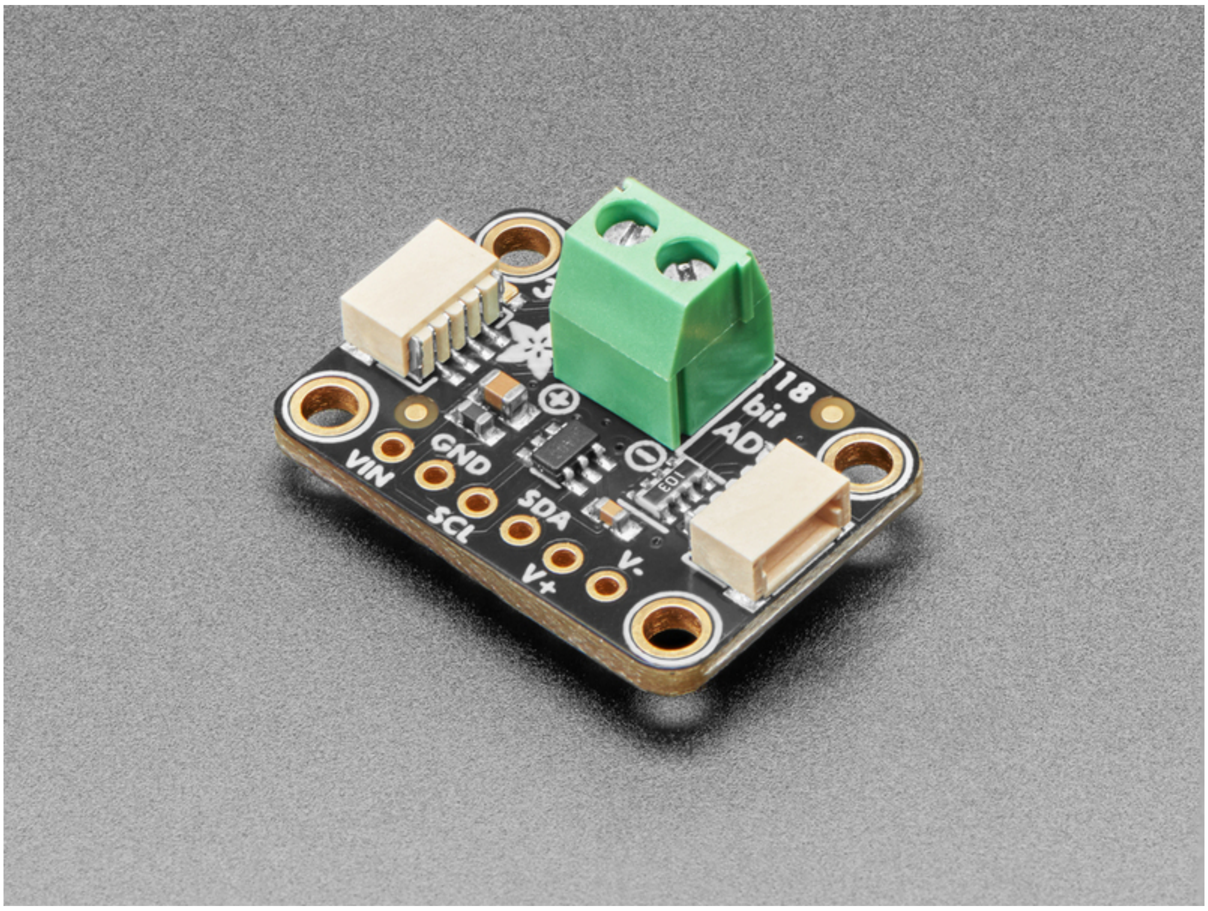
Adafruit MCP3421 18-Bit ADC

Created by Liz Clark

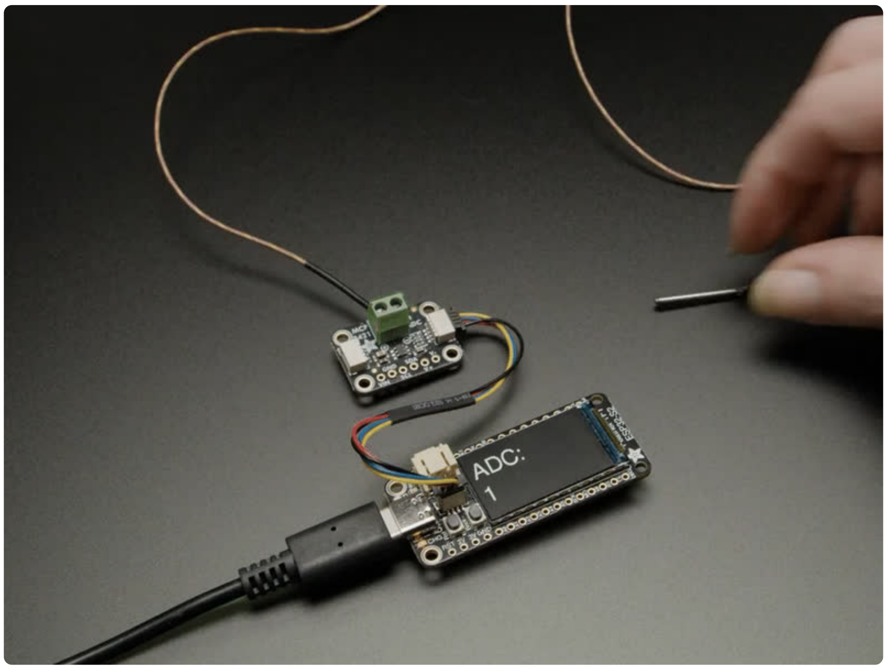


https://learn.adafruit.com/adafruit-mcp3421-18-bit-adc

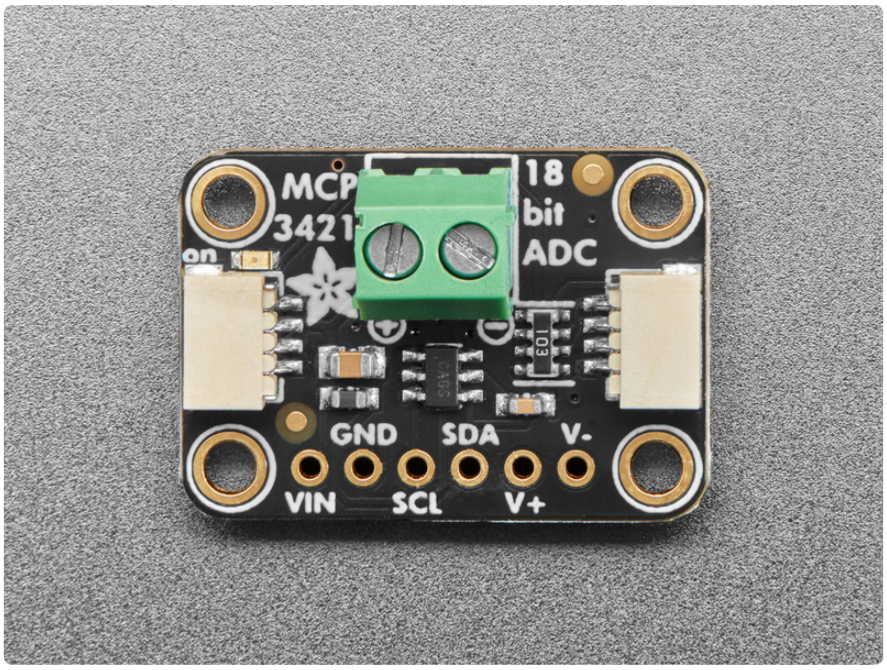
Last updated on 2024-02-16 11:39:20 AM EST

Table of Contents

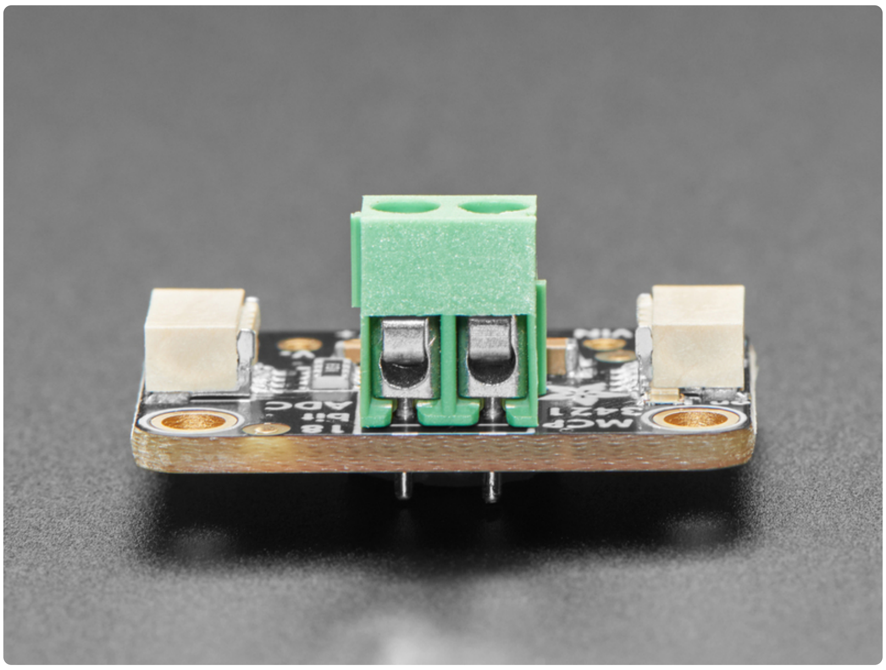
Overview



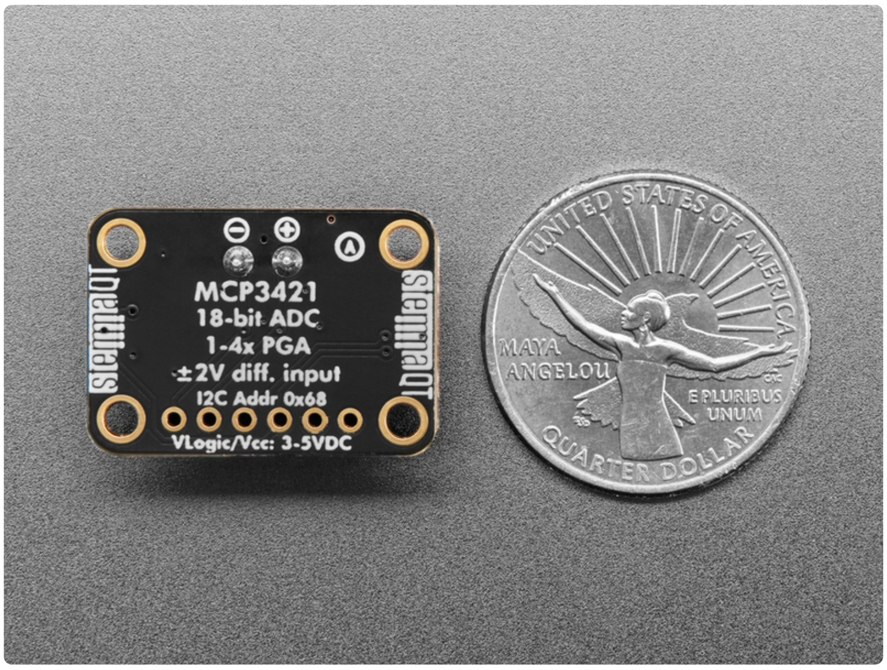
The **Adafruit MCP3421 18-Bit ADC** is a simple, inexpensive, and easy to use 18-bit, 240 SPS, single-channel ADC with an I2C interface that can run up to 3.4MHz clock rate. A perfect component whenever you need an ADC that has differential inputs, adjustable gain, and a built-in precision/low-drift reference voltage.



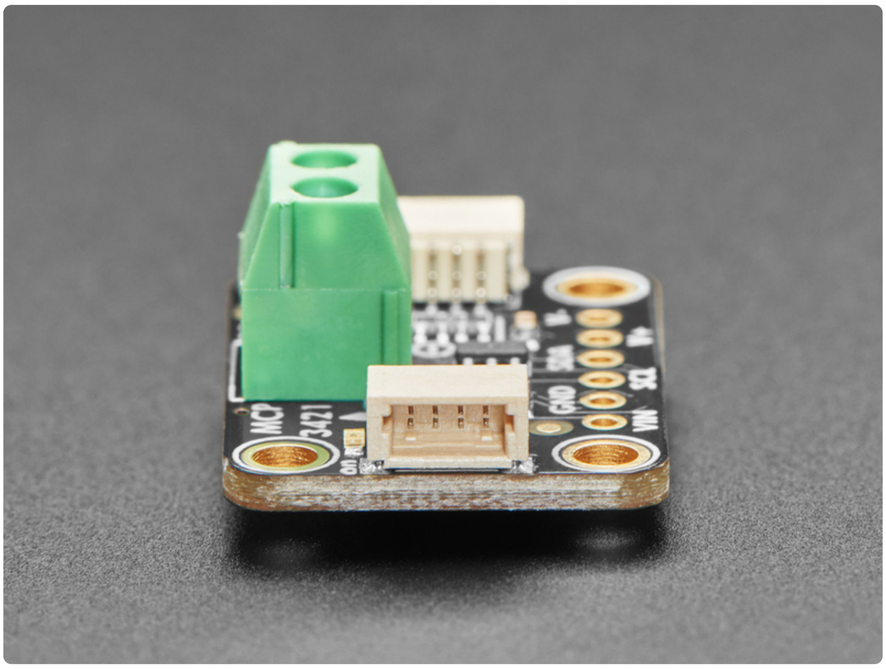
One of the trade-offs with getting 18-bit precision is that the ADC is not going to be very fast: you can configure the chip to do a faster 12-bit conversion at 240 SPS, but at 18-bits it slows down to 3.5 SPS. That's because the way a sigma-delta ADC (https://adafru.it/19bP) works, it 'guesses' the analog voltage and uses a comparator to determine whether the input is higher or lower. Each 'guess' takes an extra step, and thus halves the throughput, so 12-bit is 240 SPS, and 14-bit is 1/4 (2-bit) slower, 60 SPS. Ditto 16-bit is 1/4 slower, 15 SPS, and finally 18-bit is 3.75 SPS.



The MCP3421 is already set up for differential inputs, which means that you can read positive or negative differences between the two inputs, **as long as both signals are between 0 and 2.048V**. This means its not going to be great for reading stuff like potentiometers, where you have a single-end reading referenced to ground, and you want to read the full range from 0 to Vcc. It is great, however, for reading sensors like strain gauges, pressure sensors, or thermocouples.



We have a ready to go Arduino library that makes usage simple (https://adafru.it/19bQ): select the sample rate / precision you want and one-shot or continuous mode. Then read values over I2C! This sensor has a fixed address, so you may want to use a multiplexor (https://adafru.it/ZlC) if you want to connect multiple MCP3421's on a single I2C bus.

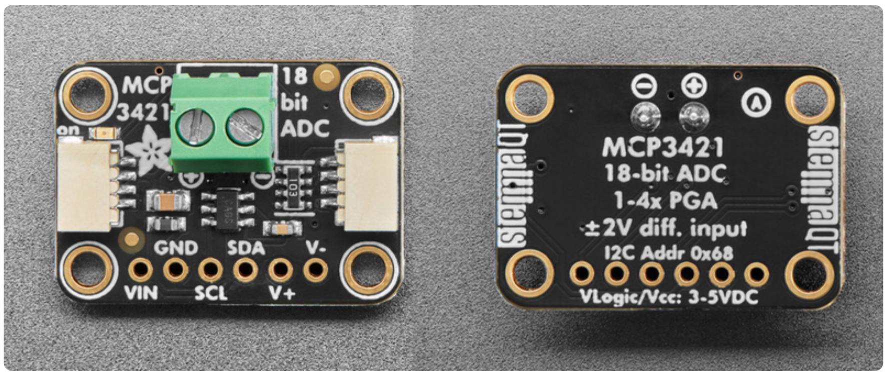


To get you going fast, we spun up a custom-made PCB in the **STEMMA QT** form factor (https://adafru.it/LBQ), making it easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the MCP 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).

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

Pinouts



The default I2C address is **0x68**.

Power Pins

- **VIN** - this is the power pin. Since the ADC chip uses 3-5 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.
- **GND** - common ground for power and logic.

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)

ADC Inputs

The MCP3421 has two inputs: **V+** and **V-**. **V+** is the positive input and **V-** is the negative input. The inputs are available via the terminal block at the top of the board or the two pins labeled **V+** and **V-** at the bottom of the board.

The MCP3421 is set up for differential inputs, which means that you can read positive or negative differences between the two inputs, **as long as both signals are between 0 and 2.048V**.

Power LED

- **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.

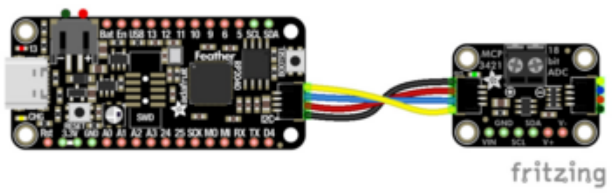
CircuitPython and Python

It's easy to use the **MCP3421** with Python or CircuitPython, and the Adafruit_CircuitPython_MCP3421 (https://adafru.it/19bR) module. This module allows you to easily write Python code to read input 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).

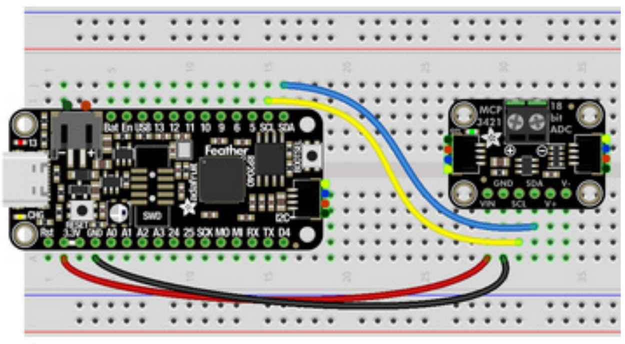
CircuitPython Microcontroller Wiring

First wire up the breakout to your board exactly as follows. The following is the breakout wired to a Feather RP2040 (shown using a STEMMA QT cable):



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)**

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

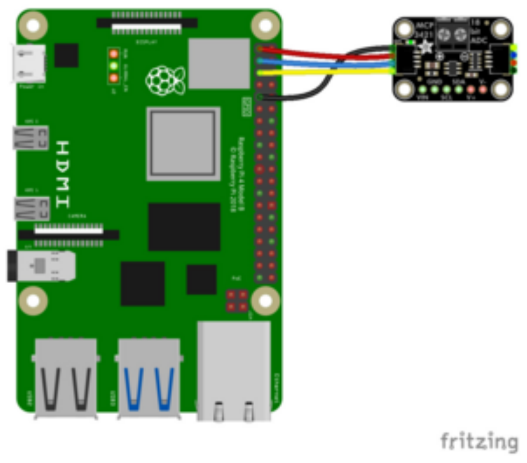


Board 3V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**

Python Computer Wiring

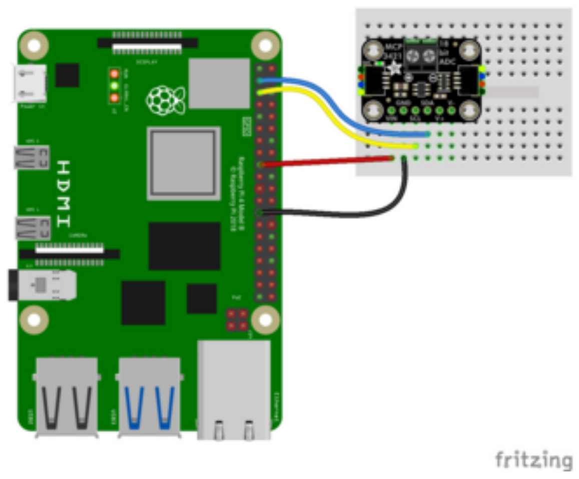
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C using a STEMMA QT 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)**

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)**

Python Installation of MCP3421 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-mcp3421`

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

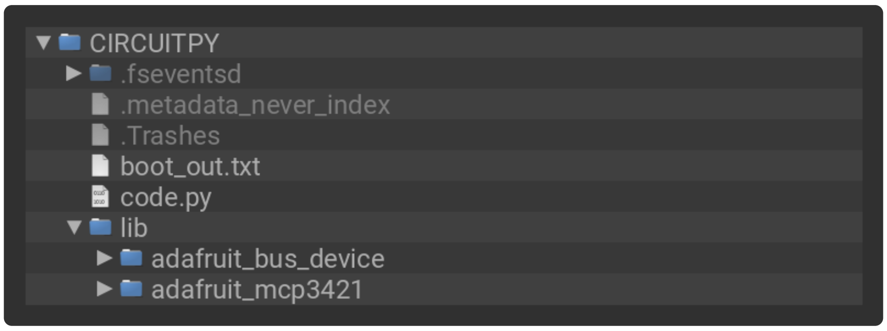
To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_MCP3421** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

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_mcp3421/**



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) (https://adafru.it/Bec) to see the data printed out!

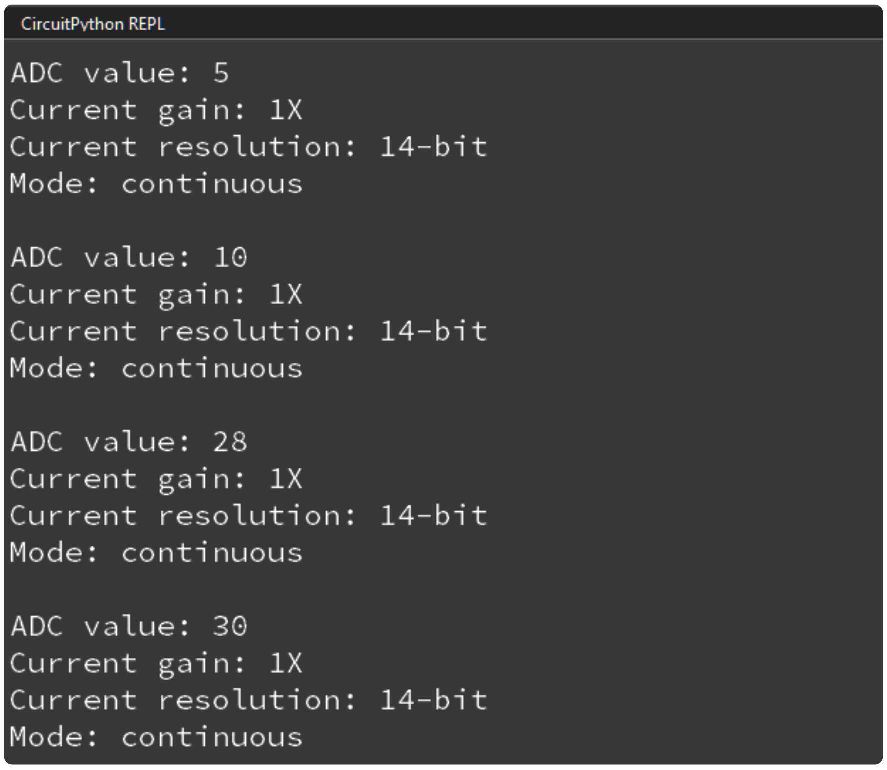
If running Python: The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: Copyright (c) 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_mcp3421.mcp3421 as ADC
from adafruit_mcp3421.analog_in import AnalogIn

i2c = board.I2C()

adc = ADC.MCP3421(i2c, gain=1, resolution=14, continuous_mode=True)
adc_channel = AnalogIn(adc)
# gain, resolution and mode can also be set after instantiation:

# set gain to 1, 2, 4 or 8x
# defaults to 1
# adc.gain = 1

# set resolution to 12, 14, 16 or 18
# defaults to 14
# adc.resolution = 14

# set continuous read mode True or False for one-shot
# defaults to True
# adc.continuous_mode = True
```

```
while True:
    print(f"ADC value: {adc_channel.value}")
    print(f"Current gain: {adc.gain}X")
    print(f"Current resolution: {adc.resolution}-bit")
    if adc.continuous_mode:
        mode = "continuous"
    else:
        mode = "one-shot"
    print(f"Mode: {mode}")
    print()
    time.sleep(0.01)
```

The MCP3421 is initialized over I2C. Then in the loop, the analog input is read and printed to the serial monitor, along with the current gain, resolution and mode settings.

This ADC is great for reading sensors like strain gauges, pressure sensors, or thermocouples. It's not going to be great for reading stuff like potentiometers, where you have a single-end reading referenced to ground, and you want to read the full range from 0 to Vcc.



```
ADC value: 5
Current gain: 1X
Current resolution: 14-bit
Mode: continuous

ADC value: 10
Current gain: 1X
Current resolution: 14-bit
Mode: continuous

ADC value: 28
Current gain: 1X
Current resolution: 14-bit
Mode: continuous

ADC value: 30
Current gain: 1X
Current resolution: 14-bit
Mode: continuous
```

Python Docs

Python Docs (https://adafru.it/19bG)

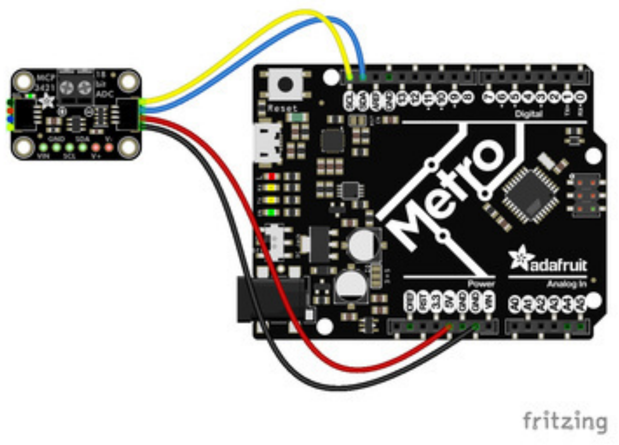
Arduino

Using the MCP3421 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_MCP3421 (https://adafru.it/19bQ) library, and running the provided example code.

Wiring

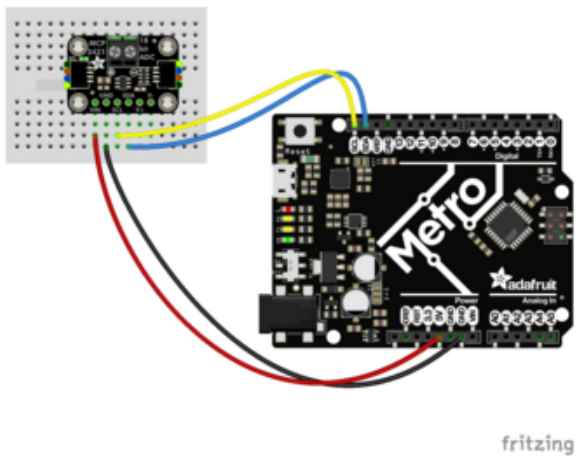
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

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



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)**

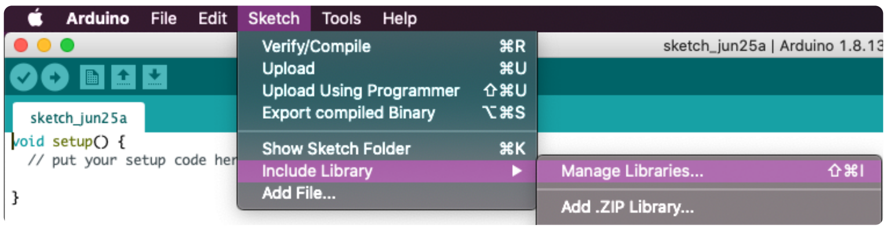
Here is an Adafruit Metro wired up using a solderless breadboard:



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

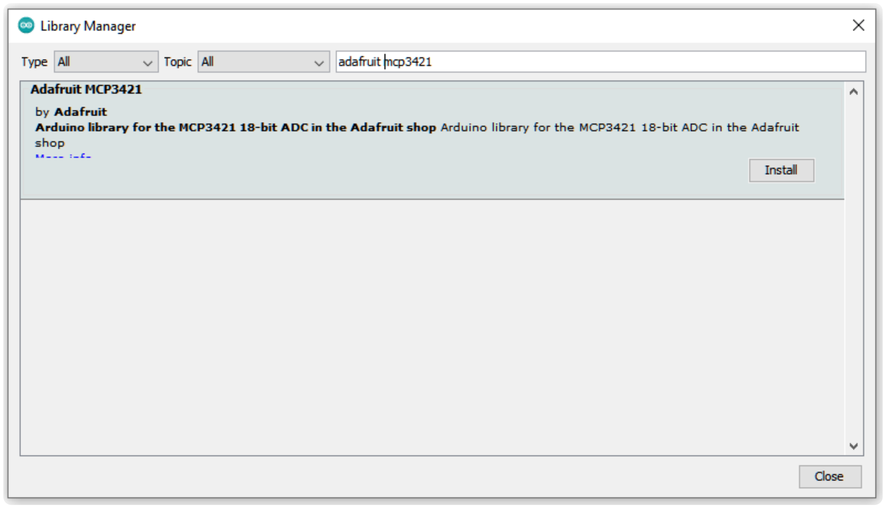
Library Installation

You can install the **Adafruit_MCP3421** library for Arduino using the Library Manager in the Arduino IDE.



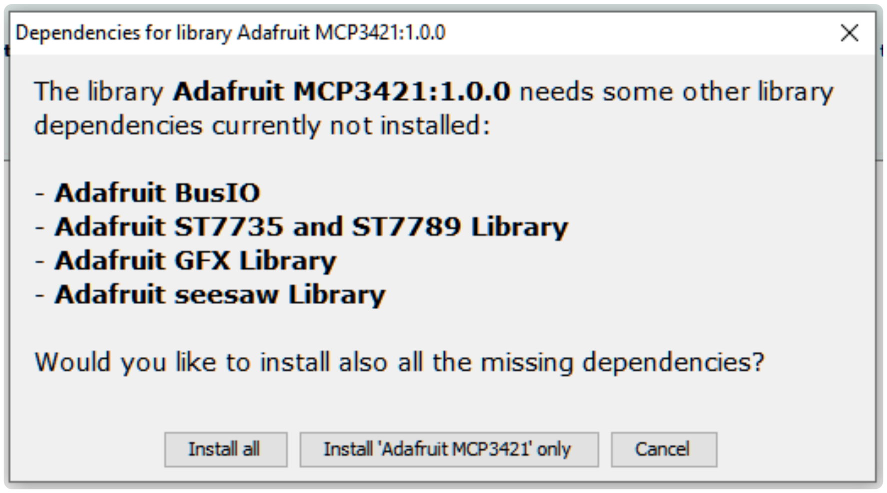
```
void setup() {
  // put your setup code her
}
```

Click the **Manage Libraries ...** menu item, search for **Adafruit_MCP3421**, and select the **Adafruit MCP3421** library:



Adafruit MCP3421
by **Adafruit**
Arduino library for the MCP3421 18-bit ADC in the Adafruit shop Arduino library for the MCP3421 18-bit ADC in the Adafruit shop

If asked about dependencies, click "Install all".


Dependencies for library Adafruit MCP3421:1.0.0

The library **Adafruit MCP3421:1.0.0** needs some other library dependencies currently not installed:

- **Adafruit BusIO**
- **Adafruit ST7735 and ST7789 Library**
- **Adafruit GFX Library**
- **Adafruit seesaw Library**

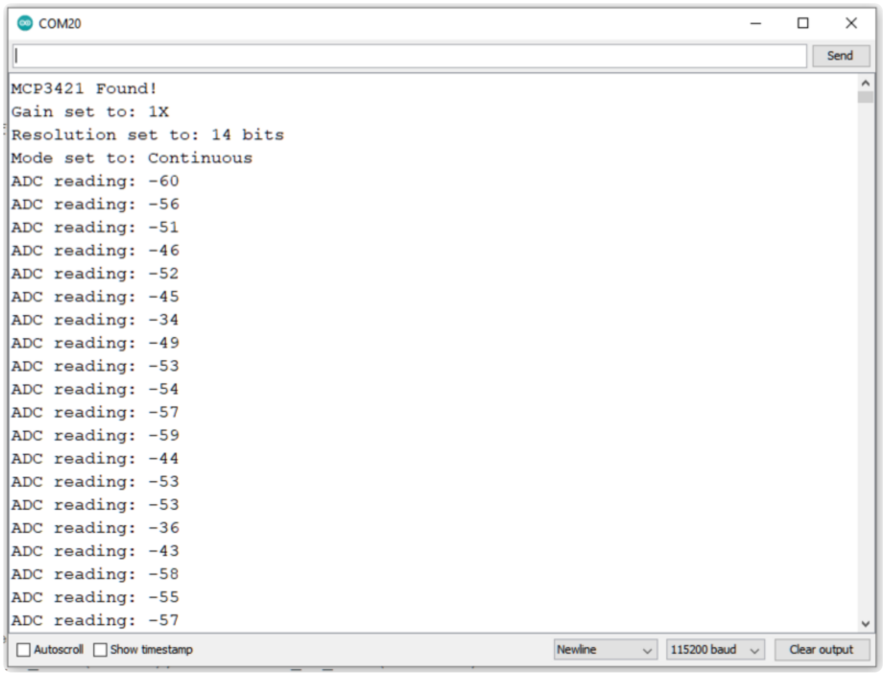
Would you like to install also all the missing dependencies?

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

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

Example Code

```
#include "Adafruit_MCP3421.h"

Adafruit_MCP3421 mcp;

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10); // Wait for serial port to connect. Needed for native USB port only
  }

  // Begin can take an optional address and Wire interface
  if (!mcp.begin(0x68, &Wire)) {
    Serial.println("Failed to find MCP3421 chip");
    while (1) {
      delay(10); // Avoid a busy-wait loop
    }
  }
  Serial.println("MCP3421 Found!");

  // Options: GAIN_1X, GAIN_2X, GAIN_4X, GAIN_8X
  mcp.setGain(GAIN_1X);
  Serial.print("Gain set to: ");
  switch (mcp.getGain()) {
    case GAIN_1X: Serial.println("1X"); break;
    case GAIN_2X: Serial.println("2X"); break;
    case GAIN_4X: Serial.println("4X"); break;
    case GAIN_8X: Serial.println("8X"); break;
  }

  // The resolution affects the sample rate (samples per second, SPS)
  // Other options: RESOLUTION_14_BIT (60 SPS), RESOLUTION_16_BIT (15 SPS),
RESOLUTION_18_BIT (3.75 SPS)
  mcp.setResolution(RESOLUTION_14_BIT); // 240 SPS (12-bit)
  Serial.print("Resolution set to: ");
  switch (mcp.getResolution()) {
    case RESOLUTION_12_BIT: Serial.println("12 bits"); break;
    case RESOLUTION_14_BIT: Serial.println("14 bits"); break;
    case RESOLUTION_16_BIT: Serial.println("16 bits"); break;
    case RESOLUTION_18_BIT: Serial.println("18 bits"); break;
  }

  // Test setting and getting Mode
  mcp.setMode(MODE_CONTINUOUS); // Options: MODE_CONTINUOUS, MODE_ONE_SHOT
  Serial.print("Mode set to: ");
  switch (mcp.getMode()) {
    case MODE_CONTINUOUS: Serial.println("Continuous"); break;
    case MODE_ONE_SHOT: Serial.println("One-shot"); break;
  }
}

uint32_t lastSecond = millis(); // Store the last time we printed SPS
uint32_t sampleCount = 0; // Count how many samples were taken

void loop() {
    // Check if MCP3421 has completed a conversion in continuous mode
    if (mcp.isReady()) {
        int32_t adcValue = mcp.readADC(); // Read ADC value
        Serial.print("ADC reading: ");
        Serial.println(adcValue);
        sampleCount++; // Increment the sample count
    }

    uint32_t currentMillis = millis();
    if (currentMillis - lastSecond >= 1000) { // Check if a second has passed
        Serial.print("SPS (Samples Per Second): ");
        Serial.println(sampleCount);

        // Reset the counter and update the time
        sampleCount = 0;
        lastSecond = currentMillis;
    }
}
```



```
MCP3421 Found!
Gain set to: 1X
Resolution set to: 14 bits
Mode set to: Continuous
ADC reading: -60
ADC reading: -56
ADC reading: -51
ADC reading: -46
ADC reading: -52
ADC reading: -45
ADC reading: -34
ADC reading: -49
ADC reading: -53
ADC reading: -54
ADC reading: -57
ADC reading: -59
ADC reading: -44
ADC reading: -53
ADC reading: -53
ADC reading: -36
ADC reading: -43
ADC reading: -58
ADC reading: -55
ADC reading: -57
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the MCP3421 recognized over I2C. The example sets the gain to 1X, resolution to 14-bit and the mode to continuous. The reading from the ADC will print to the Serial Monitor.

This ADC is great for reading sensors like strain gauges, pressure sensors, or thermocouples. It's not going to be great for reading stuff like potentiometers, where you have a single-end reading referenced to ground, and you want to read the full range from 0 to Vcc.

Arduino Docs

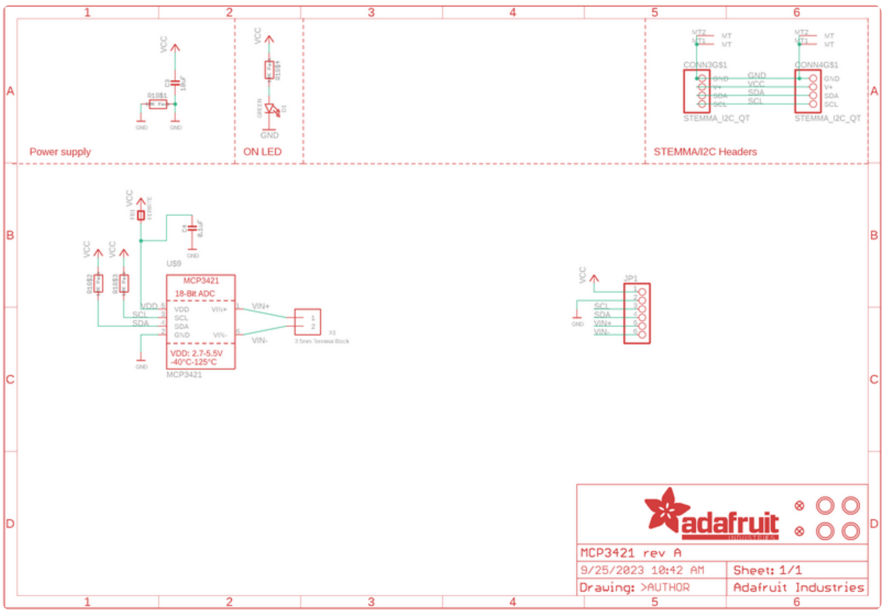
Arduino Docs (https://adafru.it/19bq)

Downloads

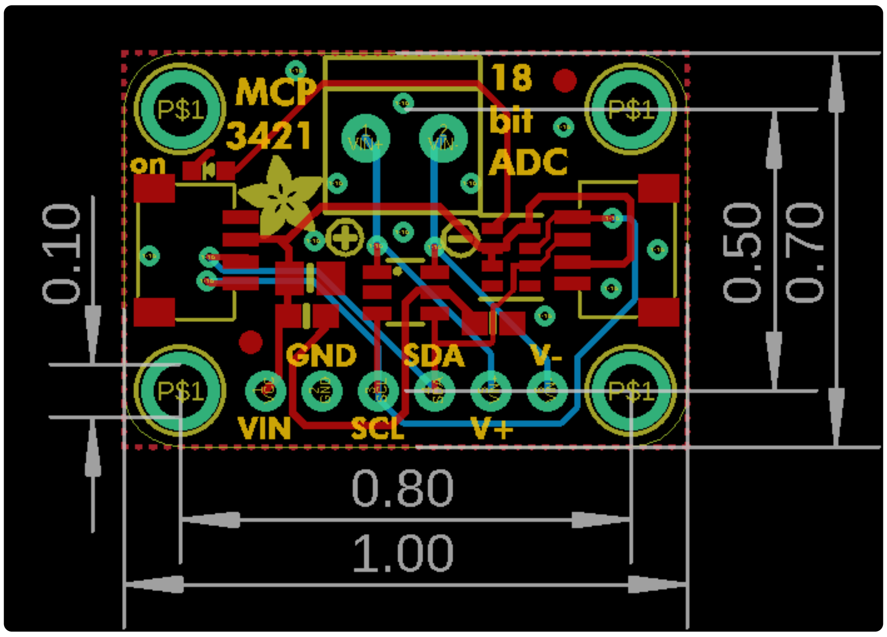
Files

- MCP3421 datasheet (https://adafru.it/19bS)
- EagleCAD PCB files on GitHub (https://adafru.it/19bT)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/19bU)

Schematic and Fab Print



Power supply
ON LED
STEMMA/I2C Headers
MCP3421
18-Bit ADC
VDD: 2.7-5.5V
-40°C-125°C
adafruit
MCP3421 rev A
9/25/2023 10:42 AM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries



MCP
3421
18
bit
ADC
on
GND
SDA
V-
VIN
SCL
V+
0.10
0.50
0.70
0.80
1.00