

PhantomSDR-Plus

Documentación completa

16 documentos · generado 2026-10-03

Índice

1. README	8
Funciones principales	8
Hardware compatible	10
Inicio rápido	11
Pruebas de rendimiento	12
Uso	12
Configuración	13
Suelo de la cascada — waterfall.sh	14
Personalización	15
Documentación	17
Lista de WebSDR en línea	17
Resolución de problemas	17
Contribuir	18
Licencia	18
Autores y créditos	18
Enlaces	18
Soporte	18
Consejos de rendimiento	18
2. Guía de instalación	19
Índice	19
Requisitos del sistema	19
Preparación previa a la instalación	20
Instalación de dependencias	21
Instalación de Node.js y npm	22
Instalación de OpenCL (opcional pero recomendable)	23
Instalación de PhantomSDR-Plus	25
Instalación del códec de audio Opus	29
Autorun Spot Reporter (FT8/FT4/WSPR)	30
Emulación de clientes KiwiSDR (opcional)	32
Configuración	32
Configuración específica por dispositivo SDR	35
Pruebas y verificación	38
Configuración del arranque automático	39
Protección térmica de la CPU	41
Resolución de problemas	43
Optimización del rendimiento	47
Actualización de PhantomSDR-Plus	47
Copia de seguridad y restauración	50
Consideraciones de seguridad	51
Cómo obtener ayuda	51
Apéndice A: lista completa de dependencias	52
Apéndice B: ejemplos de configuración	52
3. Configuración del panel de administración	54
Descripción general	54
Qué le ofrece el panel de administración	54

Requisitos	55
Paso 1 — Archivos	55
Paso 2 — Ejecutar el script de configuración	55
Paso 3 — Iniciar / detener / reiniciar	56
Paso 4 — Abrir el puerto en el cortafuegos	58
Paso 5 — Función de expulsar usuarios	59
Configuración — admin_config.json	59
El ajuste sdr_host	60
proxy.py — qué es y cuándo lo necesita	60
Cambiar los puertos tras la configuración	61
Comandos manuales útiles	62
Archivos de registro	62
Resumen de acceso	64
Eliminación de mensajes del chat	64
Mensajes difundidos en la cascada	65
Gráficas del sistema	66
Thermal Guard	67
Spot Reporting (autorun FT8/FT4/JS8/WSPR)	70
Resolución de problemas	75
4. Thermal Guard	78
Contenido	78
1. Qué hace	78
2. Inicio rápido	79
3. Cómo decide	79
4. Los cuatro modos	80
5. Elegir el modo	82
6. Referencia de configuración	83
7. Funcionamiento sin el panel de administración	83
8. Activar la fase throttle sin root	84
9. Funcionamiento desatendido	85
10. Probarlo antes de confiar en él	86
11. Leer el registro	87
12. Lockout y límite de frecuencia	87
13. Resolución de problemas	88
14. Desactivarlo o eliminarlo	88
Véase también	88
5. Límites de conexión	90
Contenido	90
1. De qué se está defendiendo realmente	90
2. Las tres capas	91
3. Límites por dirección	91
4. Conexiones inactivas	92
5. La protección en el núcleo	93
6. Referencia de configuración	94
7. Qué ve un oyente rechazado	95
8. Leer los registros y los contadores	96
9. Elegir sus números	96
10. Trampas que conviene conocer	97
11. Desactivarlo todo	97

6. Varios receptores	98
Contenido	98
1. Cómo funciona	98
2. Dos maneras de publicar los receptores	99
3. Añadir un segundo receptor	99
4. receivers.toml	101
5. Arrancar y detener	101
6. Lo que ven los oyentes	102
7. El S-meter por encima de 30 MHz	102
8. Seguridad	103
9. Cuántos receptores caben	103
10. Trampas que conviene conocer	104
7. Servidor de estadísticas	105
Descripción general	105
Requisitos previos	105
Métodos de instalación	105
Configura tu aplicación PhantomSDR	108
Probar la instalación	110
Gestión del servicio	111
Instalación manual (Opción 2)	111
Cambiar el puerto después de la instalación	113
Solución de problemas	113
Lo que verás	116
Detección de temperatura	116
Resumen de funciones	117
Notas de seguridad	117
Soporte y recursos	117
8. Decodificadores	120
Índice	120
1. Cómo iniciar un decodificador	120
2. FT8	123
3. FT4-FT2	124
4. JS8	125
5. CW — código Morse	127
6. QRSS Grabber	128
7. WSPR	131
8. FAX de HF / WEFAX	132
9. NAVTEX	133
10. FSK / RTTY — incluidos PSK31 y Olivia	135
11. SSTV	138
12. Consejos generales	140
9. RADE README	142
La instalación — la vía rápida	142
Índice	142
Cómo funciona	143
Descripción de la arquitectura	144
RADEL frente a RADEU	144
Requisitos previos	145

Instalación a mano	145
Desplegar los archivos parcheados	145
Control del sidecar	146
El puerto 8074	147
Usar RADE en el navegador	147
Verificación y depuración	148
Variables de entorno	149
Archivos modificados	150
Resumen del flujo de señal	150
Resolución de problemas	151
Medir la concurrencia en su propio hardware	153
Actualizar RADE v1	155
10. RADE instalación general (Linux)	158
Paso 1 — Paquetes del sistema	158
Paso 2 — Paquetes de Python	159
Paso 3 — Clonar y compilar el repositorio radae	160
Paso 4 — Verificar la cadena de decodificación	161
Paso 5 — Compilar el frontend de PhantomSDR-Plus	162
Paso 6 — Control del sidecar	162
rade.sh sigue incluyéndose para uso autónomo: ejecutar el sidecar sin el lanzador de PhantomSDR-Plus, o probarlo por separado. Sus órdenes, su registro y las advertencias sobre el vigilante están en Control del sidecar.	162
Paso 7 — Abrir el puerto 8074	162
Paso 8 — Arrancar el servidor	163
Paso 9 — Arrancar RADE	164
Paso 10 — Usar RADE en el navegador	164
Nota sobre la CPU	164
Actualizar RADE en el futuro	165
11. Edición de variantes	166
1. Qué son las variantes	166
2. Qué hace cada archivo	167
3. Hacer un cambio que afecte solo a algunas variantes	169
4. Dónde están los conmutadores de variante	169
5. Otros ajustes que conviene conocer	170
6. Después de editar — reconstruir	171
7. Cómo comprobar su cambio antes de fiarse de él	171
12. Guía del usuario	173
Índice	173
Introducción	173
Primeros pasos	174
Descripción de la interfaz	175
Operaciones básicas	176
Funciones avanzadas	179
Modos de demodulación	182
Decodificadores digitales	185
Atajos de teclado	188
Marcadores	189
Uso en móviles	190
Consejos y buenas prácticas	191

Resolución de problemas	192
Preguntas frecuentes	194
Recursos	195
Apéndice: frecuencias habituales	195
Glosario	196
13. Emulación de cliente KiwiSDR	198
Índice	198
1. Qué obtiene un cliente Kiwi	198
2. Instalar el puente	199
3. Activarlo	199
4. Referencia de configuración	200
5. Conectar un cliente	200
6. Nivel de audio	200
7. El S-meter	201
8. Cascada y espectro	202
9. Si algo no funciona bien	203
10. Hasta dónde se ha verificado	204
14. Diversidad de recepción	205
Qué hace y qué no	205
Puesta en marcha rápida	206
Usar un WebSDR	207
Cómo leer el panel	208
Remote SNR trim	209
Elegir un segundo receptor	210
Resolución de problemas	210
Limitaciones	211
15. Control del transceptor	212
Qué necesita	212
Receptores compatibles	212
Qué hace y qué no hace	213
Inicio rápido	213
Cómo llegar al equipo	214
Ajustes del puerto serie	215
Sincronización	215
Modos	216
Ancho de filtro	217
Silencio al transmitir	217
Desplazamiento de frecuencia	217
El menú Rig	218
Linux	218
Windows	218
TCI-CAT en la página del receptor	218
Para operadores de receptores	228
Para desarrolladores: la interfaz de la página	228
Solución de problemas	229
Limitaciones conocidas	229
16. Estructura del proyecto	230
Índice	230

Árbol de directorios	231
Directorio raíz	244
Código fuente (src/)	247
Frontend (frontend/)	248
Listas de frecuencias (frequencylist/)	253
Archivos de configuración	253
Sistema de compilación	255
Dependencias entre archivos	255
Flujo de datos	256
Guía de modificación de archivos	256
Rutas importantes	257
Operaciones habituales con archivos	257
Control de versiones	258

CAPÍTULO 1

README

README.md

PhantomSDR-Plus

Un servidor WebSDR de código abierto mejorado, con funciones avanzadas y diseño futurista

PhantomSDR-Plus es un fork de PhantomSDR que ofrece un servidor web de radio definida por software (SDR) de alto rendimiento, capaz de atender a cientos de usuarios simultáneos. Incorpora una interfaz de usuario mejorada, compatibilidad con múltiples decodificadores, visualización del plan de bandas y compatibilidad con diversas plataformas de hardware SDR.

Funciones principales

Rendimiento y escalabilidad

- **Compatibilidad multiusuario:** cientos de usuarios simultáneos según el hardware
- **Alta tasa de muestreo:** admite SDR de hasta 70 MSPS (real) / 35 MSPS (IQ)
- **Aceleración por hardware:** compatibilidad con OpenCL y CUDA para procesamiento acelerado por GPU
- **Transmisión optimizada:** compresión de audio FLAC y Opus de baja latencia. El navegador decodifica cada paquete FLAC en el momento en que llega, y su búfer de reproducción se ajusta a la conexión del oyente, guardando solo el audio necesario: normalmente 100-150 ms del receptor al altavoz en un teléfono con Wi-Fi

Interfaz de usuario

- **Diseño futurista:** interfaz web moderna y adaptable
- **Optimizada para móviles:** interfaz móvil mejorada para escuchar en movimiento
- **Visualización del plan de bandas:** cascada interactiva con superposición de bandas de frecuencia
- **Mapas de color personalizables:** varios esquemas de color para la cascada
- **S-metro doble:** a elegir entre indicación de señal analógica o digital
- **Botones de decodificador de una pulsación:** una fila de decodificadores en el panel principal, justo debajo del selector de modos (FT8, FT4, FT2, JS8, CW, WSPR, FAX, SSTV, NAVTX, RTTY), con RADEL y RADEU junto al propio selector de modos y repetidos en las ventanas emergentes de modos y bandas. Una pulsación inicia el decodificador y abre su ventana; una segunda lo detiene. La antigua fila de ancho de banda se ha eliminado.
- **Escáner de canales:** recorre la banda, o exactamente lo que muestra la cascada, y se detiene en el primer canal que supere en los dB elegidos el propio suelo de ruido de la banda; continúa solo en cuanto ese canal queda en silencio. El paso sigue al modo y las paradas caen en la rejilla de canales (1 kHz SSB, 100 Hz CW, 5 kHz AM de onda corta, 9/10 kHz onda media, 9 kHz onda larga). Un modo *Saltar vacíos* lee el espectro de la cascada y salta directamente a las señales, y un botón de bloqueo saca del barrido un canal siempre ocupado. Todo es local al navegador del oyente, de modo que explorar no mueve el receptor para nadie más (véase la [Guía del usuario](#))
- **Diversidad de recepción:** combina el receptor con un segundo receptor situado en otro lugar y reproduce el que en cada momento tiene mejor señal, de modo que un desvanecimiento en uno lo cubre el otro. El segundo puede ser otro PhantomSDR-Plus, un KiwiSDR, un UberSDR o un WebSDR;

los tres primeros no necesitan nada más que el navegador del oyente, y solo un WebSDR requiere un pequeño relé en su propio servidor. La alineación tarda unos quince segundos, y un ajuste remoto de SNR equilibra ambas estaciones entre sí. Cada oyente guarda en su propio navegador una lista de segundos receptores con nombre, en el orden que prefiera, y puede exportarla a un archivo e importarla de nuevo (véase [Diversidad de recepción](#))

- **Varios receptores, una página:** un ordenador puede hacer funcionar dos o más receptores a la vez — por ejemplo un RX-888 para HF y un RTL-SDR para 2 m — y los oyentes pasan de uno a otro con botones en la cabecera de la página. Cada receptor conserva su propia cascada, chat, marcadores, lista de oyentes y datos de estación, y por encima de 30 MHz el S-meter indica unidades S de VHF ($S9 = -93$ dBm) (vea [Varios receptores](#))

Procesamiento de señal

- **Múltiples modos de demodulación:** AM, FM, USB, LSB, CW y más. También se ha implementado un decodificador RADE versión 1.
- **Detección de AM síncrona:** mejor calidad de recepción en AM
- **Reducción de ruido:** NR (espectral), NB (supresor de impulsos), NS (supresión del ruido de fondo) y AN (notch automático), todos en la escucha y nunca en los decodificadores
- **Opciones de AGC:** varios modos de control automático de ganancia
- **Squelch automático:** umbral de squelch automático basado en el ruido
- **Silenciador por subtono CTCSS:** el audio FM solo pasa mientras se recibe el subtono elegido — o cualquiera estándar; el tono se elige en una pequeña ventana junto al botón CTCSS, que luego lo muestra
- **Reducción de ruido por IA:** red neuronal RNNoise en el navegador del oyente, elimina el ruido de banda de la voz (solo modos de voz; véase la [Guía del usuario](#))

Funciones avanzadas

- **Decodificadores digitales:** FT8, FT4, FT2, JS8, CW, QRSS Grabber, WSPR, FAX de HF, SSTV, NAVTEX, FSK/RTTY, PSK31, Olivia y FreeDV RADE, cada uno ejecutándose en su propio hilo en segundo plano para que la decodificación nunca interrumpa el audio (véase [Decodificadores](#))
- **Decoder ID:** nombra el modo digital presente en el paso de banda y ofrece el decodificador adecuado con un clic, a partir del ancho de banda, la separación de tonos, la velocidad de símbolo y la temporización de las ráfagas medidos, teniendo en cuenta además la frecuencia como indicio adicional, que es también lo que distingue **FT8 de JS8**, dos modos de señal idéntica. Solo sugiere, nunca cambia por su cuenta, y prefiere callar a adivinar cuando la señal es demasiado débil. El desvanecimiento no lo silencia: cuando el QSB parte en trozos una transmisión WSPR de dos minutos, sigue nombrando el modo, leyendo la cadencia de todo el historial de escucha, y baja la confianza mostrada para indicar que la prueba es más débil. Entre unos 400 Hz y 2700 Hz la posición en el paso de banda es indiferente; fuera de ese margen aparece un botón ámbar **Recentre & retry**, que mueve el dial una vez e inicia una medición nueva. Cuando acepta una sugerencia, el receptor queda preparado para trabajar ese modo: la frecuencia sintonizada pasa al centro de la cascada, la vista queda en unos 100 kHz a su alrededor, y la banda lateral y el paso de banda pasan a ser los propios del modo — la subbanda de 3 kHz para la familia FT8, 1350–1650 Hz para WSPR, ± 250 Hz y CW para la telegrafía, etcétera. Desactivado por defecto, y en torno al 0,5 % de un núcleo mientras funciona (véase [Decodificadores](#))
- **Preajustes de modos digitales:** pulsar FT8, FT4, FT2, JS8 o WSPR ajusta a la vez la banda lateral y el paso de banda
- **Panel de conversación JS8:** los mensajes de varias tramas se reensamblan en frases completas
- **Panel de estadísticas:** estadísticas del servidor y de los usuarios en tiempo real

- **Lista de usuarios conectados:** todos los que están escuchando, con un botón de sintonía en cada fila. Su propia sesión aparece marcada **you**, tanto en la ventana de la interfaz como al abrir `users.html` como página independiente
- **Reporte automático de spots:** un demonio autorun decodifica FT8, FT4 y WSPR en el servidor y sube spots a PSK Reporter y WSPRnet, con dos contadores en el panel de administración: paneles por decodificador para la ejecución actual y un total histórico junto a cada casilla de banda/modo
- **Gráficas del servidor:** página **Gráficos** del panel de administración con la frecuencia de CPU, la carga, la temperatura y los usuarios conectados en los últimos 15 minutos a 24 horas (véase [Panel de administración](#))
- **Protección térmica:** un guardián en el panel de administración detiene el servidor si la CPU se sobrecalienta y lo reinicia cuando se enfría, con umbrales derivados del límite crítico de su propia CPU y sin suposiciones sobre cómo arranca o detiene usted el servidor; se entrega en modo de solo registro, así que no hace nada hasta que lo active (véase [Thermal Guard](#)) En un receptor desatendido se **recomienda encarecidamente** ejecutar el panel como unidad systemd: el guardián vive dentro de él, así que de lo contrario un reinicio se lleva la protección (véase [Panel de administración](#)).
- **Emulación de clientes KiwiSDR:** un puente opcional que responde también al protocolo KiwiSDR en el mismo host y puerto, de modo que software Kiwi como **AetherSDR** y `kiwiclient` se conecta directamente al receptor — con sintonía real, cascada y un S-metro en la misma escala que el de la página web. Apagado hasta que añada `[kiwi_emulation] enabled = true` a su configuración (véase [Instalación](#))
- **Control del transceptor (CAT):** mantiene su propio equipo y la ventana del receptor en la misma frecuencia, modo y ancho de filtro, en un sentido o en ambos — mediante el menú Rig de [Desktop PhantomSDR+](#) (Icom, Yaesu, Kenwood, Elecraft, FlexRadio y QRP Labs directamente, cualquier otro equipo mediante Hamlib, o flrig) o con la CATsync Tool for WebSDRs en un navegador. Las páginas ofrecen funciones `catsync_*` para frecuencia, modo, ancho de filtro y silencio. Funciona con receptores PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR y UberSDR (vea [Control del transceptor](#))
- **Sistema de marcadores:** importación/exportación de marcadores de frecuencia
- **Atajos de teclado:** navegación y control eficientes
- **Compatibilidad con la rueda del ratón:** sintonía de frecuencia intuitiva
- **Directorio WebSDR:** integración con <https://sdr-list.xyz>

Hardware compatible

PhantomSDR-Plus es compatible con una amplia gama de receptores SDR:

Dispositivo	Tasa de muestreo	Formato	Interfaz
RX888 MK2	Hasta 64 MHz	16 bits	Nativa
RTL-SDR	Hasta 3.2 MHz	8 bits	rtl_sdr
HackRF One	Hasta 20 MHz	8 bits	hackrf_transfer
Airspy HF+ / Discovery	Hasta 912 kHz	16 bits	SoapySDR (SoapyAirspyHF) + rx_sdr
SDRplay RSP1A	Hasta 10 MHz	16 bits	SoapySDR (libmirisdr-5 + SoapyMiri, o la API de SDRplay) + rx_sdr
RigExpert Fobos SDR	50 MHz (HF, muestreo directo) / hasta 20 MHz (RF)	16 bits / float de 32 bits	SoapySDR (SoapyFobosSDR) + rx_sdr
Otros dispositivos	Variable	Varios	SoapySDR/rx_tools

Inicio rápido

Requisitos del sistema

Mínimos:

- Ubuntu 22.04 LTS (recomendado) o Fedora
- CPU de 2 núcleos
- 4 GB de RAM
- 10 GB de espacio en disco

Recomendados:

- Ubuntu 24.04 LTS
- CPU de 4 o más núcleos (Ryzen 5 2600 o Intel i5-6500T o superior)
- 8 GB de RAM
- GPU compatible con OpenCL (opcional, pero muy recomendable)
- Almacenamiento SSD

Instalación

```
# Clonar el repositorio
git clone --recursive https://github.com/svlbtl/PhantomSDR-Plus
cd PhantomSDR-Plus

# Hacer ejecutables los scripts
chmod +x *.sh

# Ejecutar la instalación automática
./install.sh
```

Nota: después de ejecutar `install.sh`, reinicie el terminal antes de continuar.

¿Ya tiene PhantomSDR-Plus en marcha? No lo instale otra vez: actualícelo. Descargue la herramienta de actualización una vez y ejecútela; su configuración, marcadores, contraseña de administración, lista de frecuencias e historial de chat nunca se tocan, y lo que usted haya modificado se le pregunta en lugar de sobrescribirse:

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

Esa última línea sólo *informa* de lo que cambiaría y no escribe nada; `./update.sh --apply` lo hace. Los detalles están en el capítulo *Actualización de PhantomSDR-Plus*. Volver a pasar el instalador sobre una estación en marcha sólo hace falta si la reconstrucción falla por paquetes del sistema que faltan.

Use `install_fedora.sh` en Fedora, `install_arch.sh` en Arch o `install_opensuse.sh` en openSUSE Tumbleweed: hacen el mismo trabajo con otro gestor de paquetes. Todas las versiones de Debian y Ubuntu las cubre el propio `install.sh`.

No hay que instalar nada a mano de antemano. El script instala las dependencias de compilación y Node.js, compila el backend, compila el driver del receptor que elija (y con el RX888 también las reglas udev), abre `site_information.json` para que lo rellene, compila el frontend, instala OpenCL cuando el hardware lo admite, y después configura el **panel de administración**, el **decodificador FreeDV RADE** y el **servidor de estadísticas** — los tres por defecto, responda `n` para omitir alguno — y termina con un `recompile.sh` completo. Transcurre en 17 pasos numerados y se detiene en siete preguntas, cada una enmarcada por un recuadro «SE NECESITA SU RESPUESTA»; `PHANTOM_NONINTERACTIVE=1` las responde todas con sus valores por defecto. Cuente de veinte minutos a más de una hora.

Para la descripción paso a paso, véase [INSTALLATION.md → Qué hace el instalador](#). Para instrucciones de instalación detalladas, consulte [INSTALLATION.md](#).

Pruebas de rendimiento

Hardware	Tasa de muestreo	Uso de CPU	Capacidad de usuarios
Ryzen 5 2600 (todos los núcleos)	64 MHz (32 MHz IQ)	38-40 %	Más de 100 usuarios
AMD RX 580 (GPU)	64 MHz (32 MHz IQ)	28-35 %	Más de 100 usuarios
Intel i5-6500T (con OpenCL)	60 MHz (30 MHz IQ)	10-12 %	Más de 100 usuarios

Nota: la carga adicional de CPU por usuario es mínima (<1 % por usuario) cuando la aceleración por hardware está activada.

Uso

Funcionamiento básico

- Configure su SDR:** edite el archivo de configuración correspondiente (p. ej., `config-rtl.toml`)
- Actualice la información del sitio:** edite `frontend/site_information.json`
- Inicie el servidor:** ejecute el script de arranque adecuado: `bash ./start-rtl.sh` # Para RTL-SDR `./start-rspla.sh` # Para SDRplay RSP1A `./start-airspyhf.sh` # Para Airspy HF+ `./start-fobos-hf.sh` # Para RigExpert Fobos SDR, HF1/HF2 (0-25 MHz) `./start-fobos.sh` # Para RigExpert Fobos SDR, RF (25-6000 MHz) `./start-hackrf.sh` # Para HackRF One `./start-rx888mk2.sh` # Para RX888 MK2 El RSP1A, el Airspy HF+ y el Fobos funcionan a través de SoapySDR y `rx_sdr`; el instalador prepara su

driver cuando los elige (opciones 3, 6 y 5), o ejecute `./setup-rspla.sh`, `./setup-airspyhf.sh` o `./setup-fobos.sh` en una estación ya instalada. Vea [INSTALLATION.md](#). El HackRF One (opción 7) solo necesita el paquete `hackrf` de la distribución: `./setup-hackrf.sh`. Cada script de arranque es **autónomo**: detiene cualquier instancia en ejecución, lanza el receptor + `spectrumserver`, **pasa a segundo plano** (sobrevive al cierre del terminal) y ejecuta un **watchdog** que reinicia automáticamente la cadena si esta se cae. El progreso se registra en `logwebsdr.txt`: obsérvelo con `tail -f logwebsdr.txt`. Volver a ejecutar un script de arranque equivale a un reinicio limpio, y un bloqueo `flock` garantiza que solo funcione un receptor a la vez. Opcional: `SPECTRUM_CORES=0-3` para fijar CPU, `RX_ARGS=""` para sustituir los argumentos del receptor sin editar el archivo (`RX888_ARGS` para el RX888), `RX_DRIVER=miri|sdrplay` para elegir el driver del RSP1A.

4. **Acceda a la interfaz**: abra el navegador en `http://localhost:PORT` (el puerto predeterminado varía según la configuración)

Detener el servidor

Un único script de parada compartido sirve para **todos** los receptores: primero detiene el watchdog (para que no pueda reiniciar el sistema) y después el receptor y `spectrumserver`:

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

Configuración

Archivos de configuración esenciales

1. `config-[dispositivo].toml`: configuración del servidor y del SDR
2. Ajustes del servidor (puerto, hilos, raíz HTML)
3. Ajustes de entrada (tasa de muestreo, tamaño de FFT, frecuencia)
4. Opciones de registro en el directorio WebSDR
5. Ajustes de compresión de audio
6. Tras modificarlo, reinicie el servidor.
7. `frontend/site_information.json`: información pública del sitio
8. Datos del operador (indicativo, correo electrónico, ubicación)
9. Información sobre hardware y antena
10. Ajustes de región y ancho de banda
11. Activación/desactivación del chat
12. Tras modificarlo, ejecute `recompile.sh` en un terminal.
13. `markers.json`: marcadores de frecuencia y plan de bandas
14. `frontend/src/bands-config.js`: configuración del plan de bandas Este archivo se encuentra en `frontend/src/bands-config.js` y define las bandas que crea el SysOp.
15. Verá algo parecido a lo siguiente y puede editarlo a su gusto: ``
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 ``` Donde:
```

18. **ITU** es la región en la que se encuentra el servidor,
19. **min y max** son los límites de brillo de la cascada para la banda indicada,
20. **initFreq** es la frecuencia inicial a la que sintonizar cuando se selecciona la banda,
21. **publishBand** determina si la banda se muestra: 1 para bandas de radioaficionado, 2 para bandas de radiodifusión,
22. **startFreq y endFreq** definen los límites de la banda,
23. **stepi** es el paso predeterminado de la rueda del ratón para la banda, y **modes** es el modo predeterminado preferido para la banda.
24. Después de realizar cambios, **ejecute recompila.sh en un terminal** desde la carpeta PhantomSDR-Plus.

Para ver ejemplos de configuración, consulte los archivos de muestra incluidos:

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

Suelo de la cascada — `waterfall.sh`

`waterfall.sh` (en la raíz del repositorio) cambia el **nivel mínimo de cascada predeterminado** en dB, es decir, el valor desde el que arranca la cascada en una visita nueva.

Ese valor está en dos archivos fuente, en cinco lugares distintos (`frontend/src/waterfall.js` → `this.minWaterfall`, y `frontend/src/App.svelte` → el `let min_waterfall` inicial, los dos casos de reinicio «min» y el valor predeterminado de los marcadores). Editarlos a mano es fácil de equivocar, así que el script lo hace por usted. Nunca usa números de línea fijos: localiza cada punto por patrón, de modo que sigue funcionando después de actualizar las fuentes.

- Un valor **más alto** (p. ej. `-15`) da una cascada **más oscura**.
- Un valor **más bajo** (p. ej. `-45`) da una cascada **más brillante**.

```
./waterfall.sh          # interactivo: muestra los valores actuales y pide el nuevo
./waterfall.sh -v -15   # fija el valor sin preguntar
./waterfall.sh -v -15 -y # ... y omite todas las confirmaciones
./waterfall.sh -s       # solo muestra los valores actuales, sin cambiar nada
```

Antes de editar guarda una copia de seguridad de ambos archivos con marca de tiempo (`*.bak-AAAAmmdd-HHMMSS`) e imprime la orden para restaurarlos. Solo se tocan los literales iguales al valor en uso, de modo que otros números de esos archivos nunca pueden reescribirse por accidente. Si después el

nuevo valor no aparece, el script restaura las copias por sí mismo y sale con error. También avisa cuando los dos archivos están desincronizados y ajusta ambos.

- Tras el cambio hay que reconstruir el frontend: el script se ofrece a ejecutar `recompile.sh` por usted (con `-y` omite la reconstrucción, así que ejecute `./recompile.sh` usted mismo cuando esté listo).

Personalización

Cambiar la imagen de fondo

Sustituya `frontend/src/assets/background.jpg` por la imagen que prefiera (conservando el mismo nombre de archivo).

- Tras modificarla, ejecute `recompile.sh` en un terminal.

Selección del códec de audio

Elija entre FLAC y Opus en su configuración `.toml`:

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

- Tras modificarlo, reinicie el servidor.

Calibración del S-metro

Los dos instrumentos están deliberadamente en **cadena separadas**, por lo que se ajustan con dos mandos distintos.

Los dos mandos

Ambos están en la sección `[input]` del fichero de configuración con el que se arranca el servidor (`config.toml`, `config-rx888mk2.toml`, etc.):

```
[input]
smeter_offset=5          # solo la barra digital
analog_smeter_offset=5  # solo la aguja analógica
```

Se leen en `src/websocket.cpp` y se envían al navegador en la primera trama `basic_info`, así que **basta con reiniciar el servidor** — no hace falta recompilar el frontend.

Cadena analógica

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

- La ganancia visual de 1,1 se aplica *después* del offset, con pivote en -130 dBm. Así que 1 unidad de `analog_smeter_offset` desplaza la lectura 1,1 dB. Para desplazar la indicación X dB: `analog_smeter_offset ~= 0,91 x X`. El valor TOML se lee como entero, así que solo hay pasos enteros.
- La ley de la aguja (`powerFromDbm` en `frontend/src/lib/SMeterAnalog.svelte`) sitúa **S9 en -73 dBm**, en 60 de 100 unidades de aguja, con una curva `pow(...,0,6)` de -130 a -73 por debajo de S9 y una curva `pow(...,0,8)` de -73 a -13 (S9+60) por encima.
- **No** es una escala lineal de 6 dB por punto S, de modo que el punto de calibración con sentido es S9 — las marcas por debajo de S9 no seguirán a un generador en pasos de 6 dB.

Cadena digital

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

- Obsérvese el $/150 \times 100$: la escala digital está **comprimida a 2/3** y nunca ve ni `analog_smeter_offset` ni la ganancia visual de 1,1. Por eso la barra y la aguja nunca coinciden exactamente — es intencionado.
- 35 segmentos sobre 100 unidades significa que un segmento son 2,86 unidades de offset, unos 4,3 dB brutos. Es decir, `smeter_offset=3` equivale aproximadamente a un segmento.
- `DIGITAL_BAR_TRIM` en `frontend/src/lib/SMeterDigital.svelte` (ahora 0) desplaza la barra por segmentos enteros. Cambiarlo es tocar el código fuente y **sí** exige recompilar el frontend.

Las ventanas LED se alimentan de la cadena analógica en ambos instrumentos

Las ventanas dBm / dBμV / SNR / NF de **ambos** instrumentos leen el valor calibrado analógico y suman un `VISUAL_DBM_OFFSET` fijo de 5, escrito en el código de `SMeterAnalog.svelte` y `SMeterDigital.svelte`. Por tanto:

- `smeter_offset` **no** cambia ningún número de las ventanas, solo la longitud de la barra.
- $\text{dB}\mu\text{V} = \text{dBm} + 107$ (50 ohmios), y $\text{NF} = \text{dBm} - \text{SNR}$ por construcción, así que NF sigue automáticamente.
- Si se quieren los **números** correctos sin tocar la posición de la aguja, el mando es `VISUAL_DBM_OFFSET` — pero es una constante del código fuente, requiere recompilar y hay que cambiarla en **los dos** ficheros para mantener la coherencia.

Procedimiento

1. Inyectar un nivel conocido en la entrada de antena (generador a -73 dBm = S9), en SSB/CW con un ancho de banda estable. Las lecturas dependen del ancho de banda y del AGC, así que fija primero modo y ancho.
2. Anotar lo que muestra la ventana dBm analógica y poner `analog_smeter_offset ~= 0,91 x (-73 - valor mostrado)`. Reiniciar el servidor y volver a comprobar. Con una iteración basta; la aguja debe quedar en S9.
3. Pasar al instrumento digital y ajustar `smeter_offset` para que la longitud de la barra sitúe S9 donde se desee — unas 3 unidades por segmento. Este paso es puramente estético y no afecta a las lecturas.
4. Sin generador, una buena alternativa es una banda tranquila con una baliza de nivel conocido, o simplemente anclar el suelo de ruido a un valor plausible (p. ej. -120 dBm en 20 m con una antena decente) — asumiendo que así se calibra toda la cadena, incluida la ganancia de antena, y no solo el receptor.

Variantes de la interfaz gráfica

Las cuatro variantes de la interfaz (S-meter analógico/digital x disposición v1/v2) viven en una sola construcción. Los visitantes cambian entre ellas con el menú ☙ de la esquina superior derecha — no se recarga nada, así que el audio, la cascada y cualquier decodificador en marcha continúan — y la elección se recuerda en cada navegador.

- Ejecute `recompile.sh` en un terminal para fijar la variante *inicial*, la que ve quien visita el sitio por primera vez.
- **EDITING_VARIANTS.md**: cómo editar las variantes del frontend (S-meter y disposición) y reconstruirlas

Documentación

- **INSTALLATION.md**: guía de instalación completa para operadores de sistema
 - **ADMIN_PANEL_SETUP.md**: guía de instalación para administradores del sistema
 - **USER_GUIDE.md**: guía para el usuario final sobre el manejo del WebSDR
 - **THERMAL_GUARD.md** - Manual del sysop para la protección contra sobrecalentamiento de la CPU: los cuatro modos y qué hacer con cada uno
 - **CONNECTION_LIMITS.md** - Manual del sysop para los límites de conexión: proteger un receptor público frente a avalanchas de conexiones
 - **MULTI_RECEIVER.md** - Manual del sysop para varios receptores en un ordenador: instancias, `receivers.toml`, el selector de receptor y `start-all.sh`
 - **PROJECT_STRUCTURE.md**: estructura de directorios y organización del código
 - **EDITING_VARIANTS.md**: cómo editar las variantes del frontend (S-meter y disposición) y reconstruirlas
-

Lista de WebSDR en línea

Registre su WebSDR en el directorio oficial: <https://sdr-list.xyz>

Establezca `register_online=true` en su archivo de configuración `.toml` para registrarlo automáticamente.

Resolución de problemas

Problemas frecuentes

OpenCL no funciona

- Asegúrese de que los controladores estén correctamente instalados
- Compruébelo con el comando `clinfo`
- Consulte `INSTALLATION.md` para la configuración detallada de OpenCL

Problemas de latencia de audio

- Mantenga FLAC como códec: es el de menor latencia, mientras que Opus retiene hasta 20 ms por paquete
- Abra la página añadiendo `?diag=1` a su dirección (por ejemplo `/mobile?diag=1`) para ver el retardo dividido en paquete, búfer y salida; el búfer se reduce solo en uno o dos minutos tras un problema de red
- Si el audio se corta, suba el búfer un paso (vea la Guía del usuario, *Búfer de audio y retardo*)
- Asegúrese de disponer de recursos de CPU/GPU suficientes

Fallos de compilación

- Compruebe que todas las dependencias estén instaladas
 - Pruebe a limpiar el directorio de compilación: `rm -rf build && meson setup build`
 - Busque versiones de bibliotecas en conflicto
-

Contribuir

¡Las contribuciones son bienvenidas! Este es un fork independiente con funciones adicionales. Por favor:

1. Haga un fork del repositorio
 2. Cree una rama para la nueva función
 3. Confirme sus cambios
 4. Envíe la rama al repositorio remoto
 5. Cree un Pull Request
-

Licencia

Este proyecto se distribuye bajo la Licencia Pública General de GNU v3.0; consulte el archivo [LICENSE](#) para más detalles.

Autores y créditos

- **SV1BTL y SV2AMK:** desarrollo y mejoras de PhantomSDR-Plus
 - Basado en el proyecto original PhantomSDR
-

Enlaces

- **Demostración en directo:** <http://phantomsdr.no-ip.org:8900/>
 - **Directorio WebSDR:** <https://sdr-list.xyz>
 - **Repositorio de GitHub:** <https://github.com/sv1btl/PhantomSDR-Plus>
-

Soporte

- **Incidencias:** informe de errores a través de [GitHub Issues](#)
 - **Correo electrónico:** contacto sv1btl@otenet.gr
-

Consejos de rendimiento

1. **Active OpenCL/CUDA** para la aceleración por GPU: reduce drásticamente el uso de CPU
 2. **Use almacenamiento SSD** para un mejor rendimiento de E/S
 3. **Asigne suficiente RAM:** se recomiendan al menos 8 GB
 4. **Optimice el tamaño de la FFT:** una FFT mayor = mejor resolución, pero más uso de CPU
 5. **Considere una GPU dedicada:** AMD o NVIDIA con compatibilidad OpenCL
-

73 de SV1BTL & SV2AMK

Para instrucciones de configuración detalladas, consulte *INSTALLATION.md* Para la guía de manejo del usuario final, consulte *USER_GUIDE.md*

CAPÍTULO 2

Guía de instalación

INSTALLATION.md

Guía de instalación de PhantomSDR-Plus para operadores de sistema

Esta guía completa le acompañará en la instalación y configuración de PhantomSDR-Plus en su servidor.

Índice

1. Requisitos del sistema
2. Preparación previa a la instalación
3. Instalación de PhantomSDR-Plus — el script y qué hace
4. Autorun Spot Reporter (FT8/FT4/WSPR)
5. Emulación de clientes KiwiSDR (opcional)
6. Configuración
7. Configuración específica por dispositivo SDR
8. Pruebas y verificación
9. Configuración del arranque automático
10. Protección térmica de la CPU
11. Resolución de problemas

Solo como referencia: el instalador ya hace todo esto por usted. Lea estas secciones si usa una distribución que ninguno de los scripts cubre, o si necesita reparar un paso a mano:

- Instalación de dependencias
 - Instalación de Node.js y npm
 - Instalación de OpenCL
 - Compilar a mano
 - Instalación del códec de audio Opus
-

Requisitos del sistema

Sistemas operativos compatibles

Principal (recomendado):

- Ubuntu 22.04 LTS (Jammy), 24.04 LTS (Noble) y 26.04 LTS (Resolute) — se recomienda 24.04
- Debian 12 (Bookworm) y Debian 13 (Trixie)

Alternativa:

- Fedora (última versión estable)
- Arch Linux (rolling)
- openSUSE Tumbleweed (no Leap — vea la nota más abajo)

Requisitos de hardware

Configuración mínima:

- CPU: procesador de doble núcleo (2 GHz o más)
- RAM: 4 GB
- Almacenamiento: 10 GB libres
- Red: conexión de 100 Mbps

Configuración recomendada:

- CPU: cuatro núcleos o mejor (Ryzen 5 2600, Intel i5-6500T o superior)
- RAM: 8 GB o más
- Almacenamiento: SSD de 20 GB o más
- GPU: AMD/NVIDIA compatible con OpenCL (muy recomendable)
- Red: conexión de 1 Gbps

Configuración de alto rendimiento:

- CPU: 6 núcleos o más (Ryzen 7, Intel i7 o mejor)
- RAM: 16 GB o más
- Almacenamiento: SSD NVMe
- GPU: GPU dedicada compatible con OpenCL/CUDA
- Red: 1 Gbps o superior

Preparación previa a la instalación

1. Actualice el sistema

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

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

2. Compruebe la versión de Ubuntu

```
lsb_release -a
```

La salida esperada debería mostrar: Ubuntu 24.04 LTS

3. Compruebe el espacio libre en disco

```
df -h
```

Asegúrese de tener al menos 10 GB libres en su directorio personal.

4. Consulte la información de la CPU

```
lscpu
```

Anote el número de núcleos/hilos para optimizar la configuración.

Instalación de dependencias

IMPORTANTE

No necesita esta sección para una instalación normal. `./install.sh` — o el instalador de su distribución — lo hace todo por usted; véase [Qué hace el instalador](#). Lo que sigue es una referencia para una instalación manual, para una distribución que ningún script cubre, o para reparar una pieza a mano.

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
```

Verifique la instalación

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

Instalación de Node.js y npm

IMPORTANTE

No necesita esta sección para una instalación normal. `./install.sh` — o el instalador de su distribución — lo hace todo por usted; véase [Qué hace el instalador](#). Lo que sigue es una referencia para una instalación manual, para una distribución que ningún script cubre, o para reparar una pieza a mano.

PhantomSDR-Plus necesita Node.js para compilar la interfaz web. Usaremos NVM (Node Version Manager) para instalarlo.

1. Instale NVM

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

2. Cargue NVM

IMPORTANTE: cierre y vuelva a abrir el terminal, o ejecute:

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

3. Compruebe la instalación de NVM

```
nvm --version
```

Salida esperada: 0.40.4 o similar

4. Instale 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. Opcional: instale otras versiones de 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
```

Instalación de OpenCL (opcional pero recomendable)

IMPORTANTE

No necesita esta sección para una instalación normal. `./install.sh` — o el instalador de su distribución — lo hace todo por usted; véase [Qué hace el instalador](#). Lo que sigue es una referencia para una instalación manual, para una distribución que ningún script cubre, o para reparar una pieza a mano.

OpenCL mejora enormemente el rendimiento al trasladar los cálculos de FFT a la GPU. Esta sección cubre la gráfica integrada de Intel. Para GPU de AMD/NVIDIA, consulte la documentación del fabricante.

CPU Intel con gráficos integrados

1. Instale los componentes básicos de 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. Descargue 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. Instale el entorno de ejecución OpenCL de Intel

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

Si aparecen errores de dependencias:

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

4. Instale el cargador ICD de OpenCL

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

Si aparecen errores:

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

5. Verifique la instalación de OpenCL

```
sudo clinfo
```

Debería ver información sobre su plataforma y dispositivos OpenCL. Busque:

- Número de plataformas: 1 (o más)
- Nombre de la plataforma: Intel(R) OpenCL (o similar)
- Tipo de dispositivo: GPU o CPU

6. Reinicie

```
sudo reboot
```

OpenCL en GPU AMD

Para GPU de AMD, instale 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 en GPU NVIDIA

Para GPU de NVIDIA, instale 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
```

Instalación de PhantomSDR-Plus

Clonar el repositorio, dar permisos a los scripts y ejecutar el instalador

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

⚠ **IMPORTANTE:** Cuando termine el instalador, **reinicie el terminal**: hasta entonces Node.js y Rust recién instalados no están en su PATH.

¿Ya tiene PhantomSDR-Plus en marcha? No lo instale otra vez: actualícelo. Descargue la herramienta de actualización una vez y ejecútela; su configuración, marcadores, contraseña de administración, lista de frecuencias e historial de chat nunca se tocan, y lo que usted haya modificado se le pregunta en lugar de sobrescribirse:

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

Esa última línea sólo *informa* de lo que cambiaría y no escribe nada; `./update.sh --apply` lo hace. Los detalles están en el capítulo *Actualización de PhantomSDR-Plus*. Volver a pasar el instalador sobre una estación en marcha sólo hace falta si la reconstrucción falla por paquetes del sistema que faltan.

Use el script que corresponda a su sistema. Todos hacen el mismo trabajo y hacen las mismas preguntas; solo cambia el gestor de paquetes:

Sistema	Script
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>

Un único instalador para todas las versiones de Debian y Ubuntu. `install.sh` lee `/etc/os-release` y la versión de Boost instalada y se adapta solo; los antiguos `install_ubuntu22.sh`, `install-Deb12.sh` e `install_ubuntu26.sh` han desaparecido. Establece `DEBIAN_FRONTEND=noninteractive`, para que el `tzdata` que llega con `python3-matplotlib` no pueda detener la ejecución preguntando por su zona horaria y tragarse después la respuesta destinada a la siguiente pregunta. En Jammy llama directamente a `install_rade_ubuntu22.sh`, porque el `python3-websockets` de Jammy es 10.1 y RADE necesita 11.0 o posterior. Sabe dónde vive `intel-openssl-icd` en cada versión: en los repositorios en 22.04 y 26.04, en `non-free` en Debian 12, y en el propio repositorio gráfico de Intel en 24.04 y Debian 13. Y donde Boost es 1.87 o posterior, el parche de cabeceras de `websocketpp` deja de ser opcional: el instalador verifica que se aplicó y se niega a compilar sin él. Nunca hace falta un compilador más nuevo: el GCC de serie acepta `-std=c++23` en las cinco versiones, así que no instale `gcc-12` para esto.

La instalación transcurre en 17 pasos claramente numerados, y cada punto en el que se le espera va enmarcado por un recuadro **SE NECESITA SU RESPUESTA**, de modo que una pregunta no pueda confundirse con la salida que va pasando. Las siete preguntas se enumeran al principio, antes de instalar nada. `PHANTOM_NONINTERACTIVE=1` las responde todas con sus valores por defecto; consulte la cabecera de `install.sh` para las variables `PHANTOM_*`.

Cada ejecución escribe `install.txt`. Cuando el instalador termina —o muere a medio camino— escribe un informe en `install.txt` dentro del directorio `PhantomSDR-Plus`: el resultado, cada uno de los 17 pasos como OK / SKIPPED / PARTIAL / FAILED, lo que detectó (distribución, Boost, compilador, Node.js), qué componentes se instalaron, y cada aviso que apareció. Una ejecución fallida deja un informe que termina en el paso que falló, con el motivo y la nota de que el instalador puede volver a ejecutarse sin riesgo. Es el primer archivo que hay que leer cuando algo no funcionó, y el primero que adjuntar a un informe de error. Cada ejecución lo sobrescribe, así que guarde una copia si quiere comparar dos instalaciones.

Ubuntu 26.04, Arch y openSUSE Tumbleweed compilan, pero no se han puesto en el aire.

Los tres traen un Boost más nuevo que 1.87, que eliminó la API `io_service` para la que se escribió el `websocketpp` 0.8.2 incluido. Las cabeceras parcheadas que copia el instalador salvan esa distancia (`io_context`, `executor_work_guard`, `boost::asio::post`, el resolver moderno), y donde Boost es 1.87 o posterior el instalador trata ese parche como obligatorio y no opcional: verifica que las copias se aplicaron y se niega a compilar sin ellas. Una instalación completa, con el driver del receptor y todos los componentes opcionales, se ha verificado de principio a fin en contenedores con Boost 1.90 (Ubuntu 26.04), 1.91 (openSUSE Tumbleweed) y 1.92 (Arch). Eso demuestra que el servidor compila y arranca, no que sirva un receptor durante horas. Considere los tres sin probar en producción hasta que alguien informe.

openSUSE aquí significa Tumbleweed. Ahí es donde está verificado `install_opensuse.sh`. Leap 15.6 no funciona: sus repositorios no incluyen `liquid-dsp-devel`, que el backend necesita, y Boost solo está disponible con nombres de paquete versionados. Dar soporte a Leap exigiría añadir repositorios OBS de terceros, así que queda fuera del alcance por ahora.

Qué hace el instalador

No hay que preparar nada a mano de antemano: ninguna lista de paquetes que pegar, ningún Node.js que descargar, ningún paquete de OpenCL que buscar. Transcurre en 19 pasos numerados y se detiene a hacerle hasta diez preguntas, cada una enmarcada por un recuadro «SE NECESITA SU RESPUESTA»: quédese al teclado, o bien ponga `PHANTOM_NONINTERACTIVE=1` y deje que responda todo con sus valores por defecto (véase más abajo), y cuente entre unos veinte minutos y bastante más de una hora según la máquina y según cuántos extras conserve.

#	Paso	Qué se le pregunta
1	Enumera los servicios de PhantomSDR-Plus que están en marcha en ese momento — panel de administración, proxy inverso, servidor de estadísticas, receptor — y ofrece detenerlos antes de tocar nada. Solo cuenta como receptor un script de arranque en ejecución; uno que solo está abierto en un editor no se toca.	confirmar, sí por defecto
2-6	Detecta la distribución e instala todas las dependencias de compilación (compilador, meson/ninja, FFTW, Boost, FLAC, Opus, liquid-dsp, zlib/zstd, libcurl ...), instalando Node.js 22 mediante nvm si el sistema no lo tiene o lo tiene demasiado antiguo	nada
7	Compila el backend con meson	nada
8	Compila el driver de su receptor — RX888 MkII / RX888, RTL-SDR (el Blog V4 se pregunta aparte), SDRplay RSP1A, RigExpert Fobos SDR, Airspy HF+, HackRF One o ninguno. Al elegir el RX888 también instala las reglas udev , para que el servidor nunca necesite sudo para el dispositivo. El RSP1A, el Fobos y el Airspy HF+ funcionan a través de SoapysDR: su opción ejecuta setup-rsp1a.sh, setup-fobos.sh o setup-airspyhf.sh, que compilan el driver y rx_sdr e instalan también una regla udev (vea Receptores sobre SoapysDR (RSP1A, Fobos, Airspy HF+)) El HackRF One no necesita SoapysDR: su opción ejecuta setup-hackrf.sh, que instala el paquete hackrf de la distribución y una regla udev.	qué SDR tiene
9	Abre frontend/site_information.json en su editor	indicativo, locator, hardware, antena — no se lo salte
10-11	Instala las dependencias del frontend y compila las páginas de escritorio y /mobile	nada
12	Instala OpenCL eligiendo el proveedor según el hardware que encuentra (GPU Intel / AMD / NVIDIA, o el runtime de CPU x86). Si no hay ningún dispositivo compatible, lo dice y continúa	confirmar, sí por defecto
13-15	Instala el panel de administración , el decodificador FreeDV RADE V1 y el servidor de estadísticas — los tres por defecto	confirmar cada uno, sí por defecto; cada uno tiene sus preguntas
16	Vuelve a aplicar las cinco cabeceras parcheadas de websocketpp sobre el subproyecto de meson y verifica que se copiaron. Tres de ellas son el trabajo de compatibilidad con Boost ≥ 1.87, sin el cual el backend no compila con Boost 1.90; las otras dos son cambios propios del proyecto, uno de ellos la corrección que necesita el registro en websdr.org	nada
17	Instala la emulación de clientes KiwiSDR ejecutando kiwi_install.sh, para que clientes Kiwi como AetherSDR puedan conectarse a este receptor. Parchea las fuentes y añade [kiwi_emulation] a los ficheros de configuración de la raíz del repositorio — véase Emulación de clientes KiwiSDR	confirmar, sí por defecto
18	Ejecuta recompile.sh, para que todo se compile a partir de las fuentes parcheadas	[3] Both backend and frontend → variante inicial → [1] build-all.sh
19	Imprime el resumen: cada paso con su veredicto y cada componente con lo que se instaló	nada

Instalación desatendida

Cada pregunta tiene una variable de entorno que la sustituye, y el instalador pasa por sí solo a los valores por defecto cuando la entrada estándar no es un terminal (una tubería, un contenedor, un trabajo de CI). Funciona igual en los cuatro: `install.sh`, `install_arch.sh`, `install_fedora.sh` e `install_opensuse.sh`. Ponga `PHANTOM_NONINTERACTIVE=1` y la ejecución termina sin preguntar nada:

Variable	Efecto
<code>PHANTOM_NONINTERACTIVE=1</code>	answer every question with its default
<code>PHANTOM_SDR=1...7</code>	RX888 · RTL-SDR · SDRplay RSP1A · skip · Fobos · Airspy 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>	continuar sin curlpp — solo Arch y openSUSE (por defecto y)
<code>PHANTOM_FIX_CLOCK_SKEW=y\ n</code>	restablecer las marcas de tiempo de los fuentes fechadas en el futuro, para que meson pueda compilar (por defecto y)

Los tres sub-instaladores marcados *n desatendido* son guiones interactivos por su cuenta, así que una ejecución desatendida los omite en lugar de quedarse esperando. Nómbralos explícitamente para incluirlos:

```
# 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
```

Tras reiniciar, las reglas udev del RX888 y cualquier driver de OpenCL quedan plenamente activos; el instalador le avisa cuando hace falta reiniciar o volver a iniciar sesión.

Al configurar el panel de administración, el instalador ofrece también las dos unidades `systemd` y **recomienda encarecidamente** aceptarlas: la protección contra sobrecalentamiento se ejecuta dentro del panel, así que sin ellas un reinicio o un fallo deja la máquina desprotegida. El paso se omite automáticamente donde `systemd` no está en marcha (contenedores, WSL1, OpenRC) — véase la [guía del panel](#).

Cada extra es un script normal que también puede ejecutar por separado en cualquier momento — `./setup_admin.sh` ([manual](#)), `./install_rade.sh` ([manual](#)), `./install-stats-server.sh` ([manual](#)), `./setup-rx888-udev.sh` y los scripts de los receptores `./setup-rsp1a.sh`, `./setup-fobos.sh`, `./setup-airspyhf.sh` y `./setup-hackrf.sh` — son exactamente los que llama el instalador.

Compilar a mano (referencia)

IMPORTANTE

No necesita esta sección para una instalación normal. `./install.sh` — o el instalador de su distribución — lo hace todo por usted; véase [Qué hace el instalador](#). Lo que sigue es una referencia para una instalación manual, para una distribución que ningún script cubre, o para reparar una pieza a mano.

Por si alguna vez tiene que compilar sin el instalador — una distribución que ningún script cubre, o reparar una compilación a medias:

Compilar el 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
```

Compilar el frontend

```
cd frontend

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

# Return to project root
cd ..
```

Verifique la instalación

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

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

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

Instalación del códec de audio Opus

IMPORTANTE

No necesita esta sección para una instalación normal. `./install.sh` — o el instalador de su distribución — lo hace todo por usted; véase [Qué hace el instalador](#). Lo que sigue es una referencia para una instalación manual, para una distribución que ningún script cubre, o para reparar una pieza a mano.

Opus ofrece mejor calidad de audio y menor latencia que FLAC.

1. Instale la biblioteca de sistema libopus

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

2. Instale el decodificador Opus para el frontend

```
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. Verifique la instalación de 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)

PhantomSDR-Plus incluye un **autorun spot reporter** opcional (en el directorio `autorun/`). Decodifica FT8/FT4/WSPR en el servidor, directamente del receptor, y sube los spots a las redes de reporte:

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

Se controla íntegramente desde la pestaña **panel de administración → «Spot Reporting»**, y el **reporte está DESACTIVADO por defecto**: no se transmite ni se sube nada hasta que lo active ahí. El decodificador se ejecuta como un demonio Node.js aparte y toma el audio del receptor localmente, por lo que no añade hardware de RF adicional.

Requisitos

Requisito	Notas
Node.js 22+	El mismo entorno de ejecución que ya necesita la compilación del frontend; se instala en el paso de Node.js .
Paquetes npm ws + cbor-x	Se resuelven a través de <code>autorun/node_modules</code> , un enlace simbólico al <code>node_modules</code> del frontend (ambos paquetes están declarados en <code>frontend/package.json</code>).
util-linux (taskset)	El botón «Start» del panel fija el demonio a los E-cores con <code>taskset</code> . Está presente en prácticamente todas las distribuciones; los instaladores lo añaden de forma explícita.
Indicativo + localizador	Se leen de <code>frontend/site_information.json</code> (<code>siteSysop</code> / <code>siteGridSquare</code>) salvo que se sustituyan en la pestaña de administración. Los spots se suben con ese indicativo.

⚠ **Reporte solo lo que realmente recibe.** Los spots se suben a redes públicas con su indicativo: active únicamente las bandas y modos que su receptor oye de verdad, y use su localizador correcto.

Instalación automática

`install.sh` (y las variantes por distribución: `install_arch.sh`, `install_fedora.sh`, `install_opensuse.sh`) se encargan de todo: instalan Node.js 22 y `util-linux`, ejecutan el `npm install` del frontend y después crean automáticamente el enlace simbólico `autorun/node_modules`. No hacen falta pasos adicionales: la función está lista en cuanto termina `install.sh`.

Instalación manual

NOTA

El instalador ya hace esto por usted: crea el enlace simbólico e instala `util-linux`. Use los pasos siguientes solo para reparar una instalación a mano.

Si instaló manualmente (opción B), cree usted mismo el enlace simbólico tras el `npm install` del frontend para que el demonio pueda resolver sus dependencias:

```
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
```

Nota: `autorun/node_modules` está excluido por git, así que un `git clone` nuevo nunca lo contiene: el enlace debe (re)crearse después de cada descarga limpia. Los instaladores lo hacen por usted; el comando anterior es solo para instalaciones manuales.

Comprobación

```
# 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
```

Después abra el panel de administración, vaya a la pestaña **Spot Reporting**, indique su identidad, marque las bandas y modos que desea decodificar, active los destinos y pulse **Start**. Aparecerá un distintivo «REPORTING» en la cascada principal mientras el reporte esté activo. (PSK Reporter sube por lotes cada 5 minutos y wsprnet cada 2 minutos, así que un demonio recién iniciado muestra «0 sent» durante los primeros minutos: es normal.)

A partir de ahí se muestran dos contadores distintos, fáciles de confundir. Los paneles **SPOTS UPLOADED PER DECODER** cuentan solo la ejecución actual, de modo que Stop/Start los pone a cero; el número junto a cada casilla de banda/modo es el total **histórico** de esa ranura, guardado en `autorun-totals.json` para que sobreviva a los reinicios. Consulte la [guía del panel de administración](#) para ambos,

y para la página **Gráficos**, que representa la frecuencia de CPU, la carga, la temperatura y los usuarios conectados desde 15 minutos hasta 24 horas.

Si su máquina se calienta, el panel incluye además una **protección térmica** que detiene el servidor cuando la CPU alcanza una temperatura peligrosa y lo reinicia una vez fría. Deriva sus umbrales del límite crítico de su propia CPU, así que no hay nada que calcular, y funciona con cualquier método de arranque/parada. Se instala con el panel pero arranca en modo de solo registro: anota lo que *habría* hecho y no cambia nada hasta que usted la active en Ajustes. La decodificación continua mantiene la CPU ocupada las 24 horas, así que conviene leer la pestaña CRASH tras una semana de autorun para ver hasta dónde llega realmente su máquina.

El puerto del servidor se detecta automáticamente. El demonio se conecta directamente al `[server] port` del propio spectrumserver (bucle local + token, sin pasar por el proxy). Lee ese puerto del archivo de configuración con el que se lanzó el spectrumserver en ejecución, de modo que funciona en cualquier puerto sin configuración; la línea `[autorun] tap backend: ...` de `autorun.log` muestra lo que ha resuelto. Si su servidor no estaba en marcha cuando arrancó el demonio, o si tiene una instalación poco habitual, fíjelo en `autorun.json: json "server": { "host": "127.0.0.1", "port": 9002 }` Un puerto incorrecto se manifiesta como `504 / tap closed 1006` en `autorun.log`, con `decodes` atascado en 0.

Emulación de clientes KiwiSDR (opcional)

Desde la v4.1.0 PhantomSDR-Plus también puede responder al **protocolo KiwiSDR**, de modo que el software escrito para un KiwiSDR — **AetherSDR**, `kiwiclient` y los demás — se conecta directamente a su receptor, en el mismo host y puerto que ya publica. Está apagado hasta que se añade `[kiwi_emulation] enabled = true` a la configuración con la que arranca su receptor.

El instalador lo ofrece como paso 17; `./kiwi_install.sh` lo aplica a un árbol ya instalado.

Documentación completa: **Emulación de clientes KiwiSDR** — qué hace el puente, cómo instalarlo, cada clave de `[kiwi_emulation]`, conectar un cliente, el nivel de audio, el S-meter, la tasa de la cascada y una tabla de síntomas.

Configuración

1. Elija su archivo de configuración

Seleccione el archivo de configuración adecuado para su SDR:

- `config-rtl.toml`: llaves RTL-SDR
- `config-rsp1a.toml`: SDRplay RSP1A
- `config-airspyhf.toml`: Airspy HF+ Discovery
- `config-fobos-hf.toml`: RigExpert Fobos SDR, HF1/HF2 en muestreo directo (0-25 MHz)
- `config-fobos.toml`: RigExpert Fobos SDR, rama RF (25-6000 MHz)
- `config-rx888mk2.toml`: RX888 MK2
- `config-hackrf.toml`: HackRF One
- `config.example.hackrf.toml`: HackRF One

En este ejemplo usaremos el RTL-SDR.

2. Edite el archivo de configuración

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

Ajustes clave

```
[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
```

Orientación sobre la tasa de muestreo

Cobertura	Tasa de muestreo	Tamaño de FFT
2 MHz	2048000	131072
3.2 MHz	3200000	131072
10 MHz	10000000	1048576
30 MHz	30000000	2097152
60 MHz	60000000	4194304

3. Configure la información del sitio

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

Edite los siguientes campos:

```
{
  "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
}
```

Regiones de la IARU:

- **1:** Europa, África, Oriente Medio, norte de Asia
- **2:** América (Norte, Centro y Sur), Caribe
- **3:** Asia-Pacífico, Oceanía

4. Personalice los marcadores de frecuencia (opcional)

```
nano markers.json
```

Añada sus frecuencias favoritas, repetidores y emisoras de radiodifusión.

5. Edite el script de arranque

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

El script de arranque es un lanzador autónomo con watchdog integrado. Edite únicamente el bloque **RECEIVER CONFIGURATION** cerca del principio, para que los argumentos del receptor coincidan con su instalación:

```
# == 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)
```

Parámetros dentro de RX_ARGS:

- **-f 145000000:** frecuencia central (145 MHz)
- **-s 2048000:** tasa de muestreo (2,048 MSPS)

CONFIG apunta a su `.toml`. **No** necesita editar nada más en el script: la lógica de arranque/reinicio/watchdog/registro es genérica. También puede sustituir los argumentos del receptor al lanzarlo sin editar el archivo: `RX_ARGS="-f 145000000 -s 2048000 -" ./start-rtl.sh` (RX888_ARGS para `start-rx888mk2.sh`).

Configuración específica por dispositivo SDR

RTL-SDR

Instale las herramientas RTL-SDR

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

Pruebe el RTL-SDR

```
rtl_test
```

Pulse Ctrl+C para detenerlo. Debería ver información sobre la tasa de muestreo.

Edite la configuración

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

Ajustes habituales para RTL-SDR:

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

Receptores sobre SoapySDR (RSP1A, Fobos, Airspy HF+)

El SDRplay RSP1A, el RigExpert Fobos SDR y el Airspy HF+ llegan al servidor de la misma forma: un driver SoapySDR para el dispositivo, y `rx_sdr` (de `rx_tools`) que pasa sus muestras a `spectrumserver`. Un script por receptor instala la cadena completa:

Receptor	Script	Opción del instalador	Script de arranque	Configuración
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>

Cada script:

- instala git, cmake, un compilador, libusb y **SoapySDR** desde los paquetes de su distribución — reconoce apt, dnf, pacman y zypper, así que funciona igual en Debian/Ubuntu, Fedora, Arch y openSUSE;
- compila el driver del receptor desde el código fuente en `sdr_drivers/` (fuera de git);
- compila **rx_tools** para `rx_sdr`, salvo que `rx_sdr` ya esté instalado — una copia sirve a todos los receptores;
- instala una **regla udev** y le añade al grupo `plugdev`, para que el script de arranque nunca necesite `sudo` para el dispositivo. **Cierre la sesión y vuelva a entrar una vez** para que el grupo surta efecto;
- termina comprobando que SoapySDR lista de verdad el driver, y se detiene con un error si no lo hace.

Ejecútelos como su usuario normal — llaman a `sudo` por sí mismos. Son idempotentes: volver a ejecutar uno actualiza las fuentes del driver con `git pull` y recompila. El instalador llama exactamente a estos scripts; en una estación ya instalada, ejecute por separado el de su receptor. La parte común está en `setup-sdr-common.sh`, que no se ejecuta por sí solo. Dos variables opcionales valen para los tres: `SDR_USER=<nombre>` pone otra cuenta en `plugdev`, y `SDR_SKIP_DEPS=1` omite la instalación de paquetes.

Con el dispositivo conectado, compruebe que el receptor se detecta:

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

SDRplay RSP1A

`setup-rsp1a.sh` instala el **driver de código abierto**: `libmirisdr-5`, el driver obtenido por ingeniería inversa para los chips Mirics MSi2500/MSi001 del RSP1 y el RSP1A, con su módulo `SoapySDR SoapyMiri` (`driver=soapyMiri`). No necesita ningún servicio en segundo plano ni `root`. El script además pone en la lista negra los drivers del kernel `msi2500` y `msi001` en `/etc/modprobe.d/blacklist-msi2500.conf`, porque reclaman el RSP1A como radio V4L2 antes de que `rx_sdr` pueda abrirlo. Si el dispositivo estaba conectado mientras estaban cargados, desconéctelo y vuelva a conectarlo una vez (o reinicie).

La API cerrada de `SDRplay` (`driver=sdrplay`, con `SoapySDRPlay` y el servicio `sdrplay`) sigue funcionando — el archivo incluido `instructions-for-rsp1a` describe esa instalación. `start-rsp1a.sh` elige entre ambos por sí mismo: si `SoapySDR` tiene el driver `sdrplay` usa la API y antes reinicia el servicio `sdrplay`; si no, usa `libmirisdr-5`. Una estación montada sobre la API sigue funcionando sin cambios. Para forzar uno, arranque con `RX_DRIVER=miri` o `RX_DRIVER=sdrplay`. El driver elegido queda escrito en `logwebsdr.txt` como RSP1A driver: `miri` o RSP1A driver: `sdrplay`.

RigExpert Fobos SDR

El Fobos entrega **una sola rama a la vez**, así que tiene dos scripts de arranque que comparten el bloqueo del `watchdog` como cualquier `start-*.sh` — arrancar uno detiene el otro:

- **`start-fobos-hf.sh`** — las entradas HF1/HF2 en muestreo directo, 0-25 MHz, en el puerto 9003. En este modo no hay oscilador local: el ADC digitaliza 0-25 MHz en todo momento (`-f 0` es intencionado, y el aviso *Failed to set center freq* de `rx_sdr` es de esperar). `rx_sdr` entrega CF32 a 50 Msps; el pequeño conversor `cf32_to_real` (compilado por `setup-fobos.sh` a partir de `cf32_to_real.c`) conserva el canal I y lo escala a `s16`, por eso `config-fobos-hf.toml` usa `signal="real"` y `format="s16"`.
- **`start-fobos.sh`** — la rama RF, 25-6000 MHz, IQ, en el puerto 9002. Elija la ventana antes del primer arranque: `-f` (centro) y `-s` (frecuencia de muestreo) en la línea `RX_ARGS` deben coincidir con `frequency=` y `sps=` de `config-fobos.toml`. Los valores incluidos (97 MHz, 20 Msps) son solo un punto de partida para comprobar que funciona.

IMPORTANTE

`config-fobos-hf.toml` viene con `accelerator="opencl"`: un flujo de 50 Msps a través de una FFT de 1M bins pesa tanto como los 60 Msps del RX888, y en la CPU se pierden muestras y la máquina se calienta. En una máquina sin dispositivo OpenCL, `spectrumserver` no arranca con ese ajuste — ponga allí `accelerator="none"`.

`SoapyFobosSDR` necesita para compilarse las cabeceras de `libfobos` y de `libfobos-sdr-agile`, así que `setup-fobos.sh` compila ambas, incluso para el firmware estándar. El soporte del Fobos procede de un script que escribió otra estación tras poner en marcha un Fobos con `PhantomSDR-Plus`; los valores HF de `config-fobos-hf.toml` son los que funcionaron allí.

Airspy HF+

`setup-airspyhf.sh` instala `libairspyhf` desde la distribución donde está empaquetada (Debian y Ubuntu, Fedora, openSUSE) y la compila desde el código fuente donde no lo está (Arch la tiene solo en el AUR); después compila `SoapyAirspyHF` (`driver=airspyhf`). `start-airspyhf.sh` entrega 912 ksps IQ en CS16, que `config-airspyhf.toml` lee como `format="s16"`. El archivo incluido `instructions-for-airspy` describe el camino manual.

RX888 MK2

Instale las herramientas 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
```

Ejecutar `rx888_stream` sin `sudo` (reglas `udev`)

Por defecto, el dispositivo USB Cypress FX3 del RX-888 solo es accesible para `root`, de modo que `rx888_stream` necesitaría `sudo`. Ejecute una vez el asistente incluido para instalar las reglas `udev` y añadirse al grupo `plugdev`:

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

Escribe `/etc/udev/rules.d/99-rx888.rules` para los tres identificadores de producto Cypress FX3 (`04b4:00f1/00f3` gestor de arranque y `04b4:8613` con el firmware cargado), recarga `udev` y crea un enlace estable `/dev/rx888`. **Cierre la sesión y vuelva a entrar** (por el cambio de grupo) y **reconecte el dispositivo** una vez; después, `rx888_stream` —y el lanzador `start-rx888mk2.sh`— funcionan sin `sudo`. Es idempotente, así que puede volver a ejecutarlo sin problema. Para configurar otra cuenta: `RX888_USER=<nombre> ./setup-rx888-udev.sh`.

Edite la configuración

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

Ajustes habituales para el RX888:

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

HackRF One

Para el HackRF no se compila nada: `hackrf_transfer`, del paquete `hackrf` de su distribución, pasa las muestras directamente a `spectrumserver`. `./setup-hackrf.sh` (opción 7 del instalador) instala ese paquete con `apt`, `dnf`, `pacman` o `zypper` y añade una regla `udev` y el grupo `plugdev`, para que `start-hackrf.sh` funcione sin `sudo` — cierre la sesión y vuelva a entrar una vez después.

```
./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` entrega una ventana cada vez. Elíjala antes del primer arranque: `-f` (centro) y `-s` (frecuencia de muestreo) en su línea `RX_ARGS` deben coincidir con `frequency=` y `sps=` de `config-hackrf.toml`. `-l` y `-g` son las ganancias del LNA (0-40 dB, pasos de 8 dB) y del VGA (0-62 dB, pasos de 2 dB), y `-a 1` enciende el amplificador de RF. Los valores incluidos (98 MHz, 20 Msps) son solo un punto de partida para comprobar que funciona.

`hackrf_transfer` escribe IQ de 8 bits **con signo**, así que la configuración necesita `format="s8"`. `config.example.hackrf.toml` decía `u8` hasta septiembre de 2026; con `u8` cada muestra está desplazada en 128 y la cascada solo muestra ruido — revise cualquier copia antigua.

Varios receptores en un ordenador

Un ordenador puede hacer funcionar a la vez más de uno de los receptores anteriores — por ejemplo un RX-888 para HF y un RTL-SDR para 2 m — cada uno como servidor propio, con un selector de receptor en la página. Cada receptor adicional se arranca con `INSTANCE=<nombre> ./start-<radio>.sh` y se configura en `instances/<nombre>; receivers.toml` los enumera para el proxy y para `start-all.sh`. Todo está en [Varios receptores](#). La forma más rápida de añadir un receptor es `./add-receiver.sh`, que los instaladores ofrecen al final: instala el controlador (`setup-rtlsdr.sh` para un RTL-SDR, el `setup-*.sh` correspondiente para los demás) y escribe por usted la carpeta de la instancia y `receivers.toml`.

Pruebas y verificación

1. Prueba de arranque

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

Esto inicia el servidor **en segundo plano** (se desvincula y devuelve el control de inmediato) y empieza a registrar en `logwebsdr.txt`.

2. Compruebe si hay errores

El script registra su progreso en `logwebsdr.txt`; obsérvelo en directo:

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

Busque:

- `Starting the initialization script of the PhantomSDR Server`
- `<receiver> running (PID ...)`
- `Server started normally`
- `Server ... FAILED ...` o `ERROR: receiver binary '...' not found` (corrija los argumentos del receptor o el `.toml`, o instale la herramienta del receptor, y vuelva a ejecutar el script)

3. Acceda a la interfaz web

Abra el navegador en:

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

(Sustituya el número de puerto por el que haya configurado)

4. Verifique el funcionamiento

- La cascada debe mostrarse
- El audio debe sonar al hacer clic en las señales
- El S-metro debe responder a las señales
- El contador de usuarios debe mostrar «1»

5. Pruebe desde otro dispositivo

Desde otro ordenador de su red:

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

6. Compruebe el uso de recursos

```
# Monitor CPU usage
htop

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

# Monitor network usage
iftop
```

7. Detenga el servidor

El servidor se ejecuta en segundo plano bajo un watchdog, así que **Ctrl+C no lo detendrá** (y el watchdog volvería a arrancarlo). Use el script de parada compartido, que sirve para cualquier receptor: primero detiene el watchdog y después el receptor y `spectrumserver`:

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

Configuración del arranque automático

Con systemd (recomendado)

1. Cree el archivo de servicio

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

Añada el siguiente contenido (ajuste rutas y usuario):

```
[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
```

El proceso `--watchdog` ya reinicia por su cuenta el receptor y `spectrumserver`; `Restart=always` es solo una red de seguridad para el caso raro de que el propio watchdog termine. Como aquí systemd supervisa el servidor, puede usar `systemctl start/stop/restart` y `journalctl -u phantomsdr -f` en lugar de ejecutar los scripts a mano.

2. Active e inicie el servicio

```
# 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. Gestione el servicio

```
# Start
sudo systemctl start phantomsdr

# Stop
sudo systemctl stop phantomsdr

# Restart
sudo systemctl restart phantomsdr

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

Con Screen (alternativa)

Normalmente innecesario: `./start-rtl.sh` ya se desvincula al segundo plano (mediante `setsid`) y sigue funcionando después de cerrar la sesión, con su propio watchdog. Screen solo resulta útil si quiere específicamente una sesión interactiva para ejecutar la forma en primer plano `./start-rtl.sh --watchdog`.

1. Instale Screen

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

2. Arranque en una sesión de Screen

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

Pulse Ctrl+A y después D para desvincularse.

3. Vuelva a la sesión

```
screen -r phantomsdr
```

Protección térmica de la CPU

Manual completo: [THERMAL_GUARD.md](#) — los cuatro modos y qué debe hacer el sysop en cada uno, funcionamiento sin el panel, la fase throttle sin root, pruebas y resolución de problemas.

PhantomSDR-Plus incluye un guardián que detiene el servidor si la CPU alcanza una temperatura peligrosa y lo vuelve a arrancar una vez fría. Deriva sus umbrales del límite crítico que publica su propia CPU, así que no hay nada que calcular, y funciona con cualquier método de arranque/parada.

Si usa el panel de administración, ya lo tiene: se ejecuta dentro del panel y se configura en su página de Ajustes. Véase la [guía del panel de administración](#). El resto de esta sección es para instalaciones **sin** panel.

1. Compruebe qué ve el guardián en su máquina

```
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
```

Solo se usan sensores reales del die de la CPU (`coretemp`, `k10temp`, `zenpower`, `cpu_thermal`); `acpitz` y las zonas térmicas sin etiquetar se ignoran a propósito, porque a menudo informan de una temperatura de caja o placa decenas de grados por debajo de la CPU. Si aparece `sensor : NONE`, esa máquina no puede protegerse —habitual en un VPS o dentro de un contenedor— y el guardián permanecerá inactivo en lugar de fingir.

No hay nada que instalar: `thermal_guard.py` solo usa la biblioteca estándar de Python.

2. Configúrelo

El guardián lee `admin_config.json` junto a `thermal_guard.py`. Créelo si no lo tiene: sin el panel no lo tendrá. El archivo mínimo útil:

```
{
  "sdr_process_name": "spectrumserver",
  "stop_script": "stop-websdr.sh",
  "start_script": "start-rx88mk2.sh",
  "thermal_mode": "stop+restart"
}
```

- `thermal_mode` — `log` (solo registrar, el valor por defecto), `throttle`, `stop` o `stop+restart`. **Este es el que debe fijar**, porque el guardián se entrega inerte: tal cual, solo anota lo que *habría* hecho. También puede pasarlo en la línea de órdenes como `--mode stop+restart`, que prevalece sobre el archivo: si le valen todos los demás valores por defecto, no necesita archivo de configuración alguno.
- `stop_script` / `start_script` — cómo detiene y arranca su servidor. Sin script de parada recurre a `SIGTERM` y, tras 10 segundos, a `SIGKILL` sobre `sdr_process_name`. Sin script de arranque detiene pero nunca reinicia.
- Los umbrales y tiempos también pueden ponerse aquí, con los mismos nombres de clave que usa el panel; la tabla completa está en la [guía del panel de administración](#).

Si systemd supervisa su servidor SDR, haga que `stop_script` apunte a un pequeño script que ejecute `systemctl stop su-unidad`, en lugar de dejar que el guardián señale el proceso directamente. El guardián ganará de todos modos —mientras la CPU esté demasiado caliente repite la parada cada 2 segundos, de modo que cualquier cosa que reviva el servidor queda deshecha—, pero una parada limpia es mejor que una pelea cada dos segundos.

3. Ejecútelo como servicio

El repositorio incluye una unidad lista, `thermal-guard.service`. Edite en ella `User=` y las dos rutas, y después:

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

`User=` debe ser una cuenta autorizada a ejecutar sus scripts de arranque/parada, normalmente el mismo usuario que ejecuta el servidor SDR. **Un guardián que se ejecuta como un usuario que no puede detener el servidor le deja desprotegido mientras aparenta protegerle.**

Para armar el guardián desde la unidad en lugar del archivo de configuración, añada `--mode` a `ExecStart`:

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

Sin `systemd`:

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

`python3 thermal_guard.py --help` enumera todas las opciones, incluida `--config` para un archivo de configuración que no esté junto al script.

4. Obsérvelo y después confíe en él

Todo lo que hace el guardián se escribe en `crash.log`, en el directorio de `PhantomSDR-Plus`, un evento por línea:

```
[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
```

Deje `thermal_mode` en `"log"` durante una semana y lea ese archivo después de sus momentos duros: una recompilación completa, una tarde calurosa. Si no aparece nada, su máquina nunca se acercó. Después ponga el modo en `stop`.

Antes de confiar en él, compruebe todo el recorrido una vez añadiendo una lectura falsa por encima de su umbral de parada:

```
"thermal_test_temp": 97
```

El guardián la trata como real, así que `warn` → `stop` → bloqueo → recuperación → reinicio se ejecutan cuando usted quiera. Quite la línea (o póngala en `null`) para volver al sensor real. Con el modo en `stop` esto detiene de verdad el servidor y desconecta a sus oyentes, así que hágalo cuando no haya nadie, o pruebe en `log`, donde se ve lo que *habría* pasado sin tocar nada.

Resolución de problemas

Errores de compilación

Problemas de dependencias

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

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

Falla la configuración de Meson

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

meson setup termina con `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'
```

El árbol de fuentes no tiene ningún problema. Una instalación antigua con `pip install --user meson` dejó un script de arranque en `~/local/bin`, que en el `PATH` va antes que `/usr/bin` y oculta la copia que instaló el gestor de paquetes. Una actualización de la distribución (por ejemplo Ubuntu 24.04 → 26.04) cambia Python a una versión nueva, el antiguo `site-packages` que contenía `mesonbuild` ya no está en la ruta de importación, y el script muere antes de hacer nada. Lo mismo le puede ocurrir a `ninja`.

El instalador lo detecta y lo esquivo durante la ejecución, avisándole, pero repare el sistema:

```
rm -f ~/.local/bin/meson
hash -r
meson --version          # debe imprimir una versión
```

Si prefiere conservar meson instalado con pip, reinstálelo para el Python que tiene ahora este sistema:

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

meson setup se detiene con Clock skew detected

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

El árbol de fuentes no tiene ningún problema: es el reloj del sistema el que va por detrás de los archivos. meson y ninja se niegan a compilar cuando una entrada es más reciente que la hora actual, porque no pueden saber qué está obsoleto. Ocurre en una Raspberry Pi sin pila en su soporte RTC — cada arranque parte de la última hora conocida, y una compilación lanzada antes de que NTP se sincronice ve todo el árbol fechado en el futuro — y en cualquier árbol descomprimido o copiado desde una máquina adelantada.

Corrija primero el reloj:

```
timedatectl                      # ¿la hora es correcta? ¿NTP está sincronizado?
sudo timedatectl set-ntp true
```

Espere unos segundos y vuelva a ejecutar el instalador. Comprueba esto antes de llamar a meson y ofrece restablecer las marcas de tiempo afectadas (`PHANTOM_FIX_CLOCK_SKEW=y|n`). A mano, desde el árbol de fuentes:

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

El backend se compiló pero no hay página web

El síntoma es una instalación que parece casi lograda: `build/spectrumserver` existe, el servidor arranca y el navegador no recibe nada — porque `frontend/dist/` nunca se generó. Busque en la salida del instalador:

```
[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` pide `@swc/core ^1.12.14`, así que un `npm install` limpio resuelve a 1.16.0, cuyo `printSync()` rechaza el árbol sintáctico que le entrega el complemento. No tiene nada que ver con su distribución: todos los instaladores fijan ahora la versión que funciona, y `frontend/package.json` también. Una máquina instalada antes de esa fijación sigue funcionando hasta que se borre su `node_modules`, y por eso esto aparece de la nada en una reinstalación.

Si está reparando a mano una copia más antigua:

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

Errores en ejecución

El puerto ya está en uso

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

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

Permiso denegado para el SDR

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

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

Para un **RX-888 MkII** esto no basta por sí solo: el dispositivo Cypress FX3 necesita además reglas udev. Ejecute el asistente incluido (instala las reglas y le añade a `plugdev`), después cierre la sesión, vuelva a entrar y reconecte el dispositivo:

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

Problemas de audio

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

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

Problemas con OpenCL

clinfo no muestra dispositivos

```
# 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
```

El rendimiento no mejora

```
# 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.
```

Problemas de red

No se puede acceder desde otros dispositivos

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

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

Latencia alta

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

# Increase buffer size
buffer_size = 16384

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

Problemas con el dispositivo SDR

No se encuentra el RTL-SDR

```
# 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
```

No se encuentra el SDRplay

`grep "RSP1A driver" logwebsdr.txt` muestra qué driver eligió `start-rspla.sh`.

Con el driver abierto (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"
```

Con la API de SDRplay (RSP1A driver: sdrplay):

```
# Check API installation
systemctl status sdrplay

# Restart API
sudo systemctl restart sdrplay

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

No se encuentra el Fobos o el Airspy HF+

```
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
```

Si el driver falta en la lista, vuelva a ejecutar `./setup-fobos.sh` o `./setup-airspyhf.sh`: se detiene indicando el motivo si falla la compilación o la comprobación de SoapySDR.

Optimización del rendimiento

Optimización de la CPU

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

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

Optimización de la memoria

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

Optimización de la red

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

Actualización de PhantomSDR-Plus

Desde la versión 4.1.0 el repositorio incluye `update.sh`, una herramienta que pone al día una instalación con el árbol publicado **sin tocar los archivos que la hacen ser su estación**. Sustituye al script `git pull` que las ediciones anteriores de esta guía le pedían escribir, y no necesita git en absoluto: el árbol publicado se descarga como tarball y se compara con el suyo archivo por archivo, de modo que funciona igual tanto si clonó el repositorio como si descomprimió un `update.zip` o copió el árbol desde una memoria USB.

Si su instalación aún no tiene update.sh

Un árbol antiguo no contiene el script. Descárguelo una vez: es el único paso de todo este procedimiento que hará a mano:

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

A partir de ahí todo — código, frontend, documentación, instaladores y el propio `update.sh` — llega a través de la herramienta.

Paso 1 — ver qué cambiaría (no escribe nada)

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

Descarga el árbol publicado, lo compara con el suyo e imprime un informe. No escribe absolutamente nada, así que puede ejecutarlo en cualquier momento, incluso con el receptor en el aire. El código de salida es `0` si ya está al día y `10` si hay una actualización esperando, de forma que una tarea de cron puede avisarle cuando haya algo que hacer.

Paso 2 — aplicarla

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

Hay tres clases de archivo y se tratan de forma distinta; esa diferencia es lo esencial:

Archivos	Qué ocurre
<code>config.toml</code> , <code>config-<receptor>.toml</code> , <code>markers.json</code> , <code>admin_config.json</code> , <code>autorun.json</code> , <code>frequencylist/</code> , <code>chat_history.txt</code> , <code>frontend/variant.json</code> , <code>frontend/site_information.json</code> , los registros, <code>build/</code> , <code>frontend/dist/</code>	Nunca se tocan ni aparecen en ninguna pregunta. Son los que hacen que la máquina sea <i>su</i> receptor.
<code>start-*.sh</code> , <code>stop-websdr.sh</code> , las unidades <code>*.service</code> , <code>install*.sh</code> , <code>recompile.sh</code> , <code>setup_admin.sh</code> , <code>smeter_theme.sh</code> , <code>proxy.py</code> , <code>admin_server.py</code> , <code>thermal_guard.py</code> , <code>websdr_relay.py</code> , <code>frontend/src/bands-config.js</code> , las configuraciones de ejemplo <code>config.example.*.toml</code>	Siempre se pregunta , porque son los archivos que un sysop tiene motivos para haber modificado.
Todo lo demás	Se actualiza, tras guardar una copia del archivo antiguo en el archivo de copia de seguridad (vea <i>Deshacer una actualización</i>).

Para cada archivo del grupo intermedio se muestran las diferencias y se ofrecen tres opciones:

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

- **Keep mine** — su archivo se deja exactamente como está.
- **upstream** — se instala la versión nueva, después de respaldar la suya.
- **both** — la versión nueva se escribe junto a la suya como `start-rx888mk2.sh.new`, para que incorpore sus propios cambios cuando quiera.

Cómo es la primera ejecución. La primera vez no hay registro de qué versión tenían sus archivos, así que se le presenta cada archivo del grupo intermedio: unas diez preguntas. Respóndalas así:

Su situación	Respuesta
Nunca tocó ese archivo	<code>u</code> — tome la versión nueva. El caso habitual.
Lo modificó (sus <code>RX888_ARGS</code> , fijación de CPU, una unidad ajustada)	<code>b</code> — se conserva el suyo y el nuevo queda al lado como <code><archivo>.new</code> .
No está seguro	<code>ENTER</code> — se conserva el suyo, no se pierde nada y podrá comparar después.

Su configuración nunca entra en esto: las preguntas son siempre sobre scripts y unidades de servicio.

`update.sh` anota en `.update-state/` la versión de cada archivo que instala. Por eso, a partir de la segunda ejecución distingue un archivo que **usted** editó de otro que simplemente es antiguo, y sólo se detiene a preguntar por los que realmente tocó.

Antes de escribir nada detiene el receptor, el panel de administración, el proxy inverso y el relé de diversidad WebSDR **de la instalación que está actualizando**: lo que sirve a otro directorio se enumera y se deja en marcha, de modo que un segundo clon puede actualizarse mientras el primero sigue en el aire. Al terminar vuelve a arrancar exactamente lo que detuvo. Si han cambiado archivos de código o del frontend, le ofrece ejecutar `recompile.sh`. Nunca borra nada: los archivos que han desaparecido del repositorio se comunican y sólo se eliminan si lo pide con `--prune`.

Deshacer una actualización

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

Todo archivo que una actualización sobrescribe se guarda antes, en un archivo fechado por ejecución dentro de la carpeta normal y visible `update-backups/`:

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

El archivo contiene solo los archivos que esa ejecución sustituyó, cada uno en la ruta que tiene en su árbol, así que puede copiarse a otra máquina como un único archivo, abrirse en cualquier parte o descomprimirse directamente sobre la instalación. Donde `zip` no está instalado se escribe como `.tar.gz`. Se conservan los tres más recientes. Cualquiera de estas órdenes devuelve lo que una actualización sustituyó:

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

Cada archivo lleva además su propio `restore.sh`, para el caso de que el problema sea el propio `update.sh`: descomprima el archivo y pásele el directorio de su instalación, por ejemplo `./restore.sh /home/pi/PhantomSDR-Plus`. Durante la restauración `update.sh` se ejecuta desde una copia fuera del árbol, porque normalmente es uno de los archivos que se devuelven.

Las copias anteriores al 23 de septiembre de 2026 están en la carpeta oculta `.update-backups/<marca de tiempo>/`; se restauran igual y cuentan entre las tres que se conservan.

Otras opciones

```
./update.sh --apply --yes      # no pregunta nunca; TODO lo que usted editó se CONSERVA
./update.sh --ref v5.0.0      # una etiqueta, rama o commit en lugar del árbol actual
./update.sh --list-excludes    # imprime las reglas de "no tocar" tal como se aplican aquí
./update.sh --verbose          # lista todos los archivos, no sólo los 40 primeros
```

Puede añadir sus propias reglas de "no tocar" escribiendo un patrón por línea en `update-exclude.txt`, en la carpeta raíz de la instalación.

Si la reconstrucción falla en una instalación muy antigua

`update.sh` actualiza archivos, no paquetes del sistema. Si su árbol es tan antiguo que la compilación necesita bibliotecas que no tiene, `recompile.sh` se detendrá con un error del compilador o de meson. No es una actualización rota: faltan las dependencias:

```
./install.sh
```

El instalador se actualiza en la misma ejecución, y su configuración también sobrevive a él.

Actualizar a mano

Si prefiere aplicar usted mismo un paquete de archivos:

```
cd ~
unzip -o update.zip
cd PhantomSDR-Plus
./stop-websdr.sh
cp -a ~/update/. .
chmod +x *.sh
./recompile.sh
./start-rx88mk2.sh          # o el script de arranque de su receptor
```

Haga antes una copia de su configuración — `config-*.toml`, `frontend/site_information.json`, `admin_config.json` y `markers.json` —, porque un paquete de archivos no puede distinguir sus cambios de los de la versión publicada. Ese es justamente el problema que `update.sh` resuelve.

Copia de seguridad y restauración

Archivos que conviene respaldar

- Archivos de configuración: `*.toml`
- Información del sitio: `frontend/site_information.json`
- Marcadores: `markers.json`
- Scripts personalizados: `start-*.sh`, `stop-*.sh`
- Historial del chat: `chat_history.txt`
- Imagen de fondo: `frontend/src/assets/background.jpg`

Comando de copia de seguridad

```
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
```

Comando de restauración

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

Consideraciones de seguridad

Configuración del cortafuegos

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

Proxy inverso (opcional)

Considere usar nginx o Apache como proxy inverso para:

- cifrado SSL/TLS
- asignación de nombre de dominio
- equilibrio de carga
- control de acceso

Límite de usuarios

Edite `config.toml`:

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

Cómo obtener ayuda

Recursos

- **Documentación:** esta guía, README.md, USER_GUIDE.md
- **GitHub Issues:** <https://github.com/sv1btl/PhantomSDR-Plus/issues>
- **Demostración en directo:** <http://phantomsdr.no-ip.org:8900/>

Cómo informar de problemas

Al informar de un problema, incluya:

1. sistema operativo y versión
2. modelo del dispositivo SDR
3. contenido del archivo de configuración
4. mensajes de error
5. uso de recursos del sistema (CPU, RAM, GPU)

Soporte de la comunidad

- Consulte las incidencias existentes en GitHub antes de crear otras nuevas
- Aporte información detallada sobre su instalación
- Incluya registros y mensajes de error
- Sea paciente y respetuoso

Apéndice A: lista completa de dependencias

Lista de paquetes de 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)
```

Apéndice B: ejemplos de configuración

Ejemplo 1: RTL-SDR para VHF/UHF

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

[input.driver]
format = "u8"

[input.defaults]
frequency = 145000000
modulation = "FM"
```

Ejemplo 2: RX-888 mk2 para HF (0-30 MHz)

El receptor que usa la mayoría de los sysops. Esta es una sección `[input]` completa, no un fragmento, con los valores del `config-rx888mk2.toml` que acompaña al repositorio.

```
[input]
sps = 60000000          # 0-30 MHz por muestreo directo
fft_size = 4194304      # vea la nota de abajo
fft_threads = 8
brightness_offset = -9  # más negativo si la cascada muestra zonas negras
frequency = 0           # banda base: el RX-888 muestrea desde DC
signal = "real"         # no "iq": el muestreo directo da una señal real
accelerator = "opencl"  # "none" si no hay entorno OpenCL
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"
```

Se alimenta desde `rx888_stream`, cosa que `start-rx888mk2.sh` hace por usted:

```
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
```

Sobre `fft_size`: a 60 MSPS el tamaño adecuado es 4194304. 8388608 duplica la resolución de la cascada y con ella la memoria y el coste de CPU de cada transformada, lo que en la mayoría de las máquinas compra bins más finos a cambio de fotogramas perdidos. Empiece en 4194304 y suba solo si el servidor va sobrado.

Ejemplo 3: HackRF para FM de banda ancha

```
[input]
sps = 100000000
frequency = 1009000000
signal = "iq"

[input.driver]
format = "s8"

[input.defaults]
frequency = 1009000000
modulation = "WBFM"
```

¡Instalación completada! Ya debería tener un servidor PhantomSDR-Plus plenamente funcional.

73 de SV1BTL & SV2AMK

CAPÍTULO 3

Configuración del panel de administración

ADMIN_PANEL_SETUP.md

Panel de administración de PhantomSDR-Plus — Guía de configuración

⚠ **Aviso de seguridad:** mantenga este panel de administración en su red doméstica o detrás de una VPN. No se recomienda exponer públicamente el puerto de administración sin un refuerzo adecuado de la autenticación.

Descripción general

El panel de administración consta de dos servicios de Python:

Servicio	Archivo	Función
Panel de administración	<code>admin_server.py</code>	Aplicación web Flask, se enlaza a 127.0.0.1 (solo interna)
Proxy inverso	<code>proxy.py</code>	Expone tanto el SDR como el panel de administración en un único puerto público

Ambos servicios leen su configuración de `admin_config.json`, que escribe `setup_admin.sh`. **No edite las constantes de puerto directamente en los archivos de Python:** todos los puertos y la IP del host SDR se guardan en `admin_config.json`.

Qué le ofrece el panel de administración

- **Panel de control** — estado del servidor en directo, CPU/RAM, procesos principales, salida reciente de los registros, terminal
- **Editor de configuración** — ver, editar y guardar cualquier archivo `.toml`, `.sh`, `.json`, `.h` o `.cpp` de su instalación
- **Visor de registros** — seguir cualquier archivo de registro en tiempo real (servidor SDR, panel de administración, RADE, fallos, autorun, proxy)
- **Marcadores** — ver y editar los marcadores de frecuencia
- **Historial del chat** — ver el registro del chat del WebSDR, borrarlo por completo o eliminar mensajes concretos sin reiniciar el servidor
- **Mensaje en la cascada** — difundir un cartel persistente a todos los usuarios conectados, visible en tiempo real en la cascada
- **Usuarios** — lista en directo de oyentes conectados con su frecuencia, modo, duración y un botón Kick
- **Eliminación de mensajes del chat** — botón Delete que quita de inmediato ese único mensaje del registro

- **Mensajes difundidos en la cascada** — permite enviar un cartel de texto persistente a la cascada de cada usuario conectado
- **Spot Reporting** — iniciar/detener el decodificador autorun de FT8/FT4/JS8/WSPR, elegir bandas y modos e informar de spots a PSK Reporter / wsprnet (desactivado por defecto), con dos contadores: paneles por decodificador para la ejecución actual y un total histórico junto a cada casilla de banda/modo
- **Gráficas** — frecuencia de CPU, carga de CPU, temperatura de CPU y usuarios conectados, representados en los últimos 15 minutos / 1 hora / 4 horas / 12 horas / 24 horas
- **Protección térmica (Thermal Guard)** — detiene el servidor si la CPU se sobrecalienta y lo reinicia cuando se enfría, con umbrales derivados de su propia CPU; funciona con cualquier método de arranque/parada y se entrega en modo de solo registro, por lo que no hace nada hasta que usted lo active
- **Ajustes** — cambiar la contraseña de administrador, el directorio base del SDR, el nombre del proceso y el puerto público

Requisitos

- PhantomSDR-Plus ya instalado y en funcionamiento
- Python 3.8+

NOTA

Normalmente el instalador ya hace esto por usted. `./install.sh` — igual que cada uno de los cuatro instaladores por distribución, los dos instaladores de RADE y el del servidor de estadísticas — configura el panel por defecto, instalando antes `pip` con el gestor de paquetes del sistema si falta. Esta página describe lo que ocurre en ese paso y qué hacer si lo omitió o quiere cambiar algo después.

Las bibliotecas de Python del panel (`flask`, `psutil`, `aiohttp`, `tomli-w`) las instala `setup_admin.sh`; si eso falla, imprime el comando exacto que debe ejecutar.

Paso 1 — Archivos

Coloque estos cuatro archivos dentro de su directorio PhantomSDR-Plus:

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

Haga ejecutables los scripts de shell:

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

Paso 2 — Ejecutar el script de configuración

```
./setup_admin.sh
```

El script hará lo siguiente:

1. Comprobar que Python 3 está instalado
2. Verificar que existen `admin_server.py` y `manage_admin.sh`
3. Registrar `127.0.0.1` como `sdr_host` en la configuración (véase [El ajuste sdr_host](#))
4. Pedir tres números de puerto. Cada pregunta ofrece un valor predeterminado entre corchetes que se acepta con un simple Enter, de modo que una instalación normal son tres pulsaciones:
5. **Puerto de spectrumserver** — el puerto en el que escucha su servidor SDR (predeterminado `8900`, que es justo el que trae cada `config-*.toml` del repositorio)
6. **Puerto interno del panel de administración** — donde `admin_server.py` se enlaza localmente (predeterminado `3000`)
7. **Puerto público del proxy** — el único puerto externo que combina SDR y administración (predeterminado `8902`)

Las respuestas no válidas se rechazan y se vuelven a pedir, pero solo cinco veces; después se usa el valor predeterminado. Una ejecución cuya entrada no es un terminal (por tubería, cron, desatendida) toma los valores predeterminados de inmediato en lugar de esperar una entrada que nunca llegará.

1. Instalar `flask`, `psutil`, `aiohttp` y `tomli-w` mediante `pip`
2. Conceder a `ss` la capacidad `cap_net_admin` (ruta alternativa de la función de expulsar usuarios, solo necesaria si el panel funciona sin el proxy)
3. Preguntar qué script arranca y cuál detiene el receptor — el panel maneja el servidor a través de ellos, y la protección térmica también
4. Preguntar hasta dónde puede llegar por sí sola la protección térmica de la CPU — `log`, `throttle`, `stop` o `stop+restart` — y ofrecer ejecutar `setup-cpufreq-perms.sh` para que la fase `throttle` funcione sin root (véase [THERMAL_GUARD.md](#))
5. Escribir `admin_config.json` con todos los ajustes
6. Ofrecer la instalación de dos unidades `systemd` (panel + proxy) — arranque automático y reinicio tras un fallo. **Recomendado, y es la opción por defecto** (basta con pulsar Enter): la protección contra sobrecalentamiento solo funciona mientras el panel está en marcha, así que sin las unidades un reinicio deja la máquina desprotegida. El paso se omite automáticamente donde `systemd` no es PID 1 (contenedores, WSL1, OpenRC) o donde no tenga `sudo`.

Tras la configuración, abra el navegador a través del proxy:

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

Contraseña por defecto: `admin`

⚠ **Cambie la contraseña de inmediato:** vaya a Settings en el primer inicio de sesión. Un asistente de primera ejecución le guiará para establecer el directorio del SDR, el nombre del proceso, el puerto público y una contraseña nueva.

Paso 3 — Iniciar / detener / reiniciar

Hay dos formas de ejecutar el panel y el proxy. **Elija una y manténgala** — la advertencia al final de esta sección explica qué ocurre si las mezcla.

El Método B (systemd) se recomienda encarecidamente siempre que haya systemd. La protección contra sobrecalentamiento se ejecuta dentro del panel, así que un panel que muere a las 03:00 o una máquina que se reinicia dejan la CPU desprotegida hasta que alguien la arranque a mano. `setup_admin.sh` ya ofrece las unidades por defecto. Use el Método A cuando no haya systemd — contenedores, WSL1, OpenRC/sysvinit — o cuando quiera supervisarlo usted deliberadamente.

Método A — `manage_admin.sh` (sin root, sin instalar nada)

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

Deliberadamente **no hay restart**. Antes mataba el panel y arrancaba de inmediato su propia copia, lo que choca con systemd en cualquier máquina que use el Método B. Para reiniciar con el Método A, ejecute `stop` y luego `start` — véase [Reiniciar el panel](#) más abajo.

Un único script controla **ambos** procesos. Comprueba que `aiohttp` esté instalado antes de arrancar el proxy y muestra un error claro con el comando de solución si falta.

Nada se inicia solo: tras un reinicio, o si el panel se cae, lo arranca usted a mano. Es aceptable para un receptor que vigila, pero tenga en cuenta que la protección térmica de la CPU se ejecuta dentro del panel: mientras el panel esté caído, nada protege la máquina (véase [THERMAL_GUARD.md](#)).

Método B — unidades systemd (arranca al inicio, se reinicia tras un fallo)

El repositorio incluye dos unidades listas, `phantomsdr-admin.service` y `phantomsdr-proxy.service`. Ajuste `User=` y las rutas en ambas, y después:

```
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
```

Los comandos del día a día pasan a ser:

```
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
```

Los mismos cuatro con `phantomsdr-proxy`, o ambos a la vez: `sudo systemctl restart phantomsdr-admin phantomsdr-proxy`.

`admin.log` y `proxy.log` siguen funcionando igual que antes — las unidades escriben en esos mismos archivos.

Algo que conviene hacer una vez: instalar la regla `tmpfiles` de `tmpfiles/phantomsdr-logs.conf` (cambie antes sus rutas y el usuario). systemd crea esos dos registros como `root` antes de bajar a `User=`, y entonces el botón **Clear Logs** del panel falla con `Permission denied` — véase más abajo «Clear Logs dice Permission denied». `setup_admin.sh` instala la regla por usted.

La unidad del panel lleva `SupplementaryGroups=cpufreq`, que es lo que permite a la fase `throttle` de la protección térmica bajar el reloj de la CPU sin que el panel se ejecute como `root`. Cree ese grupo primero con `./setup-cpufreq-perms.sh`, o systemd se negará a arrancar la unidad; elimine la línea si no usa la fase `throttle`.

Para volver al Método A:

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

⚠ **No mezcle ambos.** Con las unidades activadas, `./manage_admin.sh stop` lo deshace `systemd` cinco segundos después, y `./manage_admin.sh start` le deja una segunda copia sin gestionar peleando con la de `systemd` por el mismo puerto. (`restart` se eliminó del script justo por esto; ahora imprime un error remitiéndole a `systemctl`.) `./manage_admin.sh status` solo lee, así que sigue siendo útil en ambos casos.

Reiniciar el panel

El resto de este documento dice «reinicie el panel» en varios sitios. Significa el que corresponda a su método:

```
# Método B – unidades systemd (la configuración normal)
sudo systemctl restart phantomsdr-admin phantomsdr-proxy

# Método A – sin unidades instaladas
./manage_admin.sh stop && ./manage_admin.sh start
```

La unidad incluye `KillMode=process`, y esa línea es esencial. Si arranca el SDR desde el panel, el receptor es hijo de la unidad de administración y hereda su cgroup. Con el valor por omisión de `systemd`, `KillMode=control-group`, reiniciar el panel se llevaría por delante `spectrumserver`, su `watchdog` y el demonio de `autorun`, tras bloquearse los 90 segundos completos del tiempo de espera de parada. Con `KillMode=process` `systemd` solo señala al panel, de modo que `sudo systemctl restart phantomsdr-admin` deja al aire un receptor con oyentes.

Actualizar una unidad instalada antes de la v4.1.0. Los ficheros de unidad antiguos no tienen esa línea, y reiniciar el panel no la añade: `systemctl restart` vuelve a ejecutar el programa, no cambia su configuración. Edite la copia *instalada* — la del repositorio es solo una plantilla que `systemd` nunca lee:

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

Añada `KillMode=process` en la sección `[Service]`, junto a `Restart=always`. Después haga que `systemd` relea el fichero y compruébelo:

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

El segundo comando debe imprimir `KillMode=process`. Si sigue imprimiendo `control-group`, `systemd` está usando su copia en caché — `systemctl show phantomsdr-admin -p NeedDaemonReload` indica `yes` mientras quede una recarga pendiente. Los espacios iniciales se ignoran en los ficheros de unidad, así que la indentación no importa.

Paso 4 — Abrir el puerto en el cortafuegos

Abra en el cortafuegos únicamente el **puerto público del proxy**. No es necesario exponer al exterior el puerto interno del panel de administración:

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

Paso 5 — Función de expulsar usuarios

La página Users muestra a todos los oyentes conectados en ese momento. Cada fila indica la IP del usuario, la frecuencia sintonizada, el modo y cuánto tiempo lleva conectado. Para desconectar a un usuario, pulse el botón **Kick** de su fila: solo se corta esa conexión TCP. Los demás oyentes siguen conectados y no notan nada.

La expulsión es instantánea y quirúrgica: el servidor no se reinicia, no se interrumpe el audio de nadie más y el usuario expulsado puede volver a conectarse de inmediato (la función sirve para eliminar conexiones problemáticas o bloqueadas, no para vetar).

Internamente, la expulsión usa `ss -K dst <IP> dport <port>` para cerrar ese socket TCP concreto. Esto requiere una capacidad de Linux que el script de configuración concede automáticamente. Si se saltó la configuración o el botón informa de un error, aplíquela manualmente:

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

Configuración — admin_config.json

Toda la configuración de ejecución reside en `admin_config.json`, dentro de su directorio PhantomSDR-Plus. `setup_admin.sh` escribe el archivo inicial; los cambios posteriores pueden hacerse desde la página Settings o editando el archivo directamente.

Campos clave escritos por `setup_admin.sh`:

Clave	Descripción
<code>port</code>	Puerto interno del panel de administración (donde se enlaza <code>admin_server.py</code>)
<code>public_port</code>	Puerto de spectrumserver (usado por la página Users para consultar <code>/users</code>)
<code>proxy_port</code>	Puerto público en el que escucha el proxy
<code>sdr_host</code>	Host que usa el proxy para llegar a spectrumserver — <code>127.0.0.1</code> (véase más abajo)
<code>password_hash</code>	Hash SHA-256 de la contraseña de administrador
<code>sdr_base_dir</code>	Ruta de su instalación de PhantomSDR-Plus
<code>sdr_process_name</code>	Nombre del proceso a vigilar (por defecto: <code>spectrumserver</code>)
<code>start_script</code> / <code>stop_script</code>	Scripts que el panel ejecuta para arrancar y detener el servidor SDR — también son la vía por la que actúa la protección térmica

¿Varios receptores en un ordenador? Ponga `start_script` en `start-all.sh`. El script de parada ya detiene todos los receptores, y Restart y la protección térmica usan el script de arranque; con `start-all.sh` devuelven todos los receptores de `receivers.toml` en lugar de solo el principal. La página Users, el recuento de oyentes y la expulsión abarcan entonces también todos los receptores. Vea [Varios receptores](#).

Claves de la protección térmica (todas opcionales: el panel las escribe al guardar los ajustes de Thermal Guard, y las que falten usan estos valores por defecto):

Clave	Por defecto	Descripción
thermal_enabled	true	Interruptor general de la protección
thermal_mode	"log"	log (solo registrar, nunca actuar) · throttle · stop · stop+restart
thermal_warn	null	Temperatura de aviso °C; null = derivar del límite crítico de su CPU
thermal_throttle	null	Temperatura a la que se baja el techo de frecuencia de la CPU; null = automático
thermal_stop	null	Temperatura a la que se detiene el servidor; null = automático
thermal_resume	null	Temperatura por debajo de la cual debe bajar la CPU para recuperarse; null = automático
thermal_sustain_s	60	Segundos que debe mantenerse la temperatura de parada antes de actuar
thermal_warn_sustain_s	30	Segundos que deben mantenerse las temperaturas de aviso/throttle
thermal_resume_s	300	Segundos por debajo de la temperatura de recuperación antes de liberar el bloqueo
thermal_max_stops_hour	2	Paradas térmicas por hora antes de desactivar el reinicio automático
thermal_test_temp	null	Temperatura ficticia para probar la protección; null = usar el sensor real

Los cambios surten efecto en la siguiente muestra — en un par de segundos, sin reiniciar.

Para cambiar los puertos tras la configuración inicial, edite `admin_config.json` y reinicie:

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

El ajuste `sdr_host`

`proxy.py` se conecta a `spectrumserver` por bucle local — `sdr_host` es `127.0.0.1` en `admin_config.json`, escrito ahí por `setup_admin.sh`. No debería necesitar cambiarlo.

Póngale una dirección real solo si `spectrumserver` se ejecuta en una **máquina distinta** a la del proxy. Después [Reiniciar el panel](#).

Cambio: las versiones anteriores exigían que `sdr_host` fuera la IP local de la máquina, porque `spectrumserver` cerraba los WebSocket que llegaban desde la dirección de bucle local. Ese filtro se ha eliminado, así que las conexiones locales funcionan con normalidad. Dos consecuencias: abrir `http://localhost:<port>` en la propia máquina del servidor ahora muestra la interfaz completa (antes cargaba la página pero se quedaba en blanco), y el proxy ya no se rompe cuando DHCP reasigna la IP de la máquina. Si está actualizando y su `admin_config.json` todavía contiene una IP local, cámbiela a `127.0.0.1` o vuelva a ejecutar `setup_admin.sh`.

proxy.py — qué es y cuándo lo necesita

`proxy.py` coloca tanto el servidor SDR como el panel de administración en un **único puerto público**, enrutando según el prefijo de la ruta:

Ruta	Se dirige a
/admin*	Panel de administración (localhost:<port>)
Todo lo demás	Spectrumserver (<sdr_host>:<public_port>)

Sin el proxy tendría que exponer dos puertos por separado. Con él solo expone uno.

El proxy también elimina el límite de 4 MB en el tamaño de los mensajes WebSocket (importante para las tramas FFT grandes del RX-888 a 60 MSPS), reenvía las IP reales de los clientes mediante cabeceras X-Forwarded-* y mantiene vivas las conexiones de larga duración a través de NAT con un latido cada 30 segundos.

Cambiar los puertos tras la configuración

Edite `admin_config.json` directamente:

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

Después reinicie ambos servicios:

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

Si además necesita que `admin_server.py` se enlace a otro puerto al arrancar (p. ej., en el servicio `systemd`), puede sobrescribirlo con la variable de entorno:

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

Por defecto, `admin_server.py` solo se enlaza a `127.0.0.1`. Para ejecutarlo por separado sin el proxy (desarrollo/pruebas), enlázelo a todas las interfaces:

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

Comandos manuales útiles

```
# 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
```

Archivos de registro

Archivo	Contenido
admin.log	Salida del panel de administración
proxy.log	Cartel de arranque del proxy + una línea de registro de acceso por cada petición reenviada
logwebsdr.txt	Registro del lanzador y el watchdog: arranque/parada, driver del receptor, registro en websdr.org
spectrumserver.log	La salida del propio proceso receptor: configuración de FFT y OpenCL, conexiones de cada cliente, errores. El mayor de estos ficheros; se rota, así que puede haber un spectrumserver.log.1 a su lado
rade.log	Registro del sidecar RADE/FreeDV
autorun.log	Registro del demonio decodificador — spots, SNR, deriva
crash.log	Informes de fallos y eventos térmicos: trazas [CRASH] / [TERMINATE], notas [EXIT] y líneas [THERMAL] de la protección. Normalmente vacío: cualquier cosa que contenga merece leerse

Todos ellos pueden leerse desde las pestañas del **Log Viewer** del panel (LOGWEBSDR, SPECTRUMSERVER, ADMIN, RADE, CRASH, AUTORUN, PROXY). El visor muestra las últimas 150 líneas de la pestaña seleccionada y se actualiza cada 3 segundos con la actualización automática activada; solo lee el final del fichero, así que su tamaño da igual.

Rotación de proxy.log y admin.log

proxy.log registra cada petición reenviada y admin.log cada petición que llega al propio panel. Ninguno de los dos se rota por defecto. Antes las consultas del panel dominaban ambos archivos — /admin/api/status se lanza cada ~3 segundos mientras haya una pestaña abierta—, pero ahora se filtran de los dos registros (véase más abajo), así que la rotación sirve para acotar un crecimiento lento, no una avalancha.

El repositorio incluye una configuración de logrotate en logrotate/phantomsdr (diaria, o antes si un registro supera los 10 MB; conserva 7 archivos comprimidos en logproxy/, para que no llenen la raíz del proyecto). **Es específica de cada instalación: ábrala y cambie las tres rutas de registro, la ruta olddir y el usuario su para que coincidan con su máquina antes de instalarla**, y después:

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

logrotate/phantomsdr en el repositorio y /etc/logrotate.d/phantomsdr son dos copias **independientes**: cp no las enlaza. Editar el archivo del repositorio no cambia nada en un sistema en marcha hasta que vuelva a ejecutar el sudo cp anterior. Logrotate solo lee la copia de /etc.

La configuración usa copytruncate, algo imprescindible: manage_admin.sh arranca ambos procesos con >> <log> y ninguno reabre su salida estándar, de modo que una rotación basada en renombrado los dejaría escribiendo en el archivo rotado mientras el nuevo registro quedaría vacío para siempre.

Los archivos comprimidos van a logproxy/ (creado automáticamente por createolddir) y se llaman proxy.log.1.gz ... proxy.log.7.gz, el más reciente primero. Lea uno con zcat logproxy/proxy.log.1.gz | less, o búsquelos todos a la vez con zgrep "pattern" logproxy/*.gz.

El botón **Clear Logs** del panel vacía **todos** los archivos de la tabla anterior, incluidos autorun.log y proxy.log (CLEAR_EXCLUDE en admin_server.py está vacío). Tenga en cuenta lo que eso cuesta en autorun.log: su historial de decodificación es el único registro de qué se escuchó y cuándo, y no puede reconstruirse; de un vaciado solo sobrevive lo que logrotate ya haya rotado a logproxy/, así que se descarta todo lo decodificado desde la última rotación. Vaciarlo con el decodificador en marcha es seguro: el demonio lo mantiene abierto con 0_APPEND y sigue escribiendo en el mismo inodo. El aviso indica qué se vació, y si algún archivo no se pudo escribir —por ejemplo un proxy.log que pertenece a otro usuario en una instalación con cuentas separadas— nombra ese archivo y el error en lugar de informar de un éxito, y deja en pantalla el panel de ese registro en vez de vaciarlo. Los archivos se truncan en el sitio, nunca se borran: tanto manage_admin.sh como las unidades de systemd los abren con 0_APPEND, de modo que un archivo borrado seguiría llenando un inodo invisible hasta reiniciar el servicio.

Las consultas correctas se filtran de **ambos** registros de acceso, mediante el QuietPollFilter de proxy.py y el de admin_server.py. El panel llama a /admin/api/status cada 3 segundos y a /admin/api/logs, /admin/api/autorun/status, /admin/api/users, /admin/api/graph-stats y /admin/api/chat cada 5 segundos; sin filtrar serían el ~95 % de los dos archivos. Solo se descartan los 200 simples en esas rutas: cualquier otro estado, ruta o método se registra como antes, de modo que una consulta fallida sigue siendo visible y las acciones que cambian el estado (kick, autorun/start, ...) usan rutas distintas y quedan siempre registradas. La única excepción deliberada es /admin/api/logs/clear, filtrada en ambos archivos: su propia línea de acceso se escribe *después* del vaciado, así que sin el filtro cada limpieza correcta dejaba una línea recién escrita en el registro que acababa de vaciar, y el botón parecía averiado.

Como proxy.log es un registro de acceso, contiene direcciones IP de visitantes y cadenas de user-agent; téngalo en cuenta antes de compartirlo al pedir ayuda.

«Clear Logs» dice Permission denied

En una máquina donde las unidades systemd se instalaron sobre un árbol limpio, el botón puede responder:

```
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)
```

Siempre son esos dos archivos y nunca los otros cinco, porque son los únicos que crea el propio systemd. StandardOutput=append: lo abre PID 1 **antes** de bajar a User=, así que un registro que aún no existe se crea como root:root 0644. El panel se ejecuta como su usuario y no puede vaciarlo. Los otros cinco registros los crean los scripts de arranque, que ya se ejecutan como su usuario, por eso se limpian con normalidad. Donde los dos archivos son anteriores a las unidades —manage_admin.sh los creó con >>

— ya tienen el propietario correcto y el botón funciona; por eso el problema solo aparece en una instalación nueva que empezó directamente con systemd.

Compruébelo con `ls -l admin.log proxy.log`: los que fallan dicen `root root`. Repárelos:

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

No hace falta reiniciar: systemd sigue escribiendo por el descriptor de archivo que ya tiene abierto, y añadir datos no cambia la propiedad. Pulse **Clear Logs** de nuevo y ambos se vacían.

No borre los archivos en su lugar. Ambos están abiertos con `O_APPEND`; si elimina uno, systemd seguirá llenando un inodo invisible hasta que el servicio se reinicie, y el archivo que reaparece vuelve a ser de root.

Para que no vuelva a ocurrir, instale la regla tmpfiles que incluye el repositorio en `tmpfiles/phantomsdr-logs.conf`. Recrea ambos registros con el propietario correcto en cada arranque, antes de que se inicien las unidades, de modo que un registro eliminado —limpieza manual, reinstalación, directorio de instalación movido— nunca vuelva siendo de root. **Es específica de cada máquina: ábrala y cambie primero las dos rutas y el user:group para que coincidan con su equipo**, y luego:

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

Igual que con `logrotate/phantomsdr`, el archivo del repositorio y la copia en `/etc` son independientes: editar el del repositorio no cambia nada hasta que vuelva a ejecutar el `cp`. En systemd 254 y posteriores la regla también repara la propiedad y el modo de un registro que ya existe, así que soluciona tanto el fallo actual como los futuros; en versiones anteriores es el `chown` de arriba lo que desbloquea el botón.

`setup_admin.sh` hace todo esto por usted cuando instala las unidades, así que una configuración hecha con el script nunca se topa con esto. Los pasos anteriores son para un par instalado a mano, o para una instalación anterior al paso de propiedad de los registros del script.

Logrotate no reintroduce el problema: la configuración incluida usa `copytruncate`, que conserva el mismo inodo y, por tanto, el mismo propietario.

Resumen de acceso

Configuración	SDR	Panel de administración
Sin proxy	<code>http://YOUR_IP:<public_port></code>	<code>http://YOUR_IP:<port>/admin</code>
Con proxy	<code>http://YOUR_IP:<proxy_port></code>	<code>http://YOUR_IP:<proxy_port>/admin</code>

Eliminación de mensajes del chat

La página Chat History enumera todos los mensajes del registro de chat actual. Cada entrada tiene un botón **Delete** que quita de inmediato ese único mensaje del registro, sin reiniciar el servidor.

Cómo funciona:

- El panel de administración reescribe el archivo de registro del chat sobre la marcha, eliminando solo la línea del mensaje seleccionado.

- El cambio surte efecto para cualquier usuario que recargue el chat; el historial ya cargado en pestañas abiertas no se actualiza retroactivamente.
- Vaciar todo el registro (botón **Clear All**) trunca el archivo, lo que también surte efecto sin reiniciar.
- Una **respuesta del chat** se guarda como un mensaje corriente que empieza con una marca que nombra el mensaje al que responde, por ejemplo [RE:2026-09-25 12:00:01|SV1BTL] , así que en esta lista aparece con esa marca delante. Borrar una respuesta elimina solo esa respuesta. Borrar el mensaje al que responde deja sus respuestas en su sitio, mostrando solo el nombre y la hora a los que responden.

El botón de eliminar se muestra de forma coherente en las cinco variantes de la aplicación Svelte. Si una eliminación no parece surtir efecto, confirme que el panel de administración tiene permiso de escritura sobre el archivo de registro del chat:

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

Mensajes difundidos en la cascada

El panel **Waterfall Message** permite enviar un cartel de texto persistente a la cascada de cada usuario conectado sin tocar el proceso del servidor.

Enviar un mensaje

1. Abra el panel de administración y vaya a **Waterfall Message**.
2. Escriba el texto del mensaje y elija un color (hexadecimal, p. ej., #ffdd00).
3. Pulse **Send**: el mensaje aparece de inmediato en todas las vistas de cascada activas.

Borrar el mensaje

Pulse **Clear** para quitar el cartel de todas las cascadas. El estado del mensaje se mantiene en memoria en el panel de administración; se borra automáticamente si el proceso del panel se reinicia.

Cómo funciona

El panel de administración expone dos puntos finales internos que el proxy reenvía:

Punto final	Método	Finalidad
/admin/api/waterfall-message	POST	Fijar o borrar el texto y el color del cartel actual
/admin/api/waterfall-message	GET	Devolver el estado actual del mensaje en JSON

El frontend consulta el estado del mensaje y lo dibuja superpuesto sobre el lienzo de la cascada. En el cliente no hace falta ni reconectar el WebSocket ni recargar la página.

Usos habituales

- Anunciar un mantenimiento programado: "Server restart in 10 minutes"
- Señalar las condiciones de banda: "Solar flux 180 – 10m wide open"
- Mensaje de bienvenida: "Welcome to SV1BTL WebSDR – Athens, KM17"

Gráficas del sistema

La página **Gráficas** representa cuatro magnitudes sobre un eje de tiempo común:

- **Frecuencia de CPU** — frecuencia media de todos los núcleos, en GHz
- **Carga de CPU** — utilización total, en porcentaje
- **Temperatura de CPU** — en °C, con guías discontinuas a 70 °C (caliente) y 80 °C (crítica)
- **Usuarios conectados** — oyentes conectados al WebSDR en ese momento

Elija el intervalo con **15 MIN / 1 HOUR / 4 HOURS / 12 HOURS / 24 HOURS**. Al pasar el ratón por la gráfica aparece una cruz con los cuatro valores de ese instante. Las cuatro tarjetas superiores muestran siempre la muestra más reciente.

Cómo se muestrea

Un hilo en segundo plano de `admin_server.py` toma una muestra cada **2 segundos**. Las muestras se guardan en dos niveles: la última **hora** a resolución completa de 2 segundos (1800 puntos) y **24 horas** de promedios de 30 segundos (2880 puntos) para los intervalos largos. La página elige el nivel que cubre el intervalo elegido e indica `30s averages` en la cabecera cuando está mostrando el nivel promediado.

Guardar un día entero a 2 segundos serían ~43000 puntos: decenas de MB de memoria, una primera descarga de varios megabytes y unos 40 puntos por píxel de lienzo, que ninguna pantalla puede representar. Promediar a 30 segundos mantiene un día en 2880 puntos.

El muestreo arranca junto con el panel de administración, no al abrir la página por primera vez, de modo que la página se abre sobre un historial que ya existe.

El búfer está **solo en memoria**: no se escribe nada en disco y el historial se pierde al reiniciar el panel. Es deliberado: mantiene la función fuera de la rotación de registros y del disco.

2 segundos ofrecen una vista casi en vivo y es un intervalo lo bastante corto para no falsear la frecuencia de CPU, que en un procesador moderno salta entre el reloj de reposo y el turbo de una muestra a la siguiente. Una muestra cuesta alrededor de un milisegundo, así que el muestreo es insignificante frente al servidor SDR. La página consulta al ritmo de cada nivel: cada 2 segundos en los intervalos en vivo y cada 30 segundos en los promediados, porque preguntar más a menudo de lo que se generan puntos solo devuelve respuestas vacías.

Para cambiar la cadencia o la retención, edite estas constantes al principio del bloque del muestreador en `admin_server.py` y reinicie:

```
GRAPH_INTERVAL_S    = 2          # segundos entre muestras finas
GRAPH_FINE_S        = 3600       # alcance del nivel fino (1 h)
GRAPH_COARSE_EVERY  = 15         # muestras finas por punto agregado (15 x 2 s = 30 s)
GRAPH_COARSE_S      = 24 * 3600 # alcance del nivel agregado (24 h)
```

De dónde salen los números

Magnitud	Origen
Frecuencia de CPU	<code>psutil.cpu_freq()</code> , con respaldo en <code>/sys/devices/system/cpu/cpu*/cpufreq/</code> . Son los mismos datos del kernel que formatea <code>cpufreq-info</code> , por lo que cpufrequtils no es necesario .
Carga de CPU	<code>psutil.cpu_percent()</code>
Temperatura de CPU	<code>psutil.sensors_temperatures()</code> , con respaldo en <code>/sys/class/thermal</code> y después la orden <code>sensors</code>
Usuarios conectados	El propio endpoint <code>/users</code> de <code>spectrumserver</code> — la lista de sesiones autorizada. Contar sockets con <code>ss</code> no ve las IP reales de los clientes detrás de <code>proxy.py</code> .

Si una magnitud no está disponible en su máquina, ese panel lo indica en lugar de dibujar una línea plana a cero. Una lectura fallida interrumpe la línea, de modo que un hueco nunca parezca un valor medido.

Por qué cuatro paneles y no una sola gráfica

GHz, porcentaje, grados y un recuento de personas no comparten escala. Dibujarlos en una sola gráfica exige varios ejes y, los cruces que se producen son artefactos del escalado, no hechos sobre el servidor. Cuatro paneles apilados sobre un eje de tiempo mantienen la comparación honesta. Cada panel arranca en cero, así la altura de la curva sigue siendo proporcional al valor.

El color queda reservado para la temperatura, donde el ámbar y el rojo marcan los umbrales indicados. Los demás paneles comparten un color porque cada uno contiene una sola serie que su propio título ya nombra.

Endpoint

`GET /admin/api/graph-stats?since=<epoch>` (requiere inicio de sesión) devuelve las muestras posteriores a `since`, de modo que la página consulta de forma incremental en lugar de descargar todo el búfer en cada ciclo.

Thermal Guard

Manual completo: [THERMAL_GUARD.md](#) — los cuatro modos y qué debe hacer el sysop en cada uno, funcionamiento sin el panel, la fase `throttle` sin `root`, pruebas y resolución de problemas.

Detiene el servidor SDR cuando la CPU alcanza una temperatura peligrosa y vuelve a arrancarlo una vez fría. Forma parte del panel —no hay nada más que instalar— y aparece como tarjeta **THERMAL GUARD** en el Panel de control, con sus ajustes en **Ajustes**.

Se entrega **inerte**: el modo empieza en `log`, así que de fábrica solo registra lo que *habría* hecho. Nada toca su servidor hasta que usted cambie el modo.

Funciona con cualquier método de arranque/parada

La protección nunca intenta averiguar qué supervisa su servidor. Actúa mediante los scripts que haya fijado como **Default start script** y **Default stop script** en Ajustes, y mientras la CPU esté demasiado caliente repite la parada en **cada comprobación** (cada 2 s). Cualquier cosa que devuelva el servidor a la vida —el `watchdog` dentro de `start-*.sh`, una unidad de `systemd`, una tarea `cron`, una sesión de `tmux` — queda deshecha en un par de segundos, hasta que la máquina se enfríe. Eso es el *bloqueo* (lockout).

Si no hay script de parada configurado, envía `SIGTERM` al proceso indicado en `sdr_process_name` y `SIGKILL` tras 10 segundos de gracia.

Los umbrales salen de su propia CPU

En lugar de una cifra fija que sería errónea en la mayoría del hardware, la protección lee el límite crítico que el núcleo publica para su CPU y trabaja hacia atrás desde él:

Su CPU declara	aviso	throttle	parada	recuperación
Intel, límite 100 °C	88	92	95	75
AMD Ryzen, límite 95 °C	83	87	90	70
Raspberry Pi, límite 85 °C	73	77	80	60

Esto importa: unos 95 °C fijos son casi normales en un Intel, pero en un Ryzen —cuya Tctl se sitúa en 95 °C por diseño bajo boost— detendrían el servidor en una máquina sana, y en una Pi, que reduce a 80 °C, no se activarían nunca. Cualquier umbral puede fijarse a mano en Ajustes; deje el campo en blanco para *automático*.

Para ver cómo percibe la protección su máquina:

```
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
```

Las cuatro etapas

Detener el servidor es el último recurso, no la única herramienta:

1. **warn** — se registra, nada más.
2. **throttle** — baja un 20 % el techo de frecuencia de la CPU. Necesita un controlador `cpufreq` y permiso de escritura en `scaling_max_freq`, es decir, normalmente `root`; donde no lo hay (lo habitual en un panel sin privilegios y en la mayoría de VPS) la etapa **se desactiva sola** y la protección pasa de `warn` directamente a `stop`. `./setup-cpufreq-perms.sh` concede ese permiso de escritura a un grupo `cpufreq`, de modo que la etapa funciona sin `root` — véase [THERMAL_GUARD.md](#).
3. **stop** — ejecuta su script de parada y mantiene el servidor abajo (véase el bloqueo).
4. **resume** — cuando la CPU lleva `thermal_resume_s` (5 minutos por defecto) por debajo de la temperatura de recuperación, se restaura el techo de frecuencia y, en modo `stop+restart`, se ejecuta de nuevo su script de arranque.

Cada etapa debe **mantenerse** durante su tiempo — 30 s para `warn` y `throttle`, 60 s para `stop` —, de modo que un pico breve durante una compilación nunca la dispare.

Las activaciones repetidas están limitadas por `thermal_max_stops_hour` (2 por defecto). Alcanzado ese límite se desactiva el *reinicio automático* mientras la protección sigue funcionando, para que una máquina que se sobrecalienta no oscile entre funcionar y estar parada.

Qué sensores se usan

Solo sensores reales del die de la CPU: `coretemp` (Intel), `k10temp` / `zenpower` (AMD), `cpu_thermal` / `soc_thermal` (ARM, Raspberry Pi). Se toma la lectura más alta del chip, no la media.

acpitz y las zonas térmicas sin etiquetar se **ignoran a propósito**: a menudo informan de la temperatura de la caja o de la placa, decenas de grados por debajo de la CPU real, así que una protección que se fiara de ellas nunca actuaría cuando hiciera falta. Si su máquina no tiene un sensor utilizable (habitual en un VPS o dentro de un contenedor), la tarjeta lo dice claramente (`no trusted CPU sensor`) y la protección queda inactiva en lugar de fingir que le protege.

Qué escribe

Todo va a `crash.log`, es decir, a la pestaña **CRASH** del Visor de registros, un evento por línea:

```
[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-rx888mk2.sh
```

La línea de bloqueo se limita a una por minuto, para que un supervisor que se resista continuamente no inunde el registro.

Pruébalo antes de confiar en él

Ponga **TEST TEMPERATURE** en Ajustes por encima de su umbral de parada. La protección lo trata como una lectura real, así que todo el recorrido —warn, throttle, stop, bloqueo, recuperación, reinicio— se ejecuta cuando usted quiera. Vacíe el campo para volver al sensor real.

Si hace esta prueba con el modo en `stop`, el servidor se detiene de verdad y sus oyentes se desconectan. Hágala cuando no haya nadie, o deje el modo en `log`, donde la prueba muestra lo que *habría* pasado sin tocar nada.

Forma recomendada de adoptarlo

1. Deje el modo en `log` durante una semana y siga con su rutina.
2. Lea la pestaña CRASH después de los momentos duros: una recompilación completa, una tarde calurosa. Si no aparece nada, su máquina nunca se acercó.
3. Si las temperaturas registradas tienen buen aspecto, ponga el modo en `stop` (o `stop+restart` para que vuelva solo).
4. Compruébelo una vez con el campo de temperatura de prueba y después déjelo estar.

Salir de un bloqueo

CLEAR LOCKOUT en la tarjeta del Panel de control borra el bloqueo y el límite de repeticiones. No desarma la protección: si la CPU sigue demasiado caliente, la siguiente comprobación simplemente volverá a detener el servidor. Úselo después de haber arreglado la refrigeración.

Endpoint

`GET /admin/api/thermal` (requiere sesión) devuelve todo el estado de la protección: sensor, umbrales, etapa, bloqueo y último evento. Un `POST` con `{"action": "reset"}` hace lo mismo que el botón CLEAR LOCKOUT.

Sin el panel de administración

`thermal_guard.py` solo usa la biblioteca estándar y funciona por su cuenta, leyendo el mismo `admin_config.json`:

```
python3 thermal_guard.py --once          # mostrar sensor, límite crítico y umbrales
python3 thermal_guard.py                # ejecutarlo como servicio (solo registro hasta
armarlo)
python3 thermal_guard.py --mode stop+restart # armarlo sin escribir ningún archivo de configuración
python3 thermal_guard.py --config /path/to/other.json
python3 thermal_guard.py --help
```

--mode prevalece sobre `thermal_mode` del archivo de configuración, así que una máquina sin panel puede armarse desde su propia unidad de systemd, sin JSON alguno.

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

La pestaña **Spot Reporting** controla el demonio autorun (`autorun/index.js`), un proceso Node.js que decodifica FT8/FT4/JS8/WSPR en el servidor, directamente del receptor, y sube los spots a las redes de reporte:

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

JS8 se reporta solo a velocidad Normal — el ciclo de llamada de 15 s, donde están los heartbeats y las CQ. Los spots proceden de heartbeats, tramas compound y mensajes dirigidos; los destinos de grupo como @ALLCALL y los indicativos que el decodificador no pudo resolver (`<...>`) nunca se reportan. JS8 comparte la cola de PSK Reporter con FT8 y FT4 y se contabiliza por separado, igual que ellos.

El reporte está DESACTIVADO por defecto. No se sube nada hasta que active un destino y pulse **Start**. Decodificar y subir son procesos independientes: el demonio puede decodificar y registrar con el reporte desactivado, de modo que pueda comprobar la actividad antes de que nada se haga público.

⚠ Los spots se suben a redes públicas con **su indicativo**. Active únicamente las bandas y modos que su receptor oiga de verdad, e indique el localizador correcto.

Requisitos previos

El demonio autorun necesita Node.js 22+, los paquetes npm `ws + cbor-x` (resueltos mediante el enlace simbólico `autorun/node_modules → frontend/node_modules`) y `util-linux (taskset)`. Los instaladores lo preparan todo automáticamente; véase la [sección Autorun Spot Reporter de INSTALLATION.md](#). Si **Start** falla, la causa habitual es ese enlace simbólico o `taskset` (véase la resolución de problemas más abajo).

Uso de la pestaña

1. **Identidad** — indicativo y localizador. Se rellenan previamente desde `frontend/site_information.json` (`siteSysop / siteGridSquare`, recortado a 6 caracteres); modifíquelos aquí si es necesario.
2. **Matriz banda × modo** — marque las combinaciones que desea decodificar. Las filas son bandas y las columnas FT8 / FT4 / JS8 / WSPR; las celdas no admitidas están desactivadas. WSPR cubre además las bandas de LF/MF (2200 m, 630 m) y un canal adicional de 80 m para Europa.
3. **Destinos** — active **PSK Reporter** y/o **wsprnet** (ambos desactivados por defecto).
4. **Max slots** — límite de seguridad sobre cuántas ranuras banda/modo pueden ejecutarse a la vez.
5. **Start / Stop** — lanza o detiene el demonio (fijado a los núcleos de CPU superiores con `taskset`, elegidos automáticamente según la máquina; véase *Requisitos de recursos y limitaciones* más abajo). **Save / Reload** guardan y releen la configuración; **Free All** vacía todas las ranuras.

La tarjeta de estado se actualiza cada 5 s y muestra el estado de ejecución, los contadores de decodificación/subida y la hora de la última subida. Aparece un distintivo **REPORTING** en la cascada principal mientras el reporte está activo.

Ver «0 sent» durante los primeros minutos es normal. PSK Reporter sube por lotes cada **5 minutos** y wsprnet cada **2 minutos**: los spots esperan en cola hasta el siguiente envío. La tarjeta de estado muestra el número pendiente y la cadencia.

Cambiar bandas o modos mientras está en marcha

Los cambios no surten efecto hasta que se reinicia el demonio. `autorun/index.js` lee `autorun.json` una sola vez, al arrancar. No hay vigilancia del archivo ni señal de recarga: las únicas señales que atiende son SIGINT y SIGTERM, que significan apagar. Un demonio en marcha sigue por tanto decodificando las ranuras con las que se lanzó, guarde usted lo que guarde después.

Para aplicar un cambio:

1. Marque / desmarque las bandas y modos
2. **SAVE CONFIG**
3. **STOP**
4. **START**

O simplemente **STOP** → cambiar las casillas → **START**, ya que START guarda la configuración antes de lanzar.

⚠ **Pulsar START sin parar antes no aplica el cambio.** START guarda la configuración y después la petición de arranque responde `already running` y no se relanza nada. Verá un aviso de «guardado» junto a un error «already running» mientras el demonio continúa con las bandas **antiguas**: es fácil tomarlo por un éxito. Pare siempre primero.

Qué hace un reinicio con los contadores:

- **Los totales junto a cada casilla se conservan:** están en `autorun-totals.json`, que no se borra al apagar.
- **Las tarjetas por decodificador vuelven a cero**, son por ejecución.
- **No se pierde nada de la cola:** el apagado envía los spots pendientes.
- Una banda **recién activada** empieza en 0; una banda ya **usada antes** retoma su total anterior.
- Una banda que **desmarque conserva su número** junto a la casilla ahora vacía; el historial no se borra.

La decodificación solo se detiene los pocos segundos entre STOP y START.

Contadores de spots

Se muestran dos recuentos distintos, que responden a preguntas distintas.

SPOTS UPLOADED PER DECODER — una tarjeta por decodificador (FT8 / FT4 / JS8 / WSPR) con los spots subidos **desde el último arranque del demonio** y, debajo, el número de decodificaciones y cuántos siguen en cola. Stop/Start los pone a cero. Un decodificador cuyo destino está desactivado muestra sus decodificaciones y `reporting off` en lugar de un 0 a secas, porque ahí cero subidas es un ajuste, no un fallo.

El número junto a cada casilla en BANDS & MODES son los spots subidos **de todo el tiempo** para esa banda y modo. Se guardan en `autorun-totals.json` y sobreviven a los reinicios. **En ámbar y con un punto delante** (·123) significa que esa ranura ha decodificado pero aún no ha subido nada. Es normal entre envíos —PSK Reporter sube cada 5 minutos y wsprnet cada 2—, así que los contadores de FT8/FT4

quedan en ámbar los primeros minutos tras un arranque. También sigue en ámbar si el destino de ese decodificador está apagado. Un 0 a secas, sin punto, significa que esa ranura está activada pero aún no ha decodificado nada: WSPR se queda en cero varios minutos tras un arranque, porque funciona en un ciclo de 2 minutos mientras FT8 ya cuenta por centenares. Una ranura que el demonio nunca ha ejecutado no muestra nada. Al pasar el ratón se ven las subidas, las decodificaciones y la hora de la última subida.

FT8 y FT4 comparten una única cola de subida a PSK Reporter, así que el demonio imputa cada spot a su propio modo y banda en el momento del envío; el reparto no se estima después a partir de los totales.

Poner a cero los contadores. FREE ALL SLOTS desmarca todas las bandas y modos, apaga ambos destinos, detiene el demonio y **borra los contadores históricos**: después no se muestra ningún número junto a ninguna casilla. Es la única forma de reiniciarlos y no se puede deshacer. El botón detiene el demonio y espera a que termine antes de borrar `autorun-totals.json`, porque el demonio reescribe ese archivo al apagarse; borrarlo con el demonio en marcha simplemente devolvería los números.

Requisitos de recursos y limitaciones

El demonio es deliberadamente ligero y funciona en hardware modesto (un i5 de cuatro núcleos basta), pero conviene conocer sus límites reales.

La fijación de CPU de autorun tiene en cuenta la configuración y es automática. El demonio **autorun** siempre lo lanza el panel de administración (`admin_server.py`), así que su fijación se establece en cada instalación sin configuración alguna: deriva un rango de núcleos para `taskset` a partir del número de CPU, reservando algunos de los núcleos superiores para la decodificación, o se ejecuta sin fijación con ≤ 4 núcleos.

La fijación de spectrumserver depende de su propio método de arranque. La forma de lanzar spectrumserver varía entre instalaciones (tipo de SDR, herramientas, scripts personales), así que este documento no presupone ningún lanzador concreto. El panel de administración no arranca ni fija spectrumserver: eso depende por completo del comando o servicio que use para ejecutarlo. Si quiere mantener spectrumserver fuera de los núcleos que usa el demonio autorun, fíjelo usted mismo con `taskset` en su propio comando de arranque (véase la sección de sustitución más abajo). Si no lo hace, simplemente se ejecuta sin fijación y el planificador del sistema lo equilibra: es correcto y seguro, solo pierde la separación deliberada de núcleos.

Como referencia, el demonio autorun reserva estos núcleos superiores (así que, si fija spectrumserver, manténgalo en los inferiores para evitar solapamientos):

CPU lógicas	Usa el demonio autorun	Dejar para spectrumserver
≤ 4	<i>sin fijar</i>	<i>sin fijar</i>
6	5	0-4
8	6-7	0-5
12 (p. ej., híbrido 8P+4E)	8-11	0-7
16	12-15	0-11

En una máquina de **4 núcleos (o menos)** no hay nada que separar, así que el demonio autorun también se ejecuta *sin fijar* y comparte todos los núcleos: es correcto y seguro, pero la FFT del SDR y las ráfagas de decodificación compiten por los mismos núcleos.

Limitaciones conocidas:

1. **El verdadero cuello de botella es spectrumserver y el ancho de banda del SDR, no el demonio.** Un RX888 a 30 MHz necesita OpenCL/GPU; una CPU con pocos núcleos y sin una GPU

capaz no puede sostener esa FFT. Combine el hardware modesto con un SDR más estrecho (RSP1A $\approx$ 10 MHz, RTL-SDR $\approx$ 2.4 MHz).

2. **WSPR es lo que más CPU consume.** Su decodificador de Fano es un port a JS ($\sim$ 30 s por banda, de un solo hilo). El grupo de 4 workers ejecuta 4 decodificaciones en paralelo; activar **más de ~4 bandas de WSPR** a la vez puede hacer que la cola supere la ranura de 120 s, sobre todo mientras el SDR compite por los núcleos. Las decodificaciones de FT8/FT4 son baratas en comparación.
3. **Escalado del número de ranuras.** La cifra de $\sim$ 2-2,5 núcleos / 600-700 MB corresponde a las $\sim$ 28 ranuras completas en una máquina de 8 núcleos. Con 4 núcleos compartidos, límitese a unas pocas bandas (orientación: $\leq$ 6 FT8/FT4 + $\leq$ 3 WSPR) y vigile la estadística `queued` del grupo.
4. **La fijación presupone una topología híbrida de Intel** (núcleos bajos = P-cores más rápidos). En AMD o en CPU con numeración SMT intercalada, el reparto automático sigue siendo válido (sin solapamientos ni fallos), pero «los núcleos bajos son más rápidos» puede no cumplirse literalmente; en un chip 4C/8T con hyperthreading, los núcleos superiores son hermanos SMT: quedan acotados, no del todo aislados. Cuando el rango automático no sea el idóneo, use la **sustitución manual** que se describe abajo.
5. **La cobertura de bandas la limita el receptor.** La configuración del RX888 (`sps=600000000` $\rightarrow$ Nyquist de 30 MHz) no llega a 6 m ni por encima; la tabla de bandas va de 160 m a 10 m. Otros SDR cubren lo que permita su ventana sintonizada.
6. **Cadencia de reporte.** PSK Reporter envía cada 5 min y wsprnet cada 2 min: ver «0 sent» los primeros minutos es normal, no un fallo.

Sustitución manual de la fijación de CPU (avanzado)

El reparto automático de núcleos es adecuado para la mayoría de las máquinas, pero puede forzar una fijación concreta donde no lo sea (CCX/CCD de AMD, ARM big.LITTLE, SMT intercalado). Hay dos sustituciones independientes, cada una acepta una lista de núcleos al estilo `taskset -c (2-3, 0,2,4, 0-2,5)` o `none/off/unpinned` para desactivar la fijación. Un valor no válido se ignora y se aplica la derivación automática: una errata nunca podrá impedir un arranque.

Demonio autorun — variable de entorno `AUTORUN_CORES`, o un campo `"cores"` en `autorun.json` (el entorno tiene prioridad). El campo `"cores"` se conserva al pulsar **Save** en el panel, de modo que una edición manual permanece. Dos formas de establecerlo:

Opción A — `autorun.json` (persistente, recomendada). Añada una línea `"cores"` a la configuración en la raíz del repositorio y luego haga **Stop $\rightarrow$ Start** en la pestaña Spot Reporting:

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

Opción B — *variable de entorno (puntual)*. El demonio lo lanza el panel de administración, así que la variable debe estar en el entorno **del panel**: defínala y reinicie el panel (el entorno gana sobre el valor de `autorun.json`):

```
# Método A — la variable debe estar en el proceso que arranca:
./manage_admin.sh stop && AUTORUN_CORES=2-3 ./manage_admin.sh start

# Método B — systemd ignora una variable puesta en la línea de órdenes
# de systemctl, así que colóquela en la unidad:
sudo systemctl set-environment AUTORUN_CORES=2-3
sudo systemctl restart phantomsdr-admin
```

spectrumserver — fíjelo usted mismo anteponiendo `taskset -c <núcleos>` al comando con el que arranque el servidor. Funciona con cualquier método de arranque (lanzamiento directo, canalización de SDR, unidad de servicio, etc.):

```
# 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
```

⚠ **Haga que sobreviva a los reinicios.** Un `taskset` en línea solo se aplica a ese arranque concreto. Si algo reinicia el servidor automáticamente (un watchdog, un servicio systemd, una entrada de cron/@reboot, ...), añada el prefijo `taskset` dentro del script o la unidad que realmente lo arranca; de lo contrario, el reinicio pierde la fijación.

Compruebe que la fijación surtió efecto — muestre la afinidad de CPU real de los procesos en ejecución:

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

Valores admitidos (en ambas sustituciones): una lista `taskset -c (2-3, 0,2,4, 0-2,5)`, o `none/off/unpinned` para no fijar. Cualquier valor mal formado se ignora y se aplica la derivación automática, así que una errata nunca podrá bloquear el arranque.

En una Raspberry Pi o cualquier equipo de ≤ 4 núcleos no suele hacer falta ninguna de las dos: la vía automática ya ejecuta ambos procesos sin fijar, que es lo correcto en una CPU pequeña y homogénea. La sustitución está pensada para máquinas mayores que no sean híbridos de Intel.

Archivos y puntos finales

Elemento	Finalidad
<code>autorun.json</code>	Configuración guardada (identidad, ranuras, destinos, máximo de ranuras, sustitución opcional <code>cores</code>). La escribe la pestaña; git la ignora.
<code>autorun-status.json</code>	Estado en directo que lee la tarjeta de estado (pid, contadores, última subida). Se escribe cada 15 s; se elimina al parar.
<code>frontend/dist/autorun-active.json</code>	Datos públicos del distintivo, servidos en <code>/autorun-active.json</code> . Vacío cuando el reporte está desactivado.
<code>autorun.log</code>	Salida estándar y de error del demonio.
GET/POST <code>/admin/api/autorun/config</code>	Leer / escribir <code>autorun.json</code> (valida indicativo, localizador, combinaciones banda/modo y límite de ranuras).
GET <code>/admin/api/autorun/status</code>	Estado de ejecución (pgrep) + <code>autorun-status.json</code> .
POST <code>/admin/api/autorun/start / stop</code>	Lanza (<code>taskset -c <auto> node autorun/index.js</code>) / envía SIGTERM. El rango de núcleos se deriva de la CPU (véase más arriba).

Nota: tras actualizar `admin_server.py` debe **Reiniciar el panel** para cargar las nuevas rutas de autorun o una matriz banda/modo actualizada. El demonio se ejecuta como un proceso aparte, así que sobrevive a los reinicios del panel de administración.

Resolución de problemas

Problema	Solución
El panel de administración no arranca	Consulte <code>admin.log</code> : suele faltar un paquete de Python
El proxy no arranca	Ejecute <code>python3 -c "import aiohttp"</code> ; si falla: <code>pip3 install aiohttp --break-system-packages</code>
<code>proxy.log</code> está vacío	Compruebe primero que el proxy realmente se está ejecutando (<code>./manage_admin.sh status</code>). Si es así, tiene una versión antigua: el lanzador debe usar <code>python3 -u</code> (sin búfer; de lo contrario el cartel de arranque nunca sale del búfer de 8 KB) y el <code>main()</code> de <code>proxy.py</code> debe llamar a <code>logging.basicConfig()</code> (si no, el registrador de accesos de <code>aiohttp</code> no tiene manejador y se descarta cada línea de petición, porque <code>web.AppRunner</code> —a diferencia de <code>web.run_app</code> — no configura el registro).
El proxy arranca pero la cascada aparece en blanco	Compruebe <code>sdr_host</code> en <code>admin_config.json</code> : normalmente <code>127.0.0.1</code> . Si todavía contiene una IP local antigua de una versión anterior, cámbiela y Reiniciar el panel
El botón Kick dice «no connections»	Ejecute <code>getcap \$(which ss)</code> ; si no aparece <code>cap_net_admin</code> , ejecute <code>sudo setcap cap_net_admin+ep \$(which ss)</code>
El estado siempre muestra OFFLINE	Vaya a Settings → SDR Process Name y ponga el nombre exacto que muestra <code>ps -eo comm,args \ grep -v grep</code>
La página Users no muestra datos	Verifique que el punto final funciona: <code>curl http://127.0.0.1:<public_port>/users</code>
Un navegador externo da NetworkError	Compruebe que el proxy está en marcha: <code>./manage_admin.sh status</code>
El proxy dejó de funcionar tras un cambio de red	Solo aplica si <code>sdr_host</code> todavía contiene una IP local: póngalo en <code>127.0.0.1</code> y Reiniciar el panel
El botón de borrar del chat no hace nada	Compruebe el permiso de escritura del archivo de registro del chat: <code>ls -l ~/PhantomSDR-Plus/chat.jsonl</code>
El mensaje de la cascada no aparece	Confirme que el proxy está en marcha (<code>./manage_admin.sh status</code>) y que el frontend es de una compilación reciente que incluye el renderizador de la superposición
El mensaje de la cascada se pierde tras reiniciar	Es lo previsto: el estado del mensaje solo está en memoria; vuelva a enviarlo tras reiniciar el panel
El «Start» de Spot Reporting falla / el demonio se cierra al instante	Consulte <code>autorun.log</code> . Suele faltar el enlace simbólico <code>autorun/node_modules</code> (<code>ln -sf ../frontend/node_modules autorun/node_modules</code>) o <code>taskset</code> no está instalado (<code>sudo apt install -y util-linux</code>).
Spot Reporting: el demonio funciona pero decodes sigue en 0 y <code>autorun.log</code> muestra 504 / tap closed 1006	La toma de audio no llega a <code>spectrumserver</code> . El demonio detecta el puerto automáticamente a partir de la configuración del servidor en ejecución ; la línea de registro <code>[autorun] tap backend: HOST:PORT</code> debe coincidir con su <code>[server] port</code> . Si es incorrecta (o el servidor no estaba en marcha al arrancar), fíjelo en <code>autorun.json</code> : <code>"server": { "host": "127.0.0.1", "port": 9002 }</code> , y después Stop→Start.
Spot Reporting muestra «0 sent»	Normal durante los primeros minutos: PSK Reporter envía cada 5 min y <code>wspnrt</code> cada 2 min. Confirme que hay un destino activado y que el demonio está en marcha.
Las bandas o modos nuevos no aparecen en la matriz	Reiniciar el panel — la matriz se carga al arrancar el panel.

Problema	Solución
autorun.log muestra <pre>MODULE_TYPELESS_PACKAGE_JSON /</pre> «Reparsing as ES module ... performance overhead»	Aviso cosmético (la decodificación sigue funcionando). A frontend/src/modules/package.json le falta "type": "module"; añada esa línea cerca del principio y haga Stop→Start. No hace falta recompilar el frontend: el demonio importa el archivo fuente directamente.
El «Start» de Spot Reporting falla con <pre>taskset: ... Invalid argument</pre> en un PC antiguo o modesto	No debería ocurrir con el lanzador que tiene en cuenta la configuración: el rango de núcleos se deriva del número de CPU y no se fija con ≤ 4 núcleos. Si sucede, su admin_server.py es anterior a ese cambio; actualícelo (_autorun_taskset_prefix) y Reiniciar el panel.
El distintivo REPORTING no aparece en la cascada	El reporte debe estar activado con al menos una ranura; el distintivo lee /autorun-active.json. Recompile el frontend si dist/index.html es anterior al distintivo.

CAPÍTULO 4

Thermal Guard

THERMAL_GUARD.md

Thermal Guard — Manual del sysop

Protección contra sobrecalentamiento de la CPU para PhantomSDR-Plus.

Un WebSDR es una de las pocas cosas que mantienen ocupados todos los núcleos, todo el día, para siempre. Si un ventilador se atasca, una bomba se para o la sala se calienta en agosto, nadie se lo dice: el servidor sigue decodificando hasta que la CPU se está cociendo. El thermal guard es la pieza que se da cuenta y desenchufa por usted.

Se entrega **inerte**: de fábrica solo escribe en un registro. Nada está protegido hasta que usted elija un modo. Este manual trata de tomar esa decisión y convivir con ella.

¿Con prisa? Ejecute `python3 thermal_guard.py --once`, lea los umbrales que imprime y ponga `thermal_mode` en `stop+restart` desde el panel de administración. Ese es todo el trabajo. Todo lo demás es el *porqué*.

Contenido

1. Qué hace
2. Inicio rápido
3. Cómo decide
4. Los cuatro modos — qué hace cada uno y qué debe hacer usted
5. Elegir el modo
6. Referencia de configuración
7. Funcionamiento sin el panel de administración
8. Activar la fase throttle sin root
9. Funcionamiento desatendido — cuando nadie mira
10. Probarlo antes de confiar en él
11. Leer el registro
12. Lockout y límite de frecuencia
13. Resolución de problemas
14. Desactivarlo o eliminarlo

1. Qué hace

Cada 2 segundos el guard lee la temperatura del die de su CPU y la compara con cuatro umbrales. Cuando la temperatura se mantiene por encima de uno de ellos el tiempo suficiente, sube un escalón:

```

normal → warn → throttle → stop → (enfriar) → restart
        línea en  bajar la  ejecutar el  el servidor  ejecutar el
        el registro frecuencia script de  sigue caído script de
                                   parada      arranque

```

Dos decisiones de diseño hacen que funcione en la máquina de *cualquier* sysop, y conviene entenderlas antes de confiar en él:

Nunca pregunta cómo ejecuta usted su servidor. systemd, un bucle watchdog, cron, tmux, un simple shell: el guard no lo sabe ni le importa. En su lugar, mientras la máquina está por encima de temperatura mantiene un **lockout**: repite la parada en *cada* ciclo. Si su watchdog reinicia el servidor, el guard lo vuelve a parar en dos segundos. Lo que sea que intente mantenerlo vivo pierde, hasta que la CPU se haya enfriado. Esta es la parte que lo hace universal, y está probada contra un watchdog real.

Nunca fija una temperatura por código. «Parar a 95 °C» es una mala regla: en un Intel con Tjmax 100 eso es casi carga normal, en un Ryzen queda *por debajo* de donde se sitúa Tctl con boost por diseño, y en una Raspberry Pi es inalcanzable porque el SoC se limita solo a 80. Así que el guard lee el punto crítico que publica su propio kernel (tempN_crit) y calcula hacia atrás. El mismo código, cifras razonables en todas partes.

2. Inicio rápido

1. Vea cómo percibe el guard su máquina. Siempre lo primero:

```
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
```

Si `sensor` dice `NONE`, deténgase aquí y lea la [Resolución de problemas](#): el guard no puede proteger una máquina que no puede medir, y se lo dice en lugar de fingir.

2. Déjelo unos días en modo log. Ya se está ejecutando dentro del panel de administración. Déjelo observar una semana normal — tardes de verano, una compilación grande, un sábado de concurso movido — y compruebe si habría actuado alguna vez.

3. Actívelo. Panel de administración → **Settings** → **THERMAL GUARD** → **MODE** → `stop+restart` → **SAVE THERMAL SETTINGS**. Surte efecto en unos 2 segundos; no hace falta reiniciar.

3. Cómo decide

El sensor

El guard solo confía en sensores reales del die de la CPU:

Fuente	Hardware típico
<code>coretemp</code>	Intel
<code>k10temp</code> , <code>zenpower</code> , <code>zenpower3</code>	AMD
<code>cpu_thermal</code> , <code>soc_thermal</code>	Raspberry Pi, placas ARM
Zona térmica <code>x86_pkg_temp</code>	alternativa en x86

acpitz y las zonas térmicas sin etiqueta se excluyen **a propósito**. En un equipo de sobremesa típico acpitz marca unos 28 °C mientras el paquete de la CPU está a 59 °C: un guard que confiara en eso sencillamente nunca actuaría. Informar de «desactivado, sin sensor fiable» es más honesto que fingir protección.

psutil se utiliza si resulta estar instalado, pero nunca es necesario. El guard usa solo la biblioteca estándar.

Los umbrales

Cada uno es un margen por debajo del punto crítico que publica la propia máquina:

Umbral	Margen bajo crit	Qué significa
warn	−12 °C	Algo va mal, regístrelo
throttle	−8 °C	Intentar enfriar bajando la frecuencia
stop	−5 °C	Demasiado cerca del límite: parar el servidor
resume	−25 °C	Realmente frío otra vez, se puede volver

Que en la práctica queda así:

Su CPU informa	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

Si la máquina no publica ningún punto crítico, el guard recurre a un conjunto fijo conservador y lo indica. Puede sobrescribir cualquier umbral con un valor absoluto (véase la [Referencia de configuración](#)), pero no debería hacer falta.

Tiempos de sostenimiento — por qué un pico nunca actúa

No pasa nada con una sola lectura caliente. Un umbral debe mantenerse **de forma continua**:

- **warn** y **throttle**: 30 s (`thermal_warn_sustain_s`)
- **stop**: 60 s (`thermal_sustain_s`)
- **resume**: 300 s por debajo del umbral de resume antes de un re arranque (`thermal_resume_s`)

Un pico de compilación, una ráfaga de decodificación, un ventilador acelerando: todo demasiado breve para contar. Una sola lectura por debajo del umbral reinicia el reloj. Además, tras arrancar el guard exige 3 lecturas válidas consecutivas antes de actuar, y descarta cualquier valor fuera de 20–125 °C como fallo del sensor y no como temperatura.

4. Los cuatro modos

`thermal_mode` es el único interruptor que decide hasta dónde puede subir el guard por la escalera. Cada modo **incluye todo lo que hacen los anteriores**.

Modo	Avisa	Limita	Para	Rearranca	¿Puede caerse el servidor?
log	✓	—	—	—	No — no toca nada
throttle	✓	✓	—	—	No
stop	✓	✓	✓	—	Sí, sigue caído hasta que actúe
stop+restart	✓	✓	✓	✓	Sí, vuelve por sí solo

log — observar e informar (el valor de fábrica)

Qué hace: todo lo que hacen los demás modos, salvo actuar. Recorre toda la escalera y escribe en `crash.log` lo que *habría* hecho. Nunca ejecuta su script de parada, nunca toca la frecuencia de la CPU, nunca interrumpe a un oyente.

Qué debe hacer usted: nada — ya está funcionando. Al cabo de una semana, abra la pestaña **CRASH** del panel o ejecute:

```
grep THERMAL crash.log
```

- **¿No hay nada?** Su refrigeración está bien y puede activar el guard con confianza.
- **¿STOP level reached ... NOT stopping?** El guard habría apagado su servidor. Eso es un problema térmico real con el que ha estado funcionando: arregle la refrigeración y active el guard.

Úselo cuando: acaba de instalar el guard y aún no sabe cómo se comporta su máquina. Es un modo de rodaje, no un destino — un guard en `log` no protege nada.

throttle — bajar la CPU, sin interrumpir nunca el servicio

Qué hace: en el umbral de throttle reduce un 20 % la frecuencia máxima de la CPU y la restaura cuando la temperatura vuelve por debajo de la línea de aviso. Su WebSDR sigue en pie todo el tiempo. Los oyentes pueden notar algo menos de rendimiento con carga alta; la mayoría no notará nada.

Si aun así la temperatura sigue subiendo y supera el umbral de parada, este modo **lo registra y no hace nada**: no parará el servidor.

Qué debe hacer usted:

1. Dar al guard permiso de escritura sobre el límite de frecuencia de la CPU — lo necesita, y de fábrica solo lo tiene root. Ejecute `./setup-cpufreq-perms.sh` una vez; véase la [sección 8](#).
2. Confirmar con `python3 thermal_guard.py --once` que imprime `cpufreq : writable, throttle stage available`. Si dice *not writable*, este modo no hace nada más allá de lo que hace `log`, y lo indicará cada vez en el registro.

Úselo cuando: persiga un problema de refrigeración al límite y quiera defender la máquina sin perder nunca oyentes. **Tenga en cuenta que no es protección completa:** si limitar la frecuencia no basta, no ocurre nada más. Buen primer paso, mala respuesta final.

stop — apagar el servidor y dejarlo apagado

Qué hace: en el umbral de parada ejecuta su script de parada y mantiene el **lockout**: vuelve a parar el servidor cada 2 segundos mientras la máquina esté por encima de temperatura, de modo que nada lo reviva a sus espaldas. Cuando la CPU lleve 5 minutos por debajo del umbral de resume, el lockout se libera. **El servidor no vuelve solo.** Lo rearranca usted cuando esté convencido de que la causa está resuelta.

Qué debe hacer usted:

1. Asegurarse de que el guard sabe cómo parar su servidor. En el panel es el script de parada configurado (`./stop-websdr.sh` con los lanzadores incluidos). Si no hay ninguno, el guard recurre a `SIGTERM` sobre el nombre del proceso y luego a `SIGKILL` tras 10 segundos de gracia — funciona, pero un script de parada propio es más limpio.
2. Aceptar que **su WebSDR estará fuera de línea hasta que se dé cuenta**. Tenga alguna forma de enterarse: una monitorización de disponibilidad, la pestaña CRASH, o simplemente mirar a diario.

Úselo cuando: la máquina importa más que el servicio, suele estar usted cerca, o un episodio térmico es lo bastante serio como para querer inspeccionar el equipo antes de que siga. También es la elección correcta si no se fía de un rearranque automático en su hardware.

stop+restart — apagar, enfriarse y volver solo

Qué hace: todo lo de `stop`, y además: una vez que la CPU lleva 5 minutos por debajo del umbral de resume, ejecuta su script de arranque y el WebSDR vuelve por sí mismo.

El vaivén se evita con `thermal_max_stops_hour` (2 por defecto). Tras dos paradas térmicas en una hora, el guard **desactiva el rearranque automático** y deja el servidor caído para que usted lo revise. Fíjese en el detalle importante: el límite desactiva el *rearranque*, nunca la *protección* — el lockout sigue parando el servidor mientras esté caliente, por muchas veces que ya haya saltado.

Qué debe hacer usted:

1. Asegurarse de que **ambos** scripts están configurados y funcionan por sí solos, tanto el de arranque como el de parada. Pruébelos a mano: `./stop-websdr.sh` y luego `./start-rx888mk2.sh`.
2. Nada más. Este es el modo de ponerlo y olvidarse.

Úselo cuando: el WebSDR sea público, desatendido o remoto, que es la mayoría de los casos. Es el modo recomendado para una instalación normal.

5. Elegir el modo

Su situación	Modo
Acaba de instalar el guard, aún no conoce la máquina	log durante una semana
WebSDR público, desatendido, que se cuide solo	stop+restart
Emplazamiento remoto de difícil acceso	stop+restart
Quiere inspeccionar la máquina tras cada episodio térmico	stop
Hardware al que no confía un rearranque automático	stop
Refrigeración al límite, no puede perder oyentes	throttle, y después stop+restart
Máquina sin sensor de CPU fiable	ninguno funcionará — arregle antes el sensor

Para la mayoría de los sysops la respuesta es `stop+restart`, y el resumen honesto de los demás es: `log` no protege nada y `throttle` protege un poco.

6. Referencia de configuración

Todas las claves están en `admin_config.json`, junto a `thermal_guard.py`, y todas ellas se pueden editar también en **Settings → THERMAL GUARD**. El guard **relee el archivo en cada ciclo**, así que los cambios surten efecto en ~2 segundos, sin reiniciar.

Clave	Por defecto	Significado
<code>thermal_enabled</code>	<code>true</code>	Interruptor principal. <code>false</code> desactiva el guard por completo.
<code>thermal_mode</code>	<code>"log"</code>	<code>log</code> <code>throttle</code> <code>stop</code> <code>stop+restart</code> — véase la sección 4 .
<code>thermal_warn</code>	<code>null</code>	Umbral de aviso absoluto en °C. <code>null</code> = derivar del punto crítico de la CPU.
<code>thermal_throttle</code>	<code>null</code>	Umbral de throttle absoluto. <code>null</code> = derivado.
<code>thermal_stop</code>	<code>null</code>	Umbral de parada absoluto. <code>null</code> = derivado.
<code>thermal_resume</code>	<code>null</code>	Umbral de resume absoluto. <code>null</code> = derivado.
<code>thermal_sustain_s</code>	<code>60</code>	Segundos que debe mantenerse el umbral de parada antes de parar. Mínimo 5.
<code>thermal_warn_sustain_s</code>	<code>30</code>	Segundos para las fases de aviso y throttle. Mínimo 5.
<code>thermal_resume_s</code>	<code>300</code>	Segundos por debajo del umbral de resume antes de liberar el lockout. Mínimo 30.
<code>thermal_max_stops_hour</code>	<code>2</code>	Paradas térmicas por hora tras las cuales se desactiva el re arranque automático. La protección continúa.
<code>thermal_test_temp</code>	<code>null</code>	Finge que la CPU está a esta temperatura. Solo para pruebas — véase la sección 10 .

El guard autónomo usa además `start_script`, `stop_script`, `sdr_process_name` y `sdr_base_dir` del mismo archivo.

Sobrescribir un umbral. Hágalo solo con motivo. Uno frecuente: su máquina se calienta legítimamente durante las compilaciones y prefiere que eso no cuente. Subir `thermal_stop` le da margen a costa de la distancia al punto crítico: nunca lo ponga por encima de su valor crit, o la CPU alcanzará antes su propio apagado de emergencia y el guard quedará como adorno.

7. Funcionamiento sin el panel de administración

Muchos sysops no instalan nunca el panel. El guard funciona perfectamente por su cuenta: el mismo archivo, el mismo comportamiento.

Pruébalo primero en primer plano:

```
cd ~/PhantomSDR-Plus
python3 thermal_guard.py --once          # ¿qué pasaría aquí?
python3 thermal_guard.py --mode log      # verlo en vivo, Ctrl-C para salir
```

`--mode` sobrescribe `thermal_mode` desde la línea de órdenes, así que nunca tiene que escribir JSON a mano:

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

Después hágalo permanente con la unidad de ejemplo incluida en el repositorio:

```
sudo cp thermal-guard.service /etc/systemd/system/
sudo nano /etc/systemd/system/thermal-guard.service # ajuste User= y las rutas
sudo systemctl daemon-reload
sudo systemctl enable --now thermal-guard
systemctl status thermal-guard
```

Lea los comentarios del principio de `thermal-guard.service` antes de activarla. Dos cosas son lo más importante:

- **User= debe ser una cuenta capaz de ejecutar realmente sus scripts de arranque y parada.** Ejecutar el guard como root suele ser la respuesta equivocada: también ejecutaría esos scripts como root.
- Apúntelo a un `admin_config.json` que contenga `start_script` y `stop_script`, o pase `--mode` en la línea `ExecStart`. Basta con una configuración mínima:

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

Todo lo que hace el guard sigue yendo a `crash.log` junto a ese archivo de configuración, además del journal de systemd (`journalctl -u thermal-guard`).

8. Activar la fase throttle sin root

Por defecto, `python3 thermal_guard.py --once` informa:

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

Es normal y no es un fallo. El kernel crea

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

propiedad de `root:root`, con modo `0644`. Un panel de administración que corre como usuario normal no puede escribir ahí, así que el guard se salta la fase de throttle y lo dice en el registro en lugar de fallar en silencio.

La solución equivocada es ejecutar el guard como root. También ejecuta sus scripts de arranque y parada, y esos deben seguir sin privilegios.

La solución correcta es dar permiso de escritura sobre esos archivos concretos a un grupo dedicado. Un script lo hace:

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

Pide su contraseña una vez — y solo una, durante la instalación — y a continuación:

1. crea un grupo de sistema `cpufreq` y le añade a él;
2. instala `/etc/tmpfiles.d/99-phantomsdr-cpufreq.conf` para que el grupo y el modo `0664` se vuelvan a aplicar **en cada arranque**: los permisos de `sysfs` no sobreviven a un reinicio por sí solos;
3. aplica el cambio de inmediato, para que pueda probar sin reiniciar.

Después, en este orden:

```
# 1. cierre la sesión y vuelva a entrar (o reinicie): un grupo nuevo solo
# llega a sus procesos a través de un inicio de sesión nuevo
id | grep cpufreq

# 2. reinicie el panel; el guard comprueba el permiso una vez, al arrancar
sudo systemctl restart phantomsdr-admin
# (así no hace falta cerrar sesión – systemd reconstruye la lista de
# grupos en cada arranque. Solo si las unidades NO están instaladas:
# ./manage_admin.sh stop && ./manage_admin.sh start – nunca ambos, pelean por el puerto 3000)

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

Para deshacerlo por completo: `sudo ./setup-cpufreq-perms.sh --revoke`.

Notas

- La fase de throttle solo *actúa* si `thermal_mode` es `throttle` o superior. Conceder el permiso no cambia nada por sí solo.
- El guard baja un 20 % el límite vigente en ese momento y restaura exactamente ese valor. Si ya limita su CPU (por ejemplo con `MAX_SPEED` en `/etc/init.d/cpufrequtils`), lo que se restaura es su límite: el guard no le devolverá a escondidas una frecuencia más alta de la que pidió.
- Algunas máquinas no exponen ningún controlador `cpufreq`: muchos VPS y casi todos los contenedores. El script lo detecta y no cambia nada. Allí use `stop` o `stop+restart`, que no dependen de `cpufreq`.

9. Funcionamiento desatendido

La pregunta más frecuente sobre el guard: *¿qué pasa cuando nadie mira?* La respuesta corta es que se construyó precisamente para ese caso.

No necesita un navegador abierto. El guard es un hilo en segundo plano dentro de `admin_server.py`, cadenciado por el muestreador de gráficas cada 2 segundos. Funciona haya o no alguien conectado al panel, haya o no un navegador abierto, esté usted despierto o no. En modo autónomo es un servicio de `systemd`, aún más independiente. La tarjeta del panel es una *ventana* al guard, no el guard.

Actúa sin preguntar. No hay diálogo de confirmación ni notificación que esperar. En el umbral de parada detiene el servidor, y punto. De eso se trata exactamente.

Pero solo si usted le dio permiso. En modo `log` registrará fielmente cómo una máquina se cuece y no hará nada al respecto. Si lee esta sección porque «*no estoy para vigilarlo*», el modo que quiere es `stop+restart`: es el único que protege la máquina y además devuelve el servicio sin usted.

Solo ayuda mientras el panel siga vivo. El guardián forma parte de `admin_server.py`: un panel que murió a las 02:00 se lleva la protección con él, y tras un reinicio la máquina queda desprotegida hasta que usted entre y lo arranque de nuevo. Si no hay nadie vigilando, deje que vigile `systemd`: el repositorio incluye `phantomsdr-admin.service` y `phantomsdr-proxy.service`, que arrancan al inicio y reinician el panel

a los cinco segundos de un fallo. Véase [ADMIN_PANEL_SETUP.md](#). En un receptor desatendido esto se **recomienda encarecidamente**: armar `stop+restart` sin ello solo le protege hasta el siguiente reinicio.

Cómo se entera después. Por orden aproximado de utilidad:

Dónde	Qué obtiene
<code>crash.log</code> / pestaña CRASH	Cada acción, con marca de tiempo. El registro permanente.
Tarjeta THERMAL GUARD del panel	Estado en vivo: temperatura, fase, lockout, paradas de la última hora.
Página Graphs (temperatura de CPU, 24 h)	La forma del episodio: cuán rápido subió, cuánto tardó en enfriarse.
<code>journalctl -u thermal-guard</code>	Solo en modo autónomo.

Una rutina sensata para un emplazamiento desatendido: `stop+restart` activo y `grep THERMAL crash.log` cada vez que entre por cualquier motivo. Si está vacío, no hay nada que saber.

Un ejemplo real. Fallo de ventilador a las 03:00 en una máquina en `stop+restart`:

```
03:14 la temperatura pasa de 88 °C, se mantiene 30 s → aviso registrado
03:16 pasa de 92 °C, se mantiene 30 s → throttle aplicado (si está permitido)
03:19 pasa de 95 °C, se mantiene 60 s → STOP – script de parada, lockout activo
03:19 el watchdog rearranca el servidor → el guard lo vuelve a parar y lo registra
03:26 la CPU lleva 5 minutos por debajo de 75 °C → lockout liberado, script de arranque
03:41 se sobrecalienta y para por segunda vez → límite alcanzado, no más rearranques
                                         el servidor queda caído, la protección
sigue
```

Se despierta con un WebSDR caído, un `crash.log` que dice exactamente por qué y, sobre todo, una CPU que nunca se acercó a su punto crítico.

10. Probarlo antes de confiar en él

No espere a una ola de calor real para descubrir si su script de parada funciona.

`thermal_test_temp` hace que el guard crea que la CPU está a la temperatura que usted elija. Todo lo demás se comporta con total normalidad: los tiempos de sostenimiento, la escalera, el lockout, sus scripts reales.

1. Panel → **Settings** → **THERMAL GUARD** → **TEST TEMPERATURE**.
2. Introduzca un valor por encima de su umbral de parada (por ejemplo 97 en un Intel).
3. Guarde y observe la tarjeta del panel y la pestaña CRASH.
4. **Vacíe el campo al terminar.** Una temperatura de prueba olvidada significa que el guard ya no lee el sensor real.

En modo `log` esto le muestra toda la escalera sin arriesgar nada: la primera prueba segura, y la que conviene hacer antes de activar nada.

En modo `stop+restart` esto es una **prueba en vivo**: su servidor se parará de verdad y volverá de verdad unos cinco minutos después de vaciar el campo. Hágalo cuando no haya nadie escuchando. Merece la pena hacerlo una vez, porque es la única manera de saber que sus scripts de parada y arranque funcionan cuando los invoca algo que no es usted.

Equivalentes en modo autónomo:

```
python3 thermal_guard.py --once          # solo umbrales, sin actuar
python3 thermal_guard.py --mode log      # ver la escalera en vivo
```

11. Leer el registro

Todo se añade a `crash.log`, junto al archivo de configuración, un evento por línea y con el prefijo `[THERMAL]`:

```
grep THERMAL crash.log
```

Las líneas son deliberadamente de una sola línea y aptas para grep. Lo que verá:

Línea	Significado
warn: 88.4C (warn=88.0 crit=100) sustained 30s	Primer escalón. No se ha hecho nada.
throttle: 92.1C sustained 30s – cpufreq max lowered 20%	Frecuencia de CPU reducida.
throttle level reached: ... (no cpufreq write access – stage skipped)	Véase la sección 8 .
STOP level reached: ... mode is 'log', NOT stopping	Habría parado el servidor. Actívalo.
STOPPED server: 95.2C sustained 60s (stop=95.0 crit=100) – ran stop-websdr.sh	El caso real.
lockout: process reappeared at 96.0C – re-stopped (...) [12 time(s) so far]	Algo está rearmando su servidor; el guard gana. Limitado a una línea por minuto.
rate limit: 2 thermal stops within the hour – automatic restart is now DISABLED	Queda caído para que investigue. La protección continúa.
recovered: 54.0C held below 75.0 – lockout cleared	Frío de nuevo.
auto-restart: ran start-rx888mk2.sh	De vuelta en línea.
back to normal: 59.0C (warn=88.0)	Bajó de la escalera sin llegar a parar.

12. Lockout y límite de frecuencia

Estos dos comportamientos son los que más sorprenden, así que conviene decirlos claramente.

El lockout es lo que hace que el guard funcione precisamente contra *su* instalación. Cuando para el servidor no se limita a enviar una parada y confiar. Se marca como bloqueado y, en cada ciclo de 2 segundos mientras la temperatura siga por encima del umbral de resume, comprueba si el proceso ha vuelto — y si es así, lo vuelve a parar. Su watchdog, el `Restart=always` de systemd, una tarea de cron, un sysop impaciente: todos pierden esa discusión hasta que la CPU se enfría. Se libera solo cuando la temperatura se ha mantenido por debajo del umbral de resume durante `thermal_resume_s` (5 minutos por defecto).

El botón CLEAR LOCKOUT de la tarjeta del panel libera el lockout y el contador de paradas por hora, para cuando ha arreglado la refrigeración y no quiere esperar. **No** desarma el guard: si la máquina sigue caliente, la comprobación siguiente parará el servidor otra vez. Eso es intencionado.

El límite de frecuencia (`thermal_max_stops_hour`, 2 por defecto) evita que una máquina averiada esté subiendo y bajando toda la noche. Alcanzado el límite, el guard **desactiva el rearranque automático** y deja el servidor caído. Léalo con atención: desactiva el rearranque, no la protección. El lockout sigue

parando el servidor mientras esté por encima de temperatura, por muchas veces que ya haya saltado. Un guard que se callara justo cuando la máquina está peor sería peor que no tener guard.

13. Resolución de problemas

sensor : NONE – no trusted CPU sensor on this machine El guard no encontró ningún sensor de la lista permitida y se niega a adivinar. Pruebe `sensors` (de `lm-sensors`; `sudo apt install lm-sensors && sudo sensors-detect`). Dentro de una máquina virtual o un contenedor a menudo no hay realmente un sensor de die expuesto: el guard no puede proteger esa máquina y se declara desactivado en lugar de fingir.

cpufreq : not writable – throttle stage disabled Es lo esperado si no ha ejecutado `./setup-cpufreq-perms.sh` — véase la [sección 8](#). Solo afecta a la fase de throttle; `stop` y `stop+restart` no se ven afectados.

Ejecuté `setup-cpufreq-perms.sh` y sigue diciendo not writable Dos causas probables: no ha cerrado la sesión y vuelto a entrar desde entonces (un grupo nuevo solo llega a los procesos mediante un inicio de sesión nuevo; compruébelo con `id`), o el panel no se ha reiniciado desde entonces (el guard comprueba el permiso una vez, al arrancar). Haga ambas cosas, en ese orden.

Los umbrales parecen incorrectos para mi CPU Compruebe lo que publica la máquina: `cat /sys/class/hwmon/hwmon*/temp*_crit`. Si su crit es inusual, o no existe, ponga valores absolutos en `thermal_stop` y compañía.

Paró mi servidor y no creo que estuviera caliente Compruebe si `thermal_test_temp` sigue puesto de alguna prueba. Es, con mucho, la causa más común.

Nunca actúa aunque la máquina se calienta Confirme que el modo no es `log`, que `thermal_enabled` es `true`, y compare la temperatura que ve con los umbrales de `--once`. Recuerde que el umbral debe mantenerse *de forma continua* durante el tiempo de sostenimiento.

El servidor no para de rearmar durante un episodio térmico Eso es el lockout funcionando, y las líneas `lockout` son su parte de éxito: su supervisor sigue intentándolo y el guard sigue deshaciéndolo. No hay nada que arreglar.

El panel no está en marcha, ¿está funcionando el guard? No: en modo panel el guard vive dentro de `admin_server.py`. Si quiere protección independiente del panel, use el servicio autónomo de la [sección 7](#).

14. Desactivarlo o eliminarlo

- **Pausarlo:** ponga `thermal_mode` en `log`. Sigue observando e informando, pero nunca actúa.
- **Desactivarlo del todo:** ponga `thermal_enabled` en `false`.
- **Autónomo:** `sudo systemctl disable --now thermal-guard`.
- **Deshacer los permisos de cpufreq:** `sudo ./setup-cpufreq-perms.sh --revoke`.

Borrar `thermal_guard.py` también es seguro: `admin_server.py` lo importa dentro de un `try` y simplemente informa de que el guard no está disponible si falta.

Véase también

- [Instalación del panel de administración](#) — la pestaña Thermal Guard en su contexto
- [Instalación](#) — protección térmica durante una instalación nueva

- [Estructura del proyecto](#) — dónde está `thermal_guard.py`

CAPÍTULO 5

Límites de conexión

CONNECTION_LIMITS.md

Límites de conexión — Manual del sysop

Proteger un receptor público frente a avalanchas de conexiones.

Un WebSDR es un servicio público que entrega a quien lo pida un flujo continuo de audio y espectro. Ese es su propósito, y ahí está también el problema: para el servidor, nada distingue a un oyente normal de alguien que abre sesiones en bucle. El 16 de septiembre de 2026 este receptor sufrió exactamente eso — una sola dirección abrió 134 sesiones de escucha en un minuto y mantuvo 181 a la vez, mientras el público real de la estación rondaba la docena.

Este manual trata de las tres capas que lo detienen, de lo que cada una puede y no puede hacer, y de cómo saber si las suyas están bien ajustadas.

¿Con prisa? El `config.toml` de ejemplo ya viene con valores sensatos, así que una instalación nueva está protegida sin que haga usted nada. Si su propio `.toml` es anterior y no tiene claves `[limits]` más allá de `audio/waterfall/events`, los límites por dirección están simplemente **desactivados** — copie el bloque del ejemplo para activarlos. El único comando que conviene conocer después es `grep Refused spectrumsrvr.log`, que le dice si se está rechazando a alguien.

Contenido

1. De qué se está defendiendo realmente
2. Las tres capas
3. Límites por dirección — la política de oyentes
4. Conexiones inactivas — el ataque silencioso
5. La protección en el núcleo — `setup-firewall.sh`
6. Referencia de configuración
7. Qué ve un oyente rechazado
8. Leer los registros y los contadores
9. Elegir sus números
10. Trampas que conviene conocer
11. Desactivarlo todo

1. De qué se está defendiendo realmente

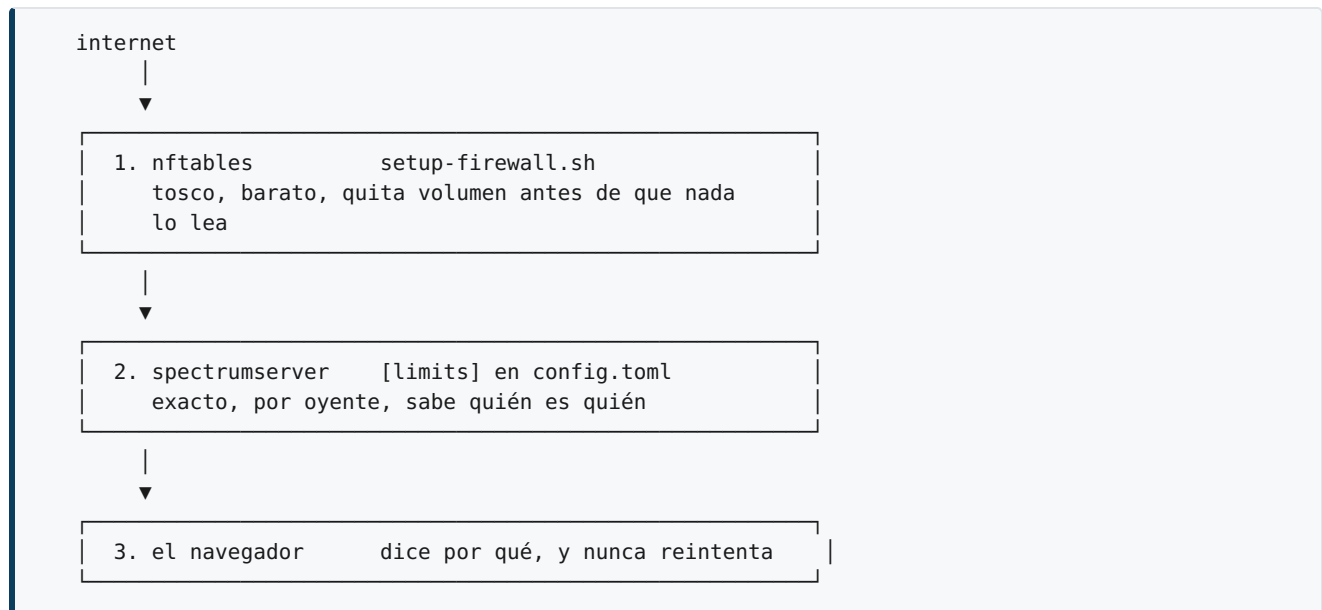
Conviene ser preciso, porque la palabra «DDoS» cubre dos cosas muy distintas y solo una de ellas puede resolverse en su propia máquina.

Avalancha desde una sola fuente. Un equipo, o unos pocos, abriendo conexiones tan rápido como pueden. Es lo que ocurrió aquí, y tiene arreglo completo: las conexiones llegan de una dirección que se puede contar, y contar basta.

Un ataque distribuido de verdad. Miles de equipos, a menudo con direcciones de origen falsificadas, llenando su enlace. Cuando esos paquetes llegan a su tarjeta de red, el ancho de banda ya se ha gastado. **Nada de lo descrito en este manual ayuda**, ni puede ayudar nada que instale en el servidor — la única respuesta es un servicio por delante de usted, lo que es otra conversación que implica un dominio real y un proveedor como Cloudflare delante del receptor.

Todo lo que sigue aborda el primer caso. No es una limitación de la que haya que disculparse: el primer caso es lo que realmente les ocurre a las estaciones de aficionado.

2. Las tres capas



No son redundantes entre sí, y es deliberado. La capa del núcleo es rápida pero tonta: cuenta paquetes de una dirección y no sabe nada de oyentes. La capa del servidor es precisa, pero solo ve una conexión una vez aceptada y analizada, lo cual cuesta algo. La capa del navegador existe para que un oyente rechazado sepa qué ha pasado, en lugar de mirar una página que nunca carga.

3. Límites por dirección

Qué cuenta como un oyente

Un oyente es **una conexión /audio** — o, donde `[kiwi_emulation]` está activado, **una conexión de sonido Kiwi**, que cuenta exactamente igual. Es la misma unidad que ya usan el contador de usuarios, `users.json` y las etiquetas sobre la cascada, así que un cliente Kiwi consume una plaza y aparece en la lista de usuarios igual que un navegador.

Esto importa porque una sola pestaña de navegador de escritorio abre **cuatro** conexiones a su servidor:

Conexión	Función
/audio	el flujo de audio demodulado — este es el oyente
/waterfall	la representación del espectro
/events	lista de usuarios, frecuencias de los demás oyentes
/chat	el panel de chat

Un teléfono en `/mobile` abre dos (`/audio` y `/chat`), porque no tiene cascada. Así que un límite que contara conexiones en bruto rechazaría a todo el mundo: tres pestañas son doce conexiones.

Las tres claves

```
[limits]
per_ip=3          # oyentes simultáneos desde una dirección
per_ip_sockets=0  # conexiones simultáneas de cualquier tipo; 0 da per_ip * 4 + 4
per_ip_rate=40    # conexiones nuevas por minuto desde una dirección
per_ip_ban_s=600  # cuánto se rechaza a una dirección que rompe el ritmo
```

`per_ip` es la cifra principal y en la que pensará. Tres es un buen punto de partida.

`per_ip_sockets` existe porque `/audio` no es lo único que merece la pena inundar. `/waterfall` es el flujo más pesado que produce su servidor, y un límite que contara solo oyentes se esquivaría sin más abriendo conexiones de cascada. Dejado en 0 se convierte en `per_ip * 4 + 4`, que da cabida a `per_ip` pestañas de escritorio completas con algo de margen para conexiones que el núcleo aún no ha liberado tras recargar la página.

`per_ip_rate` es lo que de verdad termina con una avalancha. El tope por sí solo no lo hace: rechazar una conexión es barato pero no gratis, y un atacante puede seguir reconectando tan rápido como vuelven los rechazos, quemando su CPU en montar y desmontar. El límite de ritmo le da un bloqueo en lugar de una respuesta. Recuerde que una pestaña cuesta cuatro, así que 40/minuto son unas diez cargas de página por minuto desde una dirección.

Un orden deliberado. El ritmo se cobra *antes* de comprobar el tope de simultáneas. Parece al revés hasta que se piensa en una dirección que ya está en su tope y reconecta en bucle: si el rechazo barato de «estás en el tope» le respondiera siempre, las infracciones de ritmo nunca se acumularían y el bloqueo que termina el bucle no llegaría nunca.

Quién queda exento

El loopback nunca se limita. El tap de autorun para el reporte de spots, el panel de administración y un navegador abierto en la propia máquina del servidor llegan todos desde `127.0.0.1`, y dejar que consuman plazas significaría que su propia estación deja fuera a sus oyentes.

Esto cubre también `proxy.py`: las conexiones a través del proxy llegan a `spectrumserver` desde loopback, y la dirección real del cliente llega en la cabecera `X-Forwarded-For`, que es sobre la que se aplican los límites.

Desde la v5.0.0 esa cabecera se cree **solo si viene de un par loopback** — un proxy en la misma máquina. Enviada por cualquier otro se ignora y cuenta la dirección TCP, porque de lo contrario un visitante directo podría escribir `X-Forwarded-For: 127.0.0.1` y pasar como loopback por encima de todos los límites de esta página. `proxy.py` además descarta cualquier `X-Forwarded-For` o `X-Real-IP` que envíe un cliente antes de añadir los suyos, y solo responde a la expulsión del sysop (`/~kick`) para clientes de la misma máquina. Una estación con varios receptores detrás del proxy se describe en [Varios receptores](#); allí los límites se aplican por receptor.

4. Conexiones inactivas

Hay un ataque que nada de lo anterior puede ver.

Abrir una conexión TCP. No enviar nada. Mantenerla.

Ese interlocutor nunca completa una petición, así que nunca alcanza el código donde viven los límites por dirección — no hay oyente, no hay objeto de cliente, no hay nada con nombre que contar. Mientras

tanto, a él le cuesta unos pocos bytes y a usted un descriptor de fichero y una plaza mientras la mantenga.

```
[limits]
idle_per_ip=8      # conexiones inactivas (previas a la petición) por dirección
idle_total=512     # tope para todas las direcciones juntas
```

El límite está en **cuántas**, no en **cuánto tiempo**, y esa distinción es todo el diseño. **No hay plazo alguno para una conexión inactiva, y es deliberado.** Un plazo parece la respuesta obvia y es una trampa (vea §10): hay interlocutores legítimos que se conectan y luego permanecen callados mucho tiempo antes de hablar. Lo que los separa de un ataque es la cantidad — un interlocutor legítimo mantiene una de esas conexiones, un atacante miles.

Una conexión inactiva vive por tanto hasta que hace una petición o muere, y su plaza se libera en ambos casos.

`idle_total` acota una avalancha repartida entre muchas direcciones de origen. Al alcanzar el tope, se rechazan las **nuevas** conexiones inactivas en vez de expulsar las existentes, de modo que una conexión legítima y de larga vida nunca es lo que se sacrifica para hacer sitio.

Ambos se aplican en el momento en que se acepta el socket, usando la dirección del interlocutor — todavía no hay petición, así que no hay `X-Forwarded-For` que consultar. No pasa nada: el loopback no se cuenta, lo que cubre todo lo que llega por `proxy.py`, y una avalancha directa trae su propia dirección.

5. La protección en el núcleo

`setup-firewall.sh` instala una tabla de nftables que quita volumen antes de que `spectrumserver` lea un solo byte. Es deliberadamente tosca, y sus números quedan muy por encima de los de `config.toml`, de modo que un oyente normal nunca puede quedar atrapado por ambos.

```
./setup-firewall.sh --show      # imprime las reglas, no cambia nada
sudo ./setup-firewall.sh --check # valida contra su núcleo
sudo ./setup-firewall.sh --apply # carga, con reversión automática a los 60 s
sudo ./setup-firewall.sh --persist # recarga al arrancar
sudo ./setup-firewall.sh --status # los contadores de paquetes por regla
sudo ./setup-firewall.sh --remove # deshacer todo
```

Cubre cuatro cosas: un tope de conexiones simultáneas por dirección de origen en sus puertos de proxy y receptor, un ritmo de conexiones nuevas por origen, un freno contra la fuerza bruta en SSH, y la compartición de archivos de Windows cerrada a todo lo que esté fuera de los rangos privados — `smbd` escucha en `0.0.0.0` en muchas máquinas, y que eso sea alcanzable depende de un router que el script no puede ver.

Los puertos se leen de `admin_config.json`, el mismo fichero que usa `proxy.py`, así que sigue su instalación.

Por qué no puede dejarle fuera

Dos propiedades, ambas deliberadas:

- La política de la tabla es **accept**, y solo descarta patrones concretos y nombrados. Una regla que no casa deja pasar el paquete a lo que sea que tenga configurado. No puede dejar la máquina incomunicada.
- **Las conexiones establecidas se aceptan en la primerísima regla.** La sesión SSH en la que está escribiendo nunca se ve afectada por nada de lo que sigue.

Además, `--apply` arma una reversión automática: carga las reglas y luego espera confirmación en 60 segundos, o las quita. No diga nada, cierre el terminal, pierda la conexión — el conjunto de reglas desaparece solo.

Aproveche bien esa ventana. Compruebe **desde otro dispositivo, fuera de su propia red**, que el receptor sigue cargando y que se abre una sesión SSH *nueva*. Probar con la sesión que ya tiene no demuestra nada, porque la aceptó la regla uno.

6. Referencia de configuración

Todas las claves de abajo viven bajo `[limits]` en su fichero `.toml`. **Todas vienen desactivadas o con valores generosos**, así que una configuración que no las mencione se comporta exactamente como siempre.

Clave	Por defecto	Significado
<code>per_ip</code>	0 (desactivado)	Oyentes simultáneos por dirección
<code>per_ip_sockets</code>	$0 \rightarrow \text{per_ip} * 4 + 4$	Conexiones simultáneas de cualquier tipo por dirección
<code>per_ip_rate</code>	0 (desactivado)	Conexiones nuevas por minuto y dirección
<code>per_ip_ban_s</code>	600	Segundos de rechazo tras romper el ritmo
<code>idle_per_ip</code>	8	Conexiones inactivas, previas a la petición, por dirección
<code>idle_total</code>	512	Conexiones inactivas en todas las direcciones

Poner cualquier clave a 0 desactiva esa comprobación concreta.

Las tres que no son límites

`[limits]` contiene además `audio`, `waterfall` y `events`, heredadas del PhantomSDR original, y **ninguna de las tres se aplica**. `waterfall` y `events` no las lee ningún código. `audio` se lee en un único sitio: se envía como `max_users` en el JSON de registro a los directorios de `[websdr] register_urls`, así que es la cifra que muestran `sdr-list.xyz` y los demás, y no hace absolutamente nada cuando `register_online=false`. Si quiere un tope de oyentes, `per_ip` de más arriba es el que actúa.

Una más, bajo `[server]`

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

No es un límite sino un rechazo, y como comparte el mismo código de cierre, su sitio está aquí. Las páginas anuncian la versión desde la que se cargaron como `/audio?v=N`, y todo lo que esté por debajo de `min_client_version` se rechaza con «*esta página está desactualizada — recárguela*».

Existe porque el servidor no puede alcanzar el JavaScript que ya se está ejecutando en el navegador de alguien. Una pestaña conserva el código que cargó hasta que alguien la recarga, así que un cambio en la página solo llega a los oyentes que por casualidad recargan — y una estación que acaba de cambiar el comportamiento de las sesiones puede necesitar que las páginas antiguas desaparezcan ya, no algún día.

Solo se comprueba `/audio`, porque es donde vive una sesión, y el loopback y las rutas Kiwi quedan exentos para que el tap de autorun y los clientes Kiwi no se vean afectados. 0 lo desactiva y es el valor por defecto. Dejarlo activo una vez agotadas las páginas antiguas no supone problema: un cliente de diversidad actual al que se rechaza en un `/audio` sin más vuelve a intentarlo con el marcador, de modo

que solo se sigue rechazando a clientes de diversidad con una versión anterior de PhantomSDR-Plus y a herramientas de terceros que abren `/audio` sin `?v=`. Vuelva a ponerlo en `0` si otra estación le informa de que ya no puede usarle como pareja de diversidad.

7. Qué ve un oyente rechazado

Nada se reconecta nunca

Empiece por aquí, porque gobierna todo lo demás: **una conexión `/audio` caída termina la sesión**. No hay reconexión automática, y es deliberado.

Reconectar es el instinto correcto para una aplicación de chat y el equivocado para un receptor. `/waterfall` y `/events` nunca volvían con ella, así que una sesión reintentada era una conexión de audio viva atornillada a una cascada congelada; y detener el servidor ya no lo vaciaba, porque todos los oyentes estaban de vuelta unos segundos después, llenando la lista de usuarios con sesiones en las que en realidad no había nadie. La página se queda en silencio y el oyente la vuelve a cargar — que es justo lo que hacía este receptor antes de que se añadiera reconexión alguna.

Para los límites de este manual eso elimina además una trampa: frente a un límite de ritmo, un reintentado automático alargaría precisamente el bloqueo que intentaba sortear.

Los dos códigos de cierre

Código	Significado	Quién lo envía
4003	rechazado — por encima de un límite por dirección, o página obsoleta	spectrumserver
4001	expulsado por el sysop	spectrumserver, cuando el panel llama a <code>/~~kick</code>

Ambos están en el rango privado 4000–4999, así que ninguno puede confundirse con un estado de protocolo, y ambos se tratan como definitivos en todas las páginas. El texto del motivo viaja en la trama de cierre y no como mensaje, porque la primera trama de `/audio` está reservada a los ajustes del receptor y cualquier cosa enviada antes rompería toda conexión normal.

En la página

Rechazado al cargar la página — el oyente ni siquiera arranca, así que ambas páginas se explican. El escritorio muestra un panel: «*Este receptor ha rechazado la conexión*», el motivo, y la nota de que todos los que comparten una conexión a internet cuentan como una dirección. `/mobile` muestra lo mismo en su zona de avisos.

Rechazado o expulsado a mitad de sesión — la página de escritorio simplemente se detiene, deliberadamente sin mensaje: el bucle de dibujo de la cascada se para y nada más. `/mobile` muestra una línea diciendo al oyente que recargue, salvo tras una expulsión, donde también guarda silencio.

Ese silencio tras una expulsión es intencionado. **Expulsar es el sysop terminando una sesión, no un castigo** — no se anuncia nada, y quien quiera volver carga la página otra vez y es un oyente corriente un segundo después. Lo único que aporta el código 4001 es que el navegador sepa que aquello no fue una conexión caída, de modo que la expulsión surta efecto en lugar de deshacerse sola.

De la expulsión se encarga spectrumserver mismo, no proxy.py: los oyentes se conectan directamente al puerto del receptor, así que el proxy no posee ninguna de sus conexiones y su expulsión no alcanza a nadie. El panel llama a `/~~kick?ip=&secs=`, accesible solo por loopback y comprobado contra el interlocutor TCP real, no contra X-Forwarded-For, que lo envía el cliente. `secs` vale cero por defecto —

sin bloqueo — pero acepta una duración si quiere dejar la dirección fuera un rato, y eso reutiliza el mismo mecanismo de bloqueo que `per_ip_rate`, de modo que la reconexión se rechaza en la puerta.

8. Leer los registros y los contadores

El servidor

```
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
```

Si ahí no hay nada, no se está rechazando a nadie. El registro está limitado a como mucho una línea por dirección cada cinco segundos — sin eso, un ataque se convertiría en escrituras a disco sin límite, que no es más que una forma más lenta de tumbar la estación.

El núcleo

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

Lea el **counter packets N** de cada regla; ese es el número realmente descartado. Todo a cero significa que no se ha bloqueado nada y que sus umbrales están cómodamente holgados.

No se alarme por las listas `elements = { ... }` de los conjuntos que hay sobre las reglas. Esos son sus oyentes normales siendo *seguidos* respecto a los límites — una entrada aparece en la primera conexión de una dirección y caduca un par de minutos después. Seguir no es bloquear.

9. Elegir sus números

No hace falta adivinar; sus propios registros se lo dirán. Esta es la medición que fijó aquí `per_ip=3`, sobre una semana de `logs/users_*.jsonl`:

Sesiones simultáneas desde una dirección	Número de direcciones
1	682
2	54
3	13
más de 3	8

758 direcciones distintas en siete días, y un límite de tres habría afectado a ocho de ellas. La avalancha estaba en 181.

Para repetirlo en su propia estación, empareje los identificadores de sesión por dirección en `logs/users_*.jsonl` y tome el solapamiento máximo. Tenga en cuenta que el registro anota eventos `tune` y `disconnect` pero no tiene evento `connect`, así que el inicio de una sesión debe deducirse de su primer `tune`.

Recuerde el NAT. Un club, un colegio, una oficina o un operador móvil aparecen como una sola dirección, y todos los que están detrás comparten un cupo. Esto es inherente a cualquier límite basado en la dirección; la única mitigación es elegir un número con el que se sienta cómodo. Si alguna vez un

oyente informa de que no puede conectar, esto es lo primero que hay que sospechar — mire `grep Refused` antes que nada.

10. Trampas que conviene conocer

Nunca un plazo para una conexión inactiva

Ya no existe la clave `handshake_s`, y este es el motivo. Un plazo de 30 segundos para conexiones inactivas parece la defensa obvia frente a un `slowloris`. Se construyó, se activó y en un minuto produjo esto:

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

192.87.173.88 es `etgd-websdr.ewi.utwente.nl` — **el propio equipo de retorno de `websdr.org`**. Se conecta de vuelta a su servidor y luego permanece inactivo, bastante más de 30 segundos, antes de enviar su `GET /~~orgstatus`. El plazo lo cortaba cada vez y el registro dejó de funcionar. La opción se eliminó en lugar de dejarla disponible: un disparo en el pie que rompe su ficha en el directorio no merece las sesenta líneas que lo implementan.

El mismo razonamiento vale para `open_handshake_timeout` de `websocketpp`, que sigue ahí. El valor de diez minutos en `spectrumserver.cpp` parece absurdo y es estructural: es lo que permite que el retorno de `websdr.org` pueda estar inactivo. Hay una segunda razón — el manejador de `/~~orgstatus` responde desde un hilo que mantiene un `dup()` del socket, y `websocketpp` termina una conexión caducada con `shutdown()`, una llamada a nivel de socket que atraviesa el duplicado y rompería el retorno aunque el tiempo de inactividad no fuera un problema.

Si necesita acotar las conexiones silenciosas, acote su número, no su edad. Eso es lo que son `idle_per_ip` e `idle_total`.

Un fichero de `nftables` debe ser idempotente

`nft -f` **añade** a una tabla que ya existe en lugar de reemplazarla. Un fichero de reglas sin un preámbulo `delete table` duplica por tanto toda la cadena en cada recarga — que es justo lo que hace una unidad de `systemd` ingenua en cada reinicio del servicio. `setup-firewall.sh` genera el preámbulo por usted; si escribe sus propias reglas, haga lo mismo.

```
sudo ./setup-firewall.sh --status | grep -c dport # debe dar 8, no 16
```

11. Desactivarlo todo

Los límites del servidor: ponga las claves a `0`, o bórrelas de su `.toml`, y reinicie el receptor. No hay más estado; nada sobrevive a un reinicio.

El cortafuegos:

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

Eso borra la tabla, elimina el conjunto de reglas guardado y desactiva el servicio de arranque. Su máquina vuelve exactamente al estado en que estaba antes — lo que, en la mayoría de instalaciones, significa sin cortafuegos alguno. Merece la pena pensarlo antes de quitarlo.

CAPÍTULO 6

Varios receptores

MULTI_RECEIVER.md

Varios receptores — Manual del sysop

Dos o más receptores en un mismo ordenador, con un selector de receptor en la página.

Desde la v5.0.0 un ordenador PhantomSDR-Plus puede hacer funcionar varios receptores a la vez — por ejemplo un RX-888 para HF y un RTL-SDR para 2 m — y los oyentes pasan de uno a otro con botones en la cabecera de la página, igual que OpenWebRX ofrece sus perfiles. Cada receptor conserva su propia cascada, chat, marcadores, lista de oyentes y datos de estación; nadie en un receptor resulta nunca molestado por alguien en otro.

¿Prisa? Una estación con un solo receptor no necesita nada de esto y no nota ningún cambio. Para un segundo receptor: cree `instances/<nombre>/` con su propio `config.toml`, arránquelo con `INSTANCE=<nombre> ./start-<radio>.sh`, anote ambos receptores en `receivers.toml` y reinicie el proxy. El ejemplo de la [sección 3](#) lo recorre línea a línea.

La manera más fácil: ejecute `./add-receiver.sh` (los instaladores lo ofrecen al final). Pregunta qué receptor y qué debe cubrir, instala su controlador, escribe todo lo descrito en la [sección 3](#) y la [sección 4](#), y ofrece arrancar el receptor y reiniciar el proxy. El resto de este manual explica lo que hace, por si quiere cambiar algo a mano.

Contenido

1. Cómo funciona
2. Dos maneras de publicar los receptores
3. Añadir un segundo receptor
4. `receivers.toml`
5. Arrancar y detener
6. Lo que ven los oyentes
7. El S-meter por encima de 30 MHz
8. Seguridad
9. Cuántos receptores caben
10. Trampas que conviene conocer

1. Cómo funciona

Cada receptor es un `spectrumserver` completo y propio, con su propio SDR, su propia configuración y su propio puerto interno. Nunca comparten muestras, ajustes ni oyentes. Lo que los une es `proxy.py`, el proxy inverso que ya sirve el panel de administración: lee `receivers.toml` y envía cada petición a un receptor.

```

oyentes → proxy.py → { :8900 spectrumserver RX-888 (HF, el receptor principal)
                        127.0.0.1:9002 spectrumserver RTL-SDR (2 m, instancia "vhf")

```

El proxy decide con estos datos, en este orden:

1. `?rx=<id>` en la dirección — `http://su.host:8899/?rx=vhf`. La página lo añade a cada socket, consulta y enlace que abre, de modo que una pestaña se queda en su receptor.
2. La cookie `rx`, que el proxy fija cada vez que una petición lleva un `?rx=` válido.
3. La cabecera `Host`, si un receptor tiene ese nombre en `hostnames`.
4. Si no, el receptor por defecto.

El **receptor principal** es el que se arranca de la forma habitual (`./start-rx888mk2.sh`). Cualquier otro receptor es una **instancia con nombre**: se arranca con `INSTANCE=<nombre>`, se configura en `instances/<nombre>/` y tiene su propio registro (`logwebsdr-<nombre>.txt`), registro del servidor, bloqueo del watchdog y FIFO. Los procesos se distinguen por una etiqueta `PHANTOMSDR_INSTANCE` en su entorno, no por su nombre, así que dos receptores pueden ejecutar exactamente el mismo programa — dos `rtl_sdr`, o `rx_sdr` para un RSP1A y un Airtyspy — y reiniciar uno nunca afecta al otro.

2. Dos maneras de publicar los receptores

A — el receptor principal conserva su propio puerto. El receptor principal se queda exactamente donde estaba (por ejemplo `:8900`), sus oyentes no notan nada, y a los demás receptores se llega por el puerto del proxy (`proxy_port` en `admin_config.json`, por ejemplo `:8899`) con `?rx=`. No hay que mover nada ni nada sale del aire; a cambio hay dos puertos públicos.

```

HF: http://su.host:8900/
2 m: http://su.host:8899/?rx=vhf

```

B — un solo puerto público para todo. El proxy se queda con el puerto público y el receptor principal pasa a uno interno. Defina `[front] port` en `receivers.toml`, mueva el `[server] port` del receptor principal y ponga en él `host = "127.0.0.1"`, ponga en `public_port` de `admin_config.json` el nuevo puerto interno y añada `public_port = <el puerto público>` bajo `[websdr]`, para que el directorio siga anunciando el correcto. Así todo queda detrás de un puerto; el precio es un breve corte durante el cambio.

```

HF: http://su.host:8900/
2 m: http://su.host:8900/?rx=vhf

```

El selector de receptor funciona igual en ambos casos. La disposición A es el comienzo más seguro; la B puede llegar después sin cambiar nada del segundo receptor.

3. Añadir un segundo receptor

add-receiver.sh hace todos los pasos siguientes — ejecútelo y responda a sus preguntas; también instala el controlador (`setup-rtlsdr.sh` para un RTL-SDR, el `setup-*.sh` correspondiente para los demás), elige un puerto interno libre y nombra el receptor según lo que cubre. Los pasos se detallan aquí para que pueda comprobar su trabajo o cambiarlo más tarde.

El ejemplo añade un RTL-SDR Blog V4 para 2 m como instancia `vhf`, junto a un RX-888 que conserva el puerto 8900 (disposición A).

1. El controlador. El RTL-SDR Blog V4 necesita el controlador de RTL-SDR Blog en lugar del paquete `rtl-sdr` de la distribución, y el controlador DVB-T del núcleo debe soltar el dispositivo. `install.sh` (opción de receptor 2, «RTL-SDR Blog V4: sí») hace ambas cosas. Compruébelo con `rtl_test -t`, que debe mostrar RTL-SDR Blog V4 Detected.

2. La carpeta de la instancia. Todo lo que pertenece al receptor vive en `instances/vhf/`, que nunca se sube al repositorio y que ninguna actualización sobrescribe:

```
instances/vhf/
├── config.toml          la configuración de su spectrumserver (obligatoria)
├── instance.env         opcional: argumentos del receptor y núcleos de CPU
├── markers.json        sus propios marcadores
├── www/                sus propias copias de los archivos de estación de la página
│   ├── site_information.json
│   └── wf-message.json
```

`spectrumserver` trabaja con `instances/vhf/` como directorio de trabajo, de modo que el historial del chat, los marcadores, el FFTW wisdom y `logs/` también son suyos.

3. config.toml. Parta de `config-rtl.toml` y cambie:

```
[server]
port=9002                # su propio puerto interno
host="127.0.0.1"         # solo accesible a través del proxy
html_root="www/"         # sus pocos archivos propios...
html_fallback_root="../../frontend/dist/" # ...y la página común para el resto

[input]
sps=2400000              # igual que -s en instance.env
frequency=145000000       # igual que -f en instance.env

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

[kiwi_emulation]
enabled = false          # vea las trampas más abajo
```

`html_fallback_root` permite que la instancia guarde solo sus propios `site_information.json` y `wf-message.json`, mientras todo lo demás — la propia página compilada — viene de `frontend/dist`; así una sola compilación del frontend llega a todos los receptores. Una copia de `frontend/dist` hecha con enlaces simbólicos no funciona: el servidor rechaza archivos que acaban fuera de su propia carpeta.

4. instance.env. El script de arranque lo lee después de sus propios ajustes:

```
RX_ARGS="-f 145000000 -s 2400000 -g 29.7 -" # -f y -s iguales a config.toml
SPECTRUM_CORES=8-11                       # fuera de los núcleos del receptor principal
```

5. Los datos de su página. Copie `frontend/site_information.json` a `instances/vhf/www/` y edite: `siteReceiver`, `siteAntenna`, `siteSDRBaseFrequency` y `siteSDRBandwidth` (estos deciden qué botones de banda aparecen), y las claves descritas en la [sección 6](#).

6. Arránquelo.

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

7. Anótelos en receivers.toml y reinicie el proxy (`sudo systemctl restart phantomsdr-proxy`). El segundo receptor está ahora en `http://su.host:8899/?rx=vhf`, y ambas páginas muestran el selector.

4. receivers.toml

Copie `receivers.toml.example` a `receivers.toml`. Sin ese archivo el proxy sirve un único receptor, como siempre.

```
# [front]
# port = 8900          # solo disposición B: el proxy escucha también en este puerto público

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

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

Clave	Significado
<code>id</code>	Nombre corto usado en <code>?rx=</code> ; letras, dígitos, <code>-</code> y <code>_</code>
<code>name</code>	El texto del botón del receptor
<code>port</code>	El <code>[server]</code> <code>port</code> de su <code>spectrumserver</code>
<code>host</code>	Opcional; dónde funciona ese <code>spectrumserver</code> (por defecto <code>127.0.0.1</code>)
<code>default</code>	El receptor para las peticiones que no nombran ninguno; como mucho uno
<code>launcher</code>	Su <code>start-*.sh</code> , para <code>start-all.sh</code> ; omítalo en un receptor que este ordenador no arranca
<code>instance</code>	El nombre de su instancia; omítalo en el receptor principal
<code>url</code>	Adónde envía su botón al oyente; sin ella, <code>/?rx=<id></code> en el proxy
<code>hostnames</code>	Nombres DNS opcionales que llevan directamente a este receptor

El proxy sirve también la lista como `/receivers.json` (ids, nombres y enlaces — nunca los puertos internos), que es lo que lee el selector.

5. Arrancar y detener

Orden	Hace
<code>./start-rx888mk2.sh</code>	Arranca o reinicia el receptor principal
<code>INSTANCE=vhf ./start-rtl.sh</code>	Arranca o reinicia el receptor <code>vhf</code>
<code>./start-all.sh</code>	Arranca cada receptor de <code>receivers.toml</code> que tenga <code>launcher</code>
<code>./stop-websdr.sh vhf</code>	Detiene solo <code>vhf</code>
<code>./stop-websdr.sh main</code>	Detiene solo el receptor principal
<code>./stop-websdr.sh</code>	Detiene todos los receptores, como siempre

Mientras un receptor está detenido, su botón desaparece del selector — el proxy solo enumera los receptores que responden, y las páginas abiertas vuelven a leer la lista una vez por minuto — y vuelve cuando el receptor arranca de nuevo; si solo queda un receptor en marcha, la línea *Receivers:* desaparece por completo. Un receptor detenido sí vuelve por sí solo cada vez que se arrancan todos: `./start-all.sh`, el **Restart** del panel de administración y un reinicio de la protección térmica arrancan todo lo que figura en `receivers.toml`. Para dejar uno apagado de forma permanente, quite (o comente) su bloque `[[receiver]]` y reinicie el proxy.

Ponga el **Default start script** del panel de administración en `start-all.sh`. Su Stop ya detiene todos los receptores, y su Restart y la protección térmica usan el script de arranque; con `start-all.sh` devuelven todos los receptores en lugar de solo el principal.

El auxiliar RADE pertenece solo al receptor principal. Una instancia con nombre nunca lo arranca ni lo detiene.

6. Lo que ven los oyentes

El selector de receptor. En una estación con más de un receptor la cabecera de la página recibe una línea propia, *Receivers:*, con un botón por receptor; el actual aparece en amarillo. La página `/mobile` tiene los mismos botones como segunda fila de su barra superior, y abren la página `/mobile` del otro receptor.

Los datos propios de cada receptor. Una página abierta en un segundo receptor carga el `site_information.json` de ese receptor antes de arrancar, de modo que los botones de banda, la frecuencia inicial, la lista de oyentes, los datos de estación y los enlaces de Receiver y Antenna son todos suyos. Una visita normal al receptor principal no hace ninguna petición adicional.

Las claves de `site_information.json` para esto:

Clave	Significado
<code>siteReceiverId</code>	El receptor al que pertenece esta página (su <code>id</code>); omítala en el receptor principal
<code>siteReceiversList</code>	De dónde lee el selector la lista, p. ej. <code>http://su.host:8899/receivers.json</code> ; necesaria en un receptor cuya página no se sirve a través del proxy
<code>siteReceiverURL</code>	Adónde enlaza el nombre de <i>Receiver</i> en <i>Open Additional Info</i>
<code>siteAntennaURL</code>	Adónde enlaza el nombre de <i>Antenna</i> ; "" muestra el nombre sin enlace

Cuándo el selector queda oculto. El selector solo muestra una lista que nombra el host con el que se abrió la página. Abierta por dirección de la red local (`http://192.168.1.10:8900/`) queda oculto; abierta como `http://su.host:8900/` aparece. Es deliberado: una estación que copió el `site_information.json` de otra sin editarlo nunca muestra los receptores de aquella como propios.

7. El S-meter por encima de 30 MHz

La norma de la IARU Región 1 sitúa S9 en -73 dBm por debajo de 30 MHz y en **-93 dBm por encima**, con seis dB por unidad S en ambos casos. Los S-meter (la aguja analógica, la barra digital y la barra de `/mobile`) siguen la frecuencia sintonizada: por encima de 30 MHz indican unidades S de VHF, por debajo como antes. Las cifras en dBm y dBμV no cambian nunca. Si prefiere que los medidores se queden en 0 con el canal vacío, como en un transceptor de VHF/UHF, añada `"siteSMeterGateDb": 6` al `site_information.json` del receptor (sin recompilar): por encima de **60 MHz** la aguja y la barra solo suben entonces cuando una señal supera en esos dB el ruido de su propio entorno (el punto más fuerte

de la banda de paso frente al espectro ± 100 kHz alrededor). Está **desactivado por defecto** — un medidor calibrado que muestra el ruido real de la banda es la lectura más honesta. La página /mobile, que no tiene cascada, estima el ruido a partir del propio nivel recibido.

Calibre los dBm de un receptor de VHF con `analog_smeter_offset` (aguja y cifras) y `smeter_offset` (barra digital) en `[input]` de su `config.toml`, y luego reinicie ese receptor. Fije primero la ganancia: un RTL-SDR informa de los niveles respecto a su propio fondo de escala, así que cada cambio de ganancia mueve las lecturas. Sin generador de señales, una carga de $50\ \Omega$ en lugar de la antena, en USB con filtro de 2,7 kHz, debería marcar unos -136 dBm (el ruido térmico en 2,7 kHz es $-139,7$ dBm, más la figura de ruido del dispositivo).

Para un RTL-SDR, `add-receiver.sh` escribe una calibración de partida: una ganancia fija `-g 29.7` en `instance.env` (la ganancia automática del sintonizador no se puede calibrar) y `analog_smeter_offset=-55`, `smeter_offset=-38`, medidos en un RTL-SDR Blog V4 con esa ganancia y una señal de -71 dBm ($63\ \mu\text{V}$). Otro dispositivo del mismo modelo suele quedar a pocos dB; compruébelo con una señal conocida y recalibre si cambia la ganancia.

8. Seguridad

Poner el proxy delante de los oyentes cerró dos huecos, ambos corregidos en la v5.0.0:

- **La expulsión del sysop a través del proxy.** `spectrumserver` solo permite `/~~kick` desde su propio ordenador, y para él cada petición reenviada por el proxy viene de su propio ordenador. Cualquiera que llegara al puerto del proxy podía, por tanto, desconectar y bloquear a cualquier oyente. Ahora el proxy solo responde a `/~~kick` para clientes del mismo ordenador.
- **Direcciones de cliente falsas.** `spectrumserver` creía la cabecera `X-Forwarded-For` de cualquiera, de modo que un visitante podía decir que era `127.0.0.1` y saltarse los límites por dirección. Ahora la cabecera solo se cree si viene de un proxy local, y el proxy descarta cualquier copia que envíe un cliente antes de añadir la suya.

Ate cada receptor al que se llega a través del proxy a `127.0.0.1` (`[server] host`), para que su puerto no pueda usarse rodeando el proxy. Los límites por dirección de [Límites de conexión](#) se aplican por receptor.

9. Cuántos receptores caben

El software no pone límite; el hardware sí. Los receptores de banda estrecha son baratos: un RTL-SDR a 2,4 Msps usa un 5–10 % de un núcleo, unas decenas de MB de memoria y unos 38 Mbit/s de USB.

- **El USB 2.0 es el techo habitual.** Todos los dispositivos USB 2 comparten un bus de 480 Mbit/s, incluso en un conector azul USB 3. Tres o cuatro RTL-SDR por bus es una cifra segura; añádalos de uno en uno y vigile cada cascada por si aparecen huecos.
- **Alimentación:** con más de dos dispositivos use un hub alimentado.
- **Un segundo receptor de banda ancha** (otro RX-888, o un HackRF a 20 Msps) es otra cosa: compite con el primero por USB 3, CPU y GPU.
- **Los oyentes cuestan más que los receptores.** Cada oyente cuesta CPU en su receptor y ancho de banda de subida, así que cuenta más el número total de oyentes que el número de receptores.

10. Trampas que conviene conocer

- **Dos dispositivos del mismo modelo tienen el mismo número de serie.** Todo RTL-SDR Blog V4 indica 00000001. Dé a cada uno el suyo con `rtl_eeeprom -s <serie>` (con un solo dispositivo conectado cada vez) y nómbrelo en `instance.env` (`RX_ARGS="-d <serie> ..."`), o tras un reinicio los dispositivos pueden intercambiarse los receptores.
- **Los clientes Kiwi no pueden elegir receptor.** Un cliente KiwiSDR marca una dirección y un puerto sin más; a través del proxy siempre acaba en el receptor por defecto. Mantenga `[kiwi_emulation]` desactivado en los demás.
- **Las cookies son por host, no por puerto.** Un visitante que usó el segundo receptor en `:8899` lleva `rx=vhf` también a `:8900`. Por eso una página decide qué receptor es a partir de `siteReceiverId`, nunca de la cookie.
- **Entradas en los directorios.** Dé a cada receptor su propio nombre en `[websdr]`, y en un receptor detrás del proxy ponga `[websdr] public_port` al puerto que usan los oyentes, o la entrada anunciará el interno.
- **Una compilación del frontend muestra brevemente «Not Found».** `frontend/dist` se recompila en su sitio, así que durante unos segundos de la compilación la página falta en todos los receptores.
- **Nada arranca los receptores al encender el ordenador**, salvo que usted lo prepare. Tras un reinicio, `./start-all.sh` los devuelve todos.

CAPÍTULO 7

Servidor de estadísticas

sdr-stats/README.md

Guía de instalación de System Stats

Una guía completa para añadir monitorización de servidor en tiempo real a tu aplicación PhantomSDR mediante el script de instalación automatizado.

Descripción general

Esta función añade un botón **Stats** a tu interfaz de PhantomSDR que muestra información del sistema en tiempo real:

- Uso de CPU, núcleos, frecuencia, temperatura y procesos principales
- Uso de memoria (usada/total/porcentaje)

Las estadísticas se actualizan automáticamente cada 5 segundos mientras la ventana modal está abierta.

No es lo mismo que la página «Gráficos» del panel de administración. Esta ventana es una instantánea en vivo que se refresca mientras está abierta y no guarda histórico. El panel de administración tiene una página **Gráficos** aparte que muestrea la frecuencia de CPU, la carga, la temperatura y los usuarios conectados cada 2 segundos y los representa en los últimos 15 minutos a 24 horas: véase la [guía del panel de administración](#).

Requisitos previos

- **Servidor Linux** (se recomienda Ubuntu/Debian)
- **Conexión a Internet** (para descargar Node.js y los paquetes)
- **Acceso Sudo** (para la configuración del servicio systemd)
- **PhantomSDR-Plus** ya instalado y en ejecución
- **Im-sensors** (recomendado para la temperatura de CPU Intel/AMD, instalado automáticamente por el script)

Métodos de instalación

NOTA

El instalador principal ya lo ofrece. `./install.sh` — y los cuatro instaladores por distribución — ejecutan `install-stats-server.sh` por defecto dentro de la instalación normal de PhantomSDR-Plus, así que en una máquina nueva el servidor de estadísticas suele estar ya en su sitio. Use esta página para instalarlo por separado o para cambiar después su puerto, su dirección o su servicio.

Opción 1: Instalación automatizada (Recomendada)

El script automatizado se encarga de todo por ti.

Paso 1: Ejecutar el script

```
cd ~/PhantomSDR-Plus
chmod +x install-stats-server.sh # only if it is not already executable
./install-stats-server.sh
```

IMPORTANTE: NO lo ejecutes como root ni con sudo. Ejecútalo como tu usuario normal.

Paso 2: Seguir las indicaciones interactivas

El script te guiará a través de:

1. Instalación de Node.js (si es necesario)

```
△ Node.js / npm is not installed
Would you like to install Node.js 22 via nvm now? (y/n): y
```

El script instala Node.js 22 mediante nvm, en cualquier distribución de Linux: sin root y sin fuente apt. Hace la misma pregunta cuando Node.js está presente pero es anterior a la 22.

2. Directorio de instalación

```
Installation directory [/home/user/sdr-stats-server]:
```

Presiona Enter para el valor predeterminado, o especifica una ruta personalizada.

3. Configuración del puerto (¡Mejorada!)

```
Port Configuration:
  ▫ Default port: 3001
  ▫ Common alternatives: 8080, 5000, 8888
Enter port number [3001]:
```

El script valida tu puerto:

- Comprueba que sea un número válido
- Advierte si el puerto < 1024 (requiere privilegios de root)
- Valida el rango (1-65535)
- Comprueba si el puerto ya está en uso
- Muestra qué proceso lo está usando si está ocupado
- Te permite elegir un puerto diferente

Escenarios de ejemplo:

Usando el puerto predeterminado:

```
Enter port number [3001]: ↵
✓ Port 3001 selected
```

Eligiendo un puerto personalizado:

```
Enter port number [3001]: 8080
✓ Port 8080 selected
```

Puerto ya en uso:

```

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

```

Puerto no válido:

```

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. Dirección del servidor

```

What is your server's public address?
Examples: Your_site_IP, 192.168.1.100, localhost
Server address: Your_site_IP

```

Introduce el dominio público o la dirección IP de tu servidor.

5. Confirmación

```

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. Configuración del servicio Systemd (Opcional)

```

Would you like to set up the server as a system service (auto-start on boot)? (y/n): y

```

Recomendado: Elige **y** para que el servidor se inicie automáticamente al arrancar.

Paso 3: ¡Instalación completa! ✓

El script mostrará un resumen:

```

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

```

Configura tu aplicación PhantomSDR

Después de que el script se complete, debes actualizar tu aplicación Svelte.

Paso 1: Actualizar `site_information.json`

Dado que `site_information.json` ya se editó durante la configuración inicial, solo necesitas añadir la nueva línea `siteStats` con tu puerto:

```
{
  "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", ← AÑADE ESTA LÍNEA (usa tu puerto)
  "siteSDRBaseFrequency": 0,
  "siteSDRBandwidth": 30000000,
  "siteRegion": 1,
  "siteChatEnabled": true
}
```

Nota: ¡Usa el puerto que seleccionaste durante la instalación!

Paso 2: Actualizar `App.svelte` (la última versión de GitHub ya incluye el archivo actualizado)

Realiza estos **4 cambios** en tu archivo `App.svelte`:

Cambio 1: Asegurate de que `siteStats` esté declarado (alrededor de la línea 50)

```
import {
  siteSysop,
  siteSysopEmailAddress,
  siteInformation,
  siteGridSquare,
  siteCity,
  siteHardware,
  siteSoftware,
  siteReceiver,
  siteAntenna,
  siteNote,
  siteIP,
  siteStats, // ← AÑADE ESTA LÍNEA
  siteSDRBaseFrequency,
  siteSDRBandwidth,
  siteRegion,
  siteChatEnabled,
} from "../site_information.json";
```

Cambio 2: Actualizar el objeto systemStats (alrededor de la línea 530)

```
let systemStats = {
  cpu: { usage: 0, cores: 0, temperature: null, frequency: null, topProcesses: [] }, // ← AÑADE
  topProcesses: []
  memory: { used: 0, total: 0, percent: 0 }
};
```

Cambio 3: Actualizar la URL de fetch (alrededor de la línea 541)

```
async function fetchSystemStats() {
  try {
    const response = await fetch(`${siteStats}/api/system-stats`); // ← CAMBIA ESTA LÍNEA
    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);
  }
}
```

Cambio 4: Añadir la visualización de procesos principales (en la sección CPU, alrededor de la línea 3838)

Añade este código después de la sección de temperatura:

```
{#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}

<!-- AÑADE TODA ESTA SECCIÓN: -->
{#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}
```

Paso 3: Recompilar tu aplicación

```
cd /path/to/your/phantomsdr-app  
npm ./recompile.sh
```

Probar la instalación

Prueba 1: Verificar el endpoint de la API

```
curl http://localhost:3001/api/system-stats
```

Salida esperada (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  
  }  
}
```

Prueba 2: Verificar el estado del servicio

Si lo instalaste como servicio:

```
sudo systemctl status sdr-stats.service
```

Debería mostrar: Active: active (running)

Prueba 3: Probar en el navegador

1. Abre tu interfaz web de PhantomSDR
2. Desplázate hasta la sección **Additional Info**
3. Haz clic en el botón **Open Additional Info**
4. Busca el botón **Stats** junto a la información del PC
5. Haz clic en él para abrir la ventana modal de estadísticas
6. Verifica que las estadísticas se muestren y se actualicen

Gestión del servicio

Si lo instalaste como servicio systemd, usa estos comandos:

```
# 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
```

Instalación manual (Opción 2)

IMPORTANTE

No necesita esta sección para una instalación normal. `install-stats-server.sh` lo hace todo por usted y el instalador principal lo ejecuta por defecto. Sígala solo para una instalación manual o para reparar una pieza a mano.

⚠ **Advertencia:** La instalación manual puede usar archivos desactualizados. Se **recomienda encarecidamente el script automatizado** ya que:

- Instala la última versión con detección de temperatura mejorada
- Instala y configura lm-sensors automáticamente
- Gestiona todas las dependencias automáticamente

Si aun así prefieres la instalación manual:

1. Instalar 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. Instalar lm-sensors (Requerido para la temperatura Intel/AMD)

```
sudo apt-get update
sudo apt-get install -y lm-sensors
sudo sensors-detect --auto
```

3. Crear el directorio

```
mkdir ~/sdr-stats-server
cd ~/sdr-stats-server
```

4. Descargar los archivos más recientes

Importante: los archivos del servidor ya no se guardan como copias en el repositorio: los genera el script de instalación.

En su lugar, ejecuta el script automatizado una vez para generar los archivos más recientes y luego cópialos:

```
# 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/
```

O crea manualmente los archivos usando el código más reciente del script de instalación.

5. Editar el puerto (si es necesario)

```
nano ~/sdr-stats-server/system-stats-server.js
```

Cambia la línea 6:

```
const PORT = process.env.PORT || 3001; // Change this number
```

6. Instalar dependencias

```
npm install
```

7. Ejecución de prueba

```
npm start
```

8. Configurar el servicio (Opcional)

Sigue la configuración del servicio systemd de la sección de instalación automatizada.

Cambiar el puerto después de la instalación

Si necesitas cambiar el puerto después de la instalación:

Paso 1: Editar el archivo del servidor

```
nano ~/sdr-stats-server/system-stats-server.js
```

Cambia la línea 6:

```
const PORT = process.env.PORT || 3001; // Change this number
```

Paso 2: Actualizar site_information.json

```
"siteStats": "http://Your_site_IP:NEW_PORT"
```

Paso 3: Reiniciar el servicio

```
sudo systemctl restart sdr-stats.service
```

Paso 4: Reconstruir tu aplicación Svelte

```
cd /path/to/phantomsdr-app
npm run build
```

Solución de problemas

Problema: Puerto ya en uso

Error:

```
Error: listen EADDRINUSE: address already in use :::3001
```

Solución:

1. Encuentra qué está usando el puerto:

```
sudo lsof -i :3001
```

1. Detén ese proceso:

```
kill <PID>
# or
sudo systemctl stop sdr-stats.service
```

1. Inicia el servicio de nuevo:

```
sudo systemctl start sdr-stats.service
```

Alternativa: Ejecuta el script de instalación de nuevo y elige un puerto diferente cuando se te solicite.

Problema: El servicio no arranca

Error:

```
status=217/USER
```

Solución:

1. Comprueba el archivo del servicio:

```
cat /etc/systemd/system/sdr-stats.service
```

1. Verifica que la línea `User=` coincida con tu nombre de usuario:

```
whoami
```

1. Edítalo si es necesario:

```
sudo nano /etc/systemd/system/sdr-stats.service
```

1. Recarga y reinicia:

```
sudo systemctl daemon-reload
sudo systemctl restart sdr-stats.service
```

Problema: Las estadísticas no se muestran en el navegador

Síntomas: La ventana modal se abre pero no hay datos, o muestra valores en 0

Soluciones:

1. Comprueba la consola del navegador (F12) en busca de errores
2. Verifica que la API sea accesible:

```
curl http://localhost:3001/api/system-stats
```

1. Comprueba que el servicio esté en ejecución:

```
sudo systemctl status sdr-stats.service
```

1. Verifica que `siteStats` en `site_information.json` coincida con la dirección y el puerto de tu servidor
2. Comprueba los errores de CORS: asegúrate de que el servidor permita tu dominio

Problema: La temperatura muestra null

Causa: La lectura de temperatura no está disponible en tu sistema

Soluciones:

1. **Para la mayoría de los sistemas Linux:** Comprueba si existe la zona térmica:

```
cat /sys/class/thermal/thermal_zone0/temp
```

1. Instala lm-sensors:

```
sudo apt-get install lm-sensors
sudo sensors-detect
sensors
```

1. **Nota:** Es posible que la temperatura no esté disponible en:
2. Máquinas virtuales
3. Algunos proveedores de VPS
4. Windows Subsystem for Linux (WSL)
5. Sistemas no Linux

¡Esto es normal! El resto de las estadísticas seguirá funcionando.

Problema: Los valores de las estadísticas no coinciden con `top`

Causa: El período de muestreo de la CPU era demasiado corto en versiones anteriores

Solución: La versión actual usa un muestreo de 1 segundo para lecturas precisas. Si tienes una instalación antigua:

1. Actualiza el archivo del servidor:

```
nano ~/sdr-stats-server/system-stats-server.js
```

1. Encuentra la línea 18 y cambia:

```
}, 100); // Old value
```

a:

```
}, 1000); // New value (1 second)
```

1. Reinicia:

```
sudo systemctl restart sdr-stats.service
```

O simplemente vuelve a ejecutar el script de instalación para obtener la última versión.

Problema: Node.js no encontrado

Error:

```
node: command not found
```

Solución:

1. Instala 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. Verifica:

```
node --version
npm --version
```

Lo que verás

Una vez instalado y configurado, la ventana modal de estadísticas muestra:

Sección CPU

- **Uso:** Porcentaje de utilización general de la CPU
- **Núcleos:** Cuánta CPU se usa realmente, expresada en núcleos completos —por ejemplo 1.3 / 12 in use significa que la máquina hace el trabajo de 1,3 núcleos—. Es exacto y no necesita ningún umbral. Deliberadamente no es un recuento de núcleos ocupados: el planificador reparte el trabajo de un núcleo entre muchos, así que ese recuento solo podría aproximar esta cifra.
- **Frecuencia:** Frecuencia actual media de la CPU en todos los núcleos en GHz, con el núcleo más rápido entre paréntesis (si está disponible)
- **Temperatura:** Temperatura de la CPU en grados Celsius (si está disponible), con código de color: verde hasta 69 °C, naranja 70-79 °C, rojo a partir de 80 °C
- **Procesos principales:** Los 5 procesos que más consumen CPU con porcentajes codificados por colores:
 - Verde: hasta 50 % de CPU
 - Naranja: 51-80 % de CPU
 - Rojo: más del 80 % de CPU

Sección de memoria

- **Usada:** Memoria en uso (GB)
- **Total:** Memoria total del sistema (GB)
- **Uso:** Porcentaje de uso de memoria (codificado por colores)

🔄Actualización automática

- Las estadísticas se actualizan cada **5 segundos** mientras la ventana modal está abierta
- Deja de actualizarse cuando cierras la ventana modal (ahorra recursos)

Detección de temperatura

El servidor de estadísticas utiliza una detección de temperatura avanzada con múltiples métodos:

Sistemas compatibles:

- **Procesadores Intel:** Lee de los sensores coretemp mediante lm-sensors
- **Procesadores AMD:** Lee de los sensores k10temp (Tdie/Tctl)

- **Procesadores ARM:** Lee de las zonas térmicas (Raspberry Pi, etc.)

Requisitos:

- **Intel/AMD:** Requiere el paquete `lm-sensors` (instalado automáticamente por el script)
- **ARM/Raspberry Pi:** Funciona de inmediato, no se necesitan paquetes adicionales

Prioridad de instalación:

1. Intenta el comando `sensors` para la temperatura Package/Core (Intel/AMD)
2. Recurre a la lectura directa de `coretemp` (Intel)
3. Recurre a `thermal_zone0` (ARM/Raspberry Pi)

Si la temperatura muestra `null`, instala `lm-sensors`:

```
sudo apt-get install lm-sensors
sudo sensors-detect --auto
sudo systemctl restart sdr-stats.service
```

Resumen de funciones

Monitorización en tiempo real - Estadísticas del sistema en vivo desde tu servidor **Métricas de CPU** - Uso, núcleos, temperatura, procesos principales **Seguimiento de memoria** - Usada, total y porcentaje **Actualizaciones automáticas** - Se actualiza cada 5 segundos **Codificado por colores** - Indicadores visuales para las advertencias **Ligero** - Uso mínimo de recursos **Instalación fácil** - El script automatizado se encarga de todo **Validación de puerto** - Selección inteligente de puerto con detección de conflictos **Gestión del servicio** - Inicio automático al arrancar

Notas de seguridad

- El servidor de estadísticas acepta conexiones de cualquier origen (CORS: *)
- Para producción, considera restringir CORS solo a tu dominio
- La selección de puerto valida la entrada y comprueba conflictos
- La lectura de la temperatura de la CPU es de solo lectura y segura
- No se expone información sensible del sistema

Soporte y recursos

Archivos incluidos:

En esta carpeta (el script de instalación está en la raíz, como `PhantomSDR-Plus/install-stats-server.sh`: solo existe una copia):

- `package.json` - Dependencias, idénticas a las que escribe el script
- Este documento, disponible también en de, el, es, fr, hr y ru

Creados por el script en `~/sdr-stats-server/`:

- `system-stats-server.js` - Servidor backend, generado con el puerto elegido
- `package.json` - Dependencias

- `node_modules/` - instalado por `npm install`

Obtener ayuda:

- Consulta la sección de solución de problemas anterior
- Revisa los registros del servicio: `sudo journalctl -u sdr-stats.service -n 50`
- Prueba la API manualmente: `curl http://localhost:3001/api/system-stats`
- Verifica los archivos de configuración

Desinstalar el servidor de estadísticas:

- Elimina la carpeta `/sdr-stats-server`
- Si has creado un servicio, simplemente elimínalo usando:

```
sudo rm /etc/systemd/system/sdr-stats.service`
```

- Modifica `App.svelte`: Encuentra la parte del código:

```

        <!-- 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>
    ...
y reemplázala con:
```

```

        <!-- 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>
        -->
```

...

- Recompila usando el comando:

```

    cd PhantomSDR-Plus
    ./recompile.sh
    ```

- Reinicia el PC e inicia el servidor

Autor: Creado para PhantomSDR-Plus
Versión: 1.1 (Detección mejorada de la temperatura de la CPU para Intel/AMD) **Actualizado:**
Enero de 2026
Licencia: MIT

Referencia rápida de comandos

```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

```

¡Feliz monitorización!

CAPÍTULO 8

Decodificadores

DECODERS.md

PhantomSDR-Plus — Manual del usuario de los decodificadores

Esta guía cubre todos los decodificadores integrados disponibles en PhantomSDR-Plus. Todos los decodificadores comparten el mismo procedimiento de activación, descrito a continuación, seguido de las instrucciones de configuración de cada uno.

Índice

1. [Cómo iniciar un decodificador](#)
2. [FT8](#)
3. [FT4-FT2](#)
4. [JS8](#)
5. [CW — código Morse](#)
6. [QRSS Grabber](#)
7. [WSPR](#)
8. [FAX de HF / WEFAX](#)
9. [NAVTEX](#)
10. [FSK / RTTY — incluidos PSK31 y Olivia](#)
11. [SSTV](#)
12. [Consejos generales](#)

1. Cómo iniciar un decodificador

Se accede a todos los decodificadores desde la sección **Decoder Options**, situada bajo los controles del espectrograma de audio en el panel principal.

Pasos:

1. Haga clic en el botón **Decoder: OFF** para cambiarlo a **Decoder: ON** (se vuelve azul cuando está activo).
2. Abra el menú desplegable que aparece a la derecha del botón y seleccione el decodificador que desee.
3. El panel del decodificador elegido aparecerá bajo los controles; siga las instrucciones específicas de ese decodificador en la sección correspondiente de esta guía.
4. Para detener la decodificación, seleccione — **Select decoder** — en el desplegable, o haga clic en el botón **Decoder: ON** para volver a ponerlo en OFF.

Botones de decodificador de una pulsación

El desplegable no es la única vía. El panel principal incluye una fila de botones **Decoders** — justo debajo de **Wheel Tuning Steps** — con un botón por decodificador:

Botón	Decodificador	Botón	Decodificador
FT8	FT8	SSTV	SSTV
FT4	FT4	NAVTEX	NAVTEX
FT2	FT2	RTTY	FSK / RTTY
CW	CW		
WSPR	WSPR	FAX	HF FAX / WEFAX

Pulsar un botón hace toda la secuencia de una vez: selecciona el decodificador, pone el decodificador en ON y desplaza su ventana hasta hacerla visible. El botón se pone azul mientras su decodificador funciona. **Pulse el mismo botón otra vez para detener el decodificador:** su ventana se cierra con él.

RADEL y **RADEU** no están en esta fila deliberadamente. Son modos de voz digital y no decodificadores de texto, por lo que tienen su propio par de botones junto al encabezado **Modes selector** y dentro de las ventanas emergentes **Modes** y **Bands**. Se comportan exactamente igual: pulsar para iniciar, volver a pulsar para detener. Consulte el [manual de RADE](#).

Los botones, el desplegable y el botón ON/OFF gobiernan el mismo estado, así que use el que use, los demás lo siguen.

Solo puede haber un decodificador activo a la vez. Cambiar a otro decodificador detiene automáticamente el anterior.

El **grabber QRSS** no forma parte de este menú desplegable: tiene su propia sección **QRSS** y puede funcionar al mismo tiempo que un decodificador.

La decodificación se ejecuta en segundo plano. SSTV, FAX de HF, NAVTEX, FSK/RTTY y CW se ejecutan cada uno en un hilo Web Worker independiente, de modo que el trabajo de decodificación nunca compite con la reproducción de audio ni con la cascada. Iniciar o detener un decodificador no interrumpe el audio, y la interfaz sigue respondiendo mientras se recibe una imagen o una página. Todos los decodificadores reciben audio en bruto tomado *antes* del AGC, la reducción de ruido y el silenciado, así que silenciar el receptor o ajustar esos parámetros a su gusto no afecta a la decodificación.

Un decodificador en marcha manda sobre el modo y el paso de banda. Normalmente el modo sigue el plan de bandas de `bands-config.js`: lleve el dial a un segmento marcado LSB o AM y el receptor cambia a él. Mientras un decodificador está en marcha eso ya no ocurre. El decodificador mantiene el modo que necesita (USB en casi todos, el suyo propio en RADE) y el paso de banda que necesita — PSK31 unos ± 100 Hz, Olivia todo su ancho, RTTY su shift— y ambos sobreviven a la resintonía, incluso a un salto a otra banda. Sin esto, sintonizar FT8 en 40 m pasaría el receptor a LSB en cuanto se moviera el dial, y los pasos de banda estrechos de los decodificadores se abrirían de nuevo al filtro SSB completo.

El modo propio de la banda vuelve en cuanto apaga el decodificador. Siempre puede imponer el modo a mano: los botones de modo son una elección deliberada y siempre ganan. El decodificador de CW es la excepción a todo esto: decodifica en el modo en que usted esté escuchando y nunca toma el control del receptor.

Decoder ID — reconocer el modo automáticamente

Sobre los botones de decodificador se encuentra el marco **Decoder ID**. Responde a la pregunta «¿qué estoy escuchando?» cuando encuentre una señal digital que no sepa identificar, y a continuación le ofrece el decodificador adecuado con un solo clic.

Está **desactivado de forma predeterminada** y no consume nada mientras está apagado. Pulse **On** y deje la señal sintonizada. Durante aproximadamente medio minuto muestra *Listening...* con una barra

de progreso — las medidas de temporización necesitan ver dos ciclos completos de FT8 antes de tener sentido — y después nombra el modo:

Decoder ID [0n] NAVTEX / SITOR-B 99% 170 Hz shift · 100.00 Bd · 43 dB S/N [Use NAVTEX]

- El **modo** y una cifra de **confianza**. Verde por encima del 75 %, ámbar por encima del 50 %, naranja por debajo — trate una respuesta naranja como una pista, no como un veredicto.
- **Lo que realmente se midió**: ancho de banda ocupado, separación de tonos, velocidad de símbolo, duración de la ráfaga, la rejilla UTC a la que se ajusta, relación señal/ruido. Ésta es la evidencia detrás del veredicto, para que pueda juzgar usted mismo.
- **Use** arranca el decodificador correcto, exactamente como si hubiera pulsado su botón — y prepara el receptor para trabajar ese modo. Para los modos que viven en el panel FSK además selecciona la variante correcta — RTTY, Weather RTTY, PSK31 u Olivia.
- Los candidatos siguientes aparecen tras **or**. Pulse uno para probarlo, de modo que una primera respuesta equivocada no le cueste nada.
- ******** significa que la frecuencia en la que está sintonizado es una frecuencia de llamada conocida para ese modo, y que eso contó en la clasificación.

Nunca cambia su decodificador por sí solo. Únicamente sugiere. Y en cuanto arranca un decodificador — desde la fila de botones, desde la lista desplegable o desde el propio botón **Use** de Decoder ID — Decoder ID se apaga solo: su tarea era elegir uno, y ahora el decodificador quiere ese mismo audio.

Prepara el receptor para el modo. Elegir un modo es decidir trabajarlo, así que el receptor le sigue: la frecuencia sintonizada pasa al centro de la cascada y la vista queda en unos 100 kHz a su alrededor, la banda lateral pasa a ser aquella en la que se trabaja ese modo y el paso de banda se estrecha hasta la ventana en la que vive ese modo — la subbanda completa de 3 kHz para FT8, FT4, FT2 y JS8, 1350–1650 Hz para WSPR, 250–750 Hz para NAVTEX, 800–2700 Hz para HF FAX, 900–2600 Hz para SSTV y ± 250 Hz a cada lado del dial para CW, que además conmuta el receptor a CW. Un paso de banda estrecho mantiene las señales vecinas fuera del decodificador, que es lo que habría hecho usted a mano. Dos excepciones: SSTV conserva la banda lateral que eligió por sí mismo — LSB en 80 y 40 m por convención, donde forzar USB invertiría su correspondencia de tonos — y los modos FSK conservan la ventana que su propio panel deduce del shift o del ancho de banda de la variante, más estrecha que cualquier cifra fija. Pulsar **Use** en el decodificador que ya está funcionando sigue apagándolo, y entonces el receptor se queda exactamente como estaba.

Los modos lentos tardan más. WSPR va en una rejilla de dos minutos y las imágenes SSTV duran uno o dos minutos, así que esos dos se nombran más tarde que el resto — déles tres o cuatro minutos en la frecuencia. HF FAX y SSTV solo se reconocen, además, si están sintonizados de la forma habitual — con la banda de imagen entre unos 1500 y 2300 Hz de audio, que es donde también la esperan sus decodificadores. Cambiar de frecuencia reinicia la medición desde cero, porque todo lo reunido hasta entonces describe la frecuencia que acaba de abandonar.

Qué reconoce: FT8, FT4, JS8, WSPR, CW, NAVTEX/SITOR-B, RTTY a 45,45 Bd, Weather RTTY (DWD), PSK31, Olivia, HF FAX, SSTV y DSC. En esta versión no hay decodificador de DSC, así que DSC se nombra y se marca como *no decoder*.

Recentre & retry. El lugar que ocupa la señal dentro del paso de banda de audio no importa entre unos 400 Hz y 2700 Hz. Fuera de ese margen la medida se degrada y aparece un botón ámbar **Recentre & retry**. Al pulsarlo, el dial se desplaza para que la señal quede en el centro del paso de banda y comienza una medida nueva. Es un único movimiento calculado, no una búsqueda.

Tres límites honestos:

- **FT8, JS8 y FT2 son la misma señal.** La misma modulación 8-FSK, la misma separación de tonos, y JS8 a velocidad Normal usa la cadencia de 15 segundos de FT8. Nada en la señal misma los separa.

La frecuencia sí: en una frecuencia de llamada de FT8 informa FT8, en una de JS8 informa JS8, y en cualquier otro sitio informa honestamente la familia — *FT8 / JS8 / FT2*. Arranque cualquiera de los dos decodificadores y vea cuál produce texto.

- **Calla en lugar de adivinar.** Por debajo de unos 10 dB de relación señal/ruido informa *Nothing above the noise* o *Not recognised* en vez de nombrar un modo que no puede ver realmente. Es deliberado: una respuesta equivocada y segura de sí misma es peor que ninguna respuesta.
- **Una señal con desvanecimiento se lee de otra manera.** WSPR transmite 110 segundos seguidos, más de lo que dura un desvanecimiento QSB corriente, de modo que un desvanecimiento profundo parte la transmisión en trozos y su ritmo de dos minutos ya no puede leerse en ellos. Entonces se recupera por otra vía: se compara todo el historial de escucha con la rejilla de dos minutos en lugar de con transmisiones sueltas, y la línea de medidas indica *120 s grid (through fading)*. Es una prueba más débil que una transmisión oída entera, así que la confianza mostrada es deliberadamente menor. Un desvanecimiento lo bastante profundo como para hundir la señal en el ruido no se recupera en absoluto. Olivia es más estricta que las demás: no se nombra hasta haber medido su velocidad de símbolo, por lo que enmudece antes.

2. FT8

Qué es: FT8 es un modo digital de señal débil muy popular entre los radioaficionados de todo el mundo. Las transmisiones duran exactamente 15 segundos y el protocolo puede copiar señales hasta 20-25 dB por debajo del nivel de ruido. Es el modo más utilizado para contactos a larga distancia (DX).

Frecuencias recomendadas (USB)

Banda	Frecuencia
160 m	1.840 MHz
80 m	3.573 MHz
40 m	7.074 MHz
30 m	10.136 MHz
20 m	14.074 MHz
17 m	18.100 MHz
15 m	21.074 MHz
12 m	24.915 MHz
10 m	28.074 MHz

Configuración

1. Sintonice una de las frecuencias de FT8 indicadas arriba y ponga el modo en **USB**.
2. Active el decodificador y seleccione **FT8** en el desplegable. O basta con pulsar el botón **FT8**.
3. El panel **FT8 Messages** aparece automáticamente debajo.

Lectura de la salida

La lista de mensajes muestra las transmisiones decodificadas a medida que llegan. Cada ciclo de 15 segundos produce un nuevo lote de mensajes. El campo **Farthest** (arriba a la derecha del panel) indica la mayor distancia decodificada en la sesión actual, en kilómetros.

Formato de mensaje típico: CQ DX AA1BB FN31 — una llamada CQ del indicativo AA1BB situado en el cuadrante FN31.

La **SNR** que se muestra con cada decodificación es una relación señal/ruido real, medida sobre la señal decodificada frente al ruido que la rodea y referida a un ancho de banda de 2500 Hz, como la da WSJT-X. Las estaciones débiles marcan valores negativos, hasta unos -20 dB, y las fuertes positivos; con las mismas grabaciones ambos programas coinciden en promedio, normalmente con unos 3 dB de diferencia en una decodificación aislada. FT4 y FT2 usan el mismo método.

Tras una actualización: el decodificador FT8/FT4/FT2 es un archivo aparte que los navegadores conservan mucho tiempo. Si todas las SNR siguen entre unos 0 y +5 dB, recargue la página a fondo (**Ctrl+Mayús+R**, o **Cmd+Mayús+R** en un Mac).

Consejo: FT8 está estrechamente sincronizado en el tiempo. Su navegador usa el reloj del ordenador; si el reloj del sistema se desvía más de un par de segundos, la decodificación fallará. Mantenga la hora del sistema sincronizada por NTP.

3. FT4-FT2

Qué es: FT4 es una variante más rápida de FT8 diseñada para operar en concursos. Cada ciclo de transmisión dura 7,5 segundos (la mitad que FT8), lo que lo hace el doble de rápido pero exige una señal algo más fuerte. FT2 es una variante aún más rápida de FT8. Es un modo ultrarrápido de 77 bits con periodos TR de 3,75 segundos (= la mitad que FT4); todavía experimental.

Frecuencias recomendadas (USB)

Banda	Frecuencia FT4
80 m	3.575 MHz
40 m	7.047 MHz
30 m	10.140 MHz
20 m	14.080 MHz
15 m	21.140 MHz
10 m	28.180 MHz

Banda	Frecuencia FT2
160 m	1.843 a 1.846
80 m	3.578 a 3.581
60 m	5.360 (consulte la normativa regional local)
40 m	7.052 a 7.062
30 m	10.144
20 m	14.084
17 m	18.108
15 m	21.144
12 m	24.923
10 m	28.184

Configuración

1. Sintonice una frecuencia de FT4 y ponga el modo en **USB**.
2. Active el decodificador y seleccione **FT4** o **FT2** en el desplegable. O basta con pulsar el botón **FT4** / **FT2**.
3. El panel **FT4 Messages** o **FT2 Messages** aparece debajo, con la misma disposición que el panel de FT8.

Nota: FT4 y FT8 usan formatos espectrales distintos y no son intercambiables. Asegúrese de estar en una frecuencia de FT4 al usar este decodificador.

4. JS8

Qué es: JS8 (el modo que usa JS8Call) toma el motor de señal débil de FT8 y lo convierte en un modo de conversación de teclado a teclado. Mientras FT8 envía intercambios fijos de 13 caracteres, JS8 envía texto libre en pequeños fragmentos y vuelve a unir las transmisiones consecutivas en frases completas, de modo que todo lo que supere unos doce caracteres llega a lo largo de varios ciclos de 15 segundos. Decodifica aproximadamente tan profundo como FT8, por lo que sirve cuando la voz y la CW ya no pasan.

Frecuencias recomendadas (USB)

Banda	Frecuencia
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

Velocidades

JS8 tiene cinco velocidades. Todas las estaciones de una conversación deben usar la misma. **Normal** es la velocidad de llamada y concentra casi todo el tráfico: empiece ahí.

Velocidad	Ciclo	Ancho de banda	Cuándo se usa
Slow	30 s	25 Hz	Trayectos muy débiles; la más sensible
Normal	15 s	50 Hz	La velocidad de llamada habitual
Fast	10 s	80 Hz	Intercambios más rápidos, exige más señal
Turbo	6 s	160 Hz	Señales locales fuertes
Ultra	4 s	250 Hz	Experimental, rara vez se ve

Configuración

1. Sintonice una frecuencia de JS8 y ponga el modo en **USB**.
2. Active el decodificador y seleccione **JS8** en la lista, o pulse el botón **JS8**.
3. Aparece el panel **JS8 Decoder**. Deje **Speed** en *Normal* salvo que sepa que la estación que busca usa otra.

Cómo leer la salida

El panel tiene dos listas.

Mensajes que aún están llegando aparecen arriba en verde, con un cursor parpadeante. Un mensaje JS8 puede tardar cuatro ciclos —un minuto entero en Normal—, así que aquí se ve la frase construirse. No pasa nada si permanece así un rato.

Mensajes completos llenan la lista principal, una fila cada uno:

Columna	Significado
Mode	Siempre JS8
Hz	Frecuencia de audio de la señal dentro del paso de banda
dB	Relación señal/ruido en un ancho de referencia de 2500 Hz
Message	El texto decodificado; los indicativos se muestran en verde

Mensajes típicos:

- SV1BTL KM17: HB — un heartbeat: la estación anuncia que está en el aire, con su localizador.
- KN4CRD: K00G SNR -05 — un mensaje dirigido: KN4CRD indica a K00G que lo recibe a -5 dB.
- MP 100W 8M/BALUN JN58KH AUGSBURG, MARTIN — texto libre, aquí una descripción de estación llegada en cuatro ciclos.

Una fila **atenuada y en cursiva** expiró antes de que llegara su última trama. El texto es real, pero puede estar cortado.

Sync offset

El deslizador **Sync offset** fija cuánto después del límite del ciclo UTC empieza la captura, compensando el retardo de la cadena de audio. Deje **Auto** marcado: el decodificador mide la temporización de las señales que oye y se ajusta solo.

Marcas de espectro

La pequeña tira de espectro del panel dibuja una línea vertical amarilla en la frecuencia de cada señal decodificada en el ciclo anterior, de modo que la columna Hz se lee directamente en la pantalla.

Ver sus propios spots

Si el receptor sube spots JS8 a PSK Reporter (véase la pestaña Spot Reporting del panel de administración), el botón **JS8 map** del recuadro **Note:** abre el mapa de PSK Reporter filtrado por el indicativo de este receptor y el modo JS8 — lo mismo que hacen **FT8 map** y **FT4 map** para esos modos.

Consejo: JS8 es mucho más tranquilo que FT8. Una tarde puede ver una transmisión cada pocos minutos, y los silencios largos son normales. Déle cinco o diez minutos antes de pensar que algo falla. 20m (14.078) y 40m (7.078) son los sitios habituales.

Consejo: Como FT8, JS8 depende de una sincronización estricta y usa el reloj de su ordenador. Si la hora del sistema se desvía más de uno o dos segundos, no decodificará nada. Manténgala sincronizada por NTP.

5. CW — código Morse

Qué es: el decodificador de CW escucha señales en código Morse (onda continua) y las convierte en texto en tiempo real. Sigue automáticamente la frecuencia de la señal y se adapta a la velocidad de manipulación del operador.

Frecuencias recomendadas

La CW está activa en todas las bandas de aficionado, normalmente en la parte baja de cada banda. Puntos habituales:

Banda	Segmento
40 m	7.000–7.040 MHz
20 m	14.000–14.070 MHz
15 m	21.000–21.080 MHz

Configuración

1. Sintonice una señal de CW usando el modo de demodulación **CW** o **CW-L** según proceda.

2. Active el decodificador y seleccione **CW** en el desplegable. O basta con pulsar el botón **CW**.
3. El panel **CW Decoder** aparece debajo.

Lectura de la salida

- La cabecera del panel muestra la frecuencia detectada de la señal en Hz (p. ej., ≈ 700 Hz) y la velocidad de manipulación estimada en palabras por minuto (p. ej., $\cdot 22$ WPM).
- Si no se detecta señal, la cabecera muestra **scanning...**
- En un canal vacío no se imprime nada, y no se muestran ni la velocidad ni el tono. El decodificador solo empieza a escribir cuando hay una diferencia clara entre manipulador pulsado y suelto, algo que el ruido solo no produce, mientras que una señal aún legible a unos -10 dB la tiene de sobra.
- El texto decodificado se desplaza en caracteres ámbar de paso fijo. El cursor parpadeante (█) marca dónde se está escribiendo.
- Haga clic en **Clear** para borrar el búfer de salida.

Consejos: - Centre el paso de banda en el tono de CW. El decodificador funciona mejor cuando la señal de CW se sitúa aproximadamente entre 400 y 900 Hz en el espectro de audio. - Se copian velocidades de unas 8 a 40 PPM, también con manipulación Farnsworth (caracteres enviados rápido con pausas más largas entre ellos): el enganche de velocidad se mantiene y las letras no se parten en palabras sueltas. El Morse muy manual (irregular) todavía puede reducir la precisión. - El decodificador rinde mejor con una única señal limpia. Un QRM fuerte de señales cercanas en la misma banda puede confundirlo.

6. QRSS Grabber

Qué es: QRSS es CW enviado tan despacio que un solo punto dura segundos en lugar de milisegundos. A esa velocidad la señal ocupa solo una fracción de hercio, de modo que un análisis lo bastante estrecho puede sacar la traza desde 20–30 dB *por debajo* del nivel de ruido. No hay nada que leer de oído: el QRSS **se mira**, no se escucha. El grabber calcula su propia FFT muy larga sobre el audio del receptor y dibuja la presentación clásica de grabber: frecuencia en el eje vertical, tiempo desplazándose de izquierda a derecha, con la columna más reciente siempre en el borde derecho.

Casi todo lo que verá procede de balizas MEPT (Manned Experimental Propagation Transmitter): transmisores de muy baja potencia, a menudo unos cientos de milivatios sobre un hilo, que se identifican continuamente en Morse muy lento o con patrones FSK.

Dónde buscar

Las balizas QRSS viven en ventanas estrechas de 100–200 Hz cerca del extremo inferior de cada banda. No hace falta recordarlas: el grabber tiene una lista **Band** con las ventanas de abajo, y elegir una sintoniza allí y prepara el receptor para ello (véase *Abrir el grabber*).

Banda	Centro de la ventana	Nota
30m	10.140,00 kHz	La ventana QRSS principal: con diferencia la más concurrida, de día y de noche
40m	7.039,90 kHz	
40m	7.000,85 kHz	Ventana Knights, 7.000,8–7.000,9 kHz
20m	14.096,90 kHz	
80m	3.569,90 kHz	
80m	3.568,60 kHz	Ventana antigua
80m	3.500,85 kHz	Ventana Knights, 3.500,8–3.500,9 kHz
160m	1.837,90 kHz	
160m	1.843,30 kHz	Ventana antigua
630m	476,10 kHz	
2200m	137,70 kHz	QRSS / DFCW en LF
60m	5.288,55 kHz	
17m	18.105,90 kHz	
15m	21.095,90 kHz	
12m	24.925,90 kHz	
10m	28.125,70 kHz	
10m	28.000,85 kHz	Ventana antigua
10m	28.322,00 kHz	Alternativa
6m	50.294,30 kHz	

Varias bandas aparecen dos veces porque hay realmente dos convenciones en uso. Las ventanas modernas se sitúan 200 Hz por debajo de la frecuencia WSPR de esa banda; las antiguas ventanas *Knights* están en otro sitio por completo, y en 40 y 80 m mucho más abajo. Si una banda está tranquila en una, pruebe la otra.

Son convenciones, no reglamentos, y algunas son regionales: las cifras de 10 m en particular varían. Tome la lista como punto de partida y siga la práctica local.

Abrir el grabber

El grabber QRSS **no** está en el desplegable de decodificadores. Tiene su propia sección **QRSS**, junto a los controles del espectrograma y de los decodificadores.

1. Pulse **Show**. Aparece el lienzo del grabber y sus controles se despliegan junto al botón.
2. Elija una ventana en la lista **Band** y pulse **Tune**. Eso hace de una vez todo lo que el modo necesita: sintoniza la ventana, pasa el receptor a **CW** y fija un paso de banda justo lo bastante ancho para la porción que está mirando. El dial marca entonces la frecuencia QRSS real, y las trazas caen sobre la línea central de la presentación.
3. Ajuste **Speed** a la duración del punto de la baliza.
4. **Centre** y **Span** funcionan como antes; cambiar cualquiera de los dos reajusta el paso de banda, de modo que la presentación y el receptor siguen acompasados.
5. Pulse **Hide** para detenerlo. Vuelve el modo normal de la banda.

También puede seguir haciéndolo a mano: sintonice en **USB** alrededor de 1 kHz por debajo de la ventana para que las trazas caigan cerca de 800 Hz de audio. La lista **Band** solo le ahorra la aritmética.

Mientras hay sintonizada una ventana de la lista, el grabber retiene el receptor igual que un decodificador: CW y su paso de banda sobreviven a la resintonía, y la ventana sigue al dial, así que la traza se mantiene en la línea central mientras recorre la banda. Devuelve el receptor cuando oculta el grabber, pulsa un botón de modo o arranca un decodificador.

Velocidades

Ajuste	Duración del punto	Resolución	Ventana de análisis	Nueva columna cada
QRSS 3	3 s	0,73 Hz	1,4 s	0,7 s
QRSS 6	6 s	0,37 Hz	2,7 s	1,4 s
QRSS 10	10 s	0,18 Hz	5,5 s	2,7 s
QRSS 30	30 s	0,09 Hz	10,9 s	5,5 s
QRSS 60	60 s	0,05 Hz	21,8 s	10,9 s

Un ajuste más rápido que la baliza desperdicia sensibilidad: la traza sale fina y ruidosa. Un ajuste más lento emborrona puntos y rayas consecutivos en una sola barra. El panel arranca en **QRSS 6**, un punto de partida razonable ante una señal desconocida: más sensible que QRSS 3, y más rápido de actualizar y más tolerante a la deriva que QRSS 10. En cuanto distinga la forma del manipulado, pase a la velocidad que la baliza envía realmente: cada paso hacia un ajuste más rápido cuesta 3 dB de sensibilidad.

Controles

Control	Qué hace
Band	Las ventanas QRSS de la tabla anterior. Tune va a la seleccionada, en CW y con un paso de banda a medida.
Speed	Fija la longitud de la transformada, es decir el compromiso entre resolución en frecuencia y en tiempo (tabla anterior).
Centre	Frecuencia de audio en el centro de la presentación, 100–3000 Hz.
Span	Altura de la franja mostrada: 20, 50, 100 o 200 Hz, con 100 Hz al arrancar: el ancho de una subbanda QRSS. Una franja más estrecha reparte cada traza en más píxeles. En las franjas anchas caben en el panel más bins que filas de píxeles tiene; cada fila muestra entonces el bin más fuerte que abarca, de modo que nada puede esconderse entre filas.
Gain	De –10 a +40 dB. Desplaza el mapa de color respecto al nivel de ruido medido; súbalo para aclarar trazas débiles.
Color	Rainbow, Green o Grayscale.
Clear	Borra el lienzo y reinicia la referencia de ruido.

Cómo leer la presentación

Las marcas de frecuencia recorren el borde izquierdo, con la frecuencia más alta arriba. Bajo el lienzo, una línea de estado muestra la velocidad seleccionada, la resolución realmente en uso (p. ej. $0.183 \text{ Hz/bin} \cdot 5.5 \text{ s window}$) y el ritmo de columnas (p. ej. 2.7 s/column).

El mapa de color se refiere al ruido **actual**, medido de forma continua a partir de las zonas vacías de cada columna, y no a un nivel absoluto. Por eso los cambios de volumen, AGC, antena o banda no obligan a reajustar Gain: la presentación mantiene un contraste constante frente al ruido de banda que

haya en cada momento. Como en los decodificadores, el audio se toma antes del AGC, la reducción de ruido y el silenciamiento, así que puede silenciar el receptor y seguir mirando.

Qué significan las formas:

- Una **portadora estable** dibuja una línea horizontal recta.
- El **Morse lento** dibuja esa línea como una cadena de segmentos cortos y largos: puntos y rayas que se leen de izquierda a derecha.
- Las **trazas curvadas o a la deriva** corresponden a un oscilador de baliza sin estabilizar que se calienta o se enfría. Es normal, y a menudo es precisamente esa firma de deriva la que permite a un observador habitual reconocer la estación.
- Las **rayas verticales** que cruzan toda la franja son estáticos o ráfagas de ruido local, no señal.

Consejos: - Tenga paciencia. En QRSS 30 una sola columna tarda 5,5 segundos, de modo que un indicativo completo puede necesitar diez minutos o más para cruzar la pantalla. Déjelo funcionando. - Sintonizar desde la lista **Band** ya estrecha el paso de banda alrededor de la ventana. Si sintonizó a mano, hágalo usted: no cambia lo que resuelve la transformada, pero evita que vecinos fuertes accionen el AGC del receptor. - El grabber es independiente de los decodificadores: puede dejarlo funcionando mientras un decodificador trabaja en otra cosa. - Un span ancho a velocidad lenta es la combinación más pesada; toda la transformada se ejecuta en el navegador, así que en una máquina modesta es preferible un span de 50 Hz.

7. WSPR

Qué es: WSPR (Weak Signal Propagation Reporter, pronunciado «whisper») es un modo baliza de señales ultradébiles que cartografía las trayectorias de propagación en HF por todo el mundo. Cada transmisión dura unos 110 segundos y cabe en una ranura de 200 Hz de ancho. El decodificador espera a una ranura completa de 2 minutos alineada con UTC antes de decodificar.

Frecuencias recomendadas (USB, dial)

Banda	Frecuencia de dial
160 m	1.836.600 MHz
80 m	3.568.600 MHz
40 m	7.038.600 MHz
30 m	10.138.700 MHz
20 m	14.095.600 MHz
17 m	18.104.600 MHz
15 m	21.094.600 MHz

Configuración

1. Sintonice una de las frecuencias de dial de WSPR anteriores y ponga el modo en **USB**.
2. La señal WSPR ocupa el margen de audio 1400-1600 Hz. Seleccionar **WSPR** hace el resto por usted: el receptor pasa a USB y el paso de banda se estrecha a 1350-1650 Hz, que cubre todo el margen en el que busca el decodificador. No hace falta ajustarlo más.
3. Active el decodificador y seleccione **WSPR** en el desplegable. O basta con pulsar el botón **WSPR**.
4. El panel **WSPR-2 Decoder** aparece debajo.

Lectura de la salida

El panel muestra una barra de progreso para la ranura de 2 minutos en curso:

- **Barra cian llenándose:** recogida de datos de señal (de 0 a 116 s dentro de la ranura).
- **Barra ámbar parpadeando:** decodificación en curso (los últimos ~4 s de la ranura).
- **Barra vacía:** esperando al siguiente minuto UTC par.

Cada punto (spot) decodificado con éxito se muestra en una tabla con las siguientes columnas:

Columna	Significado
UTC	Hora del spot (minuto par)
Callsign	La estación que transmitió
Grid	Localizador Maidenhead del transmisor
Power	Potencia transmitida en dBm
Freq	Frecuencia de audio exacta (Hz) dentro del paso de banda de WSPR
SNR	Relación señal/ruido en dB

Haga clic en **Clear** para borrar la lista de spots.

Consejo: la decodificación de WSPR requiere una hora del sistema muy precisa (± 1 segundo respecto a UTC). La primera ranura tras activar el decodificador comenzará en el siguiente minuto UTC par; una breve espera es normal.

8. FAX de HF / WEFAX

Qué es: el radiofax de HF (también conocido como WEFAX) lo utilizan los servicios de guardacostas y meteorológicos de todo el mundo para difundir mapas del tiempo, cartas del estado del mar y análisis de superficie por onda corta. El decodificador reconstruye la imagen línea a línea a medida que se recibe.

Configuración

1. Active el decodificador y seleccione **HF FAX / WEFAX** en el desplegable. O basta con pulsar el botón **FAX**.
2. El panel **HF FAX / WEFAX Receiver** aparece debajo.
3. **Seleccione una estación** en el desplegable Station. Hay más de 20 estaciones disponibles que cubren Europa, Asia, Oceanía y América (p. ej., DDH3/DDK3 Alemania, SVJ4/GR Grecia, JMH Japón, NMG EE. UU. Nueva Orleans). También está CBM Punta Arenas (Chile). Los horarios siguen el programa mundial de radiofax de la NOAA (edición del 7 de marzo de 2025); Charleville (VMC, Australia) incluye sus emisiones adicionales solo en 11030 kHz.
4. Si la estación emite en más de una frecuencia, seleccione la deseada en el subdesplegable **Frequency**.
5. Haga clic en ► **Tune** para sintonizar automáticamente la cascada en esa estación.
6. El modo se fuerza automáticamente a **USB**.

Horario de emisiones

Al seleccionar una estación aparece una tabla de cuenta atrás **Next Transmissions** con las 4 próximas emisiones programadas en UTC y una cuenta atrás en directo:

- Normal (gris/verde): la transmisión está próxima.
- **Ámbar** : la transmisión empieza en menos de 3 minutos; prepare ya el decodificador.
- **Rojo** ●: la transmisión empieza en menos de 30 segundos; la recepción es inminente.

Parámetros

La mayoría de las estaciones usan los valores estándar por defecto (marcados con ★). Cámbielos solo si sabe que la estación emplea ajustes no estándar.

Parámetro	Por defecto	Descripción
LPM	120 ★	Líneas por minuto: la velocidad de giro del tambor
IOC	576 ★	Índice de cooperación: determina los píxeles por línea
Shift	800 Hz ★	Desplazamiento de frecuencia entre los tonos de negro y blanco

Controles

- ⇄ **Auto-align**: activado por defecto. Se sincroniza automáticamente con la señal de fase al principio de cada imagen. Desactívelo solo si tiene problemas de alineación con una señal que sabe que es buena.
- ↕ **Invert**: intercambia blanco y negro. Úselo si la imagen aparece en negativo (zonas blancas donde debería haber negro).
- ↻ **Refresh**: borra el lienzo y reinicia el decodificador. Úselo entre transmisiones o si la imagen se rompe o se desplaza.
- ↓ **Save PNG**: guarda el lienzo actual como archivo PNG en su ordenador.

Indicadores de estado

Bajo la imagen, dos indicadores de tono muestran:

- **300 Hz phasing**: se ilumina en cian cuando se detecta el tono de fase de inicio de imagen.
- **450 Hz stop**: se ilumina en rojo cuando se detecta el tono de parada de fin de imagen.

Ambas señales se reconocen por su ritmo a lo ancho de toda la línea, que las señales reales de inicio y fin siempre llenan y el contenido de una carta casi nunca, de modo que las zonas rayadas de una carta no se toman por una señal de inicio. Se reconocen hasta unos 0 dB de SNR, y con señales muy débiles la imagen se alinea con la media de varias líneas de fase, de modo que la carta sigue empezando en el borde izquierdo.

Nota: la imagen se desplaza hacia arriba; la línea recibida más reciente siempre aparece en la parte inferior del lienzo. Si ve **[PHASING]** en la cabecera, el decodificador se ha enganchado al inicio de una nueva imagen.

9. NAVTEX

Qué es: NAVTEX es el sistema internacional de radiodifusión marítima de información de seguridad costera: avisos a la navegación, previsiones meteorológicas y avisos de búsqueda y rescate. Utiliza FSK a 100 baudios (SITOR-B con FEC) y se recibe en canales dedicados en todo el mundo.

Canales disponibles

Canal	Frecuencia	Uso
Internacional	518 kHz	En inglés, internacional
Nacional	490 kHz	Emisiones en el idioma nacional
HF (x5)	4209.5 / 6314 / 8416.5 / 12579 / 16806.5 kHz	NAVTEX de HF de largo alcance

Configuración

1. Active el decodificador y seleccione **NAVTEX** en el desplegable. O basta con pulsar el botón **NAVTEX**. El receptor pasa automáticamente a **USB**.
2. Aparece el panel **NAVTEX Receiver**.
3. Seleccione el canal deseado en el desplegable **Station** (p. ej., `International – 518 kHz`).
4. Haga clic en **⇒ Tune & Set IF** para sintonizar automáticamente la cascada y estrechar el paso de banda hasta la ventana de audio correcta. El modo se pone automáticamente en **USB**. El dial se sitúa 500 Hz por debajo del centro del canal, de modo que la señal NAVTEX aparece a 500 Hz en audio.

Horario de emisiones

La tabla de horarios enumera todas las estaciones conocidas del canal seleccionado con:

- su letra identificadora de la UIT y la bandera del país
- la hora de la próxima emisión en UTC
- una cuenta atrás en directo hasta la próxima transmisión
- **Ámbar** a menos de 2 minutos / **Rojo** ● a menos de 30 segundos: prepare ya el decodificador

Lectura de la salida

El texto decodificado aparece en color verde azulado y paso fijo. Los límites de los mensajes están claramente marcados:

```
— ZCZC MA12 —
... message content ...
— NNNN —
```

ZCZC marca el inicio de un mensaje. Los cuatro caracteres siguientes identifican la estación (M), el tema (A = avisos a la navegación) y el número de secuencia (12). El encabezado espera a tener la cabecera completa, así que muestra ZCZC MA12 y no ZCZC ???? . NNNN marca el final.

Si sintoniza cuando un mensaje ya ha empezado, o su cabecera se pierde en el ruido, el texto aparece igualmente, bajo un encabezado **— (header missed) —**. Solo se abre cuando llegan seguidos varios caracteres en los que coinciden las dos transmisiones NAVTEX, algo que el ruido casi nunca produce. Se cierra en NNNN como siempre, con **— (signal lost) —** si la señal desaparece antes, o cuando empieza un mensaje nuevo.

Haga clic en **Clear** para borrar el búfer de mensajes.

Nota: NAVTEX funciona en frecuencias de MF y LF (518/490 kHz). El alcance de recepción suele ser de 200 a 400 millas náuticas desde el transmisor. Los canales de HF (4-17 MHz) ofrecen un alcance mucho mayor.

10. FSK / RTTY — incluidos PSK31 y Olivia

Qué es: un decodificador de uso general para modos de texto de banda estrecha, con cinco variantes de funcionamiento seleccionables desde un único menú. Tres son FSK auténtico (modulación por desplazamiento de frecuencia): FSK marítimo (SITOR), RTTY meteorológico y RTTY de aficionado. Las otras dos no son FSK en absoluto, pero comparten la misma ventana: **PSK31**, que es modulación por desplazamiento de fase, y **Olivia**, que es FSK multitono con corrección de errores. Cada variante incluye un preajuste adaptado a sus parámetros estándar.

Variantes y preajustes

Variante	Centro	Shift	Baudios	Trama	Codificación
FSK marítimo / SITOR	500 Hz	170 Hz	100	7N1	CCIR-476
RTTY meteorológico	1000 Hz	450 Hz	50	5N1.5	ITA2
RTTY de aficionado	1000 Hz	170 Hz	45.45	5N1.5	ITA2
PSK31 (BPSK)	1000 Hz	—	31.25	—	Varicode
Olivia (MFSK)	1000 Hz	—	según Mode	—	7 bits + FEC

El panel se adapta a la variante elegida. **Shift**, **Baud**, **Framing**, **Encoding**, **Invert mark / space** y **Auto shift detect** se ocultan para PSK31 y Olivia, porque ninguno de los dos modos tiene un par de tonos mark/space ni trama de tipo UART. En su lugar, **Center audio** pasa a ser un campo numérico de entrada libre (la portadora puede situarse en cualquier punto del paso de banda), y Olivia añade un selector **Mode** y un deslizador **Squelch**.

Configuración

1. Active el decodificador y seleccione **FSK / RTTY** en el menú. O simplemente pulse el botón **RTTY**.
2. Aparece el panel del decodificador. Su título sigue a la variante — *FSK / RTTY Decoder*, *PSK31 Decoder* u *Olivia Decoder*.
3. Seleccione la **Variante**. Los parámetros se actualizan automáticamente.
4. Para Olivia, ajuste **Mode** (tonos / ancho de banda) para que coincida con la transmisión — véanse las notas sobre Olivia más abajo.
5. Utilice el menú **Known frequency** para elegir una frecuencia habitual de la variante seleccionada y pulse **Tune** para saltar a ella.
6. Afine la sintonía hasta que el texto decodificado sea estable y legible.

Frecuencias conocidas por variante

FSK marítimo / SITOR

- 518,0 kHz — NAVTEX internacional
- 490,0 kHz — NAVTEX nacional
- 4209,5 / 6314,0 / 8416,5 / 12579,0 / 16806,5 / 22376,0 kHz — SITOR en HF

RTTY meteorológico

- 4583,0 / 7646,0 / 10100,8 / 11039,0 / 14467,3 kHz — DWD (Servicio Meteorológico Alemán)

RTTY de aficionado

- 3590 kHz (80 m), 7043 kHz (40 m), 10143 kHz (30 m), 14083 kHz (20 m), 21083 kHz (15 m), 28083 kHz (10 m)

PSK31

- 3580,15 kHz (80 m), 7040,15 kHz (40 m), 10142,15 kHz (30 m), 14070,15 kHz (20 m), 18100,15 kHz (17 m), 21080,15 kHz (15 m), 24920,15 kHz (12 m), 28120,15 kHz (10 m)

Olivia

- 3577,75 kHz (80 m), 7073,75 kHz (40 m), 10142,25 kHz (30 m), 14075,5 kHz (20 m), 18103,75 kHz (17 m), 21075,75 kHz (15 m), 24921,75 kHz (12 m), 28123,75 kHz (10 m)

Parámetros

Parámetro	Se aplica a	Descripción
Center audio (Hz)	todas	La frecuencia de audio del punto medio entre mark y space; en PSK31 la portadora, en Olivia el centro del bloque de tonos. Un menú desplegable en las variantes FSK, un campo de entrada libre en PSK31 y Olivia
Shift (Hz)	solo FSK	Diferencia de frecuencia entre los tonos mark y space
Baud	solo FSK	Velocidad de símbolo
Framing	solo FSK	Bits de datos, paridad, bits de parada (p. ej. 7N1 = 7 datos, sin paridad, 1 parada)
Encoding	solo FSK	Juego de caracteres (CCIR-476, ITA2/Baudot o ASCII)
Invert mark / space	solo FSK	Intercambia los tonos mark y space
Auto shift detect	solo FSK	Intenta medir el shift automáticamente a partir de la señal recibida
Mode (tonos / Hz)	solo Olivia	Número de tonos y ancho de banda — debe coincidir exactamente con la transmisión
Squelch (FEC S/N)	solo Olivia	Cuán fuerte debe ser la coincidencia de la corrección de errores antes de imprimir texto
Squelch (SNR)	RTTY de aficionados y meteorológico	Evita que el ruido se imprima como letras al azar. Empieza en -5 dB para el RTTY de aficionados y en -8 dB para el meteorológico; del todo a la izquierda (off) queda desactivado. La cifra de SNR se vuelve gris mientras el squelch está cerrado

Métricas de señal

La barra de estado muestra medidas en tiempo real, y los campos cambian según la variante:

Variante	Campos mostrados
Variantes FSK	Mark / Space (frecuencias de tono medidas), SNR , Lock , Timing
PSK31	Carrier (Hz, tras la corrección automática de frecuencia), IMD (dB), SNR , Lock , Timing
Olivia	Centre (Hz), Mode , S/N , FEC (%), Sync

Timing/Sync indica LOCKED/SYNCED cuando el decodificador sigue la señal, y SEARCH mientras aún la busca.

La **SNR** de las variantes FSK y de PSK31 se mide sobre la propia señal y se da en 3 kHz de ancho de banda, así que ambas se leen en la misma escala: con un error de 1 dB aproximadamente de $+20$ dB a -5 dB. Una frecuencia vacía marca unos -12 a -15 dB, y en PSK31 0 dB ya es una señal fuerte. La **S/N** de Olivia es otra cosa: la cifra de calidad de la corrección de errores.

El RTTY de aficionados (45,45 baudios, 170 Hz) se copia sin errores hasta unos 0 dB, con alrededor de un 1,5 % de caracteres erróneos a -3 dB y un 20 % a -6 dB.

Controles adicionales

- ⇒ **Set IF Band-Pass** — estrecha la banda de paso del receptor para ceñir la señal. El ancho sigue a la variante: 300 Hz para el RTTY de aficionados (lo bastante estrecho para quitar una estación más fuerte a 250 Hz y aun así copiar una señal desintonizada 40 Hz), 594 Hz para el RTTY meteorológico, mark/space más margen para el FSK marítimo, unos ± 100 Hz para PSK31 y todo el ancho del bloque de tonos más margen para Olivia. La banda de paso sigue al decodificador cuando **Auto-tune** o **Auto shift detect** mueven los tonos, y cuando se cambia el ajuste **Baud**.
-  **Auto-tune Center** — búsqueda automática de la señal. En las variantes FSK busca un par equilibrado de tonos, en PSK31 localiza la portadora, y en Olivia encuentra el bloque más fuerte del ancho de banda seleccionado. **Auto shift detect** solo cambia el desplazamiento cuando hay una verdadera señal de dos tonos, así que ya no salta con el ruido.

Notas sobre PSK31

PSK31 es el modo de teclado a teclado más común en HF. Ocupa solo 62 Hz, de modo que varios QSO conviven uno junto a otro dentro de unos pocos cientos de hercios alrededor de la frecuencia de reunión, y sintonizar consiste en elegir una traza del grupo en la cascada.

- El decodificador corrige por sí mismo su error de sintonía en unos **± 25 Hz**, así que basta con acercarse. **Carrier**, en la fila de métricas, indica dónde se ha estabilizado realmente.
- **IMD** mide la calidad del transmisor, no la recepción: una señal limpia da unos -20 dB o mejor. Una lectura pobre significa que la otra estación está sobreexcitando su equipo, no que usted esté mal sintonizado.
- El texto aparece carácter a carácter sin corrección de errores, por lo que una señal débil se degrada en letras erróneas ocasionales en lugar de detenerse.

Notas sobre Olivia

Olivia sacrifica velocidad a cambio de robustez. Es mucho más lento que PSK31, pero decodifica señales inaudibles al oído, lo que lo hace popular para señales débiles y contactos a larga distancia.

- **El ajuste Mode debe coincidir exactamente con la transmisión.** Una combinación errónea de tonos y ancho de banda no decodifica absolutamente nada — no texto ilegible, sino silencio. El panel se abre en **8 / 250**, la configuración estrecha que se deja funcionando en las frecuencias de llamada; **16 / 500** y **32 / 1000** son las otras dos de uso común, y también se ofrece **16 / 1000**.
- Olivia no envía preámbulo, por lo que el decodificador debe buscar la sincronización. **Deje pasar unos segundos** tras sintonizar antes de que aparezca texto. El campo **Sync** muestra SEARCH hasta que engancha.
- La corrección de errores trabaja por bloques, así que el texto llega **a ráfagas en lugar de en flujo continuo**, con un retardo de varios bloques entre la transmisión y su presentación.
- **Squelch (FEC S/N)** fija la confianza que debe alcanzar la corrección de errores antes de imprimir. El valor por defecto de 4,0 mantiene fuera el ruido; 3,0 es el mínimo, por debajo del cual el ruido aleatorio empieza a imprimir caracteres sueltos. Una buena señal marca 8-9 en el indicador **FEC**, de modo que hay margen de sobra para subir el squelch en una banda concurrida.

Nota sobre el modo: el decodificador toma el control del modo de demodulación y del paso de banda de FI mientras está activo. Ambos se restauran automáticamente al desactivarlo. Las cinco variantes usan **USB**.

Nota sobre la polaridad (solo variantes FSK): en RTTY meteorológico normalmente hay que marcar **Invert mark / space**. En FSK marítimo (tipo SITOR/NAVTEX) y RTTY de aficionado, déjelo sin marcar — el RTTY de aficionado envía mark como la radiofrecuencia más alta, y USB la mantiene como el tono de audio más alto, que es justamente el caso sin marcar. Si el texto decodificado sale ilegible, lo primero que conviene probar es cambiar esta casilla. La casilla se oculta en PSK31 y Olivia, que no tienen par mark/space.

Nota sobre letras/cifras (solo variantes FSK): el código Baudot mantiene letras y cifras en dos estados separados, y el ruido puede llevar al decodificador al estado equivocado, lo que corrompe todos los caracteres siguientes y no solo el dañado. Por eso el decodificador vuelve a letras en cada espacio; es la práctica habitual y repara un cambio corrompido en una o dos palabras en lugar de una línea entera. El coste es que los grupos de cifras separados por espacios exigen que el transmisor repita el cambio a cifras después de cada espacio, como suelen hacer los equipos.

11. SSTV

Qué es: la televisión de barrido lento transmite imágenes fijas por un canal de voz SSB normal, línea a línea, como un tono modulado en frecuencia entre 1500 Hz (negro) y 2300 Hz (blanco). Una imagen completa tarda entre 36 segundos y 4 minutos y medio según el modo.

Frecuencias recomendadas (USB)

Banda	Frecuencia	Modo	Notas
20 m	14.230 MHz	USB	La principal frecuencia internacional de llamada SSTV, con diferencia la más activa
20 m	14.233 MHz	USB	Secundaria, se usa cuando 14.230 está ocupada
15 m	21.340 MHz	USB	
10 m	28.680 MHz	USB	Activa durante las aperturas de banda
40 m	7.171 MHz	LSB	
80 m	3.845 MHz	LSB	Regional, por las tardes

La banda lateral se elige por usted. Al iniciar el decodificador se selecciona **LSB por debajo de 10 MHz** y **USB por encima**, siguiendo la práctica habitual de radioafición. En la banda lateral equivocada la correspondencia de tonos queda invertida y la imagen no se decodificará. Si se encuentra con una estación que ignora la convención, cambie la banda lateral a mano: el decodificador sigue funcionando, borra el cuadro y empieza de nuevo con el nuevo ajuste.

Configuración

1. Active el decodificador y seleccione **SSTV** en el desplegable. O basta con pulsar el botón **SSTV**. El receptor cambia de banda lateral automáticamente: USB por encima de 10 MHz, LSB por debajo.
2. Sintonice de modo que los tonos de la imagen caigan en el centro del paso de banda. Una señal bien sintonizada tiene sus impulsos de sincronismo a 1200 Hz y el contenido de imagen entre 1500 y 2300 Hz.
3. Deje **Mode** en **Auto** salvo que ya sepa qué se está enviando. La imagen se va formando línea a línea según se recibe.

Modos

Ajuste	Imagen	Duración
Auto	Detectado automáticamente	—
Martin M1 / M2	320×256 color	114 s / 58 s
Scottie S1 / S2	320×256 color	110 s / 71 s
Scottie DX	320×256 color	269 s
Robot 36 / 72	320×240 color	36 s / 72 s

Auto funciona de dos maneras: lee la **cabecera VIS** —el código digital del modo enviado en los primeros 300 ms de una transmisión— y, si se perdió la cabecera (sintonizó tarde o se perdió por QSB), identifica el modo a partir de la temporización de los impulsos de sincronismo. Seleccionar un modo concreto lo fuerza, pero el decodificador sigue verificando el sincronismo antes de dibujar nada, de modo que una elección equivocada no produce imagen en lugar de ruido.

Lectura de la salida

La línea de estado bajo los controles informa de lo que está haciendo el decodificador:

Estado	Significado
Waiting for VIS / AUTO lock	Escuchando; aún no se ha identificado nada
Martin M1? verifying sync...	Se ha encontrado un candidato y se está confirmando con las líneas siguientes
Martin M1 lock (VIS)	Enganchado a partir de la cabecera VIS
Martin M1 lock (AUTO)	Enganchado a partir de la temporización de sincronismo
Martin M1 lock (MANUAL)	Iniciado con el botón ● Force
Candidate rejected (no sync)	El candidato no se confirmó; normal con ruido
— sync lost, resetting	La señal desapareció a mitad de imagen
Frame complete	Se recibió la imagen completa

El modo detectado también aparece en verde junto al título **SSTV**, y el contador de líneas muestra el avance.

Controles

Botón	Acción
► Start	Arma el decodificador. No dibuja una imagen por sí solo: el enganche debe venir de la cabecera VIS o de la detección de sincronismo.
■ Stop	Detiene la decodificación.
↺ Reset	Borra el lienzo y vuelve a armar en el sitio. Úsalo entre imágenes o después de resintonizar.
● Force	Empieza a dibujar de inmediato en el modo seleccionado en el desplegable, omitiendo por completo el VIS y la detección de sincronismo. Solo está disponible cuando Mode no está en Auto .
Save	Guarda la imagen actual como PNG.

Cuándo usar Force. Si ve una imagen en la cascada pero el decodificador no consigue engancharse — un modo poco común, una señal demasiado débil o distorsionada para los detectores, o una cabecera que se perdió —, seleccione el modo a mano y pulse **● Force**. El cuadro se ancla en el momento del clic, y la etiqueta de modo muestra **MANUAL** para que lo distinga de un enganche por VIS o AUTO. El decodificador sigue ajustándose a un impulso de sincronismo real si lo encuentra, así que un clic algo adelantado o retrasado se corrige.

Como Force omite todas las comprobaciones de seguridad, pintará ruido sin más si lo pulsa en un canal vacío, y no se detendrá solo: pulse **■ Stop** o **↺ Reset**.

Notas

- **El ruido no arranca el decodificador.** Antes bastaban los chasquidos atmosféricos, las chispas y el zumbido de la red para disparar una decodificación. Ahora cada etapa de detección comprueba que el tono sea un tono real y confirma el candidato con las líneas siguientes antes de dibujar un solo píxel. Es normal que el decodificador permanezca en silencio en una banda vacía.
- **La imagen se inclina en diagonal si está fuera de frecuencia.** SSTV no perdona los errores de sintonía. Si las líneas se tuercen, ajuste el dial en pequeños pasos y deje que empiece la siguiente imagen.
- El equilibrio de color y la nitidez son fijos; no hay nada que ajustar.

12. Consejos generales

Primero hay que activar Decoder: ON. El desplegable está desactivado (en gris) hasta que hace clic en el botón Decoder.

Un decodificador cada vez. Seleccionar un nuevo decodificador en el desplegable detiene automáticamente el que estuviera activo y deshace los cambios de modo o paso de banda que hubiera hecho.

El modo se gestiona por usted. El FAX de HF y NAVTEX ponen el receptor en USB en cuanto los selecciona, SSTV elige la banda lateral según la banda (LSB por debajo de 10 MHz, USB por encima) y FAX, NAVTEX y FSK también ajustan el paso de banda al pulsar su botón Tune. FT8, FT4, FT2, JS8 y WSPR hacen lo mismo en cuanto los selecciona — desde la fila de botones o desde el desplegable: el receptor pasa a USB y toma su propio paso de banda — la subbanda completa de 3 kHz para la familia FT8, 1350-1650 Hz para WSPR. Al detener el decodificador se restaura el modo por defecto de la banda.

Activar un decodificador ya no interrumpe el audio. Los decodificadores se ejecutan en sus propios hilos, así que no hay huecos, chasquidos ni cortes al iniciar, detener o cambiar de decodificador.

La precisión del reloj del sistema importa. FT8, FT4 y WSPR son críticos en el tiempo. Decodifican en ventanas fijas alineadas con UTC. Si el reloj del ordenador se desvía más de 1 o 2 segundos, la tasa de decodificación caerá notablemente. Use un cliente NTP para mantenerlo preciso.

Los filtros de ruido no llegan a los decodificadores. NR, NB, NS y AN son ayudas de escucha, solo para sus oídos. Todos los decodificadores — FT8, FT4/FT2, CW, WSPR, FAX, NAVTEX, FSK/RTTY/PSK31/Olivia, SSTV y el grabber QRSS — toman el audio *antes* de esos filtros, así que ajústelos como mejor suene, sin preocuparse por la calidad de la decodificación. Por la misma razón los decodificadores siguen funcionando con el receptor silenciado o con el squelch cerrado: puede apagar el altavoz y dejar un decodificador, o una captura QRSS nocturna, recogiendo datos. Lo único que sí sigue lo que usted oye es el Espectrograma de Audio, pensado para mostrar el audio filtrado.

La calidad de la señal importa más que su fuerza. La mayoría de estos decodificadores están pensados para señales débiles. Una banda más tranquila y con menos ruido suele ser más productiva

que una señal fuerte llena de interferencias. Use la cascada y los controles de paso de banda para identificar y evitar el QRM antes de activar un decodificador.

Use los botones Refresh o Clear sin reparos. Las imágenes de FAX se desplazan si la frecuencia de dial está algo desviada, y los decodificadores de texto acumulan caracteres espurios. Empezar de cero tras ajustar la sintonía suele producir una salida mucho más limpia.

Reporte automático de spots y gráficas del servidor

Las decodificaciones de FT8, FT4 y WSPR también puede subirlas el propio servidor — FT8/FT4 a PSK Reporter y WSPR a WSPRnet — mediante un demonio autorun que el sysop inicia desde el panel de administración. Es independiente de los decodificadores de su navegador: sigue funcionando haya o no oyentes, y nada de lo que usted decodifique en el navegador se reporta.

El sysop lo sigue con dos contadores, fáciles de confundir entre sí: los paneles por decodificador cuentan los spots subidos desde el último arranque del demonio, mientras que el número junto a cada casilla de banda/modo es el total histórico de esa ranura y sobrevive a los reinicios. El mismo panel tiene una página **Gráficos** que representa la frecuencia de CPU, la carga, la temperatura y los usuarios conectados en los últimos 15 minutos a 24 horas. Ambos se describen en la [guía del panel de administración](#).

PhantomSDR-Plus — fork de sv1btl — phantomsdr.no-ip.org

CAPÍTULO 9

RADE README

RADE_README.md

Voz digital RADE v1 para PhantomSDR-Plus

RADE (Radio AutoencoDEr) es el modo estrella de voz digital en HF de FreeDV. Utiliza un híbrido de aprendizaje automático y DSP (el vocoder neuronal FARGAN) para ofrecer voz de alta calidad por HF con relaciones señal/ruido de hasta -2 dB, en apenas 1500 Hz de ancho de banda de RF: más estrecho que una señal de BLU.

Este documento cubre la integración completa de la recepción de RADE v1 en PhantomSDR-Plus, implementada como un proceso sidecar de Python (`rade_helper.py`) que conecta el navegador con la cadena de decodificación `radae_rxe.py` + `lpcnet_demo`.

La instalación — la vía rápida

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

Esa es toda la instalación. El script instala los paquetes del sistema con el gestor de paquetes que tenga la máquina (`apt`, `pacman`, `dnf` o `zypper`), recurre a `pip` para lo que una distribución no incluya, actualiza `websockets` si la versión de la distribución es anterior a la 11.0 que necesita el sidecar, clona y compila `radae`, comprueba los pesos del modelo, recompila el frontend y arranca el sidecar. En Ubuntu 22.04 pasa el testigo a `install_rade_ubuntu22.sh`. `./install.sh` ofrece ejecutarlo como parte de la instalación normal de PhantomSDR-Plus, así que en una máquina nueva RADE ya está.

Este documento explica cómo funciona RADE y cómo hacerlo funcionar; no es el procedimiento de instalación. Si necesita instalarlo a mano, en un sistema que el script no cubre o para reparar un paso, siga la guía enlazada más abajo: los pasos están escritos allí y solo allí. Todo lo que hay aquí — la arquitectura, los archivos parcheados, el puerto, el uso de RADE en el navegador, las variables de entorno, la resolución de problemas y la medición de concurrencia — se aplica a cualquier instalación, sea como fuere que se hizo.

- **Instalación manual en Linux** - la vía manual, paso a paso, para Ubuntu, Debian, Fedora, Arch y Raspberry Pi OS

Índice

1. La instalación — la vía rápida
2. Cómo funciona
3. Descripción de la arquitectura
4. RADEL frente a RADEU
5. Requisitos previos
6. Instalación a mano
7. Desplegar los archivos parcheados
8. Control del sidecar
9. El puerto 8074

10. Usar RADE en el navegador
 11. Verificación y depuración
 12. Variables de entorno
 13. Archivos modificados
 14. Resumen del flujo de señal
 15. Resolución de problemas
 16. Medir la concurrencia en su propio hardware
 17. Actualizar RADE v1
-

Cómo funciona

RADE v1 no puede ejecutarse en el navegador: necesita PyTorch y el vocoder neuronal FARGAN, demasiado grandes para WASM. La solución es un proceso sidecar de Python (`rade_helper.py`) que se ejecuta en el mismo servidor que PhantomSDR-Plus.

Importante: `freedv_rx` del repositorio `codec2` **no** admite RADEV1. La cadena de decodificación de RADE reside por completo en el repositorio independiente `radae` de David Rowe (VK5DGR). No intente usar `freedv_rx` de `codec2` para RADE.

Cuando selecciona RADE en el navegador:

1. El frontend fija la demodulación subyacente en USB o LSB: el servidor en C++ demodula la señal BLU con normalidad.
2. El PCM demodulado sin procesar se toma en `audio.js` **antes** de cualquier silenciado o puerta de squelch: `radae_rxe.py` necesita una entrada continua para mantener el sincronismo de trama.
3. Cada bloque de PCM se rellena con ceros para pasar de f32 real a f32 complejo (real + 0.0 imaginario): eso es lo que `radae_rxe.py` espera en su entrada estándar.
4. El sidecar canaliza esas muestras hacia `radae_rxe.py`, que produce características de vocoder.
5. `lpcnet_demo -fargan-synthesis` convierte las características en voz s16 a 16000 Hz.
6. El sidecar convierte s16 → f32 y lo devuelve al navegador en tramas binarias de WebSocket.
7. El navegador reproduce la voz decodificada mediante la Web Audio API a 16000 Hz.

El servidor en C++ (`spectrumserver.cpp`) no se modifica en absoluto.

Descripción de la arquitectura



RADEL frente a RADEU

RADE v1 siempre se transmite por BLU. Por convención:

Modo	Banda lateral	Usar en las bandas
RADEL	LSB	160 m, 80 m, 40 m (≤ 10 MHz)
RADEU	USB	20 m, 17 m, 15 m, 12 m, 10 m (> 10 MHz)

Requisitos previos

Requisito	Versión	Notas
PhantomSDR-Plus	cualquiera	con frontend Vite/Svelte
Linux	Ubuntu 24.04+ / Debian Bookworm+	probado
Python	3.8+	para <code>rade_helper.py</code> y <code>radae_rxe.py</code>
repositorio radae	último master	aporta <code>radae_rxe.py</code> y <code>lpcnet_demo</code>
cmake	3.10+	para compilar <code>lpcnet_demo</code> desde radae
PyTorch	2.0+	lo requiere <code>radae_rxe.py</code>
Node.js	16+	para <code>npm run build</code>
websockets (Python)	10-16+	<code>pip3 install websockets</code> — todas las versiones admitidas
matplotlib	cualquiera	lo requiere <code>radae_rxe.py</code> al importarse
numpy	1.23+	obligatorio; también habilita el remuestreo
scipy	cualquiera	opcional, habilita un remuestreo preciso

Instalación a mano

`./install_rade.sh` hace la instalación completa. Cuando quiera hacerla usted mismo — en un sistema que el script no cubre, o para reparar un paso — el procedimiento está en la [Guía de instalación manual de RADE v1](#), y solo allí. Cubre Ubuntu, Debian, Fedora, Arch y Raspberry Pi OS de una sola pasada:

Paso 1	Paquetes del sistema y la comprobación de Node.js 22+
Paso 2	Paquetes de Python, con la opción de la PEP 668 y la rueda de torch solo para CPU
Paso 3	Clonar y compilar el repositorio radae, verificar <code>lpcnet_demo</code> y los pesos del modelo
Paso 4	Verificar la cadena de decodificación sin conexión, antes de conectarla al navegador
Paso 5	Compilar el frontend de PhantomSDR-Plus
Paso 6	Control del sidecar
Paso 7	Abrir el puerto 8074
Pasos 8-10	Arrancar el servidor, verificar RADE y usarlo en el navegador

El resto de este documento da eso por hecho.

Desplegar los archivos parcheados

Copie los siguientes archivos del conjunto de parches a su repositorio de PhantomSDR-Plus.

Archivo nuevo — colóquelo en la raíz del repositorio, junto al script de arranque que use:

```
rade_helper.py
```

Archivos parcheados del frontend — colóquelos en `frontend/src/`:

```
audio.js
App.svelte
```

Qué hacen los parches

audio.js — 5 cambios:

- Constructor: 5 campos de estado nuevos para RADE (`decodeRADE`, `_radeSideband`, `_radeSocket`, `_radeCallback`, `_radeReady`, `_radeNextTime`)
- Guardado del PCM previo al realce: `pcmArrayPreBoost` se guarda antes del realce de 300× de FLAC para que RADE reciba la amplitud original (el realce saturaría `rade_rxe.py`)
- `playAudio()`: toma del PCM para RADE usando el audio previo al realce, antes de la puerta de silencio/squelch
- `playAudio()`: guarda `if (this.decodeRADE) return`, que suprime la reproducción del BLU sin procesar mientras suena la voz decodificada por RADE
- Métodos nuevos: `setRADEDecoding()`, `setRADECallback()`, `_radePlayPCM()` con reproducción programada y sin huecos mediante el reloj `_radeNextTime` (reproduce a **16000 Hz**, la tasa de salida de `lpcnet_demo`)

Cada variante de Svelte — 6 cambios:

- `demodulationDefaults`: `RADEL {type: 'LSB', offsets: [2200, -700]}`, `RADEU {type: 'USB', offsets: [-700, 2200]}`: el paso de banda empieza a 700 Hz de la portadora y mide 1500 Hz
- Variables de estado: `radeEnabled`, `radeConnected`, `radeSynced`, `radeSnr`, `_radeDeactivate()`
- `_radeDeactivate()`: detiene el decodificador y **restaura el modo por defecto de la banda** desde `bands-config.js`, con un comportamiento idéntico a la desactivación de FAX, NAVTEX y FSK
- `_deactivateAll()`: línea de limpieza de RADE
- `activateSelectedDecoder()`: ramas `radel` y `radeu`
- Desplegable de decodificadores: dos entradas `<option>` nuevas
- Panel de estado: punto de conexión, estado de sincronismo/SNR, cartel de error

Scripts de arranque

Los lanzadores del receptor (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) arrancan el sidecar por sí mismos en cuanto el servidor está en marcha, y `stop-websdr.sh` lo detiene junto con el servidor. `rade.sh` se mantiene para ejecutar el sidecar de forma autónoma: consulte [Control del sidecar](#).

Control del sidecar

Normalmente no necesita un script de control aparte para RADE. Los lanzadores del receptor (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) arrancan el sidecar por sí mismos en cuanto se confirma que el servidor está en marcha, lo reinician si termina e informan de su estado en el resumen final —

✓ RADE sidecar running, o una nota `·` que explica por qué no se activó (no instalado, `RADE_ENABLED=0`, o

sin python3). `./stop-websdr.sh` lo detiene junto con el servidor. Para ejecutar el servidor sin RADE, arránquelo con `RADE_ENABLED=0`.

`rade.sh` se mantiene para uso autónomo: ejecutar el sidecar sin el lanzador de PhantomSDR-Plus, o probarlo por separado:

```
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
```

La actividad del sidecar se registra en `~/PhantomSDR-Plus/rade.log`:

```
tail -f ~/PhantomSDR-Plus/rade.log
```

No mezcle ambos con el servidor en marcha. Si el vigilante de un lanzador está activo, devolverá el sidecar en unos cinco segundos, de modo que `./rade.sh stop` parecerá haber fallado. Para detener RADE con el servidor en marcha, arranque el servidor con `RADE_ENABLED=0` o detenga todo con `./stop-websdr.sh`.

Si ejecuta `rade.sh` de forma autónoma, **use siempre `./rade.sh stop`**: un simple `kill -f rade_helper.py` no vence a su propio vigilante, que reengendra el proceso en 3 segundos.

El puerto 8074

El sidecar escucha en el puerto TCP **8074** y el navegador se conecta directamente a él, así que el puerto tiene que ser accesible desde fuera. Cómo abrirlo — `ufw`, `firewalld`, `iptables`, la regla NAT del router y el proxy de Nginx para cuando un proveedor bloquea el puerto por completo — es el [Paso 7 de la guía manual](#).

Dos cosas merecen repetirse aquí, porque son las que más tiempo cuestan:

```
# Is the sidecar listening?
ss -tlnp | grep 8074
```

Aviso sobre el NAT hairpin: probar con `curl` desde el propio servidor contra su nombre público suele devolver `Connection refused` incluso con el puerto abierto, porque muchos routers no devuelven el tráfico. Pruebe siempre desde una máquina externa, o use <https://portchecker.co>.

Usar RADE en el navegador

1. Abra su interfaz web de PhantomSDR-Plus
2. Busque estaciones activas en qso.freedv.org
3. Sintonice la frecuencia de dial de la estación
4. En **Decoder Options**, seleccione:
 5. **RADE v1 — RADEL (LSB)** para 40 m / 80 m / 160 m
 6. **RADE v1 — RADEU (USB)** para 20 m / 17 m / 15 m / 12 m / 10 m
7. Haga clic en **Decoder: ON**

Los botones RADEL / RADEU (una pulsación)

Los pasos 4 y 5 pueden sustituirse por una sola pulsación. Hay un par de botones **RADEL / RADEU** en tres lugares:

- junto al encabezado **Modes selector** del panel principal;
- dentro de la ventana emergente **Modes**;
- dentro de la ventana emergente **Bands**.

Al pulsar uno de ellos se selecciona el decodificador, se pone el decodificador en ON, se cierra la ventana emergente desde la que se pulsó y el panel de RADE se desplaza hasta quedar visible. El botón se pone azul mientras RADE funciona. **Pulse el mismo botón otra vez para detener RADE**: el panel se cierra y el modo y el paso de banda vuelven a su valor tal como se describe más abajo. Los botones, el desplegable **Decoder Options** y el botón ON/OFF comparten el mismo estado.

Estados del indicador del panel

Indicador	Significado
Rojo — «Connecting to sidecar...»	Puerto 8074 inalcanzable o sidecar detenido
Amarillo — «Searching for signal...»	Sidecar conectado, aún sin trama RADE detectada
Verde — «Synced · SNR x.x dB»	Decodificando: se está reproduciendo la voz

Cuando desactiva el decodificador

El modo y el paso de banda vuelven automáticamente al valor correcto por defecto para la frecuencia actual, según se define en `bands-config.js`, igual que con FAX, NAVTEX y FSK.

Verificación y depuración

¿Está funcionando el sidecar?

```
ps aux | grep rade_helper | grep -v grep
ss -tlnp | grep 8074
```

Observar la actividad en directo

```
tail -f ~/PhantomSDR-Plus/rade.log
```

Cuando se conecta un navegador:

```
[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)
```

Prueba rápida del 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
```

Consola del navegador (F12)

```
[RADE] ► ENABLED LSB @ 12000 Hz → helper ws://localhost:8074
```

Bien: el WebSocket se ha abierto. El valor en Hz coincide con el `audioOutputSps` de su servidor (normalmente 8000-12000 Hz).

```
[RADE] sidecar socket error
```

El puerto 8074 es inalcanzable: revise el cortafuegos y la regla NAT del router.

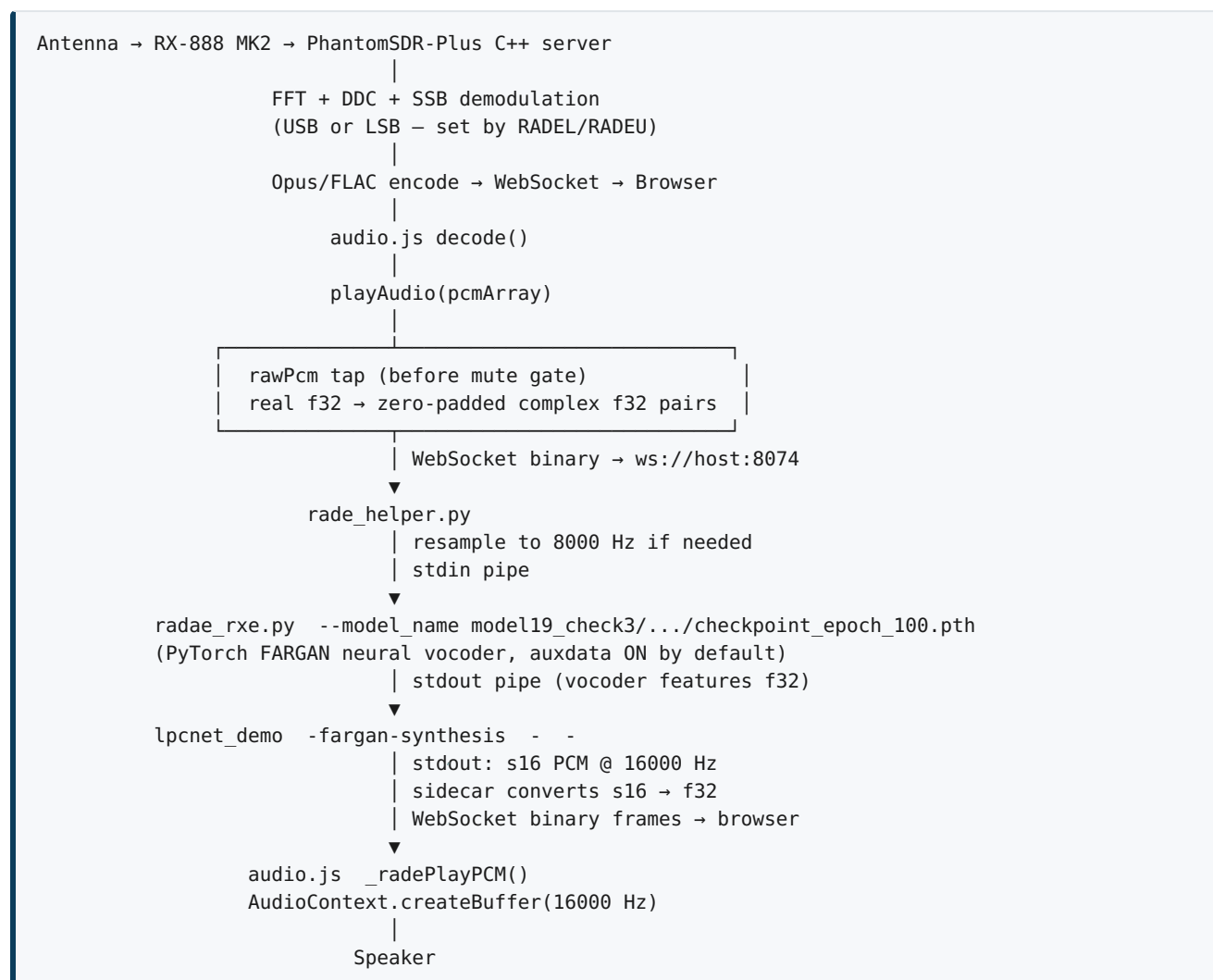
Variables de entorno

Variable	Por defecto	Descripción
<code>RADE_HELPER_PORT</code>	8074	Puerto TCP en el que escucha el sidecar
<code>RADE_HELPER_HOST</code>	0.0.0.0	Dirección de enlace (127.0.0.1 si está tras un proxy)
<code>RADAE_DIR</code>	~/radae	Raíz del repositorio radae
<code>RADE_MODEL</code>	RADAE_DIR/model19_check3/ checkpoints/checkpoint_epoch_100.pth	Pesos del modelo
<code>LPCNET_DEMO</code>	RADAE_DIR/build/src/lpcnet_demo	Binario lpcnet_demo
<code>RADE_AUXDATA</code>	1	Póngalo a 0 para pasar <code>--noauxdata</code> a <code>radae_rxe.py</code>
<code>RADE_TORCH_THREADS</code>	1	Hilos de PyTorch/OpenBLAS por instancia de <code>radae_rxe.py</code> : limita el uso de CPU
<code>RADE_PIN_CORES</code>	1	Fija cada cadena de decodificación a su propio conjunto de núcleos; 0 desactiva la fijación
<code>RADE_CORES_PER_CLIENT</code>	2	Núcleos asignados por cliente cuando la fijación está activa

Archivos modificados

Archivo	Tipo	Notas
rade_helper.py	Nuevo	Sidecar de Python: servidor WebSocket + cadena de decodificación de dos procesos
frontend/src/audio.js	Modificado	4 parches
frontend/src/App.svelte	Modificado	6 parches
rade.sh	Nuevo	Script de control del sidecar RADE para uso autónomo (start/stop/restart/status)
start-rx888mk2.sh	Modificado	arranca, vigila e informa del sidecar; igual que start-airspyhf.sh, start-rtl.sh, start-rspla.sh, start-fobos-hf.sh, start-fobos.sh y start-hackrf.sh
stop-websdr.sh	Modificado	detiene el sidecar junto con el servidor
spectrumserver.cpp	Sin cambios	No hacen falta modificaciones en C++

Resumen del flujo de señal



Resolución de problemas

Cartel rojo — «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")
```

El puerto 8074 aparece cerrado en portchecker.co pese a la regla de iptables

Puede que el ISP esté filtrando el puerto o que falte la regla NAT del router. Opciones:

1. Pruebe otro puerto: `RADE_HELPER_PORT=8080 python3 rade_helper.py`
2. Redirigir por su puerto público existente mediante Nginx (véase el [Paso 7 de la guía manual](#))

`radae_rxe.py: error: unrecognized arguments: model_path`

La ruta del modelo debe indicarse con `--model_name`, no como argumento posicional:

```
# 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` importa `matplotlib` incondicionalmente al principio del archivo.

`ModuleNotFoundError: No module named 'torch'`

```
pip3 install torch
```

`lpcnet_demo: No such file or directory`

La compilación de `radae` no llegó a completarse. Vuelva a compilar:

```
cd ~/radae/build
cmake .. && make -j$(nproc)
ls src/lpcnet_demo      # should exist now
```

Error size mismatch en inference.sh

model19_check3 requiere `--auxdata` al codificar con `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
```

Nota: al **decodificar** con `radae_rxe.py`, `--auxdata` es el valor por defecto; no lo pase.

Indicador amarillo — «Searching for signal» — nunca sincroniza

- Confirme la banda lateral correcta: RADEL para ≤ 10 MHz, RADEU para > 10 MHz
- Consulte qso.freedv.org para confirmar que hay una estación transmitiendo en ese momento
- RADE v1 usa 30 portadoras en 1500 Hz de ancho de banda: aparece como un grupo compacto en la cascada
- Deje hasta 1,5 segundos para la adquisición

underrun!!! de aplay durante la prueba de la cadena (paso 3)

Solo es esperable en las pruebas sin conexión con archivos. `radae_rxe.py` procesa más despacio que la E/S de archivos, con lo que el búfer de audio se queda sin datos. Esto no ocurre en recepción en directo porque el navegador suministra el audio a tasa de tiempo real.

ConnectionClosedError: received 1011 (internal error)

El sidecar aceptó la conexión WebSocket pero falló internamente antes de responder. La causa es una versión incompatible de `rade_helper.py`: una versión antigua intentaba pasar un `StreamReader` de asyncio como entrada estándar de un subprocesso, lo que falla en silencio y cierra con el error 1011.

Solución: sustituya `rade_helper.py` por la versión actual del conjunto de parches. La versión actual usa `os.pipe()` para la tubería entre procesos y es compatible con websockets de la 10.x a la 16.x+.

Varios usuarios simultáneos

Cada conexión de navegador genera su propio par independiente `radae_rxe.py` + `lpcnet_demo`, de modo que cada usuario puede sintonizar libremente una frecuencia distinta.

Por defecto, `radae_rxe.py` usa **todos los núcleos de CPU disponibles** para las operaciones matriciales de PyTorch, lo que provoca un uso de CPU del ~ 900 % por instancia. `rade_helper.py` lo limita con dos optimizaciones:

1. `OMP_NUM_THREADS=1` (y los equivalentes de MKL/OpenBLAS): limita PyTorch a 1 hilo
2. Funciones de conversión de audio vectorizadas con numpy: eliminan la sobrecarga de los bucles de Python

Resultado: cada instancia usa aproximadamente el **8-10 %** de un núcleo, suficiente para decodificar RADE en tiempo real. Si oye cortes de audio, suba a 2 hilos:

```
RADE_TORCH_THREADS=2 ./start-rx888mk2.sh # or your receiver's launcher
# standalone: RADE_TORCH_THREADS=2 ./rade.sh restart
```

Usuarios de RADE simultáneos	CPU aproximada
1	~9 %
5	~45 %
10	~90 %
20	~180 %

En un i5-12450H (12 hilos) con spectrumserver usando ~42 % y rx888_stream ~11 %, dispone de aproximadamente un **1000 % de margen**, suficiente para **más de 20 usuarios simultáneos** de RADE antes de que la CPU sea un problema.

La tabla anterior supone que el coste de CPU crece linealmente con los usuarios. La medición dice que no. Una ejecución de `rade_loadtest.py` en ese mismo i5-12450H mostró un coste de CPU por oyente muy por debajo de esta tabla hasta unos 24 oyentes, y luego un aumento brusco, deteniéndose la máquina por **temperatura del encapsulado con 32 oyentes**, no por CPU ni por ancho de banda. Tome estas cifras solo como guía aproximada para números pequeños y mida su propio equipo: véase [Medir la concurrencia en su propio hardware](#). Tenga en cuenta que la prueba de carga alimenta la cadena con ruido en lugar de una señal RADE real, así que el uso absoluto de CPU con tráfico auténtico puede diferir; lo fiable es la *forma* de la curva.

Medir la concurrencia en su propio hardware

Las cifras anteriores proceden de un equipo concreto. `rade_loadtest.py` (en la raíz del repositorio) mide lo mismo en **el suyo**: va añadiendo oyentes RADE sintéticos contra el sidecar por escalones e informa del último número de oyentes que su máquina sostuvo antes de que algo cediera.

Tras cada escalón deja que el equipo se estabilice y luego muestrea la temperatura del encapsulado, la CPU total, la RAM disponible, el swap y la RSS combinada de todos los procesos `radae_rxe.py` y `lpcnet_demo`. Se detiene en el primer **codo**, sea cual sea el límite que se supere primero, e imprime el último recuento estable.

Requisitos

```
sudo apt install python3-websockets python3-psutil
```

Ejécútelos **en el equipo del SDR** (lee sensores locales y la memoria de los procesos), con spectrumserver y el sidecar de RADE ya en marcha:

```
cd ~/PhantomSDR-Plus
python3 rade_loadtest.py
```

Ctrl-C cierra limpiamente todas las conexiones en cualquier momento.

Antes de la primera ejecución

Abra el archivo y revise los dos bloques del principio:

- **CONFIG:** `WS_URL` vale por defecto `ws://127.0.0.1:8074`; cámbielo si movió el sidecar de su puerto por defecto. `RADE_SPS` (12000) y `RADE_SIDEHAND` (USB) deben coincidir con lo que indica su frontend.
- **handshake_messages():** la única trama `init` que envía debe coincidir con la inicialización de RADE en `frontend/src/audio.js`. Si ese saludo se ha desincronizado, ningún oyente logra conectarse y la prueba termina de inmediato con fallos de conexión.

Umbral del codo

Ajústelos en `CONFIG` a su gusto; son deliberadamente conservadores:

Umbral	Por defecto	Se detiene cuando
<code>TEMP_KNEE_C</code>	85.0	La temperatura del encapsulado alcanza este techo
<code>MIN_AVAIL_MB</code>	800	La RAM disponible cae por debajo de este valor
<code>SWAP_GROWTH_MB</code>	50	El swap crece por encima de la línea base tomada al inicio
<code>FAIL_LIMIT</code>	3	Este número de oyentes no logra conectarse en un escalón
<code>MAX_LISTENERS</code>	60	Parada absoluta aunque nunca se alcance un codo

La forma de la rampa la controlan `STEP` (oyentes añadidos por escalón, 2 por defecto), `SETTLE_S` (25 s de estabilización antes de muestrear, que es lo que permite que se acumule el calor) y `SAMPLE_S` (ventana de promediado de CPU de 5 s). Una ejecución completa hasta el techo por defecto de 60 oyentes lleva por tanto unos 15 minutos.

Cómo leer los resultados

Cada escalón imprime una línea y añade una fila a `rade_loadtest.csv`:

Columna	Significado
<code>listeners</code>	Oyentes sintéticos conectados en este escalón
<code>connect_fails</code>	Oyentes que no lograron establecer o mantener la conexión
<code>pkg_c</code>	Temperatura del encapsulado, mediana de 5 muestras (resistente a picos)
<code>cpu_pct</code>	Porcentaje de CPU de todo el sistema , no por oyente
<code>avail_mb</code>	RAM disponible
<code>swap_mb</code>	Swap en uso
<code>dec_rss_mb</code>	RSS combinada de todos los procesos de decodificación
<code>dec_procs</code>	Número de procesos de decodificación: espere 2 por oyente (<code>radex_rxe.py</code> + <code>lpcnet_demo</code>)

Una ejecución medida

En el i5-12450H mencionado antes, una ejecución completa terminó así:

```
KNEE at n=32: temp 91.0°C ≥ 85.0
Last stable concurrency: 30 RADE listeners
```

El calor fue la restricción determinante, y con holgura. En el codo quedaban aún 7004 MB de RAM disponible —6,2 GB por encima del corte de `MIN_AVAIL_MB`— y cero fallos de conexión en todos los escalones.

Dos columnas merecen una lectura atenta:

`dec_rss_mb` exagera el coste real de memoria. Crece de forma muy constante, 295 MB por oyente (292, 294, 294 ... 295 a lo largo de los 16 escalones), pero la *RAM disponible* solo baja unos **202 MB por oyente**. Los dos procesos de cada oyente comparten páginas de bibliotecas, y la RSS cuenta esas páginas una vez por proceso. Extrapolando la pendiente de `avail_mb`, el codo de memoria llegaría cerca de los **63 oyentes**, más allá del tope absoluto de 60, así que en este equipo nunca puede dispararse. Dimensione la memoria a partir de `avail_mb`, no de `dec_rss_mb`.

cpu_pct es plano... hasta que deja de serlo. Se mantiene en torno al 5,3 % de toda la máquina hasta 24 oyentes y luego se vuelve superlineal:

Oyentes	24	26	28	30	32
cpu_pct	6,7 %	18,4 %	30,1 %	49,6 %	65,2 %

Es decir, la CPU se multiplica por 10 mientras el número de oyentes crece un tercio. La inflexión se reproduce en el mismo punto en distintas ejecuciones, así que trátela como una propiedad del equipo y no como ruido: el coste de CPU por oyente ronda el 3,4 % de un núcleo por debajo de la inflexión y en torno al 20 % por encima. No extrapole una cifra de CPU por usuario tomada con pocos oyentes.

La prueba no tiene codo de CPU. Los cuatro cortes son temperatura, RAM disponible, crecimiento del swap y fallos de conexión; la CPU se registra pero nunca termina la ejecución. En la ejecución anterior la CPU llegó al 65 % y habría seguido subiendo si no hubiera saltado antes la temperatura. Si le preocupa un techo de CPU, vigile usted mismo la columna o añada un umbral.

Qué demuestra y qué no. Cada oyente sintético transmite ruido aleatorio de baja amplitud, no una señal RADE real. Eso basta para que toda la cadena de decodificación funcione y consuma recursos, así que las cifras de recursos son significativas; pero la prueba no dice nada sobre la *calidad* de la decodificación ni sobre la continuidad del audio bajo carga. Confirme eso con oyentes reales y una señal real.

Si los núcleos se saturan antes del codo

RADE fija con fuerza un hilo por instancia, así que algunos núcleos pueden alcanzar ~90 °C mientras la CPU total parece ociosa. `rade_helper.py` reparte las cadenas entre los núcleos de forma cíclica para contrarrestarlo. Si una ejecución muestra codos térmicos tempranos, amplíe la asignación de núcleos por cliente:

```
RADE_CORES_PER_CLIENT=3 ./rade.sh restart
```

No hace falta editar ningún archivo: es una variable de entorno (véase [Variables de entorno](#)); `RADE_PIN_CORES=0` desactiva la fijación por completo.

Actualizar RADE v1

La integración con PhantomSDR-Plus (`rade_helper.py`, parches del frontend) es solo un puente: toda la lógica de decodificación de RADE vive en el repositorio `radae`. Por eso las actualizaciones son casi siempre un simple `git pull` más recompilación, sin cambios en el propio PhantomSDR-Plus.

Actualización estándar (código nuevo, mismo modelo)

```
# 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
```

Eso es todo. Sin recompilar el frontend, sin reiniciar el servidor y sin editar archivos.

Solo pesos de modelo nuevos

Si se publica un checkpoint nuevo (p. ej., `model20`) sin cambios de código, apunte el sidecar a los nuevos pesos mediante la variable de entorno; no hace falta editar ningún archivo:

```
# One-off: start with new model
RADE_MODEL=~/.radeae/model20/checkpoints/checkpoint_epoch_100.pth ./rade.sh start

# Or permanently – add to your shell profile (~/.bashrc):
export RADE_MODEL=~/.radeae/model20/checkpoints/checkpoint_epoch_100.pth
```

Qué requiere cada tipo de cambio

Qué cambió en radae	Acción necesaria
Pesos de modelo nuevos (archivo <code>.pth</code> nuevo)	Definir la variable <code>RADE_MODEL</code> , <code>./rade.sh restart</code>
Cambio de código en <code>radae_rxe.py</code>	<code>git pull</code> , <code>./rade.sh restart</code>
Cambió el código C de <code>lpcnet_demo</code>	<code>git pull</code> , recompilar, <code>./rade.sh restart</code>
<code>radae_rxe.py</code> renombrado o movido	Actualizar la ruta <code>RADAE_RX</code> en <code>rade_helper.py</code> (una línea)
Argumento <code>--model_name</code> renombrado	Actualizar <code>radae_cmd</code> en <code>rade_helper.py</code> (una línea)
Nueva tasa de muestreo de salida ($\neq 16000$ Hz)	Actualizar <code>SPS_OUT</code> en <code>rade_helper.py</code> y <code>createBuffer()</code> en <code>audio.js</code>
RADE v2 usa un binario distinto	Actualizar <code>RADAE_RX</code> en <code>rade_helper.py</code> (una línea)

Comprobar que la actualización funcionó

Tras reiniciar, confirme que se está ejecutando el código nuevo:

```
# Check sidecar picked up new radae_rxe.py
./rade.sh status

# Tail log to see startup lines
tail -20 ~/PhantomSDR-Plus/rade.log
```

El registro debería mostrar la ruta del modelo que espera:

```
[RADE] model : /home/sv1btl/radae/model19_check3/checkpoints/checkpoint_epoch_100.pth
```

Mantenerse al día

Suscríbase a la página de versiones del repositorio radae para recibir avisos de nuevos modelos y actualizaciones de código:

```
https://github.com/drowe67/radae/releases
```

Consulte el blog de FreeDV para ver anuncios sobre nuevas formas de onda y modelos de RADE:

```
https://freedv.org/blog/
```

Desarrollado y probado en PhantomSDR-Plus (fork sv1btl/PhantomSDR-Plus) con un RX-888 MK2 en Ubuntu 24.04. El servidor en C++ no se modifica.

RADE lo desarrollan David Rowe VK5DGR y el equipo de FreeDV. Véase freedv.org/radio-autoencoder.

Desinstalación completa y reinstalación limpia

Este es el desmontaje completo antes de volver a ejecutar install_rade.sh:

1. Detener el sidecar `cd ~/PhantomSDR-Plus && ./rade.sh stop`
2. Eliminar el repositorio radae y la compilación `rm -rf ~/radae`
3. Eliminar torch (instalado en el directorio local del usuario) `pip3 uninstall -y torch` `rm -rf ~/.local/lib/python3.11/site-packages/torch*`
4. Eliminar los paquetes de Python de apt (opcional: omítalo si los usan otros programas) `sudo apt-get remove -y python3-numpy python3-scipy python3-matplotlib python3-websockets` `sudo apt-get autoremove -y`
5. Comprobar que no queda nada `python3 -c "import torch" 2>&1 # should say ModuleNotFoundError` `ls ~/radae 2>&1 # should say No such file or directory`
6. Instalación limpia `chmod +x ~/PhantomSDR-Plus/install_rade.sh` `~/PhantomSDR-Plus/install_rade.sh`

CAPÍTULO 10

RADE instalación general (Linux)

RADE_General_INSTALL_MANUAL_LINUX.md

RADE v1 — Guía de instalación manual

Todo Linux — Ubuntu / Debian / Fedora / Arch / Raspberry Pi OS · PhantomSDR-Plus

Requisito previo: los archivos parcheados (`rade_helper.py`, `rade.sh`, `audio.js`, `App.svelte`) ya están colocados en el árbol de directorios de PhantomSDR-Plus. Esta guía construye todo lo demás a su alrededor.

IMPORTANTE

No necesita esta guía para una instalación normal. `./install_rade.sh`, en la carpeta de PhantomSDR-Plus, hace por usted todos los pasos siguientes — paquetes del sistema, módulos de Python, la compilación de `radare`, la comprobación de los pesos del modelo y el arranque del sidecar — y funciona igual en sistemas `apt`, `pacman`, `dnf` y `zypper`. `./install.sh` y los cuatro instaladores por distribución ofrecen ejecutarlo como parte de la instalación normal. Siga esta guía solo para una instalación manual, para un sistema que el script no cubre, o para reparar un paso a mano.

Usuarios de Raspberry Pi: esta también es su guía. Raspberry Pi OS es Debian, así que siga los bloques de Debian en todo el documento y lea las notas **Raspberry Pi / Bookworm** allí donde aparezcan: cubren las dos cosas que cambian en una Pi, la PEP 668 y la rueda de `torch` solo para CPU.

Paso 1 — Paquetes del sistema

Elija el bloque que corresponda a su distribución.

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
```

Comprobación de la versión de Node.js (todas las distribuciones)

La compilación del frontend de RADE requiere Node.js 22 o posterior:

```
node --version
```

Si es inferior a 22.x, instálelo con `nvm` (cualquier distribución, sin 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
```

No use NodeSource. `deb.nodesource.com` y `rpm.nodesource.com` responden ahora con HTTP 403 en todas las rutas del repositorio, por lo que la antigua línea `curl -fsSL https://deb.nodesource.com/setup_NN.x | sudo -E bash -` ya no funciona; además, en Debian/Ubuntu deja una fuente apt que rompe todos los `apt update` posteriores. Por eso los scripts de instalación usan `nvm`.

Paso 2 — Paquetes de Python

Ubuntu 24.04 y anteriores / Fedora / Arch

En estos sistemas funciona un `pip3 install` normal:

```
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
```

Si su servidor **sí** tiene una GPU NVIDIA con CUDA instalado, puede omitir `--index-url` para obtener la compilación completa con CUDA. Para RADE no supone ninguna ventaja de rendimiento: `radae_rxe.py` usa PyTorch solo para operaciones matriciales en CPU.

Ubuntu 23.04+ / Debian Bookworm+ / Raspberry Pi OS (sistemas con PEP 668)

Estas distribuciones bloquean el `pip3 install` sin más. Añada la opción:

```
# 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: dos reglas que muerden:

1. Cada `pip3 install` requiere `--break-system-packages` (aplicación de la PEP 668). Sin él, la instalación se bloquea por completo.
2. Un simple `pip3 install torch` descarga la rueda de CUDA (~3 GB). En una Pi no hay CUDA: use el índice solo para CPU y obtendrá la rueda ARM64 ligera (~150 MB).

Como alternativa, use un entorno virtual y evite por completo esa opción:

```
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
```

Si usa un venv, anteponga `source ~/rade-venv/bin/activate` a todas las llamadas `python3` posteriores de esta guía, o use la ruta completa `~/rade-venv/bin/python3`.

Verificación

```
python3 -c "import websockets, matplotlib, torch, numpy, scipy; print('All OK')"
```

Salida esperada:

```
All OK
```

Paso 3 — Clonar y compilar el repositorio radae

El decodificador RADE vive en un repositorio distinto al de codec2. `freedv_rx` de codec2 **no** admite RADE v1.

```
# Remove any previous incomplete clone
rm -rf ~/radae

# Clone
git clone https://github.com/drowe67/radae.git ~/radae
cd ~/radae

# Build
mkdir build && cd build
cmake ..
make -j$(nproc)
```

Compruebe que se compiló lpcnet_demo

```
ls -la ~/radae/build/src/lpcnet_demo
```

Esperado: el binario está presente y es ejecutable.

Compruebe que están los pesos del modelo

```
ls ~/radae/model19_check3/checkpoints/checkpoint_epoch_100.pth
```

Esperado: aparece el archivo `.pth`. Los pesos vienen incluidos en el repositorio; no hace falta descargarlos aparte.

Paso 4 — Verificar la cadena de decodificación

Este paso confirma que toda la cadena sin conexión funciona antes de conectarla al navegador. Ejecútelo desde `~/radeae`:

```
cd ~/radeae
```

4a — Generar una señal de prueba codificada en 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
```

Espere a que termine. Las últimas líneas impresas deberían ser:

```
loss: 0.741 Auxdata BER: 0.012
```

4b — Decodificar y reproducir

```
cat rx.f32 \
| python3 radeae_rxe.py \
--model_name model19_check3/checkpoints/checkpoint_epoch_100.pth \
| ./build/src/lpcnet_demo -fargan-synthesis - - \
| aplay -f S16_LE -r 16000
```

Debería oír una voz. La salida muestra la adquisición del sincronismo:

```
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
```

Los mensajes `underrun!!!` durante esta prueba son esperables e inocuos. Se deben a que `radae_rxe.py` procesa más despacio que la E/S de archivos. No aparecen en recepción en directo: el navegador suministra el audio a tasa de tiempo real.

4c — Comprobar que el sidecar arranca correctamente

```
python3 ~/PhantomSDR-Plus/rade_helper.py
```

Salida esperada (sin líneas WARNING):

```
[RADE] helper starting on ws://0.0.0.0:8074
[RADE] radeae_rx.py      : /home/<user>/radeae/radeae_rxe.py
[RADE] model            : /home/<user>/radeae/model19_check3/checkpoints/checkpoint_epoch_100.pth
[RADE] lpcnet_demo      : /home/<user>/radeae/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
```

Pulse **Ctrl+C** para detenerlo.

`<user>` en las rutas anteriores es la cuenta con la que ha iniciado sesión; en una imagen estándar de Raspberry Pi OS quedan como `/home/pi/radae/...`.

Paso 5 — Compilar el frontend de PhantomSDR-Plus

```
cd ~/PhantomSDR-Plus/frontend

# Only if node_modules is missing:
npm install

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

Esté atento a errores del analizador Vite/acorn. Los archivos parcheados evitan deliberadamente `?.`, `??` y los `catch {}` vacíos, para cumplir la restricción de acorn.

Paso 6 — Control del sidecar

Normalmente no necesita un script de control aparte para RADE. Los scripts de arranque (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) arrancan el sidecar por sí mismos en cuanto el servidor está en marcha, lo reinician si termina e informan de su estado; `stop-websdr.sh` lo detiene junto con el servidor. Pase directamente al Paso 7.

`rade.sh` sigue incluyéndose para uso autónomo: ejecutar el sidecar sin el lanzador de PhantomSDR-Plus, o probarlo por separado. Sus órdenes, su registro y las advertencias sobre el vigilante están en Control del sidecar.

Paso 7 — Abrir el puerto 8074

El sidecar escucha en el puerto TCP **8074**. El navegador se conecta directamente a ese puerto. Debe abrirlo manualmente.

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 (cualquier distribución, permanente)

```
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
```

Router

Añada una regla de NAT/redirección de puertos: **TCP 8074 → IP LAN del servidor : 8074**

Prueba desde el exterior

Use **https://portchecker.co** y compruebe el puerto 8074 contra su nombre de host público. **No** lo pruebe con `curl` desde el propio servidor: el NAT hairpin da falsos «Connection refused» aunque el puerto esté abierto.

Alternativa — proxy Nginx (si el ISP bloquea el puerto 8074)

Añada dentro de su bloque `server {}` existente:

```
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
```

Después edite `audio.js` dentro de `setRADEDecoding()`:

```
// Change:
var helperUri = uri || ('ws://' + window.location.hostname + ':8074');
// To:
var helperUri = uri || ('ws://' + window.location.host + '/rade');
```

Vuelva a compilar el frontend tras este cambio (`./recompile.sh`).

Paso 8 — Arrancar el servidor

Inicie PhantomSDR-Plus como de costumbre. RADE **no** se inicia automáticamente:

```
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
```

Use el lanzador que corresponda a su receptor. Cada uno incluye su propio watchdog y registro; `./stop-websdr.sh` detiene el que esté en marcha.

Paso 9 — Arrancar RADE

```
cd ~/PhantomSDR-Plus
./rade.sh start
./rade.sh status
tail -f rade.log
```

Registro esperado:

```
[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
```

Paso 10 — Usar RADE en el navegador

1. Abra su interfaz web de PhantomSDR-Plus
2. Busque estaciones activas en <https://qso.freedv.org>
3. Sintonice la frecuencia de dial de la estación
4. En **Decoder Options**, seleccione:
5. **RADE v1 — RADEL (LSB)** para 40 m / 80 m / 160 m (≤ 10 MHz)
6. **RADE v1 — RADEU (USB)** para 20 m / 17 m / 15 m / 12 m / 10 m (> 10 MHz)
7. Haga clic en **Decoder: ON**

Los pasos 4 y 5 pueden sustituirse por una sola pulsación: los botones **RADEL / RADEU** están junto al encabezado **Modes selector** y dentro de las ventanas emergentes **Modes** y **Bands**. Una pulsación selecciona el decodificador, lo pone en ON y desplaza el panel de RADE hasta hacerlo visible; púlselo de nuevo para detener RADE. Consulte el [manual de RADE](#).

Indicador	Significado
Rojo — «Connecting to sidecar...»	Puerto 8074 inalcanzable o sidecar detenido
Amarillo — «Searching for signal...»	Sidecar conectado, aún sin trama RADE detectada (deje ~1,5 s)
Verde — «Synced · SNR x.x dB»	Decodificando: se está reproduciendo la voz

Nota sobre la CPU

Cada usuario de RADE consume alrededor del 8-10 % de un núcleo (el sidecar limita el número de hilos de PyTorch a 1). Si oye cortes de audio, suba a 2 hilos:

```
RADE_TORCH_THREADS=2 ./rade.sh restart
```

Usuarios simultáneos	CPU aproximada
1	~9 %
5	~45 %
10	~90 %
20	~180 %

Estas cifras se midieron en x86_64. Una Raspberry Pi decodifica RADE sin problemas, pero el coste por usuario es mayor y la tabla anterior no es trasladable: mida su propia placa con `top` mientras un usuario está sincronizado antes de anunciar un límite de usuarios.

Actualizar RADE en el futuro

```
cd ~/radae
git pull
cmake -S . -B build && make -C build -j$(nproc)

cd ~/PhantomSDR-Plus
./rade.sh restart
```

No hace falta recompilar el frontend ni reiniciar el servidor salvo que haya cambiado el propio `rade_helper.py`.

Probado en Ubuntu 24.04 (x86_64) y Raspberry Pi 4 / Raspberry Pi OS Bookworm (ARM64, Python 3.11). Fork de PhantomSDR-Plus: [sv1bt/PhantomSDR-Plus](https://github.com/sv1bt/PhantomSDR-Plus). RADE desarrollado por David Rowe VK5DGR y el equipo de FreeDV — <https://freedv.org>

CAPÍTULO 11

Edición de variantes

EDITING_VARIANTS.md

PhantomSDR-Plus — Cómo editar las variantes del frontend

(tras fusionar los cuatro archivos `App__*_smeter_.svelte` en un único `App.svelte`)

1. Qué son las variantes

Dos elecciones independientes, que dan cuatro combinaciones:

```
smeter = "analog"    el instrumento de aguja móvil
      = "digital"    el instrumento de barra segmentada

layout = "v1"        selector de modos arriba, selector de banda abajo
      = "v2"        selector de banda arriba, selector de modos abajo
                    (mueve también la fila de botones de sintonía fina en el móvil)
```

Las cuatro viven en **una sola** construcción, servida en `/`. El selector ☞ de la esquina superior derecha cambia entre ellas dentro de la página en marcha: asigna las props `smeter/layout` de `App.svelte`, así que no se recarga nada y el audio, la cascada y los decodificadores siguen funcionando. La elección se guarda por navegador en `localStorage` bajo `phantom.variant`, y en la siguiente visita prevalece sobre el valor predeterminado de la construcción.

Por tanto, la variante que el sysop elige en `recompile.sh` es la variante **inicial**: lo que ve quien visita el sitio por primera vez, antes de tocar el menú.

URL	qué es
<code>/</code>	la página de escritorio; las cuatro variantes, conmutables en tiempo de ejecución
<code>/mobile/</code>	<i>página aparte, construida por <code>build-mobile.sh</code> — no es una variante</i>

Hasta el 10-08-2026 cada variante era su propia construcción bajo `/analog/`, `/digital/`, `/v2-analog/` y `/v2-digital/`, y el selector navegaba entre ellas: por eso cambiar de variante recargaba la página y cortaba el audio. Esas construcciones ya no existen; en cada una de las cuatro rutas queda solo un pequeño `index.html` que redirige a `/`, escrito por `frontend/make-redirect-stubs.sh` en cada construcción para que los marcadores antiguos sigan funcionando.

La variante **no** se escribe en ningún archivo fuente. `src/main.js` es un punto de entrada fijo que pasa al componente dos valores definidos en tiempo de construcción:

```
const app = new App({
  target: document.getElementById('app'),
  props: { smeter: __PHANTOM_SMETER__, layout: __PHANTOM_LAYOUT__ }
})
```

`vite.config.js` los rellena al construir, tomando el primero de estos que esté definido:

1. **PHANTOM_SMETER / PHANTOM_LAYOUT**: variables de entorno. Ya no las establece nada, ahora que no hay construcciones por variante; siguen funcionando para una construcción puntual con otra variante inicial.
2. **frontend/variant.json**: el valor predeterminado del sitio, escrito por `recompile.sh`. La construcción de la raíz (/) no pasa ninguna variante, así que acaba aquí.
3. **analog / v1**: el valor de reserva.

Esos dos valores son solo la semilla. `App.svelte` los sobrescribe al inicializarse a partir de `localStorage` si el visitante ya eligió una variante, y el selector `☞` vuelve a asignarlos en cada cambio.

Dentro de `App.svelte` las props se convierten en dos banderas en las líneas 20-21:

```
$: isAnalog = smeter !== "digital";
$: isV2      = layout === "v2";
```

Use esas dos banderas — nunca `smeter/layout` directamente — para que todo el archivo se lea de la misma manera.

2. Qué hace cada archivo

`frontend/src/main.js` — 12 líneas

El punto de entrada. Nada más que la instanciación de `App` y los dos valores `__PHANTOM__*` que pasa como props. Es **constante**: ni `recompile.sh` ni los scripts de construcción lo reescriben ya, y eso es lo que permite construir las variantes al mismo tiempo.

No ponga código de la aplicación aquí. Es el archivo de entrada, no un sitio para lógica.

`frontend/src/App.svelte` — ~17130 líneas

Toda la aplicación: cascada, sintonía, decodificadores, marcadores, paneles, disposición de escritorio Y la disposición adaptable para teléfono. Casi todas las modificaciones que haga van aquí.

`frontend/src/lib/SMeterAnalog.svelte` — 824 líneas

El instrumento de aguja móvil, completo: dibujo de la esfera, aguja, el conmutador de esfera clara/ oscura con su clave de `localStorage`, y el suavizado de la aguja.

- Prop de entrada: `dbm` — la potencia CALIBRADA en dBm.
- También acepta: `mobile` — atributos de canvas más pequeños para la disposición de teléfono.
- Ajuste: `SMOOTH_TIME_MS` en la línea 55. Menor = aguja más rápida, mayor = más suave. Actualmente 16 (un fotograma a 60 fps, en la práctica el mínimo).

Las tres esferas — y `smeter_theme.sh`

El instrumento analógico tiene tres fondos, todos dibujados por el mismo código:

```
dark      el metal cepillado oscuro original
amber     una esfera clara, gris pálido
vintage   una esfera cálida de ámbar envejecido
```

No lo confunda con la variante `smeter = analog / digital` de arriba. La variante elige *qué componente de instrumento* se muestra; la esfera elige *cómo se pinta* el analógico. Se guardan bajo claves distintas de `localStorage` (`phantom.variant` y `smeterTheme`) y se fijan en lugares distintos.

El lado del visitante. El lienzo del instrumento es un botón (`role="button"`, también Enter/Espacio desde el teclado, con la indicación «click to switch style» debajo). Cada clic recorre `dark` → `amber` → `vintage` y guarda la elección en `localStorage.smeterTheme`. Se vuelve a leer en cada carga de página, así que la elección del visitante sobrevive a recargas y reinicios del navegador de forma indefinida. Deliberadamente **no** es una cookie: una cookie se enviaría al servidor en cada petición de cascada y de audio, para un valor que el servidor nunca lee.

El lado del sysop. Dos cosas en `SMeterAnalog.svelte` controlan lo que reciben todos los demás:

Línea	Qué hace
<code>let smeterTheme = 'dark';</code>	La esfera que recibe un navegador sin elección guardada .
<code>const SMETER_PREF_VERSION = 2;</code>	Un contador. Cuando la versión guardada de un navegador está por detrás, olvida su esfera guardada una vez y adopta el valor por defecto anterior. Incrementéelo para mover a los visitantes que ya han elegido.

Cambiar solo el valor por defecto llega, por tanto, únicamente a los visitantes totalmente nuevos. Ambos cambios juntos mueven a todos: para eso está `smeter_theme.sh` en la raíz del repositorio:

```
./smeter_theme.sh          # informar del valor actual y elegir en un menú
./smeter_theme.sh vintage  # fijarlo directamente y ofrecer la reconstrucción
./smeter_theme.sh dark --build # ejecutar frontend/build-all.sh sin preguntar
./smeter_theme.sh amber --no-reset # solo visitantes nuevos – no tocar a los existentes
```

Informa del valor actual, fija el nuevo, incrementa el contador de reinicio (siempre hacia arriba — **nunca lo baje**, un número menor simplemente impide que el reinicio se dispare) y después ofrece la misma elección de construcción del frontend que `recompile.sh`. **No** toca el backend y **no** reescribe `variant.json`.

Un cambio llega a un oyente conectado solo cuando recarga su página: el valor por defecto está compilado en el paquete JS y el reinicio se ejecuta al cargar. No hay envío en vivo. Basta con una recarga normal, ya que vite aplica hash a los nombres de archivo del paquete.

El borrado único del contador se impone a todos, incluidos los visitantes que eligieron deliberadamente su propia esfera. Es intencionado: es la única forma de devolver todo el sitio a un mismo aspecto. Use `--no-reset` si prefiere dejarlos en paz.

frontend/src/lib/SMeterDigital.svelte — 184 líneas

El instrumento de barra segmentada, completo.

- Props de entrada: `rawDb` — el valor EN BRUTO de `audio.getPowerDb()`
- `smeterOffset` — `audio.smeter_offset`
- `mobile` — atributos de canvas más pequeños

NOTA: este instrumento se alimenta deliberadamente de la potencia EN BRUTO más el offset, NO del dBm calibrado que usa la esfera analógica. Siempre ha hecho su propia correspondencia. Darle el valor calibrado desplazaría todas las lecturas en cada estación digital. No «unifique» estas dos entradas.

frontend/src/lib/BandSelector.svelte — 69 líneas

La rejilla de botones de banda. Se renderiza dos veces en `App.svelte` (una por disposición).

frontend/src/lib/ModesSelector.svelte — 75 líneas

La rejilla de botones de modo. Se renderiza dos veces en `App.svelte`.

frontend/src/lib/StatusIndicators.svelte — 122 líneas

Los pilotos de estado (mute, squelch, NR, NB, NS, AN, CTCSS). Acepta una prop `wide`: la disposición digital usa pilotos más anchos que la analógica. Ambas clases de Tailwind (`w-8` y `w-10`) están escritas por completo dentro del componente — el escáner de Tailwind solo ve nombres de clase literales, así que nunca los construya por concatenación de cadenas.

Todos los demás archivos de `frontend/src/lib/` (Spectrogram, PassbandTuner, FrequencyInput, VersionSelector, ...) son anteriores a este trabajo y no han cambiado.

3. Hacer un cambio que afecte solo a algunas variantes

Envuélvalo en la bandera. En el marcado:

```
{#if isAnalog}
  ...solo las estaciones analógicas ven esto...
{:else}
  ...solo las estaciones digitales ven esto...
{/if}

{#if isV2} ... {/if}           {#if !isV2} ... {/if}
```

En un atributo `class`:

```
class="p-4 {isAnalog ? 'text-xs' : 'text-sm'} rounded-md"
```

Escriba ambas alternativas como literales completos, como arriba. Tailwind rastrea el código fuente buscando nombres de clase completos; `text-{size}` o un nombre concatenado se purga del CSS y el estilo desaparece en silencio.

En el bloque `<script>`: intente NO ramificar en absoluto. El script calcula deliberadamente las entradas de AMBOS instrumentos en cada ciclo —

```
smeterDbm      = powerDb;           // esfera analógica
smeterRawDb    = audio.getPowerDb(); // esfera digital
smeterOffset   = audio.smeter_offset; // esfera digital
```

— de modo que solo el marcado tiene que elegir. Por eso hay apenas 15 conmutadores de variante en un archivo de 17000 líneas. Manténgalo así; la ramificación a nivel de script fue lo que hizo que los cuatro archivos antiguos divergieran.

4. Dónde están los conmutadores de variante

Quince lugares en `App.svelte`. Los números de línea se desplazan al editar — la forma fiable de encontrarlos todos es:

```
cd frontend/src && grep -n 'isAnalog\|isV2' App.svelte
```

En el momento de escribir esto:

Línea	Bandera	Qué conmuta
20		isAnalog declarada
21		isV2 declarada
8062	isV2	escritorio: selector de banda primero (orden v2)
8090	isV2	escritorio: tamaño del encabezado del panel
8686	isV2	escritorio: selector de modos primero (orden v1)
8717	isAnalog	ancho mínimo del panel del instrumento (la aguja es más estrecha)
8730	isAnalog	solo analógico: línea de fecha/hora sobre el instrumento
8819	isAnalog	solo analógico: pilotos de estado + separador
8834	isAnalog	solo digital: línea de hora, pilotos anchos y la propia elección <code><SMeterDigital></code> / <code><SMeterAnalog></code>
8861	isAnalog	espaciado vertical del bloque de frecuencia
8864	isAnalog	margen superior del bloque de sintonía fina
12288	isV2	teléfono: fila de sintonía fina arriba (orden v2)
12347	isAnalog	teléfono: qué instrumento se renderiza
12433	isV2	teléfono: fila de sintonía fina abajo (orden v1)
13204	isAnalog	tamaño de texto del botón de noise gate

Todo lo demás en el archivo es común a las cuatro variantes.

5. Otros ajustes que conviene conocer

Ubicación	Constante	Propósito
App.svelte línea 5034	<code>const visualGain = 1.1;</code>	Calibración del instrumento analógico. Cada 0,10 equivale aproximadamente a 5 dBm en la lectura.
SMeterAnalog.svelte línea 55	<code>const SMOOTH_TIME_MS = 16;</code>	Constante de tiempo del suavizado de la aguja, en milisegundos.
SMeterAnalog.svelte línea 64	<code>let smeterTheme = 'dark';</code>	Esfera por defecto del instrumento analógico para un navegador sin elección guardada. Fíjela con <code>./smeter_theme.sh</code> , no a mano.
SMeterAnalog.svelte línea 69	<code>const SMETER_PREF_VERSION = 2;</code>	Contador de reinicio único de la esfera guardada. Incrementelo para llevar a los visitantes existentes al valor por defecto; solo sube.
SMeterDigital.svelte línea 149	<code>const DIGITAL_BAR_TRIM = 0;</code>	Desplaza la barra en segmentos enteros.
SMeterDigital.svelte línea 31	<code>const numberOfDots = 35;</code>	Número de segmentos de la barra.

6. Después de editar — reconstruir

Lo más sencillo, desde la raíz del repositorio:

```
./recompile.sh
```

La opción de menú **[2]** reconstruye el frontend y pregunta qué variante debe ver quien visita el sitio por primera vez. Esa elección se registra en `frontend/variant.json`: no se copia ni se reescribe ningún archivo fuente. (Antes también regeneraba `VersionSelector.svelte` a partir de un heredoc; eso se eliminó cuando el selector dejó de navegar, porque habría revertido la conmutación en tiempo de ejecución en cada ejecución.) La opción **[1]** es solo el backend, la **[3]** ambos.

O directamente, desde `frontend/`:

```
./build-all.sh      construye la página de escritorio Y /mobile/ &lt;-- normalmente use este
./build-default.sh   solo la página de escritorio
./build-mobile.sh    solo /mobile/
```

`build-default.sh` deja que vite vacíe `dist/`, lo que elimina `dist/mobile/` hasta el siguiente `build-all.sh` o `build-mobile.sh`. Por eso `build-all.sh` es la elección normal.

`build-all.sh` tiene su propia copia de la lógica de construcción (`build_version SMETER LAYOUT NAME OUTDIR BASE LOG`) en lugar de llamar a `build-default.sh`. Si cambia cómo se construye la página, cámbielo en AMBOS sitios o las dos rutas discreparán.

Algunos detalles que conviene conocer:

- Establece `PHANTOM_KEEP_OUTDIR=1` para que Vite no vacíe el directorio de salida, y limpia `dist/` él mismo: borra el contenido pero conserva `users.json`, que el `spectrumserver` en marcha reescribe cada vez que un oyente se conecta o se desconecta.
- Construye `/mobile` en último lugar, cuando la salida de escritorio ya está completa.
- Su maquinaria de construcción en paralelo (`PHANTOM_BUILD_JOBS`, 3 por defecto) es un resto de cuando había cinco construcciones de escritorio. Con una sola, nada hace cola.

Bits de ejecución: `recompile.sh` ejecuta `chmod +x` sobre un script antes de lanzarlo, y `build-all.sh` invoca deliberadamente `bash build-mobile.sh` en lugar de `./build-mobile.sh`, así que un bit ausente ya no rompe la construcción. Aun así, subir un script por la interfaz web de GitHub lo devuelve a 644, de modo que si ejecuta alguno directamente:

```
chmod +x frontend/build-*.sh
```

7. Cómo comprobar su cambio antes de fiarse de él

Un `vite build` en verde **NO** basta. La construcción no puede ver un identificador que se perdió al mover código entre archivos: eso es un `ReferenceError` en tiempo de ejecución, y la página sencillamente no carga. Ha ocurrido dos veces.

1) Comprobación de identificadores no definidos (detecta exactamente ese fallo)

```
cd frontend
npx eslint --no-eslintrc \
  --parser svelte-eslint-parser \
  --rule '{"no-undef":"error"}' \
  --env browser,es2022 \
  src/App.svelte src/lib/SMeter*.svelte
```

Se espera un hallazgo conocido e inofensivo:

```
'dBmCalOffset' is not defined
```

`typeof x === 'number'` sobre un nombre no declarado nunca lanza error, así que esa expresión ya vale 0. Cualquier otra cosa es un problema real.

2) Construya y luego CARGUE LA PÁGINA Y RECORRA LAS CUATRO VARIANTES

No solo la que cambió: una edición bajo `isAnalog/isV2` puede romper las otras tres mientras la suya funciona. Abra `/`, recorra las cuatro entradas del menú ☙ y luego abra `/mobile/`.

Abra la consola del navegador en cada una. Una página en blanco o dibujada a medias con un `ReferenceError` en la consola es la firma del error anterior.

3) Nunca deje `frontend/dist/` a medias

Un `npx vite build` a secas vacía el directorio de salida y se lleva con él `dist/mobile/` y el `dist/users.json` vivo del servidor: la página móvil responde 404 hasta la siguiente construcción correcta. Construya siempre mediante los scripts: `build-all.sh` establece `PHANTOM_KEEP_OUTDIR=1` y hace la limpieza él mismo, conservando `dist/users.json`. En caso de duda, vuelva a ejecutar `./build-all.sh`.

CAPÍTULO 12

Guía del usuario

USER_GUIDE.md

Guía del usuario de PhantomSDR-Plus

¡Bienvenido a PhantomSDR-Plus! Esta guía le ayudará a sacar el máximo partido a su experiencia de escucha con WebSDR.

Índice

1. Introducción
 2. Primeros pasos
 3. Descripción de la interfaz
 4. Operaciones básicas
 5. Funciones avanzadas
 6. Modos de demodulación
 7. Decodificadores digitales
 8. Atajos de teclado
 9. Marcadores
 10. Uso en móviles
 11. Consejos y buenas prácticas
 12. Resolución de problemas
 13. Preguntas frecuentes
-

Introducción

¿Qué es PhantomSDR-Plus?

PhantomSDR-Plus es una radio definida por software (SDR) basada en web que le permite escuchar señales de radio a través de Internet. No necesita software ni hardware especial por su parte: ¡basta con un navegador web moderno!

¿Qué puede escuchar?

Según la configuración del WebSDR, podrá sintonizar:

- **Radioafición:** radioaficionados de todo el mundo
- **Emisoras de radiodifusión:** radio AM/FM, emisiones en onda corta
- **Aviación:** control del tráfico aéreo, comunicaciones aeronáuticas
- **Marítimo:** comunicaciones barco-costa, meteorología marítima
- **Satélites meteorológicos:** NOAA, METEOR-M
- **Modos digitales:** FT8, RTTY, PSK31 y más
- **Estaciones de servicio:** señales horarias, militares, gubernamentales

Requisitos del sistema

- **Navegador:** Chrome/Edge (recomendado), Firefox, Safari
 - **Conexión:** Internet de banda ancha (se recomienda 1 Mbps o más)
 - **Audio:** altavoces o auriculares en funcionamiento
 - **Opcional:** ratón con rueda para sintonizar con mayor comodidad
-

Primeros pasos

1. Acceda al WebSDR

Abra su navegador y vaya a la dirección del WebSDR que le haya facilitado el operador.

Ejemplo: `http://websdr.example.com:9002`

2. Carga inicial de la página

Al cargarse la página verá:

- una cascada de colores que muestra la actividad radioeléctrica
- un panel de control con el indicador de frecuencia y los botones
- un S-metro que indica la intensidad de la señal
- un indicador del número de usuarios

3. Empiece a escuchar

1. **Haga clic sobre una señal** en la cascada
 2. **El audio comenzará automáticamente**
 3. **Ajuste el volumen** con el control del navegador o el deslizador en pantalla
 4. **Afine la frecuencia** haciendo clic con precisión sobre la señal
-

Descripción de la interfaz

Componentes principales

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]	

Los tres paneles bajo la cascada se desplazan como una sola página; en una ventana estrecha se apilan en lugar de ponerse uno al lado del otro. **RADEL** y **RADEU** están en la propia línea del encabezado *Modes selector*, a su derecha.

1. Cascada (waterfall)

La cascada es una representación visual de las señales de radio:

- **Eje horizontal:** la frecuencia
- **Eje vertical:** el tiempo (se desplaza hacia abajo)
- **Colores:** la intensidad de la señal
- **Azul oscuro/negro:** sin señal (nivel de ruido)
- **Verde/amarillo:** señales débiles o moderadas
- **Naranja/rojo:** señales fuertes
- **Blanco:** señales muy fuertes

2. Indicador de frecuencia

Muestra la frecuencia sintonizada en distintos formatos:

- **MHz:** 7.100.000 MHz (bandas de HF)
- **kHz:** 14200 kHz
- **Hz:** 145500000 Hz (VHF/UHF)

3. S-metro (analógico o digital)

Muestra la intensidad de la señal:

- **S1-S9:** escala estándar de intensidad de señal
- **+dB:** señales por encima de S9 (p. ej., S9+20dB)
- **Analógico o digital:** según la configuración

4. Botones de modo

Seleccione el modo de demodulación:

- **AM**: modulación de amplitud
- **FM**: modulación de frecuencia
- **USB**: banda lateral superior
- **LSB**: banda lateral inferior
- **CW**: código Morse (onda continua)
- **WBFM**: FM de banda ancha (radiodifusión)
- **QUAM**: AM estéreo C-QUAM — la etiqueta se pone verde cuando se detecta un piloto estéreo

Junto al encabezado **Modes selector** están los botones **RADEL** y **RADEU**, que inician la voz digital RADE v1 con una sola pulsación — consulte [Decodificadores digitales](#).

5. Panel de control

Controles adicionales:

- **AGC**: control automático de ganancia
- **NR**: reducción de ruido (espectral)
- **NB**: supresor de impulsos
- **NS**: supresión del ruido de fondo
- **AN**: notch automático
- **CTCSS**: silenciador por subtono (FM). Al pulsarlo se abre junto al botón una pequeña ventana con *Any tone* y los 49 subtonos estándar (67,0–254,1 Hz); al elegir uno, el botón lo muestra (por ejemplo 88.5, o ANY) y el audio solo pasa mientras se recibe ese subtono — tarda unos 0,7 s en abrir. Pulsar de nuevo el botón desactiva el CTCSS; Escape o un clic fuera cierra la ventana sin cambiar nada. Los marcadores guardan el tono elegido.
- **SQL**: silenciador (squelch)
- **AI**: reducción de ruido por IA (solo voz)
- **Zoom**: nivel de ampliación de la cascada
- **Wheel Tuning Steps**: paso de sintonía de la rueda del ratón
- **Decoders**: botones que inician y detienen cada decodificador con una pulsación

6. Superposición del plan de bandas

Barras de color que muestran las atribuciones de frecuencia:

- colores distintos para servicios distintos
- ayuda a identificar qué está permitido en cada frecuencia

Operaciones básicas

Sintonizar una frecuencia

Método 1: hacer clic en la cascada

1. Haga clic directamente sobre una señal de la cascada
2. El receptor sintonizará esa frecuencia
3. Comenzará a reproducirse el audio

Método 2: escribir la frecuencia

1. Haga clic en el indicador de frecuencia
2. Escriba la frecuencia deseada
3. Pulse Intro

Ejemplos:

- 7100 → 7.100 MHz
- 14200.0 → 14.200 MHz
- 145.500 → 145.500 MHz

Método 3: usar la rueda del ratón

1. Sitúe el puntero sobre el indicador de frecuencia
2. Rueda hacia arriba para aumentar la frecuencia
3. Rueda hacia abajo para reducirla

Método 4: botones de paso

1. Utilice los botones ▲ y ▼ junto a la frecuencia
2. El paso varía según el modo:
3. **AM/FM**: pasos de 1 kHz
4. **SSB**: pasos de 100 Hz
5. **CW**: pasos de 10 Hz

Escáner

El escáner recorre el receptor por un margen de canales y se detiene en el primero que lleva señal. Está en la fila **Fine Tuning (kHz)**, a la derecha del todo:

7 152.0 ◀ ■ ▶ ⦿ 30 dB · 4 ▼

Control	Qué hace
◀ ▶	Explorar hacia abajo / arriba. Detenido sobre una señal, una flecha reanuda la exploración.
■	Detener.
⦿	Excluir este canal de la exploración — para una portadora fija o un espurio.
30 dB · 4 ▼	El umbral, y a su lado el nivel que se mide en este momento. Abre los ajustes.

El texto de la izquierda indica qué hace el escáner: **Scanner** en reposo, la frecuencia alcanzada mientras explora, **3s** mientras cuenta para reanudar, **30·22s** mientras el canal sigue ocupado, **hold** cuando se quedará ahí.

Ajustes

- **Range** — *Scan Band* recorre la banda donde empieza la exploración; *Scan Visible* recorre exactamente lo que muestra la cascada y la sigue si hace zoom o la arrastra.
- **Scan** — *Every channel* sintoniza y escucha cada canal por turno; *Skip empty* lee el espectro y salta directamente a las señales.
- **Stop at** — cuánto por encima del ruido de fondo de la banda debe estar un canal para detener la exploración, en dB. El ruido se sigue de forma continua, así que el mismo ajuste vale de día y de noche. Un canal vacío no marca 0 dB — observe un momento la cifra en el botón y ponga el umbral por encima.

- **Resume after** — cuánto tiempo debe permanecer en silencio un canal antes de seguir. Las pausas del habla no lo reinician. *Hold* se queda hasta que pulse un botón.
- **Max stay** — sigue adelante pasado este tiempo aunque la señal continúe, para que una portadora permanente no retenga la exploración indefinidamente.

El paso sigue al modo — 1 kHz en SSB, 0,1 kHz en CW, 5 kHz en AM, 9 o 10 kHz en onda media y 9 kHz en onda larga — y las paradas caen en la retícula de canales. La exploración se mantiene dentro de su margen y de lo que el receptor puede sintonizar; en un extremo continúa por el otro.

Los ajustes y los canales excluidos se recuerdan en su navegador.

Seleccionar el modo de demodulación

Elija el modo adecuado para la señal:

Para comunicaciones de voz:

- **AM**: aviación, radiodifusión en AM, algo de radioafición
- **FM**: repetidores de VHF/UHF, radiodifusión en FM
- **USB**: radioafición en HF (20 m, 17 m, 15 m, 12 m, 10 m)
- **LSB**: radioafición en HF (160 m, 80 m, 40 m, 30 m)

Para datos/digital:

- **USB**: la mayoría de los modos digitales (FT8, PSK31, RTTY)
- **LSB**: algunos modos digitales en las bandas bajas de HF

Para código Morse:

- **CW**: señales de telegrafía/Morse

Ajustar el volumen

- **Deslizador en pantalla**: arrastre el control de volumen
- **Volumen del navegador**: use los controles multimedia del navegador
- **Volumen del sistema**: ajuste el volumen del ordenador
- **Teclado**: use las teclas + y - (si están admitidas)

Uso del S-metro

El S-metro indica la intensidad de la señal:

- **S0-S3**: señal muy débil, difícil de copiar
- **S4-S6**: señal débil o aceptable
- **S7-S9**: señal buena o fuerte
- **S9+**: señal extremadamente fuerte

Consejo: para obtener el mejor audio, sintonice señales que marquen S7 o más.

Cambiar la esfera del instrumento: en el S-meter analógico (de aguja), haga clic sobre el propio instrumento —o enfóquelo y pulse Enter o Espacio— para recorrer tres fondos: metal cepillado oscuro, una esfera clara gris pálido y una cálida esfera ámbar de estilo vintage. Su navegador recuerda la elección, así que sigue ahí tras una recarga o un reinicio. Es por navegador y por dirección: abrir el receptor por nombre de host y por IP da dos ajustes separados, y una ventana privada siempre parte del valor por defecto del sitio.

Por encima de 30 MHz el S-metro sigue la norma de VHF de la IARU Región 1: allí S9 está en -93 dBm en lugar de -73 dBm en HF, igualmente con seis dB por unidad S. La aguja y la barra cambian solas con

la frecuencia sintonizada; las cifras en dBm y dBμV muestran siempre el nivel medido. Algunas estaciones configuran además sus receptores de VHF/UHF para que los medidores se queden en **0 con el canal vacío**, como en un transceptor de VHF/UHF: por encima de 60 MHz la aguja y la barra solo suben entonces cuando una señal destaca sobre el ruido. Las cifras de dBm, dBμV, SNR y NF muestran siempre los valores reales.

Elegir un receptor

Algunas estaciones tienen más de un receptor — por ejemplo uno para HF y otro para la banda de 2 m. Muestran en la cabecera de la página una línea *Receivers*: con un botón por receptor; aquel en el que está aparece en amarillo. Al pulsar otro botón se abre la página de ese receptor, con su propia cascada, bandas, chat y lista de oyentes. En la página /mobile los mismos botones están en la barra superior. Una estación con un solo receptor no muestra esa línea.

Funciones avanzadas

Control automático de ganancia (AGC)

El AGC ajusta automáticamente los niveles de audio:

- **Off**: sin ajuste automático de ganancia
- **Slow**: cambios de nivel graduales (ideal para SSB)
- **Medium**: respuesta equilibrada
- **Fast**: ajuste rápido (ideal para AM)

Recomendación: empiece con «Fast» para AM y «Slow» para SSB.

Los cuatro controles de ruido

NR, NB, NS y AN son botones de encendido/apagado independientes, cada uno contra un tipo distinto de ruido. Son independientes entre sí — encender uno no enciende otro — y se pueden combinar libremente.

Ninguno llega a los decodificadores: FT8, CW, WSPR, SSTV, FAX, NAVTEX, RTTY/PSK31/Olivia y el grabber QRSS leen el audio *antes* de estos filtros, así que puede ajustarlos puramente de oído sin afectar a lo que se decodifica. Véase el [manual de decodificadores](#).

Reducción de ruido (NR)

Reducción de ruido espectral. Estima el nivel de ruido en cada parte del espectro de audio y baja esas partes, dejando en paz lo que sobresale del ruido.

Úselo cuando: oiga un siseo constante o ruido blanco detrás de la señal.

Los tonos constantes — una nota de CW, una portadora — se reconocen como señal y quedan protegidos, de modo que NR no se come una señal de CW como haría un filtro ingenuo. Efecto típico: 10 dB menos de ruido entre las palabras, a cambio de unas décimas de dB de la voz misma.

Supresor de impulsos (NB)

Elimina el ruido impulsivo: clics, chasquidos, descargas atmosféricas, encendido de motores y ruido de la red eléctrica. Vigila la envolvente del audio y silencia solo las muestras que se disparan muy por encima de ella — alrededor de un milisegundo por descarga, con entrada y salida suavizadas para que el propio recorte no produzca un clic.

Úselo cuando: oiga clics o chasquidos de líneas eléctricas, motores, tormentas o encendido de vehículos.

Los notch de zumbido de red de 50 Hz y 60 Hz siguen a este botón.

Supresión del ruido de fondo (NS)

Mide durante varios segundos el propio suelo de ruido de la banda y aplica un corte fijo a lo que se sitúa en él, más profundo en las frecuencias de audio altas que en las bajas. Donde NR reacciona instante a instante, NS es la mano lenta y firme: baja el siseo de la banda sin cambiar cómo suena la señal.

Úselo cuando: la banda esté tranquila pero siseante y quiera bajar el ruido sin la calidad "submarina" de una NR agresiva.

Activo solo en USB, LSB y AM: CW, FM y los modos digitales quedan intactos. Vuelve a medir el suelo cada vez que cambia de frecuencia o de modo, y se asienta en pocos segundos.

Notch automático (AN)

Encuentra y elimina automáticamente tonos interferentes constantes — heterodinos, portadoras, silbidos — sin que usted tenga que colocar un notch a mano. Se adapta de forma continua, así que puede quitar varios tonos a la vez y mantener notchado uno que derive.

Úselo cuando: oiga un silbido o tono encima de la señal que quiere.

Se desactiva en CW, donde la señal deseada es un tono constante.

Nota sobre el retardo: NR y NS añaden cada uno unos 40 ms de retardo de audio mientras están activos (los dos juntos, unos 80 ms). NB y AN no añaden ninguno. Esto afecta solo a la escucha, nunca a los decodificadores.

Silenciador automático (SQL)

Silencia el audio cuando no hay señal:

- **Off:** siempre suena (se oye la estática)
- **Auto:** fija el umbral automáticamente
- **Manual:** ajuste manual del umbral

Úselo cuando: vigile una frecuencia a la espera de actividad.

Reducción de ruido por IA (AI)

El botón **AI**, con un deslizador de intensidad al lado, está en el panel Audio & Buffer, justo debajo de SQ. En la página /mobile está en la pestaña Audio, debajo de Squelch. Elimina el ruido de banda de la voz mediante RNNNoise, una pequeña red neuronal entrenada con voz.

Todo ocurre en su propio navegador: no se envía nada a ningún servidor externo, no hace falta cuenta y el receptor no tiene trabajo adicional.

- **Encendido/apagado:** pulse **AI**; el botón se vuelve azul. La primera pulsación descarga el módulo (unos 1,3 MB) y el botón parpadea mientras carga.
- **Intensidad:** fija con qué intensidad actúa mientras alguien habla: los valores más bajos conservan más del sonido original. El extremo superior de la escala es a propósito más suave que un corte total, para que la voz no se dañe ni siquiera al 100 %, que aun así limpia más que el 50 %. Entre palabras actúa con más fuerza por sí sola, de modo que el soplo baja aún más ahí. El valor por defecto es 50 %. Puede ajustarlo con la IA apagada.
- **Indicador de estado:** el indicador AI bajo la frecuencia se enciende en cian mientras AI trabaja, y aparece medio atenuado cuando AI está activo pero el modo actual no es de voz.

Úselo cuando: escuche voz en SSB o AM en una banda ruidosa. El soplado entre palabras suele bajar unos 10 dB con el 50 % predeterminado, y más al 100 %, hasta unos 20 dB, mientras la voz conserva su nivel.

Solo modos de voz — USB, LSB, AM y SAM. En CW, FM, modos digitales y C-QUAM el audio pasa sin cambios, porque la red trata un tono de CW o la música como ruido. Como los cuatro controles de ruido, nunca llega a los decodificadores. En estaciones débiles (por debajo de unos +6 dB de SNR) mide ella misma la relación señal/ruido y suaviza más sus decisiones, para que la voz no adquiera el sonido burbujeante, «acuoso», que puede dar este tipo de filtro; las estaciones fuertes se procesan exactamente igual que antes. Si una señal muy débil aún suena procesada, baje el control. Añade unos 50 ms de retardo al audio.

Búfer de audio y retardo

El audio llega al navegador en pequeños paquetes, de 35 ms cada uno en un receptor típico, y el navegador guarda un poco de reserva para que un paquete que llega tarde no provoque un hueco. Esa reserva es el **búfer**, y es la mayor parte del retardo entre el receptor y su altavoz.

- **Escritorio:** en el panel Audio & Buffer, el botón con el icono de red sube el búfer un paso por clic (se pone verde por encima del paso más bajo), y el control deslizante de al lado elige directamente un paso del 1 al 6. El paso más bajo va bien con una conexión por cable; los teléfonos empiezan en el paso 2.
- **/mobile:** el botón **BUF** y el control **Buffer** de la pestaña Audio hacen lo mismo.

El búfer se ajusta solo. La página mide lo irregular que es la entrega del audio en su conexión y guarda solo la reserva que el último minuto demostró necesaria, más el margen de seguridad del paso elegido: 20 ms en el paso 1, 50 ms en el paso 2 y más en los pasos superiores. Tras un momento de problemas de red, el retardo extra se vuelve a absorber en uno o dos minutos, y una pequeña diferencia entre el reloj de su tarjeta de sonido y el del receptor ya no provoca un corte cada pocos minutos. Los ajustes son de unos pocos milisegundos cada vez, se hacen donde el sonido se repite y no se oyen. Los decodificadores y sus grabaciones nunca se ven afectados: reciben todas las muestras.

Si el audio se corta, suba el búfer un paso. En un teléfono con Wi-Fi doméstico el retardo del receptor al altavoz suele ser de unos 100–150 ms; la red entre usted y el receptor se suma a eso.

Chat

Cuando el operador ha activado el chat, tiene su propio panel en la página de escritorio y una pestaña **Chat** en /mobile. Escriba su nombre o indicativo, luego su mensaje, y envíelo.

Para responder a un mensaje concreto, pulse ↩ **Reply** junto a su hora. Una línea sobre el cuadro de entrada muestra a quién y a qué responde; escriba y envíe como siempre, o pulse × o Esc para cancelar. La respuesta aparece sangrada bajo el mensaje al que responde, con una breve línea de cita. Una respuesta a una respuesta se une al mismo hilo. El servidor guarda los últimos 20 mensajes; cuando el original ya ha salido de ese historial, la respuesta muestra solo el nombre y la hora a los que responde.

Función de zoom

Amplía la cascada:

- **1x:** vista normal (cobertura amplia)
- **2x:** ampliación ×2
- **4x:** ampliación ×4
- **8x:** ampliación ×8

Úsela cuando: necesite ver las señales con más claridad o sintonizar con precisión.

Control del transceptor (CAT)

Su propio transceptor y la página del receptor pueden mantenerse en la misma frecuencia: gire el dial del equipo y la cascada lo sigue, o haga clic en una señal de la cascada y el equipo se sintoniza en ella. Solo afecta a *su* sesión de escucha; nadie más en el receptor lo nota.

Con Desktop PhantomSDR+ (4.0 o posterior). La aplicación de escritorio tiene un menú **Rig**. *Rig* → *Rig control...* abre una ventana donde elige su equipo y la forma de conectarlo, y el propio menú activa y desactiva la sincronización. Sincroniza frecuencia, modo y ancho de filtro, en un sentido o en ambos, y puede silenciar el receptor mientras transmite. Además de PhantomSDR-Plus maneja del mismo modo receptores **KiwiSDR**, **PA3FWM WebSDR** y **UberSDR**. Llega al equipo de una de estas cuatro formas:

- **Integrado** — sin otro software: Icom (CI-V), Yaesu (CAT nuevo y FT-817/857/897), Kenwood, Elecraft, FlexRadio SmartSDR CAT, QRP Labs y otros equipos compatibles con Kenwood.
- **Hamlib** — cualquier equipo que conozca Hamlib, más de 300, elegido de una lista con búsqueda. Los instaladores de Windows incluyen Hamlib; en Linux instale `libhamlib-utils`.
- **rigctld en la red** — un `rigctld` que ya está en marcha.
- **flrig** — para un equipo que flrig ya comparte con fldigi, WSJT-X o un programa de registro.

La aplicación, sus instaladores y su manual completo están en [Desktop PhantomSDR+ \(Dropbox\)](#).

Con un navegador web. La [CATsync Tool for WebSDRs](#) (Windows) acopla un equipo a la página del receptor abierta en su navegador. Sincroniza frecuencia y modo.

El ancho de filtro y el silencio al transmitir funcionan en receptores KiwiSDR, WebSDR y UberSDR, y en un receptor PhantomSDR-Plus con la 4.1.0 o posterior. En un PhantomSDR-Plus más antiguo la frecuencia y el modo siguen sincronizándose; el filtro no.

El manual completo — cada ajuste, los equipos admitidos, cómo se evita que los dos lados se peleen y la solución de problemas — es [Control del transceptor](#).

Para desarrolladores. Cada página de receptor ofrece estas funciones en `window`, que son las que usan ambas herramientas:

Función	Qué hace
<code>catsync_getFrequency()</code> / <code>catsync_setFrequency(hz)</code>	Frecuencia sintonizada, en Hz
<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>	Ancho total de la banda de paso, en Hz. Fíjelo después del modo: cambiar de modo restablece la banda de paso
<code>catsync_getMute()</code> / <code>catsync_setMute(on)</code>	Silencio, mediante el propio botón de silencio de la página
<code>catsync_ready</code>	<code>true</code> en cuanto las funciones están instaladas

Las tres últimas filas son nuevas en la 4.1.0; compruebe que una función existe antes de llamarla.

Modos de demodulación

AM (modulación de amplitud)

Se usa para:

- comunicaciones aeronáuticas

- radiodifusión en AM
- parte de la radioafición
- comunicaciones marítimas

Características:

- ancho de banda amplio (normalmente 10 kHz)
- sensible al ruido
- fácil de sintonizar (basta con hacer clic en la señal)

Buenas prácticas:

- use AGC rápido
- active el supresor de impulsos si oye chasquidos
- sintonice el pico de la señal en la cascada

QUAM (AM estéreo C-QUAM)

Se usa para:

- Emisoras de onda media que transmiten AM estéreo

Características:

- 10 kHz de ancho, se decodifica como par estéreo en lugar de AM mono
- La etiqueta del botón se pone **verde** por sí sola cuando está presente el piloto estéreo de 25 Hz, así se ve qué emisoras están realmente en estéreo antes de seleccionarlo
- Desde AM, pulsar de nuevo el botón pasa a detección síncrona sobre la portadora; la etiqueta se pone **amarilla** y muestra SAM. Una tercera pulsación vuelve al AM normal
- El audio C-QUAM viaja con Opus; el resto de modos usa FLAC

Buenas prácticas:

- Busque la etiqueta verde en señales fuertes de onda media por la noche
- Si el estéreo suena inestable, SAM es más firme con portadora débil

FM (modulación de frecuencia)

Se usa para:

- repetidores de radioafición de VHF/UHF
- servicios públicos (policía, bomberos, emergencias)
- radio comercial bidireccional
- algunas comunicaciones por satélite

Características:

- ancho de banda estrecho (normalmente 12,5 o 25 kHz)
- excelente inmunidad al ruido
- «efecto captura» (gana la señal más fuerte)

Buenas prácticas:

- sintonice con precisión el centro de la señal
- use el silenciador para enmudecer en reposo
- desactive la reducción de ruido (no hace falta)

USB (banda lateral superior)

Se usa para:

- radioafición en HF (por encima de 10 MHz)
- la mayoría de los modos digitales de HF
- comunicaciones marítimas (por encima de 8 MHz)

Características:

- ancho de banda estrecho (normalmente 2.4 kHz)
- uso eficiente del espectro
- exige sintonía precisa

Buenas prácticas:

- use AGC lento
- sintonice el borde inferior de la señal en la cascada
- active la reducción de ruido si hace falta

LSB (banda lateral inferior)

Se usa para:

- radioafición en HF (por debajo de 10 MHz)
- algunos modos digitales de HF
- comunicaciones marítimas (por debajo de 8 MHz)

Características:

- igual que USB pero en imagen especular
- convención: LSB en las bandas bajas de HF

Buenas prácticas:

- use AGC lento
- sintonice el borde superior de la señal en la cascada
- active la reducción de ruido si hace falta

CW (onda continua / código Morse)

Se usa para:

- telegrafía de radioaficionados
- radiobalizas de navegación
- estaciones de señales horarias

Características:

- ancho de banda muy estrecho (100-500 Hz)
- gran eficiencia
- requiere conocer el código Morse para entenderlo

Buenas prácticas:

- use un filtro estrecho (400-500 Hz)
- sintonice con precisión el centro del tono
- active el filtro de audio para obtener mejor tono

CW-L (CW, banda lateral inferior)

El mismo filtro Morse de ± 250 Hz que **CW**, pero el tono se toma por debajo de la portadora en lugar de por encima. Úselo cuando una señal se copie mejor por el lado bajo, o cuando una portadora interferente quede justo por encima de la deseada.

No hay botón CW-L en la página de escritorio. Su fila de modos es **USB · LSB · CW · AM · QUAM · FM**. CW-L se ofrece en la página simplificada `/mobile`, y en escritorio todavía puede fijarse mediante un enlace o un marcador que lo nombre.

WBFM (FM de banda ancha)

Se usa para:

- radiodifusión en FM (88-108 MHz)
- algunos enlaces descendentes de satélite

Características:

- ancho de banda muy amplio (200 kHz)
- excelente calidad de audio
- alta fidelidad

Buenas prácticas:

- sintonice exactamente la frecuencia central
- no hace falta silenciador para la radiodifusión
- ¡disfrute de un audio de gran calidad!

Decodificadores digitales

PhantomSDR-Plus incluye decodificadores integrados para modos digitales. Para una guía completa, consulte [Decodificadores](#).

La vía más rápida es la fila de botones **Decoders** del panel principal, justo debajo de **Wheel Tuning Steps**: diez botones — **FT8, FT4, FT2, JS8, CW, WSPR, FAX, SSTV, NAVTX, RTTY** — que, con una sola pulsación, activan su decodificador, ponen en ON el interruptor principal Decoder y desplazan la ventana del decodificador hasta hacerla visible. El botón permanece azul mientras el decodificador funciona; púlselo de nuevo para detenerlo y cerrar su ventana. El desplegable **Decoder Options** sigue funcionando igual que antes y se mantiene sincronizado con los botones.

RADEL y **RADEU** (voz digital RADE v1) tienen sus propios botones junto al encabezado **Modes selector** y dentro de las ventanas emergentes **Modes** y **Bands**, y funcionan del mismo modo: pulsar para iniciar, volver a pulsar para detener.

Los decodificadores de SSTV, FAX de HF, NAVTEX, FSK/RTTY y CW se ejecutan cada uno en su propio hilo en segundo plano, de modo que activar o desactivar un decodificador nunca interrumpe el audio y la cascada sigue fluida mientras se decodifica. Reciben audio tomado antes del AGC, la reducción de ruido y el silenciado: puede silenciar el receptor y la decodificación continúa sin verse afectada.

Mientras un decodificador está en marcha, retiene el modo y el paso de banda. Normalmente el modo sigue el plan de bandas: sintonice un segmento marcado LSB o AM y el receptor cambia. Un decodificador en marcha manda por encima de eso y conserva el modo y el paso de banda estrecho que necesita, aunque se vaya a otra banda. El modo propio de la banda vuelve en cuanto apaga el decodificador, y los botones de modo funcionan en todo momento si quiere tomar el mando. El decodificador de CW es la excepción: decodifica en el modo en que usted esté escuchando.

Decodificador FT8, FT4

¿Qué son FT8 y FT4?

- modos digitales muy populares en radioafición
- comunicación con señales débiles
- transmisiones de 15 segundos en FT8 y de 7,5 segundos en FT4

Cómo se usa:

1. Sintonice frecuencias de FT8 o FT4
2. Seleccione el modo USB
3. Active el decodificador FT8: pulse el botón **FT8** o selecciónelo desde el menú
4. Observe cómo aparecen los mensajes decodificados

Frecuencias habituales de FT8 (USB):

- 160 metros: 1.840 MHz
- 80 metros: 3.573 MHz
- 60 metros: 5.357 MHz
- 40 metros: 7.074 MHz
- 30 metros: 10.136 MHz
- 20 metros: 14.074 MHz
- 17 metros: 18.100 MHz
- 15 metros: 21.074 MHz
- 12 metros: 24.915 MHz
- 10 metros: 28.074 MHz
- 6 metros: 50.313 MHz (50.323 MHz para DX)

Frecuencias habituales de FT4 (USB):

- 80 m: 3.575 MHz
- 40 m: 7.0475 MHz
- 30 m: 10.140 MHz
- 20 m: 14.080 MHz
- 17 m: 18.104 MHz
- 15 m: 21.140 MHz
- 12 m: 24.919 MHz
- 10 m: 28.180 MHz
- 6 m: 50.318 MHz

Decodificador JS8

¿Qué es JS8?

- El motor de señal débil de FT8 usado para conversación de teclado a teclado
- Texto libre en vez de intercambios fijos: los mensajes largos llegan en varios ciclos
- Cinco velocidades; **Normal** (15 s) es la de llamada y concentra casi todo el tráfico

Cómo usarlo:

1. Sintonice una frecuencia de JS8
2. Seleccione modo **USB**

3. Active el decodificador JS8 — botón **JS8** o desde el menú

4. Deje **Speed** en *Normal* y el sync offset en **Auto**

Cómo leer el panel:

- Los mensajes que aún llegan se muestran arriba en verde con un cursor parpadeante — en Normal un mensaje puede tardar un minuto entero
- Los completos se listan debajo como `Mode | Hz | dB | Message`, con **los indicativos en verde**
- Una fila atenuada y en cursiva expiró antes de su última trama: el texto es real pero puede estar cortado

Frecuencias JS8 habituales (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 es mucho más tranquilo que FT8: una transmisión cada pocos minutos es normal y los silencios no son un fallo. Guía completa: [Decodificadores](#).

Decodificador de CW

Basta con pulsar el botón CW y aparecerá el texto decodificado. Pulse el botón CW una vez más para limpiar la ventana y reiniciar el decodificador.

WSPR

WSPR (Weak Signal Propagation Reporter, pronunciado «whisper») es un modo baliza de señales ultradébiles que cartografía las trayectorias de propagación en HF por todo el mundo. Cada transmisión dura unos 110 segundos y cabe en una ranura de 200 Hz de ancho. El decodificador espera a una ranura completa de 2 minutos alineada con UTC antes de decodificar. Active el decodificador y seleccione **WSPR** en la lista desplegable. O basta con pulsar el botón **WSPR**.

FAX de HF / WEFAX

El radiofax de HF (también conocido como WEFAX) lo utilizan los servicios de guardacostas y meteorológicos de todo el mundo para difundir mapas del tiempo, cartas del estado del mar y análisis de superficie por onda corta. El decodificador reconstruye la imagen línea a línea a medida que se recibe.

NAVTEX

NAVTEX es el sistema internacional de radiodifusión marítima de información de seguridad costera: avisos a la navegación, previsiones meteorológicas y avisos de búsqueda y rescate.

FSK / RTTY, PSK31 y Olivia

Un decodificador de uso general para modos de texto de banda estrecha, con cinco variantes en una sola ventana: FSK marítimo (SITOR), RTTY meteorológico, RTTY de aficionado, **PSK31** (modulación por desplazamiento de fase, 31,25 baudios) y **Olivia** (FSK multitono con corrección de errores). El panel se adapta a la variante — los controles de shift, baudios y trama desaparecen en PSK31 y Olivia, y Olivia añade un selector de Mode y un deslizador de squelch.

El RTTY de aficionados y el meteorológico tienen un control **Squelch (SNR)** que evita que el ruido se imprima como letras al azar, y todas las variantes salvo Olivia muestran una SNR real en la misma escala de 3 kHz. Dos cosas que conviene saber: PSK31 corrige por sí mismo su error de sintonía en unos ± 25 Hz, así que basta con acercarse; Olivia necesita que **Mode** (tonos / ancho de banda) coincida exactamente con la transmisión, no envía preámbulo y por tanto tarda unos segundos en sincronizarse antes de que aparezca texto. El panel se abre en Olivia **8 / 250**.

Grabber QRSS

QRSS es CW enviado tan despacio que un solo punto dura segundos, y se mira en lugar de escucharse: el grabber dibuja la traza en su propia presentación. No está en el desplegable de decodificadores; tiene su propia sección **QRSS** y puede funcionar a la vez que un decodificador.

Pulse **Show**, elija después una ventana en la lista **Band** y pulse **Tune**: el receptor va a esa frecuencia en **CW** con un paso de banda a la medida del modo, y la traza cae sobre la línea central de la presentación. La ventana más concurrida es la de 30 m (10.140,00 kHz). Ajuste **Speed** a la baliza —**QRSS 10** si no lo sabe— y tenga paciencia: un indicativo puede tardar diez minutos en cruzar la pantalla. Todos los detalles en [Decodificadores](#).

SSTV

La televisión de barrido lento envía imágenes fijas por un canal SSB normal, línea a línea. Active el decodificador, seleccione **SSTV** y sintonice una frecuencia de SSTV: 14.230 MHz es la principal frecuencia internacional de llamada. Deje **Mode** en **Auto**: el decodificador lee la cabecera VIS de la transmisión y, si sintonizó después de la cabecera, identifica el modo a partir de la temporización de sincronismo. Se admiten los modos Martin, Scottie y Robot, y la imagen se va formando línea a línea según llega. O basta con pulsar el botón **SSTV**.

Reporte automático de spots y gráficas del servidor

Las decodificaciones de FT8, FT4 y WSPR también puede subirlas el propio servidor — FT8/FT4 a PSK Reporter y WSPR a WSPRnet — mediante un demonio autorun que el sysop inicia desde el panel de administración. Es independiente de los decodificadores de su navegador: sigue funcionando haya o no oyentes, y nada de lo que usted decodifique en el navegador se reporta.

El sysop lo sigue con dos contadores, fáciles de confundir entre sí: los paneles por decodificador cuentan los spots subidos desde el último arranque del demonio, mientras que el número junto a cada casilla de banda/modo es el total histórico de esa ranura y sobrevive a los reinicios. El mismo panel tiene una página **Gráficos** que representa la frecuencia de CPU, la carga, la temperatura y los usuarios conectados en los últimos 15 minutos a 24 horas. Ambos se describen en la [guía del panel de administración](#).

Atajos de teclado

Atajos de teclado para trabajar más rápido.

Control de frecuencia

El indicador de frecuencia es una fila de dígitos que se maneja directamente. **Haga clic primero en un dígito:** eso lo selecciona, y todo lo de abajo actúa sobre la selección.

- **Flecha izquierda / derecha:** mueve la selección al dígito contiguo (ocho dígitos, de 100 MHz a 10 Hz)
- **Flecha arriba / abajo:** sube o baja el dígito *seleccionado* según su valor posicional: 1 MHz en el dígito de los MHz, 10 Hz en el último
- **0 - 9:** escribe ese dígito directamente en la posición seleccionada

Rueda del ratón sobre el indicador de frecuencia: avanza con el paso de sintonía de la banda (1 kHz salvo que el plan de bandas indique otro). Mantenga **Shift** para 1 kHz y **Alt** para 10 kHz.

Rueda del ratón sobre la cascada: hace zoom. Mantenga **Ctrl** (o **Cmd**) o **Shift** para sintonizar en lugar de ampliar, y **Shift + Ctrl** juntos para ajustar al kHz entero más cercano.

No hay atajos de Página arriba / Página abajo.

Marcadores

Guarde sus frecuencias favoritas para acceder a ellas rápidamente. También puede exportar la lista de marcadores y guardarla localmente, y después importarla en cualquier otro PhantomSDR.

Cuando los marcadores propios y los del plan de bandas se solapan:

Los marcadores azules aparecen encima

Los marcadores amarillos, debajo

Los clics en los marcadores azules tienen prioridad

Añadir un marcador

1. Sintonice la frecuencia deseada
2. Haga clic en el botón «Bookmarks»
3. Haga clic en «Add Bookmark»
4. Escriba una descripción
5. Haga clic en «Save»

			Bookmark name Label (optional) [Add]
--	--	--	--

Cómo se usa:

Añadir un marcador:

- Nombre: «Local News Station»
- Etiqueta: «NEWS»
- Haga clic en Add

Verlo en la cascada:

- amplíe hasta que aparezcan los marcadores
- verá un recuadro azul marino con «NEWS» en amarillo y negrita

Hacer clic en el marcador:

- sintoniza la frecuencia

- fija el modo de demodulación
- funciona igual que hacer clic en un marcador del plan de bandas

Gestionar los marcadores

- **Editar:** haga clic en el icono del lápiz junto al marcador
- **Eliminar:** haga clic en el icono de la papelera junto al marcador
- **Exportar:** descargue los marcadores como archivo JSON
- **Importar:** cargue marcadores desde un archivo JSON

Compartir marcadores

1. Haga clic en «Export Bookmarks»
2. Comparta el archivo JSON con otras personas
3. Los destinatarios hacen clic en «Import Bookmarks»
4. Seleccionan su archivo

Uso en móviles

¡PhantomSDR-Plus funciona de maravilla en dispositivos móviles!

Dos vistas móviles

Hay dos formas de usar el receptor desde el teléfono:

- **http://su_servidor:PUERTO/mobile:** la página simplificada. Sin cascada, por lo que consume aproximadamente la mitad de datos. Entrada de frecuencia, pasos de sintonía, modos, S-meter, bandas, marcadores, usuarios y chat.
- **La vista ampliada:** la disposición para teléfono de la interfaz principal, con la cascada y todos los controles.

Se cambia con los botones de la parte inferior de `/mobile` (**Mobile extended view, Full desktop view**) y con **Simplified mobile** en la vista ampliada.

Su frecuencia le acompaña. Cambiar de vista le mantiene en la misma señal: la frecuencia y el modo viajan en el enlace, así que ya no acaba en la frecuencia predeterminada del receptor. La barra de direcciones también sigue su sintonía, de modo que recargar la página, guardarla como marcador o enviar el enlace a alguien devuelven exactamente a esa frecuencia.

El modo sigue el plan de bandas en ambas páginas. Sintonice un segmento marcado como AM, LSB, USB o CW —escribiendo una frecuencia, avanzando a pasos o pulsando un botón de banda— y el receptor cambia a ese modo, tanto en la página simplificada como en la interfaz completa. Un modo elegido a mano se mantiene mientras se mueve dentro del mismo segmento, y fuera de las bandas definidas su modo no se toca. Mientras **RADE** (RADEL/RADEU) está en marcha conserva el receptor, de modo que sintonizar no lo interrumpe.

Al cambiar entre las dos vistas su modo actual viaja con usted, pero si la vista a la que pasa no tiene un equivalente, el plan de bandas decide para esa frecuencia: una frecuencia de radiodifusión llega en AM, 40 m en LSB, un segmento de CW en CW. **SAM** en la página simplificada se convierte en AM con el detector síncrono en la vista ampliada, y a la inversa. **RADEL** y **RADEU** solo existen en la página simplificada: al salir de uno de ellos se conserva la frecuencia y el plan de bandas elige el modo.

Una frecuencia fuera de la cobertura del receptor se ajusta al extremo más cercano, de manera que un enlace antiguo nunca puede dejarle fuera de banda.

Funciones específicas para móviles

- **Controles táctiles:** botones y deslizadores grandes
- **Deslizar para sintonizar:** deslice a izquierda/derecha sobre la cascada
- **Pellizcar para ampliar:** pellizque la cascada para acercar o alejar
- **Modo horizontal:** gire el dispositivo para ver mejor

Consejos para móviles

1. **Use wifi:** la transmisión de audio consume datos
2. **Orientación horizontal:** mejor vista de la cascada
3. **Auriculares:** mejor calidad de audio
4. **Guarde sus favoritas como marcadores:** es más fácil volver a las emisoras
5. **Cierre otras aplicaciones:** garantiza un funcionamiento fluido

Navegadores móviles recomendados

- **Android:** Chrome o Samsung Internet
- **iOS:** Mozilla
- **Ambos:** asegúrese de que el navegador esté actualizado

Consejos y buenas prácticas

Para la mejor recepción

1. **Elija señales fuertes:** busque el naranja/rojo en la cascada
2. **Sintonice con precisión:** haga clic justo en el centro de la señal
3. **Seleccione el modo correcto:** adecuado al tipo de señal
4. **Ajuste el AGC:** rápido para AM, lento para SSB
5. **Use NR/NB:** ayudan en condiciones ruidosas

Encontrar actividad

1. **Observe la cascada:** los colores indican la intensidad de la señal
2. **Escuche en frecuencias populares:**
3. 40 m: 7.100-7.300 MHz (LSB)
4. 20 m: 14.200-14.350 MHz (USB)
5. 2 m: 145.200-145.600 MHz (FM)
6. **Consulte la superposición del plan de bandas:** muestra las atribuciones de frecuencia
7. **Use los marcadores:** acceso rápido a frecuencias activas

Entender las condiciones de propagación

HF de día:

- las bandas altas funcionan mejor (20 m, 15 m, 10 m)
- es posible la comunicación a larga distancia (DX)
- se oyen emisoras de radiodifusión

HF de noche:

- las bandas bajas funcionan mejor (80 m, 40 m)

- pautas de propagación distintas
- se oyen emisoras distintas

VHF/UHF:

- principalmente con visibilidad directa
- comunicaciones locales
- condiciones más constantes

Etiqueta

1. **No monopolice el receptor:** otras personas también quieren escuchar
 2. **Use el chat con respeto:** sea cortés con los demás usuarios
 3. **Informe de los problemas:** ayude al operador a mantener la estación
 4. **No pida soporte técnico:** esta es una plataforma de escucha. Envíe un mensaje al SysOp si necesita ayuda.
-

Resolución de problemas

No hay audio**Posibles causas:**

1. Navegador silenciado → compruebe los controles de volumen del navegador
2. Sistema silenciado → compruebe el volumen del ordenador
3. Señal débil → sintonice una señal más fuerte (S7+)
4. Modo incorrecto → pruebe otros modos de demodulación

Soluciones:

1. Haga clic en una señal fuerte (naranja/roja en la cascada)
2. Compruebe que el navegador no esté silenciado (icono de silencio en la pestaña)
3. Pruebe otra frecuencia
4. Recargue la página (F5)

Audio distorsionado**Posibles causas:**

1. Señal saturada → señal demasiado fuerte
2. Modo incorrecto → señal de AM escuchada en SSB, etc.
3. Interferencias → señales adyacentes que se cuelan

Soluciones:

1. Baje el volumen
2. Pruebe otro modo de demodulación
3. Use un filtro más estrecho
4. Aléjese de las señales interferentes

El audio se corta o va con retraso

Posibles causas:

1. Conexión irregular → los paquetes llegan tarde y el búfer se vacía
2. Teléfono u ordenador ocupado → el navegador no da abasto

Soluciones:

1. Suba el búfer un paso (vea *Búfer de audio y retardo*)
2. Prefiera Wi-Fi o cable a los datos móviles
3. Cierre otras pestañas y aplicaciones

Para verlo con detalle, abra la página añadiendo `?diag=1` a su dirección (por ejemplo `http://your_server:PORT/mobile?diag=1`). La página `/mobile` muestra entonces bajo sus controles el retardo, lo irregular que es la conexión y el estado del búfer; la página de escritorio escribe la misma línea en la consola del navegador (F12) cada diez segundos. Útil al comunicar un problema al operador de la estación.

La cascada no se actualiza

Posibles causas:

1. Problema de red → conexión lenta o interrumpida
2. Rendimiento del navegador → demasiadas pestañas abiertas
3. Sobrecarga del servidor → demasiados usuarios

Soluciones:

1. Compruebe la conexión a Internet
2. Cierre las pestañas innecesarias
3. Recargue la página (F5)
4. Inténtelo más tarde, cuando haya menos usuarios conectados

No se puede sintonizar una frecuencia

Posibles causas:

1. Frecuencia fuera de rango → el SDR no cubre esa frecuencia
2. Formato de escritura incorrecto → use el formato correcto (p. ej., «14200» y no «14.200.000»)

Soluciones:

1. Compruebe la cobertura de frecuencias del SDR (indicada en la página)
2. Use los ejemplos de formato de frecuencia facilitados
3. Haga clic en la cascada en su lugar

Audio entrecortado

Posibles causas:

1. conexión a Internet lenta
2. carga elevada del servidor
3. problemas de rendimiento del navegador

Soluciones:

1. Cierre otras aplicaciones que consuman ancho de banda
2. Inténtelo fuera de las horas punta

3. Cierre las pestañas innecesarias
4. Use conexión por cable en lugar de wifi

Preguntas frecuentes

Preguntas generales

P: ¿Necesito equipo especial para usar WebSDR? R: ¡No! Solo un ordenador o dispositivo móvil con acceso a Internet.

P: ¿Es gratuito usar WebSDR? R: Sí, la mayoría de los WebSDR son gratuitos. Los mantienen voluntarios.

P: ¿Puedo transmitir con WebSDR? R: No, WebSDR es solo de recepción. No se puede transmitir.

P: ¿Qué frecuencias puedo escuchar? R: Depende de la configuración del WebSDR. Consulte la información de la estación.

P: ¿Puedo grabar el audio? R: Algunos navegadores permiten grabar. Compruebe las funciones de su navegador.

Preguntas técnicas

P: ¿Qué tasa de muestreo usa el SDR? R: Varía según la estación. Consulte la página de información de la estación.

P: ¿Cuál es la latencia? R: Normalmente unas décimas de segundo entre la señal de radio y sus altavoces: el procesado en el receptor, la red y el búfer de reproducción del navegador, que se ajusta a su conexión (unos 100–150 ms en un teléfono con Wi-Fi doméstico). Vea *Búfer de audio y retardo*.

P: ¿Puedo abrir varias instancias? R: Normalmente sí, pero puede sobrecargar el servidor. Sea considerado.

P: ¿Funciona sin conexión? R: No, WebSDR requiere conexión a Internet.

P: ¿Qué navegadores son compatibles? R: Chrome, Firefox, Edge y Safari (versiones recientes)

Preguntas sobre el uso

P: ¿Cuántas personas pueden escuchar a la vez? R: Depende de la capacidad del servidor. A menudo entre 50 y más de 200 usuarios.

P: ¿Puedo ver qué escuchan los demás? R: Si está habilitado, sí. Busque los indicadores de «otros usuarios».

P: ¿Puedo chatear con otros oyentes? R: Si el operador lo ha activado. Busque el cuadro de chat; ➡ **Reply** responde a un mensaje concreto (vea *Chat* más arriba).

P: ¿Por qué algunas frecuencias no muestran nada? R: En ese momento no hay señales en esa frecuencia. ¡Pruebe otras!

P: ¿Qué son las bandas de color de la cascada? R: La superposición del plan de bandas, que muestra las atribuciones de frecuencia.

Recursos

Aprender más sobre radio

- **Planes de bandas:** busque «amateur radio band plan» + su región
- **Propagación:** infórmese sobre la propagación de las ondas de HF
- **Modos digitales:** investigue sobre FT8, PSK31 y RTTY
- **Radioafición:** ¡plantéese obtener una licencia de radioaficionado!

Encontrar más WebSDR

- **Directorio WebSDR:** <http://sdr-list.xyz>
- **WebSDR.org:** <http://websdr.org>
- **KiwiSDR:** <http://kiwisdr.com/public/>

Obtener ayuda

1. **Operador de la estación:** consulte los datos de contacto en la página
2. **Chat de usuarios:** pregunte a otros oyentes (si está disponible)
3. **Foros en línea:** busque comunidades de WebSDR
4. **Documentación:** ¡consulte esta guía!

Apéndice: frecuencias habituales

Bandas de radioafición de HF

Banda	Margen de frecuencias	Modo	Actividad
160 m	1.800-2.000 MHz	LSB	Noche/local
80 m	3.500-4.000 MHz	LSB	Noche/regional
40 m	7.000-7.300 MHz	LSB	Día/noche/DX
30 m	10.100-10.150 MHz	USB	Solo datos/CW
20 m	14.000-14.350 MHz	USB	Día/DX
17 m	18.068-18.168 MHz	USB	Día/DX
15 m	21.000-21.450 MHz	USB	Día/DX
12 m	24.890-24.990 MHz	USB	Día/DX
10 m	28.000-29.700 MHz	USB	Esporádica/DX

Bandas de radioafición de VHF/UHF

Banda	Margen de frecuencias	Modo	Actividad
6 m	50.000-54.000 MHz	USB/FM	Esporádica
2 m	144.000-148.000 MHz	FM	Muy activa
70 cm	420.000-450.000 MHz	FM	Activa

Bandas de radiodifusión

Servicio	Margen de frecuencias	Modo
Radio AM	530-1710 kHz	AM
Onda corta	2.3-26.1 MHz	AM
Radio FM	88-108 MHz	WBFM

Aviación

Servicio	Margen de frecuencias	Modo
Control del tráfico aéreo	118-137 MHz	AM
ACARS (datos)	130-136 MHz	Datos

Marítimo

Servicio	Margen de frecuencias	Modo
VHF marina	156-162 MHz	FM
HF marina	2-22 MHz	USB

Glosario

AGC: control automático de ganancia — ajusta los niveles de audio automáticamente

AM: modulación de amplitud — modo de voz usado en aviación y radiodifusión

Ancho de banda: el margen de frecuencias de una señal

CW: onda continua — señales en código Morse

DX: comunicación a larga distancia

FFT: transformada rápida de Fourier — convierte el dominio del tiempo en el de la frecuencia

FM: modulación de frecuencia — modo de voz para VHF/UHF

HF: alta frecuencia (3-30 MHz) — bandas de larga distancia

kHz: kilohercio (1.000 Hz)

LSB: banda lateral inferior — modo de voz para las bandas bajas de HF

MHz: megahercio (1.000.000 Hz)

NB: supresor de impulsos — elimina el ruido impulsivo

NR: reducción de ruido — reduce el ruido de fondo

PSK: modulación por desplazamiento de fase — modo digital

RTTY: radioteletipo — modo digital de texto

S-metro: medidor de intensidad de señal

SDR: radio definida por software

SQL: silenciador — enmudece el audio cuando no hay señal

SSB: banda lateral única (USB o LSB)

USB: banda lateral superior — modo de voz para las bandas altas de HF

VHF: muy alta frecuencia (30-300 MHz) — visibilidad directa

UHF: ultra alta frecuencia (300-3000 MHz) — visibilidad directa

Cascada: representación visual del espectro radioeléctrico a lo largo del tiempo

¡Disfrute explorando el espectro radioeléctrico con PhantomSDR-Plus!

73 (saludos cordiales) de SV1BTL & SV2AMK

Para las instrucciones de instalación, consulte [INSTALLATION.md](#). Para los detalles técnicos, consulte [PROJECT_STRUCTURE.md](#).

CAPÍTULO 13

Emulación de cliente KiwiSDR

Aether_config.md

Emulación de clientes KiwiSDR

Conectar AetherSDR, kiwiclient y otro software KiwiSDR a PhantomSDR-Plus

Desde la v4.1.0 PhantomSDR-Plus también puede responder al **protocolo KiwiSDR**, de modo que el software escrito para un KiwiSDR — **AetherSDR**, **kiwiclient** y los demás — se conecta directamente a su receptor. Es un puente dentro del mismo proceso del servidor, en el mismo host y el mismo puerto que ya publica: sin un segundo servicio, sin un segundo puerto, sin proxy.

Está apagado hasta que usted lo active. Nada del puente se ejecuta — ni un socket, ni una comprobación — mientras `[kiwi_emulation]` falte o sea `false`, que es el estado de un árbol recién instalado.

Índice

1. Qué obtiene un cliente Kiwi
2. Instalar el puente
3. Activarlo
4. Referencia de configuración
5. Conectar un cliente
6. Nivel de audio
7. El S-meter
8. Cascada y espectro
9. Si algo no funciona bien
10. Hasta dónde se ha verificado

1. Qué obtiene un cliente Kiwi

Audio	<code>ws://<host>:<port>/kiwi/<id>/SND</code> — audio demodulado en el formato SND propio del Kiwi
Cascada	<code>ws://<host>:<port>/kiwi/<id>/W/F</code> — el espectro, en tramas W/F del Kiwi
Sintonía	real, no cosmética: <code>SET mod=...</code> desde el cliente cambia frecuencia, banda lateral y ancho de paso en el lado de PhantomSDR, exactamente como lo haría un oyente desde el navegador
S-meter	alimentado por la potencia por bloque del propio demodulador, en la misma escala que el S-meter web

La sintonía se admite para una entrada **real** (`signal = "real"` en su `.toml`). Con una entrada IQ la relación entre bin y frecuencia es distinta y no está implementada, así que un receptor IQ servirá audio y cascada pero no seguirá la sintonía de un cliente Kiwi.

2. Instalar el puente

El puente es un conjunto de parches a las fuentes del backend más una cabecera nueva, `src/kiwi_bridge.h`. Hay dos formas de incorporarlos.

Con el instalador. `install.sh` (y `install_arch.sh`, `install_fedora.sh`, `install_opensuse.sh`) tienen un paso propio para ello — el paso 17, *Kiwi client emulation*. Se ofrece, no se impone: responda `n` y no se parchea nada. Para una ejecución desatendida lo decide `PHANTOM_KIWI=y|n`, y por omisión es `sí`.

En un árbol ya instalado, ejecute el script que viene en la raíz del repositorio:

```
cd ~/PhantomSDR-Plus
./kiwi_install.sh
./recompile.sh          # elija [1] Solo backend
```

`kiwi_install.sh` respalda primero cada archivo que toca en `backup_kiwi_bridge_<marca-de-tiempo>/` y aplica cada parche por coincidencia exacta de texto — si un anclaje no está donde lo espera (árbol modificado localmente, otra versión), se detiene ahí mismo, nombra el archivo y lo deja intacto. Es idempotente: en un árbol que ya tiene el puente informa de cada parche como ya aplicado y no cambia nada.

Si acaba de aplicar un archivo de actualización que ya contiene el puente, no necesita ejecutarlo. Las fuentes parcheadas vienen en el paquete; `kiwi_install.sh` solo le diría que cada parche ya está puesto.

Los archivos que parchea son `src/client.h`, `src/signal.cpp`, `src/waterfall.cpp`, `src/spectrumserver.h`, `src/spectrumserver.cpp`, `src/websocket.cpp` y `src/http.cpp`, y copia `kiwi_bridge.h` a `src/`.

Instala el puente; no lo actualiza. Si algún parche *falla* en un árbol donde el puente ya funciona, el código ha avanzado por delante del script — y es el script lo que hay que actualizar, no su árbol.

3. Activarlo

Añada esto al `.toml` con el que realmente arranca su receptor — el mismo archivo que pasa a `spectrumserver`, el que nombra su `start-<radio>.sh`. No `config.example.*`:

```
[kiwi_emulation]
enabled = true
```

Después reinicie el receptor:

```
./stop-websdr.sh
./start-<su-radio>.sh
```

`kiwi_install.sh` añade el bloque por usted, con `enabled = true`, a los que existan de entre `config.toml`, `config-rx888mk2.toml`, `config-airspyhf.toml`, `config-rtl.toml`, `config-rspla.toml`, `config-fobos.toml`, `config-fobos-hf.toml` y `config-hackrf.toml` en la raíz del repositorio. No puede saber de una configuración que usted guarde en otro sitio o con otro nombre, y **los archivos de actualización nunca incluyen un `.toml`**, así que tras una actualización el bloque le corresponde ponerlo a mano. Los `config.example.*.toml` que se distribuyen lo llevan documentado y en `false`.

4. Referencia de configuración

Todo lo que el puente lee vive en una única sección `[kiwi_emulation]`. Solo `enabled` es obligatorio; los cuatro ajustes finos tienen por omisión el valor calibrado, así que un bloque con solo `enabled = true` ya es correcto.

Clave	Por omisión	Qué hace
<code>enabled</code>	<code>false</code>	Responder al protocolo KiwiSDR. Apagado = completamente inerte.
<code>audio_gain</code>	<code>0</code>	dB. Ganancia de salida, solo para clientes Kiwi. Véase Nivel de audio .
<code>smeter_offset</code>	<code>input.analog_smeter_offset</code>	dB. Sustituye el desplazamiento del S-meter, solo para clientes Kiwi. Véase El S-meter .
<code>wf_cal</code>	<code>0</code>	dB. Ajuste de visualización para la cascada y el espectro. Véase Cascada y espectro .
<code>wf_fps_max</code>	<code>23</code>	Cuadros de cascada por segundo. Véase Cascada y espectro .

Un bloque completo, con los valores con los que funciona el receptor de este proyecto:

```
[kiwi_emulation]
enabled      = true
audio_gain   = 60.0    # dB, solo clientes Kiwi. 0 = sin tocar.
wf_cal       = -10.0   # dB, cuestión de gusto. 0 = concuerda con el S-meter.
wf_fps_max   = 28      # cuadros/s. 23 es el máximo del protocolo.
# smeter_offset = 5.0  # dB. Por omisión, input.analog_smeter_offset.
```

Todas estas claves se leen al arrancar. Cambiar cualquiera de ellas es un **reinicio**, no una recompilación — `./stop-websdr.sh` y luego su script de arranque. Nunca hace falta recompilar para reajustarlas.

Se aceptan tanto la forma entera como la decimal: 28 y 28.0 se leen igual. Solo una clave realmente ausente cae a su valor por omisión.

5. Conectar un cliente

Apunte el cliente a la misma dirección y puerto que usan sus oyentes — sin ruta, sin prefijo. AetherSDR pide un host y un puerto; `kiwiclient` toma `-s` y `-p`:

```
# kiwiclient, grabando 30 segundos de 7100 kHz LSB
python3 kiwirecorder.py -s su.receptor.example -p 8073 -f 7100 -m lsb --tlimit=30
```

El cliente sintoniza, la cascada se llena y el S-meter marca. Si no ocurre absolutamente nada, el puente casi con seguridad sigue desactivado — véase la [sección 3](#).

6. Nivel de audio

Lo primero que casi todo el mundo nota es que un cliente Kiwi suena más delgado que la página web de PhantomSDR. No es que el puente pierda nada: un solo búfer alimenta a todos los codificadores, y la misma frecuencia con el mismo ancho de paso, medida por la ruta de audio del navegador y por `/SND`,

da muestras idénticas. La página web suena fuerte porque `audio.js` reconstruye el sonido en el navegador — realce de graves, filtro de paso de banda, realce de presencia, un compresor con ganancia de compensación y el control de volumen. Un cliente Kiwi no tiene nada de eso y no se le puede dar.

`audio_gain` cierra esa diferencia, solo para clientes Kiwi:

```
[kiwi_emulation]
enabled      = true
audio_gain = 55.0      # dB, solo clientes Kiwi. 0 = sin tocar.
```

Empiece alrededor de 55-60 dB. 55 dB no es aritmética: es donde un cliente Kiwi y la página web de PhantomSDR se midieron al mismo nivel, escuchando una señal local fuerte en 729 kHz. En la práctica se suele preferir un poco más — este receptor se quedó en 60 — así que elájalo de oído.

No puede estropear el audio con esto. Un **limitador de picos con anticipación** se sitúa entre la ganancia y el recorte de 16 bits: cada muestra se retrasa 4 ms mientras la ganancia se calcula a partir de audio que aún no se ha enviado, de modo que un pasaje fuerte llega a una ganancia que ya ha bajado para él, y los picos se pliegan en vez de aplanarse. Llevado a 60 e incluso a 70 dB — mucho más allá del margen que tiene la ganancia por sí sola — no se recortó ni una muestra de cada cuarto de millón. Por debajo del umbral del limitador la ganancia es exactamente 1,0, así que un receptor que no fuerza el audio recibe las muestras intactas.

Un valor demasiado alto cuesta volumen, no daño. Por encima de unos 60 dB la ganancia extra simplemente se limita: medido aquí, el nivel subió cerca de 1 dB de 55 a 60 y apenas se movió de 60 a 70. Más allá de ese punto está comprando compresión, no volumen.

Nada de esto toca la página web, y nada de esto mueve el S-meter — este se deriva de la potencia del demodulador, no de las muestras de audio, así que subir el volumen no puede hacer mentir al instrumento.

7. El S-meter

El S-meter del Kiwi no se calibra por separado: reproduce lo que muestra su página web, etapa por etapa. La potencia por bloque del demodulador es el punto de partida para ambos, y la página aplica después `input.analog_smeter_offset`, expande el resultado alrededor de -130 dBm y añade su propio desplazamiento de visualización. El puente hace lo mismo, de modo que un cliente Kiwi y la página muestran la misma señal con la misma intensidad — verificado en una emisora local fuerte, donde ambos marcaron -37 dBm.

Si, y solo si, quiere que los clientes Kiwi marquen distinto que su página, dele al puente un desplazamiento propio:

```
[kiwi_emulation]
enabled      = true
smeter_offset = 5.0      # sustituye a input.analog_smeter_offset solo para clientes Kiwi
```

Compare los dos instrumentos con el mismo filtro, o no los compare. La lectura procede de la potencia dentro del ancho de paso, así que un filtro más ancho recoge más ruido y marca más alto — en una banda tranquila la misma frecuencia midió 5,6 dB más fuerte con 9 kHz que con 2,4 kHz. Un cliente Kiwi con filtro de 6 kHz frente a una página web con 2,4 kHz discrepará en varios dB por muy calibrados que estén ambos. Iguale primero el modo y el ancho de banda.

Esta es la escala de la página, no una física. El instrumento web expande su rango por legibilidad, de modo que 10 dB de cambio real de señal se muestran como unos 11. Igualarla significa que los clientes Kiwi concuerdan con su receptor y difieren de un KiwiSDR auténtico en una cantidad que crece con la intensidad de la señal. Es el compromiso correcto cuando su propia página es la referencia con la que todos comparan; si prefiere que el puente se mantenga físicamente honesto, esta es la sección que hay que cambiar.

8. Cascada y espectro

La escala en dB — `wf_cal`

Cada bin de la cascada se convierte a dBm reales antes de enviarse, en la misma escala a la que está calibrado el S-meter. Por tanto una portadora marca **el mismo nivel en todos los niveles de zoom**, que es lo que da un KiwiSDR auténtico, y el espectro concuerda con el propio S-meter de la página web sobre el mismo ancho de paso.

```
[kiwi_emulation]
enabled = true
wf_cal = 0.0          # dB, solo clientes Kiwi
```

`wf_cal` no es una calibración. 0 es el valor con el que el espectro concuerda con su S-meter, y ahí es donde debe dejarlo si quiere que los números signifiquen algo. Existe porque lo *fuerte* que debe verse un espectro es cuestión de gusto y de lo que su cliente haga con el rango — el receptor de este proyecto funciona con -10 simplemente porque es lo que se ve bien en AetherSDR. Un solo valor mueve el espectro y la cascada a la vez: son los mismos bytes en la línea, y el protocolo Kiwi no puede escalarlos por separado. Si quiere que difieran, eso tiene que venir de los controles de visualización del propio cliente.

Para comprobar la escala, compare la cascada sumada sobre el ancho de paso con el S-meter en una frecuencia **tranquila**. Sobre una portadora, un ancho de paso SSB excluye la propia portadora y ambos no son comparables.

La tasa de cuadros — `wf_fps_max`

Un cliente Kiwi pide la cascada más rápida que puede tener (SET `wf_speed=4`) y acompasa su desplazamiento — y el promediado de su espectro — a la tasa que el servidor dice que va a entregar. Si las dos no coinciden, la pantalla parece lenta aunque en realidad nada llegue tarde.

El receptor produce $2 \times \text{sps} / \text{fft_size}$ espectros por segundo. Los clientes Kiwi se sirven de todos ellos y se adelgazan a la tasa que pidieron; la página web conserva su propia cadencia, más lenta, y no se ve afectada.

```
[kiwi_emulation]
enabled = true
wf_fps_max = 23      # cuadros por segundo
```

23 es el máximo que declara el propio protocolo KiwiSDR y el valor por omisión seguro. **Siempre gana el menor entre wf_fps_max y la tasa propia del receptor**, así que subirlo solo hace algo si su FFT es lo bastante rápida como para tener cuadros de sobra. Ejemplo, para un receptor a 60 Mmuestras/s con una FFT de 4194304 puntos — $2 \times 60000000 / 4194304 = 28,6$ cuadros por segundo:

wf_fps_max	entregado
ausente (por omisión 23)	23,0 fps
28	28,0 fps
40	28,6 fps — gana el techo propio del receptor

También se respeta SET wf_speed del cliente: 0 apagado, 1 = 1 fps, 2 un cuarto del máximo, 3 la mitad, 4 el máximo. Un cliente con un enlace estrecho puede así pedir menos.

Que subirlo ayude lo decide el cliente, no el servidor. Un cliente que dibuja cada cuadro según llega le da un desplazamiento más suave; un cliente que se acompasa a los 23 que espera se limitará a encolar los cuadros de más, y la latencia crecerá. Pruébalo, y si la cascada se siente más lenta en vez de más suave, vuelva a 23.

Qué determina el resto del retardo

Si va tras la latencia, la mayor parte no está en el puente. La ventana de análisis tiene `fft_size / sps` de ancho — 70 ms con una FFT de 4194304 puntos a 60 Mmuestras/s — y una muestra espera además hasta la mitad de eso a que se llene el salto. Reducir `fft_size` a la mitad reduce ambos a la mitad y duplica la tasa de cuadros, pero también reduce a la mitad su resolución en frecuencia, de la que viven los decodificadores de banda estrecha (WSPR, FT8, CW). En la mayoría de receptores es un mal intercambio por unas pocas decenas de milisegundos. Es una decisión en `[input]`, no algo que el puente pueda cambiar.

9. Si algo no funciona bien

Cada orden que envía un cliente Kiwi se escribe en `/tmp/kiwi_retune.log` — las peticiones de sintonía, los cambios de modo, la autenticación y todo lo que el puente no haya reconocido. No cuesta nada medible y está pensado para dejarlo ahí. Cuando un cliente se comporta de forma extraña, ese archivo muestra qué pidió realmente, que suele ser toda la respuesta.

Síntoma	Mire
El cliente conecta y cae de inmediato	falta <code>[kiwi_emulation] enabled = true</code> en la configuración con la que se arrancó el servidor
Sin audio, sin cascada, sin líneas de registro	el backend no se recompiló tras <code>kiwi_install.sh</code> — ejecute <code>./recompile.sh</code> , opción [1]
<code>kiwi_install.sh</code> se detiene en un parche	el mensaje nombra el archivo y el anclaje; el árbol ha divergido de lo que el parche espera, y ese archivo quedó intacto
Hay audio, pero delgado y bajo	es lo esperado — ponga <code>audio_gain</code> , véase la sección 6
La cascada se desplaza lenta	<code>wf_fps_max</code> , véase la sección 8
Sintonizar en el cliente no hace nada	una entrada IQ (<code>signal = "iq"</code>); la sintonía solo está implementada para entradas reales
Un ajuste cambiado no surtió efecto	estas claves se leen al arrancar — reinicie el receptor

El AGC es cosa del cliente. `SET agc=` desde un cliente Kiwi se acepta y se ignora — el audio llega con el AGC propio de PhantomSDR aplicado, y los controles de AGC del cliente actúan sobre lo que recibe.

10. Hasta dónde se ha verificado

El formato en la línea se ha comprobado contra un receptor en marcha y se ha leído línea por línea contra el código de los dos clientes a los que apunta — el `KiwiSdrProtocol.cpp` de AetherSDR y el `kiwi/client.py` de kiwiclient. La disposición y el tamaño de las tramas, los números de secuencia, el orden de bytes, la escala del S-meter, la sintonía y la cascada concuerdan todos. Además se ha **confirmado que el puente funciona con la propia aplicación de escritorio AetherSDR**, por su mantenedor, contra este receptor.

Medido en ese receptor, con un cliente de prueba sobre los sockets en marcha: la sintonía se aplica en 35–50 ms, el audio va en tiempo real sin deriva, y la cascada entrega la tasa configurada sin bloqueos ni cuadros perdidos.

Si aun así algo se ve mal en su propia instalación, `/tmp/kiwi_retune.log` es el primer sitio donde mirar, y un informe es bienvenido en cualquier caso.

Véase también: [Guía de instalación](#) · [Estructura del proyecto](#) · [Guía del usuario](#)

CAPÍTULO 14

Diversidad de recepción

RECEIVE_DIVERSITY.md

PhantomSDR-Plus — Diversidad de recepción

Combine su receptor con un **segundo receptor situado en otro lugar** y escuche al que en cada momento tenga la mejor señal. Cuando uno de los dos cae en un desvanecimiento, el otro normalmente no, y usted sigue copiando.

El segundo receptor puede ser otro PhantomSDR-Plus, un KiwiSDR, un UberSDR o un WebSDR. Con los tres primeros todo se ejecuta en el navegador: sin cambios en el servidor, sin ficheros de configuración, sin nada que instalar y sin acceso especial al otro receptor. El WebSDR es la excepción y necesita un pequeño programa en su propio servidor — véase [Usar un WebSDR](#) más abajo.

Qué hace y qué no

Aquí diversidad significa **selección**: en cada instante está escuchando a uno u otro emplazamiento, con un breve fundido al cambiar. No es una agrupación en fase y no suma las dos señales.

Es una decisión deliberada, y la razón es física, no de software. Dos receptores separados cientos de kilómetros oyen la misma emisión por caminos ionosféricos distintos. Las dos formas de onda llegan con fase no relacionada y un Doppler ligeramente distinto, así que sumarlas suena hueco y con filtrado de peine — el clásico efecto «phasey». La combinación coherente real exige dos receptores con un **reloj común**, alineados muestra a muestra; dos receptores independientes a través de internet nunca pueden ofrecer eso.

Lo que se gana, por tanto, es **continuidad, no intensidad de señal**:

- En un camino estable donde un receptor es simplemente mejor, oirá ese receptor y no ganará nada. Es lo normal.
- En un camino con QSB profundo, los desvanecimientos en dos emplazamientos lejanos están en buena medida incorrelados. El emplazamiento A desaparece tres segundos, el B no, y el audio continúa. Espere «*dejé de perder palabras*», no «*pasó de S3 a S7*».
- La segunda ventaja suele ser el **ruido local y el QRM**. Dos emplazamientos tienen vecinos distintos, redes eléctricas distintas y salpicaduras distintas. En una banda ruidosa esto puede pesar más que el desvanecimiento.

Tiene un coste: escucha **entre uno y tres segundos por detrás del tiempo real**, porque el flujo que va por delante debe retrasarse para casar con el otro. Irrelevante para escuchar y decodificar, molesto si intenta trabajar una estación.

NOTA

Los decodificadores digitales (FT8, JS8, WSPR, RADE y los demás) siguen usando deliberadamente su receptor **local**, no el audio combinado. Esos modos integran de forma coherente durante toda una ranura temporal, y cambiar de emplazamiento en mitad de la ranura es una discontinuidad de fase que puede costarle justo la decodificación que la diversidad pretendía salvar. La diversidad sirve al altavoz; los decodificadores conservan el flujo local continuo.

Puesta en marcha rápida

1. Abra el panel **Receive Diversity** — está justo debajo de las ventanas de decodificadores, plegado, mostrando `Receive Diversity – off`. Haga clic para desplegarlo.
2. Elija qué es el segundo receptor: **PhantomSDR+**, **KiwiSDR**, **UberSDR** o **WebSDR**.
3. Escriba su dirección — o elija una de las guardadas — y pulse **Start**.
4. Sintonice su propio receptor con normalidad. El segundo **le sigue automáticamente** — frecuencia, modo y ancho de paso — cada vez que resintoniza.

Esa es toda la configuración. No hay nada que ajustar en el otro receptor ni un mando de sintonía aparte: siempre le sigue.

Direcciones

Segundo receptor	Qué escribir	Notas
PhantomSDR+	<code>host:8900</code>	El puerto normal del receptor
KiwiSDR	<code>host:8073</code>	El puerto propio del KiwiSDR
UberSDR	<code>host</code>	Su puerto web normal , no el 8073
WebSDR	<code>host:8901</code>	El puerto propio de la estación. Necesita el relé — véase abajo

Se aceptan un nombre de host a secas, una dirección `http://` o una `ws://` completa. Si el otro receptor se sirve por HTTPS, escriba el nombre de host y se usará una conexión segura.

Receptores guardados

Cada dirección que utiliza queda memorizada, en una lista propia para cada tipo de receptor — una dirección de KiwiSDR nunca es una sugerencia útil cuando el selector dice WebSDR. Las listas viven en su propio navegador: no se envían a ninguna parte y cada oyente tiene la suya.

El botón ☰ junto al cuadro de dirección abre la lista.

- **Nombres.** Cualquier entrada puede recibir un nombre — `Twente` se lee mejor que `websdr.ewi.utwente.nl:8901`. Las entradas con nombre aparecen en verde sobre su dirección, y el nombre se muestra junto a la dirección en la lista desplegable mientras escribe. Una entrada se identifica por su dirección, así que ponerle nombre nunca hace que la misma estación figure dos veces. Mientras la diversidad está en marcha, el receptor con el que está enlazada muestra su nombre **en negrita y parpadeando lentamente**, de modo que la lista dice además cuál está escuchando realmente.
- **add** añade un receptor a mano, sin conectarse antes a él. 🖱️ corrige nombre y dirección en el sitio, ✕ elimina una entrada y **clear all** vacía la lista del tipo que está en pantalla. Enter guarda, Escape cancela.
- ******** abre la página web propia de ese receptor en una pestaña nueva. Lo guardado es la dirección que marca el software, así que primero se convierte en una dirección navegable — `ws://` pasa a `http://` y se recorta la ruta `/audio`, `/ws` o, en un KiwiSDR, `/kiwi/.../SND` que necesita cada tipo de receptor, ya que la página de la estación está en la raíz. Es un enlace de verdad, así que el clic central o la pulsación larga funcionan como siempre.
- **El orden es suyo.** Mueva una fila con ▲▼, o arrástrela y suéltela donde corresponda. Ese es el orden en que la lista desplegable ofrece las direcciones.
- **export e import.** Las listas están en el almacenamiento del propio navegador, que desaparece al borrar los datos del sitio — de eso protege esto. Export escribe las cuatro listas, con sus nombres, en un archivo JSON que puede conservar o llevar a otro navegador u otra máquina. Import lo lee de vuelta y pregunta si **fusionarlo** (merge) con lo guardado o **reemplazarlo** (replace) todo.

-  **QR.** Dibuja todas las listas como un código QR en la pantalla, para llevarlas a un teléfono — véase «En el teléfono» más abajo. Una lista demasiado grande para escanear recurre al texto que hay al lado, que puede copiarse y pegarse en cualquier parte.

No hay límite en cuántos receptores guarda.

En el teléfono

La página `/mobile` ofrece la misma función en una pestaña **Div**, junto a Audio, Bands, Marks, Users y Chat: tipo de receptor, dirección, Start y Stop, los receptores guardados como lista que se elige con un toque, las cifras en vivo y el ajuste de SNR. Cada fila lleva además el enlace `****` a la página propia de ese receptor. El segundo receptor sigue por sí solo la sintonía del teléfono, igual que en la página de escritorio. Lo que no está es la edición — renombrar, reordenar, borrar —, que se queda en la página de escritorio, donde hay sitio para ella.

La lista guardada pertenece a un navegador, así que un teléfono empieza con una vacía por muchos receptores que haya guardados en el escritorio. Para eso está el código QR: pulse  y luego  **QR** en el escritorio, escanee el código con la cámara del teléfono y pegue el texto en **Import** de la pestaña Div. Pregunta si fusionar o reemplazar, y Export en el teléfono envía una lista en sentido contrario.

Usar un WebSDR

Un WebSDR — el software de Pieter-Tjerk de Boer, PA3FWM, que funciona en Twente y en varios cientos de estaciones más — puede usarse como segundo receptor, pero no directamente desde el navegador.

El motivo es una comprobación deliberada por su parte. Un WebSDR rechaza la conexión de audio si la cabecera `Origin` no nombra su propio sitio, y `Origin` es un *nombre de cabecera prohibido*: el navegador la fija a partir de la página en la que está y ningún script puede cambiarla. No hay forma de sortearlo desde el cliente, ni debería haberla.

Por eso la conexión la realiza un pequeño programa en su propio servidor, `websdr_relay.py`. Su navegador habla con el relé, y el relé habla con el WebSDR.

Instalar el relé

Los cuatro instaladores de distribución lo ofrecen como paso opcional. Para una instalación ya existente:

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

Pide un puerto, toma su indicativo y la dirección de su estación de `frontend/site_information.json` y ofrece instalar un servicio systemd para que el relé arranque con el sistema. Los ajustes están en `websdr_relay.json` y pueden cambiarse en cualquier momento.

Hay algo que el instalador no puede hacer por usted: abrir el puerto del relé en su router. Los navegadores de los oyentes se conectan directamente al relé — no pasa por el receptor — así que sin esa redirección la diversidad con WebSDR solo funciona dentro de su propia red. Los otros tres tipos de receptor no se ven afectados.

Si el panel de administración está instalado, su dashboard muestra una tarjeta **WebSDR Diversity Relay** con el estado, el puerto y cuántas sesiones hay en curso y hacia qué estaciones.

Ser un buen invitado

El relé se conecta a receptores ajenos desde su servidor y no desde la dirección de cada oyente, por eso está hecho para comportarse:

- **Un límite de diez sesiones simultáneas a un mismo WebSDR**, y sesenta en total. Es lo que evita que un día movido en su estación parezca un ataque a la de otro.
- **Un User-Agent que nombra su estación y a su operador**, para que quien prefiera no ser usado así sepa exactamente a quién escribir. Rellénelo con honestidad.
- **Solo se reenvían órdenes de sintonía**. La conexión no puede usarse para enviar nada más.
- **Se rechazan las direcciones privadas, de loopback y de NAT de operador**, de modo que ningún visitante pueda apuntar el relé a algo dentro de su máquina o de su LAN.

IMPORTANTE

La comprobación de Origin existe porque los autores de WebSDR no querían que sus receptores se manejaran desde páginas ajenas. Usar el relé es una decisión meditada, no un descuido suyo. Deje el límite donde está, mantenga el User-Agent honesto y deténgase si un operador se lo pide.

Qué cambia con un WebSDR

- **Muchas estaciones cubren unas pocas franjas estrechas del espectro**, no un rango continuo — por ejemplo una ventana de 256 kHz en 40 m. La diversidad solo actúa dentro de ellas, y el panel lee la lista de bandas de la estación para saber dónde están.
- **La frecuencia de muestreo del audio varía** con el ancho del filtro y se aleja más del valor nominal que en los otros tipos, así que la alineación puede tardar algo más.
- **CW no se adapta** al convenio de WebSDR, que sitúa el paso de banda entero por debajo de la portadora. SSB y AM son correctos.

Cómo leer el panel

En funcionamiento, el panel muestra una pequeña tabla:

Fila	Significado
link	La conexión con el segundo receptor. <code>ready</code> es lo deseable. Ante un fallo aparece el texto de error del otro servidor y el código de cierre del WebSocket
aligned	El retardo medido entre los dos flujos una vez enganchados, o <code>searching</code>
corr	Con qué fuerza se correlan los dos flujos. Solo significativo tras la alineación
remote audio	Audio decodificado que llega del segundo receptor. <code>none</code> en ámbar significa que no se está decodificando nada
SNR local / remote	Las dos estimaciones de relación señal/ruido que se comparan
switches	Cuántas veces ha cambiado de emplazamiento

Junto al título, una palabra le dice en qué punto está:

off (gris)	no está en marcha
Please wait... (ámbar)	conectando, o conectado y todavía alineando
Ready (verde)	enlazado, enganchado y siguiendo al mejor emplazamiento
el error del propio receptor (rojo)	ha fallado y no se recuperará solo

La distinción entre ámbar y rojo es la que ahorra tiempo: ámbar significa siga esperando, rojo significa deje de esperar y lea el mensaje.

El punto de color añade a qué emplazamiento está escuchando realmente — verde su receptor, cian el remoto.

La alineación tarda unos 15 segundos

Es normal. Los dos flujos se alinean correlando sus **envolventes de audio**, y eso necesita una ventana de audio. La búsqueda se hace en dos etapas: una búsqueda estrecha de $\pm 1,5$ segundos puede empezar en cuanto se han reunido unos 9 segundos de sonido, y una segunda medición independiente, cinco segundos más tarde, debe **coincidir** con la primera antes de dar la alineación por buena. Cuando las dos estaciones están más separadas en el tiempo que eso — uno o dos segundos de almacenamiento adicional en algún punto del camino — toma el relevo la búsqueda completa de ± 4 segundos, a los 14 segundos, y la confirmación llega a los 19.

Ese paso de confirmación importa. Cuando dos emplazamientos se desvanecen en antifase — sin llevar nunca la señal a la vez —, una sola correlación engancha tan campante un retardo rotundamente equivocado, que suena a eco. Exigir dos mediciones independientes que coincidan lo descarta.

Una segunda cosa debe ser correcta antes de que un enganche se sostenga: las velocidades de muestreo de los dos receptores. Dos receptores son dos relojes y dos cadenas de diezmado, así que sus flujos de audio **no** llegan exactamente a la misma velocidad aunque ambos declaren 12 kHz — un 2% de diferencia es normal, es decir 240 muestras por segundo de deriva, muy por encima de la tolerancia de alineación. Por eso el flujo remoto se remuestrea continuamente para igualar la velocidad de su propio receptor. Esa relación se mide **contando muestras**, no correlando, así que no necesita enganche propio y queda establecida en los primeros segundos — antes incluso de que se ejecute la primera búsqueda de alineación. Una señal débil o intermitente puede necesitar dos o tres intentos, así que 30 segundos no son motivo de preocupación; un minuto sin nada sí lo es.

Mientras dice `searching`, ambas cifras de SNR marcan 0.0. Es intencionado: las dos estimaciones se miden sobre muestras **alineadas en contenido**, así que no se mide nada hasta que hay alineación. 0.0 / 0.0 significa «aún no enganchado», no «sin señal».

Remote SNR trim

Este deslizador sesga la elección entre los dos emplazamientos. Se suma al SNR medido del receptor remoto antes de la comparación — positivo favorece al remoto, negativo al suyo. No cambia **nada más**: ni el nivel de audio ni la combinación, solo qué emplazamiento gana.

Existe porque las dos cifras de SNR no siempre son comparables. Ambas se miden igual — como recorrido entre percentiles del audio —, pero un receptor cuyo códec eleva su propio suelo de ruido marca menos de lo que merece. Un KiwiSDR aplica mucha ganancia y un limitador, lo que comprime el recorrido; un códec con pérdidas fija un suelo de ruido por debajo del cual la señal nunca baja.

Cómo ajustarlo:

1. Sintonice ambos receptores a una señal presente de forma fiable y espere a `aligned`.

2. Observe las dos cifras de SNR durante medio minuto. Anote la diferencia habitual.
3. Use los extremos como prueba de escucha: **+15** fuerza el emplazamiento remoto, **-15** el suyo. Escuche unos segundos cada uno. Es la única forma de oír un emplazamiento por separado.
4. Si el remoto marca, digamos, 5 dB menos pero suena igual de bien, ponga **+5**. Está haciendo que las dos lecturas coincidan cuando los dos emplazamientos suenan igual.
5. Vigile el contador **switches** durante diez minutos de escucha. Un conmutar constante indica que el valor está demasiado cerca del empate; no conmutar nunca aunque su receptor se desvanezca de forma audible indica que se ha pasado en el otro sentido.

Puntos de partida: **0** para otro PhantomSDR-Plus, que usa una cadena de audio idéntica y es comparable por construcción; un valor **positivo** pequeño para un KiwiSDR o un UberSDR.

CONSEJO

Si necesita más de ± 8 dB aproximadamente, deje de ajustar. A esas alturas la explicación honesta suele ser que un receptor realmente es peor para ese camino, y sesgar más allá de una diferencia real solo significa escuchar al emplazamiento más flojo.

El ajuste se aplica al momento y se recuerda en su navegador.

Elegir un segundo receptor

La distancia importa. Los desvanecimientos se descorrelan con la distancia — unos cientos de kilómetros en HF es un buen objetivo. Dos receptores en la misma ciudad se desvanecen a la vez y no aportan nada. Demasiado lejos, y el segundo receptor puede no oír su señal en absoluto.

Ambos deben oír realmente la señal. Este requisito no lo sortea ningún software. Si el segundo receptor no oye lo que usted escucha, la correlación se queda baja y no enganchará — a propósito, porque una alineación equivocada suena peor que no tener diversidad.

Cobertura. Un segundo PhantomSDR-Plus o un KiwiSDR anuncian el rango de frecuencias que cubren, y el panel le avisa si sintoniza fuera de él. Un WebSDR anuncia sus bandas, que a menudo son franjas estrechas en lugar de un rango continuo. UberSDR no anuncia cobertura, así que allí el no enganchar es su única señal.

IMPORTANTE

Los receptores públicos los mantienen voluntarios y tienen un número limitado de plazas de oyente. Una sesión de diversidad ocupa una mientras dure, exactamente igual que un oyente humano. Sea considerado antes de dejar una conexión indefinidamente, y pregunte al operador si piensa usar su receptor de forma intensiva.

Resolución de problemas

Link no sale de connecting, o alterna conectar/cerrar Probablemente la dirección o el puerto son incorrectos. Consulte la tabla de arriba — en particular, UberSDR usa su puerto web normal, no el 8073. El panel muestra el código de cierre del WebSocket y la consola del navegador (F12) el error subyacente.

Link muestra un error con texto detrás Ese texto viene del otro receptor y suele ser concreto: sesión rechazada, servidor lleno o un receptor que no admite clientes externos.

El KiwiSDR rechaza la conexión Algunos KiwiSDR piden contraseña, y en un UberSDR el punto de acceso compatible con KiwiSDR está desactivado de fábrica. Para un UberSDR use el tipo **UberSDR** — esa vía está siempre disponible.

WebSDR: the WebSDR relay is not reachable El relé no está en marcha, o su navegador no alcanza su puerto. Debe ser accesible en el **mismo nombre de host desde el que se sirve la página del receptor**, porque el navegador se conecta directamente a él y no a través del receptor. Abra `http://<ese nombre de host>:<puerto del relé>/status` en el mismo navegador para ver cuál de los dos casos es. Si responde en su LAN pero no desde fuera, el puerto no está redirigido en el router.

WebSDR: un error que menciona una conexión rechazada Esa estación está más restringida que la comprobación habitual de `Origin`, o su operador ha bloqueado esta estación.

Link está en ready pero remote audio marca none No se está decodificando audio. Recargue la página; si persiste, la consola del navegador mostrará qué decodificador falló.

Llega audio pero nunca se alinea Los dos receptores no están oyendo lo mismo. Pruebe una emisora de radiodifusión potente, o una banda donde ambos emplazamientos tengan buena propagación. Una banda tranquila con solo ruido en ambos extremos nunca correlará — y no debe hacerlo.

Conmuta sin parar Los dos emplazamientos están demasiado igualados en SNR. Ajuste ligeramente el trim para que uno quede preferido de forma estable.

Nunca usa el segundo receptor Compruebe que `remote audio` va subiendo y que las cifras de SNR tienen sentido. Si el remoto marca mucho menos de lo que suena, para eso está el trim.

Limitaciones

- **No es coherente.** Sin phasing, sin cancelación, sin radiogoniometría. Cancelar una fuente de ruido exige dos antenas con un reloj común en un mismo emplazamiento; esa es otra técnica para otro problema.
- **De uno a tres segundos de retardo**, inevitablemente.
- **Solo dos receptores.**
- **Mono.** Si su receptor está en un modo estéreo como C-QUAM, el audio pasa intacto y la diversidad no entra en juego.
- **La alineación necesita señal.** Por debajo de unos 10 dB de SNR en uno de los dos emplazamientos, espere que se quede en `searching`.
- **Un WebSDR necesita el relé** en su propio servidor, con su puerto redirigido. Los otros tres tipos de receptor no necesitan ninguna de las dos cosas.

CAPÍTULO 15

Control del transceptor

RIG_CONTROL.md

PhantomSDR-Plus — Control del transceptor (CAT)

Mantenga **su propio transceptor** y un **receptor PhantomSDR-Plus** en la misma frecuencia, modo y filtro. Gire el dial del equipo y la cascada lo sigue; haga clic en una señal de la cascada y el equipo se sintoniza en ella. Al transmitir, el receptor puede quedar en silencio para no devolverle su propia señal.

Funciona con receptores **PhantomSDR-Plus**, **KiwiSDR**, **PA3FWM WebSDR** y **UberSDR**, suyos o ajenos, y solo mueve *su* sesión de escucha: nadie más en el receptor oye ni ve nada. El operador del receptor no tiene que instalar ni configurar nada.

Qué necesita

Dos formas de conectar un equipo y un requisito en el receptor:

Pieza	Qué es	Sincroniza
Desktop PhantomSDR+ 4.0 o posterior	La aplicación de escritorio, con un menú Rig . Linux (PC y Raspberry Pi) y Windows.	Frecuencia, modo, ancho de filtro, silencio al transmitir — en un sentido o en ambos
CATsync Tool for WebSDRs	Un programa aparte para Windows que acopla un equipo a la página del receptor en su navegador.	Frecuencia y modo
TCI-CAT en la página del receptor	Un botón en la propia página de PhantomSDR-Plus, en cualquier navegador. Habla TCI con ExpertSDR, AetherSDR o Thetis, o con cualquier equipo compatible con Hamlib a través de un pequeño puente. Solo en receptores PhantomSDR-Plus con la versión 4.1.0 o posterior.	Frecuencia, modo y ancho de filtro en ambos sentidos; silencio al transmitir
El receptor	PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR o UberSDR. Un receptor PhantomSDR-Plus necesita la 4.1.0 o posterior para ancho de filtro y silencio.	Un PhantomSDR-Plus más antiguo sigue sincronizando frecuencia y modo

La mayor parte de este manual describe Desktop PhantomSDR+. El botón TCI-CAT tiene su propia sección, [TCI-CAT en la página del receptor](#). La CATsync Tool tiene su propia documentación en su sitio web.

Receptores compatibles

La aplicación reconoce el tipo de receptor de una ventana de estación y lo maneja mediante los propios controles de esa página. La ventana Rig control muestra el tipo encontrado junto al nombre de la estación.

Receptor	Frecuencia y modo	Ancho de filtro	Silencio al transmitir
PhantomSDR-Plus	Sí	Con la 4.1.0 o posterior	Con la 4.1.0 o posterior
KiwiSDR (incluido Web-888)	Sí	Sí	Sí
PA3FWM WebSDR	Sí, con cambio de banda en sitios multibanda	Sí	Sí
UberSDR	Sí	Sí	Sí

Un receptor web tiene menos modos que la mayoría de equipos, así que algunos modos del equipo comparten un modo del receptor: KiwiSDR y WebSDR tienen un único CW para CW y CW-R. Los modos de receptor que nombra este manual son los de PhantomSDR-Plus; los demás receptores usan el equivalente más cercano. Un WebSDR mantiene un filtro CW por debajo de 1 kHz y los demás filtros en 1 kHz o más, porque así distingue la propia página el CW, y una frecuencia fuera de todas las bandas de un sitio WebSDR se deja en paz. UberSDR ajusta su filtro en pasos del deslizador, así que un ancho puede quedar hasta unos 50 Hz del del equipo. Cualquier otro tipo de página — OpenWebRX, por ejemplo — muestra *not a receiver this app can drive* y no se sincroniza nada.

Qué hace y qué no hace

- Sincroniza **un equipo con una ventana de receptor** a la vez.
- Lee ambos lados varias veces por segundo y, cuando no coinciden, ajusta uno para que coincida con el otro. **No** transmite, no activa el PTT ni envía audio a ninguna parte.
- Mover el receptor solo cambia su propia sesión. Los demás oyentes del mismo receptor no se ven afectados y el operador no tiene que permitirlo.
- Un puerto serie solo puede abrirlo **un programa a la vez**. Si WSJT-X, un programa de registro o una utilidad del fabricante ya tiene el puerto, use la opción **flrig** o **rigctld on network** para compartir el equipo en lugar de pelear por el puerto.

Inicio rápido

1. Abra una estación en Desktop PhantomSDR+ como de costumbre.
2. **Rig → Rig control...**
3. En **Connection**, elija **Built-in** si su equipo está en la lista; si no, **Hamlib (all rigs)**.
4. Elija su equipo, el puerto serie y la velocidad a la que está configurado el menú CAT o CI-V del equipo.
5. En **Sync**, deje seleccionado **Both directions**.
6. Pulse **Connect**. Las dos lecturas de arriba — transceptor y receptor — deberían mostrar la misma frecuencia en un segundo.

Cada ajuste se guarda al cambiarlo. La próxima vez basta con **Rig → Connect**, o marque **Connect when the app starts**.

Cómo llegar al equipo

Opción	Úsela cuando	Necesita
Built-in	Su equipo está en la lista de abajo.	Nada más
Hamlib (all rigs)	Su equipo es cualquier otro — Hamlib conoce más de 300. La aplicación inicia el <code>rigctld</code> de Hamlib por usted, en un puerto local privado, y lo detiene al desconectar.	Windows: nada, Hamlib va incluido. Linux: <code>sudo apt install libhamlib-utils</code>
rigctld on network	Ya hay un <code>rigctld</code> en marcha, en este ordenador u otro de su red.	Host y puerto (4532 por defecto)
flrig	flrig ya controla el equipo para fldigi, WSJT-X o un programa de registro.	flrig en marcha, con su puerto XML-RPC (12345 por defecto)

Equipos con controlador integrado

La velocidad y la dirección CI-V que aparecen son los valores de fábrica que rellena la aplicación. **Son solo un punto de partida: póngalos como indique el menú de su equipo.**

Familia	Equipos	Velocidad por defecto	Notas
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 y cualquier otro equipo CI-V	19200	Dirección CI-V por modelo (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	Dirección 70; compruébela en el menú
Yaesu CAT nuevo	FTDX101D/MP, FTDX10, FT-710, FT-991/A, FT-891, FTDX5000, FTDX3000, FTDX1200, FT-950, FT-2000, FT-450/450D	38400	
Yaesu CAT clásico	FT-817/818, FT-857/857D, FT-897/897D	38400	2 bits de parada; sintoniza en pasos de 10 Hz
Kenwood	TS-990S, TS-890S, TS-590S/SG	115200	
	TS-480, TS-2000, TS-870S	57600	
Elecraft	K4, K3/K3S, KX3, KX2	38400	Sincroniza el ancho de filtro
Compatibles Kenwood	FlexRadio SmartSDR CAT (puerto virtual), QRP Labs QMX/QMX+/QDX, (tr)uSDX, Lab599 Discovery TX-500, otros compatibles Kenwood	9600-38400	

Un equipo que debería ser compatible pero no se entiende con un controlador integrado suele funcionar con **Hamlib**, que tolera muchas más variantes.

Ajustes del puerto serie

Ajuste	Qué poner
Serial port	El puerto del equipo. Los adaptadores USB y los equipos con puerto USB aparecen primero. Other / network address... acepta un puerto que no esté en la lista — COM7, /dev/ttyUSB1 — o tcp://host:port para un puerto serie servido por la red con ser2net o similar.
Speed (baud)	Exactamente lo que diga el menú de velocidad CAT / CI-V del equipo. Una velocidad errónea parece un equipo que nunca responde.
Stop bits	1 para casi todo; 2 para la familia FT-817/857/897.
CI-V address	Solo Icom, en hexadecimal (94, no 148). Debe coincidir con el menú de dirección CI-V.
DTR / RTS	Déjelos desactivados salvo que su interfaz los necesite. Muchos cables CAT activan el transmisor, o reinician el equipo, con una de esas líneas.
Hardware flow control	Desactivado, salvo que el manual pida RTS/CTS.

Con **Hamlib** los mismos ajustes pasan a `rigctld`. Allí los bits de parada tienen además *Rig default*, y **Extra rigctld options** acepta cualquier otra opción de `rigctld`, por ejemplo `--set-conf=post_write_delay=10`. **rigctld program** permite apuntar a un `rigctld` concreto si tiene más de uno instalado.

Sincronización

Sentido

Opción	Qué ocurre
Rig → receiver	La ventana del receptor sigue al equipo. Un cambio hecho en la cascada vuelve a la frecuencia del equipo.
Receiver → rig	El equipo sigue a la ventana del receptor. Girar el dial del equipo se deshace.
Both directions	Gana el lado que haya tocado en último lugar . Al conectar, antes de tocar ninguno, gana el equipo.

El sentido también se cambia desde el menú **Rig** con la conexión activa.

Cómo se evita que los dos lados se peleen

Cada valor que escribe la aplicación aparece un momento después como un cambio en el otro lado. Si se tomara al pie de la letra, el equipo y el receptor se perseguirían sin fin. La aplicación lo evita de tres formas:

- La página del receptor aplica un cambio al instante, así que se vuelve a leer justo después de escribir y esa lectura pasa a ser el nuevo punto de partida.
- Un equipo aplica un cambio un poco después, así que se recuerda cada valor enviado. Cuando el equipo informa de ese valor, se reconoce como escritura de la aplicación y no como una mano en el dial.
- Un valor que el equipo rechaza — FM ancha en un equipo de HF, por ejemplo — se envía **dos veces** y después se deja en paz hasta que cambie el lado de origen, en lugar de repetirse varias veces por segundo.

Resintonizar la página del receptor puede hacer que elija el modo por defecto de la banda (LSB por debajo de 10 MHz, por ejemplo). Cuando manda el equipo, la aplicación repone enseguida el modo del equipo, así que un equipo en USB en 40 m mantiene el receptor en USB.

Qué ventana de receptor

Receiver window elige qué estación sigue al equipo:

- **The station window last in front** (por defecto) — con dos estaciones abiertas, haga clic en una y el equipo la sigue.
- **Una estación concreta** — el equipo queda fijado a ella, esté delante o no. Si esa estación no está abierta, no se sincroniza nada hasta que lo esté.

Frecuencia de actualización

Update every fija cada cuánto se leen ambos lados: 150 ms, 300 ms (por defecto), 500 ms o 1 s. Más rápido se nota más inmediato en el dial; más lento es más amable con un equipo antiguo a 4800 o 9600 baudios, donde cada lectura tarda tiempo real en la línea.

Modos

Modo del equipo	El receptor escucha en
USB, LSB	USB, LSB
CW	CW
CW-R (invertido)	CW-L
AM, AM síncrona, DSB	AM
FM, FM estrecha	FM
FM ancha	WBFM
RTTY / FSK	LSB
RTTY-R / FSK-R	USB
Modos de datos (USB-D, DATA-U, PKTUSB, DIG)	USB
Datos LSB, datos FM	LSB, FM

Modo del receptor	El equipo se pone en
USB, LSB	USB, LSB
CW	CW
CW-L	CW-R
AM, QUAM	AM
FM	FM
WBFM	WFM — la mayoría de equipos de HF lo rechazan y se les deja en paz tras dos intentos
RADE (superior / inferior)	USB / LSB

Un equipo en **modo de datos** se queda en él: el USB del receptor se considera coincidente con el USB-D del equipo, así que el receptor nunca saca al equipo del modo de datos.

Ancho de filtro

Marque **Sync filter width** para que coincidan las bandas de paso. Diferencias de menos de 60 Hz se consideran iguales, porque no hay dos filtros con los mismos pasos.

Conexión del equipo	Ancho de filtro
Hamlib	Sí, donde Hamlib lo admita para ese equipo
flrig	Sí
Icom CI-V integrado	Sí — pasos de 50 Hz hasta 500 Hz, luego de 100 Hz hasta 3,6 kHz; AM en pasos de 200 Hz hasta 10 kHz; no en FM
Elecraft integrado	Sí, en pasos de 10 Hz
Kenwood, Yaesu y familia FT-817 integrados	No — estos equipos eligen filtros de tablas propias de cada modelo. Use Hamlib si necesita el filtro

Los receptores KiwiSDR, WebSDR y UberSDR tienen siempre el control. Un receptor PhantomSDR-Plus debe tener la **4.1.0** o posterior; en uno más antiguo la frecuencia y el modo siguen sincronizándose, y la ventana Rig control explica por qué el filtro no.

Silencio al transmitir

Marque **Mute receiver while transmitting**. Mientras el equipo transmite, la ventana del receptor queda en silencio, y al dejar de transmitir vuelve el sonido. El botón de silencio del receptor lo muestra y puede quitar el silencio a mano.

Si ya había silenciado el receptor usted mismo, sigue silenciado después.

Necesita una conexión que informe del estado de transmisión — todos los controladores integrados, flrig y Hamlib en la mayoría de equipos — y, en un receptor PhantomSDR-Plus, la 4.1.0 o posterior.

Desplazamiento de frecuencia

Frequency offset se suma a la frecuencia del equipo para obtener la del receptor:

frecuencia del receptor = frecuencia del equipo + desplazamiento

Instalación	Desplazamiento
Transverter de 2 m con equipo de 10 m (144,100 MHz aparece como 28,100 MHz)	116000000
Transverter de 70 cm con equipo de 2 m (432 → 144)	288000000
Sin transverter	0

El menú Rig

Elemento	Qué hace
Rig control...	Abre la ventana Rig control
Connect / Disconnect <i>nombre del equipo</i>	Inicia o detiene la sincronización; con Hamlib también inicia o detiene <code>rigctl</code>
Rig to receiver / Receiver to rig / Both directions	Sentido de la sincronización
Sync filter width	Sí / no
Mute receiver while transmitting	Sí / no
Línea de estado	<i>Not connected, Connecting..., Connected: nombre</i> , o el último error

La lectura de frecuencia en directo está en la ventana Rig control y no en el menú, que de lo contrario se cerraría solo con cada cambio.

Linux

Permiso del puerto serie. Los puertos serie pertenecen al grupo `dialout`. Un usuario fuera de él obtiene *Could not open ttyUSB0*. Añádase una vez y luego cierre sesión y vuelva a entrar:

```
sudo usermod -aG dialout $USER
```

Hamlib. Instálelo desde su distribución:

```
sudo apt install libhamlib-utils
```

El paquete `.deb` de Desktop PhantomSDR+ lo recomienda, así que `sudo apt install ./phantomsdr-plus-desktop_4.0.0_amd64.deb` lo trae consigo; `dpkg -i` no instala paquetes recomendados. Los controladores integrados y flrig no necesitan Hamlib.

Windows

El propio `rigctl.exe` de Hamlib va incluido en el instalador de 64 bits y en el de 32 bits. Los puertos COM aparecen en la lista por su nombre (COM3). Si el equipo necesita un controlador USB, instale primero el del fabricante: hasta entonces el puerto no existe.

TCI-CAT en la página del receptor

Una página de receptor PhantomSDR-Plus también puede seguir a un transceptor **por sí misma**, sin la aplicación de escritorio: el navegador habla directamente con un servidor **TCI** en su propio ordenador. TCI es el protocolo de control por WebSocket de ExpertSDR2/ExpertSDR3 (SunSDR), AetherSDR (FlexRadio) y Thetis (Apache Labs ANAN, Hermes). Para un equipo sin TCI, un pequeño puente incluido en PhantomSDR-Plus hace que cualquier equipo de Hamlib parezca un servidor TCI.

Necesita un receptor PhantomSDR-Plus con la versión **4.1.0 o posterior**. Las páginas KiwiSDR, WebSDR y UberSDR no lo tienen.

El botón TCI-CAT

El botón está en la fila de **VFO, Modes, Bands e IF Filters** — **TCI-CAT** en el diseño ancho, **CAT** en el compacto. Su punto está **verde** mientras hay un servidor TCI conectado y **gris** en otro caso. Abre una ventana con: Hasta la 4.2.0 el botón se llamaba **QRG Sync**.

Control	Qué hace
Estado	<i>Active</i> — TCI (port 50001) cuando está conectado; <i>Inactive</i> con <i>Wait</i> mientras busca
CAT Sync	Activa o desactiva la sincronización de frecuencia, modo y filtro. El silencio al transmitir funciona en ambos casos
Host	El ordenador con el servidor TCI: <code>localhost</code> para este ordenador, o su dirección en la red local, p. ej. <code>192.168.1.42</code> . Pulse Intro o haga clic fuera para reconectar

No hay nada más que elegir: los puertos **50001** (ExpertSDR3, AetherSDR) y **40001** (ExpertSDR2, Thetis) se prueban a la vez y se reintentan cada pocos segundos, así que la página se conecta en cuanto arranca el programa. El navegador recuerda **CAT Sync** y **Host**.

Los navegadores que piden permiso antes de que una página web acceda a su red local — las versiones recientes de Chrome y Edge lo hacen — preguntan una vez, la primera vez que la página se conecta. Permítalo, o el punto seguirá gris.

Qué se mantiene sincronizado

	Comportamiento
Equipo → receptor: frecuencia	Solo se mueve el dial. Zoom, brillo y contraste quedan como los dejó; la cascada solo se desplaza cuando la frecuencia sale de la pantalla, y el plan de bandas no cambia el modo
Equipo → receptor: modo	Sigue cuando cambia el modo del equipo — USB, LSB, CW, AM, FM; los modos de datos como USB o LSB. Un modo que el receptor no tiene lo deja como está. Un decodificador en marcha conserva su propio modo
Equipo → receptor: ancho de filtro	La banda de paso del receptor toma el ancho de filtro del equipo: cambiar el filtro del equipo, o pulsar su botón FIL , cambia la banda de paso. Un decodificador en marcha conserva su propia banda de paso
Receptor → equipo: frecuencia	Escribir una frecuencia, una etiqueta o marcador, un clic en la cascada y arrastrar la banda de paso mueven el VFO del equipo. Al arrastrar solo se envían las frecuencias que cambian
Receptor → equipo: modo	Cada cambio de modo en la página — un botón de modo, el plan de bandas, un decodificador, un marcador — fija el modo del equipo (USB, LSB, CW, AM, FM; RADE como USB o LSB). Un equipo en modo de datos (USB-D) sigue en él
Receptor → equipo: ancho de filtro	Los botones de IF Filters , el deslizador IF, arrastrar la banda de paso y un cambio de modo fijan el filtro del equipo; al arrastrar solo se envía el ancho final. En los equipos Icom el puente selecciona en su lugar FIL1, FIL2 o FIL3 — ver Filtros Icom
Sin eco	Un cambio que vino del equipo nunca se le devuelve, y se ignora un aviso del modo o filtro anterior del equipo que llegue justo después de un cambio en la página
Silencio al transmitir	Siempre activo, incluso con CAT Sync apagado: la página se silencia mientras el equipo transmite. Mover el deslizador de volumen durante la transmisión la mantiene en silencio; al recibir, el sonido vuelve con la posición actual del deslizador
No se gestiona	Split, VFO B, RIT/XIT y desplazamientos de transverter. Solo se sigue el VFO A del primer receptor

El modo y el ancho de filtro viajan con las órdenes propias de TCI (`modulation`, `rx_filter_band`), así que ExpertSDR, AetherSDR y Thetis también deberían seguirlos; solo se ha probado el puente Hamlib. Para un desplazamiento de transverter, use Desktop PhantomSDR+. No use ambos a la vez con el mismo equipo — dos controladores se pelean por el dial.

Con qué transceptores funciona

Transceptor	Funciona	Cómo
SunSDR (ExpertSDR2/3), FlexRadio (AetherSDR), ANAN/Hermes (Thetis)	Sí	Directamente — active el servidor TCI en el programa. Aún no probado con estos programas; probado con un servidor TCI simulado
Un equipo con puerto CAT compatible con Hamlib — la mayoría de Icom, Yaesu, Kenwood, Elecraft, Xiegu, QRP Labs	Sí	Mediante el puente Hamlib descrito abajo. Probado con un Icom IC-7300 en Linux y Windows; otros controladores pueden diferir en los nombres de modo o en informar del PTT
Un equipo sin puerto CAT, o no compatible con Hamlib	No	No hay de dónde leer la frecuencia

El puente Hamlib (IC-7300 y otros equipos sin TCI)

`tci-bridge/tci-rigctld.mjs`, en el árbol de PhantomSDR-Plus, convierte cualquier equipo que Hamlib pueda controlar en un servidor TCI para la página. Se ejecuta en **su** ordenador — el conectado al equipo, junto al navegador — no en el receptor. Cuatro veces por segundo pregunta a Hamlib la frecuencia, el modo, el ancho de filtro y el estado de transmisión, envía a la página solo lo que cambió y pasa al equipo los cambios de frecuencia, modo y filtro de la página.

Llega a Hamlib por una de dos vías:

Vía	Cadena	Cuándo
<code>--rigctl</code> (recomendada)	transceptor → <code>rigctl</code> → <code>tci-rigctld.mjs</code> → página	Normalmente. El puente arranca por sí mismo el <code>rigctl</code> de Hamlib: una ventana, ningún puerto de red. Úsela en Windows , donde <code>rigctld.exe</code> suele rechazarse con <i>Access is denied</i>
<code>rigctld</code>	transceptor → <code>rigctld</code> → <code>tci-rigctld.mjs</code> → página	Otro programa, como WSJT-X, debe usar el equipo al mismo tiempo — ver Compartir el equipo

Qué necesita

	Linux	Windows
Hamlib	<code>sudo apt install libhamlib-utils</code> , o el paquete hamlib de su distribución	<code>hamlib-w64-....zip</code> de github.com/Hamlib/Hamlib/releases , extraído en <code>C:\hamlib</code> — <code>rigctl.exe</code> está en <code>C:\hamlib\bin</code>
Node.js	18 o posterior	El <i>Windows Installer (.msi)</i> LTS de nodejs.org , con las opciones por defecto
El paquete <code>ws</code>	Se encuentra automáticamente dentro del árbol PhantomSDR-Plus	Copie <code>tci-rigctld.mjs</code> a una carpeta como <code>C:\tci-bridge</code> y ejecute allí una vez <code>npm install ws</code>
El puerto del equipo	<code>ls /dev/serial/by-id/</code> . Únase una vez al grupo <code>dialout</code> : <code>sudo usermod -aG dialout \$USER</code> , luego cierre la sesión y vuelva a entrar	Instale el controlador USB del fabricante; el número COM está en Administrador de dispositivos → Puertos (COM y LPT)

Solo un programa puede tener el puerto CAT del equipo. Cierre WSJT-X, flrig, JS8Call, RS-BB1 o la conexión de equipo de Desktop PhantomSDR+ antes de arrancar el puente.

Ejemplo: Icom IC-7300

En el equipo: **MENU → SET → Connectors → CI-V — CI-V USB Baud Rate 115200, CI-V Transceive ON**. El número de modelo de Hamlib para el IC-7300 es 3073.

Linux. El equipo es la línea que contiene IC-7300 en `ls /dev/serial/by-id/`, normalmente también `/dev/ttyUSB0`.

1. Compruebe que Hamlib llega al equipo — debe mostrar la frecuencia, p. ej. 14280000: `bash rigctl -m 3073 -r /dev/ttyUSB0 -s 115200 f`
2. Arranque el puente y déjelo en marcha: `bash cd ~/PhantomSDR-Plus/tci-bridge node tci-rigctld.mjs --rigctl rigctl -m 3073 -r /dev/ttyUSB0 -s 115200`

Windows. El equipo aparece en el Administrador de dispositivos como *Silicon Labs CP210x USB to UART Bridge (COM4)* — use su propio número COM. En un Símbolo del sistema:

1. Compruebe que Hamlib llega al equipo — debe mostrar la frecuencia: `bat C:\hamlib\bin\rigctl.exe -m 3073 -r COM4 -s 115200 f`
2. Arranque el puente y deje la ventana abierta: `bat cd C:\tci-bridge node tci-rigctld.mjs --rigctl C:\hamlib\bin\rigctl.exe -m 3073 -r COM4 -s 115200` La línea es larga: asegúrese de que termina de verdad en `-s 115200`, o `rigctl` se detiene con *Type: rigctl --help*.

En ambos sistemas el puente muestra, en un segundo:

```
rig answered through rigctl
rig -> page 14280000 Hz
rig -> page mode USB
rig -> page RX
```

Después abra la página del receptor en un navegador del mismo ordenador, abra **TCI-CAT** y active **CAT Sync**. El punto se pone verde y el puente muestra `page connected`. Gire el VFO y el receptor lo sigue; haga clic en la cascada y el equipo lo sigue (`page -> rig ... Hz`); transmita y la página se silencia. **Ctrl+C** detiene el puente, y con él `rigctl`.

Todo lo que va tras `--rigctl` es el programa `rigctl` y sus propias opciones — exactamente las que funcionaron en la comprobación. Es la regla para cualquier equipo: **si `rigctl ... f` muestra la frecuencia, el puente funciona con las mismas opciones.**

Filtros Icom (FIL1, FIL2, FIL3)

Los equipos Icom como el IC-7300 no aceptan cualquier ancho de filtro: tienen tres filtros, **FIL1**, **FIL2** y **FIL3**, cada uno con un ancho fijado en el menú del equipo. Con un ancho, Hamlib seleccionaría uno de ellos y además sobrescribiría su ancho — y en el IC-7300 ese ancho puede ir a parar al filtro que estaba seleccionado antes, desordenando los ajustes. Por eso, en los equipos Icom el puente nunca envía un ancho: toma el filtro cuyo ancho de referencia está más cerca de la banda de paso de la página y solo lo **selecciona**, con la orden CI-V del propio equipo. Los anchos fijados en el equipo nunca cambian.

Modo	FIL1	FIL2	FIL3
USB, LSB (y sus modos de datos)	2700 Hz	2400 Hz	1800 Hz
CW	1200 Hz	500 Hz	250 Hz
AM	9000 Hz	6000 Hz	3000 Hz
FM	15000 Hz	10000 Hz	7000 Hz

En el IC-7300 (-m 3073) es automático. Ajuste los filtros SSB del equipo en consonancia — FIL1 2,7 kHz, FIL2 2,4 kHz, FIL3 1,8 kHz (mantenga pulsado **FIL** en el equipo) — y elegir 2,7 / 2,4 / 1,8 kHz en **IF Filters** selecciona FIL1 / FIL2 / FIL3, mientras que pulsar **FIL** en el equipo pone la banda de paso de la página al ancho de ese filtro. El puente muestra, por ejemplo, `page -> rig filter 2398 Hz LSB → FIL2 sent`.

Opción	Uso
<code>--filters 3000,2400,1800</code>	Otros anchos de referencia SSB, en el orden FIL1,FIL2,FIL3; también activa la selección de filtro para otro equipo Icom
<code>--civ A4</code>	La dirección CI-V del equipo en hexadecimal, cuando no es 94 (IC-705 A4, IC-9700 A2, IC-7610 98)
<code>--filters off</code>	Enviar anchos mediante Hamlib, como para otras marcas

Ambas opciones van antes de `--rigctl`, p. ej. `node tci-rigctld.mjs --filters 3000,2400,1800 --civ A4 --rigctl rigctl -m 3085 -r /dev/ttyACM0 -s 19200`. Probado solo con un IC-7300. Las demás marcas — Yaesu, Kenwood, Elecraft y el resto — reciben el ancho de la página mediante Hamlib, que lo fija tan cerca como el equipo permite.

Otros equipos

Los mismos pasos sirven para cualquier equipo compatible con Hamlib; solo cambian las opciones.

- Prepare el equipo.** En su menú, anote la velocidad CAT (o CI-V) y active un ajuste como *CAT por USB* o *CI-V transceive* si lo tiene. En Windows, instale el controlador USB del fabricante.
- Busque el número de modelo de Hamlib** en la lista que muestra `rigctl -l`:
- Linux: `rigctl -l | grep -i 991`
- Windows: `C:\hamlib\bin\rigctl.exe -l | findstr /i 991`
- Busque el puerto.** Linux: `ls /dev/serial/by-id/`. Windows: Administrador de dispositivos. Algunos equipos crean **dos** puertos — los FT-991A, FTDX10 y FT-710 de Yaesu los llaman *Enhanced* y *Standard*; el CAT está en el **Enhanced**.
- Compruebe y arranque el puente** con `-m <modelo> -r <puerto> -s <velocidad>`, como en el ejemplo del IC-7300: primero `rigctl -m ... -r ... -s ... f`, luego las mismas opciones tras `--rigctl`.

Algunos números de modelo, de Hamlib 4.5 — confírmelos con `rigctl -l`, ya que otra versión de Hamlib puede numerar un equipo de otra forma:

Equipo	-m	Equipo	-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

Opciones adicionales, solo si hacen falta:

- **Un equipo Icom con la dirección CI-V cambiada:** añada `-c` con la dirección en *decimal* — 94h es `-c 148`.
- **Un ajuste que Hamlib ofrece para ese equipo:** `rigctl -m <modelo> -L` los lista; fije uno con `-C nombre=valor`, p. ej. `-C post_write_delay=10` para una interfaz lenta.
- **Un equipo que transmite o se reinicia al abrir el puerto:** algunos cables CAT usan DTR o RTS para el PTT; añada `-C dtr_state=OFF -C rts_state=OFF`.

Ejemplo: Yaesu FT-991A

No probado con un FT-991A real — sigue los ajustes de Hamlib para ese equipo; la comprobación `rigctl ... f` indica al instante si funciona.

En el equipo, ponga **CAT RATE** (menú 031) en 38400. El número de modelo de Hamlib para el FT-991 y el FT-991A es 1035.

El cable USB del FT-991A crea **dos** puertos serie. El CAT está en el **Enhanced**:

- **Linux:** `ls /dev/serial/by-id/` muestra dos líneas para el equipo; la que termina en `-if00-port0` es la Enhanced, normalmente `/dev/ttyUSB0`.
- **Windows:** el Administrador de dispositivos muestra *Silicon Labs Dual CP2105 USB to UART Bridge: Enhanced COM Port (COM5)* y un *Standard COM Port*; use el número del 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
```

Si la comprobación no muestra nada útil, la causa habitual es el puerto Standard en lugar del Enhanced, o un CAT RATE distinto de `-s`.

Ejemplo: Kenwood TS-590SG

No probado con un TS-590SG real — sigue los ajustes de Hamlib para ese equipo; la comprobación `rigctl ... f` indica al instante si funciona.

En el menú del equipo, ponga la velocidad del puerto **USB** en 115200 (el número de menú está en el manual del TS-590SG). El número de modelo de Hamlib es 2037; el TS-590S anterior es 2031. En Windows, instale antes el controlador de puerto COM virtual de Kenwood para el puerto USB.

El cable USB crea **un** puerto serie:

- **Linux:** la línea del equipo en `ls /dev/serial/by-id/`, normalmente `/dev/ttyUSB0`.
- **Windows:** Administrador de dispositivos → Puertos (COM y LPT), por ejemplo `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
```

Si la comprobación falla, la causa habitual es una velocidad USB distinta de `-s`, o un cable en la toma RS-232 (COM) del equipo mientras `-r` indica el puerto USB.

Más ejemplos

Ninguno se ha probado con equipos reales. Cada fila da las opciones que van tras `rigctl` en la comprobación y tras `--rigctl rigctl` en el puente; ponga la misma velocidad en el menú del equipo. El puerto Linux es el habitual — confírmelo con `ls /dev/serial/by-id/`. En Windows, cambie el puerto por el número COM del Administrador de dispositivos y `rigctl` por `C:\hamlib\bin\rigctl.exe`.

Equipo	Menú del equipo	Opciones (Linux)	Notas
Icom IC-705 (USB)	CI-V USB Baud Rate 19200	-m 3085 -r /dev/ ttyACM0 -s 19200	Aparecen dos puertos; el CI-V es el primero (ttyACM0)
Icom IC-7100 (USB)	CI-V USB Baud Rate 19200	-m 3070 -r /dev/ ttyUSB0 -s 19200	La velocidad máxima de Hamlib para este equipo es 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 para el FTDX101MP. Dos puertos; use el Enhanced
Yaesu FTDX10	CAT RATE 38400	-m 1042 -r /dev/ ttyUSB0 -s 38400	Dos puertos; use el Enhanced, como en el FT-991A
Yaesu FT-710	CAT RATE 38400	-m 1049 -r /dev/ ttyUSB0 -s 38400	Dos puertos; use el Enhanced, como en el FT-991A
Yaesu FT-891	CAT RATE 38400	-m 1036 -r /dev/ ttyUSB0 -s 38400	Dos puertos; use el Enhanced, como en el FT-991A
Yaesu FT-450D	CAT RATE 38400	-m 1046 -r /dev/ ttyUSB0 -s 38400	Toma RS-232: use un adaptador USB-serie
Yaesu FT-817 / FT-818	CAT RATE 38400	-m 1020 -r /dev/ ttyUSB0 -s 38400	-m 1041 para el FT-818. Necesita un cable CAT en la toma ACC
Yaesu FT-857 / FT-897	CAT RATE 38400	-m 1022 -r /dev/ ttyUSB0 -s 38400	-m 1023 para el FT-897. Necesita un cable CAT
Kenwood TS-890S (USB)	Velocidad USB 115200	-m 2041 -r /dev/ ttyUSB0 -s 115200	
Kenwood TS-990S (USB)	Velocidad USB 115200	-m 2039 -r /dev/ ttyUSB0 -s 115200	
Kenwood TS-590SG / TS-590S (USB)	Velocidad USB 115200	-m 2037 -r /dev/ ttyUSB0 -s 115200	Ejemplo detallado arriba
Kenwood TS-480	Velocidad del puerto COM 57600	-m 2028 -r /dev/ ttyUSB0 -s 57600	Toma RS-232: use un adaptador USB-serie
Kenwood TS-2000	Velocidad del puerto COM 57600	-m 2014 -r /dev/ ttyUSB0 -s 57600	Toma RS-232; la velocidad máxima de Hamlib para este equipo es 57600
Elecraft K4 (USB)	Velocidad RS232 115200	-m 2047 -r /dev/ ttyUSB0 -s 115200	
Elecraft K3 / K3S	Velocidad RS232 38400	-m 2029 -r /dev/ ttyUSB0 -s 38400	K3S: USB; K3: puerto serie o cable KUSB
Elecraft KX3	Velocidad RS232 38400	-m 2045 -r /dev/ ttyUSB0 -s 38400	Cable KXUSB
Elecraft KX2	Velocidad RS232 38400	-m 2044 -r /dev/ ttyUSB0 -s 38400	Cable KXUSB
Xiegu G90	Velocidad CI-V 19200	-m 3088 -r /dev/ ttyUSB0 -s 19200	Hamlib usa la dirección CI-V por defecto del G90

Equipo	Menú del equipo	Opciones (Linux)	Notas
Xiegu X6100	Velocidad CI-V 19200	<code>-m 3087 -r /dev/ttyUSB0 -s 19200</code>	La velocidad máxima de Hamlib para este equipo es 19200
Lab599 TX-500	—	<code>-m 2050 -r /dev/ttyUSB0 -s 9600</code>	Hamlib usa solo 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>	Un puerto serie USB; la velocidad no importa, pero hay que indicarla

Orden completa de Linux para el IC-705, como ejemplo de cómo leer una fila:

```
rigctl -m 3085 -r /dev/ttyACM0 -s 19200 f
node tci-rigctld.mjs --rigctl rigctl -m 3085 -r /dev/ttyACM0 -s 19200
```

Equipos controlados por otro programa

Algunos equipos ya los controla un programa que puede hacer el trabajo — a veces sin ningún puente. Ninguna de estas vías se ha probado.

Equipo y programa	Qué hacer
SunSDR con ExpertSDR2/3, FlexRadio con AetherSDR, Apache Labs ANAN / Hermes con Thetis	Sin puente. Active el servidor TCI del programa y use TCI-CAT directamente — los puertos 50001 y 40001 se encuentran solos
FlexRadio con SmartSDR (Windows)	Añada un puerto en SmartSDR CAT . Habla CAT de Kenwood, así que un puerto serie allí funciona como un TS-2000: <code>-m 2014 -r COM8 -s 57600</code> . Un puerto TCP funciona con el modelo FlexRadio de Hamlib: <code>-m 2036 -r 127.0.0.1:<puerto></code>
Cualquier equipo controlado por flrig	Deje flrig en marcha y use el modelo flrig de Hamlib; el equipo sigue compartido con fldigi, WSJT-X y programas de registro: <code>-m 4 -r 127.0.0.1:12345</code>
Un rigctld en marcha , o un programa con servidor <i>Hamlib NET rigctl</i>	Arranque el puente sin <code>--rigctl</code> , con esa dirección: <code>node tci-rigctld.mjs 50001 127.0.0.1:4532</code>

Como siempre, las opciones van tras `rigctl` en la comprobación y tras `--rigctl rigctl` en el puente. Una dirección de red como `127.0.0.1:12345` no necesita `-s`.

Compartir el equipo: la vía rigctld

Cuando WSJT-X o un programa de registro deba usar el equipo mientras funciona el puente, arranque el servidor de Hamlib `rigctld` con las mismas opciones, déjelo en marcha y conecte ambos programas a él:

```
rigctld -m 3073 -r /dev/ttyUSB0 -s 115200
node tci-rigctld.mjs
```

Sin `--rigctl` el puente busca `rigctld` en `127.0.0.1:4532` y muestra `rigctld connected`; `node tci-rigctld.mjs 50001 192.168.1.50:4532` usa un `rigctld` de otro ordenador. En WSJT-X elija el equipo *Hamlib NET rigctl* en la misma dirección. En Windows el servidor es `C:\hamlib\bin\rigctld.exe` con las mismas opciones; si Windows responde *Access is denied*, use `--rigctl` y cierre el otro programa mientras funciona el puente.

Puertos. El puente ofrece TCI en el puerto 50001; otro puerto va primero en la línea de órdenes, p. ej. `node tci-rigctld.mjs 40001 --rigctl ...`. Si el puente y el navegador están en ordenadores distintos, ponga en **Host** la dirección del ordenador del puente.

Solución de problemas de TCI-CAT

Síntoma	Causa probable	Qué hacer
El punto sigue gris	El servidor TCI o el puente no está en marcha; Host incorrecto; se denegó el permiso de red local del navegador	Active TCI en ExpertSDR/Thetis/AetherSDR o arranque el puente; revise Host ; permita el acceso a la red local en los ajustes del sitio
<code>rigctl ... f</code> da un error o se queda esperando	Puerto o velocidad incorrectos; sin permiso <code>dialout</code> (Linux) o sin controlador USB (Windows); otro programa tiene el puerto	Ajuste <code>-s</code> al menú del equipo; revise el puerto (<code>ls /dev/serial/by-id/</code> , Administrador de dispositivos); cierre otros programas del equipo
El puente muestra <i>rigctl stopped ... Type: rigctl --help</i>	La orden está incompleta — normalmente falta la velocidad tras <code>-s</code>	Escriba de nuevo la línea completa
El puente muestra <i>The value after -s is missing</i>	La línea de órdenes se cortó al pegarla	Escriba de nuevo el final de la línea, p. ej. <code>-s 115200</code>
El filtro no sigue en uno u otro sentido	CAT Sync está apagado, hay un decodificador en marcha en la página o (IC-7300) los anchos FIL del equipo difieren de 2700 / 2400 / 1800 Hz	Active CAT Sync; detenga el decodificador; ajuste los anchos FIL en el equipo, o indique <code>--filters</code> con los anchos del equipo
El puente muestra <i>rigctl stopped ... rig_open: error</i>	<code>rigctl</code> no puede abrir el puerto	Como para <code>rigctl ... f</code> arriba
El puente muestra <i>the rig is not answering</i>	El puerto se abre pero el equipo no responde: velocidad o modelo incorrectos, CAT desactivado en el menú, o dirección CI-V de Icom cambiada	Repita la comprobación <code>rigctl ... f</code> ; añada <code>-c</code> si la dirección CI-V está cambiada
El puente muestra <i>rigctld not reachable</i>	Arrancado sin <code>--rigctl</code> , y no hay <code>rigctld</code> en marcha	Añada <code>--rigctl ...</code> , o arranque primero <code>rigctld</code>
<code>rigctld.exe</code> dice <i>Access is denied</i> (Windows)	Windows se niega a ejecutarlo	Use <code>--rigctl</code> — solo necesita <code>rigctl.exe</code>
Conectado, pero el receptor no se mueve	CAT Sync está apagado	Actívelo — el silencio al transmitir funciona sin él
El modo del equipo no sigue	Un modo sin equivalente en el receptor, o el controlador Hamlib informa de un nombre poco habitual	La frecuencia se sincroniza igualmente; ajuste el modo en la página
No hay silencio al transmitir	El equipo o su controlador Hamlib no informa del estado de transmisión	No hay nada que ajustar en la página
El dial salta de un lado a otro	Desktop PhantomSDR+ u otro programa también sincroniza el equipo	Use un solo controlador a la vez

Para operadores de receptores

Nada que configurar. El control del transceptor usa una pequeña interfaz JavaScript que ya lleva cada página de PhantomSDR-Plus; no requiere ajustes en el servidor, puertos abiertos ni permisos de administrador. Las funciones de filtro y silencio llegaron con la versión 4.1.0 — tras aplicarla, recompila el frontend (`./recompile.sh`, opción 2); no hace falta detener el receptor. Los receptores KiwiSDR, WebSDR y UberSDR tampoco necesitan nada: la aplicación usa los controles que sus páginas ya tienen.

El botón **TCI-CAT** llegó también con la 4.1.0, de nuevo solo con recompilar el frontend. No abre ningún puerto en el receptor: la conexión va del navegador de cada oyente a su propio ordenador. El puente Hamlib, `tc-bridge/tci-rigctl.d.mjs`, es para que los oyentes lo ejecuten en casa; el receptor no lo usa.

Para desarrolladores: la interfaz de la página

Tanto Desktop PhantomSDR+ como la CATsync Tool usan estas funciones, que cada página de PhantomSDR-Plus pone en `window` al cargarse (las páginas de KiwiSDR, WebSDR y UberSDR se manejan con sus propios controles, distintos):

Función	Devuelve / hace
<code>catsync_ready</code>	<code>true</code> en cuanto las funciones de abajo están instaladas
<code>catsync_getFrequency()</code>	Frecuencia sintonizada, Hz
<code>catsync_setFrequency(hz)</code>	Sintoniza <code>hz</code>
<code>catsync_getMode()</code>	USB, LSB, CW, CW-L, AM, QUAM, FM, WBFM, RADEU, RADEL
<code>catsync_setMode(mode)</code>	Fija el modo; restablece la banda de paso al valor por defecto del modo
<code>catsync_getBandwidth()</code>	Ancho total de la banda de paso, Hz
<code>catsync_setBandwidth(hz)</code>	Fija el ancho — crece hacia arriba en USB, hacia abajo en LSB y por igual en los demás. Llámela después de <code>catsync_setMode</code>
<code>catsync_getMute()</code>	<code>true</code> cuando está en silencio
<code>catsync_setMute(on)</code>	Silencia o quita el silencio, mediante el botón de silencio de la página

Las cuatro últimas llegaron con la 4.1.0, así que compruébelas antes de llamarlas:

```
if (window.catsync_ready) {
  window.catsync_setFrequency(7074000)
  window.catsync_setMode('USB')
  if (typeof window.catsync_setBandwidth === 'function') window.catsync_setBandwidth(2400)
}
```

Fijar la frecuencia resintoniza el audio, así que llame a un setter solo cuando el valor haya cambiado de verdad: llamar continuamente a un setter con el mismo valor se oye. Los antiguos puntos de entrada al estilo KiwiSDR/WebSDR (`setfreq`, `set_mode`, `freqset_complete`) siguen disponibles para las herramientas que los esperan.

Solución de problemas

Síntoma	Causa probable	Qué hacer
<i>Could not open ttyUSB0</i> (Linux)	No está en el grupo <code>dialout</code> , u otro programa tiene el puerto	<code>sudo usermod -aG dialout \$USER</code> , cerrar y abrir sesión; cierre WSJT-X, programas de registro, utilidades del equipo
La lectura del receptor dice <i>not a receiver this app can drive</i>	Otro tipo de receptor web, o la página aún está cargando	Son compatibles PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR y UberSDR; dele unos segundos a una página lenta
<i>The rig did not answer</i>	Velocidad errónea, tipo de equipo erróneo, dirección CI-V errónea, equipo apagado	Ajuste la velocidad al menú del equipo; revise la dirección CI-V; pruebe Hamlib
<i>Hamlib is not installed</i>	No se encontró <code>rigctld</code>	Linux: <code>sudo apt install libhamlib-utils</code> . O indique su ruta en rigctld program
<i>rigctld stopped: ...</i>	Hamlib no pudo abrir el equipo — sigue su propio mensaje	Suele ser el puerto o la velocidad; el texto tras los dos puntos es el motivo según Hamlib
<i>flrig is not running at ...</i>	flrig cerrado, o su puerto XML-RPC es otro	Inicie flrig; revise el puerto en su configuración
Conectado, pero el receptor no se mueve	No hay ventana de estación abierta, o Receiver window fijado a una estación cerrada	Abra la estación o elija <i>The station window last in front</i>
El equipo transmite al conectar	DTR o RTS activan el equipo a través de su interfaz	Desmarque DTR on y RTS on
El filtro no sigue	Receptor anterior a la 4.1.0, o controlador integrado Kenwood/Yaesu	Frecuencia y modo siguen sincronizando; use Hamlib para el filtro en Kenwood/Yaesu
El silencio al transmitir no hace nada	Receptor sin la actualización, o el equipo no informa del estado de transmisión	Igual que arriba
<i>Lost the rig ... reconnecting</i>	Se desconectó el cable, se apagó el equipo o se detuvo <code>rigctld</code>	Nada — reintenta cada 3 segundos y continúa cuando vuelve
Los dos lados saltan sin parar	Dos programas controlan el equipo a la vez	Deje que solo un programa ajuste el equipo, o compártalo mediante flrig

Limitaciones conocidas

- Un equipo y una ventana de receptor a la vez.
- Receptores distintos de PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR y UberSDR — OpenWebRX, por ejemplo — no son compatibles.
- Los controladores integrados siguen los protocolos publicados por los fabricantes y se probaron con equipos simulados y con Hamlib real; para un equipo que se comporte de otra forma, la alternativa es Hamlib.
- No se sincronizan split, VFO B, RIT/XIT ni canales de memoria — solo la frecuencia del VFO activo.
- TCI-CAT en la página del receptor no sincroniza split, VFO B, RIT/XIT ni desplazamientos de transverter, y solo se ha probado con un IC-7300 a través del puente Hamlib.
- En Linux los paquetes usan el Hamlib de la distribución; no incluyen uno propio.

CAPÍTULO 16

Estructura del proyecto

PROJECT_STRUCTURE.md

Estructura del proyecto PhantomSDR-Plus

Este documento ofrece una visión general completa de la estructura de directorios de PhantomSDR-Plus, la organización de los archivos y las relaciones entre componentes.

Índice

1. Árbol de directorios
 2. Directorio raíz
 3. Código fuente (`src/`)
 4. Frontend (`frontend/`)
 5. Listas de frecuencias (`frequencylist/`)
 6. Archivos de configuración
 7. Sistema de compilación
-

Árbol de directorios

```

PhantomSDR-Plus
├── add-receiver.sh          # añade un receptor más: controlador, instances/<nombre>/,
receivers.toml – vea 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: conserva I del CF32 de rx_sdr, escalado a s16 (lo
compila setup-fobos.sh)
├── chat_history.txt
├── check-go.sh             # antiguo: watchdog de la cadena 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          # no está en el repo: lo extrae update-markers.sh de a26allx2.zip
│   ├── 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      # reglas de formato Prettier para las fuentes del frontend
├─ public
│   ├── analyze_users.py
│   ├── decoders
│   │   ├── ft8_lib.wasm
│   │   ├── js8_dict.bin  # diccionario de palabras JS8, se descarga solo cuando hace falta
│   │   └─ js8.wasm      # decodificador JS8, compilado desde 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      # decodificador FLAC que reproduce cada paquete al llegar
│   ├── fsk.js
│   ├── fsk.worker.js
│   ├── fskWorkerProxy.js
│   ├── kiwiSource.js
│   ├── lib
│   │   ├── backend.js
│   │   ├── BandSelector.svelte
│   │   ├── catsync.js
│   │   ├── chatReply.js      # marca y agrupación de respuestas del chat
│   │   ├── 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
├── QrssPanel.svelte
├── receivers.js          # datos del selector de receptor (/receivers.json)
├── rnnoise.js           # carga RNNoise para la reducción de ruido por IA
├── rx.js                 # a qué receptor pertenece una página (?rx=)
├── SMeterAnalog.svelte
├── SMeterDigital.svelte
├── Spectrogram.svelte
├── StatusIndicators.svelte
├── storage.js
├── sUnits.js             # escala de unidades S según la frecuencia (S9 = -93 dBm por
encima de 30 MHz)
├── 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     # mide la conexión y mantiene pequeño el búfer de
reproducción
├── psk31.js
├── refused.js
├── remoteSource.js
└── scanner.js

```

```

| | | | | siteInfo.js # site_information.json, superpuesto con la copia propia del
receptor
| | | | | 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 # cadena de arranque antigua, sustituida por 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 # receptores adicionales, una carpeta cada uno (no en git) – vea
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          # decodificador JS8 en C, compilado a WebAssembly, con sus pruebas y
banco de referencia – véase su 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          # antiguo: mata los procesos del servidor, lo llama go.sh
├── kiwi_install.sh  # instala el puente KiwiSDR en un árbol existente
├── LICENSE
├── logproxy         # copias rotadas de los registros del panel/proxy/autorun
├── logrotate

```

```

├── phantomsdr # configuración de logrotate para proxy.log + admin.log (ajuste las
rutas antes de instalar)
├── manage_admin.sh
├── markers.json
├── meson.build
├── meson_options.txt
├── phantom_fftw_wisdom
├── phantomsdr-admin.service # unidad systemd de ejemplo para el panel (arranca al inicio, se
reinicia tras un fallo)
├── phantomsdr-proxy.service # unidad systemd de ejemplo para el proxy, se instala junto con la
del panel
├── phantomsdr-websdr-relay.service # unidad systemd de ejemplo para el relé de diversidad WebSDR
├── proxy.py
├── rade_helper.py
├── rade_loadtest.csv # salida de rade_loadtest.py (una fila por escalón de carga)
├── rade_loadtest.py # prueba de carga de RADE: cuántos decodificadores simultáneos
soporta la máquina – véase docs/RADE_README.md
├── rade.sh
├── README.md
├── recompile.sh
├── receivers.toml.example # lista de receptores para proxy.py – copiar a receivers.toml
├── _relaunch.sh # antiguo: ayudante de relanzamiento diferido de la cadena go.sh
├── request.hpp
├── setup_admin.sh
├── setup-airspyhf.sh # cadena de driver Airspy HF+: libairspyhf + SoapyAirspyHF + rx_sdr
+ udev
├── setup-cpufreq-perms.sh # da al grupo permiso de escritura en scaling_max_freq para que el
guardián pueda limitar sin root
├── setup-firewall.sh # guardia nftables opcional – vea docs/CONNECTION_LIMITS.md
├── setup-fobos.sh # cadena de driver Fobos: libfobos + SoapyFobosSDR + rx_sdr +
cf32 to real + udev
├── setup-hackrf.sh # HackRF: paquete hackrf de la distribución + udev (nada compilado)
├── setup-rspla.sh # cadena de driver RSP1A: libmirisdr-5 + SoapyMiri + rx_sdr + lista
negra msi2500 + udev
├── setup-rtl-sdr.sh # controlador RTL-SDR: paquete de la distribución, o el controlador
Blog V4 desde el código + bloqueo DVB + udev
├── setup-rx888-udev.sh
├── setup-sdr-common.sh # funciones comunes de los tres scripts setup de arriba
├── setup-websdr-relay.sh # instala el relé de diversidad WebSDR (puerto, identidad, 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 # puente del protocolo KiwiSDR – véase 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                # arranca cada receptor de receivers.toml
├─ start-fobos-hf.sh
├─ start-fobos.sh
├─ start-hackrf.sh
├─ start-rsp1a.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      # servidor TCI para equipos Hamlib (IC-7300...), se ejecuta en el PC
del oyente – ver docs/RIG_CONTROL.md
├── thermal_guard.py          # Protección contra sobrecalentamiento de la CPU para el panel
(también funciona sola)
├── thermal-guard.service    # unidad systemd de ejemplo para el guardián, sin panel de
administración
├── tmpfiles
├── phantomsdr-logs.conf     # mantiene admin.log + proxy.log a nombre del usuario del panel

```

```
(ajuste las rutas antes de instalar)
├─ update.sh
├─ waterfall.sh
├─ websdr_relay.json.example # plantilla de configuración (puerto, límites, identidad de la
estación)
├─ websdr_relay.py           # el relé en sí – véase docs/RECEIVE_DIVERSITY.md
├─ websocketpp_asio_connection.hpp # adaptador Boost >= 1.87: ws_post en lugar de
io_service::post
├─ websocketpp_asio_endpoint.hpp # adaptador Boost >= 1.87: ws_work / ws_restart,
max_listen_connections
├─ websocketpp_asio.hpp       # adaptador Boost >= 1.87 para websocketpp (io_context,
executor_work_guard)
└─ xgo.sh                    # antiguo: arranca spectrumserver, lo llama check-go.sh
```

Directorio raíz

Archivos de configuración

Archivo	Finalidad	Cuándo modificarlo
config.toml	Configuración por defecto	Configuración inicial, pruebas
config-rtl.toml	Configuración específica de RTL-SDR	Al usar un RTL-SDR
config-rspla.toml	Configuración de SDRplay RSP1A	Al usar un RSP1A
config-airspyhf.toml	Configuración de Airspy HF+	Al usar un Airspy
config-fobos-hf.toml	RigExpert Fobos SDR, HF1/HF2 en muestreo directo (0-25 MHz)	Al usar un Fobos en HF
config-fobos.toml	RigExpert Fobos SDR, rama RF (25-6000 MHz)	Al usar un Fobos por encima de 25 MHz
config-hackrf.toml	Configuración de HackRF One	Al usar un HackRF
config-rx888mk2.toml	Configuración de RX888 MK2	Al usar un RX888
config.example.hackrf.toml	Ejemplo para HackRF One	Al usar un HackRF

Scripts de arranque, parada y mantenimiento

Cada `start-*.sh` de los siguientes es un **lanzador autónomo + watchdog + registrador**: detiene cualquier instancia en marcha, levanta el receptor y `spectrumserver`, pasa a segundo plano, reinicia automáticamente la cadena si se cae y registra en `logwebsdr.txt`. Comparten un único script de parada y un solo bloqueo `flock` (solo funciona un receptor a la vez). Edite únicamente el bloque **RECEIVER CONFIGURATION** de la parte superior de cada uno (argumentos del receptor / configuración / nombre del proceso). Los seis scripts de arranque hechos con la misma plantilla (`start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos.sh`, `start-fobos-hf.sh`, `start-hackrf.sh`) son idénticos por debajo de ese bloque; `start-rx888mk2.sh` es un script propio. Desde la v5.0.0 pueden funcionar varios receptores a la vez: si un script de arranque se lanza con `INSTANCE=<nombre>`, ese receptor tiene su propia carpeta en `instances/<nombre>/` y su propio registro, bloqueo y FIFO — vea [Varios receptores](#).

Script	Finalidad
<code>install.sh</code>	Instalación y compilación automatizadas
<code>add-receiver.sh</code>	Añade un receptor más a este ordenador: pregunta qué receptor y qué cubre, instala su controlador, crea <code>instances/<nombre>/</code> y la entrada en <code>receivers.toml</code> , y ofrece arrancarlo y reiniciar el proxy. Los instaladores lo ofrecen al final — vea Varios receptores
<code>start-rtl.sh</code>	Arranque + watchdog del servidor con RTL-SDR (<code>rtl_sdr</code>)
<code>start-rspla.sh</code>	Arranque + watchdog del servidor con SDRplay RSP1A (<code>rx_sdr</code>); usa <code>libmirisdr-5</code> , o la API de SDRplay cuando está instalada (<code>RX_DRIVER</code> fuerza uno)
<code>start-airspyhf.sh</code>	Arranque + watchdog del servidor con Airspy HF+ (<code>rx_sdr</code>)
<code>start-fobos-hf.sh</code>	Arranque + watchdog del servidor con RigExpert Fobos SDR, HF1/HF2 en muestreo directo (<code>rx_sdr \ cf32_to_real</code>)
<code>start-fobos.sh</code>	Arranque + watchdog del servidor con RigExpert Fobos SDR, rama RF (<code>rx_sdr</code>)
<code>start-hackrf.sh</code>	Arranque + watchdog del servidor con HackRF One (<code>hackrf_transfer</code>)
<code>start-rx888mk2.sh</code>	Arranque + watchdog del servidor con RX888 MK2 (<code>rx888_stream</code>)
<code>start-all.sh</code>	Arranca cada receptor de <code>receivers.toml</code> con su propio script — póngalo como script de arranque del panel de administración en una estación con varios receptores; vea Varios receptores
<code>stop-websdr.sh</code>	Detiene el servidor y su watchdog: compartido por todos los receptores
<code>stop-websdr.sh</code> <code><instancia></code>	Detiene un solo receptor (<code>main</code> para el principal); sin argumento se detienen todos, como antes
<code>setup-rx888-udev.sh</code>	Instala reglas udev para que <code>rx888_stream</code> del RX-888 funcione sin sudo
<code>setup-rspla.sh</code> / <code>setup-fobos.sh</code> / <code>setup-airspyhf.sh</code>	Instalan la cadena de driver SoapySDR del RSP1A, el Fobos o el Airspy HF+ — SoapySDR desde la distribución, driver y <code>rx_tools</code> compilados en <code>sdr_drivers/</code> , una regla udev — en cualquiera de las cuatro distribuciones admitidas. Las opciones 3, 5 y 6 del instalador los ejecutan; también funcionan solos. Código común en <code>setup-sdr-common.sh</code> — vea la Guía de instalación
<code>setup-hackrf.sh</code>	Instala el paquete <code>hackrf</code> de la distribución y una regla udev para el HackRF One, en cualquiera de las cuatro distribuciones admitidas; la opción 7 del instalador lo ejecuta
<code>setup-rtlsdr.sh</code>	Instala el controlador RTL-SDR en cualquiera de las cuatro distribuciones admitidas — el paquete <code>rtl-sdr</code> de la distribución o, con <code>RTL_V4=y</code> , el controlador del RTL-SDR Blog V4 compilado desde el código —, bloquea el controlador DVB-T y añade una regla udev; lo ejecuta <code>add-receiver.sh</code>
<code>setup-firewall.sh</code>	Guardia de avalanchas opcional en el núcleo: carga una tabla <code>nftables</code> con un tope de conexiones simultáneas y una tasa por dirección de origen en los puertos del receptor, un freno contra fuerza bruta en SSH y la compartición de archivos de Windows cerrada fuera de los rangos privados. Necesita root, no puede dejarle fuera (<code>policy accept</code> , las conexiones establecidas se aceptan primero) y <code>--apply</code> se revierte solo si no se confirma en 60 s — vea Límites de conexión
<code>setup-cpufreq-perms.sh</code>	Concede a un grupo <code>cpufreq</code> permiso de escritura sobre el límite de frecuencia de la CPU, para que la fase <code>throttle</code> del guardián funcione sin root. Instala una regla <code>tmpfiles.d</code> para que sobreviva a un reinicio; <code>--revoke</code> lo deshace
<code>update.sh</code>	Actualizar la instalación desde el árbol publicado sin tocar su configuración, marcadores, lista de frecuencias ni cambios locales — véase la Guía de instalación
<code>recompile.sh</code>	Reconstruir el backend o el frontend y elegir la variante servida en <code>/</code>
<code>smeter_theme.sh</code>	

Script	Finalidad
	Fijar la esfera por defecto del S-meter analógico (dark / amber / vintage) para todos los usuarios y ofrecer la reconstrucción del frontend — véase Editar variantes
<code>waterfall.sh</code>	Cambia el nivel mínimo de cascada predeterminado (dB) en <code>waterfall.js</code> + <code>App.svelte</code> — véase README
<code>kiwi_install.sh</code>	Instala la emulación de clientes KiwiSDR en un árbol que no la tiene: parchea las fuentes del backend, copia <code>src/kiwi_bridge.h</code> y añade un bloque <code>[kiwi_emulation]</code> documentado a los archivos de configuración de la raíz. Idempotente, y respalda cada archivo que toca — véase Emulación de clientes KiwiSDR
<code>tci-bridge/tci-rigctld.mjs</code>	El receptor no lo usa. Un pequeño programa Node.js que el oyente ejecuta junto a su propio transceptor: lee frecuencia, modo y estado de transmisión del <code>rigctld</code> de Hamlib y los ofrece como TCI en el puerto 50001, para que el botón TCI-CAT de la página controle un equipo sin TCI propio, como el IC-7300 — ver Control del transceptor

Cadena de arranque antigua. `go.sh`, `xgo.sh`, `check-go.sh`, `kill.sh` y `_relaunch.sh` son la generación anterior de scripts de arranque, vigilancia y parada. Todo lo que hacían está ahora dentro de cada `start-<radio>.sh`, que es lo que debe usar. Se conservan en disco porque las instalaciones existentes los referencian, y no reciben mantenimiento.

Archivos de datos

Archivo	Finalidad	Formato
<code>markers.json</code>	Marcadores de frecuencia	JSON
<code>chat_history.txt</code>	Mensajes de chat de los usuarios	Texto plano
<code>favicon.ico</code>	Icono del sitio web	Imagen ICO
<code>fftw_wisdom</code>	Datos de optimización de la FFT	Binario FFTW
<code>phantom_fftw_wisdom</code>	Optimización adicional de la FFT	Binario FFTW

Panel de administración y reporte de spots

Archivo	Finalidad
<code>admin_server.py</code>	El propio panel de administración. Además de las páginas de gestión ejecuta el muestreador de la página Gráficos : un hilo en segundo plano toma cada 2 segundos la frecuencia de CPU, la carga, la temperatura y los usuarios conectados, y los guarda solo en memoria: 1 hora a resolución completa más 24 horas de promedios de 30 segundos. No se escribe nada en disco, así que el historial se pierde al reiniciar.
<code>admin_config.json</code>	Ajustes del panel de administración (hash de contraseña, puertos, opciones, umbrales de la protección térmica)
<code>thermal_guard.py</code>	Protección contra sobrecalentamiento de la CPU. Detiene el servidor cuando la CPU se calienta demasiado y lo reinicia una vez fría, derivando sus umbrales del límite crítico que publica su propia CPU en lugar de una cifra fija. Nunca intenta identificar qué supervisa el servidor: mientras hay exceso de temperatura repite la parada cada 2 s, de modo que un watchdog, una unidad de systemd o una tarea cron que lo reviva queda deshecha hasta que la máquina se enfríe. Solo biblioteca estándar; lo importa <code>admin_server.py</code> (temporizado por el muestreador de Gráficos) y también se ejecuta por su cuenta en instalaciones sin panel. Por defecto solo registra, así que no actúa sobre nada hasta que se active — véase [Panel de administración]véase el manual del Thermal Guard
<code>autorun/</code>	El demonio de reporte de spots — véase Autorun Spot Reporter
<code>autorun.json</code>	Bandas y modos a decodificar, identidad y destinos
<code>autorun-status.json</code>	Estado en vivo del demonio: alimenta los contadores de la ejecución actual (los paneles por decodificador), que se ponen a cero con Stop/Start
<code>autorun-totals.json</code>	Spots subidos históricos por banda y modo: el número junto a cada casilla; lo escribe el demonio para que sobreviva a los reinicios

Archivos del sistema de compilación

Archivo	Finalidad
<code>meson.build</code>	Configuración principal de compilación
<code>meson_options.txt</code>	Opciones de compilación configurables
<code>.gitattributes</code>	Atributos del repositorio Git

Código fuente (`src/`)

El directorio `src/` contiene la implementación del backend en C++.

Componentes clave

1. Aplicación principal (`main.cpp`)

- Analiza los argumentos de la línea de órdenes
- Carga el archivo de configuración
- Inicializa los componentes del servidor
- Arranca el bucle de eventos

2. Servidor de espectro (`spectrumserver.cpp`)

- Coordina todos los componentes
- Gestiona las conexiones de los usuarios

- Distribuye los datos de espectro
- Atiende las peticiones de los usuarios

3. Controladores de SDR (`drivers/`)

- Interfaz abstracta para el hardware SDR
- Lectura y formateo de los datos de muestras
- Manejo de las funciones propias de cada dispositivo

4. Motor de DSP (`dsp/`)

- Cálculo de la FFT (acelerado por CPU/GPU)
- Demodulación (AM, FM, SSB, CW, etc.)
- Filtrado y remuestreo de audio
- AGC y reducción de ruido

5. Servidor web (`server/`)

- Comunicación por WebSocket
- Servicio de archivos estáticos por HTTP
- Gestión de sesiones de usuario
- Transmisión de datos en tiempo real

6. Codificación de audio (`audio/`)

- Compresión FLAC
- Compresión Opus
- Optimización de la transmisión

Frontend (`frontend/`)

La interfaz de usuario web construida con Svelte y Vite.

1. Aplicación principal (`App.svelte`)

- Componente de nivel superior
- Estructura del diseño
- Orquestación de componentes
- **Fila de botones de decodificador** — un botón por decodificador en el panel principal, justo debajo del selector de modos; una pulsación inicia el decodificador y abre su ventana, una segunda lo detiene. Sustituyó a la antigua fila de ancho de banda. RADEL/RADEU están en `lib/ModesSelector.svelte`, junto al selector de modos y dentro de las ventanas emergentes **Modes** y **Bands**.

2. Cascada (`waterfall.js` + `lib/`)

- Representación del espectro y de la cascada en canvas, mapas de color y el ajuste automático adaptativo, todo en `waterfall.js` (JS puro, no un componente)
- Sintonía interactiva mediante `lib/PassbandTuner.svelte`
- Superposiciones del plan de bandas y de los marcadores mediante `lib/FrequencyMarkers.svelte`
- El espectrograma de audio es un componente aparte: `lib/Spectrogram.svelte`

3. Controles (`App.svelte` + `lib/`)

- Entrada e indicación de frecuencia — `lib/FrequencyInput.svelte`

- Selección de modo (AM/FM/SSB/CW) — `lib/ModesSelector.svelte`, cambio de banda — `lib/BandSelector.svelte`
- AGC/NR/NB y el resto de controles están en el propio `App.svelte`; no existe un `Controls.svelte` aparte

3a. Escáner (`scanner.js`)

- Explorador de canales: recorre el receptor por un margen y se detiene en el primer canal con señal. JS puro, no un componente — `App.svelte` aporta el VFO, el modo, el plan de bandas y la llamada de sintonía, y recibe de vuelta el estado de la interfaz por una sola función de retorno
- El umbral es en dB sobre el ruido de fondo de la banda en vez de un nivel absoluto, reutilizando el ruido que `waterfall.js` ya sigue (`snrNoiseDb`), así que no necesita calibración propia
- Dos maneras de recorrer la banda: sintonizar y escuchar cada canal, o examinar antes el espectro y sintonizar sólo lo que no pueda descartarse. El espectro puede saltarse un canal pero nunca detenerse en uno — toda parada procede de una escucha real
- El margen es la banda del plan de bandas o exactamente lo que muestra la cascada; la reanudación automática, el tiempo máximo de permanencia y la lista de exclusión también viven aquí, guardados en `localStorage`

4. Sistema de audio (`audio.js`)

- Flujo de audio por WebSocket
- Decodificación FLAC/Opus
- Control de la reproducción de audio
- Distribuye el PCM en bruto (tomado antes del AGC, la reducción de ruido y el silenciado) a los decodificadores de modo
- Reducción de ruido por IA: `lib/rnnoise.js` carga el módulo WebAssembly de RNNoise (`@jitsi/rnnoise-wasm`) en el primer uso; `audio.js` lo aplica solo en modos de voz, después de la toma de los decodificadores, así que estos nunca lo oyen
- FLAC pasa por `flacLowLatency.js`, que decodifica cada paquete en el momento en que llega y lleva la frecuencia real del flujo (por ejemplo 12016,3 Hz) a 12000 Hz con un remuestreador de 64 coeficientes (2,7 ms de retardo). El decodificador WASM de `modules/` entregaba el audio en bloques fijos de 1024 muestras y retenía unos 160 ms; se sigue usando para Opus, para las fuentes de diversidad y para FLAC cuando la página se abre con `?flac=wasm`
- Control del búfer de reproducción (`playoutControl.js`): `ArrivalMeter` mide lo irregular que es la llegada de los paquetes respecto a la propia línea de tiempo del flujo, y la deriva entre el reloj de muestreo del receptor y la tarjeta de sonido del oyente; `CushionController` mantiene el menor búfer del último minuto cerca del margen del paso de búfer elegido, recortando o repitiendo un periodo ajustado al tono (`splICEout` / `splICEin`) del audio que va al altavoz. Los decodificadores y las grabaciones reciben el flujo intacto. Se aplica a la ruta de búferes programados que usa todo oyente de una estación en `http simple`; `?drain=0` lo desactiva y `?diag=1` muestra sus cifras

4a. Decodificadores de modo y sus workers

Cada uno de los decodificadores de modo más pesados se ejecuta en su propio Web Worker, de modo que la decodificación nunca bloquea la reproducción de audio ni la cascada. Todos siguen un mismo patrón: tres archivos por decodificador.

Decodificador	Motor	Worker	Proxy en el hilo principal
SSTV	sstv.js	sstv.worker.js	sstvWorkerProxy.js
FAX de HF	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

- El **motor** es código de DSP puro que desconoce por completo los workers, así que también puede ejecutarse directamente (pruebas unitarias o el modo alternativo en el mismo hilo).
- El **worker** mantiene una instancia del motor y reenvía sus eventos tal cual.
- El **proxy** replica la superficie de métodos del motor, de modo que `audio.js` lo llama exactamente igual que llamaría al decodificador. Crea el worker de forma perezosa al activarlo por primera vez y recurre a la ejecución en el hilo principal si no hay workers disponibles.

Dos detalles son fundamentales: el PCM se **copia** en un búfer nuevo antes de transferirlo al worker (transferir una vista del acumulador de audio lo desvincularía y detendría la reproducción), y el mensaje `init` del worker vuelve a aplicar la configuración a un motor ya en marcha en lugar de suponer que es nuevo.

`fsk.js` atiende tanto NAVTEX como FSK/RTTY desde un único motor, seleccionado por instancia mediante un campo `role`; cada instancia tiene su propio estado, así que ambos pueden funcionar de forma independiente.

El rol `fsk` alberga además dos decodificadores que no son FSK en absoluto. Al seleccionar la variante `psk31` u `olivia`, `fsk.js` entrega el audio a `psk31.js` o a `olivia.js` en lugar de a su propia cadena discriminadora, pero sigue aprovechando su configuración, su worker y su gestión de eventos — de modo que `fsk.worker.js`, `fskWorkerProxy.js` y `audio.js` no necesitan saber nada de ninguno de los dos modos, y la interfaz consume en todo momento los mismos eventos `char/status/metrics`.

- `psk31.js` — BPSK31: banda base compleja, filtro adaptado, detección diferencial y varicode, con una adquisición espectral gruesa más un AFC fino que cubre unos ± 25 Hz.
- `olivia.js` — Olivia MFSK: una adaptación del receptor MFSK de Pawel Jalocho procedente de fldigi (`pj_mfsk.h`, GPL-3, igual que este proyecto), incluida la corrección de errores Walsh/Hadamard y la búsqueda ciega de sincronización sobre la fase de bloque y el desplazamiento de frecuencia.
- `broadcastSchedules.js` — los horarios UTC que los decodificadores de FAX, NAVTEX y RTTY ofrecen como preajustes, tomados de los programas de facsímil marino de NOAA/NWS y de las listas publicadas de estaciones NAVTEX

4b. Diversidad de recepción (`diversity.js`)

- Combina el receptor local con un segundo receptor en otro lugar y sigue al emplazamiento que en cada momento tiene mejor señal. JS puro, no un componente — se apoya en una única costura de `audio.js`, que le entrega el PCM local y reproduce lo que devuelve
- **Selección, no suma.** Dos emplazamientos oyen la misma emisión por caminos ionosféricos distintos, así que sus formas de onda tienen fase no relacionada; sumarlas suena a filtro de peine. La combinación coherente exigiría un reloj común, que dos receptores por internet no comparten. Solo se mezclan durante un fundido de 30 ms
- La alineación correla las dos **envolventes de audio** (potencia logarítmica a 100 Hz), nunca las formas de onda — la envolvente sobrevive al camino y a cualquier códec. Un enganche solo se acepta cuando una segunda búsqueda independiente coincide, lo que descarta el retardo rotundo pero equivocado que darían dos emplazamientos desvaneciéndose en antifase

- El flujo remoto se **enclava en velocidad** con el local antes que nada. Dos receptores son dos relojes y dos cadenas de diezmado, así que su audio llega hasta un 2% desperejado aunque ambos declaren 12 kHz — 240 muestras por segundo de deriva, que ninguna correlación aguanta. La relación se mide a partir de cuántas muestras entrega realmente cada lado y la aplica un remuestreador que arrastra su fase fraccionaria entre bloques, de modo que una relación arbitraria se sostiene indefinidamente
- La elección de emplazamiento usa un SNR por percentiles medido sobre muestras **alineadas en contenido**, con histéresis, temporizador de permanencia y una salida rápida si el emplazamiento activo se hunde. Los niveles se igualan ruido contra ruido, de modo que un cambio no altera el siseo de fondo
- Los decodificadores conservan el flujo **local**: FT8, JS8, WSPR y RADE integran de forma coherente durante una ranura, y un cambio a mitad es una discontinuidad de fase que puede costar la decodificación

4c. Fuentes de diversidad (`remoteSource.js`, `kiwiSource.js`, `uberSource.js`, `webSdrSource.js`)

- Un único contrato — `onPcm` / `onState` / `tune` / `canReceive` — de modo que `diversity.js` nunca sabe qué hay al otro lado. Añadir un tipo de receptor es un fichero nuevo
- `remoteSource.js` — otro PhantomSDR-Plus por `/audio` (cbor + FLAC), reutilizando `createDecoder()` de `lib/wrappers.js`
- `kiwiSource.js` — un KiwiSDR: tramas SND, PCM big-endian, con la marca de tiempo GPS de 10 bytes que un paquete estéreo inserta antes del audio
- `uberSource.js` — un UberSDR por su `/ws` nativo: Opus tras una cabecera de 21 bytes, resintonizando sobre la conexión abierta. El identificador de sesión debe registrarse antes con `POST /connection` y debe ser un UUID
- `webSdrSource.js` — un WebSDR, a través de `websdr_relay.py` en este servidor: el navegador no puede conectarse directamente porque WebSDR comprueba la cabecera `Origin` y ningún script puede cambiarla. La sintonía va como trama de texto por la misma conexión; la cobertura de bandas la aporta el relé
- `webSdrCodec.js` — el formato de audio de WebSDR: un flujo etiquetado byte a byte cuyos bloques comprimidos alimentan un predictor leaky-LMS de 20 tomas. Portado del cliente de WebSDR y verificado muestra a muestra contra él
- `diversityList.js` — la lista de receptores guardados y las reglas de dirección de los cuatro tipos, compartidas por el panel de escritorio y la página móvil para que un solo formato sirva a ambos. También la transferencia — un blob compacto, su código QR y un analizador que acepta todas las formas que las dos páginas han escrito alguna vez — y `browseUrl()`, que devuelve una dirección marcada a una que un navegador puede abrir
- `lib/DiversityPanel.svelte` — la interfaz: dirección, tipo de fuente, ajuste de SNR y estado en vivo, más los receptores guardados — con nombre, reordenables y exportables e importables como archivo JSON, por tipo de fuente en `localStorage`. Todo se edita dentro del panel: `prompt()` y `confirm()` bloquean el hilo principal, que es por donde el audio llega al worklet de reproducción. Un botón  **QR** dibuja la lista como código escaneable, porque `localStorage` pertenece a un navegador y un teléfono empieza vacío. `mobile/Mobile.svelte` lleva la misma función en una pestaña **Div**, con el CSS sencillo propio de esa página. Véase [Diversidad de recepción](#)

4d. Identificador de modo (`modeId.js`, `modePriors.js`)

- Responde a «¿qué estoy escuchando?». Lee la misma toma de PCM en bruto que los decodificadores y ordena los modos probables, para que el operador elija el decodificador correcto en vez de probar los diez. Nunca decodifica: mide propiedades físicas de la señal y las puntúa contra una tabla de modos conocidos

- El ancho ocupado es la anchura contigua a -15 dB alrededor del pico, deliberadamente no una cifra del 99 % de potencia: los clics de manipulación dejan colas largas en la integral de potencia y hacían que todo modo estrecho pareciera varias veces más ancho. La velocidad de símbolo sale de la **frecuencia instantánea** y no de las energías de los tonos, porque ninguna integración resuelve a la vez un desplazamiento de 170 Hz y un símbolo de 100 Bd. La velocidad de manipulación sale de la envolvente on/off y sirve además como lectura de velocidad en CW
- `modePriors.js` aporta la única pista que el audio no puede llevar: dónde está sintonizado. Una señal de 100 Bd / 170 Hz en 518 kHz es NAVTEX; la misma señal en 14,070 MHz no lo es. Solo repondera lo que la señal ya sostenía, y nunca inventa un candidato
- FT8, JS8 y FT2 se informan como un grupo a propósito: no se separan solo por ancho y separación de tonos, y fingir lo contrario sería una respuesta equivocada dicha con seguridad
- Por debajo de unos 10 dB de SNR calla en lugar de adivinar
- Corre en su propio Web Worker (`modeId.worker.js` + `modeIdWorkerProxy.js`), con el mismo patrón motor/worker/proxy que los decodificadores anteriores; el resultado es la etiqueta de `lib/ModeIdChip.svelte`

4e. Conexiones rechazadas (`refused.js`, `clientVersion.js`)

- `refused.js` es el vocabulario compartido para una conexión que el servidor rechaza: código de cierre **4003** (por encima de un límite por dirección, o la página es anterior a `[server] min_client_version`) y **4001** (una expulsión del sysop). Ambos son definitivos. `audio.js`, `waterfall.js` y `events.js` importan `isRefusal()` de ahí, porque los tres abren una conexión que puede ser rechazada y los tres deben resolver su promesa de inicio cuando ocurre — si no, la página espera para siempre una conexión que nunca llegará
- Nada reintentada. Una conexión `/audio` caída termina la sesión a propósito: `/waterfall` y `/events` nunca volvían con una reconexión, así que una sesión reintentada era audio vivo sobre una cascada congelada, y frente a un límite de tasa cada intento alargaría justo el rechazo que trataba de sortear. La página de escritorio explica un rechazo que ocurre al cargar y simplemente se detiene cuando ocurre a mitad de sesión; `/mobile` muestra una línea pidiendo al oyente que recargue
- `clientVersion.js` contiene un único entero, `CLIENT_VERSION`, que la página añade a su conexión de audio como `/audio?v=N`. El servidor rechaza todo lo inferior a `[server] min_client_version`, que es como una estación obliga a recargar a las pestañas que aún ejecutan una versión anterior — la única palanca que hay, ya que el servidor no alcanza el JavaScript que ya corre en un navegador. Incrementélo cuando un cambio del frontend no deba seguir siendo ignorado por pestañas abiertas
- Referencia completa: [Límites de conexión](#)

4f. Respuestas en el chat (`lib/chatReply.js`)

- Una respuesta es un mensaje de chat corriente que empieza con una marca que nombra el mensaje al que responde, `[RE:<marca de tiempo>|<nombre>]`; el servidor la guarda y la reenvía como cualquier otra línea y no necesita cambios
- `chatReply.js` crea y lee esa marca y agrupa las respuestas bajo su original, con un nivel de profundidad. La usan tanto `App.svelte` como `mobile/Mobile.svelte`, de modo que las dos páginas organizan el chat igual
- Cuando el original ya ha salido del historial de 20 mensajes del servidor, la respuesta se queda donde llegó y muestra solo el nombre y la hora a los que responde

5. Gestión del estado (`stores/`)

- Almacenes de datos reactivos
- Estado compartido de la aplicación
- Manejo de eventos

Funciones clave

- Procesamiento de audio en tiempo real
- Decodificación de modos digitales (FT8, RTTY, etc.)
- Filtrado de audio
- Análisis de espectro

Listas de frecuencias (`frequencylist/`)

Los marcadores de frecuencia sobre la cascada. `mymarkers.json` es la lista que el receptor muestra realmente; el resto es la materia prima que `update-markers.sh` convierte en ella, actualizada desde las programaciones en línea. El `README.md` de ese directorio explica la actualización en los siete idiomas.

```
frequencylist/
├─ mymarkers.json          # los marcadores que muestra el receptor – editados a mano,
sobreviven a una actualización
├─ shortwavestations.json  # emisoras de onda corta, generado por generate-current-shortwave.py
├─ 0.TXT                  # la programación global de HF A26, descomprimida y analizada por
update-markers.sh
├─ a26allx2.zip           # archivo de origen descargado por update-markers.sh
├─ admin.txt · antenna.txt · broadcas.txt · fmorg.txt · language.txt · site.txt
├─                        # listas auxiliares de la misma fuente (emplazamientos, idiomas,
administradores, antenas)
├─ generate-current-shortwave.py
├─ update-markers.sh      # actualiza todo lo anterior desde las programaciones en línea
└─ README.md
```

Formato (`mymarkers.json` , `shortwavestations.json`)

```
[
  { "frequency": 77500, "name": "DCF77", "mode": "CW" },
  { "frequency": 2485000, "name": "Vanuatu Broadcasting", "mode": "AM" }
]
```

Archivos de configuración

Configuración del servidor (archivos `.toml`)

Estructura de los archivos de configuración:

```

[server]
# Web server settings
port = 9002
html_root = "frontend/dist/"
threads = 2
otherusers = 1

[limits]
# Límites por dirección – todos desactivados o con un valor sensato por defecto,
# así que una configuración sin ellos se comporta como siempre. Vea CONNECTION_LIMITS.md.
per_ip = 3          # oyentes simultáneos desde una dirección
per_ip_rate = 40     # conexiones nuevas por minuto desde una dirección

[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 = 145500000
modulation = "FM"

```

Información del sitio (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
}

```

Sistema de compilación

Configuración de compilación con Meson

`meson.build` (raíz)

Define:

- Metadatos del proyecto
- Dependencias
- Opciones del compilador
- Listas de archivos fuente
- Objetivos de compilación

`meson_options.txt`

Opciones disponibles:

```
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')
```

Dependencias entre archivos

Dependencias de compilación del backend

```
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
```

Dependencias de compilación del frontend

```
frontend/dist/ depends on:
├─ Source files (frontend/src/**/*.*)
├─ npm packages (node_modules/):
│   ├── Svelte
│   ├── Vite
│   ├── @wasm-audio-decoders/opus-ml
│   └─ ...
└─ Static assets (frontend/public/)
```

Flujo de datos

Flujo de funcionamiento del servidor

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

Flujo de interacción del usuario

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

Guía de modificación de archivos

Cuando modifica el código del backend (`src/**`):

```
cd PhantomSDR-Plus
meson compile -C build
# Server restart required
```

Cuando modifica el código del frontend (`frontend/src/**`):

```
cd PhantomSDR-Plus/frontend
npm run build
cd ..
# Server restart required (for static files)
```

Cuando modifica la configuración (`.toml`, `.json`):

```
# Restart server
./stop-websdr.sh
./start-rtl.sh # (or appropriate start script)
```

Cuando modifica los marcadores (markers.json):

```
# Reload page in browser
# No server restart needed
```

Rutas importantes

Rutas en ejecución

- **Configuración:** `./config-*.toml`
- **Raíz HTML:** `./frontend/dist/`
- **Marcadores:** `./markers.json`
- **Historial del chat:** `./chat_history.txt`
- **FFTW wisdom:** `./fftw_wisdom, ./phantom_fftw_wisdom`

Rutas de compilación

- **Binario generado:** `./build/spectrumserver`
- **Salida del frontend:** `./frontend/dist/`
- **Módulos de Node:** `./frontend/node_modules/`

Rutas de código fuente

- **Fuente del backend:** `./src/`
- **Fuente del frontend:** `./frontend/src/`
- **Bibliotecas de DSP:** `./jsdsp/`

Operaciones habituales con archivos

Añadir una configuración de SDR nueva

1. Copie una configuración existente: `cp config-rtl.toml config-mydevice.toml`
2. Edite los parámetros: `nano config-mydevice.toml`
3. Cree un script de arranque: `cp start-rtl.sh start-mydevice.sh`
4. Edite el script de arranque: `nano start-mydevice.sh`; cambie únicamente el bloque **RECEIVER CONFIGURATION** de la parte superior (`RX_LABEL`, `RX_COMM` = el nombre del proceso del receptor, `RX_ARGS`, `CONFIG`, `FIFO` y el enganche `prestart` si el dispositivo lo necesita). La lógica de lanzamiento/watchdog/registro que hay debajo es genérica y no requiere cambios.
5. Hágalo ejecutable: `chmod +x start-mydevice.sh`

`stop-websdr.sh` ya detiene cualquier `start-*.sh --watchdog`; si su receptor usa un nombre de proceso distinto de `rx888_stream/rx_sdr/rtl_sdr/hackrf_transfer/cf32_to_real`, añada también allí una línea `killall -9 <nombre>`.

Personalizar el frontend

1. Modifique el código fuente: `nano frontend/src/App.svelte`
2. Recompile: `cd frontend && npm run build && cd ..`
3. Reinicie el servidor: `./stop-websdr.sh && ./start-rtl.sh`

Añadir marcadores propios

1. Edite el archivo de marcadores: `nano markers.json`
 2. Formato: `json { "markers": [ { "frequency": 145500000, "label": "2m Calling", "mode": "FM" } ] }`
 3. Recargue el navegador (no hace falta reiniciar el servidor)
-

Control de versiones

Archivos que conviene versionar en Git

- Código fuente (`src/`, `frontend/src/`, `jsdsp/`)
- Ejemplos de configuración (`config.example*.toml`)
- Sistema de compilación (`meson.build`, `meson_options.txt`)
- Documentación (`*.md`, `docs/`)
- Scripts (`*.sh`)

Archivos que conviene ignorar (`.gitignore`)

- Resultados de compilación (`build/`, `frontend/dist/`)
 - Dependencias (`frontend/node_modules/`)
 - Datos de usuario (`chat_history.txt`)
 - Configuraciones personales (`config-rtl.toml` si se ha personalizado)
 - Datos binarios (`*.o`, `*.so`)
-

Esta documentación de la estructura debería ayudarle a orientarse y a entender el código de PhantomSDR-Plus.

Para las instrucciones de instalación, consulte [INSTALLATION.md](#). Para información sobre el uso, consulte [USER_GUIDE.md](#).

73 de SV1BTL & SV2AMK