XS_ETH_12DI7DO arduino 2560工控板 配合TH-S71 多合一环境传感器使用。

准备器件：

XS_ETH_12DI7DO 工控板*1

TH-S71 传感器模块*1

方口USB线*1

以太网线*1

杜邦线*4

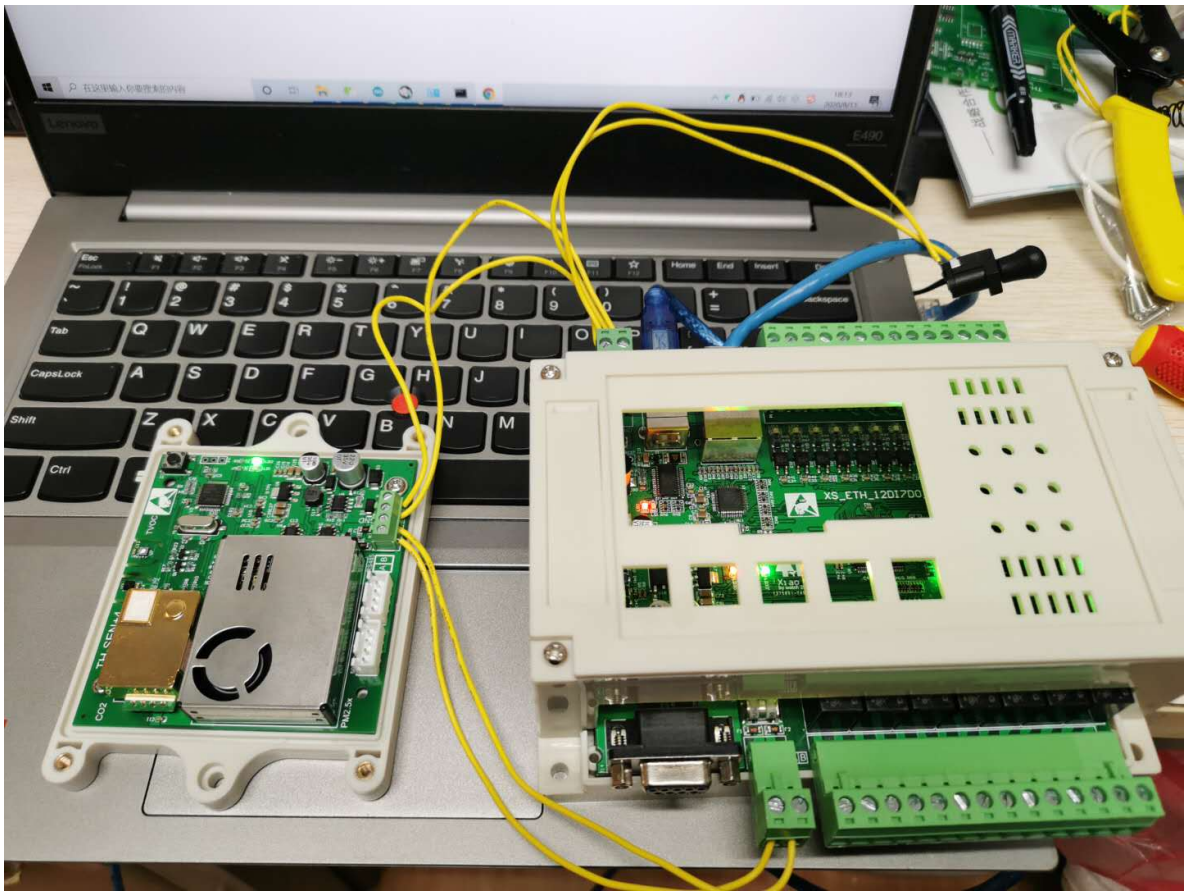
24V 直流电源*1

arduino 2560工控板 通过RS485 连接TH-S71环境传感器，使用了标准的MODBUS-RTU协议读取传感器数值，传感器协议文档

可通过以下链接下载（https://github.com/watchzhong/TH-S71-SENSOR-ALL-file/blob/master/002-Modbus-RTU%E5%A4%9A%E4%BC%A0%E6%84%9F%E5%99%A8%E7%9B%91%E6%8E%A7.md）。

2560工控板 通过以太网把数据上传到局域网进行实时查看。

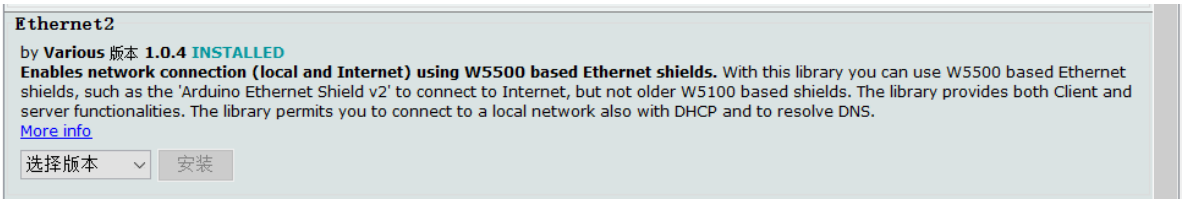
接线方式如下：

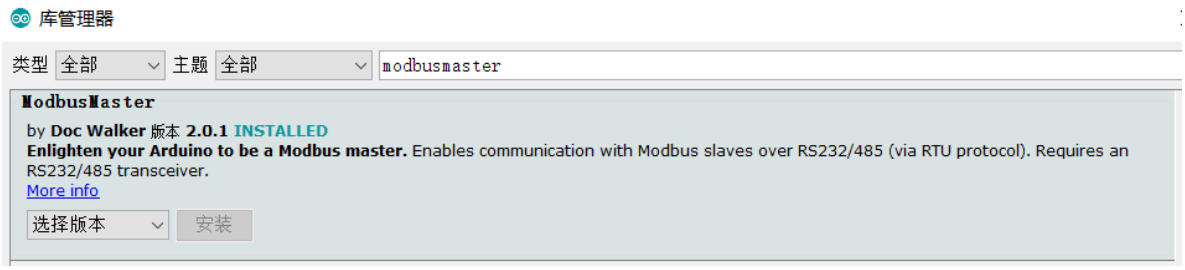


XS_ETH_12DI7DO 和TH-S71 两根RS485线连接，输入24V电源。XS_ETH_12DI7DO USB连接电脑，网线也连接电脑的以太网 网口。

程序代码如下：

先要安装ethernet2和modbusmaster库，打开arduino ide 在线安装便可。






```
#include <Dhcp.h>
#include <Dns.h>
#include <Ethernet2.h>
#include <EthernetClient.h>
#include <EthernetServer.h>
#include <EthernetUdp2.h>
#include <Twitter.h>
#include <util.h>

#include <ModbusMaster.h>

ModbusMaster node;

#define MODBUS_EN   6

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void ethernet_init()
{
```

```
  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

void preTransmission()
{
  digitalWrite(MODBUS_EN, 1);
}

void postTransmission()
{
  digitalWrite(MODBUS_EN, 0);
}

void modbus_init()
{
    // use Serial (port 0); initialize Modbus communication baud rate
  Serial2.begin(9600);

  pinMode(MODBUS_EN,OUTPUT);
  digitalWrite(MODBUS_EN,1);

  // communicate with Modbus slave ID 2 over Serial (port 0)
  node.begin(1, Serial2);
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
}

void modbus_test()
{
  static uint32_t i;
  uint8_t j, result;
  uint16_t data[6];

   i++;

  // set word 0 of TX buffer to least-significant word of counter (bits 15..0)
//  node.setTransmitBuffer(0, lowWord(i));

  // set word 1 of TX buffer to most-significant word of counter (bits 31..16)
//  node.setTransmitBuffer(1, 0x3344);

  // slave: write TX buffer to (2) 16-bit registers starting at register 0
//  result = node.writeMultipleRegisters(0, 2);

  // slave: read (6) 16-bit registers starting at register 2 to RX buffer

  delay(600);
  Serial.println("read holding register");
  result = node.readInputRegisters(1, 8);

  // do something with data if read is successful
  if (result == node.ku8MBSuccess)
  {
```

```
 Serial.println("get resp:");
 for (j = 0; j < 8; j++)
 {
   data[j] = node.getResponseBuffer(j);
   Serial.println(data[j]);
 }
 Serial.println();
```

```
 }
}

void setup() {
  // put your setup code here, to run once:
  pinMode(LED_BUILTIN,OUTPUT);

  Serial.begin(9600);

  ethernet_init();

  modbus_init();

}

void loop() {
  // put your main code here, to run repeatedly:

  //modbus get data

  uint8_t j, result;
  uint16_t data[8];

//  Serial.println("read holding register");
//  result = node.readInputRegisters(1, 8);

  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

//        Serial.write(c);
```

```
      // if you've gotten to the end of the line (received a newline
      // character) and the line is blank, the http request has ended,
      // so you can send a reply
      if (c == '\n' && currentLineIsBlank) {
        // send a standard http response header
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection: close");  // the connection will be closed
  after completion of the response
        client.println("Refresh: 2");  // refresh the page automatically every 5
  sec
```

```
        client.println();
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");

      // transmit data to TCP-w5500
```

//   if (result == node.ku8MBSuccess)
      {

```
          Serial.println("get modbus resp>");

        //  Serial.println("read holding register");
          result = node.readInputRegisters(1, 8);

          for (j = 0; j < 8; j++)
          {
            data[j] = node.getResponseBuffer(j);

          }

          uint16_t data_temp = data[0];
          uint16_t data_humi = data[1];

          uint16_t data_sgpco2 = data[2];
          uint16_t data_sgpvoc = data[3];

          uint16_t data_mhco2 = data[4];
          uint16_t data_pm1 = data[5];
          uint16_t data_pm25 = data[6];
          uint16_t data_pm10 = data[7];

          client.print("<center>");
          client.print("<h3>Multiple sensor monitor</h3>");
          client.print("<h5>Modbus RTU with W5500 nethernet</h5>");

          client.print("<table border=\"1\">");
          client.print("<tr>");
          client.print("<th>sensor</th>");
          client.print("<th>value</th>");
          client.print("</tr>");

          client.print("<tr>");
          client.print("<td>");
          client.print("SHT30 Temp");
          client.print("</td>");
          client.print("<td>");
          client.print(data_temp/100.00);
          client.print(" C");
          client.print("</td>");
          client.print("</tr>");

          Serial.print("SHT30 Temp:");
          Serial.println(data_temp/100.00);

          client.print("<tr>");
          client.print("<td>");
```

```
        client.print("SHT30 Humi");
        client.print("</td>");
        client.print("<td>");
        client.print(data_humi/100.00);
        client.print(" %");
        client.print("</td>");
        client.print("</tr>");

        Serial.print("SHT30 Humi:");
        Serial.println(data_humi/100.00);

        client.print("<tr>");
        client.print("<td>");
        client.print("SGP30 eCO2");
        client.print("</td>");
        client.print("<td>");
        client.print(data_sgpco2/10.0);
        client.print(" ppm");
        client.print("</td>");
        client.print("</tr>");

        Serial.print("SGP30 eCO2:");
        Serial.println(data_sgpco2/10.0);

        client.print("<tr>");
        client.print("<td>");
        client.print("SGP30 TVOC");
        client.print("</td>");
        client.print("<td>");
        client.print(data_sgpvoc/10.0);
        client.print(" ppb");
        client.print("</td>");
        client.print("</tr>");

        Serial.print("SGP30 TVOC:");
        Serial.println(data_sgpvoc/10.0);

        client.print("<tr>");
        client.print("<td>");
        client.print("MHZ19B CO2");
        client.print("</td>");
        client.print("<td>");
        client.print(data_mhco2*10.0);
        client.print(" ppm");
        client.print("</td>");
        client.print("</tr>");

        Serial.print("MHZ19B CO2:");
        Serial.println(data_mhco2*10);

        client.print("<tr>");
        client.print("<td>");
        client.print("PM1.0");
        client.print("</td>");
        client.print("<td>");
        client.print(data_pm1);
        client.print(" ug/m3");
        client.print("</td>");
```

```
          client.print("</tr>");

          Serial.print("PM1.0:");
          Serial.println(data_pm1);

          client.print("<tr>");
          client.print("<td>");
          client.print("PM2.5");
          client.print("</td>");
          client.print("<td>");
          client.print(data_pm25);
          client.print(" ug/m3");
          client.print("</td>");
          client.print("</tr>");

          Serial.print("PM2.5:");
          Serial.println(data_pm25);

          client.print("<tr>");
          client.print("<td>");
          client.print("PM10");
          client.print("</td>");
          client.print("<td>");
          client.print(data_pm10);
          client.print(" ug/m3");
          client.print("</td>");
          client.print("</tr>");

          Serial.print("PM10:");
          Serial.println(data_pm10);

          client.print("</table>");
          client.print("</center");

          Serial.println();

        }

        // output the value of each analog input pin
//        for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
//          int sensorReading = analogRead(analogChannel);
//          client.print("analog input ");
//          client.print(analogChannel);
//          client.print(" is ");
//          client.print(sensorReading);
//          client.println("
");
//        }

        client.println("</html>");
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
```

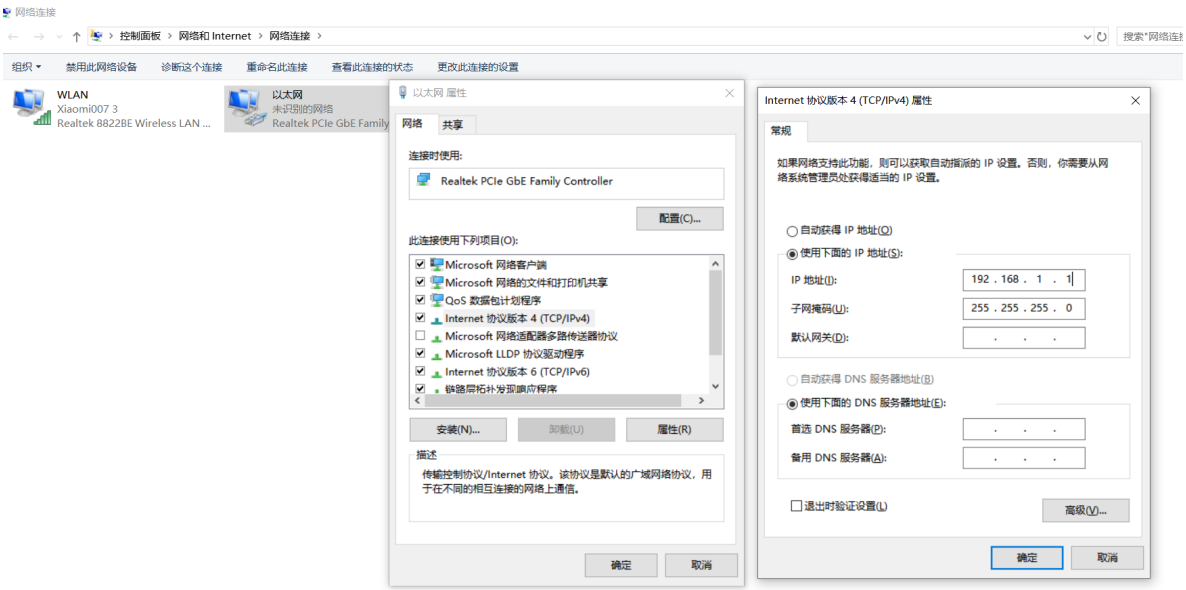
```
    }
    else if (c != '\r') {
      // you've gotten a character on the current line
      currentLineIsBlank = false;
    }
  }
}
// give the web browser time to receive the data
delay(1);
// close the connection:
client.stop();
Serial.println("client disconnected");
```

}

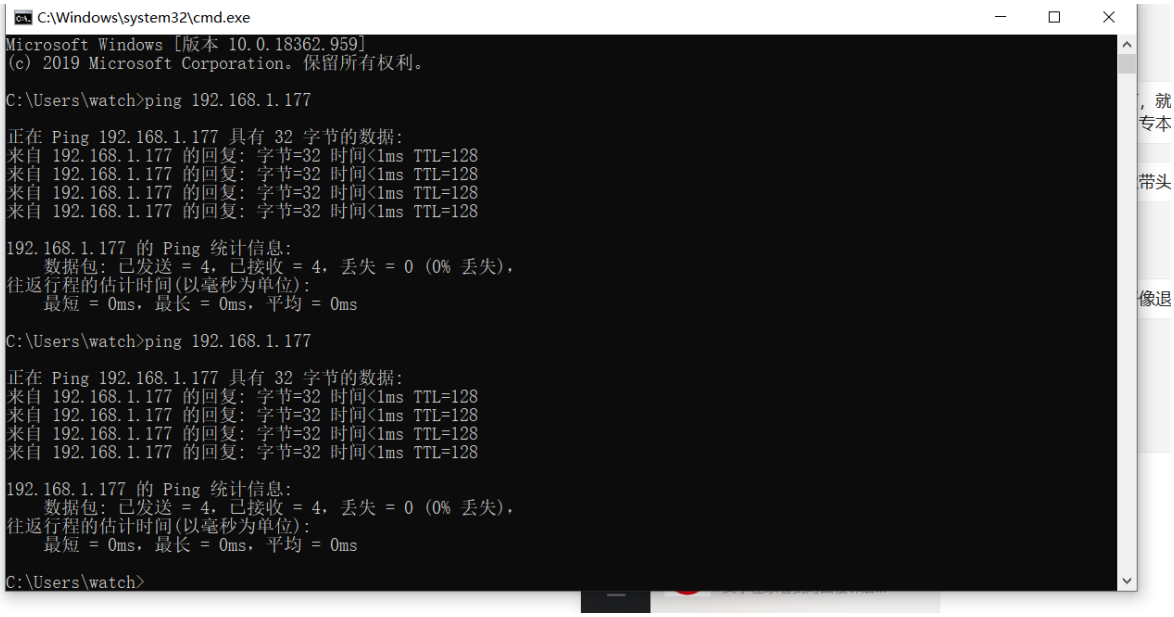
}

把代码通过USB烧录到 2560工控板，打开串口调试窗，显示本地局域网的IP地址。

这时，我们先要对电脑进行IP设置。



然后连接尝试...



如图所示，电脑端和2560以太网连接正常，即可以进行下一步的操作，在浏览器输入192.168.1.177 回车，传感器的数值如下如所示。

Multiple sensor monitor

Modbus RTU with W5500 nethernet

sensor	value
SHT30 Temp	28.52 C
SHT30 Humi	53.80 %
SGP30 eCO2	458.00 ppm
SGP30 TVOC	27.00 ppb
MHZ19B CO2	1130.00 ppm
PM1.0	135 ug/m3
PM2.5	206 ug/m3
PM10	213 ug/m3

同时，传感器数值也在串口端显示，再次打开串口调试助手即可。

```
client disconnected
new client
get modbus resp>
SHT30 Temp:28.76
SHT30 Humi:53.02
SGP30 eCO2:403.00
SGP30 TVOC:8.00
MHZ19B CO2:1150
PM1.0:135
PM2.5:214
PM10:232
```