GLOW 2025: Serial Router Firmware acceptance test & integration

Artur Kraskov
Semester 7
ICT & OL, Delta
Fontys 2025-2026

Contents

Introduction

Current document is complementary to GLOW 2025: Serial Router Firmware Analysis & Advice. It provides details about the acceptance test and integration.

Assumptions & Constraints

- **Message framing:** start token `!!`, field delimiter `:`, terminator `##`.
- **Command verbs:** e.g., `REQUEST`, `CONFIRM` per current agreement; additional verbs TBD.
- **Baud rates:** USB CDC (Mac link) at 115200; UART links default 9600 unless otherwise specified by device.
- **Electrical:** UART TTL levels compatible with Teensy 4.1; optional GPIO handshakes may be introduced (TBD).

System Under Test (SUT)

Role and Responsibilities

The Teensy 4.1 receives framed commands, validates structure, determines destination, forwards over the appropriate UART, and relays responses to the Mac Mini. Where specified, it transforms `REQUEST` to `CONFIRM` for auto-acknowledge flows.

Architecture Overview

- **Upstream:** Mac Mini via USB CDC serial (`Serial`).
- **Downstream UART links:** five ESP endpoints and a centerpiece controller.
- Routing logic encapsulated via CmdLib (named-parameter variant) and firmware in `main.cpp`.

Interface Inventory (Teensy 4.1)

- **USB:** `Serial` (Mac link)
- **UART:** `Serial1` (Centerpiece), `Serial2` (Arm2), `Serial3` (Arm3), `Serial4` (Arm4), `Serial5` (Arm5), `Serial6` (Arm1)

Table 1

Logical Port	**Teensy Serial**	**Default Baud**	**Pins (T4.1)**
mac	Serial	115200	USB port
Centerpiece	Serial1	9600	RX1=0, TX1=1
Arm1	Serial2	9600	RX6=25, TX6=24
Arm2	Serial3	9600	RX2=7, TX2=8
Arm3	Serial4	9600	RX3=15, TX3=14
Arm4	Serial5	9600	RX4=16, TX4=17
Arm5	Serial6	9600	RX5=21, TX5=20

Note: GPIO handshakes/interrupt lines are currently not mandated by the protocol; placeholders exist to add them if needed.

Communication Protocol Summary (based on Agreement)

The Serial Communication Agreement is the authoritative source for protocol rules; this report verifies conformance to that document without restating it. It defines message framing and delimiters, addressing/routing headers, verb semantics (e.g., REQUEST/CONFIRM), payload constraints and escaping, error codes, and timing/baud specifications used to derive the acceptance criteria in §5 [3].

Verification Approach

Verification follows a layered approach: static checks, bench functional tests, robustness and fault-injection, and performance characterization. Evidence is gathered via USB logs, UART taps, and (optionally) a logic analyzer.

Test Environment and Instrumentation

- **Hardware:** Teensy 4.1, Mac Mini, up to five ESP32 modules (or UART emulators), centerpiece controller/emulator.
- **Software:** PlatformIO/Teensy Loader; Python harness (`Serial_Router.py`); `screen`/CoolTerm; optional logic analyzer.
- **Configuration control:** Git-referenced firmware commit; documented wiring map; baud rates recorded per run.

Entry/Exit Criteria

- **Entry:** Build compiles; device flashes; bench wiring verified; sanity smoke passes (power-on self-log).
- **Exit:** All Acceptance Criteria (§5) met; no open severe defects; rest are reviewed and accepted.

Acceptance Criteria

1. **Protocol Conformance:** All accepted frames adhere to `!!...##`, field structure, and routing header semantics.
2. **Routing Correctness:** Each destination receives frames losslessly and unaltered (except deliberate verb transforms).
3. **Robustness:** Malformed frames and noise do not crash or deadlock the router; subsequent valid traffic proceeds normally.
4. **Performance:** End-to-end latency remains within calculated line time + 20 ms router overhead (unless otherwise specified by subsystem needs).
5. **Observability:** USB logs provide timestamped traces sufficient for incident reconstruction without overwhelming the link.

6. **Safety:** No unintended cross-port broadcast; messages are delivered exactly once to the addressed endpoint.

Acceptance Test Design and Procedures

Each procedure specifies purpose, setup, method, and acceptance checks. Representative procedures are listed; the complete catalog is maintained in the QA tracker and can be expanded.

AT-001: Power-On and Readiness Announcement

- **Purpose:** Verify boot diagnostics and readiness banner.
- **Setup:** Teensy flashed; USB connected; serial monitor open at 115200.
- **Method:** Power-cycle; observe first 5 seconds of output.
- **Acceptance:** Banner present; no stack traces; ports initialized (baud settings logged).

AT-002: Mac→Arm Unicast Routing

- **Purpose:** Confirm unicast delivery to Arm1.
- **Setup:** ESP/emulator on Arm1 (Serial6, 9600); Mac console open.
- **Method:** Send `!![ARM1]:REQUEST:STATUS##` from Mac; observe Arm1 UART.
- **Acceptance:** Frame appears on Arm1 with identical payload; Mac receives any Arm1 response without corruption.

AT-003: Arm→Mac Return Path

- **Purpose:** Validate upstream delivery without mutation.
- **Setup:** ESP on Arm2.
- **Method:** ESP sends `!!MASTER:REQUEST:MAKE_STAR##`.
- **Acceptance:** Mac receives same frame; no duplicates; timestamp delta consistent with line time.

AT-004: Auto-Acknowledge Transform

- Purpose: Verify `REQUEST→CONFIRM` behavior when configured.
- Setup: ESP on Arm3.
- Method: Send `!!MASTER:REQUEST:MAKE_STAR##`.
- Acceptance: ESP receives `!!MASTER:CONFIRM:MAKE_STAR##`; Mac log records action.

AT-005: Centerpiece Addressing

- **Purpose:** Validate directed delivery to centerpiece only.
- **Setup:** Centerpiece on Serial1.
- **Method:** Send `!!CENTERPIECE:REQUEST:SYNC##` from Mac.
- **Acceptance:** Appears on Serial1 only; no frames on other arm ports.

AT-006: Unknown Destination Handling

- **Purpose:** Confirm graceful rejection of unknown routes.
- **Method:** Send `!![UNKNOWN]:REQUEST:TEST##` from Mac.

- **Acceptance:** Router emits error/diagnostic upstream; no crash; router continues processing subsequent frames.

AT-007: Throughput and Backpressure
- **Purpose:** Assess sustained throughput and buffer management.
- **Method:** Send scripted bursts (e.g., 100 frames/min) from Mac to rotating arms.
- **Acceptance:** No buffer overruns; no dropped frames; observed latency within §5 limits.

AT-008: Malformed Frame Recovery
- **Purpose:** Ensure resilience to missing terminator.
- **Method:** Inject frames lacking `##` and then valid frames.
- **Acceptance:** Malformed frames are discarded after timeout; valid frames thereafter are processed correctly.

AT-009: Noise and Line Disturbance (Optional)
- **Purpose:** Characterize behavior under simulated noise.
- **Method:** Introduce random byte noise between frames (emulator); observe.
- **Acceptance:** Parser resynchronizes on next `!!` within bounded time; no wedging.

Measurement, Evidence, and Reporting

- **Evidence:** USB and UART logs with timestamps; optional logic traces.
- **Metrics:** Delivery ratio, duplication rate (should be 0), end-to-end latency distributions, recovery time after malformed frames.
- **Reporting:** Test log with case outcomes, anomalies, and links to artifacts. Defects carry reproduction steps and suspected layer.

Risk Register

- Hardware substitutions may alter UART levels/latency → Validate with each hardware spin.
- Protocol drift across teams → Gate testing on a frozen protocol tag; add compatibility notes when breaking.
- Emulator fidelity vs. real ESP firmware → Test both where possible.

Integration Plan

This section outlines how the Serial Router will be integrated with the broader system. Details (interfaces, calendars, test data) will be populated as subsystem teams finalize their deliverables. To avoid duplication, this plan references the Acceptance Criteria (§5) and Procedures (§6) instead of redefining tests; integration focuses on sequencing, readiness, and governance.

Objectives

- Establish reliable communication pathways between Mac Mini and all microcontroller endpoints via the router.
- Demonstrate end-to-end behaviors required for show operation under nominal and degraded conditions.

Strategy

Adopt a staged, bottom-up integration with simulation first, then progressive hardware inclusion. At each stage, execute the minimal subset of acceptance tests necessary to demonstrate readiness before advancing:

1. Router Module Validation → Smoke subset of AT-001 (boot/readiness) and parser checks (targeted harness).
2. Mac Link Bring-Up → AT-001, AT-003 (upstream return path), logging sanity.
3. Single-Arm Integration → AT-002 (Mac→Arm unicast), AT-004 (auto-ack, if configured) for the connected arm.
4. Multi-Arm Expansion → Repeat AT-002 across added arms; confirm isolation (no cross-talk) consistent with §5.6.
5. Centerpiece Integration → AT-005 (centerpiece addressing) and any timing coordination checks TBD.
6. System Scenarios → Throughput and resilience via AT-007 (throughput/backpressure), AT-008 (malformed recovery), AT-009 (noise, optional).

Entry/Exit per Stage

- **Entry:** Prior stage closed; interface document baseline approved; hardware available or simulator configured.
- **Exit:** Stage-specific checklist complete; no Sev1/Sev2 issues; stage's referenced Acceptance Tests (see §11.2) passed.

Interface Contracts and Stubs

- **Contracts:** Message schemas, timing constraints, error codes (per Communication Agreement).
- **Stubs/Simulators:** Python UART emulators and loopbacks to decouple teams until hardware arrives.

Test Coverage and Data Sources

- **Coverage:** Integration stages cite Acceptance Tests (§6) as the authoritative procedures; no parallel test lists are maintained here.
- **Data sources:** Canonical command suites, negative cases, and soak profiles are inherited from Acceptance (§6–§7) and the Communication Agreement; integration does not redefine them.

Fault Injection

Refer to AT-008 (malformed frame recovery) and AT-009 (noise) for procedures and acceptance checks. Integration stages employ these tests only when gating system-level readiness (typically Stage 6) to avoid redundant execution.

Governance & Reporting

Weekly integration stand-up; shared tracker for issues; artifacts archived with run metadata. Microsoft Teams for communication and files sharing.

Dependencies and Open Items

- Align with final Communication Agreement revision and pin maps.
- Confirm whether GPIO handshakes are required by any subsystem.
- Define soak test duration and acceptance thresholds.
- Integrate Command Library