Activity No. 2	
Noise Generation and Analysis	
Course Code: CPE 027	**Program:** BSCpE
Course Title: Digital Signal Processing and Applications	**Date Performed:** 09/23/2021
Section: CPE 027-CPE41S2	**Date Submitted:** 09/23/2021
Name/s: de Vera, Altz Kienn Galit, Adrian Hernandez, Brenda Raper, Giovanni Villareal, Ervie John	**Instructor:** Engr. Cris Paulo Hate

1. Objective:

This activity deals with the programmatic creation of a noise signal by analyzing and quantifying properties in a pre-existing dataset.

2. Intended Learning Outcomes (ILOs):

After completion of this activity the students should be able to:

Develop a program that manipulates the statistical properties of signals and noise.

3. Discussion :

Noisy data is meaningless data. The term has often been used as a synonym for corrupt data. However, its meaning has expanded to include any data that cannot be understood and interpreted correctly by machines, such as unstructured text. Any data that has been received, stored, or changed in such a manner that it cannot be read or used by the program that originally created it can be described as noisy.

Noisy data unnecessarily increases the amount of storage space required and can also adversely affect the results of any data mining analysis. Statistical analysis can use information gleaned from historical data to weed out noisy data and facilitate data mining.

Noisy data can be caused by hardware failures, programming errors and gibberish input from speech or optical character recognition (OCR) programs. Spelling errors, industry abbreviations and slang can also impede machine reading.

4. Resources:

The activity will require the following software, tools and equipment:

- Python and Libraries
- Anaconda Navigator
- Jupyter
- Browse
- Markdown

5. Directions:

1. Develop a program to extract data from the given wav files, **snsd.wav and obama2.wav**.
2. Capture the first few seconds of the audio signals and apply an aliasing filter to it.
3. Reconstruct the signal using a preferred DAC method.
3. Compare and contrast the two versions (original and aliased) using descriptive and statistical means.
4. Provide an analysis of the results.

6. Procedures

1. Develop a program to extract data from the given wav files, snsd.wav and obama2.wav.

a. Import libraries

In [7]:
```
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import animation
import scipy.signal as signal
import numpy as np
import wave
import sys

from IPython.display import Audio, display, HTML
from ipywidgets import interact

from scipy.io import wavfile
import requests

%matplotlib inline
matplotlib.rcParams['animation.writer'] = 'avconv'
matplotlib.rcParams['figure.figsize'] = "8,3"
```

Figure 6.1
This figure shows how to import all the necessary libraries

b. Extract the audio

In [9]:
```
def loadAudio(url, start, length):

    rate, data = wavfile.read(url)
    if len(data.shape) > 1:
        data = data.sum(axis=1)
    data = (1.0 * data / abs(data).max()).astype(np.float32)
    return rate, data[rate*start+np.arange(rate*length)]

url_voice = "drive-download-20210922T100705Z-001/obama2.wav"
url_music =  "drive-download-20210922T100705Z-001/snsd.wav"
rate_voice, data_voice = loadAudio(url_voice, 0, 15)
rate_music, data_music = loadAudio(url_music, 0, 15)
```

Figure 6.2
This figure shows how to import the experimented audio's

2. Capture the first few seconds of the audio signals and apply an aliasing filter to it.

In [10]:
```python
def presentAliasingAudio(original, rate, factor):
    down_aliased = original[::factor] # dumb downsampling, no anti-aliasing
    b = signal.firwin(155, 1.0/factor-0.01); a=1  # design the AAF
    lowpass = signal.lfilter(b, a, original)      # apply the AAF
    down_nonaliased = lowpass[::factor]           # perform Downsampling

    '''
    display(HTML("Original:"), Audio(data=original, rate=rate))
    print(np.mean(original))
    plt.figure(1)
    plt.title("Signal Wave...")
    print("mean power: " + str(np.mean(original)))
    print("deviation: " + str(np.std(original)))
    plt.plot(original)
    plt.show()
    '''

    display(HTML("With Aliasing:"), Audio(data=down_aliased, rate=rate/factor))
    #print("mean power: " + str(np.mean(down_aliased)))
    #print("deviation: " + str(np.std(down_aliased)))
    #plt.figure(1)
    #plt.title("Signal Wave...")
    #plt.plot(down_aliased)
    #plt.show()
```

Figure 6.3
This figure shows the limit the audio file to a few seconds to apply the aliasing filter

a. aliased obama2.wav

In [11]:
```python
presentAliasingAudio(data_voice, rate_voice, 6)
```

With Aliasing:

Figure 6.4
This figure shows First aliased audio, obama2.wav

b. aliased snsd.wav

In [12]:
```python
presentAliasingAudio(data_music, rate_music, 6)
```

With Aliasing:

Figure 6.5
This figure shows Second aliased audio, snsd.wav

4. Compare and contrast the two versions (original and aliased) using descriptive and statistical means.

In [13]:
```
def compareOriginalAliased(original, rate, factor):
    down_aliased = original[::factor] # dumb downsampling, no anti-aliasing
    #b = signal.firwin(155, 1.0/factor-0.01); a=1  # design the AAF
    #lowpass = signal.lfilter(b, a, original)      # apply the AAF
    #down_nonaliased = lowpass[::factor]           # perform Downsampling

    display(HTML("Original:"), Audio(data=original, rate=rate))
    print(np.mean(original))
    plt.figure(1)
    plt.title("Signal Wave...")
    print("mean power: " + str(np.mean(original)))
    print("deviation: " + str(np.std(original)))
    plt.plot(original)
    plt.show()

    display(HTML("Aliased:"), Audio(data=down_aliased, rate=rate/factor))
    print("mean power: " + str(np.mean(down_aliased)))
    print("deviation: " + str(np.std(down_aliased)))
    plt.figure(2)
    plt.title("Signal Wave...")
    plt.plot(down_aliased)
    plt.show()
```

Figure 6.6

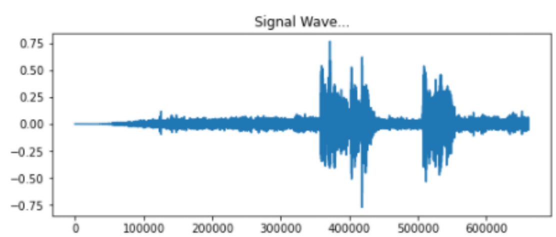
This figure shows comparison of the unfiltered and filtered with aliasing

a. original and aliased obama2.wav

In [14]:
```
compareOriginalAliased(data_voice, rate_voice, 6)
```

Original:


```
2.2859835e-05
mean power: 2.2859835e-05
deviation: 0.052557915
```



Aliased:


```
mean power: 5.4189262e-05
deviation: 0.052600697
```

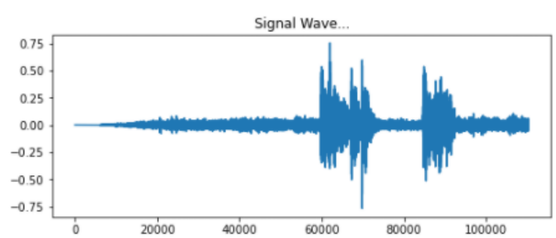


Figure 6.7

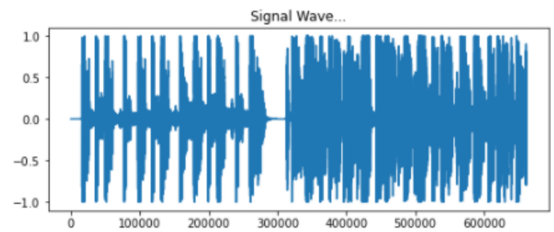
This figure shows the comparison between the obama2.wav, filtered and original

b. original and aliased snsd.wav

```
In [15]: compareOriginalAliased(data_music, rate_music, 6)
```

Original:

```
2.5627176e-05
mean power: 2.5627176e-05
deviation: 0.27828807
```



Aliased:

```
mean power: 1.9564466e-05
deviation: 0.27834904
```

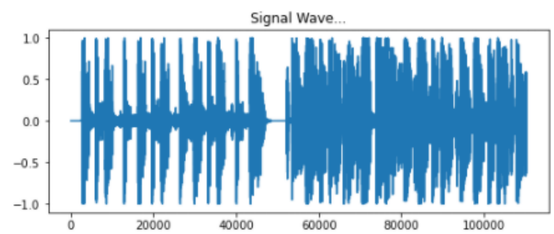
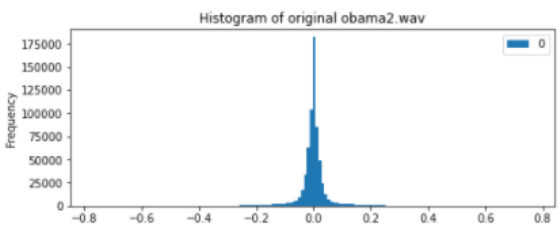


Figure 6.8

This figure shows the comparison between the snsd.wav, filtered and original

Compare and Contrast of obama2.wav

In [20]:
```
import pandas as pd

factor = 6
down_aliased = data_voice[::factor]

# orignal
# Calling DataFrame constructor on list
df = pd.DataFrame(data_voice)
Hist_title = "Histogram of original obama2.wav"
df.plot(kind='hist', title=Hist_title, bins = 150)
plt.savefig(Hist_title+'.png', facecolor='white', transparent=False) # saving the plot into png file
plt.show()
plt.clf() # to clear the current plot to make new

# aliased
# Calling DataFrame constructor on list
df = pd.DataFrame(down_aliased)
Hist_title = "Histogram of aliased obama2.wav"
df.plot(kind='hist', title=Hist_title, bins = 150)
plt.savefig(Hist_title+'.png', facecolor='white', transparent=False) # saving the plot into png file
plt.show()
plt.clf() # to clear the current plot to make new
```



<Figure size 576x216 with 0 Axes>

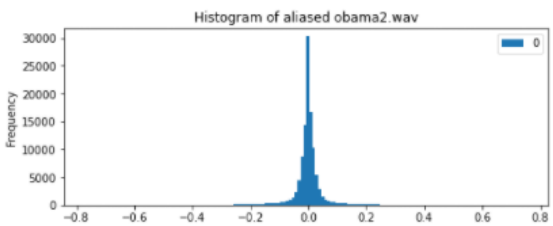
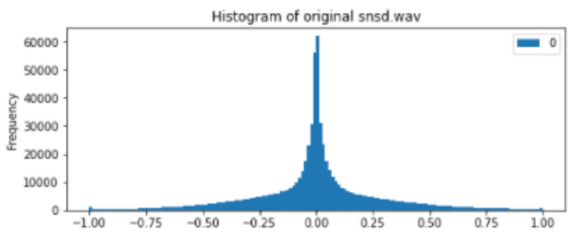


Figure 6.9

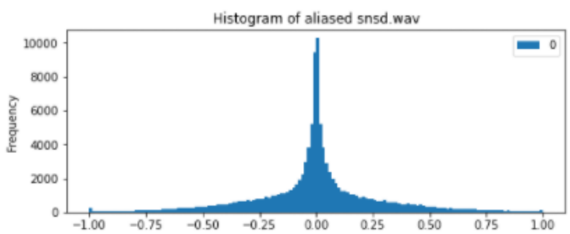
This figure shows the comparison of the histogram of the aliased obama2.wav and original, also the saving of the histogram file in the directory

Compare and Contrast of snsd.wav

In [21]:
```
import pandas as pd

factor = 6
down_aliased = data_music[::factor]

# original
# Calling DataFrame constructor on list
df = pd.DataFrame(data_music)
Hist_title = "Histogram of original snsd.wav"
df.plot(kind='hist', title=Hist_title, bins = 150)
plt.savefig(Hist_title+'.png', facecolor='white', transparent=False) # saving the plot into png file
plt.show()
plt.clf() # to clear the current plot to make new

# aliased
# Calling DataFrame constructor on list
df = pd.DataFrame(down_aliased)
Hist_title = "Histogram of aliased snsd.wav"
df.plot(kind='hist', title=Hist_title, bins = 150)
plt.savefig(Hist_title+'.png', facecolor='white', transparent=False) # saving the plot into png file
plt.show()
plt.clf() # to clear the current plot to make new
```



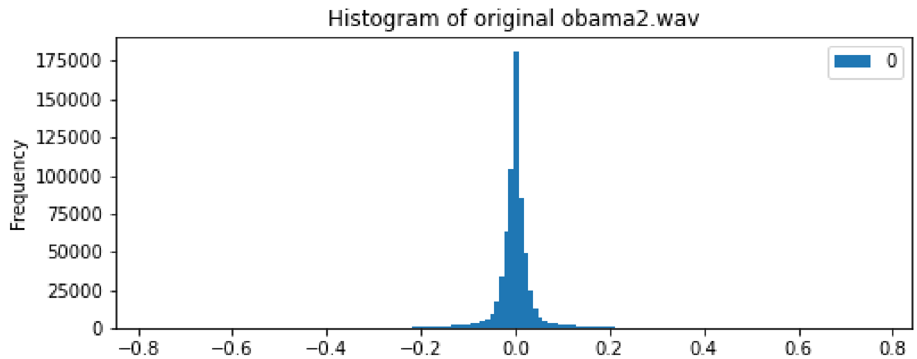

```
<Figure size 576x216 with 0 Axes>
```

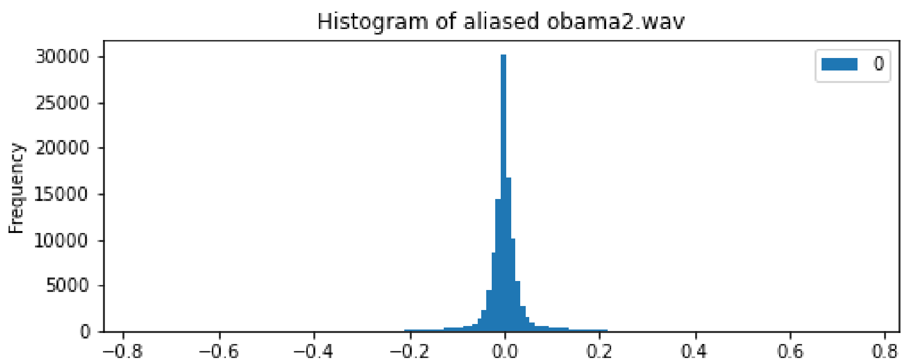


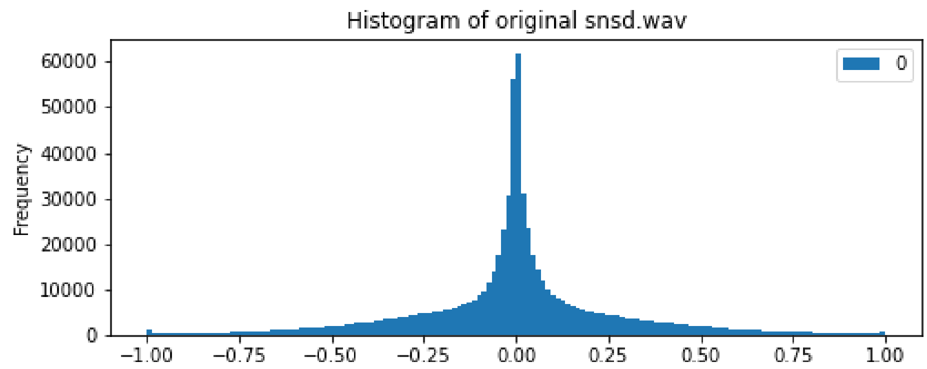

```
<Figure size 576x216 with 0 Axes>
```

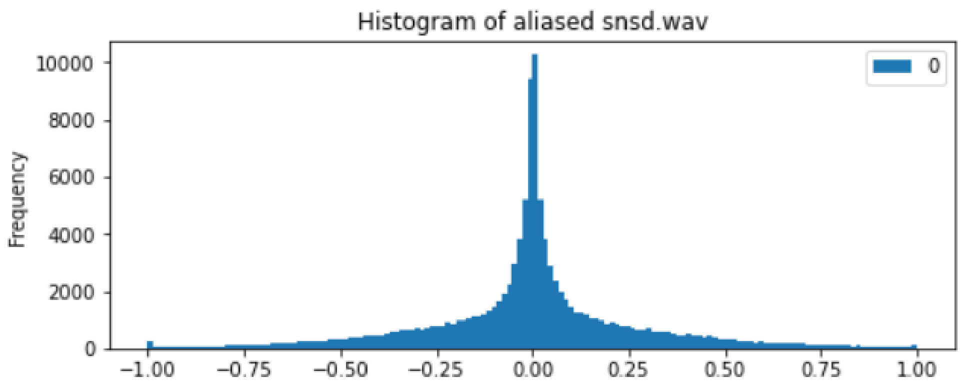
Figure 6.10
This figure shows the comparison of the histogram of the aliased snsd.wav and original, also the saving of the histogram file in the directory

7. Results(sample)









GOOGLE DRIVE LINK:
https://drive.google.com/drive/folders/1oxVcgx1iN2ki4_fLsl3hj5yVLOU79t3f?usp=sharing

8. Data Analysis

As what is presented in the signal waves of both original and aliased obama.wav and snsd.wav, it shows that the noises on the aliased *obama.wav* and *snsd.wav* have been filtered and reduced, it got smoother than the original. As per the histogram of *obama.wav*, the highest frequency of the original is 175,000 Hz, while on the aliased version is 30,000 Hz.. In *the snsd.wav*, he highest frequency of the original is 60,000, while on the aliased version is 10,000.

9. Summary and Conclusions

In this laboratory activity, we were able to deal with the programmatic creation of a noise signal by analyzing and quantifying properties in a pre-existing dataset. Data that is noisy is useless data. It increases the amount of storage space required unnecessarily and can also have a negative impact on the outcomes of any data mining analysis. In this activity, we develop a simple program that manipulates the statistical properties of signals and noise. We are given wav files data and applied an aliasing filter to it. Having the two versions, which are the original and aliased version helped us to conclude that the aliased version or the filtered sound is smoother than the original version since the frequency based on the histogram is lower than the original.

10. Learnings and Contributions of each member

DEVERA - In this activity, we perform a manual degradation of signal by applying aliasing to a sound. Physically, we can achieve an aliased signal when the sampling rate is lower than the Nyquist rate; in this activity, we filter the original signal by applying a low pass filter, adjusting the sample rate and factor. Comparing the original and the aliased signal by listening to the output sound, we can differentiate the aliased signal as it reduces quality and clearness; it also lowers the higher frequencies of the signal making the audio sound low. In performing this group activity, I impart some analysis in comparing the original and aliased signal.

GALIT - In this laboratory activity, we are able to see the difference between the original version and the aliased version of a given sound. Through comparison and contrast, I have learned that aliasing is important in order to filter the sound, in which the high frequencies which are inaudible to the human ear could be removed and converted to a much lower frequency. As a group member in this activity, I contributed to the process of analysis, including the comparison and contrast of two different versions (original and aliased) of the given data. This activity also helped our group to broaden our skills and knowledge in python since it is the programming language that was used to develop a program for filtering sounds.

RAPER - Contributions also did the activity and through this activity I learned that through the use of programing particularly about the ADC, DAC and aliasing through it we can manage to clear and filter the noise in the audio, thus we can clearly understand what audio is all about. To add to that, filtering the noise helps the machine to easier understand, interpret the audio to help the program of optical character recognition (OCR) and reduce the storage required and it significantly affects the results of the data mining analysis And differentiating the audios we can see that there is a difference in terms of original vs the aliased one since the frequency of the original is quite higher than the aliased one.

HERNANDEZ - I learned that in analysing the noise data, you can manipulate it by using or creating a program in python language, you can see the signal waved and histogram to compare the frequencies. In this activity, we have a sample audio that we get the sound waves from and create an aliased audio from the original to get the sound waves of it also and compare the two. Aliasing is a sampling effect in which various signals become indistinguishable. I helped with the data analysis.

VILLAREAL - I learned that using aliasing can reduce some background noises of audio. From the code, I contributed the extracting of 2 data set samples, and then I used the example code from Lab Alias.ipynb file to display the mean, standard deviation, and time series plot of 2 data sets. I also used the histogram to plot the comparison and contrast of 2 data sets and save it as a png file.