CRYSOUND

CRY2110 Series Noise Sensor

User Manual

Catalog

1 Product Brief Introduction

The CRY2110 series noise sensor is a new type of noise testing equipment independently developed by our company for industrial applications. The product performance complies with IEC61672 standards.

Fig 1.1 CRY2110 Series Noise Sensor

The CRY2110 series noise sensor is compact, with a sturdy and corrosion-resistant stainless steel housing. It is equipped with a precision measurement microphone and features an integrated preamplifier, high-precision ADC, and high-speed DSP processor. Its excellent hardware design provides a dynamic range of 110 dB, enabling it to measure ambient noise as low as 25 dBA.

The Modbus version of the CRY2110 series noise sensor uses an RS-485 bus interface as the data communication port and employs a communication protocol based on Modbus RTU. Additionally, to facilitate industrial automation applications, the CRY2110 series noise sensor (Modbus version) also offers 4-20 mA current output and 1-5 V or 2-10 V voltage output interfaces.

2 Features

[Perform Standard] IEC 61672 - 1

[Two Output Methods] 4-20 mA and RS485 output, transmission distance exceeds 1 km

[Easy for Integration] Connect to PC, PLC, DCS and other controllers, get noise data easily

[Perfect Performance] 110 dB dynamic range, can measurement lower to 25 dBA noise

[Small Appearance] Small size with 25 mm diameter and 115 mm length

[Wide Range Power Supply] From DC 5 V to 24 V

[Reliable and Durable] Stainless steel shell, robust in harsh environments

3 Typical Application

Fig 3.1 System Integrated Noise Monitoring

Fig 3.2 Construction Noise Monitoring

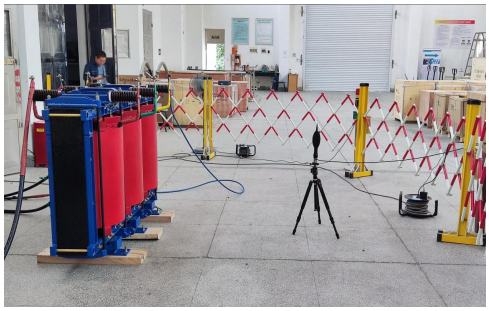

Fig 3.3 Product Noise Loudness Monitoring

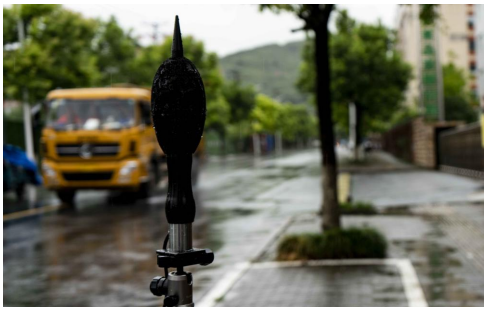

Fig 3.4 Urban Traffic Noise Monitoring

4 Technical Specifications

Product Model	CRY2112	CRY2110
Applicable standards	IEC61672 Class 1	IEC61672 Class 2
Measurement Range	25 ~ 130 dBA with CRY331/CRY333 microphone（Default） 20 ~ 130 dBA with CRY311 microphone 25 ~ 140 dBA with CRY371 microphone	
Dynamic Range	≥ 110 dB, no range selection needed	
AD Sampling rate	48k Hz	
Detection Method	Digital	

Background Noise	19 dB(A), 20 dB(C), 23 dB(Z)	22 dB(A), 23 dB(C), 24 dB(Z)
Frequency Range	10 Hz ~ 20k Hz	
Frequency Weighting	A / C / Z	
Time Weighting	F / S	
Measurement Function	Lp (RS485 output, Voltage or Current output) Leq (RS485 output, Integration time can be customized)	
SPL Output	RS-485; 4-20 mA (default) / DC 1-5 V / DC 2-10 V (Choose one in three)	
Power Supply	DC 5-24 V	
Dimensions	ϕ24.5 mm × 115 mm	
Weight	115 g	
Temp Range	-20 ~ +50 ℃; Relative Humidity ≤ 80%	
Shell Material	Stainless steel, robust in harsh environments	

5 Modbus RTU Communication Protocol

The CRY2110 series noise sensors have developed communication protocols based on the Modbus RTU bus protocol at the protocol layer.

5.1 Modbus Register Definition

Twenty-four types of sound pressure level data are stored in the Input Registers and can be read using Function Code 03. Five device parameters are stored in the Holding Registers and can be modified using Function Code 06.

Noise Sensor Modbus RTU Communication Protocol Description:

Function Code	Function Description	Operative Register Address Range
03	Read Holding Register Status	0x0101-0x0118
06	Write Single Register Status	0x0001-0x0005

Device Parameter Registers:

Data Content			Remarks
Address	Content Definition		/
	High Byte	Low Byte	/
0x0000	0x00	Address	Default is 1, i.e., 0x0001
0x0002	0x00	Current Output Type	/
0x0003	0x00	Leq Calculation Time	Unit in seconds, default is 600 seconds, i.e., 0x0258
0x0004	0x00	Lpmax Calculation Time	Calculated in conjunction with Lpeak, Lmax, and Lmin;unit in seconds, default is 600 seconds, i.e., 0x0258
0x0005	0x00	Target Calibration Sound Pressure Level	The calibration target is generally 93.9dB, set to 0x03AB

Modbus Communication Protocol Register Content Definition:

Sound Pressure Level Data Registers				
Address	0x0101	0x0102	0x0103	0x0104
Parameter	LAF	LAS	LCF	LCS
Address	0x0105	0x0106	0x0107	0x0108

Parameter	LZF	LZS	LAFmax	LASmax
Address	0x0109	0x010A	0x010B	0x010C
Parameter	LCFmax	LCSmax	LZFmax	LZSmax
Address	0x010D	0x010E	0x010F	0x0110
Parameter	LAFmin	LASmin	LCFmin	LCSmin
Address	0x0111	0x0112	0x0113	0x0114
Parameter	LZFmin	LZSmin	LApeak	LCpeak
Address	0x0115	0x0116	0x0117	0x0118
Parameter	LZpeak	LAeq, T	LCeq, T	LZeq, T

Serial Communication Parameters:

Baud Rate	9600	Data Bits	8bit
Parity Bit	None	Stop Bits	1bit
Data Format	HEX		

5.2 CRY2110 series Modbus RTU Command Construction and Response

5.2.1 Function Code 03 Command

Request Structure:

bit Address + 8-bit Function Code + 16-bit Starting Address of Registers to Read + 16-bit Number of Registers to Read + 16-bit CRC Check.

Protocol Response:

8-bit Address + 8-bit Function Code + 8-bit Byte Count + Several Groups of 16-bit Data + 16-bit CRC Check.

5.2.2 Function Code 06 Command

Request Structure:

8-bit Address + 8-bit Function Code + 16-bit Address of Register + 16-bit Number of Registers + 16-bit CRC Check.

Protocol Response:

bit Address + 8-bit Function Code + 16-bit Address of Register + 16-bit Number of Registers + 16-bit CRC Check.

Note: Data segments are structured with high byte first and low byte second. The CRC check value is structured with low byte first and high byte second.

5.3 CRY2110 series Modbus RTU Sound Pressure Level Data Parsing

Input Registers	01XX High Byte	01XX Low Byte
Definition	Sound Pressure Level Data High Byte	Sound Pressure Level Data Low Byte

Sound Pressure Level Calculation Formula:

Sound Pressure Level = (HiByte × 256 + LowByte) / 10

Where::

- **HiByte:** The value of the high byte of the Modbus register data in decimal.
- **LowByte:** The value of the low byte of the Modbus register data in decimal.

Address data is represented in hexadecimal. The sensor only uses the low 8 bits as the device address, with the high 8 bits defaulting to 0x00 when stored.

Holding Registers	0x0001 High Byte	0x0001 Low Byte
Definition	Reserved (Read 0x00)	8-bit Address

5.4 CRY2110 series Modbus RTU Data Communication Example

E.g.1: Request the sensor with address 01 to return the sound pressure level LAF.

Master Sends:

0x01 0x03 0x01 0x00 0x00 0x01 CR85 CRF6

(Note: 0x indicates that the following numbers are in hexadecimal. CRCH CRCL is the CRC check, which can be automatically added using a serial debugging assistant with Modbus CRC16.)

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	03	
Register Address High Byte	01	
Register Address Low Byte	00	
Register Count High Byte	00	
Register Count Low Byte	01	
CRC Check Low Byte	85	
CRC Check High Byte	F6	

Slave Responds:

0x01 0x03 0x02 0x01 0xD9 CR78 CR4E

Slave Responding Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	03	
Returned Data Count	02	02 indicates returning 2 groups of 8-bit data
Register Address High Byte	01	
Register Address Low Byte	D9	
CRC Check Low Byte	78	
CRC Check High Byte	4E	

E.g.2: Read All Current Sound Pressure Level Parameters for Device Address 01

Master Sends:

0x01 0x03 0x01 0x00 0x00 0x18 CR44 CR3C

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	03	
Register Address High Byte	01	
Register Address Low Byte	00	

Register Count High Byte	00	
Register Count Low Byte	18	
CRC Check Low Byte	44	
CRC Check High Byte	3C	

Slave Responds:

0x01 0x03 0x30 Parameter 1 - Parameter 24 CRCH CRCL

Slave Responding Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	04	
Returned Data Count	12	12 indicates returning 18 groups of 8-bit data
Register Address High Byte	01	
...	...	...
...	...	...
Register Address Low Byte	A7	
CRC Check Low Byte	CH	
CRC Check High Byte	CL	

E.g.3: Modify Device Address (Change Sensor Address 01 to 02)

Master Sends:

0x01 0x06 0x00 0x00 0x00 0x02 CR08 CRDB

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	06	
Register Address High Byte	00	
Register Address Low Byte	00	
Register Count High Byte	00	
Register Count Low Byte	02	
CRC Check Low Byte	08	
CRC Check High Byte	DB	

Slave Does Not Return Data

E.g.4:Modify Unknown Device Address to Sensor Address 01

Master Sends:

0x00 0x06 0x00 0x00 0x00 0x01 CR49 CRDB

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	00	
Function Code	06	

Register Address High Byte	00	
Register Address Low Byte	00	
Register Count High Byte	00	
Register Count Low Byte	01	
CRC Check Low Byte	49	
CRC Check High Byte	DB	

Slave Does Not Return Data

E.g.5: Set Calibration Target Sound Pressure Level to 93.9 dB

Master Sends:

0x01 0x06 0x00 0x05 0x03 0xAB CRD8 CR84

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	06	
Register Address High Byte	00	
Register Address Low Byte	05	

Register Count High Byte	03	
Register Count Low Byte	AB	
CRC Check Low Byte	D8	
CRC Check High Byte	84	

Slave Responds:

0x01 0x06 0x00 0x05 0x03 0xAB CRD8 CR84

Slave Responding Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	06	
Returned Data Count	00	02 indicates returning 2 groups of 8-bit data
Register Address High Byte	03	
Register Address Low Byte	AB	
CRC Check Low Byte	D8	
CRC Check High Byte	84	

E.g.6: Set Leq Value to 120 s

Master Sends:

0x01 0x06 0x00 0x03 0x00 0x78 CR79 CRE8

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	06	
Register Address High Byte	00	
Register Address Low Byte	03	
Register Count High Byte	00	
Register Count Low Byte	78	
CRC Check Low Byte	79	
CRC Check High Byte	E8	

Slave Responds:

0x01 0x06 0x00 0x03 0x00 0x78 CR79 CRE8

Slave Responding Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	06	
Register Address High Byte	00	
Register Address Low	03	

Byte		
Register Count High Byte	00	
Register Count Low Byte	78	
CRC Check Low Byte	79	
CRC Check High Byte	E8	

E.g.7: Sound Signal Calibration

Master Sends:

0x01 0x05 0x00 0x01 0xFF 0x00 CRDD CRFA（Set）

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	05	
Register Address High Byte	00	
Register Address Low Byte	01	
Register Count High Byte	FF	
Register Count Low Byte	00	

CRC Check Low Byte	DD	
CRC Check High Byte	FA	

Slave Responds:

0x01 0x05 0x00 0x01 0xFF 0x00 CRDD CRFA

Slave Responding Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	05	
Register Address High Byte	00	
Register Address Low Byte	01	
Register Count High Byte	FF	
Register Count Low Byte	00	
CRC Check Low Byte	DD	
CRC Check High Byte	FA	

Master Sends:

0x01 0x03 0x01 0x00 0x00 0x01 CR85 CRF6(Verify)

Master Sending Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	03	

Register Address High Byte	01	
Register Address Low Byte	00	
Register Count High Byte	00	
Register Count Low Byte	01	
CRC Check Low Byte	85	
CRC Check High Byte	F6	

Slave Responds:

0x01 0x03 0x02 0x03 0xAB CRF9 CR0B

Slave Responding Data	HEX(Hexadecimal)	Remarks
Slave Address	01	
Function Code	03	
Register Address High Byte	02	02 indicates returning 2 groups of 8-bit data
Register Address Low Byte	03	
Register Count High Byte	AB	
Register Count Low	F9	

Byte		
CRC Check Low Byte	0B	

6 Connection and Communication of Serial Port

6.1 Communication Parameters of RS-485 Interface

RS-485 serial port communication protocol:

Baud rate	9600	Data bit	8
Parity bit	None	Stop bit	1
Data format	Hexadecimal		

6.2 RS-485 to USB Connector (optional)

Use optional connector which has been supplied 5 V power, standard cable constitutes the following connection mode.

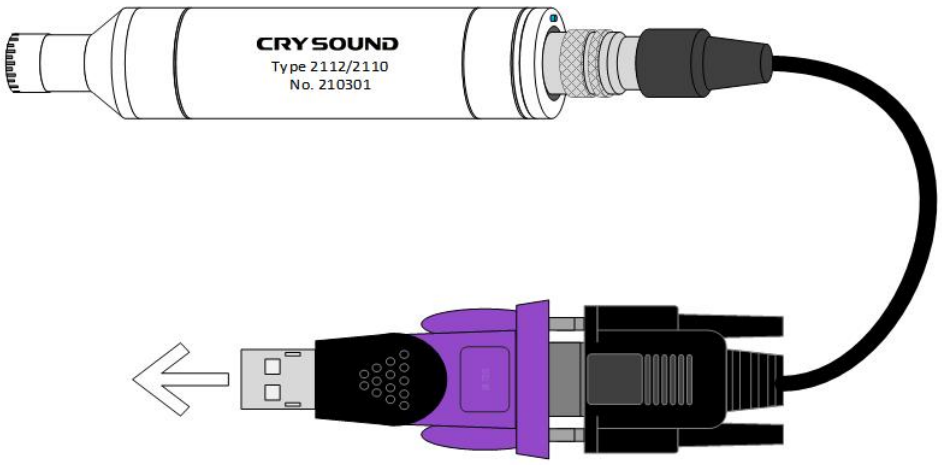

Fig 6.1 Connection diagram

After connection, use serial port send data for getting corresponding response.

6.3 CRY208 Channel Allocator (optional)

CRY208 can connect to PC, 8 slave interfaces can connect to noise sensor or host interface of other CRY208 to meet the cascade requirements. Computer controls all noise sensors by software to get data.

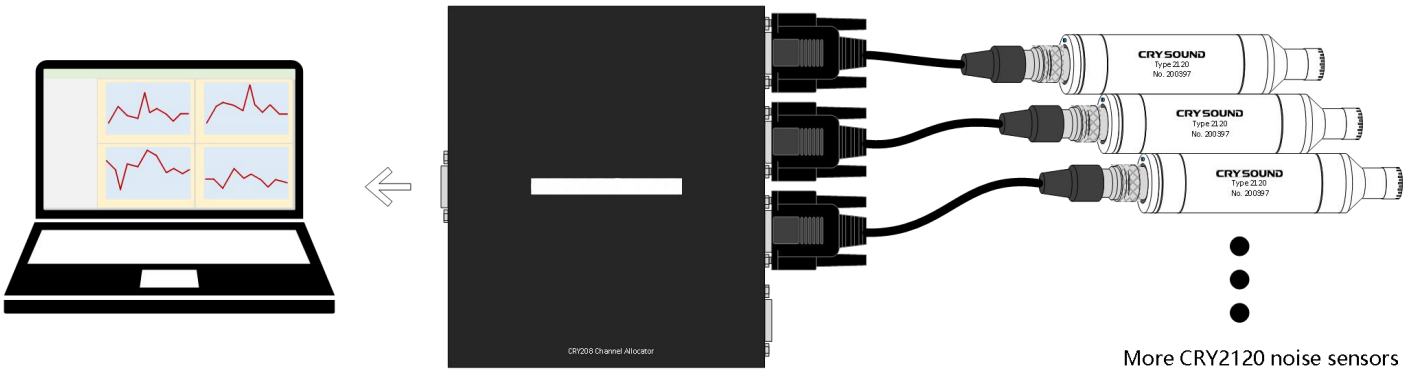

Fig 6.2 CRY208 connection diagram

7 Analog Quantity Connection and Signal Transmission

7.1 4-20 mA Analog Quantity and SPL Calculation

4-20 mA output has three pins which are power pin (5-24 V), 4-20 mA analog output pin and GND pin.

When using 4-20 mA analog quantity output, 20-24 V power supply voltage is recommended.

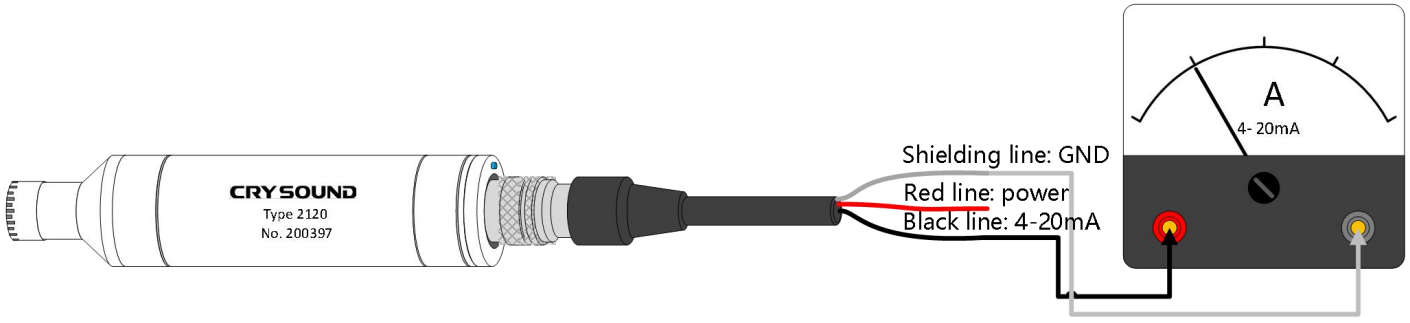

Fig 7.1 4-20 mA analog quantity output connection diagram

SPL output range 20-140 dB, corresponding current 4-20 mA, formula of calculating SPL according to current:

In above formula:

SPL: actual SPL

I: current value of analog output

Current corresponding to SPL:

Current(mA)	SPL(dBSPL)	Current(mA)	SPL(dBSPL)
4	20	13	87.5
5	27.5	14	95
6	35	15	102.5
7	42.5	16	110
8	50	17	117.5
9	57.5	18	125
10	65	19	132.5
11	72.5	20	140
12	80		

7.2 Voltage Analog Quantity and SPL Calculation

Voltage output has three pins which are power pin (5-24 V), 4-20 mA analog output pin (4-20 mA pin outputs voltage analog quantity) and GND pin. When using voltage analog quantity output, 20-24 V power supply voltage is recommended.

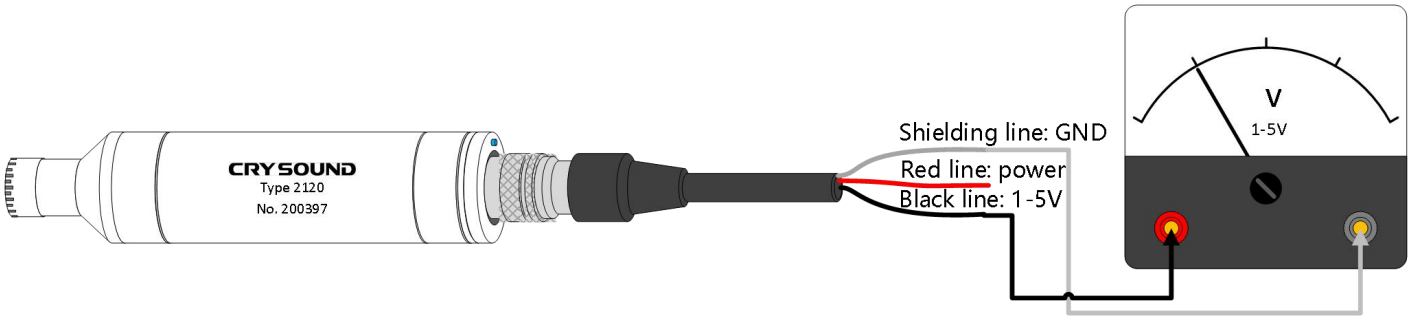

Fig 7.2 Voltage analog quantity output connection diagram

Sensor of voltage analog quantity output: 1-5 V, 2-10 V optional

Formula of calculating SPL and voltage:

(1-5 V)

(2-10 V)

In above formula:

— SPL: actual SPL

— U: voltage analog quantity output value

Voltage corresponding to SPL:

Voltage 1-5 V	Voltage 2-10 V	SPL (dBSPL)
1	2	20
1.5	3	35
2	4	50
2.5	5	65
3	6	80
3.5	7	95
4	8	110
4.5	9	125
5	10	140

8 Calibration

Calibration steps show below:

1. Use sound calibrator CRY5611(1k Hz, 94 dB);
2. Connect the noise sensor correctly;
3. Open sound calibrator after insert noise sensor into it.
4. Press the tail calibration button for one or two seconds, and the LED starts to blink, that means calibration starts.
5. If the rear LED blinks quicker than the start time, it means calibration failed. Please keep all the things steady and try again. If the rear LED does not blink quickly, it means calibration is successful.

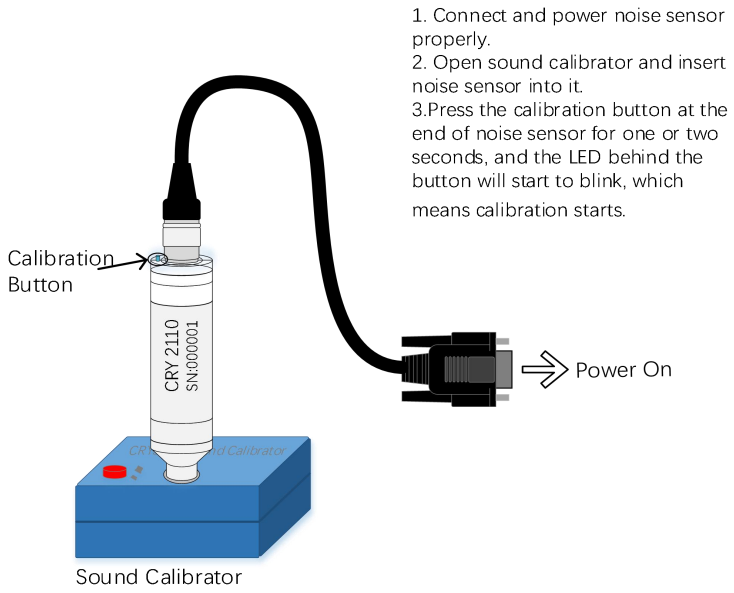

Fig 8.1 Sensor calibration steps

9 Interface Definition

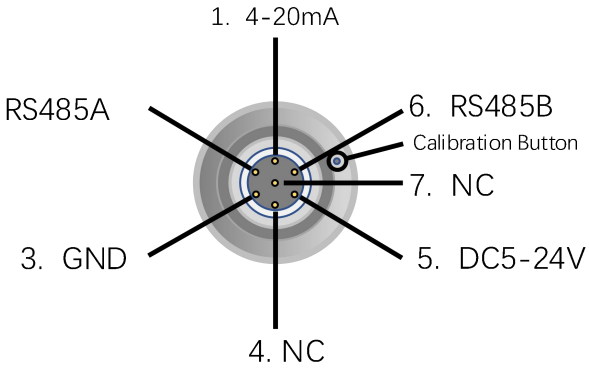

Fig 9.1 Noise sensor's rear ports definition.

For ease of use, the other side of the cable has been connected to DB9 port. The DB9 port definition is as follows:

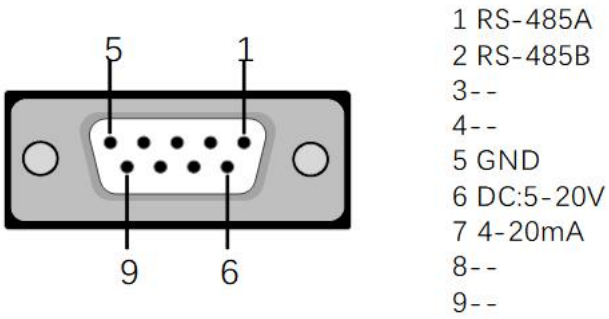

Fig 9.2 The DB9 interface definition

DB9 interface pin definition and cable color definition

Pin	Definition	Color
1	RS485A	Green
2	RS485B	White
3	None	
4	None	
5	GND	Shield
6	DC 5-24 V	Red

7	4-20 mA	Black
8	None	
9	None	

10 Configuration

Typical configuration:

NO	Name	Quantity
1	Noise sensor (CRY2120)	1
2	Windscreen	1
3	Adapter plate	1
4	Standard cable (2 m)	1
5	User manual in paper	1
6	Delivery certificate and warranty card	1

Appendix A How to Use the Adapter Plate

In order to facilitate users to self-wire, each sensor distribution an adapter plate, adapter plate can be directly connected to the standard DB9 female. After wiring the terminal of the adapter plate according to the pin definition of the data line, cable can be secured with white locking tabs and black screws to prevent disconnection when pulling the cable , fix the black screws from external to internal.

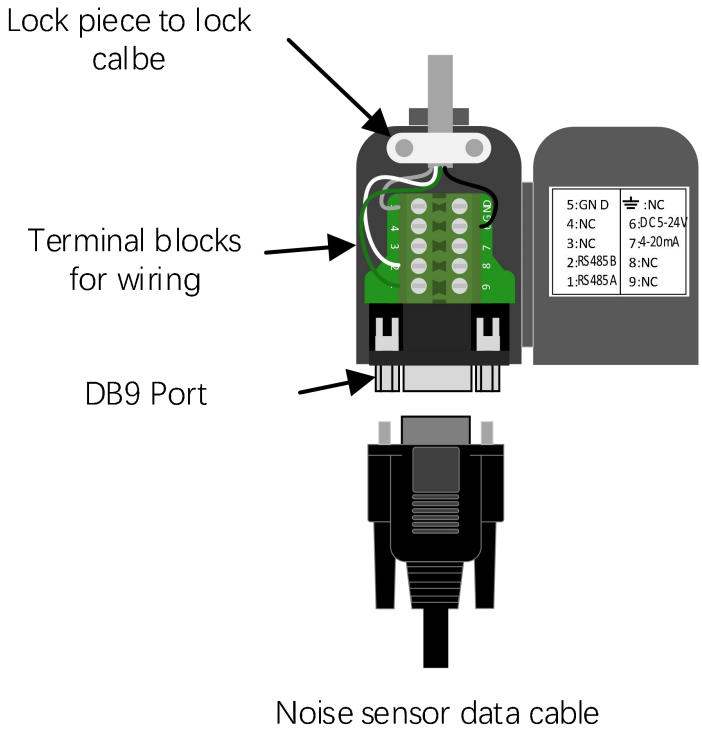

Fig 10.1 Noise sersor data cable

Appendix B Modbus RTU Communication Protocol CRC Check Algorithm

static const unsigned char aucCRCHi[] = {

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,=0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40

};

static const unsigned char aucCRCLo[] = {

0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19,

```
0xD9,0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,0x14, 0xD4, 0xD5, 0x15, 0xD7,
0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2,
0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA,
0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F,
0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1,
0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64,
0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78,
0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75,
0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93,
0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E,
0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E,
0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C, 0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43,
0x83, 0x41, 0x81, 0x80, 0x40
    };
    unsigned short int usMBCRC16 (unsigned char * pucFrame, unsigned short int usLen )
    {
    unsigned char ucCRCHi = 0xFF;
    unsigned char ucCRCLo = 0xFF;
    int    iIndex;

    while( usLen-- ){
    iIndex = ucCRCLo ^ *( pucFrame++ );
    ucCRCLo = ( UCHAR )( ucCRCHi ^ aucCRCHi[iIndex] );
```

```
ucCRCHi = aucCRCLo[iIndex];

}

return (unsigned short int)( ucCRCHi << 8 | ucCRCLo );

}
```