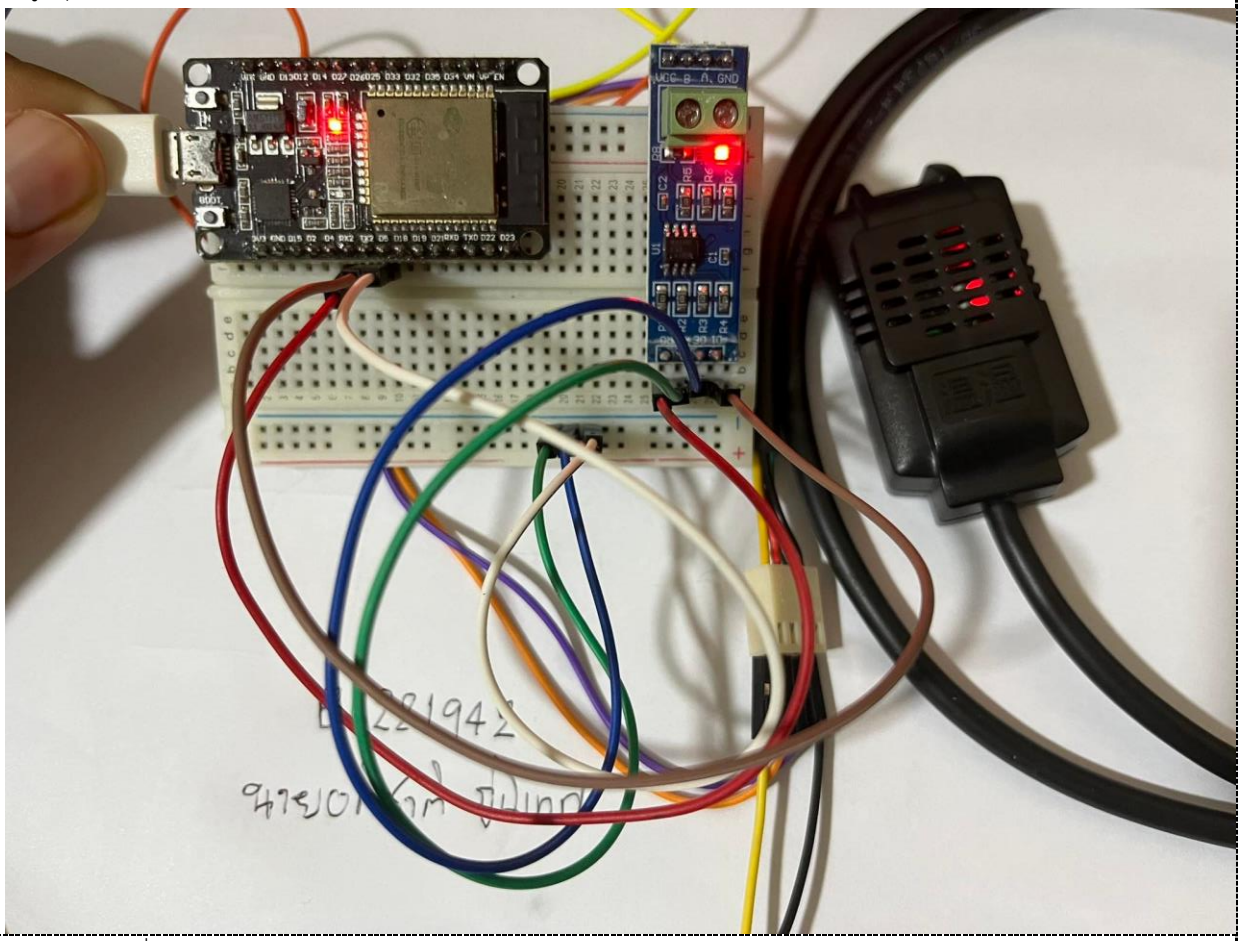
การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร M2M – Intelligence Machine Control
ชื่อ-สกุล : นายอติชาติ ภู่นิเทศ

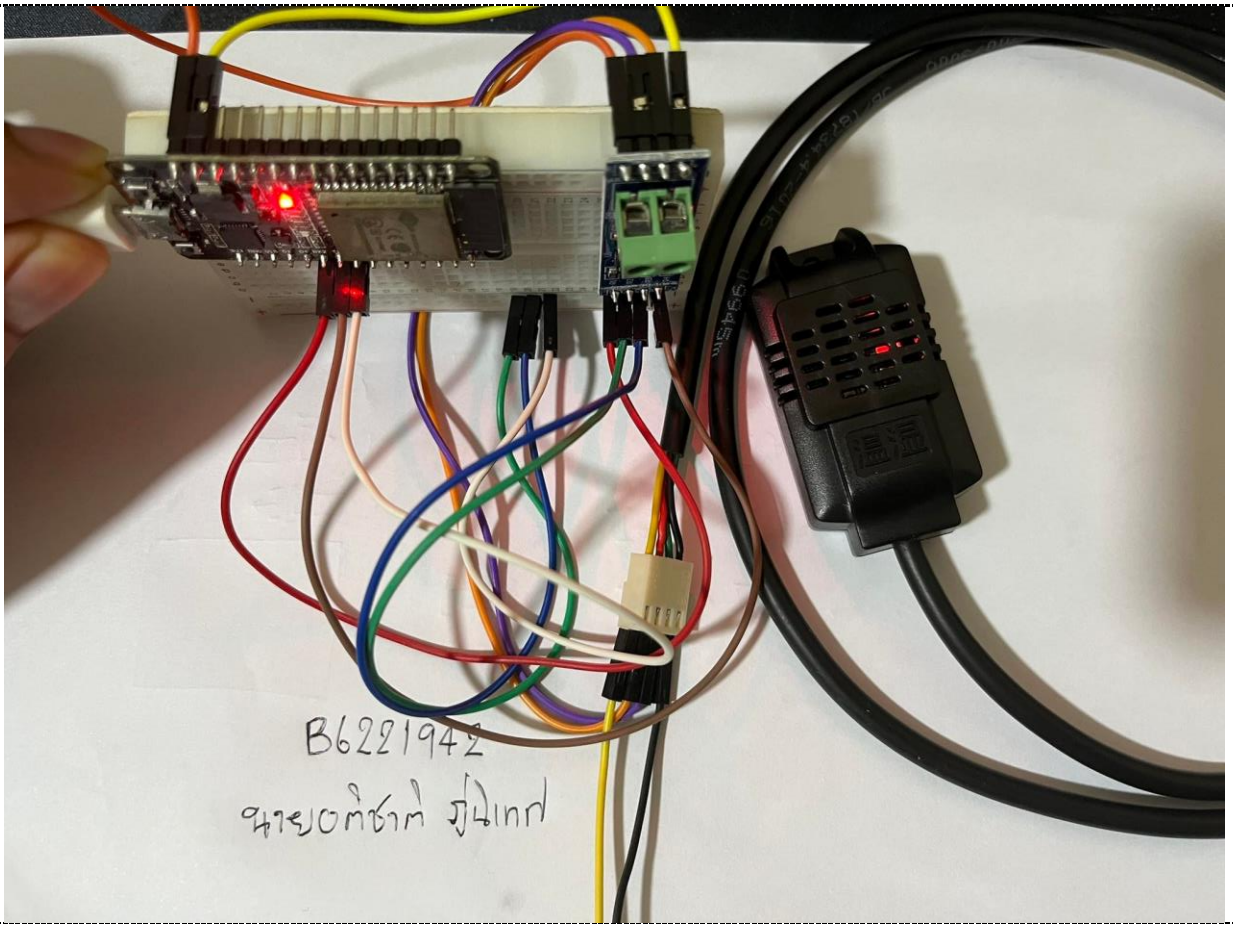
4/4: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_201 – Read Modbus RTU

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >



< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

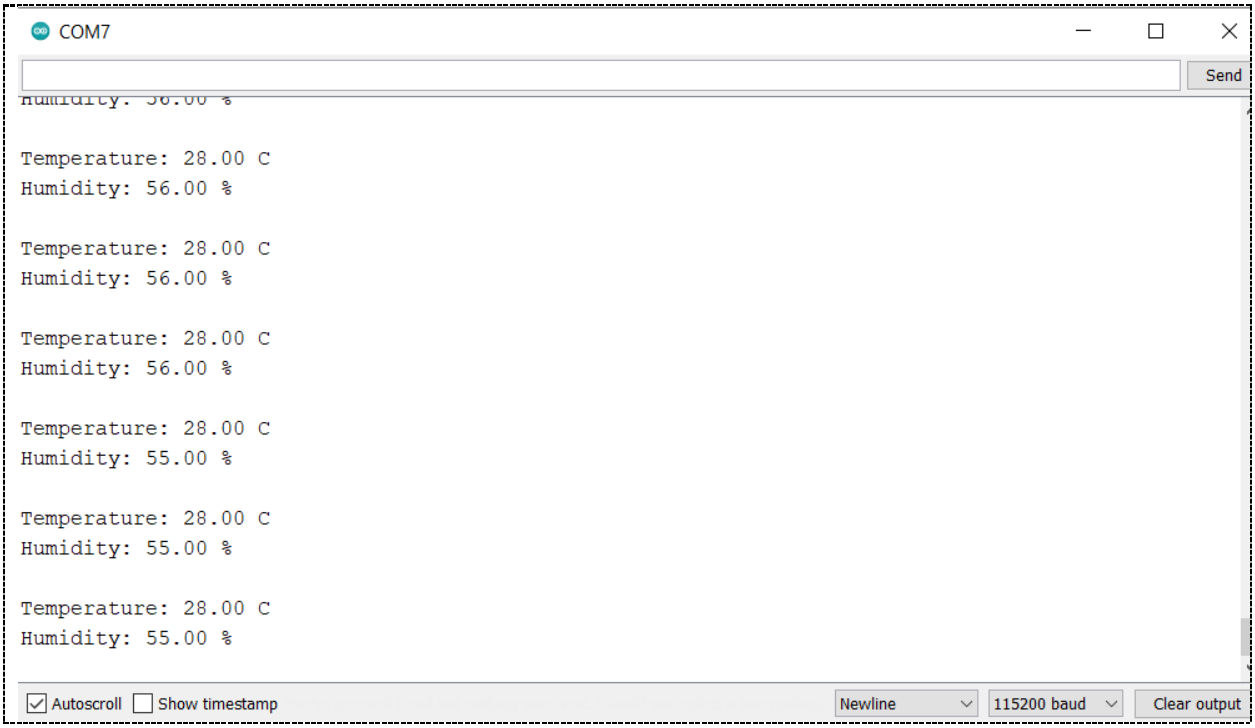


< โปรแกรมทดสอบ >

```
#include "ModbusMaster.h" //https://github.com/4-20ma/ModbusMaster
#define Slave_ID 1
#define MAX485_RE_NEG 5
#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() {
```

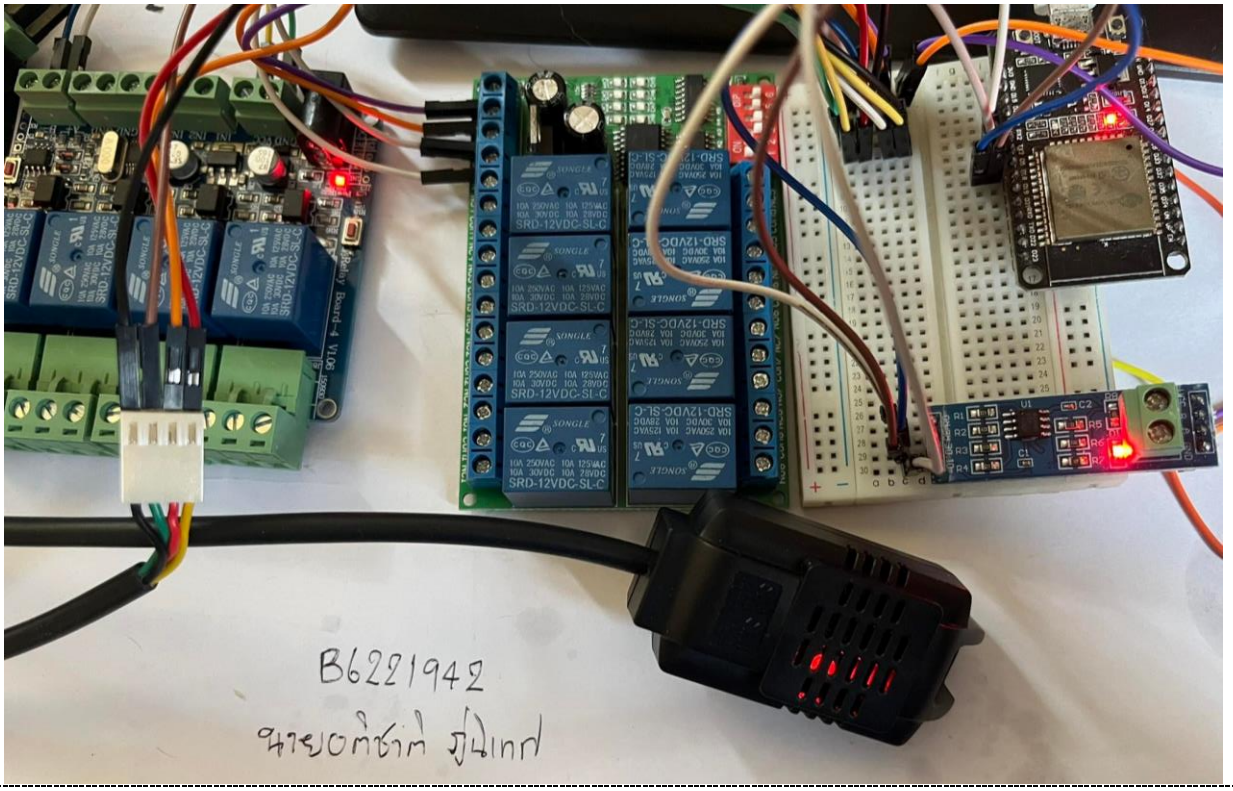
```
  long currentMillis = millis();
  if (currentMillis - lastMillis > 1000) {
    uint8_t result = modbus.readHoldingRegisters(0, 2);
    if (getResultMsg(&modbus, result)) {
      Serial.println();
      double res_dbl = modbus.getResponseBuffer(0) / 10;
      String res = "Temperature: " + String(res_dbl)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

< ผลการทดสอบ >

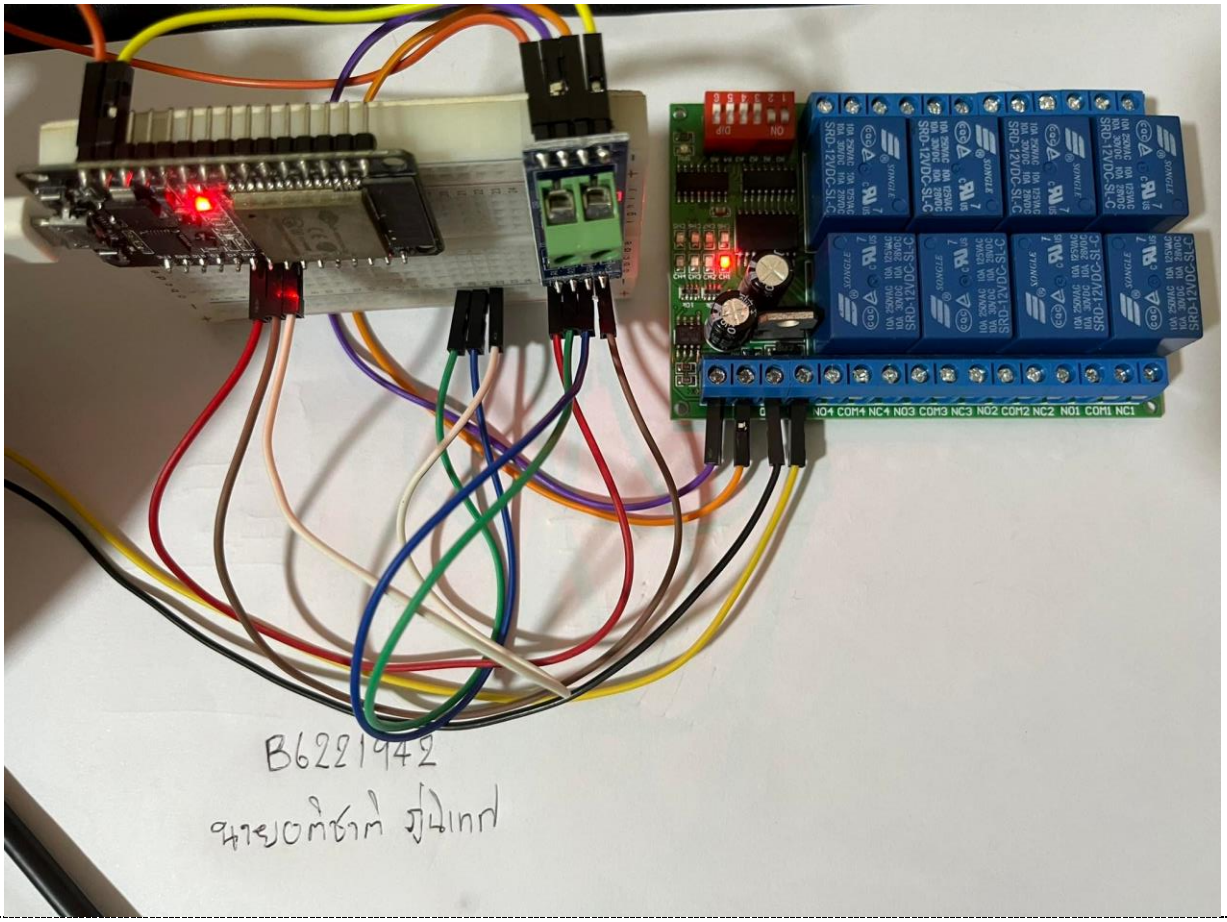


Quiz_202 – Write Modbus RTU

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >



< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

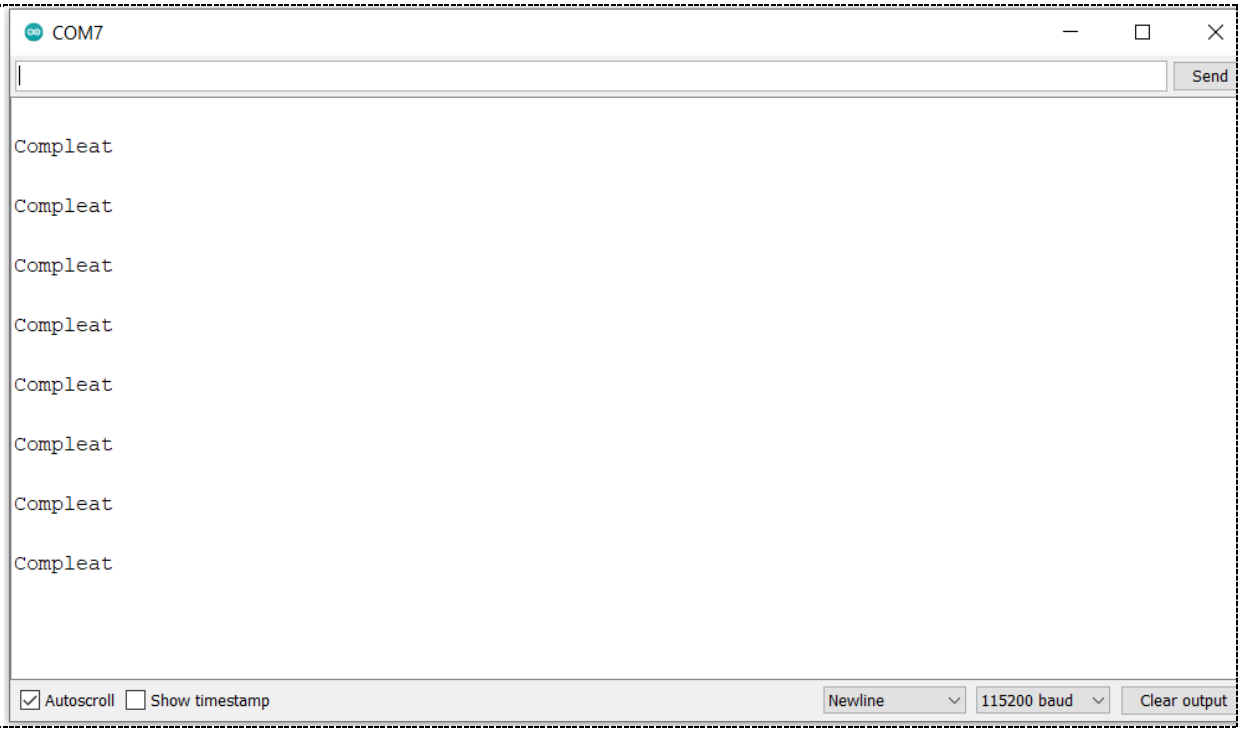


< โปรแกรมทดสอบ >

```
#include "ModbusMaster.h" //https://github.com/4-20ma/ModbusMaster
#define Slave_ID 3
#define MAX485_RE_NEG 5
#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;
```

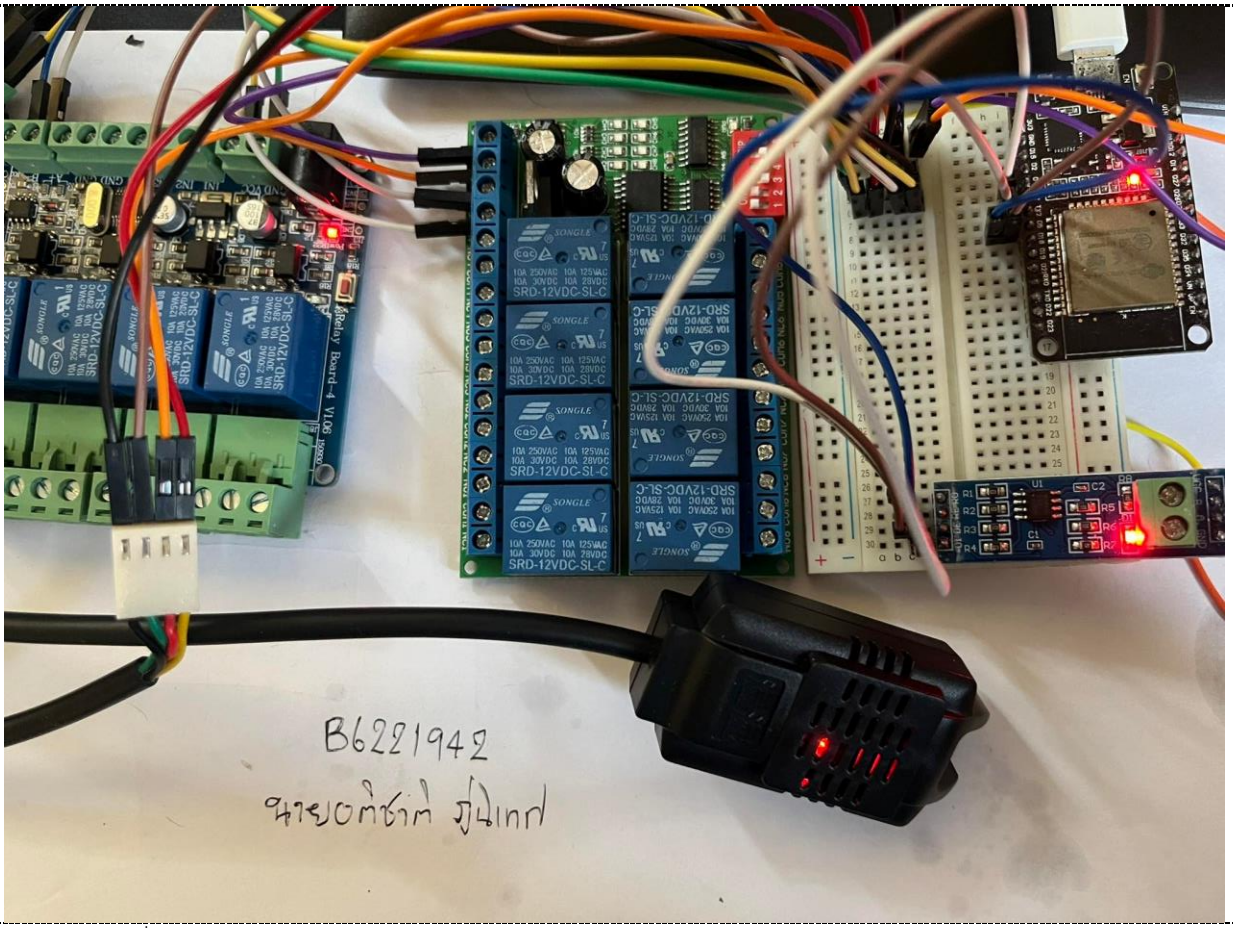
```
void loop() {
  uint8_t result;
  result = modbus.writeSingleRegister(1, 0x0100); // Relay1 On
  getResultMsg(&modbus, result);
  delay(5000);
  result = modbus.writeSingleRegister(1, 0x0200); // Relay1 Off
  getResultMsg(&modbus, result);
  delay(5000);
}
bool getResultMsg(ModbusMaster *node, uint8_t 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;
}
```

< ผลการทดสอบ >



Quiz_203 – Read/Write Modbus RTU

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >



< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

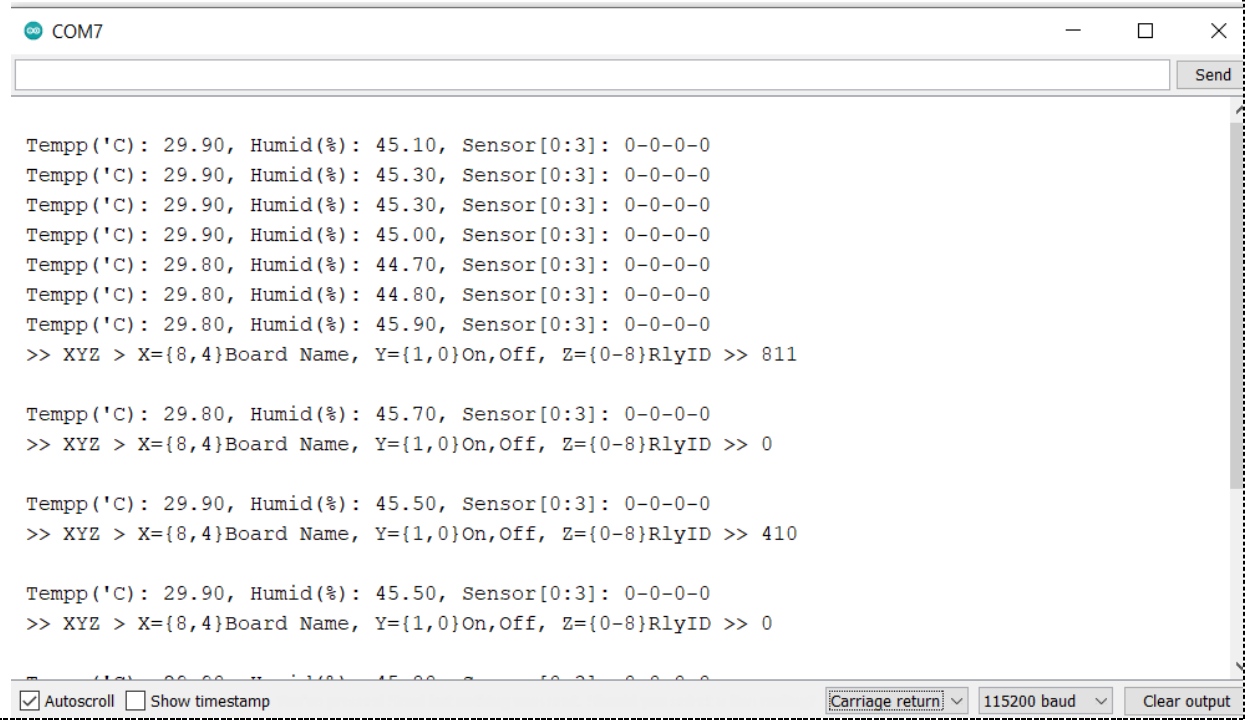
< โปรแกรมทดสอบ >

```
#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 CTempp, 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.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);
}
```

< ผลการทดสอบ >



Quiz_204 – PLC Test

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
< โปรแกรมทดสอบ >
< ผลการทดสอบ >