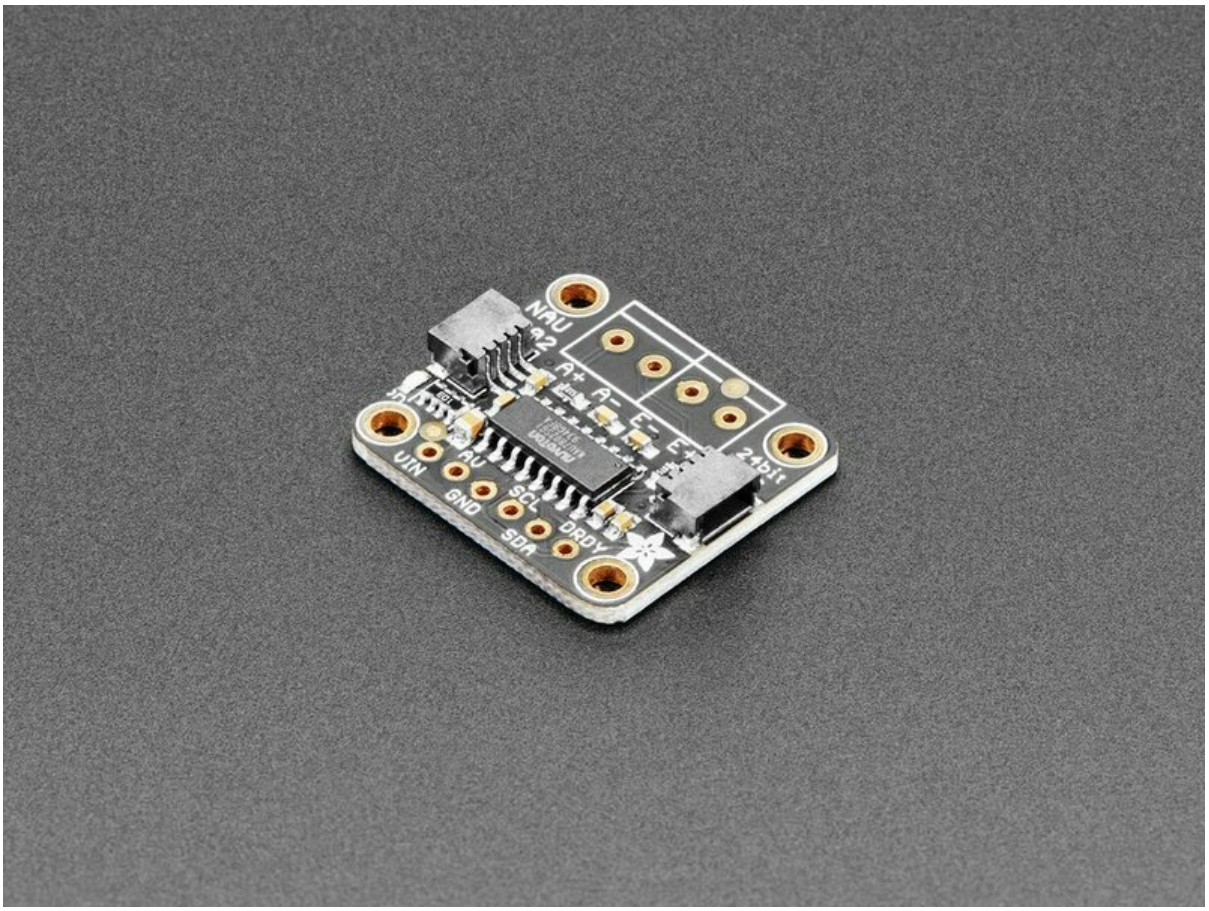
Adafruit NAU7802 24-Bit ADC - STEMMA QT / Qwiic

Created by Liz Clark



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

Last updated on 2024-03-08 04:05:58 PM EST

Table of Contents

Overview 3

Pinouts 5

- Power Pins
- I2C Logic Pins
- Terminal Block Pins
- Data Ready Output Pin
- Power LED

Python & CircuitPython 6

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

Python Docs 11

Arduino 12

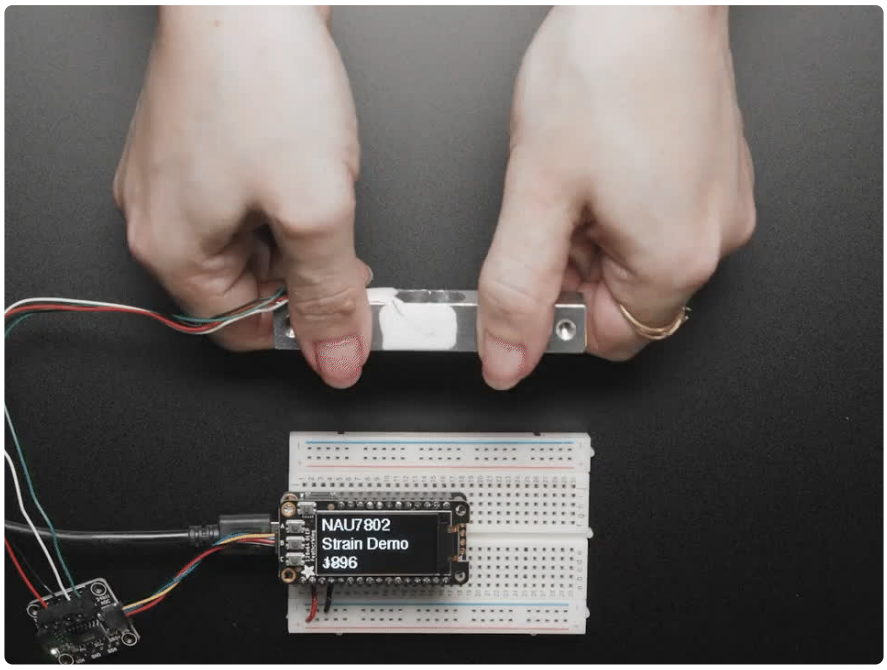
- Wiring
- Library Installation
- Load Example

Arduino Docs 15

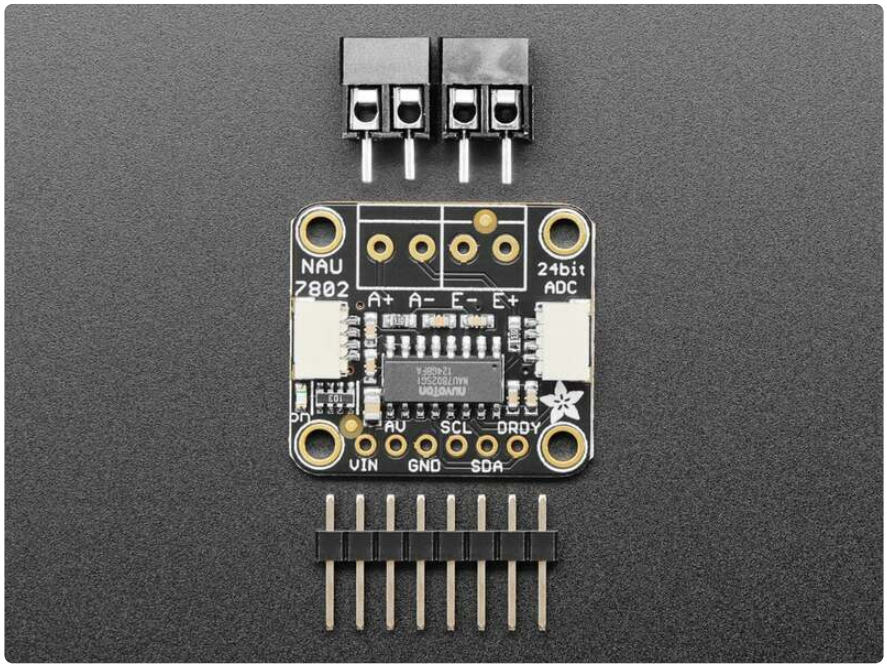
Downloads 15

- Files
- Schematic and Fab Print

Overview



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).

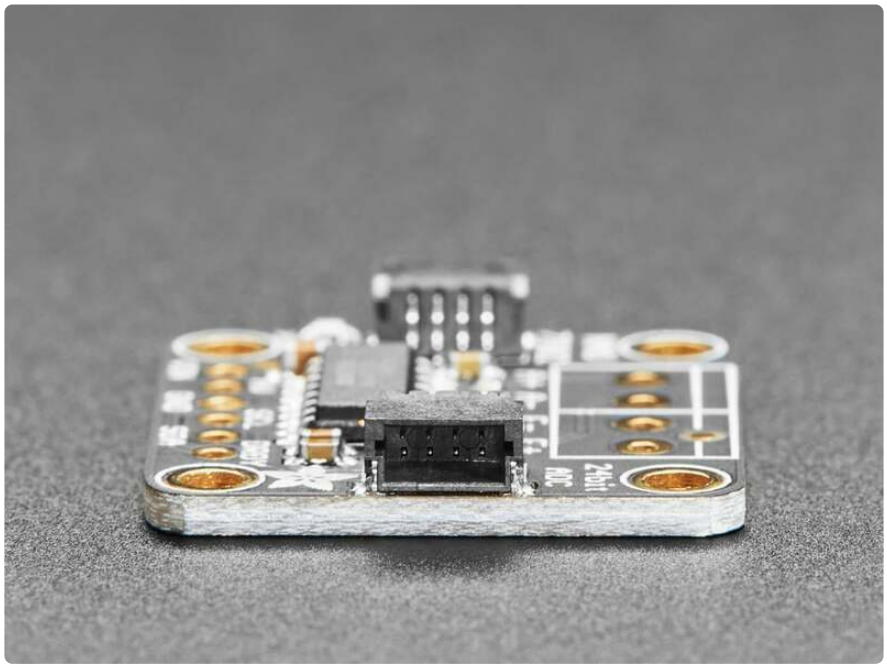


Each breakout comes with a NAU7802 ADC chip, plus some support circuitry, and 4 terminal blocks that can be used to connect a 4-wire sensor.

back

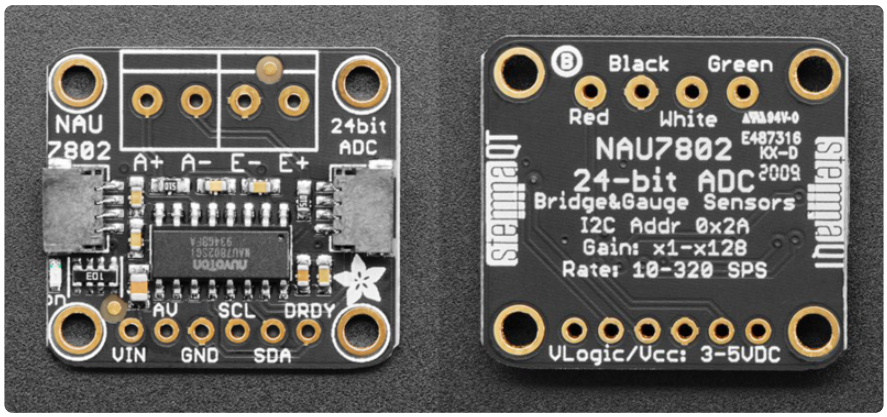
The **E-** pad connects to ground (often a black wire), the **E+** pad connects to a generated voltage from the NAU7802 that can be configured from 2.4V to 4.0V for a solid positive reference (often a red wire).

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, if multiple sensors are desired on one I2C bus, a multiplexer can be used (https://adafru.it/-af).

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.

- **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

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

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

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.
- **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.

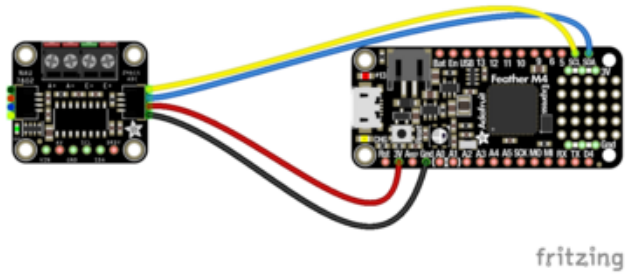
Python & CircuitPython

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

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).

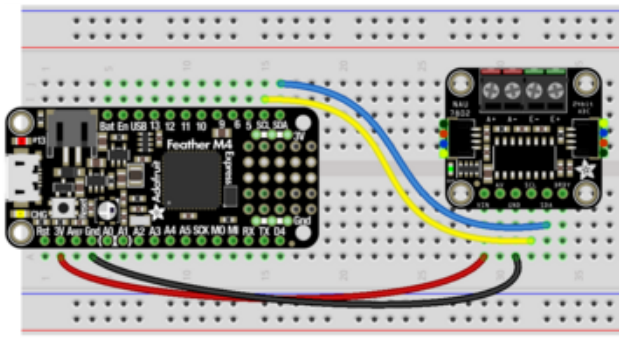
CircuitPython Microcontroller Wiring

First, wire up 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)**

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

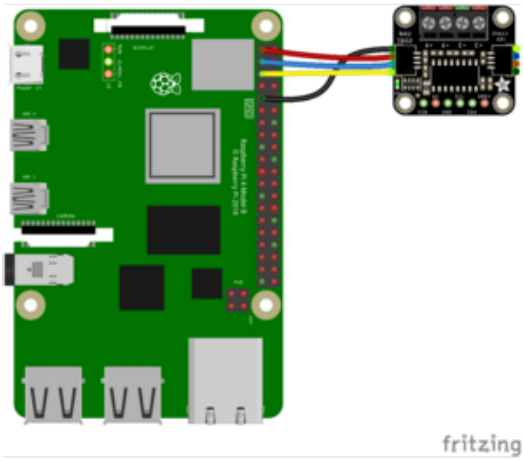


Board 3V to **ADC VIN (red wire)**
Board GND to **ADC GND (black wire)**
Board SCL to **ADC SCL (yellow wire)**
Board SDA to **ADC SDA (blue wire)**

Python Computer Wiring

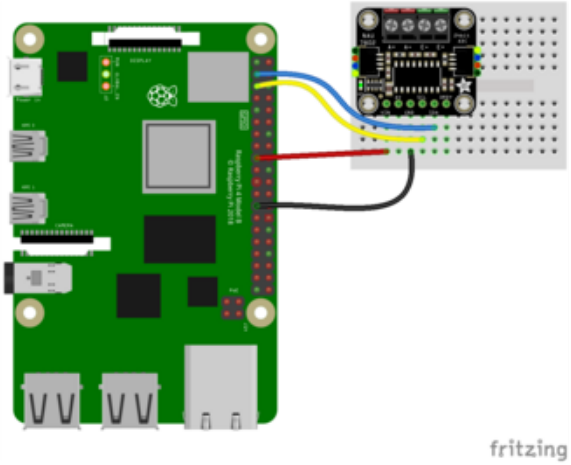
Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

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)**

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)**

Python Installation of NAU7802 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:

- `git clone https://github.com/CedarGroveStudios/CircuitPython_NAU7802`

This clones the **CircuitPython_NAU7802** repository from GitHub. Then, you'll need to copy the **cedargrove_nau7802.py** library file to your Python packages folder running the following command:

- `sudo cp -r ~/CircuitPython_NAU7802/cedargrove_nau7802.py /usr/local/lib/python3.9/dist-packages`

CircuitPython Usage

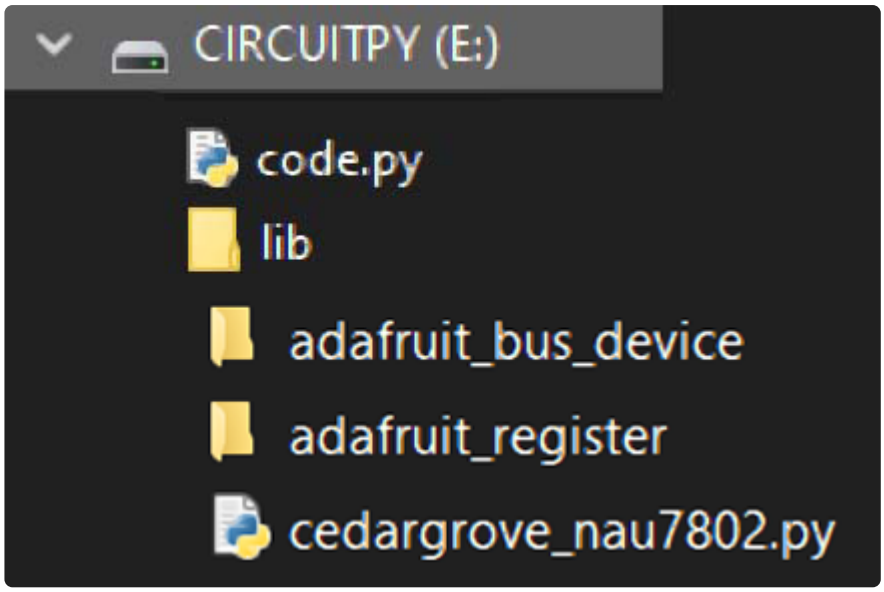
To use with CircuitPython, you need to first install the NAU7802 library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

1. First, download the latest CircuitPython Community Bundle (https://adafru.it/ENC)
2. Copy the **cedargrove_nau7802.mpy** file to the **CIRCUITPY/lib** folder.
3. Copy the **nau7802_simpletest.py** file from the **examples** folder to the **CIRCUITPY** drive.
4. Rename **nau7802_simpletest.py** to **code.py**.

Next, download the latest CircuitPython Library bundle (https://adafru.it/ENC). Copy the **adafruit_bus_device** and **adafruit_register** library folders to the **CIRCUITPY/lib** folder.

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
```

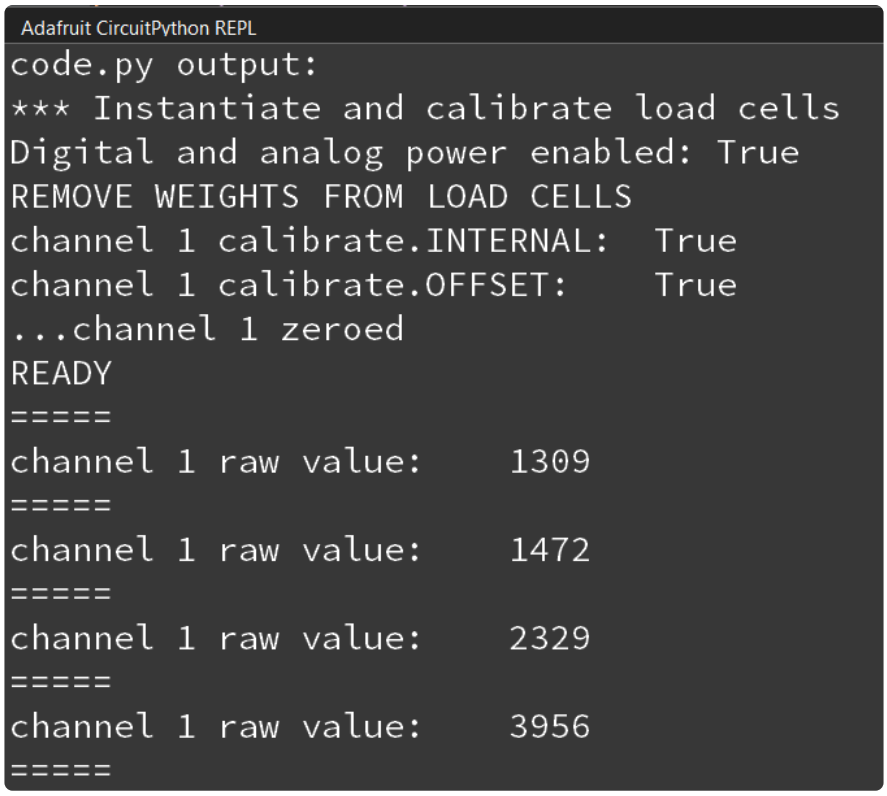
Example Code

nau7802_simpletest.py

https://adafru.it/-pf

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.



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)

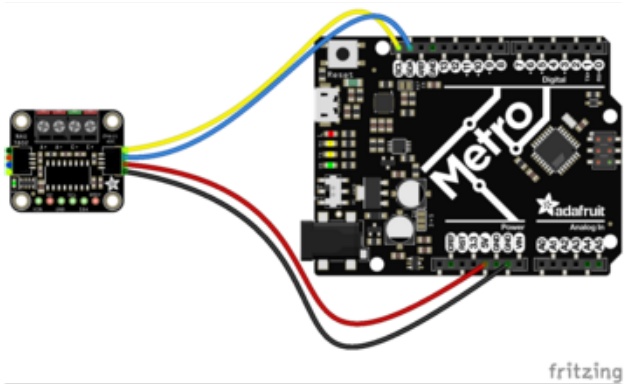
Arduino

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

Wiring

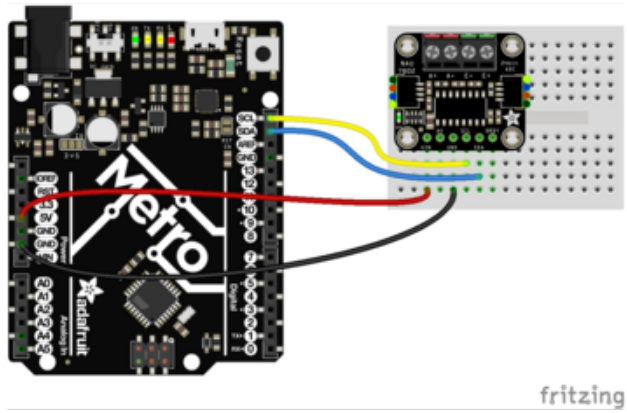
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the NAU7802 VIN.

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)**

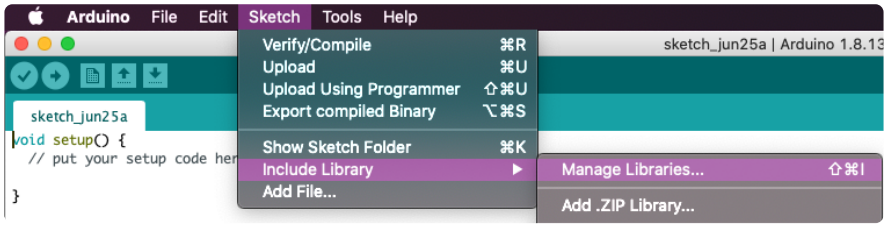
Here is an Adafruit Metro wired up using a solderless breadboard:



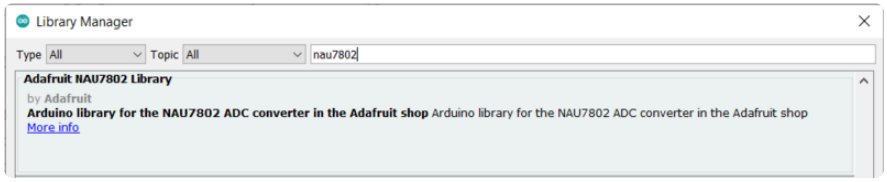
Board 5V to **ADC VIN (red wire)**
Board GND to **ADC GND (black wire)**
Board SCL to **ADC SCL (yellow wire)**
Board SDA to **ADC SDA (blue wire)**

Library Installation

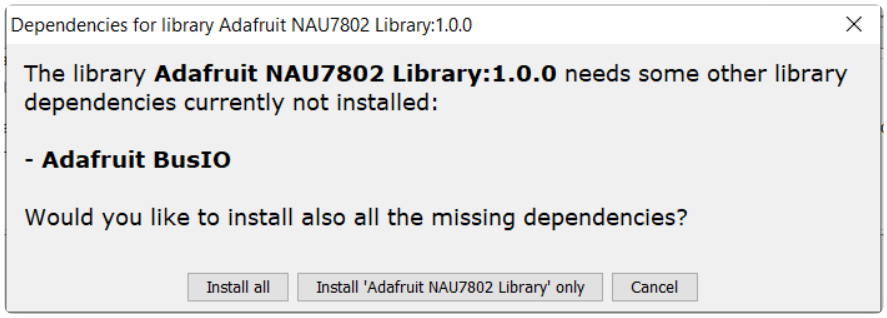
You can install the **NAU7802** library for Arduino using the Library Manager in the Arduino IDE.



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");
  }
  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;
  }

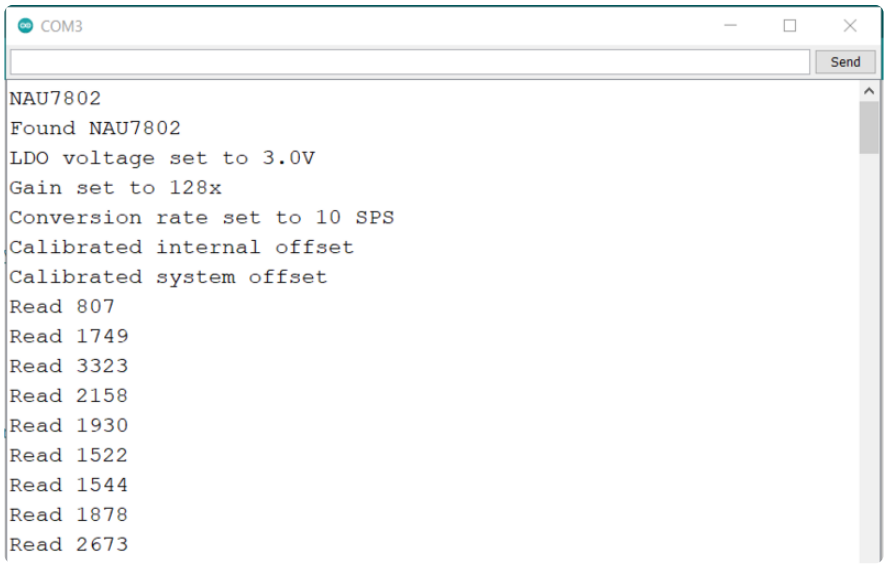
  // Take 10 readings to flush out readings
  for (uint8_t i=0; i<10; i++) {
    while (! nau.available()) delay(1);
    nau.read();
  }

  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

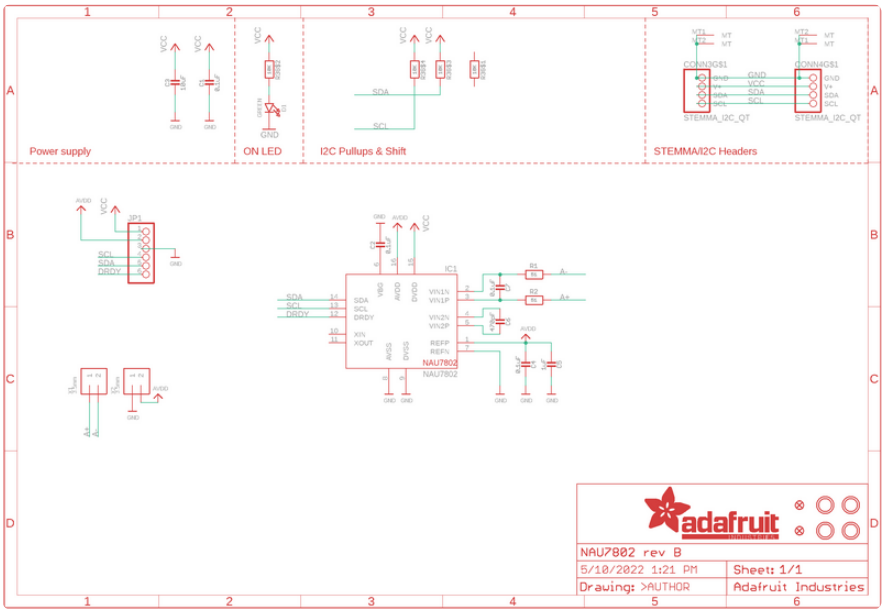
Arduino Docs (https://adafru.it/-fd)

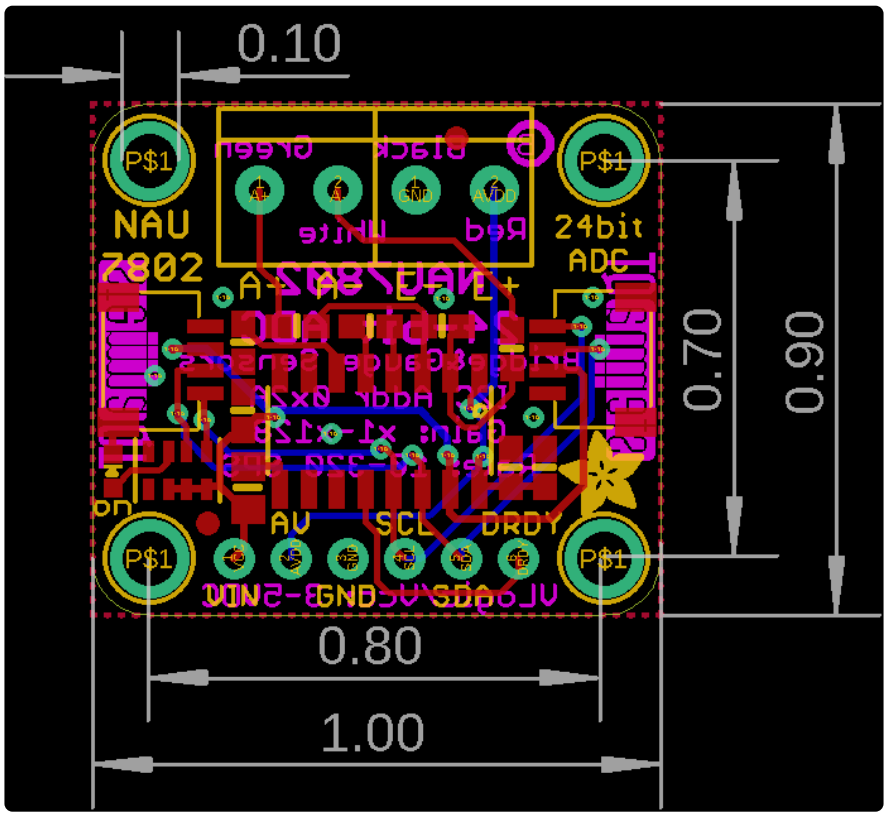
Downloads

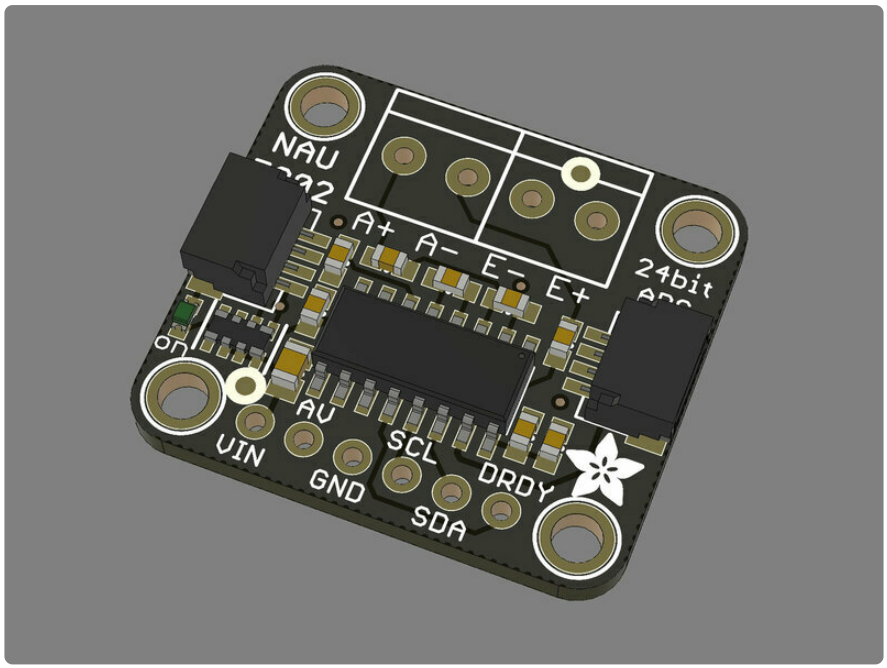
Files

- 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)
- 3D models on GitHub (https://adafru.it/18fl)

Schematic and Fab Print






A+
A-
E-
E+
24bit
VIN
3V
GND
SCL
SDA
DRDY