**멀티미디어통신 – 오디오실습**

201620350 김지영

**1. 유성음과 무성음의 파형의 특성을 기술하시오.**

**Code**

import scipy

from scipy.io import wavfile

import matplotlib

from matplotlib import pyplot as plt

import numpy as np

sample\_rate, X = scipy.io.wavfile.read('느.wav')

plt.plot(X)

plt.show()

plt.plot(X[31000:35000])

plt.show()

sample\_rate, X = scipy.io.wavfile.read('트.wav')

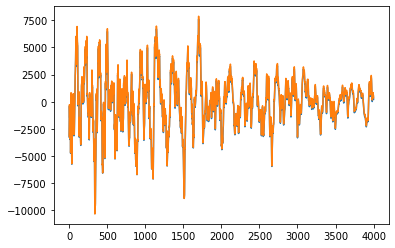
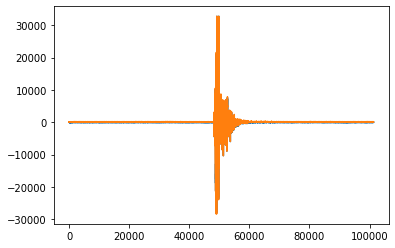
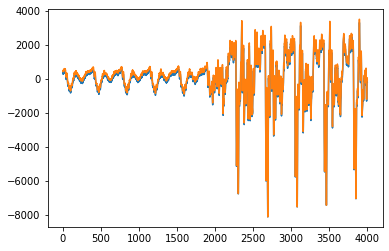
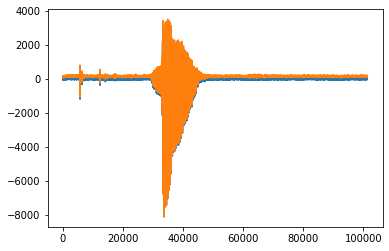
plt.plot(X)

plt.show()

plt.plot(X[51000:55000])

plt.show()

**실행 결과**



**해석**

발성할 때 성대의 진동을 유발하는 소리는 유성음이고, 진동 없이 성대를 통과하는 소리는 무성음입니다. 유성음에는 ‘ㄴ, ㅁ, ㅇ, ㄹ’ 같은 대부분의 모음이 그 역할을 하며 성대의 진동 속도에 따라 준주기적인 파형 신호를 띄게됩니다. 위의 실행결과에서 첫번째 행을 보면, ‘느’ 발음이 나는 순간에 파형이 나타나며 무성음보다는 주기를 가진 준주기적 파형 모습을 나타냈습니다. 반면에, 두번째 행을 보면, ‘트’ 발음이 나는 동안에는 다양한 주파수 성분이 고르게 포함된 백색소음과 같은 파형 모양을 나타냈습니다.

**2. ‘보고할 시간입니다.wav’와 ‘samul.wav'에 대해서 파형의 특성과 spectrogram의 특성을 기술 하시오.**

**Code**

import scipy

from scipy.io import wavfile

import matplotlib

from matplotlib import pyplot as plt

import numpy as np

sample\_rate, X = scipy.io.wavfile.read('보고할시간입니다.wav') # 1 channel

print(sample\_rate)

plt.plot(X)

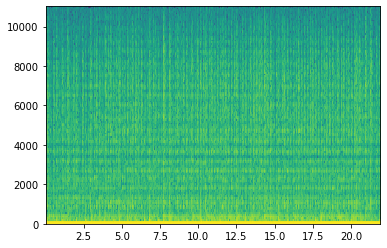
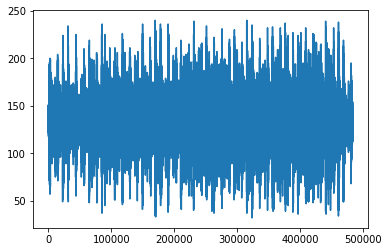
plt.show()

plt.specgram(X, Fs=sample\_rate) # Spectrogram

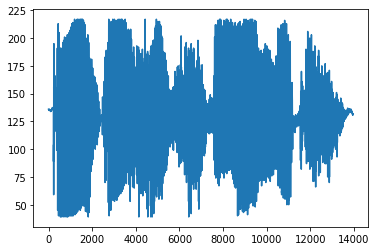
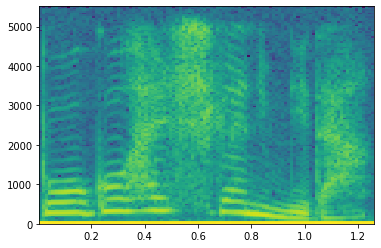
plt.show()

**실행 결과**

(1) 보고할시간입니다.wav



(2) SAMUL.wav

**해석**

(1)과 (2)의 파형을 먼저 살펴보면, (1)은 처음부터 끝까지 일관성있는 기계목소리로 무성음과 같이 주기가 없는 파형을 보였고, (2)는 사물놀이 소리가 들리며 반복적으로 큰소리가 나는 파형이 그려졌습니다.

스펙트럼을 분석해보면, 주파수가 밝은 노란색 열로 나타난 것을 확인할 수 있습니다.

**3. ‘보고할 시간입니다.wav’와 ‘samul.wav'를 low-pass filtering하고 spectrogram의 변화를 관찰하시오. 또한 소리의 변화는 어떠한가?**

**Code**

import scipy

from scipy.io import wavfile

from scipy import signal

import matplotlib

from matplotlib import pyplot as plt

import numpy as np

# import module in subpackage (package.subpackage.module – scipy.io.wavfile)

sample\_rate, x = scipy.io.wavfile.read('SAMUL.wav') # 1 channel, 8 bits/sample

print(sample\_rate)

plt.plot(x)

plt.show()

plt.plot(x[3000:4000])

plt.show()

plt.specgram(x, Fs=sample\_rate) # Spectrogram

plt.show()

down = 3.0

h = signal.firwin(32, 1. / down, window='hamming') # Lowpass filter

num\_fft=512

fr\_f=np.fft.fft(h,n=num\_fft); # FFT (fast Fourier transform) of lowpass filter

famp\_f=np.absolute(fr\_f); # Amplitude of frequency response for lowpass f ilter

famp\_fdB=20\*np.log10(famp\_f) # Amplitude of frequency response for lowpass (dB scale)

matplotlib.pyplot.stem(h) # Plot lowpass filter in time domain

matplotlib.pyplot.show()

matplotlib.pyplot.plot(famp\_f) # Plot frequency response of lowpass filter

matplotlib.pyplot.show()

matplotlib.pyplot.plot(famp\_fdB) # Plot frequency response of lowpass filter (dB scale)

matplotlib.pyplot.show()

y = np.convolve(x,h) # Lowpass filtering

plt.plot(y)

plt.show()

plt.plot(y[3000:4000])

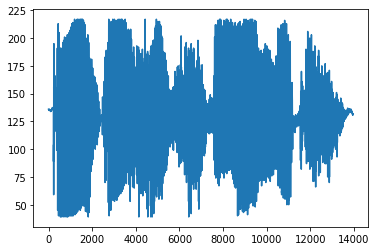
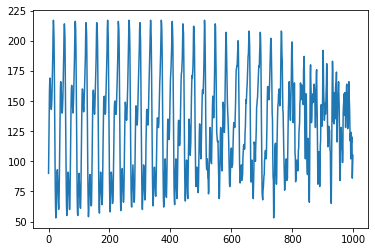
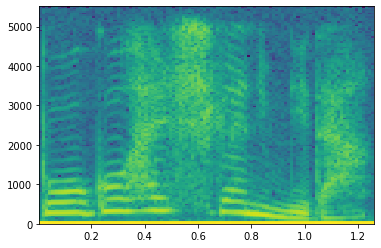
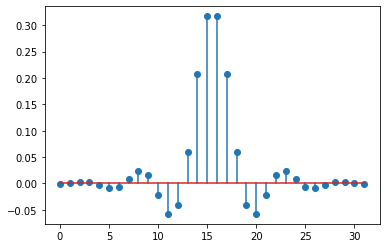
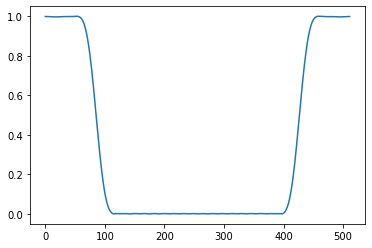
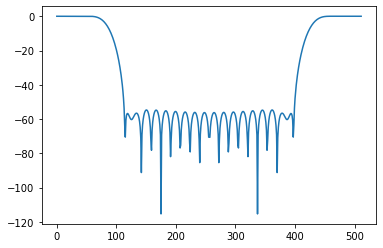
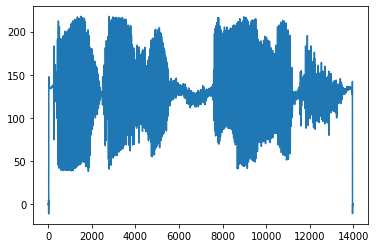
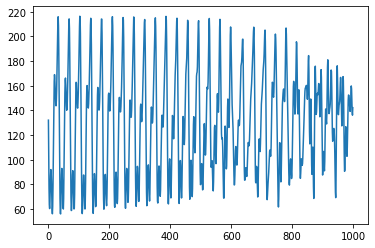
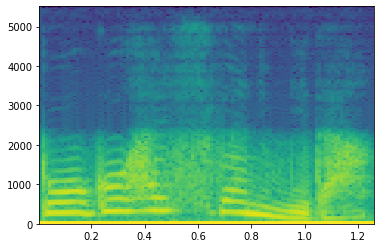
plt.show()

plt.specgram(y, Fs=sample\_rate) # Spectrogram

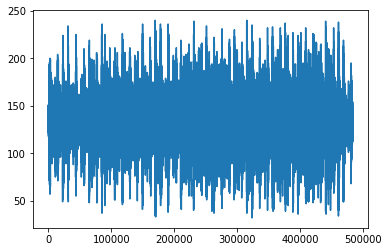
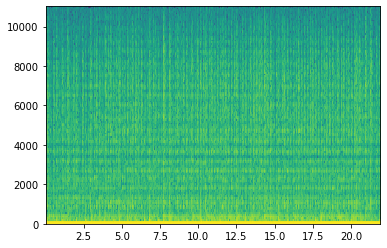
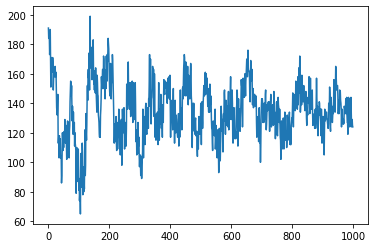
plt.show()

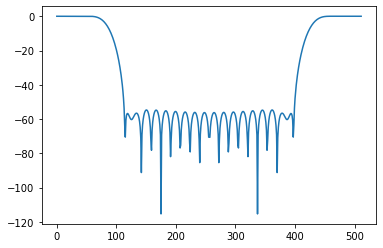
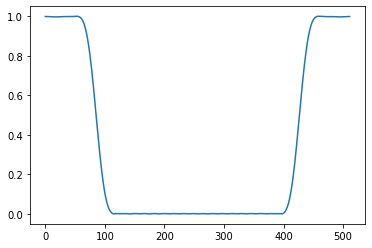
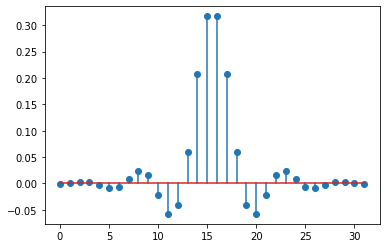
**실행 결과**

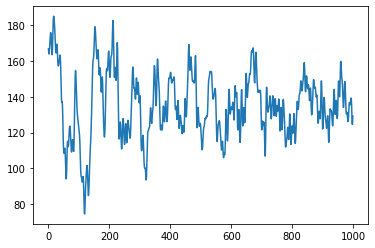
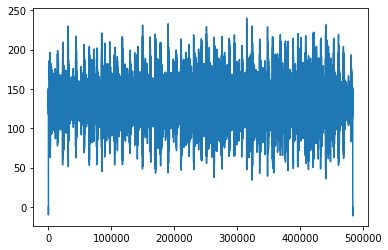
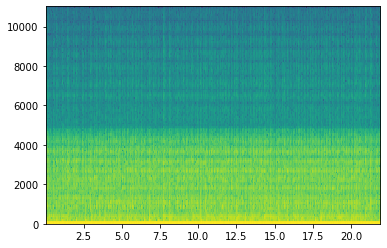
(1) 보고할 시간입니다.wav

(2) SAMUL.wav



**해석**

(1)과 (2)모두 filtering한 결과 원본 음성신호와 모양이 달라 소리가 일부 손상된 것을 확인할 수 있습니다. 또한, 두 신호 모두 low-pass filtering을 했기 때문에 spectrogram 결과에서 높은 주파수성분 대신 주파수가 낮은 부분이 노란색으로 표시되었습니다.