7Semi

RS-485 Temperature, Humidity, Pressure and Gas

sensor Probe

Version 1.0

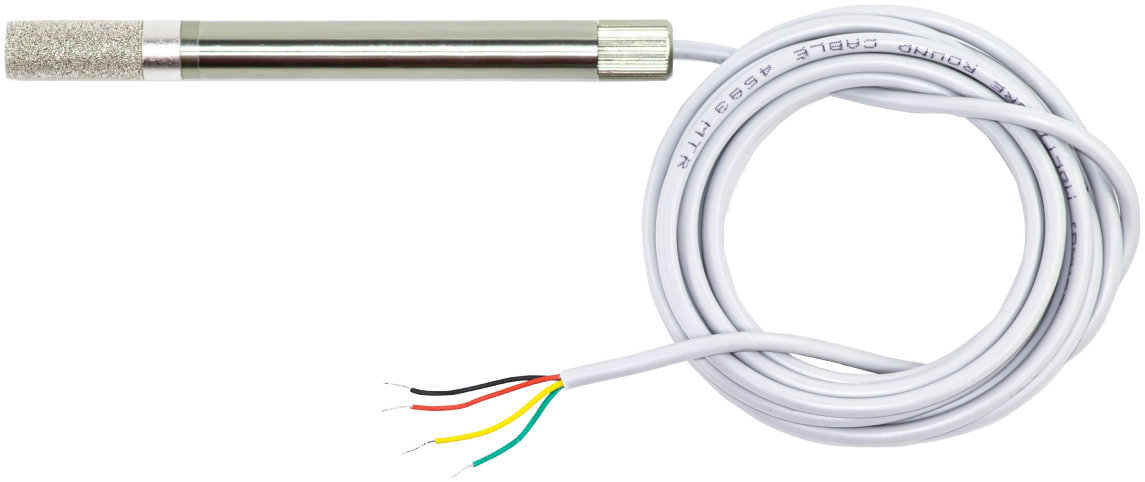
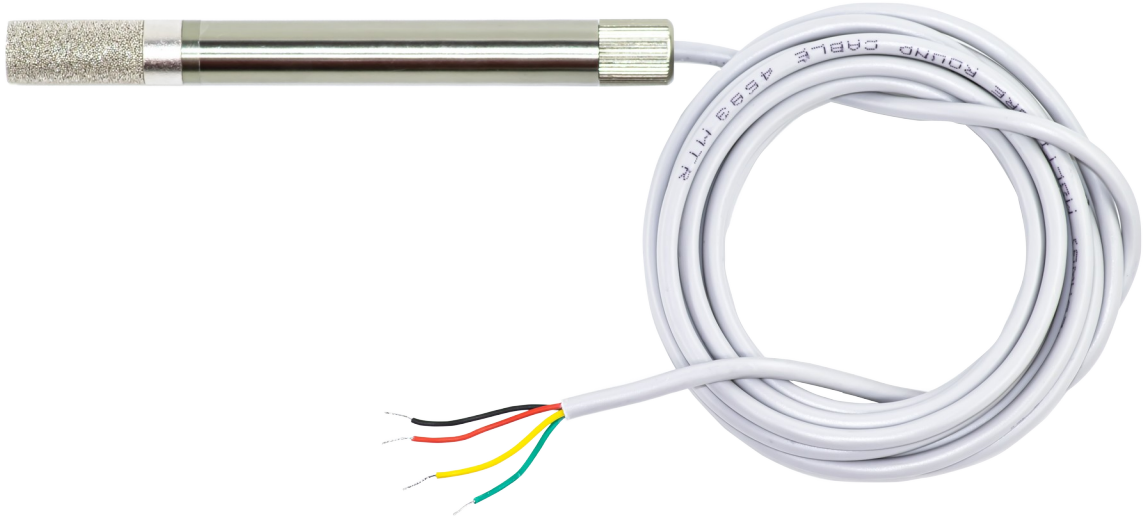


Table of Contents

1.Introduction



The **7Semi ENV-485 Probe** is a compact, multi-parameter environmental sensing module designed to monitor four essential air quality and climate parameters: temperature, relative humidity, barometric pressure, and a gas resistance. Engineered for continuous operation in both industrial and indoor environments, it delivers accurate, stable measurements over a robust **RS-485 Modbus RTU** interface.

All measurement processing, scaling, compensation, and communication management are performed internally, ensuring consistent output and reliable responses to any Modbus master. With its long-life construction and digital interface, the ENV-485 Probe is ideal for HVAC systems, building automation, smart agriculture, data centers, and general environmental monitoring applications.

4-in-1 environmental sensing

- Temperature, relative humidity, barometric pressure and gas resistance

RS-485 Modbus-RTU slave

- Standard Modbus RTU over RS-485 (2-wire, half-duplex)

Configurable communication

- Slave ID: 1…247 (configurable via Modbus)
- Baudrate code: 0…4 → 9600, 19200, 38400, 57600, 115200

Configurable behaviour

- Adjustable measurement period
- Temperature & humidity calibration offsets
- Heater configuration

Compact RS-485 field device

- Standard Modbus RTU over RS-485 (2-wire, half-duplex)
- Designed for easy integration into panels, ducts, or wall enclosures

Temperature (T)

- Measures ambient air temperature in **°C**.
- Important for correcting other sensor readings (humidity, gas).

Humidity (RH)

- Measures relative humidity as **%** (0–100%).
- Indicates moisture content in the air.
- Useful for environmental monitoring and comfort assessment.

Pressure (P)

- Measures atmospheric pressure in **Pa** (can convert to hPa or kPa).
- Can be used to estimate **altitude** or monitor weather changes.

Gas Resistance (Rs)

- Measures resistance of the gas sensor in **Ω**.
- Lower Rs → higher VOC levels → poorer air quality.
- Requires temperature and humidity correction for accurate air quality estimation.
- **Rs is a relative measure** of VOC presence, not specific gases or ppm.
- It helps track air quality trends, with lower resistance correlating to higher VOC levels.
- **Environmental factors** like temperature and humidity affect Rs readings.

2.Technical specification

The **7Semi RS-485 Temperature, Humidity, Pressure & Gas Sensor Probe** is an industrial-grade Modbus RTU device designed for high-accuracy environmental monitoring and seamless integration into automation systems. It simultaneously measures **temperature**, **relative humidity**, **barometric pressure**, and **gas resistance** with a configurable

Applications

Building management systems (BMS)

HVAC monitoring and control

Clean room and laboratory monitoring

Environmental monitoring in industrial cabinets

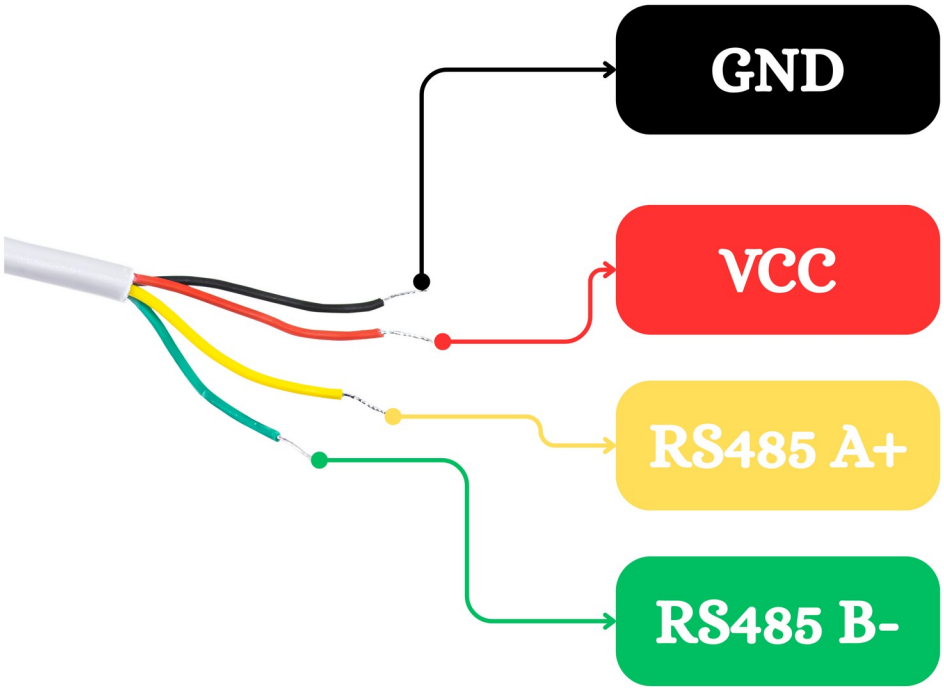
Data logging and remote monitoring via Modbus

2.1 General specifications

- **Temperature Measurement Range:** -40 ~ 85°C
- **Humidity Measurement Range:** 0 ~ 100% RH
- **Temperature Accuracy:** ±0.5°C (0-65°C)
- **Humidity Accuracy:** ±3% RH (25°C)
- **Barometric Pressure:** 300 – 1100hPa
- **Sampling Period:** Configurable 0.25 – 10 s (default 1 s)
- **Power Supply Voltage:** 9 ~ 36V (DC)
- **Communication Interface:** RS-485
- **Product Size:** 200mm(L) × 15.7mm(D)
- **Output Format:** RS-485 digital signal
- **Communication Protocol:** Standard MODBUS RTU protocol (8N1 format)
- **Supported Baud Rates:** 9600 (default), 19200, 38400, 57600, 115200
- **Display Resolution:** Temperature: 0.1°C; Humidity: 0.1% RH

3. Hardware setup

3.1 Pinouts

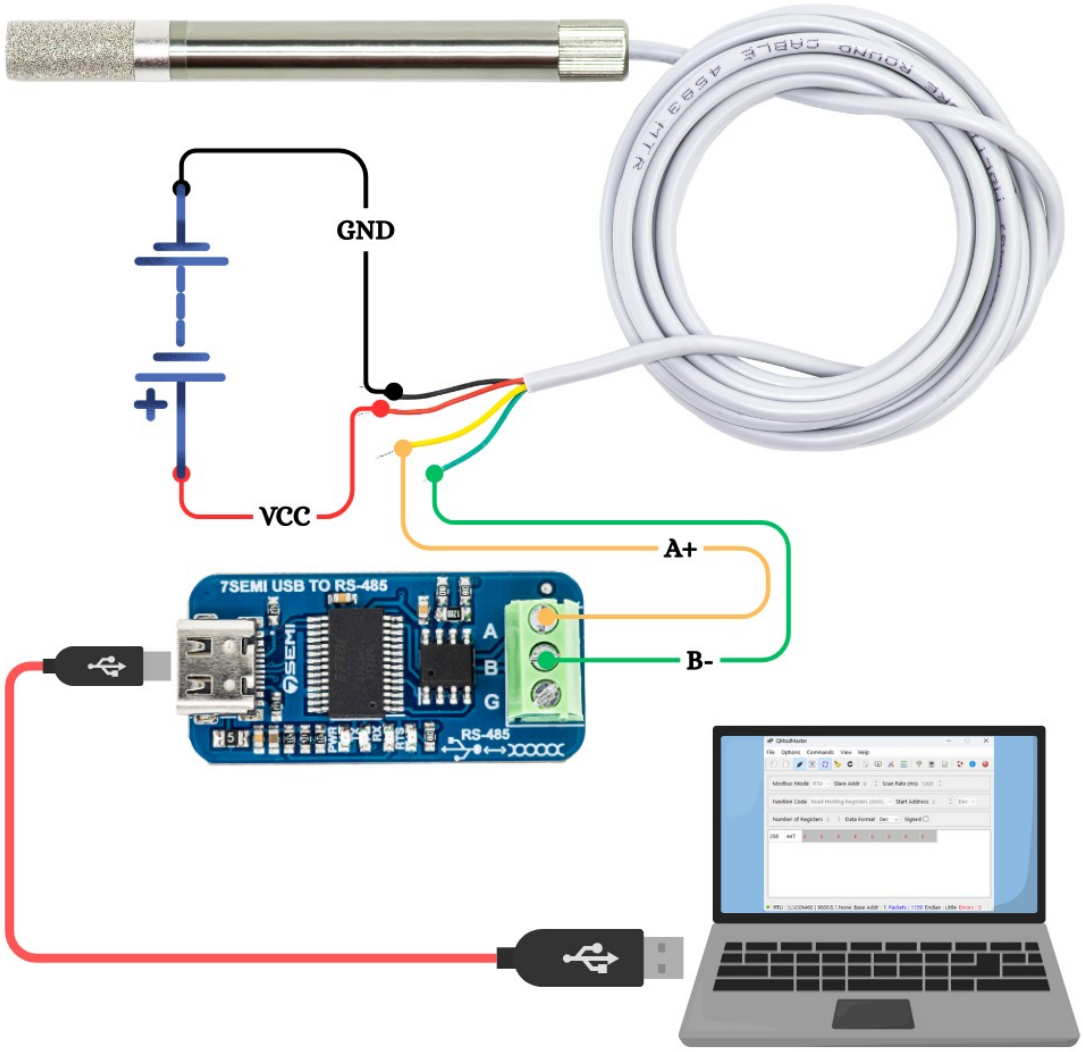


Wire Color	Function
Yellow	RS485 A+
Green	RS485 B−
Red	VCC
Black	GND

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 white 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 RS-485 Temperature, Humidity, Pressure and Gas Sensor Probe** communicates using the **Modbus RTU** protocol over an RS-485 electrical interface. Modbus RTU is a widely adopted, robust, and efficient industrial communication standard used in systems like PLCs, HMIs, SCADA, data loggers, and building automation networks.

This probe operates strictly as a **Modbus RTU slave device**, meaning it does not initiate communication on its own. Instead, it waits for commands from a Modbus master and responds with sensor readings or configuration data as requested. Communication is performed using compact binary frames with **CRC16 (Cyclic Redundancy Check)** to ensure high-integrity data transfer, even in electrically noisy industrial environments.

All key operational parameters such as **slave ID**, **baud rate**, **measurement interval**, and **calibration offsets** are accessible through Modbus registers. This communication structure makes the probe highly compatible with industrial controllers, allowing seamless integration into BMS, HVAC controllers, industrial monitoring cabinets, and real-time data acquisition systems.

Request Frame:

SA	FC	RA	RC	BC	DF	CRC
1 Bytes	1 Bytes	2 Bytes	2 Bytes	1 Byte	N Bytes	2 Bytes

Request example:

```
01 03 00 00 00 06 C5 C8
```

Responses Frame:

SA	FC	BC	DF	CRC
1 Bytes	1 Bytes	1 Bytes	N Bytes	2 Bytes

Response (example):

```
01 03 0C 09 DE 1A F4 00 01 00 02 00 03 39 85
```

SA – Slave Address
Identifies the Modbus device being addressed (range 1–247).

FC – Function Code
Indicates the requested operation (e.g., read registers, write register).

RA – Register Address
The starting register address, sent as high byte first then low byte.

RC – Register Count
Number of registers to read or write.

BC – Byte Count
Used only for write-multiple-register (0x10). Indicates how many data bytes follow.

DF – Data Field
Contains the actual register values (high byte first). Only present for write operations.

CRC – CRC-16 checksum
Used for error checking. Sent as low byte first, then high byte.

Modbus Timing Requirement

The device uses **IDLE-line detection** to determine the end of a Modbus RTU frame.
A **minimum silent interval of 50 ms** is required on the RS-485 bus to mark the end of a message.

This timing is important for correct frame reception and must be ensured by the Modbus master.

4.1 Supported Modbus Function Code`

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, humidity, pressure, gas resistance), Read Input Registers, Write Single Register and Write Multiple Registers

4.2 CRC-16 Calculation

Modbus RTU frames use a **16-bit CRC (Cyclic Redundancy Check)** for error detection. This CRC ensures reliable communication even in electrically noisy environments.

CRC-16 Modbus Specification

Parameter	Value
Polynomial	0xA001 (reversed form of 0x8005)
Initial Value	0xFFFF
Input / Output Reflected	Yes
Final XOR	None
Byte Order (Transmission)	**Low byte first**, then high byte
Length	16 bits

- C Example

```
uint16_t Modbus_CRC16(const uint8_t *data, uint16_t length)
{
   uint16_t crc = 0xFFFF;

   for (uint16_t i = 0; i < length; i++) {
      crc ^= data[i];

      for (uint8_t j = 0; j < 8; j++) {
         if (crc & 1)
            crc = (crc >> 1) ^ 0xA001;
         else
            crc >>= 1;
      }
   }

   return crc;
}
```

5 Register Map

Address	Name	Type	Unit / Scaling
0x00	Temperature	int16	°C × 100
0x01	Humidity RH	uint16	%RH × 100
0x02	Pressure (high word)	int16	Pa (high 16 bits of 32-bit value)
0x03	Pressure (low word)	int16	Pa (low 16 bits of 32-bit value)
0x04	Gas resistance (Hi)	uint16	Ω (high 16 bits of 32-bit value)
0x05	Gas resistance (Lo)	uint16	Ω (low 16 bits of 32-bit value)
0xF0	Sensor status flags	uint16	Status flags
0x11	Firmware version	uint16	
0x20	Temperature offset	int16	°C × 100
0x21	Humidity offset	int16	%RH × 1000
0x30	Heater temperature	uint16	°C
0x31	Heater duration	uint16	millisecond
0x32	Heater enable	uint16	1 or 0
0x33	Measurement period	uint16	millisecond
0x24	Baud rate code	uint16	0 - 4

Temperature Register 0x0000
Returns the measured temperature in hundredths of a degree Celsius.

- Example: a value of **2534** represents **25.34 °C**.
- Stored as a **signed 16-bit value**.
- Valid physical range: approximately −40.00 °C to +85.00 °C.

Humidity Register 0x0001
Returns the relative humidity in hundredths of a percent.

- Example: a value of **6543** represents **65.43 %RH**.
- Stored as an **unsigned 16-bit value**.
- Valid range: 0.00 to 100.00 %RH.

Pressure (High Word) Register 0x0002
Pressure (Low Word) Register 0x0003
Returns the measured pressure as a 32-bit unsigned value in Pascals (Pa), split across two registers.

- Combine registers as:
  Pressure = (HighWord << 16) | LowWord
- Example: High = 0x0001, Low = 0x86A0 → Pressure = 100000 Pa.
- Stored as **unsigned 16-bit words** forming a 32-bit reading.

Gas Resistance (High Word) Register 0x0004

Gas Resistance (Low Word) Register 0x0005

Returns the measured gas resistance from the sensor as a 32-bit unsigned value in ohms, split across two registers.

- Combine registers as:
  GasResistance = (HighWord << 16) | LowWord
- Example: High = 0x0002, Low = 0x3A98 → Resistance = 150,000 Ω.
- Stored as **unsigned 16-bit words** forming a 32-bit reading.

Gas Resistance 0x0009
The gas resistance is reported as a **32-bit unsigned value** in **ohms**.
The upper 16 bits are stored at address **0x0004**, and the lower 16 bits at **0x0005**.
To obtain the full gas resistance value, combine both words using:

GasResistance = (HighWord << 16) | LowWord

Temperature Offset 0x0020
Stores a temperature correction value in ×100 format.
Positive values increase the reported temperature, and negative values decrease it.
This register is used to apply calibration or system-level correction.

Example:
• If you write **+50**, the sensor reading increases by **+0.50 °C**.
• If you write **–125**, the sensor reading decreases by **–1.25 °C**.
• If raw temperature = **25.34 °C** and offset = **–50**, final output = **24.84 °C**.

Humidity Offset 0x0021
Stores a humidity correction value in ×100 format.
Positive values increase the reported humidity, and negative values decrease it.
Used for humidity calibration or system-level correction.

Example:
• Writing **+120** increases the humidity reading by **+1.20 %RH**.
• Writing **–75** decreases the humidity reading by **–0.75 %RH**.
• If raw humidity = **62.50 %RH** and offset = **–125**, final output = **61.25 %RH**.

Baud Rate Code 0x0024
Defines the communication speed used on the RS-485 Modbus interface.
This register accepts a numeric code that sets the UART baud rate. The default baud rate 9600.

Supported values:
• **0 → 9600 bps**
• **1 → 19200 bps**
• **2 → 38400 bps**
• **3 → 57600 bps**
• **4 → 115200 bps**

Heater Temperature 0x0030
Defines the target heater **temperature** (°C) used by the internal gas-sensing element.
This value determines how hot the gas sensor heater operates during a measurement cycle.

Sets or reports the heater ON time in milliseconds for the gas-sensor heating cycle.
Valid configuration range: **100 to 400 ms**.

The heater draws a **significantly higher current** during operation in some cases **up to 200 mA**.

Heater Duration 0x0031
Sets or reports the heater **ON time** in milliseconds for the gas-sensor heating cycle.
This value controls how long the heater remains active during each measurement.

Valid configuration range: **100 to 400 ms**

Heater Enable 0x0032
Controls the internal heater state.
A value of 0 corresponds to heater disabled, and 1 value corresponds to heater enabled.

Measurement Period 0x0033
Defines the measurement interval in milliseconds.
This is the time between consecutive sensor sampling cycles.
Typical supported range is from 250 ms to 10,000 ms.

Sensor Status 0x00F0
Contains bit-mapped status information about the sensor and system state.
Individual bits indicate conditions such as sensor OK, warm-up active, communication issues, or internal error flags, depending on the firmware definition.

Soft Reset 0x00F1
Used to trigger a controlled software reset of the device.
Writing the defined reset key value causes the sensor to restart and reinitialize its internal state.

Find Slave ID 0x0100
When read, this register returns the currently configured Modbus slave address stored in non-volatile memory.
Useful when the device ID is unknown and needs to be rediscovered by the master.

Change Slave ID 0x0101
When written with a valid value in the range 1 to 247, this register updates the Modbus slave address of the device.
The new ID is stored in non-volatile memory and will be used after the device completes its internal update and restart sequence.

6 Testing Steps For QModMaster

1) Basic sensor read

1. **Slave Address :** Keep original slave address(1…247)
2. **Function Code:** Read Holding Registers (0x03)
3. **Start Address: 0x00**
4. **Number of Registers: 6**
5. **Temperature is signed uint16_t if temperature is negative its show high temperature, convert in to signed format.**
    - Expect T and RH as integers ×100. E.g., `2579` → 25.79 °C; `6665` → 66.65 %RH.
    - Also read: Gas resistance and pressure Gas resistance high word<<8 | Gas resistance low word similarly for pressure.

2) Status bits

Sensor Status (0x00F0) Bits

- Bit 0 → Temperature/Humidity sensor OK
- Bit 1 → I2C communication error
- Bit 2 → Warm-up active
- Bit 3 → Gas sensor/Heater OK
- Bit 4 → Conditioning active
- Bits 5–15 → Reserved

3) Firmware version

1. Read **Start Address0x11**
    - You should see **0x0100 = 256 decimal** for v**1.0**.

4) Real ID discovery (no matter which ID you queried)

1. Read **Start Address** 0x0100.
    - Returns the device's **actual** slave ID.

6) Change measurement period (persisted)

1. **Function:** *Write Single Register (0x06)*
2. **Start Address: 0x33**
3. **Value:** e.g. **1000** (ms)
    - Observe new readings update every ~1 s. Power-cycle and re-read: value remains.

7) Apply Temperature & Humidity Offsets

- **Temperature Offset (0x0020)**
    - Format: value ×100
    - Example: **–0.50°C → write –50**
    - If negatives not allowed in QModMaster:
        - Convert: **65536 – 50 = 65486**, write **65486**
- **Humidity Offset (0x0021)**
    - Format: value ×100
    - Example: **+1.25 %RH → write 125**

8) Baud-rate change with safe switch

- Write to Register 0x0024
- Use Write Single Register (0x06)
- Baud codes:
    - **0 → 9600**
    - **1 → 19200**
    - **2 → 38400**
    - **3 → 57600**
    - **4 → 115200**

9) Change slave ID

1. Open QModMaster and select **Function Code 0x06 (Write Single Register)**.
2. Set **Register Address = 0x0101**.
3. Set **Number of Registers = 1**.
4. Select **Data Format = Decimal**.
5. Enter the **new slave ID (valid range: 1–247)** in the value field.
6. Click **Read/Write** to apply the new ID.
7. The device reboots briefly and the ID is updated.
8. Change the **Slave Address** in QModMaster to the new ID.
9. Confirm by reading registers:
10. Function Code: **0x03 (Read Holding Registers)**
11. Start Address: **0x0001**

12. Number of Registers: **6**

10) How to actually recover the ID

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 = 0x0101, Number of Registers = 1.
2. Hit Read.
    - If wiring and baud rate are correct, the register at address 1 will return the *real* slave ID stored in flash.
    - 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.

11) Broadcast write (ID 0, no reply)

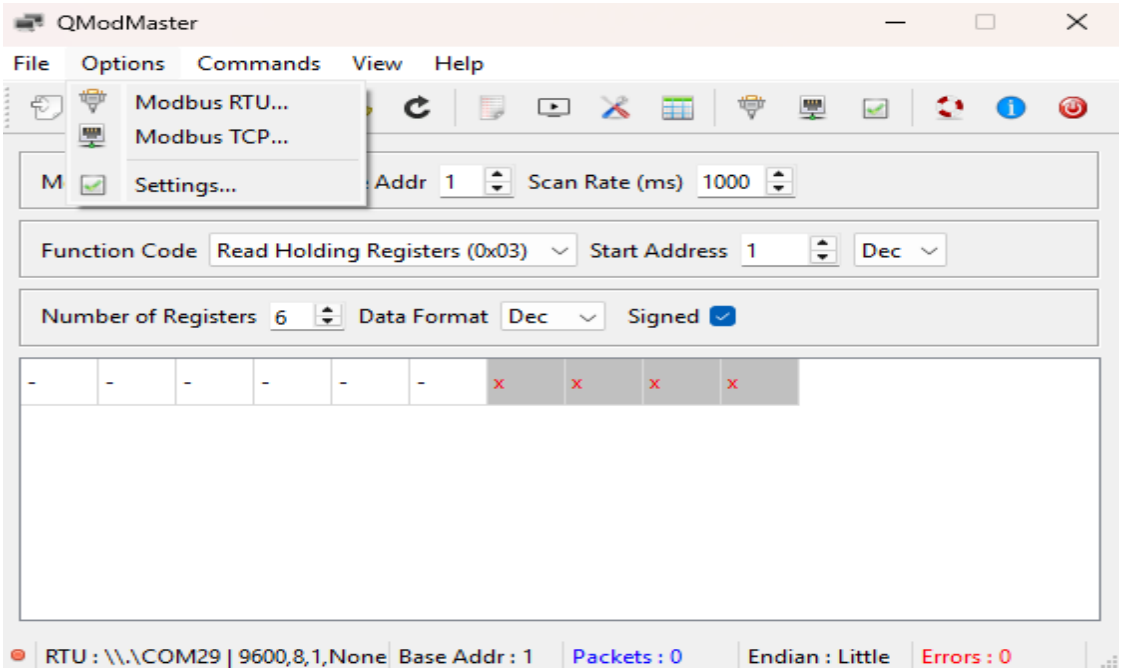
1. Set **Slave Addr = 0** (broadcast).
2. **Function:** *Write Multiple Registers (0x10)*
3. **Start Address:** 0x0202, **Number of Registers: 1**.
4. In the data cell type **2000** (ms) and send.
    - There is **no reply** by design; QModMaster may show a timeout—**that's expected**.
    - If you have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

7. Testing Steps For QModMaster

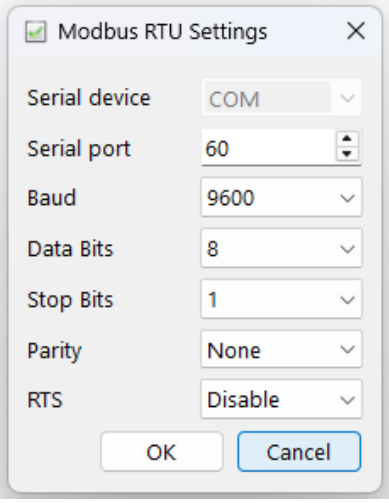
7.1 Modbus output

Follow the steps below for the software setup:-

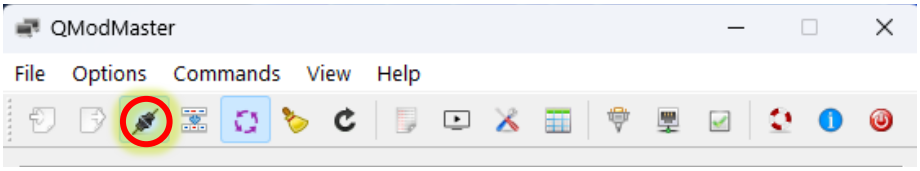
1. Open the software (qModMaster-Win64-exe-0.5.3-beta)
2. Go to *Options* → *Modbus RTU*



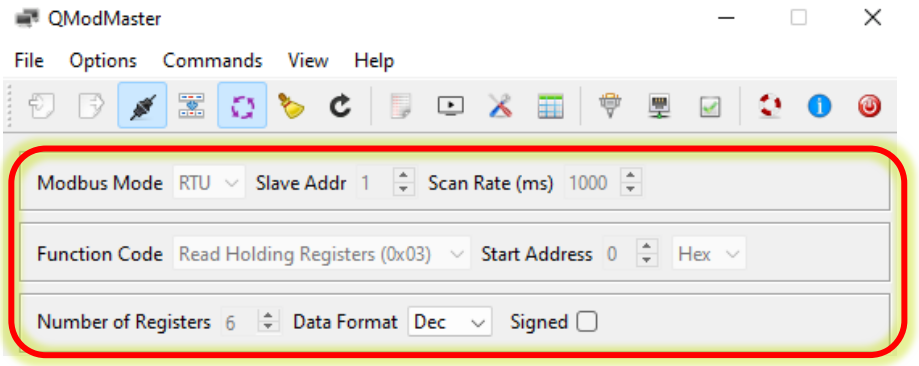
3. Check your COM port in device manager and select the other settings as per the image below. Click *OK*



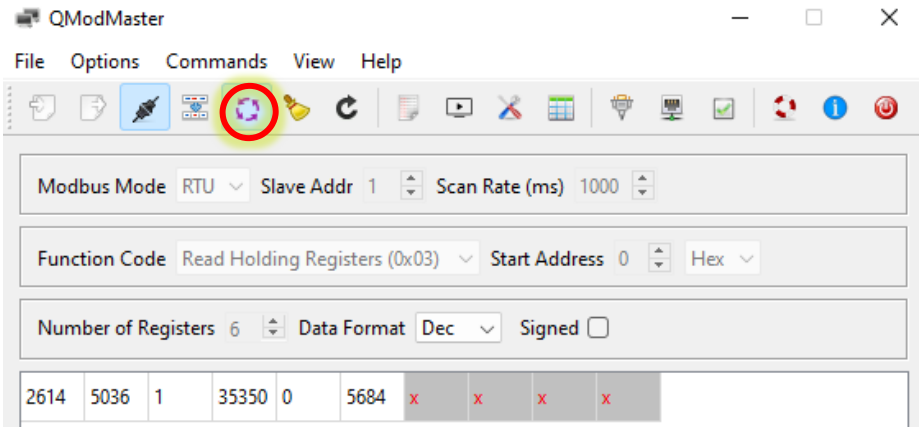
4. Connect your system to the correct COM port and click on this icon.



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



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



Gas resistance low word (0<< 16)|6788 = 6788

Gas resistance high word
Pressure Low word (1<<16)|35350 = 100886
Pressure high word
Humidity value, 5036 / 100 = 50.36%
Temperature value, 2614 / 100 = 26.14°C

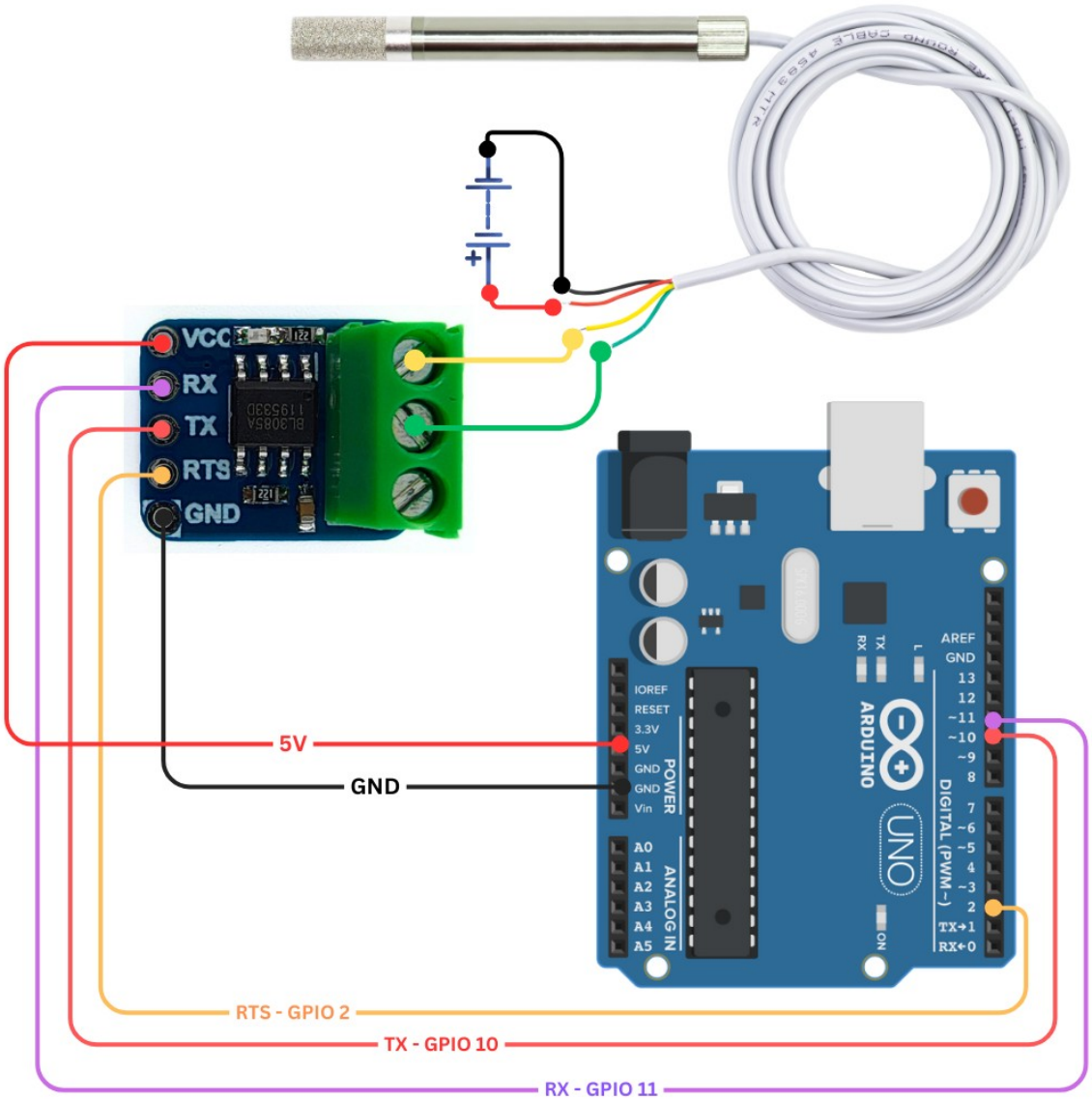
7.2 How to open the probe?

1

2

3

7.3 Connection and sample code for Arduino



Sample output image:-

```
T: 25.80 C  RH: 49.62 %  P: 100799 Pa
T: 25.75 C  RH: 49.79 %  P: 100800 Pa
T: 25.75 C  RH: 49.79 %  P: 100800 Pa
T: 25.70 C  RH: 49.89 %  P: 100800 Pa
T: 25.70 C  RH: 49.89 %  P: 100800 Pa
T: 25.66 C  RH: 50.01 %  P: 100802 Pa
T: 25.66 C  RH: 50.01 %  P: 100802 Pa
T: 25.62 C  RH: 50.11 %  P: 100799 Pa
T: 25.62 C  RH: 50.11 %  P: 100799 Pa    |  Gas: 16120 Ohm
T: 27.94 C  RH: 50.08 %  P: 100814 Pa
T: 26.33 C  RH: 48.73 %  P: 100806 Pa
T: 26.33 C  RH: 48.73 %  P: 100806 Pa
T: 25.93 C  RH: 49.02 %  P: 100798 Pa
T: 25.93 C  RH: 49.02 %  P: 100798 Pa
T: 25.79 C  RH: 49.36 %  P: 100798 Pa
```

7.Mechanical specification