ECE167L

Xingyu Zhou

Max Dunne

2020/01/25

Lab 1 Report

Introduction

This lab is an introduction to sensors that utilize resistances. The lab is divided into 4 parts. The first three parts are pretty similar to lab 0—making music using the flex sensor and the piezo sensor. The last part is math heavy—we need to figure out the transfer function of various filters such as high/low pass filter as well as the Sallen-Key filter, plot the theoretical responses, build the actual filter and plot the actual responses again.

Part 1: Flex Sensor

The flex sensor is one of the resistive sensors and looks like a thin strip. What resistive sensor means is that the resistance will change based on the shape of the flex sensor. For example, the more you bend the strip, the higher resistance you will get. To begin with, we need to connect the flex sensor with the clincher connector provided with the lab kit. This is pretty simple since the detailed instruction is provided by TAs.

The next part is linearization. According to the lab manual, the flex sensor nonlinear, which means that the changes will not be in a straight line if I want to plot them out on paper. If I want to generate a smooth sound using the flex sensor, I need to linearize it. The first thing I do is to connect the flex sensor to one of the AD pins on the Uno32. Figure 1 is a diagram showing how this is connected. I chose a resistor of 82k Ω in series to further reduce the noise that I got from the AD readings so that I can get the linearized result more accurately.

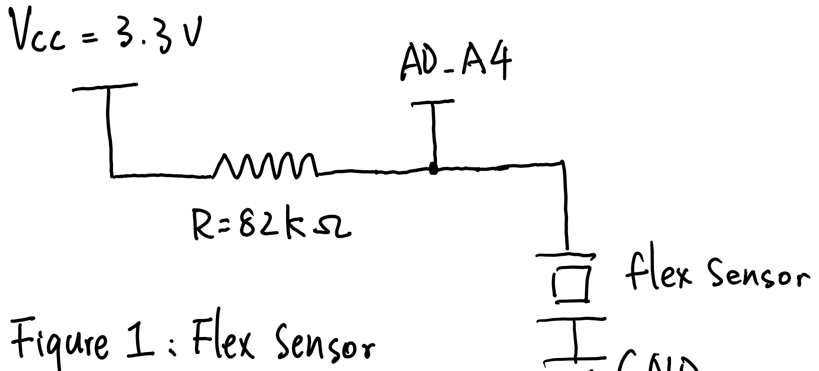
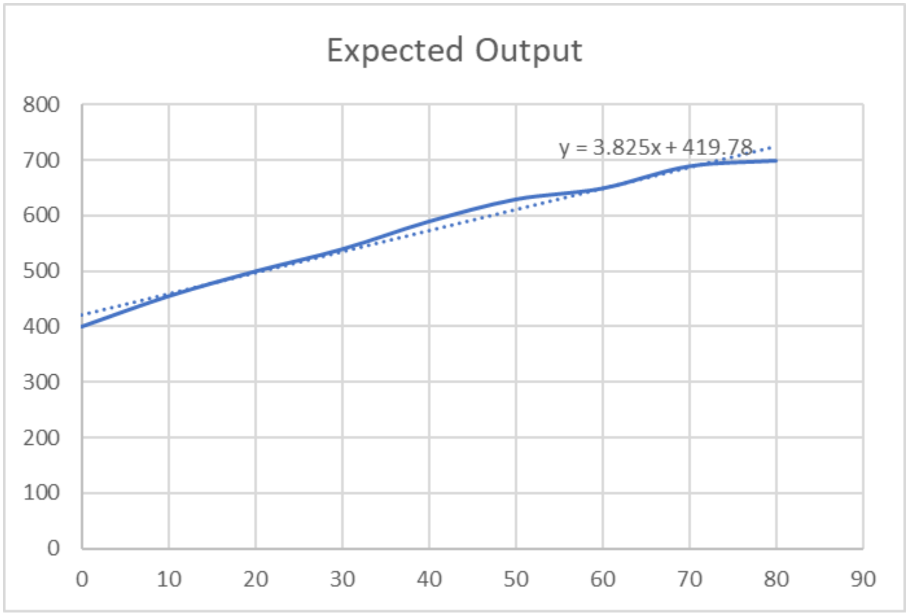


Figure 1: Flex Sensor linearization setup.

Then I printed out an angle sheet and held my flex sensor still. Starting from 90 degrees, I bent the flex sensor by 10 degrees each time and recorded the corresponding AD reading displayed on the IO shield. There were 9 sets of data in total recorded and I used Excel to get the linearized result. The final formula I got is

$$y = 3.825x + 419.78$$

In which "y" is the AD reading and "x" stands for the angle of the flex sensor. These data are provided in figure 3 below.



Input (Degree)	Expected Output
0	400
10	455
20	500
30	540
40	590
50	630
60	650
70	690
80	700

Figure 2: Flex Sensor Linearization result

After the linearization has been done, we were asked to make some music. This is very similar to Lab0 but instead of using the onboard potentiometer, we were asked to use the flex sensor. I used the same setup that I built in Lab 0 with the addition of the flex sensor (detailed diagram will be provided at the end of the report) as well as a potentiometer to adjust the volume. For the coding part, I took the code from Lab0 but modified it to make sound using the AD reading of the flex sensor. One thing to mention is that I used the inverse function of my linearization formula, which was:

$$x = \left(\frac{y - 419.78}{3.825}\right) * 10 + 100$$

In this formula, “x” is the bending angle of my flex sensor, “y” is the AD reading, “*10” is to make sure that the sound at lower angle is audible (for example, at 10 degree, the output is 100) while the “+100” is to make sure that I can still make a sound when the bending angle is 0.

Result of Part 1: Flex sensor

To my surprise, the reading is quite stable due to the software filter that I have implemented in lab 0 and that resulted in a quite smooth tone being generated by the flex sensor. This indicates that my linearization is correct, and I would add another capacitor (maybe around 10 microFarad) in parallel with my flex sensor if I want to improve the sound.

Part 2 & 3: Piezo Sensor

In this part we were introduced to a new component—the piezo sensor. The piezo sensor is used to capture vibration or tap, and it will generate a voltage spike when it’s being pressed. Figure 3 is the data that I recorded using the oscilloscope that shows the value of the voltage that the piezo sensor is generated.

Taps Average			
Peak to Peak	Max	Min	AVG
10.21V	5.11V	-5.11V	-1.31V
100.9V	49.8V	-51.1V	-1.33V
143V	48V	-94V	-1.31V
101V	32V	-68V	-1.31V
145V	48V	-96V	-1.31V
127V	40V	-86V	-1.33V
117V	38V	-78V	-1.33V
133V	44V	-88V	-1.34V
161V	54V	-107V	-1.36V
121V	40V	-80V	-1.35V

Figure 3: Datasheet of the voltage spikes generated by piezo sensor

As you might have noticed, the voltage spike that the piezo sensor generated is very high, so we were asked to use a “snubbing circuit” to lower the voltage so that the pins on Uno32 are protected. Figure 4 is an improved version of the snubbing circuit provided in the lab manual because I wanted to make sure the maximum voltage that the sensor generated was below 4V.

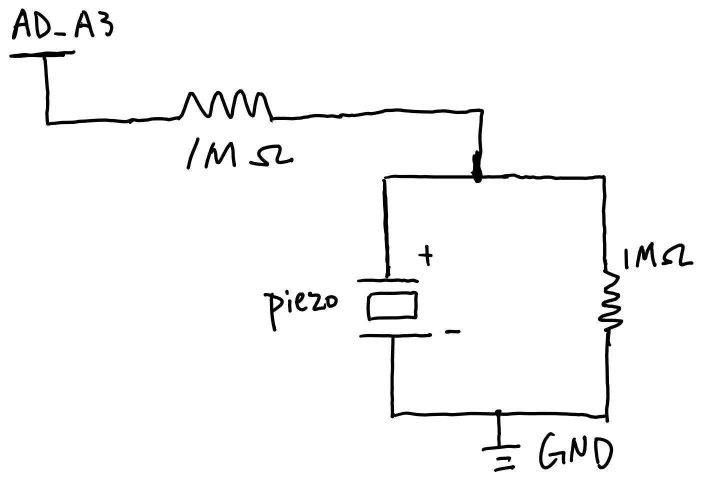


Figure 4: Snubbing circuit for the piezo sensor

I have tested it with the oscilloscope and make sure that the maximum voltage stayed within the maximum range of my power supply, which was also 3.3V. The next step is to make music again with both the piezo sensor and the flex sensor. To my own understanding, the piezo sensor acted basically like a switch. One it was pressed, the speaker would make sound and I can change the tone using the flex sensor. The first thing I did was to print out the reading of the piezo sensor on the IO shield. For my case, with each tap the AD reading would reach 100, so I used this as a threshold—once the AD reading reached 100, the piezo sensor would be considered pressed. Then I reused the code from part 1 but added some more modification. I have placed everything inside of a for-loop—according to the lab manual, the sound should activate for a short time and then stop. Once the piezo sensor was pressed, I would enter the for-loop and start making sounds. If the sensor was pressed again, the counter would be cleared thus extends the time of the tone that is being played.

<u>Result for part 2 & 3: piezo sensor</u>

I was able to produce a clear, smooth sound based on the change of the flex sensor, and the sound would only activate if the piezo sensor is pressed. The sound would stop after a set amount of time (around 1 second) and if the piezo sensor was pressed again during this period, the speaker would continue to make sound.

<u>Part 4: Simple Analog Filter Analysis</u>

In this part we were asked to work with the low-pass filter, high-pass filter, band-pass filter as well as the Sallen-Key filter. We were asked to figure out the transfer function for each filter, use the transfer function to plot the theoretical response and build the actual filter and plot the experimental response.

Low-Pass Filter:

The first one is the low-pass filter. The diagram is already provided in the lab manual, so I'll directly use it here.

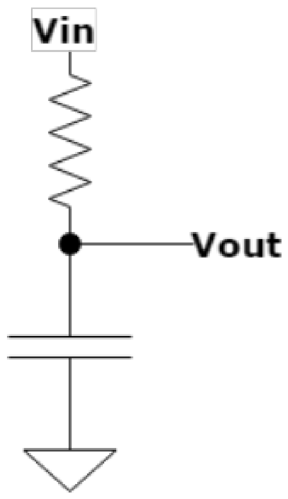
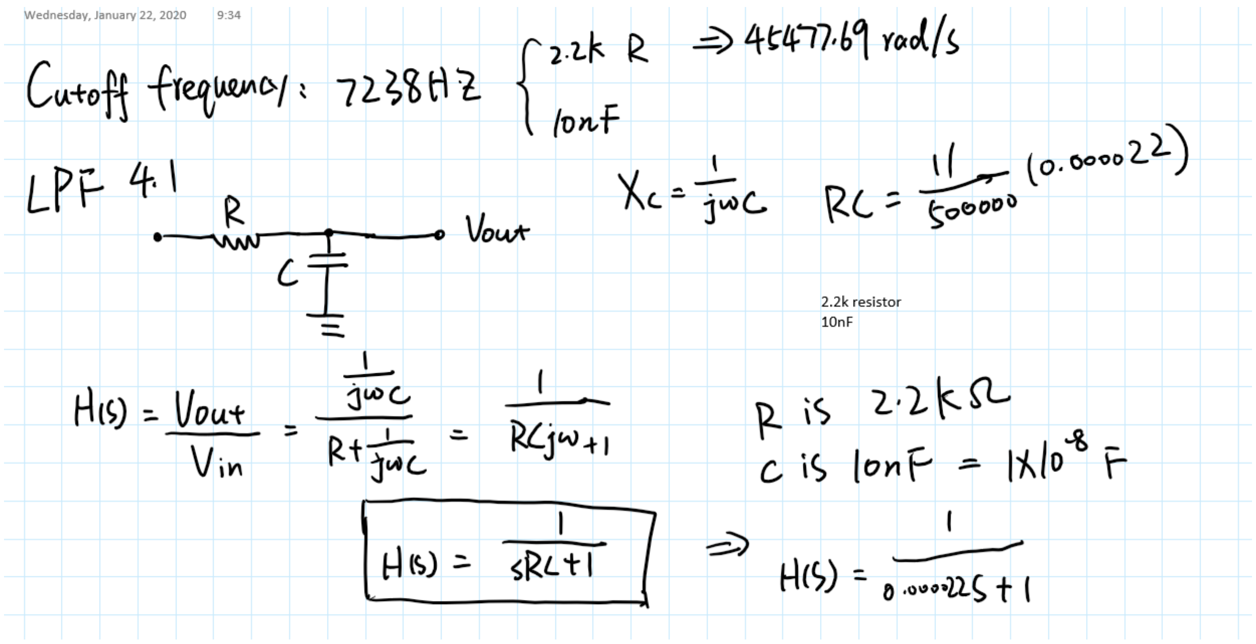


Figure 5: Low-Pass filter

The transfer function is given by the ration between the output vs input ($\frac{Vout}{Vin}$). Figure 6 shows the process of me figuring out the transfer function using the impedance.



Wednesday, January 22, 2020 9:34

Cutoff frequency: 7238HZ $\begin{cases} 2.2k\ R \Rightarrow 45477.69\ rad/s \\ 10nF \end{cases}$

LPF 4.1

$X_c = \frac{1}{j\omega C}$ $RC = \frac{11}{500000}$ (0.000022)

2.2k resistor
10nF

$$H(s) = \frac{Vout}{Vin} = \frac{\frac{1}{j\omega C}}{R + \frac{1}{j\omega C}} = \frac{1}{RCj\omega + 1}$$

R is 2.2kΩ
C is 10nF = 1×10^{-8} F

$$\boxed{H(s) = \frac{1}{sRC+1}} \Rightarrow H(S) = \frac{1}{0.000022S + 1}$$

Figure 6: Transfer function of Low-Pass Filter

The resistance I chose is 2.2kΩ and the capacitance I chose is 10 nF. This gave me a cutoff frequency of 7238Hz, which was equivalent to 45477.69 rad/s. According to the lab manual, I should get a loss of 3dB at cutoff frequency and figure 7 shows that I did get a loss of 3dB at 45900 rad/s(around my cutoff frequency).

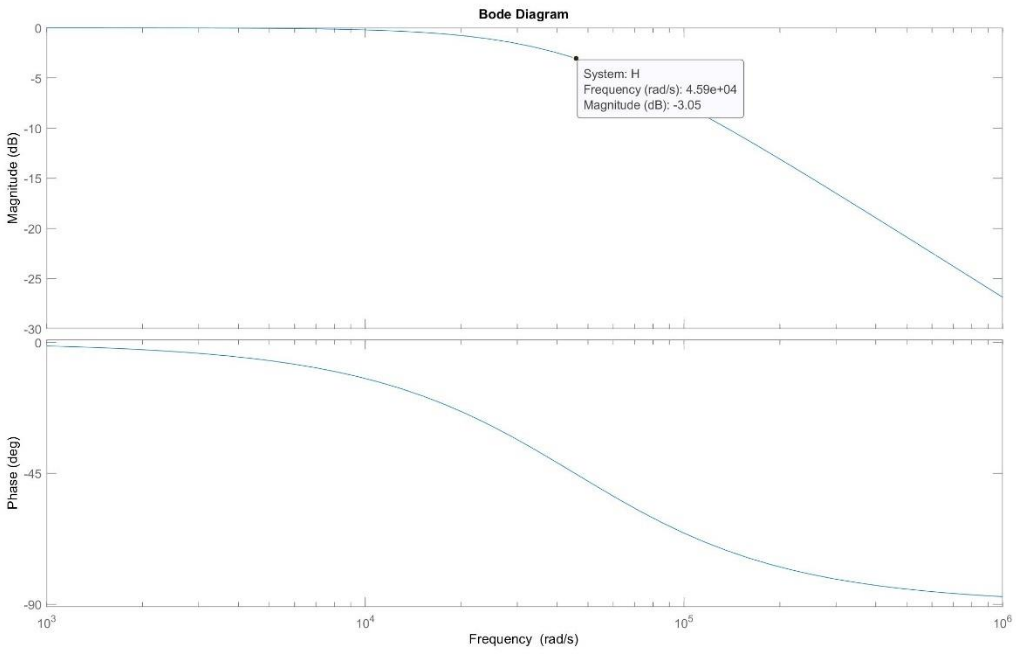


Figure 7: Low-Pass filter theoretical response

Then I went ahead and built the actual circuit and record the data and plot the following graph.

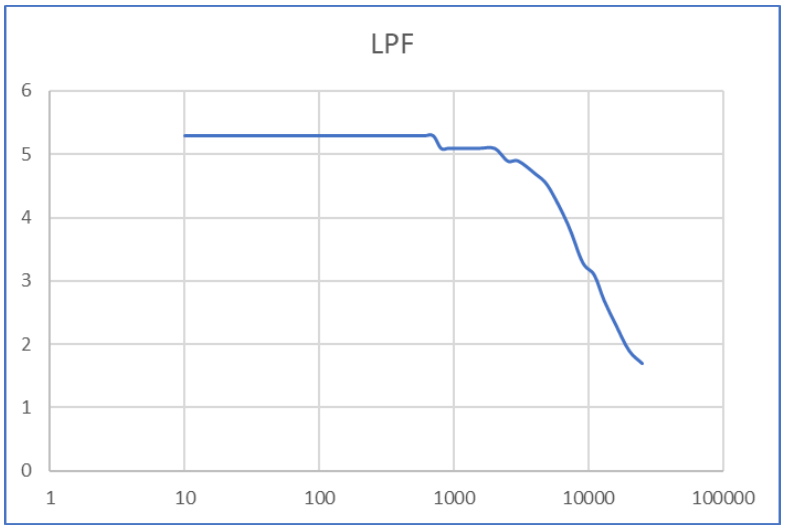


Figure 8: Low-Pass filter experimental response

As it's shown in the graph, it's not as smooth as the theoretical response. This can be caused by many reasons and one of them is the wire. In theoretical response, I didn't need to consider the wire or other components connected to the circuit but in real life, wires do have some resistances, and all the delays can cause a difference in the cutoff frequency as well. I would say my model is somewhat accurate, but not perfect.

High-Pass Filter:

The next filter that we were asked to built is a high-pass filter. The circuit is shown in figure 9.

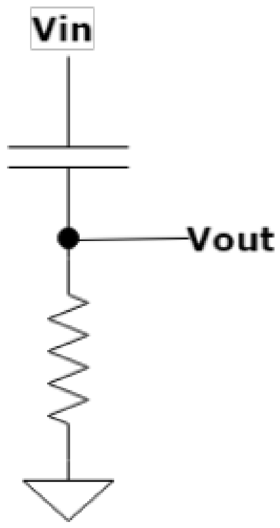
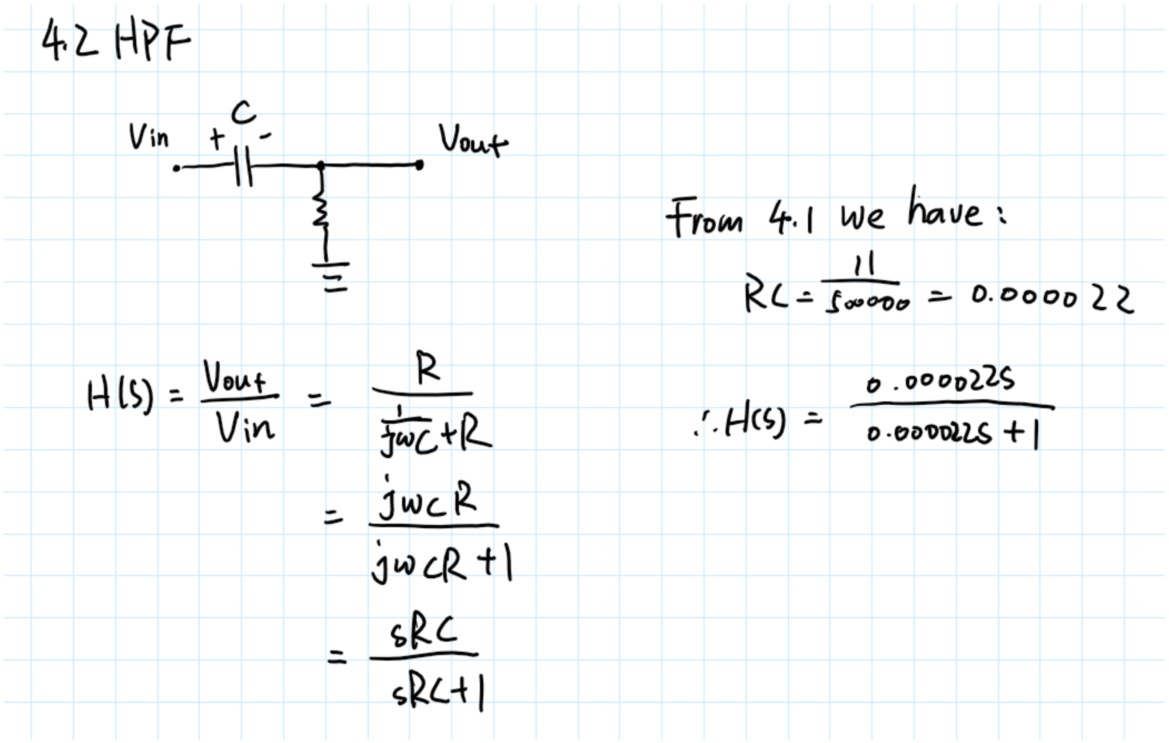


Figure 9: High-Pass Filter

Again, by following the same method for the low-pass filter, this is what I got for the transfer function.

$$H(s) = \frac{sRC}{sRC + 1}$$

Figure 10 provides the process of how I got to the transfer function above and figure 11 provides the theoretical response of the high-pass filter. Then I set up the actual circuit and recorded the data. The experimental graph is shown in figure 12. Again, the experimental response is different from the theoretical response due to various delays or resistance from other components, so I would say it's somewhat accurate. In order to make the graph as smooth as possible, I took about 30 measurements.



4.2 HPF

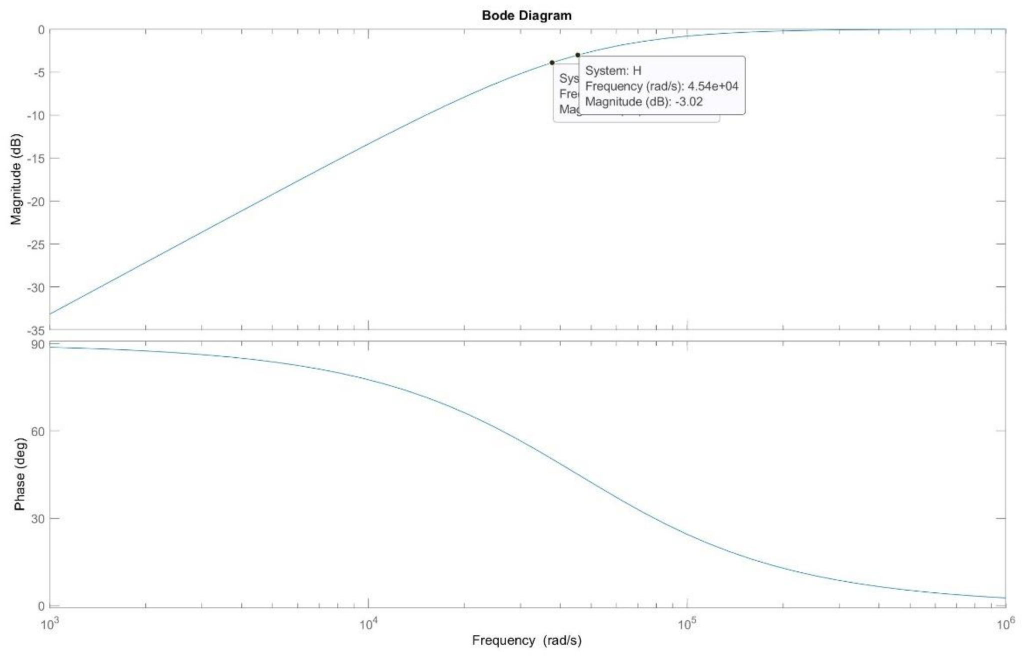
$$H(s) = \frac{V_{out}}{V_{in}} = \frac{R}{\frac{1}{j\omega C}+R} = \frac{j\omega CR}{j\omega CR+1} = \frac{sRC}{sRC+1}$$

From 4.1 we have:

$$RC = \frac{11}{500000} = 0.000022$$

$$\therefore H(s) = \frac{0.000022s}{0.000022s+1}$$

Figure 10: Transfer function for high-pass filter



System: H
Frequency (rad/s): 4.54e+04
Magnitude (dB): -3.02

Figure 12: High-Pass Filter theoretical response

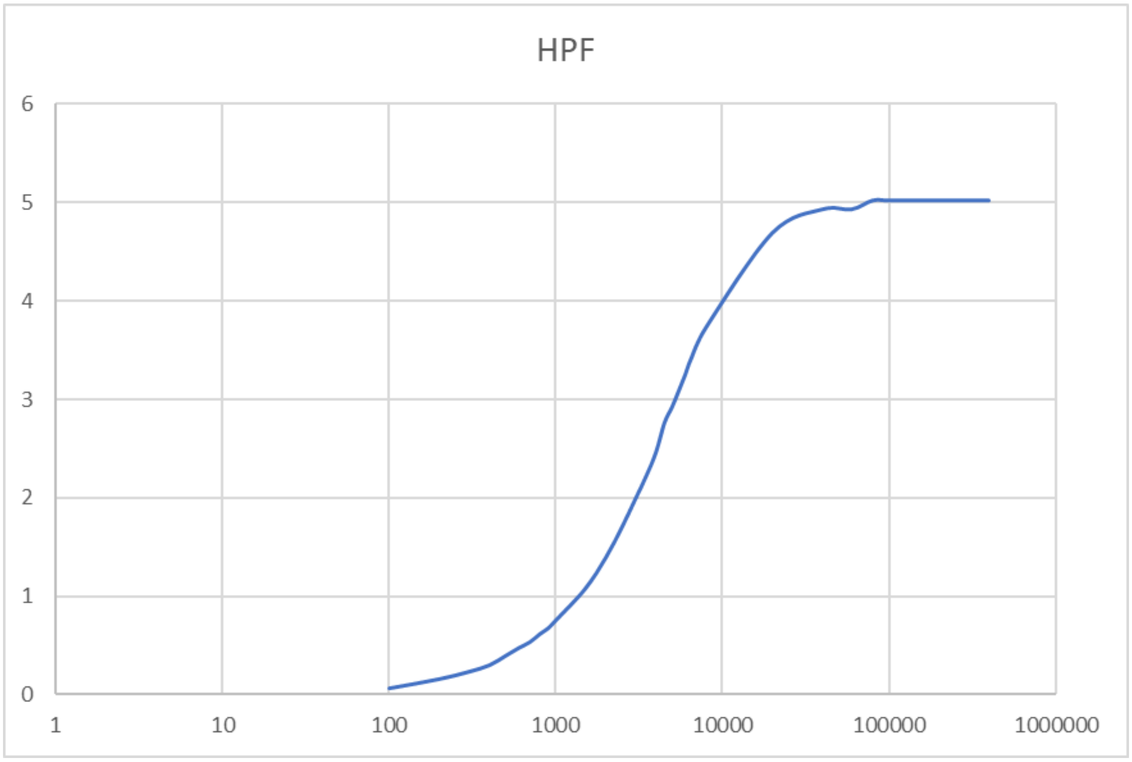


Figure 13: High-Pass Filter experimental response

Band-Pass Filter:

There were two separate steps that we were asked to do in this section. The first thing was to just simply multiply the transfer function of both the low-pass and high-pass filter. Figure 15 shows the process of how I got to the conclusion:

$$H(s) = \frac{(\frac{s}{RC})}{\mathrm{s}^2 + \frac{2\mathrm{s}}{\mathrm{RC}} + (\frac{1}{(\mathrm{RC})^2})}$$

In this function, the quality factor (Q) is ½ and the natural frequency Ѡn is 1/RC.

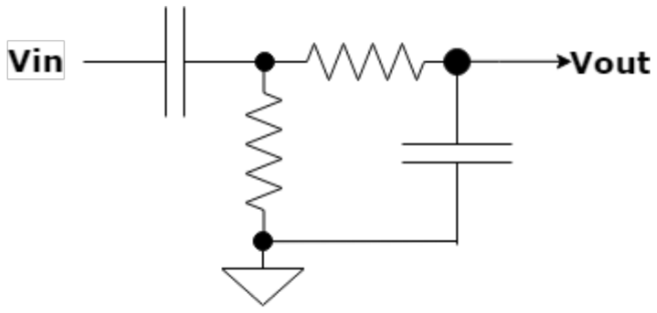
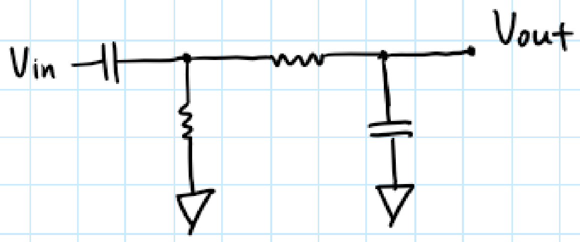


Figure 14: Band-Pass Filter



$$H(s)_{HPF} = \frac{sRC}{sRC+1} \qquad H(s)_{LPF} = \frac{1}{sRC+1}$$

$$H(s)_{HPF} \cdot H(s)_{LPF} = \frac{sRC}{sRC+1} \times \frac{1}{sRC+1}$$

$$= \frac{sRC}{(sRC+1)^2}$$

$$= \frac{sRC}{s^2R^2C^2+2sRC+1} \Rightarrow \frac{\frac{\omega_n}{Qs}}{s^2+\frac{\omega_n}{Qs}+\omega_n^2}$$

$$= \frac{\frac{s}{RC}}{s^2+\frac{2s}{RC}+\frac{1}{(RC)^2}}$$

$$\omega_n^2 = \left(\frac{1}{RC}\right)^2 \Rightarrow \boxed{\omega_n = \frac{1}{RC}}$$

$$\frac{2}{RC}s = \frac{\omega_n}{Q}s \Rightarrow \omega_n = \frac{1}{RC} \Rightarrow \boxed{Q = \frac{1}{2}}$$

Figure 15: Transfer function of Band-Pass filter

After using Matlab, figure 16 is the theoretical response I got for the theoretical response of the band-pass filter obtained by multiplying two transfer function together.

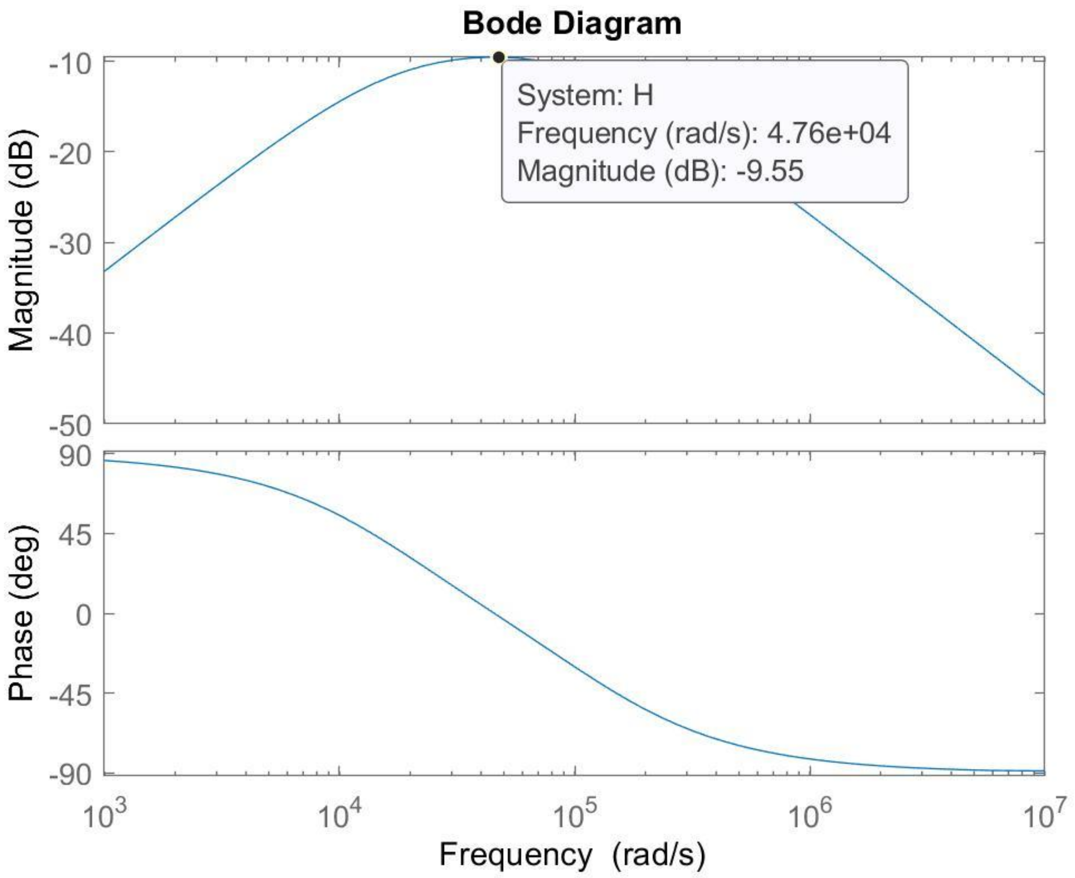


Figure 16: Band-Pass filter theoretical response

To physically test the band-pass filter, we can't just simply connect two filter together. To achieve this, we were introduced a new component—MCP-6004 OpAmp. This OpAmp was used as a buffer so that I could use the output of the high-pass filter as the input of the low-pass filter. The schematic is provided in figure 17 below.

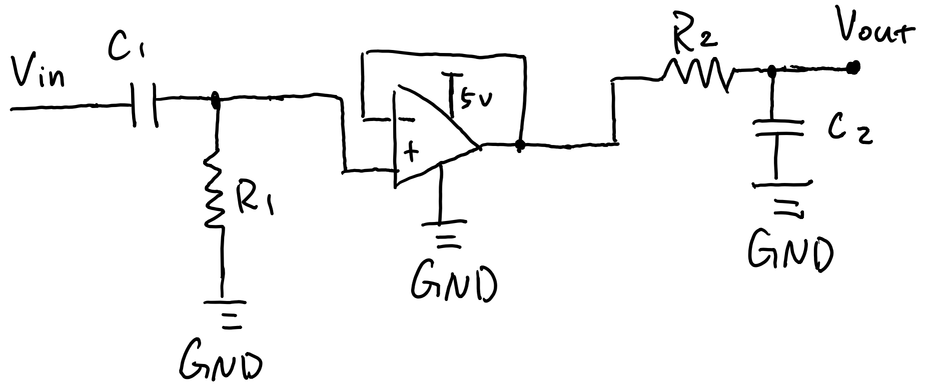


Figure 17: Experimental Band-Pass filter

In this circuit, all capacitors were 10nF and all resistors were 2.2kΩ. One thing to mention is that when powering the OpAmp, a DC bias should be added since we don't want a negative input to be sent to the OPAmp. Another thing is that input, output as well as power should be connected to a different ground instead of the common ground rail on the bread board. Once I have everything setup, I connected Vin with the function generator and power the OpAmp with 5V and an offset of 1.5V and measure the output using the oscilloscope. Figure 18 shows the plot of the experimental response.

Bandpass Filter Experiment

Frequency(Hz)	Magnitude (V)
100	0.04
500	0.177
1000	0.33
2000	0.563
3000	0.72
5000	0.84
10000	0.84
20000	0.68
30000	0.52
50000	0.36
100000	0.2
200000	0.12
300000	0.08
900000	0.04

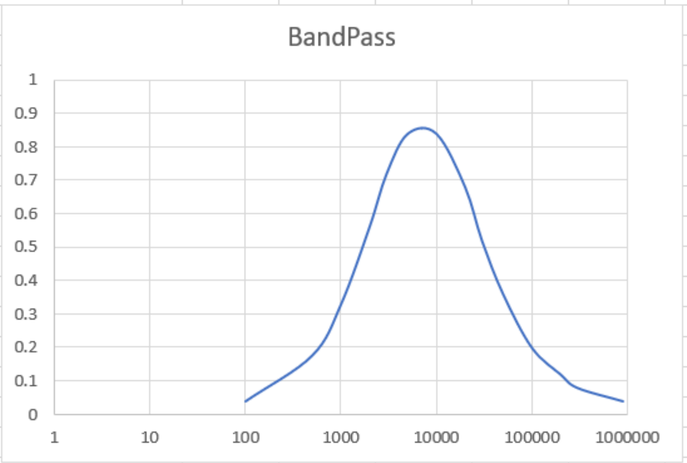
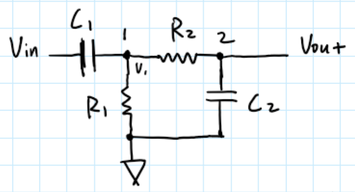


Figure 18: Experimental response of band-pass filter

Finally, in order to obtain the actual transfer function of the band-pass filter shown in figure 14, we had to do circuit analysis, mainly KCL for each node. It was an extremely long process and I'll attach my process below.



Node 1:

$$Z_c = \frac{V}{I} \Rightarrow i = \frac{V}{Z_c} = \frac{V}{\frac{1}{j\omega C}}$$

$$\Rightarrow i_{c_1} = \frac{V_1 - V_{in}}{\frac{1}{j\omega C_1}}$$

Current passing resistors $\begin{cases} i_{R_1} = \frac{V_1}{R_1} \\ i_{R_2} = \frac{V_1 - V_{out}}{R_2} \end{cases}$

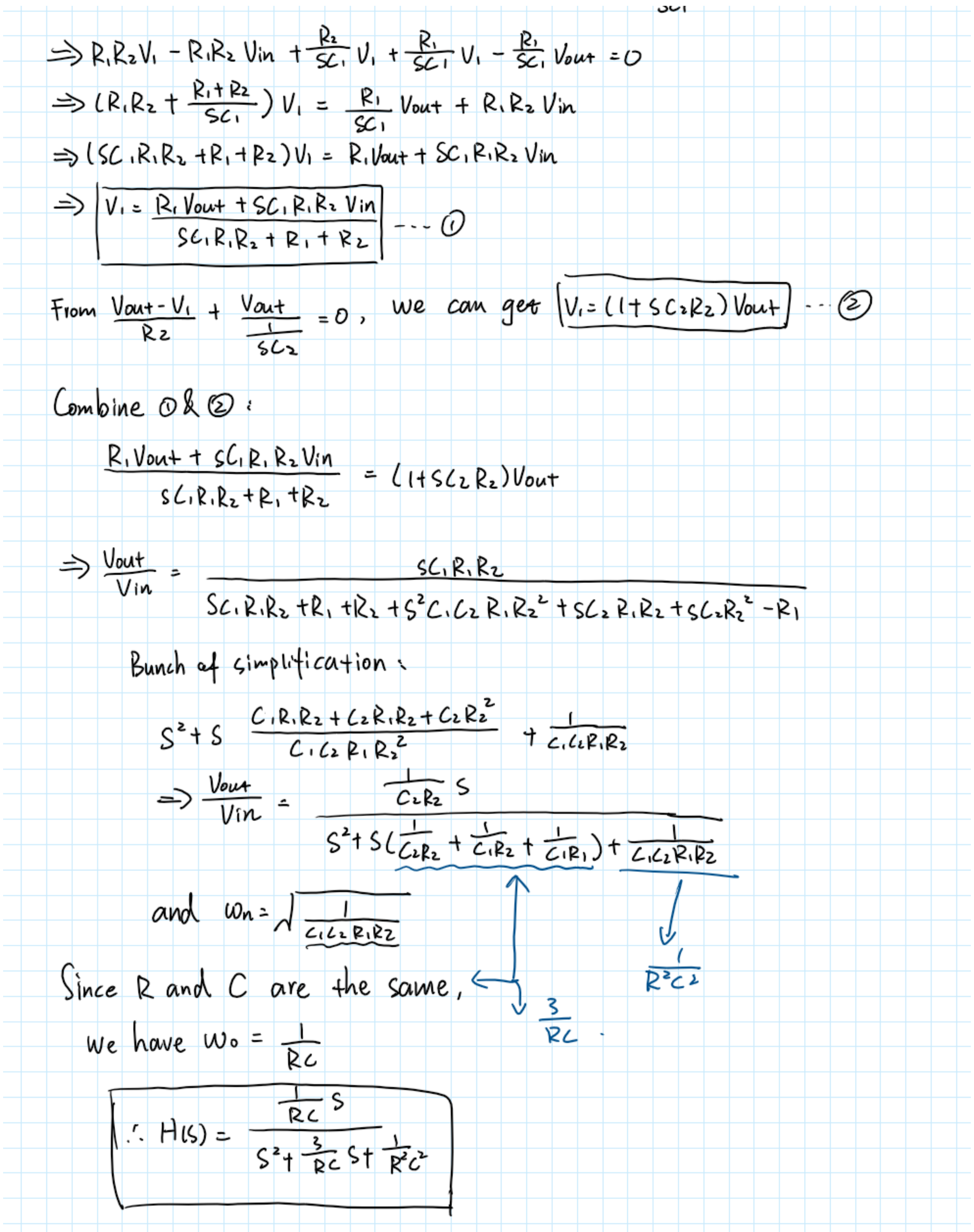
$$\frac{V_1 - V_{in}}{\frac{1}{j\omega C_1}} + \frac{V_1}{R_1} + \frac{V_1 - V_{out}}{R_2} = 0 \quad (\text{KCL})$$

Node 2:

$$i_{R_2} = \frac{V_{out} - V_1}{R_2}, \quad i_{C_2} = \frac{V_{out}}{\frac{1}{j\omega C_2}}$$

$$\frac{V_{out} - V_1}{R_2} + \frac{V_{out}}{\frac{1}{j\omega C_2}} = 0$$

$$\begin{cases} \frac{V_1 - V_{in}}{\frac{1}{j\omega C_1}} + \frac{V_1}{R_1} + \frac{V_1 - V_{out}}{R_2} = 0 \\ \frac{V_{out} - V_1}{R_2} + \frac{V_{out}}{\frac{1}{j\omega C_2}} = 0 \\ \frac{V_1 - V_{in}}{\frac{1}{sC_1}} + \frac{V_1}{R_1} + \frac{V_1 - V_{out}}{R_2} = 0 \end{cases}$$



$\Rightarrow R_1R_2V_1 - R_1R_2V_{in} + \frac{R_2}{sC_1}V_1 + \frac{R_1}{sC_1}V_1 - \frac{R_1}{sC_1}V_{out} = 0$

$\Rightarrow \left(R_1R_2 + \frac{R_1+R_2}{sC_1}\right)V_1 = \frac{R_1}{sC_1}V_{out} + R_1R_2V_{in}$

$\Rightarrow (sC_1R_1R_2 + R_1 + R_2)V_1 = R_1V_{out} + sC_1R_1R_2V_{in}$

$\Rightarrow \boxed{V_1 = \frac{R_1V_{out} + sC_1R_1R_2V_{in}}{sC_1R_1R_2 + R_1 + R_2}} \cdots ①$

From $\frac{V_{out}-V_1}{R_2} + \frac{V_{out}}{\frac{1}{sC_2}} = 0$, we can get $\boxed{V_1 = (1+sC_2R_2)V_{out}} \cdots ②$

Combine ① & ②:

$$\frac{R_1V_{out} + sC_1R_1R_2V_{in}}{sC_1R_1R_2 + R_1 + R_2} = (1+sC_2R_2)V_{out}$$

$$\Rightarrow \frac{V_{out}}{V_{in}} = \frac{sC_1R_1R_2}{sC_1R_1R_2 + R_1 + R_2 + s^2C_1C_2R_1R_2^2 + sC_2R_1R_2 + sC_2R_2^2 - R_1}$$

Bunch of simplification:

$$s^2 + s\,\frac{C_1R_1R_2 + C_2R_1R_2 + C_2R_2^2}{C_1C_2R_1R_2^2} + \frac{1}{C_1C_2R_1R_2}$$

$$\Rightarrow \frac{V_{out}}{V_{in}} = \frac{\frac{1}{C_2R_2}s}{s^2 + s\underbrace{\left(\frac{1}{C_2R_2} + \frac{1}{C_1R_2} + \frac{1}{C_1R_1}\right)}_{\frac{3}{RC}} + \underbrace{\frac{1}{C_1C_2R_1R_2}}_{\frac{1}{R^2C^2}}}$$

and $\omega_n = \sqrt{\frac{1}{C_1C_2R_1R_2}}$

Since R and C are the same,

we have $\omega_0 = \frac{1}{RC}$

$$\boxed{\therefore H(s) = \frac{\frac{1}{RC}s}{s^2 + \frac{3}{RC}s + \frac{1}{R^2C^2}}}$$

Figure 19: Theoretical Transfer function of band-pass filter by analysis

Sallen-Key Filter:

This is the last filter that we were asked to analyze. Figure 20 shows the diagram of the Sallen-Key filter (taken from the lab manual). By doing KCL on each of the node, I managed to figure out the transfer function:

$$H(s) = \frac{(\frac{s}{RC})}{s^2 + \frac{3s}{RC} + (\frac{2}{(RC)^2})}$$

It looks similar to the band-pass filter, but the layout is different. The quality factor (Q) is ¼ and the natural frequency Ϣn is $\frac{\sqrt{2}}{RC}$. By using the transfer function above, I was able to plot the theoretical response using Matlab (see figure 21). I have to be honest that I'm not quite sure what's the major difference between Sallen-Key and the band-pass filter that I have built above—both graphs look similar to each other except the readings are different.

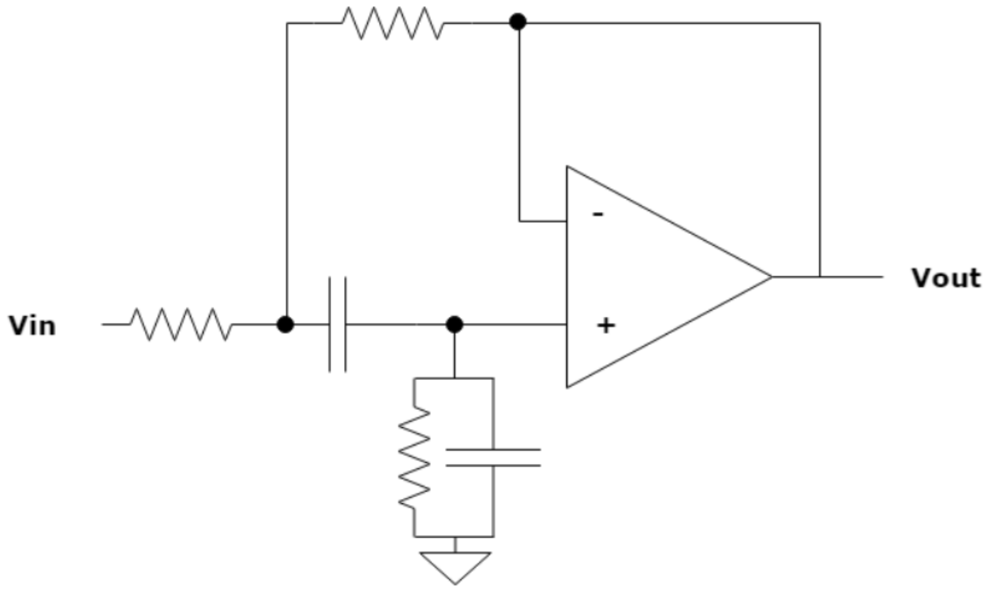


Figure 20: Sallen-Key filter

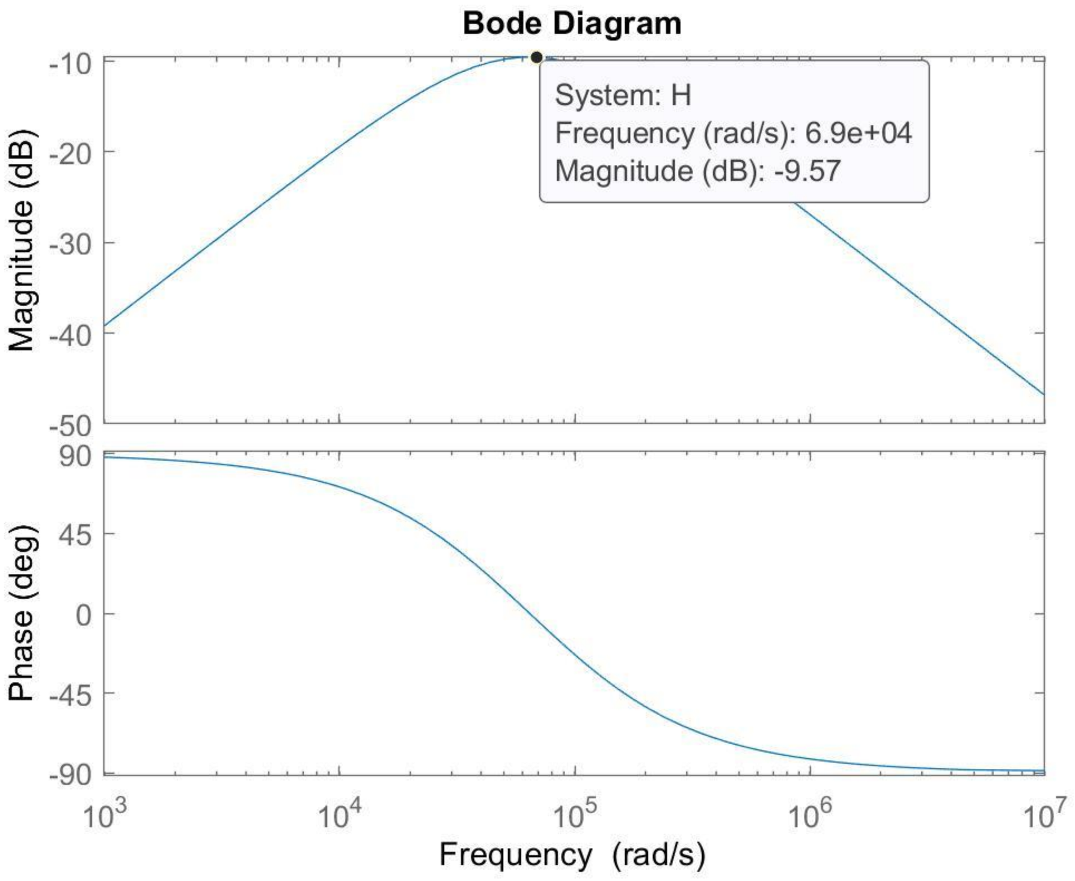
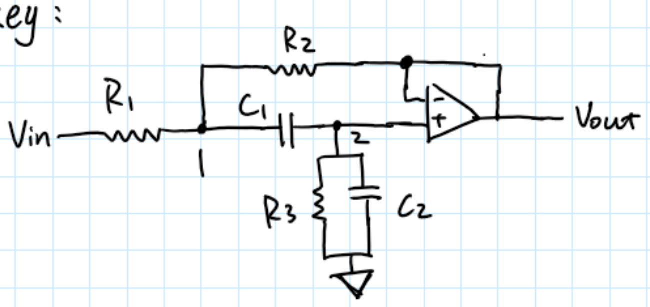


Figure 21: Sallen-Key filter theoretical plot

Sallen-key :



At node 1:

$$\frac{V_1 - V_{in}}{R_1} + \frac{V_1 - V_{out}}{R_2} + \frac{V_1 - V_{out}}{\frac{1}{sC}} = 0$$

$$\Rightarrow \frac{2V_1 - V_{in} - V_{out}}{R} + \frac{V_1 - V_{out}}{\frac{1}{sC}} = 0$$

$$\Rightarrow 2V_1 - V_{in} - V_{out} + sRCV_1 - sRCV_{out} = 0$$

$$\Rightarrow V_1 = \frac{(1+sRC)V_{out} + V_{in}}{2+sRC}$$

At node 2:

$$\frac{V_{out} - V_1}{\frac{1}{sC_1}} + \frac{V_{out}}{R_3} + \frac{V_{out}}{\frac{1}{sC_2}} = 0$$

$$\Rightarrow RV_{out} - RV_1 + \frac{V_{out}}{sC} + RV_{out} = 0$$

$$\Rightarrow (1 + 2sRC)V_{out} = sRCV_1$$

$$\Rightarrow V_1 = \frac{(1+2sRC)V_{out}}{sRC}$$

$$\Rightarrow sRC(1+sRC)V_{out} + sRCV_{in} = (1+2sRC)V_{out}(2+sRC)$$

$$sRCV_{in} = (2 + 5sRC + 2s^2R^2C^2 - sRC - s^2R^2C^2)V_{out}$$

$$\boxed{\Rightarrow \frac{V_{out}}{V_{in}} = \frac{\frac{1}{RC}\cdot s}{s^2 + \frac{3}{RC}s + \frac{2}{R^2C^2}}}$$

Figure 22: Transfer function for Sallen-Key filter

<u>Conclusion:</u>

I would like to call this lab a refresher of what I have learned in both EE101 and EE103 since I’ve almost forgot everything. More importantly, I get to see what the actual low/high pass filters respond to difference frequencies. I definitely gain more understanding in those filters than before, so I really like this lab. The coding part is fun since I get to make my own music again using the new components.

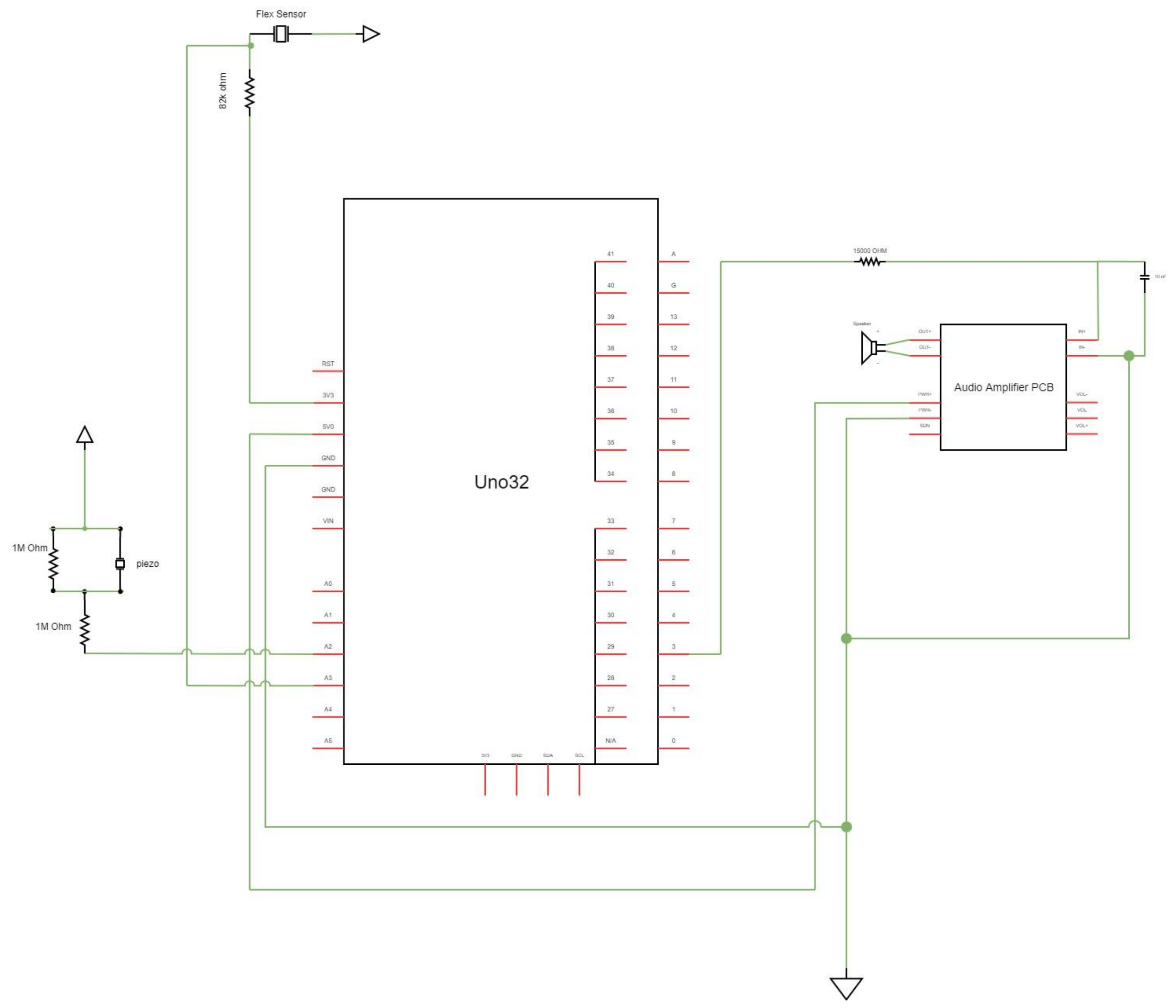


Figure 23: External wiring diagram