7SEMI

CO2, Temperature & Humidity Sensor Probe

Version 1.0

7SEMI +91 8655821342 info@7semi.com www.7semi.com

Table of Contents

1.Introduction

The **7Semi CO2, Temperature & Humidity Sensor Probe (I2C, 400–5000 PPM)** is designed for accurate and efficient monitoring of indoor air quality parameters. With one of the smallest form factors available, this sensor unlocks new possibilities in compact and cost-sensitive applications. Housed in a **durable metal probe with a 1-metre cable**, it offers high breathability, corrosion resistance, and tolerance to challenging environments.

Built on advanced thermal conductivity sensing technology, this factory-calibrated probe ensures accurate CO2 measurement while also monitoring temperature and relative humidity. it is well-suited for battery-powered devices, IoT systems, smart HVAC equipment, greenhouses, and environmental monitoring projects.

1.1 Features

- CO_2 range: 400–5,000 ppm (extendable to 32,000 ppm)
- Temperature range: –40 to +125 °C
- Humidity range: 0–100 %RH
- High resolution: 1 ppm CO_2
- Factory-calibrated with temperature and humidity compensation
- Low power consumption (950 µA continuous / 100 µA single-shot)
- I^2C digital interface, 3.3 V supply
- Compact probe with 1 m cable

2.Technical Specification

The **Technical Specification** table provides detailed information about **7Semi CO2, Temperature & Humidity Sensor Probe (I2C, 400–5000 PPM) 1M Cable**, including its operating voltage, current consumption, and electrical characteristics. This data helps users understand the power requirements, communication parameters, and performance capabilities of the sensor. It ensures compatibility with different microcontrollers and embedded systems while providing guidelines for efficient integration into various applications.

2.1 General Specification :

Parameter	Value
Supply Voltage	3.3V
Interface	I²C
CO_2 Measurement Range	400–5000 ppm (32,000 ppm extended)
CO_2 Accuracy	±100.0 ppm ±10.0 %m.v.
Resolution (CO_2)	1 ppm
Temperature Accuracy	±0.1 °C (0–65 °C)
Temperature Range	–40 to 125 °C
Humidity Accuracy	±1.0 %RH (0–100 %RH)
Humidity Range	0–100 %RH
Typical Response Time	2 s (T), 4 s (RH)
Operating Temperature	10–40 °C recommended for CO_2
Current Consumption	950 µA (continuous) / 100 µA (single-shot) / 1 µA (sleep)
Cable Length	1 m
Probe Housing	Metal probe with sintered filter
Microcontroller Compatibility	Arduino, STM32, ESP32, ESP8266, nRF52, Raspberry Pi

2.2 I²C Communication Protocol

I²C Overview

- The CO_2TH sensor communicates via the I²C protocol, compliant with the NXP I²C-bus specification.
- Supported Modes: Standard (100 kHz), Fast (400 kHz), and Fast Plus (1 MHz).
- Clock Stretching: *Not supported.*
- All data transfers follow MSB-first (big-endian) byte order.

I²C Address

Address Type	Hex	Description
7-bit Address	0x64	Default operating address

Data Transfer Format

- Each 16-bit word is followed by an 8-bit CRC checksum for data integrity.
- All communication starts with Start (S) and ends with Stop (P) conditions.

Transfer Types

Type	Format
Write	[S] [Addr+W] [CMD] [Data + CRC] [P]
Read	[S] [Addr+W] [CMD] [Sr] [Addr+R] [Data + CRC]...[P]
Combined	Write command → repeated start → read data

CRC-8 Details

Property	Value
Polynomial	0x31 ($x^8 + x^5 + x^4 + 1$)
Init Value	0xFF
Final XOR	0x00
Example	CRC(0xBEEF) = 0x92

Command Set Summary

Command	Hex Code
Start Continuous Measurement	0x218B
Stop Continuous Measurement	0x3F86
Read Measurement	0xEC05
Measure Single Shot	0x219D
Enter Sleep Mode	0x3650
Exit Sleep Mode	0x0000
Perform Conditioning	0x29BC
Perform Soft Reset	0x0006
Perform Factory Reset	0x3632
Perform Self-Test	0x278C

Command	Hex Code
Enable Testing Mode	0x3FBC
Disable Testing Mode	0x3F3D
Perform Forced Recalibration	0x362F
Get Product ID	0x365B

Example Command Explanation

1. **Start Continuous Measurement**
    - Command: 0x218B
    - Function: Starts continuous CO_2 measurement with a 1 s sampling interval.
    - Execution Time: 1000 ms
    - Executable During Measurement: No
2. **Stop Continuous Measurement**
    - Command: 0x3F86
    - Function: Stops ongoing continuous measurement and switches the sensor to idle mode.
    - Execution Time: 1200 ms
    - Executable During Measurement: Yes

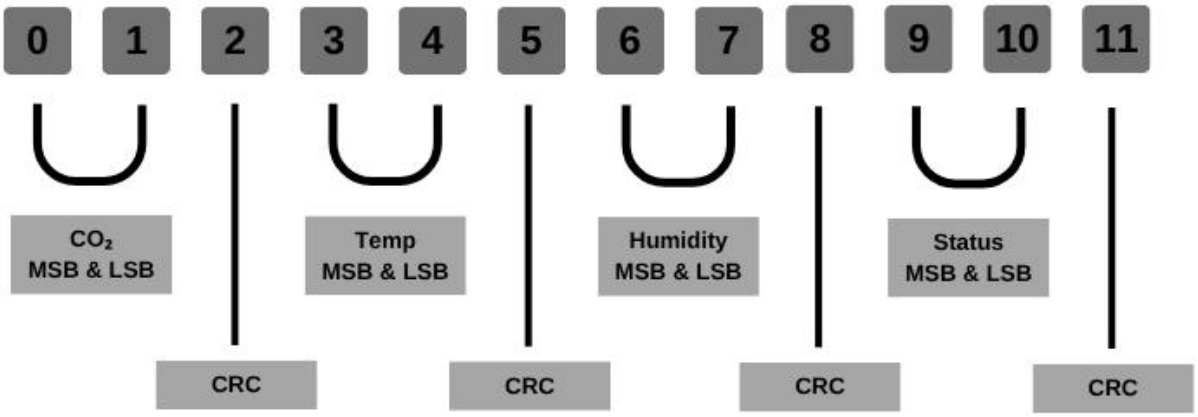
Notes:

- Sensor completes the current measurement before stopping.
- During execution, the device will not acknowledge its I^2C address or accept new commands.

3. Read Measurement (0xEC05)

- Command: 0xEC05
- Function: Reads CO_2, temperature, humidity, and status data from the sensor.
- Execution Time: 1 ms
- Executable During Measurement: Yes

Data Format:



Notes:

- Data buffer clears after each read.
- If no new data, sensor sends NACK.
- Master may stop read with NACK + STOP.

4. Measure Single Shot

- Command: 0x219D
- Function: Performs a single on-demand CO_2 measurement.
- Execution Time: 500 ms
- Executable During Measurement: No

5. Enter Sleep Mode

- Command: 0x3650
- Function: Sets the sensor from idle mode into sleep mode.
- Execution Time: 1 ms
- Executable During Measurement: No

Notes:

- In sleep mode, all analog blocks are powered down to save energy.
- Previously written temperature, humidity, pressure, and ASC states are retained.
- The device must be awakened using the exit_sleep_mode (0x00) command before the next measurement.

6. Exit Sleep Mode

- Command: 0x00
- Function: Wakes the sensor from sleep mode to idle mode.
- Execution Time: 5 ms
- Executable During Measurement: No

Notes:

- Send the I²C address with write direction and a payload byte = 0x00.
- The payload byte is not acknowledged (NACK) by the sensor.
- Successful wake-up can be verified by reading the Product ID (0x365B).

7. Perform Conditioning

- Command: 0x29BC
- Function: Conditions the sensor to stabilize CO_2 measurement accuracy after long idle periods (>3 hrs).
- Execution Time: 22,000 ms
- Executable During Measurement: No

Operation Steps:

- Power up or wake sensor using exit_sleep_mode (0x00).
- Send 0x29BC and wait 22 s for conditioning to complete.
- After completion, start measurement using start_continuous_measurement (0x218B) or measure_single_shot (0x219D).

8. Perform Soft Reset

- Command: 0x06
- Function: Performs a soft reset via the I²C general call, restoring the sensor to its power-up state.
- Execution Time: 10 ms
- Executable During Measurement: Yes

Notes:

- Uses I²C general call address (0x00).
- The command is not acknowledged by the sensor.
- During reset, the device does not respond to any I²C communication.
- All devices on the same bus supporting general call reset will also reset.

9. Perform Factory Reset

- Command: 0x3632
- Function: Resets the FRC (Forced Recalibration) and ASC (Automatic Self-Calibration) history and re-enables the bypass phase.
- Execution Time: 90 ms
- Executable During Measurement: No

Read Response:

- Word[0] = 0x0000 → Command passed
- Word[0] = 0xFFFF → Command failed

Notes:

- Restores calibration and self-calibration state to factory defaults.
- Use when sensor shows calibration drift or persistent offset errors.

10. Perform Self-Test

- Command: 0x278C
- Function: Performs an internal self-test to verify sensor functionality.
- Execution Time: 360 ms
- Executable During Measurement: No

Read Data Format:

Byte	Description
0	Self-Test Result MSB
1	Self-Test Result LSB
2	CRC

Expected Results:

- `0x0000` or `0x0010` → Test Passed
- Other values → Error condition

Bit Definition (for nonzero values):

Bits	Meaning
0	Supply voltage out of range
1	Debug codes
2	Debug codes
3	Debug codes
4	External sensor not detected
5	Memory error
6	Memory error

If Memory Error Occurs:

- Send `perform_soft_reset (0x06)` and retry.
- If persistent, power cycle the device and repeat.
- If still failing, measurement data may be unreliable.

11. Enable Testing Mode

- Command: 0x3FBC
- Function: Enables testing mode by pausing the ASC (Automatic Self-Calibration) algorithm and freezing its state updates.
- Execution Time: —
- Executable During Measurement: Yes

Notes:

- Used for evaluating CO_2 sensing performance during sensor qualification or stability testing.
- While active, ASC updates are halted but measurements continue normally.
- The sensor is in testing mode when bit[14] (2nd MSB) of Byte 10 in the read_measurement output is set to 1.

12. Disable Testing Mode

- Command: 0x3F3D
- Function: Disables testing mode and resumes normal ASC (Automatic Self-Calibration) operation.
- Execution Time: —
- Executable During Measurement: Yes

Notes:

- Exits testing mode and re-enables ASC state updates.
- The sensor is **not in testing mode** when **bit[14] (2nd MSB) of Byte 10** in the read_measurement output is **0**.

13. Perform Forced Recalibration

- Command: 0x362F
- Function: Adjusts the CO_2 output to match a known reference CO_2 concentration (FRC).
- Execution Time: 90 ms
- Executable During Measurement: No

Write Data Format:

Byte	Description
0	Target CO_2 concentration MSB
1	Target CO_2 concentration LSB
2	CRC

Read Data Format:

Byte	Description
0	CO_2 correction value MSB
1	CO_2 correction value LSB
2	CRC

Notes:

- Acceptable CO_2 input range: 0 – 32,000 ppm.
- Output 0xFFFF indicates calibration failure.
- Ensure stable CO_2 level and environment before performing FRC.
- Recommended Operation Sequence:
- Operate sensor ≥30 s in continuous mode, or 30 single-shot readings (~5 min @10 s interval).
- If in continuous mode, stop with stop_continuous_measurement (0x3F86) and wait 1.2 s.
- Send perform_forced_recalibration (0x362F) with target CO_2 value.
- Optionally read back the correction data.

14. Get Product ID

- Command: 0x365B
- Function: Reads the sensor's Product ID and unique 64-bit serial number for identification and communication verification.
- Execution Time: 1 ms
- Executable During Measurement: No

Read Data Format:

Data	Description
4 Bytes + CRC	Product ID
8 Bytes + CRC	Unique Serial Number

Example Output:

Product ID: 0x0901018A

Notes:

- Confirms correct sensor model and I^2C communication integrity.
- Useful for device tracking and production testing

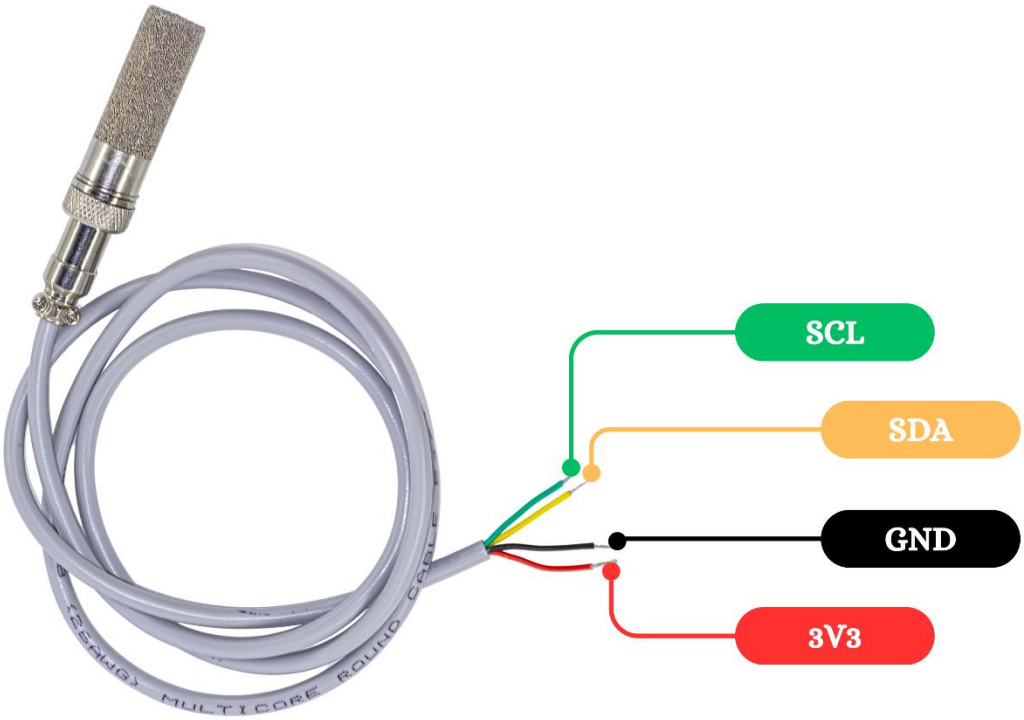
Conversion Formulas

Signal	Formula	Example
Temperature (Output)	T[°C] = 175 × (Output / 65535) − 45	25 °C → Output = 26,214
Humidity (Output)	RH[%] = 125 × (Output / 65535) − 6	Output = 29,360 → 50 %RH
Pressure (Input)	Input = P[Pa] / 2	101,300 Pa → 50,650
CO_2	Output = CO_2 ppm	500 ppm → 500

Notes:

- All outputs are digitally calibrated values read via I²C.
- CRC (8-bit, polynomial 0x31) applies to every 16-bit data word.
- Input scaling ensures correct compensation across full measurement ranges.

3.Pinouts

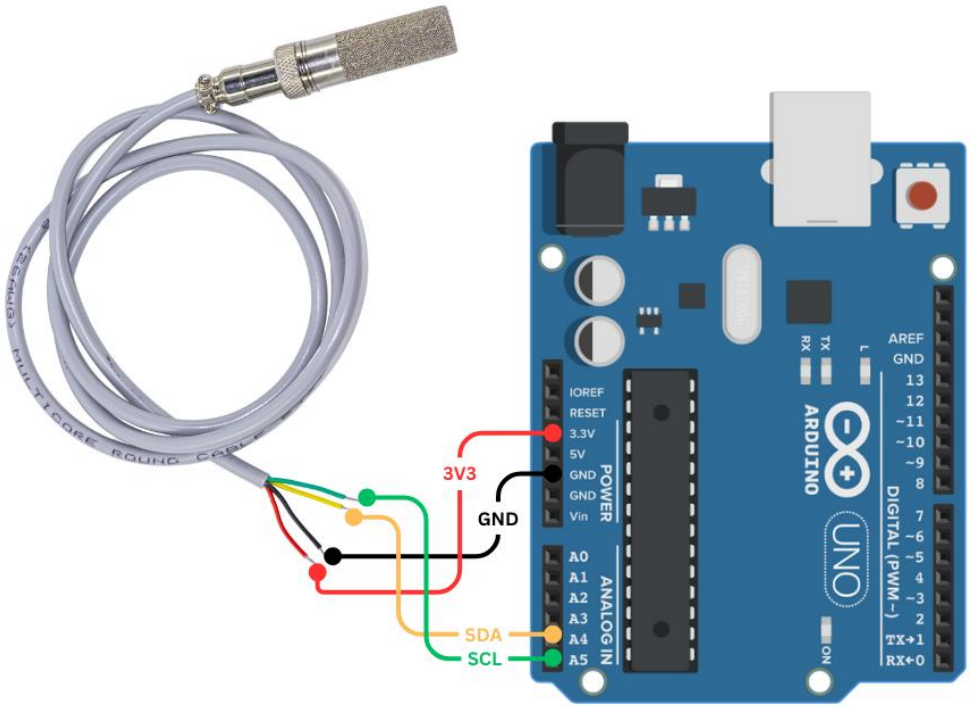


Pin	Name	Description
1	**VCC**	Power Input (3.3V DC)
2	**SCL**	I²C Clock Line
3	**SDA**	I²C Data Line
4	**GND**	Ground Connection

Connection Guidelines

- Power Supply: Connect the red wire to a 3.3V DC power source and the black wire to ground (GND). Ensure the power supply is stable and within the rated voltage range.
- I²C Communication: Connect the yellow wire of prob to SDA (I²C Data) of controller and the Green wire of prob to SCL (I²C Clock) controller.

4.Hardware Interface



Connection Explanation :

- The VCC (Red Wire) of the sensor prob is connected to 3.3V on the Arduino UNO.
- The GND (Black Wire) of the sensor is connected to the GND pin of the Arduino UNO
- The SCL (Clock Line) of the sensor prob is connected to A5 (SCL) on the Arduino UNO.
- The SDA (Data Line) of the sensor is connected to A4 (SDA) on the Arduino UNO.
- These connections allow the Arduino UNO to communicate with 7Semi CO2, Temperature & Humidity Sensor Probe (400–5000 PPM) 1M Cable using the I²C protocol.

5.Example code link

```
#include <7Semi_CO2TH.h>

/** - One driver instance */
CO2TH_7Semi CO2TH;

/** - Read period (ms): match sensor frame interval */
static const unsigned long kPeriodMs = 1200;

void setup() {
  /** - Serial + I2C */
  Serial.begin(115200);
  while (!Serial)
    ;
  // Ardiuno UNO
  err_t e = CO2TH.Begin();
  // err_t e = CO2TH.Begin(/*sda*/ 21, /*scl*/ 22, /*freq*/ 400000,
/*port*/ 0);
  if (e != NO_ERROR) {
    Serial.print(F("Begin failed, err="));
    Serial.println((int)e);
    while (true) { delay(1000); }
  }

  /** - Start continuous measurement */
  e = CO2TH.StartContinuousMeasurement();
  if (e != NO_ERROR) {
    Serial.print(F("Start failed, err="));
    Serial.println((int)e);
    while (true) { delay(1000); }
  }

  /** - Give the sensor one frame to produce the first result */
  delay(kPeriodMs);
}

void loop() {
  static unsigned long last = 0;
  const unsigned long now = millis();
  if (now - last < kPeriodMs) return;
  last = now;
```

```
  /** - Read and print values */
  int16_t co2 = 0;
  float temp = 0.0f;
  float rh = 0.0f;
  uint16_t st = 0;

  err_t e = CO2TH.ReadMeasurement(co2, temp, rh, st);
  if (e != NO_ERROR) {
    Serial.print(F("Read failed, err="));
    Serial.println((int)e);
    return;
  }

  Serial.print(F("CO2 = "));
  Serial.print(co2);
  Serial.print(F(" ppm, T = "));
  Serial.print(temp, 2);
  Serial.print(F(" °C, RH = "));
  Serial.println(rh, 2);
}
```

1. Library Inclusion

- #include <7Semi_CO2TH.h> — includes the 7Semi driver for the CO_2TH probe. This library internally handles I²C communication, CRC validation, and command sequences.

2. Driver Initialization

- CO2TH_7Semi CO2TH; — creates an instance of the driver class.
- CO2TH.Begin(0x64, 21, 22, 400000, 0);
    - Initializes the I²C bus with:
        - **Address:** 0x64 (default)
        - **SDA:** A4
        - **SCL:** A5
        - **Frequency:** 400 kHz
    - If address is unknown, passing 0 scans 0x08–0x77 automatically.

3. **Measurement Start**
    - CO2TH.StartContinuousMeasurement(); — enables continuous sampling mode at 1 Hz.
      The sensor begins outputting CO_2, temperature, and humidity data frames.
4. **Data Read**
    - CO2TH.ReadMeasurement(co2, temp, rh, st); — retrieves:
        - co2 → CO_2 concentration (ppm)
        - temp → Temperature (°C)
        - rh → Relative Humidity (%)
        - st → Status flags (0 = OK)
    - Measurements are printed every 1.2 seconds to match the sampling interval.

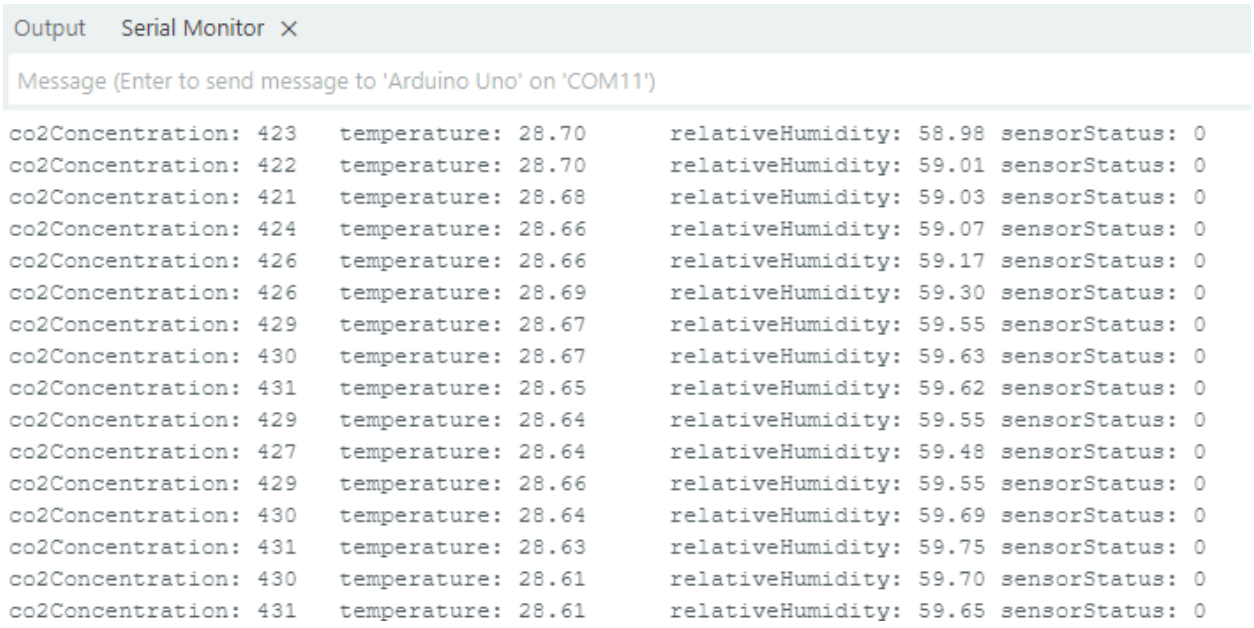
How to Install 7Semi Arduino Library

- 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.

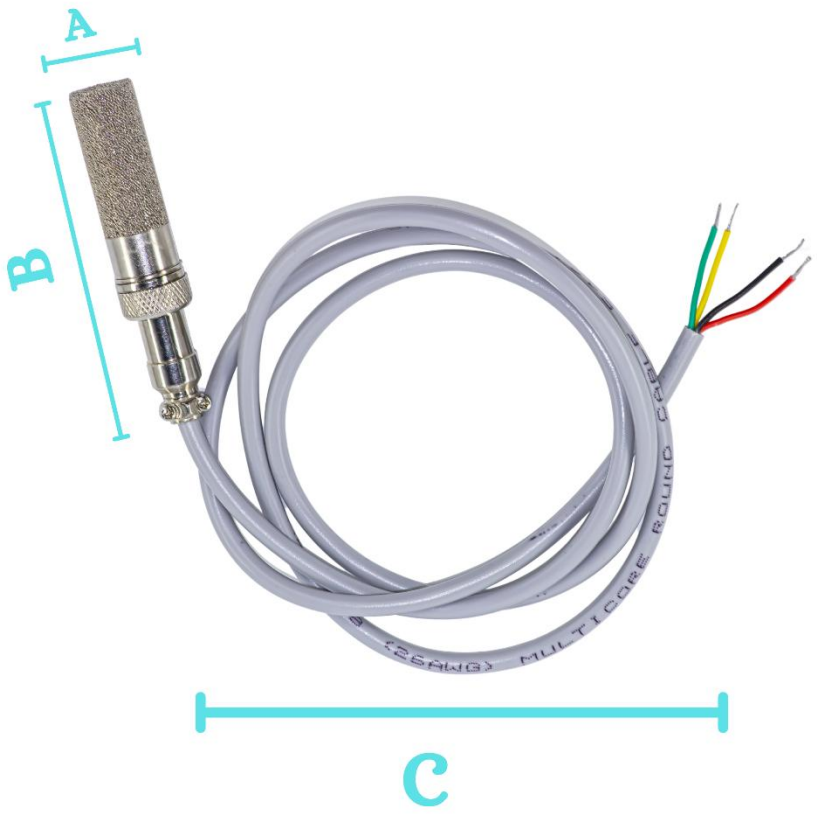
Sample code link for manual download:- "7Semi_CO2TH-sensor-probe"

5.1 Sample Serial Output Arduino

Sample output Image of Arduino:



6.Mechanical Specification



S.No	Symbol	Mesurement
1	A (Probe Diameter)	14mm
2	B (Probe length)	93mm
3	C (wire length)	1meter