PRODUCT

Retail Kiosk

Banking Retail Retail Kiosk Raspberry Pi

SiliconCentric | Independent Verification & Validation

I

PRODUCT

Introduction

Our client is a retail-systems integrator. Their product is a self-service payment kiosk — hardware that fails in public, in front of a queue, mid-transaction. They were mid-development — carrier rev B in progress, a fleet purchase order pending its results — when we collaborated with them on this project by providing independent verification and validation of the board. The design pairs a Raspberry Pi CM4 with a dedicated Raspberry Pi Pico W companion that supervises power rails and peripheral health independently of the application processor.

- Compute topology: CM4 plus external Pico W companion; the CM4 runs the kiosk application and peripheral stack, the Pico W supervises rails, watchdogs the CM4, and sequences peripheral power.
- Under validation: a 15-inch projected-capacitive touch panel over USB, an EMV contact/contactless card reader on USB with PCI-DSS-adjacent handling constraints, a thermal receipt printer on USB with a high-inrush 24 V motor rail, a 2D barcode scanner on USB, four load-cell weight sensors through an HX711-class ADC for bag detection, a PoE++ (802.3bt) power stage, GbE, and eMMC.
- Program state: mid-development — rev A carriers on the bench, rev B layout open, fleet order gated on validation results.
- Independence: no technical, managerial, or financial stake in the design; the acquirer bank's technical reviewers explicitly preferred third-party evidence.
- Bounded scope: independent validation only — no kiosk-application work, no payment-flow development, no firmware authoring.
- Working inputs: the hardware design package — schematics, ICDs, BOM — with no access to firmware source.

2

PRODUCT

Hardware Topology

The topology is SoC-plus-companion-uC with a reliability motive: the Pico W exists so the kiosk can detect and recover a wedged peripheral without a truck roll. Scope enclosed the carrier, its peripheral complex, and its power architecture; the payment application, EMV kernel certification, and PCI assessments stayed outside.

- Topology classification: CM4 plus external Pico W, interconnected by UART for health traffic and I2C for power-rail telemetry, with the Pico W holding per-peripheral power-switch GPIOs and a hardware watchdog on the CM4.
- The Pico W's job: rail supervision, per-peripheral power cycling on fault, CM4 watchdog with staged recovery, and tamper-line monitoring on the enclosure switches.
- In-scope interfaces: USB 2.0 x4 (touch, EMV reader, printer, scanner), GbE, eMMC, I2C x2 (rail telemetry, ambient sensors), the HX711-class load-cell chain, PoE++ stage with 51 W budget, enclosure tamper GPIOs.
- Envelope in scope: 0 °C to +40 °C inside a sealed kiosk head warmed by its own display, 24/7 retail duty, and the power quality of store back-office wiring.
- Out of scope: EMV level-1/2/3 certification, PCI-DSS assessment, kiosk application software — we produce integration evidence; the client's acquirer, labs, and assessors judge it.
- The Raspberry Pi family 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

PRODUCT

Problem Statement

Rev A demoed well, the fleet purchase order sat pending, and every peripheral vendor's compliance sheet was in a folder — while nothing proved the assembled board survived its own peripherals interacting. Payment kiosks compound ordinary embedded risk with money: a hardware fault mid-transaction is a dispute, a refund, and a merchant complaint at once.

- Proven in isolation, never as a board: the EMV reader carried lab certifications and the printer vendor published load specs — but this carrier's USB tree under simultaneous peripheral load, its 24 V inrush behavior, its PoE++ budget arithmetic, and its load-cell analog chain beside a switching printer rail had never been exercised together.
- Firmware coverage is feature-shaped, not board-shaped: the application exercised the normal transaction path daily, but concurrent printer-plus-scanner-plus-touch load, the weight-sensor tare drift, the Pico W's staged recovery ladder, and the tamper lines fell outside the sprint tests.
- No stress, no extremes, no duration: no 40 °C sealed-head soak through a simulated retail day, no multi-week 24/7 endurance, no brown-out campaign against store wiring events, no transaction-storm testing at queue-rush rates.
- Nothing runs alongside development: validation was an end-gate, so a USB dropout under load would ship into the fleet and surface as intermittent "reader not found" tickets — the most expensive support category a kiosk fleet has.
- The acquirer's technical review demanded documented reliability evidence before the fleet order; PCI-DSS-adjacent handling constrained how the reader's power and data paths could behave under fault.
- The 51 W PoE++ budget had never been audited against worst-case simultaneous draw with printer inrush on top.
- A fleet-wide hardware defect discovered post-deployment multiplies by every store in the estate — the integrator carried that risk contractually.

PRODUCT

Customer Requirements

4

The client's requirements were direct: validate the whole board, under stated limits, before the fleet order, with evidence shaped for the acquirer's review.

- Operating temperature: the board shall operate from 0 °C to +40 °C inside the sealed kiosk head, 24/7, including self-heating from the display.
- Supply: PoE++ (802.3bt) input; total draw shall not exceed the 51 W class budget under worst-case concurrency including printer inrush. The 5 V distribution shall not sag below 4.75 V during printer motor inrush; the 24 V printer rail shall be isolated from the logic rails.
- Brown-out and power interruption: the board shall survive store-wiring brown-outs and AC interruptions with no eMMC corruption and no undefined peripheral state; on power loss the Pico W shall record the event with no record loss.
- Over-current: each USB peripheral port shall have a per-port power switch with the specified 2.1 A limit on the printer port; a shorted or wedged peripheral shall be power-cycled by the Pico W without disturbing the EMV reader's session.
- Over-voltage: the PoE stage shall protect downstream rails against input transients per 802.3bt; over-voltage on any rail shall shut the affected domain within 10 ms.
- Humidity: the board shall operate at up to 85% RH non-condensing and shall not be exposed beyond that.
- Full-board functional validation: the CM4's cores and memory, eMMC, all four USB peripherals, GbE, the load-cell chain, the PoE++ stage, both power paths, the tamper lines, and the Pico W supervision loop — one integrated system.
- Coverage independent of firmware scope: every subsystem validated regardless of the application's transaction flow, through a HAL built from the design package.
- Stress matrix: 0–40 °C sealed-head sweep, three-week 24/7 endurance at simulated queue-rush rates, brown-out and power-interruption campaigns, PoE++ budget audit under worst-case concurrency with inrush capture, and load-cell drift characterization — vibration waived, ESD deferred to the compliance lab.
- Continuous validation alongside development: CI integration revalidating every firmware candidate on rev A, then rev B.
- Evidence: timestamped, operator-attributed, version-pinned records with bidirectional traceability, packaged for the acquirer's technical review and the integrator's fleet-contract obligations.
- A web console the integrator's QA staff run without embedded expertise.
- An immutable, append-only evidence store anchoring the fleet-order decision and future revision baselines.
- Setup in minutes: documented bench — carrier, peripheral set, PoE++ source, programmable AC path, host — reproducible by any QA operator.

PRODUCT

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 — necessary because USB-tree behavior under concurrent peripheral load was the central question, and heavy on-target tooling would have been its own USB and CPU load.

- Polaris OS (the validation OS) — a reproducible Linux image for the CM4 booting to a known state, one baseline across rev A and rev B carriers.
- HAL — the single board-specific layer, exercising the USB tree, GbE, eMMC, the load-cell chain, rail telemetry, tamper lines, and the Pico W link behind a uniform interface that keeps the retail suites portable across revisions.
- Client — the on-target agent driving tests through the HAL; gRPC on the CM4, FlatBuffers to the Pico W's FreeRTOS image; three timestamps per event (origin, relay, server); light enough that USB scheduling under test reflects the product.
- Server — orchestration, scheduling, and the append-only records store on the host, six carriers in parallel, dashboard and API hosted.
- Dashboard — the QA console: provision a carrier, configure and launch, live USB device states, rail telemetry, weight-channel readings, PoE draw, time-series review; every state-changing action operator-attributed.
- Python SDK — automation and CI; scripts everything the console does; C bindings available, unused here.
- The OS and HAL are the two board-specific layers and were heavily modified for this carrier and this silicon; 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

The binding constraints: four USB peripherals that must never interfere, a 51 W ceiling with a motor on the wrong side of it, a sealed head that heats itself, and an evidence standard set by a bank rather than an engineering team.

- Validation OS design — read-only rootfs with A/B rollback, because a fleet image that cannot roll back is a fleet incident; USB device-class drivers were pinned to validated versions — the trade-off being drift from upstream, accepted for reproducibility. The Pico W runs FreeRTOS with the recovery ladder modeled as an explicit state machine.
- HAL design — partitioned by schematic sheet against ICD clauses; USB events, tamper edges, and rail excursions are interrupt-driven while the load-cell chain polls at 10 Hz. HX711-class parts are sensitive to readout timing — irregular polling injects apparent weight noise — solved with a dedicated timer-driven read.
- Client design — origin timestamps at the driver boundary; transaction-storm campaigns replay scripted peripheral sequences with timing assertions, and on power events the client flushes before the Pico W's staged recovery acts, trading recovery-latency purity for complete records.
- Server design — three-timestamp causality orders a printer inrush against the USB dropout it caused; retention sized for three-week 24/7 streams from six carriers, with per-transaction peripheral traces kept raw through the acquirer review.
- Dashboard design — USB device-state matrix, rail waterfall, PoE draw against budget, and weight-channel drift are first-class views — what a kiosk QA lead triages; run health and drop counts on every screen.
- SDK design — the client scripted the acceptance gate, transaction-storm profiles, and the concurrency matrix; the direct API path served single-peripheral 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 QA workflow while their structure and working style stayed constant.

7

PRODUCT

Implementation

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

- Validation OS — Yocto image for the CM4: inherited BSP and bootloader, carrier device tree covering the USB tree, load-cell chain, tamper bank, and rail telemetry, minimal kernel with pinned USB stack; OTA with A/B rollback; boot under 25 seconds; per-build SBOM with CVE tracking; the Pico W FreeRTOS image built in the same tree.
- HAL — implemented sheet by sheet and bench-verified as written: USB-tree instrumentation with per-port event capture, timer-driven load-cell readout, tamper-edge handlers, rail-telemetry drivers, Pico W recovery-ladder command interface; the design-package pass surfaced a schematic-to-BOM mismatch — the printer port's downstream USB power switch specified a 2.1 A current-limit part in the schematic while the BOM carried a 1.1 A variant, guaranteeing spurious trips under printer load before any test ran.
- Client — C++ daemon, event and polling paths, storm-replay engine, power-event-hardened shutdown, gRPC upstream, FlatBuffers to the Pico W.
- Server — telemetry receiver, versioned schema, scheduler, REST API, dashboard hosting, six-carrier parallelism.
- Dashboard — carrier provisioning, campaign configuration, live USB and rail streaming, time-series review, export for the acquirer review, operator attribution.
- SDK — Python bindings, example scripts per peripheral, CI wiring firing the acceptance suite on every firmware candidate.
- Board-specific: OS delta and HAL. Configuration: client storm profiles, server retention, dashboard views, SDK examples.

PRODUCT

Test - IV&V

8

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, boot integrity checked on accept and reject paths.
- Bench setup: carrier with the full peripheral set in a representative head, PoE++ source and programmable AC interruption path, provisioning from the dashboard — documented, under thirty minutes.
- Functional campaign: touch-report integrity suites, EMV reader enumeration and session-persistence suites, printer job and cutter cycles, scanner decode-rate runs, weight-channel accuracy and tare-drift suites, GbE throughput, tamper-line verification, Pico W recovery-ladder exercises.
- CM4 compute and memory checks: DRAM pattern stress and eMMC endurance profiling under the 24/7 duty model, CPU thermal-throttling behavior characterized in the self-heating sealed head, and boot-path verification across repeated power interruptions.
- Security posture: enclosure tamper lines exercised through open, short, and slow-transition cases, boot-chain integrity verified on accept and reject paths, and debug-port access confirmed disabled in the fleet configuration — an open serial console on a payment kiosk is an acquirer finding waiting to happen.
- Inter-processor link robustness: the CM4↔Pico W UART and I2C traffic driven through framing corruption, heartbeat timeout, and retry paths, confirming the watchdog and recovery ladder act on real faults rather than link noise.
- One-shot tests as the acceptance gate: one pass/fail per subsystem per firmware candidate.
- Long-running: the three-week 24/7 endurance at queue-rush transaction rates caught weight-channel tare drifting with head temperature — a load-cell amplifier grounding issue corrected in rev B — and printer-cutter current signatures degrading measurably by week two on one mechanism, flagged to the vendor.
- Event-based: silent runs recording on USB disconnects, rail excursions, tamper edges, and watchdog events — this mode caught the main defect: the EMV reader dropping off the bus for 400 ms whenever a receipt printed, traced to printer motor inrush sagging the 5 V rail through shared distribution, exactly the mid-transaction failure the acquirer feared; rev B split the distribution and added bulk capacitance.
- Monitoring: passive telemetry — rails, head temperature, PoE draw, USB error counters — through simulated retail days; the PoE audit produced its own hard number: worst-case concurrent draw with printer inrush reached 54 W against the 51 W class budget, a spec violation masked in bench testing by the USB-C debug supply, fixed by re-sequencing printer power under Pico W control.
- Instrumentation correlation: oscilloscope and current-probe captures anchored the inrush, rail-sag, and budget findings; barcode-scanner enumeration raced the touch panel on cold boot in roughly 1 in 40 starts, captured on the bus analyzer and fixed with hub-port power sequencing.
- 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 acquirer's reviewers.

9

PRODUCT

Deliverables

The integrator received the complete platform fitted to this carrier — 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 through rev B.
- Compiled client for the CM4 plus the Pico W FreeRTOS build.
- Server and dashboard as deployment-ready containers on the integrator's QA bench.
- Complete suites — peripheral functional, concurrency matrix, transaction storm, endurance, power-interruption, PoE audit — with recorded rev A baselines for rev B comparison.
- Python SDK with documentation, example scripts, and CI integration gating every firmware candidate.
- The append-only records store as client property — the evidence base for the acquirer review and fleet contract.
- The independent validation report: the EMV dropout mechanism, the PoE budget violation, the USB power-switch BOM error, the tare-drift grounding issue, the enumeration race, metrics, pass/fail, coverage.
- Bench procedures for acceptance and re-runs; full documentation; no lock-in.

10

PRODUCT

Conclusion

The fleet order proceeded against a rev B whose transaction-killing defects had already been found, measured, and fixed: a reader that vanished when receipts printed, a power budget that only held under the debug supply, and a current-limit part that was never the one on the schematic.

- The EMV dropout was the finding the fleet contract turned on — intermittent, load-dependent, invisible to feature testing, and certain to generate dispute-grade failures across an estate; it cost a distribution split and capacitors instead of fleet-wide service tickets, with bus-analyzer records the acquirer's reviewers read directly.
- The suites gate every firmware drop in CI and carry to rev B with only the HAL delta re-verified. We produce the evidence — the client's acquirer, labs, and assessors judge it.

II

PRODUCT

Lessons Learned

The engagement changed our defaults for multi-peripheral USB systems and for any bench powered differently than the field.

- Concurrency is the test, not the peripheral: every USB device passed alone and the failure lived only in the printer-plus-reader combination, so scripted worst-case concurrency storms now precede single-peripheral depth on any kiosk-class board — and validation benches must power the board exactly as the field does, because the USB-C debug supply hid a hard PoE budget violation for weeks.
- The CM4/Pico W recovery ladder needed adversarial validation of its own: the Pico W correctly power-cycled a wedged printer but its staged ladder rebooted the CM4 for a fault the printer caused, taking a healthy kiosk down with its sick peripheral — fault-attribution logic in supervisor microcontrollers is now something we test with injected faults on every supervised design.

CONTACT

SiliconCentric

EMAIL sandesh@siliconcentric.com

PHONE (669) 356-1998

WEB https://www.siliconcentric.com

ADDRESS San Jose, CA, USA