

Tugas Hands On 1

NAMA : MARSELLA YESI NATALIA SINAGA

NIM : 121140174

MATA KULIAH : SISTEM / TEKNOLOGI MULTIMEDIA

1. Buktiakan bahwa terdapat perbedaan yang signifikan pada visualisasi audio menggunakan 5 buah sumber audio yang berbeda:
 - Pertama, bersuara pelan. Kedua, relatif bersuara dengan volume sedang. Ketiga, relatif bersuara dengan volume tinggi. Keempat, bersuara volume sedang, namun mengucapkan kata-kata dengan huruf `s` yang panjang (`essssss`). Kelima, bersuara volume sedang, mengucapkan kata `brrbrbrbrb` / lip trill.
 - Masing-masing silahkan membuat file wav/mp3/flac yang sesuai dengan ketentuan tersebut (5 file).
 - Gunakan waveforms dan spectrogram untuk memvisualisasikan ketiga sumber audio tersebut.
 - Berikan analisa anda terhadap perbedaan visualisasi audio tersebut.

```
In [91]: import matplotlib.pyplot as plt
import numpy as np
import os
import wave
import librosa
import librosa.display
```

```
In [92]: volumerendah = os.path.join(os.getcwd(), 'data', 'spoken.wav')
```

```
In [93]: # Periksa jika file tidak ditemukan
if not os.path.exists(volumerendah):
    print(f"File {volumerendah} not found.")
    exit()
```

```
In [94]: # Membaca file WAV
with wave.open(volumerendah, 'r') as wav_file:
    frames = wav_file.readframes(-1)
    panjangisisample = np.frombuffer(frames, dtype='int16')
    frsample = wav_file.getframerate()

    # Mendapatkan informasi channel
    volumerendahchannel = wav_file.getnchannels()
```

```
In [95]: # Tampilkan informasi file WAV
print(f"Isi dari sound_data: {panjangisisample}")
```

```
print(f"Panjang dari sound_data: {len(panjangisisample)}")
print(f"Sampling rate: {frsample}")
print(f"Jumlah channel: {volumerendahchannel}")
```

```
Isi dari sound_data: [0 0 0 ... 0 0 0]
Panjang dari sound_data: 347904
Sampling rate: 44100
Jumlah channel: 2
```

```
In [96]: # Membagi channel menjadi dua bagian (kiri dan kanan)
panjangisisample = panjangisisample.reshape(-1, volumerendahchannel)
print(f"Panjang isi sample setelah direshape: {len(panjangisisample)}")

audiokiri_volrendah = panjangisisample[:, 0]
audiokanan_volrendah = panjangisisample[:, 1]

print(f"Panjang audio kiri: {len(audiokiri_volrendah)}")
print(f"Panjang audio kanan: {len(audiokanan_volrendah)}")
```

```
Panjang isi sample setelah direshape: 173952
Panjang audio kiri: 173952
Panjang audio kanan: 173952
```

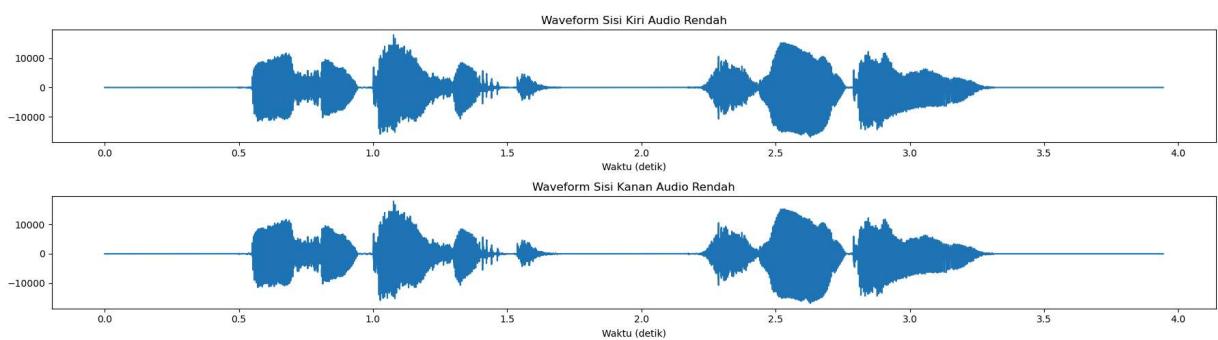
```
In [97]: # Melakukan plotting audio kiri
panjangsampelkiri = len(audiokiri_volrendah)
print(f"Panjang sampel per sisi: {panjangsampelkiri}")

durasi_spoken = panjangsampelkiri / frsample
print(f"Panjang spoken.wav: {durasi_spoken}")

# Membuat sumbu waktu
time_axis = np.linspace(0, durasi_spoken, panjangsampelkiri)
```

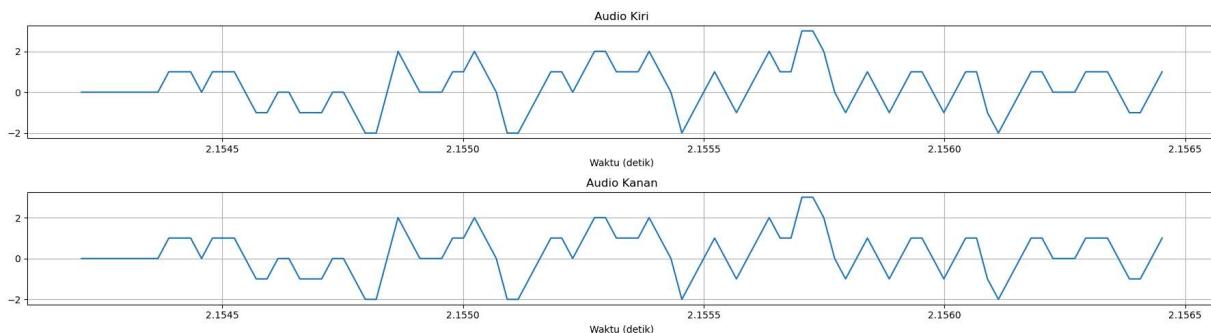
```
Panjang sampel per sisi: 173952
Panjang spoken.wav: 3.944489795918367
```

```
In [98]: # Plotting audio kiri dan kanan
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis, audiokiri_volrendah, label='Kiri')
fig[0].set_title('Waveform Sisi Kiri Audio Rendah')
fig[0].set_xlabel('Waktu (detik)')
fig[1].plot(time_axis, audiokanan_volrendah, label='Kanan')
fig[1].set_title('Waveform Sisi Kanan Audio Rendah')
fig[1].set_xlabel('Waktu (detik)')
plt.tight_layout()
plt.show()
```



Dalam waveform audio stereo menunjukkan dua kanal, yaitu kiri dan kanan, yang merekam suara dengan durasi sekitar 4 detik. Aktivitas suara utama terjadi antara detik 0,5 dan 1,5, di mana amplitudo mencapai puncaknya. Setelah itu, suara melemah dan relatif tenang di akhir rekaman. Kedua kanal memiliki bentuk gelombang yang serupa, menunjukkan bahwa rekaman ini bersifat stereo dengan sumber suara kemungkinan di tengah atau direkam dengan mikrofon mono. Variasi amplitudo yang signifikan terlihat di sepanjang rekaman, menunjukkan perbedaan volume di bagian-bagian tertentu.

```
In [99]: # Plotting audio kiri dan kanan dengan zoom di rentang durasi 2 detik
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis[95000:95100], audiokiri_volrendah[95000:95100], label='Kiri')
fig[0].set_title('Audio Kiri')
fig[0].set_xlabel('Waktu (detik)')
fig[0].grid()
fig[1].plot(time_axis[95000:95100], audiokanan_volrendah[95000:95100], label='Kanan')
fig[1].set_title('Audio Kanan')
fig[1].set_xlabel('Waktu (detik)')
fig[1].grid()
plt.tight_layout()
plt.show()
```



```
In [100...]: print(f"Isi audio kiri: {audiokiri_volrendah[95000:95100]}")
print(f"Isi audio kanan: {audiokanan_volrendah[95000:95100]}")
```

```
Isi audio kiri: [ 0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  1  1  1  0 -1 -1  0  0 -1 -1 -1  0
 0 -1 -2 -2  0  2  1  0  0  0  1  1  2  1  0 -2 -2 -1  0  1  1  0  1  1  2  1  1  3  3  2  0 -1  0
 2  1  1  1  2  1  0 -2 -1  0  1  0 -1  0  1  2  1  1  3  3  2  0 -1  0
 1  0 -1  0  1  1  0 -1  0  1  1 -1 -2 -1  0  1  1  0  0  0  1  1  1  0
 -1 -1  0  1]
Isi audio kanan: [ 0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  1  1  1  0 -1 -1  0  0 -1 -1 -1  0
 -1  0
 0 -1 -2 -2  0  2  1  0  0  0  1  1  2  1  0 -2 -2 -1  0  1  1  0  1  1  2
 2  1  1  1  2  1  0 -2 -1  0  1  0 -1  0  1  2  1  1  3  3  2  0 -1  0
 1  0 -1  0  1  1  0 -1  0  1  1 -1 -2 -1  0  1  1  0  0  0  1  1  1  0
 -1 -1  0  1]
```

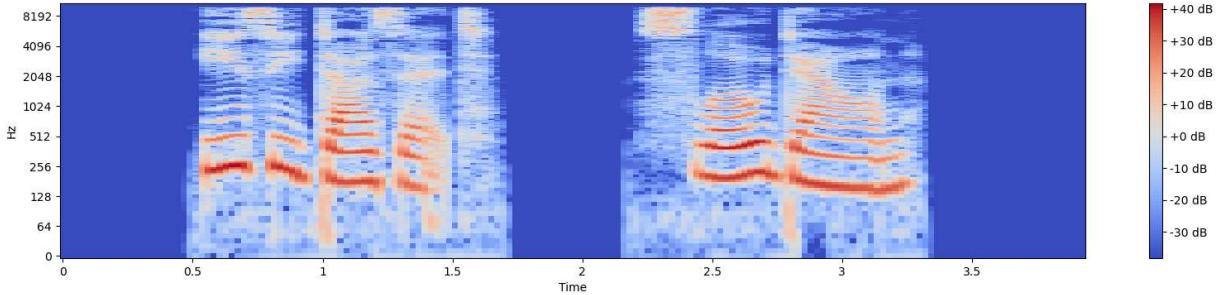
```
In [101...]: # Memuat audio menggunakan Librosa
data_audio, sr = librosa.load(volumerendah)
S = librosa.stft(data_audio)
S_db = librosa.amplitude_to_db(np.abs(S))

print(f"Sampling Rate: {sr}")
```

Sampling Rate: 22050

In [102...]

```
# Plotting spektrogram
plt.figure(figsize=(20, 4))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='coolwarm')
plt.colorbar(format='%+2.0f dB')
plt.show()
```



audio tersebut memiliki dominasi frekuensi rendah, terutama pada rentang frekuensi 100-200 Hz, yang menandakan adanya bass yang kuat. Terlihat juga adanya puncak frekuensi pada rentang 500-600 Hz, yang menunjukkan suara vokal atau instrumental yang kuat pada frekuensi tersebut. Namun, spektrogram ini juga menunjukkan bahwa audio tersebut memiliki kurangnya frekuensi tinggi, terutama pada rentang 8000-10000 Hz, yang menandakan bahwa treble dalam audio ini kurang kuat. Selain itu, terdapat pola berulang pada spektrogram, terutama di rentang frekuensi 100-200 Hz dan 500-600 Hz, yang mengindikasikan adanya ritme atau pola suara yang konsisten dalam audio. Berdasarkan karakteristik visual tersebut, dapat disimpulkan bahwa audio ini memiliki bass yang kuat, vokal atau instrumen yang jelas, tetapi dengan treble yang kurang kuat, serta menunjukkan pola suara yang konsisten.

In [103...]

```
volumesedang = os.path.join(os.getcwd(), 'data', 'wind.wav')
```

In [104...]

```
# Periksa jika file tidak ditemukan
if not os.path.exists(volumesedang):
    print(f"File {volumesedang} not found.")
    exit()
```

In [105...]

```
# Membaca file WAV
with wave.open(volumesedang, 'r') as wav_file:
    framessedang = wav_file.readframes(-1)
    panjangsamplesedang = np.frombuffer(framessedang, dtype='int16')
    frsamplesedang = wav_file.getframerate()

    # Mendapatkan informasi channel
    volumesedangchannel = wav_file.getnchannels()
```

In [106...]

```
# Tampilkan informasi file WAV
print(f"Isi dari sound_data: {panjangsamplesedang}")
print(f"Panjang dari sound_data: {len(panjangsamplesedang)}")
print(f"Sampling rate: {frsamplesedang}")
print(f"Jumlah channel: {volumesedangchannel}")
```

```
Isi dari sound_data: [0 0 0 ... 0 0 0]
```

```
Panjang dari sound_data: 965376
```

```
Sampling rate: 44100
```

```
Jumlah channel: 2
```

```
In [107...]
```

```
# Membagi channel menjadi dua bagian (kiri dan kanan)
panjangsamplesedang = panjangsamplesedang.reshape(-1, volumesedangchannel)
print(f"Panjang isi sample setelah direshape: {len(panjangsamplesedang)}")

audiokiri_volsedang = panjangsamplesedang[:, 0]
audiokanan_volsedang = panjangsamplesedang[:, 1]
print(f"Panjang audio kiri: {len(audiokiri_volsedang)}")
print(f"Panjang audio kanan: {len(audiokanan_volsedang)}")
```

```
Panjang isi sample setelah direshape: 482688
```

```
Panjang audio kiri: 482688
```

```
Panjang audio kanan: 482688
```

```
In [108...]
```

```
# Melakukan plotting audio kiri
panjangsampelsedang = len(audiokiri_volsedang)
print(f"Panjang sampel per sisi: {panjangsampelsedang}")

durasi_wind = panjangsampelsedang / frsamplesedang
print(f"Panjang wind.wav: {durasi_wind}")

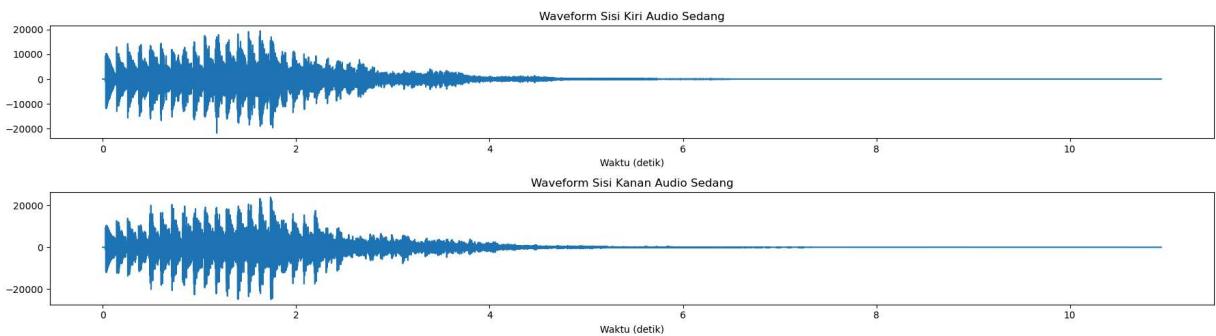
# Membuat sumbu waktu
time_axis = np.linspace(0, durasi_wind, panjangsampelsedang)
```

```
Panjang sampel per sisi: 482688
```

```
Panjang wind.wav: 10.94530612244898
```

```
In [109...]
```

```
# Plotting audio kiri dan kanan
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis, audiokiri_volsedang, label='Kiri')
fig[0].set_title('Waveform Sisi Kiri Audio Sedang')
fig[0].set_xlabel('Waktu (detik)')
fig[1].plot(time_axis, audiokanan_volsedang, label='Kanan')
fig[1].set_title('Waveform Sisi Kanan Audio Sedang')
fig[1].set_xlabel('Waktu (detik)')
plt.tight_layout()
plt.show()
```



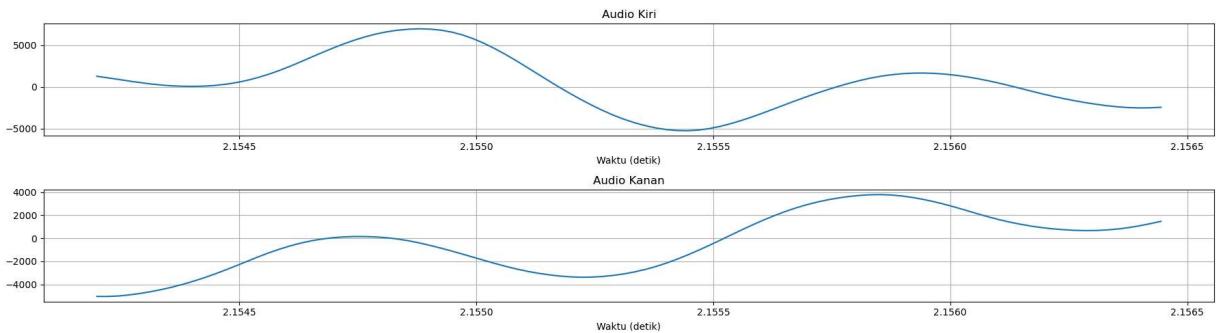
Dalam waveform audio menampilkan dua kanal stereo, yaitu sisi kiri dan kanan, dengan durasi keseluruhan sekitar 10 detik. Suara utama terdengar dari detik 0 hingga sekitar detik 4,5, sedangkan setelah detik 4,5 hingga akhir grafik terlihat garis lurus yang menunjukkan

tidak adanya suara (silence). Pada fase awal, antara detik 0 hingga 1,5, amplitudo gelombang memperlihatkan puncak-puncak tajam dan fluktuasi yang konstan, menandakan adanya suara dengan volume tinggi atau intensitas kuat. Secara bertahap, amplitudo suara menurun hingga mendekati nol pada sekitar detik 4,5, yang menandakan bahwa energi suara perlahan memudar dan akhirnya menghilang.

Kedua kanal, baik kiri maupun kanan, memiliki pola gelombang yang hampir serupa, menunjukkan bahwa rekaman audio memiliki keseimbangan yang baik di kedua sisi. Tidak ada perbedaan signifikan antara kanal kiri dan kanan, yang mengindikasikan perekaman stereo yang seimbang atau audio mono yang dipancarkan ke kedua kanal. dan terlihat bahwa suara dimulai dengan intensitas yang tinggi dan secara bertahap berkurang sampai akhirnya berhenti. Bagian akhir dari audio yang tenang, mulai dari detik 4,5 hingga 10, menunjukkan kemungkinan jeda atau berakhirnya rekaman.

In [110...]

```
# Plotting audio kiri dan kanan dengan zoom di rentang durasi 2 detik
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis[95000:95100], audiokiri_volsedang[95000:95100], label='Kiri')
fig[0].set_title('Audio Kiri')
fig[0].set_xlabel('Waktu (detik)')
fig[0].grid()
fig[1].plot(time_axis[95000:95100], audiokanan_volsedang[95000:95100], label='Kanan')
fig[1].set_title('Audio Kanan')
fig[1].set_xlabel('Waktu (detik)')
fig[1].grid()
plt.tight_layout()
plt.show()
```



In [111...]

```
print(f"Isi audio kiri: {audiokiri_volsedang[95000:95100]}")
print(f"Isi audio kanan: {audiokanan_volsedang[95000:95100]}")
```

```

Isi audio kiri: [ 1276 1093 902 708 524 359 225 131 78 67 99
182
      323 525 794 1130 1528 1982 2489 3034 3595 4147 4678 5176
      5629 6025 6357 6618 6806 6924 6970 6936 6810 6583 6244 5793
      5242 4609 3911 3158 2369 1564 763 -19 -772 -1488 -2162 -2789
     -3360 -3861 -4286 -4638 -4918 -5119 -5233 -5251 -5171 -4994 -4730 -4396
     -4009 -3580 -3116 -2631 -2142 -1659 -1192 -746 -324 74 443 775
      1063 1296 1473 1592 1655 1663 1617 1523 1383 1201 978 720
      433 124 -198 -525 -844 -1142 -1415 -1663 -1888 -2087 -2258 -2391
     -2479 -2517 -2503 -2438]
Isi audio kanan: [-5050 -5053 -5008 -4918 -4793 -4637 -4453 -4240 -3998 -3725 -3422
-3093
     -2742 -2373 -1994 -1620 -1265 -944 -660 -416 -215 -61 49 117
      147 144 110 40 -70 -221 -406 -621 -857 -1108 -1372 -1642
     -1913 -2178 -2431 -2660 -2860 -3027 -3165 -3274 -3349 -3384 -3374 -3321
     -3229 -3098 -2926 -2712 -2455 -2154 -1811 -1434 -1031 -607 -168 282
      727 1158 1567 1950 2303 2620 2896 3134 3333 3497 3625 3718
     3773 3784 3745 3655 3521 3349 3145 2912 2655 2382 2105 1840
     1599 1386 1200 1041 911 811 736 688 666 676 721 803
      923 1080 1264 1468]

```

In [112...]

```

# Memuat audio menggunakan Librosa
data_audio, sr = librosa.load(volumesedang)
S = librosa.stft(data_audio)
S_db = librosa.amplitude_to_db(np.abs(S))

print(f"Sampling Rate: {sr}")

```

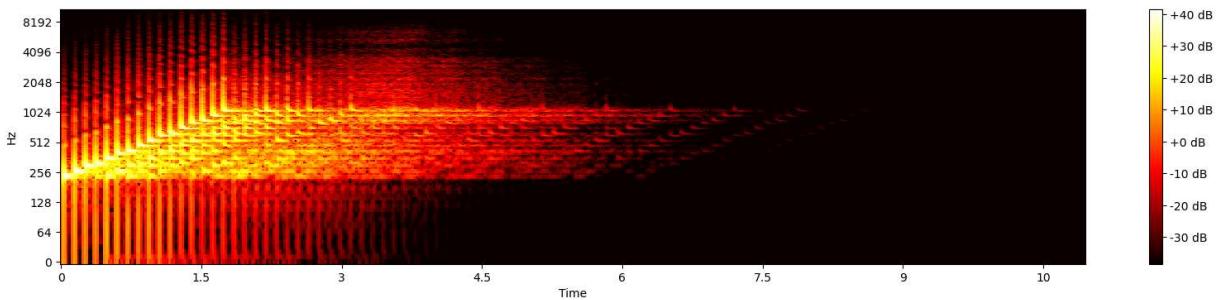
Sampling Rate: 22050

In [113...]

```

# Plotting spektrogram
plt.figure(figsize=(20, 4))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='hot')
plt.colorbar(format='%.2f dB')
plt.show()

```



audio ini memberikan representasi visual dari frekuensi audio yang muncul dari waktu ke waktu. Sumbu horizontal menunjukkan waktu dalam detik, sedangkan sumbu vertikal menunjukkan frekuensi dalam Hertz (Hz). Warna pada spektrogram mewakili amplitudo atau intensitas frekuensi audio, dengan warna yang lebih cerah menunjukkan amplitudo tinggi dan warna yang lebih gelap menunjukkan amplitudo rendah. Analisis menunjukkan bahwa sebagian besar energi audio terkonsentrasi pada frekuensi rendah hingga menengah, kira-kira dari 0 Hz hingga 2048 Hz. Terdapat beberapa pita frekuensi dominan, seperti garis horizontal yang jelas di sekitar 128 Hz, menunjukkan suara bernada rendah yang konstan,

dan beberapa pita frekuensi di sekitar 512 Hz yang berfluktuasi, menandakan instrumen musik atau suara vokal. Selain itu, terdapat percikan energi audio pada frekuensi tinggi di sekitar 4096 Hz yang muncul secara sporadis, mungkin menunjukkan suara perkusi atau efek suara transient. Audio ini memiliki rentang dinamis yang luas, terlihat dari variasi warna yang cukup besar, menunjukkan adanya bagian yang keras dan lembut. Pola-pola temporal yang menarik juga terlihat, dengan lonjakan energi pada frekuensi tinggi yang menandakan event musik atau suara yang tiba-tiba, serta pola berulang pada frekuensi menengah yang menunjukkan melodi atau ritme musik

```
In [114...]: volumetinggi = os.path.join(os.getcwd(), 'data', 'highnote.wav')
```

```
In [115...]: # Periksa jika file tidak ditemukan
if not os.path.exists(volumetinggi):
    print(f"File {volumetinggi} not found.")
    exit()
```

```
In [116...]: # Membaca file WAV
with wave.open(volumetinggi, 'r') as wav_file:
    frames = wav_file.readframes(-1)
    panjangsamples = np.frombuffer(frames, dtype='int16')
    frsample = wav_file.getframerate()

    # Mendapatkan informasi channel
    volumetinggichannel = wav_file.getnchannels()
```

```
In [117...]: # Tampilkan informasi file WAV
print(f"Isi dari sound_data: {panjangsamples}")
print(f"Panjang dari sound_data: {len(panjangsamples)}")
print(f"Sampling rate: {frsample}")
print(f"Jumlah channel: {volumetinggichannel}")
```

```
Isi dari sound_data: [0 0 0 ... 0 0 0]
Panjang dari sound_data: 1131264
Sampling rate: 48000
Jumlah channel: 2
```

```
In [118...]: # Membagi channel menjadi dua bagian (kiri dan kanan)
panjangsamples = panjangsamples.reshape(-1, volumetinggichannel)
print(f"Panjang isi sample setelah direshape: {len(panjangsamples)}")

audiokiri_volttinggi = panjangsamples[:, 0]
audiokanan_volttinggi = panjangsamples[:, 1]
print(f"Panjang audio kiri: {len(audiokiri_volttinggi)}")
```

```
Panjang isi sample setelah direshape: 565632
Panjang audio kiri: 565632
```

```
In [119...]: # Melakukan plotting audio kiri
panjangsampel = len(audiokiri_volttinggi)
print(f"Panjang sampel per sisi: {panjangsampel}")

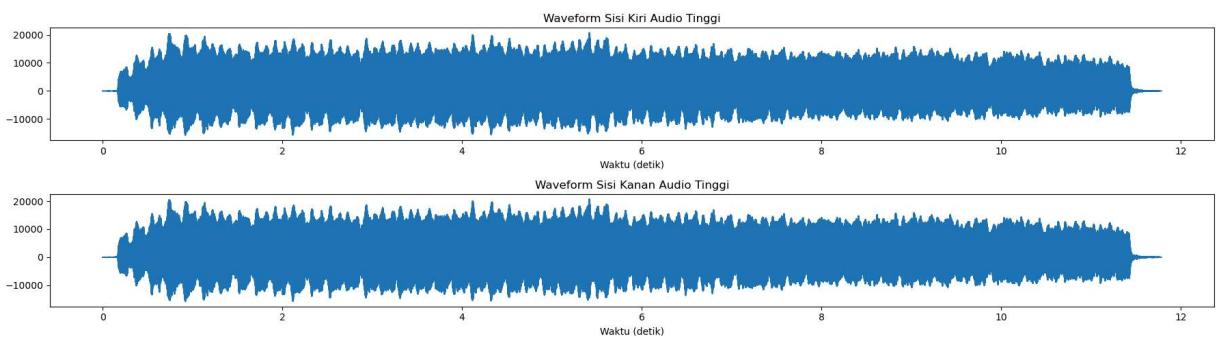
durasi_highnote = panjangsampel / frsample
print(f"Panjang highnote.wav: {durasi_highnote}")
```

```
# Membuat sumbu waktu
time_axis = np.linspace(0, durasi_highnote, panjangsampel)
```

```
Panjang sampel per sisi: 565632
Panjang highnote.wav: 11.784
```

In [120...]

```
# Plotting audio kiri dan kanan
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis, audiokiri_voltinggi, label='Kiri')
fig[0].set_title('Waveform Sisi Kiri Audio Tinggi')
fig[0].set_xlabel('Waktu (detik)')
fig[1].plot(time_axis, audiokanan_voltinggi, label='Kanan')
fig[1].set_title('Waveform Sisi Kanan Audio Tinggi')
fig[1].set_xlabel('Waktu (detik)')
plt.tight_layout()
plt.show()
```

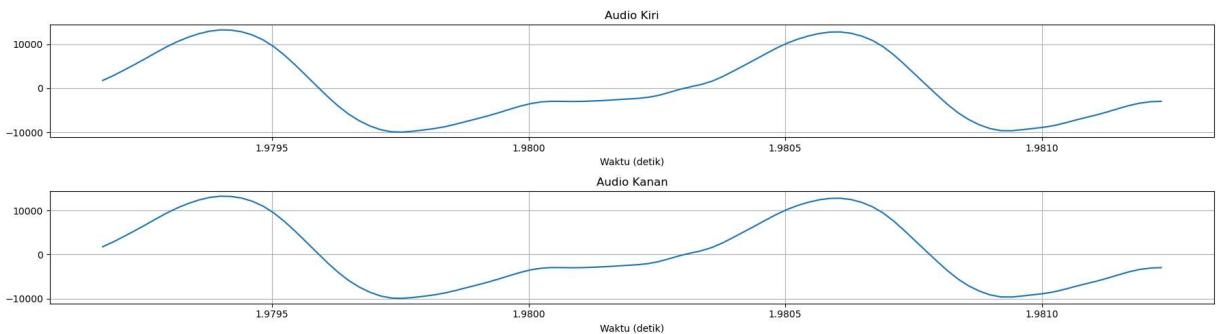


Dalam waveform audio menunjukkan dua kanal stereo, yaitu sisi kiri dan sisi kanan, dengan durasi total sekitar 5 detik, aktivitas suara mulai muncul secara signifikan pada sekitar detik 1, ditandai dengan peningkatan amplitudo yang tajam. Amplitudo yang lebih besar mengindikasikan suara dengan volume yang lebih tinggi, yang tampak berlangsung hingga akhir visualisasi pada detik ke-5.

Pada awal rekaman, dari detik 0 hingga mendekati detik 1, kedua kanal menunjukkan hampir tidak ada aktivitas suara dengan garis yang mendekati nol. Hal ini menandakan bahwa tidak ada atau hanya sedikit suara yang terrekam pada bagian awal tersebut. Setelah itu, amplitudo mulai meningkat secara signifikan, menandakan adanya suara dengan intensitas yang semakin meningkat. Puncak amplitudo terbesar terjadi setelah detik pertama dan berlanjut dengan fluktuasi yang stabil hingga akhir rekaman, yang menunjukkan adanya suara yang konstan dan cukup kuat selama rentang waktu tersebut.

Baik sisi kiri maupun kanan memperlihatkan pola gelombang yang hampir identik, yang mengindikasikan bahwa rekaman memiliki keseimbangan antara kedua kanal audio. Tidak ada perbedaan signifikan antara kanal kiri dan kanan, yang dapat mengindikasikan bahwa audio direkam dalam format stereo simetris atau audio mono yang disebarluaskan ke kedua kanal secara merata. Secara keseluruhan, rekaman ini menunjukkan adanya aktivitas suara yang stabil dan intensitas tinggi dari detik 1 hingga detik ke-5, tanpa ada jeda suara atau penurunan yang signifikan hingga akhir rekaman.

```
In [121... # Plotting audio kiri dan kanan dengan zoom di rentang durasi 2 detik
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis[95000:95100], audiokiri_voltinggi[95000:95100], label='Kiri')
fig[0].set_title('Audio Kiri')
fig[0].set_xlabel('Waktu (detik)')
fig[0].grid()
fig[1].plot(time_axis[95000:95100], audiokanan_voltinggi[95000:95100], label='Kanan')
fig[1].set_title('Audio Kanan')
fig[1].set_xlabel('Waktu (detik)')
fig[1].grid()
plt.tight_layout()
plt.show()
```



```
In [122... print(f"Isi audio kiri: {audiokiri_voltinggi[95000:95100]}")
print(f"Isi audio kanan: {audiokanan_voltinggi[95000:95100]}")
```

```
Isi audio kiri: [ 1770  2875  4121  5404  6731  8105  9432 10602 11584 12383 12962 1
3247
13198 12818 12099 10992 9453 7517 5304 2952 556 -1800 -3991 -5875
-7389 -8565 -9413 -9878 -9949 -9759 -9482 -9162 -8732 -8167 -7544 -6927
-6289 -5584 -4828 -4099 -3494 -3104 -2961 -2978 -3009 -2970 -2875 -2756
-2614 -2463 -2309 -2068 -1611 -937 -236 333 869 1614 2668 3911
5194 6487 7807 9092 10224 11143 11869 12420 12751 12782 12475 11836
10843 9423 7559 5382 3095 811 -1457 -3642 -5573 -7126 -8313 -9164
-9618 -9646 -9412 -9137 -8863 -8460 -7874 -7221 -6614 -6018 -5347 -4613
-3920 -3380 -3063 -2973]
Isi audio kanan: [ 1770  2875  4121  5404  6731  8105  9432 10602 11584 12383 12962
13247
13198 12818 12099 10992 9453 7517 5304 2952 556 -1800 -3991 -5875
-7389 -8565 -9413 -9878 -9949 -9759 -9482 -9162 -8732 -8167 -7544 -6927
-6289 -5584 -4828 -4099 -3494 -3104 -2961 -2978 -3009 -2970 -2875 -2756
-2614 -2463 -2309 -2068 -1611 -937 -236 333 869 1614 2668 3911
5194 6487 7807 9092 10224 11143 11869 12420 12751 12782 12475 11836
10843 9423 7559 5382 3095 811 -1457 -3642 -5573 -7126 -8313 -9164
-9618 -9646 -9412 -9137 -8863 -8460 -7874 -7221 -6614 -6018 -5347 -4613
-3920 -3380 -3063 -2973]
```

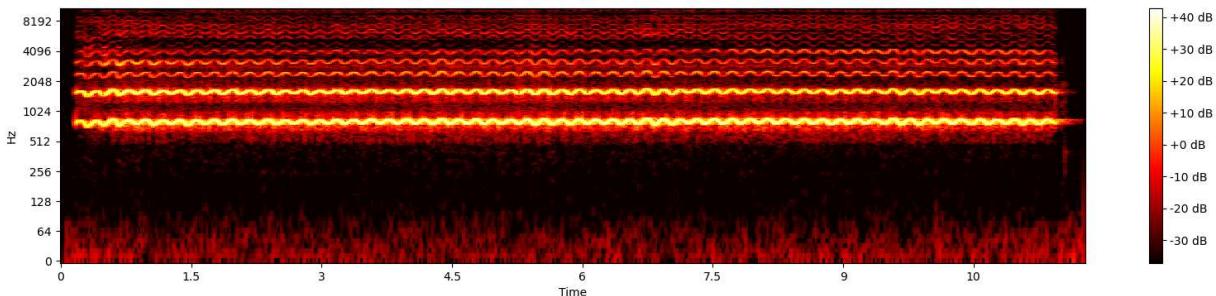
```
In [123... # Memuat audio menggunakan Librosa
data_audio, sr = librosa.load(volumetinggi)
S = librosa.stft(data_audio)
S_db = librosa.amplitude_to_db(np.abs(S))

print(f"Sampling Rate: {sr}")
```

Sampling Rate: 22050

```
In [124...]
```

```
# Plotting spektrogram
plt.figure(figsize=(20, 4))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='hot')
plt.colorbar(format='%+2.0f dB')
plt.show()
```



Sumbu x (waktu) menunjukkan bahwa audio ini memiliki durasi lebih dari 10 detik. Sumbu y menunjukkan frekuensi dalam Hz, dengan rentang dari 0 Hz hingga 8192 Hz. Pada frekuensi yang lebih rendah, sekitar 500 Hz hingga 2000 Hz, terlihat garis-garis horizontal yang kuat. Garis-garis ini mengindikasikan komponen sinyal yang dominan pada frekuensi tersebut. Beberapa frekuensi mendominasi sinyal, terlihat jelas di sekitar 1000 Hz, 2000 Hz, dan 3000 Hz. Amplitudo tertinggi berada pada +40 dB, yang diwakili oleh warna kuning terang, sedangkan bagian-bagian dengan amplitudo rendah mendekati -30 dB diwakili oleh warna hitam.

```
In [125...]
```

```
sss_sample = os.path.join(os.getcwd(), 'data', 'ssss.wav')
```

```
In [126...]
```

```
# Periksa jika file tidak ditemukan
if not os.path.exists(sss_sample):
    print(f"File {sss_sample} not found.")
    exit()
```

```
In [127...]
```

```
# Membaca file WAV
with wave.open(sss_sample, 'r') as wav_file:
    frames_sss = wav_file.readframes(-1)
    sample_sss = np.frombuffer(frames_sss, dtype='int16')
    fr_sss = wav_file.getframerate()

    # Mendapatkan informasi channel
    channel_sss = wav_file.getnchannels()
```

```
In [128...]
```

```
# Tampilkan informasi file WAV
print(f"Isi dari sound_data: {sample_sss}")
print(f"Panjang dari sound_data: {len(sample_sss)}")
print(f"Sampling rate: {fr_sss}")
print(f"Jumlah channel: {channel_sss}")
```

```
Isi dari sound_data: [      0       0       0 ...     662   -2968   -2968]
```

```
Panjang dari sound_data: 491520
```

```
Sampling rate: 48000
```

```
Jumlah channel: 2
```

In [129...]

```
# Membagi channel menjadi dua bagian (kiri dan kanan)
sample_sss = sample_sss.reshape(-1, channel_sss)
print(f"Panjang isi sample setelah direshape: {len(sample_sss)}")

audio_kiri_sss = sample_sss[:, 0]
audio_kanan_sss = sample_sss[:, 1]
print(f"Panjang audio kiri: {len(audio_kiri_sss)}")
print(f"Panjang audio kanan: {len(audio_kanan_sss)}")
```

Panjang isi sample setelah direshape: 245760

Panjang audio kiri: 245760

Panjang audio kanan: 245760

In [130...]

```
# Melakukan plotting audio kiri
panjang_sampel_sss = len(audio_kiri_sss)
print(f"Panjang sampel per sisi: {panjang_sampel_sss}")

durasi_sss = panjang_sampel_sss / fr_sss
print(f"Panjang sss.wav: {durasi_sss}")

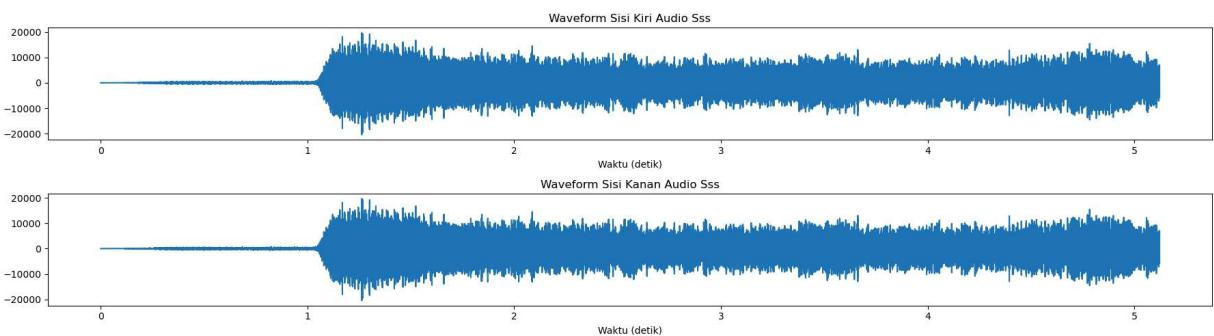
# Membuat sumbu waktu
time_axis = np.linspace(0, durasi_sss, panjang_sampel_sss)
```

Panjang sampel per sisi: 245760

Panjang sss.wav: 5.12

In [131...]

```
# Plotting audio kiri dan kanan
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis, audio_kiri_sss, label='Kiri')
fig[0].set_title('Waveform Sisi Kiri Audio Sss')
fig[0].set_xlabel('Waktu (detik)')
fig[1].plot(time_axis, audio_kanan_sss, label='Kanan')
fig[1].set_title('Waveform Sisi Kanan Audio Sss')
fig[1].set_xlabel('Waktu (detik)')
plt.tight_layout()
plt.show()
```

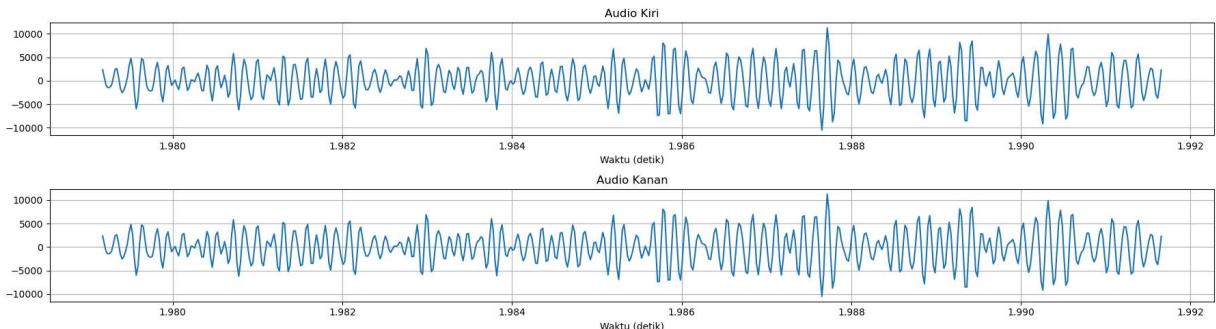


Dalam waveform audio ini menunjukkan dua kanal yaitu dengan kanal kiri dan kanal kanan. ada sedikit perbedaan kecil dalam frekuensi atau intensitas suara. Pada awal audio, antara 0 hingga 0,7 detik, amplitudo sangat rendah, mendekati nol, mengindikasikan adanya fase keheningan atau suara yang sangat pelan. Setelah itu, amplitudo meningkat secara signifikan, yang mengindikasikan masuknya suara yang lebih keras dan lebih dominan.

Selanjutnya, suara tetap terdengar dengan intensitas tinggi hingga akhir durasi audio, sekitar 5,5 detik. Puncak amplitudo pada kedua saluran mencapai sekitar ± 20.000 , menandakan suara dengan intensitas yang tinggi, kemungkinan berupa suara berkelanjutan atau bunyi yang cukup kuat. Setelah peningkatan amplitudo awal, suara tetap stabil tanpa penurunan amplitudo yang signifikan, menunjukkan bahwa audio ini mungkin berisi suara terus menerus seperti dengungan.

In [132...]

```
# Plotting audio kiri dan kanan dengan zoom di rentang durasi 1 detik
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis[95000:95600], audio_kirisss[95000:95600], label='Kiri')
fig[0].set_title('Audio Kiri')
fig[0].set_xlabel('Waktu (detik)')
fig[0].grid()
fig[1].plot(time_axis[95000:95600], audio_kanansss[95000:95600], label='Kanan')
fig[1].set_title('Audio Kanan')
fig[1].set_xlabel('Waktu (detik)')
fig[1].grid()
plt.tight_layout()
plt.show()
```



In [133...]

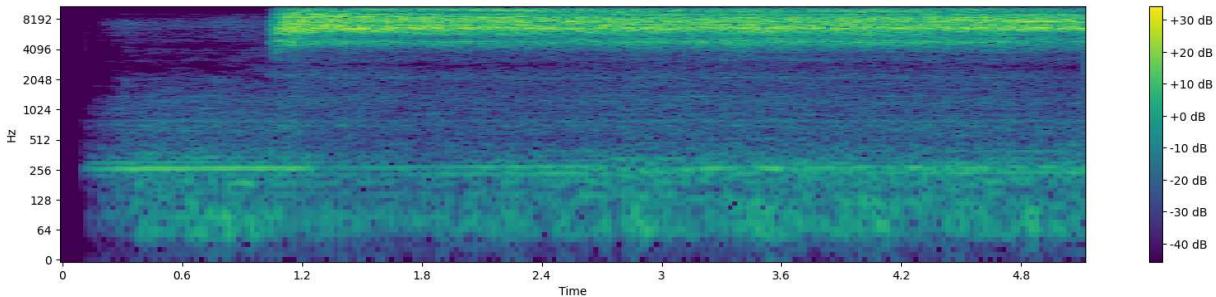
```
# Memuat audio menggunakan Librosa
data_audio, sr = librosa.load(sss_sample)
S = librosa.stft(data_audio)
S_db = librosa.amplitude_to_db(np.abs(S))

print(f"Sampling Rate: {sr}")
```

Sampling Rate: 22050

In [134...]

```
# Plotting spektrogram
plt.figure(figsize=(20, 4))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
plt.colorbar(format='%.2f dB')
plt.show()
```



Bagian utama dari sinyal audio terjadi pada awal sinyal, dengan aktivitas frekuensi tinggi dan menengah yang kuat. Ada aktivitas yang dominan pada rentang frekuensi antara 256 Hz hingga 4096 Hz, terutama pada detik pertama hingga sekitar detik 1,5. Setelah itu, energi frekuensi lebih tersebar, tetapi masih tetap aktif di frekuensi tinggi. Skala warna menunjukkan intensitas dalam dB. Area dengan warna kuning-hijau menunjukkan energi yang lebih kuat (sekitar +20 dB hingga +30 dB), sementara area dengan warna biru ke ungu menunjukkan energi yang lebih rendah (di bawah -20 dB). Pola ini menunjukkan suara frikatif seperti "Sss" yang stabil pada awalnya, yang kemudian perlahan menghilang.

```
In [135...]: brrb_sample = os.path.join(os.getcwd(), 'data', 'brrb.wav')
```

```
In [136...]: # Periksa jika file tidak ditemukan
if not os.path.exists(brrb_sample):
    print(f"File {brrb_sample} not found.")
    exit()
```

```
In [137...]: # Membaca file WAV
with wave.open(brrb_sample, 'r') as wav_file:
    frames_brrb = wav_file.readframes(-1)
    sample_brrb = np.frombuffer(frames_brrb, dtype='int16')
    fr_brrb = wav_file.getframerate()

    # Mendapatkan informasi channel
    channel_brrb = wav_file.getnchannels()
```

```
In [138...]: # Tampilkan informasi file WAV
print(f"Isi dari sound_data: {sample_brrb}")
print(f"Panjang dari sound_data: {len(sample_brrb)}")
print(f"Sampling rate: {fr_brrb}")
print(f"Jumlah channel: {channel_brrb}")
```

```
Isi dari sound_data: [ 0 0 0 ... -246 -233 -233]
Panjang dari sound_data: 538624
Sampling rate: 48000
Jumlah channel: 2
```

```
In [139...]: # Membagi channel menjadi dua bagian (kiri dan kanan)
sample_brrb = sample_brrb.reshape(-1, channel_brrb)
print(f"Panjang isi sample setelah direshape: {len(sample_brrb)}")

audio_kiri_brrb = sample_brrb[:, 0]
audio_kanan_brrb = sample_brrb[:, 1]
```

```
print(f"Panjang audio kiri: {len(audio_kiri_brrb)}")
print(f"Panjang audio kanan: {len(audio_kanan_brrb)}")
```

```
Panjang isi sample setelah direshape: 269312
Panjang audio kiri: 269312
Panjang audio kanan: 269312
```

In [140...]

```
# Melakukan plotting audio kiri
panjang_sampel_brrb = len(audio_kiri_brrb)
print(f"Panjang sampel per sisi: {panjang_sampel_brrb}")

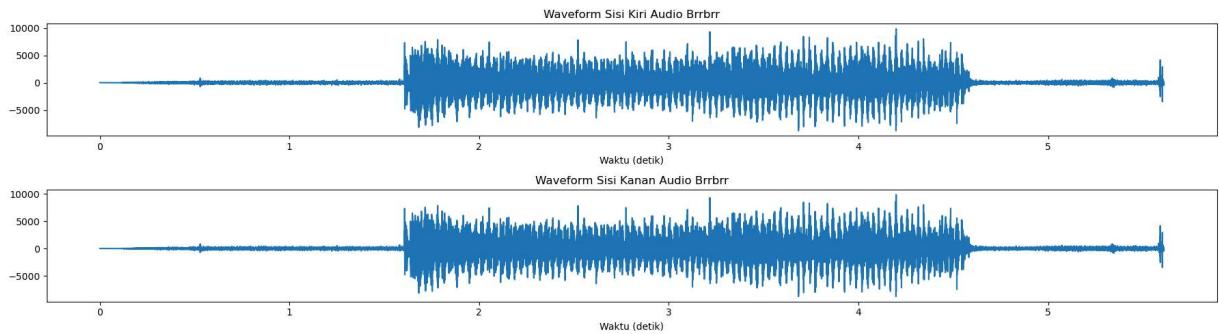
durasi_brrb = panjang_sampel_brrb / fr_brrb
print(f"Durasi brbrrr.wav: {durasi_brrb}")

# Membuat sumbu waktu
time_axis = np.linspace(0, durasi_brrb, panjang_sampel_brrb)
```

```
Panjang sampel per sisi: 269312
Durasi brbrrr.wav: 5.6106666666666667
```

In [141...]

```
# Plotting audio kiri dan kanan
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis, audio_kiri_brrb, label='Kiri')
fig[0].set_title('Waveform Sisi Kiri Audio Brrbrr')
fig[0].set_xlabel('Waktu (detik)')
fig[1].plot(time_axis, audio_kanan_brrb, label='Kanan')
fig[1].set_title('Waveform Sisi Kanan Audio Brrbrr')
fig[1].set_xlabel('Waktu (detik)')
plt.tight_layout()
plt.show()
```



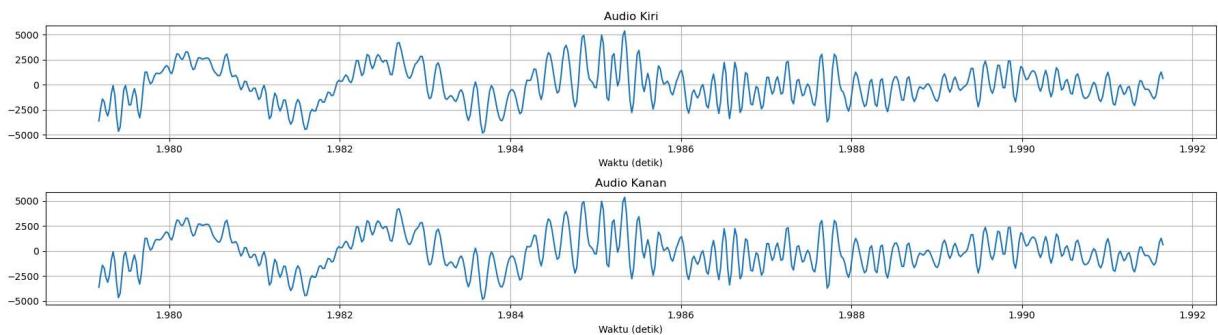
Dalam waveform audio ini menunjukkan dua kanal yaitu dengan kanal kiri dan kanal kanan. Pada bagian awal waveform, sekitar 0 hingga 1,5 detik, amplitudo sangat rendah mendekati nol, yang menunjukkan bahwa bagian awal audio relatif tenang atau hampir tidak ada suara yang terdengar. Namun, setelah titik ini, amplitudo mengalami peningkatan yang tajam, menandakan masuknya suara yang lebih keras dan lebih intens. Fase ini dimulai sekitar 1,5 detik dan berlangsung hingga sekitar 4,5 detik, di mana amplitudo tetap tinggi, mengindikasikan adanya aktivitas suara yang kuat dan berkelanjutan selama periode tersebut.

Amplitudo puncak di saluran kiri mencapai sekitar ± 10.000 , sementara di saluran kanan mencapai nilai yang sedikit lebih tinggi, menandakan bahwa ada sedikit perbedaan dalam intensitas suara antara kedua saluran, meskipun pola dasar tetap serupa. Setelah puncak

amplitudo ini, gelombang mulai menurun secara bertahap dan kembali ke amplitudo rendah sekitar 4,5 detik, menandakan bahwa suara secara bertahap berkurang hingga akhirnya hilang mendekati akhir durasi audio, yang berlangsung hingga sekitar 5,5 detik.

In [142...]

```
# Plotting audio kiri dan kanan dengan zoom di rentang durasi 1 detik
ax, fig = plt.subplots(2, 1, figsize=(18, 5))
fig[0].plot(time_axis[95000:95600], audio_kiri_brrb[95000:95600], label='Kiri')
fig[0].set_title('Audio Kiri')
fig[0].set_xlabel('Waktu (detik)')
fig[0].grid()
fig[1].plot(time_axis[95000:95600], audio_kanan_brrb[95000:95600], label='Kanan')
fig[1].set_title('Audio Kanan')
fig[1].set_xlabel('Waktu (detik)')
fig[1].grid()
plt.tight_layout()
plt.show()
```



In [143...]

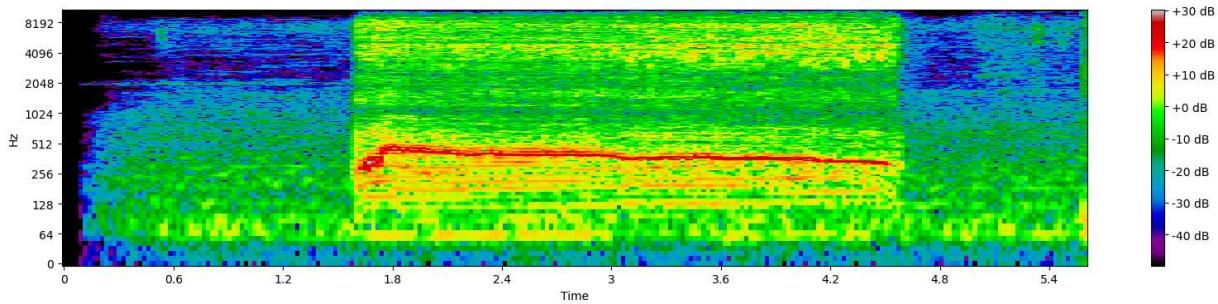
```
# Memuat audio menggunakan Librosa
data_audio, sr = librosa.load(brrb_sample)
S = librosa.stft(data_audio)
S_db = librosa.amplitude_to_db(np.abs(S))

print(f"Sampling Rate: {sr}")
```

Sampling Rate: 22050

In [144...]

```
# Plotting spektrogram
plt.figure(figsize=(20, 4))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='nipy_spect'
plt.colorbar(format='%+2.0f dB')
plt.show()
```



Sebagian besar energi berada di frekuensi 256 Hz hingga 512 Hz, terutama di bagian tengah-tengah durasi audio. Ada sedikit aktivitas di frekuensi rendah (0 hingga 128 Hz) dan

aktivitas frekuensi tinggi (di atas 4096 Hz), tetapi intensitasnya rendah. Pada detik 1,8 terjadi peningkatan intensitas signifikan yang bertahan hingga sekitar detik 4,5. Setelah itu, amplitudo kembali menurun. Aktivitas frekuensi tertinggi dan energi tertinggi tampak berada pada frekuensi 256 Hz hingga 512 Hz, di mana terdapat garis merah yang sangat stabil di sekitar frekuensi 300 Hz. Berdasarkan frekuensi dominan di sekitar 300 Hz dan onset intensitas tiba-tiba di sekitar detik 1,8, sinyal ini berasal dari suara yang berosilasi atau suara seperti getaran yang stabil, seperti suara "brrr".

2. Tanyalah kepada AI-LLM bagaimana membuat teknik fading yang non linear.

Implementasikan hal tersebut. Jangan lupa copy/paste hasil percakapan anda dengan AI LLM ke notebook anda.

- Untuk mahasiswa ber-nim akhir ganjil, implementasikan teknik fading Logarithmic Scale Fading
- Untuk mahasiswa ber-nim akhir genap, implementasikan teknik fading Exponential
- Lakukan fading untuk bagian awal dan akhir dari audio

Pertanyaan dengan GPT.

Q: bagaimana cara melakukan fading non linear dengan teknik eksponensial?

A: Untuk melakukan **fading non-linear** dengan teknik **eksponensial**, kita bisa memodifikasi amplitudo sinyal audio dengan fungsi eksponensial. Fading eksponensial memungkinkan perubahan volume terjadi secara bertahap, tidak linear, tetapi dengan laju yang semakin meningkat atau melambat, tergantung apakah itu fading in atau fading out.

1. Exponential Fade In:

Pada **fade in**, volume secara perlahan meningkat dari 0 ke nilai maksimum secara eksponensial.

2. Exponential Fade Out:

Pada **fade out**, volume secara eksponensial menurun dari nilai maksimum ke 0.

Langkah-langkah:

1. **Buat Envelope Eksponensial:** Kita buat sebuah fungsi eksponensial yang akan bertindak sebagai *envelope* untuk melakukan fading pada audio. Ini bisa dilakukan dengan menggunakan `np.exp` untuk eksponensial.

2. **Terapkan Envelope pada Sinyal Audio:** Setelah envelope eksponensial dibuat, kita menerapkannya dengan mengalikan envelope tersebut pada sinyal audio.

Contoh Kode Fading Eksponensial:

```

import numpy as np
import soundfile as sf
import matplotlib.pyplot as plt

# Fungsi untuk melakukan fading eksponensial
def exponential_fade(audio, fade_type='in'):
    length = len(audio)

    # Generate time vector untuk fading
    t = np.linspace(0, 1, length)

    if fade_type == 'in':
        envelope = np.exp(t) - 1
        envelope = envelope / max(envelope) # Normalisasi
    elif fade_type == 'out':
        envelope = np.exp(-t)
        envelope = envelope / max(envelope) # Normalisasi

    faded_audio = audio * envelope
    return faded_audio

# Load audio file (ambil satu contoh audio untuk demo)
audio_file = "/mnt/data/spoken.wav"
y, sr = sf.read(audio_file)

# Lakukan fade in dan fade out
audio_fade_in = exponential_fade(y, fade_type='in')
audio_fade_out = exponential_fade(y, fade_type='out')

# Plot untuk visualisasi
fig, ax = plt.subplots(3, 1, figsize=(18, 8))

# Plot audio original
ax[0].plot(y)
ax[0].set_title('Original Audio')
ax[1].plot(audio_fade_in)
ax[1].set_title('Exponential Fade In')
ax[2].plot(audio_fade_out)
ax[2].set_title('Exponential Fade Out')
plt.tight_layout()
plt.show()

# Save hasil audio
sf.write('faded_in.wav', audio_fade_in, sr)
sf.write('faded_out.wav', audio_fade_out, sr)

```

Penjelasan:

1. Envelope Eksponensial:

- Untuk **fade in**, envelope `np.exp(t) - 1` membuat amplitudo meningkat secara eksponensial dari 0 menuju nilai maksimum.
- Untuk **fade out**, envelope `np.exp(-t)` membuat amplitudo menurun secara eksponensial dari nilai maksimum ke 0.

2. **Terapkan Envelope:** Envelope eksponensial dikalikan dengan sinyal audio untuk melakukan fading.

3. Plotting:

- Audio asli, audio dengan **fade in**, dan audio dengan **fade out** divisualisasikan dengan grafik untuk melihat perubahannya.

4. **Simpan Hasil:** Audio hasil fading disimpan ke dalam file `faded_in.wav` dan `faded_out.wav`.

Dengan metode ini, kamu bisa membuat **fading eksponensial** baik untuk **fade in** maupun **fade out** pada audio.

```
In [145...]: import soundfile as sf
import numpy as np
import os
import wave
import matplotlib.pyplot as plt
import librosa
import librosa.display
```

```
In [146...]: fadingpercobaan = os.path.join(os.getcwd(), 'data', 'bird.wav')
```

```
In [147...]: # Periksa jika file tidak ditemukan
if not os.path.exists(fadingpercobaan):
    print(f"File {fadingpercobaan} not found.")
    exit()
```

```
In [148...]: # Membaca file WAV
with wave.open(fadingpercobaan, 'r') as wav_file:
    frames = wav_file.readframes(-1)
    sample = np.frombuffer(frames, dtype='int16')
    fr = wav_file.getframerate()

    # Mendapatkan informasi channel
    channel = wav_file.getnchannels()
```

```
In [149...]: # Tampilkan informasi file WAV
print(f"Isi dari sound_data: {sample}")
print(f"Panjang dari sound_data: {len(sample)}")
print(f"Sampling rate: {fr}")
print(f"Jumlah channel: {channel}")
```

```
Isi dari sound_data: [ 7561  5161 -1797 ... -183 -410 -640]
Panjang dari sound_data: 1795328
Sampling rate: 32000
Jumlah channel: 1
```

```
In [150...]
# Fungsi untuk melakukan fading eksponensial
def fadingeksponensial(fadingpercobaan, fade_type='in'):
    length = len(fadingpercobaan)

    # Generate time vector untuk fading
    t = np.linspace(0, 1, length)

    if fade_type == 'in':
        # Fade in eksponensial
        envelope = np.exp(t) - 1
        envelope = envelope / max(envelope) # Normalisasi
    elif fade_type == 'out':
        # Fade out eksponensial
        envelope = np.exp(-t)
        envelope = envelope / max(envelope) # Normalisasi

    # Terapkan envelope pada sinyal audio
    faded_audio = fadingpercobaan * envelope
    return faded_audio
```

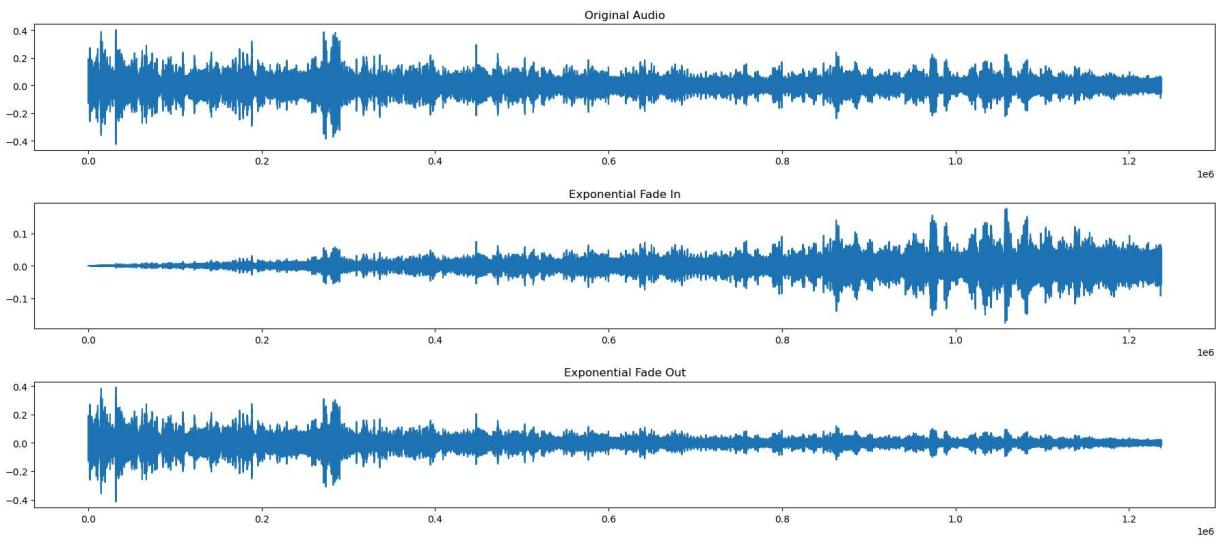
Untuk melakukan fading teknik eksponensial, yang pertama adalah membuat fungsi fadingeksponensial. Di length, panjang (jumlah sampel) dari sinyal audio dihitung menggunakan len() untuk menentukan berapa banyak titik yang ada pada envelope fade yang akan dibuat. Kemudian normalisasi dilakukan dengan membagi envelope dengan nilai maksimum agar skalanya tetap 0 hingga 1. Setelah envelope dihitung, envelope tersebut diterapkan pada sinyal audio dengan mengalikan nilai envelope ke setiap sampel pada sinyal audio (fadingpercobaan). Hasilnya adalah sinyal audio yang mengalami fading eksponensial.

```
In [151...]
# Load audio file
y, sr = librosa.load(fadingpercobaan)

# Lakukan fade in dan fade out
audio_fade_in = fadingeksponensial(y, fade_type='in')
audio_fade_out = fadingeksponensial(y, fade_type='out')
```

Tahapan berikutnya adalah penerapan fading eksponensial pada sinyal audio, baik untuk fade in (peningkatan volume) maupun fade out (penurunan volume).

```
In [152...]
# Plot untuk visualisasi
fig, ax = plt.subplots(3, 1, figsize=(18, 8))
# Plot audio original
ax[0].plot(y)
ax[0].set_title('Original Audio')
# Plot audio dengan fade in
ax[1].plot(audio_fade_in)
ax[1].set_title('Exponential Fade In')
# Plot audio dengan fade out
ax[2].plot(audio_fade_out)
ax[2].set_title('Exponential Fade Out')
plt.tight_layout()
plt.show()
```



Pada tahapan ini, akan menampilkan bentuk sinyal audio sebelum dan sesudah diterapkan efek fading eksponensial (baik fade in maupun fade out).

```
In [153...]: # Save hasil audio di folder 'data'
sf.write('data/bird_faded_in.wav', audio_fade_in, sr)
sf.write('data/bird_faded_out.wav', audio_fade_out, sr)
```

Dengan cara ini, file audio hasil fading akan disimpan di dalam folder data.

3. Dengan menggunakan file audio yang anda rekam sendiri (suara anda), lakukanlah kompresi dan normalisasi hingga loudness LUFS mencapai kira-kira sekitar -14 LUFS. Berikan penjelasan langkah-langkah yang anda lakukan untuk menyelesaikan tugas ini.

```
In [1]: import soundfile as sf
import numpy as np
import os
import wave
import matplotlib.pyplot as plt
import librosa
import librosa.display
```

```
In [2]: percobaan3 = os.path.join(os.getcwd(), 'data', 'aboutme.wav')
```

```
In [3]: # Periksa jika file tidak ditemukan
if not os.path.exists(percobaan3):
    print(f"File {percobaan3} not found.")
    exit()
```

```
In [4]: # Membaca file WAV
with wave.open(percobaan3, 'r') as wav_file:
    frames_percobaan3 = wav_file.readframes(-1)
    sample_percobaan3 = np.frombuffer(frames_percobaan3, dtype='int16')
    fr_percobaan3 = wav_file.getframerate()
```

```
# Mendapatkan informasi channel
channel_percobaan3 = wav_file.getnchannels()
```

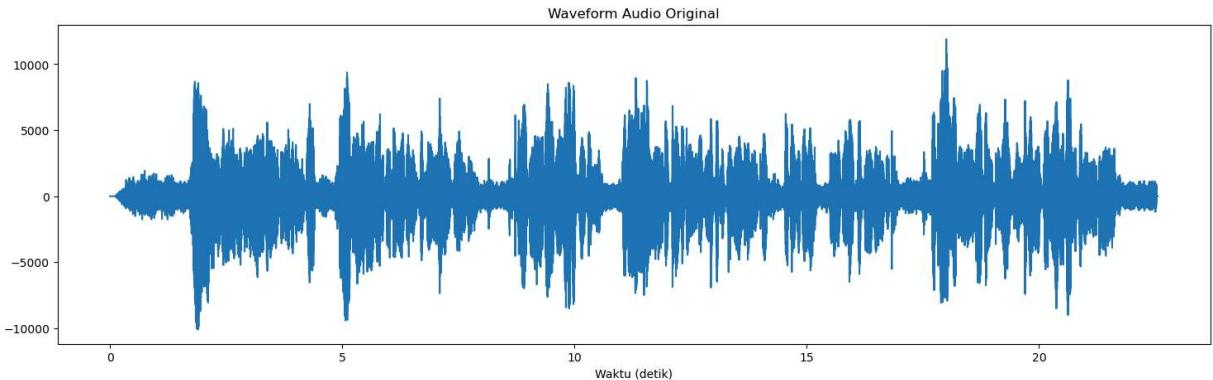
```
In [5]: # Tampilkan informasi file WAV
print(f"Isi dari sound_data: {sample_percobaan3}")
print(f"Panjang dari sound_data: {len(sample_percobaan3)}")
print(f"Sampling rate: {fr_percobaan3}")
print(f"Jumlah channel: {channel_percobaan3}")
```

```
Isi dari sound_data: [0 0 0 ... 0 0 0]
Panjang dari sound_data: 1082880
Sampling rate: 48000
Jumlah channel: 1
```

```
In [6]: # Membuat sumbu waktu
time_axis_original = np.linspace(0, len(sample_percobaan3) / fr_percobaan3, len(sample_percobaan3))
print(f"Panjang time axis : {len(time_axis_original)}")
```

```
Panjang time axis : 1082880
```

```
In [7]: # Plotting audio original
plt.figure(figsize=(18, 5))
plt.plot(time_axis_original, sample_percobaan3)
plt.title('Waveform Audio Original')
plt.xlabel('Waktu (detik)')
plt.show()
```



```
In [8]: # Mencari nilai maksimal dari audio
nilai_maksimal = np.max(np.abs(sample_percobaan3))
print(f"Nilai maksimal dari audio: {nilai_maksimal}")

# Membuat threshold kompresi
thresholdkompresi = 0.4 * nilai_maksimal
print(f"Threshold kompresi: {thresholdkompresi}")
```

```
Nilai maksimal dari audio: 11899
Threshold kompresi: 4759.6
```

```
In [9]: # Melakukan kompresi
kompresiperobaan3 = sample_percobaan3.copy()
```

```
In [10]: # Mengganti nilai yang lebih kecil dari threshold dengan nilai threshold
mask = np.abs(kompresiperobaan3) > thresholdkompresi
```

```
In [11]: print(f"Jumlah nilai yang lebih besar dari threshold: {np.sum(mask)}")
```

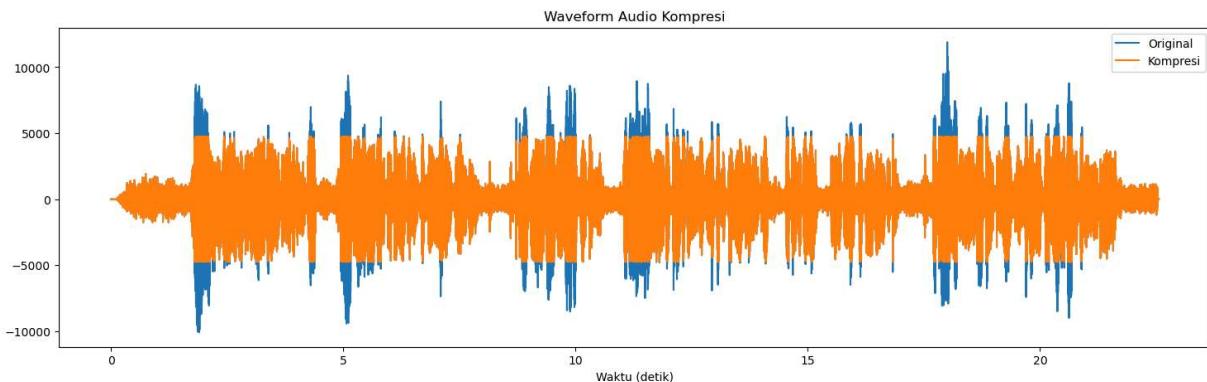
Jumlah nilai yang lebih besar dari threshold: 13712

```
In [12]: rasiokompresi = 5
```

```
# Kompresi nilai yang lebih kecil dari threshold
kompresipercobaan3[mask] = kompresipercobaan3[mask] / rasiokompresi
kompresipercobaan3 = np.clip(kompresipercobaan3, -nilai_maksimal, nilai_maksimal)
```

```
In [13]: time_axis_kompresi = np.linspace(0, len(kompresipercobaan3) / fr_percobaan3, len(ko
```

```
In [14]: plt.figure(figsize=(18, 5))
plt.plot(time_axis_original, sample_percobaan3, label='Original')
plt.plot(time_axis_kompresi, kompresipercobaan3, label='Kompresi')
plt.title('Waveform Audio Kompresi')
plt.xlabel('Waktu (detik)')
plt.tight_layout()
plt.legend()
plt.show()
```



```
In [15]: # Menentukan a dan b
b = np.array([1.53512485958697, -2.69169618940638, 1.19839281085285])
a = np.array([1.0, -1.69065929318241, 0.73248077421585])
```

```
In [16]: # Membuat array kosong untuk menyimpan hasil filter
floatpercobaan3 = sample_percobaan3.astype(np.float32) / np.iinfo(np.int16).max
percobaan3difilter = np.zeros_like(floatpercobaan3)

# Melakukan filter
for i in range(len(floatpercobaan3)):
    if i < 2:
        percobaan3difilter[i] = floatpercobaan3[i]
    else:
        percobaan3difilter[i] = b[0] * floatpercobaan3[i] + b[1] * floatpercobaan3[i-1] + b[2] * floatpercobaan3[i-2]
```

```
In [17]: # Menghitung mean square
mean_square = np.mean(np.square(percobaan3difilter))
print(f"Mean square dari audio: {mean_square}")
```

Mean square dari audio: 0.002437644638121128

```
In [18]: # Periksa jika mean_square > 0
if mean_square > 0:
    lufs = -0.691 + 10 * np.log10(mean_square)
else:
    lufs = float('nan') # output jika mean_square <= 0

print(f"LUFS: {lufs}")
```

```
LUFS: -26.821297183990477
```

Agar LUFS mendekati -14, maka bisa dikalikan dengan gain factor. Gain factor digunakan untuk menyesuaikan level loudness dari audio agar mencapai target loudness yang diinginkan. Rumus untuk menghitung gain factor adalah sebagai berikut:

Gain factor = $10 \times (\text{Target LUFS} - \text{LUFS Sementara}) / 20$

Di mana:

- Target LUFS adalah level loudness yang diinginkan = -14 LUFS.
- LUFS Sementara adalah level loudness saat ini.

Sehingga, gain factor yang dibutuhkan adalah 1.38

```
In [34]: gain_factor = 1.38
```

```
In [35]: percobaan3normalisasi = float(percobaan3 * gain_factor)
percobaan3normalisasi_int16 = (percobaan3normalisasi * np.iinfo(np.int16).max).astype(np.int16)

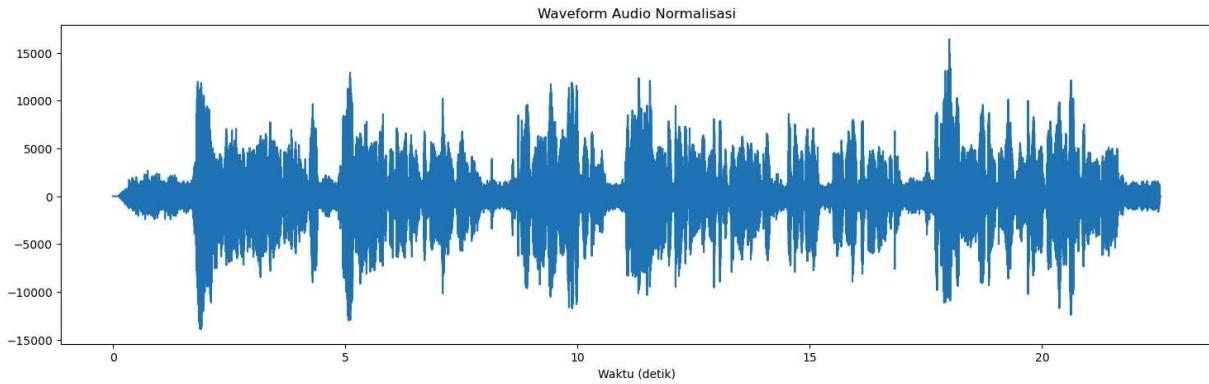
print(f"Nilai maksimal dari audio setelah normalisasi: {np.max(np.abs(percobaan3nor...")
```

```
Nilai maksimal dari audio setelah normalisasi: 16420
```

```
In [36]: # Simpan audio yang dinormalisasi
sf.write('data/normalisasi_percobaan3.wav', percobaan3normalisasi_int16, fr_percoba...
```

```
In [37]: # Waveform audio yang dinormalisasi
time_axis_normalisasi = np.linspace(0, len(percobaan3normalisasi_int16) / fr_percoba...
```

```
plt.figure(figsize=(18, 5))
plt.plot(time_axis_normalisasi, percobaan3normalisasi_int16)
plt.title('Waveform Audio Normalisasi')
plt.xlabel('Waktu (detik)')
plt.show()
```



```
In [38]: with wave.open ('data/normalisasi_percobaan3.wav', 'r') as wav_file:
    frames_normalisasi = wav_file.readframes(-1)
    sample_normalisasi = np.frombuffer(frames_normalisasi, dtype='int16')
    fr_normalisasi = wav_file.getframerate()

    # Mendapatkan informasi channel
    channel_normalisasi = wav_file.getnchannels()
```

```
In [39]: # Tampilkan informasi file WAV
print(f"Isi data audio yang dinormalisasi: {sample_normalisasi}")
print(f"Panjang data audio yang dinormalisasi: {len(sample_normalisasi)}")
print(f"Sampling rate: {fr_normalisasi}")
print(f"Jumlah channel: {channel_normalisasi}")
```

```
Isi data audio yang dinormalisasi: [0 0 0 ... 0 0 0]
Panjang data audio yang dinormalisasi: 1082880
Sampling rate: 48000
Jumlah channel: 1
```

```
In [40]: sample_normalisasi = sample_normalisasi / 10390
```

```
In [41]: def calculate_lufs(sample_normalisasi):
    # Langkah 1: Hitung Mean Square (MS)
    ms = np.mean(np.square(sample_normalisasi))

    # Langkah 2: Hitung LUFS dengan rumus Logaritmik
    lufs = -0.691 + 10 * np.log10(ms)
    return lufs

lufs_normalisasi = calculate_lufs(sample_normalisasi)
```

```
In [42]: print(f"LUFS dari audio yang dinormalisasi: {lufs_normalisasi}")
```

```
LUFS dari audio yang dinormalisasi: -14.627620918559584
```

4. Rekamlah sebuah audio dengan menggunakan handphone / laptop anda. Cara anda merekam haruslah sedikit unik:

- Anda harus berbicara selama 20 detik.
- Bacaan / percakapan yang anda rekam haruslah berisi informasi mengenai diri anda, seperti nama, asal daerah, hobi, dll.

- Lakukanlah perekaman di dekat sumber noise statis seperti kipas angin, AC, atau kipas laptop anda (atau apapun yang merupakan noise frekuensi tinggi)
- Lakukanlah equalisasi pada audio tersebut dengan menggunakan bandpass. Temukan frekuensi cutoff untuk bandpass yang paling sesuai dengan karakteristik audio yang anda rekam.
- Visualisasikan spektrum frekuensi dari audio sebelum di filter dan setelah di filter (dengan ketiga filter yang telah anda buat)

```
In [266...]: import soundfile as sf
import numpy as np
import os
import wave
import matplotlib.pyplot as plt
import librosa
import librosa.display

In [267...]: percobaanke4 = os.path.join(os.getcwd(), 'data', 'aboutme.wav')

In [268...]: # Periksa jika file tidak ditemukan
if not os.path.exists(percobaanke4):
    print(f"File {percobaanke4} not found.")
    exit()

In [269...]: # Membaca file WAV
with wave.open(percobaanke4, 'r') as wav_file:
    frames_percobaanke4 = wav_file.readframes(-1)
    sample_percobaanke4 = np.frombuffer(frames_percobaanke4, dtype='int16')
    fr_percobaanke4 = wav_file.getframerate()

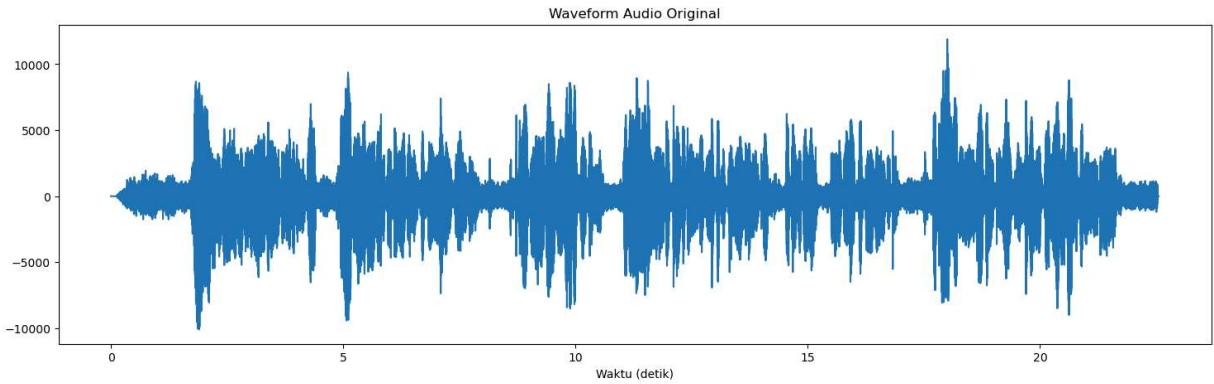
    # Mendapatkan informasi channel
    channel_percobaanke4 = wav_file.getnchannels()

In [270...]: # Tampilkan informasi file WAV
print(f"Isi dari data audio: {sample_percobaanke4}")
print(f"Panjang data audio: {len(sample_percobaanke4)}")
print(f"Sampling rate audio: {fr_percobaanke4}")
print(f"Jumlah channel audio: {channel_percobaanke4}")

Isi dari data audio: [0 0 0 ... 0 0 0]
Panjang data audio: 1082880
Sampling rate audio: 48000
Jumlah channel audio: 1

In [271...]: # Membuat sumbu waktu
time_axis_original = np.linspace(0, len(sample_percobaanke4) / fr_percobaanke4, len(sample_percobaanke4))

# Plotting audio original
plt.figure(figsize=(18, 5))
plt.plot(time_axis_original, sample_percobaanke4)
plt.title('Waveform Audio Original')
plt.xlabel('Waktu (detik)')
plt.show()
```



```
In [272...]: from scipy.signal import butter, filtfilt
```

```
In [274...]: def plot_frequency_spectrum(sample_percobaanke4, fr_percobaanke4, title):
    fft_data = np.fft.fft(sample_percobaanke4)
    frequencies = np.fft.fftfreq(len(fft_data), 1 / fr_percobaanke4)

    positive_frequencies = frequencies[:len(frequencies) // 2]
    magnitude = np.abs(fft_data[:len(frequencies) // 2])

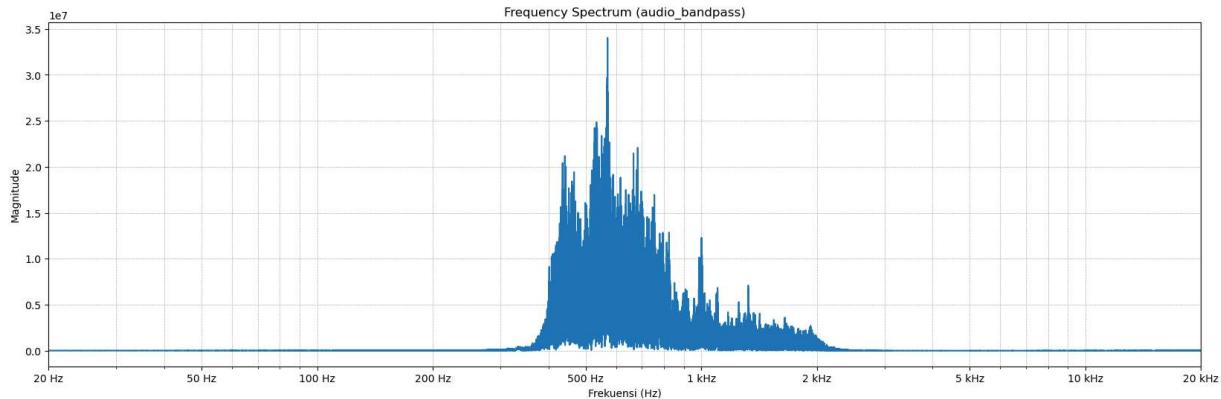
    plt.figure(figsize=(20, 6))
    plt.plot(positive_frequencies, magnitude)
    plt.xscale('log')
    plt.xlim(20, 20000) # Rentang frekuensi yang relevan untuk audio
    plt.xticks([20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000],
               ['20 Hz', '50 Hz', '100 Hz', '200 Hz', '500 Hz', '1 kHz', '2 kHz', ''])
    plt.title(title)
    plt.xlabel('Frekuensi (Hz)')
    plt.ylabel('Magnitude')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.show()
```

```
In [275...]: cut_off_low = 400 # hz, artinya frekuensi di bawah 400 hz akan dibuang
cut_off_high = 2000 # hz, artinya frekuensi di atas 2000 hz akan dibuang
order = 6

cut_off_low_nyquist = cut_off_low / (fr_percobaanke4 / 2)
cut_off_high_nyquist = cut_off_high / (fr_percobaanke4 / 2)

b,a = butter(order, [cut_off_low_nyquist, cut_off_high_nyquist], btype='band')
audio_bandpass = filtfilt(b, a, sample_percobaanke4)
audio_bandpass = audio_bandpass.astype(np.int16)

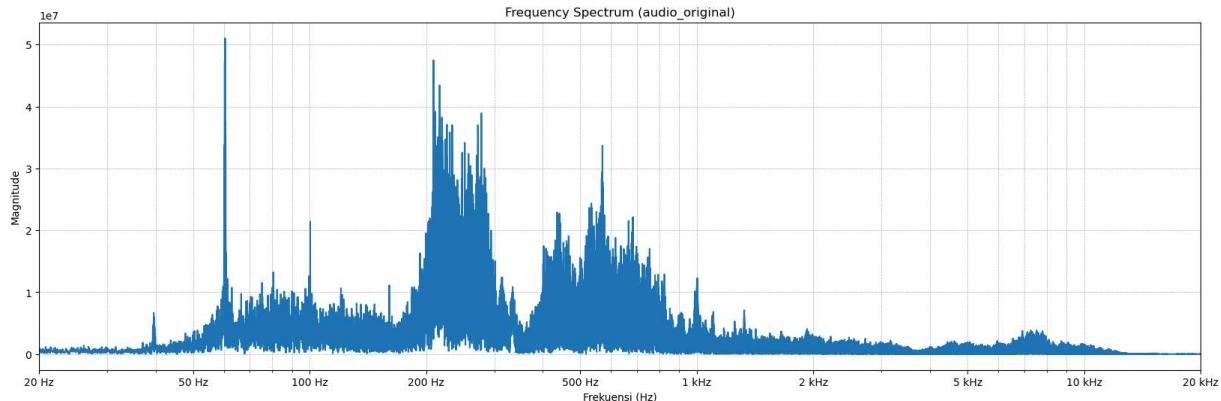
plot_frequency_spectrum(audio_bandpass, fr_percobaanke4, 'Frequency Spectrum (audio
```



```
In [277]: bandpasssample5 = os.path.join(os.getcwd(), 'data', 'aboutme_bandpass.wav')

with wave.open(bandpasssample5, 'w') as wav_file:
    wav_file.setnchannels(channel_percobaanke4)
    wav_file.setsampwidth(2)
    wav_file.setframerate(fr_percobaanke4)
    wav_file.writeframes(audio_bandpass.tobytes())
```

```
In [279]: plot_frequency_spectrum(sample_percobaanke4, fr_percobaanke4, 'Frequency Spectrum (
```



5. Tanyakan pada AI/LLM bagaimana cara membuat noise gate pada audio. Lalu implementasikan noise gate ini pada audio yang telah anda rekam. Jangan lupa copy/paste hasil percakapan anda dengan AI LLM ke notebook anda. Gunakan file audio dari soal nomor 4.

Q : langkah langkah membuat noise gate pada audio A : **Noise gate** adalah teknik yang digunakan dalam pengolahan audio untuk mengurangi atau menghilangkan kebisingan latar belakang (background noise) dengan menutup "gate" ketika sinyal audio berada di bawah ambang batas tertentu (threshold). Ini memungkinkan bagian audio yang lebih keras lewat, sementara bagian yang lebih pelan (seperti noise atau desisan) dihilangkan atau direddam.

Berikut adalah langkah-langkah untuk membuat **noise gate** pada audio:

1. Membaca Audio

Langkah pertama adalah memuat file audio yang ingin kita proses.

2. Menentukan Threshold

Kita perlu menetapkan **threshold** atau ambang batas, yaitu level amplitudo minimum yang diperbolehkan untuk melewati "gate". Sinyal audio dengan amplitudo di bawah threshold akan diredam (mute), sedangkan sinyal di atas threshold akan dibiarkan lewat.

3. Membuat Masking

Masking digunakan untuk menentukan bagian audio mana yang berada di bawah threshold dan harus diredam.

4. Terapkan Noise Gate

Pada bagian audio yang amplitudonya berada di bawah threshold, kita menerapkan pengurangan atau penghilangan sinyal.

5. Simpan Hasil Audio

Setelah noise gate diterapkan, simpan hasilnya ke file audio baru.

Penjelasan Langkah-Langkah:

1. Baca File Audio:

- `y, sr = sf.read(audio_file)` : Membaca file audio ke dalam array `y` dan menyimpan sampling rate dalam `sr`.

2. Menentukan Threshold:

- **Threshold** adalah level di mana sinyal audio diizinkan lewat. Sinyal dengan amplitudo di bawah threshold akan diredam.
- Dalam contoh ini, threshold ditetapkan pada **0.01** (nilai amplitudo linear), tetapi dapat disesuaikan tergantung pada tingkat noise dan sinyal.

3. Membuat Masking:

- `mask = np.abs(y) > threshold` : Masking digunakan untuk menentukan bagian sinyal yang amplitudonya lebih besar dari threshold. Bagian sinyal di bawah threshold akan dihilangkan.

4. Terapkan Noise Gate:

- `y_gated = y * mask` : Mengalikan sinyal asli dengan mask, sehingga bagian yang di bawah threshold menjadi nol (mute).

5. Simpan Hasil:

- `sf.write()` : Menyimpan hasil audio yang sudah diproses ke file baru.

6. (Optional) Visualisasi:

- Dengan menggunakan `matplotlib`, kita bisa memvisualisasikan bentuk gelombang audio sebelum dan sesudah penerapan noise gate.

Opsi Tambahan untuk Noise Gate:

- **Attack Time** dan **Release Time**: Noise gate sering kali memiliki **attack time** (waktu yang dibutuhkan untuk membuka gate) dan **release time** (waktu untuk menutup gate setelah sinyal turun di bawah threshold). Ini bisa ditambahkan untuk memberikan kontrol yang lebih halus.
- **Knee**: Dalam beberapa noise gate, ada konsep "knee", yang memungkinkan transisi yang lebih halus antara sinyal yang melewati gate dan yang tidak.

Penyesuaian:

- **Threshold**: Sesuaikan threshold berdasarkan tingkat kebisingan yang ingin kamu hilangkan. Misalnya, jika noise berada pada level yang sangat rendah, threshold bisa lebih tinggi.
- **Ratio**: Kamu juga bisa menambahkan pengaturan **ratio** untuk menentukan seberapa agresif gate akan memblokir sinyal di bawah threshold.

Dengan mengikuti langkah-langkah ini, kamu dapat menerapkan **noise gate** pada audio untuk menghilangkan kebisingan yang tidak diinginkan. Jika ada pertanyaan lebih lanjut atau jika kamu ingin menambahkan fitur lain pada noise gate, beri tahu saya!

In [280...]

```
import soundfile as sf
import numpy as np
import os
import wave
import matplotlib.pyplot as plt
import librosa
import librosa.display
```

In [281...]

```
noisegatesample = os.path.join(os.getcwd(), 'data', 'aboutme.wav')
```

In [282...]

```
# Periksa jika file tidak ditemukan
if not os.path.exists(noisegatesample):
    print(f"File {noisegatesample} not found.")
    exit()
```

In [283...]

```
# Membaca file WAV
with wave.open(noisegatesample, 'r') as wav_file:
    frames_noisegate = wav_file.readframes(-1)
    sample_noisegate = np.frombuffer(frames_noisegate, dtype='int16')
    fr_noisegate = wav_file.getframerate()
```

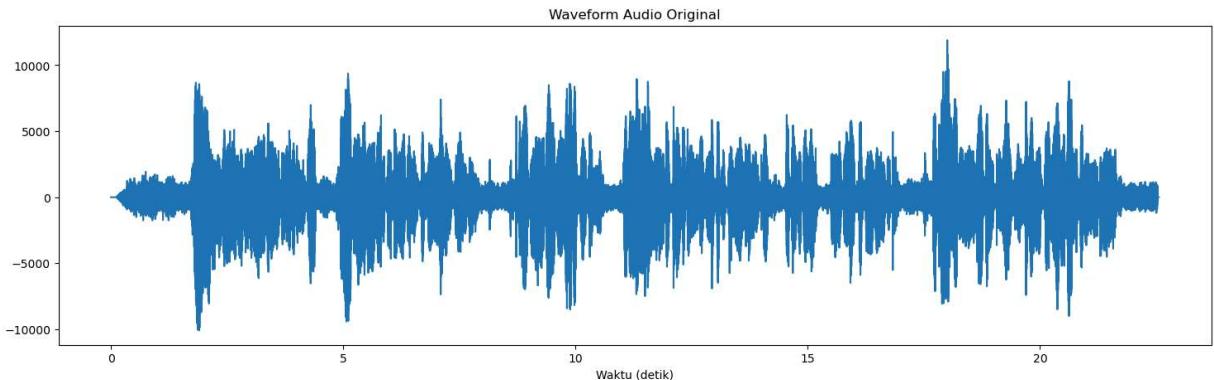
```
# Mendapatkan informasi channel
channel_noisegate = wav_file.getnchannels()
```

```
In [284...]: # Tampilkan informasi file WAV
print(f"Isi dari data audio: {sample_noisegate}")
print(f"Panjang data audio: {len(sample_noisegate)}")
print(f"Sampling rate audio: {fr_noisegate}")
print(f"Jumlah channel audio: {channel_noisegate}")
```

```
Isi dari data audio: [0 0 0 ... 0 0 0]
Panjang data audio: 1082880
Sampling rate audio: 48000
Jumlah channel audio: 1
```

```
In [285...]: # Membuat sumbu waktu
time_axis_original = np.linspace(0, len(sample_noisegate) / fr_noisegate, len(sample_noisegate))

# Plotting audio original
plt.figure(figsize=(18, 5))
plt.plot(time_axis_original, sample_noisegate)
plt.title('Waveform Audio Original')
plt.xlabel('Waktu (detik)')
plt.show()
```



```
In [286...]: # Tentukan threshold untuk noise gate
threshold = 0.01
```

```
In [288...]: # Buat mask untuk menentukan bagian audio di bawah threshold
mask = np.abs(sample_noisegate) > threshold
```

```
In [289...]: sample_noisegate_gated = sample_noisegate * mask
```

```
In [290...]: # Simpan hasil audio setelah diterapkan noise gate
sf.write('data/aboutme_noise_gate.wav', sample_noisegate_gated, sr)
```

```
In [295...]: aboutme_noise_gate = os.path.join(os.getcwd(), 'data', 'aboutme_noise_gate.wav')
```

```
In [296...]: # WAV file
with wave.open(aboutme_noise_gate, 'r') as wav_file:
    frames_noise_gate = wav_file.readframes(-1)
    sample_noisegate_gated = np.frombuffer(frames_noise_gate, dtype='int16')
    fr_noisegate = wav_file.getframerate()
```

```
# Mendapatkan informasi channel  
channel_noise_gate = wav_file.getnchannels()
```

```
In [297...]  
print(f"Isi dari data audio: {sample_noise_gate}")  
print(f"Panjang data audio: {len(sample_noise_gate)}")  
print(f"Sampling rate audio: {fr_noise_gate}")  
print(f"Jumlah channel audio: {channel_noise_gate}")
```

```
Isi dari data audio: [0 0 0 ... 0 0 0]  
Panjang data audio: 1082880  
Sampling rate audio: 22050  
Jumlah channel audio: 1
```

```
In [298...]  
# Tampilkan waveform audio noise gate  
time_axis_noise_gate = np.linspace(0, len(sample_noise_gate) / fr_noise_gate, len(s  
  
plt.figure(figsize=(18, 5))  
plt.plot(time_axis_noise_gate, sample_noise_gate)  
plt.title('Waveform Audio Noise Gate')  
plt.xlabel('Waktu (detik)')  
plt.show()
```

