Week 2: A Raspberry Pi From 2015 Joined the Swarm

February 10, 2026 · by Mark Bartlett

Download this post as PDF

Last week I had four nodes on a wired LAN routing chat to a 32B model. This week I have five nodes across two subnets, persistent memory, a document library with RAG, agentic tool routing, a one-line installer, and a laptop on WiFi using a GPU across the house. Oh, and a Raspberry Pi 2 from 2015 is part of the swarm.

Here's what happened.

The Raspberry Pi Moment

I had a Pi 2 sitting in a drawer. 921MB of RAM. ARMv7 processor from 2015. It can't run inference — it can barely run Python. But it can search the web and process files.

I installed mycoSwarm, started the daemon, and within seconds it appeared in the swarm. The other nodes saw it, the orchestrator added it to the rotation, and it started handling web search tasks alongside the Lenovo M710Qs.

A $35 board from a decade ago, contributing to a distributed AI system. That's when the manifesto stopped being abstract:

> "If a student in Lagos with two old laptops can't participate, the framework has failed."

The Pi proves the floor is low enough. Anything with a network connection and Python can join.

Tests First, Features Second

I started the week with a codebase that had grown fast — five new handlers, a plugin system, streaming inference — and zero tests. One bad refactor could break everything silently.

So before touching features, I wrote 38 tests covering hardware detection, capability classification, orchestrator routing, the API layer, worker handlers, and the plugin system. All mocked — no Ollama, no network, no GPU needed. They run in 1.4 seconds.

The rule: if it's not tested, it's not built. Every feature that followed came with tests. By the end of the week: 94 tests, all passing, all offline.

Single-Node Mode

The biggest barrier to adoption wasn't the swarm — it was requiring a swarm. If someone just wants local ChatGPT on their machine, they shouldn't need to understand daemons and mDNS.

Now they don't:

```
pip install mycoswarm
mycoswarm chat
```

Two commands. If no daemon is running, it talks directly to Ollama. No config, no setup, no swarm knowledge required. When they're ready to add a second machine, they start the daemon and the swarm forms automatically.

The path from "curious" to "chatting" is about 60 seconds.

Persistent Memory

Chat sessions are ephemeral by default — restart and the model forgets everything. That's useless for real work.

mycoSwarm now has two memory layers:

Facts — things you explicitly tell it to remember:

```
/remember I teach Tai Chi on Thursdays and Sundays at 11am
/remember I keep bees using Layens hives
```

These persist in a JSON file and get injected into the system prompt on every chat session.

Session summaries — when you `/quit`, the model summarizes the conversation and appends it to a log. Next session, the last 10 summaries are loaded as context. The model knows what you discussed yesterday without carrying the full conversation history.

I told it I keep bees. Next session, I asked “where can I buy Layens hive frames?” It searched the web and came back with Miller Bee Supply, Mad Cat Farm, and Natural Beekeeping — with assembled vs. unassembled options. It knew the context without being told.

Document Library with RAG

Drop files into `~/mycoswarm-docs/` and run:

```
mycoswarm library ingest
```

PDFs, markdown, text, HTML, CSV, JSON — everything gets chunked, embedded via Ollama, and stored in ChromaDB. Then ask questions:

```
mycoswarm rag "what features are planned but not built yet?"
```

The system searches across all indexed documents, finds the relevant chunks, and synthesizes an answer with citations. I indexed CLAUDE.md, PLAN.md, and README.md — it correctly identified which features were built and which were still planned, pulling from all three files.

Local Perplexity. No API keys. No data leaving the network. Your documents stay on your hardware.

Honest AI (The Hallucination Fix)

I asked the model “what’s the weather in Berkeley?” and it confidently told me it was 72°F and sunny. It was 54°F and cloudy.

Models hallucinate. They’re trained to be helpful, so they fabricate rather than admit ignorance. The fix is three layers:

Layer 1 (done): The system prompt now explicitly states what the model can and can’t do. “You do NOT have internet access during chat. Never fabricate current data.”

After the fix:

```
🍄> what is the weather in Berkeley today?
I don't have access to real-time information. You can try:
mycoswarm research weather in Berkeley for web-sourced answers.
```

Honest. Useful. Trustworthy.

Layer 2 (done): Agentic tool routing. The model now automatically detects when it needs tools and uses them.

Agentic Chat

This is the big one. Before every response, the model classifies the query:

- **"answer"** — can answer from training knowledge and memory
- **"web_search"** — needs current information
- **"rag"** — answerable from the user's documents
- **"web_and_rag"** — check both

Then it acts, showing you what it's doing:

```
🍄> what's the weather in Berkeley?
   🔍 Searching the web... 5 results
It's 54°F and cloudy in Berkeley right now...

🍄> what does CLAUDE.md say about the orchestrator?
   📚 Checking your documents... 5 excerpts
The orchestrator routes tasks based on capability and load...

🍄> what is Wu Wei?
Wu Wei is the Taoist concept of effortless action...
```

Three different queries. Three different strategies. All transparent — the user sees every tool call. No magic, no mystery.

The WiFi Laptop Test

The real test wasn't the wired LAN nodes. It was this: can someone on a laptop, connected by WiFi, use a GPU on a different subnet?

I pulled out a Lenovo ThinkPad Yoga 260 — a 2015 ultrabook with 8GB RAM, no GPU. Installed mycoSwarm from PyPI. The installer detected the hardware, pulled gemma3:4b (sized for 8GB), and got everything running.

First test, no swarm — just the laptop's CPU: **178 seconds** for a response. Painfully slow but it worked.

Then I started the daemon. The laptop discovered Miu's RTX 3090 across the WiFi-to-ethernet bridge. I asked the same question: **19.5 seconds.** The laptop sent the request over WiFi, Miu's 3090 ran gemma3:27b, and the response streamed back.

A 9x speedup. A $200 laptop borrowing a $850 GPU's brain across the house.

That's the pitch: the weakest machine in the swarm gets access to the strongest model.

Bugs Found (And Fixed)

Real-world testing on real hardware found three bugs that no unit test would catch:

1. **Daemon bound to one interface.** The API only listened on the wired IP, so WiFi devices couldn't connect. Fix: bind to 0.0.0.0.
2. **mDNS advertised one address.** Miu has both ethernet (192.168.50.1) and WiFi (192.168.1.29). It only advertised the ethernet address. WiFi peers couldn't reach it. Fix: advertise all non-loopback IPs, and when discovering a peer with multiple addresses, TCP-probe each one to find the reachable one.
3. **Model mismatch on remote inference.** The laptop asked Miu to run gemma3:4b — a model Miu doesn't have. Fix: the orchestrator now checks the remote peer's available models and swaps to the best one. The laptop requests 4b, Miu runs 27b.

These are the bugs you only find by plugging in actual hardware on actual networks. The test suite catches logic errors. Real hardware catches integration errors.

The Stack Right Now

Node	Hardware	Cost	Role
Miu	RTX 3090, 64GB RAM	~$850	GPU inference — gemma3:27b

Node	Hardware	Cost	Role
naru	M710Q, 8GB RAM	$50	Web search, file processing
uncho	M710Q, 8GB RAM	$50	Web search, coordination
boa	M710Q, 8GB RAM	$50	Web search, code execution
raspberrypi	Pi 2, 1GB RAM	$35	Search, lightweight tasks
Reyoko	ThinkPad Yoga 260, 8GB RAM	~$200	WiFi client, uses Miu's GPU

Total hardware: ~$1,235. Monthly cost: $0.

94 tests. macOS compatible. One-line installer. On PyPI. Landing page at mycoswarm.org.

What's Next

- **PyPI 0.1.1** — all the cross-subnet fixes
- **Dashboard** — web UI for swarm topology and node health
- **Agentic planner** — multi-step task execution across nodes
- **mTLS** — encrypted node-to-node communication
- **The real question:** demo video. Showing is better than telling.

Week 1 was "can this work?" Week 2 was "can anyone use this?" The laptop test proved the answer is yes. Now it's about polish, documentation, and getting it in front of people who care about local AI.

---

mycoSwarm is open source. MIT license. Two commands to start:

```
pip install mycoswarm
mycoswarm chat
```

Repo: github.com/msb-msb/mycoSwarm Site: mycoswarm.org

Every node counts.

Source: https://insiderllm.com/blog/week-2-raspberry-pi-joins-swarm/

Free guides for running AI locally