PRODUCT

# Automotive – ADAS

Automotive Adas Nvidia Jetson

SiliconCentric | Independent Verification & Validation

SiliconCentric

PRODUCT

# Introduction

Our client is a Tier-1 automotive supplier. Their product is a forward-facing ADAS domain controller: an NVIDIA Jetson system-on-module on a custom carrier, paired with an external STM32 acting as safety monitor and power sequencer. First-article boards were back from fabrication, their firmware team was scheduled to start in six weeks, and no independent party had confirmed the board worked as an integrated system. We collaborated with them on this project by providing independent verification and validation of the board.

- The compute topology is an SoC plus dedicated companion microcontroller: a Jetson SoM for perception workloads, an external STM32 for power sequencing, rail supervision, and watchdog duty.
- Hardware under validation: four GMSL2 camera links through MAX9296-class deserializers, two CAN-FD radar channels, a 1000BASE-T1 Ethernet port for LiDAR, an SPI-attached IMU, GNSS over UART, NVMe storage on PCIe, and the 12 V automotive power tree.
- The client was at first-article stage — boards fabricated, unpowered beyond a smoke test, firmware not yet started.
- Our assessment carries no stake in the design: we did not draw the schematic, route the board, or write any of its firmware, and our findings do not affect our compensation.
- Scope was bounded to independent validation only — no feature development, no out-of-scope engineering, no product firmware authoring.
- We worked from the hardware design package alone: schematics, interface control documents, and the BOM, with no access to firmware source.

2

PRODUCT

# Hardware Topology

The board is an SoC-plus-companion-uC topology: the Jetson SoM handles sensor-fusion compute, and the STM32 owns everything that must work before and after Linux is alive. Validation scope covered every interface on the carrier and the interconnect between the two processors; product firmware and certification sign-off sat outside the scope.

- Topology: Jetson SoM plus external STM32, linked by UART for command traffic and two discrete GPIO lines for reset and fault signaling.
- The STM32's job: sequence six carrier rails at power-on, supervise them continuously, run the external watchdog, and hold the SoM in reset on rail faults.
- Buses in scope: GMSL2 x4, CAN-FD x2, 1000BASE-T1, PCIe Gen4 x4 to NVMe, SPI (IMU), I2C (camera control, PMIC telemetry, EEPROM), UART (GNSS, STM32 link).
- Operating envelope in scope: −40 °C to +85 °C carrier rating, 9–16 V automotive input with load-dump transients per ISO 7637-2 pulse profiles.
- Out of scope: perception firmware, application code, ISO 26262 certification sign-off — we produce evidence; the client's assessor judges it.
- The Jetson family is one of the six architectures Polaris already runs on, so the OS baseline — BSP, bootloader, kernel — was inherited, and engineering effort went to this carrier's delta rather than silicon bring-up.

3

PRODUCT

# Problem Statement

Boards existed, the firmware team's start date was contractually committed to the OEM, and the only evidence the design worked was that each major device had a datasheet and a vendor dev kit behind it. Under ISO 26262 ASIL-B, the client would eventually have to show an assessor evidence of hardware integration testing — evidence that did not yet exist.

- Every device was proven in isolation, never as a board: NVIDIA validated the SoM on its own carrier, the deserializer vendor on its EVM — but nothing had exercised this carrier's GMSL2 routing, PCIe lane mapping, six-rail power tree, and STM32 handoff together.
- Firmware coverage would be feature-shaped, not board-shaped: the product uses two of four camera links at launch, so links three and four, the second CAN-FD channel, and the PMIC telemetry bus would ship untested by the feature team.
- No stress, no extremes, no duration was planned: no −40 °C cold-crank cycling, no 85 °C soak under full camera load, no multi-day burn-in, no brown-out campaign against the 9 V floor this board sees on every engine start.
- Nothing was set to run alongside development: validation was a gate at program end, so any hardware fault would surface as a firmware bug, debugged by engineers who did not design the carrier.
- ISO 26262 requires traceable integration-test evidence for ASIL-B items; ad-hoc bench notes do not survive an assessor.
- The GMSL2 links run at 6 Gbps over a four-connector stack — the kind of channel that passes at 25 °C and fails at temperature.
- A field failure here is a camera blackout in a driver-assistance function, with recall cost attached.

PRODUCT

# Customer Requirements

4

The client's requirements were direct: validate the whole board, under stated limits, with evidence that survives their ISO 26262 assessor rather than internal review.

- Operating temperature: the carrier shall operate from −40 °C to +85 °C.
- Supply voltage: 12 V nominal automotive input, operating window 9 V to 16 V. The board shall survive load-dump transients per ISO 7637-2 up to 36 V without damage; over-voltage protection shall clamp above 16 V continuous.
- Brown-out: the board shall ride through input sag to 6 V during engine crank and recover without operator intervention; below 6 V it shall shut down cleanly with no record loss.
- Over-current: each of the six carrier rails shall have over-current protection; rail OCP shall trip within 10 ms and the STM32 shall hold the SoM in reset until the fault clears.
- Humidity: the board shall operate at up to 95% RH non-condensing and shall not be exposed beyond that.
- Full-board functional validation: both Jetson CPU clusters, the GPU, all DRAM regions, all four GMSL2 links, both CAN-FD channels, the T1 port, PCIe/NVMe, IMU, GNSS, and every I2C device on the carrier, exercised as one system.
- Coverage independent of firmware scope: every subsystem validated regardless of the launch feature set, driven through a HAL built from the design package, not from firmware.
- Stress matrix: thermal sweep −40 °C to +85 °C, 72-hour endurance under camera load, 5,000-cycle power cycling, brown-out to 6 V, and GMSL2 link-quality margining at temperature.
- Continuous validation alongside firmware development: CI integration so every firmware candidate revalidates on real hardware throughout the program.
- Regulatory evidence: timestamped, operator-attributed, version-pinned records with bidirectional requirement-to-result traceability for the ASIL-B case.
- A web console any test technician can run a campaign from, learned once, identical on every program.
- An immutable, append-only evidence store — records that cannot be modified or deleted, reusable for the next carrier revision.
- Setup measured in minutes: a bench any operator can bring from boxed board to running campaign, with no rig to assemble.

PRODUCT

# Architecture

The platform splits across two machines: the target carrier runs the validation OS, the HAL, and the client; an external x86 host runs the server, dashboard, and SDK. Test execution stays isolated from data processing, the Jetson keeps its compute budget for behavior representative of the field, and the host absorbs database, aggregation, and UI load — so camera-saturation tests measured the board, not the tooling.

- Polaris OS (the validation OS) — a hardened Linux image for the Jetson SoM that boots straight into a known validation state, giving the client a reproducible baseline.
- HAL — the only board-specific layer; on this carrier it exercises GMSL2, CAN-FD, T1 Ethernet, PCIe, SPI, I2C, and the STM32 link behind a uniform interface, which keeps the automotive test suites portable across carrier revisions.
- Client — the lightweight on-target agent driving every test through the HAL; gRPC on the Jetson, FlatBuffers on the STM32's RTOS; three timestamps per event (origin, relay, server); load light enough that measured camera-path latency reflects field behavior.
- Server — orchestration, scheduling, and the append-only records store on the x86 host; it ran four first-article carriers in parallel from one machine and hosts the dashboard and API.
- Dashboard — the operator console: connect a carrier, configure and launch, watch live rail and link telemetry during runs, review time series after; every state-changing action operator-attributed for the ASIL-B record.
- Python SDK — automation and CI; drives everything the console does; C/C++ bindings were available but this team stayed in Python.
- The OS and HAL are the two board-specific layers and were heavily modified for this carrier and this SoC; the client, server, dashboard, and Python SDK were configured to the client's requirements, but their core structure and working style remain the same across every board and every industry.

6

PRODUCT

# Design

Three constraints drove the design: the −40 °C to +85 °C envelope, the 6 Gbps serial links that only fail at the corners, and an ASIL-B evidence chain that must attribute every record. Everything below is a trade-off against one of those.

- Validation OS design — read-only rootfs with A/B partitions and automatic rollback, because a bricked image inside a thermal chamber costs a full chamber cycle; the trade-off is slower iteration on OS changes, accepted. The STM32 runs FreeRTOS, chosen over Zephyr for the client's existing in-house familiarity.
- HAL design — partitioned by schematic sheet, each function traced to an ICD section; interrupt-driven capture for CAN-FD and GMSL2 lock-status because polling at the required rate would disturb the measurement; interrupt storms under fault injection are bounded with rate limiting.
- Client design — origin timestamps taken in the interrupt path, not the reporting path, so brown-out event ordering is correct; on rail-fault the client flushes buffered records before the STM32 pulls reset, trading a 40 ms shutdown delay for zero record loss.
- Server design — three-timestamp causality lets a rail sag be ordered against the PCIe error it caused; retention sized for 72-hour runs across four boards; records are batched and compacted.
- Dashboard design — rail voltages, GMSL2 lock state, and CAN-FD error counters are first-class views because those are what an automotive operator triages first; run health and drop counts sit on every screen.
- SDK design — the client's engineers script the acceptance gate and the overnight thermal sweep; a direct API path exists for single-peripheral checks at the bench, at the cost that such runs are not evidence (see recording rule below).
- OS and HAL design was board-specific and heavily reworked; the upper four components were configured to this client's workflow while their structure and working style stayed constant.

7

PRODUCT

# Implementation

Project work concentrated in the two board-specific layers — the OS delta for this carrier and the HAL written from the design package — plus the automotive campaigns. The client, server, dashboard, and SDK existed on day one and took configuration, not construction.

- Validation OS — Yocto image for the Jetson SoM: inherited BSP and bootloader, carrier-specific device tree (camera deserializers, T1 PHY, STM32 UART), kernel config trimmed to validated drivers; OTA with A/B rollback; boot to validation-ready in under 30 seconds; per-build SBOM with CVE tracking; the STM32 FreeRTOS image built under the same stack.
- HAL — implemented sheet by sheet: memory-mapped register interfaces for the deserializers, interrupt handlers for CAN-FD, DMA paths for camera frame capture, bit-banged recovery for a stuck I2C bus; building from the design package surfaced a schematic-to-BOM mismatch — the camera-control I2C bus specified 4.7 kΩ pull-ups on the schematic that were absent from the BOM.
- Client — C++ daemon on the Jetson, event and polling paths into the HAL, hardware-clock timestamping, graceful shutdown without record loss, gRPC upstream; FlatBuffers link to the STM32.
- Server — telemetry receiver, versioned record schema, campaign scheduler, REST API, dashboard hosting, four-carrier parallelism on one host.
- Dashboard — carrier provisioning, campaign configuration, live rail and link streaming, time-series review, CSV/MDF4 export for the client's existing road-load toolchain, operator attribution on every action.
- SDK — Python bindings, example scripts for each bus, and CI wiring that fires the acceptance suite on every firmware candidate the client's team produces.
- Board-specific work: OS delta and HAL. Configuration work: client transport settings, server retention, dashboard views, SDK examples.

PRODUCT

# Test - IV&V

The strategy ran in three rings: prove the validation stack itself, prove the OS/HAL on this carrier, then run the board campaigns — one-shot, long-running, event-based, and monitoring — with everything from the dashboard written to the append-only store.

- Stack self-verification: 100% test coverage of the client, server, dashboard, and Python SDK — the platform is proven before it produces evidence about someone else's board.
- OS/HAL verification: A/B boot and rollback exercised, OTA path verified end to end, secure boot checked on both accept and reject paths.
- Bench setup: carrier cabled to the x86 host, programmable supply and chamber configured, board provisioned from the dashboard — a fifteen-minute repeatable procedure.
- Functional campaign: per-link GMSL2 lock and CRC suites, CAN-FD frame and error-injection suites, T1 link and PTP checks, PCIe/NVMe throughput, IMU and GNSS sanity, STM32 sequencing verification against the ICD.
- CPU, GPU, and memory campaign: cache and DRAM pattern stress across both CPU clusters, GPU load with thermal-throttling characterization at 85 °C, DMA integrity under camera saturation, and interrupt-latency measurement with all sensor paths active.
- Watchdog and fault-injection campaign: the STM32's external watchdog exercised with induced SoM hangs, per-rail fault injection confirming reset-hold and clean-recovery behavior, and the SoM↔STM32 link driven through heartbeat-timeout, frame-corruption, and retry paths — the reset/boot-order contract tested as its own suite.
- Debug posture: JTAG/SWD access verified locked on production-fused configuration and retained on first articles, where trace and PMU counters anchored the latency measurements.
- One-shot tests served as the acceptance gate: configure, execute, capture, single pass/fail per subsystem.
- Long-running: 72-hour full-sensor-load soak with periodic snapshots, catching a slow NVMe temperature climb traceable to an underspecified thermal pad.
- Event-based: silent runs emitting records only on GMSL2 lock loss, CAN error-counter thresholds, rail excursions, or brown-out — this mode caught link four dropping lock intermittently above 80 °C, isolated to a wrong-value PoC filter inductor.
- Monitoring: passive telemetry — rails, SoM and NVMe temperatures, CPU load — alongside a representative perception workload supplied as a binary.
- Oscilloscope and logic-analyzer captures anchored the electrical evidence: the power-sequencing campaign showed the STM32 releasing the SoM rail 8 ms before the NVMe rail settled, violating the module's power-on spec and explaining cold-boot enumeration failures at −20 °C.
- Recording rule: every run launched from the dashboard is recorded, always, into the append-only store. The Python SDK can run 100% of what the dashboard can run — module, peripheral, or system level — for quick independent checks, and SDK-launched runs of this kind are not written to the records store. The store is the evidence; the SDK path is the bench tool.
- Traceability: each ASIL-B-relevant requirement traced bidirectionally to its tests and records; schemas version-pinned; every run reproducible.

9

PRODUCT

# Deliverables

The client received the complete validation platform tailored to their carrier and silicon — full source, their records, no lock-in.

- Validation OS image for the Jetson carrier with the full Yocto build system, recipes, and per-build SBOM.
- Board-specific BSP and HAL source covering every peripheral, maintained against the design package and carried forward as the carrier revs.
- Compiled client for the Jetson plus the STM32 FreeRTOS build.
- Server and dashboard as deployment-ready containers configured for their bench.
- Complete test suites — functional, thermal, endurance, power, GMSL2 margining, CAN-FD error injection — with recorded baselines.
- Python SDK with documentation, example scripts, and the CI integration revalidating every firmware candidate on hardware.
- The append-only records store, delivered as the client's property — their ASIL-B qualification evidence.
- An independent validation report: the sequencing violation, the PoC filter finding, the pull-up mismatch, performance metrics, pass/fail status, and coverage.
- Bench procedures for board acceptance and campaign re-runs; no lock-in.

10

PRODUCT

# Conclusion

The carrier was proven on the bench before the firmware team committed a single sprint to it. The power-sequencing violation, the marginal GMSL2 link at temperature, and the missing I2C pull-ups were corrected in a carrier spin at the cheapest point in the program — before tooling, before firmware, before the OEM milestone.

- Three hardware defects that would have cost months of firmware-side debugging were fixed in one board revision, with assessor-ready evidence rather than internal assertion.
- The suites, CI gate, and records format carry forward to the next carrier revision — only the HAL delta gets re-verified — and validation now runs continuously in CI rather than as a gate. The standing caveat holds: we produce the evidence; the client's assessor judges it.

II

PRODUCT

# Lessons Learned

This engagement tightened how we sequence power-tree validation and document review on automotive carriers.

- Sequencing and document review move to day one: the 8 ms rail violation and the schematic-to-BOM pull-up mismatch were both findable before any functional suite ran — on the next automotive engagement, rail-sequencing checks and design-package cross-check precede everything else.
- SoC and uC integration costs real time: the Jetson module's undocumented sensitivity to NVMe rail timing at cold, and UART framing errors during STM32-driven resets, both argue for validating the reset/boot-order contract between processors as its own campaign, not as a side effect of other suites.

CONTACT

# SiliconCentric

EMAIL sandesh@siliconcentric.com

PHONE (669) 356-1998

WEB https://www.siliconcentric.com

ADDRESS San Jose, CA, USA