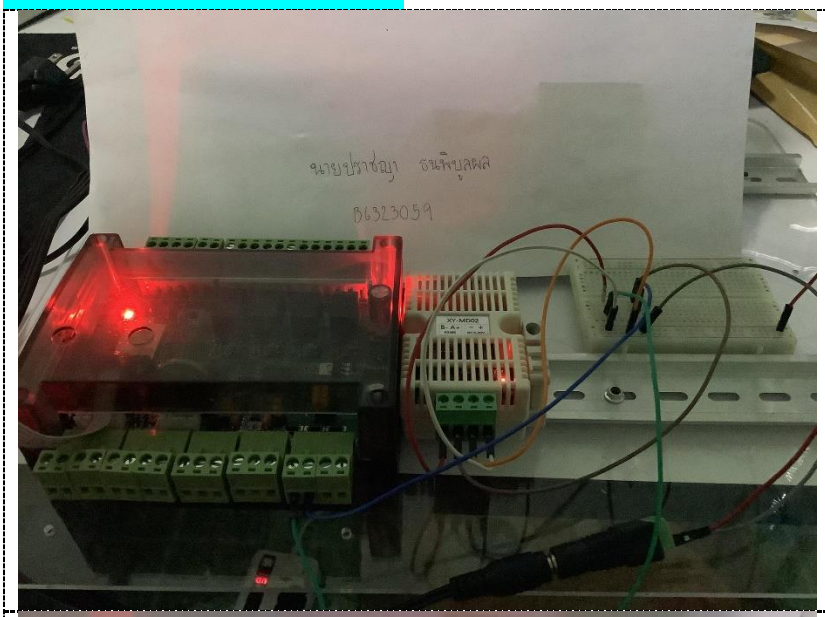
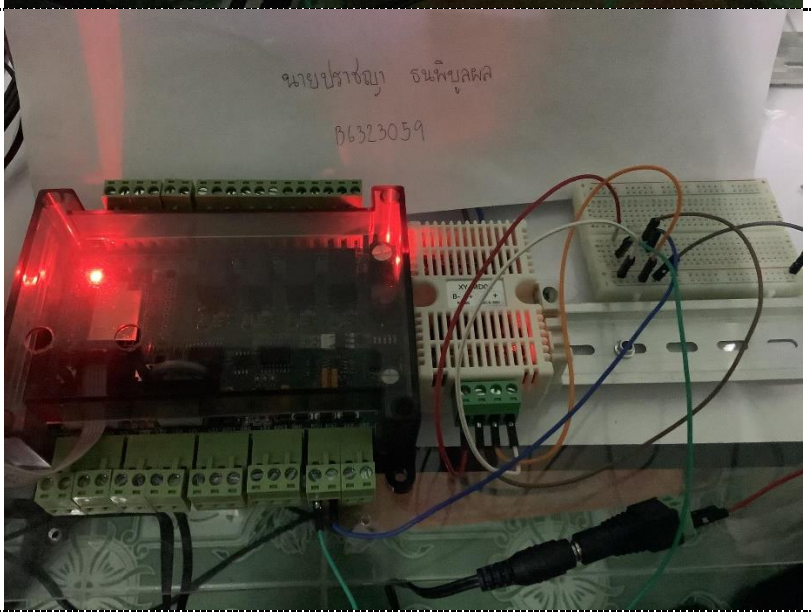
การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร **M2M - Intelligence Machine Control**	
ชื่อ-สกุล : นายปราชญา ธนพิบูลผล	**รหัสนักศึกษา : B6323059**

4/4: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_201 – Read Modbus RTU






```
#include "ModbusMaster.h" //hIps://github.com/4-20ma/ModbusMaster
#define SlaveID_White 11
#define SlaveID_Black 12
```

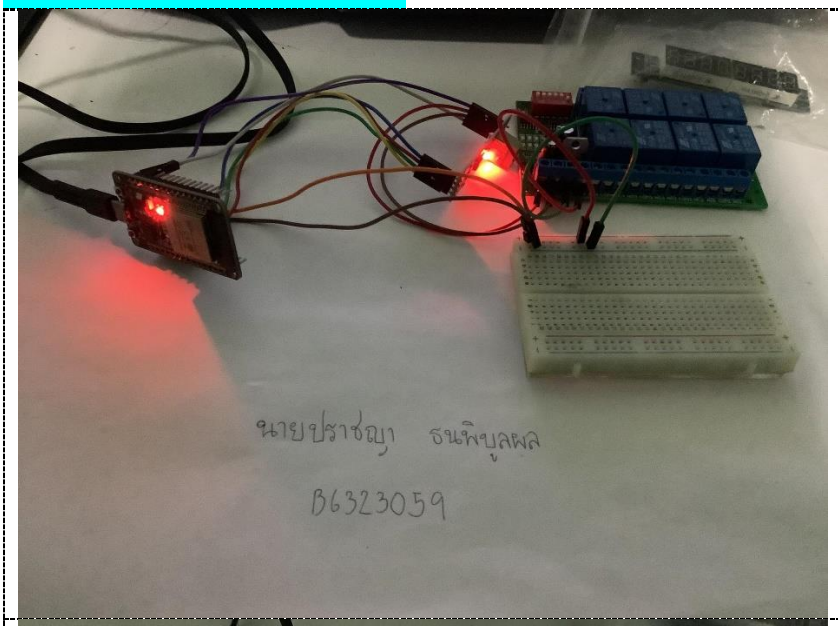
```
#define RX_PIN 26
#define TX_PIN 27
ModbusMaster modbus1, modbus2;
void setup() {
  Serial.begin(115200, SERIAL_8N1);
  Serial2.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  modbus1.begin(SlaveID_White, Serial2);
  modbus2.begin(SlaveID_Black, Serial2);
}
double GetData_DBL;
uint8_t result;
void loop() {
  Serial.println();
  delay(2000);
  Serial.print(" White = ");
  result = modbus1.readInputRegisters(1, 2);
  if (getResultMsg(&modbus1, result)) {
    GetData_DBL = modbus1.getResponseBuffer(0) / 10.0;
    Serial.print(GetData_DBL);
    Serial.print(",");
    GetData_DBL = modbus1.getResponseBuffer(1) / 10.0;
    Serial.print(GetData_DBL);
  }
  delay(2000);
  Serial.print(" Black = ");
  result = modbus2.readInputRegisters(1, 2);
  if (getResultMsg(&modbus2, result)) {
    GetData_DBL = modbus2.getResponseBuffer(0) / 10.0;
    Serial.print(GetData_DBL);
    Serial.print(",");
    GetData_DBL = modbus2.getResponseBuffer(1) / 10.0;
    Serial.print(GetData_DBL);
  }
}
bool getResultMsg(ModbusMaster *node, uint8_t result) {
  String tmpstr2 = "\r\n";
```

```
  switch (result) {
    case node->ku8MBSuccess:
      return true;
      break;
    case node->ku8MBIllegalFunction:
      tmpstr2 += "Illegal FuncSon";
      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 FuncSon";
      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;
}
```

```
White = 27.00,46.50 Black = 27.70,43.70
White = 27.00,46.50 Black = 27.70,43.70
White = 27.00,46.50 Black = 27.60,43.80
White = 27.00,46.60 Black = 27.60,43.80
White = 27.00,46.60 Black = 27.60,43.80
White = 27.00,46.50 Black = 27.60,43.80
White = 27.00,46.40 Black = 27.60,43.90
White = 27.00,46.40 Black = 27.60,43.90
White = 27.00,46.40 Black = 27.60,44.00
```

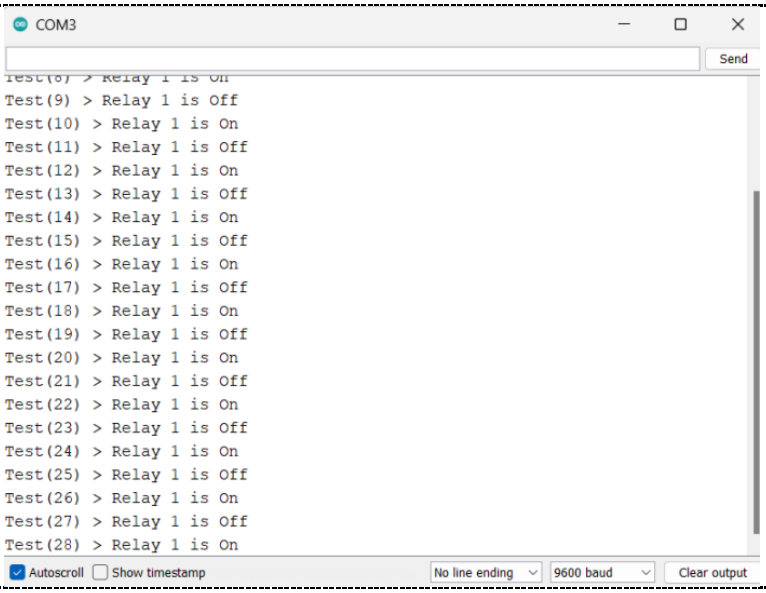
Autoscroll | Show timestamp | No line ending | 115200 baud | Clear output

Quiz_202 – Write Modbus RTU

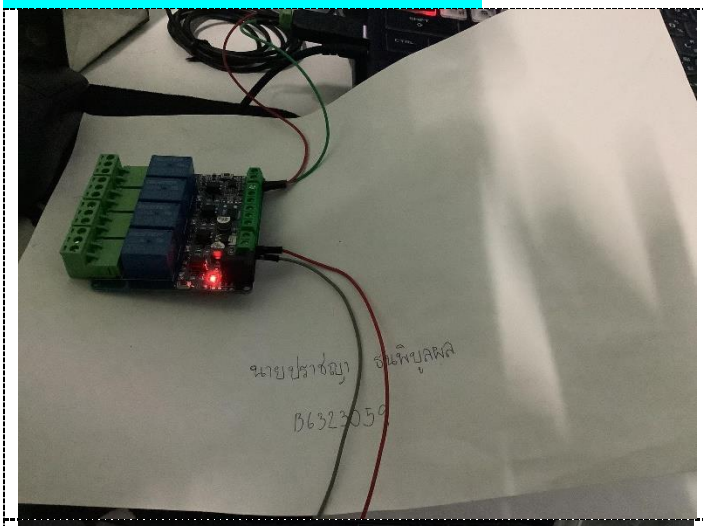


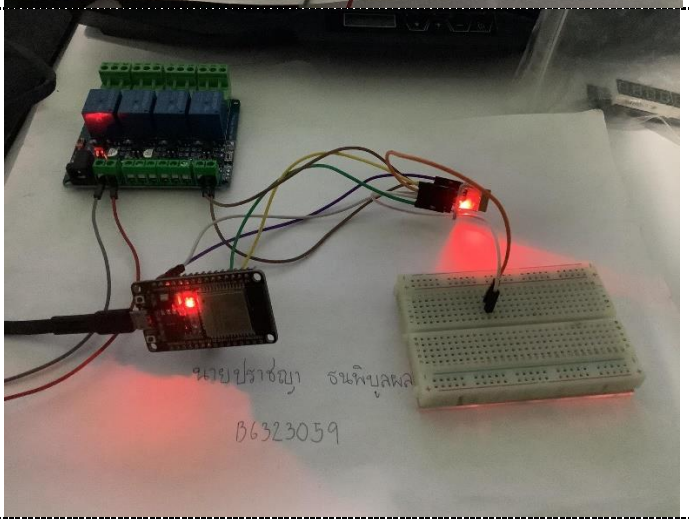
```
#define RS485TX HIGH
#define RS485RX LOW
#define RS485CTRL 5
#define LED_MONITOR 2
int stepCount = 0;
int eindex = 0;
byte byteSend;
byte echo[20];
byte cmd_on[] = {0x03, 0x06, 0x00, 0x01, 0x01, 0x00, 0xD8, 0x78};
byte cmd_off[] = {0x03, 0x06, 0x00, 0x01, 0x02, 0x00, 0xD8, 0x88};
byte cmd_read[] = {0x03, 0x03, 0x00, 0x01, 0x00, 0x01, 0xD4, 0x28};
String mapData(byte number) {
  switch (number) {
    case 1 : return "On";
    case 2 : return "Off";
    default : return "Invalid State";
  }
}
void setup() {
  pinMode(RS485CTRL, OUTPUT);
  pinMode(LED_MONITOR, OUTPUT);
  TN07_002 -- M2M - Intelligence Machine Control ⟶ Page 8 of 30
  Serial.begin(9600);
  Serial2.begin(9600);
  digitalWrite(RS485CTRL, RS485RX);
  Serial.println("Start Test MODBUS RTU");
}
void loop() {
  Serial.print("\nTest(");
  Serial.print(++stepCount);
  Serial.print(") > ");
  digitalWrite(LED_MONITOR, HIGH);
  digitalWrite(RS485CTRL, RS485TX);
  delay(10);
  if ((stepCount % 2) == 0) {
    for (int i = 0; i < sizeof(cmd_on); i++) {
```

```
      Serial2.write(cmd_on[i]);
    }
  }
  else {
    for (int i = 0; i < sizeof(cmd_off); i++) {
      Serial2.write(cmd_off[i]);
    }
  }
  delay(10);
  digitalWrite(RS485CTRL, RS485RX);
  digitalWrite(LED_MONITOR, LOW);
  delay(10);
  digitalWrite(LED_MONITOR, HIGH);
  digitalWrite(RS485CTRL, RS485TX);
  delay(10);
  for (int i = 0; i < sizeof(cmd_read); i++) {
    Serial2.write(cmd_read[i]);
  }
  delay(10);
  digitalWrite(RS485CTRL, RS485RX);
  digitalWrite(LED_MONITOR, LOW);
  TN07_002 -- M2M - Intelligence Machine Control ⟶ Page 9 of 30
  eindex = 0;
  for (long int i = 0; i < 600000; i++) {
    if (Serial2.available())
      echo[eindex++] = Serial2.read();
    if (eindex > 12) i = 999999;
  }
  Serial.print("Relay " + String(echo[1]) + " is " + mapData(echo[2]));
  delay(5000);
}
```



Quiz_203 – Read/Write Modbus RTU

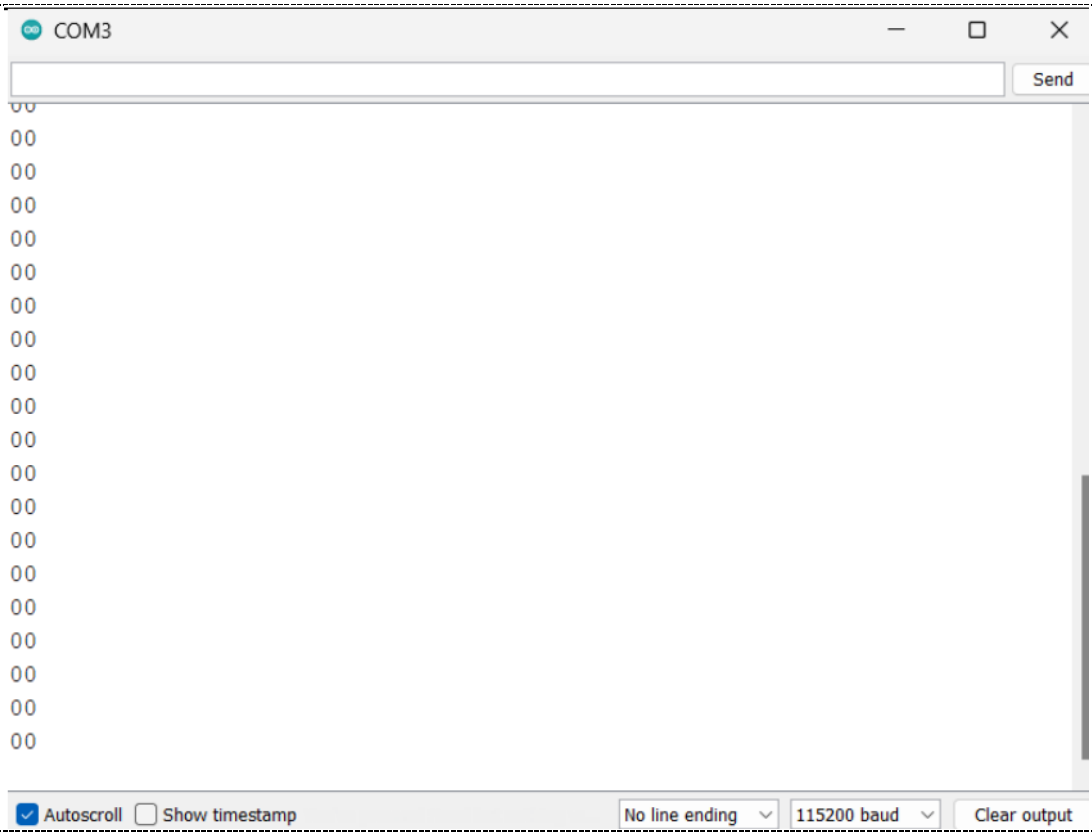




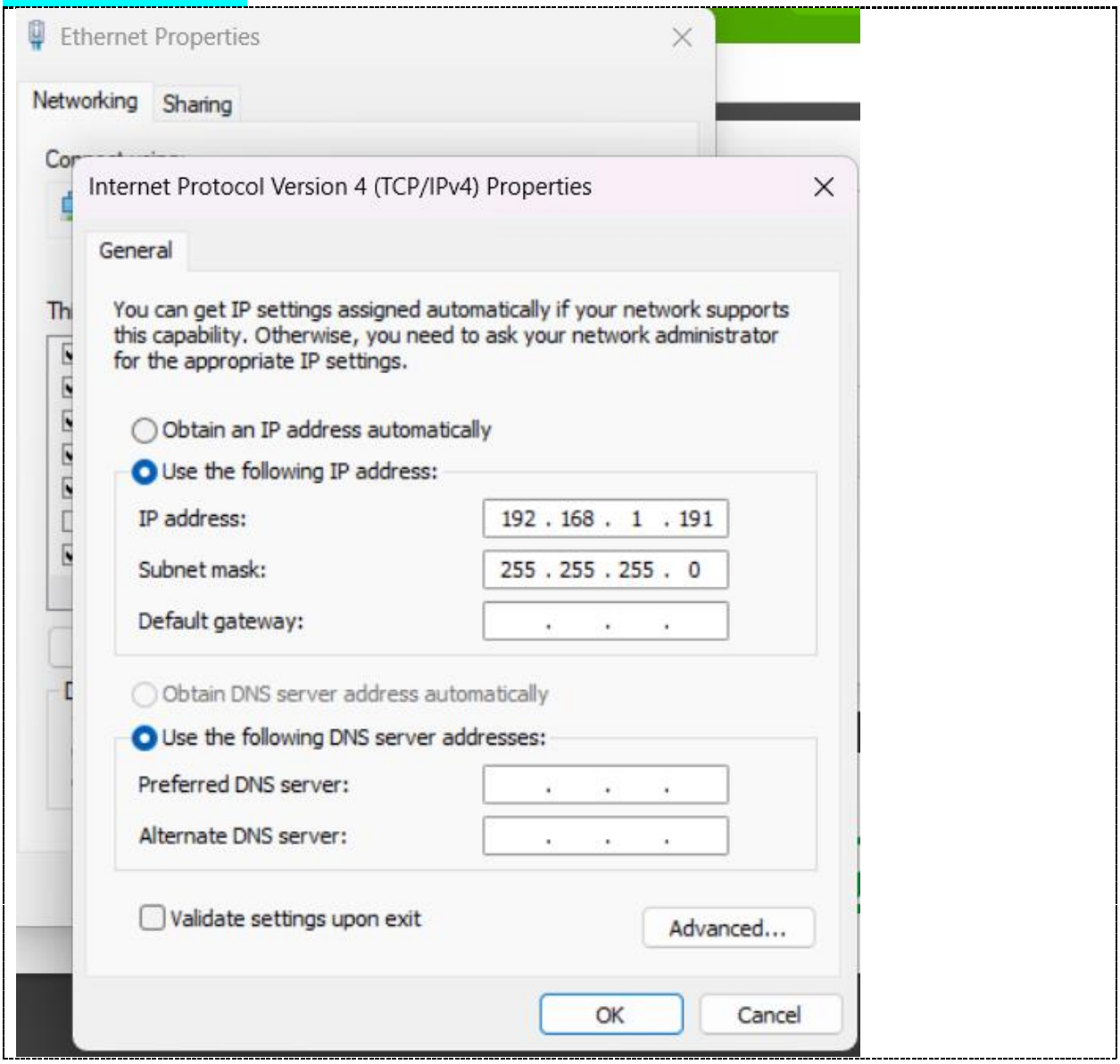
```
#define RS485TX HIGH
#define RS485RX LOW
#define RS485CTRL 5
#define LED_MONITOR 2
int stepCount = 0;
int eindex = 0;
byte echo[20];
byte slaveID = 0x01;
byte modbusCMD = 0x05;
byte h_relayID = 0x00;
byte l_relayID = 0x00;
byte relay_on = 0xFF;
byte relay_off = 0x00;
byte on_off_delay = 0x00;
byte h_byteCRC = 0;
byte l_byteCRC = 0;
void setup() {
  pinMode(RS485CTRL, OUTPUT);
  pinMode(LED_MONITOR, OUTPUT);
  Serial.begin(9600);
  Serial2.begin(9600);
  digitalWrite(RS485CTRL, RS485RX);
  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);
  TN07_002 -- M2M - Intelligence Machine Control ⟶ Page 18 of 30
  if (inData < 0x10) Serial.print("0");
  Serial.print(inData, HEX);
  Serial.print(" ");
  tempCRC = CRC16_Update(tempCRC, inData);
  return tempCRC;
}
void relayCTRL(int relay_id, byte relay_cmd) {
  uint16_t calculateCRC = 0xFFFF;
  h_relayID = highByte(relay_id);
  l_relayID = lowByte(relay_id);
  digitalWrite(LED_MONITOR, HIGH);
  digitalWrite(RS485CTRL, RS485TX);
  delay(10);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, slaveID);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, modbusCMD);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, h_relayID);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, l_relayID);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, relay_cmd);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, on_off_delay);
  h_byteCRC = highByte(calculateCRC);
  l_byteCRC = lowByte(calculateCRC);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, l_byteCRC);
  calculateCRC = sendByte_CRCUpdate(calculateCRC, h_byteCRC);
  delay(10);
  digitalWrite(RS485CTRL, RS485RX);
  digitalWrite(LED_MONITOR, LOW);
  Serial.println();
}
void loop() {
  for (int relay = 0; relay < 4; relay++) {
    relayCTRL(relay, relay_on);
    delay(3000);
  }
```

```
  for (int relay = 0; relay < 4; relay++) {
    relayCTRL(relay, relay_off);
    delay(3000);
  }
}
```

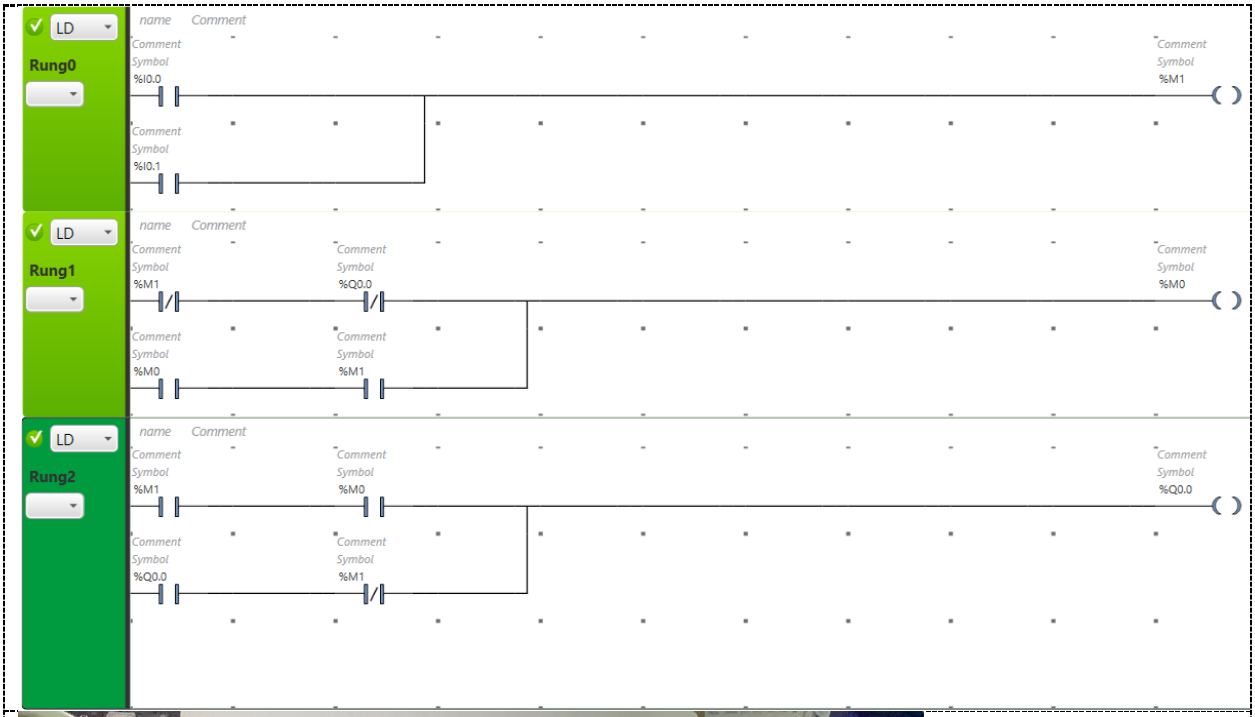


Quiz_204 – PLC Test




SoMachine Basic 1.5 build 58934
Why register?
Register now
Projects
Connect
Templates
Help
About
Exit
Local Devices
Ethernet Devices
192.168.1.190
Keep Modbus driver parameters
Unit ID
1
Remote Lookup
Enter a URL or an IP address
Add
Found: New project
Reference
Firmware
Controller
TM221CE16R
1.5.0.0
Schneider Electric


Reference
Power supply
TM221C24R 100...240 Vac
TM221C24T 24 Vdc
TM221C24U 24 Vdc
TM221C40R 100...240 Vac
TM221C40T 24 Vdc
TM221C40U 24 Vdc
TM221CE16R 100...240 Vac
TM221CE16T 24 Vdc
TM221CE16U 24 Vdc
TM3 Digital I/O Modules
TM3 Analog I/O Modules
TM2 Digital I/O Modules
TM2 Analog I/O Modules
TM3 Expert I/O Modules
M221 Cartridges
Device information
Messages
Device description
TM221CE16R (screw)


LD
Rung0
name
Comment
Comment Symbol
%I0.0
%I0.1
%M1
LD
Rung1
%M1
%Q0.0
%M0
%M0
%M1
LD
Rung2
%M1
%M0
%Q0.0
%Q0.0
%M1