```
!rm -rf ./ENEE3309-2-2022
!git clone https://github.com/mkjubran/ENEE3309-2-2022.git
```

```
Cloning into 'ENEE3309-2-2022'...
remote: Enumerating objects: 55, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 55 (delta 3), reused 5 (delta 1), pack-reused 47
Unpacking objects: 100% (55/55), done.
```

```
import wave
from scipy.io import wavfile
import numpy as np
import matplotlib.pyplot as plt
```

```
def upsampler(input_signal, upsamplerate):
   L=input_signal.shape[0]
   return np.reshape(np.transpose(np.ones([upsamplerate,1])*input_signal),[upsamplerate*L]
```

```
def downsampler(input_signal, downsamplerate):
   L=input_signal.shape[0]
   return input_signal[0::downsamplerate]
```

```
def plotTimeFreq(y, Fs, BWrange):
  n = len(y) # length of the signal
  k = np.arange(n)
  T = n/Fs

  t = np.arange(0,n*Ts,Ts) # time vector

  frq = k/T # two sides frequency range
  fcen=frq[int(len(frq)/2)]
  frq_DS=frq-fcen
  frq_SS = frq[range(int(n/2))] # one side frequency range

  Y = np.fft.fft(y) # fft computing and normalization
  yinv= np.fft.ifft(Y).real # ifft computing and normalization
  Y_DS=np.roll(Y,int(n/2))
  Y_SS = Y[range(int(n/2))]

  fcenIndex = (np.abs(frq_DS)).argmin()
  RangeIndex = (np.abs(frq_DS-BWrange)).argmin() - fcenIndex

  RangeIndexMin = fcenIndex-RangeIndex
  if RangeIndexMin < 0:
    RangeIndexMin = 0

  RangeIndexMax = fcenIndex+RangeIndex
  if RangeIndexMax > len(frq_DS)-1:
    RangeIndexMax = len(frq_DS)-1
```

```
  fig, ax = plt.subplots(2, 1,  figsize=(16, 6))
  ax[0].plot(t,y)
  ax[0].set_xlabel('Time')
  ax[0].set_ylabel('Amplitude')
  ax[1].set_xlabel('Freq (Hz)')
  ax[1].set_ylabel('|Y(freq)|')
  ax[1].plot(frq_DS[RangeIndexMin:RangeIndexMax],abs(Y_DS[RangeIndexMin:RangeIndexMax]),'r
  ax[1].set_xlabel('Freq (Hz)')
  ax[1].set_ylabel('|Y(freq)|')

  return yinv


yanA = '/content/ENEE3309-2-2022/SunnyDayFilteredBW3000.wav'
yanB = '/content/ENEE3309-2-2022/Athan1FilteredBW3000.wav'
yanC = '/content/ENEE3309-2-2022/CountingFilteredBW3000.wav'
yanD = '/content/ENEE3309-2-2022/RainFilteredBW3000.wav'
yanE = '/content/ENEE3309-2-2022/SummerFilteredBW3000.wav'
mixaudio = '/content/ENEE3309-2-2022/FDMAMixedAudio.wav'
filenameWavefiltered='./Filtered.wav'
filenameWavewithoutfilter='./Withoutfilter.wav'
BWrange=10000
fc1 = 50000
fc2 = 60000
fc3 = 70000
fc4 = 80000
fc5 = 90000
#-----------------------------------
fcmax=90000
BW=30000
BWrange=100000

########-------------------
upsamplerate = int(fcmax/BW)
downsamplerate = int(fcmax/BW)

rate1, data1 = wavfile.read(yanA)
rate2, data2 = wavfile.read(yanB)
rate3, data3 = wavfile.read(yanC)
rate4, data4 = wavfile.read(yanD)
rate5, data5 = wavfile.read(yanE)
ratemin=np.min([rate1,rate2,rate3,rate4,rate5])

data1 = downsampler(data1, int(rate1/ratemin))
data2 = downsampler(data2, int(rate2/ratemin))
data3 = downsampler(data3, int(rate3/ratemin))
data4 = downsampler(data4, int(rate4/ratemin))
data5 = downsampler(data5, int(rate5/ratemin))
Lmin = np.min([len(data1), len(data2), len(data3), len(data4), len(data5)])

data1 = data1[0:Lmin];data2 = data2[0:Lmin];data3 = data3[0:Lmin];data4 = data4[0:Lmin];da

data1 = upsampler(data1, upsamplerate)
```

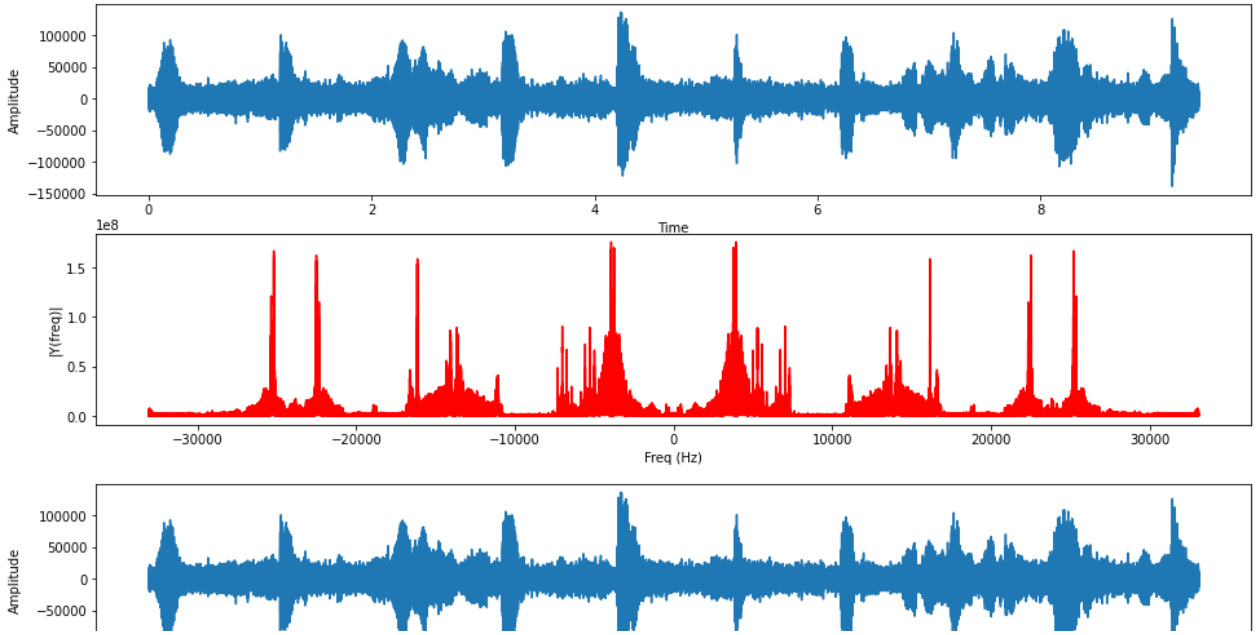
```
data2 = 0.5*upsampler(data2, upsamplerate)
data3 = 5*upsampler(data3, upsamplerate)
data4 = 10*upsampler(data4, upsamplerate)
data5 = 1.5*upsampler(data5, upsamplerate)

Fs=ratemin*upsamplerate;
Ts = 1.0/Fs; # sampling interval
t = np.arange(0,len(data1)*Ts,Ts) # time vector

## module Signal #1
y=[float(x) for x in data1]
carrier_signal = np.cos(2*np.pi*fc1*t)
output_signal = y*carrier_signal
output_signal_1 = output_signal

## module Signal #2
y=[float(x) for x in data2]
carrier_signal = np.cos(2*np.pi*fc2*t)
output_signal = y*carrier_signal
output_signal_2 = output_signal

## module Signal #2
y=[float(x) for x in data3]
carrier_signal = np.cos(2*np.pi*fc3*t)
output_signal = y*carrier_signal
output_signal_3 = output_signal

## module Signal #2
y=[float(x) for x in data4]
carrier_signal = np.cos(2*np.pi*fc4*t)
output_signal = y*carrier_signal
output_signal_4 = output_signal

## module Signal #2
y=[float(x) for x in data5]
carrier_signal = np.cos(2*np.pi*fc5*t)
output_signal = y*carrier_signal
output_signal_5 = output_signal
##---------------
##-----------------------------------------------
## Mixxing the modulated signals
yama = output_signal_1 + output_signal_2 + output_signal_3 + output_signal_4 + output_sign

### Plot in the time and frequency domain
yinv = plotTimeFreq(yama, Fs, BWrange)
print("MIX AUDIO")

### Downsample
#yinv = downsampler(yinv, downsamplerate)

yinv_int16=yinv.astype(np.int16)
wavfile.write(mixaudio, ratemin, yinv_int16)
rate = ratemin
##-----------------------------------------------
```

```
filtereddata = np.fft.rfft(yama, axis=0)

filteredwrite = np.fft.irfft(filtereddata, axis=0)
################################################
## Generate Signal and add save it to text file
Fs=rate;
Ts = 1.0/Fs; # sampling interval
t = np.arange(0,len(yama)*Ts,Ts) # time vector

y=[float(x) for x in yama]
## Write values to a file
#Open new data file
if write!=0:
   f = open("Signal_in_text.txt", "w")
   for i in range(len(y)):
       f.write( str(y[i]) + " " + str(float(t[i])) + "\n"  )
   f.close()


## Read values from a file
if read !=0:
   with open('Signal_in_text.txt') as f:
      w=f.read()
   y=[];
   t=[];
   for x in w.split('\n'):
      if x != '':
         y.append(float(x.split()[0]))
         t.append(float(x.split()[1]))


n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
fcen=frq[int(len(frq)/2)]
frq_DS=frq-fcen
frq_SS = frq[range(int(n/2))] # one side frequency range

Y = np.fft.fft(y) # fft computing and normalization
yinv= np.fft.ifft(Y).real # ifft computing and normalization
Y_DS=np.roll(Y,int(n/2))
Y_SS = Y[range(int(n/2))]

fcenIndex = (np.abs(frq_DS)).argmin()
RangeIndex = (np.abs(frq_DS-BWrange)).argmin() - fcenIndex

RangeIndexMin = fcenIndex-RangeIndex
if RangeIndexMin < 0:
  RangeIndexMin = 0

RangeIndexMax = fcenIndex+RangeIndex
if RangeIndexMax > len(frq_DS)-1:
  RangeIndexMax = len(frq_DS)-1
```

```
fig, ax = plt.subplots(2, 1,  figsize=(16, 6))
ax[0].plot(t,y)
ax[0].set_xlabel('Time')
ax[0].set_ylabel('Amplitude')
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')
ax[1].plot(frq_DS[RangeIndexMin:RangeIndexMax],abs(Y_DS[RangeIndexMin:RangeIndexMax]),'r')
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')

plt.show()

y=np.array(y)
y_int=y.astype(np.int16)

yinv=np.array(yinv)
yinv_int=yinv.astype(np.int16)
wavfile.write(filenameWavewithoutfilter, rate, y_int)

B = int(input ("Enter BW :"))

fBWIndex = (np.abs(frq_DS - B)).argmin()
B = frq_DS[fBWIndex]

Mask_DS=np.ones(len(frq_DS))
Yf_DS=np.copy(Y_DS)
Bmax=frq_DS[len(frq_DS)-1]
Bmin=0
Bold=0

Yf_DS=np.copy(Y_DS)
for cnt in range(len(frq_DS)):
  if ~(((frq_DS[cnt])>-1*B) and ((frq_DS[cnt])<B)):
    Mask_DS[cnt]=0;
    #print(B,frq_DS[cnt],Yf_DS[cnt])
    Yf_DS[cnt]=Y_DS[cnt]*0;


Yf=np.roll(Yf_DS,int(n/2))
yinv= np.fft.ifft(Yf).real # ifft computing and normalization
yinv=np.array(yinv)
yinv_int=yinv.astype(np.int16)

wavfile.write(filenameWavefiltered, rate, yinv_int)

fig, ax = plt.subplots(3, 1,  figsize=(16, 9))
ax[0].plot(frq_DS[RangeIndexMin:RangeIndexMax],abs(Mask_DS[RangeIndexMin:RangeIndexMax]),'
ax[0].set_xlabel('Freq (Hz)')
ax[0].set_ylabel('|H(freq)|')
ax[1].plot(frq_DS[RangeIndexMin:RangeIndexMax],abs(Yf_DS[RangeIndexMin:RangeIndexMax]),'r'
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')
ax[2].plot(t,yinv,'g') # plotting the spectrum
ax[2].set_xlabel('Time')
ax[2].set_ylabel('Amplitude')
```

```
    plt.show()
```

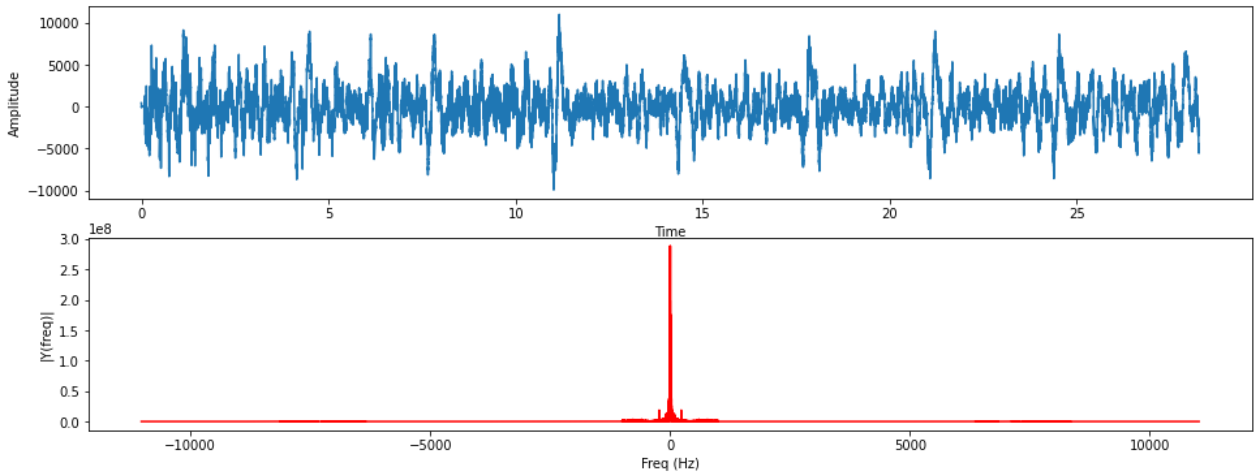
MIX AUDIO



AUDIO MIX SIGNAL

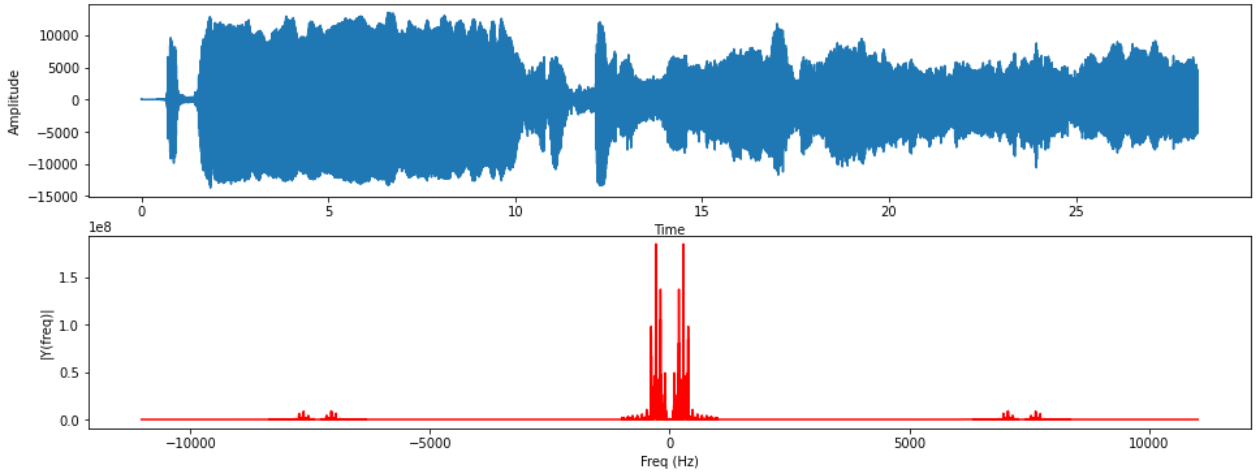
```
yinA = plotTimeFreq(data1, Fs, BWrange)
print("DATA 1")
```

DATA 1



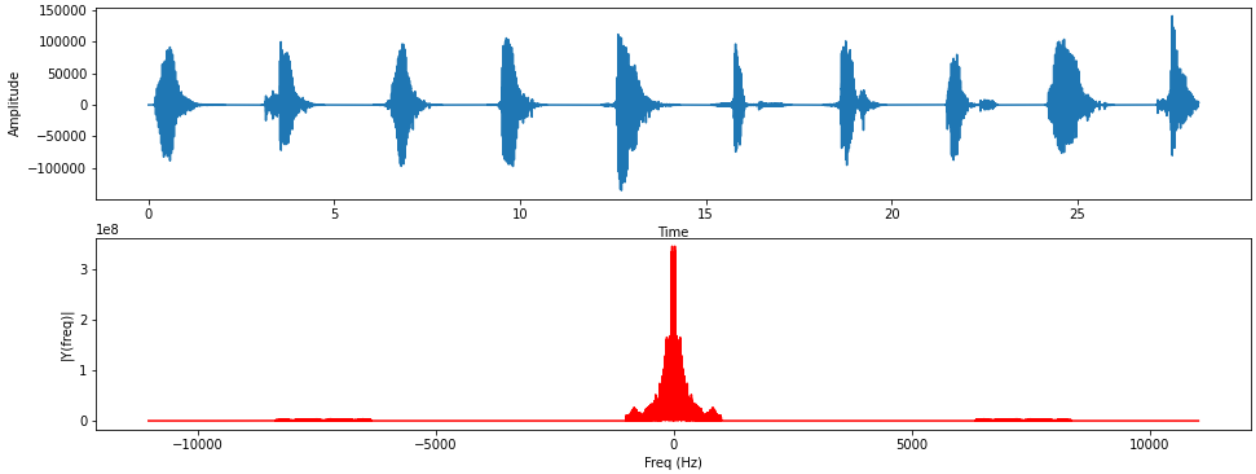

```
yinB = plotTimeFreq(data2, Fs, BWrange)
print("DATA 2")
```

DATA 2



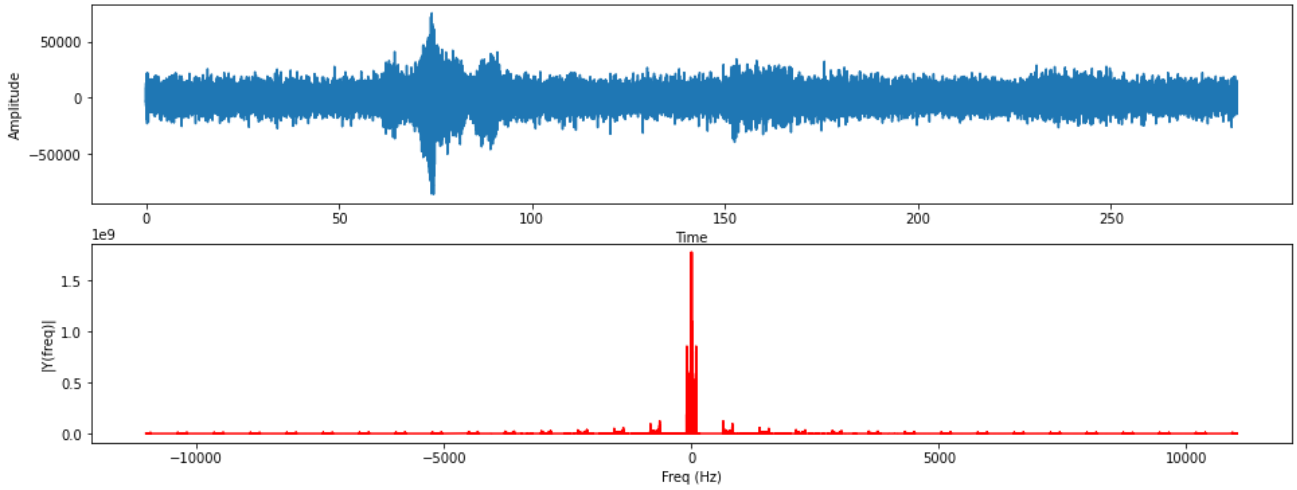

```
yinC = plotTimeFreq(data3, Fs, BWrange)
print("DATA 3")
```

DATA 3



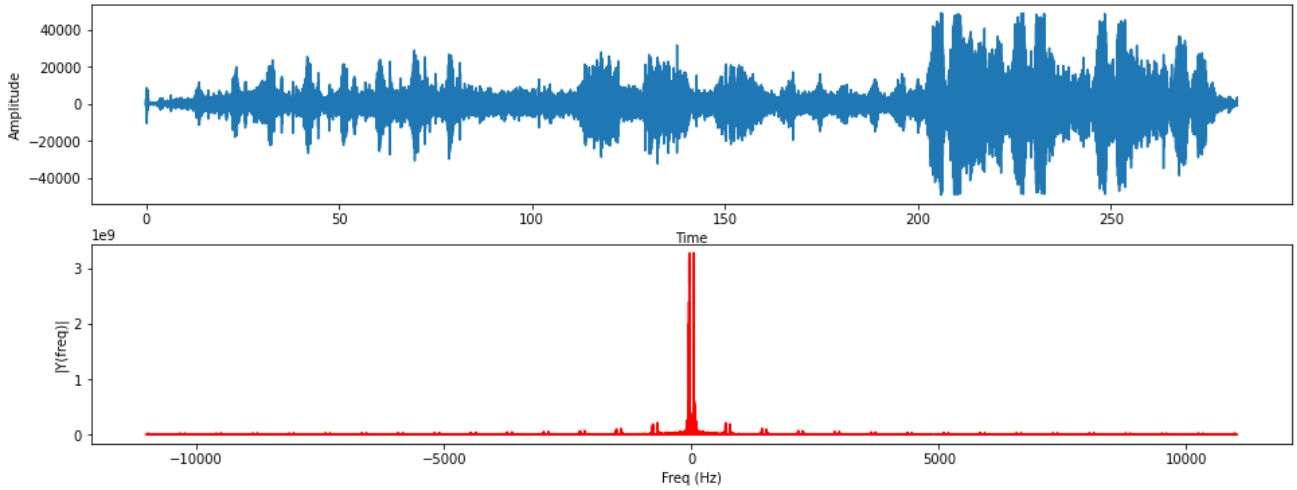

```
yinD = plotTimeFreq(data4, Fs, BWrange)
print("DATA 4")
```

DATA 4




```
yinE = plotTimeFreq(data5, Fs, BWrange)
print("DATA 5")
```

DATA 5



AUDIO 1

```
from IPython.display import Audio
# Generate a player for mono sound
Audio(yanA,rate=rate)
```

0:00 / 0:20

AUDIO 2

```
from IPython.display import Audio
# Generate a player for mono sound
Audio(yanB,rate=rate)
```

0:00 / 3:02

AUDIO 3

```
from IPython.display import Audio
# Generate a player for mono sound
Audio(yanC,rate=rate)
```

0:00 / 0:20

AUDIO 4

```
from IPython.display import Audio
# Generate a player for mono sound
Audio(yanD,rate=rate)
```

0:00 / 0:31

AUDIO 5

```
from IPython.display import Audio
# Generate a player for mono sound
Audio(yanD,rate=rate)
```

0:00 / 0:31

MIX AUDIO

```
from IPython.display import Audio
# Generate a player for mono sound
Audio(mixaudio,rate=rate)
```

0:00 / 4:42

✓ 1s completed at 11:41 PM