OpenClaw, Bob, and a Small Taste of the Future

March 8, 2026 / Gavin Jackson

openclaw ai automation telegram kimi productivity bob

I've tried a few AI assistants over the last year, and most of them have felt like tools I visit rather than systems I actually live with.

OpenClaw was the first one that felt different.

Not because it was perfect. It definitely isn't. But because it let me build something that felt personal, local, and oddly futuristic.

In my case, that system is called **Bob**.

Bob runs on a Mac that lives on the top shelf in my pantry. I talk to him from my phone through a Telegram bot. Behind the scenes, OpenClaw handles the orchestration, **Kimi K2.5** handles the model side, and **Brave Search API** gives Bob a way to look things up on the web.

That setup is not quite Jarvis from Iron Man, and Bob is not KITT from Knight Rider.

But it is absolutely a taste of that future.

What Makes OpenClaw Interesting

OpenClaw is not just another browser chat UI. It is an assistant framework that can sit inside your own environment, keep context, access local tools, and communicate through messaging channels.

That is what made it click for me.

Instead of opening a website, pasting context, and hoping the model remembers what I meant, I can message Bob directly and have him operate in an environment that already knows about my setup, my tools, and my preferred workflows.

That shift matters more than it sounds.

My Bob Architecture

At a high level, the setup looks like this:

```
Phone
 |
Telegram
 |
OpenClaw bot / gateway
 |
Bob on pantry Mac
 |
|- Kimi K2.5 API for model inference
|- Brave Search API for web research
|- local tools, files, scripts, and automations
```

The hardware side is gloriously unglamorous. Bob lives on a Mac sitting on the top shelf in my pantry, quietly doing his thing in the background.

The software side is more interesting:

- **Telegram** is the user interface
- **OpenClaw** is the runtime and orchestration layer
- **Kimi K2.5** is the model Bob talks to
- **Brave Search API** gives him current web access

That combination turned out to be a sweet spot for me.

Telegram means I can talk to Bob from anywhere without needing to VPN into home or sit at a terminal. OpenClaw gives Bob a real operating environment instead of just a chat box. Kimi K2.5 gives him the language model. Brave gives him a way to verify things and pull in current information when needed.

The Cost

This setup has also been pleasantly affordable.

I bought **US$50 of Kimi API credits**, which is enough to do real experimentation without feeling like every query needs a budgeting committee. I also pay **US$5 per month for the Brave API**, which is a pretty reasonable price for giving Bob the ability to search the web.

That is one of the more exciting parts of the whole exercise. A system that would have felt wildly out of reach a few years ago is now possible with a spare Mac, a messaging app, and a fairly small monthly spend.

Why This Feels Different

The magic is not just "AI in chat".

The magic is that Bob feels like he exists somewhere.

He has a home. He has tools. He has channels. He has a persistent identity. I do not think "I am going to use an LLM now". I think "I'll ask Bob".

That sounds silly until you experience it.

Once an assistant becomes reachable from your phone, can search when needed, can act in a local environment, and can carry some continuity between interactions, it stops feeling like a demo and starts feeling like the beginning of a real personal agent.

That is where the Jarvis and KITT comparisons start to make sense. Not because the implementation is anywhere near that polished, but because the interaction model starts to resemble those fictional assistants more than a normal app does.

What OpenClaw Does Well

There are a few things OpenClaw genuinely does very well.

1. It Bridges AI and Real Systems

This is the big one.

OpenClaw is valuable because it connects models to an actual runtime environment. It can work with local files, tools, scripts, and messaging channels in a way that feels much more agentic than a normal web chat.

2. Telegram Works Beautifully as a Front End

For my use case, Telegram is an excellent interface. It is fast, available everywhere, and much more natural for quick requests than opening a browser or remote desktop session.

OpenClaw's Telegram support is documented here: OpenClaw Telegram docs.

3. Search Makes the Bot More Useful

Adding Brave Search API made Bob much more practical. Without web access, assistants often sound confident but drift badly on current facts. With search, Bob can look things up, verify, and bring back something current when that matters.

4. It Feels Personal

This is hard to quantify, but important.

Bob is not a generic assistant in a generic interface. He is *my* assistant, running in *my* environment, wired into the channels and tools that make sense for me. That gives the whole system a different feel.

The Rough Edges

OpenClaw is impressive, but it is not friction-free.

Some of the problems I ran into seem to be common enough that they are reflected in the docs, issue tracker, or community conversations.

Cron Jobs Felt Flaky

I found the daily cron jobs a bit unreliable.

To be fair, cron support is very much a real feature in OpenClaw, and the docs cover it in detail: Cron jobs docs. The gateway configuration docs also expose specific cron controls like concurrency limits, run log retention, and session retention, which suggests the project is still actively hardening that part of the system: Gateway configuration.

My own experience was that scheduled jobs were not always as confidence-inspiring as I wanted for something intended to run unattended every day. That does not make the feature useless, but it does mean I would be cautious about depending on it for anything critical without extra monitoring.

Apple Mail Felt Limited

I found the Apple Mail integration fairly limited.

Part of that may simply be platform reality. Apple ecosystem integrations can be awkward to automate cleanly and consistently, especially compared with simpler messaging channels. OpenClaw's strongest, best-documented channels seem to be things like Telegram and webhook-based flows rather than deep Apple Mail workflows.

That does not mean Apple-side integrations are impossible, just that they felt less mature and less central to the platform than some of the other channels.

Gmail Was Surprisingly Hard

Gmail was the opposite problem: clearly possible, but much harder to set up than it probably should be.

The OpenClaw docs do document Gmail hook support, but it is definitely not "click here and you're done". It involves webhook-style configuration, routing, and security considerations: Hooks and Gmail mapping in gateway config.

So when I say Gmail felt almost impossible to set up, I do not mean OpenClaw has no answer for it. I mean the answer is involved enough that it creates real setup friction, especially if you just want to get moving quickly.

Context Window Problems Are Real

The most obvious behavioural issue I saw was what I can only describe as **serious amnesia** once the context window got close to 100%.

This is not unique to OpenClaw, of course. It is a general LLM problem. But agent frameworks feel it more sharply because the whole point is continuity over time.

OpenClaw's own session and memory docs acknowledge that memory is bounded and has to be managed rather than assumed: Sessions and memory. In practice, once a session gets very full, the assistant can become noticeably worse at remembering key details, holding plans together, or preserving the exact thread of what it was doing.

So yes, Bob occasionally becomes forgetful. In a weird way, that makes him feel even more like an early-generation sci-fi assistant: amazing when it works, slightly baffling when it drops something obvious on the floor.

The Broader Pattern

None of those downsides changed my view of the project.

If anything, they made it clearer what OpenClaw really is: not a polished mass-market assistant product, but an unusually compelling preview of where things are heading.

That is why I keep coming back to the Jarvis and KITT comparison.

We are not at the stage where a personal AI assistant is effortless, invisible, and always reliable. We are at the stage where, if you are willing to do a bit of setup and tolerate some rough edges, you can build something that already feels meaningfully different from the software experiences we have had up until now.

That is exciting.

Final Thoughts

Bob is not perfect.

He lives on a pantry shelf. He talks to me through Telegram. He runs on API credits and a search subscription. His cron jobs can be temperamental. His email integrations are uneven. And when the context gets too full, he can forget things in a way that would be hilarious if it were not occasionally inconvenient.

But he is also one of the most convincing glimpses of the future I have had on my own hardware.

That, to me, is what makes OpenClaw worth paying attention to.

It lets you move beyond "AI as a website" and toward **AI as a persistent, personal, reachable system**.

Today that system is called Bob, and he lives in my pantry.

A few years from now, I suspect setups like this will feel completely normal.

Relevant Links

- OpenClaw GitHub
- OpenClaw Telegram docs
- OpenClaw gateway configuration
- OpenClaw sessions and memory docs