแล้ว conveyor ของกล่องจะหยุดและ conveyor ของ apple จะเริ่มทำงานเมื่อ apple ตกลงบนกล่องครบ 10 ลูก conveyor ของ apple จะหยุดทำงานและconveyorของกล่องจะเริ่มเคลื่อนที่อีกครั้งเป็นcycleต่อไปและจะหยุดเมื่อกด PB2 (STOP Push Button) จงเขียนโปรแกรม PLC เพื่อควบคุมระบบดังกล่าว



การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร M2M - Intelligence Machine Control
ชื่อ-สกุล :

4/4: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_201 – Read Modbus RTU

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

Quiz_202 – Write Modbus RTU

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

Quiz_203 – Read/Write Modbus RTU

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

Quiz_204 – PLC Test

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