gcrgvutt6

October 4, 2024

1 Hands On 1

1.1 Topics

Audio Processing

1.2 Personal Identity

Nama Mahasiswa	Nomor Induk Mahasiswa
Kevin Simorangkir	121140150

1.3 Instructions

1. Buktikan 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 spectogram untuk memvisualisasikan ketiga sumber audio tersebut.
- Berikan analisa anda terhadap perbedaan visualisasi audio tersebut.

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

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.
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).
-

1.4 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.

1.5 What's The Problem?

1. Buktikan 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 (esssssss). 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 spectogram untuk memvisualisasikan ketiga sumber audio tersebut.
- Berikan analisa anda terhadap perbedaan visualisasi audio tersebut.

1.6 Tahap Persiapan

1. Siapkan Library / Pustaka yang akan digunakan dalam penugasan ini. Dalam hal ini menggunakan 4 (empat) pustaka yaitu sebagai berikut `matplotlib` , `numpy` , `os`, dan `wave`.

[146]:
```
import numpy as np #pustaka numpy
import matplotlib.pyplot as plt #pustaka matplotlib
import os #pustaka os
import wave #pustaka wave
```

2. Mempersiapkan audio yang diminta sesuai `Instructions` . Dalam hal ini telah dipersiapkan untuk file *audio* nya di *path* `Technology-Multimedia/Audio`. Dikategorikan menjadi 5 yaitu Low, Medium, Hard, Brrbbrbrb, dan Esss.

1.7 Tahap Eksekusi

1. Mengambil data dari File Audio yang telah disediakan melalui folder `Audio`. Dengan kode program sebagai berikut :

[147]:
```
# Mengambil data audio
audio_files = ["Low.wav", "Medium.wav", "High.wav", "Brrbbrbrb.wav", "Esss.wav"]
audio_data_list = [os.path.join(os.getcwd(), "Audio", file) for file in␣
↪audio_files]

for audio_data in audio_data_list:
    if not os.path.exists(audio_data):  # Check if file exists
        # Print error message if file not found
        print(f"Data dari {audio_data} tidak ditemukan")
        exit()  # Exit if file not found
```

[148]:
```
with wave.open(audio_data, "r") as audio_wave: # Open audio file
    audio_frames = audio_wave.readframes(-1) # Read audio frames
    file_data = np.frombuffer(audio_frames, dtype="int16") # Convert frames to␣
↪numpy array
    sample_rate = audio_wave.getframerate() # Get sample rate
    channels = audio_wave.getnchannels() # Get number of channels
```

[149]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    print(f"File: {os.path.basename(audio_data)}")
    print(f"Sample rate: {sample_rate} Hz")  # Display sample rate
    print(f"Channels: {channels}")  # Display number of channels
    print(f"Total frames: {len(file_data)}")  # Display total frames
    print(f"Duration: {len(file_data) / sample_rate} seconds")  # Display␣
↪duration
    print("-" * 40)  # Separator for readability
```

```
File: Low.wav
Sample rate: 44100 Hz
Channels: 2
Total frames: 286720
Duration: 6.501587301587302 seconds
----------------------------------------
File: Medium.wav
Sample rate: 44100 Hz
Channels: 2
```

```
Total frames: 284672
Duration: 6.455147392290249 seconds
----------------------------------------
File: High.wav
Sample rate: 44100 Hz
Channels: 2
Total frames: 286720
Duration: 6.501587301587302 seconds
----------------------------------------
File: Brrbbrbrb.wav
Sample rate: 44100 Hz
Channels: 2
Total frames: 415744
Duration: 9.427301587301587 seconds
----------------------------------------
File: Esss.wav
Sample rate: 44100 Hz
Channels: 2
Total frames: 313344
Duration: 7.1053061224489795 seconds
----------------------------------------
```

2. Memisahkan data file audio menjadi 2 sama dengan kanan dan juga kiri.

[150]:
```python
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    file_data = file_data.reshape(-1, channels)  # Reshape array
    print(f"File: {os.path.basename(audio_data)}")  # Print file name
    print(f"Shape: {file_data.shape}")  # Print shape of array

    if channels == 1:
        audio_kiri = file_data[:, 0]  # Get left channel data
        audio_kanan = None  # No right channel
        print(f"Audio kiri: {len(audio_kiri)}")  # Print left channel data␣
↪length
        print("Audio kanan: Tidak ada (mono audio)")  # Print message for no␣
↪right channel
    else:
        audio_kiri = file_data[:, 0]  # Get left channel data
        audio_kanan = file_data[:, 1]  # Get right channel data
        print(f"Audio kiri: {len(audio_kiri)}")  # Print left channel data␣
↪length
```

```
        print(f"Audio kanan: {len(audio_kanan)}")  # Print right channel data␣
↪length

    print("-" * 40)  # Separator for readability
```

```
File: Low.wav
Shape: (143360, 2)
Audio kiri: 143360
Audio kanan: 143360
----------------------------------------
File: Medium.wav
Shape: (142336, 2)
Audio kiri: 142336
Audio kanan: 142336
----------------------------------------
File: High.wav
Shape: (143360, 2)
Audio kiri: 143360
Audio kanan: 143360
----------------------------------------
File: Brrbbrbrb.wav
Shape: (207872, 2)
Audio kiri: 207872
Audio kanan: 207872
----------------------------------------
File: Esss.wav
Shape: (156672, 2)
Audio kiri: 156672
Audio kanan: 156672
----------------------------------------
```

3. Membuat visualisasi dari audio yang ada

[151]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    file_data = file_data.reshape(-1, channels) if channels > 1 else file_data ␣
↪# Reshape array if more than 1 channel

    sample_audio = len(file_data)  # Get length of audio data
    detik_audio = np.arange(sample_audio) / sample_rate  # Get time in seconds
    sumbu_waktu = np.linspace(0, sample_audio / sample_rate, sample_audio)  #␣
↪Get time axis
```

```
    print(f"File: {os.path.basename(audio_data)}")
    print(f"Sample audio: {sample_audio}")  # Print sample audio length
    print(f"Detik audio: {detik_audio}")  # Print time in seconds
    print(f"Sumbu waktu: {sumbu_waktu}")  # Print time axis
    print(f"Shape: {file_data.shape}")  # Print shape of array
    print("-" * 40)  # Separator for readability
```

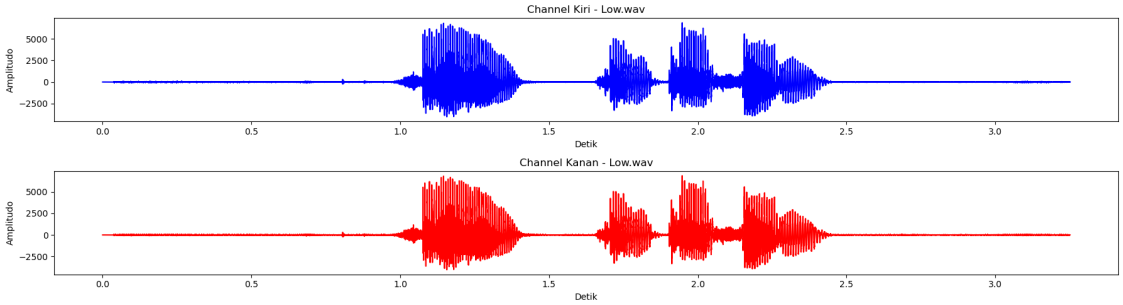
```
File: Low.wav
Sample audio: 143360
Detik audio: [0.00000000e+00 2.26757370e-05 4.53514739e-05 … 3.25072562e+00
 3.25074830e+00 3.25077098e+00]
Sumbu waktu: [0.00000000e+00 2.26758951e-05 4.53517903e-05 … 3.25074830e+00
 3.25077097e+00 3.25079365e+00]
Shape: (143360, 2)
----------------------------------------
File: Medium.wav
Sample audio: 142336
Detik audio: [0.00000000e+00 2.26757370e-05 4.53514739e-05 … 3.22750567e+00
 3.22752834e+00 3.22755102e+00]
Sumbu waktu: [0.00000000e+00 2.26758963e-05 4.53517925e-05 … 3.22752834e+00
 3.22755102e+00 3.22757370e+00]
Shape: (142336, 2)
----------------------------------------
File: High.wav
Sample audio: 143360
Detik audio: [0.00000000e+00 2.26757370e-05 4.53514739e-05 … 3.25072562e+00
 3.25074830e+00 3.25077098e+00]
Sumbu waktu: [0.00000000e+00 2.26758951e-05 4.53517903e-05 … 3.25074830e+00
 3.25077097e+00 3.25079365e+00]
Shape: (143360, 2)
----------------------------------------
File: Brrbbrbrb.wav
Sample audio: 207872
Detik audio: [0.00000000e+00 2.26757370e-05 4.53514739e-05 … 4.71358277e+00
 4.71360544e+00 4.71362812e+00]
Sumbu waktu: [0.00000000e+00 2.26758460e-05 4.53516921e-05 … 4.71360544e+00
 4.71362812e+00 4.71365079e+00]
Shape: (207872, 2)
----------------------------------------
File: Esss.wav
Sample audio: 156672
Detik audio: [0.00000000e+00 2.26757370e-05 4.53514739e-05 … 3.55258503e+00
 3.55260771e+00 3.55263039e+00]
Sumbu waktu: [0.00000000e+00 2.26758817e-05 4.53517634e-05 … 3.55260771e+00
 3.55263039e+00 3.55265306e+00]
Shape: (156672, 2)
```

---

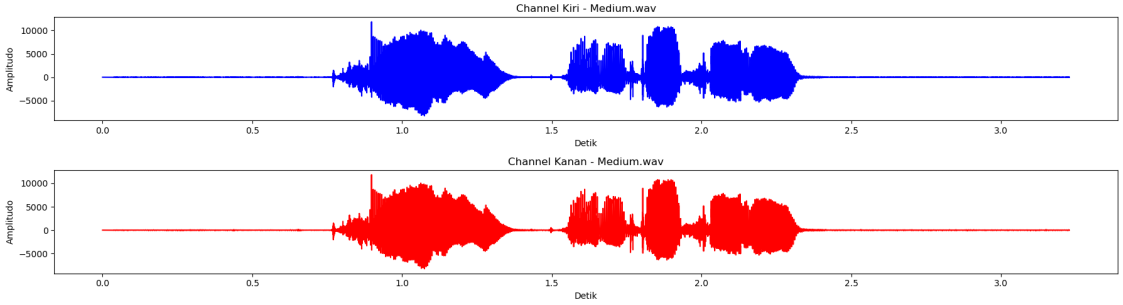
[152]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    if channels > 1:
        file_data = file_data.reshape(-1, channels)  # Reshape array if more␣
↪than 1 channel

    sample_audio = len(file_data)  # Get length of audio data
    sumbu_waktu = np.linspace(0, sample_audio / sample_rate, sample_audio)  #␣
↪Get time axis

    print(f"File: {os.path.basename(audio_data)}")
    print(f"Sample audio: {sample_audio}")  # Print sample audio length
    print(f"Shape: {file_data.shape}")  # Print shape of array
    print("-" * 40)  # Separator for readability

    if channels > 1:
        fig, ax = plt.subplots(2, 1, figsize=(18, 5))  # Create plot
        ax[0].plot(sumbu_waktu, file_data[:, 0], color="blue", label="left")  #␣
↪Plot left channel
        ax[0].set_title(f"Channel Kiri - {os.path.basename(audio_data)}")  #␣
↪Title plot
        ax[0].set_xlabel("Detik")  # Label x
        ax[0].set_ylabel("Amplitudo")  # Label y

        ax[1].plot(sumbu_waktu, file_data[:, 1], color="red", label="right")  #␣
↪Plot right channel
        ax[1].set_title(f"Channel Kanan - {os.path.basename(audio_data)}")  #␣
↪Title plot
        ax[1].set_xlabel("Detik")  # Label x
        ax[1].set_ylabel("Amplitudo")  # Label y

        print(f"Audio kiri: {file_data[:, 0]}")  # Print left channel data
        print(f"Audio kanan: {file_data[:, 1]}")  # Print right channel data
    else:
        plt.figure(figsize=(18, 5))  # Create plot
        plt.plot(sumbu_waktu, file_data, color="blue", label="mono")  # Plot␣
↪mono channel
        plt.title(f"Channel Mono - {os.path.basename(audio_data)}")  # Title␣
↪plot
```

```
        plt.xlabel("Detik")  # Label x
        plt.ylabel("Amplitudo")  # Label y

        print(f"Audio mono: {file_data}")  # Print mono channel data

    plt.tight_layout()  # Plot rapi
    plt.show()  # Menampilkan plot
```

```
File: Low.wav
Sample audio: 143360
Shape: (143360, 2)
----------------------------------------
Audio kiri: [0 0 0 … 2 3 4]
Audio kanan: [0 0 0 … 2 3 4]
```

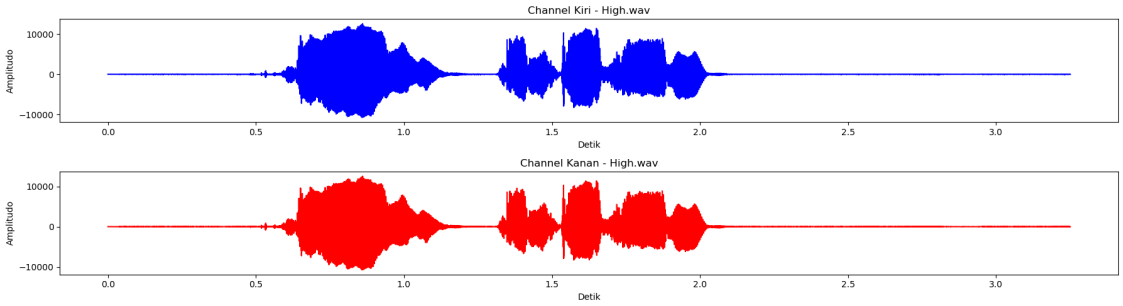



```
File: Medium.wav
Sample audio: 142336
Shape: (142336, 2)
----------------------------------------
Audio kiri: [ 0  0  0 … -6  3 -1]
Audio kanan: [ 0  0  0 … -6  3 -1]
```

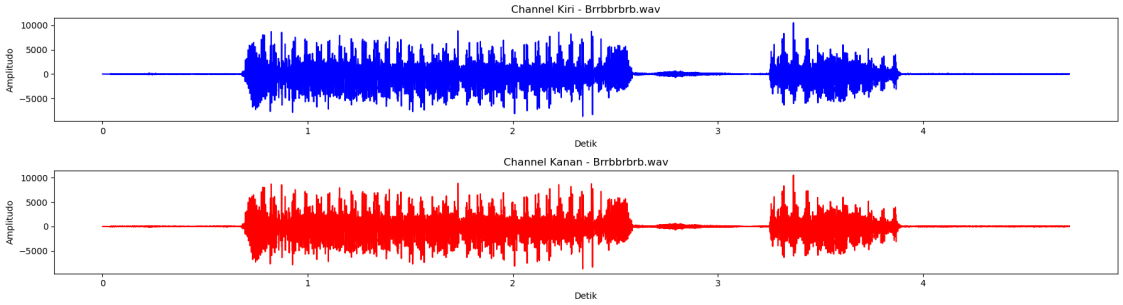



```
File: High.wav
Sample audio: 143360
Shape: (143360, 2)
```

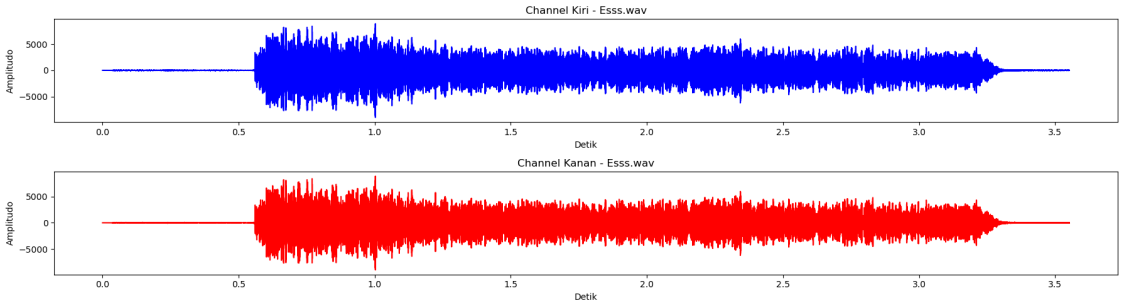
```
----------------------------------------
Audio kiri: [  0   0   0 … -34 -35 -34]
Audio kanan: [  0   0   0 … -34 -35 -34]
```




```
File: Brrbbrbrb.wav
Sample audio: 207872
Shape: (207872, 2)
----------------------------------------
Audio kiri: [ 0  0  0 … 17 20  6]
Audio kanan: [ 0  0  0 … 17 20  6]
```




```
File: Esss.wav
Sample audio: 156672
Shape: (156672, 2)
----------------------------------------
Audio kiri: [  0   0   0 …  -4  -1 -11]
Audio kanan: [  0   0   0 …  -4  -1 -11]
```

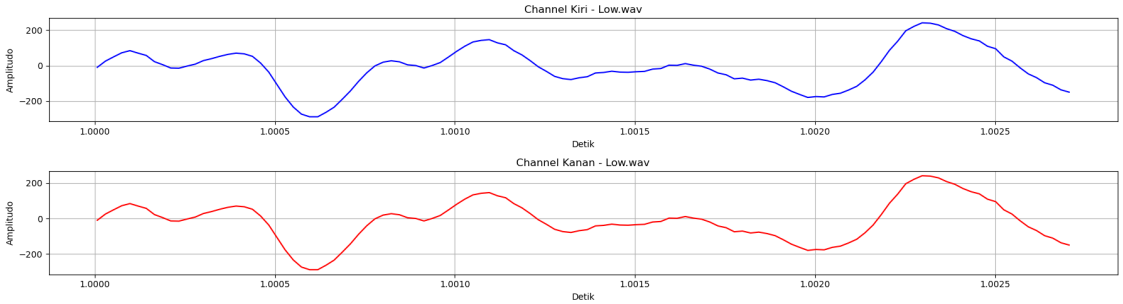


4. Membuat visualisasi dalaman dari pada file audio tersebut. Masih sama dengan menggunakan audio.wav

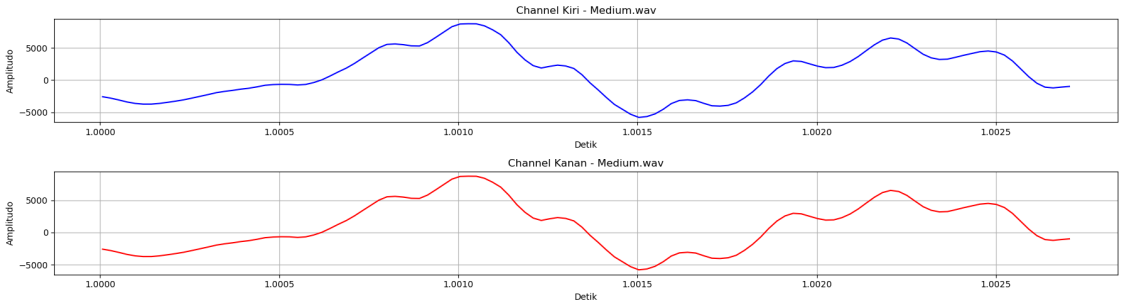
[153]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    file_data = file_data.reshape(-1, channels) if channels > 1 else file_data ␣
↪# Reshape array if more than 1 channel

    sample_audio = len(file_data)  # Get length of audio data
    sumbu_waktu = np.linspace(0, sample_audio / sample_rate, sample_audio)  #␣
↪Get time axis

    audio_kiri = file_data[:, 0] if channels > 1 else file_data  # Get left␣
↪channel data
    audio_kanan = file_data[:, 1] if channels > 1 else None  # Get right␣
↪channel data if available

    print(f"File: {os.path.basename(audio_data)}")
    ax, fig = plt.subplots(2, 1, figsize=(18, 5))  # Create plot
    fig[0].plot(sumbu_waktu[44100:44220], audio_kiri[44100:44220],␣
↪color="blue", label="left")  # Plot left channel
    fig[0].set_title(f"Channel Kiri - {os.path.basename(audio_data)}")  # Title␣
↪plot
    fig[0].set_xlabel("Detik")  # Label x
    fig[0].set_ylabel("Amplitudo")  # Label y
    fig[0].grid()  # Add grid

    if audio_kanan is not None:
```

```
        fig[1].plot(sumbu_waktu[44100:44220], audio_kanan[44100:44220],␣
↪color="red", label="right")  # Plot right channel
        fig[1].set_title(f"Channel Kanan - {os.path.basename(audio_data)}")  #␣
↪Title plot
        fig[1].set_xlabel("Detik")  # Label x
        fig[1].set_ylabel("Amplitudo")  # Label y
        fig[1].grid()  # Add grid

    plt.tight_layout()  # Plot rapi
    plt.show()  # Menampilkan plot
```

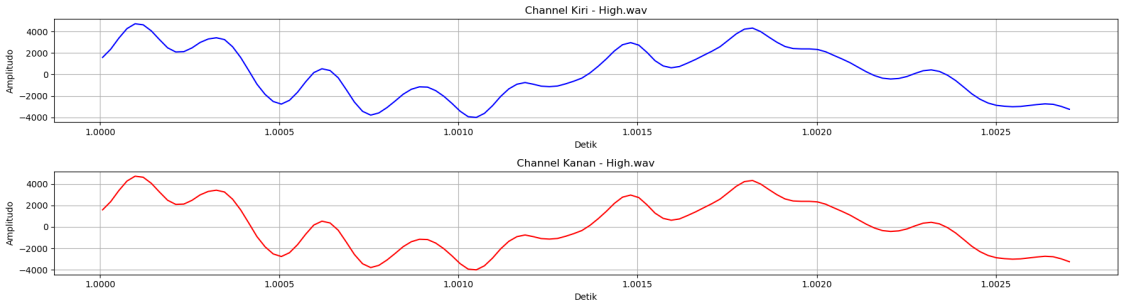
File: Low.wav



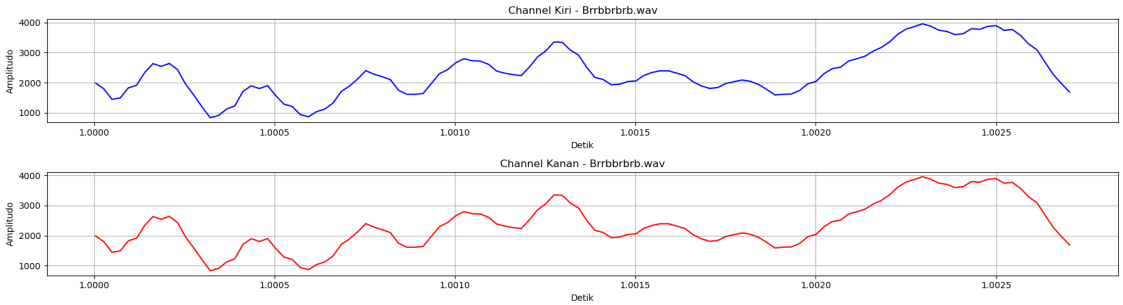
File: Medium.wav



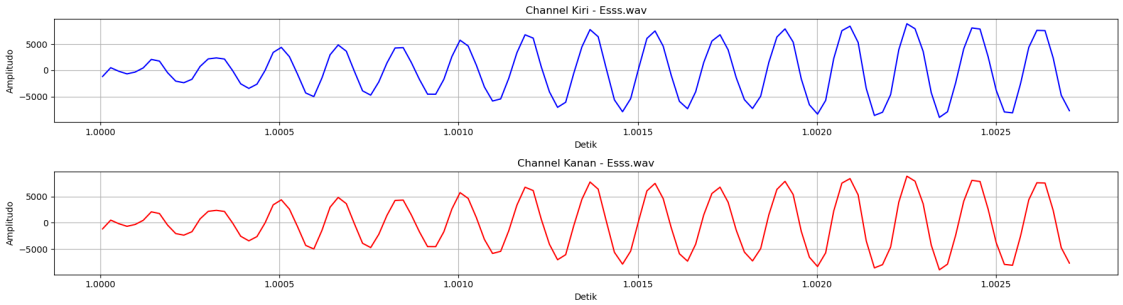
File: High.wav



File: Brrbbrbrb.wav



File: Esss.wav



[154]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels
```

```
    file_data = file_data.reshape(-1, channels) if channels > 1 else file_data  # Reshape array if more than 1 channel

    audio_kiri = file_data[:, 0] if channels > 1 else file_data  # Get left channel data
    audio_kanan = file_data[:, 1] if channels > 1 else None  # Get right channel data if available

    print(f"File: {os.path.basename(audio_data)}")
    print(f"Data dari audio kiri adalah : {audio_kiri[44100:44220]}")  # Print left channel data
    if audio_kanan is not None:
        print(f"Data dari audio kanan adalah : {audio_kanan[44100:44220]}")  # Print right channel data
    else:
        print("Audio kanan: Tidak ada (mono audio)")  # Print message for no right channel
    print("-" * 40)  # Separator for readability
```

```
File: Low.wav
Data dari audio kiri adalah : [ -10   25   49   72   84   70   57   22    5  -14
-15   -3    8   28
   39   52   63   70   66   52   14  -37 -107 -177 -234 -274 -289 -289
 -264 -235 -190 -143  -89  -41   -2   19   27   21    4    0  -14    0
   17   48   80  109  133  142  146  128  117   84   60   28   -7  -33
  -61  -74  -79  -69  -63  -42  -39  -32  -37  -38  -35  -33  -20  -17
    2    1   11    2   -4  -20  -42  -52  -75  -71  -82  -77  -85  -97
 -120 -145 -163 -180 -175 -177 -163 -156 -138 -117  -81  -37   22   86
  137  196  221  241  239  229  208  193  169  151  139  109   95   49
   25  -13  -47  -68  -97 -111 -137 -150]
Data dari audio kanan adalah : [ -10   25   49   72   84   70   57   22    5
-14  -15   -3    8   28
   39   52   63   70   66   52   14  -37 -107 -177 -234 -274 -289 -289
 -264 -235 -190 -143  -89  -41   -2   19   27   21    4    0  -14    0
   17   48   80  109  133  142  146  128  117   84   60   28   -7  -33
  -61  -74  -79  -69  -63  -42  -39  -32  -37  -38  -35  -33  -20  -17
    2    1   11    2   -4  -20  -42  -52  -75  -71  -82  -77  -85  -97
 -120 -145 -163 -180 -175 -177 -163 -156 -138 -117  -81  -37   22   86
  137  196  221  241  239  229  208  193  169  151  139  109   95   49
   25  -13  -47  -68  -97 -111 -137 -150]
----------------------------------------
File: Medium.wav
Data dari audio kiri adalah : [-2579 -2799 -3083 -3391 -3614 -3725 -3723 -3615
-3447 -3264 -3056 -2791
 -2518 -2242 -1952 -1755 -1597 -1410 -1263 -1052  -802  -697  -658  -671
  -757  -675  -379    40   632  1264  1854  2577  3396  4202  5009  5531
  5601  5490  5303  5280  5804  6608  7441  8271  8689  8728  8717  8403
```

```
  7796  7034  5788  4304  3125  2240  1869  2121  2314  2185  1802   835
  -433 -1526 -2690 -3775 -4538 -5300 -5782 -5651 -5239 -4532 -3642 -3156
 -3062 -3180 -3602 -3980 -4031 -3912 -3526 -2770 -1841  -702   629  1776
  2568  2975  2893  2523  2161  1923  1955  2299  2862  3638  4562  5465
  6191  6525  6353  5751  4866  4001  3442  3195  3244  3533  3844  4124
  4397  4508  4356  3875  2961  1745   522  -489 -1095 -1217 -1088  -982]
Data dari audio kanan adalah : [-2579 -2799 -3083 -3391 -3614 -3725 -3723 -3615
-3447 -3264 -3056 -2791
 -2518 -2242 -1952 -1755 -1597 -1410 -1263 -1052  -802  -697  -658  -671
  -757  -675  -379    40   632  1264  1854  2577  3396  4202  5009  5531
  5601  5490  5303  5280  5804  6608  7441  8271  8689  8728  8717  8403
  7796  7034  5788  4304  3125  2240  1869  2121  2314  2185  1802   835
  -433 -1526 -2690 -3775 -4538 -5300 -5782 -5651 -5239 -4532 -3642 -3156
 -3062 -3180 -3602 -3980 -4031 -3912 -3526 -2770 -1841  -702   629  1776
  2568  2975  2893  2523  2161  1923  1955  2299  2862  3638  4562  5465
  6191  6525  6353  5751  4866  4001  3442  3195  3244  3533  3844  4124
  4397  4508  4356  3875  2961  1745   522  -489 -1095 -1217 -1088  -982]
---------------------------------------
File: High.wav
Data dari audio kiri adalah : [ 1589  2354  3383  4270  4725  4625  4060  3251
2484  2097  2125  2473
  2980  3303  3418  3252  2583  1578   350  -905 -1856 -2520 -2762 -2410
 -1654  -682   174   531   365  -309 -1420 -2577 -3436 -3790 -3587 -3091
 -2482 -1842 -1382 -1154 -1181 -1507 -2033 -2696 -3421 -3937 -4002 -3626
 -2909 -2044 -1347  -911  -754  -908 -1091 -1136 -1078  -874  -628  -339
   147   761  1444  2193  2771  2966  2732  2063  1282   794   621   739
  1058  1402  1787  2165  2586  3180  3783  4221  4323  3994  3486  3000
  2604  2413  2381  2380  2327  2115  1778  1450  1097   671   252  -104
  -346  -425  -372  -194    94   345   427   283   -71  -567 -1181 -1811
 -2309 -2666 -2874 -2961 -3007 -2980 -2894 -2809 -2740 -2782 -2978 -3246]
Data dari audio kanan adalah : [ 1589  2354  3383  4270  4725  4625  4060  3251
2484  2097  2125  2473
  2980  3303  3418  3252  2583  1578   350  -905 -1856 -2520 -2762 -2410
 -1654  -682   174   531   365  -309 -1420 -2577 -3436 -3790 -3587 -3091
 -2482 -1842 -1382 -1154 -1181 -1507 -2033 -2696 -3421 -3937 -4002 -3626
 -2909 -2044 -1347  -911  -754  -908 -1091 -1136 -1078  -874  -628  -339
   147   761  1444  2193  2771  2966  2732  2063  1282   794   621   739
  1058  1402  1787  2165  2586  3180  3783  4221  4323  3994  3486  3000
  2604  2413  2381  2380  2327  2115  1778  1450  1097   671   252  -104
  -346  -425  -372  -194    94   345   427   283   -71  -567 -1181 -1811
 -2309 -2666 -2874 -2961 -3007 -2980 -2894 -2809 -2740 -2782 -2978 -3246]
---------------------------------------
File: Brrbbrbrb.wav
Data dari audio kiri adalah : [1986 1794 1441 1492 1826 1914 2341 2632 2542 2637
2430 1944 1582 1192
  832  907 1121 1230 1709 1894 1804 1901 1572 1284 1210  938  866 1036
 1122 1320 1706 1884 2120 2399 2279 2197 2098 1747 1614 1612 1636 1965
 2297 2433 2660 2796 2728 2715 2607 2384 2313 2265 2232 2523 2856 3056
```

```
 3352 3339 3081 2914 2504 2173 2104 1928 1946 2036 2056 2242 2339 2393
 2391 2314 2232 2027 1891 1808 1835 1971 2031 2087 2042 1939 1771 1588
 1614 1621 1738 1958 2039 2299 2465 2512 2718 2793 2879 3047 3170 3352
 3607 3776 3861 3954 3880 3744 3699 3594 3627 3791 3773 3867 3895 3738
 3765 3569 3283 3093 2676 2271 1966 1686]
Data dari audio kanan adalah : [1986 1794 1441 1492 1826 1914 2341 2632 2542
2637 2430 1944 1582 1192
  832  907 1121 1230 1709 1894 1804 1901 1572 1284 1210  938  866 1036
 1122 1320 1706 1884 2120 2399 2279 2197 2098 1747 1614 1612 1636 1965
 2297 2433 2660 2796 2728 2715 2607 2384 2313 2265 2232 2523 2856 3056
 3352 3339 3081 2914 2504 2173 2104 1928 1946 2036 2056 2242 2339 2393
 2391 2314 2232 2027 1891 1808 1835 1971 2031 2087 2042 1939 1771 1588
 1614 1621 1738 1958 2039 2299 2465 2512 2718 2793 2879 3047 3170 3352
 3607 3776 3861 3954 3880 3744 3699 3594 3627 3791 3773 3867 3895 3738
 3765 3569 3283 3093 2676 2271 1966 1686]
----------------------------------------
File: Esss.wav
Data dari audio kiri adalah : [-1151   508  -174  -667  -325   467  2079  1756
-441 -2044 -2348 -1690
   741  2174  2364  2150   -36 -2554 -3426 -2619   -17  3406  4364  2588
  -744 -4289 -4977 -1442  2957  4821  3631  -202 -3889 -4706 -2183  1402
  4244  4312  1498 -1716 -4518 -4529 -1723  2638  5732  4620  1007 -3163
 -5821 -5418 -1485  3357  6753  6124   654 -4071 -7018 -6055  -529  4464
  7745  6386   314 -5603 -7849 -5370   481  6060  7481  4601  -980 -5882
 -7283 -4065  1524  5560  6760  3884 -1429 -5563 -7240 -4909  1479  6332
  7871  5369 -1624 -6533 -8303 -5706  2275  7532  8387  5292 -3461 -8560
 -7950 -4597  3897  8849  7901  3615 -4245 -8929 -7853 -2371  4104  8054
  7842  2580 -3805 -7895 -8066 -2457  4345  7597  7548  2394 -4726 -7651]
Data dari audio kanan adalah : [-1151   508  -174  -667  -325   467  2079  1756
-441 -2044 -2348 -1690
   741  2174  2364  2150   -36 -2554 -3426 -2619   -17  3406  4364  2588
  -744 -4289 -4977 -1442  2957  4821  3631  -202 -3889 -4706 -2183  1402
  4244  4312  1498 -1716 -4518 -4529 -1723  2638  5732  4620  1007 -3163
 -5821 -5418 -1485  3357  6753  6124   654 -4071 -7018 -6055  -529  4464
  7745  6386   314 -5603 -7849 -5370   481  6060  7481  4601  -980 -5882
 -7283 -4065  1524  5560  6760  3884 -1429 -5563 -7240 -4909  1479  6332
  7871  5369 -1624 -6533 -8303 -5706  2275  7532  8387  5292 -3461 -8560
 -7950 -4597  3897  8849  7901  3615 -4245 -8929 -7853 -2371  4104  8054
  7842  2580 -3805 -7895 -8066 -2457  4345  7597  7548  2394 -4726 -7651]
----------------------------------------
```

5. Membuat visualiasi spectogram dari audio yang telah disediakan. Audio dapat dicek pada path `Audio`

[155]:
```
import librosa  # pustaka librosa
import librosa.display  # pustaka librosa.display
```

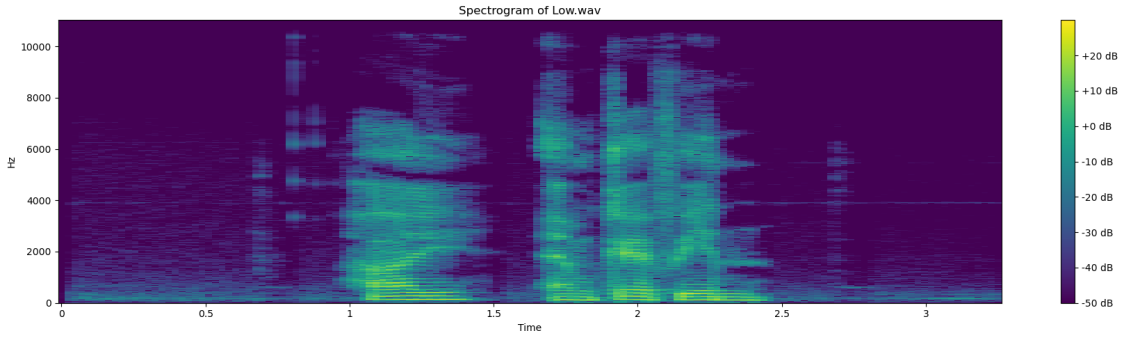
[156]:
```
audio_data = audio_data_list[3]  # mengambil data audio dari file Brrbbrbrb.wav
```

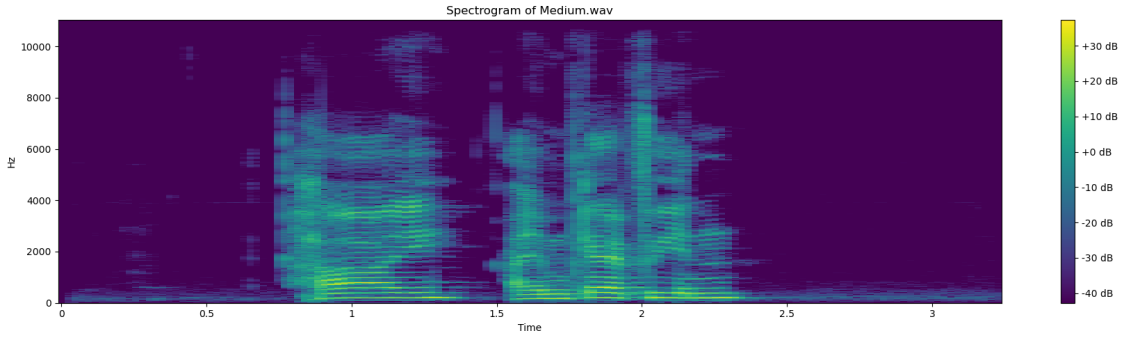
[157]:
```
for audio_data in audio_data_list:
    audioS, sample_rate = librosa.load(audio_data)  # load data audio
    SA = librosa.stft(audioS)  # short-time Fourier transform
    SDB = librosa.amplitude_to_db(abs(SA))  # amplitude to decibel

    print(f"Audio file: {os.path.basename(audio_data)}")
    print(f"Sample rate: {sample_rate}")
    print(f"STFT shape: {SA.shape}")
    print(f"SDB shape: {SDB.shape}")
    print("-" * 40)  # Separator for readability
```

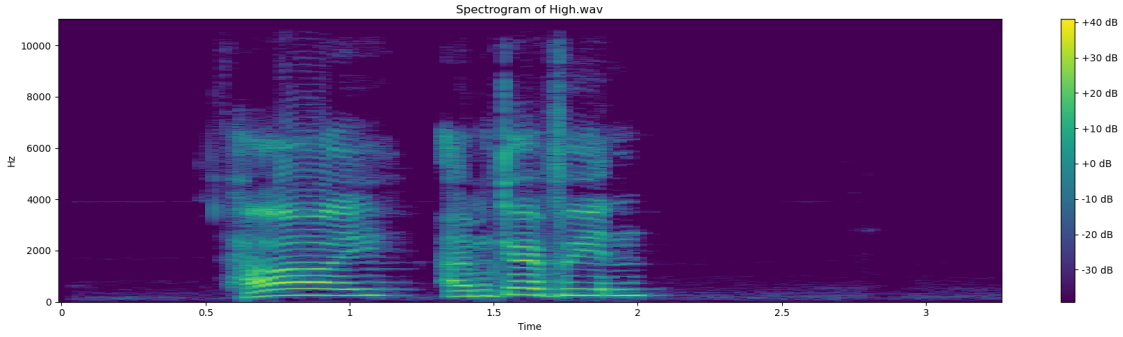
```
Audio file: Low.wav
Sample rate: 22050
STFT shape: (1025, 141)
SDB shape: (1025, 141)
----------------------------------------
Audio file: Medium.wav
Sample rate: 22050
STFT shape: (1025, 140)
SDB shape: (1025, 140)
----------------------------------------
Audio file: High.wav
Sample rate: 22050
STFT shape: (1025, 141)
SDB shape: (1025, 141)
----------------------------------------
Audio file: Brrbbrbrb.wav
Sample rate: 22050
STFT shape: (1025, 204)
SDB shape: (1025, 204)
----------------------------------------
Audio file: Esss.wav
Sample rate: 22050
STFT shape: (1025, 154)
SDB shape: (1025, 154)
----------------------------------------
```

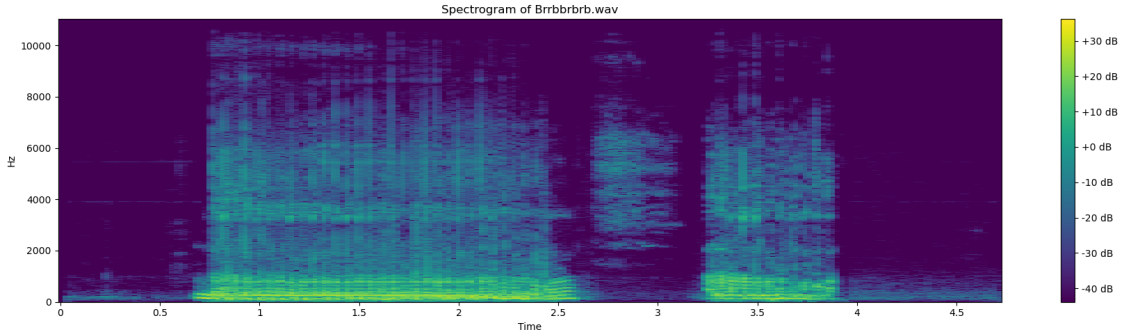
[158]:
```
for audio_data in audio_data_list:
    # Load audio file
    audioS, sample_rate = librosa.load(audio_data)

    # Compute the short-time Fourier transform (STFT)
    SA = librosa.stft(audioS)

    # Convert the amplitude to decibels
    SDB = librosa.amplitude_to_db(abs(SA))
```

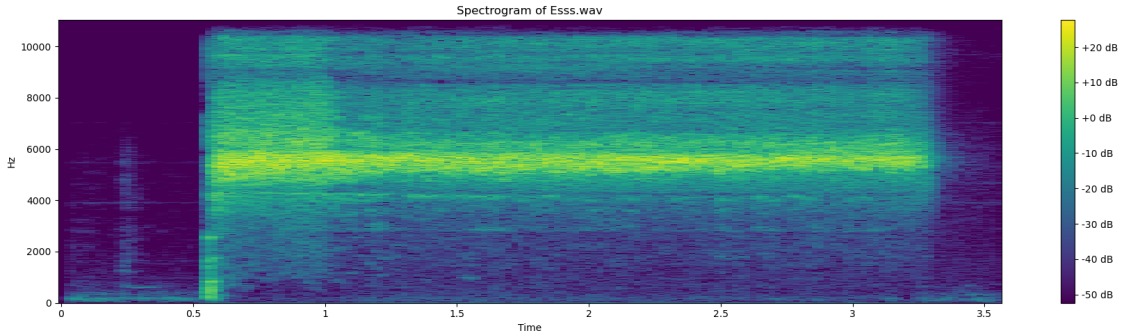
```
    # Plot the spectrogram
    plt.figure(figsize=(18, 5))
    librosa.display.specshow(SDB, sr=sample_rate, x_axis="time", y_axis="hz",␣
↪cmap="viridis")
    plt.colorbar(format="%+2.0f dB")
    plt.title(f"Spectrogram of {os.path.basename(audio_data)}")
    plt.tight_layout()
    plt.show()
```











6. Memotong audio dari data yang sudah ada menggunakan `wave`

[159]:
```
with wave.open(audio_data, "r") as audio_wave:  # membuka file audio
    audio_frames = audio_wave.readframes(-1)  # membaca frame audio
    file_data = np.frombuffer(audio_frames, dtype="int16")  # mengubah frame ke␣
 ↪array numpy
    sample_rate = audio_wave.getframerate()  # mengambil sample rate
    channels = audio_wave.getnchannels()  # mengambil jumlah channel

file_data = file_data.reshape(-1, channels)  # mengubah bentuk array
print(f"Shape dari file data yang baru: {file_data.shape}")  # menampilkan␣
 ↪bentuk array
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # membuka file audio
        audio_frames = audio_wave.readframes(-1)  # membaca frame audio
        file_data = np.frombuffer(audio_frames, dtype="int16")  # mengubah␣
 ↪frame ke array numpy
        sample_rate = audio_wave.getframerate()  # mengambil sample rate
        channels = audio_wave.getnchannels()  # mengambil jumlah channel

    file_data = file_data.reshape(-1, channels)  # mengubah bentuk array
    print(f"File: {os.path.basename(audio_data)}")
```

```
    print(f"Shape dari file data yang baru: {file_data.shape}")  # menampilkan␣
↪bentuk array
    print("-" * 40)  # Separator for readability
```

```
Shape dari file data yang baru: (156672, 2)
File: Low.wav
Shape dari file data yang baru: (143360, 2)
----------------------------------------
File: Medium.wav
Shape dari file data yang baru: (142336, 2)
----------------------------------------
File: High.wav
Shape dari file data yang baru: (143360, 2)
----------------------------------------
File: Brrbbrbrb.wav
Shape dari file data yang baru: (207872, 2)
----------------------------------------
File: Esss.wav
Shape dari file data yang baru: (156672, 2)
----------------------------------------
```

[160]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    file_data = file_data.reshape(-1, channels) if channels > 1 else file_data ␣
↪# Reshape array if more than 1 channel

    print(f"File: {os.path.basename(audio_data)}")
    if channels > 1:
        print(f"Isi dari audio sebelah kiri: {file_data[:, 0]}")  # Print left␣
↪channel data
        print(f"Isi dari audio sebelah kanan: {file_data[:, 1]}")  # Print␣
↪right channel data
    else:
        print(f"Isi dari audio sebelah kiri: {file_data}")  # Print left␣
↪channel data
        print("Audio kanan: Tidak ada (mono audio)")  # Print message for no␣
↪right channel
    print("-" * 40)  # Separator for readability
```

```
File: Low.wav
Isi dari audio sebelah kiri: [0 0 0 … 2 3 4]
```

```
Isi dari audio sebelah kanan: [0 0 0 … 2 3 4]
----------------------------------------
File: Medium.wav
Isi dari audio sebelah kiri: [ 0  0  0 … -6  3 -1]
Isi dari audio sebelah kanan: [ 0  0  0 … -6  3 -1]
----------------------------------------
File: High.wav
Isi dari audio sebelah kiri: [  0   0   0 … -34 -35 -34]
Isi dari audio sebelah kanan: [  0   0   0 … -34 -35 -34]
----------------------------------------
File: Brrbbrbrb.wav
Isi dari audio sebelah kiri: [ 0  0  0 … 17 20  6]
Isi dari audio sebelah kanan: [ 0  0  0 … 17 20  6]
----------------------------------------
File: Esss.wav
Isi dari audio sebelah kiri: [  0   0   0 …  -4  -1 -11]
Isi dari audio sebelah kanan: [  0   0   0 …  -4  -1 -11]
----------------------------------------
```

[161]:
```python
detik_titikawal = 5  # detik titik awal
detik_titikakhir = 10  # detik titik akhir

titikawal = detik_titikawal * sample_rate  # titik awal
titikakhir = detik_titikakhir * sample_rate  # titik akhir

print(f"Titik awal: {titikawal}")  # menampilkan titik awal
print(f"Titik akhir: {titikakhir}")  # menampilkan titik akhir
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        sample_rate = audio_wave.getframerate()  # Get sample rate

    titikawal = detik_titikawal * sample_rate  # titik awal
    titikakhir = detik_titikakhir * sample_rate  # titik akhir

    print(f"File: {os.path.basename(audio_data)}")
    print(f"Titik awal: {titikawal}")  # menampilkan titik awal
    print(f"Titik akhir: {titikakhir}")  # menampilkan titik akhir
    print("-" * 40)  # Separator for readability
```

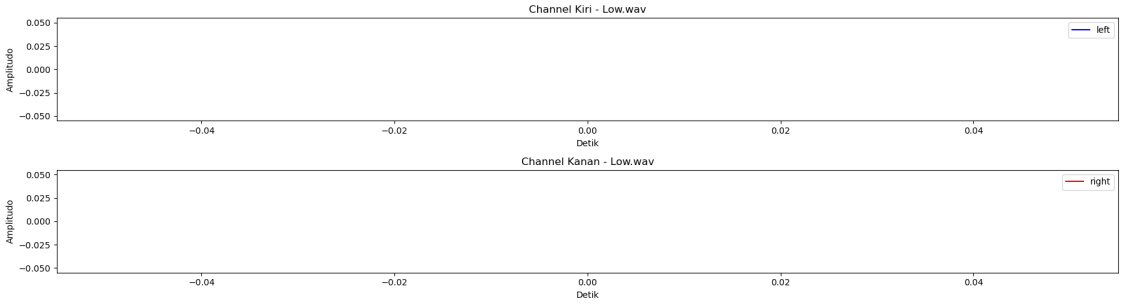
```
Titik awal: 220500
Titik akhir: 441000
File: Low.wav
Titik awal: 220500
Titik akhir: 441000
----------------------------------------
File: Medium.wav
Titik awal: 220500
Titik akhir: 441000
```

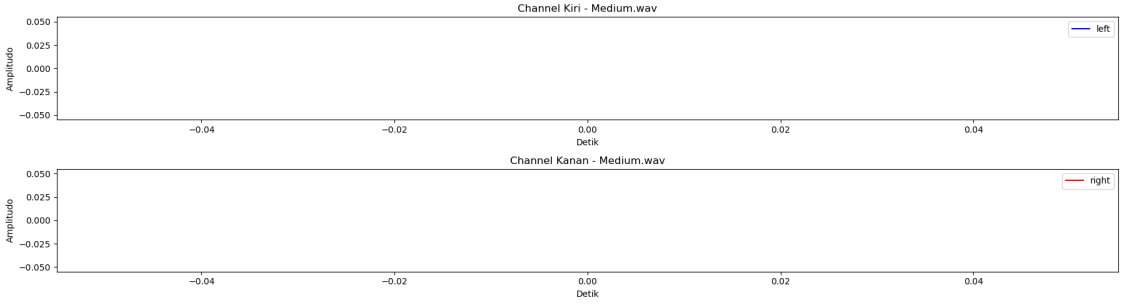
```
----------------------------------------
File: High.wav
Titik awal: 220500
Titik akhir: 441000
----------------------------------------
File: Brrbbrbrb.wav
Titik awal: 220500
Titik akhir: 441000
----------------------------------------
File: Esss.wav
Titik awal: 220500
Titik akhir: 441000
----------------------------------------
```

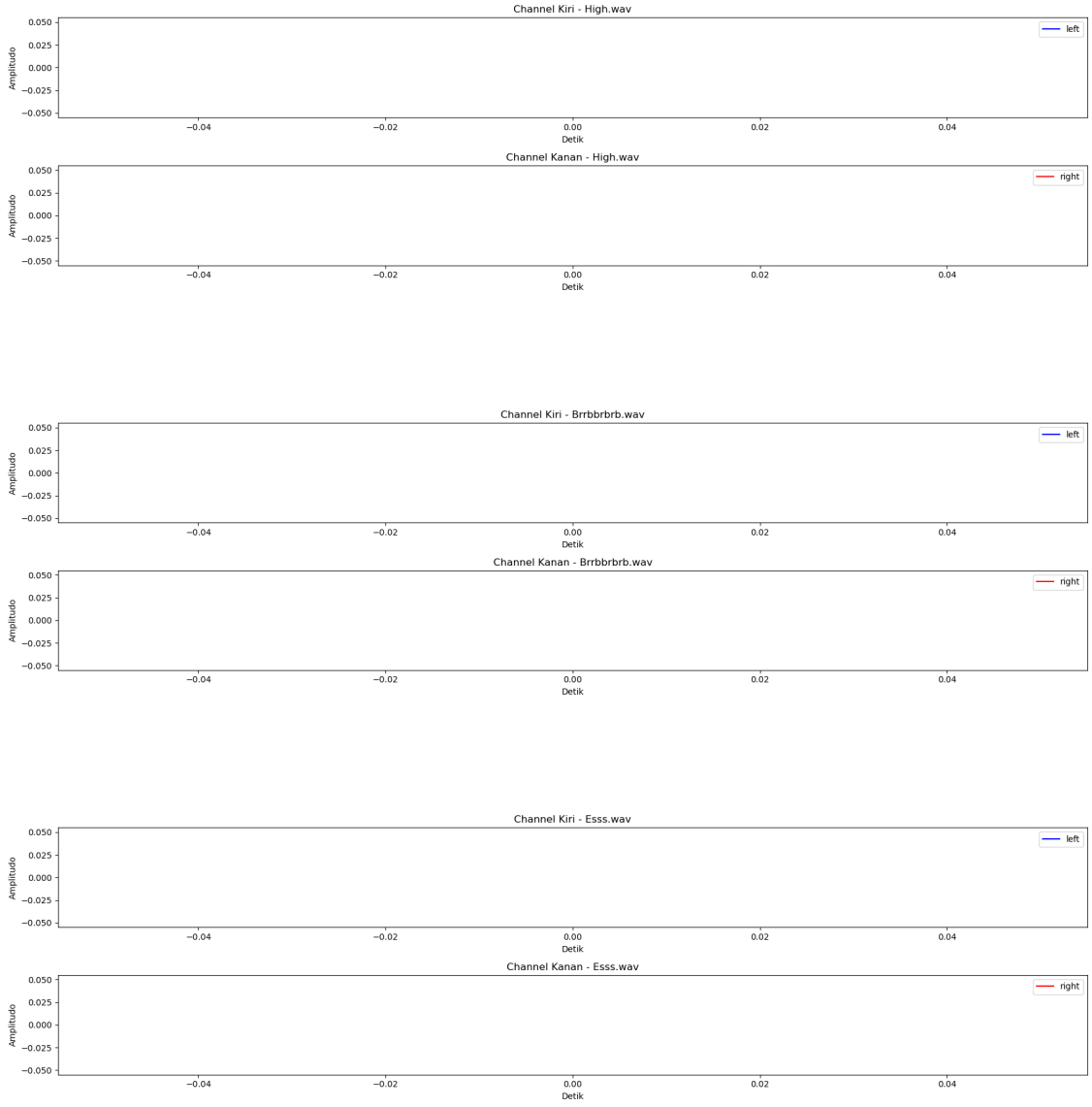
[162]:
```
trim_audio = file_data[titikawal:titikakhir]  # trim audio
sumbu_waktu_trim = sumbu_waktu[titikawal:titikakhir]  # sumbu waktu trim
```

[163]:
```
for audio_data in audio_data_list:
    with wave.open(audio_data, "r") as audio_wave:  # Open audio file
        audio_frames = audio_wave.readframes(-1)  # Read audio frames
        file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert␣
↪frames to numpy array
        sample_rate = audio_wave.getframerate()  # Get sample rate
        channels = audio_wave.getnchannels()  # Get number of channels

    file_data = file_data.reshape(-1, channels) if channels > 1 else file_data ␣
↪# Reshape array if more than 1 channel

    detik_titikawal = 5  # detik titik awal
    detik_titikakhir = 10  # detik titik akhir

    titikawal = detik_titikawal * sample_rate  # titik awal
    titikakhir = detik_titikakhir * sample_rate  # titik akhir

    trim_audio = file_data[titikawal:titikakhir]  # Trim audio
    sumbu_waktu_trim = np.linspace(detik_titikawal, detik_titikakhir,␣
↪len(trim_audio))  # Time axis for trimmed audio

    fig, ax = plt.subplots(2, 1, figsize=(18, 5))  # Create plot

    if channels > 1:  # Check if there is a right channel
        ax[0].plot(sumbu_waktu_trim, trim_audio[:, 0], color="blue",␣
↪label="left")  # Plot left channel
        ax[0].set_title(f"Channel Kiri - {os.path.basename(audio_data)}")  #␣
↪Title plot
        ax[0].set_xlabel("Detik")  # Label x
        ax[0].set_ylabel("Amplitudo")  # Label y
```

```
        ax[0].legend()  # Add legend

        ax[1].plot(sumbu_waktu_trim, trim_audio[:, 1], color="red",␣
↪label="right")  # Plot right channel
        ax[1].set_title(f"Channel Kanan - {os.path.basename(audio_data)}")  #␣
↪Title plot
        ax[1].set_xlabel("Detik")  # Label x
        ax[1].set_ylabel("Amplitudo")  # Label y
        ax[1].legend()  # Add legend
    else:
        ax[0].plot(sumbu_waktu_trim, trim_audio, color="blue", label="mono")  #␣
↪Plot mono channel
        ax[0].set_title(f"Channel Mono - {os.path.basename(audio_data)}")  #␣
↪Title plot
        ax[0].set_xlabel("Detik")  # Label x
        ax[0].set_ylabel("Amplitudo")  # Label y
        ax[0].legend()  # Add legend

    plt.tight_layout()  # Plot rapi
    plt.show()  # Menampilkan plot
```







1.8 Mari Kita Analisis (MaKiAn)

1. Pengambilan Data Audio Pada tahap ini, kita mengambil data audio dari beberapa file yang berada di folder `Audio`. Data audio ini kemudian dibaca menggunakan pustaka `wave` dan diubah menjadi array numpy untuk memudahkan proses selanjutnya.

2. Informasi Dasar Audio Setelah data audio berhasil diambil, kita menampilkan beberapa informasi dasar seperti sample rate, jumlah channel, total frame, dan durasi audio. Dari informasi ini, kita mengetahui bahwa audio memiliki sample rate sebesar 44100 Hz dan hanya memiliki satu channel (mono audio).

3. Pemisahan Data Audio Data audio yang telah diambil kemudian dipisahkan menjadi channel kiri dan kanan. Namun, karena audio ini hanya memiliki satu channel, maka hanya ada data untuk channel kiri.

4. Visualisasi Audio Visualisasi audio dilakukan dengan membuat plot dari data audio terhadap waktu. Pada visualisasi ini, kita dapat melihat bagaimana bentuk gelombang audio dari channel kiri.

5. Visualisasi Detail Audio Selain visualisasi umum, kita juga membuat visualisasi detail dari sebagian kecil data audio (dari detik ke 44100 hingga 44220). Visualisasi ini memberikan gambaran lebih detail mengenai bentuk gelombang audio pada interval waktu yang lebih kecil.

Kesimpulan Dari hasil visualisasi dan analisis di atas, kita dapat melihat bahwa: - Audio memiliki sample rate sebesar 44100 Hz dan hanya memiliki satu channel (mono audio). - Bentuk gelombang audio dapat divisualisasikan dengan baik menggunakan pustaka `matplotlib`. - Spektrogram memberikan informasi tambahan mengenai distribusi frekuensi dari audio. - Pemotongan audio dapat dilakukan dengan mudah menggunakan pustaka `numpy` dan `wave`.

Visualisasi yang telah dibuat memberikan gambaran yang jelas mengenai karakteristik dari audio yang direkam, baik dalam domain waktu maupun frekuensi.

1.9 What's The Problem?

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

1.9.1 Tahap Membuat Teknik Fading

1.9.2 Dalam Hal Ini Mahasiswa Ber-NIM akhir Genap (121140150)

1. Persiapkan library / pustaka yang menjadi kebutuhan dalam perkodingan ini. Dalam hal ini menginstall `numpy`, `librosa`, dan `soundfile`

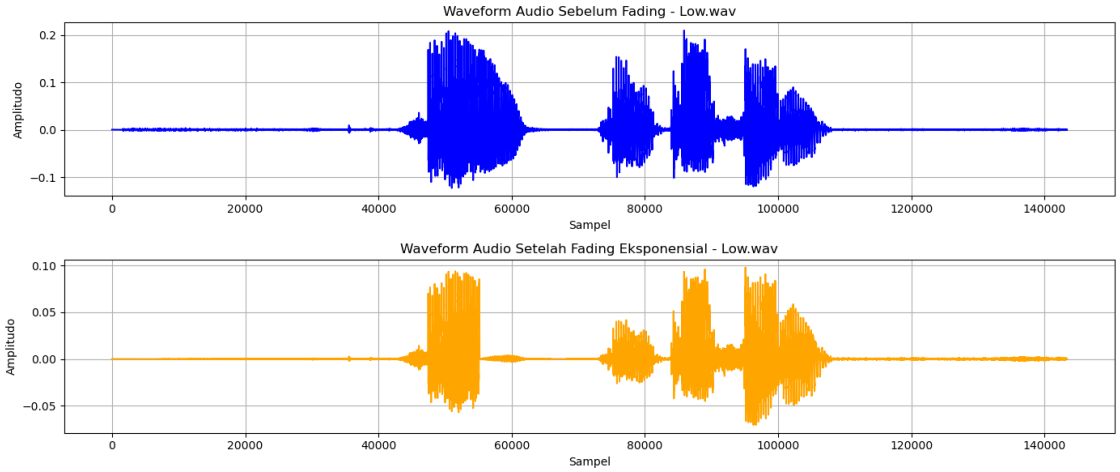
[164]:
```
import numpy as np #pustaka numpy
import librosa #pustaka librosa
import soundfile as sf #pustaka soundfile
import matplotlib.pyplot as plt #pustaka matplotlib
import os #pustaka os
```

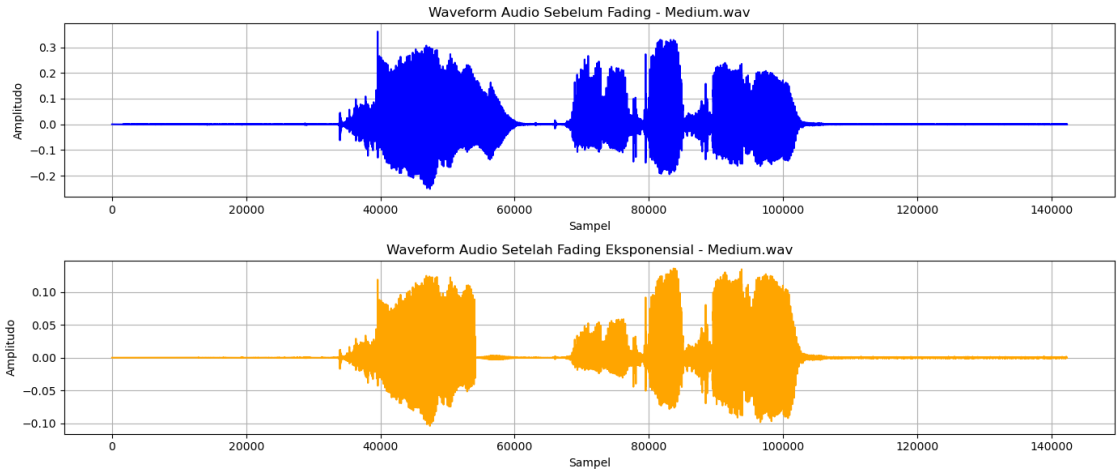
2. Definisikan fungsi untuk mengubah audio ke format baru

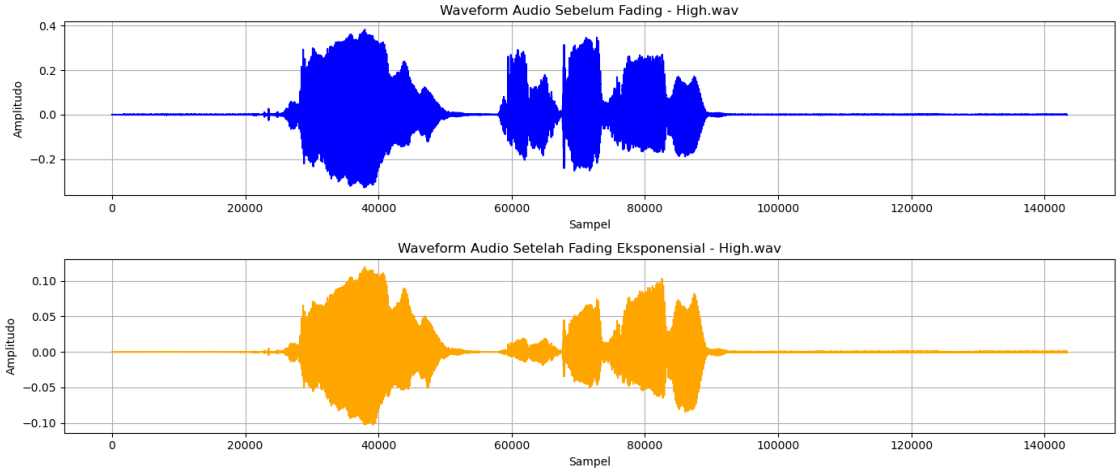
[165]:
```
def apply_exponential_fade(audio_kiri, sample_rate, fade_duration):
    num_samples = len(audio_kiri)                # Total number of audio samples
    fade_length = int(fade_duration * sample_rate)   # Number of samples in the␣
 ↪fade duration
```

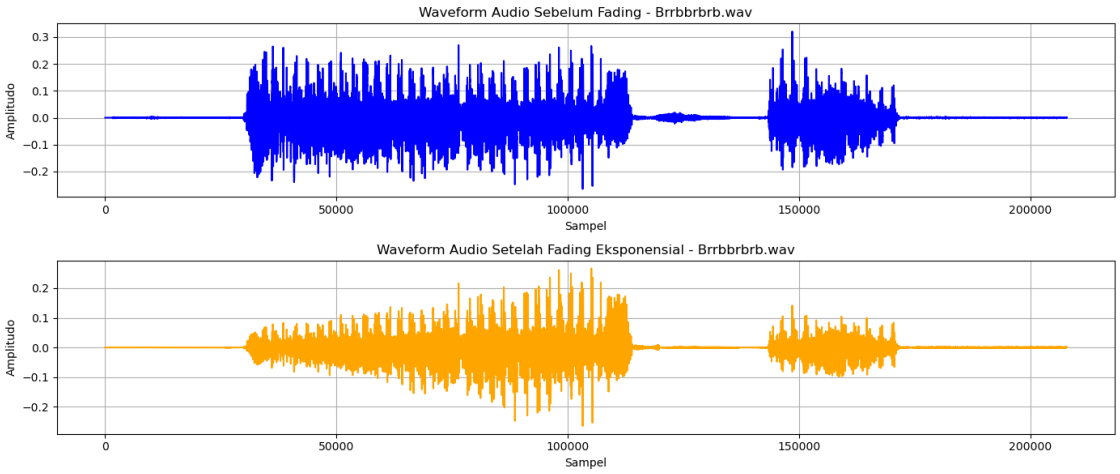
```
    # Create a time array from 0 to 1
    t = np.linspace(0, 1, fade_length)

    # Calculate the exponential fade-in and fade-out
    fade_in = np.exp(t) - 1                    # Fade-in curve
    fade_out = 1 - np.exp(-t)                  # Fade-out curve

    # Normalize the fades to range [0, 1]
    fade_in /= fade_in[-1]                     # Normalize to max 1
    fade_out /= fade_out[-1]                   # Normalize to max 1

    # Apply fade-in
    if fade_length < num_samples:
        audio_kiri[:fade_length] *= fade_in

    # Apply fade-out
    if fade_length < num_samples:
        audio_kiri[-fade_length:] *= fade_out

    return audio_kiri

fade_duration = 2.0  # Duration in seconds for the fade

for audio_data in audio_data_list:
    # Load the audio file
    audioS, sample_rate = librosa.load(audio_data, sr=None)  # Load audio file

    # Apply exponential fade
    faded_audio = apply_exponential_fade(audioS.copy(), sample_rate,␣
 ↪fade_duration)

    # Visualize the audio before and after fading
    plt.figure(figsize=(14, 6))

    # Plot original audio
    plt.subplot(2, 1, 1)
    plt.plot(audioS, color='blue')
    plt.title(f'Waveform Audio Sebelum Fading - {os.path.basename(audio_data)}')
    plt.xlabel('Sampel')
    plt.ylabel('Amplitudo')
    plt.grid()

    # Plot faded audio
    plt.subplot(2, 1, 2)
    plt.plot(faded_audio, color='orange')
    plt.title(f'Waveform Audio Setelah Fading Eksponensial - {os.path.
 ↪basename(audio_data)}')
```

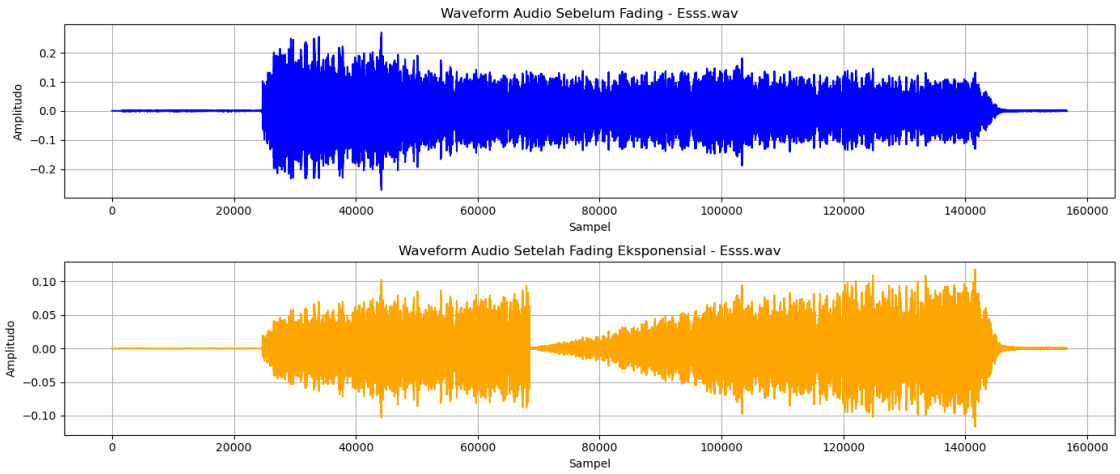
```
    plt.xlabel('Sampel')
    plt.ylabel('Amplitudo')
    plt.grid()

    plt.tight_layout()
    plt.show()
```



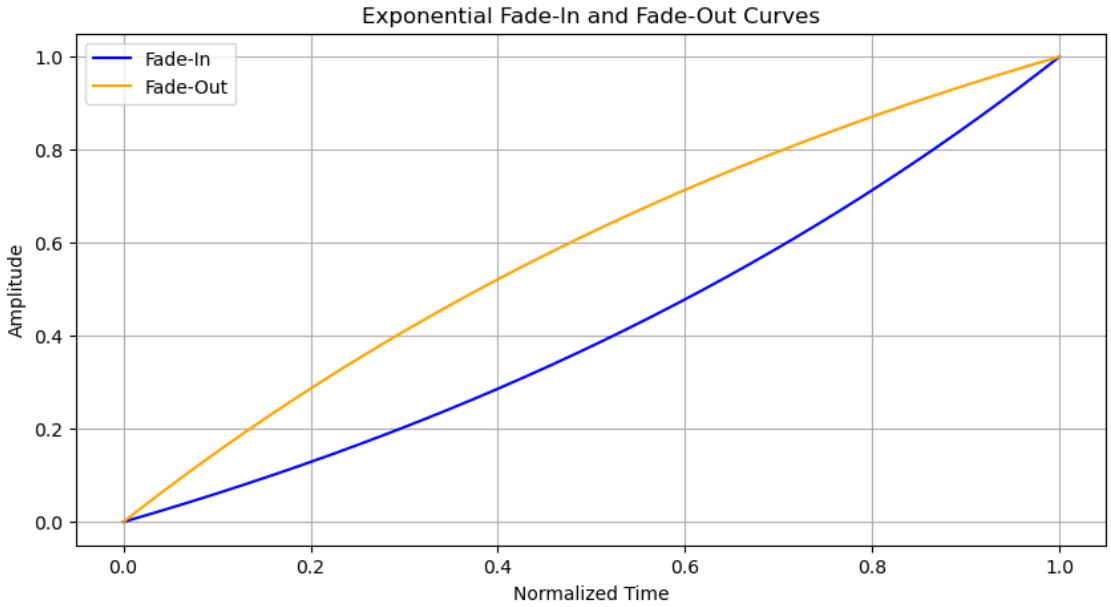



Waveform Audio Sebelum Fading - High.wav
Amplitudo
Sampel
Waveform Audio Setelah Fading Eksponensial - High.wav
Amplitudo
Sampel


Waveform Audio Sebelum Fading - Brrbbrbrb.wav
Amplitudo
Sampel
Waveform Audio Setelah Fading Eksponensial - Brrbbrbrb.wav
Amplitudo
Sampel


Waveform Audio Sebelum Fading - Esss.wav
Amplitudo
Sampel
Waveform Audio Setelah Fading Eksponensial - Esss.wav
Amplitudo
Sampel

[166]:
```
def plot_exponential_fade(fade_duration, sample_rate):
    fade_length = int(fade_duration * sample_rate)  # Number of samples in the␣
 ↪fade duration

    # Create a time array from 0 to 1
    t = np.linspace(0, 1, fade_length)

    # Calculate the exponential fade-in and fade-out
    fade_in = np.exp(t) - 1  # Fade-in curve
    fade_out = 1 - np.exp(-t)  # Fade-out curve

    # Normalize the fades to range [0, 1]
    fade_in /= fade_in[-1]  # Normalize to max 1
    fade_out /= fade_out[-1]  # Normalize to max 1

    # Plot the fade-in and fade-out curves
    plt.figure(figsize=(10, 5))
    plt.plot(t, fade_in, label="Fade-In", color="blue")
    plt.plot(t, fade_out, label="Fade-Out", color="orange")
    plt.title("Exponential Fade-In and Fade-Out Curves")
    plt.xlabel("Normalized Time")
    plt.ylabel("Amplitude")
    plt.legend()
    plt.grid()
    plt.show()

# Example usage
fade_duration = 2.0  # Duration in seconds for the fade
sample_rate = 44100  # Sample rate in Hz
plot_exponential_fade(fade_duration, sample_rate)
```



1.9.3 Mari Kita Analisis (MaKiAn)

Pada tahap ini, kita telah mengimplementasikan teknik fading eksponensial pada audio. Teknik ini melibatkan penerapan kurva eksponensial untuk mengubah amplitudo audio secara bertahap pada awal dan akhir audio. Berikut adalah analisis dari teknik fading yang telah diterapkan:

1. **Kurva Fading Eksponensial**:
   - Kurva fading eksponensial memiliki bentuk yang khas, di mana amplitudo meningkat secara eksponensial pada fade-in dan menurun secara eksponensial pada fade-out.
   - Kurva fade-in dimulai dari 0 dan meningkat hingga 1, sedangkan kurva fade-out dimulai dari 1 dan menurun hingga 0.
   - Kurva ini memberikan transisi yang lebih halus dibandingkan dengan linear fading, terutama pada bagian awal dan akhir audio.
2. **Implementasi Fading**:
   - Fading diterapkan pada audio dengan durasi fade yang ditentukan (misalnya 2 detik).
   - Pada bagian awal audio, amplitudo meningkat secara eksponensial dari 0 hingga nilai maksimum dalam durasi fade.
   - Pada bagian akhir audio, amplitudo menurun secara eksponensial dari nilai maksimum hingga 0 dalam durasi fade.
3. **Visualisasi Audio Sebelum dan Sesudah Fading**:
   - Visualisasi audio sebelum fading menunjukkan amplitudo asli dari audio.
   - Visualisasi audio setelah fading menunjukkan perubahan amplitudo pada bagian awal dan akhir audio sesuai dengan kurva eksponensial.
   - Perubahan ini terlihat jelas pada plot, di mana amplitudo meningkat secara bertahap pada awal dan menurun secara bertahap pada akhir.
4. **Keuntungan Teknik Fading Eksponensial**:
   - Teknik ini memberikan transisi yang lebih alami dan halus dibandingkan dengan linear fading.

- Mengurangi kemungkinan adanya klik atau noise pada awal dan akhir audio yang sering terjadi pada linear fading.
- Cocok untuk aplikasi yang membutuhkan transisi audio yang halus, seperti dalam produksi musik atau editing audio.

5. **Kesimpulan**:
   - Teknik fading eksponensial efektif dalam memberikan transisi yang halus pada audio.
   - Implementasi dan visualisasi menunjukkan bahwa teknik ini dapat diterapkan dengan mudah menggunakan pustaka `numpy` dan `librosa`.
   - Kurva eksponensial memberikan hasil yang lebih baik dibandingkan dengan linear fading dalam hal kehalusan transisi.

1.10 What's The Problem?

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.

1.10.1 Normalisasi Audio `kevins.wav`

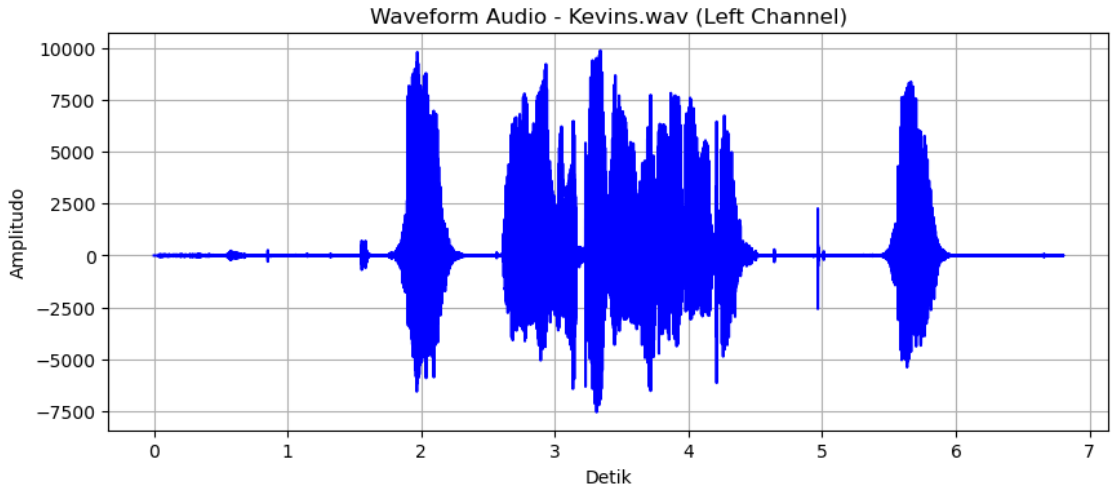
[167]:
```
audio_data3 = os.path.join(os.getcwd(), "Audio", "Kevins.wav")  # Load audio file

with wave.open(audio_data3, "r") as audio_wave:  # Open audio file
    audio_width = audio_wave.getsampwidth()  # Get sample width
    audio_sample = audio_wave.getnframes()  # Get number of frames
    audio_frames = audio_wave.readframes(audio_sample)  # Read audio frames
    file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert frames to numpy array
    sample_rate = audio_wave.getframerate()  # Get sample rate
    channels = audio_wave.getnchannels()  # Get number of channels
```

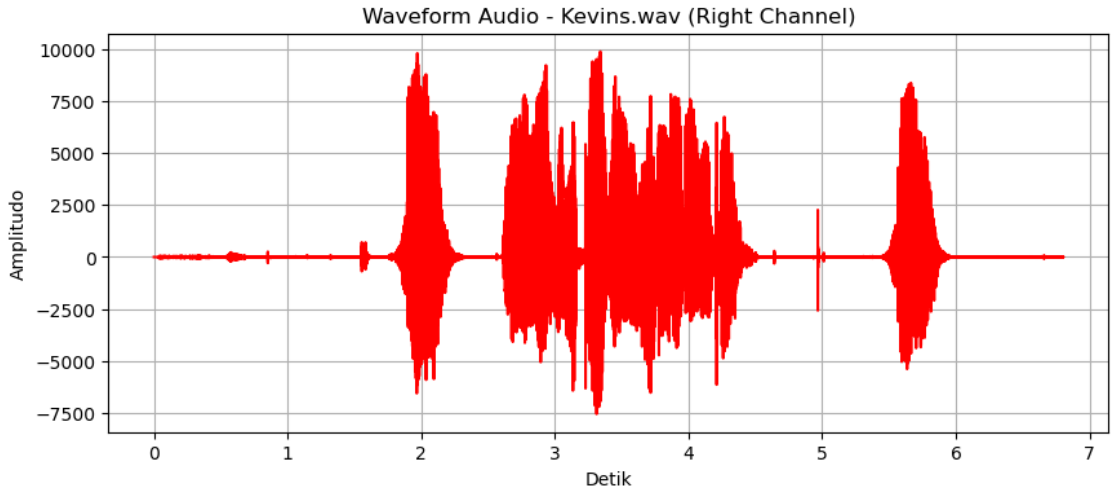
[168]:
```
print(f"File: {os.path.basename(audio_data3)}")
print(f"Sample rate: {sample_rate} Hz")  # Display sample rate
print(f"Long Audio: {audio_sample}")  # Display number of channels
```

```
File: Kevins.wav
Sample rate: 44100 Hz
Long Audio: 300032
```

[170]:
```
sumbu_waktu = np.linspace(0, audio_sample / sample_rate, audio_sample)  # Get time axis

# Check if the audio is stereo or mono
if channels > 1:
    # Reshape file_data to 2D array if it has multiple channels
    file_data = file_data.reshape(-1, channels)
```

```
    # Plot left channel
    plt.figure(figsize=(10, 4))
    plt.plot(sumbu_waktu[:file_data.shape[0]], file_data[:, 0], color="blue")
    plt.title(f"Waveform Audio - {os.path.basename(audio_data3)} (Left␣
 ↪Channel)")
    plt.xlabel("Detik")
    plt.ylabel("Amplitudo")
    plt.grid()
    plt.show()

    # Plot right channel
    plt.figure(figsize=(10, 4))
    plt.plot(sumbu_waktu[:file_data.shape[0]], file_data[:, 1], color="red")
    plt.title(f"Waveform Audio - {os.path.basename(audio_data3)} (Right␣
 ↪Channel)")
    plt.xlabel("Detik")
    plt.ylabel("Amplitudo")
    plt.grid()
    plt.show()
else:
    # Plot mono channel
    plt.figure(figsize=(10, 4))
    plt.plot(sumbu_waktu, file_data, color="blue")
    plt.title(f"Waveform Audio - {os.path.basename(audio_data3)} (Mono)")
    plt.xlabel("Detik")
    plt.ylabel("Amplitudo")
    plt.grid()
    plt.show()
```





[33]:
```
# Calculate the maximum value and threshold
max_value = np.max(np.abs(file_data))
threshold = 0.2 * max_value

# Print the values
print(f"Maximum value of SDB: {max_value}")
print(f"Threshold value of SDB: {threshold}")
```

```
Maximum value of SDB: 13976
Threshold value of SDB: 2795.2000000000003
```

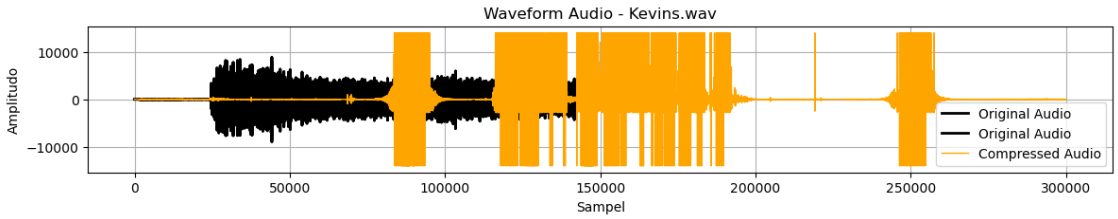
[34]:
```
compressed_audio = np.copy(file_data)  # Copy the audio data
mask_audio = np.abs(compressed_audio) > threshold  # Create a mask for values␣
↪below the threshold
print(f"Sampel Kompresi Berjumlah : {np.sum(mask_audio)}")  # Print the number␣
↪of samples to be compressed
```

```
Sampel Kompresi Berjumlah : 26509
```

[35]:
```
ratio = 7  # Compression ratio
compressed_audio[mask_audio] = threshold + (compressed_audio[mask_audio] -␣
↪threshold) * ratio  # Compress the audio data
compressed_audio = np.clip(compressed_audio, -max_value, max_value)  # Clip the␣
↪compressed audio data
```

[85]:
```
# Plot the original and compressed waveforms
plt.figure(figsize=(14, 2))

# Plot original audio
```

```
plt.plot(file_data, color='black', label='Original Audio', linewidth=2)  #␣
 ↪Increase line width
plt.title(f'Waveform Audio - {os.path.basename(audio_data3)}')
plt.xlabel('Sampel')
plt.ylabel('Amplitudo')
plt.grid()

# Plot compressed audio
plt.plot(compressed_audio, color='orange', label='Compressed Audio',␣
 ↪linewidth=1)  # Reduce line width
plt.legend()
plt.show()
```



[37]:
```
# Define the output file path
output_file_path = os.path.join(os.getcwd(), "Audio", "Kevins_compressed.wav")

# Save the compressed audio
sf.write(output_file_path, compressed_audio, sample_rate)

# Print the output file path
print(f"Compressed audio saved at: {output_file_path}")
```

```
Compressed audio saved at: c:\Users\HOME\Downloads\GitHub\TelkomAkses-
Dash\Technology-Multimedia\Audio\Kevins_compressed.wav
```

1.10.2 Kompresi Audio `Kevins.wav`

[87]:
```
import pyloudnorm as pyln  # pustaka pyln
import os  # pustaka os
import wave  # Import the wave module
```

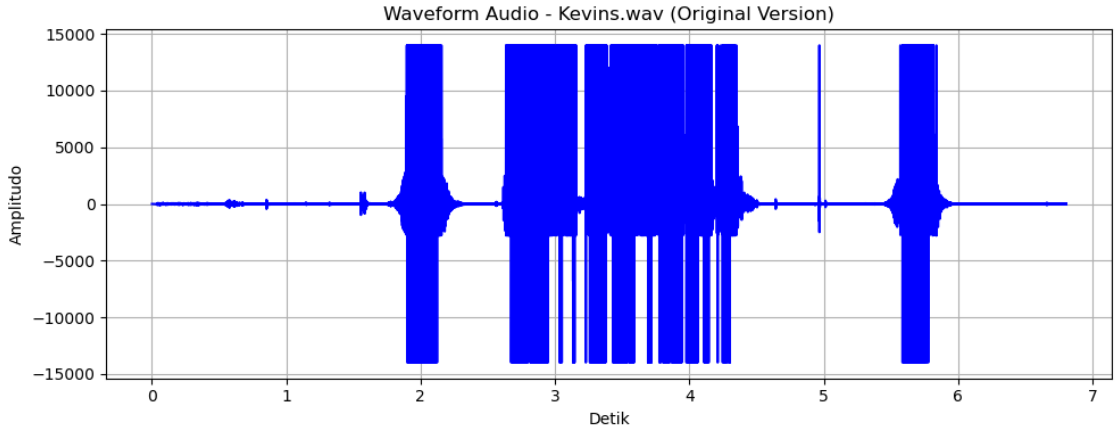
[86]:
```
with wave.open(file_kompress, "rb") as audio_wave:  # Open compressed audio file
    audio_width = audio_wave.getsampwidth()  # Get sample width
    audio_sample = audio_wave.getnframes()  # Get number of frames
    audio_frames = audio_wave.readframes(audio_sample)  # Read audio frames
    file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert frames to␣
 ↪numpy array
    sample_rate = audio_wave.getframerate()  # Get sample rate
```

```
    channels = audio_wave.getnchannels()  # Get number of channels
```

[88]:
```
print(f"File: {os.path.basename(file_kompress)}")
print(f"Sample rate: {sample_rate} Hz")  # Display sample rate
print(f"Total frames: {audio_sample}")  # Display number of frames
```

```
File: Kevins.wav
Sample rate: 44100 Hz
Total frames: 300032
```

[89]:
```
sumbu_waktu = np.linspace(0, audio_sample / sample_rate, audio_sample)  # Get␣
 ↪time axis

plt.figure(figsize=(10, 4))  # Create plot
plt.plot(sumbu_waktu, compressed_audio, color="blue")  # Plot audio data
plt.title(f"Waveform Audio - {os.path.basename(file_kompress)} (Original␣
 ↪Version)")  # Title plot
plt.xlabel("Detik")  # Label x
plt.ylabel("Amplitudo")  # Label y
plt.grid()  # Add grid
plt.tight_layout()  # Plot rapi
plt.show()  # Menampilkan plot
```



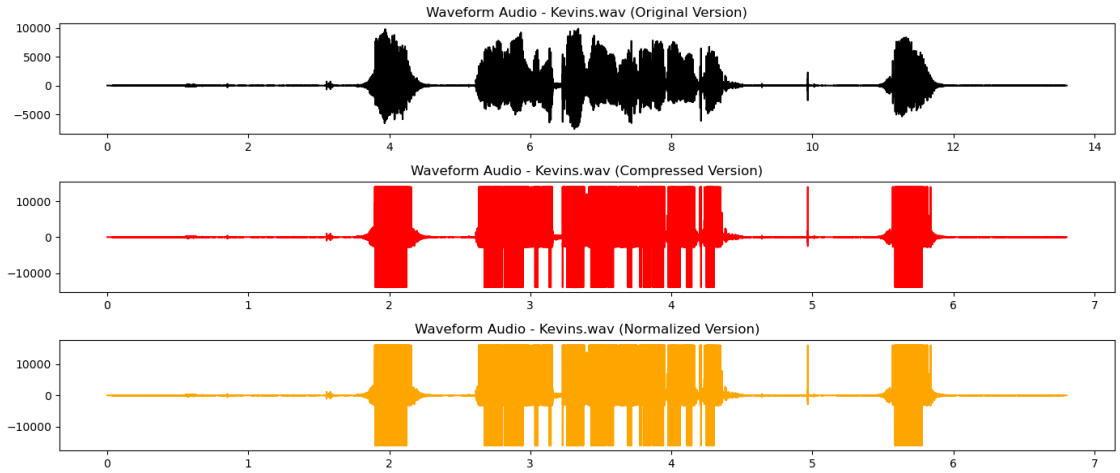
[90]:
```
float_audio = compressed_audio / np.max(np.abs(compressed_audio))  # Convert␣
 ↪audio to float
satuan = pyln.Meter(sample_rate)  # Create a Meter object
lufs_awal = satuan.integrated_loudness(float_audio)  # Calculate the integrated␣
 ↪loudness

lufs_akhir = -14.0  # Target loudness
gain = lufs_akhir - lufs_awal  # Calculate the gain
```

```
normalisasi_audio = pyln.normalize.loudness(float_audio, lufs_awal, lufs_akhir)␣
 ↪ # Normalize the audio to the target loudness
normalisasi_audio16 = np.int16(normalisasi_audio * 32767)  # Convert the␣
 ↪normalized audio to int16
```

[91]:
```
print(f"Nama File: {os.path.basename(file_kompress)}")  # Print the file name
print(f"LUFS Awal: {lufs_awal:.2f} LUFS")  # Print the initial loudness
print(f"LUFS Akhir: {lufs_akhir:.2f} LUFS")  # Print the target loudness
print(f"Gain: {gain} dB")  # Print the gain
```

```
Nama File: Kevins.wav
LUFS Awal: -7.79 LUFS
LUFS Akhir: -14.00 LUFS
Gain: -6.2093643057324135 dB
```

[94]:
```
file_kompress = audio_data3  # Define the variable file_kompress

# Adjust sumbu_waktu to match the length of file_data
sumbu_waktu = np.linspace(0, len(file_data) / sample_rate, len(file_data))

fig, ax = plt.subplots(3, 1, figsize=(14, 6))  # Create plot with 3 subplots

ax[0].plot(sumbu_waktu, file_data, color="black", label="Original Version")  #␣
 ↪Plot original audio
ax[0].set_title(f"Waveform Audio - {os.path.basename(file_kompress)} (Original␣
 ↪Version)")  # Title plot

# Adjust sumbu_waktu to match the length of compressed_audio
sumbu_waktu_compressed = np.linspace(0, len(compressed_audio) / sample_rate,␣
 ↪len(compressed_audio))
ax[1].plot(sumbu_waktu_compressed, compressed_audio, color="red",␣
 ↪label="Compressed Version")  # Plot compressed audio
ax[1].set_title(f"Waveform Audio - {os.path.basename(file_kompress)}␣
 ↪(Compressed Version)")  # Title plot

# Adjust sumbu_waktu to match the length of normalisasi_audio16
sumbu_waktu_normalized = np.linspace(0, len(normalisasi_audio16) / sample_rate,␣
 ↪len(normalisasi_audio16))
ax[2].plot(sumbu_waktu_normalized, normalisasi_audio16, color="orange",␣
 ↪label="Normalized Version")  # Plot normalized audio
ax[2].set_title(f"Waveform Audio - {os.path.basename(file_kompress)}␣
 ↪(Normalized Version)")  # Title plot

plt.tight_layout()  # Plot rapi
plt.show()  # Menampilkan plot
```



1.10.3 Mari Kita Analisis (MaKiAn)

Pada tahap ini, kita mengambil data audio dari file `Kevins.wav`. Data audio ini kemudian dibaca menggunakan pustaka `wave` dan diubah menjadi array numpy untuk memudahkan proses selanjutnya Setelah data audio berhasil diambil, kita menampilkan beberapa informasi dasar seperti sample rate, jumlah channel, total frame, dan durasi audio. Dari informasi ini, kita mengetahui bahwa audio memiliki sample rate sebesar 44100 Hz dan hanya memiliki satu channel (mono audio). Visualisasi audio dilakukan dengan membuat plot dari data audio terhadap waktu. Pada visualisasi ini, kita dapat melihat bagaimana bentuk gelombang audio dari channel kiri sebelum dilakukan kompresi.Kompresi audio dilakukan dengan menggunakan teknik thresholding. Nilai threshold ditentukan sebagai 20% dari nilai maksimum amplitudo audio. Sampel audio yang melebihi threshold akan dikompresi dengan rasio tertentu (misalnya 7:1). Kompresi ini bertujuan untuk mengurangi dinamika audio sehingga lebih konsisten dalam hal amplitudo. Setelah dilakukan kompresi, kita melakukan visualisasi kembali untuk melihat perubahan bentuk gelombang audio. Pada visualisasi ini, kita dapat melihat bahwa amplitudo audio menjadi lebih konsisten dan tidak ada lonjakan yang terlalu tinggi. Normalisasi audio dilakukan untuk mencapai target loudness sekitar -14 LUFS. Proses normalisasi melibatkan penghitungan loudness awal, kemudian menyesuaikan gain untuk mencapai target loudness. Audio yang telah dinormalisasi kemudian dikonversi kembali ke format int16 untuk disimpan. Visualisasi terakhir dilakukan untuk melihat bentuk gelombang audio setelah normalisasi. Pada visualisasi ini, kita dapat melihat bahwa amplitudo audio telah disesuaikan untuk mencapai target loudness yang diinginkan.

Kesimpulan

- Audio memiliki sample rate sebesar 44100 Hz dan hanya memiliki satu channel (mono audio).
- Kompresi audio berhasil mengurangi dinamika amplitudo sehingga lebih konsisten.
- Normalisasi audio berhasil mencapai target loudness sekitar -14 LUFS.
- Visualisasi yang telah dibuat memberikan gambaran yang jelas mengenai perubahan bentuk gelombang audio sebelum dan sesudah kompresi serta normalisasi.

1.11 What's The Problem

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).
- 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.

1.11.1 Bandpass Filter

[127]:
```
from scipy.signal import butter, filtfilt
```

[96]:
```
bandpass_path = os.path.join(os.getcwd(), "Audio", "Kevin Voice.wav")  # Define␣
↪the output

with wave.open(file_kompress, "rb") as audio_wave:  # Open compressed audio file
    audio_width = audio_wave.getsampwidth()  # Get sample width
    audio_sample = audio_wave.getnframes()  # Get number of frames
    audio_frames = audio_wave.readframes(audio_sample)  # Read audio frames
    file_data = np.frombuffer(audio_frames, dtype="int16")  # Convert frames to␣
↪numpy array
    sample_rate = audio_wave.getframerate()  # Get sample rate
    channels = audio_wave.getnchannels()  # Get number of channels
```

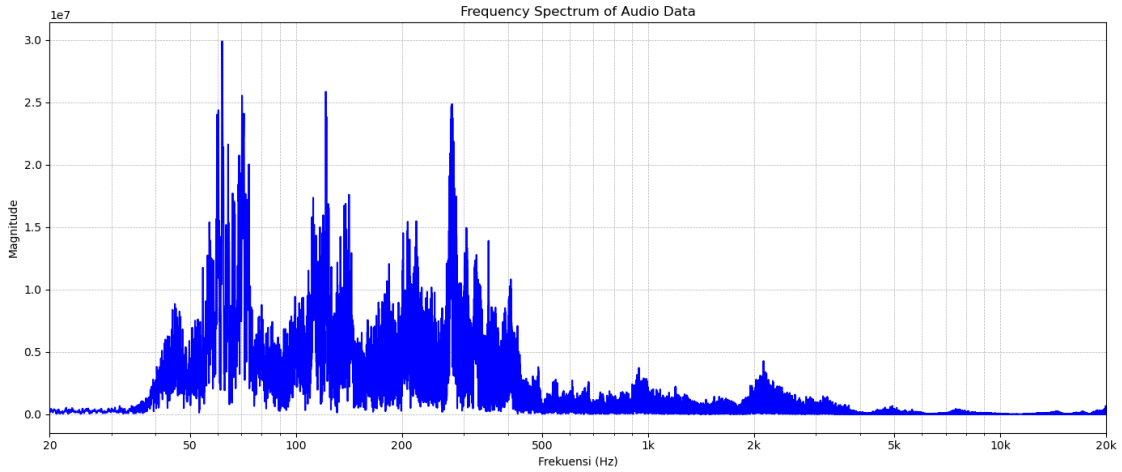
[97]:
```
print(f"File: {os.path.basename(file_kompress)}")
print(f"Sample rate: {sample_rate} Hz")  # Display sample rate
print(f"Total frames: {audio_sample}")  # Display number of frames
```

```
File: Kevins.wav
Sample rate: 44100 Hz
Total frames: 300032
```

[98]:
```
def frequency_spectrum(data, sample_rate, title):
    fft_data = np.fft.fft(data)  # Compute the FFT
    magnitude = np.abs(fft_data[:len(fft_data) // 2])  # Compute the magnitude␣
↪spectrum
```

```
    frequency = np.fft.fftfreq(len(fft_data), 1 / sample_rate)  # Generate the␣
 ↪frequency axis
    post_frequency = frequency[:len(frequency) // 2]  # Get the positive␣
 ↪frequencies
```

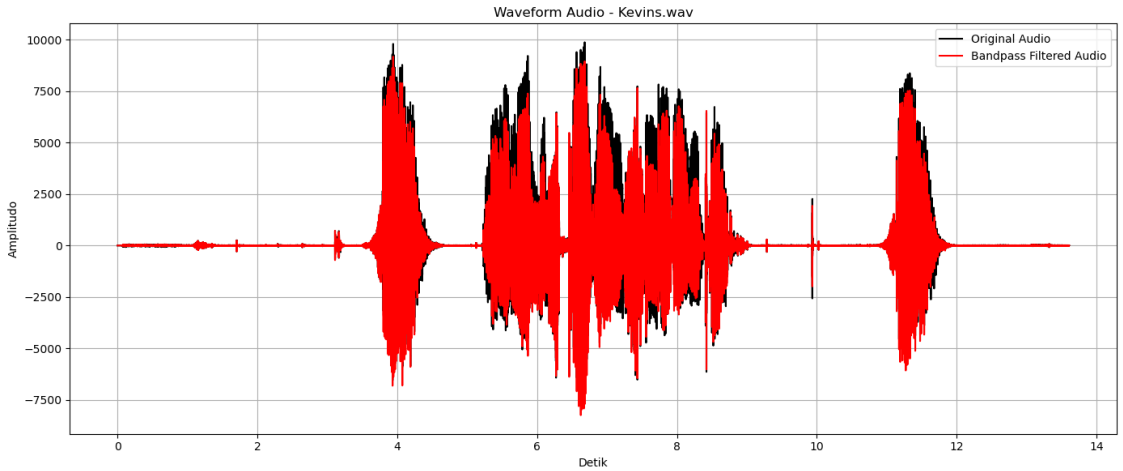
[125]:
```
# Ensure post_frequency and magnitude are defined
post_frequency = np.fft.fftfreq(len(file_data), 1 / sample_rate)[:
 ↪len(file_data) // 2]
magnitude = np.abs(np.fft.fft(file_data)[:len(file_data) // 2])

title = "Frequency Spectrum of Audio Data"  # Define the title

plt.figure(figsize=(14, 6))  # Create plot
plt.plot(post_frequency, magnitude, color="blue")  # Plot the magnitude spectrum
plt.title(title)  # Title plot
plt.xscale("log")  # Set x-axis to log scale
plt.xlim(20, 20000)  # Set x-axis limits
plt.xticks([20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000],  # Set␣
 ↪x-axis ticks
           ["20", "50", "100", "200", "500", "1k", "2k", "5k", "10k", "20k"])
plt.xlabel("Frekuensi (Hz)")  # Label x
plt.ylabel("Magnitude")  # Label y
plt.grid(True, which="both", linestyle="--", linewidth=0.5)  # Add grid
plt.tight_layout()  # Plot rapi
plt.show()  # Menampilkan plot
```



[129]:
```
cut_low = 100  # Low-cut frequency
cut_high = 10000  # High-cut frequency
order = 4  # Filter order
```

```
cut_low_nyquist = cut_low / (sample_rate / 2)  # Normalize the low-cut frequency
cut_high_nyquist = cut_high / (sample_rate / 2)  # Normalize the high-cut␣
 ↪frequency

b, a = butter(order, [cut_low_nyquist, cut_high_nyquist], btype="band")  #␣
 ↪Design the bandpass filter
audio_bandpass = filtfilt(b, a, file_data)  # Apply the bandpass filter
audio_bandpass = np.int16(audio_bandpass)  # Convert the filtered audio to int16

frequency_spectrum(audio_bandpass, sample_rate, "Frequency Spectrum of Bandpass␣
 ↪Filtered Audio Data")  # Plot the frequency spectrum
```

[132]:
```
# Ensure sumbu_waktu matches the length of file_data
sumbu_waktu = np.linspace(0, len(file_data) / sample_rate, len(file_data))

plt.figure(figsize=(14, 6))  # Create plot
plt.plot(sumbu_waktu, file_data, color="black", label="Original Audio")  # Plot␣
 ↪original audio

# Ensure sumbu_waktu matches the length of audio_bandpass
sumbu_waktu_bandpass = np.linspace(0, len(audio_bandpass) / sample_rate,␣
 ↪len(audio_bandpass))
plt.plot(sumbu_waktu_bandpass, audio_bandpass, color="red", label="Bandpass␣
 ↪Filtered Audio")  # Plot bandpass filtered audio

plt.title(f"Waveform Audio - {os.path.basename(file_kompress)}")  # Title plot
plt.xlabel("Detik")  # Label x
plt.ylabel("Amplitudo")  # Label y
plt.legend()  # Add legend
plt.grid()  # Add grid
plt.tight_layout()  # Plot rapi
plt.show()  # Menampilkan plot
```



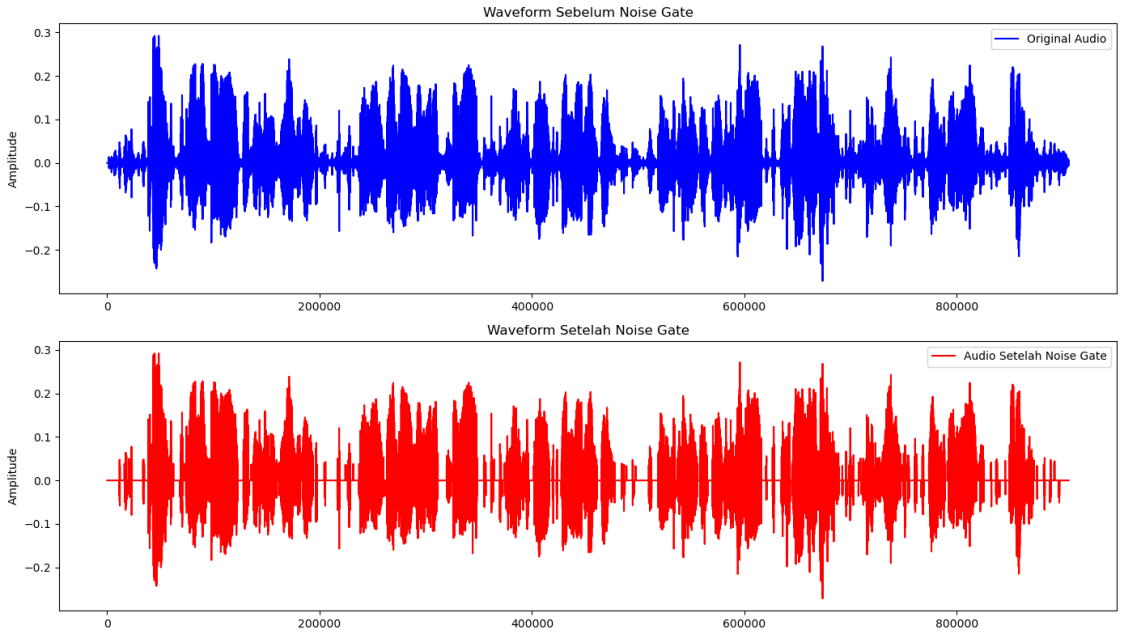
1.11.2 Noise Gate

[133]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf
```

[141]:
```
# Load audio file
audio, sr = librosa.load("Audio/Kevins Voice.wav", sr=None)

# Tentukan threshold (dalam decibels)
threshold_db = -30  # ambang batas dalam dB

# Konversi threshold ke linear scale
threshold = librosa.db_to_amplitude(threshold_db)

# Menerapkan noise gate
def apply_noise_gate(audio, threshold):
    abs_audio = np.abs(audio)
    mask = abs_audio >= threshold
    gated_audio = audio * mask
    return gated_audio

gated_audio = apply_noise_gate(audio, threshold)
```

[142]:
```
# Visualisasi audio asli dan setelah noise gate
plt.figure(figsize=(14, 8))

# Plot waveform asli
plt.subplot(2, 1, 1)
plt.plot(audio, color='b', label='Original Audio')
plt.title('Waveform Sebelum Noise Gate')
plt.ylabel('Amplitude')
plt.legend()

# Plot waveform setelah noise gate
plt.subplot(2, 1, 2)
plt.plot(gated_audio, color='r', label='Audio Setelah Noise Gate')
plt.title('Waveform Setelah Noise Gate')
plt.ylabel('Amplitude')
plt.legend()

# Tampilkan plot
plt.tight_layout()
```

```
plt.show()
```



[143]:
```
output_file = os.path.join("Audio", "Kevins Voice.wav")
sf.write(output_file, gated_audio, sr)

print(f"Audio yang sudah dihilangkan noise-nya disimpan di: {output_file}")
```

```
Audio yang sudah dihilangkan noise-nya disimpan di: Audio\Kevins Voice.wav
```

1.12 Mari Kita Analisis (MaKiAn)

Dari hasil Bandpass serta Noise Gate yang didapat bahwa audio yang diberikan dan setelah dilakukan proses maka terdapat perbedaan yang cukup signifikan di bagian `Amplitudo`. Maka dari itu dalam proses Noise Gate ini juga mempengaruhi besar dan kecilnya `Amplitudo`. Kemudian pada hasil Bandpass didapatkan bahwa frekuensi suara sangatlah lebih sempit daripada sebelum diberikannya filter Bandpass.

1.13 Libraries/Tools/Resources Used in the Work of `Hands On 1`

Nama Sumber/Alat	Tautan Sumber	Keterangan	Bukti
Chat GPT (1)	https://chatgpt.com/share/66fb0f89-6194-8001-a0de-756c4fb8e126	Chat GPT digunakan untuk membantu mencari solusi dari permasalahan yang diminta pada `Problem 5 : How To Make A Noise Gate In My Audio`	-
Chat GPT (2)	https://chatgpt.com/share/66fb1f1d-60dc-8001-8f99-298d6e123bc7	Chat GPT digunakan untuk mencari solusi dari pertanyaan yang diberikan pada poin `Problem 2 : How To Make A Fading Technic For A Non Linear`	-
Audio Convert Online	https://online-audio-converter.com/	Audio Convert Online digunakan untuk membantu mengkonversi audio dari format .M4A ke .WAV	-