ORBITUS ROBOTICS / WHITE PAPER

Replacing a robot's host controller while preserving its actuators

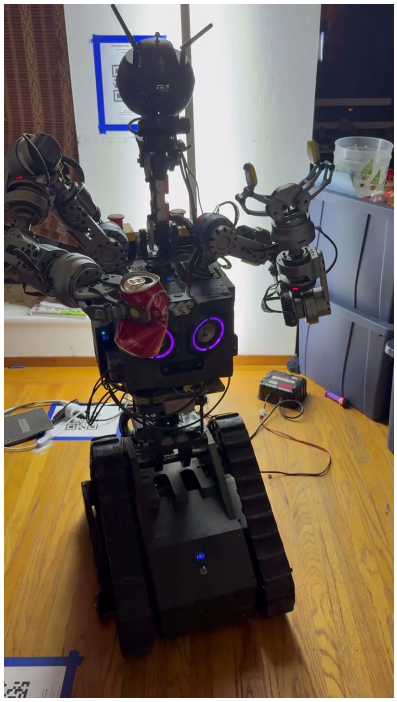
Dependency forensics, protocol compatibility and a single-writer architecture

OR-WP-2026-002 | Version 1.1 | November 4, 2026

Orbitus Robotics · Project lead: Rodolfo Aramayo

Abstract

A robot controller can use a capable trajectory library yet still mishandle ownership of motor commands. We examined ROB's installed Amber host executables, their dependency graph, command handlers and wire encoding without modifying the running controller. The audit found seven direct shared-library dependencies and 42 mapped shared objects, but no matching current wrapper source within the inspected scopes. Static analysis identified detached trajectory workers and competing writes to shared commanded state. We describe a replacement host boundary that preserves actuator firmware and native references while centralizing target ownership, feedback supervision and output serialization. A transport-free codec passed recorded tests with 42,000 frames and exhaustive signed-boundary checks. The paper distinguishes observed binary behavior, engineering interpretation and proposed deployment work. It provides a reusable audit method, not a claim that the replacement has been commissioned on physical motors.



RESEARCH RECORD

A source-bound audit of the Amber host executable, its 42 mapped libraries, and an offline-tested compatibility boundary.

Evidence originates in the 2026-11-02 report. Technical white paper · Not peer reviewed.

ROB holding a can on October 29, 2026. Still from Kierie's video. The visible mechanisms motivate the host/actuator boundary; this image predates the replacement implementation.

Keywords: robot controllers, CAN, binary analysis, software architecture, fault containment

ROB: history, purpose and software

ROB (R.O.B.) is Orbitus Robotics' experimental tracked humanoid: a platform for robotics education, human-robot interaction and supervised manipulation. Independent treads, a rotating torso, two seven-joint arms and grippers, cameras and range sensing connect software to a physical machine. Can-handling practice, stage demonstrations and spatial teleoperation give the project concrete tasks to investigate. [7,8,9]

The project's archive reconstruction records ROB-specific Mac serial-control work in 2017-2018 and a surviving Cerebro Git root on January 1, 2018. Later histories add speech, perception experiments and controller arbitration. Public galleries document builds and appearances from 2019 onward. The fresh Cerebro repository of August 5, 2025 imported an existing v5 application; it was not the project's beginning. The pre-v5 originals remain local, with provenance summarized in the supporting record. [7,9]

Codebase / platform	Role in ROB
Cerebro / macOS	Swift, Objective-C/Objective-C++, C++ and Python helpers coordinate perception, dialogue, command admission and robot I/O.
ROBController / iPhone, iPad, Watch	Human command, maps, telemetry and explicit action approval.
ROBControllerVision / visionOS	SwiftUI spatial controls, live video and time-limited arm authority.
ROBArduino + RPLidar project	Base firmware handles low-level drive; the lidar application supplies scans and map observations.
Amber host + ContinuousArm	Existing Ubuntu arm runtime; the C++ replacement owner and codec live in Cerebro/Tools/ContinuousArm.

The public source index links six repositories and suggests starting files. The architecture separates human intent and AI-assisted planning from motor-facing ownership. This paper examines that latter boundary. It does not establish that every project feature is active together, or that the replacement controller has been physically commissioned. [7,8]

1. Scope and audit method

The replacement question is narrower than rebuilding an entire robot: can the Ubuntu host role be replaced while retaining the motor firmware, joint references and existing transport? The original November 2 report separates that engineering route from a motor-ready release. This white paper preserves its dated findings and links later readiness evidence through the companion takeover paper.

The audit read the installed executable identity, mapped libraries, ELF dependency entries, imported symbols, launch configuration and selected disassembly. The dependency snapshot was captured at 05:42 UTC on November 3, corresponding to the November 2 local report date. Both installed arm cores matched the same archived executable hash. No binary attachment, patch, motor command or runtime configuration change was needed for these observations. [1,2]

Audit result	Recorded observation	Limit of the claim
Core identity	Matching ELF SHA-256 and build ID	Identifies the audited build only
Dependencies	7 direct libraries; 42 mapped shared objects including loader	Mapping does not prove a command exercised a library
Graph and imports	43 nodes including executable; 141 dependency edges; 235 core imports	Static linkage is not a runtime call trace
Source search	78,325 filenames; 30,572 C/C++ files; no matching current wrappers found	Inaccessible/private locations were excluded

The full mapped-library ledger is supplied as a public CSV and JSON with filenames, identities and linkage labels. It preserves the audit's useful inventory without redistributing vendor executables or publishing machine-specific directory structure.

2. The failure belongs at the ownership boundary

Static control flow shows a position-task handler launching a detached Ruckig worker for a new target. A worker can validate cancellation and later write the shared position array without an indivisible ownership/publication operation. Separate finite-difference samples seed velocity and acceleration. A second target can therefore introduce both inconsistent state and a stale writer. The disassembly supports this hazard; it does not prove the exact scheduling or internal values during a past physical incident. [1]

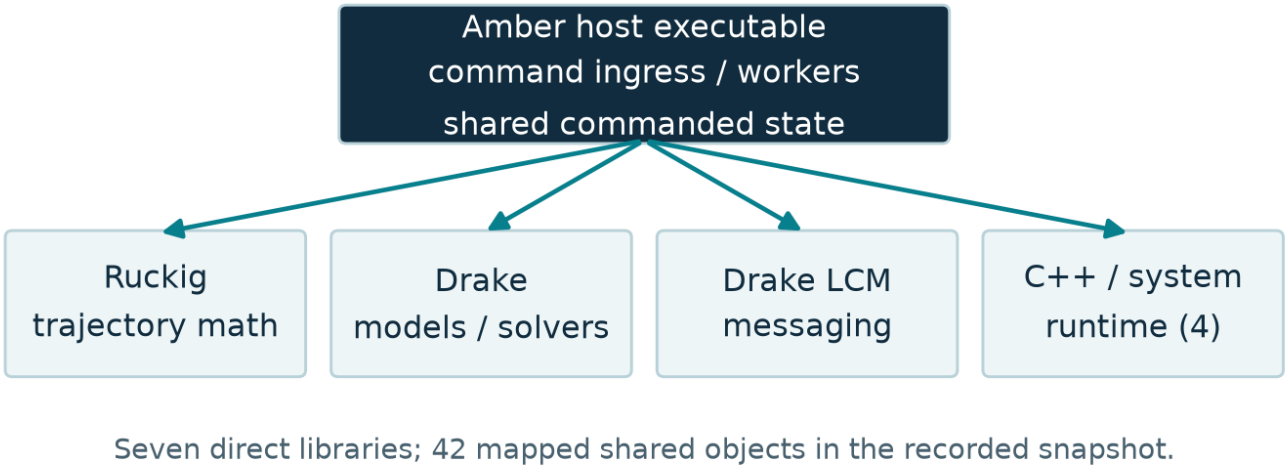


Figure 1. The host depends directly on trajectory math, modeling, messaging and runtime libraries. The replacement changes motor-facing ownership; swapping a shared library alone does not remove competing writers.

Ruckig supplies motion math. The host executable owns command ingress, threads, shared targets and output. Drake supplies model/solver facilities in other paths; direct joint command 4 bypasses that solver. The loaded Mosek and graphics branches are dependency observations, not evidence that a given arm command invoked optimization or rendering. A source filename suggesting a ROS interface likewise does not establish a ROS 2 node; the observed position handler uses LCM.

A matching-source rebuild would be attractive if the exact wrappers, generated schemas and build inputs became available. In the inspected scopes, stale objects did not establish a reproducible current build. Reusing those objects, swapping a .so, or injecting raw position messages into a still-running legacy core would leave ownership and ABI assumptions unresolved.

3. Reproduce the encoding; preserve uncertainty about semantics

The host position encoder emits seven three-byte frames per sample: CAN identifier 0x10 + joint number for joints 1 through 7, then opcode 0x40 and a big-endian 16-bit position word. The SI adapter converts native radians to degrees, applies the configured sign and truncates at 128 counts per degree. The observed negative-value convention adds 65,535 rather than 65,536. [3,4]

Compatibility example	Observed host result	Engineering consequence
+1 degree	40 00 80	One count is 1/128 degree
-1 degree	40 FF 7F	Preserve the legacy negative encoding
Raw 0x8000	Decodes as +32768	Reject negative count -32767; it would flip sign
Raw 0xFFFF	Decodes as zero	Validate exact encode/decode round trips

The codec explicitly rejects non-finite values, invalid signs, joint-bound violations and unrepresentable signed round trips. Seven frames are assembled atomically in memory. They are not an atomic physical transaction: a real transport may accept only part of the batch. Partial or ambiguous output therefore needs a fault policy rather than an automatic retry.

For eight-byte feedback, the inspected host reads position from bytes 1-2 at 1/128 degree and current from bytes 5-6 with a divisor of 256. Velocity units, current physical units and the complete status-bit contract remain unverified. Byte 0 is retained as opaque diagnostic information; byte 7 remains raw status. Feedback IDs 0x91-0x97 correspond to the seven arm joints. Gripper and mode operations stay outside the continuous position stream.

> Protocol compatibility is evidence about the inspected host implementation. It is not independent validation of drive firmware, a motor rating, a Stop contract or permission to send these frames.

4. A single owner from ingress to publication

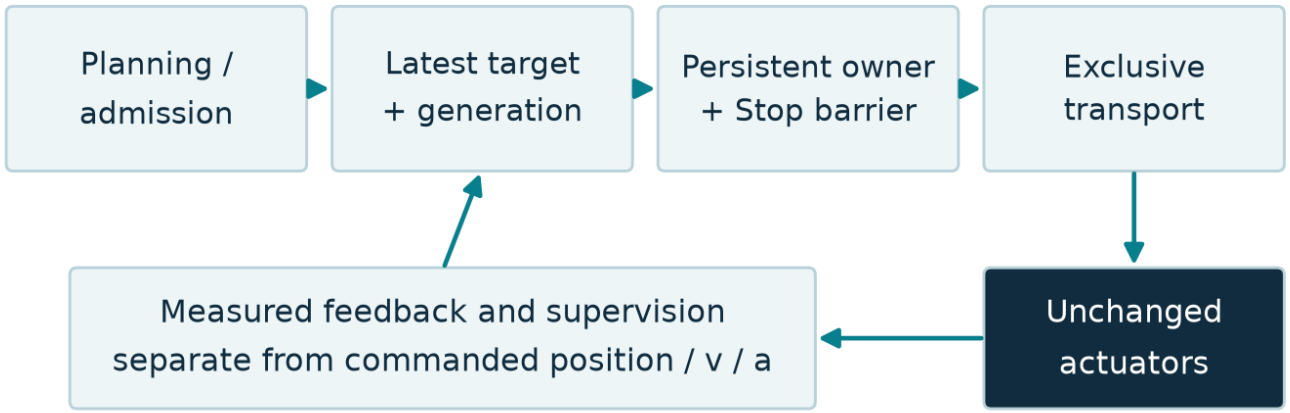


Figure 2. Proposed separation of planning, ownership and transport. Feedback is observed independently from commanded state. Physical integration and stop commissioning were open in the source report.

The target contract identifies the physical arm, native-radian units, sequence, generation and a host-validated lease. A persistent owner keeps coherent commanded position/velocity/acceleration and replaces the latest pending target. Output serialization includes the Stop/ownership decision. A transport boundary accounts for per-joint receive time, publication deadlines and uncertain writes. [6]

Exclusive ownership must include every writer and every path that can restart one. A cooperative application lock or one successful SocketCAN bind does not establish exclusivity: Linux permits multiple sockets on the same CAN interface and identifiers. The proposed production handoff removes legacy writers and controls startup/recovery, CAN access and client IPC together. [5]

Physical naming is an explicit part of the contract: ROB's RIGHT is L-10/CAN10, while LEFT is R-11/CAN11. Vendor labels are not a reliable substitute for a measured physical mapping. Firmware and stored calibration remain outside this replacement scope.

Fault containment has distinct layers. If the process can still publish, it can attempt a bounded brake from coherent commanded state. If output or feedback becomes uncertain, it latches a fault and discards old targets. If the process, bus or power is lost, an independent drive-side timeout and energy-handling strategy must be demonstrated. A software state named Held does not prove that a gravity-loaded arm is physically supported.

5. Recorded results and the remaining experiment

November 2 test record	Result	Evidence class
Owner + codec groups	19 + 8 groups pass Release, ASan/UBSan and TSan	Offline regression
Recorded-output replay	6,000 ticks; 42,000 frames; 1,358 moving retargets	Synthetic, ideal 20 Hz feedback
Quantization	Maximum error 0.000136344 rad; maximum wire step 0.00163625 rad	Encoded commands, not tracking
Boundary sweeps	917,476 full-domain round trips plus 420,014 quantization-sweep checks	Codec compatibility

The corrected codec is pinned to commit 9e054156e0fe29cc5cfde8e785eb3a1f6e0489e2. Its replay records contain hardware_io=false and installed=false. The 42,000 frames were recorded in memory, not transmitted to the robot. The observed quantization step is not a measurement of smoothness, torque or endpoint accuracy. [4]

At this stage the project had a persistent owner and inert codec, not a commissioned replacement daemon. Later Linux, virtual-CAN and inactive-host rehearsals are described separately in the takeover paper. Counting those later checks as physical validation would blur a boundary the project deliberately retains.

• Establish fresh per-joint feedback, verified units and measured position/rate/current limits.

• Prove exclusive ownership across process, serial adapter, CAN namespace and restart paths.

• Measure adapter queue behavior, write timing, partial-batch outcomes and transport age.

• Commission supported Stop, process/bus-loss behavior and full robot/world clearance before enabling physical streaming.

> Conclusion: preserve actuator compatibility while replacing the host responsibility that is actually defective. A dependency audit tells us where components sit; source-bound replay tells us what a new component does under tested inputs. Neither replaces motor commissioning.

6. Reproducibility and the library ledger

The audit is pinned to bfed885d51f36fb6f529fe68043e48f1f9b6cb4d, and codec results to 9e054156e0fe29cc5cfde8e785eb3a1f6e0489e2. Reproduction starts with the recorded identity and dependency data, then the isolated ContinuousArm build described in the source README. The public supplements contain all 42 mapped-library identities and curated test metrics. Hashes make revisions comparable; they do not certify behavior.

The ledger labels a library Direct when its SONAME appears in the core's DT_NEEDED list. Other mapped objects are labeled Mapped/transitive; this category includes the dynamic loader. It does not assert that every object has the same calling path or that an imported symbol executed. Full dependency edges remain available in the pinned project record.

This white paper adapts the fourteen-page November 2 engineering report. It excludes operator conversation, credentials, host paths and proprietary binary payloads. The original observed byte behavior is documented as an interoperability finding, with unresolved physical semantics called out rather than guessed.

References and supporting records

[1] Installed executable audit and source-search scope (bfed885)

https://github.com/RudyAramayo/Cerebro/tree/bfed885d51f36fb6f529fe68043e48f1f9b6cb4d/docs/calibration/evidence/2026-11-02-amber-executable

[2] Shared libraries and imports, full graph

https://github.com/RudyAramayo/Cerebro/blob/bfed885d51f36fb6f529fe68043e48f1f9b6cb4d/docs/calibration/evidence/2026-11-02-amber-executable/shared-libraries-and-imports.json

[3] Host protocol evidence (1e2f52b)

https://github.com/RudyAramayo/Cerebro/tree/1e2f52b95f706efb1ecc58c0db32b65983f193ec/docs/calibration/evidence/2026-11-02-amber-protocol

[4] Source-bound owner and codec regression records

https://github.com/RudyAramayo/Cerebro/tree/9e054156e0fe29cc5cfde8e785eb3a1f6e0489e2/docs/calibration/evidence/2026-11-02-amber-codec

[5] Linux kernel: SocketCAN networking documentation

https://docs.kernel.org/networking/can.html

[6] Controller source and reproduction instructions (9e05415)

https://github.com/RudyAramayo/Cerebro/tree/9e054156e0fe29cc5cfde8e785eb3a1f6e0489e2/Tools/ContinuousArm

[7] ROB project chronology and public source index (v1.1)

https://www.orbitusrobotics.com/whitepapers/rob-project-background.json

[8] ROB software platform and operator interfaces

https://www.orbitusrobotics.com/products/

[9] ROB development and Maker Faire photo archive

https://www.orbitusrobotics.com/makerfaire/

Web edition and supplements: https://www.orbitusrobotics.com/whitepapers/amber-controller-replacement-analysis/

Version 1.1 adds ROB history, purpose, functionality and a software-stack overview. Recorded test results and physical commissioning status are unchanged.

Prepared with AI assistance from project reports, source code and recorded test evidence. Technical claims are tied to the cited records. No new physical robot trial was performed for this publication.

Photography: Kierie, October 29, 2026. Diagrams: Orbitus Robotics. Correspondence: orbitus@orbitusrobotics.com