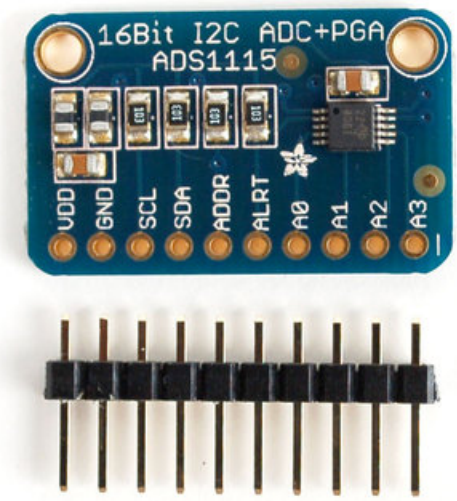
adafruit learning system

Adafruit 4-Channel ADC Breakouts

Created by Bill Earl



Last updated on 2017-11-21 02:03:21 AM UTC

Guide Contents

Guide Contents 2
Overview 3
ADS1115 Features: 3
ADS1015 Features: 3
Assembly and Wiring 5
Assembly: 5
- Prepare the header strip 5
- Position the breakout board 5
- Solder! 5

Wiring: 5
- Power 5
- I2C Connections 6
- I2C "Classic" 6
- I2C Addressing 7
- Multiple Boards 7

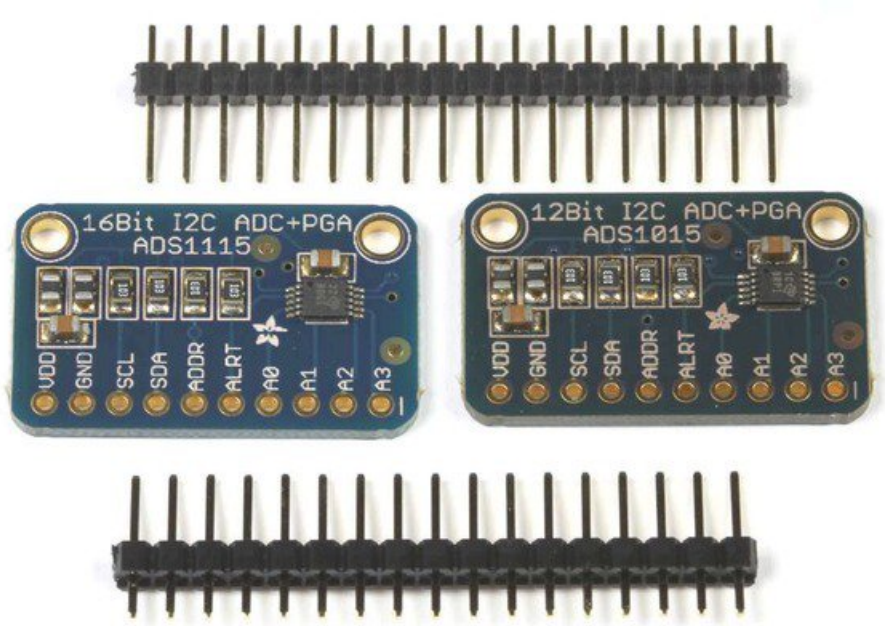
Signal Connections 9
- Single Ended vs. Differential Inputs: 9
- Which should I use? 9
- Single Ended Connections: 9
- Differential Connections: 10

Arduino Code 11
- Construction and Initialization: 11
- Single Ended Conversion: 11
- Differential Conversion: 12
- Comparator Operation: 13
- Adjusting Gain 14
- Example 15

CircuitPython Code 16
Usage 17
- Single Ended Mode 17
- Differential Mode 18

Downloads 20
Software 20
Files 20
Schematic (Identical For Both) 20
Fabrication Print (Identical For Both) 20

Overview



The ADS1115 and ADS1015 4-channel breakout boards are perfect for adding high-resolution analog to digital conversion to any microprocessor-based project. These boards can run with power and logic signals between 2v to 5v, so they are compatible with all common 3.3v and 5v processors. As many of 4 of these boards can be controlled from the same 2-wire I2C bus, giving you up to 16 single-ended or 8 differential channels. A programmable gain amplifier provides up to x16 gain for small signals.

These two boards are very similar, differing only in resolution and speed. The ADS1115 has higher resolution and the ADS1015 has a higher sample rate.

ADS1115 Features:

- Resolution: 16 Bits
- Programmable Sample Rate: 8 to 860 Samples/Second
- Power Supply/Logic Levels: 2.0V to 5.5V
- Low Current Consumption: Continuous Mode: Only 150µA Single-Shot Mode: Auto Shut-Down
- Internal Low-Drift Voltage Reference
- Internal Oscillator
- Internal PGA: up to x16
- I2C Interface: 4-Pin-Selectable Addresses
- Four Single-Ended or 2 Differential Inputs
- Programmable Comparator

ADS1015 Features:

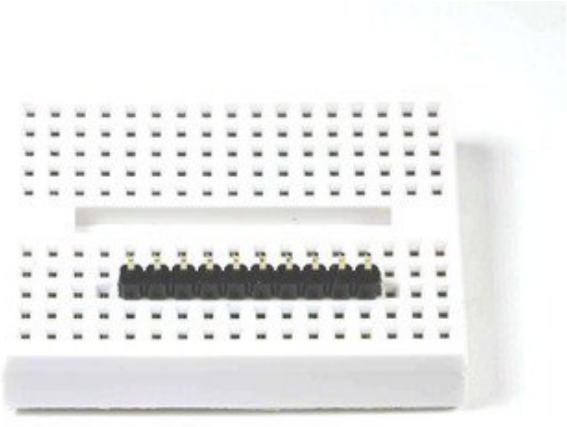
- Resolution: 12 Bits
- Programmable Sample Rate: 128 to 3300 Samples/Second
- Power Supply/Logic Levels: 2.0V to 5.5V
- Low Current Consumption: Continuous Mode: Only 150µA Single-Shot Mode: Auto Shut-Down
- Internal Low-Drift Voltage Reference
- Internal Oscillator

- Internal PGA: up to x16
- I2C Interface: 4-Pin-Selectable Addresses
- Four Single-Ended or 2 Differential Inputs
- Programmable Comparator

Assembly and Wiring

Assembly:

The board comes with all surface-mount parts pre-soldered. For breadboard use, the included header-strip should be soldered on:

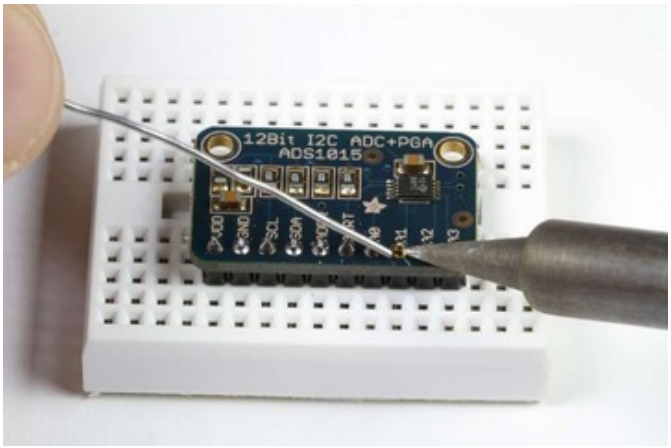


Prepare the header strip

Cut the supplied header strip to length and insert it long-pins-down in your breadboard to hold it for soldering.

Position the breakout board

Place the breakout board on the header pins.



Solder!

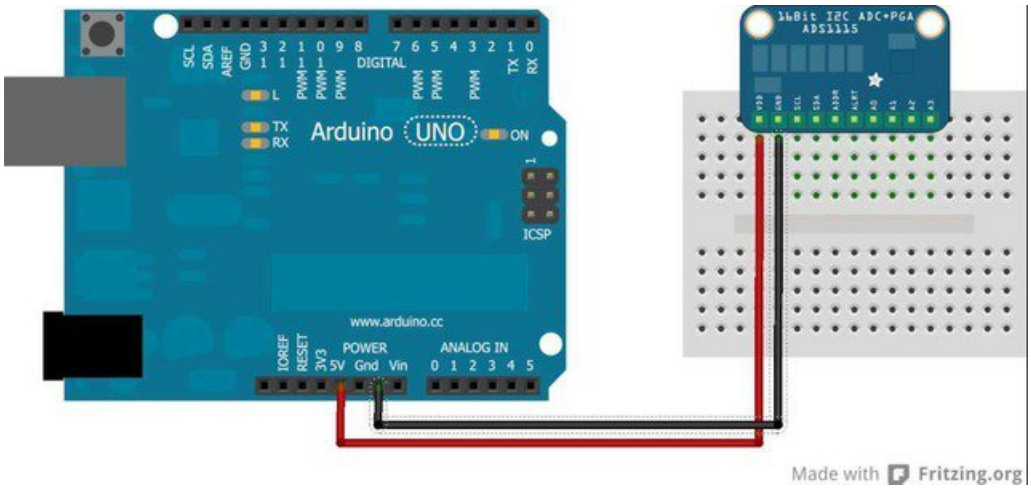
Solder each pin for a good electrical connection.

Wiring:

Power

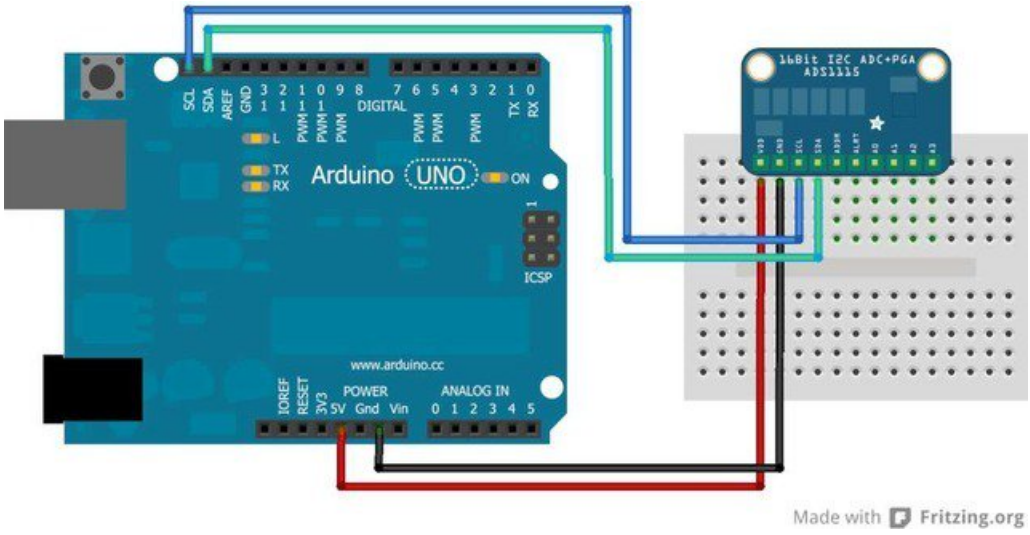
First connect VDD and GND. These boards will work with either a 3.3v or a 5v supply. The diagram below shows connection to the Arduino 5v pin.

> The absolute maximum analog input voltage is VDD + 0.3v. To avoid damage to the chip, do not attempt to measure voltages greater than VDD.



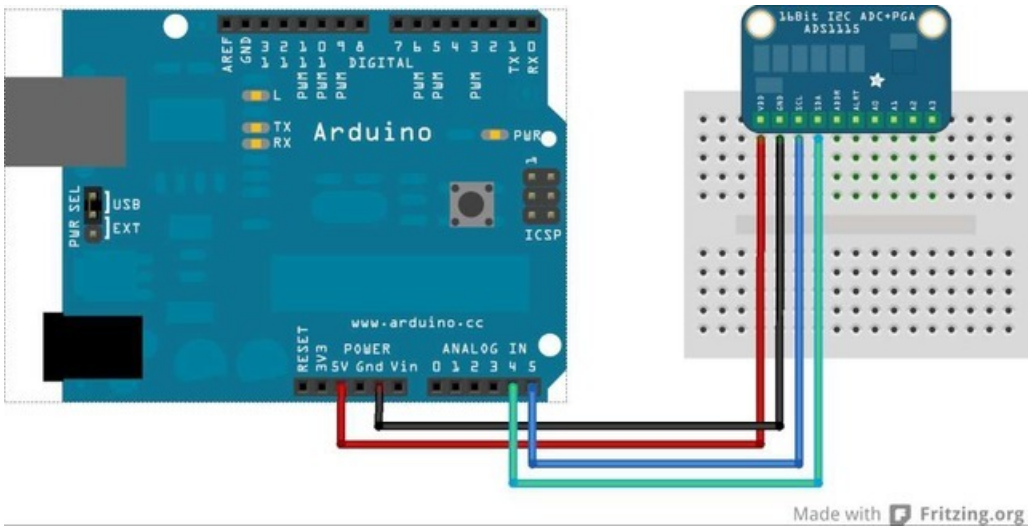
I2C Connections

I2C requires just 2 pins to communicate. These can be shared with other I2C devices. For R3 and later Arduinos (including MEGA and DUE models), connect SDA->SDA and SCL->SCL.



I2C "Classic"

For older Arduino boards without dedicated SDA and SCL pins, connect as shown below. (For older Arduino Megas, SDA and SCL are on pins 20 and 21)

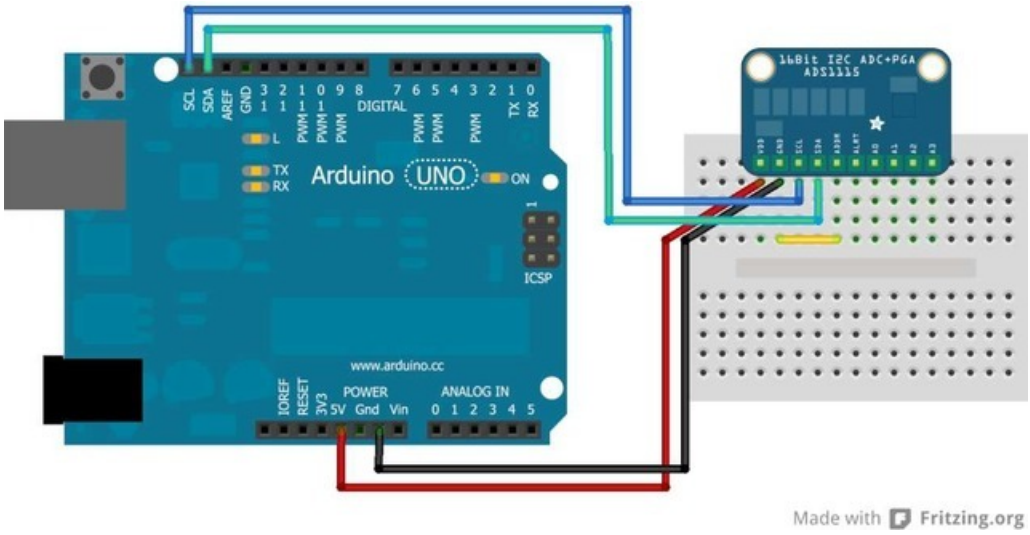


I2C Addressing

The ADS11x5 chips have a base 7-bit I2C address of 0x48 (1001000) and a clever addressing scheme that allows four different addresses using just one address pin (named **ADR** for ADdRess). To program the address, connect the address pin as follows:

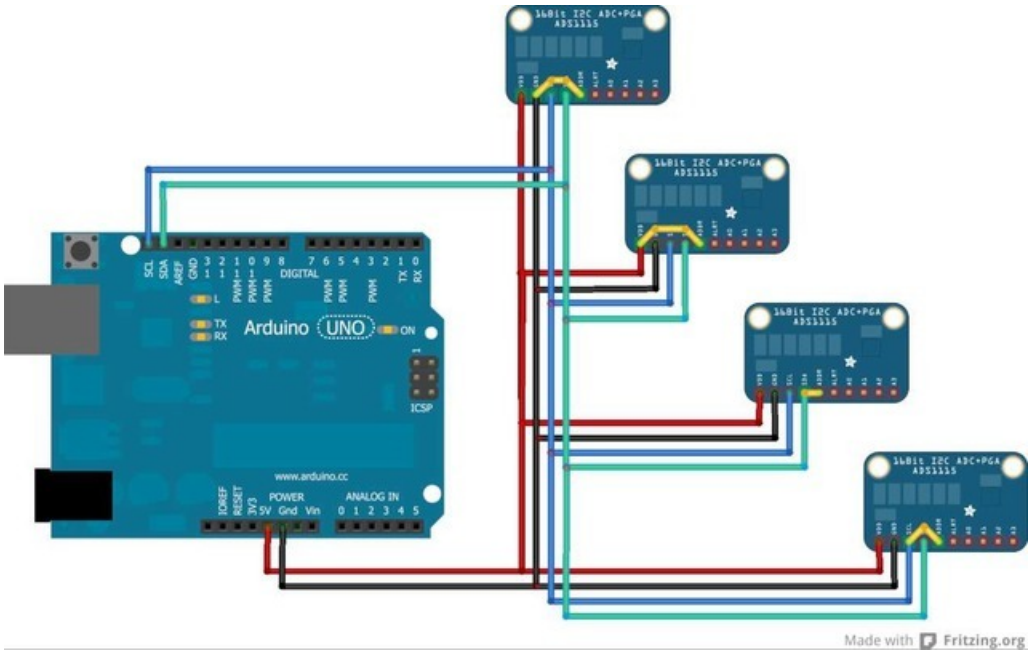
- 0x48 (1001000) ADR -> GND
- 0x49 (1001001) ADR -> VDD
- 0x4A (1001010) ADR -> SDA
- 0x4B (1001011) ADR -> SCL

The following diagram shows one board addressed as 0x48:

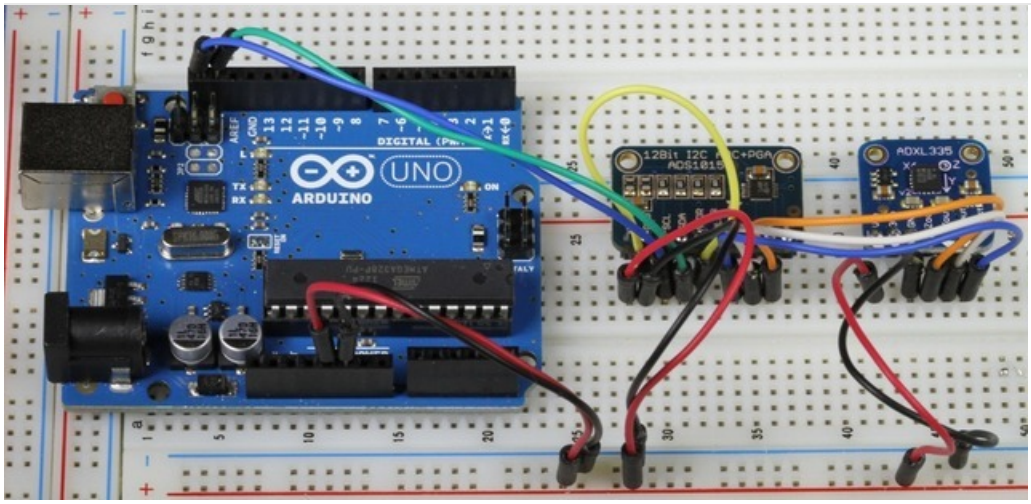


Multiple Boards

By assigning each board a different address, up to 4 boards can be connected as below:


16Bit I2C ADC+PGA
ADS1115
Arduino UNO
www.arduino.cc
Made with Fritzing.org

Signal Connections



Single Ended vs. Differential Inputs:

The ADS1x15 breakouts support up to 4 SIngle Ended or 2 Differential inputs.

Single Ended inputs measure the voltage between the analog input channel (A0-A3) and analog ground (GND).

Differential inputs measure the voltage between two analog input channels. (A0&A1 or A2&A3).

Which should I use?

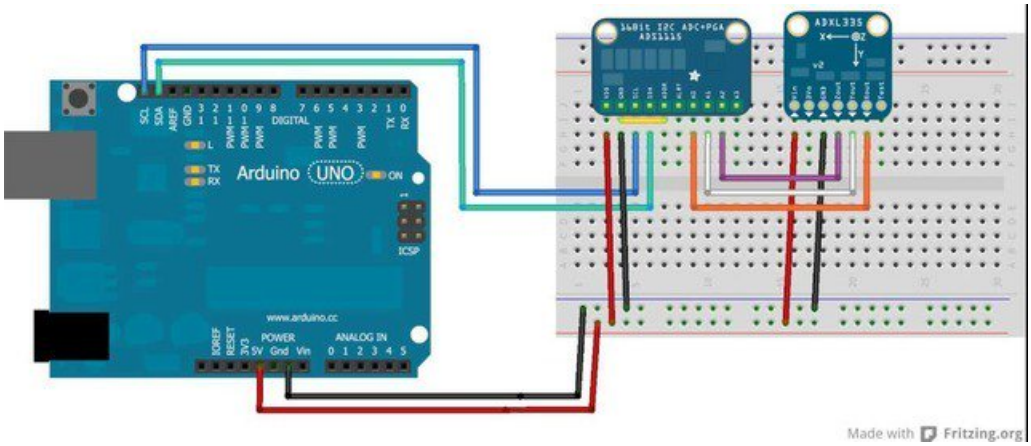
Single ended inputs give you twice as many inputs. So why would you want to use differential inputs?

Single ended inputs can, by definition, only measure positive voltages. Without the sign bit, you only get an effective 15 bit resolution.

In addition to providing the full 16 bits of resolution and the ability to measure negative voltages, Differential measurements offer more immunity from electromagnetic noise. This is useful when using long signal wires or operating in an electrically noisy environment. This is also desirable when dealing with small signals requiring high gain, since the gain will amplify the noise as well as the signal.

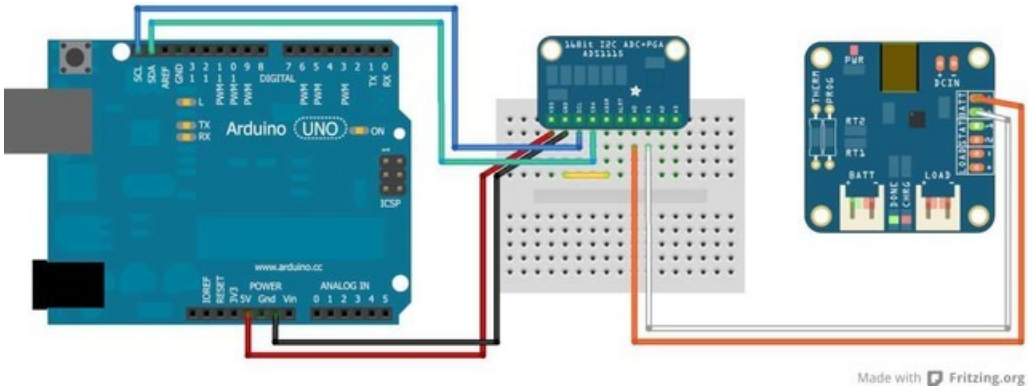
Single Ended Connections:

Connect the signal wire to one of the analog input channels (A0 - A3). Connect the ground wire to GND. This diagram shows how to connect an ADXL335 to for measurement of the X, Y and Z axis on analog channels A0, A1 and A2.



Differential Connections:

Differential measurements use a pair of input pins, either A0&A1 or A2&A3. The following diagram shows connections for differential measurement of the battery voltage on a LiPo charger board.



All input signals to these devices must be between ground potential and VCC. If your source signal produces negative voltages, they must be offset to fall within the GND to VCC range of the ASD1x15.

Arduino Code

The Adafruit_ADS1x15 library supports both single-ended and differential readings as well as comparator operations on both the ADS1015 and ADS1115 breakout boards. The library uses the wiring library for I2C communication, so wiring.h must be included.

Construction and Initialization:

Adafruit_ADS1015();
Construct an instance of an ADS1015 with the default address (0x48)

Adafruit_ADS1015(uint8_t addr);
Construct an instance of an ADS1015 with the specified address (0x48 - 0x4B)

Adafruit_ADS1115();
Construct an instance of an ADS1115 with the default address (0x48)

Adafruit_ADS1115(uint8_t addr);
Construct an instance of an ADS1115 with the specified address (0x48 - 0x4B)

void begin(void);
Initialize the ADC for operation.

Example:

> The following examples assume an ADS1015 and use a 3 mV/bit scaling factor. For the higher-resolution ADS1115, the scaling factor would be 188uV/bit.

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1015 ads1015;   // Construct an ads1015 at the default address: 0x48
Adafruit_ADS1115 ads1115(0x49); // construct an ads1115 at address 0x49

void setup(void)
{
  ads1015.begin();  // Initialize ads1015
  ads1115.begin();  // Initialize ads1115
}
```

Single Ended Conversion:

uint16_t readADC_SingleEnded(uint8_t channel);
Perform a single-ended analog to digital conversion on the specified channel.

Example:

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1015 ads1015;

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Hello!");

  Serial.println("Getting single-ended readings from AIN0..3");
  Serial.println("ADC Range: +/- 6.144V (1 bit = 3mV)");
  ads1015.begin();
}

void loop(void)
{
  int16_t adc0, adc1, adc2, adc3;

  adc0 = ads1015.readADC_SingleEnded(0);
  adc1 = ads1015.readADC_SingleEnded(1);
  adc2 = ads1015.readADC_SingleEnded(2);
  adc3 = ads1015.readADC_SingleEnded(3);
  Serial.print("AIN0: "); Serial.println(adc0);
  Serial.print("AIN1: "); Serial.println(adc1);
  Serial.print("AIN2: "); Serial.println(adc2);
  Serial.print("AIN3: "); Serial.println(adc3);
  Serial.println(" ");

  delay(1000);
}
```

Differential Conversion:

int16_t readADC_Differential_0_1(void);

Perform a differential analog to digital conversion on the voltage between channels 0 and 1.

int16_t readADC_Differential_2_3(void);

Perform a differential analog to digital conversion on the voltage between channels 2 and 3.

Example:

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1015 ads1015;

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Hello!");

  Serial.println("Getting differential reading from AIN0 (P) and AIN1 (N)");
  Serial.println("ADC Range: +/- 6.144V (1 bit = 3mV)");
  ads1015.begin();
}

void loop(void)
{
  int16_t results;

  results = ads1015.readADC_Differential_0_1();
  Serial.print("Differential: "); Serial.print(results); Serial.print("("); Serial.print(results * 3); S

  delay(1000);
}
```

Comparator Operation:

Comparator mode allows you to compare an input voltage with a threshold level and generate an alert signal (on the ALRT pin) if the threshold is exceeded. This pin can be polled with a digital input pin, or it can be configured to generate an interrupt.

void startComparator_SingleEnded(uint8_t channel, int16_t threshold);
Set the threshold and channel for comparator operation.

int16_t getLastConversionResults();
Get the last conversion result and clear the comparator.

Example:

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1015 ads1015;

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Hello!");

  Serial.println("Single-ended readings from AIN0 with >3.0V comparator");
  Serial.println("ADC Range: +/- 6.144V (1 bit = 3mV)");
  Serial.println("Comparator Threshold: 1000 (3.000V)");
  ads1015.begin();

  // Setup 3V comparator on channel 0
  ads1015.startComparator_SingleEnded(0, 1000);
}

void loop(void)
{
  int16_t adc0;

  // Comparator will only de-assert after a read
  adc0 = ads1015.getLastConversionResults();
  Serial.print("AIN0: "); Serial.println(adc0);

  delay(100);
}
```

Adjusting Gain

To boost small signals, the gain can be adjusted on the ADS1x15 chips in the following steps:

- **GAIN_TWOTHIRDS** (for an input range of +/- 6.144V)
- **GAIN_ONE** (for an input range of +/-4.096V)
- **GAIN_TWO** (for an input range of +/-2.048V)
- **GAIN_FOUR** (for an input range of +/-1.024V)
- **GAIN_EIGHT** (for an input range of +/-0.512V)
- **GAIN_SIXTEEN** (for an input range of +/-0.256V)

adsGain_t getGain(void)

Reads the current gain value (default = 2/3x)

```
adsGain_t gain = getGain();
```

void setGain(adsGain_t gain)

Sets the gain for the ADS1x15

```
ads1015.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit = 3mV (default)

// ads1015.setGain(GAIN_ONE);     // 1x gain   +/- 4.096V  1 bit = 2mV
// ads1015.setGain(GAIN_TWO);     // 2x gain   +/- 2.048V  1 bit = 1mV
// ads1015.setGain(GAIN_FOUR);    // 4x gain   +/- 1.024V  1 bit = 0.5mV
// ads1015.setGain(GAIN_EIGHT);   // 8x gain   +/- 0.512V  1 bit = 0.25mV
// ads1015.setGain(GAIN_SIXTEEN); // 16x gain  +/- 0.256V  1 bit = 0.125mV
```

Example

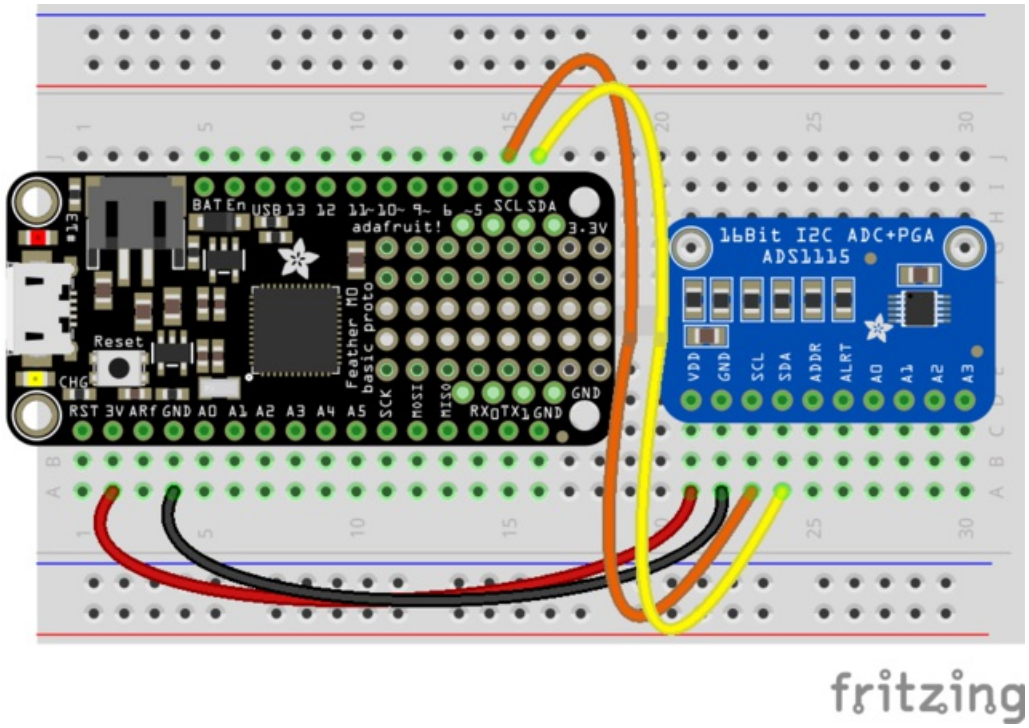
If we had an analog sensor with an output voltage ~1V (a TMP36, for example), we could set the gain on the ADC to **GAIN_FOUR**, which would give us a +/-1.024V range. This would push the 1V input signal over the entire 12-bit or 16-bit range of the ADC, compared to the very limited range 1V would cover without adjusting the gain settings

```
// Set the gain to 4x, for an input range of +/- 1.024V
// 1-bit = 0.5V on the ADS1015 with this gain setting
ads1015.setGain(GAIN_FOUR);
```

CircuitPython Code

It's easy to use the ADS1115 and ADS1015 sensor with CircuitPython and the Adafruit CircuitPython ADS1x15 module. This module allows you to easily write Python code that reads the humidity, temperature, pressure, and more from the sensor.

First wire up a ADC to your board exactly as shown on the previous pages for Arduino using an I2C interface. Here's an example of wiring a Feather M0 to the ADS1115 with I2C:



- **Board 3V** to **sensor VDD** - Remember the **maximum input voltage** to any ADC channel cannot exceed this VDD 3V value!
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

Next you'll need to install the Adafruit CircuitPython ADS1x15 library on your CircuitPython board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

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. For example the Circuit Playground Express guide has a great page on how to install the library bundle for both express and non-express boards.

Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- **adafruit_ads1x15**
- **adafruit_bus_device**

You can also download the **adafruit_ads1x15** folder from its releases page on Github.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_ads1x15,** and **adafruit_bus_device** files and folders copied over.

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

Usage

To demonstrate the usage of the sensor we'll initialize it and read the ADC channel values from the board's Python REPL. First run the following code to import the necessary modules and initialize the I2C connection:

```
import board
import busio
i2c = busio.I2C(board.SCL, board.SDA)
```

Remember if you're using a board that doesn't support hardware I2C (like the ESP8266) you need to use the **bitbangio** module instead:

```
import board
import bitbangio
i2c = bitbangio.I2C(board.SCL, board.SDA)
```

Next you need to decide if you're using the sensor in single ended or differential mode. Carefully read the datasheet to understand the differences, but at a high level single ended mode will read the voltage of each input and convert it directly to a digital value. Differential mode will read the difference in voltage between pairs of channels, like the difference between channel 0 and 1, and convert that to a digital value. Differential mode is handy when you're comparing a sensor to a known value.

Single Ended Mode

For single ended mode you need to import the single ended versions of the library:

```
import adafruit_ads1x15.single_ended as ads1x15
```

Then you need to create and instance of either the ADS1015 or ADS1115 class, depending on the board you're using. For example to create the ADS1015:

```
ads = ads1x15.ADS1015(i2c)
```

Or to create the ADS1115:

```
ads = ads1x15.ADS1115(i2c)
```

Now you can read the raw value and voltage of each of the board's 4 channels. The ads class instance actually acts like a list and you can index into it to choose the channel, then read the **volts** or **value** property.

For example to print the voltage and value of all channels:

```
for i in range(4):
    volts = ads[i].volts
    value = ads[i].value
    print('Channel {} voltage: {}V value: {}'.format(i, volts, value))
```

```
>>> for i in range(4):
...     volts = ads[i].volts
...     value = ads[i].value
...     print('Channel {} voltage: {}V value: {}'.format(i, volts, value))
...
Channel 0 voltage: 0.597268V value: 4763
Channel 1 voltage: 0.599643V value: 4778
Channel 2 voltage: 0.599768V value: 4776
Channel 3 voltage: 0.599393V value: 4792
>>>
```

Here's a complete example of reading all the channels from the ADS1115 every second and printing their voltage & value. Save this as a **main.py** on your board and look for the output from the serial REPL. Remember if using the ESP8266 and **bitbangio** module you need to adjust the initialization as mentioned above!

```
import board
import busio
import time

import adafruit_ads1x15.single_ended as ads1x15

# Initialize I2C bus and sensor.
i2c = busio.I2C(board.SCL, board.SDA)
ads = ads1x15.ADS1115(i2c)

# Main loop runs forever printing channel readings every second.
while True:
    for i in range(4):
        volts = ads[i].volts
        value = ads[i].value
        print('Channel {} voltage: {}V value: {}'.format(i, volts, value))
    time.sleep(1.0)
```

Differential Mode

For differential mode (i.e. reading the difference in voltage between pairs of channels) you need to import and use a different part of the library:

```
import adafruit_ads1x15.differential as ads1x15
```

Then create an instance of the ADS1015 or ADS1115 class depending on the board you have connected. For example to create the ADS1015:

```
ads = ads1x15.ADS1015(i2c)
```

Or the ADS1115:

```
ads = ads1x15.ADS1115(i2c)
```

Just like with single ended mode you can index into the ads object to choose which pair of channels are read. However instead of passing a single value you pass a 2-tuple of channels. The following values are supported:

- **(0, 1)** - Read the difference of channel 0 minus channel 1
- **(0, 3)** - Read the difference of channel 0 minus channel 3
- **(1, 3)** - Read the difference of channel 1 minus channel 3
- **(2, 3)** - Read the difference of channel 2 minus channel 3

Again you can use the volts and value properties to read their respective values. Here's an example of printing these values for each channel pair above:

```
pairs = ((0,1), (0, 3), (1,3), (2,3))
for pair in pairs:
    volts = ads[pair].volts
    value = ads[pair].value
    print('Difference of channel {}-{} voltage: {}V value: {}'.format(pair[0], pair[1], volts, value))
```

```
>>> pairs = ((0,1), (0, 3), (1,3), (2,3))
>>> for pair in pairs:
...     volts = ads[pair].volts
...     value = ads[pair].value
...     print('Difference of channel {}-{} voltage: {}V value: {}'.format(pa
ir[0], pair[1], volts, value))
...
Difference of channel 0-1 voltage: 0.599518V value: 0
Difference of channel 0-3 voltage: 0.0V value: 0
Difference of channel 1-3 voltage: 0.0V value: -1
Difference of channel 2-3 voltage: 0.0V value: -21
>>>
```

That's all there is to using the ADS1015 and ADS1115 with CircuitPython!

Downloads

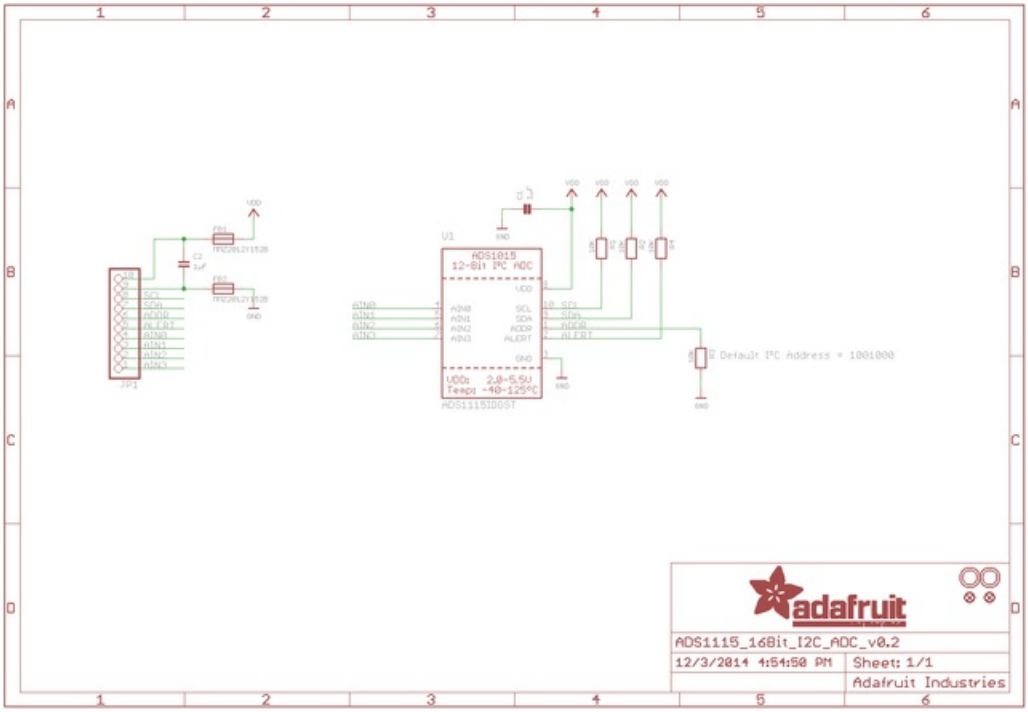
Software

- ADS1x15 Library for Arduino

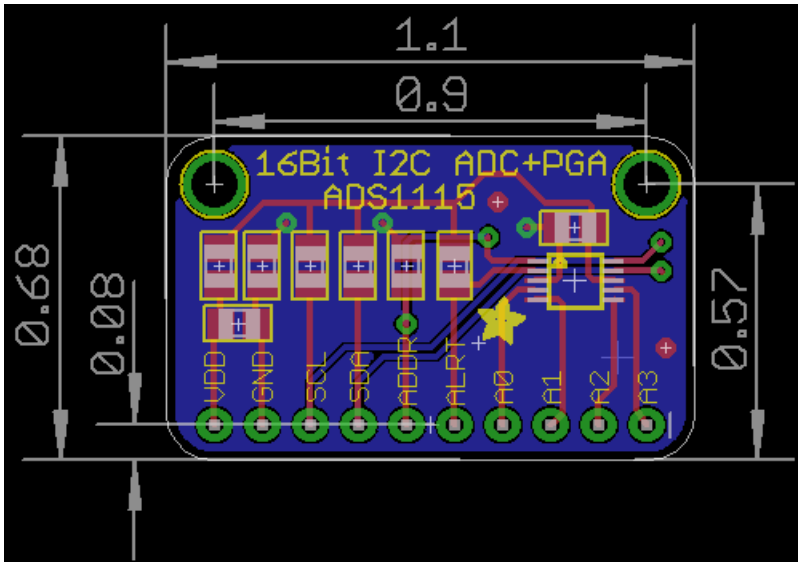
Files

- Board Files and Schematics
- Fritzing library
- ADS1015 Data Sheet
- ADS1115 Data Sheet

Schematic (Identical For Both)



Fabrication Print (Identical For Both)


1.1
0.9
16Bit I2C ADC+PGA
ADS1115
0.68
0.08
0.57
VDD
GND
SCL
SDA
ADDR
ALRT
A0
A1
A2
A3