

7Semi

RS485 Temperature & Humidity Sensor Probe

Version 2.0

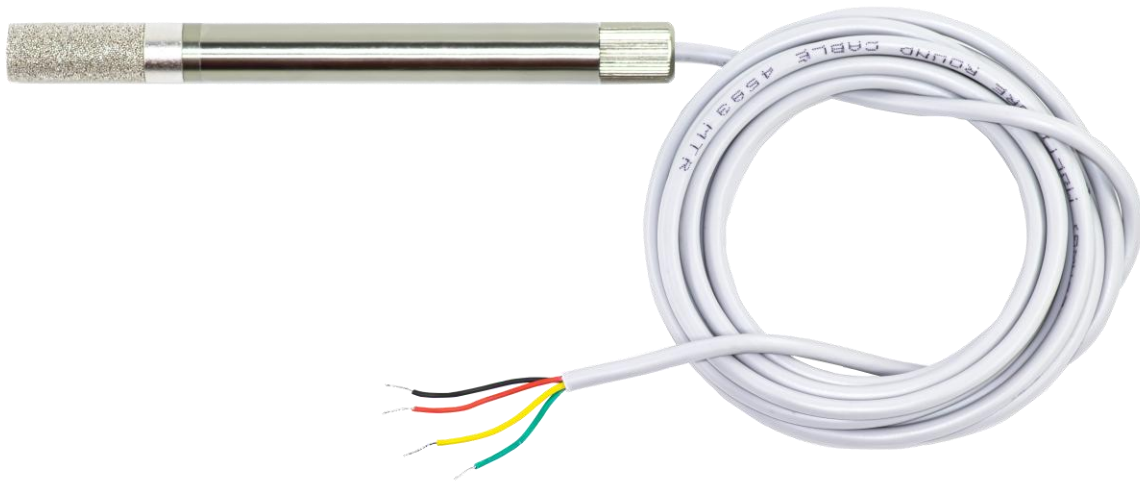


Table of Contents

1.Introduction.....	2
1.1 Features	2
2.Technical specification.....	3
2.1 General specifications	4
3. Hardware setup	5
3.1 Pinouts	5
3.2 Hardware Connection	6
4.RS485 Communication Protocol	7
4.1 Supported Modbus Function Codes.....	7
4.2 Register Address Mapping - Address Quick Reference	7
1) Basic sensor read	8
2) Warm-up & Status Bits	10
3) Uptime counter	11
4) Read Firmware version	12
5) Real ID discovery	13
6) Change measurement period (persisted)	14
7) Apply offsets (persisted)	15
8) Baud-rate change	17
9) Change slave ID	18
10) How to Recover the Slave ID	19
11) Broadcast write (Slave ID 0, No Reply)	21
12) Read diagnostics	21
13) Input Registers Mirror (0x04)	23
14) Optional: Device Identification (MEI 0x2B/0x0E)	24
4.3 Modbus output	25
4.4 How to open the probe?	29
4.5 Connection and sample code for Arduino	30
4.6 How to Install 7Semi Arduino Library.....	31
5.Mechanical specification	32

1.Introduction



This sensor is a high-precision industrial-grade RS485 temperature and humidity measurement device utilizing the advanced sensor. Designed for robustness, reliability, and accurate digital signal processing, it converts ambient temperature and humidity into reliable RS485 signals for seamless integration with centralized monitoring systems.

The sensor supports temperature measurements from -10°C to 70°C and humidity measurements from 0% to 100% RH, with high accuracy and fast response. The probe is enclosed in a **durable metal housing** with a breathable, dust-resistant design, making it suitable for demanding environmental conditions. It is ideal for HVAC systems, industrial environments, building automation, climatology stations, museums, hotels, and closed-loop control systems.

1.1 Features

- Based on the high-accuracy temperature & humidity sensor
- RS485 Modbus RTU output
- Excellent long-term stability
- Wide voltage input: 9–36V DC
- Software configurable Modbus slave address and baud rate
- Fast response and high resolution
- Breathable sensor design

2. Technical specification

The 7Semi RS485 Temperature & Humidity Sensor is engineered to provide precise real-time environmental data in a rugged, industrial-grade form factor. Integrating the advance sensor, it offers high-resolution temperature and humidity readings over a robust RS485 interface using the Modbus RTU protocol. This sensor is optimized for long-term performance and reliability in demanding conditions.

Its digital output enables straightforward integration with industrial control systems, SCADA platforms, and data loggers. The onboard signal conditioning and Modbus address/ baud rate configurability via Software further simplify deployment in diverse applications ranging from environmental monitoring to HVAC automation and industrial process control.

Applications

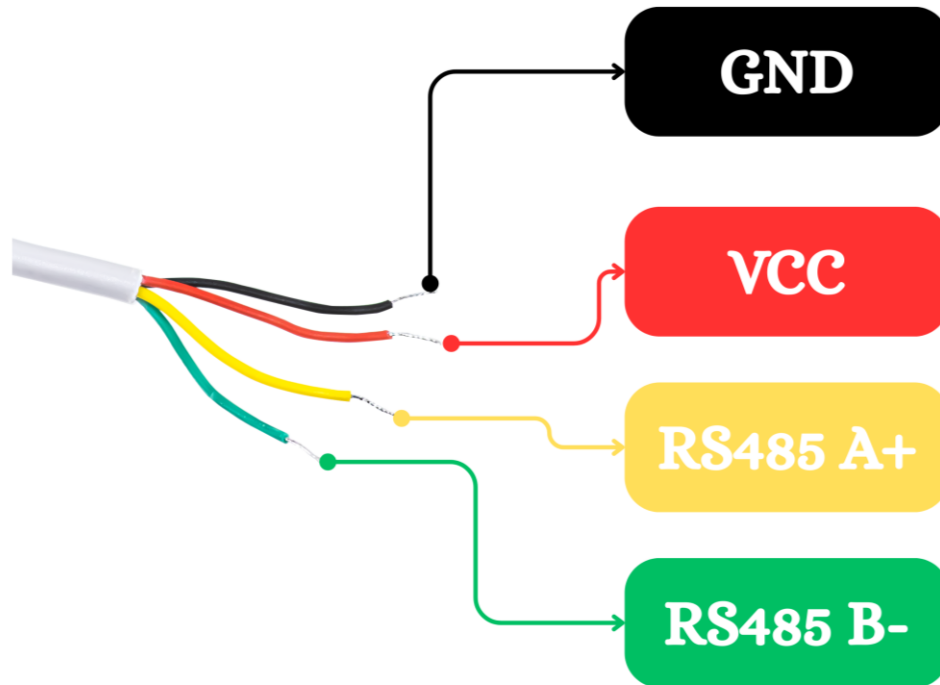
- HVAC control
- Building automation
- Warehouses and greenhouse
- Museums and archive
- Industrial process monitoring
- Weather stations and environmental studies

2.1 General specifications

- **Temperature Measurement Range:** -10°C to +70°C
- **Humidity Measurement Range:** 0 ~ 100% RH
- **Temperature Accuracy:** $\pm 0.1^{\circ}\text{C}$ at 25°C
- **Humidity Accuracy:** $\pm 1\%$ RH at 25°C
- **Sampling Cycle Period:** 1 Sec (Default)
- **Power Supply Voltage:** 9 ~ 36V (DC)
- **Product Size:** 150 mm (L) $\times$ 14.2 mm (D) / 5.91 $\times$ 0.56 in
- **Cable Length :** 2000mm (2 Meter)
- **Communication Interface:** RS485
- **Communication Protocol:** Standard MODBUS RTU protocol
- **Baud Rate:** 9600 (default); configurable to 19200, 38400, 57600, 115200
- **Display Resolution:** Temperature: 0.01°C; Humidity: 0.01% RH

3. Hardware setup

3.1 Pinouts

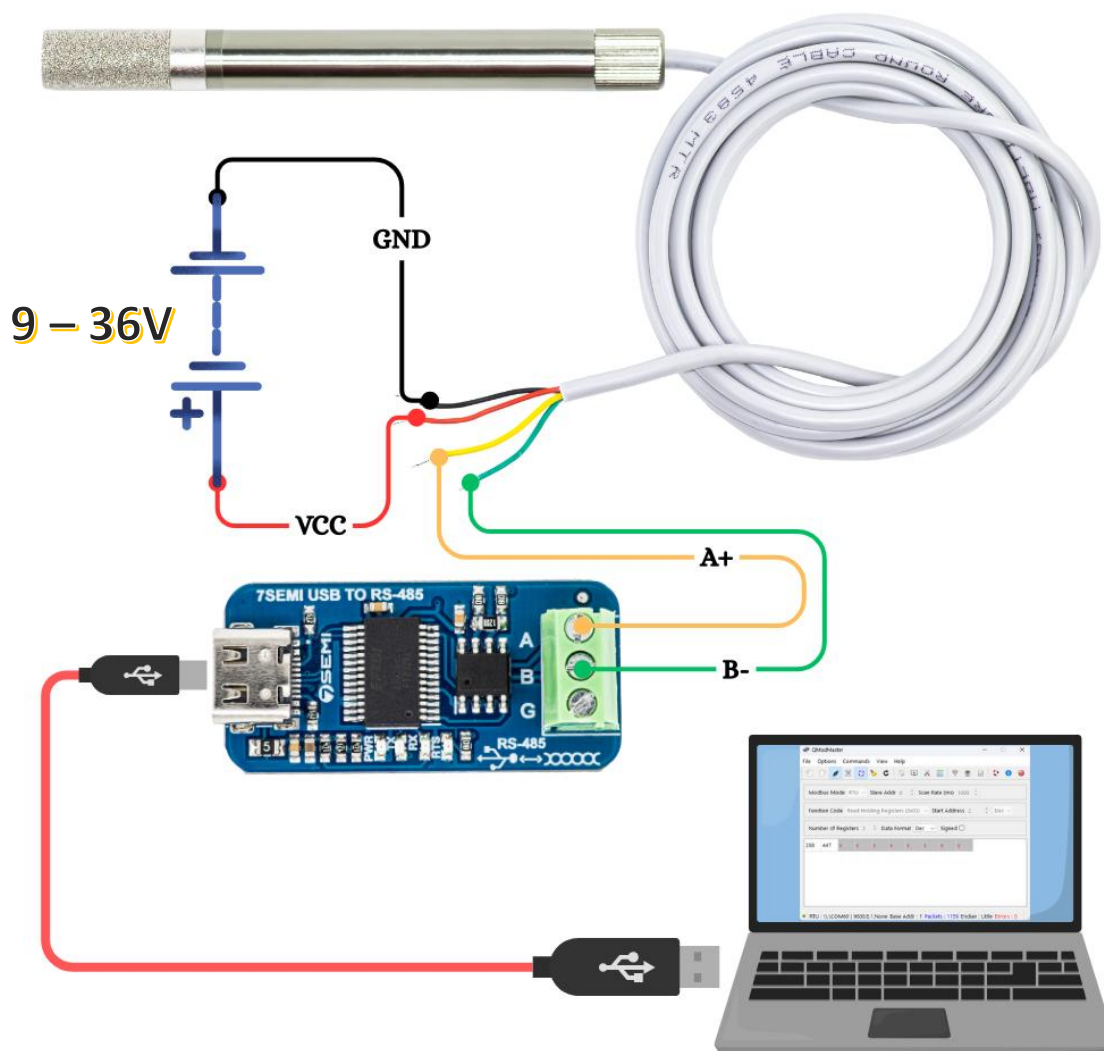


Wire Color	Function
Yellow	RS485 A+
Green	RS485 B-
Red	VCC
Black	GND

Connection Guidelines

- **Power Supply:** Connect the red wire to a 9–36V DC power source and the black wire to ground (GND). Ensure the power supply is stable and within the rated voltage range.
- **RS485 Communication:** Connect the yellow wire to RS485 A+ and the green wire to RS485 B-. Use twisted-pair shielded cable for longer transmission distances and improved noise immunity.

3.2 Hardware Connection



4.RS485 Communication Protocol

The 7Semi RS485 Temperature & Humidity Sensor utilizes the Modbus RTU protocol for digital communication over the RS485 interface. Modbus RTU is a robust, standardized protocol used widely in industrial automation for reliable communication between field devices and control systems such as PLCs, HMIs, SCADA systems, and data loggers.

This sensor acts as a Modbus slave, waiting for requests from a Modbus master and responding with the requested data. All Modbus messages are transmitted in a binary format with CRC (Cyclic Redundancy Check) error detection to ensure communication integrity

4.1 Supported Modbus Function Codes

Function Code	Description
0x03	Read Holding Registers
0x04	Read Input Registers
0x06	Write Single Register
0x10	Write Multiple Registers

- The sensor supports **function code 0x03**, **function code 0x04**, **function code 0x06** and **function code 0x10**, which allows reading holding register values (such as temperature and humidity), Read Input Registers (Mirror of 0x03), Write Single Register and Write Multiple Registers

4.2 Register Address Mapping - Address Quick Reference

Register Address	Purpose	Start Address (Dec)
0x0000	Temperature (T) ×100 (int16)	0
0x0001	Humidity RH ×100 (uint16)	1
0x00E0	diag_mb_crc_err	224
0x00E1	diag_mb_except	225
0x00E4	Uptime (sec) high	228
0x00E5	Uptime (sec) low	229
0x00F0	Status (bit0=Sensor OK	240

	bit2=WARMUP)	
0x0100	“Real” slave ID (read-only)	256
0x0101	New slave ID (write)	257
0x0200	Temp offset $\times 100$ (int16)	512
0x0202	Measurement period (ms)	514
0x0206	RH offset $\times 100$ (int16)	518
0x0210	Baud code (0..4)	528

- Temperature is provided as a signed **16-bit integer (int16)**, while relative humidity is provided as an **unsigned 16-bit integer (uint16)**. Both values are scaled by 100. Divide the raw register value by 100 to obtain the corresponding temperature in °C or relative humidity in %RH.
- Tip: for **signed** values (0x0000, 0x0200, 0x0206) tick **Data Format** → **Signed** in QModMaster.

• Testing Steps For QModMaster

1) Basic sensor read

➤ Test Steps For QModMaster

1. **Function Code:** Read Holding Registers (0x03)
2. **Start Address:** 0
3. **Number of Registers:** 2
4. **Signed:** ON
 - Expect T and RH as integers $\times 100$. E.g., 2579 $\rightarrow$ 25.79 °C; 6665 $\rightarrow$ 66.65 %RH.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: **01 03 00 00 00 02 C4 0B**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Send to the slave with address 01
Function code	1	03	Read register

Initial address	2	00 00	Start address is 0000
Read Number of registers	2	00 02	Read 2 registers, a total of 4 bytes
CRC code	2	C4 0B	The CRC calculated by the host, the low byte first(C4) and high byte behind(0B)

The message format returned by the product response: **01 03 04 Temperature (16 bits) Humidity (16 bits) CRC check code**

The following table is an example of returning a set of temperature and humidity data: **01 03 04 0A 67 15 BF 07 14**

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Data from address 01
Function code	1	03	Read the register
Number of bytes returned	1	04	Returned 2 registers, total 4 bytes
Register 0 high byte	1	0A	The content of address 0x00 (temperature high byte)
Register 0 low byte	1	67	The content of address 0x00 (temperature low byte)
Register 1 high byte	1	15	The content of address 0x00 (humidity high byte)
Register 1 low byte	1	BF	The content of address 0x00 (humidity low byte)
CRC code	2	07 14	The returned CRC calculated by the slave, the low byte first (07)

Data Conversion

- Temperature = (0x0A67) = 2663 → **26.63°C**
- Humidity = (0x15BF) = 5567 → **55.67 %RH**

2) Warm-up & Status Bits

➤ Test Steps For QModMaster

1. Read **Start Address 240, Count 1 (0x00F0)**.
 - Immediately after boot, **bit2 (value 4)** is set (WARMUP). After ~5 s it clears.
 - **bit0 (value 1)** means Sensor OK.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: **01 03 00 F0 00 01 84 39**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	03	Read holding registers
Initial address	2	00 F0	Status register
Read Number of registers	2	00 01	Read 1 register
CRC code	2	84 39	The CRC calculated by the host, the low byte first(84) and high byte behind(39)

Response Frame : **01 03 02 00 05 78 47 (~5Second) & 01 03 02 00 01 79 84**

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Device address
Function code	1	03	Read response
Number of bytes returned	1	02	2 bytes data
Data high byte	1	00	Status high byte
Data low byte	1	01	Status low byte

CRC code	2	79 84	CRC from slave
----------	---	-------	----------------

➤ **Bit Description**

- Bit2 = 1 → **WARMUP active (~5 sec after boot)**
- Bit0 = 1 → **Sensor OK**

3) Uptime counter

➤ **Test Steps For QModMaster**

1. Read **Start Address 228, Count 2** (0x00E4..0x00E5).
 - Combine: **seconds = (hi<<16) + lo**; it should increase each read.

➤ **Test Steps For Standard RS485 RTU Frame**

The format of the message sent by the host: **01 03 00 E4 00 02 84 3C**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	03	Read holding registers
Initial address	2	00 E4	Uptime register (High word)
Read Number of registers	2	00 02	Read 2 register
CRC code	2	84 3C	The CRC calculated by the host, the low byte first(84) and high byte behind(3C)

Response Frame : **01 03 04 00 00 01 F4 CRC**

• **Data Description**

Data	Meaning
First 2 bytes	High word
Next 2 bytes	Low word

➤ Conversion

Uptime (seconds) = (High << 16) + Low

4) Read Firmware version

➤ Test Steps For QModMaster

1. Read **Start Address 254, Count 1 (0x00FE)**.

- You should see **0x0100 = 256 decimal** for v1.0.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: **01 03 00 FE 00 01 E5 FA**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	03	Read holding registers
Initial address	2	00 FE	Firmware version register
Read Number of registers	2	00 01	Read 1 register
CRC code	2	E5 FA	The CRC calculated by the host, the low byte first(E5) and high byte behind(FA)

Response Frame : **01 03 02 01 00 B9 D4**

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Data from slave address 01
Function code	1	03	Read holding registers
Number of bytes returned	1	02	Returned 2 bytes (1 register)
Data high byte	1	01	Firmware version high byte

Data low byte	1	00	Firmware version low byte
CRC code	2	B9 D4	CRC calculated by slave, low byte first (B9), high byte next (D4)

➤ Interpretation

- Firmware Value = **(High Byte << 8) + Low Byte**
 - Example: **0x0100 = 256 → Version 1.0**
- Firmware version is stored as a **16-bit value**
 - Format:
 - High byte → Major version
 - Low byte → Minor version
- Example:
 - 0x0100 → Version 1.0
 - 0x0101 → Version 1.1

5) Real ID discovery

➤ Test Steps For QModMaster

- Read **Start Address 256, Count 1 (0x0100)**.
 - Returns the device's **actual** slave ID.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: **01 03 01 00 00 01 85 F6**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Recovery alias
Function code	1	03	Read holding registers
Initial address	2	01 00	Real slave ID register
Read Number of registers	2	00 01	Read 1 register
CRC code	2	85 F6	The CRC calculated by the host, the low

			byte first(85) and high byte behind (F6)
--	--	--	--

Response Frame : 01 03 02 00 07 F9 86

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Recovery alias
Function code	1	03	Read holding registers
Number of bytes returned	1	02	Data length
Data high byte	1	00	High byte
Data low byte	1	07	Actual slave ID = 7
CRC code	2	F9 86	CRC from slave

➤ Important Notes

- Device responds using **alias ID (01)** during recovery window
- Returned value is the **actual configured slave ID**
- After ~10 seconds, device will respond only to actual ID

6) Change measurement period (persisted)

➤ Test Steps For QModMaster

1. **Function:** *Write Single Register (0x06)*
2. **Start Address:** **514** (0x0202)
3. **Value:** e.g. **1000** (ms)
 - Observe new readings update every ~1 s. Power-cycle and re-read: value remains.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: 01 06 02 02 03 E8 29 0C. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	06	Write single register

Register address	2	02 02	Measurement period register
Data	2	03E8	value to update = 1000 ms
CRC code	2	29 0C	CRC (Low byte first)

Response Frame : 01 06 02 02 03 E8 29 0C

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Device address
Function code	1	06	Write confirmation
Register address	2	02 02	Measurement period register
Data	2	03 E8	Updated value = 1000 ms
CRC code	2	29 0C	CRC from slave

➤ Important Notes

- Value is in **milliseconds**
- Setting is **stored in memory (persistent)**

7) Apply offsets (persisted)

➤ Test Steps For QModMaster

- **Temp offset -0.50 °C:** Write **Start 512** (0x0200) with **-50**.
 - If QModMaster won't accept "-50", enter **65536 - 50 = 65486**.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: 01 06 02 00 FF CE 48 16. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	06	Write single register

Register address	2	02 00	Temperature offset register
Data	2	FF CE	-50 (scaled ×100)
CRC code	2	48 16	CRC (Low byte first)

Response Frame : 01 06 02 00 FF CE 48 16

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Device ID
Function code	1	06	Write single register response
Register address	2	02 00	Temperature offset register
Data high byte	1	FF	Offset high byte
Data low byte	1	CE	Offset low byte
CRC code	2	48 16	CRC calculated by slave, low byte first (48), high byte next (16)

➤ **Data Explanation**

- Offset = Value / 100
- Offset Value = **(High Byte << 8) + Low Byte** (Signed 16-bit)
 - Example : 0xFFCE = -50 → **-0.50°C**

➤ **Test Steps For QModMaster**

- **RH offset +1.25 %RH**: Write **Start 518** (0x0206) with **125**.
- Re-read T/RH (Step 1) and verify the shifts. Power-cycle → still there.

➤ **Test Steps For Standard RS485 RTU Frame**

The format of the message sent by the host: 01 06 02 06 00 7D A8 52. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	06	Write single register
Register address	2	02 06	Humidity offset register
Data	2	00 7D	+1.25 %RH

CRC code	2	A8 52	CRC (Low byte first)
----------	---	-------	----------------------

Response Frame : 01 06 02 06 00 7D A8 52

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Device ID
Function code	1	06	Write single register response
Register address	2	02 06	Humidity offset register
Data high byte	1	00	Offset high byte
Data low byte	1	7D	Offset low byte
CRC code	2	A8 52	CRC calculated by slave, low byte first (88), high byte next (6F)

➤ Data Explanation

- Offset = Value / 100
- Offset Value = **(High Byte << 8) + Low Byte**
 - Eg. 0x007D = 125 → +1.25 %RH

8) Baud-rate change

➤ Test Steps For QModMaster

1. Write **Start 528** (0x0210) with a **baud code**:
 - 0=9600, 1=19200, 2=38400, 3=57600, 4=115200.
2. You'll get a normal reply at the **old** baud; then the device switches.
3. In QModMaster, **Options** → **Port**: change to the new baud; keep the same Slave Addr.
4. Re-read T/RH to confirm.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: **01 06 02 10 00 04 88 74**. The following table is an introduction to the function codes:

Command Frame (Example: Set 115200)

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	06	Write single register
Register address	2	02 10	Baud rate register
Data	2	00 04	Set baud rate = 115200
CRC code	2	88 74	CRC (Low byte first)

Response Frame : **01 06 02 10 00 04 88 74**

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Device address
Function code	1	06	Write confirmation
Register address	2	02 10	Baud-rate register
Data	2	00 04	Updated value
CRC code	2	88 74	CRC from slave

➤ Important Notes

- Device responds at **old baud rate**, then switches
- Master must change baud rate accordingly

9) Change slave ID

➤ Test Steps For QModMaster

1. In the main window of QmodMaster, change Function Code to Write Single Register (0x06) (sometimes shown as *Preset Single Register*).
2. Set Start Address to 257 (Dec).
3. Number of Registers: 1.
4. Data Format: Dec (so you can enter the ID as a decimal number).
5. In the value grid (first cell), type the new ID (valid range 1...247).
6. Click Read/Write button, here your slave ID change Successfully.
7. Switch to the new ID and confirm.

- After the brief reboot, change Slave Address in QModMaster to the new ID.
- To verify, set Function Code to Read Holding Registers (0x03).
- Start Address = 1, Number of Registers = 2.

➤ Test Steps For Standard RS485 RTU Frame

The format of the message sent by the host: **01 06 01 01 00 0A 59 F1**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Current slave ID
Function code	1	06	Write single register
Register address	2	0101	Slave ID register
Data (New ID)	2	000A	New slave ID = 10
CRC code	2	59 F1	CRC (Low byte first: 59, High byte: F1)

Response Frame : **01 06 01 01 00 0A 59 F1**

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Device address
Function code	1	06	Write confirmation
Register address	2	01 01	Slave ID register
Data	2	00 0A	Updated value
CRC code	2	59 F1	CRC from slave

➤ Important Notes

- Valid Slave ID range: **1 To 247**
- After changing ID, communication must use **new ID**
- Device may reboot after update

10) How to Recover the Slave ID

➤ Test Steps For QModMaster

- If you don't know the current ID, Turn off for 2 second and On it, for the next ~10 s, talk to Slave = 1 (your code's recovery alias).

1. Immediately in QModMaster:
 - Set Slave Addr = 1 (the recovery alias).
 - Function Code: 0x03 Read Holding Registers.
 - Start Address = 256, Number of Registers = 1.
 - Data format = Dec
2. Hit Read.
 - If the wiring and baud rate are correct, register 0x0100 (decimal 256) will return the actual slave ID.
 - Example: If it returns 7, then your device will answer only to ID = 7 after the 10-second recovery period ends.
3. After that, switch QModMaster's Slave Addr to that returned ID for normal operation.

➤ **Test Steps For Standard RS485 RTU Frame**

The format of the message sent by the host: **01 03 01 00 00 01 85 F6**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Recovery alias
Function code	1	03	Read holding registers
Initial address	2	01 00	Real slave ID register
Read Number of registers	2	00 01	Read 1 register
CRC code	2	85 F6	The CRC calculated by the host, the low byte first(85) and high byte behind (F6)

Response Frame : **01 03 02 00 07 F9 86**

Slave response	Number of bytes	Message returned	Remarks
Slave address	1	01	Recovery alias
Function code	1	03	Read holding registers
Number of bytes returned	1	02	Data length
Data high byte	1	00	High byte
Data low byte	1	07	Actual slave ID = 7

CRC code	2	F9 86	CRC from slave
----------	---	-------	----------------

➤ **Important Notes**

- Device responds using **alias ID (01)** during recovery window
- Returned value is the **actual configured slave ID**
- After ~10 seconds, device will respond only to actual ID

11) Broadcast write (Slave ID 0, No Reply)

➤ **Test Steps For QModMaster**

1. Set **Slave Addr = 0** (broadcast).
2. **Function:** *Write Multiple Registers (0x10)*
3. **Start Address: 514** (0x0202), **Number of Registers: 1**.
4. In the data cell type **2000** (ms) and send.
 - There is **no reply** by design; QModMaster may show a timeout—**that's expected**.
 - If you have multiple nodes, all should adopt the new period.

➤ **Test Steps For Standard RS485 RTU Frame**

The format of the message sent by the host: **00 10 02 02 00 01 02 07 D0 8A 4E**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	00	Broadcast address
Function code	1	10	Write multiple registers
Register address	2	02 02	Measurement period register
Number of registers	2	00 01	One register
Byte count	1	02	Two bytes data
Data	2	07 D0	2000 ms
CRC code	2	8A 4E	CRC (Low byte first)

12) Read diagnostics

➤ Test Steps For QModMaster

- **CRC errors:** Start **224** (0x00E0) = 0 =>OK
- **Exceptions:** Start **225** (0x00E1) = 0 =>OK
 - To bump **exceptions**, try writing to a read-only address (e.g., Write 0x06 to **Start 1**)—the firmware will reject it and increment the counter.

➤ Test Steps For Standard RS485 RTU Frame

- For **CRC Errors** :

The format of the message sent by the host: **01 03 00 E0 00 01 85 FC**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	03	Read holding registers
Initial address	2	00 E0	CRC error counter register
Read Number of registers	2	00 01	Read 1 register
CRC code	2	85 FC	CRC (Low byte first: 85)

Response Frame : **01 03 02 00 00 B8 44**

Slave response	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	03	Read response
Number of bytes returned	1	02	Data length
Data high byte	1	00	High byte
Data low byte	1	00	CRC error count
CRC code	2	B8 44	CRC from slave

- For **Exceptions** :

The format of the message sent by the host: **01 03 00 E1 00 01 D4 3C**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID

Function code	1	03	Read holding registers
Initial address	2	00 E1	Exception counter register
Read Number of registers	2	0001	Read 1 register
CRC code	2	D4 3C	CRC (Low byte first: D4)

Response Frame : 01 03 02 00 00 B8 44

Slave response	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	03	Read response
Number of bytes returned	1	02	Data length
Data high byte	1	00	High byte
Data low byte	1	00	Exception count
CRC code	2	B8 44	CRC from slave

➤ Notes

- Value = 0 → **No errors (Normal)**
- Value increases when:
 - Noise on RS485 line
 - Wrong baud rate
 - Incorrect wiring (A/B reversed)
 - Send invalid command
 - Example: Write (0x06) to read-only register (0x0000)
 - Device will respond with **Modbus Exception**
 - Exception counter (**0x00E1**) will increment
 - CRC counter increasing => Noise / wrong baud / cable issue
 - Exception counter increasing => Invalid commands from master
 - Both increasing => Communication mismatch

13) Input Registers Mirror (0x04)

➤ Test Steps For QModMaster

- Repeat Step 1 but select **Function 0x04 (Read Input Registers)** and the same addresses.
 - You should get identical values (mirror of the read-only map).

➤ **Test Steps For Standard RS485 RTU Frame**

The format of the message sent by the host: **01 04 00 00 00 02 71 CB**. The following table is an introduction to the function codes:

Send by Host	Number of bytes	Message to send	Remarks
Slave address	1	01	Device ID
Function code	1	04	Read input registers
Initial address	2	00 00	Slave ID register
Read Number of registers	2	00 02	Start address
CRC code	2	71 CB	CRC (Low byte first:71, High byte: CB)

Response :- Same data format as Function 0x03.

Note :- The returned temperature and humidity values are identical to the corresponding mirrored holding-register values.

14) Optional: Device Identification (MEI 0x2B/0x0E)

➤ **Test Steps For QModMaster / Modbus Poll**

- QModMaster usually doesn't send 0x2B. If you have **Modbus Poll** or **modpoll**, you can query the MEI "Device ID" to read strings 7semi, **Industrial RS485 Temperature & Humidity Sensor Probe**, and version 1.0

➤ **Test Steps For Standard RS485 RTU Frame**

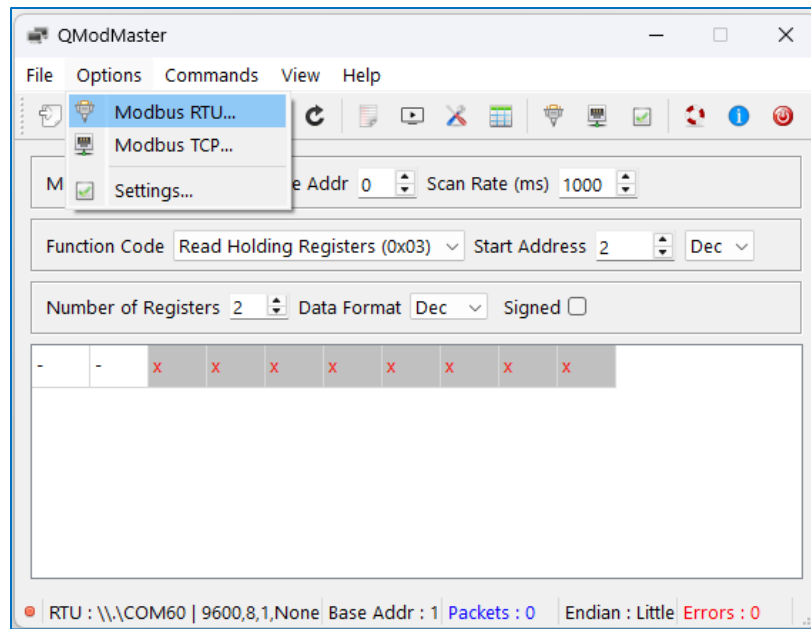
The format of the message sent by the host: **01 2B 0E 01 00 CRC**.

Field	Value	Description
Function	2B	Encapsulated interface
MEI Type	0E	Device ID
Read Code	01	Basic device info

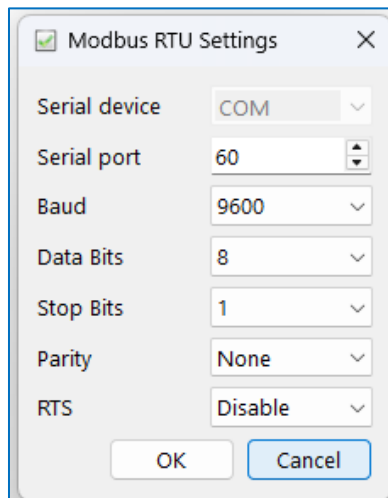
4.3 Modbus output

Follow the steps below for the software setup:-

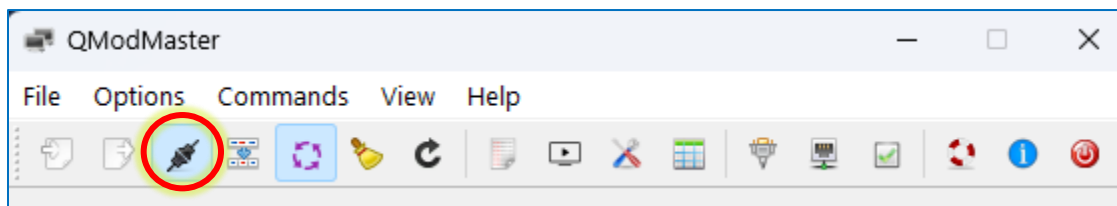
1. Open the software (qModMaster-Win64-exe-0.5.3-beta)
2. Go to *Options* → *Modbus RTU*



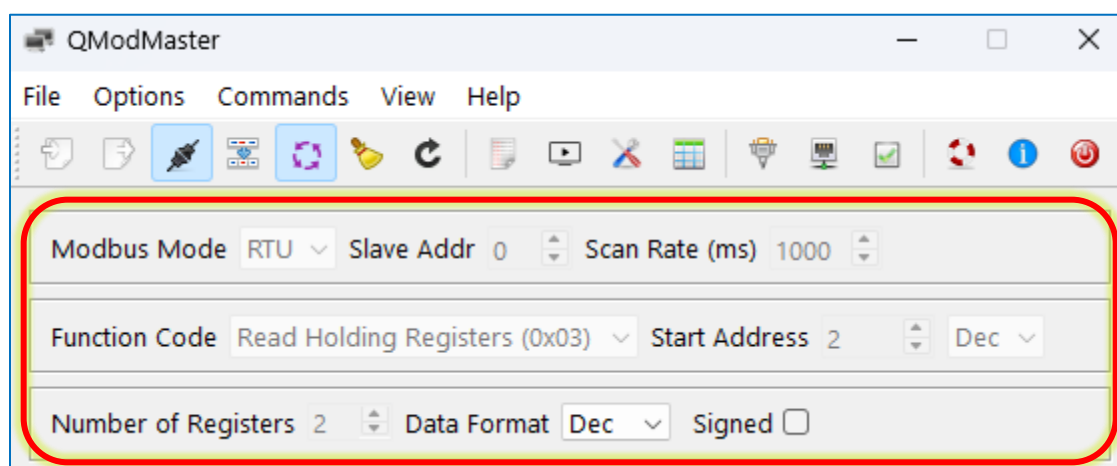
3. Check your COM port in device manager and select the other settings as per the image below. Click *OK*



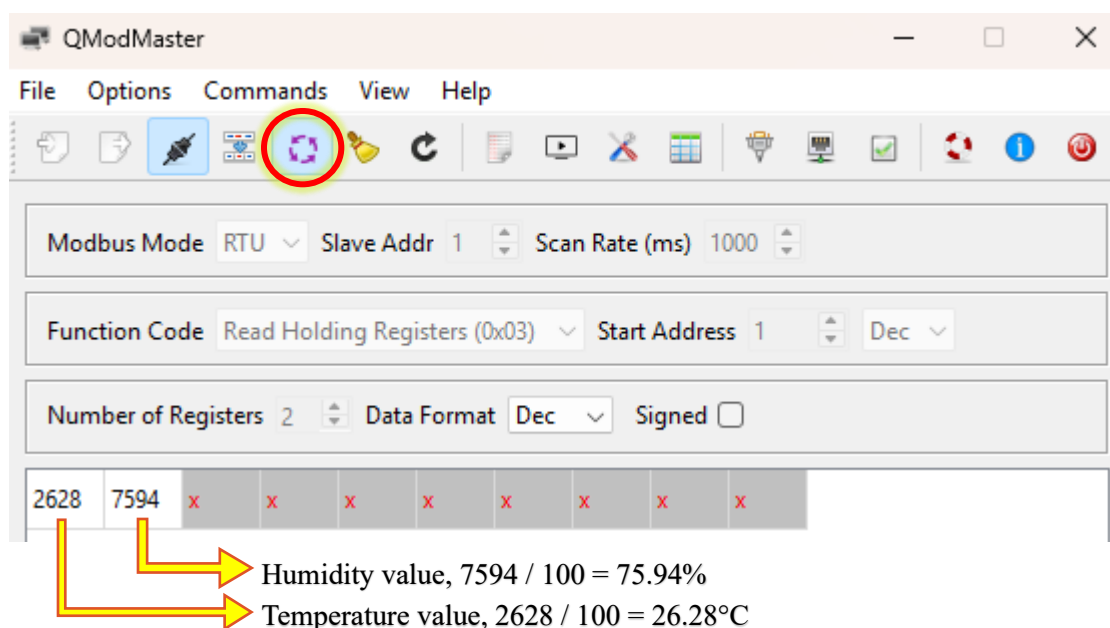
4. Connect your system to the correct COM port and click on this icon.



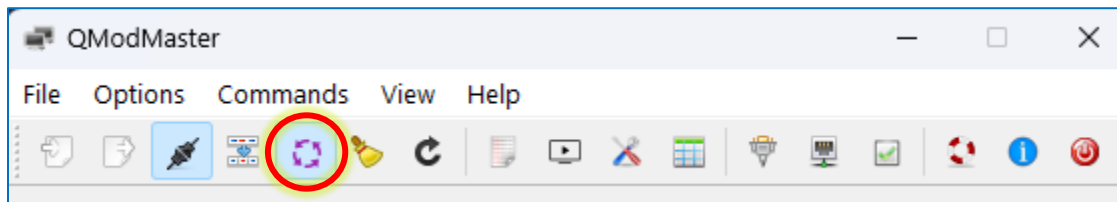
5. Also, make sure the following parameters are set properly.



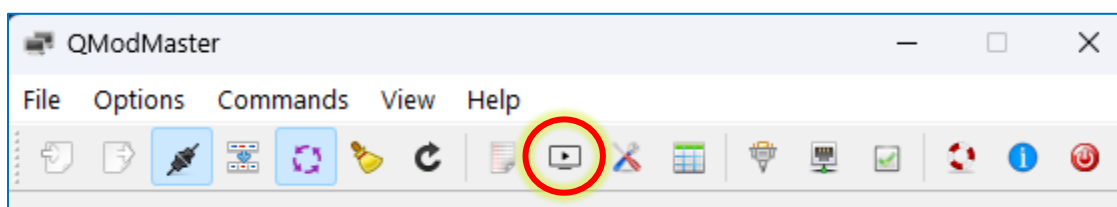
6. Now to check the data, click on this icon.



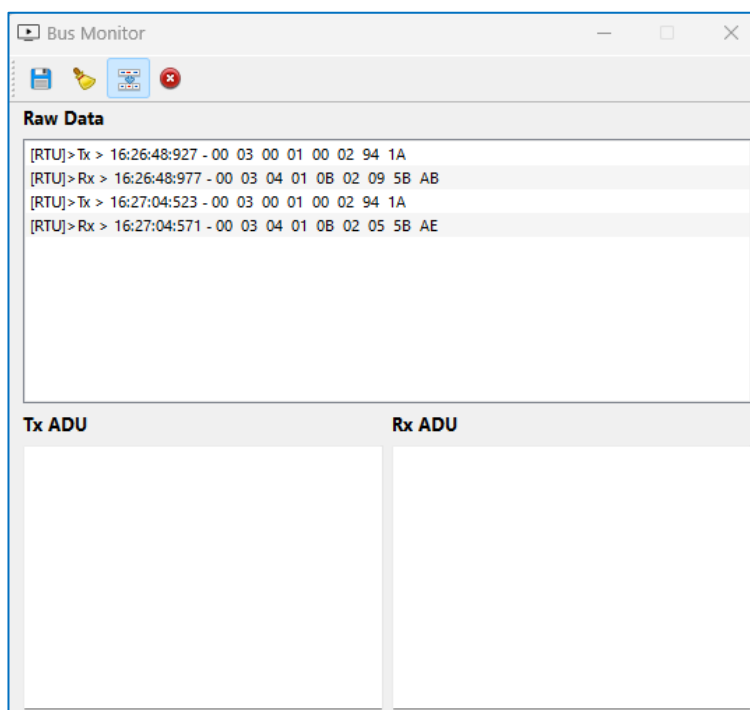
7. If you want to display the values continuously based upon the scan rate then click on this icon.

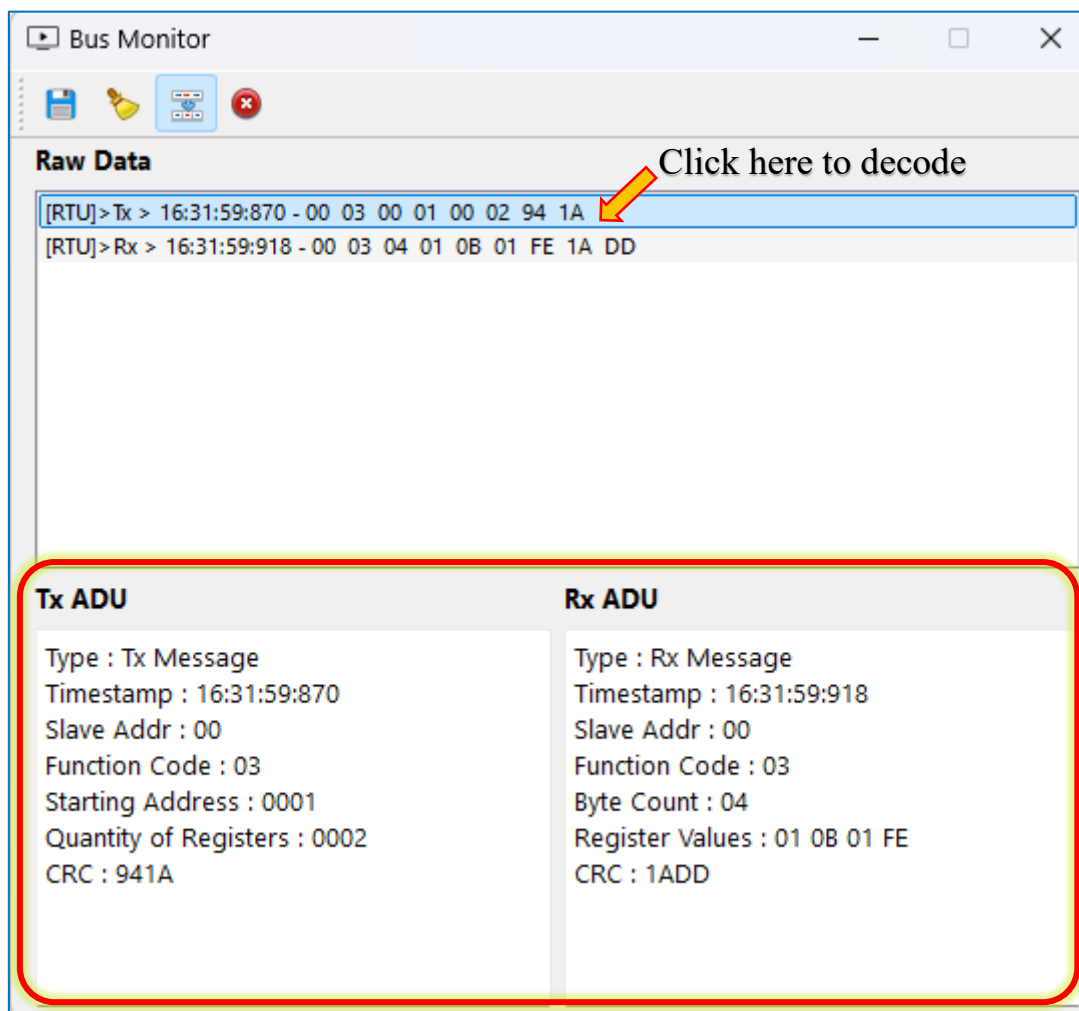


If you want to see the values in more detail you can click on this icon.



A new window will open to display the readings. To display data, follow step 6.

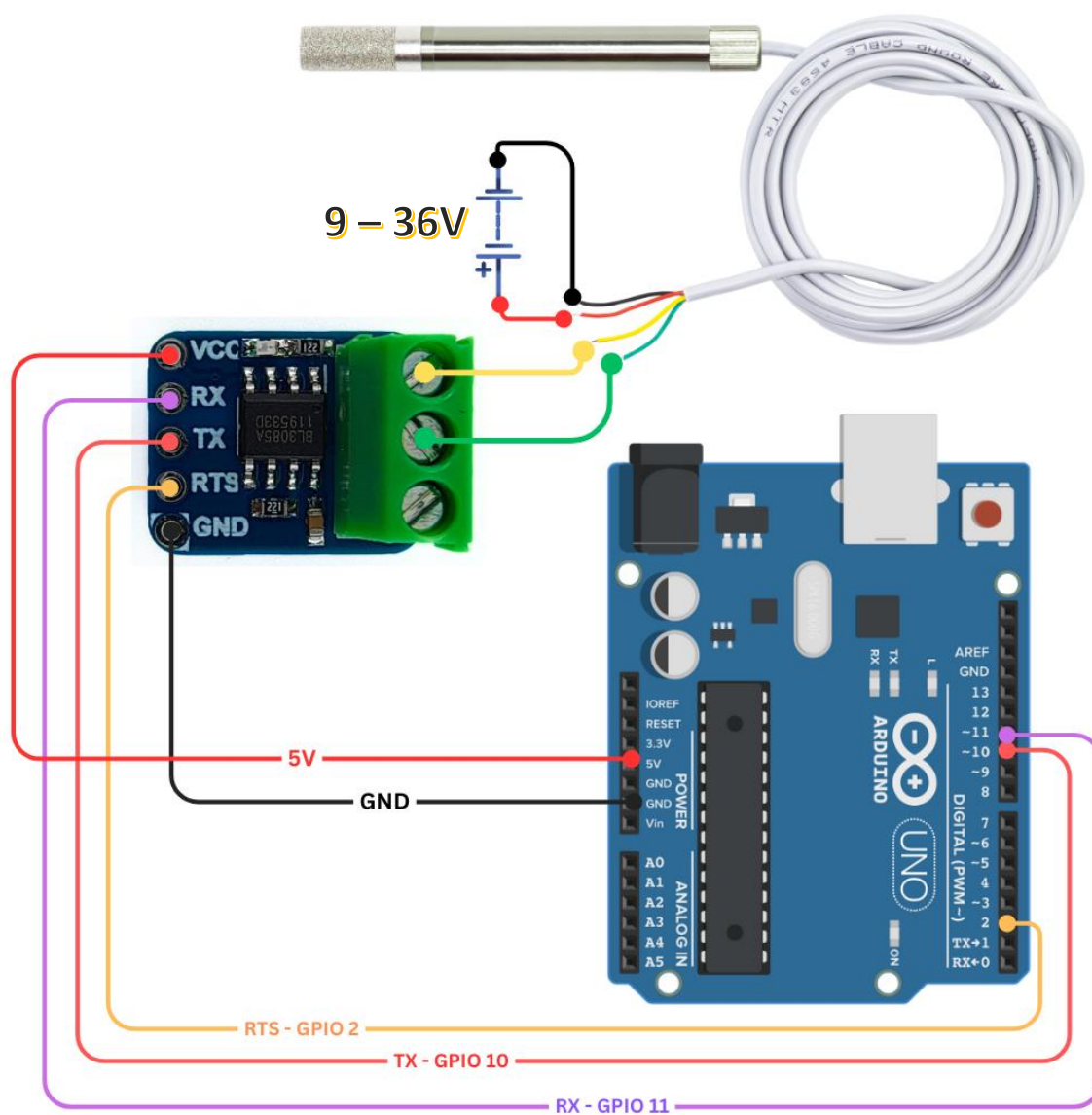




4.4 How to open the probe?



4.5 Connection and sample code for Arduino



4.6 How to Install 7Semi Arduino Library

- **Install the Arduino IDE** on your computer.
- Once installed, **open the Arduino IDE**.
- Go to **Library Manager** and search for “**7Semi-RS485-Temperature-Humidity-Probe-Arduino-Library**.”
- **Install** the library from the search results.
- After installation, navigate to **File → Examples → Basic → Example Usage**, and open the “**Basic.ino**” example.
- **Upload** the example code to your Arduino board.
- Set the **baud rate to 115200** in the Serial Monitor.
- You should now **see the output** displayed on the Serial Monitor.

Sample code link for manual download:- [“RS485-sensor-probe”](#)

Sample output image:-

Temperature: 27.70 °C	Humidity: 58.30 %RH
Temperature: 27.70 °C	Humidity: 58.30 %RH
Temperature: 27.70 °C	Humidity: 58.30 %RH
Temperature: 27.70 °C	Humidity: 58.30 %RH
Temperature: 27.70 °C	Humidity: 58.30 %RH
Temperature: 27.70 °C	Humidity: 58.30 %RH
Temperature: 27.70 °C	Humidity: 58.40 %RH
Temperature: 27.70 °C	Humidity: 60.80 %RH
Temperature: 27.80 °C	Humidity: 65.80 %RH
Temperature: 27.90 °C	Humidity: 70.00 %RH
Temperature: 28.00 °C	Humidity: 72.80 %RH
Temperature: 28.00 °C	Humidity: 72.60 %RH
Temperature: 28.10 °C	Humidity: 70.10 %RH
Temperature: 28.20 °C	Humidity: 67.50 %RH
Temperature: 28.20 °C	Humidity: 65.40 %RH
Temperature: 28.20 °C	Humidity: 63.80 %RH
Temperature: 28.30 °C	Humidity: 62.60 %RH
Temperature: 28.40 °C	Humidity: 61.70 %RH

5. Mechanical specification

