

Tier 3: ELRS/mLRS, CRSF/MAVLink and EdgeTX/ExpressLRS modifications

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

Contents

- 1. What Tier 3 covers 3
- 2. First select the actual signal path 3
- 3. Requirements and configuration record 3
- 4. CRSF Direct in GCS 4
 - 4.1. Nomad — preparing a specific CRSF variant 4
 - 4.2. GCS — USB, TCP and UDP step by step 5
 - 4.3. Backpack CRWF and Airlink — distinguish the protocols 6
- 5. MAVLink RF 6
 - 5.1. Serial and Airport connections 7
 - 5.2. mLRS/ELRS firmware and module recovery 7
- 6. EdgeTX modification: CRSF Trainer over USB-VCP 7
 - 6.1. Installation and returning to your own firmware 8
 - 6.2. Menus and trainer signal path 8
- 7. ExpressLRS ER Series modification: ER5C V2 example 8
 - 7.1. ER5C V2 GPS + IMU assembly 9
 - 7.2. Flashing ER5C V2 firmware 9
 - 7.3. Output functions and initial sensor checks 10
- 8. Diagnostics 10
 - 8.1. Steering wheel, pedals and Force Feedback 11
 - 8.2. Testing the complete setup before driving 11
- 9. Verification limits 12
- 10. Research sources 12

User manual: assembly, firmware, configuration and bench tests. Status: 14 October 2026. Based on local implementations, patches, test tools and documentation for specific firmware builds. No physical radio tests or compilation of every firmware build have been performed.

1. What Tier 3 covers

Tier 3 means control and telemetry through ELRS/mLRS RC systems, using CRSF or MAVLink interfaces and project modifications to EdgeTX and ExpressLRS, including those for ER5C V2. The name does not identify a single transmitter–receiver combination or a single baud rate.

An onboard Raspberry Pi is not required for this Tier. When radio is one of the interfaces of a Raspberry Pi platform, the complete platform belongs to Tier 4; configure its radio side using the appropriate variant in this manual.

Sound-card PPM belongs to Tier 0. An Arduino/RP2350 digital-to-PPM bridge belongs to Tier 1. Do not describe a transmitter with a CRSF-over-USB-VCP modification as a PPM bridge.

2. First select the actual signal path

- **CRSF Direct:** GCS → compatible CRSF interface → radio module → receiver. Return data depends on the sensor and firmware throughout the path.
- **MAVLink RF:** GCS → port or network bridge → ELRS/mLRS radio → MAVLink endpoint. The radio itself does not create sensor data.
- **EdgeTX CRSF VCP modification:** GCS → transmitter USB-VCP → trainer/mixer → RF module → receiver. FullDuplex adds return telemetry mirroring to USB.
- **ExpressLRS ER Series modification:** compatible receiver with additional GPS/IMU and reassigned pin functions.

CRSF is a frame format/interface, MAVLink is a message protocol, and ELRS/mLRS identify the radio system and its firmware. Compatibility on one segment does not establish compatibility across the entire path.

Video is not an automatic component of CRSF/MAVLink. A separate FPV system and HDMI/USB capture device can be configured as the GCS video source. A visible image does not confirm working control or telemetry.

3. Requirements and configuration record

Before connecting, record:

- the transmitter/module and receiver models;
- each device's firmware name and version;
- the GCS variant: CRSF Direct or MAVLink RF;
- the PC port and baud rate of the particular USB/UART segment;
- steering, throttle, arming and auxiliary-channel mapping;
- the source of every reading: GPS, IMU, battery and link statistics;
- the method of taking over control and receiver behavior after link loss.

Do not change firmware, pin assignments, baud rate and mapping simultaneously. After every change, check the nearest segment of the path separately.

4. CRSF Direct in GCS

CRSFTransceiver accepts a port, baud rate, transmission frequency and transport. Besides a serial port, it implements TCP, UDP and Backpack UDP variants. Their presence in the class does not mean an arbitrary radio module supports all of them.

The active GCS CRSF strategy transmits at a target frequency of 100 Hz. The transceiver class itself has different default arguments: 420000 bps and 250 Hz; the GCS factory has a baud-rate fallback of 400000 bps. None of these numbers is a universal radio requirement. The EdgeTX USB-VCP modification described later uses 115200 bps. Match the rate to the specific interface, not merely to the ELRS name.

Perform the first test with propulsion unable to start:

1. Start GCS following the separate GCS manual and select the vehicle profile.
2. Select the appropriate CRSF/Direct ELRS mode and the identified device port.
3. Set a compatible baud rate and keep the model disarmed.
4. Check channels in GCS, then the transmitter's channel monitor and the receiver.
5. Compare steering and throttle center, range and direction. Do not assume neutral throttle always corresponds to the same value in every model.
6. Test arming and control takeover only after the channels agree.
7. Interrupt the PC connection and, separately, the RF connection, and check actual receiver-output behavior.

When disarmed, CRSF Direct transmits neutral channels but sets CH5/AUX1 low. When armed, CH5 may be set high; this channel's default center value is replaced with a high value. Account for this role before assigning CH5 to another device, for example. This neither disconnects ESC power nor guarantees identical ARM semantics in every receiver.

4.1. Nomad — preparing a specific CRSF variant

ExpressLRS™

RadioMaster Nomad 2.4/900 TX
Firmware Rev. 4.0.1 EU868/CE_LBT

Hardware Layout

Upload target configuration (remember to press "Save Target Configuration" at the bottom of the page):

UPLOAD

or drop files here

CRSF Serial Pins		
RX pin	3	Pin used to receive CRSF signal from the handset
TX pin	1	Pin used to transmit CRSF telemetry to the handset (may be the same as the RX PIN)

Serial2 Pins		
RX pin		Serial2 RX - ESP32 targets only
TX pin		Serial2 TX - ESP32 targets only

Nomad WebUI example: ExpressLRS 4.0.1, RX pin 3 and TX pin 1

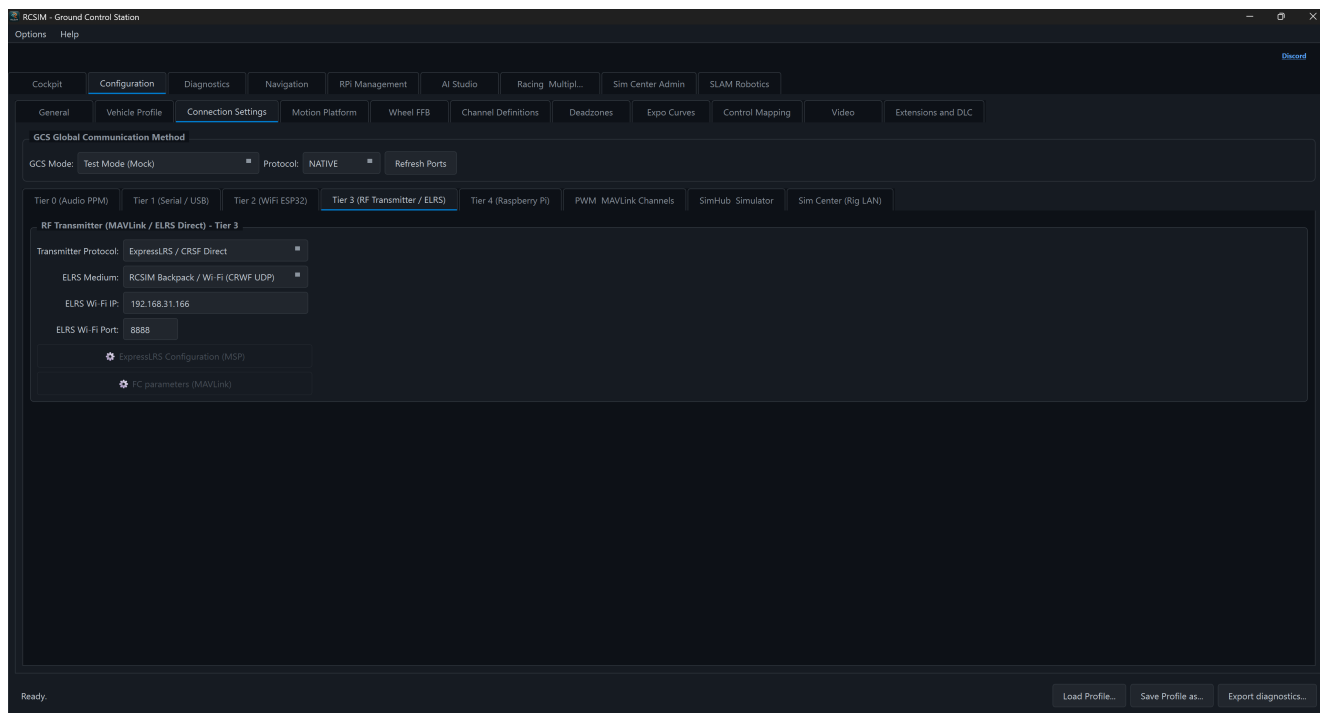
The project illustration shows **RadioMaster Nomad 2.4/900 TX**, firmware **4.0.1 EU868/CE_LBT**. In **Hardware Layout** → **CRSF Serial Pins**, **RX pin 3** and **TX pin 1** are entered. The project note specifies a change from 4/4 for this setup. This is not a CP2102 soldering diagram or a universal pinout for every Nomad revision.

1. First confirm the exact module model, current firmware and how its USB/UART interface is exposed.
2. Preserve the existing hardware configuration before changing pins.
3. Only for the compatible variant illustrated, set RX 3 and TX 1 under CRSF Serial Pins.
4. Use **Save Target Configuration** at the bottom of the page, then restart the module as its message instructs.

5. Check channel reception and telemetry return separately through the selected interface.

With an external UART adapter, connect host TX to device RX and host RX to device TX, with ground correctly connected. Module power and logic voltages must match the specific device's specifications. Do not connect an RS-232 output to a TTL UART input or power a high-power transmitter from an arbitrary adapter pin. This project does not provide a verified physical wiring diagram for every such adapter; without identifying the board pins, do not make connections using WebUI numbers alone. Before transmitting, attach the correct antenna and set RF parameters appropriate to the equipment and region.

4.2. GCS — USB, TCP and UDP step by step



Tier 3 settings: CRWF UDP example; Test Mode (Mock) remains active in this screenshot

Open **Configuration** → **Connection Settings** → **Tier 3 (RF Transmitter / ELRS)**. Opening the Tier 3 card alone does not change the global **GCS Mode** from Test Mode. Select a real CRSF Direct connection in GCS Mode and **ExpressLRS / CRSF Direct** under **Transmitter Protocol**.

- **USB Serial (COM)**: use Refresh Ports, select the actual Port and a compatible Baud. Close other applications using that COM port.
 - **Airlink / Wi-Fi (TCP)**: enter the bridge address in ELRS Wi-Fi IP and its TCP server port in ELRS Wi-Fi Port.
 - **Airlink / Wi-Fi (UDP)**: enter the address and port of the bridge receiving raw CRSF frames over UDP.
 - **RCSIM Backpack / Wi-Fi (CRWF UDP)**: uses the separate CRWF format described below; it is not interchangeable with ordinary UDP.
1. Select a medium compatible with the firmware at the other end, not just with the available network.
 2. For USB, check the port and baud rate; for networking, the IP and port. The default 192.168.4.1 and 8888 are application examples.
 3. Finish editing the fields, save the profile and return to **Cockpit**.
 4. Select **Manual (Driver)**, remain disarmed and click **Connect**.
 5. Check the channel monitor, then the actual receiver and telemetry return.
 6. If Connect merely opens UDP, do not treat that as confirmation of a response from the other device.

ExpressLRS Configuration (MSP) requires an active compatible CRSF Direct connection. Configuration is available only when the module responds to the required commands. The button does not install firmware and is unavailable for Backpack CRWF transport. **Raw IMU via CRSF (0x13)** opens information about the modification and requirements; it does not add a sensor to a stock receiver.

4.3. Backpack CRWF and Airlink — distinguish the protocols

CRWF requires a **modified RCSIM Backpack**, compatible **ExpressLRS 4**, **Link Mode: Normal**, EdgeTX **Master/CRSF** trainer mode and **TR** input mixes on the appropriate channels. This trainer variant differs from the USB-VCP setup described later; do not transfer mode names between them without checking. An ordinary stock Backpack does not become a CRWF bridge merely by selecting this option in GCS.

1. Confirm that a CRWF-compatible Backpack package for the device's exact target has been installed.
2. Prepare the EdgeTX model: Master/CRSF, a TR mix for each used channel and a control-takeover switch.
3. Check Wi-Fi connectivity, the Backpack address and listening port; enter them in GCS.
4. Select CRWF UDP, connect and check channels in the radio's monitor before testing the model.
5. Check telemetry return. After link loss or toggling head tracking, disconnect and reconnect RCSIM.

CRWF carries channels and telemetry, not CRSF/MSP configuration commands or MAVLink. Do not expect the parameter configurator to work through this transport. The transport session identifies a connection; it does not authenticate a user. The code attempts to renew the session after a transmission gap or loss of previously received telemetry; this does not replace an operator's failsafe and reconnection test. A complete matched Backpack firmware package or universal recovery procedure was not found in the local project. Installation or recovery requires the package and instructions for the exact target; do not use an ER5C binary or EdgeTX modification.

5. MAVLink RF

MAVLinkRFStrategy uses pymavlink and enforces MAVLink 2.0. The code has separate mechanisms for parameters, calibration, telemetry decoding and controller identification.

When choosing the configuration:

1. Check that the target device sends heartbeat and MAVLink messages rather than CRSF alone.
2. Match port parameters to the active radio firmware.
3. Distinguish the radio module's ID from the actual vehicle/controller. The code filters mLRS radio-module identifiers during routing.
4. Verify parameter and sensor availability from received messages, not merely from the presence of a COM port.
5. Check responses to port disconnection, RF loss and heartbeat loss as separate cases.

There is no single "MAVLink baud rate for everything." Rates on the two sides of a module may differ if its firmware provides the appropriate bridging. Each segment must match its endpoint.

5.1. Serial and Airport connections

Under **Transmitter Protocol**, select **MAVLink RF (Airport / Telemetry)** and the global MAVLink RF mode. **MAVLink Medium** selects Serial, Airport Wi-Fi TCP or UDP. Serial requires Port/Baud; TCP and UDP display **MAVLink Airport IP** and a port. For TCP, the field means the Airport server port; the default 5760 is an application value, not a rule for every bridge. For UDP, **Destination Port (Backpack Listen Port)** is the port receiving packets on the device, while **Local UDP Port (Backpack Send Port)** is the computer port receiving return telemetry. One example matching pair is destination 14555 and local 14550, if the Backpack is configured accordingly. A local value of 0 leaves port selection automatic; do not use it if the device sends exclusively to a specific PC port.

SysID refers to the target vehicle controller, not the COM number or radio module's own identifier. A value of 0 enables automatic target selection; do not assume every device uses SysID 1 or 10. **ArduPilot Telemetry Profile** reduces selected message rates after controller confirmation; it does not increase RF bandwidth or guarantee availability of every value.

1. Record the ports and baud rates on both sides of the bridge and the vehicle SysID.
2. Set the medium and corresponding fields, connect from the cockpit and wait for the vehicle heartbeat.
3. Check that the controller has been identified, not just the mLRS module.
4. Open **FC parameters (MAVLink)** only after obtaining an active, correct target.
5. Read parameters before changing them; preserve the previous values and change individual settings.
6. Check directions and RC override response on a secured bench, then the controller's ARM/DISARM confirmation.

GCS queues the newest control state for transmission at 20 Hz; the strategy accepts 1–18 RC override values. This does not describe the eight-channel MCS output: the two applications have separate implementations. ARM/DISARM in GCS is a MAVLink command to the correct endpoint and requires control authority; do not assume arming or motor shutdown without a controller response and observation of the model. Interruption of RC override, heartbeat loss and autopilot failsafe also require separate configuration and testing.

5.2. mLRS/ELRS firmware and module recovery

The Nomad name does not automatically select ELRS or mLRS. Check transmitter and receiver firmware, band, target and supported interface before attempting to change protocols. The local project contains GCS integration, but not a complete mLRS installation package for every Nomad/XR4 revision. There is no basis for specifying one flash command, memory address or BOOT sequence for all these devices. Before updating, preserve settings and prepare the original firmware for the exact target and its recovery procedure. After updating, recheck radio-pair compatibility, binding, ports and failsafe. Do not flash an ER5C V2 receiver modification onto a transmitter, Backpack or receiver with a different MCU.

6. EdgeTX modification: CRSF Trainer over USB-VCP

The project files are located in:

```
RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/EdgeTX CRSF VCP mod/
```

The local instructions distinguish FullDuplex builds from older simplex builds. According to those instructions, MT12 and Pocket were tested by the modification's author, while some other models are experimental. This research is not a repeat hardware test or confirmation of every binary's contents.

Before flashing, back up your firmware, model settings and SD card. Choose a build for the exact transmitter model; another radio's name is not a compatible substitute.

6.1. Installation and returning to your own firmware

The modification directory contains EdgeTX_v2.10_MODEL_CRSF_VCP_FullDuplex.bin files for MT12, Pocket, Boxer, TX16S, TX12, TX12MK2 and Zorro. TX12 and TX12MK2 have separate files. Do not choose by the transmitter's similar appearance.

1. Secure the model, disconnect propulsion and back up the transmitter configuration, firmware and SD card.
2. Select USB Storage (SD) or use a card reader; copy the correct .bin into FIRMWARE/ on the card.
3. Safely eject the storage and start the bootloader following the instructions for your radio model.
4. In the bootloader, select **Write Firmware**, the correct file and confirmation to write it.
5. Wait for completion, select **Exit** and check that the radio starts and the intended model is selected.
6. Prepare the trainer configuration in the next subsection and test the channel monitor with propulsion disconnected.

The bootloader-entry gesture depends on the transmitter; the two-trimmer description in an older note is not a recommendation for every model here. If the new build works incorrectly, use the bootloader and the preserved working build for the exact target, then restore settings from your own backup. If the bootloader is unavailable, use the radio-specific recovery procedure; this package is not a universal DFU guide for all seven models.

6.2. Menus and trainer signal path

Requirements recorded in crsf_vcp_test.py:

- transmitter USB mode: Serial/VCP;
- USB-VCP configured as CRSF Trainer;
- model trainer mode: CRSF;
- an active Trainer special function so channels reach the mixer.

Menu names and locations depend on the specific build. After changing settings, check the transmitter's channel monitor with RF disabled and propulsion disconnected.

1. In **SYS** → **Radio Setup**, set **USB Mode: VCP**, or Ask and select VCP/Serial when connecting the cable.
2. In **SYS** → **Hardware**, assign **USB-VCP: CRSF Trainer**.
3. In **MDL** → **Model Setup** → **Trainer**, set **Mode: CRSF** and the channels used, up to CH1–CH16.
4. In **Special Functions**, add Trainer, Value Sticks/Axis, Enable ON, preferably controlled by a deliberate takeover switch.
5. Check model mixes, directions and ranges, then the channel monitor.
6. In GCS, select CRSF Direct, USB Serial, the correct COM port and 115200 bps. Check transmission and telemetry return separately.

The FullDuplex modification mirrors existing telemetry. The transmitter's **Discover new sensors** function, if available in the particular build, discovers received sensors; it does not create an IMU or GPS measurement. If control works without return data, check for FullDuplex rather than legacy simplex and for active RF telemetry.

Both crsf_vcp_test.py and crsf_vcp_duplex_test.py actively transmit channels; they are not passive monitors. By default, the duplex variant transmits at 100 Hz and sweeps channels 1–4 through their full range while receiving frames. Do not run them on an active model assuming they only read telemetry.

7. ExpressLRS ER Series modification: ER5C V2 example

Source and artifact directory:

The repository contains instructions, binaries and an ExpressLRS v4.1.0 patch. Among other changes, the patch extends GPS, IMU support and pin configuration. Applying it to another upstream version requires separate verification; the file's presence does not mean a successful build on an arbitrary checkout.

The ER5C V2 variant instructions describe:

- CH1 as steering PWM;
- CH2 as ESC PWM;
- CH3 as the RX input for GPS-module TX;
- CH4/CH5 as SDA/SCL for a compatible IMU.

This allocation applies to the specified build and WebUI configuration. Do not automatically apply it to ER4, ER8, stock ER5C V2 or another hardware revision. Changing CH4/CH5 to I2C changes their role from ordinary PWM outputs. Check sensor power and logic levels in the documentation for the modules used.

The patch contains a SERIAL_RX_ONLY change that frees the TX pin for ESC PWM in the GPS variant. Do not remove it assuming it has no effect on propulsion.

7.1. ER5C V2 GPS + IMU assembly

The following pinout applies only to ER5C V2 with the described modification:

- CH1/GPIO0 signal → steering servo; retain the PWM function.
- CH2/GPIO1 signal → ESC; retain PWM if this is the propulsion output.
- GPS TXD → receiver CH3/GPIO3 signal; GPS RXD remains disconnected in the unidirectional variant.
- IMU SDA → CH4/GPIO9 signal; IMU SCL → CH5/GPIO10 signal.
- Receiver, GPS and IMU GND → common ground.

The servo connectors' + rail may carry BEC voltage that is not safe for every sensor. Power the GPS and IMU module according to their specifications; a bare MPU chip is not automatically a 5 V-tolerant module. I2C pull-ups and the UART signal must be compatible with 3.3 V logic. Read labels rather than relying on wire colors. Mount the IMU rigidly and record its axis orientation; a loose board can produce noise and incorrect FFB forces. Supported IMU families are MPU6050/6500/9250/9255, at 0x68 or 0x69; this does not mean arbitrary Grove IMUs are supported.

7.2. Flashing ER5C V2 firmware

The combined package is **ELRS_V4.1_RadioMaster_ER5Cv2_RCSIM_GPS_IMU**: .bin.gz for compatible WebUI, .bin for UART recovery. GPS-only and IMU-only are separate variants; do not interpret the old MPU9250_CRSF13 name as a complete description of the contents.

ER4/ER5A_C/ER6/ER6G/ER6GV/ER8/ER8G/ER8GV binaries are also present. Their presence does not establish the ER5C V2 pinout on those boards; ER5 uses ESP8285, while some ER6/ER8 models use ESP32.

1. Preserve your receiver settings and the original firmware for the exact model.
2. With propulsion disabled, start receiver Wi-Fi; the local instructions describe waiting approximately 60 s without a TX connection.
3. With factory settings, join **ExpressLRS RX**, password **expresslrs**, and open <http://10.0.0.1> or the actual address assigned to the receiver.
4. In **Update / Firmware Update**, select the correct ER5Cv2 .bin.gz and start Update.
5. Wait for the final result and restart; timing depends on the device. Do not disconnect power during writing.

6. Reopen WebUI and confirm firmware and output functions before connecting propulsion.

If ER5C V2 UART recovery is needed, use a TTL adapter with 3.3 V logic, adapter TX → receiver RX and adapter RX ← receiver TX, plus common ground. Temporarily disconnect GPS from the shared RX and the ESC from the shared TX/PWM. Identify the physical UART/BOOT points for the particular board; a CH number does not replace adapter and receiver documentation. Hold the appropriate BOOT control during power-up following the ER5C V2 instructions, and once the port is detected, run:

```
esptool.py --port COMX --baud 115200 --chip esp8266 write_flash -fm dout 0x000000
ELRS_V4.1_RadioMaster_ER5Cv2_RCSIM_GPS_IMU.bin
```

Run from the file's directory and replace COMX with the actual port. This command is for the raw ER5C V2 .bin, not .gz, an ESP32 MCU, a Nomad transmitter or the EdgeTX bootloader. After success, start the receiver normally, check configuration, and only then reconnect sensors and the ESC. Do not make connections without confirming physical access to UART/BOOT and the voltages.

7.3. Output functions and initial sensor checks

1. In WebUI **Connections / PWM Pin Functions**, set Output 1 and 2 to PWM compatible with the servo/ESC; start at 50 Hz if the devices support it.
2. Set **Output 3: Serial RX** and the main **Serial Protocol: GPS**.
3. Set **Output 4: I2C SDA**, **Output 5: I2C SCL**, and use SAVE below the relevant tables.
4. Restart the receiver and check that CH2 remains PWM.
5. Establish a compatible radio pair; check GPS and IMU return to GCS separately.

GPS must already be transmitting NMEA GGA/RMC/VTG or supported UBX. Automatic detection checks 115200, 9600, 38400, 57600, 230400 and 19200 bps. A detected baud rate does not mean a fix; position requires satellite reception and suitable conditions. Unidirectional GPS does not receive automatic-configuration commands. If you set CH2 to Serial TX and connect it to GPS RX, you lose the ESC output on CH2. Do not attempt to use the same pin simultaneously as UART TX and propulsion PWM.

Raw IMU uses frame **0x13**, an **18-byte** payload and source **crsf_raw**; it contains sample time, gyro, acceleration and temperature, without a magnetometer. **0x1E** contains orientation, **0x02** GPS and **0x03** GPS time. Do not look for the obsolete 0x86 decoder. The IMU loop targets 100 Hz, but actual reception depends on telemetry bandwidth; RF frequency is not the frequency of new samples. An I2C error should not be interpreted as a valid artificial 1 g reading.

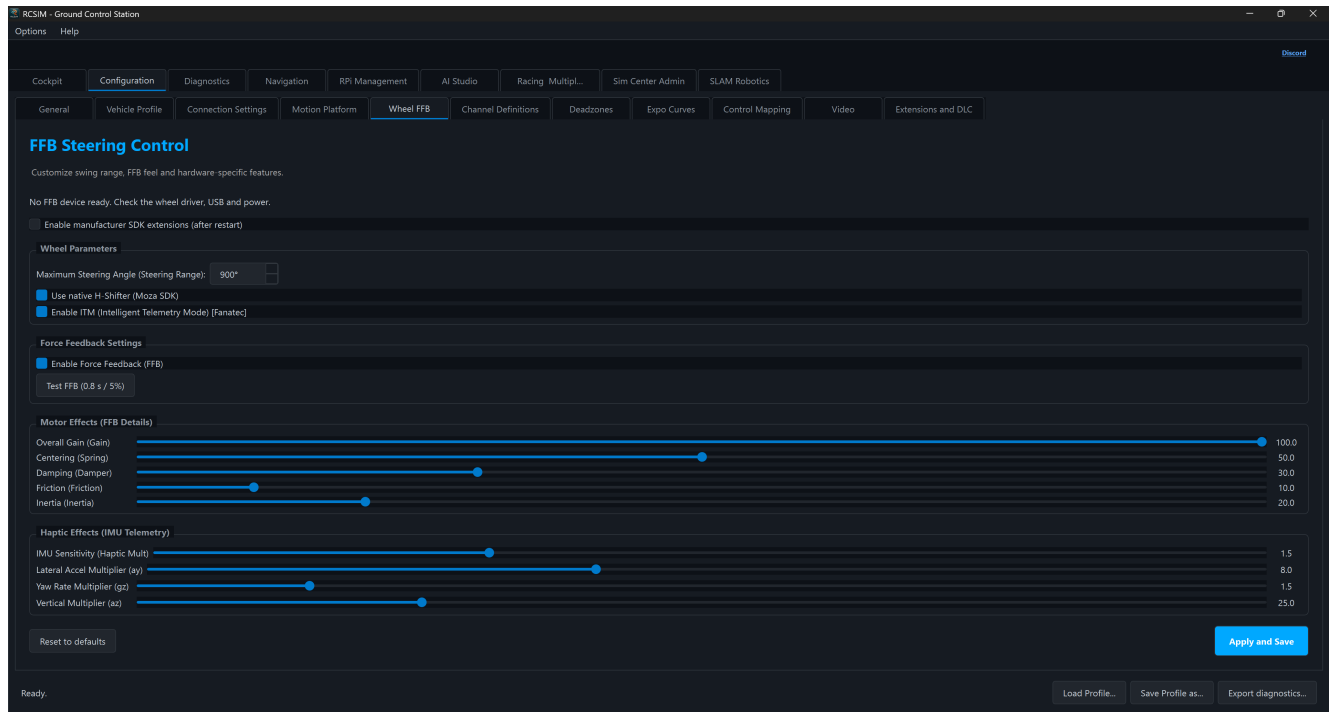
Receiving GPS and IMU does not establish ready-to-use autonomy. ER5C V2 with this modification remains a Tier 3 setup. Confirm telemetry coverage from frames actually received in GCS/MCS.

8. Diagnostics

- **COM is present, but no channels:** check protocol, baud rate, USB-VCP/Trainer and an active Trainer function.
- **Channels work, but telemetry is missing:** check FullDuplex rather than simplex, the telemetry path, sensor and frame type.
- **GPS works, but the ESC does not:** check the build and Serial TX/PWM conflict on CH2.
- **No IMU:** check CH4/CH5 functions, address and I2C levels; do not blindly change them to PWM.
- **RF works, but there is no MAVLink target:** check the actual controller heartbeat and SysID, not just the radio module.
- **Disconnecting the PC produces unexpected throttle:** check the entire trainer, mixer and receiver path; the GCS ARM state alone is insufficient.

- **CRWF does not open the configurator:** this is a medium limitation; MSP configuration is unavailable through it.
- **UDP works in only one direction:** check destination and local ports separately, bridge settings and the PC firewall.

8.1. Steering wheel, pedals and Force Feedback



Wheel FFB panel: example settings; the screenshot reports no ready FFB device

1. In **Channel Definitions**, set the steering channel as bidirectional and propulsion according to the model's electronics.
2. For two separate pedals controlling one ESC, select Split Axis; assign Throttle and Brake separately in **Control Mapping**.
3. Set directions using Invert, and check neutral throttle and ranges with propulsion disconnected.
4. Adjust Deadzones and Expo only after assigning axes correctly; save the profile.
5. In **Wheel FFB**, limit wheelbase force in hardware, set a low positive Gain and run **Test FFB (0.8 s / 5%)** without needing telemetry.
6. Check the IMU source and freshness in Diagnostics before adding ay, gz, az and Haptic Mult multipliers separately.

The full 0x13 IMU frame provides different data from orientation alone in 0x1E. With orientation alone, do not expect actual accelerometer bump measurements. Dynamic forces disappearing after fresh telemetry is lost is expected; do not compensate by increasing Gain. The GCS manual contains the full description of every slider and button.

8.2. Testing the complete setup before driving

1. Check mounting, antennas, voltages, mapping and recorded firmware versions with propulsion disconnected.
2. Confirm manual channels in GCS, radio and receiver, and the correct neutral gear.
3. Check ARM/DISARM and the CH5 role, then the physical switch for transmitter takeover.

4. Interrupt PC USB/network, host power and RF separately; observe the actual ESC output.
5. Restore the connection without touching throttle and confirm no unintended propulsion starts.
6. Check GPS, IMU, battery and link statistics; a missing sensor is not a zero measurement.
7. Only after passing these tests, reconnect propulsion and perform a short controlled drive with a means of stopping.

There is no single failsafe value appropriate for every ESC, model and autopilot. ELRS/EdgeTX configuration and local GCS state are different layers; verify each on your own path.

In the diagnostic log, record firmware versions, protocol, port, segment settings and when data disappeared. Remove identifiers/logins and other information you do not want to share.

9. Verification limits

The presence of strategy implementations and test tools, and the contents of project patches, were verified. RF range, end-to-end response time, correctness of every binary, compatibility with every hardware revision and a particular ESC's failsafe behavior were not verified.

Before driving, these tests must pass: neutral throttle, channel directions, control takeover, PC loss, RF loss and link return without unexpected propulsion startup.

10. Research sources

Paths are relative to the project directory; line numbers refer to the state checked during preparation.

- RCSIM_PC/pc_app/core/comm_manager.py:55 — communication-strategy registry.
- RCSIM_PC/pc_app/core/comm/crsf_transceiver.py:44 — CRSF parameters and transport selection.
- RCSIM_PC/pc_app/core/comm/rf_link_strategy.py:43 — 100 Hz frequency used by the GCS strategy.
- RCSIM_PC/pc_app/gui/tabs/connection_sections.py:196 and connection_behavior.py:125 — CRSF/MAVLink fields and media.
- RCSIM_PC/pc_app/core/comm/transport/backpack_transport.py:25 — CRWF session and lack of MSP configuration.
- RCSIM_PC/pc_app/core/comm/mavlink_rf_strategy.py:839 and :917 — ARM/DISARM and RC override queue.
- docs/manual/Zdjęcia/Nomad/Info.md and image.png — RX3/TX1 mapping example, not a universal electrical diagram.
- RCSIM_PC/pc_app/core/comm/crsf_transceiver.py:125 — transceiver arming and disarming.
- RCSIM_PC/pc_app/core/comm/mavlink_rf_strategy.py:20 — MAVLink 2.0 and radio-module identifiers.
- RCSIM_PC/pc_app/core/comm/mavlink_identity.py:1 — MAVLink endpoint identification.
- RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/EdgeTX CRSF VCP mod/README_PL.md:1 — modification scope, models and stated test status.
- RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/EdgeTX CRSF VCP mod/crsf_vcp_test.py:2 — transmitter-configuration requirements.
- RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/EdgeTX CRSF VCP mod/crsf_vcp_duplex_test.py:182 — arguments, 115200 bps port and active test transmission.

- `RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/ExpressLRS_ER_Series_Tier_3/mod/README_PL_RadioMaster_ER5Cv2_GPS.md:13` — GPS/IMU variant and connector mapping.
- `RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/ExpressLRS_ER_Series_Tier_3/mod/elrs_v4.1_er5cv2_gps_imu.patch:7367` — RX changes and freeing the pin for PWM.