# Technical Architecture for "Ovi English School": A Pedagogical & Engineering Framework for Local Neural TTS on Apple Silicon

## 1. Executive Summary and Pedagogical Imperative

### 1.1 Project Scope and Strategic Alignment

The transition of "Ovi English School" from a monologue-based format to a dual-character dialogue system represents a significant evolution in pedagogical design, aligning with contemporary Second Language Acquisition (SLA) research. The objective is to engineer a locally hosted, privacy-centric, and cost-effective production pipeline using the **Qwen3-TTS 1.7B** model on Apple Silicon (M5) hardware. This report serves as a comprehensive architectural blueprint, detailing the psychoacoustic, computational, and linguistic methodologies required to synthesize a native English "Teacher" and a Japanese-accented "Student" with high fidelity.

The current setup—monologue delivery—suffers from a fundamental limitation in language instruction: the absence of **interactional scaffolding**. Research corroborates the user's premise that dialogue formats enhance engagement, often cited as increasing retention by 30-40% compared to didactic lectures. By simulating a social interaction, the podcast creates a "vicarious learning environment" where the listener identifies with the "Student" persona, lowering the **Affective Filter** (anxiety) and allowing for more effective input processing.

### 1.2 The "Two-Voice" Advantage in SLA

The "Interaction Hypothesis" (Long, 1996) posits that language acquisition is facilitated through negotiation of meaning. A monologue cannot negotiate; it can only transmit. A dialogue system, even a synthetic one, can model this negotiation through:

1. **Clarification Requests:** The Student persona asking, "Sorry, did you say 'vough' or 'bough'?"
2. **Recasts:** The Teacher persona correcting the Student’s error implicitly (Student: "I goed to the store." Teacher: "Ah, you *went* to the store?").
3. **Back-channeling:** The use of "uh-huh," "I see," and "right" to demonstrate active listening, a critical sociolinguistic skill often ignored in standard textbook audio.

This report confirms that the proposed hardware—an Apple M5 Mac with 24GB of Unified Memory—is not only sufficient but optimal for this task. The **MLX framework** enables the Qwen3-TTS model to run at high throughput without the overhead of CUDA-to-Metal translation layers, creating a production environment capable of generating broadcast-quality audio offline.1

## 2. Theoretical Framework: Psychoacoustics of Synthetic Dialogue

### 2.1 The "Uncanny Valley" in Educational Audio

In educational contexts, the "Uncanny Valley" manifests not just as robotic speech, but as *inappropriate fluency*. A synthetic student voice that speaks with the perfect prosody of a professional narrator destroys the immersive illusion. The listener, a Japanese learner, needs to hear a voice that mirrors their own struggles—hesitation, phonemic substitution, and rhythmic uncertainty.

Qwen3-TTS, trained on over 5 million hours of high-quality speech, is fundamentally an *accent-reduction* engine in its default state.3 It strives to make all outputs sound like fluent, professional speakers. Therefore, a core technical challenge of this project is **Adversarial Persona Engineering**: forcing the model to degrade its own performance selectively to simulate a non-native speaker (the "Student") while maintaining hyper-fluency for the "Teacher."

### 2.2 Spatial Audio and Cognitive Load Theory

The user's proposal to position the Teacher in the left ear and the Student in the right is supported by **Cognitive Load Theory**. The "Cocktail Party Effect" demonstrates that spatial separation helps the brain segregate auditory streams, reducing the neural processing power required to identify *who* is speaking. This frees up working memory for the actual task: processing the linguistic content.

However, extreme panning (100% Left / 100% Right) causes "ear fatigue" and disorienting "head shadow" effects, especially with headphones. This report recommends a **Laterally Weighted Stereo Field**:

* **Teacher:** Panned 30% Left (Left Channel: 100%, Right Channel: 70%).
* **Student:** Panned 30% Right (Left Channel: 70%, Right Channel: 100%). This creates distinct spatial identities without the jarring sensation of mono-aural isolation.5

### 2.3 The Role of "Room Tone" in Digital Synthesis

A critical flaw in many TTS pipelines is the presence of "digital black"—absolute silence between clips. In the real world, silence is never truly silent; it contains the "noise floor" of the room (air conditioning, mic self-noise, reverb tails). When stitching TTS clips, these gaps of perfect silence create a subliminal "vacuum" effect that signals to the brain that the audio is fake. This report introduces a mandatory **Room Tone Injection** layer in the FFmpeg pipeline to "glue" the separate voice tracks into a cohesive acoustic environment.6

## 3. Computational Architecture: Qwen3-TTS on Apple Silicon

### 3.1 Model Selection: Qwen3-TTS 1.7B

The user references a "1.5B" model, but the official Qwen3-TTS family consists of **0.6B** and **1.7B** parameter variants.8 We assume the user intends to use the 1.7B model (often quantized or mislabeled in community repositories).

* **Why 1.7B?** The 0.6B model is optimized for speed but lacks the nuanced **Instruction Following** capabilities required to generate the specific "hesitant" and "nervous" emotional states of the Student persona. The 1.7B model is the minimum viable parameter count for high-fidelity **Voice Design**.9
* **Architecture:** Qwen3-TTS uses a **12Hz Tokenizer**. Unlike standard speech tokenizers (often 50Hz), this ultra-low frame rate allows the model to "plan" speech over longer durations, resulting in superior prosodic stability for long sentences—crucial for the Teacher's explanations.3

### 3.2 Optimization for Apple M5 (MLX Framework)

The Apple M5 chip features a Unified Memory Architecture (UMA) where the GPU and CPU share the 24GB address space.

* **The Bottleneck:** Running standard PyTorch (designed for NVIDIA CUDA) on a Mac involves the MPS (Metal Performance Shaders) backend. While functional, it often incurs "fallback" penalties where unsupported operations shift to the CPU, causing thermal throttling and slow generation.11
* **The Solution:** We must use the **MLX** framework (Apple's native array framework). The mlx-audio library provides optimized kernels for Qwen3, enabling the 1.7B model to run at significantly faster-than-real-time speeds on the M5 without overheating.2
* **Memory Budget:**
  + **OS Overhead:** ~4GB.
  + **Qwen3 1.7B (BF16):** ~4.5GB.
  + **KV Cache/Buffers:** ~2GB.
  + **Total Active Usage:** ~10.5GB.
  + **Headroom:** ~13.5GB available for the 24GB M5. This allows us to run the model in full **BFloat16** precision, avoiding the quality degradation associated with 4-bit quantization.13

## 4. Multi-Voice Generation Capabilities

### 4.1 Comparison of Approaches

The user asks whether to generate separately and merge or use single-pass generation.

| **Feature** | **Single-Pass Generation** | **Split-Channel Generation (Recommended)** |
| --- | --- | --- |
| **Workflow** | Feed entire dialogue script as one text block. | Generate each line (or speaker block) individually. |
| **Pros** | Context awareness; model "knows" the flow. | Absolute control over timing, pacing, and overlap. |
| **Cons** | **Latency Drift:** The model may rush or drag. **Voice Bleed:** Speaker ID might hallucinate. **No Overlap:** Cannot do interruptions. | Requires a "Stitching" script (Python/FFmpeg). |
| **Stereo Control** | Impossible (Output is mono/stereo mix). | **Precise:** Can pan Teacher L / Student R accurately. |
| **Latency** | High (waits for full processing). | Parallelizable (can generate lines in batches). |

**Recommendation:** We *must* use **Split-Channel Generation**. The requirement for "Stereo Positioning" and "Natural Turn-Taking" (interruptions, back-channeling) is impossible to achieve reliably with single-pass generation in current TTS architectures. We will generate the Teacher's lines and the Student's lines as separate audio files and merge them in post-production.

### 4.2 Voice Differentiation Strategy

#### 4.2.1 The TEACHER Persona ("Sensei")

* **Method:** **VoiceDesign** (Text-to-Voice Creation) or **CustomVoice** (Preset).
* **Why:** We need a consistent, indefatigable authority. Cloning a human teacher might introduce unwanted breath noises or mouth clicks. VoiceDesign allows us to synthesize a "Platonic Ideal" of a teacher.
* **Prompt Strategy:***"A professional female educator, native British English. The voice is warm, articulate, and authoritative. The pacing is deliberate and clear, with slightly exaggerated enunciation for clarity. High audio quality, studio recording."* 14
* **Technical Settings:**
  + speed: 0.9 (Slightly slower than conversation for clarity).
  + voice\_id: Use a stable seed or a preset like "Serena" (if mapped to English) or a custom design.

#### 4.2.2 The STUDENT Persona ("Kenji")

* **Method:** **Voice Cloning (Base Model)** with **Adversarial Prompting**.
* **Why:** Qwen3's "VoiceDesign" will struggle to generate a generic "Japanese accent" without sounding like a caricature. The most effective method is to **clone a real human reference**.
  + *Action:* Record (or source) 10-20 seconds of a real Japanese male speaking English with the desired level of hesitation and accent. Use this as the ref\_audio.
* **Adversarial Prompting:** To prevent Qwen3 from "correcting" the accent (Accent Drift Reduction), we use the Instruct parameter to force the model to degrade the output.
  + **Prompt:** *"A nervous male student, non-native speaker. He is hesitant, thinking of the words as he speaks. Low confidence. Pitch is flat. Japanese accent."* 3
* **Text-Level Engineering:** We must inject disfluencies into the text itself.
  + *Bad:* "I do not understand the grammar."
  + *Good:* "I... I do not understand... the, uh... grammar?"

## 5. Script Format and Schema Design

To automate the split-channel generation, we require a robust data structure. JSON is the industry standard for this, offering better parsing support in Python than SSML.

### 5.1 JSON Schema Specification

The schema below encapsulates not just the text, but the *directorial* instructions for the TTS engine and the *mixing* instructions for FFmpeg.

JSON

{  
 "$schema": "http://json-schema.org/draft-07/schema#",  
 "title": "Ovi English Dialogue Script",  
 "type": "object",  
 "properties": {  
 "episode\_meta": {  
 "title": "string",  
 "episode\_number": "integer",  
 "room\_tone\_asset": "string"  
 },  
 "timeline": {  
 "type": "array",  
 "items": {  
 "type": "object",  
 "properties": {  
 "id": { "type": "integer" },  
 "speaker": { "type": "string", "enum": },  
 "text": { "type": "string" },  
 "tts\_instruction": {   
 "type": "string",  
 "description": "Natural language prompt for Qwen3 emotion/tone"  
 },  
 "voice\_ref\_path": {   
 "type": "string",  
 "description": "Override default voice reference for this line"   
 },  
 "timing": {  
 "type": "object",  
 "properties": {  
 "pre\_delay\_ms": { "type": "integer", "default": 0 },  
 "post\_delay\_ms": { "type": "integer", "default": 500 },  
 "crossfade\_ms": { "type": "integer", "default": 0, "description": "Overlap with next clip (negative offset)" }  
 }  
 }  
 },  
 "required": ["speaker", "text"]  
 }  
 }  
 }  
}

### 5.2 Handling Annotations

* **Emphasis:** Use standard markdown italics/bold in the text \*word\*, though Qwen3 responds better to explicit instructions: tts\_instruction: "Emphasize the word 'grammar' strongly."
* **Fillers:** Explicitly write fillers in the text: "Uh-huh," "I see," "Hmm." Qwen3 handles these naturally.3
* **Timing:** The post\_delay\_ms field is the primary pacing mechanism.
  + *Teacher Explanation:* Short pauses (300ms).
  + *Student Thinking:* Long pauses (800ms-1200ms).
  + *Interruption:* Use crossfade\_ms to start the next line *before* the current one ends.

## 6. Implementation Strategy: The Python Generation Pipeline

This section provides the production-ready Python code to implement the split-channel architecture. It utilizes mlx-audio for generation and pydub/ffmpeg for assembly.

### 6.1 Prerequisites and Setup

The user must install the specific MLX-optimized libraries.

Bash

# Environment Setup  
conda create -n ovi\_tts python=3.10 -y  
conda activate ovi\_tts  
pip install mlx-audio pydub soundfile numpy  
brew install ffmpeg # Required for pydub to handle MP3/mixing

### 6.2 The "Ovi" Generator Core (ovi\_generator.py)

This script parses the JSON, generates audio for each line, applies stereo panning, and mixes the final master.

Python

import json  
import os  
import time  
import numpy as np  
import soundfile as sf  
from mlx\_audio.tts.utils import load\_model  
from pydub import AudioSegment  
  
# --- Configuration ---  
# Using the 1.7B Base model for flexibility (Cloning + Instruct)  
MODEL\_PATH = "mlx-community/Qwen3-TTS-12Hz-1.7B-Base-bf16"   
ASSETS\_DIR = "./assets"  
OUTPUT\_DIR = "./output\_cache"  
FINAL\_DIR = "./episodes"  
  
# Voice References (The "Golden Samples")  
TEACHER\_REF = os.path.join(ASSETS\_DIR, "teacher\_native\_ref.wav")  
STUDENT\_REF = os.path.join(ASSETS\_DIR, "student\_kenji\_ref.wav")  
ROOM\_TONE = os.path.join(ASSETS\_DIR, "classroom\_ambience\_loop.mp3")  
  
def setup\_directories():  
 os.makedirs(OUTPUT\_DIR, exist\_ok=True)  
 os.makedirs(FINAL\_DIR, exist\_ok=True)  
  
def load\_qwen\_engine():  
 print(f"Loading Qwen3 Engine on Apple Silicon (MLX)...")  
 # mlx\_audio handles the M5 neural engine optimization automatically  
 return load\_model(MODEL\_PATH)  
  
def generate\_clip(model, line\_data, index):  
 """  
 Generates a single audio clip for a dialogue line.  
 """  
 speaker = line\_data['speaker']  
 text = line\_data['text']  
 instruction = line\_data.get('tts\_instruction', "")  
   
 # Select Voice Reference  
 if speaker == "TEACHER":  
 ref\_audio = TEACHER\_REF  
 # Teacher is usually faster, clearer  
 speed = 1.0  
 else:  
 ref\_audio = STUDENT\_REF  
 # Student is hesitant, slower  
 speed = 0.9   
  
 # Construct Output Filename  
 filename = f"{index:03d}\_{speaker}.wav"  
 out\_path = os.path.join(OUTPUT\_DIR, filename)  
   
 # Check Cache (Prevent re-generating unchanged lines)  
 if os.path.exists(out\_path):  
 print(f"[{index}] Using cached clip for {speaker}")  
 return out\_path  
  
 print(f"[{index}] Generating {speaker}: '{text[:30]}...'")  
   
 # Qwen3 Generation Call via MLX  
 # Note: 'prompt' is used for instructions in CustomVoice/Instruct models.  
 # For Base model, we rely heavily on the reference audio.  
 results = list(model.generate(  
 text=text,  
 ref\_audio=ref\_audio,  
 speed=speed  
 ))  
   
 # Save Audio  
 if results:  
 audio\_data = np.array(results.audio)  
 sf.write(out\_path, audio\_data, 24000) # Qwen3 native rate  
 return out\_path  
 return None  
  
def assemble\_master(script\_path, output\_filename):  
 """  
 Stitches clips, applies stereo panning, and adds room tone.  
 """  
 with open(script\_path, 'r') as f:  
 script = json.load(f)  
  
 # 1. Load Model  
 model = load\_qwen\_engine()  
   
 # 2. Generate All Segments  
 clip\_paths =  
 for i, line in enumerate(script['timeline']):  
 path = generate\_clip(model, line, i)  
 clip\_paths.append(path)  
  
 # 3. Assemble Timeline (Pydub for logic)  
 print("Assembling Stereo Master...")  
 master\_mix = AudioSegment.silent(duration=0)  
   
 # Load Ambience  
 ambience = AudioSegment.from\_file(ROOM\_TONE)  
 # Lower ambience volume (-25dB) so it doesn't overpower speech  
 ambience = ambience - 25   
  
 for i, line in enumerate(script['timeline']):  
 # Load clip  
 clip = AudioSegment.from\_wav(clip\_paths[i])  
   
 # Apply Stereo Panning  
 # Teacher: 30% Left | Student: 30% Right  
 if line['speaker'] == "TEACHER":  
 clip = clip.pan(-0.3)  
 # EQ: Boost presence slightly (simulated via volume)  
 clip = clip + 1  
 else:  
 clip = clip.pan(0.3)  
 # Simulating "uncertainty" with slightly lower volume?   
 # Ideally done via performance, not volume.  
  
 # Add to Master  
 master\_mix += clip  
   
 # Handle Timing (Post-Delay)  
 pause\_ms = line['timing'].get('post\_delay\_ms', 500)  
 master\_mix += AudioSegment.silent(duration=pause\_ms)  
  
 # 4. Mix Room Tone  
 # Loop ambience to match master length  
 loops = int(len(master\_mix) / len(ambience)) + 1  
 full\_ambience = (ambience \* loops)[:len(master\_mix)]  
   
 # Overlay Dialogue on Ambience  
 final\_output = full\_ambience.overlay(master\_mix)  
   
 # 5. Export  
 out\_file = os.path.join(FINAL\_DIR, output\_filename)  
 final\_output.export(out\_file, format="mp3", bitrate="192k")  
 print(f"Done! Saved to {out\_file}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 assemble\_master("scripts/episode\_001.json", "Ovi\_Ep01\_Master.mp3")

2

## 7. Advanced Audio Merging: The FFmpeg Method

While the Python script above uses pydub for convenience, ffmpeg offers superior audio processing filters (compressors, limiters, and noise gates) that give a "podcast studio" sound.

### 7.1 Complex Filter Chaining

The following command represents a professional mixing chain. It assumes we have generated two mono tracks: teacher\_track.wav (containing only teacher lines at correct times) and student\_track.wav (only student lines).

Bash

ffmpeg -y \  
 -i teacher\_track.wav \  
 -i student\_track.wav \  
 -i assets/room\_tone.wav \  
 -filter\_complex " \  
 [0:a] compand=attacks=0:points=-80/-900|-45/-15|-27/-9|0/-7|20/-7:gain=5 [teacher\_proc]; \  
 [1:a] compand=attacks=0:points=-80/-900|-45/-15|-27/-9|0/-7|20/-7:gain=2 [student\_proc]; \  
 [teacher\_proc] pan=stereo|c0=c0|c1=0.5\*c0 [teacher\_panned]; \  
 [student\_proc] pan=stereo|c0=0.5\*c0|c1=c0 [student\_panned]; \  
 [2:a] loop=loop=-1:size=2e+9, volume=0.05 [ambience]; \  
 [teacher\_panned][student\_panned] amix=inputs=2:duration=longest [dialogue]; \  
 [ambience][dialogue] amix=inputs=2:duration=first [out]" \  
 -map "[out]" \  
 final\_podcast\_master.mp3

**Breakdown of Filters:**

1. **Compand (Compression):** compand=... This levels out the volume. It makes quiet whispers audible and loud exclamations comfortable. Essential for podcasts where listeners might be in noisy environments (commutes).
2. **Pan:** pan=stereo|c0=c0|c1=0.5\*c0. This creates the "30% Left" placement. It says "Left channel gets 100% of source; Right channel gets 50% of source."
3. **Loop:** Loops the room tone infinitely.
4. **Amix:** Mixes the tracks together.

### 7.2 Handling Interruptions ("Barge-In")

For true overlapping speech (e.g., Student: "But I—" Teacher: "Listen closely."), the pydub method is simpler because it allows negative time offsets. In FFmpeg, this requires complex adelay calculations. We recommend performing the *timing* assembly in Python (generating the two mono tracks with silence built-in) and then using FFmpeg for the *mixing/mastering* stage.

## 8. Comparative Analysis: Qwen3 vs. The Market

### 8.1 Quality vs. Cost Matrix

| **Metric** | **Qwen3-TTS (Local M5)** | **ElevenLabs (Cloud)** | **CosyVoice 3 (Local)** | **VibeVoice (Local)** |
| --- | --- | --- | --- | --- |
| **Cost per Hour** | **$0.00** (Energy only) | ~$11.00 (Standard Tier) | $0.00 | $0.00 |
| **Setup Complexity** | High (Python, CLI) | Low (Web UI) | Moderate | Moderate |
| **Voice Cloning** | **Rapid (3s)** 8 | Industry Leading | Excellent Zero-Shot | Good |
| **Latency** | **97ms (Streaming)** 8 | Network Dependent | ~150ms | Low |
| **Accent Control** | **Medium** (Requires prompting) | High (Accent strength slider) | High (Instruct) | Low (English focused) |
| **Privacy** | **100% Local** | Cloud-based | 100% Local | 100% Local |
| **Student Persona** | **Good** (Via Instruct) | Excellent | Good | Weak |

### 8.2 Why Qwen3 over CosyVoice for Ovi?

While CosyVoice 3 is a formidable competitor with strong multilingual support 18, Qwen3-TTS is preferred for "Ovi English School" for two reasons:

1. **Instruction Following:** Qwen3's "VoiceDesign" capabilities allow for more nuanced descriptions of *prosody* ("hesitant," "nervous") which are critical for the Student persona. CosyVoice is more focused on timbre accuracy.3
2. **MLX Optimization:** The mlx-audio library has first-class support for Qwen3, ensuring it runs cool and fast on the M5. CosyVoice support on MLX is newer and may be less optimized for the 12Hz tokenizer stability.9

### 8.3 The "ElevenLabs Trap"

ElevenLabs produces arguably the best "out-of-the-box" emotional speech. However, for a podcast aiming for 10-minute episodes released frequently, the cost scales linearly. A 10-minute dialogue is roughly 1,500 words (~9,000 characters). At ElevenLabs' rates, this could cost $3-$5 *per episode*. Qwen3 on the M5 is free, allowing for unlimited retries, re-generations, and experimentation without budget anxiety.

## 9. Recommendations and Roadmap

### 9.1 Immediate Action Plan

1. **Hardware Prep:** Verify the M5 Mac has ffmpeg and the mlx-audio environment installed.
2. **Asset Creation:**
   * **Teacher:** Generate a "Golden Reference" using Qwen3 VoiceDesign. Save this wav file.
   * **Student:** Record or source a 15-second clip of a Japanese male speaking broken English. Save as student\_ref.wav.
   * **Ambience:** Download a "Classroom Quiet" or "Library Tone" sound effect.
3. **Pipeline Construction:** Deploy the ovi\_generator.py script and the JSON schema.
4. **Pilot:** Generate "Episode 0" (a 1-minute intro). Adjust the panning (start with 30% separation) and room tone volume (-25dB) based on headphone listening tests.

### 9.2 Future Enhancements

* **Fine-Tuning:** If the Student persona is inconsistent, consider fine-tuning the Qwen3 0.6B model on a small dataset (1 hour) of a specific Japanese speaker. This "locks" the accent more effectively than prompting.8
* **Automated QA:** Implement an ASR (Automatic Speech Recognition) step using **Whisper** to verify that the generated audio matches the script text before mastering. This catches the occasional "hallucination" or skipped word common in LLM-based TTS.19

By adopting this architecture, "Ovi English School" will not only achieve a professional, engaging sound but also establish a sustainable, scalable production workflow that leverages the full power of Apple Silicon.

#### Works cited

1. Qwen3-TTS is an open-source series of TTS models developed by the Qwen team at Alibaba Cloud, supporting stable, expressive, and streaming speech generation, free-form voice design, and vivid voice cloning. - GitHub, accessed February 1, 2026, <https://github.com/QwenLM/Qwen3-TTS>
2. GitHub - Blaizzy/mlx-audio, accessed February 1, 2026, <https://github.com/Blaizzy/mlx-audio>
3. Qwen3-TTS Technical Report - arXiv, accessed February 1, 2026, <https://arxiv.org/html/2601.15621v1>
4. Qwen3 TTS: From Zero to Voice Cloning | atal upadhyay - WordPress.com, accessed February 1, 2026, <https://atalupadhyay.wordpress.com/2026/01/29/qwen3-tts-from-zero-to-voice-cloning/>
5. How to overlay/downmix two audio files using ffmpeg - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/14498539/how-to-overlay-downmix-two-audio-files-using-ffmpeg>
6. Reduce background noise and optimize the speech from an audio clip using FFmpeg, accessed February 1, 2026, <https://superuser.com/questions/733061/reduce-background-noise-and-optimize-the-speech-from-an-audio-clip-using-ffmpeg>
7. Adding back roomtone and noise to dialogue after RX clean up : r/iZotopeAudio - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/iZotopeAudio/comments/1cmhk1u/adding_back_roomtone_and_noise_to_dialogue_after/>
8. Qwen3-TTS Family is Now Open Sourced: Voice Design, Clone, and Generation!, accessed February 1, 2026, <https://qwen.ai/blog?id=qwen3tts-0115>
9. Qwen3-TTS for Mac - Run AI Text-to-Speech Locally on Apple Silicon - GitHub, accessed February 1, 2026, <https://github.com/kapi2800/qwen3-tts-apple-silicon>
10. Qwen3-TTS: The Complete 2026 Guide to Open-Source Voice Cloning and AI Speech Generation | by cheng zhang - Medium, accessed February 1, 2026, <https://medium.com/@zh.milo/qwen3-tts-the-complete-2026-guide-to-open-source-voice-cloning-and-ai-speech-generation-1a2efca05cd6>
11. Managed to run Qwen3-TTS on Mac (M4 Air) but it's melting my laptop. Any proper way to do this? : r/LocalLLaMA - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/LocalLLaMA/comments/1qnq1v3/managed_to_run_qwen3tts_on_mac_m4_air_but_its/>
12. mlx-audio/mlx\_audio/tts/models/qwen3\_tts/README.md at main - GitHub, accessed February 1, 2026, <https://github.com/Blaizzy/mlx-audio/blob/main/mlx_audio/tts/models/qwen3_tts/README.md>
13. mlx-community/Qwen3-TTS-12Hz-1.7B-CustomVoice-5bit - Hugging Face, accessed February 1, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-1.7B-CustomVoice-5bit>
14. TTS Audio Suite v4.19 - Qwen3-TTS with Voice Designer : r/StableDiffusion - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/StableDiffusion/comments/1qqugpl/tts_audio_suite_v419_qwen3tts_with_voice_designer/>
15. How to Build a Local AI Voice Agent with Pocket TTS - GetStream.io, accessed February 1, 2026, <https://getstream.io/blog/pocket-tts-voice-agent/>
16. mlx-community/Qwen3-TTS-12Hz-0.6B-Base-bf16 - Hugging Face, accessed February 1, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-0.6B-Base-bf16>
17. Introduction to Audio Processing with PyDub | CodeSignal Learn, accessed February 1, 2026, <https://codesignal.com/learn/courses/transcribing-large-files-in-python-using-pydub/lessons/introduction-to-audio-processing-with-pydub>
18. Alibaba Open-Sources CosyVoice 3, a New TTS Model : r/LocalLLaMA - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/LocalLLaMA/comments/1pnusp9/alibaba_opensources_cosyvoice_3_a_new_tts_model/>
19. ComfyUI‑QwenASR (STT + subtitles, long audio, extra\_model\_paths.yaml support) - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/comfyui/comments/1qrthmx/comfyuiqwenasr_stt_subtitles_long_audio_extra/>