การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร

M2M - Intelligence Machine Control

ชื่อ-สกุล : ณัฐพงศ์ โต๊ะแอ รหัสนักศึกษา : B6310158

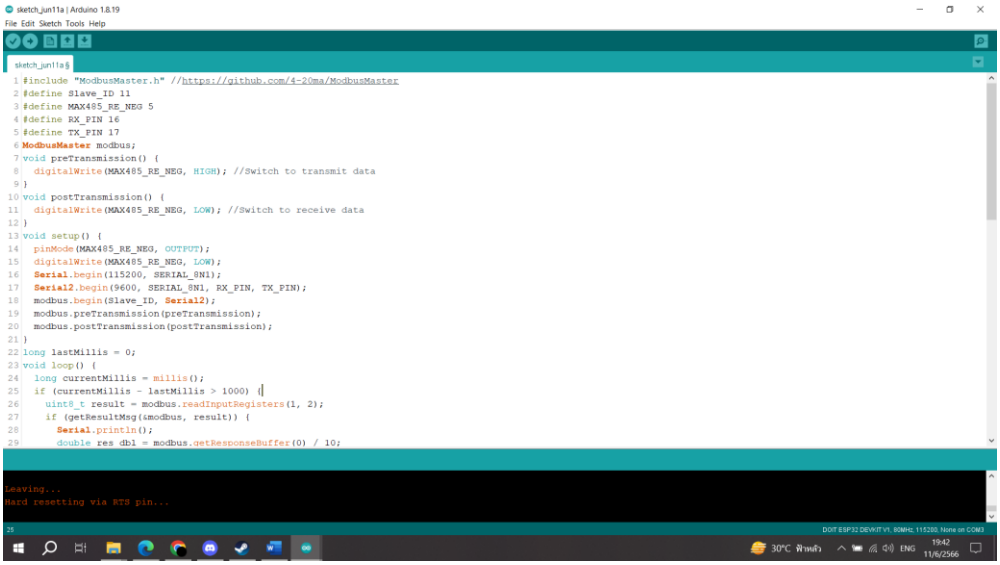
4/4: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_201 – Read Modbus RTU

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

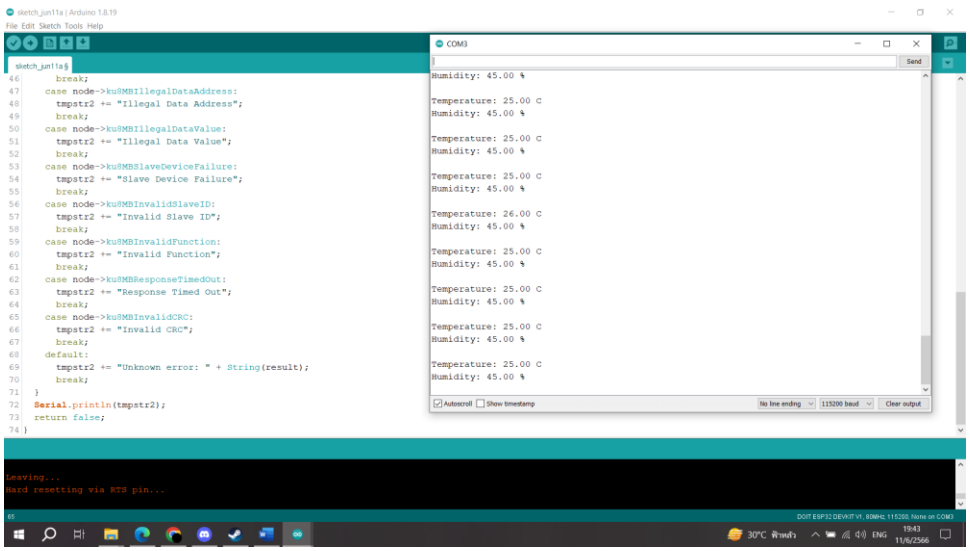
< โปรแกรมทดสอบ >




```
#include "ModbusMaster.h" //https://github.com/4-20ma/ModbusMaster
#define Slave_ID 11
#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.readInputRegisters(1, 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;
}
```

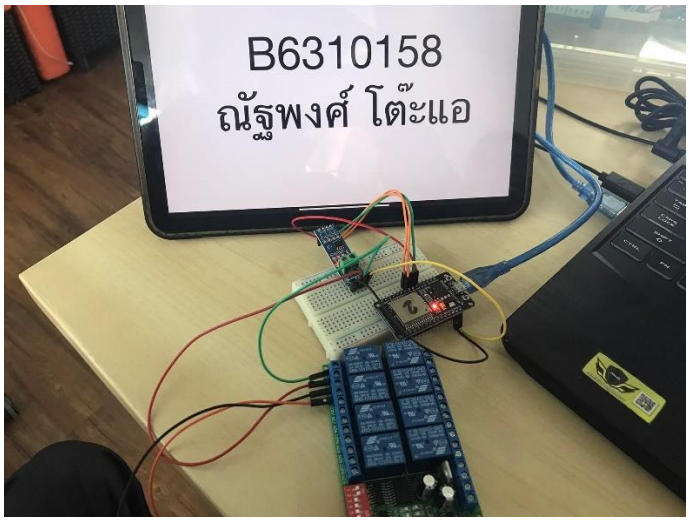
< ผลการทดสอบ >



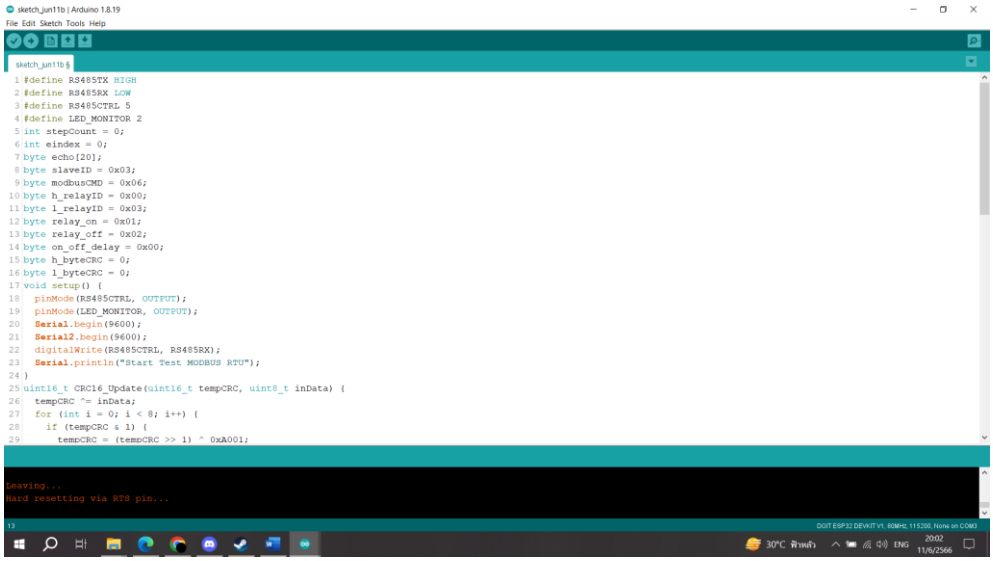
Quiz_202 – Write Modbus RTU

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >



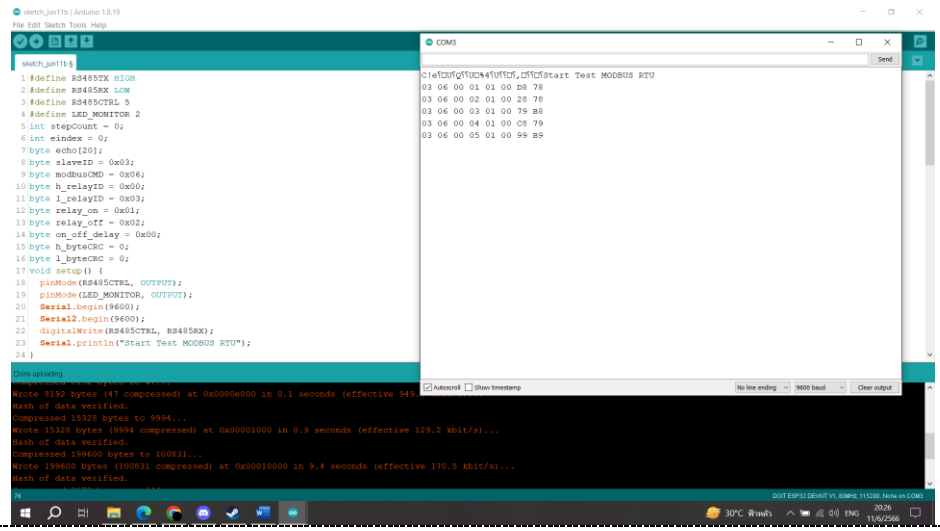
< โปรแกรมทดสอบ >




```
#define RS485TX HIGH
#define RS485RX LOW
#define RS485CTRL 5
#define LED_MONITOR 2
int stepCount = 0;
int eindex = 0;
byte echo[20];
byte slaveID = 0x03;
byte modbusCMD = 0x06;
byte h_relayID = 0x00;
byte l_relayID = 0x03;
byte relay_on = 0x01;
byte relay_off = 0x02;
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);
  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 = 1; relay <= 8; relay++) {
    relayCTRL(relay, relay_on);
    delay(3000);
  }
  for (int relay = 1; relay <= 8; relay++) {
    relayCTRL(relay, relay_off);
    delay(3000);
  }
}
```

< ผลการทดสอบ >

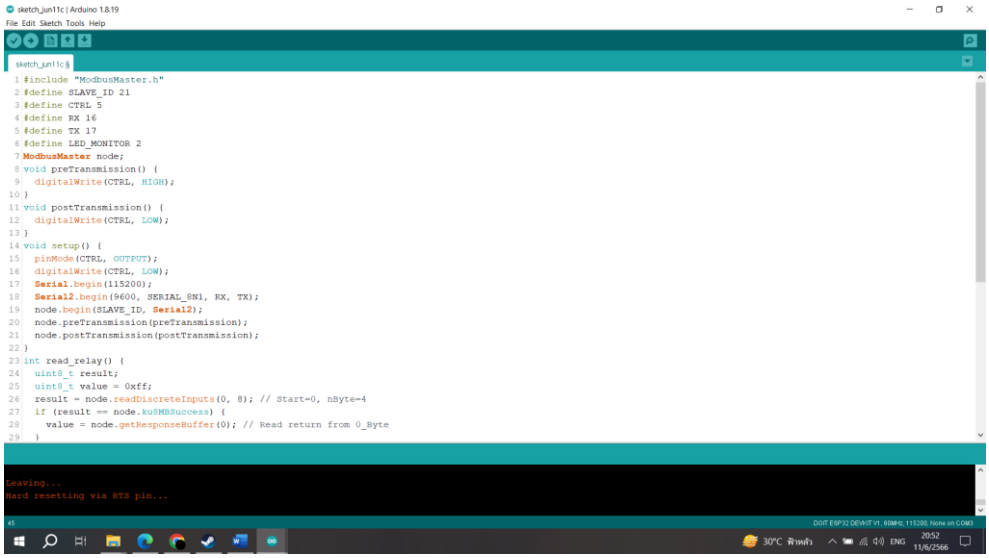


Quiz_203 – Read/Write Modbus RTU

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >

< โปรแกรมทดสอบ >

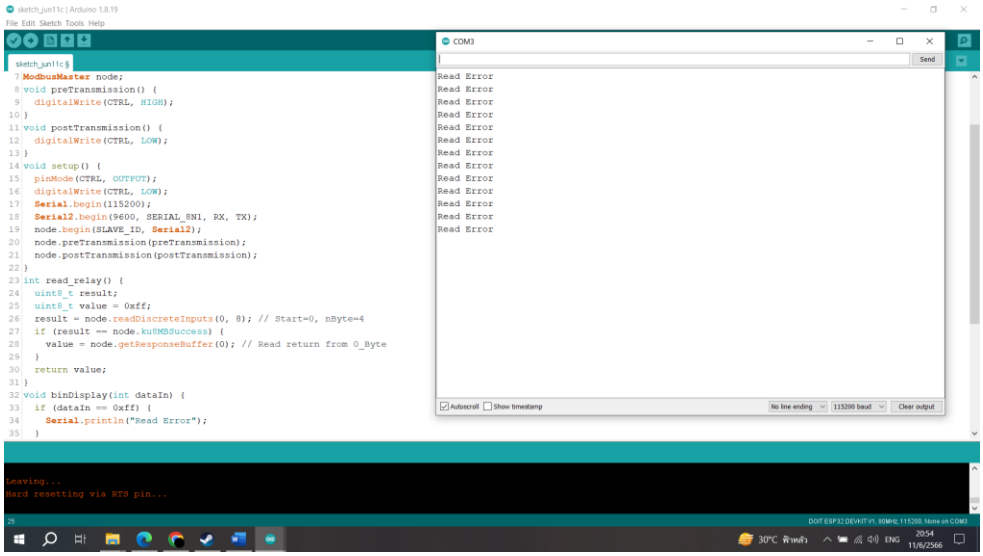



```
#include "ModbusMaster.h"
#define SLAVE_ID 21
#define CTRL 5
#define RX 16
#define TX 17
#define LED_MONITOR 2
ModbusMaster node;
void preTransmission() {
  digitalWrite(CTRL, HIGH);
}
void postTransmission() {
  digitalWrite(CTRL, LOW);
}
void setup() {
  pinMode(CTRL, OUTPUT);
  digitalWrite(CTRL, LOW);
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RX, TX);
  node.begin(SLAVE_ID, Serial2);
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
}
int read_relay() {
  uint8_t result;
  uint8_t value = 0xff;
  result = node.readDiscreteInputs(0, 8); // Start=0, nByte=4
  if (result == node.ku8MBSuccess) {
    value = node.getResponseBuffer(0); // Read return from 0_Byte
  }
  return value;
}
void binDisplay(int dataIn) {
  if (dataIn == 0xff) {
    Serial.println("Read Error");
  }
  else {
    Serial.print(dataIn >> 1 & 1);
```

```
    Serial.print(dataIn >> 0 & 1);
    Serial.println();
  }
}
void loop() {
  node.writeSingleCoil(0, 0x00FF); delay(2000); // On Relay0
  binDisplay(read_relay());
  node.writeSingleCoil(0, 0x0000); delay(2000); // Off Relay0

  node.writeSingleCoil(1, 0x00FF); delay(2000); // On Relay1
  binDisplay(read_relay());
  node.writeSingleCoil(1, 0x0000); delay(2000); // Off Relay1
}
```

< ผลการทดสอบ >



Quiz_204 – PLC Test

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