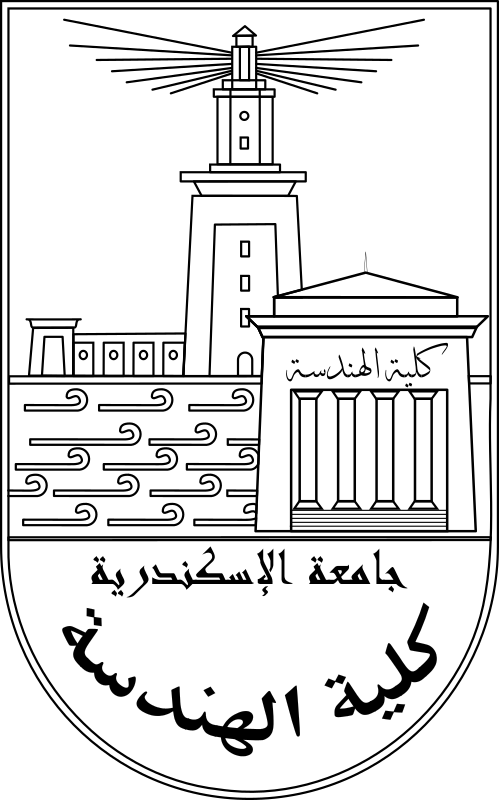
Audio equalizer

**Zeyad ahmed Fathallah-7621**

**Yamen Mohamed saad-7577**



1. ***Initialize the variables:***

properties (Access = public)

% input sound signal arguments

%(global variables)

inputSignal;

inFs;

% filters

fb1; fb2; fb3; fb4; fb5; fb6; fb7; fb8; fb9; fb10;

fa1; fa2; fa3; fa4; fa5; fa6; fa7; fa8; fa9; fa10;

% output sound signal arguments

outputSignal;

band0; band170; band310; band600; band1k; band3k; band6k; band12k; band14k; band16k;

% gui user inputs

gain1 = 0;

gain2 = 0;

gain3 = 0;

gain4 = 0;

gain5 = 0;

gain6 = 0;

gain7 = 0;

gain8 = 0;

gain9 = 0;

gain10 = 0;

ftype = 'IIR';

outFs;

% sound player

player;

soundFs;

end

methods (Access = private)

1. ***This that perform the filters:***

function performFiltration(app)

initializeFilters(app);

app.band0 = filter(app.fb1, app.fa1, app.inputSignal) \* db2mag(app.gain1);

app.band170 = filter(app.fb2, app.fa2, app.inputSignal) \* db2mag(app.gain2);

app.band310 = filter(app.fb3, app.fa3, app.inputSignal) \* db2mag(app.gain3);

app.band600 = filter(app.fb4, app.fa4, app.inputSignal) \* db2mag(app.gain4);

app.band1k = filter(app.fb5, app.fa5, app.inputSignal) \* db2mag(app.gain5);

app.band3k = filter(app.fb6, app.fa6, app.inputSignal) \* db2mag(app.gain6);

app.band6k = filter(app.fb7, app.fa7, app.inputSignal) \* db2mag(app.gain7);

app.band12k = filter(app.fb8, app.fa8, app.inputSignal) \* db2mag(app.gain8);

app.band14k = filter(app.fb9, app.fa9, app.inputSignal) \* db2mag(app.gain9);

app.band16k = filter(app.fb10, app.fa10, app.inputSignal) \* db2mag(app.gain10);

% app.band0 = filter(app.fb1, app.fa1, app.inputSignal) \* (10^(app.gain1/20));

% app.band170 = filter(app.fb2, app.fa2, app.inputSignal) \* (10^(app.gain2/20));

% app.band310 = filter(app.fb3, app.fa3, app.inputSignal) \* (10^(app.gain3/20));

% app.band600 = filter(app.fb4, app.fa4, app.inputSignal) \* (10^(app.gain4/20));

% app.band1k = filter(app.fb5, app.fa5, app.inputSignal) \* (10^(app.gain5/20));

% app.band3k = filter(app.fb6, app.fa6, app.inputSignal) \* (10^(app.gain6/20));

% app.band6k = filter(app.fb7, app.fa7, app.inputSignal) \* (10^(app.gain7/20));

% app.band12k = filter(app.fb8, app.fa8, app.inputSignal) \* (10^(app.gain8/20));

% app.band14k = filter(app.fb9, app.fa9, app.inputSignal) \* (10^(app.gain9/20));

% app.band16k = filter(app.fb10, app.fa10, app.inputSignal) \* (10^(app.gain10/20));

app.outputSignal = app.band0 + app.band170 + app.band310 + app.band600 + app.band1k + app.band3k + app.band6k + app.band12k + app.band14k + app.band16k;

end

1. ***Filters Initialization functions(fir and iir):***

function initializeFilters(app)

if(app.ftype == 'IIR')

IIRinitialization(app);

else

FIRinitialization(app);

end

end

function IIRinitialization(app)

[app.fb1, app.fa1] = getIirFilter(app, 1);

[app.fb2, app.fa2] = getIirFilter(app, 2);

[app.fb3, app.fa3] = getIirFilter(app, 3);

[app.fb4, app.fa4] = getIirFilter(app, 4);

[app.fb5, app.fa5] = getIirFilter(app, 5);

[app.fb6, app.fa6] = getIirFilter(app, 6);

[app.fb7, app.fa7] = getIirFilter(app, 7);

[app.fb8, app.fa8] = getIirFilter(app, 8);

[app.fb9, app.fa9] = getIirFilter(app, 9);

[app.fb10, app.fa10] = getIirFilter(app, 10);

end

function FIRinitialization(app)

[app.fb1, app.fa1] = getFirFilter(app, 1);

[app.fb2, app.fa2] = getFirFilter(app, 2);

[app.fb3, app.fa3] = getFirFilter(app, 3);

[app.fb4, app.fa4] = getFirFilter(app, 4);

[app.fb5, app.fa5] = getFirFilter(app, 5);

[app.fb6, app.fa6] = getFirFilter(app, 6);

[app.fb7, app.fa7] = getFirFilter(app, 7);

[app.fb8, app.fa8] = getFirFilter(app, 8);

[app.fb9, app.fa9] = getFirFilter(app, 9);

[app.fb10, app.fa10] = getFirFilter(app, 10);

end

function [b, a] = getIirFilter(app, index)

N = 4;

fs = app.outFs;

switch(index)

case 1

[b, a] = iLPF(app, N, 170, fs);

case 2

[b, a] = iBPF(app, N, 170, 310, fs);

case 3

[b, a] = iBPF(app, N, 310, 600, fs);

case 4

[b, a] = iBPF(app, N, 600, 1000, fs);

case 5

[b, a] = iBPF(app, N, 1000, 3000, fs);

case 6

[b, a] = iBPF(app, N, 3000, 6000, fs);

case 7

[b, a] = iBPF(app, N, 6000, 12000, fs);

case 8

[b, a] = iBPF(app, N, 12000, 14000, fs);

case 9

[b, a] = iBPF(app, N, 14000, 16000, fs);

case 10

[b, a] = iHPF(app, N, 16000, fs);

end

end

function [b, a] = getFirFilter(app, index)

N = 35;

fs = app.outFs;

switch(index)

case 1

[b, a] = fLPF(app, N, 170, fs);

case 2

[b, a] = fBPF(app, N, 170, 310, fs);

case 3

[b, a] = fBPF(app, N, 310, 600, fs);

case 4

[b, a] = fBPF(app, N, 600, 1000, fs);

case 5

[b, a] = fBPF(app, N, 1000, 3000, fs);

case 6

[b, a] = fBPF(app, N, 3000, 6000, fs);

case 7

[b, a] = fBPF(app, N, 6000, 12000, fs);

case 8

[b, a] = fBPF(app, N, 12000, 14000, fs);

case 9

[b, a] = fBPF(app, N, 14000, 16000, fs);

case 10

[b, a] = fHPF(app, N, 16000, fs);

end

end

% IIR Filters creation functions %

function [b, a] = iLPF(~, N, fc, fs) % order, cutoff freq, sampling freq

wn = (fc\*2) / fs; % wc = 2\*pi\*fc/fs => wn = wc/pi

[b, a] = butter(N, wn); % result = [b, a] => num & denom of TF

end

function [b, a] = iBPF(~, N, f1, f2, fs) % orded, band start freq, band end freq, sampling freq

wn1 = (f1\*2) / fs;

wn2 = (f2\*2) / fs;

wn = [wn1 wn2];

[b, a] = butter(N, wn);% for iir filter

end

function [b, a] = iHPF(~, N, fc, fs) % order, cutoff freq, sampling freq

wn = (fc\*2) / fs; % wc = 2\*pi\*fc/fs => wn = wc/pi

[b, a] = butter(N, wn, 'high'); % result = [b, a] => num & denom of TF

End

function [b, a] = fLPF(~, N, fc, fs) % order, cutoff freq, gain

wn = (fc\*2) / fs; % wc = 2\*pi\*fc/fs => wn = wc/pi

[b, a] = fir1(N, wn); % result = [b, a] => num & denom of TF

end

function [b, a] = fBPF(~, N, f1, f2, fs) % orded, band start freq, band end freq, gain

wn1 = (f1\*2) / fs;

wn2 = (f2\*2) / fs;

wn = [wn1 wn2];

[b, a] = fir1(N, wn);

end

function [b, a] = fHPF(~, N, fc, fs) % order, cutoff freq, gain

wn = (fc\*2) / fs; % wc = 2\*pi\*fc/fs => wn = wc/pi

[b, a] = fir1(N, wn, 'high'); % result = [b, a] => num & denom of TF

end

1. ***Functions for analysis and plotting:***

function plotGainPhase(~, b, a)

freqz(b, a);% freq res z domain

grid on;

end

function plotImpulseResponce(~, b, a)

impz(b, a);%impulse z domain

grid on;

end

function plotStepResponce(~, b, a)

stepz(b, a);% step res z domain

grid on;

end

function plotPZmap(~, b, a)

transferFn = tf(b, a);

pzmap(transferFn);

grid on;

end

function plotTimeDomain(~, axes, ft, fs)

tstart = 0;

% length(ft) = (tend - tstart) \* fs

tend = (length(ft) / fs) + tstart;

t = linspace(tstart, tend, length(ft));

plot(axes, t, ft);

grid on;

end

function plotFreqDomain(~, axes, fn, fs)

% frequency base vector

freq = linspace(-fs/2, fs/2, length(fn));

FN = fft(fn); %get foureir transform

FN = fftshift(FN);

plot(axes, freq, FN);

end

end

methods (Access = private)

5)***switch for choosing fir our iir:***

function FilterTypeSwitchValueChanged(app, event)

value = app.FilterTypeSwitch.Value;

if(value == 'IIR')

app.IIR\_lamp.Color = [0.00,1.00,0.00]; %green

app.FIR\_lamp.Color = [0.50,0.50,0.50]; %grey

app.ftype = 'IIR';

else

app.FIR\_lamp.Color = [0.00,1.00,0.00]; % green

app.IIR\_lamp.Color = [0.50,0.50,0.50]; %grey

app.ftype = 'FIR';

end

end

6)***audio file chooser function:***

function chooseBtnButtonPushed(app, event)

filter = {'.wav'};

[name, path] = uigetfile(filter);

directory = [path name];

app.AudioEquilizerUIFigure.Visible = false;

app.AudioEquilizerUIFigure.Visible = true;

if(name ~= 0)

[input, fs] = audioread(directory);

app.inputSignal = input;

app.outputSignal = input;

app.inFs = fs;

if(fs < 32010)

app.outFs = 32010;

else

app.outFs = fs;

end

app.soundFs = app.inFs;

app.OutputFskHzSpinner.Value = app.inFs/1000;

app.playBtn.Enable = true;

app.applyBtn.Enable = true;

app.saveBtn.Enable = true;

app.stopBtn.Enable = true;

app.playOriginalBtn.Enable = true;

app.resetBtn.Enable = true;

end

end

7) ***play button function:***

function playBtnButtonPushed(app, event)

app.player = audioplayer(app.outputSignal, app.soundFs);

stop(app.player);

play(app.player);

%clear playsnd;

%sound(app.outputSignal, app.outFs);

end

8)***sample rate input function:***

function OutputFskHzSpinnerValueChanged(app, event)

value = app.OutputFskHzSpinner.Value;

app.soundFs = value \* 1000;

end

9) ***apply filters buttom function:***

function applyBtnButtonPushed(app, event)

performFiltration(app);

app.inOutRadio.Visible = true;

app.f1Radio.Visible = true;

app.f2Radio.Visible = true;

app.f3Radio.Visible = true;

app.f4Radio.Visible = true;

app.f5Radio.Visible = true;

app.f6Radio.Visible = true;

app.f7Radio.Visible = true;

app.f8Radio.Visible = true;

app.f9Radio.Visible = true;

app.f10Radio.Visible = true;

end

11)***taking inputs from sliders in gui for filters \*10***

function band1SliderValueChanged(app, event)

value = app.band1Slider.Value;

app.gain1 = value;

end

11)***helper function for ploting:***

function inOutRadioSelectionChanged(app, event)

selectedButton = app.inOutRadio.SelectedObject;

if(selectedButton == app.InputtimedomainButton)

plotTimeDomain(app, app.plotInOut, app.inputSignal, app.inFs);

elseif(selectedButton == app.InputfrequencydomainButton)

plotFreqDomain(app, app.plotInOut, app.inputSignal, app.inFs);

elseif(selectedButton == app.OutputtimedomainButton)

plotTimeDomain(app, app.plotInOut, app.outputSignal, app.soundFs);

elseif(selectedButton == app.OutputfrequencydomainButton)

plotFreqDomain(app, app.plotInOut, app.outputSignal, app.soundFs);

end

end

12) ***function for each tab in gui for plotting\*10:***

function f1RadioSelectionChanged(app, event)

selectedButton = app.f1Radio.SelectedObject;

if(selectedButton == app.plotF1GainPhase)

figure;

plotGainPhase(app, app.fb1, app.fa1);

elseif(selectedButton == app.plotF1Impulse)

figure;

plotImpulseResponce(app, app.fb1, app.fa1);

elseif(selectedButton == app.plotF1Step)

figure;

plotStepResponce(app, app.fb1, app.fa1);

elseif(selectedButton == app.plotF1pz)

figure;

plotPZmap(app, app.fb1, app.fa1);

elseif(selectedButton == app.plotF1outTime)

plotTimeDomain(app, app.f1Plot, app.band0, app.soundFs);

elseif(selectedButton == app.plotF1outFreq)

plotFreqDomain(app, app.f1Plot, app.band0, app.soundFs);

end

end

13)***function for stop button***

function stopBtnButtonPushed(app, event)

stop(app.player);

%clear playsnd;

end

14) ***function for playing original button:***

function playOriginalBtnButtonPushed(app, event)

app.player = audioplayer(app.inputSignal, app.inFs);

stop(app.player);

play(app.player);

end

15) ***function for resting all filters value and sample rate :*** function resetBtnButtonPushed(app, event)

app.outputSignal = app.inputSignal;

app.soundFs = app.inFs;

app.gain1 = 0;

app.gain2 = 0;

app.gain3 = 0;

app.gain4 = 0;

app.gain5 = 0;

app.gain6 = 0;

app.gain7 = 0;

app.gain8 = 0;

app.gain9 = 0;

app.gain10 = 0;

app.band1Slider.Value = 0;

app.band2Slider.Value = 0;

app.band3Slider.Value = 0;

app.band4Slider.Value = 0;

app.band5Slider.Value = 0;

app.band6Slider.Value = 0;

app.band7Slider.Value = 0;

app.band8Slider.Value = 0;

app.band9Slider.Value = 0;

app.band10Slider.Value = 0;

app.OutputFskHzSpinner.Value = app.inFs/1000;

end

16) ***function for save button:***

function saveBtnButtonPushed(app, event)

filter = {'.wav'};

[name, path] = uiputfile(filter);

app.AudioEquilizerUIFigure.Visible = false;

app.AudioEquilizerUIFigure.Visible = true;

if(name ~= 0)

if(length(name) == 4)

name = 'untitled.wav';

end

directory = [path name];

audiowrite(directory, app.outputSignal, app.soundFs);

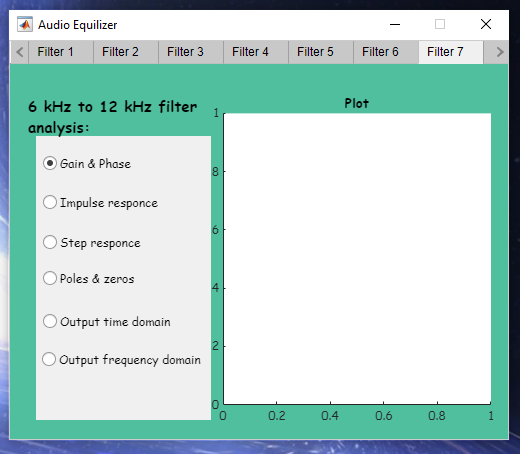
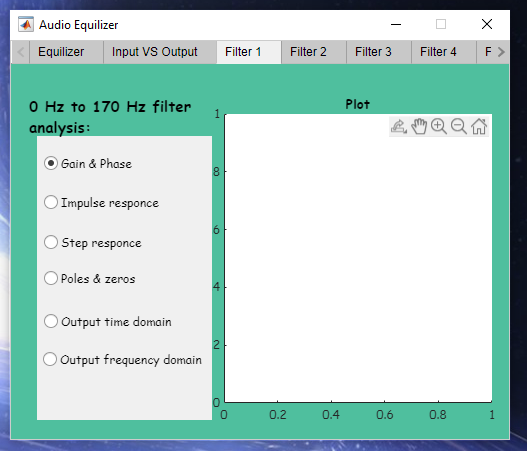
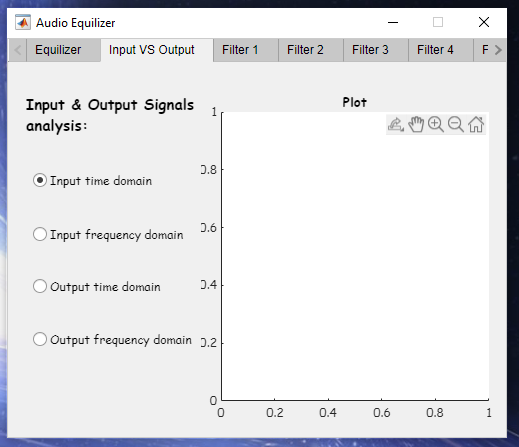
end

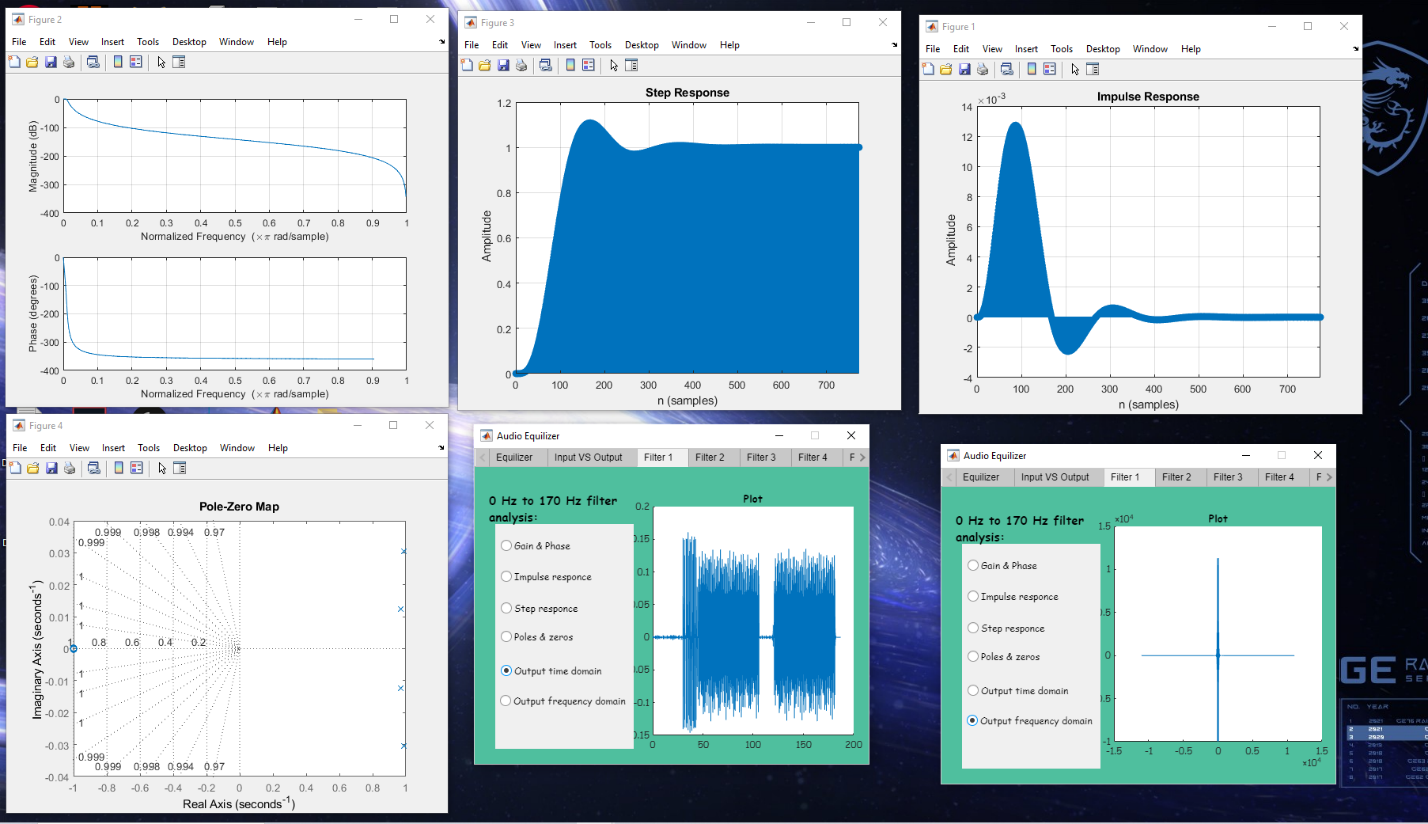
end

end

***gui appearing:***

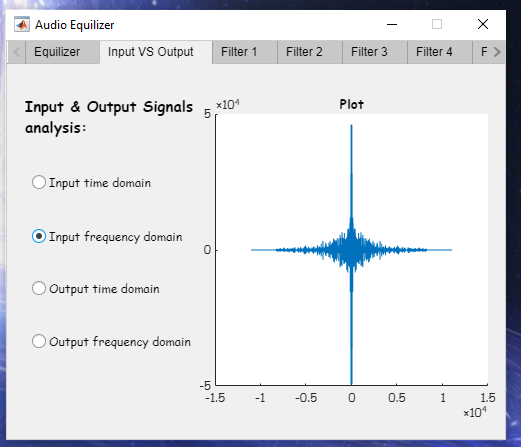
A screenshot of a computer

Description automatically generated with medium confidence

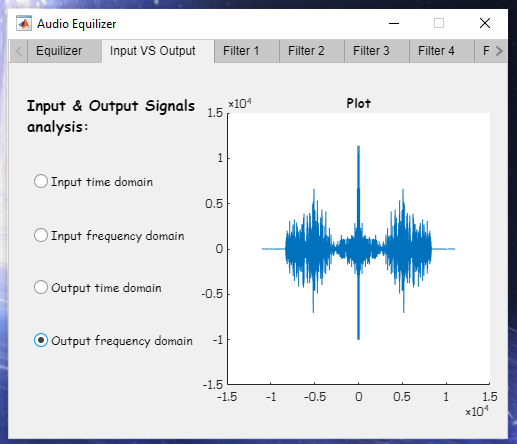
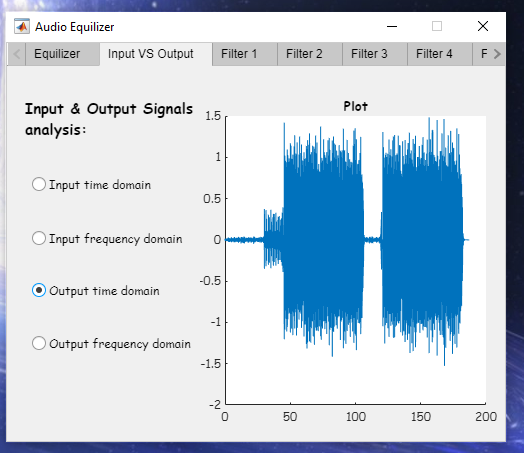
***Filters plotting:***

***Input signal plotting:***

A screenshot of a computer

Description automatically generated with medium confidence

***output signal plotting:***



***Input signal:***

***A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence***

***Double input signal plotting:***

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

***half output signal plotting:***

***A screenshot of a computer

Description automatically generated with medium confidenceA screen shot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence***