Faculty of engineering

Electrical Communication and Electronics department

Cairo university

ELC 3020

Analog Communication

AM modulator and a super-heterodyne receiver

Name: محمود سعيد غمري مصطفي

SEC: 3

ID:9231950

BN:37

Table of Contents

Table of Figures

Blocks Description

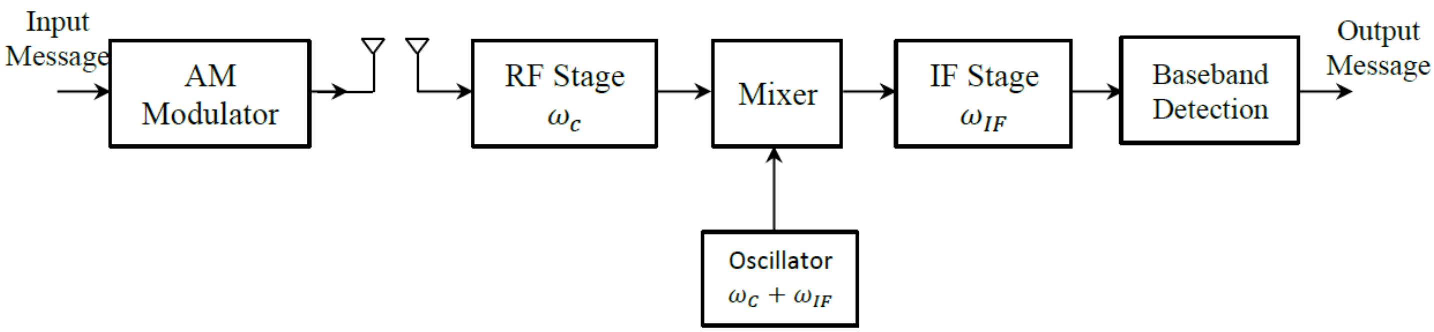

Figure 1: AM modulator and a super-heterodyne receiver

This design is used to transmit and receive signals over long distances through a channel. Let's describe each block.

AM Modulator:

This block takes the input message signal (baseband signal) and modulates it onto a higher frequency carrier signal ω_c using amplitude modulation (AM). It also performs frequency division multiplexing (FDM) to transmit 5 signals.

1- I take all the input the signal and his Fs (Sampling Frequency) and compare Fs for all signals and take the max one and resample all signal for this Fs.
2- Convert stereo to mono.
3- Padding the signal compare with the highest one.
4- Increase sampling frequency by 15 times and update it with the five signals.

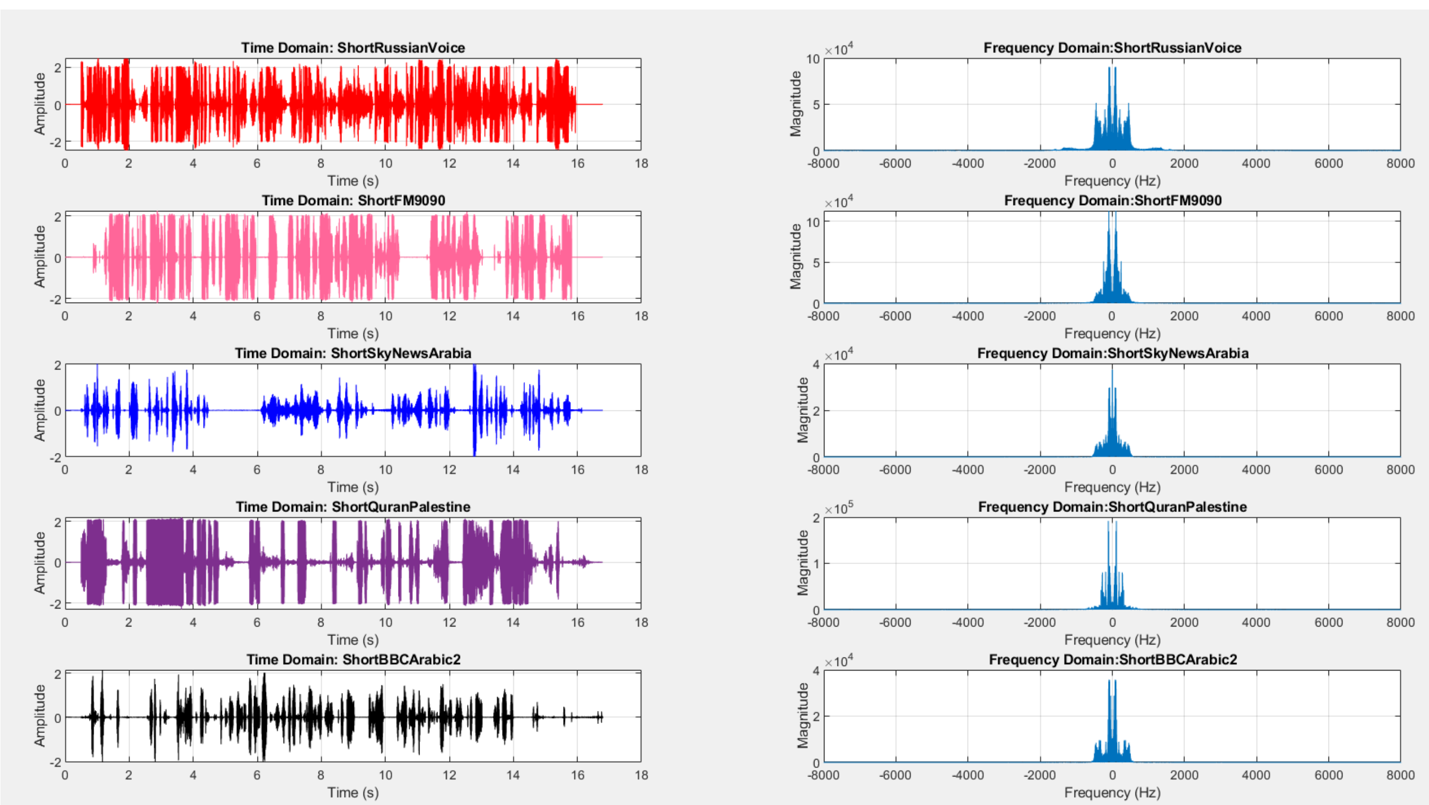

Figure 2: input signals in time and frequency domain

5- Now we make the modulation by multiplying with carrier frequency Fc = 100 KHz and the its bandwidth = 50 KHz so signal 1 at 100 KHz, signal 2 at 150 KHz, signal 3 at 200 KHz, signal 4 at 250 KHz and signal 5 at 300 KHz.

6- Sum the 5 modulated signals it is the FDM.

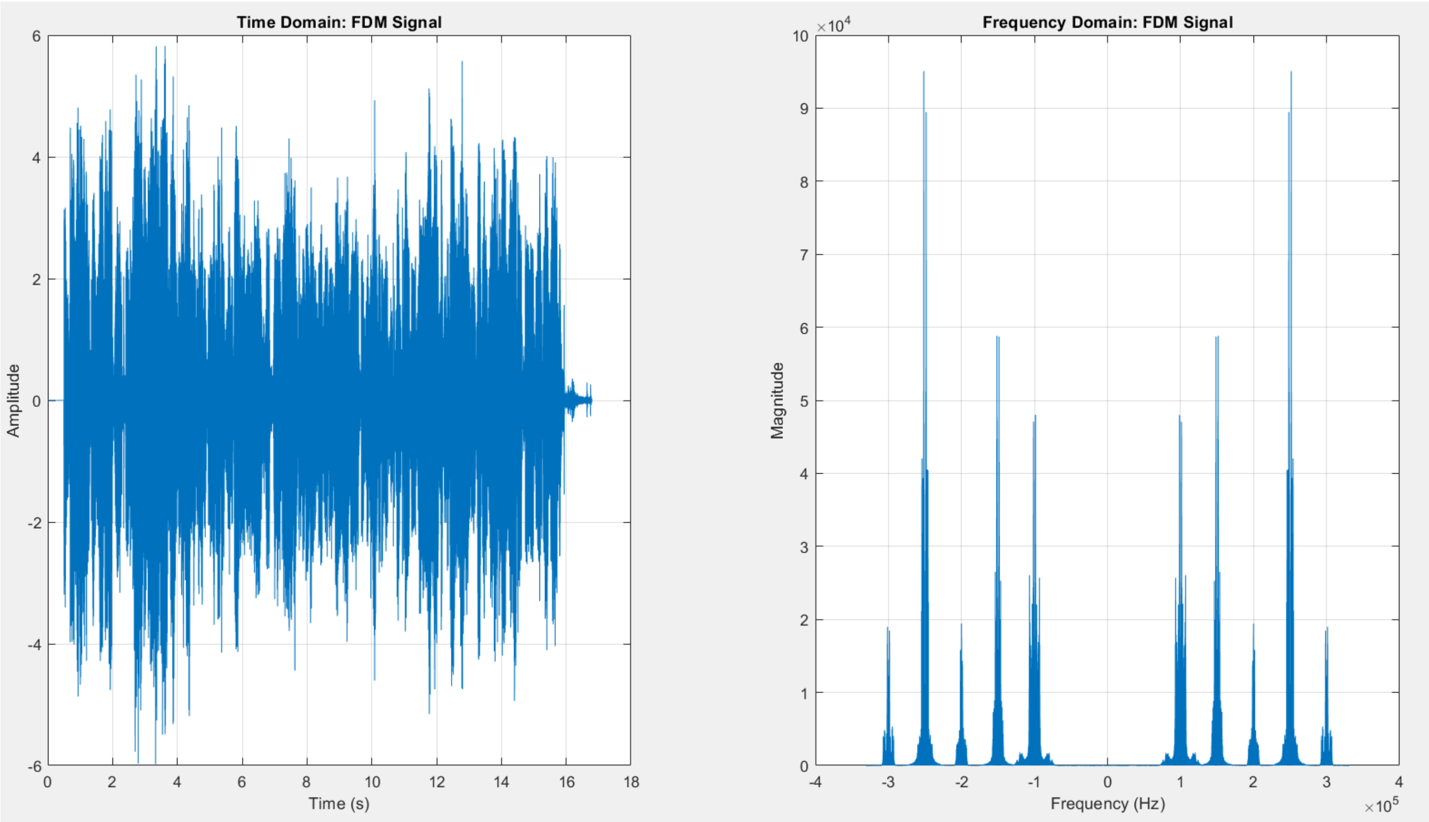

Figure 3: FDM

Channel:

It can be a cable or space but, in this project, we do not design it.

RF Stage:

The RF stage acts like a Band Pass Filter (BPF), selecting the desired signal while rejecting unwanted signals as I can and noise.

I make the BPF with lower passband at 75 KHz and upper passband 125 KHz to select the first signal.

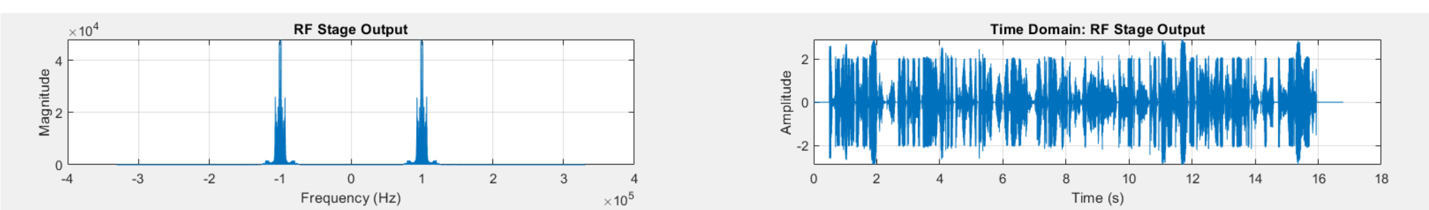

Figure 4: RF stage Output

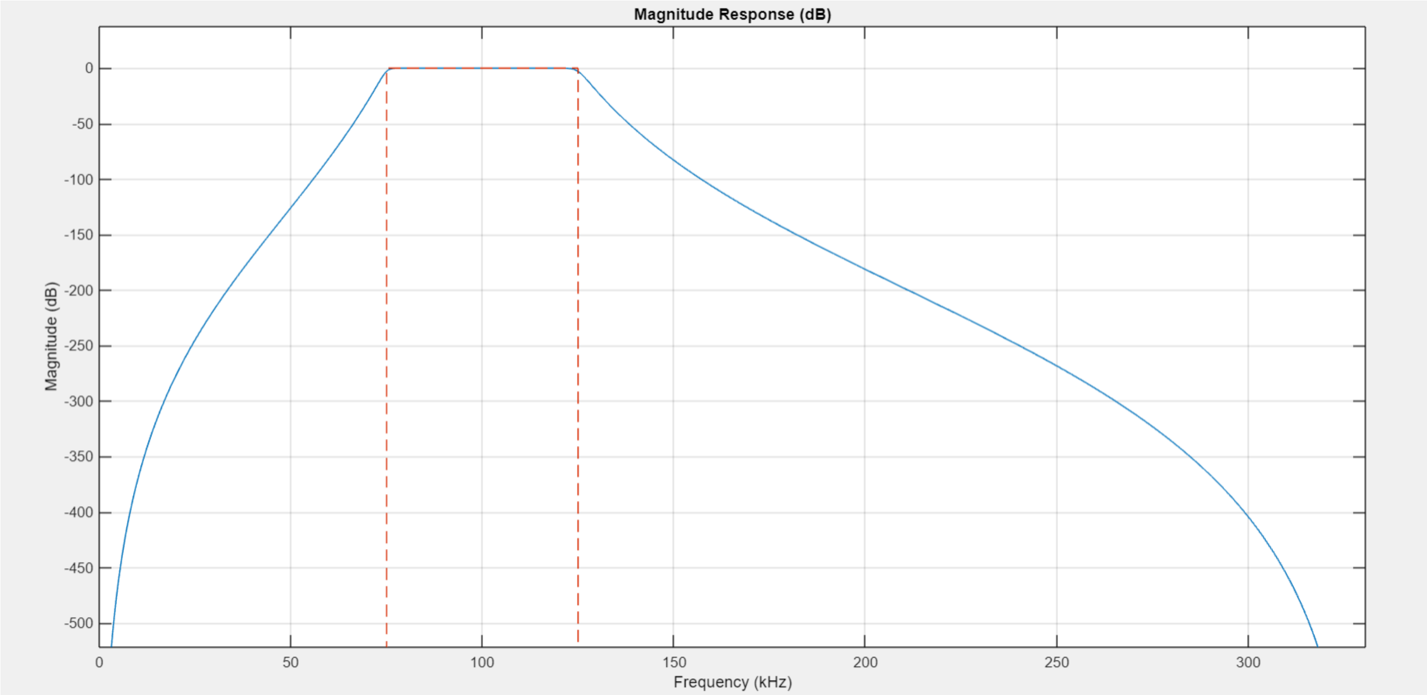

Figure 5: RF Bandpass filter

Mixer

We mixed the output of RF stage with ω_{if}, we prevent the Flicker Noise, Quadrature Error Offsets and DC offset as we can. $f_{if} = 25\ KHz$ so, the f_{lo} = 125 KHz now its at 225 KHz and 25 KHz

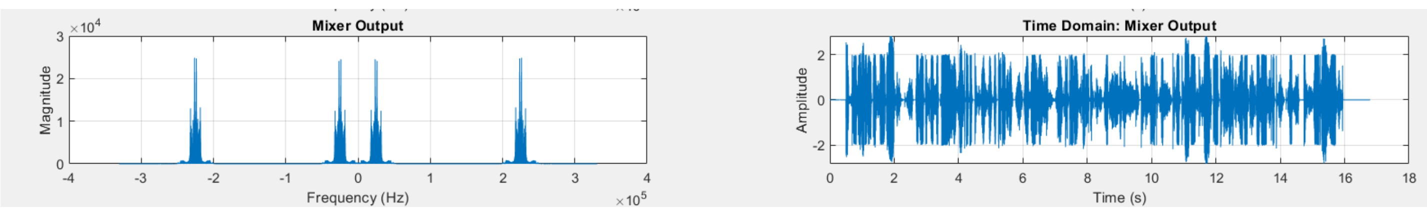

Figure 6: Mixer output

IF Stage

It is band-pass filter to select the signal and filter the dc the lower frequency 2950 Hz and upper frequency 47050 Hz

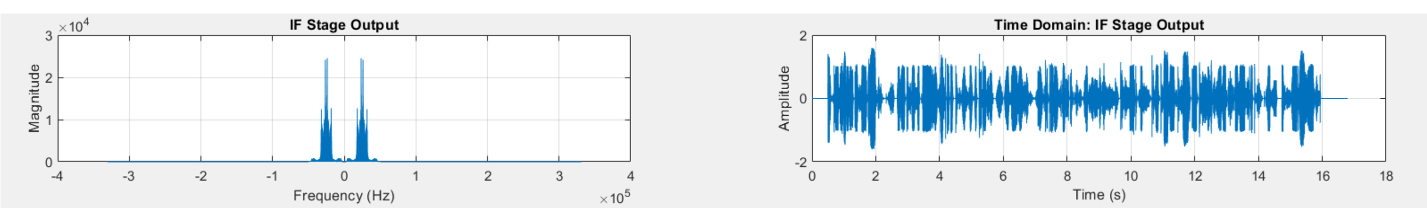

Figure 7: IF Stage output

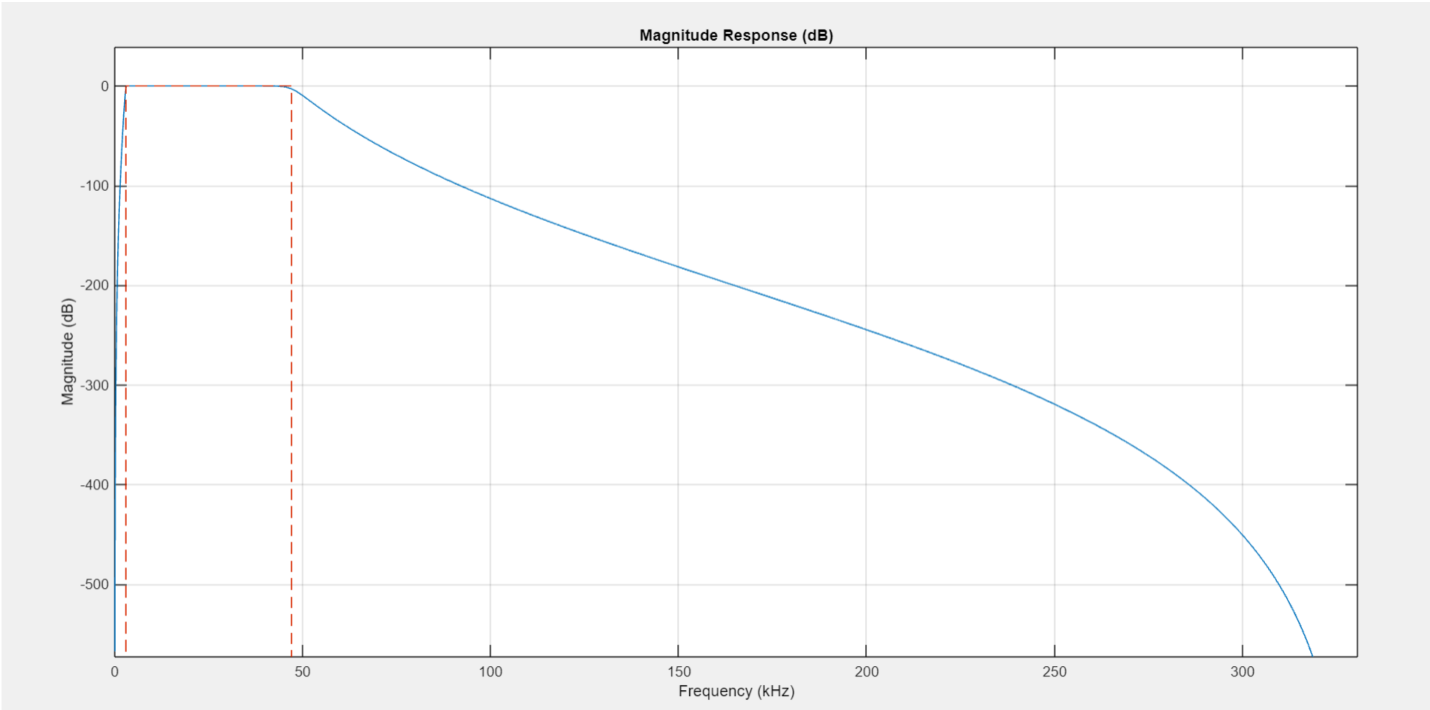

Figure 8: IF Bandpass filter

Base band Detection:

we demodulate the signal to the base band, so we mixed with f_{if} = 25 KHz and make a LPF Low Pass Filter to detect it at cutoff = 25 kHz

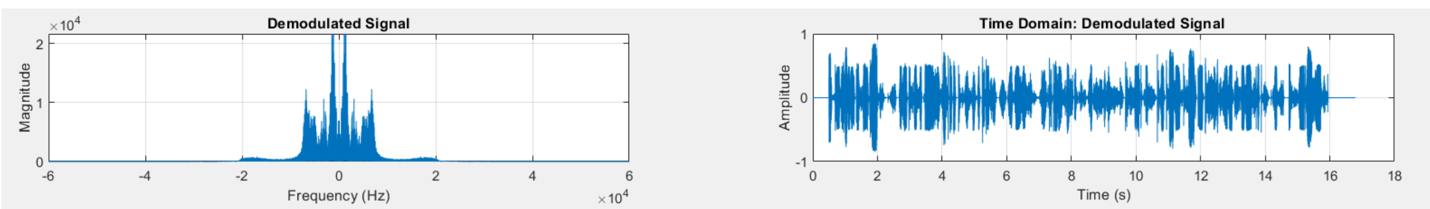

Figure 9: base band detection

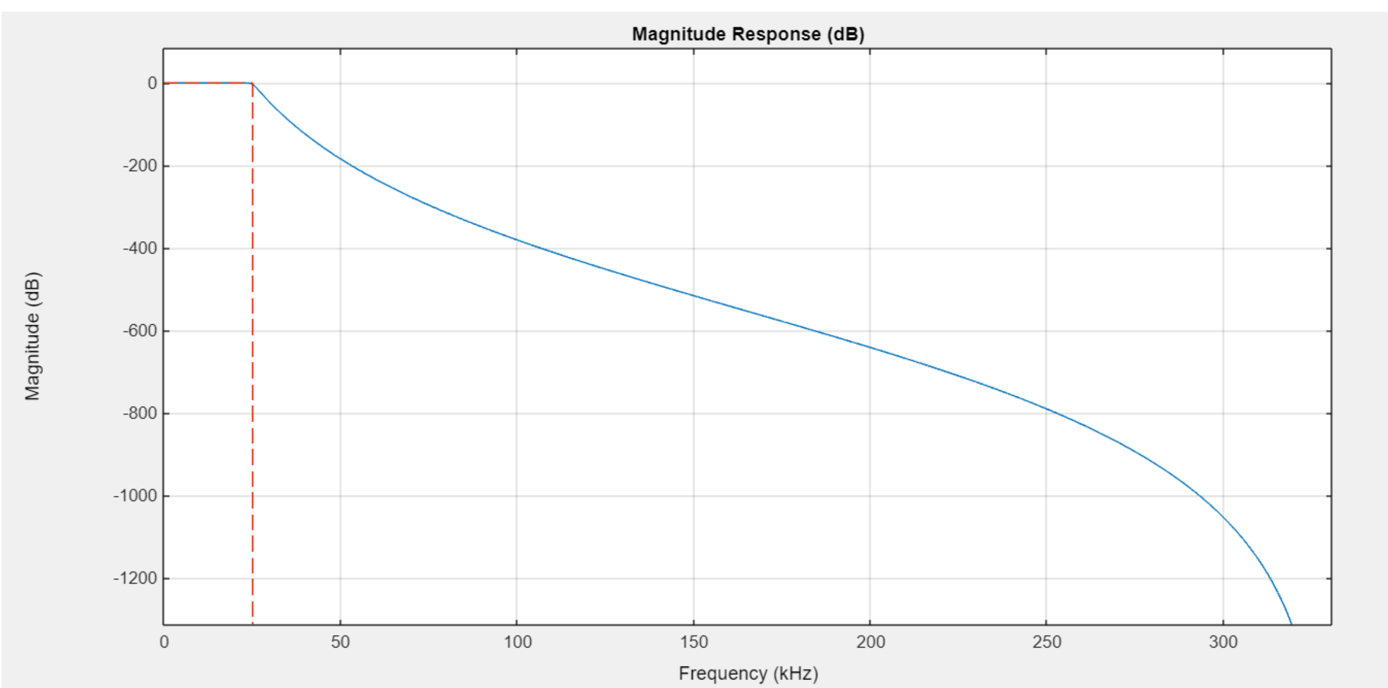

Figure 10:Low Pass Filter

Play the audio

AT Demodulation:

1- Down sample using the factor which I use to up sample the signal.
2- Normalize the signal to prevent clipping.
3- `sound (demodulated_signal_normalized, original_Fs);`

AT input → **sound (audio1, Fs1);** with his sample frequency

AFTER NOISE:

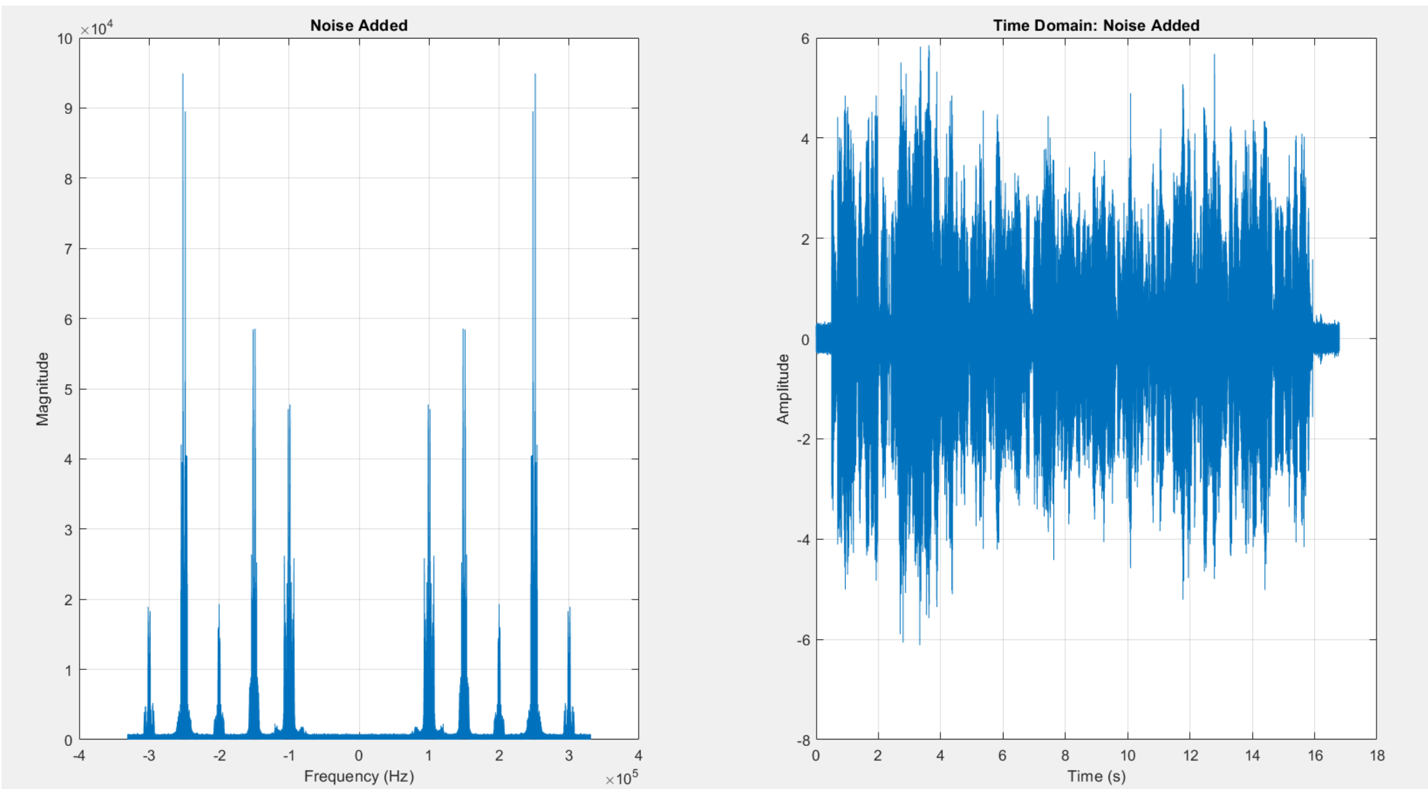

Figure 11: FDM With Noise

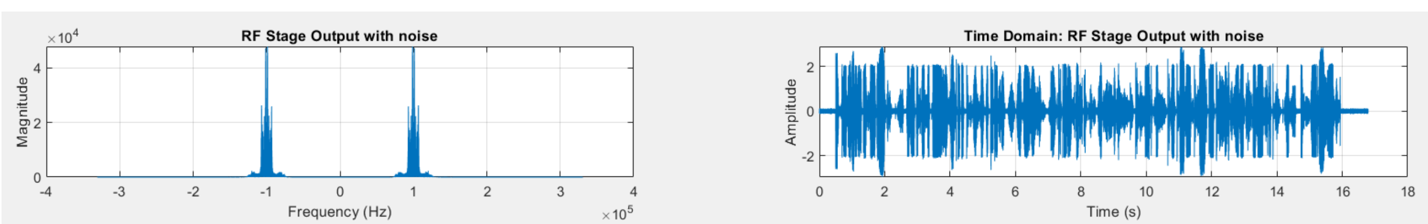

Figure12: RF output With Noise

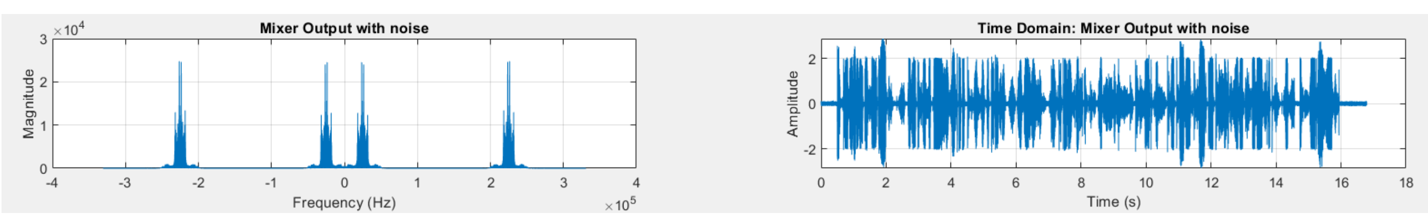

Figure13: Mixer output With Noise

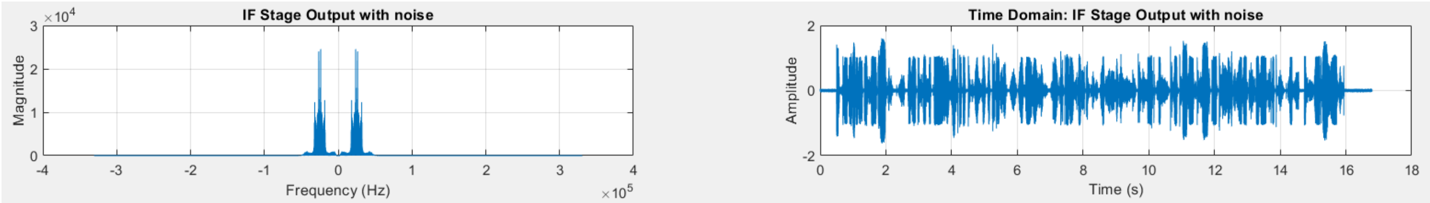

Figure14: IF stage output With Noise

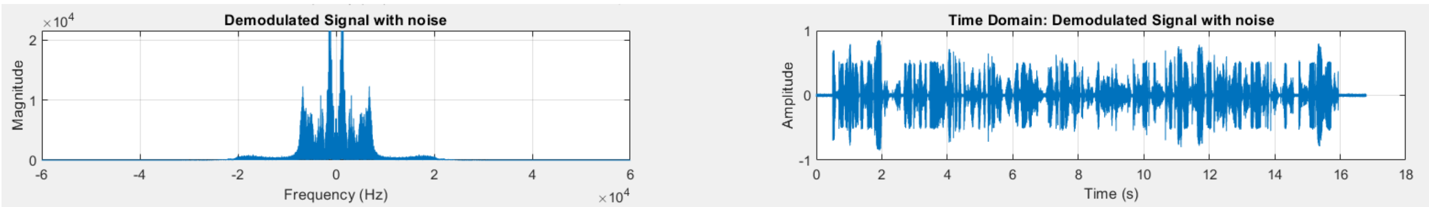

Figure15: demodulation output With Noise

The effect of Noise it like audible white noise.

AFTER REMOVE RF STAGE:

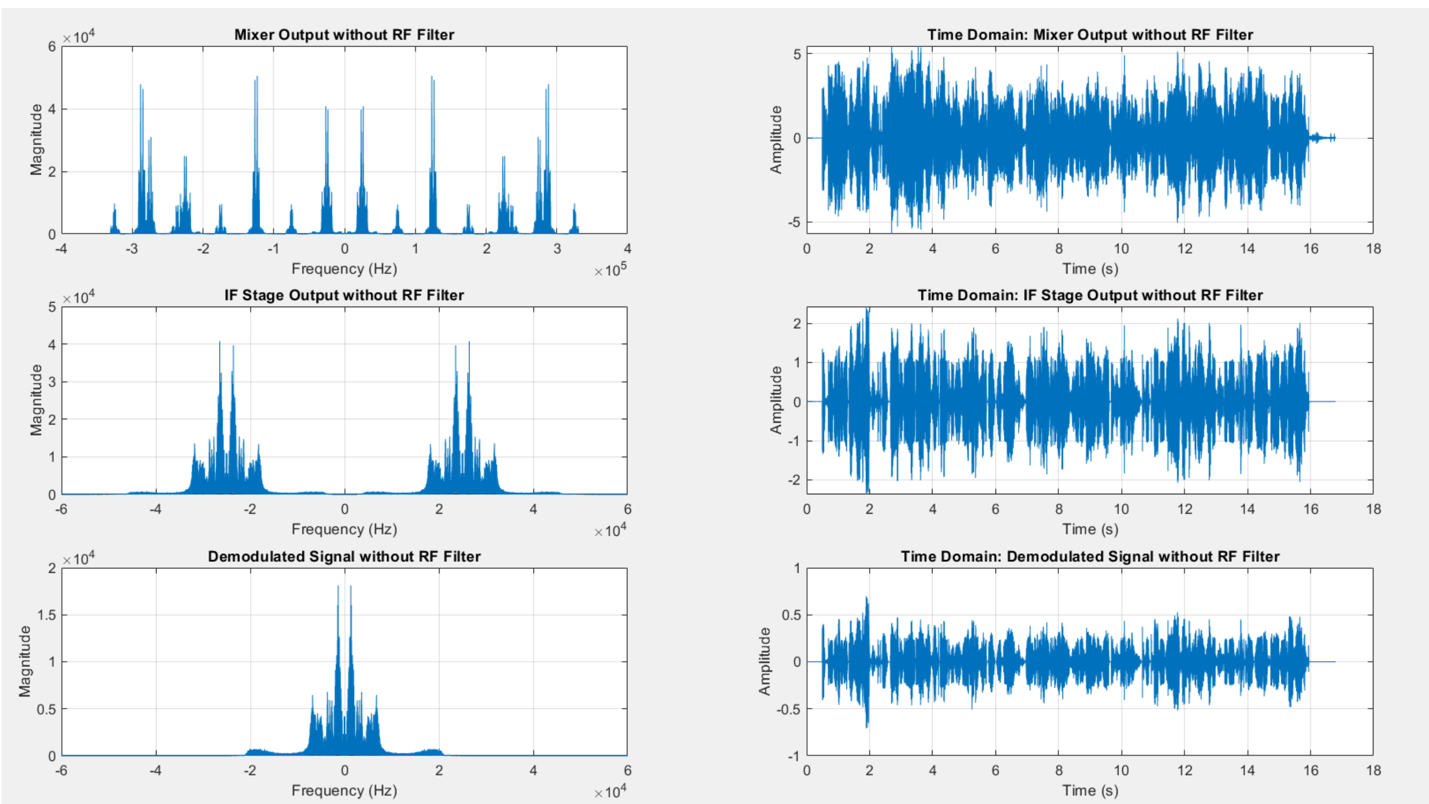

Figure16: Without RF stage

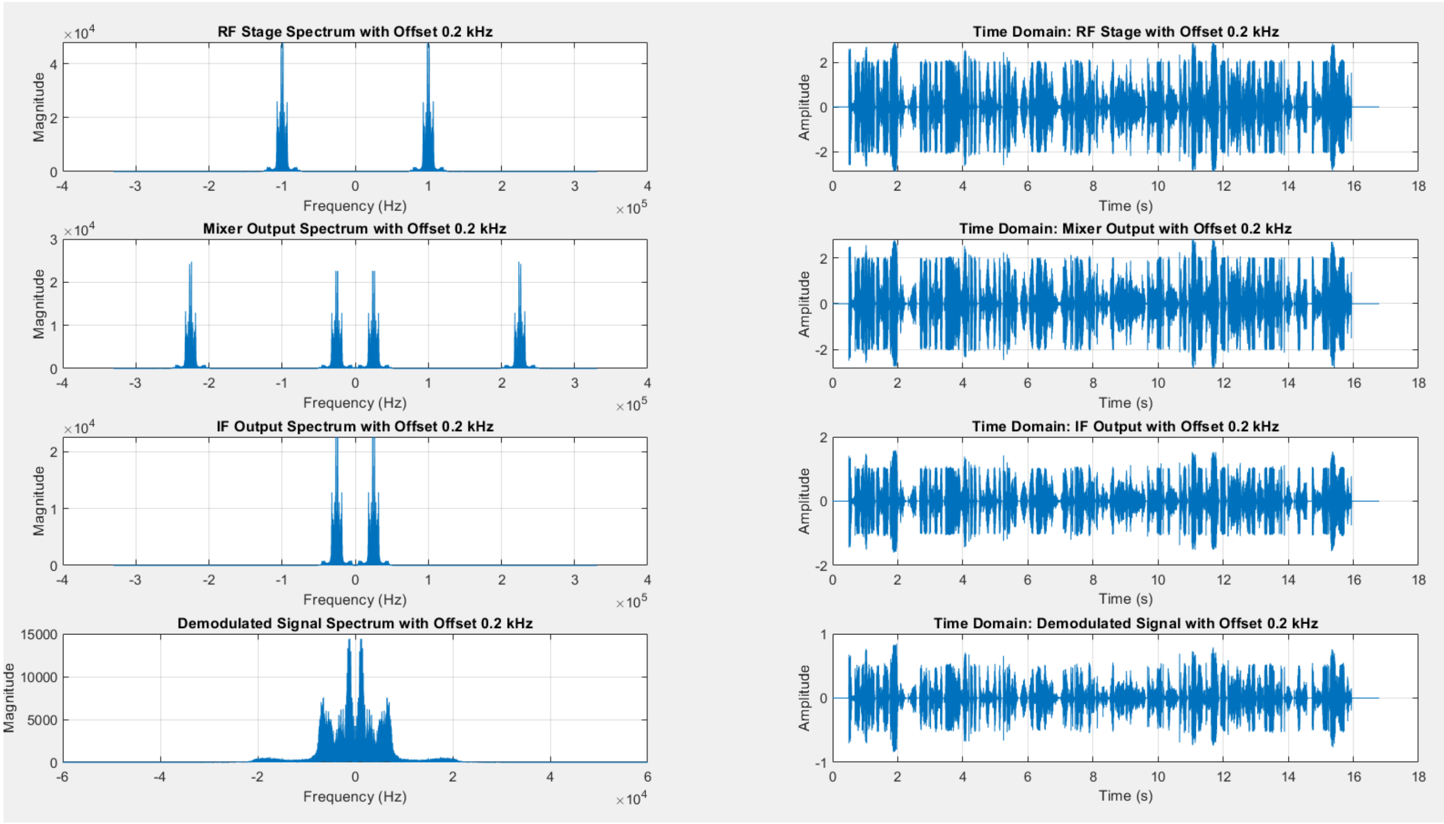

Figure17: with offset 200 Hz

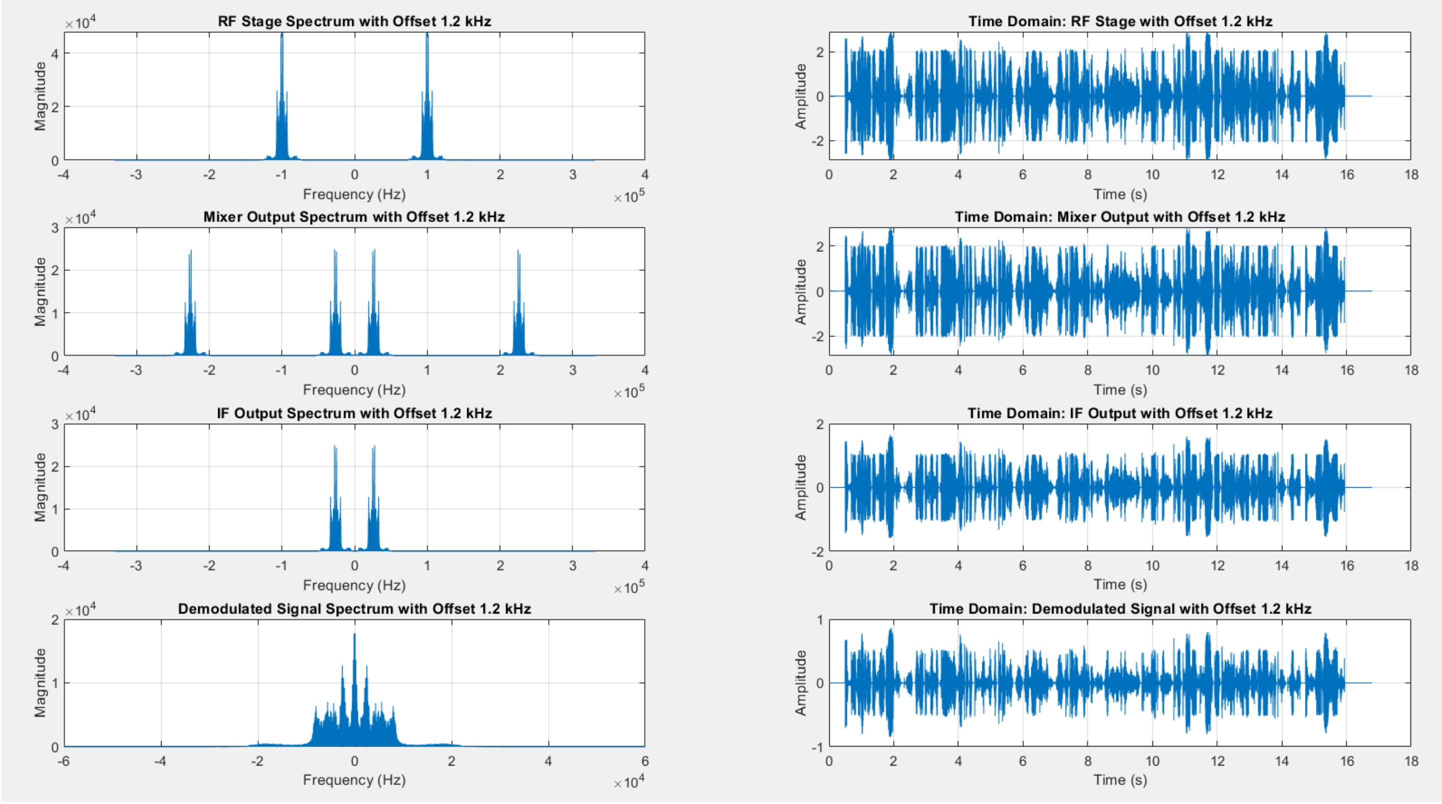

Figure18: with offset 1200 Hz

The offset makes a bad effect to the audio it is like an alien voice it is interference

Appendix

```
%% Step 1: Load and Prepare Audio Signals
% Load the stereo input signals (audio files)
[audio1, Fs1] = audioread('Short_RussianVoice.wav');
[audio2, Fs2] = audioread('Short_FM9090.wav');
[audio3, Fs3] = audioread('Short_SkyNewsArabia.wav');
[audio4, Fs4] = audioread('Short_QuranPalestine.wav');
[audio5, Fs5] = audioread('Short_BBCArabic2.wav');
%sound(audio1, Fs1);
% Ensure both signals have the same sampling frequency
Fs = max(Fs1, Fs2);%,Fs3,Fs4,Fs5
signal1 = resample(audio1, Fs, Fs1);
signal2 = resample(audio2, Fs, Fs2);
signal3 = resample(audio3, Fs, Fs3);
signal4 = resample(audio4, Fs, Fs4);
signal5 = resample(audio5, Fs, Fs5);
% Convert stereo to mono by averaging the two channels
signal1 = sum(signal1, 2);
signal2 = sum(signal2, 2);
signal3 = sum(signal3, 2);
signal4 = sum(signal4, 2);
signal5 = sum(signal5, 2);

% Pad signals to the same length
len = max(length(signal1), length(signal2));
len = max(length(len), length(signal3));
len = max(length(len), length(signal4));
len = max(length(len), length(signal5));
signal1 = [signal1; zeros(len - length(signal1), 1)];
signal2 = [signal2; zeros(len - length(signal2), 1)];
signal3 = [signal3; zeros(len - length(signal3), 1)];
signal4 = [signal4; zeros(len - length(signal4), 1)];
signal5 = [signal5; zeros(len - length(signal5), 1)];
% Increase sampling frequency by 15 times using 'interp'
upsample_factor = 15;
Fs_upsampled = Fs * upsample_factor;
signal1 = interp(signal1, upsample_factor);
signal2 = interp(signal2, upsample_factor);
signal3 = interp(signal3, upsample_factor);
signal4 = interp(signal4, upsample_factor);
signal5 = interp(signal5, upsample_factor);
% Update signal length and time vector
len = length(signal5);
t = (0:len-1)' / Fs_upsampled;

 % Plot the prepared signals
figure;
subplot(5, 2, 1);
plot(t, signal1, 'r'); % Red color
title('Time Domain: ShortRussianVoice');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5, 2, 3);
plot(t, signal2,  'Color',[1, 0.4, 0.6]); % Green color
title('Time Domain: ShortFM9090');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5, 2, 5);
plot(t, signal3, 'b'); % Blue color
title('Time Domain: ShortSkyNewsArabia');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5, 2, 7);
plot(t, signal4, 'Color',[0.4940, 0.1840, 0.5560]); % Magenta color
title('Time Domain: ShortQuranPalestine');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5, 2, 9);
plot(t, signal5, 'k'); % Black color
title('Time Domain: ShortBBCArabic2');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5, 2, 2);
plot_spectrum(signal1, Fs, 'Frequency Domain:ShortRussianVoice');
xlim([-8000 8000]);
subplot(5, 2, 4);
plot_spectrum(signal2, Fs, 'Frequency Domain:ShortFM9090');
xlim([-8000 8000]);
subplot(5, 2, 6);
plot_spectrum(signal3, Fs, 'Frequency Domain:ShortSkyNewsArabia');
xlim([-8000 8000]);
subplot(5, 2, 8);
plot_spectrum(signal4, Fs, 'Frequency Domain:ShortQuranPalestine');
xlim([-8000 8000]);
subplot(5, 2, 10);
plot_spectrum(signal5, Fs, 'Frequency Domain:ShortBBCArabic2');
xlim([-8000 8000]);

% Step 2: Perform AM Modulation (DSB-SC)
```

```
% Define carrier frequencies
Fc1 = 100e3; % Carrier frequency for signal 1 (100 kHz)
Fc2 = 150e3; % Carrier frequency for signal 2 (150 kHz)
Fc3 = 200e3; % Carrier frequency for signal 3 (200 kHz)
Fc4 = 250e3; % Carrier frequency for signal 4 (250 kHz)
Fc5 = 300e3; % Carrier frequency for signal 5 (300 kHz)

% Modulate signals
carrier1 = cos(2 * pi * Fc1 * t);
carrier2 = cos(2 * pi * Fc2 * t);
carrier3 = cos(2 * pi * Fc3 * t);
carrier4 = cos(2 * pi * Fc4 * t);
carrier5 = cos(2 * pi * Fc5 * t);

modulated1 = signal1 .* carrier1;
modulated2 = signal2 .* carrier2;
modulated3 = signal3 .* carrier3;
modulated4 = signal4 .* carrier4;
modulated5 = signal5 .* carrier5;


% Combine modulated signals to create FDM signal
FDM_signal = modulated1 + modulated2 + modulated3 + modulated4 + modulated5;

% Plot the FDM signal
figure;
subplot(1, 2, 1);
plot(t, FDM_signal);
title('Time Domain: FDM Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(1, 2, 2);
plot_spectrum(FDM_signal, Fs_upsampled, 'Frequency Domain: FDM Signal');
%xlim([70000 125000]);

Fs = Fs_upsampled;


Fc1 =100e3;

RF_fpass1 = Fc1 - 2.5e4; % Lower stopband frequency (kHz)
RF_fpass2 = Fc1 + 2.5e4; % Upper stopband frequency ( kHz)

RF_filter = designfilt('bandpassiir', ...
                       'FilterOrder', 30, ...
                       'HalfPowerFrequency1', RF_fpass1, ...
                       'HalfPowerFrequency2', RF_fpass2, ...
                       'SampleRate', Fs);
%fvtool(RF_filter);
RF_output = filter(RF_filter, FDM_signal);

% Oscillator and Mixer
IF_freq = 25e3; % 25 kHz IF frequency

LO_freq = Fc1 + IF_freq;
t_LO = t;
LO_signal = cos(2*pi*LO_freq*t_LO);
% Mixing operation
mixer_output = RF_output .* LO_signal;

iF_fpass1 =2950;
iF_fpass2 = 47050;

% Design RF bandpass filter using designfilt
IF_filter = designfilt('bandpassiir', ...
                       'FilterOrder', 30, ...
                       'HalfPowerFrequency1', iF_fpass1, ...
                       'HalfPowerFrequency2', iF_fpass2, ...
                       'SampleRate', Fs);
%fvtool(IF_filter);
IF_output = filter(IF_filter, mixer_output);

% Baseband Detection
% Final mixing with IF frequency
BB_mixer = IF_output .* cos(2*pi*IF_freq*t);

% Low-pass filter for baseband detection

LPF_fp = 25000;

% Design RF bandpass filter using designfilt
LPF_filter = designfilt('lowpassiir', ...
                       'FilterOrder', 30, ...
                       'HalfPowerFrequency', LPF_fp, ...
                       'SampleRate', Fs);
%fvtool(LPF_filter);
demodulated_signal = filter(LPF_filter, BB_mixer);

% Plot results
figure;
subplot(4,2,1);
plot_spectrum(RF_output, Fs, 'RF Stage Output');
%xlim([40000 150000]);
subplot(4,2,2);
plot(t, RF_output); % Black color
title('Time Domain: RF Stage Output');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

```
subplot(4,2,3);
plot_spectrum(mixer_output, Fs, 'Mixer Output');
subplot(4,2,4);
plot(t, mixer_output); % Black color
title('Time Domain: Mixer Output');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(4,2,5);
plot_spectrum(IF_output, Fs, 'IF Stage Output');
%xlim([-60000 60000]);
subplot(4,2,6);
plot(t, IF_output); % Black color
title('Time Domain: IF Stage Output');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(4,2,7);
plot_spectrum(demodulated_signal, Fs, 'Demodulated Signal');
xlim([-60000 60000]);
subplot(4,2,8);
plot(t, demodulated_signal); % Black color
title('Time Domain: Demodulated Signal ');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

original_Fs = 44100; % or whatever your original sampling rate was
downsample_factor = Fs_upsampled / original_Fs;
demodulated_signal_downsampled = downsample(demodulated_signal, downsample_factor);

% Normalize the signal to prevent clipping
demodulated_signal_normalized = demodulated_signal_downsampled / max(abs(demodulated_signal_downsampled));

% Method 1: Using sound function
%sound(demodulated_signal_normalized, original_Fs);
% Original Code (Previous stages up to FDM signal creation remains the same)



% Add noise to the FDM signal
SNR_dB = 20; % Signal-to-Noise Ratio in dB (you can adjust this)
noisy_FDM = awgn(FDM_signal, SNR_dB, 'measured');

figure;
subplot(1,2,1);
plot_spectrum(noisy_FDM,Fs,"Noise Added");
subplot(1, 2, 2);
plot(t, noisy_FDM);
title('Time Domain: Noise Added');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;


RF_outputwithnoise = filter(RF_filter, noisy_FDM);

mixer_outputwithnoise = RF_outputwithnoise .* LO_signal;




IF_outputwithnoise = filter(IF_filter, mixer_outputwithnoise);

BB_mixerwithnoise = IF_outputwithnoise .* cos(2*pi*IF_freq*t);
demodulated_signalwithnoise = filter(LPF_filter, BB_mixerwithnoise);

% Plot results
figure;
subplot(4,2,1);
plot_spectrum(RF_outputwithnoise, Fs, 'RF Stage Output with noise');
%xlim([40000 150000]);
subplot(4,2,2);
plot(t, RF_outputwithnoise); % Black color
title('Time Domain: RF Stage Output with noise');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(4,2,3);
plot_spectrum(mixer_outputwithnoise, Fs, 'Mixer Output with noise');
subplot(4,2,4);
plot(t, mixer_outputwithnoise); % Black color
title('Time Domain: Mixer Output with noise');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(4,2,5);
plot_spectrum(IF_outputwithnoise, Fs, 'IF Stage Output with noise');
%xlim([-60000 60000]);
subplot(4,2,6);
plot(t, IF_outputwithnoise); % Black color
title('Time Domain: IF Stage Output with noise');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(4,2,7);
plot_spectrum(demodulated_signalwithnoise, Fs, 'Demodulated Signal with noise');
xlim([-60000 60000]);
subplot(4,2,8);
plot(t, demodulated_signalwithnoise); % Black color
title('Time Domain: Demodulated Signal with noise');
```

```
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

original_Fs = 44100; % or whatever your original sampling rate was
downsample_factor = Fs_upsampled / original_Fs;
demodulated_signal_downsampled_noise = downsample(demodulated_signalwithnoise, downsample_factor);

% Normalize the signal to prevent clipping
demodulated_signal_normalized_noise = demodulated_signal_downsampled_noise / max(abs(demodulated_signal_downsampled_noise));
demodulated_signalwithnoise = filter(LPF_filter, BB_mixerwithnoise);
%sound(demodulated_signal_normalized_noise, original_Fs);
%}
mixer_output_no_RF = FDM_signal .* LO_signal;


IF_output_no_RF = filter(IF_filter, mixer_output_no_RF);

BB_mixer_no_rf = IF_output_no_RF .* cos(2*pi*IF_freq*t);
LPF_fpnorf = 22100;

% Design RF bandpass filter using designfilt
LPF_filter_no_rf = designfilt('lowpassiir', ...
                       'FilterOrder', 30, ...
                       'HalfPowerFrequency', LPF_fpnorf, ...
                       'SampleRate', Fs);


demodulated_signal_no_RF = filter(LPF_filter_no_rf, BB_mixer_no_rf);
demodulated_signal_downsampled_norf= downsample(demodulated_signal_no_RF, downsample_factor);

% Normalize the signal to prevent clipping
demodulated_signal_normalized_norf = demodulated_signal_downsampled_norf / max(abs(demodulated_signal_downsampled_norf));
demodulated_signalwithnoise = filter(LPF_filter, BB_mixerwithnoise);
sound(demodulated_signal_normalized_norf, original_Fs);
% Plot the spectrum of the output signals without RF filter

figure;
subplot(3, 2, 1);
plot_spectrum(mixer_output_no_RF,Fs,'Mixer Output without RF Filter');
subplot(3,2,2);
plot(t, mixer_output_no_RF); % Black color
title('Time Domain: Mixer Output without RF Filter');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;


subplot(3, 2, 3);
plot_spectrum(IF_output_no_RF,Fs,'IF Stage Output without RF Filter');
xlim([-60000 60000]);
subplot(3,2,4);
plot(t, IF_output_no_RF); % Black color
title('Time Domain: IF Stage Output without RF Filter');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3, 2, 5);
plot_spectrum(demodulated_signal_no_RF,Fs,'Demodulated Signal without RF Filter');
xlim([-60000 60000]);
subplot(3,2,6);
plot(t, demodulated_signal_no_RF); % Black color
title('Time Domain: Demodulated Signal without RF Filter');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Define frequency offsets
offsets = [0.2e3, 1.2e3];  % Frequency offsets in Hz (0.2 kHz and 1.2 kHz)

% Loop over the frequency offsets
for i = 1:2
    % Apply the frequency offset to the LO frequency
    LO_freq_offset = Fc1 + IF_freq + offsets(i);  % Local oscillator with offset
    LO_signal_offset = cos(2*pi*LO_freq_offset*t_LO);  % LO signal with offset
    % Mixer output with the frequency offset
    mixer_output_offset = RF_output .* LO_signal_offset;
    % Apply IF filter to the mixer output with offset
    IF_output_offset = filter(IF_filter, mixer_output_offset);
    % Apply baseband detection
    BB_mixer_offset = IF_output_offset .* cos(2*pi*IF_freq*t);  % Final mixing with IF frequency
    % Apply the low-pass filter to demodulate the signal with offset
    demodulated_signal_offset = filter(LPF_filter, BB_mixer_offset);
    N = length(IF_output_offset);
    f = (-N/2:N/2-1)*(Fs/N); % Frequency axis
    % **Plot the spectrum and time-domain signals**
    figure;
    % RF Stage
    subplot(4, 2, 1);
    plot(f, abs(fftshift(fft(RF_output)))); % RF Stage Spectrum with offset
    title(['RF Stage Spectrum with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    grid on;
    subplot(4, 2, 2);
    plot(t, RF_output); % Time-domain signal for RF stage
    title(['Time Domain: RF Stage with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
    % Mixer Output
```

```
    subplot(4, 2, 3);
    plot(f, abs(fftshift(fft(mixer_output_offset)))); % Mixer Output Spectrum with offset
    title(['Mixer Output Spectrum with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    grid on;
    subplot(4, 2, 4);
    plot(t, mixer_output_offset); % Time-domain signal for mixer output
    title(['Time Domain: Mixer Output with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
    % IF Output
    subplot(4, 2, 5);
    plot(f, abs(fftshift(fft(IF_output_offset)))); % IF Output Spectrum with offset
    title(['IF Output Spectrum with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    grid on;
    subplot(4, 2, 6);
    plot(t, IF_output_offset); % Time-domain signal for IF output
    title(['Time Domain: IF Output with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
    % Demodulated Signal
    subplot(4, 2, 7);
    plot(f, abs(fftshift(fft(demodulated_signal_offset)))); % Demodulated Signal Spectrum with offset
    xlim([-60000 60000]);
    title(['Demodulated Signal Spectrum with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    grid on;

    subplot(4, 2, 8);
    plot(t, demodulated_signal_offset); % Time-domain signal for demodulated signal
    title(['Time Domain: Demodulated Signal with Offset ', num2str(offsets(i)/1e3), ' kHz']);
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
   demodulated_signal_downsampled_offset= downsample(demodulated_signal_offset, downsample_factor);

% Normalize the signal to prevent clipping
demodulated_signal_normalized_offset = demodulated_signal_downsampled_offset / max(abs(demodulated_signal_downsampled_offset));

sound(demodulated_signal_normalized_offset, original_Fs);
    pause(5);
end

function plot_spectrum(signal, Fs, titleStr)
    N = length(signal);
    f = (-N/2:N/2-1)*(Fs/N); % Frequency axis
    spectrum = abs(fftshift(fft(signal))); % Compute FFT and shift
    plot(f, spectrum);
    xlabel('Frequency (Hz)');

    ylabel('Magnitude');
    title(titleStr);
    grid on;
end
```