Faculty of Enigeering

Cairo University

Communication Systems Engineering

Mixers and Modulators Research

Presented for ELC 3020

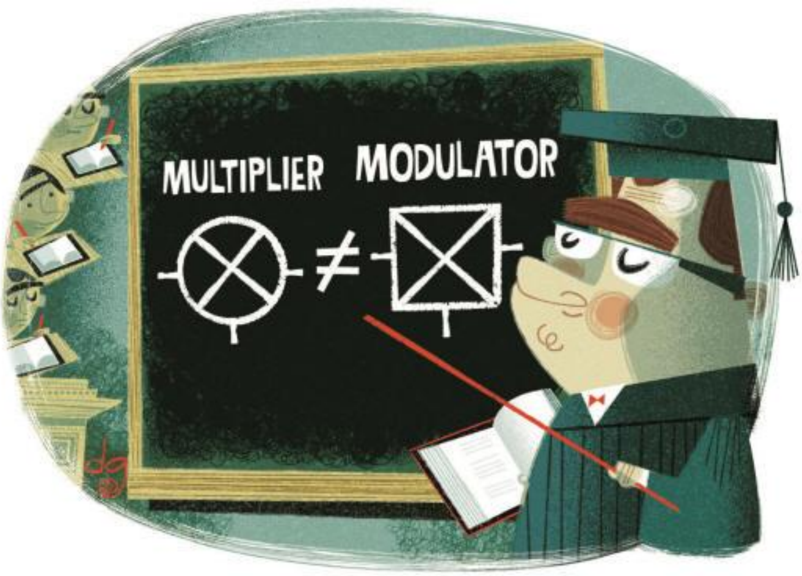
Presented to:

Dr. Ahmed Hesham

T.A: Mohamed Khalid

Email: ahesham.inquiries@gmail.com

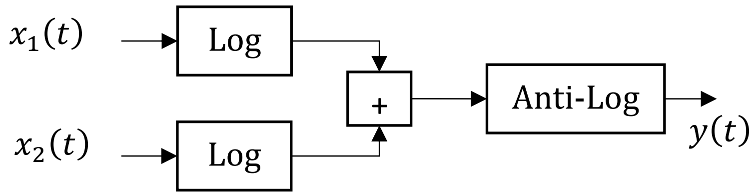
TEAM MEMBERS (3rd Year Electronics and Electrical Communication Engineers)	
مجدي أحمد عباس عبد الحميد الابرق	Sec: 3 / I.D: 9210899 / BN: 36
أحمد عادل يونس سيد أحمد	Sec: 1 / I.D: 9213073 / BN: 16



Types of AM Modulation

Product Modulator

Log Amplifier

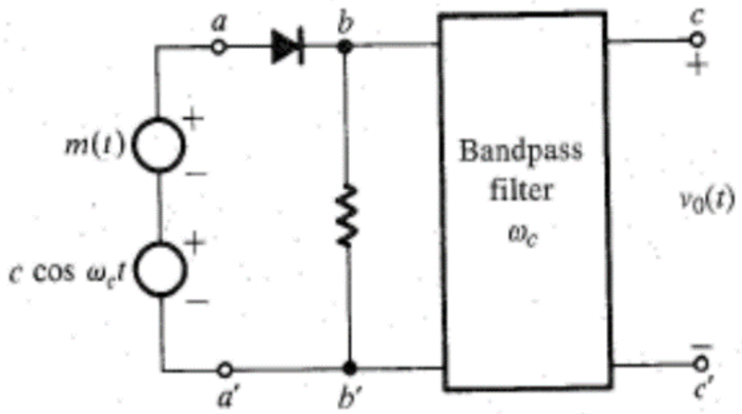


Electronic multipliers of basic design utilize logarithmic amplifiers for computation. They exploit the property that the antilogarithm of the sum of logarithms of two numbers equals the product of those numbers.
However, these multipliers have drawbacks, such as restricted bandwidth and operation confined to a single quadrant.

Switching Modulator

Diode

The switching modulator employing a diode involves the combination of a message signal and a carrier signal. This merged signal undergoes detection through a diode, serving as a rectifier to capture varying amplitude information. To prevent distortion, a resistor stabilizes the signal, and a bandpass filter selectively permits the desired frequency range.



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: مجدى أحمد عباس عبد الحميد

Sec: 3 / BN: 36 / ID: 9210899

Student Name: أحمد عادل يونس سيد أحمد

Sec: 1 / BN: 16 / ID: 9213073

Contents

Table of figures

1. *The transmitter*

This part contains the following tasks:

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

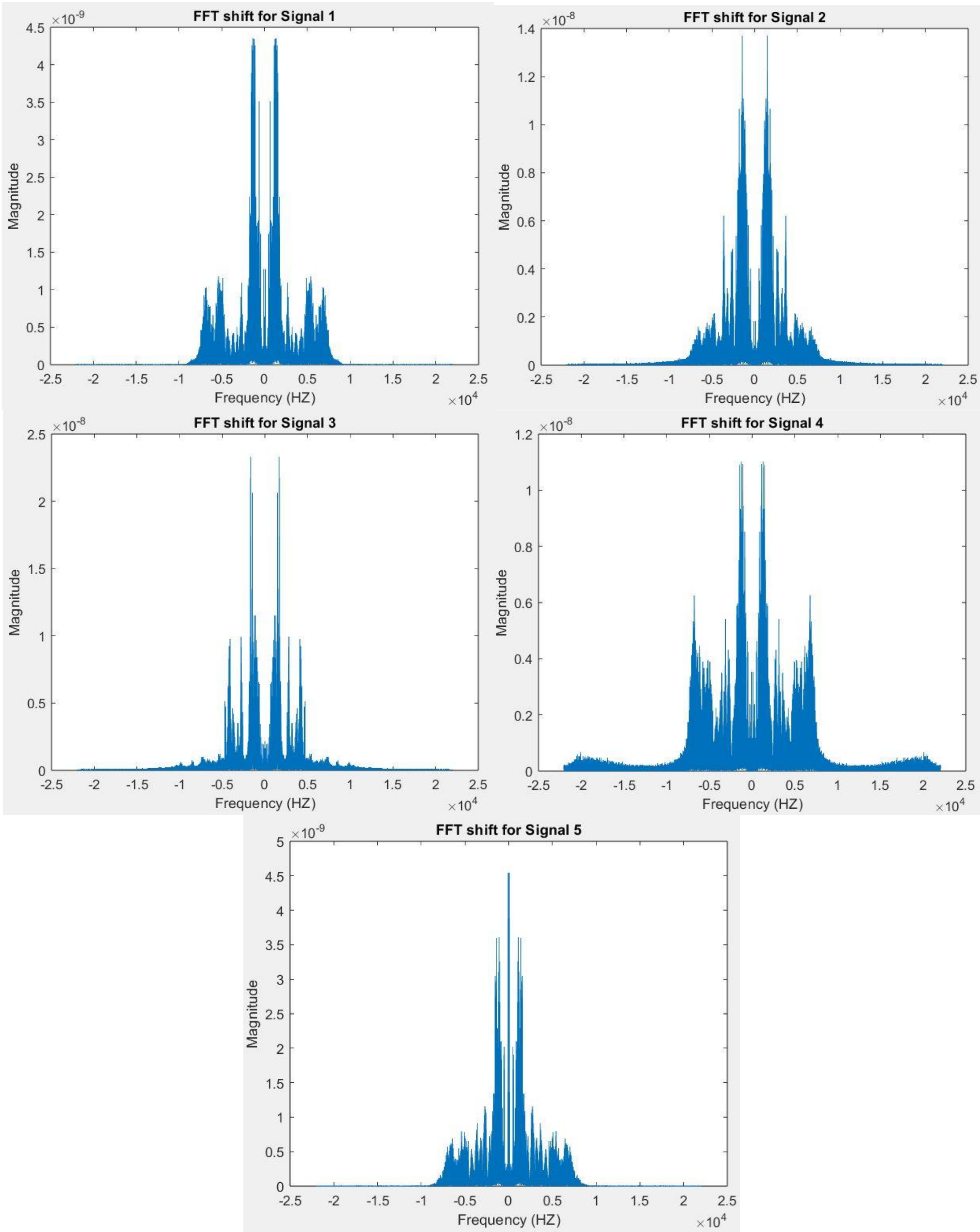
Discussion

After reading and sampling the five audio signals, they are converted into a monophonic stream to eliminate the need for two separate channels. The audio signals are then standardized to equal length through padding.
In the modulation stage, the objective is to shift the signals to a very high frequency compatible with the antenna size. This is achieved by increasing the sample rate 20 times, implementing specific carriers for each signal with a frequency difference (ΔF), and combining the signals using Frequency Division Multiplexing for transmission to the T_X Antenna.

The figures

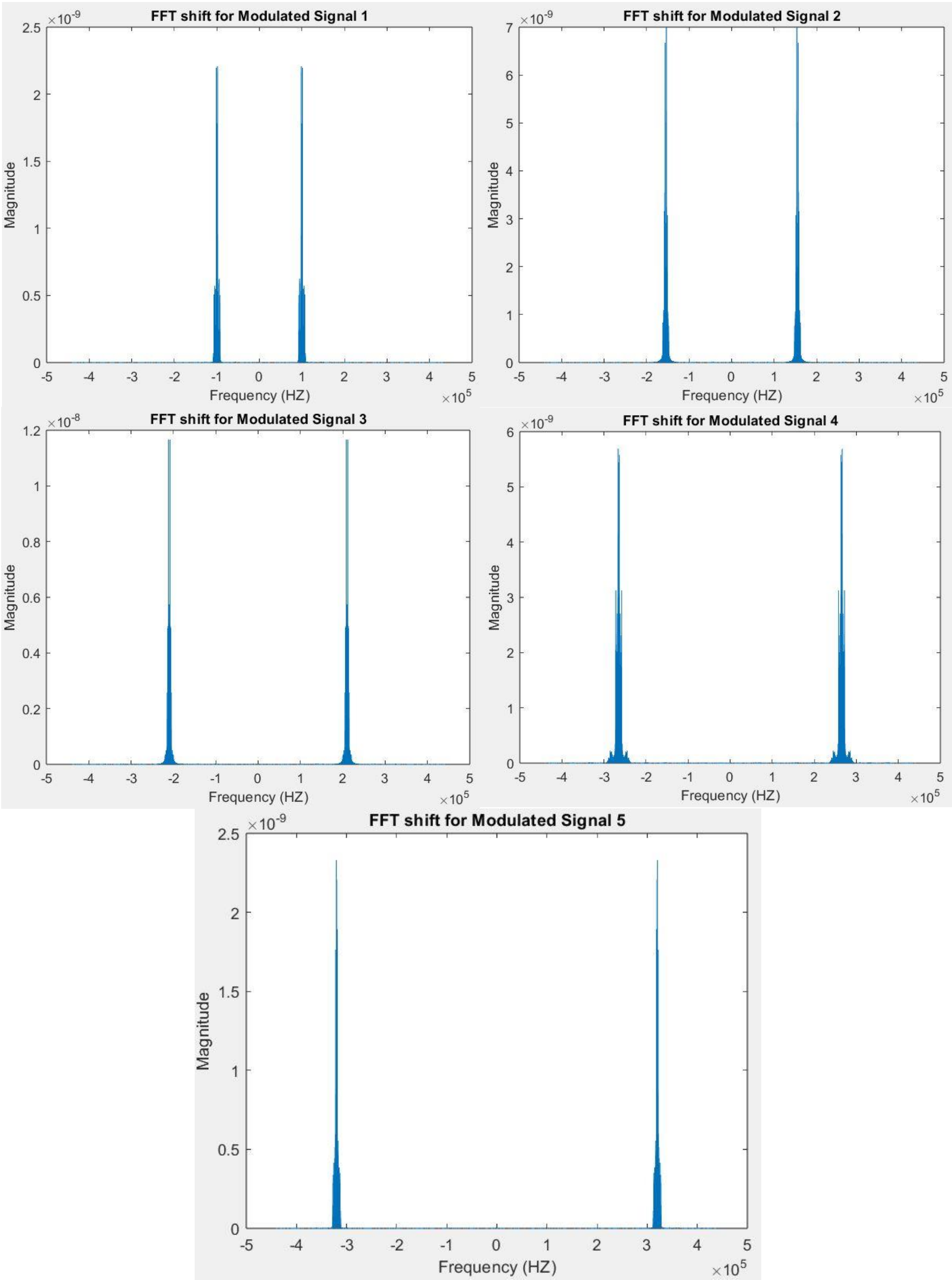
Figure 1: The spectrum of the output of the transmitter

Task 1: Reading monophonic audio signals into MATLAB

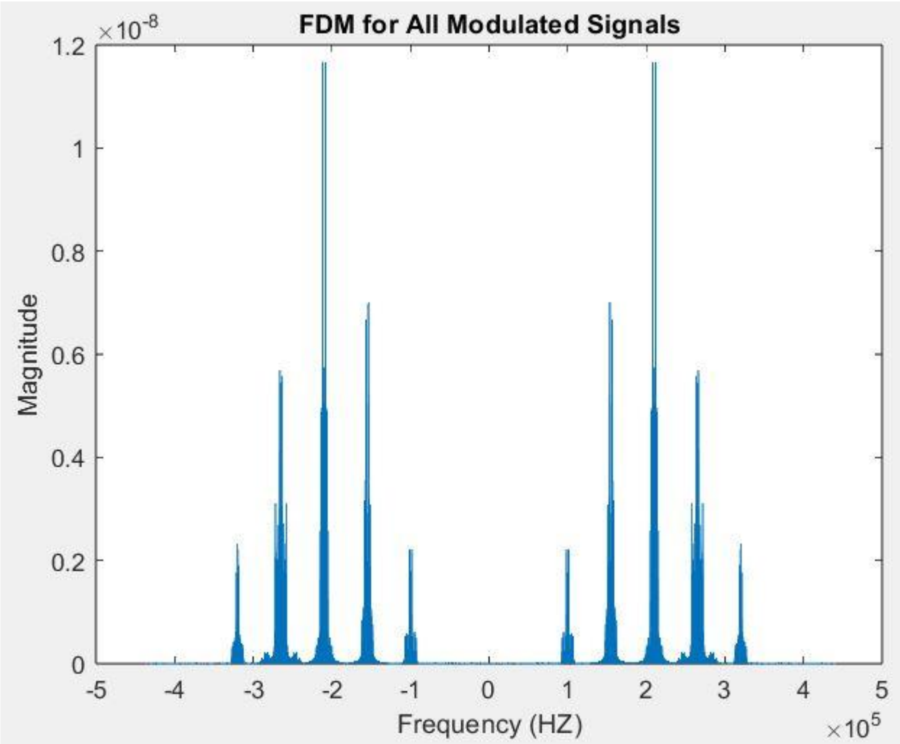


Task 2: Up sampling the audio signals.

Task 3: modulating the audio signals (each on a separate carrier).



Task 4: Addition of the modulated signals (FFT Shifted FDM)



2. The RF stage

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

Discussion

In this phase, a bandpass filter is employed to isolate the desired user signal and eliminate others that could lead to imaging at $(\omega_c + 2 \times \omega_{IF})$ Initially, the sample rate of the signals is increased by 40 times the original frequency. Following this, an oscillator with a carrier frequency $\omega_c + \omega_{IF} + \omega_{offset}$ is utilized. This carrier frequency is determined by $\omega_c = \omega_n + n \times \Delta F$, and it is multiplied by the audio signal using a new sample factor to avoid complications during the shift of the selected signal to the baseband.

The figures

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

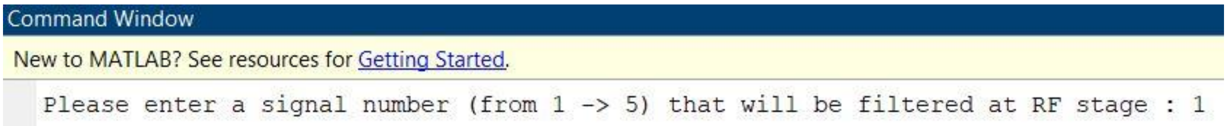


Figure 2: the output of the RF filter (before the mixer)

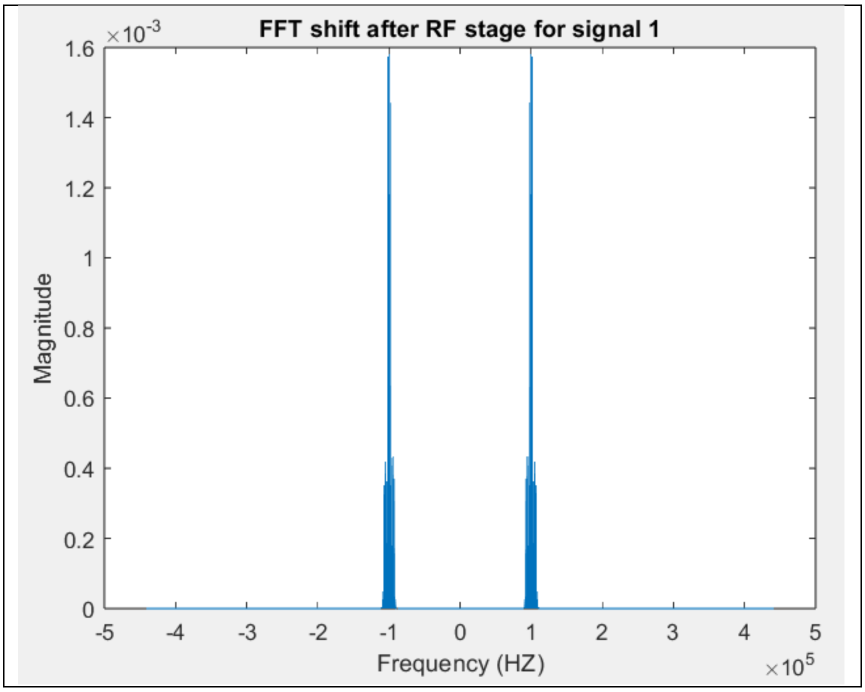
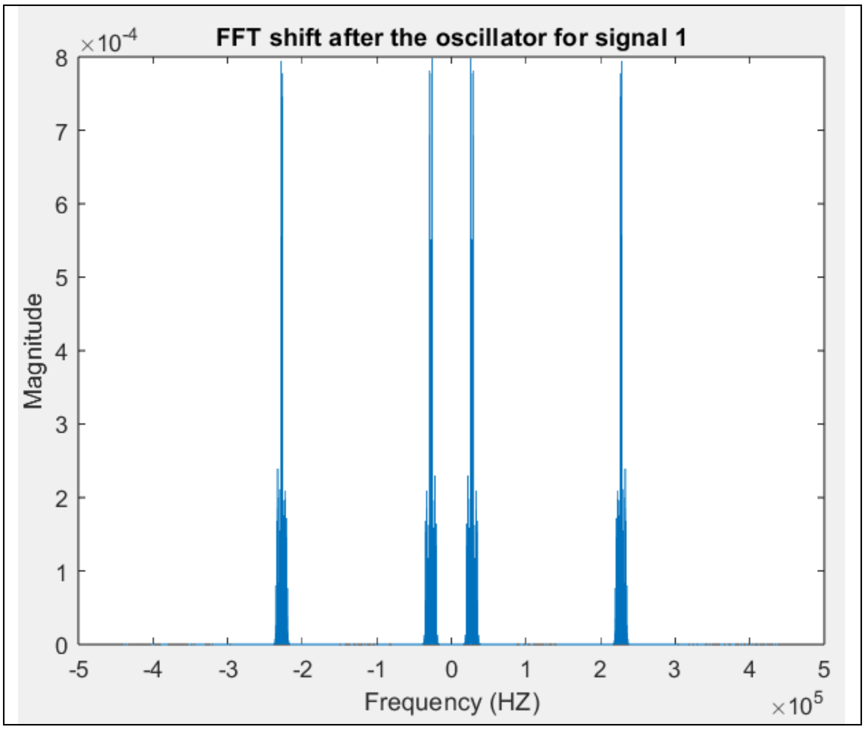


Figure 3: The output of the mixer



3. *The IF stage*

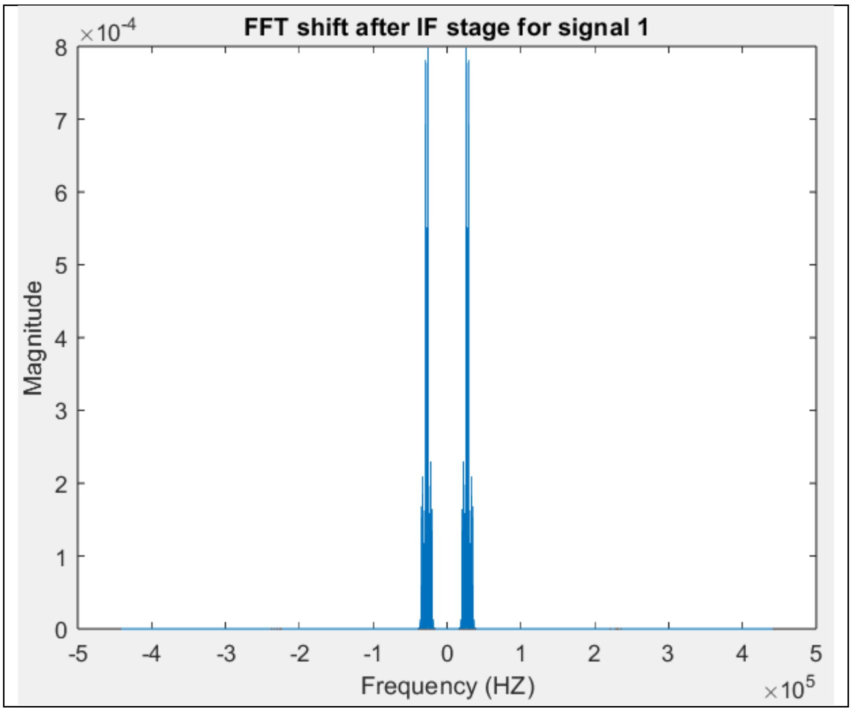
This part addresses the IF filter.

Discussion

The preceding signal has carrier at high frequency and at intermediate, necessitating a bandpass filter centered on "ω_{IF}" to eliminate the high-frequency component. This step is crucial to avoid issues during the direct demodulation of the message to the baseband. Baseband operations pose challenges such as local oscillator leakage, flicker noise, RF circuit linearity, and decreasing filter selectivity with frequency increase.
The previous signal has carrier at high frequency and at intermediate.

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

```
During this phase, the signal is shifted back to the baseband by multiplying
it with a carrier of "IF" frequency, followed by the use of a low-pass filter
to eliminate high frequencies. The audio signal is then heard after applying
a gain, calculated as the reciprocal of the multiplication of three mixers
(Oscillator, Amplitude modulator, and Demodulator at the Baseband stage) in
the original signal's path.
```

The figures

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

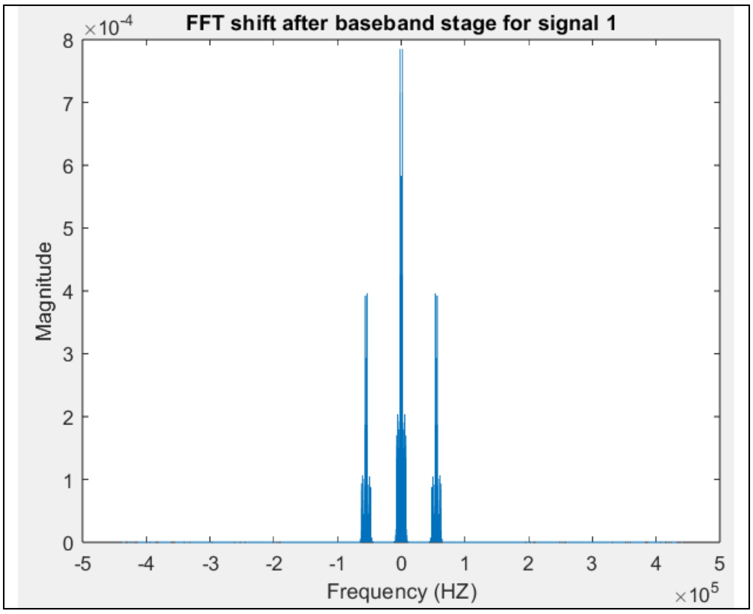
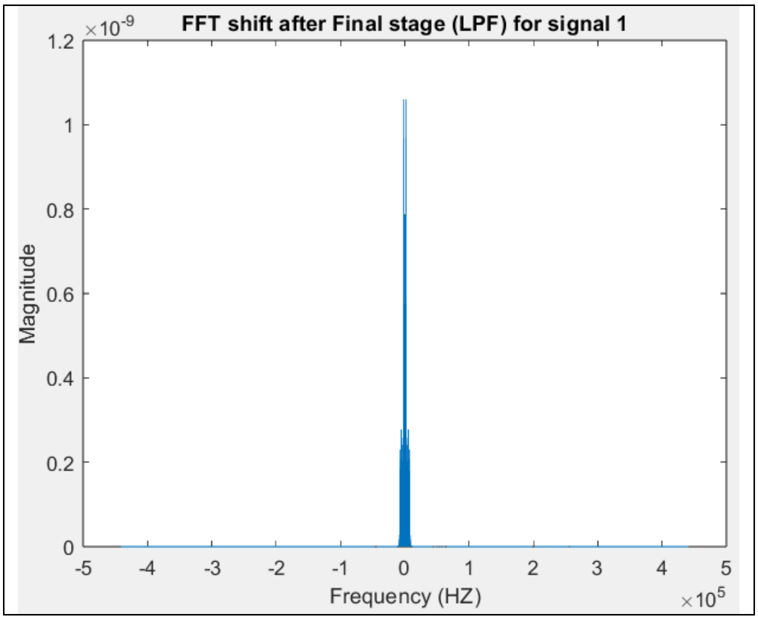


Figure 6: Output of the LPF



5. *Performance evaluation without the RF stage*

The figures

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

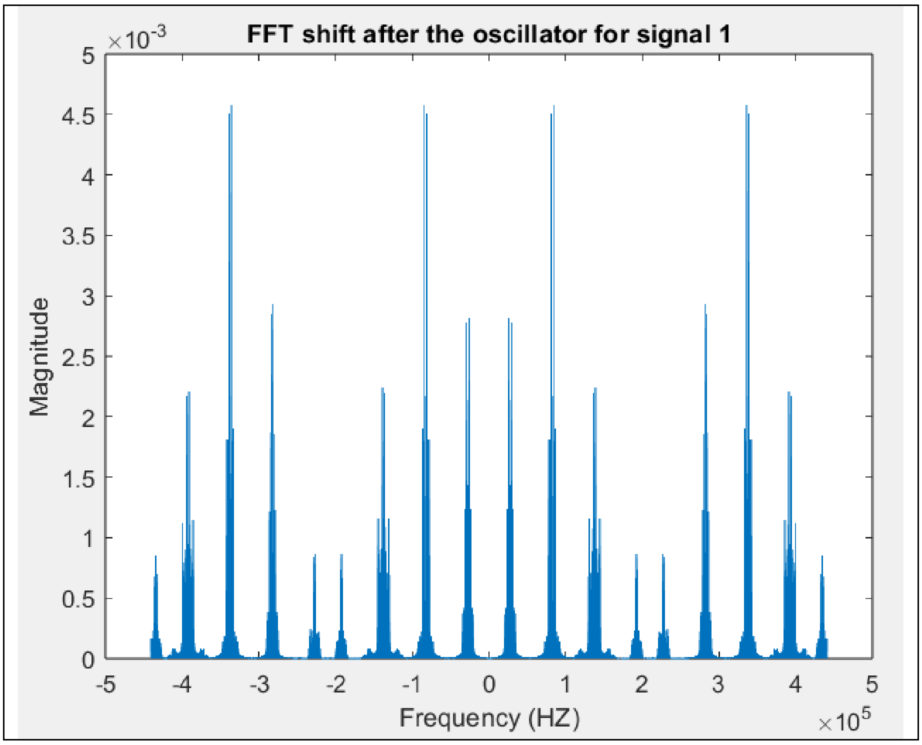


Figure 8: Output of the IF filter (no RF filter)

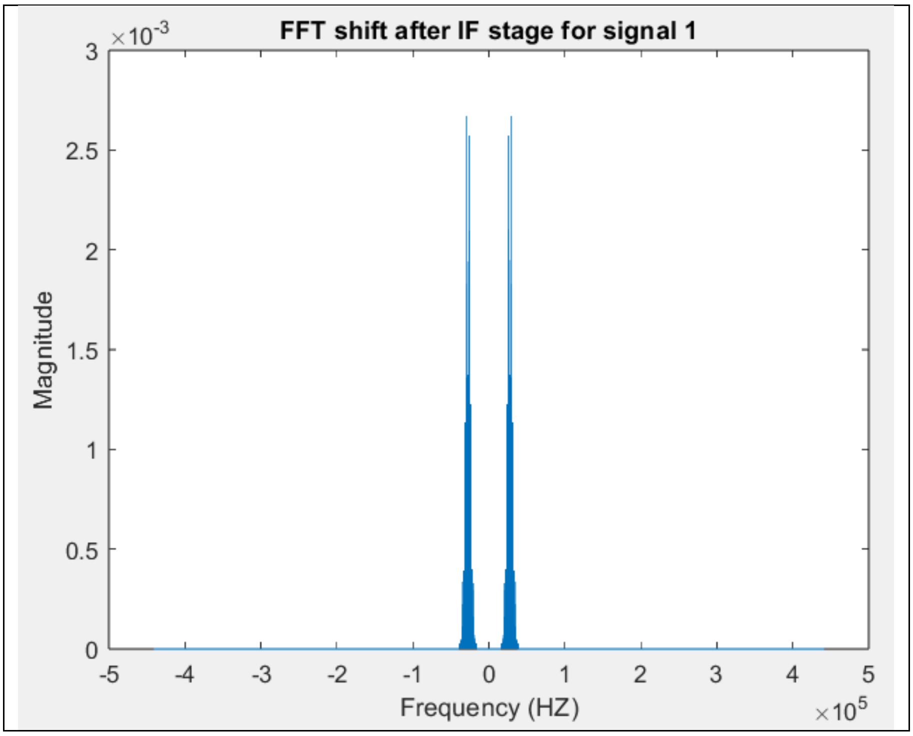


Figure 9: Output of the IF mixer before the LPF (no RF filter)

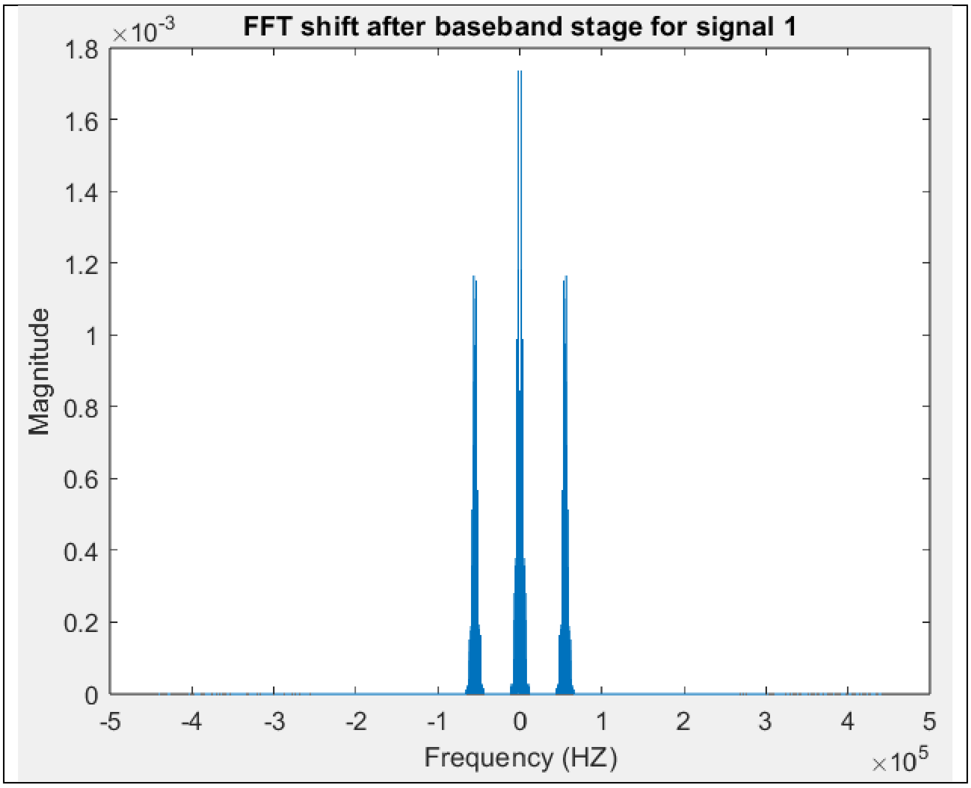
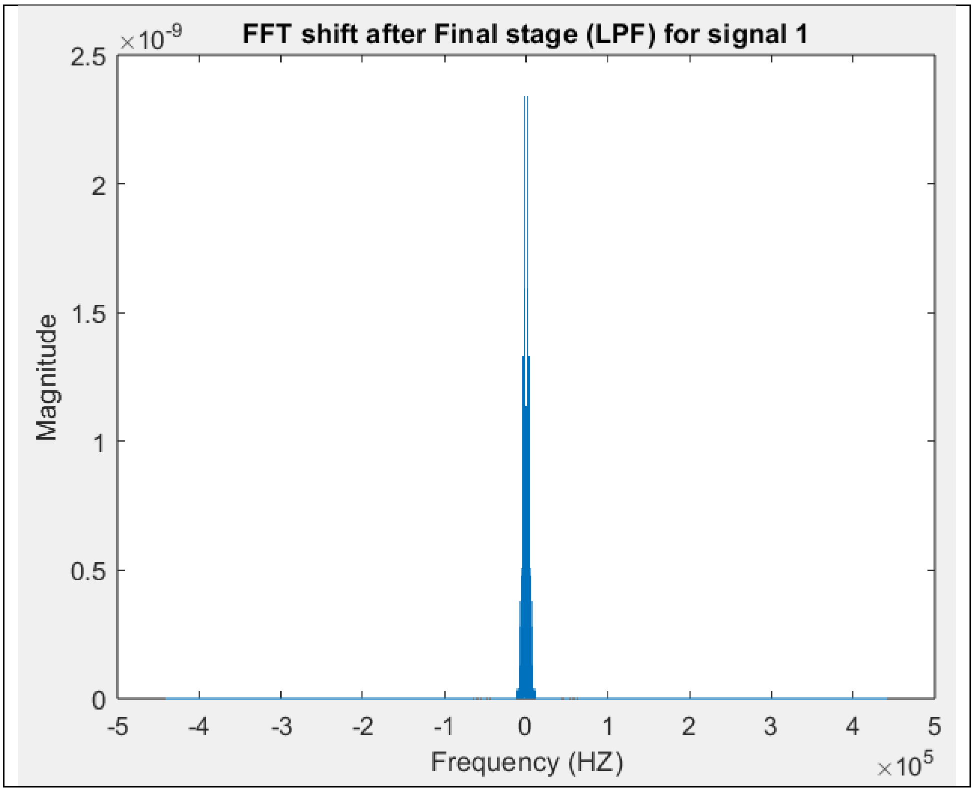


Figure 10: Output of the LPF (no RF filter)



6. Comment on the output sound

The absence of an RF filter leads to image problems when removing the RF stage, which is essential for eliminating the image signal interference. This interference occurs when the received audio interferes with other audio signals, with carrier frequencies equal to $\omega_c + 2\,\omega_{IF}$. Before removing the RF stage, a bandpass filter is utilized to isolate the desired signal and reject both unwanted signals and their images.

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

A frequency offset f_{offset} was introduced to the receiver oscillator, resulting in $\omega_{osc} = \omega_c + \omega_{IF} + \omega_{offset}$. From a spectral perspective, beyond the RF mixer stage, the sidebands of the frequency spectrum were uniformly shifted by the same offset value. Consequently, at the baseband, the interference among the sidebands of a signal led to self-distortion. Regarding audio quality, with $f_{offset} = 0.1\,kHz$ in the first scenario, the output sound displayed a high-pitched tone, yet the information remained distinguishable. Conversely, in the second scenario with $f_{offset} = 1\,kHz$, the output sound became completely distorted and inaudible.

"The more offset, the more distortion increases, and the audio quality gets worse".

7. The code

```
clc
clear
close all;

%% Audio Signals (5 Messages)

% Reading Audio Signals
[message1_BBCArabic2,Fs]     = audioread('Short_BBCArabic2.wav');
                                          % 17 Seconds , Length = 740544
[message2_FM9090,~]          = audioread('Short_FM9090.wav');
                                          % 16 Seconds , Length = 697536
[message3_QuranPalestine,~] = audioread('Short_QuranPalestine.wav');
                                          % 17 Seconds , Length = 739200
[message4_RussianVoice,~]    = audioread('Short_RussianVoice.wav');
                                          % 16 Seconds , Length = 703360
[message5_SkyNewsArabia,~]  = audioread('Short_SkyNewsArabia.wav');
                                          % 17 Seconds , Length = 711872

% Max. Length for All Signals = 740544
Length = length(message1_BBCArabic2);

% Monophonic Receiver Implementation (Single Channel for each Signal)
mono_message1_BBCArabic2     = message1_BBCArabic2(:,1)      +
                                                message1_BBCArabic2(:,2);
mono_message2_FM9090         = message2_FM9090(:,1)          +
                                                    message2_FM9090(:,2);
mono_message3_QuranPalestine = message3_QuranPalestine(:,1) +
                                            message3_QuranPalestine(:,2);
mono_message4_RussianVoice   = message4_RussianVoice(:,1)    +
                                              message4_RussianVoice(:,2);
mono_message5_SkyNewsArabia  = message5_SkyNewsArabia(:,1)   +
                                             message5_SkyNewsArabia(:,2);

% Signals Padding with Zeros so they have all Equal Length
audios_signals              = zeros(Length,5);

message1_BBCArabic2_PAD      = [mono_message1_BBCArabic2;
            zeros(Length-length(mono_message1_BBCArabic2),1)];
message2_FM9090_PAD         = [mono_message2_FM9090;
              zeros(Length-length(mono_message2_FM9090),1)];
message3_QuranPalestine_PAD = [mono_message3_QuranPalestine;
          zeros(Length-length(mono_message3_QuranPalestine),1)];
message4_RussianVoice_PAD    = [mono_message4_RussianVoice;
           zeros(Length-length(mono_message4_RussianVoice),1)];
message5_SkyNewsArabia_PAD  = [mono_message5_SkyNewsArabia;
          zeros(Length-length(mono_message5_SkyNewsArabia),1)];

% Filling The Audios Signal Array with the Padded Messages
audio_signals(:,1) = message1_BBCArabic2_PAD;
audio_signals(:,2) = message2_FM9090_PAD;
audio_signals(:,3) = message3_QuranPalestine_PAD;
audio_signals(:,4) = message4_RussianVoice_PAD;
audio_signals(:,5) = message5_SkyNewsArabia_PAD;
```

```
%% Plot The Audio Signals In Frequency Domain

Freq_range = (-Length/2:Length/2-1)*Fs/Length;

audio_signals_fft = zeros(Length,5);

for n = 1 : 5
    audio_signals_fft(:,n) = abs(fft(audio_signals(:,n))/Length);
end

% Plotting The Shifted Version for FFT of All Audio Signals
for n = 1 : 5
    figure
    plot(Freq_range,fftshift(audio_signals_fft(:,n))/Length);
    title("FFT shift for Signal " + n);
    xlabel('Frequency (HZ)');
    ylabel('Magnitude');
end

%% AM Modulator Stage

% Increase Number of Samples to avoid Nyquist Criteria (Aliasing)
audio_signals_interp = zeros(Length*20,5);
for n = 1 : 5
    audio_signals_interp(:,n) = interp(audio_signals(:,n),20);
% Where N = 20 Represent The New Sample Factor
end

% Audio Signals Carriers
Fc = 100000;
Delta_F = 55000;
Ts = 1/Fs;
Ts_New = (1/20)*Ts;
T = 0:Ts_New:(20*Length-1)*Ts_New;

audio_signals_carriers = zeros(Length*20,5);
for n = 0 : 4
    audio_signals_carriers(:,n+1) = (cos(2*pi*(Fc+n*Delta_F)*T))';
end

% Modulated Signals
modulated_audio_signals = zeros(Length*20,5);
for n = 1 : 5
    modulated_audio_signals(:,n) =
                  audio_signals_carriers(:,n).*audio_signals_interp(:,n);
end

%% Plot The Modulated Audio Signals In Frequency Domain

New_Freq_range = (-20*Length/2:20*Length/2-1)*Fs/Length;

modulated_audio_signals_fft = zeros(Length*20,5);
for n = 1 : 5
    modulated_audio_signals_fft(:,n) =
                      abs(fft(modulated_audio_signals(:,n))/(20*Length));
end
```

```
% Plotting The Shifted Version for FFT of All Audio Signals
for n = 1 : 5
    figure

plot(New_Freq_range,fftshift(modulated_audio_signals_fft(:,n))/Length);
    title("FFT shift for Modulated Signal " + n);
    xlabel('Frequency (HZ)');
    ylabel('Magnitude');
end

%% Frequency Division Multiplexing (FDM)

FDM_audio_signals = zeros(Length*20,1);

for n = 1 : 5
    FDM_audio_signals = FDM_audio_signals + modulated_audio_signals(:,n);
end

FDM_audio_signals_fft = abs(fft(FDM_audio_signals)/(20*Length));

% FDM Plotting
figure
plot(New_Freq_range,fftshift(FDM_audio_signals_fft)/Length);
title('FDM for All Modulated Signals');
xlabel('Frequency (HZ)');
ylabel('Magnitude');

%% Bandwidth Calculation From Audio Signals Figures (After Padding)

BB_BW_audio_signals = 22050;

%% The RF Stage

% infinite loop to force the user to enter a correct value from 1 --> 5
while (1)
  signal_number = input('Please enter a signal number (from 1 -> 5) that
will be filtered at RF stage : ');
    if(signal_number<1 || signal_number>5)
        disp( 'wrong input !! , please try again' );
    else
        break;
    end
end
signal_number = signal_number-1;
% signal_number will be vary from 0 --> 4 to be used directly in the fc
expression

% RF_filtered_audio_signal = zeros(Length*20,1);
fstop1  = (Fc+signal_number*Delta_F) - BB_BW_audio_signals/2 - 1000;
% Margin = 1kHz as filter is Not Ideal
fpass1  = (Fc+signal_number*Delta_F) - BB_BW_audio_signals/2;
fpass2  = (Fc+signal_number*Delta_F) + BB_BW_audio_signals/2;
fstop2  = (Fc+signal_number*Delta_F) + BB_BW_audio_signals/2 + 1000;
% Margin = 1kHz as filter is Not Ideal
BPF_OBJ1 = Bandpass_Filter_1(fstop1,fpass1,fpass2,fstop2);
% create instance from Band pass filter function
RF_filtered_audio_signal = filter(BPF_OBJ1, FDM_audio_signals);

%% Plot The Filtered Audio Signal after the RF Stage

RF_filtered_audio_signal_fft = abs(fft(RF_filtered_audio_signal));
```

```
% Plotting The Shifted Version for FFT of selected signal at RF stage
figure
plot(New_Freq_range,fftshift(RF_filtered_audio_signal_fft)/(20*Length));
title("FFT shift after RF stage for signal " + (signal_number + 1));
xlabel('Frequency (HZ)');
ylabel('Magnitude');

%% The Oscillator Stage

F_IF = 27500;          % The IF Frequency
Ts_IF = Ts_New;
T_IF = 0:Ts_IF:(20*Length-1)*Ts_IF;
F_offset = 0;

osc_audio_signal = RF_filtered_audio_signal .*
            ((cos(2*pi*(Fc+signal_number*Delta_F+F_IF+F_offset)*T_IF))');

%% Plot The Filtered Audio Signals after Oscillator

IF_Freq_range = (-20*Length/2:(20*Length/2)-1)*Fs/Length;
osc_audio_signal_fft = abs(fft(osc_audio_signal));

% Plotting FFT of selected Audio Signal after the oscillator
figure
plot(IF_Freq_range,fftshift(osc_audio_signal_fft)/(20*Length));
title("FFT shift after the oscillator for signal " +(signal_number + 1));
xlabel('Frequency (HZ)');
ylabel('Magnitude');

%% The IF Stage

fstop1   = F_IF - BB_BW_audio_signals/2 - 1000;
% Margin = 1khz as filter is Not Ideal
fpass1   = F_IF - BB_BW_audio_signals/2;
fpass2   = F_IF + BB_BW_audio_signals/2;
fstop2   = F_IF + BB_BW_audio_signals/2 + 1000;
% Margin = 1khz as filter is Not Ideal
BPF_OBJ2 = Bandpass_Filter_2(fstop1,fpass1,fpass2,fstop2);
% create instance from Band pass filter function
IF_filtered_audio_signal = filter(BPF_OBJ2, osc_audio_signal);

%% Plot The Filtered Audio Signal after the IF Stage

IF_filtered_audio_signal_fft = abs(fft(IF_filtered_audio_signal));

% Plotting The Shifted Version for FFT of selected signal at IF stage
figure
plot(IF_Freq_range,fftshift(IF_filtered_audio_signal_fft)/(20*Length));
title("FFT shift after IF stage for signal " + (signal_number + 1));
xlabel('Frequency (HZ)');
ylabel('Magnitude');

%% Baseband detection stage

base_band_audio_signal = IF_filtered_audio_signal .*
                                                    ((cos(2*pi*F_IF*T))');
base_band_audio_signal_fft = abs(fft(base_band_audio_signal));
```

```
% Plotting The Shifted Version for FFT of selected signal at baseband
stage
figure
plot(IF_Freq_range,fftshift(base_band_audio_signal_fft)/(20*Length));
title("FFT shift after baseband stage for signal " + (signal_number+1));
xlabel('Frequency (HZ)');
ylabel('Magnitude');

%% Low pass filter stage
fpass = BB_BW_audio_signals;
fstop = BB_BW_audio_signals + 1000;
% where the 1000 Hz is a margin value
LPF_OBJ = lowpass_filter(fpass,fstop);
% create instance from low pass filter function
Output_signal = filter(LPF_OBJ, base_band_audio_signal);

%% Plot The Filtered Audio Signal at the baseband Stage
(after low pass filter)

Output_signal_fft = abs(fft(Output_signal))/(20*Length);

% Plotting The Shifted Version for FFT of selected signal at IF stage
figure
plot(IF_Freq_range,fftshift(Output_signal_fft)/Length);
title("FFT shift after Final stage (LPF) for signal "+(signal_number+1));
xlabel('Frequency (HZ)');
ylabel('Magnitude');

%% Test the output signal sound using sound function

Output_signal = 8 .* Output_signal;
% where gain = 8 as we have three mixers on the path of the original
signal each path decrease the amplitude by 1/2
Output_signal = decimate(Output_signal,20);
% the decimate function is used to downsample a signal by a factor of L
"in this case L = 20"
audiowrite('filtered_audio_signal_1.wav',Output_signal,Fs);
sound(Output_signal,Fs);

%% 1st Band Pass Filter Function (RF Stage)

function Hd = Bandpass_Filter_1(fstop1,fpass1,fpass2,fstop2)
% BANDPASS_FILTER_1 Returns a discrete-time filter object.
% MATLAB Code
% Generated by MATLAB(R) 9.10 and Signal Processing Toolbox 8.6.
% Chebyshev Type II Bandpass filter designed using FDESIGN.BANDPASS.
% All frequency values are in Hz.
Fs = 882000;          % Sampling Frequency = 20 * Fs = 20 * 44100 =
882000 Hz

Fstop1 = fstop1;      % First Stopband Frequency
Fpass1 = fpass1;      % First Passband Frequency
Fpass2 = fpass2;      % Second Passband Frequency
Fstop2 = fstop2;      % Second Stopband Frequency
Astop1 = 100;         % First Stopband Attenuation (dB)
Apass  = 1;           % Passband Ripple (dB)
Astop2 = 100;         % Second Stopband Attenuation (dB)

% Construct an FDESIGN object and call its CHEBY2 method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'cheby1');
end
```

```
%% 2nd Band Pass Filter Function (IF Stage)

function Hd = Bandpass_Filter_2(fstop1,fpass1,fpass2,fstop2)
% BANDPASS_FILTER_2 Returns a discrete-time filter object.
% MATLAB Code
% Generated by MATLAB(R) 9.10 and Signal Processing Toolbox 8.6.
% Chebyshev Type II Bandpass filter designed using FDESIGN.BANDPASS.
% All frequency values are in Hz.
Fs = 882000;
% Sampling Frequency = 20 * Fs = 20 * 44100 = 882000 Hz

Fstop1 = fstop1;       % First Stopband Frequency
Fpass1 = fpass1;       % First Passband Frequency
Fpass2 = fpass2;       % Second Passband Frequency
Fstop2 = fstop2;       % Second Stopband Frequency
Astop1 = 100;          % First Stopband Attenuation (dB)
Apass  = 1;            % Passband Ripple (dB)
Astop2 = 100;          % Second Stopband Attenuation (dB)

% Construct an FDESIGN object and call its CHEBY2 method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'cheby2');
end

%% Low pass filter function (baseband stage)

function Hd = lowpass_filter(fpass,fstop)
%LOWPASS_FILTER Returns a discrete-time filter object.
% MATLAB Code
% Generated by MATLAB(R) 9.10 and Signal Processing Toolbox 8.6.
% Chebyshev Type II Lowpass filter designed using FDESIGN.LOWPASS.

% All frequency values are in Hz.
Fs =  882000;         % Sampling Frequency =  20 * 44100

Fpass = fpass;        % Passband Frequency
Fstop = fstop;        % Stopband Frequency
Apass = 1;            % Passband Ripple (dB)
Astop = 100;          % Stopband Attenuation (dB)

% Construct an FDESIGN object and call its CHEBY2 method.
h  = fdesign.lowpass(Fpass, Fstop, Apass, Astop, Fs);
Hd = design(h, 'cheby2');
end
```