Lab 2 - I2C Comms and Datasheets

ME 451 - Introduction to Instrumentation and Measurement Systems, Spring 2019

### Lab Objectives

* Get an introduction to the Fast Fourier Transform (FFT).
* Develop skills in finding and reading datasheets and other relevant sensor information.
* Gain experience using I2C sensors.

**Lab Sensors for report:** MEMS Accelerometers. This lab is a *2 day lab*.

*Note:* We will use MATLAB for this lab. If having the right version of MATLAB is an issue, use [matlab.mathworks.com](http://matlab.mathworks.com)

# Section 1: Setting Up and Connecting to the Accelerometer

1. There are many websites that have helpful guides for using electronics like [Sparkfun](https://www.sparkfun.com/), and [Adafruit](https://www.adafruit.com/) (and many others). Thankfully, Adafruit has a guide for our accelerometer. Let’s use it to set things up: [link](https://learn.adafruit.com/adafruit-lis3dh-triple-axis-accelerometer-breakout/overview)
   1. *Note:* it’s easier to connect the accelerometer directly to your arduino with the long wires in your kit. Don’t use the breadboard!
   2. *Note:* websites like these are great resources for help if you need it on your labs or on your project. Keep them in mind!
2. Now it’s time to read the accelerometer. We will keep using the Adafruit guide.
   1. Install both libraries which they provide.
   2. There’s an easier way to add libraries to the arduino IDE: [link](https://www.arduino.cc/en/guide/libraries). Use the “importing a .zip library” option.

1. In the code, remove the lines which gives you the calculated force of gravity. Instead, output the raw value readings to the serial monitor for each axis.
   1. Make sure that you have set your accelerometer to the 2g setting.
2. **Signoff 1:** A Working Circuit:
   1. Have successfully connected accelerometer to Arduino, using 2g setting.
   2. Have a working circuit that doesn’t use the breadboard.
   3. Outputs raw value to the serial monitor.

## Section 1 Discussion Questions

**Discussion Question 1:** What are the important pins for I2C on the accelerometer? Where do they go on the Arduino? If you had a second I2C sensor, what pins would you use to hook up both sensors on the same Arduino? Why?

**DQ 2:** If we look at the LIS3DH datasheet they refer to g. What is a ‘g’? What is the reason for using it?

# Section 2: Getting Useful Data from the Accelerometer

1. Unfortunately, not all of the data on the accelerometer’s datasheet is correct (specifically the raw value to g resolution listed). This can occasionally happen with datasheets. You can access the datasheet from the Adafruit guide.
2. This means that we will have to convert the raw data to meaningful data ourselves. We will convert the readings in this way: raw value to volts, volts to g’s, g’s to degrees.
   1. *Note:* each conversion will need its own y = mx + b equation. Keep the conversions separate for this lab. You may not use the map() function.
   2. Raw value to volts:
      1. The sensitivity readings on the datasheet in section 2.1 are incorrect. As frustrating as it is, there is a way to circumvent the sensitivity. A lot of the other data on the datasheet is correct --- we will use this for our unit conversions. We will consult the first page of the datasheet (called the Sensor Overview) to make a conversion from the raw value that you get from the Arduino to volts.
      2. To go from raw value to volts you will need to find the resolution. There are two parts to this: the Full Scale Range (FSR) and the number of bits of the ADC. Finding this information is straightforward and can be done from the Sensor Overview.
         1. Typically the FSR is the same as the operating voltage of the breakout board. How many volts are you powering the accelerometer with? What is the supply voltage that the chip requires? This is your FSR.
         2. The accelerometer breakout board has its own ADC. On the sensor overview page, find how many bits the ADC is (aka how many bits it outputs).
         3. Use these numbers to calculate your raw to volts resolution. Use the resolution as the ‘m’ in your y = mx + b equation. ~~For this conversion, you do not need a ‘b’.~~
   3. Volts to g’s:
      1. We will have to take our own data to convert from volts to g’s. What we need to get are the ranges of data to convert to, similar to the map() function. You may not use the map() function, however.
      2. For each axis, take voltage readings at 0G, 1G, and -1G.
      3. It’s time to construct your y = mx + b equation (one for each axis). Use your readings at 1G and -1G to calculate your resolution. Use your reading at 0G to calculate ‘b’.
   4. G’s to degrees:
      1. Converting from G’s to degrees is deceptively simple (don’t overthink it). We can do it without consulting the datasheet and without taking your own data.
      2. *Think:* what is the number of degrees at 0G, 1G, and -1G for each axis? What simple way can you convert from G’s to Degrees with this in mind?
         1. *Note:* Do not use arcsine, arccosine, or arctan. Although elegant, these create problems that are difficult to deal with. It is ok to linearly approximate this conversion, it’s close enough for our lab.
   5. Configure your Arduino to output all of these values together.
      1. Organize the serial monitor by axis.
      2. What it should look like:
         1. time, (put some spaces), x\_volts, x\_g, x\_deg, (put some spaces) , y\_volts, y\_g, …
3. The Fast Fourier Transform (FFT) is a really powerful tool which you will learn about in lecture. We will use it in MATLAB for this lab.
   1. Look up the MATLAB FFT function. Implement the ‘Noisy Signal’ code example straight from the site in MATLAB on your computer.
   2. Read through the example and understand what you will need to do to use the FFT function on your own data.
4. **Signoff 2:** Correct conversions, implemented FFT example.
   1. Demonstrate correct conversions:
      1. Raw value to volts.
      2. Volts to g’s.
      3. G’s to degrees.
   2. Implemented FFT example.
      1. Demonstrate to your TA what you need to change to make it use your own data, not the example’s.
5. Now let’s collect some data with a simple accelerometer motion.
   1. We call it the Z-flip: The goal of this motion is to rotate the accelerometer between -1G and 1G in the Z axis.
   2. We want you to do the Z-flip at 1.5 Hz. Take some data and use FFT to verify what frequency you are flipping at. Repeat the process until you can verify with the FFT that you are rotating the accelerometer at 1.5 Hz.
      1. *Note:* Set a 50 millisecond delay at the end of your loop function. Remember to have a timestamp!
      2. *Note*: You will need to set the Fs variable based on your data. Determine the average sampling frequency from your data and enter that value. The Length variable is how long the array is that MATLAB read in.

## Section 2 Discussion Questions

**DQ 3:** What is the operating temperature of the accelerometer? Calculate what the sensitivity and 0G level offsets would be at both the low and high temperature extremes. If we had the temperature sensor from lab 1 hooked up, rewrite your volts to G equation with temperature compensation for both the sensor’s sensitivity and 0G level factored in. *Note:* You do not need to do this in your code.

**DQ 4:** Plot the data which you collected and the corresponding FFT plot. FFT also tells us what noise is present in your data. Highlight any large bumps in your data, and give a plausible reason for that noise source. *Note:* For your answer, use the subfig package to couple your data and FFT plots.

# Section 3: Tap Detection

1. The Adafruit guide/code has a built-in function for detecting single and double taps on the accelerometer. Their implementation, although nice, is limited. We want you to implement your own tap detection to detect single taps and double taps.
   1. Use the Adafruit guide’s description of how their tap detection system works to make your own tap detector.
   2. It should work with this double tap criteria:
      1. Only count taps that occur within 600 milliseconds of each other as double taps - everything else should register as single taps.
2. **Signoff 3:** Tap detector.
   1. Demonstrate your working tap detection to your TA.
3. You’re done with Lab 2.
   1. Make sure to clean up your space.

## Section 3 Discussion Questions

**DQ 5:** Describe how you implemented tap detection. Provide a plot that shows acceleration readings and it detecting two single taps (1 second apart) and a double tap (half a second apart). Describe how you detect taps, especially how you detect a double tap (as opposed to two single taps)?

# Post-Lab Questions

**Post-Lab Question 1:** Run FFT on your light sensor data from lab 1 and show both the raw data and FFT plots. What frequency is it at? What is the percent error from your lab 1 frequency calculations? If there is a major difference between the two, explain why this might be the case.

**PLQ 2:** Given the signal below, draw what the FFT plot would look like. You can use an app like Camscanner to scan a hand-drawn FFT plot for this question.

**PLQ 3:** Read the main answers from [this page](https://electronics.stackexchange.com/questions/167211/how-do-integrated-circuit-design-companies-create-their-datasheets/167220) about how datasheets are made. Who is the main person in charge of a sensor’s specifications? Who runs the tests? Who actually formats the datasheet?

### Group Names:

(in pen)

# Lab 2 Signoffs

1. \_\_\_\_\_\_\_ Successful connection to accelerometer.
2. \_\_\_\_\_\_\_ Accurately implemented conversion from raw value to degrees.
3. \_\_\_\_\_\_\_ Implemented your own tap detection to detect single and double taps.

**TA Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Date**:\_\_\_\_\_\_\_\_\_\_\_