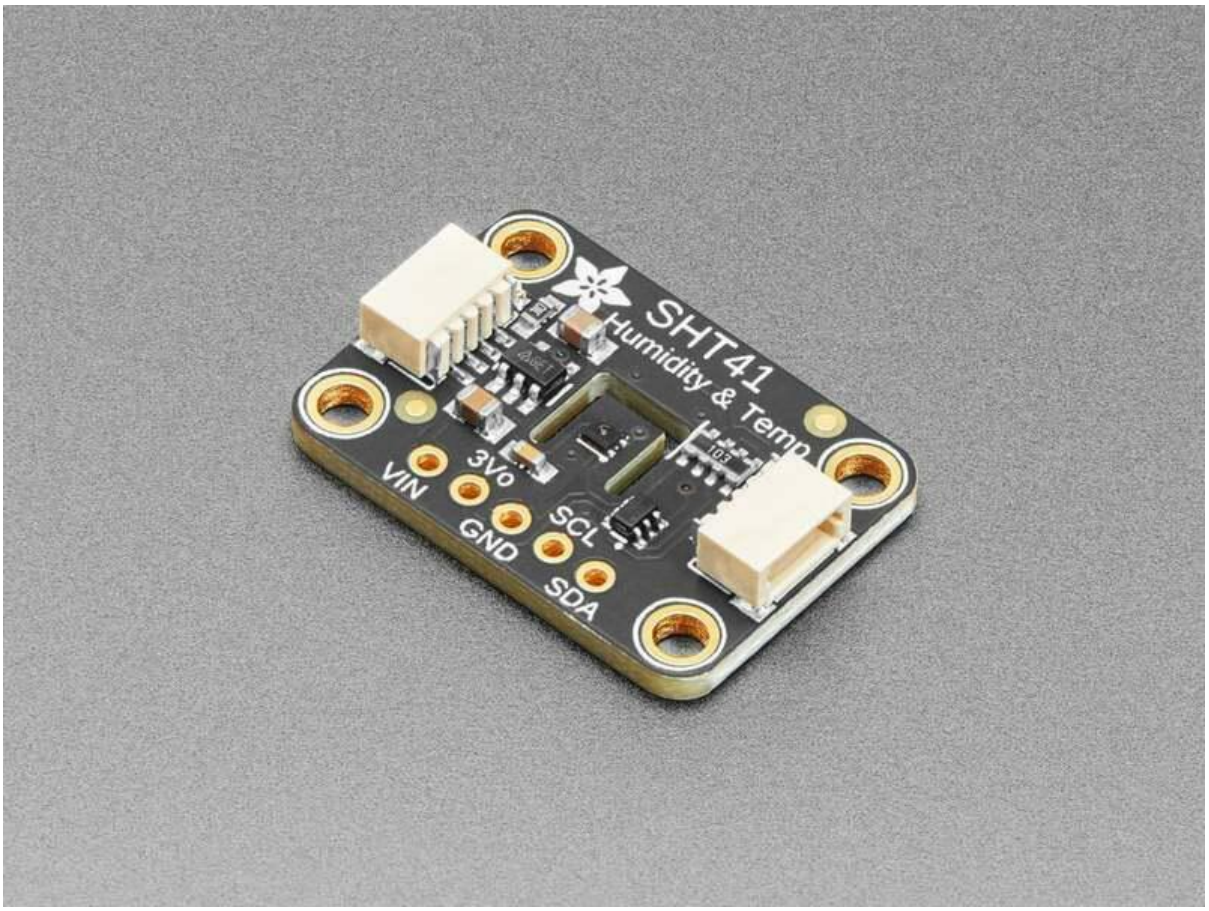
Adafruit Sensirion SHT40, SHT41 & SHT45 Temperature & Humidity Sensors

Created by Kattni Rembor



https://learn.adafruit.com/adafruit-sht40-temperature-humidity-sensor

Last updated on 2023-08-29 04:37:32 PM EDT

Table of Contents

Overview 3

Pinouts 5

- Power Pins
- I2C Logic Pins

Arduino 6

- Wiring
- Installation
- Load Example
- Example Code

Arduino Docs 10

Python & CircuitPython 10

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of SHT4x Library
- Python Installation of SHT4x Library
- CircuitPython & Python Usage
- Full Example Code

Python Docs 14

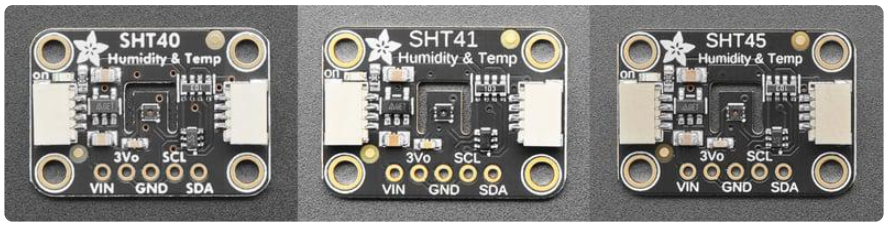
WipperSnapper 14

- What is WipperSnapper
- Wiring
- Usage

Downloads 21

- Files:
- Schematic and Fab Print for SHT40
- Schematic and Fab Print for SHT41
- Schematic and Fab Print for SHT45

Overview



Sensirion Temperature/Humidity sensors are some of the finest & highest-accuracy devices you can get. And finally, we have some that have a true I2C interface for easy reading. The SHT40, SHT41, and SHT45 sensors are the fourth generation (started at the SHT10 and worked its way up to the top!).

The SHT40 has an excellent ±1.8% typical relative humidity accuracy from 25 to 75% and ±0.2 °C typical accuracy from 0 to 75 °C.

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.

The SHT45 has an even more excellent ±1% typical relative humidity accuracy from 25 to 75% and ±0.1 °C typical accuracy from 0 to 75 °C.

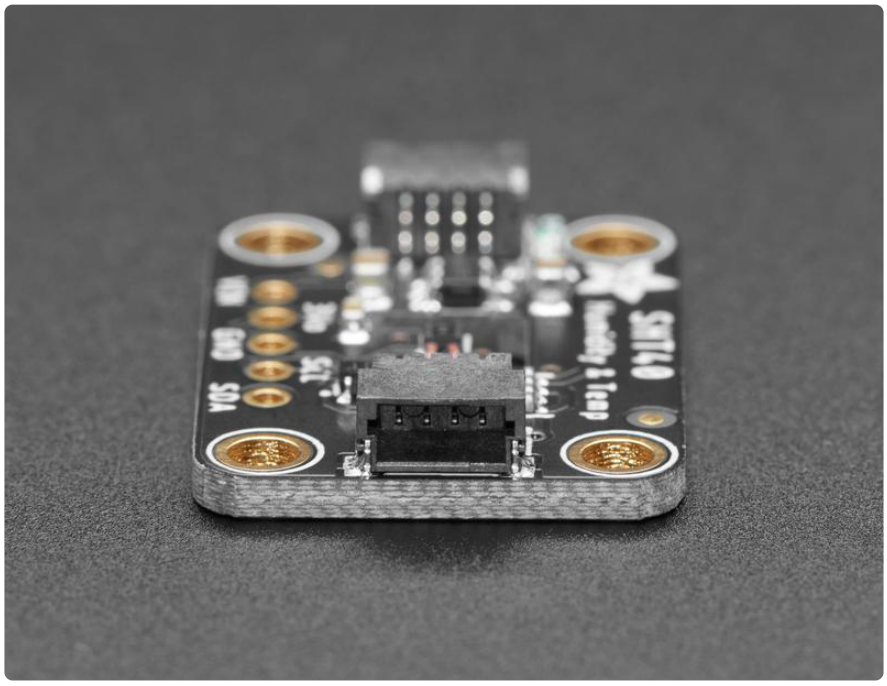
Note that compared to the SHT40, the SHT41 has the same typical accuracy of ±1.8% RH from 25-75% but has much better accuracy over the full humidity range, with a typical accuracy of ±2% over the 0-100% RH range, whereas the SHT40 has a worst-case typical accuracy of ±3%. Also, the SHT41 max accuracy is ±3% where as the SHT40 has a max accuracy of ±6%. So basically, you'll get better accuracy for not-much-more cost. If you need the 'best' of the SHT4x series, you want the SHT45.

Other than these differences in relative humidity and temperature accuracy, there is no discernible difference between these three chips. The I2C addresses and code are identical for all. So, how do you know which one you have? The text on the board indicates which breakout you have.

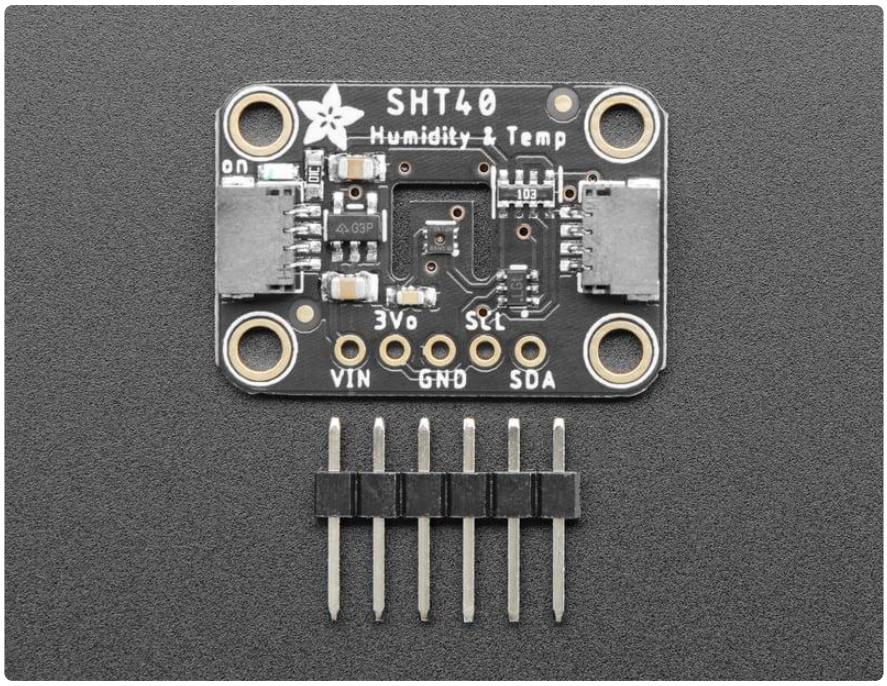
Other than accuracy, there is no difference between these chips. The only way to know which one you have is to read the text on the breakout board.



Unlike some earlier SHT sensors, these sensors have a true I2C interface for easy interfacing with only two wires (plus power and ground!). Thanks to the voltage regulator and level shifting circuitry we've included on the breakouts, they are also 3V or 5V compliant, so you can power and communicate with them using any microcontroller or microcomputer.



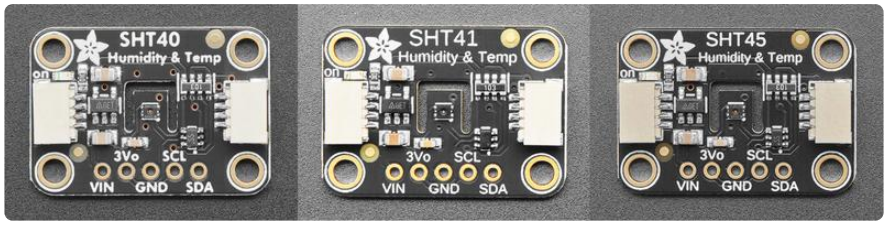
Such lovely chips - so we spun up breakout boards with the SHT4x and some supporting circuitry such as pullup resistors and capacitors. To make things even easier, we've included SparkFun Qwiic () compatible STEMMA QT () connectors for the I2C bus so you don't even need to solder! QT Cable is not included, but we have a variety in the shop ().



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 these chips, so you can use it with just about any microcontroller or single-board computer like Raspberry Pi.

Pinouts



The SHT40, SHT41, and SHT45 have identical 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 5V microcontroller like Arduino, use 5V.
- 3V - 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

The default I2C address is 0x44.

- 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 () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ().

The I2C address is 0x44 and cannot be changed (a manufacturer limitation). If you must monitor multiple sensors with one microcontroller, look at the PCA9546 4-channel () or TCA9548A 8-channel multiplexer ().

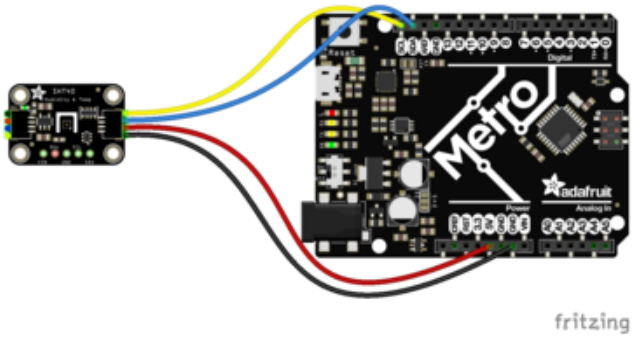
Arduino

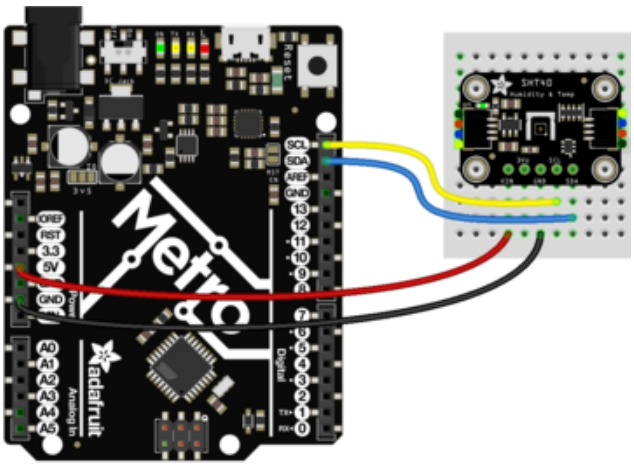
The Arduino wiring and code for the SHT40, SHT41 and the SHT45 are identical!

Even though this page references the SHT40, all of the instructions are exactly the same for the SHT41 and SHT45!

Wiring

Connecting the SHT40 to your Feather or Arduino is easy:



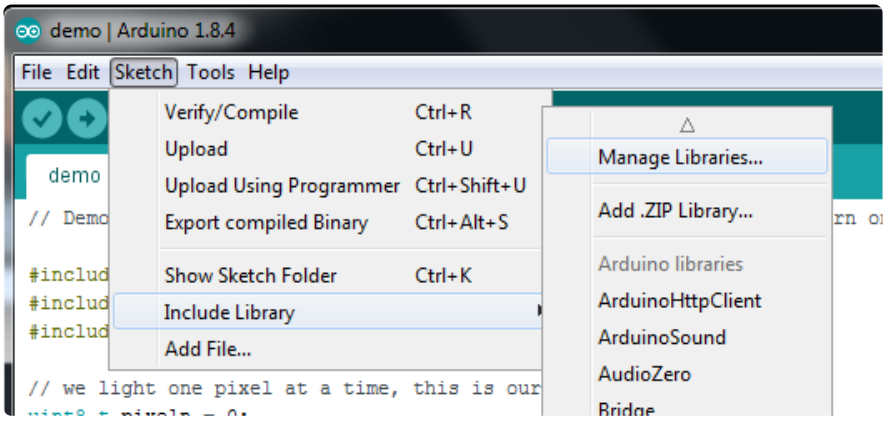


If you are running a Feather (3.3V), connect Feather 3V to board VIN
If you are running a 5V Arduino (Uno, etc.), connect Arduino 5V to board VIN
Connect Feather or Arduino GND to board GND
Connect Feather or Arduino SCL to board SCL
Connect Feather or Arduino SDA to board SDA

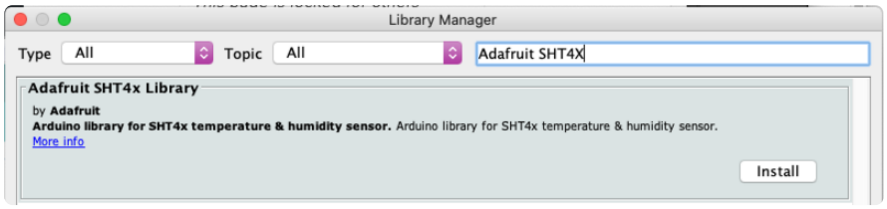
The final results should resemble the illustration above, showing an Adafruit Metro development board.

Installation

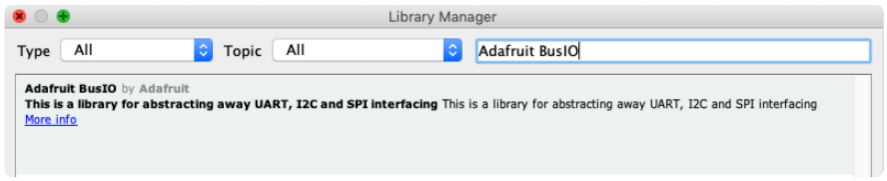
You can install the Adafruit SHT4X Library for Arduino using the Library Manager in the Arduino IDE:



Click the Manage Libraries ... menu item, search for Adafruit SHT4X, and select the Ad afruit SHT4X library:



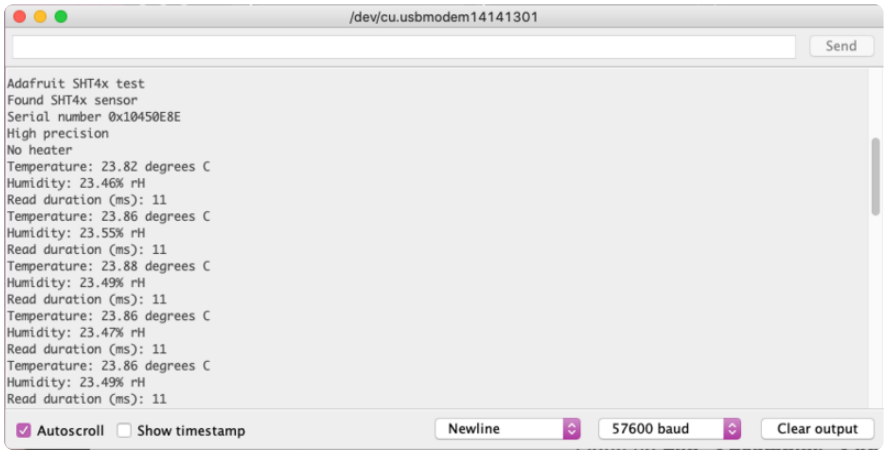
Then follow the same process for the Adafruit BusIO library.



Load Example

Open up File -> Examples -> Adafruit SHT4X -> SHT4test and upload to your Arduino wired up to the sensor.

Upload the sketch to your board and open up the Serial Monitor (Tools->Serial Monitor). You should see the the values for temperature and humidity.



Example Code

The following example code is part of the standard library, but illustrates how you can retrieve sensor data from the SHT40 for the temperature and humidity values:

```
/*************************************************
  This is an example for the SHT4x Humidity & Temp Sensor

  Designed specifically to work with the SHT4x sensor from Adafruit
  ----> https://www.adafruit.com/products/4885

  These sensors use I2C to communicate, 2 pins are required to
  interface
```

```
 ************************************************/

#include "Adafruit_SHT4x.h"

Adafruit_SHT4x sht4 = Adafruit_SHT4x();

void setup() {
  Serial.begin(115200);

  while (!Serial)
    delay(10);     // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit SHT4x test");
  if (! sht4.begin()) {
    Serial.println("Couldn't find SHT4x");
    while (1) delay(1);
  }
  Serial.println("Found SHT4x sensor");
  Serial.print("Serial number 0x");
  Serial.println(sht4.readSerial(), HEX);

  // You can have 3 different precisions, higher precision takes longer
  sht4.setPrecision(SHT4X_HIGH_PRECISION);
  switch (sht4.getPrecision()) {
     case SHT4X_HIGH_PRECISION:
       Serial.println("High precision");
       break;
     case SHT4X_MED_PRECISION:
       Serial.println("Med precision");
       break;
     case SHT4X_LOW_PRECISION:
       Serial.println("Low precision");
       break;
  }

  // You can have 6 different heater settings
  // higher heat and longer times uses more power
  // and reads will take longer too!
  sht4.setHeater(SHT4X_NO_HEATER);
  switch (sht4.getHeater()) {
     case SHT4X_NO_HEATER:
       Serial.println("No heater");
       break;
     case SHT4X_HIGH_HEATER_1S:
       Serial.println("High heat for 1 second");
       break;
     case SHT4X_HIGH_HEATER_100MS:
       Serial.println("High heat for 0.1 second");
       break;
     case SHT4X_MED_HEATER_1S:
       Serial.println("Medium heat for 1 second");
       break;
     case SHT4X_MED_HEATER_100MS:
       Serial.println("Medium heat for 0.1 second");
       break;
     case SHT4X_LOW_HEATER_1S:
       Serial.println("Low heat for 1 second");
       break;
     case SHT4X_LOW_HEATER_100MS:
       Serial.println("Low heat for 0.1 second");
       break;
  }

}


void loop() {
  sensors_event_t humidity, temp;
```

```
  uint32_t timestamp = millis();
  sht4.getEvent(&humidity, &temp);// populate temp and humidity objects with fresh
data
  timestamp = millis() - timestamp;

  Serial.print("Temperature: "); Serial.print(temp.temperature); Serial.println("
degrees C");
  Serial.print("Humidity: "); Serial.print(humidity.relative_humidity);
Serial.println("% rH");

  Serial.print("Read duration (ms): ");
  Serial.println(timestamp);

  delay(1000);
}
```

Arduino Docs

Arduino Docs ()

Python & CircuitPython

It's easy to use the Adafruit Sensirion SHT40, SHT41, and SHT45 Temperature & Humidity Sensors with CircuitPython and the Adafruit CircuitPython SHT4x () module. This module allows you to easily write Python code that reads temperature and humidity data from either sensor.

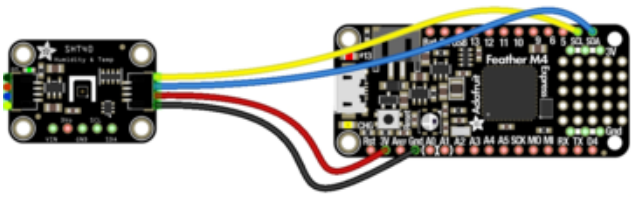
You can use these sensors with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library ().

The wiring and code for the SHT40 works exactly the same way with the SHT41 and SHT45.

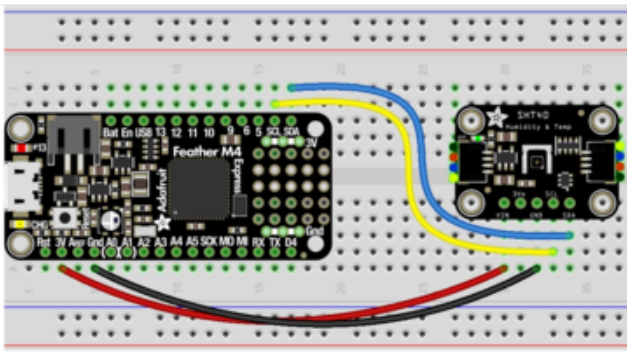
Even though this page references the SHT40, the code and wiring for the SHT41 and SHT45 are exactly the same! Follow the instructions as-is to use the SHT41 or SHT45.

CircuitPython Microcontroller Wiring

First wire up an SHT40 to your board exactly as follows. Here is an example of the SHT40 wired to a Feather using I2C:



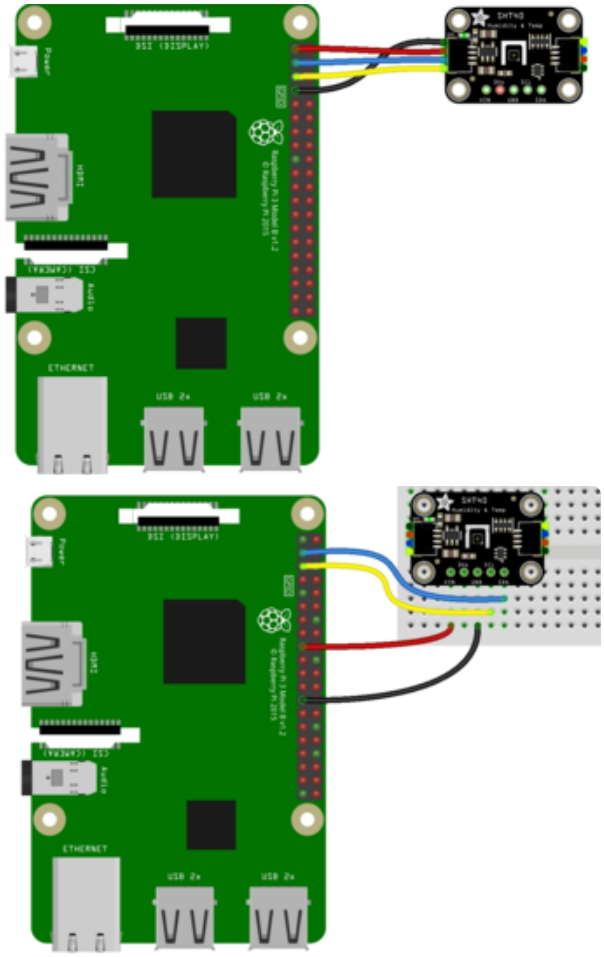
Board 3V to sensor VIN
Board GND to sensor GND
Board SCL to sensor SCL
Board SDA to sensor SDA



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 ().

Here's the Raspberry Pi wired with I2C:



Pi 3V3 to sensor VIN
Pi GND to sensor GND
Pi SCL to sensor SCL
Pi SDA to sensor SDA

CircuitPython Installation of SHT4x Library

You'll need to install the Adafruit CircuitPython SHT4x () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (). Our CircuitPython starter guide has a great page on how to install libraries from the library bundle ().

Copy the following file from the bundle to the lib folder on your CIRCUITPY drive:

- adafruit_sht4x.mpy

Before continuing make sure your board's lib folder or root filesystem has the adafruit_sht4x.mpy file copied over.

Next connect to the board's serial REPL ()so you are at the CircuitPython `>>>` prompt .

Python Installation of SHT4x 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 ()!

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

- `pip3 install adafruit-circuitpython-sht4x`

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

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read from the sensor in the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import adafruit_sht4x

sht = adafruit_sht4x.SHT4x(board.I2C())
```

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

- relative_humidity - The current relative humidity in % rH
- temperature - The current temperature in degrees celsius

```
print(sht.temperature)
print(sht.relative_humidity)
```

```
>>> print(sht.temperature)
23.8758
>>> print(sht.relative_humidity)
25.384
```

That's all there is to reading temperature and humidity from the SHT40 sensor!

Full Example Code

```
# SPDX-FileCopyrightText: Copyright (c) 2020 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_sht4x

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
sht = adafruit_sht4x.SHT4x(i2c)
print("Found SHT4x with serial number", hex(sht.serial_number))

sht.mode = adafruit_sht4x.Mode.NOHEAT_HIGHPRECISION
# Can also set the mode to enable heater
# sht.mode = adafruit_sht4x.Mode.LOWHEAT_100MS
print("Current mode is: ", adafruit_sht4x.Mode.string[sht.mode])

while True:
    temperature, relative_humidity = sht.measurements
    print("Temperature: %0.1f C" % temperature)
    print("Humidity: %0.1f %%" % relative_humidity)
    print("")
    time.sleep(1)
```

Python Docs

Python Docs ()

WipperSnapper

Though this page references the SHT40, the SHT41 and SHT45 will work the exact same way. Follow these instructions with the SHT41 or the SHT45 to use it with 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 (), a web platform designed (by Adafruit! ()) 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.

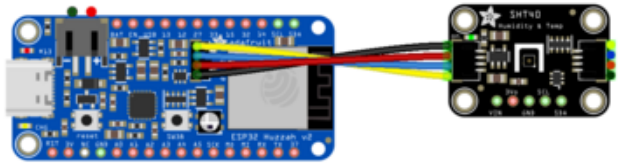
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

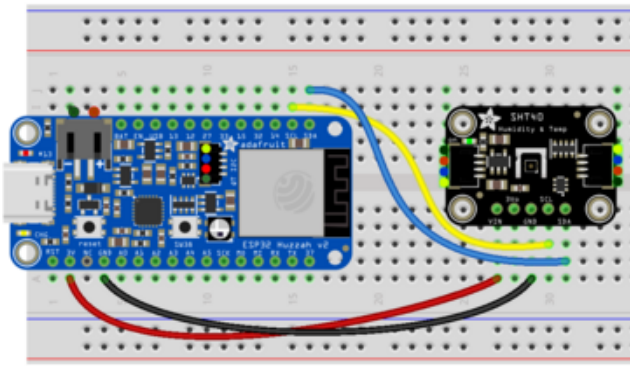
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

Quickstart: Adafruit IO WipperSnapper

Wiring

First, wire up an SHT40 to your board exactly as follows. Here is an example of the SHT40 wired to an Adafruit ESP32 Feather V2 () using I2C with a STEMMA QT cable (no soldering required) ()



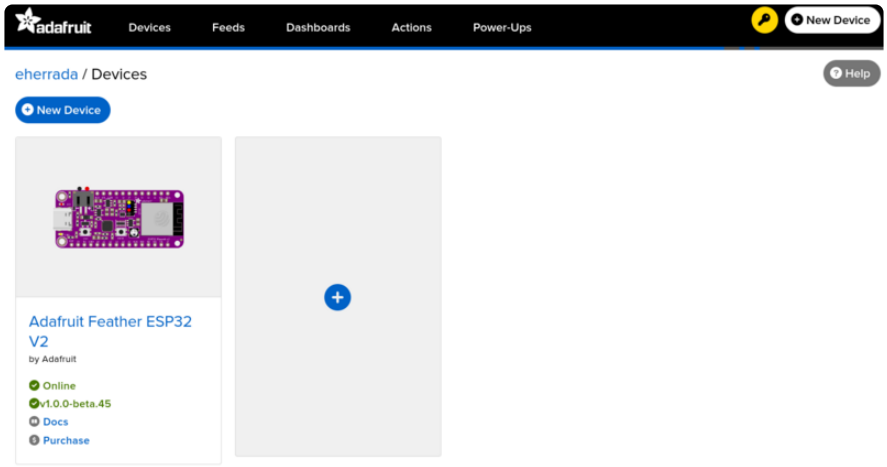


Board 3V to sensor VIN (red wire on STEMMA QT)
Board GND to sensor GND (black wire on STEMMA QT)
Board SCL to sensor SCL (yellow wire on STEMMA QT)
Board SDA to sensor SDA (blue wire on STEMMA QT)

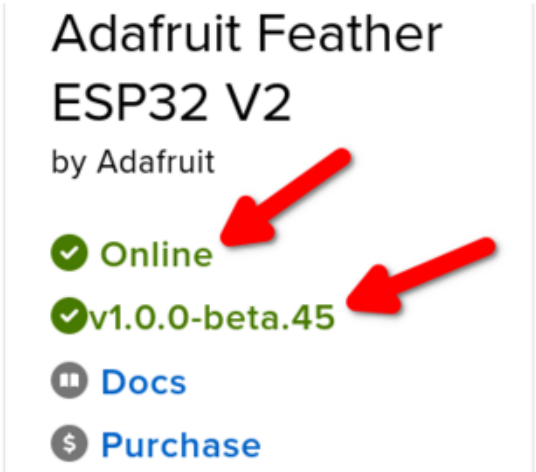
Usage

Connect your board to Adafruit IO Wippersnapper and navigate to the WipperSnapper board list ().

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 () first.

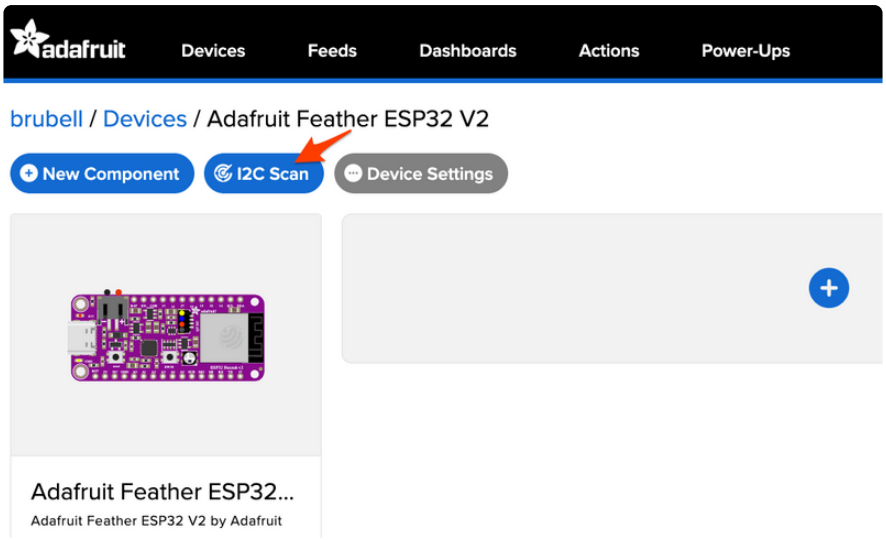


On the device page, quickly check that you're running the latest version of the WipperSnapper firmware.

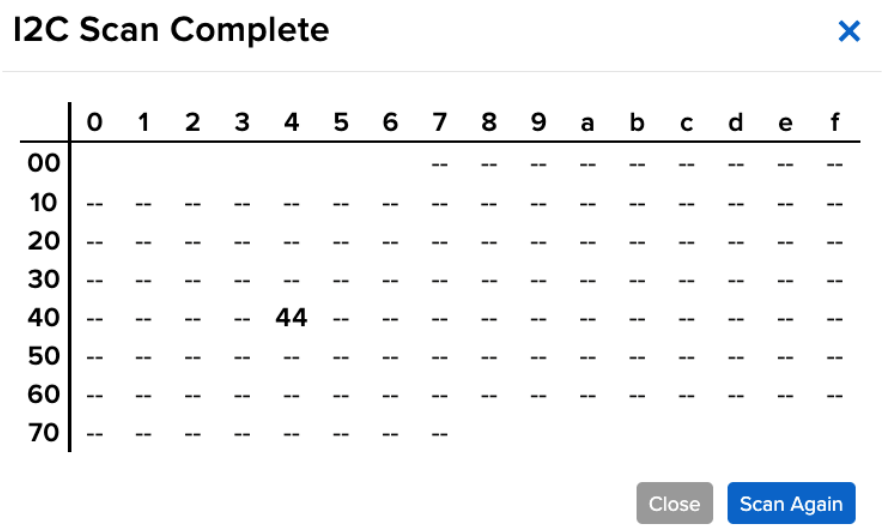
The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide.
If the firmware version is red with an "X" - update to the latest WipperSnapper firmware () 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 SHT40's default I2C address of `0x44` pop-up in the I2C scan list.



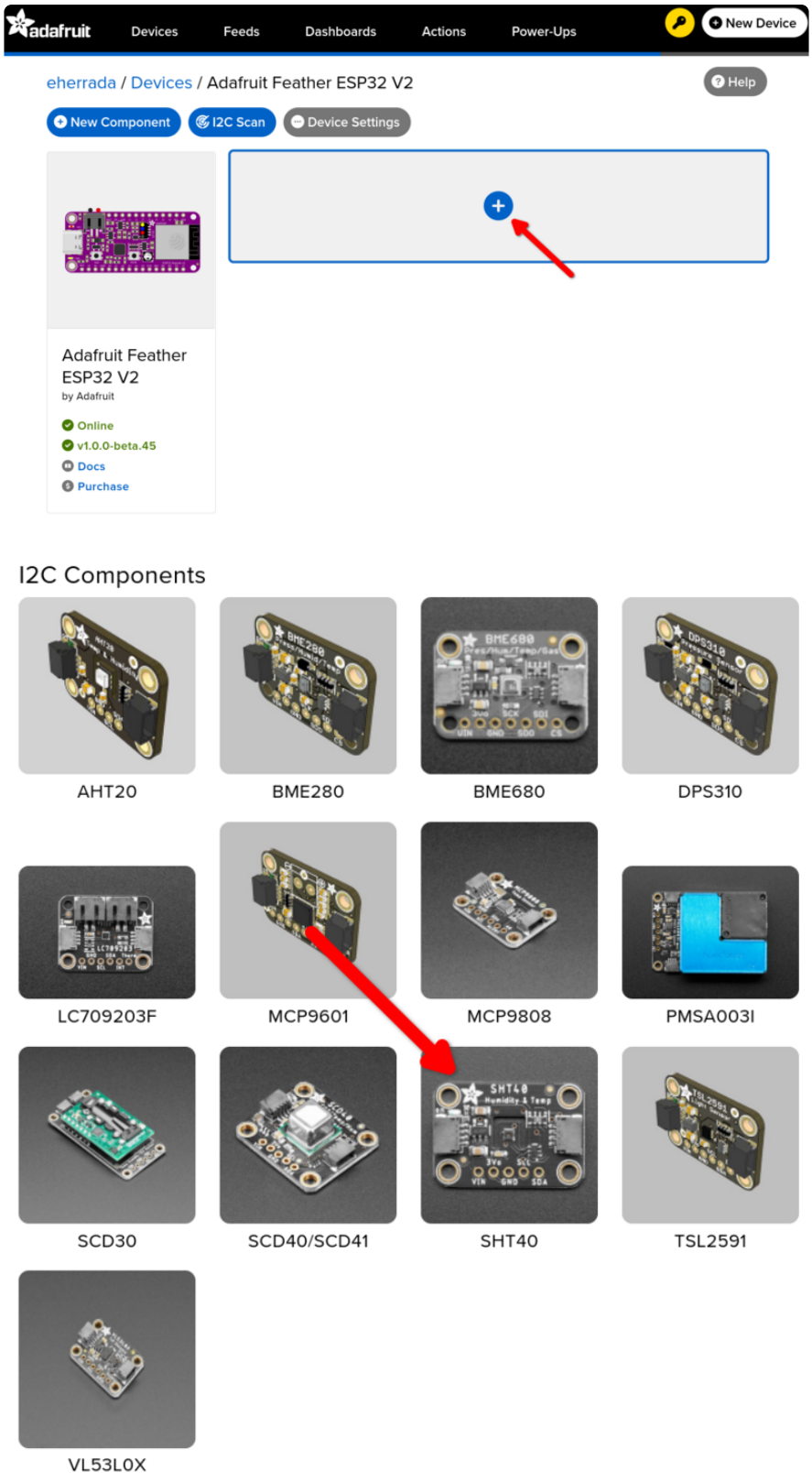
I don't see the sensor's I2C address listed!

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.

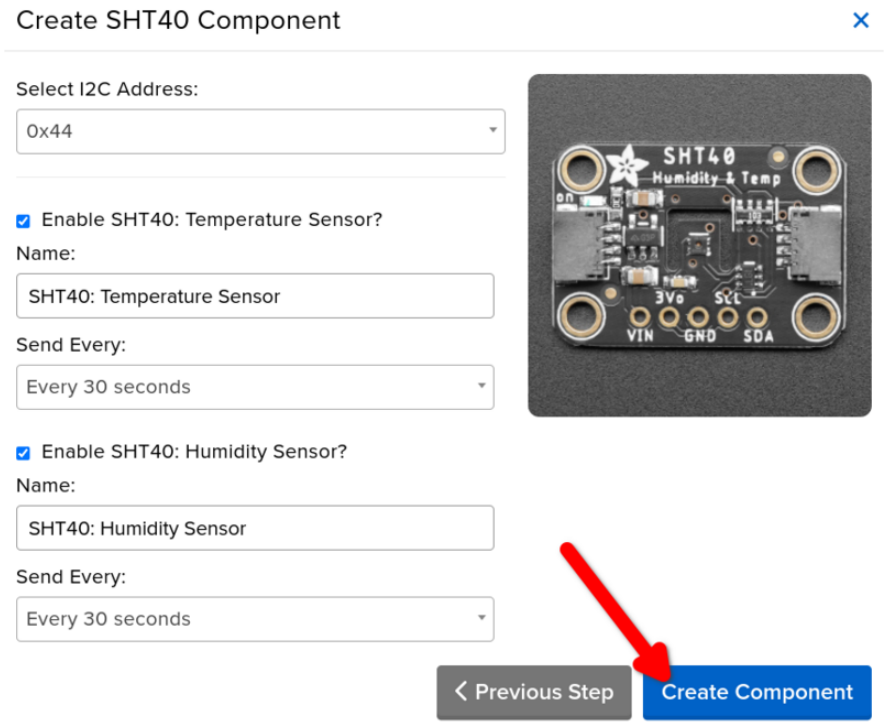


On the component configuration page, the SHT40's sensor address should be listed along with the sensor's settings.

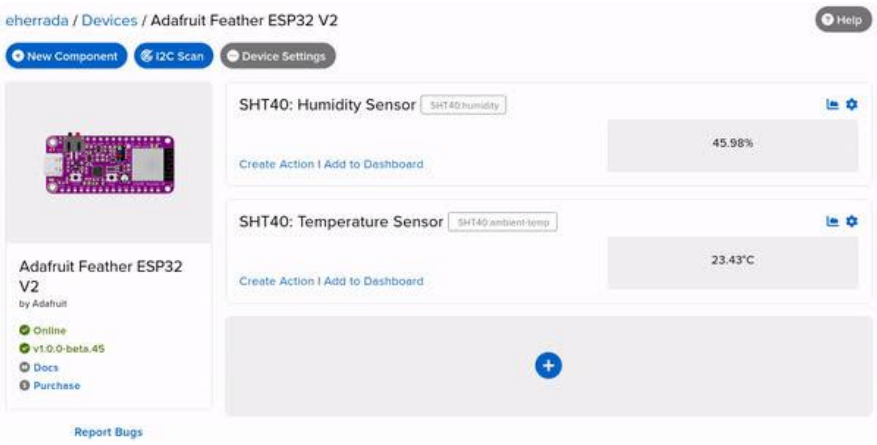
The Send Every option is specific to each sensor's measurements. This option will tell the Feather how often it should read from each of the SHT40's two sensors and send

the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

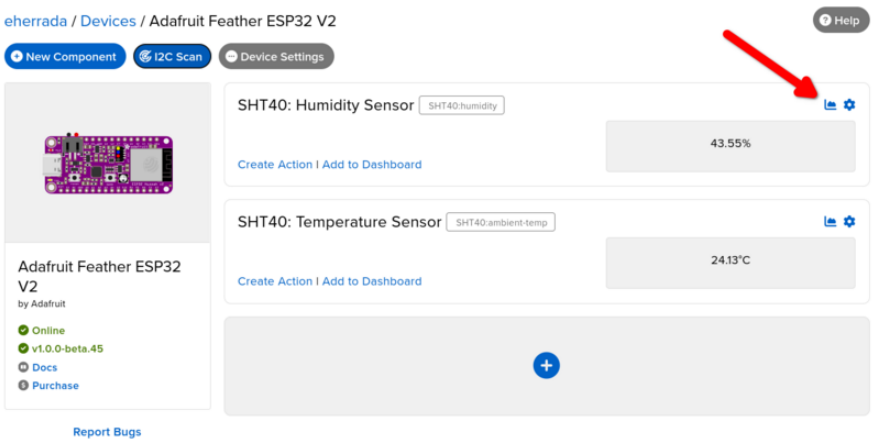
For this example, set the Send Every interval for each sensor to every 30 seconds.



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.

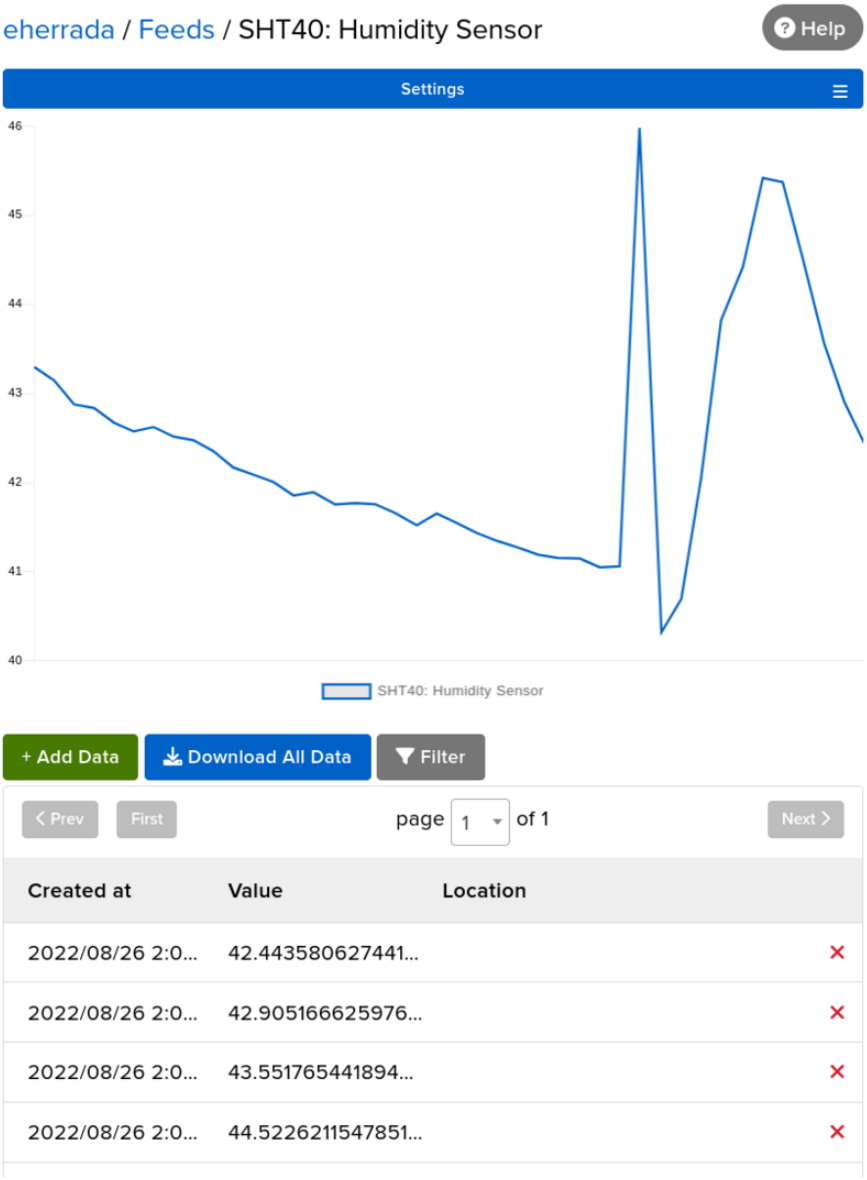


To view the data that has been logged from the sensor, click on the graph next to the sensor name.



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 ().

The SHT40 has two sensors that each have their own feeds. In this picture, we're looking at the Humidity 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 two feeds (one for each of the SHT40'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 ().

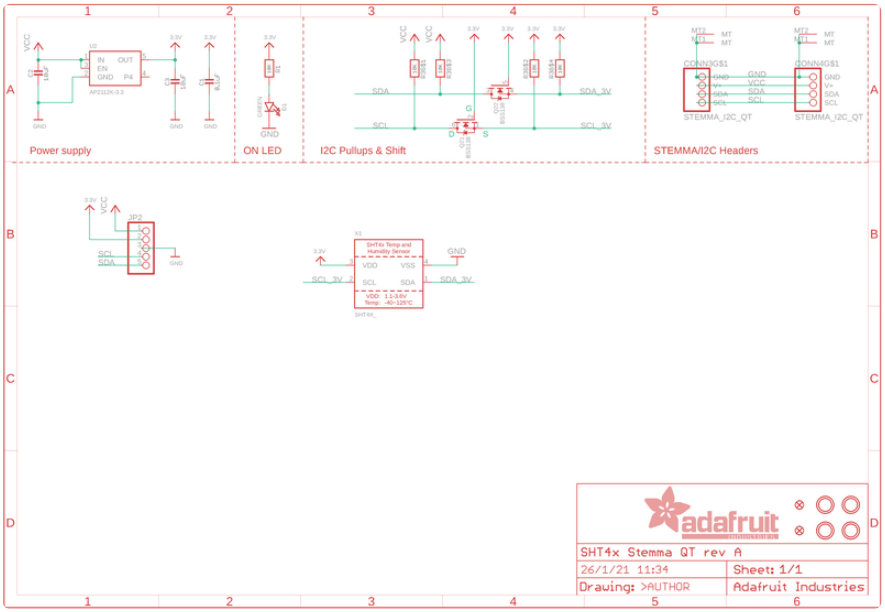
Downloads

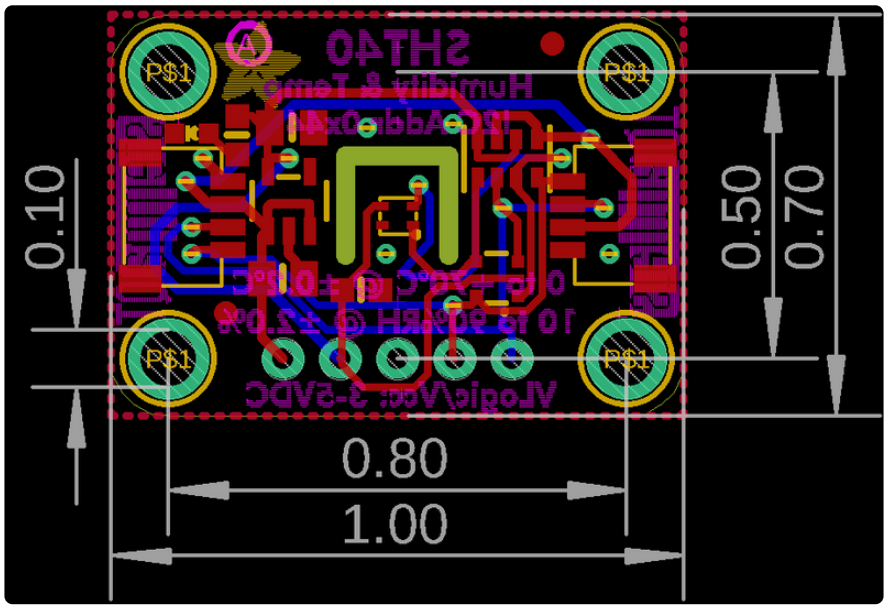
Files:

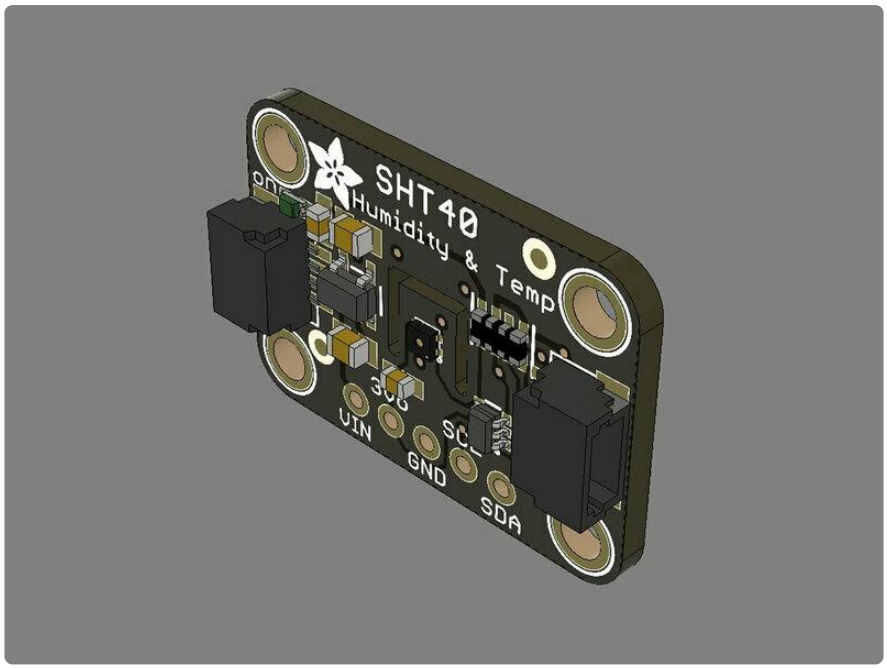
- SHT4x Datasheet ()
- SHT4x Fritzing object in the Adafruit Fritzing Library ()
- SHT4x EagleCAD PCB files on GitHub ()

- SHT40 3D models on GitHub ()

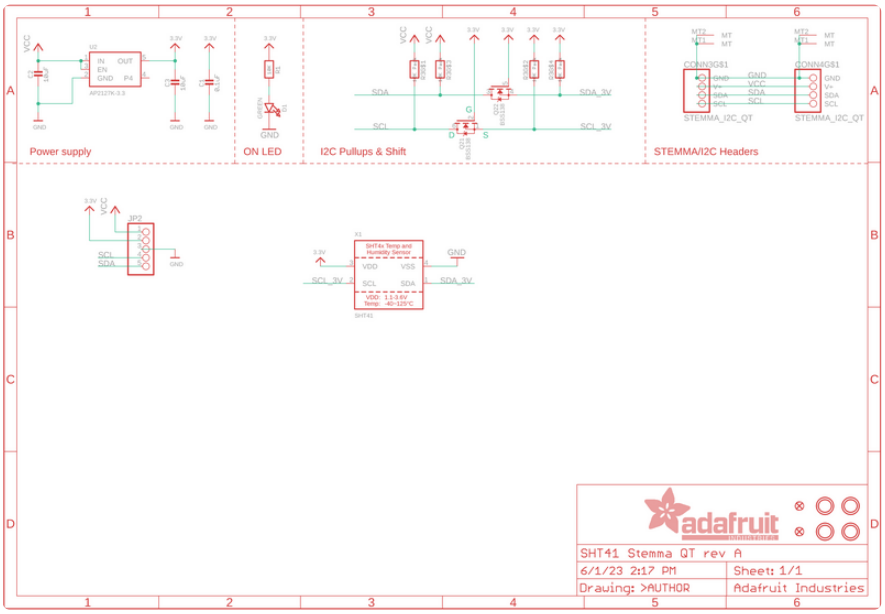
Schematic and Fab Print for SHT40

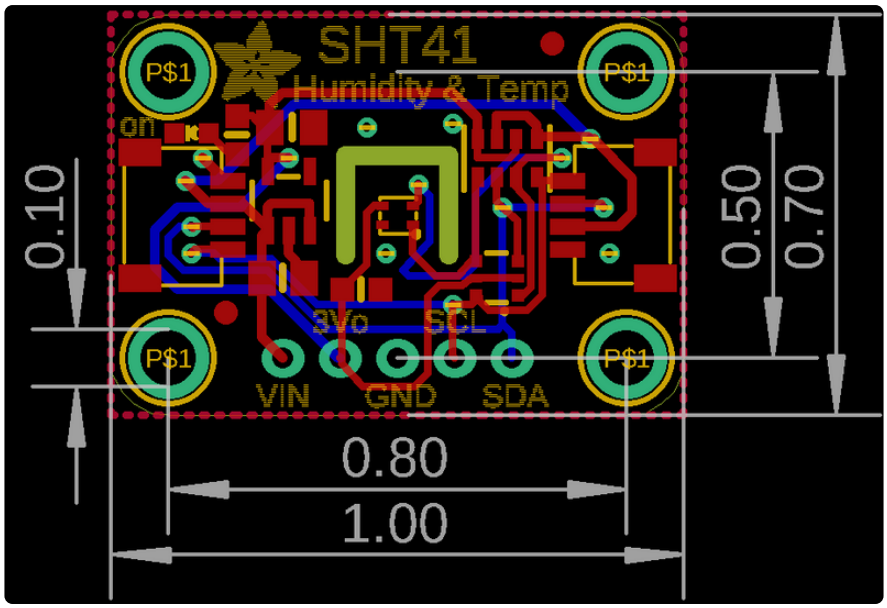




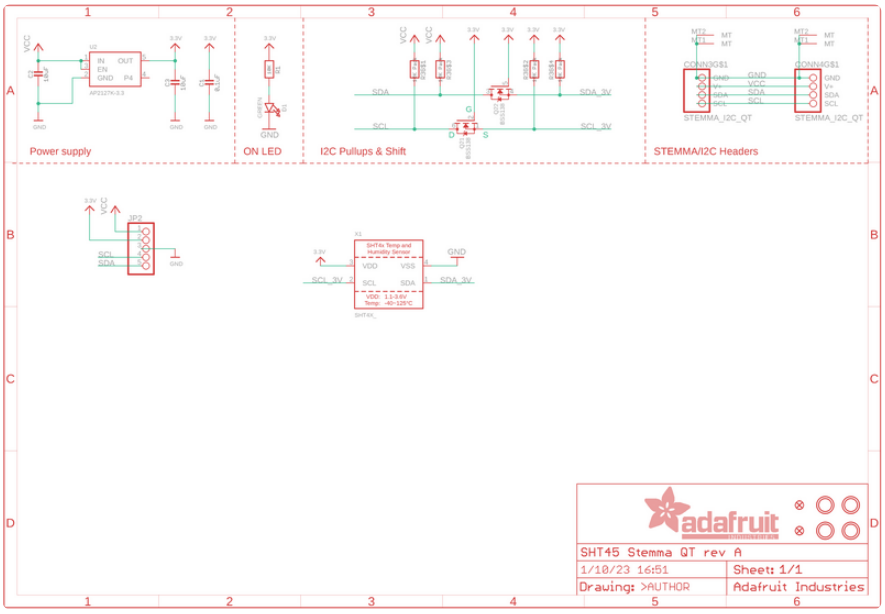


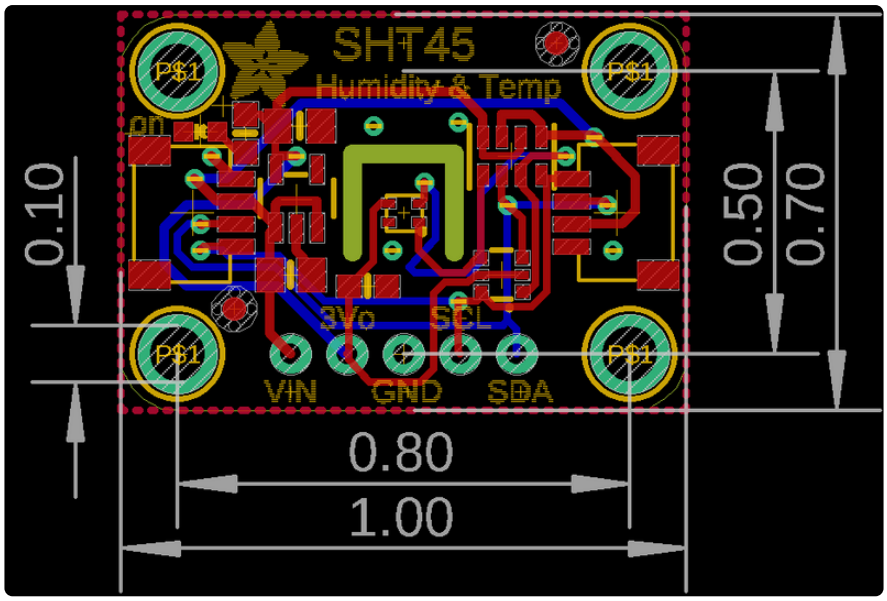
Schematic and Fab Print for SHT41





Schematic and Fab Print for SHT45




SHT45
Humidity & Temp
P$1
P$1
P$1
P$1
3Vo
SCL
VIN
GND
SDA
0.10
0.50
0.70
0.80
1.00

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Adafruit:
4885