Cairo University

Faculty of Engineering

Dept. of Electronics and Electrical Communications

Second Year

Signals

ELC 2030

Project report

Student Name	Section	ID
احمد عمرو علي	1	9220069
بلال رمضان حلمي	1	9220205

I.

A.

In this task, we first extracted each component from the image, then, in order to display it we concatenated each component of them with two zero matrices to replace the other two components.

Fig(1):red, green & blue components.

B.

In this task (the four kernel tasks), we extracted the three components then applied convolution in 2D on each of them with the edge detection kernel matrix then concatenated the three components in order to display the full image after the effect of the kernel.

1)

The kernel in this task is

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ -1 & -1 & 8 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

We chose this kernel because it must be a summation of 0 for all its elements in order to make the non-edge pixels black and the edged pixels colored. And we multiplied the center pixel by 8 and subtracted the 2 up, down, right and left pixels to apply a good edge detection.

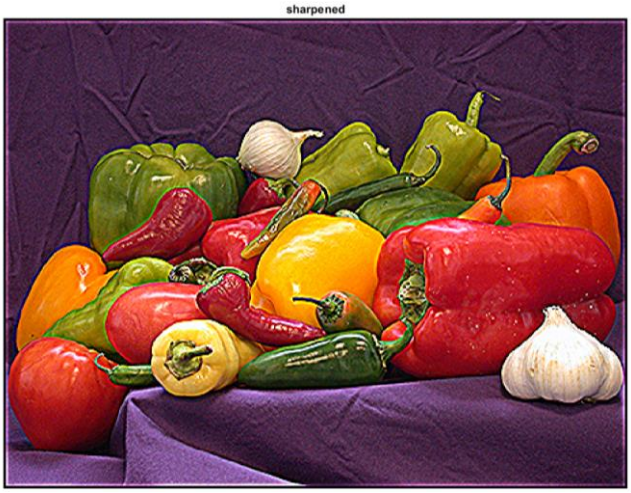
Fig(2):edge detected image.

2)

The kernel in this task is

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ -1 & -1 & 9 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Notice it's as the previous kernel but we added 1 to the center element. That's because the sharpening is the original photo plus edge detected photo. Notice also that the summation of its elements is 1 in order to keep the same illumination of the original photo.



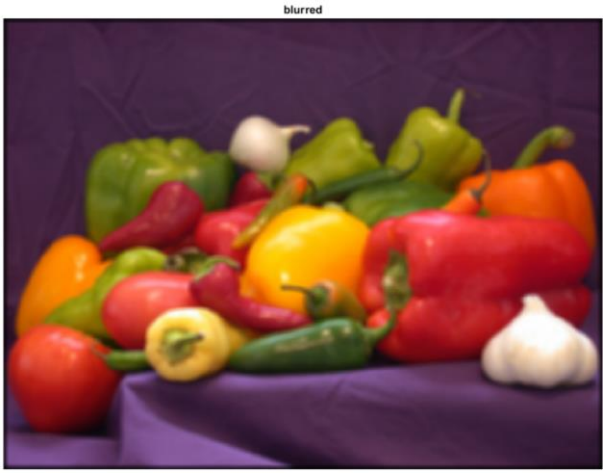
Fig(3):sharpened image.

3)

The kernel in this task is

$$\begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix}$$

We chose this kernel because blurring is simply averaging with the surrounding pixels so we multiply all of them by $\frac{1}{25}$ because our kernel is $5x5$.



Fig(4):blurred image.

4)

The kernel in this task is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We chose this kernel because horizontal motion blurring is averaging with the surrounding horizontal pixels so we multiply all the elements of the center row by $\frac{1}{5}$ because the number of columns is 5.

NOTE: if we chose another row, the image will be the same but shifted up or down.

Fig(5):horizontal motion blurred image.

C.

In this task, we made a large matrix with the same size as the horizontal motion blurred image then we divided the image by it in the frequency domain in order to reverse kernel's effect then we noticed that the image is black in the last 4 pixels from the right and down, so we removed the black pixels in order to make it a nicer look image.

Fig(6):reconstructed image.

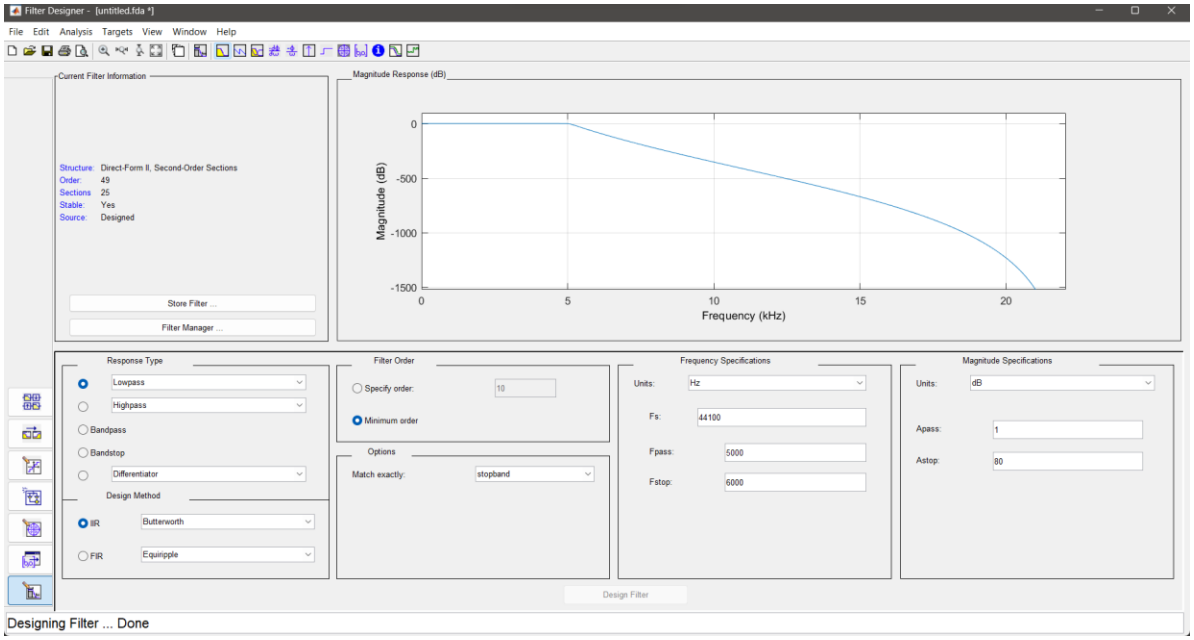
II

a.

In this part we start by recording the audio from MATLAB with sampling frequency = 44100 Hz and bit depth = 16, we chose a large f_s in order to make it greater than Nyquist rate so that the sample can be reconstructed and 16 bits depth in order to store each value in 16 bits so that the sampling quality is better. Then we read this audio in the MATLAB and use fast Fourier transform to convert the data from time domain to frequency domain.

b.

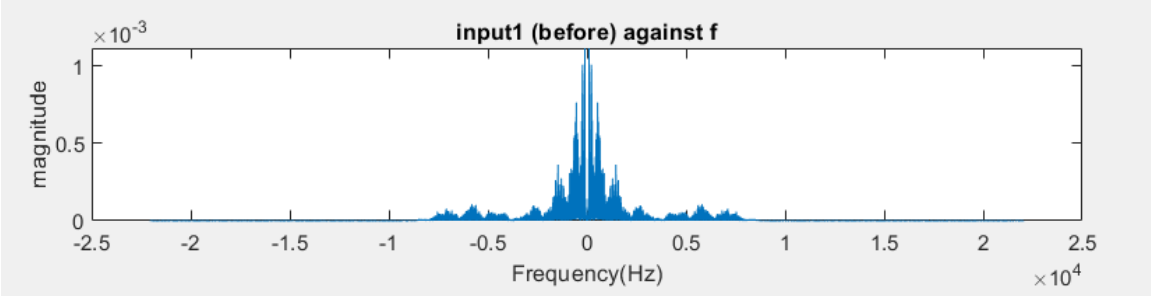
After that we use the filter designer to make our filter as follows:



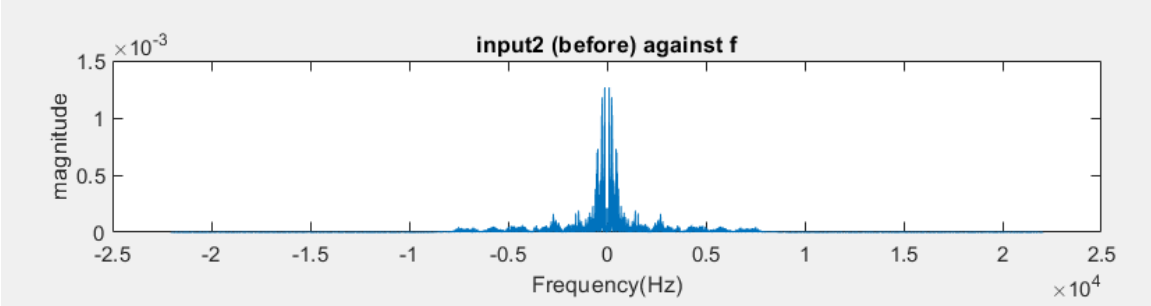
Fig(7):filter designer.

c.

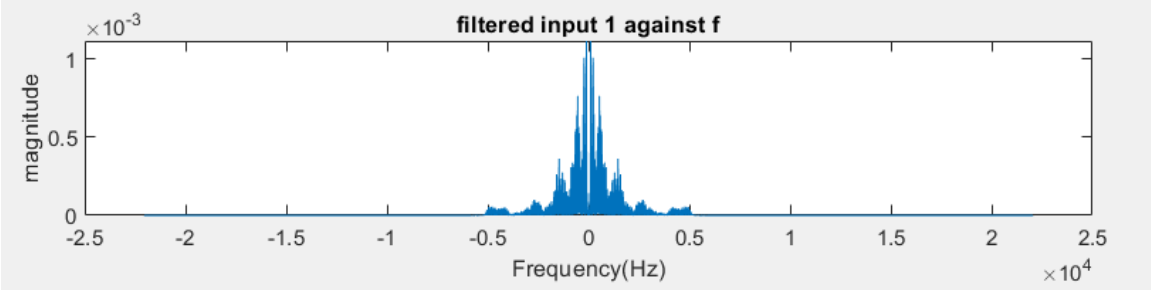
Then we export it as an object in the workspace and use this filter to filter the audio and then use FFT again and plot it before and after filtering as follows:



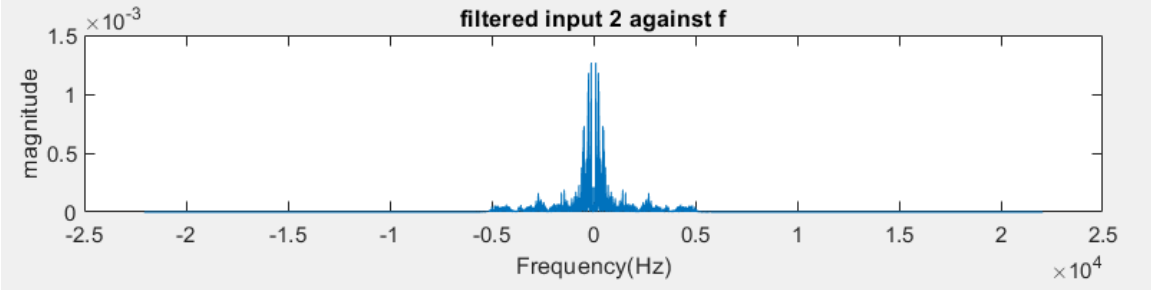
Fig(8):input1 against f.



Fig(9):input2 against f.



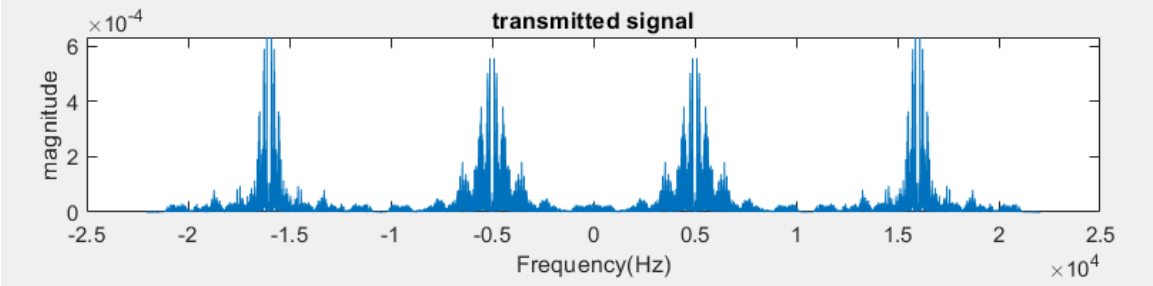
Fig(10):filtered input1 against f.



Fig(11):filtered input1 against f.

d.

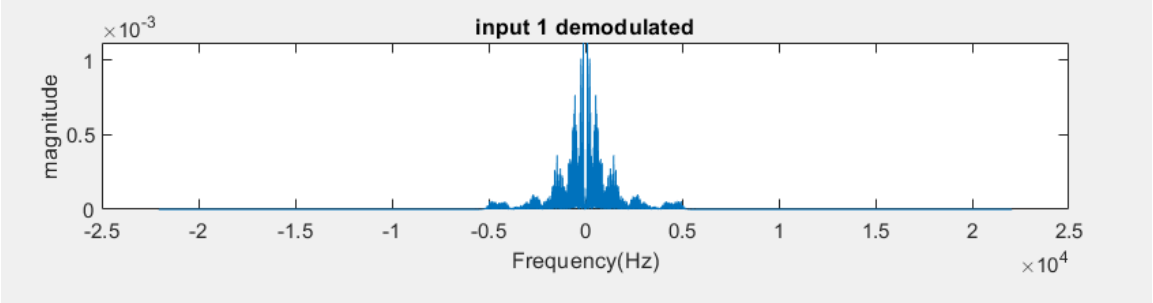
Then we use this filtered audio and start modulating it with carrier frequencies of 5000 & 16000 Hz to avoid the interference of each signal with itself or with the other ($\omega_{o1} \geq \omega_{s1}$ & $\omega_{o2} \geq \omega_{o1} + \omega_{s1} + \omega_{s2}$). Then we added the modulated signals with each other to get our transmitted signal and use FFT and plot it again as follows:



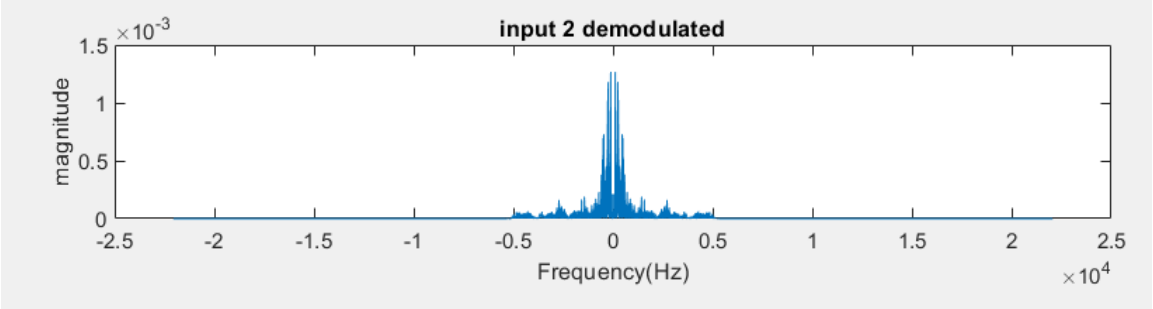
Fig(12):transmitted signal.

e.

Then we take the transmitted signal and demodulate it again and use FFT and plot it as follows:



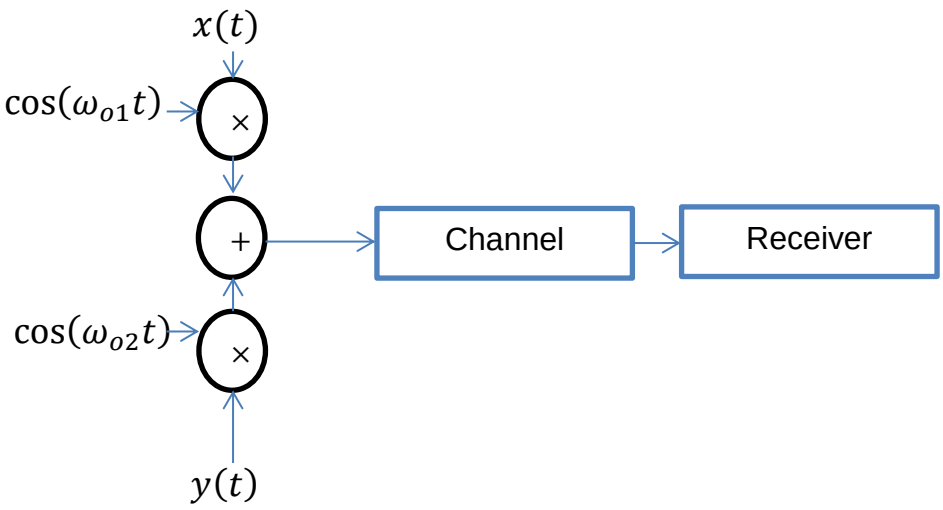
Fig(13):input1 after demodulation.



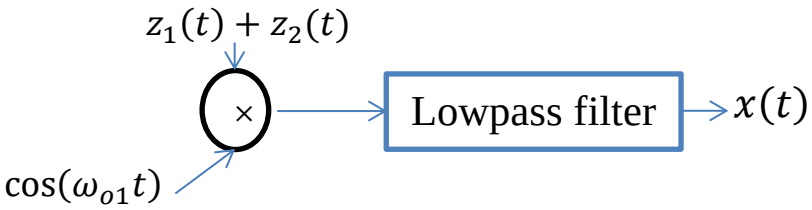
Fig(14):input2 after demodulation.

Block diagram

Transmitter:



Receiver:



Let the transmitted signal be $z(t)$

The equations in time domain:

$$x(t) = (z(t)\cos(\omega_{o1}t)) * h(t)$$

The equations in frequency domain:

$$X(\omega) = \frac{1}{2\pi}(Z(\omega) * \cos(\omega_{o1}t))H(\omega)$$

Appendix:

I.

A.

```
rgbimg=imread('peppers.png');
R=rgbimg(:,:,1);
G=rgbimg(:,:,2);
B=rgbimg(:,:,3);
Z=zeros(size(rgbimg,1),size(rgbimg,2));
red=cat(3,R,Z,Z);
green=cat(3,Z,G,Z);
blue=cat(3,Z,Z,B);
subplot(1,3,1)
imshow(red);
title('Red component');
subplot(1,3,2)
imshow(green);
title('Green component');
subplot(1,3,3)
imshow(blue);
title('Blue component');
```

B.

1)

```
rgbimg=imread('peppers.png');
R=rgbimg(:,:,1);
G=rgbimg(:,:,2);
B=rgbimg(:,:,3);
kernel=[0  0 -1  0  0;
        0  0 -1  0  0;
       -1 -1  8 -1 -1;
        0  0 -1  0  0;
        0  0 -1  0  0];

R1=conv2(im2double(R),kernel);
G1=conv2(im2double(G),kernel);
B1=conv2(im2double(B),kernel);
newimg=cat(3,R1,G1,B1);
figure
imshow(newimg);
title('edge detected');
```

2)

```
rgbimg=imread('peppers.png');
R=rgbimg(:,:,1);
G=rgbimg(:,:,2);
B=rgbimg(:,:,3);
kernel=[0  0 -1  0  0;
        0  0 -1  0  0;
       -1 -1  9 -1 -1;
        0  0 -1  0  0;
        0  0 -1  0  0];

R1=conv2(im2double(R),kernel);
G1=conv2(im2double(G),kernel);
B1=conv2(im2double(B),kernel);
newimg=cat(3,R1,G1,B1);
figure
imshow(newimg);
title('sharpened');
```

3)

```
rgbimg=imread('peppers.png');
R=rgbimg(:,:,1);
G=rgbimg(:,:,2);
B=rgbimg(:,:,3);
kernel=1/25*ones(5,5);
R1=conv2(im2double(R),kernel);
G1=conv2(im2double(G),kernel);
B1=conv2(im2double(B),kernel);
newimg=cat(3,R1,G1,B1);
figure
imshow(newimg);
title('blurred');
```

4)

```
rgbimg=imread('peppers.png');
R=rgbimg(:,:,1);
G=rgbimg(:,:,2);
B=rgbimg(:,:,3);
kernel=zeros(5,5);
kernel(3,:)=1/5;
R1=conv2(im2double(R),kernel);
G1=conv2(im2double(G),kernel);
B1=conv2(im2double(B),kernel);
newimg=cat(3,R1,G1,B1);
figure
imshow(newimg);
title('horizontal motion blurred');
```

C.

```
rgbimg=imread('peppers.png');
R=rgbimg(:,:,1);
G=rgbimg(:,:,2);
B=rgbimg(:,:,3);
kernel=zeros(5,5);
kernel(3,:)=1/5;
R1=conv2(im2double(R),kernel);
G1=conv2(im2double(G),kernel);
B1=conv2(im2double(B),kernel);
[y,z,j]=size(R1);
H=zeros(y,z);
H(1:5,1:5)=kernel;
fftH=fft2(H);
newimg=cat(3,R1,G1,B1);
FFTR=fft2(R1);
FFTG=fft2(G1);
FFTB=fft2(B1);
FFTresultR=FFTR./fftH;
FFTresultG=FFTG./fftH;
FFTresultB=FFTB./fftH;
red=ifft2(FFTresultR);
green=ifft2(FFTresultG);
blue=ifft2(FFTresultB);
reconstruction=cat(3,red,green,blue);
[i,o,m]=size(reconstruction);
reconstruction(i-3:i,:,:)=[];
reconstruction(:,o-3:o,:)=[];
figure
imshow(reconstruction);
title('reconstructed image');
```

II

```
fs=44100;    % sampling freq (Hz)
bits=16;     % Bit depth
nChannels=1;

% record audio

recorder1=audiorecorder(fs,bits,nChannels);
recorder2=audiorecorder(fs,bits,nChannels);
data='int16';
disp("start");
recordblocking(recorder1,10);
disp("end");
disp("start");
recordblocking(recorder2,10);
disp("end");
y1=getaudiodata(recorder1,data);
y2=getaudiodata(recorder2,data);
audiowrite('input1.wav',y1,fs);
audiowrite('input2.wav',y2,fs);

% read audio

[y1,fs]=audioread("input1.wav");
[y2,~]=audioread("input2.wav");
N=length(y1);

%transefer to frequency domain

Y1=fft(y1,N);
Y2=fft(y2,N);

f=(-N/2:N/2-1)*fs/N;

% plot the signals against frequency

tiledlayout(4,2);
nexttile
```

```
plot(f,abs(fftshift(Y1))/N)
title("input1 (before) against f")
xlabel('Frequency(Hz)');
ylabel('magnitude');
nexttile
plot(f,abs(fftshift(Y2))/N)
title("input2 (before) against f")
xlabel('Frequency(Hz)');
ylabel('magnitude');

% design filter

% filterDesigner;

% filter frequency

y_after1=filter(myfilter3,y1);
y_after2=filter(myfilter3,y2);
Y_after1=fft(y_after1,N);
Y_after2=fft(y_after2,N);

% plot filtered audio

nexttile
plot(f,abs(fftshift(Y_after1))/N)
title("filtered input 1 against f")
xlabel('Frequency(Hz)');
ylabel('magnitude');
nexttile
plot(f,abs(fftshift(Y_after2))/N)
title("filtered input 2 against f")
xlabel('Frequency(Hz)');
ylabel('magnitude');

% amplitude modulation variables

Ts = 1/fs;
fcarrier1 = 5000;
fcarrier2 = 16000;
```

```
% modulation and combining

y_modulated1 = y_after1 .*
cos(2*pi*fcarrier1*Ts*(1:N).');
y_modulated2 = y_after2 .*
cos(2*pi*fcarrier2*Ts*(1:N).');

TransmittedSignal = y_modulated1 + y_modulated2;
FTransmittedSignal = fft(TransmittedSignal,N);

% plot combined signal

nexttile
plot(f,abs(fftshift(FTransmittedSignal))/N);
title('transmitted signal')
xlabel('Frequency(Hz)');
ylabel('magnitude');

% demodulation and FFT

y_demodulated1 = TransmittedSignal .*
cos(2*pi*fcarrier1*Ts*(1:N).');
y_demodulated2 = TransmittedSignal .*
cos(2*pi*fcarrier2*Ts*(1:N).');

% filtering after demodulation

y_demodulated_after1 =
2*filter(myfilter3,y_demodulated1);
y_demodulated_after2 =
2*filter(myfilter3,y_demodulated2);

Y_demodulated_after1=fft(y_demodulated_after1);
Y_demodulated_after2=fft(y_demodulated_after2);

% plot demodulated signal

nexttile
```

```
plot(f,abs(fftshift(Y_demodulated_after1))/N);
title('input 1 demodulated');
xlabel('Frequency(Hz)');
ylabel('magnitude');
nexttile
plot(f,abs(fftshift(Y_demodulated_after2))/N);
title('input 2 demodulated');
xlabel('Frequency(Hz)');
ylabel('magnitude');

% writing output

audiowrite('output1.wav', y_demodulated_after1, fs);
audiowrite('output2.wav', y_demodulated_after2, fs);
```

References

1. _https://www.mathworks.com/help/matlab/math/array-indexing.html
2. https://www.mathworks.com/help/matlab/ref/double.cat.html
3. https://www.mathworks.com/help/matlab/ref/imshow.html
4. https://youtu.be/VFt3UVw7VrE?si=uOgkrkB_-KLKdmZr
5. https://www.mathworks.com/videos/how-to-make-subplots-in-matlab-using-tiledlayout-1599239984171.html
6. https://www.mathworks.com/help/matlab/import_export/record-and-play-audio.html
7. https://www.mathworks.com/help/matlab/ref/audiowrite.html