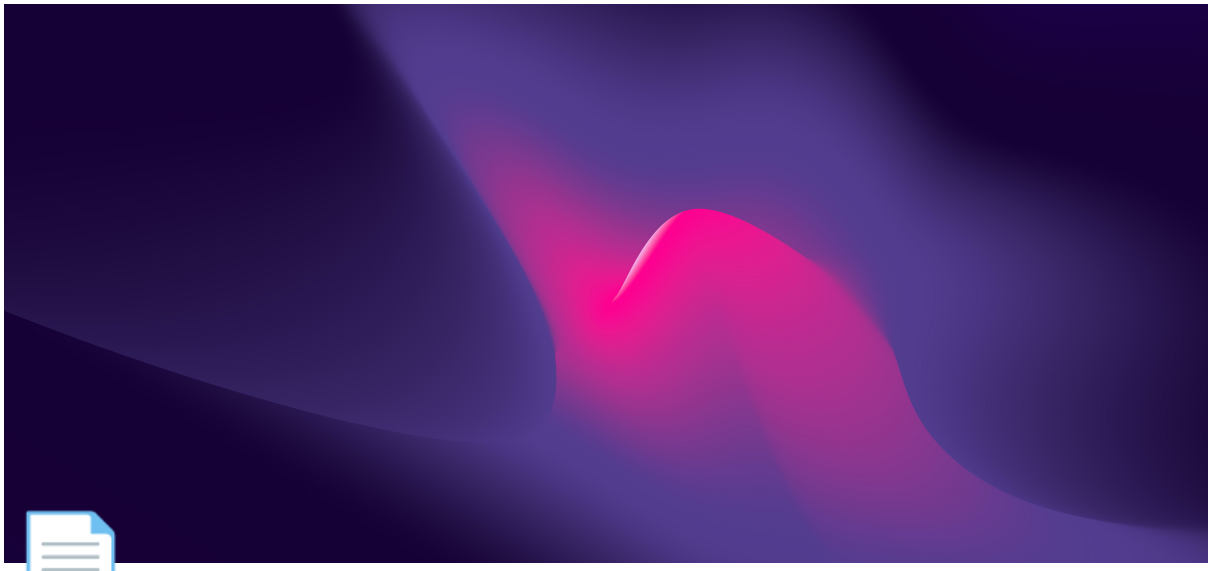


Report (SP - Project)

Team 17:

Sannidhya Gupta (2021112012), Krishna Singh (2021112005), Pratham Kumar Mishra (2021102036)

Audio Equalizer:

In this section, we were required to build an audio equalizer which can increase volume of different frequency bands of a signal, with filtering done in time domain.

1) Band identification

According to the project instructions, our team was given the value of $M = 8$ (number of frequency bands). The next step was to divide the frequency spectrum of the audio signal into 8 parts in order to find the regions of operations for our filters. The frequency spectrum of a digital audio file starts from $f_{min} = 0Hz$, to $f_{max} = \frac{Fs}{2}$, where Fs is the sampling rate of the audio file.

2) Filtering

The frequency response of each filter was considered as the end result of the equalizer is to seperate different frequency bands and adjust their volume. We were

given to implement our own filter from scratch. So, we designed band pass filters by taking difference of two low-pass filters.

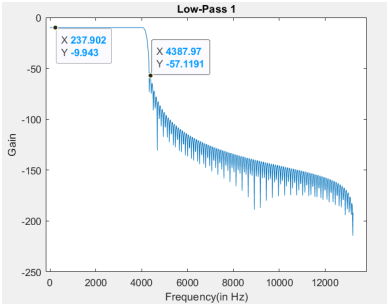
Using the formula of low-pass filter design using sinc-recon function as we derived in Lab 8 :

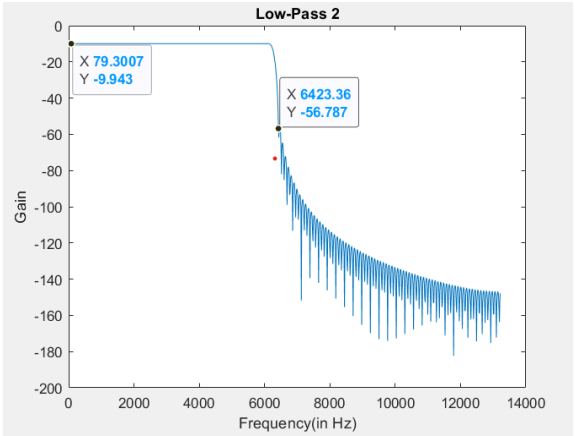
$$x[n] = \frac{k}{\pi} sinc(k(n - n_c)) \qquad k = 2\pi \frac{f_c}{Fs} \qquad n_c = \frac{N-1}{2}$$

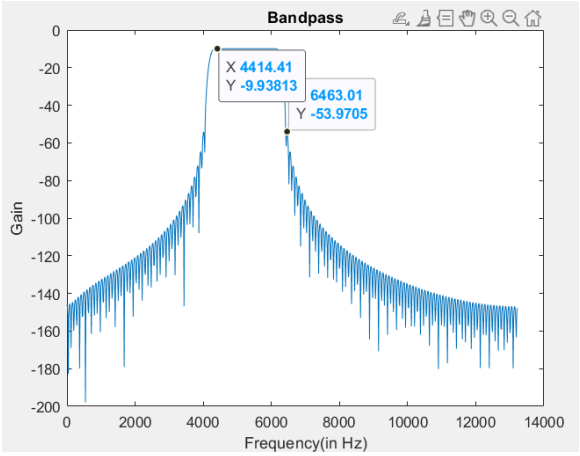
This x[n] was then convoluted by a Hann window to get an attenuation of around 40dB.

Thus, we can design low-pass and band-pass filters from scratch for different values of cut-off with stop band attenuation of around 40dB.

The plots given below show a band-pass filter of passing frequencies 4000 to 6000 Hz, built by subtraction two low-pass filters with cut-off frequencies 6000 Hz and 4000 Hz, respectively.







3) Scaling

The seperated bands from the time domain, can now be visualized in the frequency domain. After the seperated frequency bands are obtained, the amplitude of the same is increased according the desirability and requirements.

---

4) Merging

After the scaled frequency bands are obtained, they are combined into one by adding the individual outputs of all of those. Thus, the scaled version of the frequency bands is obtained, and that too in time domain. Since the output signal is dealt with, and filtered in time domain, there is no need for interconversions between time and frequency domains. and the result (output signal) is thus obtained.

Distinguishing between pad and bat signals

In this section, we were required to distinguish between two types of signals which are captured by the stump mic in the game of cricket. The stump mic captures two types of sounds: ball hitting the bat, and ball hitting the pad. The stump mic that determines where the ball hit plays a crucial role in determining results and judging various scenarios in cricket.

1) Denoising

Denoising method 1

The first task here was denoising, as the signals that were provided were noisy, and for any processing to be done on those signals, firstly the noise had to filtered out. For the denoising, we used a median filter, the difference equation of which, can be given as follows:

$$y[n] = median(y[n], y[n-1], y[n-2])$$

This filter basically takes the median of the value itself, and two previous values in order to remove sudden jumps / disturbances due to the inclusion of noise.

Denoising method 2

There was another constant form of noise in the signal, one obtained at $40Hz$ (which was identified by observing the FFT of the signal obtained after passing

through the median filter). This noise was removed simply by passing the obtained signal through a high pass filter with appropriate cutoff frequency.

---

2) Processing and type identification

Experimentation

Fourier transform of the signal was taken (using FFT) in order to observe the frequency spectra of the given signal. After denoising, we observe that for a bat signal, there is a peak obtained in the FFT at around $1300Hz$. First the denoised signal was played manually in order to be identified as a bat signal, and then the frequency peak was observed to be obtained at $1300Hz$. Using a similar methodology, in the case of a pad signal, there is a peak obtained at $160Hz$ in the FFT.

Implementation

Now for the actual identification, after denoising the signal, the frequency spectra of the signal is observed and then the highest peak of the spectra is observed and the frequency at which it exists is noted. Now, since bat signal's peak was observed to be at a higher frequency than the pad signal, a middleground frequency is chosen, say $\Omega_m = 700Hz$. So, if the obtained peak in the signal is at a frequency higher than Ω_m, then we say that the signal is that of a bat hitting the ball, whereas, if the peak is observed to be located at a frequency lower than Ω_m, then we say that the signal is that of the ball hitting the pad of the batsman.

Vocal and Instrument Seperation

Our team was assigned to design an audio seperator, that seperates tabla sounds from vocals (shrill voice in specific). The system should be able to output two files, one containing the seperated instrumentals (sounds of tabla), and the other one containing the vocals of the song.

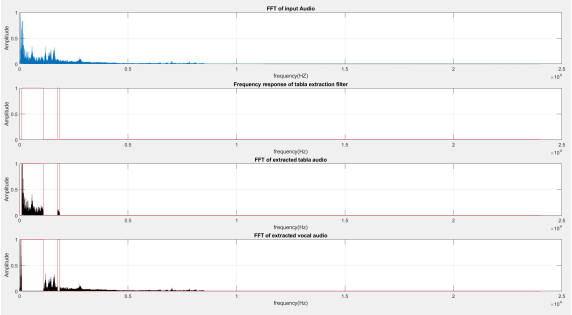
1) Identification

The vocal sounds differ from instrumental sounds, and that too in two ways. Firstly, the sounds of tabla are somewhat high pitched than the vocal range of general human speech, however, since we were required to seperate tabla sounds from a shrill voice, simply a high pass filter would not work. So different samples of tabla sounds were taken from various sources, and their FFT's were observed in order to find a suitable frequency range for tabla sounds. Another way of distinguishing tabla sounds from vocals is that the sounds of the tabla instrument exist in bursts of

amplitude, while the vocals (in time domain) rise and fall slowly in amplitude and do not exist as bursts.

2) Seperation

These frequencies were then extracted out from the original signal by using a band pass filter of the required characteristics in the frequency domain. The frequencies of the vocal part would be all except these, so a suitable band reject filter is used for the vocal part.



3) Synthesis

Next, after obtaining the frequency spectra of the tabla and vocal parts of the songs, inverse FFT (ifft) is taken in order to convert the frequency domain signal back to time domain for it to be further written in an audio file.
If we do not want to separate the signals into two separate files, we can also process the two parts differently and combine it into a single file with different characteristics by merging the two FFT's back after the processing and then taking the inverse FFT to transform it into time domain. This is similar to the first question where we were required to make an equalizer.

(a) Vocal tone enhancement
So in order to enhance the vocal part, the ampltiude of FFT of vocal part can be increased before merging back the two FFT's in order to increase the power of vocal part in the final signal.
(b) Instrumental enhancement
A similar approach can be used as done in part (a), but here instead of interfering with the vocal FFT part, here the amplitudes of the instumental(tabla sounds FFT) can be increased in order for it to be enhanced in the final signal.
(c) Vocal-instrumental balance
This is a property directly related to the balancing of processing parts (a) and (b), so instead of (c) being an explicit process in itself, it can be regulated by altering the processing done in the previous two parts.

Hardware Implementation

We implemented the speaker circuit and microphone circuit given in Lab 9 and intergrated it with the project.

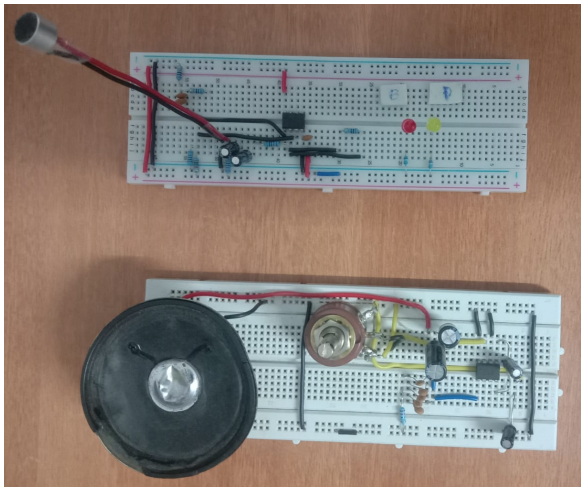
Working of speaker

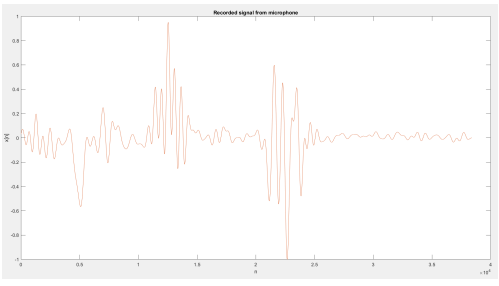
The speaker can be used to play any amplitude array, by splitting the array into very small parts, and finding out the dominant frequency of those parts from FFT and then storing these dominant frequencies in an array. Next, for playing the sound itself, the array of frequencies is passed to the speaker by using the playtone command to play the frequencies iterating through the array using a for loop. However there is a lot of delay due to the way the Arduino support toolbox works in matlab, so the required sampling rate of $8192Hz$ cannot be obtained.

Working of microphone

The microphone is used to record sounds for processing in MATLAB, and this is supposed to be done by using a for loop which reads voltage output from the mic circuit with a delay of $\frac{1}{Fs}$ and stores it in an array in order to create the discrete time domain audio signal of sampling rate $= Fs$. However, there is a very high amount of delay offered by the `readVoltage()` command in MATLAB, which, when used in a for loop even without the delay of $\frac{1}{Fs}$, it offers a delay of $0.04sec$. And after writing a working code for the microphone, we were able to achieve a maximum sampling rate of audio equal to $31Hz$. This sampling rate is too low to detect and record audio of any form. The songs we hear nowadays come at the sampling rate of at least $8192Hz$, which is still considered low quality in modern time. $31Hz$ is too low of a frequency to record any sort of practical musical data which can be used to synthesize an audio file or play a song.

Therefore the mic and speaker part of the hardware was made, and we were able to detect waveforms and peaks in the plot of the recorded signal from the microphone, but the practical real life intergration of these with the given problem statements is not possible with the given set of tools.





Recorded waveform from mic (exists at a sampling rate = 31Hz)