Digital Signal Processing (DSP) Project

Audio Crossover Unit in MATLAB & Simulink

By Arna Maity
(B217012)
Branch - ETC
Batch - 2017-21

Project Outline:

1. Why do we need an Audio Crossover Unit ?
2. Building Blocks of the Project.
3. Digital Filters.
4. Filter Specifications.
5. Algorithm.
6. The Simulink Model.
7. Converting Simulink Model to MATLAB Code.
8. Simulation Results.
9. Limitations and Future Scope.
10. Conclusion.
11. References.

Need of Audio Crossover Unit

The Audio Crossover Unit is often used to improve the sound quality of speakers systems. Because every speaker, has a certain frequency over which it works most efficiently, a single speaker is often not able to cover the entire audio spectrum effectively.

Speakers are often classified into different categories based on their frequency range of effective operation:

Woofer: Freq. Range (20Hz - 2000Hz)
Midrange: Freq. Range (250Hz - 5000Hz)
Tweeter: Freq. Range (2000Hz - 20000Hz)

NOTE: Some portion of the frequency ranges should overlap to ensure that no part of the original audio signal is lost.

But Audio signals do not come pre-separated into separate frequency ranges, so we need a system to separate the incoming Audio Signals into separate signals of fixed frequency ranges. This purpose is fulfilled by an **Audio Crossover Unit**.

Toolboxes Used:

Our MATLAB Script and Simulink model requires the given toolboxes:

- DSP Toolbox
- Audio Systems Toolbox

Common Filter Specifications

```
% Specifications common to all filters.
Fs = audioIn.SampleRate;
filtertype = 'FIR';
Rp = 0.1;
Astop = 80;
```

Fs = Sampling Frequency of the Filters.

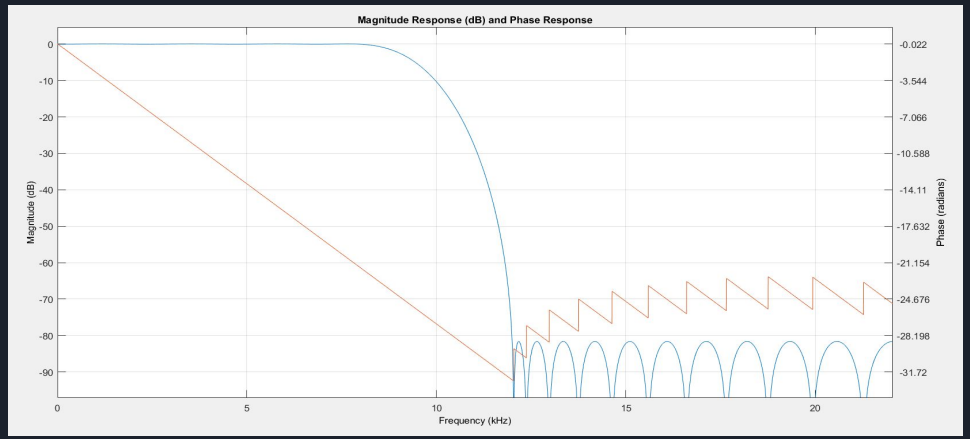
Filtertype = Whether Filter is FIR (Finite Impulse Response) or IIR (Infinite Impulse Response).

Rp = Passband ripple factor.

Astop = Stopband Attenuation.

Low Pass Filter Specifications

Fpass = Passband Frequency.
Fstop = Stopband Frequency.

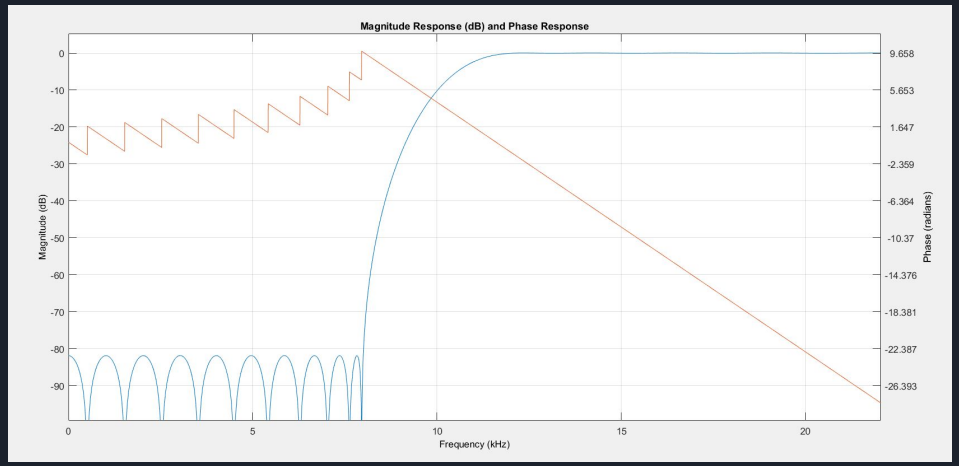



```
% A LPF System Object with the require specs.
Fpass = 200;
Fstop = 2e3;

FIRLPF = dsp.LowpassFilter('SampleRate',Fs, ...
    'FilterType',filtertype, ...
    'PassbandFrequency',Fpass, ...
    'StopbandFrequency',Fstop, ...
    'PassbandRipple',Rp, ...
    'StopbandAttenuation',Astop);
```

High Pass Filter Specifications

```
% A HPF System Object with the require specs.
Fpass = 8e3;
Fstop = 5e3;

FIRHPF = dsp.HighpassFilter('SampleRate',Fs,...
    'FilterType',filtertype,...
    'PassbandFrequency',Fpass,...
    'StopbandFrequency',Fstop,...
    'PassbandRipple',Rp,...
    'StopbandAttenuation',Astop);
```

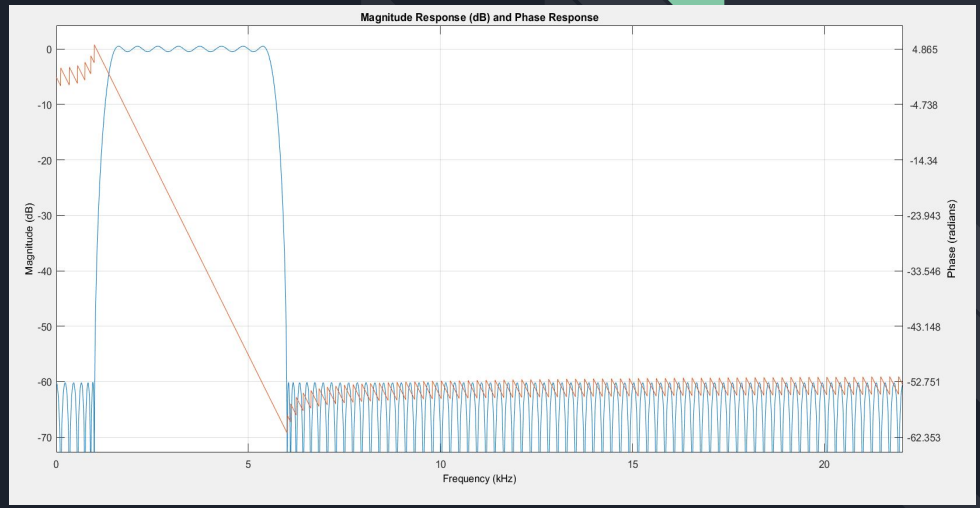
Fpass = Passband Frequency.
Fstop = Stopband Frequency.



Band Pass Filter Specifications

```
% A BPF System Object with required specs.
Fstop1 = 1000;   % First Stopband Frequency
Fpass1 = 1500;   % First Passband Frequency
Fpass2 = 5500;   % Second Passband Frequency
Fstop2 = 7000;   % Second Stopband Frequency

h = fdesign.bandpass('fst1,fp1,fp2,fst2,ast1,ap,ast2', ...
                     Fstop1, Fpass1, ...
                     Fpass2, Fstop2, Astop, Rp, Astop, Fs);

FIRBPF = design(h, 'equiripple', ...
    'MinOrder', 'any', ...
    'SystemObject', true);
```

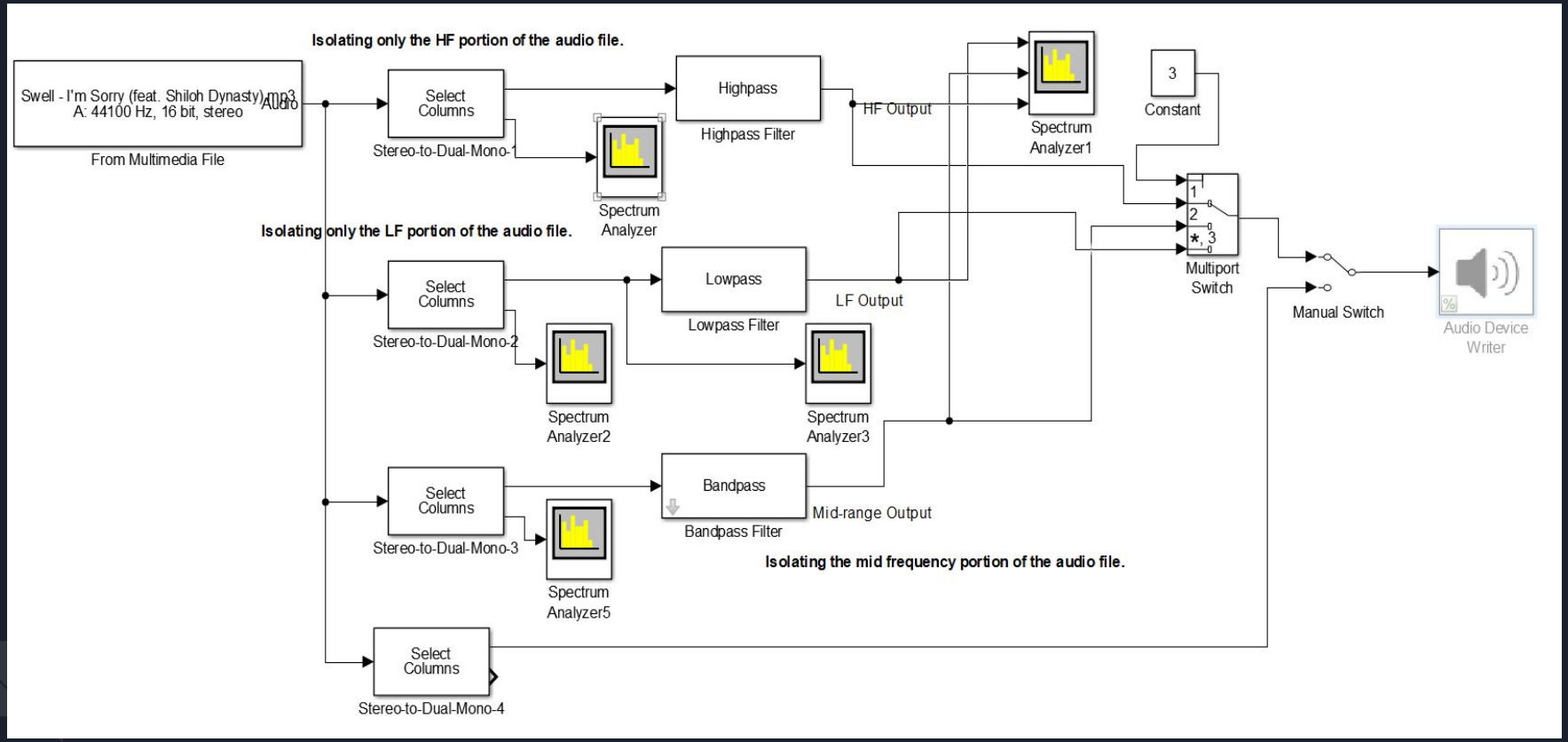
Fstop1= Stopband (Stop -> Passband Transition)
Fpass1= Passband (Stop -> Passband Transition)
Fstop2= Stopband (Pass -> Stopband Transition)
Fpass2= Passband (Pass -> Stopband Transition)



Algorithm

1. Separate the stereo audio input into 2 single channel outputs (2 mono outputs).
2. Take 3 copies of one of the 2 channels and pass them through a Low Pass Filter, Band Pass Filter & High Pass Filter respectively.
3. Pass the output of the 3 filters to a spectrum analyzer to have a look a the output of the 3 filters.
4. Depending on the output of the 3 filters choose the speaker units best suited for the 3 different frequency ranges and output the 3 different components through separate speakers.
5. Do this for both the channels.
6. Finally, we have better audio quality.

The Simulink Model



Converting Simulink Model to MATLAB Code

1.
```
[channel, Fs] = audioread('Swell.mp3');
audiowrite('Swell_mono_L.wav',channel(:,1),Fs);
audiowrite('Swell_mono_R.wav',channel(:,2),Fs);
```

2.
```
audioIn = dsp.AudioFileReader;
audioIn.Filename = 'Swell_mono.wav';
audioIn.OutputDataType = 'single';
```

3.
```
audioOut = audioDeviceWriter;
audioOut.SampleRate = audioIn.SampleRate;
```

4.
```
Fs = audioIn.SampleRate;
filtertype = 'FIR';
Rp = 0.1;
Astop = 80;
```

MATLAB Code

5.

```
% A HPF System Object with the require specs.
Fpass = 8e3;
Fstop = 5e3;

FIRHPF = dsp.HighpassFilter('SampleRate',Fs,...
    'FilterType',filtertype,...
    'PassbandFrequency',Fpass,...
    'StopbandFrequency',Fstop,...
    'PassbandRipple',Rp,...
    'StopbandAttenuation',Astop);

% Plot the Filter Impulse Response and Magnitude and Phase Response.
fvtool(FIRHPF,'Analysis','impulse');
fvtool(FIRHPF,'Analysis','freq');
```

6.

```
% A LPF System Object with the require specs.
Fpass = 200;
Fstop = 2e3;

FIRLPF = dsp.LowpassFilter('SampleRate',Fs, ...
    'FilterType',filtertype, ...
    'PassbandFrequency',Fpass, ...
    'StopbandFrequency',Fstop, ...
    'PassbandRipple',Rp, ...
    'StopbandAttenuation',Astop);

% Plot the Filter Impulse Response and Magnitude and Phase Response.
fvtool(FIRLPF,'Analysis','impulse');
fvtool(FIRLPF,'Analysis','freq');
```

7.

```
% A BPF System Object with required specs.
Fstop1 = 1000;   % First Stopband Frequency
Fpass1 = 1500;   % First Passband Frequency
Fpass2 = 5500;   % Second Passband Frequency
Fstop2 = 7000;   % Second Stopband Frequency

h = fdesign.bandpass('fst1,fp1,fp2,fst2,ast1,ap,ast2', ...
                     Fstop1, Fpass1, ...
                     Fpass2, Fstop2, Astop, Rp, Astop, Fs);

FIRBPF = design(h, 'equiripple', ...
    'MinOrder', 'any', ...
    'SystemObject', true);

% Plot the Filter Impulse Response and Magnitude and Phase Response.
fvtool(FIRBPF,'Analysis','impulse');
fvtool(FIRBPF,'Analysis','freq');
```

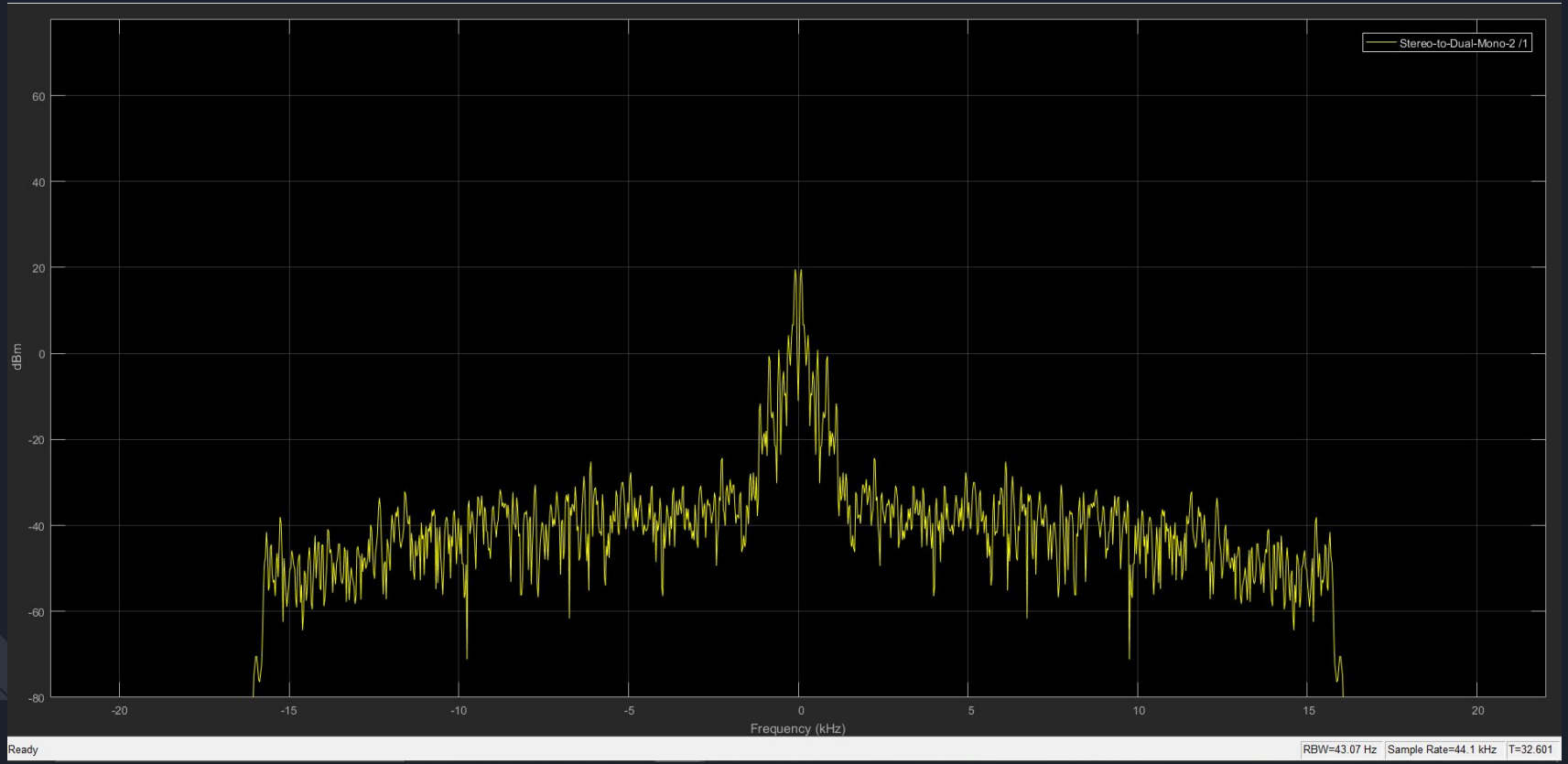
MATLAB Code

8.
```
% Create an object for a Spectrum Analyzer.
SA = dsp.SpectrumAnalyzer('SampleRate',Fs,'NumInputPorts',3,...
    'PlotAsTwoSidedSpectrum',true,...
    'ShowLegend',true,'YLimits',[-80,60]);

SA.ChannelNames = {'LPF Output','BPF Output','HPF Output'};
```

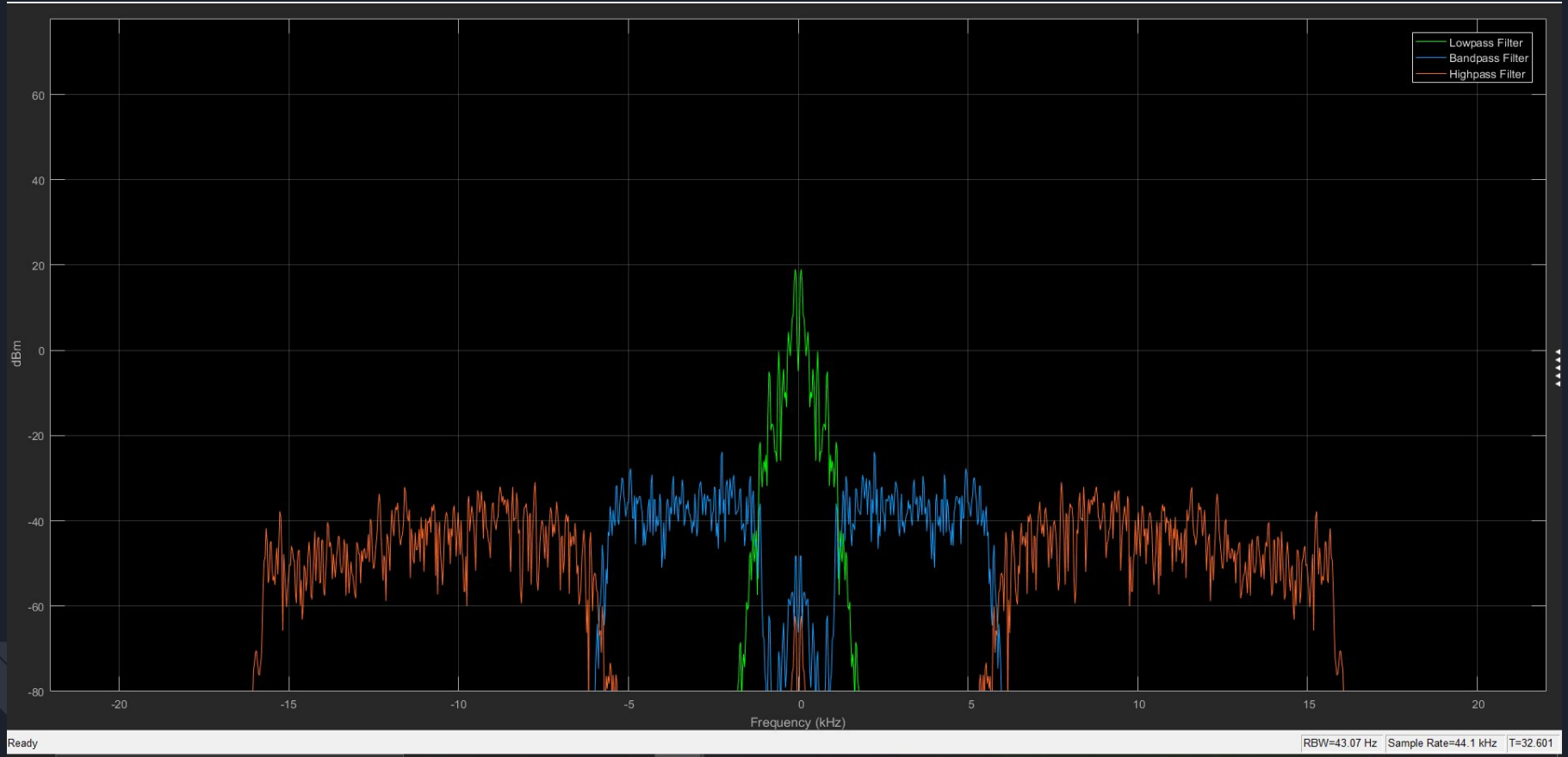
9.
```
% Interconnection of the components and playing the audio file.
while ~isDone(audioIn)
    audio = step(audioIn);    % Read audio source file
    HPFOut = step(FIRHPF,audio); % Filter the data
    LPFOut = step(FIRLPF,audio);
    BPFOut = step(FIRBPF,audio);
    step(SA,LPFOut,BPFOut,HPFOut);
    %step(audioOut,BPFOut);    % Play the filtered data
end
```

Simulation Results

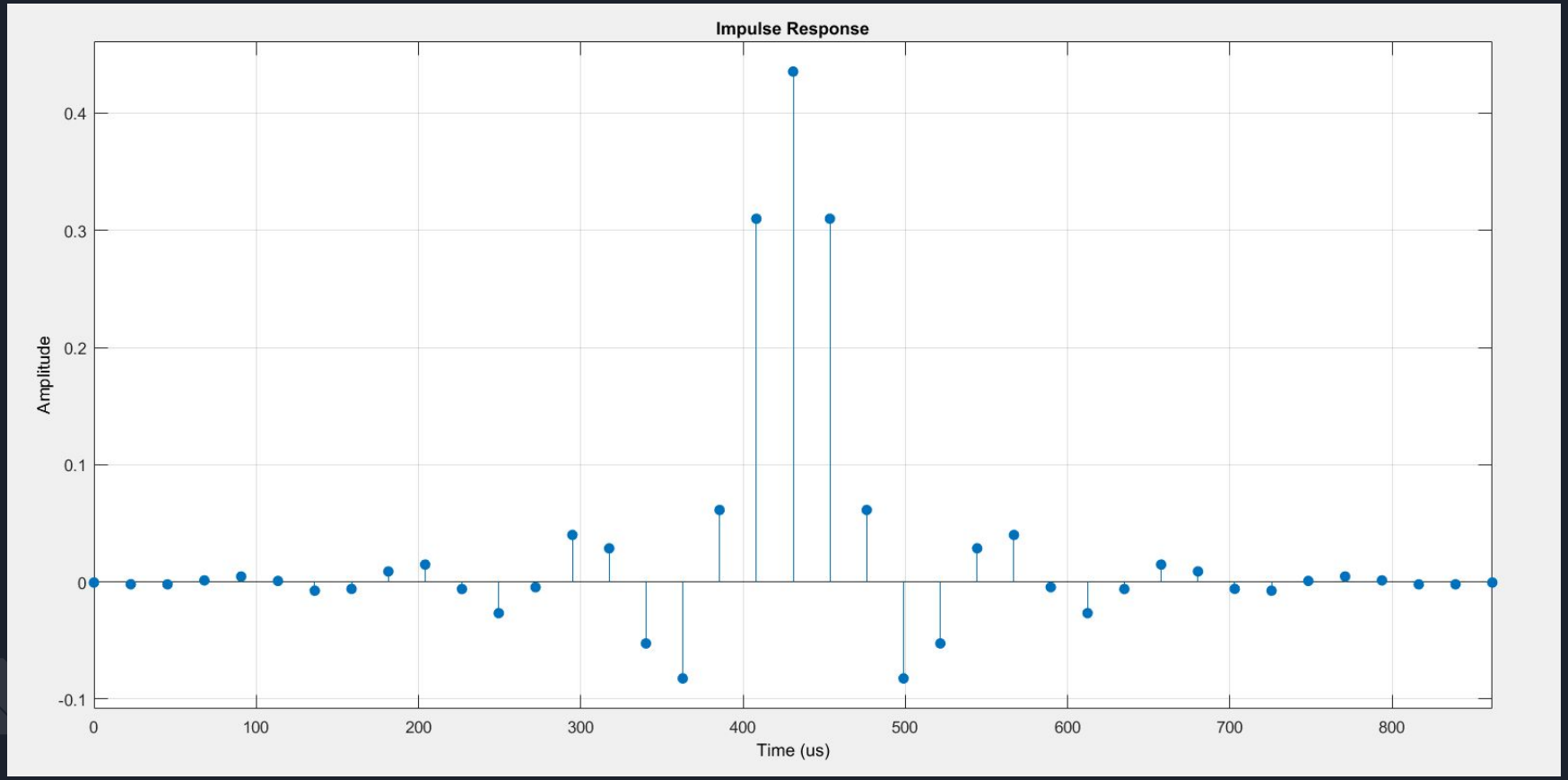
Original Spectrum of the Single Channel Output.



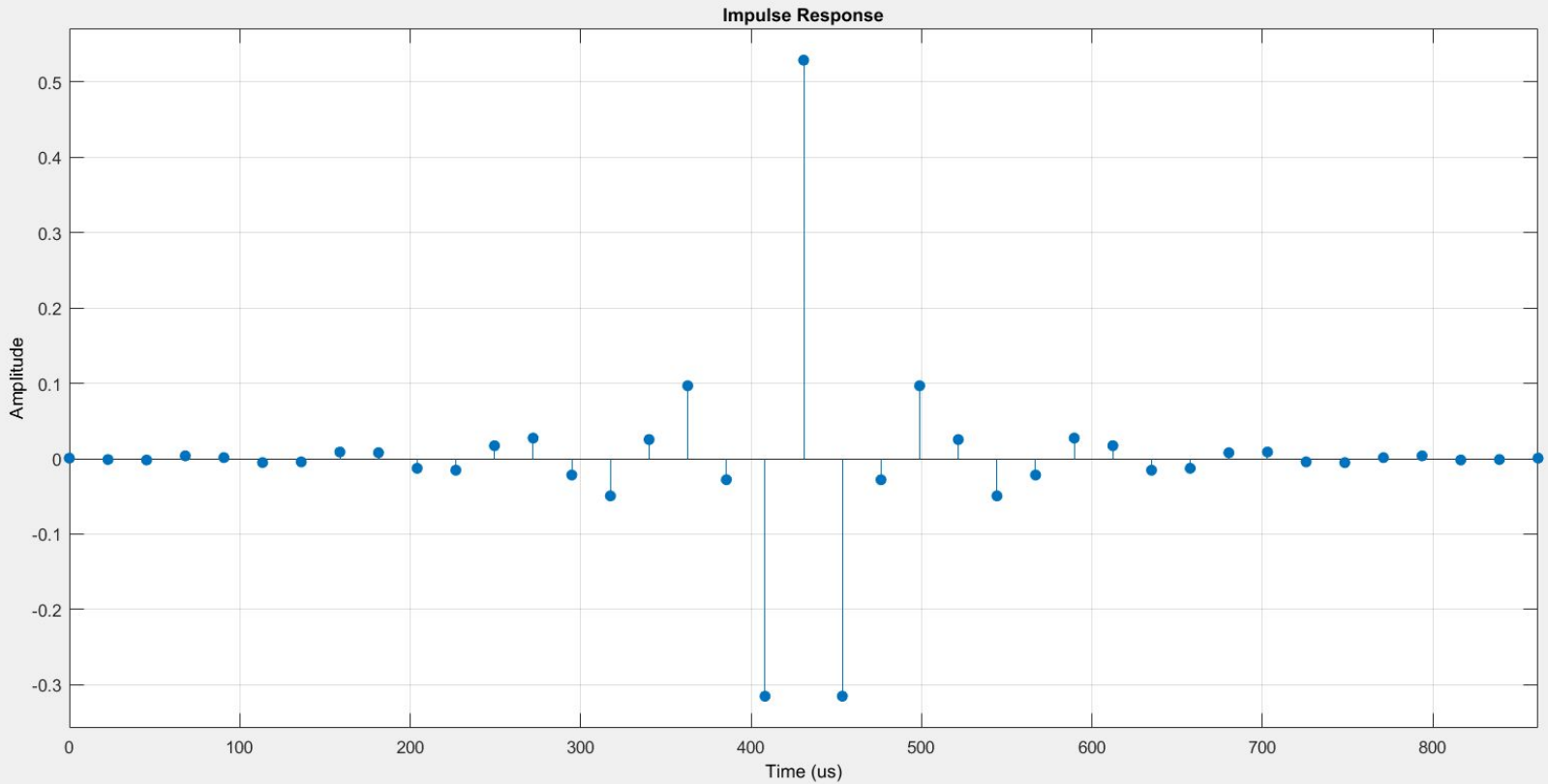
Simulation Result (Contd.)



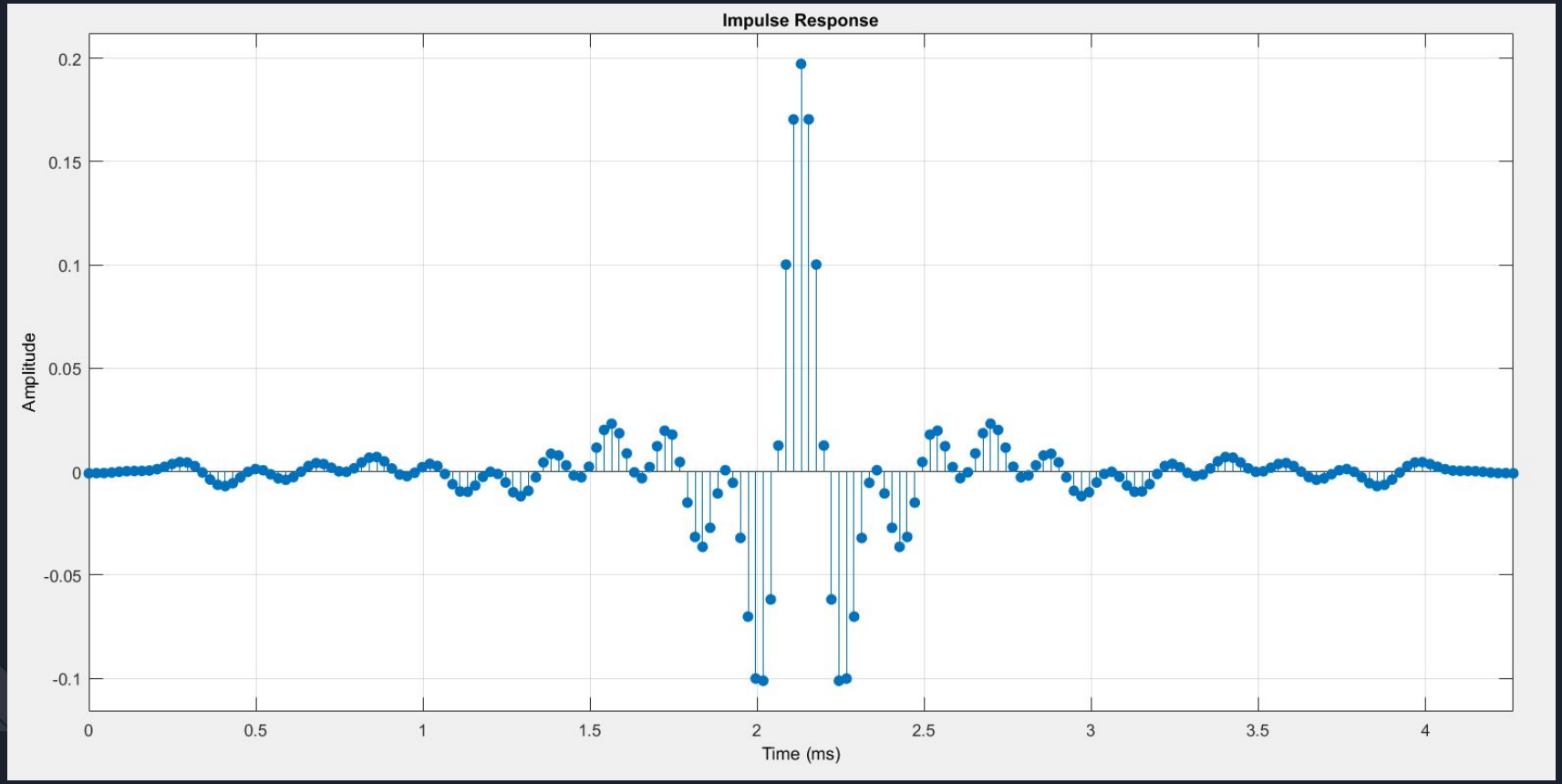
Low Pass Filter Impulse Response



High Pass Filter Impulse Response



Band Pass Filter Impulse Response



Limitations & Future Scope

1. Presently, the Audio Crossover Unit just filters the different frequency components and sends them to different speakers, but we can also employ a gain control system before sending the filtered output to the speakers. This will help in performing audio equalization and produce a more balanced sound (where each frequencies have equal gains).
2. **Application:**
   One of the most important applications of Audio Crossovers is in the Design of **Hi-Fi Systems**.

   So, what is the difference between a **Hi-Fi System** and a **Simple Stereo Speaker Setup**?

   Ans: Hi-Fi Stands for High-Fidelity which simply means that their aim is to produce near perfect audio(Provide equal gain across all the frequencies of the audio spectrum) . This is the reason you might have noticed multiple speakers on a single unit of the Hi-Fi setup.

   Have a look at the following setup...

Have a look at the above Hi-Fi Setup. Each unit for each of the two channels of the stereo signal has 3 separate speakers. The **smallest one** is the **Tweeter**, the **Medium one** is the **Midrange Speaker** & the **largest one** is the **Woofer.**

Conclusion

So, this project explains how we can drastically improve the sound quality of any sound system by using specific speakers for specific frequency ranges.

Link to a video demonstration of the project: Audio Crossover Unit.

Link to the Source Code of the project: Audio Crossover Unit.

Link to the Project Abstract: Audio Crossover Unit.

References:

1. System Objects.
2. Stream Processing in MATLAB.
3. Low Pass Filter.
4. High Pass Filter.
5. Band Pass Filter.
6. Spectrum Analyzer.