



Hands-On Audio Processing

Nama: Fawwaz Abhitah Sugiarto

NIM: 122140014

Mata Kuliah: IF4021 - Multimedia Information Processing

Link Github Repository: <https://github.com/FawwazAbhitah-122140014/Hands-on-audio>

Deskripsi Tugas

Tugas ini dirancang untuk menguji pemahaman mahasiswa terhadap konsep-konsep fundamental dalam pemrosesan audio digital, termasuk manipulasi sinyal audio, filtering, pitch shifting, normalisasi, dan teknik remix audio. Mahasiswa diharapkan dapat menerapkan teori yang telah dipelajari dalam praktik langsung menggunakan Python dan pustaka pemrosesan audio.

In [172...

```
import numpy as np
import os
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
import scipy.signal
from IPython.display import Audio, HTML, display

print("Versi Library:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {plt.matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")
print(f"SciPy: {scipy.__version__}")
```

Versi Library:

NumPy: 2.2.6

Matplotlib: 3.10.7

Librosa: 0.11.0

SciPy: 1.15.3

In [173...

```
audio_path = os.path.join("data", "Audio1.wav")

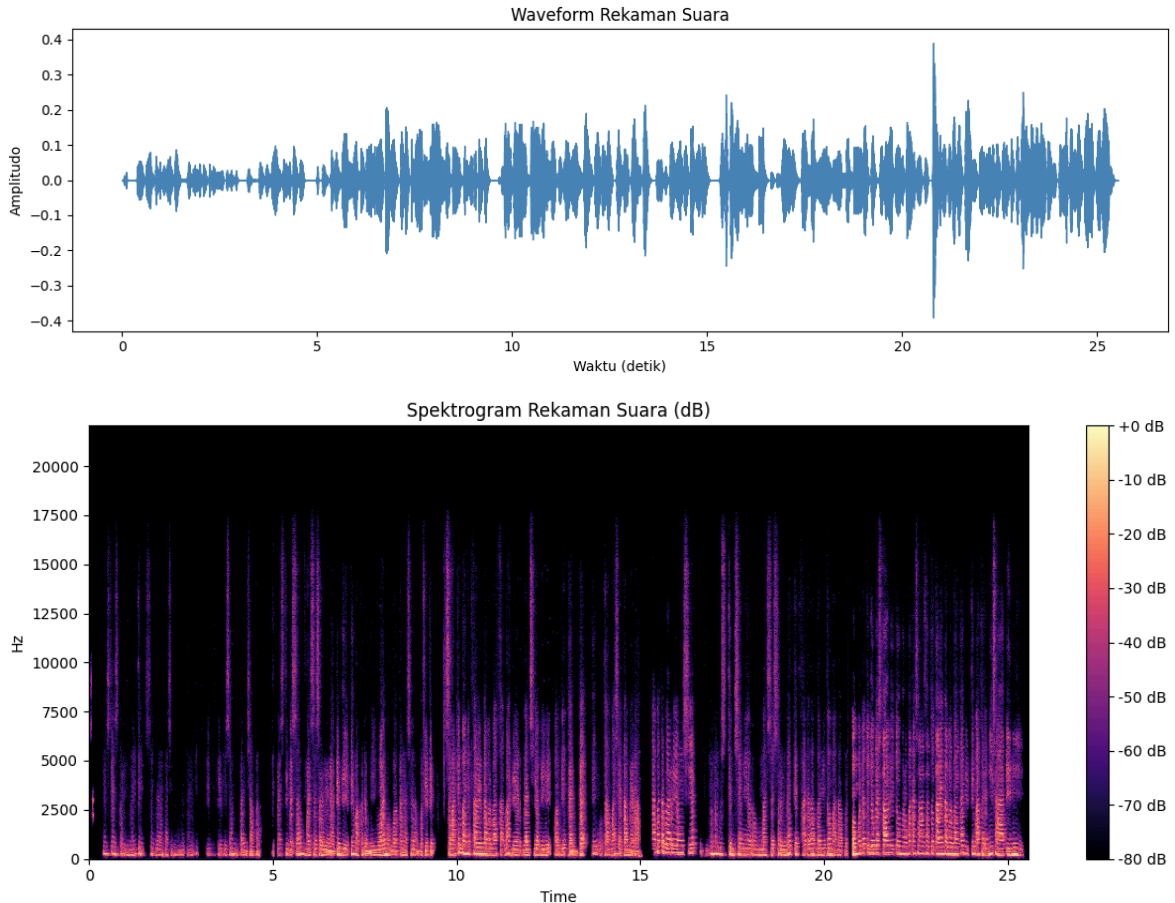
# Memutar audio
print("Rekaman Audio:")
display(Audio(y, rate=sr))

# Waveform
plt.figure(figsize=(12, 4))
librosa.display.waveshow(y, sr=sr, color='steelblue')
plt.title("Waveform Rekaman Suara")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.tight_layout()
plt.show()
```

```
# Spektrogram
plt.figure(figsize=(12, 5))
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("Spektrogram Rekaman Suara (dB)")
plt.colorbar(format="%+2.0f dB")
plt.tight_layout()
plt.show()
```

Rekaman Audio:

▶ 0:00 / 0:25 ———— 🔊 ⋮



Penjelasan hasil visualisasi diatas:

Pada waveform diatas pada detik 0 sampai 5 terlihat amplitudo yang rendah, karena audio tersebut direkam dengan suara yang pelan dan agak berisik sedikit. kemudian pada detik 5 sampai 10 amplitudo tersebut meningkat daripada sebelumnya, bisa dilihat amplitudonya yang bernilai lebih besar ketimbang sebelumnya. selanjutnya pada detik 10 sampai 15 amplitudo hampir sama dengan detik 5 sampai 10, tetapi pada detik 10 sampai 15 amplitudo konsisten dan ada yang naik signifikan. detik 15 sampai 20 amplitudo diawali dengan naik kemudian menurun. terakhir pada detik 20 sampai 25 amplitudo memiliki nilai yang lebih besar dibandingkan lainnya, hal ini dikarenakan pada detik tersebut terdapat suara yang teriak.

Pada spektrogram diatas terlihat frekuensi yang dominan pada audio tersebut berada pada kisaran 0 Hz hingga 2000 Hz, hal ini dikarenakan pada rentang frekuensi tersebut memiliki intensitas warna yang lebih terang dibandingkan dengan frekuensi lainnya.

Selain itu, terdapat juga frekuensi yang lebih tinggi hingga sekitar 8000 Hz, namun intensitasnya lebih rendah.

In [174...

```
# Path file audio
audio_path = os.path.join("data", "Audio1.wav")

# Muat audio asli
y, sr = librosa.load(audio_path, sr=None)
print(f"Sample rate asli: {sr} Hz")
print(f"Durasi audio asli: {len(y)/sr:.2f} detik")

# Resampling ke 12040 Hz
target_sr = 12040
y_resampled = librosa.resample(y, orig_sr=sr, target_sr=target_sr)
print(f"\nSample rate setelah resampling: {target_sr} Hz")
print(f"Durasi audio hasil resampling: {len(y_resampled)/target_sr:.2f} detik")

# Waveform
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
librosa.display.waveshow(y, sr=sr, color='steelblue')
plt.title(f"Waveform Asli ({sr} Hz)")
plt.ylabel("Amplitudo")

plt.subplot(2, 1, 2)
librosa.display.waveshow(y_resampled, sr=target_sr, color='tomato')
plt.title(f"Waveform Setelah Resampling ({target_sr} Hz)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.tight_layout()
plt.show()

# Spektrogram
plt.figure(figsize=(12, 8))

# Spektrogram asli
plt.subplot(2, 1, 1)
D1 = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D1, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title(f"Spektrogram Asli ({sr} Hz)")
plt.colorbar(format="%+2.0f dB")

# Spektrogram hasil resampling
plt.subplot(2, 1, 2)
D2 = librosa.amplitude_to_db(np.abs(librosa.stft(y_resampled)), ref=np.max)
librosa.display.specshow(D2, sr=target_sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title(f"Spektrogram Setelah Resampling ({target_sr} Hz)")
plt.colorbar(format="%+2.0f dB")

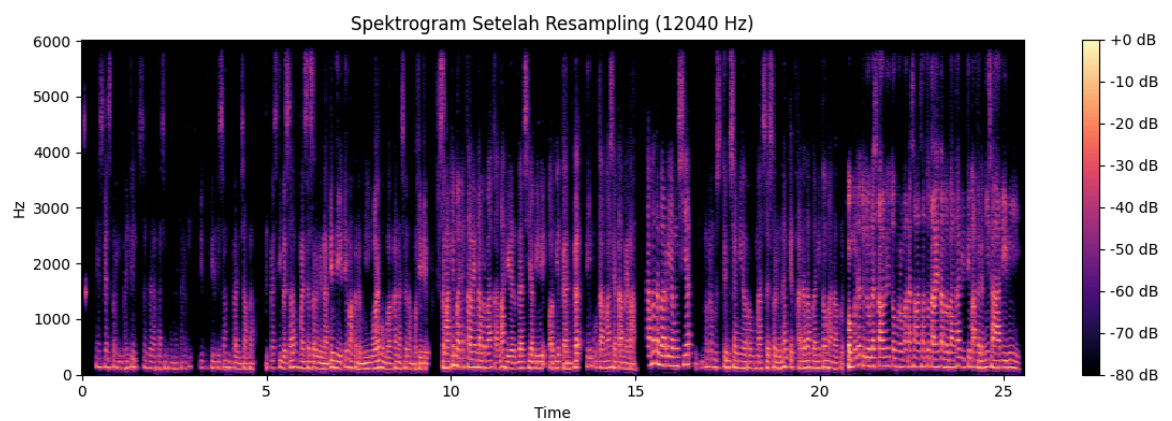
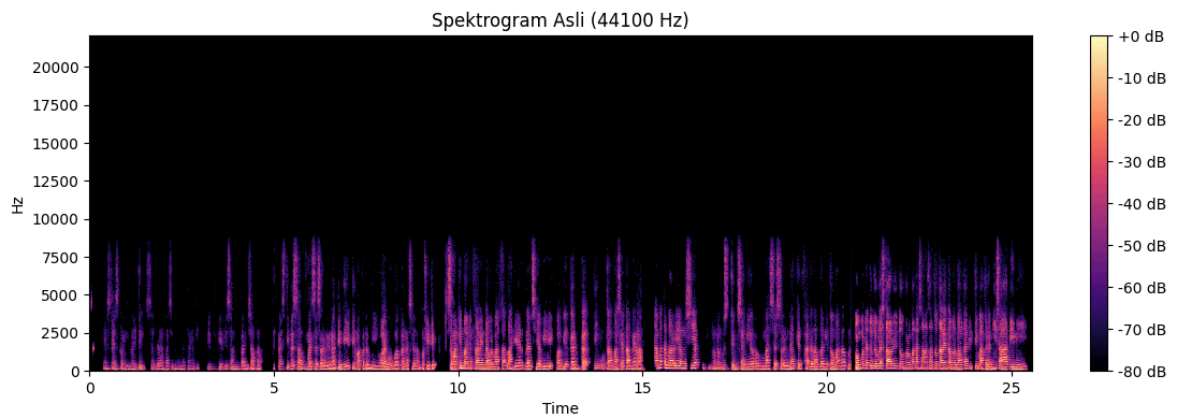
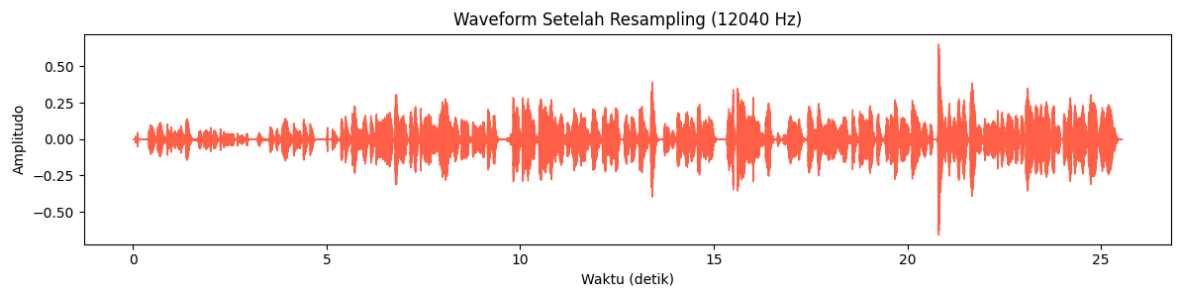
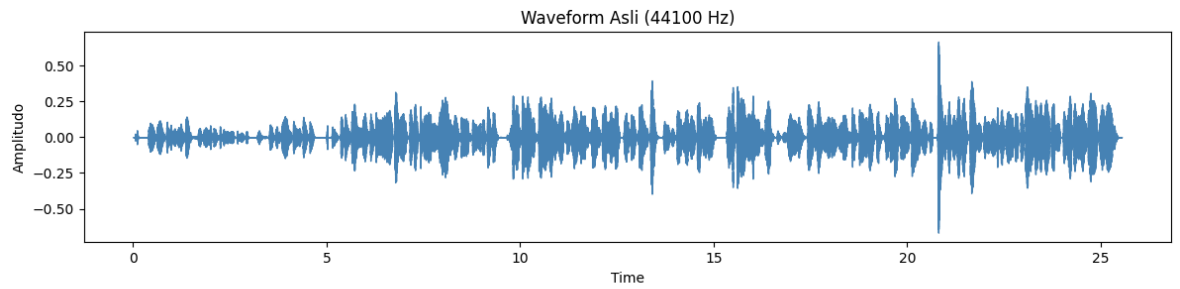
plt.tight_layout()
plt.show()

# Menampilkan audio
print("\n🔊 Audio Asli:")
display(Audio(y, rate=sr))

print("\n🔊 Audio Setelah Resampling (12040 Hz):")
display(Audio(y_resampled, rate=target_sr))
```

Sample rate asli: 44100 Hz
Durasi audio asli: 25.54 detik

Sample rate setelah resampling: 12040 Hz
Durasi audio hasil resampling: 25.54 detik



🔊 Audio Asli:

▶ 0:00 / 0:25 ————— 🔊 ⋮

🔊 Audio Setelah Resampling (12040 Hz):

▶ 0:00 / 0:25 ————— 🔊 ⋮

Penjelasan singkat

Dari hasil resampling yang dilakukan pada waveform tidak ada perubahan yang signifikan dikarenakan sample rate yang diubah, tetapi untuk spektrogram bisa dilihat yang tadinya mencapai 7500hz sekarang hanya mencapai 6000hz saja. kemudian dari hasil resampling yang saya lakukan contohnya pada detik 20 sampai 25 pada audio asli suara tersebut ada gemanya, sedangkan untuk yang sudah diresampling gempanya sudah berkurang atau hilang.

In [175...

```
audio_path = os.path.join("data", "Audio2.wav")
y, sr = librosa.load(audio_path, sr=None)
print(f"Sample rate: {sr} Hz | Durasi: {len(y)/sr:.2f} detik")

# fungsi filter umum
def butter_filter(y, sr, cutoff, btype="low", order=5):
    nyquist = 0.5 * sr
    normal_cutoff = cutoff / nyquist
    b, a = scipy.signal.butter(order, normal_cutoff, btype=btype, analog=False)
    return scipy.signal.filtfilt(b, a, y)

# Filter Low-pass (4000 Hz)
cutoff = 4000
y_low = butter_filter(y, sr, cutoff, btype="low")

# Spektrogram
plt.figure(figsize=(12, 6))

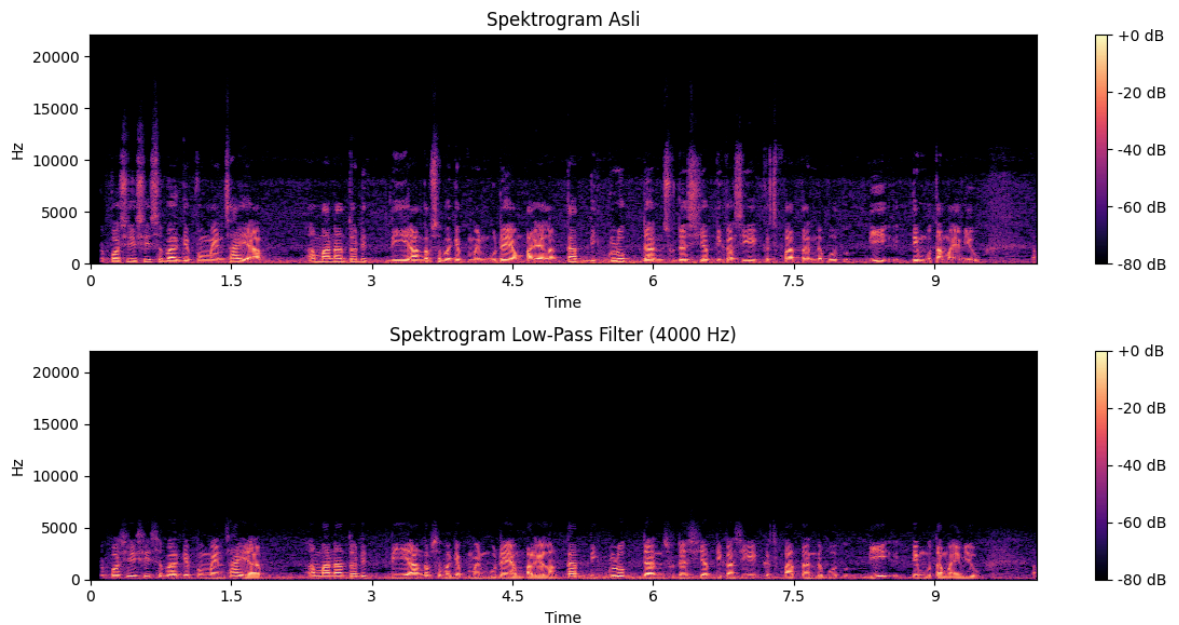
plt.subplot(2, 1, 1)
D_orig = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D_orig, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title("Spektrogram Asli")
plt.colorbar(format="%+2.0f dB")

plt.subplot(2, 1, 2)
D_low = librosa.amplitude_to_db(np.abs(librosa.stft(y_low)), ref=np.max)
librosa.display.specshow(D_low, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title(f"Spektrogram Low-Pass Filter ({cutoff} Hz)")
plt.colorbar(format="%+2.0f dB")

plt.tight_layout()
plt.show()

print("Audio Asli:")
display(Audio(y, rate=sr))
print(f"Audio Low-Pass ({cutoff} Hz):")
display(Audio(y_low, rate=sr))
```

Sample rate: 44100 Hz | Durasi: 10.08 detik



Audio Asli:



Audio Low-Pass (4000 Hz):



Penjelasan singkat low-pass

Pada filter low-pass bisa dilihat pada spectrogram mengalami penurunan frekuensi dikarenakan cutoff pada 5000 Hz sehingga frekuensi di atas 5000 Hz itu dihilangkan. Hal tersebut menyebabkan noise audio yang berkurang dan audio menjadi lebih mendep daripada sebelumnya.

```
In [176... audio_path = os.path.join("data", "Audio2.wav")
y, sr = librosa.load(audio_path, sr=None)
print(f"Sample rate: {sr} Hz | Durasi: {len(y)/sr:.2f} detik")

# Fungsi filter umum
def butter_filter(y, sr, cutoff, btype="high", order=5):
    nyquist = 0.5 * sr
    normal_cutoff = cutoff / nyquist
    b, a = scipy.signal.butter(order, normal_cutoff, btype=btype, analog=False)
    return scipy.signal.filtfilt(b, a, y)

# Filter High-pass (250 Hz)
cutoff = 250
y_high = butter_filter(y, sr, cutoff, btype="high")

# Spektrogram
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
D_orig = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D_orig, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title("Spektrogram Asli")
```



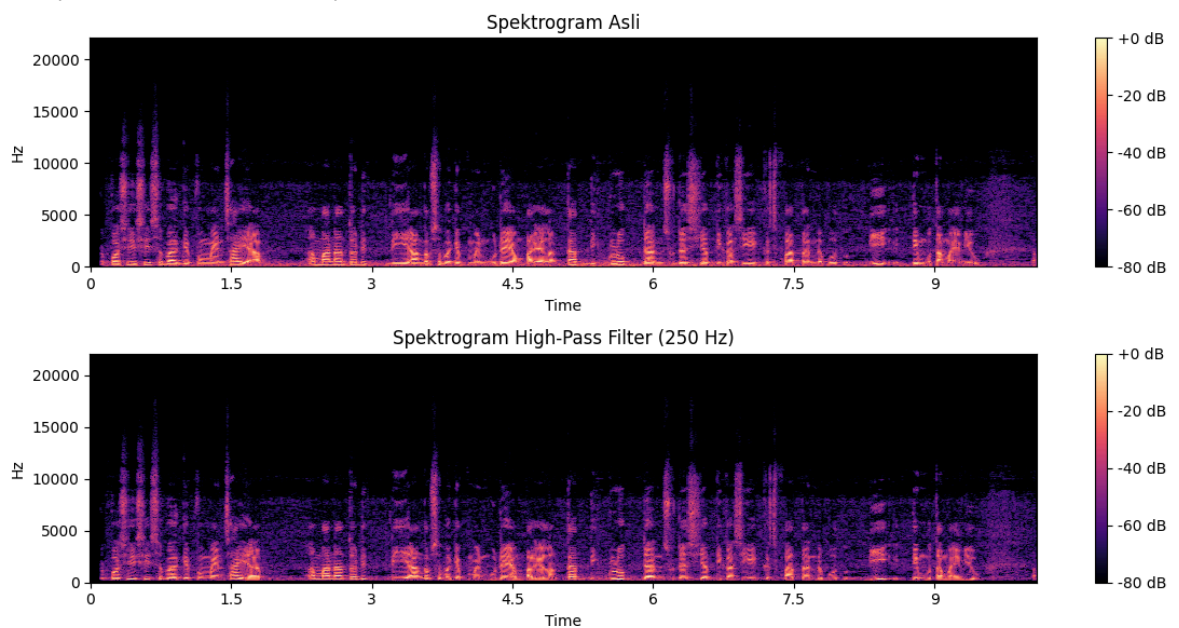
```
plt.colorbar(format="%+2.0f dB")

plt.subplot(2, 1, 2)
D_high = librosa.amplitude_to_db(np.abs(librosa.stft(y_high)), ref=np.max)
librosa.display.specshow(D_high, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title(f"Spektrogram High-Pass Filter ({cutoff} Hz)")
plt.colorbar(format="%+2.0f dB")

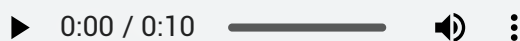
plt.tight_layout()
plt.show()

print("Audio Asli:")
display(Audio(y, rate=sr))
print(f"Audio High-Pass ({cutoff} Hz):")
display(Audio(y_high, rate=sr))
```

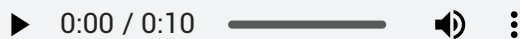
Sample rate: 44100 Hz | Durasi: 10.08 detik



Audio Asli:



Audio High-Pass (250 Hz):



Penjelasan singkat High-Pass

Pada filter high-pass bisa dilihat pada spectrogram bahwa frekuensi yang ada di bawah 250 hz hilang dikarenakan cutoff yang dipakai adalah 250 hz. Kemudian suara yang dihasilkan lebih bergema daripada audio aslinya.

In [177...

```
audio_path = os.path.join("data", "Audio2.wav")
y, sr = librosa.load(audio_path, sr=None)
print(f"Sample rate: {sr} Hz | Durasi: {len(y)/sr:.2f} detik")

# Fungsi filter umum
def butter_filter(y, sr, cutoff, btype="band", order=5):
```

```

nyquist = 0.5 * sr
normal_cutoff = np.array(cutoff) / nyquist
b, a = scipy.signal.butter(order, normal_cutoff, btype=btype, analog=False)
return scipy.signal.filtfilt(b, a, y)

# Filter Band-pass
cutoff_band = [100, 5000]
y_band = butter_filter(y, sr, cutoff_band, btype="band")

# Spektrogram
plt.figure(figsize=(12, 6))

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

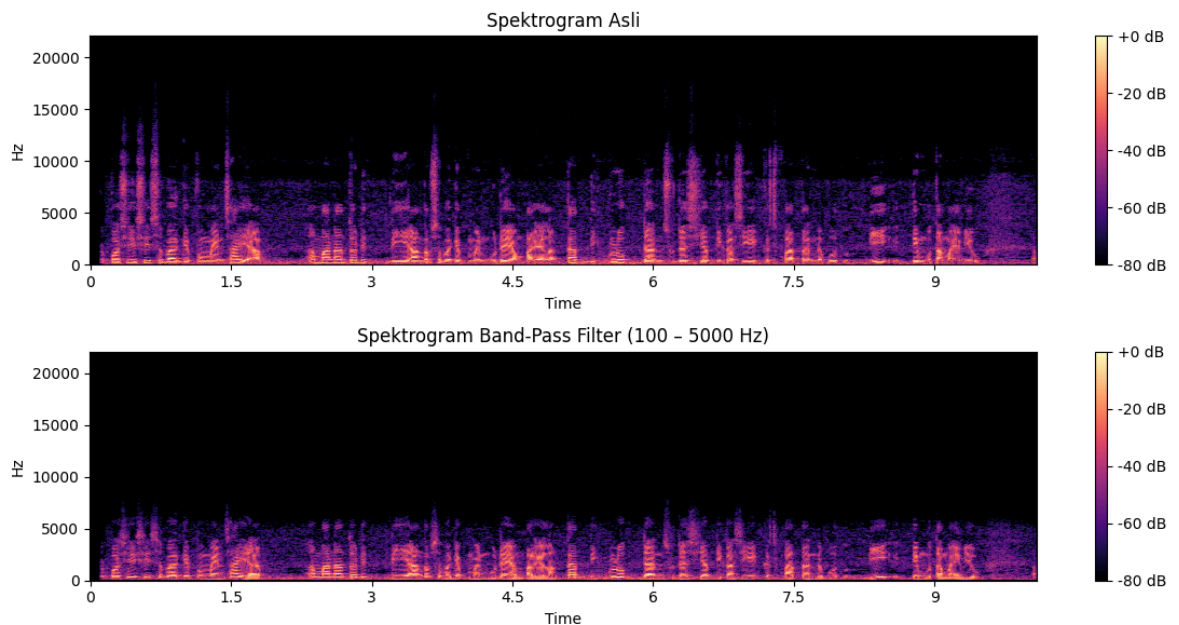
plt.subplot(2, 1, 2)
D_band = librosa.amplitude_to_db(np.abs(librosa.stft(y_band)), ref=np.max)
librosa.display.specshow(D_band, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title(f"Spektrogram Band-Pass Filter ({cutoff_band[0]} - {cutoff_band[1]} Hz")
plt.colorbar(format="%+2.0f dB")

plt.tight_layout()
plt.show()

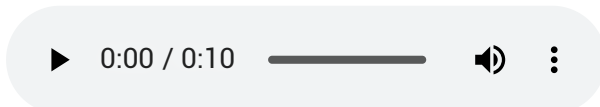
print("Audio Asli:")
display(Audio(y, rate=sr))
print(f"Audio Band-Pass ({cutoff_band[0]} - {cutoff_band[1]} Hz):")
display(Audio(y_band, rate=sr))

```

Sample rate: 44100 Hz | Durasi: 10.08 detik



Audio Asli:



Audio Band-Pass (100 - 5000 Hz):

Penjelasan singkat band-pass

Pada filter band-pass merupakan gabungan dari filter high-pass dan low-pass. Bisa dilihat pada spectrogram frekuensi yang dilewatkan hanya pada rentang antara 250 Hz hingga 5000 Hz, sedangkan frekuensi di bawah 250 Hz dan di atas 5000 Hz dihilangkan. Hal ini menyebabkan suara menjadi lebih fokus pada rentang frekuensi tersebut, mengurangi noise dari frekuensi rendah dan tinggi.

Penjelasan singkat 4 lainnya

- Jenis noise yang muncul pada rekaman Anda : Suara bunyi angin yang berasal dari sebuah kipas angin
- Filter mana yang paling efektif untuk mengurangi noise tersebut : Filter Band-Pass dikarenakan dapat menghilangkan 2 frekuensi yaitu frekuensi rendah dibawah 100 Hz dan frekuensi tinggi diatas 5000 Hz sehingga suara yang dihasilkan lebih jernih.
- Nilai cutoff yang memberikan hasil terbaik : Filter Band-Pass dengan cutoff rendah pada 100 Hz dan cutoff tinggi pada 5000 Hz
- Bagaimana kualitas suara (kejelasan ucapan) setelah proses filtering : Suara yang dihasilkan menjadi mendem dan mengurangi suara nafas.

In [178...

```
audio_path = os.path.join("data", "Audio1.wav")
y, sr = librosa.load(audio_path, sr=None)
print(f"Sample rate: {sr} Hz | Durasi: {len(y)/sr:.2f} detik")

# Pitch Shifting
y_chip7 = librosa.effects.pitch_shift(y=y, sr=sr, n_steps=7)
y_chip12 = librosa.effects.pitch_shift(y=y, sr=sr, n_steps=12)

# Waveform
plt.figure(figsize=(12, 6))

plt.subplot(3, 1, 1)
librosa.display.waveshow(y, sr=sr)
plt.title("Waveform Asli")

plt.subplot(3, 1, 2)
librosa.display.waveshow(y_chip7, sr=sr)
plt.title("Waveform Pitch Shift +7")

plt.subplot(3, 1, 3)
librosa.display.waveshow(y_chip12, sr=sr)
plt.title("Waveform Pitch Shift +12")

plt.tight_layout()
plt.show()

# Spektrogram
plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 1)
```

```

D_orig = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D_orig, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title("Spektrogram Asli")
plt.colorbar(format="%+2.0f dB")

plt.subplot(3, 1, 2)
D_chip7 = librosa.amplitude_to_db(np.abs(librosa.stft(y_chip7)), ref=np.max)
librosa.display.specshow(D_chip7, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title("Spektrogram +7")
plt.colorbar(format="%+2.0f dB")

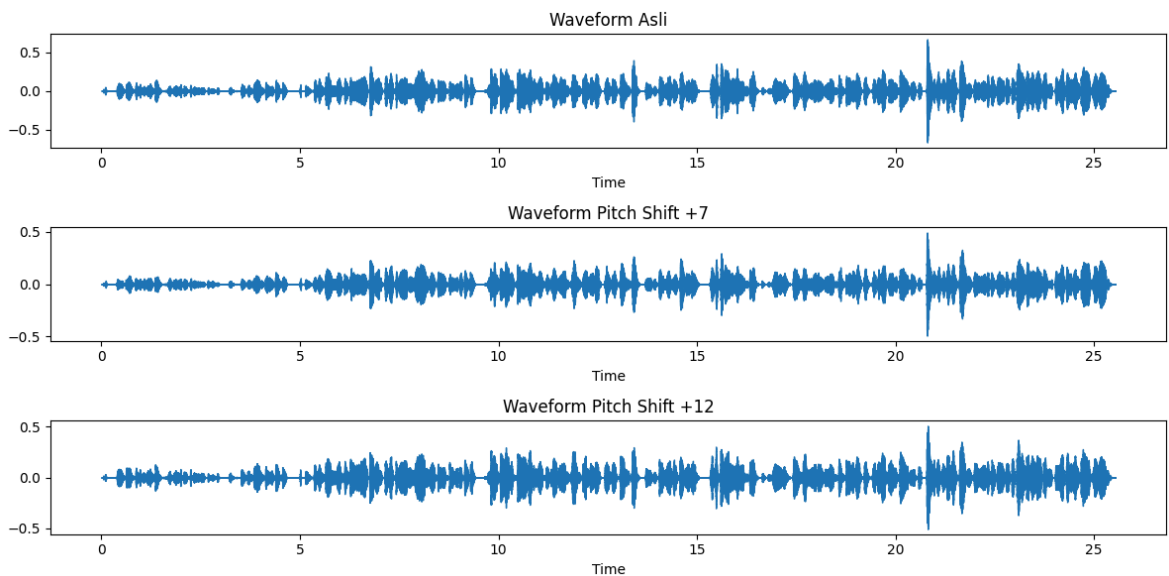
plt.subplot(3, 1, 3)
D_chip12 = librosa.amplitude_to_db(np.abs(librosa.stft(y_chip12)), ref=np.max)
librosa.display.specshow(D_chip12, sr=sr, x_axis="time", y_axis="hz", cmap="magma")
plt.title("Spektrogram +12")
plt.colorbar(format="%+2.0f dB")

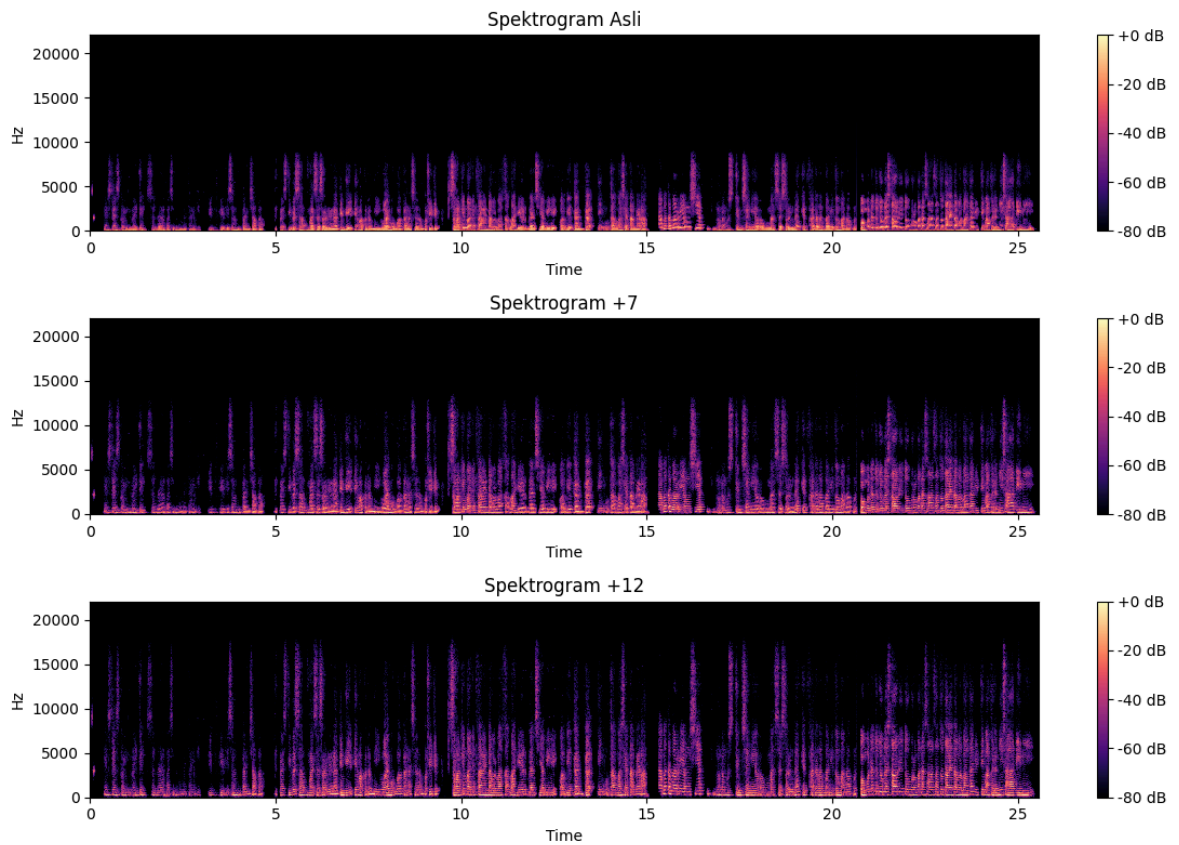
plt.tight_layout()
plt.show()

# perbandingan audio
print("Audio Asli:")
display(Audio(y, rate=sr))
print("Pitch Shift +7 :")
display(Audio(y_chip7, rate=sr))
print("Pitch Shift +12 :")
display(Audio(y_chip12, rate=sr))

```

Sample rate: 44100 Hz | Durasi: 25.54 detik





Audio Asli:

▶ 0:00 / 0:25 ————— 🔊 ⋮

Pitch Shift +7 :

▶ 0:00 / 0:25 ————— 🔊 ⋮

Pitch Shift +12 :

▶ 0:00 / 0:25 ————— 🔊 ⋮

In [179...

```
audio_path = os.path.join("data", "Audio1.wav")
y, sr = librosa.load(audio_path, sr=None)
print(f"Sample rate: {sr} Hz | Durasi: {len(y)/sr:.2f} detik")

# dua pitch shift
y_up7 = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)
y_up12 = librosa.effects.pitch_shift(y, sr=sr, n_steps=12)

# menyamakan panjang sinyal
panjang = min(len(y_up7), len(y_up12))
y_up7 = y_up7[:panjang]
y_up12 = y_up12[:panjang]

# overlay dua sinyal (rata-rata biar tidak clipping)
chipmunk_mix = (y_up7 + y_up12) / 2.0

# menyimpan hasil overlay
output_name = "Soal3_Chipmunk_Overlay.wav"
output_path = os.path.join("data", output_name)
```

```

sf.write(output_path, chipmunk_mix, sr)
print(f"✅ File hasil overlay disimpan di: {output_path}")

# Hasil audio
print("🔊 Hasil Gabungan (+7 dan +12):")
display(Audio(chipmunk_mix, rate=sr))

```

Sample rate: 44100 Hz | Durasi: 25.54 detik

✅ File hasil overlay disimpan di: data\Soal3_Chipmunk_Overlay.wav

🔊 Hasil Gabungan (+7 dan +12):

▶ 0:00 / 0:25 🔊 ⋮

Penjelasan

1. Parameter yang digunakan:

- `n_steps` : Jumlah kenaikan atau ketinggian nada suara, contohnya +7 membuat suara menjadi lebih tinggi dan menjadi tidak jelas karena tingginya sebuah nada.
- `librosa.effects.pitch_shift(y, sr, n_steps)` : Melakukan shifting frekuensi tanpa mengubah durasi audio asli.

2. Perbedaan dalam representasi visual antara suara asli dan suara yang telah dimodifikasi:

Pada waveform dapat dilihat semakin tinggi pitch yang pakai menyebabkan amplitudo menjadi rapat dan melebar. Sedangkan pada spektrogram dapat dilihat frekuensi yang lebih tinggi pada audio yang sudah di pitch shift dibandingkan dengan audio asli.

3. Bagaimana perubahan pitch memengaruhi kualitas dan kejelasan suara:

Ketika nilai pitch dinaikkan, suara akan terdengar lebih tinggi atau cempreng, karena frekuensi gelombang suara meningkat. Meskipun durasi rekaman tetap sama, suara seolah terdengar lebih cepat akibat perubahan jarak antar gelombang yang menjadi lebih rapat. Namun, semakin tinggi pitch yang diterapkan, kualitas dan kejelasan suara dapat menurun, sehingga hasilnya terdengar kurang natural atau seperti suara karakter chipmunk

In [180...

```

import os
from pydub import AudioSegment, effects
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np

# Path file
file_path = os.path.join(os.getcwd(), "data", "Soal3_Chipmunk_Overlay.wav")

# Cek apakah file ada
if os.path.exists(file_path):
    y, sr = librosa.load(file_path, sr=None)

```

```

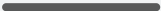
    print(f"✅ File ditemukan: {file_path}")
else:
    print("❌ File tidak ditemukan, periksa nama atau path file.")

# Play audio original
print("Audio Asli:")
display(Audio(y, rate=sr))

```

✅ File ditemukan: c:\Users\okedi\OneDrive\Documents\Multimedia\Worksheet 4\data\Soal3_Chipmunk_Overlay.wav

Audio Asli:

▶ 0:00 / 0:25  🔊 ⋮

Equalizer

In [181...

```

import numpy as np
from scipy.signal import butter, lfilter

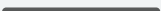
# Fungsi filter butterworth
def butter_filter(y, sr, cutoff, btype='low', order=6):
    nyquist = 0.5 * sr
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype=btype)
    return lfilter(b, a, y)

# Terapkan Band-Pass (100Hz - 6000Hz)
y_eq = butter_filter(y, sr, 100, btype='high')
y_eq = butter_filter(y_eq, sr, 6000, btype='low')

print("✅ Equalizer (Band-Pass 100Hz & 6000Hz) applied.")
display(Audio(y_eq, rate=sr))

```

✅ Equalizer (Band-Pass 100Hz & 6000Hz) applied.

▶ 0:00 / 0:25  🔊 ⋮

Fade-in & Fade-out selama 1 detik

In [182...

```


# Fade-in & fade-out selama 0.1 detik
fade_len = int(1 * sr)
fade_in = np.linspace(0, 1, fade_len)
fade_out = np.linspace(1, 0, fade_len)

y_fade = y_eq.copy()
y_fade[:fade_len] *= fade_in
y_fade[-fade_len:] *= fade_out

print("✅ Fade-in/out (1 detik) applied.")
display(Audio(y_fade, rate=sr))

```

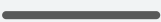
✅ Fade-in/out (1 detik) applied.

▶ 0:00 / 0:25  🔊 ⋮

Normalization (Peak ke -0 dBFS)

```
In [183... y_norm = y_fade / np.max(np.abs(y_fade))
print("✅ Normalized ke peak -0 dBFS")
display(Audio(y_norm, rate=sr))
```

✅ Normalized ke peak -0 dBFS


▶ 0:00 / 0:25  🔊 ⋮

Compression

```
In [184... def simple_compressor(y, threshold=0.2, ratio=4):
    y_comp = np.copy(y)
    mask = np.abs(y) > threshold
    y_comp[mask] = np.sign(y[mask]) * (threshold + (np.abs(y[mask]) - threshold)
    return y_comp

y_comp = simple_compressor(y_norm)
print("Compression applied (Threshold=0.2, Ratio=4:1)")
display(Audio(y_comp, rate=sr))
```


Compression applied (Threshold=0.2, Ratio=4:1)

▶ 0:00 / 0:25  🔊 ⋮

Noise Gate

```
In [185... threshold = 0.001
y_gate = np.where(np.abs(y_comp) < threshold, 0, y_comp)
print("✅ Noise Gate applied (threshold=0.001)")
display(Audio(y_gate, rate=sr))
```

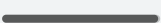
✅ Noise Gate applied (threshold=0.001)

▶ 0:00 / 0:25  🔊 ⋮

Silence Trimming

```
In [186... y_trimmed, index = librosa.effects.trim(y_gate, top_db=25)
print(f"✅ Silence trimmed (Start={index[0]/sr:.2f}s, End={index[1]/sr:.2f}s)")
display(Audio(y_trimmed, rate=sr))
```

✅ Silence trimmed (Start=0.38s, End=25.32s)

▶ 0:00 / 0:24  🔊 ⋮

Normalisasi Loudness ke -16 LUFS

```
In [187... import pyloudnorm as pyn
```



```

meter = pyln.Meter(sr)
loudness_before = meter.integrated_loudness(y_trimmed)
print(f"Loudness sebelum normalisasi: {loudness_before:.2f} LUFS")

y_loudnorm = pyln.normalize.loudness(y_trimmed, loudness_before, -16.0)
print("✅ Loudness distandarkan ke -16 LUFS")
display(Audio(y_loudnorm, rate=sr))

```

Loudness sebelum normalisasi: -22.67 LUFS

✅ Loudness distandarkan ke -16 LUFS

▶ 0:00 / 0:24 ———— 🔊 ⋮

Visualisasi Waveform dan Spectrogram

In [188...

```

def plot_waveform_and_spectrogram(y, sr, title_prefix=''):
    D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

    plt.figure(figsize=(14, 5))
    plt.subplot(2, 1, 1)
    librosa.display.waveshow(y, sr=sr)
    plt.title(title_prefix + 'Waveform')
    plt.xlabel('Waktu (detik)')
    plt.ylabel('Amplitudo')

    plt.subplot(2, 1, 2)
    librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar(format='%+2.0f dB')
    plt.title(title_prefix + 'Spectrogram')
    plt.xlabel('Waktu (detik)')
    plt.ylabel('Frekuensi (Hz)')
    plt.tight_layout()
    plt.show()

print("Sebelum:")
plot_waveform_and_spectrogram(y, sr, title_prefix='Sebelum Pemrosesan - ')

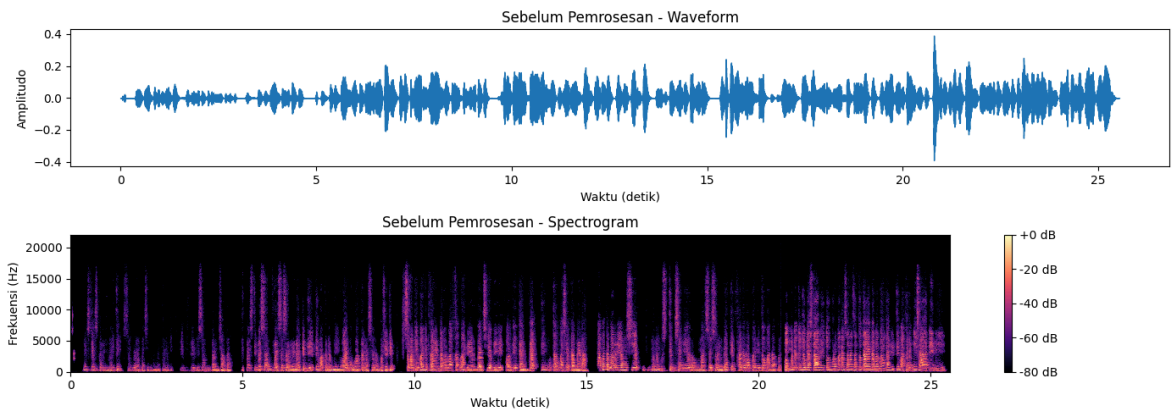
print("Sesudah(-16 LUFS):")
plot_waveform_and_spectrogram(y_loudnorm, sr, title_prefix='Sesudah Pemrosesan - ')

print("Audio Awal (Sebelum Proses):")
display(Audio(y, rate=sr))

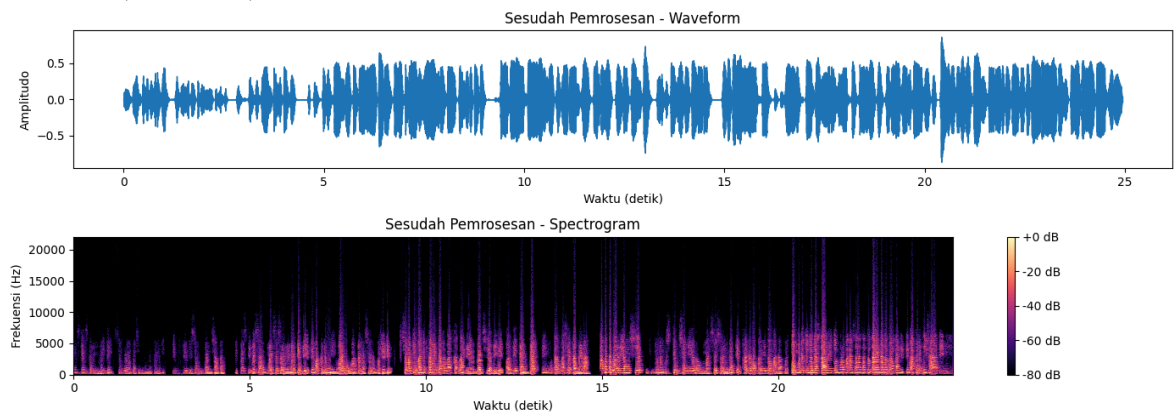
print("Audio Akhir (Sesudah Semua Tahap):")
display(Audio(y_loudnorm, rate=sr))

```

Sebelum:



Sesudah(-16 LUFS):



Audio Awal (Sebelum Proses):

▶ 0:00 / 0:25 ————— 🔊 ⋮

Audio Akhir (Sesudah Semua Tahap):

▶ 0:00 / 0:24 ————— 🔊 ⋮

Penjelasan:

1. Perubahan dinamika suara yang terjadi

Perubahan yang terjadi adalah suara menjadi lebih keras daripada sebelumnya, kemudian suara yang pelan menjadi lebih terdengar. Pada visualisasi waveform sebuah audio menjadi lebih tinggi amplitudonya atau gelombangnya menjadi lebih besar dan menjadi lebih rapat

2. Perbedaan antara normalisasi peak dan normalisasi LUFS

Normalisasi peak menyesuaikan level puncak tertinggi dari sinyal audio ke level target tertentu, misalnya 0 dBFS untuk menghindari distorsi. Sedangkan normalisasi LUFS menyesuaikan level loudness keseluruhan dari sinyal audio berdasarkan persepsi pendengaran manusia.

3. Bagaimana kualitas suara berubah setelah proses normalisasi dan loudness optimization

Sebuah suara menjadi stabil dan suara yang pelan menjadi lebih terdengar.

4. Kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara Kelebihan:

- Menjaga konsistensi volume
- Audio lebih nyaman didengar Kekurangan:
- Suaranya seperti terlalu dipaksakan tidak natural

In [189...

```
lokasi_sedih = os.path.join(os.getcwd(), "data", "Lagu Sedih.wav")
lokasi_bahagia = os.path.join(os.getcwd(), "data", "Lagu Bahagia.wav")

# Pengecekan keberadaan file dan pemuatan audio
if os.path.exists(lokasi_sedih):
    audio_sedih, sr_sedih = librosa.load(lokasi_sedih, sr=None)
    print(f"Lagu Sedih berhasil dimuat ({lokasi_sedih})")
else:
    print("Lagu Sedih tidak ditemukan!")

if os.path.exists(lokasi_bahagia):
    audio_bahagia, sr_bahagia = librosa.load(lokasi_bahagia, sr=None)
    print(f"Lagu Bahagia berhasil dimuat ({lokasi_bahagia})")
else:
    print("Lagu Bahagia tidak ditemukan!")

# Menampilkan pemutar audio
print("\nLagu Sedih:")
display(Audio(audio_sedih, rate=sr_sedih))

print("Lagu Bahagia:")
display(Audio(audio_bahagia, rate=sr_bahagia))
```

Lagu Sedih berhasil dimuat (c:\Users\okedi\OneDrive\Documents\Multimedia\Worksheet 4\data\Lagu Sedih.wav)

Lagu Bahagia berhasil dimuat (c:\Users\okedi\OneDrive\Documents\Multimedia\Worksheet 4\data\Lagu Bahagia.wav)

Lagu Sedih:

▶ 0:00 / 1:00 ————— 🔊 ⋮

Lagu Bahagia:

▶ 0:00 / 1:00 ————— 🔊 ⋮

In [190...

```
tempo_sedih, beat_sedih = librosa.beat.beat_track(y=audio_sedih, sr=sr_sedih)
tempo_bahagia, beat_bahagia = librosa.beat.beat_track(y=audio_bahagia, sr=sr_bahagia)

tempo_sedih = float(tempo_sedih[0]) if isinstance(tempo_sedih, np.ndarray) else
tempo_bahagia = float(tempo_bahagia[0]) if isinstance(tempo_bahagia, np.ndarray)

# Menampilkan hasil analisis tempo
print("HASIL DETEKSI TEMPO (BPM)\n")
print(f"Lagu Bahagia → Tempo : {tempo_bahagia:.1f} BPM | Jumlah ketukan: {len(beat_bahagia)}")
print(f"Lagu Sedih → Tempo : {tempo_sedih:.1f} BPM | Jumlah ketukan: {len(beat_sedih)}")
```

HASIL DETEKSI TEMPO (BPM)

Lagu Bahagia → Tempo : 90.7 BPM | Jumlah ketukan: 87

Lagu Sedih → Tempo : 143.6 BPM | Jumlah ketukan: 144

```
In [191... # Fungsi estimasi kunci berdasarkan chroma energy
def deteksi_kunci(audio, sr):
    chroma = librosa.feature.chroma_stft(y=audio, sr=sr)
    rata_chroma = np.mean(chroma, axis=1)
    daftar_nada = ['C', 'C#', 'D', 'D#', 'E', 'F',
                  'F#', 'G', 'G#', 'A', 'A#', 'B']
    nada_terkuat = np.argmax(rata_chroma)
    kunci = daftar_nada[nada_terkuat]
    return kunci, chroma

# Estimasi untuk masing-masing lagu
kunci_sedih, chroma_sedih = deteksi_kunci(audio_sedih, sr_sedih)
kunci_bahagia, chroma_bahagia = deteksi_kunci(audio_bahagia, sr_bahagia)

# Hasil estimasi kunci
print("\nHASIL ESTIMASI KUNCI (KEY)\n")
print(f"Lagu Sedih → Kunci dominan: {kunci_sedih}")
print(f"Lagu Bahagia → Kunci dominan: {kunci_bahagia}")
```

HASIL ESTIMASI KUNCI (KEY)

Lagu Sedih → Kunci dominan: D#

Lagu Bahagia → Kunci dominan: G#

Analisis Singkat Dari analisis yang dilakukan diperoleh BPM lagu sedih adalah 143 sedangkan lagu bahagia 90 BPM. kemudian untuk estimasi kunci lagu sedih adalah D minor lalu untuk lagu bahagia adalah G major. Berdasarkan hasil tersebut dapat disimpulkan bahwa lagu sedih memiliki tempo yang lebih cepat dibandingkan dengan lagu bahagia, hal ini mungkin karena lagu sedih memiliki ritme yang lebih dinamis untuk mengekspresikan emosi sedih. Sedangkan untuk kunci lagu, lagu sedih berada dalam kunci minor yang sering dikaitkan dengan suasana hati yang melankolis atau sedih, sementara lagu bahagia berada dalam kunci mayor yang biasanya dikaitkan dengan suasana hati yang ceria dan optimis.

```
In [192... # Tentukan target BPM (gunakan rata-rata dari kedua lagu)
target_bpm = (tempo_sedih + tempo_bahagia) / 2
print(f"\nTarget tempo untuk penyamaan: {target_bpm:.1f} BPM")

# Hitung rasio perubahan tempo
ratio_sedih = target_bpm / tempo_sedih
ratio_bahagia = target_bpm / tempo_bahagia

print(f"Rasio stretch Lagu Sedih : {ratio_sedih:.3f}")
print(f"Rasio stretch Lagu Bahagia : {ratio_bahagia:.3f}")

# Lakukan time stretch menggunakan rasio yang sudah dihitung
audio_sedih_stretch = librosa.effects.time_stretch(audio_sedih, rate=ratio_sedih)
audio_bahagia_stretch = librosa.effects.time_stretch(audio_bahagia, rate=ratio_b)

# Tampilkan audio hasil stretch
print("\nLagu Sedih (Setelah Time-Stretch):")
display(Audio(audio_sedih_stretch, rate=sr_sedih))
```

```
print("Lagu Bahagia (Setelah Time-Stretch):")
display(Audio(audio_bahagia_stretch, rate=sr_bahagia))
```

Target tempo untuk penyamaan: 117.1 BPM

Rasio stretch Lagu Sedih : 0.816

Rasio stretch Lagu Bahagia : 1.292

Lagu Sedih (Setelah Time-Stretch):

▶ 0:00 / 1:14 ————— 🔊 ⋮

Lagu Bahagia (Setelah Time-Stretch):

▶ 0:00 / 0:47 ————— 🔊 ⋮

In [193...

```
# Tentukan durasi crossfade dalam detik
durasi_crossfade = 5.0 # misalnya 5 detik
panjang_crossfade_sedih = int(durasi_crossfade * sr_sedih)
panjang_crossfade_bahagia = int(durasi_crossfade * sr_bahagia)

# Pastikan Lagu Bahagia disesuaikan dengan durasi crossfade
fade_out_sedih = np.linspace(1, 0, panjang_crossfade_sedih)
fade_in_bahagia = np.linspace(0, 1, panjang_crossfade_bahagia)

# Ambil bagian akhir lagu sedih untuk crossfade
bagian_sedih_akhir = audio_sedih_stretch[:panjang_crossfade_sedih]
bagian_sedih_fade = bagian_sedih_akhir * fade_out_sedih

# Ambil bagian awal lagu bahagia untuk crossfade
bagian_bahagia_fade = audio_bahagia_stretch[:panjang_crossfade_bahagia] * fade_in_bahagia
bagian_bahagia_akhir = audio_bahagia_stretch[panjang_crossfade_bahagia:]

# Gabungkan semua bagian dengan crossfade
hasil_crossfade = np.concatenate((bagian_sedih_fade, bagian_bahagia_akhir))

print("✅ Crossfade berhasil dilakukan!")
print(f"Durasi crossfade: {durasi_crossfade:.1f} detik")

# Tampilkan audio hasil crossfade
display(Audio(hasil_crossfade, rate=sr_sedih))
```

✅ Crossfade berhasil dilakukan!

Durasi crossfade: 5.0 detik

▶ 0:00 / 1:56 ————— 🔊 ⋮

Penjelasan

Crossfading digunakan untuk menggabungkan dua lagu secara halus sehingga transisi dari lagu pertama (Lagu Sedih) ke lagu kedua (Lagu Bahagia) tidak terdengar kasar atau terpotong mendadak. Parameter yang digunakan dalam proses crossfading ini meliputi:

- `durasi_crossfade`: Durasi waktu di mana kedua lagu akan saling tumpang

- `fade_out_sedih`: Fungsi fade-out yang diterapkan pada lagu sedih untuk mengurangi volumenya secara bertahap selama durasi crossfade.
- `fade_in_bahagia`: Fungsi fade-in yang diterapkan pada lagu bahagia untuk meningkatkan volumenya secara bertahap selama durasi crossfade.
- `np.concatenate()` : Fungsi ini digunakan untuk menggabungkan bagian akhir dari lagu sedih yang telah di-fade out dengan bagian awal dari lagu bahagia yang telah di-fade in, sehingga menghasilkan transisi yang mulus antara kedua lagu tersebut.

In [194...

```
# Buat spektrogram linear (biasa)
S = np.abs(librosa.stft(hasil_crossfade))
S_db = librosa.amplitude_to_db(S, ref=np.max)

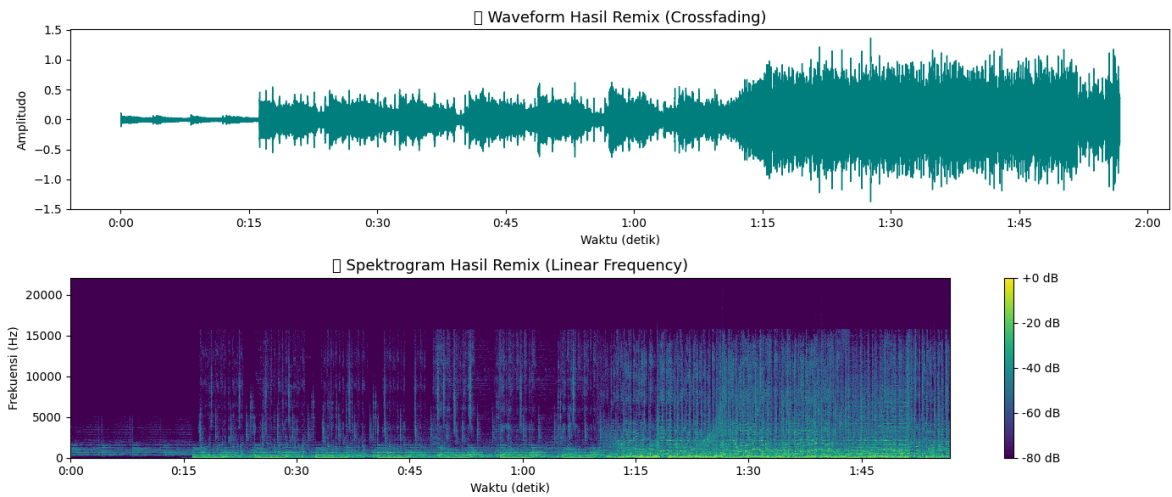
# Plot waveform dan spektrogram
plt.figure(figsize=(14, 6))

# 1 Waveform
plt.subplot(2, 1, 1)
librosa.display.waveshow(hasil_crossfade, sr=sr_sedih, color='teal')
plt.title("🎵 Waveform Hasil Remix (Crossfading)", fontsize=13)
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# 2 Spektrogram (biasa / linear)
plt.subplot(2, 1, 2)
librosa.display.specshow(S_db, sr=sr_sedih, x_axis='time', y_axis='hz', cmap='vib')
plt.colorbar(format='%+2.0f dB')
plt.title("🌈 Spektrogram Hasil Remix (Linear Frequency)", fontsize=13)
plt.xlabel("Waktu (detik)")
plt.ylabel("Frekuensi (Hz)")

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

```
C:\Users\okedi\AppData\Local\Temp\ipykernel_11584\4215059973.py:23: UserWarning:
Glyph 127925 (\N{MUSICAL NOTE}) missing from font(s) DejaVu Sans.
plt.tight_layout()
C:\Users\okedi\AppData\Local\Temp\ipykernel_11584\4215059973.py:23: UserWarning:
Glyph 127752 (\N{RAINBOW}) missing from font(s) DejaVu Sans.
plt.tight_layout()
c:\Users\okedi\OneDrive\Documents\Multimedia\Worksheet 4\multimedia-uv\lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127925 (\N{MUSICAL NO
TE}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)
c:\Users\okedi\OneDrive\Documents\Multimedia\Worksheet 4\multimedia-uv\lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127752 (\N{RAINBOW})
missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)
```

Penjelasan hasil remix dari hasil remix tersebut dapat dilihat awal lagu sedih memiliki amplitudo yang rendah sedangkan lagu bahagia amplitudonya lebih tinggi dan juga rapat, pada spectrogram juga terlihat frekuensi lagu sedih lebih banyak di frekuensi rendah dan ruang kosong sedangkan lagu bahagia frekuensi lebih merata dan rapat satu sama lainnya.

Referensi:

[Hands-on Audio Processing Week 3 – GitHub](#)

[Referensi Berita](#)

[ChatGPT](#)

[Musik 1 \(Sedih\)](#)

[Musik 2 \(Bahagia\)](#)