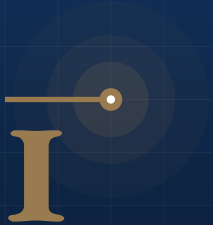
PRODUCT

# Robotics - Drone

## Robotics Drone Nvidia Jetson

SiliconCentric | Product overview

SiliconCentric



# Introduction

ROBOTICS DRONE NVIDIA JETSON

# Introduction

Our client is an inspection-drone manufacturer. Their product consolidated two boards into one: a new flight core carrying an NVIDIA Jetson SoM for perception alongside a dedicated external STM32 for stabilization, on a single carrier that had just returned from fabrication. Before committing the airframe program to it, we collaborated with them on this project by providing independent verification and validation of the board.

- Compute topology: Jetson SoM plus external STM32 companion; the Jetson runs perception, mapping, and mission logic, the STM32 runs the 2 kHz attitude loop, actuator outputs, and failsafe authority.

# Introduction

- Under validation: a dual-IMU/magnetometer/barometer sensor cluster on SPI and I2C, a multi-constellation GNSS receiver with its RF front end, eight DShot ESC outputs, a three-axis gimbal port with camera power, the RC receiver link on UART with SBUS failsafe, a LiPo battery-management path with current shunt and cell telemetry, a USB 3.0 camera port, NVMe, and the 6S power tree.
- Program state: first articles fabricated, airframe integration and flight-software work gated on hardware confidence.
- Independence: no technical, managerial, or financial stake in the design — the finding against the carrier's own routing was reported the day it was measured.

# Introduction

- Bounded scope: independent validation only — no flight-control development, no firmware authoring, no out-of-scope engineering.
- Working inputs: the hardware design package — schematics, ICDs, BOM — with no access to firmware source.

2

# Hardware Topology

ROBOTICS DRONE NVIDIA JETSON

# Hardware Topology

The topology is SoC-plus-companion-uC with a flight-safety division: the Jetson advises, the STM32 flies. Scope enclosed the carrier, its sensors, its actuator outputs, and its power path; flight-control laws, airframe integration, and any aviation-authority process stayed outside.

- Topology classification: Jetson SoM plus external STM32, interconnected by UART for mission traffic and SPI for high-rate state exchange, with a hardwired failsafe discrete from the STM32 that survives any Jetson state.

# Hardware Topology

- The STM32's job: the 2 kHz stabilization loop, DShot output generation, RC-link failsafe, battery supervision with landing-trigger authority, and arming interlocks.
- In-scope buses: SPI x3 (IMU A, IMU B, inter-processor), I2C x2 (mag, baro, BMS telemetry), UART x3 (GNSS, RC, inter-processor), DShot x8, USB 3.0, PCIe to NVMe, gimbal PWM and power.
- Envelope in scope: −10 °C to +50 °C, the 6S LiPo discharge envelope including sag under punch-out current, and the vibration profile of an eight-rotor airframe — sensor behavior under vibration was in scope on a shaker, not in flight.

# Hardware Topology

- Out of scope: flight-control laws, mission software, and airworthiness processes — we produce the evidence; the client's flight-test organization and their regulators judge it.
- Jetson is one of the six architectures Polaris already runs on, so the OS baseline was inherited and effort went to the carrier delta rather than silicon bring-up.

3

# Problem Statement

ROBOTICS DRONE NVIDIA JETSON

# Problem Statement

The carrier consolidated proven subsystems into an unproven whole, and the program's first flight was scheduled against it. Aviation-adjacent hardware punishes integration faults with crashes, and crashed prototypes set programs back months.

- Proven in isolation, never as a board: the IMUs, GNSS receiver, and SoM each had vendor characterization — none of it with this carrier's ground partitioning, its USB 3.0 routing beside the GNSS front end, its DShot line lengths, or its shared-rail sag under motor punch-out.

# Problem Statement

- Firmware coverage is feature-shaped, not board-shaped: flight software exercises the sensors and ESCs continuously, but the second IMU's failover path, the gimbal port's power limits, the BMS cell-telemetry bus, and the SBUS failsafe timing fell outside the planned feature tests.
- No stress, no extremes, no duration: no −10 °C cold-start, no 50 °C soak with full perception load, no endurance against battery-sag brown-out at punch-out current, no vibration campaign with continuous electrical monitoring — a drone's every flight includes most of these.

# Problem Statement

- Nothing runs alongside development: validation was a pre-flight gate, so a marginal sensor bus would first appear as an attitude excursion in flight test — the most expensive and dangerous place to find it.
- The client's operations-approval pathway expects documented evidence of hardware failsafe behavior; insurance underwriting for the inspection fleet expected the same.
- GNSS sensitivity is a board-level property: a receiver that loses two dB to its own carrier's digital noise flies worse in exactly the degraded conditions where it matters most.
- A crash traced to the flight core would ground the fleet and the program with it.

4

# Customer Requirements

ROBOTICS DRONE NVIDIA JETSON

# Customer Requirements

The client's requirements were direct: validate the whole board, under stated limits, with evidence supporting the flight-readiness review.

- Operating temperature: the board shall operate from −10 °C to +50 °C, including cold start at −10 °C.
- Supply voltage: 6S LiPo, 22.2 V nominal, operating window 18.0 V to 25.2 V. The board shall survive charge-level input at 25.2 V continuously; over-voltage protection shall clamp above 26 V.

# Customer Requirements

- Brown-out: the board shall ride through battery sag to 18.0 V under punch-out current without reset of either processor; below 18.0 V the STM32 shall retain failsafe and landing authority with no record loss.
- Over-current: the BMS shunt shall meter the full discharge envelope; the gimbal port shall enforce its power limit with over-current shutdown within 10 ms; a shorted gimbal or camera load shall not disturb the flight-critical rails.
- Failsafe timing: SBUS failsafe shall assert within 100 ms of RC-link loss, in both disconnect and receiver brown-out cases.

# Customer Requirements

- Humidity: the board shall operate at up to 90% RH non-condensing and shall not be exposed beyond that.
- Full-board functional validation: all Jetson cores and the GPU, every DRAM region, NVMe, both IMUs, mag, baro, GNSS, all eight DShot outputs, the gimbal port, the RC link with failsafe, the BMS path, and USB 3.0 — one integrated system.
- Coverage independent of firmware scope: every subsystem validated regardless of the flight-software feature set, through a HAL built from the design package.

# Customer Requirements

- Stress matrix: −10 °C to +50 °C sweep with cold-start cycling, 72-hour endurance under perception load, battery-sag and brown-out campaigns replaying punch-out current profiles, vibration runs with continuous monitoring, and GNSS sensitivity characterization with the carrier's digital subsystems exercised in combinations.
- Continuous validation alongside development: CI integration revalidating every flight-software candidate on the bench before it flies.
- Evidence: timestamped, operator-attributed, version-pinned records with bidirectional traceability for the flight-readiness review and insurance file.
- A web console the client's test pilots and technicians run without embedded expertise.

# Customer Requirements

- An immutable, append-only evidence store anchoring the airworthiness narrative across carrier revisions.
- Setup in minutes: documented bench — carrier, ESC load rig, RF chamber time, battery emulator, host — reproducible by any technician.

# Architecture

ROBOTICS DRONE NVIDIA JETSON

# Architecture

The carrier runs the validation OS, HAL, and client; an external x86 host runs the server, dashboard, and SDK. Execution stays isolated from data processing and the carrier keeps its resources for representative behavior — which mattered because GNSS desense had to be measured with the Jetson working, since an idle perception computer is not the vehicle that flies.

- Polaris OS (the validation OS) — a reproducible Linux image for the SoM booting to a known state, one baseline across the first-article set.

# Architecture

- HAL — the single board-specific layer, exercising both IMUs, mag, baro, GNSS, DShot outputs, gimbal port, RC link, BMS telemetry, USB 3.0, and NVMe behind a uniform interface that keeps the UAV suites portable to the next carrier revision.
- Client — the on-target agent driving tests through the HAL; gRPC on the Jetson, FlatBuffers to the STM32's Zephyr image; three timestamps per event (origin, relay, server); light enough that sensor-noise floors reflect the vehicle.
- Server — orchestration, scheduling, and the append-only records store on the host, three carriers in parallel, dashboard and API hosted.

# Architecture

- Dashboard — the console: provision a carrier, configure and launch, live IMU spectra, GNSS carrier-to-noise, per-motor DShot telemetry, battery-path data, time-series review; every state-changing action operator-attributed.
- Python SDK — automation and CI; scripts everything the console does; C++ bindings served the vibration-rig synchronization harness.
- 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

# Design

ROBOTICS DRONE NVIDIA JETSON

# Design

The binding constraints: a 2 kHz control loop that cannot jitter, sensors whose noise floors are the product, a battery that sags 20% under punch-out, and a vibration environment that turns marginal solder and marginal mounting into intermittent electronics.

# Design

- Validation OS design — read-only rootfs with A/B rollback; boot under 25 seconds because field technicians reflash between test sorties — the trade-off was excluding the client's perception stack from the validation image, with representative load supplied as a benchmark binary instead. The STM32 runs Zephyr, matching the client's eventual flight-controller RTOS so timing findings transfer directly.
- HAL design — partitioned by schematic sheet against ICD clauses; IMU and DShot paths are interrupt-and-DMA driven because the evidence is in microsecond timing, while BMS telemetry polls at 5 Hz. One real pitfall: DMA-driven dual-IMU capture on shared SPI DMA channels can silently serialize, adding 80 µs of skew — caught and rerouted to independent channels.

# Design

- Client design — origin timestamps latched at the STM32 for control-path events and at the driver boundary on the Jetson, correlated over SPI; on battery-critical the client flushes records before the STM32's landing logic takes the rails, so every event leaves a complete record.
- Server design — three-timestamp causality orders a rail sag against the IMU noise burst it caused; retention sized for 72-hour endurance plus high-rate vibration captures, with IMU spectral data stored raw — bulky, but vibration signatures are the airframe team's design input.

# Design

- Dashboard design — IMU noise spectra, GNSS C/N0 per constellation, DShot per-motor feedback, and battery sag curves are first-class views — the flight-test team's triage order; run health and drop counts on every screen.
- SDK design — the client scripted the acceptance gate, the desense matrix, and shaker-synchronized campaigns; the direct API path served single-sensor bench checks, at the stated cost that those runs produce no records.
- OS and HAL design was board-specific and heavily reworked; the four upper components were configured to this client's flight-test workflow while their structure and working style stayed constant.

7

# Implementation

ROBOTICS DRONE NVIDIA JETSON

# Implementation

Work went to the OS delta, the HAL, and the UAV campaigns; the four upper components were configured, not constructed.

- Validation OS — Yocto image for the Jetson SoM: inherited BSP and bootloader, carrier device tree covering the sensor cluster, GNSS, gimbal port, and BMS, minimal kernel; OTA with A/B rollback; boot under 25 seconds; per-build SBOM with CVE tracking; the STM32 Zephyr image built in the same tree.

# Implementation

- HAL — implemented sheet by sheet and bench-verified as written: independent-channel DMA capture for both IMUs, DShot generation and telemetry decode, GNSS UBX-protocol drivers with C/N0 extraction, BMS cell-telemetry drivers, gimbal power-limit enforcement; the design-package pass surfaced a BOM mismatch — the IMU-B anti-vibration mount pad called for a specific durometer grade in the schematic notes that the BOM replaced with a stiffer generic part, a mechanical substitution with direct sensor-noise consequences.
- Client — C++ daemon, event and polling paths, dual-domain timestamp correlation, battery-hardened shutdown, gRPC upstream, FlatBuffers to the STM32.

# Implementation

- Server — telemetry receiver, versioned schema, scheduler, REST API, dashboard hosting, three-carrier parallelism.
- Dashboard — carrier provisioning, campaign configuration, live spectra and C/N0 streaming, time-series review, export for the flight-readiness review, operator attribution.
- SDK — Python bindings, C++ shaker-sync bindings, example scripts, CI wiring firing the acceptance suite on every flight-software candidate.
- Board-specific: OS delta and HAL. Configuration: client correlation, server retention, dashboard views, SDK harness glue.

# Test - IV&V

ROBOTICS DRONE NVIDIA JETSON

# Test - IV&V

Testing ran stack-first, then OS/HAL, then the carrier across one-shot, long-running, event-based, and monitoring campaigns — every dashboard-launched run committed to the append-only store.

- Stack self-verification: 100% test coverage of the client, server, dashboard, and Python SDK before it produced evidence about this carrier.
- OS/HAL verification: A/B boot and rollback exercised, OTA verified end to end, secure boot checked on accept and reject paths.

# Test - IV&V

- Bench setup: carrier on an ESC load rig with battery emulator, GNSS signal path through a calibrated splitter, provisioning from the dashboard — documented, under thirty minutes.
- Functional campaign: dual-IMU cross-comparison and noise-floor suites, mag and baro verification, GNSS acquisition and C/N0 suites per constellation, all-eight DShot output and telemetry runs, gimbal power-limit and control checks, RC-link and SBUS-failsafe timing, BMS cell-telemetry and shunt accuracy.

# Test - IV&V

- Compute and memory campaign: DRAM pattern stress across the –10 °C to +50 °C span, cache and DMA integrity under full perception load, and the STM32's 2 kHz loop-period distribution measured with the Jetson saturated — the control loop's worst case is the number that flies.
- Watchdog and fault-injection campaign: induced Jetson hangs and resets confirming the STM32 retained failsafe and landing authority through the hardwired discrete, arming interlocks exercised through every documented state combination, and the inter-processor SPI/UART links driven through corruption, stall, and reconnect cases without control-loop disturbance; JTAG/SWD on the STM32 was verified locked in the flight configuration, retained on first articles for trace-anchored loop-timing evidence.

# Test - IV&V

- One-shot tests as the acceptance gate: one pass/fail per subsystem per flight-software candidate.
- Long-running: the 72-hour perception-load endurance caught NVMe thermal throttling propagating back-pressure into the camera pipeline above 45 °C — a heatsink revision noted for the airframe team.
- Event-based: silent runs recording on IMU disagreement, C/N0 drops, DShot telemetry gaps, rail excursions, and failsafe triggers — this mode caught the SBUS failsafe asserting 90 ms late when the RC receiver browned out rather than disconnecting, a distinct failure path the ICD had not separated, fixed in the STM32 supervision logic requirements.

# Test - IV&V

- Monitoring: passive telemetry — rails, temperatures, sensor spectra, C/N0 — under combinatorial subsystem load; the main finding surfaced here: GNSS C/N0 dropped 7 dB whenever the USB 3.0 camera port ran at full rate, 5 Gbps harmonic desense from the carrier's own routing beside the RF front end, confirmed in the RF chamber and corrected with rerouting and shielding in the revision.
- Instrumentation correlation: spectrum-analyzer, oscilloscope, and shaker-rig captures anchored the desense, vibration, and battery-sag findings — the vibration campaign also validated the IMU-mount BOM finding, with the stiffer pad measurably raising the noise floor in the 120–180 Hz band the props excite.

# Test - IV&V

- 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 is the bench tool.
- Traceability: requirements traced bidirectionally to tests and records, schemas version-pinned, runs reproducible for the flight-readiness review.

9

# Deliverables

ROBOTICS DRONE NVIDIA JETSON

PRODUCT

# Deliverables

The client received the complete platform fitted to this flight core — source, records, procedures, no lock-in.

- Validation OS image with the full Yocto build system, recipes, and per-build SBOM.
- Carrier-specific BSP and HAL source for every subsystem, maintained against the design package into the revision.
- Compiled client for the Jetson plus the STM32 Zephyr build.
- Server and dashboard as deployment-ready containers on the client's bench.

# Deliverables

- Complete suites — sensor noise, GNSS desense matrix, DShot, failsafe timing, vibration-monitored, battery-sag, endurance — with recorded first-article baselines.
- Python SDK with documentation, example scripts, shaker-sync bindings, and CI integration gating every flight-software candidate.
- The append-only records store as client property — the evidence base for flight-readiness and insurance review.
- The independent validation report: the USB 3.0 GNSS desense, the SBUS brown-out failsafe gap, the IMU-mount BOM substitution, the NVMe throttling note, metrics, pass/fail, coverage.
- Bench procedures for acceptance and re-runs; full documentation; no lock-in.

# Conclusion

ROBOTICS DRONE NVIDIA JETSON

# Conclusion

The flight core reached its readiness review with its worst behaviors already measured and designed out: a navigation receiver degraded by its own camera port, a failsafe that arrived late in exactly one failure mode, and a sensor mount quietly substituted with a noisier part.

- The desense finding alone would have cost a flight-test season — degraded GNSS in the field looks like software, weather, or luck, and only a bench matrix of subsystem combinations pins it to a routing decision; it cost a reroute instead of a fleet grounding, with chamber-anchored records behind it.

# Conclusion

- The suites gate every flight-software drop in CI and carry to the carrier revision with only the HAL delta re-verified. We produce the evidence — the client's flight-test organization and regulators judge it.

# Lessons Learned

ROBOTICS DRONE NVIDIA JETSON

# Lessons Learned

The engagement made combinatorial self-interference testing a fixture of our UAV pipeline.

- Aggressor-victim matrices — every digital subsystem exercised against every RF and analog victim — now run before functional depth on any vehicle board; the USB 3.0/GNSS coupling appeared only at full camera rate, and single-subsystem testing would have certified a defective carrier, just as failsafe testing must distinguish disconnect from brown-out because this board treated them identically until measured.

# Lessons Learned

- Mechanical items on the BOM deserve the same schematic cross-check as electrical ones: the IMU mount substitution was a sourcing decision nobody flagged, with consequences measurable only on a shaker — mechanical-part verification against design intent is now part of our document review on every sensor-carrying board.

PRODUCT

LET'S TALK

# SiliconCentric


EMAIL

sandesh@siliconcentric.com

PHONE

(669) 356-1998

WEB

https://www.siliconcentric.com

ADDRESS

San Jose, CA, USA

# Robotics - Drone

## Robotics Drone Nvidia Jetson

Built by SiliconCentric

SiliconCentric