Cairo University

Faculty of Engineering

Electronics and Electrical Communications Engineering Department

Third Year

Analog Communications

Term Project

MATLAB implementation of a superheterodyne receiver

Name	عمر محمد طلبه محمد	عمر محمد مصطفي عبد اللطيف
Sec	3	3
B.N	06	08
ID	9202985	9202989

Contents

Table of figures

1. *The transmitter*

This part contains the following tasks

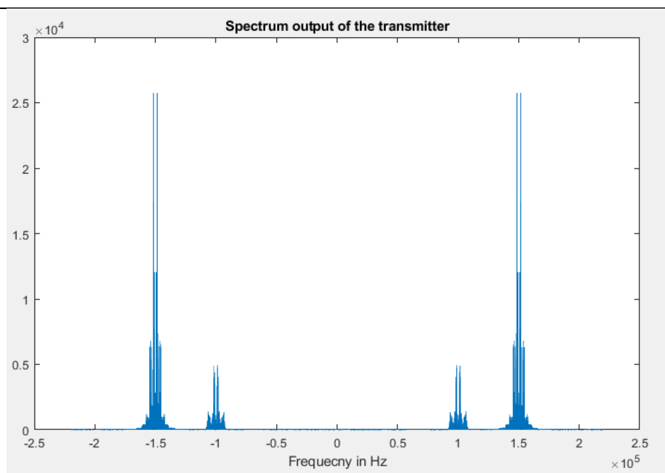
1. Reading monophonic audio signals into MATLAB.
2. Upsampling the audio signals.
3. Modulating the audio signals (each on a separate carrier).
4. Addition of the modulated signals.

Discussion

audioread() function given by MATLAB is used to read the signal and then by summing the signal two columns into one to turn the signal into monophonic to make it easier to deal with. Then we need to up samaple both signals to satisfy Nyquist criteria ($Fs \geq 2Fc$). Then we need to modulate both signals each in different Carrier to be able to Multiplex them into the channel (e.g air). Modulating the signal into higher Frequency decreases the size of antenna needed to transmit and receive the signal. then adding both signals by padding the smallest by zeros to deal with both modulated signals as one which represent the channel

The figures

Figure 1: The spectrum of the output of the transmitter



2. *The RF stage*

This part addresses the RF filter and the mixer following it.

Discussion

RF Band Pass Filter is needed to reject or attenuate the interference image or other signals except for the intended signal that my appear while De-Modulating the signal to Intermediate Frequency stage by the mixer. Mixer output which is Carrier Frequency + IF (intermediate frequency) is then multiplied to the output of the Band Pass Filter to De-Modulate the signal to intermediate frequency, which reduces flicker noise and DC offset Problem

The figures

Assume we want to demodulate the first signal (at ω_o).

Figure 2: the output of the RF filter (before the mixer)

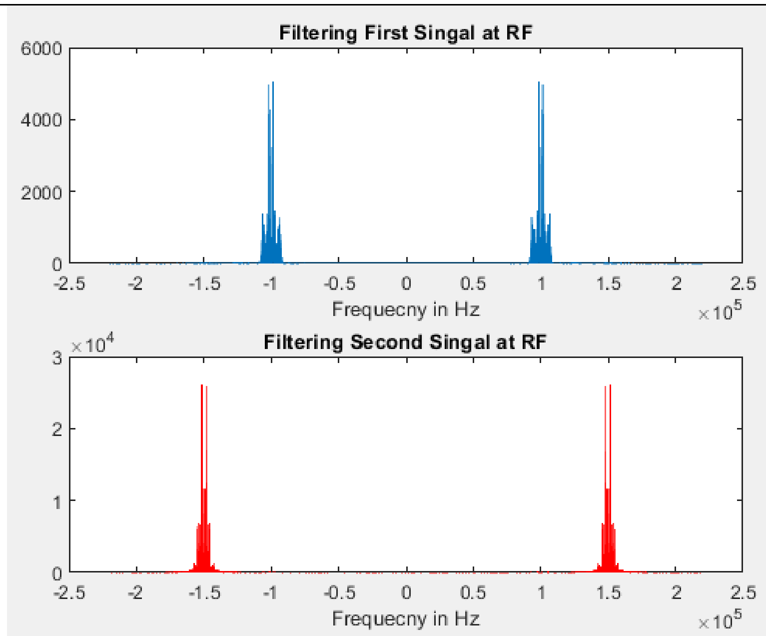
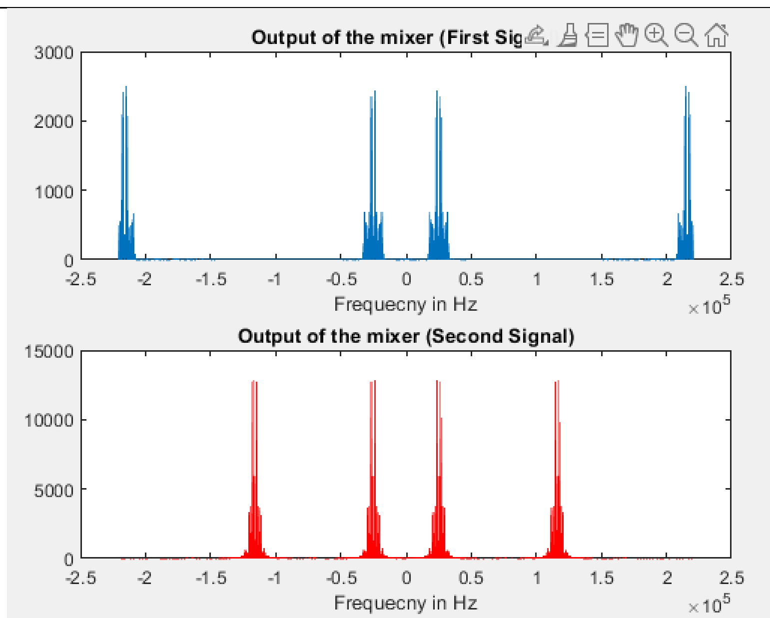


Figure 3: The output of the mixer



3. ***The IF stage***

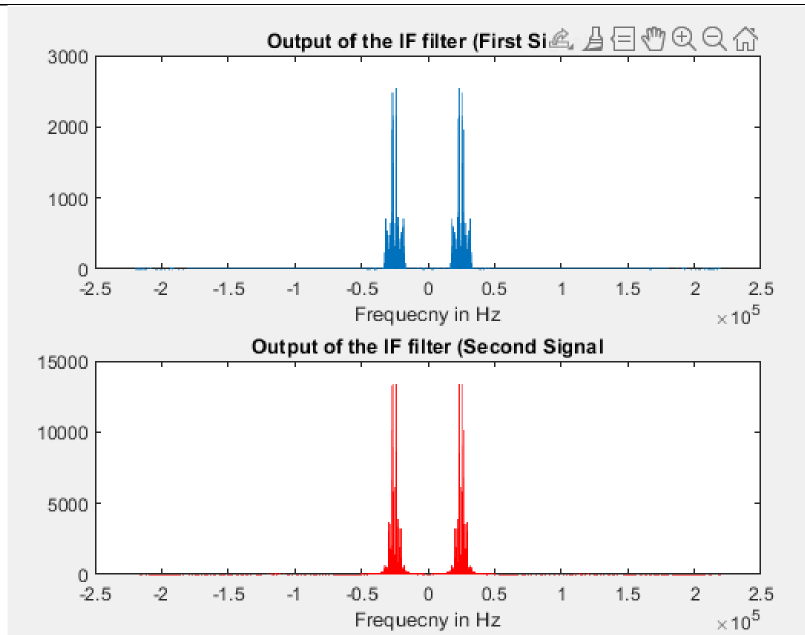
This part addresses the IF filter.

Discussion

After De-Modulating both signals to IF there will be alias of the signal at higher and lower Frequencies that can be seen if Figure 4. So, to eliminate or to attenuate these aliases we need to follow output of the mixer with IF Filter.

The figures

Figure 4: Output of the IF filter



4. ***The baseband demodulator***

This part addresses the coherent detector used to demodulate the signal from the IF stage.

Discussion

Now the signal Sands at intermediate frequency so to Demodulate the signal to the base band we multiply the signal to carrier with frequency equal to IF. Then will show the need of using LPF as we can see aliases showing again before the LPF so to eliminate or attenuate these aliases we use LPF with Stop Frequency value higher than or equal to the signal base band Bandwidth

The figures

Figure 5: Output of the mixer (before the LPF)

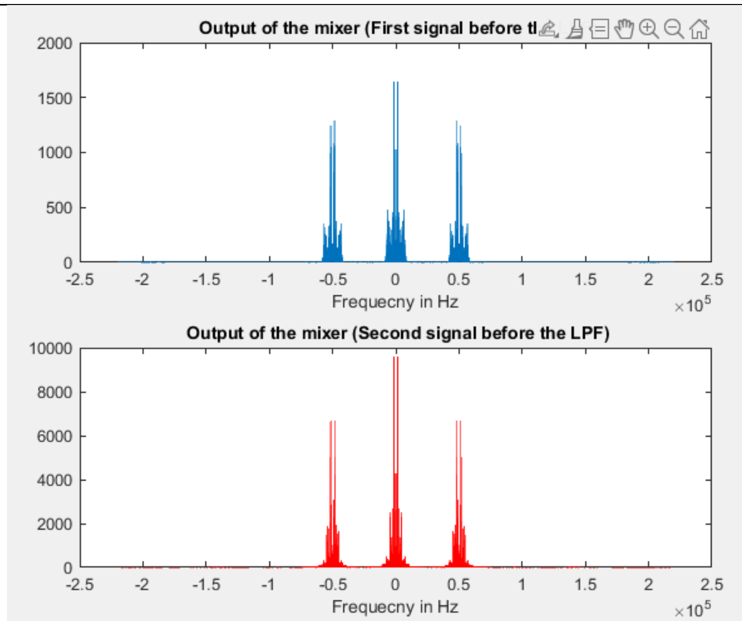
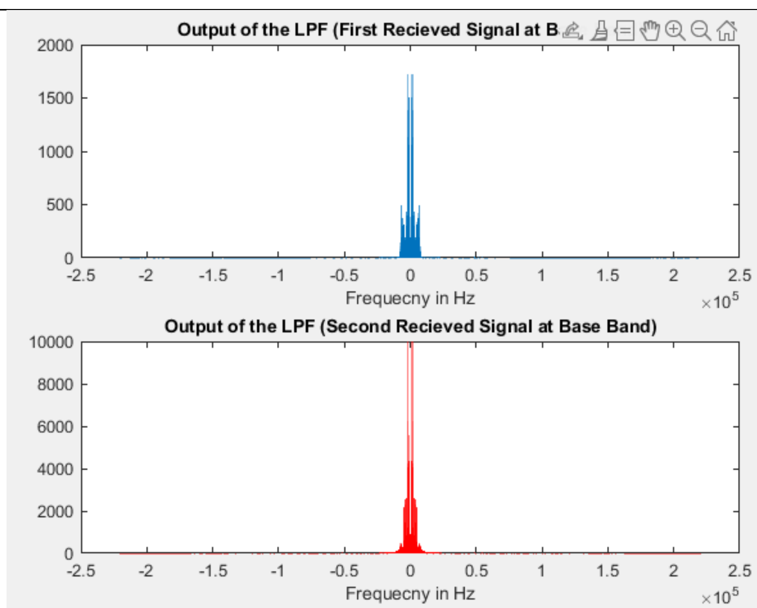


Figure 6: Output of the LPF



5. Performance evaluation without the RF stage

The figures

Figure 7: output of the RF mixer (no RF filter)

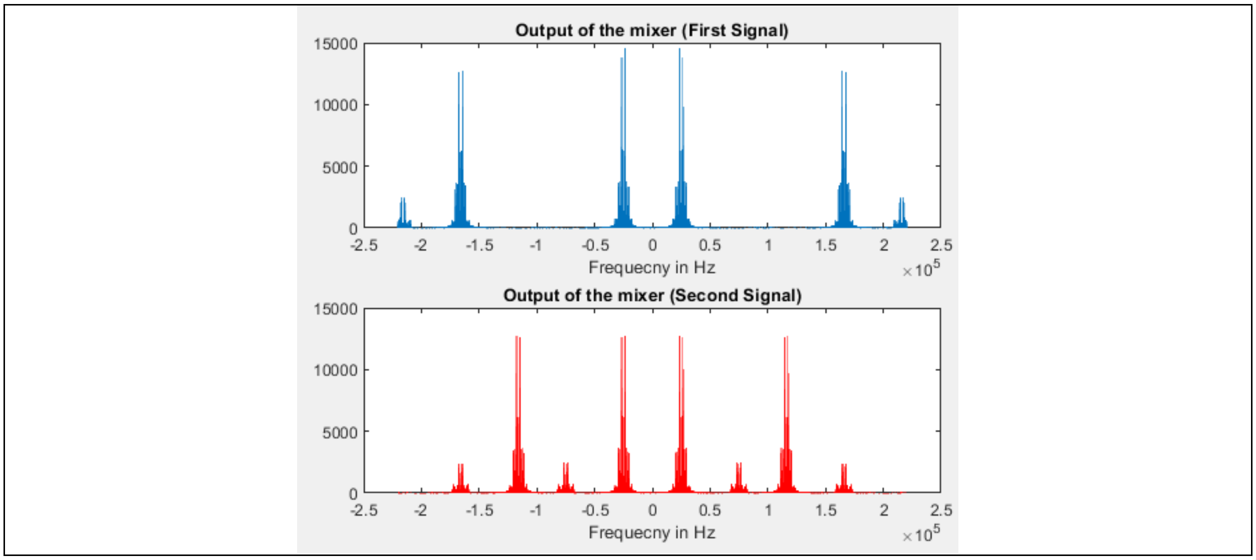


Figure 8: Output of the IF filter (no RF filter)

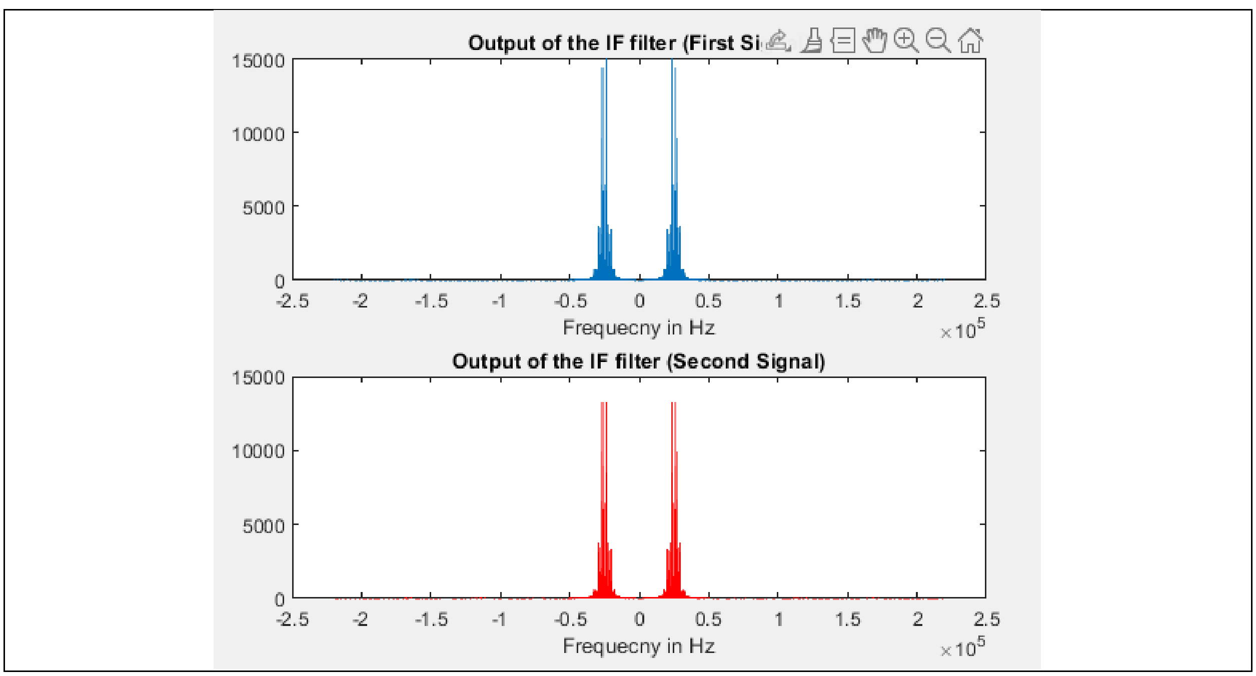


Figure 9: Output of the IF mixer before the LPF (no RF filter)

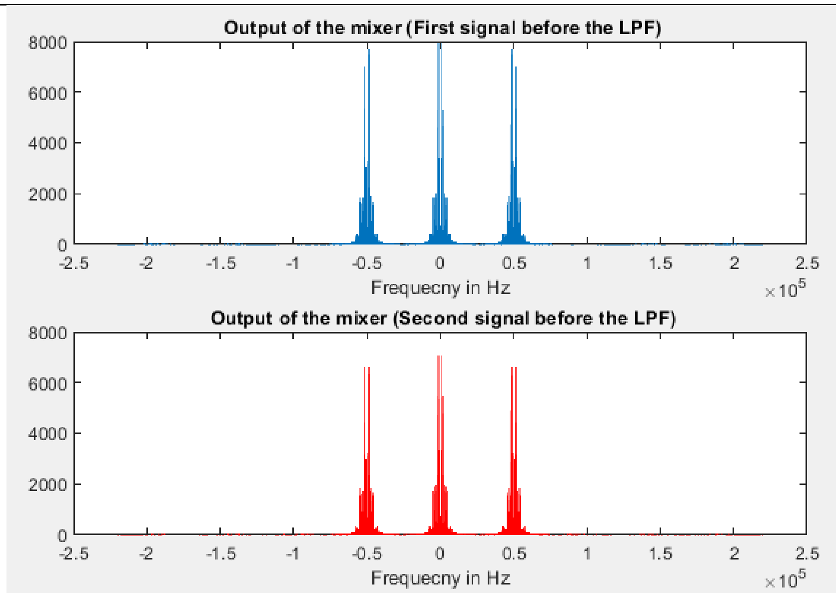
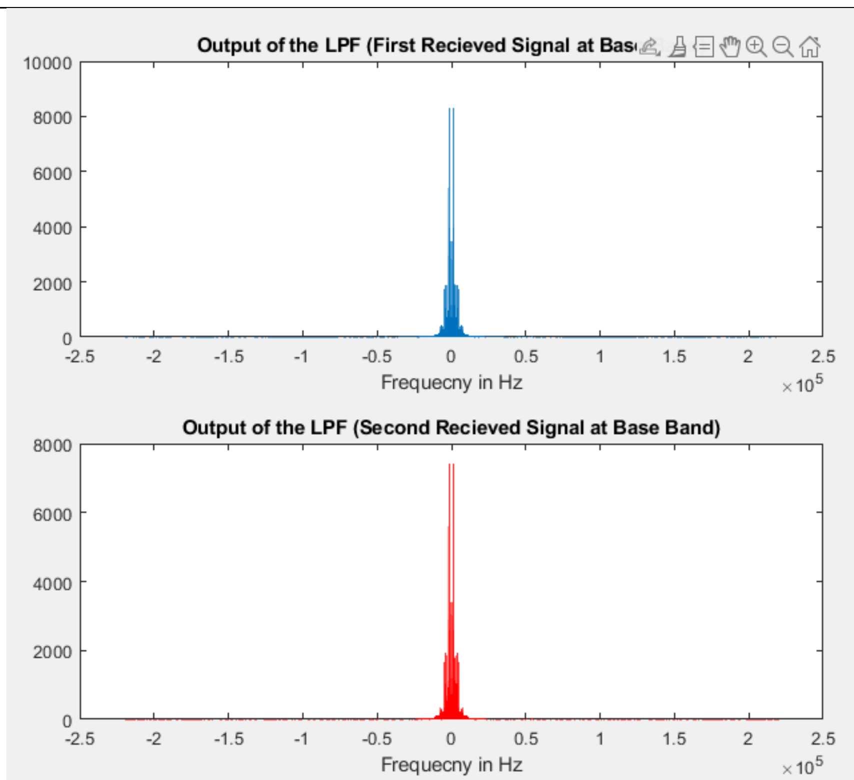


Figure 10: Output of the LPF (no RF filter)



6. ***Comment on the output sound***

Existence of RFBPF case: audio signals in transmitted and received correctly and you can hear the audio signal clearly with no distortion or interfering with any other signal.
Absence of RFBPF case: now you can hear interfered audio signals. Both signals (in our case 'Short_WRNArabic.wav' and 'Short_QuranPalestine.wav') are demodulated to IF also Image signal which was standing at $2\omega_{IF}$.
Please Note: this can be tested in the Code by un commenting lines **186, 201** and commenting lines **185, 200**.

What happens (in terms of spectrum and the sound quality) if the receiver oscillator has frequency offset by 0.1 KHz and 1 KHz

0.1KHz case: audio signal quality is badly affected although you can nearly hear and understand the sound as it was before.
1KHz case: now the heard audio signal quality is excessively decreased moreover you can't understand audio signal. This shows how phase shift between receiver and transmitter affects as the signal bandwidth interacts with itself more frequently.

7. ***The code***

Please Note: MATLAB used version is R2021a (9.10.0.1602886). also you can peek the code on GitHub by clicking ***here*** **(GitHub repo will turned public by project deadline day)**.

```
%{
 *  FILE DESCRIPTION
 *  -----------------------------------------------------------------------------------------------------
-----------
 *  File:         SuperHeterodyne.m
 *
 *  Description:  MATLAB file for Super-heterodyne Receiver Project
 *
 *  -----------------------------------------------------------------------------------------------------
-----------
 *  Author:       Omar Tolba & Omar Mustafa
 *  Date:         15/12/2022
%}
%% Reading Audio Signal and convert them into monophopic tone
% clear all WorkSpace Variables and Command Window
clear; clc;
% Reading the audio file and storing its data
[audioSample1,samplingFrequency1] = audioread('Short_WRNArabic.wav');
[audioSample2,samplingFrequency2] = audioread('Short_QuranPalestine.wav');
% Converting the two channels into monophonic
audioSample1 = audioSample1(:,1)+audioSample1(:,2);
audioSample2 = audioSample2(:,1)+audioSample2(:,2);
figure;
% Plotting audio samples
subplot(3,2,1)
plot(audioSample1);
title('Audio Samples for first signal Vs. time');
subplot(3,2,2)
plot(audioSample2,'-r');
title('Audio Samples for second signal Vs. time');

%% Calculating Base band BW
% Specifying the length of FFT
N=2^20;
```

```matlab
% Applying FFT
Y1=fft(audioSample1,N);
Y2=fft(audioSample2,N);
% Get the positive and negative frequencies
k=-N/2:N/2-1;
% Map it to actual frequencies --> note (samplingFrequency1==samplingFrequency2)
z=k*samplingFrequency1/N;
% Plotting FFT output against actual frequecnies
subplot(3,2,3);
plot(z,fftshift(abs(Y1)))
title('First Audio Signal in Freq. spectrum'); xlabel('Frequecny in Hz');
subplot(3,2,4);
plot(z,fftshift(abs(Y2)),'-r')
title('Second Audio Signal in Freq. spectrum'); xlabel('Frequecny in Hz');
% We can See that Audio Signal BaseBand Bw = 22 KHz

%% Resampling
% Resample audio data at a higher rate
resamplingFactor = 10;
audioSample1_ = interp(audioSample1,resamplingFactor);
audioSample2_ = interp(audioSample2,resamplingFactor);
% New sampling Frequency
newSamplingFrequency = resamplingFactor * samplingFrequency1;
% Sampling interval
Ts = 1/newSamplingFrequency;
% Time Intervals arrays --> note :t1!= t2
t1=0:Ts:Ts*(length(audioSample1_)-1);
t2=0:Ts:Ts*(length(audioSample2_)-1);

%% AM Modulatoion
% Carrier Frequecny for first signal
fc1 = 10e4;
% Carrier Frequecny for second signal
DeltaF=5e4; fc2 = 10e4+DeltaF;
% Carrier for first Signal
yc1 = cos(2*pi*fc1*t1);
% Carrier for second Signal
yc2 = cos(2*pi*fc2*t2);
% Modulation for first Signal
sm1 = yc1.*audioSample1_';
% Modulation for Second Signal
sm2 = yc2.*audioSample2_';
% using plot function for first signal
Ysm1=fft(sm1,N);
subplot(3,2,5);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm1)));
title('First Audio after Modulation'); xlabel('Frequecny in Hz');

% Analyzing in Freq. Spectrum for second audio singal
sa1=dsp.SpectrumAnalyzer('SampleRate',samplingFrequency2*10);
sa1.Name= 'Second signal after modulation';
%step(sa1,sm2');
% using plot function for first signal
Ysm2=fft(sm2,N);
subplot(3,2,6);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm2)),'-r');
title('Second Audio after Modulation'); xlabel('Frequecny in Hz');

%% padding the short signals with zeros so they have all equal length.
m1=length(sm1);
m2=length(sm2);
if m1>=m2
    for i=m2:m1
        sm2(1,i)=0;
        t=t1;
    end
else
     for i=m1:m2
         sm1(1,i)=0;
         t=t2;
     end
end

%% Multiplexing the two audio signals
sm_t = sm1+sm2;
% Analyzing in Freq. Spectrum for multiplixed signal
sa2=dsp.SpectrumAnalyzer('SampleRate',samplingFrequency2*10);
sa2.Name= 'Channel with the two modulated signal';
%step(sa2,sm_t');
% using plot function for the Channel with the two modulated signal
Ysm_t=fft(sm_t,N);
```

```
figure;
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm_t)));
title('Spectrum output of the transmitter'); xlabel('Frequecny in Hz');

%% RF BandBass Filter design for first signal
% important note : first signal BW = 22 kHz and modulated with 100kHz
% carrier. So we need BPF from 78kHz to 122KHz
A_stop1 = 60;               % Attenuation in the first stopband = 60 dB
F_stop1 = (fc1 - 3e4);      % Edge of the stopband = 70 kHz
F_pass1 = (fc1 - 2.5e4);    % Edge of the passband = 75 kHz
F_pass2 = (fc1 + 2.5e4);    % Closing edge of the passband = 125 kHz
F_stop2 = (fc1 + 3e4);      % Edge of the second stopband = 130 kHz
A_stop2 = 60;               % Attenuation in the second stopband = 60 dB
A_pass = 1;     % Amount of ripple allowed in the passband = 1 dB
% Creating a filter specification object with sampling fs = samplingFrequency1,2*10
BandPassSpecObj = ...
   fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2', ...
        F_stop1, F_pass1, F_pass2, F_stop2, A_stop1, A_pass, ...
        A_stop2, newSamplingFrequency);
BandPassFilt = design(BandPassSpecObj);
% fvtool(BandPassFilt) % plot the filter magnitude response

%% RF BandBass Filter design for second signal
% important note : first signal BW = 22 kHz and modulated with 150kHz
% carrier. So we need BPF from 128kHz to 172KHz
A_stop1 = 60;       % Attenuation in the first stopband = 60 dB
F_stop1 = (fc1+DeltaF - 3e4);      % Edge of the stopband = 120 kHz
F_pass1 = (fc1+DeltaF - 2.5e4);    % Edge of the passband = 125 kHz
F_pass2 = (fc1+DeltaF + 2.5e4);    % Closing edge of the passband = 175 kHz
F_stop2 = (fc1+DeltaF + 3e4);      % Edge of the second stopband = 180 kHz
A_stop2 = 60;       % Attenuation in the second stopband = 60 dB
A_pass = 1;     % Amount of ripple allowed in the passband = 1 dB
% Creating a filter specification object with sampling fs = samplingFrequency1,2*10
BandPassSpecObj2 = ...
   fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2', ...
        F_stop1, F_pass1, F_pass2, F_stop2, A_stop1, A_pass, ...
        A_stop2, newSamplingFrequency);
BandPassFilt2 = design(BandPassSpecObj2);
%fvtool(BandPassFilt2) % plot the filter magnitude response

%% RFBPF. Filtering First Singal
sm1_filtered_RFBPF = filter(BandPassFilt,sm_t');
% Analyzing in Freq. Spectrum for multiplixed signal
sa3=dsp.SpectrumAnalyzer('SampleRate',newSamplingFrequency);
sa3.Name= 'Channel with the two modulated signal after RFBPF';
%step(sa3,sm1_filtered_RFBPF);
% using plot function for Channel after BPF
Ysm1_RFBPF=fft(sm1_filtered_RFBPF,N);
figure;
subplot(2,1,1);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm1_RFBPF)));
title('Filtering First Singal at RF'); xlabel('Frequecny in Hz');

%% RFBPF. Filtering Second Singal
sm2_filtered_RFBPF = filter(BandPassFilt2,sm_t');
% Analyzing in Freq. Spectrum for multiplixed signal
sa4=dsp.SpectrumAnalyzer('SampleRate',newSamplingFrequency);
sa4.Name= 'Channel with the two modulated signal after RFBPF';
%step(sa4,sm2_filtered_RFBPF);
% using plot function for Channel after BPF
Ysm2_RFBPF=fft(sm2_filtered_RFBPF,N);
subplot(2,1,2);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm2_RFBPF)),'-r');
title('Filtering Second Singal at RF'); xlabel('Frequecny in Hz');

%% DeModulatoion-stage -> first signal
% Intermediate freq. value
Fif = 2.5e4; freqShift = 0;
% Carrier for demodulating first signal
yc1_if = cos(2*pi*(Fif+fc1+freqShift)*t);
% De-Modulation for first Signal
sm1_IFDemod = yc1_if.*sm1_filtered_RFBPF';
% sm1_IFDemod = yc1_if.*sm_t; %this line to simulate removing RFBPF
% Analyzing in Freq. Spectrum for first demodulated signal
figure;
subplot(2,1,1);
Ysm1_IFDemod=fft(sm1_IFDemod,N);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm1_IFDemod)));
title('Output of the mixer (First Signal)'); xlabel('Frequecny in Hz');

%% DeModulatoion-stage -> second signal
```

```
% Intermediate freq. value
Fif = 2.5e4;
% Carrier for demodulating second signal
yc2_if = cos(2*pi*(Fif+fc2+freqShift)*t);
% De-Modulation for second Signal
sm2_IFDemod = yc2_if.*sm2_filtered_RFBPF';
%  sm2_IFDemod = yc2_if.*sm_t; %this line to simulate removing RFBPF
% Analyzing in Freq. Spectrum for first demodulated signal
subplot(2,1,2);
Ysm2_IFDemod=fft(sm2_IFDemod,N);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm2_IFDemod)),'-r');
title('Output of the mixer (Second Signal)'); xlabel('Frequecny in Hz');

%% IF Stage. IF BandBass Filter design for first signal
% important note : first signal BW = 22 kHz and Carried at Fif= 25kHz
% carrier. So we need BPF from 3kHz to 47KHz
A_stop1 = 60;       % Attenuation in the first stopband = 60 dB
F_stop1 = (Fif-2.3e4);      % Edge of the stopband = 2 kHz
F_pass1 = (Fif-2.2e4);      % Edge of the passband = 3 kHz
F_pass2 = (Fif+2.5e4);      % Closing edge of the passband = 50 kHz
F_stop2 = (Fif+3e4);    % Edge of the second stopband = 55 kHz
A_stop2 = 60;       % Attenuation in the second stopband = 60 dB
A_pass = 1;     % Amount of ripple allowed in the passband = 1 dB
% Creating a filter specification object with sampling fs = samplingFrequency1,2*10
BandPassSpecObj3 = ...
   fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2', ...
        F_stop1, F_pass1, F_pass2, F_stop2, A_stop1, A_pass, ...
        A_stop2, newSamplingFrequency);
BandPassFilt3 = design(BandPassSpecObj3);
% fvtool(BandPassFilt3) % plot the filter magnitude response
% Please Note = both signals need same Filters specification. so, Both
% signals will use BandPassFilt3

%% IFBPF. Filtering First Singal
sm1_IF_filtered = filter(BandPassFilt3,sm1_IFDemod');
% Analyzing in Freq. Spectrum for multiplixed signal
sa5=dsp.SpectrumAnalyzer('SampleRate',newSamplingFrequency);
sa5.Name= 'Filtering First Singal at IF stage';
%step(sa5,sm1_IF_filtered);
% using plot function for Channel after BPF
Ysm1_IF_filtered=fft(sm1_IF_filtered,N);
figure;
subplot(2,1,1);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm1_IF_filtered)));
title('Output of the IF filter (First Signal)'); xlabel('Frequecny in Hz');

%% IFBPF. Filtering Second Singal
sm2_IF_filtered = filter(BandPassFilt3,sm2_IFDemod');
% Analyzing in Freq. Spectrum for multiplixed signal
sa6=dsp.SpectrumAnalyzer('SampleRate',newSamplingFrequency);
sa6.Name= 'Filtering Second Singal at IF stage';
%step(sa6,sm2_IF_filtered);
% using plot function for Channel after BPF
Ysm2_IF_filtered=fft(sm2_IF_filtered,N);
subplot(2,1,2);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm2_IF_filtered)),'-r');
title('Output of the IF filter (Second Signal)'); xlabel('Frequecny in Hz');

%% Baseband detection-stage -> first signal
% Carrier for demodulating first signal
yc1_BB = cos(2*pi*(Fif)*t);
% De-Modulation for first Signal
sm1_BB = yc1_BB.*sm1_IF_filtered';
% Analyzing in Freq. Spectrum for first demodulated signal at base band
figure;
subplot(2,1,1);
Ysm1_demod_BB=fft(sm1_BB,N);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm1_demod_BB)));
title(' Output of the mixer (First signal before the LPF)'); xlabel('Frequecny in Hz');


%% Baseband detection-stage -> second signal
% Intermediate freq. value
Fif = 2.5e4;
% Carrier for demodulating second signal
yc2_BB = cos(2*pi*(Fif)*t);
% De-Modulation for second Signal
sm2_demod = yc2_BB.*sm2_IF_filtered';
% Analyzing in Freq. Spectrum for second demodulated signal
subplot(2,1,2);
Ysm2_demod_BB=fft(sm2_demod,N);
```

```
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm2_demod_BB)),'-r');
title(' Output of the mixer (Second signal before the LPF)'); xlabel('Frequecny in Hz');

%% filtering Both Signals
%  filter specifications
Fp = 2.2e4; Fst = 2.5e4;
% Designing LowPass Filter Fp =2.2e4, Fst = 2.5e4, Ap(amount of ripple allowed in the pass band)= 1,
% And Ast(attenuation in the stop band) = 80.
LowPassFilter = design(fdesign.lowpass('Fp,Fst,Ap,Ast',Fp,Fst,1,80,newSamplingFrequency));
%fvtool(LowPassFilter);
% filtering first signal
sm1_recieved = filter(LowPassFilter,sm1_BB);
sm2_recieved = filter(LowPassFilter,sm2_demod);
% Analyzing in Freq. Spectrum for Both Signals signal
Ysm1_received=fft(sm1_recieved,N);
Ysm2_received=fft(sm2_recieved,N);
figure;
% Plotting first singal
subplot(2,1,1);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm1_received)));
title('Output of the LPF (First Recieved Signal at Base Band)'); xlabel('Frequecny in Hz');
% Plotting second singal
subplot(2,1,2);
plot(k*newSamplingFrequency/N,fftshift(abs(Ysm2_received)),'-r');
title('Output of the LPF (Second Recieved Signal at Base Band)'); xlabel('Frequecny in Hz');

%% Down Sampling Both signals
% Down Sampling first signal by factor 10
firstSignal = downsample(sm1_recieved,10);
% This line need to be unCommented to play the first signal
% sound(firstSignal,48000);
% Down Sampling Second signal by factor 10
secondSignal = downsample(sm2_recieved,10);
% This line need to be unCommented to play the second signal
sound(secondSignal,48000);
```