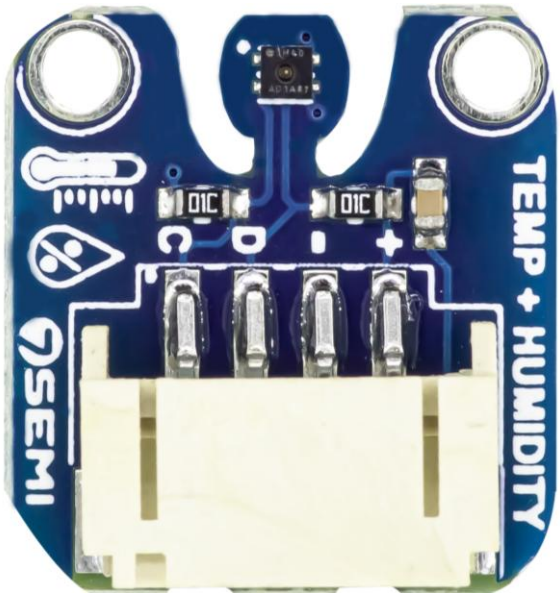
7SEMI

SHT40 Digital Humidity and Temperature Sensor Breakout with 4Pin Connector

Version 1.0

Table of Contents

1.Introduction



The **7Semi SHT40 Digital Humidity and Temperature Sensor Breakout with 4Pin Connector** integrates Sensirion's latest 4th-generation SHT4x sensor technology. This compact and high-accuracy sensor is ideal for precise humidity and temperature measurements. Designed for reliability and efficiency, the SHT40 supports I²C communication and operates across a wide voltage range, making it suitable for various embedded and IoT applications.

1.1 Features

- High accuracy: ±1.8 %RH, ±0.2 °C (typical)
- Operating range: 0% to 100% RH, -40 to 125 °C
- Fully calibrated and linearized output
- I²C digital interface (0x44 default address)
- Ultra-low power consumption: 0.4 µA average
- Optional PTFE membrane for IP67 protection
- Compatible with Arduino, ESP32, Raspberry Pi, STM32

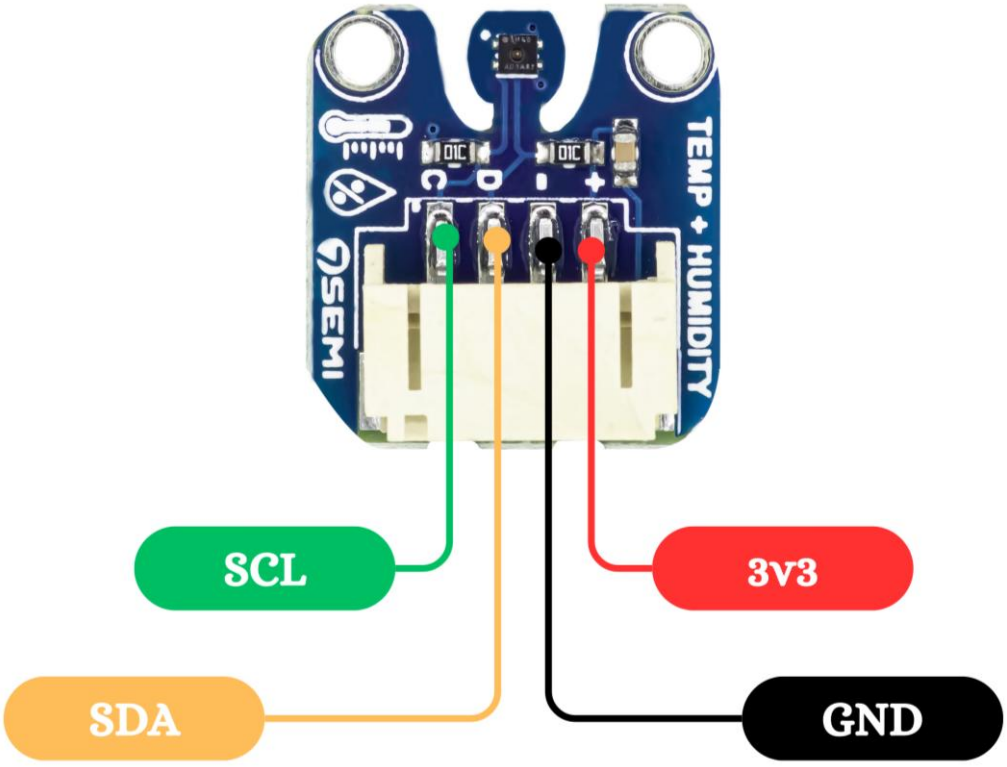
2.Technical Specification

The **Technical Specification** table provides detailed information about **7Semi SHT40 Digital Humidity and Temperature Sensor Breakout with 4Pin Connector**, 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.

SHT40 Specifications

- Relative Humidity Accuracy: ±1.8 %RH (typ.)
- Temperature Accuracy: ±0.2 °C (typ.)
- Supply Voltage: 1.08 V to 3.6 V
- Idle Current: 80 nA
- Average Current: 0.4 µA (low repeatability)
- Interface: I²C, 16-bit resolution
- Connector: 4Pin Connector
- Breakout Dimensions: 17.08 x 15.78 mm

3.Pinouts

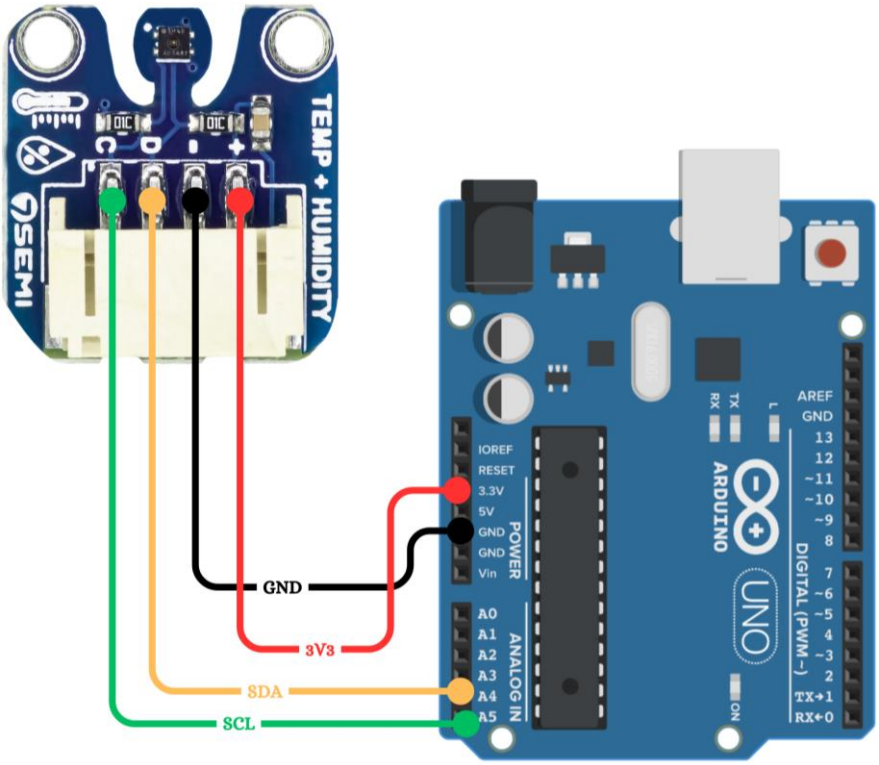


Pin	Name	Description
1	**VCC**	Power Input (3.3V DC)
2	**SCL**	I²C Clock Line
3	**SDA**	I²C Data Line
4	**GND**	Ground Connection

Connection Guidelines

- Ensure a stable 3.3V DC power source.
- Connect SCL to SCL and SDA to SDA for proper communication.

4.Hardware Interface



Connection Explanation :

- The **VCC** pin of the sensor is connected to **3.3V** on the Arduino UNO.
- The **GND** pin of the sensor is connected to the **GND** pin of the Arduino UNO to The **SCL (Clock Line)** of the sensor 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 the **SHT40 sensor** using the I²C protocol.

5.Example code link

We provide example codes to help you get started with the **7Semi SHT40 Digital Humidity and Temperature Sensor Breakout with 4Pin Connector**. These examples demonstrate how to communicate with the sensor and retrieve temperature and humidity data using the I²C protocol. The code is available for two popular platforms: Arduino and ESP32.

Code Explanation

```
#include <Arduino.h>
#include <SensirionI2cSht4x.h>
#include <Wire.h>

// macro definitions
// make sure that we use the proper definition of NO_ERROR
#ifdef NO_ERROR
#undef NO_ERROR
#endif
#define NO_ERROR 0

SensirionI2cSht4x sensor;

static char errorMessage[64];
static int16_t error;

void setup() {

    Serial.begin(115200);
    while (!Serial) {
        delay(100);
    }
    Wire.begin();
    sensor.begin(Wire, SHT40_I2C_ADDR_44);

    sensor.softReset();
    delay(10);
    uint32_t serialNumber = 0;
    error = sensor.serialNumber(serialNumber);
    if (error != NO_ERROR) {
        Serial.print("Error trying to execute serialNumber(): ");
        errorToString(error, errorMessage, sizeof errorMessage);
        Serial.println(errorMessage);
        return;
    }
    Serial.print("serialNumber: ");
    Serial.print(serialNumber);
    Serial.println();
```

```
}

void loop() {

    float aTemperature = 0.0;
    float aHumidity = 0.0;
    delay(20);
    error = sensor.measureLowestPrecision(aTemperature, aHumidity);
    if (error != NO_ERROR) {
        Serial.print("Error trying to execute measureLowestPrecision():
");
        errorToString(error, errorMessage, sizeof errorMessage);
        Serial.println(errorMessage);
        return;
    }
    Serial.print("aTemperature: ");
    Serial.print(aTemperature);
    Serial.print("\t");
    Serial.print("aHumidity: ");
    Serial.print(aHumidity);
    Serial.println();
}
```

1. **Library Inclusions**
   - `#include <Arduino.h>` → Core Arduino functions.
   - `#include <SensirionI2cSht4x.h>` → Sensirion's official driver library for SHT40.
   - `#include <Wire.h>` → Enables I²C communication.

2. **Sensor Object and Error Handling**
   - `SensirionI2cSht4x sensor;` → Creates a sensor object to access SHT40 functions.
   - `static int16_t error;` → Stores error codes from sensor function calls.
   - `static char errorMessage[64];` → Buffer to store error messages.
   - `#define NO_ERROR 0` → Represents a successful operation with no errors.

3. **Sensor Initialization (setup())**
   - Starts Serial communication at 115200 baud.
   - Waits for Serial Monitor to open (important for some boards).
   - Initializes I²C communication using `Wire.begin()`.
   - Begins SHT40 communication with `sensor.begin(Wire, SHT40_I2C_ADDR_44)`.
   - Issues a soft reset using `sensor.softReset()` to ensure a clean start.
   - Reads the sensor's unique 32-bit serial number.
   - If successful, prints the serial number; otherwise, displays an error message.

4. **Measuring Temperature and Humidity (loop())**
   - Declares `aTemperature` and `aHumidity` as float variables.
   - Waits 20ms before each measurement to ensure stability.
   - Uses `sensor.measureLowestPrecision()` for a fast, low-power measurement mode.

• If successful, prints both temperature (°C) and relative humidity (%RH) to Serial Monitor.
• If an error occurs, the corresponding message is displayed.

5. **Low Precision Mode**
   • `measureLowestPrecision()` → Provides fast results with lower resolution.
   • Ideal for energy-efficient, battery-powered applications where speed matters more than ultra-high accuracy.

6. **Error Handling**
   • After each sensor call, `if (error != NO_ERROR)` checks for issues.
   • Uses `errorToString()` to convert numeric errors into readable messages.
   • Ensures clear debugging feedback during development

7. **Arduino Example Code**

This example is designed for Arduino-compatible boards and demonstrates:

- Initializing the I²C communication with the SHT40 sensor board.
- Reading temperature and Humidity data from the sensor.
- Printing the sensor data to the Serial Monitor.

8. **ESP32 Example Code**

This example targets ESP32 boards and showcases:

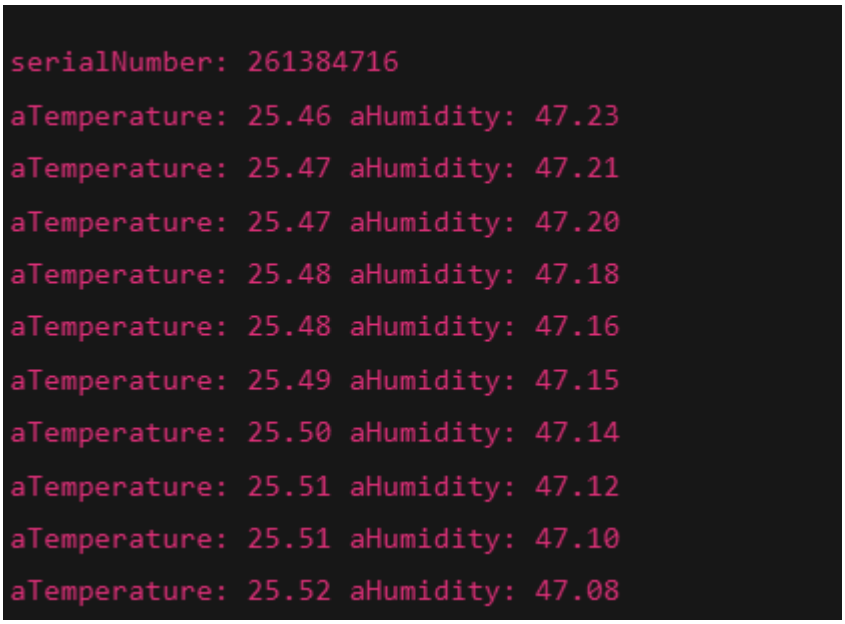
- Configuring the ESP32 I²C interface to communicate with the SHT40 sensor board.
- Reading temperature and Humidity data from the sensor.
- Printing the sensor data to the Serial Monitor.

How to Access the Code

- **Download Link for Arduino and ESP32 Example**: Click Here

5.1 Sample Serial Output Arduino

Sample output Image of Arduino:



6.Mechanical Specification