

RCSIM Deployment Tool — illustrated user manual

Illustrated user guide: configuration cards, deployment actions and step-by-step procedures. The screenshots show the English interface.

Contents

1. What the tool does and what this manual covers	4
2. Starting the tool and the first workflow	4
Open the correct panel	4
Recommended first-deployment sequence	4
3. Connection — preparing access to Raspberry Pi	6
3.1. Address, username and password	6
3.2. SSH private key	7
3.3. New SSH Password (Optional)	7
3.4. Scan Hardware (I2C/UART)	8
3.5. Saving settings versus executing an operation	8
4. Deployment settings — platform, network and equipment	9
4.1. Project Source Path — what you actually transfer	9
4.2. Application Platform Type — two different deployments	9
4.3. Network and protocol for standard RCSIM	10
4.4. IMU, GPS, LiDAR and ELRS / MAVLink Serial	11
4.5. Camera — port, sensor and video profile	12
4.6. NTRIP RTK — correction data for compatible GPS	13
4.7. Final form check	14
5. Preparation and deployment	14
What runs, and where installation takes place	14
Before starting	15
Two application types — the selection changes the installation target	15
What enters the package	15
Full Docker deployment — first installation	16
Fast Deploy and Docker Rebuild & Update	16
Deploying and updating MCS	17
6. Updating, checking results, and recovery	17
Hot Deploy Code — when it is sufficient	17

Updating only the camera	18
Backup, restart, and logs	18
Recognizing completion	18
Error or interruption	18
7. Services, logs and diagnostics	19
What the top indicators mean	20
Test Connection — check SSH before attempting repairs	20
Restart Service — restarting the application	20
Reboot Pi — restarting the entire Raspberry Pi	20
App Logs and Build Logs	21
Diagnostics — system information checks	21
Console, progress and closing the tool	22
8. Video and other maintenance tools	23
Video & Streaming — camera fields	23
Detect CSI Cameras	23
Low Latency Profile	24
Cam Config Only — applying stream settings	24
Backup Docker Vol — what the backup actually contains	24
Updating versus installing again	24
9. Troubleshooting and sources	25

1. What the tool does and what this manual covers

Deployment Tool prepares an application package on the operator's computer, connects to a Raspberry Pi over SSH and runs the selected installation or update procedure. It also configures communication, sensors and the camera for standard RCSIM, retrieves logs and provides selected maintenance actions.

Distinguish the two targets: **Standard RCSIM (Docker & MediaMTX)** is onboard-computer software, while **RCSIM MCS (Direct Services)** is the operator station. Running MCS on a Raspberry Pi does not make that Pi a Tier 4 autonomy platform. The application choice must match the source directory and the target host's role.

The current program opens a browser panel. This is **one screen containing cards**, rather than separate tabs. On a wide screen, SSH settings are on the left, configuration is in the middle, deployment and utility buttons are on the right, and the console is at the bottom. On a narrow screen, cards stack vertically; scroll to reach the remaining fields.

Illustrations: the project's actual HTML, CSS and JavaScript were run locally on 14 October 2026 with English selected and public default values. The documentation preview did not connect to a Raspberry Pi and blocked SSH and deployment actions. Offline/Off status is intentional; blank credentials and example ports are not a ready-to-use configuration for your setup. These are interface screenshots, not generated mockups. Zoom into the PDF to read small labels.

The code and interface layout were checked. No device installation, system update, Pi reboot or physical camera, radio or sensor test was performed. Limitations of visible options are explained with the relevant procedures.

2. Starting the tool and the first workflow

Open the correct panel

Start the prepared Deployment Tool application on the computer from which you will send the source files. The current launcher opens a browser and a local server. It starts at port 5000 and searches for an available port in **5000–5009**. Use the address actually reported by the program, such as `http://localhost:5000`.

By default, the server listens on `127.0.0.1`, meaning the computer running the tool. This is neither the MCS panel on port 8080 nor the Raspberry Pi's address. If you close the browser tab, you can reopen the running local server at its address. If the server itself was shut down, start the application again.

In a prepared source environment, the entry point is `RCsimRPi5deploymentapp.py` under `RCSIM_deployment_tool/RCSIM_deployment_tool`. Do not run simultaneous deployments to the same host. Before installing, prepare SSH and sudo access on the Pi, stable power, access to required packages and a copy of configuration you need to preserve.

The **English / Polski** list in the header changes the language. This manual uses the English labels shown in the illustrations. Some additional fields exist in the older desktop interface; do not assume they are present in the current Web version.

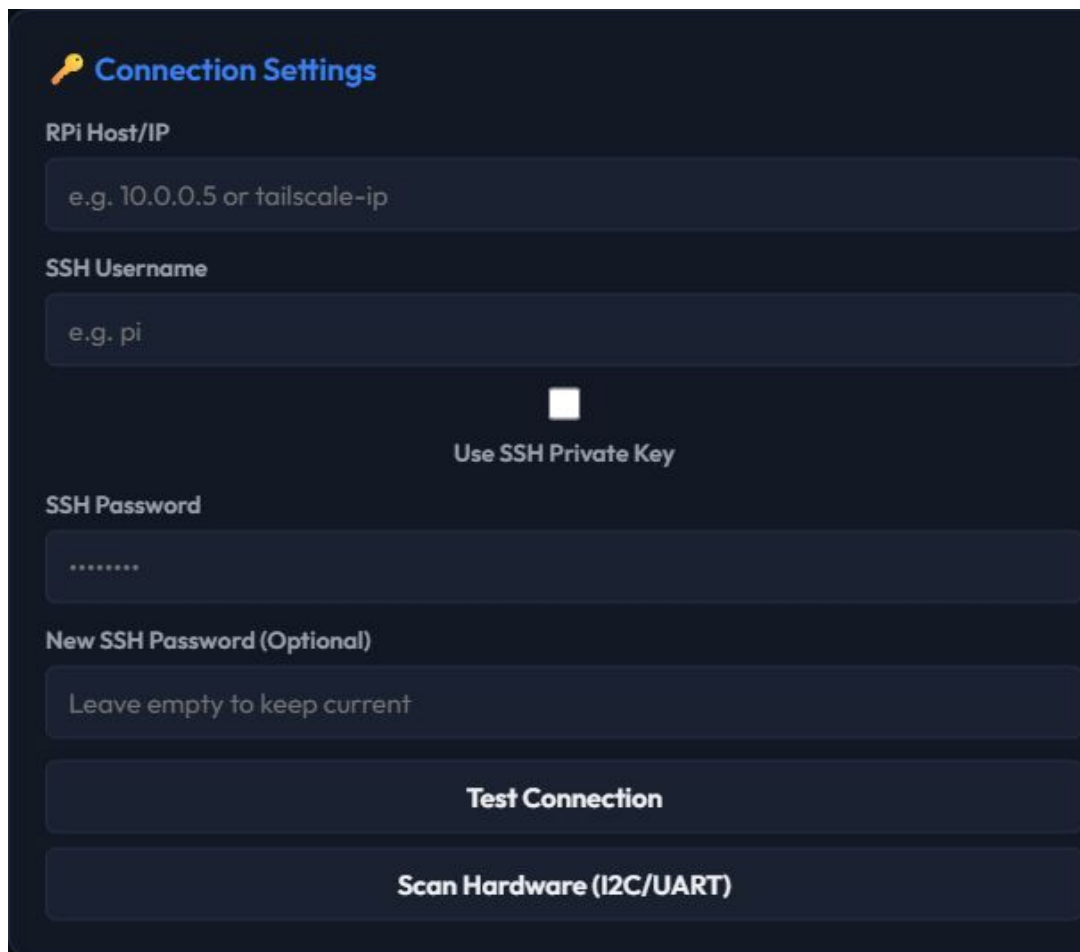
Recommended first-deployment sequence

1. Stop the model and secure the drivetrain. Establish whether you are installing onboard RCSIM or an MCS operator station.

2. In **Connection Settings**, enter the host, username and appropriate authentication method. Finish editing the field to submit the change.
3. Use **Test Connection** and read the console result. A successful ping alone does not confirm SSH login or a working vehicle application.
4. Select the correct **Project Source Path** and **Application Platform Type**. Finish editing and check that the values survive a page refresh.
5. For standard RCSIM, configure the network, protocol, sensors, camera and optional RTK to match the actual equipment. For MCS, prepare its own configuration and built frontend in the MCS source directory.
6. If you use **Scan Hardware**, read the result and refresh the form before further editing; scanning can change settings on the server.
7. Select the appropriate operation. For a first full installation, check and clear **Fast Deploy (Skip OS Config)**; its default selection does not mean the Pi's system is already prepared.
8. Start one action and watch **Deployment Live Console**. Read the final outcome, rather than relying on the progress percentage or a button becoming available again.
9. After completion, check the appropriate service, target-application connection, fresh telemetry and video if used. For MCS, check profiles and mapping. Only a separate test of the complete control path confirms model readiness.

On failure, keep the final messages and follow chapter 6's recovery procedures. The tool does not automatically roll back every change. Closing the browser tab does not cancel a command running on the Pi.

3. Connection — preparing access to Raspberry Pi



Connection Settings

RPi Host/IP
e.g. 10.0.0.5 or tailscale-ip

SSH Username
e.g. pi

☐ Use SSH Private Key

SSH Password

New SSH Password (Optional)
Leave empty to keep current

Test Connection

Scan Hardware (I2C/UART)

Deployment Tool — connection details and SSH authentication

The illustrations in this chapter show an interface preview with example settings, without executing SSH or testing hardware. Empty fields and offline status are not results from testing a particular Raspberry Pi.

Connection Settings defines administrative access to Raspberry Pi. It does not configure steering-wheel control or radio failsafe. Stop the vehicle and secure its drivetrain before deployment: later operations may stop services or restart the onboard computer.

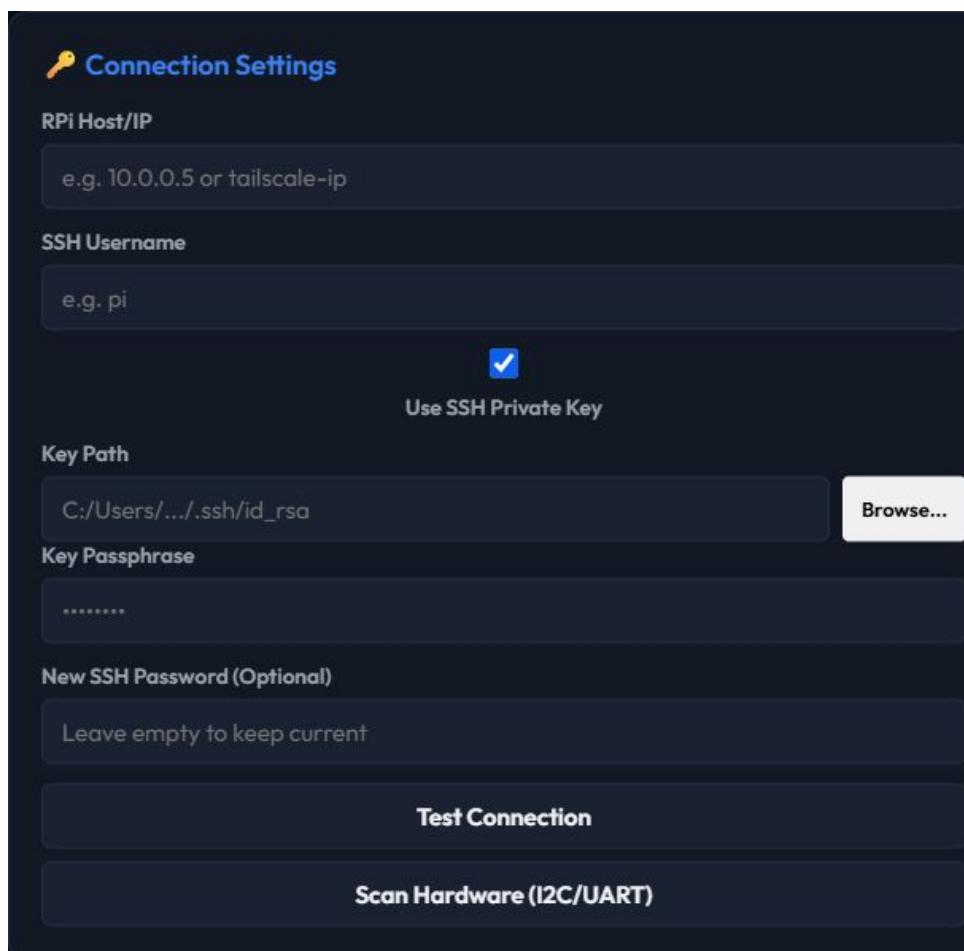
3.1. Address, username and password

- **RPi Host/IP:** enter a Raspberry Pi address reachable from the computer running Deployment Tool, such as its LAN address or a prepared Tailscale address. Enter the host/IP without `http://`, a page path or the MCS panel port. Web UI has no separate custom SSH port field; do not add `:8080` to the SSH address.
- **SSH Username:** an existing Pi user account. The `pi` hint is an example, not a guarantee that this account exists. The account needs the permissions required by the selected deployment.
- **SSH Password:** the account's current password, used when **Use SSH Private Key** is off. This is not a Wi-Fi password, NTRIP password or radio pairing code.
- **Test Connection:** checks SSH access using saved details. Wait for the console result; the button becoming available again does not mean success.

First test: enter host, username and current password, finish editing by clicking outside the field, allow the save to complete, then select Test Connection. Do not deploy until the console confirms access.

If the test fails: check that the Pi is running, the tool computer can reach its address, and SSH is enabled. Distinguish network failure from rejected credentials. **RPi: Online** at the top does not replace authentication testing or confirm that the application works.

3.2. SSH private key



The screenshot shows a 'Connection Settings' form with a dark theme. It includes fields for 'RPi Host/IP' (with a placeholder 'e.g. 10.0.0.5 or tailscale-ip'), 'SSH Username' (with a placeholder 'e.g. pi'), and a checked checkbox for 'Use SSH Private Key'. Below this is a 'Key Path' field containing 'C:/Users/.../.ssh/id_rsa' and a 'Browse...' button. There is also a 'Key Passphrase' field with a masked input '.....'. At the bottom, there is a 'New SSH Password (Optional)' field with the text 'Leave empty to keep current'. Two large buttons at the bottom are labeled 'Test Connection' and 'Scan Hardware (I2C/UART)'.

Key authentication — private-key path and passphrase

1. Select **Use SSH Private Key**. Key fields replace the account password field.
2. In **Key Path**, enter the full private-key path on the computer running Deployment Tool. Select a private key, not a public file ending in `.pub`.
3. **Browse** opens a native file picker on that computer. If viewing the panel from another browser/device, this does not upload a phone file: the dialog may open on the tool server's computer.
4. In **Key Passphrase**, enter the key's encryption passphrase if it has one. Do not automatically enter the SSH account password here.
5. Finish editing, then run **Test Connection**. A valid path alone does not prove the key is authorised on the Pi. On failure, check that the file exists on the correct computer, the key is authorised for the specified account, and the passphrase is correct. Changing the login method does not automatically install a key on Raspberry Pi.

3.3. New SSH Password (Optional)

This field represents an intended password change during system preparation, not the current login credentials. Leaving it empty preserves the current password in a supported workflow.

Web UI limitation: in the checked version, **New SSH Password (Optional)** is not saved by the configuration form. Do not assume that entering a new password here changes it. After deployment, still check login with the previously working credentials; if you deliberately want to change the password, use a supported administration procedure and verify the new login separately.

The desktop edition has its own path for passing this setting; do not automatically assume the browser panel behaves the same way. Filling the field is not confirmation that the Pi changed its password.

3.4. Scan Hardware (I2C/UART)

1. First obtain a successful **Test Connection** result.
2. Connect and power sensors according to their instructions. Do not randomly connect or rewire UART/I2C during scanning.
3. Select **Scan Hardware (I2C/UART)** and read the console result: I2C addresses, recognised GPS/LiDAR ports and camera.
4. After completion, refresh the page to load updated settings before editing further. Scanning may save detected parameters in the tool while the form still displays old values.
5. Compare the result with the actual sensor model. A shared I2C address 0x68/0x69 does not uniquely distinguish MPU6050, MPU9250 or another compatible module; the detected driver is a suggestion to verify.

Scanning does not flash firmware or prove correct units or orientation. Failure to detect a device does not justify random GPIO or supply-voltage changes. Do not leave the old form unchecked after scanning: changing another field later may save stale values again.

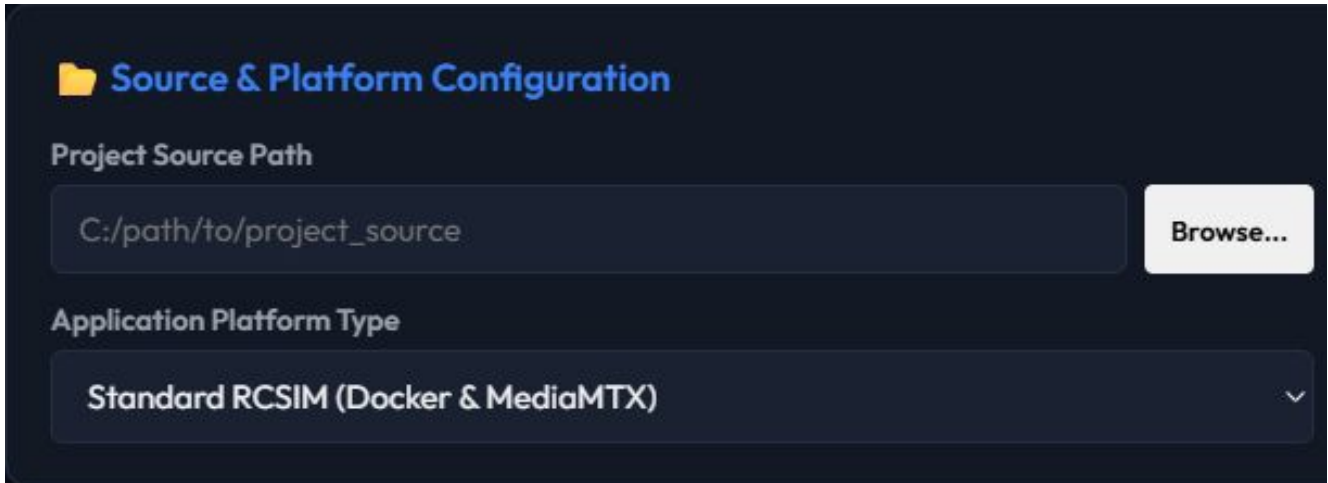
3.5. Saving settings versus executing an operation

The browser panel automatically saves the form after a field/list changes; finish editing text, for example with Tab. There is no general Apply button that immediately updates the Pi. Saving remembers details in Deployment Tool; applying them to the Pi requires the relevant deployment or dedicated update.

Before launching an operation, allow editing and saving to finish. After important changes, refresh the page and compare values. Displayed text does not confirm validity; some invalid numbers may fall back to defaults, while others can stop an operation.

The tool remembers login details in local settings. Treat this configuration as confidential; review it before attaching it to reports, and do not assume password dots mean encryption in the file.

4. Deployment settings — platform, network and equipment



Deployment Tool — source folder, platform and communication

4.1. Project Source Path — what you actually transfer

Project Source Path is a folder on the computer running the tool whose contents will be packaged for deployment. It is not the Pi destination path, a ZIP archive or the location of the PDF manual.

1. Select **Application Platform Type** before choosing the folder.
2. For **Standard RCSIM (Docke & MediaMTX)**, choose the prepared onboard project folder containing the files this variant requires, not an arbitrary folder containing the entire repository.
3. For **RCSIM MCS (Direct Services)**, choose the MCS module folder containing its application, requirements, configuration and prepared interface. Do not select only the frontend folder or an executable file.
4. Use **Browse**, check the path, finish editing and confirm it after refreshing. Folder selection occurs on the tool computer.
5. Check the source version and completeness before deploying. The tool checks whether the folder exists, but existence alone does not prove it contains the right project.

If packaging fails: check the full path, folder access and selected platform compatibility. Test Connection will not repair an empty path or a folder containing another module; SSH testing covers a different stage.

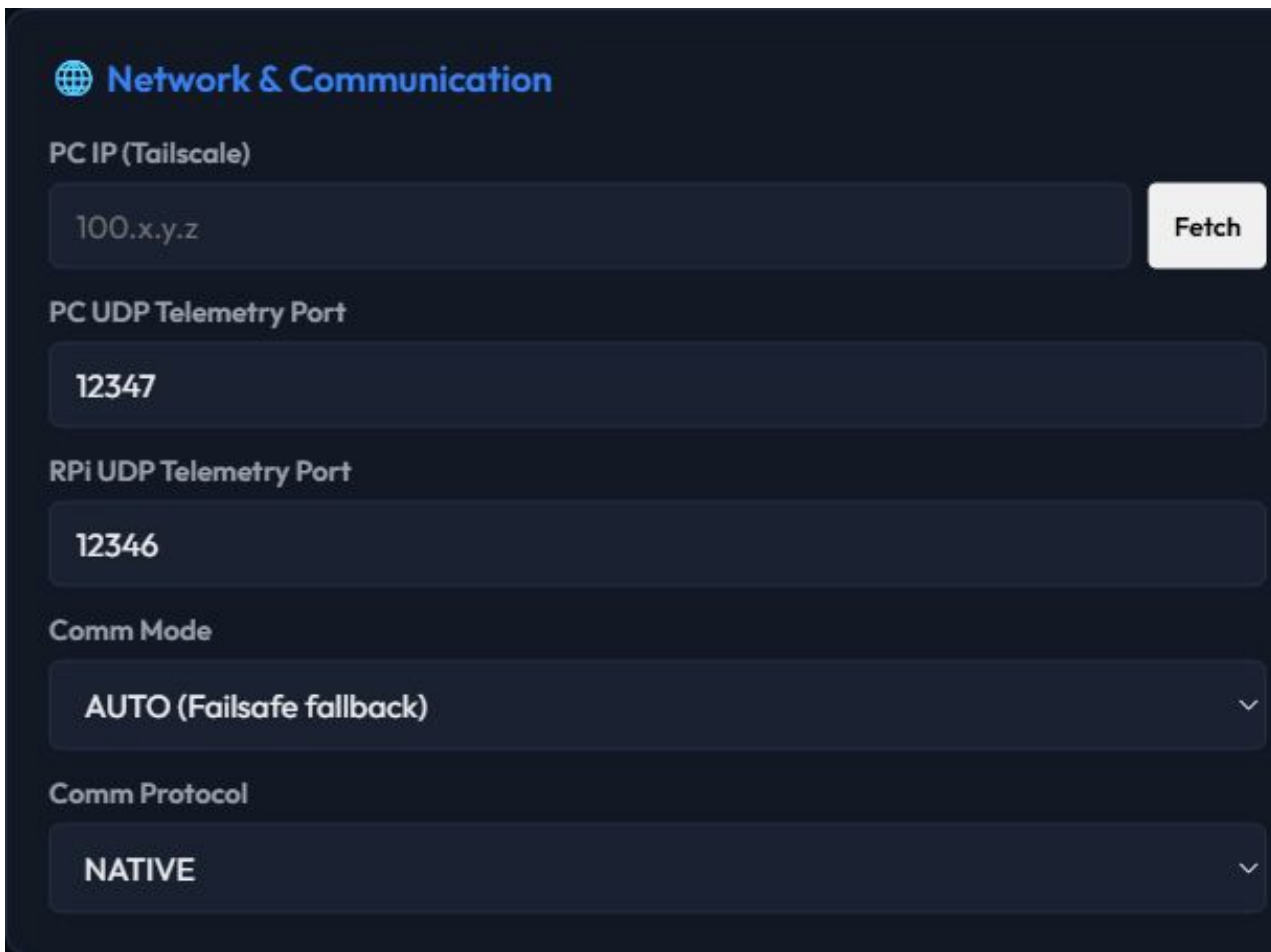
4.2. Application Platform Type — two different deployments

- **Standard RCSIM (Docke & MediaMTX):** onboard application and video variant. The form builds hardware, network, camera and RTK configuration for this variant's package.
- **RCSIM MCS (Direct Services):** mobile control station running as services. Camera and RTK cards disappear after selection; Docker-specific actions are not the equivalent of managing MCS.

Important for MCS: visible general IMU/GPS/LiDAR/ELRS and network fields are not automatically converted into the full MCS runtime configuration. This path packages the selected MCS folder; configure controller profiles, output and hardware in the actual MCS panel. Do not assume selecting an MPU in Deployment Tool overrides the MCS profile.

This Web UI has no separate manager for named deployment profiles. The last form settings are not several independent presets for different vehicles. After changing platforms, recheck folder, address, protocols and sensors.

4.3. Network and protocol for standard RCSIM



Network & Communication

PC IP (Tailscale)

100.x.y.z Fetch

PC UDP Telemetry Port

12347

RPi UDP Telemetry Port

12346

Comm Mode

AUTO (Failsafe fallback) ▾

Comm Protocol

NATIVE ▾

Network settings — PC station address, ports and protocols

- **PC IP (Tailscale):** the PC station address to which the vehicle should send the relevant data. It is not the Pi address entered for SSH.
- **Fetch:** attempts to read the tool computer's Tailscale IPv4 address; on failure it may return that computer's ordinary address. Check the result; it does not prove VPN is installed or working on the Pi.
- **PC UDP Telemetry Port:** the PC receiving port, such as 12347 only when GCS listens there.
- **RPi UDP Telemetry Port:** the Raspberry Pi-side port, such as 12346 for a matching configuration. PC and Pi ports have different roles; do not make them identical without a reason.
- **Comm Mode:** AUTO, WEBRTC or UDP. AUTO selects logic in compatible software; it is not a promise to repair every link automatically or stop the drivetrain.
- **Comm Protocol:** NATIVE or MAVLINK. Match onboard software and GCS. MAVLINK does not automatically install an autopilot or change radio firmware.

Network setup test: enter the correct receiver address, configure ports on both sides, match mode/protocol, finish editing and only then deploy. Afterwards check fresh GCS telemetry, control and video separately. Missing video is not a reason to change UDP Telemetry Port randomly.

The web panel has no separate **MAVLink System ID** or **MAVLink Throttle Hz** fields found in the desktop edition. Do not look for them in the illustration. The desktop edition has a system identifier and a 1, 2, 5, 10, 20, 50 Hz selection; do not confuse this rate with serial baudrate.

4.4. IMU, GPS, LiDAR and ELRS / MAVLink Serial

The screenshot shows a configuration window titled "Hardware & Sensors" with a gear icon. It contains several sections for configuring sensors:

- IMU Driver:** A dropdown menu currently showing "MPU9250 (Native)".
- Enable GPS:** A checked checkbox. Below it are two input fields: the first contains "/dev/ttyAMA0" and the second contains "115200".
- Enable LiDAR:** An unchecked checkbox. Below it are two input fields: the first contains "/dev/ttyUSB0" and the second contains "115200".
- Enable ELRS / MAVLink Serial:** A checked checkbox. Below it are two input fields: the first contains "/dev/ttyAMA3" and the second contains "57600".

Deployment Tool — selecting equipment drivers and ports

IMU Driver: select the actual installed module: MPU9250 (Native), MPU6050 (Native), GY-91, BMX160 + BMP388, BMX160 or BNO08x. **Simulation (Dummy IMU)** is a demonstration source, not proof of a working physical sensor. Confirm the model manually after an I2C scan.

Enable GPS: select it if the vehicle has a supported receiver. **GPS Serial Port** is the device path on the Pi, such as /dev/ttyAMA0, not a Windows COM port. Enter **Baudrate** as a positive integer matching the GPS's active output, for example 115200 only for a device configured that way.

Enable LiDAR: enables sensor configuration. **LiDAR Port** is its Pi device path, and **Baudrate** is the device's serial speed, not its scan rate in Hz. /dev/ttyUSB0 is an example that may identify another adapter after USB device changes.

Enable ELRS / MAVLink Serial: enables the relevant serial configuration. **ELRS Serial Port** is the Pi port; **Baudrate** applies to the Pi-module segment. The default 57600 is not a universal CRSF, ELRS or mLRS speed; it must match the actual firmware and protocol.

Procedure for each sensor:

1. Check its physical model, power and interface using the platform manual.
2. Use scanning as an aid, refresh the form and check the suggested port/driver.
3. Disable functions for equipment absent from the vehicle. Do not leave GPS/ELRS enabled merely because the form opened that way.

4. Enter the correct Pi port and positive Baudrate without bps or thousands separators. Do not assign the same physical line to GPS, LiDAR and radio.
5. Apply configuration through the relevant deployment. After startup, check the sensor's application data, freshness, units and directions, not just a detected-port message.

For missing data, check port permissions, another service occupying the port, and device baudrate. Selecting a driver neither physically installs the sensor nor calibrates its orientation.

4.5. Camera — port, sensor and video profile

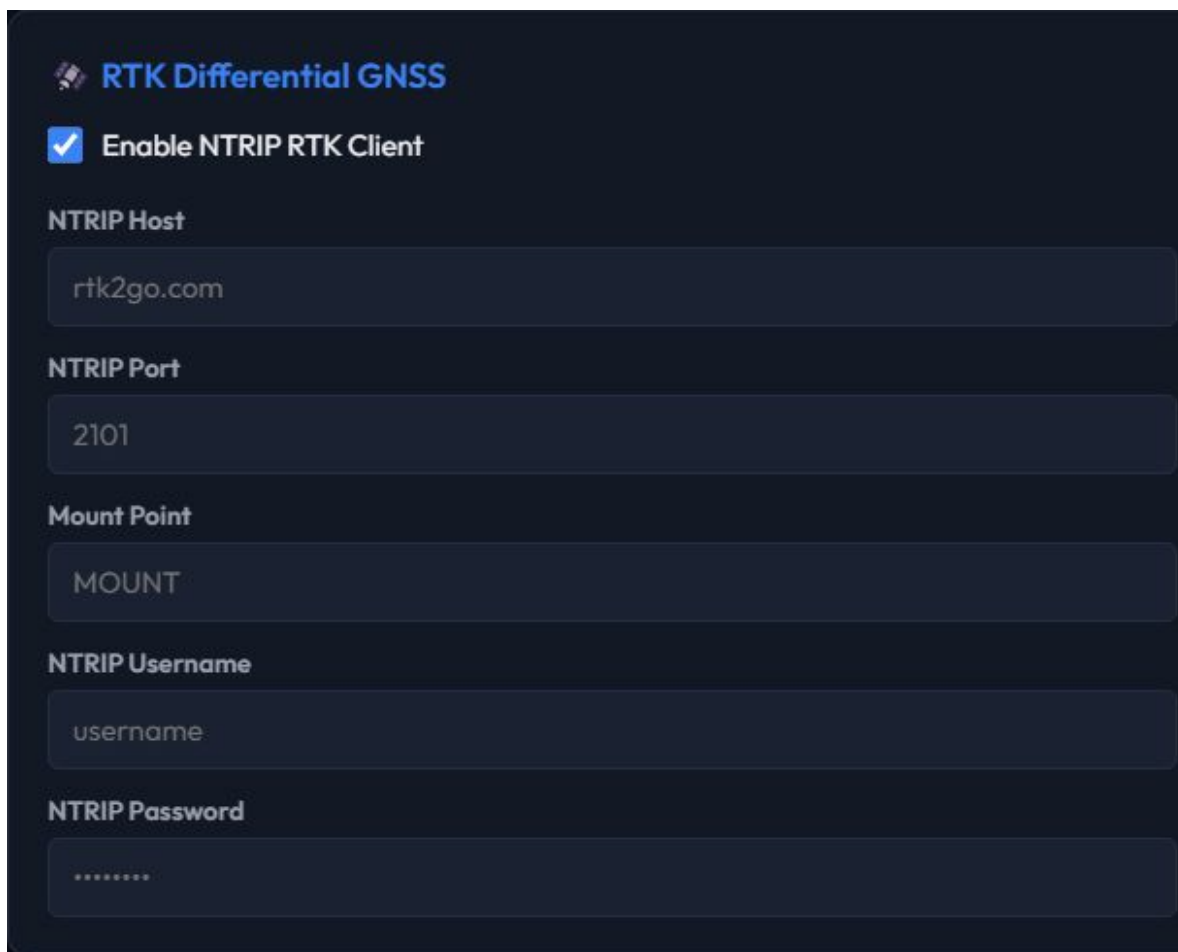
The video card is visible for standard RCSIM and hidden for MCS. **Camera CSI Port** selects cam0, cam1, /dev/video0 or /dev/video1; match the actual device path.

- **Camera Sensor Type:** AUTO (Detect), imx219, imx708, ov5647, imx477 or imx519. Match the sensor to the camera; selecting another model does not provide compatibility.
- **Resolution:** enter width and height in pixels, such as 1920x1080, with lowercase x, no unit or Full HD suffix.
- **FPS:** a positive whole number of frames per second, such as 30. This is a requested parameter, not a guaranteed output.
- **Video Bitrate:** use a form-compatible notation, such as 5 Mbps. This version expects a whole number of Mbps at the beginning; 2.5 Mbps, 5000000 or decimal-comma notation are not equivalent safe formats.
- **Detect CSI Cameras:** reads camera detection from the Pi and prints the result. Read the console; clicking alone does not mean the correct sensor/port has been selected.
- **Low Latency Profile:** locally sets **800x600, 30 FPS, 2 Mbps** and saves the form. It is a preset, not a latency measurement or an immediate Pi stream switch.

Procedure: check the camera with Detect CSI Cameras, select a compatible port and sensor, set resolution/FPS/bitrate, finish editing and deploy configuration or use the appropriate dedicated camera update. Then open the actual stream in GCS and check current video, smoothness and motion delay.

To begin with a lower transmission load, use Low Latency Profile and consciously accept lower resolution. For a black image, first check sensor detection, connector, video service and its log rather than only increasing bitrate.

4.6. NTRIP RTK — correction data for compatible GPS



RTK Differential GNSS

☒ **Enable NTRIP RTK Client**

NTRIP Host

rtk2go.com

NTRIP Port

2101

Mount Point

MOUNT

NTRIP Username

username

NTRIP Password

NTRIP RTK — correction server, mountpoint and credentials

Enable NTRIP RTK Client is available in the standard variant. Improving position requires an RTK-capable receiver and the correct correction-data path; the checkbox alone does not turn ordinary GPS into an RTK receiver.

- **NTRIP Host:** correction server name/address without a web-page address or path.
- **NTRIP Port:** service port, such as 2101 only for a provider configured that way.
- **Mount Point:** the exact correction stream name supplied by the provider, with its spelling preserved.
- **NTRIP Username / Password:** that service's credentials; independent of SSH login and Wi-Fi.

Procedure: obtain stream details, confirm receiver compatibility and internet on the Pi, complete fields, finish editing and apply configuration through deployment. After startup, check caster connection, incoming corrections and the actual GPS solution state. Coordinates, satellite count or “RTK enabled” alone do not confirm RTK FIX or a particular accuracy.

If RTK is unused, disable the client. For authentication failure, check account and mountpoint; for connection failure, host, port and Pi internet access. Do not change radio control baudrate because of an NTRIP problem.

4.7. Final form check

Before deployment confirm: correct host/account, successful SSH test, correct platform and an existing source folder. For RCSIM, check PC address, ports, protocols, actual sensors, numeric formats, camera and optional NTRIP. For MCS, check source completeness and plan subsequent configuration in the MCS panel. After finishing edits, allow saving to complete; only then select the deployment action described in the next part of the manual.

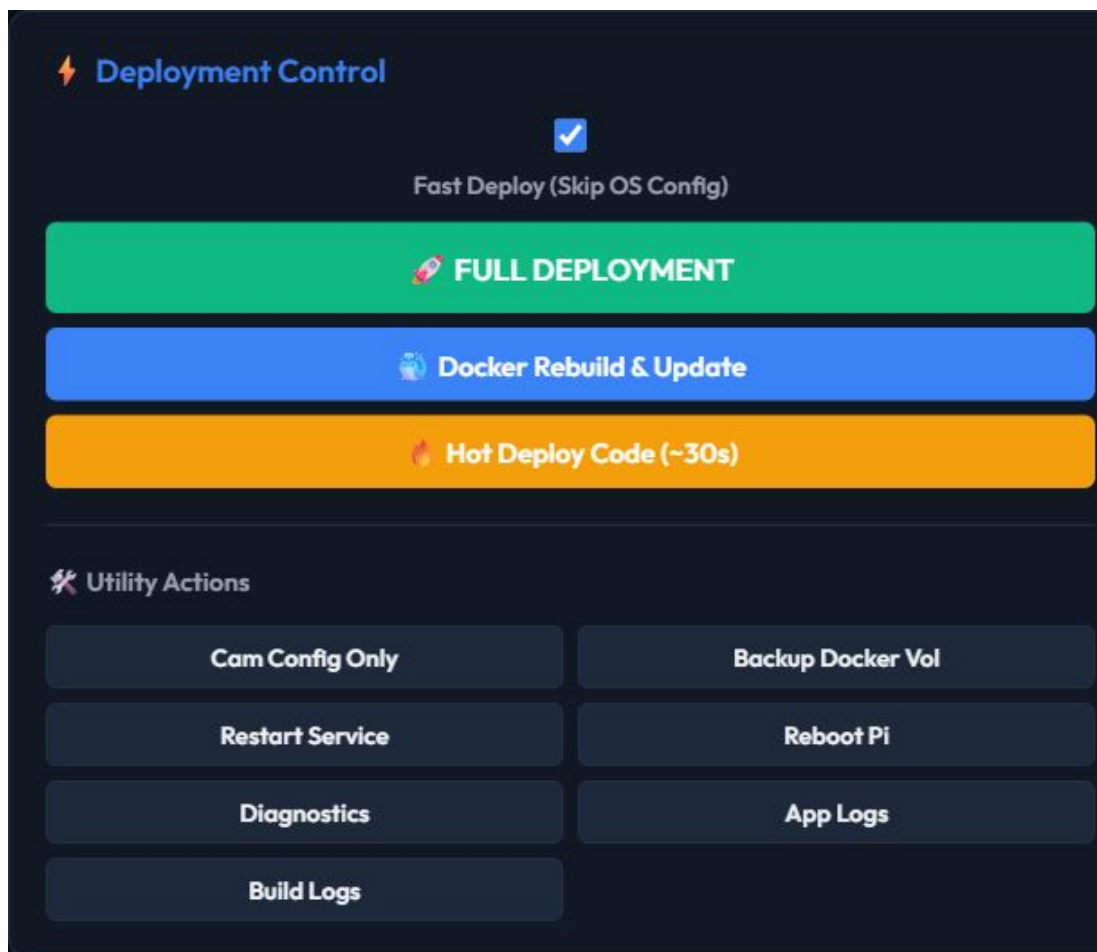
5. Preparation and deployment

What runs, and where installation takes place

The current Deployment Tool starts a local server and opens its panel in a browser. It selects an available port from **5000–5009** on 127.0.0.1. Use the address reported at startup; this is not the MCS WebUI on port 8080 or the onboard WebRTC service.

Sources and the archive are prepared on the computer running the tool. Files are transferred to Raspberry Pi over SSH/SFTP, while installation and Docker image building run on the Pi. The current panel has no separate “local / remote build” selector. The browser does not compile the image, and directory selection refers to a folder on the computer running the tool.

Actions are in the right-hand **Deployment Control** panel, not a separate Deployment tab.



Deployment control panel

The illustration shows the actual frontend without an active SSH connection. No deployment was performed for the documentation.

Before starting

1. Stop the model, disarm it, and physically secure the drivetrain. Updating interrupts control services or video; do not update while driving.
2. Check the target host, account, and SSH authentication method. Sudo access must be available for installing packages, services, system configuration, and Docker.
3. Provide stable power, card space, and RAM. Docker builds in a RAM directory under `/dev/shm`; free card space does not solve a RAM shortage.
4. Ensure the Pi can reach package repositories and required dependencies. Working SSH does not mean the Pi has Internet access.
5. Preserve your own copy of the device's active configuration and required data before replacing them. Do not assume an automatic backup or automatic return to the old version.
6. Select the correct **Application Type** and **Project Source** directory; after browsing for a folder or fetching an IP, click outside the field and check saving. These buttons populate fields, but population alone does not guarantee submission of the new value to server settings.
7. Finish editing each field before starting an action. Changes are saved on the field's change event; do not start deployment while still entering an address or path.
8. Run **Test Connection**, then proceed only after SSH is confirmed.

Two application types — the selection changes the installation target

RCSIM Docker is for the Tier 4 onboard computer. Select

`RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source`, the directory containing `Dockerfile`, `docker-compose.yml`, `entrypoint.sh`, `config.json`, and application code. Do not select the whole repository or the MCS directory.

RCSIM MCS (Direct Services) installs the MCS station as a system service. Select `RCSIM_MCS`, containing `requirements.txt`, `systemd/usb_rc.service`, `src/`, `config/`, and built files in `src/ui/web/`. This is a different program and target from the Tier 4 onboard image.

For MCS, the panel hides video and RTK settings and attempts to disable backup and build-log buttons. However, periodic status refresh may enable those buttons again after about 2.5 s. **Backup Docker Vol** and **Hot Deploy Code** still target the Docker path. The **Build Logs** backend has a separate MCS installation-log reader, described in chapter 7. An enabled button alone does not confirm platform compatibility for other functions.

What enters the package

The tool creates a local `temp_deploy` working directory and ZIP archive. It excludes items including the Git repository, Python environments, `node_modules`, caches, logs, and images/archives. It does not automatically transfer the computer's entire runtime environment.

For Docker, a copy of the source `config.json` is merged with panel settings. Existing keys are retained, and new values replace matching fields; this does not unconditionally preserve every manual setting from the Pi. A `mediamtx.yml` is also generated from camera parameters. The application camera is configured for RTSP at `rtsp://127.0.0.1:8554/camera_ai`.

For MCS, the package copies configuration from the selected source directory. Onboard sensor/RTK fields in Deployment Tool do not convert MCS profiles. Watch the local `config/config.json`: it can replace profiles previously saved on the Pi. Build the MCS frontend before packaging and confirm that `src/ui/web/` is current; sending React sources alone does not build them on the device.

Full Docker deployment — first installation

1. Select Docker, the correct directory, and host. Set communication, installed hardware, and camera according to the actual setup.
2. For a first installation, clear **Fast Deploy (Skip OS Config)**. The full-deployment button uses the checkbox's current value; FULL in its name does not force full system configuration.
3. Select **FULL DEPLOYMENT** and watch the bottom console. The archive is created first, followed by connection, transfer, and script execution on the Pi.
4. If a Tailscale login request appears, complete it for the correct device. Opening the login page is not the end of installation.
5. Wait for the script result and final message; do not start another action merely because the ZIP was uploaded.
6. After deployment, check the application, MediaMTX, telemetry, and video, then separately check a safe control signal.

Actual scope of changes: the script installs/checks packages and dependencies, configures services, installs or updates host MediaMTX, builds an image on the Pi, replaces `/home/USER/rcsim_project`, and recreates Compose containers. It may disable the older `rtsp-camera.service`. The underlying installer also has an SSH password-change mechanism, but the current web workflow does not effectively pass the **New SSH Password** field to the operation; do not expect a password change merely from filling it in (see section 3.3).

Before building, the script clears **the whole host's** `/tmp/*`, runs `docker system prune -f`, and restarts Docker. This is not limited to RCSIM cleanup: it can remove other programs' temporary data and unused Docker resources and interrupt other containers. Use this path on a prepared host and preserve required data beforehand.

Fast Deploy and Docker Rebuild & Update

Fast Deploy (Skip OS Config) reduces some system steps, particularly initial APT cleanup/update. It does not promise no system changes. Dependency checks, MediaMTX handling, Docker building, `/tmp` cleanup, Docker prune, Docker restart, and project replacement still occur.

Docker Rebuild & Update runs the same deployment procedure with Fast Mode forced on. Choose it after changing Dockerfile, system/Python dependencies, or when updating code alone is insufficient. Back up required data and check the result as for full deployment.

The current panel has no separate **Clean Build** button, PC compilation selector, or installation of a ready-made image from a local cache. Do not assign those meanings to similarly named options; “Rebuild” refers to remote Compose building.

Deploying and updating MCS



Source & Platform Configuration

Project Source Path

C:/path/to/project_source **Browse...**

Application Platform Type

RCSIM MCS (Direct Services) ▾

Action panel for MCS — preview without an SSH connection

1. Select **RCSIM MCS (Direct Services)**, the MCS directory, and the correct host.
2. Preserve profiles from the Pi and check the package configuration. Confirm a built frontend.
3. Use **FULL DEPLOYMENT**. Fast Mode is not passed to the separate MCS installer and does not disable its system steps.
4. Wait for package installation, extraction into `/opt/usb_rc_converter`, venv preparation, and the restart of `usb_rc.service`.
5. Check the log and service. After the required Pi restart, open the MCS WebUI and confirm the profile, devices, and protocol before arming the model.

The MCS installer overwrites matching files in `/opt/usb_rc_converter`; it does not delete the whole directory before extraction, so old files absent from the new package may remain. It disables and removes the older `usb_rc_core.service` and `usb_rc_web.service` units, then installs the combined `usb_rc.service`.

It also changes I2C/UART boot configuration, disables the `ttyAMA0` serial console, and prepares a `uap0` Wi-Fi interface and `rc.local` entries. This modifies the host, not just the webpage. UART/I2C changes require restarting the Pi. Creating the interface does not confirm a working AP or compatibility with every Wi-Fi adapter.

With MCS selected, **Docker Rebuild & Update** also routes to the MCS installer; it does not rebuild an image in that case. For clarity, use explicit full MCS deployment. Do not use **Hot Deploy Code** to update MCS.

6. Updating, checking results, and recovery

Hot Deploy Code — when it is sufficient

This path requires a previous Docker installation and an existing `rcsim_industrial` container. It updates code and configuration without building a new image. It does not deliver new dependencies from `Dockerfile` or `requirements.txt`.

1. Select Docker, the correct sources, and host; preserve required data from the active installation.
2. Stop the model and run **Hot Deploy Code (~30s)**. The time in the label is approximate, not guaranteed.
3. Wait for container shutdown, synchronization, and startup.
4. Check the log for individual copy and restart errors, then check the services' actual operation.

Hot deploy can replace `config.json`, `docker-compose.yml`, and MediaMTX configuration. It replaces the **core**, **logic**, **modules**, **tools**, **models**, **web_assets** directories in the host project; your own files in those directories may disappear. It also updates selected startup files. If Compose changed, it removes and recreates the container; otherwise it restarts the existing one. Control and video are interrupted.

The final Hot Deploy message is not a complete health test. The script checks for a running container, not PWM, sensors, inference, or the WebRTC channel. Some step errors are printed without unconditionally stopping the entire procedure. Read the log, not just the last green line.

Updating only the camera

Cam Config Only handles host MediaMTX settings. After changing camera parameters, use this action instead of rebuilding the entire image. It changes configuration and restarts the service, so video may disappear during the operation. It does not update application code, camera firmware, or MCS profiles. After the service returns, check the actual resolution/video and the source match in GCS.

Backup, restart, and logs

Backup Docker Vol downloads an archive of `rcsim_project` from the Pi user's home directory to the computer. The web panel saves it as `rcsim_docker_backup.tar.gz` in the computer user's home directory; another backup uses the same name, so preserve any required older file first. Its name does not mean a whole-SD-card image, a complete system backup, or all Docker volumes belonging to other applications. Check the file after completion. Back up MCS configuration separately. Although panel refresh may enable the button again in MCS mode, this action still does not copy `/opt/usb_rc_converter` and is not an MCS backup.

Restart Service selects the service according to application type: the Docker container or MCS service. It does not restart the whole Pi. **Reboot Pi** restarts the host, interrupting SSH and all its services; wait for connectivity to return before the next action.

App Logs displays the relevant application type's log. **Build Logs** concerns Docker building, while **Diagnostics** collects diagnostic information. A build log and a running application's log answer different questions: successful compilation does not confirm working sensors.

Recognizing completion

The progress bar approximates stages. During full deployment, it may reach 100% before a script error is reported. Buttons becoming available or an archive appearing does not mean installation succeeded.

Treat the installer's result as the success message after the remote script exits with code 0. Then perform an independent check:

- **Docker:** the container is not in a restart loop, MediaMTX is active, the application log has no critical errors, and GCS receives fresh data and the correct video.
- **MCS:** `usb_rc.service` stays active, the panel responds, the profile is correct after refreshing, the host sees controllers, and the selected port and protocol match the hardware.
- **Both:** neutral throttle, steering direction, arming, and link-loss response have been checked on a secured bench. No green indicator replaces these tests.

Error or interruption

The current panel has no button to cancel an active deployment. Closing the browser tab does not request rollback. **Shutdown Server** terminates the local program but does not guarantee stopping the Pi script or restoring the previous installation. Do not use it as Cancel.

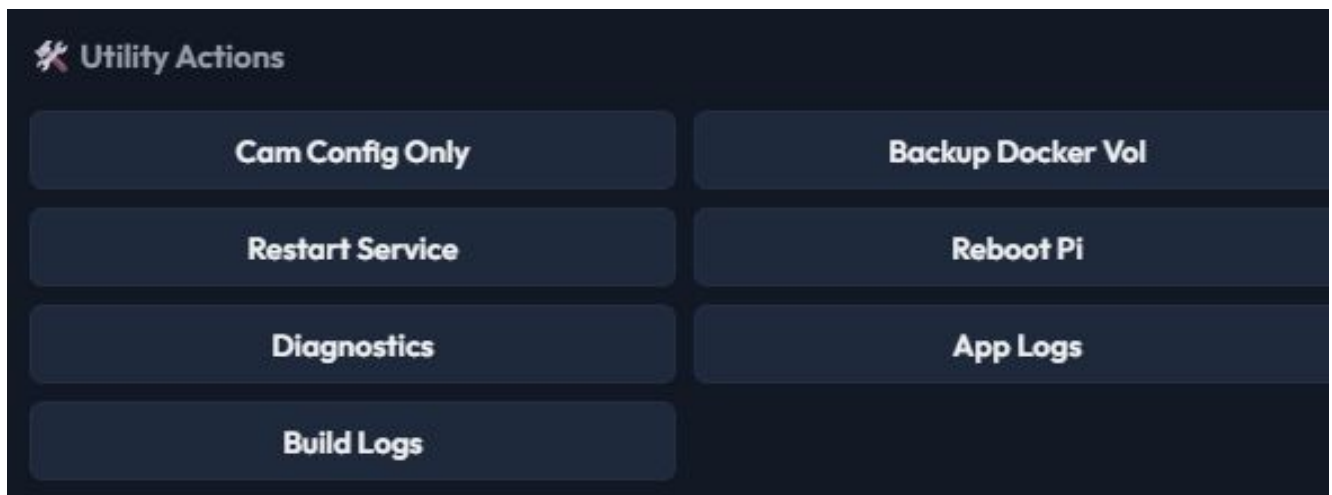
If progress stops changing, read the console and log first: package downloads or building may continue without frequent percentage changes. Do not start a second deployment. The tool blocks simultaneous actions, but losing the browser or restarting the program does not prove the remote process has finished.

After an error, check **App Logs** and **Build Logs**. Full deployment also attempts to retrieve the end of `rpi_setup_docker.log` or `rpi_setup_mcs.log`, and for Docker `docker_build.log`, from the Pi user's home directory. Preserve the first cause of failure, not just the final message.

If memory/space ran out, check RAM and the card separately. If SSH failed, check the host, account, key/password, and any password change performed by the installer. If the container will not start, check dependencies and host MediaMTX. If Hot Deploy has no container to update, perform a correct full Docker deployment.

Recovery means deliberately redeploying verified sources/configuration or restoring your own backup. The panel does not provide automatic rollback. After repair, repeat the entire control-path check before driving. These procedures were verified against code; no deployment to a physical Pi was performed while preparing this manual.

7. Services, logs and diagnostics



Maintenance controls preview; no SSH connection or remote operation was performed

Deployment Tool has a single page with settings cards and maintenance buttons. There is no separate Services tab. Use **Utility Actions** and read the results in the **Deployment Live Console** at the bottom. Check the Raspberry Pi address and selected platform before every operation. Restarting or updating while driving interrupts control services; secure the vehicle and end the session first.

What the top indicators mean



Status indicators preview without SSH; this is not a measurement of a physical Pi's state

RPI: Online means a successful ping response, not a working application or successful SSH login. After a successful ping, the tool attempts to check services over SSH. If ping is blocked, other indicators may remain inactive even when SSH is available. Do not use an indicator's color alone as a conclusive connection test.

For the Docker platform, **Industrial** checks the main `rcsim_industrial` application, while **Video** checks MediaMTX. For **RCSIM_MCS**, **Core Service** and **Web Service** come from the same `usb_rc.service` check. They are not two independent process tests or proof that a browser can open the page. Service status does not confirm camera images, channel reception or physical sensor operation.

Test Connection — check SSH before attempting repairs

1. Check the host, username and password or SSH key file.
2. Click **Test Connection**.
3. Read the console result; correct connection errors before retrying installation operations.
4. After changing settings, wait for them to save and test again.

This is the appropriate first step for access problems. Successful SSH does not yet confirm permission for every `sudo` command or application-service readiness.

Restart Service — restarting the application

Restart Service acts on the platform selected in the form:

- Docker: restarts the `rcsim_industrial` container.
- MCS: restarts `usb_rc.service`.

It does not reboot Raspberry Pi or separately restart MediaMTX. This panel version does not display a separate confirmation for this button.

1. Secure the model and make sure the correct platform is selected.
2. Click Restart Service and wait for a console message.
3. After a short wait, check the status and **App Logs**.
4. Recheck actual control or the MCS panel before resuming operation.

A message that a restart was sent is not a health check of the restarted application. If the service fails again, read its logs instead of repeatedly restarting it.

Reboot Pi — restarting the entire Raspberry Pi

Reboot Pi sends a system-reboot command over SSH after confirmation. It interrupts connections, video and running applications on the target Pi.

1. End the drive and make sure the target is the correct host.
2. Click Reboot Pi and confirm.
3. Wait for the system to boot again; a temporary Offline state is expected.
4. Repeat Test Connection and check service indicators and logs.

Reboot command sent means the command was sent, not that reboot has completed. If the host does not return, check power, networking and a possible address change.

App Logs and Build Logs

App Logs retrieves the last 100 lines: from the `usb_rc.service` journal for MCS or from the `rcsim_industrial` container for Docker. This is a one-time retrieval, not a continuous remote log stream. To check a new error, retrieve the logs again. The current read path returns the command's standard output; some information written to standard error may not reach the panel. An empty result does not prove there are no problems.

Build Logs retrieves a build/installation log. For Docker, this is `~/docker_build.log` on Raspberry Pi. In MCS mode, it reads `~/rpi_setup_mcs.log`. The panel attempts to disable the button when MCS is selected, but periodic status refresh may enable it again. If the button is available, retrieval still requires the correct file to exist on the host. Do not assume it contains every update's logs: the file must exist and relate to the intended operation.

1. After a startup failure, choose App Logs; after a build failure, choose Build Logs if available.
2. Find the first specific error and the messages immediately before it.
3. Preserve the text before clearing the console; note the platform and operation performed.
4. Do not treat the final failure sentence as the only information needed for troubleshooting.

Diagnostics — system information checks

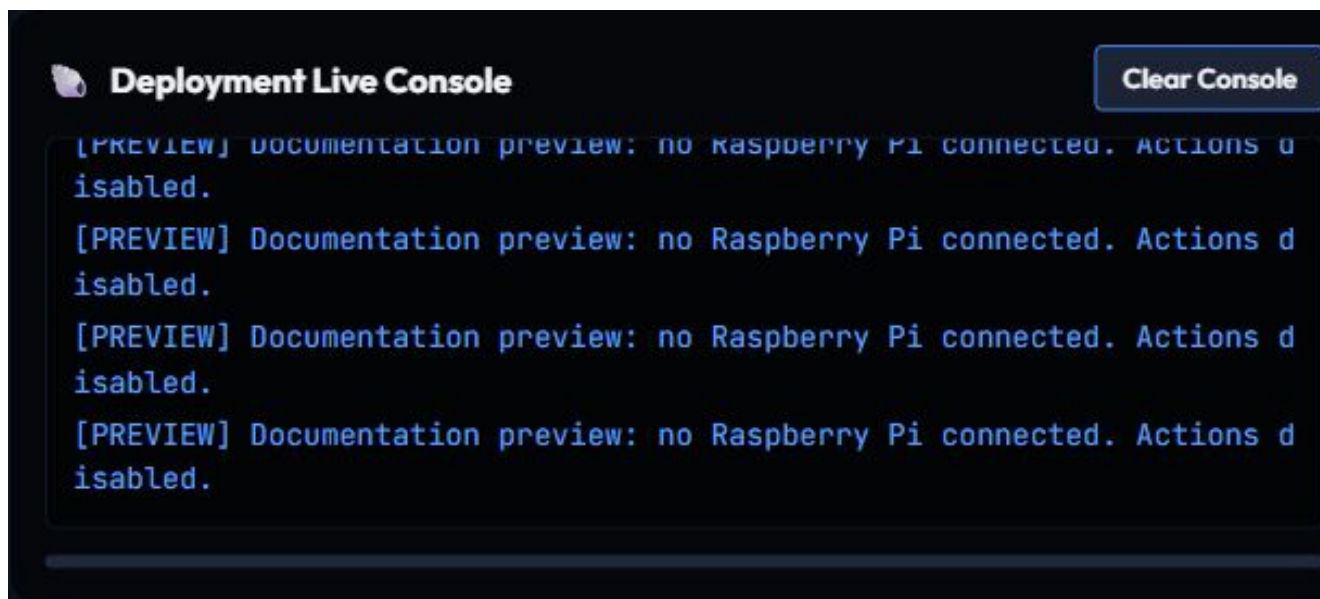
Diagnostics retrieves uptime, disk and memory information, Docker containers, MediaMTX status and its last 20 log lines, power/temperature throttling information, CPU temperature and Tailscale status. Results appear sequentially in the console under each checked area's name.

The set of commands is currently fixed and includes Docker/MediaMTX even when MCS is selected. The absence of such a service in an MCS installation does not automatically mean MCS is broken. The tool does not assess each command's exit status separately and may finish with OK despite an individual check failing. Empty sections also need checking; they do not represent a successful test.

1. Run Diagnostics after checking SSH.
2. Compare its sections with the host's equipment and installation type.
3. If storage is low, investigate disk usage; for a MediaMTX issue, examine its logs.
4. For a control problem, also check App Logs and actual channel reception.

Diagnostics does not physically test servos, receiver failsafe, sensors or driving latency.

Console, progress and closing the tool



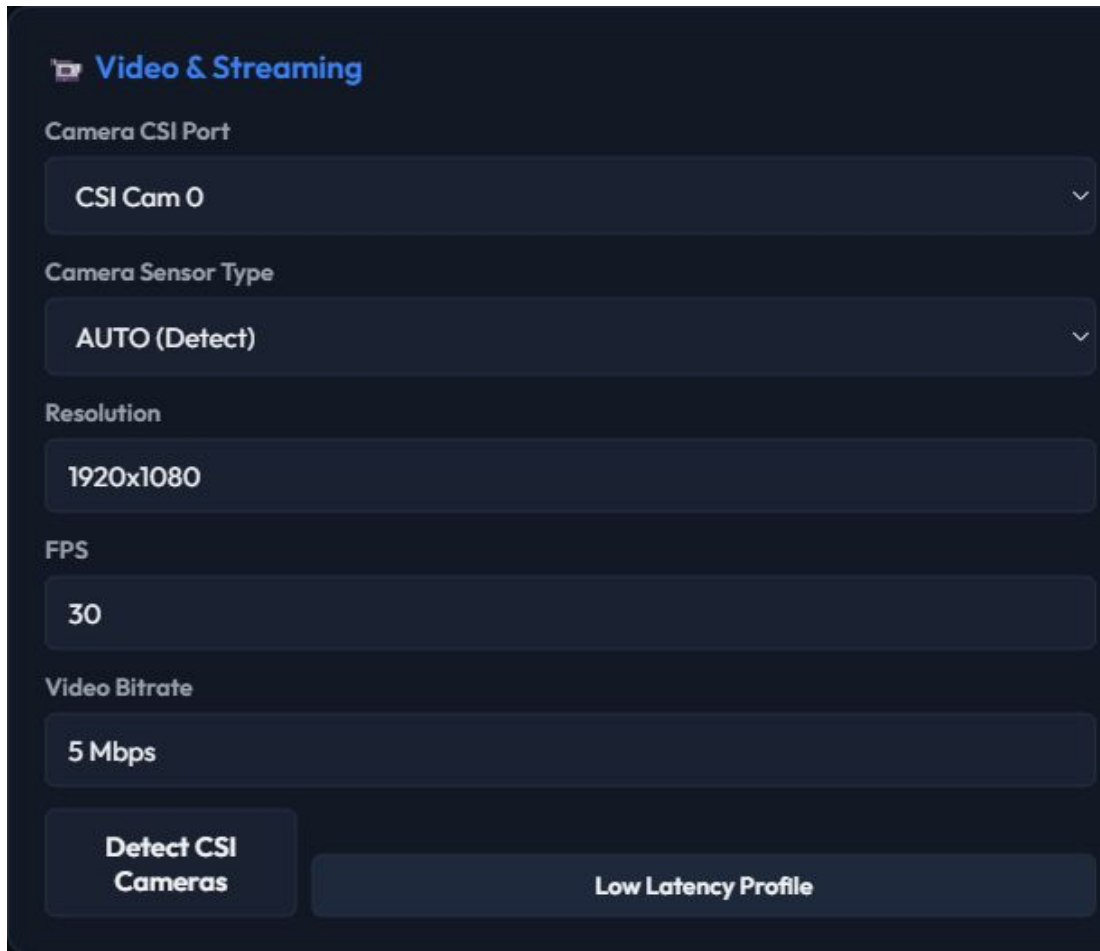
Console before running tasks; this illustration does not show a deployment result

The console displays task messages and results. The progress bar represents operation stages; it is not a precise measurement of remaining time. During a lengthy step, check for new messages instead of launching the same operation again. A response to a button click means a background task was started, not that it succeeded. Wait for the final result of the particular operation.

Clear Console removes visible entries from the current view. It does not delete Raspberry Pi logs, reset service errors or interrupt a task. Do not treat this button as a way to stop deployment.

Shutdown Server at the top closes the local Deployment Tool server after confirmation. It does not shut down Raspberry Pi or power down the vehicle. The panel stops responding until you start the tool again. Do not use it during an update as a Cancel button. This panel has no separate remote Pi shutdown or AI/Hailo installation command.

8. Video and other maintenance tools



Video & Streaming

Camera CSI Port

CSI Cam 0

Camera Sensor Type

AUTO (Detect)

Resolution

1920x1080

FPS

30

Video Bitrate

5 Mbps

Detect CSI Cameras

Low Latency Profile

Video & Streaming card preview; no camera test has been performed

Video & Streaming — camera fields

This card is visible for the Docker platform; selecting MCS hides it together with the RTK card. It is not a video player and does not show a live camera image.

- **Camera CSI Port:** CSI Cam 0, CSI Cam 1, /dev/video0 or /dev/video1.
- **Camera Sensor Type:** AUTO (Detect), imx219, imx708, ov5647, imx477 or imx519.
- **Resolution:** width and height, for example 1920x1080, with an x between the numbers.
- **FPS:** requested frames per second, for example 30.
- **Video Bitrate:** requested bitrate, for example 5 Mbps.

These are requested settings. The actual combination depends on the camera, driver and service on the Pi. Changing a form value does not confirm a new stream or measure actual FPS. Record the previous settings before changing several parameters.

Detect CSI Cameras

Detect CSI Cameras connects to the Pi and runs system camera detection. Read the console result and compare it with the sensor actually installed. This is not a universal scanner for USB, IP or HDMI capture sources. It does not automatically configure the entire transmission profile from the camera name. If the result is empty, check host camera tools, the ribbon cable, connector and power.

Low Latency Profile

Low Latency Profile enters **800x600, 30 FPS** and **2 Mbps**, then saves the tool settings. It does not itself send configuration to the running camera. It is not a guarantee of measured latency and does not improve the radio link. After selecting it, apply the appropriate camera-configuration update and check video in GCS.

Cam Config Only — applying stream settings

1. Check the Pi address and actual deployment type.
2. Set Resolution, FPS and Video Bitrate, or use Low Latency Profile.
3. Click **Cam Config Only**.
4. Wait for the final result. The video service is restarted, so the stream may disappear briefly.
5. Check the Video indicator, MediaMTX logs and the actual image in GCS.

This quick operation generates MediaMTX configuration, replaces `/etc/mediamtx.yml` and restarts MediaMTX. It applies resolution, FPS and bitrate; it is not a complete camera-driver or system-settings update. In particular, do not treat it as confirmation that every CSI Port and Sensor Type choice has been applied. The generated stream uses the Raspberry Pi Camera source; selecting `/dev/video0` alone does not confirm USB-camera configuration.

Backup Docker Vol — what the backup actually contains

Despite its **Backup Docker Vol** label, the current function archives the `rcsim_project` directory inside the Raspberry Pi user's home, then downloads the archive to the computer running Deployment Tool. It is not an automatic backup of every Docker volume, container image or the entire SD card. In the web version, it saves `rcsim_docker_backup.tar.gz` in the local tool user's home directory. It does not automatically appear in the phone's Downloads folder when the browser runs on a phone. Subsequent backups use the same filename; preserve the previous file under another name if needed.

1. Check the target host/account and available space on both ends.
2. Run Backup Docker Vol and wait for downloading to finish.
3. Read the file path in the final message.
4. Check the archive's presence and contents before considering it useful for recovery.

The panel provides no separate automatic restore button for this backup. When MCS is selected, the panel attempts to disable the backup button, but status refresh may enable it again. Its availability does not mean MCS backup is supported: the operation still uses the `rcsim_project` directory belonging to the Docker workflow described above.

Updating versus installing again

Docker Rebuild & Update runs the deployment workflow while skipping part of system configuration. **Hot Deploy Code (~30s)** provides a quick code/configuration replacement in an existing environment; for Docker, it requires a previously created container. The “~30s” label is approximate, not a guarantee of duration or uninterrupted control. **Fast Deploy (Skip OS Config)** means skipping system configuration, not verifying that the system does not need it. If dependencies, hardware or system requirements have changed, do not assume a quick replacement is sufficient. **FULL DEPLOYMENT** is the broader installation/configuration operation described in the deployment procedure. Choosing these buttons does not start AI training or provide separate Hailo-model management in this panel.

9. Troubleshooting and sources

- **RPI is Online, but the service is inactive:** check SSH, platform selection and App Logs; ping does not test the application.
- **RPI is Offline, but external SSH works:** the host may not respond to ping; use Test Connection.
- **MCS displays two identical statuses:** Core/Web refer to the same service here, not two independent tests.
- **Restart Service does not fix video:** restarting the main application does not restart MediaMTX; check its diagnostics and configuration.
- **Diagnostics finishes with OK, but some sections are empty:** inspect individual results and logs; final OK does not guarantee every command succeeded.
- **App Logs is empty:** confirm the correct platform; an empty result does not exclude messages on standard error.
- **Build Logs or backup availability changes in MCS:** status refresh may override button disabling. Build Logs reads the MCS log file, but backup still targets `rcsim_project`; an enabled button does not confirm that its function suits the platform.
- **Low Latency Profile did not change video:** it changes the saved form settings; apply camera configuration and check the stream in GCS.
- **A USB camera does not work after Cam Config Only:** the quick generator uses Raspberry Pi Camera; choosing a USB port alone does not configure that source.
- **The backup is missing from the phone:** the file is downloaded to the computer running the Deployment Tool server, at the console-reported path.
- **The page stops working after Shutdown Server:** start the local tool again; that button closes its server.
- **An operation takes longer than the button label suggests:** check its console stage and wait for the result rather than starting another deployment in parallel.

The illustrations identify panel controls; they do not confirm deployment to a physical Raspberry Pi. Sources read on 14 October 2026, relative to `RCSIM_deployment_tool/RCSIM_deployment_tool/`:

- `web/templates/index.html` and `web/static/js/main.js`: cards, buttons, camera preset, confirmations, console and status.
- `web_server.py`, lines 310–364, 446–624 and 638–645: monitoring, tasks and local-server shutdown.
- `core/deployment_logic.py`: stream configuration, camera detection, installation, updates, directory backup, diagnostics, restarts and logs. `core/service_monitor.py` belongs to the alternative desktop interface.