AHSANULLAH UNIVERSITY OF SCIENCE & TECHNOLOGY

Project Report

Course Title: Digital Signal Processing Lab.

Prepared By

Mohsin Islam Rifat

Department: EEE
Section : A2
Year: 3^{rd} Semester: 2^{nd}

Task-02: Designing a voice signal recorder and use of filtering.

In this task,we have to design a voice signal recorder with sampling frequency 44KHz for 12 seconds.Then we have to design a low pass filter using IIR filter method which frequency is 3.7 KHz.

After designing the LPF ,we will pass the recorded voice signal through the filter. The filter will filter out the noise signal containing above 3.7 KHz frequency.

Explaination of different Steps with Figure:

Step I:

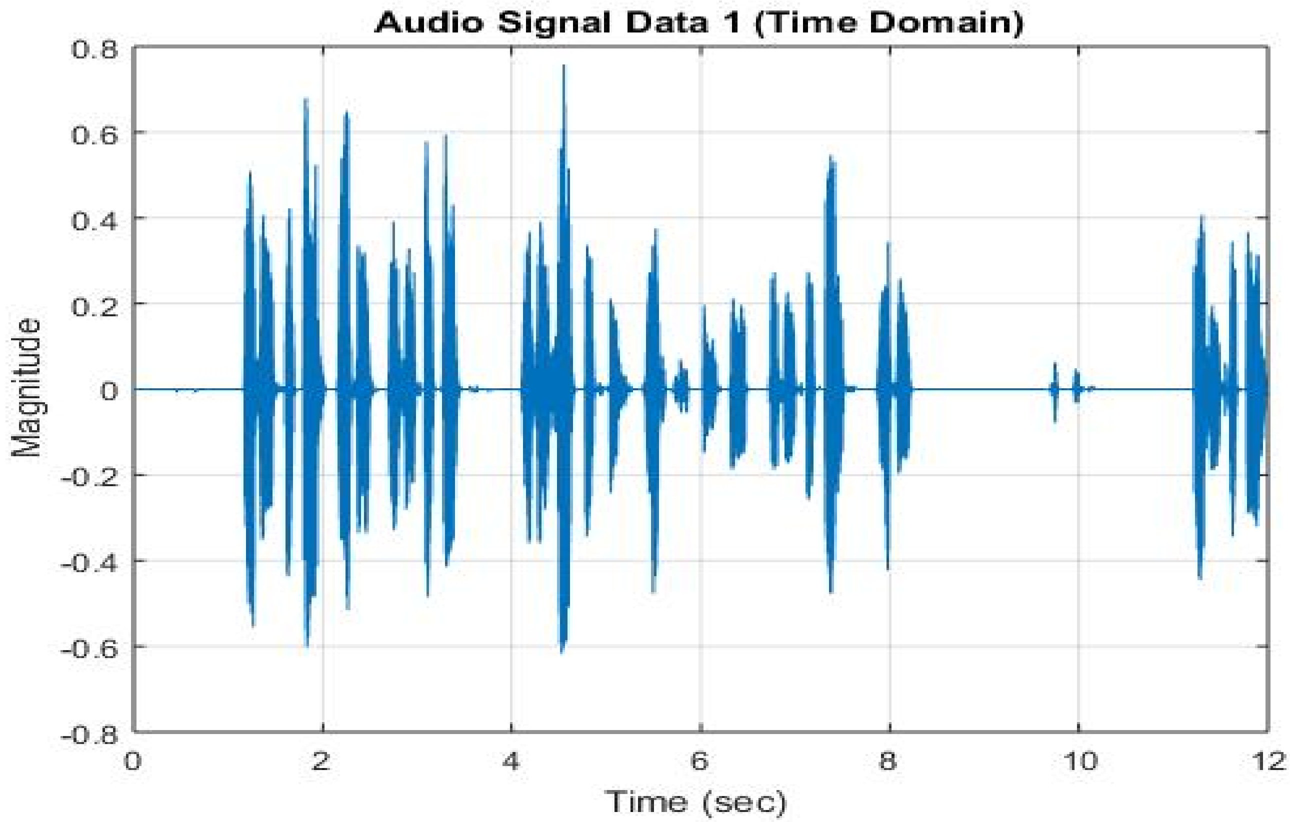
Here ,we have record a audio signal In matlab using the code given below;

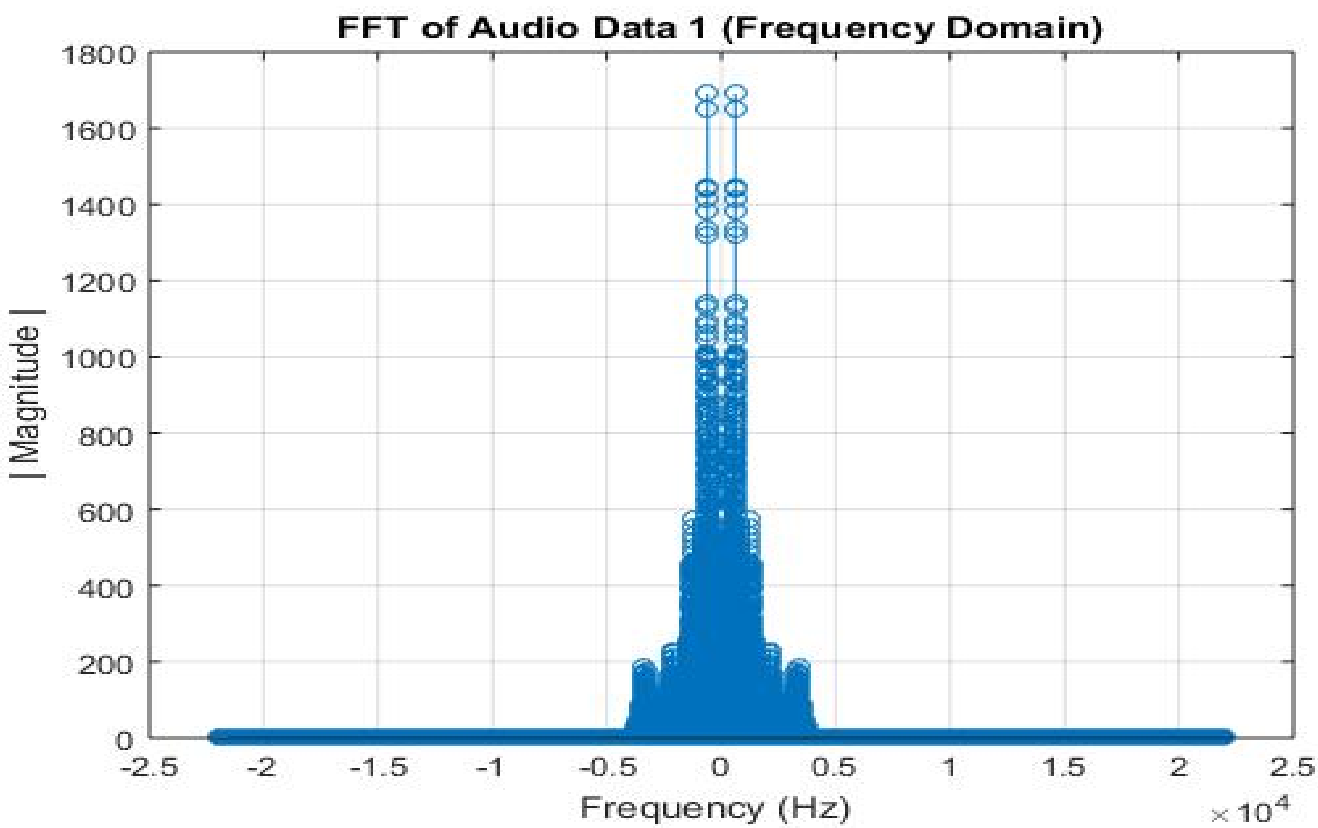
```
%% Record Audio

Fs = 44000 ; % sampling frequency 44Khz
ch = 1 ; % number of channels (Mono)
data_type = 'uint8' ; % Data type
nbits = 16 ; % number of bits
Nseconds = 12 ; % duration of the record

recorder_1=audiorecorder(Fs,nbits,ch)
disp('start speaking')
recordblocking(recorder_1,Nseconds);
disp('stop')
x1=getaudiodata(recorder_1,data_type);
audiowrite('Original_audio.wav',x1,Fs)

%%saving the audio data as "audio_data_1"
r_1=audioread('Original_audio.wav')

audio_data_1 = r_1.' ;

```

I used here matlab built in functions and kept the signal data at "audio_data_1" as 'Original_audio.wav'.

Step II:

The given figure shows that audio signal in Time domain & frequency domain;





Step III:

LPF designing using Ideal IIR filter design method where cut-off frequency is 3.7KHz.

Cutt-Off Frequency

Fc = 3700 Hz

sampling period

Ts = 1/Fs

Filter Pre-Wraped Frequency Calculation:

Digital Frequency

Wd = 2*pi*Fc

Pre-Wraped Frequency

Wa = (2/Ts)*tan((Wd*Ts)/2)

Analog Filter Coefficients ,

H(s) = 1/(1+s)

Numerator Coefficients

num = 1

Denominator Coefficients

den = 1 ,1

Filter Transformation from Low Pass to Low Pass :

[A, B] = lp2lp(num, den, Fc)

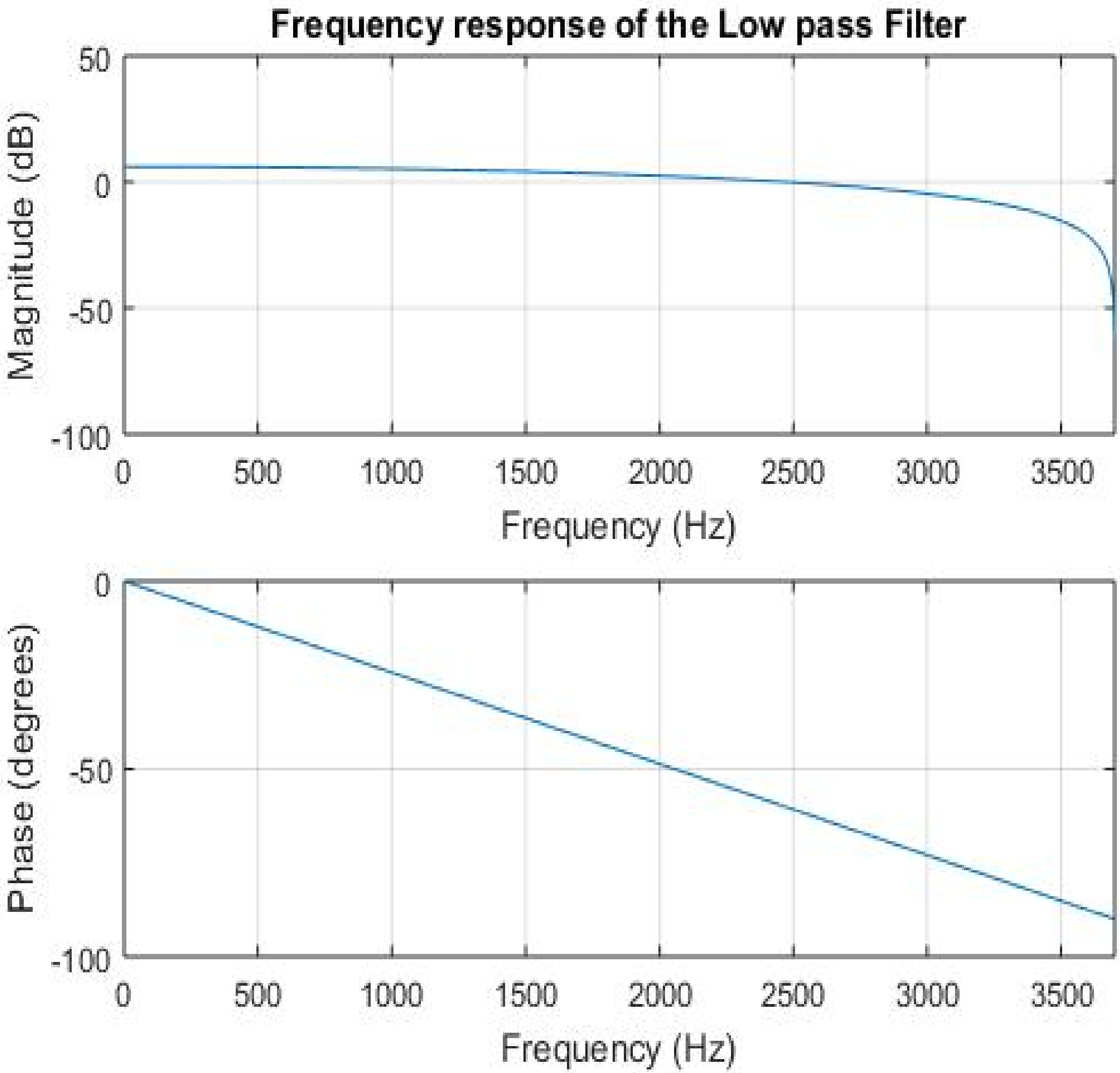
[a, b] = bilinear(A, B, Fs)

Frequency Response

[hz, fz] = freqz(a, b, N, Fs)

phi = 180*unwrap(angle(hz))/pi

Frequency response of the LPF filter is given below using Matlab :



Step IV:

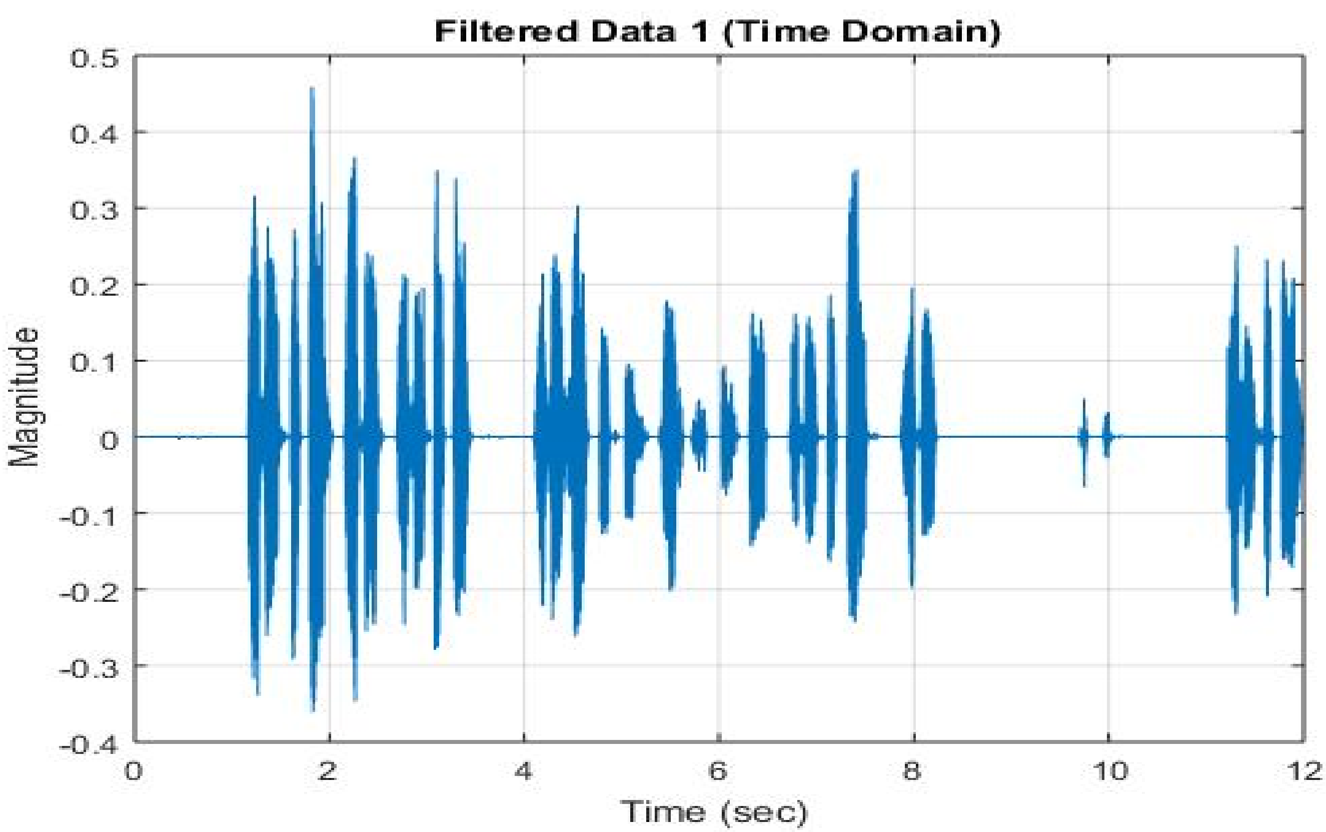
In this step,we pass the audio signal which is recorded using matlab to filter out the signal which contains of above frequency 3.7 KHz.

MATLAB CODE FOR FILTERING:

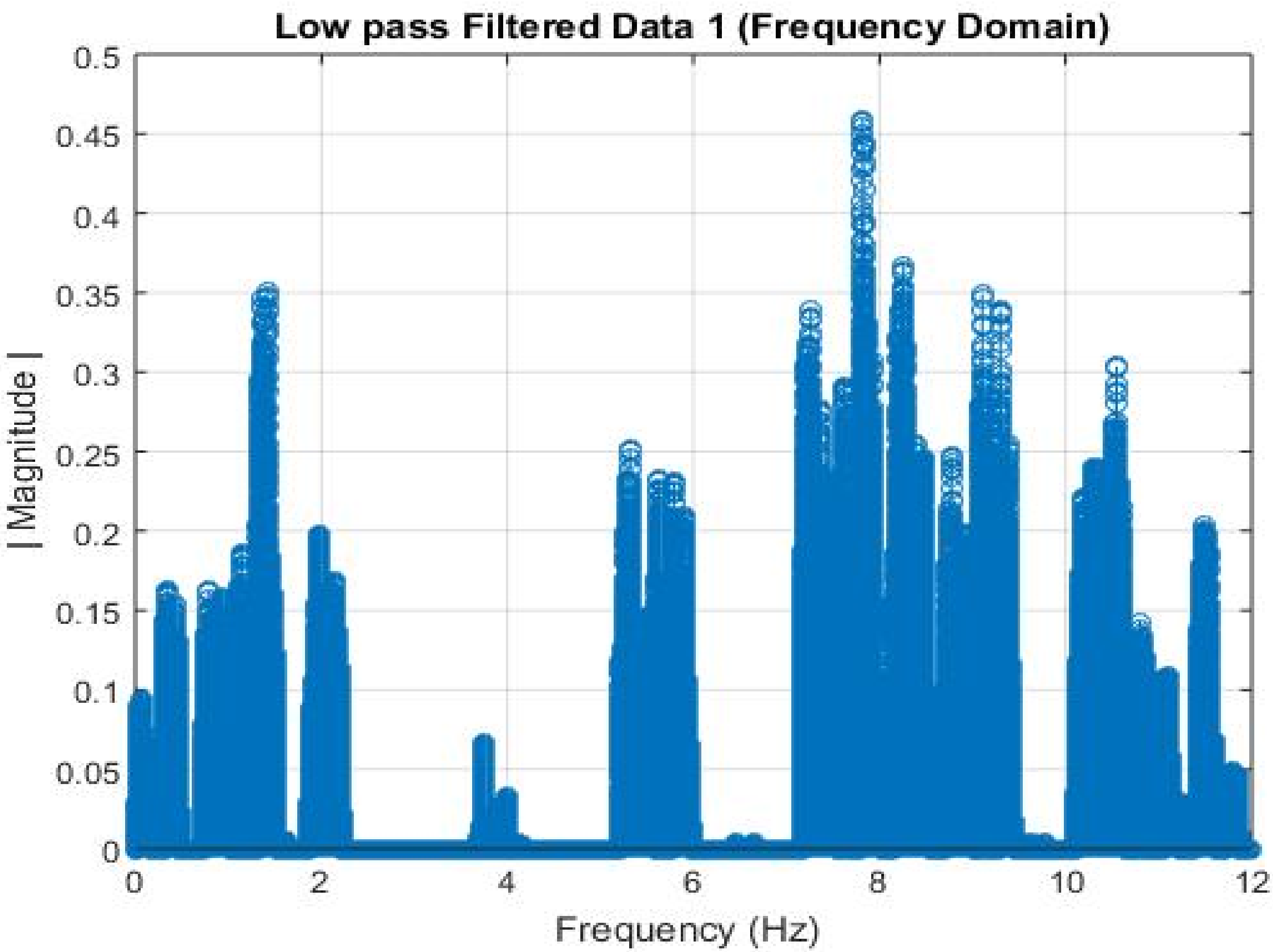
```
%% Filtering the Audio Data


% Filtering Audio Data 1
filtered_audio_data_1 = filter(a,b,audio_data_1) ;
audiowrite('filtered_output.wav',filtered_audio_data_1 ,Fs);
%% Ploting IIR Low Pass Filter Output
```

Output Of the LPF :

Filtered output in Time domain;



Filtered output in frequency domain;



Then the filter output is reconstructed as audio signal in .wav format.

Codes for audio reconstructed signal:

```
% Filtering Audio Data 1
filtered_audio_data_1 = filter(a,b,audio_data_1) ;
audiowrite('filtered_output.wav',filtered_audio_data_1 ,Fs);
```

From the original recorded signal’s frequency spectrum, we can see some noise & magnitude was to high.Then playing the sound it is observed that the sound is not so clear .

After filtering the signal, we can see the spectrum where the magnitude is belong to average value.And playing the filtered sound we can say that the soudnd is very smooth & it is like noise free.

MATLAB Code for Task-2:

```
clc;
clear all;
close all;

%% Record Audio

Fs = 44000 ; % sampling frequency 44Khz
ch = 1 ; % number of channels (Mono)
data_type = 'uint8' ; % Data type
nbits = 16 ; % number of bits
Nseconds = 12 ; % duration of the record

recorder_1=audiorecorder(Fs,nbits,ch)
disp('start speaking')
recordblocking(recorder_1,Nseconds);
disp('stop')
x1=getaudiodata(recorder_1,data_type);
audiowrite('Original_audio.wav',x1,Fs)

%%saving the audio data as "audio_data_1"
r_1=audioread('Original_audio.wav')

audio_data_1 = r_1.' ;


% Define Time Axis
dt = 1/Fs ;
t=0:1/Fs:(length(x1)-1)/Fs ;




%% Ploting the Audio Data
figure(1)
plot(t,audio_data_1) ;
title('Audio Signal Data 1 (Time Domain)')
grid on
xlabel('Time (sec) ')
ylabel('  Magnitude  ')


%% FFT of Audio Data

N = 262144 ; % FFT Point Number
df = Fs/N ;

% Define f axis for N point FFT
f = -Fs/2 : df : Fs/2-df ;

fft_audio_data_1 = fft(audio_data_1,N) ; % FFT of Audio Data 1

%% Ploting the Frequency Spectrums of Audio Data
figure(2)
stem(f,fftshift(abs(fft_audio_data_1)))
grid on
title(' FFT of Audio Data 1 (Frequency Domain) ')
grid on
xlabel('Frequency (Hz) ')
ylabel(' | Magnitude | ')

%% DesigLOW PASS IIR FILTER
```

```
%%
Fc = 3700 ; % Cutt-Off Frequency
Ts = 1/Fs ; % sampling period

% Filter Pre-Wraped Frequency Calculation
Wd = 2*pi*Fc ; % Digital Frequency
Wa = (2/Ts)*tan((Wd*Ts)/2) ; %pre-Wraped Frequency

% Analog Filter Coefficients H(s) = 1/(1+s)
num = 1 ; % Numerator Coefficients
den = [1 1] ; % Denominator Coefficients

% Filter Transformation from Low Pass to Low Pass
[A, B] = lp2lp(num, den, Fc) ;
[a, b] = bilinear(A, B, Fs) ;

% Frequency Response
[hz, fz] = freqz(a, b, N, Fs) ;
phi = 180*unwrap(angle(hz))/pi ;

%% Filtering the Audio Data

% Filtering Audio Data 1
filtered_audio_data_1 = filter(a,b,audio_data_1) ;
audiowrite('filtered_output.wav',filtered_audio_data_1 ,Fs);

%% Ploting IIR Low Pass Filter Output

figure(3)
plot(t,filtered_audio_data_1)
title(' Filtered Data 1 (Time Domain) ')
grid on
xlabel('Time (sec) ')
ylabel('  Magnitude  ')

figure(4)
stem(t,fftshift(abs(filtered_audio_data_1)))
title(' Low pass Filtered Data 1 (Frequency Domain) ')
grid on
xlabel('Frequency (Hz) ')
ylabel(' | Magnitude | ')

figure(5)
freqz(den,num,Fc,2*Fc)
title(' Frequency response of the Low pass Filter ')
```