Student Name: Yash Shah
Student Number: 2078614

MATLAB Based Graphic Equaliser

Basic Overview of Design

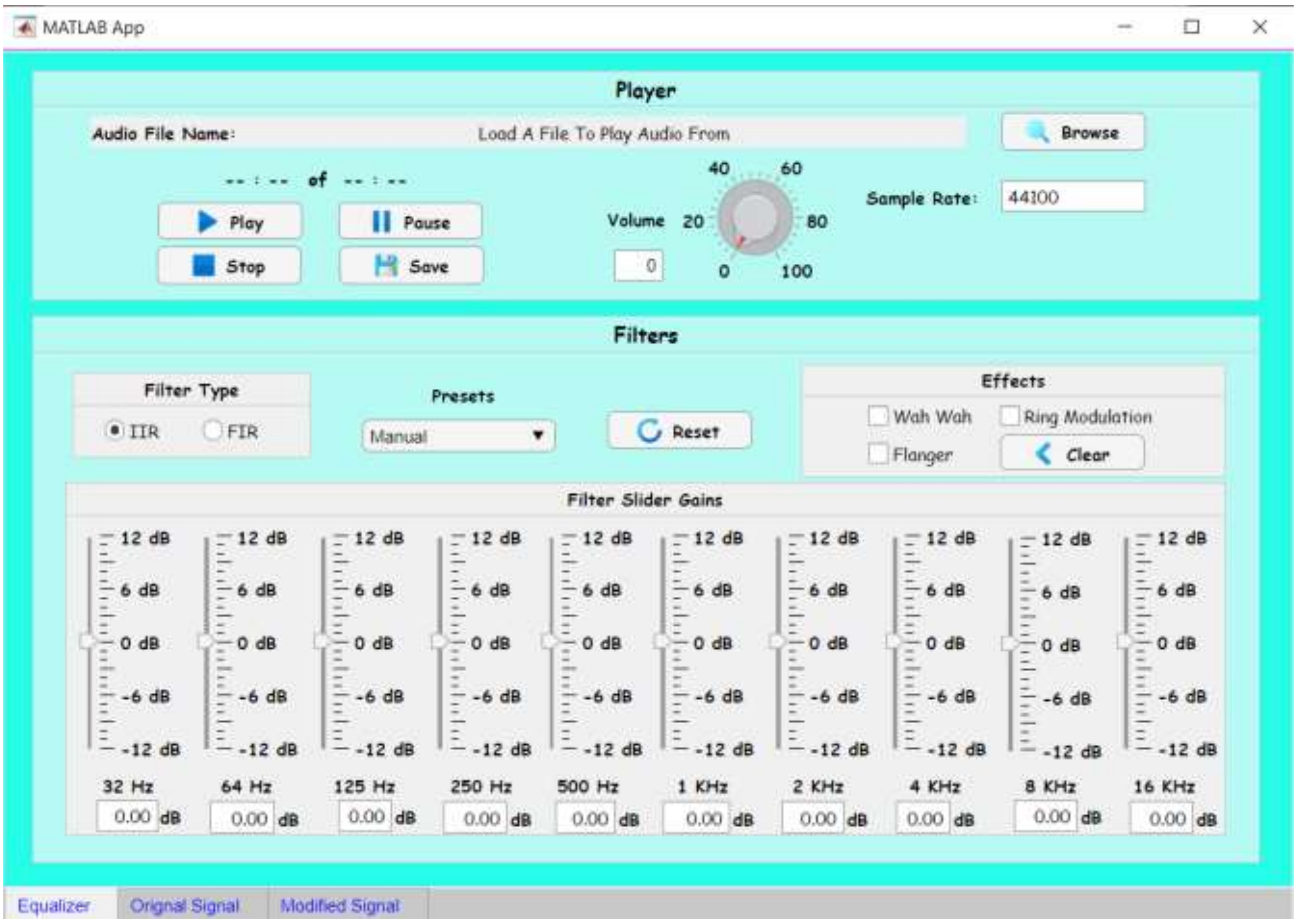
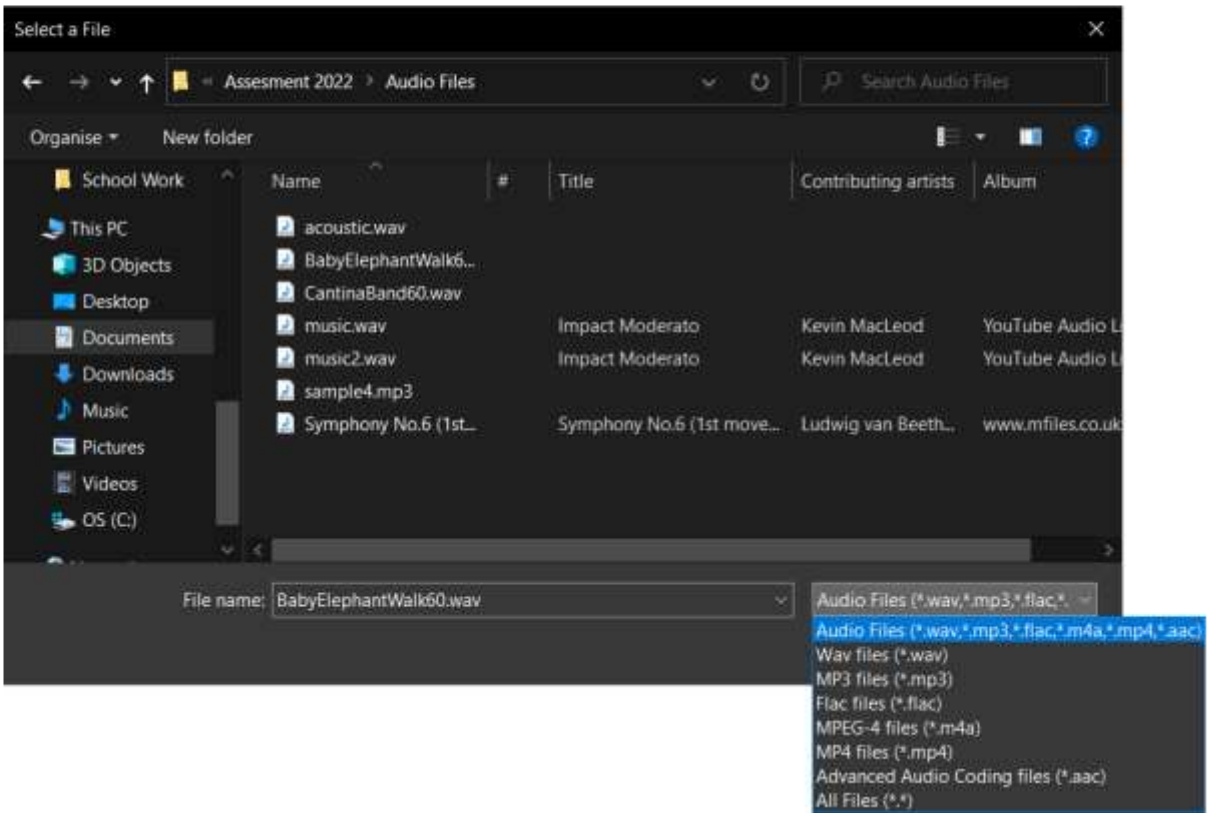


Figure 1

When the user runs the program, they are presented with a view as presented in **Figure 1**. The user is presented with many options in which the user first needs to load an audio file. Keeping the options flexible, the user can choose between a variety of audio file they are able to load in. For the volume a knob has been used which is similar to a volume knob in the real world. The user is able to change the sample rate at which the file is read by modifying the value in the sample rate field if not the default value of 44100Hz is used. The user is then able to play the audio file using the play button or the user can pause the audio file as well as completely stop the audio file from playing using the stop button. Giving the user flexibility the user can save the audio file after they have added filters to the original audio in which a new audio file is created and saved in the users current working directory. The user is able to choose between two different filter types IIR and FIR depending on the user's needs. If the user is not concerned about linear phase characteristic, they can use IIR otherwise FIR would be the choice for linear-phase characteristics. Each filter type has its pros and cons but IIR is better for lower-order tapping, were as FIR is suitable for higher-order tapping. Making it easier for the user some common pre-sets have been preconfigured which will automatically adjust the sliders based on the pre-set chosen or the user can choose to manually adjust each slider. Saving the user time, a reset button has been added which will reset all the sliders back to zero. Lastly for the extra features the user can apply different effects to the audio by using the check boxes.

Student Name: Yash Shah
Student Number: 2078614

Steps to get audio setup



Name	#	Title	Contributing artists	Album
acoustic.wav				
BabyElephantWalk6...				
CantinaBand60.wav				
music.wav		Impact Moderato	Kevin MacLeod	YouTube Audio L
music2.wav		Impact Moderato	Kevin MacLeod	YouTube Audio L
sample4.mp3				
Symphony No.6 (1st...		Symphony No.6 (1st move...	Ludwig van Beeth...	www.mfiles.co.uk

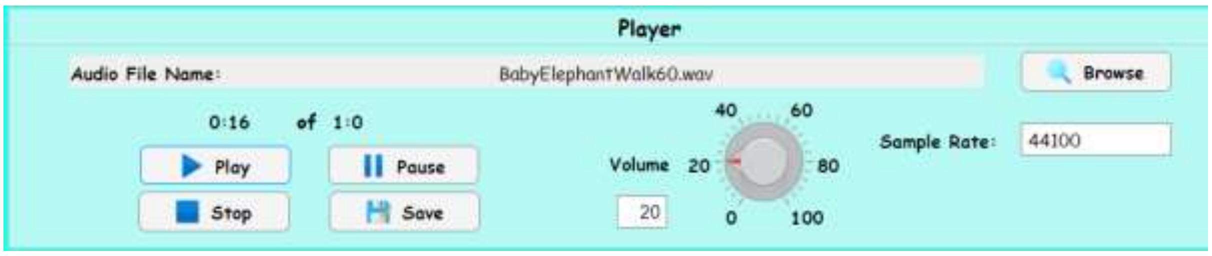
File name: BabyElephantWalk60.wav

Audio Files (*.wav,*.mp3,*.flac,*.

Audio Files (*.wav,*.mp3,*.flac,*.m4a,*.mp4,*.aac)
Wav files (*.wav)
MP3 files (*.mp3)
Flac files (*.flac)
MPEG-4 files (*.m4a)
MP4 files (*.mp4)
Advanced Audio Coding files (*.aac)
All Files (*.*)

Figure 2

User first chooses a file as can be seen in **Figure 2**. Making the program more dynamic the user is able to choose between a variety of audio file types. File types present are: wav, flac, MP3 and MPEG-4 AAC.



Audio File Name: BabyElephantWalk60.wav

0:16 of 1:0

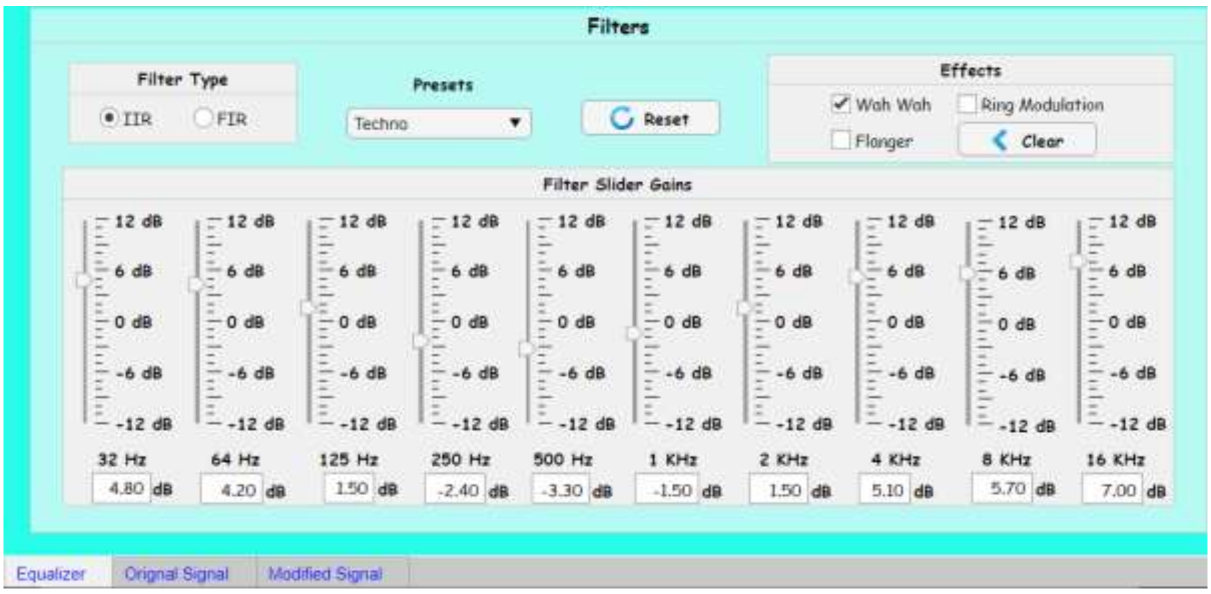
Volume: 20

Sample Rate: 44100

Figure 3

Once an audio file has been chosen the file name of the audio file is displayed to the user as can been seen in **Figure 3**. The user can the adjust the volume to their preference and also change the sample rate at which the file is being read. Once the user has adjusted the necessary setting the user can play the audio file. When the user plays the audio file a timer is displayed which shows the current playing time and the total time of the audio file.

Student Name: Yash Shah
Student Number: 2078614



Filter Type: [✓ IIR] FIR
Presets: Techno
Effects: [✓ Wah Wah] Ring Modulation Flanger

32 Hz	64 Hz	125 Hz	250 Hz	500 Hz	1 KHz	2 KHz	4 KHz	8 KHz	16 KHz
4.80 dB	4.20 dB	1.50 dB	-2.40 dB	-3.30 dB	-1.50 dB	1.50 dB	5.10 dB	5.70 dB	7.00 dB

Figure 4

Once the user has loaded a file the user can choose a Filter Type, the user can select a pre-set and an audio effect to apply to the audio. The audio signal graphs of the original audio signal and the modified audio signal after applying different filters and effects can be view in the tabs that can be seen at the bottom in **Figure 4.**

Original Signal

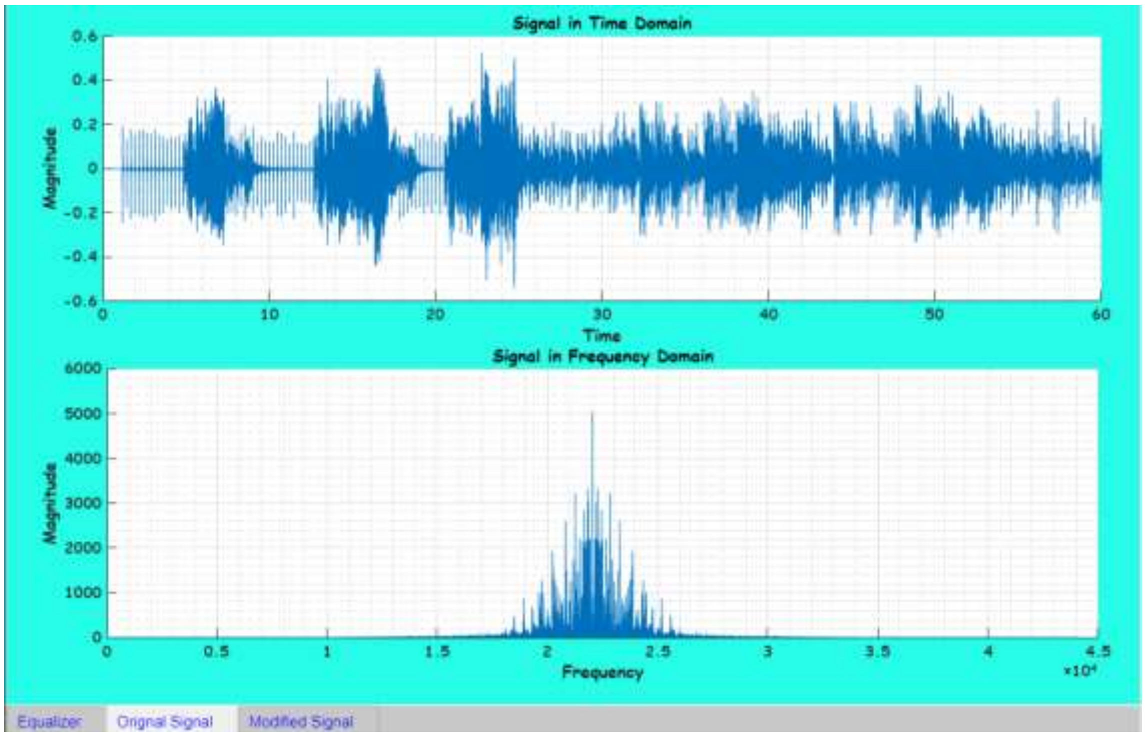


Figure 5

Student Name: Yash Shah
Student Number: 2078614

Modified Signal

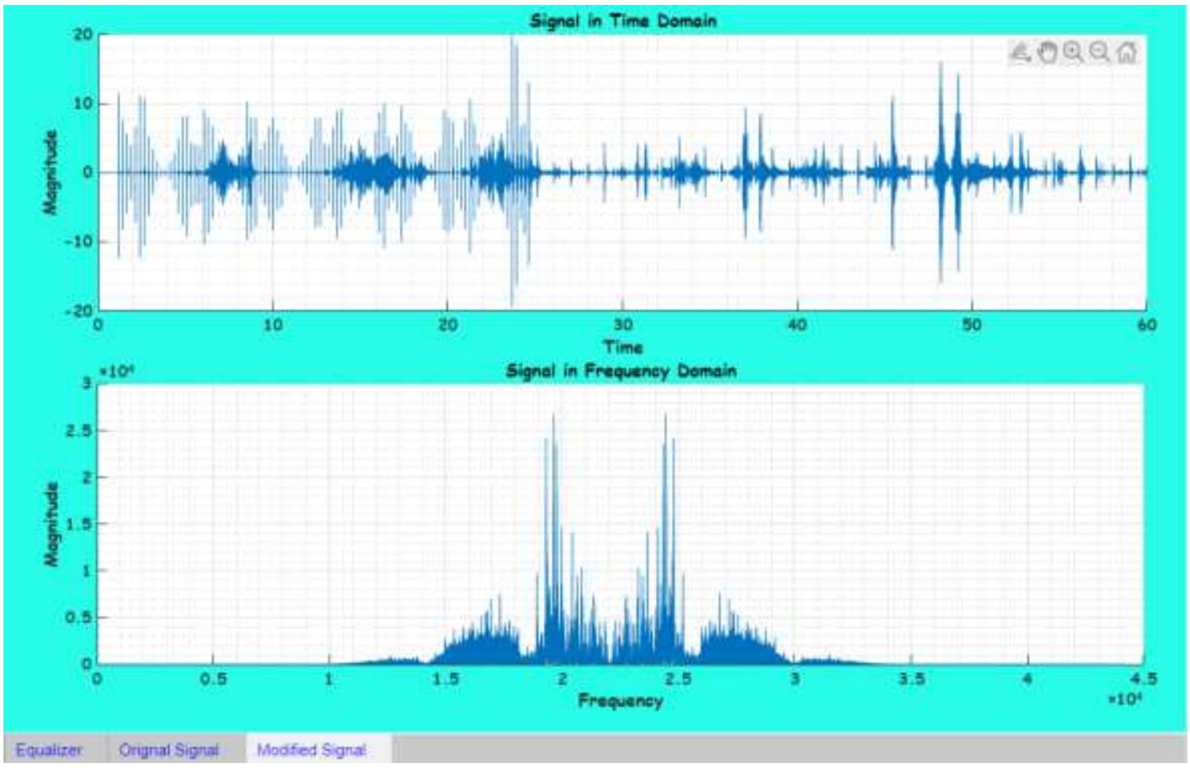
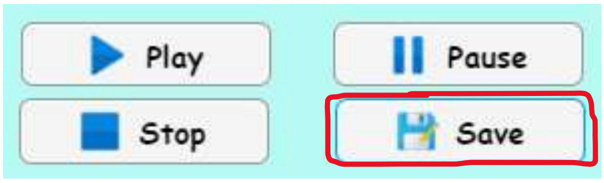


Figure 6

Clicking the Save Button



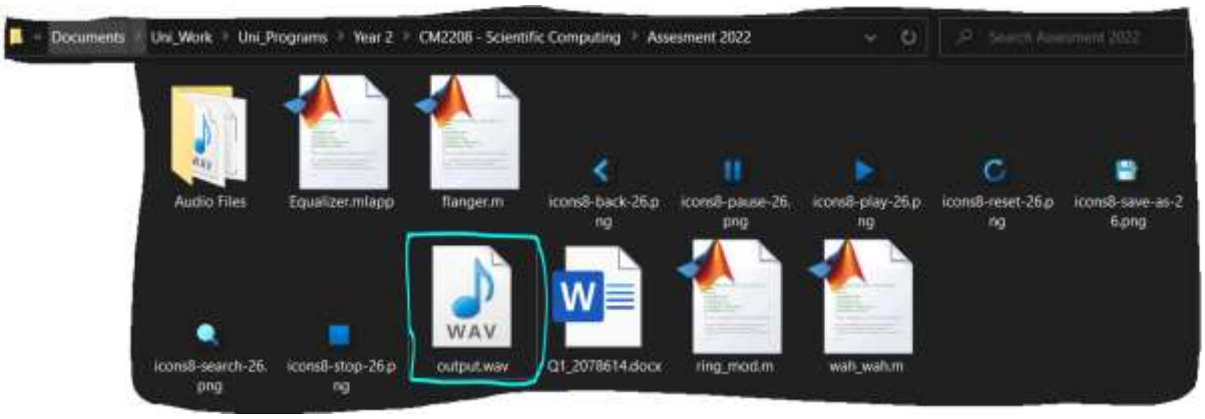
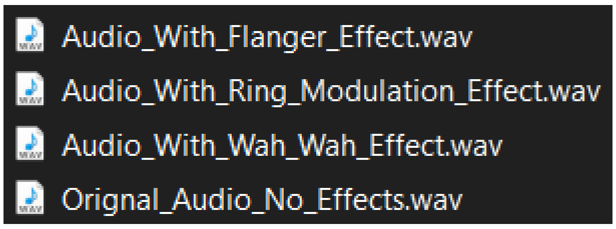


Figure 7

Once the user has applied effects to the audio file, they can save this new audio file. The new audio file is saved as "output.wav" (shown in **Figure 7**) in the current working directory which is opened in MATLAB.



Audio_With_Flanger_Effect.wav
Audio_With_Ring_Modulation_Effect.wav
Audio_With_Wah_Wah_Effect.wav
Orignal_Audio_No_Effects.wav

Some of the saved files with different effects can be found in the folder shown.

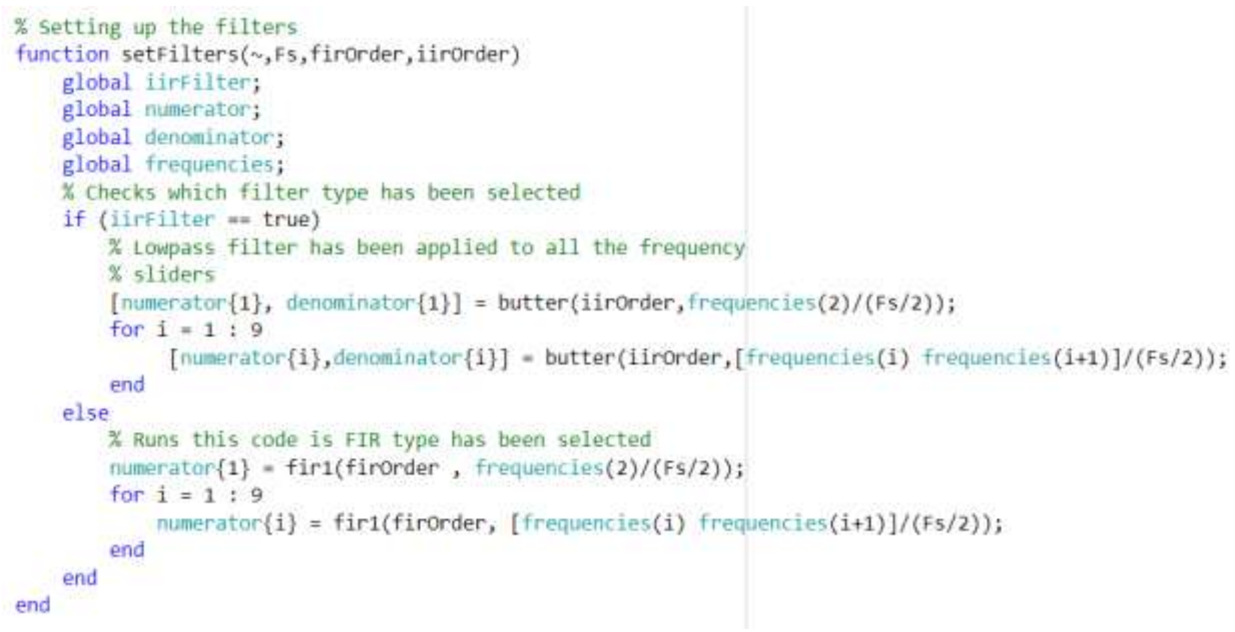
Student Name: Yash Shah
Student Number: 2078614

Basic Requirements Met

```
% Button pushed function: BrowseButton
function BrowseButtonPushed(app, event)
    % Allowing the user to load in an audio file
    [file,path] = uigetfile({'*.wav;*.mp3;*.flac;*.m4a;*.mp4;*.aac',['Audio Files ' ...
        '(*.wav,*.mp3,*.flac,*.m4a,*.mp4,*.aac)'];'*.wav;','Wav files (*.wav)'; ...
        '*.mp3;','MP3 files (*.mp3)';'*.flac;','Flac files (*.flac)';'*.m4a;', ...
        'MPEG-4 files (*.m4a)';'*.mp4;','MP4 files (*.mp4)';'*.aac;', ...
        'Advanced Audio Coding files (*.aac)';'*.*','All Files (*.*)'},'Select a File');
    % Storing file path and audio file name to display it to
    % the user so they know which file they are current playing.
    if ~isequal(file,0)
        app.file_path = fullfile(path,file);
        app.directoryLabel.Text = file;
        app.playing = false;
        app.paused = false;
    end
end
```

Figure 8

As shown before in **figure 2** and **figure 8** the user is able is able to select from different file formats and the system is able to read those different formats and play it back to the user.



```
% Setting up the filters
function setFilters(~,Fs,firOrder,iirOrder)
    global iirFilter;
    global numerator;
    global denominator;
    global frequencies;
    % Checks which filter type has been selected
    if (iirFilter == true)
        % Lowpass filter has been applied to all the frequency
        % sliders
        [numerator{1}, denominator{1}] = butter(iirOrder,frequencies(2)/(Fs/2));
        for i = 1 : 9
            [numerator{i},denominator{i}] = butter(iirOrder,[frequencies(i) frequencies(i+1)]/(Fs/2));
        end
    else
        % Runs this code is FIR type has been selected
        numerator{1} = fir1(firOrder , frequencies(2)/(Fs/2));
        for i = 1 : 9
            numerator{i} = fir1(firOrder, [frequencies(i) frequencies(i+1)]/(Fs/2));
        end
    end
end
```

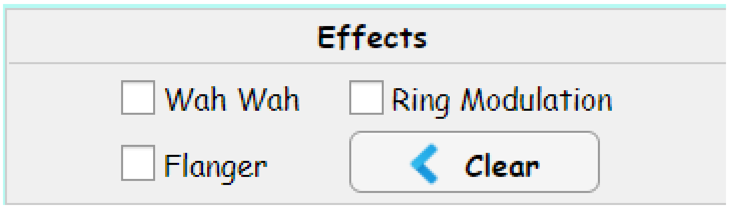
Figure 9

As shown in **figure 1** a multi-band graphic equaliser has been designed and the user is able to change the gain of each band frequency. As can be seen in **figure 9** the butter worth filter has been used which applies a low pass filter to all the bands if the user selects the IIR Filter Type. If the user selects FIR for the filter type, then a window-based FIR filter is applied to all the bands. In addition, the user is able to choose from preconfigured pre-sets for filtering the audio as shown in **Figure 1** and the user is also able to manually adjust the sliders for each band.

The user is able to adjust the volume of the audio play back using the volume control knob. The volume can only be adjusted between 0 and 100.

Student Name: Yash Shah
Student Number: 2078614

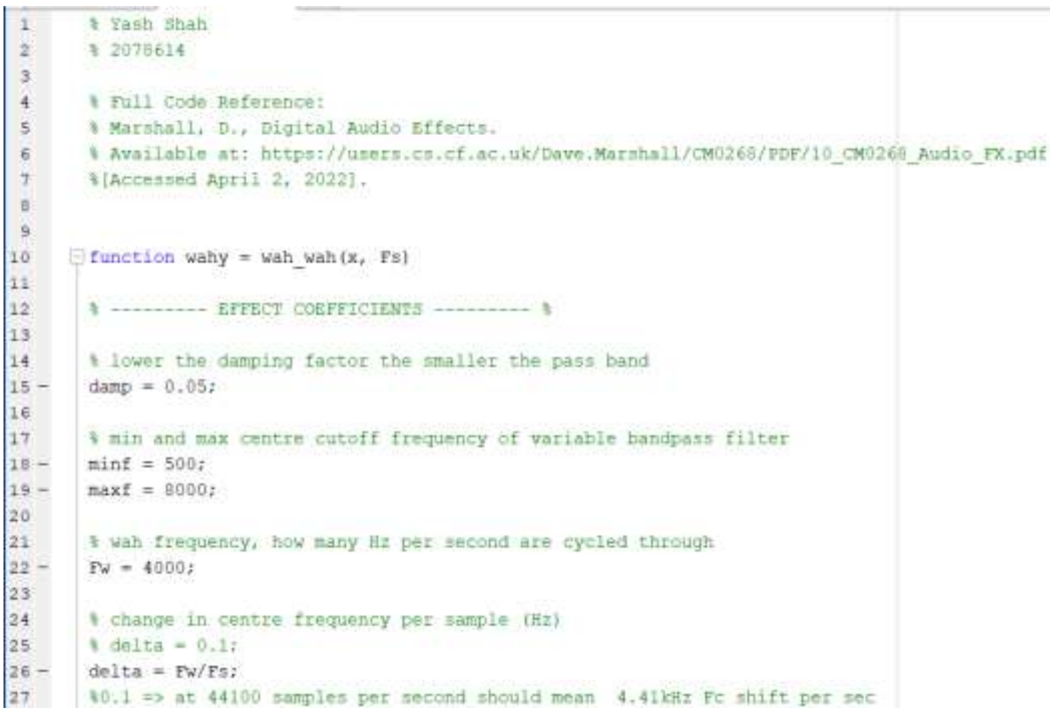
Novel Extension



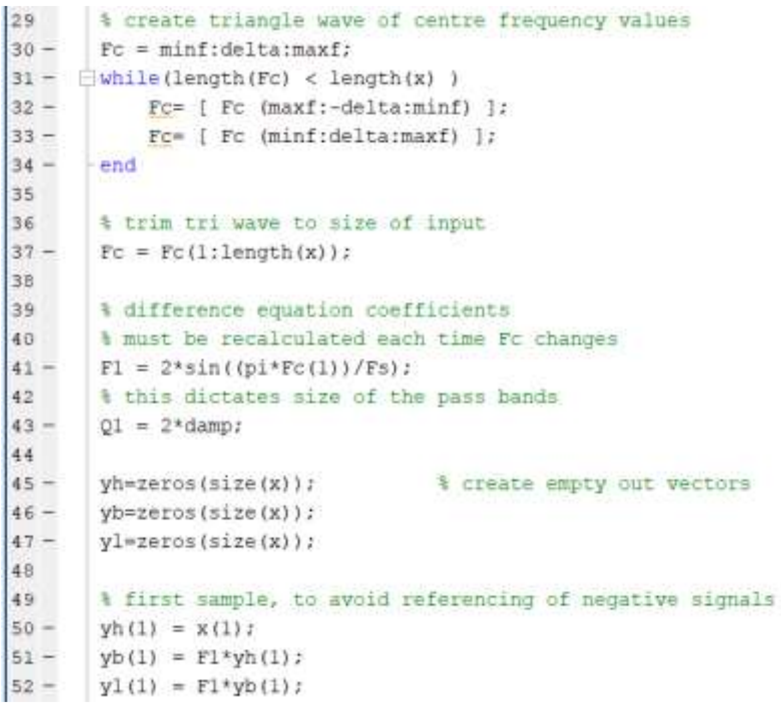
For the novel features different effects have been implemented and the user can simply apply an effect by using the check boxes.

Code used for these different effects are found on Dave Marshall's Digital Audio Effects slides.

Wah Wah Effect

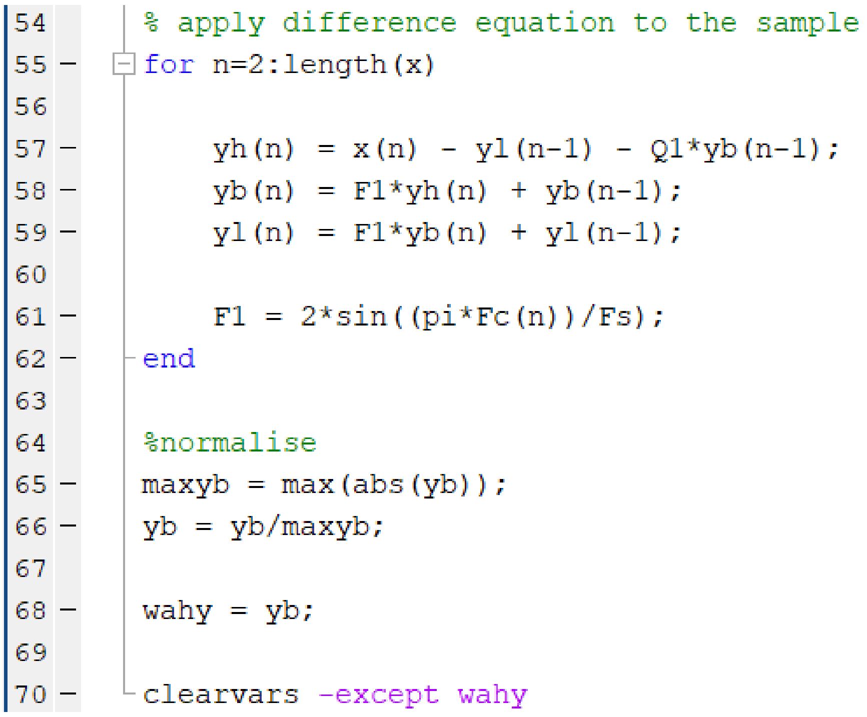


```
% Yash Shah
% 2078614

% Full Code Reference:
% Marshall, D., Digital Audio Effects.
% Available at: https://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf
%[Accessed April 2, 2022].


function wahy = wah_wah(x, Fs)

% --------- EFFECT COEFFICIENTS --------- %

% lower the damping factor the smaller the pass band
damp = 0.05;

% min and max centre cutoff frequency of variable bandpass filter
minf = 500;
maxf = 8000;

% wah frequency, how many Hz per second are cycled through
Fw = 4000;

% change in centre frequency per sample (Hz)
% delta = 0.1;
delta = Fw/Fs;
%0.1 => at 44100 samples per second should mean  4.41kHz Fc shift per sec
```



```
% create triangle wave of centre frequency values
Fc = minf:delta:maxf;
while(length(Fc) < length(x) )
    Fc= [ Fc (maxf:-delta:minf) ];
    Fc= [ Fc (minf:delta:maxf) ];
end

% trim tri wave to size of input
Fc = Fc(1:length(x));

% difference equation coefficients
% must be recalculated each time Fc changes
F1 = 2*sin((pi*Fc(1))/Fs);
% this dictates size of the pass bands
Q1 = 2*damp;

yh=zeros(size(x));          % create empty out vectors
yb=zeros(size(x));
yl=zeros(size(x));

% first sample, to avoid referencing of negative signals
yh(1) = x(1);
yb(1) = F1*yh(1);
yl(1) = F1*yb(1);
```

Student Name: Yash Shah
Student Number: 2078614



```
% apply difference equation to the sample
for n=2:length(x)

    yh(n) = x(n) - yl(n-1) - Q1*yb(n-1);
    yb(n) = F1*yh(n) + yb(n-1);
    yl(n) = F1*yb(n) + yl(n-1);

    F1 = 2*sin((pi*Fc(n))/Fs);
end

%normalise
maxyb = max(abs(yb));
yb = yb/maxyb;

wahy = yb;

clearvars -except wahy
```

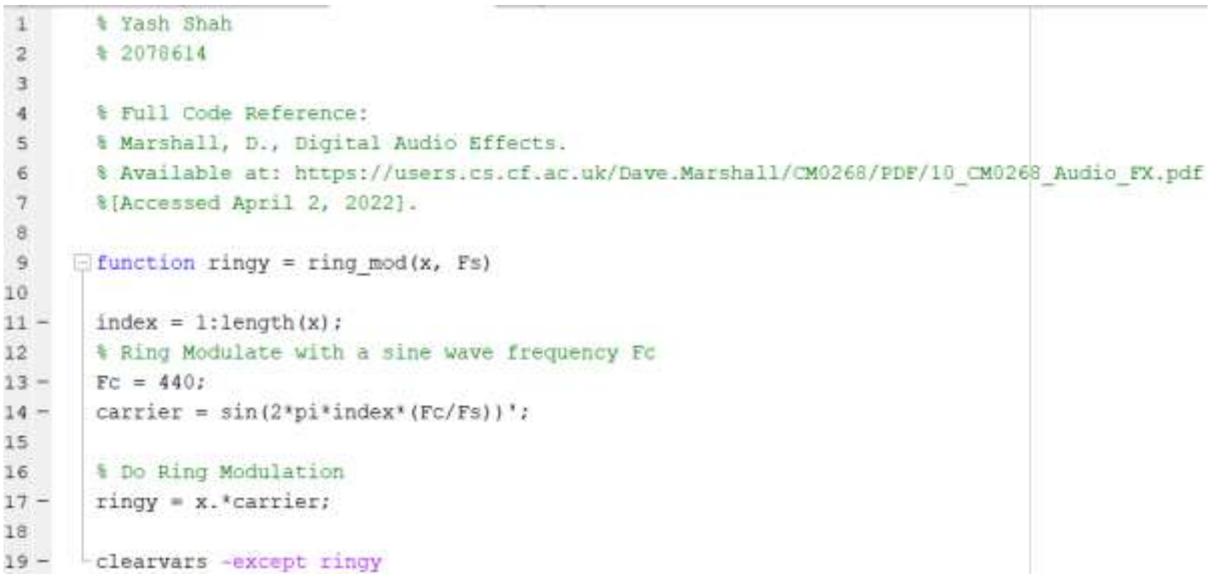
The code above performs three main functions:

1. Creates a triangle wave to modulate the centre frequency of the bandpass filter

2. Implements a state variable filter.

3. Repeated recalculation if centre frequency within the state variable filter loop.

Ring Modulation

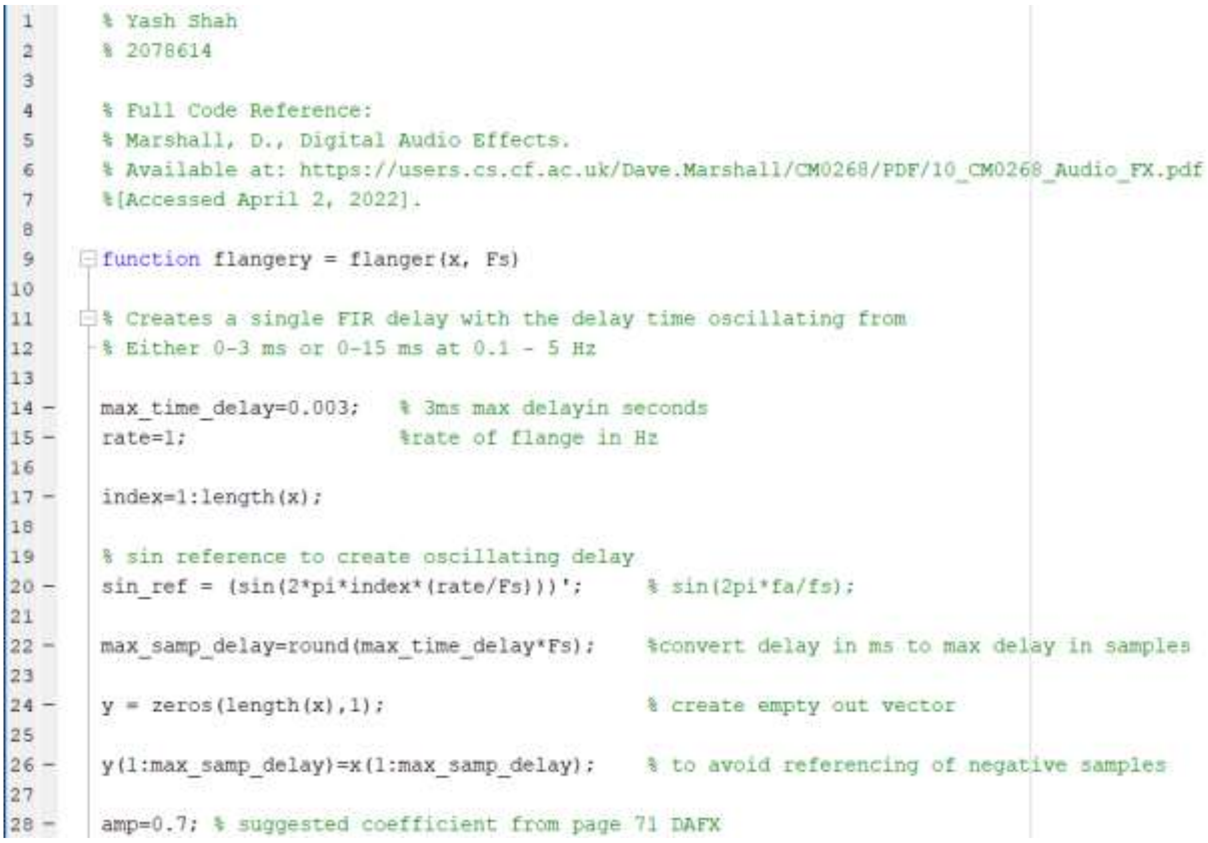


```
% Yash Shah
% 2078614

% Full Code Reference:
% Marshall, D., Digital Audio Effects.
% Available at: https://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf
%[Accessed April 2, 2022].

function ringy = ring_mod(x, Fs)

index = 1:length(x);
% Ring Modulate with a sine wave frequency Fc
Fc = 440;
carrier = sin(2*pi*index*(Fc/Fs))';

% Do Ring Modulation
ringy = x.*carrier;

clearvars -except ringy
```

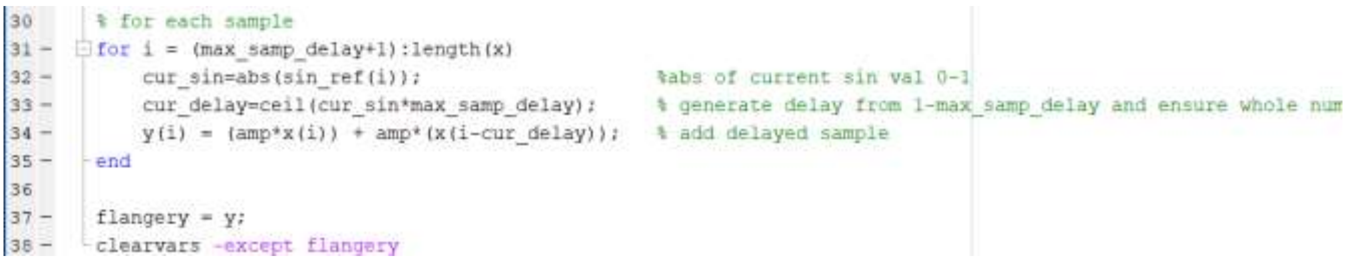
The code above multiplies a sinewave with a carrier frequency FC with an audio modulator signal which produces an output wave. The sinewave is usually the carrier signal and the other signal which is the audio file is the input of the modulator signal.

Student Name: Yash Shah
Student Number: 2078614

Flanger



```
% Yash Shah
% 2078614

% Full Code Reference:
% Marshall, D., Digital Audio Effects.
% Available at: https://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf
%[Accessed April 2, 2022].

function flangery = flanger(x, Fs)

% Creates a single FIR delay with the delay time oscillating from
% Either 0-3 ms or 0-15 ms at 0.1 - 5 Hz

max_time_delay=0.003;   % 3ms max delayin seconds
rate=1;                 %rate of flange in Hz

index=1:length(x);

% sin reference to create oscillating delay
sin_ref = (sin(2*pi*index*(rate/Fs)))';      % sin(2pi*fa/fs);

max_samp_delay=round(max_time_delay*Fs);     %convert delay in ms to max delay in samples

y = zeros(length(x),1);                      % create empty out vector

y(1:max_samp_delay)=x(1:max_samp_delay);     % to avoid referencing of negative samples

amp=0.7; % suggested coefficient from page 71 DAFX
```



```
% for each sample
for i = (max_samp_delay+1):length(x)
    cur_sin=abs(sin_ref(i));                    %abs of current sin val 0-1
    cur_delay=ceil(cur_sin*max_samp_delay);     % generate delay from 1-max_samp_delay and ensure whole num
    y(i) = (amp*x(i)) + amp*(x(i-cur_delay));   % add delayed sample
end

flangery = y;
clearvars -except flangery
```

The code above works by mixing the same audio signal twice where once signal is played at a slightly slower speed. This creates the effect of two tape recordings playing simultaneously, but one going slightly slower than the other.

Reference

% Marshall, D., Digital Audio Effects.
% Available at:
https://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf
% [Accessed April 2, 2022].