Technical Research Report: Offline Voice Processing Solutions for Raspberry Pi 5

Project: Athina Super Car Voice Assistant
Date: 2025-07-05
Audience: Embedded Systems Developers, AI Engineers
Objective: To conduct a comprehensive technical comparison of offline voice processing solutions for the Raspberry Pi 5, providing detailed analysis and recommendations for the selection of Text-to-Speech (TTS), Speech-to-Text (STT), and Wake Word detection components for the Athina project.

Executive Summary

This report presents a detailed technical analysis of offline voice processing technologies suitable for deployment on the Raspberry Pi 5, specifically for the development of the Athina super car voice assistant. The primary objective is to identify the optimal components for Wake Word Detection, Speech-to-Text (STT), and Text-to-Speech (TTS) that meet the stringent requirements of an in-vehicle environment: low latency, high accuracy, minimal resource consumption, and robust offline functionality.

The Raspberry Pi 5 serves as a powerful and cost-effective hardware platform, offering a 2-3x performance increase over its predecessor, with a 2.4GHz quad-core Arm Cortex-A76 CPU, enhanced memory bandwidth, and a PCIe interface for high-speed storage. These advancements make it a viable platform for running sophisticated AI models at the edge.

Our analysis evaluates leading open-source and commercial solutions in each category. For Wake Word Detection, we compare `openWakeWord` and `Picovoice Porcupine`. For Speech-to-Text, we analyze `Vosk` and `Whisper.cpp`. For Text-to-Speech, we examine `Piper TTS` and `Coqui TTS`. The evaluation criteria include performance benchmarks, CPU and memory usage, integration complexity, model quality, and licensing considerations.

Based on this comprehensive analysis, the following recommendations are made for the Athina project:

- **Wake Word Detection:** `openWakeWord` is recommended as the primary choice due to its open-source nature, strong accuracy claims, built-in noise suppression and VAD features, and a flexible, programmatic approach to training custom wake words. Its performance on Raspberry Pi-class hardware is well-documented and sufficient for the Athina system.
- **Speech-to-Text (STT):** `Whisper.cpp` is the recommended solution, specifically using the `tiny.en` or `base.en` models. It offers state-of-the-art accuracy and robustness, and performance tests indicate that these smaller models can achieve the real-time transcription necessary for a responsive voice assistant on the Raspberry Pi 5.
- **Text-to-Speech (TTS):** `Piper TTS` is the ideal choice for the Athina assistant. It is explicitly optimized for Raspberry Pi, delivering fast, low-latency, and high-quality speech synthesis entirely offline. The availability of high-quality, permissively licensed female voice models and its active development make it a reliable and superior option for this application.

This combination of technologies provides a powerful, private, and responsive voice interaction stack, leveraging the full capabilities of the Raspberry Pi 5 to deliver a premium user experience for the Athina super car voice assistant.

1. Introduction and Project Context

The development of the Athina super car voice assistant represents a strategic initiative to integrate a sophisticated, responsive, and private voice interface into a luxury automotive environment. Unlike cloud-dependent voice assistants, Athina must operate entirely offline, ensuring functionality regardless of network connectivity and guaranteeing the privacy of user interactions. The chosen hardware platform for this edge AI system is the Raspberry Pi 5, a single-board computer that offers a significant leap in computational power and I/O capabilities, making it suitable for real-time AI workloads.

This report provides a foundational technical analysis for the Athina development team. It evaluates and compares leading software solutions for the three core components of the voice processing pipeline: Wake Word Detection, Speech-to-Text (STT), and Text-to-Speech (TTS). The analysis is tailored to the specific constraints and opportunities presented by the Raspberry Pi 5, focusing on performance metrics such as latency and resource consumption, as well as qualitative factors like accuracy, ease of integration, and customization potential.

The Raspberry Pi 5's architecture, featuring a 2.4GHz quad-core 64-bit Arm Cortex-A76 processor, LP-DDR4X RAM, and a PCIe 2.0 interface, provides a robust foundation. This platform is 2-3 times faster than the Raspberry Pi 4, with significantly improved I/O that can leverage high-speed NVMe storage. This performance uplift is critical for minimizing latency in the voice interaction loop, from a user speaking the wake word to receiving an audible response from the system.

The selection of each component will be guided by its ability to contribute to a seamless and natural user experience. The wake word engine must be highly accurate and efficient, consuming minimal resources while continuously listening. The STT engine must rapidly and accurately transcribe user commands, even in the potentially noisy environment of a car cabin. The TTS engine must synthesize a clear, natural-sounding voice with minimal delay, embodying the premium quality of the Athina brand. This report will provide clear, data-driven recommendations to guide the selection of these critical technologies.

2. Wake Word Detection Engine Comparison

The wake word engine is the "always-on" gateway to the Athina voice assistant. It must be exceptionally lightweight, as it runs continuously, and highly accurate to prevent both false activations (triggering without the wake word) and false rejections (failing to trigger when the wake word is spoken). We evaluate two leading solutions, `openWakeWord` and `Picovoice Porcupine`, for their suitability on the Raspberry Pi 5.

2.1. Technical Overview and Architecture

`Picovoice Porcupine` is a commercial, highly optimized, and lightweight wake word engine designed for resource-constrained environments, including IoT devices and microcontrollers. It is built on deep neural networks trained in real-world conditions to ensure robustness against noise and across various accents. Porcupine's core philosophy is to provide extreme efficiency and cross-platform compatibility, with explicit support listed for the Raspberry Pi 5. A key feature is the Picovoice Console, a web-based tool that allows developers to create custom, production-ready wake word models in seconds by simply typing the desired phrase, eliminating the need for manual data collection. The engine's architecture is proprietary but is known to be compact, using approximately 1 MB of memory and less than 4% of a single CPU core on a Raspberry Pi 3, which suggests its resource footprint on the more powerful Raspberry Pi 5 would be negligible.

In contrast, `openWakeWord` is a fully open-source alternative under the Apache 2.0 license, offering transparency and flexibility. Its architecture is based on a three-part model: a melspectrogram pre-processing function, a shared feature extraction backbone derived from a pre-trained Google speech embedding model, and a simple classification model. This design allows for efficient operation, as adding new wake words only requires training a small new classification head, while the larger feature extractor remains shared. A significant innovation of `openWakeWord` is its training methodology, which relies entirely on 100% synthetic speech generated from TTS models. This democratizes the creation of custom wake words, making it possible to train a new model in under an hour using a provided Google Colab notebook, without any manual data gathering. It also integrates optional Speex noise suppression and Silero Voice Activity Detection (VAD) to enhance performance in noisy environments.

2.2. Performance and Accuracy on Raspberry Pi

Performance on the Raspberry Pi 5 is a critical differentiator. Porcupine's documented efficiency on the Raspberry Pi 3 is a strong indicator of its performance on the Pi 5. Consuming minimal CPU and memory, it would leave ample system resources for the more demanding STT and TTS tasks. Picovoice provides extensive benchmarks, claiming over 97% detection accuracy with fewer than one false alarm every 10 hours, even when tested with background speech and noise from the DEMAND dataset at a 10 dB signal-to-noise ratio. This level of performance is crucial for the in-car environment of Athina. However, one user report noted that Porcupine's accuracy can decline if a command is spoken immediately after the wake word, a scenario common in natural interaction that warrants specific testing for the Athina use case.

`openWakeWord` also demonstrates impressive efficiency. Its developers state that a single core of a Raspberry Pi 3 can run 15-20 `openWakeWord` models simultaneously in real-time. This implies that running a single wake word model for Athina on a Raspberry Pi 5 would be a trivial task. The project makes a bold claim regarding accuracy, stating that for specific test data, its models can be "more accurate than Porcupine." This is supported by its own evaluation, which shows favorable false-accept/false-reject curves against Porcupine for the "alexa" wake word. The `openWakeWord` evaluation methodology is robust, using the challenging Dinner Party Corpus to measure a target false-accept rate of less than 0.5 per hour and simulating realistic noise and reverberation for false-reject testing. The inclusion of VAD and noise suppression further bolsters its potential for high accuracy in the variable noise conditions of a car.

2.3. Integration, Customization, and Licensing

The integration and customization workflow presents a clear trade-off. Porcupine offers a polished, user-friendly experience through its web-based Console for creating custom wake words. This is ideal for rapid prototyping and for teams that prefer a managed, no-code solution. However, Porcupine is a commercial product. While it offers a free tier, commercial use in a product like the Athina super car would require a commercial license from Picovoice. Its proprietary nature means the underlying model architecture is a black box.

`openWakeWord`, being fully open-source, offers maximum flexibility and control. The process for training a custom wake word, while requiring some familiarity with Python and Google Colab, is well-documented and highly accessible due to its reliance on synthetic data. This allows the Athina team to create and iterate on custom wake words (e.g., "Hey Athina") without any licensing fees or reliance on a third-party service. The Apache 2.0 license is permissive and suitable for commercial use. This open approach allows for deeper integration and potential modification of the engine itself to perfectly suit the project's needs. The ability to programmatically control the entire pipeline, from data generation to model training, is a significant advantage for an advanced engineering team.

3. Speech-to-Text (STT) Engine Comparison

The Speech-to-Text (STT) or Automatic Speech Recognition (ASR) engine is responsible for converting the user's spoken commands into machine-readable text. For the Athina assistant, this component must be highly accurate to understand user intent correctly and fast enough to enable a fluid, conversational experience. We compare `Vosk`, a lightweight engine designed for edge devices, with `Whisper.cpp`, a high-performance C++ port of OpenAI's state-of-the-art Whisper model.

3.1. Technical Overview and Architecture

Vosk is an open-source offline speech recognition toolkit built upon the well-established Kaldi speech recognition framework. It is specifically designed for versatility and efficiency on edge devices, offering support for numerous platforms, including Raspberry Pi, and providing APIs in several languages. Vosk's key strength lies in its model flexibility; it offers a range of pre-trained models for various languages that vary in size from a very compact 36 MB to a more comprehensive 3.2 GB. This allows developers to make a direct trade-off between model size, resource consumption, and accuracy, making it a practical choice for hardware with limited memory and processing power. Its architecture is geared towards continuous, real-time dictation and command recognition.

`Whisper.cpp` is a C++ port of OpenAI's Whisper model, meticulously optimized for high-performance CPU-based inference. The original Whisper model was trained on a massive and diverse dataset of 680,000 hours of multilingual audio, giving it exceptional "zero-shot" performance, meaning it exhibits high accuracy on a wide range of accents, languages, and noisy conditions without specific fine-tuning. `Whisper.cpp` translates this power into an efficient, dependency-free implementation that can run on platforms like the Raspberry Pi. It supports various optimizations, including integer quantization and architecture-specific intrinsics, to maximize speed. Like the original, it offers several model sizes, from `tiny` (75 MB on disk) to `large` (2.9 GB on disk), allowing for a similar trade-off between accuracy and performance.

3.2. Accuracy and Robustness

In terms of raw transcription accuracy, Whisper is widely regarded as the leader among open-source solutions. Its extensive training data makes it incredibly robust to background noise, different accents, and varied speaking styles, all of which are critical factors in an automotive environment. Formal benchmarks on standard datasets like LibriSpeech show Whisper achieving a Word Error Rate (WER) as low as 2.7%, significantly outperforming many other open-source models. While `Whisper.cpp` can exhibit a minor accuracy drop compared to the original Python implementation due to quantization and other optimizations, tests show its accuracy remains "extremely impressive." The smaller models, such as `tiny.en` and `base.en`, still provide excellent transcription quality for clear speech.

Vosk's accuracy is generally considered good, particularly for its intended use case of real-time dictation and command-and-control. However, it does not typically reach the same level of robustness and zero-shot performance as Whisper. In some comparisons, Vosk has been found to be less accurate than other traditional ASR toolkits. Its performance is highly dependent on the specific language model chosen; larger models will yield better accuracy at the cost of higher resource usage. For the Athina project, where understanding complex or nuanced commands in a potentially noisy car cabin is essential, Whisper's superior accuracy and robustness present a compelling advantage.

3.3. Performance and Resource Usage on Raspberry Pi 5

The performance on the Raspberry Pi 5 is where the trade-offs become most apparent. Vosk, with its smaller models, is designed from the ground up for devices like the Raspberry Pi and is known to per-

form well for real-time recognition tasks. Its lightweight nature ensures that it can run efficiently without overwhelming the system.

Running `Whisper.cpp` on the Raspberry Pi 5 requires careful model selection. The larger models are not viable; the `large` model requires over 10 GB of RAM, and tests on a 4GB Pi 5 showed that the `medium.en` model consumed all available memory and swap space, causing the system to freeze. The `small.en` model, while able to run, was too slow for real-time use, taking over 30 seconds to process a 10-second audio clip. The viability for real-time transcription begins with the smaller models. The `base.en` model uses around 800 MB of memory and can process a 10-second audio chunk in approximately 10 seconds, making it borderline for real-time use. The `tiny.en` model, however, is the clear winner for this platform. It consumes about 700 MB of memory and can transcribe a 10-second audio clip in an average of 6 seconds on a Raspberry Pi 5. This is faster than real-time, ensuring that the processing queue does not become backlogged and enabling a fluid conversational flow. Tests on similar hardware (Orange Pi 5) confirm that the `tiny` model can achieve inference times under 1.5 seconds for short inputs, providing a good user experience.

4. Text-to-Speech (TTS) Engine Comparison

The Text-to-Speech (TTS) engine gives Athina its voice. The quality of this voice is paramount for a luxury brand experience, requiring natural intonation, clarity, and low latency. The synthesis must happen entirely offline on the Raspberry Pi 5. We compare `Piper TTS`, a system optimized for Raspberry Pi, and `Coqui TTS`, a powerful and flexible deep learning toolkit.

4.1. Technical Overview and Architecture

`Piper TTS` is a fast, local neural text-to-speech system developed by the Open Home Foundation with a specific focus on high performance on resource-constrained devices like the Raspberry Pi 4 and 5. It is built upon the VITS (Variational Inference with adversarial learning for Text-to-Speech) architecture, which is an end-to-end model that integrates the vocoder into a single neural network, often resulting in more natural-sounding speech. For maximum efficiency, Piper’s voice models are exported to the ONNX (Open Neural Network Exchange) format and run using the onnxruntime. This focus on optimization makes it exceptionally well-suited for the Athina project’s hardware. Piper is actively developed and is a core component of projects like Home Assistant’s “Year of the Voice.”

`Coqui TTS` is a comprehensive deep learning toolkit for TTS, originating from MozillaTTS. It is also built on advanced neural network architectures, including VITS, Tacotron, and its own powerful models like XTTS. Coqui TTS is designed to be a versatile framework for both research and production, offering extensive capabilities for training and fine-tuning models. Its flagship model, XTTS-v2, is particularly noteworthy for its high-quality, multi-lingual voice cloning, which can replicate a voice from just a six-second audio sample. While Coqui TTS is a powerful and feature-rich framework, its history is more complex. The original startup, Coqui AI, has shut down, and the project’s future now relies on community maintenance, which introduces a degree of uncertainty compared to the actively managed Piper project.

4.2. Voice Quality and Customization

The perceived quality of the synthesized voice is a critical factor. `Piper TTS` offers a range of pre-trained voice models at different quality levels (low, medium, high), which correspond to audio sample rates and model parameter counts. The “high” quality models use 22.05 kHz audio and have 28-32 million parameters, producing very natural and clear speech. There is a growing library of community-contributed voices with permissive licenses, including several high-quality US and UK English female voices like “LJSpeech (high)” and “Cori (high).” These voices are trained for hundreds or thousands of

epochs, resulting in a polished and pleasant output. Piper also provides clear documentation for training new custom voices, though it requires a quality dataset and access to a GPU for the training process itself.

`Coqui TTS` is renowned for its advanced voice cloning capabilities, particularly with the XTTS-v2 model. This allows for unparalleled customization, as the Athina team could create a unique brand voice by cloning a professional voice actor's recording. XTTS-v2 supports 17 languages and can even perform cross-language voice cloning. The quality of the output is generally high, operating at a 24kHz sampling rate. However, the flexibility of Coqui comes with greater complexity. While it can run on a CPU, it is more resource-intensive than Piper, and achieving optimal quality and performance often benefits from a GPU. The quality of single-speaker models can vary, and the best results are typically achieved with the more advanced VITS or XTTS models.

4.3. Performance, Latency, and Resource Usage

For an in-car assistant, latency is crucial. The time from when the STT engine finalizes the text to when the user hears the first sound of the response must be minimal. `Piper TTS` is explicitly designed for low-latency, local inference. Its ability to stream raw audio as it's produced allows for a near-instantaneous response, which is ideal for interactive conversation. Combined with its optimization for the Raspberry Pi's ARM architecture and efficient ONNX runtime, it is expected to deliver excellent performance with low CPU and memory overhead on the Pi 5. One user building a DIY assistant found Piper to be the fastest local TTS option available.

While specific latency benchmarks for Coqui TTS on a Raspberry Pi 5 are not available, its more complex models, like XTTS, are inherently more computationally demanding than Piper's. Running XTTS on a CPU is possible but will likely result in higher latency and CPU usage compared to Piper. General TTS latency benchmarks (not on Pi 5) highlight the importance of streaming capabilities for reducing perceived delay. Picovoice Orca, a local streaming TTS, achieved a "First Token to Speech" (FTTS) time of 130 ms on a desktop CPU, whereas non-streaming cloud services were in the 1000-2000 ms range. Piper's streaming capability aligns with this low-latency approach, making it a strong candidate for the responsive experience required by Athina. Coqui's resource requirements for training are substantial, often requiring a powerful GPU, and while inference is less demanding, it remains a heavier solution than the purpose-built Piper.

5. Recommendations for the Athina Voice Assistant

After a thorough analysis of the available offline voice processing solutions and their performance characteristics on the Raspberry Pi 5, the following components are recommended for the development of the Athina super car voice assistant. These selections prioritize a balance of high performance, superior user experience, development flexibility, and robust offline capability.

5.1. Recommended Wake Word Engine: openWakeWord

For the wake word component, **`openWakeWord`** is the primary recommendation. Its open-source nature, under the permissive Apache 2.0 license, provides the Athina development team with maximum control, transparency, and freedom from licensing costs. The innovative training methodology, which uses 100% synthetic data, dramatically simplifies the process of creating a unique, high-quality wake word like "Hey Athina," making it accessible without the need for extensive and costly data collection campaigns.

From a technical standpoint, `openWakeWord` is highly compelling. Its performance on Raspberry Pi-class hardware is proven to be efficient, capable of running numerous models on a single core of a Pi 3, which ensures it will have a negligible resource footprint on the far more powerful Pi 5. Furthermore,

its accuracy claims are robust, with a stated goal of outperforming leading commercial alternatives like Porcupine. The built-in integration of Speex noise suppression and Silero Voice Activity Detection (VAD) is a significant advantage for an in-car environment, as it will help mitigate false activations from road noise, music, and other non-speech sounds, leading to a more reliable user experience.

While Picovoice Porcupine is an excellent and highly efficient commercial alternative, its proprietary nature and the licensing costs associated with a commercial product make `openWakeWord` a more strategically advantageous choice for the Athina project, offering comparable or superior performance with greater flexibility.

5.2. Recommended Speech-to-Text Engine: Whisper.cpp

For the Speech-to-Text component, **`Whisper.cpp`** is the clear recommendation, utilizing the **`tiny.en`** or **`base.en`** model. The primary driver for this decision is Whisper's state-of-the-art accuracy and robustness. Trained on an unparalleled dataset, it excels at understanding speech in challenging real-world conditions, including the variable background noise and diverse accents that will be encountered in a vehicle. This superior accuracy is fundamental to user satisfaction, as it minimizes frustrating transcription errors and ensures user commands are understood correctly the first time.

Performance analysis confirms that the Raspberry Pi 5 is capable of running the smaller Whisper models in real-time. The `tiny.en` model, in particular, processes audio faster than it is recorded, consuming a manageable ~700 MB of memory. This leaves sufficient system resources for the other components of the voice stack. While Vosk is a capable engine designed for edge devices, it cannot match the sheer accuracy and robustness of the Whisper architecture. For a premium product like the Athina assistant, the slight increase in resource consumption for `Whisper.cpp` is a worthwhile trade-off for its significant gains in transcription quality. The active development community around `Whisper.cpp` also ensures continuous improvements and optimizations for platforms like the Raspberry Pi.

5.3. Recommended Text-to-Speech Engine: Piper TTS

For the Text-to-Speech component, **`Piper TTS`** is the unequivocally recommended solution. It is purpose-built for the exact use case of the Athina project: providing fast, high-quality, and local speech synthesis on a Raspberry Pi. Its architecture is heavily optimized for the ARM platform using the efficient ONNX runtime, ensuring minimal latency and low resource usage. The ability to stream audio as it is generated is a critical feature for creating a responsive, conversational feel, drastically reducing the perceived delay between a command and its spoken response.

The quality of Piper's voice models is another key advantage. The availability of "high" quality female voices, trained on extensive datasets and released under permissive licenses, provides an excellent starting point for Athina's voice. These voices sound natural and clear, befitting a luxury user experience. The active development of Piper, driven by its adoption in major open-source projects like Home Assistant, provides confidence in its long-term support and continued improvement.

While Coqui TTS offers powerful voice cloning features, it is more resource-intensive, and the uncertainty surrounding its maintenance makes it a riskier choice for a long-term commercial project. Piper offers a more direct, reliable, and efficient path to achieving a high-quality, low-latency offline voice for the Athina assistant, making it the superior choice.

References

coqui/xtts · Is there any way to run this demo on cpu? - Hugging Face (https://huggingface.co/spaces/coqui/xtts/discussions/6)

TTS 0.22.0 documentation - Coqui (https://docs.coqui.ai/en/latest/index.html)

Tutorial For Nervous Beginners - TTS 0.22.0 documentation - Coqui (https://docs.coqui.ai/en/latest/tutorial_for_nervous_beginners.html)

STT Fast, Lean, and Ubiquitous / Blog / Coqui (https://coqui.ai/blog/stt/deepspeech-0-6-speech-to-text-engine)

How To Check Your Raspberry Pi 5 CPU & RAM Usage - SlashGear (https://www.slashgear.com/1569801/how-to-check-raspberry-pi-5-cpu-ram-and-usage/)

Text-To-Speech Applications With Raspberry Pi | Restackio (https://d2wozrt205r2fu.cloudfront.net/p/text-to-speech-knowledge-raspberry-pi-cat-ai)

General question about memory use estimates on TTS models - GitHub (https://github.com/coqui-ai/TTS/discussions/942)

Easy Guide to Text-to-Speech on Raspberry Pi 5 Using Piper TTS - Medium (https://medium.com/@vadikus/easy-guide-to-text-to-speech-on-raspberry-pi-5-using-piper-tts-cc5ed537a7f6)

Best Offline Text-to-Speech (TTS) for Raspberry Pi in 2024: Is Piper ... - Reddit (https://www.reddit.com/r/RASPBERRY_PI_PROJECTS/comments/1c5pxhv/best_offline_texttospeech_tts_for_raspberry_pi_in/)

GitHub - rhasspy/piper: A fast, local neural text to speech system (https://github.com/rhasspy/piper)

Is OpenAI Whisper, Google TTS, or Piper TTS faster? ⏩ - Medium (https://augmentedstartups.medium.com/is-openai-whisper-google-tts-or-piper-tts-faster-d58508300bdb)

Transcription and speech synthesis — Raspberry Pi Official Magazine (https://magazine.raspberrypi.com/articles/transcription-and-speech-synthesis)

Raspberry Pi 5 Benchmarks: Significantly Better Performance ... - Phoronix (https://www.phoronix.com/review/raspberry-pi-5-benchmarks)

Vosk is better than whisper for voice dictation. - Reddit (https://www.reddit.com/r/RSI/comments/1dst9xf/vosk_is_better_than_whisper_for_voice_dictation/)

Accuracy Benchmarks: Top Free & Open-Source Speech-to-Text Offerings - whisperapi.com (https://whisperapi.com/accuracy-benchmarks-top-free-open-source-speech-to-text-offerings)

Comparing 4 popular open-source speech-to-text neural network models - Medium (https://medium.com/@nick.nagari/comparing-4-popular-open-source-speech-to-text-neural-network-models-92676a9f9265)

Compare vosk-api vs whisper - libhunt (https://www.libhunt.com/compare-vosk-api-vs-whisper)

Audio transcription with OpenAI Whisper on Raspberry Pi 5 - Medium (https://gektor650.medium.com/audio-transcription-with-openai-whisper-on-raspberry-pi-5-3054c5f75b95)

speech-recognition-experiments - GitHub (https://github.com/fquirin/speech-recognition-experiments)

Whisper and other speech recognition models - alphacephei.com (https://alphacephei.com/nsh/2022/12/11/whisper-other.html)

openwakeword 0.6.0 - PyPI (https://pypi.org/project/openwakeword/)

Wake Word Benchmark - Picovoice (https://picovoice.ai/docs/benchmark/wake-word/)

Best wake word detection engines? - Reddit (https://www.reddit.com/r/speechtech/comments/1an082e/best_wake_word_detection_engines/)

GitHub - Picovoice/porcupine (https://github.com/Picovoice/porcupine/)

Porcupine Wake Word FAQs - Picovoice (https://picovoice.ai/docs/faq/porcupine/)

Bryce's Piper TTS Voices (https://bryce.co/piper-tts-voices/)

Piper Voice Samples (https://rhasspy.github.io/piper-samples/)

Piper TTS Voice Training Guide (https://github.com/rhasspy/piper/blob/master/TRAINING.md)

rhasspy/piper-voices at main - Hugging Face (https://huggingface.co/rhasspy/piper-voices/tree/main)

Piper-voices - PromptLayer (https://promptlayer.com/community/provider/rhasspy/piper-voices)

Home Assistant Piper Custom Voice - Reddit (https://www.reddit.com/r/homeassistant/comments/18w5k1p/piper_custom_voice/)

Coqui Dialogue Audio Pack - Coqui (https://coqui.ai/cpml-1-0-2-dialogue-audio-pack)

XTTS-v2 is here! - Coqui (https://coqui.ai/blog/tts/xtts-v2)

coqui_tts extension - Oobabooga (https://github.com/oobabooga/text-generation-webui/blob/main/extensions/coqui_tts/README.md)

Coqui TTS Documentation - GitHub (https://github.com/coqui-ai/TTS)

Coqui Studio - Coqui (https://coqui.ai/platform)

Coqui TTS Voice Samples (https://coqui.ai/cpml-1-0-2-voice-samples)

Coqui TTS Voice Samples (GitHub) (https://coqui.ai/cpml-1-0-2-voice-samples-github)

What are the differences between the major open source TTS projects? - Reddit (https://www.reddit.com/r/MachineLearning/comments/133hanr/d_what_are_the_differences_between_the_major_open/)

Compare TTS vs piper - libhunt (https://www.libhunt.com/compare-TTS-vs-piper)

Text to Talk: Analyzing Open-Source TTS Alternatives - tderflinger.com (https://www.tderflinger.com/en/text-to-talk-analyzing-open-source-tts-alternatives)

TTS Comparison Guide - chirptts.com (https://www.chirptts.com/blog/tts-comparison-guide)

Open-Source TTS Alternatives Compared - smallest.ai (https://smallest.ai/blog/open-source-tts-alternatives-compared)

piper - libhunt (https://www.libhunt.com/r/piper)

Raspberry Pi 5 Product Brief - raspberrypi.com (https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf)

Raspberry Pi 5 Technical Specifications and Mechanical Drawings - element14.com (https://community.element14.com/products/raspberry-pi/b/blog/posts/raspberry-pi-5-technical-specifications-and-mechanical-drawings)

Raspberry Pi 5 Specs - deepseadev.com (https://www.deepseadev.com/en/blog/raspberry-pi-5-specs/)

Raspberry Pi 5 - raspberrypi.com (https://www.raspberrypi.com/products/raspberry-pi-5/)

Raspberry Pi 5 - raspians.com (https://raspians.com/raspberry-pi-5/)

Pi5 Raspberry Pi 5 specs - dataconomy.com (https://dataconomy.com/2023/09/28/pi5-raspberry-pi-5-specs-bcm2712/)

Benchmarking Raspberry Pi 5 - raspberrypi.com (https://www.raspberrypi.com/news/benchmarking-raspberry-pi-5/)

Raspberry Pi 5 Benchmarks - Raspberry Pi Forums (https://forums.raspberrypi.com/viewtopic.php?t=357062)

Raspberry Pi 5 with Hailo-8L Benchmark - Hailo Community (https://community.hailo.ai/t/raspberry-pi-5-with-hailo-8l-benchmark/746)

Text-to-Speech Latency Benchmark - Picovoice (https://picovoice.ai/docs/benchmark/tts-latency/)

tts-latency-benchmark - GitHub (https://github.com/Picovoice/tts-latency-benchmark)

TTS Arena: A Head-to-Head Battle of Text-to-Speech Models - Hugging Face (https://huggingface.co/blog/arena-tts)

whisper.cpp - GitHub (https://github.com/ggml-org/whisper.cpp)

Federated Finetuning of Whisper on Raspberry Pi 5 - Flower Labs (https://flower.ai/blog/2023-10-11-federated-finetuning-of-whisper-on-raspberry-pi-5/)

Real-time audio transcription with Whisper on Raspberry Pi 5 - Medium (https://gektor650.medium.com/real-time-audio-transcription-with-whisper-on-raspberry-pi-5-3054c5f75b95)

NPU support for whisper.cpp - GitHub Discussions (https://github.com/ggerganov/whisper.cpp/discussions/1016)

whisper.cpp on Raspberry Pi 4 - GitHub Issues (https://github.com/ggerganov/whisper.cpp/issues/153)

whisper.cpp slower than original whisper on RPI4 - GitHub Issues (https://github.com/ggerganov/whisper.cpp/issues/113)

whisper.cpp on Orange Pi 5 - GitHub Issues (https://github.com/ggerganov/whisper.cpp/issues/463)