Speech Processing Lab - Week 3

Name: S U Swakath

Roll number: 180020036

Google Colab Link : https://colab.research.google.com/drive/1USEdY1uOpee5OhuIj85xVCcgEtlPb4tK?usp=sharing

Aim

- To identify optimal sampling frequency for speech signal processing.
- To identify optimal bit resolution for speech signal processing.
- To understand the significance of telephone bandwidth.
- To understand the difference between narrowband and wideband speech.

Theory

Introduction

Speech signal processing on a digital machine needs sampling and storing of the analog version of the speech signal generated at the output of microphone. Sampling frequency is the parameter that controls the sampling process. The number of bits per sample is the parameter that controls the bit resolution. The intelligibility, amount of information and also the perceptual quality of speech depends on these two parameters. The aim of the experiment is to learn the significance of these two parameters.

Sampling theorem and sampling frequency

The sampling of analog signal is based on Nyquist-Shannon sampling theorem. The sampling theorem states that if f_m is the maximum frequency component in the analog signal, then the information present in the signal can be represented by its sampled version, provided the number samples taken per second is greater than or equal to twice the maximum frequency component. The number of samples/second is more commonly termed as sampling frequency f_s. Hence, according to sampling theorem, f_s should be greater than or equal to $2*f_m$.

The speech signal has frequency components in the audio frequency range (20 Hz to 20 kHz) of the electromagnetic spectrum. This is the reason for perceiving the information present in the speech signal by human ears. The fundamental question is upto what range of audio frequency, the speech signal has frequency components. We can analyze this experimentally by considering the whole audio range. The standard sampling frequency to sample the entire audio range is 44.1 kHz. This is because, 20 kHz is the maximum frequency component and allowing some guard band, the sampling frequency has been set at 44.1 kHz. In this lab we to use a sampling frequencies less than 44.1kHz and analyse the speech signal. We try to find an optimal sampling frequency to minimize the number of total samples and retain the intelligibe information of the audio.

Bit resolution for speech

After the sampling frequency, the next important parameter in the digitization process of speech is bit resolution. The number of bits used for storing each sample of speech is termed as bit resolution. The number of bits/sample in turn depends on the number of quantization levels used during analog to digital conversion. More the number of quantization levels, finer will be the quantization step and hence better will be the information preserved in the digitized form. However, more will be the requirement of number of bits/sample. Hence it is a trade off between the number of bits and information representation. The effect of bit resolution can be analyzed experimentally. For this experiment the optimal sampling frequency of 16 kHz can be used as proposed earlier.

All speech signal processing applications invariably use 16 bits/sample as bit resolution. The number of quantization levels will therefore be $2^{16}=65536$ and are found to be optimal for preserving information present in the analog version of the speech signal. The next lower bit length possible with binary power is 8 bits. With 8 bits, the number of quantization levels will be $2^{8}=256$. As it can be observed, the number of quantization levels are significantly lower compared to the 16 bit case and hence poor representation of information in the quantized signal. In this lab we observe this phenomina using a practical example.

Problem A

Study of Sampling Frequency

1. Record the word ‘Speech’ using a sampling frequency of 44.1kHz save it in a .wav file. Plot the complete speech signal and the frequency spectrum for different sounds.
2. Resample the above speech signal to 16kHz and plot the complete speech signal along with the frequency spectra for different sounds. Comment on the intelligibility of the speech signal sampled at 16kHz as compared to the speech signal sampled at 44.1kHz and whether this is a good choice for the sampling frequency.
3. Resample the speech signal obtained from (a) to 8kHz and plot the complete speech signal along with the frequency spectra for different sounds. Comment on the intelligibility of the speech signal sampled at 8kHz comparing it to the above signals and whether this is a good choice for the sampling frequency.
4. Resample the speech signal obtained from (a) to 4kHz and plot the complete speech signal along with the frequency spectra for different sounds. Comment on the intelligibility of the speech signal sampled at 4kHz comparing it to the above signals and whether this is a good choice for the sampling frequency.

Procedure

1. Record the word 'Speech' using wavesurfer in 44.1kHz sampling frequency, save the recoring in .wav format and upload it in drive and access it in colab.
2. Plot the time domain plot of the audio and extract the individual sound components of the audio and plot its corresponding magnitude spectrum.
3. Resample the original audio to 16kHz, 8kHz and 4kHz and plot the time domain plot and individual sound components magnitude spectrum in each case.
4. Save the resampled audio in .wav format in google drive and comment on the intelleigibility of the audio.

In [51]:
```
# Mounting Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

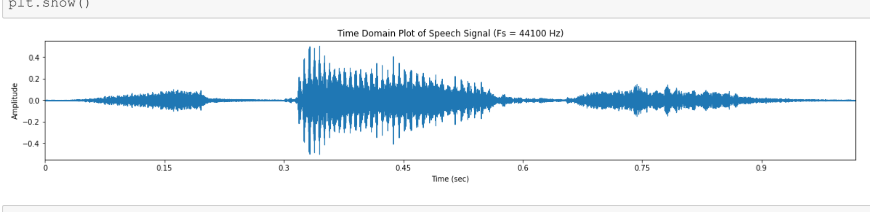
In [2]:
```
# Changing directory
%cd /content/gdrive/MyDrive/Sem6/Speech Lab/Week3
!ls
```

```
/content/gdrive/MyDrive/Sem6/Speech Lab/Week3
audio_16k.wav  audio_4k.wav  audio_8k.wav  Lab3.ipynb  week3audio.wav
```

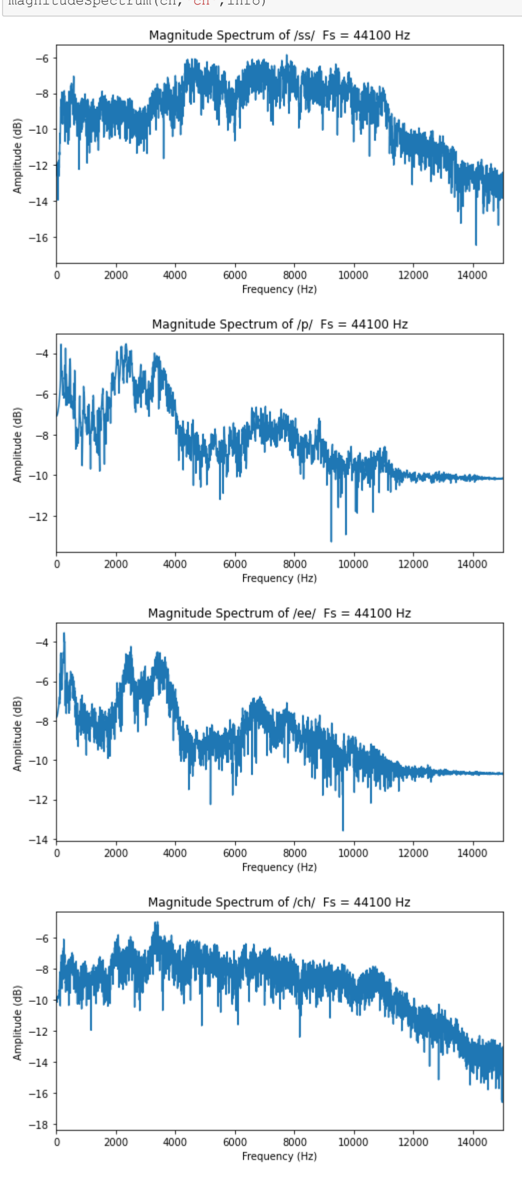
In [9]:
```
# Importing Libraries
import numpy as np
from matplotlib import pyplot as plt
from scipy.fft import fft, fftfreq,fftshift
from scipy import signal
from scipy.io import wavfile
import librosa
import librosa.display
import soundfile as sf

#Functions

# Extracting different categories of sound in the speech
# The time stamp for each sound component was extracted from wavesurfer and they
# are as follows:
# /ss/ - 0.030 s to 0.298 s
# /p/ - 0.298 s to 0.405 s
# /ee/ - 0.405 s to 0.620 s
# /ch/ - 0.620 s to 0.980 s
def extractSound(audio,fs):
  ss = audio[int(0.03*fs):int(0.298*fs)]
  p = audio[int(0.298*fs):int(0.405*fs)]
  ee = audio[int(0.405*fs):int(0.620*fs)]
  ch = audio[int(0.620*fs):int(0.98*fs)]
  return ss,p,ee, ch

# Magnitude spuctrum plot function
def magnitudeSpectrum(sound, sound_name,sound_info):

  # Computing the FFT of the sound
  sound_len = sound.shape[0]
  sound_fft = fft(sound)/sound_len

  # Computing the frequency array
  freqs = fftfreq(sound_len, 1/fs)

  # Plotting graph
  plt.figure(figsize=(8,4))
  plt.plot(freqs[0:sound_len//2], 2*np.log10(np.abs(sound_fft[0:sound_len//2])))
  plt.title("Magnitude Spectrum of "+ "/" + sound_name + "/ " + sound_info)
  plt.xlim((0, 15000))
  plt.xlabel('Frequency (Hz)')
  plt.ylabel('Amplitude (dB)')
  plt.show()
```

Time domain plot and magnitude spectrum of sound components of the audio at 44.1kHz

In [29]:
```
# Loading the audio into colab. Fs = 44.1kHz
audio, fs = librosa.load("week3audio.wav", sr = 44100)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio, sr=fs);
plt.title("Time Domain Plot of Speech Signal (Fs = " +str(fs)+" Hz)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()
```



In [22]:
```
# Extracting the individual sound components
ss,p,ee,ch = extractSound(audio,fs)

# Plotting the magnitude spectrum
info = '  Fs = ' + str(fs)+ ' Hz'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```

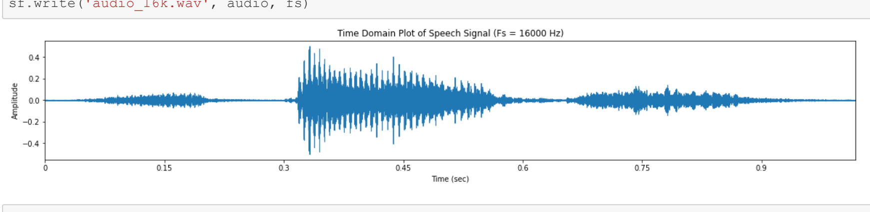


Observation (for fs = 44.1kHz)

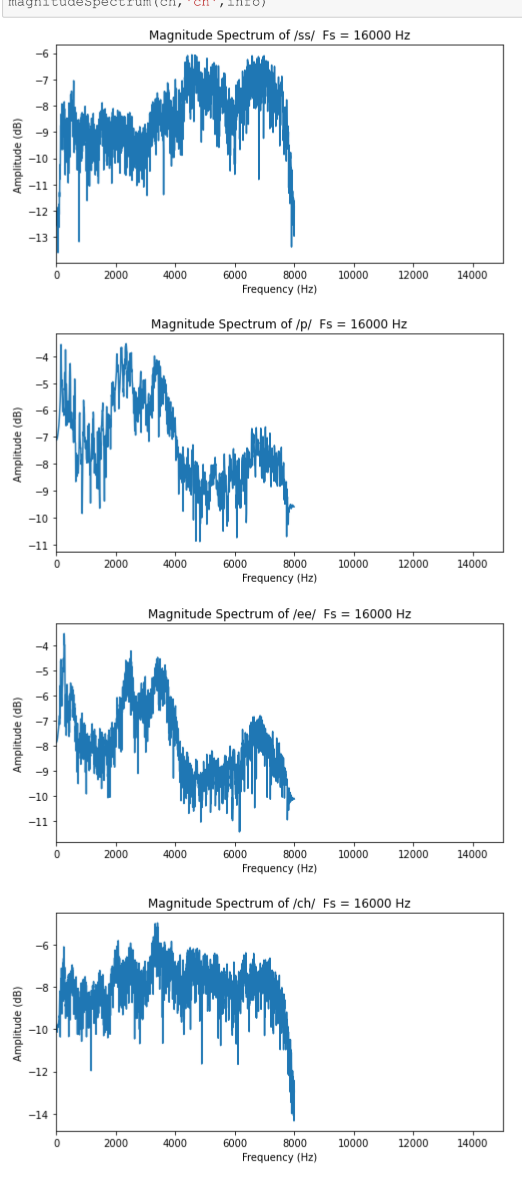
From the above magnitude spectrum we observe that, there are no significant frequency components in the spectrum beyond about 7.5 kHz (less than -10dB). This observation shows that 44.1 kHz sampling is too high value to capture the information present in the speech signal. Since information is upto about 7.5 kHz, Fs = 16 kHz seems to be optimal. In the next subpart we analyse the magnitude spectrum using Fs=16kHz.

Time domain plot and magnitude spectrum of sound components of the audio at 16kHz

In [30]:
```
# Loading the audio into colab. Fs = 16kHz
audio, fs = librosa.load("week3audio.wav", sr = 16000)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio, sr=fs);
plt.title("Time Domain Plot of Speech Signal (Fs = " +str(fs)+" Hz)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_16k.wav', audio, fs)
```



In [24]:
```
# Extracting the individual sound components
ss,p,ee,ch = extractSound(audio,fs)

# Plotting the magnitude spectrum
info = '  Fs = ' + str(fs)+ ' Hz'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```

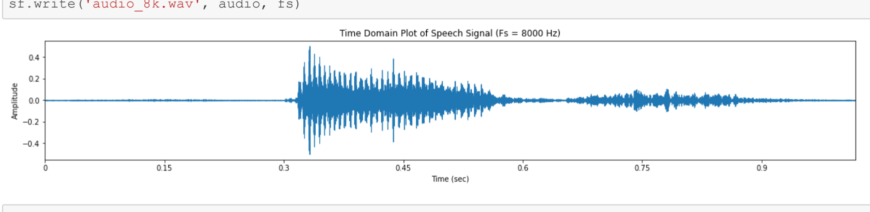


Observation (for fs = 16kHz)

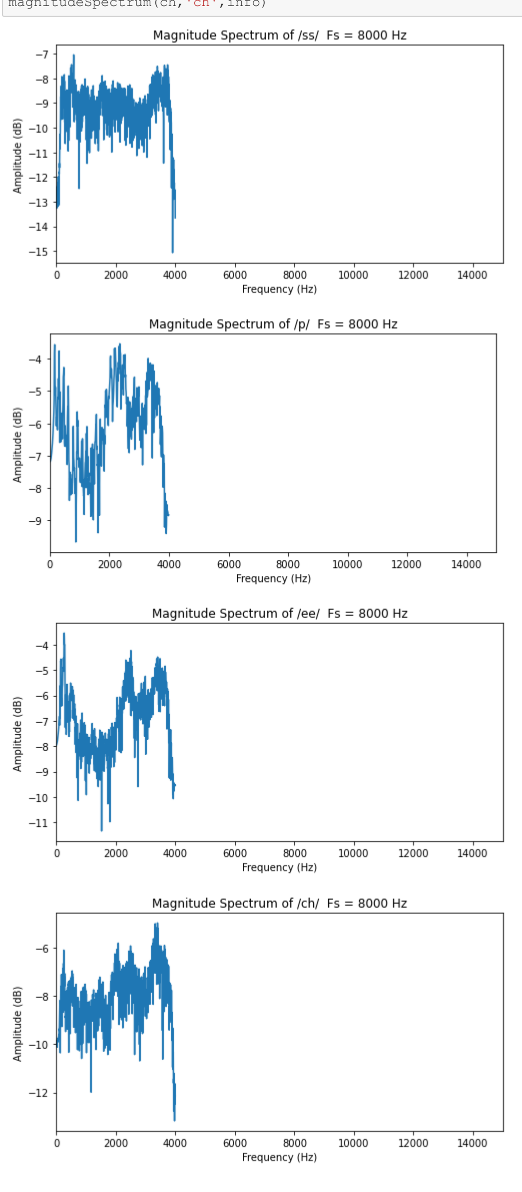
From the above magnitude spectrum we can observe that the speech has no significant frequencies above 8kHz and the resampled audio saved in .wav format is interpretable when played. Hence, Fs = 16kHz looks an optimal sampling frequency.

Time domain plot and magnitude spectrum of sound components of the audio at 8kHz

In [31]:
```
# Loading the audio into colab. Fs = 8kHz
audio, fs = librosa.load("week3audio.wav", sr = 8000)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio, sr=fs);
plt.title("Time Domain Plot of Speech Signal (Fs = " +str(fs)+" Hz)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_8k.wav', audio, fs)
```



In [26]:
```
# Extracting the individual sound components
ss,p,ee,ch = extractSound(audio,fs)

# Plotting the magnitude spectrum
info = '  Fs = ' + str(fs)+ ' Hz'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```



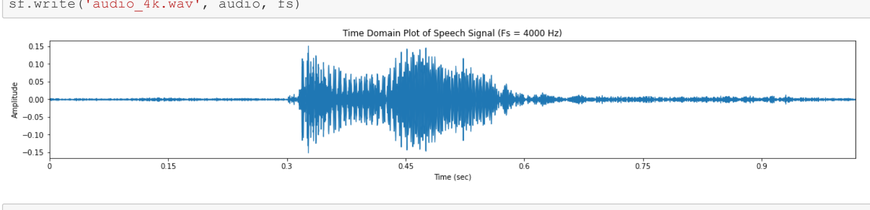
Observation (for fs = 8kHz)

We can be observed from the above magnitude spectrum that information beyond 4 kHz is eliminated. The speech signals that have bandwith 4kHz is classified as telephone bandwidth speech.

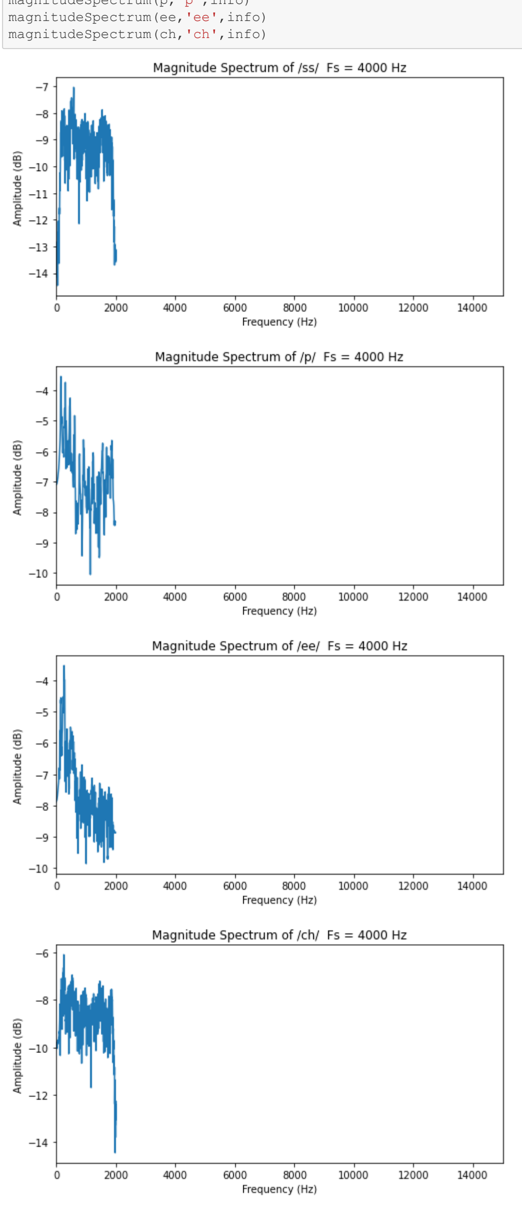
In case of telephone bandwidth speech even though the sampling frequency seems to be fine for sounds like a and aa, it severely affects other sounds like s and ch (frecatives), as frecatives have high frequency components in it and these components are lost during the sampling process. However, information upto 4 kHz bandwidth seem to be sufficient for intelligible speech. If the sampling frequency is further decreased from 8 kHz, then intelligibility of speech degrades significantly. This can be observed in the next subpart. Hence 8 kHz was chosen as the sampling frequency for the telephone communication.

Time domain plot and magnitude spectrum of sound components of the audio at 4kHz

In [37]:
```
# Loading the audio into colab. Fs = 4kHz
audio, fs = librosa.load("week3audio.wav", sr = 4000)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio, sr=fs);
plt.title("Time Domain Plot of Speech Signal (Fs = " +str(fs)+" Hz)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_4k.wav', audio, fs)
```



In [28]:
```
# Extracting the individual sound components
ss,p,ee,ch = extractSound(audio,fs)

# Plotting the magnitude spectrum
info = '  Fs = ' + str(fs)+ ' Hz'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```



Observation (for fs = 4kHz)

As discussed in the previous subpart.If the sampling frequency is further decreased from 8 kHz, then intelligibility of speech degrades significantly, this can be observed in case of Fs = 4kHz. Since the bandwidth is limited to 2000Hz, the information of frecatives are lost, this is evident in the time domain plot. The /s/ and /sh/ sound information is lost. By listening to the four files, namely, file sampled at 44.1 kHz, 16 kHz 8 kHz, and 4kHz one can observe difference in the naturalness between the speech sampled at 4 kHz and other higher sampling frequencies. **Thus wherever possible, the speech signal sampled 16 kHz should be used for signal processing for better results.**

Problem B

Study of Bit Resolution

1. For this study, record the word ‘Speech’ using a sampling frequency of 16kHz and a bit resolution of 16 bits/sample. Plot the complete speech signal and the frequency spectrum of different sounds using the bit resolution of 16 bits/sample.
2. Now, using the same speech signal sampled at 16kHz, now use a bit resolution of 8 bits/sample. Plot the complete speech signal and the frequency spectrum of different sounds. Comment on the frequency spectrum of the different sounds, intelligibility and quality of the speech signal comparing it with the above speech signal obtained using 16 bits/sample
3. Using the same speech signal sampled at 16kHz, now use a bit resolution of 4 bits/sample. Plot the complete speech signal and the frequency spectrum of different sounds. Comment on the frequency spectrum of the different sounds, intelligibility and quality of the speech signal comparing it with the above speech signals.
4. Using the same speech signal sampled at 16kHz, now use a bit resolution of 1 bit/sample. Plot the complete speech signal and the frequency spectrum of different sounds. Comment on the frequency spectrum of the different sounds, intelligibility and quality of the speech signal comparing it with the above speech signals.

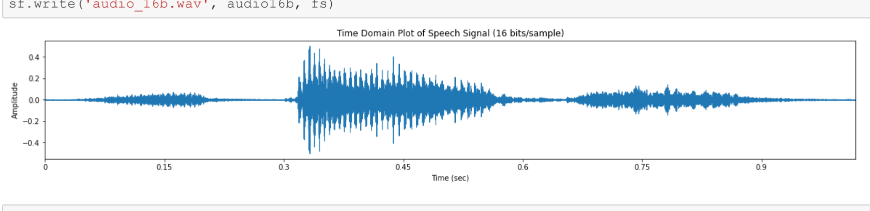
Procedure

1. Record the word 'Speech' using wavesurfer in 44.1kHz sampling frequency, save the recoring in .wav format and upload it in drive and access it in colab.
2. Load the audio using librosa with sampling frequency 16kHz.
3. Change the bit resoluting of the audio for 16 bits/sample,8 bits/sample,4 bit/sample, and 1 bit/sample.
4. Incase of 1bit/sample, we take any +ve amplitude sample as 1 and others as 0.
5. Plot the time domain plot of the audio and extract the individual sound components of the audio and plot its corresponding magnitude spectrum.
6. Save the new audio files in .wav format.

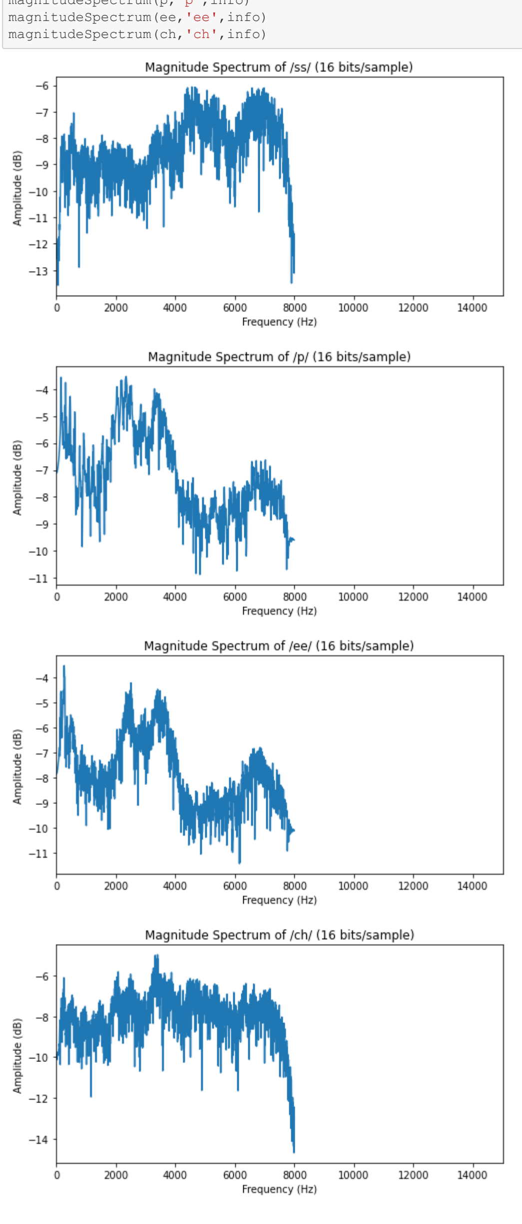
In [8]:
```
# Functions
# Function to change the bit resolution of a audio signal to n bits/sample
def rounder(audio, n=16):
  maxVal = np.max(np.abs(audio))
  temp = audio/maxVal
  temp *= 2**(n-1)
  temp = np.around(temp)
  temp /= 2**(n-1)
  resampledAudio = temp * maxVal
  return resampledAudio
```

Time domain plot and magnitude spectrum of sound components of the audio at 16bits/sample

In [44]:
```
# Loading the audio into colab. Fs = 16kHz
audio, fs = librosa.load("week3audio.wav", sr = 16000)

# Changing the bit resolution to 16 bits/sample
audio16b = rounder(audio,16)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio16b, sr=fs);
plt.title("Time Domain Plot of Speech Signal (16 bits/sample)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_16b.wav', audio16b, fs)
```

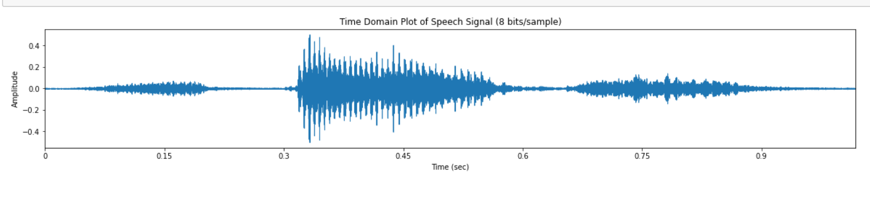


In [45]:
```
# Extracting the individual sound components
ss,p,ee,ch = extractSound(audio16b,fs)

# Plotting the magnitude spectrum
info = ' (16 bits/sample)'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```

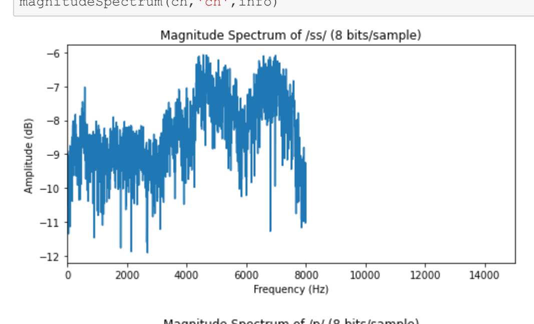


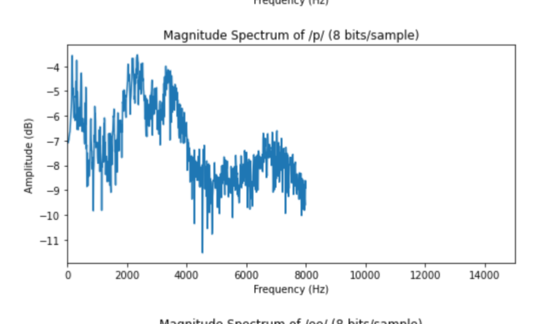
Time domain plot and magnitude spectrum of sound components of the audio at 8 bits/sample

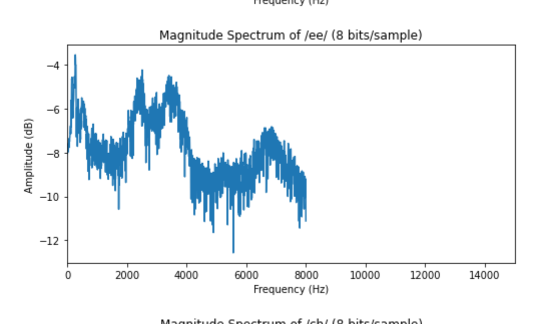
In [46]:
```
# Changing the bit resolution to 8 bits/sample
audio8b = rounder(audio,8)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio8b, sr=fs);
plt.title("Time Domain Plot of Speech Signal (8 bits/sample)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_8b.wav', audio8b, fs)
```

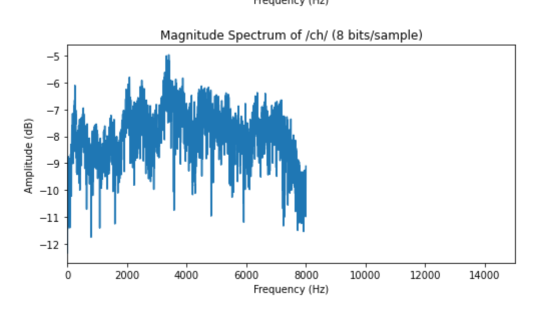


```
In [47]: # Extracting the individual sound components
ss,p,ee,ch = extractSound(audio8b,fs)

# Plotting the magnitude spectrum
info = ' (8 bits/sample)'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```







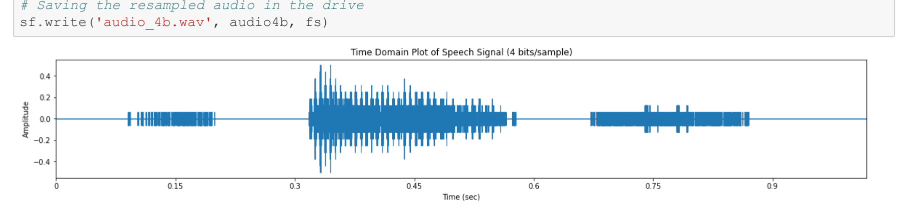


Observation (for 8 bits/sample)

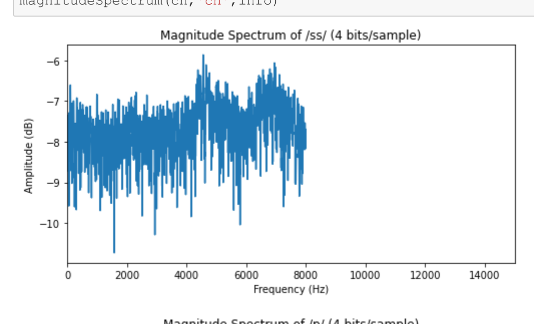
- The magnitude spectrum for 8 bits/sample is very similar to that of 16 bits/samples.
- The audio saved is easily comprehensible to the human ears.

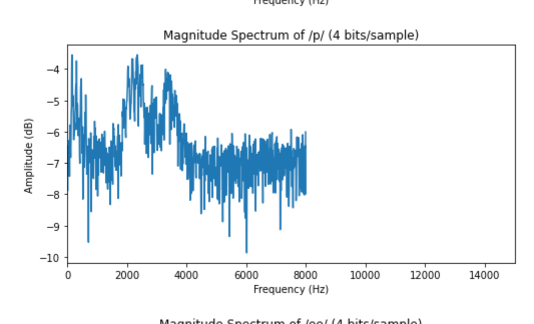
Time domain plot and magnitude spectrum of sound components of the audio at 4 bits/sample

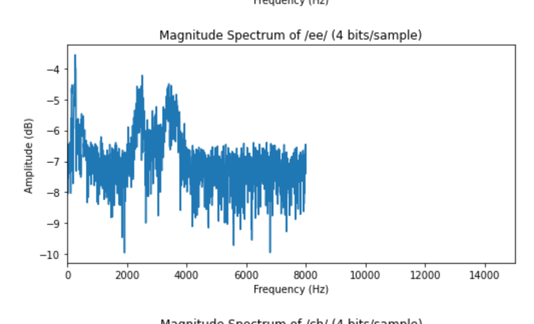
```
In [48]: # Changing the bit resolution to 4 bits/sample
audio4b = rounder(audio,4)

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio4b, sr=fs);
plt.title("Time Domain Plot of Speech Signal (4 bits/sample)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_4b.wav', audio4b, fs)
```

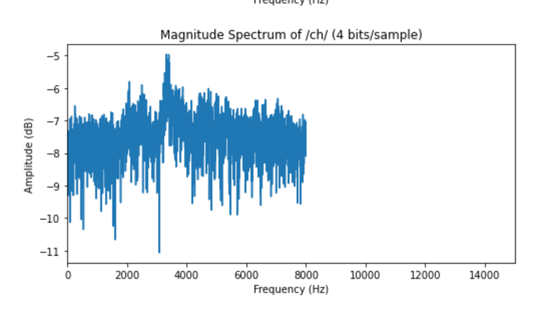



```
In [49]: # Extracting the individual sound components
ss,p,ee,ch = extractSound(audio4b,fs)

# Plotting the magnitude spectrum
info = ' (4 bits/sample)'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```







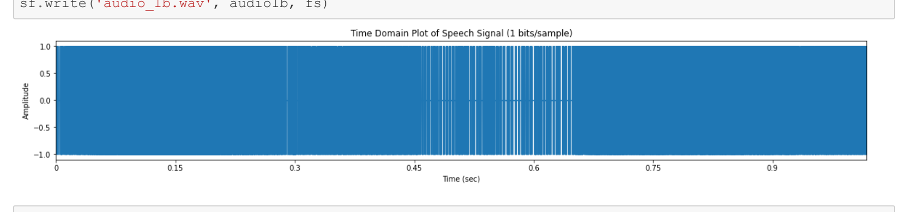


Observation (for 4 bits/sample)

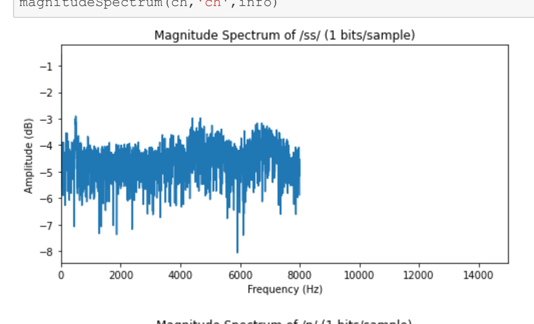
- As the quantization level is reduced significantly low when compaired to 16 bits/sample, we observe quantization noise in the magnitude spectrum.
- In the save audio is comprehensible to the human ears but has significant noise due to very few quantization level.
- We can observe /s/ and /sh/ is getting truncated significantly in the time domain plot.
- Due to these reasons we prefer to use 16 bits/sample for signal processing.

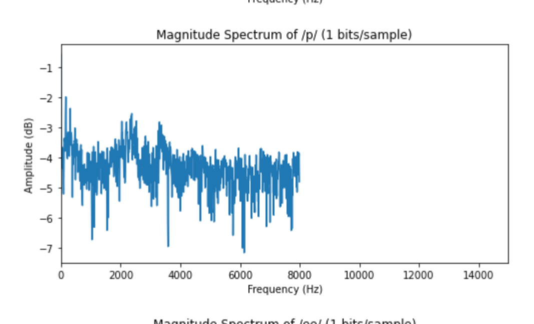
Time domain plot and magnitude spectrum of sound components of the audio at 1 bits/sample

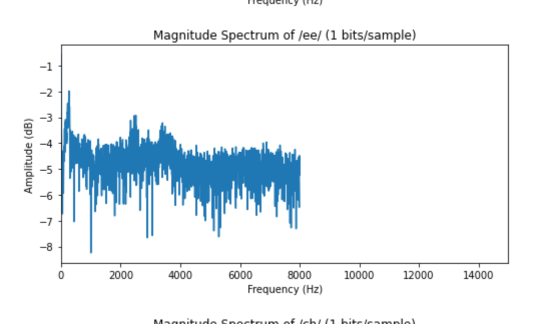
```
In [50]: # Setting the bit resolution to 1 bit/sample
bins = np.array([0., 1.])
audio1b = bins[np.digitize(audio, bins)]

# Plotting time domain plot of the audio
plt.figure(figsize=(20,3))
librosa.display.waveplot(audio1b, sr=fs);
plt.title("Time Domain Plot of Speech Signal (1 bits/sample)")
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.show()

# Saving the resampled audio in the drive
sf.write('audio_1b.wav', audio1b, fs)
```

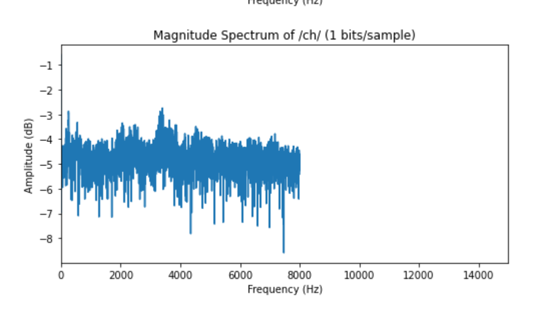



```
In [52]: # Extracting the individual sound components
ss,p,ee,ch = extractSound(audio1b,fs)

# Plotting the magnitude spectrum
info = ' (1 bits/sample)'
magnitudeSpectrum(ss,'ss',info)
magnitudeSpectrum(p,'p',info)
magnitudeSpectrum(ee,'ee',info)
magnitudeSpectrum(ch,'ch',info)
```









Observation (for 1bit/sample)

- The over all magnitude of magnitude spectrum has increased for all the sound components.
- Since there are only 2 discrete levels, the amplitude information is lost and the small insignificant background noise is amplified.
- The audio is very noisy but it is sill comprehensible.
- **Hence we can conclude that 16 bits/sample bit resolution is the prefered as it retains most of the information. 8 bits/sample can be used in cases if we have limited storage capacity.**

References and Tools

1. For theory concepts :- https://vlab.amrita.edu/index.php?sub=59&brch=164&sim=474&cnt=1
2. Basics Floating-Point Arithmetic (IEEE 754) :- https://en.wikipedia.org/wiki/IEEE_754
3. Wavesurfer:- https://sourceforge.net/projects/wavesurfer/