Department of Electrical Engineering

Prince Mohammad Bin Fahd University

EEEN 4341 – Communication Systems

Fall semester 2022- 2023

Course Project

Title: Basic Communication System

Prepared by:

Munirah Mohammed Aldossary – 201901477

Instructor: Dr. Hiren Mewada

Table of Contents

Table of Figures:

Abstract:

In this project I'll develop a simple communication system. The project goes through several phases. These include using an audio file, users can display the input sound signal in time and frequency domain and obtain the sound wave from it. I will then combine the sound signal with various channels on the transmitter side and observe how it affects the original sound signal. Additionally, generate noise and add it to the sound signal while observing the effects and changes the noise has on the original signal. After sending the transmitted signal through the low pass filter when it is in the receiver, I will apply a low pass filter to eliminate noise from the message signal and verify the signal's quality.

Introduction:

Today, communication plays a significant role in our daily life. The invention of wireless transmission laid the groundwork for modern communications. The establishment of a connection between two geographically distinct locations requires effective communication systems. Using these techniques ensures efficient and rapid data transfer between these two locations. Usage of the technology was followed by telecommunication and wireless communication technologies. Since the development of satellite communication technology, information flow has massively enhanced.

In a communication system, a system model describes a communication exchange between two stations, a transmitter, and a receiver. Signals or information are transmitted from one place to another over a channel. It demonstrates how the signal makes use of it during its travel from a source to a destination. Noise, attenuation, and distortion are some of the signal transmission constraints of this medium.

There are two different communication systems, depending on the modulation process, which is the process that moves the message signal into a specific frequency spectrum that is determined by the physical channel. The first kind of communication system is known as carrier communication, which is the communication that uses modulation to change a signal's frequency spectrum. The Baseband, on the other hand, is a different type of communication system that is used to identify the frequency band of the first message signal from the source or input transducer. Baseband communication cannot be successfully transmitted over radio (wireless) networks due to its low frequency content.

Noise is one of the channel faults that impacts the signal that is received at the target location. Noise can come from both internal and external sources. Examples of external sources include cross talk from nearby transmitted signals, interference from a natural source like lightning, solar or cosmic radiation, radiation from moving vehicles, etc. The amount of outside noise can be minimized or even eliminated with correct channel design and cable shielding. Digital transmission can also significantly minimize outside noise. Thermal noise from charge carrier dispersion and recombination in other electronic devices, as well as noise from the collision and random mobility of electrons in conductors, are examples of internal sources of noise. Internal noise can be decreased by cooling and employing digital technologies for transmission.

Procedure:

In this project, I'll build a very simple communication system, as shown in figure 1. It can be seen from the communication system's block diagram, which will be utilized in this project, that baseband communication will be used. Since there is no demodulation or modification of the message signals during baseband communication, they are transmitted directly.

To start with, I will extract the input signal from the audio file for the transmitter part. Next, I will pass the input signal through five different channels and observe the adjustments. After that, we'll generate random white noise, add it to the input signal, and observe what happens to it and how it affects the input signal.

Additionally, I will pass the noisy signal I will receive from the transmitter into a low pass filter for the receiving part. Because the transmitted signal will have a high frequency and the white noise will have a flat power spectrum, I will use a low pass filter (LPF) to remove the undesirable low frequencies produced by the noise and recover the high frequencies that are related to the transmitted message signal.

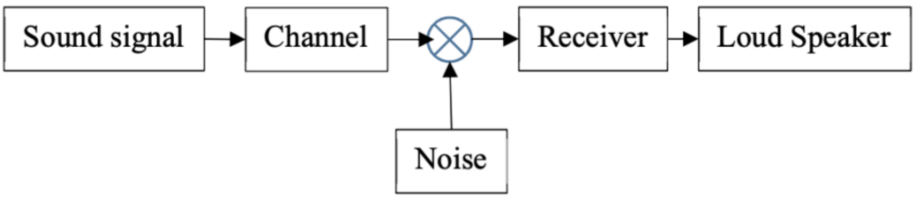

Figure 1 - Block Diagram of a Baseband Communication System

In the following parts I will implement the procedures:

1. Transmitter

Requirements:

1. Play your sound using MATLAB.
2. Plot the sound signal in time domain and frequency domain.

Results:

1- Record from a microphone input.

```
[sample_voise,Fs] = audioread('dove.mp3');
```

2- Play the voice to verify that the sound was clear and uploaded

```
sound(sample_voise,2*Fs);
```

3. Then plot the time domain of the audio sample

```
%time domain
```

```
time_values = linspace(0,length(sample_voise)/Fs,length(sample_voise));
plot(time_values,sample_voise);
xlabel('Time (ms)');
ylabel('Amplitude');
title('Audio Sample in Time Domain');
```

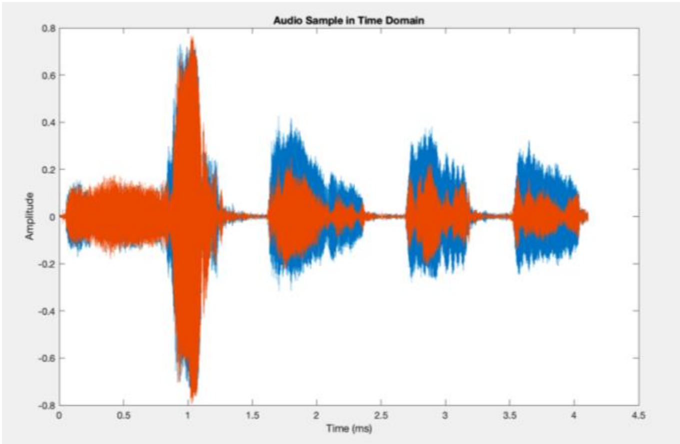

Figure 2 - Audio Sample in Time Domain

4. Plotting the frequency domain

```
%frequency domain
Y=fftshift(fft(sample_voise));
magnitudeY = abs(Y);          % Magnitude of the FFT
phaseY = unwrap(angle(Y));   % Phase of the FFT
dB_mag=mag2db(magnitudeY);
Frequency_time_values = linspace(-Fs/2,Fs/2,length(magnitudeY));

%Magnitude plot
plot(Frequency_time_values, magnitudeY);
title('Magnitude response of signal');
ylabel('Magnitude');
```

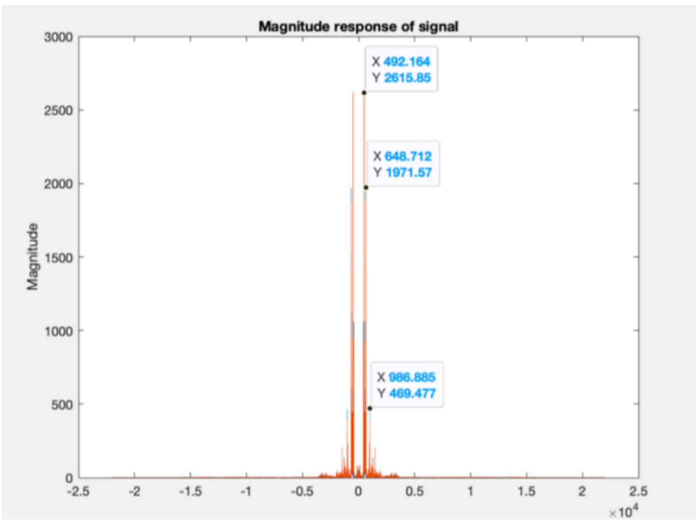

Figure 3 - Magnitude Response of the Audio Signal

```
%Phase plot
plot(Frequency_time_values, phaseY);
title('Phase response of signal');
xlabel('Frequency in kHz')
ylabel('radians');
```

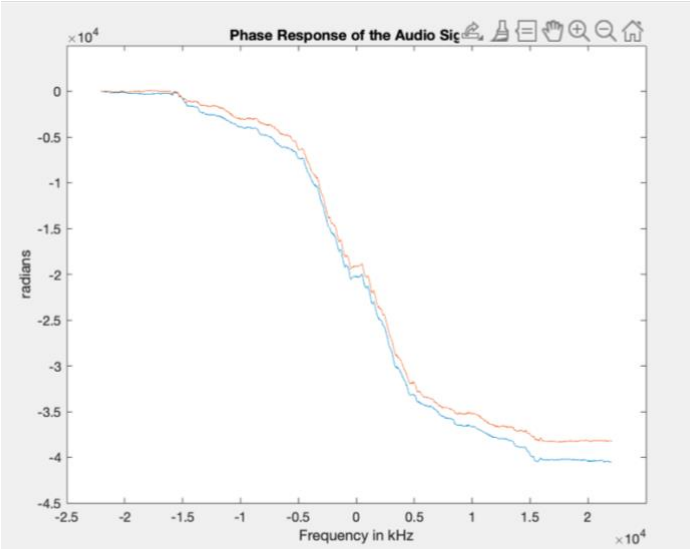

Figure 4 - Phase Response of the Audio Signal

Discussion:

In this section, the audio signal of the recorded voice file was saved into a variable called sample voice that contains the audio file's electrical signal using the audioread function. The audio sample's amplitude range is between -0.8 and 0.8, as seen in figure 2 of the time domain plot I generated earlier. I then took the audio signal's Fourier transform and relocated the zero-frequency component to the center of the grid to plot the magnitude response of the audio signal as seen in figure 3 above. I could see that the audio signal's frequency components are 4921.64 kHz, 6487.12 kHz, and 9868.85 kHz. As illustrated in figure 5 below, its magnitude is as follows: 2615.85, 1971.57, and 469.477. I could represent figure 3 in a mathematical expression as follow:

$$M(\omega) = \pi[\delta(\omega - 4921.64 \text{ kHz}) + \delta(\omega - 6487.12 \text{ kHz}) + \delta(\omega - 9868.85 \text{ kHz})]$$

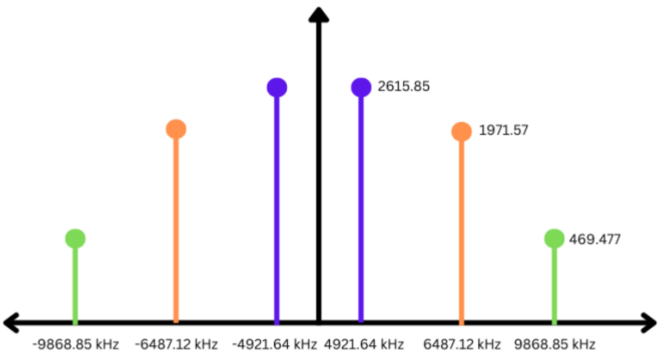

Figure 5 - Frequencies components of the Audio Signal

2. Channel:

The following channel responses are available. You have to pass the sound signal over these channels.

You have four channel responses:

1. Delta function 2. exp(-2pi*5000t)

3. exp(-2pi*1000t) 4. The channel has the following impulse response

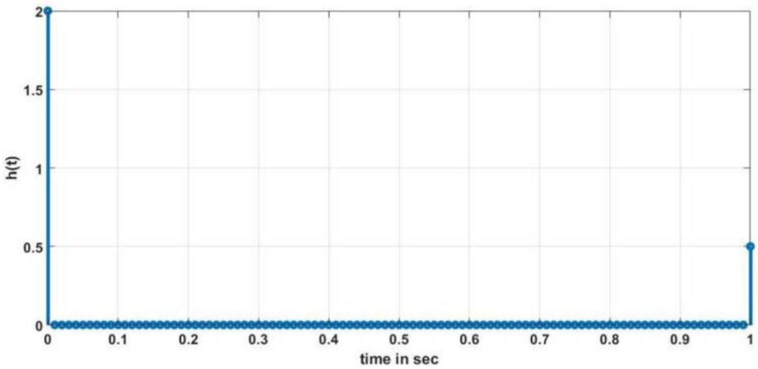

Figure 6 - Channel Impulse Response

Requirements:

1. Try these channel responses and develop code for transmission.
2. Compare the effect of first three channel response on given sound signal.

   They Almost have the same effect on the sound signal.

Results:

1. Delta function

```
%Channels responses
% passing the sound signal over delta function
delta_func= 1;
delta_func_sample=linspace(-Fs/2,Fs/2,length(delta_func));
delta_sound_1= conv(sample_voise(:,1), delta_func);
delta_sound_2= conv(sample_voise(:,2), delta_func);
delta_signa = [delta_sound_1; delta_sound_2];

plot(delta_sound_1(1:end/2));
title('Convolution between delta and sound signals');
xlabel('Time')
ylabel('Amplitude');
```

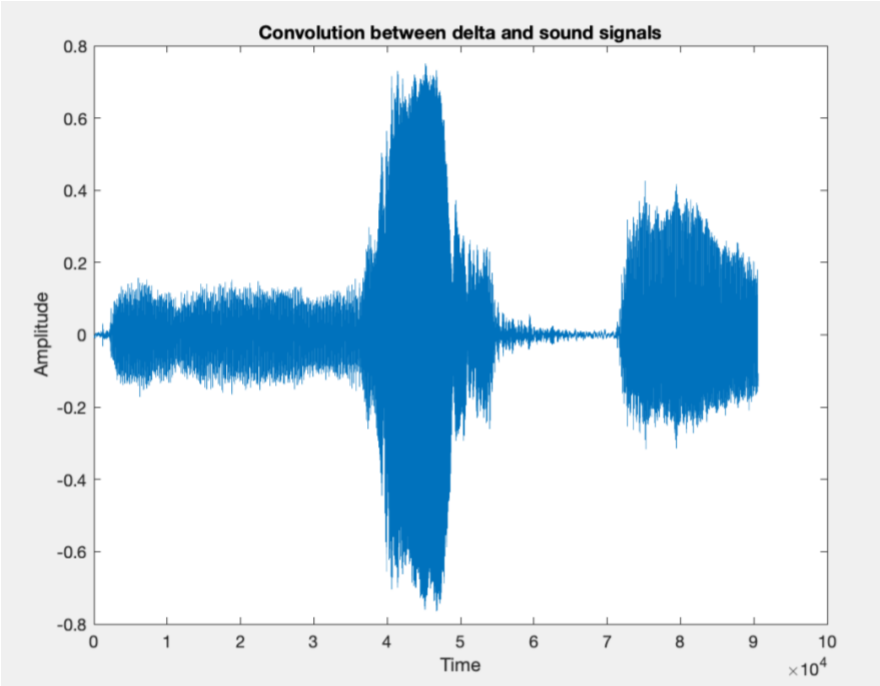

Figure 7 - Convolution Between Delta and Input Signals

```
%frequency domain
Y=fftshift(fft(delta_sound_1));
magnitudeY = abs(Y);          % Magnitude of the FFT
phaseY = unwrap(angle(Y));   % Phase of the FFT
dB_mag=mag2db(magnitudeY);
Frequency_time_values = linspace(-Fs/2,Fs/2,length(magnitudeY));

%Magnitude plot
plot(Frequency_time_values, magnitudeY);
title( 'Convolution between delta and sound signals');
ylabel('Magnitude');
```

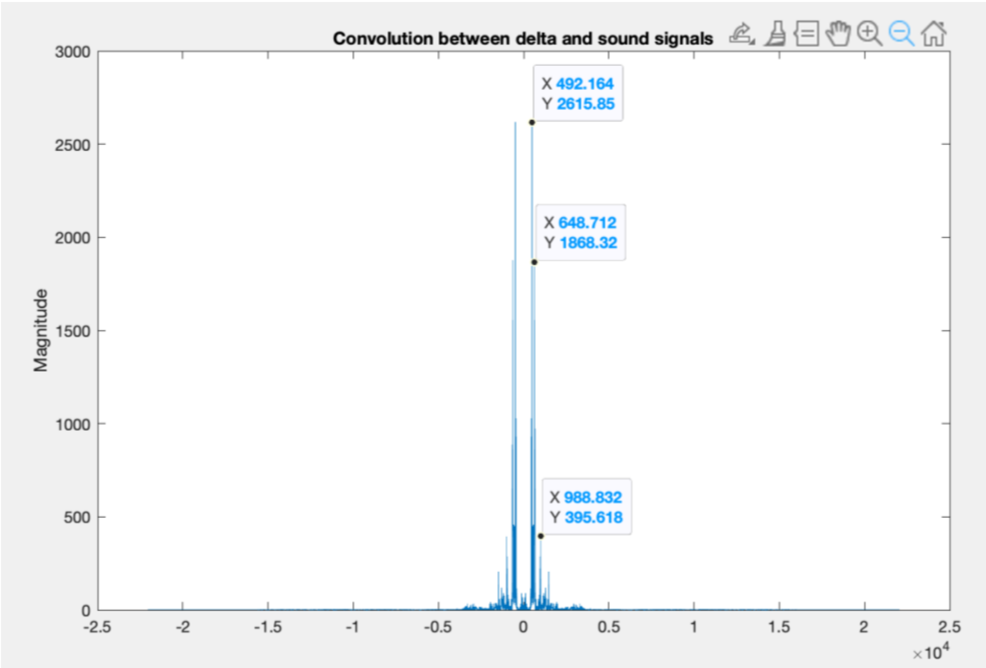

Figure 8 - Magnitude response of the Convolution between delta and sound signals

2. exp(-2pi*5000t)

```
% passing the sound signal over exp(-2pi*5000t)
sec_func= exp(-2*pi*5000*time_values) ;
sec_func_sample=linspace(-Fs/2,Fs/2,length(sec_func));
sec_conv_signal_1= conv(sample_voise(:,1), sec_func);
sec_conv_signal_2= conv(sample_voise(:,2), sec_func);
sec_func_signal = [sec_conv_signal_1; sec_conv_signal_2];

plot(sec_conv_signal_1(1:end/2));
title('Convolution between exp(-2pi*5000t) and sound signals');
xlabel('Frequency in kHz')
ylabel('Omega(w)');
```

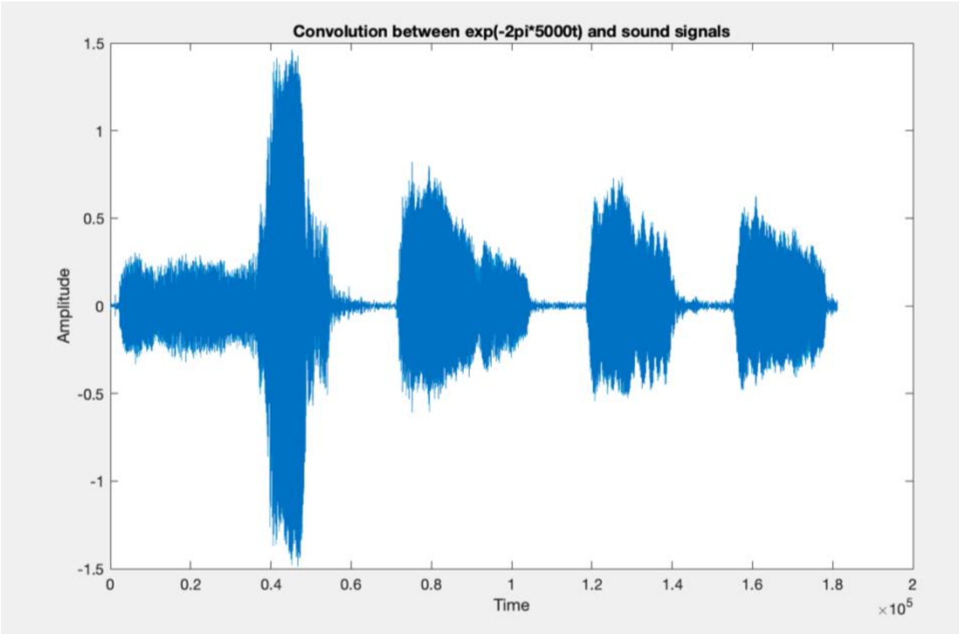

Figure 9 - Convolution Between exp(-2pi*5000t) and Input Signals

```
%frequency domain
Y=fftshift(fft(sec_conv_signal_1));
magnitudeY = abs(Y);         % Magnitude of the FFT
phaseY = unwrap(angle(Y));  % Phase of the FFT
dB_mag=mag2db(magnitudeY);
Frequency_time_values = linspace(-Fs/2,Fs/2,length(magnitudeY));

%Magnitude plot
plot(Frequency_time_values, magnitudeY);
title('Convolution between delta and sound signals');
ylabel('Magnitude');
```

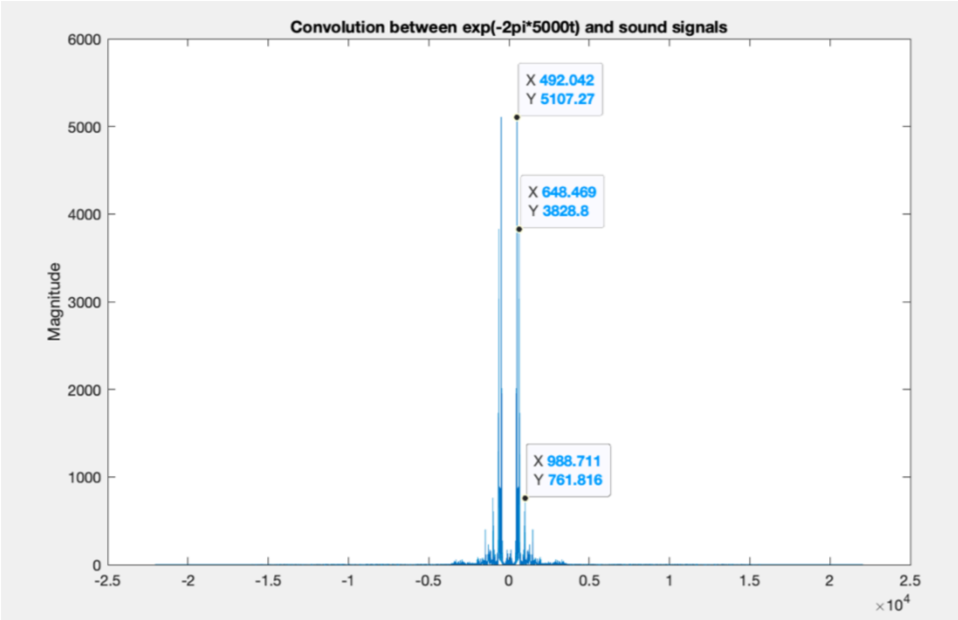

*Figure 10 - Magnitude response of the Convolution between exp(-2pi*5000t) and sound signal.*

3. exp(-2pi*1000t)

```
% passing the sound signal over exp(-2pi*1000t)
third_func= exp(-2*pi*1000*time_values);
third_func_sample=linspace(-Fs/2,Fs/2,length(third_func));
third_conv_signal_1= conv(sample_voise(:,1),third_func);
third_conv_signal_2= conv(sample_voise(:,1),third_func);
third_func_signal = [third_conv_signal_1; third_conv_signal_2];

plot(third_conv_signal(1:end/2));
title('Convolution between exp(-2pi*1000t) and sound signals');
xlabel('Frequency in kHz')
ylabel('Omega(w)');
```

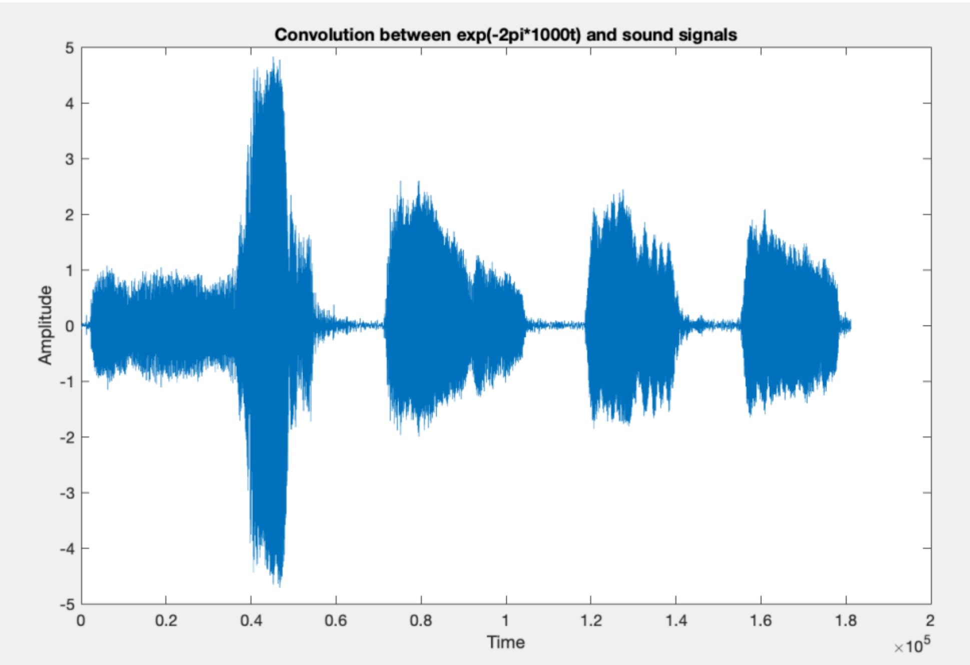

Figure 11 - Convolution Between exp(-2pi*1000t) and Input Signals

```
%frequency domain
Y=fftshift(fft(third_conv_signal_1));
magnitudeY = abs(Y);          % Magnitude of the FFT
phaseY = unwrap(angle(Y));   % Phase of the FFT
dB_mag=mag2db(magnitudeY);
Frequency_time_values = linspace(-Fs/2,Fs/2,length(magnitudeY));

% %Magnitude plot
plot(Frequency_time_values, magnitudeY);
title('Convolution between exp(-2pi*1000t) and sound signals');
ylabel('Magnitude');
```

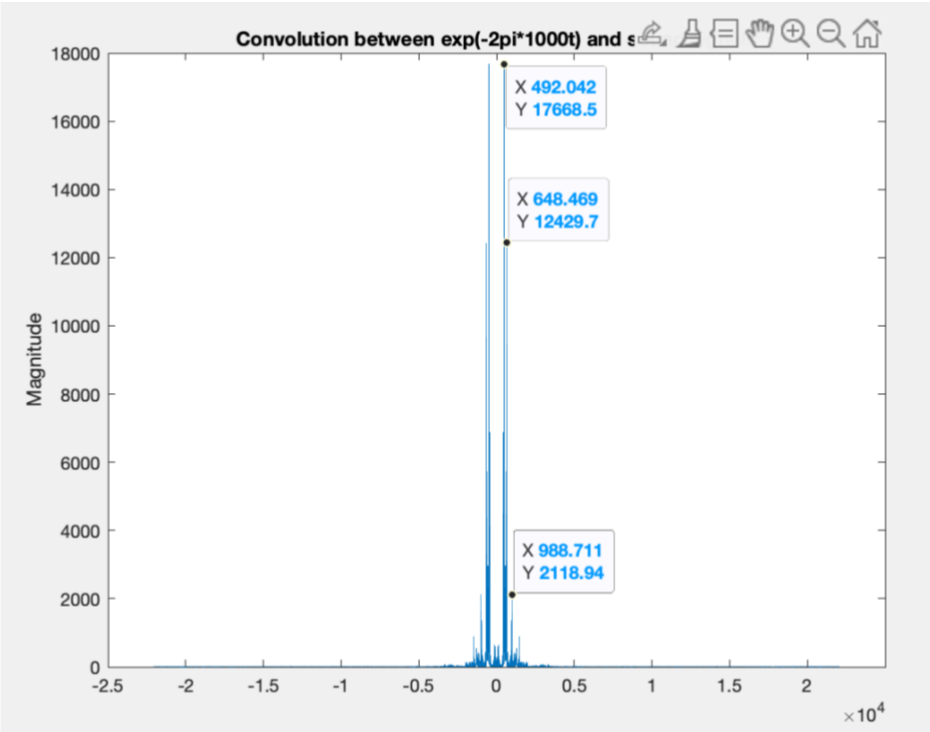

*Figure 12 - Magnitude response of the Convolution between exp(-2pi*1000t) and sound signal.*

4. The channel has the following impulse response

```
%passing the sound signal over impulse response
imp = [1; zeros(1,1)];
b = 2;
a = [1 -0.5];
impulse_func = filter(b,a,imp);
%stem(0:1,impulse_func)
impulse_conv_signal= conv(sample_voise(:,1),impulse_func);

plot(impulse_conv_signal);
title('Convolution between impulse response and sound signals');
xlabel('Frequency in kHz')
ylabel('Omega(w)');
```

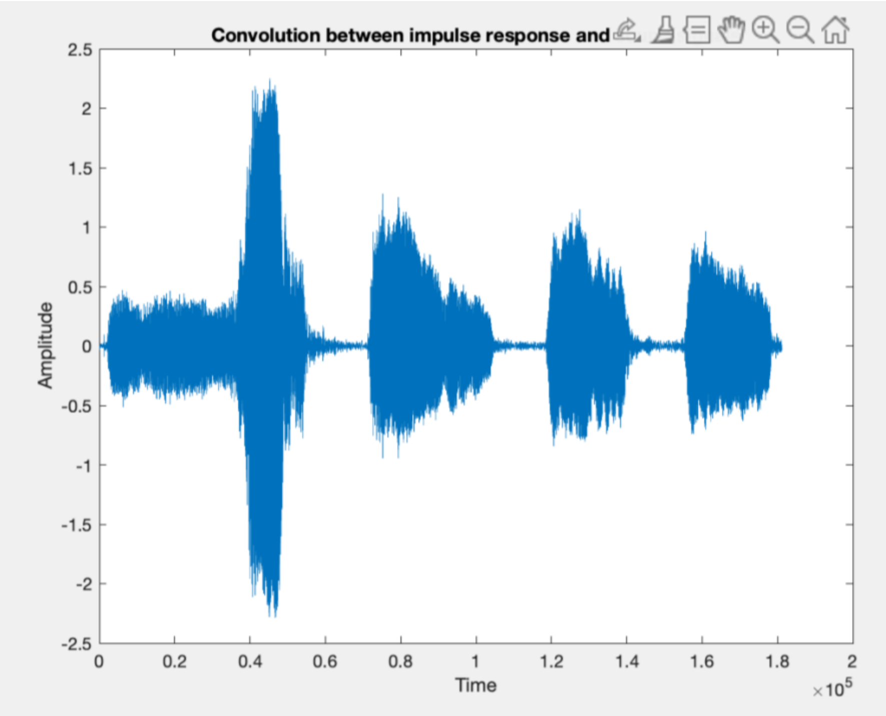

Figure 13 - Convolution Between Impulse Response and Input Signals

```
%frequency domain
Y=fftshift(fft(impulse_conv_signal));
magnitudeY = abs(Y);          % Magnitude of the FFT
phaseY = unwrap(angle(Y));   % Phase of the FFT
dB_mag=mag2db(magnitudeY);
Frequency_time_values = linspace(-Fs/2,Fs/2,length(magnitudeY));

%Magnitude plot
plot(Frequency_time_values, magnitudeY);
title('Magnitude response of Convolution between impulse response and sound
signal');
ylabel('Magnitude');
```

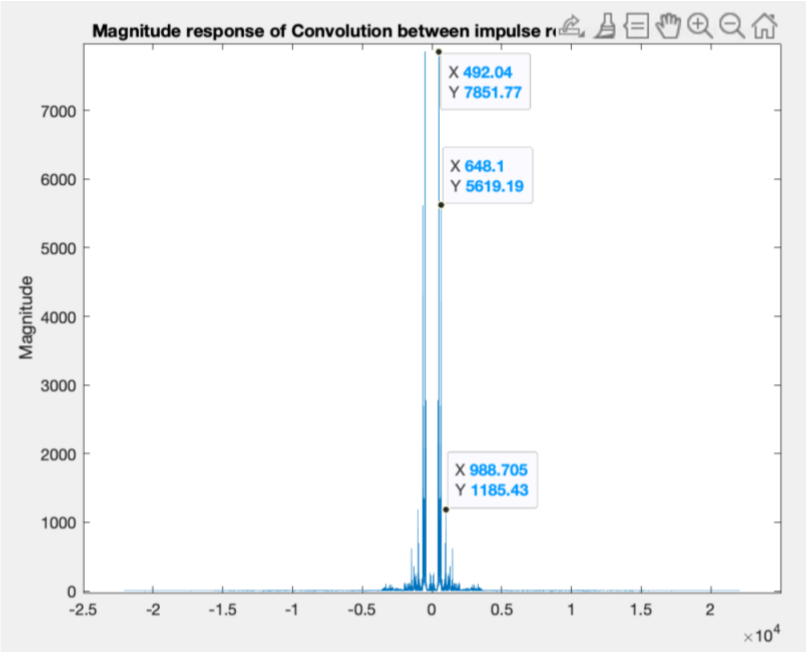

Figure 14 - Magnitude response of Convolution between impulse response and sound signal

Discussion:

In this part, I pass the audio signal into five different channels. Therefore, I convolute the audio signal with the channel signal in the time domain using the convolutional technique, and then I plot the time domain. I also plot the magnitude response using the convolutional output's Fourier transform. The magnitude response that I obtained allowed us to distinguish the variations between the channels.

To demonstrate the magnitude response for the first channel, I convoluted the audio signal in the time domain with the delta function as shown in figure 7, The sound signal won't be impacted by this channel, and the output signal will be the same as the input signal after I obtained the Fourier transform. This transformation may also be expressed mathematically as follows:

$$m(t) * \delta(t) \iff M(\omega)(1)$$

For the following channel, which is exp(-2pi*5000t) convolved with the audio signal as shown in figure 9 and its magnitude response as shown in figure 10, An train impulse function with a 10k pie shifted to the right will be the output signal. I may express this transformation mathematically as follows:

$$m(t) * e^{-2\pi 5000t} \iff M(\omega - 10000\pi)$$

To represent this transformation mathematically, I could write the following expression for the third channel, which is exp(-2pi*1000t) convolved with the audio signal shown in figure 11 and its magnitude response shown in figure 12, and the output will be an train impulse function with a 2k pie shifted to the right:

$$m(t) * e^{-2\pi 1000t} \iff M(\omega - 2000\pi)$$

For the fourth channel, which is an impulse response convolve with the audio signal as shown in figures 13 and 14, and whose magnitude response is shown in figure 14, I could represent this transformation as the following mathematical expression:

$$m(t) * \frac{1}{2}\delta(t - t_0) + \frac{1}{2}\delta(t + t_0) \iff M(\omega)\left(\frac{1}{2}\cos(\omega t_0)\right)$$

3. Noise:

An unwanted signal that interferes with a desired signal is referred to as noise in general. These undesirable signals come from a range of sources that can be categorized into two primary groups: Interference, typically created by humans (man-made), or Random noise that occurs naturally. With consideration that channel is noisy, above channel will add noise before the reception of the signal. The random noise can be generated in MATLAB using following commands:

Z(t) =sigma*randn(1,length(x))

Where x is a vector represents the output of the channel and the user should enter the value of the sigma at this stage. The output will be a Gaussian distributed noise with zero mean and standard deviation of sigma.

Requirement:

1. Play your sound file after adding noise
2. Plot your sound file in time domain and the frequency domain

Results:

```
%Adding Noise
sigma = 0.8;
%Generaing the noise
noise = sigma*randn(size(sample_voise));
%Adding noise to the signal
voice_noisy = sample_voise + noise;
%Play Sound with noise
sound(voice_noisy, 2*Fs);
%time domain
time_noise_values = linspace(0,length(voice_noisy)/Fs,length(voice_noisy));
plot(time_noise_values,voice_noisy);
xlabel('Time (ms)');
ylabel('Amplitude');
title('Audio & Noise Sample in Time Domain');
```

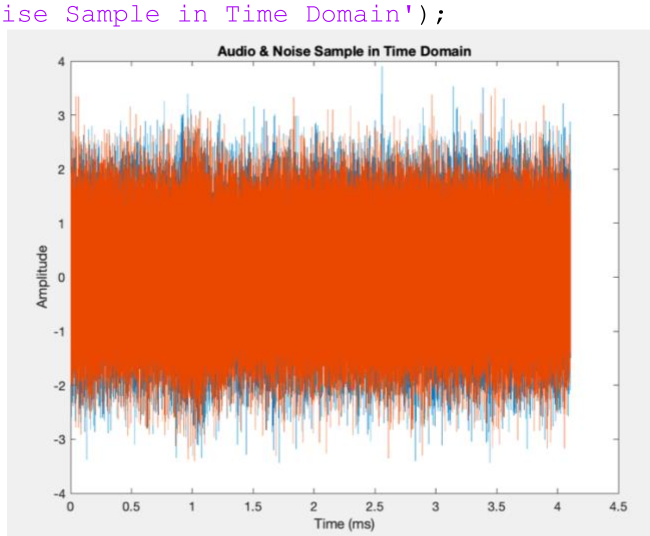

Figure 15 - Audio & Noise Sample in Time Domain

```
%frequency domain
Z=fftshift(fft(voice_noisy));
magnitudeZ = abs(Z);          % Magnitude of the FFT
phaseZ = unwrap(angle(Z));   % Phase of the FFT
dB_magZ=mag2db(magnitudeZ);
Frequency_noise_values = linspace(-Fs/2,Fs/2,length(magnitudeZ));

%Magnitude plot
plot(Frequency_noise_values, magnitudeZ);
title('Magnitude response of Noise');
ylabel('Magnitude(dB)');
```

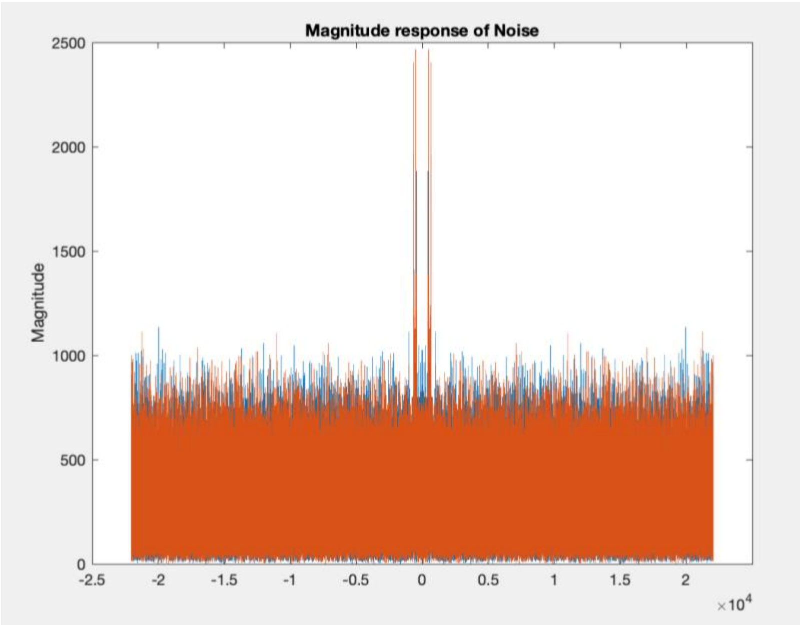

Figure 16 - Magnitude Response of Noise & Audio Signals

```
%Magnitude in dB plot
plot(Frequency_noise_values, dB_magZ);
title('Magnitude response of Noise');
ylabel('Magnitude(dB)');
```

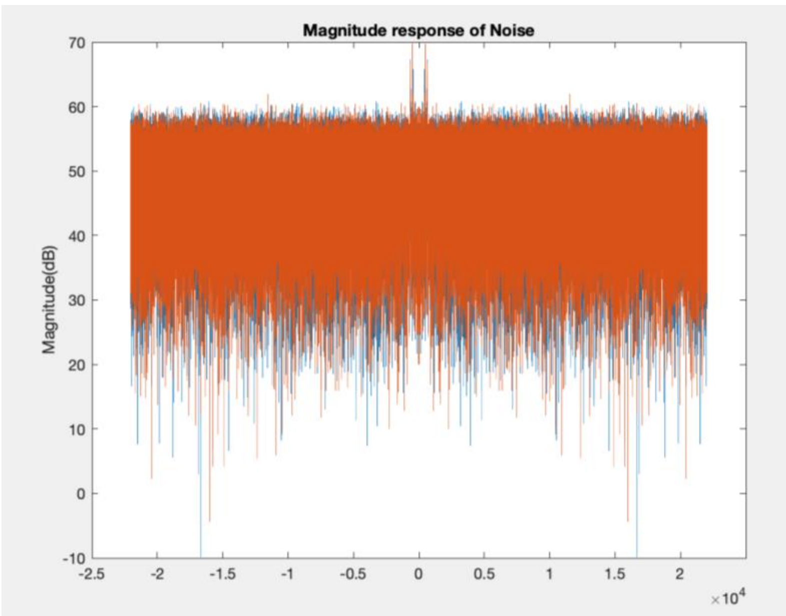

Figure 17 - Magnitude Response in dB of Noise & Audio Signals

```
%Phase plot
plot(Frequency_noise_values, phaseZ);
title('Phase response of Noise');
xlabel('Frequency in kHz')
ylabel('radians');
```

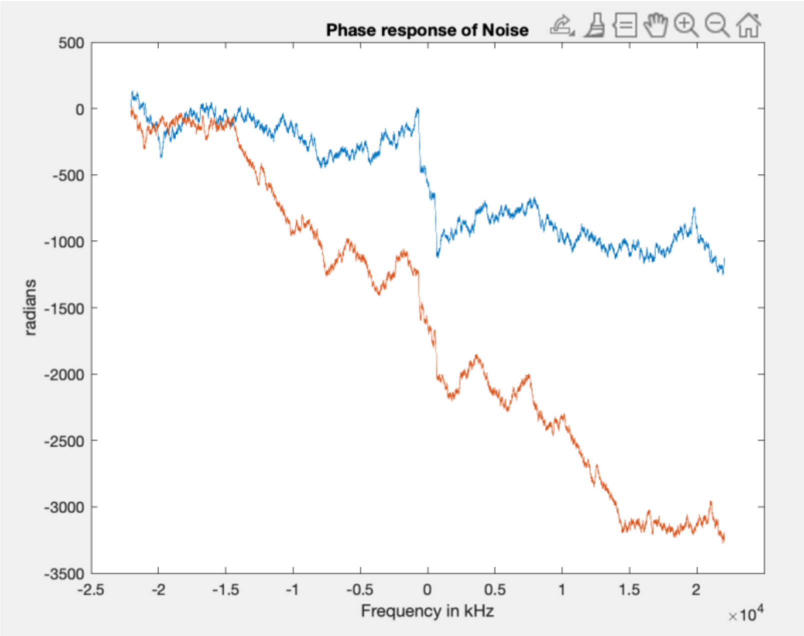

Figure 18 - Phase response of Noise & Audio Signals

Discussion:

Figure 15 illustrates how I produced white noise for this section and combined it with the audio signal. In this figure, the white noise and sound signal are overlapping in time domain. As a result, I developed the frequency domain of the audio and noise signals to avoid the overlapping. As seen in figure 16, the sound signal has low frequencies, making it simple to distinguish between the two in the frequency domain. Additionally, the white noise's spectrum is flat on a spectrogram, as seen in figures 16 and 17, because all its frequencies have the same power. I also developed a phase plot, as seen in figure 18, to further highlight the differences between the sound signal and background noise.

4. Receiver:

At receiver, you need to recover the signal removing the noise from the sound signal.

You will construct an ideal low pass filter which has a cut off of 3400 KHz. The frequency response of the filter as shown in figure. And Pass the noisy sound over the ideal filter.

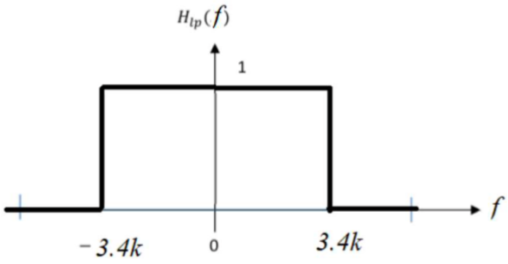

Figure 19 - Low Pass Filter in the frequency domain

Requirement:

1- Play the sound file after the filter
2- Plot the output sound file in time domain and the frequency domain

Results and discussion:

```
%Receiver:
n = length(magnitudeZ);
sampPerFreq = int64(n/Fs);
limit = (sampPerFreq*(Fs/2 - 3400));
magnitudeZ([1:limit n-limit+1:end])=0;
real_Z=real(ifft(ifftshift(magnitudeZ)));
%Play the sound file after the filter
sound(real_Z);

%Plot the output sound file in time domain
plot(B);
xlabel('Time (ms)');
ylabel('Amplitude');
title('LPF in Time Domain');
```

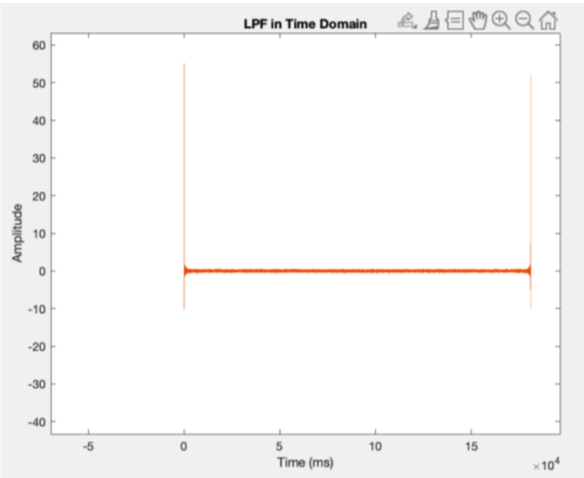

Figure 20 - The Noise & Audio Signals Response after applying the LPF

```
%Plot the output sound file in the frequency domain
%frequency domain
```

```
 Y=ifftshift(ifft(magnitudeZ));
 magnitudeY = abs(Y);         % Magnitude of the FFT
 phaseY = unwrap(angle(Y));   % Phase of the FFT
 dB_mag=mag2db(magnitudeY);
 Frequency_noise_values1 = linspace(-
Fs/2,Fs/2,length(Frequency_noise_values));

%Magnitude plot
plot(Frequency_noise_values1, magnitudeY);
title('Magnitude response of Noise & Audio Signals after the LPF');
ylabel('Magnitude');
```

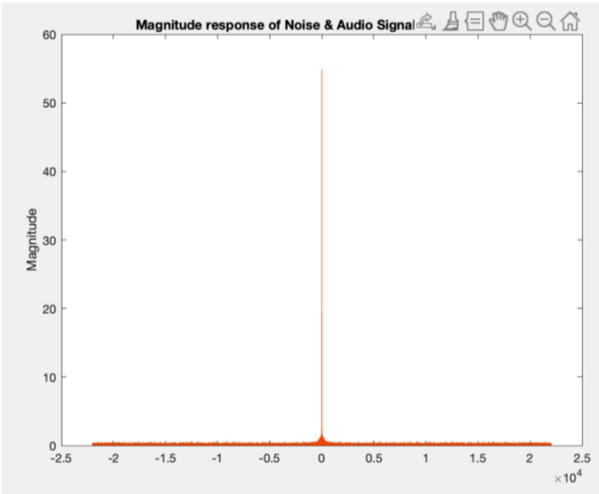

Figure 21 - Magnitude response of Noise & Audio Signals after the LPF

```
%Magnitude plot in dB
plot(Frequency_noise_values1, dB_mag);
title('Magnitude response of Noise & Audio Signals after the LPF');
ylabel('Magnitude(dB)');
```

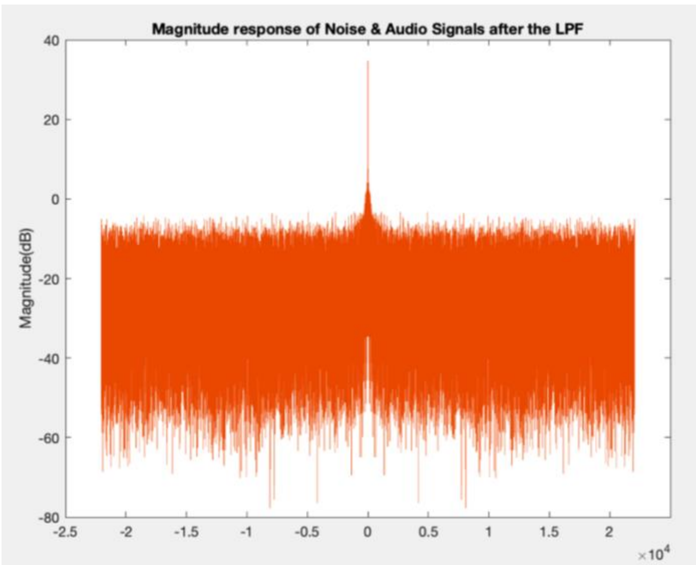

Figure 22 - Magnitude response in dB of Noise & Audio Signals after the LPF

```
%Phase plot
plot(Frequency_noise_values1, phaseY);
title('Phase response of Noise & Audio Signals after the LPF');
xlabel('Frequency in kHz')
ylabel('radians');
```

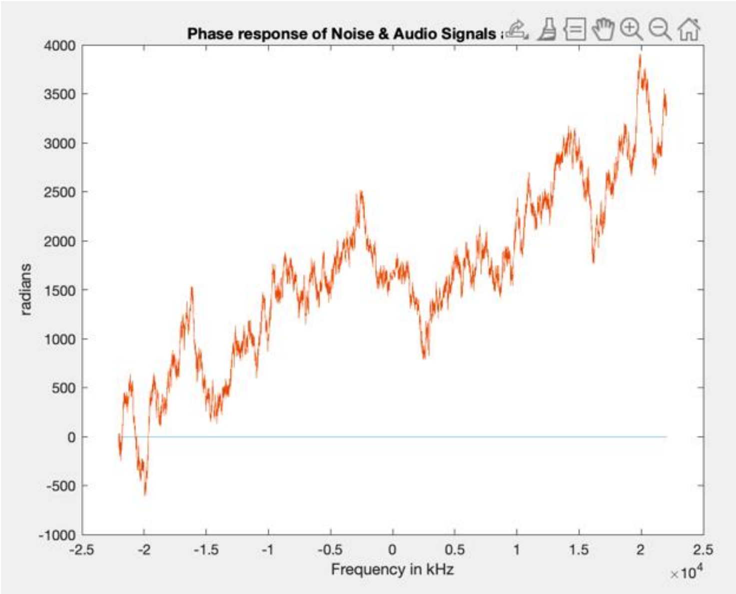

Figure 23 – Phase response of Noise & Audio Signals after the LPF

Discussion:

I completed the receiver side in the final section. I created a low pass filter with a cutoff frequency of 3400 kHz to recover the message signal. I calculated the low pass filter's limits with a 3400kHz range and stored the length of the noisy signal to apply that into action. Any values outside the LPF range should then be changed to zero. Next, display the time domain as shown in figure 20, the magnitude response as shown in figure 21, the magnitude response in dB as shown in figure 22, and the phase response as shown in figure 23 in the frequency domain using the real values of the complex noisy signal after it has been passed through the LPF.

Conclusion:

In conclusion, our modern world, communication is crucial to daily existence. Modern communications were established by the development of wireless transmission. For this project, I designed a basic communication system. The project is divided into several phases. One of these techniques involves the use of an audio file to display the input sound signal in the time and frequency domain and extract the sound wave from it. then combine the sound signal with other channels on the transmitter side and observe how it modifies the original sound signal. Additionally, noise was generated and added to the audio wave while analyzing the effects and modifications the noise has on the original sound. After the sent signal has passed through the low pass filter in the receiver, it is used to remove noise from the message signal and assess the signal's quality.