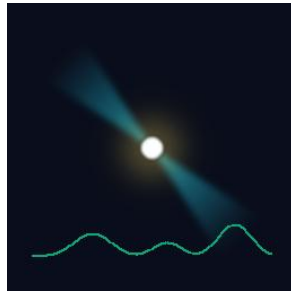


DSES Spectrum Analyzer

Installation Guide



Version 1.0.0

Richard M Hambly (K0GD)

DSES

June 2026

Contents

Contents	2
DSES Spectrum Analyzer — Installation Guide	4
1. What the recipient needs to know first	4
System requirements	4
2. Installing Radioconda	5
2.1 Windows 11	5
2.2 Linux	5
2.3 macOS (Intel and Apple Silicon)	6
2.4 Add the packages the analyzer needs (all platforms)	7
3. Installing the DSES Spectrum Analyzer	7
3.1 Windows 11	8
3.2 Linux	8
3.3 macOS	9
3A. Extra setup for SDRPlay receivers (RSP1A / RSP1B / RSPduo / RSPdx)	9
3A.1 Install the SDRplay API	10
3A.2 Install the SoapySDRPlay module	10
4. First-run checklist	12
Recommended one-time VOLK tuning	12
5. Updating to a new version	13
Auto-update notifications	13
Performing the update	13
What is preserved across versions	13
Backing up your settings	13
6. Troubleshooting	14
“Radioconda not found in standard locations”	14
“Missing required packages (PySide6, pyqtgraph, and/or scipy)” — or `ModuleNotFoundError: No module named 'PySide6'`	14
“No UHD Devices Found”	14
RTL-SDR not detected (or `[ERROR] ... librtlsdr.2.dylib` / `librtlsdr.so` at startup)	15
Persistent “O” overflows at high sample rates	15

Spectrum looks flat at -140 dB	16
macOS: “developer cannot be verified”	16
Settings won’t persist.....	16
7. Appendix A — Settings file reference	16
8. Appendix B — Device selection and SigMF playback.....	17
Choosing among multiple radios	17
SigMF playback mode (no-device fallback).....	17
9. Appendix C — Operating Guide	18
Starting up — device selection	18
Playback mode	18
Sidebar controls (right side).....	19
Spectrum (top plot).....	20
Waterfall (bottom plot).....	21
Persistence	21
Update checks.....	21
Tips for pulsar work	21

DSES Spectrum Analyzer — Installation Guide

Version 1.0.0 Author: Richard M Hambly (K0GD) — rick@cncssys.com License: GPL-3.0-or-later

This document covers:

- How to install the DSES Spectrum Analyzer on Windows 11, Linux, and macOS (Intel and Apple Silicon).
- How to update to a new version.
- Common troubleshooting and a reference for advanced settings.

The application itself is one Python file plus a few launchers and an icon. The heavy machinery — GNU Radio, UHD, SoapySDR, PySide6, NumPy, SciPy — is supplied by **Radioconda**, which you install separately as a one-time prerequisite.

Supported radios. The program is designed around the Ettus USRP B210 but also drives any SoapySDR-compatible receiver: SDRPlay RSP1A / RSP1B / RSPduo / RSPdx (with one extra setup step — see §3A), RTL-SDR, HackRF, Airspy / Airspy HF+, BladeRF, LimeSDR, and PlutoSDR. The sample-rate combo and gain slider adapt automatically to whichever device you pick at startup.

1. What the recipient needs to know first

Each recipient performs **two installs**, in this order (plus a third optional step if you're using SDRPlay):

1. **Install Radioconda** (one-time). Radioconda is a curated conda distribution that bundles GNU Radio, UHD, SoapySDR, and the surrounding scientific Python stack for software-defined radio. Use a current build — releases are date-stamped (e.g. 2025.03.14) and any recent one ships UHD ≥ 4.8 , which the B210 needs. Download from the releases page:

- **Primary:** <<https://github.com/radioconda/radioconda-installer/releases>>
- **Backup:** <<https://github.com/ryanvolz/radioconda/releases>> (redirects to the primary)

Per-OS file names and steps are in §2 below. Then add three packages the app needs that Radioconda doesn't bundle — pyside6, pyqtgraph, scipy — with one `conda install` (§2.4). (§2)

2. **Install this app**, which is just a small zip of Python and launcher scripts. (§3)
3. **If using an SDRPlay receiver:** after the two installs above, do the extra SDRPlay setup in §3A (install the SDRplay API + the SoapySDRPlay module). It comes last because it copies a file from the app's zip into Radioconda's folders. Other supported radios need no extra setup.

Updates to the app afterwards are a small zip replace — Radioconda does not need to be re-installed for every release.

System requirements

Component	Minimum
OS	Windows 11; Ubuntu/Debian/Fedora/Arch

	(current); macOS 11 Big Sur or newer
CPU	x86_64 or Apple Silicon (M1/M2/M3/M4)
RAM	4 GB free
USB	One USB 3.0 port (5 Gbps) for the B210
Hardware	Ettus USRP B210 or any SoapySDR-supported receiver (SDRPlay RSPx, RTL-SDR, HackRF, Airspy, BladeRF, LimeSDR, PlutoSDR). With one radio attached the app uses it silently; with several, a picker appears. See §8 for details.
Disk	~3 GB for Radioconda, plus ~1 MB for this app
Display	1280 × 800 minimum; 14" MacBook Pro is the layout target

2. Installing Radioconda

2.1 Windows 11

- Download the Windows installer — the asset named ``radioconda-<version>-Windows-x86_64.exe`` (e.g. `radioconda-2025.03.14-Windows-x86_64.exe`) — from the Radioconda releases page. Pick the newest release and grab the `...-Windows-x86_64.exe` file.
 - Primary:** <https://github.com/radioconda/radioconda-installer/releases>
 - Backup:** <https://github.com/ryanvolz/radioconda/releases> (redirects to the primary)
- Double-click the installer. Accept defaults; the installer offers to put Radioconda under `C:\ProgramData\radioconda` (system-wide) or `%LOCALAPPDATA%\radioconda` (per-user). Either works; the app's launcher checks both.
- If you have a radio, plug it into a USB 3 port. A B210 silently installs its UHD driver from Radioconda the first time it's connected (no prompt).
- (Optional) Confirm your radio is detected — open the **Anaconda Prompt (Radioconda)** shortcut from the Start menu and run the command for your device:
 - B210 / USRP:** `uhd_find_devices` — lists the unit with its own serial number.
 - RTL-SDR, HackRF, Airspy, BladeRF, LimeSDR, PlutoSDR:** `SoapySDRUtil --find` — lists the device under its driver name.
 - SDRPlay (RSPx):** do §3A first, then `SoapySDRUtil --find` shows it as `driver = sdrplay`.
 - No radio yet:** skip this — the app starts in playback mode using the bundled sample file (see §8), which is ideal for training.

2.2 Linux

- Download the Linux installer — the asset named ``radioconda-<version>-Linux-x86_64.sh`` (e.g. `radioconda-2025.03.14-Linux-x86_64.sh`) — from the releases page:
 - Primary:** <https://github.com/radioconda/radioconda-installer/releases>

- **Backup:** <<https://github.com/ryanvolz/radioconda/releases>> (redirects to the primary)

(ARM and POWER builds — ...-Linux-aarch64.sh, ...-Linux-ppc64le.sh — are published too if you're on one of those.)

9. From a terminal, run the file you downloaded (substitute the real version):

```
bash radioconda-<version>-Linux-x86_64.sh
```

Accept the license and let it install to ~/radioconda (the launcher checks this path automatically).

10. **USB permissions (B210 / USRP only).** If you have a B210, activate the Ettus udev rules so the device is reachable without root:

```
sudo cp ~/radioconda/lib/uhd/uhd-usrp.rules /etc/udev/rules.d/  
sudo udevadm control --reload-rules && sudo udevadm trigger
```

Without this you'll get permission errors when the B210 is plugged in. Then add yourself to the usrp (or plugdev) group if your distro uses one, and log out/in.

Other USB SDRs (RTL-SDR, HackRF, Airspy, etc.) have their own udev rules, usually installed with the device's own package. SDRPlay uses its API service (see §3A) and needs no udev rules.

11. (Optional) Confirm your radio is detected, using the command for your device:

```
~/radioconda/bin/uhd_find_devices      # B210 / USRP  
~/radioconda/bin/SoapySDRUtil --find  # RTL-SDR, HackRF, Airspy, SDRPlay (after §3A), ...
```

No radio yet? Skip this — the app starts in playback mode from the bundled sample (see §8).

2.3 macOS (Intel and Apple Silicon)

Download from the releases page:

- **Primary:** <<https://github.com/radioconda/radioconda-installer/releases>>
- **Backup:** <<https://github.com/ryanvolz/radioconda/releases>> (redirects to the primary)

Radioconda publishes builds for both Mac architectures — pick the one matching your CPU (uname -m prints x86_64 or arm64):

- **Intel:** radioconda-<version>-MacOSX-x86_64.sh (e.g. radioconda-2025.03.14-MacOSX-x86_64.sh)
- **Apple Silicon (M1/M2/M3/M4):** radioconda-<version>-MacOSX-arm64.sh

(Each Mac build also comes as a double-click .pkg graphical installer — ...-MacOSX-x86_64.pkg / ...-MacOSX-arm64.pkg — if you prefer that over the shell script.)

12. From a terminal, run the file you downloaded (substitute the real version + arch):

```
bash radioconda-<version>-MacOSX-<arch>.sh
```

13. Accept the default install location (~/.radioconda).

14. If you have a radio, plug it in. macOS needs no driver install for the B210.

15. (Optional) Confirm your radio is detected, using the command for your device:

```
~/radioconda/bin/uhd_find_devices      # B210 / USRP
```

```
~/radioconda/bin/SoapySDRUtil --find  # RTL-SDR, HackRF, Airtyspy, SDRPlay (after §3A), ...
```

No radio yet? Skip this — the app starts in playback mode from the bundled sample (see §8).

16. **Quarantine note.** If you ever see a Gatekeeper “cannot verify developer” dialog on a Radioconda binary, run:

```
xattr -dr com.apple.quarantine ~/.radioconda
```

2.4 Add the packages the analyzer needs (all platforms)

Stock Radioconda bundles GNU Radio, UHD, and SoapySDR, but **not** the GUI and plotting packages this app uses (pyside6, pyqtgraph, scipy). Install them once into Radioconda — this is a one-time step; future app updates don’t repeat it.

Open an **activated Radioconda shell**:

- **Windows:** the **Anaconda Prompt (Radioconda)** shortcut in the Start menu.
- **Linux / macOS:** any terminal whose prompt shows (base) (or run `source ~/.radioconda/bin/activate`).

Then run:

```
conda install -c conda-forge pyside6 pyqtgraph scipy
```

Accept the prompt; conda downloads and installs the three packages. If the launcher later reports “Missing required packages,” this is the step that was skipped.

3. Installing the DSES Spectrum Analyzer

Download the application from the distribution site. You can either use the direct links below, or browse the folder <https://gpstime.com/sw_distribution/b210_sa/> and pick the newest `dses-spectrum-analyzer-*.zip`:

- **Application (zip):** <https://gpstime.com/sw_distribution/b210_sa/dses-spectrum-analyzer-1.0.0.zip>
- **This guide (PDF):**
<https://gpstime.com/sw_distribution/b210_sa/DSES_RFI_Spectrum_Analyzer_Installation.pdf>

The zip is roughly 60 MB — it includes a short sample recording so the program can run in playback mode when no radio is attached. (Replace 1.0.0 in the link with a newer version number if a later release has been published.)

3.1 Windows 11

17. Extract the zip anywhere you have write permission. A common choice is Documents\DSES-Spectrum-Analyzer. The extracted folder will be `dses-spectrum-analyzer-1.0.0\` and will contain `dses_spectrum_analyzer.py`, `launcher.bat`, `launcher.ps1`, `install-shortcut.ps1`, `LICENSE`, the `icons\` folder, and this guide.
18. Double-click `launcher.bat` to start the application.
19. The first time you run it, the launcher searches for Radioconda in this order: the `RADIOCONDA_ROOT` environment variable, any currently-activated conda env, `%LOCALAPPDATA%\radioconda`, `C:\ProgramData\radioconda`, `%USERPROFILE%\radioconda`, a cached config file, then `conda info -base` if conda is on PATH. If none of these find a working install, you'll get a prompt asking for the path; type it in and the launcher remembers it for next time.
20. (Optional) Create a desktop shortcut (with the app's pulsar icon) by running `install-shortcut.ps1`. The reliable way — which works regardless of your PowerShell execution policy — is to open PowerShell in the extracted folder and run:

```
powershell -NoProfile -ExecutionPolicy Bypass -File .\install-shortcut.ps1
```

It prints `Created: ...\DSES Spectrum Analyzer.lnk`. (Right-clicking the script → “Run with PowerShell” also works *if* your machine's execution policy allows local scripts; if nothing appears, the policy blocked it — use the command above instead.) The shortcut is named **DSES Spectrum Analyzer** and launches the app via `launcher.ps1`.

After the first run, the app's window opens with the spectrum and waterfall plots. Tuning, sample-rate, gain, and recording controls are in the sidebar on the right.

3.2 Linux

21. Extract the zip somewhere under your home directory where you have write permission — a conventional spot for a per-user app is `~/Applications`. For example:

```
mkdir -p ~/Applications && cd ~/Applications
unzip ~/Downloads/dses-spectrum-analyzer-1.0.0.zip
cd dses-spectrum-analyzer-1.0.0
chmod +x launcher.sh
```

The extracted `dses-spectrum-analyzer-1.0.0/` folder contains `dses_spectrum_analyzer.py`, `launcher.sh`, `dses-spectrum-analyzer.desktop`, `LICENSE`, the `icons/` folder, and this guide. (Avoid system locations like `/opt` unless you extract with `sudo` — keeping it in your home directory avoids permission issues.)

22. Launch:

```
./launcher.sh
```

23. (Optional) Install a desktop entry so the app appears in your application menu:

```
INSTALL_DIR="$(pwd)"  
  
sed "s|__INSTALL_DIR__|$INSTALL_DIR|g" dses-spectrum-analyzer.desktop \  
    > ~/.local/share/applications/dses-spectrum-analyzer.desktop  
  
update-desktop-database ~/.local/share/applications/ 2>/dev/null || true
```

(The .desktop file uses __INSTALL_DIR__ as a placeholder so the launch path is correct on whichever machine installs it. The sed line substitutes the current directory into the placeholder.)

3.3 macOS

The steps are the same as Linux. Extract it into your personal `~/Applications` folder (Finder shows it as your own Applications folder, separate from the system `/Applications`):

```
mkdir -p ~/Applications && cd ~/Applications  
  
unzip ~/Downloads/dses-spectrum-analyzer-1.0.0.zip  
  
cd dses-spectrum-analyzer-1.0.0  
  
chmod +x launcher.sh  
  
./launcher.sh
```

Keeping it under your home directory (rather than the system `/Applications`) avoids permission prompts and Gatekeeper friction.

On Apple Silicon, **make sure** you installed the arm64 build of Radioconda. Mixing an x86_64 Radioconda with a native arm64 Python or Qt will produce confusing errors at startup. There is no Rosetta-only step required if both halves match the CPU.

If the Finder warns about an unidentified developer when running `launcher.sh`, clear the quarantine attribute on the unzipped folder:

```
xattr -dr com.apple.quarantine dses-spectrum-analyzer-1.0.0
```

3A. Extra setup for SDRPlay receivers (RSP1A / RSP1B / RSPduo / RSPdx)

Do this only if you're using an SDRPlay receiver — and only *after* you've installed Radioconda (§2) and extracted this app (§3), because the steps below put files into Radioconda's folders and use a file that ships inside the app's zip. B210, RTL-SDR, HackRF, Airspy, BladeRF, LimeSDR, and PlutoSDR users can skip this section entirely.

SDRPlay needs two extra pieces: the manufacturer's API driver, and the SoapySDRPlay module that bridges that driver to the SoapySDR layer the program uses.

3A.1 Install the SDRplay API

Download and install it from <<https://www.sdrplay.com/api/>> — pick the installer for your OS.

- **Windows:** run the .exe installer; accept defaults. A reboot may be needed.
- **Linux:** run the .run installer with sudo.
- **macOS:** run the .pkg installer.

After installation, verify the API can see your unit with the SDRplay Service / Status app (Windows) or SDRplayService (Linux/macOS). You should see your RSP listed.

3A.2 Install the SoapySDRPlay module

This module is **not** available through `conda install` on any platform, so the steps differ by OS.

Windows — copy the pre-built module that ships in this distribution:

The extracted release folder (from §3) contains `sdrplay\sdrPlaySupport.dll`. Copy it into Radioconda's SoapySDR module directory. That directory is under `C:\ProgramData`, so you need an **Administrator** PowerShell (right-click Windows PowerShell → "Run as administrator"):

```
Copy-Item ".\sdrplay\sdrPlaySupport.dll" `
"C:\ProgramData\radioconda\Library\lib\SoapySDR\modules0.8\" -Force
```

(Run it from the extracted release folder, or give the full path to the DLL.) If your Radioconda is installed somewhere else, adjust the path — the target is always `...\Library\lib\SoapySDR\modules0.8\`.

Then start the SDRplay API service (also from the Administrator PowerShell — the installer leaves it stopped):

```
Set-Service SDRplayAPIService -StartupType Automatic
Start-Service SDRplayAPIService
```

Linux / macOS — build the module from source against Radioconda.

You might expect `apt install soapysdr-module-sdrplay` (Linux) or `brew install soapysdrplay3` (macOS) to work, but they **don't** for this app: those package managers build the module against the *system* SoapySDR and install it into a system directory. Radioconda only scans its own modules dir (`~/radioconda/lib/SoapySDR/modules0.8/`) and loads its own SoapySDR library, so a system-built module is never found. The module has to be compiled against Radioconda's SoapySDR and dropped into Radioconda's modules dir — which is a short one-time build.

First install the build tools (a C++ compiler, CMake ≥ 3.10, and Git):

```
macOS:          xcode-select --install      # C++ toolchain (if not already present)

                brew install cmake git

Debian / Ubuntu: sudo apt install -y build-essential cmake git

Fedora:          sudo dnf install -y gcc-c++ cmake git
```

Then clone, configure against Radioconda's SoapySDR + the SDRplay API from §3A.1, build, and copy the module into place:

```
git clone --depth 1 https://github.com/pothosware/SoapySDRPlay3.git /tmp/SoapySDRPlay3

cmake -S /tmp/SoapySDRPlay3 -B /tmp/SoapySDRPlay3/build \
  -DCMAKE_BUILD_TYPE=Release \
  -DSoapySDR_DIR="$HOME/radioconda/share/cmake/SoapySDR" \
  -DCMAKE_PREFIX_PATH="$HOME/radioconda" \
  -DCMAKE_POLICY_VERSION_MINIMUM=3.5 \
  -DCMAKE_OSX_ARCHITECTURES=arm64

cmake --build /tmp/SoapySDRPlay3/build -j

cp /tmp/SoapySDRPlay3/build/libsdrrPlaySupport.so ~/radioconda/lib/SoapySDR/modules0.8/
```

Notes:

- `-DCMAKE_OSX_ARCHITECTURES=arm64` is **macOS only**, and assumes Apple Silicon with the arm64 Radioconda (the normal case). On an Intel Mac use `x86_64` instead; on Linux **drop that line entirely**.
- `-DCMAKE_POLICY_VERSION_MINIMUM=3.5` is only needed if Homebrew/your distro gave you **CMake 4.x** (it refuses the project's older `cmake_minimum_required` without it). Harmless on CMake 3.x — leave it in.
- If `cmake` reports it can't find `sdrrplay_api`, the SDRplay API from §3A.1 isn't installed where CMake looks (`/usr/local/include` + `/usr/local/lib`); reinstall it.
- The configure step should print `Found libsdrrplay: ... and Module sdrrPlaySupport configured...`; the build ends with `Built target sdrrPlaySupport`.

Verify it loaded (any OS, from a Radioconda prompt):

```
SoapySDRUtil --info
```

...should list `sdrrplay` in the "Available factories" line. If it doesn't:

- Re-check that the SDRplay API from §3A.1 installed correctly.
- On Windows, confirm the DLL landed in `modules0.8\` and the SDRplayAPIService is running (Get-Service SDRplayAPIService).
- On Linux/macOS, confirm `libsdrPlaySupport.so` is in `~/radioconda/lib/SoapySDR/modules0.8/` and was built against Radioconda (not a system SoapySDR).

When you launch the spectrum analyzer with an RSP attached, it appears in the device picker as e.g. RSP1B – 240513BE60 [sdrplay].

4. First-run checklist

Regardless of OS, before declaring the install good:

- The window title bar reads DSES Spectrum Analyzer – v1.0.0 – <radio model> – <serial> (e.g. ... – USRP B210 – 3273A91 or ... – RSP1B – 240513BE60). The version number must match the bundle you installed; the radio portion confirms which device is being used.
- Pull up **Help** → **User Guide** from the menu bar. The guide should open.
- Pull up **Help** → **About**. The author, version, license, and the path to the settings file should be readable.
- The spectrum plot should show live data (not a flat line at –140 dB). If it's flat, the radio isn't streaming — see Troubleshooting §6.
- Click the **408 MHz** preset under Pulsar Band. The center frequency should retune and the trace should redraw.

Recommended one-time VOLK tuning

GNU Radio's number-crunching uses **VOLK**, which can profile your CPU once and pick the fastest SIMD kernels (AVX2, NEON, etc.) for the rest of your machine's life. Without the profile, you'll see one of these warnings every time you launch the app:

```
[WARNING] SoapyVOLKConverters: no VOLK config file found.
```

```
Run volk_profile for best performance.
```

```
volk: no arch found, returning generic impl
```

To run it once and silence the warning:

Windows: open the "Anaconda Prompt (Radioconda)" shortcut, then:

```
volk_profile
```

On Linux / macOS, `volk_profile` lives next to the Radioconda python, so call it by full path (it's usually not on your PATH unless you've activated the env). Adjust the prefix if you installed Radioconda elsewhere:

```
Linux: ~/radioconda/bin/volk_profile
```

```
macOS: ~/radioconda/bin/volk_profile
```

Takes about 30 seconds. Writes the chosen kernels to %APPDATA%\volk\volk_config on Windows or ~/.volk/volk_config on Linux/macOS. The warning disappears next time you launch the analyzer, and FFT throughput improves on machines with newer SIMD instruction sets.

5. Updating to a new version

Auto-update notifications

The program checks for new versions at launch (no more than once per 24 hours). If a newer version is available, a small dialog opens with the release notes, a button to **Open Download Page** in your browser, a **Skip this version** button (won't nag again about that specific version), and a **Remind me later** button.

You can also trigger a check at any time via **Help → Check for Updates....**

The check is read-only — the program never auto-downloads or auto-installs anything. To disable the auto-check entirely, edit your settings.ini and set auto_check = false under [updates].

Performing the update

Each release is a self-contained zip. To update:

24. **Close** the running app.
25. **Extract** the new zip alongside the old one (or over the top of the old folder). The new folder is named with the new version (e.g. dses-spectrum-analyzer-1.0.1).
26. **Run** launcher.bat or launcher.sh from the **new** folder.

What is preserved across versions

Your settings and window geometry are kept outside the app folder, so updates never wipe them. The on-disk locations are:

OS	Settings file
Windows	%APPDATA%\DSES_Analyzer\settings.ini
macOS	~/Library/Application Support/DSES_Analyzer/settings.ini
Linux	~/.local/share/DSES_Analyzer/settings.ini

The settings file is plain text and editable with any editor while the app is closed. To reset everything to defaults, either delete the file or use **Help → About → Restore Defaults...** in the running app.

Window size and position are saved in the same settings.ini, in the [window] section (plain x / y / width / height integers). You don't need to touch it; setting width/height to 0 (or deleting the lines) makes the window open at its default size next time.

Backing up your settings

The settings file is small (~1 KB). Copy it somewhere before a major upgrade if you want a quick rollback path:

```
Windows: copy "%APPDATA%\DSES_Analyzer\settings.ini" "%USERPROFILE%\Desktop\settings.ini.bak"

macOS: cp "~/Library/Application Support/DSES_Analyzer/settings.ini"
~/Desktop/settings.ini.bak

Linux: cp ~/.local/share/DSES_Analyzer/settings.ini ~/Desktop/settings.ini.bak
```

6. Troubleshooting

“Radioconda not found in standard locations”

The launcher couldn’t auto-detect a Radioconda install. Either:

- Install Radioconda to one of the standard paths the launcher checks (see §3.1 / §3.2 / §3.3), **or**
- Type the path when the launcher prompts you on first run, **or**
- Set the RADIOCONDA_ROOT environment variable to the install root before launching.

When you type the path, give the **install root** — the folder that contains bin/python (macOS/Linux) or python.exe (Windows), e.g. ~/radioconda — **not** its bin/ subfolder. A leading ~ is expanded to your home directory. On macOS/Linux, the launcher also accepts an already-activated conda environment, so if your shell prompt shows (base) for a Radioconda base env, just running the launcher from that shell is enough.

“Missing required packages (PySide6, pyqtgraph, and/or scipy)” — or `ModuleNotFoundError: No module named 'PySide6'`

Radioconda was found, but the app’s GUI/plotting packages aren’t installed in it. Do the one-time install from §2.4 — open an activated Radioconda shell and run:

```
conda install -c conda-forge pyside6 pyqtgraph scipy
```

Then launch again.

“No UHD Devices Found”

The OS doesn’t see the B210. In order:

27. Unplug the B210 from USB. Wait 5 seconds. Plug it back in. UHD’s USB endpoint can hang on some systems after an abrupt kill of the previous run. **If the B210 is running on external power, also disconnect the power during this reset** — with external power applied, pulling only the USB cable doesn’t fully reset the board, and it can stay invisible to uhd_find_devices until both USB and power have been removed and reconnected.
28. Make sure you’re using a USB 3 port (not USB 2 — the B210 will be flaky at high sample rates).
29. From a terminal:

```
Windows: C:\ProgramData\radioconda\Library\bin\uhd_find_devices.exe
```

```
Linux:    ~/radioconda/bin/uhd_find_devices
macOS:    ~/radioconda/bin/uhd_find_devices
```

If `uhd_find_devices` doesn't see it either, the problem is below our software — check the B210's LEDs and try a different USB cable.

30. If `uhd_find_devices` reports a **firmware/image error** rather than “no devices” — e.g. Could not load firmware, `ihex_reader::read(): No EOF record found`, or a missing FPGA image — your Radioconda's UHD images are incomplete or corrupt (seen on some macOS installs). Download them once with UHD's own tool, then re-check:

```
Windows:  C:\ProgramData\radioconda\Library\bin\uhd_images_downloader.exe
Linux:     ~/radioconda/bin/uhd_images_downloader
macOS:     ~/radioconda/bin/uhd_images_downloader
```

It fetches ~100 MB of firmware/FPGA images into Radioconda (needs internet). After it finishes, `uhd_find_devices` should detect the B210.

RTL-SDR not detected (or `[ERROR] ... librtlsdr.2.dylib` / `'librtlsdr.so' at startup`)

Some Radioconda builds (notably on macOS) ship the SoapySDR RTL-SDR *module* (`librtlsdrSupport.so`) but not the library it depends on (`librtlsdr`). The app no longer prints the loader error — it silences backends that can't load, since a missing optional backend just means that radio won't appear — but the RTL-SDR also won't work until you install the backing library. Install it once into Radioconda from an activated Radioconda shell:

```
conda install -c conda-forge librtlsdr
```

This is only needed if you actually use an RTL-SDR dongle. B210, SDRPlay, HackRF, Airspy, BladeRF, LimeSDR, and PlutoSDR users can ignore it. (Windows Radioconda already includes `librtlsdr`, so this applies to macOS/Linux.)

Persistent “O” overflows at high sample rates

Some “O” characters in the Overflow box on the right sidebar at startup are normal as the USB pipe warms up. A continuous stream during steady-state operation usually means the host can't keep up with the sample rate:

- USB 2 ports cannot sustain 20 or 25 MS/s. Move to a USB 3 port.
- Other USB devices on the same controller (especially other 5 Gbps devices, or a heavily used external drive) compete for bandwidth. Move the B210 to its own controller if you have one.
- Reduce sample rate to 10 MHz or 16 MHz.

The Overflow box auto-clears after 15 seconds of no new overflows, so once the rate is stable you should see the box empty.

Spectrum looks flat at -140 dB

The GR flow graph is running but no samples are arriving. Usual causes:

- The B210 is in a powered-but-disconnected state. Unplug and replug.
- Antenna is disconnected (this app uses RX2 by default). Confirm a cable is on RX2.
- Sample rate is set to a value the B210 can't actually achieve. Try 20 MHz.

macOS: “developer cannot be verified”

```
xattr -dr com.apple.quarantine /path/to/dses-spectrum-analyzer-1.0.0
```

Settings won't persist

Settings should be saved automatically on every change and on app close. If they aren't:

- Check the path printed in **Help → About**. The file should exist and be writable.
- If the directory's parent doesn't exist, the app will fail to create it silently. On first run after install, click **Help → About → Open Settings Folder** to verify it opens cleanly.

7. Appendix A — Settings file reference

The settings INI is plain text and editable while the app is closed. Sections:

- [tuning] — preset/coarse/fine/manual frequency in Hz.
- [rx] — sample rate in Hz, RX gain in dB, device_driver (e.g. uhd_b200, sdrplay, rtl-sdr), device_serial.
- [recording] — SigMF recording folder.
- [spectrum] — FFT size, window, averaging, max/min hold, Y-axis range, grid, axis-label toggles, dark/light background, trace styling for each background.
- [waterfall] — intensity range, colormap (per background), grid/axis-label toggles, row count.
- [ui] — control-panel visibility.
- [window] — saved window position/size as x / y / width / height. Set width/height to 0 (or delete the lines) to reset to the default window size.
- [updates] — auto_check (set to false to disable update notifications), plus internal book-keeping fields the program manages on its own.

Missing keys are filled in from built-in defaults on next launch. The file is rewritten on close with a header comment explaining what it is.

To reset every value to its default: use **Help → About → Restore Defaults...**, or just delete the file and launch the app.

8. Appendix B — Device selection and SigMF playback

Choosing among multiple radios

The program enumerates everything supported at launch — Ettus B210s via UHD, plus any SoapySDR-recognised receiver (SDRPlay, RTL-SDR, HackRF, Airspy, Airspy HF+, BladeRF, LimeSDR, PlutoSDR). Behavior:

- **No radio attached, no playback sample:** an error dialog says “No radio found” and explains how to enable playback (see below). The app exits.
- **No radio attached, but `sample.sigmf-data` + `sample.sigmf-meta`` are present next to the program:** the app falls back to **SigMF playback mode** (see below). An informational dialog announces the fallback.
- **One radio attached:** the app opens it silently and remembers it in `settings.ini`.
- **Two or more radios attached** (any mix of B210s and SoapySDR receivers): a picker always appears so you choose which one to use. Your previously-used radio is pre-selected, so pressing Enter reuses the same one. The app remembers your selection for next launch.

The **RX** sidebar group has a button labeled Device: `<product> – <serial>` showing which radio the current session is using. Clicking it re-opens the picker; the new choice is saved and takes effect on the next launch.

The sample-rate combo and gain-slider range adapt automatically to whichever radio is open. SDRPlay caps at 10 MHz; RTL-SDR maxes around 3.2 MHz; HackRF goes to 20 MHz; the B210 to 25 MHz. Saved gain is clamped to the new device’s range if you switch to a narrower one.

To pin a specific device permanently without using the picker, edit `settings.ini` and set `device_driver + device_serial` under `[rx]`. Use `device_serial = auto` to restore the default “first found” behavior.

SigMF playback mode (no-device fallback)

The distribution bundle includes a short SigMF recording so that users without any SDR attached can still launch the program, see live spectrum, and try the controls.

- Filenames: `sample.sigmf-data`` (raw IQ) and `sample.sigmf-meta`` (JSON metadata). Both live next to `dses_spectrum_analyzer.py` in your install folder.
- The file is read in a continuous loop, paced to match the original capture’s sample rate.
- The window title shows **[Playback]**, the Device button shows the file name, and the Recording-status label reads Playback (looping): `sample.sigmf-data`.
- **Sample Rate, RX Gain, and Recording** controls are **disabled** — they have no meaning for a recorded file. Sample rate comes from the file’s metadata.
- **Tuning IS enabled** in playback mode and works as a digital frequency shift on the file’s baseband. Tuning to the file’s actual center frequency (read from the `.sigmf-meta`) shows the recording’s true content. Tuning to other frequencies within $\pm(\text{sample_rate}/2)$ of the file’s center lets you explore the recorded bandwidth — useful for poking around inside the capture. Tune well outside that window and

you'll see noise / wrap-around, since the recording doesn't contain data at those frequencies. This is the expected behavior for true I/Q data, not a bug.

- All visualization controls (Spectrum panel, Waterfall panel, including the dark/light background toggle, FFT averaging, max/min hold, intensity range, colormap, etc.) work normally.
- To replace the bundled sample with your own recording, do a real recording in live mode, then rename the resulting two files to `sample.sigmf-data` and `sample.sigmf-meta` and drop them next to the program. No source-code change is required.
- To skip playback mode and force an error exit when no SDR is attached, delete or rename either of the two `sample.sigmf-*` files.

9. Appendix C — Operating Guide

> This appendix is a copy of the program's built-in **Help → User Guide**. If the two ever differ, the in-app Help is authoritative for the version you're actually running (it ships inside the application).

This is a live spectrum analyzer and waterfall display for the Ettus USRP B210 and other software-defined radios (SDRPlay RSP1A/RSP1B/RSPduo/RSPdx, RTL-SDR, HackRF, Airspy, BladeRF, LimeSDR, PlutoSDR via SoapySDR), designed for pulsar RFI investigation but useful for general-purpose spectrum monitoring.

Starting up — device selection

At launch the program enumerates attached SDRs (UHD + SoapySDR) and decides what to use:

- **One supported SDR attached:** opens it silently and remembers the driver and serial in the settings file.
- **Two or more radios attached:** a picker dialog always appears so you can choose which one to use. Your previous choice is pre-selected, so you can just press Enter to use the same radio as last time. Your selection is remembered for next launch.
- **No SDR attached, but a bundled SigMF sample is present:** the program falls back to **playback mode** — see below.
- **No SDR and no sample:** an error dialog explains how to fix it and the program exits.

The window title always shows which radio is feeding the display, and the **RX** group has a `Device:` button you can click to re-open the picker. The new choice takes effect on the next launch.

Playback mode

If no SDR is attached, the program looks next to the application file for `sample.sigmf-data` + `sample.sigmf-meta` and, if both are found, plays the file back in a continuous loop. The window title shows **[Playback]**. The data source is the file, paced to the original capture's sample rate.

- **Sample Rate, RX Gain, and Recording** controls are disabled — they have no meaning for a recorded file. Sample rate comes from the file's metadata.
- **Tuning is enabled** and works as a digital frequency shift on the file's baseband. Tuning to the file's actual center frequency (read from the `.sigmf-meta`) shows the recording's true content; tuning to other frequencies within $\pm(\text{sample_rate}/2)$ of the file's center lets you "look around" inside the

recorded bandwidth. Tune well outside that window and you'll just see noise / wrap-around — exactly what you'd expect, since the recording doesn't contain data at those frequencies.

- All visualization controls (Spectrum panel, Waterfall panel) work normally.
- To replace the sample with your own, save a SigMF recording, rename the two files to `sample.sigmf-data` and `sample.sigmf-meta`, and drop them next to the program. (You don't need to change any code.)

Sidebar controls (right side)

Mode

- **Live:** real-time FFT of the radio's instantaneous bandwidth around one tuned center frequency (the traditional view).
- **Sweep:** stepped scan across a wide range. Useful for RFI surveys that span more than the radio's instantaneous bandwidth (e.g. surveying 100 MHz–2 GHz on an SDRPlay whose IBW is 10 MHz). See the **Sweep** group below. Sweep mode is unavailable in Playback.

Tuning

- **Pulsar Band:** preset frequencies for common pulsar observation bands. Choose *Manual* to use the Manual Frequency field instead. Presets outside the connected radio's tuning range are greyed out (e.g. the 2304 MHz preset is disabled on an SDRPlay RSP, which tops out at 2 GHz).
- **Coarse Tune:** ± 100 MHz offset from the selected preset (or from the manual frequency).
- **Fine Tune:** ± 10 MHz offset, layered on top of Coarse Tune.
- **Manual Frequency:** used when the *Manual* preset is selected. Accepts engineering notation, e.g. 1.42G or 408M. The value (and the combined center frequency) is clamped to the radio's tuning range — hover the field to see that range.

The Tuning group is disabled in Sweep mode (the center frequency is chosen automatically per step).

Sweep (visible in Sweep mode)

- **Start / Stop:** bottom and top of the swept range. Both are clamped to the connected radio's tuning range (so on an SDRPlay RSP the maximum Stop is 2 GHz; on a B210 it's 6 GHz).
- **Step:** Hz per tuning step. Type auto to use ~80% of the current sample rate (recommended — keeps the clean middle of each FFT and avoids the DC spike at the tuner center plus the band-edge rolloff). For tighter step spacing enter a smaller value, e.g. 2M.
- **Status:** shows the current step number and tune frequency, or *Idle* when Live mode is selected.

Sweep runs continuously, redrawing the wide-spectrum trace after each completed pass. The waterfall plot adds one row per pass, giving a time-vs-frequency view of the whole scanned band. Sample-rate and gain still apply (they affect each step's FFT). Averaging and Max/Min hold are forced off while sweeping (they would smear across retunes) and restored when you switch back to Live.

RX

- **Sample Rate:** per-radio. The combo shows the rates the connected radio actually supports (e.g. B210: 1–25 MHz; SDRPlay: 2–10 MHz; RTL-SDR: 0.25–3.2 MHz). Higher rate = wider spectrum but more disk usage when recording.
- **RX Gain:** per-radio range and meaning. The slider's min/max matches what the driver reports (e.g. B210: 0–76 dB on the AD9361 gain table; SDRPlay: 0–48 dB, internally inverted so higher = stronger signal; RTL-SDR: 0–49.6 dB). AGC, if the driver defaults it on, is disabled at startup so the slider always takes effect.
- **Antenna:** appears only when the open radio has more than one RF input. For a B210 this lists all four physical connectors as A : RX2, A : TX/RX, B : RX2, B : TX/RX — receiver A and receiver B, each with its two SMA ports — and switching includes hopping between the two receivers. An RSPduo lists its two tuners. Pick the connector your cable is actually plugged into; the choice is remembered per radio. Single-port radios (most RTL dongles, the RSP1B) don't show this control.
- **Device:** shows the currently-open radio and re-opens the picker on click.

Recording

- **Folder:** where SigMF capture pairs (.sigmf-meta / .sigmf-data) land. Defaults to ~/Documents/DSES_SA_Recordings.
- **Record:** *Stopped* / *Recording*. Recording always starts *Stopped* on launch.

Spectrum (top plot)

Live FFT magnitude in dB. Use the control panel on the right side to adjust:

- **FFT Size:** 256–8192. Larger = finer frequency resolution but slower response and more averaging-window flicker.
- **Window:** Blackman-Harris is the default — low sidelobes, good for RFI hunting. Hann/Hamming have narrower main lobes; Rectangular has the sharpest peak but the worst sidelobes.
- **Avg α :** exponential averaging. 1.0 = no smoothing (every frame is a fresh measurement). Smaller = more smoothing.
- **Max / Min hold:** overlay traces showing the highest/lowest value ever seen at each bin. Use **Reset** to clear.
- **Y-Axis:** dB min/max, or click **Autoscale** to fit the current data. **Reset Axes** snaps the plot back to the default dB range and full-span frequency view — handy after you've zoomed/panned with the mouse or nudged the min/max and want to get un-lost. It leaves FFT size, window, traces, and colors untouched.
- **Linear scale:** plots linear magnitude instead of dB (the default log scale). In linear mode the Y axis auto-fits and the dB Min/Max boxes are disabled. Affects the spectrum plot only — the waterfall stays in dB.
- **Trace:** color, line width, alpha, label.

Waterfall (bottom plot)

Scrolling 2-D image of FFT vs. time. Newest row at the bottom.

- **Intensity Min/Max:** dB range that maps to the colormap. **Autoscale intensity** picks the 5%–99% percentile of the current data.
- **Colormap:** viridis (default), plasma, inferno, magma, turbo, cividis, gray.
- **Rows:** how many history rows to display (default 256).

Persistence

All selections are saved to a plain-text INI file and restored on next launch. The file location is shown in **Help → About**; you can open the folder directly with the **Open Settings Folder** button.

To revert everything to factory defaults, use **Restore Defaults** in the About dialog.

Update checks

If the developer has configured a manifest URL, the program checks for a newer release in the background at launch (no more than once every 24 hours). When a newer version is found, a non-modal dialog opens with the release notes and a button that opens the download page in your browser — you can ignore it and keep using the app, or click **Skip this version** to not be reminded about that particular version again.

The check is read-only and never auto-downloads or auto-installs. To trigger a check manually, use **Help → Check for Updates....** To disable auto-checks, set `auto_check = false` under [updates] in the settings file. If the manifest URL has not been configured yet, the auto-check is silently skipped.

Tips for pulsar work

- 1422 MHz preset is centered on the neutral-hydrogen line (HI).
- 1666 MHz preset covers the OH maser band.
- Use **Avg $\alpha \approx 0.05$** and **Max hold** to find intermittent RFI sources.
- The waterfall reveals time-structured interference (e.g. radar sweeps, ADS-B bursts) that the live spectrum smears out.