NERO PROJECT

Call Nero - Phone Line to AI Staff

Running my AI staff from a walk around the lake.

Paul Schindler | Built with Claude | November 2026

Executive summary

I wanted to direct my AI staff hands-free from the car, the boat or a walk, the way a CEO calls staff. I gave my AI system, Nero, a phone number. When I call, an AI desk running on Claude answers, starts a new work session or picks up one already running. Then it speaks the answers back to me. Speech recognition and the house voice run on my own hardware, and the system calls me back when a long task finishes. The line has been live since October 2026. I designed it, set its rules and tested it on real calls; AI agents wrote the code.

Live
since October 25, 2026

134
existing work sessions reachable by phone at launch

1-3 s
desk reply time once a call is under way

99
call transcripts reviewed in one pass for fixes

What is in this report

Page	Contents
2	The problem, what I ruled out and the design choices I set
3	How a call works, what runs where and how the system calls me back
4	Picking up live work - one conversation, never two copies - and measured speed
5	Lessons from real calls and the changes they drove
6	Rules and quality checks, status and next steps, then what the project shows about how I work

At a glance

My role	I set the requirements, made every design call, tested it on real calls and decided each fix. AI agents wrote the code.
The AI	Claude Opus through Claude Code, on my existing subscription
Local models	Silero (speech detection), Whisper (speech to text) and Kokoro (voice) on my home workstation
Timeline	Work began April 2026; build began October 15; live October 25, 2026

This is a public summary. It omits the phone number, the service providers, the network design and the caller checks on purpose.

1. The problem and why it mattered

My AI staff lives in work sessions on a home workstation. Each session holds the full history of one project - a renovation plan, a research question, a software build. Directing that staff meant sitting at a keyboard. Away from the desk I could read a summary on my phone, but I could not hold a working conversation.

> *I want to talk with my AI staff hands-free on a walk, on the boat or in the car - like a CEO calling staff.*
> My brief for the build, October 2026

I set four requirements before any code was written. The call must be an ordinary phone call, so a locked screen, earbuds and car audio all work. It must reach the same sessions I use at the keyboard, not a copy. Speech must run on my own hardware. And it must add no new model subscription, so the AI desk runs on my existing Claude plan.

What I ruled out

Option	Why I ruled it out
Claude's own phone app	I tested it in October 2026. Its coding sessions offered dictation and read-aloud only, with no two-way voice mode.
Voice notes over Telegram	A Telegram bot cannot make or take calls, and swapping voice notes is not a conversation.
A calling app on a private network	It needed network software running on my phone at all times. I wanted a plain phone call.
Letting my always-on messaging agent take calls	By evening it carries a full day of chatter (103,000 tokens of context by 5:30 PM on the day I measured), which would slow every spoken reply. It also has deliberately narrow powers.
Cloud speech and voice services at launch	Local models kept my voice on my own hardware and cost nothing per call. I revisited this in November 2026 (page 3).

The design choices I set

1. **One opening question.** Every call starts with "New conversation, or pick one up?"
2. **Confirm before connecting.** The desk names the session it found and waits for a yes. A wrong match would add my words to the wrong conversation.
3. **Warn, never block.** If a session looks open on a laptop, the desk says so and asks. "Open" is often stale after I walk away.
4. **Stay on the line for short waits.** The desk offers a callback only when work will take several minutes.
5. **A spoken "go" counts.** It approves a proposed change exactly like a typed "go" at the keyboard.
6. **One call note per call.** After each call I get a text with what we discussed, what is still running and what is owed.

How an opening sounds, from the design brief

Desk	"New conversation, or pick one up?"
Me	"There's one about the office remodel."
Desk	"I found 'Office redesign with classical elements' from the 20th. That one?"
Me	"Yes." The desk puts me through, and the session picks up where I left it at the keyboard.

2. How a call works

1. I dial
Or I text "call me" and the phone rings.

2. Desk answers
"New conversation, or pick one up?"

3. Put through
It resumes a running session or opens a new one.

4. We talk
The session hears me and answers in its own voice.

5. I hang up
The work keeps going on the workstation.

6. Call back
Nero rings me when the work is done, or texts.

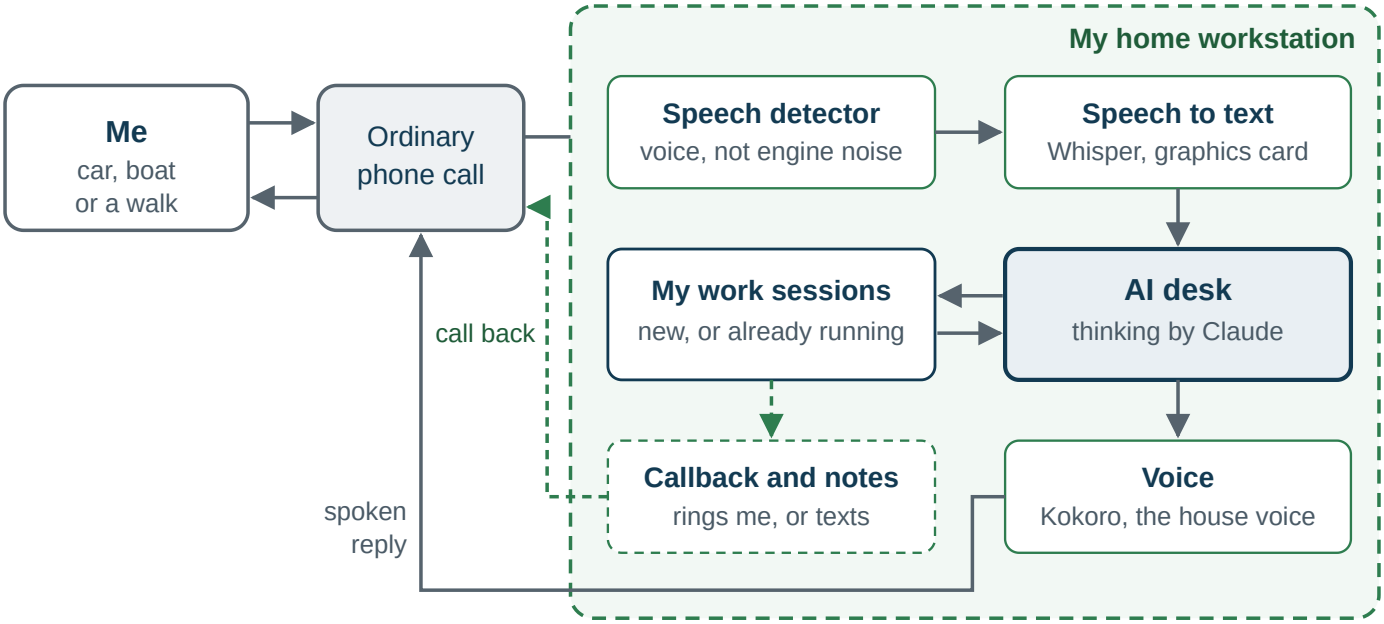


Figure 1. The path of one call. Green boxes run on my own hardware; the AI desk and the sessions think with Claude. Dashed green lines show the callback path after I hang up.

Job	Where it runs	What it does
Telling speech from noise	My workstation	Silero, a small open speech-detection model, separates my voice from wind, engine and water noise.
Speech to text	My workstation's graphics card	Whisper large-v3-turbo, an open speech-recognition model, chosen for muddy phone audio.
Thinking and work	Claude	Claude Opus through Claude Code on my existing subscription. Replies stream out, so speech starts on the first sentence.
Voice	My workstation	Kokoro, the open voice model the house already uses. Each session gets its own voice by topic, so a transfer sounds like a different staff member picked up.

How the work comes back

If I hang up mid-task, the work finishes. Nero rings me back if I asked; otherwise it texts the result. Any keyboard session can also ring me when it finishes, and the desk offers to put me through.

Premium voice, added November 2026

I later added an optional premium mode that sends the voice and hearing to a commercial voice service for a more natural sound. It is now the default on my calls. On any error the local models take over for that sentence.

3. Picking up live work

The hard part was not speech. It was letting a phone call join a conversation that is already open on a laptop without splitting it in two.

Every session open in my laptop app keeps an idle copy running on the workstation. If the phone started a second copy, two processes would write the same conversation and fork it. So the phone takes over. The call program closes the laptop's idle copy after my spoken yes, then resumes the session itself. Claude Code's own safety check will not let an agent close those processes, so the call program does it in code. In the first live test the laptop app reloaded the session with the phone turns at the bottom.

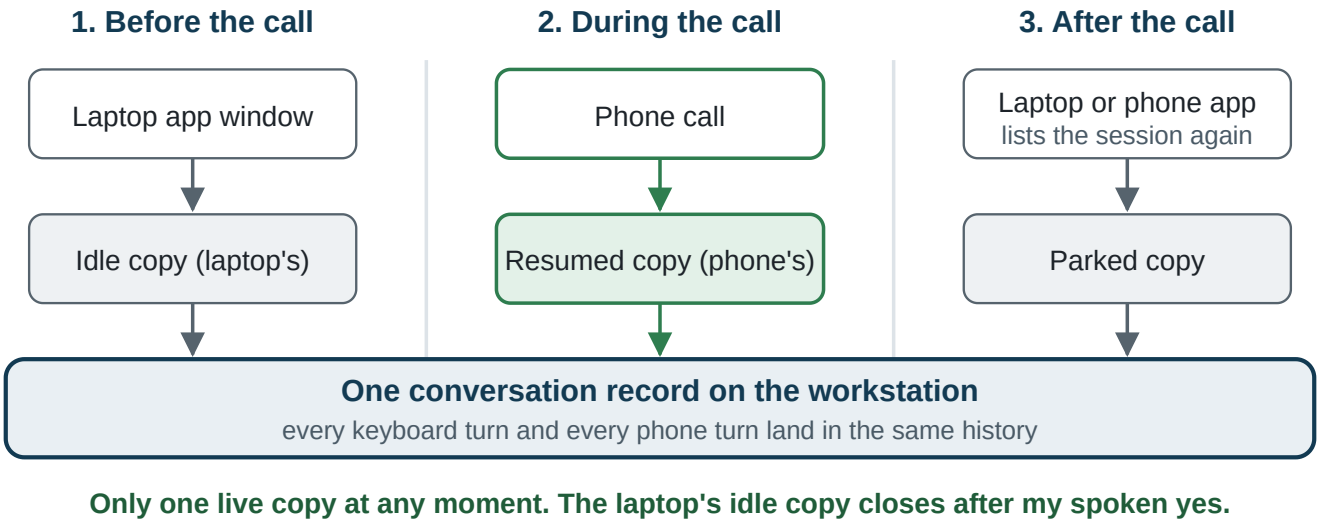


Figure 2. Takeover, not a second copy. After the call, every session the call touched reappears in my laptop and phone app lists, so I can carry on at the keyboard.

Switching is code, not a promise

In an early test the model said "switching you back" without running the switch. Now the call program itself listens for switching phrases - "take me back to the desk", "start something new", "transfer me to" another project - and acts on them. A model's sentence alone never moves a call.

Measured speed

Moment in a call	Measured
Desk reply once the call is under way	1-3 seconds
First reply of a call	3.5 seconds
Reply from a connected project session	3-6 seconds
Pause after I stop talking, house voice, three practice calls	2.9, 3.3 and 3.4 seconds
Digging through old session files (the desk says it is still with me)	15-30 seconds

Most of each pause is thinking time in Claude. Local hearing takes 0.08 seconds per sentence and the local voice 0.14 to 0.5 seconds, measured on the workstation in November 2026.

4. Lessons from real calls

Each fix below started in a call transcript, not a guess. In November 2026 I had an agent read all 99 call transcripts from the first three weeks; that one pass produced nine fixes.

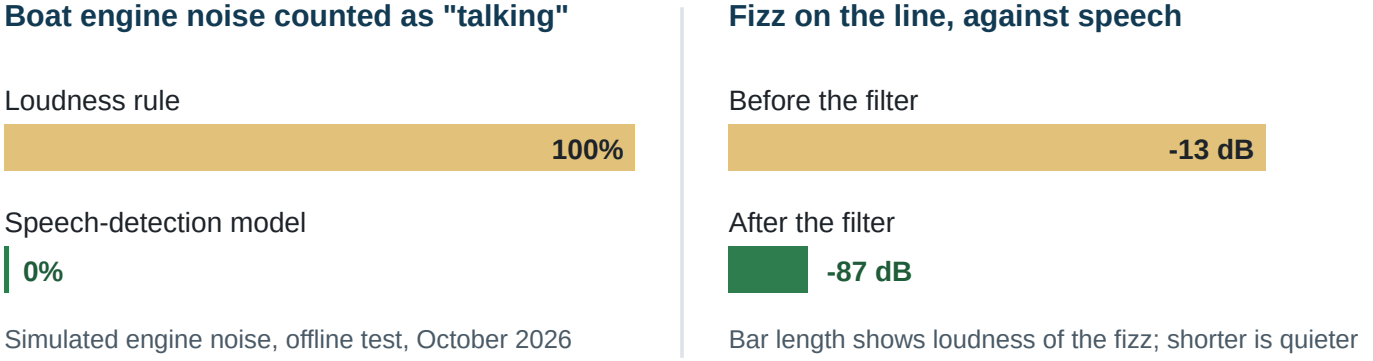


Figure 3. Two early fixes. Left, a loudness rule would never hear me stop talking over an engine; a small speech-detection model fixed it and still caught speech over the noise. Right, filtering the voice before converting it to phone quality removed audible fizz.

What went wrong	What I changed
I could not interrupt it	The program checked "still sending audio?" when it needed "still playing?", and it sends audio faster than the phone plays it. It now tracks playback, so I can cut in.
"Yeah" cut it off	Short acknowledgments like "okay" and "uh-huh" now neither stop a reply nor start a new one. I matched the cut-off threshold to a common open-source voice framework's default.
It talked past its own question	Words I said while it was still talking reached it afterward, as if they answered its new question. They now arrive marked as said earlier, and each reply asks one question, last, then stops.
Menus failed by voice	A list of session titles and "which one?" kept getting "yes, that one" back. The desk now offers one best guess as a yes-or-no.
A word list made hearing worse	Giving the speech recognizer a list of expected words made it invent those words from pure noise in 9 of 9 test clips. I rejected the change.
Nothing could end a call	"Hang up" or "end the call" now ends it. A plain "bye" does not, because people on a merged call say bye to each other.
"Text me when it's done" left me on hold	The desk now confirms, hangs up and sends the result by text.
Silence had no limit	Nero now asks "Still there?" after 45 seconds of quiet and says goodbye at 60. "Give me a sec" buys a quiet wait of up to 10 minutes.
Links read aloud	Written links and source lists no longer reach the voice; it says the name instead.

Checked against the industry, November 2026

I compared the line against four widely used voice-agent frameworks and adopted three of their practices. Only what I heard counts, so when I cut Nero off, it learns where I stopped listening. Acknowledgments are not interruptions. And each reply asks one question at most.

5. Rules and quality checks

1. **Same rules as my keyboard.** A session on a call runs under the same permission rules and guardrails as at my desk. It cannot do more by phone than I allow in person.
2. **Code owns the switches.** Switching sessions, closing a laptop copy and hanging up run in the call program, never on a model's word alone.
3. **Read back before acting.** The desk reads back anything costly to mishear - an amount, a date, a name, a file to delete - and waits for a yes.
4. **Independent cold review.** On launch night a second AI model reviewed the code without my reasoning in front of it. It found 18 issues; I confirmed 17 against the code and had them fixed the same night.
5. **Simulated calls before live ones.** A test harness plays scripted callers into a spare copy of the program - interrupting, talking over it, pausing, going silent, hanging up - and every case must pass before a change goes live.
6. **A health probe every 10 minutes** tests the whole public path and alerts me once after two failures in a row.

6. Results, status and what is next

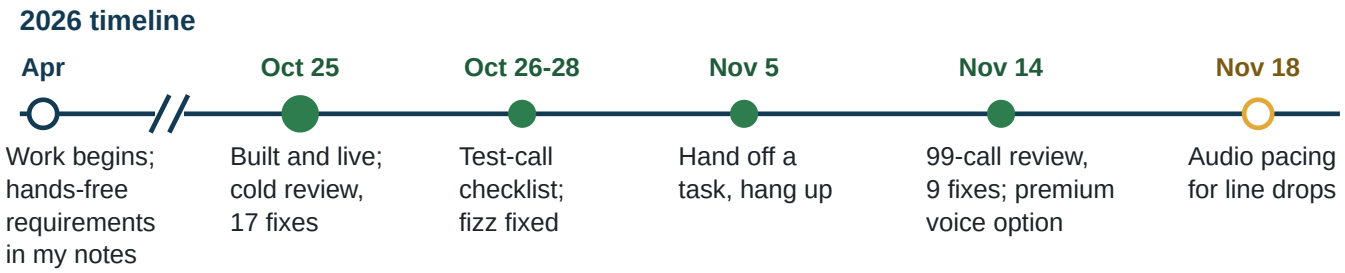


Figure 4. From requirements in April 2026 to live calls on October 25, then steady fixes driven by real calls.

Piece	Status	Note
Calls in and out, session takeover, new sessions, callbacks, call notes	LIVE	Since October 25, 2026
Hand a task to a session and hang up; it texts me when done	LIVE	Since November 5, 2026
Occasional dropped lines	OPEN	Cause not proven. I changed audio pacing on November 18, 2026.
Listen-only mode for conference calls	PROPOSED	Not built

Cost. The AI desk runs on my existing Claude subscription and local speech costs nothing per call; the phone number costs $1.15 a month. Six similar public projects already put Claude Code on a phone call, so I claim no first. None I found combined a real phone number, local speech and taking over sessions already running.

7. What it shows about how I work

I ran this the way I run a program. I set the requirements and the rules, ruled out options that failed a real test and let call transcripts decide what to fix next. A cold review and simulated calls stand between every change and my phone. That is the same discipline I bring to partnerships and AI adoption. Start from how people actually work, test with real use and fix what the record shows.