Course Name: Signals and Systems

Engineering Department

University of Massachusetts, Boston

Semester: Fall 2022

Instructor: Dr. Michael Rahaim

Project Report for:

Signal Analysis

Names: Camille Baptiste, Miguel Chang, Raymond Christensen, Brandon Tran, Nick Yu

Student IDs: 01893778, 01855343, 01920722, 01921070, 01892184

Date: December 16, 2022

I pledge to uphold the governing principles of the Code of Student Conduct of the University of Massachusetts Boston. I will refrain from any form of academic dishonesty or deception, cheating, and plagiarism. I pledge that all the work submitted here is my own, and that I have clearly acknowledged and referenced other people's work. I am aware that it is my responsibility to turn in other students who have committed an act of academic dishonesty; and if I do not, then I am in violation of the Code. I will report to formal proceedings if summoned.

Signatures (Initials): CB, MC, RC, BT, NY

CONTENTS

PROJECT OVERVIEW

Our project goal is to analyze and compare the similarities and differences between two signals. The audio that is tested comes from World 1 Level 1 of the New Super Mario Bros. game on the Nintendo 3DS. One of the signals is going to be a very clean downloaded version of the audio. The other is going to be a recording of the level being played on a Nintendo 3DS. There are a couple of major differences between the two audios that will be looked at through our experimentation with the signals. Firstly, the two audio files being used for analysis are going to be of slightly different quality, given that one is a recording of a speaker playing the audio and the other is downloaded directly on the computer. The recorded file will also have other sounds added into the audio as the level is being played such as the sounds picking up coins and stomping on goombahs. Our goal was to try isolating the sound effects of the game in the recorded file and separating them from the music of the level, leaving us with a cleaner audio signal---similar to the downloaded version. We can attempt to do this through putting the music audio file and recorded file through the same filters and analyzing the results to see if the sound effects remain the same.

EXPECTED OUTCOMES

Prior to the project, the group was trying to figure out what would be the end goal. That was the hardest thing to determine because we didn't know what MATLAB functions would work with the different types of recording. Eventually, we decided to try to separate the sound effects of the level. Once this was decided no one really knew what to expect. The consensus was that we'd be able to hear both the recorded and downloaded music, but we weren't sure if the filters would allow us to clearly pick out and play each of the different sound effects. Another thought from the group was that we may be able to take the sounds out of the recorded file and it would end up being close in frequency to the downloaded version.

DATA COLLECTION AND ANALYSIS METHOD

Our signal collection started by deciding what audio we wanted to use, choosing the first level of the game seemed ideal as we knew we had access to it without having to go through the entirety of the game. To start, we found the recording of the level music online and made sure that it would work with MATLAB. Next, we recorded the audio from the game multiple times to make sure we got a usable audio that was compatible with MATLAB. The recording that we analyzed was recorded close to a speaker system so that we did not lose too much volume.

Both of our signals come from the same source, an audio file of the game. However, the original file has not been put through any system and it is a clean signal. The recording that we took changed the signal by introducing the speaker, phone, and subsequent file conversion to the original signal. This cascaded system introduced some signal loss and noise that makes our recording different from the original file.

The only code we used was to input the audio files into MATLAB so that we could use the signal analyzer add-on to put the signals through the different filters we have. The process used to signal

analysis was a walkthrough on how to extract multiple voices from a singular file. This will help us as the process is putting the file through low, high, and bandpass filter and how they are rebuilt and original signal.

Using the Signal Analyzer toolbox, we trimmed our audio files so that they had the same length. This is an important step as they need to match, otherwise, the analysis will use samples instead of time. After trimming our signals, we made three sets of copies of the original signal and our recording. From quickly analyzing the original signal and the recording, we saw that the maximum frequency was 24 kHz. Given that we had 3 filters, we split this range into three sections for each filter.

To use the lowpass filter, we designated 8 kHz as the first bandpass. Similarly, for the high pass filter we selected 16 kHz as the bandpass. The bandpass filter required 2 frequencies, we selected 8 kHz as the lower frequency bandpass and 16 kHz as the higher frequency bandpass. In this step, we expected the signals inside the filter to retain the same magnitude as the original signal and any frequency outside would get diminished.

To determine the frequency and time relationship of the signals, we use signal analyzer spectrogram tool to visualize the signal. Both signals look similar, with most of the power spectrum at low and medium frequencies. We can see that at any instance of time, the signal will be dominated by both components.

Looking at our analyzed signals, we can see from the original audio that most of the sound is composed of medium frequencies. Comparing our recording and we can see the same thing, but once we play the audio, we can hear extra sounds that were not included in the original recording. These sounds are sound effects from the game i.e., Mario jumping or hitting a block.

RESULTS AND OBSERVATIONS

Experimental Results

For our experimental results, we were expecting our signal to have some attenuation and to include extra components that were not present in the original signal.

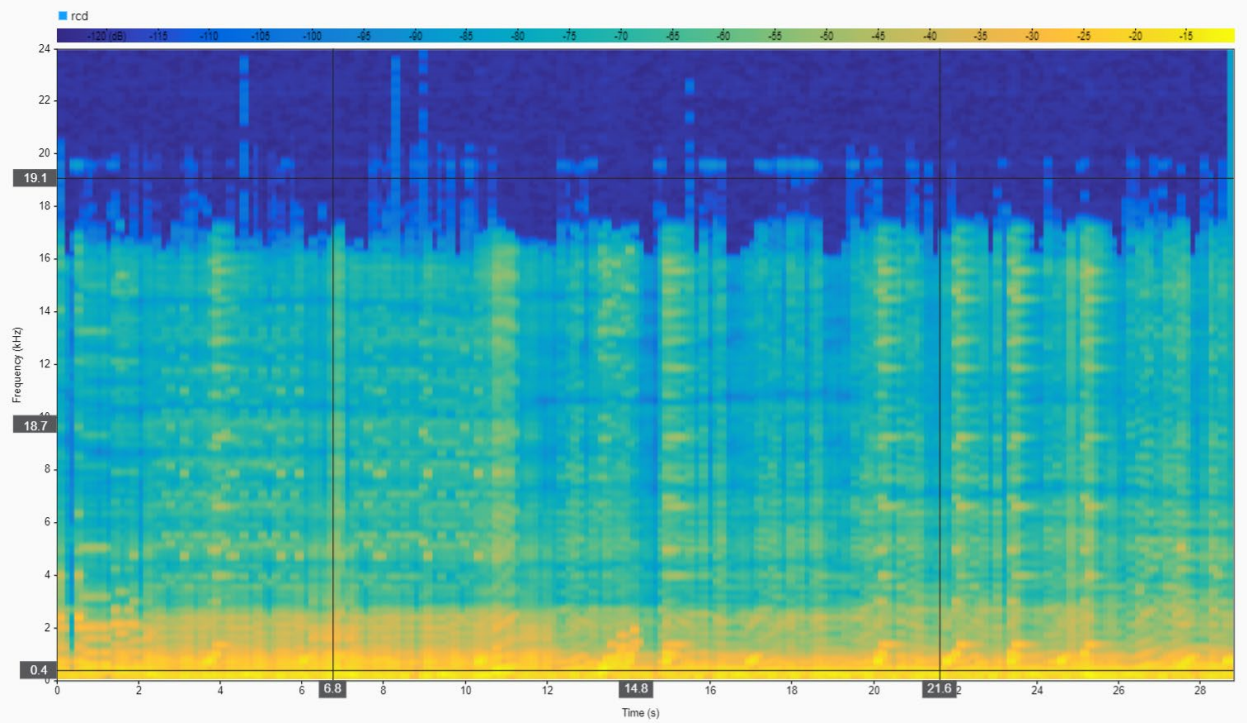

Figure 1: Spectrogram of the recording we took with a phone. Compared to our spectrogram of the original signal, this has lower magnitude, but still dominated by medium frequencies.

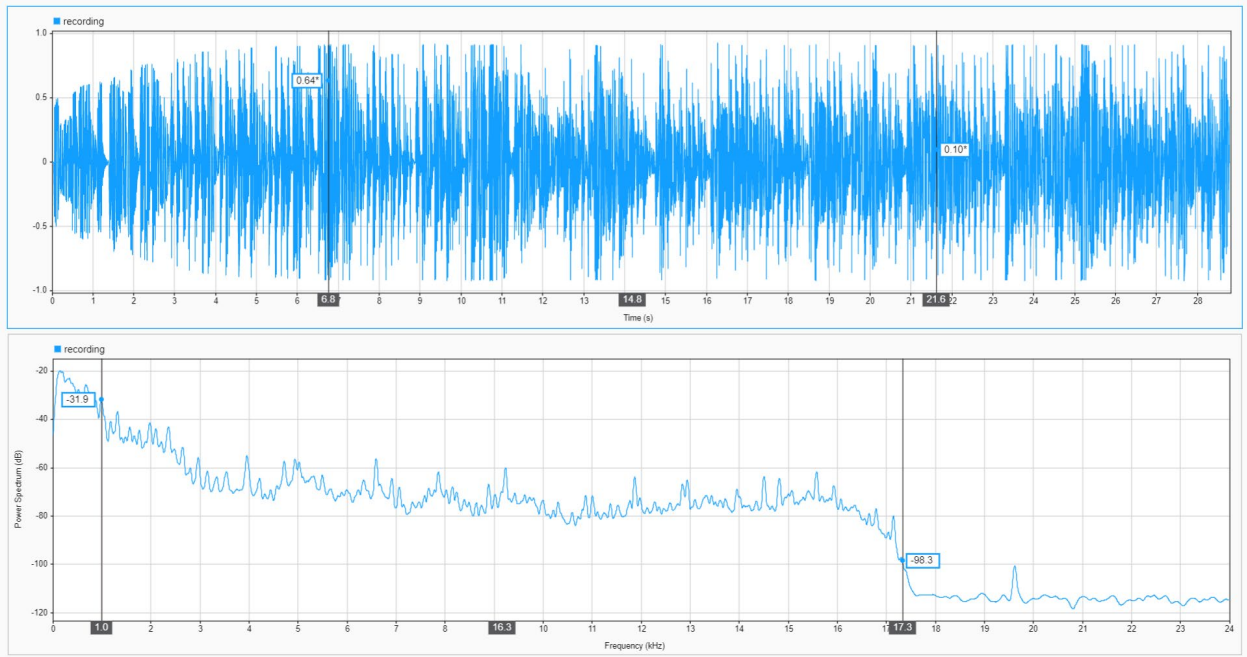

Figure 2: Waveform of the recorded signal and frequency domain plot showing all the components of the signal. This signal is the input to the 3 filters we used. In the following figures we show how the original frequency plot looks compared to a filtered plot.

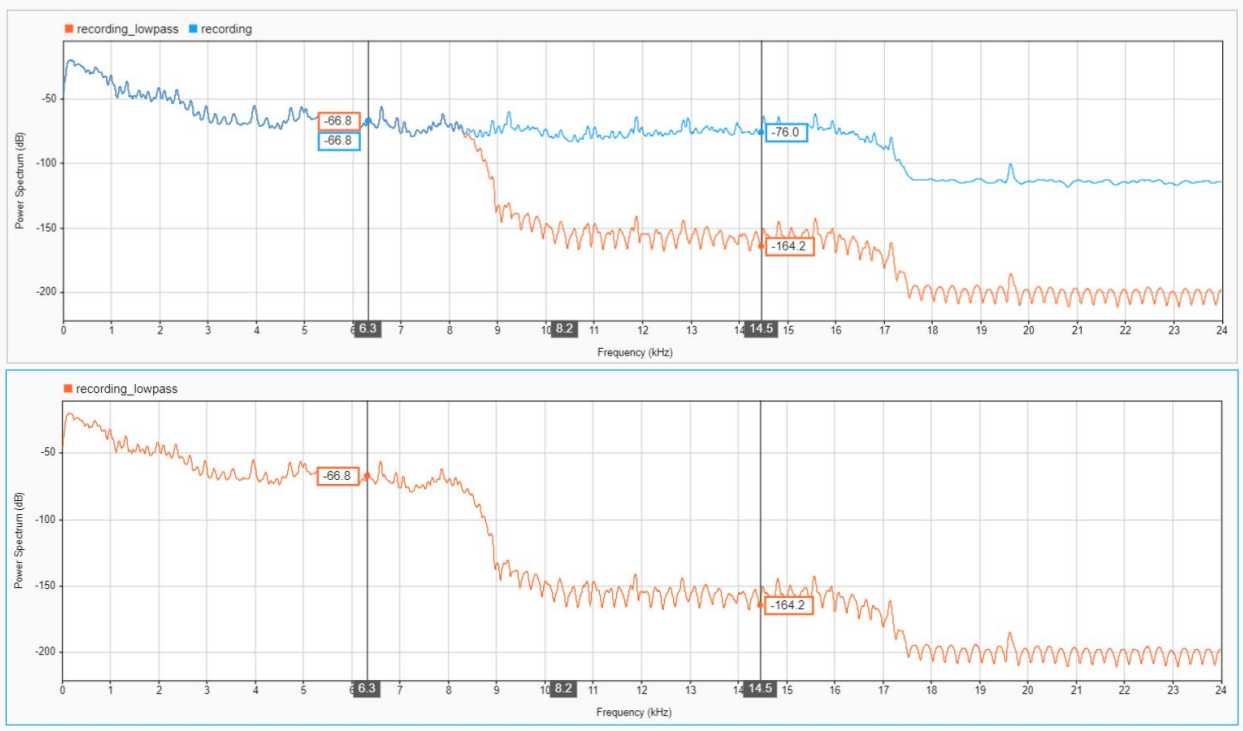

Figure 3: Recorded signal after a low pass filter was used. In the top graph, we can see that the low frequency components match with the unfiltered signal, but after the 8 kHz bandpass the signal gets attenuated.

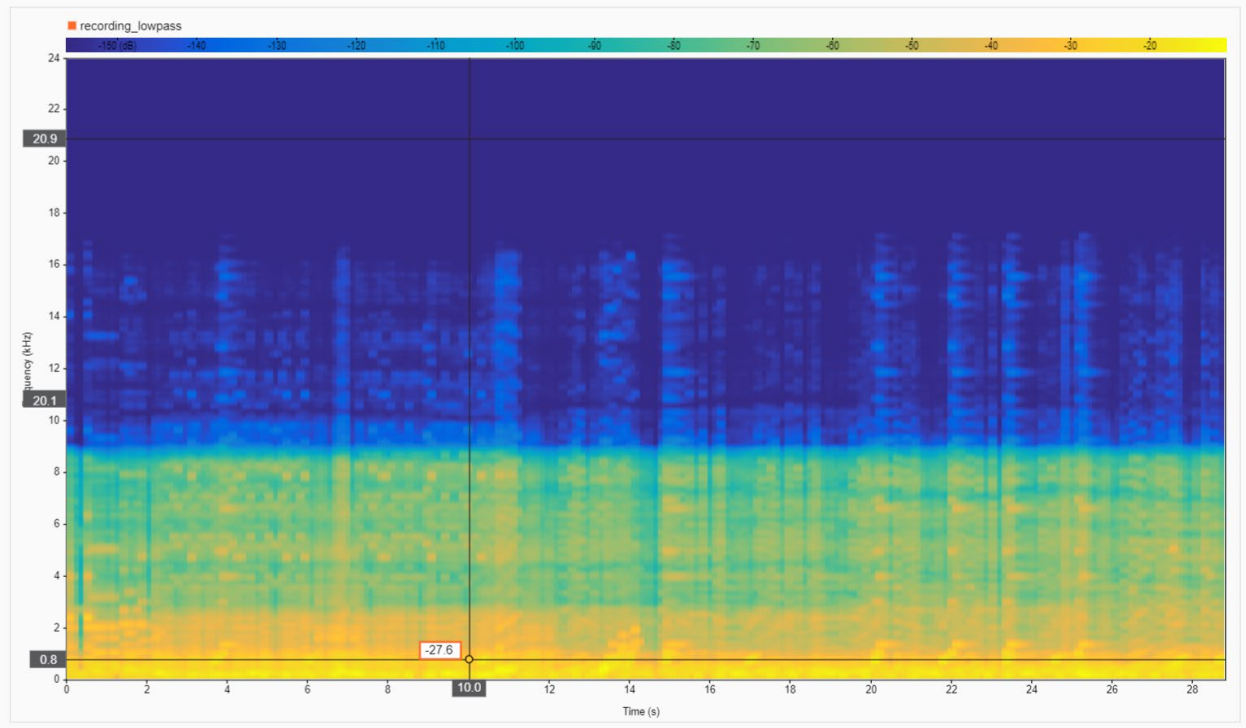

Figure 4: Spectrogram of low pass filtered signal. The lower frequencies are seen to be stronger below 9 kHz as a lowpass filter is placed on the signal, and the strength of the signal becomes reduced as you move up the plot towards higher frequencies.

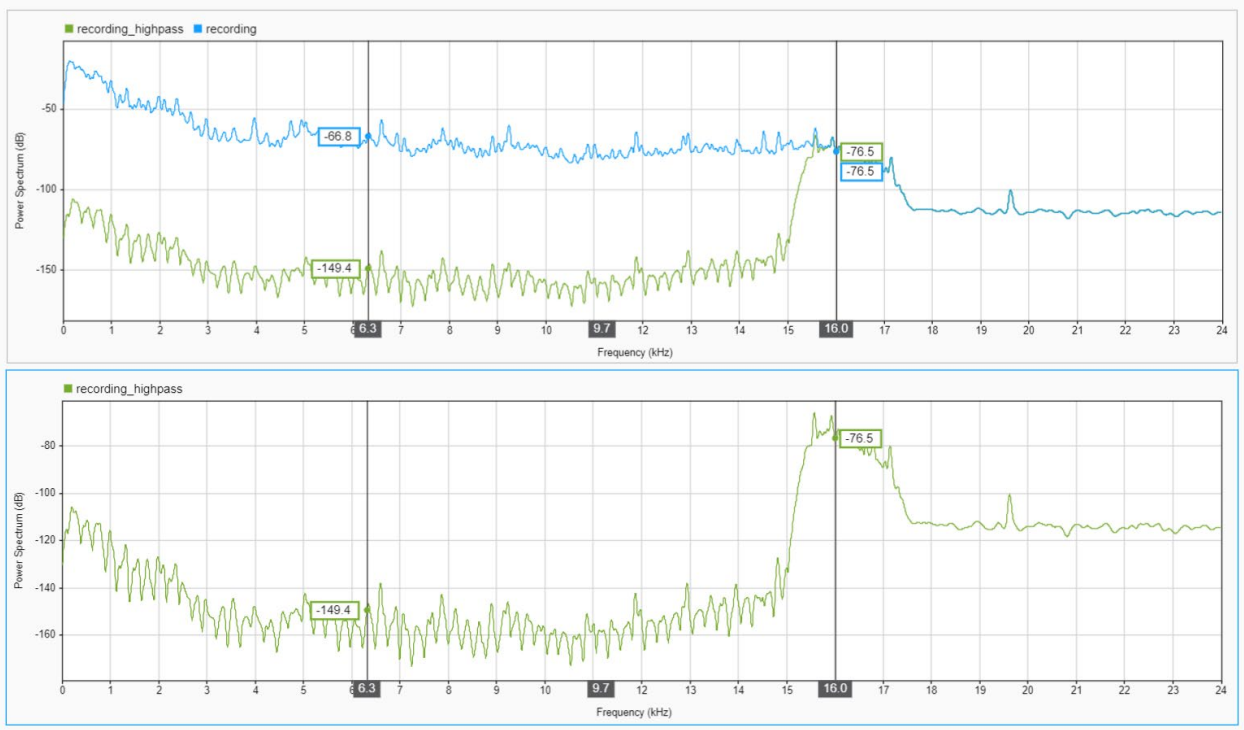

Figure 55: High Pass filter. In the top plot, it can be seen that the two signals differ in decibels at the beginning and middle before matching up after 16 kHz.

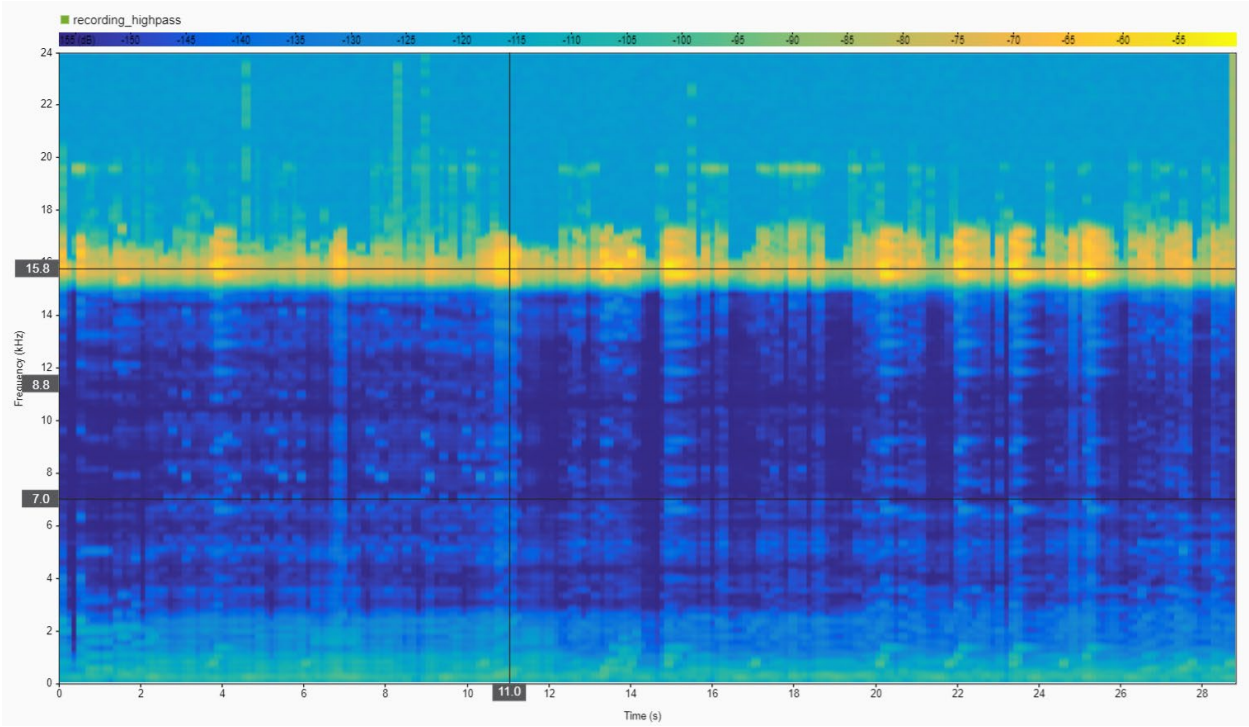

Figure 66: Spectrogram of high pass filtered signal. The signal of the higher frequencies are stronger as the lower frequencies are filtered out.

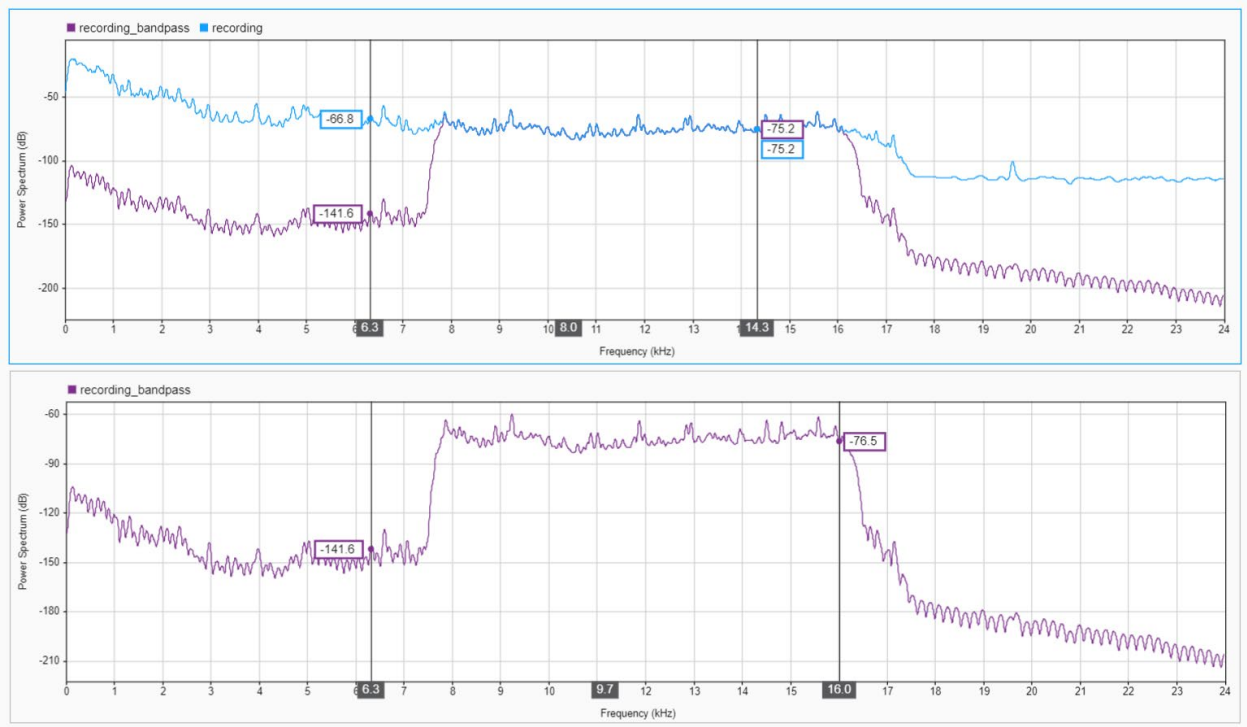

Figure 77: Bandpass. In the top plot, the unfiltered signal matches with the bandpass frequency components between 8 kHz and 16 kHz, while differing in the lower and higher frequencies.

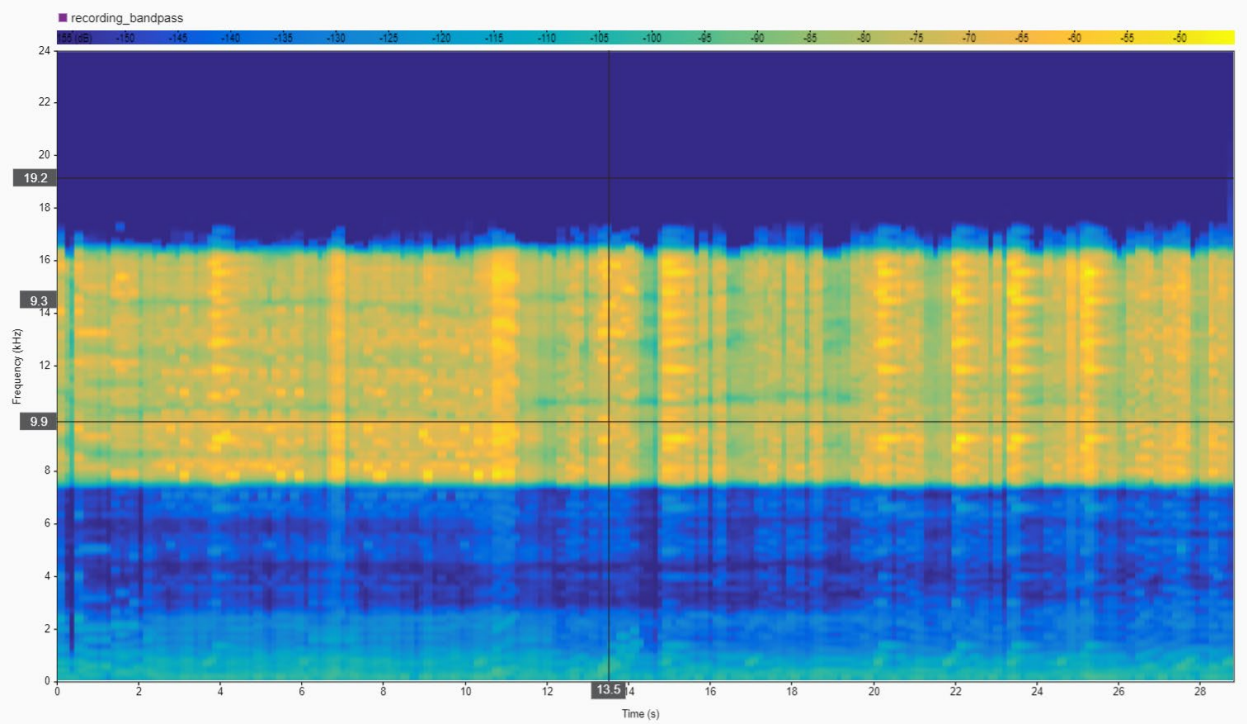

Figure 88: Spectrogram of bandpass filtered signal. It is seen that the frequency is strong where the bandpass occurs in the middle between 8 kHz and 16 kHz and is attenuated outside of the bandpass on the top and bottom.

From class, we know that if we put all of our filtered signals together, we get back the signal before it went through a filter. The following plots show the reconstruction of our filtered signals back into the signal before it was filtered.

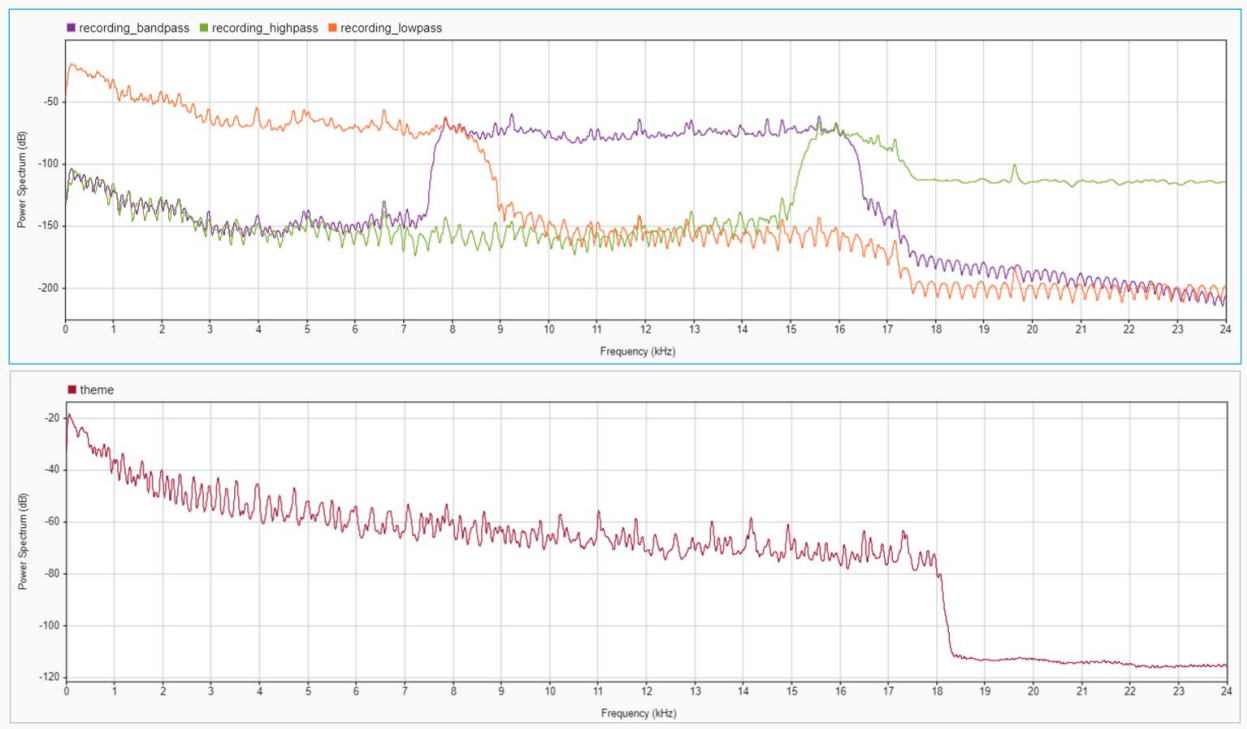

Figure 9: On top, our three filtered signals in the same frequency spectrum plot. On bottom: The recording frequency magnitude plot.

Theoretical Results

We wanted to compare both an original signal with a recorded version. We know that recording an audio signal without any professional equipment will result in loss of signal and quality. Our project wanted to show this with our small recording compared with the highest quality and bitrate file we could find online.

This signal was our theoretical result. From class we understand how an ideal filter looks and how it looks in real life. We included plots that show our expected signal filtering and the spectrogram of the original recording. Using our theoretical signal and comparing it with our recording we can see how they differ from each other.

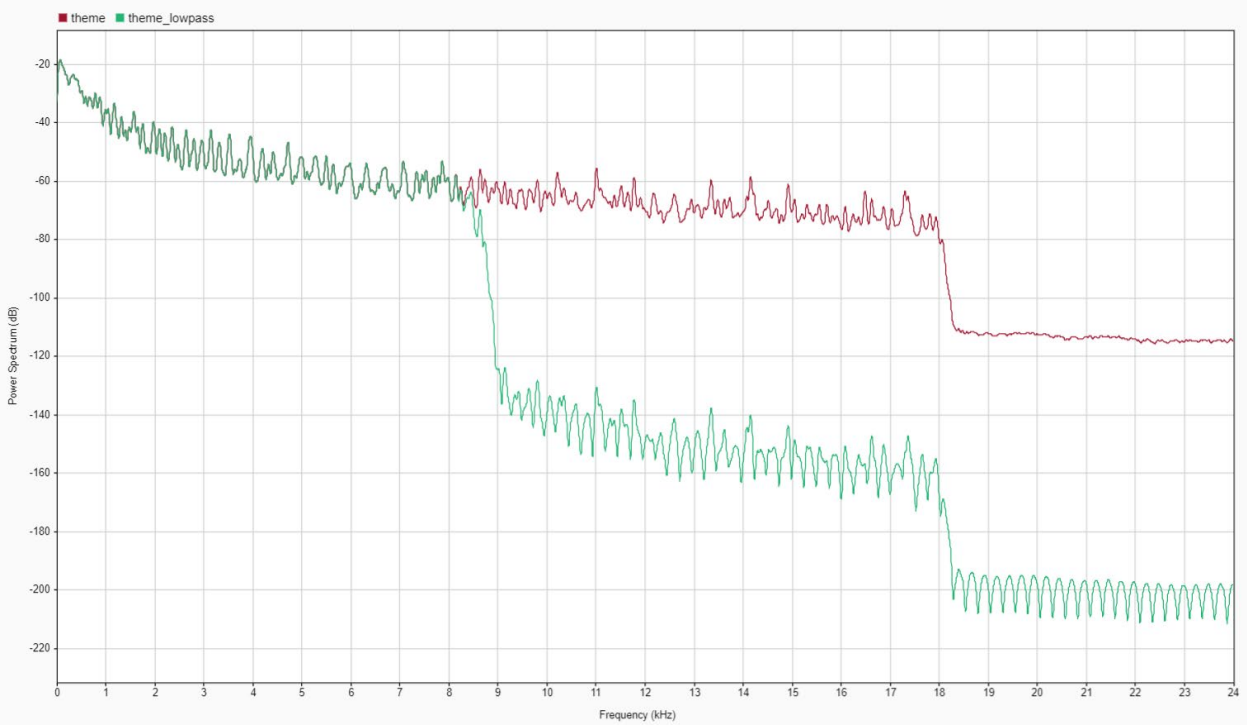

Figure 10: Expected low pass filter representation of our signal with the original signal in the back.

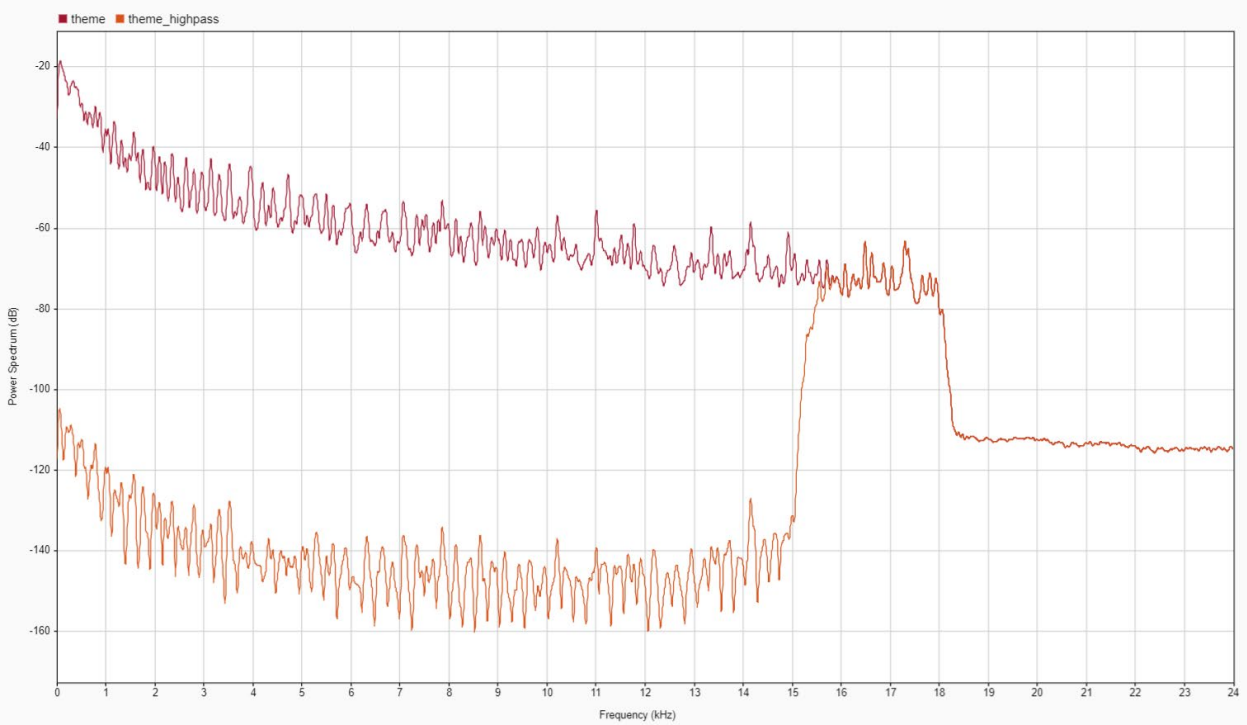

Figure 11: Expected high pass filter representation of our signal.

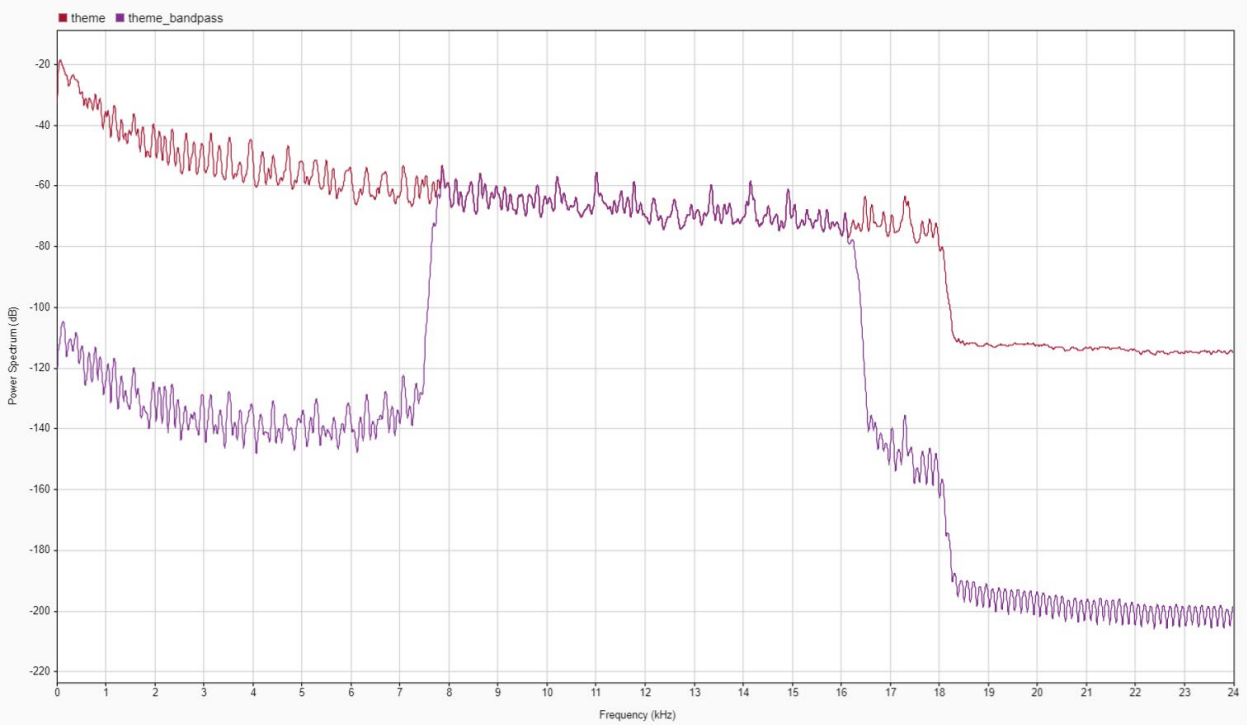

Figure 12:Expected bandpass representation of our signal.

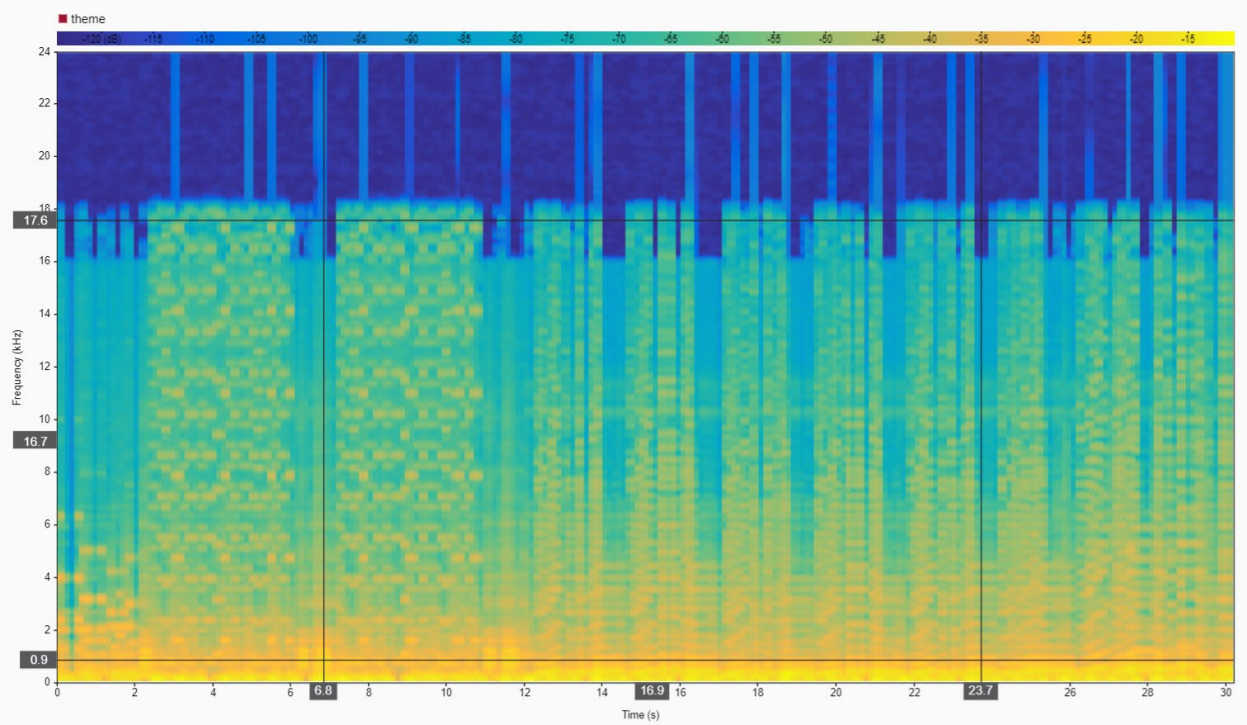

Figure 1313: Spectrogram of original soundtrack recording. We can see that most of the amplitude is at lower to mid frequencies.

Observations

Using our filtered signals and playing them inside MATLAB, we could tell there was some difference between them. The following figures show how our expected filtered signal differed from the recorded filtered signal.

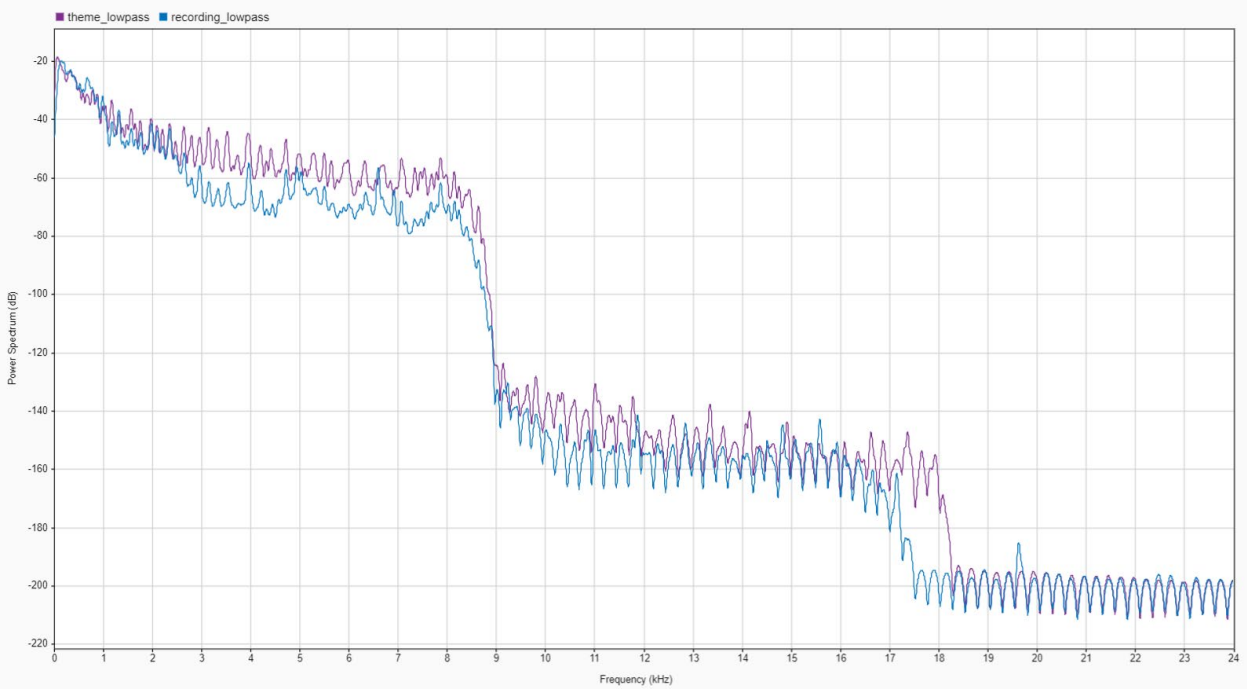

Figure 14: Output signals of original file and recording after passing a low pass filter.

In our filtered signals, we can see that the signal keeps a magnitude level until it reaches the next harmonic and decreases in magnitude again. We understood this would happen as our filter is not ideal, but rather looks akin to a sinc function. Analyzing our plots further we can see how the signal decreases in magnitude after it reaches a certain frequency. This classifies a filter as a low pass, high pass, band pass, or stop gap.

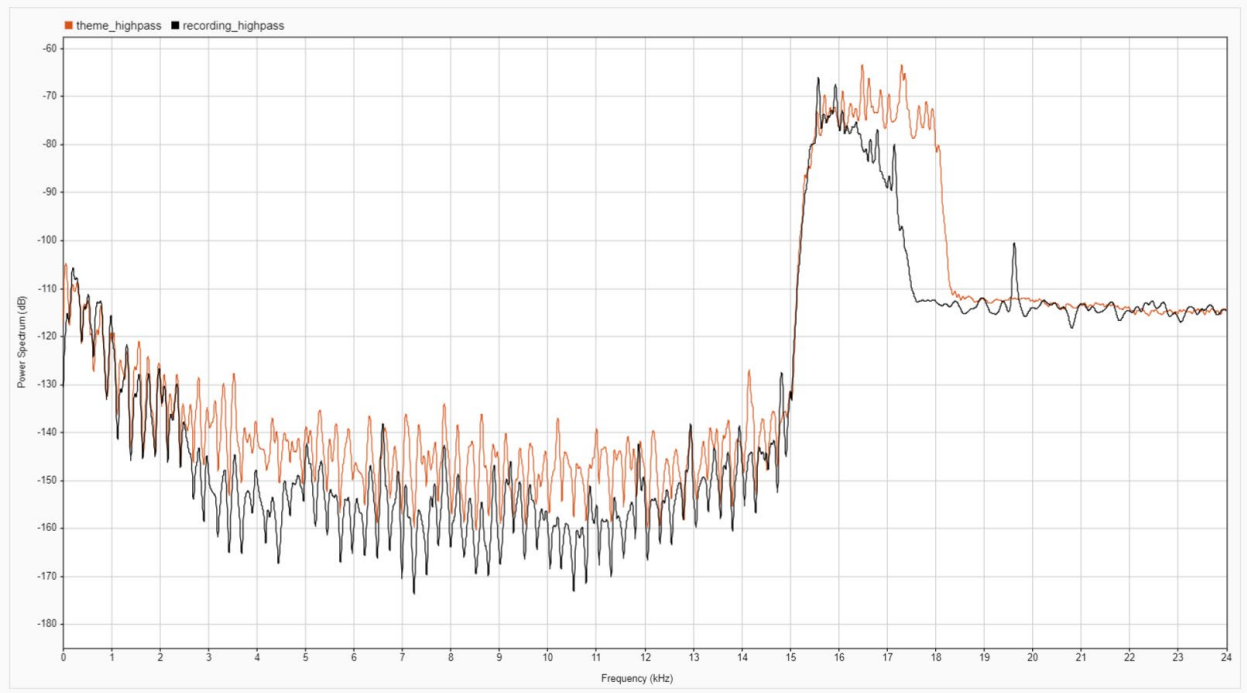

Figure 1615: Expected vs recorded signal after they were filtered with a high pass filter.

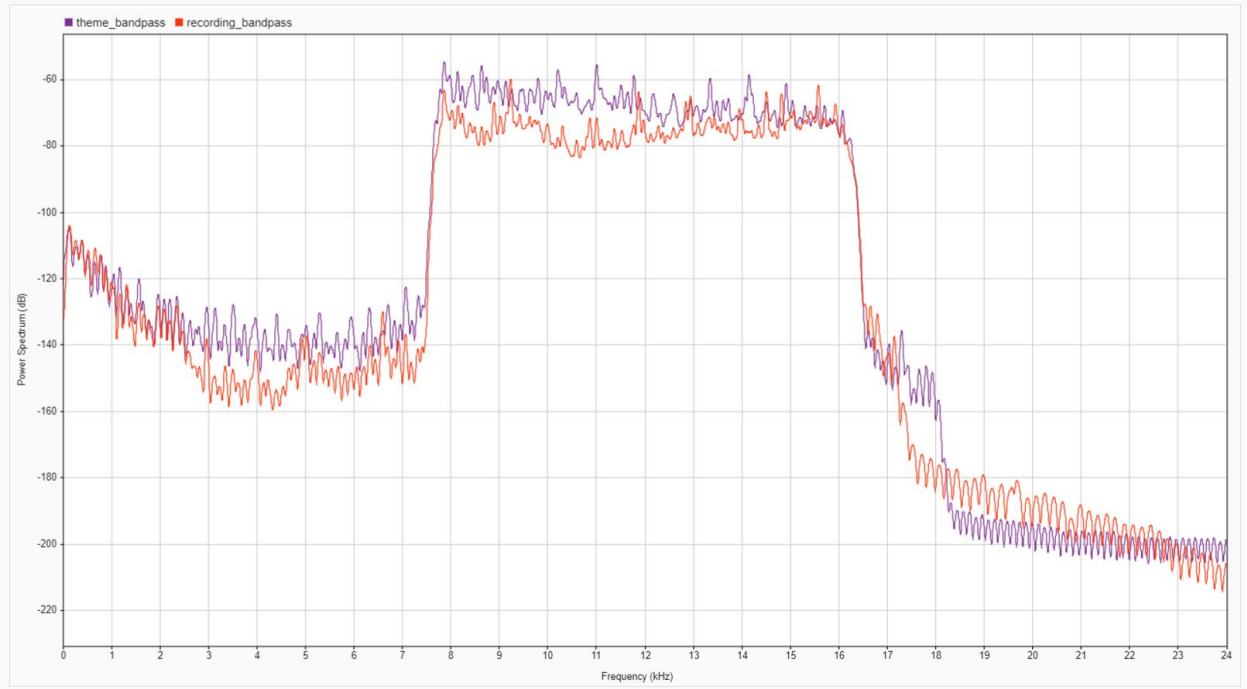

Figure 1716: Bandpass filtered applied to original signal and recorded signal. We can see that they look similar in some frequencies and in other frequencies there is a decrease in magnitude.

We can see that in all the plots, the frequency components are similar but not equal. This could be due to signal loss from the speaker, that microphone could not record those signals due to noise from the

environment, and the artificially added noise from the sound effects of the game. Those components have different magnitudes as we had expected.

SUMMARY

We did the lowpass filter, highpass filter, and bandpass filter to the original signals that we downloaded directly from the computer and the recorded signal from Nintendo 3DS, and we made the comparison to these two signals. Therefore, the concept of highpass filter, lowpass filter, and bandpass filter that were being talked in class are applying to our project. We got similar results between original signals and recorded signals. In the lowpass filter, we have a strong magnitude when we are having the lower frequency, and when the frequency reaches 8kHz, the signal starts to reduce as the frequency goes up. In the highpass filter, we start with low magnitude and when we increase the frequency until 16kHz, the magnitude of the signal starts to increase as the frequency goes up. Lastly, in bandpass filters, the signal starts with a low magnitude, at 8kHz the magnitude increases and stays for another 8kHz, then at 16kHz the magnitude reduces as the frequency goes up. Therefore, our project meets the concept of the filters that we have learned in class.

APPENDICES

```
clc
clear all
close all

%Import of original audio file
[theme, fs] = audioread('D:\UMass\2022\Fall\ENGIN321\Project\theme30-
sec.mp3');

%Import of recorded audio file
[rcd, sr] =
audioread('D:\UMass\2022\Fall\ENGIN321\Project\recording-.mp3');

%Creating the timetable for each of the files
mario = timetable(seconds((0:length(theme)-1)'/fs),theme);
recording = timetable(seconds((0:length(rcd)-1)'/sr),rcd);

%. mat files for original signal and recorded signal
load ('D:\UMass\2022\Fall\ENGIN321\Project\full-audio.mat')
load ('D:\UMass\2022\Fall\ENGIN321\Project\theme_lowpass.mat')
load ('D:\UMass\2022\Fall\ENGIN321\Project\theme_highpass.mat')
load ('D:\UMass\2022\Fall\ENGIN321\Project\theme_bandpass.mat')

%Playing the sound files after we analyzed and filtered them
%sound(mario_theme_highpass.mario_theme_highpass, sr)
%sound(recording_bandpass, sr)
%sound(recording_lowpass,sr)
%sound(recording_highpass,sr)
```

NINTENDO. NEW SUPER MARIO BROS. (NINTENDO DS) ORIGINAL SOUNDTRACK.
https://ia803102.us.archive.org/7/items/Httpwww.originalgamescores.comdownloadable-albumsalbums-n-s21406868/05%20-%20Overworld%20Theme.mp3