Accelerometer Tutorial (ADXL-345) GetFit

This document guides the user through the following:

1. Wiring and set up depending on communication protocol use
2. Verification that the accelerometer is working as intended
3. Using the accelerometer as a pedometer for step counting

Table of Contents

1 - Introduction

An accelerometer is a tool used to measure acceleration. It can be used in many applications, including step counting, seismic activity, and to measure inclination or speed. Keep in mind, when you're measuring raw data from the accelerometer while it's still, you would expect all readings to be 0, but it will always measure the acceleration due to gravity (9.81 m^2/s) and it may give you values in the other axes due to tilting if the accelerometer is not perfectly flat.

The component we used, the ADXL345, can be purchased online or through Lee's Electronics in Vancouver. It is capable of two forms of communication: SPI and I2C. What's the difference?

SPI is a **FULL** duplex system - the device can send AND receive data simultaneously
I2C is a **HALF** duplex system - the device can send OR receive data

The data sheet: [ADXL345 Data Sheet]
And Pin-out:

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

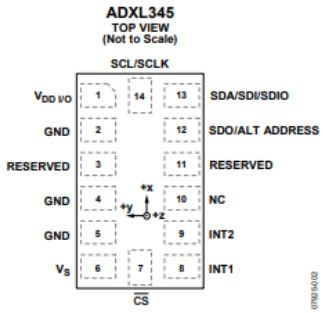


Figure 2. Pin Configuration

Table 4. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	Must be connected to ground.
3	Reserved	Reserved. This pin must be connected to V_S or left open.
4	GND	Must be connected to ground.
5	GND	Must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	Reserved	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output/Alternate I²C Address Select.
13	SDA/SDI/SDIO	Serial Data (I²C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock.

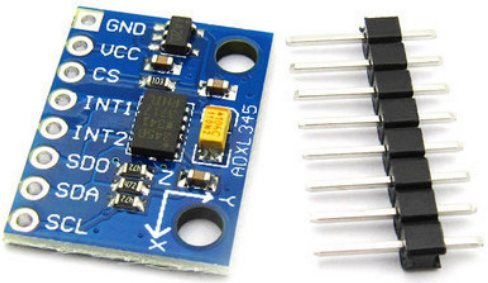
(Found in Pin Configuration section of Data Sheet)

2 - Set Up

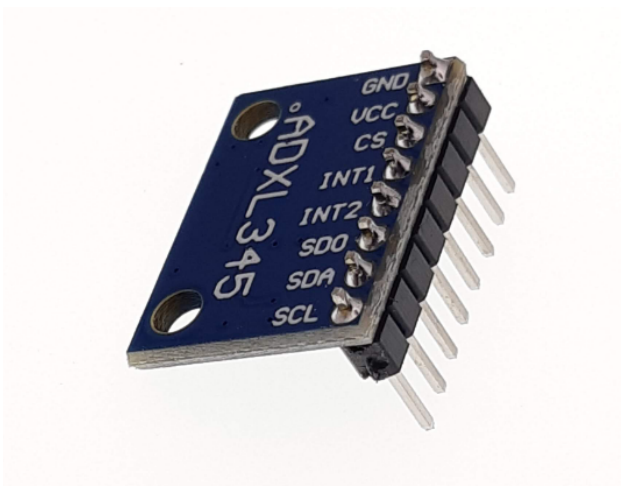
You will need:

- The ADXL345
- (at least) 4 male-to-male wires - Ground, VCC, CLK, and SDA or DO
- (optional) 470 ohm resistors

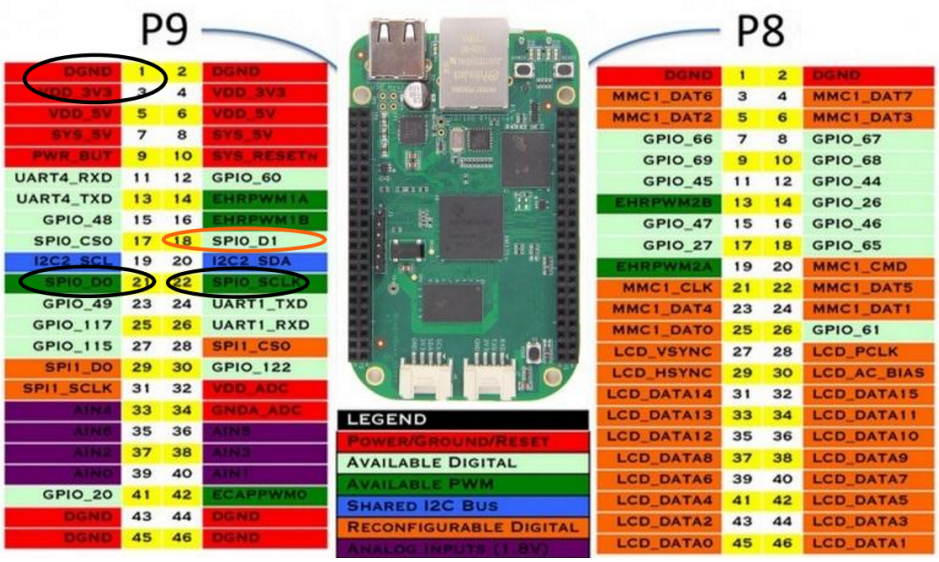
When you receive the ADXL345 component, you will need to solder the pins yourself. Ensure that the board is on top of the short end of the pins before you solder, since it'll be difficult to desolder and fix that mistake.



It should look like this if you've done it correctly:

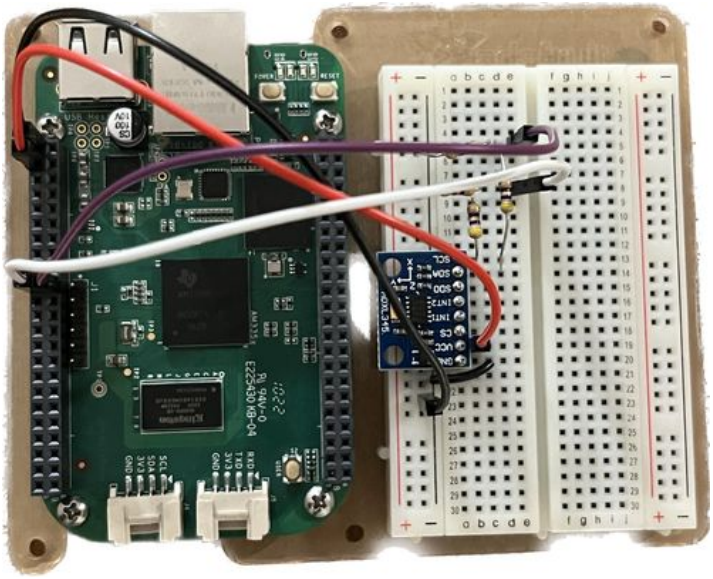


2.1 Wiring for SPI



***Black - necessary (SPIO_DO for Output Only)**

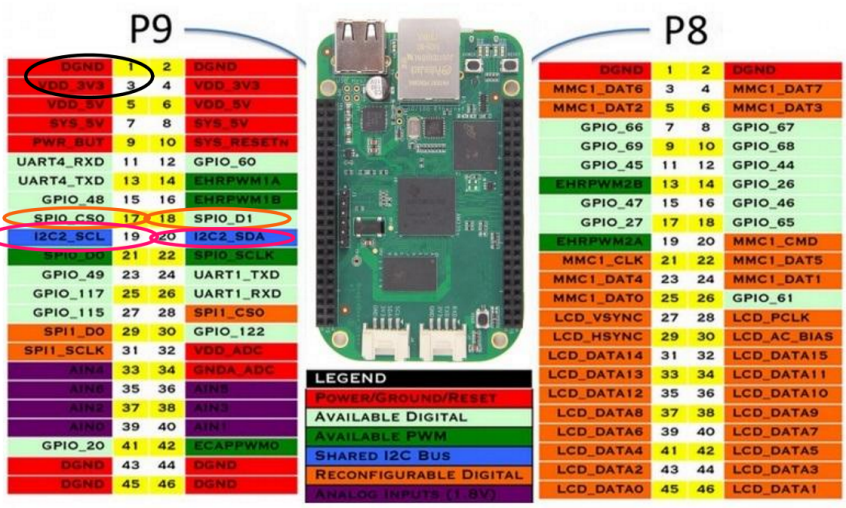
Orange - alternative (because SPIO_D1 can be an Input/Output Pin)



RED - VCC
BLACK - GRD
PURPLE - I2C SDA
WHITE - I2C CLK
**CS (chip select) may need to be tied to VCC

2.2 Wiring for I2C

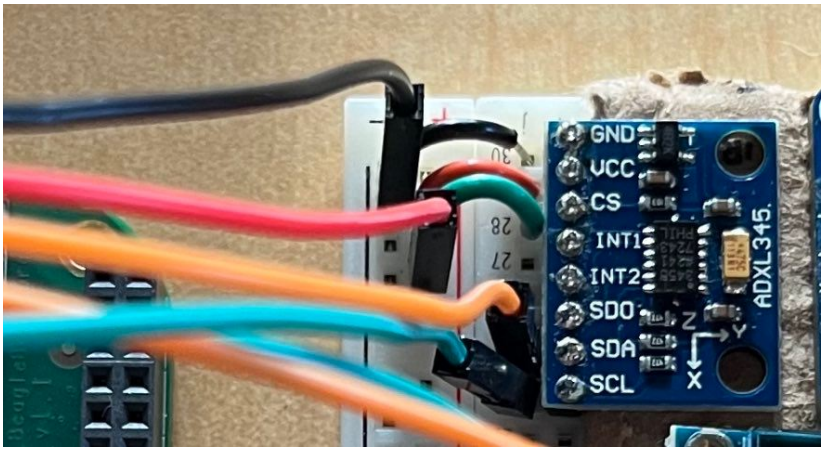
For reference, our group used I2C for the project, our code will be using the protocol as well.



***Black - necessary**

Orange and Pink - you can use either of them, just be sure that you use the correct I2C bus in your code (more information later in this guide) and don't mix them up-

- use P9_17 and P9_18 together (orange, BUS1) OR
- P9_19 and P9_20 together (pink, BUS2)



RED - VCC
BLACK - GRD
SMALL GREEN - CS (chip select) tied to VCC
ORANGE M2M - I2C SDA
GREEN M2M - I2C CLK

3 - Reading Data and Verification

The accelerometer can collect data for three different axes (X,Y,Z) and send that data through the SDA pin. These are the following registers needs to collect raw data from the accelerometer:

0x32	50	DATAX0	R	00000000	X-Axis Data 0.
0x33	51	DATAX1	R	00000000	X-Axis Data 1.
0x34	52	DATAY0	R	00000000	Y-Axis Data 0.
0x35	53	DATAY1	R	00000000	Y-Axis Data 1.
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0.
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1.

(Found in Register Map section of Data Sheet)

To read and write data to and from the accelerometer, you can use the code to read/write from a register from Dr.Brian's I2CGuide: [I2C Guide]

Each axis is split into data0 and data1, which will need to be formatted to create one value for each axis. The following code can be used to format the data0 and data1 value into one:

```
double formatRawData( int data0, int data1)
{
    static double maxResolution = 256.00;
    data1 = data1 << 8;
    int dataOut = data0 + data1;

    double data = (double) dataOut / maxResolution;
    return data;
}
```

The one variable that you can change from this, maxResolution, will depend on your specifications. The accelerometer has user-selectable resolution with corresponding g Ranges. The default resolution for the accelerometer is 2g, therefore, the raw data is divided by 256 to reflect its proper value.

OUTPUT RESOLUTION	Each axis				
All *g* Ranges	10-bit resolution		10		Bits
±2 *g* Range	Full resolution		10		Bits
±4 *g* Range	Full resolution		11		Bits
±8 *g* Range	Full resolution		12		Bits
±16 *g* Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	232	256	286	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	3.5	3.9	4.3	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	116	128	143	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	7.0	7.8	8.6	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	58	64	71	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	14.0	15.6	17.2	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	29	32	36	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	28.1	31.2	34.3	mg/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C

(Found in Specifications section of Data Sheet)

BEFORE TRYING TO READ ANY DATA, ensure that you have done all of the following:

1. You have installed the i2c tools on your beaglebone
    a. `debian@beaglebone:/$ sudo apt install i2c-tools`
2. You have checked that the device is attached to the correct bus [53 should appear for bus 1], if you get all X's or all - -, double check your pinout assignment
    a. `debian@beaglebone:/$ i2cdetect -y -r 1`
3. You have enabled the pins you're using
    a. For bus 1:
    ```
    config-pin -q P9_17
    config-pin -q P9_18
    ```
4. You have taken the device out of power-saving mode (by writing 0x08 to the POWER_CTL register, 0x2d) either by:
    a. Writing it directly in terminal
    ```
    debian@beaglebone:/$ i2cset -y 1 0x53 0x2d 0x08
    ```
    b. Writing to it in your program upon initialization
    ```
    int i2cFileDesc = initI2cBus(I2CDRV_LINUX_BUS1, I2C_ACCEL_ADDRESS);
    // all on one line
    writeI2cReg(i2cFileDesc, 0x2d,0x08);
    ```
5. Tilt the board and use the command to ensure that you're receiving data (ex. 0x34 is Y0)
    a. `debian@beaglebone:/$ i2cget -y 1 0x53 0x34`
    b. See troubleshooting guide below for more information

4 - Using the Accelerometer as a Pedometer

Similarly to the way your phone is able to read and track your steps, so can your beaglebone! Here is some context/background knowledge to turn your accelerometer into a pedometer.

Keep in mind that it may not be 100% accurate, similarly to the way that you can shake your phone or watch and the steps may increase. Additional, proper filtering can be done to make your pedometer as accurate as possible.

To convert the XYZ acceleration vectors into their corresponding magnitudes, use some linear algebra and calculate the magnitude with the following formula:

$$\|v\| = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}$$

As you walk, your phone or device will bounce with each step, (+/- a certain threshold in the up and down axis, in the case of the ADXL345, that is the z axis). If you were constantly sampling data and you lived in a perfect world where the device would stay perfectly still as you walked, the Z axis magnitude will form a sine wave. A "step" would be each peak of the sine wave. You can play around with threshold values to ensure that the pedometer is doing what is intended.

Troubleshooting Guide

- “It keeps saying ‘failure to read I2C bus’ even though I have it connected”
    - Be sure to enable the pins you’re using before reading

      For i2c BUS1:
      ```
      config-pin -q P9_17
      config-pin -q P9_18
      ```

      For i2c BUS2:
      ```
      config-pin -q P9_19
      config-pin -q P9_20
      ```

    - Be sure to make the accelerometer leave it’s power saving mode, if you’re using BUS1, send this command to write the value 0x08 to the POWER_CTL register (0x2d): `i2cset -y 1 0x53 0x2d 0x08`

- “I have set it up properly but when I read raw data, the values don’t change”
    - Be sure to check that you’re using the right I2C bus in your code,

      P9_17 and P9_18 correspond to bus 1
      ```
      #define I2CDRV_LINUX_BUS1 "/dev/i2c-1"
      #define I2C_ACCEL_ADDRESS 0x53
      ```

      P9_17 and P9_18 correspond to bus 2
      ```
      #define I2CDRV_LINUX_BUS1 "/dev/i2c-2"
      ```

    - Be sure to make the accelerometer leave it’s power saving mode, if you’re using BUS1, send this command to write the value 0x08 to the POWER_CTL register (0x2d): `i2cset -y 1 0x53 0x2d 0x08`
    - Check if your pins are shorted. Use a DMM and use one probe on VCC and GRD, then probe SDA and have the other one on all other pins

- “The values I read aren’t what I expected”
    - It sounds silly, but make sure that the accelerometer is as flat as possible. Any tilting may result in “unexpected results”
    - Check your soldering; lack of solder or a short may give you weird values
    - Check your data type for reading the value, signed/unsigned/char/int/double may be affecting what you see
    - Check your resolution and ensure that you’re dividing by the correct value for your specifications (or the default!)
    - Change the I2C bus and try the other one!

Useful References

Accelerometer Resources:
How does an accelerometer work?
https://www.youtube.com/watch?v=i2U49usFo10

ADXL345 Product Information
https://www.analog.com/en/products/adxl345.html#product-overview

I2c set up
https://github.com/nghiaphamsg/BBB-linux-app/blob/master/10_I2C_ADXL345/README.md

Pedometer Resources:
How does a pedometer work? Physics and calculations
https://www.aosabook.org/en/500L/a-pedometer-in-the-real-world.html

Analyzing data for a pedometer
https://www.mathworks.com/help/supportpkg/beagleboneblue/ref/counting-steps-using-beagleboneblue-hardware-example.html