Lab 3 - Software Sensor Filtering

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

## Lab Objectives

* Learn and implement basic filtering techniques on an arduino.
* Learn about interrupts and its advantages over sensor polling.
* Understand how sample rate affects music.
* Apply filtering techniques to music files.

**Lab Sensors for Report:** Button, Moving average filter\*. This is a 3 day lab.

\*although this is a filter and not a sensor, you can still describe what it does and how it can make sensors late and wrong (don’t worry about noisy).

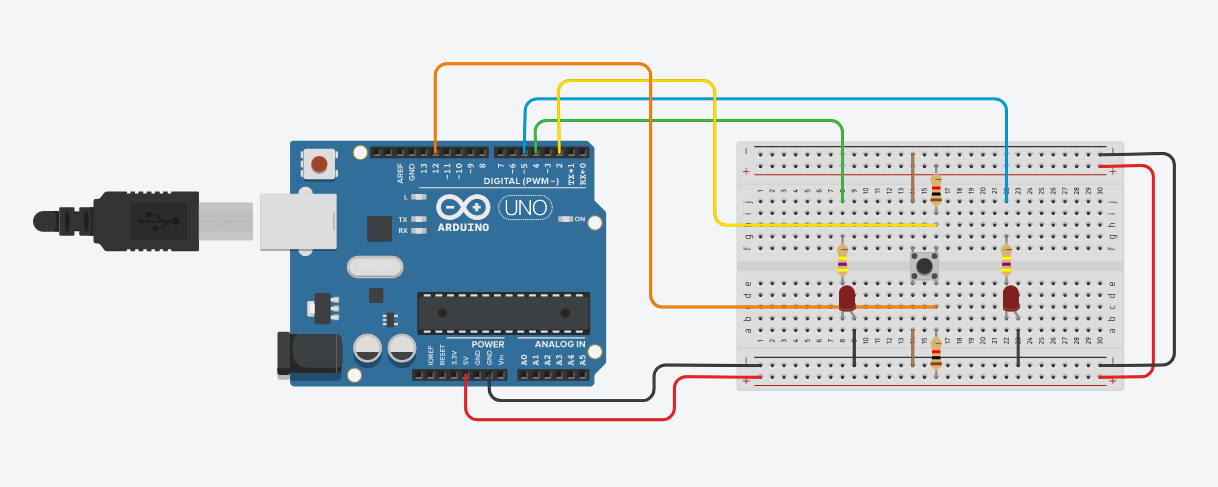
# Section 1: Real-time Moving Average Filter

1. To implement a moving average filter we are going to combine our Lab 2 (Section 2) code with the smoothing example (provided by Arduino, go to [File -> Examples -> 03.Analog -> Smoothing]).
   1. *Note:* combine the code in a way so that the smoothing function is using the values that you get from the accelerometer in the Z axis. Output the unfiltered number and the filtered number to your serial monitor, so you can compare the unfiltered and filtered signals.
   2. *Note:* refer to the documentation for the smoothing function on the Arduino site for more information about that example.
2. Reduce the amount of information that you output to the serial monitor to simplify your code. You will only need to print the G values for the Z axis of the accelerometer.
   1. Print both the unfiltered and filtered values, along with a timestamp.
3. **Signoff 1:** Correctly implemented moving average filter, using a moving average window size of 50.
   1. Successfully merged smoothing example code with your accelerometer code from Lab 2, putting the raw and filtered data in one graph.
   2. You can tell the TA what you need to change to make the filtering window larger or smaller.
   3. Can describe what a moving average filter does.
4. We will record data with a moving average window of 10, 25, 50, and 100 samples, using the same Z-flip motion from Lab 2.
   1. You will have to record data four times, each time using a different moving window.

## Section 1 Discussion Questions

**Discussion Question 1:** Show the data and FFT plots for each moving average window size. Make sure to also plot the raw data that you collected. For both the raw data and FFT plots, show both the unfiltered and filtered data values. Describe the effect each window size has on the raw data and how the filtered data responds to the Z-flip motion. In your opinion, which window size is the most useful? Why?

# Section 2: Learning about Interrupts

1. Buttons are actually really noisy, but we won’t see how bad they are until we implement a new way to read sensors: called the interrupt. First, you will implement a test circuit to understand the advantages of the interrupt over reading the sensor in the loop function, which is called polling.
2. Download the Lab 3 zip file and open the provided interrupt code.
   1. The buttons in our kit have two button circuits built into one. In the code, an interrupt is set up to read one side of the button. We will read the button on the other side using polling.
   2. Set up the circuit below for the provided code:
3. **Signoff 2:** Got the example code and circuit set up correctly.
   1. Demonstrate an LED race to your TA.
4. Record the trigger times for the interrupt and polling 5 times. Then, increase the delay by 100 milliseconds and repeat your 5 trials.
   1. Compare the difference in polling-interrupt Δt. Average the time difference between polling-interrupt Δt for each data set. Keep increasing the delay until you get an average polling delay of 500ms.
      1. *Note:* This simulates having other operations running.
   2. You can assume that the time that the interrupt was triggered is the actual time that the button was pressed.

## Section 2 Discussion Questions

**DQ 2:** Put all of your LED race data (the times that the interrupt and button triggered at) into a table. Calculate the time differences between the interrupt and button trigger times, and calculate the average difference and standard deviation for each set of button presses. How large is the standard deviation? Why might the time difference between interrupt and polling be 1ms one time and 400ms the other for the same delay? What is the difference between polling and interrupts? Make your own analogy.

# Section 3: Button Debouncing

1. Now that we have worked with interrupts with some example code, it's time to implement your own interrupt code to count button presses.
   1. Output the number of button presses to the serial monitor.
   2. Modify your circuit for this part:
      1. Remove the led circuits.
      2. Remove the setup connected to pin 5.
      3. Keep the setup connected to pin 2.
   3. *Note:* use the code example described in the attachInterrupt function on the arduino website as a reference. The code should be very similar to this example.
   4. Once you have the button bouncing code, it's time to see noisy buttons in action.
      1. Press the button 20 times at a slow pace (~1 second per press). Record the number of presses. Repeat this 5 times.
         1. *Note:* You can reset the count on your arduino by hitting the reset button on the arduino board.
      2. Now press the button 50 times at a fast pace (hit the buttons as fast as possible). Record the number of presses. Repeat this 5 times.
2. **Signoff 3:** Implemented button press counting interrupt code.
   1. Code recognizes a button press and displays a button press count to the serial monitor.
   2. Can describe what a button bounce is.
3. To deal with bouncing, we will use a software filter. Implement the debouncing code given [here](http://www.instructables.com/id/Arduino-Software-debouncing-in-interrupt-function/).
   1. Where the code says “//Do something” make a variable that is incremented by 1 inside the function.
   2. In loop, print the value of that variable to the serial monitor.
   3. Now repeat our button pressing test:
      1. Press the button 20 times at a slow pace (~1 second per press). Record the number of presses. Repeat this 5 times.
         1. *Note:* You can reset the count on your arduino by hitting the reset button on the arduino board.
      2. Now press the button 50 times at a fast pace (hit the buttons as fast as possible). Record the number of presses. Repeat this 5 times.

## Section 3 Discussion Questions

**DQ 3:** What causes a button to bounce? What do button designers do to try to mitigate bouncing (hardware-wise, not software)? Make sure to cite your references.

**DQ 4:** Put all of the data you collected for button presses into a table. How does the debouncing code work? What is the best debouncing time for the slow and fast scenarios? Why? What can happen if your debouncing time is too long?

# Section 4: Filtering Music Files

*Note:* We highly suggest that you use headphones for this section of the lab.

*Note:* You will also need at least MATLAB 2016 installed. If having the right version of MATLAB is an issue, use [matlab.mathworks.com](http://matlab.mathworks.com)

1. For this section, we will filter some music files to remove instruments. Provided is a handy instrument frequency spectrum for this section: [link](https://www.guitarbuilding.org/wp-content/uploads/2014/06/Instrument-Sound-EQ-Chart.pdf).
   1. Provided in the zip file is a song snippet. We will use the [filter 1 function](https://www.mathworks.com/matlabcentral/fileexchange/53534-filter1) to filter the music.
      1. *Note:* use the ‘audioread’ function to read in the music file. This is the ‘original’ plot in DQ 6.
   2. First, using filter1’s lowpass filter functionality, find the cutoff frequency that filters out the high frequency percussion without interfering with the other instruments (do this as best as you can). This is the ‘lowpass’ plot in DQ 6.
      1. *Note:* if you want to listen to your music file without exporting it, you can use the ‘sound’ function to play your data. You can stop the music anytime by typing ‘clear sound’ in the command window.
   3. Next, using filter1’s highpass functionality, find the cutoff frequency that filters out the ringing from the cymbals without interfering with the other instruments (do this as best as you can). This is the ‘highpass’ plot in DQ 6.
2. **Signoff 4:** Implemented MATLAB filtering.
   1. Demonstrated code that can filter music using the filter1 command.
   2. Show the TA your music with the high frequency percussion and bass filtered out (separately).
      1. The TA will need to be able to hear the difference (they will also have headphones).
3. Now we will downsample the music to make it at a lower frequency. We will see how the filter affects the lower sample-rate music.
   1. Downsample the music data to half of the sample frequency (DQ 6, downsample1). Without changing your cutoff frequencies, run your lowpass and highpass filters (separately) on the downsampled data (DQ 6, lowpass\_downsampled, highpass\_downsampled, respectively).
      1. Record the effect that the filter has and any differences you can discern between the filtering of the original music vs. the downsampled music.
   2. Downsample the music data by half again (DQ 6, downsampled2). Record your impressions of the music compared to the original sound.

## Section 4 Discussion Questions

**DQ 5:** Why do we use .wav files? What is special about the mp3 file, and how do we make one from a .wav file?

**DQ 6:** Make raw data and FFT plots of all of the music you generated, combine plots where applicable:

* original & lowpass & highpass
* downsampled1 & lowpass\_downsampled & highpass\_downsampled
* original & downsampled1 & downsampled2

Describe the sound difference between these plots:

* original vs lowpass
* original vs. highpass
* original vs downsampled1 vs downsampled2

Also answer: in the filter1 documentation, there is the option to change the order of the filter. What does this do? What effect does it have on the filter? Implement a filter with an order of 10 and describe how it's different from a filter with an order of 1. Provide a plot comparison for evidence.

**DQ 7:** What is the importance of sampling rate? What is the nyquist frequency? Why is there such a difference between the downsampled and original music? Hint: look up harmonics.

# Post-Lab Questions

**Post-lab Question 1:** In the last few labs we used sensors that were noisy. You will use filters on your old data to see their effect. You will use a moving average filter in MATLAB (see [here](https://www.mathworks.com/help/matlab/ref/movmean.html)) on the data.

* 1. *Temperature data*: Use this moving average filter on your temperature data from lab 1. Use a starting average window of 10 samples.
     1. What is the result?
     2. Increase the number of samples calculated in the average to 25, 50, 100, and 1000. How is the signal affected?
     3. Choose a window size that you believe is the most suitable for this sensor if you were to use it in real time. Which window size did you choose and why?
  2. Now use the moving average code on your light sensor data, again starting with a 10 sample average window.
     1. How is the data affected?
     2. Increase the number of samples to 25, 50, 100, and 1000. Is there much difference?
     3. Is this type of filter suitable for this sensor? Why or why not?
  3. Using your results from the above subquestions, discuss the pros and cons of the moving average filtering technique.

**PLQ 2:** *Hardware filters:* What does a high-pass filter do? What does a low-pass filter do? What does a bandpass filter do? Show the circuit for a first order, passive implementation of each filter. What components would you need to make each circuit?

**PLQ 3:** Compare and contrast between the software and hardware filtering techniques for sensor filtering. What are the advantages and disadvantages of both? Which would you prefer? Why?

**PLQ 4:** What are active filters? How do they differ from passive filters? What exactly makes them active? What is an instrumentation amplifier?

**PLQ 5:** *Debouncing:* in the lab you implemented software debouncing. Provide an example of an hardware debouncing circuit (RC version), and describe how it works. What are the advantages and disadvantages of using a hardware debouncing circuit?

### 

### 

### Group Names:

(in pen)

# Lab 3 Signoffs

1. \_\_\_\_\_\_\_ Correctly implemented real-time moving average filter.
2. \_\_\_\_\_\_\_ Correctly set up the led race experiment.
3. \_\_\_\_\_\_\_ Implemented interrupt button counting code.
4. \_\_\_\_\_\_\_ Demonstrated music filtering MATLAB code, with lowpass and highpass already implemented.

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

### 