Episode Book

EPISODE #6549

Plugging a Custom Voice Model into Wayland

PUBLISHED	13 November 2026 at 12:33 UTC
DURATION	26:03
LISTEN	myweirdprompts.com/episode/custom-voice-model-wayland-65049/
HOSTS	Corn, Herman, Desmond, Hilbert
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
BOOK GENERATED	14 November 2026 at 06:47 UTC

SYNOPSIS

Most open-source voice projects assume you need speech recognition, leaving a developer who already trained their own models with nowhere to plug them in. Exploring tools like VoiceMode, Fono, VoiceUse, and the OpenAI Agents SDK reveals a gap between voice channels and actual Wayland desktop control. The conversation covers how to pair a fast command classifier with an LLM fallback, and why Wayland's input-blocking security forces workarounds through compositor-specific APIs, ydotool, wtype, or libei.

TRANSCRIPT



Corn

Daniel has already done the part that sounds hard. He trained his own speech recognition model, and he trained a second model that turns a spoken phrase into a command. What he doesn't have is anything on the far side of them. He wants to say "open Chrome", "close Chrome", and then something looser, to a Linux laptop running Wayland, and have it happen. We're skipping the command grammar tools and the dictation apps on purpose, because for him they're a dead end.



Herman

So the useful news comes first, and it's mixed. Nobody has built this end to end on a Wayland desktop. Not a mature project, not even a popular immature one. What exists is a scatter of early projects that each cover one slice, and the slice most of them cover is the least interesting one for Daniel, which is the voice channel.



Corn

He has the voice channel. He built it.



Herman

He built the front of it, yes. And that's what makes the field look odd from where he's standing. Most of these projects are designed to hand you speech recognition, and he's looking for somewhere to hand his in.



Corn

And there are plenty of those, nearly all pointed at somebody's coding agent.



Herman

Fair. Let me do it by where his model plugs in, because that's what decides whether a project is any use to him. There are three styles. One is an OpenAI-compatible HTTP endpoint. VoiceMode is the example. It's an MCP server that gives Claude Code, or any agent that speaks MCP, a voice conversation. Its local services mimic the OpenAI speech API, so if Daniel serves his model behind that same shape, it should slot in. I didn't find the exact setting for the base address, the README sends you to a configuration guide, so treat that part as likely rather than confirmed.



Corn

And VoiceMode controls what, exactly?



Herman

Nothing. That's the catch. It's a voice channel into an agent that already has its own tools. Say "open Chrome" and whatever agent is on the other end decides how. It's about fifteen hundred stars on GitHub, MIT licence, Linux supported, and no wake word as far as I saw.



Corn

So the agent does the work and the project is the microphone.



Herman

That pattern repeats. The second plug-in style is Wyoming, which is the protocol Home Assistant uses to swap speech components in and out. Fono speaks it as both client and server. Fono is a Rust single binary, GPL, labelled early preview, and it does dictation, a voice assistant, and voice control for coding agents. It runs on Wayland and X eleven. And it moves fast. A version from June, then version zero point twenty point one on the seventh of November.



Corn

That's a lot of versions for five months.



Herman

Active, is the polite word. Wyoming is the confirmed way in. Whether it also takes a plain OpenAI-style endpoint, I couldn't confirm. The third style is just code. The OpenAI Agents SDK has a voice pipeline where you subclass a speech-to-text class and implement a transcribe method, and a streaming session if you want partial results. That's the most explicit custom-model hook I found anywhere. RealtimeSTT, the library, has the same idea with a custom engine guide and wake word support.



Corn

Hold on. Those two are libraries. Neither one knows what a window is.



Herman

Correct. They're sockets. You plug the model in and the rest is yours.



Corn

Then which of these is closest to a desktop that does something?



Herman

VoiceUse. A wake word, the word "Computer", through Porcupine, or hold the right control key. It has allow-listed system commands, runs them without a shell, asks for a spoken "yes" before anything destructive, and has a hand-off to an external agent over MCP. It can even do vision clicking through Codex or Anthropic's computer use. The problem is the rest of the stack. Its speech recognition defaults to hosted Whisper on Groq, and I saw no custom endpoint. All the inference is cloud. And on Linux it leans on xdotool and wmctrl, which means X eleven. The project itself recommends XWayland, not Wayland.



Corn

So a hand-built voice agent that trusts the cloud and assumes a display server Daniel isn't running.



Herman

Version zero point one point three, May. It's a good design to read. I wouldn't build on it.



Corn

What about the one with the hardware? The Open Interpreter box.



Herman

The zero one. Five thousand stars, AGPL, and the README says itself that it's under rapid development and lacks basic safeguards. It's built around a little ESP thirty-two device and a LiveKit server, with desktop clients. It's a gadget for talking to a computer that isn't the one in your hands. For a laptop he's already sitting at, it's the wrong shape.



Corn

Basic safeguards missing, on a thing that runs code. That's a bold README.



Herman

The community path is more promising. Goose from Block has voice through a community extension, Speech MCP, using local faster-whisper. There's a reported bug of it transcribing phrases out of silence, which matters more than it sounds. And two older projects are worth a sentence each. Leon is an intent-based assistant that was reportedly rebuilding around tools and agents this year, though I can't tell you whether that shipped. And OpenVoiceOS has a persona plugin that sends anything no skill handles to an LLM. That's the fallback pattern already built, but its install is a device setup, not a laptop.



Corn

So what you get is a microphone plugged into somebody's agent.



Herman

Which, if I'm Daniel, is actually fine. The voice side is a commodity. I'd choose the agent first, and then ask who gives me the cleanest socket.



Corn

I'd go further. He should not adopt anybody's whole voice stack. He already has the two parts people charge for.



Herman

Then the real design question is the one he asked second. Where does his command model sit relative to an LLM?



Corn

My instinct says it sits in front and the LLM is the apology.



Herman

For a command like "open Chrome", there's no argument. A sub-second classifier that returns a command with a confidence. If the confidence is high, execute, and the LLM never wakes up. If it's low, or the model says none of the above, then hand the transcript to the LLM with tools. Red Hat's Anthony, the GNOME voice orchestrator, does exactly that. Pattern-matched commands under a hundred milliseconds, open-ended ones to a local LLM.



Corn

Everybody draws that box. What's the part people get wrong?



Herman

The threshold, and what happens below it. Home Assistant has a setting, prefer handling commands locally. Local sentence matching first, agent only if nothing matches. In the forum there's a user who measured about one and a half to three seconds with the option off, and forty to sixty seconds with it on for a long question that didn't match. One anecdote, I'd keep that in proportion. But the shape of the failure is real. If the fallback is slow, the fast path is the only part that feels alive.



Corn

Forty seconds. At that point I'd have opened Chrome myself and gone for lunch.



Herman

There's a cleaner reference for his case. A developer called Zetaphor wrote up a fully local desktop voice agent. A small speech model, about ninety milliseconds for a short command. Then, instead of giving the LLM every tool, it embeds the tool descriptions in a vector database, matches the spoken command by similarity, and gives only the winning tool to a small LLM, about one point seven billion parameters, to fill in the parameters. Hardcoded phrases like "mute" skip the LLM entirely.



Corn

That's the same shape as Daniel's thing, with embeddings instead of a trained classifier.



Herman

That's why I like it. His classifier replaces the embedding step, probably with better accuracy on his own voice. And there's a library, semantic-router, from Aurelio. You define routes by example utterances, it picks the nearest, and it returns nothing if no route is confident. That "nothing" is your fallback trigger.



Corn

He doesn't need the library, though. His model already outputs a label and a score.



Herman

He doesn't need the library. He needs the idea that no match is a first-class result. Most people design the fast path and treat the miss as an error. Make the miss a route.



Corn

Here's what bothers me. Say the classifier is ninety-five percent sure it heard "close Chrome", and the user meant "close chrome tab". Confidence about the label isn't confidence about the intent.



Herman

Good, and that's where I'd push the threshold by the cost of being wrong. Opening something, you can be loose. Closing something, higher. Anything that deletes, sends or pays, don't trust the fast path at all, route it through a spoken confirmation. VoiceUse does a version of that, and I'd borrow it.



Corn

What would you do with a miss, Herman? The LLM answers, the user waits, and then?



Herman

I'd log the miss. Every time the LLM handles something, that transcript and the action it chose is a free training example for the classifier. After a month the fast path has eaten the common cases, and the LLM only sees the strange ones. Is that how you'd do it?



Corn

I'd add one thing. I'd tell the user out loud when it fell through. A little tone, a different one. Otherwise you can't tell slow from broken.



Herman

A tone for the slow path. Yes. I'd steal that.



Corn

And now it's time for Desmond's Money Tip.

D

Desmond

Desmond's Money Tip. Boil only exactly one cup of water in the kettle, measured, not guessed.

D

Desmond

Do that twice a day and over a year you save roughly four pounds and ten pence. Over a lifetime, perhaps two hundred and sixty pounds, which is not a house.

D

Desmond

I keep a plastic measuring jug by the kettle and I have written the word one on it in marker. I have not had a spontaneous cup of tea since twenty nineteen.

D

Desmond

Money is mostly just the slow, deliberate removal of small comforts. That is Desmond's Money Tip.



Herman

Not one spontaneous cup of tea since twenty nineteen, Corn. Put the jug down.



Corn

Back to the action itself. Does "open Chrome" need an agent at all?



Herman

No, and it's the case where people reach for the heaviest tool anyway. You launch it from its desktop entry. Closing it is a window-manager question or a process question. And the window-manager side is where Wayland gets interesting, because there's no single API.



Corn

There's no xdotool equivalent that works everywhere.



Herman

Not even close. On GNOME you go through the shell's D-Bus interface or an extension. On KDE, KWin has D-Bus scripting for listing and focusing windows. Hyprland has its own control command, and niri, the i-three window manager and sway each have IPC. So a router that says "focus the browser window" ends up with a small adapter per compositor. Daniel only needs the one he runs.



Corn

Which one does he run?



Herman

I don't know, he didn't say, and it changes the answer more than almost anything else. Though to be fair to him, we did an episode on why GNOME on Wayland breaks voice-to-text tools, and the root cause is the same one here.



Corn

Wayland doesn't let one program fake another one's keyboard.



Herman

By design. A client can't inject input into other clients, and that's the security feature that also blocks the voice tools. So each project found a different back door. The crudest is ydotool or dotool, which write to the kernel's virtual input device. Works anywhere, needs permission to that device, a daemon or a udev rule, and has no idea which window has focus. Then wtype, which only works on wlroots compositors. And the proper route is the RemoteDesktop portal with libei, the emulated input library. A small tool called eitype is built on it and works on GNOME and KDE.



Corn

So the polite way to press a key on Wayland is to ask the desktop's permission, and the impolite way is to go under the floor.



Herman

And the polite way has a permission prompt, which for a voice agent is its own comedy. You say "open Chrome" and a dialog appears that you need a mouse to click.



Corn

Is there a persistent grant?



Herman

Eitype advertises session persistence. How well that survives a reboot on either desktop, I haven't confirmed.



Corn

Fine. Back to the shell, then. For the open-ended cases, does the LLM just get a terminal?



Herman

That's the first fork. The cheap, reliable layer is command line and D-Bus. Launch things, list windows, move them, set the volume, open a URL, query the calendar. The LLM picks from a set of tools, each tool wraps a command, and you can test each one. No pixels involved.



Corn

And the second layer?



Herman

The semantic layer, AT-SPI, the accessibility tree. Every button and text field in a well-behaved app is exposed as an object with a name and a role. The agent can read the tree as text, which is cheap, and press a button by name. That works on Wayland because it isn't input injection, it's the accessibility bus. Anthony uses it to read dialogs and press buttons.



Corn

And what's the catch? There's always one.



Herman

Coverage. Native GTK and Qt apps are good. Chrome and the Electron apps are patchier unless accessibility is switched on, and I haven't confirmed how well that works in practice. So it's the first thing I'd test, since the browser is where he starts.



Corn

Chrome being the example is a bit mean, then.



Herman

I think it's a good test. If the tree is thin there, you go to the third layer, which is screenshots and a vision model.



Corn

Which is the part everybody talks about.



Herman

Because it's the fun one. There's a Rust MCP server called computer-use-linux, and it's the most complete thing I found for this. It combines most of what we just went through. The accessibility tree, portals for screenshots and input, GNOME Shell and KWin hooks for windows, Hyprland, niri, sway, and the typing tools as fallbacks. Its tool list reads like what Daniel would otherwise write himself. Click, type text, press a key, list windows, activate a window, set a value.



Corn

He could just say yes to that and move on.



Herman

With a caveat that matters. The author validated it by hand on one setup, Ubuntu twenty-five ten, GNOME fifty, Wayland. The other backends are covered by contract tests only. So if he's on KDE or Hyprland, it's a promising draft.



Corn

Does it have any brakes?



Herman

More than most. Every tool is annotated read-only, UI-mutating or destructive, though the README calls those hints, not authorization. It refuses to act on a stale snapshot, it keeps the shell tool off by default, and it masks password fields. I'd still put my own gate in front, because a hint is something a model can ignore.



Corn

What about the full computer-use agents? The ones with the big benchmark numbers.



Herman

Be careful there. Anthropic's own documentation recommends a dedicated virtual machine with minimal privileges, no sensitive data, and a human confirming anything consequential. Prompt injection through what's on screen is named as a risk. The reference setup is a Docker image running its own X eleven desktop, not your real Wayland session. And each screenshot costs on the order of a thousand to eighteen hundred input tokens, depending on size.



Corn

So I can't just point it at my own laptop.



Herman

You can, it just isn't what it was tested on. The benchmark number everybody quotes is OSWorld. The human baseline is about seventy-two percent. The leaderboards now show scores in the eighties, which is above the human, and that's mostly self-reported with different step budgets. And it's virtual machines with tidy tasks. A laptop with forty tabs and a notification in the corner is a different animal.



Corn

Forty tabs is a modest laptop.



Herman

Yours, maybe.



Corn

Mine has one tab. It's asleep.



Herman

There's an open option too. UI-TARS is an open-weight model that people use for grounding, which means finding where on the screen a thing is. Agent S three from Simular paired a large model for planning with UI-TARS for grounding, and claims over seventy-two percent on OSWorld, and Agent S four came out on the twentieth of October with no score I could confirm. I'd treat all of it as a fallback for the tail, the thing you try when nothing semantic works.



Corn

Then let me say the architecture back and you tell me what I got wrong. Voice goes into his speech model. The transcript goes to his command model. High confidence, a handler runs, a launcher or a window call. Low confidence or none, the LLM gets the transcript with a small set of tools. Those tools are command line first, accessibility tree second, screenshots last.



Herman

That's right, and I'd put one more thing between the first and second step. A cheap check on whether it was a command at all. The silent-phantom-transcription bug in the Goose extension is the version of that you don't want, a room that's quiet and a laptop that acts.



Corn

He wants wake word or push to talk.



Herman

Push to talk, at first. Fono and VoiceUse both have hotkeys. And, Corn, I'd ask you something. If you were Daniel, would you glue this together yourself, or adopt a harness like VoiceMode and give it computer-use-linux as the hands?



Corn

Adopt. Writing a voice agent is a year. Writing a router and four tools is a weekend. I'd let VoiceMode or Fono own the microphone and the agent loop, and write the part nobody else has, the fast path.



Herman

The thing I'd worry about is that your fast path then has to be inside someone else's loop.



Corn

Then it goes ahead of the loop. Before the transcript reaches the agent, his classifier gets first look. Hit, it executes and the agent never hears about it. Miss, it forwards the text as if it had been typed.



Herman

A thin proxy. That's maybe a few hundred lines. The tricky bit is the agent being told what just happened, so it doesn't redo it.



Corn

Before the glue, one more thing about the gate. A spoken yes for anything destructive sounds sensible until you consider how a yes gets heard. The same model that mishears "close" can mishear "yes".



Herman

So what would you put on the confirmation, Corn? Because I keep going back and forth. A spoken yes is natural, but it's the same channel that might be wrong.



Corn

Different channel. A key press. Say the dangerous thing, the laptop reads back what it understood, and nothing happens until you hit a key. Slow, on purpose, and only for the short list of verbs that delete, send or pay.



Herman

I like that, because it also solves the tool annotation problem. The MCP server tells you which actions are destructive, but only as a hint. Your gate keys on your own list of verbs, and the hint is a second opinion.



Corn

And the agent never sees the gate. It asks, the proxy intercepts.



Herman

The same applies to the shell. In the desktop MCP server the shell tool is off by default, and I'd leave it off. The model can launch a browser through a tool that launches browsers. It doesn't need a terminal to do that.



Corn

So the glue is a proxy, a confidence rule, a handful of command tools, a confirmation gate and a log. That's less than I feared.



Herman

It's the part that's missing from every project I read. And for a screenshot step, you budget about two thousand input tokens per screenshot, so you set a hard cap on steps per request.



Hilbert

A thousand to eighteen hundred, Herman. Not two thousand. And they want it downscaled to twelve eighty wide at most.



Herman

Fine, a thousand to eighteen hundred. The cap stays.



Hilbert

Dennis from the darts team keeps the cricket open on the club laptop on league nights. I rang him while you were on the proxy. Asked him to stand in front of it and say close Chrome. Nothing happened. It isn't that sort of laptop. The barman looked over, though.



Corn

Dennis is going to need a different laptop.



Hilbert

I don't want Dennis to have one. I want to know whether that thing of yours would close Chrome for me, from the bar, when I say it, on his machine. Not for him. For whoever says it. Herman, you keep saying sockets. Does the socket know whose voice it is?



Herman

Not by itself. A speech model turns audio into words, and a command classifier turns words into a command. Neither one checks who spoke. Speaker verification is a third model, and it sits in front of everything.



Corn

In the version where Daniel's laptop is at home with the door shut, nobody cares. In a bar the cricket goes.



Hilbert

Dennis turned it to face the wall. Took him a few seconds. He says the league is calmer.



Corn

A microphone that can't hear you because the laptop has its back turned. Cheaper than a voiceprint.



Herman

It's push to talk done with furniture. For a first version that's the answer. Nothing listens until a key is held, and speaker checks come later if ever.



Hilbert

Dennis hasn't got a right control. The key's worn off. He'd have to hold something else.



Herman

Then it's a configurable key, and Daniel picks one that exists. And the order I'd build in matters, because it decides how much he wastes. First thing, before any agent: does Chrome expose its interface tree on his machine? If it does, half the long tail never needs a screenshot. If it doesn't, he finds out in an afternoon, not a month.



Corn

Second thing is the permission prompt. If the input portal asks him to click a dialog every session, the whole voice idea is dead on arrival, and better to know before writing the proxy.



Herman

Third is which compositor he's on, because that picks the window adapter. If it's GNOME, the one tested setup for that desktop MCP server is his setup, and that's a real advantage. If it's something else, he's the second tester.



Corn

I'd still start with the fast path. A classifier plus launch and close for five apps, push to talk, no LLM at all. If that feels good by next week, the LLM is a bonus.



Herman

It's worth remembering how little the LLM has to do in that picture. It's the exception handler.



Corn

One detail that didn't fit anywhere. Bytebot is a self-hosted desktop agent that lives in a Docker container with a web interface, and it controls the container's own desktop, not the machine you're sitting at. Most of the polished computer-use demos turn out to be that: a very capable agent in a very tidy room that isn't yours.



Herman

A tidy room is also what the leaderboard scores are measured in.



Corn

The way I'd see this turning out is that Daniel doesn't wait for anyone to ship it. The fast path is his, the hands exist in at least a draft, and the part to decide is who is allowed to speak to the machine. Turning the laptop to face the wall is a pretty good floor for version one.



Herman

Thank you to Hilbert Flumingtop for producing this one.



Corn

This has been My Weird Prompts. If you like it, you'll find the rest at my weird prompts dot com.



Herman

See you tomorrow.