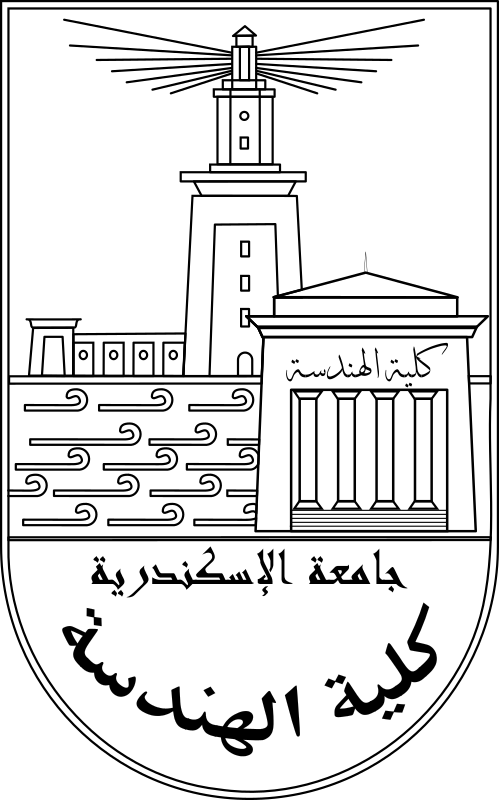
Audio equalizer

**Peter Anton Naguib - 7406**

**Fadi Sarwat Farouk - 7432**



# Source code:

## Function to generate the numerator and denominator coefficients of a low-pass filter (FIR or IIR) based on the specified parameters.

% Function to design a low-pass filter

% Inputs:

% samplerate: Sample rate of the signal

% cutoffFreq: Cut-off frequency of the filter

% order: Order of the filter

% filter: Type of filter (FIR or IIR)

% Outputs:

% a: Coefficients of the filter's denominator polynomial

% b: Coefficients of the filter's numerator polynomial

function [a, b] = low\_pass\_filter(samplerate, cutoffFreq, order, filter)

% Calculate the Nyquist frequency

fm = samplerate / 2;

% Normalize the cut-off frequency

wn = cutoffFreq / fm;

if (filter == "FIR")

% Design a finite impulse response (FIR) filter using the FIR1 function

[a, b] = fir1(order, wn, 'low');

% Ensure equal length of the coefficient vectors a and b

[a, b] = eqtflength(a, b);

elseif filter == "IIR"

% Design an infinite impulse response (IIR) filter using the BUTTER function

[a, b] = butter(order, wn);

% Ensure equal length of the coefficient vectors a and b

[a, b] = eqtflength(a, b);

end

end

## Function to generate the numerator and denominator coefficients of a band-pass filter (FIR or IIR) based on the specified parameters.

function [a, b] = band\_pass\_filter(samplerate, cutoffFreq1, cutoffFreq2, order, filter)

% Calculate the Nyquist frequency

fm = samplerate / 2;

% Normalize the cut-off frequencies

wn = [cutoffFreq1 cutoffFreq2] / fm;

if (filter == "FIR")

% Design a finite impulse response (FIR) filter using the FIR1 function

[a, b] = fir1(order, wn, 'bandpass');

% Ensure equal length of the coefficient vectors a and b

[a, b] = eqtflength(a, b);

else

% Design an infinite impulse response (IIR) filter using the BUTTER function

[a, b] = butter(order, wn, 'bandpass');

% Ensure equal length of the coefficient vectors a and b

[a, b] = eqtflength(a, b);

end

end

## Function to apply a band-pass filter to an input signal using the numerator and denominator coefficients.

function result = apply\_filter(filter\_type, gains, audio\_data, sample\_rate)

% Define cutoff frequency ranges for each band

freq1 = [0, 170, 300, 610, 1005, 3000, 6000, 12000, 14000];

freq2 = [170, 300, 610, 1005, 3000, 6000, 12000, 14000, 20000];

filter\_order=0;

% Set filter order based on filter type

if filter\_type == "FIR"

filter\_order = 600;

elseif filter\_type== "IIR"

filter\_order = 4;

end

% Apply low-pass filter to the audio data for the first band

[a, b] = low\_pass\_filter(sample\_rate, 170, filter\_order, filter\_type);

filtered\_data = filter(a, b, audio\_data);

result = filtered\_data \* gains(1);

% Apply band-pass filters to the audio data for the remaining bands

for i = 2:9

[a, b] = band\_pass\_filter(sample\_rate, freq1(i), freq2(i), filter\_order, filter\_type);

filtered\_data = filter(a, b, audio\_data);

result = result + filtered\_data \* gains(i);

end

end

## Function to analyze the input and output of a filter. It analyzes the impulse response, step response, magnitude and phase, and poles and zeros of the filter.

function analyze\_signal = analyze\_signal(samplerate, filter\_type, audidata)

freq1 = [0 170 300 610 1005 3000 6000 12000 14000];

freq2 = [170 300 610 1005 3000 6000 12000 14000 20000];

num\_samples = length(audidata);

fs = samplerate;

t = linspace(0, num\_samples/samplerate, num\_samples);

f = linspace(-fs/2, fs/2, num\_samples);

if (filter\_type == "FIR")

order = 600;

else

order = 8;

end

[a, b] = low\_pass\_filter(samplerate, 170, order, filter\_type);

%get transfer function based on the ni=um and denum of low pass filter

TF = tf(a, b);

% Filter analysis

figure();

subplot(2, 1, 1);

freqz(a, b, num\_samples);

title('Magnitude and phase of input low pass filter frequency (0-170)');

subplot(2, 1, 2);

pzmap(tf);

title('Poles and zeros of low pass filtered signal (0-170)');

figure();

subplot(2, 1, 1);

impulse(TF);

title('Impulse response of low pass filtered signal (0-170)');

subplot(2, 1, 2);

step(TF);

title('Step response of low pass filtered signal (0-170)');

% Output analysis

y = filter(a, b, audidata);

Y = fftshift(fft(y));

figure();

subplot(3, 1, 1);

plot(t, y);

title('Time domain of the output signal (0-170)');

subplot(3, 1, 2);

plot(f, abs(Y));

title('Magnitude of the output signal (0-170)');

subplot(3, 1, 3);

plot(f, angle(Y));

title('Phase of the output signal (0-170)');

for i = 2:9

[a, b] = band\_pass\_filter(samplerate, freq1(i), freq2(i), order, filter\_type);

TF = tf(a, b);

% Filter analysis

figure();

subplot(2, 1, 1);

freqz(a, b, num\_samples);

range = strcat(' (', int2str(freq1(i)), 'Hz - ', int2str(freq2(i)), 'Hz)');

title(strcat('Magnitude & Phase of H', int2str(i), '(Z)', range));

subplot(2, 1, 2);

pzmap(tf);

title(strcat('Poles and zeros', int2str(i), '(Z)', range));

figure();

subplot(2, 1, 1);

impulse(TF);

title(strcat('Impulse response', int2str(i), '(Z)', range));

subplot(2, 1, 2);

step(TF);

title(strcat('Step response', int2str(i), '(Z)', range));

% Output analysis

y = filter(a, b, audidata);

Y = fftshift(fft(y));

figure();

subplot(3, 1, 1);

plot(t, y);

title(strcat('Time domain of the output signal', int2str(i), '(Z)', range));

subplot(3, 1, 2);

plot(f, abs(Y));

title(strcat('Magnitude of the output signal', int2str(i), '(Z)', range))

subplot(3, 1, 3);

plot(f, angle(Y));

title(strcat('Phase of the output signal', int2str(i), '(Z)', range));

end

end

## GUI code

properties (Access = public)

filename;

pathname;

fs=0;

fs\_original=0;

audio;

audio\_old;

gainsArr = ones(1, 9);

end

methods (Access = private)

function apply(app)

app.audio = apply\_filter(app.FilterDropDown.Value, app.gainsArr, app.audio\_old, app.fs\_original);

app.draw();

end

function draw(app)

plot(app.originalPlot, app.audio\_old);

plot(app.enhancedPlot, app.audio);

end

end

% Callbacks that handle component events

methods (Access = private)

% Button pushed function: browseButton

function browseButtonPushed(app, event)

[file, Pathname] = uigetfile('\*.wav','wav file selector');

app.filename = file;

app.pathname = strcat(file, Pathname);

[app.audio, app.fs] = audioread(file);

app.audio\_old = app.audio;

app.fs\_original=app.fs;

app.fileaddressEditField.Value = app.filename;

app.samplerateEditField.Value = app.fs;

app.draw();

end

% Button pushed function: play\_Button

function play\_ButtonPushed(app, event)

clear sound

sound(app.audio, app.fs);

end

% Button pushed function: stop\_Button

function stop\_ButtonPushed(app, event)

clear sound;

end

% Value changing function: HzSlider\_2

function HzSlider\_2ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(2) = value;

app.slider\_disp\_2.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider

function HzSliderValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(1) = value;

app.slider\_disp\_1.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_3

function HzSlider\_3ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(3) = value;

app.slider\_disp\_3.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_4

function HzSlider\_4ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(4) = value;

app.slider\_disp\_4.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_5

function HzSlider\_5ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(5) = value;

app.slider\_disp\_5.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_6

function HzSlider\_6ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(6) = value;

app.slider\_disp\_6.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_7

function HzSlider\_7ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(7) = value;

app.slider\_disp\_7.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_8

function HzSlider\_8ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(8) = value;

app.slider\_disp\_8.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Value changing function: HzSlider\_9

function HzSlider\_9ValueChanged(app, event)

value = 10^(event.Value / 20);

app.gainsArr(9) = value;

app.slider\_disp\_9.Text = mat2str(value, 2);

app.ApplyFiltersButton.Enable = 1;

end

% Button pushed function: ApplyFiltersButton

function ApplyFiltersButtonPushed(app, event)

app.ApplyFiltersButton.Enable = 0;

app.apply(); % slow function

clear sound;

sound(app.audio, app.fs);

end

% Button pushed function: SaveButton

function SaveButtonPushed(app, event)

audiowrite('enhanced.wav', app.audio, app.fs);

end

% Value changed function: samplerateEditField

function samplerateEditFieldValueChanged2(app, event)

value = app.samplerateEditField.Value;

if value == app.fs

app.audio = app.audio\_old;

elseif value > app.fs

app.audio = upsample(app.audio\_old, value/app.fs);

app.fs=value;

app.apply

elseif value < app.fs

downsamplingFactor = floor(app.fs/value); % Round down to nearest integer

app.audio = downsample(app.audio\_old, downsamplingFactor);

app.fs=value;

app.apply

end

app.ApplyFiltersButton.Enable = 1;

end

% Button pushed function: AnalyzeButton

function AnalyzeButtonPushed(app, event)

analyze\_signal(app.fs, app.FilterDropDown.Value, app.audio);

end

% Value changed function: FilterDropDown

function FilterDropDownValueChanged(app, event)

app.ApplyFiltersButton.Enable = 1;

end

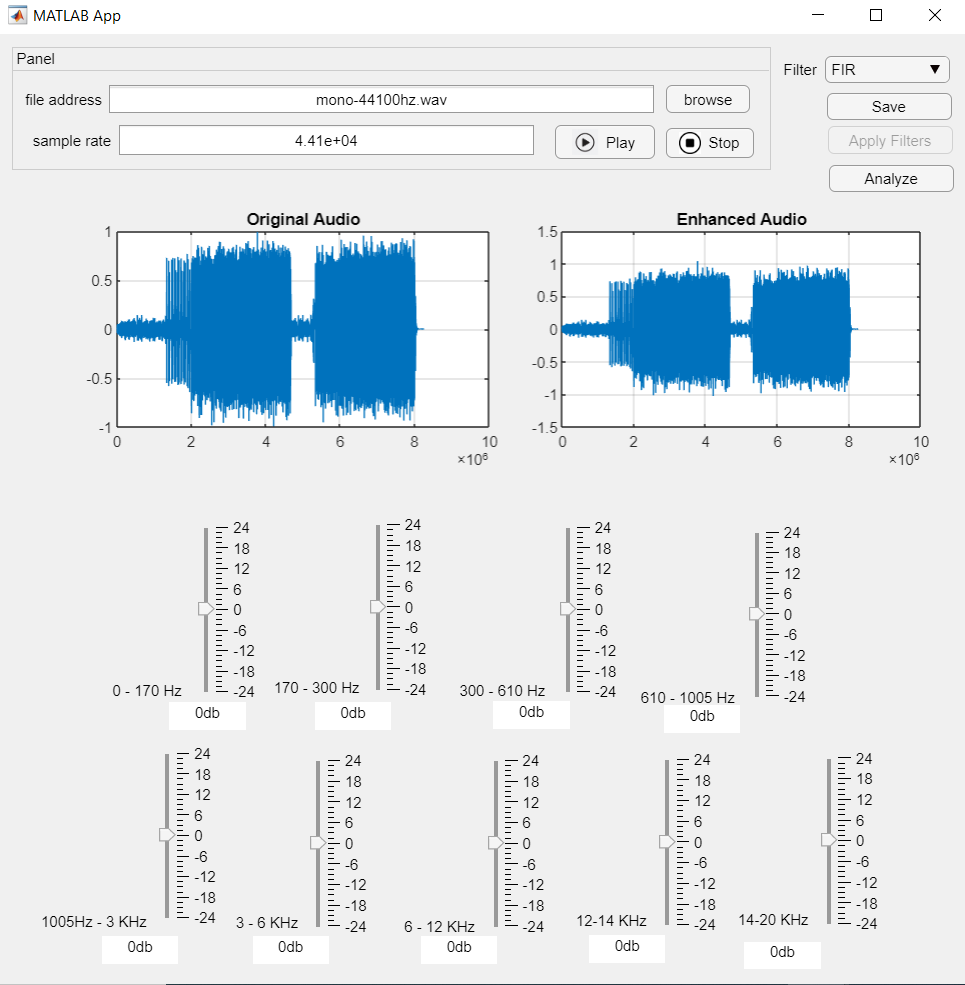
end

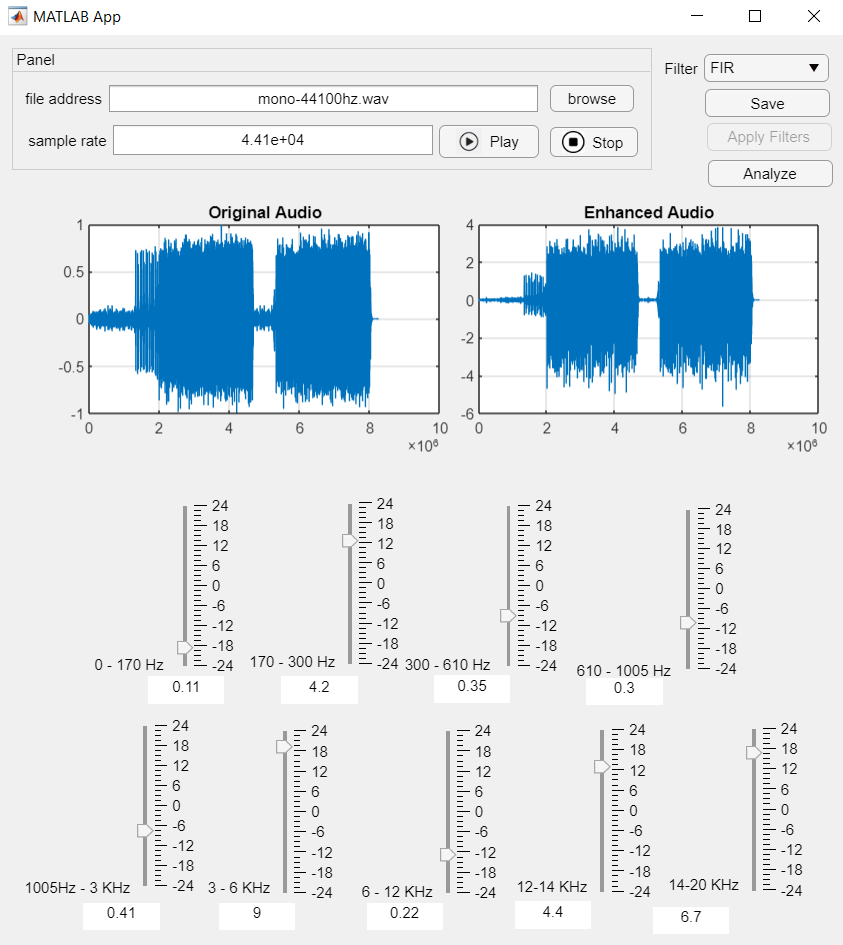
A screenshot of a computer

Description automatically generated with medium confidence

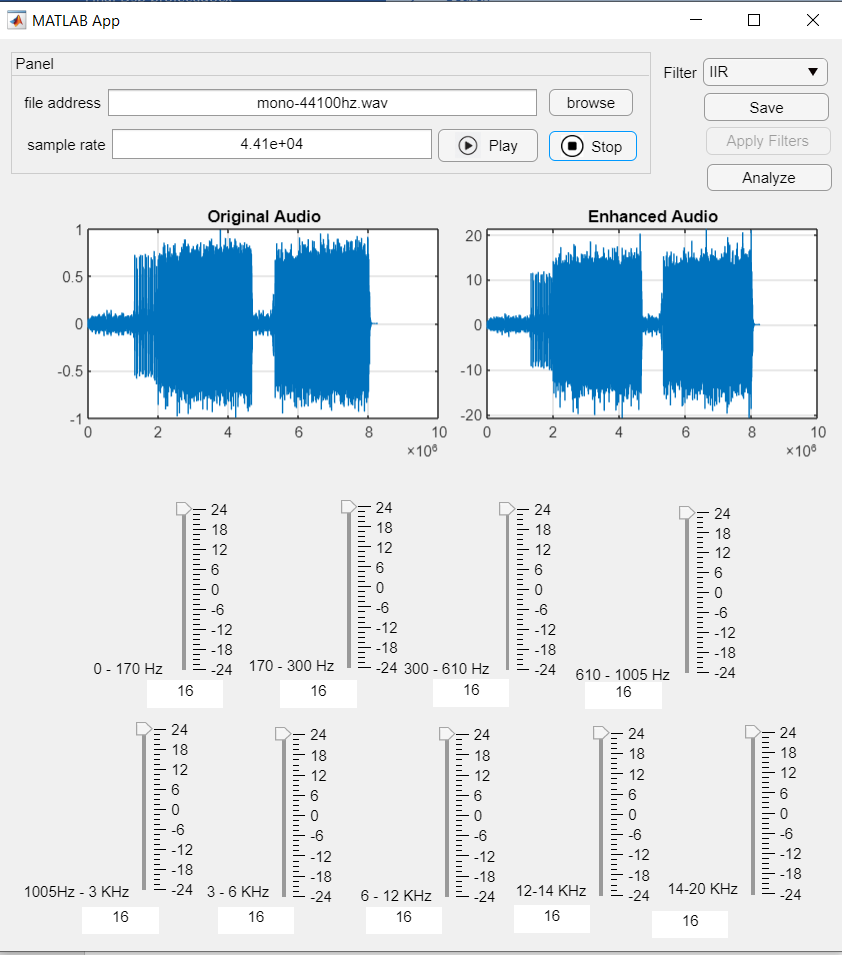
## Sample Runs

1. A screenshot of a computer

   Description automatically generatedUsing fir filter-sample rate 44100

****A screenshot of a computer

Description automatically generated

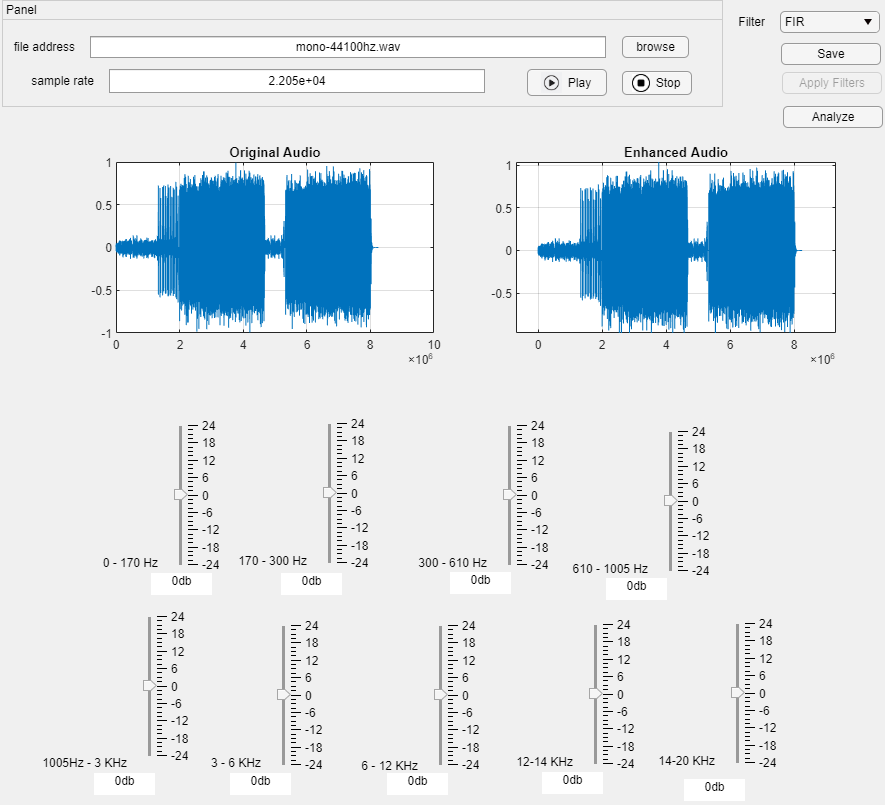
1. **A screenshot of a computer

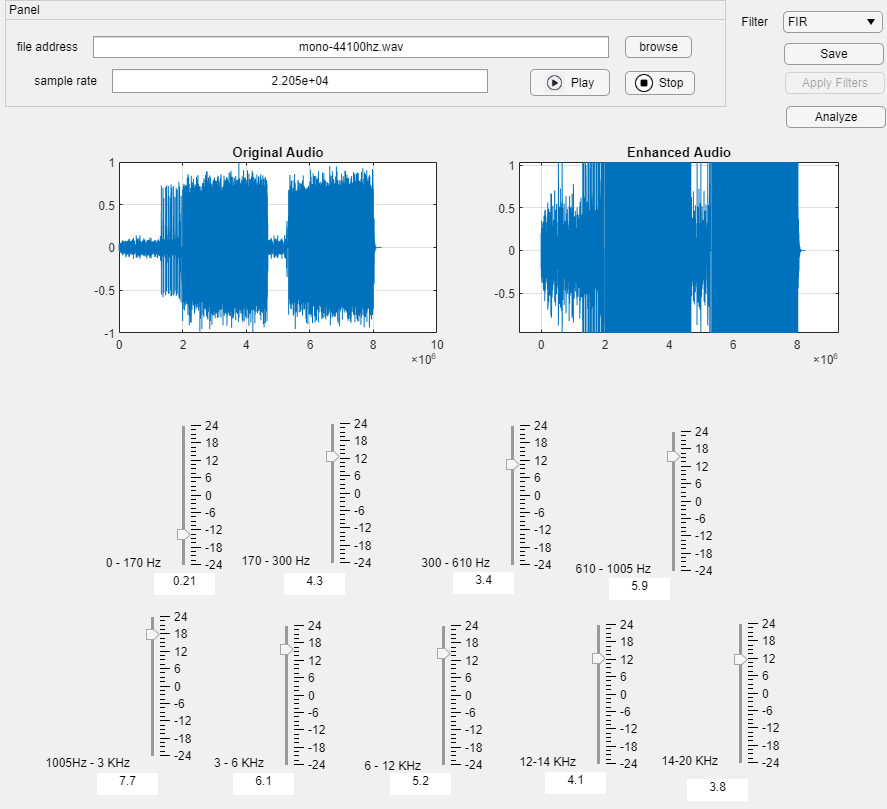
   Description automatically generatedA screenshot of a computer

   Description automatically generatedA screenshot of a computer

   Description automatically generated**Different gains using IIR with sample rate 44100.

## A screenshot of a computer Description automatically generatedDown Sample from 44100 to 22050

A screenshot of a computer

Description automatically generated

## A screenshot of a computer Description automatically generated with medium confidenceA screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generated with medium confidenceUp Sample from 44100 to 88200