

RCSIM Tier 4 — Raspberry Pi Robotic/Autonomous 5G Platform: User Manual

Illustrated user guide: tabs, controls, settings and step-by-step procedures. The screenshots show the English interface.

Contents

1. Definition and Scope of Tier 4	2
2. Critical Distinctions Before Assembly	2
3. Platform Hardware Components	2
4. Hardware Assembly and Actuator Wiring	3
5. Deployment via Deployment Tool	4
6. Onboard Configuration (config.json)	5
7. IP/5G Networking and Service Ports	6
8. Initial Bench Bring-Up (Without Autonomy)	6
9. Navigation, SLAM, and AI Extensions	7
9.1. Onboard LiDAR SLAM	7
9.2. Hailo-8 Edge AI Inference	7
10. Safety Systems and Failsafe Architecture	7
11. Post-Deployment Diagnostics	8
12. Source Code Verification Trails	8

Edition: 14 October 2026. This manual serves as the operator and system integrator guide for the onboard computing platform. The content is grounded in the onboard RPi codebase, configuration files, deployment scripts, and AI inference components. It does not replace physical vehicle testing or bench verification of your specific hardware assembly.

1. Definition and Scope of Tier 4

Tier 4 designates an onboard robotics and autonomous vehicle platform powered by a Raspberry Pi 5 single-board computer (SBC), integrating physical sensors, hardware actuator control, and high-speed IP/5G network connectivity. The Raspberry Pi executes core onboard services; GCS on PC functions as the remote operator station. RCSIM-MCS is a distinct mobile station stack described in a separate manual.

SLAM, Edge AI inference (Hailo-8), RTK GPS, auxiliary RC links, and system watchdogs are modular onboard extensions of this platform. Adding Hailo NPU acceleration does not create a "Tier 5" or "Tier 6", nor does it immediately confirm autonomous readiness.

Using an ELRS or mLRS radio link without an onboard computer belongs to Tier 3. Integrating an RC receiver with the Raspberry Pi is treated here as an optional auxiliary control/telemetry channel with dedicated arbitration logic.

2. Critical Distinctions Before Assembly

Process Architecture: `core/supervisor.py` manages the application lifecycle and health monitoring, while `core/main_service.py` coordinates hardware drivers, sensors, and network commands. The application container and host MediaMTX streaming service run as separate components.

Port Allocation and Service Conflicts: The onboard WebRTC signaling endpoint (`/offer`) operates on HTTP port 8080. The MCS WebUI also defaults to port 8080. Never run both stacks simultaneously on the same host interface without remapping ports. Because `docker-compose.yml` utilizes `network_mode: host`, containerization alone does not isolate port conflicts.

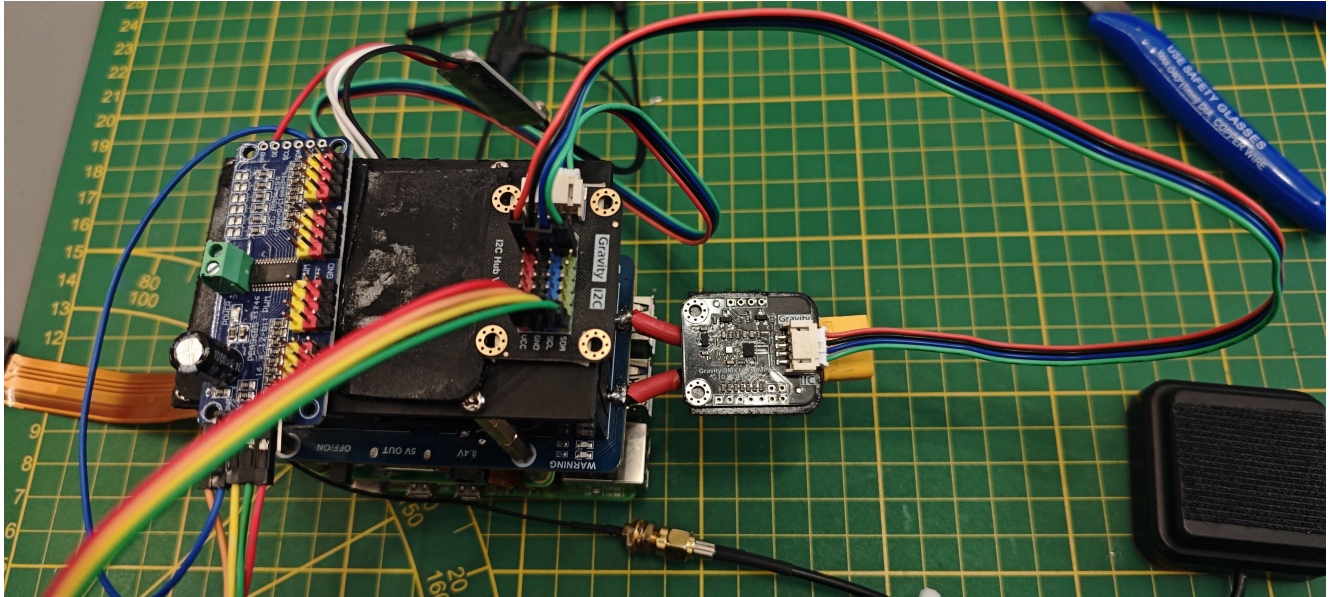
3. Platform Hardware Components

Component	Function	Operational Requirement
Raspberry Pi 5 (ARM64)	Onboard compute host	64-bit OS, hardware access permissions, adequate cooling.
PCA9685 I2C PWM Driver	Steering & throttle actuation	Separate logic (VCC) and servo rail (V+) power rails.
IMU / GPS / LIDAR	Navigation & localization	Supported serial/I2C drivers, verified live sensor data.
Camera & MediaMTX	FPV video & computer vision	Supported sensor (CSI/USB), host RTSP/WebRTC service.
5G Modem / Router	Low-latency IP transport	Stable power, SIM card/data service, verified routing.
Hailo-8 M.2 NPU	Real-time neural network inference	HailoRT runtime, valid HEF model, input tensor scaling.
Auxiliary RC Receiver	Backup manual control / failsafe	Verified serial protocol (CRSF/SBUS) and takeover switch.

The Tier 4 designation does not imply a universal driver for all 5G modems. Establish reliable host-level IP connectivity first, before configuring RCSIM remote services.

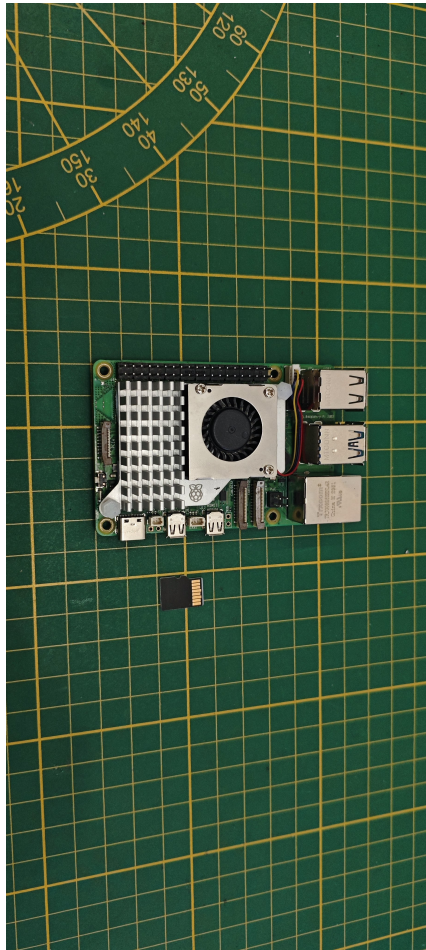
4. Hardware Assembly and Actuator Wiring

Always elevate the vehicle on a stand so drive wheels cannot contact the ground during testing. Connect or rewire GPIO and I2C buses only when the power is completely off.



Hardware stack: Raspberry Pi 5 with Hailo-8 AI accelerator

Electrically isolate logic power from servo/ESC actuator power rails. The Raspberry Pi GPIO pins cannot supply motor or servo current. On the PCA9685, VCC powers the internal I2C logic (3.3V/5V), while V+ powers the servo rail.



RPi 5 integration on RC vehicle chassis with sensors and wiring

Default config.json hardware parameters:

- `hardware.pca9685.calibration_channel: 15;`
- `hardware.pca9685.calibration_gpio: 17;`
- `hardware.pca9685.oe_pin: 27;`
- `hardware.pca9685.auto_calibrate: true;`
- Steering and throttle endpoints defined under hardware.

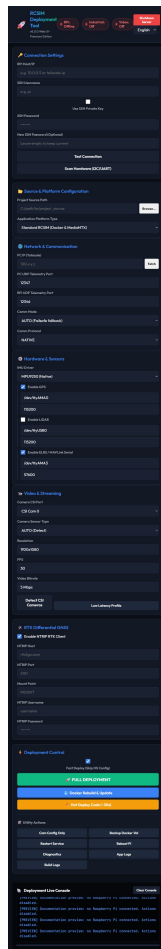
When auto-calibration is enabled, PCA9685 CH15 outputs a calibration pulse that feeds back into RPi GPIO17. Do not mistake channel CH15 for an IC pin number.

`modules/pca9685.py` measures pulse width by polling GPIO17 to determine true oscillator frequency. Following calibration, verify actual servo movement with physical measurements.

Output Enable (OE): Driving GPIO27 LOW enables PWM outputs; driving HIGH disables them. Disabling PWM does not disconnect battery power from the ESC; verify ESC failsafe behavior upon loss of PWM signal.

5. Deployment via Deployment Tool

The Deployment Tool GUI is located in
`RCSIM_deployment_tool/RCSIM_deployment_tool/RCSimRPi5deploymentapp.py`.



Deployment Tool: configuring and deploying onboard container to Raspberry Pi

1. Prepare a clean 64-bit ARM64 OS (Raspberry Pi OS Lite) and verify SSH connectivity.
2. Confirm host key identity, user credentials, and device permissions.
3. In Deployment Tool, select the project source directory `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source` and target host IP.
4. Select installed hardware sensors and verify assigned ports. The generated `config.json` must reflect your physical wiring.
5. Execute a **Full Deploy** before attempting incremental updates. The hot-deploy mechanism expects an existing `rcsim_industrial` container.
6. Verify the host MediaMTX service, container status, application logs, and volume mounts.
7. Export a backup of your configuration before modifying production settings.

The Dockerfile builds upon Debian Trixie ARM64, copies files into `/app`, and launches `entrypoint.sh`. The `entrypoint` verifies host MediaMTX on `127.0.0.1:8554` before launching the supervisor; missing video services will prevent container startup.

6. Onboard Configuration (config.json)

The primary configuration file is `config.json` located in the application root directory. It is distinct from the GCS `rc_config.json`.

Field / Section	Function
pc_ip	Target GCS IP address for UDP telemetry (waits for handshake if empty).
comm_mode, comm_protocol	Active telemetry and control communication strategy.
hardware.gps, hardware.lidar	Enabled status, serial port, and baud rate.
hardware.pca9685	I2C address, calibration pins, and OE pin.
camera, video	Video device, resolution, and RTSP stream endpoint.
ai	Model file path (HEF), runtime backend, and confidence thresholds.
autonomous_navigation	Path planner parameters, grid resolution, and obstacle margins.
ntrip	NTRIP caster credentials for RTK differential GPS corrections.

Do not enable uninstalled hardware in `config.json`. Keep NTRIP and cellular credentials out of public repositories.

7. IP/5G Networking and Service Ports

Port / Service	Confirmed Function in Code
UDP 12346	Inbound control datagrams from GCS.
UDP 12347	Outbound telemetry packets to GCS.
HTTP 8080, /offer	WebRTC signaling endpoint for onboard WebRTC strategy.
UDP 12348	Supervisor management and healthcheck socket.
RTSP 8554	MediaMTX RTSP video ingestion port.
HTTP 8889	MediaMTX WebRTC / WHEP video streaming endpoint.

HTTP signaling and WebRTC media streams use dynamic UDP ports negotiated via ICE. Endpoint availability does not confirm DTLS handshake completion.

5G and VPN Routing (STUN Requirement): When connecting across cellular NAT or VPNs (such as Tailscale), MediaMTX must be configured with a public STUN server in `webrtcICEServers` (`stun:stun.l.google.com:19302` in `mediamtx.yml`). Without STUN, the server advertises private Docker/LAN addresses, causing DTLS to establish while dropping video UDP traffic.

Never expose control, supervisor, or configuration ports to the public internet without an encrypted tunnel (VPN/WireGuard).

8. Initial Bench Bring-Up (Without Autonomy)

1. Verify stable power and elevate drive wheels.
2. Confirm the supervisor and `main_service.py` processes are active.
3. Review startup logs and compare detected devices against `config.json`.
4. Connect GCS using your chosen transport (UDP or WebRTC), keeping vehicle disarmed in Manual mode.
5. Inspect telemetry freshness: moving physical sensors must produce matching changes in GCS readouts.

6. Verify the video stream, steering servo travel, and throttle response.
7. Test failsafe: disconnect the network, simulate app termination, and verify that the ESC neutralizes without unexpected acceleration.
8. Only after manual verification should you proceed to enable autonomous navigation or AI.

A supervisor PONG response confirms socket responsiveness; it does not confirm physical PWM generation, sensor calibration, or autonomous safety.

9. Navigation, SLAM, and AI Extensions

9.1. Onboard LiDAR SLAM

SlamManager executes BreezySLAM (RMHC engine) in a dedicated worker thread. The grid is configured for **800 × 800 pixels**, representing a **40 × 40 m** area with 360-sample scans. Missing C libraries will drop the manager into a simulated fallback mode.

1100-Byte MTU Limit (SLAM Chunking): Occupancy grid maps exceed standard IP MTU limits. The onboard module `chunking.py` fragments map packets into chunks tagged with `__c: 1`, preventing IP fragmentation and packet drops over cellular 5G networks.

The onboard BreezySLAM engine is distinct from the PC-side GraphSLAM/ICP solver in GCS; their parameters and map representations are not interchangeable.

9.2. Hailo-8 Edge AI Inference

AIManager manages neural network execution via HailoRT. If the runtime or HEF file is missing, the manager falls back to a Mock pipeline.

Hailo-8 hardware HEF models require static input tensor dimensions. Incoming video frames must be resized (`cv2.resize`) and expanded to include the batch dimension (`np.expand_dims`) before inference. Alforge provides architecture targets for both `hailo8` and `hailo8l`; an ONNX export must be compiled into the matching HEF target.

Always verify that inference outputs produce changing, physically plausible control vectors and confirm manual radio override before engaging autonomous driving.

10. Safety Systems and Failsafe Architecture

PCA9685 Watchdog: The driver implements an internal 0.5 s timeout calculated from the latest `set_servo_pulse` invocation. Actuator updates refresh the timer. If expired, PWM signals are suppressed.

Simulation Fallback Guard: If the I2C bus fails to initialize, the driver silently falls back to a simulation stub (`driver_type` remains empty). An active UI and telemetry stream do not prove physical PWM output; verify electrical signals at the pins.

SafetySupervisor continuously evaluates sensor health, AI inference frame rates, and radio/IP link quality. Upon dual link loss, the system triggers either emergency STOP or Return-To-Home (RTH), depending on AI readiness.

A geofence boundary drawn in GCS is an operational limit, not a physical power cut. Safety policies must be validated end-to-end with the vehicle's ESC.

11. Post-Deployment Diagnostics

Execute diagnostic commands directly on the Raspberry Pi host terminal:

```
sudo docker ps --filter name=rctsim_industrial
sudo docker logs --tail 100 rctsim_industrial
sudo journalctl -u mediatrix -n 50 --no-pager
sudo docker inspect --format '{{json .Mounts}}' rctsim_industrial
sudo docker exec rctsim_industrial python3 tools/diagnostics/healthcheck.py
```

Symptom	Primary Check
Container fails to start, no telemetry	Check entrypoint logs and host MediaMTX on port 8554.
Supervisor responds, vehicle stationary	Check arming status, PCA9685 OE pin (GPIO27), and ESC power.
Operates locally, fails over 5G/VPN	Check routing, firewall rules, and MediaMTX STUN configuration.
Model reports Mock AI mode	Verify HailoRT driver, PCIe interface, HEF path, and architecture.
SLAM map does not update position	Verify BreezySLAM library, LiDAR serial data stream, and MTU chunking.
I2C communication errors	Check bus wiring, pull-up resistors, device addresses, and logic voltage.

12. Source Code Verification Trails

- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/Dockerfile:1 — ARM64 base image and container dependencies.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/entrypoint.sh:8 — MediaMTX check and supervisor launch.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/supervisor.py:42 — port 12348, process watchdog, lifecycle.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/main_service.py:707 — WebRTC and UDP socket bring-up.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/webrtc_manager.py:172 — /offer signaling handler.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/telemetry_builder.py:32 — telemetry packet assembly.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/config.json:1 — configuration schema.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/modules/pca9685.py:61 — calibration, OE pin, and failsafe routines.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/modules/managers/slam_manager.py:19 — BreezySLAM parameters.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/modules/ai_manager.py:69 — Hailo model loading and Mock fallback.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/safety_supervisor.py:123 — link loss arbitration and RTH logic.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/tools/diagnostics/healthcheck.py:1 — supervisor PING/PONG check.

- `RCSIM_deployment_tool/RCSIM_deployment_tool/core/deployment_logic.py:887` — deployment archiving and configuration sync.
- `RCSIM_AIforge/RCSIM_AIforge/src/hailo_converter.py:237` — Hailo-8 vs Hailo-8L architecture selection.