

Tugas Analisis Multimedia: Audio Processing

Mata Kuliah: Sistem & Teknologi Multimedia

Nama: Zidan Raihan **NIM:** 122140100

Import Library

In [171...]

```
# === Cell 1: Import library ===
# Semua Library yang dibutuhkan digabungkan di sini

# Untuk Soal 1 (Analisis Audio)
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import Audio

# Untuk Soal 2, 3, 4, 5 (Manipulasi Audio)
from pydub import AudioSegment
from pydub.effects import normalize

# Library 'os' untuk menggabungkan path file (sesuai permintaan)
import os
```

Soal 1

Proses yang Dilakukan:

1. **Loading Audio:** Audio `audio_1.wav` dimuat menggunakan librosa dengan sampling rate default (22050 Hz)
2. **Visualisasi Awal:** Menampilkan waveform dan spektrogram dari audio asli
3. **Resampling:** Audio di-resample dari 22050 Hz ke 16000 Hz menggunakan `librosa.resample()`
4. **Perbandingan Visual:** Membandingkan waveform sebelum dan sesudah resampling

In [172...]

```
# === Tentukan Path & Muat Audio (Librosa) ===
# Menggunakan os.path.join untuk memuat audio
base_path = "/content"
audio_filename = "audio_1.wav"
file_path = os.path.join(base_path, audio_filename)

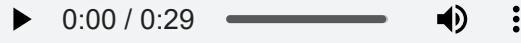
y, sr = librosa.load(file_path)

# === Tampilkan Info & Putar Audio ===
durasi = librosa.get_duration(y=y, sr=sr)
print(f"Durasi audio: {durasi:.2f} detik")
print(f"Sampling rate: {sr} Hz")
```

```
# Putar audio
Audio(data=y, rate=sr)
```

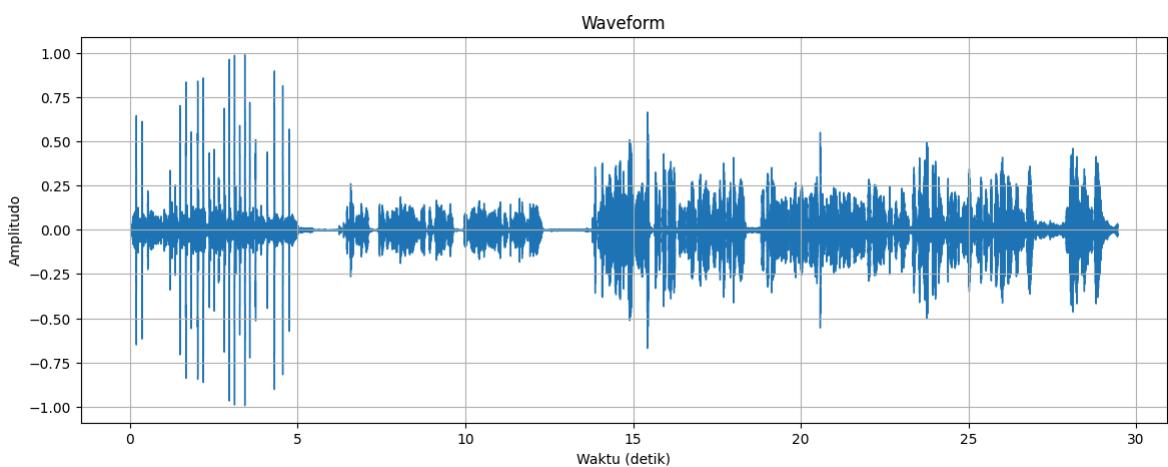
Durasi audio: 29.44 detik
Sampling rate: 22050 Hz

Out[172...]



In [173...]

```
# === Waveform ===
plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.grid(True)
```

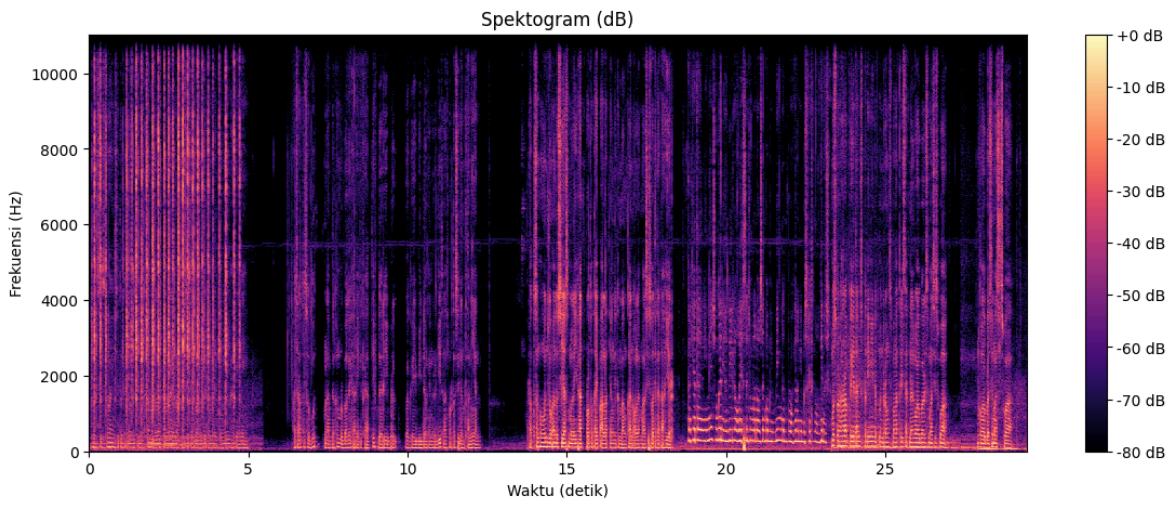


In [174...]

```
# === Spektrogram ===
D = librosa.stft(y) # Short-Time Fourier Transform
S_db = librosa.amplitude_to_db(np.abs(D), ref=np.max) # Konversi ke dB

plt.figure(figsize=(14, 5))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spektrogram (dB)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
```

Out[174...]: Text(0, 0.5, 'Frekuensi (Hz)')



```
In [175...]: # === Resampling dan perbandingan ===

# Resampling ke 16 kHz
target_sr = 16000
y_resampled = librosa.resample(y, orig_sr=sr, target_sr=target_sr)

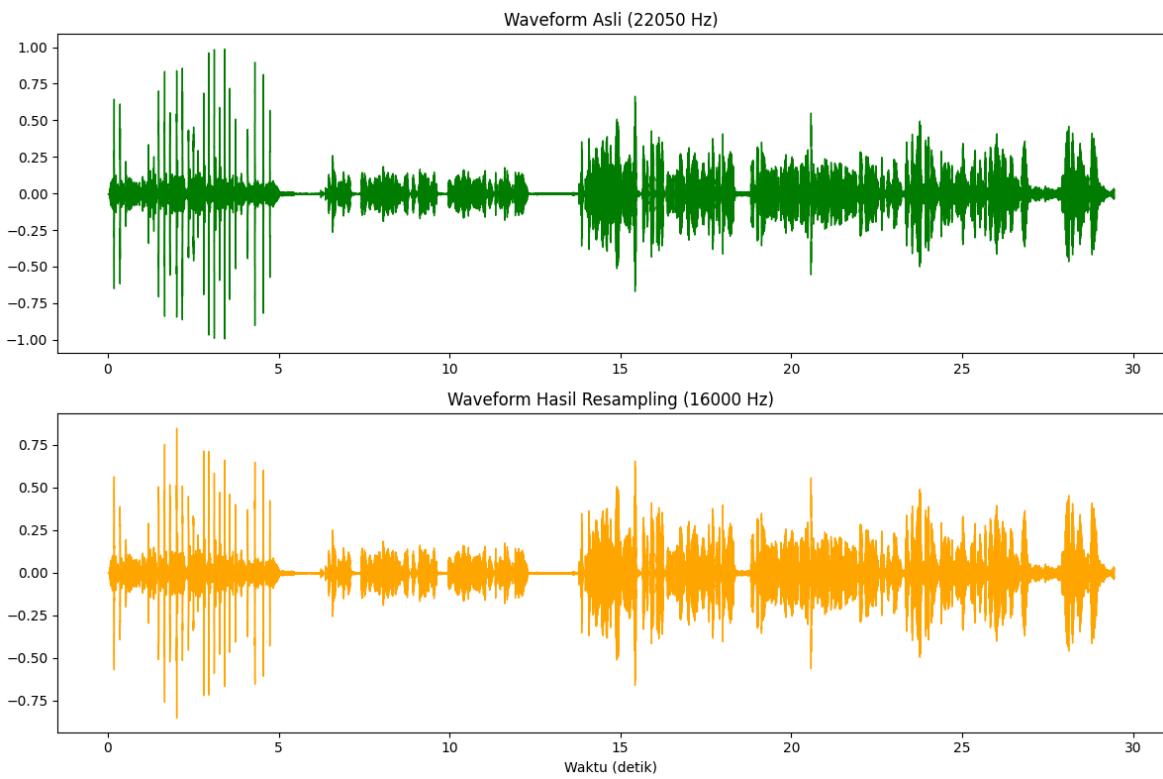
print(f"Sebelum resampling: {sr} Hz, Durasi: {librosa.get_duration(y=y, sr=sr)}")
print(f"Setelah resampling: {target_sr} Hz, Durasi: {librosa.get_duration(y=y_resampled)}")

# Visualisasi perbandingan waveform
plt.figure(figsize=(12, 8)) # Sedikit menambah tinggi figur agar tidak terlalu rapat
plt.subplot(2, 1, 1)
librosa.display.waveshow(y, sr=sr, color='green')
plt.title(f"Waveform Asli ({sr} Hz)")
plt.xlabel("") # Menghapus label x di plot atas agar tidak tumpang tindih

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

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

Sebelum resampling: 22050 Hz, Durasi: 29.44 detik
 Setelah resampling: 16000 Hz, Durasi: 29.44 detik



Hasil Analisis Visualisasi:

Perubahan Amplitudo: Secara visual, amplitudo tampak mengecil setelah proses resampling. Amplitudo yang mengecil ketika resampling merupakan hal yang normal karean melibatkan interpolasi dan filtering.

Perbedaan Setelah Resampling: Perbedaan dapat lebih jelas terlihat ketika dilakukan zoom pada grafik. Durasi audio tetap sama (dalam detik), namun jumlah sample berkurang sesuai dengan pengurangan sampling rate. Kualitas audio sedikit menurun karena informasi frekuensi tinggi di atas Nyquist frequency baru (8 kHz) akan hilang. Secara bentuk gelombangnya tetap mempertahankan karakteristik aslinya.

Soal 2

Proses yang Dilakukan:

1. **Loading Audio:** Audio `audio_2.wav` dimuat menggunakan librosa dengan sample rate asli terjaga
2. **Implementasi Filter Butterworth:** Dibuat fungsi `butter_filter()` untuk menerapkan berbagai jenis filter (lowpass, highpass, bandpass)
3. **Eksperimen Filter:** Diterapkan berbagai filter dengan cutoff frequencies 500Hz, 1000Hz, dan 2000Hz
4. **Visualisasi Perbandingan:** Setiap hasil filter divisualisasikan dengan waveform dan spektrogram
5. **Pemilihan Filter Optimal:** Dipilih filter band-pass dengan rentang 1000Hz - 4000Hz sebagai filter final

```
In [176...]: # === Load audio ===  
file_path2 = os.path.join(base_path, "audio_2.wav")  
  
y2, sr2 = librosa.load(file_path2, sr=None) # sr=None agar sample rate asli terjaga  
  
print(f"Durasi audio: {librosa.get_duration(y=y2, sr=sr2):.2f} detik")  
print(f"Sampling rate: {sr2} Hz")  
  
# Tampilkan player  
Audio(data=y2, rate=sr2)
```

Durasi audio: 8.50 detik
Sampling rate: 44100 Hz

Out[176...]:

```
In [177...]: # === Terapkan filter equalisasi ===  
  
# Filter  
def butter_filter(data, sr, cutoff, filter_type, order=5):  
    nyquist = 0.5 * sr  
    if filter_type == 'band':  
        low, high = cutoff  
        low /= nyquist  
        high /= nyquist  
        b, a = butter(order, [low, high], btype='band')  
    else:  
        normal_cutoff = cutoff / nyquist  
        b, a = butter(order, normal_cutoff, btype=filter_type)  
    return filtfilt(b, a, data)  
  
# Eksperimen dengan beberapa cutoff  
cutoffs = [500, 1000, 2000]  
filtered_results = {}  
  
for c in cutoffs:  
    filtered_results[f"lowpass_{c}Hz"] = butter_filter(y2, sr2, c, 'low')
```

```

    filtered_results[f"highpass_{c}Hz"] = butter_filter(y2, sr2, c, 'high')

# Contoh band-pass di antara 500–2000 Hz
filtered_results["bandpass_500_2000Hz"] = butter_filter(y2, sr2, (500, 2000), 'bandpass')

print("Filter berhasil diterapkan untuk berbagai cutoff.")

```

Filter berhasil diterapkan untuk berbagai cutoff.

```

In [178]: # === Visualisasi & hasil tiap filter ===

def plot_spectrogram(y, sr, title):
    S = librosa.stft(y)
    S_db = librosa.amplitude_to_db(np.abs(S), ref=np.max)
    librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
    plt.title(title)
    plt.colorbar(format='%+2.0f dB')
    plt.xlabel('Waktu (detik)')
    plt.ylabel('Frekuensi (Hz)')

plt.figure(figsize=(16, 5))

# === Spektrogram Asli ===
plt.subplot(1, 2, 1)
plot_spectrogram(y2, sr2, "Spektrogram Asli")
plt.subplot(1, 2, 2)
librosa.display.waveform(y2, sr=sr2, color='gray')
plt.title("Waveform Asli")

plt.tight_layout()
plt.show()

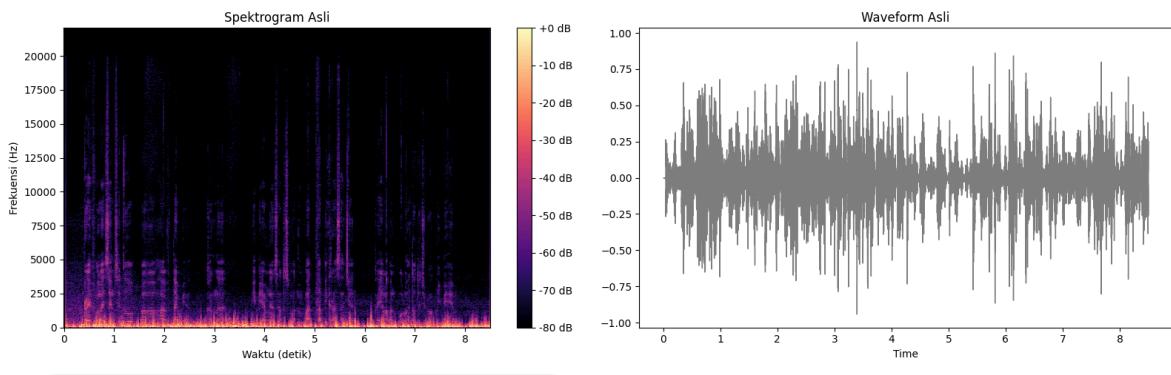
display(Audio(y2, rate=sr2))
print("Audio Asli\n" + "*30 + \n")

# === Tampilkan hasil tiap filter ===
for name, data in filtered_results.items():
    plt.figure(figsize=(14, 6))
    plt.subplot(2, 1, 1)
    librosa.display.waveform(data, sr=sr2, color='teal')
    plt.title(f"Waveform - {name}")

    plt.subplot(2, 1, 2)
    plot_spectrogram(data, sr2, f"Spektrogram - {name}")
    plt.tight_layout()
    plt.show()

# Putar audio hasil filter
display(Audio(data, rate=sr2))
print(f"Audio Hasil Filter: {name}\n")

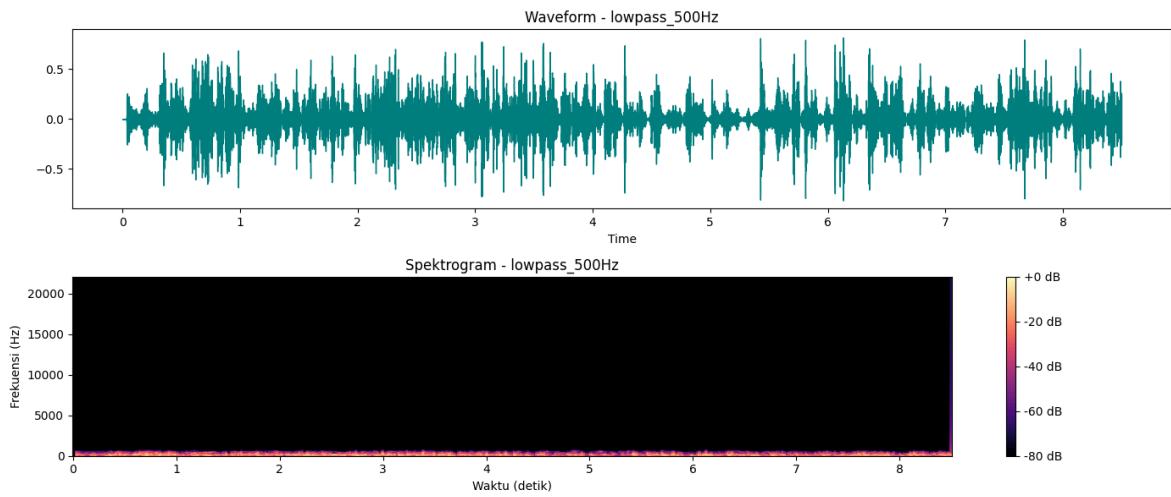
```



▶ 0:00 / 0:08 ⏴ ⋮

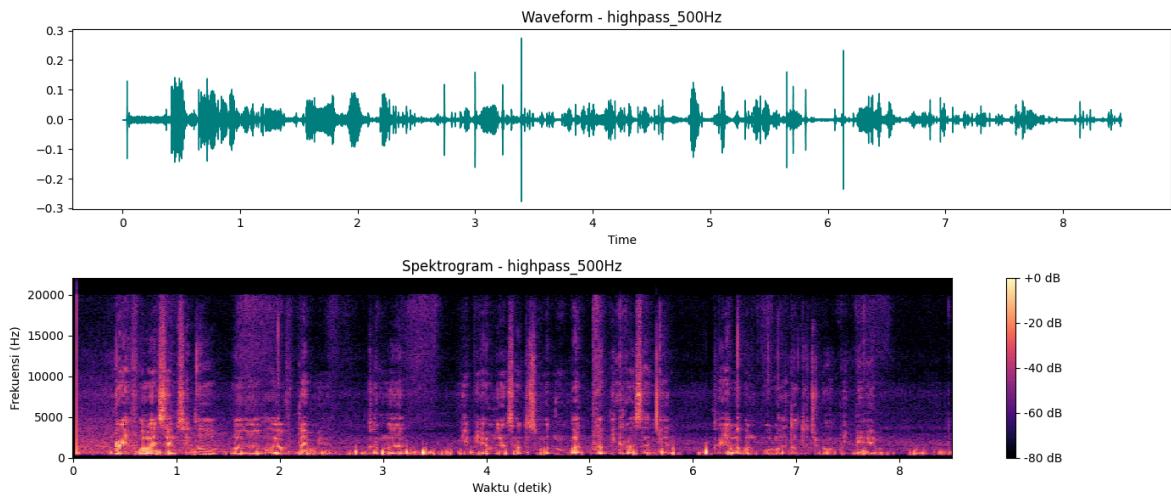
Audio Asli

=====



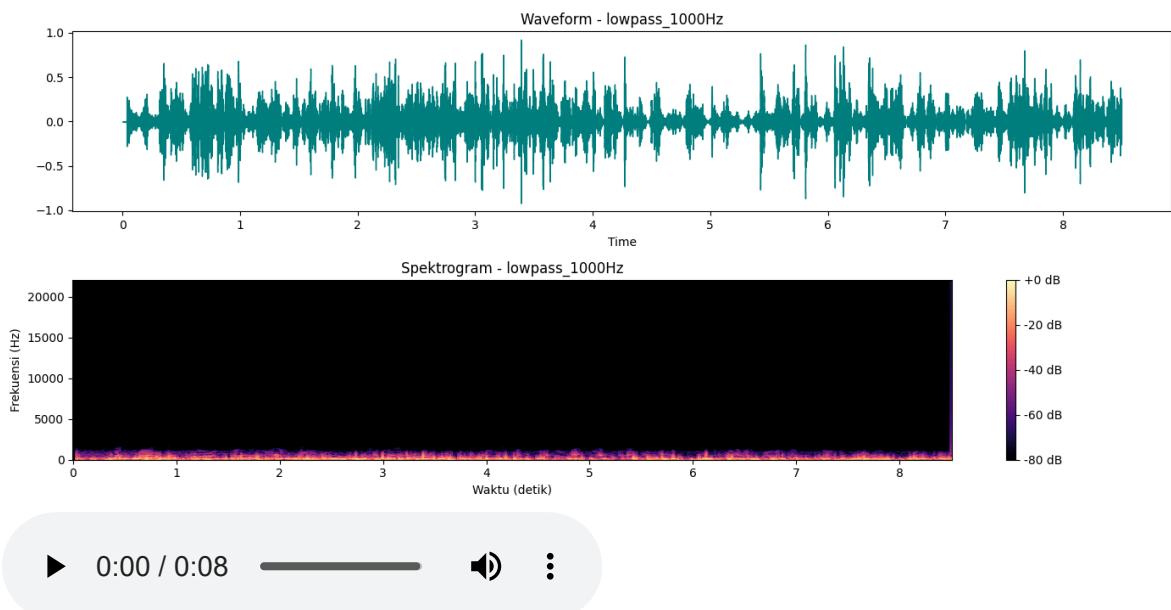
▶ 0:00 / 0:08 ⏴ ⋮

Audio Hasil Filter: lowpass_500Hz

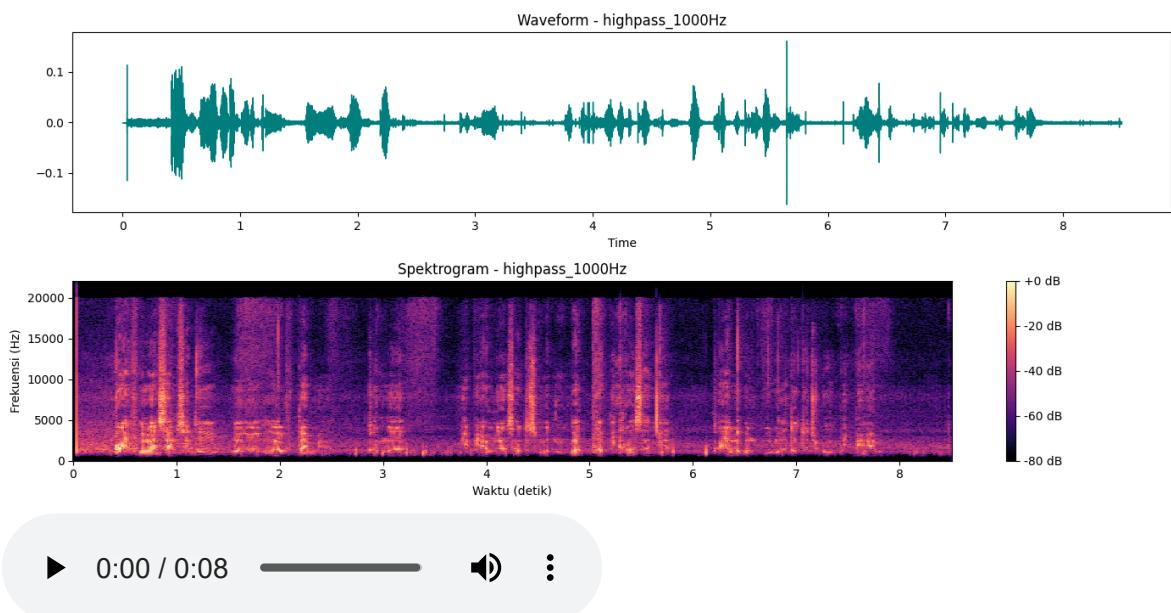


▶ 0:00 / 0:08 ⏴ ⋮

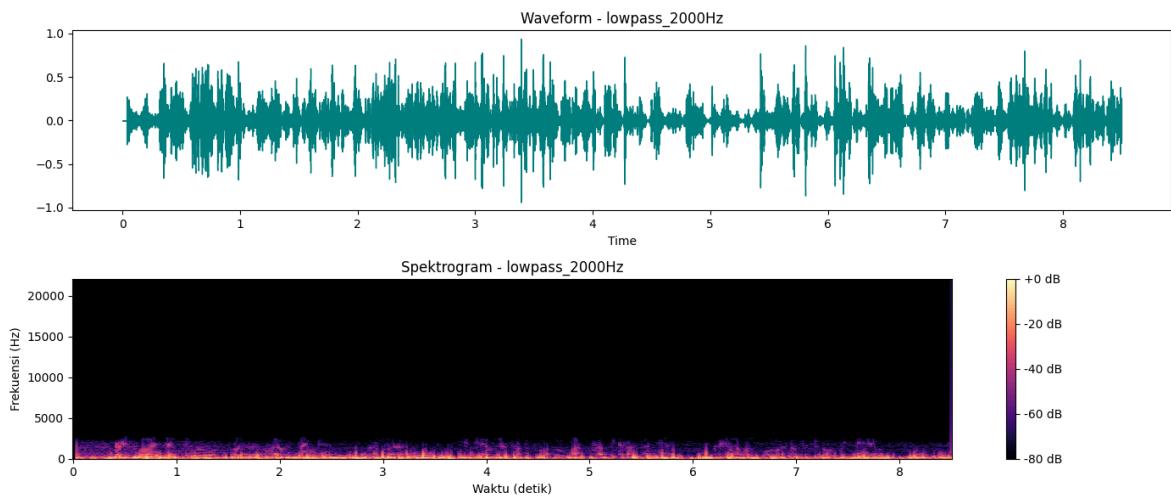
Audio Hasil Filter: highpass_500Hz



Audio Hasil Filter: lowpass_1000Hz

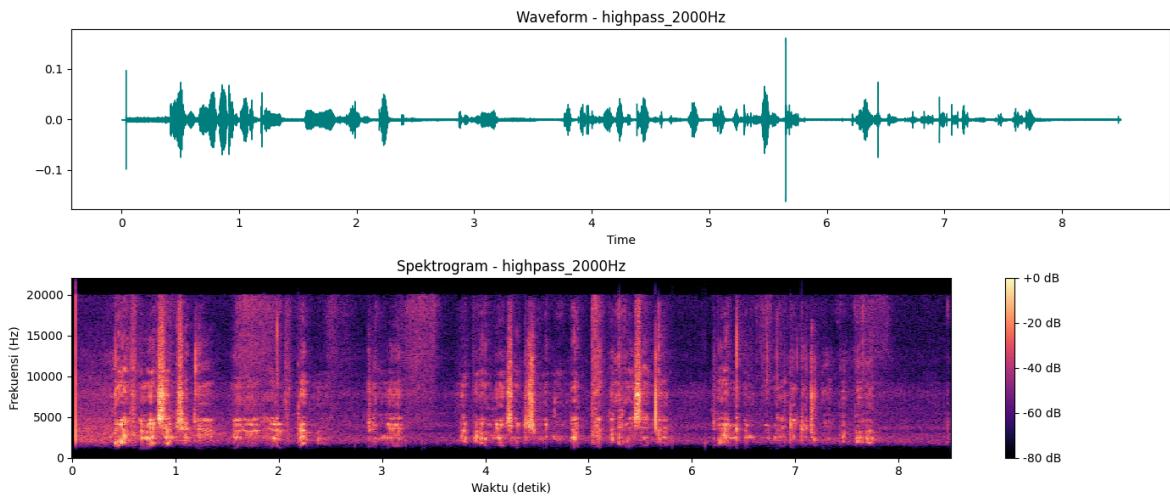


Audio Hasil Filter: highpass_1000Hz



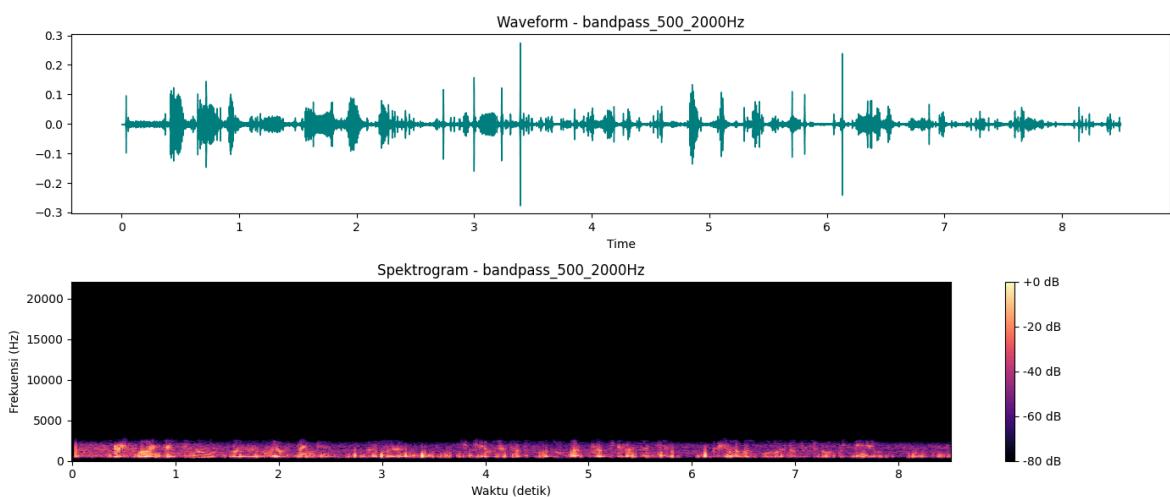
▶ 0:00 / 0:08 ⏸

Audio Hasil Filter: lowpass_2000Hz



▶ 0:00 / 0:08 ⏸

Audio Hasil Filter: highpass_2000Hz



▶ 0:00 / 0:08 ⏸

Audio Hasil Filter: bandpass_500_2000Hz

```
In [179...]: cutoff_final = (1000, 4000) # (Low_cutoff, high_cutoff)

# Terapkan filter band-pass
y_filtered_final = butter_filter(y2, sr2, cutoff_final, 'band')

print(f"Filter final 'Band-pass' (1000 Hz - 4000 Hz) diterapkan.")

# === Visualisasi Hasil Final ===
plt.figure(figsize=(14, 6))

plt.subplot(2, 1, 1)
```

```

librosa.display.waveshow(y_filtered_final, sr=sr2, color='blue')
plt.title(f"Waveform - Filter Final (Band-pass 1000 Hz - 4000 Hz)")
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

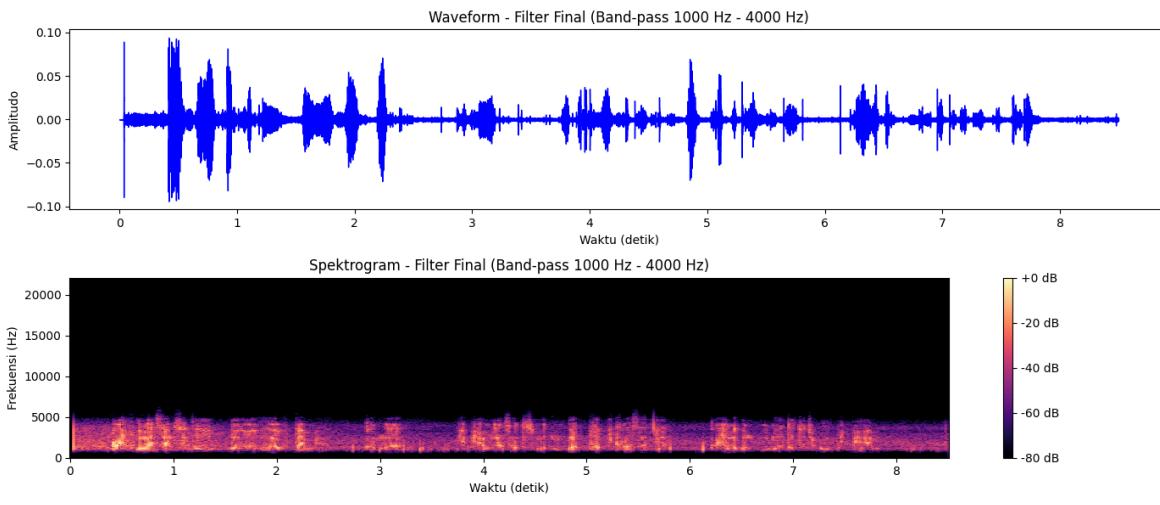
plt.subplot(2, 1, 2)
plot_spectrogram(y_filtered_final, sr2, f"Spektrogram - Filter Final (Band-pass 1000 Hz - 4000 Hz)")

plt.tight_layout()
plt.show()

# === Putar Audio Hasil Filter Final ===
display(Audio(data=y_filtered_final, rate=sr2))
print(f"Audio Hasil Filter Final: Band-pass (1000 Hz - 4000 Hz)\n")

```

Filter final 'Band-pass' (1000 Hz - 4000 Hz) diterapkan.



▶ 0:00 / 0:08 🔊 ⋮

Audio Hasil Filter Final: Band-pass (1000 Hz - 4000 Hz)

Analisis Hasil:

Jenis Noise yang Teridentifikasi:

- Noise Frekuensi Rendah (< 500Hz)**: Suara seperti geluduk dan getaran lingkungan
- Noise Frekuensi Tinggi (> 4000Hz)**: Hiss, noise angin, dan interferensi elektronik

Filter Terbaik: Berdasarkan pengamatan, **band-pass filter dengan rentang 1000Hz - 4000Hz** memberikan hasil terbaik karena:

- Mengurangi Noise Frekuensi Rendah:** Menghilangkan suara seperti geluduk yang berada di bawah 1000Hz
- Mengurangi Noise Frekuensi Tinggi:** Memfilter noise angin dan hiss yang berada di atas 4000Hz
- Mempertahankan Kejernihan Suara:** Suara tetap jernih dan jelas terdengar meski sedikit teredam

Perbandingan dengan Filter atau Nilai Cutoff Lain:

- **500Hz cutoff:** Masih membiarkan sebagian noise frekuensi rendah
- **1000Hz cutoff:** Memberikan pembersihan optimal untuk frekuensi rendah
- **2000Hz cutoff:** Terlalu agresif, menghilangkan beberapa komponen vokal penting
- **4000Hz cutoff:** Ideal sebagai batas atas, mempertahankan kejernihan suara
- **Band-pass 500-2000Hz:** Masih membiarkan beberapa noise frekuensi rendah masuk dan suara aslinya kurang jelas terdengar

Filter band-pass 1000-4000Hz terbukti memberikan keseimbangan terbaik antara pengurangan noise dan preservasi kualitas audio.

Soal 3

Deskripsi Audio Asli:

Audio dari Soal 1 berisi suara yang ramai dan berisik dengan vocal yang cempreng hingga sedikit teriak. Audio memiliki latar belakang yang bising dengan kualitas vocal yang sudah tidak optimal sejak awal.

Parameter yang Digunakan:

- **Pitch Shift +7 Semitone:** Menaikkan pitch sebesar 7 semitone (hampir 1 oktaf)
- **Pitch Shift +12 Semitone:** Menaikkan pitch sebesar 12 semitone (1 oktaf penuh)
- **Sampling Rate:** 22050 Hz (default librosa)
- **Metode:** `librosa.effects.pitch_shift()` dengan preservasi durasi

```
In [180...]: # Pitch shift +7 dan +12 semitone (tanpa ubah tempo)
y_pitch_7 = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)
y_pitch_12 = librosa.effects.pitch_shift(y, sr=sr, n_steps=12)

print("✅ Pitch shifting selesai (+7 dan +12 semitone, tanpa tempo stretch).")

# === Visualisasi ===
plt.figure(figsize=(16, 10))

# Asli
plt.subplot(3, 2, 1)
librosa.display.waveshow(y, sr=sr, color='gray')
plt.title("Waveform Asli")

plt.subplot(3, 2, 2)
S_orig = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(S_orig, sr=sr, x_axis='time', y_axis='hz')
plt.title("Spektrogram Asli")

# Pitch +7
plt.subplot(3, 2, 3)
librosa.display.waveshow(y_pitch_7, sr=sr, color='green')
plt.title("Waveform Pitch +7")

plt.subplot(3, 2, 4)
```

```

S_7 = librosa.amplitude_to_db(np.abs(librosa.stft(y_pitch_7))), ref=np.max)
librosa.display.specshow(S_7, sr=sr, x_axis='time', y_axis='hz')
plt.title("Spektrogram Pitch +7")

# Pitch +12
plt.subplot(3, 2, 5)
librosa.display.waveform(y_pitch_12, sr=sr, color='orange')
plt.title("Waveform Pitch +12")

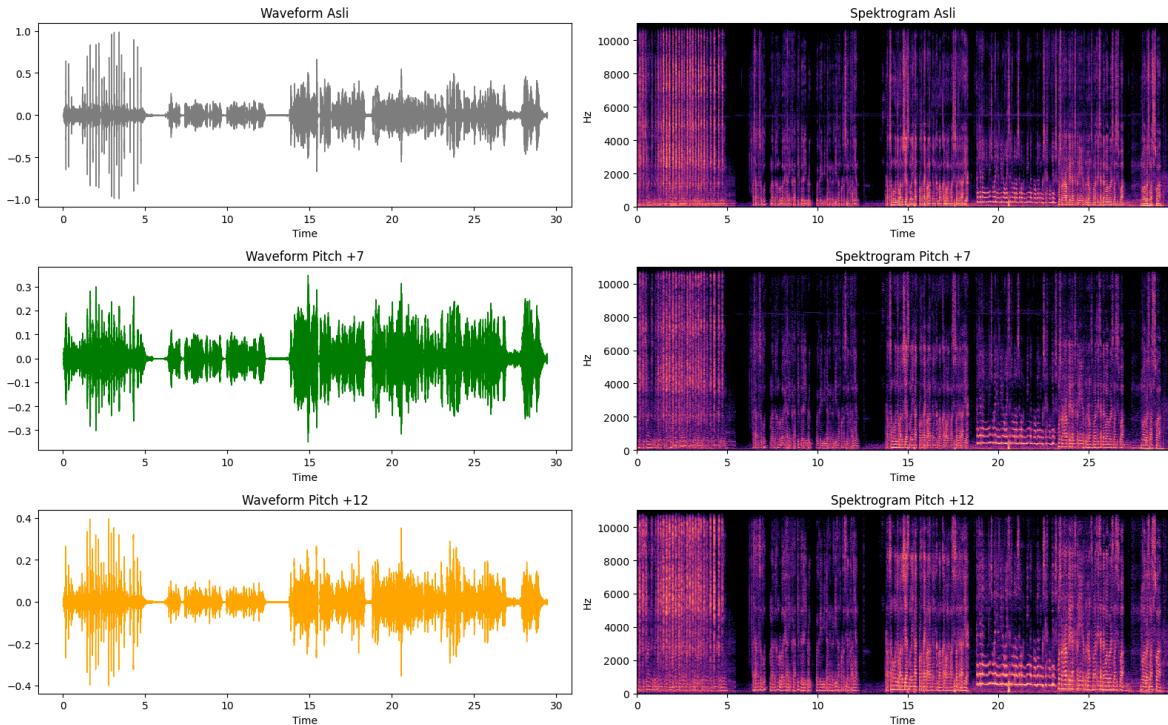
plt.subplot(3, 2, 6)
S_12 = librosa.amplitude_to_db(np.abs(librosa.stft(y_pitch_12))), ref=np.max)
librosa.display.specshow(S_12, sr=sr, x_axis='time', y_axis='hz')
plt.title("Spektrogram Pitch +12")

plt.tight_layout()
plt.show()

# === Audio playback ===
print("🎧 Audio Asli:")
display(Audio(y, rate=sr))
print("🎧 Pitch +7 Semitone:")
display(Audio(y_pitch_7, rate=sr))
print("🎧 Pitch +12 Semitone:")
display(Audio(y_pitch_12, rate=sr))

```

Pitch shifting selesai (+7 dan +12 semitone, tanpa tempo stretch).



🎧 Audio Asli:

▶ 0:00 / 0:29 ━━ 🔊 ⋮

🎧 Pitch +7 Semitone:

▶ 0:00 / 0:29 ━━ 🔊 ⋮

🎧 Pitch +12 Semitone:

```

▶ 0:00 / 0:29 ━ ⏰ ⋮

In [181...]:
# Gabungkan kedua hasil pitch shift
combined_audio = np.concatenate([y_pitch_7, y_pitch_12])

# Simpan ke file baru
output_path_pitch = os.path.join(base_path, "audio_chipmunk_combined.wav")
sf.write(output_path_pitch, combined_audio, sr)

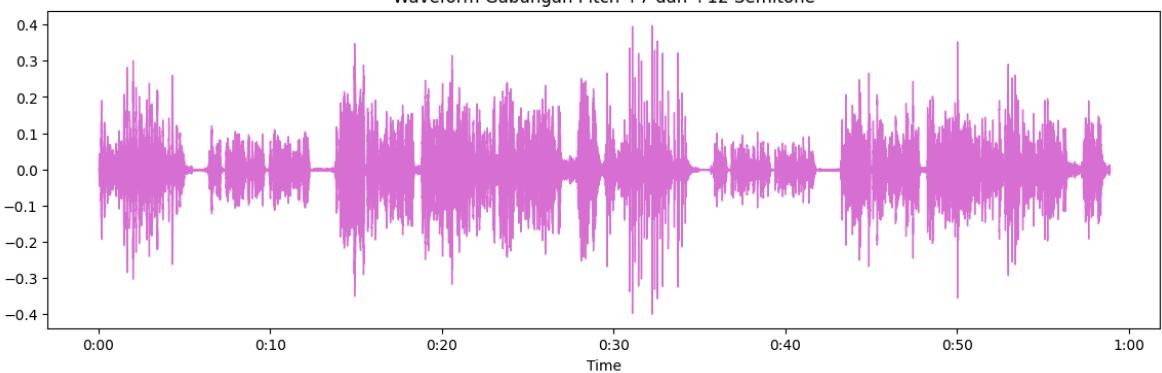
print(f"File audio gabungan berhasil disimpan: {output_path_pitch}")

# === 5. Visualisasi & Dengarkan ===
plt.figure(figsize=(14, 4))
librosa.display.waveform(combined_audio, sr=sr, color='orchid')
plt.title("Waveform Gabungan Pitch +7 dan +12 Semitone")
plt.show()

print("Audio Gabungan (Pitch +7 diikuti +12):")
display(Audio(combined_audio, rate=sr))

```

File audio gabungan berhasil disimpan: /content/audio_chipmunk_combined.wav
 Waveform Gabungan Pitch +7 dan +12 Semitone



Audio Gabungan (Pitch +7 diikuti +12):

▶ 0:00 / 0:58 ━ ⏰ ⋮

Perbedaan Representasi Visual:

1. Waveform:

- **Amplitudo:** Bentuk gelombang tetap mempertahankan pola asli, namun
- **Durasi:** Tetap sama karena pitch shifting mempertahankan timing

2. Spektrogram:

- **Pitch +7:** Semua komponen frekuensi bergeser naik sampai 5000 Hz
- **Pitch +12:** Semua komponen frekuensi bergeser naik 1 oktaf (frekuensi menjadi 2x lipat)
- **Harmonik:** Pola harmonik tetap namun bergeser ke frekuensi yang lebih tinggi

Pengaruh Perubahan Pitch terhadap Kualitas dan Kejelasan:

Pitch +7 Semitone:

- **Latar Berisik:** Vocal pelan menjadi **tidak jelas** karena pitch tinggi membuat noise lebih dominan
- **Vocal Cempreng:** Masih **dapat terdengar** meski dengan kualitas yang menurun
- **Efek Chipmunk:** Mulai terdengar efek suara "chipmunk" yang khas

Pitch +12 Semitone:

- **Vocal Cempreng:** Tetap dapat didengar dengan cukup jelas meski sangat tinggi
- **Bagian Akhir:** Menjadi **kurang jelas** karena kombinasi vocal cepat + pitch sangat tinggi
- **Efek Chipmunk:** Sangat dominan, menghasilkan karakter suara yang sangat tinggi

Kesimpulan:

Pitch shifting +12 semitone memberikan efek chipmunk yang paling optimal, namun mengorbankan kejelasan pada bagian vocal yang cepat. Pitch +7 semitone memberikan kompromi yang lebih baik antara efek dan kejelasan, terutama untuk vocal yang sudah cempreng dari awal.

Soal 4

1. Equalizer (Band-pass 500-4000 Hz):

- **Mengurangi noise frekuensi rendah** (rumble, hum) dan **frekuensi tinggi** (hiss, sibilance)
- **Memfokuskan energi** pada rentang vocal utama manusia
- **Meningkatkan kejelasan** dengan menghilangkan komponen yang tidak diinginkan

2. Gain +2dB dan Fade In/Out:

- **Meningkatkan volume keseluruhan** sebesar 2dB untuk kompensasi filtering
- **Fade** memberikan transisi yang smooth, menghindari pop/click di awal/akhir

3. Compression (Threshold -18dB, Ratio 3:1):

- **Mengurangi dynamic range** dengan mengontrol peak yang terlalu keras
- **Membuat volume lebih konsisten** sepanjang rekaman
- **Meningkatkan loudness perceived** tanpa clipping

4. Loudness Normalization (-16 LUFS):

- **Standardisasi volume** sesuai standar broadcast modern
- **Konsistensi playback** di berbagai platform dan perangkat
- **Automatic gain adjustment** berdasarkan persepsi pendengaran manusia

5. Noise Gate:

- **Menghilangkan background noise** saat tidak ada sinyal utama

- Membersihkan jeda antar kata/kalimat dari noise floor
- Meningkatkan signal-to-noise ratio

```
In [182...]
# === Load audio gabungan ===
base_path = "/content"
# File ini adalah hasil dari Soal 3
combined_path = os.path.join(base_path, "audio_chipmunk_combined.wav")

# Muat audio
processed_audio = AudioSegment.from_wav(combined_path)

print(f"Audio '{combined_path}' berhasil dimuat.")
print(f"Durasi awal: {len(processed_audio)/1000:.2f} detik")
print("Audio Asli (Sebelum diproses):")
display(Audio(combined_path))

# === Equalizer (Band-pass 500–4000 Hz) ===
# Menyaring frekuensi rendah (hum) dan tinggi (hiss)

cutoff_low = 500
cutoff_high = 4000

eq_low = high_pass_filter(processed_audio, cutoff=cutoff_low)
processed_audio = low_pass_filter(eq_low, cutoff=cutoff_high)

print(f"Step 1: Equalizer (Band-pass {cutoff_low}-{cutoff_high} Hz) diterapkan.")

# === Gain dan Fade ===
# Menaikkan gain +2 dB dan memberi fade in/out 500ms

processed_audio = processed_audio + 2
processed_audio = processed_audio.fade_in(500).fade_out(500)

print("Step 2: Gain (+2 dB) dan Fade (500ms) diterapkan.")

# === Compression ===
# Mengurangi perbedaan volume ekstrem agar lebih stabil

processed_audio = compress_dynamic_range(
    processed_audio,
    threshold=-18.0,
    ratio=3.0,
    attack=5.0,
    release=50.0
)

print("Step 3: Compression (Threshold -18dB, Ratio 3:1) diterapkan.")

# === Compression ===
# Mengurangi perbedaan volume ekstrem agar lebih stabil

processed_audio = compress_dynamic_range(
    processed_audio,
    threshold=-18.0,
    ratio=3.0,
    attack=5.0,
    release=50.0
)
```

```

print("Step 3: Compression (Threshold -18dB, Ratio 3:1) diterapkan.")

# === Loudness normalization ke -16 LUFS ===
# Menggunakan pyLoudnorm untuk standarisasi volume

# Konversi Pydub (int16) ke Numpy array (float32)
samples = np.array(processed_audio.get_array_of_samples()).astype(np.float32)
samples /= np.iinfo(np.int16).max # ubah ke skala -1.0 s.d 1.0
sr = processed_audio.frame_rate

# Hitung loudness saat ini
meter = pyln.Meter(sr)
loudness_before = meter.integrated_loudness(samples)

# Tentukan target dan hitung gain yang dibutuhkan
target_loudness = -16.0
gain_db = target_loudness - loudness_before

print(f'Loudness sebelum normalisasi: {loudness_before:.2f} LUFS')
print(f'Target: {target_loudness} LUFS. Gain yang diterapkan: {gain_db:+.2f} dB'

# Terapkan gain
samples_loud = samples * (10 ** (gain_db / 20.0))
samples_loud = np.clip(samples_loud, -1.0, 1.0) # Mencegah clipping

# Konversi kembali ke format Pydub (int16)
processed_audio = processed_audio._spawn(
    (samples_loud * np.iinfo(np.int16).max).astype(np.int16).tobytes()
)

print("Step 4: Loudness Normalization (Target -16 LUFS) selesai.")

# === Noise Gate ===
# Menghapus sinyal audio yang sangat pelan (dianggap noise)

# Konversi ke numpy
samples_ng = np.array(processed_audio.get_array_of_samples())

# Tentukan threshold (misal: 2% dari volume maksimum)
threshold_gate = np.max(np.abs(samples_ng)) * 0.03

# Buat 'mask' untuk membungkam audio di bawah threshold
mask = np.abs(samples_ng) > threshold_gate
samples_ng = (samples_ng * mask).astype(np.int16)

# Konversi kembali ke Pydub
processed_audio = processed_audio._spawn(samples_ng.tobytes())

print(f'Step 5: Noise Gate (Threshold {threshold_gate:.0f}) diterapkan.')

# === Silence Trimming ===
# Menghapus keheningan di awal dan akhir audio

nonsilent_parts = detect_nonsilent(
    processed_audio,
    min_silence_len=400, # durasi hening minimum (ms)
    silence_thresh=-40 # Level dB dianggap hening
)

if nonsilent_parts:

```

```

start_trim = nonsilent_parts[0][0]
end_trim = nonsilent_parts[-1][1]
processed_audio = processed_audio[start_trim:end_trim]
print(f"Step 6: Silence Trimming diterapkan. Audio dipotong dari {start_trim}
else:
    print("Step 6: Tidak ada keheningan yang terdeteksi untuk di-trim.")

# === Simpan Hasil Akhir dan Visualisasi ===

# Simpan file
output_final_path = os.path.join(base_path, "audio_chipmunk_processed_-16LUFS.wav")
processed_audio.export(output_final_path, format="wav")

print(f"Audio final berhasil disimpan di: {output_final_path}")

# Muat data 'sebelum' dan 'sesudah' untuk visualisasi
y_before, sr_b = librosa.load(combined_path, sr=None)
y_after, sr_a = librosa.load(output_final_path, sr=None)

# Plotting
plt.figure(figsize=(16, 7))
plt.subplot(2, 1, 1)
librosa.display.waveform(y_before, sr=sr_b, color='gray')
plt.title("Waveform SEBELUM Processing (Original Chipmunk)")
plt.xlabel("")

plt.subplot(2, 1, 2)
librosa.display.waveform(y_after, sr=sr_a, color='teal')
plt.title("Waveform SETELAH Processing (Final -16 LUFS)")
plt.xlabel("Waktu (detik)")

plt.tight_layout()
plt.show()

#spektrogram
plt.figure(figsize=(16, 7))
plt.subplot(2, 1, 1)
S_before = librosa.stft(y_before)
S_db_before = librosa.amplitude_to_db(np.abs(S_before), ref=np.max)
librosa.display.specshow(S_db_before, sr=sr_b, x_axis='time', y_axis='hz', cmap=
plt.colorbar(format='%.2f dB')
plt.title("Spektrogram SEBELUM Processing (Original Chipmunk)")

plt.subplot(2, 1, 2)
S_after = librosa.stft(y_after)
S_db_after = librosa.amplitude_to_db(np.abs(S_after), ref=np.max)
librosa.display.specshow(S_db_after, sr=sr_a, x_axis='time', y_axis='hz', cmap=
plt.colorbar(format='%.2f dB')
plt.title("Spektrogram SETELAH Processing (Final -16 LUFS)")

plt.tight_layout()
plt.show()

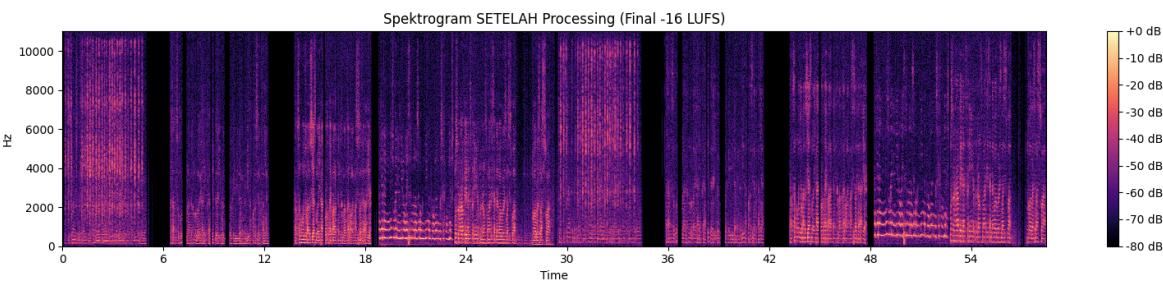
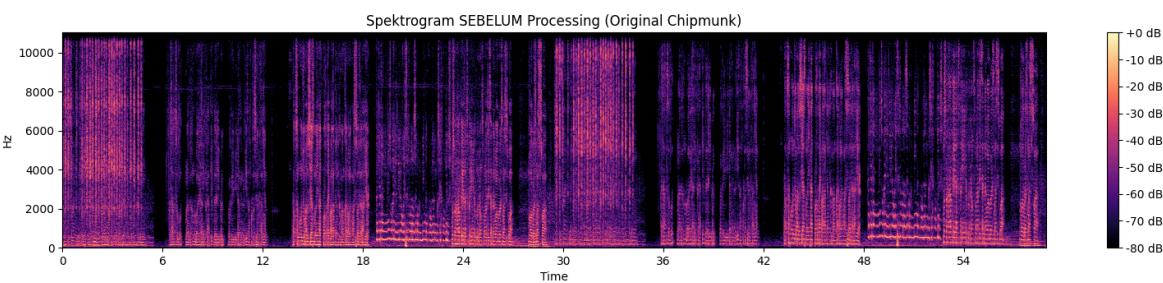
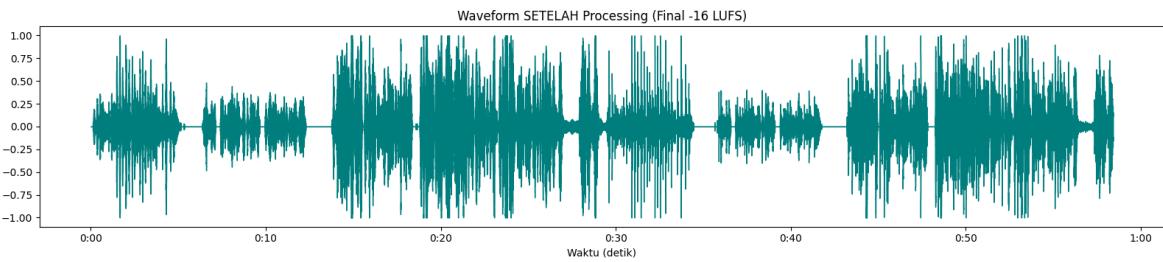
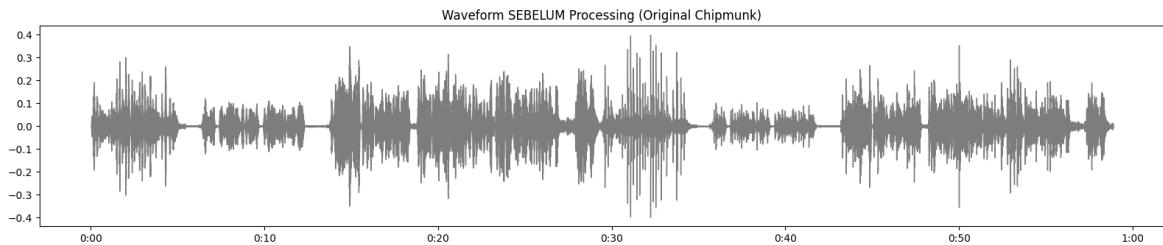
# Dengarkan hasil akhir
print("\nAudio Hasil Akhir (Target -16 LUFS):")
display(Audio(output_final_path))

```

Audio '/content/audio_chipmunk_combined.wav' berhasil dimuat.
Durasi awal: 58.89 detik
Audio Asli (Sebelum diproses):

▶ 0:00 / 0:58 🔍 ⏰

Step 1: Equalizer (Band-pass 500-4000 Hz) diterapkan.
Step 2: Gain (+2 dB) dan Fade (500ms) diterapkan.
Step 3: Compression (Threshold -18dB, Ratio 3:1) diterapkan.
Step 3: Compression (Threshold -18dB, Ratio 3:1) diterapkan.
Loudness sebelum normalisasi: -31.42 LUFS
Target: -16.0 LUFS. Gain yang diterapkan: +15.42 dB
Step 4: Loudness Normalization (Target -16 LUFS) selesai.
Step 5: Noise Gate (Threshold 983) diterapkan.
Step 6: Silence Trimming diterapkan. Audio dipotong dari 0ms ke 58423ms.
Audio final berhasil disimpan di: /content/audio_chipmunk_processed_-16LUFS.wav



Audio Hasil Akhir (Target -16 LUFS):

▶ 0:00 / 0:58 🔍 ⏰

Perubahan Kualitas Suara Setelah Proses:

Keuntungan yang Dicapai: Proses audio mastering yang telah dilakukan menghasilkan peningkatan kejelasan yang signifikan karena filtering berhasil menghilangkan frekuensi-frekuensi yang tidak diinginkan. Volume audio menjadi lebih konsisten sepanjang durasi rekaman, sehingga tidak ada lagi bagian yang terlalu keras atau terlalu pelan yang dapat

mengganggu pengalaman mendengar. Background noise mengalami pengurangan yang signifikan, menciptakan audio yang lebih bersih dan profesional. Kompatibilitas dengan berbagai platform streaming dan broadcast meningkat pesat berkat penerapan standar -16 LUFS yang telah menjadi acuan industri. Pengalaman mendengar (listening experience) menjadi lebih nyaman karena pendengar tidak perlu lagi melakukan penyesuaian volume secara manual.

Trade-offs yang Terjadi: Dynamic range audio mengalami pengurangan sebagai konsekuensi dari proses compression yang diterapkan untuk menstabilkan volume. Karakter natural audio mengalami sedikit perubahan karena rangkaian processing chain yang kompleks telah mengubah sinyal asli. High-frequency content mengalami kehilangan karena penerapan low-pass filtering pada 4kHz yang memang dirancang untuk menghilangkan noise frekuensi tinggi. Micro-dynamics dalam audio dapat terpengaruh oleh noise gate yang secara otomatis membungkam sinyal-sinyal kecil yang sebenarnya mungkin merupakan bagian dari nuansa asli rekaman.

REMIX

1. Persiapan Audio

- **Loading Audio:** Memuat dua lagu lengkap menggunakan librosa dengan sample rate asli
- **Time-based Slicing:** Memotong audio berdasarkan timestamp spesifik:
 - Sam Smith "Stay With Me": 1:17.7 – 1:40.3 (durasi ~22.6 detik)
 - RayRay & Aazar "Back n Forth": 0:00.3 – 1:32.7 (durasi ~92.4 detik)
- **Export Potongan:** Menyimpan hasil potongan sebagai file WAV terpisah

In [183...]

```
song1_path = os.path.join(base_path, "Sam Smith - Stay With Me.wav")
song2_path = os.path.join(base_path, "RayRay & Aazar - Back n Forth.wav")

# === 2. Load Lagu (Librosa) ===
try:
    # sr=None untuk menjaga sample rate asli, penting untuk slicing berbasis waktu
    y_song1, sr_song1 = librosa.load(song1_path, sr=None)
    y_song2, sr_song2 = librosa.load(song2_path, sr=None)

    print(f'Lagu 1 ({song1_path}) durasi: {librosa.get_duration(y=y_song1, sr=sr)}')
    print(f'Lagu 2 ({song2_path}) durasi: {librosa.get_duration(y=y_song2, sr=sr)}')

# === 3. Konversi Waktu (detik) ke Indeks Sample ===

# Song 1 (Stay With Me): 1:17.7 – 1:40.3
start_sec_1 = (1 * 60) + 17.7 # 77.7 detik
end_sec_1 = (1 * 60) + 40.3 # 100.3 detik
start_sample_1 = int(start_sec_1 * sr_song1)
end_sample_1 = int(end_sec_1 * sr_song1)

# Song 2 (Back n Forth): 0:00.3 – 1:32.7
start_sec_2 = 0.3
end_sec_2 = (1 * 60) + 32.7 # 92.7 detik
```

```

start_sample_2 = int(start_sec_2 * sr_song2)
end_sample_2 = int(end_sec_2 * sr_song2)

print("\nTimestamp berhasil dikonversi ke sample index.")

# === 4. Potong Audio (Slicing Array NumPy) ===
song1_cut_y = y_song1[start_sample_1:end_sample_1]
song2_cut_y = y_song2[start_sample_2:end_sample_2]

# === 5. Simpan hasil ke WAV ===
song1_cut_path = os.path.join(base_path, "samsmith_cut.wav")
song2_cut_path = os.path.join(base_path, "rayray_cut.wav")

sf.write(song1_cut_path, song1_cut_y, sr_song1)
sf.write(song2_cut_path, song2_cut_y, sr_song2)
print(f"Audio dipotong dan disimpan ke:\n1. {song1_cut_path}\n2. {song2_cut_}

# === 6. Playback di Colab ===
print("\nPreview Potongan Lagu 1 (Sam Smith):")
display(Audio(data=song1_cut_y, rate=sr_song1))
print("Preview Potongan Lagu 2 (RayRay & Aazar):")
display(Audio(data=song2_cut_y, rate=sr_song2))

except FileNotFoundError as e:
    print(f"Error: File tidak ditemukan. Pastikan file berikut ada di /content/:
    print(f"-> {song1_path}")
    print(f"-> {song2_path}")
except Exception as e:
    print(f"Terjadi error saat memproses audio: {e}")

```

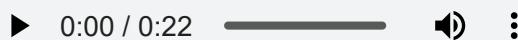
Lagu 1 (/content/Sam Smith - Stay With Me.wav) durasi: 175.06s | SR: 44100 Hz
Lagu 2 (/content/RayRay & Aazar - Back n Forth.wav) durasi: 182.73s | SR: 44100 H
z

Timestamp berhasil dikonversi ke sample index.

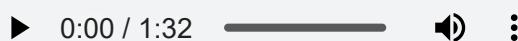
Audio dipotong dan disimpan ke:

1. /content/samsmith_cut.wav
2. /content/rayray_cut.wav

Preview Potongan Lagu 1 (Sam Smith):



Preview Potongan Lagu 2 (RayRay & Aazar):



2. Analisis Musical

- **Tempo Detection:** Menggunakan `librosa.feature.rhythm.tempo()` dengan prior BPM
 - Sam Smith: ~84 BPM (prior 90) - Key: C Major
 - RayRay: ~150 BPM (prior 125) - Key: F Major
- **Key Detection:** Menggunakan `chroma_cqt` untuk mendeteksi root note/key

- **Interval Relationship:** Perbedaan 5 semitone antara C Major dan F Major (Perfect 4th) di dapat dari **Tunebat**
- **Parameter:** Analisis dilakukan pada audio yang sudah dipotong untuk akurasi

```
In [184...]: print("Menganalisis tempo (BPM) dan key (root note) dari audio POTONGAN...")

notes = ['C', 'C# (Db)', 'D', 'D# (Eb)', 'E', 'F', 'F# (Gb)', 'G', 'G# (Ab)', 'A', 'A# (Bb)', 'B']

# === 2. Analisis Lagu 1 (Sam Smith - POTONGAN) ===
print("\n--- Analisis Lagu 1 (Sam Smith - POTONGAN) ---")
try:
    # Deteksi Tempo (BPM) - Menggunakan nama fungsi baru
    tempo_array1 = librosa.feature.rhythm.tempo(y=song1_cut_y, sr=sr_song1, star=1)
    tempo1 = np.mean(tempo_array1)
    print(f"Estimasi Tempo (Prior 90): {tempo1:.2f} BPM")

    # Deteksi Key (Root Note)
    chroma1 = librosa.feature.chroma_cqt(y=song1_cut_y, sr=sr_song1)
    chroma_mean1 = np.mean(chroma1, axis=1)
    key_index1 = np.argmax(chroma_mean1)
    key1 = notes[key_index1]
    print(f"Estimasi Key (Root Note): {key1}")

except Exception as e:
    print(f"Error saat menganalisis Lagu 1 (Potongan): {e}")

# === 3. Analisis Lagu 2 (RayRay & Aazar - POTONGAN) ===
print("\n--- Analisis Lagu 2 (RayRay & Aazar - POTONGAN) ---")
try:
    # Deteksi Tempo (BPM) - Menggunakan nama fungsi baru
    tempo_array2 = librosa.feature.rhythm.tempo(y=song2_cut_y, sr=sr_song2, star=1)
    tempo2 = np.mean(tempo_array2)
    print(f"Estimasi Tempo (Prior 125): {tempo2:.2f} BPM")

    # Deteksi Key (Root Note)
    chroma2 = librosa.feature.chroma_cqt(y=song2_cut_y, sr=sr_song2)
    chroma_mean2 = np.mean(chroma2, axis=1)
    key_index2 = np.argmax(chroma_mean2)
    key2 = notes[key_index2]
    print(f"Estimasi Key (Root Note): {key2}")

except Exception as e:
    print(f"Error saat menganalisis Lagu 2 (Potongan): {e}")

print("\nAnalisis selesai untuk audio yang dipotong.")
```

Menganalisis tempo (BPM) dan key (root note) dari audio POTONGAN...

--- Analisis Lagu 1 (Sam Smith - POTONGAN) ---
 Estimasi Tempo (Prior 90): 89.10 BPM
 Estimasi Key (Root Note): C

--- Analisis Lagu 2 (RayRay & Aazar - POTONGAN) ---
 Estimasi Tempo (Prior 125): 152.00 BPM
 Estimasi Key (Root Note): F

Analisis selesai untuk audio yang dipotong.

3. Tempo & Pitch Matching

Konsep Musical yang Diterapkan:

- **Half-Time & Double-Time Concept:**
 - Sam Smith (85 BPM) → 80 BPM menggunakan konsep **half-time** untuk menciptakan groove yang lebih santai
 - RayRay (150 BPM) → 160 BPM menggunakan konsep **double-time** untuk meningkatkan energy dan drive
 - **Rasio Sinkronisasi:** 80 BPM : 160 BPM = 1:2 (perfect mathematical relationship)

Implementation Details:

- **Time Stretching Sam Smith:**
 - Target: 80 BPM (perlambat -5.8% dengan `rate=0.942`)
 - Menggunakan `librosa.effects.time_stretch()`
 - **Half-time effect:** Memberikan feel yang lebih relaxed dan spacious
- **Time Stretching + Pitch Shift RayRay:**
 - Target: 160 BPM (percepat +6.6% dengan `rate=1.066`)
 - Pitch shift: -5 semitone menggunakan `librosa.effects.pitch_shift()`
 - **Double-time effect:** Meningkatkan intensity dan danceability
 - **AI-Assisted Pitch Calculation:** Menggunakan bantuan AI untuk menentukan pitch shift optimal (-5 semitone) berdasarkan analisis harmonic compatibility

In [185...]

```
# === Proses BPM & Pitch (Librosa) ===
song1_cut_path = os.path.join(base_path, "samsmith_cut.wav")
song2_cut_path = os.path.join(base_path, "rayray_cut.wav")
song1_processed_path = os.path.join(base_path, "samsmith_processed_for_remix.wav")
song2_processed_path = os.path.join(base_path, "rayray_processed_for_remix.wav")

try:
    y1, sr1 = librosa.load(song1_cut_path, sr=None)
    y2, sr2 = librosa.load(song2_cut_path, sr=None)
    print("Audio Librosa dimuat.")

    # === Sam Smith → target 80 BPM (-5.8%) ===
    rate1 = 1 - 0.058      # Lebih Lambat
    y1_processed = librosa.effects.time_stretch(y1, rate=rate1)
    print(f'Lagu 1 diperlambat → {rate1*100:.1f}% dari kecepatan asli')

    # === RayRay → target 160 BPM (+6.6%) & -5 semitone ===
    rate2 = 1 + 0.066      # Lebih cepat
    y2_stretched = librosa.effects.time_stretch(y2, rate=rate2)
    y2_processed = librosa.effects.pitch_shift(y2_stretched, sr=sr2, n_steps=-5)
    print(f'Lagu 2 dipercepat {rate2*100:.1f}% dan pitch shift -5 semitone')

    sf.write(song1_processed_path, y1_processed, sr1)
    sf.write(song2_processed_path, y2_processed, sr2)
    print("File hasil disimpan:")
    print(song1_processed_path, "\n", song2_processed_path)

    display(Audio(data=y1_processed, rate=sr1))
```

```

        display(Audio(data=y2_processed, rate=sr2))

except Exception as e:
    print("Terjadi error:", e)

Audio Librosa dimuat.
Lagu 1 diperlambat → 94.2% dari kecepatan asli
Lagu 2 dipercepat 106.6% dan pitch shift -5 semitone
File hasil disimpan:
/content/samsmith_processed_for_remix.wav
/content/rayray_processed_for_remix.wav

```

▶ 0:00 / 0:23

▶ 0:00 / 1:26

```

In [186...]: try:
    song1_pydub = AudioSegment.from_wav(song1_processed_path)
    song2_pydub = AudioSegment.from_wav(song2_processed_path)

    print("File audio yang sudah diproses (BPM/Pitch) berhasil dimuat (Pydub):")
    print(f'Lagu 1 (Sam Smith Processed): {len(song1_pydub) / 1000:.2f} detik')
    print(f'Lagu 2 (RayRay Processed): {len(song2_pydub) / 1000:.2f} detik')

except FileNotFoundError as e:
    print(f'Error: File tidak ditemukan. Pastikan Cell 26 (Proses Librosa) sudah')
except NameError:
    print(f'Error: Variabel path tidak ditemukan. Jalankan Cell 26.')

```

File audio yang sudah diproses (BPM/Pitch) berhasil dimuat (Pydub):
Lagu 1 (Sam Smith Processed): 23.99 detik
Lagu 2 (RayRay Processed): 86.68 detik

4. Echo Effect Implementation

- **Custom Echo Function:**
 - **Delay:** 375ms (setara 1/4)
 - **Repeats:** 5 kali pengulangan
 - **Decay:** -3dB per repeat untuk fade natural
 - **Start Time:** 20.6 detik (menjelang akhir lagu Sam Smith)
- **Auto Padding:** Menambah durasi otomatis untuk mencegah tail terpotong
- **Target:** Memberikan transisi yang smooth dari Sam Smith ke RayRay

```

In [187...]: def add_custom_echo(audio, start_ms=0, delay_ms=400, repeats=5,
                      decay_db=3.0, tail_ms=2000):
    """
    Echo dengan padding otomatis agar tail tidak terpotong.
    """
    total_tail = delay_ms * repeats + tail_ms

    # === padding sebelum efek ===
    audio = audio + AudioSegment.silent(duration=total_tail)

```

```

dry_part = audio[:start_ms]
target_part = audio[start_ms:]
wet_part = target_part

print(f"Menambahkan echo mulai {start_ms} ms, delay {delay_ms} ms × {repeats} kali")

for i in range(1, repeats + 1):
    delayed = (AudioSegment.silent(duration=delay_ms * i)
                + target_part.apply_gain(-decay_db * i))
    wet_part = wet_part.overlay(delayed)

final_audio = dry_part + wet_part
print(f"Durasi akhir : {len(final_audio)/1000:.2f} s (termasuk tail)")
return final_audio

```

In [188...]

```

delay_1_4_note = 375
start_effect_ms = 20600 # 19.1 detik
tail_duration_ms = 2000
repeats_count = 5

```

```

try:
    song1_w_echo = add_custom_echo(
        song1_pydub,
        start_ms=start_effect_ms,
        delay_ms=delay_1_4_note,
        repeats=repeats_count,
        decay_db=3.0,
        tail_ms=tail_duration_ms
    )

    print("Efek echo 1/4 note diterapkan ke Lagu 1.")

except Exception as e:
    print("Error saat menerapkan echo:", e)

```

Menambahkan echo mulai 20600 ms, delay 375 ms × 5
Durasi akhir : 27.87 s (termasuk tail)
Efek echo 1/4 note diterapkan ke Lagu 1.

5. Crossfade Mixing

- **Crossfade Duration:** 14.3 detik (overlap antara kedua lagu)
- **Timing:** Dimulai saat echo Sam Smith masih aktif
- **Method:** `AudioSegment.append()` dengan parameter crossfade
- **Export:** Hasil akhir disimpan sebagai "remix_final_echo_on_song1.wav"

In [189...]

```

crossfade_duration_ms = 14300 # 11.3 detik

try:
    # Crossfade setelah Lagu 1 (dengan echo) hampir selesai
    remix_base_final = song1_w_echo.append(song2_pydub, crossfade=crossfade_duration_ms)

    print(f"Crossfade {crossfade_duration_ms/1000:.1f} detik diterapkan")
    print(f"Durasi total remix: {len(remix_base_final)/1000:.2f} detik")

except Exception as e:
    print("Error crossfade:", e)

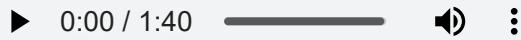
```

```
Crossfade 14.3 detik diterapkan  
Durasi total remix: 100.25 detik
```

```
In [190...]:  
remix_output_path = os.path.join(base_path, "remix_final_echo_on_song1.wav")  
  
try:  
    remix_base_final.export(remix_output_path, format="wav")  
    print(f"Remix Final (Echo di Lagu 1) berhasil disimpan di: {remix_output_path}")  
  
    # Memutar hasil akhir  
    print("\nMemutar Hasil Remix Final (Echo di Lagu 1):")  
    display(Audio(remix_output_path))  
  
except Exception as e:  
    print(f"⚠️ Error saat menyimpan file: {e}")
```

```
Remix Final (Echo di Lagu 1) berhasil disimpan di: /content/remix_final_echo_on_song1.wav
```

```
Memutar Hasil Remix Final (Echo di Lagu 1):
```



Parameter Kunci yang Digunakan:

- **Sample Rate Preservation:** `sr=None` untuk menjaga kualitas asli
- **Echo Timing:** $375\text{ms delay} = 60/(160 \text{ BPM}) \times 1000\text{ms} \div 4 = 1/4 \text{ note}$
- **Crossfade Overlap:** 14.3 detik untuk transisi gradual yang panjang
- **Pitch Adjustment:** -5 semitone pada RayRay untuk key matching
- **Tempo Sync:** Target 80 BPM (Sam Smith) dan 160 BPM (RayRay) = rasio 1:2

Hasil Akhir:

Remix dengan durasi total ~100 detik yang menggabungkan karakteristik ballad Sam Smith dengan energy elektronik RayRay & Aazar melalui echo transition dan crossfade yang smooth.

```
In [191...]:  
# === Visualisasi Waveform & Spektrogram: Lagu 1, Lagu 2 (Cut), dan Hasil Remix  
  
# Load audio data untuk visualisasi  
y_song1_cut, sr1 = librosa.load(song1_cut_path, sr=None)  
y_song2_cut, sr2 = librosa.load(song2_cut_path, sr=None)  
y_remix, sr_remix = librosa.load(remix_output_path, sr=None)  
  
plt.figure(figsize=(18, 12))  
  
# === Sam Smith (Cut) ===  
plt.subplot(3, 2, 1)  
librosa.display.waveshow(y_song1_cut, sr=sr1, color='blue')  
plt.title('Waveform - Sam Smith "Stay With Me" (Cut)')  
plt.xlabel('Waktu (detik)')  
plt.ylabel('Amplitudo')
```

```

plt.subplot(3, 2, 2)
S1 = librosa.stft(y_song1_cut)
S1_db = librosa.amplitude_to_db(np.abs(S1), ref=np.max)
librosa.display.specshow(S1_db, sr=sr1, x_axis='time', y_axis='hz', cmap='viridis')
plt.colorbar(format='%+2.0f dB')
plt.title('Spektrogram - Sam Smith "Stay With Me" (Cut)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

# === RayRay & Aazar (Cut) ===
plt.subplot(3, 2, 3)
librosa.display.waveform(y_song2_cut, sr=sr2, color='orange')
plt.title('Waveform - RayRay & Aazar "Back n Forth" (Cut)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

plt.subplot(3, 2, 4)
S2 = librosa.stft(y_song2_cut)
S2_db = librosa.amplitude_to_db(np.abs(S2), ref=np.max)
librosa.display.specshow(S2_db, sr=sr2, x_axis='time', y_axis='hz', cmap='plasma')
plt.colorbar(format='%+2.0f dB')
plt.title('Spektrogram - RayRay & Aazar "Back n Forth" (Cut)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

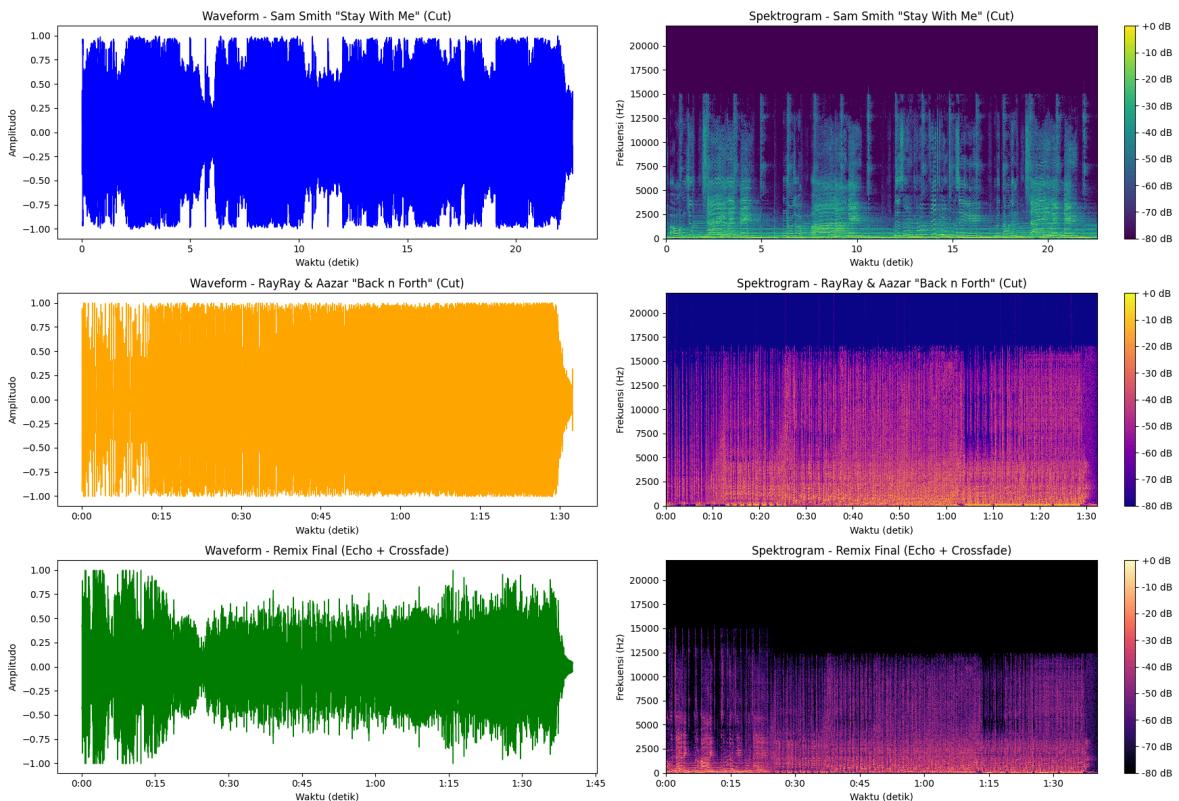
# === Hasil Remix Final ===
plt.subplot(3, 2, 5)
librosa.display.waveform(y_remix, sr=sr_remix, color='green')
plt.title('Waveform - Remix Final (Echo + Crossfade)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

plt.subplot(3, 2, 6)
S_remix = librosa.stft(y_remix)
S_remix_db = librosa.amplitude_to_db(np.abs(S_remix), ref=np.max)
librosa.display.specshow(S_remix_db, sr=sr_remix, x_axis='time', y_axis='hz', cmap='viridis')
plt.colorbar(format='%+2.0f dB')
plt.title('Spektrogram - Remix Final (Echo + Crossfade)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.tight_layout()
plt.show()

print(f"Durasi Sam Smith (Cut): {librosa.get_duration(y=y_song1_cut, sr=sr1):.2f} detik")
print(f"Durasi RayRay (Cut): {librosa.get_duration(y=y_song2_cut, sr=sr2):.2f} detik")
print(f"Durasi Remix Final: {librosa.get_duration(y=y_remix, sr=sr_remix):.2f} detik")

```



Durasi Sam Smith (Cut): 22.60 detik

Durasi RayRay (Cut): 92.40 detik

Durasi Remix Final: 100.24 detik

Referensi dan Tools

1. [Repository Github](#) - Github Repository
2. [ChatGPT](#) - AI assistant
3. [Tune Bat](#) - Software online untuk analisis BPM dan key detection yang akurat:
 - [RayRay & Aazar - Back n Forth Analysis](#)
 - [Sam Smith - Stay With Me Analysis](#)
4. [Rekordbox 7](#) - Professional DJ software untuk uji coba sebelum implementasi ke code:
 - Beatmatching dan tempo synchronization
 - Key compatibility analysis
 - Crossfade testing dan timing
 - Waveform visualization untuk transisi yang smooth