Cairo University

Faculty of Engineering

Dept. of Electronics and Electrical Communications

Second Year

Signal Project

Student Name	Section	B.N.
عبد الرحمن احمد محمد عبد اللطيف	2	32
اسلام فتحي محمد سليمان	1	26

Part1: Image filtering and restoration

A&B. Explanation & results and answers:

a)

Read the image and show it on the grayscale.
Show the 3 components of the image (red, green, blue).

b)

1. Do edge detection by using the kernel in this method

$$Kx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Ky = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Kx: the center column is neutral, the left common emphasizes the dark region to the left and the right column emphasizes the bright region to the right.
Ky: the center row is neutral; the top row emphasizes the dark region to the top and the bottom column emphasizes the bright region to the bottom.
Convolution red, green, and blue with kernel in each x and y direction.
Combining three color channels (red, green, and blue) into a single RGB image.
The result is an RGB color.

2. Sharping process:

$$kx\_sharp = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

It deducts the contributions of its neighbors and highlights the core pixel's contribution five times.
Applying convolution to each color channel (red, green, and blue) with the sharpening kernel.
Combining three color channels (red, green, and blue) into a single RGB image.
The result is an RGB color.

3. Blurring image:
k_average = ones (7) / 49.
This kernel is a 7*7 matrix each element = 1/49 and it's a simple average or smoothing operation with a 7x7 neighborhood.
Applying convolution to each color channel (red, green, and blue) with the average kernel,
Combining three color channels (red, green, and blue) into a single RGB image.
The result is an RGB color.
4. Blurring motion image:
Kernel blur motion horizontal = zeros (25).
K blur motion horizontal (13, :) = 1/25.
We use this kernel as it 25*25 matrix each element = 0 while the center row elements = 1/25 and it effectively averages the pixel values along each horizontal line.
Applying convolution to each color channel (red, green, and blue) with the chosen kernel.
Combining three color channels (red, green, and blue) into a single RGB image.
The result is an RGB image.

c) restoring the original image from the motion blur image:

First, we do motion blur to the image as mentioned above, after that, we convert the blurred motion image and the kernel used to the frequency domain by fft2.

We add a constant called epsilon to prevent division by zero.

Do a deconvolution operation In the frequency domain, by dividing each element of the blurred motion image by the corresponding element of (motion kernel ft + epsilon).

We do an inverse Fourier Transform to the restored image.

Display the restored image.

C. figures:

Figure1: image on grayscale

Figure2: image edge detection

Figure3: image sharping

Figure 4: image blur

Figure5: image blur motion

Figure6: image restoring

Part2: Communication system simulation

A&B Explanation & results and answers:

a&b&c.

1. In the beginning, we recorded two audios, and the audio will contain noise, which is a very high frequency in the range of 5000 to 10,000 hertz, audio recording object with a sampling rate of 44100 Hz this parameter represents the sampling rate, 44100 Hz is a common standard for audio recordings, a bit depth is represent the number of bits per sample, indicating the resolution of each audio sample, the value 16 indicates a standard setting for CD-quality audio, and a single audio channel.
2. I use function called **getaudiodata** to extract audio data from a recorded audio, this function retrieves the recorded audio samples as a column vector.
3. I use **audiowrite** function to store the recorded audio.
4. We generate waveform plot of audio signal; this type of plot visualizes the changes in the amplitude of the audio signal over time.
5. perform some analysis on two audio signals
   L1 and L2 store the lengths of the audio signals, k1 and k2 store indices for the audio signals.
   Performing FFT (Fast Fourier Transform)
   Plotting results:
   Three separate plots are created:
   The first plot displays the magnitude of the FFT of ad against the indices.
   The second plot visualizes the magnitude against frequency for ad.
   The third plot shows the shifted FFT of ad.
6. We designed a Low pass filter with Fpass = 4000 and Fstop =5000 based on the recorded voice and wanted range of frequency; we loaded the stored low pass filter from .mat file
7. We applied low pass filter to the two recorders and repeat step 5 to the filtered audio

d&e.

8. **Modulation:** setting the modulation parameters to be 5000,15000 to avoid over write, t1&t2 are time vectors created for modulation based on the lengths of filtered audios.
   We modulated the signals with cosine functions at the specified carrier frequency, then calculate Fast Fourier Transform to the modulated signal.
9. trans1 and trans2 perform frequency shifting on the FFT results, f is a frequency vector, fun represents the sum of the magnitudes of the shifted FFTs for both modulated signals, then plots the frequency spectrum (f) against the combined magnitude spectrum (fun), The plot appears to show the frequency spectrum of the sum of the modulated signals. The frequencies are shifted using FFT, and the result is visualized in the plot.
   The first pair of subplots shows the modulated signals in the time domain
   The second pair of subplots displays the magnitude spectrum of each modulated signal in the frequency domain.
10. **Demodulation:** are obtained by multiplying the modulated signals with a cosine function at the carrier frequencies, this process effectively shifts the modulated signals back to the baseband.
11. The low-pass filter is applied to the demodulated signals using the filter function, the factor of 2 is applied to scale the amplitude of the demodulated signals.
    This sequence of demodulation and low-pass filtering is a common process in analog communication systems to retrieve the original baseband signal, the low-pass filter is used to remove high-frequency components introduced during modulation, leaving the demodulated audio signals.
12. The first subplot shows the demodulated and filtered signal in the time domain, the second subplot displays the magnitude spectrum of the demodulated signal against frequency, the third subplot shows the shifted magnitude spectrum of the demodulated signal.
13. at the end we saved the output signal.

C. figures:

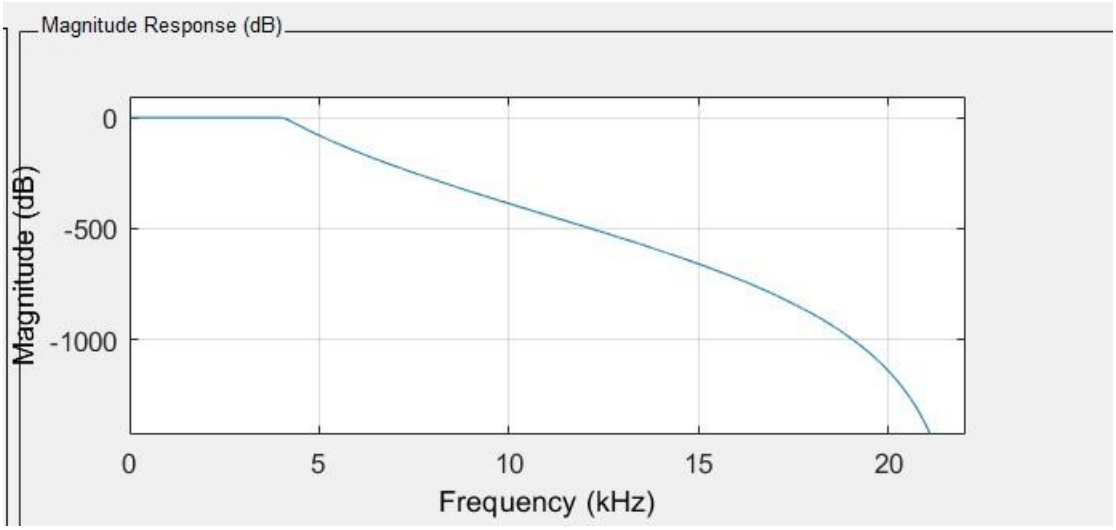

Figure7: frequency response of the filter

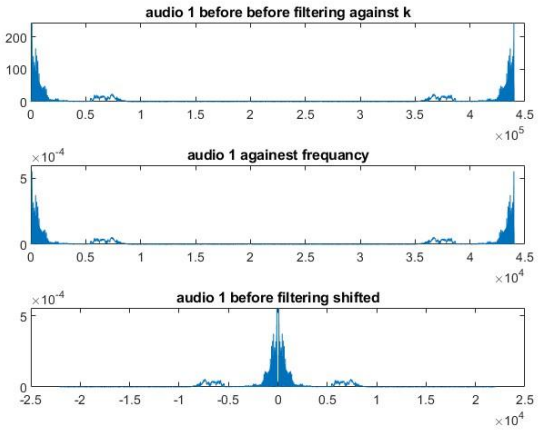

Figure 8 unfiltered audio1

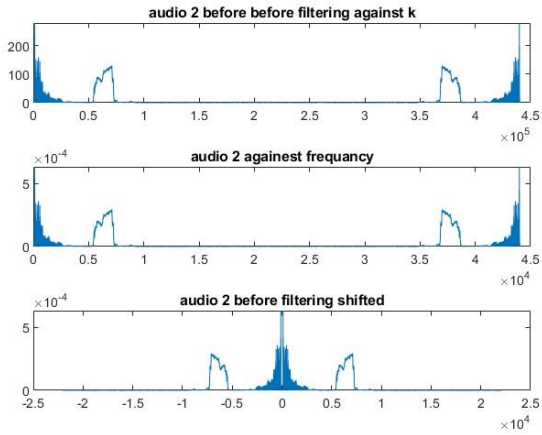

Figure9: unfiltered audio2

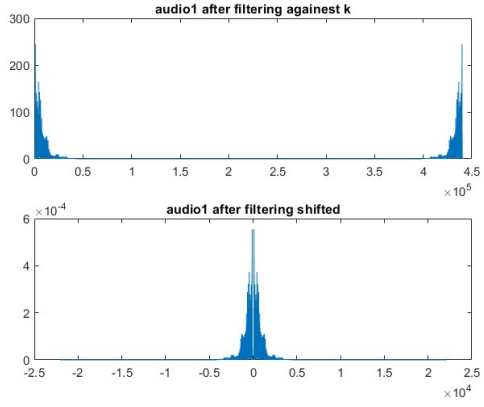

Figure10: filtered audio1

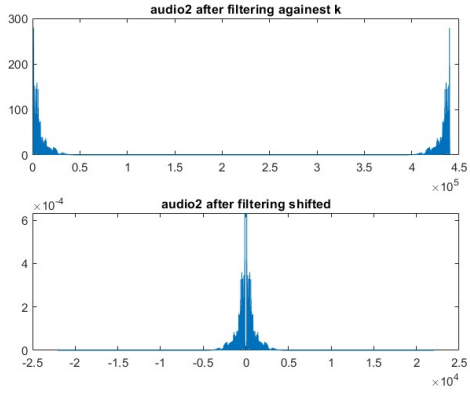

Figure11: filtered audio2

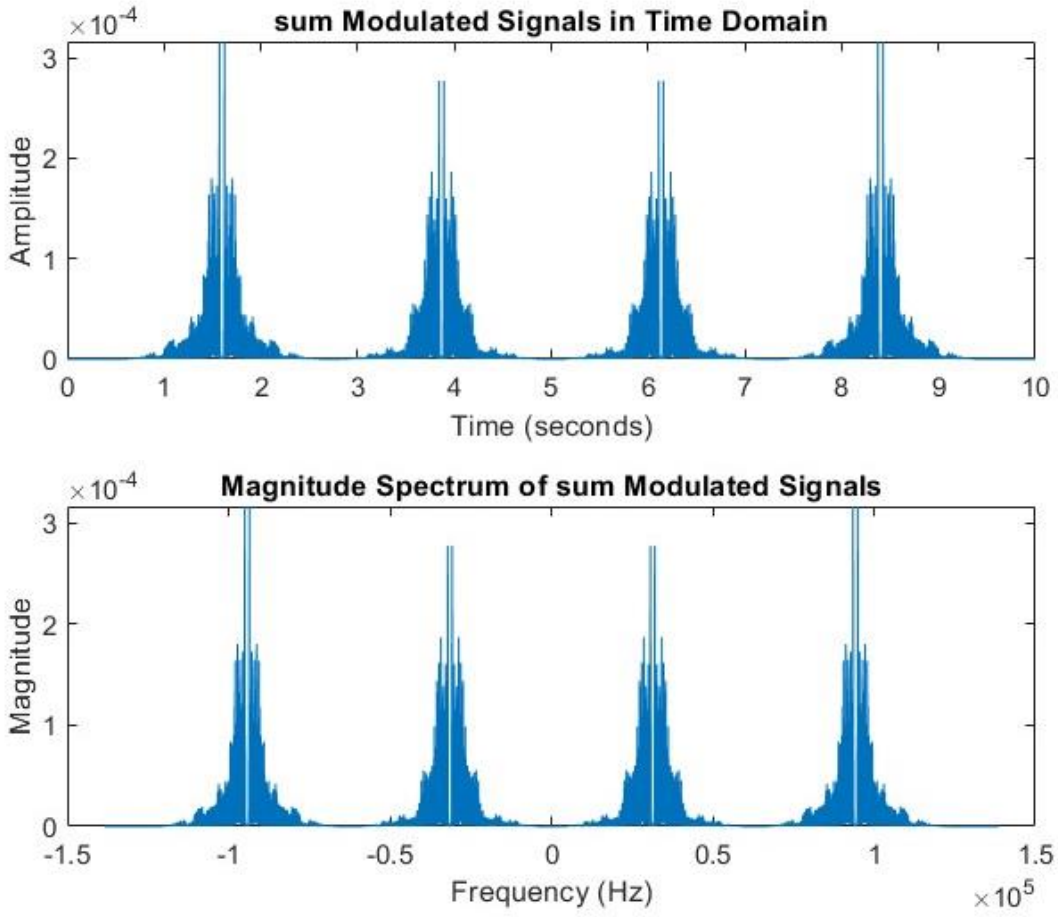

Figure12: magnitude spectrum of the transmitted signal

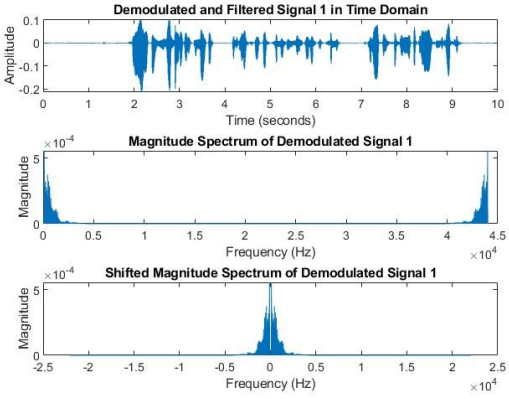

Figure13: demodulated audio1

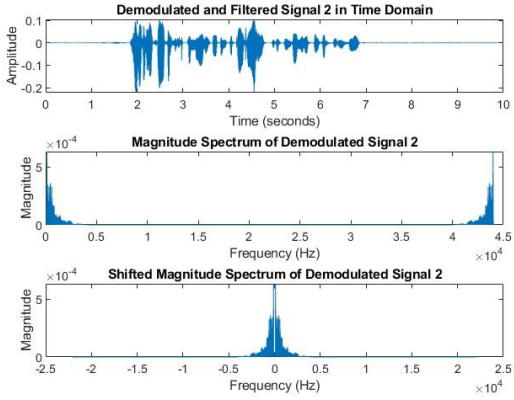

Figure 14: demodulated audio2

D. Codes:

Part1:

```
clc ;

clear;


% 1/ read the image and show it in original and gray scale



[ai,bi]=uigetfile;

im=strcat(bi,ai);

var_img=imread(im);

imshow (var_img) ;

title ('original image');

img_gray = rgb2gray(var_img);

imshow (img_gray);

title ('gray image');

imwrite(img_gray,'image1.png')


% 2/ define the r,g,b component



red = var_img (:,:,1);

green = var_img (:,:,2);

blue = var_img (:,:,3);


% 3/ edge detection process and show it
```

```
% kernel of vertical and horizontal
% ky is transpose of kx

kx = [-1 0 1 ;
      -2 0 2 ;
      -1 0 1 ; ];

ky = [ -1 -2 -1;
        0 0 0 ;
        1 2 1 ;];


%conv of r,g,b in x and y direction

rx = conv2(red,kx,'same');
gx = conv2(green,kx,'same');
bx = conv2(blue,kx,'same');


ry = conv2(red,ky,'same');
gy = conv2(green,ky,'same');
by = conv2(blue,ky,'same');


edge_r = sqrt(rx.^2 + ry .^2);
edge_g = sqrt(gx.^2 + gy .^2);
edge_b = sqrt(bx.^2 + by .^2);
```

```
img_edge_detection = uint16(cat(3,edge_r,edge_g,edge_b));

imshow (img_edge_detection);
title ('edge detection ');

imwrite(img_edge_detection,'image2.png')

% 4/ sharping process and show it

%kernel of sharping

kx_sharp = [  0 -1 0 ;
             -1 5 -1 ;
              0 -1 0 ; ];

%conv of r,g,b

colur_red = conv2(red,kx_sharp,'same');
colur_green = conv2(green,kx_sharp,'same');
colur_blue = conv2(blue,kx_sharp,'same');

img_sharping = uint16(cat(3,colur_red,colur_green,colur_blue));

imshow (img_sharping);
```

```
title ('img sharping ');

imwrite(img_sharping,'image3.png');

% 5/ bluring process with average karnel and show it

%average kernel for bluring is a 7*7 matrix all = 1/49

k_average = ones(7) / 49 ;

%conv of r,g,b

colur_red1 = conv2(red,k_average,'same');
colur_green1 = conv2(green,k_average,'same');
colur_blue1 = conv2(blue,k_average,'same');

img_blur = uint16(cat(3,colur_red1,colur_green1,colur_blue1));

imshow (img_blur);
title ('img blur');

imwrite(img_blur,'image4.png');

% 6/ bluring motion process with horizontal axes and show it
```

```
%kernel for blur horizontal is a 25*25 matrix all of them is zero
%while the middle row = 1/25

k_blur_motion_horizontal = zeros(25);
k_blur_motion_horizontal (13,:) = 1/25;



%conv of r,g,b

colur_red2 = conv2(red,k_blur_motion_horizontal,'same');
colur_green2 = conv2(green,k_blur_motion_horizontal,'same');
colur_blue2 = conv2(blue,k_blur_motion_horizontal,'same');

img_blur_motion = uint16(cat(3,colur_red2,colur_green2,colur_blue2));

imshow (img_blur_motion);
title ('img blur motion');

imwrite(img_blur_motion,'image5.png');



% 7/ restoring the original image

% do blur motion to the image

original_image=var_img;
motion_kernel= ones(1,200)/200;
```

```
blur_motion_image(:,:,1) = (conv2(original_image(:,:,1),motion_kernel));

blur_motion_image(:,:,2) = (conv2(original_image(:,:,2),motion_kernel));

blur_motion_image(:,:,3) = (conv2(original_image(:,:,3),motion_kernel));


% appling fourier transform the the motionned image and kernel


motion_kernel_ft=fft2(motion_kernel,size(blur_motion_image,1),size(blur_motion_image,2));


blur_motion_image_ft=fft2(blur_motion_image);


%add a constant epsilon and Do a deconvolution operation In the frequency domain


epsilon=.00001;

image_restored_ft =blur_motion_image_ft ./(motion_kernel_ft+epsilon);


% appling inverse fourier transform to the restored image


restored_image = abs(ifft2(image_restored_ft));


restored_image =restored_image(1:size(original_image,1),1:size(original_image,2),1:size(original_image,3));

% blur_motion_image
=motional_image(1:size(original_image,1),25:size(original_image,2),1:size(original_image,3));




  imshow(uint16(restored_image));

  title ('restored_image');

 imwrite(restored_image,'image6.png');
```

Part2:

```
clc;
clear;
%recording 2 audios
audio1=audiorecorder(44100,16,1);
disp("start speaking1");
recordblocking(audio1,10);
disp("end of recording1");
disp("wait");
pause(2);
audio2 =audiorecorder(44100,16,1);
disp("start speaking2");
recordblocking(audio2,10);
disp("end of recording2");

%get the audio data
ad1 = getaudiodata(audio1);
ad2 = getaudiodata(audio2);

% storing original
audiowrite('input1.wav', ad1, 44100);
audiowrite('input2.wav', ad2, 44100);



% play the 2 audios
y1=audioplayer(ad1,44100);
% play(y1);
```

```
% pause(11);
y2=audioplayer(ad2,44100);
% play(y2);
% pause(11);

% ploting
figure
subplot(2,1,1);
plot(ad1);
subplot(2,1,2);
plot(ad2);



L1= length(ad1);
L2= length(ad2);
k1=0:L1-1;
ad1_k= fft (ad1,L1);
ad1_f= (0 :L1-1) * (44100/L1);
ad1_shift=(-L1/2:L1/2-1)*(44100/L1);



k2=0:L2-1;
ad2_k= fft (ad2,L2);
ad2_f= (0 :L2-1) * (44100/L2);
ad2_shift=(-L2/2:L2/2-1)*(44100/L2);

% plot audio1 against k&Hz and plot shifted audio1
figure;
```

```
subplot(3,1,1);
plot(k1,abs(ad1_k));
title ('audio 1 before before filtering against k');
subplot(3,1,2);
plot(ad1_f,abs(ad1_k)/L1);
title ('audio 1 againest frequancy');
subplot(3,1,3);
plot(ad1_shift,abs(fftshift(ad1_k))/L1);
title ('audio 1 before filtering shifted');

% plot audio2 against k&Hz and plot shifted audio2
figure;
subplot(3,1,1);
plot(k2,abs(ad2_k))   ;
title ('audio 2 before before filtering against k');
subplot(3,1,2);
plot(ad2_f,abs(ad2_k)/L2)      ;
title ('audio 2 againest frequancy');
subplot(3,1,3);
plot(ad2_shift,abs(fftshift(ad2_k))/L2)
title ('audio 2 before filtering shifted');

% load the filter
load('lp.mat');

% low pass filter
filtered_audio1 = filter(lp, ad1);
y3=audioplayer(filtered_audio1,44100);
```

```
% play(y3);
% pause(11);

filtered_audio2 = filter(lp, ad2);
y4=audioplayer(filtered_audio2,44100);
% play(y4);
% pause(11);

% plot filtered audio1 against k&Hz and plot shifted filtered audio1

Lf1=length(filtered_audio1);
filtered_audio1_f=fft(filtered_audio1,Lf1);
figure;
subplot(2,1,1);
plot(k1,abs(filtered_audio1_f));
title('audio1 after filtering againest k');
filterd_audio_sh=(- L1/2 : L1/2 - 1)*44100 /L1;
subplot(2,1,2);
plot(filterd_audio_sh,abs(fftshift(filtered_audio1_f))/ L1);
title ('audio1 after filtering shifted');

% plot filtered audio2 against k&Hz and plot shifted filtered audio2
Lf2=length(filtered_audio2);
filtered_audio2_f=fft(filtered_audio2,Lf2);
figure;
subplot(2,1,1);
plot(k2,abs(filtered_audio2_f));
title('audio2 after filtering againest k');
```

```
filterd_audio2_sh=(- L2/2 : L2/2 - 1)*44100 /L2;
subplot(2,1,2);
plot(filterd_audio2_sh,abs(fftshift(filtered_audio2_f))/ L2);
title ('audio2 after filtering shifted');



% Modulation parameters
Fc_mod1 = 5000; % 5 kHz for first signal
Fc_mod2 = 15000; % 15 kHz for second signal

% Time vectors for modulation
t1 = (0:length(filtered_audio1)-1)/44100;
t2 = (0:length(filtered_audio2)-1)/44100;

% Modulate both signals
modulated_signal1 = filtered_audio1 .* cos(2 * pi * Fc_mod1 * t1)';
modulated_signal2 = filtered_audio2 .* cos(2 * pi * Fc_mod2 * t2)';

% FFT for modulated signals
fft_modulated1 = fft(modulated_signal1);
fft_modulated2 = fft(modulated_signal2);

trans1=abs(fftshift(fft(modulated_signal1)));
trans2=abs(fftshift(fft(modulated_signal2)));

f = 2*pi* (-44100/2 : 44100/Lf2 : 44100/2 - 44100/Lf2);
fun = trans1/Lf1 + trans2/Lf2;
```

```
figure
subplot(2,1,1);
plot(t1, fun);
title('sum Modulated Signals in Time Domain');
xlabel('Time (seconds)');
ylabel('Amplitude');

figure;
subplot(2,1,2);
plot(f, fun);
title('Magnitude Spectrum of sum Modulated Signals');
xlabel('Frequency (Hz)');
ylabel('Magnitude');



% Demodulation
demodulated_signal1 = modulated_signal1 .* cos(2 * pi * Fc_mod1 * t1)';
demodulated_signal2 = modulated_signal2 .* cos(2 * pi * Fc_mod2 * t2)';

% Apply the low-pass filter to the demodulated audio
demodulated_filtered_audio1 = 2 * filter(lp, demodulated_signal1);
demodulated_filtered_audio2 = 2 * filter(lp, demodulated_signal2);



% Plotting Magnitude Spectrum for Demodulated Signal


Ld1= length(demodulated_filtered_audio1);
```

```
kd1=0:Ld1-1;
demodulated_filtered_audio1_k= fft (demodulated_filtered_audio1,Ld1);
demodulated_filtered_audio1_f= (0 :Ld1-1) * (44100/Ld1);
demodulated_filtered_audio1_shift=(-Ld1/2:Ld1/2-1)*(44100/Ld1);
figure;
subplot(3,1,1);
plot(t1, demodulated_filtered_audio1);
title('Demodulated and Filtered Signal 1 in Time Domain');
xlabel('Time (seconds)');
ylabel('Amplitude');

subplot(3,1,2);
plot(demodulated_filtered_audio1_f, abs(demodulated_filtered_audio1_k)/Ld1);
title('Magnitude Spectrum of Demodulated Signal 1');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,1,3);
plot(demodulated_filtered_audio1_shift, abs(fftshift(demodulated_filtered_audio1_k))/Ld1);
title('Shifted Magnitude Spectrum of Demodulated Signal 1');
xlabel('Frequency (Hz)');
ylabel('Magnitude');



% Plotting Results for the second demodulated signal
Ld2= length(demodulated_filtered_audio2);
kd2=0:Ld2-1;
demodulated_filtered_audio2_k= fft (demodulated_filtered_audio2,Ld2);
```

```
demodulated_filtered_audio2_f= (0 :Ld2-1) * (44100/Ld2);
demodulated_filtered_audio2_shift=(-Ld2/2:Ld2/2-1)*(44100/Ld2);

figure;
subplot(3,1,1);
plot(t2, demodulated_filtered_audio2);
title('Demodulated and Filtered Signal 2 in Time Domain');
xlabel('Time (seconds)');
ylabel('Amplitude');

subplot(3,1,2);
plot(demodulated_filtered_audio2_f, abs(demodulated_filtered_audio2_k)/Ld2);
title('Magnitude Spectrum of Demodulated Signal 2');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,1,3);
plot(demodulated_filtered_audio2_shift, abs(fftshift(demodulated_filtered_audio2_k))/Ld2);
title('Shifted Magnitude Spectrum of Demodulated Signal 2');
xlabel('Frequency (Hz)');
ylabel('Magnitude');



% Play the demodulated
y5=audioplayer(demodulated_filtered_audio1,44100);
% play(y5);
% pause(11);
y6=audioplayer(demodulated_filtered_audio2,44100);
```

```
% play(y6);
% pause(11);



audiowrite('output1.wav', demodulated_filtered_audio1, 44100);
audiowrite('output2.wav', demodulated_filtered_audio2, 44100);
```