PURETENSOR
INTELLIGENCE DISTILLED

Junction Temperature Is the Wrong Fan-Control Input

A sensor-selection study on a Zen 5 workstation

Paper: PT-TN-2026-009 v1.0

Author: PureTensor Ltd

Date: 8 April 2026

Status: Published

Abstract

Junction temperature is a safety signal, not a control input. On a 32-core Zen 5 workstation, we traced persistent fan hunting to a single configuration error: the Linux fan controller was sourcing its temperature from the CPU die hotspot reading (Tctl), which swings 20–30 °C in seconds as threads migrate between chiplets. Switching to a socket-level thermistor eliminated the oscillation. The strongest evidence is a pair of simultaneous measurements on the same idle machine: Tctl varied 59–74 °C while the socket thermistor held flat at 38.5 °C. These are not noisy versions of each other; they measure different physical quantities, and only one of them reflects anything the fans can act on.

Three of the five diagnoses we attempted during this study were initially wrong. We blamed a steep control curve when the real cause was firmware override. We blamed a dead service when the real cause was hwmon renumbering. We blamed the OS controller when the Embedded Controller was ignoring it entirely. The misdiagnosis pattern is the transferable content: fan noise on a modern workstation has at least four distinct causes that mimic each other, and each must be ruled out by measurement rather than assumption.

The definitive fix, a firmware change to the board's Q-Fan profile, was never applied. The paper ends with a diagnosed but un-remediated root cause. We staggered the cron jobs that were synchronising thermal bursts, and that removed the worst spikes, but the EC still chases Tctl and will do so until someone changes the BIOS settings at the console.

This is a single-machine study. The Tctl volatility is a property of Zen 5 chiplet scheduling and should generalise; the hwmon indices, PWM channel mappings, and EC behaviour are properties of one ASUS WRX90E-SAGE board and do not.

1. The machine and the symptom

The workstation is an ASUS WRX90E-SAGE motherboard carrying an AMD Threadripper PRO 9975WX (32 cores, 64 threads, Zen 5 architecture), eight 64 GB DDR5-4800 ECC DIMMs, two RTX PRO 6000 Blackwell GPUs, a Mellanox CX-6 NIC, multiple NVMe drives, and a closed-loop AIO liquid cooler on the CPU. Temperature telemetry is available from at least seven subsystems: the CPU die via k10temp, the NCT6798 Super I/O chip (socket thermistor, board ambient, and several auxiliary channels), per-DIMM SPD5118 sensors, NIC ASIC and QSFP module sensors, NVMe controllers, GPU telemetry, and IPMI board sensors.

Fan control runs in three layers. Linux fancontrol drives the NCT6798 PWM channels. A separate systemd unit pins the AIO pump to full speed. Underneath both, the board's BIOS Embedded Controller runs its own firmware fan loop and can override whatever Linux asks for.

The symptom was audible fan hunting: fans cycling up and down with no corresponding change in workload. It had been irritating for long enough that we finally sat down to measure it.

2. Baseline thermal census

On 9 March 2026 we captured a thermal census under sustained two-GPU inference load, drawing roughly 808 W combined GPU power. The numbers came from four sources in parallel: GPU telemetry, lm-sensors across the k10temp, spd5118, mlx5, nct6798, and nvme drivers, and IPMI.

Subsystem	Reading	Headroom to limit
GPU 0	69 °C, 387 W, fan 42%	25 °C to throttle
GPU 1	87 °C, 422 W, fan 54%	7 °C to throttle
CPU Tctl	89.6 °C	4.4 °C to crit (94 °C)
CPU package (IPMI)	88 °C	n/a
CPU power	147 W	203 W to cap (350 W)
DIMMs A1/B1/C1/D1 (GPU-side)	59, 60, 57, 60 °C	All four in ALARM (threshold 55 °C)
DIMMs E1/F1/G1/H1	48, 51, 53, 51 °C	OK
PCIe slot 01	84 °C	11 °C to crit (95 °C)
NIC ASIC	53 °C	52 °C to crit (105 °C)
NIC QSFP module	45 °C	25 °C to crit (70 °C)
Board SYSTIN / CPUTIN	40 °C / 46.5 °C	n/a
NVMe drives	44.9–55.9 °C	32–43 °C to crit (87.8 °C)

Everything sat inside critical limits. The two flagged items were GPU 1's tight 7 °C headroom and the four GPU-side DIMMs running above their 55 °C high threshold from radiant heat off the GPUs. The DIMM temperatures are a chassis-airflow finding, not a fan-control finding; we note them for completeness but they are not addressed by anything that follows.

3. Two defects found on 11 March

The first defect was silent service failure. After a kernel update, the kernel's hwmon device numbering shifted: the NCT6798 moved from hwmon3 to hwmon5, and k10temp moved from hwmon5 to hwmon7. Both the fancontrol configuration and the AIO-pump unit held hard-coded numeric paths. They refused to start, the systemd units reported failure, and the BIOS SmartFan defaults took over.

The trap is in the specific numbers. hwmon5 remained a valid path; it simply pointed at a different chip. A configuration that had asked for the CPU die temperature was now reading the Super I/O. This is a wrong-sensor failure, not a missing-file failure. Fan control failed silently in the direction that looks like working hardware.

The second defect was the sensor choice itself. Even when the configuration was working, it sourced its control temperature from k10temp Tctl, the die junction hotspot. On Zen 5, Tctl swings roughly 30 °C in seconds as threads hop between CCDs (the chiplet clusters that make up the CPU). It is a valid safety signal and a useless control input. Any proportional controller

fed Tctl will oscillate by construction.

We fixed both: resolved the hwmon paths by chip name, switched the control source to the NCT6798 CPUTIN socket-level thermistor, added AVERAGE=10 smoothing, and widened the temperature range to 35–60 °C to suit CPUTIN's lower absolute values.

A third contributor came out in the same pass. A legacy local inference service, enabled at boot with no consumers, ran a tensor-parallel worker in a busy-wait loop that pegged one core at 100%. That core was the heat source producing the Tctl spikes. The service's idle-sleep environment flag only affects GPU sleep, not the CPU scheduler loop. We stopped and disabled it.

Metric	Before	After
Tctl	54–83 °C (30 °C swings)	48 °C, steady
CPUTIN	41 °C	41 °C
PWM1	126–248 (hunting)	118–122 (stable)
fan2 RPM	969–1,783 (cycling)	~1,783 (steady)
fancontrol	failed	active
AIO pump unit	failed	active
Legacy inference service	100% CPU spin	stopped

The CPUTIN row is the important one. It never moved. The machine was never actually hot; the controller was chasing a noisy number.

A caveat applies here. The before/after comparison is observational, taken across a configuration change on a live workstation with concurrently changing workload. We disabled the busy-wait service in the same pass as the sensor change. The sensor fix and the load removal are not cleanly separated in these numbers.

4. The cleaner measurement

On 16 March we re-measured both candidate inputs on the same idle machine, with the busy-wait service already gone.

CPUTIN held rock-stable at 38.5 °C (38,500 millidegrees, no variation across all samples). Tctl swung 59–74 °C within seconds. The PWM registers (pwm1=104, pwm2=62) held constant across a 20-second window, yet fan2 drifted 1,047–1,477 RPM and fans 3–5 drifted 470–572 RPM.

This is the strongest single result in the study. The two sensors are not noisy versions of each other. Tctl measures the hottest point on a die that heats and cools in milliseconds; CPUTIN measures the socket, a thermal mass that changes over minutes. Only one of them reflects anything the fans can act on.

We also confirmed that the pump channel is not software-controllable on this board. pwm7_enable=0 indicates firmware full-speed mode; the systemd unit that tries to set it to manual is a no-op that nonetheless reports success. The pump runs at roughly 3,924 RPM regardless of what Linux asks.

Two hypotheses emerged from this session. First, CPUTIN idling at 38.5 °C sat only 3.5 °C above the configured MINTEMP of 35, on a steep part of the linear curve, so small movements between 5-second polling intervals might still oscillate PWM. Second, the board's SmartFan was fighting fancontrol on the same channels. We thought the first was more likely. We were wrong.

A separate investigation the same day confirmed that loud fans are sometimes correct. One episode traced cleanly to GPU 0 at 99% utilisation, 78 °C and 496 W serving a genuine

inference request from a remote agent framework, with GPU 1 idle at 3%/56 °C and CPU load at 3.5%. The controller was behaving correctly; the machine was simply busy. The control-defect hypothesis must be falsified against actual load before configuration is touched.

5. Cooling-zone separation

By 21 March the configuration had regressed to driving pwm1 through pwm6 all from CPUTIN, tying chassis fan behaviour to CPU temperature. This is wrong. CPU temperature has nothing to do with chassis air temperature; driving them from the same input makes chassis acoustics a function of an irrelevant variable.

We restored the intended topology:

- AIO pump: fixed full speed, firmware-controlled (pwm7=255).
- CPU cooling: fancontrol owns pwm1 only, sourced from CPUTIN, targeting fan2.
- Chassis fans: a dedicated persistent service owns pwm2–pwm5, driven by a composite of GPU, board ambient, DIMM, and NIC temperatures.

Verification over multiple sampling intervals: Tctl moved between roughly 52.9 °C and 70.5 °C, CPUTIN stayed at 41.0 °C, pwm2–pwm5 held at 120 throughout, pwm1 held at 85. CPU thermal spikes no longer modulated the chassis headers.

We left pwm6 in firmware auto mode (enable=5) rather than reassigning it by guesswork; our header-mapping notes were inconsistent and the channel needs empirical mapping at the physical case.

6. The firmware loop

Fan pulsing returned in April with a distinct signature: a slow breathing rather than the earlier fast hunting. We spent three sessions between 5 and 7 April separating three different causes.

The first two were crash-looping services. A text-to-speech service had a misconfigured model path pointing at a directory that no longer existed; it threw a validation error on load, exited, and restarted, accumulating thousands of restarts before we noticed. A second unit was crash-looping on a 30-second restart interval because a required signing-key environment variable was absent from its environment file; it had been looping since roughly 06:00 that morning. Restart intervals print themselves onto the acoustics. A unit restarting every 30 seconds produces a 30-second thermal pulse.

After disabling both, a 3-minute observation at 15-second intervals showed no meaningful hunting: fan2 792–801 RPM, fan3 400–401, fan4 401–403, fan5 401–403, fan7 3,890–3,947; Tctl 48.2–48.9 °C, CPUTIN 42.5 °C, SYSTIN 41.0 °C, NIC ASIC 63.0 °C. A follow-up 4-minute monitor-only loop at 5-second intervals found no chassis hunting at all: chassis PWM held at 120, chassis fans 389–391 RPM, high-speed fan roughly 3,900 RPM.

The breathing came back anyway.

On 7 April, a roughly 3-hour session with a 10-minute continuous monitor (14 samples at 30-second intervals) plus high-resolution captures established that the board's Embedded Controller runs its own firmware fan loop that watches Tctl and overrides the Linux PWM settings.

The mechanism, captured at 1-second granularity over 40 samples:

1. Tctl spikes 49 °C to 70 °C in 8 seconds from a brief single-threaded CPU burst.
2. The EC ramps the fans with a 4–6 second lag.

3. By the time fan speed peaks (1,133 RPM), Tctl has already fallen to 58 °C. The fans overshoot.
4. Fans slow; the next burst starts the cycle again.
5. Net result: a 40–60 second breathing period.

The decisive evidence is that the PWM registers stayed constant while RPM swung 40–50%. Chassis PWM locked at 120, CPU PWM at 96–100, yet CPU fan RPM varied 795–1,182 and chassis fans varied 395–550. Linux fancontrol was correctly using CPUTIN (stable at 41–43 °C) throughout. The EC simply ignores it. A 2-second capture over 30 samples confirmed the pattern across all fan channels.

PWM constant while RPM swings is the diagnostic fingerprint of firmware override. It cannot be produced by the OS controller, and it distinguishes an EC problem from a software problem in a single capture.

We identified two workload drivers, both classified as normal rather than pathological. The primary driver was bursty single-threaded agent processes (five concurrent): Node.js single-threaded bursts pin individual cores past 100%, and one hot CCD drives Tctl up 10–20 °C while total CPU utilisation reads only 3–5%. The secondary driver was a cron thundering herd every 5 minutes at :00/:05, firing simultaneously: a fabric health probe (iperf3, 182% CPU), a memory-metrics job (Python, 230% CPU), a cloud-CLI metrics job (56% CPU), a backup rsync (73% CPU), plus Prometheus exporters, a temperature-alert job, and a routing watcher. It coincided with a GPU inference request that took GPU 0 to 453 W at 100% utilisation.

We staggered the cron entries: two light jobs to 1-59/5, one light job to 2-59/5, the heavy iperf3 fabric probe to 3-59/5, and one moderate job to 4-59/5. That removed the worst synchronised spikes.

The definitive fix, setting the board's Q-Fan profile to a silent mode with wider hysteresis so the EC itself stops chasing Tctl, was not applied. It requires BIOS access via KVM or physically at the machine. The paper ends here: diagnosed, un-remediated.

7. What we learned, and what we did not

Ten findings emerged from this study. We state them as distinct claims with their scope.

Claims that generalise beyond this machine:

1. Junction temperature is a safety signal, not a control input. Zen 5 Tctl swings 20–30 °C in seconds from single-core bursts. Any proportional controller fed Tctl will oscillate by construction.
2. A socket thermistor is the correct input for CPU fan control on this class of board, with averaging. CPUTIN never moved across the entire hunting episode.
3. hwmon numeric paths are not stable identifiers. A kernel change can renumber chips; worse, a stale index can silently resolve to the wrong chip. Fan configurations must resolve devices by chip name.
4. Cooling zones must be separated by heat source. Driving chassis headers from CPU temperature makes chassis acoustics a function of an irrelevant variable.
5. PWM constant while RPM swings is the diagnostic fingerprint of firmware override.
6. Restart intervals print themselves onto the acoustics. A crash-looping service is audible.
7. Loud fans are sometimes correct. The control-defect hypothesis must be falsified against actual load.

Claims specific to this board:

1. A software controller cannot win an argument with the EC. The correct sensor selection was in force in April and the breathing still occurred.
2. The pump channel is not software-controllable on this board; the unit that tries to set it reports success while its writes are ignored.

A meta-claim:

1. Fan noise has at least four distinct causes on one machine, and they mimic each other: wrong sensor (fast hunting), dead controller (BIOS defaults), crash-loop churn (periodic ramping at the restart interval), synchronised cron (5-minute ramping), and firmware EC lag (40–60 second breathing). Each was misdiagnosed as the others at least once during this study.

The recurring institutional lesson is that several of these controls failed in the direction that looks like success. A stale hwmon index that still resolves. A pump unit that reports active while its writes are ignored. A fancontrol daemon correctly reading the right sensor while the EC overrides its output. A control that cannot report its own ineffectiveness is not a control.

8. Limitations

This is n=1. One board, one CPU, one cooler, one chassis. The Tctl-volatility claim is a property of Zen 5 chiplet scheduling and should generalise to other Zen 5 systems. The specific hwmon indices, PWM channel map, and EC behaviour are properties of this ASUS WRX90E-SAGE board and may not apply to other boards, even other WRX90-based boards.

No controlled A/B was run. The March 11 before/after comparison confounds the sensor change with the removal of the busy-wait service. The March 16 re-measurement, taken with the load already gone, is the cleaner evidence for the sensor claim, but it is still observational.

The definitive fix was never applied or verified. We diagnosed the EC loop, staggered the cron jobs, and stopped there. Whether a firmware Q-Fan profile change actually eliminates the breathing is unknown; we have only the hypothesis.

The DIMM alarm figures (four channels at 57–60 °C against a 55 °C high threshold) are a radiant-heat-from-GPU finding. Nothing in this study addresses them.

We do not know which physical fan headers correspond to pwm6. The local notes were inconsistent and we did not trace the wiring.

The service-health context on 7 April showed zero crash-looping services and all 16 key services at 0 restarts in the preceding 6 hours. Two pre-existing unrelated failures (a Drive-verification timer and a memory-reflection job timing out on embedding-model load) were present but not implicated. A browser with 65 renderer tabs and 16 GB resident was a memory consumer but not a CPU driver. We mention these to bound the state of the machine during the final captures, not to claim the machine was otherwise healthy.

9. The thesis, restated

Fan-control quality is a sensor-selection problem before it is a control-theory problem. The whole March intervention amounted to changing which number the loop reads, and it converted a 30 °C-amplitude input into a 0.5 °C-amplitude one. No PID controller was needed; a better input made the linear controller adequate.

The honest arc of this study is that we were wrong three times before we were right, and the final root cause remains unfixed. That is the normal shape of debugging work on a live system. We report the wrong turns because the misdiagnosis pattern is the transferable content.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. PT-TN-2026-009. Measurements taken 9–21 March 2026 and 5–7 April 2026. Raw artefacts retained.