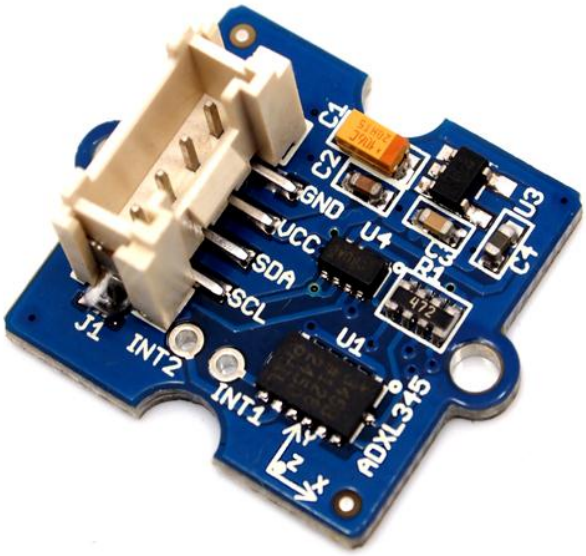
RB-See-224

SeeedStudio Grove ±16g Triple Axis Accelerometer (ADXL345)



Based on the excellent ADXL-345, this digital 3-axis accelerometer has excellent EMI protection. Its variable output makes it suitable for a wide range of applications:

- HDD shock protection
- Vibration sensor
- Game controller input
- Robotics
- Smart vehicles
- Anywhere you need to obtain motion-sensing & orientation information.

The excellent sensitivity (3.9mg/LSB @2g) provids high-precision output up to ±16g.
The sensor is utilizes a 4-pin Grove interface (3-5VDC) for easy connectivity with your standard Arduino device or Seeed Stalker.

Features

- Wide power range DC3V to 5V
- Grove module
- 3 axis sensing
- I2C digital interface
- High sensitivity
- Small, low-profile package: 14-Terminal LGA
- Low power 0.1 μA in standby mode at VS = 2.5 V (typical)
- 10,000 g shock survival
- RoHS/WEEE lead-free compliant

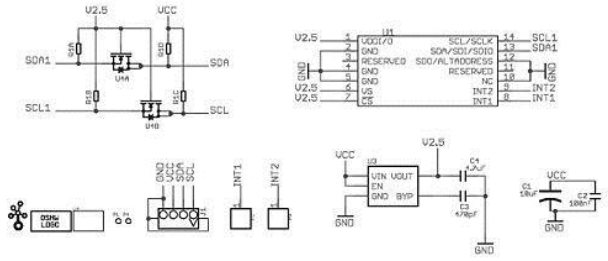
Application Ideas

- Motion Sensor
- Shock detector
- Vibration sensor
- Toy car
- Robotics
- Disk drive Shock Protection
- Game Controller

Cautions

Notice: Before you got this sensor, we have tested it to ensure that the sensor is working right. Sometimes there might be a small offset of the sensor, but it is within the acceptable error range. You can set the offset register according to the chip's datasheet to regulate the sensor.
Warning: wrong operation can cause danger.

Schematic

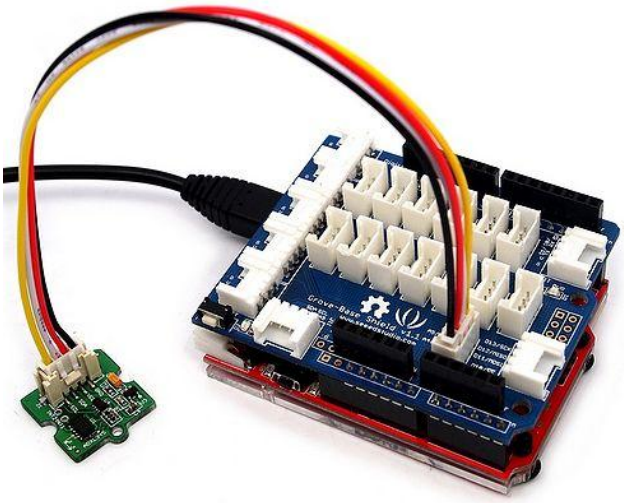


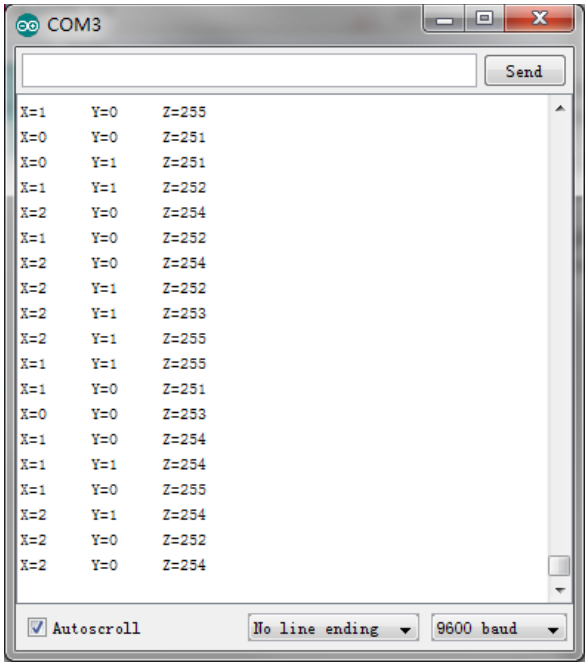
Usage

The sensor communicates with the MCU via I^2C interface. Connected the sensor to the I^2C port via grove 4 pin wires if you are using the Grove Base Shield.

For the first use you should download the ADXL345 Driver for Arduino.zip and unpack into arduino-1.0\libraries in your Arduino installation folder.

If you are using jumper wire to connect the sensor, make sure that the sensor's SDA pin is connect to the analog port A4 and the SCL pin is connect to A5. This two pins are used as the I^2C interface at your Arduino.





Programming

```
//Arduino 1.0+ Only!
//Arduino 1.0+ Only!

#include <Wire.h>
#include <ADXL345.h>


ADXL345 adxl; //variable adxl is an instance of the ADXL345 library

void setup(){
  Serial.begin(9600);
  adxl.powerOn();

  //set activity/ inactivity thresholds (0-255)
  adxl.setActivityThreshold(75); //62.5mg per increment
  adxl.setInactivityThreshold(75); //62.5mg per increment
  adxl.setTimeInactivity(10); // how many seconds of no activity is inactive?

  //look of activity movement on this axes - 1 == on; 0 == off
  adxl.setActivityX(1);
  adxl.setActivityY(1);
  adxl.setActivityZ(1);

  //look of inactivity movement on this axes - 1 == on; 0 == off
  adxl.setInactivityX(1);
  adxl.setInactivityY(1);
  adxl.setInactivityZ(1);
```

```
  //look of tap movement on this axes - 1 == on; 0 == off
  adxl.setTapDetectionOnX(0);
  adxl.setTapDetectionOnY(0);
  adxl.setTapDetectionOnZ(1);

  //set values for what is a tap, and what is a double tap (0-255)
  adxl.setTapThreshold(50); //62.5mg per increment
  adxl.setTapDuration(15); //625µs per increment
  adxl.setDoubleTapLatency(80); //1.25ms per increment
  adxl.setDoubleTapWindow(200); //1.25ms per increment

  //set values for what is considered freefall (0-255)
  adxl.setFreeFallThreshold(7); //(5 - 9) recommended - 62.5mg per increment
  adxl.setFreeFallDuration(45); //(20 - 70) recommended - 5ms per increment

  //setting all interupts to take place on int pin 1
  //I had issues with int pin 2, was unable to reset it
  adxl.setInterruptMapping( ADXL345_INT_SINGLE_TAP_BIT,   ADXL345_INT1_PIN );
  adxl.setInterruptMapping( ADXL345_INT_DOUBLE_TAP_BIT,   ADXL345_INT1_PIN );
  adxl.setInterruptMapping( ADXL345_INT_FREE_FALL_BIT,    ADXL345_INT1_PIN );
  adxl.setInterruptMapping( ADXL345_INT_ACTIVITY_BIT,     ADXL345_INT1_PIN );
  adxl.setInterruptMapping( ADXL345_INT_INACTIVITY_BIT,   ADXL345_INT1_PIN );

  //register interupt actions - 1 == on; 0 == off
  adxl.setInterrupt( ADXL345_INT_SINGLE_TAP_BIT, 1);
  adxl.setInterrupt( ADXL345_INT_DOUBLE_TAP_BIT, 1);
  adxl.setInterrupt( ADXL345_INT_FREE_FALL_BIT,  1);
  adxl.setInterrupt( ADXL345_INT_ACTIVITY_BIT,   1);
  adxl.setInterrupt( ADXL345_INT_INACTIVITY_BIT, 1);
}

void loop(){

  //Boring accelerometer stuff
  int x,y,z;
  adxl.readAccel(&x, &y, &z); //read the accelerometer values and store them in variables  x,y,z

  // Output x,y,z values - Commented out
  Serial.print(x);
  Serial.print('\t');
  Serial.print(y);
  Serial.print('\t');
  Serial.println(z);
  delay(500);
/*

  //Fun Stuff!
  //read interrupts source and look for triggerd actions
```

```
  //getInterruptSource clears all triggered actions after returning value
  //so do not call again until you need to recheck for triggered actions
   byte interrupts = adxl.getInterruptSource();

  // freefall
  if(adxl.triggered(interrupts, ADXL345_FREE_FALL)){
    Serial.println("freefall");
    //add code here to do when freefall is sensed
  }

  //inactivity
  if(adxl.triggered(interrupts, ADXL345_INACTIVITY)){
    Serial.println("inactivity");
     //add code here to do when inactivity is sensed
  }

  //activity
  if(adxl.triggered(interrupts, ADXL345_ACTIVITY)){
    Serial.println("activity");
     //add code here to do when activity is sensed
  }

  //double tap
  if(adxl.triggered(interrupts, ADXL345_DOUBLE_TAP)){
    Serial.println("double tap");
     //add code here to do when a 2X tap is sensed
  }

  //tap
  if(adxl.triggered(interrupts, ADXL345_SINGLE_TAP)){
    Serial.println("tap");
     //add code here to do when a tap is sensed
  } */

}
```