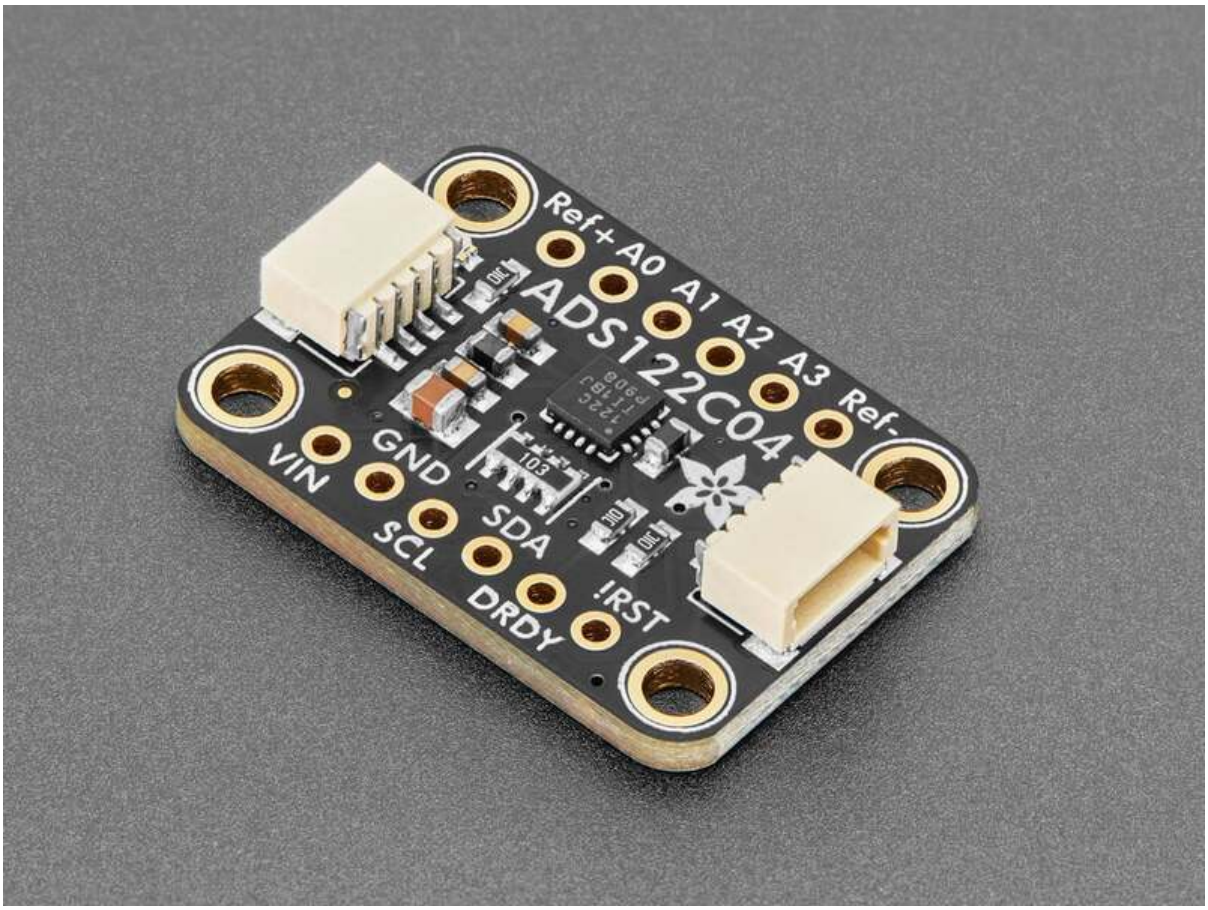
Adafruit ADS122C04 24-Bit ADC

Created by Liz Clark



https://learn.adafruit.com/adafruit-ads122c04-24-bit-adc

Last updated on 2026-03-27 04:10:13 PM UTC

Table of Contents

Overview 3

Pinouts 6

- Power Pins
- I2C Logic Pins
- ADC Inputs
- Address Jumpers
- Other Pins
- Power LED and Jumper

CircuitPython and Python 8

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of ADS122C04 Library
- CircuitPython Usage
- Python Usage
- Example Code

Python Docs 14

Arduino 14

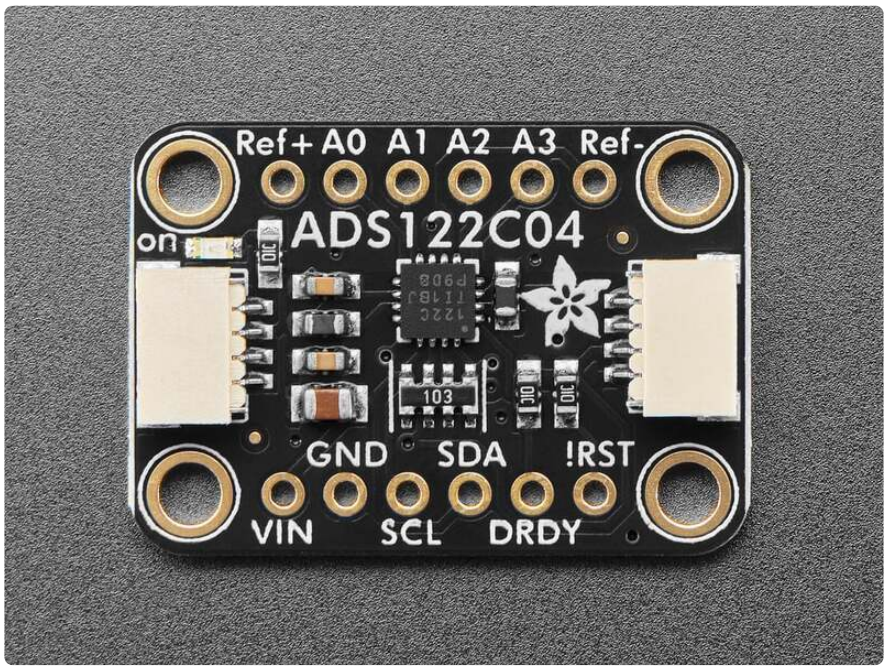
- Wiring
- Library Installation
- Example Code

Arduino Docs 20

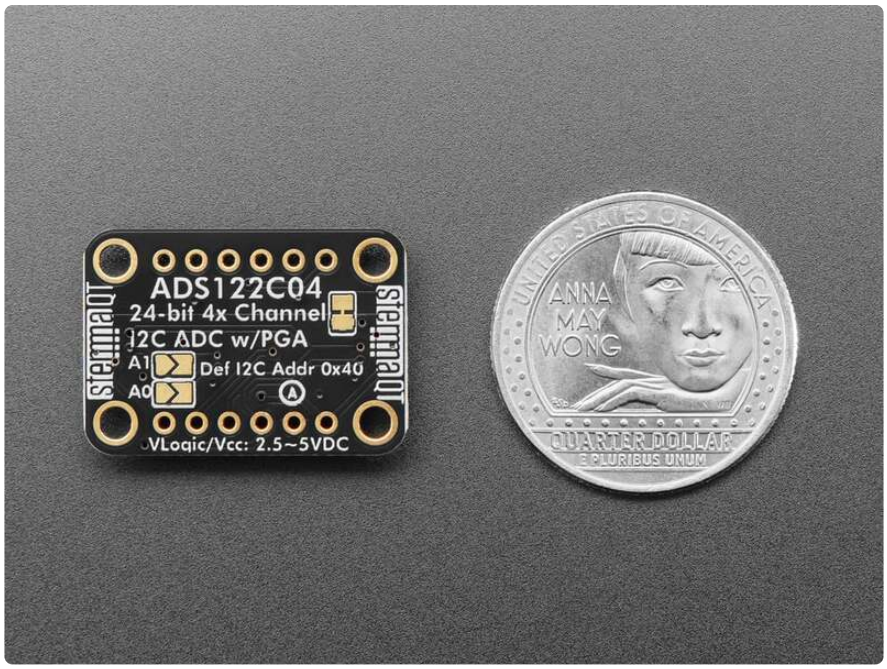
Downloads 20

- Files
- Schematic and Fab Print

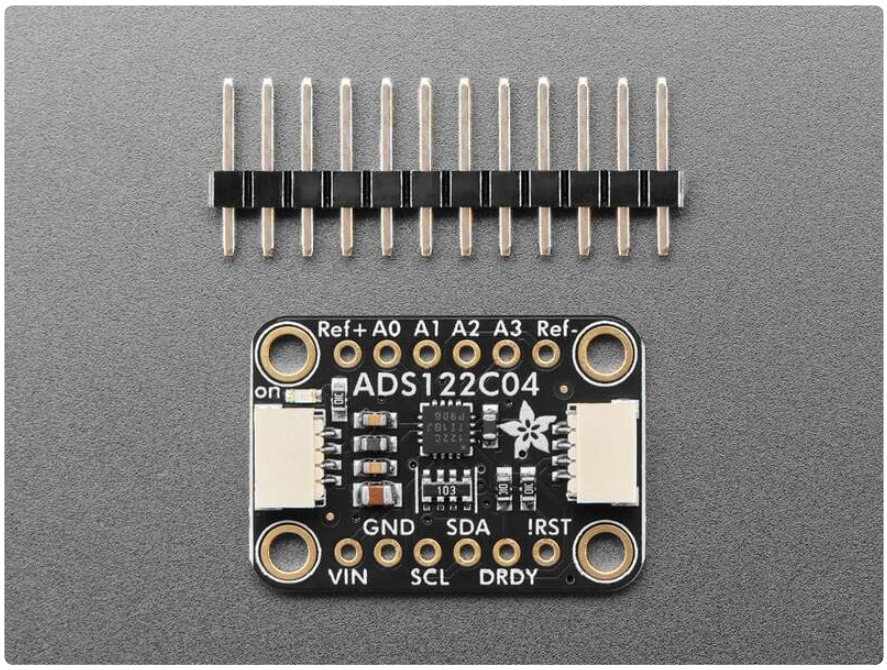
Overview



The **Adafruit ADS122C04 24-Bit ADC with 4 Channel 2-kSPS** is an excellent way to take very high-precision measurements of single-ended or differential analog signals over I2C. Most 24-bit ADCs are found on strain-gauge sensors like the NAU7802 (http://adafru.it/4538) and don't have high speeds or configurations. And most fast-ish I2C ADCs like the ADS1115 (http://adafru.it/1085) top out at 16 bits. The ADS122C04 is a bit more expensive than either, but it's one of the few ADCs we've seen that gives you up to 2000 samples-per-second (on one channel) at 24-bit resolution.



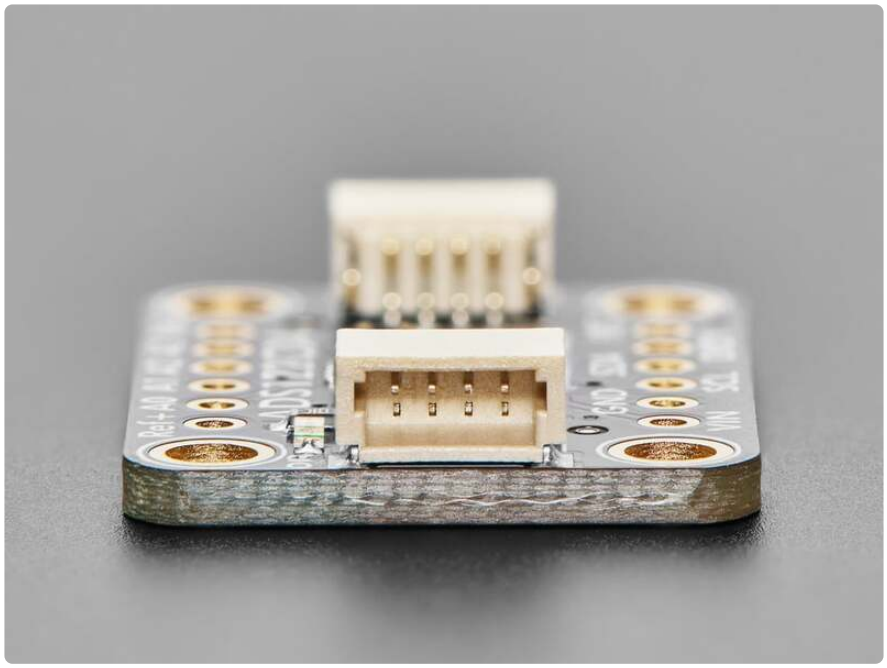
As you can expect from TI, it's a great quality ADC with lots of extras like a reset pin, internal temperature sensor, either single-ended or differential on any of the four inputs, internal 2.048V reference voltage, external positive and negative reference voltage input pins, 3V or 5V power and logic levels, and up to 128x gain. It even has two I2C address pins so you can put four of these on the same I2C bus.



This board would make an excellent data acquisition interface for lots of different scientific or engineering measurements. The I2C interface means it's easy to get it working with any microcontroller or microcomputer, from the 8-bit Arduino-compatible

to the highest end Raspberry Pi. Note that to get 2kSPS you'll need to run the I2C bus at 1 MHz and process that data from I2C quickly!

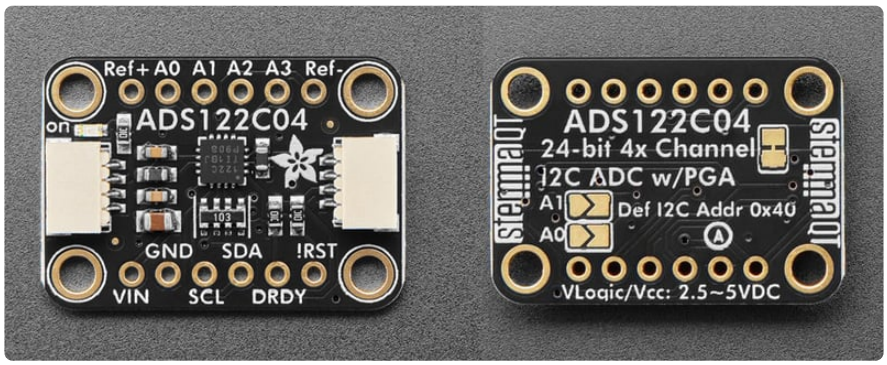
It comes with a bit of 0.1" standard header in case you want to use it with a breadboard or perfboard. It has four mounting holes for easy attachment.



To get you going fast, we spun up a custom-made PCB in the **STEMMA QT** form factor (https://adafru.it/LBQ), making it easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the ADS122C04 or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB). (**QT Cable is not included**, but we have a variety in the shop (https://adafru.it/17VE).)

We have example code for both the Raspberry Pi, Arduino, and CircuitPython (https://adafru.it/1aC4). Simply connect GND to ground, VDD to your 3V~5V logic power supply, and SCL/SDA to your microcontroller's I2C port and run the example code to start reading data.

Pinouts



The default I2C address is **0x40**.

Power Pins

- **VIN** - this is the power pin. Since the ADC chip uses 2.5-5 VDC to power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V.
- **Ref+** - this is the positive reference input pin. By default it is not connected and the ADC will use the internal 2.048V reference voltage.
- **GND** - common ground for power and logic.
- **Ref-** - this is the negative reference input pin.

I2C Logic Pins

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin 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 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 connectors to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6)

e that to get 2kSPS you'll need to run the I2C bus at 1 MHz and process data from I2C quickly!

ADC Inputs

- **A0-A3** - The 4 inputs to the ADC. These inputs can get up to 2k samples per second (you'll need 1MHz I2C speed support for max speed rate), and 24-bits of resolution.

Address Jumpers

On the back of the board are **two address jumpers**, labeled **A0** and **A1**. These jumpers allow you to chain up to 4 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads:

ADDR	A0	A1
0x40	L	L
0x41	H	L
0x44	L	H
0x45	H	H

Other Pins

- **DRDY -** This is the data ready pin. It is an output pin that is active low.
- **!RST -** This is the reset pin. To reset the board, pull this pin low.

Power LED and Jumper

- **Power LED -** In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled **on**. It is the green LED.
- **LED jumper** - In the upper right corner on the back of the board is a jumper for the power LED. If you wish to disable the power LED, cut the trace on this jumper.

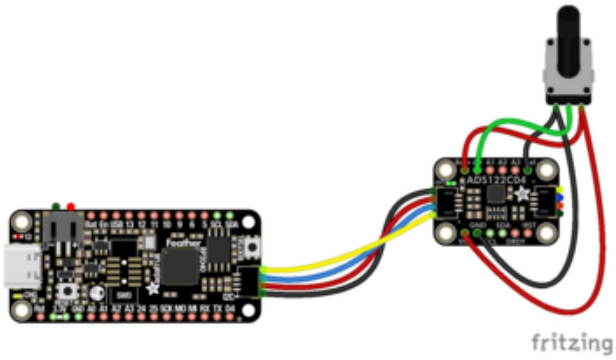
CircuitPython and Python

It's easy to use the **ADS122C04** with Python or CircuitPython, and the Adafruit_CircuitPython_ADS122C04 (https://adafru.it/1aC5) module. This module allows you to easily write Python code to read the 4 analog inputs on the ADC.

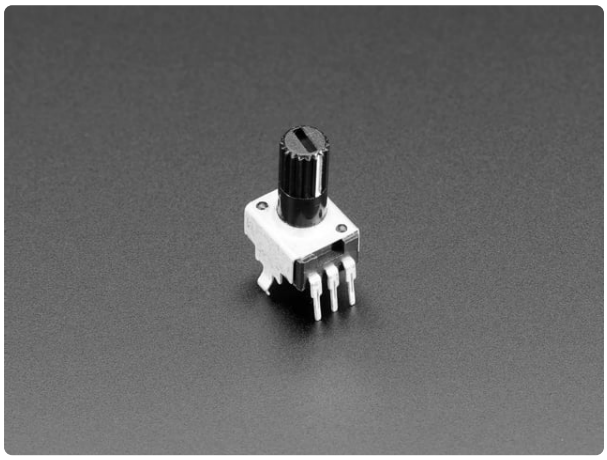
You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up the breakout to your board exactly as follows. The following is the breakout wired to a Feather RP2040 using the STEMMA connector along with a potentiometer connected to analog input 0 (**A0**) on the ADC:

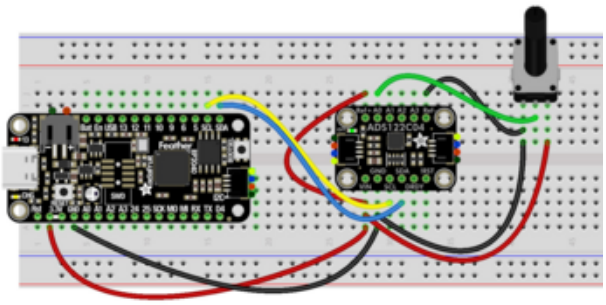


Board STEMMA 3V to **breakout VIN (red wire)**
Board STEMMA GND to **breakout GND (black wire)**
Board STEMMA SCL to **breakout SCL (yellow wire)**
Board STEMMA SDA to **breakout SDA (blue wire)**
Breakout VIN to **potentiometer positive (red wire)**
Breakout A0 to **potentiometer wiper (green wire)**
Breakout GND to **potentiometer negative (black wire)**
Breakout VIN to **breakout Ref+ (red wire)**
Breakout GND to **breakout Ref- (black wire)**



Potentiometer with Built In Knob - 10K ohm
Oh say can you seeBy the knob's early light...Sorry - we thought that was clever. And while it wasn't really, this potentiometer definitely...
https://www.adafruit.com/product/4133

The following is the breakout wired to a Feather RP2040 using a solderless breadboard:

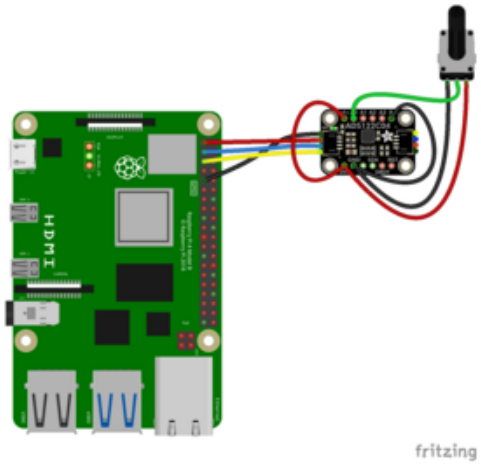


Board 3V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout VIN to **potentiometer positive (red wire)**
Breakout A0 to **potentiometer wiper (green wire)**
Breakout GND to **potentiometer negative (black wire)**
Breakout VIN to **breakout Ref+ (red wire)**
Breakout GND to **breakout Ref- (black wire)**

Python Computer Wiring

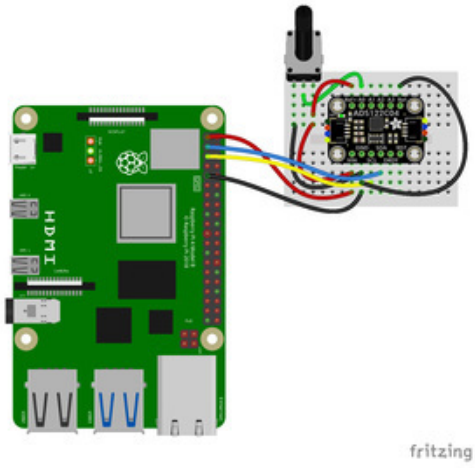
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C using the STEMMA connector:



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)**
Breakout VIN to **potentiometer positive (red wire)**
Breakout A0 to **potentiometer wiper (green wire)**
Breakout GND to **potentiometer negative (black wire)**
Breakout VIN to **breakout Ref+ (red wire)**
Breakout GND to **breakout Ref- (black wire)**

Here's the Raspberry Pi wired with I2C 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)**
Breakout VIN to **potentiometer positive (red wire)**
Breakout A0 to **potentiometer wiper (green wire)**
Breakout GND to **potentiometer negative (black wire)**
Breakout VIN to **breakout Ref+ (red wire)**
Breakout GND to **breakout Ref- (black wire)**

Python Installation of ADS122C04 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-ads122c04`

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

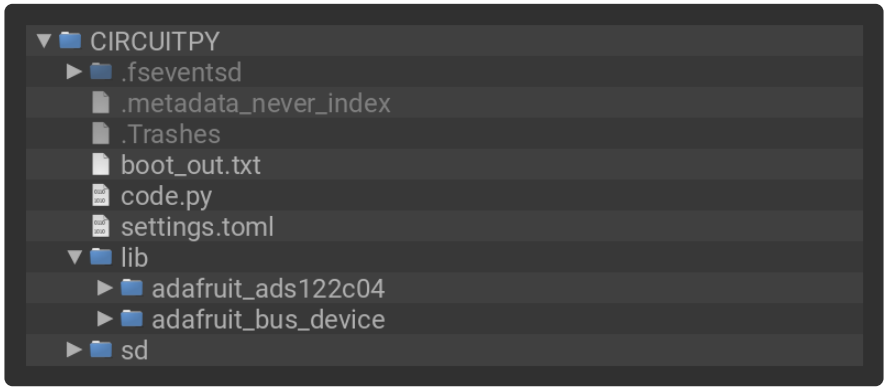
To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_ADS122C04** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

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:

- **adafruit_bus_device/**
- **adafruit_ads122c04/**



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

from adafruit_ads122c04.ads122c04 import ADS122C04, VREF_EXTERNAL
from adafruit_ads122c04.analog_in import AnalogIn

i2c = board.I2C()
adc = ADS122C04(i2c)
```

```
chan0 = AnalogIn(adc, 0)
chan1 = AnalogIn(adc, 1)
chan2 = AnalogIn(adc, 2)
chan3 = AnalogIn(adc, 3)

channels = (chan0, chan1, chan2, chan3)

adc.vref_input = VREF_EXTERNAL
adc.reference_voltage = 3.3

while True:
    for i, chan in enumerate(channels):
        print(f"A{i}: {chan.value:5d}  ({chan.voltage:.4f} V)")
    print("-" * 30)
    time.sleep(1)
```

The ADS122C04 is initialized over I2C. Then in the loop, the analog input on A0 is read and printed to the serial monitor. As you adjust the potentiometer connected to A0, you'll see the values change.



ıake code from different ADC's all compatible, we always return a 16-bit e (0 to 65535) even though, in this case, the underlying hardware is 24-

```
A0: 65496  (3.2983 V)
A1: 45350  (0.2746 V)
A2:    16  (0.0008 V)
A3:    16  (0.0008 V)
------------------------------
A0: 65526  (3.2996 V)
A1: 44705  (0.2344 V)
A2:    16  (0.0008 V)
A3:    16  (0.0008 V)
------------------------------
A0:  6414  (0.3119 V)
A1:    33  (0.0008 V)
A2:    16  (0.0008 V)
A3:    16  (0.0008 V)
------------------------------
A0: 38953  (2.0626 V)
A1: 18766  (0.0008 V)
A2:    16  (0.0008 V)
A3:    16  (0.0008 V)
------------------------------
A0: 49253  (2.4801 V)
A1: 27455  (0.0008 V)
A2:    16  (0.0008 V)
A3:    16  (0.0008 V)
```



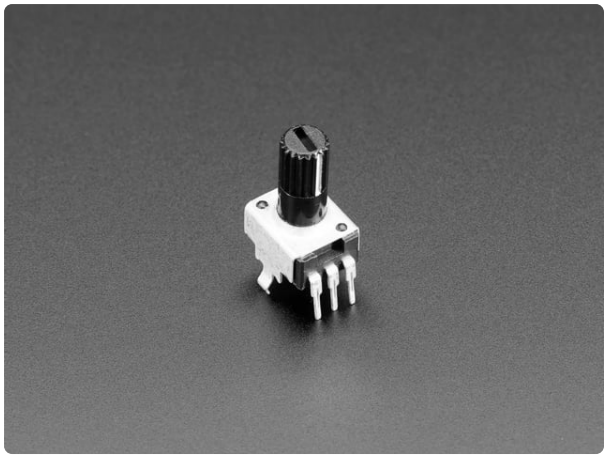
remaining 3 channels (A1, A2 and A3) are 'floating' - not connected to hing, so you will see varying random values.

Python Docs

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

Arduino

Using the ADS122C04 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller with a potentiometer, installing the Adafruit_ADS122C04 (https://adafru.it/1aC6) library, and running the provided example code.

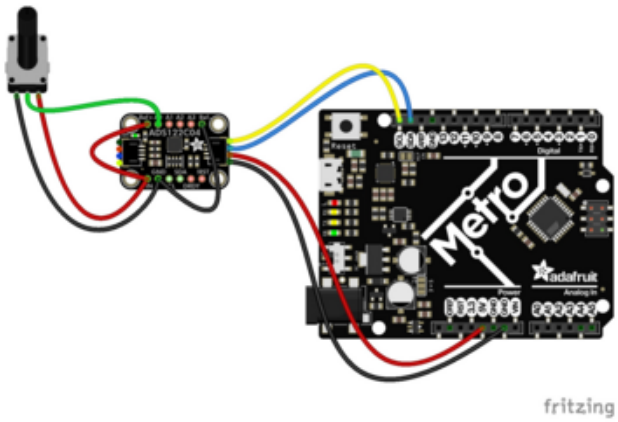


[Potentiometer with Built In Knob - 10K ohm]
Oh say can you seeBy the knob's early light...Sorry - we thought that was clever. And while it wasn't really, this potentiometer definitely...
https://www.adafruit.com/product/4133

Wiring

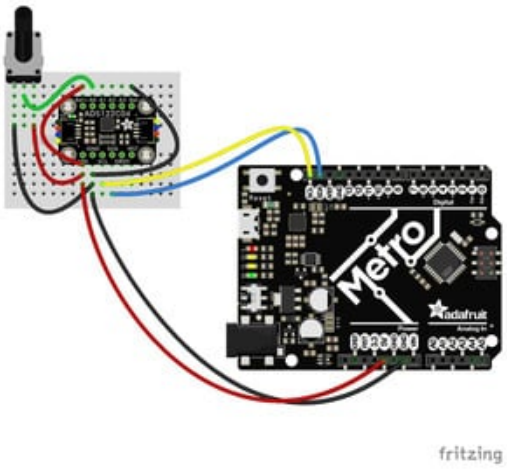
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

Here is an Adafruit Metro wired up to the breakout using the STEMMA QT connector. A potentiometer is connected to input 0 (**A0**) on the ADC:



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)**
Breakout VIN to **potentiometer positive (red wire)**
Breakout A0 to **potentiometer wiper (green wire)**
Breakout GND to **potentiometer negative (black wire)**
Breakout VIN to **breakout Ref+ (red wire)**
Breakout GND to **breakout Ref- (black wire)**

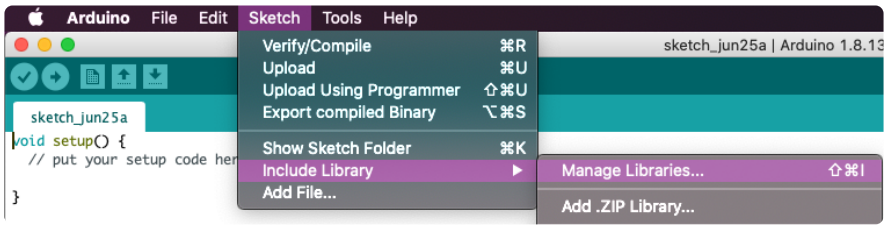
Here is an Adafruit Metro wired up using a solderless breadboard:



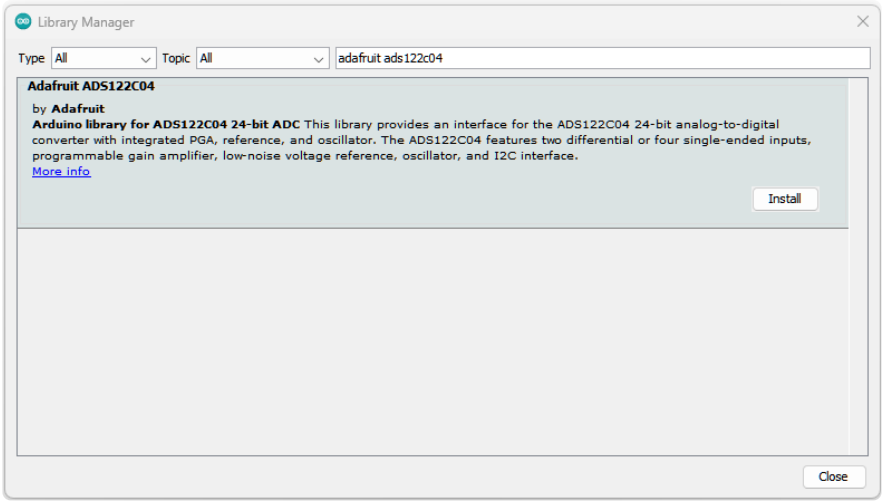
Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout VIN to **potentiometer positive (red wire)**
Breakout A0 to **potentiometer wiper (green wire)**
Breakout GND to **potentiometer negative (black wire)**
Breakout VIN to **breakout Ref+ (red wire)**
Breakout GND to **breakout Ref- (black wire)**

Library Installation

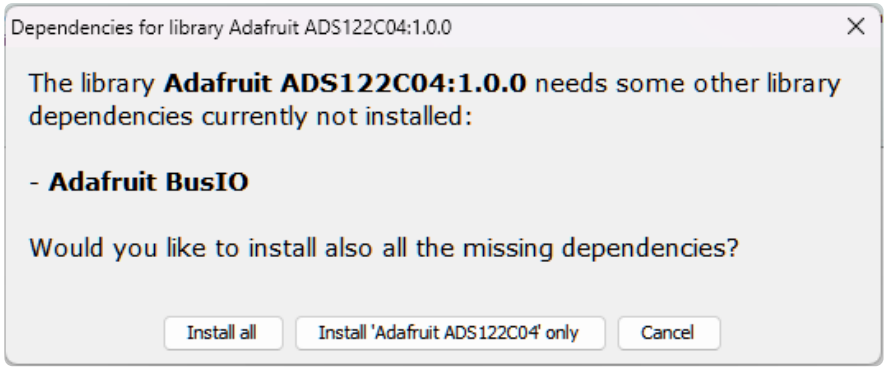
You can install the **Adafruit_ADS122C04** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_ADS122C04**, and select the **Adafruit ADS122C04** library:



If asked about dependencies, click "Install all".



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

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

Example Code

```
// SPDX-FileCopyrightText: 2026 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
 * Simple ADS122C04 example - reads all 4 single-ended analog inputs
```

```
 * and prints voltages to the Serial Monitor.
 *
 * If any inputs are floating, will read noise
 */

#include <Adafruit_ADS122C04.h>

Adafruit_ADS122C04 adc;

#ifdef ARDUINO_AVR_METRO
float voltage_ref = 5.0; // 5V from metro
#else
float voltage_ref = 3.3; // else assume 3.3V logic
#endif

// The four single-ended MUX settings (AINx vs AVSS)
const ads122c04_mux_t channels[] = {
  ADS122C04_MUX_AIN0,
  ADS122C04_MUX_AIN1,
  ADS122C04_MUX_AIN2,
  ADS122C04_MUX_AIN3
};

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);

  Serial.println("ADS122C04 4-Channel Single-Ended Test");

  if (!adc.begin()) {
    Serial.println("Failed to find ADS122C04 - check wiring!");
    while (1) delay(10);
  }
  Serial.println("ADS122C04 found!");

  // Use analog supply (AVDD-AVSS) as reference for full 0-5V range
  adc.setVoltageReference(ADS122C04_VREF_SUPPLY);
  Serial.print("Setting reference voltage to: ");
  Serial.print(voltage_ref);
  Serial.println("V");
  adc.setReferenceVoltage(voltage_ref);

  // Gain 1, PGA bypassed
  adc.setGain(ADS122C04_GAIN_1);
  adc.enablePGA(false);

  // Single-shot mode at 20 SPS
  adc.setContinuousMode(false);
  adc.setDataRate(ADS122C04_RATE_20SPS);

  Serial.println("Config: AVDD ref, gain=1, single-shot 20 SPS");
  Serial.println();
}

void loop() {
  for (uint8_t ch = 0; ch < 4; ch++) {
    // Select the channel
    adc.setMux(channels[ch]);

    // Trigger a single conversion
    adc.startSync();

    // Wait for data ready (at 20 SPS, conversion takes ~50ms)
    while (!adc.isDataReady()) {
      delay(1);
    }

    // Read raw ADC value, then convert to voltage
    int32_t raw = adc.readData();
```

```
    float voltage = adc.convertToVoltage(raw);

    Serial.print("A");
    Serial.print(ch);
    Serial.print(": ");

    if (isnan(voltage)) {
      Serial.print("READ ERROR");
    } else {
      Serial.print("raw=");
      Serial.print(raw);
      Serial.print(" -> ");
      Serial.print(voltage, 6);
      Serial.print(" V");
    }

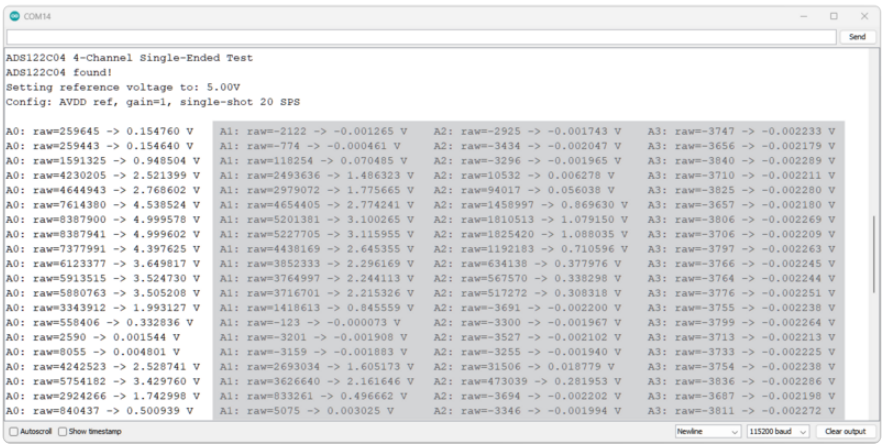
    Serial.print("\t");
  }

  Serial.println();
  delay(500);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the ADS122C04 recognized over I2C. The reference voltage is set to 5V for a Metro and 3.3V for other boards. As you turn the potentiometer attached to A0, you'll see the raw value and voltage reading change.



remaining 3 channels (A1, A2 and A3) are 'floating' - not connected to hing, so you will see varying random values.



Arduino Docs

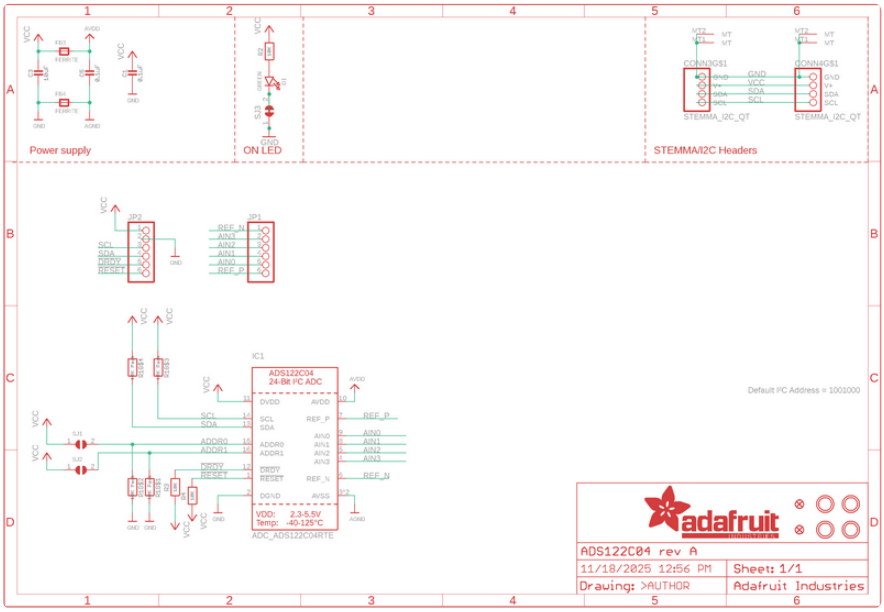
Arduino Docs (https://adafru.it/1aBL)

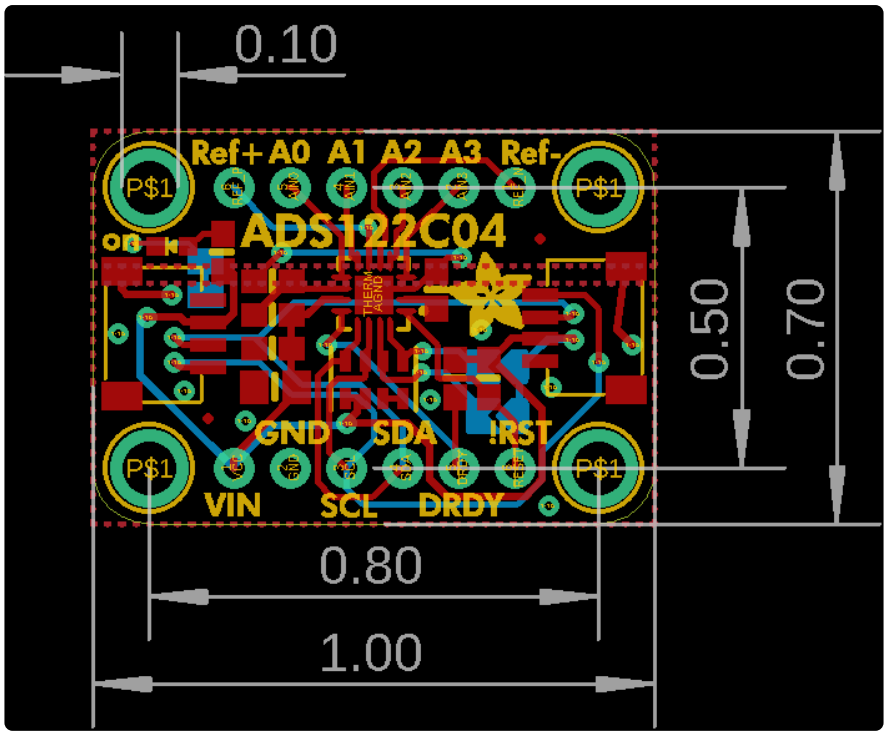
Downloads

Files

- ADS122C04 Datasheet (https://adafru.it/1aC7)
- EagleCAD PCB files on GitHub (https://adafru.it/1aC8)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1aC9)

Schematic and Fab Print




0.10
Ref+ A0 A1 A2 A3 Ref-
P$1
P$1
ADS122C04
0.50
0.70
GND SDA !RST
P$1
P$1
VIN SCL DRDY
0.80
1.00