RUMWaiter / Dumbwaiter Mk2 — Test Harness Development Requirements Specification

Prepared for: Mark Pickering — Date: 2025-08-11

1. Purpose & Scope

This document defines the requirements for the development of the **RUMWaiter Mk2 Test Harness Environment**, including the user interface (UI), Arduino-compatible emulation, and simulation features. The purpose is to provide a robust development and testing platform for the RUMWaiter Mk2 firmware, allowing re-compiled Arduino HEX files to be loaded into the emulator and executed with full access to simulated inputs, outputs, sensors, and motor physics. The test harness must support both interactive visualisation and automated headless testing to ensure firmware reliability before deployment to physical hardware.

2. Mechanical Context — 1:1 Pulley

• Pulley ratio: Fixed at 1:1 (drum direct to carriage anchor).
• Line pull at SWL 500 kg: 5,771 N.
• Drum torque at r = 50 mm: 289 N·m.
• Rope travel for 2.0 m lift: 2.0 m.
• Rope speed for 0.2 m/s lift: 0.20 m/s (~38 rpm).
• Rope type: high-strength, low-stretch synthetic (≥10× sheave diameter).
• Drum to have keeper flanges and diameter sized for rope bending limits.
• Passive brake optional; worm-drive gearbox backdrive resistance acceptable for holding.
• Motor/gearbox stall torque to exceed calculated requirement by safety factor ≥1.5.

3. Electrical & Control Context

• Controller: Arduino-compatible MCU.
• Motor driver: Cytron MD13S (baseline) with MD30C as possible upgrade.
• Motor: Worm-drive DC motor (12 V) matching torque and speed specs above.
• Sensors:
• - VL53L1X TOF distance sensor for carriage position.
• - INA226 current/voltage sensor for overload detection.
• - SSD1306 128×64 OLED for status display.
• Safety interlocks: door switch, top and bottom limit switches.
• Overload strategy: stop or safe descent when current exceeds limit for set duration.
• Power supply: 12 VDC, ≥350 W.

4. Simulation & Test Harness Requirements

The test harness must emulate the complete RUMWaiter Mk2 system in software, including the Arduino MCU, peripherals, sensors, motor, and carriage physics. It should run firmware without code modification by loading compiled `.hex` files. Interactive controls and a live visualisation of carriage position, sensor states, and outputs must be provided. The harness must include fault injection, logging, and automated scenario playback for regression testing.

5. Prescribed Runtime Environment

• Host OS: Windows 10/11, macOS 13+, or Ubuntu 22.04+.
• Node.js LTS 20.x, PNPM 9.x, Git 2.44+, Python 3.11+, Arduino CLI 0.35.x.
• Project uses a monorepo with separate folders for sim app, engine, MCU emulation, and firmware.
• Key libs: TypeScript 5.x, React 18.x, Vite 5.x, Vitest 2.x, avr8js, Zod, RxJS.
• Simulator runs in fixed-step (1 ms) and real-time modes; seedable RNG for reproducibility.
• Artifacts: static web bundle, CSV/JSON logs, HTML reports.
• Acceptance: fresh clone to UI in <2 min; firmware builds without errors; CI test suite passes.

6. Acceptance Criteria

• Physical build lifts and lowers SWL payload 2.0 m at target speed without faults.
• Simulator matches physical behaviour within defined tolerance.
• Overload, interlock, and limit switch behaviour verified in both sim and physical.
• All required documentation, code, and wiring diagrams provided.