"MODELING FREQUENCY DIVISION MULTIPLEXING/DE-MULTIPLEXING"

LAB # 09

FALL 2023

DCSE Digital Signals Processing Lab

Submitted By: Esha Rizwan

Registration No: 21PWCSE2010

Section: C

"On my honor, as a student of University of Engineering and Technology, I have neither given not received unauthorized assistance on this academic work."

Submitted To:

Sir Yasir Saleem

January 8th, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Lab Objectives:

- To understand and learn the concept of multiplexing.
- To understand the concept of de-multiplexing.
- To learn to use MATLAB for multiplexing and de-multiplexing.

MULTIPLEXING:

Multiplexing is a technique used in telecommunications and data transmission to combine multiple signals into a single signal that can be transmitted over a shared medium, such as a cable or a communication channel, and then separated back into individual signals at the receiving end. Types of multiplexing are:

- Time Division Multiplexing (TDM)
- Frequency Division Multiplexing (FDM)
- Wavelength Division Multiplexing (WDM)
- Code Division Multiplexing (CDM)

Multiplexing significantly increases the efficiency of communication channels by allowing multiple signals to share the same medium, reducing costs and maximizing bandwidth utilization.

DE-MULTIPLEXING:

De-multiplexing is the process of separating a combined signal, which contains multiple individual signals that were multiplexed together, back into its original constituent signals at the receiving end.

When multiple signals are multiplexed together for transmission over a shared medium, such as in techniques like Time Division Multiplexing (TDM), Frequency Division Multiplexing (FDM), or others, they are combined into a single signal for transmission. At the receiving end, the de-multiplexing process is employed to separate and extract each individual signal from the combined signal.

LAB TASKS:

Implement the following steps in MATLAB to multiplex three input voice signals at the transmitter end and de-multiplex and play them back at the Receiver end. Add random noise to the signal while propagating via the channel.

SETTING UP THE PARAMETERS.

CODE:

Editor - D:\Esha\University\5th semester\DSP Lab\dsplab9\task1.m

lab8.m | task11.m | task1.m | task1.m | task2.m

```
%parameters
bandwidth = 4000;
guard_band = 300;
signal_to_noise_ratio = 20;
ssb_modulation = 1;

carrier_frequency1 = bandwidth * 3;
carrier_frequency2 = bandwidth * 4;
carrier_frequency3 = bandwidth * 5;

sampling_frequency = carrier_frequency3 * 2 + 5000;
cutoff_frequency = 2500;

show_graphics = 1;
play_sounds = 1;

[B, A] = butter(4, cutoff_frequency / (sampling_frequency / 2));
low_pass_filter = @(signal) filter(B, A, signal);

[C1, D1] = butter(2, [bandwidth*2 + guard_band, bandwidth*3 - guard_band] / (sampling_frequency / 2));
band_filter3 = @(signal) filter(C1, D1, signal);

[C2, D2] = butter(2, [bandwidth*3 + guard_band, bandwidth*4 - guard_band] / (sampling_frequency / 2));
band_filter4 = @(signal) filter(C2, D2, signal);

[C3, D3] = butter(2, [bandwidth*4 + guard_band, bandwidth*5 - guard_band] / (sampling_frequency / 2));
band_filter5 = @(signal) filter(C3, D3, signal);
```

```
signal1 = audioread('male1.wav');
lsignal1 = length(signal1);

signal2 = audioread('male22.wav');
lsignal2 = length(signal2);

signal3 = audioread('femalee.wav');
lsignal3 = length(signal3);

beep_sound = audioread('beep.wav');
beep_player = audioplayer(beep_sound, 44100);

min_length = min([lsignal1, lsignal2]);

time = linspace(0, 5, min_length);
```

THE SIGNALS ARE REPRODUCED AS THEY ARRIVE.

CODE:

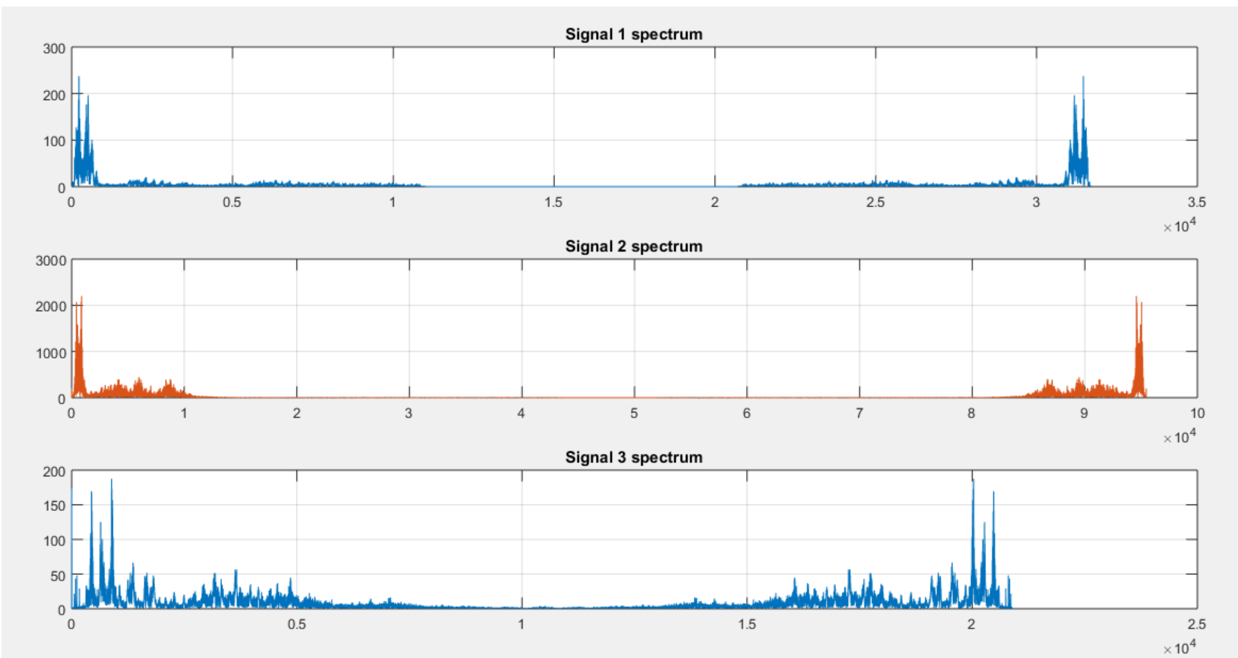
```
disp('STEP - 1, The signals are reproduced as they arrive');

if(play_sounds > 0)
    p1 = audioplayer(signal1, 44100);
    playblocking(p1);
    playblocking(beep_player);

    p2 = audioplayer(signal2, 44100);
    playblocking(p2);
    playblocking(beep_player);

    p3 = audioplayer(signal3, 44100);
    playblocking(p3);
end
```

PLOT THE SPECTRA OF THE SIGNALS AS THEY ARRIVE (USE FFT AND DSP.SPECTRUMANALYZER FOR COMPARISON).

CODE:

```
%%
disp('STEP - 2, Plot the spectra of the signals as they arrive');

if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(signal1));
    subplot(3,1,1), plot(spectrum1), grid on, zoom, title('Signal 1 spectrum');

    spectrum2 = abs(fft(signal2));
    subplot(3,1,2), plot(spectrum2), grid on, zoom, title('Signal 2 spectrum');

    spectrum3 = abs(fft(signal3));
    subplot(3,1,3), plot(spectrum3), grid on, zoom, title('Signal 3 spectrum');
end
```

PLOT:

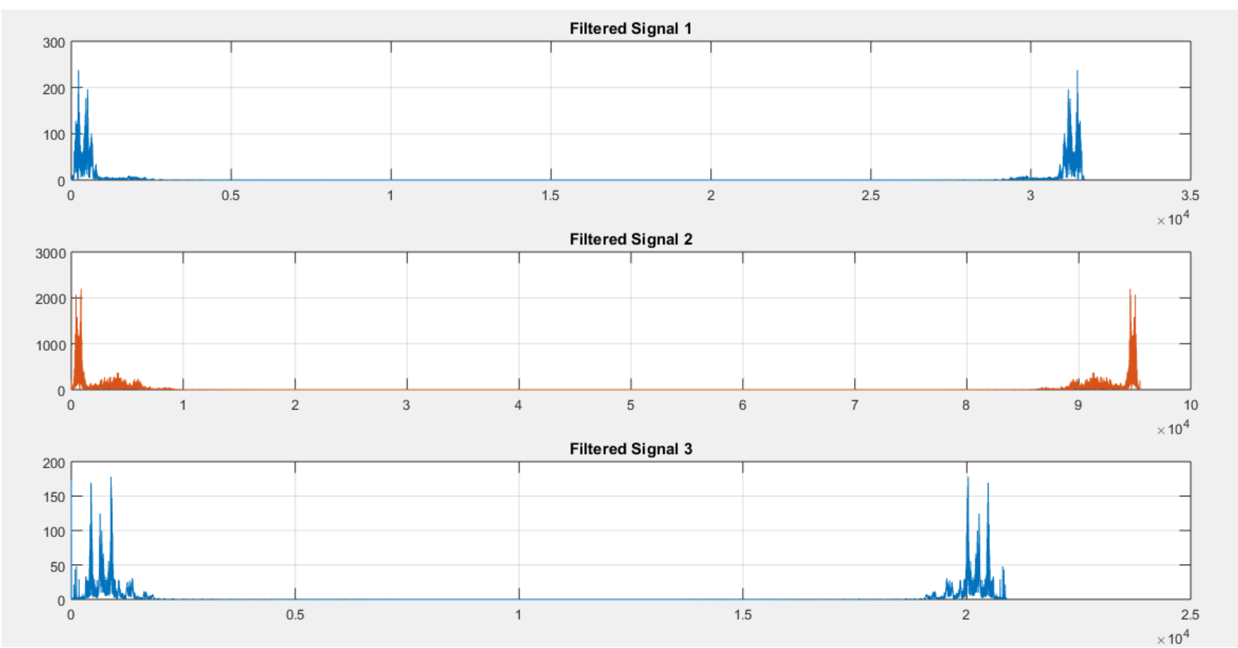


THE SIGNALS ARE PASSED THROUGH A LOW PASS FILTER AND PLOTTED.

CODE:

```
disp('STEP - 3, The signals are passed through a low pass filter and played');

signal1 = low_pass_filter(signal1);
signal2 = low_pass_filter(signal2);
signal3 = low_pass_filter(signal3);

if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(signal1));
    subplot(3,1,1), plot(spectrum1), grid on, zoom, title('Filtered Signal 1');

    spectrum2 = abs(fft(signal2));
    subplot(3,1,2), plot(spectrum2), grid on, zoom, title('Filtered Signal 2');

    spectrum3 = abs(fft(signal3));
    subplot(3,1,3), plot(spectrum3), grid on, zoom, title('Filtered Signal 3');
end
```

PLOT:



REPRODUCE THE SIGNALS AFTER PASSING THEM THROUGH THE FILTER.

CODE:

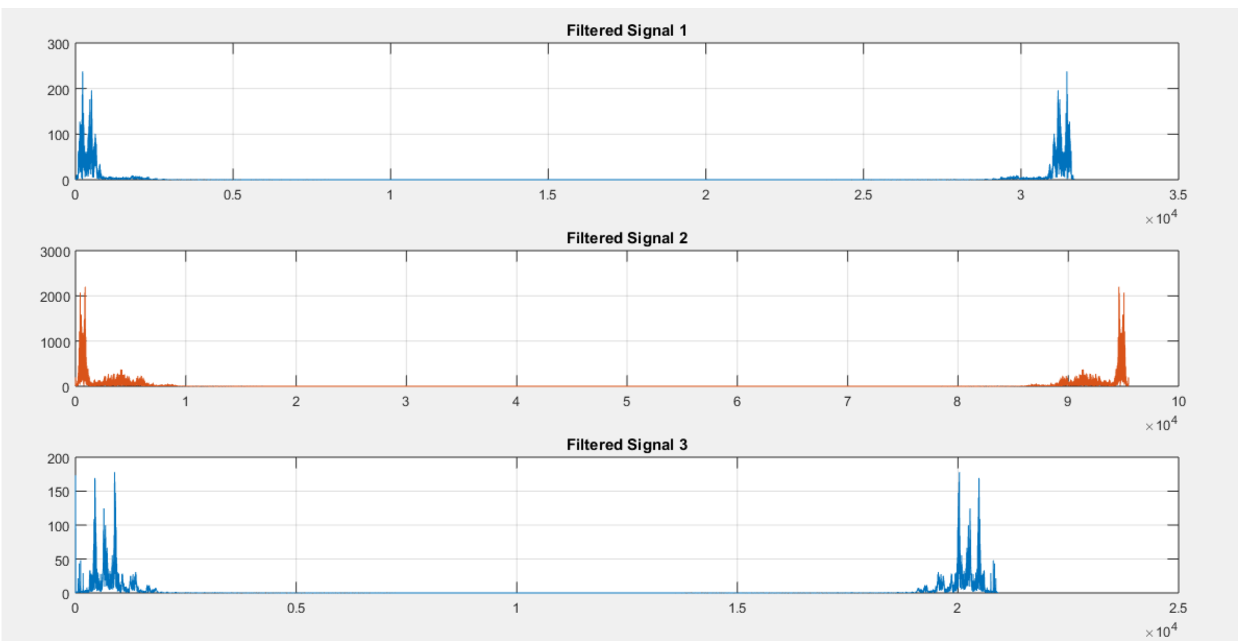
```
%%
disp('STEP - 4, Repoduce the signals after passing through the filter');

if(play_sounds > 0)
    beep_player = audioplayer(beep_sound, 44100);

    p1 = audioplayer(signal1, 44100);
    playblocking(p1);
    playblocking(beep_player);

    p2 = audioplayer(signal2, 44100);
    playblocking(p2);
    playblocking(beep_player);

    p3 = audioplayer(signal3, 44100);
    playblocking(p3);
    playblocking(beep_player);
end
```

THE SIGNALS ARE MODULATED TO DIFFERENT CARRIERS.

CODE:

```
disp('STEP - 5, The signals are modulated to different carriers');

if(ssb_modulation > 0)
    modulated_signal1 = ssbmod(signal1, carrier_frequency1, sampling_frequency);
    modulated_signal2 = ssbmod(signal2, carrier_frequency2, sampling_frequency);
    modulated_signal3 = ssbmod(signal3, carrier_frequency3, sampling_frequency);
else
    modulated_signal1 = ammod(signal1, carrier_frequency1, sampling_frequency);
    modulated_signal2 = ammod(signal2, carrier_frequency2, sampling_frequency);
    modulated_signal3 = ammod(signal3, carrier_frequency3, sampling_frequency);
end

if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(signal1));
    subplot(3,1,1), plot(spectrum1), grid on, zoom, title('Filtered Signal 1');

    spectrum2 = abs(fft(signal2));
    subplot(3,1,2), plot(spectrum2), grid on, zoom, title('Filtered Signal 2');

    spectrum3 = abs(fft(signal3));
    subplot(3,1,3), plot(spectrum3), grid on, zoom, title('Filtered Signal 3');
end
```

PLOT:

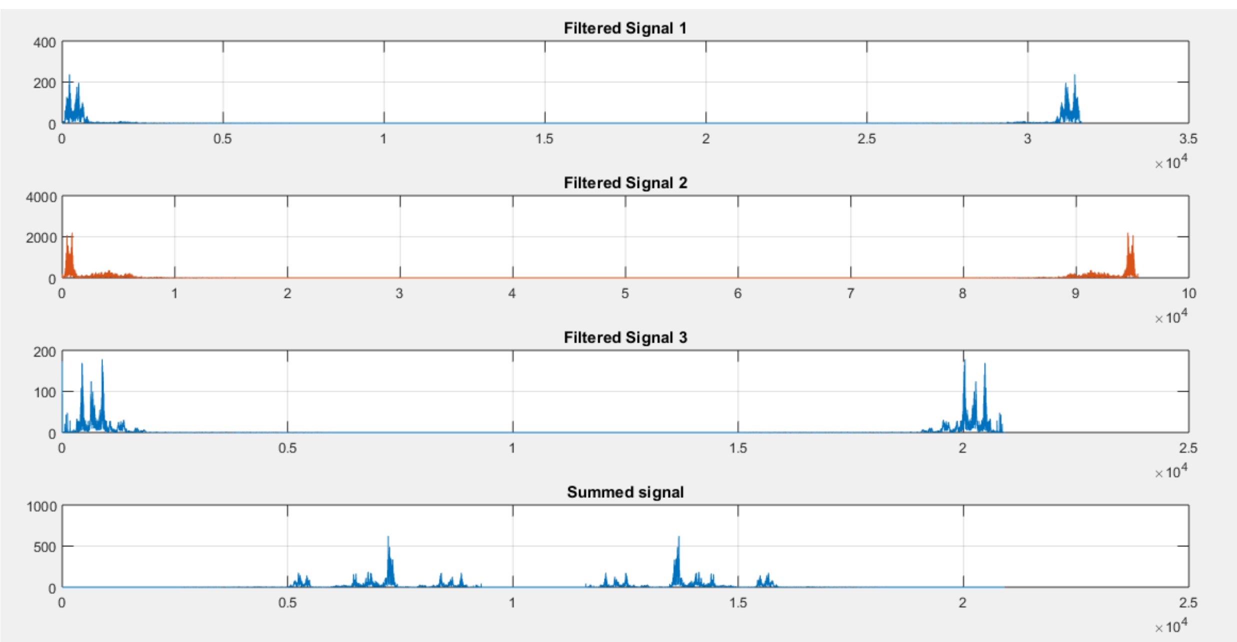


THE MODULATED SIGNALS ARE FILTERED IN THE GIVEN BAND AND ADDED TOGETHER.

CODE:

```
disp('STEP - 6, The modulated signals are filtered in the defined bands and added');

filtered_signal1 = band_filter3(modulated_signal1);
filtered_signal2 = band_filter4(modulated_signal2);
filtered_signal3 = band_filter5(modulated_signal3);

% Find the minimum length among the signals
min_length = min([length(filtered_signal1), length(filtered_signal2), length(filtered_signal3)]);

% Sum the signals element-wise up to the minimum length
complete_signal = zeros(min_length, 1);
for i = 1:min_length
    complete_signal(i) = filtered_signal1(i) + filtered_signal2(i) + filtered_signal3(i);
end


if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(signal1));
    subplot(4,1,1), plot(spectrum1), grid on, zoom, title('Filtered Signal 1');

    spectrum2 = abs(fft(signal2));
    subplot(4,1,2), plot(spectrum2), grid on, zoom, title('Filtered Signal 2');

    spectrum3 = abs(fft(signal3));
```

PLOT:

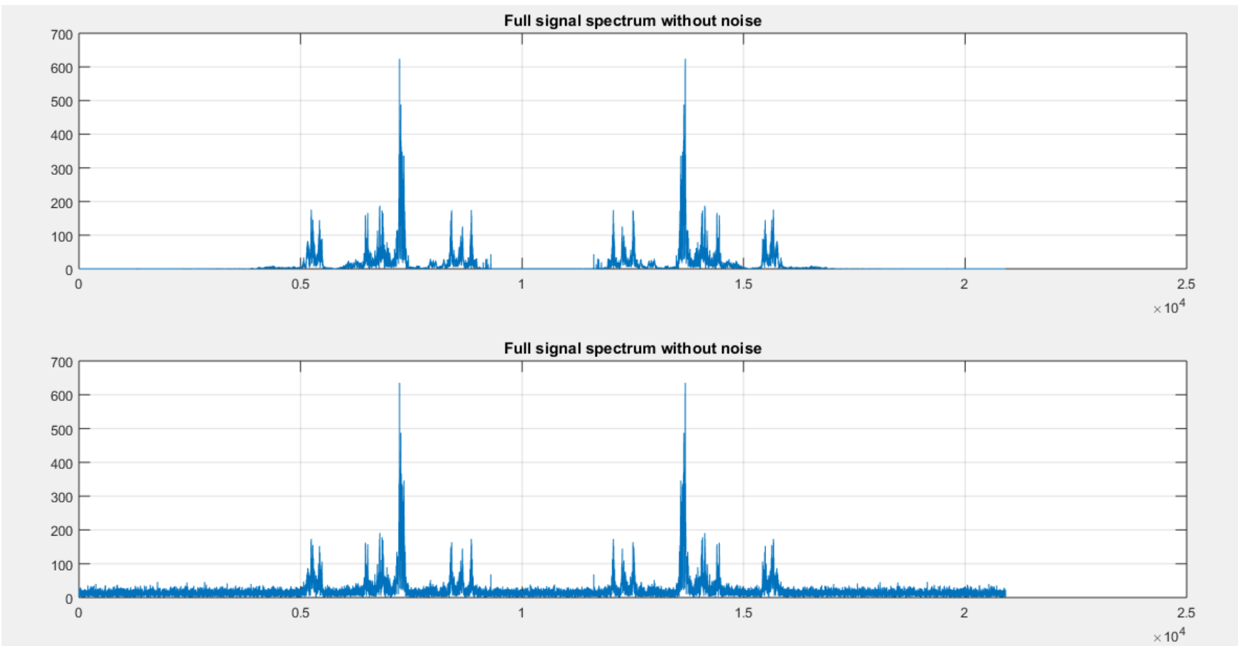


SOME NOISE IS ADDED TO THE TRANSMITTED SIGNAL.

CODE:

```
disp('STEP - 7, Spectrum of transmitted signal before adding noise');

if (show_graphics > 0)
    figure

    total_spectrum = abs(fft(complete_signal));
    subplot(2,1,1), plot(total_spectrum), grid on, zoom, title('Full signal spectrum without noise');
end

complete_signal = awgn(complete_signal, signal_to_noise_ratio);

if (show_graphics > 0)
    total_spectrum = abs(fft(complete_signal));
    subplot(2,1,2), plot(total_spectrum), grid on, zoom, title('Full signal spectrum without noise');
end
```

PLOT:

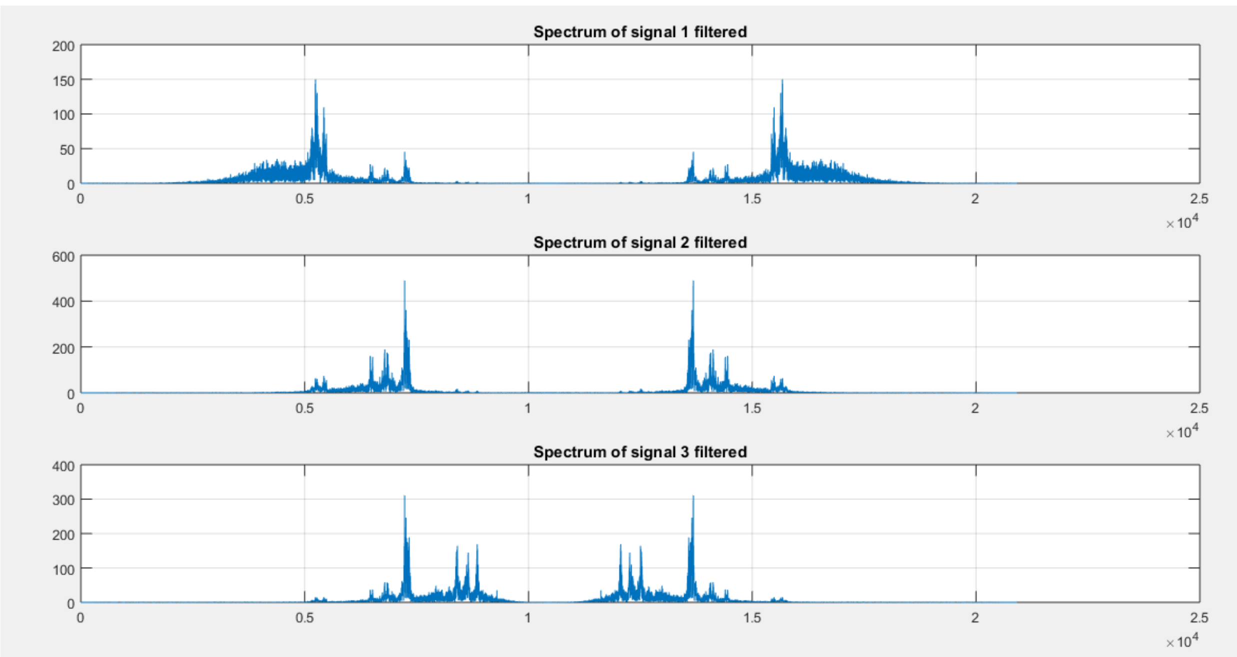


UPON ARRIVAL EACH BAND IS FILTERED.

CODE:

```
disp('STEP - 8, Upon arrival each band is filtered');

demod_signal1 = band_filter3(complete_signal);
demod_signal2 = band_filter4(complete_signal);
demod_signal3 = band_filter5(complete_signal);

if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(demod_signal1));
    subplot(3,1,1), plot(spectrum1), grid on, zoom, title('Spectrum of signal 1 filtered');

    spectrum2 = abs(fft(demod_signal2));
    subplot(3,1,2), plot(spectrum2), grid on, zoom, title('Spectrum of signal 2 filtered');

    spectrum3 = abs(fft(demod_signal3));
    subplot(3,1,3), plot(spectrum3), grid on, zoom, title('Spectrum of signal 3 filtered');
end
```

PLOT:

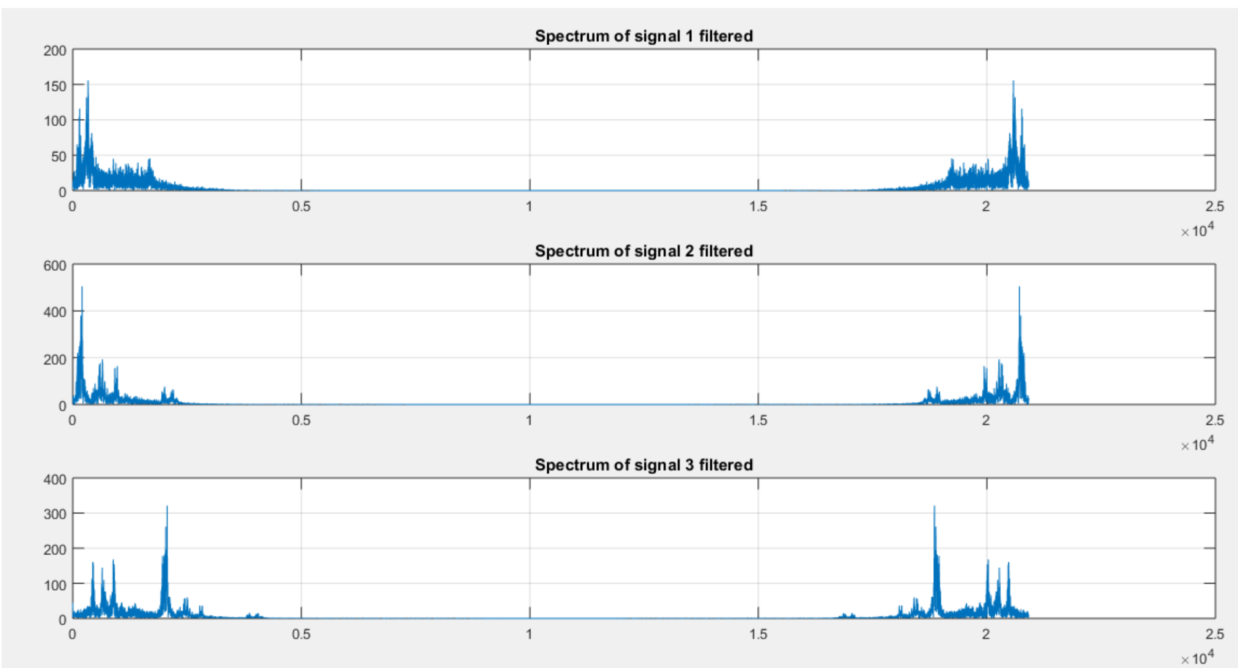


EACH RECOVERED BAND IS DEMODULATED TO RETURN THE SIGNAL AT THE INDICATED FREQUENCY.

CODE:

```
disp('STEP - 9, Each recovered band is demodulated to return the signal to the baseband frequency');

if(ssb_modulation > 0)
    demod_signal1 = ssbmod(demod_signal1, carrier_frequency1, sampling_frequency);
    demod_signal2 = ssbmod(demod_signal2, carrier_frequency2, sampling_frequency);
    demod_signal3 = ssbmod(demod_signal3, carrier_frequency3, sampling_frequency);
else
    demod_signal1 = ammod(demod_signal1, carrier_frequency1, sampling_frequency);
    demod_signal2 = ammod(demod_signal2, carrier_frequency2, sampling_frequency);
    demod_signal3 = ammod(demod_signal3, carrier_frequency3, sampling_frequency);
end

if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(demod_signal1));
    subplot(3,1,1), plot(spectrum1), grid on, zoom, title('Spectrum of signal 1 filtered');

    spectrum2 = abs(fft(demod_signal2));
    subplot(3,1,2), plot(spectrum2), grid on, zoom, title('Spectrum of signal 2 filtered');

    spectrum3 = abs(fft(demod_signal3));
    subplot(3,1,3), plot(spectrum3), grid on, zoom, title('Spectrum of signal 3 filtered');
end
```

PLOT:

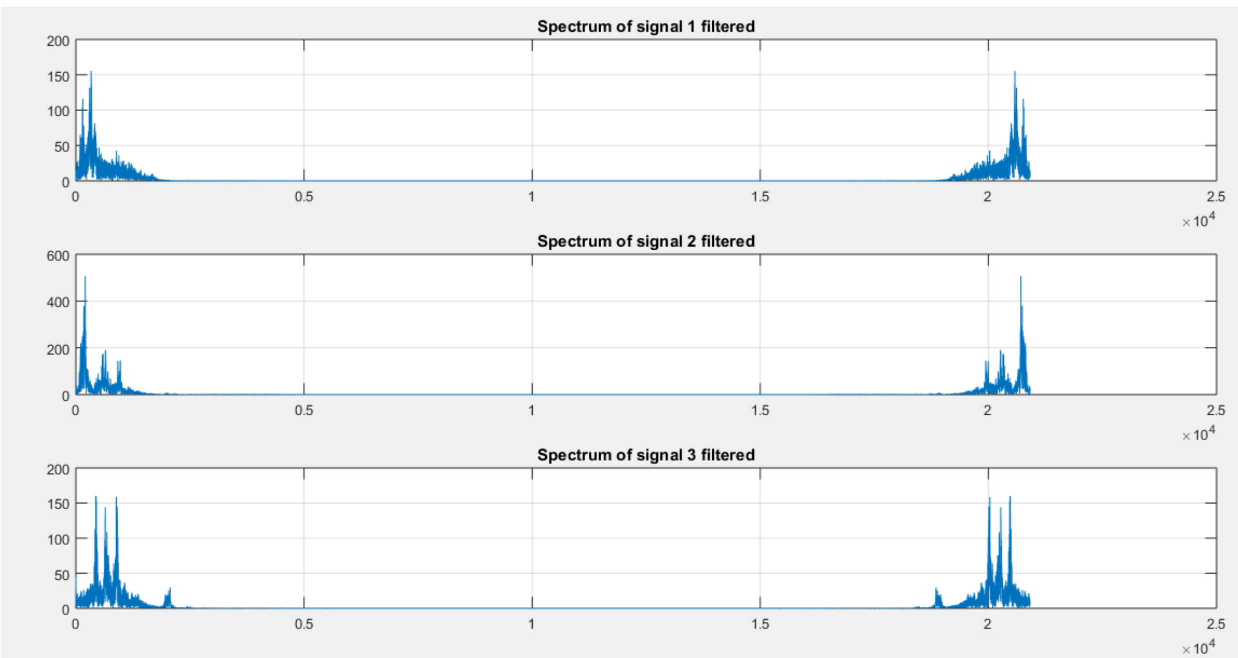


THE RECOVERED SIGNAL IS PASSED THROUGH A LOW PASS FILTER.

CODE:

```
%%
disp('STEP - 10, The recovered signal is passed through a low pass filter');

demod_signal1 = low_pass_filter(demod_signal1);
demod_signal2 = low_pass_filter(demod_signal2);
demod_signal3 = low_pass_filter(demod_signal3);

if(show_graphics > 0)
    figure

    spectrum1 = abs(fft(demod_signal1));
    subplot(3,1,1), plot(spectrum1), grid on, zoom, title('Spectrum of signal 1 filtered');

    spectrum2 = abs(fft(demod_signal2));
    subplot(3,1,2), plot(spectrum2), grid on, zoom, title('Spectrum of signal 2 filtered');

    spectrum3 = abs(fft(demod_signal3));
    subplot(3,1,3), plot(spectrum3), grid on, zoom, title('Spectrum of signal 3 filtered');
end
```

PLOT:



SIGNAL REPRODUCED AFTER TRANSMISSION.

CODE:

```
%%
disp('STEP - 11, Play the reproduced signal after the transmission');

player4 = audioplayer(demod_signal1, 44100);
playblocking(player4);
playblocking(beep_player);

player5 = audioplayer(demod_signal2, 44100);
playblocking(player5);
playblocking(beep_player);

player6 = audioplayer(demod_signal3, 44100);
playblocking(player6);
```