Common Operations

Running in Development

```
bun run tauri dev
```

This command does the following in parallel:

- Starts the Vite/SvelteKit dev server on `http://localhost:1420`
- Compiles the Rust backend (first run takes several minutes)
- Opens the Tauri application window

The frontend supports hot module replacement — changes to `.svelte` and `.ts` files appear immediately without restarting. Rust changes require a recompile, which Tauri handles automatically but takes longer.

> **First build warning:** the initial `cargo build` downloads and compiles all Rust dependencies including GStreamer bindings. This can take 5–15 minutes depending on your machine. Subsequent builds are fast due to incremental compilation.

Building for Production

> One of Tauri's dependencies, libc (the C standard library), is forward compatible but not backward compatible

The demo laptop has an old verison of linux so you need to use docker to build the app if you want to use it in it.
The first time you build you have to build docker first:

```
sudo docker build -t tauri-ubuntu2204 .
```

For building the app in linux the subsequent times for the demo laptop use the command:

```
sudo docker run --rm \
    -v $(pwd):/app \
    -v tauri-cargo-cache:/root/.cargo/registry \
```

```
    tauri-ubuntu2204
```

> You don't have to docker build every time, just the first time and if you change anything in the dockerfile

Testing Without Rover Hardware

You do not need a physical rover to develop or test the UI. The backend includes a full simulator.

Video Feeds

Fake camera

`fake_camera_gstreamer/` — a GStreamer-based test source that sends H.264 RTP streams on the expected UDP ports (4500, 4501, 4502). It can be run from by running the following command from `erc-software-basestation/fake_camera_gstreamer/`

```
cargo run --bin fake_camera_gstreamer
```

Stream from Webcam

To test the video feed with your webcam open a terminal and use the following command.

Linux

```
gst-launch-1.0 v4l2src ! videoconvert ! x264enc tune=zerolatency bitrate=800 speed-preset=ultrafast ! rtph264pay ! udpsink host=127.0.0.1 port=4500
```

Windows

```
gst-launch-1.0 ksvideosrc ! videoconvert ! x264enc tune=zerolatency bitrate=800 speed-preset=ultrafast ! rtph264pay ! udpsink host=127.0.0.1 port=4500
```

Dummy data streams (rover telemetry)

Once the app is running, go to `/settings` and use the simulator controls:

- **Start dummy general stream** — starts the full multi-stream simulator sending fake IMU, GPS, arm, drive, and sensor data to the app over UDP. Use this to test all telemetry UI at once.
- **Start dummy IMU stream** — starts an IMU-only stream with no jitter or packet loss. Use this for isolated IMU component testing.
- **Stop dummy general/IMU stream** — stops whichever simulator is running.

The simulator runs inside the Rust backend so it works regardless of whether a rover is connected.

Connecting to the Rover

> **TODO**

The base station listens for incoming UDP packets on the address set on lib.rs, must set static address in laptop settings

---

Revision #17
Created 2026-04-15 09:07:39 UTC by Candela Cimadevilla Gonzalez
Updated 2026-05-25 10:05:41 UTC by Candela Cimadevilla Gonzalez