OmNet: A Homestead Network Where Code Decides

Compiled sentinels on one small computer, a radio mesh, and a model asked only about things the code has not seen before

Charles E. Forsyth III
Research Computing
University of California, Riverside
Riverside, California, USA
charles.forsyth@ucr.edu

Abstract

OmNet is the network that watches my homestead in north-central Pennsylvania: weather, frost, the chicken coop, cameras, power and the network itself. Since September 2026 it has run on one low-cost single-board computer I call Core. Its founding rule is that nothing running around the clock calls an AI model. The routine work is done by 24 sentinels in one compiled Rust program.

In late October I bent that rule on purpose. Core writes about 19,500 journal lines and 1,250 broker messages a day, and no fixed pattern list covers them. A new service, logwall, reduces each line to a shape and asks a hosted model about a shape only the first time it appears. Code, not the model, decides what becomes an alert. It has run in shadow mode for days, not months, so I have no false alert rate yet.

CCS Concepts

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Software and its engineering** → *Maintaining software*; • **Computing methodologies** → *Artificial intelligence.*

Keywords

edge computing, sensor networks, LoRa, Meshtastic, log parsing, large language models, guardrails, gray failure

Author note

The work was directed by me and carried out with AI agents under written briefs, independent checks and explicit approval steps. Figures come from the OmNet architecture specification, the logwall build, heal and camera plans, and the incident notes in my personal vault. This note was drafted with AI assistance from those records and reviewed by me. No representative values are used: every count, time and measurement is as recorded. Hardware is described by class rather than model, and no device names, addresses or locations finer than the region are given.

Cite this note. C. E. Forsyth III. 2026. OmNet: A Homestead Network Where Code Decides. Field Notes.
doi:10.5281/zenodo.23752284

1 Context

A false frost alert costs little. But everything here runs on one small computer on a rural internet link, plus a few radios and some cameras, and it has to keep working when the internet drops. What goes wrong at home is quiet: a sentinel reports success for hours while writing nothing, or a model bill creeps up. Both risks shaped the design, and the first has already happened. Cloud operators know the first kind as gray failure, where a component looks healthy to its own checks while the work it exists to do has stopped [1], and hardware that degrades without failing outright is common enough in large production systems to have its own study [2].

This is a personal project, not a Research Computing service. It belongs in Field Notes because the same questions come up on research clusters: what should run as plain code, where a model earns its cost, and how you know a quiet system is healthy rather than dead. The same AI agent that drafts my work email built most of it, under the rules in the companion note, *A Seneschal for the Research Computing Desk* [3].

2 Related Work

Unattended sensing in rural places has a long research record. The Great Duck Island deployment set out the requirements most homestead builders meet again: small nodes, scarce power, a gateway with an unreliable uplink, and data that must survive when the link is down [4]. Edge computing gives the general argument for doing the routine work next to the sensors rather than in a data center, mainly for latency, bandwidth and keeping working offline [5]. FarmBeats is the closest farm-scale system: an IoT platform for agriculture whose design plans for weather-related power and internet outages, run for six months on two US farms [6]. OmNet follows all three at a much smaller scale. Its new element is a policy about when a remote model may be consulted at all.

The radio side rests on LoRa. Measurements of LoRaWAN show how little capacity a long-range, low-rate channel has once many devices share it and duty-cycle limits apply [7]. Meshtastic uses the same modulation in a different, mesh-style network, and its documentation states that each step up in spreading factor doubles the airtime [8]. The same project documents what its encryption does: one pre-shared key per channel, readable by anyone who holds it, with per-node keys only for direct messages [9]. Both points set limits on Ask Core below.

Reducing log lines to templates is a mature research area. Drain parses streaming logs with a fixed-depth tree and masks variable tokens to recover templates online [10]; Drain3 is a maintained open-source implementation of it [11]. Benchmarks across many parsers and log sources [12] and a survey of automated log analysis for reliability engineering [13] place template extraction as the usual first step before anomaly detection. logwall's masking is simpler than Drain's tree, and it uses the template as a cache key

for a model's verdict instead of feeding a trained detector. Language models have also been prompted to parse logs and flag anomalies directly, without in-domain training and with a written explanation for each answer [14]; logwall asks a similar question but only once per template.

Two lines of LLM work bear on the cost controls. FrugalGPT describes ways to cut what applications spend on large models, including caching answers and calling a cheaper model first [15]. GPTCache stores model answers so that repeated or similar queries are served locally [16]. logwall's cache is the exact-match case: one verdict per shape. On safety, NeMo Guardrails puts programmable rules around a model's inputs and outputs [17], and indirect prompt injection shows that text an outside party can write, once it reaches a model, can change what the model does [18]. The log text logwall sends is partly of that kind.

Machine learning in production carries hidden dependencies and feedback loops that let a system degrade without any visible error [19]; that risk is why logwall records verdicts in shadow mode before it may act. The NIST AI Risk Management Framework asks for measured performance and human oversight proportionate to impact [20], and NIST's zero trust architecture treats every network location as untrusted by default [21]. OmNet's trust in its home network is a deliberate departure from the second, made for one household. The closest working systems are Home Assistant [22] and Frigate [23], compared below.

3 The Core and the Sentinels

Core is an older single-board computer with about 1 GB of memory and an SD card for storage. It runs the message broker, the telemetry database, a small web dashboard and the radio gateway. My laptop is the build machine and the vault holds the design records, but if the two disagree, Core is right and the record gets corrected. The architecture specification was last re-read against the live Core on 9 October 2026. Figure 1 shows the layout.

The founding principle is "zero-LLM" for anything that runs around the clock, where LLM means large language model. Network checks, weather math, sun and moon times, frost windows and camera schedules run as compiled Rust and Go programs, with some older Python. They cost nothing per run and need no internet. The main binary, `omnet-sentinel`, is one Rust program of about 7,000 lines with 24 sentinels as subcommands, cross-compiled on the laptop for Core's processor. A Go program, `omnet-mesh-logger`, pulls weather for 22 nearby points in one request every 30 minutes. The Go command-line client `omnet` is how I and the agent read the state. No model polls anything.

The specification also carries a rule I adopted on 24 September after finding fake numbers: *measured data only*. Every value shown is a sensor reading, a stored value from an outside service, or a calculation on those. A missing reading is `null` and shows as two dashes, never a default. A failed measurement is recorded as failed with its error, not as zero.

Camera vision without a model. `omnet-vision`, about 2,400 lines of Rust on a separate machine, receives each new motion clip, checks its hash and judges it with fixed rules: background subtraction at 4 frames per second, a coherent track that must last 5 frames and move a set distance, a blur test for a fogged lens, and sound

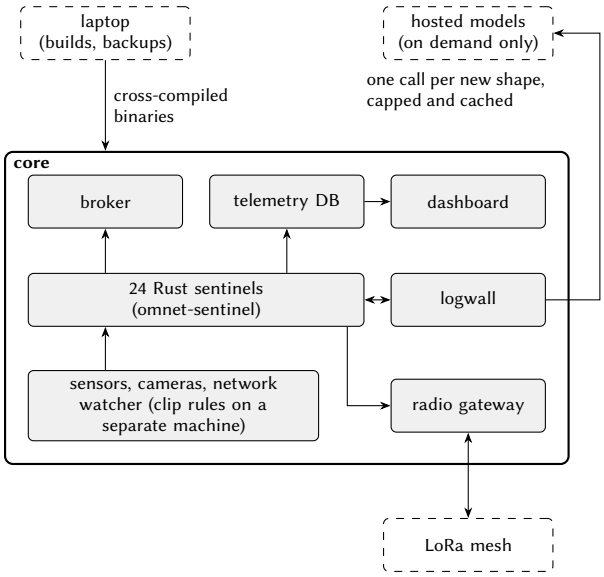


Figure 1: OmNet in outline. Everything inside the box runs as plain code on Core. Models sit outside it and are called only by code, only for new events, under the limits in Table 1.

measured against the clip's own noise floor. Checked by eye on 16 clips on 23 September, it marked chickens and a cat as real and filtered insects in infrared, cloud shadow and a fogged lens. The rule set it replaced had found 0 of 3 real events in that sample. It then worked through a backlog of 739 clips with no failures. The samples are small, so treat this as a sanity check.

Known devices. A watcher keeps a table of every device on the home network and labels each one known, transient or unknown against a list. On 9 October every device on it was known. An unknown device raises a private alert over the radio mesh.

Trust inside the house. The home network trusts the private network by design, a deliberate, written decision that keeps the internal plumbing simple. That choice rests on the network having no open inbound ports, which a sentinel checks, and on the unknown device alert. It suits one household where every device is mine. I would not make it on campus, and I revisit it when the device list changes.

Storage wear. Core keeps its database on an SD card, and SD cards wear out under constant small writes. Three things limit the damage. A maintenance sentinel prunes each table to a set retention window every night at 04:15, from 14 days for raw radio packets to 90 days for camera audits, and the raw broker log keeps 7 days. logwall keeps its own database, batches its writes and uses write-ahead logging [24]. And every night the laptop takes an online snapshot of the database with an integrity check and copies it, with Core's code and configuration, to two off-site stores; a restore was tested on 23 September. The nightly cleanup itself rewrites the database file, so it adds wear of its own. The database was about 95 MB on 9 October. I have not measured the card's remaining life.

4 The Radio Mesh

A Meshtastic radio hangs off Core by USB. Meshtastic is open-source firmware that turns small LoRa radios into a text mesh; LoRa is a long-range, low-power radio method that works with no internet. Core speaks the radio's serial protocol directly and stores real per-packet signal values. It sends debounced private alerts on our own channel and a public weather beacon on the open channel.

Since 24 October Core can take questions from radios on our private channel, so far in a dry mode that records replies instead of transmitting them. Ten commands (`status`, `coop`, `wx`, `power` and others) are answered by plain code with no internet. Anything else goes to a small hosted model with a snapshot of Core's own data and a 180-character limit. Each radio gets one open question per 30 seconds, and model calls count against logwall's daily cap.

What the channel does and does not protect. The private channel is encrypted with a key that every radio on it shares, as the Meshtastic encryption documentation describes [9]. That keeps outsiders from reading it. It does not prove who sent a given message: anyone who holds the key, including whoever finds a lost radio, can send as any node, and channel messages carry no replay protection. Newer firmware offers per-node keys for direct messages; Ask Core does not rely on them today. The software has an optional list of allowed radios, and it is not switched on yet. For a weather question that is fine. It is the main reason Ask Core will stay limited to read-only answers.

Airtime. LoRa is slow on purpose. On long-range settings a single packet can take on the order of a second to send, and each step up in spreading factor doubles the airtime, per the Meshtastic radio settings [8]. Every alert, beacon and answer shares that channel with other people's radios. Ask Core splits a reply into numbered parts of up to 200 bytes and paces them 1.5 s apart, and alerts are debounced. The dashboard shows the gateway's channel use, but I have not yet recorded it over time, so I cannot say how close to busy the channel runs. That number comes before Ask Core leaves dry mode.

5 Where a Model Is Allowed

Fixed rules miss events nobody wrote a rule for. So on 24 and 25 October the agent built `omnet-logwall`, a Rust service that reduces every log line and broker message to a *shape*. Numbers, addresses, paths and quoted strings are masked, leaving a template, much as log-template miners like Drain3 do [11]. The first time a shape appears it gets one call to a small hosted model, which answers six questions at once in a JSON (JavaScript Object Notation) object: is it an error, is it security-relevant, does a person need to look, is it noise, which subsystem, how severe. The verdict is cached. Known shapes cost nothing.

An alert fires only if the model's "needs a person" score is at least 0.7, or security is at least 0.7 at medium severity or above, or severity is high. Table 1 lists the guard rails, all from the build plan I approved on 24 October.

A model call from Core took 0.77 to 1.16 s in testing. The same pattern gives a second opinion on camera clips: when the rules say a clip is real, Core sends three still frames to a vision model and asks what the subject is. Code, not the model, decides whether that subject is a chicken predator, because in testing on 24 October one model rated a cat 0.8 to 0.9 on its own "predator" score. Several models called a black cat a raccoon; the one chosen got person, person and cat right.

Table 1: Guard rails on logwall's model calls.

Guard	Rule
Shadow mode	Default: verdicts recorded, nothing broadcast
Spending cap	300 calls a day, 30 a minute; more than 50 new shapes in 5 minutes trips a breaker and queues the rest
Human overrule	My verdict on a shape wins and is never re-judged
Double check	A shape flagged at 0.7 or above is asked twice; disagreement goes once to a larger model
Scrubbing	Tokens, keys, passwords and long encoded strings are removed before anything leaves Core; large binary blobs are never sent
Offline safe	If the model service is down, shapes wait as pending; Core never blocks on the cloud

The design has two known holes. A verdict is cached per shape, and masking strips out the very fields that often carry security news, such as a new address. An attack that fits a shape already judged harmless produces no new call and no alert. And the log text sent to the model is partly written by outside parties, such as host names and user agents, so a crafted line could try to talk the classifier out of flagging it. The double check and my overrule help with the second. Nothing yet addresses the first.

5.1 Healing Without Letting the Model Act

Core already had a plain Python healer that runs every 30 minutes, checks a dozen subsystems and restarts failed services, confirming each restart worked. On 24 October I approved a plan for a faster layer inside logwall that sees failures within seconds in the journal and groups repeated symptoms into one incident.

The model never runs a command. The plan lists 20 known failures, each with a code-defined fix, a check before acting, a check after, a cooldown and at most 2 attempts in 6 h. For an incident the model may pick only one fix from the entries whose trigger matched, or none, and code checks that the pick is on that list. Two failed attempts move the incident to "needs your hands," which goes to the dashboard and to my vault's daily note. Brakes apply across all fixes: at most 6 actions an hour, and no healing at all while the computer reports low supply voltage, because restarts during a brownout make things worse. Like the rest of logwall it runs in shadow mode, recording what it would have done, and I promote fixes one at a time.

Some things it is not allowed to heal: the power supply, a radio that has dropped off USB, a battery, or a sensor at the edge of range. Those become a plain note with the manual step.

6 Why Not Home Assistant

The obvious question from anyone who runs a smart home is why this is not Home Assistant [22] with Frigate [23] for the cameras. Home Assistant automations are, in effect, the same thing as my sentinels, with a large community and hundreds of integrations behind them. Frigate does local object detection on camera streams with an inexpensive accelerator and would likely beat my clip rules on accuracy.

I did not run a side-by-side test, so I will not claim OmNet is better. The differences I can state are these. OmNet grew one sentinel at a time from scripts, and the whole runtime fits in small static binaries on a computer with about 1 GB of memory. Its records are plain files and a SQLite database the agent can read and audit directly. The parts I care most about here, the measured-data-only rule, the radio questions with no internet and logwall's once-per-shape model calls, are things I would have had to build as add-ons either way. If I were starting today with no code, I would start with Home Assistant and add those pieces to it.

7 What Went Wrong

Silent success. On 22 September a sentinel that stores position fixes from a radio stopped saving them and kept reporting success for hours. It wrote while its own read cursor held the table, every insert failed with "database is locked", and the error was thrown away. A systematic audit the next day found worse: the 06:30 morning briefing had shown a hard-coded growing-degree-day value of 1,845.0 every day because of a wrong column name behind a bare exception handler. The real value was 1,825.5. That is where the measured-data-only rule came from.

Hardware the code cannot heal. On 27 September a low-voltage event knocked the radio gateway off USB. It came back under a different device name, and the daemon, hard-coded to the old name, waited 14 h while the service manager reported it healthy. It now finds the radio by serial number. In the 7 days to 24 October, Core logged 281 under-voltage events. The new power supply is still on the buy list.

A wrong label repeated with confidence. The audio classifier marked 4 clips as coyote calls. The vision second opinion found a person, chickens twice and a fogged lens. The cameras trigger on motion, not sound, so the label was the classifier misreading ordinary audio. Those shapes are now marked as noise by hand.

Records drift from the live system. On 9 October one sentinel still carried old configured targets that had been corrected in the dashboard a week before, and Core ran an older build of the command-line client than the laptop. The specification now records both.

The zero-LLM rule bent. logwall and the camera second opinion mean a model is now called from Core, automatically, when something new happens. That is a real change to the founding principle, and the specification text has not caught up.

One sensor is dark. On 28 October the "needs your hands" list held one item: a small Wi-Fi sensor had stopped sending its heartbeat on 24 October. Software flagged it. It still needs me to walk out and power-cycle it.

8 Lessons

(1) Keep models out of loops. A compiled sentinel costs nothing per run and keeps working when the internet drops.
(2) When code does call a model, let the model answer and let code decide, with a cap, a cache and a human overrule. In my tests one model called a cat a likely predator.
(3) Treat a missing number as missing. The made-up 1,845.0 sat in my morning briefing until an audit found it.
(4) Check a service's output, not only its status. The radio gateway was "active" for 14 h while it read nothing.
(5) Know what your radio encryption proves. A shared channel key keeps outsiders from reading; it does not tell you who sent a message.

9 What's Next

- Replace the Core power supply, then see how many open hardware faults disappear with it.
- Run logwall in shadow mode through a full season, from first frost to spring, and hand-label every alert it would have sent. That false-alert rate, with the share of proposed fixes that a check afterwards confirms worked, is the measurement that would show whether letting code decide holds up. Until then the main claim of this note is a design, not a result.
- Record channel use on the mesh over a few weeks, and turn on the allowed-radio list, before Ask Core transmits.
- Update the OmNet specification so it describes the capped, cached model calls honestly.

Sources

The OmNet repositories are private because they describe a home. Private records used, by name: the repositories `omnet-sentinel`, `omnet-cli`, `omnet-mesh-logger`, `omnet-vision` and `omnet-logwall`, including the logwall build plan, heal plan, Ask Core and camera second opinion documents; the OmNet master architecture specification (audited 9 October 2026), the Core hardware record, the silent failure audit of 23 September 2026 and the "needs your hands" note of 28 October 2026, all in my personal vault. External literature and documentation are listed under References.

References

[1] Peng Huang, Chuanxiong Guo, Lidong Zhou, Jacob R. Lorch, Yingnong Dang, Murali Chintalapati, and Randolph Yao. 2017. Gray Failure: The Achilles' Heel of Cloud-Scale Systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS '17)*. ACM, New York, NY, USA, 150–155. doi:10.1145/3102980.3103005

[2] Haryadi S. Gunawi, Riza O. Suminto, Russell Sears, Casey Golliher, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, Deepthi Srinivasan, Biswaranjan Panda, Andrew Baptist, Gary Grider, Parks M. Fields, Kevin Harms, Robert B. Ross, Andree Jacobson, Robert Ricci, Kirk Webb, Peter Alvaro, H. Birali Runesha, Mingzhe Hao, and Huaicheng Li. 2018. Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems. *ACM Transactions on Storage* 14, 3 (2018), 1–26. doi:10.1145/3242086

[3] Charles E. Forsyth III. 2026. A Seneschal for the Research Computing Desk. Field Note. https://charles-forsyth.github.io/papers/2026-10-29-a-seneschal-for-the-research-computing-desk.html

[4] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. 2002. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*. ACM, New York, NY, USA, 88–97. doi:10.1145/570738.570751

[5] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646. doi:10.1109/JIOT.2016.2579198

[6] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Ashish Kapoor, Sudipta N. Sinha, Madhusudhan Sudarshan, and Sean Stratman. 2017. FarmBeats: An IoT Platform for Data-Driven Agriculture. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*. USENIX Association, Boston, MA, USA, 515–529. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/vasisht

[7] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. 2017. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine* 55, 9 (2017), 34–40. doi:10.1109/MCOM.2017.1600613

[8] Meshtastic. 2026. Radio Settings. Meshtastic documentation. https://meshtastic.org/docs/overview/radio-settings/ Accessed 28 October 2026.

[9] Meshtastic. 2026. Encryption. Meshtastic documentation. https://meshtastic.org/docs/overview/encryption/ Accessed 28 October 2026.

[10] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 33–40. doi:10.1109/ICWS.2017.13

[11] LogPAI. 2026. Drain3: A Robust Streaming Log Template Miner. GitHub repository. https://github.com/logpai/Drain3 Accessed 28 October 2026.

[12] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. 2019. Tools and Benchmarks for Automated Log Parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 121–130. doi:10.1109/ICSE-SEIP.2019.00021

[13] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R. Lyu. 2021. A Survey on Automated Log Analysis for Reliability Engineering. *Comput. Surveys* 54, 6 (2021), 1–37. doi:10.1145/3460345

[14] Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. 2024. LogPrompt: Prompt Engineering Towards Zero-Shot and Interpretable Log Analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24)*. ACM, New York, NY, USA, 364–365. doi:10.1145/3639478.3643108

[15] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. arXiv:2305.05176. https://arxiv.org/abs/2305.05176

[16] Fu Bang. 2023. GPTCache: An Open-Source Semantic Cache for LLM Applications Enabling Faster Answers and Cost Savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. Association for Computational Linguistics, 212–218. doi:10.18653/v1/2023.nlposs-1.24

[17] Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 431–445. doi:10.18653/v1/2023.emnlp-demo.40

[18] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23)*. ACM, New York, NY, USA, 79–90. doi:10.1145/3605764.3623985

[19] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*. https://proceedings.neurips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

[20] Elham Tabassi. 2023. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. National Institute of Standards and Technology. doi:10.6028/NIST.AI.100-1

[21] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. 2020. *Zero Trust Architecture*. NIST Special Publication 800-207. National Institute of Standards and Technology. doi:10.6028/NIST.SP.800-207

[22] Home Assistant. 2026. Home Assistant: Open Source Home Automation. Project website. https://www.home-assistant.io/ Accessed 28 October 2026.

[23] Frigate. 2026. Frigate NVR Documentation. Project documentation. https://docs.frigate.video/ Accessed 28 October 2026.

[24] SQLite. 2026. Write-Ahead Logging. SQLite documentation. https://www.sqlite.org/wal.html Accessed 28 October 2026.