DECEMBER 30, 2023

Communications Project

NAME	SEC	BN	CODE
1) Mustafa Tarek Salah	2	21	9211178
2) Mohammed Yasser	2	13	9211066

Table of Contents

Explanation of work

1) Reading Three Audio Signals:

The initial step involves reading three audio signals (audio1, audio2, and audio3) from the files "input1.wav," "input2.wav," and "input3.wav" using MATLAB's audioread function. The sampling frequency (FS) is also acquired during this process.

2) Resampling:

To ensure consistency in processing, the three audio signals are resampled to a common sample rate (sample_rate = 3 * FS). This action involves extending the signals to match the length of the longest audio clip and then resampling them using MATLAB's resample function.

3) Applying LPF on Each Signal:

A low-pass filter (lpf) is designed to limit the bandwidth of each audio signal. The cutoff frequency of this filter is set to 6000 Hz. The signals (audio1, audio2, and audio3) are then individually filtered using the designed low-pass filter.

4) Modulation

The modulated signals are obtained by multiplying each filtered audio signal with a cosine waveform. This modulation is performed for a frequency-division multiplexing (FDM) system, where each signal is modulated with a carrier frequency (FC) of 20 kHz, 40 kHz, and 60 kHz, respectively.

Explanation of work

5) Bandlimiting before Demodulation:

To isolate the upper sidebands and reject the lower sidebands, bandpass filters (bandpassFilter1, bandpassFilter2, and bandpassFilter3) are designed and applied to the modulated signals, These filters only allow the upper sidebands to pass through to achieve SSB modulation, effectively preparing the signals for transmission and summing them in one signal in to achieve frequency-division multiplexing (FDM) system.

6) Demodulation and Applying LPF:

To recover the original signals, the modulated signals are demodulated by multiplying them with cosine waveforms of the same carrier frequencies used in the modulation process. The resulting signals still contain both upper and lower sidebands so we apply to it low pass filter with 6000 Hz frequency to restore the original signal before modulation.

Results & Answers

1) Sampling Frequency Choice:

Selecting an appropriate sampling frequency for voice recording depends on the frequency content of the human voice and the Nyquist sampling theorem, which states that the sampling frequency must be at least **twice the maximum frequency of the signal of interest**.
The typical range of human speech is around 85 Hz to 255 Hz for male voices and 165 Hz to 255 Hz for female voices.
It's common to use a sampling frequency that is higher than the Nyquist rate. **A common choice is 44.1 kHz**, which is also the standard for audio CDs.

2) Cut-off Frequency Choice:

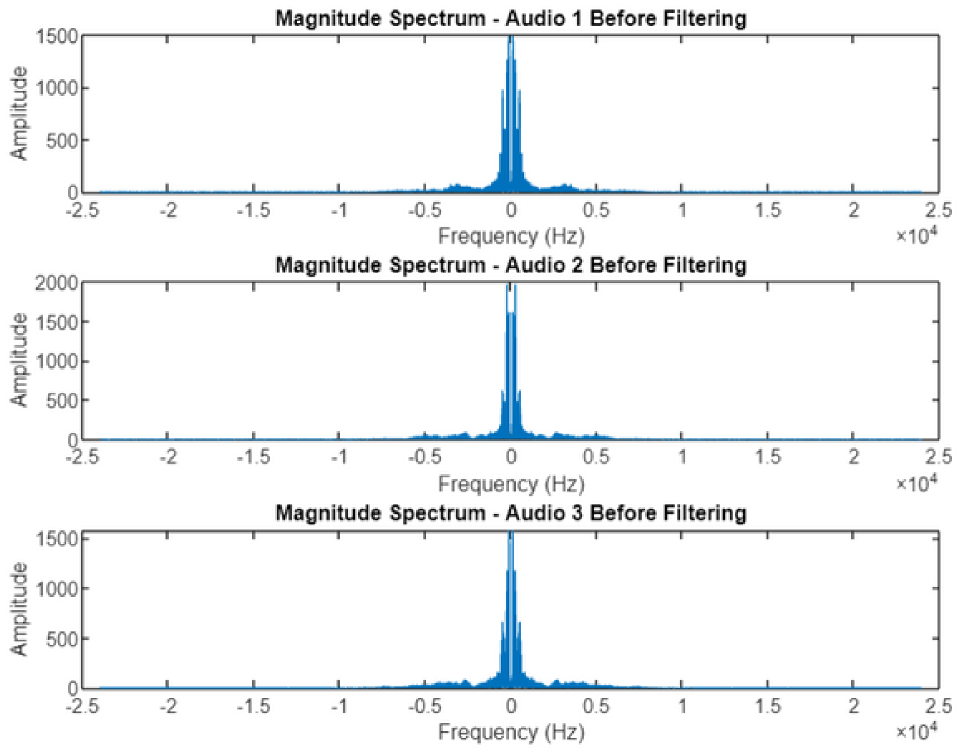
After many trials we choose it **6000 Hz** because this didn’t affect the original input signal

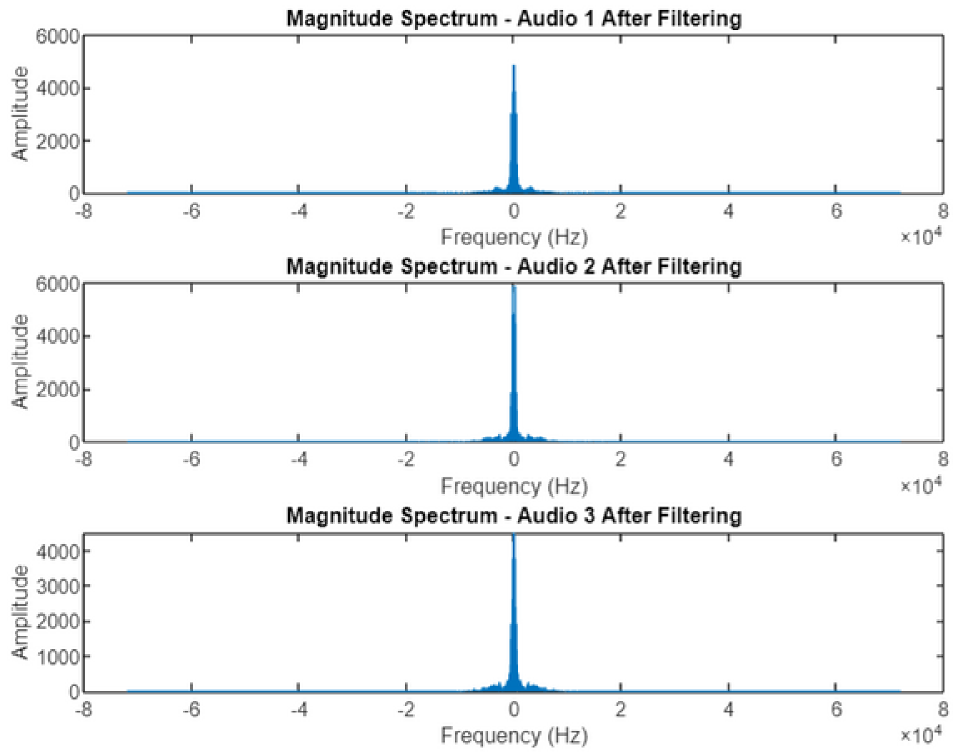
3) Carrier Frequency Choice:

The three signals are modulated with a **Fc = 20 kHz** ,**2*Fc= 40kHz** and **3*Fc= 60 kHz** carrier,
To satisfy the Nyquist criterion, the sampling frequency (Fs) should be greater than or equal to double the carrier frequency (2*Fc).
This criterion is applied to the third carrier frequency, which has the highest value.
Therefore, Fs >= 2 * (3 * Fc). Calculating the new sample rate as in code yields 3 * 44100 = 132.3 kHz.
Ensuring Fs >= 2 * (3 * Fc), the minimum value for the carrier frequency is determined as Fc <= Fs / 6, resulting in Fc <= 132.3 / 6 = 22.05 kHz.

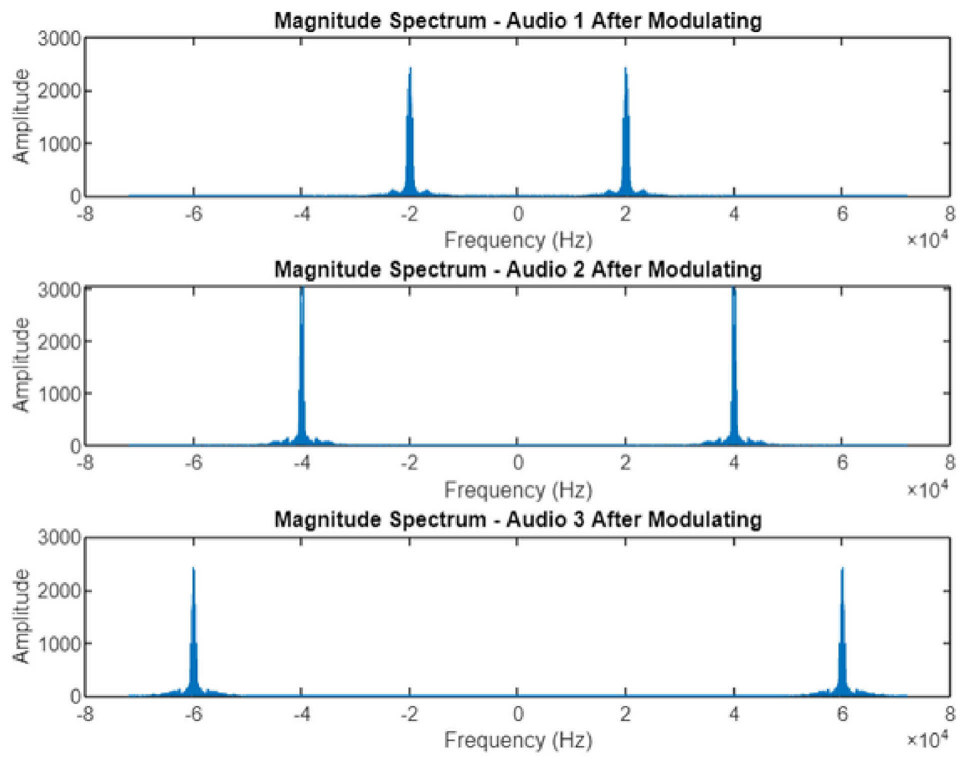
So, we choose **Fc = 20 kHz** which is lower than 22.05 kHz

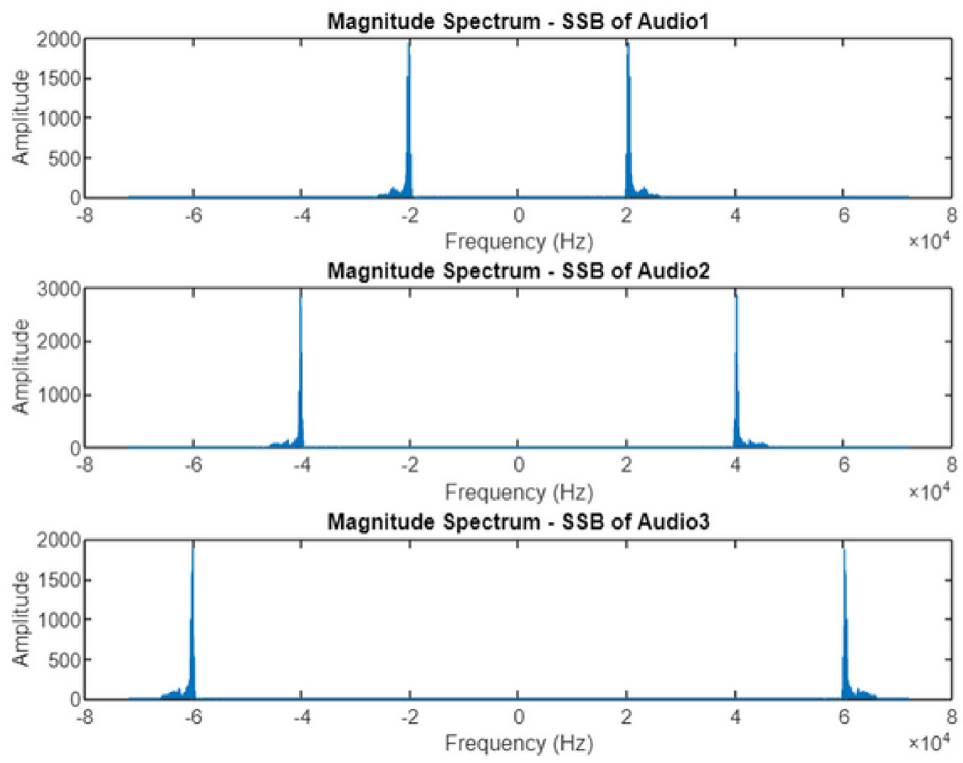
Figures


Magnitude Spectrum - Audio 1 Before Filtering
Amplitude
Frequency (Hz)
×10^4
Magnitude Spectrum - Audio 2 Before Filtering
Amplitude
Frequency (Hz)
×10^4
Magnitude Spectrum - Audio 3 Before Filtering
Amplitude
Frequency (Hz)
×10^4

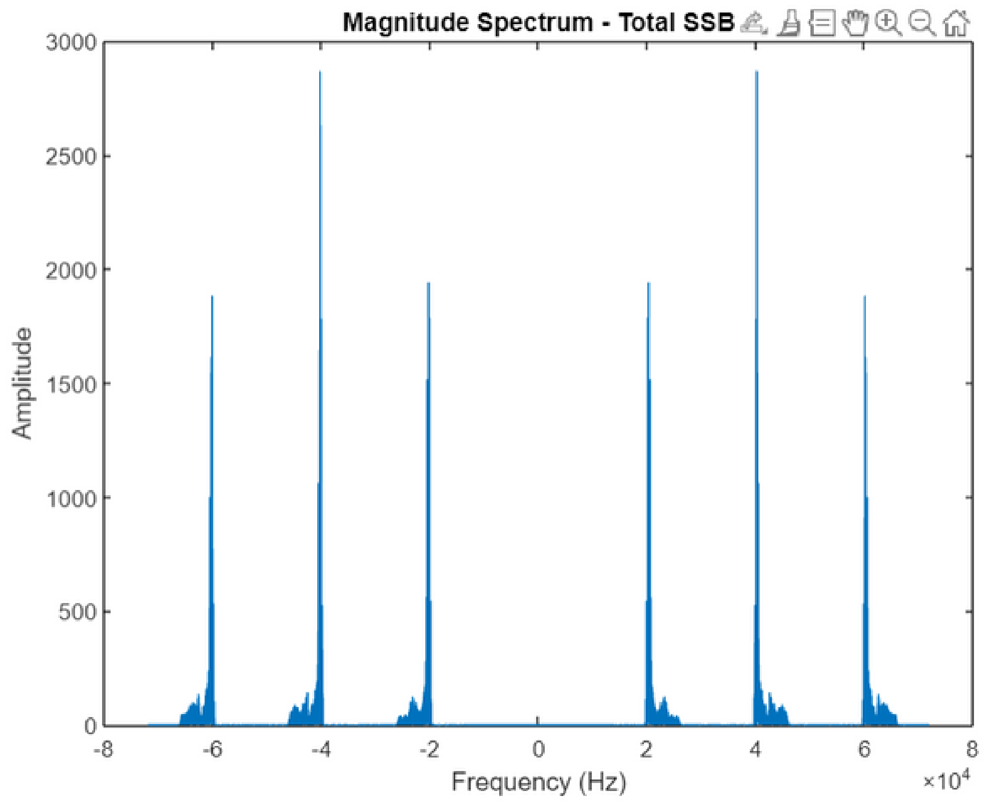

Magnitude Spectrum - Audio 1 After Filtering
Amplitude
Frequency (Hz)
×10^4
Magnitude Spectrum - Audio 2 After Filtering
Amplitude
Frequency (Hz)
×10^4
Magnitude Spectrum - Audio 3 After Filtering
Amplitude
Frequency (Hz)
×10^4

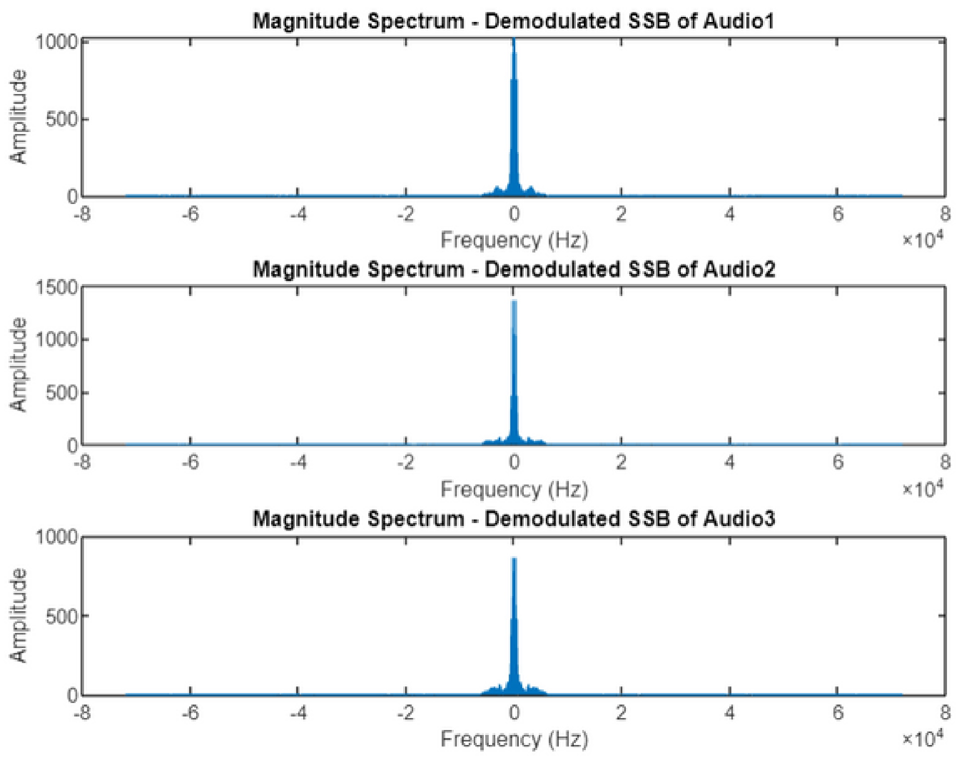
Figures


Magnitude Spectrum - Audio 1 After Modulating
Amplitude
3000
2000
1000
0
-8
-6
-4
-2
0
2
4
6
8
Frequency (Hz)
×10^4
Magnitude Spectrum - Audio 2 After Modulating
Magnitude Spectrum - Audio 3 After Modulating


Magnitude Spectrum - SSB of Audio1
Amplitude
2000
1500
1000
500
0
-8
-6
-4
-2
0
2
4
6
8
Frequency (Hz)
×10^4
Magnitude Spectrum - SSB of Audio2
3000
Magnitude Spectrum - SSB of Audio3

Figures


Magnitude Spectrum - Total SSB
Amplitude
3000
2500
2000
1500
1000
500
0
-8
-6
-4
-2
0
2
4
6
8
Frequency (Hz)
×10^4


Magnitude Spectrum - Demodulated SSB of Audio1
Amplitude
1000
500
0
Frequency (Hz)
×10^4
Magnitude Spectrum - Demodulated SSB of Audio2
Amplitude
1500
1000
500
0
Frequency (Hz)
×10^4
Magnitude Spectrum - Demodulated SSB of Audio3
Amplitude
1000
500
0
-8
-6
-4
-2
0
2
4
6
8
Frequency (Hz)
×10^4

Code

File 1: audioGeneration.m

```
%-------------1.Record voice segments and set sampling frequency-----------%

% Set the sampling frequency (Fs) appropriately.
Fs = 44100;

% Record three voice segments of 10 seconds each.
recording1 = audiorecorder(Fs, 16, 1); % 16 bits, 1 channel
disp('Start speaking for the first segment input.');
recordblocking(recording1, 10);
disp('Recording input1 is completed.');
input1 = getaudiodata(recording1);
audiowrite('input1.wav', input1, Fs);

recording2 = audiorecorder(Fs, 16, 1); % 16 bits, 1 channel
disp('Start speaking for the second segment input.');
recordblocking(recording2, 10);
disp('Recording input2 is completed.');
input2 = getaudiodata(recording2);
audiowrite('input2.wav', input2, Fs);

recording3 = audiorecorder(Fs, 16, 1); % 16 bits, 1 channel
disp('Start speaking for the third segment input.');
recordblocking(recording3, 10);
disp('Recording input3 is completed.');
input3 = getaudiodata(recording3);
audiowrite('input3.wav', input3, Fs);
```

Code

File 2: plotMagnitudeSpectrum.m

```
% Function to plot the spectrum of siganls.
function plotMagnitudeSpectrum(signal, fs, titleText)
 dfs = fs / length(signal);
 freqRange = (-fs / 2 : dfs : fs / 2 - dfs);
 ffreq = fft(signal);
 fSignal = fftshift(ffreq);
 plot(freqRange, abs(fSignal));
 title(titleText);
 xlabel('Frequency (Hz)');
 ylabel('Amplitude');
end
```

Code

File 3: main.m

```
%------------------------------------1.Choosing The Audio------------------------------------%
[audio1, FS] = audioread("input1.wav");
[audio2, FS] = audioread("input2.wav");
[audio3, FS] = audioread("input3.wav");

%----------------------------------2.Design and apply LPF filter------------------------------%

%Plotting the three audios before filtering.
figure(1);

subplot(3,1,1);
plotMagnitudeSpectrum(audio1, FS, 'Magnitude Spectrum - Audio 1 Before Filtering');

subplot(3,1,2);
plotMagnitudeSpectrum(audio2, FS, 'Magnitude Spectrum - Audio 2 Before Filtering');

subplot(3,1,3);
plotMagnitudeSpectrum(audio3, FS, 'Magnitude Spectrum - Audio 3 Before Filtering');

% Choosing the cutoff frequency.
cutoff_frequency = 6000;
sample_rate=3*FS;
% Determine the maximum length of the audio signals
maxLength = max([length(audio1), length(audio2), length(audio3)]);
audio1 = [audio1;zeros(maxLength -length(audio1),1)];
audio2 = [audio2;zeros(maxLength -length(audio2),1)];
audio3 = [audio3;zeros(maxLength -length(audio3),1)];

% Resampling the signals.
audio1 = resample(audio1, sample_rate, FS);
audio2 = resample(audio2, sample_rate, FS);
audio3 = resample(audio3, sample_rate, FS);

% Design a low-pass filter for signals.
lpf = designfilt('lowpassfir', 'FilterOrder', 200, ...
'CutoffFrequency', cutoff_frequency, 'SampleRate', FS);

% Apply the filter to audio1.
filtered_input1 = filter(lpf, audio1);

% Apply the filter to audio2.
filtered_input2 = filter(lpf, audio2);

% Apply the filter to audio3.
filtered_input3 = filter(lpf, audio3);
```

Code

File 3: main.m

```
%Plotting the three audios after filtering.
figure(2);

subplot(3,1,1);
plotMagnitudeSpectrum(filtered_input1, sample_rate, 'Magnitude Spectrum - Audio 1
After Filtering');

subplot(3,1,2);
plotMagnitudeSpectrum(filtered_input2, sample_rate, 'Magnitude Spectrum - Audio 2
After Filtering');

subplot(3,1,3);
plotMagnitudeSpectrum(filtered_input3, sample_rate, 'Magnitude Spectrum - Audio 3
After Filtering');

%----------------------------------3.Perform SSB Modulation---------------------------------%
%y(t) = (x(t)) cos ωt
FC=20000;

% Create a time vector with the maximum length.
t = linspace(0,maxLength*3/sample_rate,maxLength*3);

% Modulate the three audios.
modulatedSignal1 = filtered_input1'.*cos(2*pi*(FC)*t);
modulatedSignal2 = filtered_input2'.*cos(2*pi*(2*FC)*t);
modulatedSignal3 = filtered_input3'.*cos(2*pi*(3*FC)*t);

% Plotting the three audios after modulating.
figure(3);

subplot(3,1,1);
plotMagnitudeSpectrum(modulatedSignal1, sample_rate, 'Magnitude Spectrum - Audio 1
After Modulating');

subplot(3,1,2);
plotMagnitudeSpectrum(modulatedSignal2, sample_rate, 'Magnitude Spectrum - Audio 2
After Modulating');

subplot(3,1,3);
plotMagnitudeSpectrum(modulatedSignal3, sample_rate, 'Magnitude Spectrum - Audio 3
After Modulating');
```

Code

File 3: main.m

```
% Design the bandpass filters.
bandpassFilter1 = designfilt('bandpassfir','FilterOrder', 700,'CutoffFrequency1',...
20000,'CutoffFrequency2', 26000,'SampleRate', sample_rate);
bandpassFilter2 = designfilt('bandpassfir','FilterOrder', 700,'CutoffFrequency1',...
40000,'CutoffFrequency2', 46000,'SampleRate', sample_rate);
bandpassFilter3 = designfilt('bandpassfir','FilterOrder', 700,'CutoffFrequency1',...
60000,'CutoffFrequency2', 66000,'SampleRate', sample_rate);

SSB1=filter(bandpassFilter1, modulatedSignal1);
SSB2=filter(bandpassFilter2, modulatedSignal2);
SSB3=filter(bandpassFilter3, modulatedSignal3);

% Plotting the three audios after modulating.
figure(4);

subplot(3,1,1);
plotMagnitudeSpectrum(SSB1, sample_rate, 'Magnitude Spectrum - SSB of Audio1');

subplot(3,1,2);
plotMagnitudeSpectrum(SSB2, sample_rate, 'Magnitude Spectrum - SSB of Audio2');

subplot(3,1,3);
plotMagnitudeSpectrum(SSB3, sample_rate, 'Magnitude Spectrum - SSB of Audio3');

% Summing the siganls to make FDM system.
SSB_total =SSB1+SSB2+SSB3;

figure(5);
plotMagnitudeSpectrum(SSB_total, sample_rate, 'Magnitude Spectrum - Total SSB');

%--------------------------------4.Perform SSB Demodulation-------------------------------%
demodulatedAudio1 = SSB_total.*cos(2*pi*(FC)*t);
demodulatedAudio2 = SSB_total.*cos(2*pi*(2*FC)*t);
demodulatedAudio3 = SSB_total.*cos(2*pi*(3*FC)*t);

% Design a low-pass filter for audios.
lpf2 = designfilt('lowpassfir', 'FilterOrder', 200, 'CutoffFrequency', cutoff_frequency,
'SampleRate', sample_rate);

% Apply the filter to demodulatedAudio1.
output1 = filter(lpf2, demodulatedAudio1);
audiowrite('output1.wav', output1, sample_rate);
```

Code

File 3: main.m

```
% Apply the filter to demodulatedAudio2.
output2 = filter(lpf2, demodulatedAudio2);
audiowrite('output2.wav', output2, sample_rate);

% Apply the filter to demodulatedAudio3.
output3 = filter(lpf2, demodulatedAudio3);
audiowrite('output3.wav', output3, sample_rate);

% Plotting the outputs.
figure(6);

subplot(3,1,1);
plotMagnitudeSpectrum(output1, sample_rate, 'Magnitude Spectrum - Output1');

subplot(3,1,2);
plotMagnitudeSpectrum(output2, sample_rate, 'Magnitude Spectrum - Output2');

subplot(3,1,3);
plotMagnitudeSpectrum(output3, sample_rate, 'Magnitude Spectrum - Output3');
```