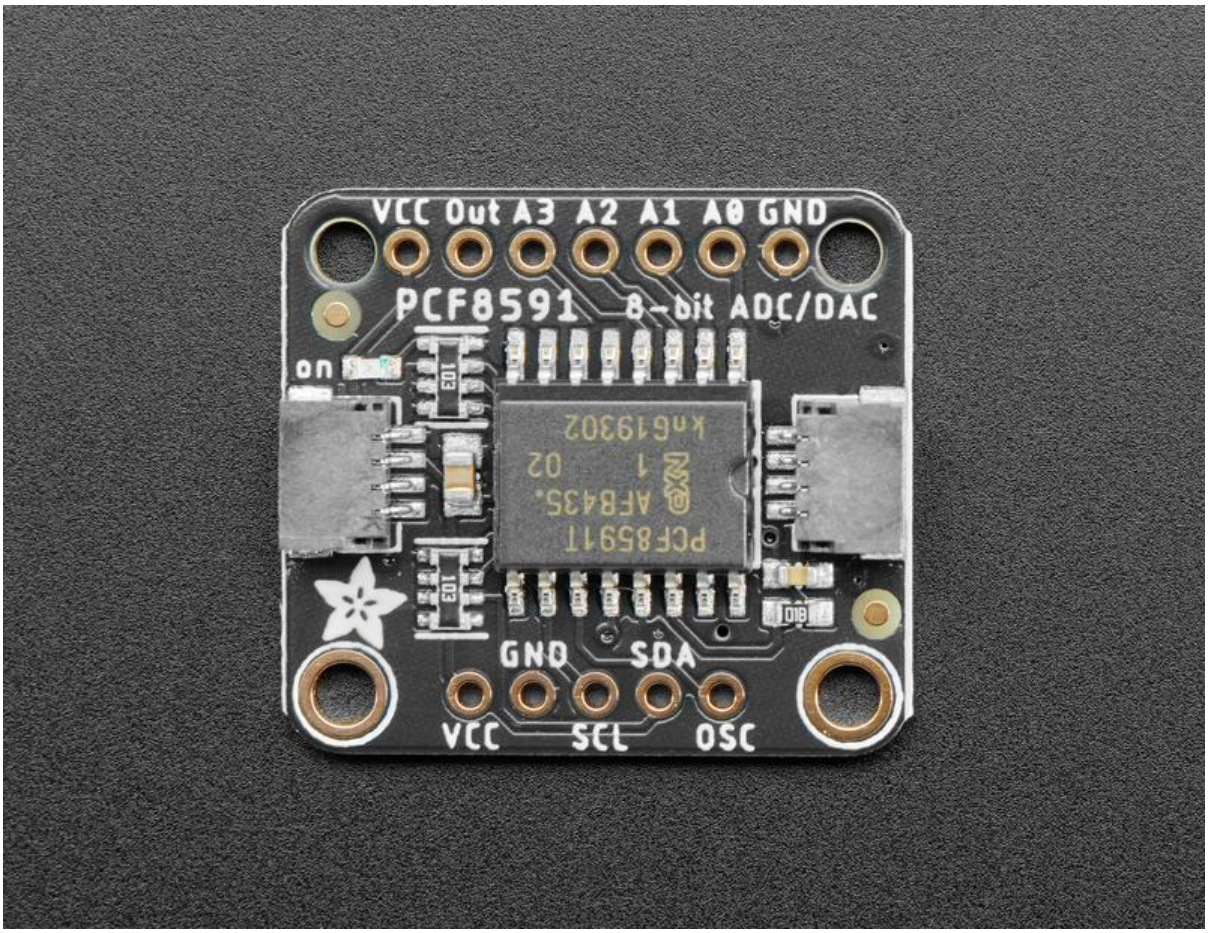
Adafruit PCF8591 Basic 4 x ADC + DAC Breakout

Created by Bryan Siepert

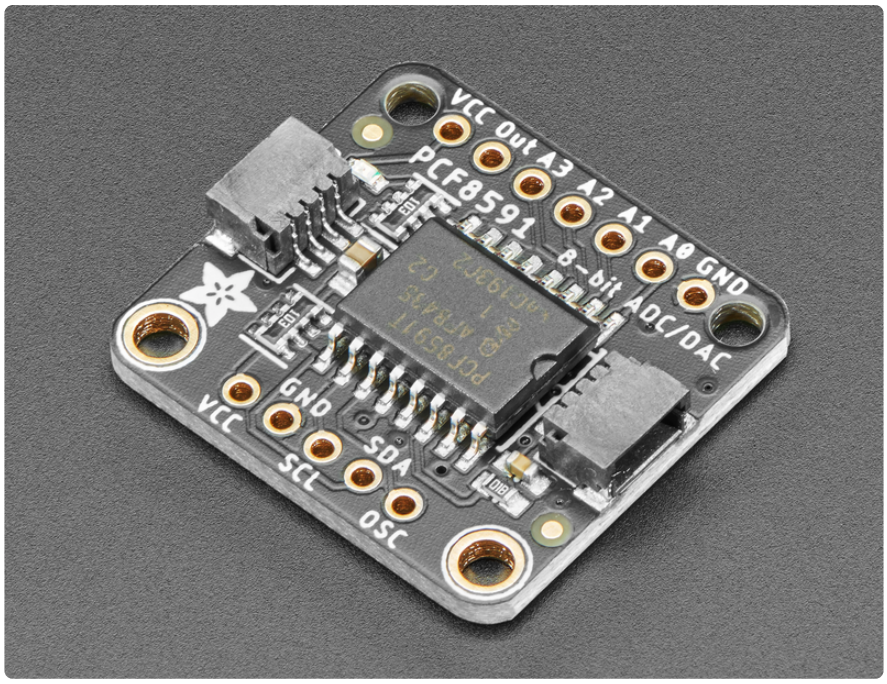


https://learn.adafruit.com/adafruit-pcf8591-adc-dac

Last updated on 2022-12-01 03:54:49 PM EST

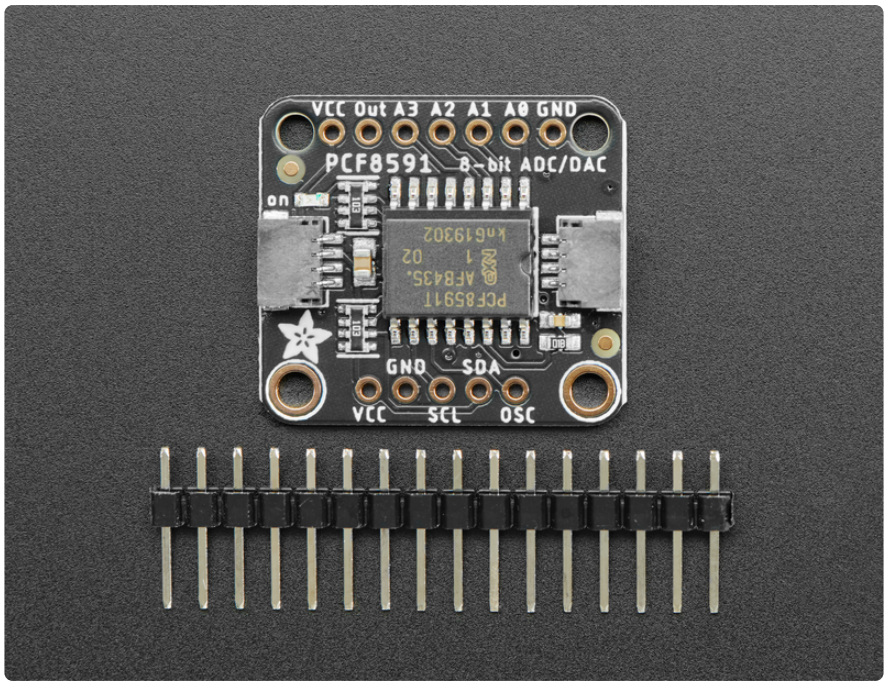
Table of Contents

Overview

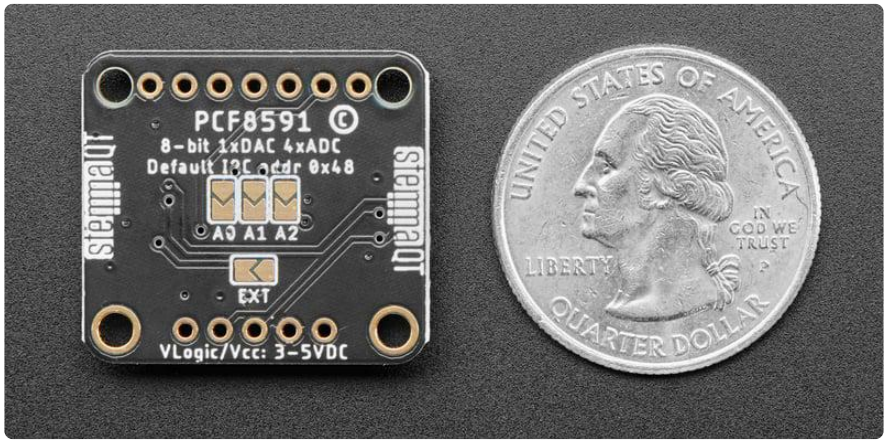


Measuring voltage and adjusting it is what electronics is all about so you won't get far without friends like the PCF8591 ADC+DAC combo (). Analog to Digital Converters help by measuring an analog voltage and turning it into something a microcontroller like a Metro or Arduino can understand. If you're using a single board computer like a Raspberry Pi, you might not have any other way to measure a voltage because even though they are well equipped for digital circuits, many boards of that type don't have any pins that can measure analog voltages.

Adding a PCF8591 to your electronics project will give you not one, not two, but four 8-bit analog inputs that you can use to measure voltages from. If knobs are just the thing to complete your project, just add a PCF8591 and some potentiometers and you're ready to twist, turn, twiddle and tweak to get things just right.

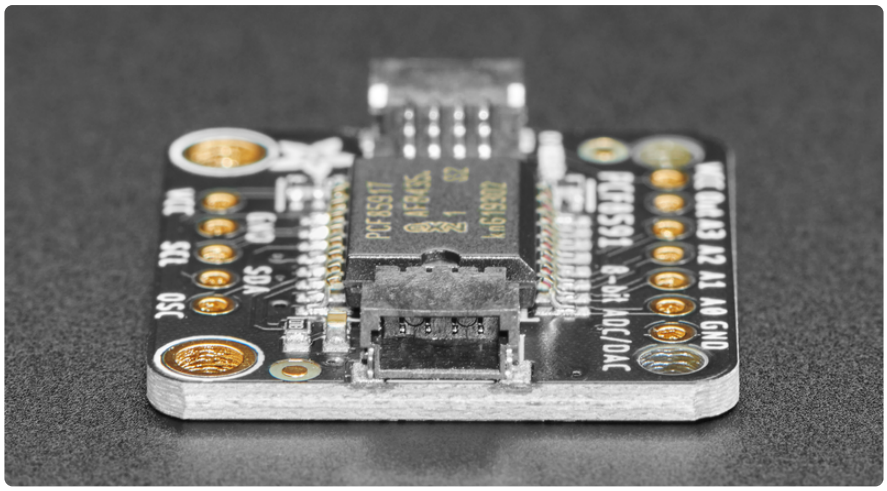


Along with four 8-bit ADC channels, the PCF8591 comes complete with an 8-bit Digita l to Analog Converter converter as well! Not only can you measure voltages, but now you can create them just as you want them. You can even use the DAC and ADC together to create an input to a circuit and measure the results with the ADC. The possibilities abound!



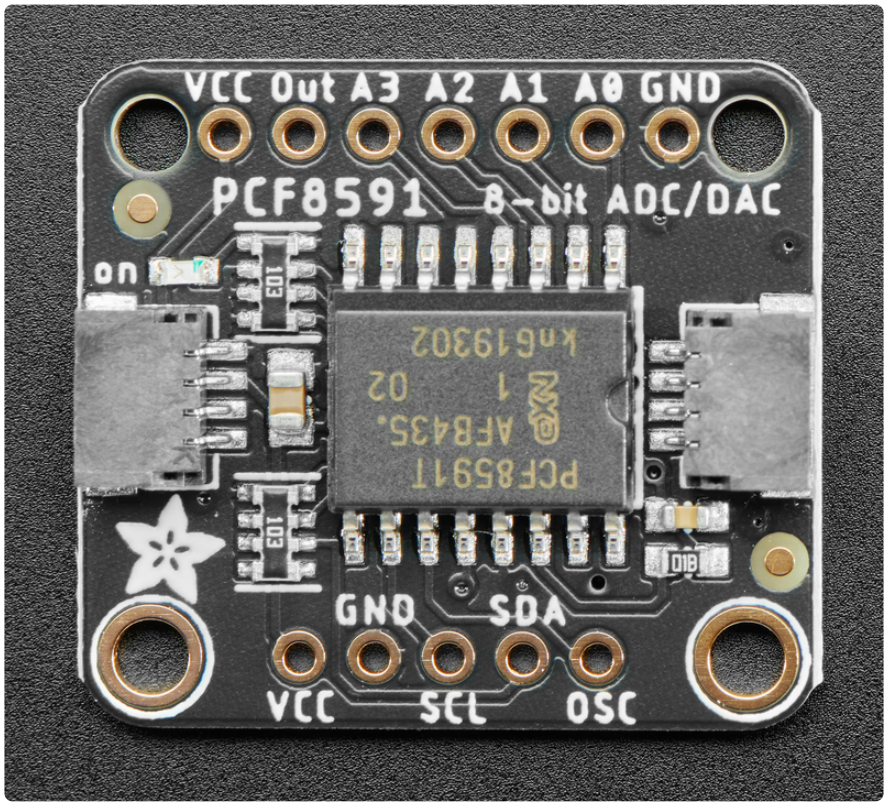
"Wow!" you say, "That sounds great, but surely with all that fun in one small package, I can only use one at a time."

Well my friend, I've got good news. On the back of each PCF8591 breakout are three jumpers that allow you to set the I2C address, allowing the use of eight PCF8591s on the same I2C bus! That's up to a a whopping 32 channels of analog measurement and 8 channels of analog signal generation! If you do decide to go all out with your project, you won't break the bank because the PCF8591 breakouts are reasonably priced compared to some higher-end ADCs.



As if 4 ADCs and one DAC in a single package weren't enough, we've made it even easier to use by mounting the PCF8591's chonky SO16 surface mount package onto a breakout with standard 0.1"/2.54mm headers and SparkFun Qwiic () compatible STEM MA QT () connectors for the I2C bus, making it simple to integrate with your project. Using the wiring diagrams and example code on the pages that follow, you'll be able to use our Python () and Arduino () libraries to easily measure as many voltages as you need (as long as that number is 32 or less).

Pinouts



Power Pins

- VCC - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V
- GND - common ground for power and logic

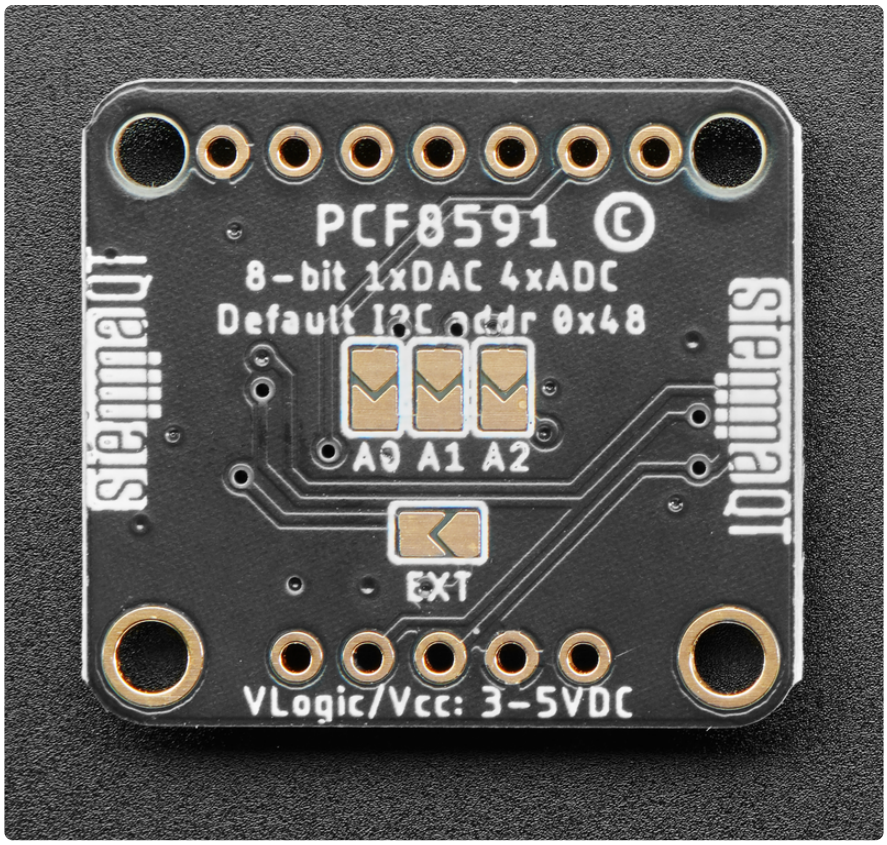
The voltage given to the VCC pin will serve as the ADC's reference voltage

I2C Logic Pins

- SCL - I2C clock pin, connect to your microcontroller I2C clock line. This pin is 3.3V and 5V compatible, and there's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontroller I2C data line. This pin is 3.3V and 5V compatible, and there's a 10K pullup on this pin.
- STEMMA QT () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()

Analog Pins

- A0-A3 - ADC Inputs, Connect these to the voltage you want to measure
- Out - DAC Output. Connect to a circuit to create a voltage



Other Pins and Jumpers

- OSC - Oscillator input/output. Outputs the oscillator signal used by the ADC. You can use it to supply an external oscillator using the EXT jumper
- EXT Solder Jumper - OSC Switch. Short to use an external oscillator
- AD0, AD1, AD2 Jumpers - I2C Address jumpers. Bridging the solder jumpers on the back will allow you to change the I2C address from the default 0x48 (72) to one between 0x49 (73) and 0x4F (79).

Shorting A0 will add 1 to the address, shorting A1 will add 2, and A2 will add 4. For example

```
0x48 + 1 (A0) + 4(A2) = 0x48+5 =0x4D
```

or

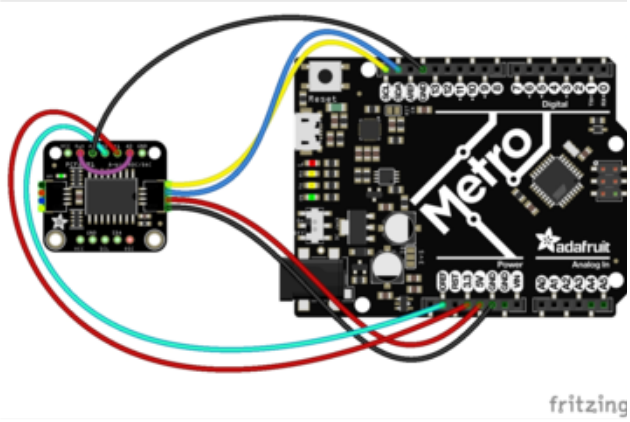
```
0x48 + 1 (A0 + 2 (A2) = 0x48 + 3 = 0x4B
```

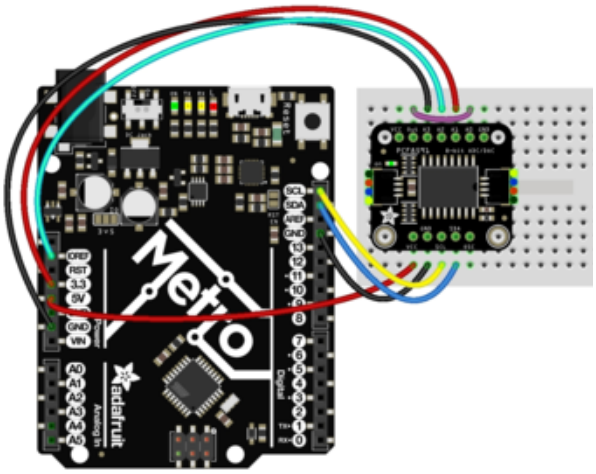
Arduino

Using the PCF8591 with Arduino is a simple matter of wiring up it to your Arduino-compatible microcontroller, installing the Adafruit PCF8591 () library we've written, and running the provided example code.

I2C Wiring

Wiring the PCF8591 is made simple by using the I2C interface. The default I2C address for the PCF8591 is 0x48 but it can be switched to several up to 0x4F by shorting the address jumpers. See the Pinouts page for details



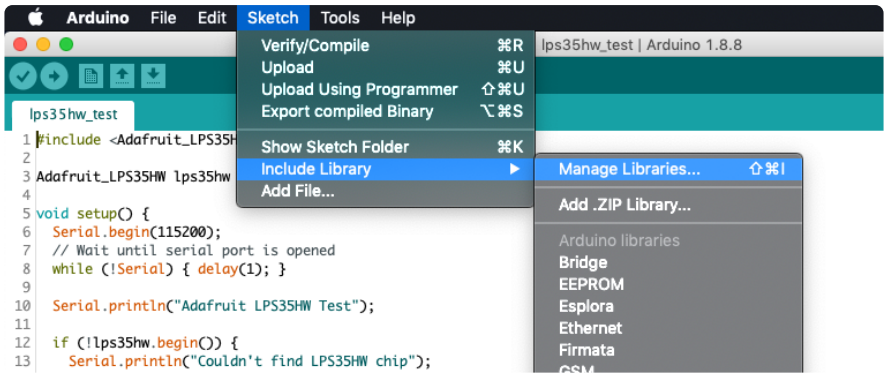


Connect PCF8591 VCC (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.
Connect PCF8591 GND (black wire) to Arduino GND
Connect PCF8591 SCL (yellow wire) to Arduino SCL
Connect PCF8591 SDA (blue wire) to Arduino SDA
For our test program, we expect you'll wire up the ADC / DAC pins as well:

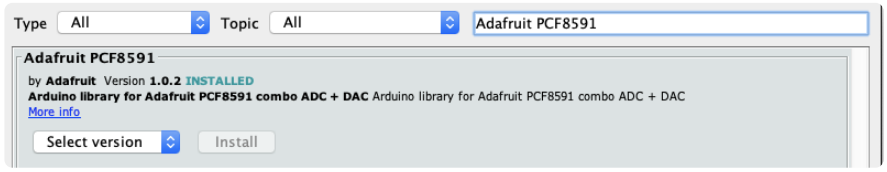
Use a jumper wire to connect the PCF8591's A0 pin to the Out pin
Use another wire to connect PCF8591's A1 pin to the Arduino 3.3V pin.
Connect PCF8591 A2 pin to Arduino IORef
Connect PCF8691 A3 to Arduino GND

Library Installation

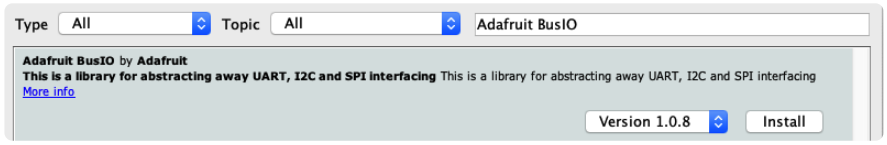
You can install the Adafruit PCF8591 () library for Arduino using the Library Manager in the Arduino IDE.



Click the Manage Libraries ... menu item, search for Adafruit PCF8591, and select the Adafruit PCF8591 library:



Follow the same process for the Adafruit BusIO library.



Load Example

Open up File -> Examples -> Adafruit PCF8591 -> PCF8591

After opening the demo file, upload to your Arduino wired up to the sensor. Once you upload the code, you will see the ADC values being printed when you open the Serial Monitor (Tools->Serial Monitor) at 115200 baud, similar to this:

```
# Adafruit PCF8591 demo
# Adafruit PCF8591 found
AIN0, AIN1, AIN2, AIN3
0.00V, 3.29V, 5.00V, 0.00V
0.02V, 3.29V, 5.00V, 0.00V
0.02V, 3.29V, 5.00V, 0.00V
0.04V, 3.29V, 5.00V, 0.00V
0.06V, 3.29V, 5.00V, 0.00V
0.08V, 3.29V, 5.00V, 0.00V
0.10V, 3.29V, 5.00V, 0.00V
0.12V, 3.29V, 5.00V, 0.00V
0.14V, 3.29V, 5.00V, 0.00V
0.16V, 3.29V, 5.00V, 0.00V
0.18V, 3.29V, 5.00V, 0.00V
```

Example Code

```
#include <Adafruit_PCF8591.h>
// Make sure that this is set to the value in volts of VCC
#define ADC_REFERENCE_VOLTAGE 5.0
Adafruit_PCF8591 pcf = Adafruit_PCF8591();

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

  Serial.println("# Adafruit PCF8591 demo");
  if (!pcf.begin()) {
    Serial.println("# Adafruit PCF8591 not found!");
    while (1)
      delay(10);
  }
  Serial.println("# Adafruit PCF8591 found");
  pcf.enableDAC(true);

  Serial.println("AIN0, AIN1, AIN2, AIN3");
}

uint8_t dac_counter = 0;

void loop() {
  // Make a triangle wave on the DAC output
  pcf.analogWrite(dac_counter++);

  Serial.print(int_to_volts(pcf.analogRead(0), 8, ADC_REFERENCE_VOLTAGE));
  Serial.print("V, ");
  Serial.print(int_to_volts(pcf.analogRead(1), 8, ADC_REFERENCE_VOLTAGE));
  Serial.print("V, ");
  Serial.print(int_to_volts(pcf.analogRead(2), 8, ADC_REFERENCE_VOLTAGE));
  Serial.print("V, ");
  Serial.print(int_to_volts(pcf.analogRead(3), 8, ADC_REFERENCE_VOLTAGE));
  Serial.print("V");
  Serial.println("");
  delay(1000);
}

float int_to_volts(uint16_t dac_value, uint8_t bits, float logic_level) {
  return (((float)dac_value / ((1 << bits) - 1)) * logic_level);
}
```

Arduino Docs

Arduino Docs ()

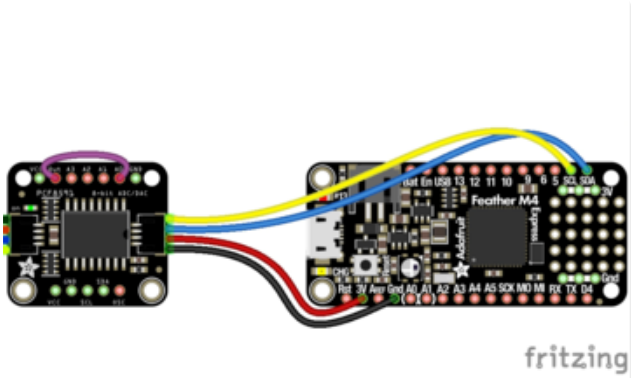
Python & CircuitPython

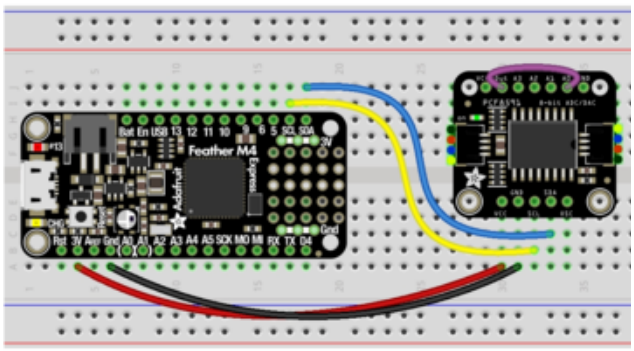
It's easy to use the PCF8591 with Python or CircuitPython, and the Adafruit CircuitPython PCF8591 () module. This library allows you to easily write Python code that reads ADC values and set DAC voltages

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 ().

CircuitPython Microcontroller Wiring

Wire up a PCF8591 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using a solderless breadboard



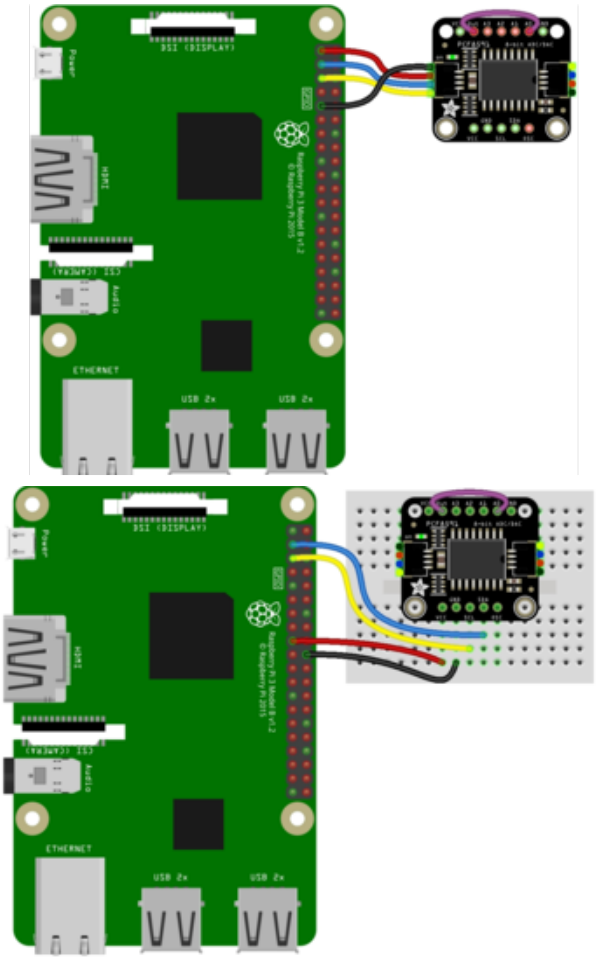


Board 3V to PCF8591 VIN (red wire)
Board GND to PCF8591 GND (black wire)
Board SCL to PCF8591 SCL (yellow wire)
Board SDA to PCF8591 SDA (blue wire)
PCF8591 A0 to PCF8591 Out

Python Computer Wiring

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

Here's the Raspberry Pi wired to the sensor using I2C and a solderless breadboard



Board 3V to PCF8591 VIN (red wire)
Board GND to PCF8591 GND (black wire)
Board SCL to PCF8591 SCL (yellow wire)
Board SDA to PCF8591 SDA (blue wire)
PCF8591 A0 to PCF8591 Out

CircuitPython Installation of PCF8591 Library

You'll need to install the Adafruit CircuitPython PCF8591 () library on your CircuitPython board.

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

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle ().

Our CircuitPython starter guide has a great page on how to install the library bundle ().

Before continuing make sure your board's lib folder or root filesystem has the adafruit_PCF8591.mpy file and adafruit_bus_device folder copied over.

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

Python Installation of PCF8591 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3.

Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready ()!

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

```
sudo pip3 install adafruit-circuitpython-pcf8591
```

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

CircuitPython & Python Usage

To demonstrate the usage of the ADC and DAC we'll initialize it and set the DAC and read the ADC from the board's Python REPL.

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

```
import time
import board

import adafruit_pcf8591.pcf8591 as PCF
from adafruit_pcf8591.analog_in import AnalogIn
from adafruit_pcf8591.analog_out import AnalogOut

i2c = board.I2C()
pcf = PCF.PCF8591(i2c)
```

The Adafruit_CircuitPython_PCF8591 library assumes a 3.3V logic level and reference voltage. See the library documentation for details on how to specify a different reference voltage

Next we will create an `AnalogIn` and `AnalogOut` from the PCF8591 to allow us to set and read voltages the same way as with other CircuitPython boards with built in DACs and ADCs

```
pcf_in_0 = AnalogIn(pcf, PCF.A0)
pcf_out = AnalogOut(pcf, PCF.OUT)
```

```
>>> pcf_in_0 = AnalogIn(pcf, PCF.A0)
>>> pcf_out = AnalogOut(pcf, PCF.OUT)
>>>
```

Now that we are all set up, we will use the code below to set the DAC's value and then read the voltage that was set using the ADC. A little math will translate the raw 16-bit (scaled from 8-bit) value to a voltage level.

```
pcf_out.value = 65535
raw_value = pcf_in_0.value
scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage
print("Pin 0: %0.2fV" % (scaled_value))
print("")
```

```
>>> pcf_out.value = 65535
>>> raw_value = pcf_in_0.value
>>> scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage
>>> print("Pin 0: %0.2fV" % (scaled_value))
Pin 0: 2.85V
>>>
```

Next we'll repeat the process but this time we'll specify a DAC value that was half the previous amount. Note how the voltage level changes as a result

```
pcf_out.value = 32767
raw_value = pcf_in_0.value
scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage

print("Pin 0: %0.2fV" % (scaled_value))
```

```
>>> pcf_out.value = 32767
>>> raw_value = pcf_in_0.value
>>> scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage
>>>
>>> print("Pin 0: %0.2fV" % (scaled_value))
Pin 0: 0.83V
>>>
```

Finally we'll set the DAC value to zero and use the ADC to verify that the DAC can go all the way down to 0V/GND

```
pcf_out.value = 0
raw_value = pcf_in_0.value
scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage

print("Pin 0: %0.2fV" % (scaled_value))
```

```
>>> pcf_out.value = 0
>>> raw_value = pcf_in_0.value
>>> scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage
>>>
>>> print("Pin 0: %0.2fV" % (scaled_value))
Pin 0: 0.00V
>>>
```

Example Code

```
# SPDX-FileCopyrightText: Copyright (c) 2020 Bryan Siepert for Adafruit Industries
# SPDX-License-Identifier: MIT
import time
import board

import adafruit_pcf8591.pcf8591 as PCF
from adafruit_pcf8591.analog_in import AnalogIn
from adafruit_pcf8591.analog_out import AnalogOut

############ AnalogOut & AnalogIn Example ##########################
#
# This example shows how to use the included AnalogIn and AnalogOut
# classes to set the internal DAC to output a voltage and then measure
# it with the first ADC channel.
#
# Wiring:
# Connect the DAC output to the first ADC channel, in addition to the
# normal power and I2C connections
#
####################################################################
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
pcf = PCF.PCF8591(i2c)

pcf_in_0 = AnalogIn(pcf, PCF.A0)
pcf_out = AnalogOut(pcf, PCF.OUT)

while True:

    print("Setting out to ", 65535)
    pcf_out.value = 65535
    raw_value = pcf_in_0.value
    scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage

    print("Pin 0: %0.2fV" % (scaled_value))
    print("")
    time.sleep(1)

    print("Setting out to ", 32767)
    pcf_out.value = 32767
    raw_value = pcf_in_0.value
    scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage

    print("Pin 0: %0.2fV" % (scaled_value))
    print("")
    time.sleep(1)

    print("Setting out to ", 0)
    pcf_out.value = 0
    raw_value = pcf_in_0.value
    scaled_value = (raw_value / 65535) * pcf_in_0.reference_voltage

    print("Pin 0: %0.2fV" % (scaled_value))
    print("")
    time.sleep(1)
```

Python Docs

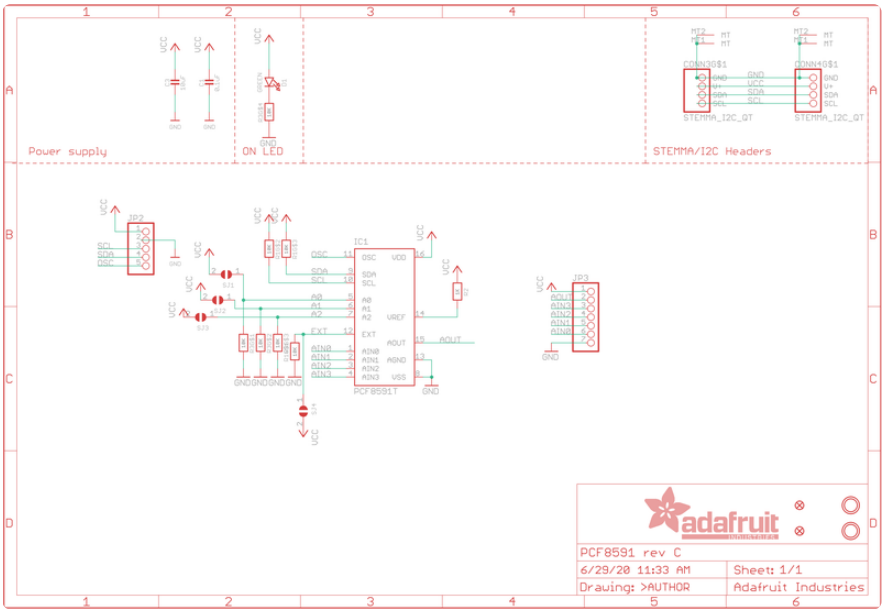
Python Docs ()

Downloads

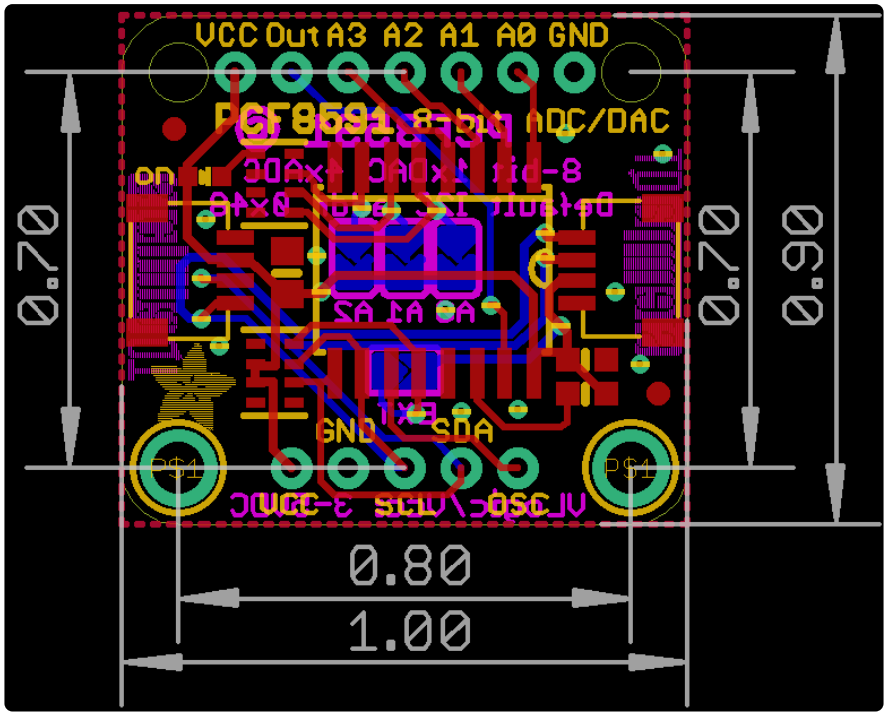
Files

- PCF8591 Datasheet ()
- EagleCAD files on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic



Fab Print


VCC Out A3 A2 A1 A0 GND
PCF8591
ADC/DAC
GND
SDA
0.70
0.70
0.90
0.80
1.00