

Nama : Elma Nurul Fatika NIM : 122140069 Mata Kuliah : Sistem Teknologi Multimedia

Link Repository GitHub : <https://github.com/122140069-ElmaNF/Multimedia-hands-on/tree/main>

Note : pak disini saya buat struktur foldernya dipisah per soal karena waktu saya jadiin satu di githubnya dia filenya tidak bisa dibuka (warning: "sorry, this is too big to display"). terimakasih pak.

Soal 1: Rekaman dan Analisis Suara Multi-Level

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
from IPython.display import Audio, HTML, display
import os
import matplotlib

%matplotlib inline

print("Library versions:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")
```

Library versions:

NumPy: 2.2.6
Matplotlib: 3.10.5
Librosa: 0.11.0

```
In [ ]: # Load audio file
audio_path = 'data/Audio1.wav'

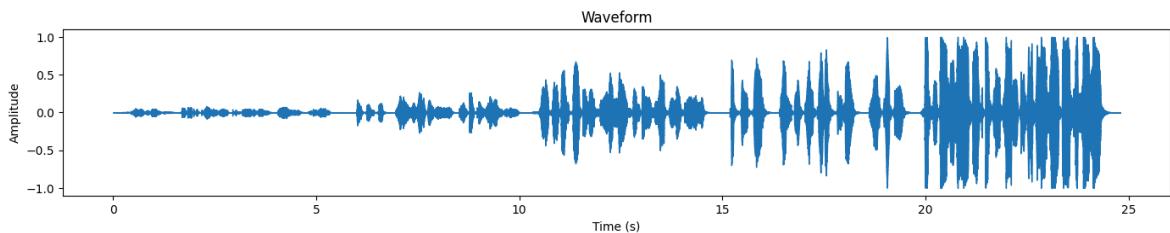
y, sr = librosa.load(audio_path, sr=None)

# Load waveform
plt.figure(figsize=(14, 5))
plt.subplot(2, 1, 1)
plt.title('Waveform')
librosa.display.waveform(y, sr=sr)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')

display(Audio(y_original, rate=sr_original))

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





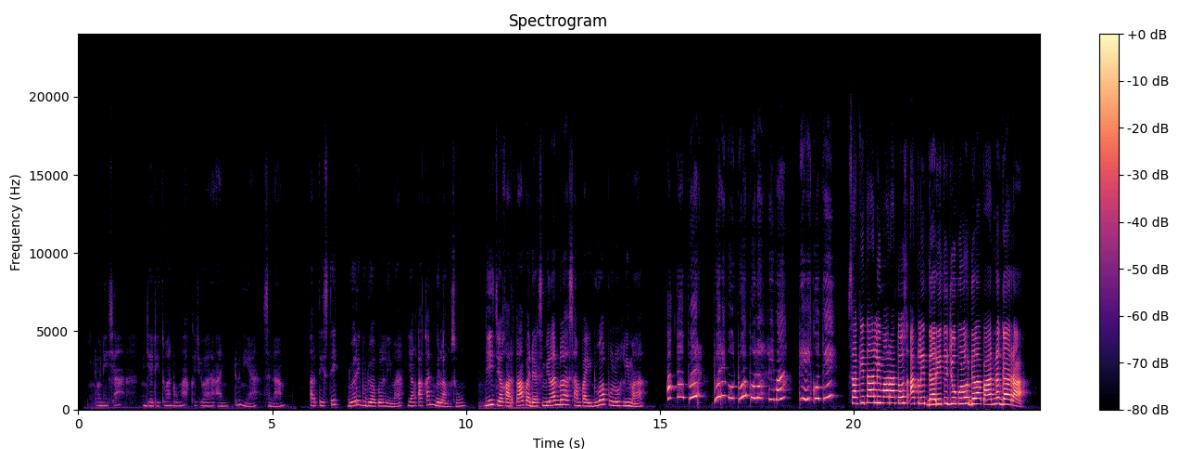
Wavefrom diatas menampilkan perubahan amplitudo dari durasi selama 25 detik. Pada 5 detik awal amplitudonya kecil karena suara pada rekaman berbisik, kemudian semakin lama semakin meningkat amplitudonya dikarenakan suara yang direkam semakin tinggi dan kuat. Secara keseluruhan wavefrom ini menampilkan perubahan dari suara tenang hingga suara semakin keras di akhir.

```
In [ ]: #Load audio
audio_path = 'data/Audio1.wav'

y, sr = librosa.load(audio_path, sr=None)

#spectrogram
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

plt.figure(figsize=(14, 5))
plt.title('Spectrogram')
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.colorbar(format='%+2.0f dB')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.tight_layout()
plt.show()
```



Gambar spectrogram diatas menampilkan perubahan suara selama 25 detik. Pada spectrogram diatas memiliki frekuensi rendah sampai menengah sekitar 0-5000 Hz terutama di akhir rekaman. Hal ini berarti suara yang ada di akhir rekaman lebih keras dan memiliki nada rendah. Jadi, spectrogram ini menunjukkan bahwa intensitas suara meningkat dan bervariasi.

```
In [8]: audio_path = 'data/Audio1.wav'

y_original, sr_original = librosa.load(audio_path, sr=None)

print("Informasi dari audio asli ")
```

```
print(f"Sampling rate : {sr_original} Hz")
print(f"Durasi       : {librosa.get_duration(y=y_original, sr=sr_original):.2f}")

display(Audio(y_original, rate=sr_original))
```

Informasi dari audio asli
 Sampling rate : 48000 Hz
 Durasi : 24.79 detik



In [22]:

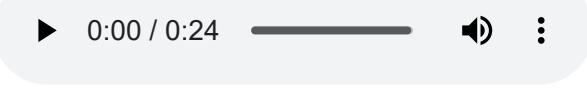
```
#melakukan resampling dengan mengubah sampling rate jadi 3000 Hz

target_sr = 3000
y_resampled = librosa.resample(y_original, orig_sr=sr_original, target_sr=target_sr)

print("Informasi audio setelah resampling")
print(f"Sampling rate : {target_sr} Hz")
print(f"Durasi       : {librosa.get_duration(y=y_resampled, sr=target_sr):.2f}")

# Putar audio hasil resampling
display(Audio(y_resampled, rate=target_sr))
```

Informasi audio setelah resampling
 Sampling rate : 3000 Hz
 Durasi : 24.79 detik



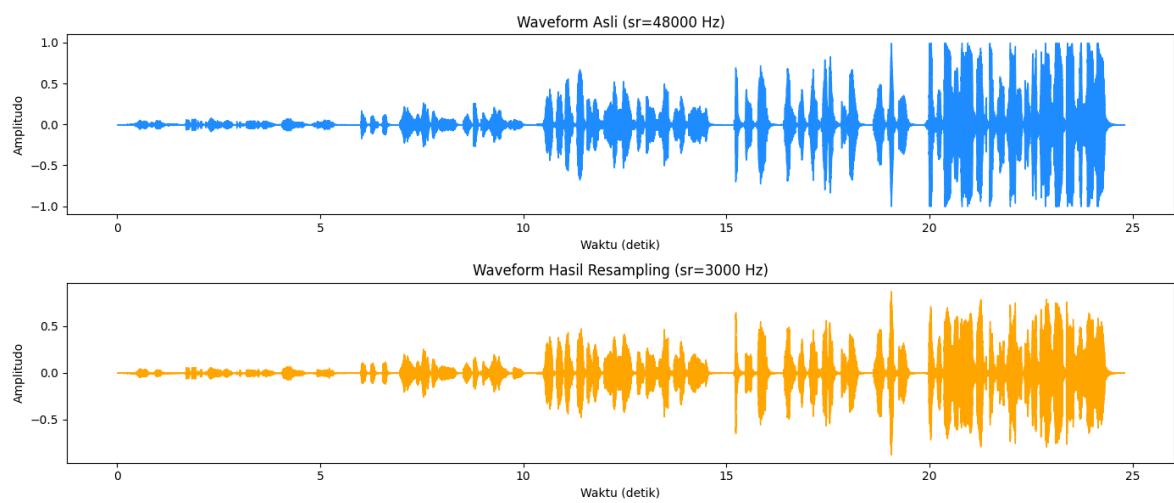
In []:

```
# Visualisasi perbandingan waveform
plt.figure(figsize=(14, 6))

# Waveform asli
plt.subplot(2, 1, 1)
librosa.display.waveshow(y_original, sr=sr_original, color='dodgerblue')
plt.title(f"Waveform Asli (sr={sr_original} Hz)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# Waveform hasil resampling
plt.subplot(2, 1, 2)
librosa.display.waveshow(y_resampled, sr=target_sr, color='orange')
plt.title(f"Waveform Hasil Resampling (sr={target_sr} Hz)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

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



bandingkan kualitas dan durasinya: Hasil dari audio yang telah saya resampling menjadi 3000 Hz yaitu pada 10 detik awal suaranya menjadi mendem, namun ketika memasuki durasi ke 11 detik sampai 25 detik suaranya menjadi jelas per kata. Secara visual dari waveform, hasil resampling lebih tajam-tajam dibandingkan dengan waveform dari audio asli.

Soal 2: Noise Reduction dengan Filtering

```
In [ ]:
import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
from IPython.display import Audio, HTML, display
import os
import matplotlib
from scipy.signal import butter, filtfilt, spectrogram
from IPython.display import Audio, display
from pydub import AudioSegment

%matplotlib inline

print("Library versions:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")
```

Library versions:
 NumPy: 2.2.6
 Matplotlib: 3.10.5
 Librosa: 0.11.0

```
In [ ]:
import os
print(os.getcwd())

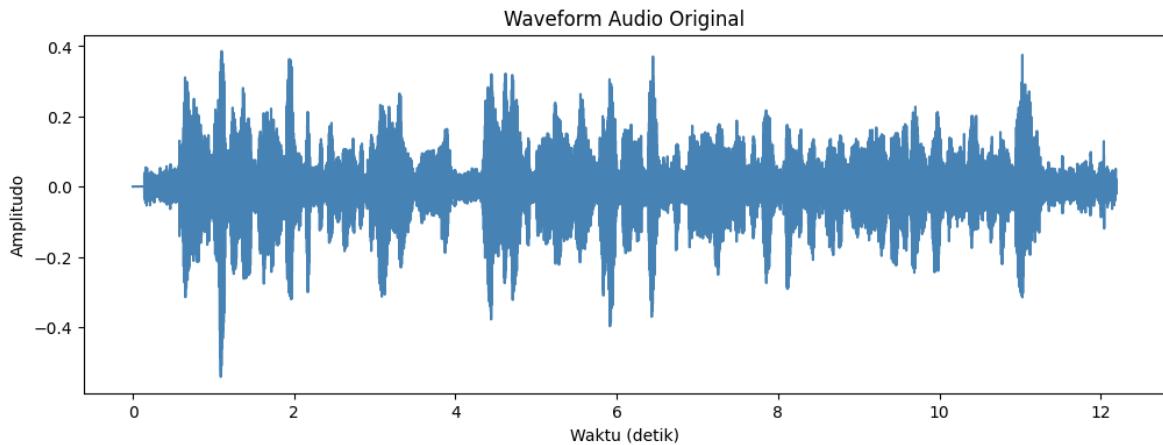
# Load file audio
file_path = os.path.join("../", "Data", "Audio2.wav")
y, sr = librosa.load(file_path, sr=None)

# Tampilkan info dasar
print(f"Sample rate (frekuensi sampling): {sr} Hz")
print(f"Durasi audio: {len(y)/sr:.2f} detik")

# Plot waveform
plt.figure(figsize=(12, 4))
plt.plot(np.linspace(0, len(y)/sr, len(y)), y, color='steelblue')
plt.title("Waveform Audio Original")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.show()

# Dengarkan audio
Audio("../Data/Audio2.wav")
```

c:\Multimedia\Handson-audio\Soal2
 Sample rate (frekuensi sampling): 48000 Hz
 Durasi audio: 12.19 detik



Out[]: ▶ 0:00 / 0:12 ⏪ ⏴ ⏵

```
In [ ]: # Load audio
audio_path = "../Data/Audio2.wav"
y, sr = librosa.load(audio_path, sr=None)

# Fungsi high-pass filter
def highpass_filter(data, cutoff, fs, order=6):
    nyq = 0.5 * fs # frekuensi Nyquist
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    return filtfilt(b, a, data)

# Terapkan filter
cutoff = 300 #cutoff di 300 Hz
y_hpf = highpass_filter(y, cutoff, sr)

sf.write("Audio2_highpass.wav", y_hpf, sr)

# Plot hasil
plt.figure(figsize=(12, 4))
plt.plot(y, label="Original", alpha=0.6)
plt.plot(y_hpf, label=f"High-pass {cutoff} Hz", alpha=0.8)
plt.legend()
plt.title("High-Pass Filter")
plt.xlabel("Sampel")
plt.ylabel("Amplitudo")
plt.show()

# Hitung spektrogram
D_original = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
D_hpf = librosa.amplitude_to_db(np.abs(librosa.stft(y_hpf)), ref=np.max)

# Visualisasi spektrogram
plt.figure(figsize=(12, 6))

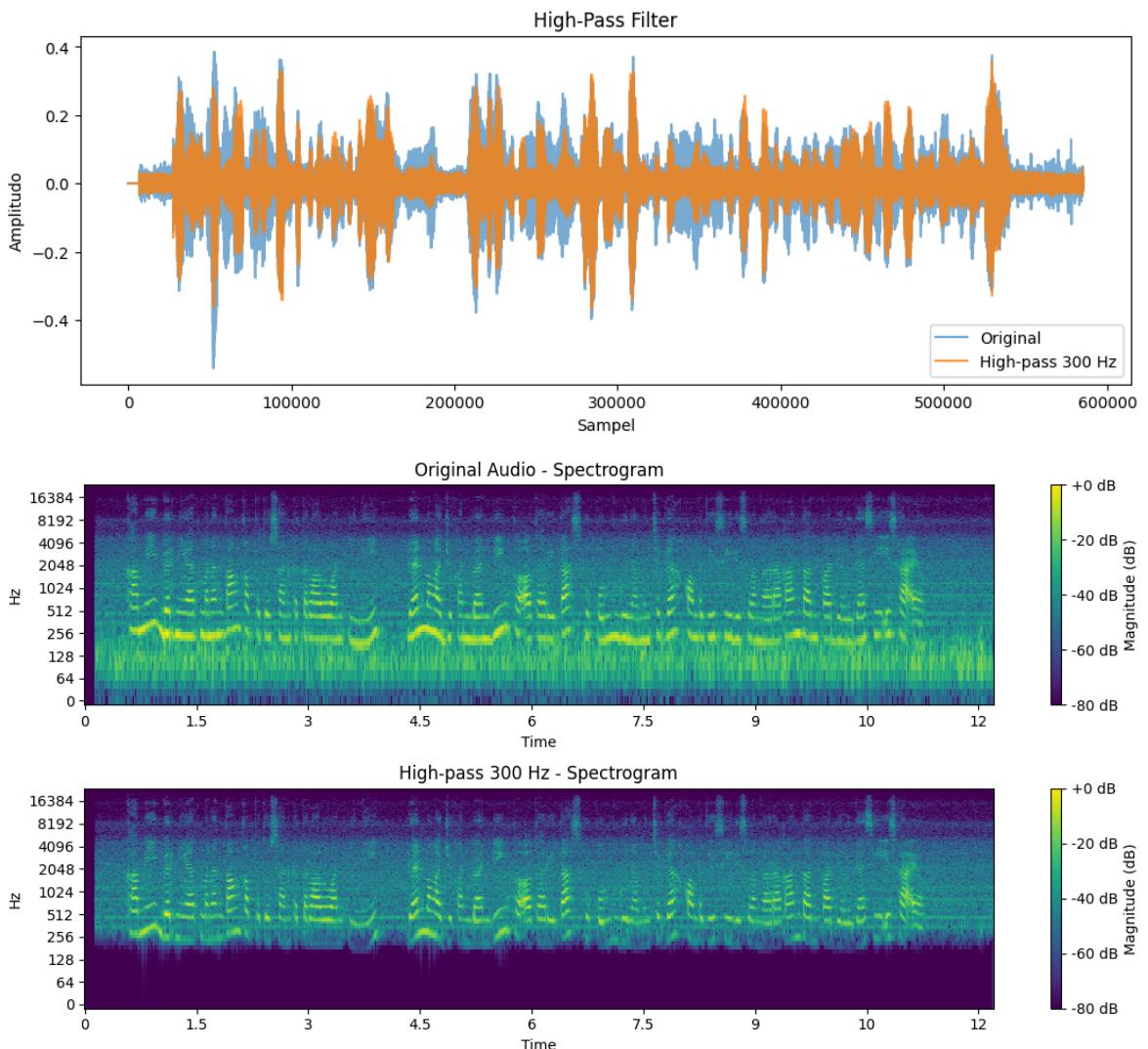
plt.subplot(2, 1, 1)
librosa.display.specshow(D_original, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
plt.title("Original Audio - Spectrogram")
plt.colorbar(format='%.2f dB', label='Magnitude (dB)')

plt.subplot(2, 1, 2)
librosa.display.specshow(D_hpf, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
```

```
plt.title(f"High-pass {cutoff} Hz - Spectrogram")
plt.colorbar(format='%+2.0f dB', label='Magnitude (dB)')

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

Audio("Audio2_highpass.wav")



Out[]:

▶ 0:00 / 0:12

```
# Load audio
y, sr = librosa.load("../Data/Audio2.wav", sr=None)

# Fungsi low-pass filter
def lowpass_filter(data, cutoff, fs, order=6):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return filtfilt(b, a, data)

# Terapkan filter
cutoff = 3000
y_lpf = lowpass_filter(y, cutoff, sr)
```

```

sf.write("Audio2_lowpass.wav", y_lpf, sr)

# Plot hasil
plt.figure(figsize=(12, 4))
plt.plot(y, label="Original", alpha=0.6)
plt.plot(y_lpf, label=f"Low-pass {cutoff} Hz", alpha=0.8)
plt.legend()
plt.title("Low-Pass Filter")
plt.xlabel("Sampel")
plt.ylabel("Amplitudo")
plt.show()

# Hitung spektrogram
D_original = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
D_lpf = librosa.amplitude_to_db(np.abs(librosa.stft(y_lpf)), ref=np.max)

# Visualisasi spektrogram berdampingan
plt.figure(figsize=(12, 6))

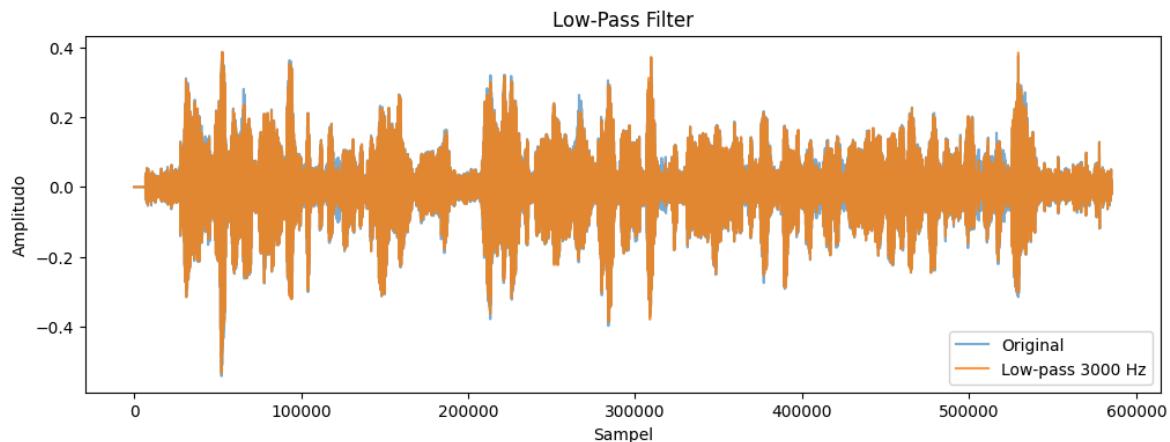
plt.subplot(2, 1, 1)
librosa.display.specshow(D_original, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
plt.title("Original Audio - Spectrogram")
plt.colorbar(format='%+2.0f dB', label='Magnitude (dB)')

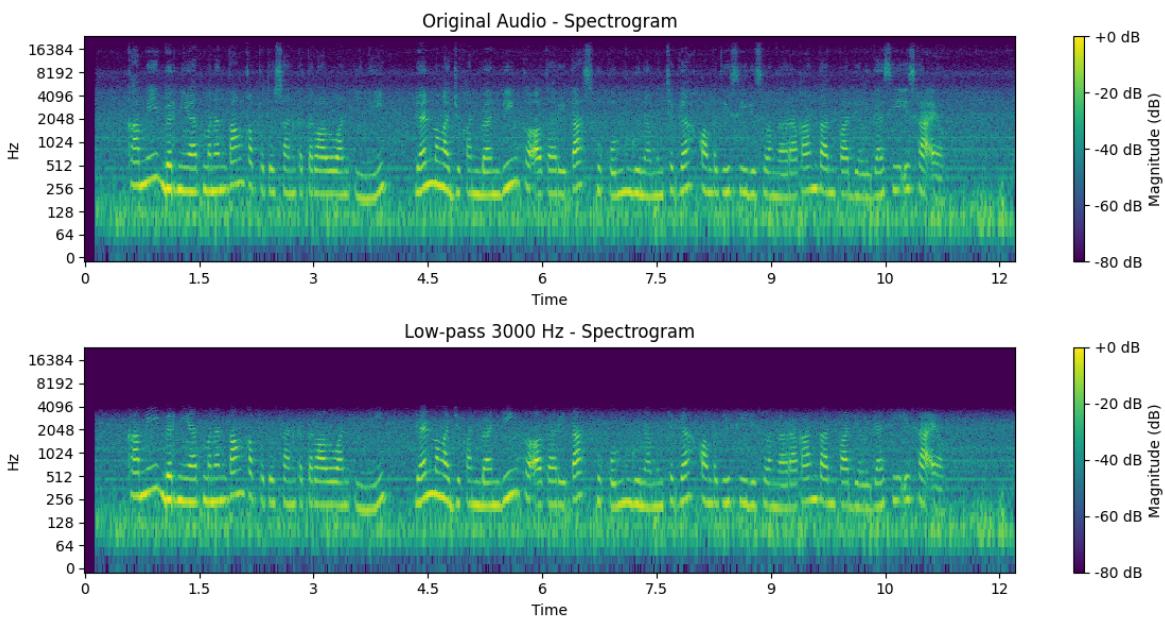
plt.subplot(2, 1, 2)
librosa.display.specshow(D_lpf, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
plt.title(f"Low-pass {cutoff} Hz - Spectrogram")
plt.colorbar(format='%+2.0f dB', label='Magnitude (dB)')

plt.tight_layout()
plt.show()

# Dengarkan hasil
Audio("Audio2_lowpass.wav")

```





Out[]:

▶ 0:00 / 0:12 ◀ ▶ ⋮

In []:

```
# Load audio
y, sr = librosa.load("../Data/Audio2.wav", sr=None)

# Fungsi band-pass filter
def bandpass_filter(data, lowcut, highcut, fs, order=6):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band', analog=False)
    return filtfilt(b, a, data)

# Terapkan filter
lowcut = 300
highcut = 3000
y_bpf = bandpass_filter(y, lowcut, highcut, sr)

# Simpan hasil
sf.write("Audio2_bandpass.wav", y_bpf, sr)

# Plot hasil
plt.figure(figsize=(12, 4))
plt.plot(y, label="Original", alpha=0.6)
plt.plot(y_bpf, label=f"Band-pass {lowcut}-{highcut} Hz", alpha=0.8)
plt.legend()
plt.title("Band-Pass Filter")
plt.xlabel("Sampel")
plt.ylabel("Amplitudo")
plt.show()

# Hitung spektrogram
D_original = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
D_bpf = librosa.amplitude_to_db(np.abs(librosa.stft(y_bpf)), ref=np.max)

# Visualisasi spektrogram berdampingan
plt.figure(figsize=(12, 6))
```

```

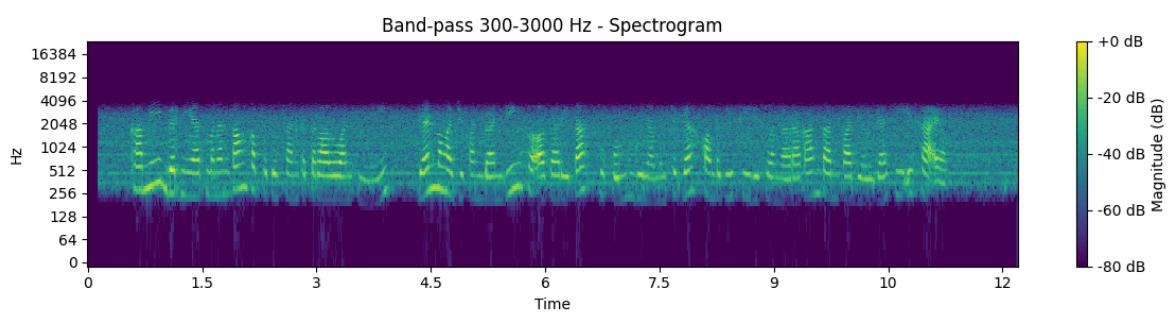
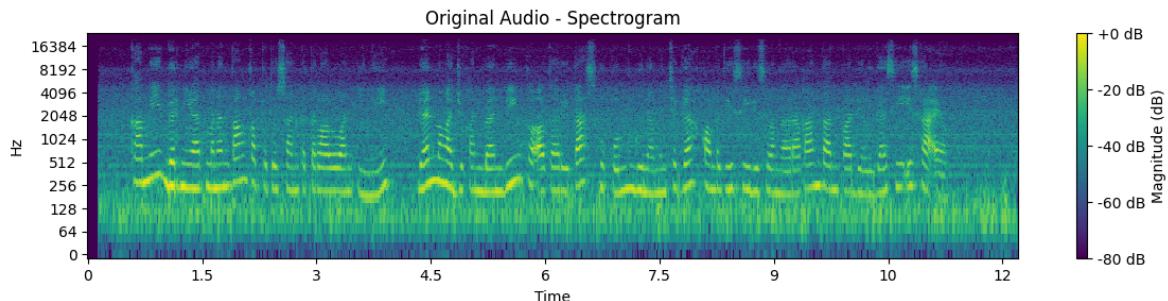
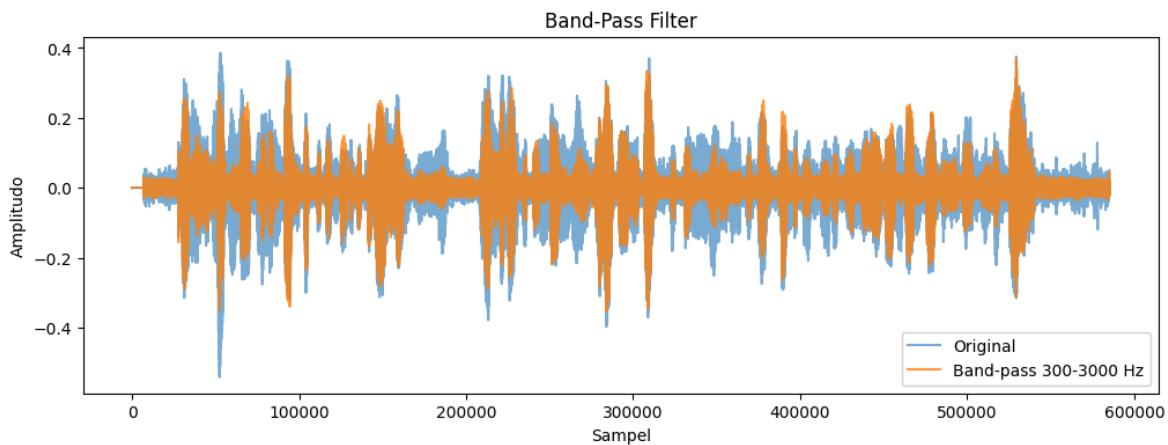
plt.subplot(2, 1, 1)
librosa.display.specshow(D_original, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
plt.title("Original Audio - Spectrogram")
plt.colorbar(format='%+2.0f dB', label='Magnitude (dB)')

plt.subplot(2, 1, 2)
librosa.display.specshow(D_bpf, sr=sr, x_axis='time', y_axis='log', cmap='viridis')
plt.title(f"Band-pass {lowcut}-{highcut} Hz - Spectrogram")
plt.colorbar(format='%+2.0f dB', label='Magnitude (dB)')

plt.tight_layout()
plt.show()

# Dengarkan hasil
Audio("Audio2_bandpass.wav")

```



Out[]:

Jelaskan:

1. Jenis noise yang muncul pada rekaman anda
2. Filter mana yang paling efektif untuk mengurangi noise tersebut
3. Nilai cutoff yang memberikan hasil terbaik
4. Bagaimana kualitas suara (kejelasan ucapan) setelah proses filtering

Jawab:

1. Noise yang muncul pada rekaman saya yaitu suara kipas angin yang kencang, disitu saya merekam sangat dekat dengan kipas angin sehingga mengeluarkan suara "tnggggggg".
2. menurut saya filter high pass yang paling efektif diantara low pass dan band pass. karena menurut saya di ketiga filter tersebut noisenya masih sama sama terdengar. namun ketika di high pass, suara saya terdengar lebih nyaring sehingga sangat kontras dengan suara noisenya. ketika di low pass itu noisenya sangat berkurang menurut saya, tapi suara saya juga jadi besar dan agak dengung/mendep sedikit, jadi sangat kurang efektif jika menggunakan filter low pass. dan untuk band pass dia juga cukup jelas untuk suara saya, tapi seperti seimbang dengan noise, jadi mirip suara di radio jaman dahulu, dan sedikit kurang menurut saya.
3. untuk nilai cutoff terbaik ada pada 300 Hz dengan filter high pass
4. kualitas suara/kejelasan ucapan saya menjadi bervariasi. pada filter highpass suara saya sangat jelas, ketika menggunakan filter lowpass suara saya menjadi besar dan kurang jelas, ketika menggunakan filter bandpass suara saya menjadi jelas juga.

Soal 3: Pitch Shifting dan Audio Manipulation

In [2]:

```

import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
from IPython.display import Audio, HTML, display
import os
import matplotlib
from scipy.signal import butter, filtfilt, spectrogram
from IPython.display import Audio, display
from pydub import AudioSegment

%matplotlib inline

print("Library versions:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")

```

Library versions:

NumPy: 2.2.6
 Matplotlib: 3.10.5
 Librosa: 0.11.0

C:\Users\Elma\AppData\Roaming\Python\Python313\site-packages\pydub\utils.py:170:
 RuntimeWarning: Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may no
 t work
 warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work",
 RuntimeWarning)

In []:

```

# 1. Load Audio Asli
file_path = "../Data/Audio1.wav"
y, sr = librosa.load(file_path, sr=None)
print(f"Durasi: {librosa.get_duration(y=y, sr=sr):.2f} detik | Sample Rate: {sr}")

# 2. Visualisasi Waveform dan Spectrogram Asli
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
librosa.display.waveform(y, sr=sr)
plt.title("Waveform Asli")

plt.subplot(2, 1, 2)
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title("Spectrogram Asli")
plt.colorbar(format="%+2.0f dB")
plt.tight_layout()
plt.show()

# 3. Pitch Shifting
# pitch +7 dan +12
y_pitch7 = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)
y_pitch12 = librosa.effects.pitch_shift(y, sr=sr, n_steps=12)

# 4. Simpan hasil pitch shifting
sf.write("Audio_Pitch7.wav", y_pitch7, sr)
sf.write("Audio_Pitch12.wav", y_pitch12, sr)

```

```
# 5. Visualisasi Hasil Pitch Shifting
fig, ax = plt.subplots(2, 2, figsize=(12, 8))

# Waveform pitch +7
librosa.display.waveshow(y_pitch7, sr=sr, ax=ax[0,0])
ax[0,0].set_title("Waveform Pitch +7")

# Spectrogram pitch +7
D7 = librosa.amplitude_to_db(np.abs(librosa.stft(y_pitch7)), ref=np.max)
librosa.display.specshow(D7, sr=sr, x_axis='time', y_axis='hz', ax=ax[0,1], cmap='magma')
ax[0,1].set_title("Spectrogram Pitch +7")

# Waveform pitch +12
librosa.display.waveshow(y_pitch12, sr=sr, ax=ax[1,0])
ax[1,0].set_title("Waveform Pitch +12")

# Spectrogram pitch +12
D12 = librosa.amplitude_to_db(np.abs(librosa.stft(y_pitch12)), ref=np.max)
librosa.display.specshow(D12, sr=sr, x_axis='time', y_axis='hz', ax=ax[1,1], cmap='magma')
ax[1,1].set_title("Spectrogram Pitch +12")

plt.tight_layout()
plt.show()

audio7 = AudioSegment.from_file("Audio_Pitch7.wav")
audio12 = AudioSegment.from_file("Audio_Pitch12.wav")
combined = audio7 + audio12 # gabung berurutan

combined.export("Audio_Chipmunk_Combined.wav", format="wav")

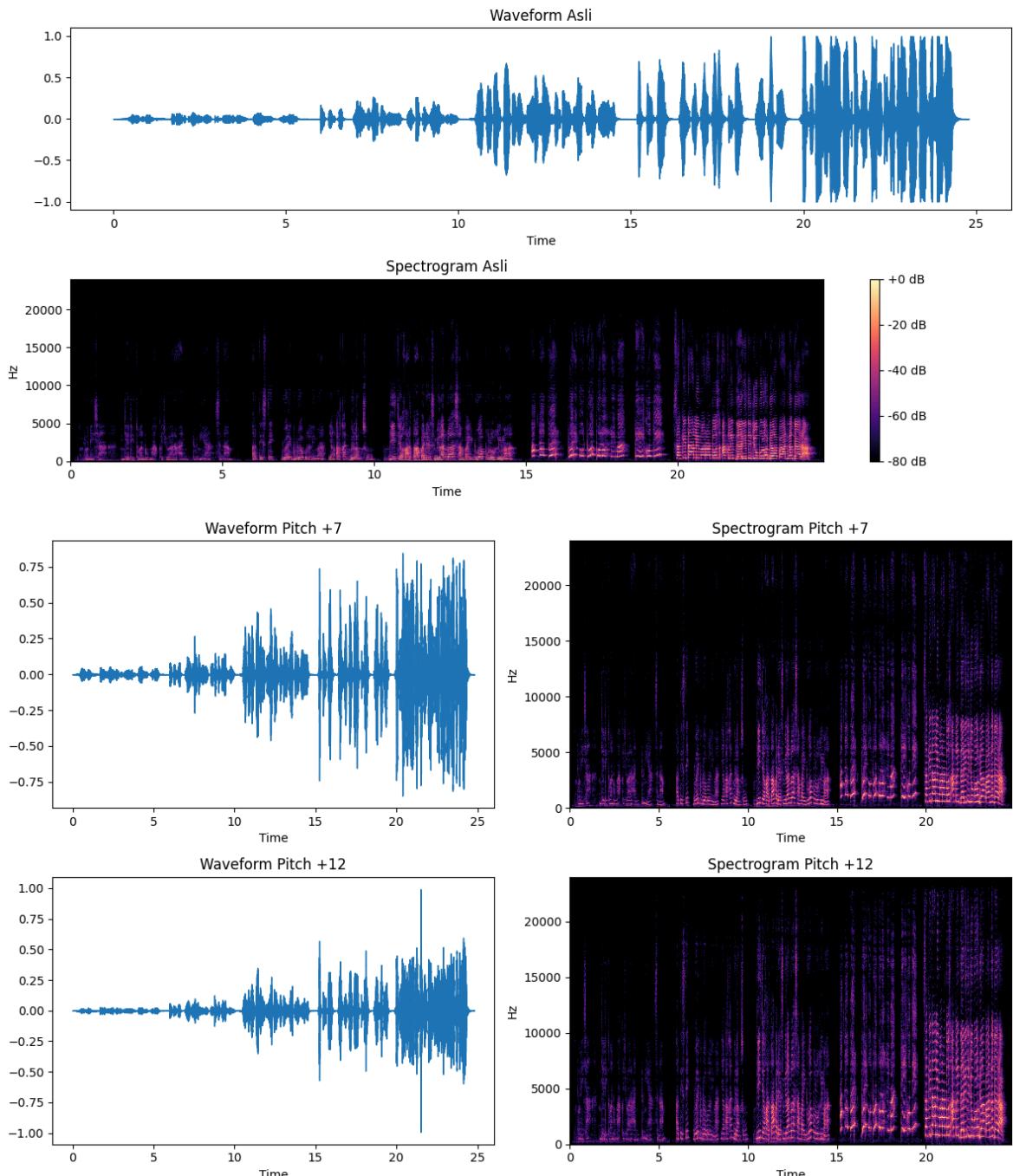
# 7. Putar hasil langsung di notebook
print("Audio Asli:")
display(Audio(file_path))

print("Pitch +7:")
display(Audio("Audio_Pitch7.wav"))

print("Pitch +12:")
display(Audio("Audio_Pitch12.wav"))

print("Gabungan (+7 dan +12):")
display(Audio("Audio_Chipmunk_Combined.wav"))
```

Durasi: 24.79 detik | Sample Rate: 48000 Hz



Audio Asli:

▶ 0:00 / 0:24 ━━ 🔊 ⋮

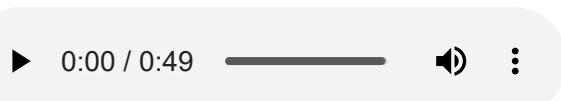
Pitch +7:

▶ 0:00 / 0:24 ━━ 🔊 ⋮

Pitch +12:

▶ 0:00 / 0:24 ━━ 🔊 ⋮

Gabungan (+7 dan +12):



1. proses pitch shifting yang saya lakukan pada audio1.wav adalah menaikkan nada suara sehingga menghasilkan suara seperti chipmunk. prosesnya dilakukan menggunakan effect pitch shift di librosa dan menggunakan dua parameter pergeseran nada, pakai +7 dan +12 semitone. perubahan pitch ini dilakukan tanpa mengubah durasi suaranya menggunakan metode phase vocoder, yang bekerja dengan cara menggeser frekuensi fundamental dan harmonik ke arah yang lebih tinggi. hasilnya pada pitch +7 suara saya terdengar lebih ringan dan nyaring tetapi masih natural, sedangkan pitch +12 suara saya terdengar jauh lebih tinggi dan lebih tipis menyerupai chipmunk.
2. parameter yang saya gunakan yaitu: y: sinyal audio input sr: sampel rate audio n_steps: jumlah pergeseran nada(Semitone), yaitu +7 dan +12 metode yang digunakan adalah phase vocoder, berfungsi untuk mengubah frekuensi tanpa memengaruhi durasi suara.
3. perbedaan representasi visual pada waveform hasil pitch shifting adalah dia terlihat lebih rapat karena frekuensinya meningkat. kemudian pada spectrogram, energi suaranya bergeser ke frekuensi yang lebih tinggi dan menunjukkan warna terang di bagian atasnya. berarti ini menunjukkan pitchnya meningkat daripada suara aslinya.
4. pengaruh terhadap kualitas dan kejernihan suara menggunakan pitch yaitu suaranya lebih nyaring dan tipis. pada pitch +7 dia masih cukup natural tapi pada pitch +12 suaranya menjadi seperti chipmunk, lebih tinggi lagi dan kurang jelas.

Soal 4: Audio Processing Chain

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
from IPython.display import Audio, HTML, display
import os
import matplotlib
from scipy.signal import butter, filtfilt, spectrogram
from IPython.display import Audio, display
from pydub import AudioSegment

%matplotlib inline

import librosa.display
from pydub import AudioSegment, effects, silence
import pyloudnorm as pyln

print("Library versions:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")
```

Library versions:
 NumPy: 2.2.6
 Matplotlib: 3.10.5
 Librosa: 0.11.0

```
In [ ]: # 1. Load Audio
input_path = "..\Soal3\Audio_Chipmunk_Combined.wav"
out_path = "Audio_Chipmunk_ProCESSED_Final.wav"

y, sr = librosa.load(input_path, sr=None)
print(f"Durasi awal: {librosa.get_duration(y=y, sr=sr):.2f} detik | Sample Rate: {sr:.2f} Hz")

# Visualisasi Sebelum Normalisasi
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
librosa.display.waveform(y, sr=sr)
plt.title("Waveform Sebelum Normalisasi")

plt.subplot(2, 1, 2)
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title("Spectrogram Sebelum Normalisasi")
plt.colorbar(format="%+2.0f dB")
plt.tight_layout()
plt.show()

# 2. Konversi ke AudioSegment
audio = AudioSegment.from_file(input_path)
audio = audio.set_frame_rate(44100).set_sample_width(2).set_channels(1)

# 3. Equalizer
audio = audio.high_pass_filter(200)    # buang frekuensi rendah
audio = audio.low_pass_filter(5000)     # batasi frekuensi tinggi
```

```

# 4. Gain & Fade
audio = audio.apply_gain(-2)           # kurangi sedikit gain
audio = audio.fade_in(100).fade_out(300) # transisi halus di awal & akhir

# 5. Normalisasi berdasarkan LUFS (-16 LUFS target)
temp_path = "temp_audio.wav"
audio.export(temp_path, format="wav")

data, rate = sf.read(temp_path)
meter = pyln.Meter(rate)
loudness = meter.integrated_loudness(data)

target_loudness = -16.0
loudness_diff = target_loudness - loudness
audio = audio.apply_gain(loudness_diff)

print(f'Loudness awal: {loudness:.2f} LUFS → Sesudah disesuaikan ke {target_loudness:.2f} LUFS')

# 6. Compression
audio = effects.compress_dynamic_range(
    audio, threshold=-20.0, ratio=3.0, attack=5, release=50
)

# 7. Noise Gate + Silence Trimming
# Pisahkan bagian yang memiliki suara (non-silence)
audio_chunks = silence.split_on_silence(
    audio,
    min_silence_len=200,      # minimal durasi diam 200ms
    silence_thresh=-40        # ambang diam
)

# Jika ada hasil potongan, gabungkan ulang semuanya
if len(audio_chunks) > 0:
    trimmed_audio = audio_chunks[0]
    for chunk in audio_chunks[1:]:
        trimmed_audio += chunk
    audio = trimmed_audio

# Tambahkan fade halus untuk menghindari pop
audio = audio.fade_in(50).fade_out(50)

# 8. Simpan hasil akhir
audio.export(out_path, format="wav")
print(f'Hasil akhir disimpan ke: {out_path}')

# 9. Visualisasi Sesudah Normalisasi
y2, sr2 = librosa.load(out_path, sr=None)

plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
librosa.display.waveform(y2, sr=sr2)
plt.title("Waveform Setelah Normalisasi")

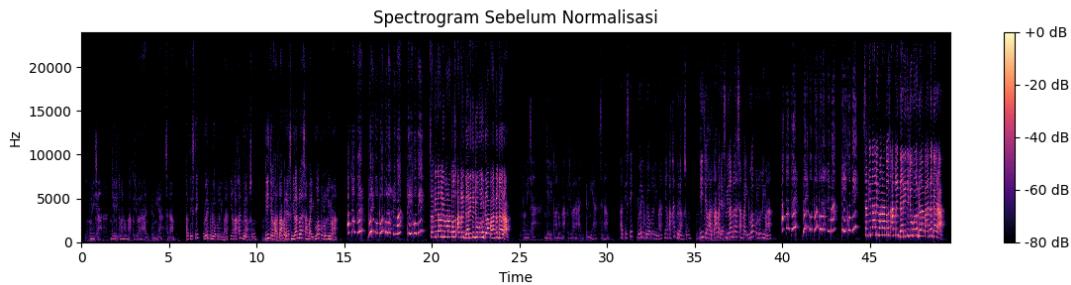
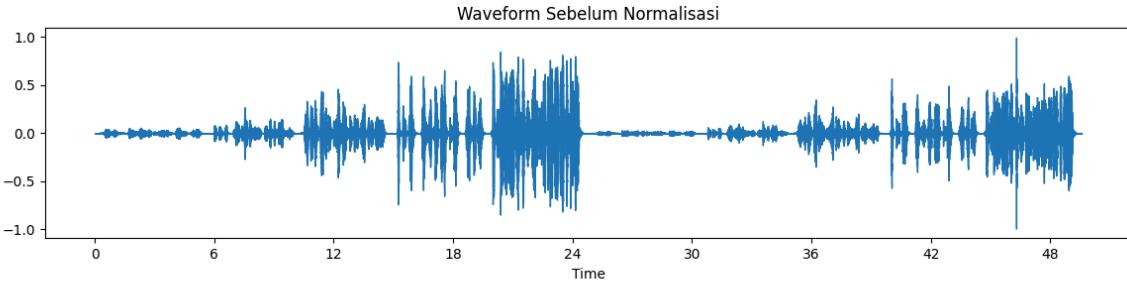
plt.subplot(2, 1, 2)
D2 = librosa.amplitude_to_db(np.abs(librosa.stft(y2)), ref=np.max)
librosa.display.specshow(D2, sr=sr2, x_axis='time', y_axis='hz', cmap='magma')
plt.title("Spectrogram Setelah Normalisasi")
plt.colorbar(format="%+2.0f dB")
plt.tight_layout()
plt.show()

```

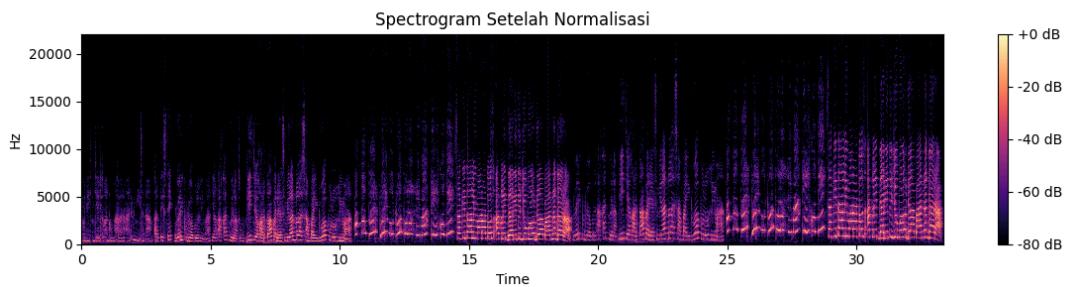
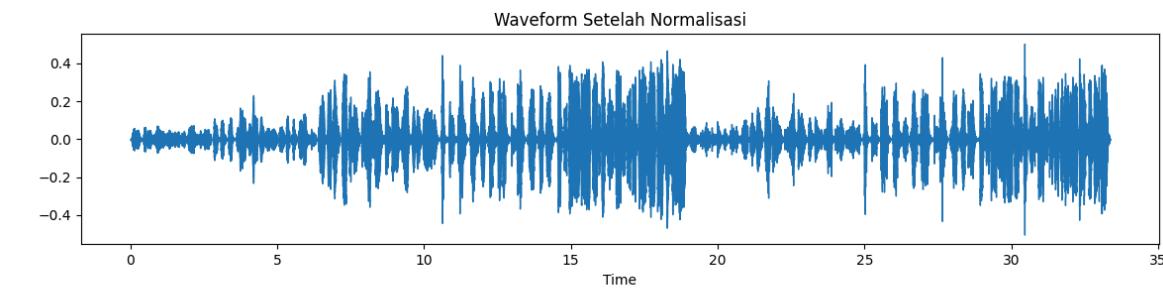
```
# 10. Putar audio sebelum dan sesudah
print("Audio Sebelum Processing:")
display(Audio(input_path))

print("Audio Setelah Processing:")
display(Audio(out_path))
```

<>:2: SyntaxWarning: invalid escape sequence '\S'
 <>:2: SyntaxWarning: invalid escape sequence '\S'
 C:\Users\Elma\AppData\Local\Temp\ipykernel_5196\2451974657.py:2: SyntaxWarning: i
 nvalid escape sequence '\S'
 input_path = "..\Soal3\Audio_Chipmunk_Combined.wav"
 Durasi awal: 49.58 detik | Sample Rate: 48000 Hz



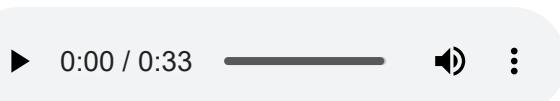
Loudness awal: -21.59 LUFS → Sesudah disesuaikan ke -16.0 LUFS
 Hasil akhir disimpan ke: Audio_Chipmunk_Processed_Final.wav



Audio Sebelum Processing:



Audio Setelah Processing:



1. Perubahan dinamika yang terjadi setelah melalui proses equalizer, gain/fade, normalization, compression, noise gate, dan silence trimming suaranya menjadi lebih seimbang, bagian yang pelan bisa terdengar, dan bagian yang keras tidak terlalu keras. kemudian durasinya juga menjadi lebih singkat dari 49 detik menjadi 33 detik karena jeda yang ada pada suara di potong.
2. perbedaan antara normalisasi peak dan normalisasi LUFS adalah kalau normalisasi peak dia menyesuaikan puncak tertinggi dari gelombang suara supaya tidak melebihi batas tertentu (biasanya 0 dB) tapi kenyaringannya/loudnessnya tidak selalu berubah. kalau normalisasi LUFS dia mengatur kenyaringan rata-rata supaya sesuai dengan standar pendengaran manusia (contohnya -16 LUFS). LUFS lebih akurat untuk memastikan volume antar file terdengar konsisten.
3. kualitas suara setelah proses normalisasi dan loudness optimization yaitu suaranya lebih seimbang antara yg pelan dan keras, kemudian lebih jernih sehingga lebih enak untuk di dengar.
4. kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara adalah: kelebihan:
 - volume suara jadi lebih konsisten
 - lebih enak untuk didengar
 - cocok digunakan untuk distribusi digital (youtube, spotify, dll) yang memiliki standar loudness
 - kalau terlalu kuat, dinamika suara aslinya bisa hilang
 - adanya resiko distortion/suara terdengar datar karena dikompres berlebihan

Soal 5: Music Analysis dan Remix

In [2]:

```

import librosa
import numpy as np
import librosa.display
import matplotlib.pyplot as plt

# 1. Load Audio
path_sedih = "../Data/sedih.wav"
path_ceria = "../Data/ceria.wav"

y_sedih, sr_sedih = librosa.load(path_sedih, sr=None)
y_ceria, sr_ceria = librosa.load(path_ceria, sr=None)

print(f"Audio sedih: {len(y_sedih)/sr_sedih:.2f} detik | Sample Rate: {sr_sedih}")
print(f"Audio ceria: {len(y_ceria)/sr_ceria:.2f} detik | Sample Rate: {sr_ceria}")

# 2. Deteksi Tempo (BPM)
tempo_sedih, _ = librosa.beat.beat_track(y=y_sedih, sr=sr_sedih)
tempo_ceria, _ = librosa.beat.beat_track(y=y_ceria, sr=sr_ceria)

# pastikan hasil berupa float
tempo_sedih = float(tempo_sedih)
tempo_ceria = float(tempo_ceria)

# 3. Estimasi Key / Nada Dasar
chroma_sedih = librosa.feature.chroma_stft(y=y_sedih, sr=sr_sedih)
chroma_ceria = librosa.feature.chroma_stft(y=y_ceria, sr=sr_ceria)

notes = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']

key_sedih = notes[int(np.argmax(np.mean(chroma_sedih, axis=1)))]
key_ceria = notes[int(np.argmax(np.mean(chroma_ceria, axis=1)))]

# 4. Tampilkan Hasil
print("\n==== HASIL ANALISIS AUDIO ===")
print(f"Lagu Sedih : {tempo_sedih:.1f} BPM | Perkiraan Nada Dasar: {key_sedih}")
print(f"Lagu Ceria : {tempo_ceria:.1f} BPM | Perkiraan Nada Dasar: {key_ceria}")

# 6. Visualisasi
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
librosa.display.specshow(chroma_sedih, x_axis='time', y_axis='chroma', cmap='copper')
plt.title(f"Chroma - Lagu Sedih ({key_sedih})")

plt.subplot(1, 2, 2)
librosa.display.specshow(chroma_ceria, x_axis='time', y_axis='chroma', cmap='copper')
plt.title(f"Chroma - Lagu Ceria ({key_ceria})")

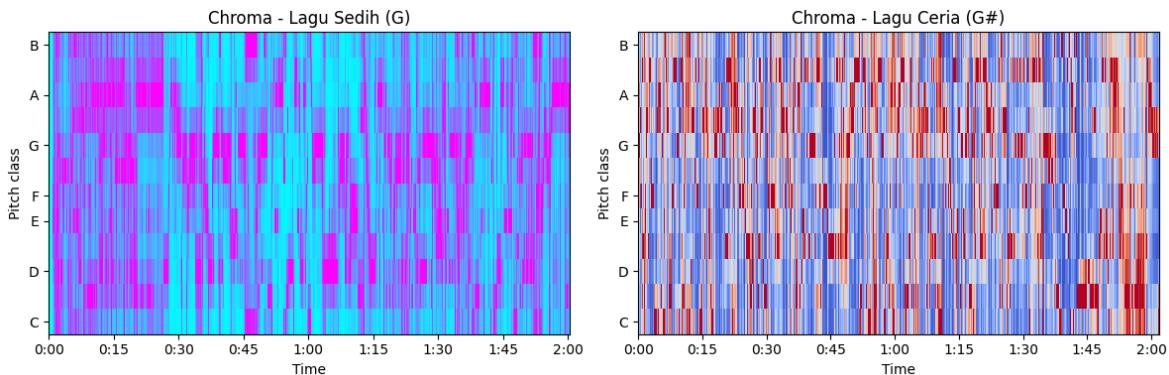
plt.tight_layout()
plt.show()

```

Audio sedih: 60.19 detik | Sample Rate: 44100

Audio ceria: 60.91 detik | Sample Rate: 44100

```
C:\Users\Elma\AppData\Local\Temp\ipykernel_7396\2413897524.py:21: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will err or in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
    tempo_sedih = float(tempo_sedih)
C:\Users\Elma\AppData\Local\Temp\ipykernel_7396\2413897524.py:22: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will err or in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
    tempo_ceria = float(tempo_ceria)
== HASIL ANALISIS AUDIO ==
Lagu Sedih : 161.5 BPM | Perkiraan Nada Dasar: G
Lagu Ceria : 90.7 BPM | Perkiraan Nada Dasar: G#
```



```
In [4]: import librosa
import librosa.display
import numpy as np
from pydub import AudioSegment
import matplotlib.pyplot as plt
import soundfile as sf
from IPython.display import Audio

# 1. Load Audio
path_sedih = "../Data/sedih.wav"
path_ceria = "../Data/ceria.wav"

y_sedih, sr_sedih = librosa.load(path_sedih, sr=None)
y_ceria, sr_ceria = librosa.load(path_ceria, sr=None)

# 2. Deteksi Tempo (BPM)
tempo_sedih, _ = librosa.beat.beat_track(y=y_sedih, sr=sr_sedih)
tempo_ceria, _ = librosa.beat.beat_track(y=y_ceria, sr=sr_ceria)
tempo_sedih, tempo_ceria = float(tempo_sedih), float(tempo_ceria)

# 3. Estimasi Nada Dasar (Key)
chroma_sedih = librosa.feature.chroma_stft(y=y_sedih, sr=sr_sedih)
chroma_ceria = librosa.feature.chroma_stft(y=y_ceria, sr=sr_ceria)
notes = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']
key_sedih = notes[int(np.argmax(np.mean(chroma_sedih, axis=1)))]
key_ceria = notes[int(np.argmax(np.mean(chroma_ceria, axis=1)))]

print("== INFO AWAL ==")
print(f'Lagu Sedih : {tempo_sedih:.1f} BPM | Key: {key_sedih}')
print(f'Lagu Ceria : {tempo_ceria:.1f} BPM | Key: {key_ceria}')

# 4. Time Stretch (Samakan Tempo)
target_bpm = np.mean([tempo_sedih, tempo_ceria])
```

```

rate_sedih = target_bpm / tempo_sedih
rate_ceria = target_bpm / tempo_ceria

def stretch_audio(y, rate):
    D = librosa.stft(y)
    D_stretch = librosa.phase_vocoder(D, rate=rate)
    return librosa.istft(D_stretch)

y_sedih_stretch = stretch_audio(y_sedih, rate_sedih)
y_ceria_stretch = stretch_audio(y_ceria, rate_ceria)

print(f"\nTempo disamakan ke sekitar {target_bpm:.1f} BPM")

# 5. Pitch Shift (Samakan Key)
idx_sedih = notes.index(key_sedih)
idx_ceria = notes.index(key_ceria)
shift_steps = idx_ceria - idx_sedih

y_sedih_pitch = librosa.effects.pitch_shift(y=y_sedih_stretch, sr=sr_sedih, n_st
print(f"Pitch lagu sedih digeser {shift_steps:+} semitone → disamakan ke {key_ce

# 6. Simpan hasil sementara
sf.write("sedih_sync.wav", y_sedih_pitch, sr_sedih)
sf.write("ceria_sync.wav", y_ceria_stretch, sr_ceria)

# 7. Crossfade Remix
audio_sedih = AudioSegment.from_file("sedih_sync.wav")
audio_ceria = AudioSegment.from_file("ceria_sync.wav")

min_len = min(len(audio_sedih), len(audio_ceria))
audio_sedih = audio_sedih[:min_len]
audio_ceria = audio_ceria[:min_len]

crossfade_ms = 5000 # 5 detik
remix = audio_sedih.append(audio_ceria, crossfade=crossfade_ms)

# 8. Filter Tambahan
remix = remix.low_pass_filter(9000)
remix = remix.fade_in(2000).fade_out(2000)

# 9. Simpan hasil remix
remix.export("Remix_Final.wav", format="wav")
print("\n✓ Remix selesai dan disimpan sebagai 'Remix_Final.wav'")

# 10. Visualisasi Waveform
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
librosa.display.waveshow(y_sedih_pitch, sr=sr_sedih, color='b')
plt.title("Waveform Sedih (setelah Stretch + Pitch Shift)")
plt.subplot(1, 2, 2)
librosa.display.waveshow(y_ceria_stretch, sr=sr_ceria, color='orange')
plt.title("Waveform Ceria (setelah Stretch)")
plt.tight_layout()
plt.show()

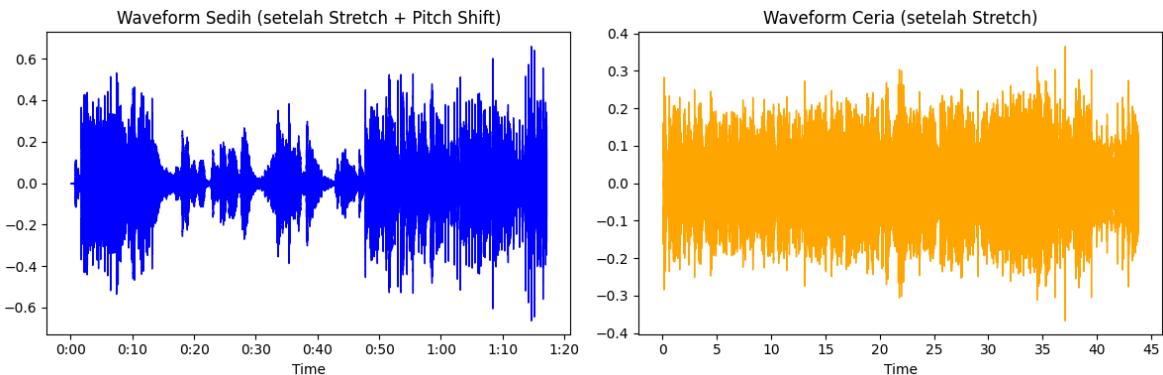
Audio("Remix_Final.wav")

```

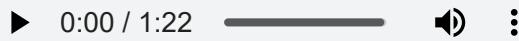
```
C:\Users\Elma\AppData\Local\Temp\ipykernel_7396\4213629070.py:20: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will err or in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
    tempo_sedih, tempo_ceria = float(tempo_sedih), float(tempo_ceria)
== INFO AWAL ==
Lagu Sedih : 161.5 BPM | Key: G
Lagu Ceria : 90.7 BPM | Key: G#
```

Tempo disamakan ke sekitar 126.1 BPM
 Pitch lagu sedih digeser +1 semitone → disamakan ke G#

Remix selesai dan disimpan sebagai 'Remix_Final.wav'



Out[4]:



- proses dan parameter yang digunakan pada saat menganalisis karakter musik dasar dari dua lagu sedih dan ceria digunakan untuk mengetahui tempo(BPM) dan nada dasar(key). pertama menggunakan librosa.load() untuk membaca file audio. kemudian librosa.beat.beat_track() untuk mendeteksi kecepatan iramanya dalam satuan BPM. kemudian fitur chroma digunakan untuk mendeteksi nada dominan dari lagu (intensitas energi nada musik). kemudian librosa.display.specshow() untuk memvisualisasikan BPM dan nada dasar tiap lagu dalam bentuk chroma plot. kemudian untuk remix lagu langkah awal sama menampilkan audio kemudian melakukan time strech dengan fungsi strech_audio() untuk menyamakan tempo menggunakan target_bpm dimana ini digunakan untuk mencari rata-rata dari kedua tempo lagu, kemudian dihitung rasio kecepatannya (rate) untuk menyesuaikan durasi. kemudian pitch_shift() untuk menggeser nada dasar lagu sedih agar sama dengan lagu ceria. kemudian untuk menggabungkan audionya menggunakan pydub. dan menambahkan filter low pass untuk menghasilkan suara yang lebih lembut, fadein dan fade out untuk membuat musik perlahan di awal dan di akhir.

In []:

```
# 1. Load hasil remix
remix_path = "Remix_Final.wav"
y_remix, sr_remix = librosa.load(remix_path, sr=None)

print(f"Durasi Remix : {len(y_remix)/sr_remix:.2f} detik | Sample Rate: {sr_remix}")

# 2. Visualisasi Waveform
plt.figure(figsize=(14, 4))
librosa.display.waveshow(y_remix, sr=sr_remix, color='purple')
```

```

plt.title("Waveform Hasil Remix")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(alpha=0.3)
plt.show()

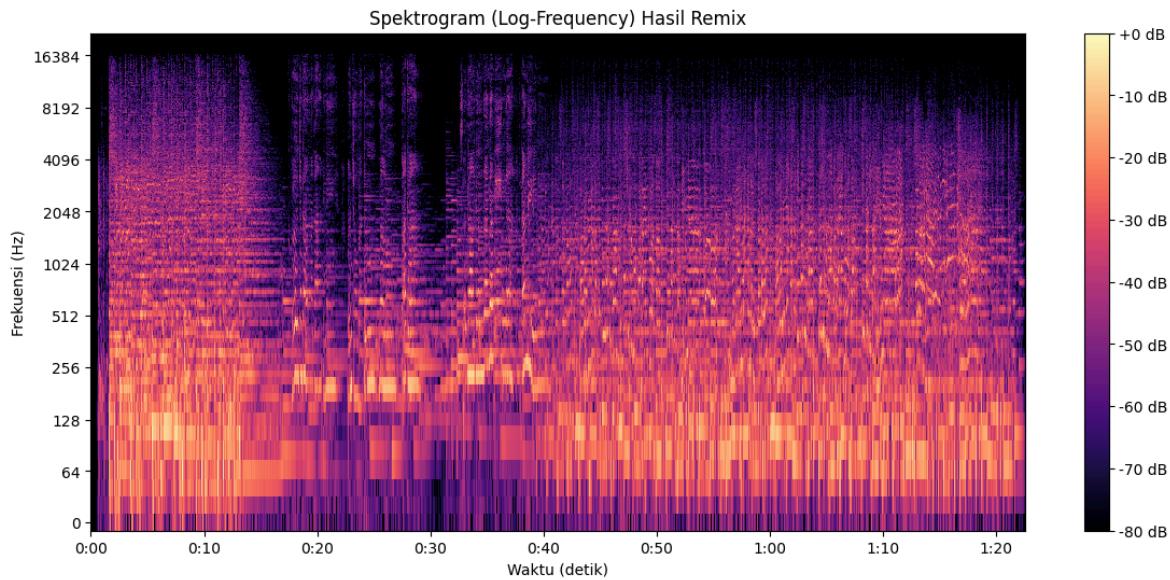
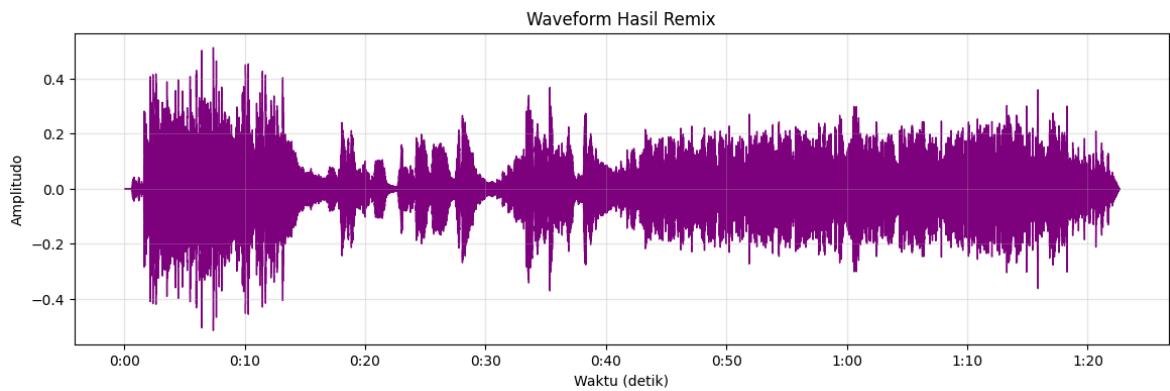
# 3. Visualisasi Spektrogram
plt.figure(figsize=(14, 6))

# Konversi ke spektrogram dB
S = librosa.stft(y_remix)
S_db = librosa.amplitude_to_db(np.abs(S), ref=np.max)

librosa.display.specshow(S_db, sr=sr_remix, x_axis='time', y_axis='log', cmap='magma')
plt.colorbar(format="%+2.0f dB")
plt.title("Spektrogram (Log-Frequency) Hasil Remix")
plt.xlabel("Waktu (detik)")
plt.ylabel("Frekuensi (Hz)")
plt.show()

```

Durasi Remix : 82.61 detik | Sample Rate: 44100



Hasil dari remix kedua audio sedih dan ceria telah berhasil di gabungkan. tempo dan nada dasarnya telah disamakan, dan transisi antar lagu mulus tidak terlalu jomplang dengan menggunakan efek crossfading. secara visual dari waveform dan spectogram tidak ada lonjakan energi yang ekstrim, sehingga hasil remix dari kedua lagu ini cukup stabil

LAMPIRAN:

- GPT : <https://chatgpt.com/c/68ef4ddd-c1a4-8320-b754-702e51869008>
- GPT : <https://chatgpt.com/share/68ef63ac-3bf4-8005-b7a8-7080f4140463>
- YOUTUBE : https://www.youtube.com/results?search_query=ketika+push+ke+github+filenya+sorry+this+is+too+big+to+display
- credit : <https://youtu.be/fO1iJqBdXrc?si=Gk8VXAwqUXbifanY>
- credit : https://youtu.be/wKimR0fyeyo?si=mJx_6Ze78qnZcjq-