Leonid Shuster

ECE 167

1/26/20

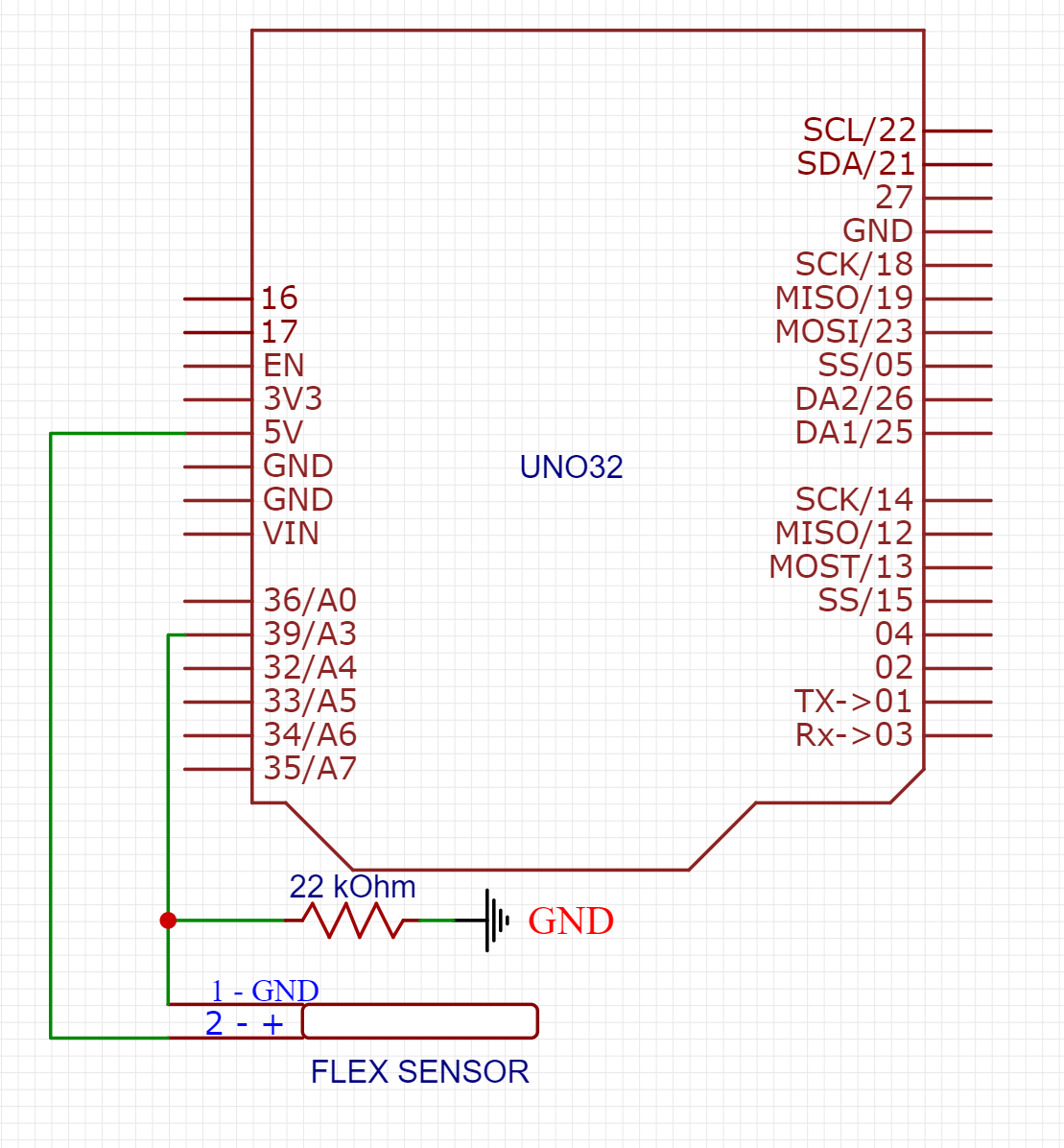
Lab Report 1

**Introduction**

In this lab, we were tasked with learning how to use resistive sensors such as flex and piezo sensors, as well as modelling filters and then experimentally validating them. In the first part, we linearized a flex sensor and then used it to produce a tone based on its flex. In the second part, we set up a piezo sensor that was used to play a tone when its was flicked. In the third part, we used both the flex and piezo sensors for a musical instrument that would play a tone for a certain amount of time based on the flex of the flex sensor when the piezo sensor was flicked. And in the fourth part, we created models of various simple filters and then later validated those models.

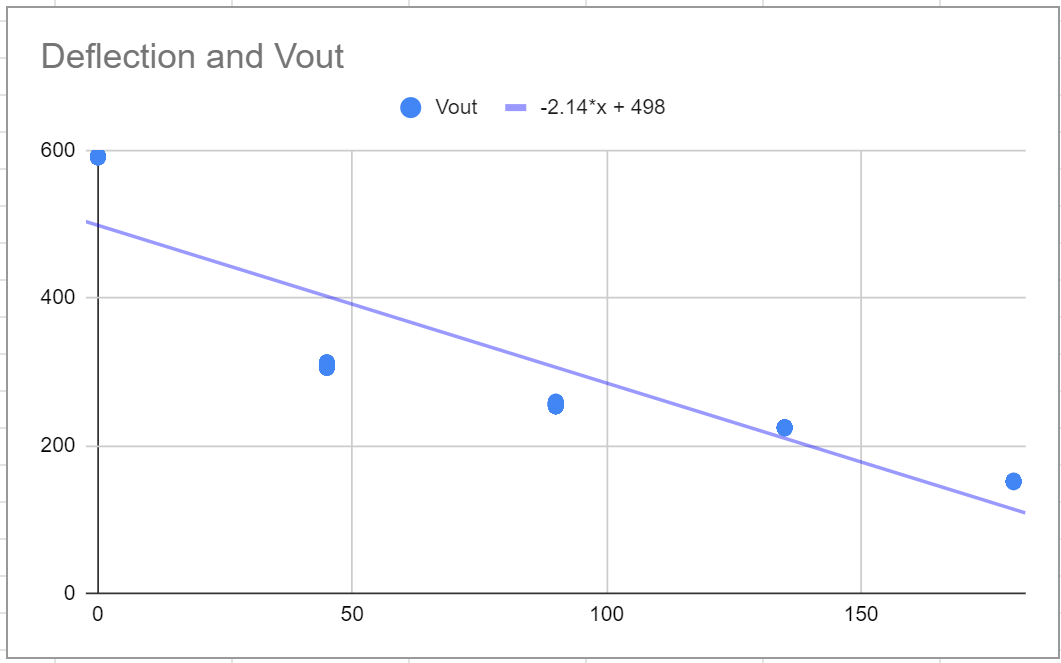
**Part 1: Flex Sensor**

In the first part of the lab, we were told to use a flex sensor to play a tone based on the flex of the flex sensor. After assembling the flex sensor, the first thing we had to do was linearize it, as the flex sensor was nonlinear by nature. To do so, we had to create a transfer function that mapped a linear relation between the input and output of it, where the input was the angle of flex and the output was the output voltage of the flex sensor. I integrated the flex sensor into a circuit in which the flex sensor and a 22 kiloOhm resistor acted as resistors in a voltage divider circuit, and then read the output voltage of it by using one of the analog pins on the Uno32 (see Figure 1).



*Figure 1: Part 1 Flex Sensor Setup*

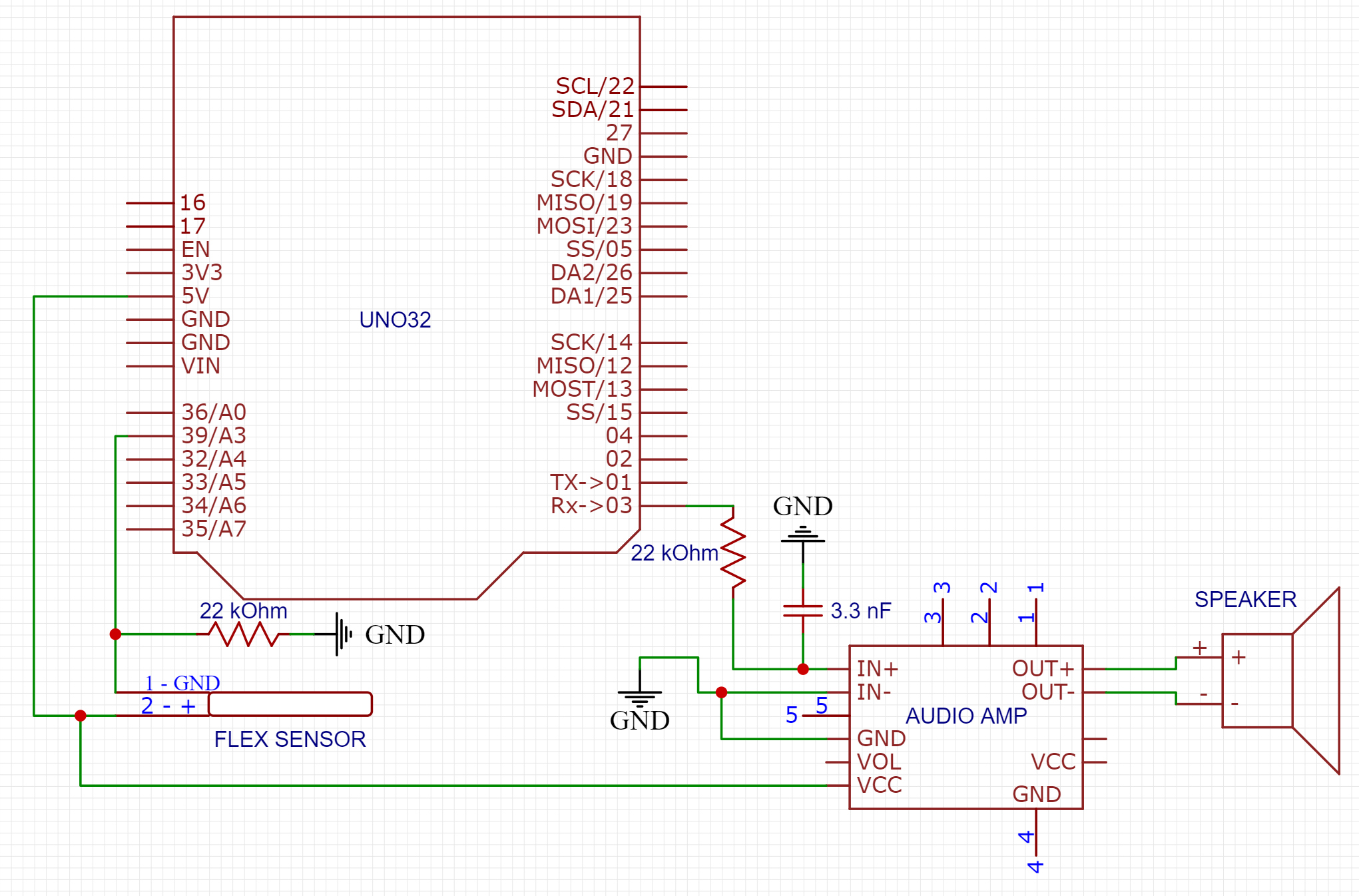
I then took many data samples of the output voltage as the flex changed and recorded the values in Google Sheets. Next, I plotted flex vs. output voltage, and used Google Sheets’s builtin trendline feature to produce a linear equation and thus a transfer function (see Figure 2).



*Figure 2: Part 1 Linear Transfer Function*

The transfer function used flex as the independent variable and output voltage as the dependent variable, but in order for flex to later be used to set the frequency of a tone, I had to find the inverse of the transfer function so that flex would become the dependent variable.

In Lab1\_Part1.3Main.c, I read from the analog pin connected to the flex sensor’s output voltage, plugged that value into the inverse transfer function in order to get the angle of flex, checked to make sure the angle stayed within bounds of 0-180 degrees, and then scaled the angle to a range of 0-1023 that would then be used to set the frequency of the tone being produced. On the outside of the Uno32, I also connected the audio amp and speaker along with a low-pass filter (see Figure 3).

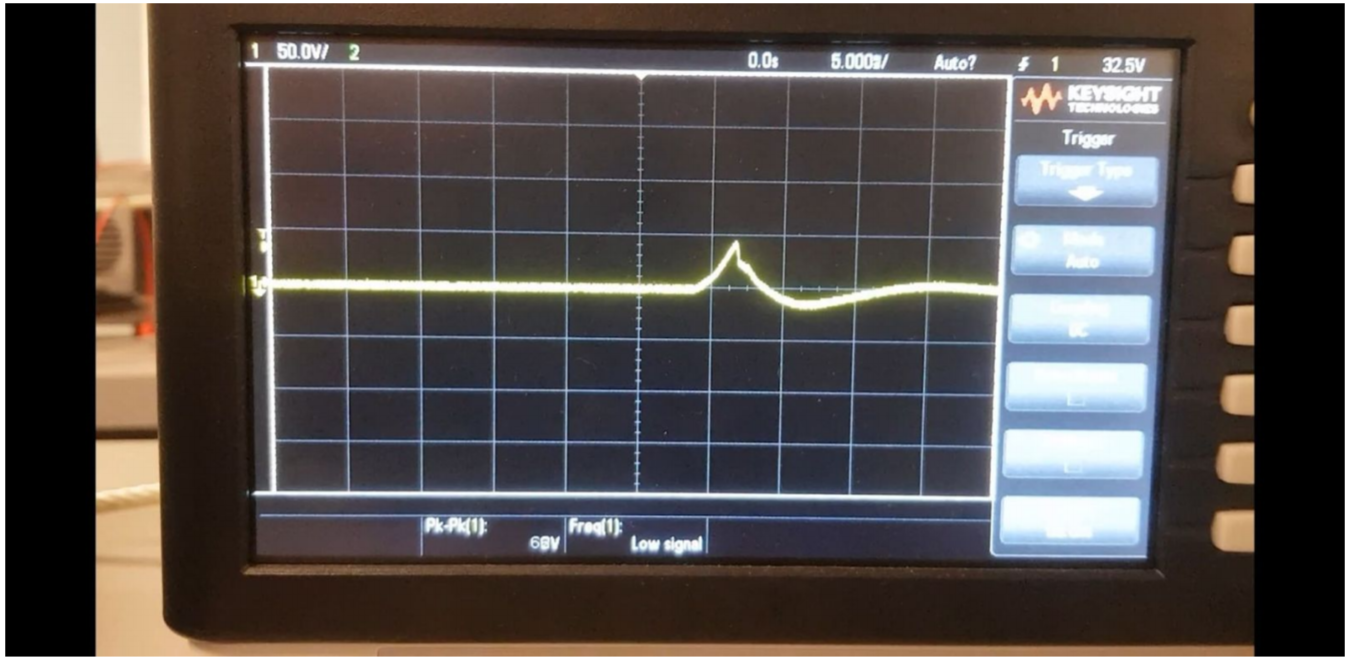


*Figure 3: Part 1 External Schematic*

When I listened to the tone of the speaker, I noticed that the change in frequency from the speaker was a lot more fluid when I changed the flex of the sensor, which meant I had linearized it correctly. However, the sound from the speaker was still kind of scratchy, even though I had implemented filtering. In addition, everytime I took out my flex sensor and put it back in the circuit, I noticed that the values I was getting from the output voltage changed every time. This made it difficult to use a single transfer function; I had to change it several times.

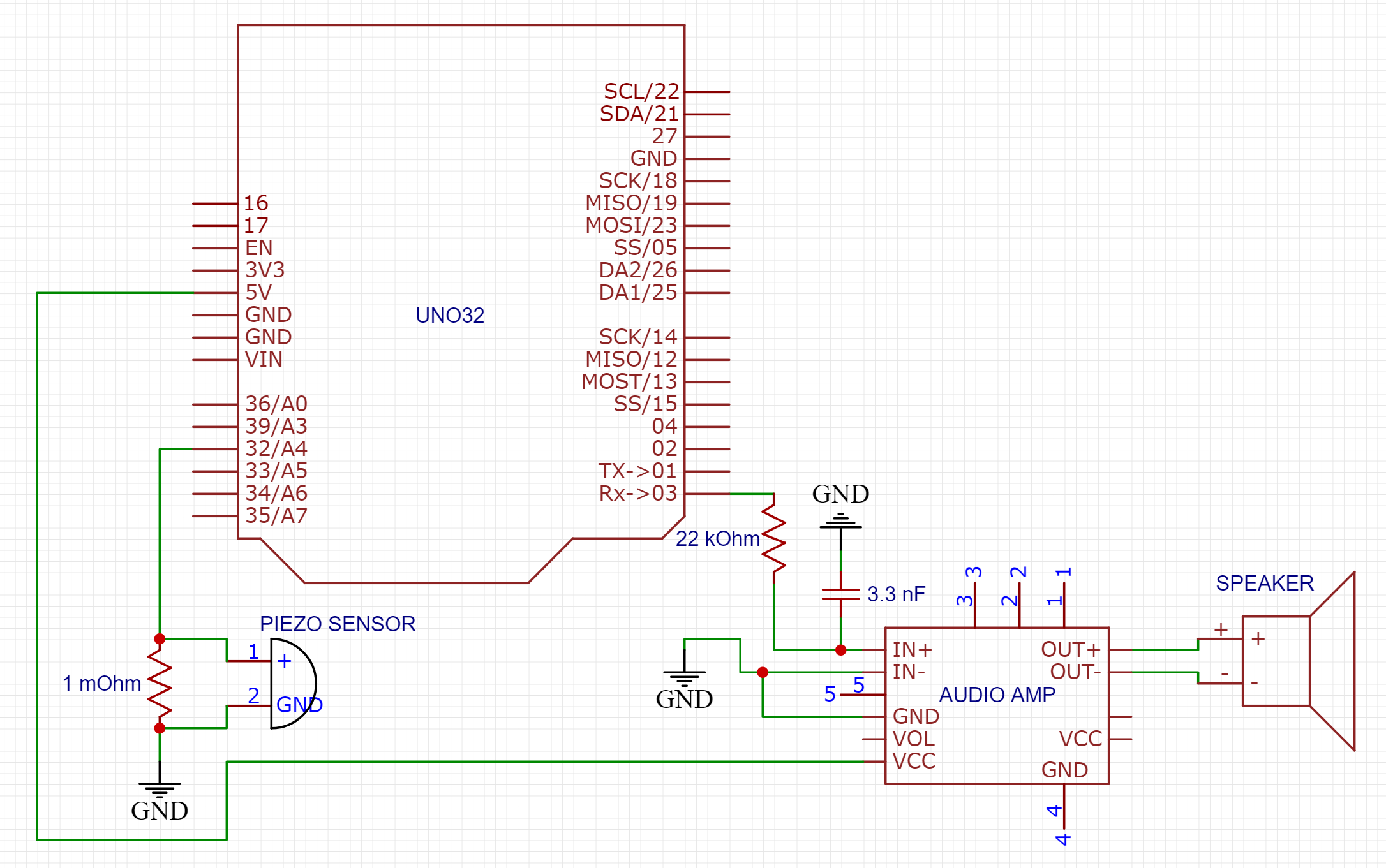
**Part 2: Piezoelectric Sensor**

In the next part of the lab, we were told to use a piezoelectric sensor to trigger a tone to play when flicked. The first thing to do was to verify that the piezo sensor produced a voltage spike when pressed. I hooked up the piezo sensor to the oscilloscope, flicked it, and observed that there was indeed a voltage spike (see Figure 4).



*Figure 4: Part 2 Piezo Sensor Voltage Spike*

The next thing to do was to produce a tone when the piezo sensor was flicked. Because the piezo sensor produced a high voltage spike and could damage the pins on the Uno32, we had to snub the voltage coming from the piezo sensor by using a 1 megaOhm resistor. In my circuit, I connected the positive side of the piezo sensor to one side of the resistor and an analog pin on the Uno32, and the negative side to the other side of the resistor and to ground. I also added the audio amp and speaker as well as the low-pass filter (see Figure 5).

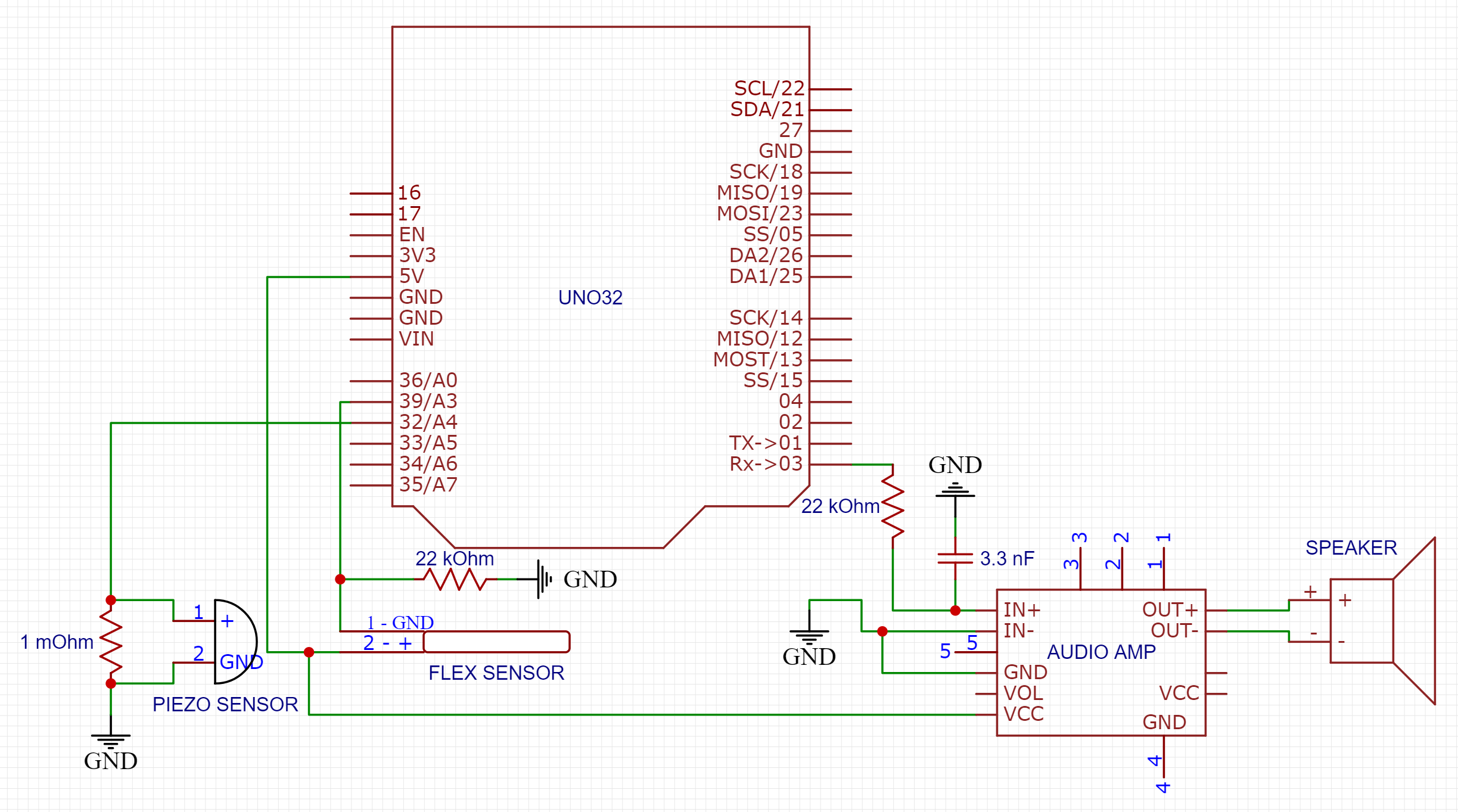


*Figure 5: Part 2 External Schematic*

In Lab1\_Part2.1Main.c, I read from the analog pin connected to the piezo sensor by checking for a value greater than 100 from the voltage spike when pressed, and if so played a tone for a certain amount of time by creating a delay function from an idle while loop.

**Part 3: Musical Instrument Redux**

In the next part of the lab, we were told to create a musical instrument in which a tone would be played for a certain amount of time based on the flex of the flex sensor when the piezo sensor was flicked, and that if the piezo sensor was pressed again while the tone was playing, the duration of time it would play for would be reset. The first thing I did was create a circuit that combined everything from the previous parts so that it had the flex and piezo sensors, audio amp with low-pass filter, and speaker (see Figure 6).



*Figure 6: Part 3 External Schematic*

In Lab1\_Part3Main.c, I read from both the flex and piezo sensors, used the inverse transfer function from before to linearize the flex sensor and get its flex, scale the range to 0-1023 for tone frequencies, and then checked if the piezo sensor as pressed. If it was, the scaled flex value from the flex sensor would be played as a tone, and then stay on for a certain amount of time by entering an idle while loop that kept reading from the piezo sensor and would reset if the piezo sensor was pressed again.

One problem I ran into was getting the duration of the time of the tone being played to reset if the piezo sensor was pressed again. The piezo sensor’s value stayed higher than my threshold for longer than just one cycle when it was pressed, so it played a tone for much longer than I wanted it to, and sometimes even got stuck in the idle loop because it kept resetting thinking it was being pressed. I fixed this by continuously checking the value coming from the piezo sensor in my while loop, and reset the loop only if the value updated.

**Part 4: Simple Analog Filtering Analysis**

In the last part of the lab, we were tasked with creating mathematical models of several simple bandpass filters, and validating them, followed by modelling and testing a more complicated active filter later on.

The first filter that we had to model was a low-pass filter. After using KCL and complex impedances, I found the transfer function to be where s = jw. I then choose the corner frequency to be around 2000 Hz with a 22 kiloOhm resistor and a 3.3 nF capacitor, and plotted the theoretical magnitude vs. frequency curve (see Figure 7).



*Figure 7: Part 4 Theoretical Low-pass Filter Response*

The next filter we had to model was a high-pass filter. Again using KCL and complex impedances, I found the transfer function to be where s = jw. With the same corner frequency, I then plotted the theoretical magnitude vs. frequency curve (see Figure 8).



*Figure 8: Part 4 Theoretical High-pass Filter Response*

The next filter we had to model was a band-pass filter, which could be done by putting a high-pass and low-pass filter together, where R1 and C1 belonged to the high-pass filter and R2 and C2 to the low-pass one. To find the transfer function of the band-pass filter, you could multiply the transfer functions of a high-pass and low-pass filter, which I found to bewith the natural frequency (ωn ) being 1 and the quality factor (Q) equal to R1C1. To verify this, I also did KCL and complex impedances on a band-pass filter schematic and got with the natural frequency (ωn ) also being 1 and the quality factor (Q) equal to R2C2 . I also plotted the theoretical magnitude vs. frequency curve (see Figure 9).

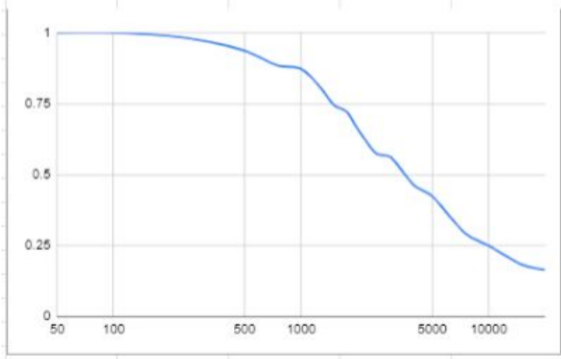


*Figure 9: Part 4 Theoretical Band-pass Filter Response*

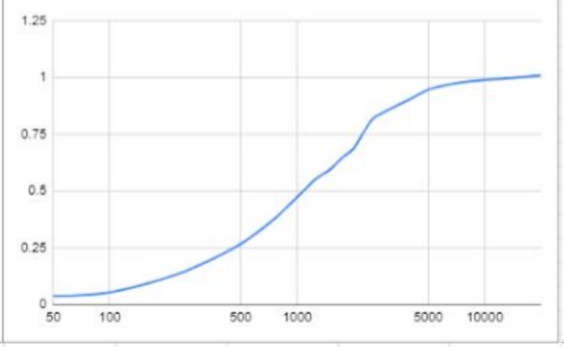
For the Sallen-Key active band-pass filter, I found the transfer function to be

with the natural frequency (ωn ) being C1 and the quality factor (Q) equal to R1.

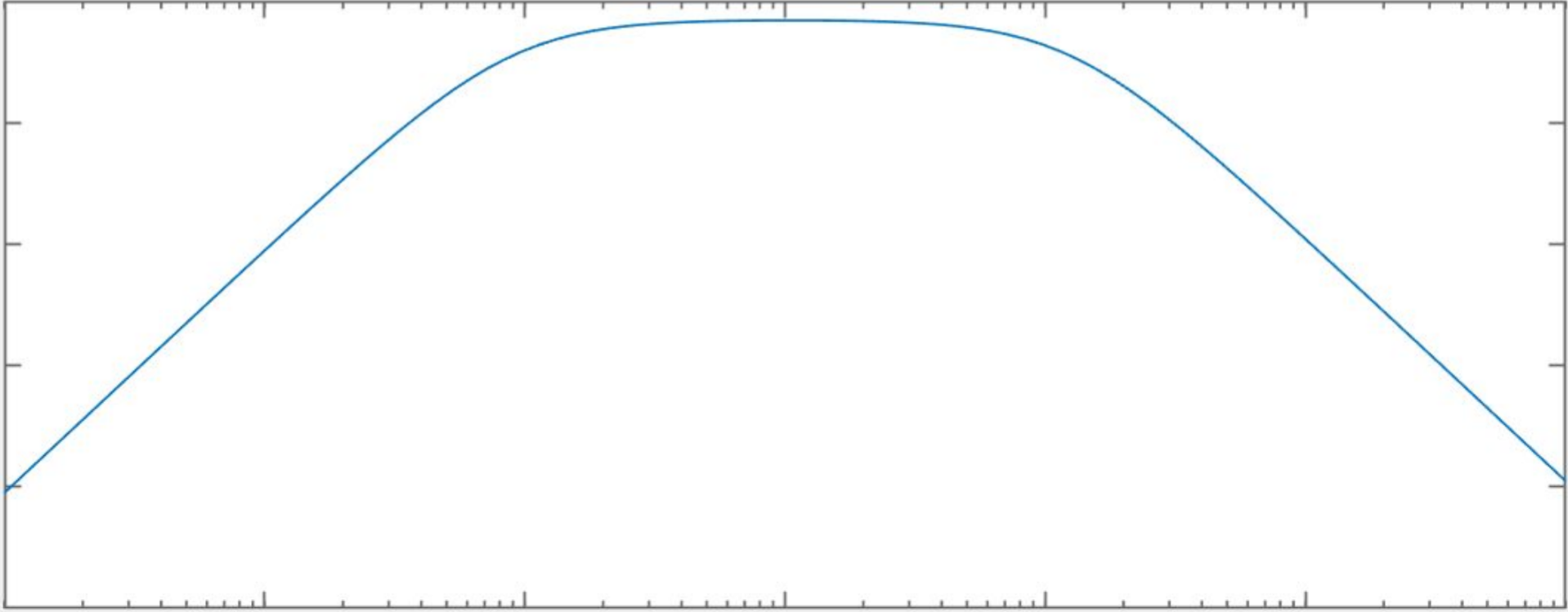
Finally, we validated our functions and plots by building these various filters, getting data, and then plotting them. For the low-pass, high-pass, and band-pass filters, the plots were what I had expected (see Figures 10-12).



*Figure 10: Part 4 Low-pass Filter Recorded Response*



*Figure 11: Part 4 High-pass Filter Recorded Response*



*Figure 12: Part 4 Band-pass Filter Recorded Response*

**Conclusion**

After having gone through the lab, I feel like I fundamentally understand how to use the flex and piezo sensors, as well as how simple filters look modelled versus how they actually turn out to be. From using a flex sensor in the first part to play a tone based on its flex, playing a tone when a piezo sensor is flicked in the second part, using the flex and piezo sensors to create a musical instrument that plays a tone for a certain amount of time based on the flex of the flex sensor when the piezo sensor is flicked in the third part, and modelling filters and validating them in the fourth part, I feel confident that I can effectively use the flex and piezo sensors, as well as model filters and test them. If I were to do this lab again, I would spend more time on understanding how the more advanced filters work, since I didn’t get enough time to experiment with them.