

PhantomSDR-Plus

Полная документация

16 документов · создано 2026-10-03

Содержание

1. README	8
Основные возможности	8
Поддерживаемое оборудование	10
Быстрый старт	11
Результаты тестов производительности	12
Использование	12
Конфигурация	13
Нижний порог водопада — waterfall.sh	14
Настройка внешнего вида	15
Документация	17
Онлайн-каталог WebSDR	17
Устранение неполадок	17
Участие в разработке	18
Лицензия	18
Авторы и благодарности	18
Ссылки	18
Поддержка	18
Советы по производительности	18
2. Руководство по установке	20
Содержание	20
Системные требования	20
Подготовка к установке	21
Установка зависимостей	22
Установка Node.js и npm	23
Установка OpenCL (необязательно, но рекомендуется)	24
Установка PhantomSDR-Plus	26
Установка аудиокодека Opus	31
Autorun Spot Reporter (FT8/FT4/WSPR)	32
Эмуляция клиентов KiwiSDR (необязательно)	34
Настройка	35
Настройка для конкретных устройств SDR	38
Проверка и тестирование	41
Настройка автозапуска	42
Тепловая защита процессора	44
Устранение неполадок	46
Оптимизация производительности	50
Обновление PhantomSDR-Plus	50
Резервное копирование и восстановление	53
Вопросы безопасности	54
Получение помощи	54
Приложение А: полный список зависимостей	55
Приложение Б: примеры конфигурации	55
3. Настройка панели администратора	57
Обзор	57
Что даёт панель администратора	57

Требования	58
Шаг 1 — Файлы	58
Шаг 2 — Запустите сценарий настройки	58
Шаг 3 — Запуск / остановка / перезапуск	59
Шаг 4 — Откройте порт в межсетевом экране	62
Шаг 5 — Функция отключения пользователей	62
Конфигурация — admin_config.json	62
Настройка sdr_host	63
proxu.ru — что это и когда нужно	64
Изменение портов после настройки	64
Полезные команды вручную	65
Файлы журналов	65
Сводка доступа	67
Удаление сообщений чата	67
Трансляция сообщений на водопад	68
Системные графики	69
Thermal Guard	70
Spot Reporting (autorun FT8/FT4/JS8/WSPR)	73
Устранение неполадок	79
4. Thermal Guard	82
Содержание	82
1. Что он делает	82
2. Быстрый старт	83
3. Как он принимает решение	83
4. Четыре режима	85
5. Выбор режима	87
6. Справочник настроек	87
7. Работа без панели администратора	88
8. Включение этапа throttle без root	89
9. Работа без присмотра	90
10. Проверьте его, прежде чем доверять	91
11. Чтение журнала	91
12. Lockout и ограничение частоты остановок	92
13. Устранение неполадок	92
14. Отключение или удаление	93
Смотрите также	93
5. Ограничения подключений	94
Содержание	94
1. От чего вы на самом деле защищаетесь	94
2. Три уровня	95
3. Ограничения по адресу	95
4. Простаивающие подключения	97
5. Защита в ядре	97
6. Справочник настроек	98
7. Что видит слушатель, которому отказали	99
8. Чтение журналов и счётчиков	100
9. Выбор ваших чисел	101
10. Ловушки, о которых стоит знать	101
11. Как всё отключить	102

6. Несколько приёмников	103
Содержание	103
1. Как это работает	103
2. Два способа опубликовать приёмники	104
3. Добавление второго приёмника	104
4. receivers.toml	106
5. Запуск и остановка	107
6. Что видят слушатели	107
7. S-метр выше 30 МГц	108
8. Безопасность	108
9. Сколько приёмников поместится	109
10. Ловушки, о которых стоит знать	109
7. Сервер статистики	110
Обзор	110
Предварительные требования	110
Методы установки	110
Настройте ваше приложение PhantomSDR	113
Тестирование установки	115
Управление службой	116
Ручная установка (Вариант 2)	116
Изменение порта после установки	118
Устранение неполадок	118
Что вы увидите	121
Определение температуры	121
Сводка функций	122
Примечания по безопасности	122
Поддержка и ресурсы	122
8. Декодеры	125
Содержание	125
1. Как запустить декодер	125
2. FT8	128
3. FT4-FT2	129
4. JS8	130
5. CW — азбука Морзе	132
6. QRSS Grabber	133
7. WSPR	136
8. HF FAX / WEFAX	137
9. NAVTEX	138
10. FSK / RTTY — включая PSK31 и Olivia	140
11. SSTV	143
12. Общие советы	145
9. RADE README	147
Установка — коротким путём	147
Содержание	147
Как это работает	148
Обзор архитектуры	149
RADEL против RADEU	149
Предварительные требования	150

Установка вручную	150
Размещение исправленных файлов	150
Управление sidecar	151
Порт 8074	152
Использование RADE в браузере	152
Проверка и отладка	153
Переменные окружения	154
Изменённые файлы	155
Сводка прохождения сигнала	155
Устранение неполадок	156
Измерение параллелизма на своём оборудовании	158
Обновление RADE v1	160
10. RADE общая установка (Linux)	163
Шаг 1 — Системные пакеты	163
Шаг 2 — Пакеты Python	164
Шаг 3 — Клонирование и сборка репозитория rade	165
Шаг 4 — Проверка цепочки декодирования	166
Шаг 5 — Сборка веб-интерфейса PhantomSDR-Plus	167
Шаг 6 — Управление sidecar	167
rade.sh по-прежнему поставляется для автономного применения — запуска sidecar без пускового сценария PhantomSDR-Plus или его отдельной проверки. Его команды, журнал и связанные с ним оговорки о стороже описаны в разделе Управление sidecar.	168
Шаг 7 — Откройте порт 8074	168
Шаг 8 — Запуск сервера	169
Шаг 9 — Запуск RADE	169
Шаг 10 — Использование RADE в браузере	169
Замечание о нагрузке на процессор	170
Обновление RADE в дальнейшем	170
11. Редактирование вариантов	171
1. Что такое варианты	171
2. Что делает каждый файл	172
3. Изменение, затрагивающее только часть вариантов	174
4. Где находятся переключатели вариантов	175
5. Другие полезные регулировки	176
6. После правки — пересборка	176
7. Как проверить правку, прежде чем ей доверять	177
12. Руководство пользователя	178
Содержание	178
Введение	178
Начало работы	179
Обзор интерфейса	180
Основные операции	181
Дополнительные возможности	184
Виды демодуляции	188
Цифровые декодеры	190
Горячие клавиши	194
Закладки	194
Использование на мобильных устройствах	195
Советы и рекомендации	196

Устранение неполадок	197
Часто задаваемые вопросы	199
Ресурсы	200
Приложение: часто используемые частоты	201
Глоссарий	202
13. Эмуляция клиента KiwiSDR	203
Содержание	203
1. Что получает клиент Kiwi	203
2. Установка моста	204
3. Включение	204
4. Справочник настроек	205
5. Подключение клиента	205
6. Уровень звука	206
7. S-метр	206
8. Водопад и спектр	207
9. Если что-то выглядит не так	208
10. Насколько это проверено	209
14. Разнесённый приём	210
Что это делает и чего не делает	210
Быстрый старт	211
Использование WebSDR	212
Как читать панель	213
Remote SNR trim	215
Выбор второго приёмника	215
Устранение неполадок	216
Ограничения	217
15. Управление трансивером	218
Что понадобится	218
Поддерживаемые приёмники	218
Что делает и чего не делает	219
Быстрый старт	219
Выбор способа подключения	220
Настройки последовательного порта	221
Синхронизация	221
Виды модуляции	222
Ширина фильтра	223
Заглушение при передаче	223
Сдвиг частоты	224
Меню Rig	224
Linux	224
Windows	225
TSCI-CAT на странице приёмника	225
Для операторов приёмников	235
Для разработчиков: интерфейс страницы	235
Устранение неполадок	236
Известные ограничения	236
16. Структура проекта	238
Содержание	238

Дерево каталогов	239
Корневой каталог	252
Исходный код (src/)	255
Веб-интерфейс (frontend/)	256
Списки частот (frequencylist/)	261
Файлы конфигурации	261
Система сборки	263
Зависимости файлов	263
Поток данных	264
Руководство по изменению файлов	264
Важные пути	265
Типичные операции с файлами	265
Контроль версий	266

ГЛАВА 1

README

README.md

PhantomSDR-Plus

Расширенный WebSDR-сервер с открытым исходным кодом, продвинутыми функциями и футуристическим дизайном

PhantomSDR-Plus — это форк PhantomSDR, предоставляющий высокопроизводительный веб-сервер программно-определяемого радио (SDR), способный обслуживать сотни одновременных пользователей. Он предлагает улучшенный интерфейс, поддержку нескольких декодеров, отображение band plan и совместимость с различными аппаратными платформами SDR.

Основные возможности

Производительность и масштабируемость

- **Многопользовательский режим:** сотни одновременных пользователей в зависимости от оборудования
- **Высокая частота дискретизации:** поддержка SDR до 70 MSPS (реальный сигнал) / 35 MSPS (IQ)
- **Аппаратное ускорение:** поддержка OpenCL и CUDA для обработки на GPU
- **Оптимизированная передача:** аудиосжатие FLAC и Opus с малой задержкой. Браузер декодирует каждый пакет FLAC в момент его прихода, а его буфер воспроизведения подстраивается под соединение слушателя и держит ровно столько звука, сколько нужно, — обычно 100–150 мс от приёмника до громкоговорителя на телефоне через Wi-Fi

Интерфейс пользователя

- **Футуристический дизайн:** современный адаптивный веб-интерфейс
- **Оптимизация для мобильных устройств:** улучшенный мобильный интерфейс для прослушивания в дороге
- **Отображение band plan:** интерактивный водопад с наложением частотных диапазонов
- **Настраиваемые цветовые схемы:** несколько палитр водопада
- **Двойной S-метр:** на выбор аналоговая или цифровая индикация уровня сигнала
- **Кнопки декодеров в одно нажатие:** ряд Decoders на главной панели, прямо под селектором режимов (FT8, FT4, FT2, JS8, CW, WSPR, FAX, SSTV, NAVTX, RTTY), а RADEL и RADEU — рядом с самим селектором режимов и продублированы во всплывающих окнах режимов и диапазонов. Одно нажатие запускает декодер и открывает его окно, второе — останавливает. Прежний ряд полосы пропускания удалён.
- **Сканер каналов:** проходит по диапазону — или ровно по тому, что показывает водопад, — и останавливается на первом канале, который поднимается на заданное число дБ над собственным шумовым порогом диапазона, а как только канал стихает, идёт дальше сам. Шаг следует режиму, а остановки попадают на сетку каналов (1 кГц SSB, 100 Гц CW, 5 кГц AM на коротких волнах, 9/10 кГц средние волны, 9 кГц длинные волны). Режим *Пропускать пустые* читает спектр водопада и переходит сразу к сигналам, а кнопка блокировки убирает из обзора постоянно занятый канал. Всё работает локально в браузере слушателя, поэтому

сканирование ни для кого другого приёмник не перестраивает (см. [Руководство пользователя](#))

- **Разнесённый приём:** объединяет ваш приёмник со вторым, находящимся в другом месте, и воспроизводит тот из двух, у которого в данный момент сигнал лучше, так что замирание на одной площадке перекрывается другой. Вторым может быть ещё один PhantomSDR-Plus, KiwiSDR, UberSDR или WebSDR; первым трём не нужно ничего, кроме браузера слушателя, и только для WebSDR требуется небольшой релей на вашем собственном сервере. Выравнивание занимает около пятнадцати секунд, а удалённая подстройка SNR уравнивает обе площадки между собой. Каждый слушатель держит в своём браузере именованный список вторых приёмников в том порядке, какой ему удобен, и может выгрузить его в файл и загрузить обратно (см. [Разнесённый приём](#))
- **Несколько приёмников, одна страница:** один компьютер может одновременно обслуживать два и более приёмника — например, RX-888 для KB и RTL-SDR для 2 м, — а слушатели переключаются между ними кнопками в шапке страницы. У каждого приёмника свой водопад, чат, метки, список слушателей и сведения о станции, а выше 30 МГц S-метр показывает баллы S для UKB ($S9 = -93$ дБм) (см. [Несколько приёмников](#))

Обработка сигнала

- **Несколько видов демодуляции:** AM, FM, USB, LSB, CW и другие. Также был реализован декодер RADE версии 1.
- **Синхронное детектирование AM:** улучшенное качество приёма AM
- **Шумоподавление:** NR (спектральное), NB (подавитель импульсных помех), NS (подавление фонового шума) и AN (автоматический нотч) — все на пути прослушивания и никогда на декодерах
- **Варианты АРУ:** несколько режимов автоматической регулировки усиления
- **Автоматический шумоподаватель:** автоматический порог squelch по уровню шума
- **Шумоподаватель по субтону CTCSS:** звук FM проходит только пока принимается выбранный субтон — или любой стандартный; тон выбирается в небольшом окне рядом с кнопкой CTCSS, которая затем его показывает
- **Шумоподавление на основе ИИ:** нейросеть RNNoise в браузере слушателя убирает шум диапазона из речи (только голосовые режимы; см. [Руководство пользователя](#))

Дополнительные возможности

- **Цифровые декодеры:** FT8, FT4, FT2, JS8, CW, QRSS Grabber, WSPR, HF FAX, SSTV, NAVTEX, FSK/RTTY, PSK31, Olivia и FreeDV RADE — каждый работает в отдельном фоновом потоке, поэтому декодирование никогда не прерывает звук (см. [Декодеры](#))
- **Decoder ID:** определяет цифровой вид модуляции в полосе пропускания и одним щелчком предлагает подходящий декодер — по измеренной полосе, расстоянию между тонами, символьной скорости и временной структуре посылок, с учётом частоты как дополнительного признака, что заодно отличает **FT8** от **JS8**, два вида модуляции с одинаковым сигналом. Он только предлагает, никогда не переключает сам, и предпочитает молчать, а не гадать, когда сигнал слишком слаб. Замирания его не заглушают: когда QSB дробит двухминутную посылку WSPR на фрагменты, он всё равно называет вид модуляции, считывая ритм по всей истории прослушивания, и снижает показанную уверенность, отмечая, что свидетельство слабее. Примерно между 400 Гц и 2700 Гц положение в полосе пропускания не имеет значения; за этими пределами появляется янтарная кнопка **Recentre & retry**, которая один раз сдвигает настройку и начинает новое измерение. Когда вы принимаете подсказку, приёмник настраивается на работу этим видом: настроенная частота встаёт в середину водопада, обзор становится около 100 кГц вокруг неё, а боковая полоса и полоса пропускания становятся теми, с которыми этот вид работают — подполоса 3 кГц для семейства FT8, 1350–1650 Гц для

WSPR, ± 250 Гц и CW для телеграфии, и так далее. По умолчанию выключено, в работе — около 0,5 % одного ядра (см. [Декодеры](#))

- **Предустановки цифровых режимов:** FT8, FT4, FT2, JS8 или WSPR задают и боковую полосу, и полосу пропускания
- **Панель разговора JS8:** сообщения из нескольких кадров собираются в целые предложения
- **Панель статистики:** статистика сервера и пользователей в реальном времени
- **Список подключённых пользователей:** все, кто слушает, с кнопкой настройки в каждой строке. Ваш собственный сеанс помечен **you** — и в окне интерфейса, и когда `users.html` открыт как отдельная страница
- **Автоматическая отправка спотов:** служба autorun декодирует FT8, FT4 и WSPR на сервере и выгружает споты в PSK Reporter и WSPRnet, с двумя счётчиками в панели администратора — плитки по декодерам за текущий запуск и итог за всё время рядом с каждым флажком диапазона/вида связи
- **Графики сервера:** страница **Графики** в панели администратора с частотой процессора, нагрузкой, температурой и числом слушателей за период от 15 минут до 24 часов (см. [Панель администратора](#))
- **Тепловая защита:** страж в панели администратора останавливает сервер при перегреве процессора и запускает его снова после остывания; пороги выводятся из критического порога вашего собственного процессора, и нет никаких предположений о том, как вы запускаете и останавливаете сервер. Поставляется в режиме только записи, поэтому ничего не делает, пока вы её не включите (см. [Thermal Guard](#)) Для приёмника без присмотра **настоятельно рекомендуется** запускать панель как юнит systemd: страж живёт внутри неё, иначе перезагрузка уносит защиту с собой (см. [Панель администратора](#)).
- **Эмуляция клиентов KiwiSDR:** необязательный мост, который на том же хосте и порту отвечает и на протокол KiwiSDR, так что программы Kiwi вроде **AetherSDR** и `kiwiclient` подключаются к приёмнику напрямую — с настоящей перестройкой, водопадом и S-метром по той же шкале, что и на веб-странице. Выключен, пока в вашей конфигурации нет `[kiwi_emulation] enabled = true` (см. [Установка](#))
- **Управление трансивером (CAT):** держит ваш трансивер и окно приёмника на одной частоте, виде модуляции и ширине фильтра, в одну сторону или в обе — через меню RIG в [Desktop PhantomSDR+](#) (Icom, Yaesu, Kenwood, Elecraft, FlexRadio и QRP Labs напрямую, любой другой трансивер через Hamlib, или flrig) либо с помощью CATsync Tool for WebSDRs в браузере. Страницы предоставляют функции `catsync_*` для частоты, вида модуляции, ширины фильтра и заглушения. Работает с приёмниками PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR и UberSDR (см. [Управление трансивером](#))
- **Система закладок:** импорт/экспорт частотных закладок
- **Горячие клавиши:** удобная навигация и управление
- **Поддержка колеса мыши:** интуитивная настройка частоты
- **Каталог WebSDR:** интеграция с <https://sdr-list.xyz>

Поддерживаемое оборудование

PhantomSDR-Plus поддерживает широкий спектр приёмников SDR:

Устройство	Частота дискретизации	Формат	Интерфейс
RX888 MK2	до 64 МГц	16 бит	нативный
RTL-SDR	до 3.2 МГц	8 бит	rtl_sdr
HackRF One	до 20 МГц	8 бит	hackrf_transfer
Airspy HF+ / Discovery	до 912 кГц	16 бит	SoapySDR (SoapyAirspyHF) + rx_sdr
SDRplay RSP1A	до 10 МГц	16 бит	SoapySDR (libmirisdr-5 + SoapyMiri или API SDRplay) + rx_sdr
RigExpert Fobos SDR	50 МГц (HF, прямая оцифровка) / до 20 МГц (RF)	16 бит / 32-битный float	SoapySDR (SoapyFobosSDR) + rx_sdr
Другие устройства	переменная	различные	SoapySDR/rx_tools

Быстрый старт

Системные требования

Минимальные:

- Ubuntu 22.04 LTS (рекомендуется) или Fedora
- 2-ядерный процессор
- 4 ГБ ОЗУ
- 10 ГБ дискового пространства

Рекомендуемые:

- Ubuntu 24.04 LTS
- процессор с 4+ ядрами (Ryzen 5 2600 или Intel i5-6500T либо лучше)
- 8 ГБ ОЗУ
- GPU с поддержкой OpenCL (необязательно, но крайне желательно)
- SSD-накопитель

Установка

```
# Клонировать репозиторий
git clone --recursive https://github.com/sv1btl/PhantomSDR-Plus
cd PhantomSDR-Plus

# Сделать скрипты исполняемыми
chmod +x *.sh

# Запустить автоматическую установку
./install.sh
```

Примечание: после выполнения `install.sh` перезапустите терминал, прежде чем продолжать.

PhantomSDR-Plus у вас уже работает? Не устанавливайте его заново — обновите.

Загрузите средство обновления один раз и запустите; ваша конфигурация, метки, пароль администратора, список частот и история чата никогда не затрагиваются, а всё, что вы правили сами, предъявляется вам, а не перезаписывается:

```
bash cd ~/PhantomSDR-Plus curl -fLO https://raw.githubusercontent.com/sv1btl/PhantomSDR-Plus/main/update.sh chmod +x update.sh ./update.sh
```

Последняя строка лишь *сообщает*, что изменилось бы, и ничего не записывает; `./update.sh --apply` выполняет обновление. Подробности — в главе *Обновление PhantomSDR-Plus*. Повторно запускать установщик на работающей станции нужно только тогда, когда сборка падает из-за отсутствующих системных пакетов.

На Fedora используйте `install_fedora.sh`, на Arch — `install_arch.sh`, на openSUSE Tumbleweed — `install_opensuse.sh`: они делают ту же работу другим менеджером пакетов. Все выпуски Debian и Ubuntu покрывает сам `install.sh`.

Ничего не нужно устанавливать вручную заранее. Сценарий ставит зависимости сборки и Node.js, собирает бэкенд, собирает драйвер выбранного вами приёмника (а для RX888 — и правила udev), открывает `site_information.json` для заполнения, собирает фронтенд, ставит OpenCL, если оборудование это позволяет, затем настраивает **панель администратора, декодер FreeDV RADE и сервер статистики** — все три по умолчанию, ответьте `n`, чтобы пропустить — и завершает полным `recompile.sh`. Он идёт в 17 пронумерованных шагов и останавливается на семи вопросах, каждый в рамке «ТРЕБУЕТСЯ ВАШ ОТВЕТ»; `PHANTOM_NONINTERACTIVE=1` отвечает на все значениями по умолчанию. Рассчитывайте от двадцати минут до более часа.

Пошаговое описание см. в [INSTALLATION.md](#) → [Что делает установщик](#). Подробные инструкции по установке см. в [INSTALLATION.md](#).

Результаты тестов производительности

Оборудование	Частота дискретизации	Загрузка ЦП	Число пользователей
Ryzen 5 2600 (все ядра)	64 МГц (32 МГц IQ)	38-40 %	100+ пользователей
AMD RX 580 (GPU)	64 МГц (32 МГц IQ)	28-35 %	100+ пользователей
Intel i5-6500T (с OpenCL)	60 МГц (30 МГц IQ)	10-12 %	100+ пользователей

Примечание: дополнительная нагрузка на ЦП в расчёте на пользователя минимальна (<1 % на пользователя) при включённом аппаратном ускорении.

Использование

Основные операции

1. **Настройте свой SDR:** отредактируйте соответствующий файл конфигурации (например, `config-rtl.toml`)
2. **Обновите сведения о площадке:** отредактируйте `frontend/site_information.json`
3. **Запустите сервер:** выполните подходящий скрипт запуска: `bash ./start-rtl.sh` # Для RTL-SDR `./start-rspla.sh` # Для SDRplay RSP1A `./start-airspyhf.sh` # Для Airspy HF+ `./start-fobos-hf.sh` # Для RigExpert Fobos SDR, HF1/HF2 (0-25 МГц) `./start-fobos.sh` # Для RigExpert Fobos SDR, RF (25-6000 МГц) `./start-hackrf.sh` # Для HackRF One `./start-rx888mk2.sh` # Для RX888 MK2 RSP1A,

Airspy HF+ и Fobos работают через SoapySDR и `rx_sdr`; установщик ставит их драйвер, если выбрать их (пункты 3, 6 и 5), а на уже установленной станции запустите `./setup-rspla.sh`, `./setup-airspyhf.sh` или `./setup-fobos.sh`. См. [INSTALLATION.md](#). HackRF One (пункт 7) нужен только пакет `hackrf` из дистрибутива: `./setup-hackrf.sh`. Каждый скрипт запуска **самодостаточен**: он останавливает запущенный экземпляр, запускает приёмник + `spectrumserver`, **уходит в фоновый режим** (переживает закрытие терминала) и запускает **сторожевой процесс**, который автоматически перезапускает цепочку при её завершении. Ход работы записывается в `logwebsdr.txt` — следите за ним командой `tail -f logwebsdr.txt`. Повторный запуск скрипта равнозначен чистому перезапуску, а блокировка `flock` гарантирует, что одновременно работает только один приёмник. Необязательно: `SPECTRUM_CORES=0-3` для привязки к ядрам ЦП, `RX_ARGS="..."` для замены аргументов приёмника без редактирования файла (`RX888_ARGS` для `RX888`), `RX_DRIVER=miri|sdrplay`, чтобы выбрать драйвер RSP1A.

4. **Откройте интерфейс**: перейдите в браузере на `http://localhost:PORT` (порт по умолчанию зависит от конфигурации)

Остановка сервера

Один общий скрипт остановки работает для **любого** приёмника — он сначала останавливает сторожевой процесс (чтобы тот не перезапустил всё автоматически), затем приёмник и `spectrumserver`:

```
./stop-websdr.sh
```

Конфигурация

Основные файлы конфигурации

1. `config-[устройство].toml` — конфигурация сервера и SDR
2. Настройки сервера (порт, потоки, корневой каталог HTML)
3. Настройки входа (частота дискретизации, размер FFT, частота)
4. Параметры регистрации в каталоге WebSDR
5. Настройки сжатия звука
6. После изменения перезапустите сервер.
7. `frontend/site_information.json` — публичная информация о площадке
8. Сведения об операторе (позывной, e-mail, местоположение)
9. Информация об оборудовании и антенне
10. Настройки региона и полосы
11. Включение/отключение чата
12. После изменения выполните `recompile.sh` в терминале.
13. `markers.json` — частотные маркеры и band plan
14. `frontend/src/bands-config.js` — конфигурация band plan Этот файл расположен в `frontend/src/bands-config.js` и определяет диапазоны, которые создаёт SysOp.
15. Вы увидите примерно следующее и можете редактировать это по своему усмотрению: ``
16. `const bands = .....`

17. { ITU: 1, name: '40m', min: -30, max: 110, initFreq: '7120000', publishBand: '1', startFreq: 7000000, endFreq: 7200000, stepi: 1000, color: 'rgba(50, 168, 72, 0.6)', modes: [{ mode: MODES.CW, startFreq: 7000000, endFreq: 7040000 }, { mode: MODES.LSB, startFreq: 7040000, endFreq: 7200000 }] }, { ITU: 2, name: '40m', min: -30, max: 110, initFreq: '', publishBand: '1', startFreq: 7000000, endFreq: 7300000, stepi: 1000, color: 'rgba(50, 168, 72, 0.6)', modes: [{ mode: MODES.CW, startFreq: 7000000, endFreq: 7050000 }, { mode: MODES.LSB, startFreq: 7050000, endFreq: 7300000 }] }, { ITU: 3, name: '40m', min: -30, max: 110, initFreq: '', publishBand: '1', startFreq: 7000000, endFreq: 7200000, stepi: 1000, color: 'rgba(50, 168, 72, 0.6)', modes: [{ mode: MODES.CW, startFreq: 7000000, endFreq: 7040000 }, { mode: MODES.LSB, startFreq: 7040000, endFreq: 7200000 }] } etc `` **Где:**

18. **ITU** — регион, в котором расположен сервер,
19. **min и max** — пределы яркости водопада для данного диапазона,
20. **initFreq** — начальная частота настройки при выборе диапазона,
21. **publishBand** задаёт, отображается ли диапазон: 1 — любительские диапазоны, 2 — вещательные,
22. **startFreq и endFreq** определяют границы диапазона,
23. **stepi** — шаг колеса мыши по умолчанию для диапазона, а **modes** — предпочтительный режим по умолчанию для диапазона.
24. После внесения изменений **выполните recompile.sh в терминале** из папки PhantomSDR-Plus.

Примеры конфигурации см. во включённых образцах файлов:

- config-rtl.toml — конфигурация RTL-SDR
- config-rspla.toml — конфигурация SDRplay RSP1A
- config-airspyhf.toml — конфигурация Airspy HF+
- config-fobos-hf.toml / config-fobos.toml — конфигурация RigExpert Fobos SDR (HF / RF)
- config-hackrf.toml — HackRF One
- config-rx888mk2.toml — конфигурация RX888 MK2

Нижний порог водопада — `waterfall.sh`

`waterfall.sh` (в корне репозитория) меняет **минимальный уровень водопада по умолчанию** в dB — то значение, с которого водопад начинается при новом заходе на сайт.

Это значение находится в двух исходных файлах в пяти разных местах (`frontend/src/waterfall.js` → `this.minWaterfall`, а также `frontend/src/App.svelte` → начальное `let min_waterfall`, два случая сброса «min» и значение по умолчанию для закладок). Править их вручную легко с ошибкой, поэтому это делает скрипт. Он никогда не использует жёстко заданные номера строк — каждое место находится по шаблону, так что он продолжает работать и после обновления исходников.

- **Более высокое** значение (например, -15) даёт **более тёмный** водопад.
- **Более низкое** значение (например, -45) даёт **более яркий** водопад.

```
./waterfall.sh          # интерактивно — показывает текущие значения и запрашивает новое
./waterfall.sh -v -15   # задать значение без вопросов
./waterfall.sh -v -15 -y # ... и пропустить все подтверждения
./waterfall.sh -s       # только показать текущие значения, ничего не меняя
```

Перед правкой он создаёт резервные копии обоих файлов с меткой времени (`*.bak-ГГГГммдд-ччммсс`) и выводит команду восстановления. Затрагиваются только литералы, равные

используемому сейчас значению, поэтому посторонние числа в этих файлах никогда не будут переписаны по ошибке. Если новое значение после этого не появилось, скрипт сам восстанавливает резервные копии и завершается с ошибкой. Он также предупреждает, когда два файла рассинхронизированы, и приводит оба к новому значению.

- После изменения фронтенд нужно пересобрать — скрипт предлагает запустить `recompile.sh` за вас (с `-y` пересборка пропускается, тогда запустите `./recompile.sh` самостоятельно, когда будете готовы).

Настройка внешнего вида

Изменение фонового изображения

Замените `frontend/src/assets/background.jpg` предпочитаемым изображением (сохранив то же имя файла).

- После изменения выполните `recompile.sh` в терминале.

Выбор аудиокодека

Выберите FLAC или Opus в вашей конфигурации `.toml`:

```
[input]
audio_compression="opus" # or "flac"
```

- После изменения перезапустите сервер.

Калибровка S-метра

Два прибора намеренно работают по **раздельным трактам**, поэтому настраиваются двумя разными регулировками.

Две регулировки

Обе находятся в секции `[input]` конфигурации, с которой запускается сервер (`config.toml`, `config-rx888mk2.toml` и т. д.):

```
[input]
smeter_offset=5          # только цифровая шкала
analog_smeter_offset=5  # только аналоговая стрелка
```

Они читаются в `src/websocket.cpp` и передаются в браузер первым кадром `basic_info`, поэтому **достаточно перезапустить сервер** — пересборка фронтенда не нужна.

Аналоговый тракт

```
показываемые dBm = -130 + (rawDb + analog_smeter_offset + 130) x 1,1
```

- Визуальное усиление 1,1 применяется *после* смещения, с точкой опоры -130 dBm. Таким образом, 1 единица `analog_smeter_offset` сдвигает показание на 1,1 dB. Чтобы сдвинуть показание на X dB: `analog_smeter_offset` $\approx 0,91 \times X$. Значение в TOML читается как целое, поэтому доступны только целые шаги.
- Закон отклонения стрелки (`powerFromDbm` в `frontend/src/lib/SMeterAnalog.svelte`) ставит **S9 на -73 dBm**, при 60 из 100 единиц стрелки, с кривой `pow(...,0,6)` от -130 до -73 ниже S9 и кривой `pow(...,0,8)` от -73 до -13 (S9+60) выше.

- Это **не** линейная шкала 6 dB на балл S, поэтому осмысленная точка калибровки — S9; отметки ниже S9 не будут следовать за генератором шагами по 6 dB.

Цифровой тракт

```
value      = (rawDb / 150) x 100 + smeter_offset  -> ограничение [-100, 0]
segments = round((value + 100) x 35 / 100) + DIGITAL_BAR_TRIM
```

- Обратите внимание на `/150 x 100`: цифровая шкала **сжата до 2/3** и никогда не видит ни `analog_smeter_offset`, ни визуальное усиление 1,1. Поэтому шкала и стрелка никогда не совпадают точно — так задумано.
- 35 сегментов на 100 единиц означают, что один сегмент — это 2,86 единицы смещения, примерно 4,3 dB исходных. То есть `smeter_offset=3` — примерно один сегмент.
- `DIGITAL_BAR_TRIM` в `frontend/src/lib/SMeterDigital.svelte` (сейчас 0) сдвигает шкалу на целые сегменты. Его изменение — правка исходного кода и **требует** пересборки фронтенда.

Окна LED в обоих приборах питаются от аналогового тракта

Окна dBm / dBμV / SNR / NF в **обоих** приборах читают аналогово откалиброванное значение и прибавляют фиксированный `VISUAL_DBM_OFFSET`, равный 5, жёстко прописанный в `SMeterAnalog.svelte` и `SMeterDigital.svelte`. Следовательно:

- `smeter_offset` **не** меняет ни одного числа в окнах, только длину шкалы.
- $\text{dB}\mu\text{V} = \text{dBm} + 107$ (50 Ом), а $\text{NF} = \text{dBm} - \text{SNR}$ по построению, поэтому NF следует автоматически.
- Если нужны правильные **числа** без изменения положения стрелки, регулировка — `VISUAL_DBM_OFFSET`, но это константа в исходном коде, требует пересборки и должна быть изменена в **обоих** файлах, чтобы они остались согласованы.

Порядок действий

1. Подайте известный уровень на антенный вход (генератор -73 dBm = S9), в SSB/CW со стабильной полосой пропускания. Показания зависят от полосы и АРУ, поэтому сначала зафиксируйте режим и полосу.
2. Посмотрите, что показывает аналоговое окно dBm, и задайте `analog_smeter_offset` $\approx 0,91 \times (-73 - \text{показание})$. Перезапустите сервер и проверьте снова. Одной итерации достаточно; стрелка должна встать на S9.
3. Перейдите к цифровому прибору и подберите `smeter_offset` так, чтобы длина шкалы ставила S9 туда, куда нужно — примерно 3 единицы на сегмент. Этот шаг чисто косметический и на числовые показания не влияет.
4. Без генератора хорошей заменой служит тихий диапазон с маяком известного уровня, либо просто «привязка» уровня шума к правдоподобному значению (например, -120 dBm на 20 м с приличной антенной) — с пониманием, что так калибруется весь тракт вместе с усилением антенны, а не только приёмник.

Варианты графического интерфейса

Все четыре варианта интерфейса (аналоговый/цифровой S-метр x компоновка v1/v2) находятся в одной сборке. Посетители переключают их меню ☿ в правом верхнем углу — ничего не перезагружается, звук, водопад и работающие декодеры продолжают работу — и выбор запоминается в каждом браузере.

- Выполните `recompile.sh` в терминале, чтобы задать *стартовый* вариант — тот, который видит посетитель при первом заходе.

- **EDITING_VARIANTS.md** — как редактировать варианты фронтенда (S-метр и компоновка) и пересобирать их

Документация

- **INSTALLATION.md** — полное руководство по установке для системных операторов
- **ADMIN_PANEL_SETUP.md** — руководство по установке панели администратора
- **USER_GUIDE.md** — руководство конечного пользователя по работе с WebSDR
- **THERMAL_GUARD.md** - Руководство сисопа по защите от перегрева процессора: четыре режима и что делать в каждом
- **CONNECTION_LIMITS.md** - Руководство сисопа по ограничениям подключений: защита публичного приёмника от лавины подключений
- **MULTI_RECEIVER.md** - Руководство сисопа по нескольким приёмникам на одном компьютере: экземпляры, `receivers.toml`, выбор приёмника и `start-all.sh`
- **PROJECT_STRUCTURE.md** — структура каталогов и организация кода
- **EDITING_VARIANTS.md** — как редактировать варианты фронтенда (S-метр и компоновка) и пересобирать их

Онлайн-каталог WebSDR

Зарегистрируйте свой WebSDR в официальном каталоге: <https://sdr-list.xyz>

Укажите `register_online=true` в файле конфигурации `.toml` для автоматической регистрации.

Устранение неполадок

Частые проблемы

OpenCL не работает

- Убедитесь, что драйверы установлены правильно
- Проверьте командой `clinfo`
- Подробную настройку OpenCL см. в `INSTALLATION.md`

Проблемы с задержкой звука

- Оставляйте FLAC в качестве кодека: у него меньше задержка, а Opus удерживает до 20 мс на пакет
- Откройте страницу, добавив к адресу `?diag=1` (например, `/mobile?diag=1`), чтобы увидеть задержку по частям — пакет, буфер и вывод; буфер сам сокращается за минуту-две после сбоя сети
- Если звук прерывается, поднимите буфер на одну ступень (см. Руководство пользователя, *Звуковой буфер и задержка*)
- Обеспечьте достаточные ресурсы ЦП/GPU

Ошибки сборки

- Убедитесь, что установлены все зависимости
- Попробуйте очистить каталог сборки: `rm -rf build && meson setup build`

- Проверьте наличие конфликтующих версий библиотек

Участие в разработке

Вклад приветствуется! Это независимый форк с дополнительными функциями. Пожалуйста:

1. Сделайте форк репозитория
2. Создайте ветку для новой функции
3. Зафиксируйте изменения
4. Отправьте ветку в репозиторий
5. Создайте Pull Request

Лицензия

Проект распространяется по лицензии GNU General Public License v3.0 — подробности см. в файле [LICENSE](#).

Авторы и благодарности

- **SV1BTL и SV2AMK** — разработка и доработки PhantomSDR-Plus
- Основано на исходном проекте PhantomSDR

Ссылки

- **Живая демонстрация:** <http://phantomsdr.no-ip.org:8900/>
- **Каталог WebSDR:** <https://sdr-list.xyz>
- **Репозиторий GitHub:** <https://github.com/sv1btl/PhantomSDR-Plus>

Поддержка

- **Проблемы:** сообщайте об ошибках через [GitHub Issues](#)
- **E-mail:** контакт sv1btl@otenet.gr

Советы по производительности

1. **Включите OpenCL/CUDA** для ускорения на GPU — резко снижает нагрузку на ЦП
2. **Используйте SSD** для лучшей производительности ввода-вывода
3. **Выделите достаточно ОЗУ** — рекомендуется не менее 8 ГБ
4. **Оптимизируйте размер FFT** — больший FFT = лучшее разрешение, но выше нагрузка на ЦП
5. **Рассмотрите отдельную видеокарту** — AMD или NVIDIA с поддержкой OpenCL

73 de SV1BTL & SV2AMK

Подробные инструкции по настройке см. в `INSTALLATION.md` Руководство по эксплуатации для конечного пользователя см. в `USER_GUIDE.md`

ГЛАВА 2

Руководство по установке

INSTALLATION.md

Руководство по установке PhantomSDR-Plus для системных операторов

Это подробное руководство проведёт вас через установку и настройку PhantomSDR-Plus на вашем сервере.

Содержание

1. Системные требования
2. Подготовка к установке
3. Установка PhantomSDR-Plus — сценарий и что он делает
4. Autorun Spot Reporter (FT8/FT4/WSPR)
5. Эмуляция клиентов KiwiSDR (необязательно)
6. Настройка
7. Настройка для конкретных устройств SDR
8. Проверка и тестирование
9. Настройка автозапуска
10. Тепловая защита процессора
11. Устранение неполадок

Только для справки — установщик уже делает всё это за вас. Читайте эти разделы, если у вас дистрибутив, который не покрывает ни один из сценариев, или если нужно починить отдельный шаг вручную:

- Установка зависимостей
 - Установка Node.js и npm
 - Установка OpenCL
 - Сборка вручную
 - Установка аудиокодека Opus
-

Системные требования

Поддерживаемые операционные системы

Основная (рекомендуется):

- Ubuntu 22.04 LTS (Jammy), 24.04 LTS (Noble) и 26.04 LTS (Resolute) — рекомендуется 24.04
- Debian 12 (Bookworm) и Debian 13 (Trixie)

Альтернатива:

- Fedora (последний стабильный выпуск)

- Arch Linux (rolling)
- openSUSE Tumbleweed (не Leap — см. примечание ниже)

Требования к оборудованию

Минимальная конфигурация:

- Процессор: двухъядерный (от 2 ГГц)
- ОЗУ: 4 ГБ
- Накопитель: 10 ГБ свободного места
- Сеть: соединение 100 Мбит/с

Рекомендуемая конфигурация:

- Процессор: четырёхъядерный или лучше (Ryzen 5 2600, Intel i5-6500T или лучше)
- ОЗУ: 8 ГБ и более
- Накопитель: SSD от 20 ГБ
- GPU: AMD/NVIDIA с поддержкой OpenCL (крайне желательно)
- Сеть: соединение 1 Гбит/с

Высокопроизводительная конфигурация:

- Процессор: 6 и более ядер (Ryzen 7, Intel i7 или лучше)
- ОЗУ: 16 ГБ и более
- Накопитель: NVMe SSD
- GPU: отдельная видеокарта с поддержкой OpenCL/CUDA
- Сеть: 1 Гбит/с и выше

Подготовка к установке

1. Обновите систему

```
# Ubuntu/Debian
sudo apt update
sudo apt upgrade -y
sudo reboot

# Fedora
sudo dnf update -y
sudo reboot
```

2. Проверьте версию Ubuntu

```
lsb_release -a
```

Ожидаемый вывод должен показать: Ubuntu 24.04 LTS

3. Проверьте свободное место на диске

```
df -h
```

Убедитесь, что в вашем домашнем каталоге есть не менее 10 ГБ свободного места.

4. Проверьте сведения о процессоре

```
lscpu
```

Запомните число ядер/потоков для оптимизации конфигурации.

Установка зависимостей

ВАЖНО

Для обычной установки этот раздел не нужен. `./install.sh` — или установщик вашего дистрибутива — делает всё это за вас; см. [Что делает установщик](#). Ниже — справочный материал для ручной установки, для дистрибутива, который не покрывает ни один сценарий, или для починки отдельной части вручную.

Ubuntu 24.04 LTS

```
sudo apt install -y \
  build-essential \
  cmake \
  pkg-config \
  meson \
  libfftw3-dev \
  libwebsocketpp-dev \
  libflac++-dev \
  zlib1g-dev \
  libzstd-dev \
  libboost-all-dev \
  libopus-dev \
  libliquid-dev \
  git \
  psmisc \
  wget \
  curl
```

Fedora

```
sudo dnf install -y \
  g++ \
  meson \
  cmake \
  fftw3-devel \
  websocketpp-devel \
  flac-devel \
  zlib-devel \
  boost-devel \
  libzstd-devel \
  opus-devel \
  liquid-dsp-devel \
  git \
  psmisc \
  wget \
  curl
```

Проверьте установку

```
# Check if key dependencies are installed
pkg-config --modversion fftw3
cmake --version
meson --version
```

Установка Node.js и npm

ВАЖНО

Для обычной установки этот раздел не нужен. `./install.sh` — или установщик вашего дистрибутива — делает всё это за вас; см. [Что делает установщик](#). Ниже — справочный материал для ручной установки, для дистрибутива, который не покрывает ни один сценарий, или для починки отдельной части вручную.

PhantomSDR-Plus требует Node.js для сборки веб-интерфейса. Для установки воспользуемся NVM (Node Version Manager).

1. Установите NVM

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.4/install.sh | bash
```

2. Загрузите NVM

ВАЖНО: закройте и снова откройте терминал либо выполните:

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
```

3. Проверьте установку NVM

```
nvm --version
```

Ожидаемый вывод: `0.40.4` или похожий

4. Установите Node.js

```
# Install Node.js 22 - the version every install script requires
nvm install 22

# Make it the default, so systemd units and cron see it too
nvm alias default 22

# Verify installation
node --version
npm --version
```

5. Необязательно: установите другие версии Node

```
# Install latest version
nvm install node

# Install specific version (if needed)
nvm install 22.22.3

# List installed versions
nvm list

# Use specific version
nvm use 22
```

Установка OpenCL (необязательно, но рекомендуется)

ВАЖНО

Для обычной установки этот раздел не нужен. `./install.sh` — или установщик вашего дистрибутива — делает всё это за вас; см. [Что делает установщик](#). Ниже — справочный материал для ручной установки, для дистрибутива, который не покрывает ни один сценарий, или для починки отдельной части вручную.

OpenCL существенно повышает производительность, перенося вычисления FFT на GPU. Этот раздел посвящён встроенной графике Intel. Для GPU AMD/NVIDIA обратитесь к документации производителя.

Процессор Intel со встроенной графикой

1. Установите базовые компоненты OpenCL

```
sudo apt install -y \
  build-essential \
  cmake \
  pkg-config \
  meson \
  libfftw3-dev \
  libwebsocketpp-dev \
  libflac++-dev \
  zlib1g-dev \
  libzstd-dev \
  libboost-all-dev \
  libopus-dev \
  libliquid-dev \
  libclfft-dev \
  ocl-icd-opencl-dev \
  clinfo
```

2. Скачайте Intel Compute Runtime

```
mkdir -p ~/neo
cd ~/neo

wget https://github.com/intel/compute-runtime/releases/download/19.07.12410/intel-
gmmlib_18.4.1_amd64.deb
wget https://github.com/intel/compute-runtime/releases/download/19.07.12410/intel-igc-
core_18.50.1270_amd64.deb
wget https://github.com/intel/compute-runtime/releases/download/19.07.12410/intel-igc-
opencl_18.50.1270_amd64.deb
wget https://github.com/intel/compute-runtime/releases/download/19.07.12410/intel-
opencl_19.07.12410_amd64.deb
wget https://github.com/intel/compute-runtime/releases/download/19.07.12410/intel-
ocloc_19.07.12410_amd64.deb
```

3. Установите среду выполнения Intel OpenCL

```
sudo dpkg -i *.deb
```

Если возникают ошибки зависимостей:

```
sudo apt --fix-broken install
```

4. Установите загрузчик OpenCL ICD

```
sudo apt update
sudo apt install intel-opencl-icd
```

Если возникают ошибки:

```
sudo apt --fix-broken install
sudo apt install intel-opencl-icd
```

5. Проверьте установку OpenCL

```
sudo clinfo
```

Вы должны увидеть сведения о вашей платформе и устройствах OpenCL. Обратите внимание на:

- Число платформ: 1 (или больше)
- Название платформы: Intel(R) OpenCL (или похожее)
- Тип устройства: GPU или CPU

6. Перезагрузитесь

```
sudo reboot
```

OpenCL на GPU AMD

Для видеокарт AMD установите ROCm:

```
# Add ROCm repository
wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | sudo apt-key add -
echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/debian/ ubuntu main' | sudo tee /etc/apt/sources.list.d/rocm.list

# Install ROCm
sudo apt update
sudo apt install rocm-opencl rocm-clinfo

# Add user to video group
sudo usermod -a -G video $USER

# Reboot
sudo reboot

# Verify
clinfo
```

OpenCL на GPU NVIDIA

Для видеокарт NVIDIA установите CUDA:

```
# Install NVIDIA drivers
sudo apt install nvidia-driver-525

# Install CUDA toolkit
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-keyring_1.0-1_all.deb
sudo dpkg -i cuda-keyring_1.0-1_all.deb
sudo apt update
sudo apt install cuda

# Reboot
sudo reboot

# Verify
nvidia-smi
clinfo
```

Установка PhantomSDR-Plus

Клонирование репозитория, права на сценарии, запуск установщика

```
cd ~
git clone --recursive https://github.com/sv1btl/PhantomSDR-Plus
cd PhantomSDR-Plus
chmod +x *.sh
./install.sh
```

⚠ **ВАЖНО:** после завершения установки **перезапустите терминал** — только тогда свежее установленные Node.js и Rust окажутся в вашем PATH.

PhantomSDR-Plus у вас уже работает? Не устанавливайте его заново — обновите.

Загрузите средство обновления один раз и запустите; ваша конфигурация, метки, пароль администратора, список частот и история чата никогда не затрагиваются, а всё, что вы правили сами, предъявляется вам, а не перезаписывается:

```
bash cd ~/PhantomSDR-Plus curl -fLO https://raw.githubusercontent.com/sv1btl/PhantomSDR-Plus/main/update.sh chmod +x update.sh ./update.sh
```

Последняя строка лишь *сообщает*, что изменилось бы, и ничего не записывает; `./update.sh --apply` выполняет обновление. Подробности — в главе *Обновление PhantomSDR-Plus*. Повторно запускать установщик на работающей станции нужно только тогда, когда сборка падает из-за отсутствующих системных пакетов.

Возьмите сценарий, соответствующий вашей системе. Все они делают одну и ту же работу и задают одни и те же вопросы; отличается только менеджер пакетов:

Система	Сценарий
Ubuntu 22.04 / 24.04 / 26.04, Debian 12 / 13	<code>./install.sh</code>
Fedora	<code>./install_fedora.sh</code>
Arch	<code>./install_arch.sh</code>
openSUSE Tumbleweed	<code>./install_opensuse.sh</code>

Один установщик для всех выпусков Debian и Ubuntu. `install.sh` читает `/etc/os-release` и установленную версию Boost и подстраивается сам; прежние `install_ubuntu22.sh`, `install-Deb12.sh` и `install_ubuntu26.sh` удалены. Он задаёт `DEBIAN_FRONTEND=noninteractive`, чтобы `tzdata`, приходящий вместе с `python3-matplotlib`, не мог остановить установку вопросом о вашем часовом поясе и затем проглотить ответ, предназначенный следующему вопросу. На Jammy он прямо вызывает `install_rade_ubuntu22.sh`, поскольку `python3-websockets` в Jammy — 10.1, а RADE требует 11.0 или новее. Он знает, где в каждом выпуске находится `intel-onekl-icd`: в репозиториях на 22.04 и 26.04, в `non-free` на Debian 12 и в собственном графическом репозитории Intel на 24.04 и Debian 13. А там, где Boost 1.87 или новее, патч заголовков `websocketpp` перестает быть необязательным: установщик проверяет, что он применён, и без него отказывается собирать. Более новый компилятор не нужен никогда: штатный GCC принимает `-std=c++23` на всех пяти выпусках, так что не устанавливайте ради этого `gcc-12`.

Установка идёт в 17 чётко пронумерованных шагов, и каждое место, где она вас ждёт, обрамлено рамкой  **ТРЕБУЕТСЯ ВАШ ОТВЕТ**, чтобы вопрос нельзя было спутать с пробегающим выводом. Семь вопросов перечислены в самом начале, до того как что-либо будет установлено.

`PHANTOM_NONINTERACTIVE=1` отвечает на все значениями по умолчанию; переменные `PHANTOM_*` описаны в шапке `install.sh`.

Каждый запуск пишет `install.txt`. Когда установщик завершается — или обрывается на полпути — он записывает отчёт в `install.txt` в каталоге PhantomSDR-Plus: результат, каждый из 17 шагов как OK / SKIPPED / PARTIAL / FAILED, что было обнаружено (дистрибутив, Boost, компилятор, Node.js), какие компоненты установлены и каждое возникшее предупреждение. Неудачный запуск оставляет отчёт, который заканчивается на упавшем шаге, с причиной и замечанием, что установщик можно безопасно запустить снова. Это первый файл, который стоит прочитать, когда что-то не сработало, и первый, который прикладывают к сообщению об ошибке. Каждый запуск перезаписывает его, поэтому сохраните копию, если хотите сравнить две установки.

Ubuntu 26.04, Arch и openSUSE Tumbleweed собираются, но в эфире не работали. Все три несут Boost новее 1.87, где удалён API `io_service`, под который написан встроенный `websocketpp` 0.8.2. Исправленные заголовки, которые копирует установщик, закрывают этот разрыв (`io_context`, `executor_work_guard`, `boost::asio::post`, современный `resolver`), а там, где Boost 1.87 или новее, установщик считает этот патч обязательным, а не необязательным: он проверяет, что копии на месте, и без них собирать отказывается. Полная установка — с драйвером приёмника и всеми необязательными компонентами — проверена от начала до конца в контейнерах на Boost 1.90 (Ubuntu 26.04), 1.91 (openSUSE Tumbleweed) и 1.92 (Arch). Это доказывает, что сервер собирается и запускается, а не что он часами обслуживает приёмник. Считайте все три непроверенными в продакшене, пока кто-нибудь не сообщит.

openSUSE здесь означает Tumbleweed. Именно там проверен `install_opensuse.sh`. Leap 15.6 не работает: в его репозиториях вообще нет `liquid-dsp-devel`, который нужен бэкенду, а Boost доступен только под версионированными именами пакетов. Поддержка Leap потребовала бы подключения сторонних репозиториев OBS, поэтому пока она вне рамок.

Что делает установщик

Ничего не нужно готовить вручную заранее — не надо вставлять список пакетов, качать Node.js или искать пакеты OpenCL. Он идёт в 19 пронумерованных шагов и останавливается, чтобы задать до десяти вопросов, каждый в рамке «ТРЕБУЕТСЯ ВАШ ОТВЕТ», — так что либо оставайтесь за клавиатурой, либо задайте `PHANTOM_NONINTERACTIVE=1` и позвольте ему ответить на всё значениями по умолчанию (см. ниже), и рассчитывайте от примерно двадцати минут до заметно более часа — в зависимости от машины и от того, сколько дополнений вы оставите.

#	Шаг	О чём спрашивает
1	Перечисляет службы PhantomSDR-Plus, работающие прямо сейчас, — панель администратора, обратный прокси, сервер статистики, приёмник — и предлагает остановить их, прежде чем что-либо трогать. Приёмником считается только запущенный сценарий запуска; сценарий, просто открытый в редакторе, не трогается.	подтверждение, по умолчанию да
2–6	Определяет дистрибутив и ставит все зависимости сборки (компилятор, meson/ninja, FFTW, Boost, FLAC, Opus, liquid-dsp, zlib/zstd, libcurl ...), в том числе Node.js 22 через nvm, если в системе его нет или он слишком старый	ни о чём
7	Собирает бэкенд с помощью meson	ни о чём
8	Собирает драйвер вашего приёмника — RX888 MkII / RX888, RTL-SDR (Blog V4 спрашивается отдельно), SDRplay RSP1A, RigExpert Fobos SDR, Airspy HF+, HackRF One или ничего. При выборе RX888 ставятся и правила udev , чтобы серверу никогда не требовался <code>sudo</code> для устройства. RSP1A, Fobos и Airspy HF+ работают через SoapySDR: их выбор запускает <code>setup-rsp1a.sh</code> , <code>setup-fobos.sh</code> или <code>setup-airspyhf.sh</code> , которые собирают драйвер и <code>rx_sdr</code> и тоже ставят правило <code>udev</code> (см. Приёмники через SoapySDR (RSP1A, Fobos, Airspy HF+)) HackRF One не нужен SoapySDR: его выбор запускает <code>setup-hackrf.sh</code> , который ставит пакет <code>hackrf</code> из дистрибутива и правило <code>udev</code> .	какой у вас SDR
9	Открывает <code>frontend/site_information.json</code> в вашем редакторе	позывной, локатор, оборудование, антенна — не пропускайте
10–11	Ставит зависимости фронтенда и собирает страницы для компьютера и / <code>mobile</code>	ни о чём
12	Ставит OpenCL, выбирая поставщика по найденному оборудованию (GPU Intel / AMD / NVIDIA либо x86-runtime для ЦП). Если подходящего устройства нет, прямо сообщает об этом и идёт дальше	подтверждение, по умолчанию да
13–15	Ставит панель администратора, декодер FreeDV RADE V1 и сервер статистики — все три по умолчанию	подтверждение каждого, по умолчанию да; у каждого свои вопросы
16	Заново накладывает пять исправленных заголовков <code>websocketpp</code> поверх подпроекта <code>meson</code> и проверяет, что они на месте. Три из них — работа по совместимости с <code>Boost ≥ 1.87</code> , без которой <code>backend</code> не собирается на <code>Boost 1.90</code> ; остальные два — собственные изменения проекта, одно из которых нужно для регистрации на <code>websdr.org</code>	ни о чём
17	Ставит эмуляцию клиентов KiwiSDR , запуская <code>kiwi_install.sh</code> , чтобы клиенты Kiwi вроде AetherSDR могли подключаться к этому приёмнику. Правит исходники и добавляет <code>[kiwi_emulation]</code> в файлы конфигурации в корне репозитория — см. Эмуляция клиентов KiwiSDR	подтверждение, по умолчанию да
18	Запускает <code>recompile.sh</code> , чтобы всё собралось из пропатченных исходников	<code>[3] Both backend and frontend → стартовый вариант → [1] build-all.sh</code>
19	Печатает сводку: каждый шаг с его итогом и каждый компонент с тем, что было установлено	ни о чём

Установка без присмотра

У каждого вопроса есть переменная окружения, которая его переопределяет, и установщик сам переходит к значениям по умолчанию, когда stdin не терминал (конвейер, контейнер, задача CI). Это одинаково работает во всех четырёх: `install.sh`, `install_arch.sh`, `install_fedora.sh` и `install_opensuse.sh`. Задайте `PHANTOM_NONINTERACTIVE=1` — и весь прогон завершится, ничего не спросив:

Переменная	Действие
<code>PHANTOM_NONINTERACTIVE=1</code>	answer every question with its default
<code>PHANTOM_SDR=1...7</code>	RX888 · RTL-SDR · SDRplay RSP1A · skip · Fobos · Aircspy HF+ · HackRF (default 4)
<code>PHANTOM_RTLSDR_V4=y\ n</code>	RTL-SDR Blog V4 driver (default n)
<code>PHANTOM_SITE_EDIT=y\ n</code>	open <code>site_information.json</code> in an editor
<code>PHANTOM_OPENCL=y\ n</code>	install OpenCL (default y)
<code>PHANTOM_OPENCL_PROVIDER=1\ 2\ 3</code>	Intel · Mesa/Rusticl · POCL (default: from the detected hardware)
<code>PHANTOM_ADMIN=y\ n</code>	admin panel (y interactive, n unattended)
<code>PHANTOM_RADE=y\ n</code>	RADE / FreeDV (y interactive, n unattended)
<code>PHANTOM_STATS=y\ n</code>	statistics server (y interactive, n unattended)
<code>PHANTOM_KIWI=y\ n</code>	KiwiSDR client emulation (default y)
<code>PHANTOM_RECOMPILE=y\ n</code>	final rebuild (default y)
<code>PHANTOM_CURLPP=y\ n</code>	продолжить без curlpp — только Arch и openSUSE (по умолчанию y)
<code>PHANTOM_FIX_CLOCK_SKEW=y\ n</code>	сбросить временные метки исходников, датированные будущим, чтобы meson смог собрать (по умолчанию y)

Три под-установщика, помеченные *n* без присмотра, сами интерактивны, поэтому прогон без присмотра пропускает их, а не зависает на их вопросах. Назовите их явно, чтобы включить:

```
# a complete unattended install for an RX888 MkII
PHANTOM_NONINTERACTIVE=1 PHANTOM_SDR=1 ./install.sh

# ... and with the admin panel, RADE and the statistics server too
PHANTOM_NONINTERACTIVE=1 PHANTOM_SDR=1 \
  PHANTOM_ADMIN=y PHANTOM_RADE=y PHANTOM_STATS=y ./install.sh
```

После перезагрузки правила udev для RX888 и драйвер OpenCL действуют полностью; установщик сам сообщает, когда нужна перезагрузка или повторный вход в систему.

При настройке панели администратора установщик предлагает также два юнита systemd и **настоятельно рекомендует** согласиться: защита от перегрева процессора работает внутри панели, поэтому без них перезагрузка или сбой оставляют машину без защиты. Шаг автоматически пропускается там, где systemd не работает (контейнеры, WSL1, OpenRC) — см. [руководство по панели](#).

Каждое дополнение — обычный сценарий, который можно запустить и отдельно в любой момент: `./setup_admin.sh` ([руководство](#)), `./install_rade.sh` ([руководство](#)), `./install-stats-server.sh` ([руководство](#)), `./setup-rx888-udev.sh` и сценарии приёмников `./setup-rsp1a.sh`, `./setup-fobos.sh`, `./setup-airspyhf.sh` и `./setup-hackrf.sh` — именно их и вызывает установщик.

Сборка вручную (справочно)

ВАЖНО

Для обычной установки этот раздел не нужен. `./install.sh` — или установщик вашего дистрибутива — делает всё это за вас; см. [Что делает установщик](#). Ниже — справочный материал для ручной установки, для дистрибутива, который не покрывает ни один сценарий, или для починки отдельной части вручную.

Если когда-нибудь придётся собирать без установщика — дистрибутив, который не покрыт сценариями, или починка незавершённой сборки:

Сборка backend

```
# Clean any previous builds, Configure build with optimization, Compile
rm -rf build
meson setup build --buildtype=release --optimization=3
meson compile -C build

# Verify binary was created
ls -lh build/spectrumserver
```

Сборка frontend

```
cd frontend

# Install dependencies, Fix any vulnerabilities, Build the frontend
npm install
npm audit fix
npm run build

# Return to project root
cd ..
```

Проверьте установку

```
# Check if binary exists
ls -lh build/spectrumserver

# Check if frontend was built
ls -lh frontend/dist/

# List important files
ls -lh *.sh *.toml
```

Установка аудиокодека Opus

ВАЖНО

Для обычной установки этот раздел не нужен. `./install.sh` — или установщик вашего дистрибутива — делает всё это за вас; см. [Что делает установщик](#). Ниже — справочный материал для ручной установки, для дистрибутива, который не покрывает ни один сценарий, или для починки отдельной части вручную.

Opus обеспечивает лучшее качество звука и меньшую задержку по сравнению с FLAC.

1. Установите системную библиотеку libopus

```
sudo apt-get update
sudo apt-get install -y libopus0 libopus-dev
```

2. Установите декодер Opus для веб-интерфейса

```
cd PhantomSDR-Plus/frontend

# Install npm dependencies if not already done, Install Opus WASM decoder, Fix any vulnerabilities,
# Rebuild frontend
npm install
npm install @wasm-audio-decoders/opus-ml
npm audit fix
npm run build

# Return to project root
cd ..
```

3. Проверьте установку Opus

```
# Check if Opus system library is installed
pkg-config --modversion opus

# Check if Opus npm package is installed
cd frontend
npm list @wasm-audio-decoders/opus-ml
cd ..
```

Autorun Spot Reporter (FT8/FT4/WSPR)

В состав PhantomSDR-Plus входит необязательный **autorun spot reporter** (в каталоге `autorun/`). Он декодирует FT8/FT4/WSPR на стороне сервера прямо с приёмника и выгружает споты в сети отчётности:

- **FT8 / FT4 → PSK Reporter** (IPFIX/UDP)
- **WSPR → wsprnet.org** (HTTP)

Управление ведётся целиком из вкладки **панель администратора → «Spot Reporting»**, и **отчётность по умолчанию ВЫКЛЮЧЕНА** — ничего не передаётся и не выгружается, пока вы её там не включите. Декодер работает как отдельная служба Node.js и получает звук приёмника локально, поэтому дополнительного РЧ-оборудования не требуется.

Требования

Требование	Примечания
Node.js 22+	Та же среда выполнения, что уже нужна для сборки веб-интерфейса — устанавливается на шаге Node.js .
Пакеты npm ws + cbor-x	Разрешаются через <code>autorun/node_modules</code> — символическую ссылку на <code>node_modules</code> веб-интерфейса (оба пакета объявлены в <code>frontend/package.json</code>).
util-linux (taskset)	Кнопка «Start» в админке привязывает службу к Е-ядрам через <code>taskset</code> . Есть практически в любом дистрибутиве; установщики добавляют его явно.
Позывной + локатор	Читаются из <code>frontend/site_information.json</code> (<code>siteSysop</code> / <code>siteGridSquare</code>), если не переопределены во вкладке администратора. Споты выгружаются под этим позывным.

⚠ **Сообщайте только то, что действительно принимаете.** Споты выгружаются в публичные сети под вашим позывным — включайте только те диапазоны и виды связи, которые ваш приёмник действительно слышит, и указывайте правильный локатор.

Автоматическая установка

`install.sh` (и варианты для дистрибутивов: `install_arch.sh`, `install_fedora.sh`, `install_opensuse.sh`) делают всё за вас: устанавливают Node.js 22 и `util-linux`, выполняют `npm install` для веб-интерфейса и затем автоматически создают символическую ссылку `autorun/node_modules`. Дополнительных шагов не требуется — функция готова, как только `install.sh` завершится.

Установка вручную

ПРИМЕЧАНИЕ

Установщик уже делает это за вас: создаёт символическую ссылку и устанавливает `util-linux`. Шаги ниже нужны только для ручной починки.

Если вы устанавливали ручную (вариант Б), создайте символическую ссылку сами после `npm install` веб-интерфейса, чтобы служба смогла разрешить свои зависимости:

```
cd ~/PhantomSDR-Plus

# 1. Frontend deps must be installed first (ws + cbor-x live there)
cd frontend && npm install && cd ..

# 2. Link the autorun daemon to the frontend's node_modules
ln -sf ../frontend/node_modules autorun/node_modules

# 3. Ensure taskset is available (Debian/Ubuntu shown; use your package manager)
sudo apt install -y util-linux
```

Примечание: `autorun/node_modules` игнорируется `git`, поэтому свежий `git clone` никогда его не содержит — ссылку нужно (пере)создавать после каждой чистой выгрузки. Установщики делают это за вас; команда выше нужна только при ручной настройке.

Проверка

```
# The symlink should point at ../frontend/node_modules
ls -l autorun/node_modules

# ws + cbor-x must resolve
ls -d frontend/node_modules/ws frontend/node_modules/cbor-x

# taskset must be on PATH
command -v taskset
```

Затем откройте панель администратора, перейдите на вкладку **Spot Reporting**, укажите свои данные, отметьте диапазоны и виды связи для декодирования, включите приёмники отчётов и нажмите **Start**. На главном водопаде появится значок «REPORTING», пока отчётность активна. (PSK Reporter выгружает пакетами каждые 5 минут, а wsprnet — каждые 2 минуты, поэтому только что запущенная служба первые минуты показывает «0 sent» — это нормально.)

Далее показываются два разных счётчика, которые легко перепутать. Плитки **SPOTS UPLOADED PER DECODER** считают только текущий запуск, поэтому Stop/Start обнуляет их; число рядом с каждым флажком диапазона/вида связи — это **итог за всё время** для этой позиции, он хранится в `autorun-totals.json` и переживает перезапуски. Оба счётчика описаны в [руководстве по панели администратора](#), там же — страница **Графики**, где строятся частота процессора, нагрузка, температура и число слушателей за период от 15 минут до 24 часов.

Если ваша машина греется, в панели есть и [тепловая защита](#), которая останавливает сервер при опасной температуре процессора и запускает его снова после остывания. Пороги она выводит из критического порога вашего собственного процессора, так что считать ничего не нужно, и работает она с любым способом запуска и остановки. Устанавливается вместе с панелью, но начинает в режиме только записи: отмечает, что *сделала бы*, и ничего не меняет, пока вы не включите её в Настройках. Непрерывное декодирование нагружает процессор круглые сутки, поэтому стоит заглянуть во вкладку CRASH после недели работы autorun и посмотреть, насколько близко ваша машина подходит на самом деле.

Порт сервера определяется автоматически. Служба подключается напрямую к `[server]` port самого spectrumserver (петлевой интерфейс + токен, минуя прокси). Она читает этот порт из файла конфигурации, с которым был запущен работающий spectrumserver, поэтому работает на любом порту без настройки — строка `[autorun] tap backend: ...` в `autorun.log` показывает, что было определено. Если ваш сервер не работал в момент запуска службы или у вас нестандартная конфигурация, укажите порт явно в `autorun.json`: `json "server": { "host": "127.0.0.1", "port": 9002 }` Неверный порт проявляется как 504 / tap closed 1006 в `autorun.log`, а `decodes` остаётся равным 0.

Эмуляция клиентов KiwiSDR (необязательно)

Начиная с версии 4.1.0 PhantomSDR-Plus умеет отвечать и на **протокол KiwiSDR**, так что программы, написанные для KiwiSDR — **AetherSDR**, `kiwiclient` и прочие — подключаются к вашему приёмнику напрямую, на том же хосте и порту, который вы уже публикуете. Он выключен, пока в конфигурацию, с которой работает ваш приёмник, не добавлено `[kiwi_emulation] enabled = true`.

Установщик предлагает его шагом 17; `./kiwi_install.sh` применяет его к уже установленному дереву.

Полная документация: [Эмуляция клиентов KiwiSDR](#) — что делает мост, как его установить, каждый ключ `[kiwi_emulation]`, подключение клиента, уровень звука, S-метр, частота водопада и таблица симптомов.

Настройка

1. Выберите файл конфигурации

Выберите файл конфигурации, подходящий для вашего SDR:

- `config-rtl.toml` — свистки RTL-SDR
- `config-rspla.toml` — SDRplay RSP1A
- `config-airspyhf.toml` — Airspy HF+ Discovery
- `config-fobos-hf.toml` — RigExpert Fobos SDR, HF1/HF2 с прямой оцифровкой (0-25 МГц)
- `config-fobos.toml` — RigExpert Fobos SDR, тракт RF (25-6000 МГц)
- `config-rx888mk2.toml` — RX888 MK2
- `config-hackrf.toml` — HackRF One
- `config.example.hackrf.toml` — HackRF One

В этом примере используем RTL-SDR.

2. Отредактируйте файл конфигурации

```
nano config-rtl.toml
```

Ключевые параметры

```
[server]
port = 9002                # Web interface port (or everything else)
html_root = "frontend/dist/" # Frontend location
otherusers = 1             # Show other users (1=yes, 0=no)
threads = 2                # Number of server threads

[websdr]
register_online = true      # Register on sdr-list.xyz (true/false)
name = "Your WebSDR Name"  # Display name
antenna = "Your Antenna Type" # e.g., "Vertical", "Loop", "Dipole"
grid_locator = "AB12cd"    # Your Maidenhead grid square
hostname = "your.domain.com" # Your domain or IP address

[input]
sps = 2048000              # Sample rate (adjust for your coverage)
fft_size = 131072          # FFT size (higher = better resolution)
brightness_offset = -10    # Waterfall brightness adjustment
frequency = 145000000       # Base frequency in Hz (145 MHz for 2m)
signal = "iq"              # "iq" for complex, "real" for real sampling
fft_threads = 2            # FFT processing threads
accelerator = "opencl"     # "none", "cuda", or "opencl"
audio_sps = 12000          # Audio sample rate (keep at 12000)
audio_compression = "opus" # "flac" or "opus"
smeter_offset = -2         # S-meter calibration
waterfall_size = 1024      # Waterfall FFT size
waterfall_compression = "zstd" # Waterfall compression

[input.driver]
name = "stdin"             # Input driver
format = "u8"              # Sample format for RTL-SDR

[input.defaults]
frequency = 145500000       # Default tuning frequency
modulation = "FM"          # Default modulation mode
```

Ориентиры по частоте дискретизации

Охват	Частота дискретизации	Размер FFT
2 МГц	2048000	131072
3.2 МГц	3200000	131072
10 МГц	10000000	1048576
30 МГц	30000000	2097152
60 МГц	60000000	4194304

3. Настройте сведения о площадке

```
nano frontend/site_information.json
```

Отредактируйте следующие поля:

```
{
  "siteSysop": "YourCallsign",
  "siteSysopEmailAddress": "your@email.com",
  "siteGridSquare": "AB12cd",
  "siteCity": "Your City, Country",
  "siteInformation": "https://github.com/sv1btl/PhantomSDR-Plus",
  "siteHardware": "Computer specifications",
  "siteSoftware": "PhantomSDR-Plus v5.0.0",
  "siteReceiver": "Your SDR model",
  "siteAntenna": "Antenna description",
  "siteNote": "Additional information",
  "siteIP": "http://your.domain.com:9002",
  "siteSDRBaseFrequency": 0,
  "siteSDRBandwidth": 2048000,
  "siteRegion": 1,
  "siteChatEnabled": true
}
```

Районы IARU:

- **1:** Европа, Африка, Ближний Восток, Северная Азия
- **2:** Америка (Северная, Центральная, Южная), Карибский бассейн
- **3:** Азиатско-Тихоокеанский регион, Океания

4. Настройте частотные маркеры (необязательно)

```
nano markers.json
```

Добавьте свои любимые частоты, ретрансляторы и вещательные станции.

5. Отредактируйте скрипт запуска

```
nano start-rtl.sh
```

Скрипт запуска — самодостаточный лаунчер со сторожевым процессом. Правьте только блок **RECEIVER CONFIGURATION** в начале файла, чтобы аргументы приёмника соответствовали вашей конфигурации:

```
# == RECEIVER CONFIGURATION (the only receiver-specific part) ==
RX_LABEL="RTL-SDR"
RX_COMM="rtl_sdr"                # process name to monitor/kill
CONFIG="$PHANTOMDIR/config-rtl.toml" # your .toml
FIFO="$PHANTOMDIR/rtl.fifo"
RX_ARGS="${RX_ARGS:-f 145000000 -s 2048000 -}" # receiver args (edit these)
```

Параметры внутри RX_ARGS:

- **-f 145000000:** центральная частота (145 МГц)
- **-s 2048000:** частота дискретизации (2,048 MSPS)

CONFIG указывает на ваш .toml. Больше **ничего** в скрипте править не нужно — логика запуска/перезапуска/сторожа/журналирования универсальна. Аргументы приёмника можно также переопределить при запуске, не редактируя файл: `RX_ARGS="-f 145000000 -s 2048000 -" ./start-rtl.sh` (RX888_ARGS для start-rx888mk2.sh).

Настройка для конкретных устройств SDR

RTL-SDR

Установите инструменты RTL-SDR

```
sudo apt install -y rtl-sdr
```

Проверьте RTL-SDR

```
rtl_test
```

Нажмите Ctrl+C для остановки. Вы должны увидеть сведения о частоте дискретизации.

Отредактируйте конфигурацию

```
nano config-rtl.toml
```

Типичные настройки для RTL-SDR:

```
sps = 2048000
frequency = 145000000
signal = "iq"
format = "u8"
```

Приёмники через SoapySDR (RSP1A, Fobos, Airspy HF+)

SDRplay RSP1A, RigExpert Fobos SDR и Airspy HF+ подключаются к серверу одинаково: драйвер SoapySDR для устройства и `rx_sdr` (из `rx_tools`), который передаёт его отсчёты в `spectrumserver`. Один сценарий на приёмник ставит всю цепочку:

Приёмник	Сценарий	Пункт установщика	Сценарий запуска	Конфигурация
SDRplay RSP1A	<code>./setup-rspla.sh</code>	3	<code>start-rspla.sh</code>	<code>config-rspla.toml</code>
RigExpert Fobos SDR	<code>./setup-fobos.sh</code>	5	<code>start-fobos-hf.sh</code> (HF) / <code>start-fobos.sh</code> (RF)	<code>config-fobos-hf.toml</code> / <code>config-fobos.toml</code>
Airspy HF+	<code>./setup-airspyhf.sh</code>	6	<code>start-airspyhf.sh</code>	<code>config-airspyhf.toml</code>

Каждый сценарий:

- ставит `git`, `cmake`, компилятор, `libusb` и **SoapySDR** из пакетов вашего дистрибутива — он распознаёт `apt`, `dnf`, `pacman` и `zypper`, поэтому одинаково работает на Debian/Ubuntu, Fedora, Arch и openSUSE;
- собирает драйвер приёмника из исходников в `sdr_drivers/` (не отслеживается `git`);
- собирает **rx_tools** для `rx_sdr`, если `rx_sdr` ещё не установлен — одна копия служит всем приёмникам;
- ставит **правило udev** и добавляет вас в группу `plugdev`, чтобы сценарию запуска никогда не требовался `sudo` для устройства. **Один раз выйдите из системы и войдите снова**, чтобы группа вступила в силу;
- в конце проверяет, что SoapySDR действительно видит драйвер, и иначе останавливается с ошибкой.

Запускайте их от обычного пользователя — они сами вызывают `sudo`. Они идемпотентны: повторный запуск обновляет исходники драйвера через `git pull` и пересобирает. Установщик вызывает именно эти сценарии; на уже установленной станции запустите отдельно сценарий своего приёмника. Общая часть находится в `setup-sdr-common.sh`, который сам по себе не запускается. Две необязательные переменные действуют для всех трёх: `SDR_USER=<имя>` добавляет в `plugdev` другую учётную запись, а `SDR_SKIP_DEPS=1` пропускает установку пакетов.

При подключённом устройстве проверьте, что приёмник обнаружен:

```
SoapySDRUtil --find="driver=soapyMiri"    # RSP1A
SoapySDRUtil --find="driver=fobos"        # Fobos
SoapySDRUtil --find="driver=airspyhf"     # Airspy HF+
```

SDRplay RSP1A

`setup-rspla.sh` ставит **драйвер с открытым кодом**: `libmirisdr-5`, полученный обратной разработкой драйвер для микросхем Mirics MSi2500/MSi001 внутри RSP1 и RSP1A, с его модулем `SoapySDR` — `SoapyMiri` (`driver=soapyMiri`). Ему не нужны фоновая служба и `root`. Сценарий также вносит драйверы ядра `msi2500` и `msi001` в чёрный список в `/etc/modprobe.d/blacklist-msi2500.conf`, потому что они захватывают RSP1A как радиоустройство V4L2 раньше, чем его сможет открыть `rx_sdr`. Если устройство было подключено, пока они были загружены, один раз отключите и снова подключите его (или перезагрузитесь).

Закрытый API `SDRplay` (`driver=sdrplay`, с `SoapySDRPlay` и службой `sdrplay`) по-прежнему работает — приложенный файл `instructions-for-rspla` описывает такую установку. `start-rspla.sh` сам выбирает между ними: если в `SoapySDR` есть драйвер `sdrplay`, он использует API и сначала перезапускает службу `sdrplay`, иначе — `libmirisdr-5`. Станция, настроенная на API, поэтому продолжает работать без изменений. Чтобы задать драйвер принудительно, запускайте с `RX_DRIVER=miri` или `RX_DRIVER=sdrplay`. Выбранный драйвер записывается в `logwebsdr.txt` как `RSP1A driver: miri` или `RSP1A driver: sdrplay`.

RigExpert Fobos SDR

Fobos передаёт **только один тракт одновременно**, поэтому у него два сценария запуска, которые делят блокировку сторожа, как любой `start-*.sh`, — запуск одного останавливает другой:

- **`start-fobos-hf.sh`** — входы HF1/HF2 с прямой оцифровкой, 0-25 МГц, на порту 9003. В этом режиме нет гетеродина: АЦП всё время оцифровывает 0-25 МГц (`-f 0` задано намеренно, а предупреждение *Failed to set center freq* от `rx_sdr` ожидаемо). `rx_sdr` выдаёт CF32 на 50 Msps; маленький преобразователь `cf32_to_real` (его собирает `setup-fobos.sh` из `cf32_to_real.c`) оставляет канал I и масштабирует его в `s16`, поэтому в `config-fobos-hf.toml` стоят `signal="real"` и `format="s16"`.
- **`start-fobos.sh`** — тракт RF, 25-6000 МГц, IQ, на порту 9002. Выберите окно до первого запуска: `-f` (центр) и `-s` (частота дискретизации) в строке `RX_ARGS` должны совпадать с `frequency=` и `sps=` в `config-fobos.toml`. Поставляемые значения (97 МГц, 20 Msps) — лишь отправная точка, чтобы проверить, что всё работает.

ВАЖНО

`config-fobos-hf.toml` поставляется с `accelerator="opencl"`: поток 50 Msps через БПФ на 1М бинов так же тяжёл, как 60 Msps у RX888, и на CPU он теряет отсчёты и греет машину. На машине без устройства OpenCL `spectrumserver` с этой настройкой не запустится — поставьте там `accelerator="none"`.

`SoapyFobosSDR` для сборки нужны заголовки и `libfobos`, и `libfobos-sdr-agile`, поэтому `setup-fobos.sh` собирает обе, даже для штатной прошивки. Поддержка Fobos пришла из сценария, который

написала другая станция, запустив Fobos на PhantomSDR-Plus; значения HF в `config-fobos-hf.toml` — те, что сработали там.

Airspy HF+

`setup-airspyhf.sh` ставит `libairspyhf` из дистрибутива там, где она есть в пакетах (Debian и Ubuntu, Fedora, openSUSE), и собирает её из исходников там, где её нет (в Arch она есть только в AUR), затем собирает `SoapyAirspyHF` (`driver=airspyhf`). `start-airspyhf.sh` выдаёт 912 kbps IQ в CS16, что `config-airspyhf.toml` читает как `format="s16"`. Приложенный файл `instructions-for-airspy` описывает ручной путь.

RX888 MK2

Установите инструменты RX888

```
# Install rx_tools
git clone https://github.com/rxseger/rx_tools.git
cd rx_tools
mkdir build && cd build
cmake ..
make -j4
sudo make install
sudo ldconfig

# Install RX888 firmware and support
# Follow manufacturer instructions
```

Запуск rx888_stream без sudo (правила udev)

По умолчанию USB-устройство Cypress FX3 у RX-888 доступно только root, поэтому `rx888_stream` потребовал бы `sudo`. Выполните один раз входящий в комплект вспомогательный скрипт, чтобы установить правила udev и добавить себя в группу `plugdev`:

```
./setup-rx888-udev.sh          # re-runs itself with sudo automatically
```

Он записывает `/etc/udev/rules.d/99-rx888.rules` для всех трёх идентификаторов продуктов Cypress FX3 (`04b4:00f1/00f3` — загрузчик и `04b4:8613` — с загруженной прошивкой), перезагружает udev и создаёт постоянную ссылку `/dev/rx888`. **Выйдите из системы и войдите снова** (чтобы применилось изменение группы) и один раз **переподключите устройство**; после этого `rx888_stream` — и лаунчер `start-rx888mk2.sh` — работают без `sudo`. Скрипт идемпотентен, его безопасно запускать повторно. Для другой учётной записи: `RX888_USER=<имя> ./setup-rx888-udev.sh`.

Отредактируйте конфигурацию

```
nano config-rx888mk2.toml
```

Типичные настройки для RX888:

```
sps = 60000000
frequency = 0
signal = "real"
format = "s16"
fft_size = 4194304
```

HackRF One

Для HackRF ничего не собирается: `hackrf_transfer` из пакета `hackrf` вашего дистрибутива передаёт отсчёты прямо в `spectrumserver`. `./setup-hackrf.sh` (пункт 7 установщика) ставит этот пакет через `apt`, `dnf`, `pacman` или `zypper` и добавляет правило `udev` и группу `plugdev`, чтобы `start-hackrf.sh` работал без `sudo`, — после этого один раз выйдите из системы и войдите снова.

```
./setup-hackrf.sh    # installs the hackrf package and the udev rule
hackrf_info         # with the HackRF plugged in: must list the board
```

`start-hackrf.sh` передаёт одно окно одновременно. Выберите его до первого запуска: `-f` (центр) и `-s` (частота дискретизации) в его строке `RX_ARGS` должны совпадать с `frequency=` и `sps=` в `config-hackrf.toml`. `-l` и `-g` — усиление LNA (0-40 дБ, шаг 8 дБ) и VGA (0-62 дБ, шаг 2 дБ), а `-a 1` включает усилитель RF. Поставляемые значения (98 МГц, 20 Msps) — лишь отправная точка, чтобы проверить, что всё работает.

`hackrf_transfer` пишет 8-битный IQ **со знаком**, поэтому в конфигурации нужен `format="s8"`. В `config.example.hackrf.toml` до сентября 2026 года стояло `u8`; с `u8` каждый отсчёт сдвинут на 128 и водопад показывает только шум — проверьте старые копии.

Несколько приёмников на одном компьютере

Один компьютер может одновременно обслуживать несколько перечисленных выше приёмников — например, RX-888 для KB и RTL-SDR для 2 м, — каждый как отдельный сервер, с выбором приёмника на странице. Каждый дополнительный приёмник запускается командой `INSTANCE=<имя> ./start-<радио>.sh` и настраивается в `instances/<имя>/receivers.toml` перечисляет их для прокси и для `start-all.sh`. Всё описано в разделе [Несколько приёмников](#). Быстрее всего добавить приёмник командой `./add-receiver.sh`, которую установщики предлагают в конце: она устанавливает драйвер (`setup-rtlsdr.sh` для RTL-SDR, соответствующий `setup-*.sh` для остальных) и сама записывает папку экземпляра и `receivers.toml`.

Проверка и тестирование

1. Пробный запуск

```
# For RTL-SDR
./start-rtl.sh
```

Это запускает сервер **в фоне** (скрипт отсоединяется и сразу возвращает управление) и начинает вести журнал в `logwebsdr.txt`.

2. Проверьте наличие ошибок

Скрипт записывает ход работы в `logwebsdr.txt` — следите за ним в реальном времени:

```
tail -f logwebsdr.txt
```

Обратите внимание на:

- `Starting the initialization script of the PhantomSDR Server`
- `<receiver> running (PID ...)`
- `Server started normally`

- `Server ... FAILED ...` или `ERROR: receiver binary '...' not found` (исправьте аргументы приёмника либо `.toml`, или установите утилиту приёмника, затем снова запустите скрипт)

3. Откройте веб-интерфейс

Перейдите в браузере по адресу:

```
http://localhost:9002
```

(Замените номер порта на настроенный вами)

4. Проверьте работоспособность

- Водопад должен отображаться
- Звук должен воспроизводиться при щелчке по сигналам
- S-метр должен реагировать на сигналы
- Счётчик пользователей должен показывать «1»

5. Проверьте с другого устройства

С другого компьютера в вашей сети:

```
http://YOUR_SERVER_IP:9002
```

6. Проверьте потребление ресурсов

```
# Monitor CPU usage
htop

# Monitor GPU usage (if OpenCL/CUDA enabled)
nvidia-smi # For NVIDIA
radeontop  # For AMD

# Monitor network usage
iftop
```

7. Остановите сервер

Сервер работает в фоне под сторожевым процессом, поэтому **Ctrl+C его не остановит** (а сторож просто перезапустит его). Используйте общий скрипт остановки, работающий для любого приёмника: он сначала останавливает сторожа, затем приёмник и `spectrumserver`:

```
./stop-websdr.sh
```

Настройка автозапуска

С помощью systemd (рекомендуется)

1. Создайте файл службы

```
sudo nano /etc/systemd/system/phantomsdr.service
```

Добавьте следующее содержимое (поправьте пути и пользователя):

```
[Unit]
Description=PhantomSDR-Plus WebSDR Server
After=network.target

[Service]
Type=simple
User=youruser
WorkingDirectory=/home/youruser/PhantomSDR-Plus
# Run the watchdog in the FOREGROUND (note the --watchdog flag) so systemd can
# track it. Do NOT use the plain "./start-rtl.sh" here – that form detaches into
# the background and exits, which systemd would treat as the service stopping.
ExecStart=/home/youruser/PhantomSDR-Plus/start-rtl.sh --watchdog
ExecStop=/home/youruser/PhantomSDR-Plus/stop-websdr.sh
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Процесс `--watchdog` уже сам перезапускает приёмник и `spectrumserver`; `Restart=always` — лишь подстраховка на редкий случай, когда завершится сам сторож. Поскольку здесь сервером управляет `systemd`, можно использовать `systemctl start/stop/restart` и `journalctl -u phantomsdr -f` вместо ручного запуска скриптов.

2. Включите и запустите службу

```
# Reload systemd
sudo systemctl daemon-reload

# Enable service to start on boot
sudo systemctl enable phantomsdr.service

# Start service now
sudo systemctl start phantomsdr.service

# Check status
sudo systemctl status phantomsdr.service
```

3. Управление службой

```
# Start
sudo systemctl start phantomsdr

# Stop
sudo systemctl stop phantomsdr

# Restart
sudo systemctl restart phantomsdr

# View logs
sudo journalctl -u phantomsdr -f
```

С помощью Screen (альтернатива)

Обычно не требуется: `./start-rtl.sh` уже уходит в фон (через `setsid`) и продолжает работать после выхода из системы, со своим сторожевым процессом. `Screen` полезен, только если вам специально нужен интерактивный сеанс для запуска формы переднего плана `./start-rtl.sh --watchdog`.

1. Установите Screen

```
sudo apt install -y screen
```

2. Запустите в сеансе Screen

```
screen -S phantomsdr
./start-rtl.sh
```

Нажмите Ctrl+A, затем D, чтобы отсоединиться.

3. Вернитесь к сеансу

```
screen -r phantomsdr
```

Тепловая защита процессора

Полное руководство: [THERMAL_GUARD.md](#) — четыре режима и что должен сделать сисоп в каждом, работа без панели, этап throttle без root, проверка и устранение неполадок.

В составе PhantomSDR-Plus есть страж, который останавливает сервер, если процессор достигает опасной температуры, и запускает его снова после остывания. Пороги он выводит из критического порога, публикуемого вашим собственным процессором, так что считать ничего не нужно, и работает он с любым способом запуска и остановки.

Если вы используете панель администратора, она у вас уже есть — защита работает внутри панели и настраивается на её странице «Настройки». См. [руководство по панели администратора](#). Остальная часть этого раздела — для установок **без** панели.

1. Посмотрите, что защита видит на вашей машине

```
cd ~/PhantomSDR-Plus
python3 thermal_guard.py --once
```

```
sensor      : hwmon:coretemp
temperature: 58.0 C
crit        : 100.0 C
mode        : log
thresholds  : warn 88.0  throttle 92.0  stop 95.0  resume 75.0 (C)
cpufreq     : not writable – throttle stage disabled
```

Используются только настоящие датчики на кристалле процессора (`coretemp`, `k10temp`, `zenpower`, `cpu_thermal`); `asptitz` и безымянные тепловые зоны намеренно игнорируются, поскольку они часто сообщают температуру корпуса или платы на десятки градусов ниже процессорной. Если выводится `sensor : NONE`, эту машину защитить нельзя — обычное дело на VPS или внутри контейнера — и защита останется неактивной, а не будет делать вид.

Устанавливать ничего не нужно: `thermal_guard.py` использует только стандартную библиотеку Python.

2. Настройте её

Защита читает `admin_config.json` рядом с `thermal_guard.py`. Создайте его, если у вас такого нет — без панели его и не будет. Минимально полезный файл:

```
{
  "sdr_process_name": "spectrumserver",
  "stop_script": "stop-websdr.sh",
  "start_script": "start-rx888mk2.sh",
  "thermal_mode": "stop+restart"
}
```

- `thermal_mode` — `log` (только запись, по умолчанию), `throttle`, `stop` или `stop+restart`. **Именно его нужно задать**, потому что защита поставляется бездействующей: если её не трогать, она лишь записывает, что *сделала бы*. Его же можно передать в командной строке как `--mode stop+restart` — это имеет приоритет над файлом, так что если все прочие значения по умолчанию вас устраивают, файл конфигурации вообще не нужен.
- `stop_script` / `start_script` — чем она останавливает и запускает ваш сервер. Без скрипта остановки она прибегает к `SIGTERM`, а через 10 секунд — к `SIGKILL` по `sdr_process_name`. Без скрипта запуска она останавливает, но никогда не перезапускает.
- Пороги и времена тоже можно задать здесь, теми же именами ключей, что и в панели, — полная таблица есть в [руководстве по панели администратора](#).

Если вашим SDR-сервером управляет systemd, направьте `stop_script` на небольшой скрипт, выполняющий `systemctl stop` ваш-юнит, вместо того чтобы защита посылала сигнал процессу напрямую. Защита победит в любом случае — пока процессор слишком горяч, она повторяет остановку каждые 2 секунды, так что всё, что оживляет сервер, будет отменено, — но чистая остановка лучше борьбы каждые две секунды.

3. Запустите её как службу

В репозитории есть готовый юнит `thermal-guard.service`. Отредактируйте в нём `User=` и два пути, затем:

```
sudo cp thermal-guard.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable --now thermal-guard
systemctl status thermal-guard
```

`User=` должен быть учётной записью, которой разрешено запускать ваши скрипты запуска и остановки, — обычно тот же пользователь, что запускает SDR-сервер. **Защита, работающая от пользователя, который не может остановить сервер, оставляет вас без защиты, при этом выглядя защищённой.**

Чтобы взвести защиту из юнита, а не из файла конфигурации, добавьте `--mode` в `ExecStart`:

```
ExecStart=/usr/bin/python3 -u /home/sysop/PhantomSDR-Plus/thermal_guard.py --mode stop+restart
```

Без `systemd`:

```
cd ~/PhantomSDR-Plus
nohup python3 -u thermal_guard.py --mode stop+restart >> thermal.log 2>&1 &
```

`python3 thermal_guard.py --help` перечисляет все параметры, включая `--config` для файла конфигурации не рядом со скриптом.

4. Понаблюдайте, потом доверяйте

Всё, что делает защита, записывается в `crash.log` в каталоге `PhantomSDR-Plus`, по одному событию на строку:

```
[THERMAL] warn: 89.0C (warn=88.0 crit=100.0) sustained 31s
[THERMAL] STOPPED server: 96.0C sustained 62s (stop=95.0 crit=100.0) – Started stop-websdr.sh
[THERMAL] lockout: process reappeared at 94.0C – re-stopped [3 time(s) so far]
[THERMAL] recovered: 74.0C held below 75.0 – lockout cleared
```

```
grep THERMAL crash.log
```

Сначала оставьте `thermal_mode` в `"log"` на неделю и почитайте этот файл после тяжёлых моментов — полной пересборки, жаркого дня. Если там ничего нет, ваша машина даже близко не подходила. После этого переключите режим на `stop`.

Прежде чем полагаться на защиту, проверьте весь путь один раз, добавив фиктивное показание выше вашего порога остановки:

```
"thermal_test_temp": 97
```

Защита воспримет его как настоящее, и `warn` → `stop` → блокировка → восстановление → перезапуск пройдут по вашему желанию. Уберите строку (или задайте `null`), чтобы вернуться к реальному датчику. В режиме `stop` сервер при этом действительно останавливается и ваши слушатели отключаются, поэтому делайте это, когда никого нет, или проверяйте в режиме `log`, где видно, что *произошло бы*, ничего не трогая.

Устранение неполадок

Ошибки сборки

Проблемы с зависимостями

```
# Ubuntu: Install missing dependencies
sudo apt install --fix-broken

# Fedora: Install missing dependencies
sudo dnf install <package-name>
```

Не удаётся выполнить конфигурацию Meson

```
# Clean and reconfigure
rm -rf build
meson setup build --buildtype=release
```

meson setup завершается с `ModuleNotFoundError: No module named 'mesonbuild'`

```
Traceback (most recent call last):
  File "/home/<user>/local/bin/meson", line 3, in <module>
    from mesonbuild.mesonmain import main
ModuleNotFoundError: No module named 'mesonbuild'
```

С исходным деревом всё в порядке. Оставшаяся установка `pip install --user meson` оставила скрипт запуска в `~/local/bin`, который в `PATH` идёт раньше `/usr/bin` и потому скрывает рабочую копию, установленную менеджером пакетов. Обновление дистрибутива (например, Ubuntu 24.04 → 26.04) переводит Python на новую версию, старый `site-packages` с `mesonbuild` больше не находится в пути импорта, и скрипт падает, ничего не успев сделать. То же самое может случиться с `ninja`.

Установщик это распознаёт и обходит на время запуска, предупреждая вас, но систему всё же почините:

```
rm -f ~/.local/bin/meson
hash -r
meson --version          # должна вывести версию
```

Если вы предпочитаете оставить meson, установленный через pip, переустановите его для того Python, который сейчас есть в системе:

```
python3 -m pip install --user --force-reinstall --break-system-packages meson
```

meson setup **останавливается с Clock skew detected**

```
ERROR: Clock skew detected. File /home/pi/PhantomSDR-Plus/build/../meson.build has a time stamp
10392.2144s in the future.
```

С исходным деревом всё в порядке: системные часы отстают от файлов. meson и ninja отказываются собирать, когда входной файл новее текущего времени, — они не могут определить, что устарело. Так бывает на Raspberry Pi без батарейки в держателе RTC: каждая загрузка начинается с последнего известного времени, и сборка, запущенная до синхронизации NTP, видит всё дерево датированным будущим. То же самое — с деревом, распакованным или скопированным с машины, часы которой спешат.

Сначала исправьте часы:

```
timedatectl                # время верное? NTP синхронизирован?
sudo timedatectl set-ntp true
```

Подождите несколько секунд и запустите установщик снова. Он проверяет это перед вызовом meson и предлагает сбросить проблемные временные метки (PHANTOM_FIX_CLOCK_SKEW=y|n). Вручную, из каталога с исходниками:

```
find . -path ./build -prune -o -path ./git -prune -o -newermt now -print0 | xargs -0r touch
```

Бэкенд собрался, но веб-страницы нет

Симптом — установка, которая выглядит почти удавшейся: build/spectrumserver на месте, сервер запускается, а браузер не получает ничего, потому что каталог frontend/dist/ так и не был создан. Поищите в выводе установщика:

```
[vite-plugin-top-level-await] missing field `type`
  at Compiler.printSync (node_modules/@swc/core/index.js:257:29)
error during build:
  Failed: Default
```

vite-plugin-top-level-await требует @swc/core ^1.12.14, поэтому чистый npm install разрешается в 1.16.0, чей printSync() отвергает синтаксическое дерево, которое ему передаёт плагин. К вашему дистрибутиву это отношения не имеет: теперь рабочую версию закрепляет каждый установщик, а также frontend/package.json. Машина, установленная до появления этого закрепления, продолжает работать, пока не удалён её node_modules — вот почему при переустановке проблема возникает словно из ниоткуда.

Если вы чините более старую копию вручную:

```
cd frontend
npm pkg set 'overrides.@swc/core=1.15.33'
npm install
./build-all.sh
```

Ошибки времени выполнения

Порт уже занят

```
# Find process using port
sudo lsof -i :9002

# Kill process
sudo kill -9 <PID>
```

Отказано в доступе к SDR

```
# Add user to plugdev group
sudo usermod -a -G plugdev $USER

# Reload groups (or log out and back in)
newgrp plugdev
```

Для **RX-888 MkII** одного этого мало — устройству Cypress FX3 нужны ещё и правила udev. Запустите входящий в комплект скрипт (он устанавливает правила и добавляет вас в plugdev), затем выйдите и войдите снова и переподключите устройство:

```
./setup-rx888-udev.sh
```

Проблемы со звуком

```
# Try different codec
# Edit config file and change:
audio_compression = "flac" # or "opus"

# Rebuild
cd frontend && npm run build && cd ..
```

Проблемы с OpenCL

clinfo не показывает устройств

```
# Verify drivers are loaded
sudo clinfo

# Check OpenCL ICD files
ls /etc/OpenCL/vendors/

# Reinstall drivers (Intel example)
sudo apt remove --purge intel-opencl-icd
sudo apt install intel-opencl-icd
```

Производительность не выросла

```
# Verify OpenCL is enabled in config
accelerator = "opencl"

# Try different device
# Add to config file:
[input.opencl]
device_id = 0 # Try 0, 1, 2, etc.
```

Проблемы с сетью

Нет доступа с других устройств

```
# Check firewall
sudo ufw status
sudo ufw allow 9002/tcp

# Or disable firewall temporarily for testing
sudo ufw disable
```

Высокая задержка

```
# Reduce waterfall size in config
waterfall_size = 512

# Increase buffer size
buffer_size = 16384

# Use Opus instead of FLAC
audio_compression = "opus"
```

Проблемы с устройством SDR

RTL-SDR не найден

```
# Check if device is recognized
lsusb | grep Realtek

# Test with rtl_test
rtl_test

# Try blacklisting DVB-T drivers
echo "blacklist dvb_usb_rtl28xxu" | sudo tee /etc/modprobe.d/blacklist-rtl.conf
sudo rmmod dvb_usb_rtl28xxu
```

SDRplay не найден

`grep "RSP1A driver" logwebsdr.txt` показывает, какой драйвер выбрал `start-rspla.sh`.

С открытым драйвером (libmirisdr-5, RSP1A driver: miri):

```
# The kernel's msi2500 driver must not hold the device
lsmod | grep msi2500 && sudo modprobe -r msi2500 msi001

# Test with SoapySDR
SoapySDRUtil --find="driver=soapyMiri"
```

С API SDRplay (RSP1A driver: sdrplay):

```
# Check API installation
systemctl status sdrplay

# Restart API
sudo systemctl restart sdrplay

# Test with SoapySDR
SoapySDRUtil --find="driver=sdrplay"
```

Fobos или Airspy HF+ не найден

```
SoapySDRUtil --info | grep -i factories # must list fobos / airspyhf
SoapySDRUtil --find="driver=fobos"      # or driver=airspyhf
id | grep plugdev                       # log out and back in if missing
```

Если драйвера нет в списке, запустите `./setup-fobos.sh` или `./setup-airspyhf.sh` ещё раз: при сбое сборки или проверки SoapySDR он остановится и назовёт причину.

Оптимизация производительности

Оптимизация процессора

```
# Adjust thread counts based on CPU cores
[server]
threads = 4 # Set to number of cores

[input]
fft_threads = 4 # Set to number of cores
```

Оптимизация памяти

```
# Reduce memory usage
[input]
waterfall_size = 512 # Lower value = less memory
fft_size = 65536     # Lower value = less memory
```

Оптимизация сети

```
# Optimize for limited bandwidth
[input]
audio_compression = "opus" # More efficient than FLAC
waterfall_compression = "zstd" # Efficient compression
```

Обновление PhantomSDR-Plus

Начиная с версии 4.1.0 в репозитории есть `update.sh` — средство обновления, которое приводит установленный приёмник в соответствие с опубликованным деревом, **не трогая файлы, которые делают его именно вашей станцией**. Оно заменяет самодельный сценарий с `git pull`, который предлагали писать прежние редакции этого руководства, и `git` ему вовсе не нужен: опубликованное дерево загружается как tarball и сравнивается с вашим файлом за файлом. Поэтому оно работает одинаково — клонировали ли вы репозиторий, распаковали `update.zip` или скопировали дерево с флешки.

Если в вашей установке ещё нет update.sh

Старое дерево этого сценария не содержит. Загрузите его один раз — это единственный шаг всей процедуры, который вы когда-либо делаете вручную:

```
cd ~/PhantomSDR-Plus
curl -fLO https://raw.githubusercontent.com/sv1btl/PhantomSDR-Plus/main/update.sh
chmod +x update.sh
```

Дальше всё — исходники, frontend, документация, установщики и сам update.sh — приходит через этот инструмент.

Шаг 1 — посмотреть, что изменилось бы (ничего не записывается)

```
cd ~/PhantomSDR-Plus
./update.sh
```

Загружает опубликованное дерево, сравнивает его с вашим и печатает отчёт. Не записывает ровно ничего, поэтому запускать можно в любой момент, в том числе на работающем в эфире приёмнике. Код возврата 0 — вы уже в актуальном состоянии, 10 — обновление ждёт; так задание cron сможет сообщить вам, что появилась работа.

Шаг 2 — применить

```
./update.sh --apply
```

Три вида файлов обрабатываются по-разному, и в этом различии весь смысл:

Файлы	Что происходит
config.toml, config-<приёмник>.toml, markers.json, admin_config.json, autorun.json, frequencylist/, chat_history.txt, frontend/variant.json, frontend/site_information.json, журналы, build/, frontend/dist/	Никогда не трогаются и даже не появляются в вопросах. Именно они делают машину <i>вашим</i> приёмником.
start-*.sh, stop-websdr.sh, юниты *.service, install*.sh, recompile.sh, setup_admin.sh, smeter_theme.sh, proxy.py, admin_server.py, thermal_guard.py, websdr_relay.py, frontend/src/bands-config.js, примеры конфигураций config.example*.toml	Спрашивается всегда , потому что это файлы, которые оператор имеет основания изменить.
Всё остальное	Обновляется после того, как копия старого файла сохранена в архиве резервной копии (см. <i>Отмена обновления</i>).

Для каждого файла средней группы показываются различия и предлагаются три варианта:

```
start-rx888mk2.sh [K]eep mine / [u]pstream / [b]oth (ENTER = Keep mine)
```

- **Keep mine** — ваш файл остаётся ровно таким, как есть.
- **upstream** — устанавливается новая версия, ваша предварительно сохраняется.
- **both** — новая версия кладётся рядом с вашей под именем start-rx888mk2.sh.new, чтобы вы перенесли свои правки, когда будет удобно.

Каким будет первый запуск. В первый раз нет записи о том, из какой версии происходят ваши файлы, поэтому вам предъявляется каждый файл средней группы — около десяти вопросов. Отвечайте так:

Ваш случай	Ответ
Вы никогда не правили этот файл	u — взять новую версию. Обычный случай.
Вы его правили (свои RX888_ARGS, привязка к ядрам, изменённый юнит)	b — ваш остаётся, новый кладётся рядом как <файл>.new.
Вы не уверены	ENTER — ваш остаётся, ничего не теряется, сравните потом.

Ваша конфигурация в этом никогда не участвует: вопросы касаются только сценариев и юнитов служб.

update.sh записывает в .update-state/ версию каждого установленного файла. Поэтому уже со второго запуска он отличает файл, изменённый **вами**, от файла, который просто устарел, и останавливается с вопросом только по тем, которых вы действительно касались.

Прежде чем что-либо записать, он останавливает приёмник, панель администратора, обратный прокси и ретранслятор разнесённого приёма WebSDR **того каталога, который обновляет**: то, что обслуживает другой каталог, перечисляется и оставляется работать, так что второй клон можно обновлять, пока первый остаётся в эфире. По завершении он запускает обратно ровно то, что остановил. Если изменились исходные файлы или файлы frontend, он предложит запустить для вас recompile.sh. Ничего никогда не удаляется: файлы, исчезнувшие из репозитория, просто сообщаются и удаляются только по вашему требованию — ключом --prune.

Отмена обновления

```
./update.sh --restore LAST
```

Каждый файл, который перезаписывает обновление, сначала сохраняется — в одном датированном архиве на каждый запуск, в обычной видимой папке update-backups/:

```
update-backups/phantomsdr-backup-20260923-164530.zip
```

Архив содержит только файлы, заменённые этим запуском, каждый по своему пути в вашем дереве, поэтому его можно скопировать на другую машину одним файлом, открыть где угодно или распаковать прямо поверх установки. Если zip не установлен, он записывается как .tar.gz. Хранятся три последних. Любая из этих команд возвращает то, что заменило обновление:

```
./update.sh --restore LAST
./update.sh --restore phantomsdr-backup-20260923-164530.zip
```

В каждом архиве также есть собственный restore.sh на случай, если проблема в самом update.sh: распакуйте архив и передайте ему каталог вашей установки, например ./restore.sh /home/pi/PhantomSDR-Plus. Во время восстановления update.sh запускается из копии вне дерева, потому что обычно и сам входит в число возвращаемых файлов.

Резервные копии, сделанные до 23 сентября 2026 года, лежат в скрытой папке .update-backups/<отметка времени>; они восстанавливаются так же и входят в число трёх хранимых.

Другие возможности

```
./update.sh --apply --yes      # никогда не спрашивает; всё изменённое вами СОХРАНЯЕТСЯ
./update.sh --ref v5.0.0      # метка, ветка или коммит вместо текущего дерева
./update.sh --list-excludes    # печатает правила «не трогать» так, как они действуют здесь
./update.sh --verbose          # перечисляет все файлы, а не только первые 40
```

Свои правила «не трогать» добавляются по одному шаблону в строке в файле `update-exclude.txt` в корневой папке установки.

Если сборка не проходит на очень старой установке

`update.sh` обновляет файлы, а не системные пакеты. Если ваше дерево настолько старое, что сборке теперь нужны библиотеки, которых у вас нет, `recompile.sh` остановится с ошибкой компилятора или `meson`. Это не сломанное обновление — не хватает зависимостей:

```
./install.sh
```

Установщик обновляется тем же запуском, и ваша конфигурация переживает и его.

Обновление вручную

Если вы предпочитаете применять набор файлов сами:

```
cd ~
unzip -o update.zip
cd PhantomSDR-Plus
./stop-websdr.sh
cp -a ~/update/. .
chmod +x *.sh
./recompile.sh
./start-rx888mk2.sh          # или сценарий запуска вашего приёмника
```

Сначала сохраните копию своей конфигурации — `config-*.toml`, `frontend/site_information.json`, `admin_config.json` и `markers.json`, — потому что набор файлов не может отличить ваши правки от правок выпуска. Именно эту задачу и решает `update.sh`.

Резервное копирование и восстановление

Файлы для резервной копии

- Файлы конфигурации: `*.toml`
- Сведения о площадке: `frontend/site_information.json`
- Маркеры: `markers.json`
- Пользовательские скрипты: `start-*.sh`, `stop-*.sh`
- История чата: `chat_history.txt`
- Фоновое изображение: `frontend/src/assets/background.jpg`

Команда резервного копирования

```
cd ~/PhantomSDR-Plus
tar -czf phantomsdr-backup-$(date +%Y%m%d).tar.gz \
  *.toml \
  *.sh \
  markers.json \
  chat_history.txt \
  frontend/site_information.json \
  frontend/src/assets/background.jpg
```

Команда восстановления

```
tar -xzf phantomsdr-backup-YYYYMMDD.tar.gz
```

Вопросы безопасности

Настройка межсетевого экрана

```
# Allow only necessary ports
sudo ufw allow 9002/tcp
sudo ufw enable
```

Обратный прокси (необязательно)

Рассмотрите использование nginx или Apache в качестве обратного прокси для:

- шифрования SSL/TLS
- привязки доменного имени
- балансировки нагрузки
- контроля доступа

Ограничение числа пользователей

Отредактируйте `config.toml`:

```
[server]
max_users = 100 # Limit concurrent users
```

Получение помощи

Ресурсы

- **Документация:** это руководство, README.md, USER_GUIDE.md
- **GitHub Issues:** <https://github.com/sv1btl/PhantomSDR-Plus/issues>
- **Живая демонстрация:** <http://phantomsdr.no-ip.org:8900/>

Сообщения о проблемах

Сообщая о проблеме, укажите:

1. операционную систему и её версию
2. модель устройства SDR
3. содержимое файла конфигурации
4. сообщения об ошибках
5. потребление системных ресурсов (ЦП, ОЗУ, GPU)

Поддержка сообщества

- Прежде чем создавать новые issue, просмотрите существующие на GitHub
- Приводите подробные сведения о своей конфигурации
- Прилагайте журналы и сообщения об ошибках

- Будьте терпеливы и уважительны

Приложение А: полный список зависимостей

Список пакетов Ubuntu 24.04

```
build-essential
cmake
pkg-config
meson
libfftw3-dev
libwebsocketpp-dev
libflac++.dev
zlib1g-dev
libzstd-dev
libboost-all-dev
libopus-dev
libliquid-dev
git
util-linux (taskset – for the autorun spot reporter)
psmisc
wget
curl
rtl-sdr (for RTL-SDR)
airspy (for Airspy)
hackrf (for HackRF)
libclfft-dev (for OpenCL)
ocl-icd-opencl-dev (for OpenCL)
clinfo (for OpenCL)
```

Приложение Б: примеры конфигурации

Пример 1: RTL-SDR для УКВ

```
[input]
sps = 2048000
frequency = 145000000
signal = "iq"

[input.driver]
format = "u8"

[input.defaults]
frequency = 145500000
modulation = "FM"
```

Пример 2: RX-888 mk2 для КВ (0-30 МГц)

Приёмник, который использует большинство сисопов. Это полный раздел `[input]`, а не фрагмент, со значениями из `config-rx888mk2.toml`, входящего в репозиторий.

```
[input]
sps = 60000000          # 0-30 МГц прямой оцифровкой
fft_size = 4194304      # см. примечание ниже
fft_threads = 8
brightness_offset = -9  # отрицательнее, если на водопаде чёрные области
frequency = 0           # основная полоса: RX-888 оцифровывает от постоянного тока
signal = "real"         # не "iq" - прямая оцифровка даёт вещественный поток
accelerator = "opencl"  # "none", если среды OpenCL нет
audio_sps = 12000
audio_compression = "flac"
waterfall_size = 1024
waterfall_compression = "zstd"
smeter_offset = 5
analog_smeter_offset = 5

[input.driver]
name = "stdin"
format = "s16"

[input.defaults]
frequency = 7120000
modulation = "LSB"
```

Его питает `rx888_stream` — это делает за вас `start-rx888mk2.sh`:

```
rx888_stream -f /path/to/SDDC_FX3.img -s 60000000 -g 50 -a 0 -m low --pga -d -r -o - \
| build/spectrumserver --config config-rx888mk2.toml
```

О `fft_size`: при 60 MSPS правильный размер — 4194304. 8388608 удваивает разрешение водопада, а вместе с ним память и процессорную стоимость каждого преобразования; на большинстве машин более узкие бины оплачиваются потерянными кадрами. Начните с 4194304 и повышайте только если у сервера есть заметный запас.

Пример 3: HackRF для широкополосной ЧМ

```
[input]
sps = 10000000
frequency = 100900000
signal = "iq"

[input.driver]
format = "s8"

[input.defaults]
frequency = 100900000
modulation = "WBFM"
```

Установка завершена! Теперь у вас должен быть полностью работоспособный сервер PhantomSDR-Plus.

73 de SV1BTL & SV2AMK

ГЛАВА 3

Настройка панели администратора

ADMIN_PANEL_SETUP.md

Панель администратора PhantomSDR-Plus — руководство по настройке

⚠ **Замечание по безопасности:** держите эту панель администратора в домашней сети или за VPN. Публичное открытие административного порта не рекомендуется без надлежащего усиления аутентификации.

Обзор

Панель администратора состоит из двух служб Python:

Служба	Файл	Роль
Панель администратора	admin_server.py	Веб-приложение Flask, привязано к 127.0.0.1 (только внутренне)
Обратный прокси	proxu.py	Открывает и SDR, и панель администратора на одном публичном порту

Обе службы читают конфигурацию из `admin_config.json`, который создаёт `setup_admin.sh`. **Не правьте константы портов прямо в файлах Python** — все порты и IP-адрес узла SDR хранятся в `admin_config.json`.

Что даёт панель администратора

- **Панель мониторинга** — состояние сервера в реальном времени, ЦП/ОЗУ, основные процессы, свежий вывод журналов, терминал
- **Редактор конфигурации** — просмотр, правка и сохранение любого файла `.toml`, `.sh`, `.json`, `.h`, `.cpp` в вашей установке
- **Просмотр журналов** — слежение за любым журналом в реальном времени (сервер SDR, панель администратора, RADE, аварии, autorun, прокси)
- **Маркеры** — просмотр и правка частотных маркеров
- **История чата** — просмотр журнала чата WebSDR, полная очистка или удаление отдельных сообщений без перезапуска сервера
- **Сообщение на водопаде** — трансляция постоянного баннера всем подключённым пользователям, видимого на водопаде в реальном времени
- **Пользователи** — живой список подключённых слушателей с их частотой, режимом, длительностью и кнопкой Kick
- **Удаление сообщений чата** — кнопка Delete, немедленно убирающая из журнала именно это сообщение
- **Трансляция сообщений на водопад** — позволяет вывести постоянный текстовый баннер на водопад каждого подключённого пользователя

- **Spot Reporting** — запуск/остановка декодера autorun FT8/FT4/JS8/WSPR, выбор диапазонов/режимов и отправка спотов в PSK Reporter / wsprnet (по умолчанию выключено), с двумя счётчиками: плитки по декодерам за текущий запуск и итог за всё время рядом с каждым флажком диапазона/вида связи
- **Графики** — частота ЦП, загрузка ЦП, температура ЦП и подключённые пользователи за последние 15 минут / 1 час / 4 часа / 12 часов / 24 часа
- **Тепловая защита (Thermal Guard)** — останавливает сервер при перегреве процессора и запускает его снова, когда тот остынет; пороги выводятся из вашего собственного процессора. Работает с любым способом запуска и остановки и поставляется в режиме только записи, поэтому ничего не делает, пока вы её не включите
- **Настройки** — смена пароля администратора, базового каталога SDR, имени процесса, публичного порта

Требования

- PhantomSDR-Plus уже установлен и работает
- Python 3.8+

ПРИМЕЧАНИЕ

Обычно установщик делает это за вас. `./install.sh` — как и каждый из четырёх установщиков для дистрибутивов, оба установщика RADE и установщик сервера статистики — настраивает панель по умолчанию, предварительно ставя `pip` менеджером пакетов системы, если его нет. На этой странице описано, что происходит на том шаге и что делать, если вы его пропустили или хотите что-то изменить позже.

Библиотеки Python для панели (`flask`, `psutil`, `aiohttp`, `tomli-w`) устанавливает `setup_admin.sh`; если это не удастся, он выведет точную команду для ручного запуска.

Шаг 1 — Файлы

Поместите эти четыре файла в каталог PhantomSDR-Plus:

```
admin_server.py
manage_admin.sh
setup_admin.sh
proxy.py
```

Сделайте сценарии оболочки исполняемыми:

```
chmod +x setup_admin.sh manage_admin.sh
```

Шаг 2 — Запустите сценарий настройки

```
./setup_admin.sh
```

Сценарий:

1. Проверит, установлен ли Python 3

2. Убедится в наличии `admin_server.py` и `manage_admin.sh`
3. Запишет `127.0.0.1` как `sdr_host` в конфигурацию (см. [Настройка sdr_host](#))
4. Запросит три номера портов. Каждый запрос предлагает значение по умолчанию в скобках, которое принимается простым Enter, так что обычная настройка — это три нажатия клавиш:
5. **Порт spectrumserver** — порт, на котором слушает ваш сервер SDR (по умолчанию 8900 — именно он прописан в каждом `config-*.toml` репозитория)
6. **Внутренний порт панели администратора** — где локально привязывается `admin_server.py` (по умолчанию 3000)
7. **Публичный порт прокси** — единственный внешний порт, объединяющий SDR + администрирование (по умолчанию 8902)

Неверные ответы отклоняются, и вопрос задаётся снова, но лишь пять раз — после этого берётся значение по умолчанию. Запуск, ввод которого не является терминалом (конвейер, cron, без присмотра), сразу берёт значения по умолчанию, вместо того чтобы ждать ввода, которого не будет.

1. Установит `flask`, `psutil`, `aiohttp` и `tomli-w` через `pip`
2. Предоставит `ss` возможность `cap_net_admin` (запасной путь функции отключения пользователей, нужен только если панель работает без прокси)
3. Спросит, какой сценарий запускает, а какой останавливает приёмник — через них панель управляет сервером, и тепловая защита тоже
4. Спросит, как далеко тепловая защита процессора может действовать сама — `log`, `throttle`, `stop` или `stop+restart` — и предложит запустить `setup-cpufreq-perms.sh`, чтобы этап `throttle` работал без `root` (см. [THERMAL_GUARD.md](#))
5. Запишет `admin_config.json` со всеми настройками
6. Предложит установить два юнита `systemd` (панель + прокси) — запуск при загрузке и перезапуск после сбоя. **Рекомендуется и выбрано по умолчанию** (достаточно нажать Enter): защита от перегрева процессора работает только пока работает панель, поэтому без юнитов после перезагрузки машина остаётся без защиты. Шаг автоматически пропускается там, где `systemd` не является PID 1 (контейнеры, WSL1, OpenRC) или где нет `sudo`.

После настройки откройте браузер через прокси:

```
http://YOUR_SERVER_IP:<proxy_port>/admin
```

Пароль по умолчанию: `admin`

⚠ **Немедленно смените пароль** — при первом входе перейдите в Settings. Мастер первого запуска проведёт вас через выбор каталога SDR, имени процесса, публичного порта и нового пароля.

Шаг 3 — Запуск / остановка / перезапуск

Панель и прокси можно запускать двумя способами. **Выберите один и придерживайтесь его** — предупреждение в конце раздела объясняет, что будет, если их смешать.

Способ В (systemd) настоятельно рекомендуется везде, где systemd доступен. Защита от перегрева процессора работает внутри панели, поэтому панель, умершая в 03:00, или перезагрузившаяся машина оставляют процессор без защиты, пока кто-нибудь не запустит её вручную. `setup_admin.sh` теперь предлагает юниты по умолчанию. Способ А используйте там, где systemd нет — контейнеры, WSL1, OpenRC/sysvinit — или когда вы сознательно хотите присматривать за всем сами.

Способ А — `manage_admin.sh` (без root, ничего не устанавливается)

```
./manage_admin.sh start      # start admin panel and proxy
./manage_admin.sh stop       # stop both
./manage_admin.sh status     # show running status and PIDs
```

Команды **restart намеренно нет**. Раньше она убивала панель и тут же запускала собственную копию, что конфликтует с systemd на любой машине со Способом В. Чтобы перезапустить по Способу А, выполните `stop`, затем `start` — см. [Перезапуск панели](#) ниже.

Один сценарий управляет **обоими** процессами. Перед запуском прокси он проверяет, установлен ли `aiohttp`, и при отсутствии выводит понятную ошибку с командой исправления.

Ничего не запускается само: после перезагрузки или падения панели вы запускаете её вручную. Для приёмника, за которым вы следите, это приемлемо, но учтите, что тепловая защита процессора работает внутри панели: пока панель не работает, машину ничто не защищает (см. [THERMAL_GUARD.md](#)).

Способ В — юниты systemd (запуск при загрузке, перезапуск после сбоя)

В репозитории есть два готовых юнита, `phantomsdr-admin.service` и `phantomsdr-proxy.service`. Исправьте в обоих `User=` и пути, затем:

```
sudo cp phantomsdr-admin.service phantomsdr-proxy.service /etc/systemd/system/
sudo systemctl daemon-reload
./manage_admin.sh stop          # free the ports first
sudo systemctl enable --now phantomsdr-admin
sudo systemctl enable --now phantomsdr-proxy
```

Повседневные команды тогда становятся такими:

```
sudo systemctl start   phantomsdr-admin
sudo systemctl stop    phantomsdr-admin
sudo systemctl restart phantomsdr-admin
systemctl status       phantomsdr-admin
sudo journalctl -u phantomsdr-admin -f
```

Те же четыре с `phantomsdr-proxy` или обе сразу: `sudo systemctl restart phantomsdr-admin phantomsdr-proxy`.

`admin.log` и `proxy.log` работают точно так же, как раньше — юниты дописывают в те же файлы.

Одно действие стоит выполнить один раз: установить правило `tmpfiles` из `tmpfiles/phantomsdr-logs.conf` (сначала исправьте в нём пути и пользователя). systemd создаёт эти два журнала от `root` до перехода на `User=`, и тогда кнопка **Clear Logs** в панели не срабатывает с ошибкой `Permission denied` — см. ниже «Clear Logs сообщает Permission denied». `setup_admin.sh` устанавливает правило за вас.

Юнит панели содержит `SupplementaryGroups=cpubfreq`: именно это позволяет `etapn throttle` тепловой защиты снижать частоту процессора, не запуская панель от `root`. Сначала создайте эту группу с

помощью `./setup-cpufreq-perms.sh`, иначе `systemd` откажется запускать юнит; удалите строку, если этап `throttle` вам не нужен.

Возврат к способу A:

```
sudo systemctl disable --now phantomsdr-admin
sudo systemctl disable --now phantomsdr-proxy
```

⚠ **Не смешивайте эти два способа.** При включённых юнитах `./manage_admin.sh stop` `systemd` отменит через пять секунд, а `./manage_admin.sh start` даст вам вторую, неуправляемую копию, которая дерётся за тот же порт с копией `systemd`. (`restart` убран из скрипта именно поэтому; теперь он выводит ошибку и отправляет к `systemctl`.) `./manage_admin.sh status` только читает, поэтому остаётся полезным в обоих случаях.

Перезапуск панели

Далее в этом документе в нескольких местах сказано «перезапустите панель». Имеется в виду то, что соответствует вашему способу:

```
# Способ B — юниты systemd (обычная установка)
sudo systemctl restart phantomsdr-admin phantomsdr-proxy

# Способ A — юниты не установлены
./manage_admin.sh stop && ./manage_admin.sh start
```

Юнит поставляется с `KillMode=process`, и эта строка принципиальна. Если вы запускаете SDR из панели, приёмник является потомком юнита панели и наследует его `cgroup`. При стандартном для `systemd` значении `KillMode=control-group` перезапуск панели снял бы вместе с ней `spectrumserver`, его сторожевой таймер и демон `autorun` — предварительно застряв на все 90 секунд тайм-аута остановки. С `KillMode=process` `systemd` подаёт сигнал только самой панели, поэтому `sudo systemctl restart phantomsdr-admin` оставляет загруженный приёмник в эфире.

Обновление юнита, установленного до v4.1.0. В старых файлах юнита этой строки нет, и перезапуск панели её не добавит: `systemctl restart` заново запускает программу, но не меняет её конфигурацию. Правьте *установленную* копию — файл в репозитории лишь шаблон, `systemd` его никогда не читает:

```
sudo nano /etc/systemd/system/phantomsdr-admin.service
```

Добавьте `KillMode=process` в раздел `[Service]`, рядом с `Restart=always`. Затем заставьте `systemd` перечитать файл и проверьте:

```
sudo systemctl daemon-reload
systemctl show phantomsdr-admin -p KillMode
```

Вторая команда должна вывести `KillMode=process`. Если по-прежнему выводится `control-group`, `systemd` работает с кэшированной копией — `systemctl show phantomsdr-admin -p NeedDaemonReload` показывает `yes`, пока перезагрузка конфигурации не выполнена. Ведущие пробелы в файлах юнитов игнорируются, поэтому отступы значения не имеют.

Шаг 4 — Откройте порт в межсетевом экране

Откройте в межсетевом экране только **публичный порт прокси**. Выставлять наружу внутренний порт панели администратора не нужно:

```
sudo ufw allow <proxy_port>
```

Шаг 5 — Функция отключения пользователей

Страница Users показывает каждого подключённого слушателя. В каждой строке указаны IP пользователя, настроенная частота, режим и длительность подключения. Чтобы отключить пользователя, нажмите в его строке кнопку **Kick** — разрывается только это TCP-соединение. Все остальные слушатели остаются на связи и ничего не замечают.

Отключение мгновенное и точечное: сервер не перезапускается, звук ни у кого не прерывается, а отключённый пользователь может сразу переподключиться (функция нужна для устранения проблемных или зависших соединений, а не для блокировки).

Внутри отключение использует `ss -K dst <IP> dport <port>`, чтобы закрыть конкретный TCP-сокеты. Для этого требуется возможность Linux, которую сценарий настройки предоставляет автоматически. Если вы пропустили настройку или кнопка сообщает об ошибке, задайте её вручную:

```
sudo setcap cap_net_admin+ep $(which ss)
# Verify:
getcap $(which ss)    # should show: cap_net_admin=ep
```

Конфигурация — admin_config.json

Вся конфигурация времени выполнения находится в `admin_config.json` в каталоге PhantomSDR-Plus. `setup_admin.sh` создаёт исходный файл; последующие изменения можно вносить через страницу Settings или правкой файла напрямую.

Ключевые поля, записываемые `setup_admin.sh`:

Ключ	Описание
<code>port</code>	Внутренний порт панели администратора (сюда привязывается <code>admin_server.py</code>)
<code>public_port</code>	Порт spectrumserver (используется страницей Users для запроса <code>/users</code>)
<code>proxy_port</code>	Публичный порт, который слушает прокси
<code>sdr_host</code>	Узел, по которому прокси обращается к spectrumserver — <code>127.0.0.1</code> (см. ниже)
<code>password_hash</code>	Хеш SHA-256 пароля администратора
<code>sdr_base_dir</code>	Путь к вашей установке PhantomSDR-Plus
<code>sdr_process_name</code>	Имя отслеживаемого процесса (по умолчанию: <code>spectrumserver</code>)
<code>start_script / stop_script</code>	Скрипты, которыми панель запускает и останавливает SDR-сервер, — через них действует и тепловая защита

Несколько приёмников на одном компьютере? Установите `start_script` в `start-all.sh`. Скрипт остановки и так останавливает все приёмники, а Restart и тепловая защита используют скрипт запуска; с `start-all.sh` они возвращают все приёмники из `receivers.toml`, а не только основной. Страница Users, счётчик слушателей и отключение тогда тоже охватывают все приёмники. См. [Несколько приёмников](#).

Ключи тепловой защиты (все необязательны — панель записывает их при сохранении настроек Thermal Guard, а отсутствующие берут эти значения по умолчанию):

Ключ	По умолчанию	Описание
<code>thermal_enabled</code>	<code>true</code>	Общий выключатель защиты
<code>thermal_mode</code>	<code>"log"</code>	<code>log</code> (только запись, никаких действий) · <code>throttle</code> · <code>stop</code> · <code>stop+restart</code>
<code>thermal_warn</code>	<code>null</code>	Температура предупреждения °C; <code>null</code> = вывести из критического порога вашего процессора
<code>thermal_throttle</code>	<code>null</code>	Температура, при которой снижается верхний предел частоты; <code>null</code> = автоматически
<code>thermal_stop</code>	<code>null</code>	Температура, при которой сервер останавливается; <code>null</code> = автоматически
<code>thermal_resume</code>	<code>null</code>	Температура, ниже которой должен опуститься процессор для восстановления; <code>null</code> = автоматически
<code>thermal_sustain_s</code>	<code>60</code>	Секунд удержания температуры остановки до действия
<code>thermal_warn_sustain_s</code>	<code>30</code>	Секунд удержания температур предупреждения и throttle
<code>thermal_resume_s</code>	<code>300</code>	Секунд ниже температуры восстановления до снятия блокировки
<code>thermal_max_stops_hour</code>	<code>2</code>	Тепловых остановок в час до отключения автоматического перезапуска
<code>thermal_test_temp</code>	<code>null</code>	Фиктивная температура для проверки; <code>null</code> = использовать реальный датчик

Изменения вступают в силу со следующим замером — через пару секунд, без перезапуска.

Чтобы изменить порты после первоначальной настройки, отредактируйте `admin_config.json` и перезапустите:

```
sudo systemctl restart phantomsdr-admin phantomsdr-proxy # B
./manage_admin.sh stop && ./manage_admin.sh start # A
```

Настройка `sdr_host`

`proxy.py` подключается к `spectrumserver` по петлевому адресу — `sdr_host` равен `127.0.0.1` в `admin_config.json`, его записывает туда `setup_admin.sh`. Менять это обычно не требуется.

Указывайте реальный адрес только в том случае, если `spectrumserver` работает на **другой машине**, чем прокси. После этого выполните [Перезапуск панели](#).

Что изменилось: прежние версии требовали, чтобы `sdr_host` был локальным IP машины, потому что `spectrumserver` закрывал WebSocket-соединения, приходящие с петлевого адреса. Этот фильтр убран, поэтому локальные соединения работают нормально. Два следствия: открытие `http://localhost:<port>` на самой машине сервера теперь показывает полный интерфейс (раньше страница загружалась, но оставалась пустой), и прокси больше не ломается, когда DHCP выдаёт машине новый IP. Если вы обновляетесь и в вашем `admin_config.json` всё ещё стоит локальный IP, замените его на `127.0.0.1` или запустите `setup_admin.sh` заново.

проху.ру — что это и когда нужно

`проху.ру` размещает и сервер SDR, и панель администратора на **одном публичном порту**, маршрутизируя по префиксу пути:

Путь	Направляется в
<code>/admin*</code>	Панель администратора (<code>localhost:<port></code>)
Всё остальное	Spectrumserver (<code><sdr_host>:<public_port></code>)

Без прокси вам пришлось бы открывать два порта отдельно. С ним вы открываете только один.

Прокси также снимает ограничение размера сообщения WebSocket в 4 МБ (важно для больших кадров FFT с RX-888 при 60 MSPS), передаёт настоящие IP клиентов через заголовки `X-Forwarded-*` и поддерживает долгоживущие соединения через NAT с помощью 30-секундного heartbeat.

Изменение портов после настройки

Отредактируйте `admin_config.json` напрямую:

```
{
  "port":      3000,
  "public_port": 9001,
  "proxy_port": 9002,
  "sdr_host":   "127.0.0.1"
}
```

Затем перезапустите обе службы:

```
sudo systemctl restart phantomsdr-admin phantomsdr-proxy # B
./manage_admin.sh stop && ./manage_admin.sh start      # A
```

Если требуется, чтобы `admin_server.py` при запуске привязывался к другому порту (например, в службе `systemd`), это можно переопределить переменной окружения:

```
ADMIN_PORT=3000 python3 admin_server.py
```

По умолчанию `admin_server.py` привязывается только к `127.0.0.1`. Чтобы запустить его отдельно без прокси (разработка/тестирование), привяжите ко всем интерфейсам:

```
ADMIN_BIND=0.0.0.0 python3 admin_server.py
```

Полезные команды вручную

```
# Start admin panel manually (foreground)
python3 ~/PhantomSDR-Plus/admin_server.py

# Kill the admin panel
pkill -f admin_server.py

# Kill the proxy
pkill -f proxy.py

# Free a port that is stuck in use
sudo fuser -k 3000/tcp

# Check what is listening on a port
ss -tlnp | grep 3000
```

Файлы журналов

Файл	Содержимое
admin.log	Вывод панели администратора
proxy.log	Стартовый баннер прокси + по одной строке доступа на каждый проксированный запрос
logwebsdr.txt	Журнал скрипта запуска и сторожа — старт/остановка, драйвер приёмника, регистрация на websdr.org
spectrumserver.log	Вывод самого процесса приёмника — настройка FFT и OpenCL, подключения клиентов, ошибки. Самый крупный из этих файлов; он ротится, поэтому рядом может лежать spectrumserver.log.1
rade.log	Журнал sidcar-процесса RADE/FreeDV
autorun.log	Журнал демона декодера — споты, SNR, дрейф
crash.log	Отчёты о сбоях и тепловые события: трассировки [CRASH] / [TERMINATE], заметки [EXIT] и строки [THERMAL] от защиты. Обычно пуст — всё, что в нём есть, стоит прочитать

Всё это доступно для чтения на вкладках **Log Viewer** панели (LOGWEBSDR, SPECTRUMSERVER, ADMIN, RADE, CRASH, AUTORUN, PROXY). Просмотрщик показывает последние 150 строк выбранной вкладки и обновляется каждые 3 секунды при включённом автообновлении; он читает только конец файла, поэтому его размер не важен.

Ротация proxy.log и admin.log

proxy.log записывает каждый проксированный запрос, а admin.log — каждый запрос, который доходит до самой панели. По умолчанию не ротится ни один из них. Раньше опросы панели преобладали в обоих файлах — /admin/api/status выполняется каждые ~3 секунды, пока открыта вкладка, — но теперь эти опросы отфильтровываются из обоих журналов (см. ниже), поэтому ротация ограничивает медленный рост, а не поток.

В репозитории есть конфигурация logrotate в logrotate/phantomsdr (ежедневно или раньше, если журнал превысит 10 МБ; хранит 7 сжатых архивов в logproxy/, чтобы они не засорили корень проекта). **Она зависит от площадки — откройте её и измените три пути к журналам, путь olddir и пользователя su под свою машину перед установкой, затем:**

```
sudo cp logrotate/phantomcdr /etc/logrotate.d/phantomcdr
sudo logrotate -d /etc/logrotate.d/phantomcdr      # dry run, should list all three logs
```

logrotate/phantomcdr в репозитории и /etc/logrotate.d/phantomcdr — две **независимые копии**; cp их не связывает. Правка файла в репозитории ничего не меняет на работающей системе, пока вы снова не выполните sudo cp выше. Logrotate читает только копию в /etc.

Конфигурация использует copytruncate, и это обязательно: manage_admin.sh запускает оба процесса с >> <log>, и ни один не переоткрывает свой стандартный вывод, поэтому ротация через переименование оставила бы их писать в перемещённый файл, а новый журнал навсегда остался бы пустым.

Архивы попадают в logproxy/ (создаётся автоматически благодаря createolddir) и называются proxy.log.1.gz ... proxy.log.7.gz, самый свежий первым. Прочитать один можно командой zcat logproxy/proxy.log.1.gz | less, а искать сразу во всех — zgrep "pattern" logproxy/*.gz.

Кнопка **Clear Logs** панели очищает **все** файлы из таблицы выше, включая autorun.log и proxy.log (CLEAR_EXCLUDE в admin_server.py пуст). Учтите, чего это стоит в случае autorun.log: его история декодирования — единственная запись о том, что и когда было принято, и восстановить её невозможно; после очистки остаётся лишь то, что logrotate уже перенёс в logproxy/, то есть теряется всё декодированное с момента последней ротации. Очищать его при работающем декодере безопасно: демон держит файл открытым с 0_APPEND и продолжает писать в тот же inode. Всплывающее сообщение называет, что очищено, а если файл не удалось записать — например, proxy.log, принадлежащий другому пользователю при отдельных учётных записях, — оно называет этот файл и ошибку вместо сообщения об успехе и оставляет панель этого журнала на экране, а не очищает её. Файлы обнуляются на месте и никогда не удаляются: и manage_admin.sh, и юниты systemd открывают их с 0_APPEND, поэтому удалённый файл продолжал бы наполнять невидимый inode до перезапуска службы.

Успешные опросы отфильтровываются из **обоих** журналов доступа — фильтром QuietPollFilter в proxy.py и таким же в admin_server.py. Панель обращается к /admin/api/status каждые 3 секунды, а к /admin/api/logs, /admin/api/autorun/status, /admin/api/users, /admin/api/graph-stats и /admin/api/chat — каждые 5 секунд; без фильтрации это ~95 % обоих файлов. Отбрасываются только обычные 200 по этим путям: любой другой статус, путь или метод записывается как прежде, поэтому неудачный опрос остаётся виден, а изменяющие состояние действия (kick, autorun/start, ...) идут по отдельным путям и записываются всегда. Единственное намеренное исключение — /admin/api/logs/clear, отфильтрованный в обоих файлах: его собственная строка доступа пишется *после* очистки, поэтому без фильтра каждая успешная очистка оставляла одну свежую строку в только что очищенном журнале, и кнопка выглядела неработающей.

Поскольку proxy.log — журнал доступа, он содержит IP-адреса посетителей и строки user-agent; учитывайте это, прежде чем делиться им при обращении за помощью.

«Clear Logs» сообщает Permission denied

На машине, где юниты systemd были установлены на чистое дерево, кнопка может вернуть:

```
x Could not clear admin.log ([Errno 13] Permission denied: /home/you/PhantomSDR-Plus/admin.log),
proxy.log ([Errno 13] Permission denied: /home/you/PhantomSDR-Plus/proxy.log)
```

Это всегда те же два файла и никогда остальные пять, потому что только их создаёт сам systemd. StandardOutput=append: открывается из PID 1 **до** перехода на User=, поэтому ещё не существующий журнал создаётся как root:root 0644. Панель работает от вашего пользователя и не может его очистить. Остальные пять журналов создают сценарии запуска, которые и так работают от вашего пользователя, поэтому они очищаются нормально. Там, где эти два файла

появились раньше юнитов — их создал `manage_admin.sh` через `>>` — владелец уже правильный и кнопка работает; именно поэтому проблема проявляется только на свежей установке, сразу начатой с `systemd`.

Проверьте командой `ls -l admin.log proxy.log`: у сбойных будет `root root`. Исправьте их:

```
sudo chown "$(id -un):$(id -gn)" admin.log proxy.log
sudo chmod 664 admin.log proxy.log
```

Перезапуск не нужен — `systemd` продолжает писать через уже открытый файловый дескриптор, а дозапись не меняет владельца. Нажмите **Clear Logs** ещё раз, и оба файла очистятся.

Не удаляйте файлы вместо этого. Оба открыты с `O_APPEND`; удаление оставит `systemd` заполнять невидимый inode до перезапуска службы, а вновь появившийся файл снова будет принадлежать `root`.

Чтобы это не повторялось, установите правило `tmpfiles`, которое поставляется в репозитории: `tmpfiles/phantomsdr-logs.conf`. Оно заново создаёт оба журнала с правильным владельцем при каждой загрузке, до запуска юнитов, так что удалённый журнал — ручная уборка, переустановка, перенесённый каталог установки — никогда не вернётся принадлежащим `root`.

Правило зависит от машины: откройте его и сначала замените два пути и `user:group` на свои, затем:

```
sudo cp tmpfiles/phantomsdr-logs.conf /etc/tmpfiles.d/99-phantomsdr-logs.conf
sudo systemd-tmpfiles --create /etc/tmpfiles.d/99-phantomsdr-logs.conf
```

Как и с `logrotate/phantomsdr`, файл в репозитории и копия в `/etc` независимы — правка репозиторной копии ничего не меняет, пока вы не выполните `cp` заново. В `systemd 254` и новее правило также исправляет владельца и права уже существующего журнала, то есть устраняет и текущий сбой; в более старых версиях кнопку разблокирует именно `chown` выше.

`setup_admin.sh` делает всё это за вас при установке юнитов, поэтому установка через сценарий с этим не сталкивается. Шаги выше — для пары, установленной вручную, или для установки, сделанной раньше, чем в сценарии появился шаг с владельцем журналов.

`Logrotate` не возвращает проблему: поставляемая конфигурация использует `copytruncate`, который сохраняет тот же inode и, следовательно, того же владельца.

Сводка доступа

Конфигурация	SDR	Панель администратора
Без прокси	<code>http://YOUR_IP:<public_port></code>	<code>http://YOUR_IP:<port>/admin</code>
С прокси	<code>http://YOUR_IP:<proxy_port></code>	<code>http://YOUR_IP:<proxy_port>/admin</code>

Удаление сообщений чата

Страница Chat History перечисляет каждое сообщение текущего журнала чата. У каждой записи есть кнопка **Delete**, которая немедленно убирает именно это сообщение из журнала — без перезапуска сервера.

Как это работает:

- Панель администратора переписывает файл журнала чата на месте, удаляя только строку выбранного сообщения.
- Изменение вступает в силу для любого пользователя, который перезагрузит чат; уже загруженная история в открытых вкладках задним числом не обновляется.
- Очистка всего журнала (кнопка **Clear All**) обнуляет файл, что также действует без перезапуска.
- **Ответ в чате** хранится как обычное сообщение, начинающееся с метки, которая называет сообщение, на которое он отвечает, например [RE:2026-09-25 12:00:01|SV1BTL] , поэтому в этом списке он показан с этой меткой впереди. Удаление ответа убирает только этот ответ. Удаление сообщения, на которое отвечают, оставляет его ответы на месте — они показывают только имя и время, на которые отвечают.

Кнопка удаления одинаково отображается во всех пяти вариантах приложения Svelte. Если удаление, похоже, не срабатывает, убедитесь, что у панели администратора есть право записи в файл журнала чата:

```
ls -l ~/PhantomSDR-Plus/chat.jsonl # path depends on your config
```

Трансляция сообщений на водопад

Панель **Waterfall Message** позволяет вывести постоянный текстовый баннер на водопад каждого подключённого пользователя, не трогая процесс сервера.

Отправка сообщения

1. Откройте панель администратора и перейдите в **Waterfall Message**.
2. Введите текст сообщения и выберите цвет (шестнадцатеричный, например #ffdd00).
3. Нажмите **Send** — сообщение сразу появится во всех активных видах водопада.

Очистка сообщения

Нажмите **Clear**, чтобы убрать баннер со всех водопадов. Состояние сообщения хранится в памяти панели администратора; оно очищается автоматически при перезапуске процесса панели.

Как это работает

Панель администратора предоставляет две внутренние конечные точки, которые пересылает прокси:

Конечная точка	Метод	Назначение
/admin/api/waterfall-message	POST	Задать или очистить текст и цвет текущего баннера
/admin/api/waterfall-message	GET	Вернуть текущее состояние сообщения в виде JSON

Веб-интерфейс опрашивает состояние сообщения и отрисовывает его наложением на холст водопада. На стороне клиента не требуется ни перепоключение WebSocket, ни перезагрузка страницы.

Типичные применения

- Объявление о плановом обслуживании: "Server restart in 10 minutes"

- Сообщение об условиях прохождения: "Solar flux 180 – 10m wide open"
- Приветствие: "Welcome to SV1BTL WebSDR – Athens, KM17"

Системные графики

Страница **Графики** отображает четыре величины на общей оси времени:

- **Частота ЦП** — средняя частота по всем ядрам, в ГГц
- **Загрузка ЦП** — общая загрузка, в процентах
- **Температура ЦП** — в °C, с пунктирными отметками на 70 °C (тепло) и 80 °C (критично)
- **Пользователи онлайн** — слушатели, подключённые к WebSDR в этот момент

Диапазон выбирается кнопками **15 MIN / 1 HOUR / 4 HOURS / 12 HOURS / 24 HOURS**. При наведении на график появляется перекрестие со всеми четырьмя значениями на этот момент. Четыре карточки над графиком всегда показывают самый свежий отсчёт.

Как работает выборка

Фоновый поток в `admin_server.py` берёт один отсчёт каждые **2 секунды**. Отсчёты хранятся в двух уровнях: последний **час** в полном разрешении 2 секунды (1800 точек) и **24 часа** тридцатисекундных средних (2880 точек) для длинных диапазонов. Страница выбирает уровень, покрывающий выбранный диапазон, и пишет `30s averages` в заголовке, когда показывает усреднённый уровень.

Целые сутки с разрешением 2 секунды — это ~43000 точек: десятки МБ памяти, первая передача в несколько мегабайт и около 40 точек на пиксель, чего не может отобразить ни один экран. Усреднение до 30 секунд уместит сутки в 2880 точек.

Выборка запускается вместе с панелью администратора, а не при первом открытии страницы, поэтому страница открывается на уже накопленной истории.

Буфер находится **только в памяти**: на диск ничего не пишется, а история теряется при перезапуске панели. Это сделано намеренно — так функция остаётся вне ротации журналов и не занимает диск.

2 секунды дают почти живую картину и достаточно коротки, чтобы не искажать частоту ЦП, которая на современном процессоре от отсчёта к отсчёту прыгает между частотой простоя и турбо. Один отсчёт стоит около миллисекунды, поэтому выборка ничтожна на фоне SDR-сервера. Страница опрашивает в ритме своего уровня — каждые 2 секунды на живых диапазонах и каждые 30 секунд на усреднённых, поскольку более частые запросы возвращают лишь пустые ответы.

Чтобы изменить период или глубину хранения, отредактируйте эти константы в начале блока выборки в `admin_server.py` и перезапустите:

```
GRAPH_INTERVAL_S    = 2          # секунд между точными отсчётами
GRAPH_FINE_S        = 3600       # глубина точного уровня (1 ч)
GRAPH_COARSE_EVERY  = 15         # точных отсчётов на укрупнённую точку (15 x 2 с = 30 с)
GRAPH_COARSE_S      = 24 * 3600  # глубина укрупнённого уровня (24 ч)
```

Откуда берутся значения

Величина	Источник
Частота ЦП	<code>psutil.cpu_freq()</code> , иначе <code>/sys/devices/system/cpu/cpu*/cpufreq/</code> . Это те же данные ядра, которые форматирует <code>cpufreq-info</code> , поэтому cpufrequtils не требуется .
Загрузка ЦП	<code>psutil.cpu_percent()</code>
Температура ЦП	<code>psutil.sensors_temperatures()</code> , иначе <code>/sys/class/thermal</code> , затем команда <code>sensors</code>
Пользователи онлайн	Собственная конечная точка <code>/users</code> у <code>spectrumserver</code> — достоверный список сессий. Подсчёт сокетов через <code>ss</code> не видит реальные IP клиентов за <code>proxu.ru</code> .

Если величина недоступна на вашей машине, соответствующая панель сообщает об этом, а не рисует нулевую линию. Неудавшееся измерение разрывает линию, чтобы пропуск никогда не выглядел как измеренное значение.

Почему четыре панели, а не один график

ГГц, проценты, градусы и число людей не имеют общей шкалы. Для одного графика понадобилось бы несколько осей Y, а возникающие при этом пересечения — артефакт масштабирования, а не факт о сервере. Четыре панели на одной оси времени сохраняют сравнение честным. Каждая панель начинается с нуля, поэтому высота кривой остаётся пропорциональной значению.

Цвет зарезервирован за температурой, где янтарный и красный отмечают указанные пороги. Остальные панели используют один цвет, поскольку каждая содержит один ряд, который уже назван её заголовком.

Конечная точка

`GET /admin/api/graph-stats?since=<epoch>` (требуется вход) возвращает отсчёты новее `since`, поэтому страница опрашивает инкрементально, а не скачивает весь буфер на каждом цикле.

Thermal Guard

Полное руководство: [THERMAL_GUARD.md](#) — четыре режима и что должен сделать сисоп в каждом, работа без панели, этап `throttle` без `root`, проверка и устранение неполадок.

Останавливает SDR-сервер, когда процессор становится опасно горячим, и запускает его снова, когда тот остынет. Она входит в состав панели — устанавливать ничего не нужно — и появляется карточкой **THERMAL GUARD** на Панели управления, а её настройки находятся в разделе **Настройки**.

Поставляется **бездействующей**: режим начинается с `log`, поэтому изначально она лишь записывает, что *сделала бы*. Ничто не трогает ваш сервер, пока вы не смените режим.

Работает с любым способом запуска и остановки

Защита никогда не пытается выяснить, что присматривает за вашим сервером. Она действует через скрипты, заданные как **Default start script** и **Default stop script** в Настройках, и пока процессор перегрет, повторяет остановку при **каждой проверке** (каждые 2 с). Всё, что возвращает сервер к жизни — `watchdog` внутри `start-*.sh`, юнит `systemd`, задание `cron`, сеанс `tmux` — отменяется за пару секунд, пока машина не остынет. Это и есть **блокировка** (lockout).

Если скрипт остановки не задан, она посылает SIGTERM процессу, названному в `sdr_process_name`, а через 10 секунд отсрочки — SIGKILL.

Пороги берутся из вашего собственного процессора

Вместо фиксированного числа, которое было бы неверным на большинстве машин, защита читает критический порог, публикуемый ядром для вашего процессора, и отсчитывает от него назад:

Ваш процессор сообщает	предупреждение	throttle	остановка	восстановление
Intel, предел 100 °C	88	92	95	75
AMD Ryzen, предел 95 °C	83	87	90	70
Raspberry Pi, предел 85 °C	73	77	80	60

Это важно: фиксированные 95 °C почти нормальны для Intel, но на Ryzen — где Tctl по замыслу держится на 95 °C под бустом — они остановили бы сервер на совершенно здоровой машине, а на Pi, который снижает частоту уже при 80 °C, не сработали бы никогда. Любой порог можно задать вручную в Настройках; оставьте поле пустым для *автоматического* значения.

Чтобы увидеть, каким защита видит ваш компьютер:

```
python3 thermal_guard.py --once
```

```
sensor      : hwmon:coretemp
temperature: 58.0 C
crit        : 100.0 C
mode        : log
thresholds  : warn 88.0  throttle 92.0  stop 95.0  resume 75.0 (C)
cpufreq     : not writable — throttle stage disabled
```

Четыре ступени

Остановка сервера — крайняя мера, а не единственный инструмент:

1. **warn** — записывается в журнал, и только.
2. **throttle** — снижает верхний предел частоты процессора на 20 %. Требуется драйвер `cpufreq` и право записи в `scaling_max_freq`, то есть обычно `root`; там, где этого нет (обычный случай для панели без привилегий и для большинства VPS), ступень **отключает себя сама**, и защита переходит от `warn` сразу к `stop`. `./setup-cpufreq-perms.sh` даёт это право записи группе `cpufreq`, так что ступень работает без `root` — см. [THERMAL_GUARD.md](#).
3. **stop** — выполняет ваш скрипт остановки и удерживает сервер выключенным (см. блокировку).
4. **resume** — после того как процессор `thermal_resume_s` (по умолчанию 5 минут) держится ниже температуры восстановления, предел частоты возвращается, а в режиме `stop+restart` снова выполняется ваш скрипт запуска.

Каждая ступень должна **удерживаться** своё время — 30 с для `warn` и `throttle`, 60 с для `stop` — так что короткий всплеск во время компиляции никогда её не запустит.

Повторные срабатывания ограничены параметром `thermal_max_stops_hour` (по умолчанию 2). По достижении предела *автоматический перезапуск* отключается, а защита продолжает работать, чтобы перегревающаяся машина не металась между работой и остановкой.

Какие датчики используются

Только настоящие датчики на кристалле процессора: `coretemp` (Intel), `k10temp` / `zenpower` (AMD), `cpu_thermal` / `soc_thermal` (ARM, Raspberry Pi). Берётся самое высокое показание чипа, а не среднее.

`ascpitz` и безымянные тепловые зоны **намеренно игнорируются** — они часто сообщают температуру корпуса или платы на десятки градусов ниже реальной температуры процессора, поэтому защита, доверяющая им, не сработала бы тогда, когда это важно. Если на вашей машине нет пригодного датчика (обычное дело на VPS или внутри контейнера), карточка прямо сообщает об этом (`no trusted CPU sensor`), и защита остаётся неактивной, а не делает вид, что защищает вас.

Что она записывает

Всё попадает в `crash.log`, то есть во вкладку **CRASH** просмотрщика журналов, по одному событию на строку:

```
[THERMAL] warn: 89.0C (warn=88.0 crit=100.0) sustained 31s
[THERMAL] throttle: 93.0C sustained 30s — cpufreq max lowered 20%
[THERMAL] STOPPED server: 96.0C sustained 62s (stop=95.0 crit=100.0) — Started stop-websdr.sh
[THERMAL] lockout: process reappeared at 94.0C — re-stopped [3 time(s) so far]
[THERMAL] recovered: 74.0C held below 75.0 — lockout cleared
[THERMAL] auto-restart: Started start-rx88mk2.sh
```

Строка блокировки записывается не чаще раза в минуту, чтобы упорно сопротивляющийся надзиратель не затопил журнал.

Проверьте, прежде чем доверять

Задайте **TEST TEMPERATURE** в Настройках выше вашего порога остановки. Защита воспримет это как настоящее показание, и весь путь — `warn`, `throttle`, `stop`, блокировка, восстановление, перезапуск — пройдёт по вашему желанию. Очистите поле, чтобы вернуться к реальному датчику.

Если провести эту проверку в режиме `stop`, сервер действительно остановится и ваши слушатели будут отключены. Делайте это, когда никого нет, или оставьте режим `log`, где проверка показывает, что *произошло бы*, ничего не трогая.

Как её лучше внедрить

1. Оставьте режим `log` на неделю и работайте как обычно.
2. Читайте вкладку CRASH после тяжёлых моментов — полной пересборки, жаркого дня. Если там ничего нет, ваша машина даже близко не подходила.
3. Если записанные температуры выглядят правильно, переключите режим на `stop` (или `stop+restart`, чтобы сервер возвращался сам).
4. Подтвердите работу один раз полем тестовой температуры и оставьте её в покое.

Снятие блокировки

Кнопка **CLEAR LOCKOUT** на карточке Панели управления снимает блокировку и ограничение частоты срабатываний. Она не разоружает защиту: если процессор всё ещё слишком горяч, следующая проверка просто снова остановит сервер. Пользуйтесь ею после того, как исправили охлаждение.

Endpoint

GET /admin/api/thermal (требуется вход) возвращает полное состояние защиты — датчик, пороги, ступень, блокировку, последнее событие. POST с {"action": "reset"} делает то же, что и кнопка CLEAR LOCKOUT.

Без панели администратора

thermal_guard.py использует только стандартную библиотеку и работает самостоятельно, читая тот же admin_config.json:

```
python3 thermal_guard.py --once           # вывод датчика, критического порога и порогов
python3 thermal_guard.py                 # запуск как службы (только запись, пока не взведена)
python3 thermal_guard.py --mode stop+restart # взвести без единого файла конфигурации
python3 thermal_guard.py --config /path/to/other.json
python3 thermal_guard.py --help
```

--mode имеет приоритет над thermal_mode из файла конфигурации, так что машину без панели можно взвести прямо из её systemd-юнита, вообще без JSON.

Spot Reporting (autorun FT8/FT4/JS8/WSPR)

Вкладка **Spot Reporting** управляет службой autorun (autorun/index.js) — процессом Node.js, который декодирует FT8/FT4/JS8/WSPR на стороне сервера прямо с приёмника и выгружает споты в сети отчётности:

- **FT8 / FT4 / JS8 → PSK Reporter** (IPFIX/UDP)
- **WSPR → wsprnet.org** (HTTP)

JS8 сообщается только на скорости Normal — 15-секундном вызывном цикле, где идут heartbeat и CQ. Споты берутся из heartbeat, compound-кадров и направленных сообщений; групповые адресаты вроде @ALLCALL и позывные, которые декодер не смог распознать (<...>), не сообщаются никогда. JS8 использует ту же очередь PSK Reporter, что FT8 и FT4, и считается отдельно, точно так же, как они.

Отчётность по умолчанию ВЫКЛЮЧЕНА. Ничего не выгружается, пока вы не включите приёмник отчётов и не нажмёте **Start**. Декодирование и выгрузка независимы: служба может декодировать и вести журнал при выключенной отчётности, так что вы можете проверить активность, прежде чем что-либо станет публичным.

△ Споты выгружаются в публичные сети под **вашим позывным**. Включайте только диапазоны и режимы, которые ваш приёмник действительно слышит, и указывайте правильный локатор.

Предварительные требования

Службе autorun нужны Node.js 22+, пакеты npm ws + cbor-x (разрешаются через символическую ссылку autorun/node_modules → frontend/node_modules) и util-linux (taskset). Установщики настраивают всё это автоматически — см. [раздел Autorun Spot Reporter в INSTALLATION.md](#). Если **Start** не срабатывает, обычная причина — эта ссылка или taskset (см. «Устранение неполадок» ниже).

Работа с вкладкой

1. **Идентификация** — позывной и локатор. Предзаполняются из `frontend/site_information.json` (`siteSysop` / `siteGridSquare`, усечены до 6 символов); при необходимости переопределите здесь.
2. **Матрица диапазонов × режим** — отметьте комбинации для декодирования. Строки — диапазоны, столбцы — FT8 / FT4 / JS8 / WSPR; неподдерживаемые ячейки отключены. WSPR дополнительно охватывает диапазоны НЧ/СЧ (2200 м, 630 м) и дополнительный европейский канал 80 м.
3. **Приёмники отчётов** — включите **PSK Reporter** и/или **wsprnet** (оба по умолчанию выключены).
4. **Max slots** — предохранительное ограничение на число одновременно работающих слотов диапазон/режим.
5. **Start / Stop** — запускает/завершает службу (привязанную к старшим ядрам ЦП через `taskset`, выбираемым автоматически для машины — см. *Требования к ресурсам и ограничения* ниже). **Save / Reload** сохраняют и пересчитывают конфигурацию; **Free All** очищает все слоты.

Карточка состояния обновляется каждые 5 с и показывает состояние работы, счётчики декодирования/выгрузки и время последней отправки. Значок **REPORTING** появляется на главном водопаде, пока отчётность активна.

«0 sent» в первые минуты — это нормально. PSK Reporter выгружает пакетами каждые 5 минут, а wsprnet — каждые 2 минуты; споты ждут очередной отправки. Карточка состояния показывает число ожидающих и периодичность.

Смена диапазонов или режимов во время работы

Изменения не вступят в силу до перезапуска демона. `autorun/index.js` читает `autorun.json` ровно один раз, при запуске. Ни слежения за файлом, ни сигнала перезагрузки нет — обрабатываются только SIGINT и SIGTERM, оба означают завершение. Поэтому работающий демон продолжает декодировать те слоты, с которыми был запущен, что бы вы ни сохранили после.

Чтобы применить изменение:

1. Отметьте / снимите отметки с диапазонов и режимов
2. **SAVE CONFIG**
3. **STOP**
4. **START**

Или проще: **STOP** → изменить флажки → **START**, поскольку START сохраняет конфигурацию перед запуском.

⚠ **Нажатие START без предварительного STOP изменение не применяет.** START сохраняет конфигурацию, затем запрос на запуск отвечает `already running`, и ничего не перезапускается. Вы увидите сообщение «сохранено» рядом с ошибкой «already running», пока демон продолжает работать со **старыми** диапазонами — это легко принять за успех. Всегда сначала STOP.

Что перезапуск делает со счётчиками:

- **Итоги рядом с каждым флажком сохраняются** — они в `autorun-totals.json`, который при завершении не удаляется.
- **Плитки по декодерам обнуляются**, они считаются за один запуск.
- **Ничего из очереди не теряется**: при завершении ожидающие споты отправляются.

- **Только что включённый** диапазон начинает с 0; диапазон, использовавшийся **ранее**, продолжает прежний итог.
- Диапазон, с которого вы **сняли отметку, сохраняет своё число** рядом с теперь пустым флажком; история не удаляется.

Декодирование прерывается лишь на несколько секунд между STOP и START.

Счётчики спотов

Показаны два разных счёта, отвечающие на разные вопросы.

SPOTS UPLOADED PER DECODER — плитка на каждый декодер (FT8 / FT4 / JS8 / WSPR) со спотами, выгруженными **с последнего запуска демона**, а под ней число декодирований и сколько ещё в очереди. Stop/Start обнуляет их. Декодер, у которого назначение выключено, показывает свои декодирования и `reporting off` вместо голого 0, потому что ноль выгрузок здесь — настройка, а не сбой.

Число рядом с каждым флажком в BANDS & MODES — это спотов выгружено **за всё время** для этой комбинации диапазона и режима. Они хранятся в `autorun-totals.json` и переживают перезапуски. **Янтарный цвет с точкой впереди** (· 123) означает, что слот декодировал, но пока ничего не выгрузил. Между отправками это нормально — PSK Reporter выгружает каждые 5 минут, wsprnet каждые 2 — поэтому счётчики FT8/FT4 первые минуты после запуска стоят янтарными. Янтарным он остаётся и если назначение этого декодера выключено. Простой 0 без точки означает, что слот включён, но пока ничего не декодировал: WSPR стоит на нуле несколько минут после запуска, поскольку работает в двухминутном цикле, тогда как FT8 уже считает сотнями. Слот, который демон никогда не запускал, не показывает ничего. При наведении видны выгрузки, декодирования и время последней выгрузки.

FT8 и FT4 используют одну очередь отправки в PSK Reporter, поэтому демон относит каждый спот к своему режиму и диапазону в момент отправки; разбивка не оценивается задним числом из итогов.

Обнуление счётчиков. FREE ALL SLOTS снимает отметки со всех диапазонов и режимов, выключает оба назначения, останавливает демон и **стирает итоговые счётчики** — после этого рядом с флажками не показывается ни одного числа. Это единственный способ их обнулить, и отменить его нельзя. Кнопка останавливает демон и ждёт его завершения, прежде чем удалить `autorun-totals.json`, поскольку демон перезаписывает этот файл при завершении; удаление при работающем демоне просто вернуло бы числа.

Требования к ресурсам и ограничения

Служба намеренно нетребовательна и работает на слабом оборудовании (четырёхъядерного i5 достаточно), но есть реальные ограничения, о которых стоит знать.

Привязка autorun к ядрам учитывает конфигурацию и выполняется автоматически.

Служба autorun всегда запускается панелью администратора (`admin_server.py`), поэтому её привязка задаётся при любой установке без настройки: из числа процессоров выводится диапазон ядер для `taskset`, резервирующий несколько старших ядер под декодирование, либо на ≤ 4 ядрах служба работает без привязки.

Привязка spectrumserver зависит от вашего способа запуска. То, как вы запускаете `spectrumserver`, различается от установки к установке (тип SDR, инструменты, личные сценарии), поэтому этот документ не предполагает какого-либо конкретного лаунчера. Панель администратора не запускает и не привязывает `spectrumserver` — это целиком зависит от команды или службы, которой вы его запускаете. Если вы хотите держать `spectrumserver` вне ядер, используемых службой `autorun`, привяжите его сами через `taskset` в своей команде запуска (см. раздел о переопределении ниже). Если нет, он просто работает без привязки, а планировщик

ОС балансирует нагрузку — это верно и безопасно, вы лишь теряете намеренное разделение ядер.

Для справки, служба `autorun` резервирует эти старшие ядра (поэтому, если вы привязываете `spectrumserver`, оставьте его на младших ядрах во избежание пересечения):

Логических ЦП	Использует служба <code>autorun</code>	Оставить для <code>spectrumserver</code>
≤ 4	без привязки	без привязки
6	5	0-4
8	6-7	0-5
12 (например, гибрид 8P+4E)	8-11	0-7
16	12-15	0-11

На машине с **4 ядрами (или меньше)** разделять нечего, поэтому служба `autorun` тоже работает *без привязки* и делит все ядра — это верно и безопасно, но FFT приёмника и всплески декодирования конкурируют за одни и те же ядра.

Известные ограничения:

- Настоящее узкое место — `spectrumserver` и полоса SDR, а не служба.** RX888 на 30 МГц требует OpenCL/GPU; процессор с малым числом ядер без подходящей видеокарты не потянет такой FFT. Сочетайте слабое оборудование с более узкополосным SDR (RSP1A $\approx$ 10 МГц, RTL-SDR $\approx$ 2.4 МГц).
- WSPR — главный потребитель ЦП.** Его декодер Фано — порт на JS (~30 с на диапазон, однопоточный). Пул из 4 воркеров выполняет 4 декодирования параллельно; включение **более ~4 диапазонов WSPR** одновременно может вывести очередь за пределы 120-секундного интервала, особенно когда SDR конкурирует за ядра. Декодирование FT8/FT4 по сравнению с этим дёшево.
- Масштабирование числа слотов.** Оценка ~2-2,5 ядра / 600-700 МБ относится к полным ~28 слотам на восьмиядерной машине. На 4 общих ядрах ограничьтесь несколькими диапазонами (ориентир: ≤ 6 FT8/FT4 + ≤ 3 WSPR) и следите за показателем `queued` пула.
- Привязка предполагает гибридную топологию Intel** (младшие ядра — более быстрые P-ядра). На AMD или процессорах с чередующейся нумерацией SMT автоматическое разделение остаётся корректным (без пересечений и сбоев), но утверждение «младшие ядра быстрее» может не выполняться буквально; на процессоре 4C/8T с `hyperthreading` старшие ядра — это SMT-близнецы, то есть ограничение есть, но полной изоляции нет. Там, где автоматический диапазон неидеален, используйте **ручное переопределение** ниже.
- Покрывание диапазонов ограничено приёмником.** Конфигурация RX888 (`sps=600000000` → Найквист 30 МГц) не достаёт до 6 м и выше; таблица диапазонов — от 160 м до 10 м. Другие SDR покрывают то, что позволяет их окно настройки.
- Периодичность отчётности.** PSK Reporter отправляет каждые 5 мин, `wsprnet` — каждые 2 мин; «0 sent» в первые минуты нормально, это не неисправность.

Ручное переопределение привязки к ядрам (для опытных)

Автоматически выводимое разделение ядер подходит большинству машин, но там, где это не так (AMD CCX/CCD, ARM big.LITTLE, чередующийся SMT), можно задать привязку явно. Два независимых переопределения, каждое принимает список ядер в формате `taskset` -с (2-3, 0,2,4, 0-2,5) либо `none/off/unpinned` для отключения привязки. Некорректное значение игнорируется и применяется автоматический вывод — опечатка никогда не помешает запуску.

Служба autorun — переменная окружения `AUTORUN_CORES` или поле `"cores"` в `autorun.json` (окружение приоритетнее). Поле `"cores"` сохраняется при **Save** в панели администратора, поэтому ручная правка не теряется. Два способа задать его:

Вариант А — `autorun.json` (постоянный, рекомендуемый). Добавьте строку `"cores"` в конфигурацию в корне репозитория, затем выполните **Stop → Start** на вкладке Spot Reporting:

```
// autorun.json
{ "identity": { "callsign": "SV1BTL", "grid": "KM17VX" },
  "reporting": { "pskreporter": true, "wsprnet": false },
  "slots": [ /* ... */ ],
  "cores": "2-3" }           // or "none" to run unpinned
```

Вариант В — переменная окружения (разово). Службу запускает панель администратора, поэтому переменная должна быть в окружении **панели** — задайте её и перезапустите панель (окружение приоритетнее значения в `autorun.json`):

```
# Способ А — переменная должна быть у запускающего процесса:
./manage_admin.sh stop && AUTORUN_CORES=2-3 ./manage_admin.sh start

# Способ В — systemd игнорирует переменную, заданную в командной строке
# systemctl, поэтому пропишите её в юните:
sudo systemctl set-environment AUTORUN_CORES=2-3
sudo systemctl restart phantomsdr-admin
```

spectrumserver — привяжите его сами, добавив `taskset -c <ядра>` перед той командой, которой вы запускаете сервер. Это работает при любом способе запуска (прямой вызов, конвейер SDR, юнит службы и т. п.):

```
# pin the server to cores 0-3, direct launch:
taskset -c 0-3 ./build/spectrumserver --config <your-config>.toml &lt; <your-input>
# or in an SDR pipeline:
<your-sdr-source> | taskset -c 0-3 ./build/spectrumserver --config <your-config>.toml
```

⚠ **Сделайте так, чтобы это пережило перезапуски.** Встроенный `taskset` действует только для одного конкретного запуска. Если что-то перезапускает сервер автоматически (сторожевой процесс, служба `systemd`, запись `cron/@reboot`, ...), добавьте префикс `taskset` внутрь того сценария или юнита, который его действительно запускает, иначе при перезапуске привязка потеряется.

Проверьте, что привязка применилась — выведите фактическую привязку работающих процессов:

```
taskset -cp "$(pgrep -f 'autorun/index.js')" # autorun daemon
taskset -cp "$(pgrep -x spectrumserver)"    # spectrumserver
```

Допустимые значения (для обоих переопределений): список `taskset -c (2-3, 0,2,4, 0-2,5)` либо `none/off/unpinned` для отсутствия привязки. Всё некорректное игнорируется, и применяется автоматический вывод, поэтому опечатка никогда не заблокирует запуск.

На Raspberry Pi / любой машине с ≤ 4 ядрами обычно не нужно ни то, ни другое — автоматический путь и так запускает оба процесса без привязки, что для небольшого однородного процессора и есть правильный выбор. Переопределение предназначено для более крупных машин, не являющихся гибридами Intel.

Файлы и конечные точки

Элемент	Назначение
<code>autorun.json</code>	Сохранённая конфигурация (идентификация, слоты, приёмники отчётов, максимум слотов, необязательное переопределение <code>coges</code>). Записывается вкладки; игнорируется <code>git</code> .
<code>autorun-status.json</code>	Актуальное состояние, читаемое карточкой (<code>pid</code> , счётчики, последняя выгрузка). Записывается каждые 15 с; удаляется при остановке.
<code>frontend/dist/autorun-active.json</code>	Публичные данные значка, отдаваемые по <code>/autorun-active.json</code> . Пусто, когда отчётность выключена.
<code>autorun.log</code>	Стандартный вывод и ошибки службы.
<code>GET/POST /admin/api/autorun/config</code>	Чтение/запись <code>autorun.json</code> (проверяет позывной, локатор, комбинации диапазон/режим, предел слотов).
<code>GET /admin/api/autorun/status</code>	Состояние работы (<code>pgrep</code>) + <code>autorun-status.json</code> .
<code>POST /admin/api/autorun/start / stop</code>	Запуск (<code>taskset -c <auto> node autorun/index.js</code>) / <code>SIGTERM</code> . Диапазон ядер выводится из ЦП (см. выше).

Примечание: после обновления `admin_server.py` нужен [Перезапуск панели](#), чтобы загрузились новые маршруты `autorun` или обновлённая матрица диапазон/режим. Служба работает как отдельный процесс, поэтому переживает перезапуски панели администратора.

Устранение неполадок

Проблема	Решение
Панель администратора не запускается	Проверьте <code>admin.log</code> — обычно не хватает пакета Python
Прокси не запускается	Выполните <code>python3 -c "import aiohttp"</code> — при ошибке: <code>pip3 install aiohttp --break-system-packages</code>
<code>проху.log</code> пуст	Сначала проверьте, действительно ли прокси работает (<code>./manage_admin.sh status</code>). Если да, у вас старая версия: лаунчер должен использовать <code>python3 -u</code> (без буферизации — иначе стартовый баннер никогда не покинет 8-КБ буфер), а <code>main()</code> в <code>проху.py</code> должен вызывать <code>logging.basicConfig()</code> (иначе у журнала доступа <code>aiohttp</code> нет обработчика и каждая строка запроса теряется, поскольку <code>web.AppRunner</code> , в отличие от <code>web.run_app</code> , не настраивает журналирование).
Прокси запускается, но водопад пуст	Проверьте <code>sdr_host</code> в <code>admin_config.json</code> — обычно <code>127.0.0.1</code> . Если там остался старый локальный IP от прежней версии, исправьте его и выполните Перезапуск панели
Кнопка Kick сообщает «no connections»	Выполните <code>getcap \$(which ss)</code> — если нет <code>cap_net_admin</code> , выполните <code>sudo setcap cap_net_admin+ep \$(which ss)</code>
Состояние всегда OFFLINE	Перейдите в Settings → SDR Process Name — укажите точное имя, которое показывает <code>ps -eo comm,args grep -v grep</code>
Страница Users не показывает данных	Проверьте работу конечной точки: <code>curl http://127.0.0.1:<public_port>/users</code>
Внешний браузер получает NetworkError	Проверьте, работает ли прокси: <code>./manage_admin.sh status</code>
Прокси перестал работать после изменения сети	Актуально, только если в <code>sdr_host</code> всё ещё локальный IP — задайте <code>127.0.0.1</code> и выполните Перезапуск панели
Кнопка удаления чата не действует	Проверьте право записи в файл журнала чата: <code>ls -l ~/PhantomSDR-Plus/chat.jsonl</code>
Сообщение на водопаде не появляется	Убедитесь, что прокси работает (<code>./manage_admin.sh status</code>) и веб-интерфейс собран из свежей версии, включающей отрисовщик наложения
Сообщение на водопаде пропало после перезапуска	Так и задумано — состояние сообщения хранится только в памяти; отправьте его заново после перезапуска панели
«Start» в Spot Reporting не срабатывает / служба сразу завершается	Проверьте <code>autorun.log</code> . Обычно отсутствует символическая ссылка <code>autorun/node_modules</code> (<code>ln -sf ./frontend/node_modules autorun/node_modules</code>) или не установлен <code>taskset</code> (<code>sudo apt install -y util-linux</code>).
Spot Reporting: служба работает, но <code>decodes</code> остаётся 0, а в <code>autorun.log</code> видно <code>504 / tap closed 1006</code>	Аудиоотвод не может достучаться до <code>spectrumserver</code> . Служба определяет порт автоматически по конфигурации работающего сервера; строка журнала <code>[autorun] tap backend: HOST:PORT</code> должна совпадать с вашим <code>[server] port</code> . Если она неверна (или сервер не работал при запуске), задайте порт явно в <code>autorun.json</code> : <code>"server": { "host": "127.0.0.1", "port": 9002 }</code> , затем Stop→Start.
Spot Reporting показывает «0 sent»	Нормально в первые минуты — PSK Reporter отправляет каждые 5 мин, <code>wspnet</code> — каждые 2 мин. Убедитесь, что включён приёмник отчётов и служба работает.
Новые диапазоны/режимы не появляются в матрице	Перезапуск панели — матрица загружается при старте.

Проблема	Решение
В <code>autorun.log</code> видно <code>MODULE_TYPELESS_PACKAGE_JSON / «Reparsing as ES module ... performance overhead»</code>	Косметическое предупреждение (декодирование работает). В <code>frontend/src/modules/package.json</code> отсутствует <code>"type": "module"</code> ; добавьте эту строку ближе к началу, затем Stop→Start. Пересборка веб-интерфейса не нужна — служба импортирует исходный файл напрямую.
«Start» в Spot Reporting завершается с <code>taskset: ... Invalid argument</code> на старом/слабом ПК	С учитывающим конфигурацию лаунчером такого быть не должно — диапазон ядер выводится из числа процессоров и на ≤ 4 ядрах привязка не применяется. Если это произошло, ваш <code>admin_server.py</code> старше этого изменения; обновите его (<code>_autorun_taskset_prefix</code>) и выполните Перезапуск панели .
Значок REPORTING не появляется на водопаде	Отчётность должна быть включена хотя бы с одним слотом; значок читает <code>/autorun-active.json</code> . Пересоберите веб-интерфейс, если <code>dist/index.html</code> старше значка.

Thermal Guard

THERMAL_GUARD.md

Thermal Guard — руководство сисопа

Защита от перегрева процессора для PhantomSDR-Plus.

WebSDR — одна из немногих задач, которая занимает все ядра, целый день, бесконечно. Если засорится вентилятор, встанет помпа или в августе нагреется помещение, вам об этом никто не скажет: сервер просто продолжит декодировать, пока процессор не начнёт вариться. Thermal guard — это та часть, которая замечает это и выдёргивает вилку за вас.

Он поставляется **неактивным**: из коробки он только пишет в журнал. Ничего не защищено, пока вы не выберете режим. Это руководство — о том, как сделать такой выбор и жить с ним.

Некогда? Выполните `python3 thermal_guard.py --once`, прочитайте выведенные пороги и поставьте `thermal_mode` в `stop+restart` в панели администратора. Это вся работа. Всё остальное ниже — это *почему*.

Содержание

1. Что он делает
2. Быстрый старт
3. Как он принимает решение
4. Четыре режима — что делает каждый и что должны сделать вы
5. Выбор режима
6. Справочник настроек
7. Работа без панели администратора
8. Включение этапа throttle без root
9. Работа без присмотра — когда никто не смотрит
10. Проверьте его, прежде чем доверять
11. Чтение журнала
12. Lockout и ограничение частоты остановок
13. Устранение неполадок
14. Отключение или удаление

1. Что он делает

Каждые 2 секунды страж считывает температуру кристалла процессора и сравнивает её с четырьмя порогами. Когда температура достаточно долго держится выше одного из них, он поднимается на ступень выше:

норма	→	warn	→	throttle	→	stop	→	(остывание)	→	restart
		строка в		снижение		запуск		сервер		запуск
		журнале		частоты		stop-скрипта		остаётся выкл.		start-скрипта

Два проектных решения делают его пригодным для машины *любого* сисопа, и их стоит понять, прежде чем на него полагаться:

Он никогда не спрашивает, как вы запускаете сервер. systemd, цикл watchdog, cron, tmux, обычная оболочка — страж этого не знает, и ему это не нужно. Вместо этого, пока машина перегрета, он держит **lockout**: он повторяет команду остановки на *каждом* такте. Если ваш watchdog перезапустит сервер, страж остановит его снова в течение двух секунд. Что бы ни пыталось удержать сервер живым — оно проигрывает, пока процессор не остынет. Именно это делает его универсальным, и это проверено против настоящего watchdog.

Он никогда не зашивает температуру в код. «Останавливать при 95 °C» — плохое правило: на Intel с Tjmax 100 это почти обычная нагрузка, на Ryzen это *ниже* того значения, на котором Tctl держится под бустом по замыслу, а на Raspberry Pi этого не достичь, потому что SoC сам сбрасывает частоту при 80. Поэтому страж считывает критический порог, который публикует ваше собственное ядро (`tempN_crit`), и отсчитывает от него назад. Один и тот же код, разумные значения везде.

2. Быстрый старт

1. Посмотрите, какой ваша машина выглядит для стража. Всегда сначала это:

```
cd ~/PhantomSDR-Plus
python3 thermal_guard.py --once
```

```
sensor      : hwmon:coretemp
temperature: 65.0 C
crit        : 100.0 C
mode        : log
thresholds  : warn 88.0  throttle 92.0  stop 95.0  resume 75.0 (C)
cpufreq     : not writable – throttle stage disabled
```

Если в `sensor` стоит `NONE`, остановитесь здесь и прочитайте [Устранение неполадок](#): страж не может защитить машину, которую не может измерить, и он говорит вам об этом, а не притворяется.

2. Оставьте его на несколько дней в режиме log. Он уже работает внутри панели администратора. Дайте ему понаблюдать обычную неделю — летние дни, крупную сборку, оживлённую субботу с соревнованиями — и посмотрите, сработал бы он вообще.

3. Введите его. Панель администратора → **Settings** → **THERMAL GUARD** → **MODE** → `stop+restart` → **SAVE THERMAL SETTINGS**. Вступает в силу примерно за 2 секунды; перезапуск не нужен.

3. Как он принимает решение

Датчик

Страж доверяет только настоящим датчикам на кристалле процессора:

Источник	Типичное оборудование
coretemp	Intel
k10temp, zenpower, zenpower3	AMD
cpu_thermal, soc_thermal	Raspberry Pi, платы ARM
Тепловая зона x86_pkg_temp	запасной вариант на x86

asptz и тепловые зоны без метки исключены **намеренно**. На обычном настольном компьютере asptz показывает около 28 °C, когда пакет процессора нагрет до 59 °C: страж, доверяющий такому датчику, попросту никогда бы не сработал. Сообщить «отключён, нет доверенного датчика» честнее, чем изображать защиту.

psutil используется, если он случайно установлен, но никогда не требуется. Страж обходится одной стандартной библиотекой.

Пороги

Каждый — это отступ ниже критического порога, который машина публикует сама:

Порог	Отступ ниже crit	Что означает
warn	−12 °C	Что-то не так, записать
throttle	−8 °C	Попробовать охладить, снизив частоту
stop	−5 °C	Слишком близко к пределу — остановить сервер
resume	−25 °C	Действительно снова холодно, возвращаться безопасно

На практике получается так:

Ваш процессор сообщает	warn	throttle	stop	resume
Intel, crit 100 °C	88	92	95	75
AMD Ryzen, crit 95 °C	83	87	90	70
Raspberry Pi, crit 85 °C	73	77	80	60

Если машина вообще не публикует критическую точку, страж переходит на консервативный фиксированный набор и сообщает об этом. Любой порог можно переопределить абсолютным значением (см. [Справочник настроек](#)), но необходимости в этом быть не должно.

Время удержания — почему всплеск никогда не срабатывает

Одно горячее показание ни к чему не приводит. Порог должен удерживаться **непрерывно**:

- **warn** и **throttle**: 30 с (thermal_warn_sustain_s)
- **stop**: 60 с (thermal_sustain_s)
- **resume**: 300 с ниже порога возврата перед перезапуском (thermal_resume_s)

Всплеск при компиляции, всплеск декодирования, раскручивающийся вентилятор — всё это слишком коротко, чтобы иметь значение. Одно показание ниже порога сбрасывает счётчик. Кроме того, после запуска страж требует 3 подряд идущих корректных измерения, прежде чем вообще что-то предпринять, и отбрасывает любое значение вне диапазона 20–125 °C как сбой датчика, а не как температуру.

4. Четыре режима

`thermal_mode` — единственный переключатель, определяющий, как высоко стражу позволено подняться по лестнице. Каждый режим **включает всё, что делают предыдущие**.

Режим	Предупреждает	Снижает частоту	Останавливает	Перезапускает	Может ли сервер упасть?
<code>log</code>	✓	—	—	—	Нет — ничего не трогает
<code>throttle</code>	✓	✓	—	—	Нет
<code>stop</code>	✓	✓	✓	—	Да, останется выключенным до вашего вмешательства
<code>stop+restart</code>	✓	✓	✓	✓	Да, возвращается сам

`log` — наблюдать и сообщать (заводская настройка)

Что делает: всё то же, что и остальные режимы, кроме собственно действий. Он проходит всю лестницу и записывает в `crash.log`, что он *сделал бы*. Он никогда не запускает ваш `stop`-скрипт, не трогает частоту процессора и не прерывает ни одного слушателя.

Что нужно сделать вам: ничего — он уже работает. Через неделю откройте вкладку **CRASH** в панели или выполните:

```
grep THERMAL crash.log
```

- **Ничего нет?** С охлаждением всё в порядке, и стража можно спокойно взводить.
- **STOP level reached ... NOT stopping?** Страж выключил бы ваш сервер. Это настоящая тепловая проблема, с которой вы работали — исправьте охлаждение и взведите стража.

Используйте, когда: вы только что установили стража и ещё не знаете, как ведёт себя ваша машина. Это режим обкатки, а не конечная точка — страж в режиме `log` не защищает ничего.

`throttle` — снизить частоту, ни разу не прервав работу

Что делает: на пороге `throttle` снижает максимальную частоту процессора на 20 %, а затем восстанавливает её, как только температура вернётся ниже линии предупреждения. Ваш WebSDR всё это время остаётся в эфире. Слушатели под высокой нагрузкой могут заметить чуть худшую производительность; большинство не заметит ничего.

Если температура всё равно продолжит расти и превысит порог остановки, этот режим **запишет это и не сделает ничего** — сервер он не остановит.

Что нужно сделать вам:

1. Дать стражу право записи в ограничение частоты процессора — оно ему нужно, а по умолчанию есть только у `root`. Выполните один раз `./setup-cpufreq-perms.sh`; см. [раздел 8](#).
2. Убедиться через `python3 thermal_guard.py --once`, что выводится `cpufreq: writable, throttle stage available`. Если написано *not writable*, этот режим не делает ничего сверх того, что делает `log`, и будет сообщать об этом в журнале каждый раз.

Используйте, когда: вы разбираетесь с пограничной проблемой охлаждения и хотите защитить машину, ни разу не потеряв слушателей. **Учтите, что это не полная защита** — если снижения частоты не хватит, больше ничего не произойдёт. Хорош как первый шаг, плох как окончательный ответ.

stop — выключить сервер и оставить выключенным

Что делает: на пороге остановки запускает ваш stop-скрипт, а затем удерживает **lockout** — повторно останавливает сервер каждые 2 секунды, пока машина перегрета, чтобы ничто не подняло его за вашей спиной. Когда процессор продержится 5 минут ниже порога возврата, lockout снимается. **Сервер сам не возвращается.** Вы запускаете его, когда убедитесь, что причина устранена.

Что нужно сделать вам:

1. Убедиться, что страж знает, как останавливать ваш сервер. В панели это настроенный stop-скрипт (`./stop-websdr.sh` для входящих в комплект пусковых скриптов). Если он не задан, страж переходит к SIGTERM по имени процесса, а затем к SIGKILL после 10 секунд отсрочки — это работает, но нормальный stop-скрипт чище.
2. Принять, что **ваш WebSDR будет офлайн, пока вы этого не заметите.** Обеспечьте себе способ узнать: мониторинг доступности, вкладка CRASH или просто ежедневная проверка.

Используйте, когда: машина важнее сервиса, вы обычно рядом, или тепловое событие достаточно серьёзно, чтобы осмотреть машину, прежде чем она продолжит работу. Также правильный выбор, если вы не доверяете автоматическому перезапуску на своём оборудовании.

stop+restart — выключить, остыть и вернуться самому

Что делает: всё то же, что и `stop`, плюс: как только процессор продержится 5 минут ниже порога возврата, запускается ваш start-скрипт и WebSDR возвращается сам.

От «дребезга» защищает `thermal_max_stops_hour` (по умолчанию 2). После двух тепловых остановок в течение часа страж **отключает автоматический перезапуск** и оставляет сервер выключенным, чтобы вы разобрались. Обратите внимание на важную деталь: ограничение отключает *перезапуск*, но никогда — *защиту*: lockout продолжает останавливать сервер, пока тот горячий, сколько бы раз он уже ни срабатывал.

Что нужно сделать вам:

1. Убедиться, что **оба** скрипта настроены и действительно работают сами по себе — и start, и stop. Проверьте их вручную: `./stop-websdr.sh`, затем `./start-rx888mk2.sh`.
2. Больше ничего. Это режим «поставил и забыл».

Используйте, когда: WebSDR публичный, без присмотра или удалённый — то есть в большинстве случаев. Это рекомендуемый режим для обычной установки.

5. Выбор режима

Ваша ситуация	Режим
Страж только установлен, машина ещё не изучена	log на неделю
Публичный WebSDR без присмотра, должен справляться сам	stop+restart
Удалённая площадка, куда трудно добраться	stop+restart
Хотите осматривать машину после каждого теплового события	stop
Оборудование, которому вы не доверяете автоперезапуск	stop
Пограничное охлаждение, терять слушателей нельзя	throttle, затем перейти на stop+restart
Машина без доверенного датчика процессора	ни один не сработает — сначала разберитесь с датчиком

Для большинства сисопов ответ — stop+restart, а честное резюме остальных: log не защищает ничего, а throttle защищает немного.

6. Справочник настроек

Все ключи находятся в admin_config.json рядом с thermal_guard.py, и каждый из них также редактируется в **Settings** → **THERMAL GUARD**. Страж **перечитывает файл на каждом такте**, поэтому изменения вступают в силу за ~2 секунды — без перезапуска.

Ключ	По умолчанию	Значение
thermal_enabled	true	Главный выключатель. false полностью отключает стража.
thermal_mode	"log"	log throttle stop stop+restart — см. раздел 4 .
thermal_warn	null	Абсолютный порог предупреждения в °C. null = вывести из критического порога.
thermal_throttle	null	Абсолютный порог throttle. null = выводится.
thermal_stop	null	Абсолютный порог остановки. null = выводится.
thermal_resume	null	Абсолютный порог возврата. null = выводится.
thermal_sustain_s	60	Секунд удержания порога остановки перед остановкой. Минимум 5.
thermal_warn_sustain_s	30	Секунд для этапов предупреждения и throttle. Минимум 5.
thermal_resume_s	300	Секунд ниже порога возврата до снятия lockout. Минимум 30.
thermal_max_stops_hour	2	Тепловых остановок в час, после которых отключается автоматический перезапуск. Защита продолжает работать.
thermal_test_temp	null	Делает вид, что процессор имеет эту температуру. Только для проверки — см. раздел 10 .

Автономный страж дополнительно использует start_script, stop_script, sdr_process_name и sdr_base_dir из того же файла.

Переопределение порога. Делайте это только при наличии причины. Частая: ваша машина законно нагревается при сборках, и вы не хотите, чтобы это считалось. Поднятие `thermal_stop` даёт запас за счёт расстояния до критической точки — никогда не ставьте его выше своего значения `crit`, иначе процессор первым дойдёт до собственного аварийного отключения, а страж превратится в украшение.

7. Работа без панели администратора

Многие сисопы никогда не ставят панель. Страж прекрасно работает сам по себе — тот же файл, то же поведение.

Сначала попробуйте его на переднем плане:

```
cd ~/PhantomSDR-Plus
python3 thermal_guard.py --once          # что бы здесь произошло?
python3 thermal_guard.py --mode log      # смотреть вживую, Ctrl-C для выхода
```

`--mode` переопределяет `thermal_mode` из командной строки, так что писать JSON вручную не придётся:

```
python3 thermal_guard.py --mode stop+restart
python3 thermal_guard.py --config /etc/phantomsdr/thermal.json
```

Затем сделайте это постоянным с помощью примера юнита, поставляемого с проектом:

```
sudo cp thermal-guard.service /etc/systemd/system/
sudo nano /etc/systemd/system/thermal-guard.service # задайте User= и пути
sudo systemctl daemon-reload
sudo systemctl enable --now thermal-guard
systemctl status thermal-guard
```

Прочитайте комментарии в начале `thermal-guard.service`, прежде чем включать его. Важнее всего две вещи:

- **User= должен быть учётной записью, которая действительно может выполнить ваши start- и stop-скрипты.** Запускать стража от `root` обычно неверно: тогда он выполнял бы эти скрипты тоже от `root`.
- Либо укажите ему `admin_config.json`, содержащий `start_script` и `stop_script`, либо передайте `--mode` в строке `ExecStart`. Минимальной конфигурации достаточно:

```
{
  "sdr_base_dir": "/home/вашпользователь/PhantomSDR-Plus",
  "sdr_process_name": "spectrumserver",
  "start_script": "start-rx888mk2.sh",
  "stop_script": "stop-websdr.sh",
  "thermal_mode": "stop+restart"
}
```

Всё, что делает страж, по-прежнему попадает в `crash.log` рядом с этим файлом конфигурации, а также в журнал `systemd` (`journalctl -u thermal-guard`).

8. Включение этапа throttle без root

По умолчанию `python3 thermal_guard.py --once` сообщает:

```
cpufreq : not writable – throttle stage disabled
```

Это нормально и не является ошибкой. Ядро создаёт

```
/sys/devices/system/cpu/cpuN/cpufreq/scaling_max_freq
```

с владельцем `root:root` и правами `0644`. Панель администратора, работающая от обычного пользователя, писать туда не может, поэтому страж пропускает этап throttle и сообщает об этом в журнале, а не отказывает молча.

Неправильное решение — запускать стража от root. Он также выполняет ваши start- и stop-скрипты, а те должны оставаться непривилегированными.

Правильное решение — отдать право записи именно на эти файлы отдельной группе. Это делает один скрипт:

```
cd ~/PhantomSDR-Plus
./setup-cpufreq-perms.sh
```

Он один раз спросит ваш пароль — и только один раз, при настройке — а затем:

1. создаст системную группу `cpufreq` и добавит вас в неё;
2. установит `/etc/tmpfiles.d/99-phantomsdr-cpufreq.conf`, чтобы группа и права `0664` применялись заново **при каждой загрузке**: права в `sysfs` сами по себе перезагрузку не переживают;
3. применит изменение немедленно, чтобы вы могли проверить без перезагрузки.

Затем, в таком порядке:

```
# 1. выйдите из системы и войдите снова (или перезагрузитесь) – новая группа
#   доходит до ваших процессов только через новый вход в систему
id | grep cpufreq

# 2. перезапустите панель; страж проверяет право записи один раз, при старте
sudo systemctl restart phantomsdr-admin
#   (так переход не нужен – systemd заново строит список групп при каждом
#   запуске. Только если юниты НЕ установлены: ./manage_admin.sh stop && ./manage_admin.sh start
#   – никогда оба сразу, они борются за порт 3000)

# 3. подтвердите
python3 thermal_guard.py --once
#   cpufreq : writable, throttle stage available
```

Чтобы отменить всё полностью: `sudo ./setup-cpufreq-perms.sh --revoke`.

Примечания

- Этап throttle *действует* только если `thermal_mode` равен `throttle` или выше. Само по себе разрешение ничего не меняет.
- Страж снижает на 20 % то ограничение, которое действует *в данный момент*, и восстанавливает именно это значение. Если вы уже ограничиваете процессор (например, через `MAX_SPEED` в `/etc/init.d/cpufrequtils`), восстановится именно ваше ограничение — страж не вернёт вам исподтишка частоту выше запрошенной.

- На некоторых машинах драйвер `cpufreq` вообще не доступен — многие VPS, большинство контейнеров. Скрипт это распознаёт и ничего не меняет. Там используйте `stop` или `stop+restart`; они от `cpufreq` не зависят.

9. Работа без присмотра

Самый частый вопрос о страже: *что происходит, когда никто не смотрит?* Короткий ответ: именно для этого случая он и создавался.

Ему не нужен открытый браузер. Страж — это фоновый поток внутри `admin_server.py`, тактируемый сборщиком графиков каждые 2 секунды. Он работает независимо от того, вошёл ли кто-то в панель, открыт ли браузер и не спите ли вы. В автономном режиме это служба `systemd`, ещё более независимая. Карточка на панели — это *окно* к стражу, а не сам страж.

Он действует, не спрашивая. Нет ни диалога подтверждения, ни уведомления, которого нужно ждать. На пороге остановки он останавливает сервер, и всё. Ровно в этом и смысл.

Но только если вы ему это разрешили. В режиме `log` он честно запишет, как машина варится, и ничего с этим не сделает. Если вы читаете этот раздел потому, что «меня нет рядом, чтобы следить», нужный вам режим — `stop+restart`: только он и защищает машину, и возвращает сервис без вас.

Он помогает, только пока панель жива. Страж — часть `admin_server.py`: панель, упавшая в 02:00, уносит защиту с собой, а после перезагрузки машина остаётся без присмотра, пока вы не войдёте и не запустите её снова. Если никто не следит, пусть следит `systemd`: в репозитории есть `phantomsdr-admin.service` и `phantomsdr-proxy.service`, которые стартуют при загрузке и перезапускают панель в течение пяти секунд после сбоя. См. [ADMIN_PANEL_SETUP.md](#). Для приёмника без присмотра это **настоятельно рекомендуется**: включить `stop+restart` без этого — защита только до следующей перезагрузки.

Как вы узнаете об этом потом. Примерно по убыванию полезности:

Где	Что вы получаете
<code>crash.log</code> / вкладка CRASH	Каждое действие с меткой времени. Постоянная запись.
Карточка THERMAL GUARD на панели	Состояние вживую: температура, этап, <code>lockout</code> , остановки за последний час.
Страница Graphs (температура CPU, 24 ч)	Форма события — как быстро росла, сколько остывала.
<code>journalctl -u thermal-guard</code>	Только в автономном режиме.

Разумный порядок для площадки без присмотра: взведённый `stop+restart` и `grep THERMAL crash.log` всякий раз, когда вы всё равно заходите. Если пусто — узнавать нечего.

Разобранный пример. Отказ вентилятора в 03:00 на машине в режиме `stop+restart`:

03:14	температура переходит 88 °C, держится 30 с	→	записано предупреждение
03:16	переходит 92 °C, держится 30 с	→	применён <code>throttle</code> (если разрешён)
03:19	переходит 95 °C, держится 60 с	→	STOP – <code>stop</code> -скрипт, <code>lockout</code> включён
03:19	<code>watchdog</code> перезапускает сервер	→	страж снова его останавливает и пишет
03:26	процессор 5 минут ниже 75 °C	→	<code>lockout</code> снят, запуск <code>start</code> -скрипта
03:41	перегревается и останавливается второй раз	→	предел достигнут, перезапусков больше нет сервер остаётся выключен, защита остаётся

Вы просыпаетесь с выключенным WebSDR, с `crash.log`, который точно объясняет почему, и — что важнее всего — с процессором, который так и не приблизился к своей критической точке.

10. Проверьте его, прежде чем доверять

Не ждите настоящей жары, чтобы выяснить, работает ли ваш stop-скрипт.

`thermal_test_temp` заставляет стража считать, что процессор нагрет до выбранной вами температуры. Всё остальное ведёт себя совершенно обычно — времена удержания, лестница, lockout, ваши настоящие скрипты.

1. Панель → **Settings** → **THERMAL GUARD** → **TEST TEMPERATURE**.
2. Введите значение выше своего порога остановки (например, 97 на Intel).
3. Сохраните и наблюдайте за карточкой панели и вкладкой CRASH.
4. **Очистите поле, когда закончите.** Оставленная тестовая температура означает, что страж больше не читает настоящий датчик.

В режиме `log` это показывает вам всю лестницу, ничем не рискуя — безопасная первая проверка и та, которую стоит сделать до взведения чего-либо.

В режиме `stop+restart` это **проверка вживую**: ваш сервер действительно остановится и действительно вернётся примерно через пять минут после очистки поля. Делайте это, когда никто не слушает. Один раз это стоит сделать, потому что только так вы узнаете, что ваши stop- и start-скрипты работают, когда их вызывает не вы, а что-то другое.

Эквиваленты в автономном режиме:

```
python3 thermal_guard.py --once          # только пороги, без действий
python3 thermal_guard.py --mode log      # наблюдать лестницу вживую
```

11. Чтение журнала

Всё дописывается в `crash.log` рядом с файлом конфигурации, по одному событию на строку, с префиксом `[THERMAL]`:

```
grep THERMAL crash.log
```

Строки намеренно однострочные и удобные для grep. Что вы увидите:

Строка	Значение
warn: 88.4C (warn=88.0 crit=100) sustained 30s	Первая ступень. Ничего не сделано.
throttle: 92.1C sustained 30s – cpufreq max lowered 20%	Частота процессора снижена.
throttle level reached: ... (no cpufreq write access – stage skipped)	См. раздел 8 .
STOP level reached: ... mode is 'log', NOT stopping	Он остановил бы сервер. Взведите его.
STOPPED server: 95.2C sustained 60s (stop=95.0 crit=100) – ran stop-websdr.sh	Настоящий случай.
lockout: process reappeared at 96.0C – re-stopped (...) [12 time(s) so far]	Что-то перезапускает ваш сервер; страж побеждает. Не чаще одной строки в минуту.
rate limit: 2 thermal stops within the hour – automatic restart is now DISABLED	Остаётся выключенным, чтобы вы разобрались. Защита продолжается.
recovered: 54.0C held below 75.0 – lockout cleared	Снова холодно.
auto-restart: ran start-rx888mk2.sh	Снова в эфире.
back to normal: 59.0C (warn=88.0)	Сошёл с лестницы, так и не остановившись.

12. Lockout и ограничение частоты остановок

Именно эти два поведения удивляют чаще всего, так что скажем прямо.

Lockout — это то, что заставляет стража работать именно против *вашей* конфигурации. Останавливая сервер, он не просто отдаёт одну команду и надеется. Он помечает себя заблокированным и на каждом такте в 2 секунды, пока температура выше порога возврата, проверяет, не вернулся ли процесс — и если вернулся, останавливает его снова. Ваш watchdog, Restart=always в systemd, задание cron, нетерпеливый сисоп: все они проигрывают этот спор, пока процессор не остынет. Он снимается сам, как только температура продержится ниже порога возврата в течение thermal_resume_s (по умолчанию 5 минут).

Кнопка CLEAR LOCKOUT на карточке панели снимает lockout и счётчик остановок за час — для случая, когда вы починили охлаждение и не хотите ждать. Она **не** разоружает стража: если машина всё ещё горячая, следующая же проверка снова остановит сервер. Так и задумано.

Ограничение (thermal_max_stops_hour, по умолчанию 2) не даёт неисправной машине всю ночь то включаться, то выключаться. По достижении предела страж **отключает автоматический перезапуск** и оставляет сервер выключенным. Прочитайте внимательно: он отключает перезапуск, а не защиту. Lockout продолжает останавливать сервер, пока тот перегрет, сколько бы раз он уже ни срабатывал. Страж, который умолкал бы ровно тогда, когда машине хуже всего, был бы хуже, чем никакого стража.

13. Устранение неполадок

sensor : NONE – no trusted CPU sensor on this machine Страж не нашёл ни одного разрешённого датчика и отказывается гадать. Попробуйте sensors (из lm-sensors; sudo apt install lm-sensors && sudo sensors-detect). В виртуальной машине или контейнере

датчика кристалла часто действительно нет — страж не может защитить такую машину и правильно объявляет себя отключённым вместо того, чтобы притворяться.

`cpufreq : not writable – throttle stage disabled` Ожидаемо, пока вы не выполнили `./setup-cpufreq-perms.sh` — см. [раздел 8](#). Затрагивает только этап `throttle`; `stop` и `stop+restart` не задеты.

Выполнил `setup-cpufreq-perms.sh`, а всё равно пишет `not writable` Две вероятные причины: вы с тех пор не вышли и не вошли в систему заново (новая группа доходит до процессов только через новый вход — проверьте через `id`), либо панель с тех пор не перезапускалась (страж проверяет право записи один раз, при старте). Сделайте и то и другое, в этом порядке.

Пороги выглядят неверными для моего процессора Посмотрите, что публикует машина: `cat /sys/class/hwmon/hwmon*/temp*_crit`. Если ваш `crit` необычен или отсутствует, задайте абсолютные значения в `thermal_stop` и родственных ключах.

Он остановил сервер, хотя, по-моему, тот не был горячим Проверьте, не остался ли `thermal_test_temp` заданным после проверки. Это самая частая причина.

Он никогда не срабатывает, хотя машина греется Убедитесь, что режим не `log`, что `thermal_enabled` равен `true`, и сравните наблюдаемую температуру с порогами из `--once`. Помните, что порог должен быть превышен *непрерывно* в течение всего времени удержания.

Сервер постоянно перезапускается во время теплового события Это работает `lockout`, а строки `lockout`: — его отчёт об успехе: ваш супервизор продолжает пытаться, страж продолжает отменять. Исправлять нечего.

Панель не запущена — работает ли тогда страж? Нет — в режиме с панелью страж живёт внутри `admin_server.py`. Если нужна защита независимо от панели, используйте автономную службу из [раздела 7](#).

14. Отключение или удаление

- **Приостановить:** поставьте `thermal_mode` в `log`. Он продолжит наблюдать и сообщать, но никогда не будет действовать.
- **Полностью отключить:** поставьте `thermal_enabled` в `false`.
- **Автономный режим:** `sudo systemctl disable --now thermal-guard`.
- **Отменить права `cpufreq`:** `sudo ./setup-cpufreq-perms.sh --revoke`.

Удалить `thermal_guard.py` тоже безопасно — `admin_server.py` импортирует его внутри `try` и просто сообщает, что страж недоступен, если файла нет.

Смотрите также

- [Настройка панели администратора](#) — вкладка Thermal Guard в контексте
- [Установка](#) — тепловая защита при новой установке
- [Структура проекта](#) — где находится `thermal_guard.py`

ГЛАВА 5

Ограничения подключений

CONNECTION_LIMITS.md

Ограничения подключений — руководство системного оператора

Защита публичного приёмника от лавины подключений.

WebSDR — это публичная служба, которая выдаёт любому желающему непрерывный поток звука и спектра. В этом весь её смысл, и в этом же вся проблема: для сервера обычный слушатель ничем не отличается от того, кто открывает сессии в цикле. 16 сентября 2026 года этот приёмник столкнулся именно с таким случаем — один адрес открыл 134 слушательские сессии за одну минуту и держал 181 одновременно, тогда как настоящая аудитория станции составляла около десятка человек.

Это руководство — о трёх уровнях, которые такое останавливают, о том, что каждый из них может и чего не может, и о том, как понять, разумно ли настроены ваши.

Спешите? Пример `config.toml` уже содержит разумные значения, так что свежая установка защищена без каких-либо действий с вашей стороны. Если ваш собственный `.toml` старше и в нём нет ключей `[limits]`, кроме `audio/waterfall/events`, то ограничения по адресу просто **выключены** — скопируйте блок из примера, чтобы их включить. Единственная команда, которую стоит знать потом, — `grep Refused spectrumserver.log`: она скажет, отказывают ли кому-нибудь.

Содержание

1. От чего вы на самом деле защищаетесь
2. Три уровня
3. Ограничения по адресу — политика для слушателей
4. Простаивающие подключения — тихая атака
5. Защита в ядре — `setup-firewall.sh`
6. Справочник настроек
7. Что видит слушатель, которому отказали
8. Чтение журналов и счётчиков
9. Выбор ваших чисел
10. Ловушки, о которых стоит знать
11. Как всё отключить

1. От чего вы на самом деле защищаетесь

Здесь стоит быть точным, потому что слово «DDoS» покрывает две очень разные вещи, и только одна из них решается на вашей собственной машине.

Лавина из одного источника. Одна машина или несколько открывают подключения так быстро, как только могут. Именно это здесь и произошло, и это полностью поправимо: подключения приходят с адреса, который можно сосчитать, а счёта достаточно.

Настоящая распределённая атака. Тысячи машин, часто с подделанными адресами источника, забивают ваш канал. К тому моменту, когда эти пакеты доходят до вашей сетевой карты, полоса уже израсходована. **Ничто из описанного в этом руководстве не помогает**, и ничто из устанавливаемого на сервер помочь не может — единственный ответ находится выше вас по потоку, и это другой разговор, требующий настоящего домена и провайдера вроде Cloudflare перед приёмником.

Всё дальнейшее относится к первому случаю. Это не то ограничение, за которое стоит извиняться: именно первый случай и происходит с любительскими станциями на практике.

2. Три уровня



Они намеренно не дублируют друг друга. Уровень ядра быстр, но глуп — он считает пакеты с адреса и ничего не знает о слушателях. Уровень сервера точен, но видит подключение лишь после того, как оно принято и разобрано, а это чего-то стоит. Уровень браузера существует для того, чтобы слушатель, которому отказали, узнал, что произошло, а не смотрел на страницу, которая никогда не загрузится.

3. Ограничения по адресу

Что считается одним слушателем

Слушатель — это **одно подключение** `/audio` — или, там где включён `[kiwi_emulation]`, **одно звуковое подключение Kiwi**, которое считается точно так же. Это та же единица, которой уже пользуются счётчик пользователей, `users.json` и подписи на водопаде, так что клиент Kiwi занимает место и появляется в списке пользователей наравне с браузером.

Это важно, потому что одна-единственная вкладка настольного браузера открывает к вашему серверу **четыре** подключения:

Подключение	Назначение
/audio	поток демодулированного звука — вот это и есть слушатель
/waterfall	отображение спектра
/events	список пользователей, частоты других слушателей
/chat	панель чата

Телефон на /mobile открывает два (/audio и /chat), потому что водопада у него нет. Так что ограничение, считающее подключения как таковые, отказало бы всем: три вкладки — это двенадцать подключений.

Три ключа

```
[limits]
per_ip=3          # одновременные слушатели с одного адреса
per_ip_sockets=0  # одновременные подключения любого вида; 0 даёт per_ip * 4 + 4
per_ip_rate=40    # новые подключения в минуту с одного адреса
per_ip_ban_s=600  # на сколько отказывают адресу, превысившему темп
```

per_ip — главное число и то, о котором вы будете думать. Три — хорошая отправная точка.

per_ip_sockets существует потому, что /audio — не единственное, что имеет смысл заливать. /waterfall — самый тяжёлый поток, который производит ваш сервер, и ограничение, считающее только слушателей, обходилось бы простым открытием водопадных подключений. Оставленный в 0, он становится $\text{per_ip} * 4 + 4$, куда помещаются **per_ip** полных настольных вкладок с небольшим запасом на подключения, которые ядро ещё не успело убрать после перезагрузки страницы.

per_ip_rate — то, что действительно прекращает лавину. Один только потолок этого не делает: отказать в подключении дёшево, но не бесплатно, и атакующий может переподключаться так же быстро, как возвращаются отказы, сжигая ваш процессор на установке и разборе соединений. Ограничение темпа даёт ему блокировку вместо ответа. Помните, что одна вкладка стоит четырёх, так что 40 в минуту — это примерно десять загрузок страницы в минуту с одного адреса.

Намеренный порядок. Темп списывается до проверки потолка одновременных. Это выглядит шиворот-навыворот, пока не подумаешь об адресе, который уже на своём потолке и переподключается в цикле: если бы ему каждый раз отвечал дешёвый отказ «ты на потолке», нарушения темпа никогда бы не накопились, и блокировка, прекращающая цикл, никогда бы не сработала.

Кто освобождён

Петля обратной связи не ограничивается никогда. Autorun-ответвление для отправки спотов, панель администратора и браузер, открытый на самой машине сервера, приходят с 127.0.0.1, и если позволить им расходовать места, ваша собственная станция запирает бы ваших слушателей снаружи.

Это покрывает и проху.ру: подключения через прокси доходят до spectrumserver с петли обратной связи, а настоящий адрес клиента приходит в заголовке X-Forwarded-For — именно по нему и ведётся учёт.

Начиная с v5.0.0 этому заголовку верят **только от loopback-партнёра** — прокси на той же машине. Присланный кем-либо ещё, он игнорируется и учитывается TCP-адрес, иначе прямой посетитель мог бы написать X-Forwarded-For: 127.0.0.1 и пройти как loopback мимо всех

ограничений этой страницы. Кроме того, `proxu.py` отбрасывает любой `X-Forwarded-For` или `X-Real-IP`, присланный клиентом, прежде чем добавить свои, и отвечает на отключение сисопом (`/~kick`) только клиентам на той же машине. Станция с несколькими приёмниками за прокси описана в [Несколько приёмников](#); ограничения там действуют для каждого приёмника.

4. Простаивающие подключения

Есть атака, которую ничто из вышеперечисленного увидеть не может.

Открыть TCP-подключение. Ничего не отправлять. Держать.

Такой собеседник никогда не завершает запрос, а значит никогда не доходит до кода, где живут ограничения по адресу — нет слушателя, нет объекта клиента, нет ничего с именем, что можно было бы сосчитать. При этом ему это стоит нескольких байтов, а вам — файлового дескриптора и места на всё время удержания.

```
[limits]
idle_per_ip=8      # простаивающие подключения (до запроса) на адрес
idle_total=512     # потолок по всем адресам вместе
```

Ограничение — на **сколько их**, а не на **как долго**, и в этом различии весь замысел. **Никакого срока для простаивающего подключения нет, и это сделано намеренно.** Срок кажется очевидным ответом и является ловушкой (см. §10): существуют законные собеседники, которые подключаются, а потом долго молчат, прежде чем заговорить. От атаки их отделяет количество — законный собеседник держит одно такое подключение, атакующий тысячи.

Поэтому простаивающее подключение живёт до тех пор, пока не сделает запрос или не умрёт, и место освобождается в обоих случаях.

`idle_total` ограничивает лавину, размазанную по множеству адресов источника. При достижении потолка отказывают **новым** простаивающим подключениям, а не вытесняют существующие, — так что законное долгоживущее подключение никогда не оказывается тем, чем жертвуют ради места.

Оба применяются в момент принятия сокета, по адресу собеседника — запроса ещё нет, а значит нет и `X-Forwarded-For`, к которому можно обратиться. Это не беда: петля обратной связи не считается, чем покрыто всё приходящее через `proxu.py`, а прямая лавина несёт собственный адрес.

5. Защита в ядре

`setup-firewall.sh` устанавливает таблицу `nftables`, которая срезает объём до того, как `spectrumserver` прочтёт хотя бы один байт. Она намеренно груба, а её числа лежат заметно выше тех, что в `config.toml`, так что обычный слушатель никогда не может быть пойман обоими сразу.

```
./setup-firewall.sh --show      # напечатать правила, ничего не менять
sudo ./setup-firewall.sh --check # проверить на вашем ядре
sudo ./setup-firewall.sh --apply # загрузить, с автооткатом через 60 с
sudo ./setup-firewall.sh --persist # перезагружать при старте системы
sudo ./setup-firewall.sh --status # счётчики пакетов по каждому правилу
sudo ./setup-firewall.sh --remove # отменить всё
```

Покрываются четыре вещи: потолок одновременных подключений с одного адреса источника на ваших портах прокси и приёмника, темп новых подключений с источника, тормоз против перебора паролей SSH и общий доступ к файлам Windows, закрытый для всего за пределами частных диапазонов адресов — на многих машинах `smbd` слушает на `0.0.0.0`, а достижимо ли это снаружи, зависит от маршрутизатора, которого скрипт увидеть не может.

Порты читаются из `admin_config.json`, того же файла, которым пользуется `proxu.py`, так что настройка следует вашей.

Почему она не может запереть вас снаружи

Два свойства, оба намеренные:

- Политика таблицы — **accept**, и отбрасывает она только конкретные названные образцы. Правило, которое не совпало, пропускает пакет дальше — ко всему, что вы настроили помимо. Сделать машину недостижимой оно не может.
- **Установленные подключения принимаются самым первым правилом.** SSH-сессия, в которой вы печатаете, никогда не затрагивается тем, что идёт следом.

Сверх того `--apply` взводит автоматический откат: загрузить правила, затем подтвердить в течение 60 секунд, иначе они снимаются. Ничего не говорите, закройте терминал, потеряйте связь — набор правил исчезнет сам.

Используйте это окно как следует. Проверьте **с другого устройства, вне вашей собственной сети**, что приёмник по-прежнему загружается и что открывается *новая* SSH-сессия. Проверка той сессией, которая у вас уже есть, не доказывает ничего, потому что её принял первый же пункт правил.

6. Справочник настроек

Каждый ключ ниже живёт под `[limits]` в вашем файле `.toml`. **Все они по умолчанию выключены или щедры**, так что конфигурация, их не упоминающая, ведёт себя ровно так же, как и раньше.

Ключ	По умолчанию	Значение
<code>per_ip</code>	0 (выкл.)	Одновременные слушатели на адрес
<code>per_ip_sockets</code>	$0 \rightarrow \text{per_ip} * 4 + 4$	Одновременные подключения любого вида на адрес
<code>per_ip_rate</code>	0 (выкл.)	Новые подключения в минуту на адрес
<code>per_ip_ban_s</code>	600	Секунд отказа после превышения темпа
<code>idle_per_ip</code>	8	Простаивающие подключения, до запроса, на адрес
<code>idle_total</code>	512	Простаивающие подключения по всем адресам

Установка любого ключа в 0 отключает соответствующую проверку.

Три, которые не ограничения

В `[limits]` есть также `audio`, `waterfall` и `events`, унаследованные от исходного PhantomSDR, и **ни один из трёх не применяется**. `waterfall` и `events` не читает никакой код. `audio` читается в одном единственном месте: он отправляется как `max_users` в JSON регистрации каталогам из `[websdr] register_urls`, то есть это та цифра, которую показывают `sdr-list.xyz` и остальные, и он не делает совершенно ничего при `register_online=false`. Если нужен предел по слушателям, действует `per_ip` выше.

Ещё один, под [server]

```
[server]
min_client_version=0
```

Это не ограничение, а отказ, и, поскольку он использует тот же код закрытия, место ему здесь. Страницы сообщают сборку, из которой были загружены, в виде `/audio?v=N`, и всё, что ниже `min_client_version`, отправляется прочь с сообщением «*эта страница устарела — перезагрузите её*».

Он существует потому, что сервер не может дотянуться до JavaScript, уже работающего в чьём-то браузере. Вкладка хранит загруженный код, пока кто-нибудь её не перезагрузит, так что изменение страницы доходит лишь до тех слушателей, кто случайно перезагрузился, — а станции, только что изменившей поведение сессий, старые страницы могут быть не нужны прямо сейчас, а не когда-нибудь.

Проверяется только `/audio`, потому что именно там живёт сессия, а петля обратной связи и пути Kiwi освобождены, чтобы autorun-ответвление и клиенты Kiwi остались нетронутыми. `0` отключает проверку и является значением по умолчанию. Оставить её включённой после того, как старые страницы иссякнут, можно: текущий клиент diversity, получив отказ на голом `/audio`, повторяет попытку с меткой, так что отказ по-прежнему получают только клиенты diversity со старой сборкой PhantomSDR-Plus и сторонние инструменты, открывающие `/audio` без `?v=`. Верните её в `0`, если другая станция сообщит, что больше не может использовать вас как партнёра по diversity.

7. Что видит слушатель, которому отказали

Ничто никогда не переподключается

Начните отсюда, потому что это управляет всем остальным: **упавшее подключение `/audio` завершает сессию**. Автоматического переподключения нет, и это сделано намеренно.

Переподключаться — правильный рефлекс для чата и неправильный для приёмника. `/waterfall` и `/events` вместе с ним никогда не возвращались, так что повторённая сессия была живым звуковым подключением, привинченным к замороженному водопаду; а остановка сервера больше его не очищала, потому что каждый слушатель возвращался через несколько секунд, наполняя список пользователей сессиями, в которых на самом деле никого не было. Страница умолкает, и слушатель загружает её снова — именно так этот приёмник и вёл себя до того, как переподключение вообще появилось.

Для ограничений из этого руководства это вдобавок убирает ловушку: против ограничения темпа автоматическая попытка продлевала бы ровно ту блокировку, которую пыталась обойти.

Два кода закрытия

Код	Значение	Кто отправляет
4003	отказано — превышено ограничение по адресу либо страница устарела	spectrumserver
4001	отключён системным оператором	spectrumserver, когда панель вызывает <code>/~~kick</code>

Оба лежат в частном диапазоне 4000–4999, так что ни один нельзя спутать с состоянием протокола, и оба считаются окончательными на любой странице. Текст причины едет в кадре

закрытия, а не сообщением, потому что первый кадр на `/audio` зарезервирован под настройки приёмника, и всё отправленное раньше него сломало бы любое нормальное подключение.

На странице

Отказ при загрузке страницы — слушатель даже не начинает, поэтому обе страницы объясняются. Настольная версия показывает панель: «Этот приёмник отклонил подключение», причину и пояснение, что все, кто делит один интернет-канал, считаются одним адресом. `/mobile` показывает то же самое в своей области уведомлений.

Отказ или отключение посреди сессии — настольная страница просто останавливается, намеренно без сообщения: цикл отрисовки водопада прекращается, и это всё. `/mobile` показывает одну строку с просьбой перезагрузить страницу — кроме случая отключения оператором, когда и она молчит.

Это молчание после отключения намеренное. **Отключение — это системный оператор, завершающий сессию, а не наказание:** ничего не объявляется, и тот, кто хочет вернуться, загружает страницу снова и через секунду становится обычным слушателем. Единственное, что даёт код 4001, — браузер понимает, что это было не оборвавшееся подключение, поэтому отключение действительно срабатывает, а не отменяет само себя.

Отключение выполняет сам `spectrumserver`, а не `proxy.py`: слушатели подключаются прямо к порту приёмника, так что прокси не владеет ни одним их подключением и его отключение не достаёт никого. Панель вызывает `/~kick?ip=&secs=` — доступный только с петли обратной связи и проверяемый по настоящему TCP-собеседнику, а не по `X-Forwarded-For`, который присылает клиент. `secs` по умолчанию равен нулю, то есть без блокировки, но принимает и длительность, если адрес нужно поддержать снаружи; при этом используется тот же механизм блокировки, что и у `per_ip_rate`, поэтому переподключение отклоняется на входе.

8. Чтение журналов и счётчиков

Сервер

```
grep Refused spectrumserver.log
```

```
Refused /audio from 203.0.113.5: too many simultaneous connections
Refused /audio from 198.51.100.7: too many connection attempts
```

Если там ничего нет, никому не отказывают. Запись в журнал ограничена не более чем одной строкой на адрес каждые пять секунд — без этого атака превратилась бы в неограниченную запись на диск, а это просто более медленный способ положить станцию.

Ядро

```
sudo ./setup-firewall.sh --status
```

Смотрите на `counter packets N` у каждого правила; это и есть число действительно отброшенных. Сплошные нули означают, что ничего не блокировалось и ваши пороги стоят с удобным запасом.

Не пугайтесь списков `elements = { ... }` в наборах над правилами. Это ваши обычные слушатели, которых *отслеживают* относительно ограничений — запись появляется при первом подключении адреса и истекает через пару минут. Отслеживание — не блокировка.

9. Выбор ваших чисел

Гадать не нужно; ваши же журналы вам всё скажут. Вот измерение, которое задало здесь `per_ip=3`, по неделе `logs/users_*.jsonl`:

Одновременных сессий с одного адреса	Число адресов
1	682
2	54
3	13
больше 3	8

758 различных адресов за семь дней, и ограничение в три затронуло бы восемь из них. Лавина же сидела на 181.

Чтобы повторить это на своей станции, сопоставьте идентификаторы сессий по адресу в `logs/users_*.jsonl` и возьмите максимальное перекрытие. Учтите, что журнал пишет события `tune` и `disconnect`, но события `connect` в нём нет, так что начало сессии приходится выводить из её первого `tune`.

Помните про NAT. Клуб, школа, офис или оператор мобильной связи выглядят как один адрес, и все, кто за ним, делят одну квоту. Это присуще любому ограничению, привязанному к адресу; единственное смягчение — выбрать число, с которым вам спокойно. Если слушатель когда-нибудь сообщит, что не может подключиться, подозревать нужно в первую очередь это — загляните в `grep Refused` прежде всего остального.

10. Ловушки, о которых стоит знать

Никогда не ставьте срок простаивающему подключению

Настройки `handshake_s` больше нет, и вот почему. Срок в 30 секунд для простаивающих подключений выглядит очевидной защитой от `slowloris`. Его написали, включили, и за минуту он выдал вот это:

```
Idle deadline: closed 1 connection(s) idle past 30s: 192.87.173.88
```

192.87.173.88 — это `etgd-websdr.ewi.utwente.nl`, **собственный хост обратного вызова `websdr.org`**. Он подключается обратно к вашему серверу и затем простаивает, заметно дольше 30 секунд, прежде чем отправить свой `GET /~orgstatus`. Срок обрубал его каждый раз, и регистрация перестала работать. Настройку убрали, а не оставили как опцию: выстрел себе в ногу, ломающий вашу запись в каталоге, не стоит тех шестидесяти строк, которые его реализуют.

То же рассуждение относится к `open_handshake_timeout` в `websocketpp`, который на месте. Десятиминутное значение в `spectrumserver.cpp` выглядит абсурдным и является несущим: именно оно позволяет обратному вызову `websdr.org` простаивать. Есть и вторая причина — обработчик `/~orgstatus` отвечает из потока, удерживающего `dup()` сокета, а `websocketpp` завершает истёкшее подключение вызовом `shutdown()`, который работает на уровне сокета, дотягивается сквозь дубликат и сломал бы обратный вызов, даже если бы время простоя не было проблемой.

Если нужно ограничить молчащие подключения, ограничивайте их число, а не возраст. Именно этим и являются `idle_per_ip` и `idle_total`.

Файл nftables обязан быть идемпотентным

nft -f **дописывает** в уже существующую таблицу, а не заменяет её. Файл правил без преамбулы `delete table` поэтому удваивает всю цепочку при каждой перезагрузке — а именно это и делает наивный юнит systemd при каждом перезапуске службы. `setup-firewall.sh` формирует преамбулу за вас; если вы пишете собственные правила, поступайте так же.

```
sudo ./setup-firewall.sh --status | grep -c dport    # должно быть 8, а не 16
```

11. Как всё отключить

Ограничения сервера: поставьте ключи в 0 или удалите их из своего `.toml` и перезапустите приёмник. Другого состояния нет; ничто не переживает перезапуска.

Межсетевой экран:

```
sudo ./setup-firewall.sh --remove
```

Это удаляет таблицу, убирает сохранённый набор правил и отключает службу автозапуска. Ваша машина возвращается ровно в то состояние, в котором была до того, — что на большинстве установок означает полное отсутствие межсетевого экрана. Об этом стоит подумать, прежде чем его убирать.

ГЛАВА 6

Несколько приёмников

MULTI_RECEIVER.md

Несколько приёмников — Руководство сисопа

Два и более приёмника на одном компьютере, с выбором приёмника прямо на странице.

Начиная с v5.0.0 один компьютер с PhantomSDR-Plus может одновременно обслуживать несколько приёмников — например, RX-888 для KB и RTL-SDR для 2 м, — а слушатели переключаются между ними кнопками в шапке страницы, так же как OpenWebRX предлагает свои профили. У каждого приёмника свой водопад, чат, метки, список слушателей и сведения о станции; никто на одном приёмнике никогда не мешает тем, кто на другом.

Спешите? Станции с одним приёмником ничего из этого не нужно, и для неё ничего не меняется. Для второго приёмника: создайте `instances/<имя>/` со своим `config.toml`, запустите его командой `INSTANCE=<имя> ./start-<радио>.sh`, впишите оба приёмника в `receivers.toml` и перезапустите прокси. Пример в [разделе 3](#) разбирает всё построчно.

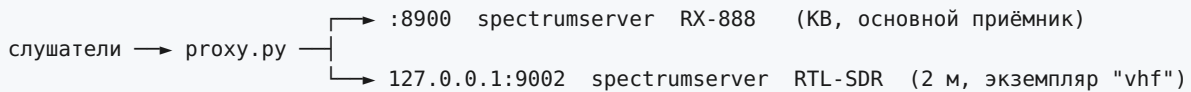
Проще всего: запустите `./add-receiver.sh` (установщики предлагают его в конце). Он спрашивает, какой приёмник и что он должен покрывать, устанавливает драйвер, записывает всё, что описано в [разделе 3](#) и [разделе 4](#), и предлагает запустить приёмник и перезапустить прокси. Остальная часть руководства объясняет, что он делает, — на случай, если вы захотите что-то изменить вручную.

Содержание

1. Как это работает
2. Два способа опубликовать приёмники
3. Добавление второго приёмника
4. `receivers.toml`
5. Запуск и остановка
6. Что видят слушатели
7. S-метр выше 30 МГц
8. Безопасность
9. Сколько приёмников поместится
10. Ловушки, о которых стоит знать

1. Как это работает

Каждый приёмник — это полноценный отдельный `spectrumserver` со своим SDR, своей конфигурацией и своим внутренним портом. Они никогда не делят отсчёты, настройки или слушателей. Связывает их `proxu.py` — обратный прокси, который уже обслуживает панель администратора: он читает `receivers.toml` и направляет каждый запрос одному приёмнику.



Прокси решает по следующим признакам, в таком порядке:

1. `?rx=<id>` в адресе — `http://ваш.хост:8899/?rx=vhf`. Страница добавляет его к каждому сокету, запросу и ссылке, которые открывает, поэтому вкладка остаётся на своём приёмнике.
2. Cookie `rx`, который прокси устанавливает всякий раз, когда запрос несёт допустимый `?rx=`.
3. Заголовок `Host`, если у приёмника это имя указано в `hostnames`.
4. Иначе — приёмник по умолчанию.

Основной приёмник — тот, что запускается обычным способом (`./start-rx888mk2.sh`). Любой другой приёмник — **именованный экземпляр**: он запускается с `INSTANCE=<имя>`, настраивается в `instances/<имя>/` и имеет свой журнал (`logwebsdr-<имя>.txt`), журнал сервера, блокировку `watchdog` и `FIFO`. Процессы различаются по метке `PHANTOMSDR_INSTANCE` в их окружении, а не по имени, поэтому два приёмника могут запускать одну и ту же программу — два `rtl_sdr` или `rx_sdr` для RSP1A и Airspy, — и перезапуск одного никогда не затрагивает другой.

2. Два способа опубликовать приёмники

А — основной приёмник сохраняет свой порт. Основной приёмник остаётся там же, где был (например, `:8900`), его слушатели ничего не замечают, а к остальным приёмникам обращаются через порт прокси (`proxy_port` в `admin_config.json`, например `:8899`) с `?rx=`. Ничего не переносится и ничто не уходит из эфира; зато публичных портов становится два.

```

KB: http://ваш.хост:8900/
2 м: http://ваш.хост:8899/?rx=vhf
  
```

Б — один публичный порт для всего. Прокси занимает публичный порт, а основной приёмник переезжает на внутренний. Задайте `[front] port` в `receivers.toml`, перенесите `[server] port` основного приёмника и укажите у него `host = "127.0.0.1"`, впишите новый внутренний порт в `public_port` файла `admin_config.json` и добавьте `public_port = <публичный порт>` в раздел `[websdr]`, чтобы каталог по-прежнему объявлял правильный. Тогда всё находится за одним портом; цена — короткий перерыв на время переезда.

```

KB: http://ваш.хост:8900/
2 м: http://ваш.хост:8900/?rx=vhf
  
```

Выбор приёмника работает одинаково в обоих случаях. Схема А — более безопасное начало; Б можно сделать позже, ничего не меняя во втором приёмнике.

3. Добавление второго приёмника

`add-receiver.sh` выполняет все шаги ниже — запустите его и ответьте на вопросы; он также устанавливает драйвер (`setup-rtlsdr.sh` для RTL-SDR, соответствующий `setup-*.sh` для остальных), выбирает свободный внутренний порт и называет приёмник по тому, что он покрывает. Шаги расписаны здесь, чтобы вы могли проверить его работу или изменить её позже.

Пример добавляет RTL-SDR Blog V4 для 2 м как экземпляр `vhf` рядом с RX-888, который сохраняет порт 8900 (схема A).

1. Драйвер. RTL-SDR Blog V4 нужен драйвер RTL-SDR Blog, а не пакет `rtl-sdr` из дистрибутива, и драйвер DVB-T ядра должен освободить свисток. `install.sh` (вариант приёмника 2, «RTL-SDR Blog V4: да») делает и то и другое. Проверьте командой `rtl_test -t` — она должна вывести RTL-SDR Blog V4 Detected.

2. Папка экземпляра. Всё, что относится к приёмнику, лежит в `instances/vhf/`; эта папка никогда не попадает в репозиторий и не перезаписывается обновлениями:

```
instances/vhf/
├─ config.toml          конфигурация его spectrumserver (обязательно)
├─ instance.env         необязательно: параметры приёмника и привязка к ядрам
├─ markers.json        его собственные метки
├─ www/                его копии файлов станции для страницы
│   └─ site_information.json
│   └─ wf-message.json
```

`spectrumserver` работает с `instances/vhf/` в качестве рабочего каталога, поэтому история чата, метки, FFTW wisdom и `logs/` тоже принадлежат ему.

3. config.toml. Возьмите за основу `config-rtl.toml` и измените:

```
[server]
port=9002                # свой внутренний порт
host="127.0.0.1"         # доступен только через прокси
html_root="www/"         # его немногие собственные файлы...
html_fallback_root="../../frontend/dist/" # ...а остальное — общая страница

[input]
sps=2400000              # как -s в instance.env
frequency=145000000       # как -f в instance.env

[input.defaults]
frequency=145500000
modulation="FM"

[kiwi_emulation]
enabled = false           # см. ловушки ниже
```

`html_fallback_root` позволяет экземпляру хранить только свои `site_information.json` и `wf-message.json`, а всё остальное — саму собранную страницу — брать из `frontend/dist`; так одна сборка фронтенда доходит до каждого приёмника. Копия `frontend/dist` из символических ссылок не работает: сервер отказывается отдавать файлы, которые оказываются за пределами его папки.

4. instance.env. Скрипт запуска читает его после своих собственных настроек:

```
RX_ARGS="-f 145000000 -s 2400000 -g 29.7 -" # -f и -s как в config.toml
SPECTRUM_CORES=8-11                       # подальше от ядер основного приёмника
```

5. Сведения его страницы. Скопируйте `frontend/site_information.json` в `instances/vhf/www/` и отредактируйте: `siteReceiver`, `siteAntenna`, `siteSDRBaseFrequency` и `siteSDRBandwidth` (они определяют, какие кнопки диапазонов появятся), а также ключи из [раздела 6](#).

6. Запустите его.

```
INSTANCE=vhf ./start-rtl.sh
```

7. Впишите его в `receivers.toml` и перезапустите прокси (`sudo systemctl restart phantomsdr-proxy`). Второй приёмник теперь доступен по адресу `http://ваш.хост:8899/?rx=vhf`, и обе страницы показывают выбор приёмника.

4. `receivers.toml`

Скопируйте `receivers.toml.example` в `receivers.toml`. Без этого файла прокси обслуживает один приёмник, как и раньше.

```
# [front]
# port = 8900          # только схема Б: прокси слушает и этот публичный порт

[[receiver]]
id      = "hf"
name    = "HF 0-30 MHz (RX-888 MkII)"
port    = 8900
default = true
launcher = "start-rx888mk2.sh"
url      = "http://ваш.хост:8900/"

[[receiver]]
id      = "vhf"
name    = "2 m (RTL-SDR Blog V4)"
port    = 9002
launcher = "start-rtl.sh"
instance = "vhf"
url      = "http://ваш.хост:8899/?rx=vhf"
```

Ключ	Значение
<code>id</code>	Короткое имя для <code>?rx=</code> ; буквы, цифры, <code>-</code> и <code>_</code>
<code>name</code>	Надпись на кнопке приёмника
<code>port</code>	<code>[server] port</code> его <code>spectrumserver</code>
<code>host</code>	Необязательно; где работает этот <code>spectrumserver</code> (по умолчанию <code>127.0.0.1</code>)
<code>default</code>	Приёмник для запросов, которые никого не называют; не более одного
<code>launcher</code>	Его <code>start-*.sh</code> для <code>start-all.sh</code> ; опустите для приёмника, который этот компьютер не запускает
<code>instance</code>	Имя его экземпляра; опустите для основного приёмника
<code>url</code>	Куда кнопка отправляет слушателя; без него — <code>/?rx=<id></code> на прокси
<code>hostnames</code>	Необязательные DNS-имена, ведущие прямо к этому приёмнику

Прокси также отдаёт список как `/receivers.json` (идентификаторы, имена и ссылки — никогда внутренние порты); именно его читает выбор приёмника.

5. Запуск и остановка

Команда	Действие
<code>./start-rx88mk2.sh</code>	Запускает или перезапускает основной приёмник
<code>INSTANCE=vhf ./start-rtl.sh</code>	Запускает или перезапускает приёмник <code>vhf</code>
<code>./start-all.sh</code>	Запускает каждый приёмник из <code>receivers.toml</code> , у которого есть <code>launcher</code>
<code>./stop-websdr.sh vhf</code>	Останавливает только <code>vhf</code>
<code>./stop-websdr.sh main</code>	Останавливает только основной приёмник
<code>./stop-websdr.sh</code>	Останавливает все приёмники, как и раньше

Пока приёмник остановлен, его кнопка исчезает из выбора — прокси перечисляет только отвечающие приёмники, а открытые страницы перечитывают список раз в минуту, — и возвращается, когда приёмник снова запущен; если работать остаётся только один приёмник, строка *Receivers:* исчезает совсем. Однако остановленный приёмник сам возвращается всякий раз, когда запускаются все приёмники: `./start-all.sh`, **Restart** в панели администратора и перезапуск тепловой защитой запускают всё, что перечислено в `receivers.toml`. Чтобы один оставался выключенным насовсем, удалите (или закомментируйте) его блок `[[receiver]]` и перезапустите прокси.

В панели администратора установите **Default start script** в `start-all.sh`. Её Stop и так останавливает все приёмники, а Restart и тепловая защита используют скрипт запуска; с `start-all.sh` они возвращают все приёмники, а не только основной.

Помощник RADE принадлежит только основному приёмнику. Именованный экземпляр никогда его не запускает и не останавливает.

6. Что видят слушатели

Выбор приёмника. На станции с несколькими приёмниками в шапке страницы появляется отдельная строка *Receivers:* с кнопкой для каждого приёмника; текущий показан жёлтым. На странице `/mobile` те же кнопки стоят вторым рядом в верхней панели и открывают страницу `/mobile` другого приёмника.

Собственные сведения каждого приёмника. Страница, открытая на втором приёмнике, перед запуском загружает `site_information.json` этого приёмника, поэтому кнопки диапазонов, начальная частота, список слушателей, сведения о станции и ссылки *Receiver* и *Antenna* — все его собственные. Обычное посещение основного приёмника не делает ни одного дополнительного запроса.

Ключи `site_information.json` для этого:

Ключ	Значение
<code>siteReceiverId</code>	Приёмник, которому принадлежит страница (его <code>id</code>); на основном приёмнике опустите
<code>siteReceiversList</code>	Откуда выбор приёмника читает список, напр. <code>http://ваш.хост:8899/receivers.json</code> ; нужно на приёмнике, страница которого отдаётся не через прокси
<code>siteReceiverURL</code>	Куда ведёт имя у <i>Receiver</i> в <i>Open Additional Info</i>
<code>siteAntennaURL</code>	Куда ведёт имя у <i>Antenna</i> ; "" показывает имя без ссылки

Когда выбор приёмника скрыт. Выбор показывает только список, в котором назван хост, по которому открыта страница. Открытая по адресу локальной сети (`http://192.168.1.10:8900/`), страница его не показывает; открытая как `http://ваш.хост:8900/` — показывает. Это сделано намеренно: станция, скопировавшая чужой `site_information.json` без правки, никогда не покажет приёмники той станции как свои.

7. S-метр выше 30 МГц

Стандарт IARU Region 1 помещает S9 на -73 дБм ниже 30 МГц и на **-93 дБм выше**, с шестью дБ на балл S в обоих случаях. S-метры (аналоговая стрелка, цифровая полоса и полоса на /mobile) следуют частоте настройки: выше 30 МГц они показывают баллы S для УКВ, ниже — как прежде. Значения в дБм и дБмкВ никогда не меняются. Если вы хотите, чтобы приборы оставались на 0 на пустом канале, как у трансивера УКВ, добавьте `"siteSMeterGateDb": 6` в `site_information.json` приёмника (пересборка не нужна): выше **60 МГц** стрелка и полоса тогда поднимаются, только когда сигнал на столько дБ превышает шум своего окружения (самая сильная точка полосы пропускания против спектра ± 100 кГц вокруг). По умолчанию это **выключено** — откалиброванный прибор, показывающий реальный шум диапазона, честнее. Страница /mobile, у которой нет водопада, оценивает шум по самому принимаемому уровню.

Откалибруйте дБм приёмника УКВ ключами `analog_smeter_offset` (стрелка и цифры) и `smeter_offset` (цифровая полоса) в разделе [input] его `config.toml`, затем перезапустите этот приёмник. Сначала задайте усиление: RTL-SDR сообщает уровни относительно своей полной шкалы, поэтому каждое изменение усиления сдвигает показания. Без генератора сигналов нагрузка 50 Ом вместо антенны, в USB с фильтром 2,7 кГц, должна показывать около -136 дБм (тепловой шум в 2,7 кГц равен $-139,7$ дБм, плюс коэффициент шума свистка).

Для RTL-SDR `add-receiver.sh` записывает начальную калибровку: фиксированное усиление `-g 29.7` в `instance.env` (автоматическое усиление тюнера откалибровать нельзя) и `analog_smeter_offset=-55, smeter_offset=-38`, измеренные на RTL-SDR Blog V4 при этом усилении сигналом -71 дБм (63 мкВ). Другой свисток той же модели обычно расходится лишь на несколько дБ; проверьте его известным сигналом и откалибруйте заново, если меняете усиление.

8. Безопасность

Установка прокси перед слушателями закрыла две дыры, обе исправлены в v5.0.0:

- **Отключение слушателя сисопом через прокси.** `spectrumserver` разрешает `/~~kick` только со своего компьютера, а для него каждый запрос, пересланный прокси, приходит со своего компьютера. Поэтому любой, кто мог достучаться до порта прокси, мог отключить и заблокировать любого слушателя. Теперь прокси отвечает на `/~~kick` только клиентам на том же компьютере.
- **Поддельные адреса клиентов.** `spectrumserver` верил заголовку `X-Forwarded-For` от кого угодно, так что посетитель мог назваться `127.0.0.1` и обойти ограничения по адресу. Теперь заголовку верят только от локального прокси, а прокси отбрасывает любую копию, присланную клиентом, прежде чем добавить свою.

Привязывайте каждый приёмник, доступный через прокси, к `127.0.0.1` (`[server] host`), чтобы его порт нельзя было использовать в обход прокси. Ограничения по адресу из [Ограничений подключений](#) действуют для каждого приёмника отдельно.

9. Сколько приёмников поместится

Программа не ставит предела; его ставит железо. Узкополосные приёмники нетребовательны: RTL-SDR на 2,4 Msps занимает около 5–10 % одного ядра, несколько десятков МБ памяти и около 38 Мбит/с USB.

- **Обычный потолок — USB 2.0.** Все устройства USB 2 делят одну шину на 480 Мбит/с, даже в синем разъёме USB 3. Три-четыре RTL-SDR на шину — надёжная цифра; добавляйте их по одному и следите, нет ли пропусков на каждом водопаде.
- **Питание:** больше двух свистков — используйте хаб с собственным питанием.
- **Второй широкополосный приёмник** (ещё один RX-888 или HackRF на 20 Msps) — другое дело: он конкурирует с первым за USB 3, CPU и GPU.
- **Слушатели обходятся дороже приёмников.** Каждый слушатель стоит CPU на своём приёмнике и исходящей полосы, поэтому общее число слушателей важнее числа приёмников.

10. Ловушки, о которых стоит знать

- **У двух свистков одной модели одинаковый серийный номер.** Каждый RTL-SDR Blog V4 сообщает 00000001. Дайте каждому свой командой `rtl_eeprom -s <номер>` (подключая по одному свистку) и укажите его в `instance.env` (`RX_ARGS="-d <номер> ..."`), иначе после перезагрузки свистки могут поменяться приёмниками.
- **Клиенты Kiwi не могут выбрать приёмник.** Клиент KiwiSDR обращается просто к адресу и порту; через прокси он всегда попадает на приёмник по умолчанию. На остальных держите `[kiwi_emulation]` выключенным.
- **Cookie действуют для хоста, а не для порта.** Посетитель, пользовавшийся вторым приёмником на `:8899`, приносит `rx=vhf` и на `:8900`. Поэтому страница определяет, какой она приёмник, по `siteReceiverId`, а не по cookie.
- **Записи в каталогах.** Дайте каждому приёмнику своё имя в `[websdr]`, а у приёмника за прокси установите `[websdr] public_port` на порт, которым пользуются слушатели, иначе запись будет рекламировать внутренний.
- **Сборка фронтенда ненадолго показывает «Not Found».** `frontend/dist` пересобирается на месте, поэтому на несколько секунд сборки страница пропадает у каждого приёмника.
- **Ничто не запускает приёмники при включении компьютера,** если вы этого не настроите. После перезагрузки `./start-all.sh` возвращает их все.

ГЛАВА 7

Сервер статистики

sdr-stats/README.md

Руководство по установке System Stats

Полное руководство по добавлению мониторинга сервера в реальном времени в ваше приложение PhantomSDR с помощью автоматизированного скрипта установки.

Обзор

Эта функция добавляет кнопку **Stats** в интерфейс PhantomSDR, которая отображает системную информацию в реальном времени:

- Загрузка ЦП, ядра, частота, температура и основные процессы
- Использование памяти (использовано/всего/процент)

Статистика автоматически обновляется каждые 5 секунд, пока открыто модальное окно.

Это не то же самое, что страница «Графики» в панели администратора. Это окно показывает мгновенный срез, который обновляется, пока окно открыто, и не хранит историю. В панели администратора есть отдельная страница **Графики**: она снимает частоту процессора, нагрузку, температуру и число слушателей каждые 2 секунды и строит их за период от 15 минут до 24 часов — см. [руководство по панели администратора](#).

Предварительные требования

- **Сервер Linux** (рекомендуется Ubuntu/Debian)
- **Подключение к Интернету** (для загрузки Node.js и пакетов)
- **Доступ Sudo** (для настройки службы systemd)
- **PhantomSDR-Plus** уже установлен и запущен
- **lm-sensors** (рекомендуется для температуры ЦП Intel/AMD — устанавливается автоматически скриптом)

Методы установки

ПРИМЕЧАНИЕ

Главный установщик уже это предлагает. `./install.sh` — и четыре установщика для дистрибутивов — запускают `install-stats-server.sh` по умолчанию в рамках обычной установки PhantomSDR-Plus, так что на новой машине сервер статистики обычно уже на месте. Эта страница нужна, чтобы установить его отдельно или позже изменить порт, адрес либо службу.

Вариант 1: Автоматизированная установка (Рекомендуется)

Автоматизированный скрипт всё сделает за вас.

Шаг 1: Запустите скрипт

```
cd ~/PhantomSDR-Plus
chmod +x install-stats-server.sh # only if it is not already executable
./install-stats-server.sh
```

ВАЖНО: НЕ запускайте от имени root или с sudo. Запускайте от имени обычного пользователя.

Шаг 2: Следуйте интерактивным подсказкам

Скрипт проведёт вас через:

1. Установка Node.js (при необходимости)

```
△ Node.js / npm is not installed
Would you like to install Node.js 22 via nvm now? (y/n): y
```

Скрипт устанавливает Node.js 22 через nvm, в любом дистрибутиве Linux — без root и без источника apt. Тот же вопрос задаётся, если Node.js установлен, но его версия ниже 22.

2. Каталог установки

```
Installation directory [/home/user/sdr-stats-server]:
```

Нажмите Enter для значения по умолчанию или укажите произвольный путь.

3. Настройка порта (Улучшено!)

```
Port Configuration:
  ▫ Default port: 3001
  ▫ Common alternatives: 8080, 5000, 8888
Enter port number [3001]:
```

Скрипт проверяет ваш порт:

- Проверяет, является ли он допустимым числом
- Предупреждает, если порт < 1024 (требуется права root)
- Проверяет диапазон (1-65535)
- Проверяет, используется ли порт уже
- Показывает, какой процесс его использует, если он занят
- Позволяет выбрать другой порт

Примеры сценариев:

Использование порта по умолчанию:

```
Enter port number [3001]: ↵
✓ Port 3001 selected
```

Выбор произвольного порта:

```
Enter port number [3001]: 8080
✓ Port 8080 selected
```

Порт уже используется:

```

Enter port number [3001]: 3001
△ Port 3001 is currently in use
Process using port 3001:
node    12345 user    20u  IPv6 123456  TCP *:3001 (LISTEN)
Choose a different port? (y/n): y
Enter port number [3001]: 3002
✓ Port 3002 selected

```

Недопустимый порт:

```

Enter port number [3001]: abc
x Port must be a number
Enter port number [3001]: 99999
x Port must be between 1 and 65535
Enter port number [3001]: 3001
✓ Port 3001 selected

```

4. Адрес сервера

```

What is your server's public address?
Examples: Your_site_IP, 192.168.1.100, localhost
Server address: Your_site_IP

```

Введите публичный домен или IP-адрес вашего сервера.

5. Подтверждение

```

Please confirm your settings:
  Installation directory: /home/user/sdr-stats-server
  Port: 3001
  Server address: Your_site_IP
  Stats URL will be: http://Your_site_IP:3001
Continue with these settings? (y/n): y

```

6. Настройка службы Systemd (Опционально)

```

Would you like to set up the server as a system service (auto-start on boot)? (y/n): y

```

Рекомендуется: Выберите **y**, чтобы сервер запускался автоматически при загрузке.

Шаг 3: Установка завершена! ✓

Скрипт отобразит сводку:

```

Installation Complete! ✓
Open the port in the router!

```

```

Configuration Summary:
Installation: /home/user/sdr-stats-server
Port: 3001
Server: Your_site_IP
API URL: http://Your_site_IP:3001/api/system-stats

```

Настройте ваше приложение PhantomSDR

После завершения работы скрипта вам необходимо обновить ваше приложение Svelte.

Шаг 1: Обновите `site_information.json`

Поскольку `site_information.json` уже был отредактирован во время начальной настройки, вам нужно только добавить новую строку `siteStats` с вашим портом:

```
{
  "siteSysop": "your name or callsign",
  "siteSysopEmailAddress": "mail@mail.net",
  "siteGridSquare": "QTH locator",
  "siteCity": "City Country",
  "siteInformation": "https://github.com/svlbtl/PhantomSDR-Plus",
  "siteHardware": "Hardware you are using, ",
  "siteSoftware": "Software you are using",
  "siteReceiver": "Receiver model",
  "siteAntenna": "Receiving Antenna.",
  "siteNote": "This is a bright new open-source WebSDR project, under active development.",
  "siteIP": "http://Your_site_IP:port",
  "siteStats": "http://Your_site_IP:3001", ← ДОБАВЬТЕ ЭТУ СТРОКУ (используйте свой порт)
  "siteSDRBaseFrequency": 0,
  "siteSDRBandwidth": 30000000,
  "siteRegion": 1,
  "siteChatEnabled": true
}
```

Примечание: Используйте порт, который вы выбрали во время установки!

Шаг 2: Обновите `App.svelte` (последняя версия с GitHub уже включает обновлённый файл)

Внесите эти **4 изменения** в ваш файл `App.svelte`:

Изменение 1: Убедитесь, что `siteStats` объявлен (около строки 50)

```
import {
  siteSysop,
  siteSysopEmailAddress,
  siteInformation,
  siteGridSquare,
  siteCity,
  siteHardware,
  siteSoftware,
  siteReceiver,
  siteAntenna,
  siteNote,
  siteIP,
  siteStats, // ← ДОБАВЬТЕ ЭТУ СТРОКУ
  siteSDRBaseFrequency,
  siteSDRBandwidth,
  siteRegion,
  siteChatEnabled,
} from "../site_information.json";
```

Изменение 2: Обновите объект systemStats (около строки 530)

```
let systemStats = {
  cpu: { usage: 0, cores: 0, temperature: null, frequency: null, topProcesses: [] }, // ← ДОБАВЬТЕ
  topProcesses: []
  memory: { used: 0, total: 0, percent: 0 }
};
```

Изменение 3: Обновите URL для fetch (около строки 541)

```
async function fetchSystemStats() {
  try {
    const response = await fetch(`${siteStats}/api/system-stats`); // ← ИЗМЕНИТЕ ЭТУ СТРОКУ
    if (response.ok) {
      systemStats = await response.json();
    } else {
      console.error('Failed to fetch system stats:', response.statusText);
    }
  } catch (error) {
    console.error('Error fetching system stats:', error);
  }
}
```

Изменение 4: Добавьте отображение основных процессов (в разделе ЦП, около строки 3838)

Добавьте этот код после раздела температуры:

```
{#if systemStats.cpu.frequency}
<div style="display: flex; justify-content: space-between; margin-bottom: 0.25rem;">
  <span>Frequency:</span>
  <span style="color: #4ade80;">{systemStats.cpu.frequency.current} GHz{#if
systemStats.cpu.frequency.max} <span style="opacity: 0.7;">(max {systemStats.cpu.frequency.max})</
span>{/if}</span>
</div>
{/if}

{#if systemStats.cpu.temperature !== null}
<div style="display: flex; justify-content: space-between; margin-bottom: 0.25rem;">
  <span>Temperature:</span>
  <span style="color: {systemStats.cpu.temperature > 70 ? '#fbbf24' :
'#4ade80'};">{systemStats.cpu.temperature}°C</span>
</div>
{/if}

<!-- ДОБАВЬТЕ ВСЕ ЭТОТ РАЗДЕЛ: -->
{#if systemStats.cpu.topProcesses && systemStats.cpu.topProcesses.length > 0}
<div style="margin-top: 0.75rem; padding-top: 0.75rem; border-top: 1px solid
rgba(255,255,255,0.1);">
  <h4 style="margin: 0 0 0.5rem 0; font-size: 0.85rem; color: rgba(0, 225, 255, 0.8);">Top
Processes:</h4>
  {#each systemStats.cpu.topProcesses as process}
    <div style="display: flex; justify-content: space-between; margin-bottom: 0.25rem; font-size:
0.85rem;">
      <span style="overflow: hidden; text-overflow: ellipsis; white-space: nowrap; max-width:
70%;">{process.name}</span>
      <span style="color: {process.cpu > 50 ? '#ef4444' : process.cpu > 25 ? '#fbbf24' :
'#4ade80'};">{process.cpu}%</span>
    </div>
  {/each}
</div>
{/if}
```

Шаг 3: Перекомпилируйте ваше приложение

```
cd /path/to/your/phantomsdr-app  
npm ./recompile.sh
```

Тестирование установки

Тест 1: Проверьте конечную точку API

```
curl http://localhost:3001/api/system-stats
```

Ожидаемый вывод (JSON):

```
{  
  "cpu": {  
    "usage": 45.2,  
    "cores": 8,  
    "coresUsed": 1.3,  
    "temperature": 62.5,  
    "frequency": {  
      "current": 3.0,  
      "max": 3.9,  
      "limit": 4.4  
    },  
    "topProcesses": [  
      {"name": "node", "cpu": 12.5},  
      {"name": "phantomsdr", "cpu": 8.3}  
    ]  
  },  
  "memory": {  
    "used": 8.5,  
    "total": 16,  
    "percent": 53  
  }  
}
```

Тест 2: Проверьте статус службы

Если вы установили её как службу:

```
sudo systemctl status sdr-stats.service
```

Должно показать: Active: active (running)

Тест 3: Протестируйте в браузере

1. Откройте ваш веб-интерфейс PhantomSDR
2. Прокрутите до раздела **Additional Info**
3. Нажмите кнопку **Open Additional Info**
4. Найдите кнопку **Stats** рядом с информацией о ПК
5. Нажмите на неё, чтобы открыть модальное окно статистики
6. Убедитесь, что статистика отображается и обновляется

Управление службой

Если вы установили её как службу systemd, используйте эти команды:

```
# Start the service
sudo systemctl start sdr-stats.service

# Stop the service
sudo systemctl stop sdr-stats.service

# Restart the service (after updating files)
sudo systemctl restart sdr-stats.service

# Check status
sudo systemctl status sdr-stats.service

# View real-time logs
sudo journalctl -u sdr-stats.service -f

# View last 50 log entries
sudo journalctl -u sdr-stats.service -n 50

# Enable auto-start on boot
sudo systemctl enable sdr-stats.service

# Disable auto-start
sudo systemctl disable sdr-stats.service

# Edit service
sudo nano /etc/systemd/system/sdr-stats.service

# Delete service
sudo rm /etc/systemd/system/sdr-stats.service
```

Ручная установка (Вариант 2)

ВАЖНО

Для обычной установки этот раздел не нужен. `install-stats-server.sh` делает всё это за вас, и главный установщик запускает его по умолчанию. Следуйте ему только для ручной установки или чтобы починить отдельную часть вручную.

⚠ **Предупреждение:** При ручной установке могут использоваться устаревшие файлы.

Настоятельно рекомендуется автоматизированный скрипт, так как он:

- Устанавливает последнюю версию с улучшенным определением температуры
- Автоматически устанавливает и настраивает lm-sensors
- Автоматически управляет всеми зависимостями

Если вы всё же предпочитаете ручную установку:

1. Установите Node.js

```
# Any distribution - nvm needs no root and no apt source.
# NOT NodeSource: deb.nodesource.com now answers HTTP 403 on every path.
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.4/install.sh | bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
nvm install 22 && nvm use 22 && nvm alias default 22
```

2. Установите lm-sensors (Требуется для температуры Intel/AMD)

```
sudo apt-get update
sudo apt-get install -y lm-sensors
sudo sensors-detect --auto
```

3. Создайте каталог

```
mkdir ~/sdr-stats-server
cd ~/sdr-stats-server
```

4. Загрузите последние файлы

Важно: файлы сервера больше не хранятся копиями в репозитории - их создаёт скрипт установки.

Вместо этого запустите автоматизированный скрипт один раз, чтобы сгенерировать последние файлы, а затем скопируйте их:

```
# Run the automated script from the repository root
cd ~/PhantomSDR-Plus
./install-stats-server.sh
# When prompted, use a temporary port like 9999
# After installation completes, copy the generated files
cp ~/sdr-stats-server/system-stats-server.js ~/your-manual-install-dir/
cp ~/sdr-stats-server/package.json ~/your-manual-install-dir/
```

ИЛИ создайте файлы вручную, используя последний код из скрипта установки.

5. Измените порт (при необходимости)

```
nano ~/sdr-stats-server/system-stats-server.js
```

Измените строку 6:

```
const PORT = process.env.PORT || 3001; // Change this number
```

6. Установите зависимости

```
npm install
```

7. Пробный запуск

```
npm start
```

8. Настройте службу (Опционально)

Следуйте инструкциям по настройке службы systemd из раздела автоматизированной установки.

Изменение порта после установки

Если вам нужно изменить порт после установки:

Шаг 1: Отредактируйте файл сервера

```
nano ~/sdr-stats-server/system-stats-server.js
```

Измените строку 6:

```
const PORT = process.env.PORT || 3001; // Change this number
```

Шаг 2: Обновите site_information.json

```
"siteStats": "http://Your_site_IP:NEW_PORT"
```

Шаг 3: Перезапустите службу

```
sudo systemctl restart sdr-stats.service
```

Шаг 4: Пересоберите ваше приложение Svelte

```
cd /path/to/phantomsdr-app
npm run build
```

Устранение неполадок

Проблема: Порт уже используется

Ошибка:

```
Error: listen EADDRINUSE: address already in use :::3001
```

Решение:

1. Найдите, что использует порт:

```
sudo lsof -i :3001
```

1. Остановите этот процесс:

```
kill <PID>
# or
sudo systemctl stop sdr-stats.service
```

1. Запустите службу снова:

```
sudo systemctl start sdr-stats.service
```

Альтернатива: Запустите скрипт установки снова и выберите другой порт по запросу.

Проблема: Служба не запускается

Ошибка:

```
status=217/USER
```

Решение:

1. Проверьте файл службы:

```
cat /etc/systemd/system/sdr-stats.service
```

1. Убедитесь, что строка `User=` соответствует вашему имени пользователя:

```
whoami
```

1. Отредактируйте при необходимости:

```
sudo nano /etc/systemd/system/sdr-stats.service
```

1. Перезагрузите и перезапустите:

```
sudo systemctl daemon-reload
sudo systemctl restart sdr-stats.service
```

Проблема: Статистика не отображается в браузере

Симптомы: Модальное окно открывается, но данных нет, или отображаются значения 0

Решения:

1. Проверьте консоль браузера (F12) на наличие ошибок
2. Убедитесь, что API доступен:

```
curl http://localhost:3001/api/system-stats
```

1. Проверьте, что служба запущена:

```
sudo systemctl status sdr-stats.service
```

1. Убедитесь, что `siteStats` в `site_information.json` соответствует адресу и порту вашего сервера
2. Проверьте наличие ошибок CORS — убедитесь, что сервер разрешает ваш домен

Проблема: Температура показывает null

Причина: Считывание температуры недоступно в вашей системе

Решения:

1. **Для большинства систем Linux:** Проверьте, существует ли тепловая зона:

```
cat /sys/class/thermal/thermal_zone0/temp
```

1. Установите lm-sensors:

```
sudo apt-get install lm-sensors
sudo sensors-detect
sensors
```

1. **Примечание:** Температура может быть недоступна на:
 2. Виртуальных машинах
 3. Некоторых VPS-провайдерах
 4. Windows Subsystem for Linux (WSL)
 5. Системах, отличных от Linux

Это нормально — остальная статистика по-прежнему будет работать!

Проблема: Значения статистики не совпадают с top

Причина: Период выборки ЦП был слишком коротким в старых версиях

Решение: Текущая версия использует выборку в 1 секунду для точных показаний. Если у вас старая установка:

1. Обновите файл сервера:

```
nano ~/sdr-stats-server/system-stats-server.js
```

1. Найдите строку 18 и измените:

```
}, 100); // Old value
```

на:

```
}, 1000); // New value (1 second)
```

1. Перезапустите:

```
sudo systemctl restart sdr-stats.service
```

Или просто повторно запустите скрипт установки, чтобы получить последнюю версию.

Проблема: Node.js не найден

Ошибка:

```
node: command not found
```

Решение:

1. Установите Node.js:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.4/install.sh | bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
nvm install 22 && nvm use 22 && nvm alias default 22
```

1. Проверьте:

```
node --version
npm --version
```

Что вы увидите

После установки и настройки модальное окно статистики отображает:

Раздел ЦП

- **Загрузка:** Общий процент использования ЦП
- **Ядра:** Сколько процессора реально используется, выраженное в целых ядрах — например `1.3 / 12 in use` означает, что машина выполняет работу 1,3 ядра. Величина точная и не требует никакого порога. Это намеренно не подсчёт занятых ядер: планировщик размазывает работу одного ядра по многим, поэтому такой подсчёт мог бы лишь приблизить это число.
- **Частота:** Средняя текущая частота ЦП по всем ядрам в ГГц, с самым быстрым ядром в скобках (если доступна)
- **Температура:** Температура ЦП в градусах Цельсия (если доступна), с цветовой маркировкой: зелёный до 69 °C, оранжевый 70-79 °C, красный от 80 °C и выше
- **Основные процессы:** 5 процессов, наиболее нагружающих ЦП, с цветовой маркировкой процентов:
 - Зелёный: до 50 % ЦП
 - Оранжевый: 51-80 % ЦП
 - Красный: выше 80 % ЦП

Раздел памяти

- **Использовано:** Используемая память (ГБ)
- **Всего:** Общий объём системной памяти (ГБ)
- **Использование:** Процент использования памяти (с цветовой маркировкой)

🔄Автообновление

- Статистика обновляется каждые **5 секунд**, пока открыто модальное окно
- Обновление прекращается при закрытии модального окна (экономит ресурсы)

Определение температуры

Сервер статистики использует продвинутое определение температуры несколькими методами:

Поддерживаемые системы:

- **Процессоры Intel:** Считывает с датчиков `coretemp` через `lm-sensors`
- **Процессоры AMD:** Считывает с датчиков `k10temp` (`Tdie/Tctl`)

- **Процессоры ARM:** Считывает из тепловых зон (Raspberry Pi и т. д.)

Требования:

- **Intel/AMD:** Требуется пакет `lm-sensors` (устанавливается автоматически скриптом)
- **ARM/Raspberry Pi:** Работает сразу, дополнительные пакеты не нужны

Приоритет установки:

1. Проверяет команду `sensors` для температуры Package/Core (Intel/AMD)
2. Переходит к прямому считыванию `coretemp` (Intel)
3. Переходит к `thermal_zone0` (ARM/Raspberry Pi)

Если температура показывает `null`, установите `lm-sensors`:

```
sudo apt-get install lm-sensors
sudo sensors-detect --auto
sudo systemctl restart sdr-stats.service
```

Сводка функций

Мониторинг в реальном времени - Статистика системы в реальном времени с вашего сервера **Метрики ЦП** - Загрузка, ядра, температура, основные процессы **Отслеживание памяти** - Использовано, всего и процент **Автообновления** - Обновляется каждые 5 секунд **Цветовая маркировка** - Визуальные индикаторы для предупреждений **Легковесность** - Минимальное использование ресурсов **Простая установка** - Автоматизированный скрипт всё сделает **Проверка порта** - Умный выбор порта с обнаружением конфликтов **Управление службой** - Автозапуск при загрузке

Примечания по безопасности

- Сервер статистики принимает подключения из любого источника (CORS: *)
- Для продакшена рассмотрите возможность ограничения CORS только вашим доменом
- Выбор порта проверяет ввод и наличие конфликтов
- Считывание температуры ЦП доступно только для чтения и безопасно
- Никакая конфиденциальная системная информация не раскрывается

Поддержка и ресурсы

Включённые файлы:

В этой папке (сам скрипт установки лежит в корне репозитория, как `PhantomSDR-Plus/install-stats-server.sh` - копия всегда одна):

- `package.json` - Зависимости, идентичные тем, что записывает скрипт
- Этот документ, доступный также на `de`, `el`, `es`, `fr`, `hr` и `ru`

Создаётся скриптом в `~/sdr-stats-server/`:

- `system-stats-server.js` - Бэкенд-сервер, созданный с выбранным вами портом

- `package.json` - Зависимости
- `node_modules/` - устанавливается командой `npm install`

Получение помощи:

- Ознакомьтесь с разделом устранения неполадок выше
- Просмотрите журналы службы: `sudo journalctl -u sdr-stats.service -n 50`
- Протестируйте API вручную: `curl http://localhost:3001/api/system-stats`
- Проверьте файлы конфигурации

Удаление сервера статистики:

- Удалите папку `/sdr-stats-server`
- Если вы создали службу, просто удалите её с помощью:

```
sudo rm /etc/systemd/system/sdr-stats.service`
```

- Измените `App.svelte`: Найдите часть кода:

```

        <!-- In case you don't want the Stats Button to appear, please comment this
button section (12 lines)-->
        <!-- System Stats Button -->
        <button
            type="button"
            class="glass-button text-white py-1 px-2 ml-2 rounded text-xs"
            on:click={openSystemStats}
            title="System Resources"
            aria-haspopup="dialog"
            aria-expanded={showSystemStats}
            aria-controls="system-stats-dialog"
            style="color:rgba(0, 225, 255, 0.993); font-size: 0.75rem;"
        >
            Stats
        </button>
    ...

```

и замените на:

```

        <!-- In case you don't want the Stats Button to appear, please comment this button
section (12 lines)-->
        <!-- System Stats Button -->
        <!--
        <button
            type="button"
            class="glass-button text-white py-1 px-2 ml-2 rounded text-xs"
            on:click={openSystemStats}
            title="System Resources"
            aria-haspopup="dialog"
            aria-expanded={showSystemStats}
            aria-controls="system-stats-dialog"
            style="color:rgba(0, 225, 255, 0.993); font-size: 0.75rem;"
        >
            Stats
        </button>
        -->
    ...

```

...

- Перекомпилируйте с помощью команды:

```

    cd PhantomSDR-Plus
    ./recompile.sh
    ```

- Перезагрузите ПК и запустите сервер

Автор: Создано для PhantomSDR-Plus
Версия: 1.1 (Улучшенное определение температуры ЦП для Intel/AMD) **Обновлено:** Январь 2026
Лицензия: MIT

Краткий справочник команд

```bash
# Installation
cd ~/PhantomSDR-Plus
chmod +x install-stats-server.sh
./install-stats-server.sh

# Service Management
sudo systemctl start sdr-stats.service      # Start
sudo systemctl stop sdr-stats.service       # Stop
sudo systemctl restart sdr-stats.service    # Restart
sudo systemctl status sdr-stats.service     # Status
sudo journalctl -u sdr-stats.service -f     # Logs

# Testing
curl http://localhost:3001/api/system-stats # Test API
curl http://localhost:3001/api/health      # Health check

# Troubleshooting
sudo lsof -i :3001                          # Check port
whoami                                       # Check username
node --version                              # Check Node.js

```

Приятного мониторинга!

Декодеры

DECODERS.md

PhantomSDR-Plus — Руководство пользователя по декодерам

Это руководство охватывает все встроенные декодеры PhantomSDR-Plus. Все декодеры включаются одинаково — порядок описан ниже, а затем приведены инструкции по настройке каждого из них.

Содержание

1. Как запустить декодер
2. FT8
3. FT4-FT2
4. JS8
5. CW — азбука Морзе
6. QRSS Grabber
7. WSPR
8. HF FAX / WEFAX
9. NAVTEX
10. FSK / RTTY — включая PSK31 и Olivia
11. SSTV
12. Общие советы

1. Как запустить декодер

Все декодеры находятся в разделе **Decoder Options** под органами управления спектрограммой звука на главной панели.

Порядок действий:

1. Нажмите кнопку **Decoder: OFF**, чтобы переключить её в **Decoder: ON** (в активном состоянии она становится синей).
2. Откройте выпадающий список справа от кнопки и выберите нужный декодер.
3. Панель выбранного декодера появится под органами управления — следуйте инструкциям для этого декодера в соответствующем разделе руководства.
4. Чтобы остановить декодирование, выберите — **Select decoder** — в выпадающем списке или нажмите кнопку **Decoder: ON**, чтобы вернуть её в положение OFF.

Кнопки декодеров в одно нажатие

Выпадающий список — не единственный путь. На главной панели есть ряд кнопок **Decoders** — сразу под **Wheel Tuning Steps** — по одной кнопке на каждый декодер:

Кнопка	Декодер	Кнопка	Декодер
FT8	FT8	SSTV	SSTV
FT4	FT4	NAVTEX	NAVTEX
FT2	FT2	RTTY	FSK / RTTY
CW	CW		
WSPR	WSPR	FAX	HF FAX / WEFAX

Нажатие кнопки выполняет всю последовательность сразу: выбирает декодер, переводит декодер в ON и прокручивает его окно в поле зрения. Кнопка остаётся синей, пока её декодер работает. **Повторное нажатие той же кнопки выключает декодер** — его окно закрывается вместе с ним.

RADEL и **RADEU** намеренно отсутствуют в этом ряду. Это режимы цифрового голоса, а не текстовые декодеры, поэтому у них своя пара кнопок рядом с заголовком **Modes selector**, а также внутри всплывающих окон **Modes** и **Bands**. Работают они точно так же: нажатие — старт, повторное нажатие — стоп. См. [руководство по RADE](#).

Кнопки, выпадающий список и кнопка ON/OFF управляют одним и тем же состоянием, поэтому чем бы вы ни воспользовались, остальное следует за ним.

Одновременно может быть активен только один декодер. Переключение на другой декодер автоматически останавливает предыдущий.

QRSS-граббер не входит в этот список — у него свой раздел **QRSS**, и он может работать одновременно с декодером.

Декодирование выполняется в фоне. SSTV, HF FAX, NAVTEX, FSK/RTTY и CW работают каждый в отдельном потоке Web Worker, поэтому декодирование никогда не конкурирует с воспроизведением звука или водопадом. Запуск или остановка декодера не прерывает звук, а интерфейс остаётся отзывчивым во время приёма изображения или страницы. Все декодеры получают исходный звук, взятый до АРУ, шумоподавления и отключения звука, — поэтому отключение звука приёмника или изменение этих настроек под себя не влияет на декодирование.

Работающий декодер владеет режимом и полосой пропускания. Обычно режим следует за планом диапазонов из `bands-config.js`: переведите настройку в участок, помеченный LSB или AM, и приёмник переключится. Пока работает декодер, этого больше не происходит. Декодер удерживает нужный ему режим (USB у большинства, собственный у RADE) и нужную ему полосу — PSK31 около ± 100 Гц, Olivia всю свою ширину, RTTY свой сдвиг — и то и другое переживает перестройку, включая переход на другой диапазон. Без этого FT8 на 40 м переключал бы приёмник в LSB, как только сдвинется настройка, а узкие полосы декодеров снова расширялись бы до полного SSB-фильтра.

Собственный режим диапазона возвращается в тот момент, когда вы выключаете декодер. Задать режим вручную можно в любой момент: кнопки режимов — осознанный выбор и всегда побеждают. Декодер CW — исключение из всего этого: он декодирует в том режиме, в котором вы слушаете, и никогда не забирает приёмник себе.

Decoder ID — автоматическое распознавание вида модуляции

Над кнопками декодеров расположено поле **Decoder ID**. Оно отвечает на вопрос «что я слушаю?», когда вам попадается цифровой сигнал, который вы не можете определить, и затем предлагает подходящий декодер одним щелчком.

По умолчанию оно **выключено** и в выключенном состоянии ничего не потребляет. Нажмите **On** и оставьте сигнал настроенным. Около полуминуты оно показывает *Listening...* с полосой выполнения — измерениям времени нужно увидеть два полных цикла FT8, чтобы иметь смысл, — а затем называет вид модуляции:

Decoder ID [On] NAVTEX / SITOR-B 99% 170 Hz shift · 100.00 Bd · 43 dB S/N [Use NAVTEX]

- **Вид модуляции** и показатель **уверенности**. Зелёный выше 75 %, жёлтый выше 50 %, оранжевый ниже — оранжевый ответ следует считать подсказкой, а не приговором.
- **Что было измерено на самом деле**: занимаемая полоса, разнос тонов, символьная скорость, длительность посылки, сетка UTC, которой сигнал соответствует, отношение сигнал/шум. Это доказательства, стоящие за выводом, чтобы вы могли судить сами.
- **Use <вид>** запускает нужный декодер точно так же, как если бы вы нажали его кнопку, — и настраивает приёмник на работу этим видом. Для видов, живущих в панели FSK, дополнительно выбирается правильный вариант — RTTY, Weather RTTY, PSK31 или Olivia.
- Следующие кандидаты перечислены после **or**. Щелчок пробует его вместо первого, так что неверный первый ответ вам ничего не стоит.
- ******** означает, что частота, на которую вы настроены, является известной частотой вызова для этого вида, и что это учтено в ранжировании.

Оно никогда не переключает ваш декодер само. Оно только предлагает. А как только вы запускаете декодер — из ряда кнопок, из выпадающего списка или собственной кнопкой **Use** в Decoder ID — Decoder ID выключается сам: его задачей было выбрать декодер, а теперь этот декодер требует тот же звук.

Оно настраивает приёмник под этот вид модуляции. Выбрать вид модуляции — значит решить им работать, поэтому приёмник следует за выбором: настроенная частота встаёт в середину водопада, а обзор становится около 100 кГц вокруг неё, боковая полоса становится той, на которой этим видом работают, а полоса пропускания сужается до окна, в котором этот вид живёт — вся подполоса 3 кГц для FT8, FT4, FT2 и JS8, 1350–1650 Гц для WSPR, 250–750 Гц для NAVTEX, 800–2700 Гц для HF FAX, 900–2600 Гц для SSTV и ± 250 Гц по обе стороны от настройки для CW, при этом приёмник дополнительно переключается в CW. Узкая полоса не пускает в декодер соседние сигналы — ровно то, что вы сделали бы вручную. Два исключения: SSTV сохраняет боковую полосу, которую выбрал сам, — по традиции LSB на 80 и 40 м, где принудительный USB перевернул бы соответствие тонов, — а виды FSK сохраняют окно, которое их собственная панель выводит из сдвига или полосы варианта и которое уже любого фиксированного значения. Нажатие **Use** на уже работающем декодере по-прежнему выключает его, и приёмник тогда остаётся ровно таким, каким был.

Медленным режимам нужно больше времени. WSPR работает по двухминутной сетке, а картинки SSTV идут минуту-другую, поэтому эти два режима называются позже остальных — дайте им три-четыре минуты на частоте. Кроме того, HF FAX и SSTV распознаются только при обычной настройке — полоса изображения должна лежать примерно в 1500–2300 Гц по звуку, там же её ждут и их декодеры. Смена частоты начинает измерение заново, потому что всё собранное до этого описывает частоту, которую вы только что покинули.

Что оно распознаёт: FT8, FT4, JS8, WSPR, CW, NAVTEX/SITOR-B, RTTY на 45,45 Бод, Weather RTTY (DWD), PSK31, Olivia, HF FAX, SSTV и DSC. В этой сборке нет декодера DSC, поэтому DSC называется и помечается как *no decoder*.

Recentre & retry. Положение сигнала внутри звуковой полосы пропускания не имеет значения примерно между 400 Гц и 2700 Гц. За пределами этого диапазона измерение ухудшается, и появляется жёлтая кнопка **Recentre & retry**. Нажатие сдвигает настройку так, чтобы сигнал

оказался в середине полосы пропускания, и начинает новое измерение. Это одно рассчитанное перемещение, а не поиск.

Три честных ограничения:

- **FT8, JS8 и FT2 — это один и тот же сигнал.** Та же модуляция 8-FSK, тот же разнос тонов, а JS8 на скорости Normal использует 15-секундный такт FT8. Ничто в самом сигнале их не разделяет. Разделяет частота: на частоте вызова FT8 сообщается FT8, на частоте JS8 — JS8, а в любом другом месте честно сообщается семейство — *FT8 / JS8 / FT2*. Запустите любой из этих декодеров и посмотрите, какой выдаёт текст.
- **Оно молчит, а не гадает.** Ниже примерно 10 дБ отношения сигнал/шум оно сообщает *Nothing above the noise* или *Not recognised*, вместо того чтобы назвать вид модуляции, который на самом деле не видит. Это сделано намеренно: уверенный неправильный ответ хуже, чем отсутствие ответа.
- **Замирающий сигнал читается иначе.** WSPR передаёт 110 секунд без перерыва — дольше, чем длится обычный провал QSB, поэтому глубокое замирание разрезает передачу на куски, и её двухминутный ритм по ним уже не прочесть. Тогда он восстанавливается другим путём: вся история прослушивания сопоставляется с двухминутной сеткой, а не с отдельными передачами, и в строке измерений появляется *120 s grid (through fading)*. Это более слабое свидетельство, чем услышанная целиком передача, поэтому показанная уверенность намеренно ниже. Замирание, достаточно глубокое, чтобы утопить сигнал в шуме, не восстанавливается вовсе. Olivia строже прочих: её не назовут, пока не измерена символьная скорость, поэтому она умолкает раньше.

2. FT8

Что это: FT8 — популярный цифровой вид связи слабыми сигналами, используемый радиолюбителями по всему миру. Передачи длятся ровно 15 секунд, а протокол способен принимать сигналы на 20-25 дБ ниже уровня шума. Это самый распространённый вид связи для дальних (DX) связей.

Рекомендуемые частоты (USB)

Диапазон	Частота
160 м	1.840 МГц
80 м	3.573 МГц
40 м	7.074 МГц
30 м	10.136 МГц
20 м	14.074 МГц
17 м	18.100 МГц
15 м	21.074 МГц
12 м	24.915 МГц
10 м	28.074 МГц

Настройка

1. Настройтесь на одну из перечисленных выше частот FT8 и установите режим **USB**.
2. Включите декодер и выберите **FT8** в выпадающем списке. Или просто нажмите кнопку **FT8**.

3. Панель **FT8 Messages** появится автоматически ниже.

Чтение результатов

Список сообщений показывает декодированные передачи по мере их поступления. Каждый 15-секундный цикл даёт новую порцию сообщений. Поле **Farthest** (вверху справа панели) показывает наибольшее расстояние, декодированное в текущей сессии, в километрах.

Типичный формат сообщения: `CQ DX AA1BB FN31` — общий вызов от позывного AA1BB, расположенного в квадрате FN31.

SNR рядом с каждым декодированием — это настоящее отношение сигнал/шум, измеренное на декодированном сигнале относительно окружающего шума и приведённое к полосе 2500 Гц, как это делает WSJT-X. Слабые станции показывают отрицательные значения, примерно до -20 дБ, сильные — положительные; на одних и тех же записях обе программы в среднем совпадают, обычно в пределах около 3 дБ для отдельного декодирования. FT4 и FT2 используют тот же метод.

После обновления: декодер FT8/FT4/FT2 — отдельный файл, который браузеры долго хранят в кэше. Если все значения SNR по-прежнему лежат примерно между 0 и +5 дБ, перезагрузите страницу полностью (**Ctrl+Shift+R**, или **Cmd+Shift+R** на Mac).

Совет: FT8 жёстко синхронизирован по времени. Браузер использует часы вашего компьютера; если системное время уходит более чем на пару секунд, декодирование не удастся. Держите системное время синхронизированным по NTP.

3. FT4-FT2

Что это: FT4 — более быстрая разновидность FT8, рассчитанная на работу в соревнованиях. Каждый цикл передачи длится 7,5 секунды (половина от FT8), что делает его вдвое быстрее, но требует чуть более сильного сигнала. FT2 — ещё более быстрая разновидность FT8. Это сверхбыстрый 77-битный режим с периодами приёма/передачи 3,75 секунды (= половина FT4) — пока экспериментальный.

Рекомендуемые частоты (USB)

Диапазон	Частота FT4
80 м	3.575 МГц
40 м	7.047 МГц
30 м	10.140 МГц
20 м	14.080 МГц
15 м	21.140 МГц
10 м	28.180 МГц

Диапазон	Частота FT2
160 м	1.843 – 1.846
80 м	3.578 – 3.581
60 м	5.360 (проверьте местные региональные правила)
40 м	7.052 – 7.062
30 м	10.144
20 м	14.084
17 м	18.108
15 м	21.144
12 м	24.923
10 м	28.184

Настройка

1. Настройтесь на частоту FT4 и установите режим **USB**.
2. Включите декодер и выберите **FT4** или **FT2** в выпадающем списке. Или просто нажмите кнопку **FT4 / FT2**.
3. Панель **FT4 Messages** или **FT2 Messages** появится ниже; её компоновка совпадает с панелью FT8.

Примечание: FT4 и FT8 используют разные спектральные форматы и не взаимозаменяемы. Убедитесь, что вы находитесь на частоте FT4, когда пользуетесь этим декодером.

4. JS8

Что это: JS8 (режим, используемый JS8Call) берёт механизм слабого сигнала FT8 и превращает его в режим разговора с клавиатуры на клавиатуру. Если FT8 передаёт фиксированные посылки по 13 символов, то JS8 передаёт свободный текст небольшими порциями и снова собирает последовательные передачи в целые фразы — поэтому всё, что длиннее примерно двенадцати символов, приходит за несколько 15-секундных циклов. По чувствительности он близок к FT8, поэтому работает там, где голос и CW уже не проходят.

Рекомендуемые частоты (USB)

Диапазон	Частота
160m	1.842 MHz
80m	3.578 MHz
40m	7.078 MHz
30m	10.130 MHz
20m	14.078 MHz
17m	18.104 MHz
15m	21.078 MHz
12m	24.922 MHz
10m	28.078 MHz

Скорости

У JS8 пять скоростей. Все станции в разговоре должны использовать одну и ту же. **Normal** — вызывная скорость, на неё приходится почти весь трафик: начинайте с неё.

Скорость	Цикл	Полоса	Когда применяется
Slow	30 с	25 Hz	Очень слабые трассы; самая чувствительная
Normal	15 с	50 Hz	Обычная вызывная скорость
Fast	10 с	80 Hz	Более быстрый обмен, нужен сигнал сильнее
Turbo	6 с	160 Hz	Сильные местные сигналы
Ultra	4 с	250 Hz	Экспериментальная, встречается редко

Настройка

1. Настройтесь на частоту JS8 и включите режим **USB**.
2. Включите декодер и выберите **JS8** из списка либо просто нажмите кнопку **JS8**.
3. Ниже появится панель **JS8 Decoder**. Оставьте **Speed** на *Normal*, если только не знаете, что нужная станция работает на другой.

Как читать вывод

В панели два списка.

Сообщения, которые ещё принимаются, показываются сверху зелёным, с мигающим курсором. Сообщение JS8 может занять четыре цикла — на Normal это целая минута, — поэтому здесь видно, как фраза собирается. Если оно задержалось, ничего не сломалось.

Завершённые сообщения заполняют основной список, по строке на каждое:

Столбец	Значение
Mode	Всегда JS8
Hz	Звуковая частота сигнала внутри полосы пропускания
dB	Отношение сигнал/шум в опорной полосе 2500 Hz
Message	Раскодированный текст; позывные показаны зелёным

Типичные сообщения:

- SV1BTL KM17: HB — heartbeat: станция сообщает, что она в эфире, с указанием квадрата.
- KN4CRD: K0OG SNR -05 — направленное сообщение: KN4CRD сообщает K0OG, что принимает его на -5 dB.
- MP 100W 8M/BALUN JN58KH AUGSBURG, MARTIN — свободный текст, здесь описание станции, пришедшее за четыре цикла.

Строка, показанная **приглушённо и курсивом**, истекла по времени до прихода последнего кадра. Текст настоящий, но может быть обрезан.

Sync offset

Ползунок **Sync offset** задаёт, через сколько после границы цикла UTC начинается запись, компенсируя задержку звукового тракта. Оставьте **Auto** включённым: декодер измеряет тайминг принимаемых сигналов и подстраивается сам.

Метки спектра

Небольшая полоса спектра в панели рисует жёлтую вертикальную линию на частоте каждого сигнала, раскодированного в предыдущем цикле, так что столбец Hz читается прямо с экрана.

Посмотреть собственные споты

Если приёмник отправляет споты JS8 на PSK Reporter (см. вкладку Spot Reporting в панели администратора), кнопка **JS8 map** в блоке **Note:** открывает карту PSK Reporter, отфильтрованную по позывному этого приёмника и режиму JS8 — то же самое, что делают **FT8 map** и **FT4 map** для своих режимов.

Совет: JS8 намного тише, чем FT8. Днём одна передача в несколько минут — это нормально, как и долгие паузы вообще без сигналов. Дайте ему пять-десять минут, прежде чем решать, что что-то не так. Обычные места — 20m (14.078) и 40m (7.078).

Совет: Как и FT8, JS8 требует точной синхронизации по времени и использует часы вашего компьютера. Если системное время уходит больше чем на секунду-две, ничего не раскодировается. Держите его синхронизированным по NTP.

5. CW — азбука Морзе

Что это: декодер CW слушает сигналы азбуки Морзе (незатухающие колебания) и преобразует их в текст в реальном времени. Он автоматически отслеживает частоту сигнала и подстраивается под скорость передачи оператора.

Рекомендуемые частоты

CW работает во всех любительских диапазонах, обычно в нижней части каждого. Типичные участки:

Диапазон	Участок
40 м	7.000–7.040 МГц
20 м	14.000–14.070 МГц
15 м	21.000–21.080 МГц

Настройка

1. Настройтесь на сигнал CW, используя режим демодуляции **CW** или **CW-L** по обстановке.
2. Включите декодер и выберите **CW** в выпадающем списке. Или просто нажмите кнопку **CW**.
3. Панель **CW Decoder** появится ниже.

Чтение результатов

- В заголовке панели указаны обнаруженная частота сигнала в Гц (например, ≈ 700 Hz) и оценка скорости передачи в словах в минуту (например, $\cdot 22$ WPM).
- Если сигнал не обнаружен, в заголовке отображается **scanning...**
- На пустом канале ничего не печатается, скорость и тон не показываются. Декодер начинает печатать только при явной разнице между нажатым и отпущенным ключом, которой чистый шум не даёт, а у сигнала, ещё читаемого около -10 дБ, её с избытком.
- Декодированный текст прокручивается янтарным моноширинным шрифтом. Мигающий курсор (█) отмечает место, куда сейчас пишется текст.
- Нажмите **Clear**, чтобы очистить буфер вывода.

Советы: - Центрируйте полосу пропускания на тоне CW. Декодер работает лучше всего, когда сигнал CW находится примерно между 400 и 900 Гц в звуковом спектре. - Принимаются скорости примерно от 8 до 40 WPM, в том числе передача Farnsworth (знаки передаются быстро, с увеличенными паузами между ними): захват скорости держится, и буквы не распадаются на отдельные слова. Сильно ручная (неравномерная) передача всё ещё может снижать точность. - Декодер лучше всего справляется с одним чистым сигналом. Сильное QRM от соседних сигналов в том же диапазоне может сбить его с толку.

6. QRSS Grabber

Что это: QRSS — это CW, передаваемый настолько медленно, что одна точка длится секунды, а не миллисекунды. На такой скорости сигнал занимает лишь доли герца, поэтому достаточно узкий анализ вытаскивает след с 20–30 дБ ниже уровня шума. На слух здесь читать нечего — QRSS **смотрят**, а не слушают. Граббер считает собственное очень длинное БПФ по звуку приёмника и рисует классическую картину граббера: частота по вертикальной оси, время бежит слева направо, самый свежий столбец всегда у правого края.

Основное, что вы увидите, — маяки MEPT (Manned Experimental Propagation Transmitter): передатчики очень малой мощности, часто несколько сотен милливатт в кусок провода, непрерывно опознающие себя очень медленным Морзе или узорами FSK.

Где искать

Маяки QRSS живут в узких окнах 100–200 Гц у нижнего края каждого диапазона. Запоминать их не нужно: у граббера есть список **Band** с приведёнными ниже окнами, и выбор одного из них настраивает приёмник на него и готовит его к работе (см. *Как открыть граббер*).

Диапазон	Центр окна	Примечание
30m	10 140,00 кГц	Основное окно QRSS — самое оживлённое, днём и ночью
40m	7 039,90 кГц	
40m	7 000,85 кГц	Окно Knights, 7 000,8–7 000,9 кГц
20m	14 096,90 кГц	
80m	3 569,90 кГц	
80m	3 568,60 кГц	Более старое окно
80m	3 500,85 кГц	Окно Knights, 3 500,8–3 500,9 кГц
160m	1 837,90 кГц	
160m	1 843,30 кГц	Более старое окно
630m	476,10 кГц	
2200m	137,70 кГц	QRSS / DFCW на НЧ
60m	5 288,55 кГц	
17m	18 105,90 кГц	
15m	21 095,90 кГц	
12m	24 925,90 кГц	
10m	28 125,70 кГц	
10m	28 000,85 кГц	Более старое окно
10m	28 322,00 кГц	Альтернативное
6m	50 294,30 кГц	

Несколько диапазонов встречаются дважды, потому что в ходу действительно две традиции. Современные окна лежат на 200 Гц ниже частоты WSPR этого диапазона; старые окна *Knights* находятся совсем в другом месте, а на 40 и 80 м — заметно ниже. Если на одном диапазон тих, попробуйте другое.

Это соглашения, а не правила, и часть из них региональна — особенно расходятся значения для 10 м. Считайте список отправной точкой и следуйте местной практике.

Как открыть граббер

QRSS-граббер **не** находится в списке декодеров. У него свой раздел **QRSS**, рядом с органами управления спектрограммой и декодерами.

1. Нажмите **Show**. Появится холст граббера, а рядом с кнопкой раскроются его органы управления.
2. Выберите окно в списке **Band** и нажмите **Tune**. Это одним действием делает всё, что нужно режиму: настраивает на окно, переводит приёмник в **CW** и задаёт полосу ровно такой ширины, какая нужна для наблюдаемого среза. Шкала после этого показывает настоящую частоту QRSS, а следы попадают на центральную линию дисплея.
3. Установите **Speed** по длительности точки маяка.
4. **Centre** и **Span** работают как прежде; изменение любого из них перестраивает и полосу, так что дисплей и приёмник остаются согласованными.
5. Нажмите **Hide** для остановки. Обычный режим диапазона возвращается.

Всё это по-прежнему можно сделать и вручную — настроиться в **USB** примерно на 1 кГц ниже окна, чтобы следы попали около 800 Гц звука. Список **Band** лишь избавляет от арифметики.

Пока настроено окно из списка, граббер удерживает приёмник так же, как декодер: CW и его полоса переживают перестройку, а окно следует за настройкой, поэтому след остаётся на центральной линии, пока вы обшариваете диапазон. Приёмник возвращается, когда вы скрываете граббер, нажимаете кнопку режима или запускаете декодер.

Скорости

Настройка	Длительность точки	Разрешение	Окно анализа	Новый столбец каждые
QRSS 3	3 с	0,73 Гц	1,4 с	0,7 с
QRSS 6	6 с	0,37 Гц	2,7 с	1,4 с
QRSS 10	10 с	0,18 Гц	5,5 с	2,7 с
QRSS 30	30 с	0,09 Гц	10,9 с	5,5 с
QRSS 60	60 с	0,05 Гц	21,8 с	10,9 с

Настройка быстрее маяка тратит чувствительность впустую — след получается тонким и шумным. Настройка медленнее маяка размазывает соседние точки и тире в одну полосу. Панель открывается на **QRSS 6** — разумной отправной точке для незнакомого сигнала: чувствительнее, чем QRSS 3, при этом обновляется быстрее и терпимее к уходу частоты, чем QRSS 10. Как только станет виден рисунок манипуляции, перейдите на ту скорость, которую маяк действительно передаёт: каждый шаг в сторону более быстрой настройки стоит 3 дБ чувствительности.

Органы управления

Элемент	Что делает
Band	Окна QRSS из таблицы выше. Tune переходит на выбранное, в CW и с подходящей полосой.
Speed	Задаёт длину преобразования — компромисс между разрешением по частоте и по времени (таблица выше).
Centre	Звуковая частота в середине дисплея, 100–3000 Гц.
Span	Высота отображаемой полосы: 20, 50, 100 или 200 Гц, при запуске 100 Гц — ширина поддиапазона QRSS. Более узкая полоса растягивает каждый след на большее число пикселей. При широких полосах в панель попадает больше бинов, чем у неё строк пикселей; каждая строка тогда показывает самый сильный бин, который она покрывает, так что ничто не может спрятаться между строками.
Gain	От –10 до +40 дБ. Сдвигает цветовую шкалу относительно измеренного уровня шума; поднимите, чтобы высветлить слабые следы.
Color	Rainbow, Green или Grayscale.
Clear	Очищает холст и сбрасывает опорный уровень шума.

Как читать дисплей

Метки частоты идут по левому краю, самая высокая частота — сверху. Под холстом строка состояния показывает выбранную скорость, реально используемое разрешение (например $0.183 \text{ Hz/bin} \cdot 5.5 \text{ s window}$) и темп столбцов (например 2.7 s/column).

Цветовая шкала привязана к **текущему** шуму, который непрерывно измеряется по пустым участкам каждого столбца, а не к абсолютному уровню. Поэтому смена громкости, АРУ, антенны или диапазона не требует заново подбирать Gain — дисплей сохраняет постоянный контраст

относительно того шума, какой есть в эфире. Как и у декодеров, звук берётся до АРУ, шумоподавления и приглушения, так что приёмник можно заглушить и продолжать смотреть.

Что означают формы:

- **Устойчивая несущая** рисует прямую горизонтальную линию.
- **Медленный Морзе** рисует эту линию цепочкой коротких и длинных отрезков — точки и тире читаются слева направо.
- **Изогнутые или уплывающие следы** — это нестабилизированный генератор маяка, который греется или остывает. Это нормально, и часто именно такая подпись ухода частоты позволяет постоянному наблюдателю узнать станцию.
- **Вертикальные полосы** через всю полосу — атмосферные разряды или местные всплески шума, а не сигнал.

Советы: - Запаситесь терпением. При QRSS 30 один столбец занимает 5,5 секунды, так что полный позывной может пересекать экран десять минут и дольше. Пусть работает. - Настройка из списка **Band** уже сужает полосу вокруг окна. Если настраивались вручную, сделайте это сами: на то, что разрешает преобразование, это не влияет, но не даёт сильным соседям управлять АРУ приёмника. - Граббер не зависит от декодеров — его можно оставить включённым, пока декодер занят чем-то другим. - Широкий span на медленной скорости — самая тяжёлая комбинация; всё преобразование выполняется в браузере, поэтому на скромной машине лучше выбрать span 50 Гц.

7. WSPR

Что это: WSPR (Weak Signal Propagation Reporter, произносится «уиспер») — маячковый режим сверхслабых сигналов, картирующий пути КВ-распространения по всему миру. Каждая передача длится около 110 секунд и укладывается в полосу шириной 200 Гц. Декодер ждёт полного двухминутного интервала, выровненного по UTC, и только затем декодирует.

Рекомендуемые частоты (USB, по шкале)

Диапазон	Частота по шкале
160 м	1.836.600 МГц
80 м	3.568.600 МГц
40 м	7.038.600 МГц
30 м	10.138.700 МГц
20 м	14.095.600 МГц
17 м	18.104.600 МГц
15 м	21.094.600 МГц

Настройка

1. Настройтесь на одну из указанных выше частот WSPR и установите режим **USB**.
2. Сигнал WSPR занимает звуковой диапазон 1400-1600 Гц. Выбор **WSPR** делает остальное за вас: приёмник переключается в USB, а полоса пропускания сужается до 1350-1650 Гц, что покрывает весь диапазон поиска декодера. Дополнительно подстраивать ничего не нужно.

3. Включите декодер и выберите **WSPR** в выпадающем списке. Или просто нажмите кнопку **WSPR**.
4. Панель **WSPR-2 Decoder** появится ниже.

Чтение результатов

Панель показывает индикатор выполнения для текущего двухминутного интервала:

- **Голубая полоса заполняется** — идёт сбор данных сигнала (0-116 с внутри интервала).
- **Янтарная полоса пульсирует** — идёт декодирование (последние ~4 с интервала).
- **Пустая полоса** — ожидание следующей чётной минуты UTC.

Каждый успешно декодированный спот выводится в таблицу со следующими столбцами:

Столбец	Значение
UTC	Время спота (чётная минута)
Callsign	Передававшая станция
Grid	Локатор Maidenhead передатчика
Power	Мощность передачи в дБм
Freq	Точная звуковая частота (Гц) внутри полосы WSPR
SNR	Отношение сигнал/шум в дБ

Нажмите **Clear**, чтобы очистить список спотов.

Совет: декодирование WSPR требует очень точного системного времени (в пределах ± 1 секунды от UTC). Первый интервал после включения декодера начнётся со следующей чётной минуты UTC — короткое ожидание нормально.

8. HF FAX / WEFAX

Что это: КВ-радиофакс (также известный как WEFAX) используется береговой охраной и метеослужбами по всему миру для передачи карт погоды, карт состояния моря и приземного анализа на коротких волнах. Декодер восстанавливает изображение строка за строкой по мере приёма.

Настройка

1. Включите декодер и выберите **HF FAX / WEFAX** в выпадающем списке. Или просто нажмите кнопку **FAX**.
2. Панель **HF FAX / WEFAX Receiver** появится ниже.
3. **Выберите станцию** в списке Station. Доступно более 20 станций, охватывающих Европу, Азию, Океанию и обе Америки (например, DDH3/DDK3 Германия, SVJ4/GR Греция, JMH Япония, NMG США, Новый Орлеан). Есть и CBM Punta Arenas (Чили). Расписания соответствуют мировому расписанию радиофакса NOAA (издание от 7 марта 2025 года); для Charleville (VMC, Австралия) добавлены его дополнительные передачи только на 11030 кГц.
4. Если станция вещает на нескольких частотах, выберите нужную во вложенном списке **Frequency**.
5. Нажмите ► **Tune**, чтобы автоматически настроить водопад на эту станцию.
6. Режим автоматически принудительно устанавливается в **USB**.

Расписание передач

При выборе станции появляется таблица обратного отсчёта **Next Transmissions**, показывающая 4 ближайшие запланированные передачи по UTC с живым отсчётом:

- Обычный (серый/зелёный) — передача предстоит.
- **Янтарный** — передача начнётся менее чем через 3 минуты; подготовьте декодер.
- **Красный** ● — передача начнётся менее чем через 30 секунд; приём вот-вот начнётся.

Параметры

Большинство станций используют стандартные значения по умолчанию (отмечены ★). Меняйте их только если знаете, что станция использует нестандартные настройки.

Параметр	По умолчанию	Описание
LPM	120 ★	Строк в минуту — скорость вращения барабана
IOC	576 ★	Индекс кооперации — определяет число пикселей в строке
Shift	800 Гц ★	Частотный сдвиг между тонами чёрного и белого

Органы управления

- ⇄ **Auto-align** — включено по умолчанию. Автоматически синхронизируется по фазирующему сигналу в начале каждого изображения. Отключайте только если испытываете проблемы с выравниванием на заведомо хорошем сигнале.
- ↕ **Invert** — меняет местами чёрное и белое. Используйте, если изображение выглядит как негатив (белые области там, где должно быть чёрное).
- ↻ **Refresh** — очищает холст и сбрасывает декодер. Используйте между передачами или если изображение рвётся или уплывает.
- ↓ **Save PNG** — сохраняет текущий холст в файл PNG на ваш компьютер.

Индикаторы состояния

Внизу изображения два тональных индикатора показывают:

- **300 Hz phasing** — загорается голубым при обнаружении фазирующего тона начала изображения.
- **450 Hz stop** — загорается красным при обнаружении тона окончания изображения.

Оба сигнала распознаются по их ритму по всей ширине строки, которую настоящие сигналы начала и конца заполняют всегда, а содержимое карты практически никогда, поэтому заштрихованные области карты не принимаются за сигнал начала. Они распознаются примерно до 0 дБ SNR, а на очень слабых сигналах изображение выравнивается по среднему нескольким строк фазирования, так что карта всё равно начинается у левого края.

Примечание: изображение прокручивается вверх — самая свежая принятая строка всегда внизу холста. Если в заголовке вы видите **[PHASING]**, декодер захватил начало нового изображения.

9. NAVTEX

Что это: NAVTEX — международная морская система передачи информации по безопасности в прибрежных водах: навигационные предупреждения, прогнозы погоды и сообщения о поисково-

спасательных операциях. Используется FSK со скоростью 100 бод (SITOR-B с FEC), приём ведётся на выделенных каналах по всему миру.

Доступные каналы

Канал	Частота	Назначение
Международный	518 кГц	На английском языке, международный
Национальный	490 кГц	Передачи на национальном языке
KB (×5)	4209.5 / 6314 / 8416.5 / 12579 / 16806.5 кГц	Дальний KB NAVTEX

Настройка

1. Включите декодер и выберите **NAVTEX** в выпадающем списке. Или просто нажмите кнопку **NAVTEX**. Приёмник автоматически переключится в **USB**.
2. Появится панель **NAVTEX Receiver**.
3. Выберите нужный канал в списке **Station** (например, `International – 518 kHz`).
4. Нажмите ⇒ **Tune & Set IF**, чтобы автоматически настроить водопад и сузить полосу пропускания до нужного звукового окна. Режим автоматически устанавливается в **USB**. Частота по шкале ставится на 500 Гц ниже центра канала, так что сигнал NAVTEX оказывается на 500 Гц в звуке.

Расписание передач

Таблица расписания перечисляет все известные станции выбранного канала с указанием:

- буквенного идентификатора МСЭ и флага страны
- времени следующей передачи по UTC
- живого обратного отсчёта до следующей передачи
- **Янтарный** — менее 2 минут / **Красный** ● — менее 30 секунд; подготовьте декодер

Чтение результатов

Декодированный текст выводится бирюзовым моноширинным шрифтом. Границы сообщений явно обозначены:

```
— ZCZC MA12 —
... message content ...
— NNNN —
```

ZCZC отмечает начало сообщения. Четыре следующих символа обозначают станцию (M), тему (A = навигационные предупреждения) и порядковый номер (12). Баннер дожидается полного заголовка, поэтому показывает **ZCZC MA12**, а не **ZCZC ????**. **NNNN** отмечает конец.

Если вы настроились, когда сообщение уже началось, или его заголовок потерялся в шуме, текст всё равно появляется — под баннером — **(header missed)** —. Он открывается только тогда, когда подряд приходят несколько символов, в которых совпадают обе передачи NAVTEX, чего шум практически никогда не даёт. Закрывается он как обычно на **NNNN**, с **— (signal lost) —**, если сигнал пропадёт раньше, или когда начинается новое сообщение.

Нажмите **Clear**, чтобы очистить буфер сообщений.

Примечание: NAVTEX работает на частотах СЧ и НЧ (518/490 кГц). Дальность приёма обычно 200-400 морских миль от передатчика. KB-каналы (4-17 МГц) обеспечивают значительно большую дальность.

10. FSK / RTTY — включая PSK31 и Olivia

Что это: универсальный декодер узкополосных текстовых видов связи с пятью вариантами работы, выбираемыми из одного меню. Три из них — настоящая частотная манипуляция (FSK): морской FSK (SITOR), метеорологический RTTY и любительский RTTY. Два других вовсе не FSK, но используют то же окно: **PSK31** — фазовая манипуляция, и **Olivia** — многотональная FSK с прямой коррекцией ошибок. Каждый вариант снабжён предустановкой, настроенной на его стандартные параметры.

Варианты и предустановки

Вариант	Центр	Сдвиг	Бод	Кадр	Кодировка
Морской FSK / SITOR	500 Гц	170 Гц	100	7N1	CCIR-476
Метеорологический RTTY	1000 Гц	450 Гц	50	5N1.5	ITA2
Любительский RTTY	1000 Гц	170 Гц	45.45	5N1.5	ITA2
PSK31 (BPSK)	1000 Гц	—	31.25	—	Varicode
Olivia (MFSK)	1000 Гц	—	см. Mode	—	7 бит + FEC

Панель подстраивается под выбранный вариант. **Shift**, **Baud**, **Framing**, **Encoding**, **Invert mark / space** и **Auto shift detect** скрываются для PSK31 и Olivia, поскольку ни у одного из этих видов связи нет пары тонов mark/space и кадрирования в стиле UART. Вместо этого **Center audio** превращается в поле свободного ввода числа (несущая может находиться в любом месте полосы пропускания), а Olivia получает переключатель **Mode** и ползунок **Squelch**.

Настройка

1. Включите декодер и выберите **FSK / RTTY** в меню. Либо просто нажмите кнопку **RTTY**.
2. Появится панель декодера. Её заголовок соответствует варианту — *FSK / RTTY Decoder*, *PSK31 Decoder* или *Olivia Decoder*.
3. Выберите **Variant**. Параметры обновятся автоматически.
4. Для Olivia установите **Mode** (число тонов / полоса) в точном соответствии с передачей — см. примечания об Olivia ниже.
5. В меню **Known frequency** выберите распространённую частоту для выбранного варианта и нажмите **Tune**, чтобы перейти на неё.
6. Подстраивайте частоту, пока декодированный текст не станет стабильным и читаемым.

Известные частоты по вариантам

Морской FSK / SITOR

- 518,0 кГц — международный NAVTEX
- 490,0 кГц — национальный NAVTEX
- 4209,5 / 6314,0 / 8416,5 / 12579,0 / 16806,5 / 22376,0 кГц — SITOR на KB

Метеорологический RTTY

- 4583,0 / 7646,0 / 10100,8 / 11039,0 / 14467,3 кГц — DWD (метеослужба Германии)

Любительский RTTY

- 3590 кГц (80 м), 7043 кГц (40 м), 10143 кГц (30 м), 14083 кГц (20 м), 21083 кГц (15 м), 28083 кГц (10 м)

PSK31

- 3580,15 кГц (80 м), 7040,15 кГц (40 м), 10142,15 кГц (30 м), 14070,15 кГц (20 м), 18100,15 кГц (17 м), 21080,15 кГц (15 м), 24920,15 кГц (12 м), 28120,15 кГц (10 м)

Olivia

- 3577,75 кГц (80 м), 7073,75 кГц (40 м), 10142,25 кГц (30 м), 14075,5 кГц (20 м), 18103,75 кГц (17 м), 21075,75 кГц (15 м), 24921,75 кГц (12 м), 28123,75 кГц (10 м)

Параметры

Параметр	Относится к	Описание
Center audio (Hz)	ко всем	Звуковая частота середины между mark и space; для PSK31 — несущая, для Olivia — центр блока тонов. Выпадающий список для вариантов FSK, поле свободного ввода для PSK31 и Olivia
Shift (Hz)	только FSK	Разность частот между тонами mark и space
Baud	только FSK	Символьная скорость
Framing	только FSK	Биты данных, чётность, стоп-биты (например, 7N1 = 7 данных, без чётности, 1 стоп)
Encoding	только FSK	Набор символов (CCIR-476, ITA2/Бодо или ASCII)
Invert mark / space	только FSK	Меняет местами тоны mark и space
Auto shift detect	только FSK	Пытается автоматически измерить сдвиг по принимаемому сигналу
Mode (тоны / Гц)	только Olivia	Число тонов и полоса — должны точно совпадать с передачей
Squelch (FEC S/N)	только Olivia	Насколько уверенным должно быть совпадение коррекции ошибок, прежде чем текст будет напечатан
Squelch (SNR)	Любительский и метео-RTTY	Не даёт шуму печататься случайными буквами. Начинается с −5 дБ для любительского RTTY и −8 дБ для метео-RTTY; в крайнем левом положении (off) отключён. Пока шумоподаватель закрыт, значение SNR показано серым

Измерения сигнала

В строке состояния отображаются измерения в реальном времени, и набор полей меняется вместе с вариантом:

Вариант	Отображаемые поля
Варианты FSK	Mark / Space (измеренные частоты тонов), SNR , Lock , Timing
PSK31	Carrier (Гц, после автоматической коррекции частоты), IMD (дБ), SNR , Lock , Timing
Olivia	Centre (Гц), Mode , S/N , FEC (%), Sync

Timing/Sync показывает LOCKED/SYNCD, когда декодер ведёт сигнал, и SEARCH, пока он ещё ищет.

SNR вариантов FSK и PSK31 измеряется на самом сигнале и приводится к полосе 3 кГц, так что оба читаются по одной шкале: с точностью около 1 дБ от +20 дБ до −5 дБ. Пустая частота показывает примерно от −12 до −15 дБ, а в PSK31 0 дБ — это уже сильный сигнал. **S/N** у Olivia — другое: это показатель качества коррекции ошибок.

Любительский RTTY (45,45 бод, 170 Гц) принимается без ошибок примерно до 0 дБ, с около 1,5 % неверных знаков при −3 дБ и 20 % при −6 дБ.

Дополнительные органы управления

- ⇒ **Set IF Band-Pass** — сужает полосу пропускания приёмника, плотно охватывая сигнал. Ширина зависит от варианта: 300 Гц для любительского RTTY (достаточно узко, чтобы убрать более сильную станцию в 250 Гц рядом и всё же принимать сигнал, расстроенный на 40 Гц), 594 Гц для метео-RTTY, mark/space плюс запас для морского FSK, около ± 100 Гц для PSK31 и полная ширина блока тонов плюс запас для Olivia. Полоса следует за декодером, когда **Auto-tune** или **Auto shift detect** сдвигает тоны, и при изменении настройки **Baud**.
- ☞ **Auto-tune Center** — автоматический поиск сигнала. Для вариантов FSK ищется сбалансированная пара тонов, для PSK31 — несущая, для Olivia — самый сильный блок выбранной полосы. **Auto shift detect** меняет сдвиг только при наличии настоящего двухтонального сигнала, поэтому больше не прыгает по шуму.

Примечания по PSK31

PSK31 — самый распространённый вид связи «клавиатура — клавиатура» на КВ. Он занимает всего 62 Гц, поэтому несколько QSO располагаются рядом в пределах нескольких сотен герц около частоты сбора, и настройка сводится к выбору одной дорожки из группы на водопаде.

- Декодер сам исправляет ошибку настройки в пределах примерно ± 25 Гц, поэтому достаточно подойти близко. Поле **Carrier** в строке измерений показывает, где он в итоге установился.
- **IMD** измеряет качество передатчика, а не приём: чистый сигнал даёт около −20 дБ или лучше. Плохое значение означает, что корреспондент перегружает свой передатчик, а не что вы расстроены.
- Текст появляется посимвольно без коррекции ошибок, поэтому слабый сигнал вырождается в отдельные неверные буквы, а не прекращается совсем.

Примечания по Olivia

Olivia жертвует скоростью ради помехоустойчивости. Она значительно медленнее PSK31, но декодирует сигналы, неразличимые на слух, из-за чего популярна для слабых сигналов и дальних связей.

- **Настройка Mode должна точно совпадать с передачей.** Неверное сочетание тонов и полосы не декодирует ровным счётом ничего — не искажённый текст, а тишину. Панель открывается на **8 / 250** — узкой конфигурации, которую оставляют работать на частотах вызова; **16 / 500** и **32 / 1000** — два других распространённых варианта, предлагается также **16 / 1000**.
- Olivia не передаёт преамбулу, поэтому декодер вынужден искать синхронизацию. **Дайте несколько секунд** после настройки, прежде чем появится текст. Поле **Synс** показывает SEARCH до захвата.
- Коррекция ошибок работает по блокам, поэтому текст приходит **пачками, а не ровным потоком**, с задержкой в несколько блоков между передачей и отображением.
- **Squelch (FEC S/N)** задаёт, насколько уверенной должна быть коррекция ошибок перед выводом. Значение по умолчанию 4,0 не пропускает шум; 3,0 — нижняя граница, ниже которой случайный шум начинает печатать отдельные символы. Хороший сигнал показывает

8–9 на индикаторе **FEC**, так что запас для поднятия шумоподавителя на загруженном диапазоне вполне достаточен.

Примечание о режиме: пока декодер активен, он берёт на себя управление режимом демодуляции и полосой ПЧ. И то и другое автоматически восстанавливается при его выключении. Все пять вариантов используют **USB**.

Примечание о полярности (только варианты FSK): для метеорологического RTTY обычно нужно установить флажок **Invert mark / space**. Для морского FSK (типа SITOR/NAVTEX) и любительского RTTY оставьте его снятым — любительский RTTY передаёт mark как более высокую радиочастоту, а USB сохраняет её как более высокий звуковой тон, что и соответствует снятому флажку. Если декодированный текст искажён, первое, что стоит попробовать, — переключить этот флажок. Для PSK31 и Olivia он скрыт, так как у них нет пары mark/space.

Примечание о буквах и цифрах (только варианты FSK): код Бодо хранит буквы и цифры в двух отдельных состояниях, и шум может перевести декодер в неверное — после чего искажается каждый последующий символ, а не только повреждённый. Поэтому декодер возвращается к буквам на каждом пробеле; это общепринятая практика, и она исправляет испорченное переключение в пределах одного-двух слов, а не целой строки. Плата за это в том, что группы цифр, разделённые пробелами, требуют, чтобы передающая сторона повторяла переключение на цифры после каждого пробела — как передатчики обычно и делают.

11. SSTV

Что это: медленное телевидение передаёт неподвижные изображения по обычному голосовому каналу SSB, строка за строкой, в виде частотно-модулированного тона между 1500 Гц (чёрный) и 2300 Гц (белый). Полное изображение занимает от 36 секунд до 4,5 минут в зависимости от режима.

Рекомендуемые частоты (USB)

Диапазон	Частота	Режим	Примечания
20 м	14.230 МГц	USB	Основная международная частота вызова SSTV — безусловно самая активная
20 м	14.233 МГц	USB	Запасная, когда 14.230 занята
15 м	21.340 МГц	USB	
10 м	28.680 МГц	USB	Активна при прохождении
40 м	7.171 МГц	LSB	
80 м	3.845 МГц	LSB	Региональная, вечерами

Боковая полоса выбирается автоматически. При запуске декодера выбирается **LSB ниже 10 МГц** и **USB выше**, в соответствии с обычной любительской практикой. На неверной боковой полосе соответствие тонов инвертировано, и изображение не декодируется. Если встретите станцию, не соблюдающую соглашение, просто смените боковую полосу вручную — декодер продолжит работу, очистит кадр и начнёт заново с новой настройкой.

Настройка

1. Включите декодер и выберите **SSTV** в выпадающем списке. Или просто нажмите кнопку **SSTV**. Приёмник автоматически сменит боковую полосу — USB выше 10 МГц, LSB ниже.
2. Настройтесь так, чтобы тоны изображения попадали в середину полосы пропускания. У правильно настроенного сигнала синхроимпульсы находятся на 1200 Гц, а содержимое изображения — между 1500 и 2300 Гц.
3. Оставьте **Mode** в положении **Auto**, если вы заранее не знаете, что передаётся. Изображение строится строка за строкой по мере приёма.

Режимы

Настройка	Изображение	Длительность
Auto	Определяется автоматически	—
Martin M1 / M2	320×256, цветное	114 с / 58 с
Scottie S1 / S2	320×256, цветное	110 с / 71 с
Scottie DX	320×256, цветное	269 с
Robot 36 / 72	320×240, цветное	36 с / 72 с

Auto работает двумя способами: читает **заголовок VIS** — цифровой код режима, передаваемый в первые 300 мс передачи, — а если заголовок был пропущен (вы настроились поздно или он потерялся из-за QSB), определяет режим по синхроимпульсам. Выбор конкретного режима задаёт его принудительно, но декодер всё равно проверяет синхронизацию, прежде чем что-либо рисовать, поэтому неверный выбор даёт отсутствие изображения, а не шум.

Чтение результатов

Строка состояния под органами управления сообщает, чем занят декодер:

Состояние	Значение
Waiting for VIS / AUTO lock	Слушает; пока ничего не опознано
Martin M1? verifying sync...	Найден кандидат, идёт подтверждение по следующим строкам
Martin M1 lock (VIS)	Захват по заголовку VIS
Martin M1 lock (AUTO)	Захват по синхроимпульсам
Martin M1 lock (MANUAL)	Запущен кнопкой ● Force
Candidate rejected (no sync)	Кандидат не подтвердился — нормально на шуме
– sync lost, resetting	Сигнал пропал посреди изображения
Frame complete	Изображение получено полностью

Определённый режим также отображается зелёным рядом с заголовком **SSTV**, а счётчик строк показывает ход приёма.

Органы управления

Кнопка	Действие
► Start	Взводит декодер. Сам по себе он изображение не рисует — захват всё равно должен прийти от заголовка VIS или от обнаружения синхронизации.
■ Stop	Останавливает декодирование.
↺ Reset	Очищает холст и снова взводит декодер на месте. Используйте между изображениями или после перестройки.
● Force	Начинает рисовать немедленно в режиме, выбранном в списке, полностью пропуская VIS и обнаружение синхронизации. Доступна только когда Mode не в положении Auto .
Save	Сохраняет текущее изображение в формате PNG.

Когда применять Force. Если вы видите изображение на водопаде, но декодер не может за него зацепиться — необычный режим, слишком слабый или искажённый для детекторов сигнал либо пропущенный заголовок — выберите режим ручную и нажмите ● **Force**. Кадр привязывается к моменту нажатия, а метка режима показывает **MANUAL**, чтобы вы отличали его от захвата по VIS или AUTO. Декодер всё равно подстраивается под настоящий синхроимпульс, если находит его, так что чуть раннее или позднее нажатие корректируется.

Поскольку Force пропускает все проверки, он с готовностью нарисует шум, если нажать его на пустом канале, и сам не остановится — нажмите ■ **Stop** или ↺ **Reset**.

Примечания

- **Шум не запускает декодер.** Раньше атмосферных разрядов, искр и сетевой помехи хватало, чтобы запустить декодирование. Теперь каждая ступень обнаружения проверяет, что тон является настоящим тоном, и подтверждает кандидата по следующим строкам, прежде чем будет нарисован хоть один пиксель. Ожидайте, что на пустом диапазоне декодер будет спокойно молчать.
- **Изображение уходит по диагонали, если вы не на частоте.** SSTV не прощает ошибок настройки. Если строки заваливаются, подстройте частоту малыми шагами и дайте начаться следующему изображению.
- Цветовой баланс и резкость зафиксированы; регулировать нечего.

12. Общие советы

Сначала нужно включить Decoder: ON. Выпадающий список недоступен (затенён), пока вы не нажмёте кнопку Decoder.

Один декодер за раз. Выбор нового декодера в списке автоматически останавливает ранее активный и отменяет сделанные им изменения режима или полосы пропускания.

Режим настраивается за вас. HF FAX и NAVTEX переключают приёмник в USB сразу при выборе, SSTV выбирает боковую полосу по диапазону (LSB ниже 10 МГц, USB выше), а FAX, NAVTEX и FSK дополнительно подстраивают полосу пропускания при нажатии их кнопки Tune. FT8, FT4, FT2, JS8 и WSPR делают то же самое сразу при выборе — из ряда кнопок или из выпадающего списка: приёмник переключается в USB и получает свою полосу пропускания — вся подполоса 3 кГц для семейства FT8, 1350-1650 Гц для WSPR. При остановке декодера восстанавливается режим диапазона по умолчанию.

Включение декодера больше не прерывает звук. Декодеры работают в собственных потоках, поэтому при запуске, остановке или смене декодера нет ни пауз, ни щелчков, ни выпадений.

Точность системных часов важна. FT8, FT4 и WSPR критичны ко времени. Они декодируют в фиксированных окнах, выровненных по UTC. Если часы компьютера уходят более чем на 1-2 секунды, число декодов существенно упадёт. Используйте клиент NTP, чтобы часы шли точно.

Шумовые фильтры до декодеров не доходят. NR, NB, NS и AN — это средства для прослушивания, только для ваших ушей. Каждый декодер — FT8, FT4/FT2, CW, WSPR, FAX, NAVTEX, FSK/RTTY/PSK31/Olivia, SSTV и граббер QRSS — берёт звук до этих фильтров, так что настраивайте их так, как лучше звучит, не думая о качестве декодирования. По той же причине декодеры продолжают работать, пока приёмник приглушён или закрыт шумоподавитель: можно выключить динамик и оставить декодер или ночную запись QRSS собирать данные. Единственное, что действительно следует за тем, что вы слышите, — аудиоспектрограмма, которая для того и сделана, чтобы показывать отфильтрованный звук.

Качество сигнала важнее его силы. Большинство этих декодеров рассчитаны на слабые сигналы. Более тихий диапазон с меньшим шумом часто продуктивнее громкого сигнала, забитого помехами. Используйте водопад и настройки полосы пропускания, чтобы выявить и обойти QRM до включения декодера.

Не стесняйтесь пользоваться кнопками Refresh и Clear. Изображения FAX уплывают, если частота слегка неточна, а текстовые декодеры накапливают шумовые символы. Начать заново после подстройки часто даёт гораздо более чистый результат.

Автоматическая отправка спотов и графики сервера

Декодирования FT8, FT4 и WSPR может выгружать и сам сервер — FT8/FT4 в PSK Reporter, WSPR в WSPRnet — через службу autorun, которую оператор запускает из панели администратора. Она не зависит от декодеров в вашем браузере: работает независимо от того, слушает кто-нибудь или нет, и ничего из декодированного вами в браузере не отправляется.

Оператор следит за ней по двум счётчикам, которые легко перепутать: плитки по декодерам считают споты, отправленные с последнего запуска службы, а число рядом с каждым флажком диапазона/вида связи — итог этой позиции за всё время, переживающий перезапуски. В той же панели есть страница **Графики** с частотой процессора, нагрузкой, температурой и числом слушателей за период от 15 минут до 24 часов. Оба описаны в [руководстве по панели администратора](#).

PhantomSDR-Plus — фопк sv1btl — phantomsdr.no-ip.org

RADE README

RADE_README.md

Цифровой голос RADE v1 для PhantomSDR-Plus

RADE (Radio AutoencoDEr) — флагманский KB-режим цифрового голоса от FreeDV. Он использует гибридный машинный обучение и ЦОС (нейронный вокодер FARGAN), обеспечивая качественную речь на KB при отношении сигнал/шум вплоть до -2 дБ в полосе всего 1500 Гц — уже, чем сигнал SSB.

Этот документ описывает полную интеграцию приёма RADE v1 в PhantomSDR-Plus, реализованную в виде sidcar-процесса на Python (`rade_helper.py`), который связывает браузер с цепочкой декодирования `radac_rxe.py` + `lpcnet_demo`.

Установка — коротким путём

```
cd ~/PhantomSDR-Plus
./install_rade.sh
```

Это и есть вся установка. Сценарий ставит системные пакеты тем менеджером пакетов, который есть на машине (`apt`, `pacman`, `dnf` или `zypper`), для отсутствующих в дистрибутиве модулей переходит на `pip`, обновляет `websockets`, если версия из дистрибутива старше нужной sidcar-у 11.0, клонирует и собирает `radac`, проверяет веса модели, пересобирает фронтенд и запускает sidcar. На Ubuntu 22.04 он передаёт работу `install_rade_ubuntu22.sh`. `./install.sh` предлагает запустить его в рамках обычной установки PhantomSDR-Plus, так что на новой машине RADE уже стоит.

Этот документ объясняет, как устроен RADE и как его запускать, — это не процедура установки. Если установку нужно выполнить вручную, на системе, которую сценарий не покрывает, или чтобы починить один шаг, следуйте руководству по ссылке ниже: шаги записаны там и только там. Всё, что есть здесь — архитектура, исправленные файлы, порт, работа с RADE в браузере, переменные окружения, устранение неполадок и измерение одновременной нагрузки, — относится к любой установке, как бы она ни была сделана.

- **Ручная установка в Linux** - ручной путь, шаг за шагом, для Ubuntu, Debian, Fedora, Arch и Raspberry Pi OS

Содержание

1. Установка — коротким путём
2. Как это работает
3. Обзор архитектуры
4. RADEL против RADEU
5. Предварительные требования
6. Установка вручную
7. Размещение исправленных файлов

8. Управление sidcar
9. Порт 8074
10. Использование RADE в браузере
11. Проверка и отладка
12. Переменные окружения
13. Изменённые файлы
14. Сводка прохождения сигнала
15. Устранение неполадок
16. Измерение параллелизма на своём оборудовании
17. Обновление RADE v1

Как это работает

RADE v1 не может работать в браузере — ему нужны PyTorch и нейронный вокодер FARGAN, слишком большие для WASM. Решение — sidcar-процесс на Python (`rade_helper.py`), работающий на том же сервере, что и PhantomSDR-Plus.

Важно: `freedv_rx` из репозитория `codec2` **не** поддерживает RADEV1. Цепочка декодирования RADE целиком находится в отдельном репозитории `radae` Дэвида Рой (VK5DGR). Не пытайтесь использовать `freedv_rx` из `codec2` для RADE.

Когда вы выбираете RADE в браузере:

1. Веб-интерфейс задаёт базовую демодуляцию USB или LSB — сервер на C++ демодулирует сигнал SSB как обычно.
2. Необработанный демодулированный PCM снимается в `audio.js` **до** любого отключения звука или шумоподавителя — `radae_rxe.py` нужен непрерывный входной поток, чтобы удерживать кадровую синхронизацию.
3. Каждый блок PCM дополняется нулями из вещественного f32 в комплексный f32 (вещественная часть + 0.0 мнимой) — именно этого `radae_rxe.py` ожидает на стандартном входе.
4. Sidcar передаёт эти отсчёты в `radae_rxe.py`, который выдаёт признаки вокодера.
5. `lpcnet_demo -fargan-synthesis` преобразует признаки в речь s16 с частотой 16000 Гц.
6. Sidcar преобразует s16 → f32 и отправляет результат обратно в браузер двоичными кадрами WebSocket.
7. Браузер воспроизводит декодированную речь через Web Audio API на 16000 Гц.

Сервер на C++ (`spectrumserver.cpp`) при этом вообще не изменяется.

Обзор архитектуры



RADEL против RADEU

RADE v1 всегда передаётся по SSB. По соглашению:

Режим	Боковая полоса	Использовать на диапазонах
RADEL	LSB	160 м, 80 м, 40 м (≤ 10 МГц)
RADEU	USB	20 м, 17 м, 15 м, 12 м, 10 м (> 10 МГц)

Предварительные требования

Требование	Версия	Примечания
PhantomSDR-Plus	любая	с веб-интерфейсом Vite/Svelte
Linux	Ubuntu 24.04+ / Debian Bookworm+	проверено
Python	3.8+	для <code>rade_helper.py</code> и <code>radae_rxe.py</code>
репозиторий radae	последний master	даёт <code>radae_rxe.py</code> и <code>lpcnet_demo</code>
cmake	3.10+	для сборки <code>lpcnet_demo</code> из <code>radae</code>
PyTorch	2.0+	требуется для <code>radae_rxe.py</code>
Node.js	16+	для <code>npm run build</code>
websockets (Python)	10-16+	<code>pip3 install websockets</code> — поддерживаются все версии
matplotlib	любая	требуется <code>radae_rxe.py</code> при импорте
numpy	1.23+	обязателен; также обеспечивает передискретизацию
scipy	любая	необязательно, обеспечивает точную передискретизацию

Установка вручную

`./install_rade.sh` выполняет установку целиком. Если вы хотите сделать её сами — на системе, которую сценарий не покрывает, или чтобы починить один шаг, — процедура находится в [Руководстве по ручной установке RADE v1](#), и только там. Оно охватывает Ubuntu, Debian, Fedora, Arch и Raspberry Pi OS за один проход:

Шаг	Описание
Шаг 1	Системные пакеты и проверка Node.js 22+
Шаг 2	Пакеты Python, включая флаг PEP 668 и колесо torch только для CPU
Шаг 3	Клонирование и сборка репозитория radae, проверка <code>lpcnet_demo</code> и весов модели
Шаг 4	Проверка цепочки декодирования в автономном режиме, до подключения к браузеру
Шаг 5	Сборка фронтенда PhantomSDR-Plus
Шаг 6	Управление sidcar
Шаг 7	Открытие порта 8074
Шаги 8-10	Запуск сервера, проверка RADE, работа в браузере

Остальная часть этого документа предполагает, что всё это сделано.

Размещение исправленных файлов

Скопируйте следующие файлы из набора исправлений в свой репозиторий PhantomSDR-Plus.

Новый файл — поместите в корень репозитория, рядом со скриптом запуска, которым пользуетесь:

```
rade_helper.py
```

Исправленные файлы веб-интерфейса — поместите в frontend/src/:

```
audio.js
App.svelte
```

Что делают исправления

audio.js — 5 изменений:

- Конструктор: 5 новых полей состояния RADE (`decodeRADE`, `_radeSideband`, `_radeSocket`, `_radeCallback`, `_radeReady`, `_radeNextTime`)
- Сохранение PCM до усиления: `pcmArrayPreBoost` сохраняется до 300-кратного усиления FLAC, чтобы RADE получал исходную амплитуду (усиление привело бы к перегрузке `radae_rxe.py`)
- `playAudio()`: отвод PCM для RADE с использованием звука до усиления, перед фильтром отключения/шумоподавителя
- `playAudio()`: защита `if (this.decodeRADE) return` — подавляет воспроизведение исходного SSB, пока звучит декодированная RADE речь
- Новые методы: `setRADEDecoding()`, `setRADECallback()`, `_radePlayPCM()` с планируемым воспроизведением без разрывов по часам `_radeNextTime` (воспроизведение на **16000 Гц** — выходная частота `lpcnet_demo`)

Каждый вариант Svelte — 6 изменений:

- `demodulationDefaults`: `RADEL {type: 'LSB', offsets: [2200, -700]}`, `RADEU {type: 'USB', offsets: [-700, 2200]}` — полоса начинается в 700 Гц от несущей, ширина 1500 Гц
- Переменные состояния: `radeEnabled`, `radeConnected`, `radeSynced`, `radeSnr`, `_radeDeactivate()`
- `_radeDeactivate()`: останавливает декодер и **восстанавливает режим диапазона по умолчанию** из `bands-config.js` — поведение идентично отключению FAX, NAVTEX, FSK
- `_deactivateAll()`: строка очистки RADE
- `activateSelectedDecoder()`: ветви `radel` и `radeu`
- Выпадающий список декодеров: две новые записи `<option>`
- Панель состояния: индикатор соединения, состояние синхронизации/SNR, баннер ошибки

Скрипты запуска

Пусковые сценарии приёмника (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) сами запускают `sidecar`, как только сервер заработал, а `stop-websdr.sh` останавливает его вместе с сервером. `rade.sh` остаётся для автономного запуска `sidecar` — см. [Управление sidecar](#).

Управление sidecar

Отдельный управляющий сценарий для RADE обычно не нужен. Пусковые сценарии приёмника (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) сами запускают `sidecar`, как только сервер подтверждённо работает, перезапускают его при завершении и сообщают его состояние в итоговой строке — ✓ `RADE sidecar running` либо пометка `·` с объяснением, почему он не был включён (не установлен,

RADE_ENABLED=0 или нет python3). `./stop-websdr.sh` останавливает его вместе с сервером. Чтобы запустить сервер без RADE, стартуйте его с `RADE_ENABLED=0`.

`rade.sh` остаётся для автономного применения — запуска `sidecar` без пускового сценария PhantomSDR-Plus или его отдельной проверки:

```
chmod +x ~/PhantomSDR-Plus/rade.sh
cd ~/PhantomSDR-Plus
./rade.sh start      # start sidecar with self-restarting watchdog
./rade.sh stop       # stop sidecar + watchdog + all lpcnet_demo children
./rade.sh restart    # stop then start cleanly
./rade.sh status     # show running / stopped + process info
```

Активность `sidecar` записывается в `~/PhantomSDR-Plus/rade.log`:

```
tail -f ~/PhantomSDR-Plus/rade.log
```

Не смешивайте эти два способа, пока сервер работает. Если сторож пускового сценария активен, он вернёт `sidecar` примерно за пять секунд, и `./rade.sh stop` будет выглядеть как неудача. Чтобы остановить RADE при работающем сервере, либо запускайте сервер с `RADE_ENABLED=0`, либо остановите всё командой `./stop-websdr.sh`.

Если вы всё же запускаете `rade.sh` автономно, **всегда используйте `./rade.sh stop`** — один лишь `kill -f rade_helper.py` не пересилит его собственного сторожа, который возрождает процесс за 3 секунды.

Порт 8074

`Sidecar` слушает TCP-порт **8074**, и браузер подключается к нему напрямую, поэтому порт должен быть доступен снаружи. Как его открыть — `ufw`, `firewalld`, `iptables`, правило NAT на маршрутизаторе и прокси Nginx на случай, когда провайдер блокирует порт целиком, — это [шаг 7 руководства по ручной установке](#).

Две вещи стоит повторить здесь, потому что они отнимают больше всего времени:

```
# Is the sidecar listening?
ss -tlnp | grep 8074
```

Осторожно с NAT hairpin: проверка через `curl` с самого сервера на его собственное публичное имя часто возвращает `Connection refused`, даже когда порт открыт, — многие маршрутизаторы не заворачивают трафик обратно. Всегда проверяйте с внешней машины или используйте <https://portchecker.co>.

Использование RADE в браузере

1. Откройте веб-интерфейс PhantomSDR-Plus
2. Найдите активные станции на qso.freedv.org
3. Настройтесь на частоту станции по шкале
4. В **Decoder Options** выберите:
5. **RADE v1 — RADEL (LSB)** для 40 м / 80 м / 160 м

6. **RADE v1 — RADEU (USB)** для 20 м / 17 м / 15 м / 12 м / 10 м

7. Нажмите **Decoder: ON**

Кнопки RADEL / RADEU (в одно нажатие)

Шаги 4 и 5 можно заменить одним нажатием. Пара кнопок **RADEL / RADEU** доступна в трёх местах:

- рядом с заголовком **Modes selector** на главной панели;
- внутри всплывающего окна **Modes**;
- внутри всплывающего окна **Bands**.

Нажатие выбирает декодер, переводит декодер в ON, закрывает всплывающее окно, из которого вы нажали, и прокручивает панель RADE в поле зрения. Кнопка остаётся синей, пока RADE работает. **Повторное нажатие той же кнопки выключает RADE** — панель закрывается, а вид модуляции и полоса пропускания возвращаются к значениям, описанным ниже. Кнопки, выпадающий список **Decoder Options** и кнопка ON/OFF используют одно и то же состояние.

Состояния индикатора панели

Индикатор	Значение
Красный — «Connecting to sidecar...»	Порт 8074 недоступен или sidecar не запущен
Жёлтый — «Searching for signal...»	Sidecar подключён, кадр RADE ещё не обнаружен
Зелёный — «Synced · SNR x.x dB»	Идёт декодирование — воспроизводится речь

Когда вы выключаете декодер

Режим и полоса автоматически возвращаются к правильному значению по умолчанию для текущей частоты, заданному в `bands-config.js`, — точно так же, как у FAX, NAVTEX, FSK.

Проверка и отладка

Работает ли sidecar?

```
ps aux | grep rade_helper | grep -v grep
ss -tlnp | grep 8074
```

Наблюдение за активностью в реальном времени

```
tail -f ~/PhantomSDR-Plus/rade.log
```

При подключении браузера:

```
[RADE] client connected: x.x.x.x:XXXXX
[RADE] x.x.x.x:XXXXX sps=12000 sideband=LSB
[RADE] x.x.x.x:XXXXX spawning pipeline (torch_threads=1)
```

Быстрая проверка WebSocket

```
python3 - <<< 'EOF'
import asyncio, websockets, json

async def test():
    async with websockets.connect('ws://localhost:8074') as ws:
        await ws.send(json.dumps({'type': 'init', 'sps': 8000, 'sideband': 'LSB'}))
        print(await ws.recv()) # expect: {"type": "status", "connected": true}

asyncio.run(test())
EOF
```

Консоль браузера (F12)

```
[RADE] ► ENABLED LSB @ 12000 Hz → helper ws://localhost:8074
```

Хорошо — WebSocket открыт. Значение в Гц соответствует `audioOutputSps` вашего сервера (обычно 8000–12000 Гц).

```
[RADE] sidecar socket error
```

Порт 8074 недоступен — проверьте межсетевой экран и правило NAT на маршрутизаторе.

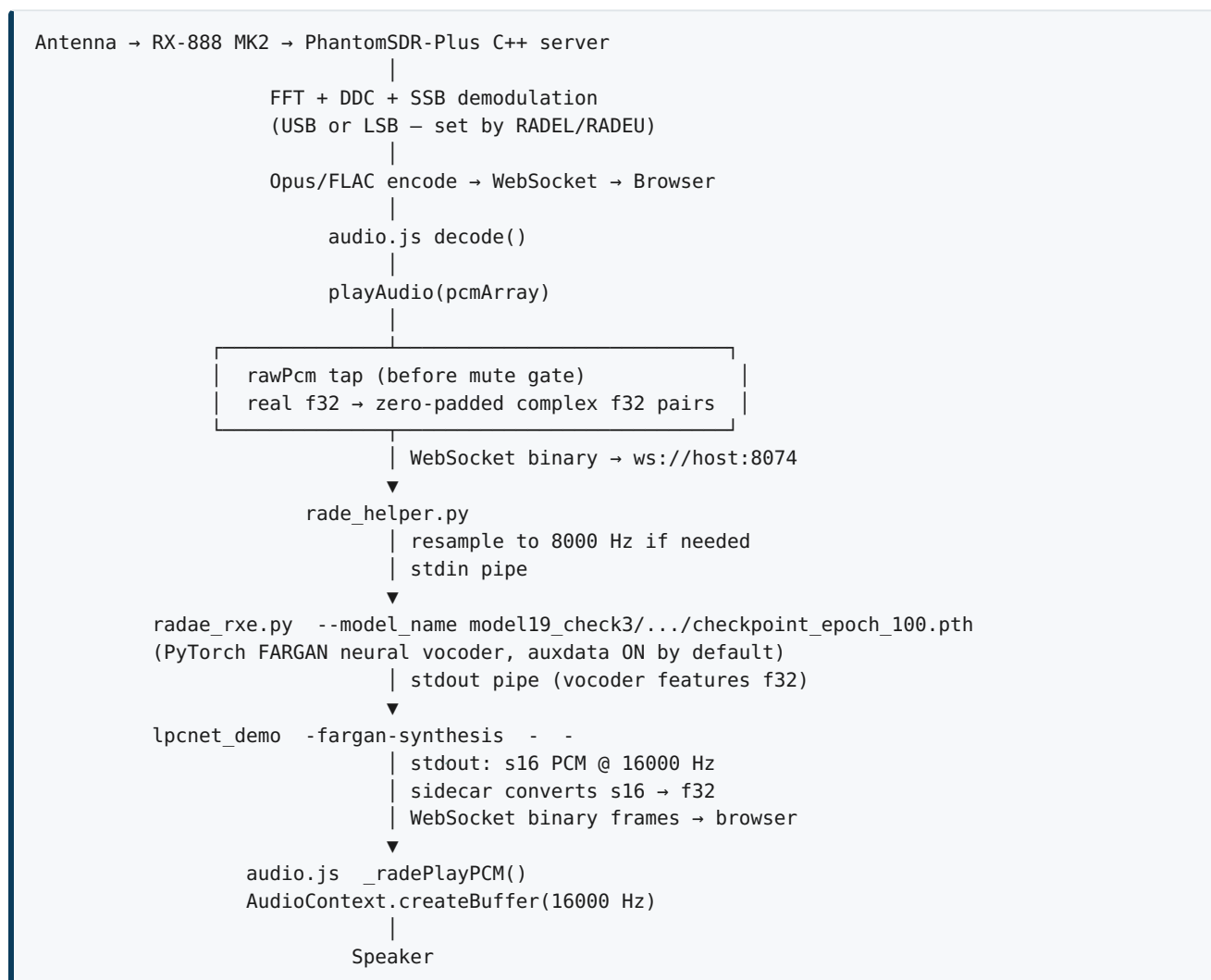
Переменные окружения

Переменная	По умолчанию	Описание
<code>RADE_HELPER_PORT</code>	8074	TCP-порт, который слушает sidecar
<code>RADE_HELPER_HOST</code>	0.0.0.0	Адрес привязки (127.0.0.1 за прокси)
<code>RADAE_DIR</code>	~/radeae	Корень репозитория radeae
<code>RADE_MODEL</code>	RADAE_DIR/model19_check3/ checkpoints/ checkpoint_epoch_100.pth	Веса модели
<code>LPCNET_DEMO</code>	RADAE_DIR/build/src/lpcnet_demo	Исполняемый файл lpcnet_demo
<code>RADE_AUXDATA</code>	1	Установите 0, чтобы передать --noauxdata в radeae_rxe.py
<code>RADE_TORCH_THREADS</code>	1	Потоки PyTorch/OpenBLAS на экземпляре radeae_rxe.py — ограничивает нагрузку на ЦП
<code>RADE_PIN_CORES</code>	1	Привязывает каждую цепочку декодирования к своему набору ядер; 0 отключает привязку
<code>RADE_CORES_PER_CLIENT</code>	2	Ядер на клиента при включённой привязке

Изменённые файлы

Файл	Тип	Примечания
rade_helper.py	Новый	Sidecar на Python: сервер WebSocket + двухпроцессная цепочка декодирования
frontend/src/audio.js	Изменён	4 исправления
frontend/src/App.svelte	Изменён	6 исправлений
rade.sh	Новый	Сценарий управления RADE sidecar для автономного применения (start/stop/restart/status)
start-rx888mk2.sh	Изменён	запускает, сторожит и сообщает о состоянии sidecar; то же для start-airspyhf.sh, start-rtl.sh, start-rspla.sh, start-fobos-hf.sh, start-fobos.sh, start-hackrf.sh
stop-websdr.sh	Изменён	останавливает sidecar вместе с сервером
spectrumserver.cpp	Без изменений	Изменения в C++ не требуются

Сводка прохождения сигнала



Устранение неполадок

Красный баннер — «Sidecar not reachable»

```
# Is sidecar running?
ps aux | grep rade_helper | grep -v grep

# Start it manually for testing
python3 ~/PhantomSDR-Plus/rade_helper.py &

# Check port is open externally – use portchecker.co NOT curl from the server
# (curl from the server uses NAT loopback and gives false "Connection refused")
```

Порт 8074 закрыт по данным portchecker.co, несмотря на правило iptables

Возможно, провайдер фильтрует порт, либо отсутствует правило NAT на маршрутизаторе.
Варианты:

1. Попробуйте другой порт: `RADE_HELPER_PORT=8080 python3 rade_helper.py`
2. Проксировать через уже открытый публичный порт с помощью Nginx (см. [шаг 7 руководства по ручной установке](#))

`radae_rxe.py: error: unrecognized arguments: model_path`

Путь к модели нужно передавать через `--model_name`, а не позиционным аргументом:

```
# WRONG – positional argument
python3 radae_rxe.py model19_check3/checkpoints/checkpoint_epoch_100.pth

# CORRECT – named argument
python3 radae_rxe.py --model_name model19_check3/checkpoints/checkpoint_epoch_100.pth
```

`ModuleNotFoundError: No module named 'matplotlib'`

```
pip3 install matplotlib
```

`radae_rxe.py` импортирует `matplotlib` безусловно в начале файла.

`ModuleNotFoundError: No module named 'torch'`

```
pip3 install torch
```

`lpcnet_demo: No such file or directory`

Сборка `radae` не завершилась. Соберите заново:

```
cd ~/radae/build
cmake .. && make -j$(nproc)
ls src/lpcnet_demo      # should exist now
```

Ошибка `size mismatch` в `inference.sh`

`model19_check3` требует `--auxdata` при кодировании через `inference.sh`:

```
./inference.sh model19_check3/checkpoints/checkpoint_epoch_100.pth \
wav/brian_g8sez.wav /dev/null \
--rate_Fs --pilots --pilot_eq --eq_ls --cp 0.004 \
--write_rx rx.f32 --auxdata # ← required for model19_check3
```

Обратите внимание: при **декодировании** с помощью `radae_rxe.py` `--auxdata` включён по умолчанию — передавать его не нужно.

Жёлтый индикатор — «Searching for signal» — синхронизация не наступает

- Убедитесь в правильной боковой полосе: RADEL для ≤ 10 МГц, RADEU для > 10 МГц
- Проверьте на qso.freedv.org, что станция сейчас передаёт
- RADE v1 использует 30 несущих в полосе 1500 Гц — на водопаде выглядит компактной группой
- Дайте до 1,5 секунды на захват

`underrun!!!` от `aplay` во время проверки цепочки (шаг 3)

Ожидается только при автономной проверке с файлами. `radae_rxe.py` обрабатывает медленнее, чем идёт файловый ввод-вывод, из-за чего звуковой буфер опустошается. При живом приёме этого не происходит, так как браузер подаёт звук в реальном времени.

`ConnectionClosedError: received 1011 (internal error)`

Sidecar принял соединение WebSocket, но аварийно завершился внутри до ответа. Причина — несовместимая версия `rade_helper.py`: более старая версия пыталась передать `asuncio StreamReader` в качестве стандартного ввода подпроцесса, что тихо не срабатывает и приводит к закрытию с ошибкой 1011.

Решение: замените `rade_helper.py` текущей версией из набора исправлений. Текущая версия использует `os.pipe()` для межпроцессного канала и совместима с `websockets` от 10.x до 16.x+.

Несколько одновременных пользователей

Каждое подключение браузера порождает собственную независимую пару `radae_rxe.py` + `lpcnet_demo`, поэтому каждый пользователь может свободно настраиваться на свою частоту.

По умолчанию `radae_rxe.py` использует **все доступные ядра ЦП** для матричных операций PyTorch, из-за чего нагрузка достигает ~900 % на экземпляр. `rade_helper.py` ограничивает это двумя оптимизациями:

1. `OMP_NUM_THREADS=1` (и аналоги для MKL/OpenBLAS) — ограничивает PyTorch одним потоком
2. Векторизованные с помощью `numru` функции преобразования звука — убирают накладные расходы циклов Python

Результат: каждый экземпляр использует примерно **8-10 %** одного ядра, чего достаточно для декодирования RADE в реальном времени. Если слышны выпадения звука, увеличьте до 2 потоков:

```
RADE_TORCH_THREADS=2 ./start-rx888mk2.sh    # or your receiver's launcher
# standalone: RADE_TORCH_THREADS=2 ./rade.sh restart
```

Одновременных пользователей RADE	Примерная нагрузка на ЦП
1	~9 %
5	~45 %
10	~90 %
20	~180 %

На i5-12450H (12 потоков), где spectrumserver занимает ~42 %, а rx888_stream ~11 %, у вас остаётся около **1000 % запаса** — этого хватит на **20 и более одновременных** пользователей RADE, прежде чем ЦП станет проблемой.

Таблица выше исходит из того, что затраты ЦП растут линейно с числом пользователей. Измерения говорят об обратном. Запуск `rade_loadtest.py` на том же i5-12450H показал нагрузку на слушателя заметно ниже этой таблицы вплоть до ~24 слушателей, а затем резкий рост, причём машина остановилась по **температуре корпуса процессора на 32 слушателях** — не по ЦП и не по полосе пропускания. Считайте эти цифры лишь грубым ориентиром для малых чисел и измеряйте собственную машину: см. [Измерение параллелизма на своём оборудовании](#). Учтите, что нагрузочный тест подаёт в цепочку шум, а не настоящий сигнал RADE, поэтому абсолютная нагрузка при реальном трафике может отличаться; надёжна именно *форма* кривой.

Измерение параллелизма на своём оборудовании

Приведённые выше числа получены на одной конкретной машине. `rade_loadtest.py` (в корне репозитория) измеряет то же самое на **вашей** — он ступенчато наращивает число синтетических слушателей RADE к sidecar и сообщает последнее число слушателей, которое ваша машина выдержала до того, как что-то сдало.

После каждого шага он даёт машине устояться, затем измеряет температуру корпуса процессора, общую загрузку ЦП, доступную ОЗУ, swap и суммарный RSS всех процессов `radae_rxe.py` и `lpcnet_demo`. Он останавливается на первом **изломе** — по тому пределу, который нарушен первым — и печатает последнее устойчивое число.

Требования

```
sudo apt install python3-websockets python3-psutil
```

Запускайте его **на самом хосте SDR** (он читает локальные датчики и память процессов), когда spectrumserver и sidecar RADE уже работают:

```
cd ~/PhantomSDR-Plus
python3 rade_loadtest.py
```

Ctrl-C в любой момент аккуратно закрывает все соединения.

Перед первым запуском

Откройте файл и проверьте два блока в начале:

- **CONFIG** — `WS_URL` по умолчанию `ws://127.0.0.1:8074`; измените его, если перенесли `sidecar` с порта по умолчанию. `RADE_SPS` (12000) и `RADE_SIDEHAND` (USB) должны совпадать с тем, что сообщает ваш веб-интерфейс.
- `handshake_messages()` — единственный отправляемый кадр `init` должен совпадать с инициализацией RADE в `frontend/src/audio.js`. Если это рукопожатие разошлось, каждый слушатель не сможет подключиться и тест сразу завершится ошибками соединения.

Пороги излома

Настройте их в `CONFIG` по вкусу — они намеренно консервативны:

Порог	По умолчанию	Останавливает, когда
<code>TEMP_KNEE_C</code>	85.0	Температура корпуса процессора достигает этого потолка
<code>MIN_AVAIL_MB</code>	800	Доступная ОЗУ опускается ниже этого значения
<code>SWAP_GROWTH_MB</code>	50	Swap вырастает сверх базового уровня, снятого при запуске
<code>FAIL_LIMIT</code>	3	Столько слушателей не смогли подключиться за один шаг
<code>MAX_LISTENERS</code>	60	Жёсткая остановка, даже если излом так и не достигнут

Форму нарастания задают `STEP` (слушателей добавляется за шаг, по умолчанию 2), `SETTLE_S` (25 с на устаканивание перед измерением — именно это позволяет накопиться теплу) и `SAMPLE_S` (окно усреднения ЦП 5 с). Полный прогон до потолка в 60 слушателей занимает поэтому около 15 минут.

Чтение результатов

Каждый шаг печатает строку и добавляет запись в `rade_loadtest.csv`:

Столбец	Значение
<code>listeners</code>	Синтетических слушателей, подключённых на этом шаге
<code>connect_fails</code>	Слушатели, не сумевшие установить или удержать соединение
<code>pkg_c</code>	Температура корпуса процессора, медиана 5 измерений (устойчива к всплескам)
<code>cpu_pct</code>	Загрузка ЦП всей системы , а не на слушателя
<code>avail_mb</code>	Доступная ОЗУ
<code>swap_mb</code>	Используемый swap
<code>dec_rss_mb</code>	Суммарный RSS всех процессов декодирования
<code>dec_procs</code>	Число процессов декодирования — ожидается 2 на слушателя (<code>radac_rxe.py</code> + <code>lpcnet_demo</code>)

Реальный прогон

На упомянутом выше i5-12450H полный прогон завершился так:

```
KNEE at n=32: temp 91.0°C ≥ 85.0
Last stable concurrency: 30 RADE listeners
```

Ограничивающим фактором оказалась температура, причём с большим отрывом. На изломе ещё оставалось 7004 МБ доступной ОЗУ — на 6,2 ГБ выше порога `MIN_AVAIL_MB` — и ноль отказов соединения на каждом шаге.

Два столбца стоит читать внимательно:

`dec_rss_mb` завышает реальные затраты памяти. Он растёт очень ровно, на 295 МБ на слушателя (292, 294, 294 ... 295 на всех 16 шагах), но *доступная* ОЗУ убывает лишь примерно на **202 МБ на слушателя**. Два процесса каждого слушателя разделяют страницы библиотек, а RSS считает эти страницы по разу для каждого процесса. Экстраполируя наклон `avail_mb`, излом по памяти пришёлся бы примерно на **63 слушателя** — за жёсткой остановкой в 60 слушателей, так что на этой машине он никогда не сработает. Оценивайте память по `avail_mb`, а не по `dec_rss_mb`.

`cpu_pct` держится ровно — а потом нет. Он остаётся около 5,3 % от всей машины до 24 слушателей, а затем растёт сверхлинейно:

Слушателей	24	26	28	30	32
<code>cpu_pct</code>	6,7 %	18,4 %	30,1 %	49,6 %	65,2 %

Это десятикратный рост нагрузки при увеличении числа слушателей на треть. Точка перегиба воспроизводится в одном и том же месте от прогона к прогону, поэтому считайте её свойством машины, а не шумом: до перегиба затраты ЦП на слушателя составляют около 3,4 % ядра, а после — около 20 %. Не экстраполируйте показатель ЦП на пользователя, снятый при малом числе слушателей.

У теста нет излома по ЦП. Четыре ограничителя — температура, доступная ОЗУ, рост `swap` и отказы соединений; загрузка ЦП фиксируется, но никогда не завершает прогон. В приведённом прогоне ЦП достиг 65 % и продолжал бы расти, если бы первой не сработала температура. Если вам важен потолок по ЦП, следите за столбцом сами либо добавьте порог.

Что это доказывает, а что нет. Каждый синтетический слушатель передаёт случайный шум малой амплитуды, а не настоящий сигнал RADE. Этого достаточно, чтобы вся цепочка декодирования работала и потребляла ресурсы, поэтому цифры по ресурсам осмысленны — но тест ничего не говорит о *качестве* декодирования или непрерывности звука под нагрузкой. Проверяйте это на реальных слушателях с реальным сигналом.

Если ядра насыщаются раньше излома

RADE жёстко закрепляет по одному потоку на экземпляр, поэтому отдельные ядра могут доходить до ~90 °C, пока общая загрузка ЦП выглядит низкой. `rade_helper.py` распределяет цепочки по ядрам по кругу, чтобы этому противодействовать. Если прогон показывает ранние тепловые изломы, расширьте выделение ядер на клиента:

```
RADE_CORES_PER_CLIENT=3 ./rade.sh restart
```

Править файлы не нужно — это переменная окружения (см. [Переменные окружения](#)); `RADE_PIN_CORES=0` полностью отключает привязку.

Обновление RADE v1

Интеграция с PhantomSDR-Plus (`rade_helper.py`, исправления веб-интерфейса) — это только мост; вся логика декодирования RADE находится в репозитории `radae`. Поэтому обновления почти всегда сводятся к простому `git pull` и пересборке, без изменений в самом PhantomSDR-Plus.

Обычное обновление (новый код, та же модель)

```
# 1. Pull latest radae code
cd ~/radae
git pull

# 2. Rebuild lpcnet_demo (in case C code changed)
cd build
cmake ..
make -j$(nproc)

# 3. Restart the sidecar – no server restart needed
cd ~/PhantomSDR-Plus
./rade.sh restart
```

Это всё. Ни пересборки веб-интерфейса, ни перезапуска сервера, ни правки файлов.

Только новые веса модели

Если вышел новый чекпойнт (например, `model20`) без изменений кода, укажите sidecar новые веса через переменную окружения — править файлы не нужно:

```
# One-off: start with new model
RADE_MODEL=~/.radae/model20/checkpoints/checkpoint_epoch_100.pth ./rade.sh start

# Or permanently – add to your shell profile (~/.bashrc):
export RADE_MODEL=~/.radae/model20/checkpoints/checkpoint_epoch_100.pth
```

Что требуется при каждом виде изменений

Что изменилось в radae	Требуемое действие
Новые веса модели (новый файл <code>.pth</code>)	Задать переменную <code>RADE_MODEL</code> , <code>./rade.sh restart</code>
Изменение кода в <code>radae_rxe.py</code>	<code>git pull</code> , <code>./rade.sh restart</code>
Изменился код C в <code>lpcnet_demo</code>	<code>git pull</code> , пересборка, <code>./rade.sh restart</code>
<code>radae_rxe.py</code> переименован или перемещён	Обновить путь <code>RADAE_RX</code> в <code>rade_helper.py</code> (одна строка)
Аргумент <code>--model_name</code> переименован	Обновить <code>radae_cmd</code> в <code>rade_helper.py</code> (одна строка)
Новая выходная частота дискретизации ($\neq 16000$ Гц)	Обновить <code>SPS_OUT</code> в <code>rade_helper.py</code> + <code>createBuffer()</code> в <code>audio.js</code>
RADE v2 использует другой исполняемый файл	Обновить <code>RADAE_RX</code> в <code>rade_helper.py</code> (одна строка)

Проверка, что обновление сработало

После перезапуска убедитесь, что работает новый код:

```
# Check sidecar picked up new radae_rxe.py
./rade.sh status

# Tail log to see startup lines
tail -20 ~/PhantomSDR-Plus/rade.log
```

В журнале должен быть ожидаемый путь к модели:

```
[RADE] model : /home/sv1btl/radae/model19_check3/checkpoints/checkpoint_epoch_100.pth
```

Как быть в курсе обновлений

Подпишитесь на страницу выпусков репозитория `radae`, чтобы получать уведомления о новых моделях и обновлениях кода:

```
https://github.com/drowe67/radae/releases
```

Следите за блогем FreeDV для анонсов новых сигналов и моделей RADE:

```
https://freedv.org/blog/
```

Разработано и проверено на PhantomSDR-Plus (форк `sv1bt/PhantomSDR-Plus`) с RX-888 MK2 на Ubuntu 24.04. Сервер на C++ не изменяется.

RADE разрабатывают David Rowe VK5DGR и команда FreeDV. См. freedv.org/radio-autoencoder.

Полное удаление и чистая переустановка

Вот полный демонтаж перед повторным запуском `install_rade.sh`:

1. Остановите sidecar `cd ~/PhantomSDR-Plus && ./rade.sh stop`
2. Удалите репозиторий `radae` и сборку `rm -rf ~/radae`
3. Удалите `torch` (установлен в локальный каталог пользователя) `pip3 uninstall -y torch` `rm -rf ~/.local/lib/python3.11/site-packages/torch*`
4. Удалите пакеты Python из `apt` (необязательно — пропустите, если их используют другие программы) `sudo apt-get remove -y python3-numpy python3-scipy python3-matplotlib python3-websockets` `sudo apt-get autoremove -y`
5. Убедитесь, что всё удалено `python3 -c "import torch" 2>&1` # should say `ModuleNotFoundError` `ls ~/radae 2>&1` # should say `No such file or directory`
6. Чистая установка `chmod +x ~/PhantomSDR-Plus/install_rade.sh` `~/PhantomSDR-Plus/install_rade.sh`

ГЛАВА 10

RADE общая установка (Linux)

RADE_General_INSTALL_MANUAL_LINUX.md

RADE v1 — Руководство по ручной установке

Все Linux — Ubuntu / Debian / Fedora / Arch / Raspberry Pi OS · PhantomSDR-Plus

Предусловие: исправленные файлы (`rade_helper.py`, `rade.sh`, `audio.js`, `App.svelte`) уже размещены в дереве каталогов PhantomSDR-Plus. Это руководство выстраивает всё остальное вокруг них.

ВАЖНО

Для обычной установки это руководство не нужно. `./install_rade.sh` в папке PhantomSDR-Plus выполняет за вас каждый шаг ниже — системные пакеты, модули Python, сборку `radea`, проверку весов модели и запуск `sidcar`-а — и одинаково работает на системах с `apt`, `pacman`, `dnf` и `zypper`. `./install.sh` и четыре установщика для дистрибутивов предлагают запустить его в рамках обычной установки. Следуйте этому руководству только для ручной установки, для системы, которую сценарий не покрывает, или чтобы починить отдельный шаг вручную.

Пользователям Raspberry Pi: это и ваше руководство. Raspberry Pi OS — это Debian, поэтому всюду следуйте блокам для Debian и читайте примечания **Raspberry Pi / Bookworm** там, где они встречаются: они охватывают два отличия Pi — PEP 668 и колесо `torch` только для CPU.

Шаг 1 — Системные пакеты

Выберите блок, соответствующий вашему дистрибутиву.

Ubuntu / Debian / Raspberry Pi OS

```
sudo apt update
sudo apt install -y \
    build-essential cmake git \
    python3 python3-pip \
    nodejs npm \
    alsa-utils
```

Fedora / RHEL / Rocky

```
sudo dnf install -y \
    gcc gcc-c++ cmake git \
    python3 python3-pip \
    nodejs npm \
    alsa-utils
```

Arch / Manjaro

```
sudo pacman -Sy --needed \
  base-devel cmake git \
  python python-pip \
  nodejs npm \
  alsa-utils
```

Проверка версии Node.js (все дистрибутивы)

Сборка веб-интерфейса RADE требует Node.js 22 или новее:

```
node --version
```

Если версия ниже 22.x, установите его через `nvm` (любой дистрибутив, без root):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.4/install.sh | bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
nvm install 22 && nvm use 22 && nvm alias default 22
```

Не через NodeSource. `deb.nodesource.com` и `rpm.nodesource.com` теперь отвечают HTTP 403 на любом пути репозитория, поэтому старая строка `curl -fsSL https://deb.nodesource.com/setup_XX.x | sudo -E bash -` больше не работает — а в Debian/Ubuntu она оставляет источник `apt`, который ломает каждый последующий `apt update`. По той же причине скрипты установки используют `nvm`.

Шаг 2 — Пакеты Python

Ubuntu 24.04 и старше / Fedora / Arch

На этих системах работает обычная команда `pip3 install`:

```
pip3 install websockets matplotlib numpy scipy

# torch — CPU-only build (~150–250 MB, avoids the ~3 GB CUDA wheel)
# Use this if the server has no GPU, which is the typical case for a WebSDR
pip3 install torch --index-url https://download.pytorch.org/whl/cpu
```

Если на вашем сервере **всё же** есть видеокарта NVIDIA с установленной CUDA, можно опустить `--index-url` и получить полную сборку с CUDA. Для RADE это не даёт выигрыша в производительности — `radea_rxe.py` использует PyTorch только для матричных операций на CPU.

Ubuntu 23.04+ / Debian Bookworm+ / Raspberry Pi OS (системы с PEP 668)

Эти дистрибутивы блокируют простую команду `pip3 install`. Добавьте флаг:

```
# All packages except torch
pip3 install --break-system-packages websockets matplotlib numpy scipy

# torch — CPU-only build
pip3 install --break-system-packages torch \
  --index-url https://download.pytorch.org/whl/cpu
```

Raspberry Pi / Bookworm — два правила, которые кусаются:

1. Каждой команде `pip3 install` требуется `--break-system-packages` (применение PEP 668). Без этого установка полностью блокируется.
2. Простая команда `pip3 install torch` скачивает сборку с CUDA (~3 ГБ). На Pi нет CUDA — используйте индекс только для CPU, чтобы получить лёгкую сборку ARM64 (~150 МБ).

Как вариант, используйте виртуальное окружение и обойдитесь без флага совсем:

```
python3 -m venv ~/rade-venv
source ~/rade-venv/bin/activate
pip install websockets matplotlib numpy scipy torch \
  --index-url https://download.pytorch.org/whl/cpu
```

При работе с `venv` предваряйте все последующие вызовы `python3` из этого руководства командой `source ~/rade-venv/bin/activate` либо используйте полный путь `~/rade-venv/bin/python3`.

Проверка

```
python3 -c "import websockets, matplotlib, torch, numpy, scipy; print('All OK')"
```

Ожидаемый вывод:

```
All OK
```

Шаг 3 — Клонирование и сборка репозитория `radeae`

Декодер RADE находится в репозитории, отдельном от `codec2`. `freedv_rx` из `codec2` **не** поддерживает RADE v1.

```
# Remove any previous incomplete clone
rm -rf ~/radeae

# Clone
git clone https://github.com/drowe67/radeae.git ~/radeae
cd ~/radeae

# Build
mkdir build && cd build
cmake ..
make -j$(nproc)
```

Убедитесь, что `lpcnet_demo` собран

```
ls -la ~/rade/build/src/lpcnet_demo
```

Ожидается: исполняемый файл присутствует и имеет права на выполнение.

Убедитесь, что веса модели на месте

```
ls ~/rade/model19_check3/checkpoints/checkpoint_epoch_100.pth
```

Ожидается: файл `.pth` в списке. Веса поставляются вместе с репозиторием — отдельная загрузка не нужна.

Шаг 4 — Проверка цепочки декодирования

Этот шаг подтверждает, что вся автономная цепочка работает, прежде чем подключать её к браузеру. Выполняйте из `~/rade`:

```
cd ~/rade
```

4a — Сформируйте тестовый сигнал, закодированный RADE

```
./inference.sh model19_check3/checkpoints/checkpoint_epoch_100.pth \
wav/brian_g8sez.wav /dev/null \
--rate_Fs --pilots --pilot_eq --eq_ls --cp 0.004 \
--write_rx rx.f32 --auxdata
```

Дождитесь завершения. Последние выведенные строки должны быть:

```
loss: 0.741 Auxdata BER: 0.012
```

4b — Декодируйте и прослушайте

```
cat rx.f32 \
| python3 rade_rxe.py \
--model_name model19_check3/checkpoints/checkpoint_epoch_100.pth \
| ./build/src/lpcnet_demo -fargan-synthesis - - \
| aplay -f S16_LE -r 16000
```

Вы должны услышать голос. Вывод показывает захват синхронизации:

```
1 state: search    ...
5 state: sync      ... SNRdB: 4.03 uw_err: 0
Playing raw data 'stdin' : Signed 16 bit Little Endian, Rate 16000 Hz, Mono
```

Сообщения `underrun!!!` во время этой проверки ожидаемы и безвредны. Они появляются потому, что `rade_rxe.py` обрабатывает медленнее, чем идёт файловый ввод-вывод. При живом приёме они не возникают — браузер подаёт звук в реальном времени.

4с — Убедитесь, что sidcar запускается корректно

```
python3 ~/PhantomSDR-Plus/rade_helper.py
```

Ожидаемый вывод (без строк WARNING):

```
[RADE] helper starting on ws://0.0.0.0:8074
[RADE] rade_rx.py      : /home/<user>/rade/rade_rxe.py
[RADE] model          : /home/<user>/rade/model19_check3/checkpoints/checkpoint_epoch_100.pth
[RADE] lpcnet_demo    : /home/<user>/rade/build/src/lpcnet_demo
[RADE] auxdata        : ON (default)
[RADE] torch threads  : 1 per instance (RADE_TORCH_THREADS to override)
[RADE] architecture   : per-connection (each user tunes independently)
[RADE] listening      : waiting for PhantomSDR-Plus clients
```

Нажмите **Ctrl+C** для остановки.

<user> в путях выше — это учётная запись, под которой вы вошли; на стандартном образе Raspberry Pi OS они выглядят как `/home/pi/rade/...`

Шаг 5 — Сборка веб-интерфейса PhantomSDR-Plus

```
cd ~/PhantomSDR-Plus/frontend

# Only if node_modules is missing:
npm install

cd ~/PhantomSDR-Plus
./recompile.sh
```

Следите за ошибками парсера Vite/acorn. Исправленные файлы намеренно избегают `?.`, `??` и пустых `catch {}`, чтобы соответствовать ограничению acorn.

Шаг 6 — Управление sidcar

Отдельный управляющий сценарий для RADE обычно не нужен. Пусковые сценарии (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) сами запускают sidcar, как только сервер поднялся, перезапускают его при завершении и сообщают его состояние; `stop-websdr.sh` останавливает его вместе с сервером. Переходите сразу к шагу 7.

`rade.sh` по-прежнему поставляется для автономного применения — запуска `sidecar` без пускового сценария **PhantomSDR-Plus** или его отдельной проверки. Его команды, журнал и связанные с ним оговорки о стороже описаны в разделе **Управление sidecar**.

Шаг 7 — Откройте порт 8074

Sidecar слушает TCP-порт **8074**. Браузер подключается напрямую к этому порту. Открыть его нужно вручную.

ufw (Ubuntu / Debian / Raspberry Pi OS)

```
sudo ufw allow 8074/tcp && sudo ufw reload
```

firewalld (Fedora / RHEL / Rocky)

```
sudo firewall-cmd --add-port=8074/tcp --permanent
sudo firewall-cmd --reload
```

iptables (любой дистрибутив, постоянно)

```
sudo iptables -I INPUT -p tcp --dport 8074 -j ACCEPT

# Ubuntu / Debian / Raspberry Pi OS — persist across reboots:
sudo apt install iptables-persistent
sudo netfilter-persistent save

# Fedora / RHEL — persist:
sudo service iptables save
```

Маршрутизатор

Добавьте правило NAT/проброса порта: **TCP 8074 → локальный IP сервера : 8074**

Проверка снаружи

Воспользуйтесь <https://portchecker.co> и проверьте порт 8074 по вашему публичному имени узла. **Не** проверяйте с помощью `curl` с самого сервера — NAT hairpin даёт ложные результаты «Connection refused», даже когда порт открыт.

Альтернатива — прокси Nginx (если порт 8074 заблокирован провайдером)

Добавьте внутрь существующего блока `server {}`:

```
location /rade {
    proxy_pass          http://127.0.0.1:8074;
    proxy_http_version 1.1;
    proxy_set_header    Upgrade      $http_upgrade;
    proxy_set_header    Connection  "upgrade";
    proxy_set_header    Host         $host;
    proxy_read_timeout  3600s;
}
```

```
sudo nginx -t && sudo nginx -s reload
```

Затем отредактируйте `audio.js` внутри `setRADEDecoding()`:

```
// Change:
var helperUri = uri || ('ws://' + window.location.hostname + ':8074');
// To:
var helperUri = uri || ('ws://' + window.location.host + '/rade');
```

После этой правки пересоберите веб-интерфейс (`./recompile.sh`).

Шаг 8 — Запуск сервера

Запустите PhantomSDR-Plus как обычно. RADE **не** запускается автоматически:

```
cd ~/PhantomSDR-Plus
./start-rx888mk2.sh      # or start-airspyhf.sh,
                        # start-rtl.sh, start-rspla.sh,
                        # start-fobos-hf.sh, start-fobos.sh,
                        # start-hackrf.sh
```

Используйте лаунчер, соответствующий вашему приёмнику. У каждого свой сторожевой процесс и журнал; `./stop-websdr.sh` останавливает тот, что запущен.

Шаг 9 — Запуск RADE

```
cd ~/PhantomSDR-Plus
./rade.sh start
./rade.sh status
tail -f rade.log
```

Ожидаемый журнал:

```
[RADE] sidecar starting at ...
[RADE] helper starting on ws://0.0.0.0:8074
[RADE] rade_rx.py      : /home/<user>/rade/rade_rxe.py
[RADE] model          : /home/<user>/rade/model19_check3/checkpoints/checkpoint_epoch_100.pth
[RADE] lpcnet_demo     : /home/<user>/rade/build/src/lpcnet_demo
[RADE] auxdata         : ON (default)
[RADE] torch threads  : 1 per instance
[RADE] architecture   : per-connection (each user tunes independently)
[RADE] listening      : waiting for PhantomSDR-Plus clients
```

Шаг 10 — Использование RADE в браузере

1. Откройте веб-интерфейс PhantomSDR-Plus
2. Найдите активные станции на <https://qso.freedv.org>
3. Настройтесь на частоту станции по шкале
4. В **Decoder Options** выберите:
5. **RADE v1 — RADEL (LSB)** для 40 м / 80 м / 160 м (≤ 10 МГц)

6. **RADE v1 — RADEU (USB)** для 20 м / 17 м / 15 м / 12 м / 10 м (> 10 МГц)

7. Нажмите **Decoder: ON**

Шаги 4 и 5 можно заменить одним нажатием: кнопки **RADEL / RADEU** расположены рядом с заголовком **Modes selector**, а также внутри всплывающих окон **Modes** и **Bands**. Одно нажатие выбирает декодер, включает его и прокручивает панель RADE в поле зрения; повторное нажатие выключает RADE. См. [руководство по RADE](#).

Индикатор	Значение
Красный — «Connecting to sidecar...»	Порт 8074 недоступен или sidecar не запущен
Жёлтый — «Searching for signal...»	Sidecar подключён, кадр RADE ещё не обнаружен (дайте ~1,5 с)
Зелёный — «Synced · SNR x.x dB»	Идёт декодирование — воспроизводится речь

Замечание о нагрузке на процессор

Каждый пользователь RADE потребляет около 8-10 % одного ядра (число потоков PyTorch ограничено значением 1 в sidecar). Если слышны выпадения звука, увеличьте до 2 потоков:

```
RADE_TORCH_THREADS=2 ./rade.sh restart
```

Одновременных пользователей	Примерная нагрузка на ЦП
1	~9 %
5	~45 %
10	~90 %
20	~180 %

Эти цифры измерены на x86_64. Raspberry Pi декодирует RADE нормально, но затраты на пользователя выше, и таблица выше не переносится — измерьте свою плату командой `top`, пока пользователь синхронизирован, прежде чем объявлять предел числа пользователей.

Обновление RADE в дальнейшем

```
cd ~/radae
git pull
cmake -S . -B build && make -C build -j$(nproc)

cd ~/PhantomSDR-Plus
./rade.sh restart
```

Пересборка веб-интерфейса и перезапуск сервера не нужны, если не изменился сам `rade_helper.py`.

Проверено на Ubuntu 24.04 (x86_64) и Raspberry Pi 4 / Raspberry Pi OS Bookworm (ARM64, Python 3.11). Форк *PhantomSDR-Plus*: [sv1btl/PhantomSDR-Plus](https://github.com/sv1btl/PhantomSDR-Plus). RADE разработан David Rowe VK5DGR и командой FreeDV — <https://freedv.org>

ГЛАВА 11

Редактирование вариантов

EDITING_VARIANTS.md

PhantomSDR-Plus — Как редактировать варианты фронтенда

(после того как четыре файла `App__*_smeter_.svelte` были объединены в один `App.svelte`)

1. Что такое варианты

Два независимых выбора, дающие четыре комбинации:

<code>smeter = "analog"</code>	прибор со стрелкой
<code> = "digital"</code>	сегментированный столбчатый прибор
 <code>layout = "v1"</code>	 селектор режимов сверху, селектор диапазонов снизу
<code> = "v2"</code>	селектор диапазонов сверху, селектор режимов снизу (также переносит ряд кнопок точной настройки на телефоне)

Все четыре живут в **одной** сборке, которая отдаётся по `/`. Селектор ☿ в правом верхнем углу переключает их прямо в работающей странице: он присваивает пропсы `smeter/layout` в `App.svelte`, поэтому ничего не перезагружается, а звук, водопад и декодеры продолжают работать. Выбор сохраняется для каждого браузера в `localStorage` под ключом `phantom.variant` и при следующем визите имеет приоритет над умолчанием сборки.

Поэтому вариант, который сисоп выбирает в `recompile.sh`, — это **стартовый** вариант: то, что видит посетитель при первом заходе, пока не тронул меню.

URL	что это
<code>/</code>	desktopная страница; все четыре варианта, переключаемые во время работы
<code>/mobile/</code>	отдельная страница, собирается <code>build-mobile.sh</code> — не вариант

До 10.08.2026 каждый вариант был отдельной сборкой в `/analog/`, `/digital/`, `/v2-analog/` и `/v2-digital/`, а селектор переходил между ними — поэтому переключение перезагружало страницу и обрывало звук. Этих сборок больше нет; по каждому из четырёх путей остаётся лишь маленький `index.html`, перенаправляющий на `/` — его пишет `frontend/make-redirect-stubs.sh` при каждой сборке, чтобы старые закладки продолжали работать.

Вариант **не** записывается ни в один исходный файл. `src/main.js` — неизменная точка входа, передающая компоненту два значения, подставляемые на этапе сборки:

```
const app = new App({
  target: document.getElementById('app'),
  props: { smeter: __PHANTOM_SMETER__, layout: __PHANTOM_LAYOUT__ }
})
```

`vite.config.js` заполняет их при сборке, беря первое из заданного:

1. `PHANTOM_SMETER` / `PHANTOM_LAYOUT` — переменные окружения. С исчезновением отдельных сборок вариантов их никто не выставляет; они по-прежнему работают для разовой сборки с другим стартовым вариантом.
2. `frontend/variant.json` — умолчание сайта, записываемое `recompile.sh`. Сборка корня (/) вариант не передаёт, поэтому попадает сюда.
3. `analog` / `v1` — запасное значение.

Эти два значения — лишь начальные. `App.svelte` перезаписывает их при инициализации из `localStorage`, если посетитель уже выбрал вариант, а селектор `☞`присваивает их заново при каждом переключении.

Внутри `App.svelte` пропсы превращаются в два флага в строках 20-21:

```
$: isAnalog = smeter !== "digital";
$: isV2      = layout === "v2";
```

Используйте эти два флага — никогда `smeter/layout` напрямую — чтобы весь файл читался единообразно.

2. Что делает каждый файл

`frontend/src/main.js` — 12 строк

Точка входа. Ничего, кроме создания `App` и двух значений `__PHANTOM_*__`, которые она передаёт как пропсы. Файл **неизменен** — ни `recompile.sh`, ни скрипты сборки его больше не переписывают, и именно это позволяет собирать варианты одновременно.

Не размещайте здесь код приложения. Это файл точки входа, а не место для логики.

`frontend/src/App.svelte` — ~17130 строк

Всё приложение: водопад, настройка, декодеры, закладки, панели, десктопная компоновка И адаптивная компоновка для телефона. Почти любая правка, которую вы когда-либо сделаете, идёт сюда.

`frontend/src/lib/SMeterAnalog.svelte` — 824 строки

Стрелочный прибор целиком: отрисовка шкалы, стрелка, переключение светлой/тёмной шкалы и его ключ в `localStorage`, а также сглаживание стрелки.

- Входной проп: `dbm` — КАЛИБРОВАННАЯ мощность в dBm.
- Также принимает: `mobile` — уменьшенные атрибуты `canvas` для компоновки телефона.
- Регулировка: `SMOOTH_TIME_MS` в строке 55. Меньше = быстрее стрелка, больше = плавнее. Сейчас 16 (один кадр при 60 fps, фактически нижний предел).

Три вида шкалы — и `smeter_theme.sh`

У аналогового прибора три фона, все рисуются одним и тем же кодом:

<code>dark</code>	исходный тёмный шлифованный металл
<code>amber</code>	светлая, бледно-серая шкала
<code>vintage</code>	тёплая шкала цвета состаренного янтаря

Не путайте это с вариантом `smeter = analog / digital` выше. Вариант выбирает, какой компонент прибора показывается; вид шкалы выбирает, как отрисовывается аналоговый. Они хранятся под разными ключами `localStorage` (`phantom.variant` и `smeterTheme`) и задаются в разных местах.

Сторона посетителя. Канвас прибора — это кнопка (`role="button"`, а также Enter/Пробел с клавиатуры, с подсказкой «click to switch style» под ним). Каждый щелчок переключает `dark` → `amber` → `vintage` и сохраняет выбор в `localStorage.smeterTheme`. Он считывается при каждой загрузке страницы, поэтому выбор посетителя переживает обновления и перезапуски браузера неограниченно долго. Это намеренно **не** cookie: cookie отправлялся бы на сервер при каждом запросе водопада и звука ради значения, которое сервер никогда не читает.

Сторона сисопа. Два места в `SMeterAnalog.svelte` определяют, что получают все остальные:

Строка	Что делает
<code>let smeterTheme = 'dark';</code>	Шкала, которую получает браузер без сохранённого выбора .
<code>const SMETER_PREF_VERSION = 2;</code>	Счётчик. Если сохранённая версия браузера отстаёт, он один раз забывает сохранённый вид и принимает значение по умолчанию выше. Увеличьте его, чтобы перевести посетителей, которые уже выбрали.

Изменение одного лишь значения по умолчанию поэтому доходит только до совершенно новых посетителей. Оба изменения вместе переводят всех — именно для этого есть `smeter_theme.sh` в корне репозитория:

```
./smeter_theme.sh           # показать текущее значение, затем выбрать из меню
./smeter_theme.sh vintage   # задать сразу, затем предложить пересборку
./smeter_theme.sh dark --build # запустить frontend/build-all.sh без вопросов
./smeter_theme.sh amber --no-reset # только новые посетители — существующих не трогать
```

Скрипт сообщает текущее значение, задаёт новое, увеличивает счётчик сброса (всегда вверх — **никогда не уменьшайте его**, меньшее число просто не даст сбросу сработать) и затем предлагает тот же выбор сборки фронтенда, что и `recompile.sh`. Он **не** трогает бэкенд и **не** переписывает `variant.json`.

Изменение доходит до подключённого слушателя только при перезагрузке его страницы — значение по умолчанию скомпилировано в JS-бандл, а сброс выполняется при загрузке. Живой рассылки нет. Обычной перезагрузки достаточно, так как `vite` добавляет хеш в имена файлов бандла.

Однократный сброс через счётчик перекрывает всех, включая посетителей, которые сознательно выбрали свою шкалу щелчком. Это сделано намеренно: только так можно вернуть весь сайт к единому виду. Используйте `--no-reset`, если предпочитаете их не трогать.

frontend/src/lib/SMeterDigital.svelte — 184 строки

Сегментированный столбчатый прибор целиком.

- Входные пропсы: `rawDb` — СЫПОЕ значение `audio.getPowerDb()`
- `smeterOffset` — `audio.smeter_offset`
- `mobile` — уменьшенные атрибуты `canvas`

ПРИМЕЧАНИЕ: этот прибор намеренно питается СЫРОЙ мощностью плюс смещение, а НЕ калиброванным dBm, который использует аналоговая шкала. У него всегда было собственное отображение. Подача калиброванного значения сдвинула бы каждое показание на каждой цифровой станции. Не «унифицируйте» эти два входа.

`frontend/src/lib/BandSelector.svelte` — 69 строк

Сетка кнопок диапазонов. Отрисовывается в `App.svelte` дважды (по одному разу на компоновку).

`frontend/src/lib/ModesSelector.svelte` — 75 строк

Сетка кнопок режимов. Отрисовывается в `App.svelte` дважды.

`frontend/src/lib/StatusIndicators.svelte` — 122 строки

Индикаторы состояния (mute, squelch, NR, NB, NS, AN, CTCSS). Принимает проп `wide`: цифровая компоновка использует более широкие индикаторы, чем аналоговая. Оба класса Tailwind (`w-8` и `w-10`) выписаны в компоненте полностью — сканер Tailwind видит только буквальное имена классов, поэтому никогда не собирайте их конкатенацией строк.

Все остальные файлы в `frontend/src/lib/` (Spectrogram, PassbandTuner, FrequencyInput, VersionSelector, ...) появились до этой работы и не изменялись.

3. Изменение, затрагивающее только часть вариантов

Оберните его во флаг. В разметке:

```
{#if isAnalog}
  ...это видят только аналоговые станции...
{:else}
  ...это видят только цифровые станции...
{/if}

{#if isV2} ... {/if}           {#if !isV2} ... {/if}
```

В атрибуте `class`:

```
class="p-4 {isAnalog ? 'text-xs' : 'text-sm'} rounded-md"
```

Выписывайте оба варианта полными литералами, как выше. Tailwind сканирует исходники на полные имена классов; `text-{size}` или составленное имя вычищается из CSS, и стиль молча исчезает.

В блоке `<script>`: старайтесь вообще не ветвиться. Скрипт намеренно вычисляет входы ОБОИХ приборов на каждом такте —

```
smeterDbm      = powerDb;           // аналоговая шкала
smeterRawDb    = audio.getPowerDb(); // цифровая шкала
smeterOffset   = audio.smeter_offset; // цифровая шкала
```

— так что выбирать приходится только разметке. Именно поэтому в файле на 17000 строк всего 15 переключателей вариантов. Так и оставьте; именно ветвление на уровне скрипта развело четыре старых файла.

4. Где находятся переключатели вариантов

Пятнадцать мест в App.svelte. Номера строк смещаются при правках — надёжный способ найти их все:

```
cd frontend/src && grep -n 'isAnalog\|isV2' App.svelte
```

На момент написания:

Строка	Флаг	Что переключает
20		объявление isAnalog
21		объявление isV2
8062	isV2	десктоп: селектор диапазонов первым (порядок v2)
8090	isV2	десктоп: размер заголовка панели
8686	isV2	десктоп: селектор режимов первым (порядок v1)
8717	isAnalog	минимальная ширина панели прибора (стрелка уже)
8730	isAnalog	только аналог: строка даты/времени над прибором
8819	isAnalog	только аналог: индикаторы состояния + разделитель
8834	isAnalog	только цифра: строка времени, широкие индикаторы и сам выбор <SMeterDigital> / <SMeterAnalog>
8861	isAnalog	вертикальные отступы блока частоты
8864	isAnalog	верхний отступ блока точной настройки
12288	isV2	телефон: ряд точной настройки сверху (порядок v2)
12347	isAnalog	телефон: какой прибор отрисовывать
12433	isV2	телефон: ряд точной настройки снизу (порядок v1)
13204	isAnalog	размер текста кнопки noise gate

Всё остальное в файле общее для всех четырёх вариантов.

5. Другие полезные регулировки

Место	Константа	Назначение
App.svelte строка 5034	<code>const visualGain = 1.1;</code>	Калибровка аналогового прибора. Каждые 0,10 — примерно 5 dBm показания.
SMeterAnalog.svelte строка 55	<code>const SMOOTH_TIME_MS = 16;</code>	Постоянная времени сглаживания стрелки, в миллисекундах.
SMeterAnalog.svelte строка 64	<code>let smeterTheme = 'dark';</code>	Шкала аналогового прибора по умолчанию для браузера без сохранённого выбора. Задавайте её через <code>./smeter_theme.sh</code> , а не вручную.
SMeterAnalog.svelte строка 69	<code>const SMETER_PREF_VERSION = 2;</code>	Счётчик однократного сброса сохранённой шкалы. Увеличьте, чтобы перевести существующих посетителей на значение по умолчанию; только растёт.
SMeterDigital.svelte строка 149	<code>const DIGITAL_BAR_TRIM = 0;</code>	Сдвигает шкалу на целые сегменты.
SMeterDigital.svelte строка 31	<code>const numberOfDots = 35;</code>	Число сегментов шкалы.

6. После правки — пересборка

Проще всего из корня репозитория:

```
./recompile.sh
```

Пункт меню **[2]** пересобирает фронтенд и спрашивает, какой вариант должен увидеть посетитель при первом заходе. Выбор записывается в `frontend/variant.json` — ни один исходный файл не копируется и не переписывается. (Раньше он заодно заново создавал `VersionSelector.svelte` из heredoc; это убрали, когда селектор перестал переходить по ссылкам, иначе при каждом запуске переключение во время работы откатывалось бы.) Пункт **[1]** — только бэкенд, **[3]** — оба.

Или напрямую, из `frontend/`:

```
./build-all.sh      собирает десктопную страницу И /mobile/ &lt;-- обычно используйте его
./build-default.sh   только десктопную страницу
./build-mobile.sh    только /mobile/
```

`build-default.sh` позволяет vite очистить `dist/`, из-за чего `dist/mobile/` исчезает до следующего `build-all.sh` или `build-mobile.sh`. Поэтому обычный выбор — `build-all.sh`.

`build-all.sh` содержит собственную копию логики сборки (`build_version SMETER LAYOUT NAME OUTDIR BASE LOG`), а не вызывает `build-default.sh`. Если вы меняете способ сборки страницы, меняйте его в ОБОИХ местах, иначе два пути разойдутся.

Несколько деталей, которые стоит знать:

- Он выставляет `PHANTOM_KEEP_OUTDIR=1`, чтобы Vite не очищал выходной каталог, и чистит `dist/` сам — удаляет содержимое, но сохраняет `users.json`, который работающий `spectrumserver` переписывает при каждом подключении и отключении слушателя.

- `/mobile` он собирает последним, когда десктопный вывод уже готов.
- Механизм параллельной сборки (`PHANTOM_BUILD_JOBS`, по умолчанию 3) остался с тех пор, когда десктопных сборок было пять. С одной ничто не ждёт в очереди.

Биты исполнения: `recompile.sh` выполняет `chmod +x` на скрипте перед запуском, а `build-all.sh` намеренно вызывает `bash build-mobile.sh`, а не `./build-mobile.sh`, поэтому отсутствующий бит больше не ломает сборку. Загрузка скрипта через веб-интерфейс GitHub всё же сбрасывает права на 644, поэтому если запускаете какой-то напрямую:

```
chmod +x frontend/build-*.sh
```

7. Как проверить правку, прежде чем ей доверять

Зелёный `vite build` — **НЕ** достаточно. Сборка не видит идентификатор, потерявшийся при переносе кода между файлами: это `ReferenceError` во время выполнения, и страница просто не загружается. Такое случалось дважды.

1) Проверка неопределённых идентификаторов (ловит именно этот отказ)

```
cd frontend
npx eslint --no-eslintrc \
  --parser svelte-eslint-parser \
  --rule '{"no-undef":"error"}' \
  --env browser,es2022 \
  src/App.svelte src/lib/SMeter*.svelte
```

Ожидается одно известное безобидное срабатывание:

```
'dBmCalOffset' is not defined
```

`typeof x === 'number'` для необъявленного имени никогда не выбрасывает исключение, так что это выражение уже даёт 0. Всё остальное — настоящая проблема.

2) Соберите, затем ОТКРОЙТЕ СТРАНИЦУ И ПРОЙДИТЕ ВСЕ ЧЕТЫРЕ ВАРИАНТА

Не только тот, который меняли: правка под `isAnalog/isV2` может сломать остальные три, пока ваш работает. Откройте `/`, пройдите все четыре пункта в меню ☰, затем откройте `/mobile/`.

Откройте консоль браузера на каждой. Пустая или наполовину отрисованная страница с `ReferenceError` в консоли — характерный признак описанной выше ошибки.

3) Никогда не оставляйте `frontend/dist/` в частичном состоянии

Голый `npx vite build` очищает выходной каталог и уносит с собой `dist/mobile/` и живой `dist/users.json` сервера — мобильная страница отвечает 404 до следующей успешной сборки. Всегда собирайте через скрипты: `build-all.sh` выставляет `PHANTOM_KEEP_OUTDIR=1` и делает очистку сам, сохраняя `dist/users.json`. Если сомневаетесь, просто запустите `./build-all.sh` заново.

Руководство пользователя

USER_GUIDE.md

Руководство пользователя PhantomSDR-Plus

Добро пожаловать в PhantomSDR-Plus! Это руководство поможет вам получить максимум от прослушивания через WebSDR.

Содержание

1. Введение
 2. Начало работы
 3. Обзор интерфейса
 4. Основные операции
 5. Дополнительные возможности
 6. Виды демодуляции
 7. Цифровые декодеры
 8. Горячие клавиши
 9. Закладки
 10. Использование на мобильных устройствах
 11. Советы и рекомендации
 12. Устранение неполадок
 13. Часто задаваемые вопросы
-

Введение

Что такое PhantomSDR-Plus?

PhantomSDR-Plus — это программно-определяемое радио (SDR) с веб-интерфейсом, позволяющее слушать радиосигналы через Интернет. С вашей стороны не требуется ни специального программного обеспечения, ни оборудования — достаточно современного браузера!

Что можно слушать?

В зависимости от конфигурации WebSDR вы можете настроиться на:

- **Любительскую радиосвязь:** радиолюбителей со всего мира
- **Вещательные станции:** радио AM/FM, коротковолновое вещание
- **Авиацию:** управление воздушным движением, связь с воздушными судами
- **Морскую связь:** связь «судно — берег», морские метеосводки
- **Метеоспутники:** NOAA, METEOR-M
- **Цифровые виды связи:** FT8, RTTY, PSK31 и другие
- **Служебные станции:** сигналы точного времени, военные, государственные

Системные требования

- **Браузер:** Chrome/Edge (рекомендуется), Firefox, Safari
 - **Соединение:** широкополосный Интернет (рекомендуется от 1 Мбит/с)
 - **Звук:** исправные динамики или наушники
 - **Необязательно:** мышь с колесом прокрутки для удобной настройки
-

Начало работы

1. Откройте WebSDR

Откройте браузер и перейдите по адресу WebSDR, который сообщил оператор.

Пример: `http://websdr.example.com:9002`

2. Первая загрузка страницы

При загрузке страницы вы увидите:

- цветной водопад, показывающий радиоактивность
- панель управления с индикатором частоты и кнопками
- S-метр, показывающий уровень сигнала
- счётчик пользователей

3. Начните слушать

1. **Щёлкните по сигналу** на водопаде
 2. **Звук начнётся автоматически**
 3. **Отрегулируйте громкость** с помощью регулятора браузера или ползунка на экране
 4. **Подстройте частоту**, щёлкнув точно по сигналу
-

Обзор интерфейса

Основные элементы

WebSDR <callsign>, located in <QTH> ⌘ [Analog S-Meter ▼] email · Keyboard Shortcuts · Users · Stats · Other servers 1 2 3 Frequency Search: MW SW HamDash PSK Reporter WSPRnet · Callsign Search [Open Additional Info]			
Spectrum + Waterfall band-plan strip along the bottom			
Audio & Buffer volume · SQ · zoom AGC Auto Fast Med Slow Compressor Equalizer	Frequency 7,120.00 band · VFO	S-Meter dBm dBµV SNR · NF	Waterfall Controls Min · Max · zoom colour-map strip Zoom In Out Max Min
Filters NR NB NS AN CTCSS Modes selector ... RADEL RADEU USB LSB CW AM QUAM FM Band selector	Fine Tuning (kHz) VFO A · Modes · Bands · IF Filters Wheel Tuning Steps Decoders FT8 FT4 FT2 JS8 CW WSPR FAX SSTV NAVTEX RTTY ...	Waterfall Spectrum Auto Adj. Height [profile ▼] [Bookmarks]	

Три панели под водопадом прокручиваются как одна страница; в узком окне они выстраиваются друг под другом, а не рядом. **RADEL** и **RADEU** стоят на самой строке заголовка *Modes selector*, справа от него.

1. Водопад

Водопад — это визуальное представление радиосигналов:

- **Горизонтальная ось:** частота
- **Вертикальная ось:** время (прокручивается вниз)
- **Цвета:** уровень сигнала
- **Тёмно-синий/чёрный:** сигнала нет (уровень шума)
- **Зелёный/жёлтый:** слабые и средние сигналы
- **Оранжевый/красный:** сильные сигналы
- **Белый:** очень сильные сигналы

2. Индикатор частоты

Показывает текущую частоту настройки в различных форматах:

- **МГц:** 7.100.000 MHz (КВ-диапазоны)
- **кГц:** 14200 kHz
- **Гц:** 145500000 Hz (УКВ)

3. S-метр (аналоговый или цифровой)

Отображает уровень сигнала:

- **S1-S9:** стандартная шкала уровня сигнала
- **+dB:** сигналы выше S9 (например, S9+20dB)
- **Аналоговый или цифровой:** в зависимости от конфигурации

4. Кнопки видов модуляции

Выберите вид демодуляции:

- **AM**: амплитудная модуляция
- **FM**: частотная модуляция
- **USB**: верхняя боковая полоса
- **LSB**: нижняя боковая полоса
- **CW**: телеграф (незатухающие колебания)
- **WBFM**: широкополосная ЧМ (вещание)
- **QUAM**: стереофоническое AM C-QUAM — надпись зеленеет при обнаружении стереопилота

Рядом с заголовком **Modes selector** расположены кнопки **RADEL** и **RADEU**, запускающие цифровой голос RADE v1 одним нажатием — см. [Цифровые декодеры](#).

5. Панель управления

Дополнительные органы управления:

- **AGC**: автоматическая регулировка усиления
- **NR**: шумоподавление (спектральное)
- **NB**: подавитель импульсных помех
- **NS**: подавление фонового шума
- **AN**: автоматический нотч
- **CTCSS**: шумоподаватель по субтону (FM). Нажатие открывает рядом с кнопкой небольшое окно с *Any tone* и 49 стандартными субтонами (67,0–254,1 Гц); после выбора кнопка показывает тон (например 88.5 или ANY), и звук проходит только пока принимается этот субтон — открытие занимает около 0,7 с. Повторное нажатие выключает CTCSS; Escape или щелчок вне окна закрывает его без изменений. Закладки сохраняют выбранный тон.
- **SQL**: шумоподаватель (squelch)
- **AI**: шумоподавление на основе ИИ (только речь)
- **Zoom**: масштаб водопада
- **Wheel Tuning Steps**: шаг перестройки колёсиком мыши
- **Decoders**: кнопки, запускающие и останавливающие каждый декодер одним нажатием

6. Наложение band plan

Цветные полосы, показывающие распределение частот:

- разные цвета для разных служб
- помогают понять, что разрешено на каждой частоте

Основные операции

Настройка на частоту

Способ 1: щелчок по водопаду

1. Щёлкните прямо по сигналу на водопаде
2. Приёмник настроится на эту частоту
3. Начнётся воспроизведение звука

Способ 2: ввод частоты

1. Щёлкните по индикатору частоты
2. Введите нужную частоту
3. Нажмите Enter

Примеры:

- 7100 → 7.100 МГц
- 14200.0 → 14.200 МГц
- 145.500 → 145.500 МГц

Способ 3: колесо мыши

1. Наведите курсор на индикатор частоты
2. Прокрутка вверх — увеличение частоты
3. Прокрутка вниз — уменьшение частоты

Способ 4: кнопки шага

1. Используйте кнопки ▲ и ▼ рядом с частотой
2. Величина шага зависит от вида модуляции:
3. **AM/FM**: шаг 1 кГц
4. **SSB**: шаг 100 Гц
5. **CW**: шаг 10 Гц

Сканер

Сканер ведёт приёмник по диапазону каналов и останавливается на первом, где есть сигнал. Он находится в строке **Fine Tuning (kHz)**, справа:



Орган управления	Что делает
◀ ▶	Сканировать вниз / вверх. Если сканер стоит на сигнале, стрелка продолжает поиск.
■	Остановить.
⊙	Исключить этот канал из поиска — для постоянной несущей или помехи.
30 dB · 4 ▼	Порог, рядом — измеряемый сейчас уровень. Открывает настройки.

Текст слева показывает, что делает сканер: Scanner в покое, достигнутую частоту во время поиска, ● 3s во время отсчёта до продолжения, ● 30·22s пока канал всё ещё занят, ● hold когда он останется на месте.

Настройки

- **Range** — *Scan Band* обходит диапазон, в котором начался поиск; *Scan Visible* обходит ровно то, что показывает водопад, и следует за ним при масштабировании или перетаскивании.
- **Scan** — *Every channel* настраивается и слушает каждый канал по очереди; *Skip empty* читает спектр и сразу переходит к сигналам.
- **Stop at** — насколько выше шумового порога диапазона должен быть канал, чтобы поиск остановился, в дБ. Шум отслеживается непрерывно, поэтому одна и та же настройка годится днём и ночью. Пустой канал показывает не 0 дБ — посмотрите мгновение на значение на кнопке и выберите порог выше.

- **Resume after** — сколько канал должен оставаться тихим, прежде чем поиск пойдёт дальше. Паузы в речи его не перезапускают. *Hold* остаётся на месте, пока вы не нажмёте кнопку.
- **Max stay** — идёт дальше по истечении этого времени, даже если сигнал ещё есть, чтобы постоянная несущая не держала поиск вечно.

Шаг следует виду модуляции — 1 кГц в SSB, 0,1 кГц в CW, 5 кГц в AM, 9 или 10 кГц на средних волнах и 9 кГц на длинных — а остановки попадают на сетку каналов. Поиск остаётся внутри своего диапазона и того, что может принять приёмник; на краю он продолжает с другого конца.

Настройки и исключённые каналы запоминаются в вашем браузере.

Выбор вида демодуляции

Выберите вид, соответствующий сигналу:

Для голосовой связи:

- **AM**: авиация, AM-вещание, часть любительской связи
- **FM**: УКВ-ретрансляторы, FM-вещание
- **USB**: любительская KB-связь (20 м, 17 м, 15 м, 12 м, 10 м)
- **LSB**: любительская KB-связь (160 м, 80 м, 40 м, 30 м)

Для данных/цифровых видов:

- **USB**: большинство цифровых видов связи (FT8, PSK31, RTTY)
- **LSB**: некоторые цифровые виды на низкочастотных KB-диапазонах

Для телеграфа:

- **CW**: телеграфные сигналы (азбука Морзе)

Регулировка громкости

- **Ползунок на экране**: перетащите регулятор громкости
- **Громкость браузера**: используйте медиаэлементы браузера
- **Системная громкость**: отрегулируйте громкость компьютера
- **Клавиатура**: клавиши + и - (если поддерживаются)

Использование S-метра

S-метр показывает уровень сигнала:

- **S0-S3**: очень слабый сигнал, тяжело принимать
- **S4-S6**: слабый или удовлетворительный сигнал
- **S7-S9**: хороший или сильный сигнал
- **S9+**: чрезвычайно сильный сигнал

Совет: для лучшего звука настраивайтесь на сигналы с уровнем S7 и выше.

Смена вида шкалы: на аналоговом (стрелочном) S-метре щёлкните по самому прибору — или выделите его и нажмите Enter либо Пробел — чтобы перебрать три фона: тёмный шлифованный металл, светлую бледно-серую шкалу и тёплую янтарную шкалу в винтажном стиле. Браузер запоминает выбор, поэтому он сохраняется после обновления страницы или перезапуска. Настройка действует для каждого браузера и каждого адреса: открытие приёмника по имени хоста и по IP даёт две отдельные настройки, а приватное окно всегда начинается со значения по умолчанию для сайта.

Выше 30 МГц S-метр следует стандарту УКВ IARU Region 1: там S9 соответствует –93 дБм вместо –73 дБм на КВ, по-прежнему шесть дБ на балл S. Стрелка и полоса переключаются сами вместе с частотой настройки; значения в дБм и дБмкВ всегда показывают измеренный уровень. Некоторые станции к тому же настраивают свои приёмники УКВ так, чтобы приборы **оставались на 0 на пустом канале**, как у трансивера УКВ: выше 60 МГц стрелка и полоса тогда поднимаются, только когда сигнал выделяется из шума. Значения дБм, дБмкВ, SNR и NF всегда показывают реальные величины.

Выбор приёмника

У некоторых станций больше одного приёмника — например, один для КВ и один для диапазона 2 м. В шапке страницы они показывают строку *Receivers*: с кнопкой для каждого приёмника; тот, на котором вы находитесь, показан жёлтым. Нажатие другой кнопки открывает страницу того приёмника, с его собственным водопадом, диапазонами, чатом и списком слушателей. На странице /mobile те же кнопки находятся в верхней панели. Станция с одним приёмником такой строки не показывает.

Дополнительные возможности

Автоматическая регулировка усиления (AGC)

APU автоматически регулирует уровень звука:

- **Off**: без автоматической регулировки
- **Slow**: плавные изменения уровня (лучше для SSB)
- **Medium**: сбалансированный отклик
- **Fast**: быстрая регулировка (лучше для AM)

Рекомендация: начните с «Fast» для AM и «Slow» для SSB.

Четыре регулятора шума

NR, NB, NS и AN — отдельные переключатели, каждый против своего рода шума. Они независимы: включение одного не включает другой, и их можно свободно сочетать.

Ни один из них не доходит до декодеров: FT8, CW, WSPR, SSTV, FAX, NAVTEX, RTTY/PSK31/Olivia и граббер QRSS читают звук *до* этих фильтров, так что настраивать их можно чисто на слух, не влияя на то, что декодируется. См. [руководство по декодерам](#).

Шумоподавление (NR)

Спектральное шумоподавление. Оценивает уровень шума в каждой части звукового спектра и приглушает эти части, оставляя нетронутым то, что поднимается над шумом.

Применяйте, когда: слышно ровное шипение или белый шум за сигналом.

Постоянные тоны — нота CW, несущая — распознаются как сигнал и защищаются, так что NR не съедает сигнал CW, как это сделал бы наивный фильтр. Типичный результат: шум между словами ниже на 10 дБ ценой нескольких десятых долей дБ на самой речи.

Подавитель импульсных помех (NB)

Убирает импульсные помехи: щелчки, трески, атмосферные разряды, помехи от зажигания и линий электропередачи. Он следит за огибающей звука и глушит только те отсчёты, которые выбиваются далеко над ней — около одной миллисекунды на разряд, с плавными фронтами, чтобы само глушение не щёлкало.

Применяйте, когда: слышны щелчки или трески от линий электропередачи, моторов, гроз или зажигания автомобилей.

Режекторы сетевого фона 50 Гц и 60 Гц следуют за этой кнопкой.

Подавление фонового шума (NS)

Несколько секунд измеряет собственный порог шума диапазона и постоянной величиной приглушает всё, что лежит на нём, — глубже на высоких звуковых частотах, чем на низких. Там, где NR реагирует от мгновения к мгновению, NS — медленная, твёрдая рука: она опускает шипение диапазона, не меняя звучания самого сигнала.

Применяйте, когда: диапазон спокоен, но шипит, и вы хотите опустить шум без «подводного» призыва агрессивного NR.

Работает только в USB, LSB и AM — CW, FM и цифровые режимы остаются нетронутыми. Порог измеряется заново при каждой смене частоты или режима и устанавливается за несколько секунд.

Автоматический нотч (AN)

Сам находит и убирает постоянные мешающие тоны — гетеродинные свисты, несущие, «птички» — без ручной установки режектора. Он подстраивается непрерывно, поэтому убирает несколько тонов сразу и удерживает подавленным уходящий по частоте тон.

Применяйте, когда: поверх нужного сигнала слышен свист или тон.

В CW обходится стороной: там полезный сигнал и есть постоянный тон.

О задержке: NR и NS во включённом состоянии добавляют примерно по 40 мс задержки звука (оба вместе — около 80 мс). NB и AN не добавляют ничего. Это касается только прослушивания, но никогда декодеров.

Автоматический шумоподаватель (SQL)

Отключает звук при отсутствии сигнала:

- **Off:** звук всегда включён (слышен шум)
- **Auto:** порог задаётся автоматически
- **Manual:** порог настраивается вручную

Применяйте, когда: наблюдаете за частотой в ожидании активности.

Шумоподавление на основе ИИ (AI)

Кнопка **AI** с ползунком силы рядом находится на панели Audio & Buffer, сразу под SQ. На странице /mobile она во вкладке Audio, под Squelch. Она убирает шум диапазона из речи с помощью RNNoise — небольшой нейронной сети, обученной на голосе.

Всё происходит в вашем собственном браузере: ничего не отправляется на сторонний сервер, учётная запись не нужна, и приёмник не получает никакой дополнительной нагрузки.

- **Вкл/выкл:** нажмите **AI**; кнопка станет синей. При первом нажатии загружается модуль (около 1,3 МБ), и кнопка пульсирует, пока он загружается.
- **Сила:** задаёт, насколько сильно оно работает, пока кто-то говорит: меньшие значения сохраняют больше исходного звука. Верх шкалы намеренно мягче полного подавления, чтобы речь не портилась даже на 100 %, которые всё равно чистят сильнее, чем 50 %. Между словами оно само работает сильнее, поэтому шум там падает ещё больше. По умолчанию 50 %. Значение можно задать и при выключенном ИИ.

- **Индикатор состояния:** индикатор AI под частотой светится голубым, пока AI работает, и показывается приглушённым наполовину, когда AI включён, но текущий режим не голосовой.

Когда использовать: при приёме речи в SSB или AM на шумном диапазоне. Шипение между словами обычно падает примерно на 10 дБ при стандартных 50 % и сильнее при 100 %, примерно до 20 дБ, а уровень речи сохраняется.

Только голосовые режимы — USB, LSB, AM и SAM. В CW, FM, цифровых режимах и C-QUAM звук проходит без изменений, потому что сеть считает тон CW или музыку шумом. Как и четыре регулятора шума, оно никогда не попадает к декодерам. На слабых станциях (ниже примерно +6 дБ SNR) оно само измеряет отношение сигнал/шум и сильнее сглаживает свои решения, чтобы речь не приобретала булькающий, «водянистый» звук, который может давать такой фильтр; сильные станции обрабатываются точно так же, как раньше. Если очень слабый сигнал всё ещё звучит обработанным, уменьшите ползунок. Добавляет около 50 мс задержки звука.

Звуковой буфер и задержка

Звук приходит в браузер небольшими пакетами, по 35 мс на типичном приёмнике, и браузер держит небольшой запас, чтобы опоздавший пакет не вызывал пропуска. Этот запас — **буфер**, и на него приходится большая часть задержки между приёмником и вашим громкоговорителем.

- **Компьютер:** на панели Audio & Buffer кнопка со значком сети поднимает буфер на одну ступень за нажатие (она становится зелёной выше самой нижней ступени), а ползунок рядом с ней сразу выбирает ступень от 1 до 6. Нижняя ступень подходит для проводного подключения; телефоны начинают со ступени 2.
- **/mobile:** кнопка **BUF** и ползунок **Buffer** на вкладке Audio делают то же самое.

Буфер подстраивается сам. Страница измеряет, насколько неравномерно ваше соединение доставляет звук, и держит ровно такой запас, какой понадобился за последнюю минуту, плюс резерв безопасности выбранной ступени: 20 мс на ступени 1, 50 мс на ступени 2, больше на верхних. После кратковременного сбоя сети лишняя задержка снова уходит за минуту-две, а небольшое расхождение между тактовой частотой вашей звуковой карты и приёмника больше не приводит к обрыву каждые несколько минут. Подстройки идут по несколько миллисекунд, делаются там, где звук повторяется, и не слышны. Декодеры и ваши записи никогда не затрагиваются: они получают каждый отсчёт.

Если звук прерывается, поднимите буфер на одну ступень. На телефоне через домашний Wi-Fi задержка от приёмника до громкоговорителя обычно около 100–150 мс; к ней добавляется сеть между вами и приёмником.

Чат

Если оператор включил чат, у него есть своя панель на странице для компьютера и вкладка **Chat** на /mobile. Введите имя или позывной, напишите сообщение и отправьте его.

Чтобы ответить на конкретное сообщение, нажмите ↩ **Reply** рядом с его временем. Строка над полем ввода показывает, кому и на что вы отвечаете; пишите и отправляйте как обычно или нажмите X либо Esc для отмены. Ответ появляется с отступом под сообщением, на которое он отвечает, с короткой строкой цитаты. Ответ на ответ попадает в ту же ветку. Сервер хранит последние 20 сообщений; если исходное сообщение уже вышло из этой истории, ответ показывает только имя и время, на которые он отвечает.

Функция масштабирования

Увеличивает изображение водопада:

- **1x:** обычный вид (широкий охват)

- **2x**: увеличение ×2
- **4x**: увеличение ×4
- **8x**: увеличение ×8

Применяйте, когда: нужно лучше рассмотреть сигналы или точно настроиться.

Управление трансивером (CAT)

Ваш собственный трансивер и страница приёмника могут оставаться на одной частоте: вращайте ручку настройки трансивера — и водопад следует за ней, или щёлкните по сигналу на водопаде — и трансивер перестроится на него. Это касается только *вашего* сеанса прослушивания; остальные слушатели приёмника ничего не заметят.

C Desktop PhantomSDR+ (версии 4.0 и новее). В настольном приложении есть меню **Rig**. *Rig → Rig control...* открывает окно, в котором выбираются трансивер и способ подключения, а само меню включает и выключает синхронизацию. Синхронизируются частота, вид модуляции и ширина фильтра, в одну сторону или в обе, и приёмник можно заглушать на время передачи. Помимо PhantomSDR-Plus, так же управляются приёмники **KiwiSDR, PA3FWM WebSDR и UberSDR**. До трансивера приложение добирается одним из четырёх способов:

- **Встроенные драйверы** — без другого ПО: Icom (CI-V), Yaesu (новый CAT и FT-817/857/897), Kenwood, Elecraft, FlexRadio SmartSDR CAT, QRP Labs и другие Kenwood-совместимые трансиверы.
- **Hamlib** — любой трансивер, известный Hamlib, более 300 моделей, выбор из списка с поиском. Установщики для Windows содержат Hamlib; в Linux установите `libhamlib-utils`.
- **rigctld в сети** — уже запущенный `rigctld`.
- **flrig** — для трансивера, который flrig уже делит с fldigi, WSJT-X или аппаратным журналом.

Приложение, установщики и полное руководство — на странице [Desktop PhantomSDR+ \(Dropbox\)](#).

Через веб-браузер. [CATsync Tool for WebSDRs](#) (Windows) связывает трансивер со страницей приёмника, открытой в браузере. Синхронизирует частоту и вид модуляции.

Ширина фильтра и заглушение при передаче работают на приёмниках KiwiSDR, WebSDR и UberSDR, а также на приёмнике PhantomSDR-Plus версии 4.1.0 или новее. На более старом PhantomSDR-Plus частота и вид модуляции по-прежнему синхронизируются, фильтр — нет.

Полное руководство — все настройки, поддерживаемые трансиверы, как стороны не дают друг другу «воевать», и устранение неполадок — [Управление трансивером](#).

Для разработчиков. Каждая страница приёмника предоставляет на `window` следующие функции — именно их используют оба инструмента:

Функция	Назначение
<code>catsync_getFrequency()</code> / <code>catsync_setFrequency(hz)</code>	Частота настройки, в Гц
<code>catsync_getMode()</code> / <code>catsync_setMode(mode)</code>	USB, LSB, CW, CW-L, AM, FM, WBFM ...
<code>catsync_getBandwidth()</code> / <code>catsync_setBandwidth(hz)</code>	Полная ширина полосы пропускания, в Гц. Задавайте после вида модуляции: смена вида сбрасывает полосу пропускания
<code>catsync_getMute()</code> / <code>catsync_setMute(on)</code>	Заглушение, через собственную кнопку отключения звука страницы
<code>catsync_ready</code>	<code>true</code> , как только функции установлены

Последние три строки появились в версии 4.1.0; перед вызовом проверяйте, что функция существует.

Виды демодуляции

АМ (амплитудная модуляция)

Применяется для:

- авиационной связи
- АМ-радиовещания
- части любительской связи
- морской связи

Особенности:

- широкая полоса (обычно 10 кГц)
- восприимчива к помехам
- легко настраивается (достаточно щёлкнуть по сигналу)

Рекомендации:

- используйте быструю АРУ
- включите подавитель импульсных помех, если слышны щелчки
- настраивайтесь на пик сигнала на водопаде

QUAM (стереофоническое АМ C-QUAM)

Используется для:

- Средневолновых станций, вещающих в стереофоническом АМ

Характеристики:

- Полоса 10 кГц, декодируется как стереопара, а не как моно-АМ
- Надпись на кнопке сама становится **зелёной**, когда присутствует стереопилот 25 Гц, — так видно, какие станции действительно в стерео, ещё до выбора режима
- Из АМ повторное нажатие переключает на синхронное детектирование по несущей; надпись становится **жёлтой** и показывает SAM. Третье нажатие возвращает обычный АМ
- Звук C-QUAM передаётся через Opus; все прочие режимы используют FLAC

Рекомендации:

- Ночью ищите зелёную надпись на сильных средневолновых сигналах
- Если стерео звучит неустойчиво, при слабой несущей SAM спокойнее

FM (частотная модуляция)

Применяется для:

- любительских УКВ-ретрансляторов
- служб общественной безопасности (полиция, пожарные, скорая)
- коммерческой двусторонней связи
- части спутниковой связи

Особенности:

- узкая полоса (обычно 12,5 или 25 кГц)
- отличная помехоустойчивость
- «эффект захвата» (побеждает самый сильный сигнал)

Рекомендации:

- настраивайтесь точно на центр сигнала
- используйте шумоподаватель для отключения звука в паузах
- отключите шумоподавление (оно не нужно)

USB (верхняя боковая полоса)**Применяется для:**

- любительской КВ-связи (выше 10 МГц)
- большинства цифровых КВ-видов связи
- морской связи (выше 8 МГц)

Особенности:

- узкая полоса (обычно 2.4 кГц)
- эффективное использование спектра
- требует точной настройки

Рекомендации:

- используйте медленную АРУ
- настраивайтесь на нижний край сигнала на водопаде
- при необходимости включите шумоподавление

LSB (нижняя боковая полоса)**Применяется для:**

- любительской КВ-связи (ниже 10 МГц)
- некоторых цифровых КВ-видов связи
- морской связи (ниже 8 МГц)

Особенности:

- то же, что USB, но зеркально
- по традиции LSB используется на низкочастотных КВ-диапазонах

Рекомендации:

- используйте медленную АРУ
- настраивайтесь на верхний край сигнала на водопаде
- при необходимости включите шумоподавление

CW (незатухающие колебания / азбука Морзе)**Применяется для:**

- любительской телеграфии
- навигационных маяков
- станций сигналов точного времени

Особенности:

- очень узкая полоса (100-500 Гц)
- высокая эффективность
- для понимания нужно знать азбуку Морзе

Рекомендации:

- используйте узкий фильтр (400-500 Гц)
- настраивайтесь точно на центр тона
- включите звуковой фильтр для более чистого тона

CW-L (CW, нижняя боковая полоса)

Тот же морзе-фильтр ± 250 Гц, что и у **CW**, но тон берётся снизу от несущей, а не сверху. Пригодится, когда сигнал лучше принимается с нижней стороны или когда мешающая несущая стоит прямо над нужной.

На настольной странице кнопки CW-L нет. Её ряд режимов — USB · LSB · CW · AM · QUAM · FM. CW-L предлагается на упрощённой странице /mobile, а на настольной его по-прежнему можно задать ссылкой или закладкой, которая его называет.

WBFM (широкополосная ЧМ)**Применяется для:**

- FM-радиовещания (88-108 МГц)
- некоторых спутниковых нисходящих каналов

Особенности:

- очень широкая полоса (200 кГц)
- отличное качество звука
- высокая верность воспроизведения

Рекомендации:

- настраивайтесь точно на центральную частоту
- для вещания шумоподаватель не нужен
- наслаждайтесь высоким качеством звука!

Цифровые декодеры

PhantomSDR-Plus включает встроенные декодеры цифровых видов связи. Полное руководство см. в [Декодеры](#).

Быстрее всего — ряд кнопок **Decoders** на главной панели, сразу под **Wheel Tuning Steps**: десять кнопок — **FT8**, **FT4**, **FT2**, **JS8**, **CW**, **WSPR**, **FAX**, **SSTV**, **NAVTX**, **RTTY** — каждая из них одним нажатием запускает свой декодер, переводит главный выключатель Decoder в ON и прокручивает окно декодера в поле зрения. Кнопка остаётся синей, пока декодер работает; нажмите её ещё раз, чтобы остановить декодер и закрыть его окно. Выпадающий список **Decoder Options** работает как прежде и остаётся синхронным с кнопками.

RADEL и **RADEU** (цифровой голос RADE v1) имеют собственные кнопки рядом с заголовком **Modes selector** и внутри всплывающих окон **Modes** и **Bands** и работают так же: нажатие — старт, повторное нажатие — стоп.

Декодеры SSTV, HF FAX, NAVTEX, FSK/RTTY и CW работают каждый в своём фоновом потоке, поэтому включение или выключение декодера никогда не прерывает звук, а водопад остаётся плавным во время декодирования. Они получают звук, взятый до АРУ, шумоподавления и отключения звука — вы можете выключить звук приёмника, и декодирование продолжится как ни в чём не бывало.

Пока декодер работает, он удерживает режим и полосу пропускания. Обычно режим следует за планом диапазонов: настройтесь на участок, помеченный LSB или AM, и приёмник переключится. Работающий декодер это перекрывает и сохраняет нужный ему режим и узкую полосу, даже если вы уйдёте на другой диапазон. Собственный режим диапазона возвращается, как только вы выключите декодер, а кнопки режимов работают в любой момент, если вы хотите взять управление на себя. Декодер CW — исключение: он декодирует в том режиме, в котором вы слушаете.

Декодер FT8, FT4

Что такое FT8 и FT4?

- популярные цифровые виды любительской связи
- связь слабыми сигналами
- передачи по 15 секунд для FT8 и по 7,5 секунды для FT4

Как пользоваться:

1. Настройтесь на частоты FT8 или FT4
2. Выберите режим USB
3. Включите декодер FT8 — нажмите кнопку **FT8** или выберите его в меню
4. Наблюдайте за появлением декодированных сообщений

Основные частоты FT8 (USB):

- 160 метров: 1.840 МГц
- 80 метров: 3.573 МГц
- 60 метров: 5.357 МГц
- 40 метров: 7.074 МГц
- 30 метров: 10.136 МГц
- 20 метров: 14.074 МГц
- 17 метров: 18.100 МГц
- 15 метров: 21.074 МГц
- 12 метров: 24.915 МГц
- 10 метров: 28.074 МГц
- 6 метров: 50.313 МГц (50.323 МГц для DX)

Основные частоты FT4 (USB):

- 80 м: 3.575 МГц
- 40 м: 7.0475 МГц
- 30 м: 10.140 МГц
- 20 м: 14.080 МГц
- 17 м: 18.104 МГц
- 15 м: 21.140 МГц
- 12 м: 24.919 МГц
- 10 м: 28.180 МГц

- 6 м: 50.318 МГц

Декодер JS8

Что такое JS8?

- Механизм слабого сигнала FT8, применённый для разговора с клавиатуры на клавиатуру
- Свободный текст вместо фиксированных посылок: длинные сообщения приходят за несколько циклов
- Пять скоростей; **Normal** (15 с) — вызывная, на неё приходится почти весь трафик

Как пользоваться:

1. Настройтесь на частоту JS8
2. Выберите режим **USB**
3. Включите декодер JS8 — кнопка **JS8** или через меню
4. Оставьте **Speed** на *Normal*, а sync offset на **Auto**

Как читать панель:

- Ещё принимаемые сообщения показываются сверху зелёным с мигающим курсором — на Normal сообщение может занять целую минуту
- Завершённые перечисляются ниже как Mode | Hz | dB | Message, **позывные зелёным**
- Приглушённая курсивная строка истекла до прихода последнего кадра: текст настоящий, но может быть обрезан

Обычные частоты JS8 (USB):

- 160m: 1.842 MHz
- 80m: 3.578 MHz
- 40m: 7.078 MHz
- 30m: 10.130 MHz
- 20m: 14.078 MHz
- 17m: 18.104 MHz
- 15m: 21.078 MHz
- 12m: 24.922 MHz
- 10m: 28.078 MHz

JS8 намного тише FT8 — одна передача в несколько минут это нормально, а паузы не являются неисправностью. Полное руководство: [Декодеры](#).

Декодер CW

Просто нажмите кнопку CW, и появится декодированный текст. Нажмите кнопку CW ещё раз, чтобы очистить окно и перезапустить декодер.

WSPR

WSPR (Weak Signal Propagation Reporter, произносится «уиспер») — маячковый режим сверхслабых сигналов, который картирует пути КВ-распространения по всему миру. Каждая передача длится около 110 секунд и уместается в полосе шириной 200 Гц. Декодер ждёт полного двухминутного интервала, выровненного по UTC, и только затем декодирует. Включите декодер и выберите **WSPR** в выпадающем списке. Или просто нажмите кнопку **WSPR**.

HF FAX / WEFAX

KB-радиофакс (также известный как WEFAX) используется береговой охраной и метеослужбами по всему миру для передачи карт погоды, карт состояния моря и приземного анализа на коротких волнах. Декодер восстанавливает изображение строка за строкой по мере приёма.

NAVTEX

NAVTEX — международная морская система передачи информации по безопасности в прибрежных водах: навигационные предупреждения, прогнозы погоды и сообщения о поисково-спасательных операциях.

FSK / RTTY, PSK31 и Olivia

Универсальный декодер узкополосных текстовых видов связи с пятью вариантами в одном окне: морской FSK (SITOR), метеорологический RTTY, любительский RTTY, **PSK31** (фазовая манипуляция, 31,25 бод) и **Olivia** (многотональная FSK с прямой коррекцией ошибок). Панель подстраивается под вариант — органы управления сдвигом, скоростью и кадром исчезают для PSK31 и Olivia, а Olivia добавляет переключатель Mode и ползунок шумоподавителя.

У любительского и метео-RTTY есть ползунок **Squelch (SNR)**, который не даёт шуму печататься случайными буквами, а все варианты, кроме Olivia, показывают настоящий SNR по одной шкале в 3 кГц. Два важных момента: PSK31 сам исправляет ошибку настройки в пределах примерно ± 25 Гц, поэтому достаточно подойти близко; Olivia требует, чтобы **Mode** (тоны / полоса) точно совпадал с передачей, не передаёт преамбулу и потому синхронизируется несколько секунд, прежде чем появится текст. Панель открывается на Olivia **8 / 250**.

QRSS-граббер

QRSS — это CW, передаваемая так медленно, что одна точка длится секунды; её смотрят, а не слушают — граббер рисует след на собственном дисплее. Его нет в списке декодеров: у него свой раздел **QRSS**, и он может работать одновременно с декодером.

Нажмите **Show**, затем выберите окно в списке **Band** и нажмите **Tune**: приёмник переходит на эту частоту в **CW** с полосой, скроенной под этот режим, а след попадает на центральную линию дисплея. Самое оживлённое окно — 30 м (10 140,00 кГц). Установите **Speed** по маяку — **QRSS 10**, если не знаете, — и запаситесь терпением: позывной может пересекать экран десять минут. Подробности в [Декодеры](#).

SSTV

Медленное телевидение (SSTV) передаёт неподвижные изображения по обычному каналу SSB, строка за строкой. Включите декодер, выберите **SSTV** и настройтесь на частоту SSTV — 14.230 МГц является основной международной частотой вызова. Оставьте **Mode** в положении **Auto**: декодер читает заголовок VIS передачи, а если вы настроились уже после заголовка, определяет режим по синхросигналам. Поддерживаются режимы Martin, Scottie и Robot, и изображение строится строка за строкой по мере поступления. Или просто нажмите кнопку **SSTV**.

Автоматическая отправка спотов и графики сервера

Декодирования FT8, FT4 и WSPR может выгружать и сам сервер — FT8/FT4 в PSK Reporter, WSPR в WSPRnet — через службу autorun, которую оператор запускает из панели администратора. Она не

зависит от декодеров в вашем браузере: работает независимо от того, слушает кто-нибудь или нет, и ничего из декодированного вами в браузере не отправляется.

Оператор следит за ней по двум счётчикам, которые легко перепутать: плитки по декодерам считают споты, отправленные с последнего запуска службы, а число рядом с каждым флажком диапазона/вида связи — итог этой позиции за всё время, переживающий перезапуски. В той же панели есть страница **Графики** с частотой процессора, нагрузкой, температурой и числом слушателей за период от 15 минут до 24 часов. Оба описаны в [руководстве по панели администратора](#).

Горячие клавиши

Горячие клавиши для более быстрой работы.

Управление частотой

Индикатор частоты — это ряд цифр, которыми вы управляете напрямую. **Сначала щёлкните по цифре** — это её выбирает, и всё нижеследующее действует на выбранную.

- **Стрелка влево / вправо:** переносит выбор на соседнюю цифру (восемь цифр, от 100 МГц до 10 Гц)
- **Стрелка вверх / вниз:** увеличивает или уменьшает *выбранную* цифру на её разрядное значение — 1 МГц на разряде мегагерц, 10 Гц на последнем
- **0 – 9:** вводит эту цифру прямо в выбранную позицию

Колесо мыши над индикатором частоты: шагает с шагом настройки диапазона (1 кГц, если план диапазонов не задаёт другой). Удерживайте **Shift** для 1 кГц, **Alt** для 10 кГц.

Колесо мыши над водопадом: масштабирует. Удерживайте **Ctrl** (или **Cmd**) либо **Shift**, чтобы вместо масштабирования настраиваться, а **Shift + Ctrl** вместе — чтобы встать на ближайший целый кГц.

Сочетаний Page Up / Page Down нет.

Закладки

Сохраняйте любимые частоты для быстрого доступа. Список закладок можно также экспортировать и сохранить локально, а затем импортировать в любой другой PhantomSDR.

Когда закладки и маркеры перекрываются:

Синие закладки отображаются сверху

Жёлтые маркеры — под ними

Щелчки по закладкам имеют приоритет

Добавление закладки

1. Настройтесь на нужную частоту
2. Нажмите кнопку «Bookmarks»
3. Нажмите «Add Bookmark»
4. Введите описание
5. Нажмите «Save»

			Bookmark name Label (optional) [Add]
--	--	--	--

Как это работает:

Добавление закладки:

- Название: «Local News Station»
- Метка: «NEWS»
- Нажмите Add

Отображение на водопаде:

- увеличивайте масштаб, пока не появятся маркеры
- вы увидите тёмно-синий прямоугольник с надписью «NEWS» жирным жёлтым

Щелчок по закладке:

- настраивает на частоту
- устанавливает вид демодуляции
- работает точно так же, как щелчок по маркеру

Управление закладками

- **Изменить:** щёлкните значок карандаша рядом с закладкой
- **Удалить:** щёлкните значок корзины рядом с закладкой
- **Экспорт:** скачайте закладки в виде файла JSON
- **Импорт:** загрузите закладки из файла JSON

Обмен закладками

1. Нажмите «Export Bookmarks»
2. Передайте файл JSON другим
3. Получатели нажимают «Import Bookmarks»
4. Выбирают ваш файл

Использование на мобильных устройствах

PhantomSDR-Plus отлично работает на мобильных устройствах!

Два мобильных вида

Пользоваться приёмником с телефона можно двумя способами:

- **http://ваш_сервер:ПОРТ/mobile** — упрощённая страница. Водопада нет, поэтому расход трафика примерно вдвое меньше. Ввод частоты, шаги настройки, режимы, S-метр, диапазоны, закладки, список слушателей и чат.
- **Расширенный вид** — телефонная раскладка основного интерфейса, с водопадом и полным набором органов управления.

Переключение — кнопками внизу `/mobile` (**Mobile extended view, Full desktop view**) и кнопкой **Simplified mobile** в расширенном виде.

Ваша частота следует за вами. Смена вида оставляет вас на том же сигнале: частота и режим передаются в ссылке, поэтому вы больше не попадаете на частоту приёмника по умолчанию.

Адресная строка тоже следует за настройкой, так что перезагрузка страницы, закладка или отправленная кому-то ссылка возвращают ровно на эту частоту.

Режим следует плану диапазонов на обеих страницах. Настройтесь на участок, указанный как AM, LSB, USB или CW — вводом частоты, шагами или кнопкой диапазона — и приёмник переключится на этот режим, как на упрощённой странице, так и в полном интерфейсе. Режим, выбранный вручную, сохраняется, пока вы двигаетесь внутри того же участка, а вне заданных диапазонов ваш режим не трогают. Пока работает **RADE** (RADEL/RADEU), он удерживает приёмник, поэтому настройка его не прерывает.

При переключении между видами текущий режим идёт с вами; если в том виде, куда вы переходите, соответствия нет, режим для этой частоты выбирает план диапазонов — вещательная частота приходит в AM, 40 м в LSB, участок CW в CW. SAM на упрощённой странице превращается в AM с синхронным детектором в расширенном виде и обратно. RADEL и RADEU есть только на упрощённой странице: уходя с одного из них, вы сохраняете частоту, а режим выбирает план диапазонов.

Частота за пределами покрытия приёмника подтягивается к ближайшему краю, так что старая ссылка никогда не оставит вас вне диапазона.

Особенности мобильной версии

- **Удобное сенсорное управление:** крупные кнопки и ползунки
- **Свайп для настройки:** проведите влево/вправо по водопаду
- **Щипок для масштаба:** сведите или разведите пальцы на водопаде
- **Альбомный режим:** поверните устройство для лучшего обзора

Советы для мобильных устройств

1. **Используйте Wi-Fi:** потоковый звук расходует трафик
2. **Альбомная ориентация:** лучше видно водопад
3. **Наушники:** лучшее качество звука
4. **Сохраняйте закладки:** проще возвращаться к станциям
5. **Закройте другие приложения:** обеспечит плавную работу

Рекомендуемые мобильные браузеры

- **Android:** Chrome или Samsung Internet
- **iOS:** Mozilla
- **Оба:** следите за актуальностью версии браузера

Советы и рекомендации

Для лучшего приёма

1. **Выбирайте сильные сигналы:** ищите оранжевый/красный на водопаде
2. **Настраивайтесь точно:** щёлкайте по центру сигнала
3. **Выбирайте правильный вид модуляции:** он должен соответствовать типу сигнала
4. **Настройте АРУ:** быстрая для AM, медленная для SSB
5. **Используйте NR/NB:** помогают в условиях помех

Как найти активность

1. **Следите за водопадом:** цвета показывают уровень сигнала
2. **Слушайте популярные частоты:**
3. 40 м: 7.100-7.300 МГц (LSB)
4. 20 м: 14.200-14.350 МГц (USB)
5. 2 м: 145.200-145.600 МГц (FM)
6. **Смотрите наложение band plan:** оно показывает распределение частот
7. **Используйте закладки:** быстрый доступ к активным частотам

Понимание условий прохождения

КВ днём:

- лучше работают высокочастотные диапазоны (20 м, 15 м, 10 м)
- возможна дальняя связь (DX)
- слышны вещательные станции

КВ ночью:

- лучше работают низкочастотные диапазоны (80 м, 40 м)
- другие механизмы распространения
- слышны другие станции

УКВ:

- в основном прямая видимость
- местная связь
- более стабильные условия

Этикет

1. **Не занимайте приёмник надолго:** другие тоже хотят послушать
2. **Уважительно пользуйтесь чатом:** будьте вежливы с другими пользователями
3. **Сообщайте о проблемах:** помогайте оператору поддерживать станцию
4. **Не просите техподдержку:** это площадка для прослушивания. За помощью напишите сисопу.

Устранение неполадок

Нет звука

Возможные причины:

1. Браузер отключил звук → проверьте регуляторы громкости браузера
2. Звук отключён в системе → проверьте громкость компьютера
3. Слабый сигнал → настройтесь на более сильный сигнал (S7+)
4. Неверный вид модуляции → попробуйте другие виды демодуляции

Решения:

1. Щёлкните по сильному сигналу (оранжевый/красный на водопаде)
2. Убедитесь, что звук браузера не отключён (значок отключения на вкладке)

3. Попробуйте другую частоту
4. Перезагрузите страницу (F5)

Искажённый звук

Возможные причины:

1. Перегрузка → сигнал слишком сильный
2. Неверный вид модуляции → сигнал AM слышится в SSB и т. п.
3. Помехи → влияние соседних сигналов

Решения:

1. Уменьшите громкость
2. Попробуйте другой вид демодуляции
3. Используйте более узкую полосу фильтра
4. Отстройтесь от мешающих сигналов

Звук прерывается или отстаёт

Возможные причины:

1. Неравномерное соединение → пакеты приходят с опозданием, и буфер пустеет
2. Загруженный телефон или компьютер → браузер не успевает

Решения:

1. Поднимите буфер на одну ступень (см. *Звуковой буфер и задержка*)
2. Предпочитайте Wi-Fi или кабель мобильному интернету
3. Закройте другие вкладки и приложения

Чтобы разобраться подробнее, откройте страницу, добавив к адресу `?diag=1` (например, `http://your_server:PORT/mobile?diag=1`). Тогда страница `/mobile` покажет под своими элементами управления задержку, неравномерность соединения и состояние буфера; страница для компьютера каждые десять секунд пишет ту же строку в консоль браузера (F12). Полезно, когда сообщаете о проблеме оператору станции.

Водопад не обновляется

Возможные причины:

1. Проблема с сетью → медленное или прерывающееся соединение
2. Производительность браузера → открыто слишком много вкладок
3. Перегрузка сервера → слишком много пользователей

Решения:

1. Проверьте интернет-соединение
2. Закройте ненужные вкладки браузера
3. Перезагрузите страницу (F5)
4. Попробуйте позже, когда пользователей будет меньше

Не удаётся настроиться на частоту

Возможные причины:

1. Частота вне диапазона → SDR не покрывает эту частоту

2. Неверный формат ввода → используйте правильный формат (например, «14200», а не «14.200.000»)

Решения:

1. Проверьте частотный охват SDR (указан на странице)
2. Используйте приведённые примеры формата частоты
3. Вместо ввода щёлкните по водопаду

Звук прерывается

Возможные причины:

1. медленное интернет-соединение
2. высокая нагрузка на сервер
3. проблемы с производительностью браузера

Решения:

1. Закройте другие приложения, использующие канал
2. Попробуйте в часы меньшей нагрузки
3. Закройте ненужные вкладки браузера
4. Используйте проводное соединение вместо Wi-Fi

Часто задаваемые вопросы

Общие вопросы

В: Нужно ли специальное оборудование для работы с WebSDR? О: Нет! Только компьютер или мобильное устройство с доступом в Интернет.

В: Пользование WebSDR бесплатно? О: Да, большинство WebSDR бесплатны. Их обслуживают энтузиасты.

В: Можно ли передавать через WebSDR? О: Нет, WebSDR работает только на приём. Передавать нельзя.

В: Какие частоты можно слушать? О: Зависит от конфигурации WebSDR. Смотрите сведения о станции.

В: Можно ли записывать звук? О: Некоторые браузеры позволяют запись. Проверьте возможности вашего браузера.

Технические вопросы

В: Какую частоту дискретизации использует SDR? О: Зависит от станции. Смотрите страницу сведений о станции.

В: Какова задержка? О: Обычно несколько десятых долей секунды между радиосигналом и вашими динамиками: обработка в приёмнике, сеть и буфер воспроизведения браузера, который подстраивается под ваше соединение (около 100–150 мс на телефоне через домашний Wi-Fi). См. *Звуковой буфер и задержка*.

В: Можно ли открыть несколько экземпляров? О: Обычно да, но это может нагрузить сервер. Будьте внимательны к другим.

В: Работает ли он офлайн? О: Нет, WebSDR требует подключения к Интернету.

В: Какие браузеры поддерживаются? О: Chrome, Firefox, Edge, Safari (все актуальные версии)

Вопросы по использованию

В: Сколько человек могут слушать одновременно? О: Зависит от возможностей сервера. Часто 50-200 и более пользователей.

В: Можно ли увидеть, что слушают другие? О: Если функция включена — да. Ищите индикаторы «другие пользователи».

В: Можно ли общаться с другими слушателями? О: Если его включил оператор. Ищите окно чата; ↪ **Reply** отвечает на конкретное сообщение (см. *Чат* выше).

В: Почему на некоторых частотах ничего нет? О: Сейчас на этой частоте нет сигналов. Попробуйте другие!

В: Что за цветные полосы на водопаде? О: Наложение band plan, показывающее распределение частот.

Ресурсы

Больше узнать о радио

- **Band plan:** поищите «amateur radio band plan» + ваш регион
- **Распространение радиоволн:** изучите распространение КВ-радиоволн
- **Цифровые виды связи:** почитайте про FT8, PSK31, RTTY
- **Радиолюбительство:** подумайте о получении любительской лицензии!

Найти другие WebSDR

- **Каталог WebSDR:** <http://sdr-list.xyz>
- **WebSDR.org:** <http://websdr.org>
- **KiwiSDR:** <http://kiwisdr.com/public/>

Получить помощь

1. **Оператор станции:** контактные данные указаны на странице
 2. **Чат пользователей:** спросите других слушателей (если доступен)
 3. **Онлайн-форумы:** поищите сообщества WebSDR
 4. **Документация:** обратитесь к этому руководству!
-

Приложение: часто используемые частоты

Любительские КВ-диапазоны

Диапазон	Полоса частот	Вид модуляции	Активность
160 м	1.800-2.000 МГц	LSB	Ночь/местная
80 м	3.500-4.000 МГц	LSB	Ночь/региональная
40 м	7.000-7.300 МГц	LSB	День/ночь/DX
30 м	10.100-10.150 МГц	USB	Только данные/CW
20 м	14.000-14.350 МГц	USB	День/DX
17 м	18.068-18.168 МГц	USB	День/DX
15 м	21.000-21.450 МГц	USB	День/DX
12 м	24.890-24.990 МГц	USB	День/DX
10 м	28.000-29.700 МГц	USB	Спорадическая/DX

Любительские УКВ-диапазоны

Диапазон	Полоса частот	Вид модуляции	Активность
6 м	50.000-54.000 МГц	USB/FM	Спорадическая
2 м	144.000-148.000 МГц	FM	Очень высокая
70 см	420.000-450.000 МГц	FM	Высокая

Вещательные диапазоны

Служба	Полоса частот	Вид модуляции
АМ-радио	530-1710 кГц	АМ
Короткие волны	2.3-26.1 МГц	АМ
FM-радио	88-108 МГц	WBFM

Авиация

Служба	Полоса частот	Вид модуляции
Управление воздушным движением	118-137 МГц	АМ
ACARS (данные)	130-136 МГц	Данные

Морская связь

Служба	Полоса частот	Вид модуляции
Морская УКВ	156-162 МГц	FM
Морская КВ	2-22 МГц	USB

Глоссарий

AGC: автоматическая регулировка усиления — автоматически подстраивает уровень звука

AM: амплитудная модуляция — голосовой вид связи в авиации и вещании

Полоса пропускания: диапазон частот, занимаемый сигналом

CW: незатухающие колебания — сигналы азбуки Морзе

DX: дальняя связь

FFT: быстрое преобразование Фурье — переводит временную область в частотную

FM: частотная модуляция — голосовой вид связи для УКВ

HF: высокие частоты (3-30 МГц) — диапазоны дальней связи

кГц: килогерц (1 000 Гц)

LSB: нижняя боковая полоса — голосовой вид связи на низкочастотных КВ-диапазонах

МГц: мегагерц (1 000 000 Гц)

NB: подавитель импульсных помех — удаляет импульсный шум

NR: шумоподавление — уменьшает фоновый шум

PSK: фазовая манипуляция — цифровой вид связи

RTTY: радиотелетайп — цифровой текстовый вид связи

S-метр: индикатор уровня сигнала

SDR: программно-определяемое радио

SQL: шумоподаватель — отключает звук при отсутствии сигнала

SSB: однополосная модуляция (USB или LSB)

USB: верхняя боковая полоса — голосовой вид связи на высокочастотных КВ-диапазонах

VHF: очень высокие частоты (30-300 МГц) — прямая видимость

UHF: ультравысокие частоты (300-3000 МГц) — прямая видимость

Водопад: визуальное отображение радиоспектра во времени

Приятного исследования радиоспектра с PhantomSDR-Plus!

73 (наилучшие пожелания) de SV1BTL & SV2AMK

Инструкции по установке см. в [INSTALLATION.md](#). Технические подробности см. в [PROJECT_STRUCTURE.md](#).

Эмуляция клиента KiwiSDR

Aether_config.md

Эмуляция клиентов KiwiSDR

Подключение AetherSDR, kiwiclient и другого ПО для KiwiSDR к PhantomSDR-Plus

Начиная с версии 4.1.0 PhantomSDR-Plus умеет отвечать и на **протокол KiwiSDR**, так что программы, написанные для KiwiSDR — **AetherSDR**, **kiwiclient** и прочие — подключаются к вашему приёмнику напрямую. Это мост внутри того же процесса сервера, на том же хосте и том же порту, который вы уже публикуете: без второй службы, без второго порта, без прокси.

Он выключен, пока вы его не включите. Ничего из моста не работает — ни сокет, ни проверка — пока `[kiwi_emulation]` отсутствует или равен `false`, а это состояние свежего дерева.

Содержание

1. Что получает клиент Kiwi
2. Установка моста
3. Включение
4. Справочник настроек
5. Подключение клиента
6. Уровень звука
7. S-метр
8. Водопад и спектр
9. Если что-то выглядит не так
10. Насколько это проверено

1. Что получает клиент Kiwi

Звук	<code>ws://<host>:<port>/kiwi/<id>/SND</code> — демодулированный звук в собственном кадровании SND у Kiwi
Водопад	<code>ws://<host>:<port>/kiwi/<id>/W/F</code> — спектр, в кадрах W/F Kiwi
Перестройка	настоящая, а не косметическая: <code>SET mod=...</code> от клиента меняет частоту, боковую полосу и полосу пропускания на стороне PhantomSDR, ровно так же, как это сделал бы слушатель в браузере
S-метр	питается побочной мощностью самого демодулятора, по той же шкале, что и веб-S-метр

Перестройка поддерживается для **вещественного** входа (`signal = "real"` в вашем `.toml`). Для входа IQ соотношение между бином и частотой иное и не реализовано, поэтому приёмник с IQ будет отдавать звук и водопад, но не станет следовать за перестройкой клиента Kiwi.

2. Установка моста

Мост — это набор правок к исходникам серверной части плюс один новый заголовочный файл, `src/kiwi_bridge.h`. Внести их можно двумя способами.

Установщиком. У `install.sh` (а также `install_arch.sh`, `install_fedora.sh`, `install_opensuse.sh`) есть отдельный шаг для этого — шаг 17, *Kiwi client emulation*. Он предлагается, а не навязывается: ответьте `n`, и ничего не будет изменено. Для автоматического прогона решает `PHANTOM_KIWI=y|n`, по умолчанию — да.

На уже установленном дереве запустите скрипт, который поставляется в корне репозитория:

```
cd ~/PhantomSDR-Plus
./kiwi_install.sh
./recompile.sh          # выберите [1] Только серверную часть
```

`kiwi_install.sh` сначала сохраняет копию каждого файла, к которому прикасается, в `backup_kiwi_bridge_<метка-времени>/` и применяет каждую правку по точному совпадению текста — если якорь оказался не там, где он его ждёт (локально изменённое дерево, другая версия), он немедленно останавливается, называет файл и оставляет его нетронутым. Скрипт идемпотентен: на дереве, где мост уже есть, он сообщает о каждой правке как об уже применённой и ничего не меняет.

Если вы только что применили архив обновления, в котором мост уже есть, запускать его не нужно. Исправленные исходники лежат в пакете; `kiwi_install.sh` лишь сообщит, что все правки уже на месте.

Он правит `src/client.h`, `src/signal.cpp`, `src/waterfall.cpp`, `src/spectrumserver.h`, `src/spectrumserver.cpp`, `src/websocket.cpp` и `src/http.cpp`, а `kiwi_bridge.h` копирует в `src/`.

Он устанавливает мост, а не обновляет его. Если правка когда-нибудь *не пройдёт* на дереве, где мост уже работает, значит исходный код ушёл вперёд скрипта — и обновлять нужно скрипт, а не ваше дерево.

3. Включение

Добавьте это в тот `.toml`, с которым ваш приёмник действительно запускается, — тот самый файл, который вы передаёте `spectrumserver`, названный в вашем `start-<radio>.sh`. Не `config.example.*`:

```
[kiwi_emulation]
enabled = true
```

Затем перезапустите приёмник:

```
./stop-websdr.sh
./start-<ваш-приёмник>.sh
```

`kiwi_install.sh` добавит блок за вас, с `enabled = true`, в те из `config.toml`, `config-rx888mk2.toml`, `config-airspyhf.toml`, `config-rtl.toml`, `config-rspla.toml`, `config-fobos.toml`, `config-fobos-hf.toml` и `config-hackrf.toml`, которые есть в корне репозитория. О конфигурации, которую вы храните в другом месте или под другим именем, он знать не может, а **архивы обновлений никогда не**

поставляют `.toml`, так что после обновления блок вам придётся добавить вручную. В поставляемых `config.example.*.toml` он присутствует с пояснениями и со значением `false`.

4. Справочник настроек

Всё, что читает мост, находится в одном разделе `[kiwi_emulation]`. Обязателен только `enabled`; у всех четырёх подстроек значение по умолчанию — откалиброванное, поэтому блок, в котором нет ничего, кроме `enabled = true`, уже правилен.

Ключ	По умолчанию	Что делает
<code>enabled</code>	<code>false</code>	Отвечать на протокол KiwiSDR вообще. Выключено = полностью бездействует.
<code>audio_gain</code>	<code>0</code>	дБ. Выходное усиление, только для клиентов Kiwi. См. Уровень звука .
<code>smeter_offset</code>	<code>input.analog_smeter_offset</code>	дБ. Переопределяет смещение S-метра, только для клиентов Kiwi. См. S-метр .
<code>wf_cal</code>	<code>0</code>	дБ. Подстройка отображения водопада и спектра. См. Водопад и спектр .
<code>wf_fps_max</code>	<code>23</code>	Кадров водопада в секунду. См. Водопад и спектр .

Полностью заполненный блок со значениями, с которыми работает приёмник этого проекта:

```
[kiwi_emulation]
enabled      = true
audio_gain   = 60.0    # дБ, только клиенты Kiwi. 0 = без изменений.
wf_cal       = -10.0   # дБ, дело вкуса. 0 = совпадает с S-метром.
wf_fps_max   = 28      # кадров/с. 23 — максимум протокола.
# smeter_offset = 5.0  # дБ. По умолчанию input.analog_smeter_offset.
```

Все эти ключи читаются при запуске. Изменение любого из них — это **перезапуск**, а не пересборка: `./stop-websdr.sh`, затем ваш скрипт запуска. Чтобы их подстроить, пересобирать не нужно никогда.

Принимаются и целая, и десятичная запись: `28` и `28.0` читаются одинаково. Только по-настоящему отсутствующий ключ возвращается к своему значению по умолчанию.

5. Подключение клиента

Направьте клиент на тот же адрес и порт, которыми пользуются ваши слушатели, — без пути, без префикса. AetherSDR спрашивает хост и порт; `kiwiclient` принимает `-s` и `-p`:

```
# kiwiclient, запись 30 секунд на 7100 кГц LSB
python3 kiwirecorder.py -s ваш.приёмник.example -p 8073 -f 7100 -m lsb --tlimit=30
```

Клиент настраивается, водопад заполняется, S-метр показывает. Если не происходит совсем ничего, мост почти наверняка всё ещё выключен — см. [раздел 3](#).

6. Уровень звука

Первое, что замечает большинство: клиент Kiwi звучит тоньше, чем веб-страница PhantomSDR. Дело не в том, что мост что-то теряет: один буфер питает все кодеки, и одна и та же частота с одной и той же полосой, измеренная через собственный звуковой тракт браузера и через /SND, даёт одинаковые отсчёты. Веб-страница звучит громко потому, что `audio.js` пересобирает звук в браузере — подъём низов, полосовой фильтр, подъём присутствия, компрессор с компенсирующим усилением и регулятор громкости. У клиента Kiwi ничего этого нет и быть не может.

`audio_gain` закрывает этот разрыв, только для клиентов Kiwi:

```
[kiwi_emulation]
enabled      = true
audio_gain = 55.0      # дБ, только клиенты Kiwi. 0 = без изменений.
```

Начинайте примерно с 55-60 дБ. 55 дБ — не результат вычисления: это значение, при котором клиент Kiwi и веб-страница PhantomSDR были измерены на одном уровне, на сильном местном сигнале на 729 КГц. На практике часто предпочитают чуть больше — этот приёмник остановился на 60 — так что выбирайте на слух.

Испортить звук этим нельзя. **Ограничитель пиков с упреждением** стоит между усилением и 16-битным ограничением: каждый отсчёт задерживается на 4 мс, пока усиление рассчитывается по звуку, который ещё не отправлен, так что громкий фрагмент встречает усиление, уже опустившееся ради него, и пики сворачиваются, а не срезаются. При 60 и даже 70 дБ — далеко за пределами запаса, который есть у одного усиления — не был срезан ни один отсчёт из четверти миллиона. Ниже порога ограничителя усиление в точности равно 1,0, так что приёмник, не гоняющий звук на пределе, получает отсчёты нетронутыми.

Слишком большое значение стоит громкости, а не качества. Выше примерно 60 дБ лишнее усиление попросту срезается ограничителем: по измерениям здесь уровень вырос примерно на 1 дБ при переходе с 55 на 60 и почти не изменился с 60 на 70. Дальше вы покупаете компрессию, а не громкость.

Ничто из этого не затрагивает веб-страницу и не сдвигает S-метр — он берётся из мощности демодулятора, а не из звуковых отсчётов, так что прибавив громкость, заставить прибор врать невозможно.

7. S-метр

S-метр Kiwi не калибруется отдельно: он воспроизводит то, что показывает ваша веб-страница, ступень за ступенью. Поблочная мощность демодулятора — отправная точка для обоих, а затем страница применяет `input.analog_smeter_offset`, растягивает результат относительно -130 дБм и добавляет собственное смещение отображения. Мост делает то же самое, поэтому клиент Kiwi и страница показывают один и тот же сигнал с одинаковой силой — проверено на сильной местной станции, где оба показали -37 дБм.

Если — и только если — вы хотите, чтобы клиенты Kiwi показывали иначе, чем ваша страница, задайте мосту собственное смещение:

```
[kiwi_emulation]
enabled      = true
smeter_offset = 5.0      # заменяет input.analog_smeter_offset только для клиентов Kiwi
```

Сравнивайте два прибора при одинаковом фильтре — или не сравнивайте вовсе.

Показание берётся из мощности в полосе пропускания, поэтому более широкий фильтр собирает больше шума и показывает выше: на тихом диапазоне одна и та же частота измерялась на 5,6 дБ сильнее при 9 кГц, чем при 2,4 кГц. Клиент Kiwi с фильтром 6 кГц против веб-страницы на 2,4 кГц разойдутся на несколько дБ, как бы ни были откалиброваны оба. Сначала согласуйте вид модуляции и полосу.

Это шкала страницы, а не физическая. Веб-прибор растягивает свой диапазон ради читаемости, поэтому 10 дБ реального изменения сигнала отображаются примерно как 11. Совпасть с ней означает, что клиенты Kiwi согласуются с вашим приёмником и расходятся с настоящим KiwiSDR на величину, растущую с силой сигнала. Это верный компромисс, когда именно ваша страница — эталон, с которым все сверяются; если вы предпочитаете, чтобы мост оставался физически честным, менять нужно этот раздел.

8. Водопад и спектр

Шкала в дБ — `wf_cal`

Каждый бин водопада перед отправкой пересчитывается в настоящие дБм, по той же шкале, к которой откалиброван S-метр. Поэтому несущая читается **на одном и том же уровне при любом увеличении**, что и даёт настоящий KiwiSDR, а спектр согласуется с собственным S-метром веб-страницы на одной и той же полосе.

```
[kiwi_emulation]
enabled = true
wf_cal = 0.0          # дБ, только клиенты Kiwi
```

wf_cal — не калибровка. 0 — это значение, при котором спектр согласуется с вашим S-метром, и именно на нём его следует оставить, если вы хотите, чтобы цифры что-то значили. Он существует потому, что насколько *сильным* должен выглядеть спектр — дело вкуса и того, как ваш клиент распоряжается диапазоном: приёмник этого проекта работает с -10 просто потому, что так это выглядит правильно в AethersDR. Одно значение двигает спектр и водопад вместе: это одни и те же байты в канале, и протокол Kiwi не может масштабировать их порознь. Если вы хотите, чтобы они различались, это должно прийти из собственных органов управления отображением клиента.

Чтобы проверить шкалу, сравните водопад, просуммированный по полосе пропускания, с S-метром на **тихой** частоте. На несущей полоса SSB исключает саму несущую, и эти две величины несопоставимы.

Частота кадров — `wf_fps_max`

Клиент Kiwi запрашивает самый быстрый водопад, какой может получить (`SET wf_speed=4`), и подстраирует свою прокрутку — и усреднение спектра — под ту частоту, которую сервер обещает выдавать. Если эти две величины расходятся, изображение выглядит вялым, хотя на самом деле ничего не запаздывает.

Приёмник выдаёт $2 \times \text{sps} / \text{fft_size}$ спектров в секунду. Клиенты Kiwi обслуживаются из каждого из них и прореживаются до запрошенной частоты; веб-страница сохраняет свой собственный, более медленный такт и не затрагивается.

```
[kiwi_emulation]
enabled      = true
wf_fps_max   = 23          # кадров в секунду
```

23 — максимум, объявленный самим протоколом KiwiSDR, и безопасное значение по умолчанию. **Всегда побеждает меньшее из `wf_fps_max` и собственной частоты приёмника**, поэтому повышение даёт эффект, только если ваш FFT достаточно быстр, чтобы кадры оставались в запасе. Пример для приёмника на 60 Мвыб/с с БПФ на 4194304 точки — $2 \times 60000000 / 4194304 = 28,6$ кадра в секунду:

<code>wf_fps_max</code>	выдаётся
отсутствует (по умолчанию 23)	23,0 к/с
28	28,0 к/с
40	28,6 к/с — побеждает собственный потолок приёмника

SET `wf_speed` от клиента тоже учитывается: 0 — выключено, 1 = 1 к/с, 2 — четверть максимума, 3 — половина, 4 — максимум. Клиент на узком канале, таким образом, может попросить меньше.

Поможет ли повышение — решает клиент, а не сервер. Клиент, рисуящий каждый кадр по мере поступления, даст более плавную прокрутку; клиент, подстроившийся под ожидаемые 23, просто поставит лишние кадры в очередь, и задержка будет расти. Попробуйте, и если водопад станет казаться более вялым, а не более плавным, верните 23.

Что определяет остальную задержку

Если вы охотитесь за задержкой, большая её часть не в мосте. Окно анализа имеет ширину `fft_size / sps` — 70 мс при БПФ на 4194304 точки и 60 Мвыб/с — и отсчёт дополнительно ждёт до половины этого времени, пока заполнится шаг. Уменьшение `fft_size` вдвое вдвое уменьшает и то и другое и удваивает частоту кадров, но вдвое уменьшает и разрешение по частоте, которым живут узкополосные декодеры (WSPR, FT8, CW). На большинстве приёмников это плохой обмен ради нескольких десятков миллисекунд. Это решение в `[input]`, а не то, на что мост может как-то повлиять.

9. Если что-то выглядит не так

Каждая команда, которую присылает клиент Kiwi, записывается в `/tmp/kiwi_retune.log` — запросы перестройки, смены вида модуляции, аутентификация и всё, чего мост не распознал. Это не стоит ничего измеримого и рассчитано на то, чтобы остаться на месте. Когда клиент ведёт себя странно, этот файл показывает, что он на самом деле запросил, — и обычно это и есть весь ответ.

Симптом	Смотреть
Клиент подключается и сразу отваливается	в конфигурации, с которой сервер был запущен , нет <code>[kiwi_emulation] enabled = true</code>
Нет звука, нет водопада, нет строк в журнале	серверная часть не пересобрана после <code>kiwi_install.sh</code> — выполните <code>./recompile.sh</code> , вариант [1]
<code>kiwi_install.sh</code> останавливается на правке	в сообщении названы файл и якорь; дерево разошлось с тем, что ожидает правка, и этот файл остался нетронутым
Звук есть, но тонкий и тихий	так и ожидается — задайте <code>audio_gain</code> , см. раздел 6
Водопад прокручивается вяло	<code>wf_fps_max</code> , см. раздел 8
Перестройка в клиенте ничего не делает	вход IQ (<code>signal = "iq"</code>); перестройка реализована только для вещественных входов
Изменённая настройка не подействовала	эти ключи читаются при запуске — перезапустите приёмник

APU — на стороне клиента. SET agc= от клиента Kiwi принимается и игнорируется: звук приходит с уже применённой собственной APU PhantomSDR, а органы APU клиента действуют на то, что он получил.

10. Насколько это проверено

Формат в канале сверен с работающим приёмником и прочитан строка за строкой по исходному коду обоих клиентов, на которые он рассчитан, — `KiwiSdrProtocol.cpp` у AetherSDR и `kiwi/client.py` у `kiwiclient`. Структура и размер кадров, порядковые номера, порядок байтов, шкала S-метра, перестройка и водопад — всё совпадает. Кроме того, **работа моста подтверждена с самим настольным приложением AetherSDR**, его сопровождающим, против этого приёмника.

По измерениям на этом приёмнике, тестовым клиентом на действующих сокетaх: перестройка отрабатывает за 35–50 мс, звук идёт в реальном времени без ухода, а водопад выдаёт заданную частоту без задержек и без потерянных кадров.

Если на вашей установке что-то всё же выглядит неправильно, `/tmp/kiwi_retune.log` — первое, куда стоит заглянуть, и сообщение о результате приветствуется в любом случае.

См. также: [Руководство по установке](#) · [Структура проекта](#) · [Руководство пользователя](#)

Разнесённый приём

RECEIVE_DIVERSITY.md

PhantomSDR-Plus — Разнесённый приём (Receive Diversity)

Объедините свой приёмник со **вторым приёмником в другом месте** и слушайте тот из двух, у которого в данный момент сигнал лучше. Когда одна площадка проваливается в замирание, вторая обычно нет — и приём продолжается.

Вторым приёмником может быть другой PhantomSDR-Plus, KiwiSDR, UberSDR или WebSDR. Для первых трёх всё работает в браузере: без изменений на сервере, без файлов конфигурации, без установки чего бы то ни было и без особого доступа к другому приёмнику. WebSDR — исключение: ему нужна небольшая программа на вашем собственном сервере, см. [Использование WebSDR](#) ниже.

Что это делает и чего не делает

Разнесение здесь означает **выбор**: в каждый момент вы слушаете одну из двух площадок, с коротким кроссфейдом при переключении. Это не фазированная решётка, и два сигнала не складываются.

Это осознанное решение, и причина в физике, а не в программе. Два приёмника, разнесённые на сотни километров, слышат одну и ту же передачу по разным ионосферным трассам. Две звуковые формы приходят с несвязанной фазой и слегка разным доплеровским сдвигом, поэтому их сложение звучит гулко и гребенчато-отфильтровано — классический эффект «phasey». Настоящее когерентное сложение требует двух приёмников с **общим тактовым генератором**, выровненных до отсчёта; два независимых приёмника через интернет этого дать не могут.

Выигрыш, таким образом, — это **непрерывность, а не уровень сигнала**:

- На устойчивой трассе, где один приёмник просто лучше, вы будете слышать его и ничего не выиграете. Это нормально.
- На трассе с глубоким QSB замирания на двух удалённых площадках в основном некоррелированы. Площадка А пропадает на три секунды, площадка В — нет, и звук продолжается. Ожидайте «я перестал терять слова», а не «из S3 стало S7».
- Вторым выигрыш часто — **местный шум и QRM**. У двух площадок разные соседи, разные электросети и разные «брызги» от вещательных станций. На шумном диапазоне это может значить больше, чем замирания.

Есть и цена: вы слушаете **с задержкой от одной до трёх секунд**, потому что опережающий поток приходится задержать, чтобы совместить его со вторым. Для прослушивания и декодирования это неважно, но мешает, если вы пытаетесь провести связь.

ПРИМЕЧАНИЕ

Цифровые декодеры (FT8, JS8, WSPR, RADE и остальные) намеренно продолжают использовать ваш **локальный** приёмник, а не объединённый звук. Эти виды работы накапливают сигнал когерентно в течение целого интервала, и смена площадки в середине интервала — это разрыв фазы, который может стоить вам ровно того декодирования, которое разнесение и должно было спасти. Разнесение обслуживает динамик; декодеры сохраняют непрерывный локальный поток.

Быстрый старт

1. Откройте панель **Receive Diversity** — она расположена прямо под окнами декодеров, свёрнутая, с надписью `Receive Diversity – off`. Нажмите, чтобы развернуть.
2. Выберите, чем является второй приёмник: **PhantomSDR+**, **KiwiSDR**, **UberSDR** или **WebSDR**.
3. Введите его адрес — или выберите один из сохранённых — и нажмите **Start**.
4. Настраивайте свой приёмник как обычно. Второй **следует автоматически** — частота, вид модуляции и полоса — при каждой перестройке.

Это вся настройка. На другом приёмнике настраивать нечего, и отдельного органа перестройки нет: он всегда следует за вами.

Адреса

Второй приёмник	Что вводить	Примечания
PhantomSDR+	<code>host:8900</code>	Обычный порт приёмника
KiwiSDR	<code>host:8073</code>	Собственный порт KiwiSDR
UberSDR	<code>host</code>	Его обычный веб-порт , не 8073
WebSDR	<code>host:8901</code>	Собственный порт станции. Нужен ретранслятор — см. ниже

Принимается просто имя хоста, адрес `http://` или полный адрес `ws://`. Если другой приёмник работает по HTTPS, введите имя хоста — соединение будет защищённым.

Сохранённые приёмники

Каждый использованный адрес запоминается — в отдельном списке для каждого типа приёмника: адрес KiwiSDR никогда не будет полезной подсказкой, когда в переключателе выбран WebSDR. Списки хранятся в вашем собственном браузере: они никуда не отправляются, и у каждого слушателя свой.

Кнопка ☰ рядом с полем адреса открывает список.

- **Имена.** Любой записи можно дать имя — `Twente` читается лучше, чем `websdr.ewi.utwente.nl:8901`. Именованные записи показываются зелёным над своим адресом, а при наборе имя стоит рядом с адресом в выпадающем списке. Запись опознаётся по своему адресу, поэтому имя никогда не помещает одну и ту же станцию в список дважды. Пока разнесённый приём работает, приёмник, с которым он соединён, показывает своё имя **жирным и медленно мигающим**, так что список говорит и о том, кого вы на самом деле слышите.
- **add** вводит приёмник вручную, без предварительного подключения к нему.  правит имя и адрес на месте, **X** убирает одну запись, а **clear all** очищает список того типа, что на экране. Enter сохраняет, Escape отменяет.

- ******** открывает собственную веб-страницу этого приёмника в новой вкладке. Сохранён адрес, который набирает программа, поэтому он сначала превращается в адрес для браузера — `ws://` становится `http://`, а путь `/audio`, `/ws` или, у KiwiSDR, `/kiwi/.../SND`, нужный каждому виду приёмника, отбрасывается, поскольку страница станции лежит в корне. Это настоящая ссылка, так что средний щелчок и долгое нажатие работают как обычно.
- **Порядок — ваш.** Двигайте строку кнопками ▲▼ или перетаскивайте её туда, где ей место. Именно в этом порядке выпадающий список предлагает адреса.
- **export и import.** Списки лежат в хранилище самого браузера, которое пропадает при очистке данных сайта — именно от этого здесь защита. Export записывает все четыре списка вместе с именами в файл JSON, который можно сохранить или перенести в другой браузер или на другую машину. Import читает его обратно и спрашивает, **объединить** (merge) его с сохранённым или всё **заменить** (replace).
- **QR.** Рисует все списки в виде QR-кода на экране, чтобы перенести их на телефон — см. «На телефоне» ниже. Слишком большой для сканирования список переходит к тексту рядом, который можно скопировать и вставить где угодно.

Число сохраняемых приёмников не ограничено.

На телефоне

Страница `/mobile` предлагает ту же функцию во вкладке **Div**, рядом с Audio, Bands, Marks, Users и Chat: тип приёмника, адрес, Start и Stop, сохранённые приёмники списком, который выбирается касанием, живые показания и подстройка SNR. В каждой строке есть и ссылка ******** на собственную страницу приёмника. Второй приёмник сам следует за настройкой телефона, точно так же, как на настольной странице. Чего нет — это правки: переименования, порядка, удаления; всё это остаётся на настольной странице, где для этого есть место.

Сохранённый список принадлежит одному браузеру, поэтому телефон начинает с пустого, сколько бы приёмников ни было сохранено на компьютере. Для этого и нужен QR-код: на компьютере нажмите , затем  **QR**, отсканируйте код камерой телефона и вставьте текст в **Import** во вкладке Div. Он спросит, объединить или заменить, а Export на телефоне отправляет список в обратную сторону.

Использование WebSDR

WebSDR — программа Питера-Тьерка де Бура, PA3FWM, работающая в Твенте и на нескольких сотнях других площадок — может служить вторым приёмником, но не напрямую из браузера.

Причина — намеренная проверка с их стороны. WebSDR отклоняет аудиосоединение, если заголовок `Origin` не называет его собственный сайт, а `Origin` — *запрещённое имя заголовка*: браузер устанавливает его по странице, на которой вы находитесь, и ни один скрипт не вправе его изменить. Обойти это на стороне клиента нельзя, да и не должно быть можно.

Поэтому соединение устанавливает небольшая программа на вашем сервере, `websdr_relay.py`. Ваш браузер говорит с ретранслятором, а ретранслятор — с WebSDR.

Установка ретранслятора

Все четыре установщика дистрибутивов предлагают его как необязательный шаг. Для уже существующей установки:

```
cd ~/PhantomSDR-Plus
./setup_websdr_relay.sh
```

Он спрашивает порт, берёт ваш позывной и адрес станции из `frontend/site_information.json` и предлагает установить службу `systemd`, чтобы ретранслятор запускался при загрузке. Настройки лежат в `websdr_relay.json` и меняются в любой момент.

Одного установщик за вас сделать не может: пробросить порт ретранслятора на вашем роутере. Браузеры слушателей подключаются к ретранслятору напрямую — он не идёт через приёмник — поэтому без проброса разнесённый приём с WebSDR работает только из вашей собственной сети. Остальных трёх типов приёмников это не касается.

Если установлена панель администратора, на её дашборде появляется карточка **WebSDR Diversity Relay** с состоянием, портом и числом идущих сессий и их станциями.

Будьте хорошим гостем

Ретранслятор подключается к чужим приёмникам с вашего сервера, а не с адреса каждого слушателя, поэтому он сделан так, чтобы вести себя прилично:

- **Не более десяти одновременных сессий к одному WebSDR** и шестидесяти в сумме. Именно это не даёт оживлённому дню у вас выглядеть как атака на чужую станцию.
- **User-Agent, называющий вашу станцию и её оператора**, чтобы оператор, которому это не по душе, точно знал, кому написать. Заполните его честно.
- **Передаются только команды настройки.** Через это соединение нельзя послать ничего другого.
- **Частные, петлевые и carrier-NAT адреса отвергаются**, чтобы посетитель не мог направить ретранслятор на что-то внутри вашей машины или локальной сети.

ВАЖНО

Проверка `Origin` существует потому, что авторы WebSDR не хотели, чтобы их приёмниками управляли с чужих страниц. Использование ретранслятора — обдуманное решение, а не их недосмотр. Оставьте предел там, где он есть, держите User-Agent честным и прекратите, если оператор вас об этом попросит.

Чем WebSDR отличается

- **Многие станции покрывают лишь несколько узких участков спектра**, а не непрерывный диапазон — например окно 256 кГц на 40 м. Разнесённый приём включается только внутри них, а панель читает список диапазонов станции, чтобы знать, где они.
- **Частота дискретизации звука меняется** вместе с шириной фильтра и отклоняется от номинала сильнее, чем у остальных типов, поэтому выравнивание может занять чуть больше времени.
- **CW не подстраивается** под соглашение WebSDR, помещающее полосу пропускания целиком ниже несущей. SSB и AM корректны.

Как читать панель

В работе панель показывает небольшую таблицу:

Строка	Значение
link	Соединение со вторым приёмником. Нужное состояние — ready . При сбое выводится текст ошибки удалённого сервера и код закрытия WebSocket
aligned	Измеренная задержка между двумя потоками после захвата, иначе searching
corr	Насколько сильно коррелируют два потока. Осмысленно только после выравнивания
remote audio	Декодированный звук, приходящий со второго приёмника. none — янтарным цветом означает, что ничего не декодируется
SNR local / remote	Две сравниваемые оценки отношения сигнал/шум
switches	Сколько раз происходила смена площадки

Рядом с заголовком одно слово говорит, на каком вы этапе:

off (серое)	не запущено
Please wait... (янтарное)	подключается, или подключено и ещё выравнивается
Ready (зелёное)	связано, захвачено и следует за лучшей площадкой
собственная ошибка приёмника (красное)	сбой, и сам он не восстановится

Различие между янтарным и красным экономит время: янтарное означает «ждите дальше», красное — «перестаньте ждать и прочтите сообщение».

Цветная точка добавляет, какую площадку вы слышите на самом деле — зелёная ваш приёмник, голубая удалённый.

Выравнивание занимает около 15 секунд

Это нормально. Два потока выравниваются по корреляции их **огнивающих звука**, а для этого нужно окно сигнала. Поиск идёт в два этапа: узкий поиск в пределах $\pm 1,5$ секунды может начаться, как только накопится около 9 секунд звука, а второе, независимое измерение через пять секунд должно **совпасть** с первым, прежде чем выравнивание будет принято. Если площадки разнесены во времени сильнее — где-то на пути добавилась секунда-другая буферизации — примерно на 14-й секунде вступает полный поиск в пределах ± 4 секунд, и подтверждение приходит на 19-й.

Этот шаг подтверждения важен. Когда две площадки замирают в противофазе — то есть никогда не несут сигнал одновременно — одиночная корреляция с готовностью захватывает уверенную, но совершенно неверную задержку, которая звучит как эхо. Требование двух независимых совпадающих измерений это отсекает.

Прежде чем захват сможет удержаться, должно быть верным ещё одно: частоты дискретизации двух приёмников. Два приёмника — это два тактовых генератора и две цепочки прореживания, поэтому их звуковые потоки приходят **не** совсем с одинаковой скоростью, даже когда оба заявляют 12 кГц: расхождение в 2% — обычное дело, то есть уход в 240 отсчётов в секунду, намного больше допуска выравнивания. Поэтому удалённый поток непрерывно передискретизируется под скорость вашего приёмника. Это отношение измеряется **подсчётом отсчётов**, а не корреляцией, поэтому ему не нужен собственный захват и оно устанавливается за первые несколько секунд — ещё до того, как запустится первый поиск выравнивания. Слабый или прерывистый сигнал может потребовать двух-трёх попыток, так что 30 секунд — не повод для тревоги; минута без результата — повод.

Пока показано `searching`, обе величины SNR читаются как `0.0`. Так задумано: обе оценки измеряются на **совмещённых по содержанию** отсчётах, поэтому до выравнивания ничего не измеряется. `0.0 / 0.0` означает «ещё не захвачено», а не «нет сигнала».

Remote SNR trim

Этот регулятор смещает выбор между двумя площадками. Он прибавляется к измеренному SNR удалённого приёмника перед сравнением — положительные значения благоприятствуют удалённому, отрицательные — вашему. Больше он не меняет **ничего**: ни уровень звука, ни само объединение, только то, какая площадка побеждает.

Он нужен потому, что две величины SNR не всегда сопоставимы. Обе измеряются одинаково — как размах между процентиями звука — но приёмник, кодек которого поднимает собственный шумовой порог, показывает меньше, чем заслуживает. KiwiSDR применяет сильное усиление и лимитер, что сжимает размах; кодек с потерями задаёт шумовой порог, ниже которого сигнал никогда не опускается.

Как его выставить:

1. Настройте оба приёмника на надёжно присутствующий сигнал и дождитесь `aligned`.
2. Понаблюдайте за двумя величинами SNR полминуты. Отметьте обычную разницу.
3. Используйте крайние положения как слуховую пробу: **+15** принудительно включает удалённую площадку, **-15** — вашу. Послушайте каждую по несколько секунд. Это единственный способ услышать одну площадку отдельно.
4. Если удалённый показывает, скажем, на 5 дБ меньше, но звучит так же хорошо, поставьте **+5**. Вы приводите два показания к согласию тогда, когда обе площадки звучат одинаково.
5. Последите за счётчиком **switches** в течение десяти минут прослушивания. Постоянные переключения означают, что значение слишком близко к равенству; полное отсутствие переключений при том, что ваш приёмник слышимо замирает, означает, что вы ушли слишком далеко в другую сторону.

Начальные значения: **0** для другого PhantomSDR-Plus, у которого тракт звука идентичен и который сопоставим по построению; небольшое **положительное** значение для KiwiSDR или UberSDR.

СОВЕТ

Если требуется больше примерно ± 8 дБ, прекратите подстройку. На этом этапе честное объяснение обычно в том, что один приёмник действительно хуже для этой трассы, а смещение сверх реальной разницы означает лишь прослушивание более слабой площадки.

Настройка действует немедленно и запоминается в браузере.

Выбор второго приёмника

Расстояние имеет значение. Замирания декоррелируют с расстоянием — несколько сотен километров на КВ — хороший ориентир. Два приёмника в одном городе замирают вместе и ничего не дают. Слишком далеко — и второй приёмник может вообще не слышать ваш сигнал.

Оба должны действительно слышать сигнал. Это требование не обходит никакая программа. Если второй приёмник не слышит того, что слушаете вы, корреляция останется низкой и захвата не произойдёт — намеренно, потому что неверное выравнивание звучит хуже, чем отсутствие разнесения.

Покрывтие. Второй PhantomSDR-Plus или KiwiSDR сообщают диапазон частот, который они охватывают, и панель предупредит вас, если вы настроитесь за его пределы. WebSDR сообщает свои диапазоны, которые часто представляют собой узкие участки, а не непрерывную полосу. UberSDR покрытие не сообщает, поэтому там отказ от захвата — ваш единственный признак.

ВАЖНО

Публичные приёмники держат добровольцы, и число мест для слушателей у них ограничено. Сеанс разнесения занимает одно из них на всё время работы, ровно как и обычный слушатель. Пожалуйста, не оставляйте соединение открытым бесконечно и спросите оператора, если планируете пользоваться его приёмником интенсивно.

Устранение неполадок

link не выходит из connecting или всё время соединяется и закрывается Скорее всего, неверны адрес или порт. Смотрите таблицу выше — в частности, UberSDR использует свой обычный веб-порт, а не 8073. Панель показывает код закрытия WebSocket, а консоль браузера (F12) — исходную ошибку.

link показывает ошибку с текстом Этот текст приходит от другого приёмника и обычно вполне конкретен: отклонённая сессия, переполненный сервер или приёмник, не принимающий внешних клиентов.

KiwiSDR отказывает в соединении Некоторые KiwiSDR требуют пароль, а на UberSDR совместимая с KiwiSDR точка доступа по умолчанию выключена. Для UberSDR используйте тип **UberSDR** — этот путь доступен всегда.

WebSDR: the WebSDR relay is not reachable Ретранслятор не запущен, или ваш браузер не достаёт до его порта. Он должен быть доступен по **тому же имени хоста, с которого отдаётся страница приёмника**, потому что браузер подключается к нему напрямую, а не через приёмник. Откройте `http://<это имя хоста>:<порт ретранслятора>/status` в том же браузере, чтобы понять, что именно. Если он отвечает в локальной сети, но не снаружи, порт не проброшен на роутере.

WebSDR: ошибка с упоминанием отклонённого соединения Эта станция закрыта строже обычной проверки `origin`, либо её оператор заблокировал вашу станцию.

link в состоянии ready, но remote audio показывает none Звук не декодируется. Перезагрузите страницу; если это повторяется, консоль браузера покажет, какой декодер не запустился.

Звук приходит, но выравнивание не наступает Два приёмника слышат разное. Попробуйте мощную вещательную станцию или диапазон, где у обеих площадок хорошее прохождение. Тихий диапазон, где с обеих сторон только шум, никогда не даст корреляции — и не должен.

Постоянно переключается туда-обратно Две площадки слишком близки по SNR. Немного сместите trim, чтобы одна предпочиталась устойчиво.

Второй приёмник никогда не используется Проверьте, растёт ли `remote audio` и выглядят ли величины SNR разумно. Если удалённый показывает намного меньше, чем звучит, именно для этого и нужен trim.

Ограничения

- **Не когерентно.** Никакого фазирования, подавления или пеленгации. Подавление источника помех требует двух антенн на общем тактовом генераторе на одной площадке; это другая техника для другой задачи.
- **От одной до трёх секунд задержки**, неизбежно.
- **Только два приёмника.**
- **Моно.** Если ваш приёмник работает в стереорежиме вроде C-QUAM, звук проходит без изменений и разнесение не включается.
- **Для выравнивания нужен сигнал.** Ниже примерно 10 дБ SNR на одной из двух площадок ожидайте, что состояние останется `searching`.
- **WebSDR требует ретранслятора** на вашем собственном сервере и проброшенного порта. Остальным трём типам приёмников не нужно ни того, ни другого.

ГЛАВА 15

Управление трансивером

RIG_CONTROL.md

PhantomSDR-Plus — Управление трансивером (CAT)

Держите **свой трансивер** и **приёмник PhantomSDR-Plus** на одной частоте, виде модуляции и фильтре. Вращайте ручку настройки трансивера — водопад следует за ней; щёлкните по сигналу на водопаде — трансивер перестроится на него. При передаче приёмник может замолкать, чтобы не возвращать вам ваш же сигнал.

Работает с приёмниками **PhantomSDR-Plus**, **KiwiSDR**, **PA3FWM WebSDR** и **UberSDR**, вашими или чужими, и всегда перемещает только *ваш* сеанс прослушивания — остальные на приёмнике ничего не слышат и не видят. Оператору приёмника ничего устанавливать или настраивать не нужно.

Что понадобится

Два способа подключить трансивер и одно требование к приёмнику:

Компонент	Что это	Синхронизирует
Desktop PhantomSDR+ 4.0 и новее	Настольное приложение с меню Rig . Linux (ПК и Raspberry Pi) и Windows.	Частоту, вид модуляции, ширину фильтра, заглушение при передаче — в одну сторону или в обе
CATsync Tool for WebSDRs	Отдельная программа для Windows, связывающая трансивер со страницей приёмника в браузере.	Частоту и вид модуляции
TCI-CAT на странице приёмника	Кнопка на самой странице PhantomSDR-Plus, в любом браузере. Работает по TCI с ExpertSDR, AetherSDR или Thetis, а через небольшой мост — с любым трансивером, который поддерживает Hamlib. Только на приёмниках PhantomSDR-Plus версии 4.1.0 или новее.	Частоту, вид модуляции и ширину фильтра в обе стороны; заглушение при передаче
Приёмник	PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR или UberSDR. Приёмнику PhantomSDR-Plus для ширины фильтра и заглушения нужна версия 4.1.0 или новее.	Более старый PhantomSDR-Plus по-прежнему синхронизирует частоту и вид модуляции

Большая часть руководства описывает Desktop PhantomSDR+. У кнопки TCI-CAT есть собственный раздел — [TCI-CAT на странице приёмника](#). У CATsync Tool своя документация на его сайте.

Поддерживаемые приёмники

Приложение распознаёт тип приёмника в окне станции и управляет им через собственные элементы управления этой страницы. Окно Rig control показывает найденный тип рядом с названием станции.

Приёмник	Частота и вид модуляции	Ширина фильтра	Заглушение при передаче
PhantomSDR-Plus	Да	С версией 4.1.0 или новее	С версией 4.1.0 или новее
KiwiSDR (в том числе Web-888)	Да	Да	Да
PA3FWM WebSDR	Да, со сменой диапазона на многодиапазонных сайтах	Да	Да
UberSDR	Да	Да	Да

У веб-приёмника меньше видов модуляции, чем у большинства трансиверов, поэтому некоторые виды трансивера делят один вид приёмника: у KiwiSDR и WebSDR один CW для CW и CW-R. Виды модуляции приёмника в этом руководстве названы так, как в PhantomSDR-Plus; остальные приёмники используют ближайший аналог. WebSDR держит фильтр CW уже 1 кГц, а остальные фильтры — 1 кГц или шире, потому что именно так сама страница отличает CW, а частота вне всех диапазонов сайта WebSDR оставляется без изменений. UberSDR задаёт фильтр шагами ползунка, так что ширина может отличаться от ширины трансивера примерно до 50 Гц. Любой другой тип страницы — например, OpenWebRX — показывает *not a receiver this app can drive*, и ничего не синхронизируется.

Что делает и чего не делает

- Синхронизирует **один трансивер с одним окном приёмника** одновременно.
- Несколько раз в секунду читает обе стороны и, если они расходятся, подстраивает одну под другую. **Не** передаёт, не включает РТТ и никуда не отправляет звук.
- Перестройка приёмника меняет только ваш сеанс. Другие слушатели того же приёмника ничего не заметят, и разрешение оператора не требуется.
- Последовательный порт может открыть только **одна программа одновременно**. Если порт уже занят WSJT-X, аппаратным журналом или утилитой производителя, выберите **flrig** или **rigctld on network**, чтобы делить трансивер, а не бороться за порт.

Быстрый старт

1. Откройте станцию в Desktop PhantomSDR+ как обычно.
2. **Rig → Rig control...**
3. В разделе **Connection** выберите **Built-in**, если ваш трансивер есть в списке, иначе **Hamlib (all rigs)**.
4. Выберите трансивер, последовательный порт и скорость, заданную в меню CAT или CI-V трансивера.
5. В разделе **Sync** оставьте выбранным **Both directions**.
6. Нажмите **Connect**. Два индикатора вверху — трансивер и приёмник — в течение секунды должны показать одну частоту.

Каждая настройка сохраняется сразу при изменении. В следующий раз достаточно **Rig → Connect**, или отметьте **Connect when the app starts**.

Выбор способа подключения

Вариант	Когда использовать	Что нужно
Built-in	Ваш трансивер есть в списке ниже.	Больше ничего
Hamlib (all rigs)	Ваш трансивер — любой другой: Hamlib знает более 300 моделей. Приложение само запускает <code>rigctld</code> из Hamlib на закрытом локальном порту и останавливает его при отключении.	Windows: ничего, Hamlib входит в комплект. Linux: <code>sudo apt install libhamlib-utils</code>
rigctld on network	<code>rigctld</code> уже запущен — на этом или другом компьютере вашей сети.	Хост и порт (по умолчанию 4532)
flrig	<code>flrig</code> уже управляет трансивером для <code>fldigi</code> , <code>WSJT-X</code> или журнала.	Запущенный <code>flrig</code> с его портом XML-RPC (по умолчанию 12345)

Трансиверы со встроенным драйвером

Скорость и адрес CI-V — заводские значения, которые подставляет приложение. **Это лишь отправная точка — укажите то, что задано в меню вашего трансивера.**

Семейство	Модели	Скорость по умолчанию	Примечания
Icom CI-V	IC-7300, IC-7610, IC-705, IC-9700, IC-905, IC-7760, IC-7851, IC-7100, IC-7410, IC-9100, IC-7600, IC-7200, IC-7700, IC-7000, IC-7800, IC-756PROIII, IC-756PROII, IC-R8600 и любой другой аппарат с CI-V	19200	Адрес CI-V подставляется по модели (IC-7300 94, IC-705 A4, IC-9700 A2, IC-7610 98 ...)
	IC-746PRO, IC-718, IC-R75	9600	
Xiegu (CI-V)	G90, X6100	19200	Адрес 70; проверьте в меню
Yaesu, новый CAT	FTDX101D/MP, FTDX10, FT-710, FT-991/A, FT-891, FTDX5000, FTDX3000, FTDX1200, FT-950, FT-2000, FT-450/450D	38400	
Yaesu, классический CAT	FT-817/818, FT-857/857D, FT-897/897D	38400	2 стоповых бита; шаг настройки 10 Гц
Kenwood	TS-990S, TS-890S, TS-590S/SG	115200	
	TS-480, TS-2000, TS-870S	57600	
Elecraft	K4, K3/K3S, KX3, KX2	38400	Ширина фильтра синхронизируется
Kenwood-совместимые	FlexRadio SmartSDR CAT (виртуальный порт), QRP Labs QMX/QMX+/QDX, (tr)uSDX, Lab599 Discovery TX-500, другие Kenwood-совместимые	9600-38400	

Трансивер, который должен быть совместим, но не находит общего языка со встроенным драйвером, обычно работает через **Hamlib** — он справляется с гораздо большим числом вариантов.

Настройки последовательного порта

Параметр	Что указать
Serial port	Порт трансивера. USB-адаптеры и трансиверы с USB-портом идут первыми. Other / network address... принимает порт, которого нет в списке — COM7, /dev/ttyUSB1 — или tcp://host:port для последовательного порта, отдаваемого по сети через ser2net или аналог.
Speed (baud)	Ровно то, что указано в меню скорости CAT / CI-V трансивера. Неверная скорость выглядит как трансивер, который никогда не отвечает.
Stop bits	1 почти для всех; 2 для семейства FT-817/857/897.
CI-V address	Только Icom, в шестнадцатеричном виде (94, а не 148). Должен совпадать с меню адреса CI-V.
DTR / RTS	Оставьте выключенными , если они не нужны вашему интерфейсу. Многие CAT-кабели по одной из этих линий включают передатчик или сбрасывают трансивер.
Hardware flow control	Выключено, если руководство не требует RTS/CTS.

Для **Hamlib** те же настройки передаются в `rigctld`. Там у стоповых битов есть ещё вариант *Rig default*, а **Extra rigctld options** принимает любые другие параметры `rigctld`, например `--set-conf=post_write_delay=10`. **rigctld program** позволяет указать конкретный `rigctld`, если их установлено несколько.

Синхронизация

Направление

Вариант	Что происходит
Rig → receiver	Окно приёмника следует за трансивером. Изменение на водопаде возвращается к частоте трансивера.
Receiver → rig	Трансивер следует за окном приёмника. Поворот ручки трансивера отменяется.
Both directions	Побеждает сторона, которой вы коснулись последней . В момент подключения, пока ни одной не касались, побеждает трансивер.

Направление можно сменить и в меню **Rig** во время подключения.

Как стороны не дают друг другу «воевать»

Каждое значение, записанное приложением, мгновение спустя появляется как изменение на другой стороне. Если понимать это буквально, трансивер и приёмник гонялись бы друг за другом бесконечно. Приложение избегает этого тремя способами:

- Страница приёмника применяет изменение сразу, поэтому сразу после записи она перечитывается, и это значение становится новой отправной точкой.
- Трансивер применяет изменение чуть позже, поэтому каждое отправленное ему значение запоминается. Когда трансивер сообщает это значение, оно распознаётся как запись приложения, а не как рука на ручке.

- Значение, которое трансивер отвергает — например, широкополосную FM на КВ-трансивере, — отправляется **дважды**, а затем остаётся в покое, пока не изменится сторона-источник, вместо того чтобы повторяться несколько раз в секунду.

Перестройка страницы приёмника может заставить её выбрать вид модуляции диапазона по умолчанию (например, LSB ниже 10 МГц). Когда ведёт трансивер, приложение сразу возвращает его вид модуляции, так что трансивер в USB на 40 м удерживает приёмник в USB.

Какое окно приёмника

Receiver window выбирает, какая станция следует за трансивером:

- **The station window last in front** (по умолчанию) — при двух открытых станциях щёлкните в одну, и трансивер будет следовать за ней.
- **Конкретная станция** — трансивер привязан к ней, на переднем она плане или нет. Если эта станция не открыта, ничего не синхронизируется, пока она не откроется.

Частота обновления

Update every задаёт, как часто читаются обе стороны: 150 мс, 300 мс (по умолчанию), 500 мс или 1 с. Быстрее — отклик на ручку ощущается мгновенным; медленнее — бережнее к старому трансиверу на 4800 или 9600 бод, где каждое чтение занимает ощутимое время на линии.

Виды модуляции

Вид на трансивере	Приёмник слушает в
USB, LSB	USB, LSB
CW	CW
CW-R (реверс)	CW-L
AM, синхронная AM, DSB	AM
FM, узкая FM	FM
Широкая FM	WBFM
RTTY / FSK	LSB
RTTY-R / FSK-R	USB
Режимы данных (USB-D, DATA-U, PKTUSB, DIG)	USB
Данные LSB, данные FM	LSB, FM

Вид на приёмнике	Трансивер переводится в
USB, LSB	USB, LSB
CW	CW
CW-L	CW-R
AM, QUAM	AM
FM	FM
WBFM	WFM — большинство КВ-трансиверов его отвергают и после двух попыток остаются в покое
RADE (верхняя / нижняя)	USB / LSB

Трансивер в **режиме данных** в нём и остаётся: USB приёмника считается совпадающим с USB-D трансивера, поэтому приёмник никогда не выводит трансивер из режима данных.

Ширина фильтра

Отметьте **Sync filter width**, чтобы полосы пропускания совпадали. Разница менее 60 Гц считается равенством, ведь у двух фильтров никогда не бывает одинаковых шагов.

Подключение трансивера	Ширина фильтра
Hamlib	Да, если Hamlib поддерживает это для данной модели
flrig	Да
Встроенный Icom CI-V	Да — шаг 50 Гц до 500 Гц, затем шаг 100 Гц до 3,6 кГц; AM шагом 200 Гц до 10 кГц; не в FM
Встроенный Elecraft	Да, шагом 10 Гц
Встроенные Kenwood, Yaesu, семейство FT-817	Нет — эти трансиверы выбирают фильтры по таблицам конкретной модели. Для фильтра используйте Hamlib

Приёмники KiwiSDR, WebSDR и UberSDR всегда имеют управление фильтром. Приёмник PhantomSDR-Plus должен иметь версию **4.1.0** или новее; на более старом частота и вид модуляции по-прежнему синхронизируются, а окно Rig control объясняет, почему фильтр не следует.

Заглушение при передаче

Отметьте **Mute receiver while transmitting**. Пока трансивер на передаче, окно приёмника заглушено, а по окончании передачи звук возвращается. Кнопка отключения звука приёмника это показывает, и звук всегда можно вернуть вручную.

Если вы уже заглушили приёмник сами, он останется заглушенным и после.

Нужно подключение, сообщающее состояние передачи, — все встроенные драйверы, flrig и Hamlib для большинства моделей, — а на приёмнике PhantomSDR-Plus ещё и версия 4.1.0 или новее.

Сдвиг частоты

Frequency offset прибавляется к частоте трансивера, давая частоту приёмника:

частота приёмника = частота трансивера + сдвиг

Схема	Сдвиг
Трансвертер на 2 м с трансивером на 10 м (144,100 МГц отображается как 28,100 МГц)	116000000
Трансвертер на 70 см с трансивером на 2 м (432 → 144)	288000000
Без трансвертера	0

Меню Rig

Пункт	Действие
Rig control...	Открывает окно Rig control
Connect / Disconnect <i>название трансивера</i>	Запускает или останавливает синхронизацию; для Hamlib также запускает или останавливает rigctld
Rig to receiver / Receiver to rig / Both directions	Направление синхронизации
Sync filter width	Вкл. / выкл.
Mute receiver while transmitting	Вкл. / выкл.
Строка состояния	<i>Not connected, Connecting..., Connected:</i> <i>название</i> или последняя ошибка

Живая индикация частоты находится в окне Rig control, а не в меню, которое иначе закрывалось бы при каждом изменении.

Linux

Доступ к последовательному порту. Последовательные порты принадлежат группе `dialout`. Пользователь вне её получает *Could not open ttyUSB0*. Добавьте себя один раз, затем выйдите из сеанса и войдите снова:

```
sudo usermod -aG dialout $USER
```

Hamlib. Установите из дистрибутива:

```
sudo apt install libhamlib-utils
```

Пакет `.deb Desktop PhantomSDR+` рекомендует его, поэтому `sudo apt install ./phantomsdr-plus-desktop_4.0.0_amd64.deb` установит его заодно; `dpkg -i` рекомендуемые пакеты не устанавливает. Встроенным драйверам и flrig Hamlib не нужен.

Windows

Собственный `rigctl.exe` из Hamlib входит и в 64-битный, и в 32-битный установщик. COM-порты отображаются в списке по имени (COM3). Если трансиверу нужен USB-драйвер, сначала установите драйвер производителя — до этого порта не существует.

TCI-CAT на странице приёмника

Страница приёмника PhantomSDR-Plus может следовать за трансивером и **сама**, без настольного приложения: браузер напрямую обращается к серверу **TCI** на вашем собственном компьютере. TCI — протокол управления через WebSocket программ ExpertSDR2/ExpertSDR3 (SunSDR), AetherSDR (FlexRadio) и Thetis (Apache Labs ANAN, Hermes). Для трансивера без TCI небольшой мост из комплекта PhantomSDR-Plus выдаёт любой трансивер Hamlib за сервер TCI.

Нужен приёмник PhantomSDR-Plus версии **4.1.0 или новее**. На страницах KiwiSDR, WebSDR и UberSDR этой функции нет.

Кнопка TCI-CAT

Кнопка находится в ряду с **VFO**, **Modes**, **Bands** и **IF Filters** — **TCI-CAT** в широкой раскладке, **CAT** в компактной. Её точка **зелёная**, пока сервер TCI подключён, и **серая** в остальных случаях. Кнопка открывает окно, в котором: До версии 4.2.0 кнопка называлась **QRG Sync**.

Элемент	Назначение
Состояние	<i>Active</i> — TCI (port 50001) при подключении; <i>Inactive</i> с <i>Wait</i> во время поиска
CAT Sync	Включает и выключает синхронизацию частоты, вида модуляции и фильтра. Заглушение при передаче работает в любом случае
Host	Компьютер с сервером TCI: <code>localhost</code> для этого компьютера, иначе его адрес в локальной сети, например <code>192.168.1.42</code> . Нажмите Enter или щёлкните вне поля, чтобы переподключиться

Больше выбирать нечего: порты **50001** (ExpertSDR3, AetherSDR) и **40001** (ExpertSDR2, Thetis) опрашиваются одновременно и повторно каждые несколько секунд, поэтому страница подключается, как только программа запущена. **CAT Sync** и **Host** браузер запоминает.

Браузеры, которые спрашивают разрешение, прежде чем веб-страница обратится к локальной сети, — так делают свежие версии Chrome и Edge — спросят один раз, при первом подключении. Разрешите, иначе точка останется серой.

Что синхронизируется

	Поведение
Трансивер → приёмник: частота	Двигается только шкала. Масштаб, яркость и контраст остаются как были; водопад прокручивается, только если частота уходит за экран, а частотный план не меняет вид модуляции
Трансивер → приёмник: вид модуляции	Следует, когда меняется вид модуляции трансивера — USB, LSB, CW, AM, FM; цифровые виды как USB или LSB. Вид, которого у приёмника нет, ничего не меняет. Работающий декодер сохраняет свой вид модуляции
Трансивер → приёмник: ширина фильтра	Полоса пропускания приёмника принимает ширину фильтра трансивера: смена фильтра или кнопка FIL на трансивере меняют полосу пропускания. Работающий декодер сохраняет свою полосу
Приёмник → трансивер: частота	Ввод частоты, метка или закладка, щелчок по водопаду и перетаскивание полосы пропускания перестраивают VFO трансивера. При перетаскивании отправляются только изменившиеся частоты
Приёмник → трансивер: вид модуляции	Любая смена вида модуляции на странице — кнопка, частотный план, декодер, закладка — устанавливает вид модуляции трансивера (USB, LSB, CW, AM, FM; RADE как USB или LSB). Трансивер в цифровом режиме (USB-D) в нём и остаётся
Приёмник → трансивер: ширина фильтра	Кнопки IF Filters , ползунок IF, перетаскивание полосы пропускания и смена вида модуляции задают фильтр трансивера; при перетаскивании отправляется только итоговая ширина. На трансиверах Icom мост вместо этого выбирает FIL1, FIL2 или FIL3 — см. Фильтры Icom
Без эха	Изменение, пришедшее от трансивера, никогда не отправляется ему обратно, а сообщение о прежнем виде модуляции или фильтре, пришедшее сразу после изменения на странице, игнорируется
Заглушение при передаче	Работает всегда, даже с выключенным CAT Sync: страница замолкает, пока трансивер передаёт. Если двигать ползунок громкости во время передачи, звук не возвращается; на приёме он включается с текущим положением ползунка
Не поддерживается	Split, VFO B, RIT/XIT и сдвиг трансвертера. Отслеживается только VFO A первого приёмника

Вид модуляции и ширина фильтра передаются собственными командами TCI (`modulation`, `rx_filter_band`), поэтому ExpertSDR, AetherSDR и Thetis тоже должны за ними следовать; проверен только мост Hamlib. Для сдвига трансвертера используйте Desktop PhantomSDR+. Не используйте оба способа одновременно с одним трансивером — два контроллера будут перетягивать шкалу.

С какими трансиверами это работает

Трансивер	Работает	Как
SunSDR (ExpertSDR2/3), FlexRadio (AetherSDR), ANAN/Hermes (Thetis)	Да	Напрямую — включите сервер TCI в программе. С этими программами ещё не опробовано; проверено с имитацией сервера TCI
Трансивер с портом CAT, который поддерживает Hamlib, — большинство Icom, Yaesu, Kenwood, Elecraft, Xiegu, QRP Labs	Да	Через мост Hamlib, описанный ниже. Опробовано на Icom IC-7300 в Linux и Windows; другие драйверы могут отличаться названиями видов модуляции или сообщением о PTT
Трансивер без порта CAT или не поддерживаемый Hamlib	Нет	Частоту неоткуда прочесть

Мост Hamlib (IC-7300 и другие трансиверы без TCI)

`tci-bridge/tci-rigctld.mjs` в каталоге PhantomSDR-Plus превращает любой трансивер, которым умеет управлять Hamlib, в сервер TCI для страницы. Он работает на **вашем** компьютере — том, что подключён к трансиверу, рядом с браузером, — а не на приёмнике. Четыре раза в секунду он спрашивает у Hamlib частоту, вид модуляции, ширину фильтра и состояние передачи, отправляет странице только изменения и передаёт трансиверу изменения частоты, вида модуляции и фильтра со страницы.

До Hamlib он добирается одним из двух способов:

Способ	Цепочка	Когда
<code>--rigctl</code> (рекомендуется)	трансивер → <code>rigctl</code> → <code>tci-rigctld.mjs</code> → страница	Обычно. Мост сам запускает <code>rigctl</code> из Hamlib: одно окно, никакого сетевого порта. Используйте его в Windows , где <code>rigctld.exe</code> часто отклоняется с сообщением <i>Access is denied</i>
<code>rigctld</code>	трансивер → <code>rigctld</code> → <code>tci-rigctld.mjs</code> → страница	Другая программа, например WSJT-X, должна работать с трансивером одновременно — см. Совместное использование трансивера

Что нужно

	Linux	Windows
Hamlib	<code>sudo apt install libhamlib-utils</code> или пакет hamlib вашего дистрибутива	<code>hamlib-w64-....zip</code> с github.com/Hamlib/Hamlib/releases , распакованный в <code>C:\hamlib</code> , — <code>rigctl.exe</code> лежит в <code>C:\hamlib\bin</code>
Node.js	18 или новее	LTS Windows Installer (.msi) с nodejs.org , с параметрами по умолчанию
Пакет <code>ws</code>	Внутри каталога PhantomSDR-Plus находится автоматически	Скопируйте <code>tci-rigctld.mjs</code> в папку, например <code>C:\tci-bridge</code> , и один раз выполните там <code>npm install ws</code>
Порт трансивера	<code>ls /dev/serial/by-id/</code> . Один раз вступите в группу <code>dialout</code> : <code>sudo usermod -aG dialout \$USER</code> , затем выйдите из системы и войдите снова	Установите USB-драйвер производителя; номер COM-порта — в Диспетчер устройств → Порты (COM и LPT)

Порт CAT трансивера может занимать только одна программа. Перед запуском моста закройте WSJT-X, flrig, JS8Call, RS-B41 или подключение к трансиверу в Desktop PhantomSDR+.

Пример: Icom IC-7300

На трансивере: **MENU → SET → Connectors → CI-V — CI-V USB Baud Rate 115200, CI-V Transceive ON**. Номер модели IC-7300 в Hamlib — 3073.

Linux. Трансивер — это строка с IC-7300 в выводе `ls /dev/serial/by-id/`, обычно это ещё и `/dev/ttyUSB0`.

1. Проверьте, что Hamlib связывается с трансивером, — должна вывестись частота, например `14280000: bash rigctl -m 3073 -r /dev/ttyUSB0 -s 115200 f`
2. Запустите мост и оставьте его работать: `bash cd ~/PhantomSDR-Plus/tci-bridge node tci-rigctld.mjs --rigctl rigctl -m 3073 -r /dev/ttyUSB0 -s 115200`

Windows. В Диспетчере устройств трансивер виден как *Silicon Labs CP210x USB to UART Bridge (COM4)* — используйте свой номер COM-порта. В командной строке:

1. Проверьте, что Hamlib связывается с трансивером, — должна вывестись частота: `bat C:\hamlib\bin\rigctl.exe -m 3073 -r COM4 -s 115200 f`
2. Запустите мост и оставьте окно открытым: `bat cd C:\tci-bridge node tci-rigctld.mjs --rigctl C:\hamlib\bin\rigctl.exe -m 3073 -r COM4 -s 115200` Строка длинная: убедитесь, что она действительно заканчивается на `-s 115200`, иначе `rigctl` остановится с сообщением `Type: rigctl --help`.

В обеих системах мост через секунду выводит:

```
rig answered through rigctl
rig -> page 14280000 Hz
rig -> page mode USB
rig -> page RX
```

Затем откройте страницу приёмника в браузере на том же компьютере, откройте **TCI-CAT** и включите **CAT Sync**. Точка станет зелёной, а мост выведет `page connected`. Покрутите VFO — приёмник следует за ним; щёлкните по водопаду — трансивер перестраивается (`page -> rig ... Hz`); перейдите на передачу — страница замолкает. **Ctrl+C** останавливает мост, а вместе с ним и `rigctl`.

Всё, что идёт после `--rigctl`, — это программа `rigctl` и её собственные параметры, те самые, что сработали при проверке. Это правило для любого трансивера: **если `rigctl ... f` выводит частоту, мост работает с теми же параметрами.**

Фильтры Icom (FIL1, FIL2, FIL3)

Трансиверы Icom, такие как IC-7300, принимают не любую ширину фильтра: у них три фильтра — **FIL1**, **FIL2** и **FIL3**, ширина каждого задаётся в меню трансивера. Получив ширину, Hamlib выбрал бы один из них и заодно перезаписал бы его ширину — а на IC-7300 эта ширина может попасть в фильтр, выбранный до этого, и перепутать настройки. Поэтому для трансиверов Icom мост никогда не отправляет ширину: он берёт фильтр, чья опорная ширина ближе всего к полосе пропускания страницы, и только **выбирает** его собственной командой CI-V трансивера. Ширины, заданные на трансивере, никогда не меняются.

Вид модуляции	FIL1	FIL2	FIL3
USB, LSB (и их цифровые режимы)	2700 Гц	2400 Гц	1800 Гц
CW	1200 Гц	500 Гц	250 Гц
AM	9000 Гц	6000 Гц	3000 Гц
FM	15000 Гц	10000 Гц	7000 Гц

На IC-7300 (`-m 3073`) это включается автоматически. Настройте SSB-фильтры трансивера соответственно — FIL1 2,7 кГц, FIL2 2,4 кГц, FIL3 1,8 кГц (удерживайте **FIL** на трансивере), — и выбор 2,7 / 2,4 / 1,8 кГц в **IF Filters** выбирает FIL1 / FIL2 / FIL3, а нажатие **FIL** на трансивере устанавливает полосу пропускания страницы по ширине этого фильтра. Мост выводит, например, `page -> rig filter 2398 Hz LSB → FIL2 sent`.

Параметр	Назначение
<code>--filters 3000,2400,1800</code>	Другие опорные ширины SSB, в порядке FIL1,FIL2,FIL3; также включает выбор фильтра для другого трансивера Icom
<code>--civ A4</code>	Адрес CI-V трансивера в шестнадцатеричном виде, если он не 94 (IC-705 A4, IC-9700 A2, IC-7610 98)
<code>--filters off</code>	Передавать ширину через Hamlib, как для других производителей

Оба параметра указываются перед `--rigctl`, например `node tci-rigctld.mjs --filters 3000,2400,1800 --civ A4 --rigctl rigctl -m 3085 -r /dev/ttyACM0 -s 19200`. Проверено только на IC-7300. Трансиверы других производителей — Yaesu, Kenwood, Elecraft и остальных — получают ширину страницы через Hamlib, который задаёт её настолько точно, насколько позволяет трансивер.

Другие трансиверы

Те же шаги подходят для любого трансивера, который поддерживает Hamlib; меняются только параметры.

- Подготовьте трансивер.** В его меню запишите скорость CAT (или CI-V) и включите настройку вроде *CAT через USB* или *CI-V transceive*, если она есть. В Windows установите USB-драйвер производителя.
- Найдите номер модели Hamlib** в списке, который выводит `rigctl -l`:
- Linux: `rigctl -l | grep -i 991`
- Windows: `C:\hamlib\bin\rigctl.exe -l | findstr /i 991`
- Найдите порт.** Linux: `ls /dev/serial/by-id/`. Windows: Диспетчер устройств. Некоторые трансиверы создают **два** порта — Yaesu FT-991A, FTDX10 и FT-710 называют их *Enhanced* и *Standard*; CAT работает на порту **Enhanced**.
- Проверьте, затем запустите мост** с `-m <модель> -r <порт> -s <скорость>`, как в примере с IC-7300: сначала `rigctl -m ... -r ... -s ... f`, затем те же параметры после `--rigctl`.

Некоторые номера моделей по Hamlib 4.5 — сверьте их с `rigctl -l`, так как в другой версии Hamlib трансивер может иметь другой номер:

Трансивер	-m	Трансивер	-m
Icom IC-7300	3073	Yaesu FT-991 / FT-991A	1035
Icom IC-705	3085	Yaesu FTDX10	1042
Icom IC-7610	3078	Yaesu FT-710	1049
Icom IC-9700	3081	Yaesu FT-891	1036
Xiegu G90	3088	Yaesu FT-817	1020
Xiegu X6100	3087	Kenwood TS-590SG	2037
Elecraft K3 / K3S	2029	Kenwood TS-890S	2041
Elecraft KX3	2045	Kenwood TS-2000	2014
Elecraft K4	2047	QRP Labs QCX / QDX	2052

Дополнительные параметры, только при необходимости:

- Трансивер Icom с изменённым адресом CI-V:** добавьте `-c` и адрес в десятичном виде — 94h это `-c 148`.

- **Настройка, которую Hamlib предлагает для этого трансивера:** `rigctl -m <модель> -L` выводит их список; задайте нужную через `-C имя=значение`, например `-C post_write_delay=10` для медленного интерфейса.
- **Трансивер переходит на передачу или перезагружается при открытии порта:** некоторые CAT-кабели используют DTR или RTS для PTT; добавьте `-C dtr_state=0FF -C rts_state=0FF`.

Пример: Yaesu FT-991A

С настоящим FT-991A не проверялось — пример следует настройкам Hamlib для этого трансивера; проверка `rigctl ... f` сразу покажет, работает ли он.

На трансивере установите **CAT RATE** (меню 031) в 38400. Номер модели Hamlib для FT-991 и FT-991A — 1035.

USB-кабель FT-991A создаёт **два** последовательных порта. CAT работает на порту **Enhanced**:

- **Linux:** `ls /dev/serial/by-id/` показывает две строки для трансивера; строка, оканчивающаяся на `-if00-port0`, — это Enhanced, обычно `/dev/ttyUSB0`.
- **Windows:** Диспетчер устройств показывает *Silicon Labs Dual CP2105 USB to UART Bridge: Enhanced COM Port (COM5)* и *Standard COM Port*; используйте номер порта Enhanced.

Linux:

```
rigctl -m 1035 -r /dev/ttyUSB0 -s 38400 f
cd ~/PhantomSDR-Plus/tci-bridge
node tci-rigctld.mjs --rigctl rigctl -m 1035 -r /dev/ttyUSB0 -s 38400
```

Windows:

```
C:\hamlib\bin\rigctl.exe -m 1035 -r COM5 -s 38400 f
cd C:\tci-bridge
node tci-rigctld.mjs --rigctl C:\hamlib\bin\rigctl.exe -m 1035 -r COM5 -s 38400
```

Если проверка не выводит ничего полезного, обычная причина — выбран порт Standard вместо Enhanced или CAT RATE не совпадает с `-s`.

Пример: Kenwood TS-590SG

С настоящим TS-590SG не проверялось — пример следует настройкам Hamlib для этого трансивера; проверка `rigctl ... f` сразу покажет, работает ли он.

В меню трансивера установите скорость порта **USB** 115200 (номер пункта меню — в руководстве TS-590SG). Номер модели Hamlib — 2037; у более старого TS-590S — 2031. В Windows сначала установите драйвер виртуального COM-порта Kenwood для USB-разъёма.

USB-кабель создаёт **один** последовательный порт:

- **Linux:** строка трансивера в выводе `ls /dev/serial/by-id/`, обычно `/dev/ttyUSB0`.
- **Windows:** Диспетчер устройств → Порты (COM и LPT), например COM6.

Linux:

```
rigctl -m 2037 -r /dev/ttyUSB0 -s 115200 f
cd ~/PhantomSDR-Plus/tci-bridge
node tci-rigctld.mjs --rigctl rigctl -m 2037 -r /dev/ttyUSB0 -s 115200
```

Windows:

```
C:\hamlib\bin\rigctl.exe -m 2037 -r COM6 -s 115200 f
cd C:\tci-bridge
node tci-rigctld.mjs --rigctl C:\hamlib\bin\rigctl.exe -m 2037 -r COM6 -s 115200
```

Если проверка не проходит, обычная причина — скорость USB не совпадает с `-s` или кабель подключён к разъёму RS-232 (COM), а `-r` указывает на USB-порт.

Другие примеры

Ни один из них не проверялся на реальном оборудовании. В каждой строке указаны параметры, которые ставятся после `rigctl` при проверке и после `--rigctl rigctl` для моста; установите в меню трансивера ту же скорость. Порт для Linux — обычный; уточните его командой `ls /dev/serial/by-id/`. В Windows замените порт номером COM из Диспетчера устройств, а `rigctl` — на `C:\hamlib\bin\rigctl.exe`.

Трансивер	Меню трансивера	Параметры (Linux)	Примечания
Icom IC-705 (USB)	CI-V USB Baud Rate 19200	-m 3085 -r /dev/ttyACM0 -s 19200	Появляются два порта; CI-V — первый (ttyACM0)
Icom IC-7100 (USB)	CI-V USB Baud Rate 19200	-m 3070 -r /dev/ttyUSB0 -s 19200	Максимальная скорость Hamlib для этого трансивера — 19200
Icom IC-7610	CI-V USB Baud Rate 115200	-m 3078 -r /dev/ttyUSB0 -s 115200	
Icom IC-9700	CI-V USB Baud Rate 38400	-m 3081 -r /dev/ttyUSB0 -s 38400	
Yaesu FTDX101D / FTDX101MP	CAT RATE 38400	-m 1040 -r /dev/ttyUSB0 -s 38400	-m 1044 для FTDX101MP. Два порта; используйте Enhanced
Yaesu FTDX10	CAT RATE 38400	-m 1042 -r /dev/ttyUSB0 -s 38400	Два порта; используйте Enhanced, как у FT-991A
Yaesu FT-710	CAT RATE 38400	-m 1049 -r /dev/ttyUSB0 -s 38400	Два порта; используйте Enhanced, как у FT-991A
Yaesu FT-891	CAT RATE 38400	-m 1036 -r /dev/ttyUSB0 -s 38400	Два порта; используйте Enhanced, как у FT-991A
Yaesu FT-450D	CAT RATE 38400	-m 1046 -r /dev/ttyUSB0 -s 38400	Разъём RS-232: используйте адаптер USB — COM
Yaesu FT-817 / FT-818	CAT RATE 38400	-m 1020 -r /dev/ttyUSB0 -s 38400	-m 1041 для FT-818. Нужен CAT-кабель к разъёму ACC
Yaesu FT-857 / FT-897	CAT RATE 38400	-m 1022 -r /dev/ttyUSB0 -s 38400	-m 1023 для FT-897. Нужен CAT-кабель
Kenwood TS-890S (USB)	Скорость USB 115200	-m 2041 -r /dev/ttyUSB0 -s 115200	
Kenwood TS-990S (USB)	Скорость USB 115200	-m 2039 -r /dev/ttyUSB0 -s 115200	
Kenwood TS-590SG / TS-590S (USB)	Скорость USB 115200	-m 2037 -r /dev/ttyUSB0 -s 115200	Подробный пример выше
Kenwood TS-480	Скорость COM-порта 57600	-m 2028 -r /dev/ttyUSB0 -s 57600	Разъём RS-232: используйте адаптер USB — COM
Kenwood TS-2000	Скорость COM-порта 57600	-m 2014 -r /dev/ttyUSB0 -s 57600	Разъём RS-232; максимальная скорость Hamlib для этого трансивера — 57600
Elecraft K4 (USB)	Скорость RS232 115200	-m 2047 -r /dev/ttyUSB0 -s 115200	
Elecraft K3 / K3S	Скорость RS232 38400	-m 2029 -r /dev/ttyUSB0 -s 38400	K3S: USB; K3: последовательный порт или кабель KUSB
Elecraft KX3	Скорость RS232 38400	-m 2045 -r /dev/ttyUSB0 -s 38400	Кабель KXUSB
Elecraft KX2	Скорость RS232 38400	-m 2044 -r /dev/ttyUSB0 -s 38400	Кабель KXUSB
Xiegu G90	Скорость CI-V 19200	-m 3088 -r /dev/ttyUSB0 -s 19200	Hamlib использует адрес CI-V G90 по умолчанию

Трансивер	Меню трансивера	Параметры (Linux)	Примечания
Xiegu X6100	Скорость CI-V 19200	<code>-m 3087 -r /dev/ttyUSB0 -s 19200</code>	Максимальная скорость Hamlib для этого трансивера — 19200
Lab599 TX-500	—	<code>-m 2050 -r /dev/ttyUSB0 -s 9600</code>	Hamlib использует только 9600
ELAD FDM-DUO	—	<code>-m 33001 -r /dev/ttyUSB0 -s 115200</code>	
QRP Labs QDX	—	<code>-m 2052 -r /dev/ttyACM0 -s 9600</code>	USB-порт; скорость не важна, но её нужно указать

Полная команда Linux для IC-705 — пример того, как читать строку:

```
rigctl -m 3085 -r /dev/ttyACM0 -s 19200 f
node tci-rigctld.mjs --rigctl rigctl -m 3085 -r /dev/ttyACM0 -s 19200
```

Трансиверы под управлением другой программы

Некоторыми трансиверами уже управляет программа, которая может взять эту работу на себя, — иногда вообще без моста. Ни один из этих способов не проверялся.

Трансивер и программа	Что делать
SunSDR с ExpertSDR2/3, FlexRadio с AetherSDR, Apache Labs ANAN / Hermes с Thetis	Мост не нужен. Включите в программе сервер TCI и используйте TCI-CAT напрямую — порты 50001 и 40001 находятся автоматически
FlexRadio со SmartSDR (Windows)	Добавьте порт в SmartSDR CAT . Он работает по протоколу Kenwood CAT, поэтому последовательный порт там подходит как TS-2000: <code>-m 2014 -r COM8 -s 57600</code> . TCP-порт работает с моделью FlexRadio в Hamlib: <code>-m 2036 -r 127.0.0.1:12345</code>
Любой трансивер под управлением flrig	Оставьте flrig запущенным и используйте модель flrig в Hamlib; трансивер остаётся общим с fldigi, WSJT-X и аппаратными журналами: <code>-m 4 -r 127.0.0.1:12345</code>
Запущенный rigctld или программа с сервером <i>Hamlib NET rigctl</i>	Запустите мост без <code>--rigctl</code> , указав этот адрес: <code>node tci-rigctld.mjs 50001 127.0.0.1:4532</code>

Как и везде, параметры ставятся после `rigctl` при проверке и после `--rigctl rigctl` для моста. Сетевому адресу вроде `127.0.0.1:12345` параметр `-s` не нужен.

Совместное использование трансивера: через rigctld

Если WSJT-X или аппаратный журнал должны работать с трансивером, пока запущен мост, запустите сервер Hamlib `rigctld` с теми же параметрами, оставьте его работать и подключите к нему обе программы:

```
rigctld -m 3073 -r /dev/ttyUSB0 -s 115200
node tci-rigctld.mjs
```

Без `--rigctl` мост ищет `rigctld` на `127.0.0.1:4532` и выводит `rigctld connected`; `node tci-rigctld.mjs 50001 192.168.1.50:4532` использует `rigctld` на другом компьютере. В WSJT-X выберите трансивер *Hamlib NET rigctl* с тем же адресом. В Windows сервер — это `C:\hamlib\bin\rigctld.exe` с

теми же параметрами; если Windows отвечает *Access is denied*, используйте `--rigctl` и закройте другую программу на время работы моста.

Порты. Мост отдаёт TCI на порту 50001; другой порт указывается первым в командной строке, например `node tci-rigctld.mjs 40001 --rigctl ...`. Если мост и браузер работают на разных компьютерах, укажите в **Host** адрес компьютера с мостом.

Устранение неполадок TCI-CAT

Признак	Вероятная причина	Что делать
Точка остаётся серой	Сервер TCI или мост не запущен; неверный Host ; в браузере отклонено разрешение на доступ к локальной сети	Включите TCI в ExpertSDR/Thetis/AetherSDR или запустите мост; проверьте Host ; разрешите доступ к локальной сети в настройках сайта
<code>rigctl ... f</code> выдаёт ошибку или зависает	Неверный порт или скорость; нет доступа <code>diout</code> (Linux) или USB-драйвера (Windows); порт занят другой программой	Установите <code>-s</code> как в меню трансивера; проверьте порт (<code>ls /dev/serial/by-id/</code> , Диспетчер устройств); закройте другие программы управления
Мост выводит <i>rigctl stopped ... Type: rigctl --help</i>	Команда неполная — обычно не хватает скорости после <code>-s</code>	Введите всю строку заново
Мост выводит <i>The value after -s is missing</i>	Командная строка обрезалась при вставке	Введите конец строки заново, например <code>-s 115200</code>
Фильтр не следует в одну или другую сторону	Выключен CAT Sync , на странице работает декодер или (IC-7300) ширины FIL трансивера отличаются от 2700 / 2400 / 1800 Гц	Включите CAT Sync; остановите декодер; задайте ширины FIL на трансивере или укажите <code>--filters</code> с ширинами трансивера
Мост выводит <i>rigctl stopped ... rig_open: error</i>	<code>rigctl</code> не может открыть порт	Как для <code>rigctl ... f</code> выше
Мост выводит <i>the rig is not answering</i>	Порт открывается, но трансивер не отвечает: неверная скорость или модель, CAT выключен в меню или изменён адрес CI-V Icom	Повторите проверку <code>rigctl ... f</code> ; при изменённом адресе CI-V добавьте <code>-c</code>
Мост выводит <i>rigctld not reachable</i>	Запущен без <code>--rigctl</code> , а <code>rigctld</code> не работает	Добавьте <code>--rigctl ...</code> или сначала запустите <code>rigctld</code>
<code>rigctld.exe</code> сообщает <i>Access is denied</i> (Windows)	Windows отказывается его запускать	Используйте <code>--rigctl</code> — ему нужен только <code>rigctl.exe</code>
Подключено, но приёмник не перестраивается	Выключен CAT Sync	Включите его — заглушение при передаче работает и без него
Вид модуляции трансивера не передаётся	Вид, которому нет соответствия на приёмнике, или драйвер Hamlib сообщает необычное название	Частота всё равно синхронизируется; выберите вид модуляции на странице
Нет заглушения при передаче	Трансивер или его драйвер Hamlib не сообщает о передаче	На странице настраивать нечего
Шкала прыгает туда-обратно	Desktop PhantomSDR+ или другая программа тоже синхронизирует трансивер	Используйте только один контроллер

Для операторов приёмников

Настраивать нечего. Управление трансивером использует небольшой JavaScript-интерфейс, который уже есть в каждой странице PhantomSDR-Plus; ему не нужны ни настройки сервера, ни открытый порт, ни права администратора. Функции фильтра и заглушения появились в версии 4.1.0 — после её установки пересоберите фронтенд (`./recompile.sh`, пункт 2); останавливать приёмник не требуется. Приёмникам KiwiSDR, WebSDR и UberSDR тоже ничего не нужно: приложение использует элементы управления, которые уже есть на их страницах.

Кнопка **TCI-CAT** появилась также в версии 4.1.0 — и снова нужна только пересборка фронтенда. Она не открывает на приёмнике никаких портов: соединение идёт из браузера каждого слушателя к его собственному компьютеру. Мост Hamlib, `tci-bridge/tci-rigctld.mjs`, слушатели запускают у себя дома; приёмник его не использует.

Для разработчиков: интерфейс страницы

И Desktop PhantomSDR+, и CATsync Tool используют эти функции, которые каждая страница PhantomSDR-Plus после загрузки размещает на window (страницами KiwiSDR, WebSDR и UberSDR приложение управляет через их собственные, другие элементы управления):

Функция	Возвращает / делает
<code>catsync_ready</code>	<code>true</code> , как только функции ниже установлены
<code>catsync_getFrequency()</code>	Частота настройки, Гц
<code>catsync_setFrequency(hz)</code>	Настраивает на <code>hz</code>
<code>catsync_getMode()</code>	USB, LSB, CW, CW-L, AM, QUAM, FM, WBFM, RADEU, RADEL
<code>catsync_setMode(mode)</code>	Задаёт вид модуляции; сбрасывает полосу пропускания к значению по умолчанию для этого вида
<code>catsync_getBandwidth()</code>	Полная ширина полосы пропускания, Гц
<code>catsync_setBandwidth(hz)</code>	Задаёт ширину — растёт вверх в USB, вниз в LSB, в остальных видах поровну. Вызывайте после <code>catsync_setMode</code>
<code>catsync_getMute()</code>	<code>true</code> , если звук отключён
<code>catsync_setMute(on)</code>	Отключает или включает звук через кнопку отключения звука страницы

Последние четыре появились в версии 4.1.0, поэтому проверяйте их наличие перед вызовом:

```
if (window.catsync_ready) {
  window.catsync_setFrequency(7074000)
  window.catsync_setMode('USB')
  if (typeof window.catsync_setBandwidth === 'function') window.catsync_setBandwidth(2400)
}
```

Установка частоты перестраивает звук, поэтому вызывайте сеттер только при реальном изменении значения — постоянный вызов сеттера с тем же значением слышен. Старые точки входа в стиле KiwiSDR/WebSDR (`setfreq`, `set_mode`, `freqset_complete`) сохранены для инструментов, которые на них рассчитывают.

Устранение неполадок

Симптом	Вероятная причина	Что делать
<i>Could not open ttyUSB0</i> (Linux)	Нет в группе <code>dialout</code> , или порт занят другой программой	<code>sudo usermod -aG dialout \$USER</code> , выйти и войти; закрыть WSJT-X, журналы, утилиты трансивера
Индикатор приёмника показывает <i>not a receiver this app can drive</i>	Другой тип веб-приёмника, или страница ещё загружается	Поддерживаются PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR и UberSDR; дайте медленной странице несколько секунд
<i>The rig did not answer</i>	Неверная скорость, неверный тип трансивера, неверный адрес CI-V, трансивер выключен	Согласовать скорость с меню трансивера; проверить адрес CI-V; попробовать Hamlib
<i>Hamlib is not installed</i>	<code>rigctld</code> не найден	Linux: <code>sudo apt install libhamlib-utils</code> . Или указать путь в rigctld program
<i>rigctld stopped: ...</i>	Hamlib не смог открыть трансивер — далее его собственное сообщение	Обычно порт или скорость; текст после двоеточия — причина от Hamlib
<i>flrig is not running at ...</i>	flrig закрыт, или его порт XML-RPC другой	Запустить flrig; проверить порт в его настройках
Подключено, но приёмник не двигается	Нет открытого окна станции, или Receiver window привязан к закрытой станции	Открыть станцию или выбрать <i>The station window last in front</i>
Трансивер переходит на передачу при подключении	DTR или RTS включает передатчик через интерфейс	Снять отметки DTR on и RTS on
Фильтр не следует	Приёмник старше версии 4.1.0, или встроенный драйвер Kenwood/Yaesu	Частота и вид модуляции синхронизируются; для фильтра на Kenwood/Yaesu используйте Hamlib
Заглушение при передаче не срабатывает	Приёмник без обновления, или трансивер не сообщает состояние передачи	Как выше
<i>Lost the rig ... reconnecting</i>	Отключён кабель, выключен трансивер или остановился <code>rigctld</code>	Ничего — повтор каждые 3 секунды, работа продолжится, когда трансивер вернётся
Обе стороны постоянно скачут	Трансивером одновременно управляют две программы	Оставьте управление одной программе или делите трансивер через flrig

Известные ограничения

- Один трансивер и одно окно приёмника одновременно.
- Приёмники, кроме PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR и UberSDR, — например, OpenWebRX — не поддерживаются.
- Встроенные драйверы следуют опубликованным протоколам производителей и проверены на имитированных трансиверах и настоящем Hamlib; для трансивера, ведущего себя иначе, запасной вариант — Hamlib.
- Split, VFO B, RIT/XIT и каналы памяти не синхронизируются — только частота активного VFO.

- TCI-CAT на странице приёмника не синхронизирует split, VFO B, RIT/XIT и сдвиг трансвертера и опробован только на IC-7300 через мост Hamlib.
- В Linux пакеты используют Hamlib из дистрибутива; собственный не входит в комплект.

ГЛАВА 16

Структура проекта

PROJECT_STRUCTURE.md

Структура проекта PhantomSDR-Plus

Этот документ даёт полный обзор структуры каталогов PhantomSDR-Plus, организации файлов и связей между компонентами.

Содержание

1. Дерево каталогов
 2. Корневой каталог
 3. Исходный код (`src/`)
 4. Веб-интерфейс (`frontend/`)
 5. Списки частот (`frequencylist/`)
 6. Файлы конфигурации
 7. Система сборки
-

Дерево каталогов

PhantomSDR-Plus

```

├─ add-receiver.sh          # добавляет ещё один приёмник: драйвер, instances/<имя>/,
receivers.toml — см. docs/MULTI_RECEIVER.md
├─ admin_config.json
├─ admin_server.py
├─ autorun
│  ├─ audiotap.js
│  ├─ bandplan.js
│  ├─ decodeworker.js
│  ├─ index.js
│  ├─ manager.js
│  ├─ package.json
│  ├─ pool.js
│  ├─ probe-ft8.js
│  ├─ probe-js8.js
│  ├─ pskreporter.js
│  ├─ spotparse.js
│  ├─ wasm-shim.js
│  └─ wsprnet.js
├─ cf32_to_real.c          # Fobos HF: оставляет I из CF32 от rx_sdr, масштабированный в s16
(собирает setup-fobos.sh)
├─ chat_history.txt
├─ check-go.sh             # старое: сторож цепочки go.sh
├─ config-airspyhf.toml
├─ config.example.hackrf.toml
├─ config.example.rtlsdr.toml
├─ config-fobos-hf.toml
├─ config-fobos.toml
├─ config-hackrf.toml
├─ config-rspla.toml
├─ config-rtl.toml
├─ config-rx888mk2.toml
├─ config.toml
├─ connection_impl.hpp
├─ docs
│  ├─ ADMIN_PANEL_SETUP.md
│  ├─ Aether_config.md
│  ├─ CONNECTION_LIMITS.md
│  └─ de
│     ├─ ADMIN_PANEL_SETUP.md
│     ├─ Aether_config.md
│     ├─ CONNECTION_LIMITS.md
│     ├─ DECODERS.md
│     ├─ EDITING_VARIANTS.md
│     ├─ INSTALLATION.md
│     ├─ MULTI_RECEIVER.md
│     ├─ PhantomSDR-Plus-Documentation-DE.pdf
│     ├─ PROJECT_STRUCTURE.md
│     ├─ RADE_General_INSTALL_MANUAL_LINUX.md
│     ├─ RADE_README.md
│     ├─ README.md
│     ├─ RECEIVE_DIVERSITY.md
│     ├─ RIG_CONTROL.md
│     ├─ THERMAL_GUARD.md
│     └─ USER_GUIDE.md
├─ DECODERS.md
├─ EDITING_VARIANTS.md
├─ el
│  ├─ ADMIN_PANEL_SETUP.md
│  ├─ Aether_config.md
│  ├─ CONNECTION_LIMITS.md
│  ├─ DECODERS.md
│  ├─ EDITING_VARIANTS.md
│  └─ INSTALLATION.md

```

- └─ MULTI_RECEIVER.md
- └─ PhantomSDR-Plus-Documentation-EL.pdf
- └─ PROJECT_STRUCTURE.md
- └─ RADE_General_INSTALL_MANUAL_LINUX.md
- └─ RADE_README.md
- └─ README.md
- └─ RECEIVE_DIVERSITY.md
- └─ RIG_CONTROL.md
- └─ THERMAL_GUARD.md
- └─ USER_GUIDE.md
- └─ es
 - └─ ADMIN_PANEL_SETUP.md
 - └─ Aether_config.md
 - └─ CONNECTION_LIMITS.md
 - └─ DECODERS.md
 - └─ EDITING_VARIANTS.md
 - └─ INSTALLATION.md
 - └─ MULTI_RECEIVER.md
 - └─ PhantomSDR-Plus-Documentation-ES.pdf
 - └─ PROJECT_STRUCTURE.md
 - └─ RADE_General_INSTALL_MANUAL_LINUX.md
 - └─ RADE_README.md
 - └─ README.md
 - └─ RECEIVE_DIVERSITY.md
 - └─ RIG_CONTROL.md
 - └─ THERMAL_GUARD.md
 - └─ USER_GUIDE.md
- └─ fr
 - └─ ADMIN_PANEL_SETUP.md
 - └─ Aether_config.md
 - └─ CONNECTION_LIMITS.md
 - └─ DECODERS.md
 - └─ EDITING_VARIANTS.md
 - └─ INSTALLATION.md
 - └─ MULTI_RECEIVER.md
 - └─ PhantomSDR-Plus-Documentation-FR.pdf
 - └─ PROJECT_STRUCTURE.md
 - └─ RADE_General_INSTALL_MANUAL_LINUX.md
 - └─ RADE_README.md
 - └─ README.md
 - └─ RECEIVE_DIVERSITY.md
 - └─ RIG_CONTROL.md
 - └─ THERMAL_GUARD.md
 - └─ USER_GUIDE.md
- └─ hr
 - └─ ADMIN_PANEL_SETUP.md
 - └─ Aether_config.md
 - └─ CONNECTION_LIMITS.md
 - └─ DECODERS.md
 - └─ EDITING_VARIANTS.md
 - └─ INSTALLATION.md
 - └─ MULTI_RECEIVER.md
 - └─ PhantomSDR-Plus-Documentation-HR.pdf
 - └─ PROJECT_STRUCTURE.md
 - └─ RADE_General_INSTALL_MANUAL_LINUX.md
 - └─ RADE_README.md
 - └─ README.md
 - └─ RECEIVE_DIVERSITY.md
 - └─ RIG_CONTROL.md
 - └─ THERMAL_GUARD.md
 - └─ USER_GUIDE.md
- └─ INSTALLATION.md
- └─ MULTI_RECEIVER.md
- └─ PhantomSDR-Plus-Documentation-EN.pdf
- └─ PROJECT_STRUCTURE.md

```

├── RADE_General_INSTALL_MANUAL_LINUX.md
├── RADE_README.md
├── README.md
├── RECEIVE_DIVERSITY.md
├── RIG_CONTROL.md
├── ru
│   ├── ADMIN_PANEL_SETUP.md
│   ├── Aether_config.md
│   ├── CONNECTION_LIMITS.md
│   ├── DECODERS.md
│   ├── EDITING_VARIANTS.md
│   ├── INSTALLATION.md
│   ├── MULTI_RECEIVER.md
│   ├── PhantomSDR-Plus-Documentation-RU.pdf
│   ├── PROJECT_STRUCTURE.md
│   ├── RADE_General_INSTALL_MANUAL_LINUX.md
│   ├── RADE_README.md
│   ├── README.md
│   ├── RECEIVE_DIVERSITY.md
│   ├── RIG_CONTROL.md
│   ├── THERMAL_GUARD.md
│   └── USER_GUIDE.md
├── sdr-stats
│   ├── package.json
│   ├── readme_de.md
│   ├── readme_el.md
│   ├── readme_es.md
│   ├── readme_fr.md
│   ├── readme_hr.md
│   ├── README.md
│   └── readme_ru.md
├── THERMAL_GUARD.md
├── USER_GUIDE.md
├── websdr2.png
├── websdr3.png
├── websdr.png
├── favicon.ico
├── fftw_wisdom
├── fix_local_geo.py
├── frequencylist
│   ├── 0.TXT
│   └── A26all00.TXT          # нет в репозитории: распаковывается из a26allx2.zip скриптом
├── update-markers.sh
│   ├── a26allx2.zip
│   ├── admin.txt
│   ├── antenna.txt
│   ├── broadcas.txt
│   ├── curl-output.txt
│   ├── currentUpdateFile.txt
│   ├── fmorg.txt
│   ├── generate-current-shortwave.py
│   ├── language.txt
│   ├── mymarkers.json
│   ├── README.md
│   ├── shortwavestations.json
│   ├── site.txt
│   └── update-markers.sh
├── frontend
│   ├── build-all.sh
│   ├── build-default.sh
│   ├── build-mobile.sh
│   ├── debug-title.sh
│   ├── favicon.ico
│   ├── fix-title-python.py
│   └── index.html

```

```

├─ jsconfig.json
├─ LICENSE
├─ make-redirect-stubs.sh
├─ mobile
│   └─ index.html
├─ package.json
├─ package-lock.json
├─ pnpm-lock.yaml
├─ postcss.config.cjs
├─ .prettierrc.json      # правила форматирования Prettier для исходников фронтенда
├─ public
│   └─ analyze_users.py
│   └─ decoders
│       └─ ft8_lib.wasm
│       └─ js8_dict.bin  # словарь слов JS8, загружается только при необходимости
│       └─ js8.wasm      # декодер JS8, собран из jsdsp/js8_wasm
│   └─ logo.jpg
│   └─ stats.html
│   └─ users.html
│   └─ wf-message.json
├─ README.md
├─ site_information.json
├─ src
│   └─ app.css
│   └─ App.svelte
│   └─ assets
│       └─ amateurfrequencies.json
│       └─ background.jpg
│       └─ shortwavestations.json
│       └─ SSTV.png
│       └─ SSTV.svg
│       └─ svelte.png
│   └─ audio.js
│   └─ audio-stream-worklet.js
│   └─ bands-config.js
│   └─ broadcastSchedules.js
│   └─ clientVersion.js
│   └─ cwDecoder.js
│   └─ cw.worker.js
│   └─ cwWorkerProxy.js
│   └─ decoder.worker.js
│   └─ diversity.js
│   └─ diversityList.js
│   └─ eventBus.js
│   └─ events.js
│   └─ fax.js
│   └─ fax.worker.js
│   └─ faxWorkerProxy.js
│   └─ fft.js
│   └─ flacLowLatency.js      # декодер FLAC, воспроизводящий каждый пакет сразу по приходе
│   └─ fsk.js
│   └─ fsk.worker.js
│   └─ fskWorkerProxy.js
│   └─ kiwiSource.js
│   └─ lib
│       └─ backend.js
│       └─ BandSelector.svelte
│       └─ catsync.js
│       └─ chatReply.js      # метка и ветки ответов в чате
│       └─ CheckButton.svelte
│       └─ colormaps.js
│       └─ Counter.svelte
│       └─ DiversityPanel.svelte
│       └─ fftRadix2.js
│       └─ freedv-reporter.js

```

```

├─ FreeDVReporter.svelte
├─ FrequencyInput.svelte
├─ FrequencyMarkers.svelte
├─ FtxSpectrum.svelte
├─ hammeractions.js
├─ LineThroughButton.svelte
├─ Logger.svelte
├─ MagicEyeIndicator.svelte
├─ ModeIdChip.svelte
├─ ModesSelector.svelte
├─ opusMlDecoder.js
├─ PassbandTuner.svelte
├─ Popover.svelte
├─ QrссPanel.svelte
├─ receivers.js          # данные выбора приёмника (/receivers.json)
├─ rnnoise.js           # загружает RNNoise для шумоподавления на основе ИИ
├─ rx.js                # к какому приёмнику относится страница (?rx=)
├─ SMeterAnalog.svelte
├─ SMeterDigital.svelte
├─ Spectrogram.svelte
├─ StatusIndicators.svelte
├─ storage.js
├─ sUnits.js            # шкала баллов S по частоте (S9 = -93 дБм выше 30 МГц)
├─ Tooltip.svelte
├─ VersionSelector.svelte
├─ VideoAreaSelector.svelte
├─ wrappers.js
├─ main.js
├─ mobile
│   ├─ backend.js
│   ├─ bookmarks.js
│   ├─ main.js
│   ├─ Mobile.svelte
│   └─ tuning.js
├─ modeId.js
├─ modeId.worker.js
├─ modeIdWorkerProxy.js
├─ modePriors.js
├─ modules
│   ├─ decode.wasm
│   ├─ encode.wasm
│   ├─ ft4.js
│   ├─ ft8.js
│   ├─ js8-decoder.js
│   ├─ js8-format.js
│   ├─ js8.js
│   ├─ js8-reassembler.js
│   ├─ js8-slots.js
│   ├─ js8-tables.js
│   ├─ package.json
│   ├─ phantomsdrdsp_bg_fallback.js
│   ├─ phantomsdrdsp_bg.js
│   ├─ phantomsdrdsp_bg.wasm
│   ├─ phantomsdrdsp_bg.wasm.d.ts
│   ├─ phantomsdrdsp.d.ts
│   ├─ phantomsdrdsp.js
│   └─ phantomsdrdsp_router.js
├─ wspr.js
├─ olivia.js
├─ playoutControl.js     # измеряет соединение, держит буфер воспроизведения малым
├─ psk31.js
├─ refused.js
├─ remoteSource.js
├─ scanner.js
├─ siteInfo.js          # site_information.json, дополненный собственной копией

```

```

приёмника
├── sstv.js
├── sstv.worker.js
├── sstvWorkerProxy.js
├── uberSource.js
├── unused
│   ├── AudioProcessor.js
│   ├── decoder.js
│   ├── decoding.js
│   └── modules-emsripten
│       ├── dav1d.js
│       ├── dav1dnoWasm.js
│       ├── dav1dnoWasm.js.mem
│       ├── dav1d.wasm
│       ├── decode_ft8.js
│       ├── decode_ft8.wasm
│       ├── FoxenFlac.js
│       ├── jsDSP.js
│       ├── jsDSPnoWasm.js
│       ├── jsDSPnoWasm.js.mem
│       ├── jsDSPnoWasm.wasm
│       ├── jsDSP.wasm
│       ├── libzstd.js
│       ├── LiquidDSP.js
│       ├── opus.js
│       ├── opusnoWasm.js
│       ├── opusnoWasm.js.mem
│       ├── opus.wasm
│       ├── redsea.js
│       └── redsea.wasm
├── unused.js
├── wrappers.js
├── videoRecorder.js
├── vite-env.d.ts
├── waterfall.js
├── webSdrCodec.js
├── webSdrSource.js
├── stats.html
├── svelte.config.js
├── tailwind.config.cjs
├── vite.config.js
├── go.sh # старая цепочка запуска, заменена на start-<radio>.sh
├── install_arch.sh
├── install_fedora.sh
├── install_opensuse.sh
├── install_rade.sh
├── install_rade_ubuntu22.sh
├── install.sh
├── install-stats-server.sh
├── instances # дополнительные приёмники, по папке на каждый (не в git) – см.
docs/MULTI_RECEIVER.md
├── instructions-for-airspy
├── instructions-for-rspla
├── jsdsp
│   ├── compilejs.sh
│   ├── configureredsea.sh
│   ├── extract_EXPORTED_FUNCTIONS.js
│   ├── ft8_wasm
│   │   ├── build_ft8_wasm.sh
│   │   ├── README.md
│   │   └── wasm_wrapper.c
│   └── include
│       ├── avif
│       │   ├── avif.h
│       │   └── internal.h

```

```

├── liquid
│   └── liquid.h
├── js8_wasm          # декодер JS8 на C, собранный в WebAssembly, с тестами и эталонной
проверкой – см. его README
│   ├── build_js8_wasm.sh
│   ├── check_against_reference.sh
│   ├── extract_js8_constants.py
│   ├── extract_js8_varicode.py
│   ├── js8_ab_test.mjs
│   ├── js8_autorun_test.mjs
│   ├── js8_chain_test.mjs
│   ├── js8_constants.c
│   ├── js8_constants.h
│   ├── js8_decode.c
│   ├── js8_decode.h
│   ├── js8_decode_test.c
│   ├── js8_encode.c
│   ├── js8_encode.h
│   ├── js8_frames_test.mjs
│   ├── js8_osd.c
│   ├── js8_osd.h
│   ├── js8_reassembly_test.mjs
│   ├── js8_roundtrip.c
│   ├── js8_slots_test.mjs
│   ├── js8_subtract.c
│   ├── js8_subtract.h
│   ├── js8_wasm_test.mjs
│   ├── js8_wasm_wrapper.c
│   ├── README.md
│   ├── reference
│   │   ├── build_in_container.sh
│   │   ├── build_reference_decoder.sh
│   │   ├── build_reference_frames.sh
│   │   ├── Containerfile
│   │   ├── Containerfile.decoder
│   │   ├── ref_decode.f90
│   │   ├── ref_decode.txt
│   │   ├── ref_frames.cpp
│   │   └── ref_frames.txt
│   ├── ref_gen.f90
│   └── run_tests.sh
├── lib
│   ├── ANR.c
│   ├── arm_funcs.h
│   ├── CMSIS_DSP
│   │   ├── BUILDING.txt
│   │   └── LICENSE.txt
│   ├── dav1d.cpp
│   ├── NB.c
│   ├── NR_spectral.c
│   └── types.h
├── redsea.js
├── redsea.wasm
├── src
│   ├── index.js
│   ├── libzstd.js
│   ├── LiquidDSP.js
│   ├── NoiseProcessing.js
│   └── wbfmpll.cpp
├── kill.sh          # старое: завершает процессы сервера, вызывается из go.sh
├── kiwi_install.sh  # устанавливает мост KiwiSDR в существующее дерево
├── LICENSE
├── logproxy         # ротированные копии журналов панели/прокси/autorun
├── logrotate
├── └── phantomsdr    # конфигурация logrotate для proxy.log + admin.log (перед установкой

```

```

поправьте пути)
├─ manage_admin.sh
├─ markers.json
├─ meson.build
├─ meson_options.txt
├─ phantom_fftw_wisdom
├─ phantomsdr-admin.service # пример юнита systemd для панели (запуск при загрузке, перезапуск
после сбоя)
├─ phantomsdr-proxy.service # пример юнита systemd для прокси, ставится вместе с юнитом панели
├─ phantomsdr-websdr-relay.service # пример systemd-юнита для ретранслятора WebSDR
├─ proxy.py
├─ rade_helper.py
├─ rade_loadtest.csv # вывод rade_loadtest.py (одна строка на ступень нагрузки)
├─ rade_loadtest.py # нагрузочный тест RADE: сколько одновременных декодеров выдерживает
машина – см. docs/RADE_README.md
├─ rade.sh
├─ README.md
├─ recompile.sh
├─ receivers.toml.example # список приёмников для проху.py – скопировать в receivers.toml
├─ _relaunch.sh # старое: помощник отложенного перезапуска цепочки go.sh
├─ request.hpp
├─ setup_admin.sh
├─ setup-airspyhf.sh # цепочка драйвера Airspy HF+: libairspyhf + SoapyAirspyHF + rx_sdr
+ udev
├─ setup-cpufreq-perms.sh # даёт группе право записи в scaling_max_freq, чтобы защита могла
снижать частоту без root
├─ setup-firewall.sh # необязательная защита nftables – см. docs/CONNECTION_LIMITS.md
├─ setup-fobos.sh # цепочка драйвера Fobos: libfobos + SoapyFobosSDR + rx_sdr +
cf32_to_real + udev
├─ setup-hackrf.sh # HackRF: пакет hackrf из дистрибутива + udev (ничего не собирается)
├─ setup-rspla.sh # цепочка драйвера RSP1A: libmirisdr-5 + SoapyMiri + rx_sdr + чёрный
список msi2500 + udev
├─ setup-rtlsdr.sh # драйвер RTL-SDR: пакет дистрибутива или драйвер Blog V4 из
исходников + блокировка DVB + udev
├─ setup-rx888-udev.sh
├─ setup-sdr-common.sh # общие функции трёх сценариев setup выше
├─ setup_websdr_relay.sh # устанавливает ретранслятор WebSDR (порт, идентификация, systemd)
├─ smeter_theme.sh
├─ src
├─ │ audio.cpp
├─ │ audio.h
├─ │ chat.cpp
├─ │ chat.h
├─ │ client.cpp
├─ │ client.h
├─ │ compression.cpp
├─ │ compression.h
├─ │ crash_handler.cpp
├─ │ crash_handler.h
├─ │ events.cpp
├─ │ events.h
├─ │ fft.cpp
├─ │ fft_cuda.cu
├─ │ fft.h
├─ │ fft_impl.cpp
├─ │ fft_mkl.cpp
├─ │ http.cpp
├─ │ kiwi_bridge.h # мост протокола KiwiSDR – см. docs/Aether_config.md
├─ │ listing
├─ │ │ software_info.cpp
├─ │ │ software_info.h
├─ │ samplerreader.cpp
├─ │ samplerreader.h
├─ │ signal.cpp
├─ │ signal.h

```

```

├── spectrumserver.cpp
├── spectrumserver.h
├── utils
│   ├── audioprocessing.cpp
│   ├── audioprocessing.h
│   ├── dsp.cpp
│   └── dsp.h
├── utils.cpp
├── utils.h
├── waterfallcompression.cpp
├── waterfallcompression.h
├── waterfall.cpp
├── waterfall.h
├── websocket.cpp
├── websocket.h
├── start-airspyhf.sh
├── start-all.sh                # запускает каждый приёмник из receivers.toml
├── start-fobos-hf.sh
├── start-fobos.sh
├── start-hackrf.sh
├── start-rspla.sh
├── start-rtl.sh
├── start-rx888mk2.sh
├── stop-websdr.sh
├── subprojects
│   ├── fftw3.wrap
│   ├── flac.wrap
│   ├── glaze.wrap
│   ├── libcds.wrap
│   ├── libflac.wrap
│   ├── libvolk.wrap
│   ├── ogg.wrap
│   ├── opus.wrap
│   ├── tomlplusplus-3.4.0
│   │   ├── CHANGELOG.md
│   │   ├── cmake
│   │   │   ├── install-rules.cmake
│   │   │   ├── project-is-top-level.cmake
│   │   │   ├── tomlplusplusConfig.cmake
│   │   │   ├── tomlplusplusConfig.cmake.meson.in
│   │   │   ├── tomlplusplusConfigVersion.cmake.meson.in
│   │   │   └── variables.cmake
│   │   ├── CMakeLists.txt
│   │   ├── CODE_OF_CONDUCT.md
│   │   ├── CONTRIBUTING.md
│   │   ├── cpp.hint
│   │   └── docs
│   │       ├── images
│   │       │   ├── badge-awesome.svg
│   │       │   ├── badge-C++17.svg
│   │       │   ├── badge-gitter.svg
│   │       │   ├── badge-license-MIT.svg
│   │       │   ├── badge-TOML.svg
│   │       │   ├── banner.ai
│   │       │   ├── banner.png
│   │       │   ├── banner.svg
│   │       │   ├── favicon.ico
│   │       │   ├── logo.ai
│   │       │   └── logo.svg
│   │       ├── pages
│   │       │   └── main_page.md
│   │       └── poxy.toml
│   └── examples
│       ├── benchmark_data.toml
│       └── CMakeLists.txt

```

```

├─ error_printer.cpp
├─ error_printer.vcxproj
├─ examples.hpp
├─ example.toml
├─ merge_base.toml
├─ merge_overrides.toml
├─ meson.build
├─ parse_benchmark.cpp
├─ parse_benchmark.vcxproj
├─ simple_parser.cpp
├─ simple_parser.vcxproj
├─ toml_generator.cpp
├─ toml_generator.vcxproj
├─ toml_merger.cpp
├─ toml_merger.vcxproj
├─ toml_to_json_transcoder.cpp
├─ toml_to_json_transcoder.vcxproj
├─ include
├─ meson.build
├─ toml++
├─ impl
├─ array.hpp
├─ array.inl
├─ at_path.hpp
├─ at_path.inl
├─ date_time.hpp
├─ formatter.hpp
├─ formatter.inl
├─ forward_declarations.hpp
├─ header_end.hpp
├─ header_start.hpp
├─ json_formatter.hpp
├─ json_formatter.inl
├─ key.hpp
├─ make_node.hpp
├─ node.hpp
├─ node.inl
├─ node_view.hpp
├─ parse_error.hpp
├─ parse_result.hpp
├─ parser.hpp
├─ parser.inl
├─ path.hpp
├─ path.inl
├─ preprocessor.hpp
├─ print_to_stream.hpp
├─ print_to_stream.inl
├─ simd.hpp
├─ source_region.hpp
├─ std_except.hpp
├─ std_initializer_list.hpp
├─ std_map.hpp
├─ std_new.hpp
├─ std_optional.hpp
├─ std_string.hpp
├─ std_string.inl
├─ std_utility.hpp
├─ std_variant.hpp
├─ std_vector.hpp
├─ table.hpp
├─ table.inl
├─ toml_formatter.hpp
├─ toml_formatter.inl
├─ unicode_autogenerated.hpp
├─ unicode.hpp

```

```

├── unicode.inl
├── value.hpp
├── version.hpp
├── yaml_formatter.hpp
├── yaml_formatter.inl
├── toml.h
├── toml.hpp
├── LICENSE
├── meson.build
├── meson_options.txt
├── README.md
├── src
│   ├── meson.build
│   └── toml.cpp
├── tests
│   ├── at_path.cpp
│   ├── conformance_burntsushi_invalid.cpp
│   ├── conformance_burntsushi_valid.cpp
│   ├── conformance_iarna_invalid.cpp
│   ├── conformance_iarna_valid.cpp
│   ├── cpp.hint
│   ├── for_each.cpp
│   ├── formatters.cpp
│   ├── impl_toml.cpp
│   ├── leakproof.hpp
│   ├── lib_catch2.hpp
│   ├── main.cpp
│   ├── manipulating_arrays.cpp
│   ├── manipulating_parse_result.cpp
│   ├── manipulating_tables.cpp
│   ├── manipulating_values.cpp
│   ├── meson.build
│   ├── odr_test_1.cpp
│   ├── odr_test_2.cpp
│   ├── parsing_arrays.cpp
│   ├── parsing_booleans.cpp
│   ├── parsing_comments.cpp
│   ├── parsing_dates_and_times.cpp
│   ├── parsing_floats.cpp
│   ├── parsing_integers.cpp
│   ├── parsing_key_value_pairs.cpp
│   ├── parsing_spec_example.cpp
│   ├── parsing_strings.cpp
│   ├── parsing_tables.cpp
│   ├── path.cpp
│   ├── settings.hpp
│   ├── tests.cpp
│   ├── tests.hpp
│   ├── user_feedback.cpp
│   ├── using_iterators.cpp
│   ├── visit.cpp
│   └── vs
│       ├── odr_test.vcxproj
│       ├── test_debug_x64_cpplatest_noexcept_unrel.vcxproj
│       ├── test_debug_x64_cpplatest_noexcept.vcxproj
│       ├── test_debug_x64_cpplatest_unrel.vcxproj
│       ├── test_debug_x64_cpplatest.vcxproj
│       ├── test_debug_x64_noexcept_unrel.vcxproj
│       ├── test_debug_x64_noexcept.vcxproj
│       ├── test_debug_x64_unrel.vcxproj
│       ├── test_debug_x64.vcxproj
│       ├── test_debug_x86_cpplatest_noexcept_unrel.vcxproj
│       ├── test_debug_x86_cpplatest_noexcept.vcxproj
│       ├── test_debug_x86_cpplatest_unrel.vcxproj
│       └── test_debug_x86_cpplatest.vcxproj

```

```

├── test_debug_x86_noexcept_unrel.vcxproj
├── test_debug_x86_noexcept.vcxproj
├── test_debug_x86_unrel.vcxproj
├── test_debug_x86.vcxproj
├── test_release_x64_cpplatest_noexcept_unrel.vcxproj
├── test_release_x64_cpplatest_noexcept.vcxproj
├── test_release_x64_cpplatest_unrel.vcxproj
├── test_release_x64_cpplatest.vcxproj
├── test_release_x64_noexcept_unrel.vcxproj
├── test_release_x64_noexcept.vcxproj
├── test_release_x64_unrel.vcxproj
├── test_release_x64.vcxproj
├── test_release_x86_cpplatest_noexcept_unrel.vcxproj
├── test_release_x86_cpplatest_noexcept.vcxproj
├── test_release_x86_cpplatest_unrel.vcxproj
├── test_release_x86_cpplatest.vcxproj
├── test_release_x86_noexcept_unrel.vcxproj
├── test_release_x86_noexcept.vcxproj
├── test_release_x86_unrel.vcxproj
├── test_release_x86.vcxproj
├── windows_compat.cpp
├── toml++.code-workspace
├── toml.hpp
├── toml++.natvis
├── toml++.props
├── toml++.sln
├── toml-test
├──   ├── meson.build
├──   ├── README.md
├──   ├── tt_decoder.cpp
├──   ├── tt_decoder.vcxproj
├──   ├── tt_encoder.cpp
├──   ├── tt_encoder.vcxproj
├──   └── tt.hpp
├── toml++.vcxproj
├── toml++.vcxproj.filters
├── tools
├──   ├── ci_single_header_check.py
├──   ├── clang_format.bat
├──   ├── generate_conformance_tests.py
├──   ├── generate_single_header.bat
├──   ├── generate_single_header.py
├──   ├── generate_windows_test_targets.py
├──   ├── requirements.txt
├──   ├── utils.py
├──   └── version.py
├── vendor
├──   ├── catch.hpp
├──   ├── json.hpp
├──   └── README.md
├── tomlplusplus.wrap
├── websocketpp.wrap
├── zlib.wrap
├── zstd.wrap
├── tci-bridge
├──   └── tci-rigctld.mjs      # сервер TCI для трансиверов Hamlib (IC-7300...), запускается на ПК
слушателя – см. docs/RIG_CONTROL.md
├── thermal_guard.py          # Защита от перегрева процессора для панели администратора (работает
и отдельно)
├── thermal-guard.service    # пример systemd-юнита для защиты, для установок без панели
├── tmpfiles
├──   └── phantomsdr-logs.conf # оставляет admin.log + проху.log во владении пользователя панели
(перед установкой поправьте пути)
├── update.sh
├── waterfall.sh

```

```

└─ websdr_relay.json.example # шаблон конфигурации (порт, пределы, идентификация станции)
└─ websdr_relay.py           # сам ретранслятор – см. docs/RECEIVE_DIVERSITY.md
└─ websocketpp_asio_connection.hpp # слой Boost >= 1.87: ws_post вместо io_service::post
└─ websocketpp_asio_endpoint.hpp   # слой Boost >= 1.87: ws_work / ws_restart,
max_listen_connections
└─ websocketpp_asio.hpp           # слой совместимости Boost >= 1.87 для websocketpp (io_context,
executor_work_guard)
└─ xgo.sh                         # старое: запускает spectrumserver, вызывается из check-go.sh

```

Корневой каталог

Файлы конфигурации

Файл	Назначение	Когда изменять
config.toml	Конфигурация по умолчанию	Первичная настройка, тесты
config-rtl.toml	Конфигурация для RTL-SDR	При использовании RTL-SDR
config-rspla.toml	Конфигурация SDRplay RSP1A	При использовании RSP1A
config-airspyhf.toml	Конфигурация Airspy HF+	При использовании Airspy
config-fobos-hf.toml	RigExpert Fobos SDR, HF1/HF2 с прямой оцифровкой (0-25 МГц)	При использовании Fobos на КВ
config-fobos.toml	RigExpert Fobos SDR, тракт RF (25-6000 МГц)	При использовании Fobos выше 25 МГц
config-hackrf.toml	Конфигурация HackRF One	При использовании HackRF
config-rx888mk2.toml	Конфигурация RX888 MK2	При использовании RX888
config.example.hackrf.toml	Пример для HackRF One	При использовании HackRF

Скрипты запуска, остановки и обслуживания

Каждый из приведённых ниже `start-*.sh` — это **самодостаточный лаунчер + сторожевой процесс

- журналирование: **он останавливает запущенный экземпляр, поднимает приёмник + spectrumserver, уходит в фоновый режим, автоматически перезапускает цепочку при её завершении и ведёт журнал в logwebsdr.txt. У них общий скрипт остановки и единая блокировка flock (одновременно работает только один приёмник). Правьте только блок RECEIVER CONFIGURATION**** в начале каждого (аргументы приёмника / конфигурация / имя процесса). Шесть сценариев запуска, сделанных по одному шаблону (`start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos.sh`, `start-fobos-hf.sh`, `start-hackrf.sh`), одинаковы ниже этого блока; `start-rx888mk2.sh` — отдельный сценарий. Начиная с v5.0.0 несколько приёмников могут работать рядом: скрипт запуска, запущенный с `INSTANCE=<имя>`, даёт приёмнику свою папку `instances/<имя>/`, свой журнал, блокировку и FIFO — см. [Несколько приёмников](#).

Скрипт	Назначение
<code>install.sh</code>	Автоматизированная установка и сборка
<code>add-receiver.sh</code>	Добавляет на этот компьютер ещё один приёмник: спрашивает, какой приёмник и что он покрывает, устанавливает драйвер, создаёт <code>instances/<имя>/</code> и запись в <code>receivers.toml</code> и предлагает запустить его и перезапустить прокси. Установщики предлагают его в конце — см. Несколько приёмников
<code>start-rtl.sh</code>	Запуск + сторожевой процесс сервера с RTL-SDR (<code>rtl_sdr</code>)
<code>start-rspla.sh</code>	Запуск + сторожевой процесс сервера с SDRplay RSP1A (<code>rx_sdr</code>); использует <code>libmirisdr-5</code> или API SDRplay, если он установлен (<code>RX_DRIVER</code> задаёт принудительно)
<code>start-airspyhf.sh</code>	Запуск + сторожевой процесс сервера с Airspy HF+ (<code>rx_sdr</code>)
<code>start-fobos-hf.sh</code>	Запуск + сторожевой процесс сервера с RigExpert Fobos SDR, HF1/HF2 с прямой оцифровкой (<code>rx_sdr \ cf32_to_real</code>)
<code>start-fobos.sh</code>	Запуск + сторожевой процесс сервера с RigExpert Fobos SDR, тракт RF (<code>rx_sdr</code>)
<code>start-hackrf.sh</code>	Запуск + сторожевой процесс сервера с HackRF One (<code>hackrf_transfer</code>)
<code>start-rx888mk2.sh</code>	Запуск + сторожевой процесс сервера с RX888 MK2 (<code>rx888_stream</code>)
<code>start-all.sh</code>	Запускает каждый приёмник из <code>receivers.toml</code> его собственным скриптом — на станции с несколькими приёмниками укажите его как скрипт запуска в панели администратора; см. Несколько приёмников
<code>stop-websdr.sh</code>	Остановка сервера и его сторожевого процесса — общий для всех приёмников
<code>stop-websdr.sh</code> <экземпляр>	Останавливает только один приёмник (<code>main</code> — основной); без аргумента останавливаются все, как и раньше
<code>setup-rx888-udev.sh</code>	Устанавливает правила udev, чтобы <code>rx888_stream</code> для RX-888 работал без <code>sudo</code>
<code>setup-rspla.sh</code> / <code>setup-fobos.sh</code> / <code>setup-airspyhf.sh</code>	Ставят цепочку драйвера SoapySDR для RSP1A, Fobos или Airspy HF+ — SoapySDR из дистрибутива, драйвер и <code>rx_tools</code> , собранные в <code>sdr_drivers/</code> , правило udev — на любом из четырёх поддерживаемых дистрибутивов. Их запускают пункты 3, 5 и 6 установщика; работают и сами по себе. Общий код в <code>setup-sdr-common.sh</code> — см. Руководство по установке
<code>setup-hackrf.sh</code>	Ставит пакет <code>hackrf</code> из дистрибутива и правило udev для HackRF One на любом из четырёх поддерживаемых дистрибутивов; его запускает пункт 7 установщика
<code>setup-rtlsdr.sh</code>	Устанавливает драйвер RTL-SDR на любом из четырёх поддерживаемых дистрибутивов — пакет <code>rtl-sdr</code> дистрибутива или, с <code>RTL_V4=y</code> , собранный из исходников драйвер RTL-SDR Blog V4, — блокирует драйвер DVB-T и добавляет правило udev; его запускает <code>add-receiver.sh</code>
<code>setup-firewall.sh</code>	Необязательная защита от лавины на уровне ядра: загружает таблицу <code>nftables</code> с потолком одновременных подключений и темпом на адрес источника для портов приёмника, тормозом против перебора паролей SSH и общим доступом Windows, закрытым вне частных диапазонов. Требуется <code>root</code> , не может запереть вас снаружи (<code>policy</code> асерт, установленные подключения принимаются первыми), а <code>--apply</code> сам откатывается, если не подтвердить за 60 с — см. Ограничения подключений
<code>setup-cpufreq-perms.sh</code>	Даёт группе <code>cpufreq</code> право записи в ограничение частоты процессора, чтобы этап <code>throttle</code> стража работал без <code>root</code> . Устанавливает правило <code>tmpfiles.d</code> , чтобы переживать перезагрузку; <code>--revoke</code> всё отменяет
<code>update.sh</code>	Обновление установки из опубликованного дерева без изменения вашей конфигурации, меток, списка частот и локальных правок — см. Руководство по установке

Скрипт	Назначение
<code>recompile.sh</code>	Пересборка бэкенда и/или фронтенда и выбор варианта, отдаваемого по /
<code>smeter_theme.sh</code>	Задать шкалу аналогового S-метра по умолчанию (dark / amber / vintage) для всех пользователей и предложить пересборку фронтенда — см. Редактирование вариантов
<code>waterfall.sh</code>	Меняет минимальный уровень водопада по умолчанию (dB) в <code>waterfall.js</code> + <code>App.svelte</code> — см. README
<code>kiwi_install.sh</code>	Устанавливает эмуляцию клиентов KiwiSDR в дерево, где её нет: правит исходники серверной части, копирует <code>src/kiwi_bridge.h</code> и добавляет документированный блок <code>[kiwi_emulation]</code> в файлы конфигурации в корне. Идемпотентен и сохраняет копию каждого файла, к которому прикасается — см. Эмуляция клиентов KiwiSDR
<code>tci-bridge/tci-rigctld.mjs</code>	Приёмником не используется. Небольшая программа на Node.js, которую слушатель запускает рядом со своим трансивером: она читает частоту, вид модуляции и состояние передачи из <code>rigctld</code> Hamlib и отдаёт их по TCI на порту 50001, чтобы кнопка TCI-CAT на странице управляла трансивером без собственного TCI, например IC-7300, — см. Управление трансивером

Старая цепочка запуска. `go.sh`, `xgo.sh`, `check-go.sh`, `kill.sh` и `_relaunch.sh` — предыдущее поколение скриптов запуска, наблюдения и остановки. Всё, что они делали, теперь находится внутри каждого `start-<radio>.sh`, и пользоваться следует именно им. Они остаются на диске, потому что на них ссылаются существующие установки, и больше не сопровождаются.

Файлы данных

Файл	Назначение	Формат
<code>markers.json</code>	Частотные закладки и маркеры	JSON
<code>chat_history.txt</code>	Сообщения чата пользователей	Обычный текст
<code>favicon.ico</code>	Значок сайта	Изображение ICO
<code>fftw_wisdom</code>	Данные оптимизации FFT	Двоичный формат FFTW
<code>phantom_fftw_wisdom</code>	Дополнительная оптимизация FFT	Двоичный формат FFTW

Панель администратора и отправка спотов

Файл	Назначение
<code>admin_server.py</code>	Сама панель администратора. Помимо страниц управления она выполняет сбор данных для страницы Графики : фоновый поток каждые 2 секунды снимает частоту процессора, нагрузку, температуру и число слушателей и держит их только в памяти — 1 час в полном разрешении плюс 24 часа усреднений по 30 секунд. На диск ничего не пишется, поэтому история теряется при перезапуске.
<code>admin_config.json</code>	Настройки панели администратора (хеш пароля, порты, параметры, пороги тепловой защиты)
<code>thermal_guard.py</code>	Защита от перегрева процессора. Останавливает сервер, когда процессор становится слишком горячим, и запускает его снова после остывания, выводя пороги из критического порога, который публикует ваш собственный процессор, а не из фиксированного числа. Она никогда не пытается определить, что присматривает за сервером: пока держится перегрев, она повторяет остановку каждые 2 с, так что watchdog, юнит systemd или задание cron, оживившие сервер, будут отменены, пока машина не остынет. Только стандартная библиотека; импортируется из <code>admin_server.py</code> (такт задаёт сборщик Графиков) и работает отдельно для установок без панели. По умолчанию только записывает, то есть ни на что не воздействует до включения — см. [Панель администратора]см. руководство Thermal Guard
<code>autorun/</code>	Служба отправки спотов — см. Autorun Spot Reporter
<code>autorun.json</code>	Диапазоны и виды связи для декодирования, данные оператора и приёмники отчётов
<code>autorun-status.json</code>	Текущее состояние службы — питает счётчики текущего запуска (плитки по декодерам), которые обнуляются при Stop/Start
<code>autorun-totals.json</code>	Итоги за всё время по диапазону и виду связи — число рядом с каждым флажком; пишется службой, поэтому переживает перезапуски

Файлы системы сборки

Файл	Назначение
<code>meson.build</code>	Основная конфигурация сборки
<code>meson_options.txt</code>	Настраиваемые параметры сборки
<code>.gitattributes</code>	Атрибуты репозитория Git

Исходный код (`src/`)

Каталог `src/` содержит реализацию серверной части на C++.

Ключевые компоненты

1. Основное приложение (`main.cpp`)

- Разбирает аргументы командной строки
- Загружает файл конфигурации
- Инициализирует компоненты сервера
- Запускает цикл событий

2. Сервер спектра (`spectrumserver.cpp`)

- Координирует все компоненты

- Управляет подключениями пользователей
- Раздаёт данные спектра
- Обрабатывает запросы пользователей

3. Драйверы SDR (`drivers/`)

- Абстрактный интерфейс для оборудования SDR
- Чтение и форматирование данных отсчётов
- Обработка особенностей конкретных устройств

4. Движок ЦОС (`dsp/`)

- Расчёт FFT (с ускорением на ЦП/GPU)
- Демодуляция (AM, FM, SSB, CW и др.)
- Фильтрация и передискретизация звука
- АРУ и шумоподавление

5. Веб-сервер (`server/`)

- Обмен по WebSocket
- Раздача статических файлов по HTTP
- Управление сеансами пользователей
- Поточковая передача данных в реальном времени

6. Кодирование звука (`audio/`)

- Сжатие FLAC
- Сжатие Opus
- Оптимизация потоковой передачи

Веб-интерфейс (`frontend/`)

Веб-интерфейс пользователя, построенный на Svelte и Vite.

1. Основное приложение (`App.svelte`)

- Компонент верхнего уровня
- Структура компоновки
- Оркестрация компонентов
- **Ряд кнопок декодеров** — по одной кнопке на декодер на главной панели, сразу под селектором режимов; одно нажатие запускает декодер и открывает его окно, второе — останавливает. Он заменил прежний ряд полосы пропускания. RADEL/RADEU находятся в `lib/ModesSelector.svelte` — рядом с селектором режимов и во всплывающих окнах **Modes** и **Bands**.

2. Отображение водопада (`waterfall.js + lib/`)

- Отрисовка спектра и водопада на canvas, цветовые палитры и адаптивная автоподстройка — всё в `waterfall.js` (обычный JS, не компонент)
- Интерактивная настройка через `lib/PassbandTuner.svelte`
- Наложения band plan и маркеров через `lib/FrequencyMarkers.svelte`
- Аудиоспектрограмма — отдельный компонент: `lib/Spectrogram.svelte`

3. Органы управления (`App.svelte + lib/`)

- Ввод и индикация частоты — `lib/FrequencyInput.svelte`

- Выбор вида модуляции (AM/FM/SSB/CW) — `lib/ModesSelector.svelte`, смена диапазона — `lib/BandSelector.svelte`
- AGC/NR/NB и остальные органы управления находятся в самом `App.svelte`; отдельного `Controls.svelte` нет

3а. Сканер (`scanner.js`)

- Сканер каналов: ведёт приёмник по диапазону и останавливается на первом канале с сигналом. Чистый JS, не компонент — `App.svelte` предоставляет VFO, вид модуляции, `band plan` и вызов настройки, а получает обратно состояние интерфейса через один обратный вызов
- Порог задаётся в дБ над шумовым порогом диапазона, а не абсолютным уровнем, и использует шум, который `waterfall.js` и так отслеживает (`snrNoiseDb`), поэтому собственная калибровка не нужна
- Два способа пройти диапазон: настроиться и послушать каждый канал либо сначала просмотреть спектр и настраиваться только на то, что нельзя исключить. Спектр может пропустить канал, но никогда не останавливается на нём — любая остановка следует из настоящего прослушивания
- Диапазон — это либо полоса из `band plan`, либо ровно то, что показывает водопад; автоматическое продолжение, предельное время стоянки и список исключённых каналов тоже здесь и сохраняются в `localStorage`

4. Аудиосистема (`audio.js`)

- Аудиопоток по WebSocket
- Декодирование FLAC/Opus
- Управление воспроизведением звука
- Раздаёт исходный PCM (снятый до АРУ, шумоподавления и отключения звука) декодерам режимов
- Шумоподавление на основе ИИ: `lib/rnnoise.js` загружает модуль WebAssembly RNNNoise (`@jitsi/rnnoise-wasm`) при первом использовании; `audio.js` применяет его только в голосовых режимах, после отвода для декодеров, поэтому декодеры его никогда не слышат
- FLAC проходит через `flacLowLatency.js`, который декодирует каждый пакет в момент его прихода и приводит фактическую частоту потока (например, 12016,3 Гц) к 12000 Гц 64-отводным передискретизатором (задержка 2,7 мс). WASM-декодер из `modules/` выдавал звук фиксированными блоками по 1024 отсчёта и удерживал около 160 мс; он по-прежнему используется для Opus, для источников разнесённого приёма и для FLAC, если страница открыта с `?flac=wasm`
- Управление буфером воспроизведения (`playoutControl.js`): `ArrivalMeter` измеряет, насколько неравномерно приходят пакеты относительно собственной шкалы времени потока, а также расхождение между тактовой частотой дискретизации приёмника и звуковой картой слушателя; `CushionController` держит наименьший запас буфера за последнюю минуту около резерва выбранной ступени буфера, вырезая или повторяя один согласованный с высотой тона период (`spliceOut` / `spliceIn`) звука, идущего на громкоговоритель. Декодеры и записи получают нетронутый поток. Действует на пути с запланированными буферами, которым пользуется каждый слушатель станции на обычном `http`; `?drain=0` отключает его, `?diag=1` показывает его показатели

4а. Декодеры режимов и их воркеры

Каждый из тяжёлых декодеров режимов работает в собственном Web Worker, поэтому декодирование никогда не блокирует воспроизведение звука и водопад. Все они следуют общей схеме — по три файла на декодер:

Декодер	Движок	Воркер	Прокси в основном потоке
SSTV	sstv.js	sstv.worker.js	sstvWorkerProxy.js
HF FAX	fax.js	fax.worker.js	faxWorkerProxy.js
NAVTEX + FSK/RTTY + PSK31 + Olivia	fsk.js, psk31.js, olivia.js	fsk.worker.js	fskWorkerProxy.js
CW	cwDecoder.js	cw.worker.js	cwWorkerProxy.js

- **Движок** — это чистый код ЦОС, ничего не знающий о воркерах, поэтому его можно запускать и напрямую (модульные тесты или запасной режим в основном потоке).
- **Воркер** держит один экземпляр движка и дословно пересылает его события.
- **Прокси** повторяет набор методов движка, так что `audio.js` вызывает его точно так же, как вызывал бы сам декодер. Он создаёт воркер лениво, при первом включении, и переходит к работе в основном потоке, если воркеры недоступны.

Две детали здесь принципиальны: PCM **копируется** в новый буфер перед передачей воркеру (передача представления на аудионакопитель отсоединила бы его и остановила воспроизведение), а сообщение `init` воркера повторно применяет конфигурацию к уже работающему движку, а не предполагает создание нового.

`fsk.js` обслуживает и NAVTEX, и FSK/RTTY из одного движка, выбор задаётся для каждого экземпляра полем `role`; у каждого экземпляра своё состояние, поэтому оба могут работать независимо.

Роль `fsk` дополнительно содержит два декодера, которые вовсе не являются FSK. При выборе варианта `psk31` или `olivia` модуль `fsk.js` передаёт звук в `psk31.js` или `olivia.js` вместо собственной дискриминаторной цепочки, продолжая при этом пользоваться его конфигурацией, воркером и системой событий — поэтому `fsk.worker.js`, `fskWorkerProxy.js` и `audio.js` ничего не знают об этих режимах, а интерфейс везде принимает одни и те же события `char/status/metrics`.

- `psk31.js` — BPSK31: комплексная основная полоса, согласованный фильтр, дифференциальное детектирование и `varicode`, со спектральным грубым захватом и точной АПЧ в пределах примерно ± 25 Гц.
- `olivia.js` — Olivia MFSK: перенос MFSK-приёмника Павла Ялохи из `fldigi` (`pj_mfsk.h`, GPL-3, как и данный проект), включая коррекцию ошибок Уолша/Адамара и слепой поиск синхронизации по фазе блока и частотному сдвигу.
- `broadcastSchedules.js` — расписания UTC, которые декодеры FAX, NAVTEX и RTTY предлагают как предустановки, из графиков морского факсимиле NOAA/NWS и опубликованных списков станций NAVTEX

4b. Разнесённый приём (`diversity.js`)

- Объединяет локальный приёмник со вторым в другом месте и следует за той площадкой, у которой сейчас сигнал лучше. Обычный JS, не компонент — опирается на единственный стык в `audio.js`, который передаёт ему локальный PCM и воспроизводит то, что он возвращает
- **Выбор, а не сложение.** Две площадки слышат одну передачу по разным ионосферным трассам, поэтому их формы сигнала имеют несвязанную фазу; их сложение звучит гребенчато. Когерентное сложение потребовало бы общего тактового генератора, которого у двух приёмников через интернет нет. Смешивание происходит только во время кроссфейда 30 мс
- Выравнивание коррелирует две **огигающие звука** (логарифмическая мощность на 100 Гц), а не формы сигнала — огибающая переживает и трассу, и любой кодек. Захват принимается

лишь тогда, когда второй, независимый поиск с ним совпал; это отсекает уверенную, но неверную задержку, которую дали бы две площадки, замирающие в противофазе

- Удалённый поток сначала **захватывается по скорости** к локальному. Два приёмника — это два тактовых генератора и две цепочки прореживания, поэтому их звук приходит с расхождением до 2% даже когда оба заявляют 12 кГц: 240 отсчётов в секунду ухода, который не удержит никакая корреляция. Отношение измеряется по тому, сколько отсчётов реально выдаёт каждая сторона, и применяется передискретизатором, переносящим дробную фазу через границы блоков, так что произвольное отношение держится сколь угодно долго
- Выбор площадки использует процентильный SNR, измеренный на **совмещённых по содержанию** отсчётах, с гистерезисом, таймером удержания и быстрым аварийным выходом, если действующая площадка провалилась. Уровни согласуются шум к шуму, поэтому переключение не меняет фоновое шипение
- Декодеры сохраняют **локальный** поток: FT8, JS8, WSPR и RADE накапливают когерентно в пределах интервала, и переключение в его середине — разрыв фазы, который может стоить декодирования

4с. Источники разнесения (`remoteSource.js`, `kiwiSource.js`, `uberSource.js`, `webSdrSource.js`)

- Один контракт — `onPcm` / `onState` / `tune` / `canReceive` — так что `diversity.js` никогда не знает, что находится на другом конце. Добавление типа приёмника — это один новый файл
- `remoteSource.js` — другой PhantomSDR-Plus через `/audio` (cbor + FLAC), с повторным использованием `createDecoder()` из `lib/wrappers.js`
- `kiwiSource.js` — KiwiSDR: кадры SND, PCM с обратным порядком байтов, включая 10-байтовую метку времени GPS, которую стереопакет вставляет перед звуком
- `uberSource.js` — UberSDR через его собственный `/ws`: Opus за 21-байтовым заголовком, перестройка по открытому соединению. Идентификатор сессии нужно сначала зарегистрировать через `POST /connection`, и он должен быть UUID
- `webSdrSource.js` — WebSDR через `websdr_relay.py` на этом сервере: браузер не может подключиться напрямую, потому что WebSDR проверяет заголовок `Origin`, а менять его скриптам нельзя. Настройка идёт текстовым кадром по тому же соединению; покрытие диапазонов даёт ретранслятор
- `webSdrCodec.js` — аудиоформат WebSDR: побайтово размеченный поток, сжатые блоки которого питают leaky-LMS предсказатель с 20 отводами. Перенесён из клиента WebSDR и сверен отсчёт за отсчётом
- `diversityList.js` — список сохранённых приёмников и правила адресов для четырёх типов, общие для настольной панели и мобильной страницы, чтобы один формат служил обоим. А также перенос — компактный блоб, его QR-код и разборщик, принимающий любую форму, которую эти две страницы когда-либо записывали — и `browserUrl()`, возвращающая набираемый адрес к тому, который может открыть браузер
- `lib/DiversityPanel.svelte` — интерфейс: адрес, тип источника, подстройка SNR и состояние в реальном времени, а также сохранённые приёмники — с именами, с произвольным порядком, с экспортом и импортом в файл JSON, по типу источника в `localStorage`. Всё правится внутри панели: `prompt()` и `confirm()` блокируют главный поток, а именно через него звук передаётся в `playback worklet`. Кнопка  **QR** рисует список в виде кода для сканирования, поскольку `localStorage` принадлежит одному браузеру, а телефон начинает пустым. `mobile/Mobile.svelte` несёт ту же функцию во вкладке **Div**, в собственном простом CSS этой страницы. См.

Разнесённый приём

4d. Определение вида работы (`modeId.js`, `modePriors.js`)

- Отвечает на вопрос «что я слушаю?». Читает тот же необработанный отвод PCM, что и декодеры, и ранжирует вероятные виды работы, чтобы оператор выбрал нужный декодер, а

не перебирал все десять. Он никогда не декодирует: он измеряет физические свойства сигнала и оценивает их по таблице известных видов

- Занимаемая полоса — это непрерывная ширина по уровню -15 дБ вокруг пика, намеренно не величина по 99 % мощности: щелчки манипуляции оставляют длинные хвосты в интеграле мощности, из-за чего любой узкий вид выглядел в несколько раз шире. Скорость манипуляции символами берётся из **мгновенной частоты**, а не из энергий тонов, потому что никакое время накопления не разрешает одновременно сдвиг 170 Гц и символ 100 Бод. Скорость телеграфной манипуляции берётся из огибающей вкл/выкл и заодно служит индикатором скорости CW
- `modePriors.js` добавляет единственную подсказку, которую звук нести не может: где вы настроены. Сигнал 100 Бод / 170 Гц на 518 кГц — это NAVTEX; тот же сигнал на 14,070 МГц — нет. Он лишь перевзвешивает то, что сигнал и так подтвердил, и никогда не выдумывает кандидата
- FT8, JS8 и FT2 намеренно выдаются одной группой: по одной только полосе и расстоянию между тонами они не разделяются, а утверждать обратное значило бы уверенно дать неверный ответ
- Ниже примерно 10 дБ SNR он молчит, а не гадает
- Работает в собственном Web Worker (`modeId.worker.js` + `modeIdWorkerProxy.js`) по той же схеме `engine/worker/proxy`, что и декодеры выше; результат — метка в `lib/ModeIdChip.svelte`

4е. Отклонённые подключения (`refused.js`, `clientVersion.js`)

- `refused.js` — общий словарь для подключения, которое сервер отклоняет: код закрытия **4003** (превышено ограничение по адресу либо страница старше `[server] min_client_version`) и **4001** (отключение системным оператором). Оба окончательные. `audio.js`, `waterfall.js` и `events.js` импортируют оттуда `isRefusal()`, потому что все три открывают подключение, которому могут отказать, и все три обязаны разрешить свой инициализационный промис, когда это случается, — иначе страница вечно ждёт подключения, которого не будет
- Ничто не повторяет попытку. Упавшее подключение `/audio` намеренно завершает сессию: `/waterfall` и `/events` с переподключением никогда не возвращались, так что повторённая сессия была живым звуком поверх замороженного водопада, а против ограничения темпа каждая попытка продлевала бы ровно тот отказ, который пыталась обойти. Настольная страница объясняет отказ, случившийся при загрузке, и просто останавливается, если он случился посреди сессии; `/mobile` показывает одну строку с просьбой перезагрузить
- `clientVersion.js` хранит единственное целое число, `CLIENT_VERSION`, которое страница добавляет к своему звуковому подключению как `/audio?v=N`. Сервер отклоняет всё ниже `[server] min_client_version` — так станция заставляет вкладки, всё ещё работающие на прежней сборке, перезагрузиться; это единственный рычаг, ведь сервер не дотянется до JavaScript, уже работающего в браузере. Увеличивайте его, когда изменение во фронтенде не должно и дальше игнорироваться открытыми вкладками
- Полная справка: [Ограничения подключений](#)

4ф. Ответы в чате (`lib/chatReply.js`)

- Ответ — это обычное сообщение чата, которое начинается с метки, называющей сообщение, на которое оно отвечает: `[RE:<отметка времени><имя>]`; сервер хранит и пересылает его как любую другую строку и не требует изменений
- `chatReply.js` создаёт и читает эту метку и группирует ответы под исходным сообщением на один уровень вглубь. Его используют и `App.svelte`, и `mobile/Mobile.svelte`, поэтому обе страницы выстраивают чат одинаково
- Если исходное сообщение уже вышло из истории сервера в 20 сообщений, ответ остаётся там, где пришёл, и показывает только имя и время, на которые он отвечает

5. Управление состоянием (stores/)

- Реактивные хранилища данных
- Общее состояние приложения
- Обработка событий

Ключевые возможности

- Обработка звука в реальном времени
- Декодирование цифровых видов связи (FT8, RTTY и др.)
- Фильтрация звука
- Спектральный анализ

Списки частот (frequencylist/)

Метки частот на водопаде. `mymarkers.json` — это список, который приёмник действительно показывает; остальное — сырьё, которое `update-markers.sh` в него превращает, обновляемое из сетевых расписаний. `README.md` в этом каталоге объясняет обновление на всех семи языках.

```
frequencylist/
├─ mymarkers.json           # метки, которые показывает приёмник — правятся вручную, переживают
  обновление
├─ shortwavestations.json   # коротковолновые станции, создаётся generate-current-shortwave.py
├─ 0.TXT                   # глобальное KB-расписание A26, распаковывается и разбирается update-
markers.sh
├─ a26allx2.zip            # исходный архив, который скачивает update-markers.sh
├─ admin.txt · antenna.txt · broadcas.txt · fmorg.txt · language.txt · site.txt
├─                         # вспомогательные списки из того же источника (площадки, языки,
администраторы, антенны)
├─ generate-current-shortwave.py
├─ update-markers.sh       # обновляет всё перечисленное из сетевых расписаний
└─ README.md
```

Формат (mymarkers.json, shortwavestations.json)

```
[
  { "frequency": 77500, "name": "DCF77", "mode": "CW" },
  { "frequency": 2485000, "name": "Vanuatu Broadcasting", "mode": "AM" }
]
```

Файлы конфигурации

Конфигурация сервера (файлы .toml)

Структура файлов конфигурации:

```

[server]
# Web server settings
port = 9002
html_root = "frontend/dist/"
threads = 2
otherusers = 1

[limits]
# Ограничения по адресу – все по умолчанию выключены или на разумном значении,
# так что конфигурация без них ведёт себя как раньше. См. CONNECTION_LIMITS.md.
per_ip = 3          # одновременные слушатели с одного адреса
per_ip_rate = 40     # новые подключения в минуту с одного адреса

[websdr]
# Online registration
register_online = true
name = "WebSDR Name"
antenna = "Antenna Type"
grid_locator = "AB12cd"
hostname = "domain.com"

[input]
# SDR input settings
sps = 2048000        # Sample rate
fft_size = 131072    # FFT size
frequency = 145000000 # Base frequency
signal = "iq"         # Signal type: "iq" or "real"
audio_sps = 12000     # Audio sample rate
audio_compression = "opus" # "flac" or "opus"
accelerator = "opencl" # "none", "cuda", "opencl"

[input.driver]
# Driver settings
name = "stdin"
format = "u8"          # Sample format

[input.defaults]
# User interface defaults
frequency = 145000000
modulation = "FM"

```

Сведения о площадке (site_information.json)

```

{
  "siteSysop": "Operator Callsign",
  "siteSysopEmailAddress": "email@example.com",
  "siteGridSquare": "AB12cd",
  "siteCity": "City, Country",
  "siteInformation": "https://github.com/sv1btl/PhantomSDR-Plus",
  "siteHardware": "Hardware specs",
  "siteSoftware": "Software version",
  "siteReceiver": "SDR model",
  "siteAntenna": "Antenna description",
  "siteNote": "Additional notes",
  "siteIP": "http://domain.com:9002",
  "siteSDRBaseFrequency": 0,
  "siteSDRBandwidth": 2048000,
  "siteRegion": 1,
  "siteChatEnabled": true
}

```

Система сборки

Конфигурация сборки Meson

`meson.build` (корень)

Определяет:

- Метаданные проекта
- Зависимости
- Параметры компилятора
- Списки исходных файлов
- Цели сборки

`meson_options.txt`

Доступные параметры:

```
option('opencl', type: 'boolean', value: false, description: 'Enable OpenCL support')
option('cuda', type: 'boolean', value: false, description: 'Enable CUDA support')
option('optimization', type: 'string', value: '3', description: 'Optimization level')
```

Зависимости файлов

Зависимости сборки серверной части

```
spectrumserver binary depends on:
├─ C++ source files (src/**/*.cpp)
├─ External libraries:
│   ├── FFTW3
│   ├── WebSocket++
│   ├── FLAC
│   ├── Opus
│   ├── Liquid-DSP
│   ├── Boost
│   ├── zlib
│   ├── zstd
│   └─ OpenCL/CUDA (optional)
└─ Subproject headers:
    ├── nlohmann/json
    └─ toml11
```

Зависимости сборки веб-интерфейса

```
frontend/dist/ depends on:
├─ Source files (frontend/src/**)
├─ npm packages (node_modules/):
│   ├── Svelte
│   ├── Vite
│   ├── @wasm-audio-decoders/opus-ml
│   └─ ...
└─ Static assets (frontend/public/)
```

Поток данных

Порядок работы сервера

1. SDR Hardware → rtl_sdr/hackrf_transfer/etc.
↓
2. Sample Stream → stdin → spectrumserver
↓
3. spectrumserver:
 - FFT calculation (waterfall)
 - Demodulation (audio)
 - Compression (FLAC/Opus)
 ↓
4. WebSocket → Browser Client
↓
5. Browser:
 - Render waterfall
 - Decode and play audio
 - Display controls

Порядок взаимодействия с пользователем

1. User clicks on waterfall
↓
2. JavaScript sends frequency change request
↓
3. WebSocket → spectrumserver
↓
4. spectrumserver:
 - Updates demodulator frequency
 - Sends new audio stream
 ↓
5. Browser receives and plays new audio

Руководство по изменению файлов

Когда вы изменяете код серверной части (src/**):

```
cd PhantomSDR-Plus
meson compile -C build
# Server restart required
```

Когда вы изменяете код веб-интерфейса (frontend/src/**):

```
cd PhantomSDR-Plus/frontend
npm run build
cd ..
# Server restart required (for static files)
```

Когда вы изменяете конфигурацию (.toml, .json):

```
# Restart server
./stop-websdr.sh
./start-rtl.sh # (or appropriate start script)
```

Когда вы изменяете маркеры (markers.json):

```
# Reload page in browser
# No server restart needed
```

Важные пути

Пути времени выполнения

- **Конфигурация:** `./config-*.toml`
- **Корень HTML:** `./frontend/dist/`
- **Маркеры:** `./markers.json`
- **История чата:** `./chat_history.txt`
- **FFTW wisdom:** `./fftw_wisdom, ./phantom_fftw_wisdom`

Пути сборки

- **Готовый исполняемый файл:** `./build/spectrumserver`
- **Результат сборки веб-интерфейса:** `./frontend/dist/`
- **Модули Node:** `./frontend/node_modules/`

Пути исходного кода

- **Исходники серверной части:** `./src/`
- **Исходники веб-интерфейса:** `./frontend/src/`
- **Библиотеки ЦОС:** `./jsdsp/`

Типичные операции с файлами

Добавление новой конфигурации SDR

1. Скопируйте существующую конфигурацию: `cp config-rtl.toml config-mydevice.toml`
2. Отредактируйте параметры: `nano config-mydevice.toml`
3. Создайте скрипт запуска: `cp start-rtl.sh start-mydevice.sh`
4. Отредактируйте скрипт запуска: `nano start-mydevice.sh` — меняйте только блок **RECEIVER CONFIGURATION** в начале (`RX_LABEL`, `RX_COMM` — имя процесса приёмника, `RX_ARGS`, `CONFIG`, `FIFO`, а также хук `prestart`, если он нужен устройству). Логика запуска/сторожа/журналирования ниже универсальна и правок не требует.
5. Сделайте его исполняемым: `chmod +x start-mydevice.sh`

`stop-websdr.sh` уже останавливает любой `start-*.sh --watchdog`; если ваш приёмник использует имя процесса, отличное от `rx888_stream/rx_sdr/rtl_sdr/hackrf_transfer/cf32_to_real`, добавьте туда строку `killall -9 <имя>` и для него.

Настройка веб-интерфейса под себя

1. Измените исходный код: `nano frontend/src/App.svelte`
2. Пересоберите: `cd frontend && npm run build && cd ..`
3. Перезапустите сервер: `./stop-websdr.sh && ./start-rtl.sh`

Добавление собственных маркеров

1. Отредактируйте файл маркеров: `nano markers.json`
 2. Формат: `json { "markers": [ { "frequency": 145500000, "label": "2m Calling", "mode": "FM" } ] }`
 3. Перезагрузите страницу в браузере (перезапуск сервера не нужен)
-

Контроль версий

Файлы, отслеживаемые в Git

- Исходный код (`src/`, `frontend/src/`, `jsdsp/`)
- Примеры конфигурации (`config.example*.toml`)
- Система сборки (`meson.build`, `meson_options.txt`)
- Документация (`*.md`, `docs/`)
- Скрипты (`*.sh`)

Файлы, которые следует игнорировать (`.gitignore`)

- Результаты сборки (`build/`, `frontend/dist/`)
 - Зависимости (`frontend/node_modules/`)
 - Пользовательские данные (`chat_history.txt`)
 - Личные конфигурации (`config-rtl.toml`, если он изменён)
 - Двоичные данные (`*.o`, `*.so`)
-

Это описание структуры поможет вам ориентироваться в кодовой базе PhantomSDR-Plus и понимать её.

Инструкции по установке см. в [INSTALLATION.md](#). Сведения об использовании см. в [USER_GUIDE.md](#).

73 de SV1BTL & SV2AMK