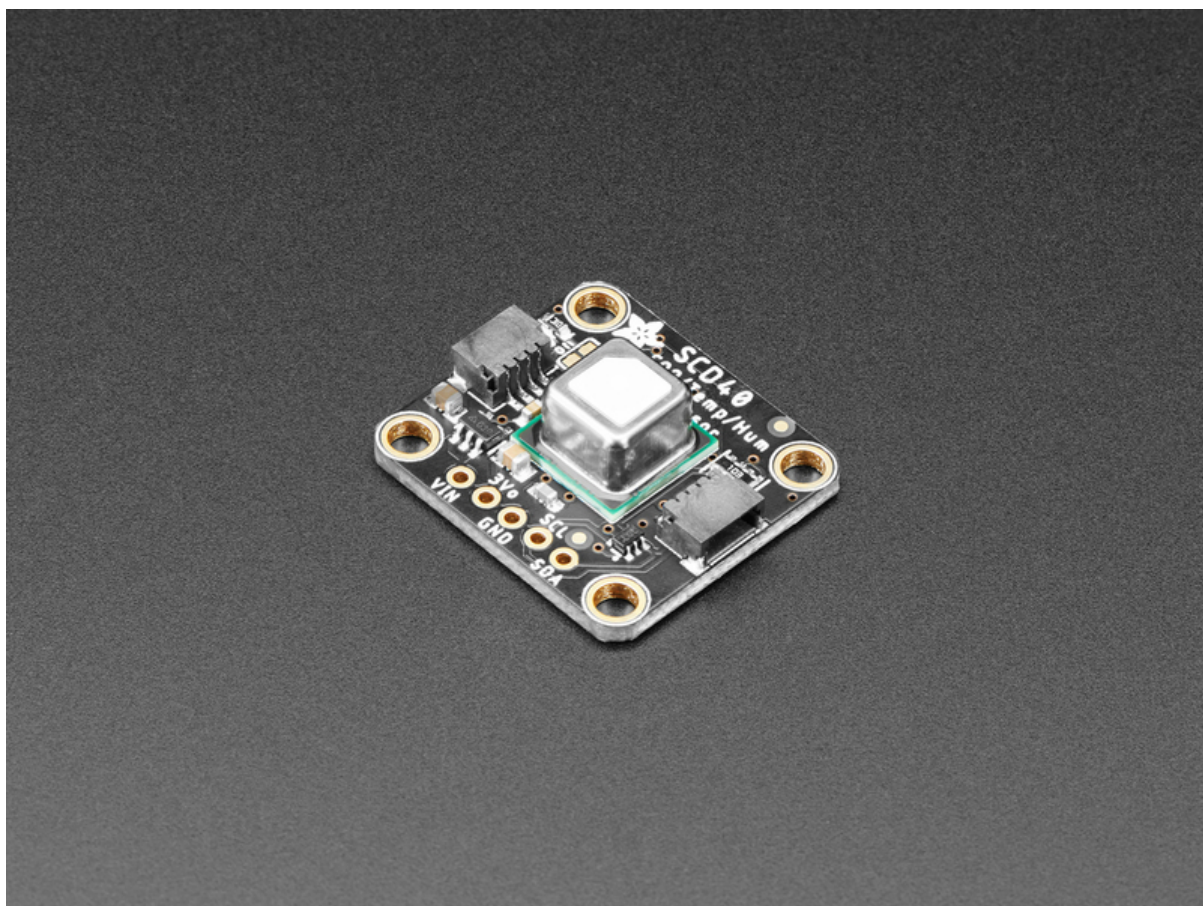




# Adafruit SCD-40 and SCD-41

Created by Kattni Rembor



<https://learn.adafruit.com/adafruit-scd-40-and-scd-41>

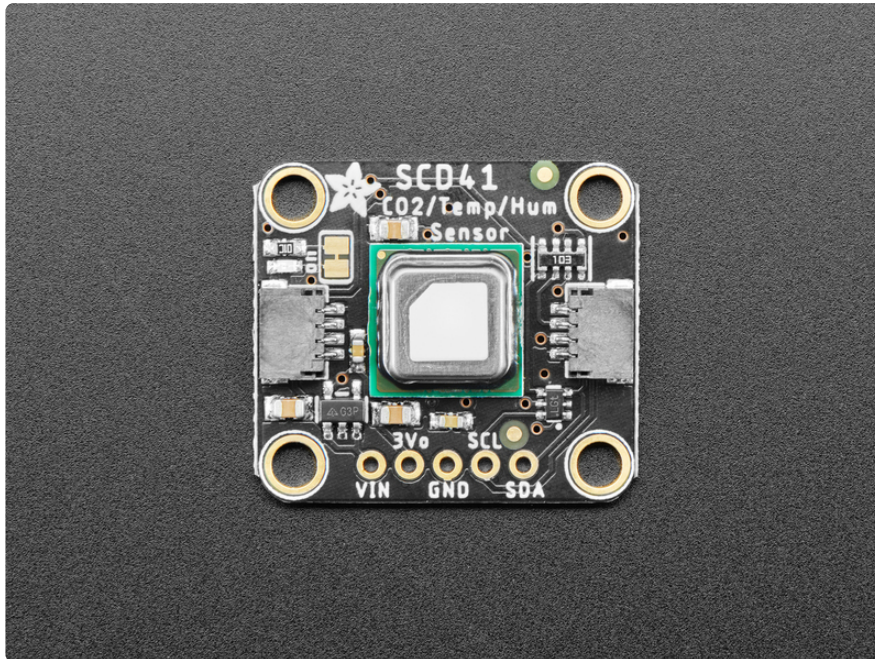
Last updated on 2024-05-15 10:40:35 AM EDT

# Table of Contents

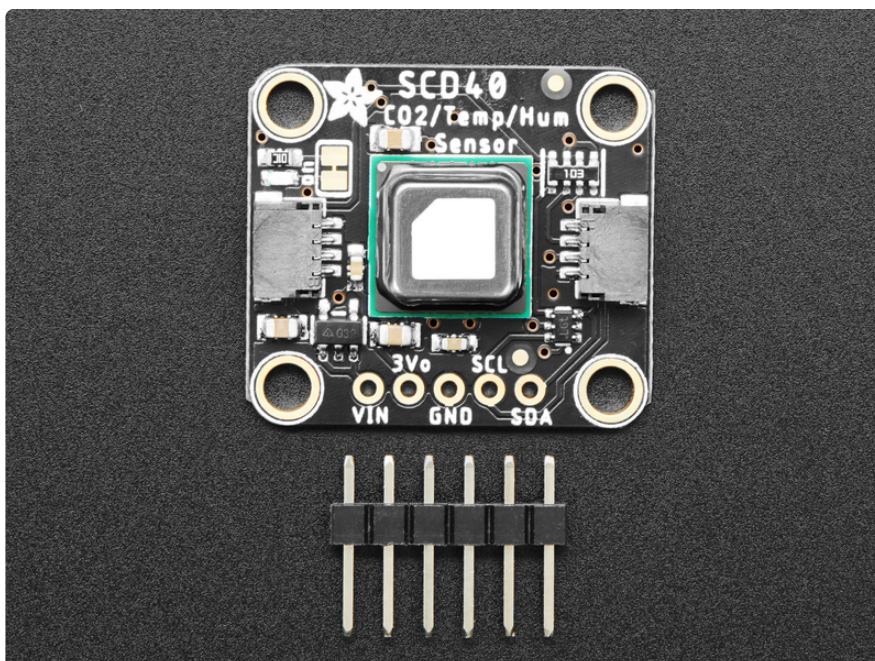
Overview	3
Pinouts	6
<ul style="list-style-type: none"><li>• Power Pins</li><li>• I2C Logic Pins</li><li>• Status LED</li><li>• Jumpers</li></ul>	
Python & CircuitPython	9
<ul style="list-style-type: none"><li>• CircuitPython Microcontroller Wiring</li><li>• Python Computer Wiring</li><li>• Python Installation of SCD4x Library</li><li>• CircuitPython Usage</li><li>• Python Usage</li><li>• Example Code:</li></ul>	
Python Docs	13
Arduino	14
<ul style="list-style-type: none"><li>• I2C Wiring</li><li>• Library Installation</li><li>• Load Example</li></ul>	
WipperSnapper	18
<ul style="list-style-type: none"><li>• What is WipperSnapper</li><li>• Wiring</li><li>• Usage</li></ul>	
Downloads	25
<ul style="list-style-type: none"><li>• Files</li><li>• Schematic and Fab Print</li></ul>	

---

# Overview



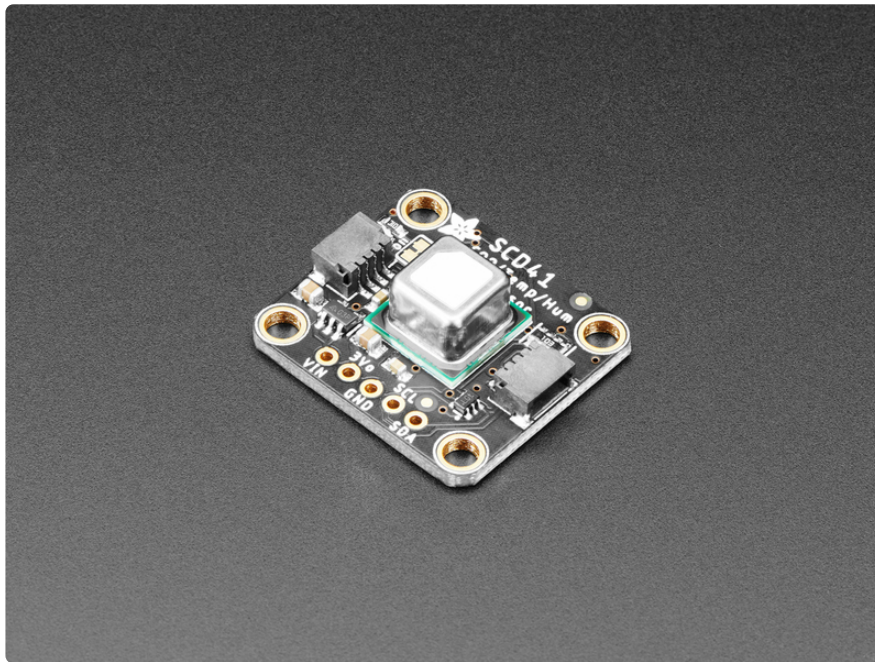
Take a deep breath in...now slowly breathe out. Mmm isn't it wonderful? All that air around us, which we bring into our lungs, extracts oxygen from and then breathes out carbon dioxide. CO2 is essential for life on this planet we call Earth - we and plants take turns using and emitting CO2 in an elegant symbiosis. But it's important to keep that CO2 balanced - you don't want too much around, not good for humans and not good for our planet.



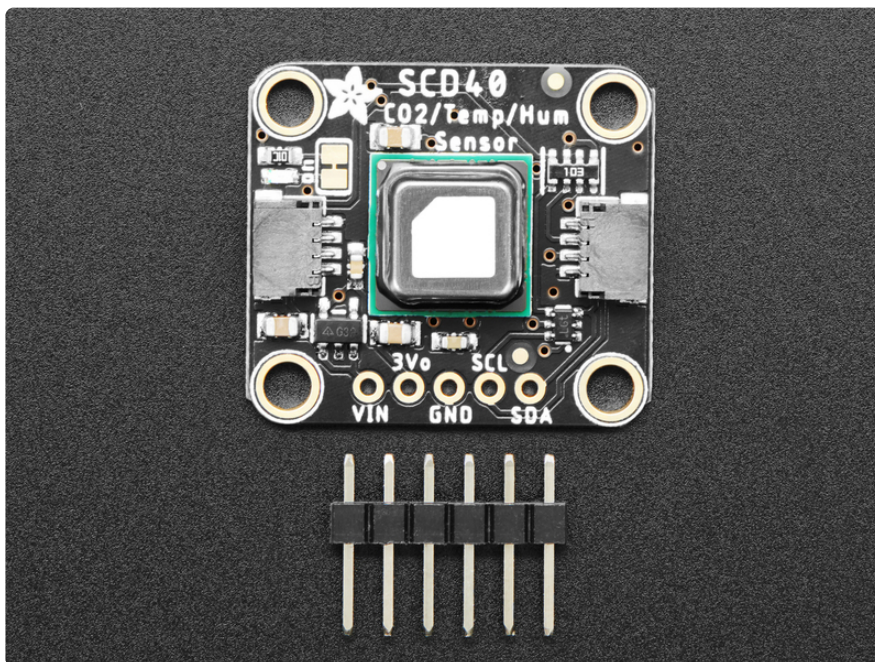
The SCD-40 and SCD-41 are photoacoustic 'true' CO2 sensors that will tell you the CO2 PPM (parts-per-million) composition of ambient air. [Unlike the SGP30, this sensor](#)



isn't approximating it from VOC gas concentration (<http://adafru.it/3709>) - they really are measuring the CO2 concentration! That means they're bigger and more expensive, but they are the real thing. Perfect for environmental sensing, scientific experiments, air quality and ventilation studies, and more.

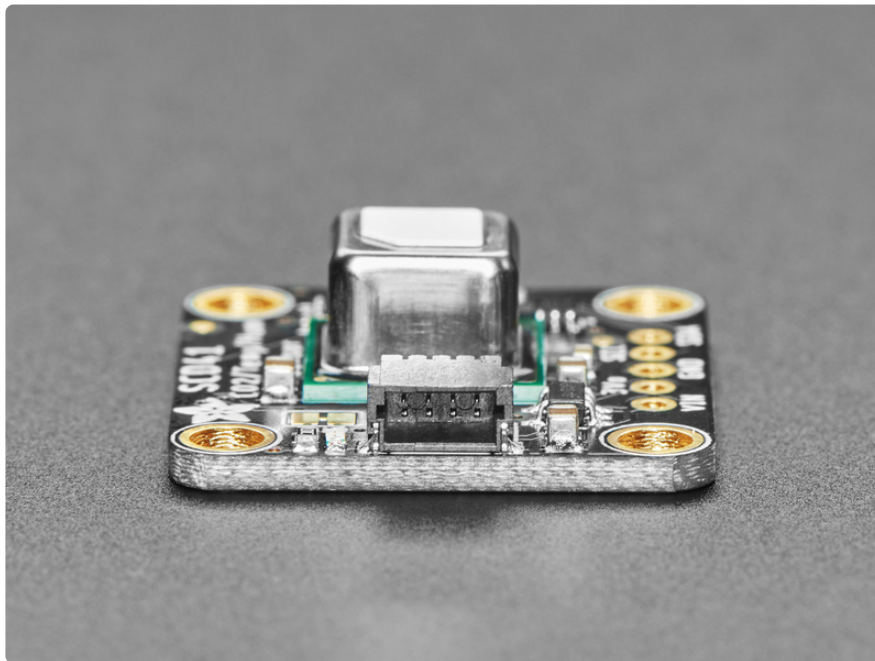


Compared to the illustrious SCD-30 (<http://adafru.it/4867>), this sensor uses a different measurement technique, which allows it to be much smaller. The overall data quality should be very similar, however. Like the SCD-30, this sensor has data read over I2C, so it works very nicely with just about any microcontroller or microcomputer. There's both Arduino and Python/CircuitPython code so you can get started in a jiffy.



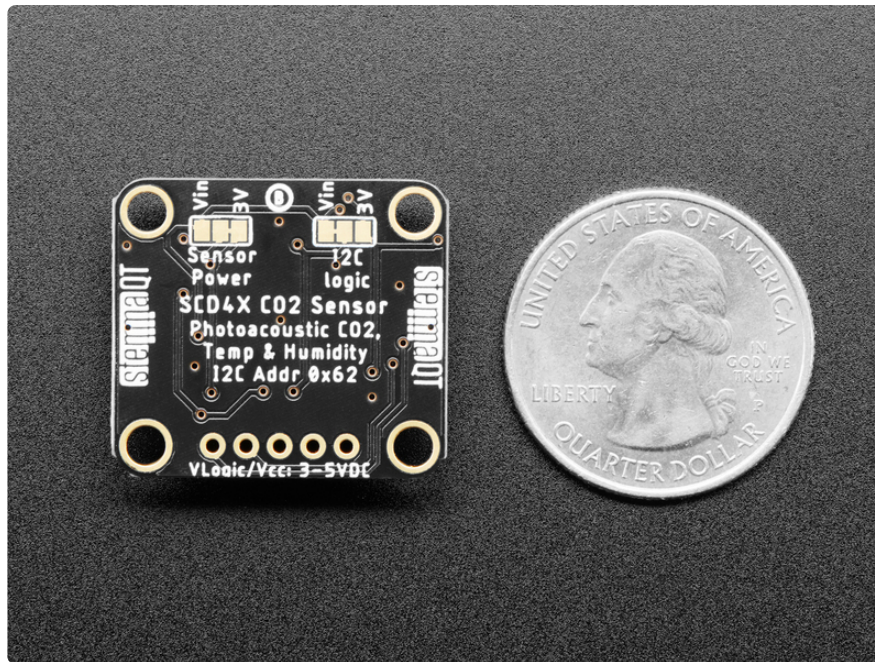
There are two variants of this sensor - the SCD-40 and SCD-41

- The **SCD-40** is lower cost and is perfect for indoor/outdoor air quality and CO<sub>2</sub> measurements. It has a range of **400~2000 ppm** with an accuracy of  $\pm(50 \text{ ppm} + 5\% \text{ of reading})$
- The **SCD-41** is more expensive, and while it can definitely be used for air quality, it's wide range means it's best used for industrial or scientific CO<sub>2</sub> measurements where the ppm can get very high. It has a range of **400~5000 ppm** with an accuracy of  $\pm(40 \text{ ppm} + 5\% \text{ of reading})$



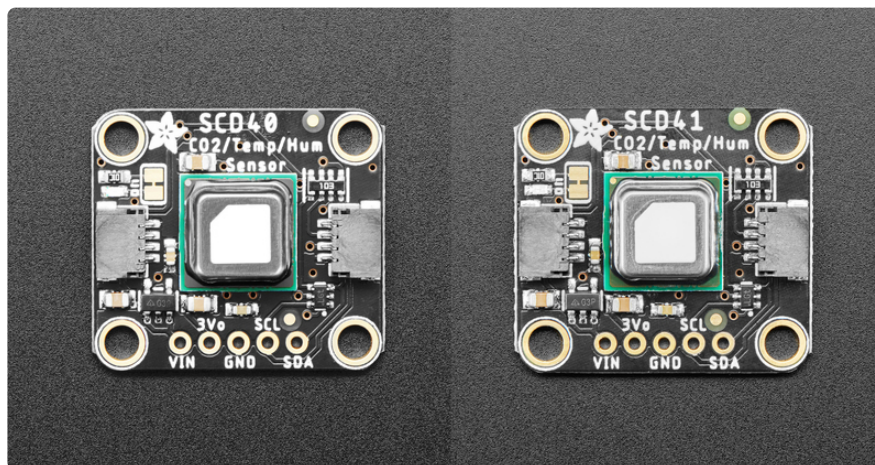
Nice sensor, right? So we made it easy for you to get right into your next project. The sensor is soldered onto a custom-made PCB in the **STEMMA QT form factor** (<https://adafru.it/LBQ>), making them easy to interface with. The **STEMMA QT connectors** (<https://adafru.it/JqB>) on either side are compatible with the **SparkFun Qwiic** (<https://adafru.it/Fpw>) I2C connectors. This allows you to make solderless connections between your development board and the SCD-4x or to chain it with a wide range of other sensors and accessories using a **compatible cable** (<https://adafru.it/JnB>).





This sensor can run from 3.3 to 5V, but it's more important for it to have a quiet power supply with low ripple, than any particular voltage. For that reason, we've added a 3.3V regulator and level shifters: when connecting to a 5V microcontroller like an Arduino UNO the 5V supply is often shared with other electronic components that add noise. The onboard regulator will keep the voltage nice and quiet. For advanced hackers, they can cut/solder the backtraces to change whether the regulator is enabled and what I2C logic level is desired.

## Pinouts



The SCD-40 and SCD-41 have the same pinouts.

## Power Pins

- **VIN** - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 3V microcontroller like a Feather M4, use 3V, or for a 5V microcontroller like Arduino, use 5V.
- **3Vo** - This is the output from the onboard 3.3V regulator. If you have a need for a clean 3.3V output, you can use this! It can provide at least 100mA output.
- **GND** - This is common ground for power and logic.

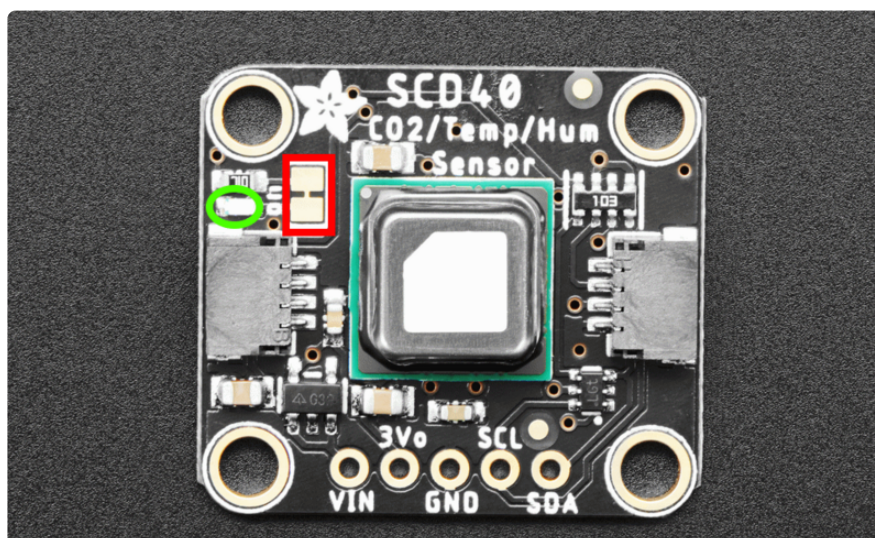
## I2C Logic Pins

The default I2C address for the SCD-4x is **0x62**.

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. There's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. There's a **10K pullup** on this pin.
- **STEMMA QT** (<https://adafru.it/Ft4>) - These connectors allow you to connect to development boards with **STEMMA QT** connectors or to other things with [various associated accessories](https://adafru.it/Ft6) (<https://adafru.it/Ft6>).

## Status LED

There is a status LED marked "on" when the sensor is powered (circled in green).





If you do not want this LED to light up (saving power or avoiding light), cut the circuit board jumper in the red box. If you ever want to reenale it, bridge the pads with a bit of solder.

## Jumpers

The SCD-4x can run from 3 to 5V DC, for both power and I2C data/logic. Normally that would mean we wouldn't put a regulator and logic level shifter on the breakout. However, the SCD-4x also does best with a quiet power supply that can also supply ~200mA peak. Since that may or may not be the same as the logic power supply of the microcontroller, advanced users can customize the power/logic level setup for the sensor.

On the left is the level-shifted **I2C logic level**. Most of the time, Vin is the power and logic level for the microcontroller. However, if you are, say, powering from 5v because it's a better power supply source, but are using a 3V logic microcontroller, you can cut and re-solder this jumper.

On the right is the **Sensor Power** jumper. By default we power the sensor through the 3V regulator. If you happen to have a nice and quiet Vin power supply, you can cut and re-solder this jumper.





The original silkscreen on the SDC4x boards labeled the jumpers backwards. Sensor Power was on the left and I2C logic was on the right, as shown below. If the image below matches your board, be aware that they are INCORRECTLY labeled!



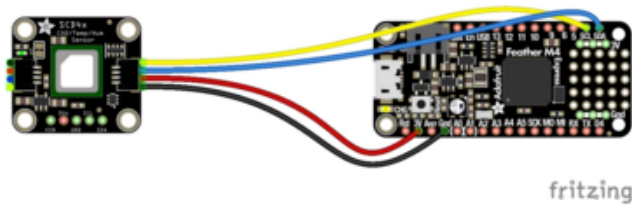
## Python & CircuitPython

It's easy to use the **SCD-4x** with Python or CircuitPython, and the [Adafruit CircuitPython SCD4x](https://adafru.it/UZe) (<https://adafru.it/UZe>) module. This module allows you to easily write Python code that reads CO2, temperature, and humidity from the **SCD-4x** sensor.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit\\_Blinka, our CircuitPython-for-Python compatibility library](https://adafru.it/BSN) (<https://adafru.it/BSN>).

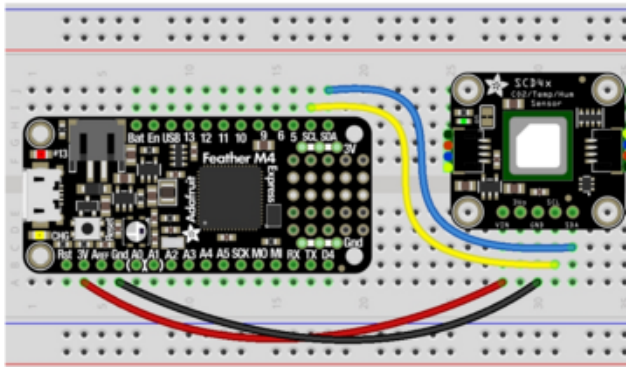
## CircuitPython Microcontroller Wiring

First wire up a SCD-4x to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy [STEMMA QT](https://adafru.it/Ft4) (<https://adafru.it/Ft4>) connectors:



Board 3V to sensor VIN (red wire)  
 Board GND to sensor GND (black wire)  
 Board SCL to sensor SCL (yellow wire)  
 Board SDA to sensor SDA (blue wire)

You can also use the standard **0.100"** pitch headers to wire it up on a breadboard:

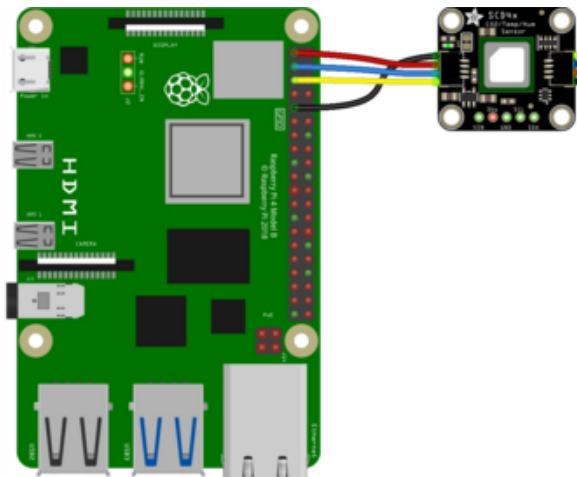


Board 3V to sensor VIN (red wire)  
 Board GND to sensor GND (black wire)  
 Board SCL to sensor SCL (yellow wire)  
 Board SDA to sensor SDA (blue wire)

## Python Computer Wiring

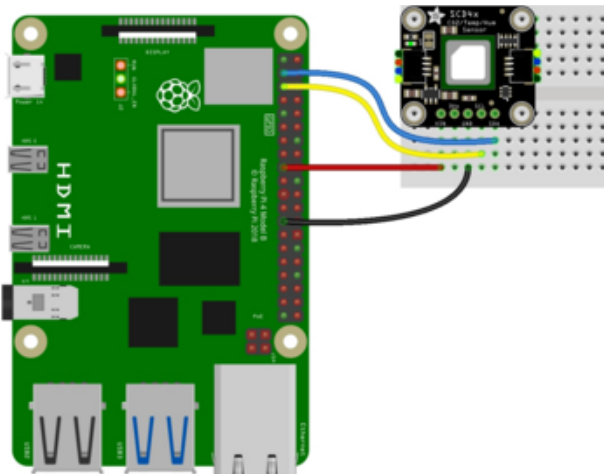
Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (<https://adafru.it/BSN>).

Here's the Raspberry Pi wired to the sensor using I2C and a [STEMMA QT](https://adafru.it/Ft4) (<https://adafru.it/Ft4>) connector:



Pi 3V to sensor VIN (red wire)  
 Pi GND to sensor GND (black wire)  
 Pi SCL to sensor SCL (yellow wire)  
 Pi SDA to sensor SDA (blue wire)

Finally here is an example of how to wire up a Raspberry Pi to the sensor using a solderless breadboard:



Pi 3V to sensor VIN (red wire)  
 Pi GND to sensor GND (black wire)  
 Pi SCL to sensor SCL (yellow wire)  
 Pi SDA to sensor SDA (blue wire)

## Python Installation of SCD4x Library

You'll need to install the **Adafruit\_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready \(https://adafru.it/BSN\)](https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `pip3 install adafruit-circuitpython-scd4x`

If your default Python is version 3, you may need to run `pip` instead. Make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!



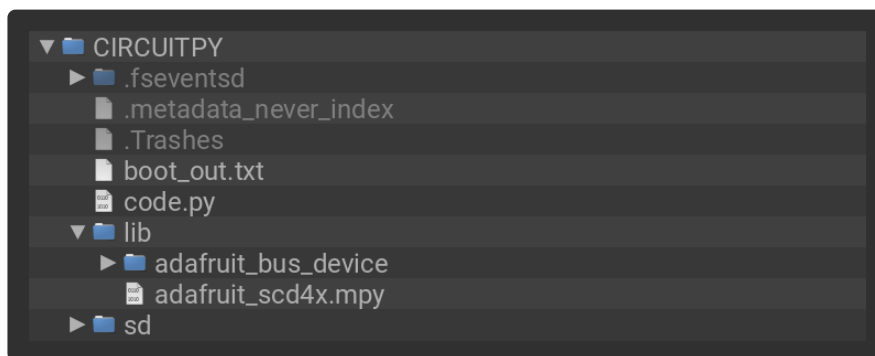
## CircuitPython Usage

To use with CircuitPython, you need to first install the SCD4x library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folder and file:

- **adafruit\_bus\_device/**
- **adafruit\_scd4x.mpy**



## Python Usage

Once you have the library **pip3** -installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file: **python3 code.py**

## Example Code:

```
# SPDX-FileCopyrightText: 2020 by Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time
import board
import adafruit_scd4x

i2c = board.I2C() # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C() # For using the built-in STEMMMA QT connector on a
# microcontroller
scd4x = adafruit_scd4x.SCD4X(i2c)
print("Serial number:", [hex(i) for i in scd4x.serial_number])
```

```

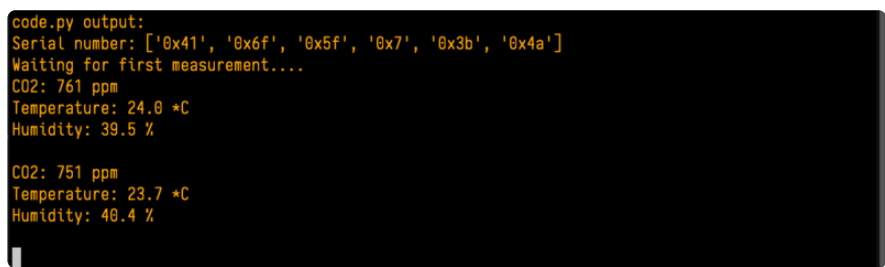
scd4x.start_periodic_measurement()
print("Waiting for first measurement....")

while True:
    if scd4x.data_ready:
        print("CO2: %d ppm" % scd4x.CO2)
        print("Temperature: %0.1f *C" % scd4x.temperature)
        print("Humidity: %0.1f %" % scd4x.relative_humidity)
        print()
        time.sleep(1)

```

If running **CircuitPython**: Once everything is saved to the **CIRCUITPY** drive, [connect to the serial console \(https://adafru.it/Bec\)](https://adafru.it/Bec) to see the data printed out!

If running **Python**: The console output will appear wherever you are running Python.



```

code.py output:
Serial number: ['0x41', '0x6f', '0x5f', '0x7', '0x3b', '0x4a']
Waiting for first measurement....
CO2: 761 ppm
Temperature: 24.0 *C
Humidity: 39.5 %

CO2: 751 ppm
Temperature: 23.7 *C
Humidity: 40.4 %

```

Here is a quick explanation of the code and library features.

First, you import the necessary modules and library, and initialize the I2C connection with the sensor. Next, you print the serial number.

Then, you start measuring data with `start_periodic_measurement()`.

Now you're ready to read values from the sensor using these properties:

- `data_ready` - Check the sensor to see if new data is available.
- `CO2` - The CO2 concentration in PPM (parts per million).
- `temperature` - The current temperature in degrees Celsius.
- `relative_humidity` - The current relative humidity in %rH.

That's all there is to using the SCD-40 and SCD-41 with CircuitPython!

## Python Docs

[Python Docs \(https://adafru.it/Uqa\)](https://adafru.it/Uqa)

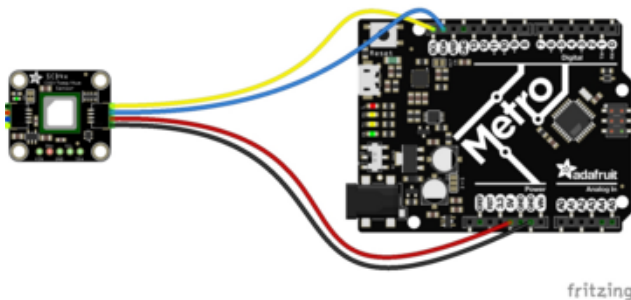
---

# Arduino

Using the SCD4x with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the [Adafruit SCD4x \(https://adafru.it/Uxc\)](https://adafru.it/Uxc) library written by Sensirion, and running the provided example code.

## I2C Wiring

Here is how to wire up the sensor using one of the [STEMMA QT \(https://adafru.it/Ft4\)](https://adafru.it/Ft4) connectors. The examples show a Metro but wiring will work the same for an Arduino or other compatible board.



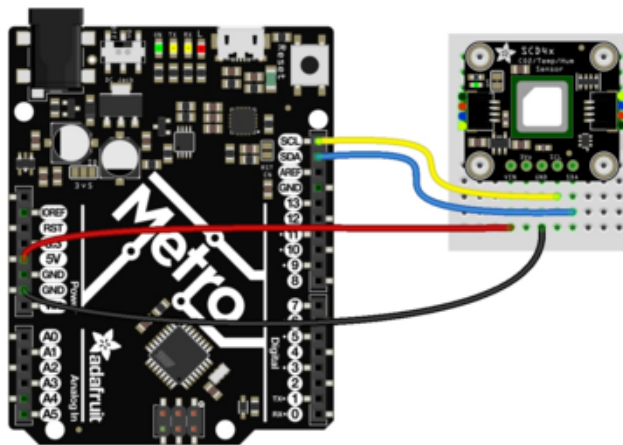
Connect **board VIN (red wire)** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V**, connect to that instead.

Connect **board GND (black wire)** to **Arduino GND**

Connect **board SCL (yellow wire)** to **Arduino SCL**

Connect **board SDA (blue wire)** to **Arduino SDA**

Here is how to wire the sensor to a board using a solderless breadboard:



Connect **board VIN (red wire)** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V**, connect to that instead.

Connect **board GND (black wire)** to **Arduino GND**

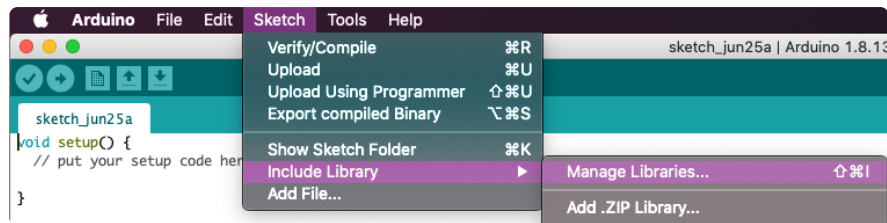
Connect **board SCL (yellow wire)** to **Arduino SCL**

Connect **board SDA (blue wire)** to **Arduino SDA**

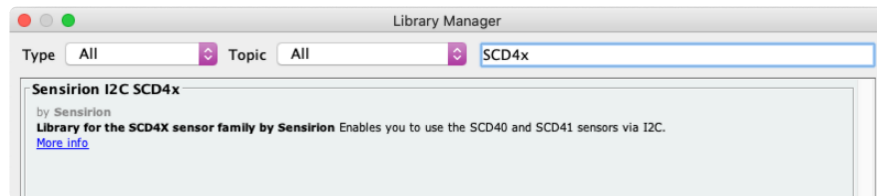
## Library Installation

You can install the **Sensirion I2C SCD4x** library for Arduino using the Library Manager in the Arduino IDE.

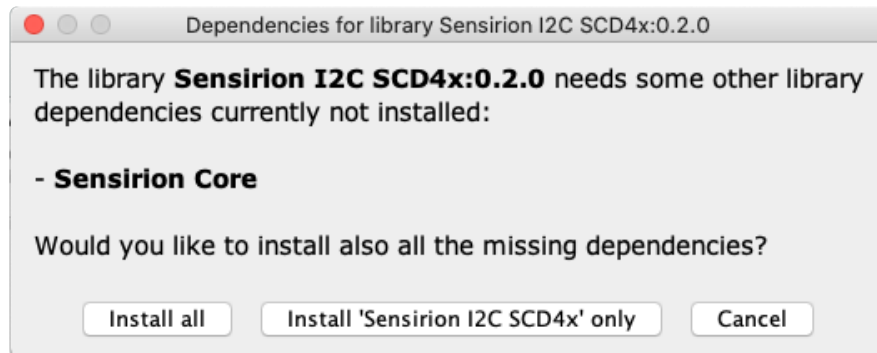




Click the **Manage Libraries ...** menu item, search for **SCD4x** , and select the **Sensirion I2C SCD4x** library:



If asked about dependencies, click "Install all".



## Load Example

Open up **File -> Examples -> Sensirion I2C SCD4x -> exampleusage**

```

/*
 * Copyright (c) 2021, Sensirion AG
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * * Redistributions of source code must retain the above copyright notice, this
 *   list of conditions and the following disclaimer.
 *
 * * Redistributions in binary form must reproduce the above copyright notice,
 *   this list of conditions and the following disclaimer in the documentation
 *   and/or other materials provided with the distribution.
 *
 * * Neither the name of Sensirion AG nor the names of its
 *   contributors may be used to endorse or promote products derived from
 *   this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE

```

```

* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*/

#include <Arduino.h>
#include <SensirionI2CScd4x.h>
#include <Wire.h>

SensirionI2CScd4x scd4x;

void printUInt16Hex(uint16_t value) {
    Serial.print(value < 4096 ? "0" : "");
    Serial.print(value < 256 ? "0" : "");
    Serial.print(value < 16 ? "0" : "");
    Serial.print(value, HEX);
}

void printSerialNumber(uint16_t serial0, uint16_t serial1, uint16_t serial2) {
    Serial.print("Serial: 0x");
    printUInt16Hex(serial0);
    printUInt16Hex(serial1);
    printUInt16Hex(serial2);
    Serial.println();
}

void setup() {
    Serial.begin(115200);
    while (!Serial) {
        delay(100);
    }

    Wire.begin();

    uint16_t error;
    char errorMessage[256];

    scd4x.begin(Wire);

    // stop potentially previously started measurement
    error = scd4x.stopPeriodicMeasurement();
    if (error) {
        Serial.print("Error trying to execute stopPeriodicMeasurement(): ");
        errorToString(error, errorMessage, 256);
        Serial.println(errorMessage);
    }

    uint16_t serial0;
    uint16_t serial1;
    uint16_t serial2;
    error = scd4x.getSerialNumber(serial0, serial1, serial2);
    if (error) {
        Serial.print("Error trying to execute getSerialNumber(): ");
        errorToString(error, errorMessage, 256);
        Serial.println(errorMessage);
    } else {
        printSerialNumber(serial0, serial1, serial2);
    }

    // Start Measurement
    error = scd4x.startPeriodicMeasurement();
    if (error) {
        Serial.print("Error trying to execute startPeriodicMeasurement(): ");
        errorToString(error, errorMessage, 256);
    }
}

```

```

        Serial.println(errorMessage);
    }

    Serial.println("Waiting for first measurement... (5 sec)");
}

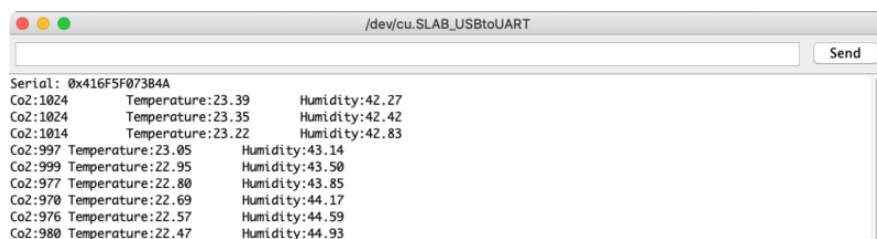
void loop() {
    uint16_t error;
    char errorMessage[256];

    delay(5000);

    // Read Measurement
    uint16_t co2;
    float temperature;
    float humidity;
    error = scd4x.readMeasurement(co2, temperature, humidity);
    if (error) {
        Serial.print("Error trying to execute readMeasurement(): ");
        errorToString(error, errorMessage, 256);
        Serial.println(errorMessage);
    } else if (co2 == 0) {
        Serial.println("Invalid sample detected, skipping.");
    } else {
        Serial.print("Co2:");
        Serial.print(co2);
        Serial.print("\t");
        Serial.print("Temperature:");
        Serial.print(temperature);
        Serial.print("\t");
        Serial.print("Humidity:");
        Serial.println(humidity);
    }
}

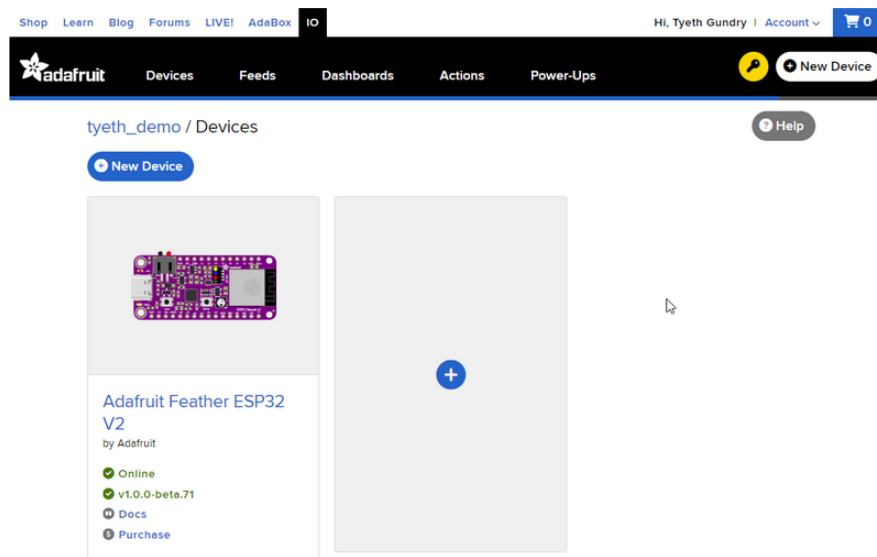
```

After opening the demo file, upload to your Arduino wired up to the sensor. Once you upload the code, you will see the **CO2, temperature, and relative humidity data** values being printed when you open the Serial Monitor (**Tools->Serial Monitor**) at **115200 baud**, similar to this:





# WipperSnapper



## What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to [Adafruit IO \(https://adafru.it/fsU\)](https://adafru.it/fsU), a web platform designed ([by Adafruit! \(https://adafru.it/Bo5\)](https://adafru.it/Bo5)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

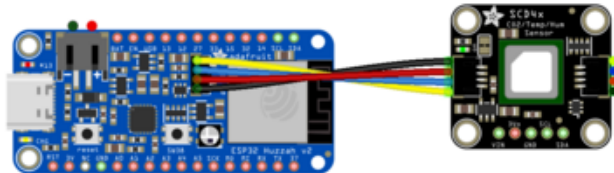
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

**Quickstart: Adafruit IO  
WipperSnapper**

<https://adafru.it/Vfd>

## Wiring

First, wire up a SCD-4x to your board exactly as follows. Here is an example of the SCD-4x wired to an [Adafruit ESP32 Feather V2](http://adafru.it/5400) (<http://adafru.it/5400>) using I2C [with a STEMMA QT cable](http://adafru.it/4210) (no soldering required) (<http://adafru.it/4210>)

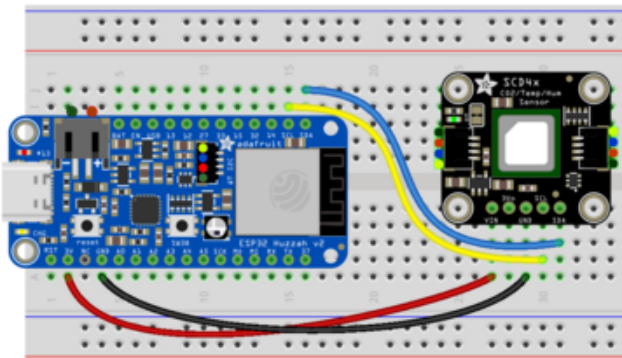


Board 3V to sensor VIN (red wire on STEMMA QT)

Board GND to sensor GND (black wire on STEMMA QT)

Board SCL to sensor SCL (yellow wire on STEMMA QT)

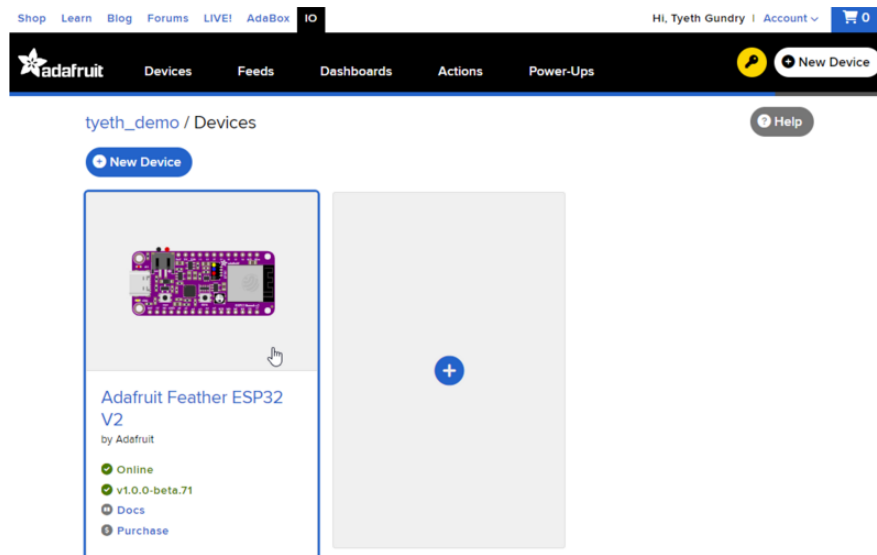
Board SDA to sensor SDA (blue wire on STEMMA QT)



## Usage

Connect your board to Adafruit IO Wippersnapper and [navigate to the WipperSnapper board list](https://adafru.it/TAu) (<https://adafru.it/TAu>).

On this page, **select the WipperSnapper board you're using** to be brought to the board's interface page.



If you do not see your board listed here - you need [to connect your board to Adafruit IO](https://adafru.it/Vfd) (<https://adafru.it/Vfd>) first.

## Adafruit Feather ESP32 V2

by Adafruit

✓ Online

✓ v1.0.0-beta.70

📖 Docs

💰 Purchase

## Adafruit Feather ESP32 V2

by Adafruit

✓ Online

! v1.0.0-beta.68 [↻ Update](#)

📖 Docs

💰 Purchase

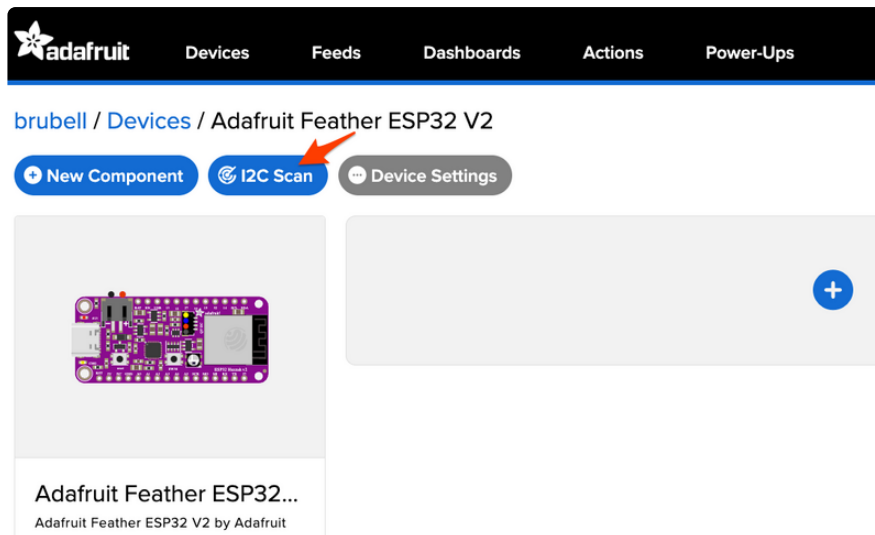
On the device page, quickly check that you're running the latest version of the WipperSnapper firmware.

The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide. If the firmware version is red with an "X" - [update to the latest WipperSnapper firmware](https://adafru.it/Vfd) (<https://adafru.it/Vfd>) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the **I2C Scan** button.





You should see the SCD-4x's default I2C address of **0x62** pop up in the I2C scan list.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00								--	--	--	--	--	--	--	--	--
10	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60	--	--	62	--	--	--	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## I don't see the sensor's I2C address listed!

First, double-check the connection and/or wiring between the sensor and the board.

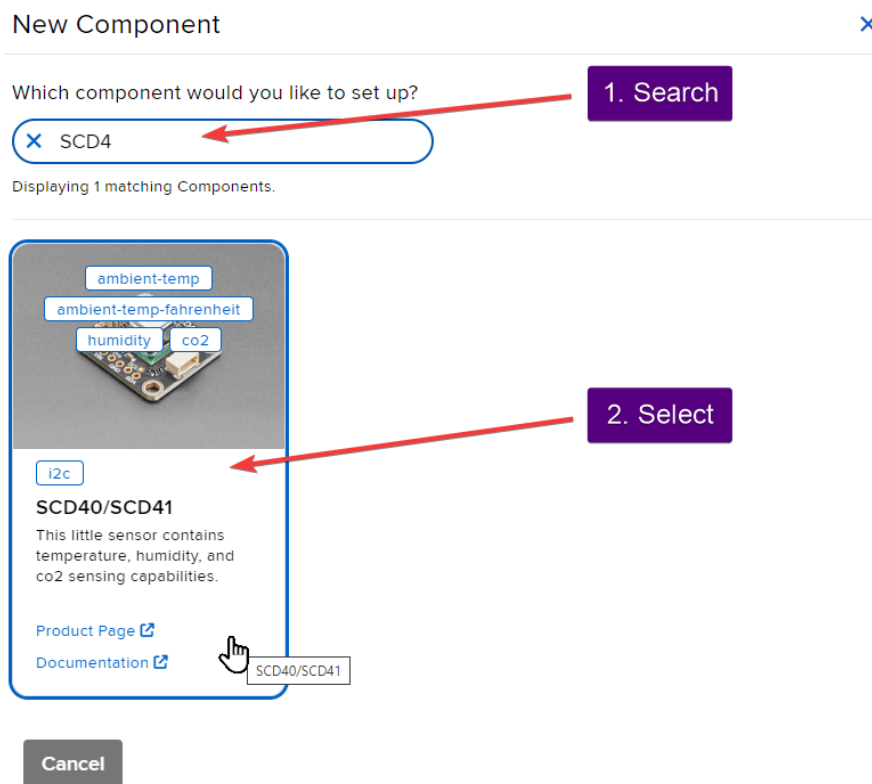
Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the **New Component** button or the **+** button to bring up the component picker.



Adafruit IO supports a large amount of components. To quickly find your sensor, type **SCD4** into the search bar, then select the **SCD-40/SCD-41** component.



On the component configuration page, the SCD-4x's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from each of the SCD-4x's three sensors and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

For this example, set the **Send Every** interval for each sensor to every 30 seconds.

### Create SCD40 Component



Select I2C Address:

0x62

☒ Enable SCD40: Temperature Sensor?

Name:

SCD40: Temperature Sensor

Send Every:

Every 30 seconds

☒ Enable SCD40: Humidity Sensor?

Name:

SCD40: Humidity Sensor

Send Every:

Every 30 seconds

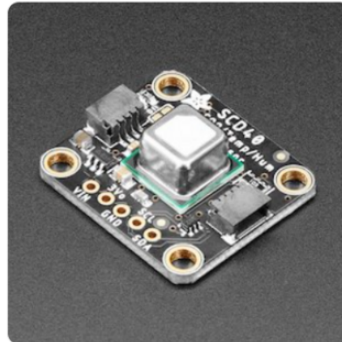
☒ Enable SCD40: CO2?

Name:

SCD40: CO2

Send Every:

Every 30 seconds



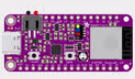
< Previous Step

Create Component

Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.

eherrada / Devices / Adafruit Feather ESP32 V2

New Component I2C Scan Device Settings Help



Adafruit Feather ESP32 V2  
by Adafruit

Online  
v1.0.0-beta.42  
Docs  
Purchase

SCD40: CO2 scd40.co2

Create Action | Add to Dashboard

731.00ppm

SCD40: Humidity Sensor scd40.humidity

Create Action | Add to Dashboard

42.20%

SCD40: Temperature Sensor scd40.ambient-temp

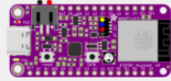
Create Action | Add to Dashboard

23.35°C

To view the data that has been logged from the sensor, click on the graph next to the sensor name.



eherrada / Devices / Adafruit Feather ESP32 V2

[New Component](#) [I2C Scan](#) [Device Settings](#) [Help](#)





Adafruit Feather ESP32 V2  
by Adafruit

[Online](#)  
[v1.0.0-beta.42](#)  
[Docs](#)  
[Purchase](#)

SCD40: CO2 scd40:co2  



700.00ppm

[Create Action](#) | [Add to Dashboard](#)

SCD40: Humidity Sensor scd40:humidity  

40.01%

[Create Action](#) | [Add to Dashboard](#)

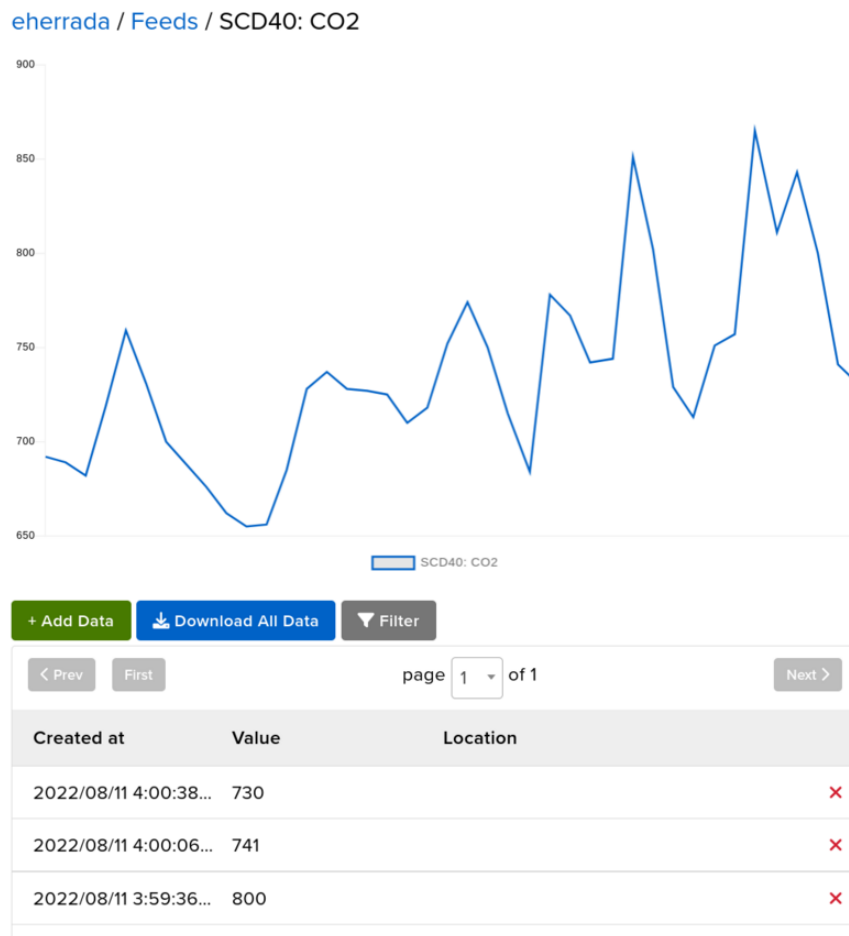
SCD40: Temperature Sensor scd40:ambient-temp  

23.16°C

[Create Action](#) | [Add to Dashboard](#)

Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, [check out this page \(https://adafru.it/10aZ\)](https://adafru.it/10aZ).

The SCD-4x has three sensors that each have their own feeds. In this picture, we're looking at the CO2 sensor, but if you click on the graph icon for the different sensors you'll see their feed history.





For IO Free accounts, feed data is stored for a maximum of 30 days and there's a maximum of 10 feeds. In this guide, you created three feeds (one for each of the SCD-4x's sensors). If you'd like to store data for more than 30 days, increase the number of feeds (components) you can use with WipperSnapper, or increase your data rate to send more sensor measurements to Adafruit IO - [upgrade your account to Adafruit IO Plus \(https://adafru.it/Eg3\)](https://adafru.it/Eg3).

---

## Downloads

### Files

- [SCD-4x datasheet \(https://adafru.it/Uxd\)](https://adafru.it/Uxd)
- [EagleCAD PCB files on GitHub \(https://adafru.it/Uxe\)](https://adafru.it/Uxe)
- [3D models on GitHub \(https://adafru.it/10fa\)](https://adafru.it/10fa)
- [Fritzing object in the Adafruit Fritzing Library \(https://adafru.it/Uxf\)](https://adafru.it/Uxf)

## Schematic and Fab Print

Fab print shows SCD-41 - the SCD-40 is the same!

