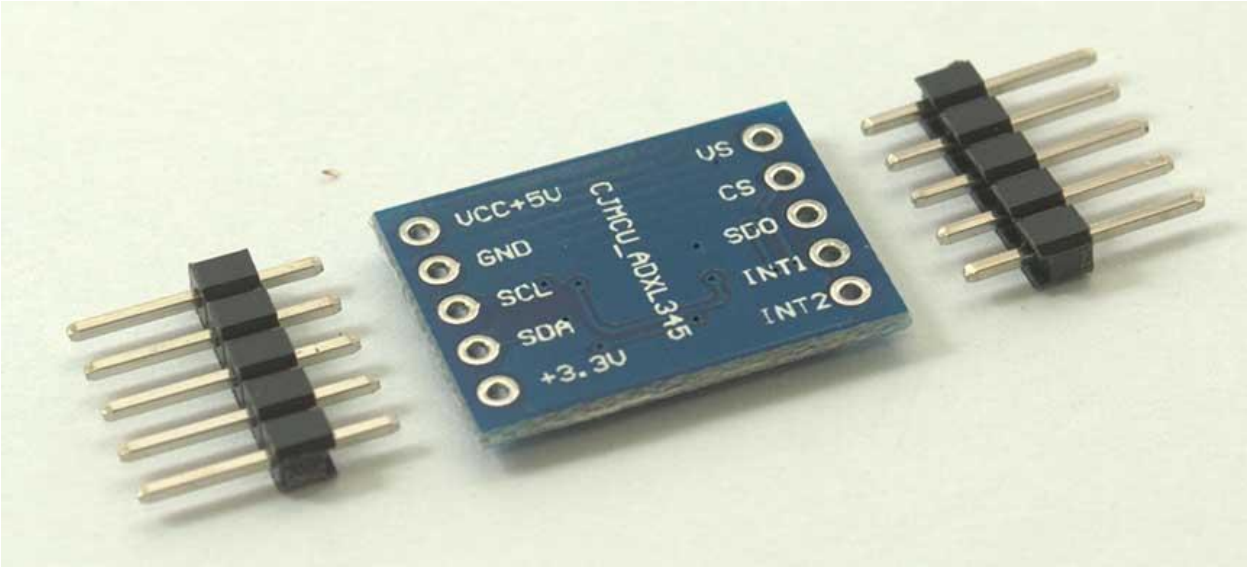
ADXL345 Digital Accelerometer



General Description

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 *g*. Digital output data is formatted as 16-bit two's complement.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 m*g*/LSB) enables measurement of inclination changes less than 1.0°.

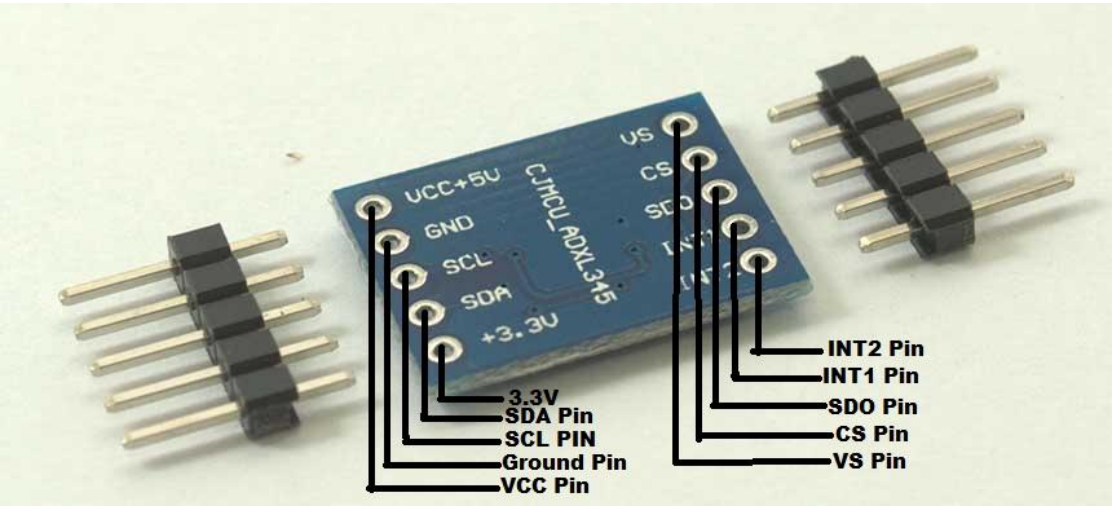
Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated, patent pending memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

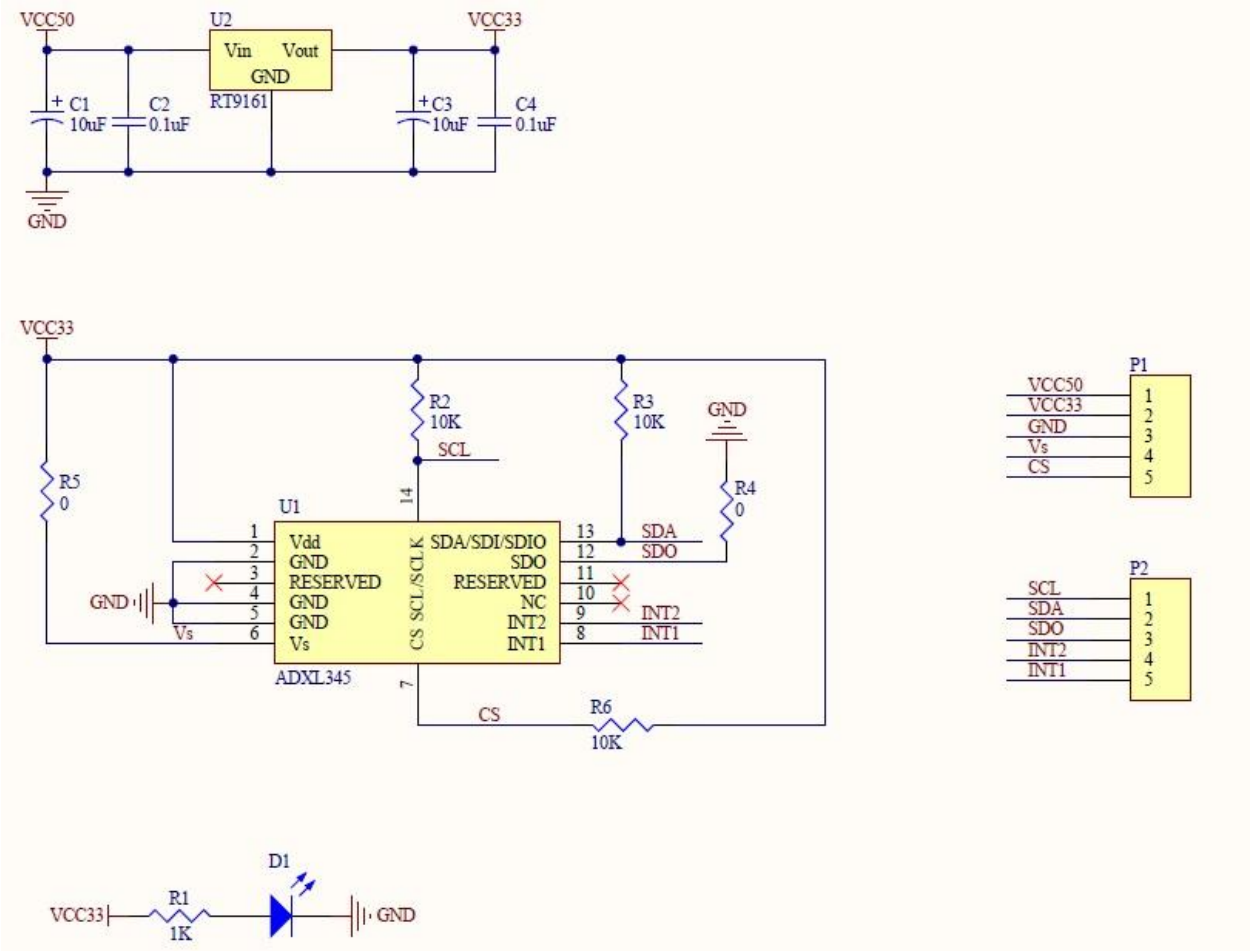
Specifications:

- Main Chipset: ADXL345
- Communication: IIC/SPI Communication Protocol
- Measuring Ranging: ±2g±16g
- Digital Output: SPI/IIC
- 3-axis, ±2g/±4g/±8g/±16g
- Compact Accelemotor/Inclinometer
- Working Voltage: 3V to 5V
- Working Temperature: -40° to 85°
- Working Current: 30uA
- Low Power Consumption
- Compatible with 51, AVR, Arduino

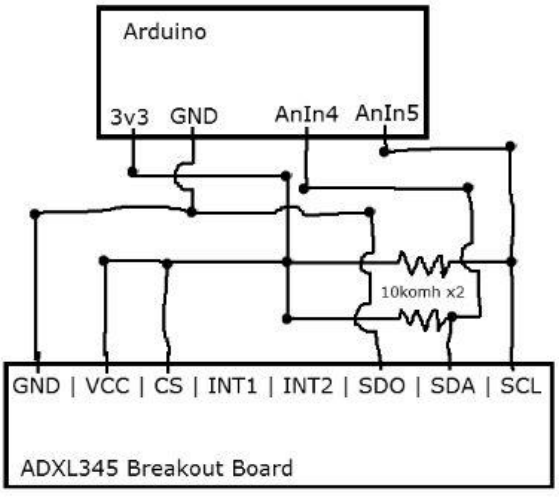
Pin Configuration:



Schematic Diagram:



Wiring Diagram



Sample Sketch

```
#include <Wire.h>
#define DEVICE (0x53)    //ADXL345 device address
#define TO_READ (6)        //num of bytes we are going to read
each time (two bytes for each axis)

byte buff[TO_READ] ;    //6 bytes buffer for saving data read
from the device
char str[512];                        //string buffer to transform
data before sending it to the serial port

void setup()
{
  Wire.begin();        // join i2c bus (address optional for
master)
  Serial.begin(9600);  // start serial for output

  //Turning on the ADXL345
  writeTo(DEVICE, 0x2D, 0);
  writeTo(DEVICE, 0x2D, 16);
  writeTo(DEVICE, 0x2D, 8);
}

void writeTo(int device, byte address, byte val) {
   Wire.beginTransmission(device); //start transmission to
device
   Wire.write(address);        // send register address
   Wire.write(val);        // send value to write
   Wire.endTransmission(); //end transmission
}

void readFrom(int device, byte address, int num, byte buff[]) {
  Wire.beginTransmission(device); //start transmission to device
  Wire.write(address);        //sends address to read from
  Wire.endTransmission(); //end transmission

  Wire.beginTransmission(device); //start transmission to device
(initiate again)
  Wire.requestFrom(device, num);    // request 6 bytes from
device

  int i = 0;
  while(Wire.available())    //device may send less than
requested (abnormal)
  {
    buff[i] = Wire.read(); // receive a byte
    i++;
  }
  Wire.endTransmission(); //end transmission
}
```

```
void loop()
{
  int regAddress = 0x32;    //first axis-acceleration-data
register on the ADXL345
  int x, y, z;

  readFrom(DEVICE, regAddress, TO_READ, buff); //read the
acceleration data from the ADXL345

   //each axis reading comes in 10 bit resolution, ie 2 bytes.
Least Significat Byte first!!
   //thus we are converting both bytes in to one int
  x = (((int)buff[1]) << 8) | buff[0];
  y = (((int)buff[3])<< 8) | buff[2];
  z = (((int)buff[5]) << 8) | buff[4];

  //if(x > 80) Serial.println("right");
  //if(x < -80) Serial.println("left");
  //if(y > 80) Serial.println("up");
  //if(y < -80) Serial.println("down");

  //we send the x y z values as a string to the serial port
  sprintf(str, "%d %d %d", x, y, z);
  Serial.print(str);
  Serial.write(10);

  //It appears that delay is needed in order not to clog the
port
  delay(100);
}
```

How to test:

The components to be used are:

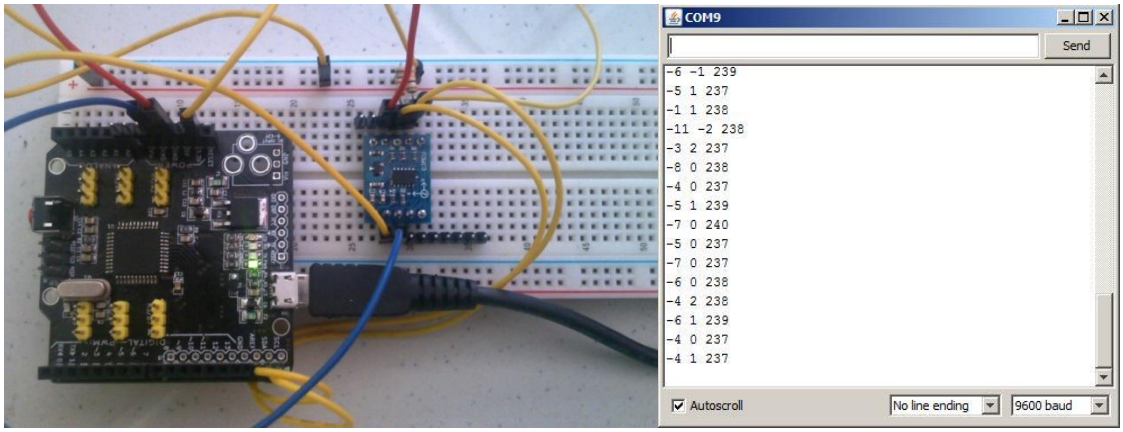
- Arduino Uno (any compatible microcontroller)
- ADXL345 digital accelerometer
- Pin connectors
- Breadboard
- USB cable

1. Connect the components based on the figure shown in the wiring diagram using pin connectors. After hardware connection, insert the sample sketch into the Arduino IDE.
2. Using a USB cable, connect the ports from the microcontroller to the computer.

3. Upload the program.
4. See the results in the serial monitor.

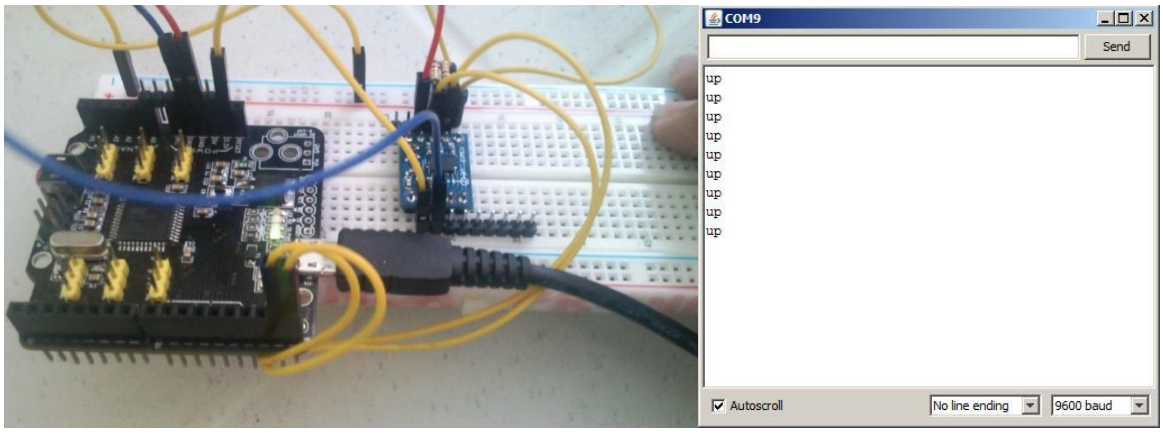
Testing Results

The serial monitor shows the x, y, and z values of the difital accelerometer on standby position. The value is around 0 in each axis at horizontal position and around 260 at vertical position.

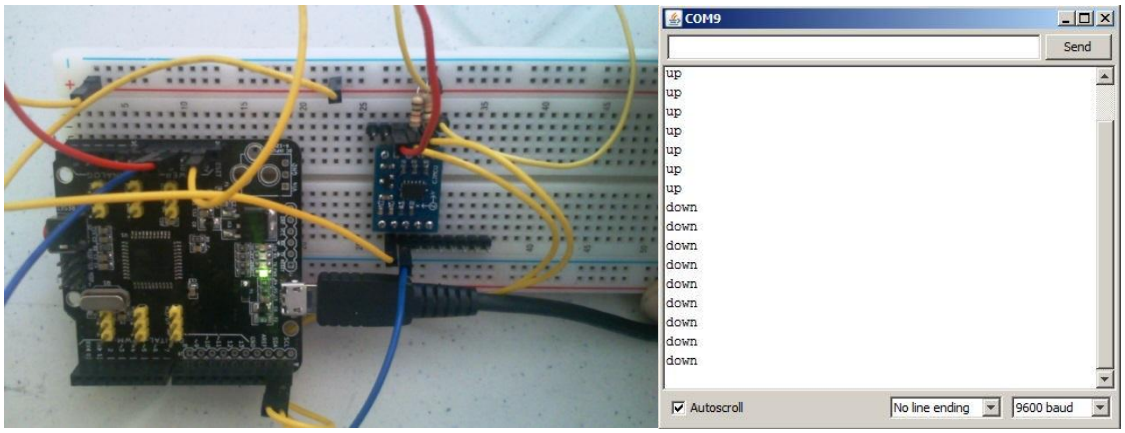


Alternatively, you can set a ±threshold value to detect the tilted position of the module. Replace the Serial.printf function to the if statements written at the last part of the sample sketch.

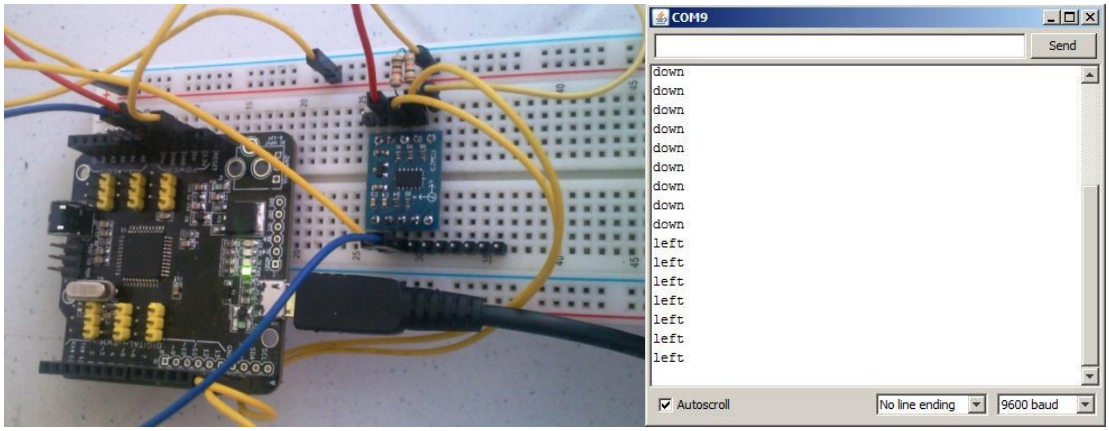
When the module is tilted upward,



When the module is tilted downward,



When the module is tilted to the left,



When the module is tilted to the right,