

# JARINGAN MULTIMEDIA



PENGAJAR :

I Dewa Made Bayu Atmaja Darmawan,S.Kom.,M.Cs.

DISUSUN OLEH:

I Kadek Peri Arta Wijaya (2208561007)

PROGRAM STUDI INFORMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS UDAYANA  
2025

**Respository Code :** [https://github.com/periartaa/Tugas1-JaringanMultimedai\\_2025.git](https://github.com/periartaa/Tugas1-JaringanMultimedai_2025.git)

**Tool & perangkat yang digunakan:**

1. Vscode
2. AI : [Chat GPT](#) & [CludeAI](#)

**Modifikasi:**

1. Menambahkan fitur error handling  
Error handling bertujuan untuk menangani kesalahan atau exception yang terjadi di server dan client secara elegan dan mencegah server dari crash yang tidak diinginkan. Fitur ini akan menangkap error seperti kesalahan input dari pengguna, masalah koneksi database, atau masalah pada proses server dan memberikan respons yang informatif kepada pengguna atau log untuk analisis lebih lanjut.
2. Menambahkan fitur log pada server.py  
Fitur logging digunakan untuk mencatat aktivitas atau kejadian-kejadian penting dalam server, seperti permintaan yang masuk, waktu pemrosesan, error, atau event lainnya. Dengan log ini, developer dapat melacak dan menganalisis masalah yang terjadi di server dan melakukan debug lebih efektif. Log juga bisa digunakan untuk memantau kinerja aplikasi.
3. Menambahkan fitur record pada client.py  
Fitur record digunakan untuk merekam atau mencatat data percakapan/suara dari client dan server yang terjadi pada sisi klien.

### **Cara Kerja**

Sistem ini merupakan aplikasi streaming audio real-time berbasis Python yang dapat beroperasi melalui protokol UDP atau TCP. Sistem terdiri dari dua komponen utama:

1. Klien (*client.py*): Merekam audio dari mikrofon, mengirimkannya melalui jaringan, dan dapat menyimpan rekaman secara lokal.
2. Server (*server.py*): Menerima paket audio dari klien, memutarnya, dan memantau metrik kinerja jaringan.

**Fungsi Klien (*client.py*)**

Kelas *AudioClient* menangani:

1. Pengambilan Audio: Menggunakan PyAudio untuk merekam audio dari perangkat mikrofon default dalam potongan 10ms.
2. Jaringan: Mengirim data audio melalui UDP atau TCP ke server yang ditentukan.
3. Manajemen Rekaman: Dapat merekam audio ke file WAV yang disimpan dalam folder "Recording" dengan fungsi:
  - Merekam untuk durasi tertentu
  - Mulai/stop rekaman secara manual

**Fungsi Server (*server.py*)**

Kelas *AdvancedAudioServer* menangani:

1. Penerimaan Jaringan: Mendengarkan data audio yang masuk melalui UDP atau TCP
2. Pemutaran Audio: Memutar audio yang diterima melalui perangkat output default
3. Pemantauan Kinerja: Melacak kehilangan paket dan integritas urutan
4. Pencatatan Log: Memelihara log detail dengan rotasi untuk debugging dan statistik

## Cara Kerja Kode

### 1. Penanganan Error (Error Handling)

Pada Server.py

```
try:
    # Kode yang mungkin menimbulkan error.....
except Exception as e:
    logging.error(f"Kesalahan menerima data: {e}")
    logging.error(traceback.format_exc())
    return False
```

Server menggunakan blok try-except di berbagai fungsi untuk menangkap error:

1. Setup Socket:
  - Menangkap error saat membuat dan mengkonfigurasi socket
  - Mencatat error dengan logging.error lalu keluar dengan sys.exit(1)
  - Logging traceback penuh untuk debugging
2. Setup Audio:
  - Menangkap error dari PyAudio
  - Keluar program dengan sys.exit(1) jika tidak dapat memulai audio
3. Receive Data:
  - Menangkap error saat menerima dan memproses paket
  - Mencatat error dengan detail dan melanjutkan eksekusi
  - Mengembalikan nilai False agar loop utama dapat bereaksi
4. Run Method:
  - Menggunakan try-finally untuk memastikan pembersihan sumber daya
  - Menangkap KeyboardInterrupt untuk penghentian yang bersih

Pada Client.py

```
try:
    # Kode yang mungkin menimbulkan error
except pyaudio.PyAudioError as e:
    print(f"PyAudio Error: {e}")
    sys.exit(1)
```

Klien menggunakan pendekatan serupa:

1. Setup Audio dan Socket:
  - Error handling spesifik untuk PyAudio dan Socket
  - Keluar program dengan sys.exit(1) saat terjadi error kritis

2. Saving Recording:
  - Menangkap exception saat menyimpan file
  - Menampilkan pesan error tanpa menghentikan program
3. Send Audio:
  - Menangkap queue.Empty untuk non-blocking
  - Menangkap socket.error saat pengiriman gagal
4. Cleanup Method:
  - Menangkap semua exception selama pembersihan
  - Memastikan semua sumber daya dibebaskan dalam kondisi apapun

## 2. Sistem Logging pada Server.py

```
# Setup Logging dengan RotatingFileHandler
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

formatter = logging.Formatter("%(asctime)s - %(levelname)s - %(message)s")

# Simpan log ke file dengan batasan ukuran 5MB, simpan 3 backup log lama
file_handler = RotatingFileHandler("audio_stream.log", maxBytes=5*1024*1024,
backupCount=3)
file_handler.setFormatter(formatter)

console_handler = logging.StreamHandler()
console_handler.setFormatter(formatter)

logger.addHandler(file_handler)
logger.addHandler(console_handler)
```

Server mengimplementasikan sistem logging komprehensif:

1. RotatingFileHandler:
  - Menyimpan log ke file "audio\_stream.log"
  - Maksimum ukuran file 5MB sebelum rotasi
  - Menyimpan hingga 3 file backup (.log.1, .log.2, .log.3)
  - Mencegah penggunaan disk yang berlebihan
2. Level Logging:
  - DEBUG: Informasi detail untuk debugging (sequence number, ukuran data)
  - INFO: Informasi status sistem normal (koneksi baru, statistik)
  - WARNING: Kondisi tidak normal (paket tidak berurutan)
  - ERROR: Error yang perlu perhatian (koneksi terputus, kegagalan operasi)
3. Format Log:
  - Timestamp: Waktu kejadian
  - Level: Tingkat keparahan (DEBUG/INFO/WARNING/ERROR)

- Pesan: Deskripsi kejadian
4. Penggunaan dalam Kode:
    - Mencatat informasi koneksi: `logging.info(f"Koneksi dari {self.client_address}")`
    - Mencatat statistik: `logging.info(f"Statistik: Total={self.total_packets}...")`
    - Mencatat warning untuk paket hilang: `logging.warning(f"Paket tidak berurutan!...")`
    - Mencatat error dengan traceback: `logging.error(traceback.format_exc())`

### 3. Fungsi Rekaman pada Client.py

```
def start_recording(self, duration=None):
    """Mulai menyimpan frame audio"""
    self.is_recording = True
    self.recorded_frames = []
    print("Recording started.")

    # Jika ada durasi, gunakan threading untuk stop otomatis
    if duration:
        threading.Thread(target=self._stop_after_duration, args=(duration,),
            daemon=True).start()
```

Klien memiliki sistem rekaman lengkap:

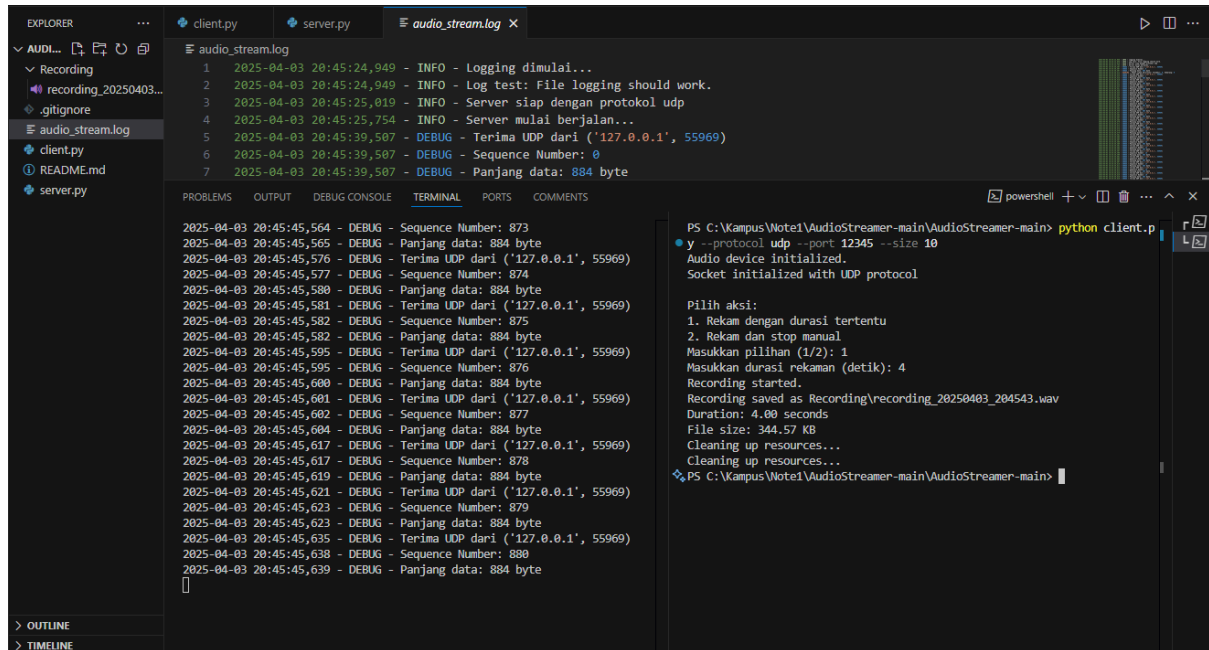
1. Inisialisasi Rekaman:
  - Membuat folder "Recording" saat inisialisasi
  - Menyiapkan format (16-bit PCM, mono, 44.1kHz)
  - Menyiapkan variabel `recorded_frames` untuk menyimpan data audio
2. Pengontrol Rekaman:
  - `start_recording()`: Memulai penyimpanan frame audio
  - `stop_recording()`: Menghentikan rekaman dan menyimpan file
  - `_stop_after_duration()`: Thread helper untuk rekaman otomatis
3. Callback Perekaman:
  - Menambahkan data audio ke `recorded_frames` saat `is_recording` aktif
  - Berjalan secara paralel dengan streaming audio
4. Penyimpanan File:
  - Menggunakan modul `wave` untuk membuat file WAV
  - Nama file otomatis dengan timestamp jika tidak ditentukan
  - Menyimpan metadata seperti channels, sample width, dan frame rate
  - Menampilkan informasi rekaman: durasi dan ukuran file
5. Implementasi Multi-threading:
  - Menggunakan thread terpisah untuk rekaman dengan durasi
  - Thread daemon untuk mencegah program menunggu thread rekaman

Fungsi utama ini memungkinkan pengguna untuk:

- Merekam audio sambil streaming ke server
- Merekam dengan durasi tetap atau manual
- Menyimpan file dengan penamaan otomatis
- Mendapatkan informasi tentang rekaman yang disimpan

## Uji Coba & Hasil

### Hasil pengujian UDP



```
EXPLORER client.py server.py audio_stream.log
└─ Recording
  └─ recording_20250403...
    └─ .gitignore
      └─ audio_stream.log
        └─ client.py
          └─ README.md
            └─ server.py

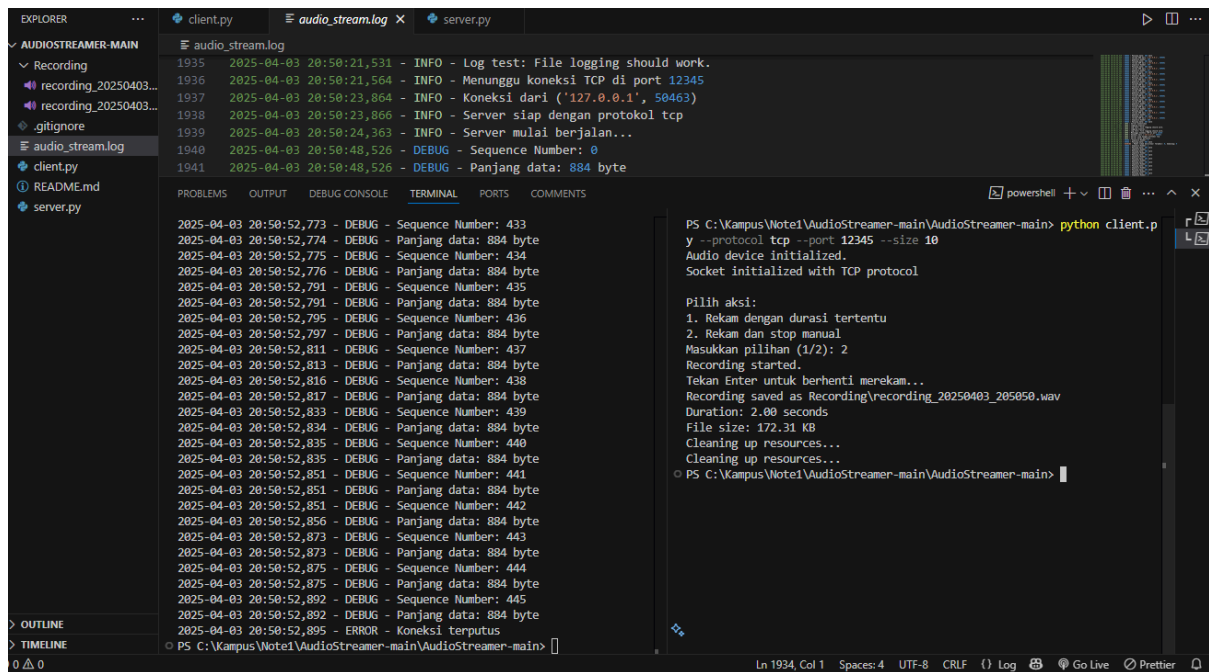
audio_stream.log
1 2025-04-03 20:45:24,949 - INFO - Logging dimulai...
2 2025-04-03 20:45:24,949 - INFO - Log test: File logging should work.
3 2025-04-03 20:45:25,019 - INFO - Server siap dengan protokol udp
4 2025-04-03 20:45:25,754 - INFO - Server mulai berjalan...
5 2025-04-03 20:45:39,507 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
6 2025-04-03 20:45:39,507 - DEBUG - Sequence Number: 0
7 2025-04-03 20:45:39,507 - DEBUG - Panjang data: 884 byte

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
2025-04-03 20:45:45,564 - DEBUG - Sequence Number: 873
2025-04-03 20:45:45,565 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,576 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,577 - DEBUG - Sequence Number: 874
2025-04-03 20:45:45,580 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,581 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,582 - DEBUG - Sequence Number: 875
2025-04-03 20:45:45,582 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,595 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,595 - DEBUG - Sequence Number: 876
2025-04-03 20:45:45,600 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,601 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,602 - DEBUG - Sequence Number: 877
2025-04-03 20:45:45,604 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,617 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,617 - DEBUG - Sequence Number: 878
2025-04-03 20:45:45,619 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,621 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,623 - DEBUG - Sequence Number: 879
2025-04-03 20:45:45,623 - DEBUG - Panjang data: 884 byte
2025-04-03 20:45:45,635 - DEBUG - Terima UDP dari ('127.0.0.1', 55969)
2025-04-03 20:45:45,638 - DEBUG - Sequence Number: 880
2025-04-03 20:45:45,639 - DEBUG - Panjang data: 884 byte

PS C:\Kampus\Vote1\AudioStreamer-main\AudioStreamer-main> python client.p
y --protocol udp --port 12345 --size 10
Audio device initialized.
Socket initialized with UDP protocol

Pilih aksi:
1. Rekam dengan durasi tertentu
2. Rekam dan stop manual
Masukkan pilihan (1/2): 1
Masukkan durasi rekaman (detik): 4
Recording started.
Recording saved as Recording\recording_20250403_204543.wav
Duration: 4.00 seconds
File size: 344.57 KB
Cleaning up resources...
Cleaning up resources...
```

### Hasil Pengujian TCP



```
EXPLORER client.py server.py audio_stream.log
└─ AUDIOSTREAMER-MAIN
  └─ Recording
    └─ recording_20250403...
      └─ .gitignore
        └─ audio_stream.log
          └─ client.py
            └─ README.md
              └─ server.py

audio_stream.log
1935 2025-04-03 20:50:21,531 - INFO - Log test: File logging should work.
1936 2025-04-03 20:50:21,564 - INFO - Menunggu koneksi TCP di port 12345
1937 2025-04-03 20:50:23,864 - INFO - Koneksi dari ('127.0.0.1', 50463)
1938 2025-04-03 20:50:23,866 - INFO - Server siap dengan protokol tcp
1939 2025-04-03 20:50:24,363 - INFO - Server mulai berjalan...
1940 2025-04-03 20:50:48,526 - DEBUG - Sequence Number: 0
1941 2025-04-03 20:50:48,526 - DEBUG - Panjang data: 884 byte

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
2025-04-03 20:50:52,773 - DEBUG - Sequence Number: 433
2025-04-03 20:50:52,774 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,775 - DEBUG - Sequence Number: 434
2025-04-03 20:50:52,776 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,791 - DEBUG - Sequence Number: 435
2025-04-03 20:50:52,791 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,795 - DEBUG - Sequence Number: 436
2025-04-03 20:50:52,797 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,811 - DEBUG - Sequence Number: 437
2025-04-03 20:50:52,813 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,816 - DEBUG - Sequence Number: 438
2025-04-03 20:50:52,817 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,833 - DEBUG - Sequence Number: 439
2025-04-03 20:50:52,834 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,835 - DEBUG - Sequence Number: 440
2025-04-03 20:50:52,835 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,851 - DEBUG - Sequence Number: 441
2025-04-03 20:50:52,851 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,851 - DEBUG - Sequence Number: 442
2025-04-03 20:50:52,856 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,873 - DEBUG - Sequence Number: 443
2025-04-03 20:50:52,873 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,875 - DEBUG - Sequence Number: 444
2025-04-03 20:50:52,875 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,892 - DEBUG - Sequence Number: 445
2025-04-03 20:50:52,892 - DEBUG - Panjang data: 884 byte
2025-04-03 20:50:52,895 - ERROR - Koneksi terputus

PS C:\Kampus\Vote1\AudioStreamer-main\AudioStreamer-main> python client.p
y --protocol tcp --port 12345 --size 10
Audio device initialized.
Socket initialized with TCP protocol

Pilih aksi:
1. Rekam dengan durasi tertentu
2. Rekam dan stop manual
Masukkan pilihan (1/2): 2
Recording started.
Tekan Enter untuk berhenti merekam...
Recording saved as Recording\recording_20250403_205050.wav
Duration: 2.00 seconds
File size: 172.31 KB
Cleaning up resources...
Cleaning up resources...
```

## **Kesimpulan**

Berdasarkan tugas yang diberikan dapat disimpulkan:

1. Adaptasi Arsitektur Client-Server : Sistem berhasil mengadaptasi arsitektur client-server dari proyek referensi dengan pemisahan fungsi yang jelas antara pengambilan audio (client) dan pemutaran audio (server).
2. Modifikasi dan Pengembangan Fungsional : Tiga modifikasi utama telah berhasil diimplementasikan untuk meningkatkan kualitas sistem:
  - Error handling komprehensif yang mencegah crash sistem dan memberikan informasi diagnostik
  - Sistem logging pada server dengan rotasi file yang memfasilitasi debugging dan analisis kinerja jangka panjang
  - Fitur rekaman pada client yang memungkinkan penyimpanan komunikasi audio secara lokal
3. Dukungan Multi-protokol : Fleksibilitas dalam pemilihan protokol UDP dan TCP telah diimplementasikan, dengan pengujian untuk memverifikasi kinerja pada kedua protokol tersebut.
4. Dokumentasi dan Pengujian : Sistem telah didokumentasikan dengan baik dalam laporan dan telah diuji untuk memastikan fungsionalitasnya berjalan sesuai harapan.