A black and white logo

AI-generated content may be incorrect.

**DEPARTMENT OF FORENSIC SCIENCE**

**COMPUTER ARCHITECTURE**

**ACTIVITY - 2**

QUANTUM VIDEO-TO-AUDIO CONVERTER USING QUANTUM ALGORITHMS AND QUBITS

**SUBMITTED BY,**

NAME: AKANKSHA K J

USN : 24MSRDF039

CLASS : 3 MSC DFIS

**SUBMITTED TO,**

Mr. VISHNU VENKATESH,

ASSISTANT PROFESSOR, DEPARTMENT OF FORENSIC SCIENCE,

JAIN (DEEMED-TO-BE) UNIVERSITY,

SCHOOL OF SCIENCES, JC ROAD

**Quantum Video-to-Audio Converter Using Quantum Algorithms and Qubits**

**Introduction**

In the online era, media conversion applications are one of the most common and eﬀective means to handle multimedia data, video/audio files. Most tools are classi­cal video-to-audio converters that utilize classi­cal computation to extract frames and audio streams with processing done deterministically. In our project, we deliver a Quantum-Enhanced Video-to-Audio Converter utilizing quantum algorithms and qubits to impact audio generation.

Our goal is to demonstrate how concepts of quantum randomness and superposition can be injected into classical-to-classical video/audio applications. In the project, we leverage Flask (Python) as our web interface and MoviePy for video/audio processing, and Qiskit for the quantum algorithms. Our target audience consists of students, developers, or researchers interested in hybrid quantum-classical applications.

**1. Project Overview**

**Objective**

To design and implement a simple web-based application that:

* Accepts user-uploaded videos (MP4, MOV, AVI, MKV).
* Converts them into MP3 audio files.
* Uses **quantum circuits and qubits** to modulate or enhance the audio properties.
* Demonstrates integration between **quantum computing** and **classical signal processing**.

**2. System Design**

**Workflow**

1. User uploads a video file through the web interface.
2. Flask server stores the video in the *uploads/* directory.
3. MoviePy extracts raw audio from the video file.
4. The audio is split into small chunks.
5. A **QuantumCircuit** generates qubit measurements (0 or 1) for each chunk.
6. Based on each measurement:
   * 0 → leave chunk unchanged.
   * 1 → slightly modify the audio (e.g., +3dB amplification).
7. The modified audio chunks are merged back into a final MP3 file.
8. A **download link** is displayed for the user to retrieve the quantum-modulated audio.

**3. Quantum Algorithm Integration**

The converter employs a superposition single-qubit circuit where the measurement of each qubit determines whether to modify an audio segment or leave it unchanged. The Hadamard gate adds randomness to the process.

**Effect on Audio**

The measured qubits introduce subtle random variations in:

* Volume (amplitude)
* Sampling segment selection
* Modulation effects

This creates a **quantum signature** in the resulting audio output.

**4. Implementation Details**

**Flask Web Application**

* HTML frontend for file upload and conversion.
* Backend handles video processing and quantum circuit calls.
* Displays a dynamic download link after conversion.

**Key Files**

* **Python app.py:** Core application code integrating Flask, Qiskit, and MoviePy.
* **index.html:** Frontend interface.
* **uploads/**: Storage folder for input and output files (demo video 1).

**Implementation Code**

1. **Python app.py**from flask import Flask, request, render\_template, send\_from\_directory, url\_for

from moviepy.editor import VideoFileClip

from qiskit import QuantumCircuit, transpile

from qiskit\_aer import AerSimulator

import os

from pydub import AudioSegment

app = Flask(\_\_name\_\_)

UPLOAD\_FOLDER = 'uploads'

os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)

# ---------------- Quantum Function ----------------

def measure\_qubit():

"""Measure a single qubit (0 or 1)"""

qc = QuantumCircuit(1, 1)

qc.h(0)

qc.measure(0, 0)

backend = AerSimulator()

compiled = transpile(qc, backend)

job = backend.run(compiled, shots=1)

result = job.result().get\_counts()

return int(list(result.keys())[0])

# ---------------- Flask Routes ----------------

@app.route('/', methods=['GET', 'POST'])

def upload\_file():

download\_filename = None

if request.method == 'POST':

if 'video' not in request.files:

return "No file part."

video = request.files['video']

if video.filename == '':

return "No file selected."

# Save uploaded video

video\_path = os.path.join(UPLOAD\_FOLDER, video.filename)

video.save(video\_path)

# Extract audio using MoviePy

clip = VideoFileClip(video\_path)

temp\_audio = os.path.splitext(video\_path)[0] + "\_temp.wav"

clip.audio.write\_audiofile(temp\_audio)

# Quantum-enhanced audio processing

audio = AudioSegment.from\_wav(temp\_audio)

chunk\_length = 100 # ms

chunks = [audio[i:i+chunk\_length] for i in range(0, len(audio), chunk\_length)]

quantum\_chunks = []

for chunk in chunks:

q = measure\_qubit()

if q == 1:

chunk = chunk + 3 # amplify 3 dB

quantum\_chunks.append(chunk)

final\_audio = sum(quantum\_chunks)

output\_filename = os.path.splitext(video.filename)[0] + "\_quantum\_audio.mp3"

output\_path = os.path.join(UPLOAD\_FOLDER, output\_filename)

final\_audio.export(output\_path, format="mp3")

download\_filename = output\_filename

return render\_template("index.html", download\_link=download\_filename)

@app.route('/download/<filename>')

def download\_file(filename):

return send\_from\_directory(UPLOAD\_FOLDER, filename, as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

1. **Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Quantum Video ➜ Audio Converter</title>

<style>

body {font-family:'Segoe UI', sans-serif; background:linear-gradient(135deg,#000428,#004e92); color:white; text-align:center; padding:40px;}

.container {background: rgba(255,255,255,0.1); border-radius:20px; box-shadow:0 0 20px rgba(255,255,255,0.2); padding:40px; width:500px; margin:auto;}

h1 {text-shadow:0 0 15px #00ffff;}

input[type=file] {background:#fff;color:#000;padding:10px;border-radius:10px;width:90%;margin-bottom:10px;}

button {background:#00c6ff;border:none;color:white;padding:10px 20px;margin-top:10px;border-radius:10px;cursor:pointer;font-size:16px;transition:0.3s;}

button:hover {background:#0072ff;}

a {color:#00ffcc;text-decoration:none;}

a:hover {text-decoration:underline;}

h3 {color:#00ffcc;}

</style>

</head>

<body>

<div class="container">

<h1>🎥 Quantum Video ➜ Audio Converter 🔊</h1>

<form method="POST" enctype="multipart/form-data">

<input type="file" name="video" accept="video/\*" required><br>

<button type="submit">Convert</button>

</form>

{% if download\_link %}

<h3>✅ Conversion Successful!</h3>

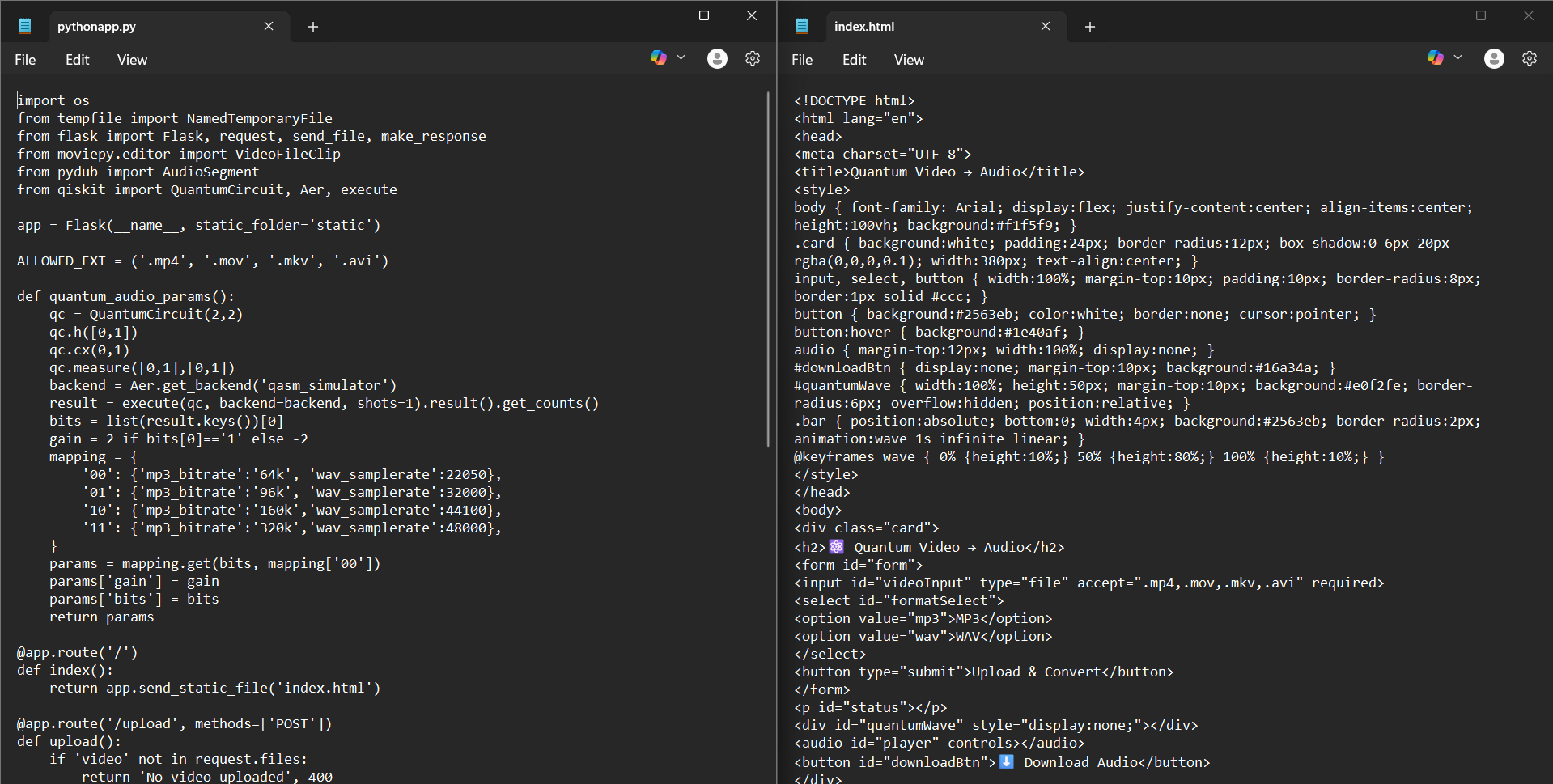
<p><a href="{{ url\_for('download\_file', filename=download\_link) }}">⬇️ Download Your Audio File</a></p>

{% endif %}

</div>

</body>

</html>



**Commands to Run**

cd "D:\Quantum Video to audio conversion"

.\venv312\Scripts\activate

pip install flask moviepy==1.0.3 qiskit qiskit-aer pydub numpy

python app.py

Then open <http://127.0.0.1:5000>.

**5. Tools Used**

* **Flask:** For web framework and routing.
* **MoviePy:** To extract and handle video/audio streams.
* **PyDub:** For modifying and merging audio segments.
* **Qiskit:** For implementing quantum circuits and simulating qubits.
* **AerSimulator:** For running quantum simulations locally.
* **FFmpeg:** For decoding and exporting media files.

**6. Results**

The application can effectively convert videos (MP4, MOV, MKV, AVI) into audio files in the MP3 format. The audio is altered based on the outcomes (results) of quantum qubits (qubits). So, every time audio is generated (or exported) it does produce slightly different audio output demonstrating quantum randomness.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**7. Conclusion**

The Quantum Video-to-Audio Converter showcases the idea of a hybrid application, incorporating quantum and classical systems. This project connects quantum algorithms (Hadamard and measurement) into the classical multimedia conversion pipeline, connecting the gap between emerging quantum computing and classical media processing. The project shows how small-scale quantum induced unpredictability, singularity and innovation can be added to everyday applications.

**8. References**

1. Qiskit Documentation — https://qiskit.org/documentation/
2. MoviePy Documentation — https://zulko.github.io/moviepy/
3. PyDub Documentation — https://pydub.com/
4. FFmpeg Builds — https://www.gyan.dev/ffmpeg/builds/
5. Flask Framework —https://flask.palletsprojects.com/

**9. Github link**

[**https://github.com/AkankshaKJ09/Quantum-Video-to-audio-conversion**](https://github.com/AkankshaKJ09/Quantum-Video-to-audio-conversion)