Report

Final Project

Audio Equalizer

EENG 1920: Project II – Introduction to Electrical Engineering

Team Members:
Shabuktagin Photon Khan
Ogbonna Christian
Juan Pineda Aguirre

University of North Texas
Department of Electrical Engineering

Contents

ABSTRACT

The Final Project gives us the opportunity to work on Mat Lab Graphical User Interface and Lab View to work on a brief report for it. The objective of the project was to make an audio equalizer and work on some modifications like adding LEDS and delay effect with it. The audio equalizer was a 3 band channel equalizer which has bass, mid and treble section only.

INTRODUCTION

Mat Lab by MathWorks is a high level computation and simulation language for technical computing. It integrates computation, visualization, and programming in an easy-to use environment. Mat Lab is used for math and computation, algorithm development, data acquisition, modeling, simulation, prototyping, data analysis, exploration and visualization, scientific and engineering graphics and finally for application. Lab VIEW is short for Laboratory Virtual Instrument Engineering Workbench which is a system-design platform and development environment for a visual programming language from National Instruments.

Final project allowed us to work as a team working together for a single purpose. Applying all the information learned throughout the course either by programming on Mat Lab or working on the Lab View to make an audio equalizer. The problems faced during the making of audio equalizer was solved by divide and conquer method, which is based on trying different variables to solve a single problem. The report also discusses the barriers we faced, how we thought on the problem, the different ways we solved the problem and finally what we learned from every aspect. This project will help us implement it to real life scenarios where we need to think on time management, teamwork, different methods and results in order to achieve a single objective.

Report

The report is divided into 5 different sections; each section represents the work done through all the different functions, the different ways we approach to the problem and pros and cons on each choice we were making as we were working along the project. The first four sections are dealt with Lab view and last portion deals with Mat Lab.

First Section: Reading the File

In this section, we will focus on how to make the audio files work on Lab View. At first, we follow few steps which is shown below to get the required functions and controls for Lab View to read the file. The Block diagram is shown in Figure 1 and Front panel is shown in Figure 2.

Functions
Programming>>Graphics and sounds>>Sounds>>Output>>Play file
Programming>>Graphics and sounds>>Sounds>>Output>>Info
Programming>>Graphics and sounds>>Sounds>>Output>>Clear
Programming>>Structures>>While Loop

Controls
Express>>LEDS>>Round LEDS

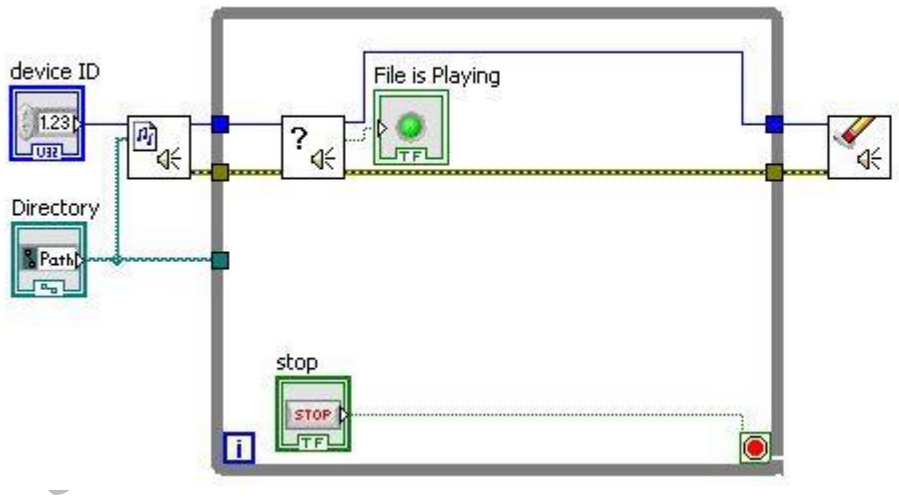


Figure 1: Block Diagram (reading the file)

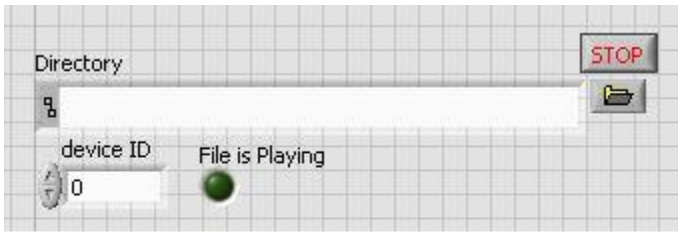


Figure 2: Front Panel (reading the file)

Function: This work helps us to read only .wav files. In this block diagram the function "Play Sound File" will allow us to create path (directory) and device id. The play files task ID and error out will go through the while Loop and to "Sound Output Info." The sound output info will detect whether the file is playing or not. If the file plays properly then the Boolean function LED in controls will glow as long as the audio keeps on playing. When the audio stops, the light of the LEDs turns off. At the right hand side of the block diagram, there is "Sound Output Clear." It helps to clears the data.

Lessons Learned: We learned how to play the .wav files and synchronize it with Boolean Function.

Second Section: Audio Equalizer Using .wav File

In this section, we will focus on how to make the audio files with equalizers to work on Lab View. At first, we follow few steps which is shown below to get the required functions and controls for Lab View. The Block diagram is shown in Figure 3 and Front panel is shown in Figure 4.

Functions
Programming>>Graphics and Sounds>>Sound>>Files>>Simple Read (Added Path and info)
Programming>>Graphics and Sounds>>Sound>>Output>>Play Waveform
Mathematics>>Numeric
Express>>Signal Analysis>>Filter

Controls
Numeric Controls>>Knob

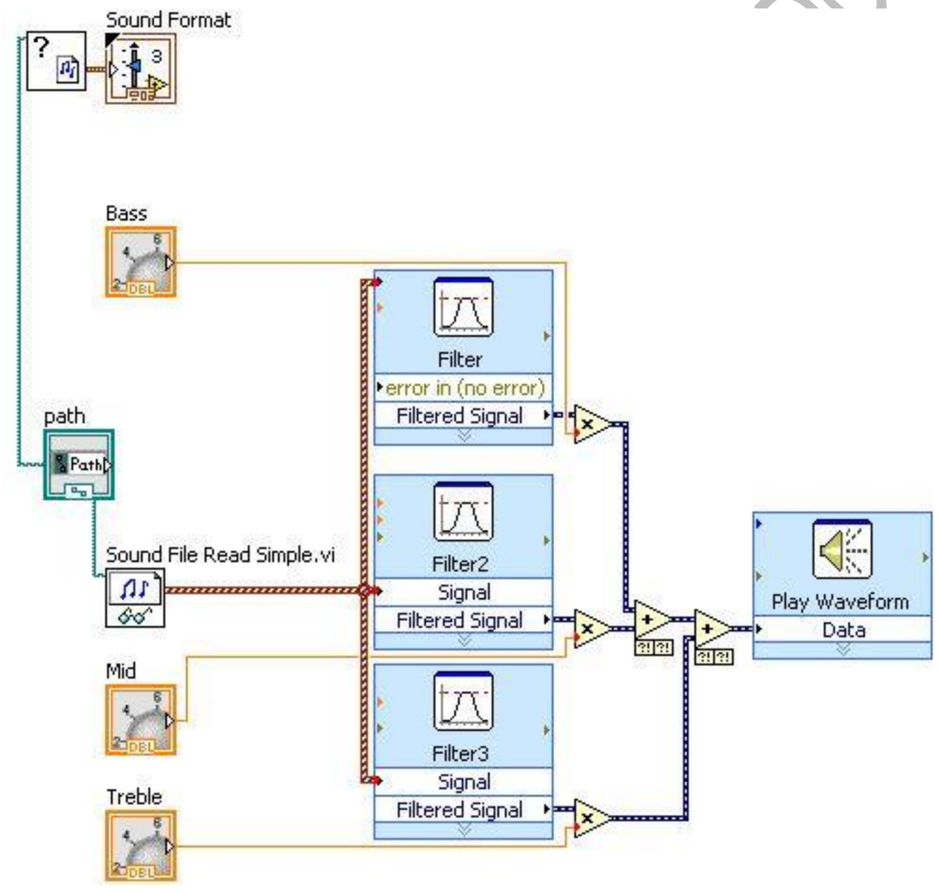
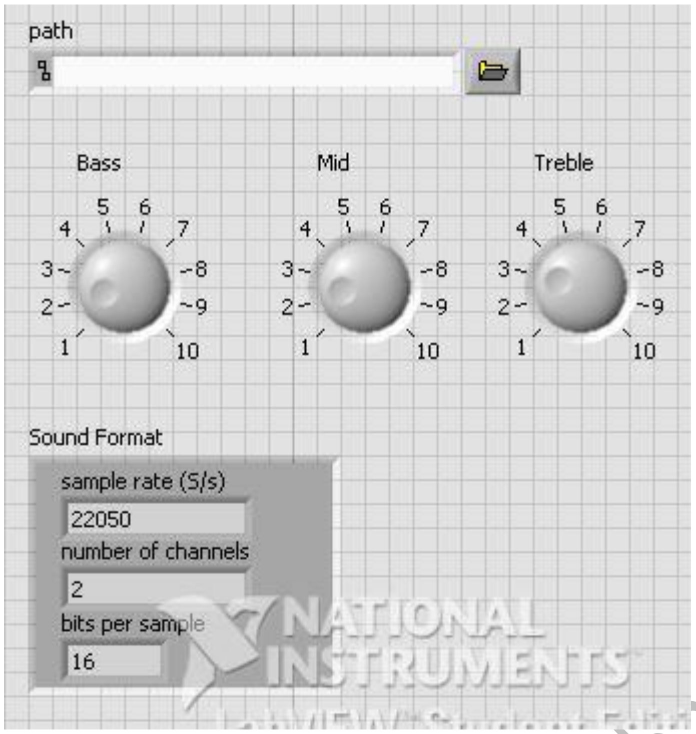


Figure 3: Block Diagram (audio equalizer using the .wav file)



sample rate (S/s)
22050
number of channels
2
bits per sample
16

Figure 4: Front Panel (audio equalizer using the .wav file)

Function: The function Sound File Read simply helps us to read the .wav files. So we created path and used the path to get the .wav audio file. The path was also used to acquire the info about the quality of sound using "Sound File Info". The data acquired from the "Sound File Read Simple" function was passed to three filters. We used three filters since we are making three band equalizer. Our hearing range is usually in between 20 to 20,000 Hz. The chart of audio frequency spectrum is shown in Figure 5. The frequency range approximately between 20Hz to 250Hz are considered as "Bass/Low." The frequency range approximately between 300Hz to 2000Hz are considered as "Mid." The frequency range above 3000Hz are considered as "Treble/High."

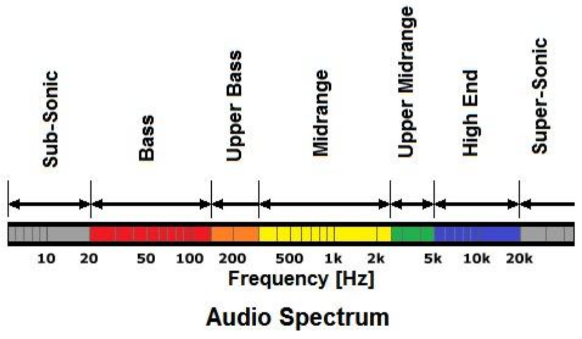


Figure 5: Audio Spectrum

Therefore the settings for each filters were different from each other. The Filter used to capture only the bass spectrum of the audio was Low Pass Filter. Low Pass Filter only allows the audio which are in low spectrum. Our cut off frequency for low pass filter was 400Hz. Figure 6 shows the Low Pass Filter. The Filter used to capture the mid spectrum of the audio was Band Pass filter. It allowed us to capture the sounds from 450Hz to 2500Hz frequency spectrum. Figure 7 shows the Band Pass Filter. The Filter used to capture the high end of the audio spectrum is High pass filter. It allowed us to capture the sounds from 5000Hz and above. Figure 8 shows the High Pass Filter.

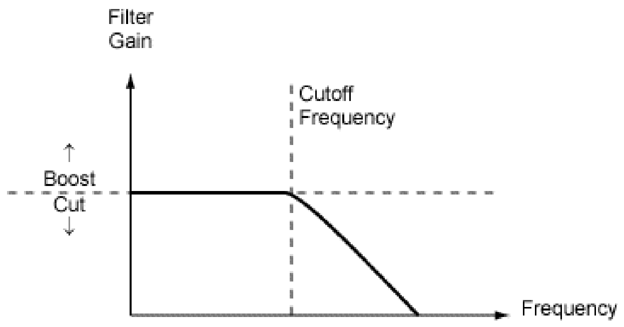


Figure 6: Low Pass Filter

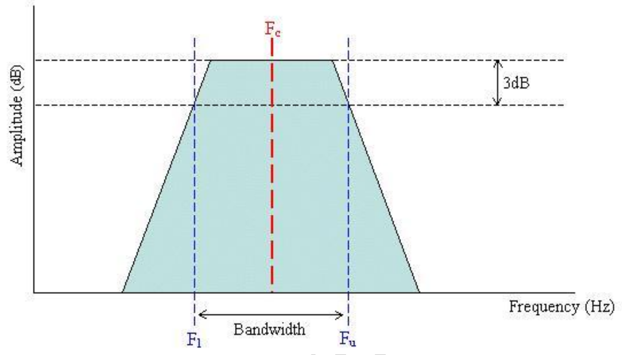


Figure 7: Band Pass Filter

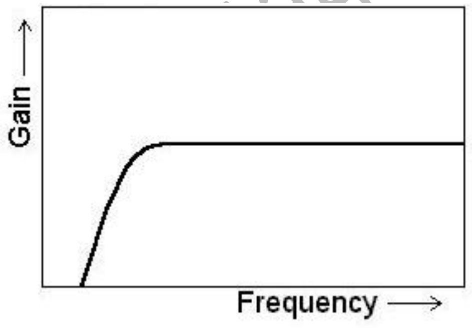


Figure 8: High Pass Filter

We separated the three sections of the sounds and then amplified using multipliers (numeric controls) then we add all the filtered signals together. This multipliers can be controlled using numeric controls. Therefore, we can hear the waveform of the whole audio with amplification using "Play Waveform."

Lessons Learned: In this VI file, we were able to use the .wav file to modify the bass, mid and treble. The problem faced for this VI file is we cannot change the settings of the bass, mid and treble when the song is playing. The settings for low, mid and high was changed before the VI file was played. We were not able to play any other files rather .wav files. Also, when we added the waveform graph in the while loop the graph was not changing with respect to sound. In order to make it happen, we used MyDAQ device to take audio inputs instantly from the cellphone and change the settings of bass, mid and treble whenever we wanted to.

Third Section: Audio Equalizer Using MyDaQ

In this section, we will focus on how to make equalizers using MyDAQ to work on Lab View. Lab View. The Block diagram is shown in Figure 9 and Front panel is shown in Figure 10

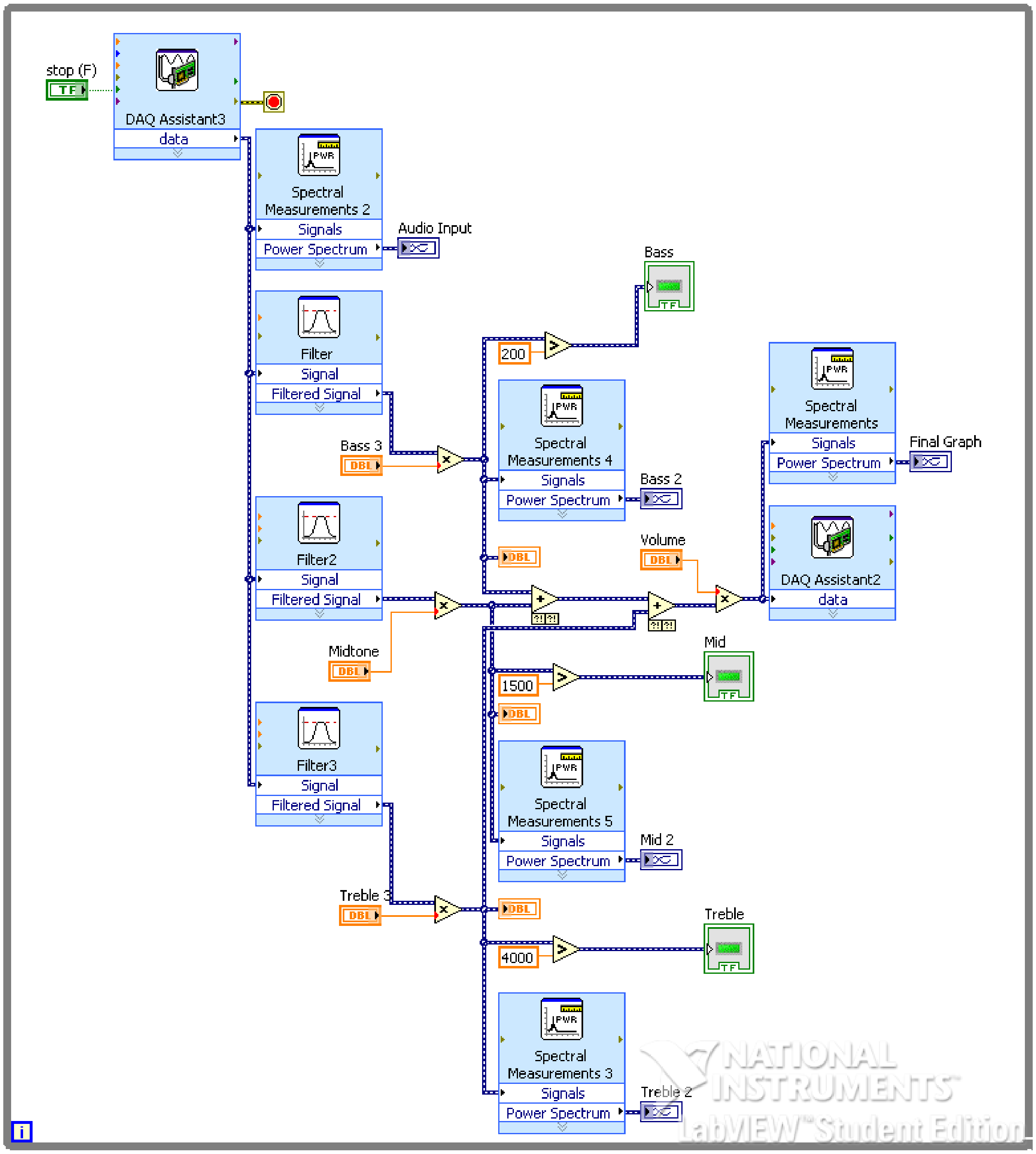


Figure 9: Block Diagram (Audio Equalizer Using MyDaQ)

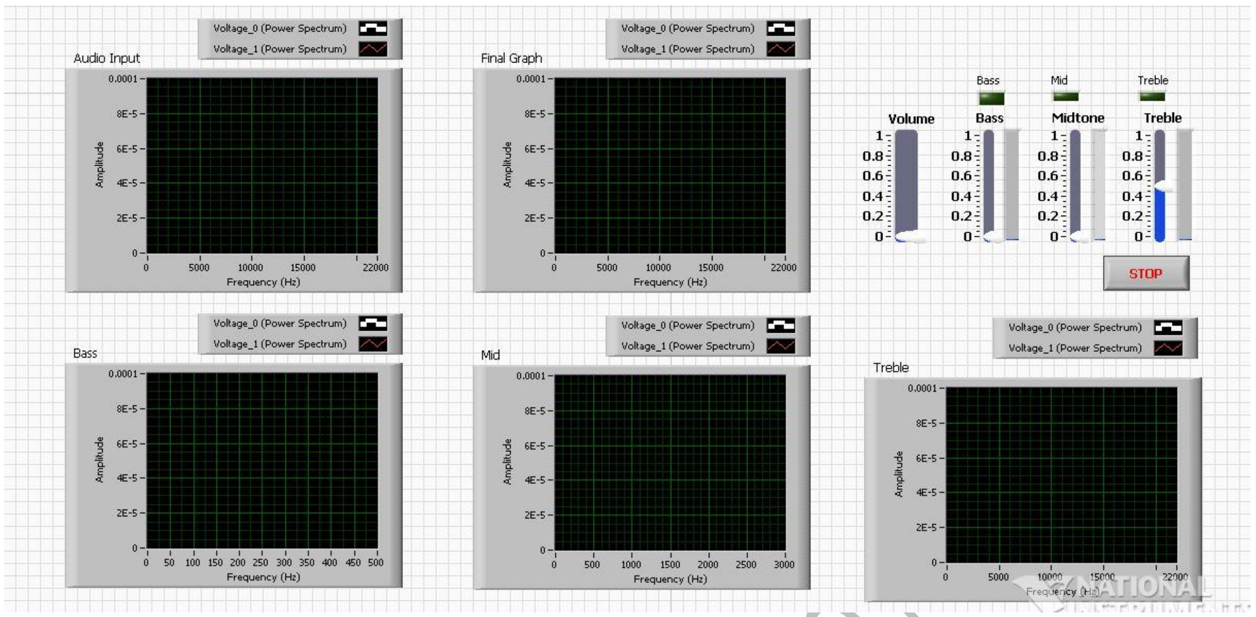


Figure 10: Front Panel (Audio Equalizer Using MyDaQ)

Function: The whole process of making this equalizer was similar to Step 2 instead we used MyDaQ assist as input and output. We would use an extra device known as MyDaQ. MyDaQ is a device that transforms analog to digital. Figure 11 shows MyDaQ. As well as we used volume pointer to control the total volume of the sounds. We used comparators with Boolean functions so that LEDs turns on and off with respect to amplitude in certain range of frequencies. We used the Spectral measurements to change the time domain to frequency domain of the graph

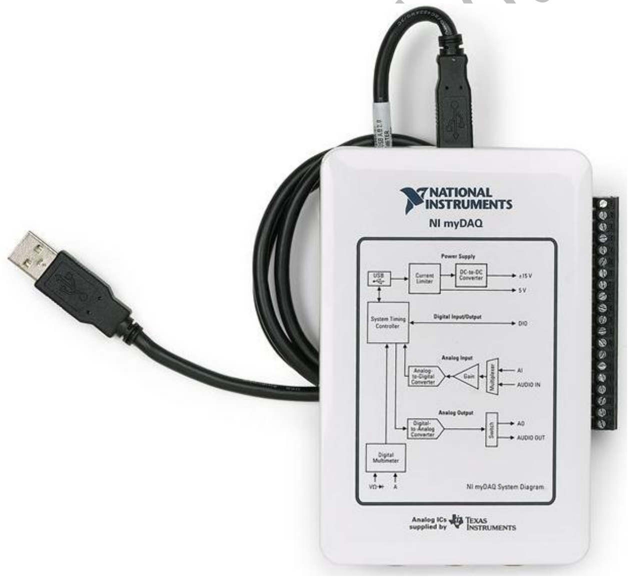


Figure 11: MyDaQ

In other words, MyDaQ helps to acquire audio signals continuously. MyDaQ has audio input and audio output. Therefore, an auxiliary cable was used to connect from the cellphone to the audio in of the Mydaq. Headphones or small speakers was attached to the audio output of the MyDaQ to hear the modified audio signal. In this VI we were able to change the settings of the low, mid and high whenever we wanted to. We made slight modification by adding 3 dB octave equalizer with MyDaQ. The mod addition to the front panel is shown in Figure 12 and mod addition in block diagram is shown in Figure 13.

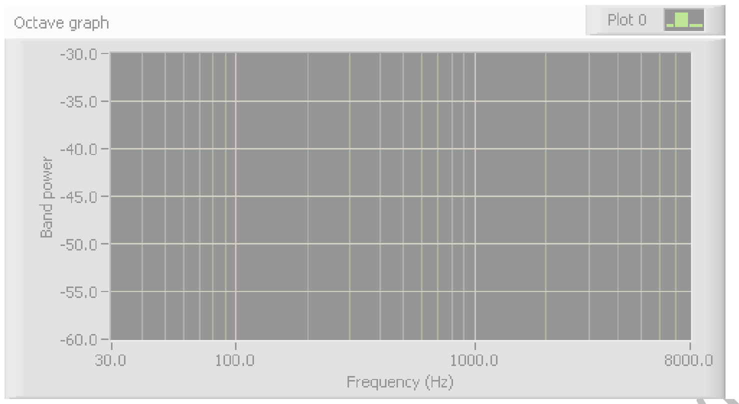


Figure 12: Front Panel with Modifications

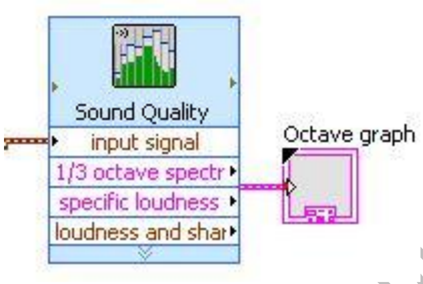


Figure 13: Block Diagram with Modifications

These modification allowed us to see the changes amplitudes changes with respect to frequencies more clearly with better vision.

Fourth Section: Adding Delays

In this section, we would explain how we made delays and how it can be used with audio equalizer even though in this VI file we only can make delays from an audio input. The Front Panel is shown in Figure 14 and the block diagram is shown in Figure 15.

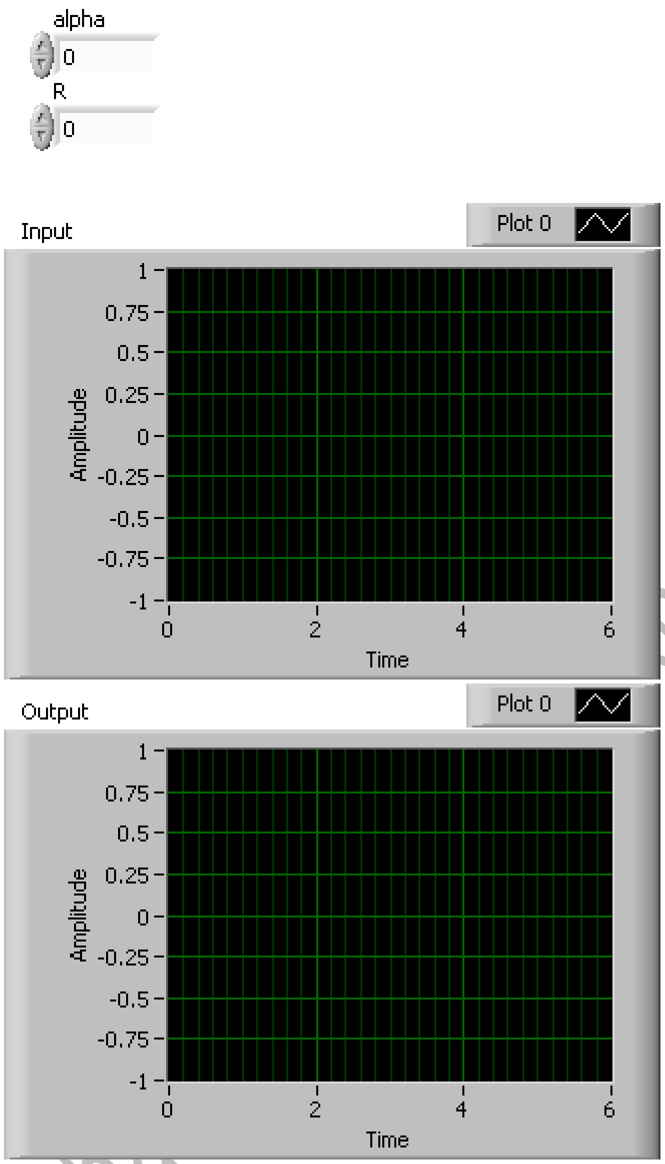
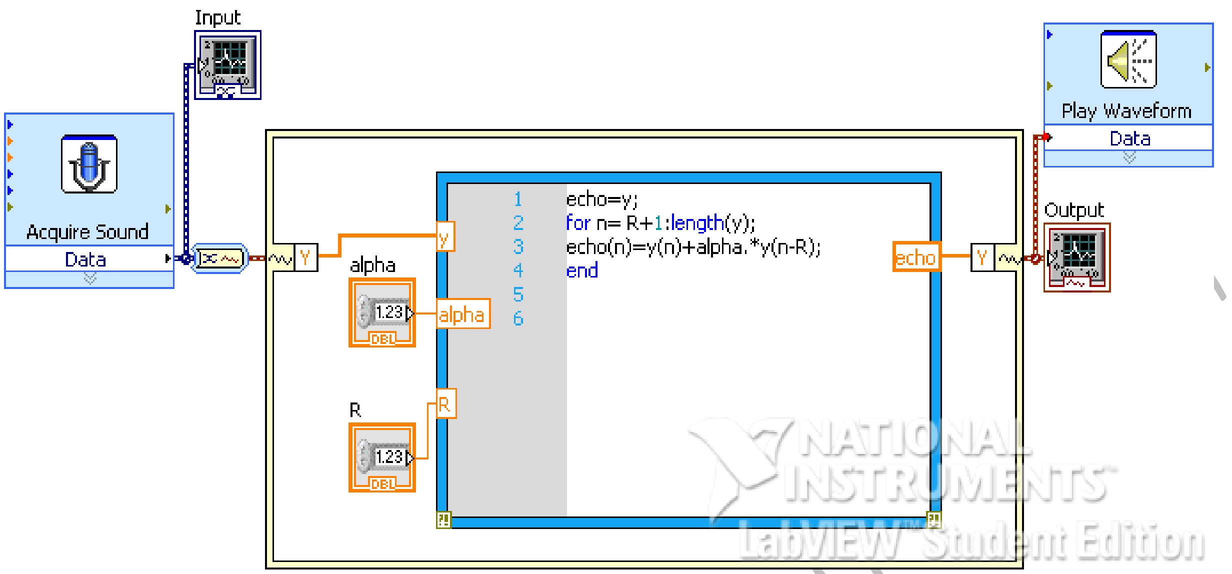


Figure 14: Front Panel (Delay Effect)



```
echo=y;
for n= R+1:length(y);
echo(n)=y(n)+alpha.*y(n-R);
end
```

Figure 15: Block Diagram (Delay Effect)

Function: In this VI, we captured the audio data using a microphone for a certain period of time which is mentioned in the properties of "Acquire Sound." Acquire sounds saves the data and then add echo effect with it. The delays and the amplitude of the echoes are changed by changing the values of R and alpha. In the Figure 15 block diagram, Mat Lab script was added. It helps to use the variable data and modify the sound that is acquired from the input.

Lessons Learned: Even though we were able to make the delay effects due to time management issues we could not apply it with audio equalizer. Even though it is possible to sum up two ideas together.

Fifth Section: Audio Equalizer with Mat Lab

The outlook of the audio equalizer built with Mat Lab is shown in Figure 16.

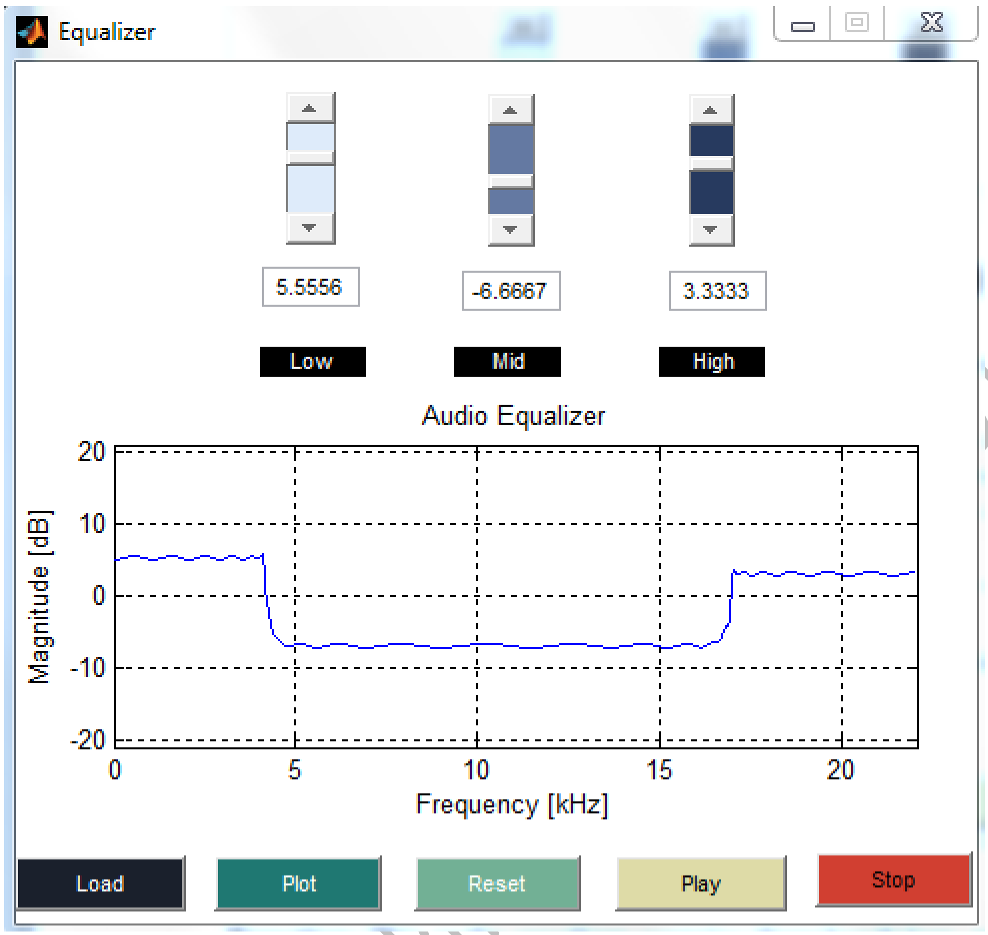
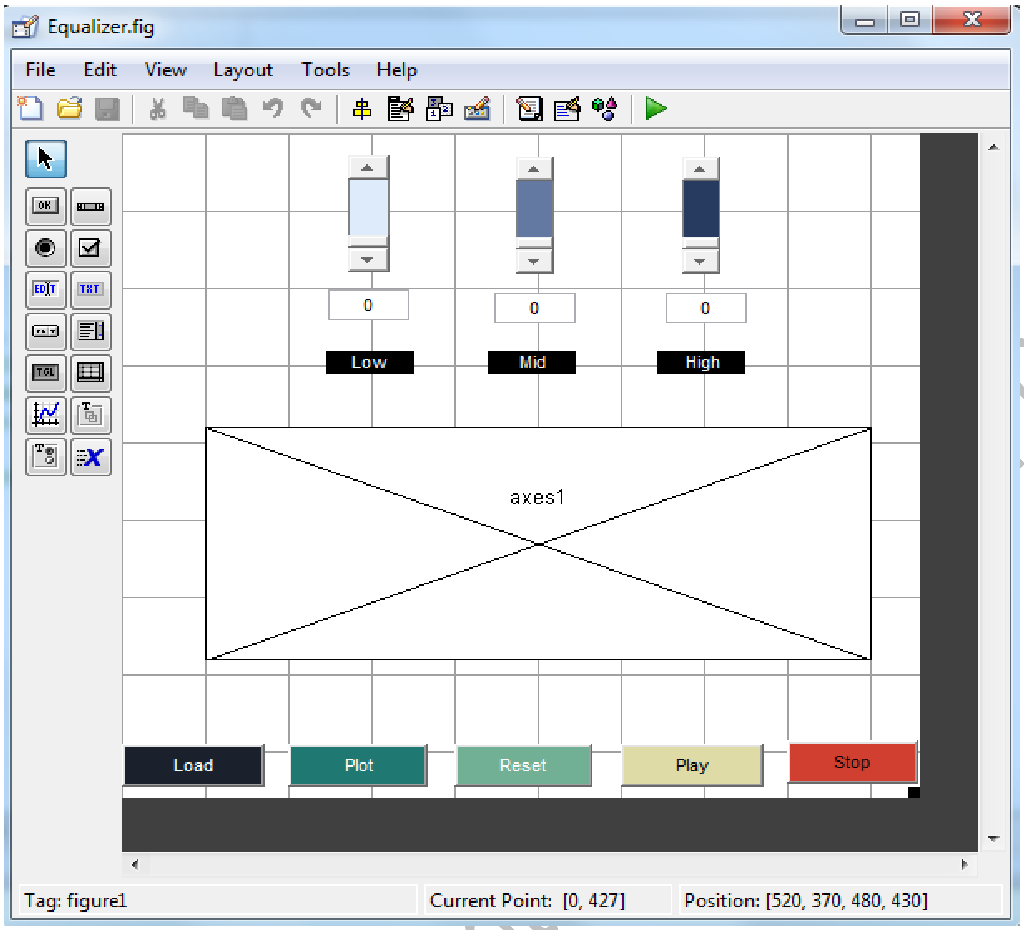


Figure 16: Audio Equalizer Using Mat Lab

Function: This audio equalizer contains three bands low, mid and high. The Slider option allows to choose value of the filter. The plot displays the characteristics of the filter. Load option allows to choose only .wav file to read. When the files is loaded using the option load and the song is played after the clicking button play, the .wav audio starts to play with respect to filter characteristics. When the song is played, the change in slider value changes sound of the audio. However, the plot characteristic change can only be seen after the audio files stops playing or before playing the audio file.



Tag: figure1 Current Point: [0, 427] Position: [520, 370, 480, 430]

Figure 16: GUI created using Mat Lab

The previous figure shows the GUI generated for the audio equalizer. At the top it shows three different slayers, each one of them is used to display the range of the filters. The first slider is the low pass filter which represents the bass of the audio, the second slider is a band-pass filter which represent the mid of the audio and finally the last slider is a high-pass filter which represents the treble of the audio. In addition, the guide has different static text used to name the different sliders and they also have edit text in order to display the exact number representation which arrange by the actual slider. Also the GUI has five different push buttons, each one of them symbolizes a function. The first push button "Load" is used to insert an audio file, where the user has the option to upload an audio file from the computer. The second push button "plot" is used to display the different filters and frequencies that are set up by the sliders. The third push button "Reset" is used to reset the graph and the sliders back to zero which is the established value. The fourth button "Play" is used to start the audio file and plays as long as the file audio is playing. The fifth button "Stop" is used only to stop the audio file that is playing. Finally the GUI has an added graph which displays the different filter values by their frequency and their magnitude as decibels.

All the different components used in GUI were also re arranges to display a certain color, a unique display size, also the string texts were changed to their respective names and a tag name was also added in order to recall the component throughout the code.

The Code for the Matlab with Explanation:

The following code represents all the programming functions that are connected to the previous GUI. A lot of the code is automatically generated by Mat lab which represents the GUI itself and the different components added.

In order to describe the functionality of the audio equalizer's code, the code will be divided by the different functions and an explanation would be add before each function.

1. The function "varargout" is an automatically generated function which calls on the GUI and describes some general state functions that will be derive along the final code.

```
    function varargout = Equalizer(varargin)
% EQUALIZER MATLAB code for Equalizer.fig
%      EQUALIZER, by itself, creates a new EQUALIZER or raises the existing
%      singleton*.
%
%      H = EQUALIZER returns the handle to a new EQUALIZER or the handle to
%      the existing singleton*.
%
%      EQUALIZER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EQUALIZER.M with the given input arguments.
%
%      EQUALIZER('Property','Value',...) creates a new EQUALIZER or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Equalizer_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Equalizer_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Equalizer

% Last Modified by GUIDE v2.5 14-Apr-2014 14:25:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Equalizer_OpeningFcn, ...
                   'gui_OutputFcn',  @Equalizer_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

2. The function "Equalizer_Opening" is the function that represents the range and value of the sliders and their respective edit text by using the set function. The tag of the low pass slider and edit text is (C1_var and C1_val) respectively and their range is determined by the min value -20, a max value 20, the standard value 0 and a string which represents the edit text input that would be display into the graph and the slider, by using the function num2str. Also the global functions where introduce by defining a sample frequency (Fs), a stop function which determines when the audio file is stop and there is a C vector that will be use later on the code. Such global functions are used in order to recall variables throughout the whole code without having to re state the variables.

```
% --- Executes just before Equalizer is made visible.
function Equalizer_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Equalizer (see VARARGIN)
global stop C Fs;
stop=1;
Fs=44100;
C=zeros(1,3);
set(handles.C1_var,'min',-20);
set(handles.C1_var,'max',20);
set(handles.C1_var,'value',0);
set(handles.C1_var,'SliderStep',[0.025,0.05]);
set(handles.C1_val,'string',num2str(0));


set(handles.C3_var,'min',-20);
set(handles.C3_var,'max',20);
set(handles.C3_var,'value',0);
set(handles.C3_var,'SliderStep',[0.025,0.05]);
set(handles.C3_val,'string',num2str(0));



set(handles.C5_var,'min',-20);
set(handles.C5_var,'max',20);
set(handles.C5_var,'value',0);
set(handles.C5_var,'SliderStep',[0.025,0.05]);
set(handles.C5_val,'string',num2str(0));

equalizer_plot();


% Choose default command line output for Equalizer
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Equalizer wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

3. Function "varargout = equalizer" is also an automatically generated function which describe the function that are returned into the command line by the application of cells.

```
% --- Outputs from this function are returned to the command line.
function varargout = Equalizer_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

4. Function "load_Callback" is the execution of pressing the load on the GUI which represents the user's option to go into their computer to find an audio file (wav audio files are only read). Such execution is done by using the uigetfile variable which brings into the GUI an external file, in this case an audio file which is identifies by the name file_name. The file_name would be recall later on the code by using the global variable.

```
% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
% hObject    handle to load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
file_name=uigetfile('*wav');
```

5. Function "plot_H_Callback" is the execution when pressing the plot button on the GUI which recalls the equalizer_plot() function which will be describe later in the code.

```
% --- Executes on button press in plot_H.
function plot_H_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to plot_H (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global C Fs;
equalizer_plot();
```

6. The function "equalizer_plot" is use to assign the variables, names and values that are displayed at the graph of the GUI. Here a for loop is implemented in order to determine the three different frequency response vectors (H), which are found by calculating the frequency supply by the (a,b) coefficients and 1024 samples, by using the freqz variable and multiply the input value C. The (a,b) coefficients will be explain in the next section and the value of C corresponds to the input value generated by the sliders.

   The plot consists on displaying the frequency ranges which is determine by the most common audio frequency range plot at the x- axis going from (0kHz to 22kHz) and the frequency response displayed in the decibels on the y-axis. In addition to the graph, different labels where assign to the graph such as the Magnitude in the y axis and the frequency on the x axis.

```
function equalizer_plot()
global C Fs;
[a,b]=coef();
H=0;
for i=1:3
    H=H+10^(C(i)/20)*abs(freqz(b{i},a{i},1024));
end
plot(1e-3*Fs*[0:1023]/2048,20*log10(H));
xlabel('Frequency [kHz]');
ylabel('Magnitude [dB]');
title('Audio Equalizer');
axis([0 1e-3*Fs/2 -21 21]);
grid on;
```

7. The function [a,b]=coef() is used in order to create the filters used in the graph. Here the variable cheb1ord and cheby1 are used. The cheb1ord describes the filter into four different characteristics, such as the pass band corner frequency, the stop band corner frequency, the attenuation of the stop band and the pass band ripples. In other words the pass band is the frequencies that are going to be allow into filter and the stop band is the attenuation of the filter which rejects some frequencies depending on the filter. For a low pass filter the pass band allows small frequencies to pass and attenuates high frequencies. The attenuation stop band is the distance between the pass band and the stop band. Then we used the variable cheby1 which generates a Chebyshev filter that consist on determining the exact filter that is being used such as low, high or band filter.

```
function [a,b]=coef()
global Fs;
% Filter 1
Rp1=0.5;
Rs1=30;
Fp1=4.1e3/(Fs/2);   %%passband
```

```
Fs1=4.5e3/(Fs/2);    %%stopband
n1=cheb1ord(Fp1,Fs1,Rp1,Rs1); %% filter order 1
[b1,a1]=cheby1(n1,Rp1,Fp1,'low'); %% 'low' to determine what kind of filter
it is (low filter)

%Filter 2
Rp3=0.5;
Rs3=30;
Fp3=1e3*[4.25,16.8]/(Fs/2);
Fs3=1e3*[3.9,17.5]/(Fs/2);
n3=cheb1ord(Fp3,Fs3,Rp3,Rs3);
[b3,a3]=cheby1(n3,Rp3,Fp3);  %(band pass)


%Filter 3
Rp5=0.5;
Rs5=30;
Fp5=1e3*17/(Fs/2);
Fs5=1e3*17.4/(Fs/2);
n5=cheb1ord(Fp5,Fs5,Rp5,Rs5);
[b5,a5]=cheby1(n5,Rp5,Fp5,'high'); %% High passs
a={a1,a3,a5};
b={b1,b3,b5};
```

8. Function “reset_Callback” is the execution of pressing the reset button on the GUI, which sets up the values of sliders and their respective edit test back to zero, including the graph.

```
      % --- Executes on button press in reset_H.
function reset_H_Callback(hObject, eventdata, handles)
% hObject    handle to reset_H (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Fs C;
C=zeros(1,5);
Fs=44100;
set(handles.C1_val,'string',num2str(0));
set(handles.C3_val,'string',num2str(0));
set(handles.C5_val,'string',num2str(0));

set(handles.C1_var,'value',0);
set(handles.C3_var,'value',0);
set(handles.C5_var,'value',0);

equalizer_plot();
```

9. The function “play Callback” is use to run the file that has been upload by the load function.

```
% --- Executes on button press in play.
function play_Callback(hObject, eventdata, handles)
% hObject    handle to play (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global stop file_name C;
stop=1;
equalizer_play();
```

10. The function “equalizer_play()” is used to cut off the complete file or the complete frequency into small segments. The segments are based on cutting of the frequency range into the frequency sample. The small segments will be used as a delay whenever there is a change in the values of the filters. In this case the filter variable is (filter) which has the same characteristics of the filters used previously, but here the filters are only related to the audio file frequency characteristics. Every time the values of the sliders are change the new values are saved as variables by the ([a,b]coff) which have some fix characteristics described by the cheb1ord function, then the filtering variable uses the new coefficients to change the audio file sound features. Using a for loop help us run the three different filters at the same time while making sure the playing function stops as the song is over or as the stop button is press (stop==0).

```
function equalizer_play()
global stop file_name C;
[x,Fs]=wavread(file_name); %% x is the data and Fs is the sampling frequnecy
[a,b]=coef();
l_segment=2*Fs;
Nb=round(length(x)/l_segment);
y=0;
for i=1:floor(Nb)
    segment=x((i-1)*l_segment+1:i*l_segment);
    for k=1:3
        y=y+filter(10^(C(k)/20)*b{k},a{k},segment); %%Amplifiyng the signal,
(B = numerator, A=denominator)
        if(stop==0)
            break;
        end
    end
    wavplay(y,Fs,'async');
    y=0;
    if(stop==0)
        break;
    end
end
```

11. The following functions “C1_var_Callback”, “C3_var_Callback” and the “C5_var_Callback” are based on describing the vector C. The vector C will take values from the three different sliders by defining the exact magnitude response according to the value of the sliders. This vector C will be recall throughout the rest of the code by using the variable “Global”. These functions are also interrelated with the C(X)_val_Callback functions in order to exchange their values with one another. To save the values as a strings the values must be change from numbers to strings by using the num2str variable.

```
% --- Executes on slider movement.
function C1_var_Callback(hObject, eventdata, handles)
% hObject    handle to C1_var (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global C;
C(1)=get(hObject,'value');
set(handles.C1_val,'string',num2str(C(1)));

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

% --- Executes on slider movement.
function C3_var_Callback(hObject, eventdata, handles)
% hObject    handle to C3_var (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global C;
C(2)=get(hObject,'value');
set(handles.C3_val,'string',num2str(C(2)));

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider


% --- Executes on slider movement.
function C5_var_Callback(hObject, eventdata, handles)
% hObject    handle to C5_var (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider
global C;
C(3)=get(hObject,'value');
set(handles.C5_val,'string',num2str(C(3)));
```

12. The following functions “C1_val_Callback”, “C3_val_Callback” and the “C5_val_Callback” are based to describe the values on the edit texts by setting the value into a string and changing the position of the sliders according to the values. It uses an if loop in order to keep the range between the min and the max values, in case the value of the C(x)_val is higher or lower than the minimum range the string would save a 0 value which would make the sliders and the edit text display to reset to zero.

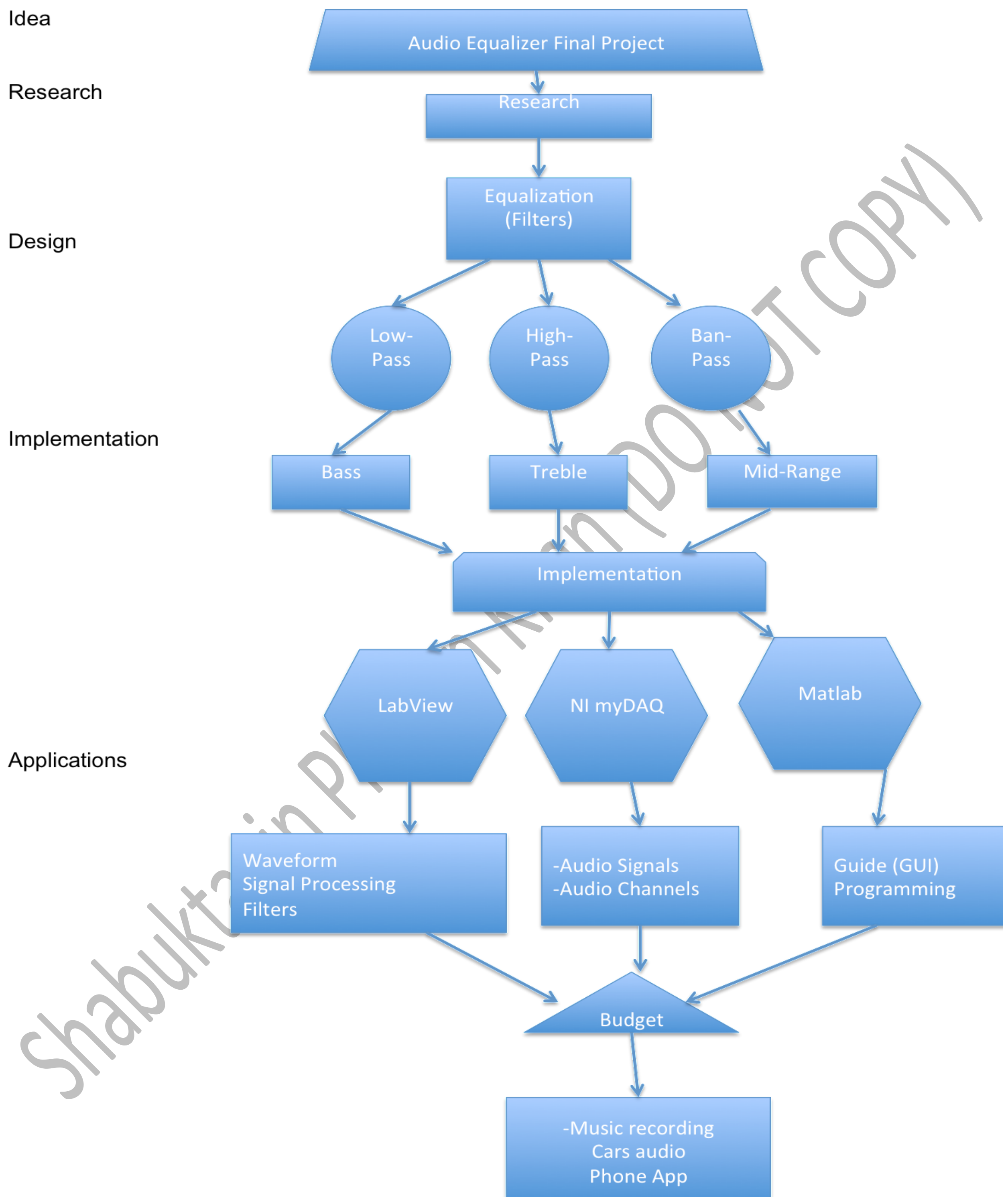
```
function C1_val_Callback(hObject, eventdata, handles)
% hObject    handle to C1_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C1_val as text
%        str2double(get(hObject,'String')) returns contents of C1_val as a
double
global C;
C(1)=str2num(get(hObject,'string'));
```

```
minn=get(handles.C1_var,'min');
maxx=get(handles.C1_var,'max');
if(C(1)<minn || C(1)>maxx)
    C(1)=get(handles.C1_var,'value');
    set(hObject,'string',num2str(0));
else
    set(handles.C1_var,'value',C(1));
end

% --- Executes during object creation, after setting all properties.
function C1_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C1_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function C3_val_Callback(hObject, eventdata, handles)
% hObject    handle to C3_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C3_val as text
%        str2double(get(hObject,'String')) returns contents of C3_val as a
double
global C;
C(2)=str2num(get(hObject,'string'));
minn=get(handles.C3_var,'min');
maxx=get(handles.C3_var,'max');
if(C(2)<minn || C(2)>maxx)
    C(2)=get(handles.C3_var,'value');
    set(hObject,'string',num2str(0));
else
    set(handles.C3_var,'value',C(2));
end



function C5_val_Callback(hObject, eventdata, handles)
% hObject    handle to C5_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C5_val as text
%        str2double(get(hObject,'String')) returns contents of C5_val as a
double
global C;
C(3)=str2num(get(hObject,'string'));
minn=get(handles.C5_var,'min');
maxx=get(handles.C5_var,'max');
if(C(3)<minn || C(3)>maxx)
    C(3)=get(handles.C5_var,'value');
    set(hObject,'string',num2str(0));
else
    set(handles.C5_var,'value',C(3));
end
```

13. The function "stop_Callback" is based stopping the audio file as the stop button is click or as the audio file length is compete.

```
% --- Executes on button press in stop.
function
stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global stop;
stop=0;
```

State Diagram



Summary and Conclusions

Problems in the Implementation

The problem throughout the process was to figure out how to constantly make the sounds alter in Lab View and see the changes in waveform which took most of our time. Also, implementing the plot and the graph in Mat Lab Graphical User Interface was not working properly at the beginning.

Team Work

With this project, we actually learned the importance of working as a team. First we decided to understand the abilities and skills of one another and how could they be applying into the project. We all knew something better at one topic than one another. With that being said we combined everything we knew at our best making the project a lot easier to work. Then we learned each other's schedule to see when meeting up was more appropriate for every one on the team. Meeting up together was quite hard as we all had very busy schedules so we decided to work on it separately, so whenever we would meet up we could exchange ideas and experiences. We also learned that communication is a key for success, as we were in contact with other at all times by using web programs such as email and Google drive so we could share information and ideas with each other.

References

Websites

Lab View
http://www.ni.com/white-paper/11433/en/
http://cnx.org/content/m14772/latest/lvt_audio-realtime-output.html

Mat Lab
http://www.mathworks.com/matlabcentral/fileexchange/23982-digital-audio-equalizer

Images
http://www.songsofthecosmos.com/encyclopedia_of_modern_music/A/audio_spectrum.html
http://en.wikipedia.org/wiki/Low-pass_filter
http://en.wikipedia.org/wiki/Band-pass_filter
http://en.wikipedia.org/wiki/High-pass_filter