ADXL345 Digital Accelerometer
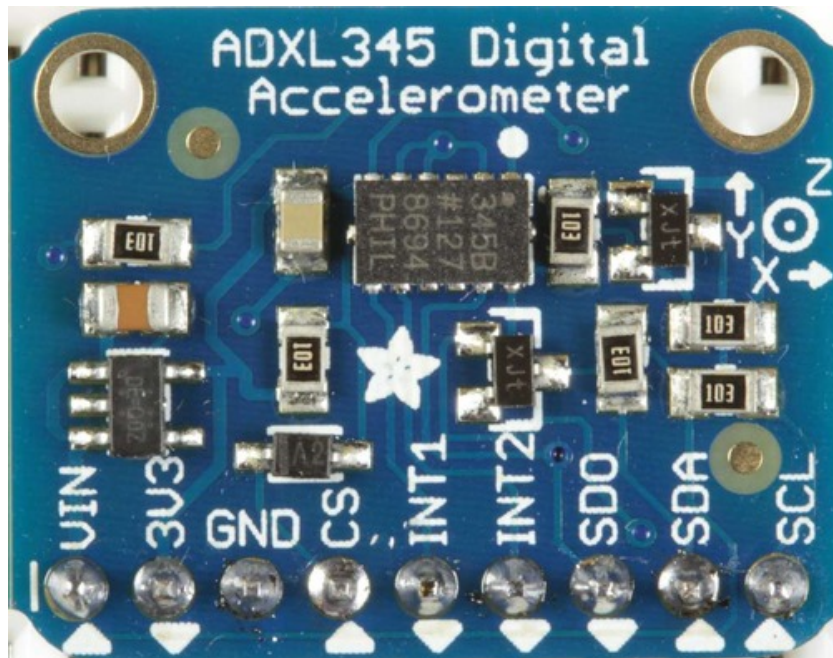
Created by William Earl
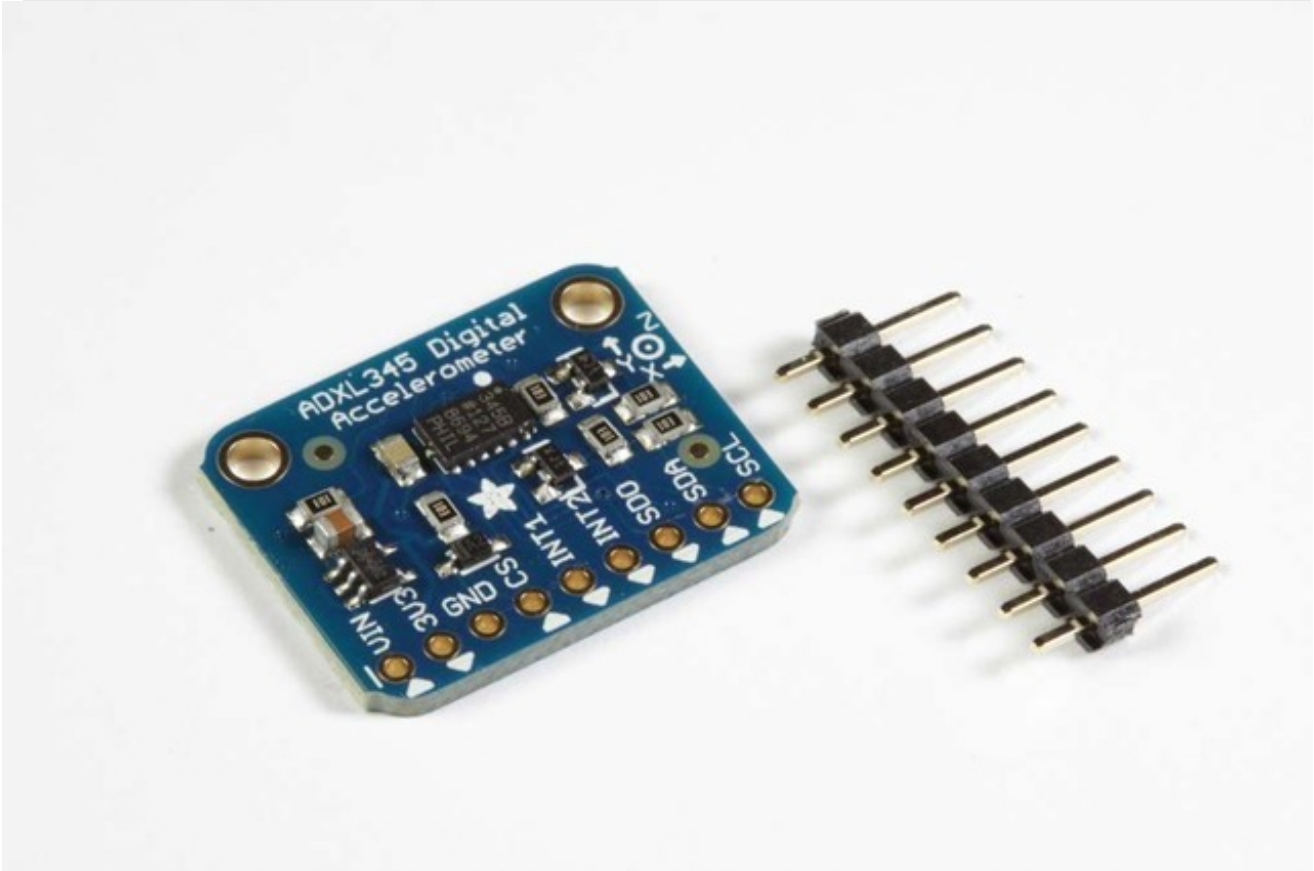

Last updated on 2014-05-07 03:00:09 PM EDT

Guide Contents

Overview

The ADXL345 is a low-power, 3-axis MEMS accelerometer modules with both I2C and SPI interfaces. The Adafruit Breakout boards for these modules feature on-board 3.3v voltage regulation and level shifting which makes them simple to interface with 5v microcontrollers such as the Arduino.

The ADXL345 features 4 sensitivity ranges from +/- 2G to +/- 16G. And it supports output data rates ranging from 10Hz to 3200Hz.

ADXL345 datasheet (http://adafru.it/c5e)

How it Works:

(http://adafru.it/c5f)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 smoothly in any direction 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.
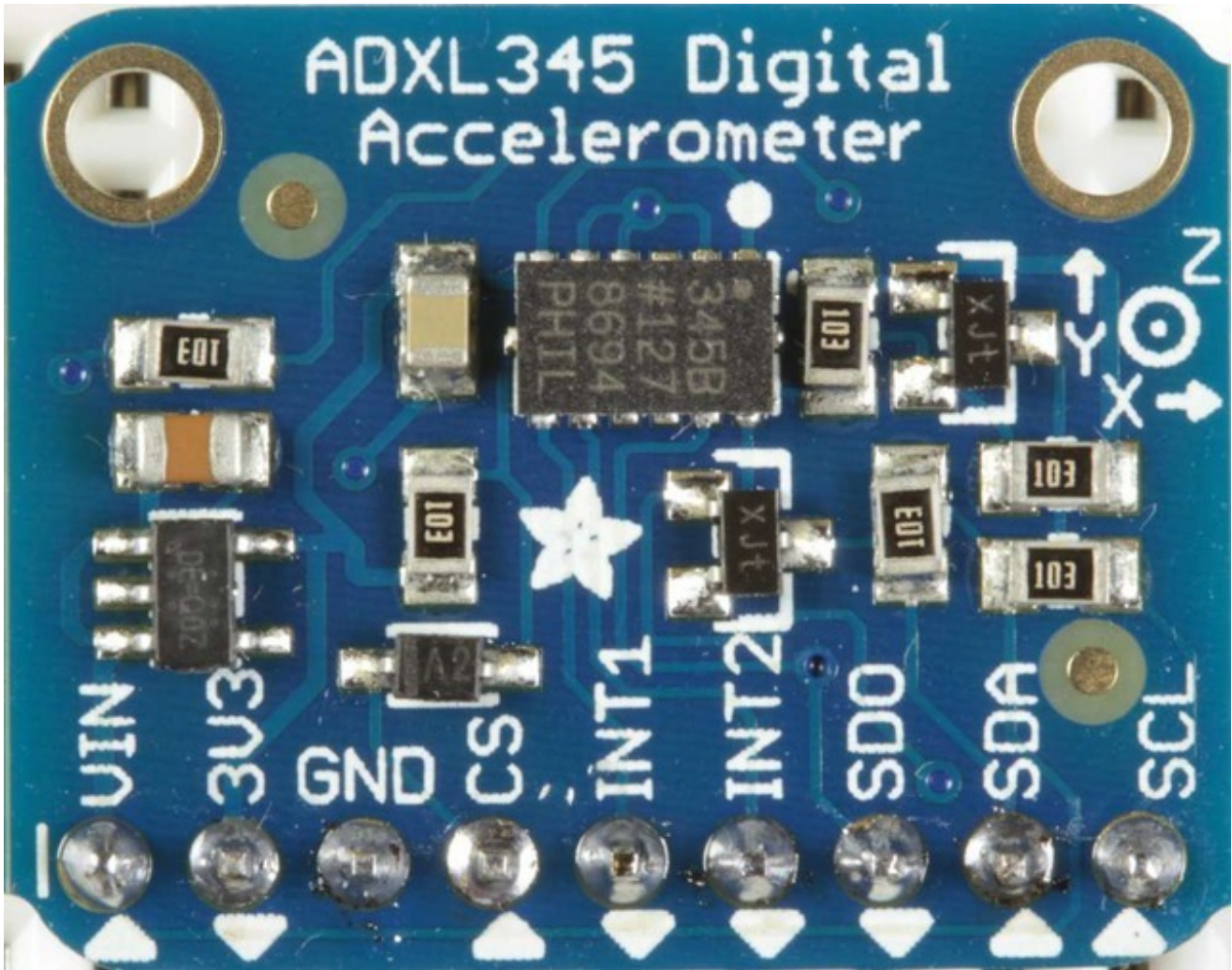
Assembly and Wiring
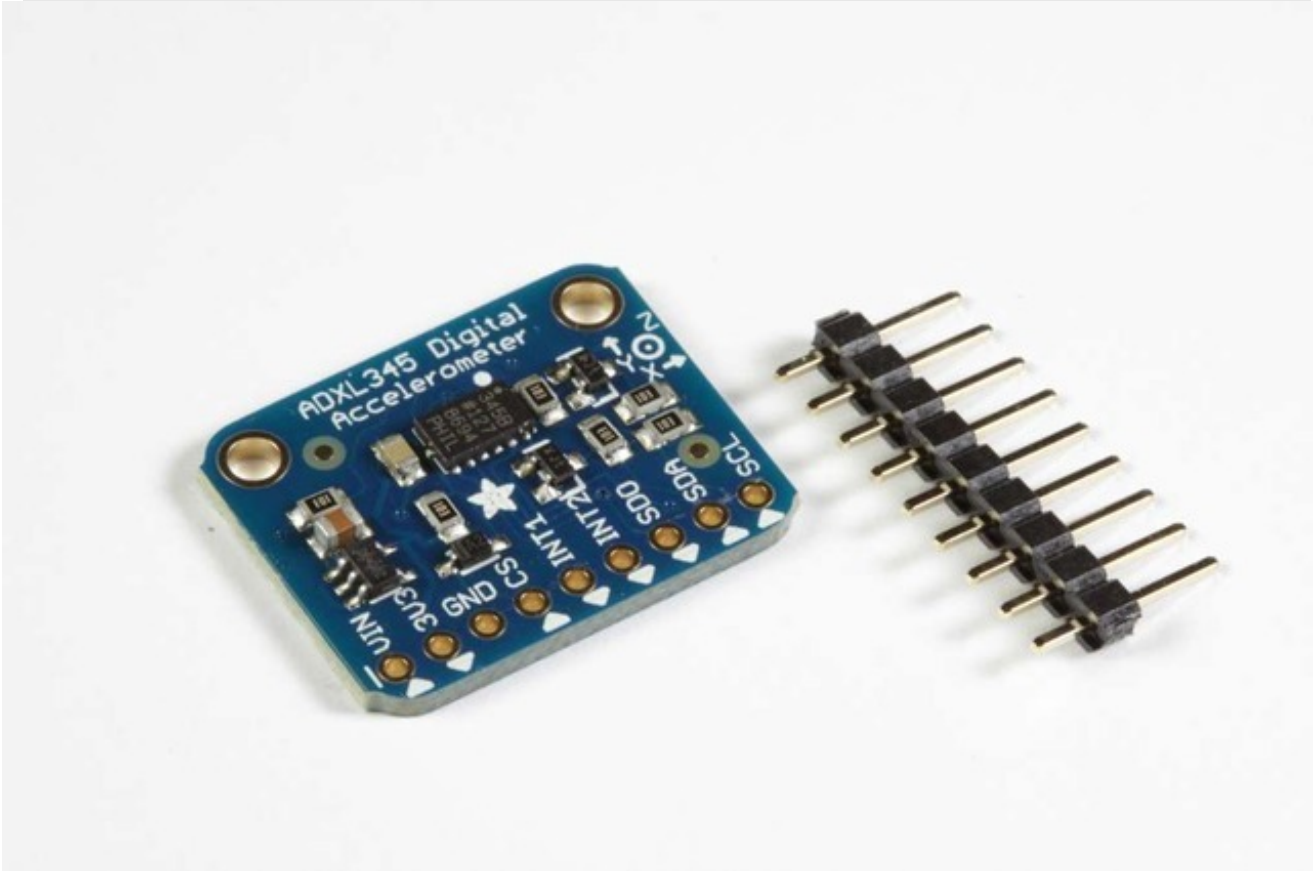
The board comes with all surface-mount components pre-soldered. The included header strip can be soldered on for convenient use on a breadboard or with 0.1" connectors. However, for applications subject to extreme accelerations, shock or vibration, locking connectors or direct soldering is advised.

Assembly:
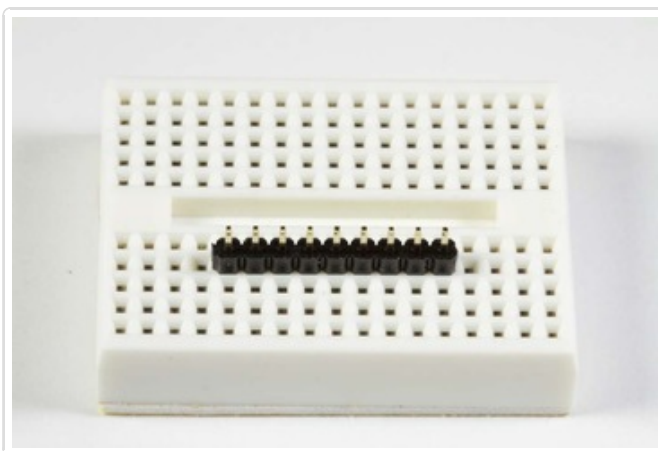
Position the Header:
Cut the header to size if necessary. Then plug the header - long pins down - into a breadboard to stabilize it for soldering.
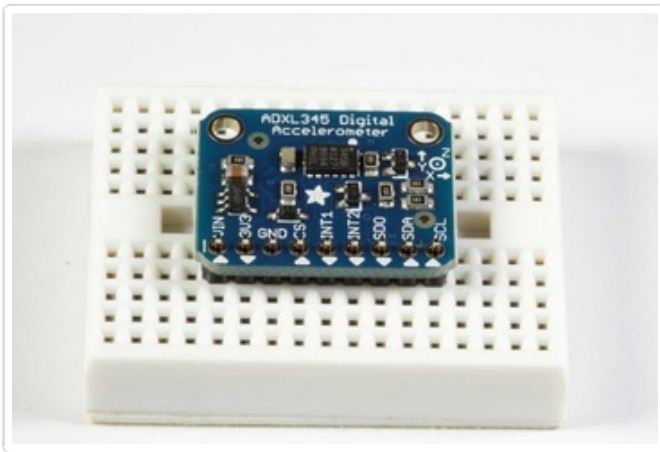
Add the Breakout:

Align the breakout board and place it over the header pins on the breadboard.
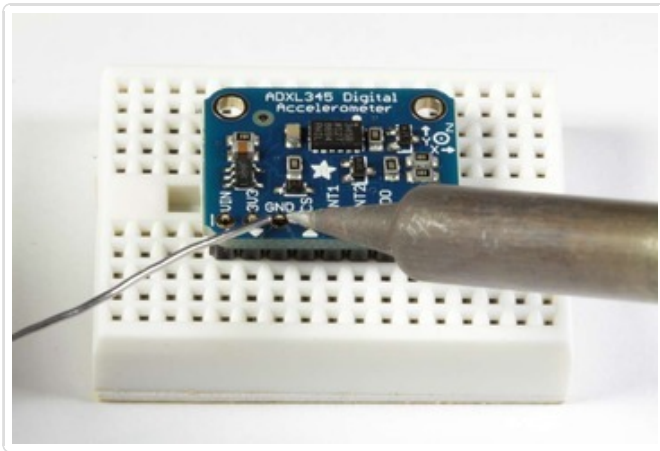
And Solder!

Be sure to solder all pins to assure good electrical contact.

I2C Wiring:

The ADXL345 Breakout has an I2C address of 0x53. It can share the I2C bus with other I2C devices as long as each device has a unique address. Only 4 connections are required for I2C communication:

- GND->GND
- VIN->+5v
- SDA->SDA (Analog 4 on "Classic Arduinos")
- SCL->SCL (Analog 5 on "Classic Arduinos")

The Adafruit breakout has level shifting and regulation circuitry so you can power it from 3-5V and use 3V or 5V logic levels for i2c
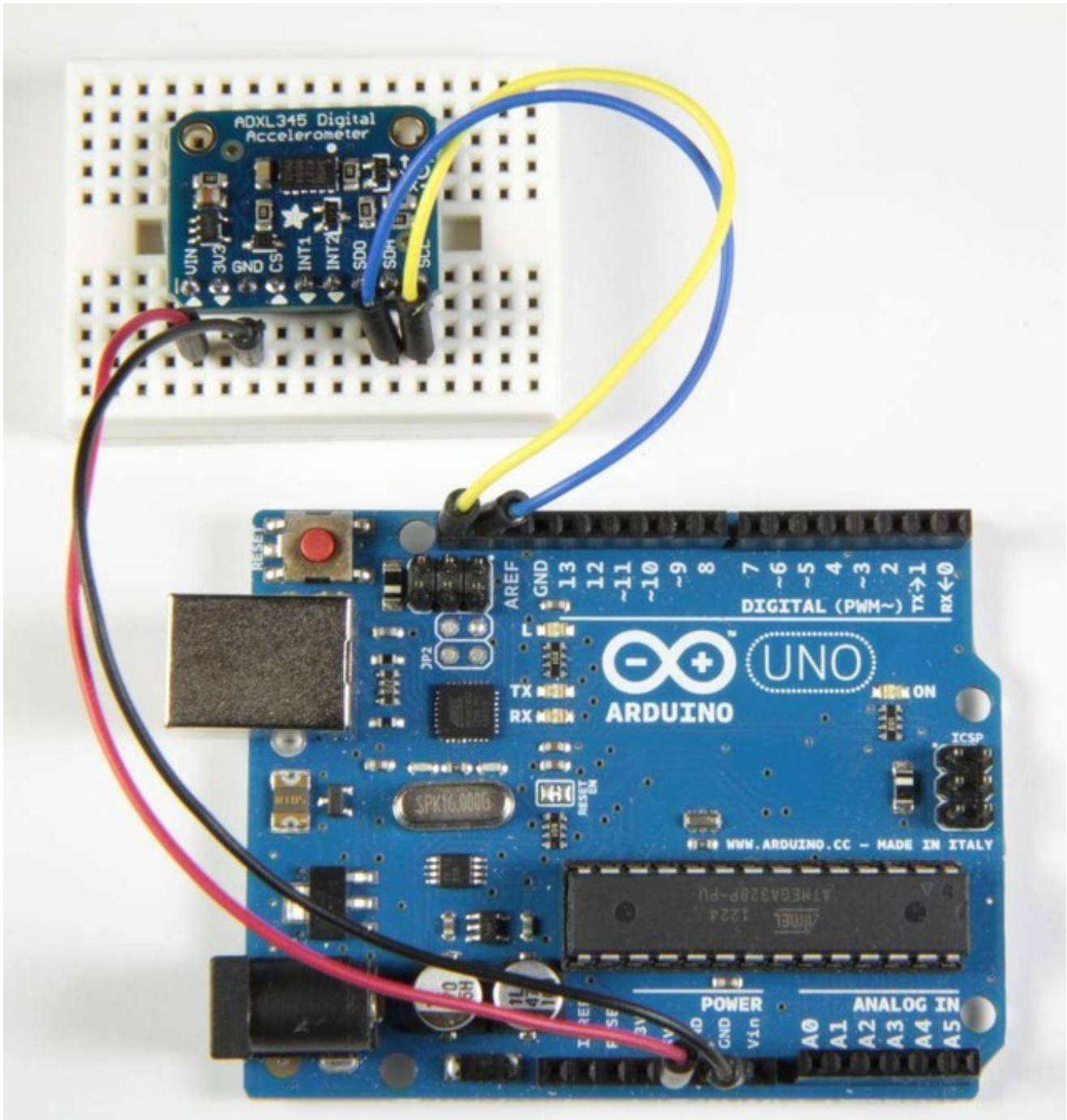
Programming and Calibration
Install the Library:

Download the ADXL345 library (http://adafru.it/aZn) and install it. You will also need the Adafruit Sensor Library (http://adafru.it/aZm) if you do not already have it installed.

This guide (http://adafru.it/aYM) will help you with the install process.

Test:

Click "File->Examples->Adafruit_ADXL345->sensortest" to load the example sketch from the library.

Then click on the compile/upload button to compile and upload the sketch to the Arduino. You should see output similar to below. Watch the values change as you move the board around.

```
COM26                                                          ▁ ▢ ✕

                                                                  Send

Accelerometer Test                                                    ▲

---------------------------------------
Sensor:        ADXL345
Driver Ver:    1
Unique ID:     12345
Max Value:     300.00 m/s^2
Min Value:     1100.00 m/s^2
Resolution:    0.01 m/s^2
---------------------------------------

Data Rate:     400  Hz
Range:           +/- 16  g

X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.14  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.14  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.14  m/s^2
X: 0.08  Y: -0.12  Z: 1.14  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.14  m/s^2
X: 0.08  Y: -0.16  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2
X: 0.08  Y: -0.12  Z: 1.10  m/s^2              ▼

☐ Autoscroll                    Newline  ▼    9600 baud  ▼
```
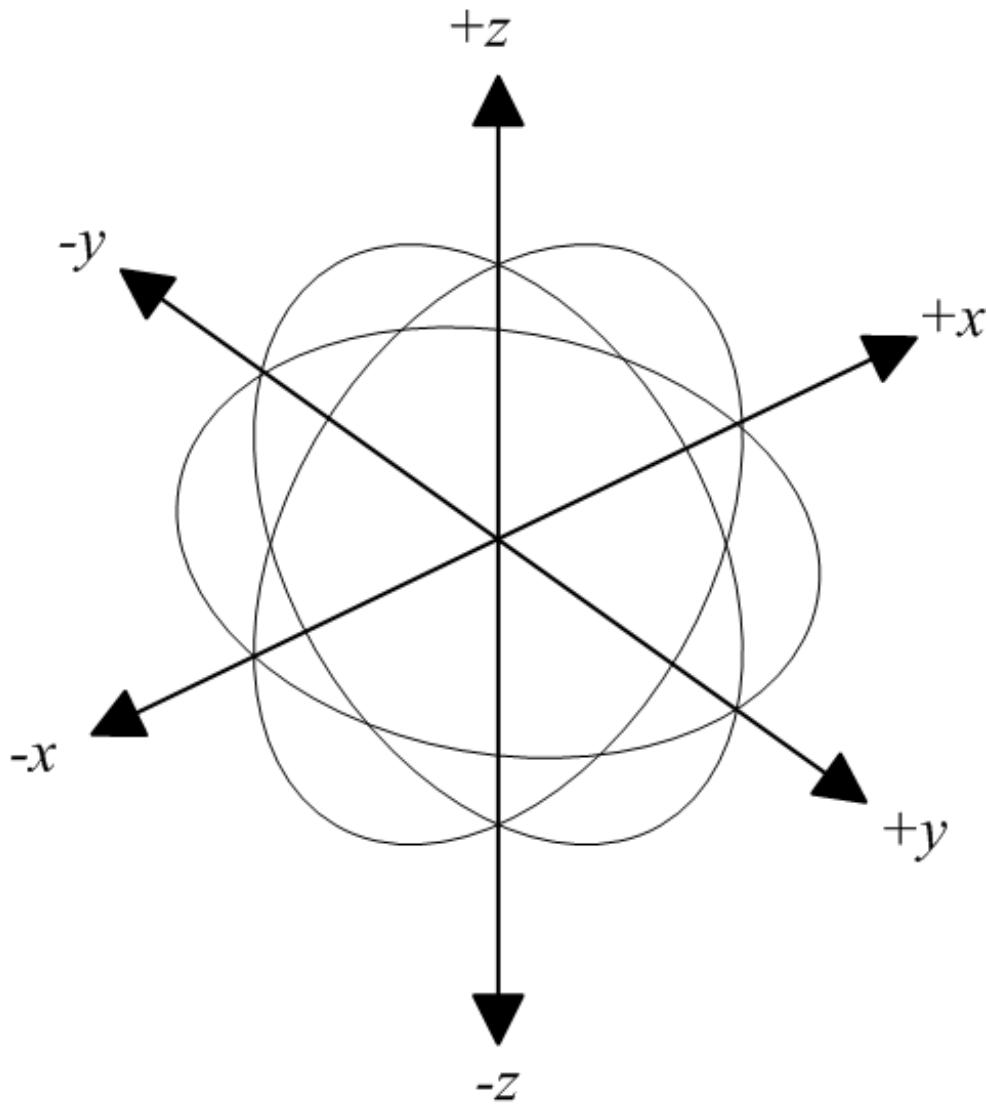
Calibrate:

The ADXL chips are calibrated at the factory to a level of precision sufficient for most purposes. For critical applications where a higher degree of accuracy is required, you may wish to re-calibrate the sensor yourself.

Calibration does not change the sensor outputs. But it tells you what the sensor output is for a known stable reference force in both directions on each axis. Knowing that, you can calculate the corrected output from a sensor reading.

Gravity as a Calibration Reference

Acceleration can be measured in units of gravitational force or "G", where 1G represents the gravitational pull at the surface of the earth. Gravity is a relatively stable force and makes a convenient and reliable calibration reference for surface-dwelling earthlings.

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 with just a block of wood.
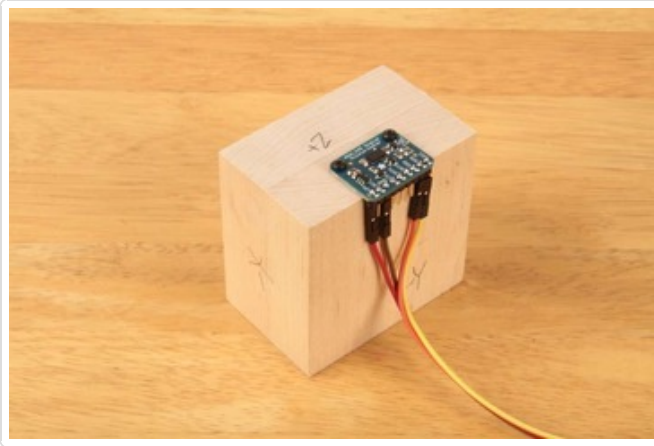
Mount the Sensor:

FIrst mount the sensor securely to a block or a box. The size is not important, as long as all

the sides are at right angles. The material is not important as long as it is fairly rigid.
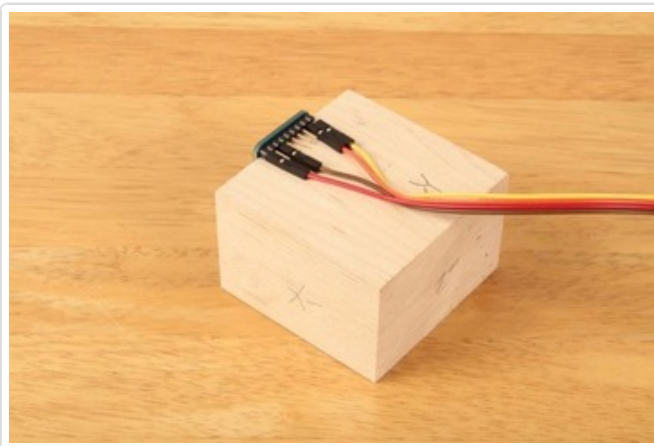
Load the Calibration Sketch:
Load and run the Calibration sketch below. Open the Serial Monitor and wait for the prompt.

Position the Block:
Place the block on a firm flat surface such as a sturdy table. Type a character in the Serial Monitor and hit return. The sketch will take a measurement on that axis and print the results.

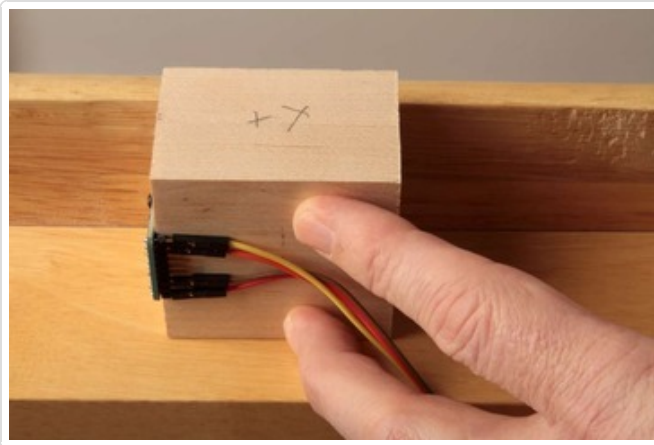## Reposition the Block:
Turn the block so a different side is flat on the table and type another key to measure that axis.

(http://adafru.it/c5g)
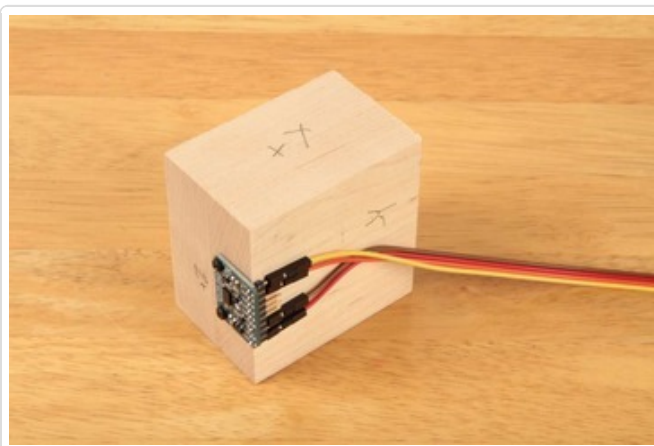
Repeat:
Repeat for all six sides of the block to measure the positive and negative aspects of each axis.

(Hint:)
For the sides obstructed by the breakout board and/or wires, press the block up against the bottom of the table while taking the reading.

Calibration Results:

Once all six sides have been sampled, the values printed in the Serial Monitor will represent actual measurements for +/- 1G forces on each axis. These values can be used to re-scale readings for better accuracy.

Calibration Sketch:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

/* Assign a unique ID to this sensor at the same time */
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);


float AccelMinX = 0;
float AccelMaxX = 0;
float AccelMinY = 0;
float AccelMaxY = 0;
float AccelMinZ = 0;
float AccelMaxZ = 0;


void setup(void)
{
  Serial.begin(9600);
  Serial.println("ADXL345 Accelerometer Calibration");
  Serial.println("");
```

```
/* Initialise the sensor */
if(!accel.begin())
{
  /* There was a problem detecting the ADXL345 ... check your connections */
  Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");
  while(1);
}
}

void loop(void)
{
  Serial.println("Type key when ready...");
  while (!Serial.available()){}  // wait for a character

  /* Get a new sensor event */
  sensors_event_t accelEvent;
  accel.getEvent(&accelEvent);

  if (accelEvent.acceleration.x < AccelMinX) AccelMinX = accelEvent.acceleration.x;
  if (accelEvent.acceleration.x > AccelMaxX) AccelMaxX = accelEvent.acceleration.x;

  if (accelEvent.acceleration.y < AccelMinY) AccelMinY = accelEvent.acceleration.y;
  if (accelEvent.acceleration.y > AccelMaxY) AccelMaxY = accelEvent.acceleration.y;

  if (accelEvent.acceleration.z < AccelMinZ) AccelMinZ = accelEvent.acceleration.z;
  if (accelEvent.acceleration.z > AccelMaxZ) AccelMaxZ = accelEvent.acceleration.z;

  Serial.print("Accel Minimums: "); Serial.print(AccelMinX); Serial.print("  ");Serial.print(AccelMinY); Seri
  Serial.print("Accel Maximums: "); Serial.print(AccelMaxX); Serial.print("  ");Serial.print(AccelMaxY); S

  while (Serial.available())
  {
    Serial.read();  // clear the input buffer
  }
}
```

Typical Calibration Output:

```
ADXL345 Accelerometer Calibration

Type key when ready...
Accel Minimums: 0.00  0.00  0.00
Accel Maximums: 0.12  0.20  1.14
```

```
Type key when ready…
Accel Minimums: 0.00  0.00  0.00
Accel Maximums: 0.12  0.20  1.14
Type key when ready…
Accel Minimums: 0.00  0.00  0.00
Accel Maximums: 0.12  0.20  1.14
Type key when ready…
Accel Minimums: 0.00  0.00  0.00
Accel Maximums: 0.12  0.20  1.14
Type key when ready…
Accel Minimums: 0.00  0.00  -0.24
Accel Maximums: 0.12  1.37  1.14
Type key when ready…
Accel Minimums: 0.00  0.00  -0.24
Accel Maximums: 0.12  1.37  1.14
Type key when ready…
Accel Minimums: 0.00  -1.22  -0.27
Accel Maximums: 0.12  1.37  1.14
Type key when ready…
Accel Minimums: 0.00  -1.22  -0.27
Accel Maximums: 0.12  1.37  1.14
Type key when ready…
Accel Minimums: -1.18  -1.22  -0.27
Accel Maximums: 0.12  1.37  1.14
Type key when ready…
```

Library Reference
Constructor:

Adafruit_ADXL345(int32_t sensorID = -1)

Constructs an instance of the ADXL345 device driver object. 'sensorID' is a device identifier. It will be returned in the sensor_event in each call to getEvent(). The sensorID has no effect on the operation of the driver or device, but is useful in managing sensor events in systems with multiple sensors.

Initialization()

bool begin(void)

The begin() function initializes communication with the device. The return value is 'true' if it succeeds in connecting to the ADXL345.

Sensor Details:

void getSensor(sensor_t*);

The getSensor() function returns basic information about the sensor. For details about the sensor_t structure, refer to the ReadMe file (http://adafru.it/aZm) for the Adafruit Sensor Library.

Getting and Setting the operating range:

void setRange(range_t range)

The setRange() function sets the operating range for the sensor. Higher values will have a wider measurement range. Lower values will have more sensitivity.

Valid range constants are:

- **ADXL345_RANGE_16_G**
- **ADXL345_RANGE_8_G**
- **ADXL345_RANGE_4_G**
- **ADXL345_RANGE_2_G** (default value)

range_t getRange(void);

The getRange() function returns the current operating range as set by setRange()

Getting and Setting the Data Rate:

void setDataRate(dataRate_t dataRate);

The setDataRate() function sets the rate at which the sensor output is updated. Rates above 100 Hz will exhibit increased noise. Rates below 6.25 Hz will be more sensitive to temperature variations. See the data sheet (http://adafru.it/c5e) for details.

Valid data rate constants are:

- **ADXL345_DATARATE_3200_HZ**
- **ADXL345_DATARATE_1600_HZ**
- **ADXL345_DATARATE_800_HZ**
- **ADXL345_DATARATE_400_HZ**
- **ADXL345_DATARATE_200_HZ**
- **ADXL345_DATARATE_100_HZ**
- **ADXL345_DATARATE_50_HZ**
- **ADXL345_DATARATE_25_HZ**
- **ADXL345_DATARATE_12_5_HZ**
- **ADXL345_DATARATE_6_25HZ**
- **ADXL345_DATARATE_3_13_HZ**
- **ADXL345_DATARATE_1_56_HZ**
- **ADXL345_DATARATE_0_78_HZ**
- **ADXL345_DATARATE_0_39_HZ**
- **ADXL345_DATARATE_0_20_HZ**
- **ADXL345_DATARATE_0_10_HZ** (default value)

dataRate_t getDataRate(void);

The getDataRate() function returns the current data rate as set by setDataRate().

Reading Sensor Events:

void getEvent(sensors_event_t*);

The getEvent() function returns the next available reading in the form of a sensor_event. The sensor_event contains the sensor_id as passed to the constructor as well as the X, Y and Z axis readings from the accelerometer. For more information about sensor_events, see the ReadMe file (http://adafru.it/aZm) for the Adafruit Sensor Library.