Episode Book

EPISODE #6734

Open Source Voice Assistants and MCP

PUBLISHED	22 November 2026 at 14:04 UTC
DURATION	28:41
LISTEN	myweirdprompts.com/episode/open-source-voice-assistants-40761/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	22 November 2026 at 16:28 UTC

SYNOPSIS

The open-source AI assistant landscape has quietly split into two distinct tiers: assistant frameworks that manage identity and tools, and voice pipelines that handle low-latency realtime audio. An exploration of the leading self-hosted platforms—including Home Assistant Assist, Open WebUI, and Letta—reveals how each bridges the Model Context Protocol (MCP) to email, calendar, and smart home controls, alongside the unexpected friction of custom wake-word hardware.

TRANSCRIPT



Corn

Every serious write-up of voice assistants lands on the same take: realtime audio is its own discipline, so you need a dedicated voice platform and you bolt the clever part on afterwards. That's the conventional wisdom, and it's mostly backwards.



Herman

Mostly. There are exactly two situations where it's true, and we'll get to them, because they're interesting. But the platforms that own your email and your house have been quietly eating the voice layer this year, and almost nobody has noticed the shape of it.



Corn

This one comes from Daniel. He wants to know what the leading open-source assistant frameworks actually are, and he's very specific about what he means by framework. Something you reach from your phone or your laptop. Self-hosted at home or cloud-deployed, but leaning self-hosted. Wired to tools over MCP. And then his scenario: a conversation where he says, please check my email and summarise what I got, or send one, or fire off a Home Assistant command. His real question is the one underneath all that, though. Is a separate voice agent platform required to get voice at all, or is there a general-purpose assistant framework that can have voice added, including custom wake words?



Herman

And the honest answer is that the field has split into two layers that are converging. The interesting part is what each layer owns, and the fact that the most complete answer comes from a place nobody expected, which is home automation.



Corn

Two layers. Spell out the split.



Herman

The assistant platforms, things like Home Assistant Assist, Open WebUI, and Letta, own identity, memory, tools, and MCP plumbing. Voice, for them, is a mode you flip on. Then there are the voice-agent frameworks, LiveKit Agents and Pipecat being the big two,



Herman (continued)

which own the realtime audio pipeline, the speech-to-text, the turn detection, the interruption handling, the transport. They have deep opinions about audio and no opinions whatsoever about your inbox.



Corn

An audio framework that's never heard of a calendar.



Herman

And the practical consequence, which I'd put up front: you generally do not need a separate voice platform if your assistant platform already ships a voice mode and a wake-word story. You do need one if you want low-latency, interruptible, telephony-grade conversation, and at that point you're writing the agent yourself.



Corn

What makes this timely rather than a taxonomy lesson?



Herman

This year the assistant platforms started absorbing the voice layer. Home Assistant now ships wake-word hardware and MCP in both directions. Open WebUI ships realtime voice calls plus native MCP. Letta has a voice demo built on LiveKit. Three different companies, same direction of travel.



Corn

There's a negative finding hiding in there too, isn't there.



Herman

There is, and it's the one I keep circling. No single open-source project out there today is simultaneously a general-purpose assistant with email and calendar and home tooling, MCP-native, always-listening with a custom wake word, and self-hostable as one product. Nobody checks all four boxes. The closest is Home Assistant, and the gaps in it are



Herman (continued)

themselves instructive.



Corn

Start there, then.



Herman

Home Assistant Assist, then. The strongest single answer to Daniel's core question, because it's the one platform where the assistant, the MCP connections, voice, and custom wake words all live in the same product. And the way it works is nicer than people assume. An assistant in Home Assistant is just a configuration with its own name, its own language, and its own conversation agent. You can run several side by side. The docs' own example is one in English and one in German.



Corn

Two assistants in the same house, disagreeing in different languages.



Herman

The built-in agent runs entirely locally, handles the basic commands. Then the AI integrations, OpenAI, Anthropic, Gemini, Ollama, drop in as model-backed agents, and there's an option to try local handling first and only fall back to a model if the local agent can't cope. Which is a sensible ordering.



Corn

It is. Cheap answers to easy questions. The MCP part is where it gets interesting, and I know you've been waiting to say the two directions thing.



Herman

I have, because the detail most people miss is that Home Assistant has two MCP integrations, and they work in opposite directions. In one, Home Assistant is the client. Your conversation agent gets extra tools from an MCP server you point it at, a project



Herman (continued)

tracker, a note-taking app, a memory server, web search. In the other, Home Assistant is the server, so ChatGPT or Claude can reach in and control your devices. The docs put it plainly: if you want to talk to Claude or ChatGPT and have it turn on your lights, you need the Model Context Protocol integration.



Corn

Same protocol, both ends of the pipe. That part is elegant.



Herman

It is, and it's where the convergence story becomes concrete rather than vibes. Now, the ceiling. The MCP client in Home Assistant supports tools. It does not support prompts, resources, sampling, or notifications. That's a deliberately partial implementation.



Corn

Which means "MCP support" on the box means about a third of MCP.



Herman

Tools are the part that matters for Daniel's scenario, so in fairness it's the right third. But if a server leans on resources, it's dead on arrival. Two more mechanics worth knowing. Home Assistant re-checks the server's tool list every thirty minutes. And it can't start local command-line MCP servers itself, so stdio-only servers need a proxy, mcp-proxy is the usual one. Transport preference is streamable HTTP first, falling back to SSE, and OAuth credentials go into the application credentials store.



Corn

Half an hour to notice that I added a tool. My kind of bureaucracy.



Herman

It's a design choice, not an accident. You don't want your assistant hammering the server every thirty seconds. But there's a two-step thing people get wrong constantly, and this is where I confess I got it wrong first. After you add the MCP integration, nothing happens. You have to go to the conversation agent's own integration, open the settings, and under AI tools, select the MCP server by name. And keep device control ticked as well if you still want home control. Only then can the agent actually use the tools.



Corn

You added an integration and then stared at an agent that pretended it didn't exist.



Herman

Twenty minutes of staring. I'd ticked the box and assumed the agent would notice. This is the documented behaviour of every smart home I've ever installed, to be fair to me.



Corn

To be fair to you. Now the wake-word story, because this is where the trap lives.



Herman

The trap is real and I want to do it justice, because Home Assistant's basic approach is clever. Wake-word detection happens inside Home Assistant, not on the device. Satellites constantly sample room audio, and when they detect a voice, they stream it to Home Assistant, which checks whether the wake word was said and then handles whatever followed. The docs have a line I love: any device that streams audio can be turned into a voice satellite, even if it isn't powerful enough to run wake-word detection locally.



Corn

Which means the dumbest speaker in your house is now a satellite.



Herman

Which means old hardware gets useful again, which is the most Corn-compatible feature in all of consumer AI.



Corn

It is, and I resent how much I approve of it. So where's the trap?



Herman

Two incompatible wake-word technologies. openWakeWord, David Scripka's project, is server-side detection, for satellites that just stream audio. Home Assistant's own guide for training your own wake word has you generate voice clips using Piper, the local neural text-to-speech system, and it warns that depending on the word, training a model may take a few iterations and a bit of tweaking. Then there's microWakeWord, which runs on the device itself, on the ESP32 microcontroller, and that one is required for the Voice Preview Edition hardware, the little box they actually sell.



Corn

So the skill you learned for your old satellites doesn't transfer to the product they ship.



Herman

Not at all. The community answers are blunt. openWakeWord models do not work on the Voice Preview Edition, because that requires microWakeWord models that can run locally on the ESP32. And out of the box, the device page offers exactly three wake words: Jarvis, Mycroft, and Okay Nabu.



Corn

Jarvis, Mycroft, or Okay Nabu. Pick your personality disorder.



Herman

And here's the four-month gap that proves the trap is real, and this is a documented thread. April this year, a user trains a custom openWakeWord model, the word was Megan, and it doesn't appear anywhere on their Voice Preview Edition device page. The fix arrives in August. It involves training a microWakeWord model and flashing custom firmware through the ESPHome Builder.



Corn

Four months between custom wake words are supported and custom wake words work on the hardware you sell.



Herman

The workaround is confirmed working, to be clear. ESPHome documents its micro wake-word component properly, and you can point it at a model file from a GitHub shorthand. But it is a firmware-flashing exercise, not a settings toggle. Custom wake words are supported, with an asterisk the size of a microcontroller.



Corn

We talked about building your own speech recognition from scratch years ago on this show, and I remember thinking the training data part would be the hard bit. Turns out the hard bit is which of two models you're allowed to train at all.



Herman

The hardware decides the model, and nobody tells you that until you're already annoyed. One more thing on the email scenario, because Daniel's example was please check my email and summarise what I got. Home Assistant has no built-in send-email action in the assistant sense, as far as I've ever seen documented. Email would arrive the way everything else does now, through an MCP server, a Gmail server or similar, wired to the agent.



Corn

Every capability in your house enters through the same door. Some door.



Herman

Now the platforms that started from chat rather than home control, because that's where the two-layer split gets its cleanest confirmation. Open WebUI first, since it's the closest thing to a general assistant framework you already run, with voice added on. Native MCP arrived at version zero point six point three one. You configure it under integrations, external tool servers, type set to MCP, streamable HTTP. Servers are admin-only and need the instance secret key. Native support is streamable HTTP only, connecting on the stateless July revision of the protocol and older.



Corn

Now the interesting parallel.



Herman

The stdio problem, solved twice. Home Assistant can't launch local command-line MCP servers, so it tells you to run mcp-proxy. Open WebUI hit the identical friction and built Tether, in zero point thirteen point four, specifically so that servers distributed as local commands work with no proxy and no port to open. Tether starts the command on your machine and passes its messages through.



Corn

Two of the leading self-hosted platforms, same wall, two different doors through it.



Herman

Which tells you the friction is real and structural, not a design failure on either side. Nobody wants to open a port on a home network to give their chatbot a to-do list.



Corn

The Open WebUI voice mode. What's actually happening when you press the call button?



Herman

Two modes, set by the admin under audio settings. Standard uses the instance's own speech-to-text and text-to-speech engines. Realtime connects the call to a speech-to-speech model from an OpenAI-compatible realtime API or Azure, and that voice model hears you directly and hands the real questions to the chat's model. The behaviour details are where it gets charming. Recording starts when it hears you and sends after about two seconds of silence.



Corn

It gives up on you after two seconds. We'd get along.



Herman

The reply is spoken sentence by sentence as it streams, so a long answer starts playing before the model has finished writing it. Reasoning and tool-call details are not read aloud, which the docs state with a straightness I respect. Tap the model's picture to interrupt, or switch on voice interruption in call and you can just talk over it.



Corn

So it prefaces a system prompt telling the model it's in a call.



Herman

It does, asking for short, warm, conversational answers, and admins can replace the template. Permissions are per role, and in realtime mode the permission gets re-checked when the call connects and then every minute during the call.



Corn

Every minute. It trusts you in sixty-second instalments.



Herman

The hard limits, though, because this is the gap versus Home Assistant. Voice mode talks to one model at a time. It needs a server-side speech-to-text engine, so if the admin's transcription is set to the browser's web API, voice mode is simply unavailable. Browsers only allow microphone access over HTTPS or localhost, so opening the instance by IP from another device usually breaks the mic. And critically, no wake word. Nothing in the docs describes always-listening. It is a call-button model, end to end.



Corn

For custom wake words, it's the wrong layer entirely.



Herman

Right layer for a phone, wrong layer for a kitchen counter. Letta is next, and I'll say up front that Letta is a serious platform. Memory-first agents, real self-hosting in two shapes, a local runtime where the CLI runs agents in-process and every message, every memory block, every provider connection stays on-device with no account required, and a self-hosted app server that hosts agents on an always-on machine and exposes them to client apps through the SDK. MCP in both directions too, they run a hosted MCP server so any MCP client can create stateful Letta agents that keep memory across conversations.



Corn

Serious platform. So what's the voice story?



Herman

A repository called letta-voice. Twenty-eight stars, MIT licence, created February of last year, and the readme says, quote, advanced voice mode, but with advanced memory. It outsources the audio to LiveKit for transport plus Deepgram and Cartesia accounts, and the self-hosted deployment is Docker plus ngrok.



Corn

Twenty-eight stars. The voice story of a serious agent platform is a weekend demo.



Herman

It's the purest illustration of the two-layer answer you could ask for. The assistant layer and the voice layer, joined by a glue repo rather than a product. No wake word. And it's not an embarrassment, that's the thing. It's the correct division of labour for where the project is. They know audio isn't their edge.



Corn

And that gets us to the voice layer itself, and the two situations where you actually need it.



Herman

LiveKit Agents and Pipecat. Both open-source Python frameworks for realtime voice agents, and they differ in how opinionated they are about transport. Pipecat is BSD-licensed, maintained by Daily, and transport-agnostic. The same bot can run over Daily, peer-to-peer WebRTC, a raw WebSocket from Twilio or Asterisk, or even a LiveKit room. It has serializers for the telephony carriers, Twilio, Telnyx, Plivo. LiveKit Agents is Apache-licensed, Python and Node, tied to LiveKit rooms over WebRTC, but in exchange you get dispatch, load balancing, SIP telephony, and an audio turn-detection model as one integrated package.



Corn

Some teams run both, don't they.



Herman

Through Pipecat's LiveKit transport, yes. You can sit on LiveKit's transport without adopting LiveKit's runtime. Neither framework ships a wake word, an email client, or a Home Assistant integration, and that's not a gap, that's the boundary. These are the separate voice platforms Daniel is asking about, and the answer is you reach for them for latency and telephony, not for voice as such.



Corn

Which reframes his whole question. Voice isn't the thing you add. Latency is.



Herman

If you want a thing your family can interrupt mid-sentence while it's dialled into a phone system, that's a LiveKit or Pipecat project and you're writing the agent. If you want to say a word across the room and have your house answer, that's Home Assistant, and the microphone hardware is the only part you buy.



Corn

Before you move on, the long tail. I want to hear about the long tail.



Herman

The long tail is my favourite part of this whole survey, because it shows the pattern has become a weekend project. Five solo developers, all within the same stretch of time, all converging on essentially the same stack. hey-spark, an always-listening laptop assistant, openWakeWord plus faster-whisper plus MCP servers, running locally on CPU, and it asks out loud before doing anything that touches a tool.



Corn

It requests permission verbally before acting. The only assistant in this entire survey with manners.



Herman

domia-core runs a full speech-to-speech pipeline with per-sentence pipelining, so it starts talking before the whole answer is generated, speech inference in-process via sherpa-onnx, local model on a llama server, MCP skills opt-in. lia-framework does Silero voice detection with an MCP plugin system. SpeechKit is Go, in beta, pin your versions, and it exposes itself as an MCP server. And then havencore, which runs fourteen containers locally, a local mixture-of-experts model on vLLM, Kokoro for the voice, wake word, transcription, tools over MCP for Home Assistant and Plex and web search and image generation, plus proactive scheduled runs, and the readme says everything is one docker compose up.



Corn

Fourteen containers to tell me the kettle's on.



Herman

Fourteen containers, one command, and a hardware bill that could quietly outrun the laptop. When five solo developers converge on the same stack in the same year, the framework question stops being about which product to adopt and starts being about whether a reference architecture has quietly standardised underneath everyone.



Corn

The stack standardised and nobody signed anything. It's the airport theory of open source.



Herman

There's an older lineage worth a minute, too, because it predates MCP entirely and shows the self-hosted assistant dream isn't new. Leon, MIT-licensed, TypeScript, first released in twenty nineteen, seventeen and a half thousand GitHub stars, runs on about two gigabytes of memory, and the pitch is that Leon lives on your server and you decide if you wish to make use of any third party. Willow, Apache-licensed, built on Espressif's ESP-IDF, targeting the ESP32-S3-BOX hardware, three thousand stars, about fifty dollars of hardware, claims response times under half a second, wake words Hi ESP or Alexa.



Corn

Alexa. On the box of the escape-from-Alexa machine.



Herman

They mean it as a compatibility claim, Alexa-grade responsiveness on hardware you own. Willow is honest hardware-project energy, kitchen-counter device, everything self-hosted, and it pairs with Home Assistant properly. Leon is the server-side purist's version of the same dream, from before anyone had thought to put a protocol between the assistant and its tools.



Corn

The older generation built the house. This generation is arguing about the wiring standard.



Herman

Which is progress, honestly. The wiring standard is the fun argument.



Corn

Hold on, we've run long enough that we owe the listeners a meal. It's time for Dorothy's Kitchen, with Corn's mother on the line. Dorothy, what are you cooking for us today?

D

Dorothy

Hello dear, it's your mother. It's Dorothy's Kitchen again, and I've got a recipe for you. It's a cup of hot water. Don't laugh, sweetheart, it's harder than it sounds.



Corn

We're not laughing, Mum.

D

Dorothy

You'll need water, one cup of it. A kettle, if you have one. And a clean cup, which is the part people skip and then they wonder why it tastes of yesterday.



Herman

Is the cup rinsed first or just clean from the shelf?

D

Dorothy

Now the method. Fill the kettle and put it on. Then wait. Waiting is most of it, and you mustn't wander off, because that is when things go wrong, and I would rather not say how I know that.



Corn

She once abandoned a kettle during a phone call in nineteen ninety-four and we still hear about it.

D

Dorothy

When it comes to the boil, pour it slowly into the cup and mind your fingers. Then let it sit a moment. If you rush it you'll burn your mouth, and then the whole thing has turned on you.



Herman

Mind the fingers. Noted.

D

Dorothy

That's it. Lovely. Now, are you eating properly? You sound thin. Ring me Sunday, dear. Bye bye now.



Herman

Corn, before we settle the verdict, the thing I keep coming back to is the fridge question. If you had to give Daniel one recommendation today, and it had to survive contact with a real family, which layer do you point him at?



Corn

Home Assistant, and I don't think it's close. Not because it's the best assistant, it isn't, it's a home automation platform that learned to talk. But it's the only one where the word assistant, the MCP plumbing, the voice, and the wake word live in the same product with the same vendor's name on the box. When his Voice PE satellite hears Megan and the firmware is wrong, there is one company to be annoyed at.



Herman

And the annoyance is documented and solvable, which in this space counts as a feature. The August firmware path works. It's ugly, it's flashing a device you paid for, but it's a path.



Corn

Whereas if he builds the Letta plus LiveKit version, the wake-word problem doesn't exist because nothing is listening until he presses a button.



Herman

Which is the trade he'd be making. He asked for always-listening with a custom wake word, self-hosted, general-purpose, MCP-native, all four. Home Assistant gets him three and a half, the half being email, which arrives through an MCP server rather than being built in. Open WebUI gets him the general-purpose chat and the MCP but no wake word. Letta gets him memory and no voice to speak of. Everybody gets him three.



Corn

Three out of four, five different ways.



Herman

So the two-layer split is structural, and the convergence is real, and the wake-word trap is still a firmware-flashing exercise. The question that leaves us with is whether that convergence finishes, whether one layer eats the other, or whether the layer that wins voice is simply always going to be a different layer from the one that wins assistants.



Corn

You're asking whether the microphone and the inbox are the same company in ten years.



Herman

History mostly says the microphone wins the living room and the inbox wins everything else, and they meet in the middle at a protocol.



Hilbert

Corn. That Piper trick for the wake-word training clips. Did either of you count how many clips it takes before the thing stops going off at the television?



Corn

The docs only say a few iterations and a bit of tweaking.



Hilbert

Eleven. Eleven goes at it, on mine. I run a satellite that streams audio, so detection happens on the Home Assistant side, and I trained my own word rather than say Jarvis to my own hallway. The word is well, go on then. Don't ask me why. It's what I say to the house anyway and it seemed rude to change.



Herman

And the television kept setting it off?



Hilbert

The six o'clock news, every night, rain or no rain. What fixed it in the end was forty clips of the weather presenter as negatives, fed in alongside the Piper samples. Twelfth run, silence. Then the bird started. Ronnie next door keeps an African grey called Kipper, and Kipper picked up the phrase off me through the open window in six days. Six. My model took eleven attempts and a parrot managed it in under a week, and now it turns my hallway lamp on whenever it fancies. We have an arrangement, Ronnie and I. The cage goes out to the balcony before nine, and I pay for the seed.



Corn

The parrot is faster than the model and cheaper to run.



Hilbert

Kipper also interrupts, which the model waits to be asked. I priced the Voice Preview Edition box when it came out, fifty-nine dollars, and didn't buy it, because my model wouldn't run on the thing anyway. It wants microWakeWord, and I wasn't training the word twice.



Herman

So the trap closed on you from both ends. The openWakeWord model you already own won't transfer, and retraining means starting over on the ESP32.



Hilbert

That's the shape of it. Sixteen weeks, that would have taken me, by the thread timings. Kipper did it in six days for free.



Corn

Six days, and it never needed the weather negatives.



Hilbert

No. It just didn't watch the news.



Corn

I'm going to sit with the six days. Not the eleven.



Herman

The bird never gets a firmware update either, so it's the only wake-word implementation in this entire episode that won't break in an upgrade. Half an hour to notice a new tool, four months to accept a new word, and a parrot that just does it. That's the state of the art.



Corn

The open question I'd leave with Daniel is the convergence one. Home Assistant absorbed voice, Open WebUI absorbed voice, Letta bolted it on through a twenty-eight-star demo. Every layer is reaching for the other one, and none of them has arrived. Whether that ends in one product or a permanent seam, nobody has built the thing that settles it. And under that, the quieter question, whether five solo developers converging on the same stack in the same year means the framework question has already answered itself, and the choice is now which end of a standardised architecture you want to stand on.



Herman

The wake-word gap is the one I'd watch. It's the only user-facing problem in the whole survey that got worse as the hardware got better, and until it's a settings toggle rather than a firmware flash, the most complete self-hosted assistant still has a trapdoor right in the middle of its headline feature.



Corn

Thanks as always to our producer, Hilbert Flumingtop, for keeping us on the air and the lamp on in the hallway. If you've enjoyed the show, a review wherever you listen helps other people find us. For more along these lines, there's episode two, Local STT For AMD GPU Owners. This has been My Weird Prompts. See you soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Home Assistant, Assist [primary]**
   Talk to your smart home, retrieved 2026-11-22
   https://www.home-assistant.io/voice_control/

2. **Home Assistant, Model Context Protocol Client integration docs, retrieved 2026-11-22 [primary]**
   https://www.home-assistant.io/integrations/mcp/

3. **Home Assistant, Model Context Protocol (server) integration docs, retrieved 2026-11-22 [primary]**
   https://www.home-assistant.io/integrations/mcp_server

4. **Home Assistant, The Home Assistant approach to wake words, retrieved 2026-11-22 [primary]**
   https://www.home-assistant.io/voice_control/about_wake_word/

5. **Home Assistant, Wake words for Assist, retrieved 2026-11-22 [primary]**
   https://www.home-assistant.io/voice_control/create_wake_word/

6. **ESPHome, Micro Wake Word component docs, retrieved 2026-11-22 [primary]**
   https://esphome.io/components/micro_wake_word/

7. **Open WebUI, Model Context Protocol (MCP) docs, retrieved 2026-11-22 [primary]**
   https://docs.openwebui.com/features/extensibility/mcp/

8. **Open WebUI, Voice Mode docs, retrieved 2026-11-22 [primary]**
   https://docs.openwebui.com/features/chat-conversations/chat-features/voice-mode/

9. **Open WebUI blog, v0.13.4 release (Tether, local MCP commands), retrieved 2026-11-22 [primary]**
   https://openwebui.com/blog/v0-13-4-ai-that-does-the-work-not-just-answers

10. **Open WebUI blog, v0.12.0 release (Realtime voice), retrieved 2026-11-22 [primary]**
    https://openwebui.com/blog/v0-12-0-a-voice-that-talks-back-and-a-face-to-go-with-it

11. **Open WebUI, Tools docs, retrieved 2026-11-22 [primary]**
    https://docs.openwebui.com/features/extensibility/plugin/tools/

12. **Letta, Self-hosting docs, retrieved 2026-11-22 [primary]**
    https://docs.letta.com/self-hosting

13. **Letta, Hosted Letta MCP server docs, retrieved 2026-11-22 [primary]**
    https://docs.letta.com/platform/hosted-mcp/index.md

14. **Letta, `letta-voice` repo (MIT, created 2025-02-19), retrieved 2026-11-22 [primary]**
    https://github.com/letta-ai/letta-voice/

15. **Letta, `letta-code` repo (Apache 2.0, created 2025-10-25), retrieved 2026-11-22 [primary]**
    https://github.com/letta-ai/letta-code/

16. **LiveKit, `agents` repo README, retrieved 2026-11-22 [primary]**
    https://github.com/livekit/agents/

17. **LiveKit blog, The best open source frameworks for building realtime voice and video AI agents, retrieved 2026-11-22 [primary]**
    https://livekit.com/blog/best-open-source-voice-and-video-ai-agent-frameworks

18. **Leon, project homepage, retrieved 2026-11-22 [primary]**
    https://getleon.ai/

19. **Willow, project homepage, retrieved 2026-11-22 [primary]**
    https://heywillow.org/

20. **Willow repo (Apache 2.0, 3,144 stars, created 2023-03-31), retrieved 2026-11-22 [primary]**
    https://github.com/HeyWillow/willow

21. **`hey-spark` repo README (MIT, openWakeWord + faster-whisper + MCP), retrieved 2026-11-22 [primary]**
    https://github.com/contextuai/hey-spark

22. **`domia-core` repo README (S2S pipeline, sherpa-onnx-node), retrieved 2026-11-22 [primary]**
    https://github.com/domia-ai/domia-core

23. **`havencore` repo README (14 containers, GLM-4.5-Air on vLLM, Kokoro TTS), retrieved 2026-11-22 [primary]**
    https://github.com/ThatMattCat/havencore

24. **`lia-framework` repo README (Silero VAD, MCP plugin system), retrieved 2026-11-22 [primary]**
https://github.com/zahanzo/lia-framework

25. **`SpeechKit` repo README (Go, beta, MCP server), retrieved 2026-11-22 [primary]**
https://github.com/kombifyio/SpeechKit/