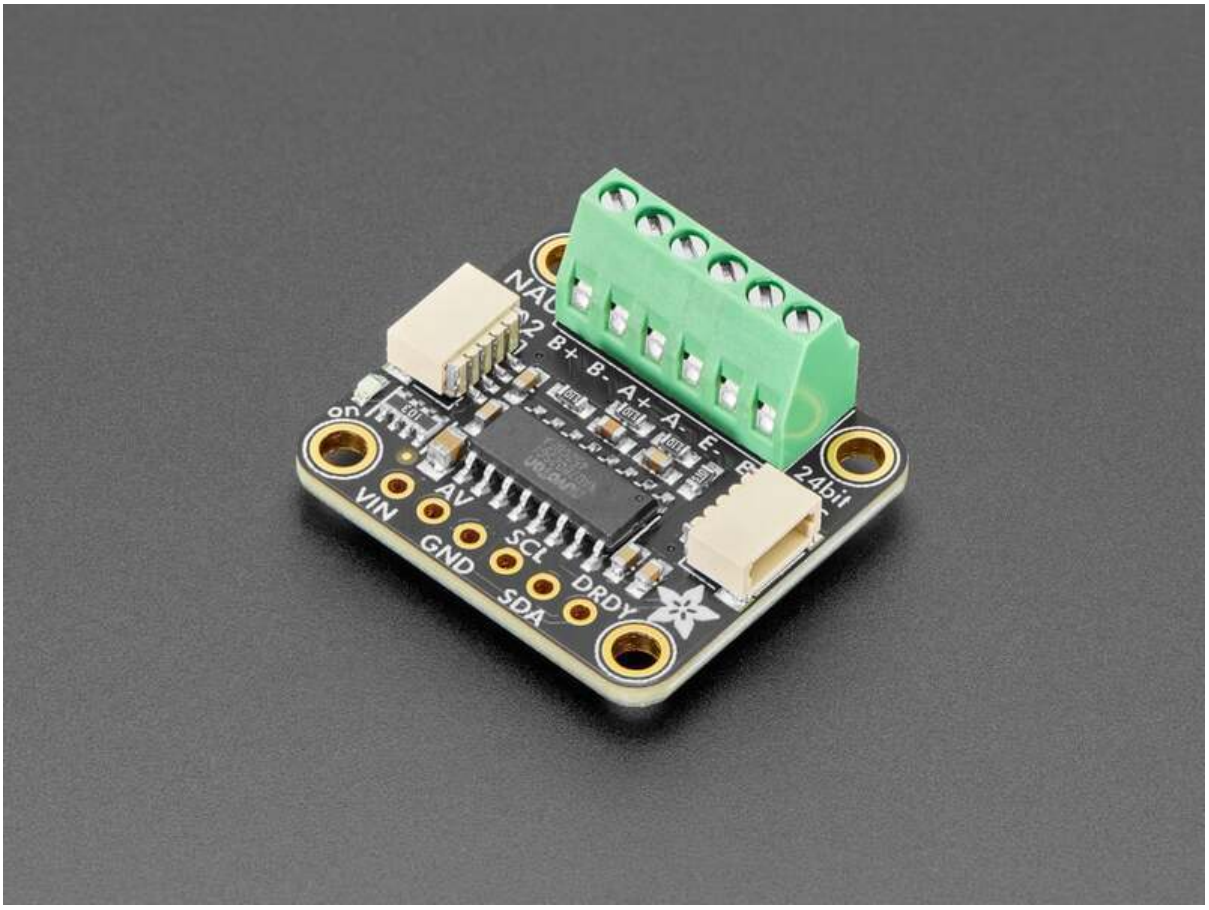
# Adafruit NAU7802 24-Bit ADC - STEMMA QT / Qwiic

Created by Liz Clark




https://learn.adafruit.com/adafruit-nau7802-24-bit-adc-stemma-qt-qwiic

Last updated on 2025-06-24 12:10:38 PM EDT

# Table of Contents


Overview 3

Pinouts 5

- Updated Rev C:
- Original Rev B:
- Power Pins
- I2C Logic Pins
- Terminal Block Pins
- Data Ready Output Pin
- Power LED

Python & CircuitPython 8

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of NAU7802 Library
- CircuitPython Usage
- Python Usage
- Example Code

Python Docs 15

Arduino 16

- Wiring
- Library Installation
- Load Example

Arduino Docs 21

WipperSnapper 21

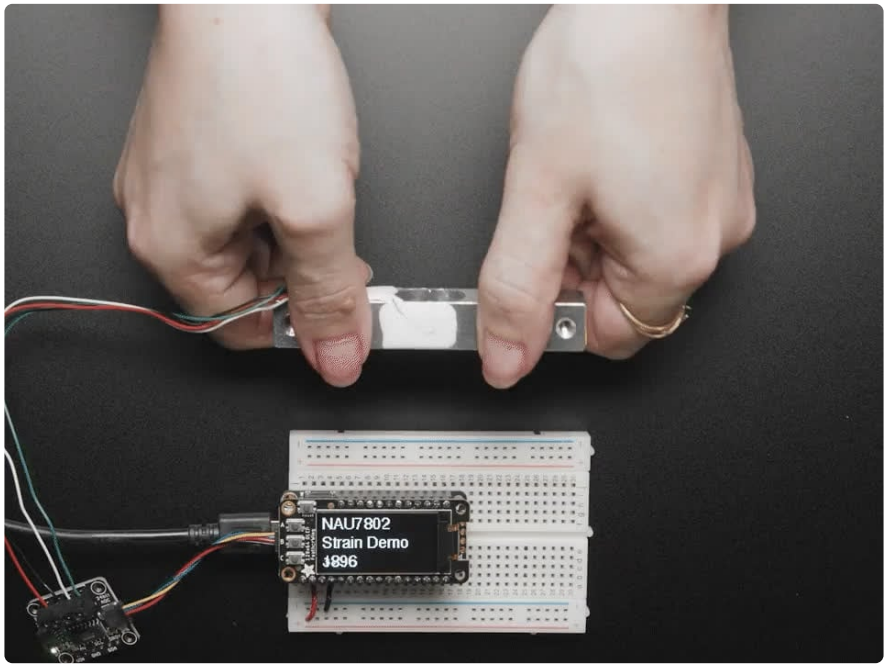
- What is WipperSnapper
- Wiring
- Usage

Downloads 27

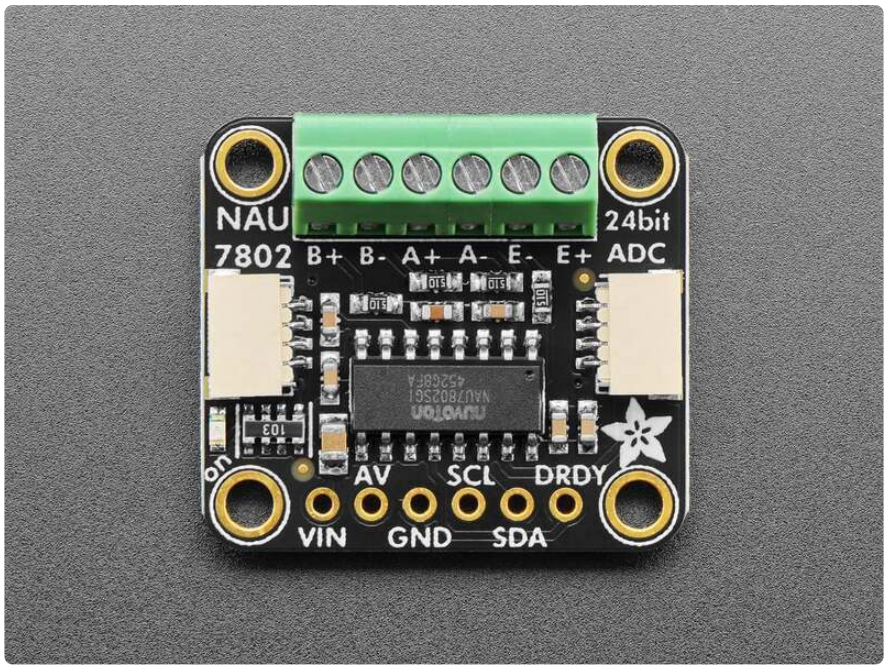
- Files
- Schematic and Fab Print

# Overview

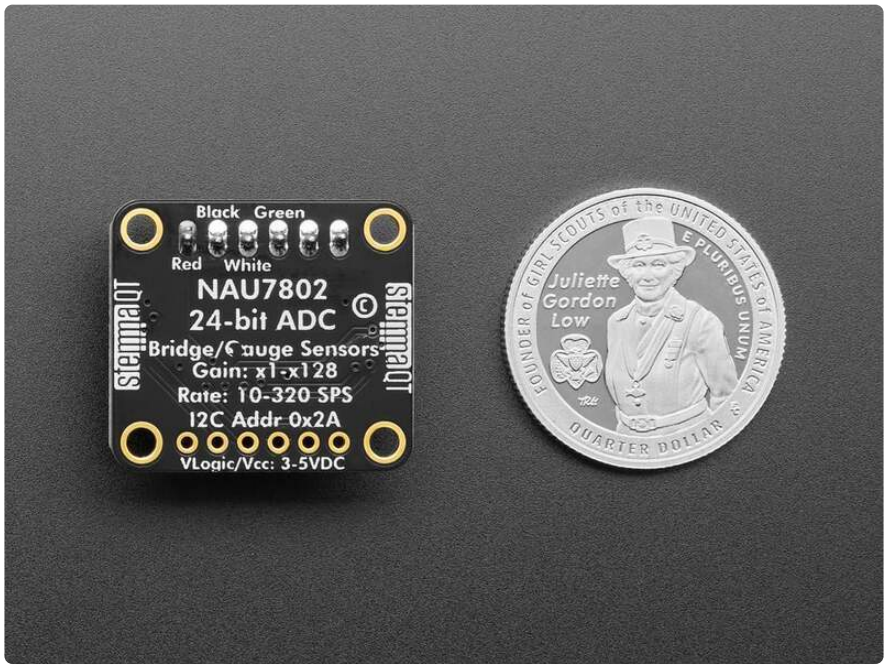
As of June 23rd, 2025 – By popular demand! We've updated this design to expose the 'second' ADC input (B+ and B-). We've updated this PCB to now come with a 6-pin terminal block port, instead of a 4-pin.




If you are feeling the stress and strain of ~~modern life~~ a Wheatstone bridge and you want to quantify it, this handy breakout will do the job, no sweat! The **Adafruit NAU7802** contains a super-high-resolution 24-Bit differential ADC with extra gain and calibration circuitry that makes it perfect for measuring strain gauges / load cells or other sensors that have four wires that are connected in a Wheatstone bridge arrangement (https://adafru.it/-kd).

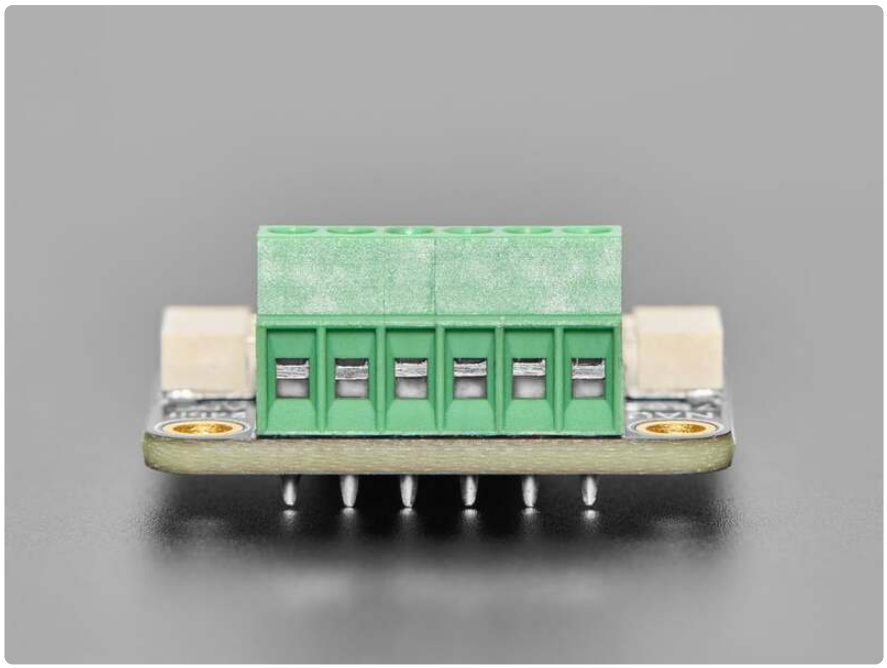



Each breakout comes with a NAU7802 ADC chip, plus some support circuitry and a terminal block that can be used to connect a 4-wire sensor.




The **E-** pad connects to ground (often a black wire), the **E+** pad connects to a generated voltage from the NAU7802 that can be configured from 2.4V to 4.0V for a solid positive reference (often a red wire).

Then **A-** and **A+** pads connect to the differential outputs from the bridge. For example, connecting to a strain gauge these tend to be the white and green wires.



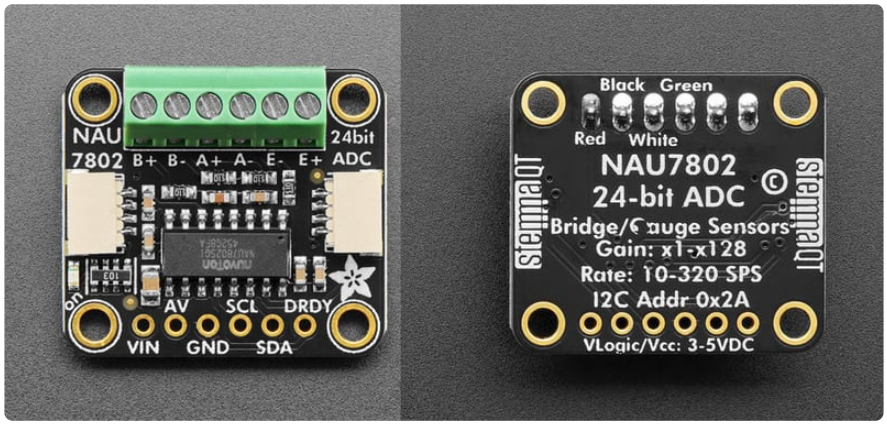
As the sensor is twisted and bent, the slight changes in resistance are converted to minuscule voltage changes that can be read by the NAU ADC for converting into force or mass measurements. You can use our Arduino library (https://adafru.it/-fd) or CedarGrove's CircuitPython/Python library (https://adafru.it/-fc) to configure and read the ADC for fast interfacing. Note that this sensor has a fixed I2C address of 0x2A. If multiple sensors are desired on one I2C bus, a multiplexer can be used (https://adafru.it/-af).

---

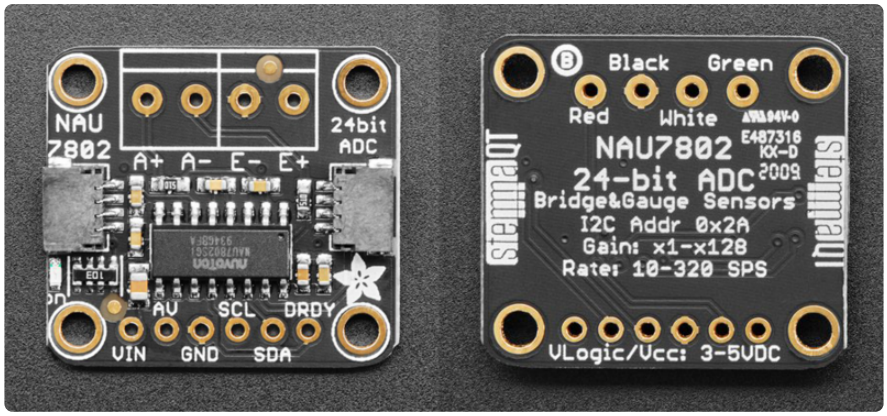
# Pinouts

As of June 23rd, 2025 – By popular demand! We've updated this design to expose the 'second' ADC input (B+ and B-), so we've updated this PCB to now come with a 6-pin terminal block port, instead of 4-pin.

## Updated Rev C:




## Original Rev B:




The default I2C address for the NAU7802 is **0x2A**.

## 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.

- **AV** - This is the AVDD pin. It is the analog power supply pin and can supply 2.4V to 4.0V.
- **GND** - This is common ground for power and logic.

## I2C Logic Pins

- **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 I2C **STEMMA QT** / Qwiic connectors or to other things with various associated accessories (https://adafru.it/JRA).

The secondary ADC (pins B+ and B-) are only broken out on rev C of the breakout.

## Terminal Block Pins

- **A+** - Connected to VIN1P, which is Non-Inverting Input #1. It is a differential output from the bridge.
- **A-** - Connected to VIN1N, which is Inverting Input #1. It is a differential output from the bridge.
- **B+** (rev C only) - Connected to VIN2P, which is Non-Inverting Input #2. It is a differential output from the bridge.
- **B-** (rev C only) - Connected to VIN2N, which is Inverting Input #2. It is a differential output from the bridge.
- **E+** - Connected to AVDD for a generated voltage from the NAU7802. It can be configured from 2.4V to 4.0V for a solid positive reference.
- **E-** - Connected to common ground.

## Data Ready Output Pin

- **DRDY** - Data Ready Output pin. It indicates when a conversion is complete and new data is available for readout

## Power LED

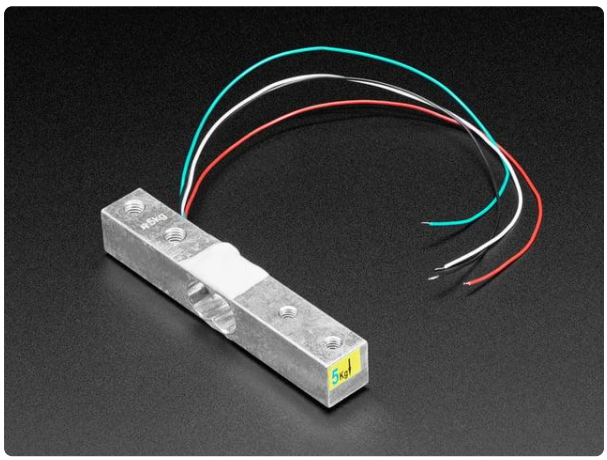
- **Power LED -** In the bottom left corner, below the STEMMA connector, on the front of the board, is the power LED, labeled **on**. It is the green LED.

# Python & CircuitPython

As of June 23rd, 2025 – By popular demand! We've updated this design to expose the 'second' ADC input (B+ and B-), so we've updated this PCB to now come with a 6-pin terminal block port, instead of 4-pin.

It's easy to use the **NAU7802** with Python or CircuitPython, and the CedarGrove CircuitPython NAU7802 (https://adafru.it/1ali) module. This module allows you to easily write Python code that reads the value from the **NAU7802** ADC with a strain gauge.

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).

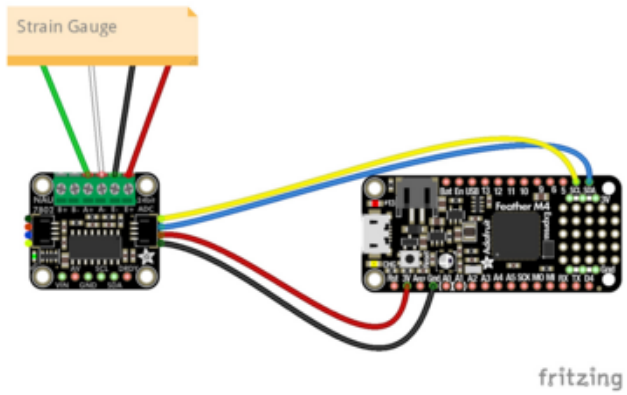


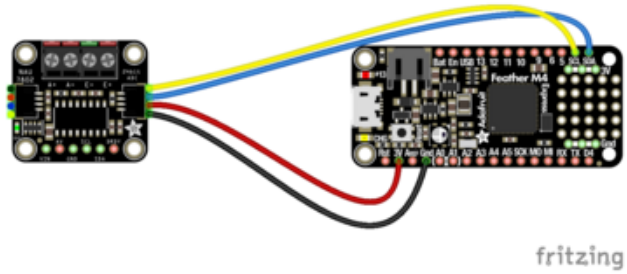
Strain Gauge Load Cell - 4 Wires - 5Kg
A strain gauge is a type of electronic sensor used to measure force or strain (big surprise there). They are made of an insulating flexible backing with a metallic...
https://www.adafruit.com/product/4541

Rev C of the Adafruit NAU7802 breakout uses both channels on the NAU7802. The previous version of the breakout only uses Channel 1.

## CircuitPython Microcontroller Wiring

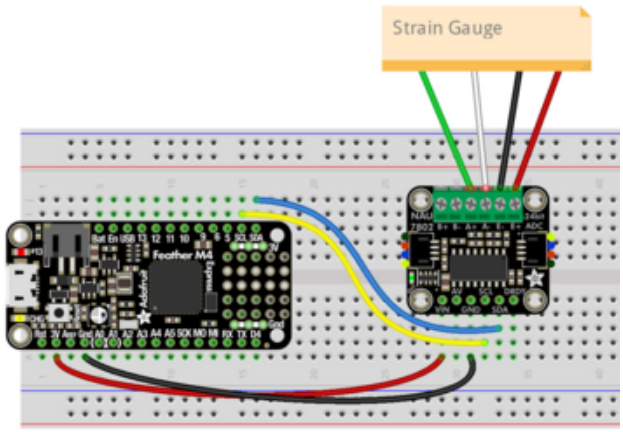
First, wire up a NAU7802 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy **STEMMA QT** (https://adafru.it/Ft4) connectors:

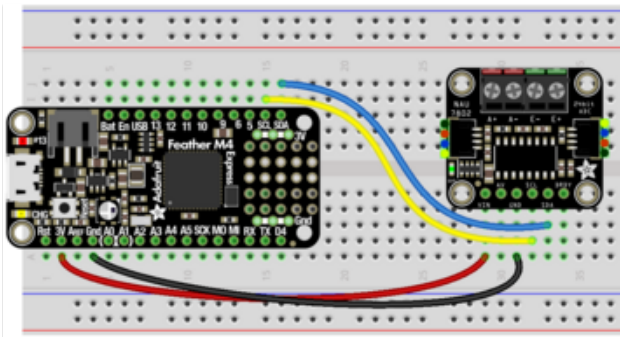



**Board 3V** to **ADC VIN (red wire)**
**Board GND** to **ADC GND (black wire)**
**Board SCL** to **ADC SCL (yellow wire)**
**Board SDA** to **ADC SDA (blue wire)**
**Strain gauge green wire** to **ADC A+**
**Strain gauge white wire** to **ADC A-**
**Strain gauge black wire** to **ADC E-**
**Strain gauge red wire** to **ADC E+**

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



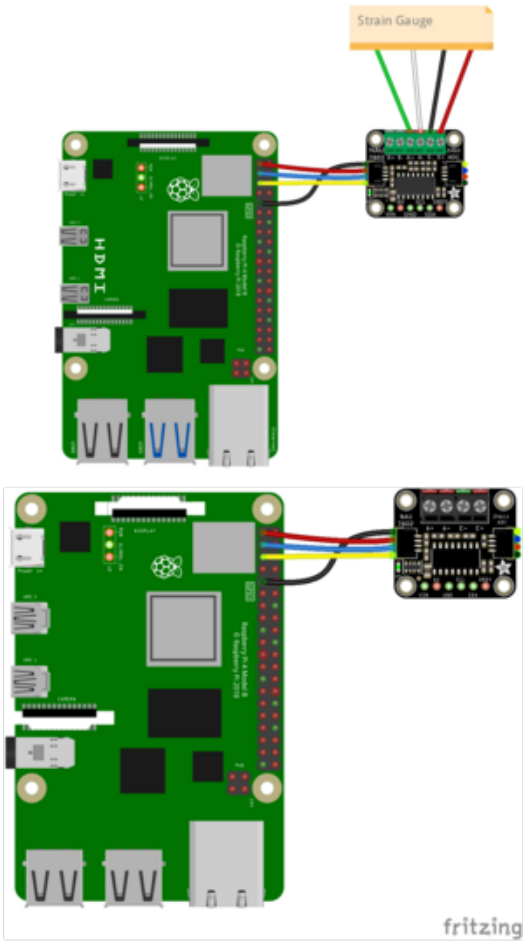



**Board 3V** to **ADC VIN (red wire)**
**Board GND** to **ADC GND (black wire)**
**Board SCL** to **ADC SCL (yellow wire)**
**Board SDA** to **ADC SDA (blue wire)**
**Strain gauge green wire** to **ADC A+**
**Strain gauge white wire** to **ADC A-**
**Strain gauge black wire** to **ADC E-**
**Strain gauge red wire** to **ADC E+**

## Python Computer Wiring

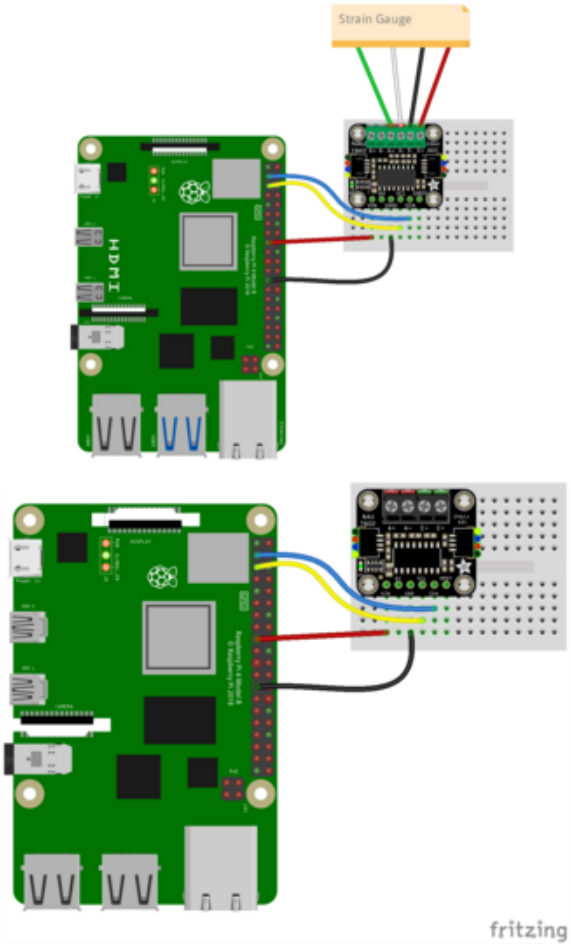
Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (https://adafru.it/BSN).

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




**Pi 3V** to **ADC VIN (red wire)**
**Pi GND** to **ADC GND (black wire)**
**Pi SCL** to **ADC SCL (yellow wire)**
**Pi SDA** to **ADC SDA (blue wire)**
**Strain gauge green wire** to **ADC A+**
**Strain gauge white wire** to **ADC A-**
**Strain gauge black wire** to **ADC E-**
**Strain gauge red wire** to **ADC E+**

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




**Pi 3V** to **ADC VIN (red wire)**
**Pi GND** to **ADC GND (black wire)**
**Pi SCL** to **ADC SCL (yellow wire)**
**Pi SDA** to **ADC SDA (blue wire)**
**Strain gauge green wire** to **ADC A+**
**Strain gauge white wire** to **ADC A-**
**Strain gauge black wire** to **ADC E-**
**Strain gauge red wire** to **ADC E+**

## Python Installation of NAU7802 Library

You'll need to install the **Adafruit_Blinka** library that provides 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 cedargrove-nau7802`

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 Usage

To use with CircuitPython, you need to first install the **CircuitPython_NAU7802** 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 folders and file:

- **adafruit_bus_device/**
- **adafruit_register/**
- **cedargrove_nau7802.mpy**


## Python Usage

Once you have the library installed on your computer, copy or download the following example (https://adafru.it/-pf) to your computer, and run the following, replacing **code.py** with whatever you named the file:

```
python3 code.py
```

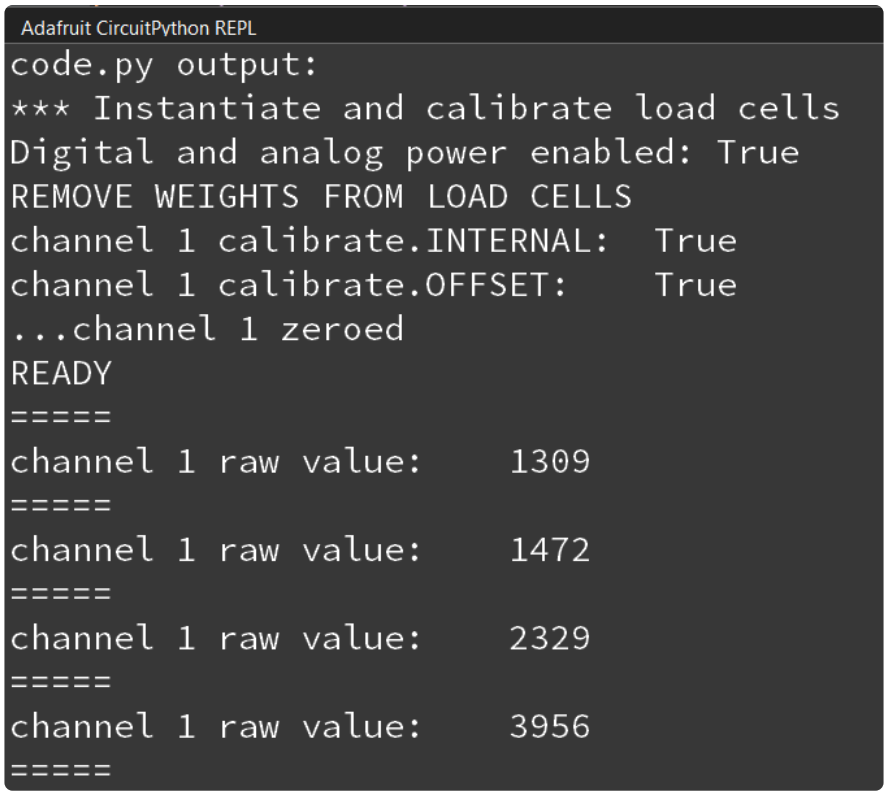
## Example Code

**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: 2023 Cedar Grove Maker Studios
# SPDX-License-Identifier: MIT

"""
nau7802_simpletest.py  2023-01-13 2.0.2  Cedar Grove Maker Studios

Instantiates two NAU7802 channels with default gain of 128 and sample
average count of 2.
"""

import time

import board
```

```
from cedargrove_nau7802 import NAU7802

# Instantiate 24-bit load sensor ADC; two channels, default gain of 128
nau7802 = NAU7802(board.I2C(), address=0x2A, active_channels=2)


def zero_channel():
    """Initiate internal calibration for current channel.Use when scale is started,
    a new channel is selected, or to adjust for measurement drift. Remove weight
    and tare from load cell before executing."""
    print(
        "channel {0:1d} calibrate.INTERNAL: {1:5s}".format(
            nau7802.channel, str(nau7802.calibrate("INTERNAL"))
        )
    )
    print(
        "channel {0:1d} calibrate.OFFSET:   {1:5s}".format(
            nau7802.channel, str(nau7802.calibrate("OFFSET"))
        )
    )
    print(f"...channel {nau7802.channel:1d} zeroed")


def read_raw_value(samples=2):
    """Read and average consecutive raw sample values. Return average raw value."""
    sample_sum = 0
    sample_count = samples
    while sample_count > 0:
        while not nau7802.available():
            pass
        sample_sum = sample_sum + nau7802.read()
        sample_count -= 1
    return int(sample_sum / samples)


# Instantiate and calibrate load cell inputs
print("*** Instantiate and calibrate load cells")
# Enable NAU7802 digital and analog power
enabled = nau7802.enable(True)
print("Digital and analog power enabled:", enabled)

print("REMOVE WEIGHTS FROM LOAD CELLS")
time.sleep(3)

nau7802.channel = 1
zero_channel()  # Calibrate and zero channel
nau7802.channel = 2
zero_channel()  # Calibrate and zero channel

print("READY")

### Main loop: Read load cells and display raw values
while True:
    print("=====")
    nau7802.channel = 1
    value = read_raw_value()
    print(f"channel {nau7802.channel:1.0f} raw value: {value:7.0f}")

    nau7802.channel = 2
    value = read_raw_value()
    print(f"channel {nau7802.channel:1.0f} raw value: {value:7.0f}")
```




Attach a strain gauge with four wires to the four terminal inputs. Now try applying force on the gauge to see the values change!

First you import the necessary modules and libraries. Then you instantiate the sensor on I2C.

Then you're ready to read data from the ADC, including the initial information printed to the serial console.

Finally, inside the loop, you check the value from the ADC.

The terminal inputs raw value will print to the REPL. You'll see the number increase or decrease depending on the amount of force you apply to the strain gauge.

That's all there is to using the NAU7802 with CircuitPython!

---

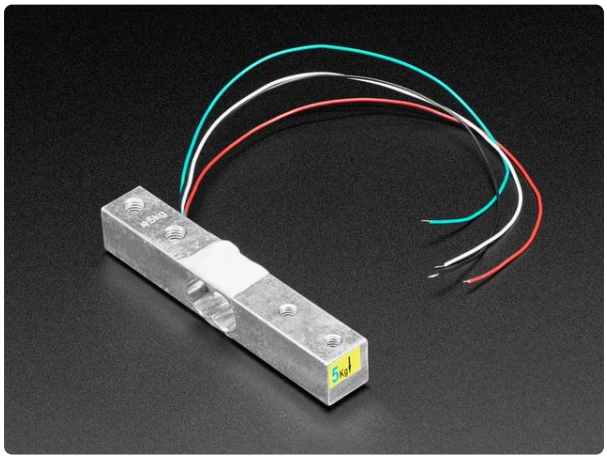
# Python Docs

Python Docs (https://adafru.it/-fc)

# Arduino

As of June 23rd, 2025 – By popular demand! We've updated this design to expose the 'second' ADC input (B+ and B-), so we've updated this PCB to now come with a 6-pin terminal block port, instead of 4-pin.

Using the NAU7802 with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller with a strain gauge, installing the Adafruit_NAU7802 (https://adafru.it/-fd) library and running the provided example code.

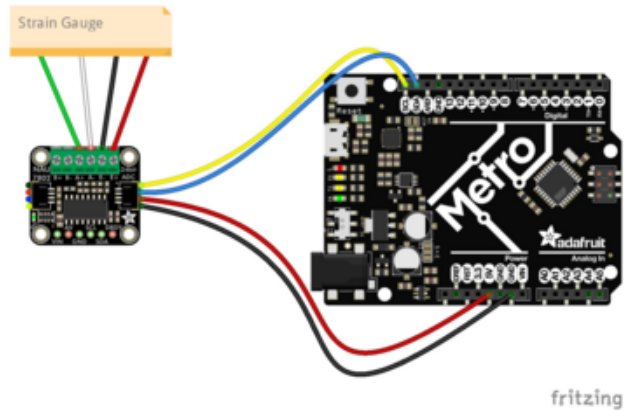


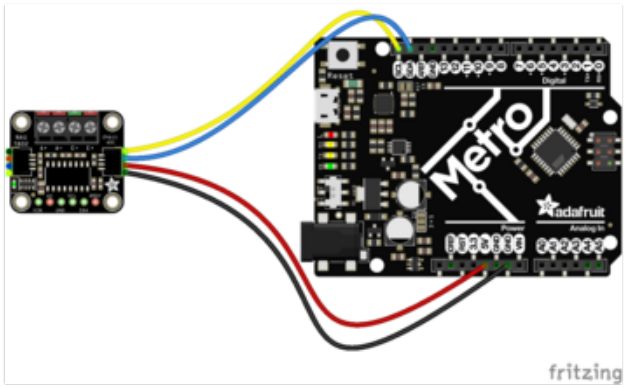
Strain Gauge Load Cell - 4 Wires - 5Kg
A strain gauge is a type of electronic sensor used to measure force or strain (big surprise there). They are made of an insulating flexible backing with a metallic...
https://www.adafruit.com/product/4541

## 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 NAU7802 VIN.

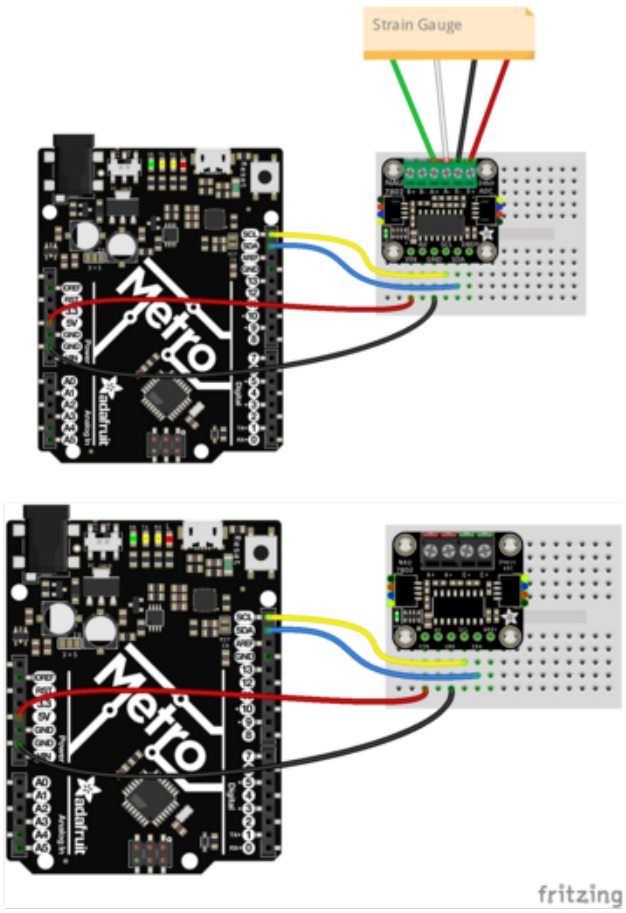
Here is an Adafruit Metro wired up to the NAU7802 using the STEMMA QT connector:




**Board 5V** to **ADC VIN (red wire)**
**Board GND** to **ADC GND (black wire)**
**Board SCL** to **ADC SCL (yellow wire)**
**Board SDA** to **ADC SDA (blue wire)**
**Strain gauge green wire** to **ADC A+**
**Strain gauge white wire** to **ADC A-**
**Strain gauge black wire** to **ADC E-**
**Strain gauge red wire** to **ADC E+**

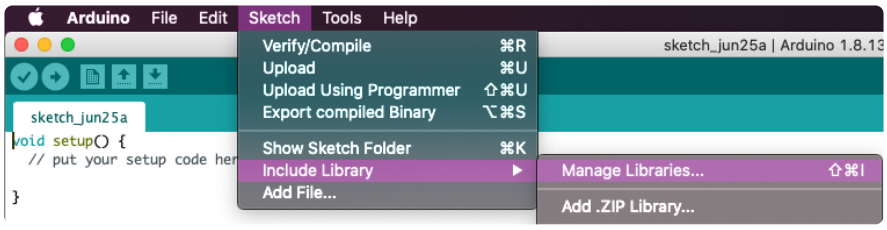
Here is an Adafruit Metro wired up using a solderless breadboard:



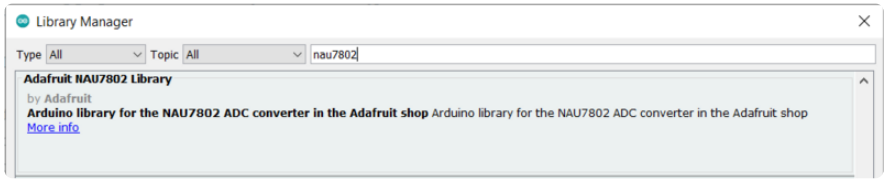

**Board 5V** to **ADC VIN (red wire)**
**Board GND** to **ADC GND (black wire)**
**Board SCL** to **ADC SCL (yellow wire)**
**Board SDA** to **ADC SDA (blue wire)**
**Strain gauge green wire** to **ADC A+**
**Strain gauge white wire** to **ADC A-**
**Strain gauge black wire** to **ADC E-**
**Strain gauge red wire** to **ADC E+**

## Library Installation

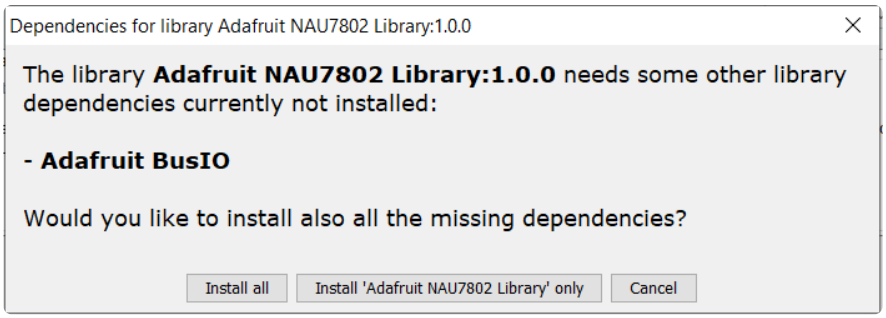
You can install the **NAU7802** library for Arduino using the Library Manager in the Arduino IDE.




Click the **Manage Libraries ...** menu item, search for **NAU7802**, and select the **Adafruit NAU7802 Library** library:




If asked about dependencies, click "Install all".




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

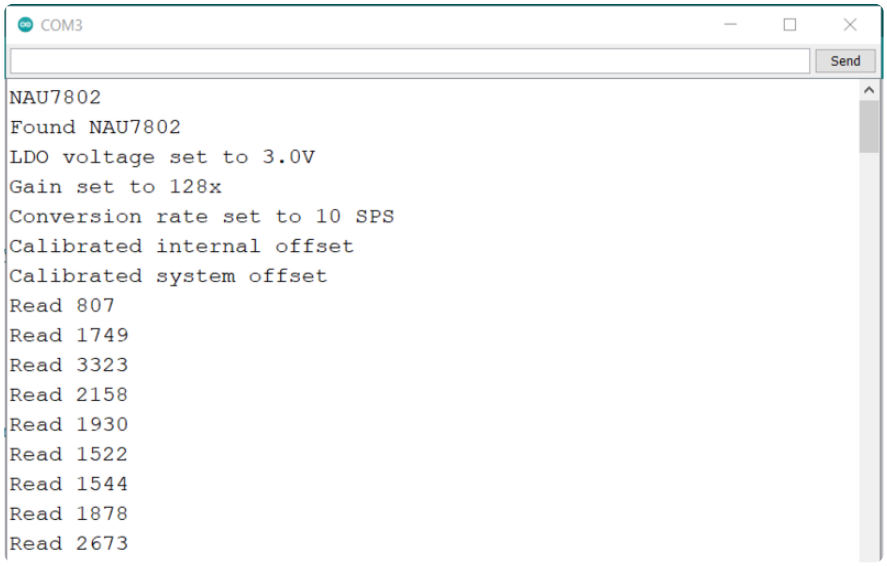
If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

## Load Example

Open up **File -> Examples -> Adafruit NAU7802 Library -> nau7802_test** and upload to your Arduino wired to the sensor.

```
#include <Adafruit_NAU7802.h>

Adafruit_NAU7802 nau;

void setup() {
  Serial.begin(115200);
  Serial.println("NAU7802");
  if (! nau.begin()) {
    Serial.println("Failed to find NAU7802");
    while (1) delay(10);  // Don't proceed.
  }
  Serial.println("Found NAU7802");

  nau.setLDO(NAU7802_3V0);
  Serial.print("LDO voltage set to ");
  switch (nau.getLDO()) {
    case NAU7802_4V5:  Serial.println("4.5V"); break;
    case NAU7802_4V2:  Serial.println("4.2V"); break;
    case NAU7802_3V9:  Serial.println("3.9V"); break;
    case NAU7802_3V6:  Serial.println("3.6V"); break;
    case NAU7802_3V3:  Serial.println("3.3V"); break;
    case NAU7802_3V0:  Serial.println("3.0V"); break;
    case NAU7802_2V7:  Serial.println("2.7V"); break;
    case NAU7802_2V4:  Serial.println("2.4V"); break;
    case NAU7802_EXTERNAL:  Serial.println("External"); break;
  }

  nau.setGain(NAU7802_GAIN_128);
  Serial.print("Gain set to ");
  switch (nau.getGain()) {
    case NAU7802_GAIN_1:  Serial.println("1x"); break;
    case NAU7802_GAIN_2:  Serial.println("2x"); break;
    case NAU7802_GAIN_4:  Serial.println("4x"); break;
    case NAU7802_GAIN_8:  Serial.println("8x"); break;
```

```
    case NAU7802_GAIN_16:  Serial.println("16x"); break;
    case NAU7802_GAIN_32:  Serial.println("32x"); break;
    case NAU7802_GAIN_64:  Serial.println("64x"); break;
    case NAU7802_GAIN_128:  Serial.println("128x"); break;
  }

  nau.setRate(NAU7802_RATE_10SPS);
  Serial.print("Conversion rate set to ");
  switch (nau.getRate()) {
    case NAU7802_RATE_10SPS:  Serial.println("10 SPS"); break;
    case NAU7802_RATE_20SPS:  Serial.println("20 SPS"); break;
    case NAU7802_RATE_40SPS:  Serial.println("40 SPS"); break;
    case NAU7802_RATE_80SPS:  Serial.println("80 SPS"); break;
    case NAU7802_RATE_320SPS:  Serial.println("320 SPS"); break;
  }

  // Take 10 readings to flush out readings
  for (uint8_t i=0; i<10; i++) {
    while (! nau.available()) delay(1);
    nau.read();
  }

  while (! nau.calibrate(NAU7802_CALMOD_INTERNAL)) {
    Serial.println("Failed to calibrate internal offset, retrying!");
    delay(1000);
  }
  Serial.println("Calibrated internal offset");

  while (! nau.calibrate(NAU7802_CALMOD_OFFSET)) {
    Serial.println("Failed to calibrate system offset, retrying!");
    delay(1000);
  }
  Serial.println("Calibrated system offset");
}

void loop() {
  while (! nau.available()) {
    delay(1);
  }
  int32_t val = nau.read();
  Serial.print("Read "); Serial.println(val);
}
```




Attach a strain gauge with four wires to the four terminal inputs on the NAU7802. Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial**

**Monitor**) at 115200 baud. You should see the values from the ADC being printed out. You'll see the value increase or decrease depending on the amount of force you are applying to the strain gauge.

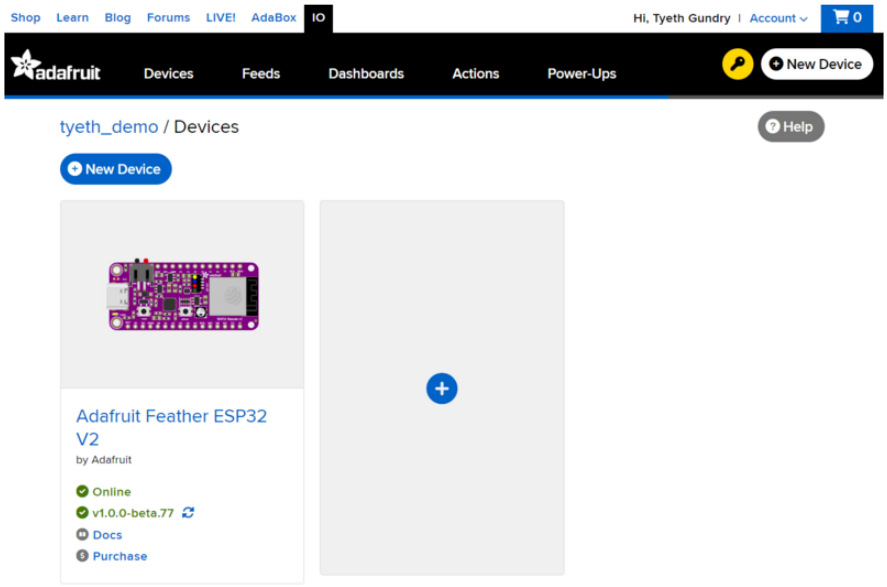
---

# Arduino Docs

Arduino Docs (https://adafru.it/-fd)

---

# 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.

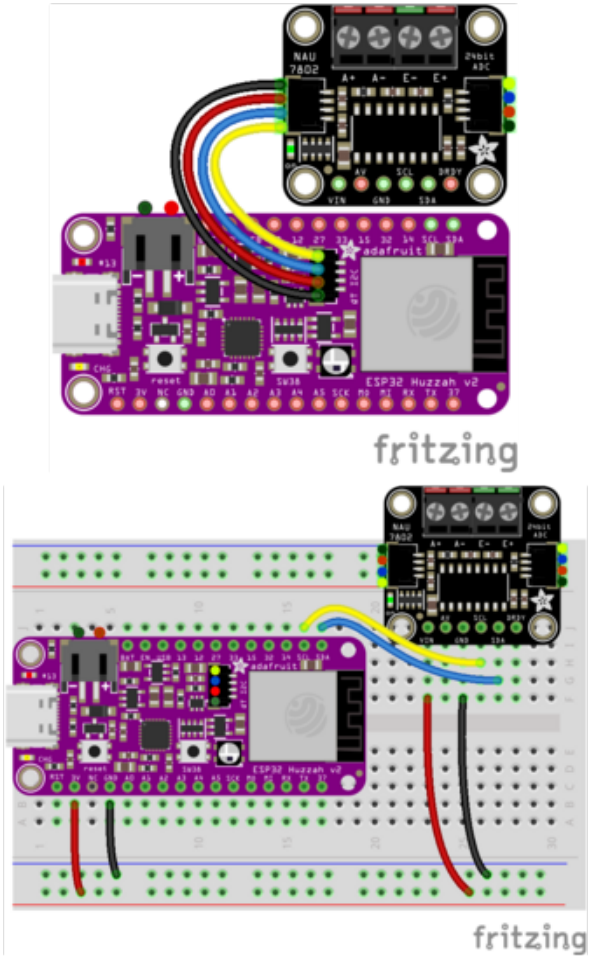
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

**Quickstart: Adafruit IO WipperSnapper**

https://adafru.it/Vfd

## Wiring

First, wire up an NAU7802 to your board exactly as follows. Here is an example of the NAU7802 wired to an Adafruit ESP32 Feather V2 (http://adafru.it/5400) using I2C with a STEMMA QT cable (no soldering required) (http://adafru.it/4210)

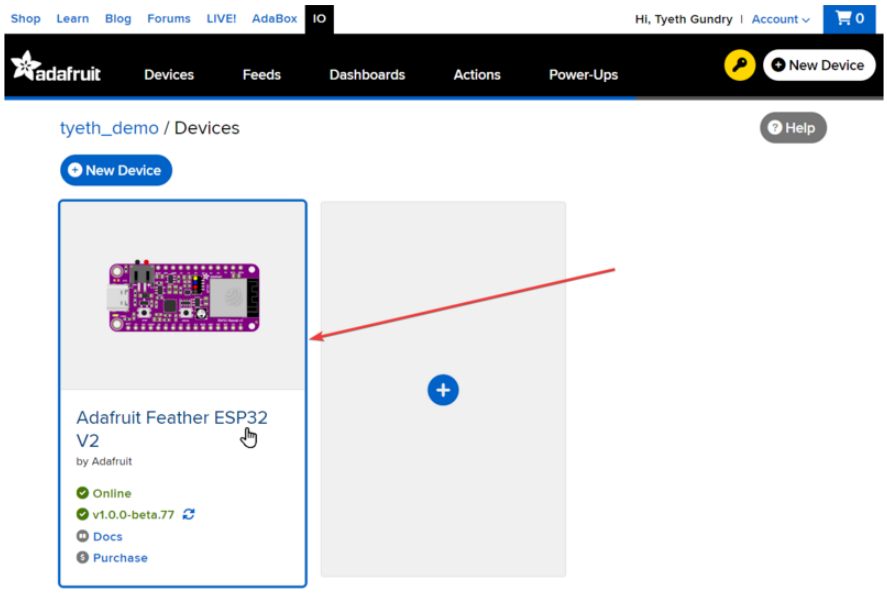



**Board 3V** to **sensor VIN (red wire on STEMMA QT)**
**Board GND** to **sensor GND (black wire on STEMMA QT)**
**Board SCL** to **sensor SCL (yellow wire on STEMMA QT)**
**Board SDA** to **sensor SDA (blue wire on STEMMA QT)**

## Usage

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

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.



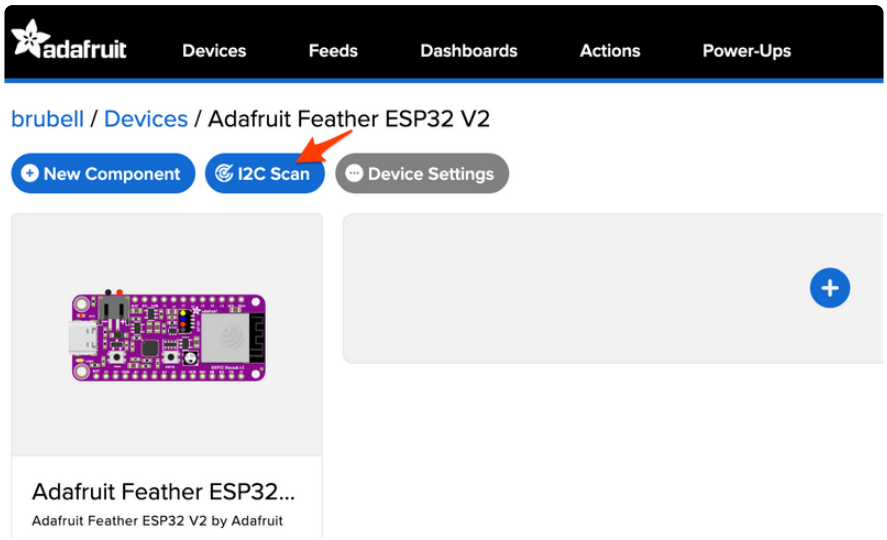


On the device page, quickly **check that you're running the latest version of the WipperSnapper firmware**.

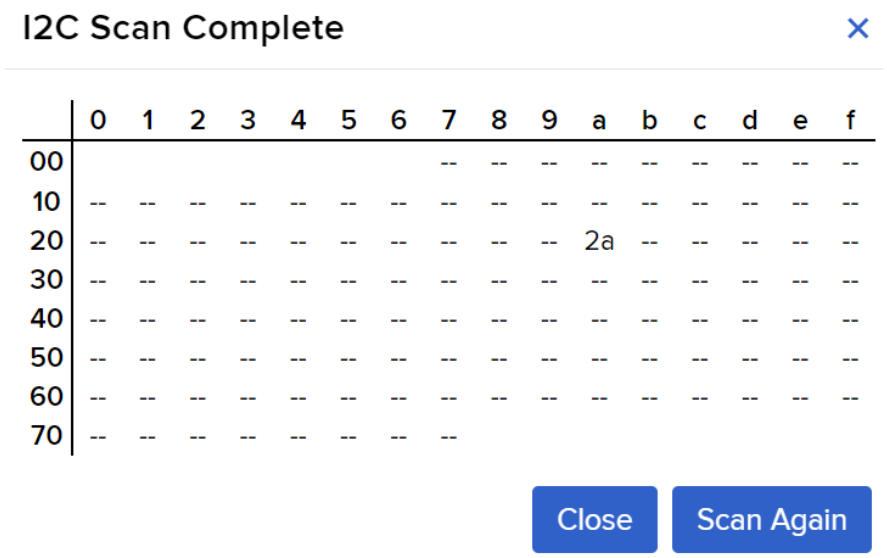
The device tile on the left indicates the version number of the firmware running on the connected board.

**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 NAU7802's default I2C address of `0x2A` pop-up in the I2C scan list.




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

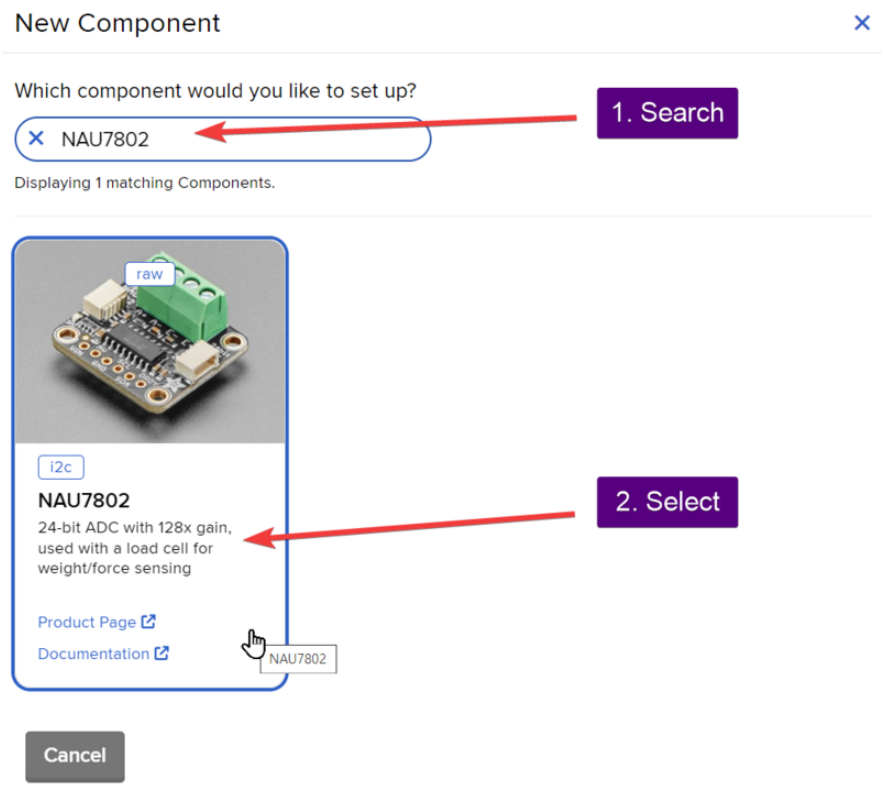
First, double-check the connection and/or wiring between the sensor and the board.

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.

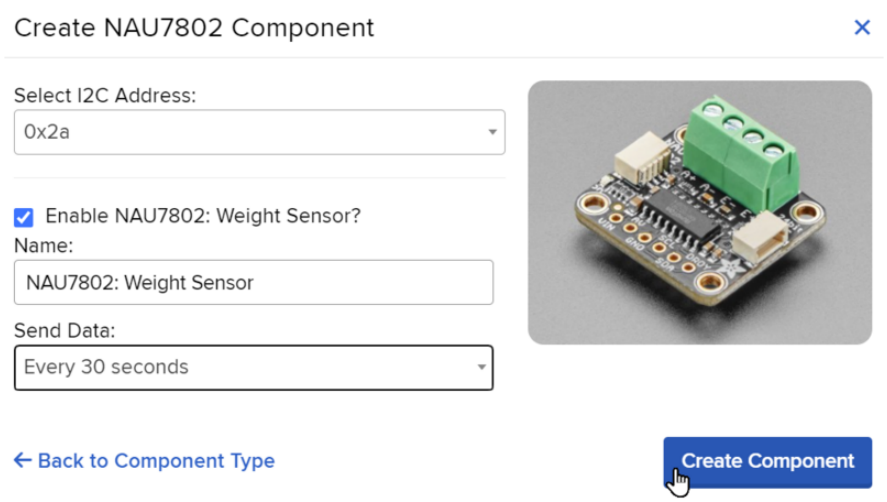

Adafruit IO supports a large amount of components. To quickly find your sensor, type `NAU7802` into the search bar, then select the **NAU7802** component.



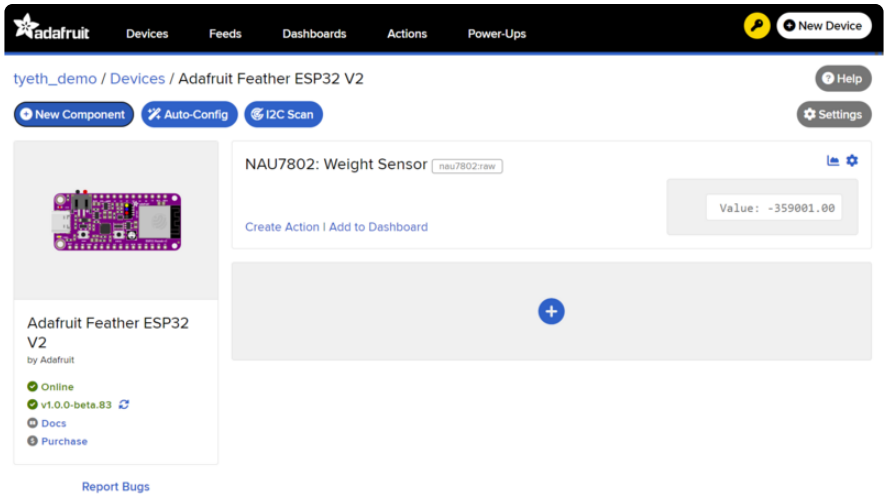

On the component configuration page, the NAU7802's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the NAU7802 sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

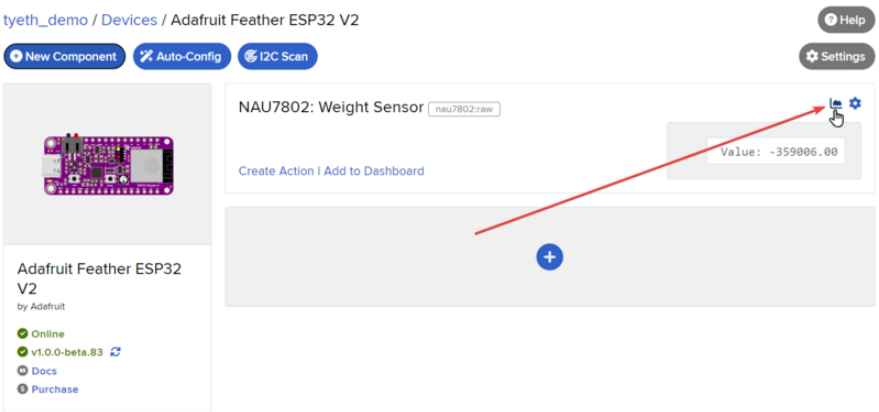
For this example, set the **Send Every** interval to every 30 seconds.



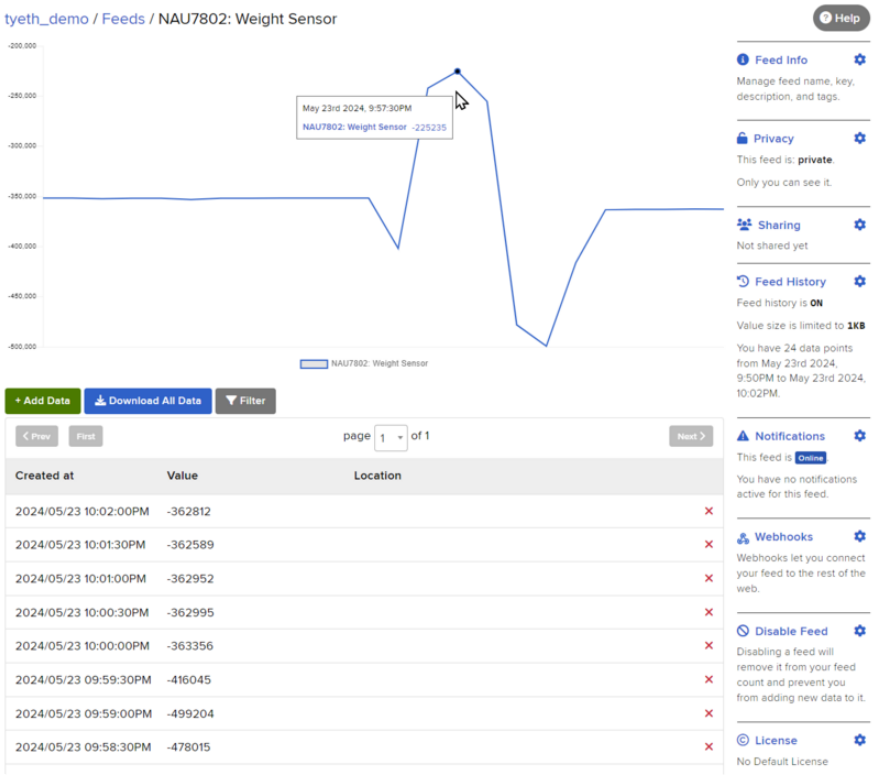

Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.




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](https://adafru.it/10aZ).




# Downloads

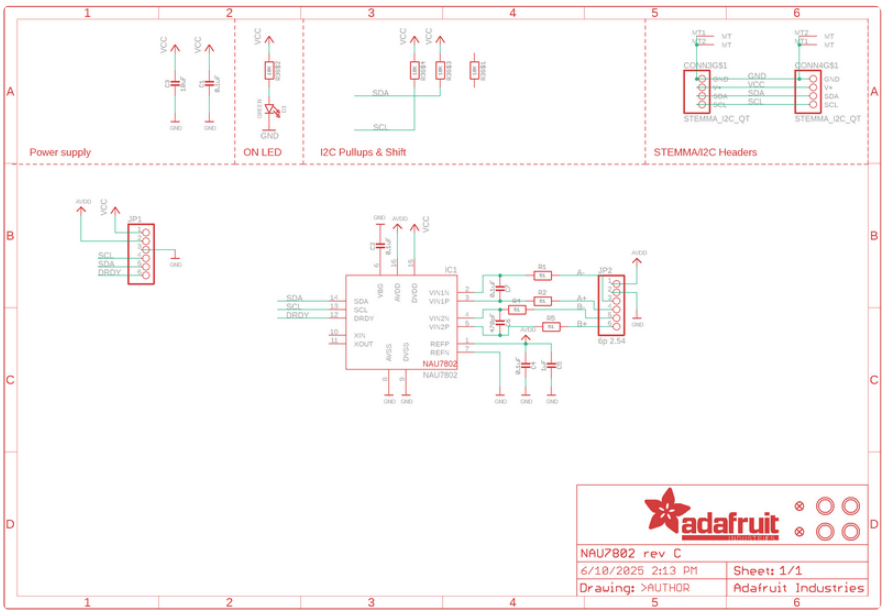
As of June 23rd, 2025 – By popular demand! We've updated this design to expose the 'second' ADC input (B+ and B-), so we've updated this PCB to now come with a 6-pin terminal block port, instead of 4-pin.

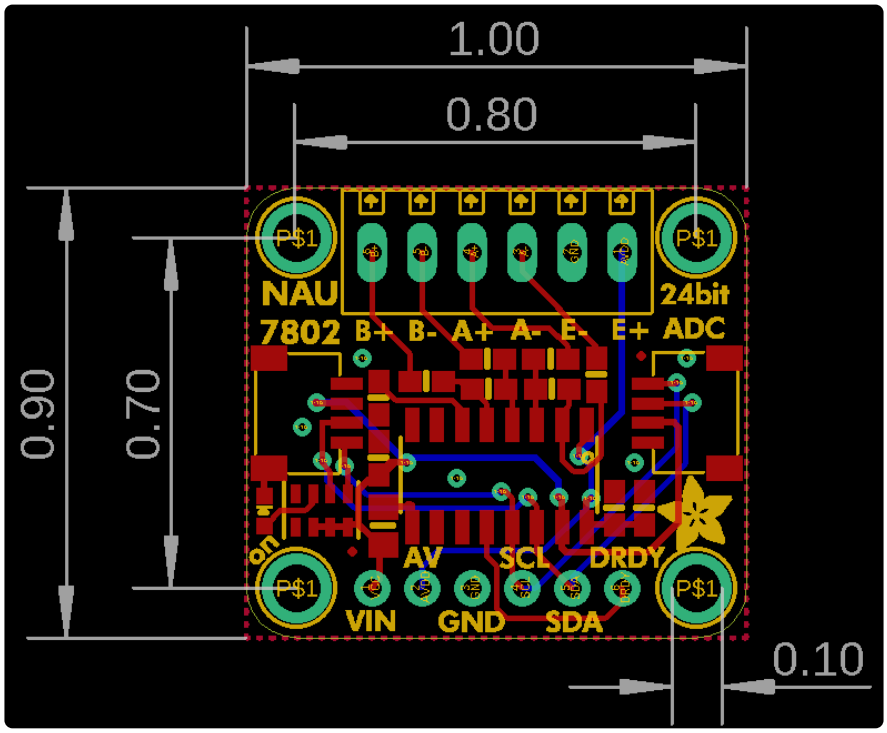
## Files

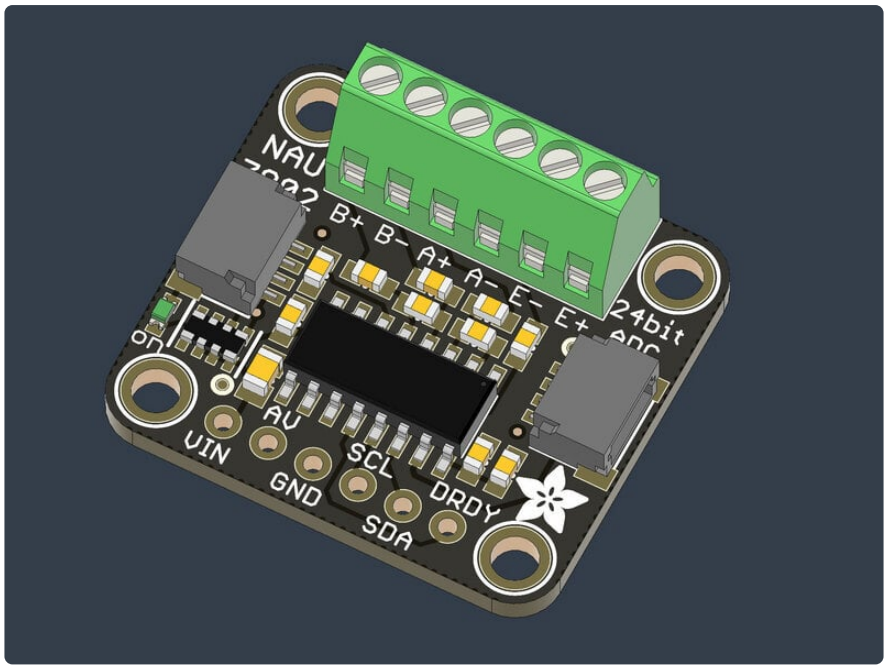
- [NAU7802 Datasheet](https://adafru.it/-pA)
- [EagleCAD PCB files on GitHub](https://adafru.it/-pB)
- [Fritzing object in the Adafruit Fritzing Library](https://adafru.it/-pC)
- [Rev C Fritzing object](https://adafru.it/1alj)
- [3D models on GitHub](https://adafru.it/18fl)

# Schematic and Fab Print

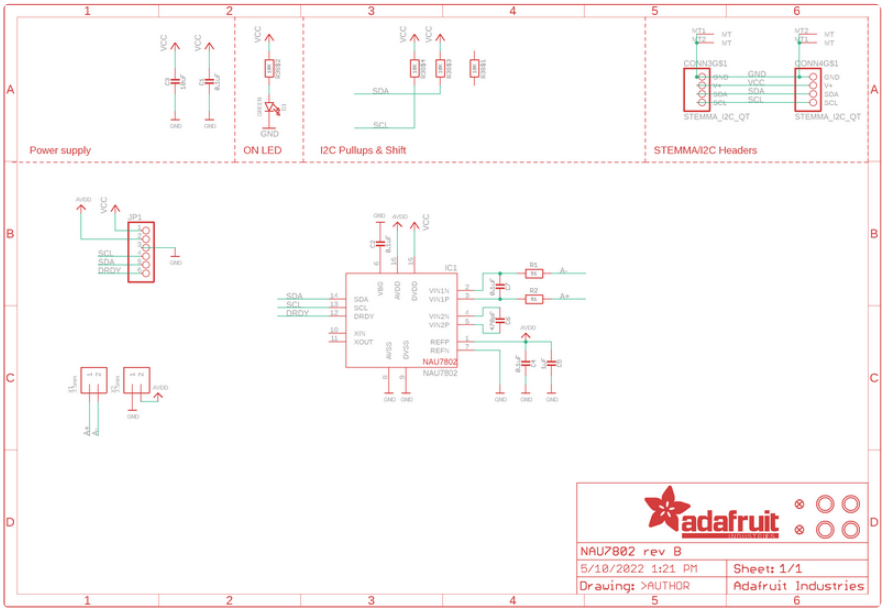
## Rev C

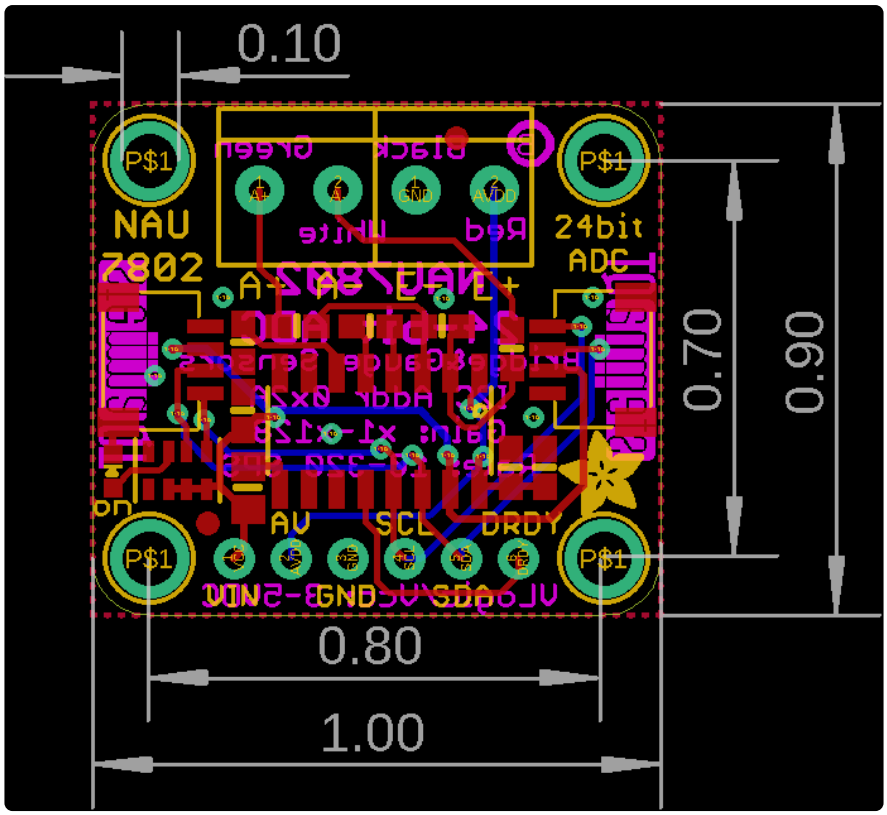


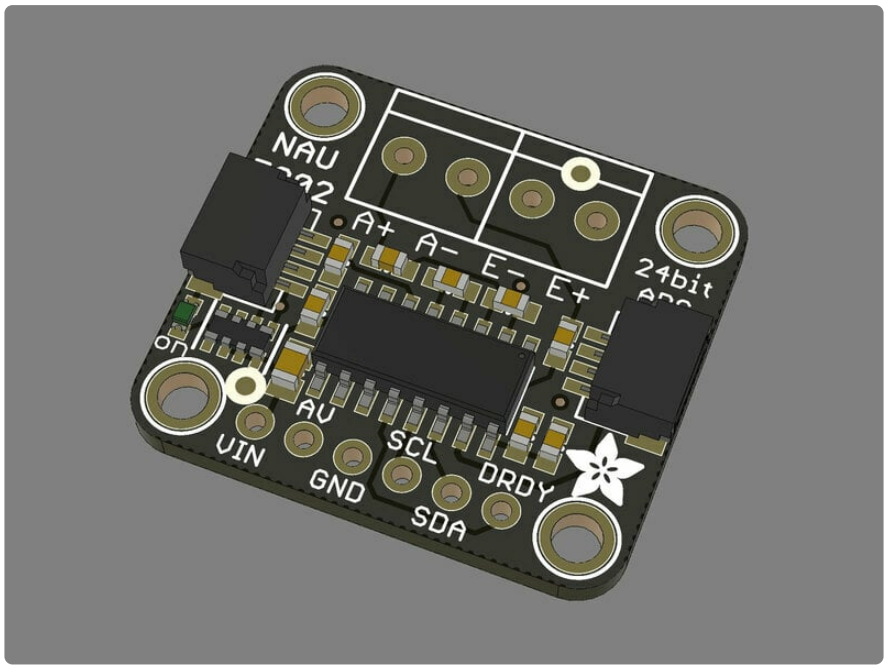




B+ B- A+ A- E- E+
ON
VIN
GND
SDA
SCL
DRDY
24bit

## Rev B


Power supply
ON LED
I2C Pullups & Shift
STEMMA/I2C Headers
NAU7802 rev B
5/10/2022 1:21 PM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries


0.10
P$1
P$1
NAU
7802
24bit
ADC
A+
A-
E-
E+
0.70
0.90
AV
SCL
DRDY
P$1
P$1
VIN
GND
SDA
0.80
1.00


NAU
A+
A-
E-
E+
24bit
AV
SCL
DRDY
VIN
GND
SDA