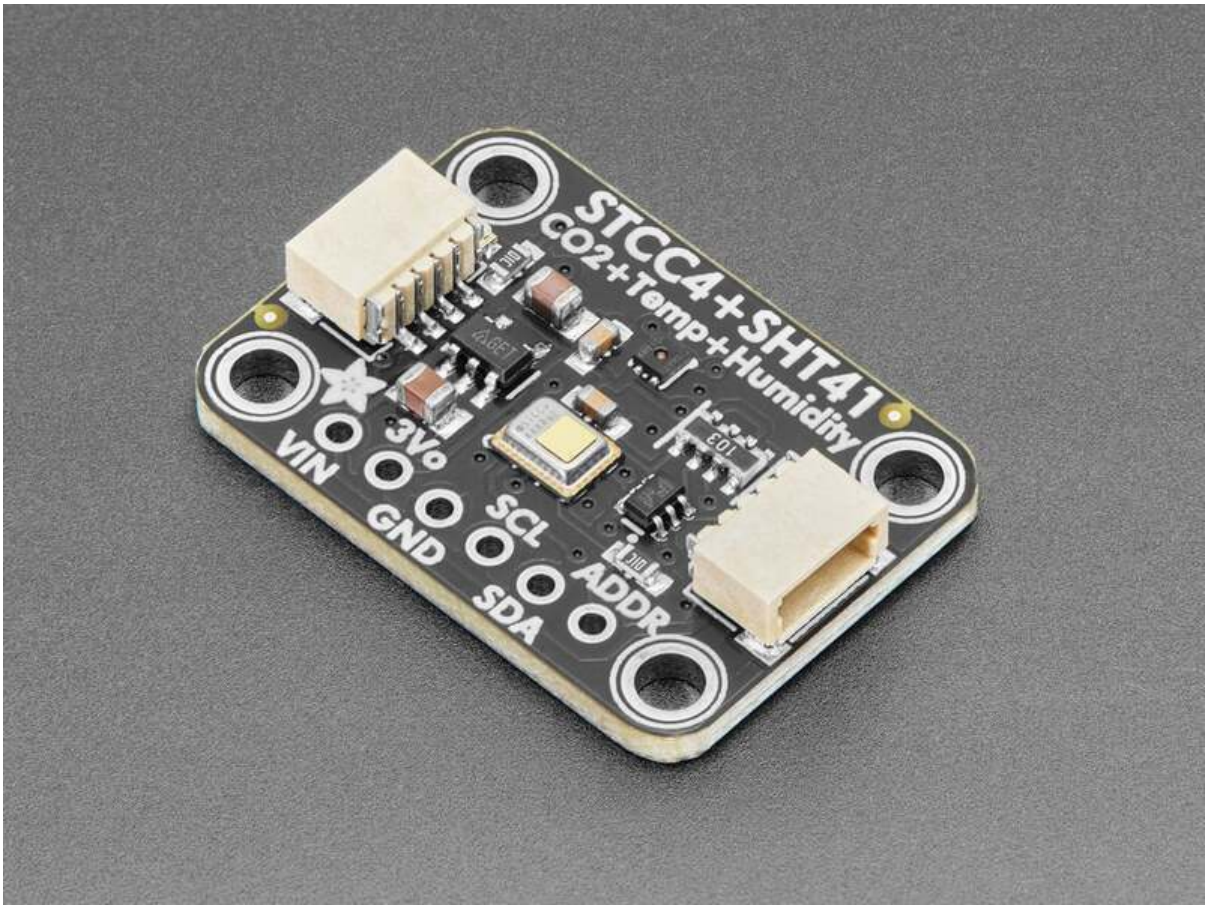
Adafruit STCC4 and SHT41 CO2, Temperature & Humidity Sensor

Created by Liz Clark



https://learn.adafruit.com/adafruit-stcc4-and-sht41-co2-temperature-humidity-sensor

Last updated on 2026-05-01 04:18:43 PM UTC

Table of Contents

Overview 3

Pinouts 6

- Power Pins
- I2C Logic Pins
- Address Pin and Jumper
- Power LED and Jumper

CircuitPython and Python 8

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of STCC4 Library
- CircuitPython Usage
- Python Usage
- Example Code

Python Docs 12

Arduino 12

- Wiring
- Library Installation
- Example Code

Arduino Docs 17

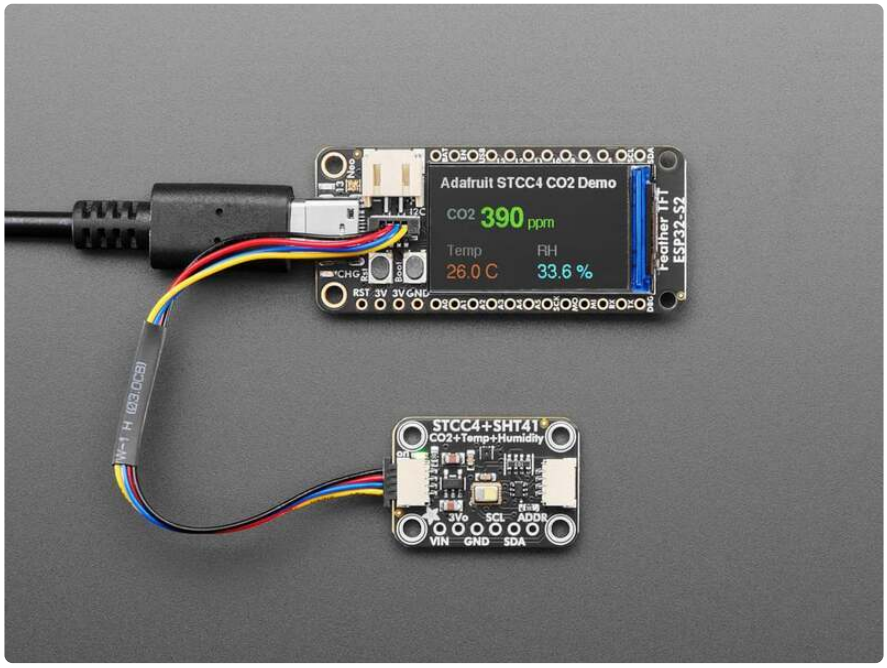
WipperSnapper 18

- What is WipperSnapper
- Wiring
- Usage

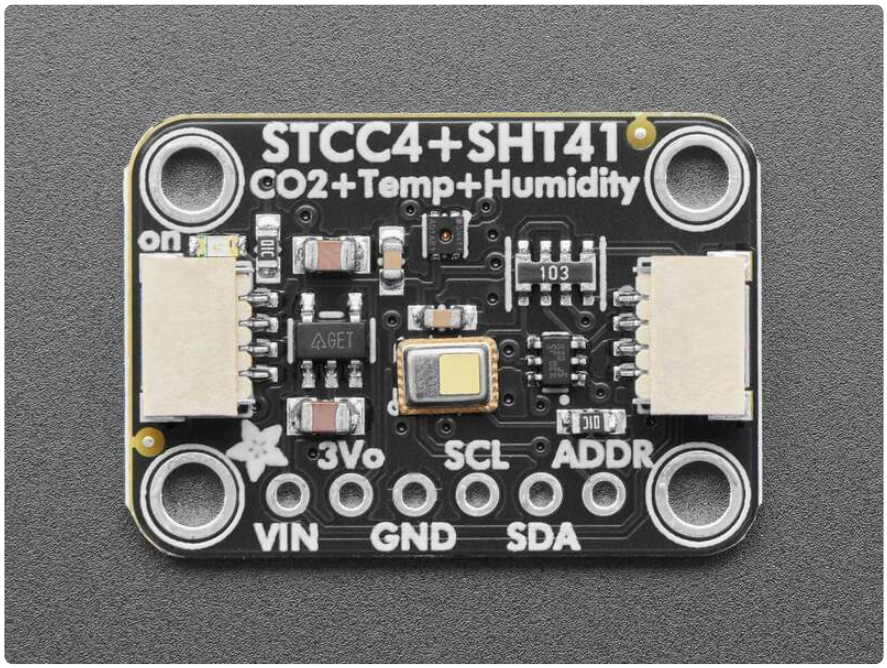
Downloads 25

- Files
- Schematic and Fab Print

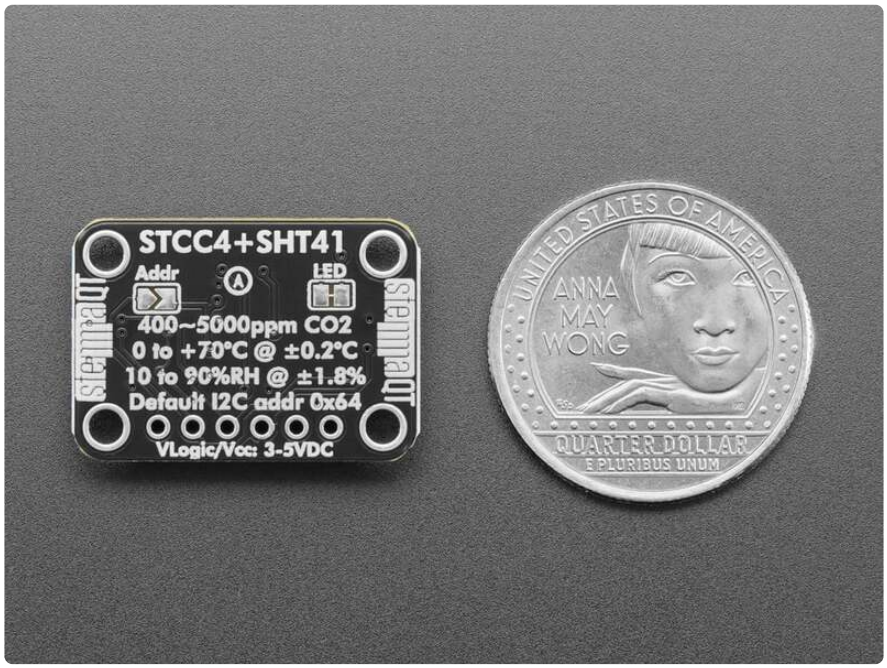
Overview



CO_2 sensors are essential for determining if a room is too 'stuffy' - high CO_2 makes humans grumpy and tired. That's why it's always nice to take a deep breath outside where CO_2 is about 400 ppm. However, until now, CO_2 sensors were always really chunky (like the SCD-30 (http://adafru.it/4867)) or pricey (like the SCD-40 (http://adafru.it/5187)). Inexpensive sensors like the SGP30 approximate the CO_2 using VOC gas concentration (http://adafru.it/3709), but they don't actually measure CO_2.

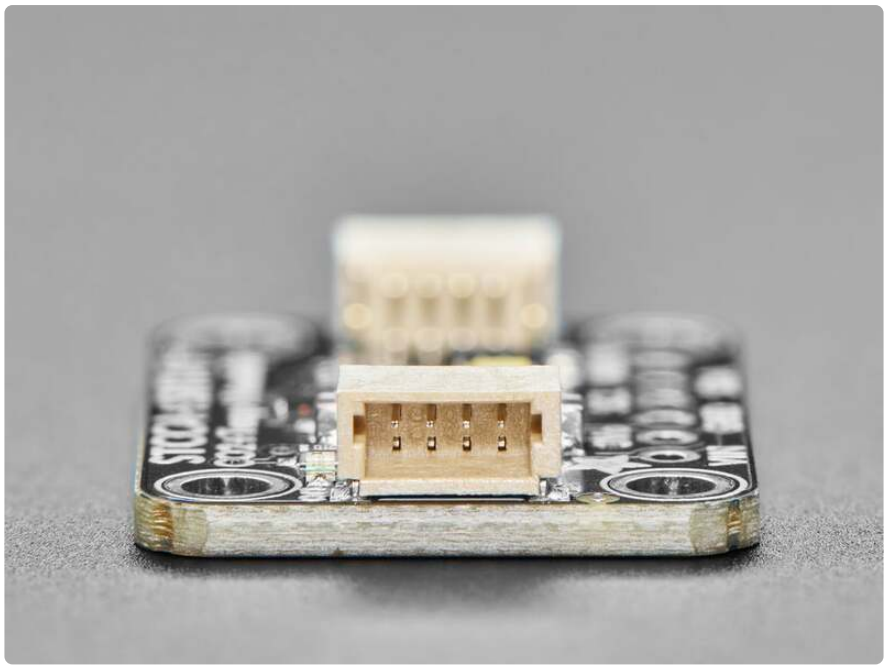


The STCC4 is a tiny (4 x 3 x 1.2mm) sensor that can measure CO_2 gas and fit in just about any enclosure. It uses thermal conductivity (TC) instead of NDIR or photoacoustic measurements. TC is based on the inherent thermal conductivity of all gases. With a thorough understanding of the gas composition in ambient environments, subtle changes in gas concentrations can be detected. The measurement principle is based on heating the air within a measurement cavity and sensing the heat transfer with a temperature sensor. [For more details check out the EYE ON NPI video we did on this sensor!](https://adafru.it/1aBg) (https://adafru.it/1aBg)



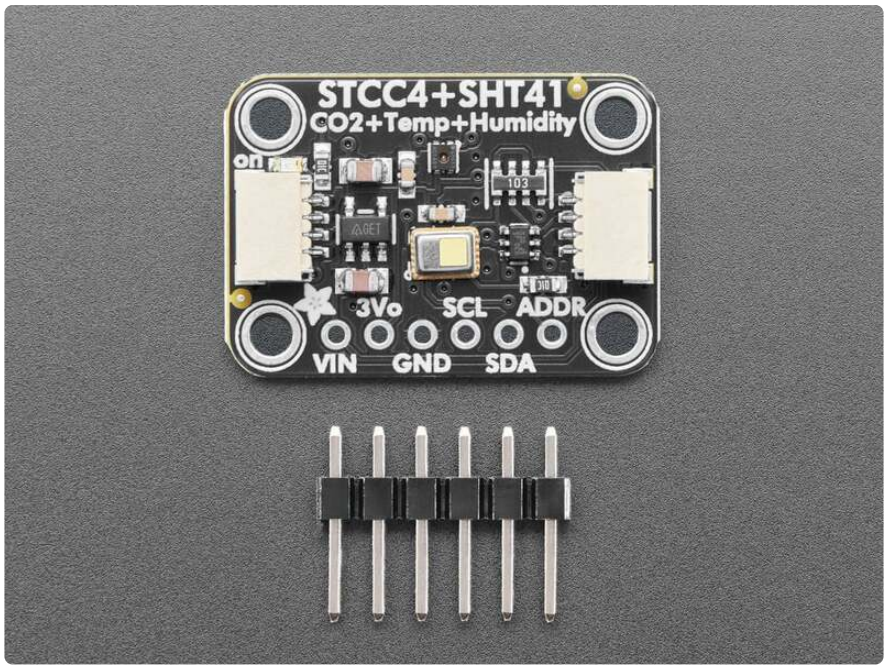
The trade-off for the small size and lower cost is that STCC4 is that it's really tuned just for indoor air measurement applications - not for scientific CO_2 sensing or where the air gas composition is 'abnormal'. That said, the vast majority of use cases we see are indoor air monitoring to reduce illness and stuffiness and for that purpose the STCC4 is great!

One thing to note, like all true CO_2 sensors, it must be self-calibrated by having it exposed to outdoor air once a week, to get a baseline 400 ppm measurement - in practice that means opening a window or making sure fresh air gets to the sensor.



To round the STCC4 out and make it a perfect mini indoor air quality sensor, we've added an SHT41 as well (http://adafru.it/5776) to improve the CO_2 measurements. The SHT41 has an excellent ±1.8% typical relative humidity accuracy from 25 to 75% and ±0.2 °C typical accuracy from 0 to 75 °C. Note it is connected to the secondary I2C port on the STCC4 so that you only have to query the STCC4 to get measurements rather than talk to both chips with two libraries.

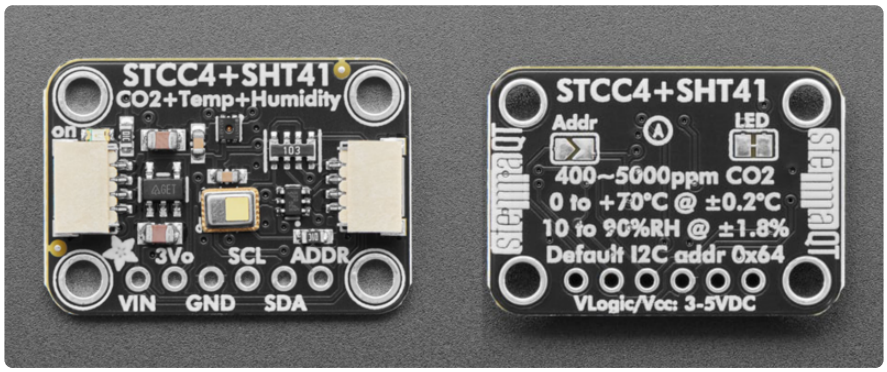
Such a lovely pair of chips - so we spun up a breakout board with the STCC4 + SHT41 and some supporting circuitry such as pullup resistors and capacitors. To make things even easier, we've included SparkFun Qwiic (https://adafru.it/Fpw) compatible STEMMA QT (https://adafru.it/Ft4) connectors for the I2C bus so you don't even need to solder! QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE).



If you prefer working on a breadboard, each order comes with one fully assembled and tested PCB breakout and a small piece of header. You'll need to solder the header onto the PCB, but it's fairly easy and takes only a few minutes even for a beginner.

We've written both Arduino and CircuitPython/Python library code for this chip, so you can use it with just about any microcontroller or single-board computer like Raspberry Pi.

Pinouts



The default I2C address is **0x64**.

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 5V microcontroller like Arduino, use 5V. For a 3.3V microcontroller, use 3.3V.
- **3Vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like.
- **GND** - common ground for power and logic.

I2C Logic Pins

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- STEMMA QT **(https://adafru.it/Ft4)** - These connectors allow you to connect to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6).

Address Pin and Jumper

- **ADDR** - The address pin for setting the I2C address. You can chain up to two of these boards together on one I2C bus. Leave this pin **low** for default I2C address **0x64** or tie it **high** for I2C address **0x65**.
- **Addr Jumper** - On the back of the board is the address jumper, labeled **Addr**. You can leave this jumper open (low) to keep the board on the default I2C address 0x64. Solder the jumper closed (high) to change the I2C address to 0x65.

ADDR	ADDR
0x64	L
0x65	H

Power LED and Jumper

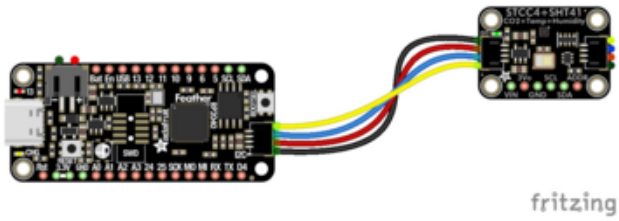
- **Power LED -** In the upper left corner, above the STEMMA QT connector, on the front of the board, is the power LED, labeled **on**. It is a green LED.
- **LED jumper** - This jumper is located on the back of the board and is labeled **LED** on the board silk. Cut the trace on this jumper to cut power to the "**on**" LED.

CircuitPython and Python

It's easy to use the **STCC4** with Python or CircuitPython, and the Adafruit_CircuitPython_STCC4 (https://adafru.it/1aBh) module. This module allows you to easily write Python code that allows you to read the **STCC4** CO2, temperature and humidity 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).

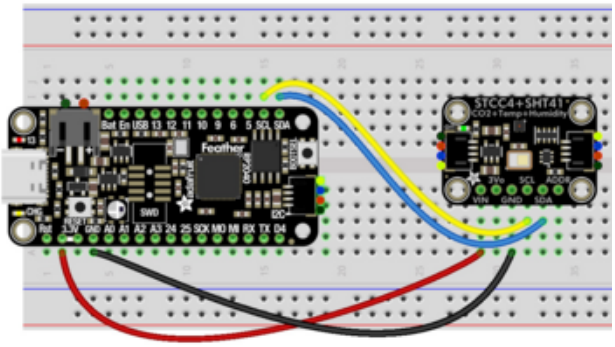
CircuitPython Microcontroller Wiring

First, wire up an STCC4 to your board exactly as shown below. Here's an example of wiring a Feather RP2040 to the STCC4 with I2C using one of the handy **STEMMA QT** (https://adafru.it/Ft4) connectors:



Board STEMMA 3V to **breakout STEMMA VIN (red wire)**
Board STEMMA GND to **breakout STEMMA GND (black wire)**
Board STEMMA SCL to **breakout STEMMA SCL (yellow wire)**
Board STEMMA SDA to **breakout STEMMA SDA (blue wire)**

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

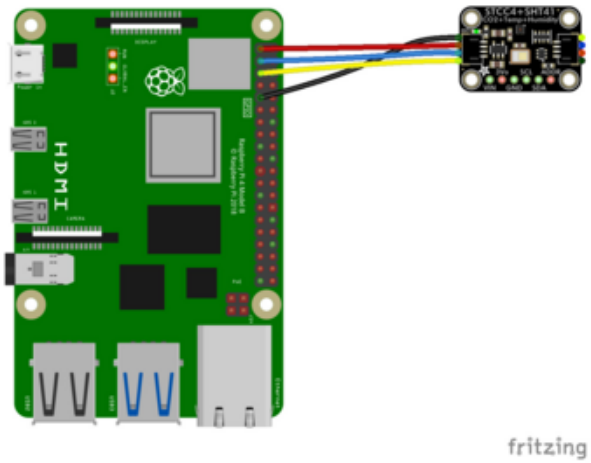


Board 3V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**

Python Computer Wiring

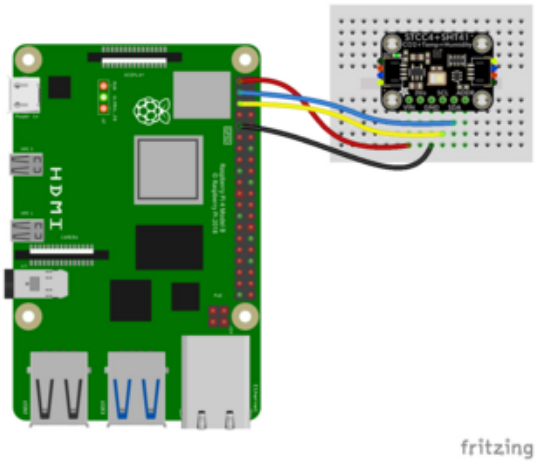
Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (https://adafru.it/BSN).

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



Pi 3V to **breakout STEMMA VIN (red wire)**
Pi GND to **breakout STEMMA GND (black wire)**
Pi SCL to **breakout STEMMA SCL (yellow wire)**
Pi SDA to **breakout STEMMA 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 **breakout VIN (red wire)**
Pi GND to **breakout GND (black wire)**
Pi SCL to **breakout SCL (yellow wire)**
Pi SDA to **breakout SDA (blue wire)**

Python Installation of STCC4 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)!

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

- `pip3 install adafruit-circuitpython-stcc4`

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

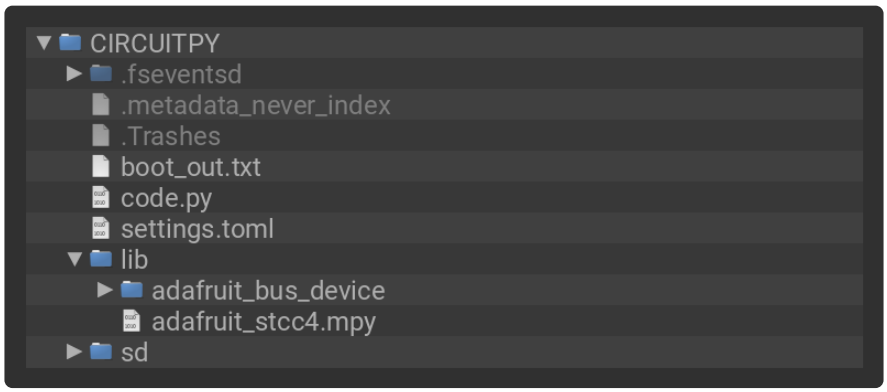
To use with CircuitPython, you need to first install the STCC4 library and its dependencies into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

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_stcc4.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

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

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

```
# SPDX-FileCopyrightText: Copyright (c) 2026 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time

import board

import adafruit_stcc4

i2c = board.I2C()
sensor = adafruit_stcc4.STCC4(i2c)
print("Starting continuous measurement...")
sensor.continuous_measurement = True

while True:
    co2 = sensor.CO2
    print(
        f"CO2: {co2} ppm | "
```

```
        f"Temperature: {sensor.temperature:.1f} °C | "
        f"Humidity: {sensor.relative_humidity:.1f} %"
    )
    time.sleep(1)
```

https://github.com/adafruit/Adafruit_CircuitPython_STCC4/blob/main/examples/stcc4_simpletest.py

In the example, the STCC4 sensor is instantiated on I2C. Then, in the loop, the CO2, temperature and humidity readings are printed to the serial console every second.

```
CO2: 808 ppm | Temperature: 23.7 °C | Humidity: 38.9 %
CO2: 805 ppm | Temperature: 23.7 °C | Humidity: 39.1 %
CO2: 796 ppm | Temperature: 23.7 °C | Humidity: 39.0 %
CO2: 808 ppm | Temperature: 23.7 °C | Humidity: 39.0 %
CO2: 816 ppm | Temperature: 23.7 °C | Humidity: 39.1 %
CO2: 833 ppm | Temperature: 23.7 °C | Humidity: 39.3 %
CO2: 835 ppm | Temperature: 23.7 °C | Humidity: 39.5 %
CO2: 836 ppm | Temperature: 23.7 °C | Humidity: 39.5 %
CO2: 850 ppm | Temperature: 23.7 °C | Humidity: 39.3 %
CO2: 856 ppm | Temperature: 23.7 °C | Humidity: 39.4 %
CO2: 860 ppm | Temperature: 23.7 °C | Humidity: 39.6 %
CO2: 863 ppm | Temperature: 23.7 °C | Humidity: 39.7 %
CO2: 859 ppm | Temperature: 23.7 °C | Humidity: 39.7 %
CO2: 859 ppm | Temperature: 23.7 °C | Humidity: 39.7 %
CO2: 861 ppm | Temperature: 23.7 °C | Humidity: 39.7 %
```

Python Docs

Python Docs (https://adafru.it/1aB0)

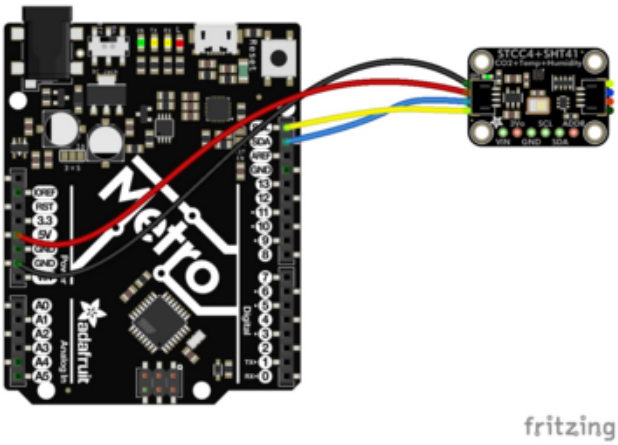
Arduino

Using the STCC4 CO2, temperature and humidity sensor with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit_STCC4 (https://adafru.it/1aBi) library and running the provided example code.

Wiring

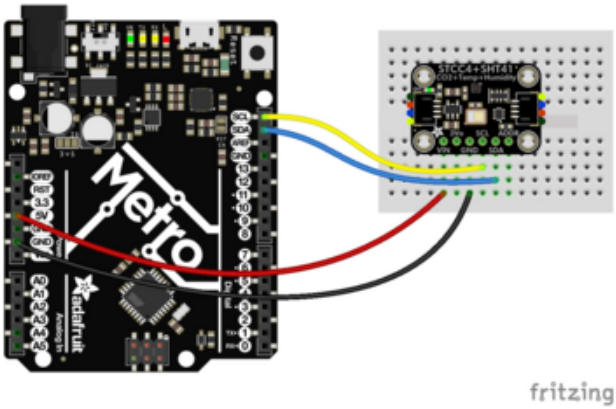
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the STCC4 VIN.

Here is an Adafruit Metro wired up to the STCC4 using the STEMMA QT connector:



Board 5V to **breakout STEMMA VIN (red wire)**
Board GND to **breakout STEMMA GND (black wire)**
Board SCL to **breakout STEMMA SCL (yellow wire)**
Board SDA to **breakout STEMMA SDA (blue wire)**

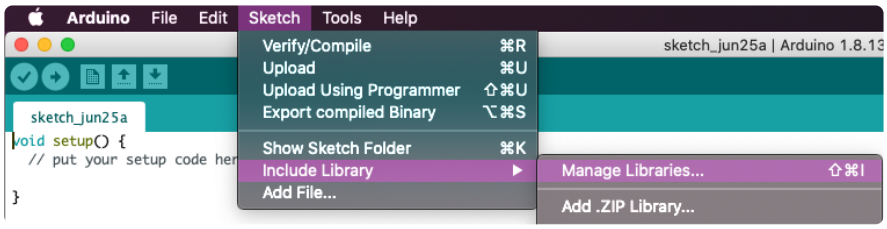
Here is an Adafruit Metro wired up using a solderless breadboard:



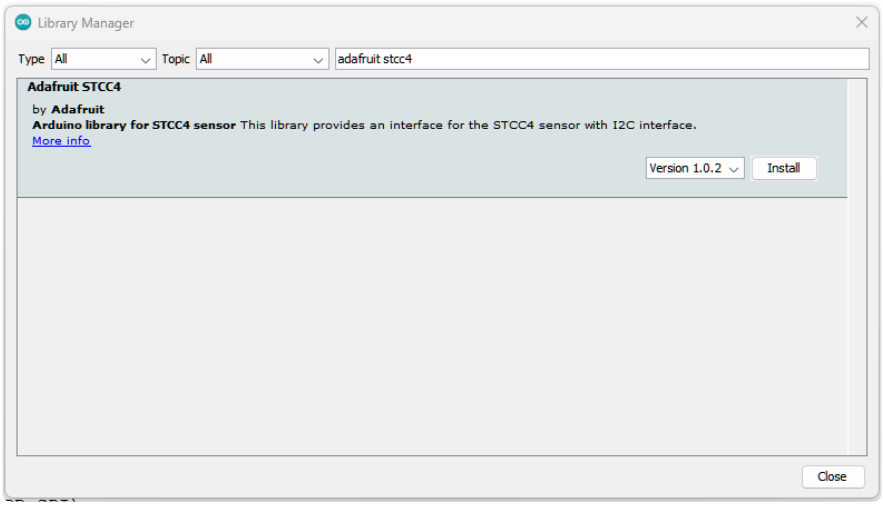
Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**

Library Installation

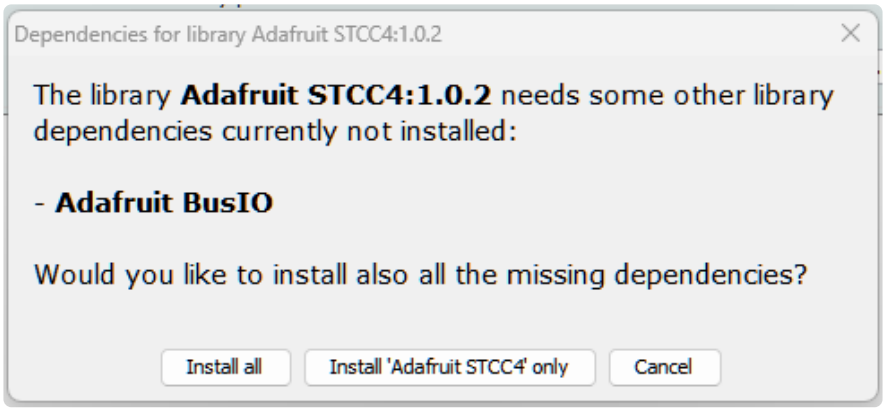
You can install the **Adafruit STCC4** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit STCC4**, and select the **Adafruit STCC4** library:



If asked about dependencies, click "**Install all**".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

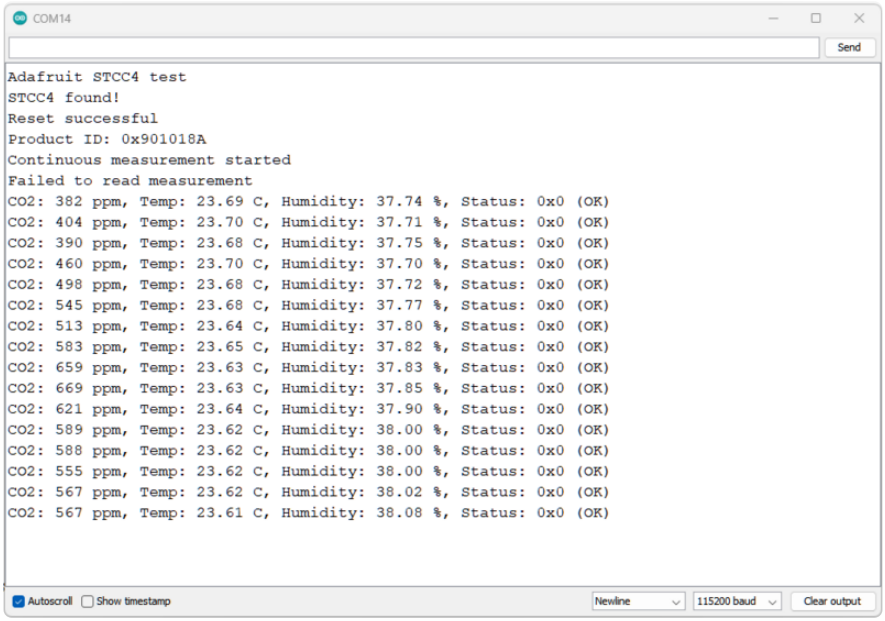
```
1  // Continuous measurement example for Adafruit STCC4 CO2 sensor
2
3  #include <Adafruit_STCC4.h>
```

```
4
5   Adafruit_STCC4 stcc4;
6
7   void printStatus(uint16_t status) {
8     Serial.print(F("Status: 0x"));
9     Serial.print(status, HEX);
10    Serial.print(F(" ("));
11
12    bool first = true;
13    if (status & STCC4_STATUS_VOLTAGE_ERROR) {
14      if (!first) Serial.print(F(", "));
15      Serial.print(F("VOLTAGE_ERROR"));
16      first = false;
17    }
18    if (status & STCC4_STATUS_DEBUG_MASK) {
19      if (!first) Serial.print(F(", "));
20      Serial.print(F("DEBUG"));
21      first = false;
22    }
23    if (status & STCC4_STATUS_SHT_NOT_CONNECTED) {
24      if (!first) Serial.print(F(", "));
25      Serial.print(F("SHT_NOT_CONNECTED"));
26      first = false;
27    }
28    if (status & STCC4_STATUS_MEMORY_ERROR_MASK) {
29      if (!first) Serial.print(F(", "));
30      Serial.print(F("MEMORY_ERROR"));
31      first = false;
32    }
33    if (status & STCC4_STATUS_TESTING_MODE) {
34      if (!first) Serial.print(F(", "));
35      Serial.print(F("TESTING_MODE"));
36      first = false;
37    }
38    if (first) {
39      Serial.print(F("OK"));
40    }
41    Serial.println(F(")"));
42  }
43
44  void setup() {
45    Serial.begin(115200);
46    while (!Serial) delay(10);
47
48    Serial.println(F("Adafruit STCC4 test"));
49
50    if (!stcc4.begin()) {
51      Serial.println(F("Failed to find STCC4 chip"));
52      while (1) delay(10);
53    }
54
55    Serial.println(F("STCC4 found!"));
56
57    if (!stcc4.reset()) {
58      Serial.println(F("Failed to reset STCC4"));
59      while (1) delay(10);
```

```
  }
  Serial.println(F("Reset successful"));

  // Test getProductID function after reset
  uint32_t productID = stcc4.getProductID();
  Serial.print(F("Product ID: 0x"));
  Serial.println(productID, HEX);

  // Uncomment to perform factory reset (clears calibration history)
  // Serial.println(F("Performing factory reset..."));
  // if (stcc4.factoryReset()) {
  //   Serial.println(F("Factory reset complete"));
  // } else {
  //   Serial.println(F("Factory reset failed"));
  // }

  if (!stcc4.enableContinuousMeasurement(true)) {
    Serial.println(F("Failed to start continuous measurement"));
    while (1) delay(10);
  }
  Serial.println(F("Continuous measurement started"));

  // Uncomment to perform conditioning (takes 22 seconds)
  // Serial.println(F("Performing conditioning..."));
  // if (stcc4.performConditioning()) {
  //   Serial.println(F("Conditioning complete"));
  // } else {
  //   Serial.println(F("Conditioning failed"));
  // }
}

void loop() {
  uint16_t co2;
  float temperature, humidity;
  uint16_t status;

  if (stcc4.readMeasurement(&co2, &temperature, &humidity, &status)) {
    Serial.print(F("CO2: "));
    Serial.print(co2);
    Serial.print(F(" ppm, Temp: "));
    Serial.print(temperature);
    Serial.print(F(" C, Humidity: "));
    Serial.print(humidity);
    Serial.print(F(" %, "));
    printStatus(status);
  } else {
    Serial.println(F("Failed to read measurement"));
  }

  delay(1000);
}
```

https://github.com/adafruit/Adafruit_STCC4/blob/main/examples/continuous_stcc4/continuous_stcc4.ino

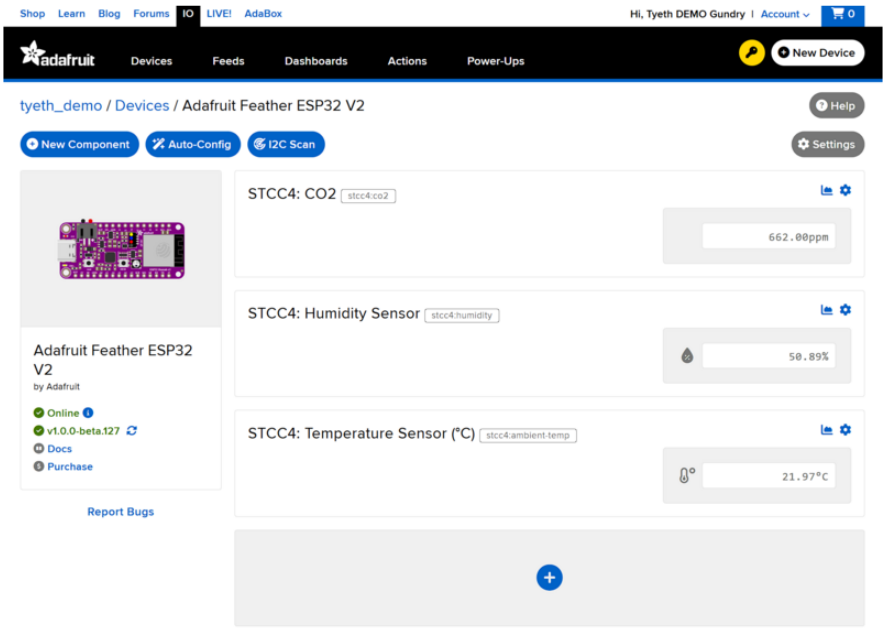
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You should see that the sketch has found your connected STCC4 sensor. Then, you'll see the CO2, temperature, humidity and status readings printed to the Serial Monitor every second.



Arduino Docs

Arduino Docs (https://adafru.it/1aB1)

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), a web platform designed (by Adafruit! (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.

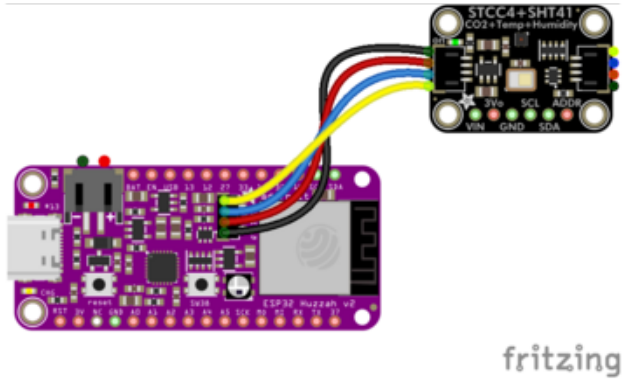
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

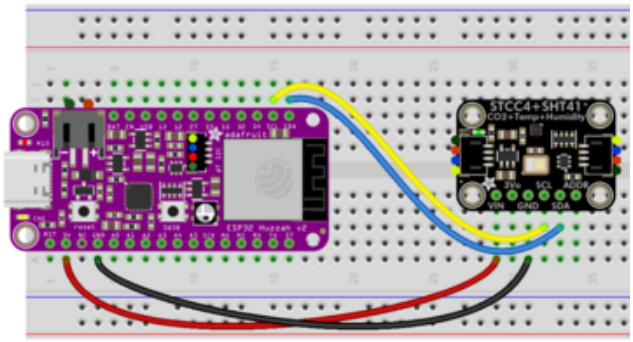
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

https://adafru.it/Vfd

Wiring

Wire up the STCC4 exactly as follows. Here it's shown connected via our convenient StemmaQT cable, or alternatively wired up on a prototyping breadboard:



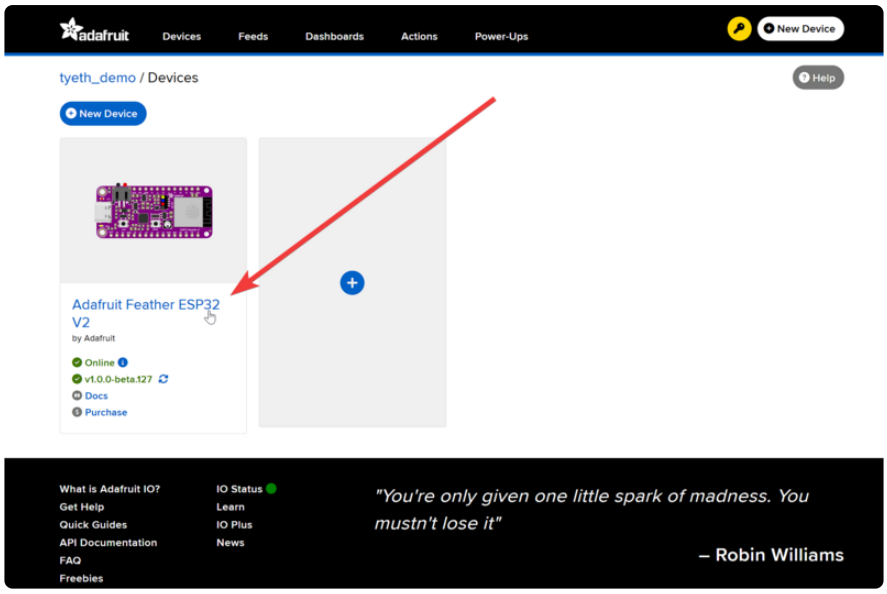


Board 5V to **breakout STEMMA VIN (red wire)**
Board GND to **breakout STEMMA GND (black wire)**
Board SCL to **breakout STEMMA SCL (yellow wire)**
Board SDA to **breakout STEMMA SDA (blue wire)**

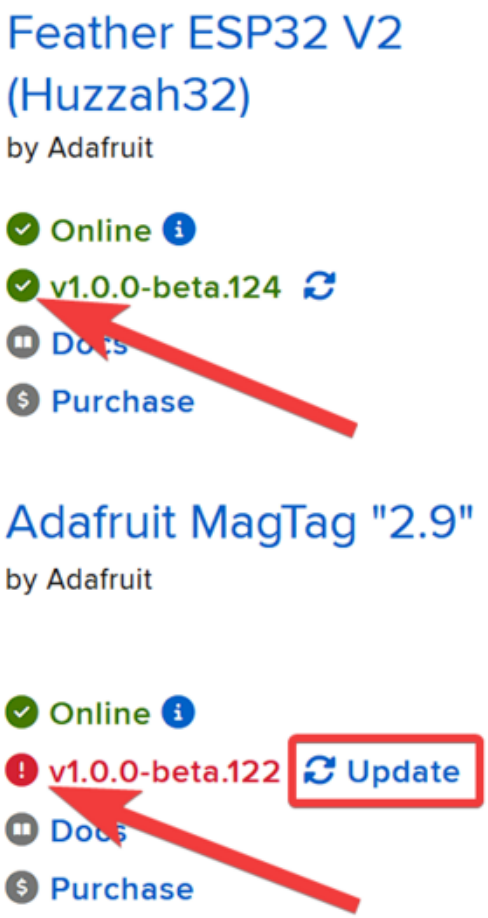
Usage

Connect your board to Adafruit IO Wippersnapper and navigate to the WipperSnapper board list **(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)](#) first.

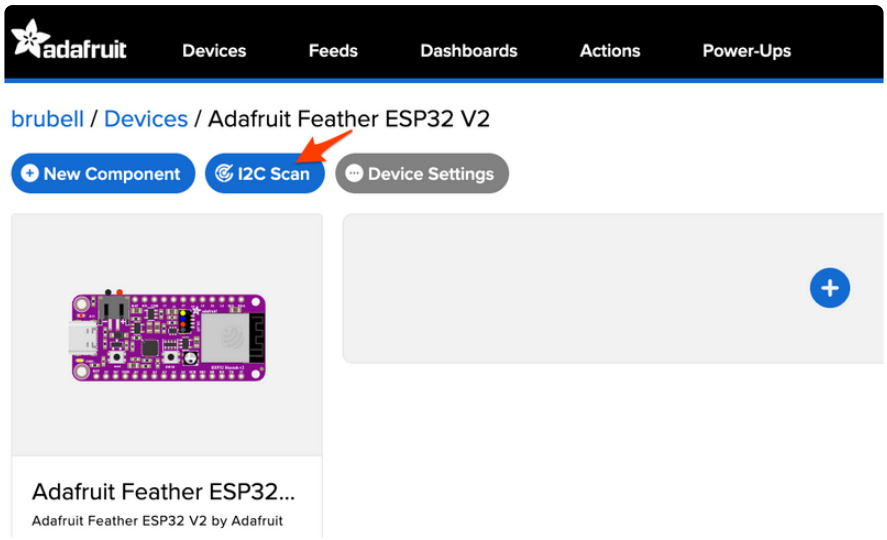


On the device page, quickly **check that you're running the latest version of the WipperSnapper firmware**.

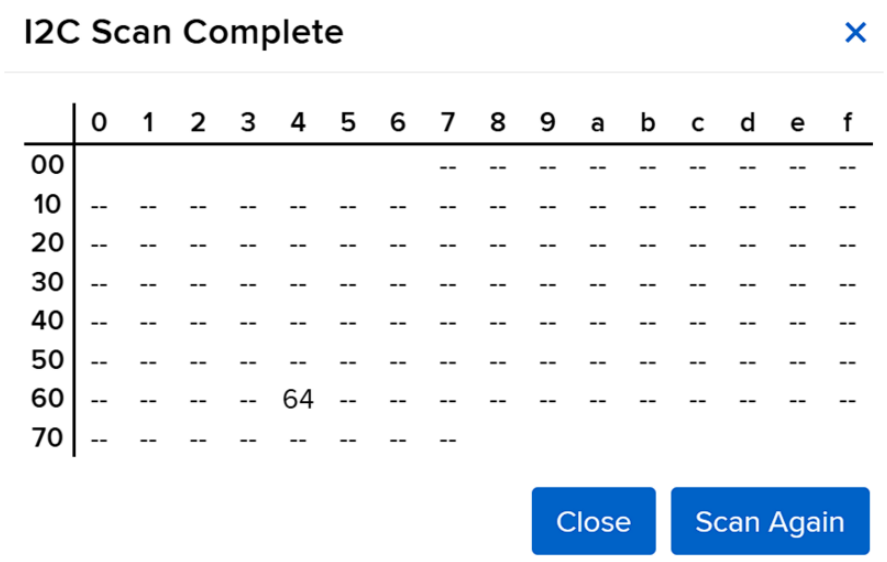
The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide.
If the firmware version is red with an exclamation mark "!" - [update to the latest WipperSnapper firmware (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 STCC4's default I2C address of `0x64` pop-up in the I2C scan list.



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

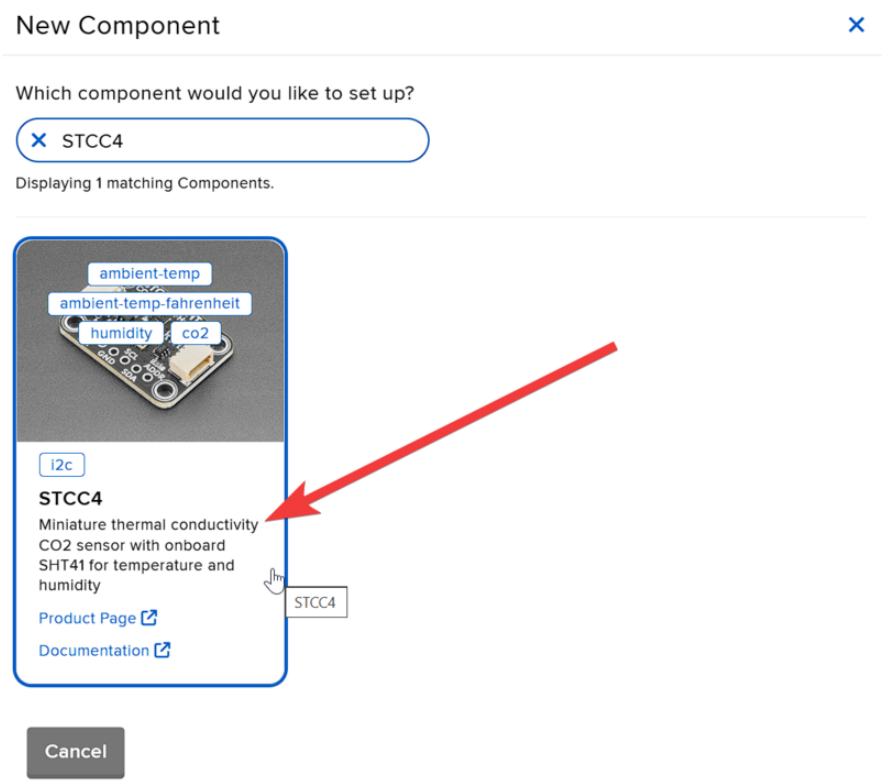
First, double-check the connection and/or wiring between the sensor and the board.

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.

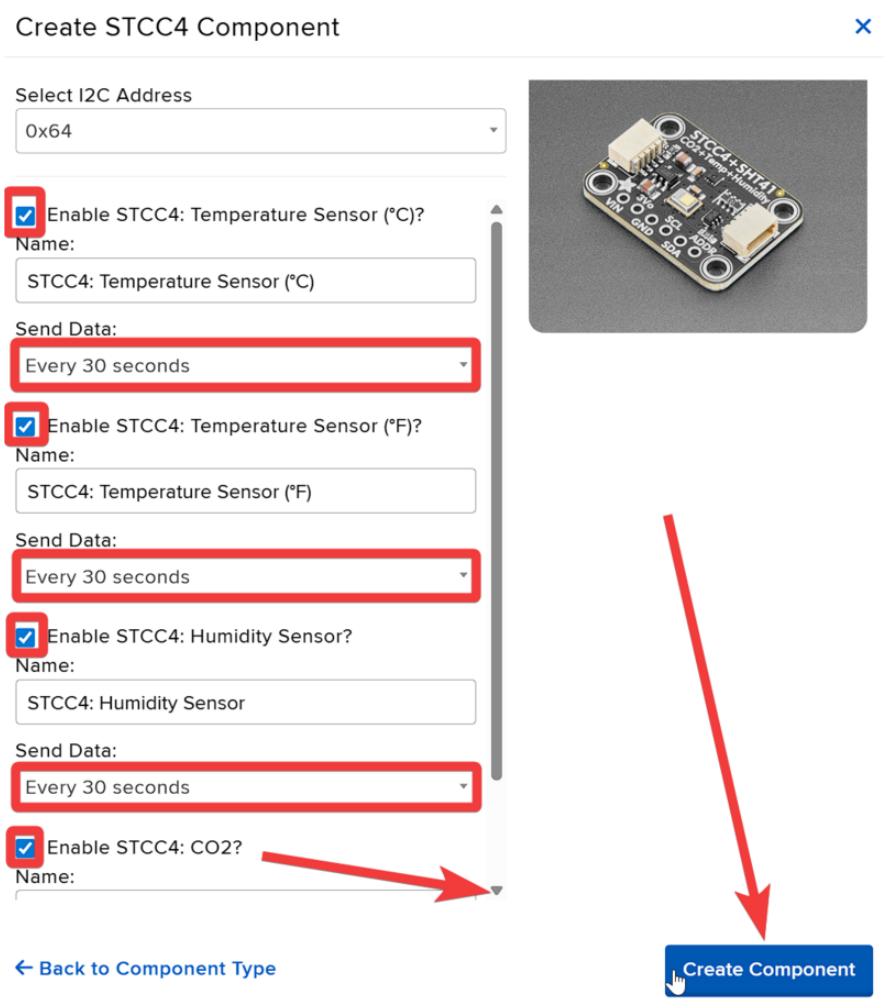
Adafruit IO supports a large amount of components. To quickly find your sensor, type `STCC4` into the search bar, then select the **STCC4** component.



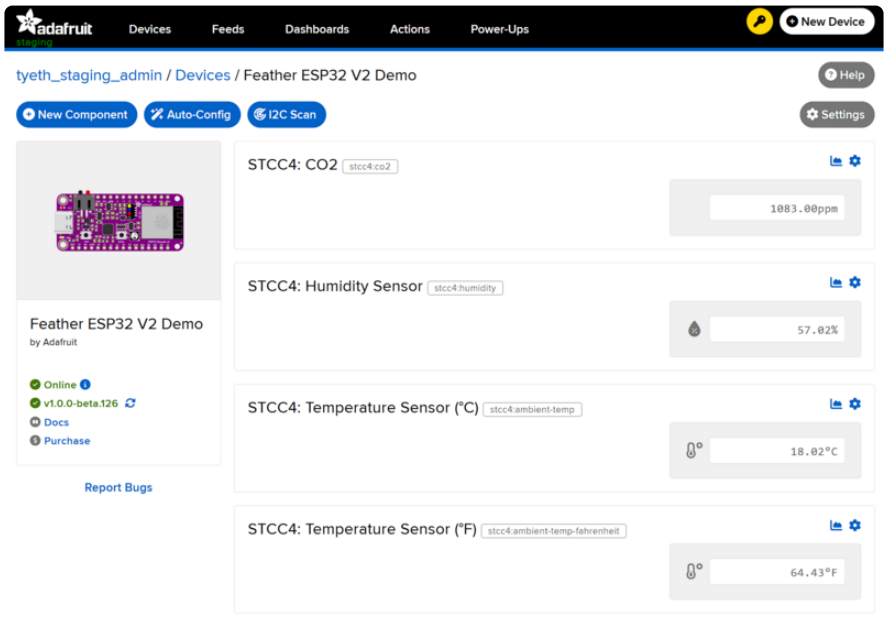
On the component configuration page, the STCC4's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the STCC4 sensor and send the data to Adafruit IO. Measurements can range from every second to every 24 hours.

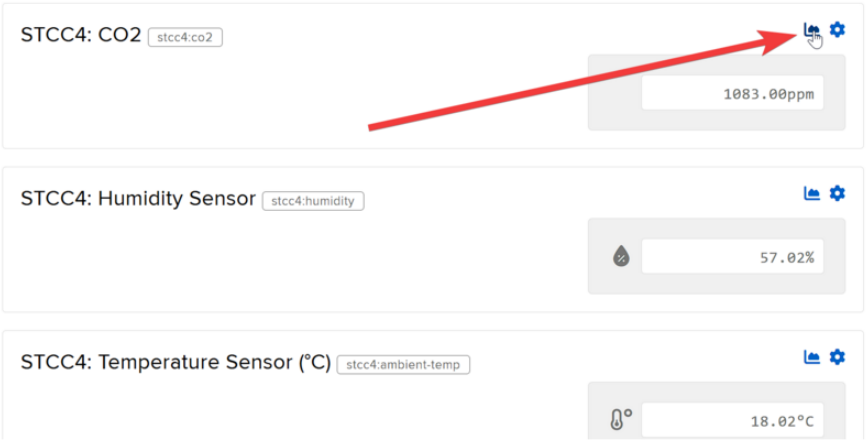
For this example, set the **Send Every** interval to every 30 seconds (scroll down to see the CO2 setting).



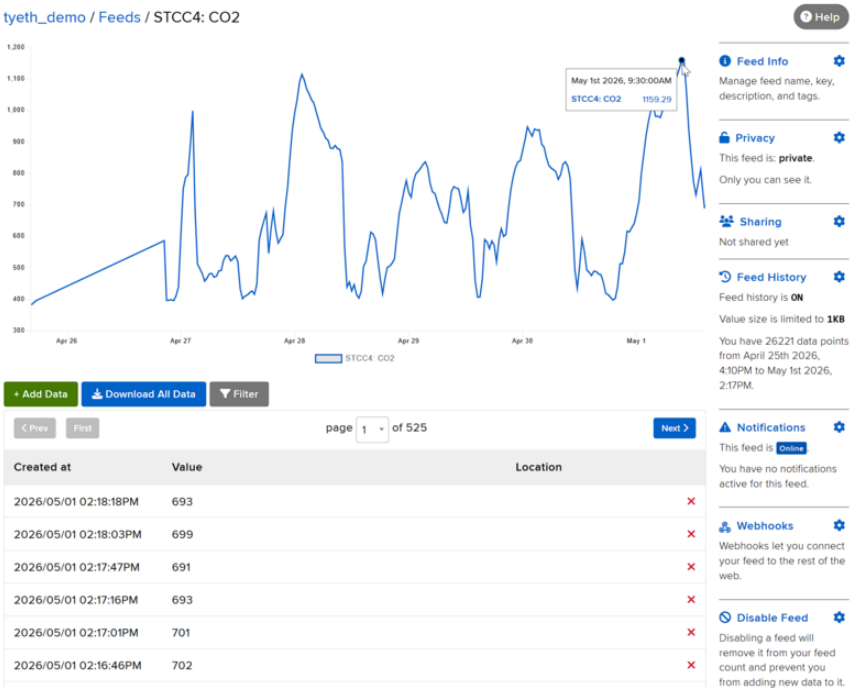
Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.



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



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).



Downloads

Files

- [STCC4 Datasheet](https://adafru.it/1aBj)

- SHT41 Datasheet (https://adafru.it/1aBk)
- EagleCAD PCB files on GitHub (https://adafru.it/1aBl)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1aBm)

Schematic and Fab Print