ADXL345 Quickstart Guide

by zagGrad | January 10, 2011 | 53 comments Skill Level: ★ Beginner

Description:

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface. The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/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 and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention. Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

Features:

- 2.0-3.6VDC Supply Voltage
- Ultra Low Power: 40uA in measurement mode, 0.1uA in standby@ 2.5V
- Tap/Double Tap Detection
- Free-Fall Detection
- SPI and I2C interfaces

Downloads:

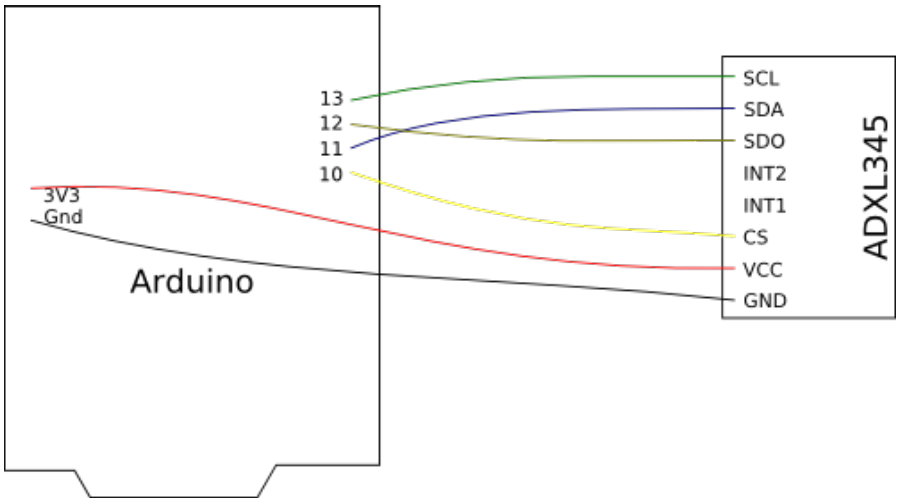
- ADXL345 Basic Arduino Code
- ADXL345 Advanced Arduino Code
- ADXL345 Advanced Processing Code

Hooking it Up:

This section of the guide illustrates how to connect an Arduino to the ADXL345 breakout board. The following is a table describing which pins on the Arduino should be connected to the pins on the accelerometer:

Arduino Pin	ADXL345 Pin
10	CS
11	SDA
12	SDO
13	SCL
3V3	VCC
Gnd	GND

Here is a diagram in case you like pictures!



Beginner Sample Code:

Let's look at a sketch to get the ADXL345 up and running with an Arduino. You can download the sketch in its entirety here. Below we'll examine what the different sections of this code does.

This is the initialization section of the sketch. It's pretty basic really. Here's what's happening.

1. The SPI.h library is added to the sketch. SPI is a communication protocol used by Arduino to communicate to the ADLX345.
2. A variable named CS is created to store the pin number of the Chip Select signal.
3. Variables are created for several of the ADXL345 registers. These variables store the address of the indicated registers and are used to set and retrieve values from the accelerometer. The datasheet shows all of the registers available on the ADXL345 and their addresses.
4. Variables are created that will hold information that has been retrieved from the accelerometer.

```
void setup(){
  //Initiate an SPI communication instance.
  SPI.begin();
  //Configure the SPI connection for the ADXL345.
  SPI.setDataMode(SPI_MODE3);
  //Create a serial connection to display the data on the ter
  Serial.begin(9600);

  //Set up the Chip Select pin to be an output from the Ardui
  pinMode(CS, OUTPUT);
  //Before communication starts, the Chip Select pin needs to
  digitalWrite(CS, HIGH);

  //Put the ADXL345 into +/- 4G range by writing the value 0x
```

The main section of the sketch is split into two parts: the setup and the loop. The setup section is used to configure different aspects of the Arduino to communicate with the ADXL345. At the end of the setup section two functions are used to put the accelerometer into the mode we want to use (+/-4g detection, and enable measurement mode). The functions will be discussed in the next section of the sketch.

The loop does most of the work in the sketch. Here's what's going on in the loop:

1. Arduino reads 6 register values from the ADXL345 using the readRegister function. This command reads the x, y and z acceleration values from the accelerometer and stores them in the buffer named "values" that was created in the initialization section of the sketch.
2. When values are read from the accelerometer they are returned in bytes. A byte is only 8 bits, but the accelerometer reports acceleration with up to 10 bits! In order to see the correct acceleration value we have to concatenate two bytes together. The x, y and z acceleration values are determined by concatenating the bytes from the values buffer.
3. The acceleration values are printed to the serial terminal
4. The sketch waits for 10ms before executing the loop again. This will allow the loop to run at roughly 100 hz.

```
//This function will write a value to a register on the ADXL3
//Parameters:
//  char registerAddress - The register to write a value to
//  char value - The value to be written to the specified reg
void writeRegister(char registerAddress, char value){
  //Set Chip Select pin low to signal the beginning of an SPI
  digitalWrite(CS, LOW);
  //Transfer the register address over SPI.
  SPI.transfer(registerAddress);
  //Transfer the desired register value over SPI.
  SPI.transfer(value);
  //Set the Chip Select pin high to signal the end of an SPI
  digitalWrite(CS, HIGH);
}
```

The readRegister command is used to read values from the ADXL345. When using the readRegister function three variables must be given to the function: the registerAddress where reading should start from, the number of registers that should be read in sequence, and the buffer where the values should be stored. When reading a register from the ADXL345 the address must be modified so that the 8th bit is set high. If more than 1 register is to be read the 7th bit must also be set high. The function accomplishes this task in the first two lines. After that the function is very similar to the writeRegister function. The CS pin is set low to start an SPI sequence, then the address is transferred. After that a loop is used to read the specified number of registers from the accelerometer and the values are stored in the specified buffer. Finally the CS pin is set high to end the SPI sequence and the function is complete.

Once you've hooked the ADXL345 up to an Arduino as specified in the Hook Up section of the guide, download the sketch. After opening the sketch in the Arduino IDE just load it to your board and open the terminal from Arduino. You'll see the X,Y and Z acceleration values start scrolling in the terminal window.

Advanced Sample Code:

Once you understand how to read and write from registers to the ADXL345 there are some really cool functions that you might want to experiment with. In this section we'll look at a sketch that can detect single and double taps, and will convert the x,y and z acceleration values to Gs (a unit of measurement corresponding to 1g). This project will use both an Arduino sketch to interface with the ADXL345 and a Processing sketch so that we can interpret the data. You can download the Arduino sketch here and the Processing sketch here. **In order for the sketch to work properly you'll need to add a wire from the INT1 pin on the ADXL345 to pin D2 on the Arduino.** Let's look at some of the special parts of this sketch.

```
//Create an interrupt that will trigger when a tap is detecte
  attachInterrupt(0, tap, RISING);

  //Put the ADXL345 into +/- 4G range by writing the value 0x
  writeRegister(DATA_FORMAT, 0x01);

  //Send the Tap and Double Tap Interrupts to INT1 pin
```

This code has been added to the setup() section of the sketch. It's a bit different from the setup() section of the ADXL345_Basic example because we're setting the ADXL345 up to detect a single and double tap event. When the accelerometer detects a tap or a double tap we want the INT1 pin to go high, then when the event is handled the INT1 pin will go back low.

When the INT1 pin goes high on the ADXL345 we want to the Arduino to generate an interrupt. In order to do this we use the attachInterrupt() function; we tell the function that we want an interrupt to occur on the Arduino Interrupt 0 (pin D2) when the pin goes from low to high, and we want the Arduino to execute the tap function when this occurs. We'll look at the tap() function later on.

After the interrupt is created we need to configure the ADXL345 to recognize single and double tap events. In order to make this configuration we must change the values in the following registers: INT_MAP, TAP_AXES, THRESH_TAP, DURATION, LATENT and WINDOW. You can read the 'TAP DETECTION' section of the datasheet to see why these values were assigned to their registers. Basically, though, the ADXL345 has been configured so that if a single or double tap is detected the INT1 pin will go from low to high; once the interrupt is handled the INT1 pin will go back low.

```
//Convert the accelerometer value to G's.
  //With 10 bits measuring over a +/-4g range we can find how
  // Gs = Measurement Value * (G-range/(2^10)) or Gs = Measur
  xg = x * 0.0078;
  yg = y * 0.0078;
  zg = z * 0.0078;
```

In the final project we'll want two types of outputs. If a single tap is detected the Arduino will output the raw accelerometer values for x,y and z; just like in the ADXL345_Basic example. However if the ADXL345 detects a double tap we want the Arduino to output the actual G values of the x,y and z axis. In order to find the G values from the raw accelerometer data we must scale the original values. To find the scale we need to know two things: the number of bits used to represent the original values and the range of acceleration values that can be represented. The ADXL345 is

set up to measure the values with 10 bits. We've set the range to +/- 4 g, or an 8g range. The scale is found by dividing the total range by the number that can be represented in 10 bits.

$$Scale = \frac{8}{2^{10}} = \frac{8}{1024} = 0.0078$$

Once we know the scale all we have to do is multiply the raw accelerometer data by the scale to find the number of gs. The variables xg, yg and zg are float variables so that they can hold decimal numbers.

```
if(tapType > 0)
  {
    if(tapType == 1){
      Serial.println("SINGLE");
      Serial.print(x);
      Serial.print(',');
      Serial.print(y);
      Serial.print(',');
      Serial.println(z);
    }
    else{
      Serial.println("DOUBLE");
      Serial.print((float)xg,2);
      Serial.print("g,");
```

When the tap() interrupt function runs (we'll cover this next) it assigns a value to the tagType variable. The variable is assigned '1' if a single tap was detected and '2' if a double tap was detected. If a single tap was detected the Arduino will write the word "SINGLE" to the terminal followed by the x,y and z accelerometer values. If a double tap was detected Arduino will write the word "DOUBLE" to the terminal followed by the g values for the x,y and z axis. After printing the values to the terminal we disable the interrupt for a little while; this prevents the Arduino from detecting any 'echoed' interrupts that may occur while the ADXL345 is still vibrating from the tap event.

```
void tap(void){
  //Clear the interrupts on the ADXL345
  readRegister(INT_SOURCE, 1, values);
  if(values[0] & (1<<5))tapType=2;
  else tapType=1;;
}
```

Because of the way the attachInterrupt() function was called the tap() function will be used whenever Arduino sees an interrupt occur on pin D2. This function is fairly straightforward: Arduino reads the INT_SOURCE register on the ADXL345 and stores the value in the 'values' buffer. The INT_SOURCE register tells us if the interrupt came from a single tap or a double tap event. Depending on which kind of tap set of the interrupt we assign the tapType variable with a 1 or a 2.

Download the Arduino sketch and the Processing sketch. After you've connected the ADXL345 to the Arduino as specified in the Hooking It Up section, and added a wire from the INT1 pin on the ADXL345 to pin D2 on the Arduino, open Arduino and download the sketch to your board. Then open processing and run the ADXL345_Advanced processing sketch. You may have to change the serial port in the sketch to reflect which port your Arduino is plugged into. If everything has been done correctly the Processing window will output the words "Single Tap" to the screen along with the raw accelerometer values when you tap the ADXL345; likewise "Double Tap" and the G values will be displayed when you double tap the ADXL345.

Comments 53 comments

Log in or register to post comments.

Member #123304 / about 4 years ago / ★ 5

Do you have any i2c code examples for the ADXL345?

> Member #634882 / about 4 months ago / ★ 1
>
> https://github.com/adafruit/Adafruit_ADXL345 and https://github.com/adafruit/Adafruit_Sensor For instructions refer to https://learn.adafruit.com/adxl345-digital-accelerometer
>
> rohorn / about 3 years ago / ★ 1
>
> I'd love to see that as well - would go well with the ITG-3200 gyro -or- IMU.

jbeale / about 4 years ago / ★ 4

Your "basic Arduino code" at http://sparkfun.com/tutorial/ADXL/ADXL345_Basic.pde has a bug.
You want the values[] array to be of type UNSIGNED CHAR, meaning you must change two declarations as below:
unsigned char values[10];
void readRegister(char registerAddress, int numBytes, unsigned char * values){
otherwise, you are getting the low-order 8 bits interpreted as a signed char (byte) (incorrect) as well as the high-order bits (which are in fact signed). Combining those two gives you garbage whenever the low byte value is greater than 127.

> Member #277474 / about a year ago / ★ 2
>
> Thank you. This is correct. I was wondering why the values would go negative

over a certain threshold.

asebastian24 / about 3 months ago / ★ 1

Thank you!

Member #79636 / about a year ago / ★ 1

Thank you so much! I have been chasing this down for hours!! Sage advice.

Member #489030 / about 6 months ago / ★ 0

Thank you!

Member #529207 / about 10 months ago / ★ 2

I connected the ADXL345 to my arduino following the instructions and downloaded the basic code to the Arduino, but the Serial monitor gives o/put (0,0,0)..Is something wrong with accelerometer?

Member #568411 / about 11 months ago / ★ 2

I hooked the ADXL345 to my RedBoard following the instructions and downloaded the basic code to the Arduino, but the Serial monitor sees only two types of outputs. 0,0,0 and if I induce freefall, -1,-1,-1. Any idea what could be going wrong?

Member #663592 / about a month ago / ★ 1

How would I change the basic code to be in the +-16 g range??

Member #663081 / about 2 months ago / ★ 1

Any one tried free fall detection using SPI?

Member #256481 / about 2 months ago / ★ 1

I'm trying to use this accelerometer to test various frequencies of vibration on a cantilevered beam and I wanted to know what the maximum vibration frequency I could find with using this accelerometer, connected to a 16MHz processor, connected to my laptop. I'm mainly a mechanical person, but after doing some research I found the following: Accelerometer has a maximum of 3200Hz output data rate (so maximum frequency would be 1600Hz) Processor can execute 16,000,000 instructions/sec so this probably is okay Communication rate is 9600 bits/sec so in the program above we read at a minimum 6 bytes for each axis, so that's 24 bits, leaving you able to read 9600/24 = 400 reads/sec resulting in a one to be capable of reading vibrations at 200Hz. If I bump up the baud rate to 115200, I should be able to then read 4800 times a second, thus measuring 2400Hz vibrations?

Can anyone tell me if my math/thinking is wrong on this? Do I need to use a NI DAQ instead?!

Member #654092 / about 2 months ago / ★ 1

Hi All, Indeed there are some bugs with the values. fixed it a little bit and got the

correct values. (i preferred using the +-2g) //The X value is stored in values[0] and values[1]. x =(int)((((unsigned int)(values[1]&255)) << 8) | (unsigned int)(values[0])&255); //The Y value is stored in values[2] and values[3]. y = (int)((((unsigned int)(values[3]&255)) << 8) | (unsigned int)(values[2])&255); //The Z value is stored in values[4] and values[5]. z = (int)((((unsigned int)(values[5]&255)) << 8) | (unsigned int)(values[4])&255);

xg = x * 4 / pow(2,10); yg = y * 4 / pow(2,10);

zg = z * 4 / pow(2,10);

Serial.print(xg, DEC); Serial.print(‘,’); Serial.print(yg, DEC); Serial.print(‘,’); Serial.println(zg, DEC);

 bboyho / about 3 months ago / ★ 1

The Arduino advanced example code is working fine but I had some issues getting data with the Processing advanced example code. I had to change line 17 in the processing code to fit my port:

```
String portName = Serial.list()[1];
```

to

```
String portName = Serial.list()[2];
```

because the COM port that my Arduino enumerated to on my computer was the third listed port in the array. You can always test this out by opening the serial terminal on Arduino and try to open the COM ports in the list with the Processing. This will give you an error stating the the COM port is being used in Processing because Arduino is using the UART.

 bboyho / about 3 months ago / ★ 1

Checking with someone else around SparkFun that has more experience with Processing, he states that it might be that the data being sent from your Arduino might be too fast for Processing to handle. We noticed that it was getting stuck in the while loop and was not able to check the if statement with “myPort.available() > 0”. Placing a *println()* function for debugging slowed it down enough where you can see the output on the display window. Try testing the advanced processing code with a *println()* function by adding it to line 58:

```
while(message_complete==0)
{
    if(myPort.available() > 0)
    {
```

or

```
while(message_complete==0)
{
    println("testing2");          //add this line to line 5
8
    if(myPort.available() > 0)
    {
```

By adding the print line, the code was able to respond better and output to the display window after tapping the accelerometer.

Member #595241 / about 8 months ago / ★ 1

Can I directly connect SDA and SCL pins of microcontroller and adxl345? As accelerometer is designed to operate at 3.3v and microcontroller operates at 5v. I have a 3.3v supply to power accelerometer but I do not know whether I can connect communication lines directly or not. PLEASE help.

Member #586287 / about 9 months ago / ★ 1

How can I use ADXL345 accelerometer with raspberry pi?

Toni_K / about 9 months ago / ★ 1

Check out the example code from Analog Devices here.

Squalor / last year / ★ 1

I set up a sketch with the basic code, jbeale's bugfix, and the conversion to g's from the advanced code. I edited the appropriate values to read in +/- 2g mode. The z & y axes behaved as I expected, but the x axis only reported values from -2 to 0. Simply adding one shifted the data to look ok, but I'm leery of it. Anyone else find a good way to calibrate?

Squalor / last year / ★ 1

Running in self-test mode seems to have fixed it. Interesting!

Member #550885 / about a year ago / ★ 1

I am using arduino mega 2560. I can't get values from ADXL345. Please help me

Dakine / about a year ago * / ★ 1

I'm getting what I think is invalid output from the basic code, but since there's no sample output included it's hard to say - Can any one confirm?

I'm seeing only 0,0,0 when motionless, which is expected, but any sort of motion produces -1,-1,-1. Rarely, and I mean once in maybe 15 seconds of shaking, I'll get something like 2,0,7943.

I've triple checked my connections, and have not changed the code. I did try jbeale's suggestions as well, no change.

Edit: I should point out that I am using a Micro, if it makes a difference.

Edit 2: Further research would indicate that given the orientation of my chip I should expect the readout to be a steady 0,0,1 when it is not in motion.

 Dakine / about a year ago * / ★ 2

Ok, so the Arduino Micro uses the ATmega32u4 versus the ATmega328P used in the “Generic” Arduino referenced in this guide. This means your pin connections are as follows:

```
      10->CS
MO(MISO)->SDA
MI(MOSI)->SDO
     SCK->SCL
      3V->VCC
     Gnd->GND
```

Resources for figuring this out:

ADXL345 Datasheet, Specifically Table 4

Wikipedia SPI Bus, Specifically the Interface Section

Ardino Micro Schematic, Specifically the top left bit.

Here’s a few lines of sample output:

(At Rest)

```
9,-8,129
9,-9,129
10,-9,129
9,-8,129
```

(Moving it around)

```
19,-25,128
14,4,129
10,-8,129
23,-11,132
6,12,137
11,41,114
36,-3,180
13,-76,99
```

 Member #529429 / about a year ago * / ★ 1

I’m having an issue with the advanced code. It keeps displaying the state of the interrupt and the accelerometer readings every time the loop runs. If I double tap it switches and continues to incrementally display. For example DOUBLE 0.06g,0.02g,0.98g DOUBLE 0.10g,0.12g,0.91g SINGLE 12,9,124 SINGLE 2,-1,46 ETC to infinity and beyond.

This happens when running the code from this page in the digital configuration and it

also does it when i have the board in the analog configuration. Is something wrong with my arduino?

Member #535047 / about a year ago / ★ 1

Hi, When I am compiling the advanced code in which attachInterrupt() is used. I am getting the error and that error is : tap was not declared in the scope. And second error is tapType is not defined. Please, if somebody know the answer of these questions , so please tell me or share the correct code.

Geetanjali

Member #534405 / about a year ago / ★ 1

Hi, I'm using an Arduino Uno, with Arduino IDE, and I don't understand how to use 2 sketch's (Arduino sketch and the Processing sketch), compile Arduino then Processing?

Best regards.

Member #483629 / about 2 years ago / ★ 1

Im also interested how the read function in the beginners example is supposed to work, anyone who can explain perhaps?

It reads: //Continue to read registers until we've read the number specified, storing the results to the input buffer. for(int SPI.transfer(0x00); }

Member #477128 / about 2 years ago / ★ 1

Confusion re. Beginner Sample Code:

1. How does the Arduino know to read/write pins 11 & 12? Only the CS (pin 10) is mentioned in code.
2. Is the SDO pin needed in this example? Since it is using SDI 3-Wire, which means SDA is Serial Data Input and Output, according to specs.

Newb, sorry if bonehead questions. Thx.

> M-Short / about 2 years ago / ★ 1
>
> 11, 12, and 13 are the SPI pins on the ATMega328. When you use the SPI library it knows by default to use these pins. Also, the naming of the pins can be a bit confusing, this board will communicate over SPI or I2C, SDA is the I2C pin name, so if you were using I2C (on A4 and A5), you would only need SDA and SCL, but your code would be different as well.
>
> > Member #557170 / last year * / ★ 1
> >
> > Hi, I implement the I2C interface with ADXL345, in VHDL, you may find the description here : phealth and the source code here lx9health github .
> >
> > please refer to the section **"pmodACL – ADXL345"**

Member #378779 / about 2 years ago / ★ 1

I've implemented the simple sample code, and I'm getting some strange readings. As I approach +1g on any axis, things work normally until I'm really close, with readings approaching +1g. Then it will instantly start to give me a reading around -1g instead of the +1g I was expecting. I can't find anything in the datasheet that would explain this behaviour, in fact on page 22, it says that I should expect a +1g reading when oriented one way, and -1g when oriented the other.

Any ideas, guys?

> Member #332483 / about 2 years ago / ★ 1
>
> If you look on page 18 of the Data sheet the data is in 2's complement… This might explain it?

Member #415223 / about 2 years ago / ★ 1

I am trying to use the Calibration Offset Registers on the ADXL345. Do you have to re-calibrate the offset if you change the range setting (e.g. from +-16g to +-2g)?

Member #386296 / about 2 years ago / ★ 1

In your beginner code, in the readRegister function, I am seeing this:

//Continue to read registers until we've read the number specified, storing the results to the input buffer.

for(int i = 0; i < numB SPI.transfer(0x00);

}

//Set the Chips Select pin high to end the SPI packet.

surely that for loop is not correct?

> Member #387241 / about 2 years ago / ★ 1
>
> It reads:
>
> //Continue to read registers until we've read the number specified, storing the results to the input buffer. for(int SPI.transfer(0x00); }
>
> And I'm also scratching my head over that one. I certainly won't compile.

Member #326558 / about 3 years ago / ★ 1

Are there any additional considerations that need to be made when using an Arduino Pro-mini instead of the Arduino Duemilanove (as this tutorial appears to use)?

AeroGuy_123 / about 3 years ago / ★ 1

Line 130 of the full code, "intType=0;" shouldn't it be "tapType=0;"? "intType" was not declared anywhere and gave me error..

Member #321796 / about 3 years ago / ★ 1

Hey,

I would really appreciate it if someone could let me know why this code does not work with xbee communications…..is it because :

The TX and RX are occupied with the use of the accelerometer

Thankyou

> AeroGuy_123 / about 3 years ago / ★ 1
> It does work with Xbee, there could be many reasons why it is not working for you. Also, accelerometers do not occupy Tx or Rx.
>
> I recommend checking your Xbee connection to Arduino, and check to see if both Xbee devices have same PAN number and configuration.

Member #301341 / about 3 years ago / ★ 1

I'm trying to hook-up two ADXL345's to a single ArduinoBT at the same time. Does anyone know what pins to use and how to change the above code to account for the second sensor?

> MikeGrusin / about 3 years ago / ★ 1
> If you use the I2C interface (which the above tutorial does not, but is very easy to do using the Arduino's Wire library), I believe you can alter the I2C address using the SDO pin. See the datasheet for details.

jtacquaiii / about 4 years ago / ★ 1

Is this in c or c++?

> zagGrad / about 4 years ago / ★ 1
> The Arduino sketch is C++, but since it was written using the Arduino IDE there are some Arduino specific commands that are included by the IDE at compile time.
> The processing sketch is written in…processing. Processing is it's own open source programming language.

Member #225388 / about 4 years ago / ★ 1

Hi,
I've used both example. The basic one runs very fast.
But the advance one is MUCH more longer. Is there any way to detect a tap at the same speed than the reading of basic example? otherwise i can't use that…too long.
Thanks
Fred

Member #166126 / about 4 years ago / ★ 1

OK, nice!
How will the G formula look if the Acc is set to full range? Then a feedback for the range is needed?

NickTheButcher / about 4 years ago / ★ 1

Can you list some example values that should be printing out in the Basic Code example? I want to make sure I am receiving the values correctly.

Member #206692 / about 4 years ago / ★ 1

I have been trying to work with the ADXL345 and Arduino Uno to create some type of “fall detection device”. I have been trying to look at the codes here and I can’t find a program to open them.
What i am trying to do is have the arduino read the data from the ADXL345 and add an algorithm (like a sum vector algorithm svm= sqrt((ax^2)+(ay^2)+(az^2))). With that in mind I wanted to have thresholds set so once the threshold is reached, I would have a piezo transducer sound for a few seconds, and then turn off.
I feel the advanced arduino code would help using the tap and double tap features. I was wondering if anyone could help me with this as soon as possible.
Thanks

jbeale / about 4 years ago / ★ 1

ADXL345 Datasheet: Digital I/O: 3.6V “ABSOLUTE MAXIMUM”. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device.
A 5V Arduino output pin can drive 40 mA current, which is putting stress on the I/O pin internal protection diodes on the ADXL345. Why not at least use a 1k series resistor to limit the current on the 5V outputs from Arduino to about 2 mA. It might save you the cost of another ADXL345 board.

collin5022 / about 4 years ago / ★ 1

Hey there. Would you mind helping me with displaying the three axis on a graph using Processing? The single and double tap stuff is cool, but I want to see these accelerations on a graph, instead of the serial monitor. I’m a newbee, so I’m not sure how to use Processing to accomplish this.
Any help would be greatly appreciated. Thanks!

Duke3k / about 4 years ago / ★ 1

I notice your not doing any logic level conversion (3.3v to 5v) for the SDA,SCL,SDO pins between the sensor and the Arduino pins.
Is this because in your example you are using a 3.3v Arduino or because it’s not neccessary for an SPI interface?
Thanks Marcus

> zagGrad / about 4 years ago / ★ 3
>
> I actually used a 5V Arduino in the example. Though this is ‘bad engineering’ it works fine. In the long run it may end up damaging the pins on the ADXL345, but I’ve yet to have this happen to me. If you’re nervous about using 5V signals to communicate with a 3.3V device you can certainly use the logic level shifter board to interface the two boards.

Member #552236 / last year / ★ 0

hi i am using ADXL with ARM7 lpc2148… i have done all the connection properly and through SPI communication is successful. But the value are wrong i am getting like 5535,5535,5535 and 0000, 0000, 0000,and even i shake the device the response is too slow. please help me