Motivation:

- To effectively filter noise from an audio clip recorded in a real world environment and modulate the amplitude of the resulting wave form.
- In this project of ours we develop a program to successfully filter most of the background noise thereby giving a clean and clear audio the user initially intended to record.
- Once the noise has been successfully filtered we try to make the signal transmission ready by modulating the amplitude of the signal.

ANALYSING THE FREQUENCY RESPONSE OF THE AUDIO FILE

- Read the audio file using the 'audioread' in-built function from MATLAB's own audio-processing toolkit. This function outputs 2 values.
- The range of frequencies which the filter will let through will be defined based on the observations we make at this step. We perform Fast Fourier Transform of the input audiofile. This operation enables us to visualise the behaviour of the input signal in the frequency domain.

IMPLEMENTATION OF AN APPROPRIATE FILTER

- We implement a bandpass filter to help us filter out the unnecessary frequencies and let only a selected range of frequencies pass through the filter.
- Preferably, we go for the butterworth filter. We make use of the in-built function of MATLAB - 'Butter' to design a butterworth filter.
- The resulting sound wave is filtered off most of the background and unnecessary frequencies which we intended to get rid of.
- The resulting waveform and the frequency response of the band pass filter is also plotted to help us visualise the dynamics of the filter and the difference it brings to the output of the audio file.

AMPLITUDE MODULATION OF THE FILTERED SIGNAL

- The message signals which are to be broadcasted in short, medium and long range are carried through Amplitude modulation.
- While in the previous step, the resulting sound wave which passed through the butterworth filter is modulated using the MATLAB inbuilt function - modulate(Z,Fc,Fs,'am'). Where the variables inside the function denotes as follows:
- Z - The resultant signal (message) of the butterworth filter
- Fc - Carrier Frequency
- Fs - Sample Frequency rate
- Am - Amplitude modulation

```
[z,Fs] = audioread('C:\Users\Bhanu\Downloads\Before.mpeg');
sound(z,Fs);
subplot(2,2,1);

plot(z);
max(z) % Normalized Amplitude
title("original Signal");
xlabel('No of Samples')
ylabel('Amplitude')

% Frequency Response of the noise Signal
L = length(z);
NEFT = 2^nextpow2(L);
Z_FFT = abs(fft(z,NEFT));
Freq = Fs/2*linspace(0,1,NEFT/2+1);
subplot(2,2,2);
plot(Freq,Z_FFT(1:length(Freq)));title("Frequency Plot of Noise Signal");
xlabel('Frequency')
ylabel('Magnitude')

% Creating Filter
order = 5;
w_n = [1200 1700]* 2/Fs;
[b, a] = butter(order, w_n, "bandpass");
[h, w] = freqz(b,a,1024,Fs);
subplot(2,2,3);
plot(w,20*log10(abs(h)));title("The Band Pass Filter");

% Filtering the noise signal
z_filter = filter(b,a,z);

for i = 1:length(z_filter)
   if(z_filter(i)>0.6)
      z_filter(i) = 0.5;
   end
end

% Frequency Response of the Filtered Signal
L = length(z_filter);
NEFT = 2^nextpow2(L);
Z_FFT = abs(fft(z_filter,NEFT));
Freq = Fs/2*linspace(0,1,NEFT/2+1);
figure
plot(Freq,Z_FFT(1:length(Freq)));title("Frequency Plot of Filtered Signal");
xlabel('Frequency')
ylabel('Magnitude')

 %sound(z_filter,Fs)



x = modulate(z_filter,0.49*Fs,Fs, 'am');
%x = ammod(z, 0.49*Fs, Fs)
L_modulated = length(x);
```

```
NEFT = 2^nextpow2(L_modulated);
x_FFT = abs(fft(x,NEFT));
Freq_mod = 0.49*Fs/2*linspace(0,1,NEFT/2+1);
figure
plot(Freq_mod,x_FFT(1:length(Freq_mod)));title("Frequency Plot of Noise Signal");
xlabel('Frequency')
ylabel('Magnitude')

%soundsc(x);
```