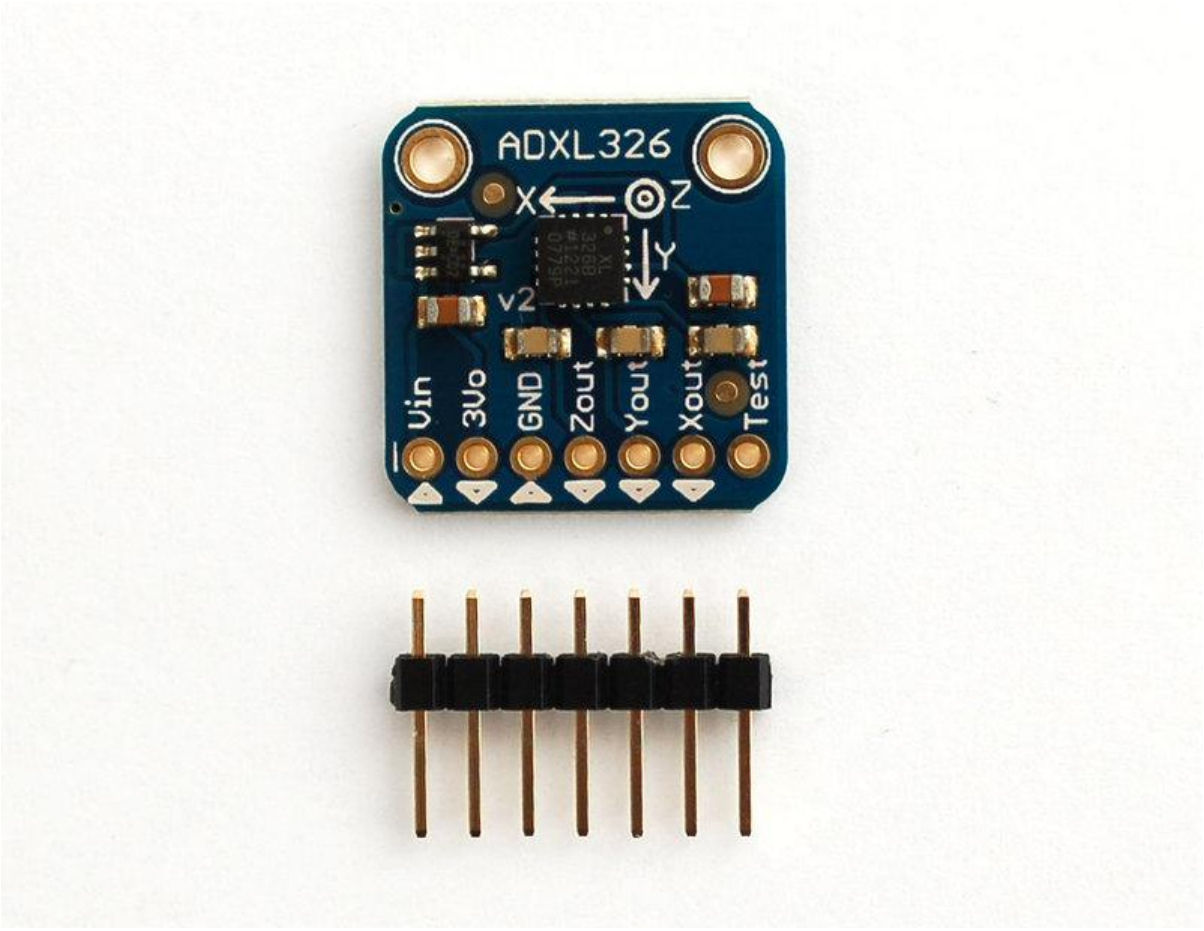
Adafruit Analog Accelerometer Breakouts

Created by Bill Earl



https://learn.adafruit.com/adafruit-analog-accelerometer-breakouts

Last updated on 2021-11-15 05:52:51 PM EST

Table of Contents

Overview 3

- How it Works: 4
- MEMS - Micro Electro-Mechanical Systems 4
- Ratiometric Output 4

Assembly 4

- Prepare the header strip: 5
- Add the breakout board: 5
- And Solder! 5

Arduino Wiring 5

- Wiring: 5
- Connect the Power: 5
- Connect the X, Y and Z Signal Outputs: 6
- Using the Voltage Reference: 6

Calibration and Programming 7

- Static Calibration: 7
- Gravity as a Calibration Reference 8
- Calibration Method: 8
- Run the Calibration Sketch 10
- Calibration Sketch Output: 10
- Calibration Sketch 11

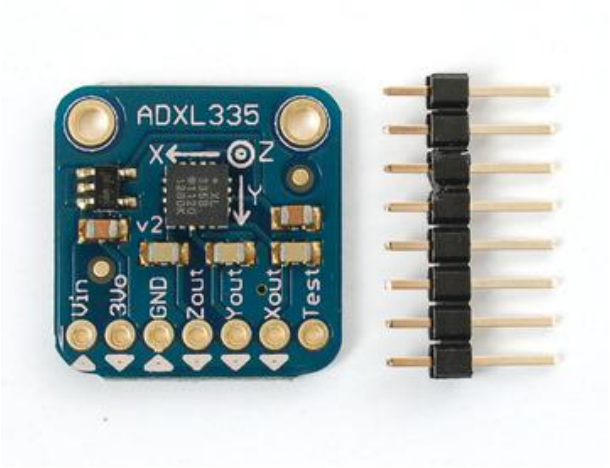
CircuitPython Code 13

Downloads 15

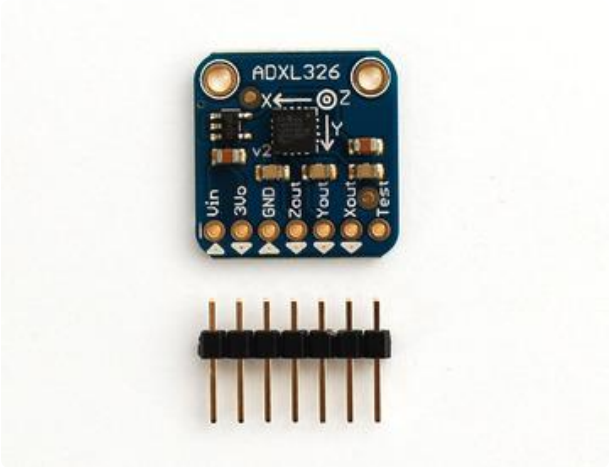
- Files 15
- ADXL326 Schematic & Fabrication Print 16
- ADXL377 Schematic & Fabrication Print 17

Overview

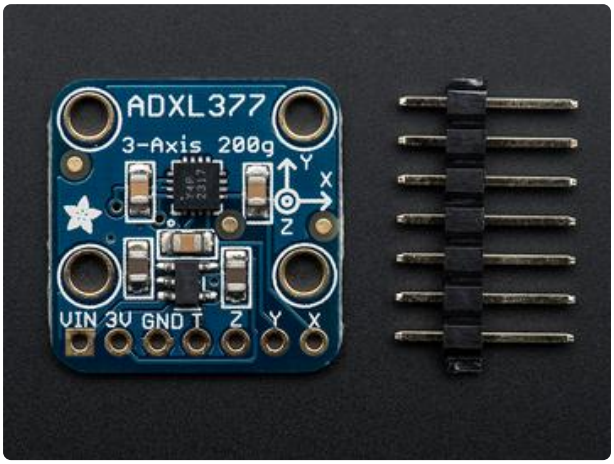
The ADXL335 , ADXL326 and ADXL377 are low-power, 3-axis MEMS accelerometer modules with ratiometric analog voltage outputs. The Adafruit Breakout boards for these modules feature on-board 3.3v voltage regulation which makes them simple to interface with 5v microcontrollers such as the Arduino.



The ADXL335 can measure at least +/- 3G in the X, Y and Z axis. It is perfect for high-resolution static acceleration measurements such as tilt-sensing, as well as for moderate dynamic accelerations from motion, shock or vibration.



The ADXL326 can measure at least +/- 16G(!) in the X, Y and Z axis. It is ideal for measuring more dynamic accelerations from high-performance land and air vehicles as well as for shock and impact measurements.



The ADXL377 can measure at least +/- 200G(!) in the X, Y and Z axis. This is the sensor for measuring extreme dynamic accelerations encountered in applications such as rocketry experiments and high-impact shock measurements.

How it Works:

MEMS - Micro Electro-Mechanical Systems

The sensor consists of a micro-machined structure on a silicon wafer. The structure is suspended by polysilicon springs which allow it to deflect in the when subject to acceleration in the X, Y and/or Z axis. Deflection causes a change in capacitance between fixed plates and plates attached to the suspended structure. This change in capacitance on each axis is converted to an output voltage proportional to the acceleration on that axis.

Ratiometric Output

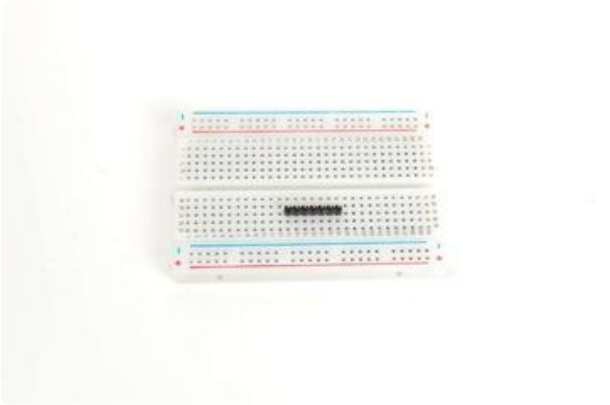
Ratiometric output means that the output voltage increases linearly with acceleration over the range.

- For the ADXL335, that is approximately 0v at -3G to 3.3v at +3G.
- For the ADXL326, that is approximately 0v at -16G to 3.3v at +16G.
- For the ADXL377, that is approximately 0v at -200G to 3.3v at +200G.
- For all modules, the output at 0G in each axis, is about 1/2 full-scale, or 1.65v.

Note that the specified device ranges are guaranteed minimum ranges. Most actual devices will have a somewhat wider usable range. Also, due to manufacturing variations the zero point may be slightly offset from exactly 1/2 scale. We will discuss how to calibrate the range and offset in the Calibration and Programming section of this guide.

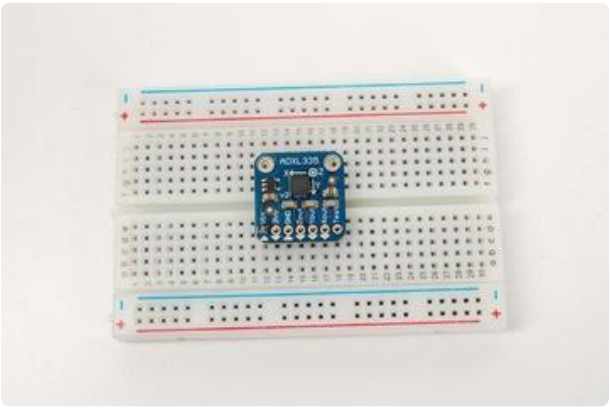
Assembly

These boards come with all surface-mount components pre-soldered. The included header strip can be soldered on for convenient use on a breadboard or with 0.1" connectors. However, for applications subject to extreme accelerations, shock or vibration, locking connectors or direct soldering plus strain relief is advised.



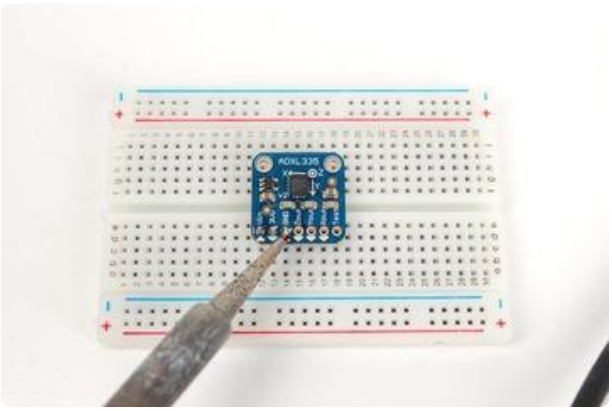
Prepare the header strip:

Cut the strip[to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down.



Add the breakout board:

Place the breakout board over the pins.



And Solder!

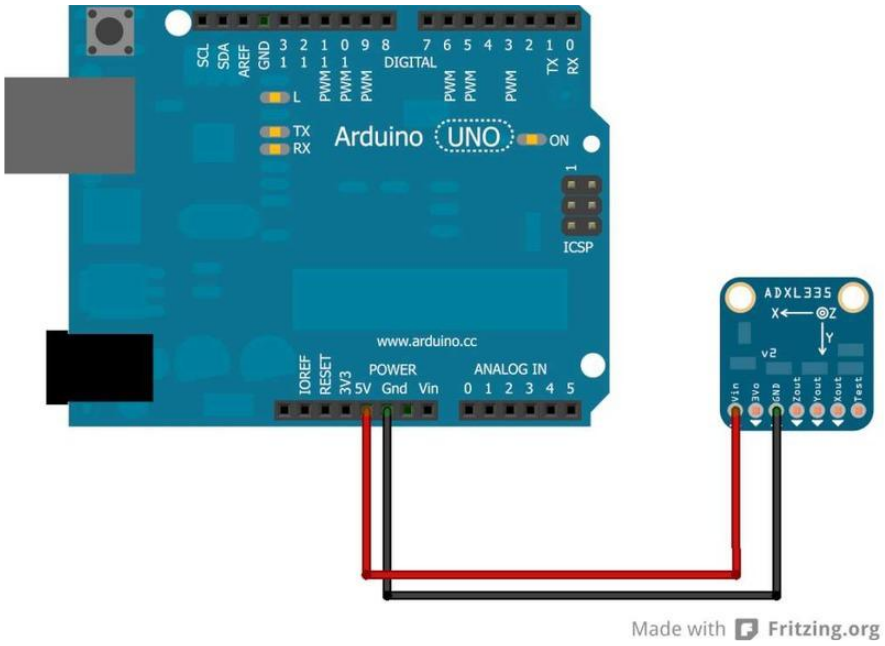
Be sure to solder all pins for reliable electrical contact.

Arduino Wiring

Wiring:

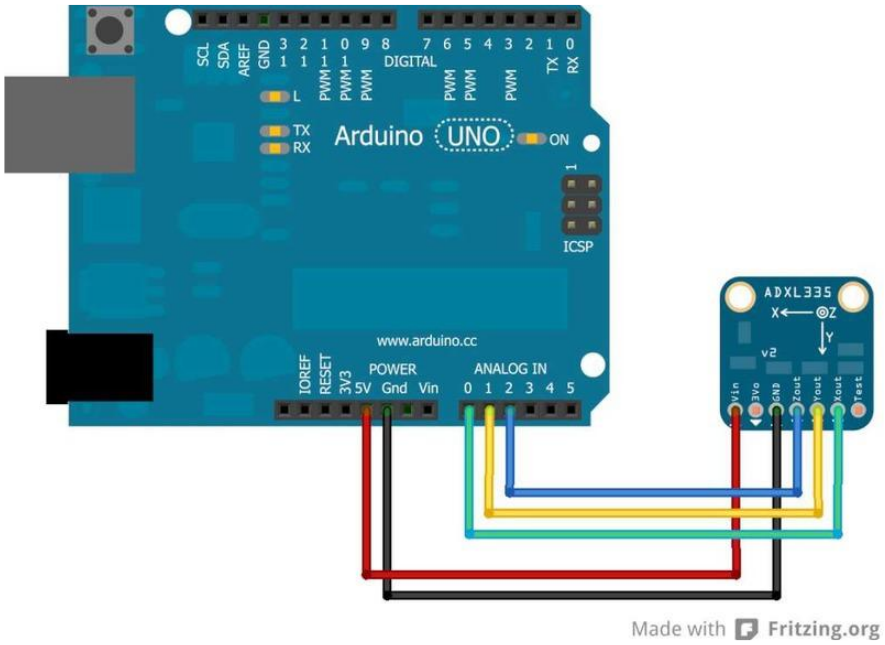
Connect the Power:

- Connect the GND pin to GND on the Arduino.
- Connect the VIN pin to the 5v pin on the Arduino.
- (For 3.3v microprocessors, connect the pin marked 3Vo to the 3.3v supply)



Connect the X, Y and Z Signal Outputs:

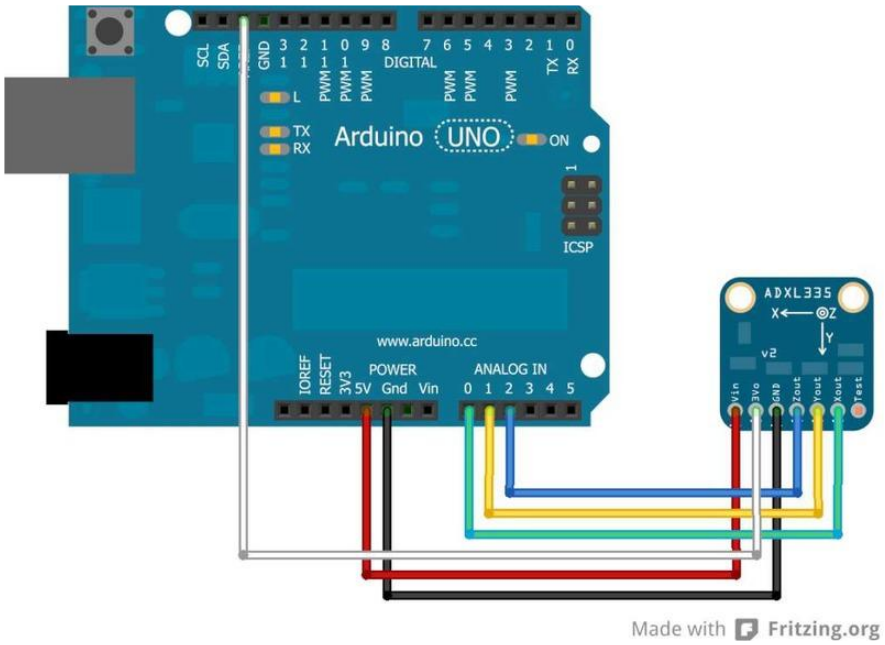
Connect X, Y and Z to the analog pins as shown below:



Using the Voltage Reference:

For the best possible accuracy and precision, you can use the output of the accelerometer boards voltage regulator as the analog reference for the Arduino. Connect the 3Vo pin on the accelerometer board to the AREF pin on the Arduino.

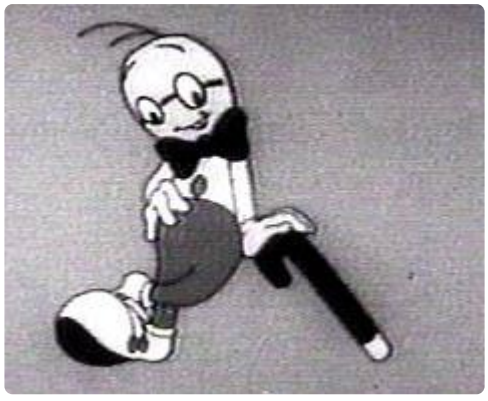
If you connect an external voltage reference to the AREF pin, you must set the analog reference to EXTERNAL before calling analogRead() (e.g. in your setup() function). Otherwise, you will short the internal reference with the external reference, possibly damaging your Arduino board.



Calibration and Programming

Static Calibration:

As with all sensors, there is some variation in output between samples of these accelerometers. For non-critical applications such as game controllers, or simple motion or tilt sensors, these variations are not important. But for applications requiring precise measurements, calibration to a reliable reference is a good idea.



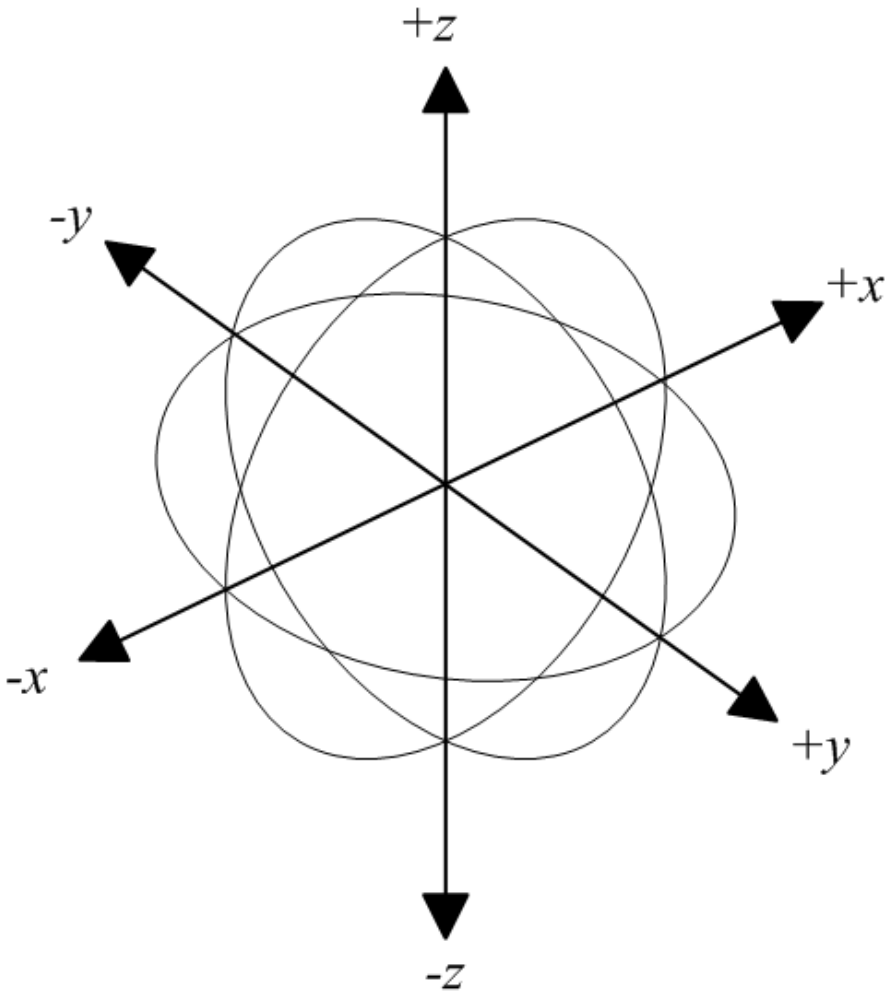
Gravity as a Calibration Reference

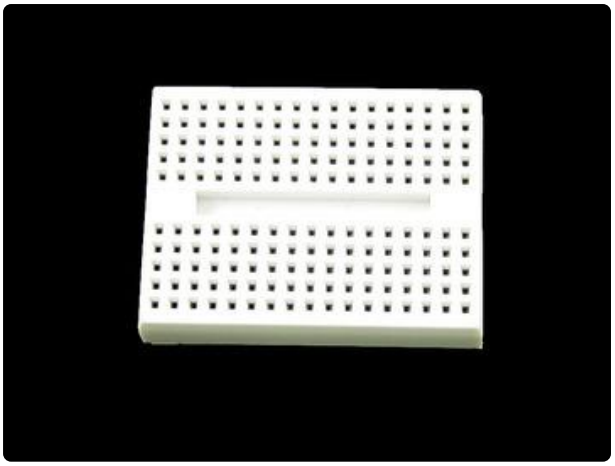
Acceleration is measured in units of gravitational force or "G", where 1G represents the gravitational pull at the surface of the earth. Despite what you may have heard (https://adafru.it/aRE), gravity is a pretty stable force and makes a convenient and reliable calibration reference if you happen to be a surface-dwelling earthling.

For High-G accelerometers such as the ADXL377, the +/- 1G range of static calibration is too small to assure good accuracy over the +/- 200G range of the sensor. Accurate calibration for extreme G-forces requires more specialized equipment to repeatably create these extreme forces in a controlled environment. One commonly used technique is to drop the accelerometer from a known height and measure the negative acceleration at impact.

Calibration Method:

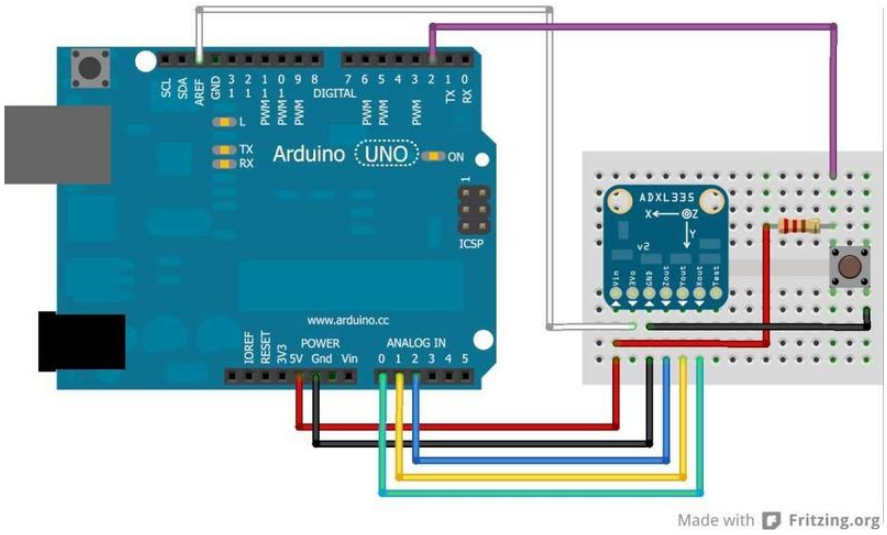
To calibrate the sensor to the gravitational reference, you need to determine the sensor output for each axis when it is precisely aligned with the axis of gravitational pull. Laboratory quality calibration uses precision positioning jigs. The method described here is simple and gives surprisingly good results.





Mount the Sensor:
First mount the sensor to a small breadboard (http://adafru.it/65) like the one on the left. The back and squared edges of the breadboard make a reasonably accurate set of reference planes to orient the sensor for calibration.

Wire the Sensor:
Wire the sensor as shown below. This is equivalent to the circuit shown on the previous page, with the addition of a switch.



Run the Calibration Sketch

- Load the sketch below onto the Arduino and run it.
- Open the Serial Monitor.
- Lay the breadboard with the sensor on a flat surface
    - Press and hold the button until you see "Calibrate" in the serial monitor.
    - This will calibrate the minimum value for the z axis.
- Stand the breadboard on the front edge and press the button again. to calibrate +y
- Repeat this for the three other edges to calibrate +x, -y and -x.
- Turn the breadboard upside down and press the button again to calibrate +z. (Hint, the underside of a table works well to steady it.)

Calibration Sketch Output:

Once calibrated, the output will show the calibrated raw range for each axis, followed by the measured "G" forces. The raw ranges can be used as constants in sketches.

```
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625
511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625
511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625
511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625
511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625
```

Calibration Sketch

```
const int xInput = A0;
const int yInput = A1;
const int zInput = A2;
const int buttonPin = 2;

// Raw Ranges:
// initialize to mid-range and allow calibration to
// find the minimum and maximum for each axis
int xRawMin = 512;
int xRawMax = 512;

int yRawMin = 512;
int yRawMax = 512;

int zRawMin = 512;
int zRawMax = 512;

// Take multiple samples to reduce noise
const int sampleSize = 10;

void setup()
{
  analogReference(EXTERNAL);
  Serial.begin(9600);
}

void loop()
{
  int xRaw = ReadAxis(xInput);
  int yRaw = ReadAxis(yInput);
  int zRaw = ReadAxis(zInput);

  if (digitalRead(buttonPin) == LOW)
  {
    AutoCalibrate(xRaw, yRaw, zRaw);
  }
  else
  {
    Serial.print("Raw Ranges: X: ");
    Serial.print(xRawMin);
    Serial.print("-");
    Serial.print(xRawMax);

    Serial.print(", Y: ");
    Serial.print(yRawMin);
    Serial.print("-");
    Serial.print(yRawMax);

    Serial.print(", Z: ");
    Serial.print(zRawMin);
    Serial.print("-");
    Serial.print(zRawMax);
    Serial.println();
    Serial.print(xRaw);
    Serial.print(", ");
    Serial.print(yRaw);
    Serial.print(", ");
    Serial.print(zRaw);

    // Convert raw values to 'milli-Gs"
    long xScaled = map(xRaw, xRawMin, xRawMax, -1000, 1000);
    long yScaled = map(yRaw, yRawMin, yRawMax, -1000, 1000);
    long zScaled = map(zRaw, zRawMin, zRawMax, -1000, 1000);
```

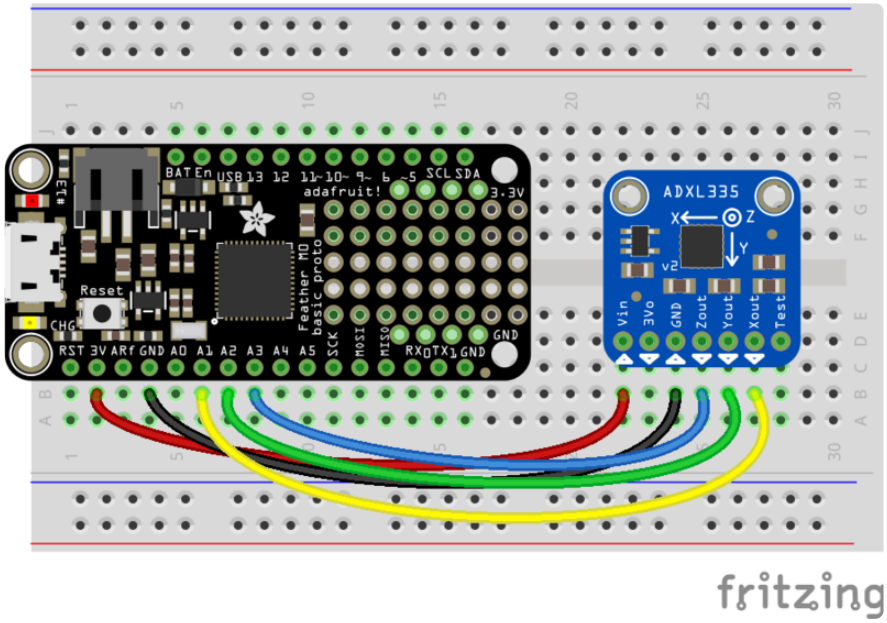
```
    // re-scale to fractional Gs
    float xAccel = xScaled / 1000.0;
    float yAccel = yScaled / 1000.0;
    float zAccel = zScaled / 1000.0;

    Serial.print(" :: ");
    Serial.print(xAccel);
    Serial.print("G, ");
    Serial.print(yAccel);
    Serial.print("G, ");
    Serial.print(zAccel);
    Serial.println("G");

  delay(500);
  }
}

//
// Read "sampleSize" samples and report the average
//
int ReadAxis(int axisPin)
{
  long reading = 0;
  analogRead(axisPin);
  delay(1);
  for (int i = 0; i &lt; sampleSize; i++)
  {
    reading += analogRead(axisPin);
  }
  return reading/sampleSize;
}

//
// Find the extreme raw readings from each axis
//
void AutoCalibrate(int xRaw, int yRaw, int zRaw)
{
  Serial.println("Calibrate");
  if (xRaw &lt; xRawMin)
  {
    xRawMin = xRaw;
  }
  if (xRaw &gt; xRawMax)
  {
    xRawMax = xRaw;
  }

  if (yRaw &lt; yRawMin)
  {
    yRawMin = yRaw;
  }
  if (yRaw &gt; yRawMax)
  {
    yRawMax = yRaw;
  }

  if (zRaw &lt; zRawMin)
  {
    zRawMin = zRaw;
  }
  if (zRaw &gt; zRawMax)
  {
    zRawMax = zRaw;
  }
}
```

CircuitPython Code

It's easy to use the analog accelerometer breakouts with CircuitPython and its built-in analog I/O module (https://adafru.it/BBj). You can read the X, Y, Z accelerometer axis values as simple analog inputs that are converted to acceleration values with simple Python code.

This page will show how to use the ADXL335 with a CircuitPython board like the Feather M0. Remember your CircuitPython board must support and have three analog inputs, so the ESP8266 and small boards like Gemma M0 won't work. Stick with a Feather or Metro M0.

Wire up the ADXL335 to your board as follows:



- Board 3V to sensor VIN
- Board GND to sensor GND
- Board A1 to sensor Xout
- Board A2 to sensor Yout
- Board A3 to sensor Zout

Remember you can use any available analog inputs for the X, Y, Z outputs. Be sure that these analog inputs are tolerant of up to 3.3 volt input (the Feather and Metro M0 powered by 3.3-5V as shown will be tolerant)!

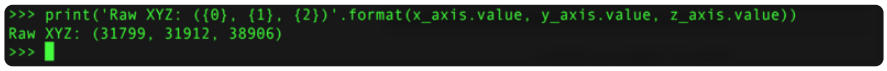
Next connect to the board's serial REPL (https://adafru.it/Awz)so you are at the CircuitPython >>> prompt.

If you haven't read it yet it will be helpful to read the analog I/O guide (https://adafru.it/BBj) as it explains the basics of reading analog inputs with the board's analog to digital converter. We will read the X, Y, Z axis values as analog inputs. First import the necessary board and analogio modules and create three inputs, one for each axis:

```
import board
import analogio
x_axis = analogio.AnalogIn(board.A1)
y_axis = analogio.AnalogIn(board.A2)
z_axis = analogio.AnalogIn(board.A3)
```

You can read the raw value of an axis with the value property:

```
print('Raw XYZ: ({0}, {1}, {2})'.format(x_axis.value, y_axis.value, z_axis.value))
```

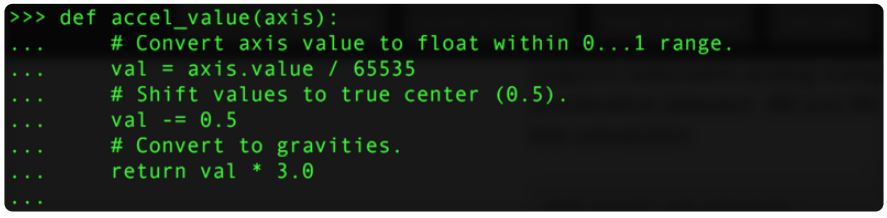


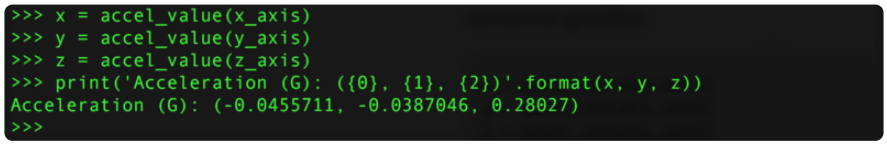
Raw values aren't that interesting though, you really want the acceleration. Luckily it's easy to convert raw values to acceleration with a simple formula. Remember these sensors output a radiometric analog voltage proportional to the acceleration between -3G and 3G. Let's define a function to do this conversion:

```
def accel_value(axis):
    # Convert axis value to float within 0...1 range.
    val = axis.value / 65535
    # Shift values to true center (0.5).
    val -= 0.5
    # Convert to gravities.
    return val * 3.0
```

Now try reading the axis values again using the function to convert to gravities:

```
x = accel_value(x_axis)
y = accel_value(y_axis)
z = accel_value(z_axis)
print('Acceleration (G): ({0}, {1}, {2})'.format(x, y, z))
```





That's all there is to using the ADXL335 with CircuitPython! You can use the built-in analog input capabilities to convert the accelerometer's X, Y, Z axis acceleration into a gravity value for your own use!

Here's a complete example of printing the X, Y, Z axis acceleration every second. Save this as main.py on your board and examine the REPL output:

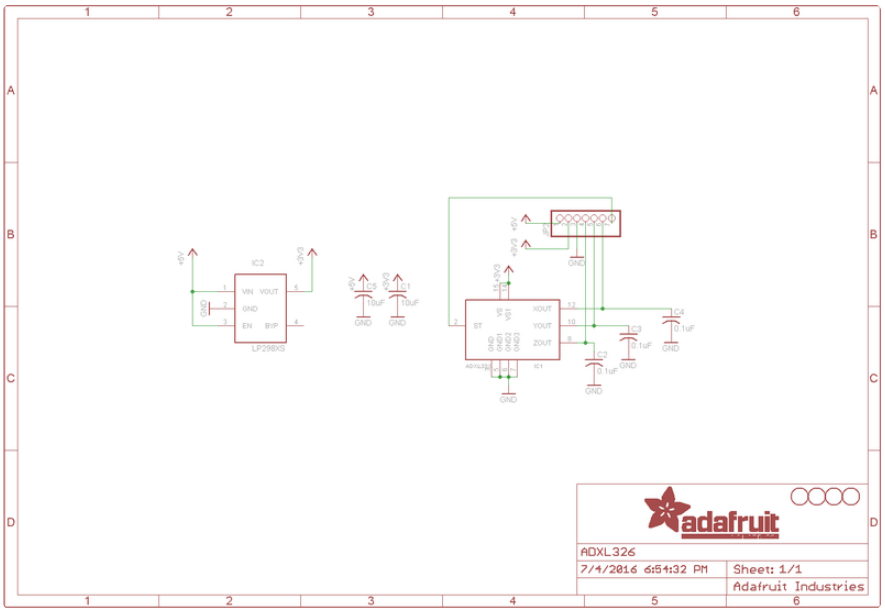
```
import analogio
import board
import time


# Create analog inputs for each ADXL335 axis.
x_axis = analogio.AnalogIn(board.A1)
y_axis = analogio.AnalogIn(board.A2)
z_axis = analogio.AnalogIn(board.A3)

# Define function to convert raw analog values to gravities.
def accel_value(axis):
    # Convert axis value to float within 0...1 range.
    val = axis.value / 65535
    # Shift values to true center (0.5).
    val -= 0.5
    # Convert to gravities.
    return val * 3.0

# Main loop prints acceleration every second.
while True:
    x = accel_value(x_axis)
    y = accel_value(y_axis)
    z = accel_value(z_axis)
    print('Acceleration (G): ({0}, {1}, {2})'.format(x, y, z))
    time.sleep(1.0)
```

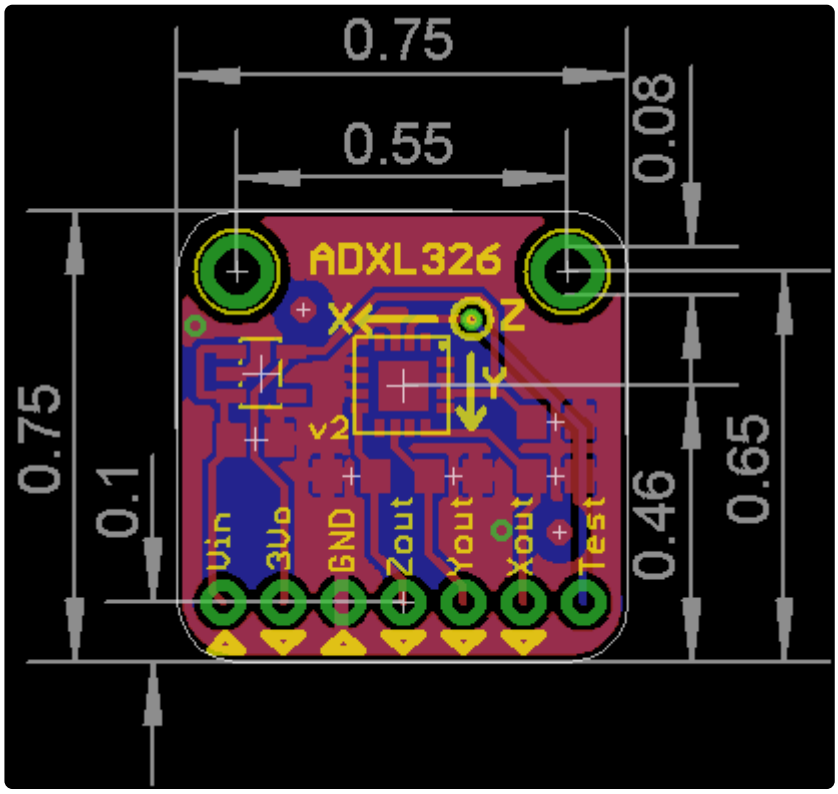
Downloads

Files

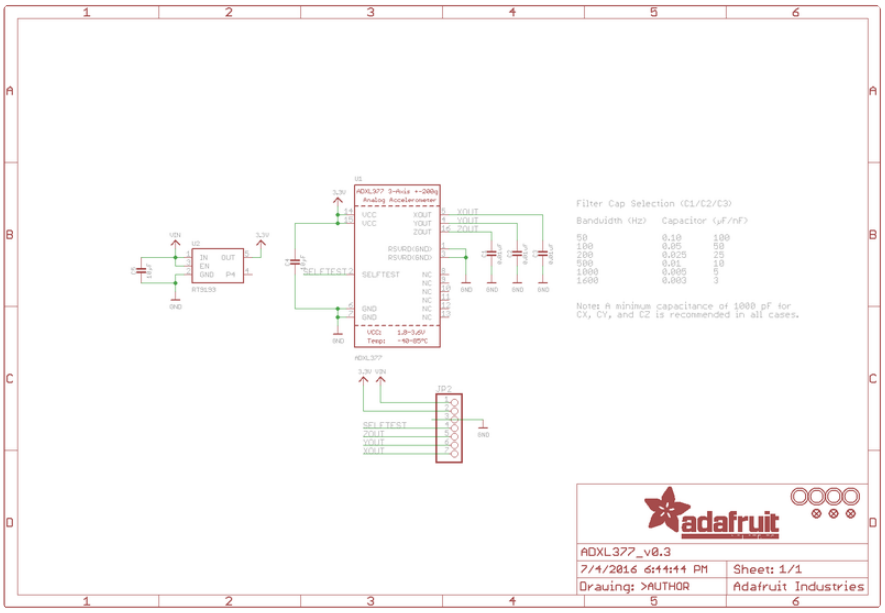
- ADXL335 Documentation Page (https://adafru.it/aRF)
- ADXL326 Documentation Page (https://adafru.it/nhe)
- ADXL377 Data Sheet (https://adafru.it/cLj)
- ADXL335, ADXL326 & ADXL377 Breakout Board Eagle Files (https://adafru.it/aRH)
- Fritzing objects in the Adafruit Fritzing Library (https://adafru.it/aP3)

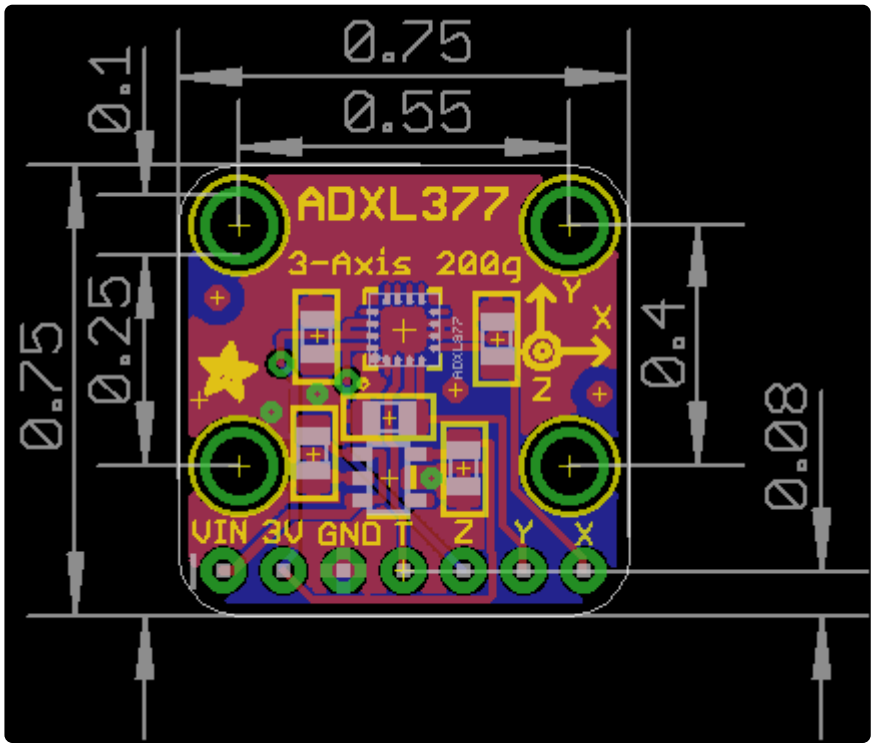
ADXL326 Schematic & Fabrication Print





ADXL377 Schematic & Fabrication Print





X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Acceleration Sensor Development Tools ***category:***

Click to view products by Adafruit ***manufacturer:***

Other Similar products are found below :

2019 EVAL-ADXL343Z-S BRKOUT-FXLN8362Q MXC6655XA-B 1018 EVAL-ADXL362-ARDZ 1231 1413 DEV-13629 2020 EVAL-ADXL343Z-DB EVAL-ADXL344Z-M EVAL-ADXL375Z-S EV-BUNCH-WSN-1Z EV-CLUSTER-WSN-2Z STEVAL-MKI033V1 EVAL-ADXL344Z-DB EVAL-ADXL346Z-DB EVAL-ADXL363Z-MLP EV-CLUSTER-WSN-1Z 2472 EVAL-ADXL312Z EVAL-ADXL343Z EVAL-ADXL344Z-S EVAL-ADXL363Z-S EVAL-ADXL375Z STEVALMKI032V1 DFR0143 SEN0032 SEN0079 SEN0168 SEN0224 MXA2500EL-B FIT0031 SEN-13963 MXP7205VW-B ASD2511-R-A 3463 SEN0140 SEN0183 SEN-11446 EVAL-KX022-1020 EVAL-KX023-1025 163 2809 4097 4344 4627 4626 ADIS16201/PCBZ