Cairo University

Faculty of Engineering

Electronics and Electrical Communications Engineering Department

Third Year

Analog Communications

Term Project

MATLAB implementation of a superheterodyne receiver

Student Name: محمد ناصر محمد كمال

Sec: 4 BN: 3 ID: 9203349

Student Name: هنا إيهاب خاطر يونس

Sec: 4 BN: 42 ID: 9203687

Contents

Table of figures

1. *The transmitter*

This part contains the following tasks

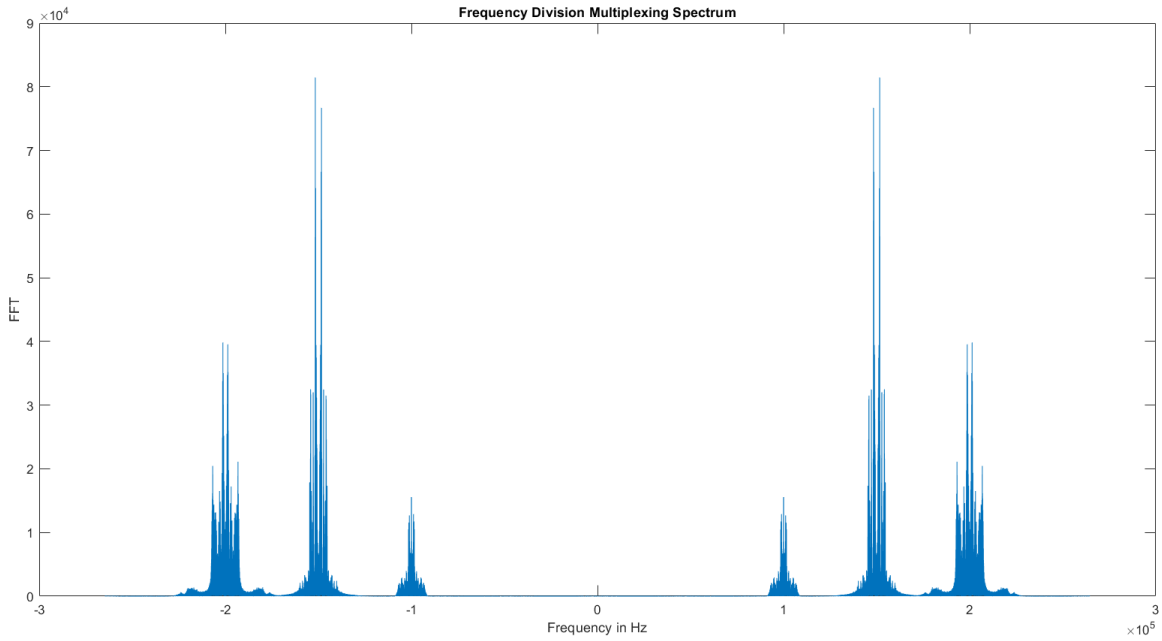
1. Reading monophonic audio signals into MATLAB.
2. Upsampling the audio signals.
3. Modulating the audio signals (each on a separate carrier).
4. Addition of the modulated signals.

Discussion

First we read 3 audio signals & added their 2 stereo columns to transform them into monophonic signals as we implement monophonic receiver, then we modified their sizes to fit the longest one using padding, after that we up-sampled the signals to fit the highest sampling rate of their highest carrier (fs>=2*200 KHz), then we generated 3 carriers, multiplied by 3 signals & adding the 3 results together to construct FDM signal.

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

As we intend to move signals to intermediate frequency IF, we may suffer the problem of IF image so we use bandpass filter to select the intended received signal & reject the others then multiply the filtered signal by a carrier with frequency = $f_c + f_{IF}$ to move the signal to IF.

The figures

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

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

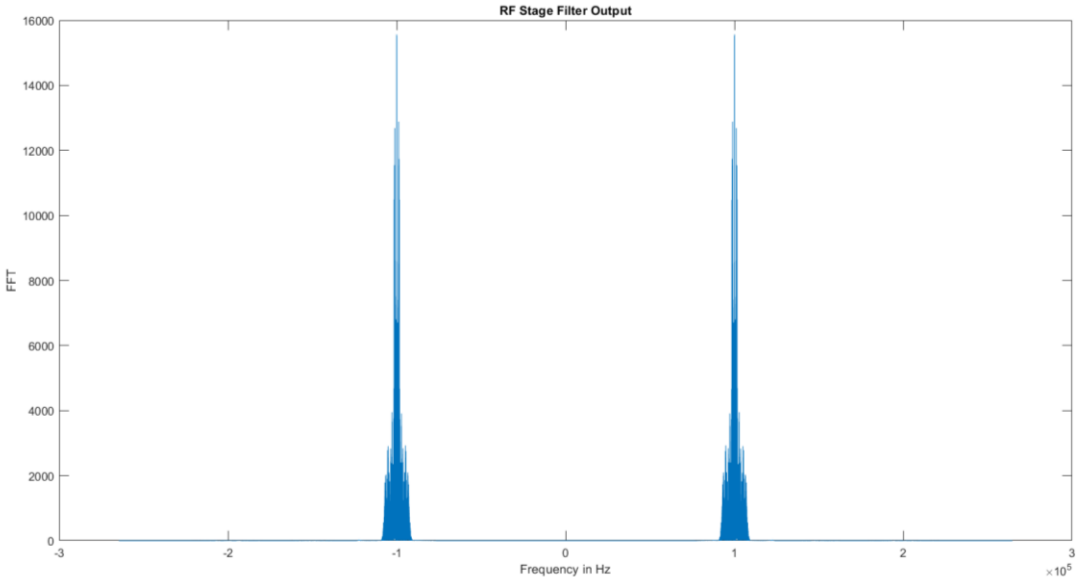
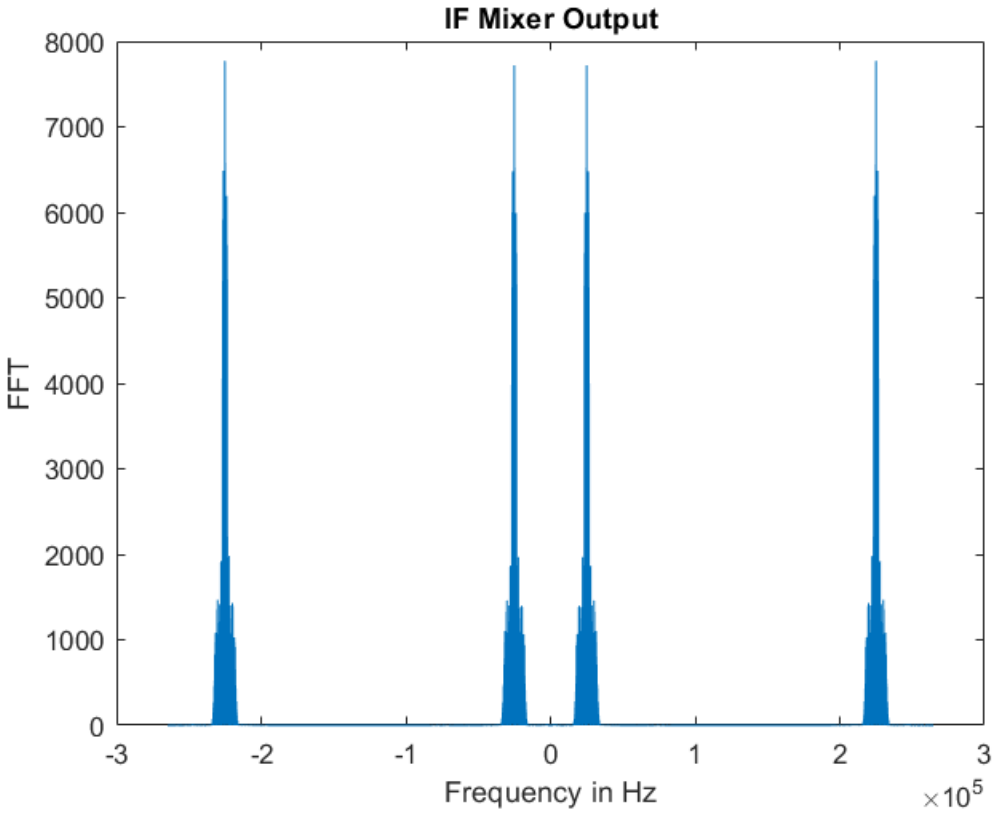


Figure 3: The output of the mixer



3. The IF stage

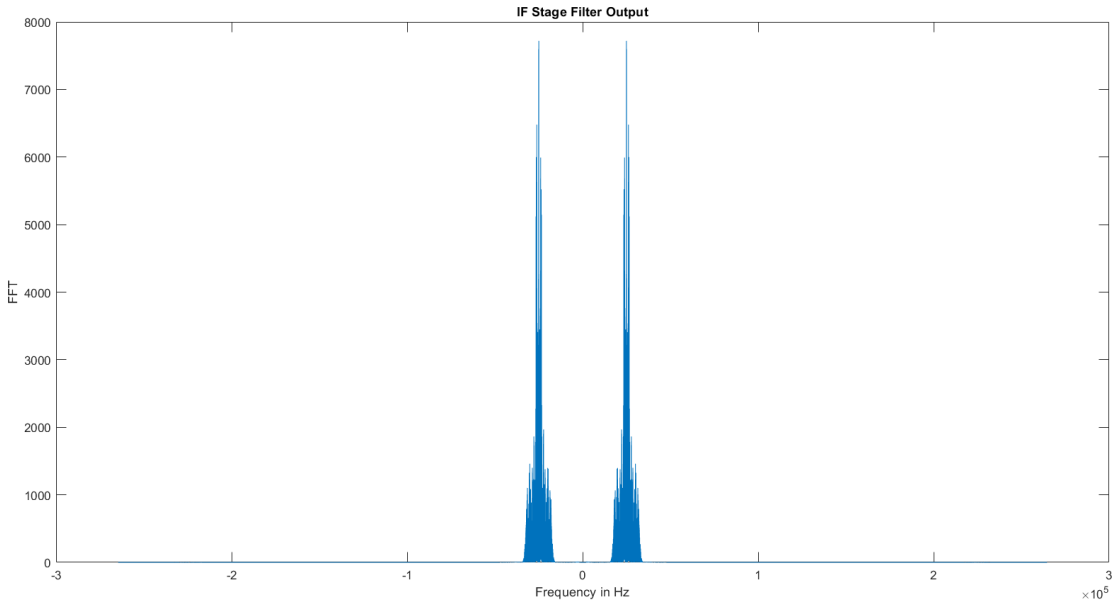
This part addresses the IF filter.

Discussion

After IF mixer stage signal is carried to IF as intended but it also has carried to higher frequency band, so we need a bandpass filter to select the IF signal & reject its higher frequency version, and the importance of carrying the received signal on IF before baseband to get rid of leakage & flicker noise & to improve the filters' selectivity.

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

The previous modulated signal needs to be demodulated to return the signal to the baseband, we will multiply it by a carrier with frequency = f_{IF} to perform coherent demodulation then use a lowpass filter to select the baseband signal & reject higher frequencies then multiplied by a gain = 4.5 (due to carriers the signal's amplitude was reduced 4 times), now if we use the sound command we can successfully listen to the original audio message.

The figures

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

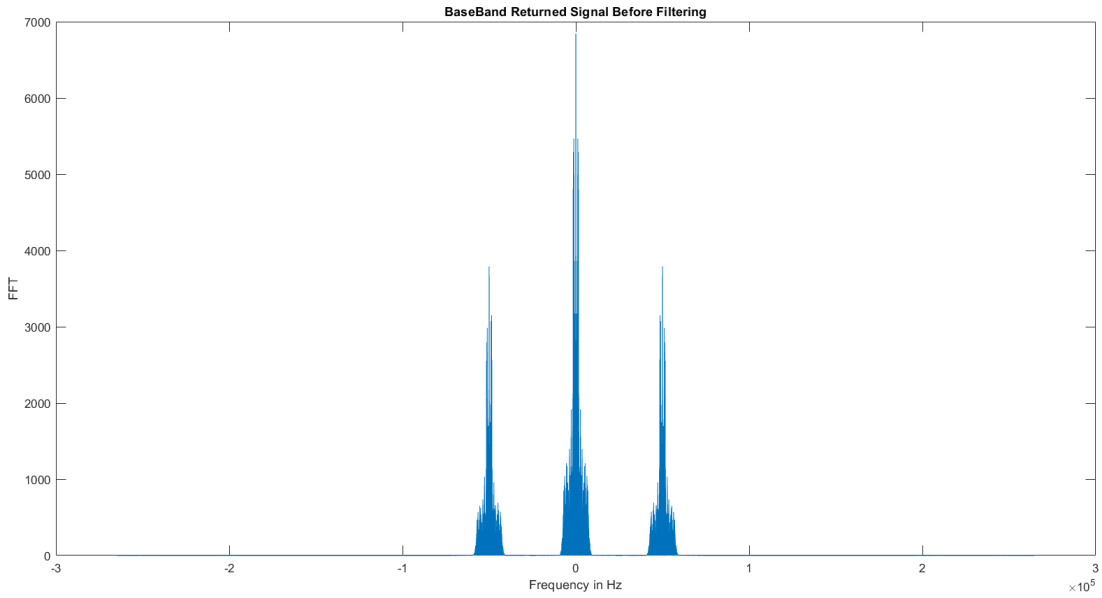
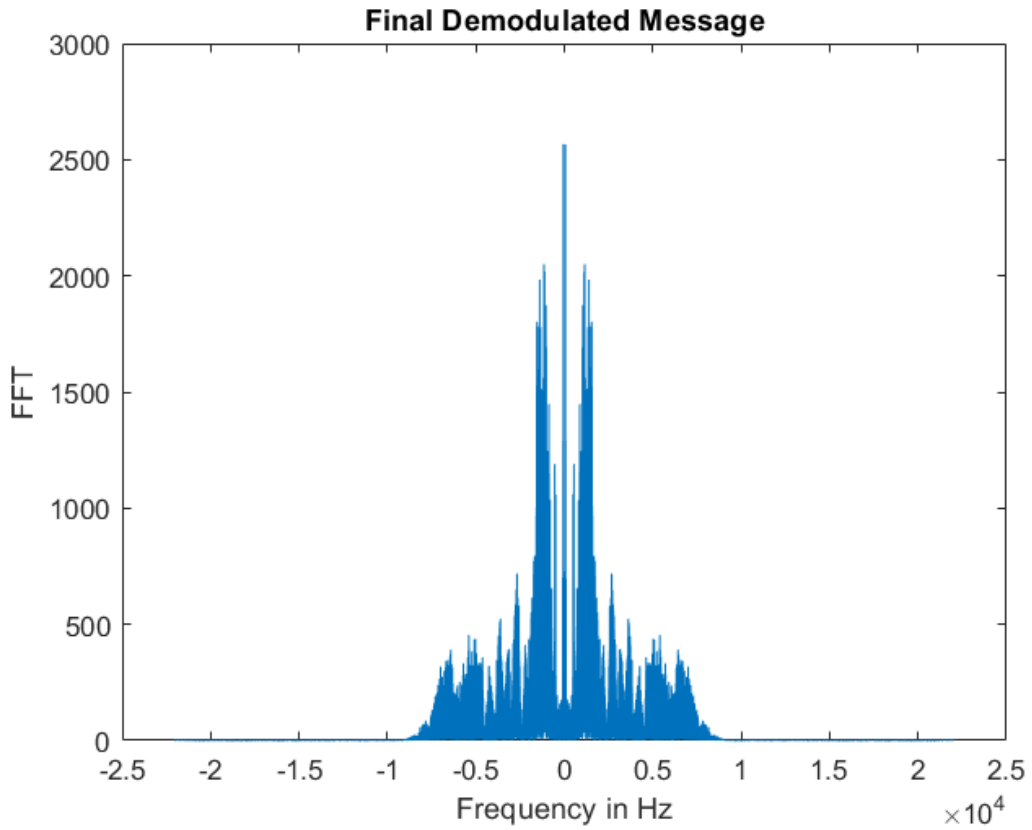


Figure 6: Output of the LPF



5. ***Performance evaluation without the RF stage***

The figures

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

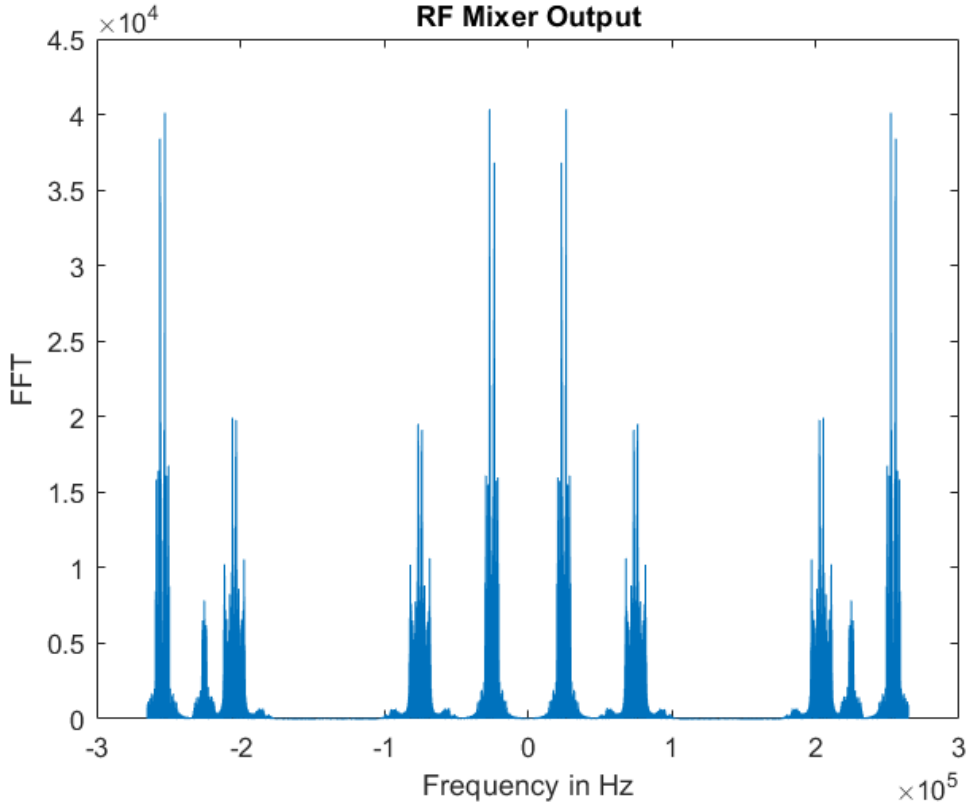


Figure 8: Output of the IF filter (no RF filter)

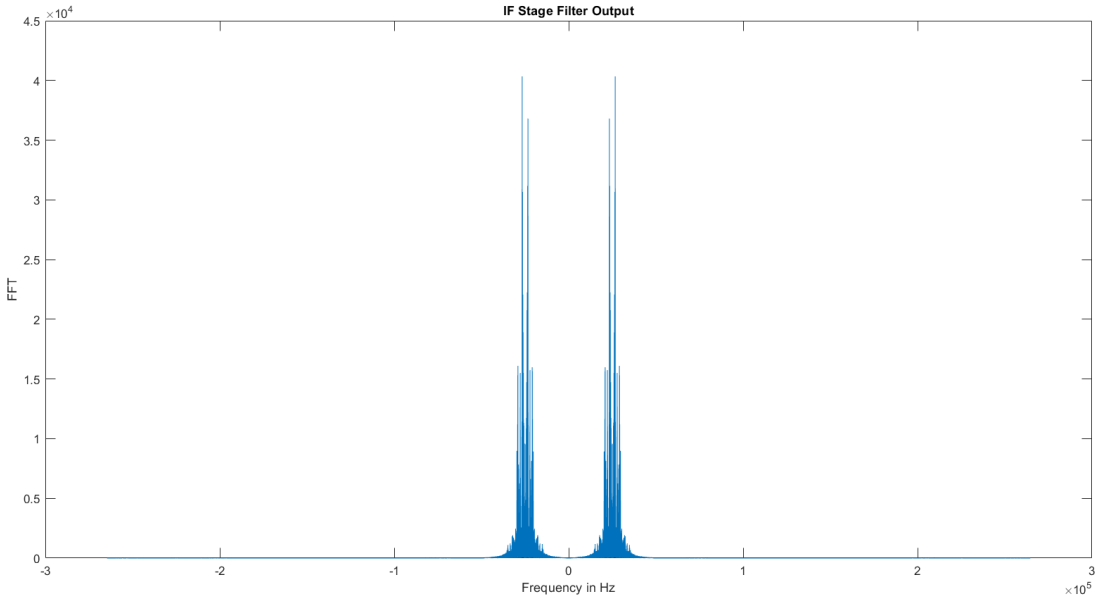


Figure 9: Output of the IF mixer before the LPF (no RF filter)

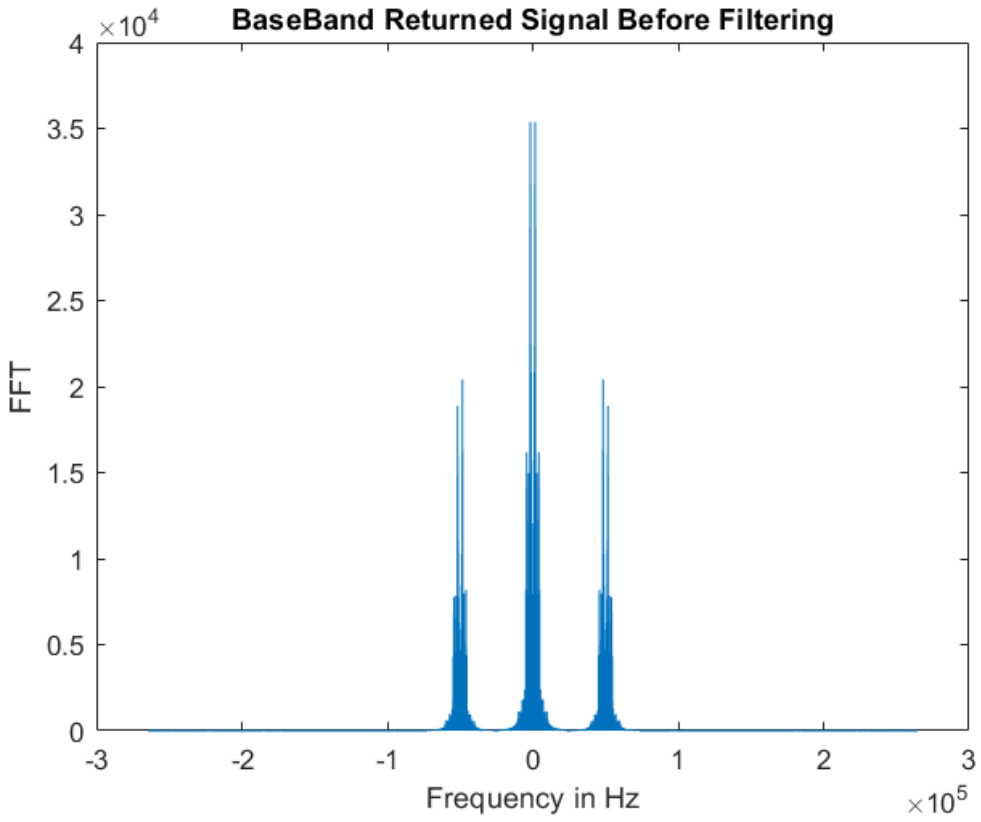
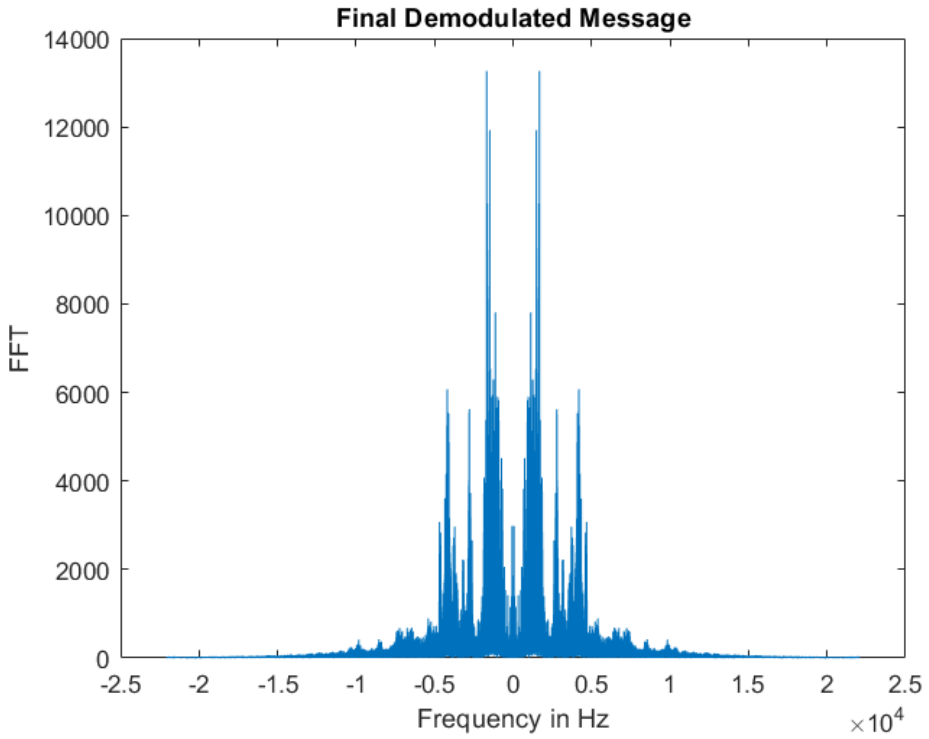


Figure 10: Output of the LPF (no RF filter)



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

In the existence of RF stage we chose the signal with f_c = 100 KHz (Sky News Message) we heard at the receiver this message clearly without any difference between it and the original one, but after removing the RF stage an interference occurred between the required signal and the signal with $f_c = 100 + 2*f_{IF}$ KHz (150 KHz) it was in our program Quran message which causes IF image to interfere with the required signal and we heard both signals at the receiver's output.

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

1. Offset = 0.1 KHz: the sound was a little bit distorted; it is different than the original sound but still recognizable & understood.

2. Offset = 1 KHz: the sound was totally distorted; it is different than the original and neither recognizable nor understood.

This is due to that the signal has not been carried to the f_{IF} exactly but it is shifted so the coherent demodulator with f = 25 KHz has not done its job correctly.

7. The code

```
%% The Transmitter
% Read Audio Signals
[sky_news_msg,sampling_rate] = audioread('Short_SkyNewsArabia.wav');
[quran_msg,sampling_rate] = audioread('Short_QuranPalestine.wav');
[russian_msg,sampling_rate] = audioread('Short_RussianVoice.wav');

% Making Signals Monophonic
sky_news_msg = sky_news_msg(:,1) + sky_news_msg(:,2);
quran_msg = quran_msg(:,1) + quran_msg(:,2);
russian_msg = russian_msg(:,1) + russian_msg(:,2);

% Padding Short Signals With Size Of Longest Signal
s1 = size(sky_news_msg);
s1 = s1(1);
s2 = size(quran_msg);
s2 = s2(1);
s3 = size(russian_msg);
s3 = s3(1);
N = max([s1 s2 s3]);
sky_news_msg = [sky_news_msg; zeros(N-s1,1)];
quran_msg = [quran_msg; zeros(N-s2,1)];
russian_msg = [russian_msg; zeros(N-s3,1)];
clear s1 s2 s3;

% BaseBand Signals Fourier Transform
messages = [sky_news_msg, quran_msg, russian_msg];
clear sky_news_msg quran_msg russian_msg;
k = -N/2 :(N/2)-1;
figure('Name','Base-Band Signals','NumberTitle','off');
subplot(1,3,1);
plot(k*sampling_rate/N,abs(fftshift(fft(messages(:,1)))));
title('Sky News Message Spectrum');
xlabel('Frequency in Hz');
ylabel('FFT');
subplot(1,3,2);
plot(k*sampling_rate/N,abs(fftshift(fft(messages(:,2)))));
title('Quran Message Spectrum');
xlabel('Frequency in Hz');
ylabel('FFT');
subplot(1,3,3);
plot(k*sampling_rate/N,abs(fftshift(fft(messages(:,3)))));
title('Russian Voice Message Spectrum');
xlabel('Frequency in Hz');
ylabel('FFT');

% Upsampling The Audio Signals To Fit The Carriers Sampling Rates
messages_interp = [interp(messages(:,1),12), interp(messages(:,2),12)...
    ,interp(messages(:,3),12)];
clear messages;
fs = 12*sampling_rate;
num_samples = 12*N;

% Generating The Carriers For The Signals
t = 0:1/fs:16.76190476;
sky_news = cos(2*pi*100*1000*t)';
quran = cos(2*pi*150*1000*t)';
russian = cos(2*pi*200*1000*t)';
carriers = [sky_news quran russian];
clear sky_news quran russian ans;

% Perform The Modulation Process For Each Audio Signal
modulated_msgs = messages_interp .* carriers;
clear messages_interp carriers;

% Create The Frequency Division Multiplexed Signal By Addition Of The Modulated Signals
FDM = modulated_msgs(:,1) + modulated_msgs(:,2) + modulated_msgs(:,3);
clear modulated_msgs;

% Plot The Frequency Division Multiplexed Spectrum
```

```
k2 = -num_samples/2 : num_samples/2 - 1;
figure('Name','Frequency Division Multiplexed Signals','NumberTitle','off');
plot(k2*fs/num_samples,abs(fftshift(fft(FDM))));
title('Frequency Division Multiplexing Spectrum');
xlabel('Frequency in Hz');
ylabel('FFT');



%% The Receiver
% Choose The Required Audio Signals
disp (" ");
disp ("The Provided Channels: ");
disp ("1. Sky News On Carrier: 100 KHz");
disp ("2. Quran On Carrier: 150 KHz");
disp ("3. Russian Voice On Carrier: 200 KHz");
signal_req = input ("Please Select The Desired Channel Carrier in KHz: ");
signal_req = 1000 * signal_req;

% The RF Stage
% The BPF BandWidth Will Always BE 44 KHZ To Be Valid For Any RF Voice
F_stop1 = signal_req - 24000;
F_pass1 = signal_req - 22000;
F_pass2 = signal_req + 22000;
F_stop2 = signal_req + 24000;
A_stop1 = 80;
A_pass = 0.001;
A_stop2 = 80;
RF_BandPassSpecs = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',...
        F_stop1, F_pass1, F_pass2, F_stop2, A_stop1, A_pass,A_stop2, fs);
RF_BandPass_filter = design(RF_BandPassSpecs, 'equiripple');
RF_stage_out = filter(RF_BandPass_filter, FDM);

% Plot The RF Stage Filtered Output
figure('Name','RF Stage Filter Output','NumberTitle','off');
plot(k2*fs/num_samples,abs(fftshift(fft(RF_stage_out))));
title('RF Stage Filter Output');
xlabel('Frequency in Hz');
ylabel('FFT');

% IF Oscillator
IF_freq = 25000;
IF_carrier = cos(2*pi*t* (IF_freq + signal_req))';
IF_stage_input = RF_stage_out .* IF_carrier;

% Plot The IF Oscilltor Output
figure('Name','RF Mixer Output','NumberTitle','off');
plot(k2*fs/num_samples,abs(fftshift(fft(IF_stage_input))));
title('RF Mixer Output');
xlabel('Frequency in Hz');
ylabel('FFT');

% The BPF PassBand Will Always BE 44 KHZ To Be Valid For Any IF Voice
F_stop1 = IF_freq - 24000;
F_pass1 = IF_freq - 22000;
F_pass2 = IF_freq + 22000;
F_stop2 = IF_freq + 24000;
A_stop1 = 80;
A_pass = 0.001;
A_stop2 = 80;
IF_BandPassSpecs = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',...
        F_stop1, F_pass1, F_pass2, F_stop2, A_stop1, A_pass,A_stop2, fs);
IF_BandPass_filter = design(IF_BandPassSpecs, 'equiripple');
IF_stage_out = filter(IF_BandPass_filter, IF_stage_input);

% Plot The IF Stage Filtered Output
figure('Name','IF Stage Filter Output','NumberTitle','off');
plot(k2*fs/num_samples,abs(fftshift(fft(IF_stage_out))));
title('IF Stage Filter Output');
xlabel('Frequency in Hz');
ylabel('FFT');
```

```
% The BaseBand Detection
% Generate The Demodulation Oscillator With IF Frequency 25 KHz
baseband_carrier = cos(2*pi*IF_freq*t)';
baseband_signal = IF_stage_out .* baseband_carrier;

% Plot The BaseBand Stage Before Filtering
figure('Name','BaseBand Demodulation','NumberTitle','off');
plot(k2*fs/num_samples,abs(fftshift(fft(baseband_signal))));
title('BaseBand Returned Signal Before Filtering');
xlabel('Frequency in Hz');
ylabel('FFT');

% The LPF To Fully Demodulate The Signal
F_pass = 22000;
F_stop = 24000;
A_pass = 0.001;
A_stop = 80;
LowPassSpecs = fdesign.lowpass('Fp,Fst,Ap,Ast',...
        F_pass, F_stop, A_pass, A_stop, fs);
LowPass_filter = design(LowPassSpecs, 'equiripple');
demodulated_message = filter(LowPass_filter, baseband_signal);
demodulated_message = 4.5 * demodulated_message;
demodulated_message = downsample(demodulated_message, 12);
clear F_pass F_stop F_pass1 F_pass2 F_stop1 F_stop2 A_pass A_stop A_stop1 A_stop2;
clear IF_carrier baseband_carrier;

% Plot The Final BaseBand Demodulated Message
figure('Name','Final Demodulated Message','NumberTitle','off');
plot(k*sampling_rate/N, abs(fftshift(fft(demodulated_message))));
title('Final Demodulated Message');
xlabel('Frequency in Hz');
ylabel('FFT');

% Listen To The Demodulated Audio
sound (demodulated_message, sampling_rate);
```