iRelay: Text Your Mac, Run an AI Agent

I created a GitHub repo from my phone while walking around the house. No terminal, no SSH, no laptop open. Just iMessage.

iRelay is a Swift daemon that listens for iMessages on your Mac, spawns Claude Code in your project directory, and sends the result back as a text. You message your Mac like you'd message a person, except it writes code.

Codeberg: codeberg.org/Mihaela/iRelay

Demo

:::tile embed url: https://www.youtube.com/embed/ehOY7BnPOf0 title: Demo video aspectRatio: 315/560 :::

How It Works

The flow is simple:

```
iPhone (iMessage) → Mac (iRelay daemon) → Claude Code → response → iMessage
```

You send something like `irelayy fix the failing test` from your phone. Your Mac picks it up, runs Claude Code against your project, and texts you back with what it did.

That's it. No web UI, no port forwarding, no VPN. Just iMessage as a transport layer.

Why the Double Y

I already run OpenClaw on the same Mac for iMessage automation. Two daemons listening to the same iMessage account would fight over every incoming message.

The fix: a prefix. iRelay claims any message starting with `irelayy` (double y). OpenClaw ignores those. Everything else goes to OpenClaw. Simple namespace partitioning over a shared channel.

What I Actually Did With It

First real test: I asked it to investigate my repos and suggest a business idea. It came back with MCPMe. Then I told it to create the repo, write the README, and push it. All from iMessage. The Mac did the work while I made coffee.

That's the use case — you're away from your desk but you want something done *now*. A quick fix, a repo setup, a refactor. Text it, walk away, get the result back on your phone.

Architecture

iRelay is designed around channels and providers:

- **Channels** handle message transport — iMessage is the one that works today. The architecture supports Telegram, Slack, Discord, Signal, Matrix, IRC, and WebChat, but those are scaffolded, not wired up yet.
- **Providers** handle LLM execution — Claude Code is the primary one. OpenAI, Ollama, and Gemini slots exist for future use.

Conversation history is preserved across messages, so context carries over. You can also save and restore sessions.

What's Next

iRelay pairs well with Cupertino — when Claude Code has offline access to Apple's documentation, the responses you get back over iMessage are significantly better. No hallucinated APIs, no deprecated patterns. I've been quietly working on a major Cupertino update that nearly doubles framework coverage, which will make this combination even stronger. More on that soon.

Built in a day. Used from the couch.