

PhantomSDR-Plus

Documentation complète

16 documents · généré le 2026-10-03

Sommaire

1. README	8
Fonctions principales	8
Matériel pris en charge	10
Démarrage rapide	11
Mesures de performance	12
Utilisation	12
Configuration	13
Plancher de la cascade — waterfall.sh	14
Personnalisation	15
Documentation	17
Liste WebSDR en ligne	17
Dépannage	17
Contribuer	18
Licence	18
Auteurs et remerciements	18
Liens	18
Assistance	18
Conseils de performance	18
2. Guide d'installation	19
Table des matières	19
Configuration système requise	19
Préparation avant installation	20
Installation des dépendances	21
Installation de Node.js et npm	22
Installation d'OpenCL (facultatif mais recommandé)	23
Installation de PhantomSDR-Plus	25
Installation du codec audio Opus	30
Rapporteur de spots autorun (FT8/FT4/WSPR)	31
Émulation des clients KiwiSDR (facultatif)	33
Configuration	34
Configuration propre à chaque appareil SDR	37
Tests et vérification	40
Mise en place du démarrage automatique	41
Protection thermique du processeur	43
Dépannage	45
Optimisation des performances	49
Mise à jour de PhantomSDR-Plus	49
Sauvegarde et restauration	52
Considérations de sécurité	53
Obtenir de l'aide	53
Annexe A : liste complète des dépendances	54
Annexe B : exemples de configuration	54
3. Configuration du panneau d'administration	56
Vue d'ensemble	56
Ce que vous apporte le panneau d'administration	56

Prérequis	57
Étape 1 — Fichiers	57
Étape 2 — Lancer le script d'installation	57
Étape 3 — Démarrer / arrêter / redémarrer	58
Étape 4 — Ouvrir le port du pare-feu	61
Étape 5 — Fonction d'éjection d'utilisateurs	61
Configuration — admin_config.json	61
Le réglage sdr_host	62
proxy.py — ce que c'est et quand il est nécessaire	63
Changer de ports après l'installation	63
Commandes manuelles utiles	64
Fichiers journaux	64
Résumé des accès	66
Suppression de messages du chat	66
Messages diffusés sur la cascade	67
Graphiques système	68
Thermal Guard	69
Spot Reporting (autorun FT8/FT4/JS8/WSPR)	72
Dépannage	78
4. Thermal Guard	81
Sommaire	81
1. Ce qu'il fait	81
2. Démarrage rapide	82
3. Comment il décide	82
4. Les quatre modes	83
5. Choisir un mode	85
6. Référence de configuration	86
7. Fonctionnement sans le panneau d'administration	86
8. Activer l'étage throttle sans root	87
9. Fonctionnement sans surveillance	88
10. Le tester avant de lui faire confiance	89
11. Lire le journal	90
12. Lockout et limitation du nombre d'arrêts	90
13. Dépannage	91
14. Le désactiver ou le supprimer	91
Voir aussi	92
5. Limites de connexion	93
Sommaire	93
1. Contre quoi vous vous défendez réellement	93
2. Les trois couches	94
3. Limites par adresse	94
4. Connexions inactives	96
5. La protection au niveau du noyau	96
6. Référence de configuration	97
7. Ce que voit un auditeur refusé	98
8. Lire les journaux et les compteurs	99
9. Choisir vos nombres	99
10. Pièges à connaître	100
11. Tout désactiver	101

6. Plusieurs récepteurs	102
Sommaire	102
1. Comment cela fonctionne	102
2. Deux façons de publier les récepteurs	103
3. Ajouter un second récepteur	103
4. receivers.toml	105
5. Démarrer et arrêter	106
6. Ce que voient les auditeurs	106
7. Le S-mètre au-dessus de 30 MHz	107
8. Sécurité	107
9. Combien de récepteurs tiennent	108
10. Pièges à connaître	108
7. Serveur de statistiques	109
Aperçu	109
Prérequis	109
Méthodes d'installation	109
Configurer votre application PhantomSDR	112
Tester l'installation	114
Gestion du service	115
Installation manuelle (Option 2)	116
Changer le port après l'installation	117
Dépannage	118
Ce que vous verrez	121
Détection de la température	121
Résumé des fonctionnalités	122
Notes de sécurité	122
Support et ressources	122
8. Décodeurs	125
Table des matières	125
1. Comment démarrer un décodeur	125
2. FT8	128
3. FT4-FT2	129
4. JS8	130
5. CW — code Morse	132
6. QRSS Grabber	133
7. WSPR	136
8. FAX HF / WEFAX	137
9. NAVTEX	138
10. FSK / RTTY — y compris PSK31 et Olivia	140
11. SSTV	143
12. Conseils généraux	145
9. RADE README	147
L'installation — la voie courte	147
Table des matières	147
Fonctionnement	148
Vue d'ensemble de l'architecture	149
RADEL contre RADEU	149
Prérequis	150

Installation à la main	150
Déployer les fichiers corrigés	150
Contrôle du sidecar	151
Le port 8074	152
Utiliser RADE dans le navigateur	152
Vérification et débogage	153
Variables d'environnement	154
Fichiers modifiés	155
Résumé du flux du signal	155
Dépannage	156
Mesurer la simultanéité sur votre propre matériel	158
Mettre RADE v1 à jour	160
10. RADE installation générale (Linux)	163
Étape 1 — Paquets système	163
Étape 2 — Paquets Python	164
Étape 3 — Cloner et compiler le dépôt radae	165
Étape 4 — Vérifier la chaîne de décodage	166
Étape 5 — Compiler le frontend de PhantomSDR-Plus	167
Étape 6 — Contrôle du sidecar	167
rade.sh reste fourni pour un usage autonome — faire tourner le sidecar sans le lanceur PhantomSDR-Plus, ou le tester seul. Ses commandes, son journal et les pièges du chien de garde qui vont avec figurent dans Contrôle du sidecar.	167
Étape 7 — Ouvrir le port 8074	167
Étape 8 — Démarrer le serveur	168
Étape 9 — Démarrer RADE	169
Étape 10 — Utiliser RADE dans le navigateur	169
Remarque sur le CPU	169
Mettre RADE à jour par la suite	170
11. Édition des variantes	171
1. Ce que sont les variantes	171
2. Le rôle de chaque fichier	172
3. Faire une modification qui ne concerne que certaines variantes	174
4. Où se trouvent les aiguillages de variante	175
5. Autres réglages à connaître	176
6. Après modification — reconstruire	176
7. Comment vérifier votre modification avant de lui faire confiance	177
12. Guide de l'utilisateur	178
Table des matières	178
Introduction	178
Premiers pas	179
Vue d'ensemble de l'interface	180
Opérations de base	181
Fonctions avancées	184
Modes de démodulation	188
Décodeurs numériques	190
Raccourcis clavier	194
Favoris	194
Utilisation sur mobile	195
Conseils et bonnes pratiques	196

Dépannage	197
Foire aux questions	199
Ressources	200
Annexe : fréquences courantes	201
Glossaire	202
13. Émulation de client KiwiSDR	203
Table des matières	203
1. Ce que reçoit un client Kiwi	203
2. Installer la passerelle	204
3. L'activer	204
4. Référence de configuration	205
5. Connecter un client	205
6. Niveau audio	205
7. Le S-mètre	206
8. Cascade et spectre	207
9. Si quelque chose ne va pas	208
10. Jusqu'où cela a été vérifié	209
14. Diversité de réception	210
Ce que cela fait, et ne fait pas	210
Prise en main rapide	211
Utiliser un WebSDR	212
Lire le panneau	213
Remote SNR trim	214
Choisir un second récepteur	215
Dépannage	215
Limites	216
15. Pilotage du transceiver	217
Ce qu'il vous faut	217
Récepteurs pris en charge	217
Ce qu'il fait, et ce qu'il ne fait pas	218
Démarrage rapide	218
Choisir comment joindre le poste	219
Réglages du port série	220
Synchronisation	220
Modes	221
Largeur de filtre	222
Coupure du son en émission	222
Décalage de fréquence	222
Le menu Rig	223
Linux	223
Windows	223
TCI-CAT sur la page du récepteur	223
Pour les opérateurs de récepteurs	233
Pour les développeurs : l'interface de la page	233
Dépannage	234
Limites connues	234
16. Structure du projet	236
Table des matières	236

Arborescence des répertoires	237
Répertoire racine	250
Code source (src/)	253
Frontend (frontend/)	254
Listes de fréquences (frequencylist/)	259
Fichiers de configuration	259
Système de compilation	261
Dépendances entre fichiers	261
Flux de données	262
Guide de modification des fichiers	262
Chemins importants	263
Opérations courantes sur les fichiers	263
Gestion de versions	264

CHAPITRE 1

README

README.md

PhantomSDR-Plus

Un serveur WebSDR open-source amélioré, doté de fonctions avancées et d'un design futuriste

PhantomSDR-Plus est un fork de PhantomSDR qui fournit un serveur web de radio logicielle (SDR) haute performance, capable de gérer des centaines d'utilisateurs simultanés. Il propose une interface utilisateur améliorée, la prise en charge de plusieurs décodeurs, la visualisation du plan de bandes et la compatibilité avec diverses plateformes matérielles SDR.

Fonctions principales

Performances et évolutivité

- **Multi-utilisateurs** : gère des centaines d'utilisateurs simultanés selon le matériel
- **Taux d'échantillonnage élevé** : prise en charge des SDR jusqu'à 70 MSPS (réel) / 35 MSPS (IQ)
- **Accélération matérielle** : prise en charge d'OpenCL et de CUDA pour un traitement accéléré par GPU
- **Diffusion optimisée** : compression audio FLAC et Opus à faible latence. Le navigateur décode chaque paquet FLAC dès son arrivée, et son tampon de lecture s'adapte à la connexion de l'auditeur en ne gardant que le son nécessaire — en général 100-150 ms du récepteur au haut-parleur sur un téléphone en Wi-Fi

Interface utilisateur

- **Design futuriste** : interface web moderne et responsive
- **Optimisée pour mobile** : interface mobile améliorée pour l'écoute en déplacement
- **Visualisation du plan de bandes** : cascade interactive avec superposition des bandes de fréquences
- **Palettes personnalisables** : plusieurs jeux de couleurs pour la cascade
- **Double S-mètre** : au choix, affichage analogique ou numérique du niveau de signal
- **Boutons de décodeur en une pression** : une rangée Decoders sur le panneau principal, juste sous le sélecteur de modes (FT8, FT4, FT2, JS8, CW, WSPR, FAX, SSTV, NAVTX, RTTY), avec RADEL et RADEU à côté du sélecteur de modes lui-même et repris dans les fenêtres surgissantes des modes et des bandes. Une pression démarre le décodeur et ouvre sa fenêtre, une seconde l'arrête. L'ancienne rangée de largeur de bande a été supprimée.
- **Scanner de canaux** : balaie la bande, ou exactement ce que montre la cascade, et s'arrête sur le premier canal dépassant d'un nombre de dB choisi le plancher de bruit de la bande elle-même, puis repart tout seul dès que ce canal redevient silencieux. Le pas suit le mode et les arrêts tombent sur la grille de canaux (1 kHz SSB, 100 Hz CW, 5 kHz AM ondes courtes, 9/10 kHz ondes moyennes, 9 kHz ondes longues). Un mode *Sauter les vides* lit le spectre de la cascade et saute directement aux signaux, et un bouton de verrouillage retire du balayage un canal toujours occupé. Tout se passe localement dans le navigateur de l'auditeur : le balayage ne déplace le récepteur pour personne d'autre (voir le [Guide de l'utilisateur](#))

- **Diversité de réception** : associe le récepteur à un second situé ailleurs et restitue celui des deux qui a le meilleur signal à cet instant, de sorte qu'un évanouissement d'un côté est couvert par l'autre. Le second peut être un autre PhantomSDR-Plus, un KiwiSDR, un UberSDR ou un WebSDR ; les trois premiers ne demandent rien de plus que le navigateur de l'auditeur, et seul un WebSDR nécessite un petit relais sur votre propre serveur. L'alignement prend une quinzaine de secondes, et un réglage de SNR distant équilibre les deux stations l'une par rapport à l'autre. Chaque auditeur conserve dans son propre navigateur une liste nommée de seconds récepteurs, dans l'ordre qu'il veut, et peut l'exporter dans un fichier puis la réimporter (voir [Diversité de réception](#))
- **Plusieurs récepteurs, une page** : un ordinateur peut faire tourner deux récepteurs ou plus à la fois — par exemple un RX-888 pour les ondes courtes et un RTL-SDR pour le 2 m — et les auditeurs passent de l'un à l'autre par des boutons dans l'en-tête de la page. Chaque récepteur garde sa propre cascade, son chat, ses repères, sa liste d'auditeurs et ses informations de station, et au-dessus de 30 MHz le S-mètre indique des points S VHF ($S9 = -93$ dBm) (voir [Plusieurs récepteurs](#))

Traitement du signal

- **Modes de démodulation multiples** : AM, FM, USB, LSB, CW et davantage. Un décodeur RADE version 1 a également été implémenté.
- **Détection AM synchrone** : meilleure qualité de réception en AM
- **Réduction de bruit** : NR (spectrale), NB (silencieux d'impulsions), NS (suppression du bruit de fond) et AN (notch automatique), tous sur l'écoute et jamais sur les décodeurs
- **Options AGC** : plusieurs modes de contrôle automatique de gain
- **Squelch automatique** : seuil de squelch calculé automatiquement d'après le bruit
- **Squelch à sous-tonalité CTCSS** : l'audio FM ne passe que tant que la sous-tonalité choisie — ou n'importe quelle tonalité normalisée — est reçue ; la tonalité se choisit dans une petite fenêtre à côté du bouton CTCSS, qui l'affiche ensuite
- **Réduction de bruit par IA** : réseau de neurones RNNoise dans le navigateur de l'auditeur, retire le bruit de bande de la parole (modes phonie seulement ; voir le [Guide de l'utilisateur](#))

Fonctions avancées

- **Décodeurs numériques** : FT8, FT4, FT2, JS8, CW, QRSS Grabber, WSPR, FAX HF, SSTV, NAVTEX, FSK/RTTY, PSK31, Olivia et FreeDV RADE, chacun s'exécutant sur son propre thread d'arrière-plan afin que le décodage n'interrompe jamais l'audio (voir [Décodeurs](#))
- **Decoder ID** : nomme le mode numérique présent dans la bande passante et propose le décodeur correspondant en un clic, à partir de la largeur de bande, de l'écart entre tons, de la rapidité de modulation et du rythme des salves mesurés, la fréquence étant prise en compte comme indice supplémentaire — ce qui distingue aussi **FT8** de **JS8**, deux modes au signal identique. Il ne fait que suggérer, ne bascule jamais de lui-même, et préfère se taire plutôt que deviner quand le signal est trop faible. Le fading ne le réduit pas au silence : quand le QSB découpe en fragments une émission WSPR de deux minutes, il nomme toujours le mode, lisant la cadence sur tout l'historique d'écoute, et abaisse la confiance affichée pour signaler une preuve plus faible. Entre 400 Hz et 2700 Hz environ, la position dans la bande passante est sans importance ; au-delà, un bouton ambre **Recentre & retry** apparaît, qui déplace l'accord une fois et relance la mesure. Lorsque vous acceptez une suggestion, le récepteur est réglé pour ce mode : la fréquence accordée passe au milieu de la cascade, la vue s'établit sur environ 100 kHz autour d'elle, et la bande latérale ainsi que la bande passante deviennent celles avec lesquelles ce mode se travaille — la sous-bande de 3 kHz pour la famille FT8, 1350–1650 Hz pour WSPR, ± 250 Hz et CW pour la télégraphie, et ainsi de suite. Désactivé par défaut, et environ 0,5 % d'un cœur en fonctionnement (voir [Décodeurs](#))
- **Préréglages des modes numériques** : FT8, FT4, FT2, JS8 ou WSPR règlent à la fois la bande latérale et la bande passante

- **Panneau de conversation JS8** : les messages en plusieurs trames sont réassemblés en phrases complètes
- **Tableau de bord de statistiques** : statistiques du serveur et des utilisateurs en temps réel
- **Liste des utilisateurs connectés** : toutes les personnes à l'écoute, avec un bouton d'accord sur chaque ligne. Votre propre session est marquée **you**, aussi bien dans la fenêtre de l'interface que lorsque `users.html` est ouvert comme page autonome
- **Report automatique des spots** : un démon autorun décode FT8, FT4 et WSPR sur le serveur et téléverse les spots vers PSK Reporter et WSPRnet, avec deux compteurs dans le panneau d'administration — des tuiles par décodeur pour l'exécution en cours et un total depuis toujours à côté de chaque case bande/mode
- **Graphiques du serveur** : une page **Graphiques** du panneau d'administration traçant la fréquence du processeur, la charge, la température et les utilisateurs connectés sur les 15 dernières minutes à 24 heures (voir [Panneau d'administration](#))
- **Protection thermique** : un garde intégré au panneau d'administration arrête le serveur si le processeur surchauffe et le relance une fois refroidi, avec des seuils déduits du seuil critique de votre propre processeur et sans aucune hypothèse sur votre façon de démarrer ou d'arrêter le serveur ; livrée en mode journalisation seule, elle ne fait rien tant que vous ne l'activez pas (voir [Thermal Guard](#)) Pour un récepteur non surveillé, il est **fortement recommandé** de faire tourner le panneau en unité systemd : le garde vit à l'intérieur, sinon un redémarrage emporte la protection avec lui (voir [Panneau d'administration](#)).
- **Émulation des clients KiwiSDR** : une passerelle facultative qui répond aussi au protocole KiwiSDR, sur le même hôte et le même port, si bien que les logiciels Kiwi comme **AetherSDR** et `kiwiclient` se connectent directement au récepteur — avec un accord réel, la cascade et un S-mètre à la même échelle que celui de la page web. Inactive tant que `[kiwi_emulation] enabled = true` n'est pas dans votre configuration (voir [Installation](#))
- **Pilotage du transceiver (CAT)** : garde votre propre poste et la fenêtre du récepteur sur la même fréquence, le même mode et la même largeur de filtre, dans un sens ou dans les deux — par le menu Rig de [Desktop PhantomSDR+](#) (Icom, Yaesu, Kenwood, Elecraft, FlexRadio et QRP Labs en direct, tout autre poste via Hamlib, ou flrig) ou avec le CATsync Tool for WebSDRs dans un navigateur. Les pages proposent des fonctions `catsync_*` pour la fréquence, le mode, la largeur de filtre et la coupure du son. Cela fonctionne avec les récepteurs PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR et UberSDR (voir [Pilotage du transceiver](#))
- **Système de favoris** : import/export de fréquences favorites
- **Raccourcis clavier** : navigation et commande efficaces
- **Prise en charge de la molette** : accord en fréquence intuitif
- **Annuaire WebSDR** : intégration avec <https://sdr-list.xyz>

Matériel pris en charge

PhantomSDR-Plus prend en charge une large gamme de récepteurs SDR :

Appareil	Taux d'échantillonnage	Format	Interface
RX888 MK2	Jusqu'à 64 MHz	16 bits	Native
RTL-SDR	Jusqu'à 3.2 MHz	8 bits	rtl_sdr
HackRF One	Jusqu'à 20 MHz	8 bits	hackrf_transfer
Airspy HF+ / Discovery	Jusqu'à 912 kHz	16 bits	SoapySDR (SoapyAirspyHF) + rx_sdr
SDRplay RSP1A	Jusqu'à 10 MHz	16 bits	SoapySDR (libmirisdr-5 + SoapyMiri, ou l'API SDRplay) + rx_sdr
RigExpert Fobos SDR	50 MHz (HF, échantillonnage direct) / jusqu'à 20 MHz (RF)	16 bits / flottant 32 bits	SoapySDR (SoapyFobosSDR) + rx_sdr
Autres appareils	Variable	Divers	SoapySDR/rx_tools

Démarrage rapide

Configuration système requise

Minimum :

- Ubuntu 22.04 LTS (recommandé) ou Fedora
- Processeur 2 cœurs
- 4 Go de RAM
- 10 Go d'espace disque

Recommandé :

- Ubuntu 24.04 LTS
- Processeur 4 cœurs ou plus (Ryzen 5 2600 ou Intel i5-6500T, ou mieux)
- 8 Go de RAM
- GPU compatible OpenCL (facultatif mais vivement recommandé)
- Stockage SSD

Installation

```
# Cloner le dépôt
git clone --recursive https://github.com/sv1btl/PhantomSDR-Plus
cd PhantomSDR-Plus

# Rendre les scripts exécutables
chmod +x *.sh

# Lancer l'installation automatique
./install.sh
```

Remarque : après avoir exécuté `install.sh`, redémarrez votre terminal avant de continuer.

PhantomSDR-Plus tourne déjà chez vous ? Ne le réinstallez pas : mettez-le à jour. Récupérez l'outil une fois et lancez-le ; votre configuration, vos marqueurs, le mot de passe d'administration, la liste des fréquences et l'historique du chat ne sont jamais touchés, et ce que vous avez modifié vous est soumis au lieu d'être écrasé :

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

Cette dernière ligne ne fait que *signaler* ce qui changerait, sans rien écrire ; `./update.sh --apply` l'exécute. Les détails sont dans le chapitre *Mise à jour de PhantomSDR-Plus*. Repasser l'installateur sur une station en service n'est nécessaire que si la recompilation échoue faute de paquets système.

Utilisez `install_fedora.sh` sur Fedora, `install_arch.sh` sur Arch ou `install_opensuse.sh` sur openSUSE Tumbleweed : ils font le même travail avec un autre gestionnaire de paquets. Toutes les versions de Debian et d'Ubuntu sont couvertes par `install.sh` lui-même.

Rien n'est à installer à la main au préalable. Le script installe les dépendances de compilation et Node.js, compile le backend, compile le pilote du récepteur que vous choisissez (et, pour le RX888, les règles udev avec), ouvre `site_information.json` pour que vous le remplissiez, construit le frontend, installe OpenCL quand le matériel le permet, puis met en place le **panneau d'administration**, le **décodeur FreeDV RADE** et le **serveur de statistiques** — les trois par défaut, répondez `n` pour en sauter un — et se termine par un `recompile.sh` complet. Il est interactif d'un bout à l'autre et peut durer de vingt minutes à plus d'une heure.

Pour le détail étape par étape, voir [INSTALLATION.md](#) → [Ce que fait l'installateur](#). Pour des instructions d'installation détaillées, voir [INSTALLATION.md](#).

Mesures de performance

Matériel	Taux d'échantillonnage	Charge CPU	Capacité utilisateurs
Ryzen 5 2600 (tous cœurs)	64 MHz (32 MHz IQ)	38-40 %	100+ utilisateurs
AMD RX 580 (GPU)	64 MHz (32 MHz IQ)	28-35 %	100+ utilisateurs
Intel i5-6500T (avec OpenCL)	60 MHz (30 MHz IQ)	10-12 %	100+ utilisateurs

Remarque : la charge CPU supplémentaire par utilisateur est minime (< 1 % par utilisateur) lorsque l'accélération matérielle est activée.

Utilisation

Fonctionnement de base

1. **Configurez votre SDR** : modifiez le fichier de configuration approprié (par ex. `config-rtl.toml`)
2. **Mettez à jour les informations du site** : modifiez `frontend/site_information.json`
3. **Démarrez le serveur** : lancez le script de démarrage approprié : `bash ./start-rtl.sh` # Pour RTL-SDR `./start-rspla.sh` # Pour SDRplay RSP1A `./start-airspyhf.sh` # Pour Airspy HF+ `./start-fobos-hf.sh` # Pour RigExpert Fobos SDR, HF1/HF2 (0-25 MHz) `./start-fobos.sh` # Pour RigExpert Fobos SDR, RF (25-6000 MHz) `./start-hackrf.sh` # Pour HackRF One `./start-rx888mk2.sh` # Pour RX888 MK2
Le RSP1A, l'Airspy HF+ et le Fobos passent par SoapySDR et `rx_sdr` ; l'installateur prépare leur pilote quand vous les choisissez (options 3, 6 et 5), ou lancez `./setup-rspla.sh`, `./setup-airspyhf.sh` ou `./setup-fobos.sh` sur une station déjà installée. Voir [INSTALLATION.md](#). Le HackRF One (option 7) n'a

besoin que du paquet `hackrf` de la distribution : `./setup-hackrf.sh`. Chaque script de démarrage est **autonome** : il arrête toute instance en cours, lance le récepteur + `spectrumserver`, **passe en arrière-plan** (il survit à la fermeture du terminal) et exécute un **chien de garde** qui relance automatiquement la chaîne si elle s'arrête. La progression est journalisée dans `logwebsdr.txt` — suivez-la avec `tail -f logwebsdr.txt`. Relancer un script de démarrage équivaut à un redémarrage propre, et un verrou `flock` garantit qu'un seul récepteur fonctionne à la fois. Facultatif : `SPECTRUM_CORES=0-3` pour épingler des cœurs CPU, `RX_ARGS=""` pour remplacer les arguments du récepteur sans modifier le fichier (`RX888_ARGS` pour le RX888), `RX_DRIVER=miri|sdrplay` pour choisir le pilote du RSP1A.

4. **Accédez à l'interface** : ouvrez votre navigateur sur `http://localhost:PORT` (le port par défaut dépend de la configuration)

Arrêt du serveur

Un seul script d'arrêt partagé fonctionne pour **tous** les récepteurs — il arrête d'abord le chien de garde (pour qu'il ne puisse pas relancer le service), puis le récepteur et `spectrumserver` :

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

Configuration

Fichiers de configuration essentiels

1. `config-[appareil].toml` – configuration du serveur et du SDR
2. Paramètres du serveur (port, threads, racine HTML)
3. Paramètres d'entrée (taux d'échantillonnage, taille FFT, fréquence)
4. Options d'enregistrement WebSDR
5. Paramètres de compression audio
6. Après modification, redémarrez le serveur.
7. `frontend/site_information.json` – informations publiques du site
8. Détails de l'opérateur (indicatif, e-mail, localisation)
9. Informations sur le matériel et l'antenne
10. Paramètres de région et de largeur de bande
11. Activation/désactivation du chat
12. Après modification, exécutez `recompile.sh` dans un terminal.
13. `markers.json` – marqueurs de fréquence et plan de bandes
14. `frontend/src/bands-config.js` – configuration du plan de bandes Ce fichier se trouve dans `frontend/src/bands-config.js` et définit les bandes créées par le SysOp.
15. Vous verrez quelque chose comme ce qui suit, et vous êtes libre de le modifier à votre guise : ``
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 `` ` Où :
```

18. **ITU** est la région dans laquelle se trouve le serveur,
19. **min et max** sont les limites de luminosité de la cascade pour la bande donnée,
20. **initFreq** est la fréquence initiale sur laquelle s'accorder lorsque la bande est sélectionnée,
21. **publishBand** définit si la bande est affichée : 1 pour les bandes amateurs, 2 pour les bandes de radiodiffusion,
22. **startFreq et endFreq** définissent les limites de la bande,
23. **stepi** est le pas par défaut de la molette pour la bande, et **modes** est le mode par défaut préféré pour la bande.
24. Après avoir effectué des modifications, **exécutez recompile.sh dans un terminal** depuis le dossier PhantomSDR-Plus.

Pour des exemples de configuration, voir les fichiers d'exemple inclus :

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

Plancher de la cascade — `waterfall.sh`

`waterfall.sh` (à la racine du dépôt) modifie le **niveau minimal de cascade par défaut**, en dB — c'est-à-dire la valeur à partir de laquelle la cascade démarre lors d'une nouvelle visite.

Cette valeur figure dans deux fichiers source, à cinq endroits différents (`frontend/src/waterfall.js` → `this.minWaterfall`, et `frontend/src/App.svelte` → le `let min_waterfall` initial, les deux cas de réinitialisation « min » et la valeur par défaut des favoris). Les éditer à la main se rate facilement : le script s'en charge pour vous. Il n'utilise jamais de numéros de ligne figés — chaque emplacement est repéré par motif, si bien qu'il continue de fonctionner après une mise à jour des sources.

- Une valeur **plus élevée** (par ex. `-15`) donne une cascade **plus sombre**.
- Une valeur **plus basse** (par ex. `-45`) donne une cascade **plus lumineuse**.

```
./waterfall.sh          # interactif – affiche les valeurs actuelles et demande la nouvelle
./waterfall.sh -v -15   # fixe la valeur sans poser de question
./waterfall.sh -v -15 -y # ... et saute toutes les confirmations
./waterfall.sh -s       # affiche seulement les valeurs actuelles, sans rien changer
```

Avant de modifier, il crée une sauvegarde horodatée des deux fichiers (`*.bak-AAAAmmjj-HHMMSS`) et affiche la commande de restauration. Seuls les littéraux égaux à la valeur en cours d'utilisation sont touchés : les autres nombres de ces fichiers ne peuvent donc jamais être réécrits par accident. Si la nouvelle

valeur n'apparaît pas ensuite, le script restaure lui-même les sauvegardes et sort en erreur. Il avertit également lorsque les deux fichiers sont désynchronisés et règle les deux.

- Après la modification, le frontend doit être reconstruit — le script propose d'exécuter `recompile.sh` pour vous (avec `-y` il saute la reconstruction : lancez alors `./recompile.sh` vous-même le moment venu).

Personnalisation

Changer l'image de fond

Remplacez `frontend/src/assets/background.jpg` par l'image de votre choix (en conservant le même nom de fichier).

- Après modification, exécutez `recompile.sh` dans un terminal.

Choix du codec audio

Choisissez entre FLAC et Opus dans votre configuration `.toml` :

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

- Après modification, redémarrez le serveur.

Étalonnage du S-mètre

Les deux instruments sont volontairement sur des **chaînes séparées** ; ils se règlent donc avec deux réglages différents.

Les deux réglages

Tous deux se trouvent dans la section `[input]` du fichier de configuration avec lequel le serveur démarre (`config.toml`, `config-rx88mk2.toml`, etc.) :

```
[input]
smeter_offset=5          # barre numérique uniquement
analog_smeter_offset=5   # aiguille analogique uniquement
```

Ils sont lus dans `src/websocket.cpp` et envoyés au navigateur dans la première trame `basic_info` : **un redémarrage du serveur suffit** — aucune recompilation du frontend n'est nécessaire.

Chaîne analogique

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

- Le gain visuel de 1,1 s'applique *après* l'offset, avec pivot à -130 dBm. Une unité d'`analog_smeter_offset` déplace donc la lecture de 1,1 dB. Pour décaler l'affichage de X dB : `analog_smeter_offset ~= 0,91 x X`. La valeur TOML est lue comme un entier : seuls des pas entiers sont possibles.
- La loi de l'aiguille (`powerFromDbm` dans `frontend/src/lib/SMeterAnalog.svelte`) place **S9 à -73 dBm**, à 60 unités d'aiguille sur 100, avec une courbe `pow(...,0,6)` de -130 à -73 sous S9 et une courbe `pow(...,0,8)` de -73 à -13 (S9+60) au-dessus.
- Ce n'est **pas** une échelle linéaire de 6 dB par point S ; le point d'étalonnage pertinent est donc S9 — les graduations sous S9 ne suivront pas un générateur par pas de 6 dB.

Chaîne numérique

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

- Noter le $/150 \times 100$: l'échelle numérique est **comprimée aux 2/3** et ne voit jamais ni `analog_smeter_offset` ni le gain visuel de 1,1. C'est pourquoi la barre et l'aiguille ne coïncident jamais exactement — c'est voulu.
- 35 segments sur 100 unités : un segment vaut 2,86 unités d'offset, soit environ 4,3 dB bruts. `smeter_offset=3` correspond donc à peu près à un segment.
- `DIGITAL_BAR_TRIM` dans `frontend/src/lib/SMeterDigital.svelte` (actuellement 0) décale la barre par segments entiers. Le modifier touche au code source et **exige** une recompilation du frontend.

Les fenêtres LED sont pilotées par l'analogique dans les deux instruments

Les fenêtres dBm / dBμV / SNR / NF des **deux** instruments lisent la valeur calibrée analogique et ajoutent un `VISUAL_DBM_OFFSET` fixe de 5, codé en dur dans `SMeterAnalog.svelte` et `SMeterDigital.svelte`. Par conséquent :

- `smeter_offset` ne change **aucun** nombre dans les fenêtres, seulement la longueur de la barre.
- dBμV = dBm + 107 (50 ohms), et NF = dBm - SNR par construction, donc NF suit automatiquement.
- Si l'on veut de bons **nombres** sans toucher à la position de l'aiguille, le réglage est `VISUAL_DBM_OFFSET` — mais c'est une constante du code source, il faut recompiler et la modifier dans **les deux** fichiers pour rester cohérent.

Procédure

1. Injecter un niveau connu à l'entrée d'antenne (générateur à -73 dBm = S9), en SSB/CW avec une bande passante stable. Les lectures dépendent de la bande passante et de l'AGC : fixer d'abord le mode et la largeur.
2. Relever ce qu'affiche la fenêtre dBm analogique, puis poser `analog_smeter_offset` $\approx 0,91 \times (-73 - \text{valeur affichée})$. Redémarrer le serveur et vérifier à nouveau. Une itération suffit ; l'aiguille doit se poser sur S9.
3. Passer à l'instrument numérique et régler `smeter_offset` pour que la longueur de barre place S9 où vous le souhaitez — environ 3 unités par segment. Cette étape est purement cosmétique et n'affecte pas les valeurs affichées.
4. Sans générateur, une bande calme avec une balise de niveau connu constitue un bon substitut, ou bien on ancre simplement le plancher de bruit à une valeur plausible (par ex. -120 dBm sur 20 m avec une antenne correcte) — en acceptant que cela étalonne toute la chaîne, gain d'antenne compris, et pas seulement le récepteur.

Variantes de l'interface graphique

Les quatre variantes de l'interface (S-mètre analogique/numérique x disposition v1/v2) tiennent dans une seule construction. Les visiteurs passent de l'une à l'autre avec le menu ☸ en haut à droite — rien n'est rechargé, l'audio, la cascade et tout décodeur en cours continuent — et le choix est conservé dans chaque navigateur.

- Exécutez `recompile.sh` dans un terminal pour définir la variante de *départ*, celle que voit un visiteur qui arrive pour la première fois.
- **EDITING_VARIANTS.md** – comment modifier les variantes du frontend (S-mètre et disposition) et les reconstruire

Documentation

- **INSTALLATION.md** – guide d'installation complet pour les opérateurs système
 - **ADMIN_PANEL_SETUP.md** – guide d'installation pour les administrateurs système
 - **USER_GUIDE.md** – guide de l'utilisateur final pour l'exploitation du WebSDR
 - **THERMAL_GUARD.md** - Manuel du sysop pour la protection contre la surchauffe du processeur : les quatre modes et ce qu'il faut faire pour chacun
 - **CONNECTION_LIMITS.md** - Manuel du sysop pour les limites de connexion : protéger un récepteur public contre les afflux de connexions
 - **MULTI_RECEIVER.md** - Manuel du sysop pour plusieurs récepteurs sur un ordinateur : instances, `receivers.toml`, le sélecteur de récepteur et `start-all.sh`
 - **PROJECT_STRUCTURE.md** – structure des répertoires et organisation du code
 - **EDITING_VARIANTS.md** – comment modifier les variantes du frontend (S-mètre et disposition) et les reconstruire
-

Liste WebSDR en ligne

Enregistrez votre WebSDR sur l'annuaire officiel : <https://sdr-list.xyz>

Définissez `register_online=true` dans votre fichier de configuration `.toml` pour un enregistrement automatique.

Dépannage

Problèmes courants

OpenCL ne fonctionne pas

- Vérifiez que les pilotes sont correctement installés
- Contrôlez avec la commande `clinfo`
- Voir `INSTALLATION.md` pour la configuration détaillée d'OpenCL

Problèmes de latence audio

- Gardez FLAC comme codec : c'est le plus faible en latence, alors qu'Opus retient jusqu'à 20 ms par paquet
- Ouvrez la page en ajoutant `?diag=1` à son adresse (par exemple `/mobile?diag=1`) pour voir le retard réparti entre paquet, tampon et sortie ; le tampon se réduit tout seul en une ou deux minutes après un incident réseau
- Si le son se coupe, montez le tampon d'un cran (voir le Guide de l'utilisateur, *Tampon audio et retard*)
- Assurez-vous de disposer de ressources CPU/GPU suffisantes

Échecs de compilation

- Vérifiez que toutes les dépendances sont installées
 - Essayez de nettoyer le répertoire de build : `rm -rf build && meson setup build`
 - Recherchez d'éventuels conflits de versions de bibliothèques
-

Contribuer

Les contributions sont les bienvenues ! Il s'agit d'un fork indépendant proposant des fonctions supplémentaires. Merci de :

1. Forker le dépôt
 2. Créer une branche de fonctionnalité
 3. Valider vos modifications
 4. Pousser la branche
 5. Créer une Pull Request
-

Licence

Ce projet est distribué sous licence GNU General Public License v3.0 – voir le fichier [LICENSE](#) pour les détails.

Auteurs et remerciements

- **SV1BTL et SV2AMK** – développement et améliorations de PhantomSDR-Plus
 - Basé sur le projet PhantomSDR d'origine
-

Liens

- **Démo en direct** : <http://phantomsdr.no-ip.org:8900/>
 - **Annuaire WebSDR** : <https://sdr-list.xyz>
 - **Dépôt GitHub** : <https://github.com/sv1btl/PhantomSDR-Plus>
-

Assistance



- **Problèmes** : signalez les bogues via [GitHub Issues](#)
 - **E-mail** : contactez sv1btl@otenet.gr
-

Conseils de performance

1. **Activez OpenCL/CUDA** pour l'accélération GPU – réduit considérablement la charge CPU
 2. **Utilisez un stockage SSD** pour de meilleures performances d'E/S
 3. **Allouez suffisamment de RAM** – au moins 8 Go recommandés
 4. **Optimisez la taille de la FFT** – une FFT plus grande = meilleure résolution mais plus de CPU
 5. **Envisagez un GPU dédié** – AMD ou NVIDIA compatible OpenCL
-

73 de SV1BTL & SV2AMK

Pour des instructions de configuration détaillées, voir [INSTALLATION.md](#) Pour le guide d'utilisation destiné à l'utilisateur final, voir [USER_GUIDE.md](#)

CHAPITRE 2

Guide d'installation

INSTALLATION.md

Guide d'installation de PhantomSDR-Plus pour les opérateurs système

Ce guide complet vous accompagne dans l'installation et la configuration de PhantomSDR-Plus sur votre serveur.

Table des matières

1. Configuration système requise
2. Préparation avant installation
3. Installation de PhantomSDR-Plus — le script et ce qu'il fait
4. Rapporteur de spots autorun (FT8/FT4/WSPR)
5. Émulation des clients KiwiSDR (facultatif)
6. Configuration
7. Configuration propre à chaque appareil SDR
8. Tests et vérification
9. Mise en place du démarrage automatique
10. Protection thermique du processeur
11. Dépannage

Pour référence seulement — l'installateur fait déjà tout cela pour vous. Lisez ces sections si vous êtes sur une distribution qu'aucun des scripts ne couvre, ou pour réparer une étape à la main :

- Installation des dépendances
 - Installation de Node.js et npm
 - Installation d'OpenCL
 - Compiler à la main
 - Installation du codec audio Opus
-

Configuration système requise

Systèmes d'exploitation pris en charge

Principal (recommandé) :

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

Alternative :

- Fedora (dernière version stable)
- Arch Linux (rolling release)

- openSUSE Tumbleweed (pas Leap — voir la note plus bas)

Configuration matérielle requise

Configuration minimale :

- Processeur : double cœur (2 GHz ou plus)
- RAM : 4 Go
- Stockage : 10 Go d'espace libre
- Réseau : connexion 100 Mbit/s

Configuration recommandée :

- Processeur : quadricœur ou mieux (Ryzen 5 2600, Intel i5-6500T ou supérieur)
- RAM : 8 Go ou plus
- Stockage : SSD de 20 Go ou plus
- GPU : AMD/NVIDIA compatible OpenCL (vivement recommandé)
- Réseau : connexion 1 Gbit/s

Configuration hautes performances :

- Processeur : 6 cœurs ou plus (Ryzen 7, Intel i7 ou mieux)
- RAM : 16 Go ou plus
- Stockage : SSD NVMe
- GPU : GPU dédié compatible OpenCL/CUDA
- Réseau : 1 Gbit/s ou mieux

Préparation avant installation

1. Mettez votre système à jour

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

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

2. Vérifiez la version d'Ubuntu

```
lsb_release -a
```

La sortie attendue doit indiquer : Ubuntu 24.04 LTS

3. Vérifiez l'espace disque disponible

```
df -h
```

Assurez-vous d'avoir au moins 10 Go libres dans votre répertoire personnel.

4. Vérifiez les informations du processeur

```
lscpu
```

Notez le nombre de cœurs/threads pour optimiser la configuration.

Installation des dépendances

IMPORTANT

Vous n'avez pas besoin de cette section pour une installation normale. `./install.sh` — ou l'installateur de votre distribution — fait tout cela pour vous ; voir [Ce que fait l'installateur](#). Ce qui suit est une référence pour une installation manuelle, pour une distribution qu'aucun script ne couvre, ou pour réparer un élément à la main.

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
```

Vérifiez l'installation

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

Installation de Node.js et npm

IMPORTANT

Vous n'avez pas besoin de cette section pour une installation normale. `./install.sh` — ou l'installateur de votre distribution — fait tout cela pour vous ; voir [Ce que fait l'installateur](#). Ce qui suit est une référence pour une installation manuelle, pour une distribution qu'aucun script ne couvre, ou pour réparer un élément à la main.

PhantomSDR-Plus a besoin de Node.js pour compiler l'interface. Nous utiliserons NVM (Node Version Manager) pour l'installation.

1. Installez NVM

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

2. Chargez NVM

IMPORTANT : fermez et rouvrez votre terminal, ou exécutez :

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

3. Vérifiez l'installation de NVM

```
nvm --version
```

Sortie attendue : `0.40.4` ou similaire

4. Installez 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. Facultatif : installez d'autres versions 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
```

Installation d'OpenCL (facultatif mais recommandé)

IMPORTANT

Vous n'avez pas besoin de cette section pour une installation normale. `./install.sh` — ou l'installateur de votre distribution — fait tout cela pour vous ; voir [Ce que fait l'installateur](#). Ce qui suit est une référence pour une installation manuelle, pour une distribution qu'aucun script ne couvre, ou pour réparer un élément à la main.

OpenCL améliore considérablement les performances en déportant les calculs de FFT sur le GPU. Cette section traite des graphiques intégrés Intel. Pour les GPU AMD/NVIDIA, reportez-vous à la documentation du fabricant.

Processeur Intel avec graphiques intégrés

1. Installez les composants OpenCL de base

```
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. Téléchargez 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. Installez le runtime OpenCL d'Intel

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

En cas d'erreurs de dépendances :

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

4. Installez le chargeur ICD OpenCL

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

En cas d'erreurs :

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

5. Vérifiez l'installation d'OpenCL

```
sudo clinfo
```

Vous devriez voir des informations sur votre plateforme et vos appareils OpenCL. Cherchez :

- Nombre de plateformes : 1 (ou plus)
- Nom de la plateforme : Intel(R) OpenCL (ou similaire)
- Type d'appareil : GPU ou CPU

6. Redémarrez

```
sudo reboot
```

OpenCL sur GPU AMD

Pour les GPU AMD, installez 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-ocl rocm-clinfo

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

# Reboot
sudo reboot

# Verify
clinfo
```

OpenCL sur GPU NVIDIA

Pour les GPU NVIDIA, installez 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
```

Installation de PhantomSDR-Plus

Cloner le dépôt, rendre les scripts exécutables, lancer l'installateur

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

⚠ **IMPORTANT** : une fois l'installateur terminé, **redémarrez votre terminal** — Node.js et Rust fraîchement installés ne sont dans votre PATH qu'après cela.

PhantomSDR-Plus tourne déjà chez vous ? Ne le réinstallez pas : mettez-le à jour. Récupérez l'outil une fois et lancez-le ; votre configuration, vos marqueurs, le mot de passe d'administration, la liste des fréquences et l'historique du chat ne sont jamais touchés, et ce que vous avez modifié vous est soumis au lieu d'être écrasé :

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

Cette dernière ligne ne fait que *signaler* ce qui changerait, sans rien écrire ; `./update.sh --apply` l'exécute. Les détails sont dans le chapitre *Mise à jour de PhantomSDR-Plus*. Repasser l'installateur sur une station en service n'est nécessaire que si la recompilation échoue faute de paquets système.

Prenez le script qui correspond à votre système. Ils font le même travail et posent les mêmes questions ; seul le gestionnaire de paquets diffère :

Système	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 seul installateur pour toutes les versions de Debian et d'Ubuntu. `install.sh` lit `/etc/os-release` et la version de Boost installée puis s'adapte de lui-même ; les anciens `install_ubuntu22.sh`, `install-Deb12.sh` et `install_ubuntu26.sh` ont disparu. Il définit `DEBIAN_FRONTEND=noninteractive`, afin que le `tzdata` qui arrive avec `python3-matplotlib` ne puisse pas interrompre l'exécution pour demander votre fuseau horaire, puis avaler la réponse destinée à la question suivante. Sur Jammy il appelle directement `install_rade_ubuntu22.sh`, car le `python3-websockets` de Jammy est en 10.1 alors que RADE exige 11.0 ou plus récent. Il sait où se trouve `intel-onekl-icd` sur chaque version : dans les dépôts sur 22.04 et 26.04, dans `non-free` sur Debian 12, et dans le dépôt graphique d'Intel sur 24.04 et Debian 13. Et là où Boost est en 1.87 ou plus récent, le correctif des en-têtes `websocketpp` cesse d'être facultatif : l'installateur vérifie qu'il a bien été appliqué et refuse de compiler sans lui. Aucun compilateur plus récent n'est jamais nécessaire : le GCC d'origine accepte `-std=c++23` sur les cinq versions, n'installez donc pas `gcc-12` pour cela.

L'installation se déroule en 17 étapes clairement numérotées, et chaque point où elle vous attend est encadré par un bandeau  **VOTRE SAISIE EST REQUISE**, pour qu'une question ne se confonde pas avec l'affichage qui défile. Les sept questions sont listées dès le départ, avant toute installation.

`PHANTOM_NONINTERACTIVE=1` y répond toutes par leurs valeurs par défaut ; voyez l'en-tête de `install.sh` pour les variables `PHANTOM_*`.

Chaque exécution écrit `install.txt`. Quand l'installateur se termine — ou s'arrête en cours de route — il écrit un rapport dans `install.txt`, à la racine du dossier `PhantomSDR-Plus` : le résultat, chacune des 17 étapes en OK / SKIPPED / PARTIAL / FAILED, ce qu'il a détecté (distribution, Boost, compilateur, Node.js), quels composants ont été installés, et chaque avertissement rencontré. Une exécution qui échoue laisse un rapport qui s'arrête à l'étape fautive, avec la raison et la mention que l'installateur peut être relancé sans risque. C'est le premier fichier à lire quand quelque chose n'a pas marché, et le premier à joindre à un rapport de bogue. Chaque exécution l'écrase : gardez-en une copie si vous voulez comparer deux installations.

Ubuntu 26.04, Arch et openSUSE Tumbleweed se compilent, mais n'ont pas été éprouvés en exploitation.

Tous trois embarquent un Boost plus récent que 1.87, qui a supprimé l'API `io_service` pour laquelle le `websocketpp` 0.8.2 embarqué a été écrit. Les en-têtes corrigés que l'installateur copie comblent cet écart (`io_context`, `executor_work_guard`, `boost::asio::post`, le `resolver` moderne), et là où Boost est en 1.87 ou plus récent l'installateur traite ce correctif comme obligatoire et non facultatif : il vérifie que les copies sont en place et refuse de compiler sans elles. Une installation complète, pilote du récepteur et tous les composants optionnels compris, a été vérifiée de bout en bout en conteneurs sur Boost 1.90 (Ubuntu 26.04), 1.91 (openSUSE Tumbleweed) et 1.92 (Arch). Cela prouve que le serveur compile et démarre — pas qu'il sert un récepteur pendant des heures. Considérez les trois comme non testés en production tant que personne n'a fait de retour.

openSUSE signifie ici Tumbleweed. C'est là que `install_opensuse.sh` est vérifié. Leap 15.6 ne fonctionne pas : ses dépôts ne contiennent aucun `liquid-dsp-devel`, dont le backend a besoin, et Boost n'existe que sous des noms de paquets versionnés. Prendre en charge Leap imposerait d'ajouter des dépôts OBS tiers ; c'est donc hors périmètre pour l'instant.

Ce que fait l'installateur

Rien n'est à préparer à la main au préalable : aucune liste de paquets à coller, aucun Node.js à récupérer, aucun paquet OpenCL à chercher. Il se déroule en 19 étapes numérotées et s'arrête pour vous poser jusqu'à dix questions, chacune encadrée par un bandeau « VOTRE SAISIE EST REQUISE » : restez au clavier, ou bien définissez `PHANTOM_NONINTERACTIVE=1` et laissez-le répondre à tout avec ses valeurs par défaut (voir plus bas) — ne le laissez donc pas sans surveillance — et comptez d'une vingtaine de minutes à bien plus d'une heure selon la machine et le nombre d'extras conservés.

#	Étape	Ce qui vous est demandé
1	Énumère les services PhantomSDR-Plus en cours d'exécution — panneau d'administration, proxy inverse, serveur de statistiques, récepteur — et propose de les arrêter avant de toucher à quoi que ce soit. Seul un script de lancement en cours d'exécution compte comme récepteur ; un script simplement ouvert dans un éditeur n'est pas touché.	confirmation, oui par défaut
2-6	Détecte la distribution et installe toutes les dépendances de compilation (compilateur, meson/ninja, FFTW, Boost, FLAC, Opus, liquid-dsp, zlib/zstd, libcurl ...), en installant Node.js 22 via nvm si le système n'en a pas ou en a un trop ancien	rien
7	Compile le backend avec meson	rien
8	Compile le pilote de votre récepteur — RX888 MkII / RX888, RTL-SDR (le Blog V4 est demandé à part), SDRplay RSP1A, RigExpert Fobos SDR, Airspy HF+, HackRF One ou aucun. Avec le RX888, les règles udev sont installées aussi , pour que le serveur n'ait jamais besoin de <code>sudo</code> pour l'appareil. Le RSP1A, le Fobos et l'Airspy HF+ passent par SoapySDR : leur choix lance <code>setup-rsp1a.sh</code> , <code>setup-fobos.sh</code> ou <code>setup-airspyhf.sh</code> , qui compile le pilote et <code>rx_sdr</code> et installe aussi une règle udev (voir Récepteurs sur SoapySDR (RSP1A, Fobos, Airspy HF+)) Le HackRF One n'a pas besoin de SoapySDR : son choix lance <code>setup-hackrf.sh</code> , qui installe le paquet <code>hackrf</code> de la distribution et une règle udev.	quel SDR vous avez
9	Ouvre <code>frontend/site_information.json</code> dans votre éditeur	indicatif, locator, matériel, antenne — ne sautez pas cette étape
10-11	Installe les dépendances du frontend et construit les pages bureau et /mobile	rien
12	Installe OpenCL en choisissant le fournisseur d'après le matériel trouvé (GPU Intel / AMD / NVIDIA, ou le runtime CPU x86). S'il n'y a aucun appareil compatible, il le dit et continue	confirmation, oui par défaut
13-15	Installe le panneau d'administration , le décodeur FreeDV RADE V1 et le serveur de statistiques — les trois par défaut	confirmation pour chacun, oui par défaut ; chacun pose ses propres questions
16	Réapplique les cinq en-têtes websocketpp corrigés par-dessus le sous-projet meson et vérifie qu'ils sont bien en place. Trois d'entre eux constituent le travail de compatibilité Boost ≥ 1.87 , sans lequel le backend ne compile pas avec Boost 1.90 ; les deux autres sont des modifications propres au projet, dont le correctif nécessaire à l'enregistrement sur websdr.org	rien
17	Installe l' émulation des clients KiwiSDR en lançant <code>kiwi_install.sh</code> , pour que des clients Kiwi comme AetherSDR puissent se connecter à ce récepteur. Corrige les sources et ajoute <code>[kiwi_emulation]</code> aux fichiers de configuration de la racine du dépôt — voir Émulation des clients KiwiSDR	confirmation, oui par défaut
18	Lance <code>recompile.sh</code> , pour que tout soit construit à partir des sources corrigées	[3] Both backend and frontend → variante de départ → [1] build-all.sh
19	Affiche le récapitulatif : chaque étape avec son verdict, et chaque composant avec ce qui a été installé	rien

Installation sans surveillance

Chaque question dispose d'une variable d'environnement qui la remplace, et l'installateur bascule de lui-même sur les valeurs par défaut lorsque l'entrée standard n'est pas un terminal (un tube, un conteneur, un travail de CI). Cela vaut de la même façon pour les quatre : `install.sh`, `install_arch.sh`, `install_fedora.sh` et `install_opensuse.sh`. Définissez `PHANTOM_NONINTERACTIVE=1` et le déroulement se termine sans rien demander :

Variable	Effet
<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>	continuer sans curlpp — Arch et openSUSE seulement (défaut y)
<code>PHANTOM_FIX_CLOCK_SKEW=y\ n</code>	réinitialiser les horodatages des sources datés dans le futur, pour que meson puisse compiler (défaut y)

Les trois sous-installateurs marqués *n sans surveillance* sont eux-mêmes interactifs ; une exécution sans surveillance les ignore donc plutôt que de bloquer sur leurs questions. Nommez-les explicitement pour les inclure :

```
# 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
```

Après un redémarrage, les règles udev du RX888 et un éventuel pilote OpenCL sont pleinement actifs ; l'installateur vous indique quand un redémarrage ou une reconnexion est nécessaire.

Lorsqu'il installe le panneau d'administration, l'installateur propose aussi les deux unités `systemd` et **recommande fortement** de les accepter : la protection thermique tourne dans le panneau, donc sans elles un redémarrage ou un plantage laisse la machine sans protection. L'étape est ignorée automatiquement là où `systemd` ne tourne pas (conteneurs, WSL1, OpenRC) — voir le [guide du panneau](#).

Chaque extra est un script normal que vous pouvez aussi lancer seul à tout moment — `./setup_admin.sh` ([manuel](#)), `./install_rade.sh` ([manuel](#)), `./install-stats-server.sh` ([manuel](#)), `./setup-rx888-udev.sh` et les scripts des récepteurs `./setup-rspla.sh`, `./setup-fobos.sh`, `./setup-airspyhf.sh` et `./setup-hackrf.sh` — ce sont exactement ceux que l'installateur appelle.

Compiler à la main (référence)

IMPORTANT

Vous n'avez pas besoin de cette section pour une installation normale. `./install.sh` — ou l'installateur de votre distribution — fait tout cela pour vous ; voir [Ce que fait l'installateur](#). Ce qui suit est une référence pour une installation manuelle, pour une distribution qu'aucun script ne couvre, ou pour réparer un élément à la main.

Si vous devez un jour compiler sans l'installateur — une distribution non couverte, ou une compilation à moitié faite à réparer :

Compiler le 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
```

Compiler le frontend

```
cd frontend

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

# Return to project root
cd ..
```

Vérifiez l'installation

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

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

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

Installation du codec audio Opus

IMPORTANT

Vous n'avez pas besoin de cette section pour une installation normale. `./install.sh` — ou l'installateur de votre distribution — fait tout cela pour vous ; voir [Ce que fait l'installateur](#). Ce qui suit est une référence pour une installation manuelle, pour une distribution qu'aucun script ne couvre, ou pour réparer un élément à la main.

Opus offre une meilleure qualité audio et une latence plus faible que FLAC.

1. Installez la bibliothèque système libopus

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

2. Installez le décodeur Opus pour le 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. Vérifiez l'installation d'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 ..
```

Rapporteur de spots autorun (FT8/FT4/WSPR)

PhantomSDR-Plus est livré avec un **rapporteur de spots autorun** facultatif (dans le répertoire `autorun/`). Il décode FT8/FT4/WSPR côté serveur, directement à partir du récepteur, et téléverse les spots vers les réseaux de report :

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

Il se pilote entièrement depuis l'onglet **panneau d'administration** → « **Spot Reporting** », et le **report est DÉACTIVÉ par défaut** — rien n'est transmis ni téléversé tant que vous ne l'activez pas là. Le décodeur s'exécute comme un démon Node.js distinct et prélève l'audio du récepteur localement : il n'ajoute donc aucun matériel RF supplémentaire.

Prérequis

Prérequis	Remarques
Node.js 22+	Le même environnement d'exécution que la compilation du frontend — installé à l' étape Node.js .
Paquets npm ws + cbor-x	Résolus via <code>autorun/node_modules</code> , un lien symbolique vers le <code>node_modules</code> du frontend (les deux paquets sont déclarés dans <code>frontend/package.json</code>).
util-linux (taskset)	Le bouton « Start » de l'admin épingle le démon sur les E-cores avec <code>taskset</code> . Présent sur pratiquement toutes les distributions ; les installateurs l'ajoutent explicitement.
Indicatif + locator	Lus dans <code>frontend/site_information.json</code> (<code>siteSysop</code> / <code>siteGridSquare</code>) sauf s'ils sont remplacés dans l'onglet d'administration. Les spots sont téléversés sous cet indicatif.

⚠ **Ne rapportez que ce que vous recevez réellement.** Les spots sont téléversés vers des réseaux publics sous votre indicatif — n'activez que les bandes/modes que votre récepteur entend véritablement, et utilisez le bon locator.

Installation automatique

`install.sh` (et les variantes par distribution : `install_arch.sh`, `install_fedora.sh`, `install_opensuse.sh`) s'occupent de tout : ils installent Node.js 22 et `util-linux`, exécutent le `npm install` du frontend, puis créent automatiquement le lien symbolique `autorun/node_modules`. Aucune étape supplémentaire n'est nécessaire — la fonction est prête dès que `install.sh` se termine.

Installation manuelle

REMARQUE

L'installateur le fait déjà pour vous : il crée le lien symbolique et installe `util-linux`. N'utilisez les étapes ci-dessous que pour réparer une installation à la main.

Si vous avez installé manuellement (option B), créez vous-même le lien symbolique après le `npm install` du frontend, afin que le démon puisse résoudre ses dépendances :

```
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 -sfn ../frontend/node_modules autorun/node_modules

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

Remarque : `autorun/node_modules` est ignoré par git : un `git clone` frais ne le contient donc jamais — le lien symbolique doit être (re)créé après chaque récupération propre. Les installateurs le font pour vous ; la commande ci-dessus ne concerne que les installations manuelles.

Vérification

```
# 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
```

Ouvrez ensuite le panneau d'administration, allez dans l'onglet **Spot Reporting**, renseignez votre identité, cochez les bandes/modes à décoder, activez la ou les destinations, puis appuyez sur **Start**. Un badge « REPORTING » apparaît sur la cascade principale chaque fois que le report est actif. (PSK Reporter téléverse par lots toutes les 5 minutes et wsprnet toutes les 2 minutes : un démon fraîchement démarré affiche donc « 0 sent » pendant les premières minutes — c'est normal.)

Deux compteurs différents sont ensuite affichés, et il est facile de les confondre. Les tuiles **SPOTS UPLOADED PER DECODER** ne comptent que l'exécution en cours : Stop/Start les remet à zéro. Le nombre à côté de chaque case bande/mode est le total **depuis toujours** de cet emplacement, conservé dans `autorun-totals.json` et donc préservé au redémarrage. Voir le [guide du panneau d'administration](#) pour les deux, ainsi que pour la page **Graphiques**, qui trace la fréquence du processeur, la charge, la température et les utilisateurs connectés sur 15 minutes à 24 heures.

Si votre machine chauffe, le panneau embarque aussi une [protection thermique](#) qui arrête le serveur lorsque le processeur atteint une température dangereuse et le relance une fois refroidi. Elle déduit ses seuils du seuil critique de votre propre processeur, il n'y a donc rien à calculer, et elle fonctionne quelle que soit votre méthode de démarrage/arrêt. Installée avec le panneau, elle démarre en mode journalisation seule : elle note ce qu'elle *aurait* fait et ne change rien tant que vous ne l'activez pas dans Paramètres. Le décodage continu occupe un processeur 24 h sur 24, il vaut donc la peine de lire l'onglet CRASH après une semaine d'autorun pour voir jusqu'où votre machine monte réellement.

Le port du serveur est détecté automatiquement. Le démon se branche directement sur le `[server] port` de spectrumserver (boucle locale + jeton, en contournant le proxy). Il lit ce port dans le fichier de configuration avec lequel le spectrumserver en cours a été lancé, ce qui lui permet de fonctionner sur n'importe quel port sans configuration — la ligne `[autorun] tap backend: ...` dans `autorun.log` indique ce qu'il a résolu. Si votre serveur ne tournait pas au démarrage du démon, ou si votre installation est atypique, fixez-le dans `autorun.json` : `json "server": { "host": "127.0.0.1", "port": 9002 }` Un mauvais port se traduit par `504 / tap closed 1006` dans `autorun.log`, avec `decodes` bloqué à 0.

Émulation des clients KiwiSDR (facultatif)

Depuis la v4.1.0, PhantomSDR-Plus sait aussi répondre au **protocole KiwiSDR**, si bien qu'un logiciel écrit pour un KiwiSDR — **AetherSDR**, `kiwiclient` et les autres — se connecte directement à votre récepteur, sur le même hôte et le même port que vous publiez déjà. C'est désactivé tant que `[kiwi_emulation] enabled = true` n'est pas ajouté à la configuration avec laquelle tourne votre récepteur.

L'installateur le propose à l'étape 17 ; `./kiwi_install.sh` l'applique à une arborescence déjà installée.

Documentation complète : Émulation des clients KiwiSDR — ce que fait la passerelle, comment l'installer, chaque clé `[kiwi_emulation]`, connecter un client, le niveau audio, le S-mètre, la cadence de la cascade et un tableau des symptômes.

Configuration

1. Choisissez votre fichier de configuration

Sélectionnez le fichier de configuration adapté à votre SDR :

- `config-rtl.toml` - clés RTL-SDR
- `config-rspla.toml` - SDRplay RSP1A
- `config-airspyhf.toml` - Airspy HF+ Discovery
- `config-fobos-hf.toml` - RigExpert Fobos SDR, HF1/HF2 en échantillonnage direct (0-25 MHz)
- `config-fobos.toml` - RigExpert Fobos SDR, voie RF (25-6000 MHz)
- `config-rx888mk2.toml` - RX888 MK2
- `config-hackrf.toml` - HackRF One
- `config.example.hackrf.toml` - HackRF One

Pour cet exemple, nous utiliserons le RTL-SDR.

2. Modifiez le fichier de configuration

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

Réglages clés à configurer

```
[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
```

Repères pour le taux d'échantillonnage

Couverture	Taux d'échantillonnage	Taille de FFT
2 MHz	2048000	131072
3.2 MHz	3200000	131072
10 MHz	10000000	1048576
30 MHz	30000000	2097152
60 MHz	60000000	4194304

3. Configurez les informations du site

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

Modifiez les champs suivants :

```
{
  "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
}
```

Régions IARU :

- **1** : Europe, Afrique, Moyen-Orient, Asie du Nord
- **2** : Amériques (Nord, Centre, Sud), Caraïbes
- **3** : Asie-Pacifique, Océanie

4. Personnalisez les marqueurs de fréquence (facultatif)

```
nano markers.json
```

Ajoutez vos fréquences préférées, relais et stations de radiodiffusion.

5. Modifiez le script de démarrage

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

Le script de démarrage est un lanceur autonome doublé d'un chien de garde. Ne modifiez que le bloc **RECEIVER CONFIGURATION** près du début, afin que les arguments du récepteur correspondent à votre installation :

```
# == 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)
```

Paramètres à l'intérieur de `RX_ARGS` :

- `-f 145000000` : fréquence centrale (145 MHz)
- `-s 2048000` : taux d'échantillonnage (2,048 MSPS)

`CONFIG` pointe vers votre `.toml`. Vous n'avez **pas** besoin de modifier autre chose dans le script — la logique de démarrage/redémarrage/chien de garde/journalisation est générique. Vous pouvez aussi remplacer les arguments du récepteur au lancement sans modifier le fichier : `RX_ARGS="-f 145000000 -s 2048000 -" ./start-rtl.sh` (`RX888_ARGS` pour `start-rx888mk2.sh`).

Configuration propre à chaque appareil SDR

RTL-SDR

Installez les outils RTL-SDR

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

Testez le RTL-SDR

```
rtl_test
```

Appuyez sur Ctrl+C pour arrêter. Vous devriez voir des informations sur le taux d'échantillonnage.

Modifiez la configuration

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

Réglages courants pour le RTL-SDR :

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

Récepteurs sur SoapySDR (RSP1A, Fobos, Airspy HF+)

Le SDRplay RSP1A, le RigExpert Fobos SDR et l'Airspy HF+ rejoignent le serveur de la même façon : un pilote SoapySDR pour l'appareil, et `rx_sdr` (de `rx_tools`) qui transmet ses échantillons à `spectrumserver`. Un script par récepteur installe toute la chaîne :

Récepteur	Script	Option de l'installateur	Script de lancement	Configuration
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>

Chaque script :

- installe `git`, `cmake`, un compilateur, `libusb` et **SoapySDR** depuis les paquets de votre distribution — il reconnaît `apt`, `dnf`, `pacman` et `zypper`, et fonctionne donc aussi bien sur Debian/Ubuntu, Fedora, Arch et openSUSE ;
- compile le pilote du récepteur depuis les sources dans `sdr_drivers/` (non suivi par `git`) ;
- compile **rx_tools** pour `rx_sdr`, sauf si `rx_sdr` est déjà installé — une seule copie sert à tous les récepteurs ;
- installe une **règle udev** et vous ajoute au groupe `plugdev`, pour que le script de lancement n'ait jamais besoin de `sudo` pour l'appareil. **Déconnectez-vous et reconnectez-vous une fois** pour que le groupe prenne effet ;
- vérifie pour finir que SoapySDR liste bien le pilote, et s'arrête sur une erreur sinon.

Lancez-les avec votre utilisateur normal — ils appellent `sudo` eux-mêmes. Ils sont idempotents : relancer l'un d'eux met à jour les sources du pilote avec `git pull` et recompile. L'installateur appelle exactement ces scripts ; sur une station déjà installée, lancez seul celui de votre récepteur. La partie commune se trouve dans `setup-sdr-common.sh`, qui ne se lance pas seul. Deux variables facultatives valent pour les trois : `SDR_USER=<nom>` met un autre compte dans `plugdev`, et `SDR_SKIP_DEPS=1` saute l'installation des paquets.

Appareil branché, vérifiez que le récepteur est détecté :

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

SDRplay RSP1A

`setup-rsp1a.sh` installe le **pilote open source** : `libmirisdr-5`, le pilote issu de la rétro-ingénierie des puces Mirics MSi2500/MSi001 des RSP1 et RSP1A, avec son module `SoapySDR SoapyMiri` (`driver=soapyMiri`). Il ne demande aucun service en arrière-plan ni `root`. Le script met aussi sur liste noire les pilotes noyau `msi2500` et `msi001` dans `/etc/modprobe.d/blacklist-msi2500.conf`, car ils prennent le RSP1A comme radio V4L2 avant que `rx_sdr` puisse l'ouvrir. Si l'appareil était branché pendant qu'ils étaient chargés, débranchez-le et rebranchez-le une fois (ou redémarrez).

L'API fermée de `SDRplay` (`driver=sdrplay`, avec `SoapySDRPlay` et le service `sdrplay`) fonctionne toujours — le fichier inclus `instructions-for-rsp1a` décrit cette installation. `start-rsp1a.sh` choisit seul entre les deux : si `SoapySDR` a le pilote `sdrplay`, il utilise l'API et redémarre d'abord le service `sdrplay`, sinon il utilise `libmirisdr-5`. Une station installée sur l'API continue donc de fonctionner sans changement. Pour en imposer un, lancez avec `RX_DRIVER=miri` ou `RX_DRIVER=sdrplay`. Le pilote choisi est écrit dans `logwebsdr.txt` sous la forme `RSP1A driver: miri` ou `RSP1A driver: sdrplay`.

RigExpert Fobos SDR

Le Fobos ne diffuse **qu'une voie à la fois** ; il a donc deux scripts de lancement qui partagent le verrou du chien de garde comme tout `start-*.sh` — lancer l'un arrête l'autre :

- **`start-fobos-hf.sh`** — les entrées HF1/HF2 en échantillonnage direct, 0-25 MHz, sur le port 9003. Dans ce mode il n'y a pas d'oscillateur local : l'ADC numérise 0-25 MHz en permanence (`-f 0` est voulu, et l'avertissement *Failed to set center freq* de `rx_sdr` est attendu). `rx_sdr` fournit du CF32 à 50 Msps ; le petit convertisseur `cf32_to_real` (compilé par `setup-fobos.sh` à partir de `cf32_to_real.c`) garde la voie I et la met à l'échelle en `s16`, d'où `signal="real"` et `format="s16"` dans `config-fobos-hf.toml`.
- **`start-fobos.sh`** — la voie RF, 25-6000 MHz, IQ, sur le port 9002. Choisissez la fenêtre avant le premier lancement : `-f` (centre) et `-s` (fréquence d'échantillonnage) sur la ligne `RX_ARGS` doivent correspondre à `frequency=` et `sps=` dans `config-fobos.toml`. Les valeurs livrées (97 MHz, 20 Msps) ne sont qu'un point de départ pour vérifier que cela tourne.

IMPORTANT

`config-fobos-hf.toml` est livré avec `accelerator="opencl"` : un flux de 50 Msps à travers une FFT de 1M bins est aussi lourd que les 60 Msps du RX888, et sur le CPU il perd des échantillons et fait chauffer la machine. Sur une machine sans périphérique OpenCL, `spectrumserver` ne démarre pas avec ce réglage — mettez-y `accelerator="none"`.

`SoapyFobosSDR` a besoin des en-têtes de `libfobos` et de `libfobos-sdr-agile` pour se compiler, donc `setup-fobos.sh` compile les deux, même pour le firmware standard. La prise en charge du Fobos vient d'un script écrit par une autre station après avoir mis un Fobos en service sous `PhantomSDR-Plus` ; les valeurs HF de `config-fobos-hf.toml` sont celles qui ont fonctionné chez elle.

Airspy HF+

`setup-airspyhf.sh` installe `libairspyhf` depuis la distribution là où elle est empaquetée (Debian et Ubuntu, Fedora, openSUSE) et la compile depuis les sources ailleurs (Arch ne l'a que dans l'AUR), puis compile `SoapyAirspyHF` (`driver=airspyhf`). `start-airspyhf.sh` fournit 912 ksps IQ en CS16, que `config-airspyhf.toml` lit en `format="s16"`. Le fichier inclus `instructions-for-airspy` décrit la voie manuelle.

RX888 MK2

Installez les outils 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
```

Exécuter `rx888_stream` sans `sudo` (règles udev)

Par défaut, le périphérique USB Cypress FX3 du RX-888 n'est accessible qu'à root, donc `rx888_stream` nécessiterait `sudo`. Exécutez une fois l'assistant fourni pour installer les règles udev et vous ajouter au groupe `plugdev` :

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

Il écrit `/etc/udev/rules.d/99-rx888.rules` pour les trois identifiants de produit Cypress FX3 (`04b4:00f1/00f3` bootloader et `04b4:8613` firmware chargé), recharge udev et crée un lien symbolique stable `/dev/rx888`. **Déconnectez-vous puis reconnectez-vous** (pour le changement de groupe) et **rebranchez l'appareil** une fois ; ensuite `rx888_stream` — et le lanceur `start-rx888mk2.sh` — s'exécutent sans `sudo`. Le script est idempotent : vous pouvez le relancer sans risque. Pour configurer un autre compte : `RX888_USER=<nom> ./setup-rx888-udev.sh`.

Modifiez la configuration

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

Réglages courants pour le RX888 :

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

HackRF One

Pour le HackRF, rien n'est compilé : `hackrf_transfer`, du paquet `hackrf` de votre distribution, transmet les échantillons directement à `spectrumserver`. `./setup-hackrf.sh` (option 7 de l'installateur) installe ce paquet avec `apt`, `dnf`, `pacman` ou `zypper` et ajoute une règle udev et le groupe `plugdev`, pour que `start-hackrf.sh` tourne sans `sudo` — déconnectez-vous et reconnectez-vous une fois ensuite.

```
./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` diffuse une fenêtre à la fois. Choisissez-la avant le premier lancement : `-f` (centre) et `-s` (fréquence d'échantillonnage) sur sa ligne `RX_ARGS` doivent correspondre à `frequency=` et `sps=` dans `config-hackrf.toml`. `-l` et `-g` sont les gains du LNA (0-40 dB, pas de 8 dB) et du VGA (0-62 dB, pas de 2 dB), et `-a 1` allume l'amplificateur RF. Les valeurs livrées (98 MHz, 20 Msps) ne sont qu'un point de départ pour vérifier que cela tourne.

`hackrf_transfer` écrit de l'IQ 8 bits **signé**, la configuration a donc besoin de `format="s8"`. `config.example.hackrf.toml` indiquait `u8` jusqu'en septembre 2026 ; avec `u8` chaque échantillon est décalé de 128 et la chute d'eau ne montre que du bruit — vérifiez toute copie plus ancienne.

Plusieurs récepteurs sur un ordinateur

Un ordinateur peut faire tourner en même temps plusieurs des récepteurs ci-dessus — par exemple un RX-888 pour les ondes courtes et un RTL-SDR pour le 2 m — chacun comme serveur à part, avec un sélecteur de récepteur sur la page. Chaque récepteur supplémentaire se démarre avec `INSTANCE=<nom> ./start-<radio>.sh` et se configure dans `instances/<nom>/` ; `receivers.toml` les recense pour le proxy et pour `start-all.sh`. Tout est décrit dans [Plusieurs récepteurs](#). Le plus rapide pour ajouter un récepteur est `./add-receiver.sh`, que les installeurs proposent à la fin : il installe le pilote (`setup-rtlsdr.sh` pour un RTL-SDR, le `setup-*.sh` correspondant pour les autres) et écrit pour vous le dossier de l'instance et `receivers.toml`.

Tests et vérification

1. Essai

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

Cela démarre le serveur **en arrière-plan** (il se détache et rend la main immédiatement) et commence à journaliser dans `logwebsdr.txt`.

2. Recherchez d'éventuelles erreurs

Le script journalise sa progression dans `logwebsdr.txt` — suivez-la en direct :

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

Cherchez :

- `Starting the initialization script of the PhantomSDR Server`
- `<receiver> running (PID ...)`
- `Server started normally`
- `Server ... FAILED ...` ou `ERROR: receiver binary '...' not found` (corrigez les arguments du récepteur ou le `.toml`, ou installez l'outil du récepteur, puis relancez le script de démarrage)

3. Accédez à l'interface web

Ouvrez votre navigateur sur :

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

(Remplacez le numéro de port par celui que vous avez configuré)

4. Vérifiez le fonctionnement

- La cascade doit s'afficher
- L'audio doit jouer lorsque vous cliquez sur des signaux
- Le S-mètre doit réagir aux signaux
- Le nombre d'utilisateurs doit indiquer « 1 »

5. Testez depuis un autre appareil

Depuis un autre ordinateur de votre réseau :

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

6. Vérifiez l'utilisation des ressources

```
# Monitor CPU usage
htop

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

# Monitor network usage
iftop
```

7. Arrêtez le serveur

Le serveur s'exécute en arrière-plan sous la surveillance d'un chien de garde : **Ctrl+C ne l'arrêtera donc pas** (et le chien de garde le relancerait). Utilisez le script d'arrêt partagé, valable pour tous les récepteurs — il arrête d'abord le chien de garde, puis le récepteur et spectrumserver :

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

Mise en place du démarrage automatique

Avec systemd (recommandé)

1. Créez le fichier de service

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

Ajoutez le contenu suivant (ajustez les chemins et l'utilisateur) :

```
[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
```

Le processus `--watchdog` relance déjà de lui-même le récepteur et `spectrumserver` ; `Restart=always` n'est qu'un filet de sécurité pour le cas rare où le chien de garde s'arrêterait lui-même. Comme `systemd` supervise ici le serveur, vous pouvez utiliser `systemctl start/stop/restart` et `journalctl -u phantomsdr -f` au lieu de lancer les scripts à la main.

2. Activez et démarrez le service

```
# 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. Gérez le service

```
# Start
sudo systemctl start phantomsdr

# Stop
sudo systemctl stop phantomsdr

# Restart
sudo systemctl restart phantomsdr

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

Avec Screen (alternative)

Généralement inutile : `./start-rtl.sh` se détache déjà en arrière-plan (via `setsid`) et continue de tourner après votre déconnexion, avec son propre chien de garde. `Screen` n'est utile que si vous voulez expressément une session interactive pour exécuter la forme au premier plan `./start-rtl.sh --watchdog`.

1. Installez Screen

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

2. Démarrez dans une session Screen

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

Appuyez sur Ctrl+A, puis D pour détacher.

3. Rattachez-vous à la session

```
screen -r phantomsdr
```

Protection thermique du processeur

Manuel complet : [THERMAL_GUARD.md](#) — les quatre modes et ce que le sysop doit faire pour chacun, le fonctionnement sans le panneau, l'étage throttle sans root, les tests et le dépannage.

PhantomSDR-Plus embarque un garde qui arrête le serveur si le processeur atteint une température dangereuse et le relance une fois refroidi. Il déduit ses seuils du seuil critique que publie votre propre processeur, il n'y a donc rien à calculer, et il fonctionne quelle que soit votre méthode de démarrage/arrêt.

Si vous utilisez le panneau d'administration, vous l'avez déjà : il tourne dans le panneau et se configure depuis sa page Paramètres. Voir le [guide du panneau d'administration](#). Le reste de cette section concerne les installations **sans** panneau.

1. Vérifier ce que le garde voit sur votre machine

```
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
```

Seuls de vrais capteurs sur la puce sont utilisés (`coretemp`, `k10temp`, `zenpower`, `cpu_thermal`) ; `acpitz` et les zones thermiques non étiquetées sont ignorés à dessein, car ils rapportent souvent une température de boîtier ou de carte inférieure de plusieurs dizaines de degrés au processeur. Si cela affiche `sensor : NONE`, cette machine ne peut pas être protégée — fréquent sur un VPS ou dans un conteneur — et le garde restera inactif plutôt que de faire semblant.

Rien à installer : `thermal_guard.py` n'utilise que la bibliothèque standard de Python.

2. Le configurer

Le garde lit `admin_config.json` à côté de `thermal_guard.py`. Créez-le si vous n'en avez pas — sans panneau, vous n'en aurez pas. Le fichier minimal utile :

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

- `thermal_mode` — `log` (journaliser seulement, la valeur par défaut), `throttle`, `stop` ou `stop+restart`. **C'est celui que vous devez définir**, car le garde est livré inerte : tel quel, il note seulement ce qu'il *aurait* fait. Vous pouvez aussi le passer en ligne de commande avec `--mode stop+restart`, ce qui l'emporte sur le fichier — si tous les autres réglages par défaut vous conviennent, vous n'avez donc besoin d'aucun fichier de configuration.
- `stop_script` / `start_script` — comment il arrête et démarre votre serveur. Sans script d'arrêt, il se rabat sur `SIGTERM` puis, après 10 secondes, sur `SIGKILL` visant `sdr_process_name`. Sans script de démarrage, il arrête mais ne relance jamais.
- Les seuils et durées peuvent aussi être définis ici, avec les mêmes noms de clés que dans le panneau — le tableau complet figure dans le [guide du panneau d'administration](#).

Si systemd supervise votre serveur SDR, faites pointer `stop_script` vers un petit script exécutant `systemctl stop` votre-unité, plutôt que de laisser le garde signaler le processus directement. Le garde l'emportera de toute façon — tant que le processeur est trop chaud, il réémet l'arrêt toutes les 2 secondes, si bien que tout ce qui relance le serveur est défilé — mais un arrêt propre vaut mieux qu'un bras de fer toutes les deux secondes.

3. L'exécuter comme service

Le dépôt contient une unité prête à l'emploi, `thermal-guard.service`. Modifiez-y `User=` et les deux chemins, puis :

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

`User=` doit être un compte autorisé à exécuter vos scripts de démarrage/arrêt — normalement le même utilisateur que celui qui fait tourner le serveur SDR. **Un garde tournant sous un utilisateur incapable d'arrêter le serveur vous laisse sans protection tout en ayant l'air de vous protéger.**

Pour armer le garde depuis l'unité plutôt que depuis le fichier de configuration, ajoutez `--mode` à `ExecStart` :

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

Sans `systemd` :

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

`python3 thermal_guard.py --help` liste toutes les options, dont `--config` pour un fichier de configuration ailleurs qu'à côté du script.

4. L'observer, puis lui faire confiance

Tout ce que fait le garde est écrit dans `crash.log`, dans le répertoire PhantomSDR-Plus, un événement par ligne :

```
[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
```

Laissez d'abord `thermal_mode` sur `"log"` pendant une semaine et lisez ce fichier après vos moments chargés — une recompilation complète, un après-midi de canicule. Si rien n'y figure, votre machine n'a jamais approché. Passez ensuite le mode à `stop`.

Avant de vous y fier, éprouvez tout le parcours une fois avec une mesure fictive au-dessus de votre seuil d'arrêt :

```
"thermal_test_temp": 97
```

Le garde la traite comme réelle, si bien que `warn` → `stop` → verrouillage → récupération → redémarrage se déroulent à la demande. Retirez la ligne (ou mettez-la à `null`) pour revenir au capteur réel. En mode `stop`, cela arrête réellement le serveur et déconnecte vos auditeurs : faites-le quand personne n'écoute, ou testez en `log`, où l'on voit ce qui *serait* arrivé sans rien toucher.

Dépannage

Erreurs de compilation

Problèmes de dépendances

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

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

La configuration Meson échoue

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

meson setup s'arrête sur 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'
```

L'arborescence des sources n'est pas en cause. Une ancienne installation par `pip install --user meson` a laissé un script de lancement dans `~/local/bin`, qui passe avant `/usr/bin` dans le `PATH` et masque donc la copie installée par le gestionnaire de paquets. Une montée de version de la distribution (Ubuntu

24.04 → 26.04, par exemple) fait passer Python à une nouvelle version, l'ancien `site-packages` qui contenait `mesonbuild` n'est plus dans le chemin d'import, et le script meurt sans avoir rien fait. La même chose peut arriver à `ninja`.

L'installateur détecte ce cas et le contourne le temps de l'exécution, en vous avertissant, mais réparez tout de même le système :

```
rm -f ~/.local/bin/meson
hash -r
meson --version      # doit afficher une version
```

Si vous préférez garder meson installé via pip, réinstallez-le pour le Python dont dispose maintenant ce système :

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

meson setup s'arrête sur Clock skew detected

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

L'arborescence des sources n'a rien d'anormal : c'est l'horloge du système qui est en retard sur les fichiers. `meson` et `ninja` refusent de compiler quand une entrée est plus récente que l'heure courante, car ils ne peuvent pas savoir ce qui est périmé. Cela arrive sur un Raspberry Pi dont le support de pile RTC est vide — chaque démarrage repart de la dernière heure connue, et une compilation lancée avant que NTP ne se synchronise voit toute l'arborescence datée dans le futur — ainsi que sur une arborescence décompressée ou copiée depuis une machine en avance.

Corrigez d'abord l'horloge :

```
timedatectl                      # l'heure est-elle juste ? NTP est-il synchronisé ?
sudo timedatectl set-ntp true
```

Attendez quelques secondes, puis relancez l'installateur. Il vérifie ce point avant d'appeler `meson` et propose de réinitialiser les horodatages fautifs (`PHANTOM_FIX_CLOCK_SKEW=y|n`). À la main, depuis l'arborescence des sources :

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

Le backend est compilé mais il n'y a pas de page web

Le symptôme est une installation qui semble avoir largement réussi : `build/spectrumserver` est là, le serveur démarre, et le navigateur ne reçoit rien — parce que `frontend/dist/` n'a jamais été produit. Cherchez dans la sortie de l'installateur :

```
[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` demande `@swc/core` `^1.12.14`, si bien qu'un `npm install` neuf résout vers 1.16.0, dont `printSync()` rejette l'arbre syntaxique que le greffon lui transmet. Cela n'a rien à voir avec votre distribution : chaque installateur épingle désormais la version qui fonctionne, et `frontend/package.json` aussi. Une machine installée avant cet épinglage continue de fonctionner jusqu'à la suppression de son `node_modules` — d'où l'apparition soudaine du problème lors d'une réinstallation.

Si vous réparez à la main une copie plus ancienne :

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

Erreurs d'exécution

Port déjà utilisé

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

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

Permission refusée pour le SDR

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

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

Pour un **RX-888 MkII**, cela ne suffit pas à soi seul — le périphérique Cypress FX3 a aussi besoin de règles udev. Exécutez l'assistant fourni (il installe les règles et vous ajoute à `plugdev`), puis déconnectez-vous/reconnectez-vous et rebranchez l'appareil :

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

Problèmes audio

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

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

Problèmes OpenCL

clinfo n'affiche aucun appareil

```
# 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
```

Aucune amélioration des performances

```
# 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.
```

Problèmes réseau

Impossible d'accéder depuis d'autres appareils

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

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

Latence élevée

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

# Increase buffer size
buffer_size = 16384

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

Problèmes d'appareil SDR

RTL-SDR introuvable

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

# Test with rtl_test
rtl_test

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

SDRplay introuvable

grep "RSP1A driver" logwebsdr.txt indique le pilote choisi par start-rspla.sh.

Avec le pilote ouvert (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"
```

Avec l'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"
```

Fobos ou Airspy HF+ introuvable

```
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 le pilote manque dans la liste, relancez `./setup-fobos.sh` ou `./setup-airspyhf.sh` : il s'arrête en donnant la raison si la compilation ou la vérification SoapySDR échoue.

Optimisation des performances

Optimisation du processeur

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

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

Optimisation de la mémoire

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

Optimisation du réseau

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

Mise à jour de PhantomSDR-Plus

Depuis la version 4.1.0, le dépôt fournit `update.sh`, un outil qui met une installation à jour à partir de l'arborescence publiée **sans toucher aux fichiers qui font qu'elle est votre station**. Il remplace le script `git pull` que les éditions précédentes de ce guide vous demandaient d'écrire, et il n'a pas besoin de git du tout : l'arborescence publiée est téléchargée sous forme d'archive tar et comparée à la vôtre fichier par fichier. Il fonctionne donc de la même façon que vous ayez cloné le dépôt, décompressé un `update.zip` ou copié l'arborescence depuis une clé USB.

Si votre installation n'a pas encore update.sh

Une arborescence ancienne ne contient pas le script. Récupérez-le une fois : c'est la seule étape de toute cette procédure que vous ferez à la main :

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

Ensuite tout — sources, frontend, documentation, installateurs et update.sh lui-même — arrive par l'outil.

Étape 1 — voir ce qui changerait (rien n'est écrit)

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

L'arborescence publiée est téléchargée, comparée à la vôtre, et un rapport est affiché. Rien n'est écrit, l'appel est donc sans danger à tout moment, y compris sur un récepteur en service. Le code de retour vaut 0 si vous êtes à jour et 10 si une mise à jour attend : une tâche cron peut ainsi vous prévenir lorsqu'il y a quelque chose à faire.

Étape 2 — l'appliquer

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

Trois catégories de fichiers sont traitées différemment, et c'est tout l'intérêt :

Fichiers	Ce qui se passe
config.toml, config-<récepteur>.toml, markers.json, admin_config.json, autorun.json, frequencylist/, chat_history.txt, frontend/variant.json, frontend/site_information.json, les journaux, build/, frontend/dist/	Jamais touchés , et jamais mentionnés dans une question. Ce sont eux qui font de la machine <i>votre</i> récepteur.
start-*.sh, stop-websdr.sh, les unités *.service, install*.sh, recompile.sh, setup_admin.sh, smeter_theme.sh, proxy.py, admin_server.py, thermal_guard.py, websdr_relay.py, frontend/src/bands-config.js, les configurations d'exemple config.example.*.toml	Toujours soumis à une question , car ce sont les fichiers qu'un sysop a de bonnes raisons d'avoir modifiés.
Tout le reste	Mis à jour, après copie de l'ancien fichier dans l'archive de sauvegarde (voir <i>Annuler une mise à jour</i>).

Pour chaque fichier du groupe du milieu, les différences sont affichées et trois choix sont proposés :

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

- **Keep mine** — votre fichier est laissé exactement tel quel.
- **upstream** — la nouvelle version est installée, la vôtre étant sauvegardée au préalable.
- **both** — la nouvelle version est écrite à côté de la vôtre sous le nom `start-rx888mk2.sh.new`, pour que vous y reportiez vos modifications à votre rythme.

À quoi ressemble une première exécution. La première fois, rien n'indique de quelle version viennent vos fichiers : chaque fichier du groupe du milieu vous est donc soumis, soit une dizaine de questions. Répondez ainsi :

Votre situation	Réponse
Vous n'avez jamais modifié ce fichier	u — prendre la nouvelle version. Le cas habituel.
Vous l'avez modifié (vos RX888_ARGS, l'épinglage CPU, une unité ajustée)	b — le vôtre est conservé et le nouveau arrive à côté sous <fichier>.new.
Vous n'êtes pas sûr	ENTRÉE — le vôtre est conservé, rien n'est perdu, vous comparerez plus tard.

Votre configuration n'entre jamais là-dedans : les questions ne portent que sur des scripts et des unités de service.

update.sh note dans .update-state/ la version de chaque fichier qu'il installe. Dès la deuxième exécution, il distingue donc un fichier que **vous** avez modifié d'un fichier simplement ancien, et ne s'arrête que sur ceux auxquels vous avez réellement touché.

Avant d'écrire quoi que ce soit, il arrête le récepteur, le panneau d'administration, le proxy inverse et le relais de diversité WebSDR **de l'installation qu'il met à jour** — ce qui dessert un autre répertoire est signalé et laissé en marche, de sorte qu'un second clone peut être mis à jour pendant que le premier reste en service — puis redémarre à la fin exactement ce qu'il a arrêté. Si des fichiers source ou du frontend ont changé, il propose de lancer recompile.sh pour vous. Rien n'est jamais supprimé : les fichiers disparus du dépôt sont signalés et ne sont retirés que si vous le demandez avec --prune.

Annuler une mise à jour

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

Chaque fichier qu'une mise à jour écrase est d'abord sauvegardé, dans une archive datée par exécution, dans le dossier ordinaire et visible update-backups/ :

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

L'archive ne contient que les fichiers que cette exécution a remplacés, chacun à son chemin dans votre arborescence : elle peut donc être copiée sur une autre machine en un seul fichier, ouverte n'importe où ou décompressée directement sur l'installation. Si zip n'est pas installé, elle est écrite en .tar.gz. Les trois plus récentes sont gardées. L'une ou l'autre de ces commandes remet ce qu'une mise à jour a remplacé :

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

Chaque archive contient aussi son propre restore.sh, pour le cas où update.sh lui-même serait le problème : décompressez l'archive et donnez-lui le répertoire de votre installation, par exemple ./restore.sh /home/pi/PhantomSDR-Plus. Pendant la restauration, update.sh s'exécute depuis une copie hors de l'arborescence, car il fait généralement partie des fichiers remis en place.

Les sauvegardes antérieures au 23 septembre 2026 se trouvent dans le dossier caché .update-backups/<horodatage>/ ; elles se restaurent de la même façon et comptent parmi les trois gardées.

Autres options

```
./update.sh --apply --yes      # ne demande jamais rien ; tout fichier modifié est CONSERVÉ
./update.sh --ref v5.0.0      # une étiquette, une branche ou un commit précis
./update.sh --list-excludes    # affiche les règles « ne pas toucher » telles qu'appliquées ici
./update.sh --verbose         # liste tous les fichiers, pas seulement les 40 premiers
```

Vous pouvez ajouter vos propres règles « ne pas toucher » en écrivant un motif par ligne dans `update-exclude.txt`, à la racine de l'installation.

Si la recompilation échoue sur une installation très ancienne

`update.sh` met à jour des fichiers, pas des paquets système. Si votre arborescence est assez ancienne pour que la compilation réclame des bibliothèques que vous n'avez pas, `recompile.sh` s'arrêtera sur une erreur du compilateur ou de meson. Ce n'est pas une mise à jour cassée : il manque les dépendances :

```
./install.sh
```

L'installateur est lui-même mis à jour par la même exécution, et votre configuration y survit également.

Mettre à jour à la main

Si vous préférez appliquer vous-même un lot de fichiers :

```
cd ~
unzip -o update.zip
cd PhantomSDR-Plus
./stop-websdr.sh
cp -a ~/update/. .
chmod +x *.sh
./recompile.sh
./start-rx888mk2.sh          # ou le script de démarrage de votre récepteur
```

Sauvegardez d'abord votre configuration — `config-*.toml`, `frontend/site_information.json`, `admin_config.json` et `markers.json` — car un lot de fichiers ne peut pas distinguer vos modifications de celles de la version publiée. C'est exactement le problème que `update.sh` résout.

Sauvegarde et restauration

Fichiers à sauvegarder

- Fichiers de configuration : `*.toml`
- Informations du site : `frontend/site_information.json`
- Marqueurs : `markers.json`
- Scripts personnalisés : `start-*.sh`, `stop-*.sh`
- Historique du chat : `chat_history.txt`
- Image de fond : `frontend/src/assets/background.jpg`

Commande de sauvegarde

```
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
```

Commande de restauration

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

Considérations de sécurité

Configuration du pare-feu

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

Proxy inverse (facultatif)

Envisagez d'utiliser nginx ou Apache comme proxy inverse pour :

- le chiffrement SSL/TLS
- l'association d'un nom de domaine
- la répartition de charge
- le contrôle d'accès

Limites d'utilisateurs

Modifiez `config.toml` :

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

Obtenir de l'aide

Ressources

- **Documentation** : ce guide, README.md, USER_GUIDE.md
- **GitHub Issues** : <https://github.com/sv1bt/PhantomSDR-Plus/issues>
- **Démo en direct** : <http://phantomsdr.no-ip.org:8900/>

Signaler un problème

Lorsque vous signalez un problème, indiquez :

1. le système d'exploitation et sa version
2. le modèle d'appareil SDR
3. le contenu du fichier de configuration
4. les messages d'erreur
5. l'utilisation des ressources système (CPU, RAM, GPU)

Soutien de la communauté

- Consultez les issues GitHub existantes avant d'en créer de nouvelles
- Fournissez des informations détaillées sur votre installation
- Joignez les journaux et les messages d'erreur

- Soyez patient et respectueux

Annexe A : liste complète des dépendances

Liste des paquets 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)
```

Annexe B : exemples de configuration

Exemple 1 : RTL-SDR pour VHF/UHF

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

[input.driver]
format = "u8"

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

Exemple 2 : RX-888 mk2 pour HF (0-30 MHz)

Le récepteur qu'utilisent la plupart des sysops. Voici une section `[input]` complète, et non un extrait, avec les valeurs du `config-rx888mk2.toml` fourni avec le dépôt.

```
[input]
sps = 60000000          # 0-30 MHz par échantillonnage direct
fft_size = 4194304      # voir la note ci-dessous
fft_threads = 8
brightness_offset = -9  # plus négatif si la cascade montre des zones noires
frequency = 0           # bande de base : le RX-888 échantillonne depuis le continu
signal = "real"         # pas "iq" : l'échantillonnage direct donne un flux réel
accelerator = "opencl"  # "none" s'il n'y a pas d'environnement 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"
```

Il est alimenté par `rx888_stream`, ce dont `start-rx888mk2.sh` se charge pour vous :

```
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
```

À propos de `fft_size` : à 60 MSPS, la bonne taille est 4194304. 8388608 double la résolution de la cascade et, avec elle, la mémoire et le coût CPU de chaque transformée : sur la plupart des machines, on paie des bins plus fins par des trames perdues. Commencez à 4194304 et ne montez que si le serveur est largement au repos.

Exemple 3 : HackRF pour la FM à large bande

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

[input.driver]
format = "s8"

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

Installation terminée ! Vous devriez maintenant disposer d'un serveur PhantomSDR-Plus pleinement fonctionnel.

73 de SV1BTL & SV2AMK

CHAPITRE 3

Configuration du panneau d'administration

ADMIN_PANEL_SETUP.md

Panneau d'administration PhantomSDR-Plus — Guide de configuration

⚠ **Avis de sécurité** : gardez ce panneau d'administration sur votre réseau domestique ou derrière un VPN. L'exposition publique du port d'administration n'est pas recommandée sans un durcissement d'authentification approprié.

Vue d'ensemble

Le panneau d'administration se compose de deux services Python :

Service	Fichier	Rôle
Panneau d'administration	admin_server.py	Application web Flask, liée à 127.0.0.1 (interne uniquement)
Proxy inverse	proxy.py	Expose à la fois le SDR et le panneau d'administration sur un seul port public

Les deux services lisent leur configuration dans `admin_config.json`, écrit par `setup_admin.sh`. **Ne modifiez pas directement les constantes de port dans les fichiers Python** — tous les ports et l'adresse IP de l'hôte SDR sont stockés dans `admin_config.json`.

Ce que vous apporte le panneau d'administration

- **Tableau de bord** — état du serveur en direct, CPU/RAM, processus principaux, sortie récente des journaux, terminal
- **Éditeur de configuration** — consultez, modifiez et enregistrez n'importe quel fichier `.toml`, `.sh`, `.json`, `.h`, `.cpp` de votre installation
- **Visionneuse de journaux** — suivez n'importe quel fichier journal en temps réel (serveur SDR, panneau d'administration, RADE, plantages, autorun, proxy)
- **Marqueurs** — consultez et modifiez les marqueurs de fréquence
- **Historique du chat** — consultez le journal du chat WebSDR, effacez-le entièrement ou supprimez des messages individuels sans redémarrer le serveur
- **Message sur la cascade** — diffusez une bannière persistante à tous les utilisateurs connectés, visible en temps réel sur l'affichage en cascade
- **Utilisateurs** — liste en direct des auditeurs connectés avec leur fréquence, leur mode, leur durée de connexion et un bouton `Kick`
- **Suppression de messages du chat** — bouton Delete qui retire immédiatement ce seul message du journal

- **Messages diffusés sur la cascade** — permet de pousser une bannière de texte persistante sur la cascade de chaque utilisateur connecté
- **Spot Reporting** — démarrez/arrêtez le décodeur autorun FT8/FT4/JS8/WSPR, choisissez les bandes/modes et rapportez les spots à PSK Reporter / wsprnet (désactivé par défaut), avec deux compteurs : des tuiles par décodeur pour l'exécution en cours et un total depuis toujours à côté de chaque case bande/mode
- **Graphiques** — fréquence du CPU, charge du CPU, température du CPU et utilisateurs connectés, tracés sur les 15 dernières minutes / 1 heure / 4 heures / 12 heures / 24 heures
- **Protection thermique (Thermal Guard)** — arrête le serveur si le processeur surchauffe et le redémarre une fois refroidi, avec des seuils déduits de votre propre processeur ; fonctionne avec n'importe quelle méthode de démarrage/arrêt et est livrée en mode journalisation seule, elle ne fait donc rien tant que vous ne l'activez pas
- **Paramètres** — modifiez le mot de passe d'administration, le répertoire de base du SDR, le nom du processus, le port public

Prérequis

- PhantomSDR-Plus déjà installé et en cours d'exécution
- Python 3.8+

REMARQUE

En temps normal, l'installateur le fait pour vous. `./install.sh` — comme chacun des quatre installateurs par distribution, les deux installateurs RADE et celui du serveur de statistiques — met le panneau en place par défaut, en installant d'abord `pip` avec le gestionnaire de paquets du système s'il manque. Cette page décrit ce qui se passe à cette étape, et quoi faire si vous l'avez sautée ou souhaitez changer quelque chose ensuite.

Les bibliothèques Python du panneau (`flask`, `psutil`, `aiohttp`, `tomli-w`) sont installées par `setup_admin.sh` ; en cas d'échec, il affiche la commande exacte à lancer.

Étape 1 — Fichiers

Placez ces quatre fichiers dans votre répertoire PhantomSDR-Plus :

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

Rendez les scripts shell exécutables :

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

Étape 2 — Lancer le script d'installation

```
./setup_admin.sh
```

Le script va :

1. Vérifier que Python 3 est installé
2. Vérifier la présence de `admin_server.py` et `manage_admin.sh`
3. Enregistrer `127.0.0.1` comme `sdr_host` dans la configuration (voir [Le réglage `sdr\_host`](#))
4. Demander trois numéros de port. Chaque invite propose une valeur par défaut entre crochets qu'une simple touche Entrée accepte, si bien qu'une installation normale tient en trois frappes :
5. **Port de spectrumserver** — le port sur lequel votre serveur SDR écoute (par défaut `8900`, c'est-à-dire celui que livre chaque `config-*.toml` du dépôt)
6. **Port interne du panneau d'administration** — là où `admin_server.py` se lie localement (par défaut `3000`)
7. **Port public du proxy** — l'unique port externe combinant SDR + administration (par défaut `8902`)

Les réponses invalides sont rejetées et redemandées, mais cinq fois seulement — ensuite la valeur par défaut est retenue. Une exécution dont l'entrée n'est pas un terminal (tube, cron, sans surveillance) prend aussitôt les valeurs par défaut au lieu d'attendre une saisie qui ne viendra jamais.

1. Installer `flask`, `psutil`, `aiohttp` et `tomli-w` via `pip`
2. Accorder à `ss` la capacité `cap_net_admin` (voie de repli de l'éjection d'utilisateurs, utile seulement si le panneau tourne sans le proxy)
3. Demander quel script démarre et quel script arrête le récepteur — c'est par eux que le panneau pilote le serveur, et la protection thermique aussi
4. Demander jusqu'où la protection thermique du processeur peut aller d'elle-même — `log`, `throttle`, `stop` ou `stop+restart` — et proposer d'exécuter `setup-cpufreq-perms.sh` pour que l'étage throttle fonctionne sans root (voir [THERMAL_GUARD.md](#))
5. Écrire `admin_config.json` avec tous les réglages
6. Proposer d'installer deux unités `systemd` (panneau + proxy) — démarrage au boot et redémarrage après un plantage. **Recommandé, et proposé par défaut** (un simple Entrée suffit) : la protection contre la surchauffe ne tourne que tant que le panneau tourne ; sans les unités, un redémarrage laisse la machine sans protection. L'étape est ignorée automatiquement là où `systemd` n'est pas PID 1 (conteneurs, WSL1, OpenRC) ou si vous n'avez pas `sudo`.

Après l'installation, ouvrez votre navigateur via le proxy :

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

Mot de passe par défaut : `admin`

⚠ **Changez immédiatement le mot de passe** — allez dans Settings à la première connexion. Un assistant de premier lancement vous guidera pour définir le répertoire SDR, le nom du processus, le port public et un nouveau mot de passe.

Étape 3 — Démarrer / arrêter / redémarrer

Il y a deux façons de faire tourner le panneau et le proxy. **Choisissez-en une et tenez-vous-y** — l'avertissement en fin de section explique ce qui se passe si vous les mélangez.

La méthode B (systemd) est fortement recommandée partout où systemd est disponible. La protection thermique tourne dans le panneau : un panneau qui meurt à 03:00, ou une machine qui redémarre, laisse le processeur sans protection jusqu'à ce que quelqu'un le relance à la main. `setup_admin.sh` propose désormais les unités par défaut. Utilisez la méthode A lorsqu'il n'y a pas de systemd — conteneurs, WSL1, OpenRC/sysvinit — ou si vous voulez délibérément assurer vous-même la supervision.

Méthode A — `manage_admin.sh` (sans root, rien à installer)

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

Il n'y a délibérément **pas de restart**. Il tuait le panneau et démarrait aussitôt sa propre copie, ce qui entre en conflit avec systemd sur toute machine utilisant la méthode B. Pour redémarrer avec la méthode A, lancez `stop` puis `start` — voir [Redémarrer le panneau](#) ci-dessous.

Un seul script pilote **les deux** processus. Il vérifie qu'`aiohttp` est installé avant de démarrer le proxy et affiche une erreur claire avec la commande de correction s'il manque.

Rien ne démarre tout seul : après un redémarrage, ou si le panneau plante, vous le relancez à la main. C'est acceptable pour un récepteur que vous surveillez, mais notez que la protection thermique du processeur tourne à l'intérieur du panneau : tant que le panneau est arrêté, rien ne protège la machine (voir [THERMAL_GUARD.md](#)).

Méthode B — unités systemd (démarrage au boot, redémarrage après plantage)

Le dépôt fournit deux unités prêtes à l'emploi, `phantomsdr-admin.service` et `phantomsdr-proxy.service`. Adaptez `User=` et les chemins dans les deux, puis :

```
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
```

Les commandes quotidiennes deviennent alors :

```
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
```

Les mêmes quatre avec `phantomsdr-proxy`, ou les deux à la fois : `sudo systemctl restart phantomsdr-admin phantomsdr-proxy`.

`admin.log` et `proxy.log` fonctionnent exactement comme avant — les unités écrivent dans ces mêmes fichiers.

Une chose à faire une fois pour toutes : installer la règle `tmpfiles` de `tmpfiles/phantomsdr-logs.conf` (modifiez d'abord ses chemins et son utilisateur). systemd crée ces deux journaux en tant que `root` avant de descendre vers `User=`, et le bouton **Clear Logs** du panneau échoue alors avec `Permission denied` — voir plus bas « Clear Logs indique Permission denied ». `setup_admin.sh` installe la règle pour vous.

L'unité du panneau porte `SupplementaryGroups=cpufreq`, ce qui permet à l'étage throttle de la protection thermique d'abaisser la fréquence du processeur sans faire tourner le panneau en root. Créez d'abord ce groupe avec `./setup-cpufreq-perms.sh`, sinon systemd refusera de démarrer l'unité ; supprimez la ligne si vous n'utilisez pas l'étage throttle.

Pour revenir à la méthode A :

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

⚠ **Ne mélangez pas les deux.** Avec les unités activées, `./manage_admin.sh stop` est annulé par systemd cinq secondes plus tard, et `./manage_admin.sh start` vous donne une seconde copie non gérée qui se dispute le même port avec celle de systemd. (`restart` a été retiré du script précisément pour cela ; il affiche désormais une erreur qui renvoie à `systemctl`.) `./manage_admin.sh status` ne fait que lire : il reste utile dans les deux cas.

Redémarrer le panneau

La suite de ce document dit « redémarrez le panneau » à plusieurs endroits. Cela désigne celle de ces commandes qui correspond à votre méthode :

```
# Méthode B – unités systemd (l'installation normale)
sudo systemctl restart phantomsdr-admin phantomsdr-proxy

# Méthode A – aucune unité installée
./manage_admin.sh stop && ./manage_admin.sh start
```

L'unité fournit `KillMode=process`, et cette ligne est déterminante. Si vous démarrez le SDR depuis le panneau, le récepteur est un fils de l'unité d'administration et hérite de son cgroup. Avec la valeur par défaut de systemd, `KillMode=control-group`, redémarrer le panneau emporterait `spectrumserver`, son chien de garde et le démon autorun — après avoir bloqué les 90 secondes entières du délai d'arrêt. Avec `KillMode=process`, systemd ne signale que le panneau, si bien que `sudo systemctl restart phantomsdr-admin` laisse à l'antenne un récepteur occupé.

Mettre à niveau une unité installée avant la v4.1.0. Les anciens fichiers d'unité ne comportent pas cette ligne, et redémarrer le panneau ne l'ajoute pas : `systemctl restart` relance le programme, il ne modifie pas sa configuration. Modifiez la copie *installée* — celle du dépôt n'est qu'un modèle que systemd ne lit jamais :

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

Ajoutez `KillMode=process` dans la section `[Service]`, à côté de `Restart=always`. Faites ensuite relire le fichier par systemd et vérifiez :

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

La seconde commande doit afficher `KillMode=process`. Si elle affiche encore `control-group`, systemd utilise sa copie en cache — `systemctl show phantomsdr-admin -p NeedDaemonReload` renvoie `yes` tant qu'un rechargement est en attente. Les espaces en début de ligne sont ignorés dans les fichiers d'unité : l'indentation n'a donc aucune importance.

Étape 4 — Ouvrir le port du pare-feu

N'ouvrez que le **port public du proxy** dans votre pare-feu. Il est inutile d'exposer le port interne du panneau d'administration :

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

Étape 5 — Fonction d'éjection d'utilisateurs

La page Users affiche chaque auditeur actuellement connecté. Chaque ligne indique l'IP de l'utilisateur, sa fréquence, son mode et depuis combien de temps il est connecté. Pour déconnecter un utilisateur, cliquez sur le bouton **Kick** de sa ligne — seule cette connexion TCP est coupée. Tous les autres auditeurs restent connectés et n'entendent rien.

L'éjection est instantanée et chirurgicale : le serveur ne redémarre pas, l'audio n'est interrompu pour personne d'autre, et l'utilisateur éjecté peut se reconnecter immédiatement (la fonction sert à supprimer des connexions problématiques ou bloquées, pas à bannir).

En interne, l'éjection utilise `ss -K dst <IP> dport <port>` pour fermer le socket TCP concerné. Cela nécessite une capacité Linux que le script d'installation accorde automatiquement. Si vous avez sauté cette étape ou si le bouton signale une erreur, appliquez-la manuellement :

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

Configuration — admin_config.json

Toute la configuration d'exécution se trouve dans `admin_config.json`, dans votre répertoire PhantomSDR-Plus. `setup_admin.sh` écrit le fichier initial ; les modifications ultérieures peuvent se faire via la page Settings ou en éditant directement le fichier.

Champs clés écrits par `setup_admin.sh` :

Clé	Description
<code>port</code>	Port interne du panneau d'administration (<code>admin_server.py</code> s'y lie)
<code>public_port</code>	Port de spectrumserver (utilisé par la page Users pour interroger <code>/users</code>)
<code>proxy_port</code>	Port public sur lequel le proxy écoute
<code>sdr_host</code>	Hôte utilisé par le proxy pour atteindre spectrumserver — <code>127.0.0.1</code> (voir ci-dessous)
<code>password_hash</code>	Empreinte SHA-256 du mot de passe d'administration
<code>sdr_base_dir</code>	Chemin de votre installation PhantomSDR-Plus
<code>sdr_process_name</code>	Nom du processus à surveiller (par défaut : <code>spectrumserver</code>)
<code>start_script</code> / <code>stop_script</code>	Scripts que le panneau exécute pour démarrer et arrêter le serveur SDR — c'est aussi par eux qu'agit la protection thermique

Plusieurs récepteurs sur un ordinateur ? Mettez `start_script` à `start-all.sh`. Le script d'arrêt arrête déjà tous les récepteurs, et Restart ainsi que la protection thermique utilisent le script de démarrage ; avec `start-all.sh`, ils ramènent tous les récepteurs de `receivers.toml` au lieu du seul principal. La page Users, le nombre d'auditeurs et l'expulsion couvrent alors aussi tous les récepteurs. Voir [Plusieurs récepteurs](#).

Clés de la protection thermique (toutes facultatives — le panneau les écrit lorsque vous enregistrez les réglages Thermal Guard, et celles qui manquent prennent ces valeurs par défaut) :

Clé	Défaut	Description
<code>thermal_enabled</code>	<code>true</code>	Interrupteur général de la protection
<code>thermal_mode</code>	<code>"log"</code>	<code>log</code> (journaliser seulement, n'agit jamais) · <code>throttle</code> · <code>stop</code> · <code>stop+restart</code>
<code>thermal_warn</code>	<code>null</code>	Température d'avertissement °C ; <code>null</code> = déduire du seuil critique de votre processeur
<code>thermal_throttle</code>	<code>null</code>	Température à laquelle le plafond de fréquence est abaissé ; <code>null</code> = automatique
<code>thermal_stop</code>	<code>null</code>	Température à laquelle le serveur est arrêté ; <code>null</code> = automatique
<code>thermal_resume</code>	<code>null</code>	Température sous laquelle le processeur doit redescendre pour récupérer ; <code>null</code> = automatique
<code>thermal_sustain_s</code>	<code>60</code>	Secondes pendant lesquelles la température d'arrêt doit être tenue
<code>thermal_warn_sustain_s</code>	<code>30</code>	Secondes pendant lesquelles les températures d'avertissement/throttle doivent être tenues
<code>thermal_resume_s</code>	<code>300</code>	Secondes sous la température de récupération avant la levée du verrou
<code>thermal_max_stops_hour</code>	<code>2</code>	Arrêts thermiques par heure avant désactivation du redémarrage automatique
<code>thermal_test_temp</code>	<code>null</code>	Température fictive pour tester la protection ; <code>null</code> = utiliser le vrai capteur

Ces changements prennent effet à la mesure suivante — en quelques secondes, sans redémarrage.

Pour changer de ports après l'installation initiale, modifiez `admin_config.json` et redémarrez :

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

Le réglage `sdr_host`

`proxy.py` se connecte à `spectrumserver` par bouclage — `sdr_host` vaut `127.0.0.1` dans `admin_config.json`, écrit là par `setup_admin.sh`. Vous ne devriez pas avoir à le modifier.

Ne mettez une adresse réelle que si `spectrumserver` tourne sur une **autre machine** que le proxy. Ensuite, [Redémarrer le panneau](#).

Changement : les versions antérieures exigeaient que `sdr_host` soit l'IP LAN de la machine, car `spectrumserver` fermait les WebSocket arrivant de l'adresse de bouclage. Ce filtre a été supprimé, les connexions locales fonctionnent donc normalement. Deux conséquences : ouvrir `http://localhost:<port>` sur la machine du serveur elle-même affiche désormais l'interface complète (auparavant la page se chargeait mais restait vide), et le proxy ne casse plus quand le DHCP réattribue l'IP de la machine. Si vous mettez à jour et que votre `admin_config.json` contient encore une IP LAN, remplacez-la par `127.0.0.1` ou relancez `setup_admin.sh`.

proxy.py — ce que c'est et quand il est nécessaire

`proxy.py` place à la fois le serveur SDR et le panneau d'administration sur un **seul port public**, en routant par préfixe de chemin :

Chemin	Redirigé vers
<code>/admin*</code>	Panneau d'administration (<code>localhost:<port></code>)
Tout le reste	Spectrumserver (<code><sdr_host>:<public_port></code>)

Sans le proxy, vous devez exposer deux ports séparément. Avec lui, vous n'en exposez qu'un.

Le proxy supprime également la limite de 4 Mo sur la taille des messages WebSocket (important pour les grandes trames FFT du RX-888 à 60 MSPS), transmet les vraies IP des clients via les en-têtes `X-Forwarded-*` et maintient les connexions de longue durée à travers le NAT grâce à un battement de cœur toutes les 30 secondes.

Changer de ports après l'installation

Modifiez directement `admin_config.json` :

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

Puis redémarrez les deux services :

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

Si vous avez également besoin que `admin_server.py` se lie à un autre port au lancement (p. ex. pour le service `systemd`), vous pouvez le remplacer par la variable d'environnement :

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

Par défaut, `admin_server.py` ne se lie qu'à `127.0.0.1`. Pour l'exécuter seul sans le proxy (développement/tests), liez-le à toutes les interfaces :

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

Commandes manuelles utiles

```
# 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
```

Fichiers journaux

Fichier	Contenu
admin.log	Sortie du panneau d'administration
proxy.log	Bannière de démarrage du proxy + une ligne de journal d'accès par requête relayée
logwebsdr.txt	Journal du lanceur et du chien de garde — démarrage/arrêt, pilote du récepteur, enregistrement sur websdr.org
spectrumserver.log	La sortie du processus récepteur lui-même — mise en place FFT et OpenCL, connexions de chaque client, erreurs. Le plus gros de ces fichiers ; il est tourné, un spectrumserver.log. 1 peut donc se trouver à côté
rade.log	Journal du sidecar RADE/FreeDV
autorun.log	Journal du démon de décodage — spots, SNR, dérive
crash.log	Rapports de plantage et événements thermiques : traces [CRASH] / [TERMINATE] , notes [EXIT] et lignes [THERMAL] de la protection. Normalement vide — tout ce qu'il contient mérite d'être lu

Tous ces fichiers sont consultables depuis les onglets **Log Viewer** du panneau (LOGWEBSDR, SPECTRUMSERVER, ADMIN, RADE, CRASH, AUTORUN, PROXY). Le visualiseur affiche les 150 dernières lignes de l'onglet choisi et se rafraîchit toutes les 3 secondes quand le rafraîchissement automatique est actif ; il ne lit que la fin du fichier, sa taille est donc sans importance.

Rotation de proxy.log et admin.log

proxy.log enregistre chaque requête relayée et admin.log chaque requête qui atteint le panneau lui-même. Aucun des deux ne fait l'objet d'une rotation par défaut. L'interrogation du tableau de bord dominait autrefois les deux fichiers — /admin/api/status est appelée toutes les ~3 secondes tant qu'un onglet reste ouvert — mais ces interrogations sont désormais filtrées des deux journaux (voir plus bas) ; la rotation sert donc à borner une croissance lente, pas un flot.

Le dépôt fournit une configuration logrotate dans logrotate/phantomsdr (quotidienne, ou plus tôt si un journal dépasse 10 Mo ; conserve 7 archives compressées dans logproxy/, afin qu'elles n'encombrent pas la racine du projet). **Elle dépend du site — ouvrez-la et changez les trois chemins de journaux, le chemin olddir et l'utilisateur su pour correspondre à votre machine avant de l'installer**, puis :

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

`logrotate/phantomsdr` dans le dépôt et `/etc/logrotate.d/phantomsdr` sont deux copies **indépendantes** — `cp` ne les relie pas. Modifier le fichier du dépôt ne change rien sur un système en fonctionnement tant que vous n'avez pas relancé le `sudo cp` ci-dessus. Seule la copie dans `/etc` est réellement lue par `logrotate`.

La configuration utilise `copytruncate`, ce qui est indispensable : `manage_admin.sh` démarre les deux processus avec `>> <log>` et aucun ne rouvre sa sortie standard ; une rotation par renommage les laisserait donc écrire dans le fichier pivoté tandis que le nouveau journal resterait éternellement vide.

Les archives arrivent dans `logproxy/` (créé automatiquement par `createolddir`) et sont nommées `proxy.log.1.gz ... proxy.log.7.gz`, la plus récente en premier. Lisez-en une avec `zcat logproxy/proxy.log.1.gz | less`, ou cherchez dans toutes à la fois avec `zgrep "pattern" logproxy/*.gz`.

Le bouton **Clear Logs** du panneau tronque **tous** les fichiers du tableau ci-dessus, `autorun.log` et `proxy.log` compris (`CLEAR_EXCLUDE` dans `admin_server.py` est vide). Notez ce que cela coûte sur `autorun.log` : son historique de décodage est le seul enregistrement de ce qui a été entendu et quand, et il est impossible à reconstituer ; d'un vidage il ne survit que ce que `logrotate` a déjà basculé dans `logproxy/`, donc tout ce qui a été décodé depuis la dernière rotation est perdu. Le vider sous un décodeur en marche est sans danger : le démon le garde ouvert en `O_APPEND` et continue d'écrire dans le même inode. La notification indique ce qui a été vidé, et si un fichier n'a pas pu être écrit — un `proxy.log` appartenant à un autre utilisateur, par exemple, sur une installation à comptes séparés — elle nomme ce fichier et l'erreur au lieu d'annoncer une réussite, et laisse le volet de ce journal à l'écran au lieu de le vider. Les fichiers sont tronqués sur place, jamais supprimés : `manage_admin.sh` comme les unités `systemd` les ouvrent en `O_APPEND`, si bien qu'un fichier supprimé continuerait de remplir un inode invisible jusqu'au redémarrage du service.

Les interrogations réussies sont filtrées des **deux** journaux d'accès, par le `QuietPollFilter` de `proxy.py` et par celui d'`admin_server.py`. Le tableau de bord appelle `/admin/api/status` toutes les 3 secondes, et `/admin/api/logs`, `/admin/api/autorun/status`, `/admin/api/users`, `/admin/api/graph-stats` et `/admin/api/chat` toutes les 5 secondes ; sans filtrage elles représenteraient ~95 % des deux fichiers. Seuls les 200 simples sur ces chemins sont écartés — tout autre statut, chemin ou méthode est journalisé comme avant, de sorte qu'une interrogation en échec reste visible et que les actions modifiant l'état (`kick`, `autorun/start`, ...) passent par des chemins distincts et sont toujours enregistrées. La seule exception délibérée est `/admin/api/logs/clear`, filtré dans les deux fichiers : sa propre ligne d'accès est écrite *après* la troncature, si bien que sans le filtre chaque effacement réussi laissait une ligne toute fraîche dans le journal qu'il venait de vider, et le bouton paraissait cassé.

Comme `proxy.log` est un journal d'accès, il contient les adresses IP des visiteurs et les chaînes `user-agent` — gardez-le à l'esprit avant de le partager pour demander de l'aide.

« Clear Logs » indique Permission denied

Sur une machine où les unités `systemd` ont été installées sur une arborescence vierge, le bouton peut répondre :

```
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)
```

Ce sont toujours ces deux fichiers et jamais les cinq autres, car ce sont les seuls que `systemd` crée lui-même. `StandardOutput=append` est ouvert par PID 1 **avant** de descendre vers `User=` ; un journal qui n'existe pas encore est donc créé en `root:root 0644`. Le panneau tourne sous votre utilisateur et ne peut pas le vider. Les cinq autres journaux sont créés par les scripts de lancement, qui tournent déjà

sous votre utilisateur : ils se vident normalement. Là où les deux fichiers sont antérieurs aux unités — `manage_admin.sh` les a créés avec `>>` —, ils appartiennent déjà au bon utilisateur et le bouton fonctionne ; c'est pourquoi le problème n'apparaît que sur une installation neuve partie directement de `systemd`.

Vérifiez avec `ls -l admin.log proxy.log` : ceux qui échouent affichent `root root`. Réparez-les :

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

Aucun redémarrage n'est nécessaire — `systemd` continue d'écrire via le descripteur de fichier qu'il détient déjà, et l'ajout ne change pas la propriété. Appuyez de nouveau sur **Clear Logs** et les deux sont vidés.

Ne supprimez pas les fichiers à la place. Tous deux sont ouverts en `O_APPEND` ; en supprimer un laisse `systemd` remplir un inœud invisible jusqu'au redémarrage du service, et le fichier qui réapparaît appartient de nouveau à `root`.

Pour que cela ne revienne pas, installez la règle `tmpfiles` fournie par le dépôt dans `tmpfiles/phantomsdr-logs.conf`. Elle recrée les deux journaux avec le bon propriétaire à chaque démarrage, avant le lancement des unités, de sorte qu'un journal supprimé — nettoyage manuel, réinstallation, répertoire d'installation déplacé — ne revienne jamais appartenant à `root`. **Elle est spécifique à la machine : ouvrez-la et modifiez d'abord les deux chemins et le `user:group` pour qu'ils correspondent au vôtre**, puis :

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

Comme pour `logrotate/phantomsdr`, le fichier du dépôt et la copie dans `/etc` sont indépendants — modifier celui du dépôt ne change rien tant que vous n'avez pas relancé le `cp`. À partir de `systemd 254`, la règle répare aussi la propriété et le mode d'un journal déjà présent : elle corrige donc l'échec actuel autant que les suivants ; sur les versions plus anciennes, c'est le `chown` ci-dessus qui débloque le bouton.

`setup_admin.sh` fait tout cela pour vous lorsqu'il installe les unités : une installation faite avec le script ne rencontre jamais ce problème. Les étapes ci-dessus concernent une paire installée à la main, ou une installation antérieure à l'étape de propriété des journaux du script.

`Logrotate` ne réintroduit pas le problème : la configuration fournie utilise `copytruncate`, qui conserve le même inœud et donc le même propriétaire.

Résumé des accès

Configuration	SDR	Panneau d'administration
Sans proxy	<code>http://YOUR_IP:<public_port></code>	<code>http://YOUR_IP:<port>/admin</code>
Avec proxy	<code>http://YOUR_IP:<proxy_port></code>	<code>http://YOUR_IP:<proxy_port>/admin</code>

Suppression de messages du chat

La page Chat History liste tous les messages du journal de chat actuel. Chaque entrée dispose d'un bouton **Delete** qui retire immédiatement ce seul message du journal — sans redémarrer le serveur.

Fonctionnement :

- Le panneau d'administration réécrit le fichier de journal du chat sur place, en supprimant uniquement la ligne du message sélectionné.
- Le changement prend effet pour tout utilisateur qui recharge le chat ; l'historique déjà chargé dans les onglets ouverts n'est pas mis à jour rétroactivement.
- Effacer tout le journal (bouton **Clear All**) tronque le fichier, ce qui prend également effet sans redémarrage.
- Une **réponse du chat** est stockée comme un message ordinaire qui commence par un marqueur nommant le message auquel elle répond, par exemple `[RE:2026-09-25 12:00:01|SV1BTL]` ; dans cette liste elle apparaît donc précédée de ce marqueur. Supprimer une réponse ne retire que cette réponse. Supprimer le message auquel elle répond laisse ses réponses en place, n'affichant plus que le nom et l'heure auxquels elles répondent.

Le bouton de suppression est affiché de manière cohérente dans les cinq variantes de l'application Svelte. Si une suppression semble sans effet, vérifiez que le panneau d'administration a le droit d'écriture sur le fichier de journal du chat :

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

Messages diffusés sur la cascade

Le panneau **Waterfall Message** permet de pousser une bannière de texte persistante sur l'affichage en cascade de chaque utilisateur connecté, sans toucher au processus du serveur.

Envoyer un message

1. Ouvrez le panneau d'administration et allez dans **Waterfall Message**.
2. Saisissez le texte de votre message et choisissez une couleur (hexadécimale, p. ex. `#ffdd00`).
3. Cliquez sur **Send** — le message apparaît immédiatement sur toutes les vues en cascade actives.

Effacer le message

Cliquez sur **Clear** pour retirer la bannière de toutes les cascades. L'état du message est conservé en mémoire par le panneau d'administration ; il est effacé automatiquement si le processus du panneau redémarre.

Fonctionnement

Le panneau d'administration expose deux points de terminaison internes que le proxy relaie :

Point de terminaison	Méthode	Objet
<code>/admin/api/waterfall-message</code>	POST	Définir ou effacer le texte et la couleur de la bannière
<code>/admin/api/waterfall-message</code>	GET	Renvoyer l'état actuel du message au format JSON

Le frontend interroge l'état du message et l'affiche en superposition sur le canevas de la cascade. Aucune reconnexion WebSocket ni rechargement de page n'est nécessaire côté client.

Usages typiques

- Annoncer une maintenance programmée : `"Server restart in 10 minutes"`
- Signaler les conditions de propagation : `"Solar flux 180 – 10m wide open"`

- Message de bienvenue : "Welcome to SV1BTL WebSDR – Athens, KM17"

Graphiques système

La page **Graphiques** trace quatre mesures sur un axe de temps commun :

- **Fréquence du CPU** — fréquence moyenne sur tous les cœurs, en GHz
- **Charge du CPU** — utilisation totale, en pourcentage
- **Température du CPU** — en °C, avec des repères pointillés à 70 °C (chaud) et 80 °C (critique)
- **Utilisateurs en ligne** — auditeurs connectés au WebSDR à cet instant

Choisissez la plage avec **15 MIN / 1 HOUR / 4 HOURS / 12 HOURS / 24 HOURS**. Le survol du graphique affiche un réticule et les quatre valeurs de cet instant. Les quatre cartes au-dessus montrent toujours l'échantillon le plus récent.

Comment se fait l'échantillonnage

Un thread d'arrière-plan de `admin_server.py` prend un échantillon toutes les **2 secondes**. Les échantillons sont conservés sur deux niveaux : la dernière **heure** en pleine résolution de 2 secondes (1800 points) et **24 heures** de moyennes sur 30 secondes (2880 points) pour les plages longues. La page choisit le niveau qui couvre la plage demandée et affiche `30s averages` dans l'en-tête lorsqu'il s'agit du niveau moyenné.

Conserver une journée entière en résolution 2 secondes représenterait ~43000 points : des dizaines de Mo en mémoire, un premier transfert de plusieurs Mo et environ 40 points par pixel, ce qu'aucun écran ne peut afficher. La moyenne sur 30 secondes tient une journée en 2880 points.

L'échantillonnage démarre avec le panneau d'administration, et non à la première ouverture de la page : celle-ci s'ouvre donc sur un historique déjà constitué.

Le tampon est **uniquement en mémoire** : rien n'est écrit sur le disque et l'historique est perdu au redémarrage du panneau. C'est voulu — cela garde la fonction hors de la rotation des journaux et hors du disque.

2 secondes donnent une vue quasi temps réel et un intervalle assez court pour ne pas fausser la fréquence du CPU, qui sur un processeur moderne passe de son horloge de repos à son turbo d'un échantillon à l'autre. Un échantillon coûte environ une milliseconde : l'échantillonneur est donc négligeable face au serveur SDR. La page interroge au rythme de chaque niveau — toutes les 2 secondes sur les plages en direct, toutes les 30 secondes sur les plages moyennées, car demander plus souvent que les points ne sont produits ne renvoie que des réponses vides.

Pour changer la cadence ou la rétention, modifiez ces constantes au début du bloc d'échantillonnage dans `admin_server.py`, puis redémarrez :

```
GRAPH_INTERVAL_S    = 2          # secondes entre les échantillons fins
GRAPH_FINE_S        = 3600       # profondeur du niveau fin (1 h)
GRAPH_COARSE_EVERY  = 15         # échantillons fins par point agrégé (15 x 2 s = 30 s)
GRAPH_COARSE_S      = 24 * 3600  # profondeur du niveau agrégé (24 h)
```

D'où viennent les chiffres

Mesure	Source
Fréquence du CPU	<code>psutil.cpu_freq()</code> , à défaut <code>/sys/devices/system/cpu/cpu*/cpufreq/</code> . Ce sont les données noyau que <code>cpufreq-info</code> met en forme : cpufrequtils n'est pas requis .
Charge du CPU	<code>psutil.cpu_percent()</code>
Température du CPU	<code>psutil.sensors_temperatures()</code> , à défaut <code>/sys/class/thermal</code> , puis la commande <code>sensors</code>
Utilisateurs en ligne	Le point d'accès <code>/users</code> de spectrumserver lui-même — la liste de sessions faisant foi. Compter les sockets avec <code>ss</code> ne voit pas les vraies IP des clients derrière proxy.py.

Si une mesure n'est pas disponible sur votre machine, le panneau le dit au lieu de tracer une ligne à zéro. Une lecture en échec interrompt la courbe, pour qu'un trou ne ressemble jamais à une valeur mesurée.

Pourquoi quatre panneaux et non un seul graphique

Des GHz, des pourcentages, des degrés et un nombre de personnes n'ont pas d'échelle commune. Les réunir demanderait plusieurs axes y, et les croisements ainsi produits sont des artefacts de l'échelle, non des faits sur le serveur. Quatre panneaux empilés sur un même axe de temps gardent la comparaison honnête. Chaque panneau part de zéro, de sorte que la hauteur de la courbe reste proportionnelle à la valeur.

La couleur est réservée à la température, où l'ambre et le rouge marquent les seuils ci-dessus. Les autres panneaux partagent une seule couleur, car chacun ne contient qu'une série que son titre nomme déjà.

Point d'accès

`GET /admin/api/graph-stats?since=<epoch>` (connexion requise) renvoie les échantillons postérieurs à `since`, ce qui permet à la page d'interroger de façon incrémentale au lieu de retélécharger tout le tampon à chaque cycle.

Thermal Guard

Manuel complet : [THERMAL_GUARD.md](#) — les quatre modes et ce que le sysop doit faire pour chacun, le fonctionnement sans le panneau, l'étage throttle sans root, les tests et le dépannage.

Arrête le serveur SDR lorsque le processeur devient dangereusement chaud et le relance une fois refroidi. Elle fait partie du panneau — rien de plus à installer — et apparaît comme une carte **THERMAL GUARD** sur le Tableau de bord, ses réglages se trouvant dans **Paramètres**.

Elle est livrée **inerte** : le mode démarre sur `log`, elle se contente donc d'enregistrer ce qu'elle *aurait* fait. Rien ne touche à votre serveur tant que vous ne changez pas le mode.

Elle fonctionne avec n'importe quelle méthode de démarrage/arrêt

La protection ne cherche jamais à savoir ce qui supervise votre serveur. Elle agit via les scripts que vous avez définis comme **Default start script** et **Default stop script** dans Paramètres, et tant que le processeur est en surchauffe elle réémet l'arrêt à **chaque vérification** (toutes les 2 s). Tout ce qui relance le serveur — le watchdog contenu dans `start-*.sh`, une unité systemd, une tâche cron, une session `tmux` — est défait en quelques secondes, jusqu'à ce que la machine ait refroidi. C'est le **verrouillage** (lockout).

Si aucun script d'arrêt n'est configuré, elle envoie `SIGTERM` au processus nommé dans `sdr_process_name`, puis `SIGKILL` après un délai de grâce de 10 secondes.

Les seuils viennent de votre propre processeur

Plutôt qu'un nombre fixe qui serait faux sur la plupart des machines, la protection lit le seuil critique que le noyau publie pour votre processeur et remonte à partir de là :

Votre processeur annonce	avertissement	throttle	arrêt	récupération
Intel, limite 100 °C	88	92	95	75
AMD Ryzen, limite 95 °C	83	87	90	70
Raspberry Pi, limite 85 °C	73	77	80	60

C'est important : 95 °C fixes sont presque normaux sur un Intel, mais sur un Ryzen — dont la Tctl se tient à 95 °C par conception sous boost — cela arrêterait le serveur sur une machine en parfaite santé, et sur un Pi, qui limite dès 80 °C, cela ne se déclencherait jamais. Chaque seuil peut être fixé à la main dans Paramètres ; laissez le champ vide pour *automatique*.

Pour voir comment la protection perçoit votre machine :

```
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
```

Les quatre étapes

Arrêter le serveur est le dernier recours, pas le seul outil :

1. **warn** — journalisé, rien d'autre.
2. **throttle** — abaisse de 20 % le plafond de fréquence du processeur. Nécessite un pilote `cpufreq` et le droit d'écriture sur `scaling_max_freq`, donc normalement `root` ; là où cela manque (le cas courant d'un panneau sans privilèges, et sur la plupart des VPS) l'étape **se désactive d'elle-même** et la protection passe de `warn` directement à `stop`. `./setup-cpufreq-perms.sh` accorde ce droit d'écriture à un groupe `cpufreq`, si bien que l'étape fonctionne sans `root` — voir [THERMAL_GUARD.md](#).
3. **stop** — exécute votre script d'arrêt et maintient le serveur à terre (voir le verrouillage).
4. **resume** — une fois le processeur resté sous la température de récupération pendant `thermal_resume_s` (5 minutes par défaut), le plafond de fréquence est rétabli et, en mode `stop+restart`, votre script de démarrage est relancé.

Chaque étape doit être **tenue** pendant sa durée — 30 s pour `warn` et `throttle`, 60 s pour `stop` — de sorte qu'un pic bref pendant une compilation ne la déclenche jamais.

Les déclenchements répétés sont plafonnés par `thermal_max_stops_hour` (2 par défaut). Une fois ce plafond atteint, le *redémarrage automatique* est désactivé tandis que la protection continue de fonctionner, afin qu'une machine en surchauffe n'oscille pas entre marche et arrêt.

Quels capteurs sont utilisés

Uniquement de vrais capteurs sur la puce : `coretemp` (Intel), `k10temp` / `zenpower` (AMD), `cpu_thermal` / `soc_thermal` (ARM, Raspberry Pi). C'est la lecture la plus élevée de la puce qui est retenue, pas la moyenne.

`acpitz` et les zones thermiques non étiquetées sont **délibérément ignorés** : ils rapportent souvent une température de boîtier ou de carte inférieure de plusieurs dizaines de degrés au processeur réel, si bien qu'une protection qui leur ferait confiance ne se déclencherait jamais au moment voulu. Si votre machine n'a aucun capteur exploitable (fréquent sur un VPS ou dans un conteneur), la carte le dit clairement (`no trusted CPU sensor`) et la protection reste inactive plutôt que de faire semblant de vous protéger.

Ce qu'elle écrit

Tout va dans `crash.log`, c'est-à-dire l'onglet **CRASH** du Log Viewer, un événement par ligne :

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

La ligne de verrouillage est limitée à une par minute, pour qu'un superviseur qui résiste sans cesse n'inonde pas le journal.

Testez-la avant de lui faire confiance

Réglez **TEST TEMPERATURE** dans Paramètres sur une valeur supérieure à votre seuil d'arrêt. La protection la traite comme une vraie mesure, si bien que tout le parcours — `warn`, `throttle`, `stop`, verrouillage, récupération, redémarrage — se déroule à la demande. Videz le champ pour revenir au capteur réel.

Si vous faites ce test avec le mode sur `stop`, le serveur s'arrête réellement et vos auditeurs sont déconnectés. Faites-le quand personne n'écoute, ou laissez le mode sur `log`, où le test montre ce qui *serait* arrivé sans rien toucher.

Manière recommandée de l'adopter

1. Laissez le mode sur `log` pendant une semaine et continuez normalement.
2. Lisez l'onglet CRASH après vos moments chargés — une recompilation complète, un après-midi de canicule. Si rien n'y figure, votre machine n'a jamais approché.
3. Si les températures relevées vous paraissent justes, passez le mode à `stop` (ou `stop+restart` pour qu'il revienne tout seul).
4. Vérifiez-le une fois avec le champ de température de test, puis laissez-le faire.

Sortir d'un verrouillage

CLEAR LOCKOUT sur la carte du Tableau de bord efface le verrouillage et le plafond de déclenchements. Cela ne désarme pas la protection : si le processeur est encore trop chaud, la vérification suivante arrêtera simplement le serveur à nouveau. À utiliser après avoir corrigé le refroidissement.

Endpoint

GET /admin/api/thermal (connexion requise) renvoie l'état complet de la protection — capteur, seuils, étape, verrouillage, dernier événement. Un POST avec {"action": "reset"} fait la même chose que le bouton CLEAR LOCKOUT.

Sans le panneau d'administration

thermal_guard.py n'utilise que la bibliothèque standard et fonctionne seul, en lisant le même admin_config.json :

```
python3 thermal_guard.py --once           # afficher capteur, seuil critique et seuils
python3 thermal_guard.py                 # exécuter comme service (journalisation seule tant
qu'il n'est pas armé)
python3 thermal_guard.py --mode stop+restart # l'armer sans écrire le moindre fichier de
configuration
python3 thermal_guard.py --config /path/to/other.json
python3 thermal_guard.py --help
```

--mode l'emporte sur thermal_mode du fichier de configuration : une machine sans panneau peut donc être armée directement depuis son unité systemd, sans aucun JSON.

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

L'onglet **Spot Reporting** contrôle le démon autorun (autorun/index.js) — un processus Node.js qui décode FT8/FT4/JS8/WSPR côté serveur, directement depuis le récepteur, et téléverse les spots vers les réseaux de report :

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

JS8 n'est spotté qu'à la vitesse Normal — le cycle d'appel de 15 s, où se trouvent les heartbeats et les CQ. Les spots proviennent des heartbeats, des trames compound et des messages dirigés ; les destinations de groupe comme @ALLCALL et les indicatifs que le décodeur n'a pas pu résoudre (<...>) ne sont jamais rapportés. JS8 partage la file PSK Reporter avec FT8 et FT4 et est compté séparément, exactement comme eux.

Le report est DÉACTIVÉ par défaut. Rien n'est téléversé tant que vous n'avez pas activé une destination et appuyé sur **Start**. Décodage et téléversement sont indépendants : le démon peut décoder et journaliser avec le report désactivé, ce qui vous permet de vérifier l'activité avant que quoi que ce soit ne devienne public.

⚠ Les spots sont téléversés vers des réseaux publics sous **votre indicatif**. N'activez que les bandes/modes que votre récepteur entend véritablement, et indiquez le bon locator.

Prérequis

Le démon autorun nécessite Node.js 22+, les paquets npm ws + cbor-x (résolus via le lien symbolique autorun/node_modules → frontend/node_modules) et util-linux (taskset). Les installeurs mettent tout cela en place automatiquement — voir la [section Rapporteur de spots autorun d'INSTALLATION.md](#). Si **Start** échoue, ce lien symbolique ou taskset en est la cause habituelle (voir Dépannage ci-dessous).

Utiliser l'onglet

1. **Identité** — indicatif et locator. Pré-remplis depuis `frontend/site_information.json` (`siteSysop` / `siteGridSquare`, tronqué à 6 caractères) ; remplacez-les ici si nécessaire.
2. **Matrice bande x mode** — cochez les combinaisons à décoder. Les lignes sont les bandes, les colonnes FT8 / FT4 / JS8 / WSPR ; les cases non prises en charge sont désactivées. WSPR couvre en outre les bandes LF/MF (2200 m, 630 m) et un canal 80 m EU supplémentaire.
3. **Destinations** — activez **PSK Reporter** et/ou **wsprnet** (désactivés par défaut).
4. **Max slots** — limite de sécurité sur le nombre de créneaux bande/mode simultanés.
5. **Start / Stop** — lance/arrête le démon (épinglé sur les cœurs CPU les plus élevés avec `taskset`, choisis automatiquement selon la machine — voir *Ressources requises et limitations* ci-dessous).
Save / Reload enregistrent et relisent la configuration ; **Free All** libère tous les créneaux.

La carte d'état s'actualise toutes les 5 s et indique l'état d'exécution, les compteurs de décodage/téléversement et l'heure du dernier envoi. Un badge **REPORTING** apparaît sur la cascade principale tant que le report est actif.

« 0 sent » pendant les premières minutes est normal. PSK Reporter téléverse par lots toutes les 5 minutes et wsprnet toutes les 2 minutes — les spots patientent jusqu'au prochain envoi. La carte d'état affiche le nombre en attente et la cadence.

Changer de bandes ou de modes en cours de fonctionnement

Les changements ne prennent effet qu'après un redémarrage du démon. `autorun/index.js` lit `autorun.json` une seule fois, au démarrage. Il n'y a ni surveillance de fichier ni signal de rechargement : les seuls signaux traités sont SIGINT et SIGTERM, qui signifient tous deux l'arrêt. Un démon en cours continue donc à décoder les créneaux avec lesquels il a été lancé, quoi que vous enregistriez ensuite.

Pour appliquer un changement :

1. Cochez / décochez les bandes et modes
2. **SAVE CONFIG**
3. **STOP**
4. **START**

Ou simplement **STOP** → modifier les cases → **START**, puisque START enregistre la configuration avant de lancer.

⚠ **Appuyer sur START sans arrêter d'abord n'applique pas le changement.** START enregistre la configuration, puis la requête de démarrage répond `already running` et rien n'est relancé. Vous voyez un message « enregistré » à côté d'une erreur « already running » alors que le démon poursuit avec les **anciennes** bandes — cela ressemble facilement à une réussite. Arrêtez toujours d'abord.

Ce qu'un redémarrage fait aux compteurs :

- **Les totaux à côté de chaque case sont conservés** : ils sont dans `autorun-totals.json`, qui n'est pas supprimé à l'arrêt.
- **Les tuiles par décodeur repartent de zéro**, elles valent par exécution.
- **Rien n'est perdu de la file** : l'arrêt envoie les spots en attente.
- Une bande **nouvellement activée** part de 0 ; une bande **déjà utilisée** reprend son total précédent.
- Une bande que vous **décochez garde son nombre** à côté de la case désormais vide ; l'historique n'est pas effacé.

Le décodage ne s'interrompt que les quelques secondes entre STOP et START.

Compteurs de spots

Deux comptages différents sont affichés ; ils répondent à des questions différentes.

SPOTS UPLOADED PER DECODER — une tuile par décodeur (FT8 / FT4 / JS8 / WSPR) avec les spots envoyés **depuis le dernier démarrage du démon**, et en dessous le nombre de décodages et ceux encore en file. Stop/Start les remet à zéro. Un décodeur dont la destination est désactivée affiche ses décodages et `reporting off` plutôt qu'un simple 0 : ici, zéro envoi est un réglage, pas une panne.

Le nombre à côté de chaque case dans BANDS & MODES correspond aux spots envoyés **depuis toujours** pour cette bande et ce mode. Ils sont conservés dans `autorun-totals.json` et survivent aux redémarrages. **En ambre avec un point devant** (·123), cela signifie que ce créneau a décodé sans encore rien envoyer. C'est normal entre deux envois — PSK Reporter émet toutes les 5 minutes et `wspnrt` toutes les 2 — de sorte que les compteurs FT8/FT4 restent ambre les premières minutes après un démarrage. Ils restent également ambre si la destination de ce décodeur est désactivée. Un 0 simple, sans point, signifie que ce créneau est activé mais n'a encore rien décodé : WSPR reste à zéro plusieurs minutes après un démarrage, car il fonctionne sur un cycle de 2 minutes alors que FT8 compte déjà par centaines. Un créneau que le démon n'a jamais exécuté n'affiche rien. Le survol affiche les envois, les décodages et l'heure du dernier envoi.

FT8 et FT4 partagent une seule file d'envoi vers PSK Reporter : le démon impute donc chaque spot à son propre mode et à sa bande au moment de l'envoi ; la répartition n'est pas estimée après coup à partir des totaux.

Remettre les compteurs à zéro. FREE ALL SLOTS décoche toutes les bandes et tous les modes, désactive les deux destinations, arrête le démon et **efface les compteurs cumulés** : ensuite, plus aucun nombre n'apparaît à côté d'une case. C'est le seul moyen de les remettre à zéro, et l'opération est irréversible. Le bouton arrête le démon et attend sa sortie avant de supprimer `autorun-totals.json`, car le démon réécrit ce fichier à l'arrêt ; le supprimer sous un démon actif ne ferait que ramener les nombres.

Ressources requises et limitations

Le démon est volontairement léger et fonctionne sur du matériel modeste (un i5 4 cœurs convient), mais il existe de vraies limites à connaître.

L'épingleage CPU d'autorun tient compte de la configuration et est automatique. Le démon **autorun** est toujours lancé par le panneau d'administration (`admin_server.py`), son épingleage est donc défini à chaque installation sans configuration : il déduit une plage de cœurs `taskset` du nombre de processeurs, en réservant quelques-uns des cœurs les plus élevés pour le décodage, ou tourne sans épingleage sur ≤ 4 cœurs.

L'épingleage de spectrumserver dépend de votre méthode de démarrage. La façon dont vous lancez spectrumserver varie d'une installation à l'autre (type de SDR, outillage, scripts personnels) ; ce document ne présuppose donc aucun lanceur particulier. Le panneau d'administration ne démarre ni n'épingle spectrumserver — cela dépend entièrement de la commande ou du service que vous utilisez. Si vous voulez tenir spectrumserver à l'écart des cœurs utilisés par le démon autorun, épinglez-le vous-même avec `taskset` dans votre propre commande de lancement (voir la section de remplacement ci-dessous). Sinon, il tourne sans épingleage et l'ordonnanceur du système l'équilibre — correct et sans risque, vous perdez simplement la séparation délibérée des cœurs.

À titre de référence, le démon autorun réserve ces cœurs élevés (si vous épinglez spectrumserver, restez donc sur les cœurs bas pour éviter le chevauchement) :

CPU logiques	Utilisés par le démon autorun	À laisser à spectrumserver
≤ 4	<i>non épinglé</i>	<i>non épinglé</i>
6	5	0-4
8	6-7	0-5
12 (p. ex. hybride 8P+4E)	8-11	0-7
16	12-15	0-11

Sur une machine à **4 cœurs (ou moins)**, il n'y a rien à séparer : le démon autorun tourne donc lui aussi *sans épinglage* et partage tous les cœurs — correct et sans risque, mais la FFT du SDR et les rafales de décodage se disputent les mêmes cœurs.

Limitations connues :

1. **Le vrai goulot d'étranglement est spectrumserver + la largeur de bande du SDR, pas le démon.** Un RX888 à 30 MHz nécessite OpenCL/GPU ; un CPU à peu de cœurs sans GPU capable ne peut pas soutenir cette FFT. Associez le matériel modeste à un SDR plus étroit (RSP1A ≈ 10 MHz, RTL-SDR ≈ 2.4 MHz).
2. **WSPR est le poste le plus coûteux en CPU.** Son décodeur de Fano est un portage JS (~30 s par bande, mono-thread). Le pool de 4 workers exécute 4 décodages en parallèle ; activer **plus de ~4 bandes WSPR** à la fois peut faire déborder la file au-delà du créneau de 120 s, surtout pendant que le SDR se dispute les cœurs. Les décodages FT8/FT4 sont peu coûteux en comparaison.
3. **Mise à l'échelle du nombre de créneaux.** Les chiffres de ~2 à 2,5 cœurs / 600 à 700 Mo correspondent aux ~28 créneaux complets sur une machine à 8 cœurs. Sur 4 cœurs partagés, limitez-vous à quelques bandes (règle : ≤ 6 FT8/FT4 + ≤ 3 WSPR) et surveillez la statistique `queued` du pool.
4. **L'épinglage suppose une topologie hybride Intel** (cœurs bas = P-cores plus rapides). Sur AMD ou sur des CPU à numérotation SMT entrelacée, la répartition automatique reste valable (pas de chevauchement, pas de plantage) mais « les cœurs bas sont plus rapides » peut ne pas être littéralement vrai ; sur une puce 4C/8T avec hyperthreading, les cœurs hauts sont des jumeaux SMT — confinés, pas totalement isolés. Là où la plage automatique n'est pas idéale, utilisez le **remplacement manuel** ci-dessous.
5. **La couverture des bandes est limitée par le récepteur.** La configuration RX888 (`sps=60000000` → Nyquist à 30 MHz) ne peut pas atteindre le 6 m et au-delà ; la table des bandes va de 160 m à 10 m. Les autres SDR couvrent ce que permet leur fenêtre d'accord.
6. **Cadence de report.** PSK Reporter envoie toutes les 5 min, wsprnet toutes les 2 min — « 0 sent » les premières minutes est normal, pas un défaut.

Remplacement manuel de l'épinglage CPU (avancé)

La répartition automatique des cœurs convient à la plupart des machines, mais vous pouvez forcer un épinglage précis là où ce n'est pas le cas (CCX/CCD AMD, ARM big.LITTLE, SMT entrelacé). Deux remplacements indépendants, chacun acceptant une liste de cœurs au format `taskset -c (2-3, 0,2,4, 0-2,5)` ou `none/off/unpinned` pour désactiver l'épinglage. Une valeur invalide est ignorée et la déduction automatique s'applique — une faute de frappe ne peut jamais empêcher un lancement.

Démon autorun — variable d'environnement `AUTORUN_CORES`, ou champ `"cores"` dans `autorun.json` (l'environnement l'emporte). Le champ `"cores"` est conservé lors du **Save** du panneau d'administration : une modification manuelle persiste donc. Deux façons de le définir :

Option A — `autorun.json` (persistant, recommandé). Ajoutez une ligne `"cores"` à la configuration à la racine du dépôt, puis faites **Stop → Start** sur l'onglet Spot Reporting :

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

Option B — variable d'environnement (ponctuel). Le démon est lancé par le panneau d'administration : la variable doit donc figurer dans l'environnement **du panneau** — définissez-la et redémarrez le panneau (l'environnement l'emporte sur la valeur d'authorun.json) :

```
# Méthode A — la variable doit être posée sur le processus qui démarre :
./manage_admin.sh stop && AUTORUN_CORES=2-3 ./manage_admin.sh start

# Méthode B — systemd ignore une variable posée sur la ligne de commande
# systemctl ; placez-la donc dans l'unité :
sudo systemctl set-environment AUTORUN_CORES=2-3
sudo systemctl restart phantomsdr-admin
```

spectrumserver — épinglez-le vous-même en préfixant `taskset -c <cœurs>` à la commande que vous utilisez pour démarrer le serveur. Cela fonctionne avec n'importe quelle méthode de démarrage (lancement direct, chaîne SDR, unité de service, 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
```

⚠ **Faites en sorte que cela survive aux redémarrages.** Un `taskset` en ligne ne s'applique qu'à ce lancement précis. Si quelque chose redémarre automatiquement le serveur (chien de garde, service systemd, entrée cron/@reboot, ...), ajoutez le préfixe `taskset` à l'intérieur du script ou de l'unité qui le lance réellement — sinon le redémarrage perd l'épinglage.

Vérifier que l'épinglage a pris — affichez l'affinité CPU réelle des processus en cours :

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

Valeurs acceptées (pour les deux remplacements) : une liste `taskset -c (2-3, 0,2,4, 0-2,5)`, ou `none/off/unpinned` pour aucun épinglage. Tout ce qui est mal formé est ignoré et la déduction automatique s'applique : une faute de frappe ne peut donc jamais bloquer le démarrage.

Sur un Raspberry Pi / toute machine à ≤ 4 cœurs, vous n'avez normalement besoin d'aucun des deux — le chemin automatique exécute déjà les deux processus sans épinglage, ce qui est le bon choix sur un CPU petit et homogène. Le remplacement s'adresse aux machines plus grandes qui ne sont pas des hybrides Intel.

Fichiers et points de terminaison

Élément	Objet
<code>autorun.json</code>	Configuration enregistrée (identité, créneaux, destinations, nombre max de créneaux, remplacement facultatif <code>cores</code>). Écrite par l'onglet ; ignorée par git.
<code>autorun-status.json</code>	État en direct lu par la carte d'état (pid, compteurs, dernier envoi). Écrit toutes les 15 s ; supprimé à l'arrêt.
<code>frontend/dist/autorun-active.json</code>	Données publiques du badge, servies à <code>/autorun-active.json</code> . Vide quand le report est désactivé.
<code>autorun.log</code>	Sortie standard/erreur du démon.
GET/POST <code>/admin/api/autorun/config</code>	Lire / écrire <code>autorun.json</code> (valide l'indicatif, le locator, les combinaisons bande/mode, la limite de créneaux).
GET <code>/admin/api/autorun/status</code>	État d'exécution (<code>pgrep</code>) + <code>autorun-status.json</code> .
POST <code>/admin/api/autorun/start / stop</code>	Lance (<code>taskset -c <auto> node autorun/index.js</code>) / envoie <code>SIGTERM</code> . La plage de cœurs est déduite du CPU (voir ci-dessus).

Remarque : après une mise à jour de `admin_server.py`, vous devez [Redémarrer le panneau](#) pour charger les nouvelles routes autorun ou une matrice bande/mode actualisée. Le démon s'exécute comme un processus distinct : il survit donc aux redémarrages du panneau d'administration.

Dépannage

Problème	Correctif
Le panneau d'administration ne démarre pas	Consultez <code>admin.log</code> — généralement un paquet Python manquant
Le proxy ne démarre pas	Exécutez <code>python3 -c "import aiohttp"</code> — en cas d'échec : <code>pip3 install aiohttp --break-system-packages</code>
<code>proxy.log</code> est vide	Vérifiez d'abord que le proxy tourne réellement (<code>./manage_admin.sh status</code>). Si oui, vous êtes sur une ancienne version : le lanceur doit utiliser <code>python3 -u</code> (non tamponné — sinon la bannière de démarrage ne sort jamais du tampon de 8 Ko), et le <code>main()</code> de <code>proxy.py</code> doit appeler <code>logging.basicConfig()</code> (sinon le journaliseur d'accès d'aiohttp n'a pas de gestionnaire et chaque ligne de requête est perdue, car <code>web.AppRunner</code> — contrairement à <code>web.run_app</code> — ne configure pas la journalisation).
Le proxy démarre mais la cascade est vide	Vérifiez <code>sdr_host</code> dans <code>admin_config.json</code> — normalement <code>127.0.0.1</code> . S'il contient encore une ancienne IP LAN d'une version précédente, corrigez-la et Redémarrer le panneau
Le bouton Kick indique « no connections »	Exécutez <code>getcap \$(which ss)</code> — sans <code>cap_net_admin</code> , lancez <code>sudo setcap cap_net_admin+ep \$(which ss)</code>
L'état affiche toujours OFFLINE	Allez dans Settings → SDR Process Name — indiquez le nom exact affiché par <code>ps -eo comm,args \ grep -v grep</code>
La page Users n'affiche aucune donnée	Vérifiez que le point de terminaison fonctionne : <code>curl http://127.0.0.1:<public_port>/users</code>
Un navigateur externe obtient NetworkError	Vérifiez que le proxy tourne : <code>./manage_admin.sh status</code>
Le proxy est cassé après un changement réseau	Concerne uniquement le cas où <code>sdr_host</code> contient encore une IP LAN — mettez-le à <code>127.0.0.1</code> et Redémarrer le panneau
Le bouton de suppression du chat n'a aucun effet	Vérifiez les droits d'écriture sur le fichier de journal du chat : <code>ls -l ~/PhantomSDR-Plus/chat.jsonl</code>
Le message sur la cascade n'apparaît pas	Vérifiez que le proxy tourne (<code>./manage_admin.sh status</code>) et que le frontend est sur une version récente incluant le rendu de la superposition
Message sur la cascade perdu après un redémarrage	Comportement attendu — l'état du message n'est qu'en mémoire ; renvoyez-le après le redémarrage du panneau
Le « Start » de Spot Reporting échoue / le démon se termine aussitôt	Consultez <code>autorun.log</code> . C'est généralement le lien symbolique <code>autorun/node_modules</code> manquant (<code>ln -sf ./frontend/node_modules autorun/node_modules</code>) ou <code>taskset</code> non installé (<code>sudo apt install -y util-linux</code>).
Spot Reporting : le démon tourne mais decode reste à 0 et <code>autorun.log</code> montre 504 / tap closed 1006	La prise audio n'atteint pas <code>spectrumserver</code> . Le démon détecte automatiquement le port depuis la configuration du serveur en cours d'exécution ; la ligne de journal <code>[autorun] tap backend: HOST:PORT</code> doit correspondre à votre <code>[server] port</code> . Si elle est fausse (ou si le serveur n'était pas lancé au démarrage), fixez-le dans <code>autorun.json</code> : <code>"server": { "host": "127.0.0.1", "port": 9002 }</code> , puis Stop→Start.
Spot Reporting affiche « 0 sent »	Normal pendant les premières minutes — PSK Reporter envoie toutes les 5 min, <code>wsprnet</code> toutes les 2 min. Vérifiez qu'une destination est activée et que le démon tourne.
	Redémarrer le panneau — la matrice est chargée au démarrage.

Problème	Correctif
Les nouvelles bandes/modes n'apparaissent pas dans la matrice	
autorun.log affiche MODULE_TYPELESS_PACKAGE_JSON / « Reparsing as ES module ... performance overhead »	Avertissement cosmétique (le décodage fonctionne toujours). Il manque "type": "module" à frontend/src/modules/package.json ; ajoutez cette ligne près du début, puis Stop→Start. Aucune recompilation du frontend n'est nécessaire — le démon importe directement le fichier source.
Le « Start » de Spot Reporting échoue avec taskset: ... Invalid argument sur un PC ancien/modeste	Ne devrait pas se produire avec le lanceur tenant compte de la configuration — la plage de cœurs est déduite du nombre de CPU et passe sans épinglage sur ≤ 4 cœurs. Si cela arrive, votre admin_server.py est antérieur à ce changement ; mettez-le à jour (_autorun_taskset_prefix) et Redémarrer le panneau.
Le badge REPORTING n'apparaît pas sur la cascade	Le report doit être activé avec au moins un créneau ; le badge lit / autorun-active.json. Recompilez le frontend si dist/index.html est antérieur au badge.

CHAPITRE 4

Thermal Guard

THERMAL_GUARD.md

Thermal Guard — Manuel du sysop

Protection contre la surchauffe du processeur pour PhantomSDR-Plus.

Un WebSDR est l'une des rares applications qui occupent tous les cœurs, toute la journée, indéfiniment. Si un ventilateur s'encrasse, si une pompe s'arrête ou si la pièce devient brûlante en août, personne ne vous prévient : le serveur continue simplement à décoder jusqu'à ce que le processeur cuise. Le thermal guard est la pièce qui s'en aperçoit et débranche à votre place.

Il est livré **inerte** : d'origine, il n'écrit que dans un journal. Rien n'est protégé tant que vous n'avez pas choisi un mode. Ce manuel porte sur ce choix et sur la façon de vivre avec.

Pressé ? Lancez `python3 thermal_guard.py --once`, lisez les seuils affichés, puis passez `thermal_mode` à `stop+restart` dans le panneau d'administration. C'est tout le travail. Le reste ci-dessous, c'est le *pourquoi*.

Sommaire

1. Ce qu'il fait
2. Démarrage rapide
3. Comment il décide
4. Les quatre modes — ce que chacun fait et ce que vous devez faire
5. Choisir un mode
6. Référence de configuration
7. Fonctionnement sans le panneau d'administration
8. Activer l'étage throttle sans root
9. Fonctionnement sans surveillance — quand personne ne regarde
10. Le tester avant de lui faire confiance
11. Lire le journal
12. Lockout et limitation du nombre d'arrêts
13. Dépannage
14. Le désactiver ou le supprimer

1. Ce qu'il fait

Toutes les 2 secondes, le garde lit la température du die de votre processeur et la compare à quatre seuils. Lorsque la température reste au-dessus de l'un d'eux assez longtemps, il gravit un échelon :

```
normal → warn → throttle → stop → (refroidir) → restart
          ligne dans  baisse de  exécution du  le serveur  exécution du
          le journal  la fréquence script d'arrêt reste arrêté script de
                                   démarrage
```

Deux choix de conception le rendent utilisable sur la machine de *n'importe quel* sysop, et il vaut la peine de les comprendre avant de compter sur lui :

Il ne demande jamais comment vous lancez votre serveur. systemd, une boucle watchdog, cron, tmux, un simple shell : le garde ne le sait pas et s'en moque. À la place, tant que la machine est en surchauffe, il maintient un **lockout** : il réémet l'arrêt à *chaque* cycle. Si votre watchdog relance le serveur, le garde l'arrête à nouveau en moins de deux secondes. Tout ce qui essaie de le maintenir en vie perd, jusqu'à ce que le processeur ait refroidi. C'est ce qui le rend universel, et c'est testé face à un vrai watchdog.

Il ne code jamais une température en dur. « Arrêter à 95 °C » est une mauvaise règle : sur un Intel avec Tjmax 100 c'est presque une charge normale, sur un Ryzen c'est *en dessous* de là où Tctl se place sous boost par conception, et sur un Raspberry Pi c'est inatteignable puisque le SoC se limite lui-même à 80. Le garde lit donc le point critique publié par votre propre noyau (`tempN_crit`) et calcule à rebours. Même code, chiffres sensés partout.

2. Démarrage rapide

1. Voyez à quoi ressemble votre machine pour le garde. Toujours en premier :

```
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` indique `NONE`, arrêtez-vous ici et lisez le [Dépannage](#) : le garde ne peut pas protéger une machine qu'il ne peut pas mesurer, et il vous le dit plutôt que de faire semblant.

2. Laissez-le quelques jours en mode `log`. Il tourne déjà dans le panneau d'administration. Laissez-le observer une semaine normale — après-midi d'été, une grosse compilation, un samedi de concours chargé — et voyez s'il se serait déclenché.

3. Armez-le. Panneau d'administration → **Settings** → **THERMAL GUARD** → **MODE** → `stop+restart` → **SAVE THERMAL SETTINGS**. Effectif en 2 secondes environ ; aucun redémarrage nécessaire.

3. Comment il décide

Le capteur

Le garde ne fait confiance qu'aux véritables capteurs de die du processeur :

Source	Matériel typique
<code>coretemp</code>	Intel
<code>k10temp</code> , <code>zenpower</code> , <code>zenpower3</code>	AMD
<code>cpu_thermal</code> , <code>soc_thermal</code>	Raspberry Pi, cartes ARM
Zone thermique <code>x86_pkg_temp</code>	solution de repli sur x86

`acpitz` et les zones thermiques sans étiquette sont **délibérément exclus**. Sur un ordinateur de bureau classique, `acpitz` affiche environ 28 °C alors que le package du processeur est à 59 °C : un garde qui s'y fierait ne se déclencherait tout simplement jamais. Annoncer « désactivé, aucun capteur fiable » est plus honnête que faire semblant de protéger.

`psutil` est utilisé s'il se trouve installé, mais n'est jamais requis. Le garde n'utilise que la bibliothèque standard.

Les seuils

Chacun est un écart sous le point critique que la machine publie elle-même :

Seuil	Écart sous crit	Signification
warn	−12 °C	Quelque chose ne va pas, le consigner
throttle	−8 °C	Tenter de refroidir en ralentissant le processeur
stop	−5 °C	Trop près de la limite — arrêter le serveur
resume	−25 °C	Vraiment refroidi, retour possible

Ce qui donne en pratique :

Votre processeur annonce	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 machine ne publie aucun point critique, le garde se rabat sur un jeu fixe prudent et le signale. Vous pouvez remplacer n'importe quel seuil par une valeur absolue (voir [Référence de configuration](#)), mais vous ne devriez pas en avoir besoin.

Temps de maintien — pourquoi un pic ne déclenche jamais rien

Rien ne se produit sur une seule lecture chaude. Un seuil doit être tenu **en continu** :

- **warn** et **throttle** : 30 s (`thermal_warn_sustain_s`)
- **stop** : 60 s (`thermal_sustain_s`)
- **resume** : 300 s sous le seuil de reprise avant un redémarrage (`thermal_resume_s`)

Un pic de compilation, une rafale de décodage, un ventilateur qui monte en régime : tout cela est bien trop bref pour compter. Une seule mesure sous le seuil remet le compteur à zéro. De plus, après le démarrage, le garde exige 3 mesures valides consécutives avant d'agir, et il rejette toute valeur hors de 20–125 °C comme une anomalie de capteur et non comme une température.

4. Les quatre modes

`thermal_mode` est l'unique interrupteur qui décide jusqu'où le garde a le droit de monter dans l'échelle. Chaque mode **inclut tout ce que font les précédents**.

Mode	Avertit	Ralentit	Arrête	Redémarre	Le serveur peut-il tomber ?
log	✓	—	—	—	Non — ne touche à rien
throttle	✓	✓	—	—	Non
stop	✓	✓	✓	—	Oui, reste arrêté jusqu'à votre intervention
stop+restart	✓	✓	✓	✓	Oui, revient tout seul

log — observer et rapporter (valeur d'usine)

Ce qu'il fait : tout ce que font les autres modes, sauf agir. Il parcourt toute l'échelle et écrit dans `crash.log` ce qu'il *aurait* fait. Il n'exécute jamais votre script d'arrêt, ne touche jamais à la fréquence du processeur, n'interrompt jamais un auditeur.

Ce que vous devez faire : rien — il tourne déjà. Au bout d'une semaine, ouvrez l'onglet **CRASH** du panneau, ou lancez :

```
grep THERMAL crash.log
```

- **Rien du tout ?** Votre refroidissement va bien et vous pouvez armer le garde en confiance.
- **STOP level reached ... NOT stopping ?** Le garde aurait éteint votre serveur. C'est un vrai problème thermique avec lequel vous fonctionniez : réparez le refroidissement et armez le garde.

À utiliser quand : vous venez d'installer le garde et ne connaissez pas encore le comportement de votre machine. C'est un mode de rodage, pas une destination — un garde en `log` ne protège rien.

throttle — ralentir le processeur, sans jamais interrompre le service

Ce qu'il fait : au seuil de throttle, il abaisse de 20 % la fréquence maximale du processeur, puis la rétablit dès que la température repasse sous la ligne d'avertissement. Votre WebSDR reste en ligne tout du long. Les auditeurs peuvent noter des performances légèrement moindres en forte charge ; la plupart ne verront rien.

Si la température continue malgré tout de grimper au-delà du seuil d'arrêt, ce mode **le consigne et ne fait rien** : il n'arrêtera pas le serveur.

Ce que vous devez faire :

1. Donner au garde le droit d'écriture sur la limite de fréquence du processeur — il en a besoin et, par défaut, seul root l'a. Lancez une fois `./setup-cpufreq-perms.sh` ; voir la [section 8](#).
2. Vérifier avec `python3 thermal_guard.py --once` qu'il affiche `cpufreq : writable, throttle stage available`. S'il indique *not writable*, ce mode ne fait rien de plus que `log`, et il le dira à chaque fois dans le journal.

À utiliser quand : vous traquez un problème de refroidissement limite et voulez défendre la machine sans jamais perdre d'auditeurs. **Attention, ce n'est pas une protection complète** : si le ralentissement ne suffit pas, rien d'autre ne se passe. Bon comme première étape, mauvais comme réponse définitive.

stop — éteindre le serveur et le laisser éteint

Ce qu'il fait : au seuil d'arrêt il exécute votre script d'arrêt, puis maintient le **lockout** : il réarrête le serveur toutes les 2 secondes tant que la machine est en surchauffe, pour que rien ne le relance dans votre dos. Lorsque le processeur est resté 5 minutes sous le seuil de reprise, le lockout se libère. **Le**

serveur ne revient pas de lui-même. Vous le relancez quand vous êtes convaincu que la cause est réglée.

Ce que vous devez faire :

1. Vous assurer que le garde sait arrêter votre serveur. Dans le panneau, c'est le script d'arrêt configuré (`./stop-websdr.sh` avec les lanceurs fournis). Si aucun n'est défini, le garde se rabat sur `SIGTERM` vers le nom du processus, puis `SIGKILL` après 10 secondes de grâce — cela fonctionne, mais un vrai script d'arrêt est plus propre.
2. Accepter que **votre WebSDR restera hors ligne jusqu'à ce que vous le remarquiez**. Prévoyez un moyen de l'apprendre : une supervision de disponibilité, l'onglet CRASH, ou simplement une vérification quotidienne.

À utiliser quand : la machine compte plus que le service, vous êtes généralement sur place, ou un incident thermique est assez sérieux pour que vous vouliez inspecter la machine avant qu'elle reparte. C'est aussi le bon choix si vous ne faites pas confiance à un redémarrage automatique sur votre matériel.

stop+restart — arrêter, refroidir, revenir tout seul

Ce qu'il fait : tout ce que fait `stop`, et en plus : une fois que le processeur est resté 5 minutes sous le seuil de reprise, il exécute votre script de démarrage et le WebSDR revient de lui-même.

Le battement est évité par `thermal_max_stops_hour` (2 par défaut). Après deux arrêts thermiques en une heure, le garde **désactive le redémarrage automatique** et laisse le serveur arrêté pour que vous regardiez. Notez le détail important : la limite coupe le *redémarrage*, jamais la *protection* — le *lockout* continue d'arrêter le serveur tant qu'il est chaud, quel que soit le nombre de déclenchements déjà survenus.

Ce que vous devez faire :

1. Vous assurer que **les deux** scripts sont configurés et fonctionnent réellement seuls — celui de démarrage comme celui d'arrêt. Testez-les à la main : `./stop-websdr.sh` puis `./start-rx888mk2.sh`.
2. Rien d'autre. C'est le mode qu'on installe et qu'on oublie.

À utiliser quand : le WebSDR est public, sans surveillance ou distant — c'est-à-dire la plupart du temps. C'est le mode recommandé pour une installation normale.

5. Choisir un mode

Votre situation	Mode
Garde tout juste installé, machine encore inconnue	log pendant une semaine
WebSDR public, sans surveillance, qui doit se débrouiller seul	stop+restart
Site distant difficile d'accès	stop+restart
Vous voulez inspecter la machine après chaque incident thermique	stop
Matériel auquel vous ne confiez pas un redémarrage automatique	stop
Refroidissement limite, interdiction de perdre des auditeurs	throttle, puis passer à stop+restart
Machine sans capteur processeur fiable	aucun ne fonctionnera — réglez d'abord le capteur

Pour la plupart des sysops la réponse est `stop+restart`, et le résumé honnête des autres est : `log` ne protège rien et `throttle` protège un peu.

6. Référence de configuration

Toutes les clés se trouvent dans `admin_config.json`, à côté de `thermal_guard.py`, et chacune est aussi modifiable dans **Settings → THERMAL GUARD**. Le garde **relit le fichier à chaque cycle**, les changements prennent donc effet en ~2 secondes — sans redémarrage.

Clé	Défaut	Signification
<code>thermal_enabled</code>	<code>true</code>	Interrupteur principal. <code>false</code> désactive complètement le garde.
<code>thermal_mode</code>	<code>"log"</code>	<code>log</code> <code>throttle</code> <code>stop</code> <code>stop+restart</code> — voir la section 4 .
<code>thermal_warn</code>	<code>null</code>	Seuil d'avertissement absolu en °C. <code>null</code> = déduit du point critique.
<code>thermal_throttle</code>	<code>null</code>	Seuil de throttle absolu. <code>null</code> = déduit.
<code>thermal_stop</code>	<code>null</code>	Seuil d'arrêt absolu. <code>null</code> = déduit.
<code>thermal_resume</code>	<code>null</code>	Seuil de reprise absolu. <code>null</code> = déduit.
<code>thermal_sustain_s</code>	<code>60</code>	Secondes de maintien du seuil d'arrêt avant d'arrêter. Minimum 5.
<code>thermal_warn_sustain_s</code>	<code>30</code>	Secondes pour les étages avertissement et throttle. Minimum 5.
<code>thermal_resume_s</code>	<code>300</code>	Secondes sous le seuil de reprise avant libération du lockout. Minimum 30.
<code>thermal_max_stops_hour</code>	<code>2</code>	Arrêts thermiques par heure au-delà desquels le redémarrage automatique est coupé. La protection continue.
<code>thermal_test_temp</code>	<code>null</code>	Fait croire que le processeur est à cette température. Pour les tests uniquement — voir la section 10 .

Le garde autonome utilise en plus `start_script`, `stop_script`, `sdr_process_name` et `sdr_base_dir` du même fichier.

Remplacer un seuil. Ne le faites qu'avec une raison. Une courante : votre machine chauffe légitimement pendant les compilations et vous préférez que cela ne compte pas. Relever `thermal_stop` vous donne de la marge au prix de la distance au point critique — ne le mettez jamais au-dessus de votre valeur crit, sinon le processeur atteindra d'abord sa propre coupure d'urgence et le garde deviendra décoratif.

7. Fonctionnement sans le panneau d'administration

Beaucoup de sysops n'installent jamais le panneau. Le garde fonctionne très bien seul — même fichier, même comportement.

Essayez-le d'abord au premier plan :

```
cd ~/PhantomSDR-Plus
python3 thermal_guard.py --once          # que se passerait-il ici ?
python3 thermal_guard.py --mode log      # observer en direct, Ctrl-C pour quitter
```

`--mode` remplace `thermal_mode` depuis la ligne de commande, vous n'avez donc jamais à écrire de JSON à la main :

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

Rendez-le ensuite permanent avec l'unité d'exemple fournie dans le dépôt :

```
sudo cp thermal-guard.service /etc/systemd/system/
sudo nano /etc/systemd/system/thermal-guard.service # réglez User= et les chemins
sudo systemctl daemon-reload
sudo systemctl enable --now thermal-guard
systemctl status thermal-guard
```

Lisez les commentaires en tête de `thermal-guard.service` avant de l'activer. Deux points comptent le plus :

- **User= doit être un compte capable d'exécuter réellement vos scripts de démarrage et d'arrêt.** Faire tourner le garde en root est généralement la mauvaise réponse : il exécuterait aussi ces scripts en root.
- Pointez-le soit vers un `admin_config.json` contenant `start_script` et `stop_script`, soit passez `--mode` sur la ligne `ExecStart`. Une configuration minimale suffit :

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

Tout ce que fait le garde va toujours dans `crash.log`, à côté de ce fichier de configuration, ainsi que dans le journal `systemd` (`journalctl -u thermal-guard`).

8. Activer l'étage throttle sans root

Par défaut, `python3 thermal_guard.py --once` indique :

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

C'est normal et ce n'est pas un bogue. Le noyau crée

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

appartenant à `root:root`, en mode `0644`. Un panneau d'administration exécuté par un utilisateur ordinaire ne peut pas y écrire, le garde saute donc l'étage throttle et le dit dans le journal plutôt que d'échouer en silence.

La mauvaise solution consiste à exécuter le garde en root. Il exécute aussi vos scripts de démarrage et d'arrêt, et ceux-ci doivent rester sans privilèges.

La bonne solution est de donner le droit d'écriture sur ces fichiers précis à un groupe dédié. Un script s'en charge :

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

Il demande votre mot de passe une fois — et une seule, à l'installation — puis :

1. crée un groupe système `cpufreq` et vous y ajoute ;
2. installe `/etc/tmpfiles.d/99-phantomsdr-cpufreq.conf` afin que le groupe et le mode `0664` soient réappliqués à **chaque démarrage** : les droits de sysfs ne survivent pas seuls à un redémarrage ;
3. applique la modification immédiatement, pour que vous puissiez tester sans redémarrer.

Ensuite, dans cet ordre :

```
# 1. déconnectez-vous et reconnectez-vous (ou redémarrez) – un nouveau groupe
# n'atteint vos processus qu'au travers d'une nouvelle session
id | grep cpufreq

# 2. redémarrez le panneau ; le garde teste le droit d'écriture une fois, au démarrage
sudo systemctl restart phantomsdr-admin
# (aucune déconnexion nécessaire ainsi – systemd reconstruit la liste des
# groupes à chaque démarrage. Seulement si les unités ne sont PAS
# installées : ./manage_admin.sh stop && ./manage_admin.sh start – jamais les deux, elles se
# disputent le port 3000)

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

Pour tout annuler : `sudo ./setup-cpufreq-perms.sh --revoke.`

Remarques

- L'étage throttle n'agit que si `thermal_mode` vaut `throttle` ou plus. Accorder la permission ne change rien en soi.
- Le garde abaisse de 20 % la limite en vigueur à cet instant et rétablit exactement cette valeur. Si vous bridez déjà votre processeur (par exemple via `MAX_SPEED` dans `/etc/init.d/cpufrequtils`), c'est votre bridage qui est rétabli : le garde ne vous rendra pas discrètement une fréquence plus élevée que celle demandée.
- Certaines machines n'exposent aucun pilote `cpufreq` — de nombreux VPS, la plupart des conteneurs. Le script le détecte et ne change rien. Utilisez-y `stop` ou `stop+restart` ; ils ne dépendent pas de `cpufreq`.

9. Fonctionnement sans surveillance

La question la plus fréquente au sujet du garde : *que se passe-t-il quand personne ne regarde ?* La réponse courte est que c'est précisément le cas pour lequel il a été construit.

Il n'a pas besoin d'un navigateur ouvert. Le garde est un fil d'exécution en arrière-plan dans `admin_server.py`, cadencé par l'échantillonneur des graphiques toutes les 2 secondes. Il tourne que quelqu'un soit connecté au panneau ou non, qu'un navigateur soit ouvert ou non, que vous soyez éveillé ou non. En mode autonome c'est un service `systemd`, encore plus indépendant. La carte du tableau de bord est une *fenêtre* sur le garde, pas le garde lui-même.

Il agit sans demander. Il n'y a ni boîte de confirmation ni notification à attendre. Au seuil d'arrêt, il arrête le serveur, point. C'est exactement l'objectif.

Mais seulement si vous lui en avez donné l'autorisation. En mode `log`, il consignera fidèlement une machine en train de cuire sans rien y faire. Si vous lisez cette section parce que « *je ne suis pas là pour surveiller* », le mode qu'il vous faut est `stop+restart` : c'est le seul qui protège la machine et remet le service en route sans vous.

Il ne protège que tant que le panneau est vivant. Le garde fait partie d'`admin_server.py` : un panneau mort à 02:00 emporte la protection avec lui, et après un redémarrage la machine reste sans surveillance jusqu'à ce que vous vous connectiez pour le relancer. Si personne ne surveille, laissez `systemd` surveiller : le dépôt fournit `phantomsdr-admin.service` et `phantomsdr-proxy.service`, qui démarrent au boot et relancent le panneau dans les cinq secondes suivant un plantage. Voir [ADMIN_PANEL_SETUP.md](#). Pour un récepteur non surveillé, c'est **fortement recommandé** : armer `stop+restart` sans cela ne vous protège que jusqu'au prochain redémarrage.

Comment l'apprendre après coup. Par ordre d'utilité approximatif :

Où	Ce que vous obtenez
<code>crash.log</code> / onglet CRASH	Chaque action, horodatée. La trace permanente.
Carte THERMAL GUARD du tableau de bord	État en direct : température, étage, lockout, arrêts de la dernière heure.
Page Graphs (température CPU, 24 h)	La forme de l'incident — vitesse de montée, durée du refroidissement.
<code>journalctl -u thermal-guard</code>	Mode autonome uniquement.

Une routine raisonnable pour un site sans surveillance : `stop+restart` armé, et `grep THERMAL crash.log` chaque fois que vous vous connectez. Si c'est vide, il n'y a rien à savoir.

Un exemple concret. Panne de ventilateur à 03 h 00 sur une machine en `stop+restart` :

```
03:14 la température passe 88 °C, tient 30 s      → avertissement consigné
03:16 passe 92 °C, tient 30 s                    → throttle appliqué (si autorisé)
03:19 passe 95 °C, tient 60 s                    → STOP – script d'arrêt, lockout actif
03:19 le watchdog relance le serveur             → le garde le rearrière et le consigne
03:26 le CPU est sous 75 °C depuis 5 minutes     → lockout libéré, script de démarrage
03:41 surchauffe et s'arrête une deuxième fois   → limite atteinte, plus de redémarrage
                                                le serveur reste arrêté, la protection reste
```

Vous vous réveillez avec un WebSDR arrêté, un `crash.log` qui dit exactement pourquoi et — surtout — un processeur qui ne s'est jamais approché de son point critique.

10. Le tester avant de lui faire confiance

N'attendez pas une vraie canicule pour découvrir si votre script d'arrêt fonctionne.

`thermal_test_temp` fait croire au garde que le processeur est à la température de votre choix. Tout le reste se comporte parfaitement normalement — les temps de maintien, l'échelle, le lockout, vos vrais scripts.

1. Panneau → **Settings** → **THERMAL GUARD** → **TEST TEMPERATURE**.
2. Saisissez une valeur au-dessus de votre seuil d'arrêt (par exemple 97 sur un Intel).
3. Enregistrez, puis observez la carte du tableau de bord et l'onglet CRASH.
4. **Videz le champ une fois terminé.** Une température de test oubliée signifie que le garde ne lit plus le vrai capteur.

En mode `log`, cela vous montre toute l'échelle sans rien risquer : le premier test sûr, et celui à faire avant d'armer quoi que ce soit.

En mode `stop+restart`, c'est un **test en conditions réelles** : votre serveur va réellement s'arrêter et réellement revenir environ cinq minutes après que vous ayez vidé le champ. Faites-le quand personne

n'écoute. Cela vaut la peine une fois, car c'est le seul moyen de savoir que vos scripts d'arrêt et de démarrage fonctionnent lorsqu'ils sont appelés par autre chose que vous.

Équivalents en mode autonome :

```
python3 thermal_guard.py --once          # seuils seulement, aucune action
python3 thermal_guard.py --mode log      # observer l'échelle en direct
```

11. Lire le journal

Tout est ajouté à `crash.log`, à côté du fichier de configuration, un évènement par ligne, chacun préfixé par `[THERMAL]` :

```
grep THERMAL crash.log
```

Les lignes sont volontairement sur une seule ligne et exploitables par `grep`. Ce que vous verrez :

Ligne	Signification
warn: 88.4C (warn=88.0 crit=100) sustained 30s	Premier échelon. Rien n'a été fait.
throttle: 92.1C sustained 30s – cpufreq max lowered 20%	Fréquence processeur réduite.
throttle level reached: ... (no cpufreq write access – stage skipped)	Voir la section 8 .
STOP level reached: ... mode is 'log', NOT stopping	Il aurait arrêté le serveur. Armez-le.
STOPPED server: 95.2C sustained 60s (stop=95.0 crit=100) – ran stop-websdr.sh	Le cas réel.
lockout: process reappeared at 96.0C – re-stopped (...) [12 time(s) so far]	Quelque chose relance votre serveur ; le garde gagne. Limité à une ligne par minute.
rate limit: 2 thermal stops within the hour – automatic restart is now DISABLED	Reste arrêté pour que vous enquêtiez. La protection continue.
recovered: 54.0C held below 75.0 – lockout cleared	De nouveau froid.
auto-restart: ran start-rx88mk2.sh	De retour en ligne.
back to normal: 59.0C (warn=88.0)	Redescendu de l'échelle sans jamais s'arrêter.

12. Lockout et limitation du nombre d'arrêts

Ce sont les deux comportements qui surprennent le plus, autant les énoncer clairement.

Le lockout est ce qui fait fonctionner le garde face à *votre* installation en particulier. Quand il arrête le serveur, il ne se contente pas d'émettre un arrêt en espérant. Il se marque verrouillé et, à chaque cycle de 2 secondes tant que la température dépasse le seuil de reprise, il vérifie si le processus est revenu — et si c'est le cas, il l'arrête de nouveau. Votre watchdog, le `Restart=always` de `systemd`, une tâche cron, un sysop impatient : tous perdent cette discussion jusqu'au refroidissement du processeur. Il se libère automatiquement dès que la température est restée sous le seuil de reprise pendant `thermal_resume_s` (5 minutes par défaut).

Le bouton CLEAR LOCKOUT de la carte du tableau de bord libère le lockout et le compteur d'arrêts par heure, pour les cas où vous avez réparé le refroidissement et ne voulez pas attendre. Il **ne désarme pas** le garde : si la machine est encore chaude, la vérification suivante arrêtera de nouveau le serveur. C'est voulu.

La limitation (`thermal_max_stops_hour`, 2 par défaut) empêche une machine défaillante de faire des allers-retours toute la nuit. Une fois la limite atteinte, le garde **désactive le redémarrage automatique** et laisse le serveur arrêté. Lisez bien : il désactive le redémarrage, pas la protection. Le lockout continue de réarrêter le serveur tant qu'il est en surchauffe, quel que soit le nombre de déclenchements déjà survenus. Un garde qui se tairait précisément au pire moment pour la machine serait pire que pas de garde du tout.

13. Dépannage

sensor : NONE – no trusted CPU sensor on this machine Le garde n'a trouvé aucun capteur de la liste autorisée et refuse de deviner. Essayez `sensors` (paquet `lm-sensors` ; `sudo apt install lm-sensors && sudo sensors-detect`). Dans une machine virtuelle ou un conteneur, il n'y a souvent réellement aucun capteur de die exposé : le garde ne peut pas protéger cette machine et se déclare désactivé plutôt que de faire semblant.

cpufreq : not writable – throttle stage disabled Attendu tant que vous n'avez pas lancé `./setup-cpufreq-perms.sh` — voir la [section 8](#). Cela n'affecte que l'étage throttle ; `stop` et `stop+restart` ne sont pas concernés.

J'ai lancé `setup-cpufreq-perms.sh`, il dit toujours not writable Deux causes probables : vous ne vous êtes pas déconnecté puis reconnecté depuis (un nouveau groupe n'atteint les processus qu'au travers d'une nouvelle session — vérifiez avec `id`), ou le panneau n'a pas été redémarré depuis (le garde teste le droit d'écriture une fois, au démarrage). Faites les deux, dans cet ordre.

Les seuils semblent faux pour mon processeur Vérifiez ce que publie la machine : `cat /sys/class/hwmon/hwmon*/temp*_crit`. Si votre crit est inhabituel, ou absent, mettez des valeurs absolues dans `thermal_stop` et compagne.

Il a arrêté mon serveur alors qu'il ne me semblait pas chaud Vérifiez si `thermal_test_temp` est resté défini depuis un test. C'est de loin la cause la plus fréquente.

Il ne se déclenche jamais alors que la machine chauffe Vérifiez que le mode n'est pas `log`, que `thermal_enabled` vaut `true`, et comparez la température observée aux seuils donnés par `--once`. Rappelez-vous que le seuil doit être dépassé *en continu* pendant tout le temps de maintien.

Le serveur n'arrête pas de redémarrer pendant un incident thermique C'est le lockout qui fonctionne, et les lignes `lockout:` sont son rapport de réussite : votre superviseur continue d'essayer, le garde continue de défaire. Rien à corriger.

Le panneau ne tourne pas, le garde tourne-t-il ? Non — en mode panneau, le garde vit dans `admin_server.py`. Si vous voulez une protection indépendante du panneau, utilisez le service autonome de la [section 7](#).

14. Le désactiver ou le supprimer

- **Le mettre en pause** : passez `thermal_mode` à `log`. Il continue d'observer et de rapporter, sans jamais agir.
- **Le désactiver complètement** : passez `thermal_enabled` à `false`.
- **Autonome** : `sudo systemctl disable --now thermal-guard`.

- **Annuler les droits cpufreq :** `sudo ./setup-cpufreq-perms.sh --revoke`.

Supprimer `thermal_guard.py` est également sans danger — `admin_server.py` l'importe dans un `try` et signale simplement le garde comme indisponible s'il manque.

Voir aussi

- [Installation du panneau d'administration](#) — l'onglet Thermal Guard dans son contexte
- [Installation](#) — protection thermique lors d'une installation neuve
- [Structure du projet](#) — où se trouve `thermal_guard.py`

CHAPITRE 5

Limites de connexion

CONNECTION_LIMITS.md

Limites de connexion — Manuel du sysop

Protéger un récepteur public contre les afflux de connexions.

Un WebSDR est un service public qui remet à quiconque le demande un flux continu d'audio et de spectre. C'est tout son intérêt, et c'est aussi le problème : pour le serveur, rien ne distingue un auditeur normal de quelqu'un qui ouvre des sessions en boucle. Le 16 septembre 2026, ce récepteur a subi exactement cela — une seule adresse a ouvert 134 sessions d'écoute en une minute et en a maintenu 181 simultanément, alors que le public réel de la station tournait autour d'une douzaine.

Ce manuel traite des trois couches qui l'arrêtent, de ce que chacune peut et ne peut pas faire, et de la manière de savoir si les vôtres sont raisonnablement réglées.

Pressé ? Le `config.toml` d'exemple est déjà livré avec des valeurs sensées : une installation neuve est donc protégée sans que vous fassiez quoi que ce soit. Si votre propre `.toml` est plus ancien et ne comporte pas de clés `[limits]` au-delà de `audio/waterfall/events`, les limites par adresse sont tout simplement **désactivées** — copiez le bloc depuis l'exemple pour les activer. La seule commande à connaître ensuite est `grep Refused spectrumserver.log`, qui vous dit si quelqu'un se fait refuser.

Sommaire

1. Contre quoi vous vous défendez réellement
2. Les trois couches
3. Limites par adresse — la politique des auditeurs
4. Connexions inactives — l'attaque silencieuse
5. La protection au niveau du noyau — `setup-firewall.sh`
6. Référence de configuration
7. Ce que voit un auditeur refusé
8. Lire les journaux et les compteurs
9. Choisir vos nombres
10. Pièges à connaître
11. Tout désactiver

1. Contre quoi vous vous défendez réellement

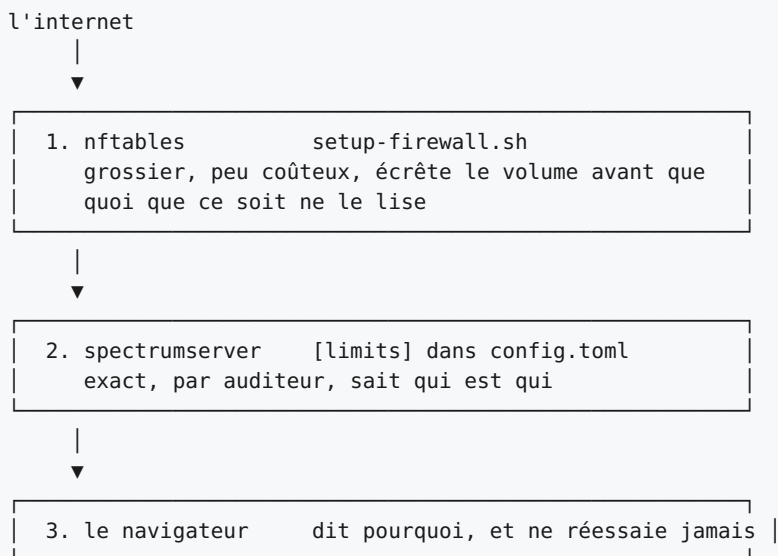
Il vaut la peine d'être précis, car le mot « DDoS » recouvre deux choses très différentes et une seule d'entre elles peut se régler sur votre propre machine.

Afflux d'une source unique. Une machine, ou une poignée, ouvrant des connexions aussi vite qu'elle le peut. C'est ce qui s'est produit ici, et cela se corrige entièrement : les connexions arrivent d'une adresse que l'on peut compter, et compter suffit.

Une véritable attaque distribuée. Des milliers de machines, souvent avec des adresses source falsifiées, qui saturent votre lien. Quand ces paquets atteignent votre carte réseau, la bande passante est déjà dépensée. **Rien de ce que décrit ce manuel n'aide**, et rien de ce que vous installez sur le serveur ne le peut — la seule réponse est un service en amont de vous, ce qui est une autre discussion, impliquant un vrai domaine et un fournisseur comme Cloudflare devant le récepteur.

Tout ce qui suit traite du premier cas. Ce n'est pas une limite dont il faille s'excuser : le premier cas est ce qui arrive réellement aux stations d'amateur.

2. Les trois couches



Elles ne sont délibérément pas redondantes. La couche noyau est rapide mais bête — elle compte les paquets venant d'une adresse et ne sait rien des auditeurs. La couche serveur est précise mais ne voit une connexion qu'une fois acceptée et analysée, ce qui coûte quelque chose. La couche navigateur existe pour qu'un auditeur refusé sache ce qui s'est passé, au lieu de fixer une page qui ne charge jamais.

3. Limites par adresse

Ce qui compte pour un auditeur

Un auditeur, c'est **une connexion /audio** — ou, là où `[kiwi_emulation]` est activé, **une connexion son Kiwi**, qui compte exactement pareil. C'est la même unité que celle qu'utilisent déjà le compteur d'utilisateurs, `users.json` et les étiquettes sur la cascade : un client Kiwi occupe donc une place et apparaît dans la liste des utilisateurs comme un navigateur.

Cela compte, car un seul onglet de navigateur sur ordinateur ouvre **quatre** connexions vers votre serveur :

Connexion	Rôle
/audio	le flux audio démodulé — c'est lui, l'auditeur
/waterfall	l'affichage du spectre
/events	liste des utilisateurs, fréquences des autres auditeurs
/chat	le panneau de discussion

Un téléphone sur /mobile en ouvre deux (/audio et /chat), car il n'a pas de cascade. Une limite qui compterait les connexions brutes refuserait donc tout le monde : trois onglets, cela fait douze connexions.

Les trois clés

```
[limits]
per_ip=3          # auditeurs simultanés depuis une adresse
per_ip_sockets=0  # connexions simultanées de tout type ; 0 donne per_ip * 4 + 4
per_ip_rate=40    # nouvelles connexions par minute depuis une adresse
per_ip_ban_s=600  # durée de refus d'une adresse qui dépasse le débit
```

per_ip est le nombre principal, celui auquel vous penserez. Trois est un bon point de départ.

per_ip_sockets existe parce que /audio n'est pas la seule chose qu'il vaille la peine d'inonder. /waterfall est le flux le plus lourd que produit votre serveur, et une limite qui ne compterait que les auditeurs se contournerait en ouvrant simplement des connexions de cascade. Laissé à 0, il devient $\text{per_ip} * 4 + 4$, ce qui loge per_ip onglets complets avec un peu de marge pour les connexions que le noyau n'a pas encore recyclées après un rechargement de page.

per_ip_rate est ce qui met réellement fin à un afflux. Le plafond seul n'y suffit pas : refuser une connexion est peu coûteux mais pas gratuit, et un attaquant peut se reconnecter aussi vite que les refus reviennent, en brûlant votre processeur en montages et démontages. La limite de débit lui donne un blocage au lieu d'une réponse. Rappelez-vous qu'un onglet coûte quatre : 40/minute, c'est environ dix chargements de page par minute depuis une adresse.

Un ordre délibéré. Le débit est débité *avant* que le plafond de simultanéité ne soit testé. Cela semble à l'envers jusqu'à ce qu'on pense à une adresse déjà à son plafond et qui se reconnecte en boucle : si le refus bon marché « tu es au plafond » lui répondait à chaque fois, les infractions de débit ne s'accumuleraient jamais et le blocage qui met fin à la boucle ne se déclencherait jamais.

Qui est exempté

La boucle locale n'est jamais limitée. Le tap autorun pour les rapports de spots, le panneau d'administration et un navigateur ouvert sur la machine du serveur elle-même arrivent tous depuis 127.0.0.1, et les laisser consommer des places reviendrait à ce que votre propre station enferme dehors vos auditeurs.

Cela couvre aussi proxy.py : les connexions passant par le proxy atteignent spectrumserver depuis la boucle locale, et l'adresse réelle du client arrive dans l'en-tête X-Forwarded-For, sur lequel les limites sont indexées.

Depuis la v5.0.0, cet en-tête n'est cru **que s'il vient d'un pair loopback** — un proxy sur la même machine. Envoyé par n'importe qui d'autre, il est ignoré et c'est l'adresse TCP qui compte, car sinon un visiteur direct pourrait écrire X-Forwarded-For: 127.0.0.1 et passer pour loopback, au-delà de toutes les limites de cette page. proxy.py supprime en outre tout X-Forwarded-For ou X-Real-IP envoyé par un client avant d'ajouter les siens, et ne répond à l'expulsion du sysop (/~kick) que pour les clients de la

même machine. Une station à plusieurs récepteurs derrière le proxy est décrite dans [Plusieurs récepteurs](#) ; les limites s'y appliquent par récepteur.

4. Connexions inactives

Il existe une attaque qu'aucun des mécanismes ci-dessus ne peut voir.

Ouvrir une connexion TCP. Ne rien envoyer. La garder.

Un tel interlocuteur ne complète jamais de requête : il n'atteint donc jamais le code où vivent les limites par adresse — pas d'auditeur, pas d'objet client, rien qui porte un nom que l'on puisse compter. Pendant ce temps, cela lui coûte quelques octets et vous coûte un descripteur de fichier et une place tant qu'elle est tenue.

```
[limits]
idle_per_ip=8      # connexions inactives (avant requête) par adresse
idle_total=512     # plafond toutes adresses confondues
```

La limite porte sur le **combien**, pas sur le **combien de temps**, et cette distinction fait toute la conception. **Il n'y a aucun délai sur une connexion inactive, et c'est délibéré.** Un délai semble la réponse évidente et c'est un piège (voir §10) : il existe des interlocuteurs légitimes qui se connectent puis restent silencieux longtemps avant de parler. Ce qui les sépare d'une attaque, c'est la quantité — un interlocuteur légitime tient une telle connexion, un attaquant en tient des milliers.

Une connexion inactive vit donc jusqu'à ce qu'elle fasse une requête ou qu'elle meure, et sa place est rendue dans les deux cas.

`idle_total` borne un afflux réparti sur de nombreuses adresses source. Une fois le plafond atteint, ce sont les **nouvelles** connexions inactives qui sont refusées plutôt que les existantes évincées : une connexion légitime et de longue durée n'est donc jamais ce que l'on sacrifie pour faire de la place.

Les deux s'appliquent au moment où la socket est acceptée, à partir de l'adresse du pair — il n'y a pas encore de requête, donc pas de X-Forwarded-For à consulter. Ce n'est pas gênant : la boucle locale n'est pas comptée, ce qui couvre tout ce qui arrive par `proxy.py`, et un afflux direct porte sa propre adresse.

5. La protection au niveau du noyau

`setup-firewall.sh` installe une table `nftables` qui écrête le volume avant que `spectrumserver` n'en lise le moindre octet. Elle est délibérément grossière, et ses nombres se situent bien au-dessus de ceux de `config.toml`, afin qu'un auditeur normal ne puisse jamais être pris par les deux.

```
./setup-firewall.sh --show      # afficher les règles, ne rien changer
sudo ./setup-firewall.sh --check # valider vis-à-vis de votre noyau
sudo ./setup-firewall.sh --apply # charger, avec retour arrière automatique à 60 s
sudo ./setup-firewall.sh --persist # recharger au démarrage
sudo ./setup-firewall.sh --status # les compteurs de paquets par règle
sudo ./setup-firewall.sh --remove # tout annuler
```

Elle couvre quatre choses : un plafond de connexions simultanées par adresse source sur vos ports de proxy et de récepteur, un débit de nouvelles connexions par source, un frein contre la force brute SSH, et le partage de fichiers Windows fermé à tout ce qui se trouve hors des plages d'adresses privées — `smbd` écoute sur `0.0.0.0` sur bien des machines, et son accessibilité depuis l'extérieur dépend d'un routeur que le script ne peut pas voir.

Les ports sont lus dans `admin_config.json`, le fichier même qu'utilise `proxy.py` : la configuration suit donc la vôtre.

Pourquoi elle ne peut pas vous enfermer dehors

Deux propriétés, toutes deux délibérées :

- La politique de la table est **accept**, et elle ne rejette que des motifs nommés précis. Une règle qui ne correspond pas laisse passer le paquet vers ce que vous avez configuré par ailleurs. Elle ne peut pas rendre la machine injoignable.
- **Les connexions établies sont acceptées dès la toute première règle.** La session SSH dans laquelle vous tapez n'est jamais affectée par ce qui suit.

Par-dessus cela, `--apply` arme un retour arrière automatique : charger les règles, puis confirmer dans les 60 secondes, faute de quoi elles sont retirées. Ne dites rien, fermez le terminal, perdez votre connexion — le jeu de règles disparaît de lui-même.

Utilisez bien cette fenêtre. Vérifiez **depuis un autre appareil, hors de votre propre réseau**, que le récepteur charge toujours et qu'une *nouvelle* session SSH s'ouvre. Tester avec la session que vous avez déjà ne prouve rien, puisqu'elle a été acceptée par la règle un.

6. Référence de configuration

Chaque clé ci-dessous vit sous `[limits]` dans votre fichier `.toml`. **Toutes sont par défaut désactivées ou généreuses** : une configuration qui ne les mentionne pas se comporte donc exactement comme avant.

Clé	Défaut	Signification
<code>per_ip</code>	0 (désactivé)	Auditeurs simultanés par adresse
<code>per_ip_sockets</code>	0 → <code>per_ip * 4 + 4</code>	Connexions simultanées de tout type par adresse
<code>per_ip_rate</code>	0 (désactivé)	Nouvelles connexions par minute et par adresse
<code>per_ip_ban_s</code>	600	Secondes de refus après dépassement du débit
<code>idle_per_ip</code>	8	Connexions inactives, avant requête, par adresse
<code>idle_total</code>	512	Connexions inactives toutes adresses confondues

Mettre n'importe quelle clé à 0 désactive ce contrôle précis.

Les trois qui ne sont pas des limites

`[limits]` contient aussi `audio`, `waterfall` et `events`, héritées du PhantomSDR d'origine, et **aucune des trois n'est appliquée**. `waterfall` et `events` ne sont lues par aucun code. `audio` est lue en un seul endroit : elle est envoyée comme `max_users` dans le JSON d'enregistrement aux annuaires de `[websdr] register_urls`, c'est donc le chiffre qu'affichent `sdr-list.xyz` et les autres, et elle ne fait absolument rien lorsque `register_online=false`. Si vous voulez un plafond d'auditeurs, c'est `per_ip` plus haut qui agit.

Une de plus, sous `[server]`

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

Ce n'est pas une limite mais un refus, et comme il partage le même code de fermeture, sa place est ici. Les pages annoncent la version depuis laquelle elles ont été chargées sous la forme `/audio?v=N`, et tout ce qui est en dessous de `min_client_version` est éconduit avec « *cette page est périmée — veuillez la recharger* ».

Cela existe parce que le serveur ne peut pas atteindre du JavaScript qui tourne déjà dans le navigateur de quelqu'un. Un onglet conserve le code qu'il a chargé jusqu'à ce que quelqu'un le recharge : une modification de la page n'atteint donc que les auditeurs qui rechargent par hasard — et une station qui vient de changer le comportement des sessions peut avoir besoin que les anciennes pages disparaissent maintenant, et non un jour.

Seul `/audio` est vérifié, car c'est là que vit une session, et la boucle locale ainsi que les chemins Kiwi sont exemptés afin que le tap autorun et les clients Kiwi ne soient pas touchés. `0` le désactive et c'est la valeur par défaut. Le laisser actif une fois les anciennes pages écoulées ne pose pas de problème : un client de diversité actuel refusé sur un `/audio` nu réessaie avec le marqueur, si bien que seuls restent éconduits les clients de diversité d'une version plus ancienne de PhantomSDR-Plus et les outils tiers qui ouvrent `/audio` sans `?v=`. Remettez-le à `0` si une autre station signale qu'elle ne peut plus vous utiliser comme partenaire de diversité.

7. Ce que voit un auditeur refusé

Rien ne se reconnecte jamais

Commencez par là, car cela gouverne tout le reste : **une connexion `/audio` qui tombe met fin à la session**. Il n'y a pas de reconnexion automatique, et c'est voulu.

Se reconnecter est le bon réflexe pour une application de discussion et le mauvais pour un récepteur. `/waterfall` et `/events` ne revenaient jamais avec elle : une session réessayée était donc une connexion audio vivante boulonnée à une cascade figée ; et arrêter le serveur ne le vidait plus, puisque chaque auditeur était de retour quelques secondes plus tard, remplissant la liste d'utilisateurs de sessions où en réalité personne ne se trouvait. La page se tait et l'auditeur la recharge — ce que faisait précisément ce récepteur avant qu'une reconnexion n'y soit ajoutée.

Pour les limites de ce manuel, cela supprime aussi un piège : face à une limite de débit, une tentative automatique prolongerait justement le blocage qu'elle cherchait à contourner.

Les deux codes de fermeture

Code	Signification	Qui l'envoie
4003	refusé — au-delà d'une limite par adresse, ou page périmée	<code>spectrumserver</code>
4001	exclu par le sysop	<code>spectrumserver</code> , quand le panneau appelle <code>/~kick</code>

Tous deux sont dans la plage privée 4000–4999, aucun ne peut donc être confondu avec un état de protocole, et tous deux sont tenus pour définitifs par toutes les pages. Le texte du motif voyage dans la trame de fermeture plutôt qu'en message, car la première trame sur `/audio` est réservée aux réglages du récepteur et tout ce qui serait envoyé avant casserait toute connexion normale.

Sur la page

Refusé au chargement de la page — l'auditeur ne démarre même pas, les deux pages s'expliquent donc. Le bureau affiche un panneau : « *Ce récepteur a refusé la connexion* », le motif, et la note que

tous ceux qui partagent un accès internet comptent pour une seule adresse. `/mobile` affiche la même chose dans sa zone d'avis.

Refusé ou exclu en cours de session — la page de bureau s'arrête simplement, délibérément sans message : la boucle de dessin de la cascade s'interrompt, et c'est tout. `/mobile` affiche une ligne invitant l'auditeur à recharger, sauf après une exclusion, où elle reste silencieuse elle aussi.

Ce silence après une exclusion est intentionnel. **Exclure, c'est le sysop qui met fin à une session, ce n'est pas une punition** — rien n'est annoncé, et qui veut revenir recharge la page et redevient un auditeur ordinaire une seconde plus tard. Tout ce qu'apporte le code 4001, c'est que le navigateur sache qu'il ne s'agissait pas d'une connexion tombée : l'exclusion prend donc effet au lieu de s'annuler d'elle-même.

L'exclusion est faite par `spectrumserver` lui-même, et non par `proxy.py` : les auditeurs se connectent directement au port du récepteur, le proxy ne possède donc aucune de leurs connexions et son exclusion n'atteint personne. Le panneau appelle `/~~kick?ip=&secs=`, joignable en boucle locale seulement et vérifié sur le vrai pair TCP, non sur X-Forwarded-For, que le client fournit. `secs` vaut zéro par défaut — aucun bannissement — mais accepte une durée si vous voulez garder l'adresse dehors un moment, et cela réutilise le mécanisme de bannissement de `per_ip_rate`, si bien que la reconnexion est refusée à la porte.

8. Lire les journaux et les compteurs

Le serveur

```
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
```

S'il n'y a rien, personne n'est refusé. La journalisation est limitée à au plus une ligne par adresse toutes les cinq secondes — sans cela, une attaque se transformerait en écritures disque sans borne, ce qui n'est qu'une façon plus lente de faire tomber la station.

Le noyau

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

Lisez le **counter packets N** de chaque règle ; c'est le nombre réellement rejeté. Tout à zéro signifie que rien n'a été bloqué et que vos seuils sont confortablement larges.

Ne vous alarmez pas des listes `elements = { ... }` dans les ensembles au-dessus des règles. Ce sont vos auditeurs ordinaires *suivis* par rapport aux limites — une entrée apparaît à la première connexion d'une adresse et expire une ou deux minutes plus tard. Suivre n'est pas bloquer.

9. Choisir vos nombres

Deviner est inutile ; vos propres journaux vous le diront. Voici la mesure qui a fixé ici `per_ip=3`, sur une semaine de `logs/users_*.jsonl` :

Sessions simultanées depuis une adresse	Nombre d'adresses
1	682
2	54
3	13
plus de 3	8

758 adresses distinctes en sept jours, et une limite de trois en aurait touché huit. L'afflux, lui, était à 181.

Pour refaire la mesure sur votre propre station, appariez les identifiants de session par adresse dans `logs/users_*.jsonl` et prenez le recouvrement maximal. Notez que le journal enregistre des événements `tune` et `disconnect` mais n'a pas d'événement `connect` : le début d'une session doit donc être déduit de son premier `tune`.

N'oubliez pas le NAT. Un club, une école, un bureau ou un opérateur mobile apparaissent comme une adresse unique, et tous ceux qui sont derrière partagent un même quota. C'est inhérent à toute limite indexée sur une adresse ; la seule atténuation est de choisir un nombre qui vous convienne. Si un auditeur signale un jour ne pas pouvoir se connecter, c'est la première chose à soupçonner — regardez `grep Refused` avant toute autre chose.

10. Pièges à connaître

Jamais de délai sur une connexion inactive

Il n'y a plus de réglage `handshake_s`, et voici pourquoi. Un délai de 30 secondes sur les connexions inactives a l'air de la défense évidente contre un slowloris. Il a été écrit, activé, et a produit ceci en une minute :

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

192.87.173.88, c'est `etgd-websdr.ewi.utwente.nl` — **la machine de rappel de websdr.org elle-même**. Elle se reconnecte vers votre serveur puis reste inactive, bien plus de 30 secondes, avant d'envoyer son `GET /~~orgstatus`. Le délai la coupait à chaque fois et l'enregistrement s'est arrêté. Le réglage a été supprimé plutôt que laissé en option : une arme qui se retourne contre vous et casse votre fiche d'annuaire ne vaut pas les soixante lignes qui l'implémentent.

Le même raisonnement vaut pour `open_handshake_timeout` de `websocketpp`, qui est toujours là.

La valeur de dix minutes dans `spectrumserver.cpp` paraît absurde et elle est porteuse : c'est elle qui permet au rappel de websdr.org de rester inactif. Il y a une seconde raison — le gestionnaire de `/~~orgstatus` répond depuis un fil qui détient un `dup()` de la socket, et `websocketpp` termine une connexion expirée par `shutdown()`, un appel au niveau de la socket qui traverse le duplicata et casserait le rappel même si le temps d'inactivité n'était pas en cause.

S'il faut borner les connexions silencieuses, bornez leur nombre, pas leur âge. C'est exactement ce que sont `idle_per_ip` et `idle_total`.

Un fichier `nftables` doit être idempotent

`nft -f` **ajoute** à une table déjà existante au lieu de la remplacer. Un fichier de règles sans préambule `delete table` double donc toute la chaîne à chaque rechargement — ce que fait précisément une unité `systemd` naïve à chaque redémarrage du service. `setup-firewall.sh` produit le préambule pour vous ; si vous écrivez vos propres règles, faites de même.

```
sudo ./setup-firewall.sh --status | grep -c dport # doit donner 8, pas 16
```

11. Tout désactiver

Les limites du serveur : mettez les clés à 0, ou effacez-les de votre `.toml`, puis redémarrez le récepteur. Il n'y a pas d'autre état ; rien ne survit à un redémarrage.

Le pare-feu :

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

Cela supprime la table, retire le jeu de règles enregistré et désactive le service de démarrage. Votre machine revient exactement dans l'état où elle était avant — ce qui, sur la plupart des installations, signifie aucun pare-feu du tout. Cela mérite réflexion avant de le retirer.

CHAPITRE 6

Plusieurs récepteurs

MULTI_RECEIVER.md

Plusieurs récepteurs — Manuel du sysop

Deux récepteurs ou plus sur un même ordinateur, avec un sélecteur de récepteur sur la page.

Depuis la v5.0.0, un ordinateur PhantomSDR-Plus peut faire tourner plusieurs récepteurs en même temps — par exemple un RX-888 pour les ondes courtes et un RTL-SDR pour le 2 m — et les auditeurs passent de l'un à l'autre par des boutons dans l'en-tête de la page, comme OpenWebRX propose ses profils. Chaque récepteur garde sa propre cascade, son chat, ses repères, sa liste d'auditeurs et ses informations de station ; personne sur un récepteur n'est jamais gêné par quelqu'un sur un autre.

Pressé ? Une station à un seul récepteur n'a besoin de rien de tout cela et ne voit aucun changement. Pour un second récepteur : créez `instances/<nom>/` avec son propre `config.toml`, démarrez-le avec `INSTANCE=<nom> ./start-<radio>.sh`, inscrivez les deux récepteurs dans `receivers.toml` et redémarrez le proxy. L'exemple de la [section 3](#) le détaille ligne par ligne.

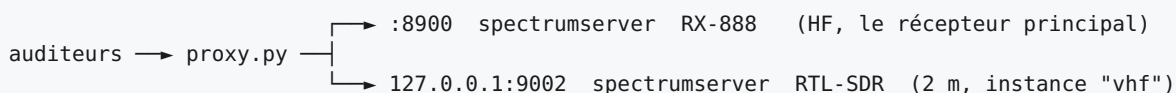
Le plus simple : lancez `./add-receiver.sh` (les installeurs le proposent à la fin). Il demande quel récepteur et ce qu'il doit couvrir, installe son pilote, écrit tout ce que décrivent la [section 3](#) et la [section 4](#), et propose de démarrer le récepteur et de redémarrer le proxy. Le reste de ce manuel explique ce qu'il fait, pour le jour où vous voulez modifier quelque chose à la main.

Sommaire

1. Comment cela fonctionne
2. Deux façons de publier les récepteurs
3. Ajouter un second récepteur
4. `receivers.toml`
5. Démarrer et arrêter
6. Ce que voient les auditeurs
7. Le 5-mètre au-dessus de 30 MHz
8. Sécurité
9. Combien de récepteurs tiennent
10. Pièges à connaître

1. Comment cela fonctionne

Chaque récepteur est un spectrumserver complet et indépendant, avec son propre SDR, sa propre configuration et son propre port interne. Ils ne partagent jamais d'échantillons, de réglages ni d'auditeurs. Ce qui les relie, c'est `proxy.py`, le proxy inverse qui sert déjà le panneau d'administration : il lit `receivers.toml` et envoie chaque requête à un récepteur.



Le proxy décide d'après ces éléments, dans cet ordre :

1. `?rx=<id>` dans l'adresse — `http://votre.hote:8899/?rx=vhf`. La page l'ajoute à chaque socket, requête et lien qu'elle ouvre, si bien qu'un onglet reste sur son récepteur.
2. Le cookie `rx`, que le proxy pose chaque fois qu'une requête porte un `?rx=` valide.
3. L'en-tête `Host`, si un récepteur cite ce nom dans `hostnames`.
4. Sinon le récepteur par défaut.

Le **récepteur principal** est celui que l'on démarre comme d'habitude (`./start-rx888mk2.sh`). Tout autre récepteur est une **instance nommée** : démarrée avec `INSTANCE=<nom>`, configurée dans `instances/<nom>/`, avec son propre journal (`logwebsdr-<nom>.txt`), journal du serveur, verrou du watchdog et FIFO. Les processus se distinguent par une étiquette `PHANTOMSDR_INSTANCE` dans leur environnement, pas par leur nom ; deux récepteurs peuvent donc exécuter exactement le même programme — deux `rtl_sdr`, ou `rx_sdr` pour un RSP1A et un Airspy — et le redémarrage de l'un ne touche jamais l'autre.

2. Deux façons de publier les récepteurs

A — le récepteur principal garde son propre port. Le récepteur principal reste exactement où il était (par exemple `:8900`), ses auditeurs ne remarquent rien, et l'on atteint les autres récepteurs par le port du proxy (`proxy_port` dans `admin_config.json`, par exemple `:8899`) avec `?rx=`. Rien ne bouge et rien ne quitte l'antenne ; en contrepartie il y a deux ports publics.

```

HF : http://votre.hote:8900/
2 m : http://votre.hote:8899/?rx=vhf
  
```

B — un seul port public pour tout. Le proxy prend le port public et le récepteur principal passe sur un port interne. Définissez `[front] port` dans `receivers.toml`, déplacez le `[server] port` du récepteur principal et mettez-y `host = "127.0.0.1"`, mettez le nouveau port interne dans `public_port` de `admin_config.json` et ajoutez `public_port = <le port public>` sous `[websdr]`, pour que l'annuaire continue d'annoncer le bon. Tout se trouve alors derrière un seul port ; le prix est une courte coupure pendant le déplacement.

```

HF : http://votre.hote:8900/
2 m : http://votre.hote:8900/?rx=vhf
  
```

Le sélecteur de récepteur fonctionne de la même façon dans les deux cas. La disposition A est le début le plus sûr ; B peut suivre plus tard sans rien changer au second récepteur.

3. Ajouter un second récepteur

add-receiver.sh effectue toutes les étapes ci-dessous — lancez-le et répondez à ses questions ; il installe aussi le pilote (`setup-rtlsdr.sh` pour un RTL-SDR, le `setup-*.sh` correspondant pour les autres), choisit un port interne libre et nomme le récepteur d'après ce qu'il couvre. Les étapes sont détaillées ici pour que vous puissiez vérifier son travail ou le modifier plus tard.

L'exemple ajoute un RTL-SDR Blog V4 pour le 2 m comme instance `vhf`, à côté d'un RX-888 qui garde le port 8900 (disposition A).

1. Le pilote. Le RTL-SDR Blog V4 a besoin du pilote de RTL-SDR Blog plutôt que du paquet `rtl-sdr` de la distribution, et le pilote DVB-T du noyau doit libérer la clé. `install.sh` (option de récepteur 2, « RTL-SDR Blog V4 : oui ») fait les deux. Vérifiez avec `rtl_test -t`, qui doit afficher `RTL-SDR Blog V4 Detected`.

2. Le dossier de l'instance. Tout ce qui appartient au récepteur se trouve dans `instances/vhf/`, jamais versionné et jamais écrasé par une mise à jour :

```
instances/vhf/
├─ config.toml          la configuration de son spectrumserver (obligatoire)
├─ instance.env         facultatif : paramètres du récepteur et cœurs de CPU
├─ markers.json         ses propres repères
└─ www/                 ses propres copies des fichiers de station de la page
    ├─ site_information.json
    └─ wf-message.json
```

`spectrumserver` tourne avec `instances/vhf/` comme répertoire de travail ; l'historique du chat, les repères, le FFTW wisdom et `logs/` sont donc aussi les siens.

3. config.toml. Partez de `config-rtl.toml` et modifiez :

```
[server]
port=9002                # son propre port interne
host="127.0.0.1"         # joignable uniquement par le proxy
html_root="www/"         # ses quelques fichiers propres...
html_fallback_root="../../frontend/dist/" # ...et la page commune pour le reste

[input]
sps=2400000              # identique à -s dans instance.env
frequency=145000000       # identique à -f dans instance.env

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

[kiwi_emulation]
enabled = false          # voir les pièges plus bas
```

`html_fallback_root` permet à l'instance de ne garder que ses propres `site_information.json` et `wf-message.json`, tandis que tout le reste — la page compilée elle-même — vient de `frontend/dist` ; une seule compilation du frontend atteint ainsi chaque récepteur. Une copie de `frontend/dist` faite de liens symboliques ne fonctionne pas : le serveur refuse les fichiers qui aboutissent hors de son propre dossier.

4. instance.env. Lu par le script de démarrage après ses propres réglages :

```
RX_ARGS="-f 145000000 -s 2400000 -g 29.7 -" # -f et -s identiques à config.toml
SPECTRUM_CORES=8-11                       # hors des cœurs du récepteur principal
```

5. Les informations de sa page. Copiez `frontend/site_information.json` dans `instances/vhf/www/` et modifiez : `siteReceiver`, `siteAntenna`, `siteSDRBaseFrequency` et `siteSDRBandwidth` (ils décident quels boutons de bande apparaissent), ainsi que les clés décrites à la [section 6](#).

6. Démarrez-le.

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

7. Inscrivez-le dans receivers.toml et redémarrez le proxy (`sudo systemctl restart phantomsdr-proxy`). Le second récepteur est maintenant à `http://votre.hote:8899/?rx=vhf`, et les deux pages affichent le sélecteur.

4. receivers.toml

Copiez `receivers.toml.example` en `receivers.toml`. Sans ce fichier, le proxy sert un seul récepteur, comme toujours.

```
# [front]
# port = 8900          # disposition B seulement : le proxy écoute aussi sur ce port public

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

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

Clé	Signification
<code>id</code>	Nom court utilisé dans <code>?rx=</code> ; lettres, chiffres, <code>-</code> et <code>_</code>
<code>name</code>	Le texte du bouton du récepteur
<code>port</code>	Le <code>[server]</code> port de son spectrumserver
<code>host</code>	Facultatif ; où tourne ce spectrumserver (par défaut <code>127.0.0.1</code>)
<code>default</code>	Le récepteur des requêtes qui n'en nomment aucun ; un seul au plus
<code>launcher</code>	Son <code>start-*.sh</code> , pour <code>start-all.sh</code> ; à omettre pour un récepteur que cet ordinateur ne démarre pas
<code>instance</code>	Le nom de son instance ; à omettre pour le récepteur principal
<code>url</code>	Où son bouton envoie l'auditeur ; sans elle, <code>/?rx=<id></code> sur le proxy
<code>hostnames</code>	Noms DNS facultatifs qui mènent directement à ce récepteur

Le proxy sert aussi la liste sous `/receivers.json` (identifiants, noms et liens — jamais les ports internes) ; c'est ce que lit le sélecteur.

5. Démarrer et arrêter

Commande	Effet
<code>./start-rx88mk2.sh</code>	Démarre ou redémarre le récepteur principal
<code>INSTANCE=vhf ./start-rtl.sh</code>	Démarre ou redémarre le récepteur vhf
<code>./start-all.sh</code>	Démarre chaque récepteur de <code>receivers.toml</code> qui a un <code>launcher</code>
<code>./stop-websdr.sh vhf</code>	Arrête seulement vhf
<code>./stop-websdr.sh main</code>	Arrête seulement le récepteur principal
<code>./stop-websdr.sh</code>	Arrête tous les récepteurs, comme toujours

Tant qu'un récepteur est arrêté, son bouton disparaît du sélecteur — le proxy ne recense que les récepteurs qui répondent, et les pages ouvertes relisent la liste une fois par minute — et il revient quand le récepteur redémarre ; s'il ne reste qu'un récepteur en marche, la ligne *Receivers:* disparaît entièrement. Un récepteur arrêté revient toutefois de lui-même chaque fois que tous les récepteurs sont démarrés : `./start-all.sh`, le **Restart** du panneau d'administration et un redémarrage par la protection thermique démarrent tout ce que liste `receivers.toml`. Pour en garder un éteint durablement, supprimez (ou commentez) son bloc `[[receiver]]` et redémarrez le proxy.

Réglez le **Default start script** du panneau d'administration sur `start-all.sh`. Son Stop arrête déjà tous les récepteurs, et son Restart ainsi que la protection thermique utilisent le script de démarrage ; avec `start-all.sh`, ils ramènent tous les récepteurs au lieu du seul principal.

L'auxiliaire RADE n'appartient qu'au récepteur principal. Une instance nommée ne le démarre ni ne l'arrête jamais.

6. Ce que voient les auditeurs

Le sélecteur de récepteur. Sur une station à plusieurs récepteurs, l'en-tête de la page reçoit une ligne à lui, *Receivers:*, avec un bouton par récepteur ; le récepteur actuel apparaît en jaune. La page */mobile* a les mêmes boutons en deuxième rangée de sa barre supérieure, et ils ouvrent la page */mobile* de l'autre récepteur.

Les informations propres à chaque récepteur. Une page ouverte sur un second récepteur charge le `site_information.json` de ce récepteur avant de démarrer ; les boutons de bande, la fréquence de départ, la liste des auditeurs, les informations de station et les liens *Receiver* et *Antenna* sont donc tous les siens. Une visite ordinaire du récepteur principal ne fait aucune requête supplémentaire.

Les clés de `site_information.json` concernées :

Clé	Signification
<code>siteReceiverId</code>	Le récepteur auquel appartient cette page (son <code>id</code>) ; à omettre sur le récepteur principal
<code>siteReceiversList</code>	Où le sélecteur lit la liste, p. ex. <code>http://votre.hote:8899/receivers.json</code> ; nécessaire sur un récepteur dont la page n'est pas servie par le proxy
<code>siteReceiverURL</code>	Vers où pointe le nom de <i>Receiver</i> dans <i>Open Additional Info</i>
<code>siteAntennaURL</code>	Vers où pointe le nom de <i>Antenna</i> ; "" affiche le nom sans lien

Quand le sélecteur reste caché. Le sélecteur n'affiche qu'une liste qui cite l'hôte sous lequel la page a été ouverte. Ouverte par l'adresse du réseau local (`http://192.168.1.10:8900/`), il reste caché ; ouverte en `http://votre.hote:8900/`, il apparaît. C'est voulu : une station qui a copié le

`site_information.json` d'une autre sans le modifier n'affiche jamais les récepteurs de celle-ci comme les siens.

7. Le S-mètre au-dessus de 30 MHz

La norme de l'IARU Région 1 place S9 à -73 dBm sous 30 MHz et à **-93 dBm au-dessus**, avec six dB par point S dans les deux cas. Les S-mètres (l'aiguille analogique, la barre numérique et la barre de / mobile) suivent la fréquence accordée : au-dessus de 30 MHz ils indiquent des points S VHF, en dessous comme avant. Les valeurs en dBm et dB μ V ne changent jamais. Si vous préférez que les instruments restent à 0 sur un canal vide, comme ceux d'un transceiver VHF/UHF, ajoutez `"siteSMeterGateDb": 6` au `site_information.json` du récepteur (sans recompilation) : au-dessus de **60 MHz**, l'aiguille et la barre ne montent alors que lorsqu'un signal dépasse de ce nombre de dB le bruit de son propre voisinage (le point le plus fort de la bande passante face au spectre ± 100 kHz autour). C'est **désactivé par défaut** — un instrument étalonné qui montre le vrai bruit de bande est la lecture la plus honnête. La page / mobile, qui n'a pas de cascade, estime le bruit à partir du niveau reçu lui-même.

Étalonnez les dBm d'un récepteur VHF avec `analog_smeter_offset` (aiguille et valeurs) et `smeter_offset` (barre numérique) sous `[input]` dans son `config.toml`, puis redémarrez ce récepteur. Fixez d'abord le gain : un RTL-SDR donne ses niveaux par rapport à sa propre pleine échelle, donc chaque changement de gain déplace les lectures. Sans générateur, une charge de 50 Ω à la place de l'antenne, en USB avec un filtre de 2,7 kHz, doit indiquer environ -136 dBm (le bruit thermique dans 2,7 kHz vaut $-139,7$ dBm, plus le facteur de bruit de la clé).

Pour un RTL-SDR, `add-receiver.sh` écrit un étalonnage de départ : un gain fixe `-g 29.7` dans `instance.env` (le gain automatique du tuner ne s'étalonne pas) ainsi que `analog_smeter_offset=-55` et `smeter_offset=-38`, mesurés sur un RTL-SDR Blog V4 à ce gain avec un signal de -71 dBm (63 μ V). Une autre clé du même modèle s'en écarte en général de quelques dB seulement ; vérifiez-la avec un signal connu et réétalonnez si vous changez le gain.

8. Sécurité

Placer le proxy devant les auditeurs a fermé deux failles, toutes deux corrigées dans la v5.0.0 :

- **L'expulsion du sysop à travers le proxy.** `spectrumserver` n'accepte `/~kick` que depuis son propre ordinateur, et pour lui chaque requête relayée par le proxy vient de son propre ordinateur. Quiconque atteignait le port du proxy pouvait donc déconnecter et bannir n'importe quel auditeur. Le proxy ne répond désormais à `/~kick` que pour les clients du même ordinateur.
- **Adresses de client falsifiées.** `spectrumserver` croyait l'en-tête `X-Forwarded-For` de n'importe qui ; un visiteur pouvait donc se dire `127.0.0.1` et passer outre les limites par adresse. L'en-tête n'est désormais cru que s'il vient d'un proxy local, et le proxy supprime toute copie envoyée par un client avant d'ajouter la sienne.

Liez chaque récepteur joint par le proxy à `127.0.0.1` (`[server] host`), afin que son port ne puisse pas servir à contourner le proxy. Les limites par adresse de [Limites de connexion](#) s'appliquent par récepteur.

9. Combien de récepteurs tiennent

Le logiciel ne fixe aucune limite ; le matériel, si. Les récepteurs à bande étroite coûtent peu : un RTL-SDR à 2,4 Msps utilise environ 5 à 10 % d'un cœur, quelques dizaines de Mo de mémoire et environ 38 Mbit/s d'USB.

- **L'USB 2.0 est le plafond habituel.** Tous les périphériques USB 2 partagent un bus de 480 Mbit/s, même dans une prise USB 3 bleue. Trois à quatre RTL-SDR par bus est un chiffre sûr ; ajoutez-les un par un et surveillez chaque cascade pour repérer des trous.
- **Alimentation :** au-delà de deux clés, utilisez un hub alimenté.
- **Un second récepteur large bande** (un autre RX-888, ou un HackRF à 20 Msps) est une autre affaire : il dispute au premier l'USB 3, le CPU et le GPU.
- **Les auditeurs coûtent plus que les récepteurs.** Chaque auditeur coûte du CPU sur son récepteur et de la bande passante montante ; le nombre total d'auditeurs compte donc plus que le nombre de récepteurs.

10. Pièges à connaître

- **Deux clés du même modèle ont le même numéro de série.** Chaque RTL-SDR Blog V4 indique 00000001. Donnez à chacune le sien avec `rtl_eeprom -s <série>` (une seule clé branchée à la fois) et nommez-le dans `instance.env` (`RX_ARGS="-d <série> ..."`), sinon après un redémarrage les clés peuvent échanger leurs récepteurs.
- **Les clients Kiwi ne peuvent pas choisir de récepteur.** Un client KiwiSDR appelle une simple adresse et un port ; à travers le proxy il aboutit toujours au récepteur par défaut. Laissez `[kiwi_emulation]` désactivé sur les autres.
- **Les cookies valent par hôte, pas par port.** Un visiteur qui a utilisé le second récepteur sur `:8899` emporte aussi `rx=vhf` vers `:8900`. C'est pourquoi une page décide quel récepteur elle est d'après `siteReceiverId`, jamais d'après le cookie.
- **Les entrées d'annuaire.** Donnez à chaque récepteur son propre nom dans `[websdr]`, et sur un récepteur derrière le proxy mettez `[websdr] public_port` au port qu'utilisent les auditeurs, sinon l'entrée annonce le port interne.
- **Une compilation du frontend affiche brièvement « Not Found ».** `frontend/dist` est recompilé sur place ; pendant quelques secondes de compilation, la page manque sur chaque récepteur.
- **Rien ne démarre les récepteurs au démarrage de l'ordinateur**, à moins que vous ne le prévoyiez. Après un redémarrage, `./start-all.sh` les ramène tous.

CHAPITRE 7

Serveur de statistiques

sdr-stats/README.md

Guide d'installation de System Stats

Un guide complet pour ajouter une surveillance de serveur en temps réel à votre application PhantomSDR à l'aide du script d'installation automatisé.

Aperçu

Cette fonctionnalité ajoute un bouton **Stats** à votre interface PhantomSDR qui affiche des informations système en temps réel :

- Utilisation du CPU, cœurs, fréquence, température et principaux processus
- Utilisation de la mémoire (utilisée/totale/pourcentage)

Les statistiques se mettent à jour automatiquement toutes les 5 secondes tant que la fenêtre modale est ouverte.

À ne pas confondre avec la page « Graphiques » du panneau d'administration. Cette fenêtre est un instantané qui se rafraîchit tant qu'elle est ouverte et ne conserve aucun historique. Le panneau d'administration dispose d'une page **Graphiques** distincte qui échantillonne la fréquence du processeur, la charge, la température et les utilisateurs connectés toutes les 2 secondes et les trace sur les 15 dernières minutes à 24 heures — voir le [guide du panneau d'administration](#).

Prérequis

- **Serveur Linux** (Ubuntu/Debian recommandé)
- **Connexion Internet** (pour télécharger Node.js et les paquets)
- **Accès Sudo** (pour la configuration du service systemd)
- **PhantomSDR-Plus** déjà installé et en cours d'exécution
- **Im-sensors** (recommandé pour la température CPU Intel/AMD - installé automatiquement par le script)

Méthodes d'installation

REMARQUE

L'installateur principal le propose déjà. `./install.sh` — et les quatre installateurs par distribution — exécutent `install-stats-server.sh` par défaut pendant l'installation normale de PhantomSDR-Plus : sur une machine neuve, le serveur de statistiques est donc généralement déjà en place. Utilisez cette page pour l'installer séparément, ou pour changer ensuite son port, son adresse ou son service.

Option 1 : Installation automatisée (Recommandée)

Le script automatisé s'occupe de tout pour vous.

Étape 1 : Exécuter le script

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

IMPORTANT : N'exécutez PAS en tant que root ou avec sudo. Exécutez en tant qu'utilisateur normal.

Étape 2 : Suivre les invites interactives

Le script vous guidera à travers :

1. Installation de Node.js (si nécessaire)

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

Le script installe Node.js 22 via nvm, sur toute distribution Linux — sans root et sans source apt. Il pose la même question lorsque Node.js est présent mais antérieur à la version 22.

2. Répertoire d'installation

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

Appuyez sur Entrée pour la valeur par défaut, ou spécifiez un chemin personnalisé.

3. Configuration du port (Améliorée !)

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

Le script valide votre port :

- Vérifie qu'il s'agit d'un nombre valide
- Avertit si le port < 1024 (nécessite des privilèges root)
- Valide la plage (1-65535)
- Vérifie si le port est déjà utilisé
- Indique quel processus l'utilise s'il est occupé
- Vous permet de choisir un port différent

Exemples de scénarios :

Utilisation du port par défaut :

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

Choix d'un port personnalisé :

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

Port déjà utilisé :

```
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
```

Port invalide :

```
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. Adresse du serveur

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

Entrez le domaine public ou l'adresse IP de votre serveur.

5. Confirmation

```
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. Configuration du service Systemd (Optionnel)

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

Recommandé : Choisissez **y** pour que le serveur démarre automatiquement au démarrage.

Étape 3 : Installation terminée ! ✓

Le script affichera un résumé :

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

Configurer votre application PhantomSDR

Une fois le script terminé, vous devez mettre à jour votre application Svelte.

Étape 1 : Mettre à jour `site_information.json`

Comme `site_information.json` a déjà été modifié lors de la configuration initiale, il vous suffit d'ajouter la nouvelle ligne `siteStats` avec votre port :

```
{
  "siteSysop": "your name or callsign",
  "siteSysopEmailAddress": "mail@mail.net",
  "siteGridSquare": "QTH locator",
  "siteCity": "City Country",
  "siteInformation": "https://github.com/sv1btl/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", ← AJOUTEZ CETTE LIGNE (utilisez votre port)
  "siteSDRBaseFrequency": 0,
  "siteSDRBandwidth": 30000000,
  "siteRegion": 1,
  "siteChatEnabled": true
}
```

Note : Utilisez le port que vous avez sélectionné pendant l'installation !

Étape 2 : Mettre à jour `App.svelte` (la dernière version GitHub inclut déjà le fichier mis à jour)

Effectuez ces **4 modifications** dans votre fichier `App.svelte` :

Modification 1 : S'assurer que `siteStats` est déclaré (vers la ligne 50)

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

Modification 2 : Mettre à jour l'objet `systemStats` (vers la ligne 530)

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

Modification 3 : Mettre à jour l'URL de récupération (vers la ligne 541)

```
async function fetchSystemStats() {
  try {
    const response = await fetch(`${siteStats}/api/system-stats`); // ← MODIFIEZ CETTE LIGNE
    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);
  }
}
```

Modification 4 : Ajouter l'affichage des principaux processus (dans la section CPU, vers la ligne 3838)

Ajoutez ce code après la section température :

```

{#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}

<!-- AJOUTEZ TOUTE CETTE SECTION : -->
{#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}

```

Étape 3 : Recompiler votre application

```

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

```

Tester l'installation

Test 1 : Vérifier le point de terminaison de l'API

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

Sortie attendue (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
  }
}
```

Test 2 : Vérifier l'état du service

Si vous l'avez installé en tant que service :

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

Devrait afficher : Active: active (running)

Test 3 : Tester dans le navigateur

1. Ouvrez votre interface web PhantomSDR
2. Faites défiler jusqu'à la section **Additional Info**
3. Cliquez sur le bouton **Open Additional Info**
4. Recherchez le bouton **Stats** à côté des infos PC
5. Cliquez dessus pour ouvrir la fenêtre modale des statistiques
6. Vérifiez que les statistiques s'affichent et se mettent à jour

Gestion du service

Si vous l'avez installé en tant que service systemd, utilisez ces commandes :

```
# 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
```

Installation manuelle (Option 2)

IMPORTANT

Vous n'avez pas besoin de cette section pour une installation normale. `install-stats-server.sh` fait tout cela pour vous et l'installateur principal le lance par défaut. Ne la suivez que pour une installation manuelle ou pour réparer un élément à la main.

⚠ **Avertissement** : L'installation manuelle peut utiliser des fichiers obsolètes. Le **script automatisé est fortement recommandé** car il :

- Installe la dernière version avec une détection de température améliorée
- Installe et configure automatiquement lm-sensors
- Gère automatiquement toutes les dépendances

Si vous préférez tout de même l'installation manuelle :

1. Installer 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. Installer lm-sensors (Requis pour la température Intel/AMD)

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

3. Créer le répertoire

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

4. Télécharger les derniers fichiers

Important : les fichiers du serveur ne sont plus conservés en copie dans le dépôt - c'est le script d'installation qui les génère.

À la place, exécutez le script automatisé une fois pour générer les derniers fichiers, puis copiez-les :

```
# 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/
```

OU créez manuellement les fichiers en utilisant le code le plus récent du script d'installation.

5. Modifier le port (si nécessaire)

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

Modifiez la ligne 6 :

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

6. Installer les dépendances

```
npm install
```

7. Test d'exécution

```
npm start
```

8. Configurer le service (Optionnel)

Suivez la configuration du service systemd de la section d'installation automatisée.

Changer le port après l'installation

Si vous devez changer le port après l'installation :

Étape 1 : Modifier le fichier du serveur

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

Modifiez la ligne 6 :

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

Étape 2 : Mettre à jour site_information.json

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

Étape 3 : Redémarrer le service

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

Étape 4 : Recompiler votre application Svelte

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

Dépannage

Problème : Port déjà utilisé

Erreur :

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

Solution :

1. Trouvez ce qui utilise le port :

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

1. Arrêtez ce processus :

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

1. Redémarrez le service :

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

Alternative : Exécutez à nouveau le script d'installation et choisissez un port différent lorsque cela vous est demandé.

Problème : Le service ne démarre pas

Erreur :

```
status=217/USER
```

Solution :

1. Vérifiez le fichier de service :

```
cat /etc/systemd/system/sdr-stats.service
```

1. Vérifiez que la ligne `User=` correspond à votre nom d'utilisateur :

```
whoami
```

1. Modifiez si nécessaire :

```
sudo nano /etc/systemd/system/sdr-stats.service
```

1. Rechargez et redémarrez :

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

Problème : Les statistiques ne s'affichent pas dans le navigateur

Symptômes : La fenêtre modale s'ouvre mais pas de données, ou affiche des valeurs à 0

Solutions :

1. Vérifiez la console du navigateur (F12) pour les erreurs
2. Vérifiez que l'API est accessible :

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

1. Vérifiez que le service est en cours d'exécution :

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

1. Vérifiez que `siteStats` dans `site_information.json` correspond à l'adresse et au port de votre serveur
2. Vérifiez les erreurs CORS - assurez-vous que le serveur autorise votre domaine

Problème : La température affiche null

Cause : Lecture de température non disponible sur votre système

Solutions :

1. **Pour la plupart des systèmes Linux :** Vérifiez si la zone thermique existe :

```
cat /sys/class/thermal/thermal_zone0/temp
```

1. **Installez lm-sensors :**

```
sudo apt-get install lm-sensors
sudo sensors-detect
sensors
```

1. **Note** : La température peut ne pas être disponible sur :
2. Les machines virtuelles
3. Certains fournisseurs VPS
4. Windows Subsystem for Linux (WSL)
5. Les systèmes non-Linux

C'est normal - le reste des statistiques fonctionnera quand même !

Problème : Les valeurs des statistiques ne correspondent pas à `top`

Cause : La période d'échantillonnage du CPU était trop courte dans les versions antérieures

Solution : La version actuelle utilise un échantillonnage d'une seconde pour des lectures précises. Si vous avez une ancienne installation :

1. Mettez à jour le fichier du serveur :

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

1. Trouvez la ligne 18 et modifiez :

```
}, 100); // Old value
```

en :

```
}, 1000); // New value (1 second)
```

1. Redémarrez :

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

Ou réexécutez simplement le script d'installation pour obtenir la dernière version.

Problème : Node.js introuvable

Erreur :

```
node: command not found
```

Solution :

1. Installez 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. Vérifiez :

```
node --version
npm --version
```

Ce que vous verrez

Une fois installé et configuré, la fenêtre modale des statistiques affiche :

Section CPU

- **Utilisation** : Pourcentage d'utilisation global du CPU
- **Cœurs** : Quelle part du CPU est réellement utilisée, exprimée en cœurs entiers — par exemple `1.3 / 12 in use` signifie que la machine effectue le travail de 1,3 cœur. C'est exact et sans aucun seuil. Ce n'est volontairement pas un décompte de cœurs occupés : l'ordonnanceur répartit le travail d'un cœur sur plusieurs, si bien qu'un tel décompte ne ferait qu'approcher ce chiffre.
- **Fréquence** : Fréquence actuelle moyenne du CPU sur tous les cœurs en GHz, avec le cœur le plus rapide entre parenthèses (si disponible)
- **Température** : Température du CPU en Celsius (si disponible), avec code couleur : vert jusqu'à 69 °C, orange 70-79 °C, rouge à partir de 80 °C
- **Principaux processus** : Les 5 processus les plus gourmands en CPU avec des pourcentages à code couleur :
 - Vert : jusqu'à 50 % CPU
 - Orange : 51-80 % CPU
 - Rouge : au-delà de 80 % CPU

Section mémoire

- **Utilisée** : Mémoire en cours d'utilisation (Go)
- **Totale** : Mémoire système totale (Go)
- **Utilisation** : Pourcentage d'utilisation de la mémoire (à code couleur)

🔄 Mise à jour automatique

- Les statistiques se rafraîchissent toutes les **5 secondes** tant que la fenêtre modale est ouverte
- La mise à jour s'arrête lorsque vous fermez la fenêtre modale (économise les ressources)

Détection de la température

Le serveur de statistiques utilise une détection de température avancée à méthodes multiples :

Systèmes pris en charge :

- **Processeurs Intel** : Lit depuis les capteurs coretemp via lm-sensors
- **Processeurs AMD** : Lit depuis les capteurs k10temp (Tdie/Tctl)
- **Processeurs ARM** : Lit depuis les zones thermiques (Raspberry Pi, etc.)

Exigences :

- **Intel/AMD** : Nécessite le paquet `lm-sensors` (installé automatiquement par le script)
- **ARM/Raspberry Pi** : Fonctionne d'emblée, aucun paquet supplémentaire nécessaire

Priorité d'installation :

1. Essaie la commande `sensors` pour la température Package/Core (Intel/AMD)
2. Se rabat sur la lecture directe de `coretemp` (Intel)
3. Se rabat sur `thermal_zone0` (ARM/Raspberry Pi)

Si la température affiche `null`, installez `lm-sensors` :

```
sudo apt-get install lm-sensors
sudo sensors-detect --auto
sudo systemctl restart sdr-stats.service
```

Résumé des fonctionnalités

Surveillance en temps réel - Statistiques système en direct depuis votre serveur **Métriques CPU** - Utilisation, cœurs, température, principaux processus **Suivi de la mémoire** - Utilisée, totale et pourcentage **Mises à jour automatiques** - Se rafraîchit toutes les 5 secondes **Code couleur** - Indicateurs visuels pour les avertissements **Léger** - Utilisation minimale des ressources

Installation facile - Le script automatisé s'occupe de tout **Validation du port** - Sélection de port intelligente avec détection des conflits **Gestion du service** - Démarrage automatique au démarrage

Notes de sécurité

- Le serveur de statistiques accepte les connexions de toute origine (CORS : *)
- Pour la production, envisagez de restreindre CORS à votre domaine uniquement
- La sélection du port valide l'entrée et vérifie les conflits
- La lecture de la température du CPU est en lecture seule et sûre
- Aucune information système sensible n'est exposée

Support et ressources

Fichiers inclus :

Dans ce dossier (le script d'installation lui-même se trouve à la racine, sous `PhantomSDR-Plus/install-stats-server.sh` - il n'en existe qu'une seule copie) :

- `package.json` - Dépendances, identiques à ce qu'écrit le script
- Ce document, également disponible en de, el, es, fr, hr et ru

Créés par le script dans `~/sdr-stats-server/` :

- `system-stats-server.js` - Serveur backend, généré avec le port choisi
- `package.json` - Dépendances
- `node_modules/` - installé par `npm install`

Obtenir de l'aide :

- Consultez la section de dépannage ci-dessus
- Examinez les journaux du service : `sudo journalctl -u sdr-stats.service -n 50`
- Testez l'API manuellement : `curl http://localhost:3001/api/system-stats`

- Vérifiez les fichiers de configuration

Désinstaller le serveur de statistiques :

- Supprimez le dossier /sdr-stats-server
- Si vous avez créé un service, supprimez-le simplement en utilisant :

```
sudo rm /etc/systemd/system/sdr-stats.service`
```

- Modifiez `App.svelte` : Trouvez la partie du code :

```

        <!-- 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>
    ...
    et remplacez par :
```

```

        <!-- 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>
        -->
```

...

- Recompilez à l'aide de la commande :

```

    cd PhantomSDR-Plus
    ./recompile.sh
    \\\

- Redémarrez le PC et lancez le serveur

---

**Auteur : ** Créé pour PhantomSDR-Plus
**Version : ** 1.1 (Détection améliorée de la température CPU pour Intel/AMD) **Mis à jour : **
Janvier 2026
**Licence : ** MIT

---

## Référence rapide des commandes

```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

```

Bonne surveillance !

## CHAPITRE 8

# Décodeurs

DECODERS.md

## PhantomSDR-Plus — Manuel d'utilisation des décodeurs

Ce guide couvre tous les décodeurs intégrés à PhantomSDR-Plus. Tous les décodeurs partagent la même procédure d'activation, décrite ci-dessous, suivie des instructions de configuration propres à chacun.

### Table des matières

1. Comment démarrer un décodeur
2. FT8
3. FT4-FT2
4. JS8
5. CW — code Morse
6. QRSS Grabber
7. WSPR
8. FAX HF / WEFAX
9. NAVTEX
10. FSK / RTTY — y compris PSK31 et Olivia
11. SSTV
12. Conseils généraux

### 1. Comment démarrer un décodeur

Tous les décodeurs se trouvent dans la section **Decoder Options**, située sous les commandes du spectrogramme audio, sur le panneau principal.

#### Étapes :

1. Cliquez sur le bouton **Decoder: OFF** pour le faire passer à **Decoder: ON** (il devient bleu lorsqu'il est actif).
2. Ouvrez le menu déroulant qui apparaît à droite du bouton et sélectionnez le décodeur souhaité.
3. Le panneau du décodeur choisi apparaît sous les commandes — suivez les instructions propres à ce décodeur dans la section correspondante de ce guide.
4. Pour arrêter le décodage, sélectionnez — **Select decoder** — dans le menu déroulant, ou cliquez sur le bouton **Decoder: ON** pour le remettre sur OFF.

#### Boutons de décodeur en une pression

Le menu déroulant n'est pas la seule voie. Le panneau principal comporte une rangée de boutons **Decoders** — juste sous **Wheel Tuning Steps** — avec un bouton par décodeur :

Bouton	Décodeur	Bouton	Décodeur
<b>FT8</b>	FT8	<b>SSTV</b>	SSTV
<b>FT4</b>	FT4	<b>NAVTEX</b>	NAVTEX
<b>FT2</b>	FT2	<b>RTTY</b>	FSK / RTTY
<b>CW</b>	CW		
<b>WSPR</b>	WSPR	<b>FAX</b>	HF FAX / WEFAX

Une pression effectue toute la séquence d'un coup : elle sélectionne le décodeur, met le décodeur sur ON et amène sa fenêtre à l'écran. Le bouton devient bleu tant que son décodeur tourne. **Appuyez de nouveau sur le même bouton pour arrêter le décodeur** — sa fenêtre se ferme avec lui.

**RADEL** et **RADEU** sont volontairement absents de cette rangée. Ce sont des modes de voix numérique et non des décodeurs de texte : ils disposent de leur propre paire de boutons à côté du titre **Modes selector**, ainsi que dans les fenêtres **Modes** et **Bands**. Leur fonctionnement est identique — une pression pour démarrer, une seconde pour arrêter. Voir le [manuel RADE](#).

Les boutons, le menu déroulant et le bouton ON/OFF pilotent le même état : quel que soit celui que vous utilisez, les autres suivent.

Un seul décodeur peut être actif à la fois. Passer à un autre décodeur arrête automatiquement le précédent.

Le **grabber QRSS** ne fait pas partie de ce menu déroulant : il possède sa propre section **QRSS** et peut fonctionner en même temps qu'un décodeur.

**Le décodage s'exécute en arrière-plan.** SSTV, FAX HF, NAVTEX, FSK/RTTY et CW s'exécutent chacun dans un Web Worker distinct : le travail de décodage n'entre jamais en concurrence avec la lecture audio ou la cascade. Démarrer ou arrêter un décodeur n'interrompt pas l'audio, et l'interface reste réactive pendant la réception d'une image ou d'une page. Tous les décodeurs reçoivent l'audio brut prélevé *avant* l'AGC, la réduction de bruit et la coupure du son — couper le son du récepteur, ou modifier ces réglages selon votre confort d'écoute, n'affecte donc pas le décodage.

**Un décodeur en marche impose le mode et la bande passante.** Normalement le mode suit le plan de bande de `bands-config.js` : amenez l'accord dans un segment marqué LSB ou AM et le récepteur bascule dessus. Tant qu'un décodeur tourne, cela n'a plus lieu. Le décodeur conserve le mode qu'il lui faut (USB pour la plupart, le sien pour RADE) et la bande passante qu'il lui faut — PSK31 environ  $\pm 100$  Hz, Olivia toute sa largeur, RTTY son shift — et les deux survivent à un changement de fréquence, y compris un saut vers une autre bande. Sans cela, FT8 sur 40 m ferait passer le récepteur en LSB dès que l'accord bouge, et les bandes passantes étroites des décodeurs se rouvriraient au filtre SSB complet.

Le mode propre à la bande revient dès que vous arrêtez le décodeur. Vous pouvez toujours forcer le mode à la main : les boutons de mode sont un choix délibéré et l'emportent toujours. Le décodeur CW fait exception à tout ceci — il décode dans le mode où vous écoutez et ne prend jamais la main sur le récepteur.

## Decoder ID — reconnaître le mode automatiquement

Au-dessus des boutons de décodeur se trouve le cadre **Decoder ID**. Il répond à la question « *qu'est-ce que j'écoute ?* » lorsque vous tombez sur un signal numérique que vous n'arrivez pas à identifier, puis vous propose le décodeur correspondant en un clic.

Il est **désactivé par défaut** et ne coûte rien tant qu'il est éteint. Appuyez sur **On** et laissez le signal accordé. Pendant environ une demi-minute il affiche *Listening...* avec une barre de progression — les

mesures de temporisation doivent voir deux cycles FT8 complets avant d'avoir un sens — puis il nomme le mode :

Decoder ID [0n] NAVTEX / SIT0R-B 99% 170 Hz shift · 100.00 Bd · 43 dB S/N [ Use NAVTEX ]

- Le **mode** et un indice de **confiance**. Vert au-dessus de 75 %, jaune au-dessus de 50 %, orange en dessous — une réponse orange est une piste, pas un verdict.
- **Ce qui a réellement été mesuré** : largeur de bande occupée, écart entre tons, rapidité de modulation, durée de la salve, la grille UTC à laquelle il correspond, rapport signal/bruit. Ce sont les éléments de preuve derrière le verdict, afin que vous puissiez juger par vous-même.
- **Use** démarre le bon décodeur, exactement comme si vous aviez appuyé sur son bouton — et règle le récepteur pour travailler ce mode. Pour les modes qui vivent dans le panneau FSK, il sélectionne en plus la bonne variante — RTTY, Weather RTTY, PSK31 ou Olivia.
- Les candidats suivants figurent après **or**. Cliquez sur l'un d'eux pour l'essayer : une première réponse erronée ne vous coûte donc rien.
- \*\*\*\* signifie que la fréquence sur laquelle vous êtes accordé est une fréquence d'appel connue pour ce mode, et que cela a compté dans le classement.

Il ne change jamais votre décodeur de lui-même. Il ne fait que suggérer. Et dès que vous démarrez un décodeur — depuis la rangée de boutons, depuis la liste déroulante ou depuis le bouton **Use** de Decoder ID lui-même — Decoder ID s'éteint tout seul : son rôle était d'en choisir un, et le décodeur veut maintenant le même signal audio.

**Il règle le récepteur pour le mode.** Choisir un mode, c'est décider de le travailler : le récepteur suit donc. La fréquence accordée passe au milieu de la cascade et la vue s'établit sur environ 100 kHz autour d'elle, la bande latérale devient celle sur laquelle ce mode se travaille et la bande passante se resserre sur la fenêtre où vit ce mode — toute la sous-bande de 3 kHz pour FT8, FT4, FT2 et JS8, 1350-1650 Hz pour WSPR, 250-750 Hz pour NAVTEX, 800-2700 Hz pour HF FAX, 900-2600 Hz pour SSTV et  $\pm 250$  Hz de part et d'autre de l'accord pour la CW, qui bascule en outre le récepteur en CW. Une bande passante étroite tient les signaux voisins à l'écart du décodeur — exactement ce que vous auriez fait à la main. Deux exceptions : la SSTV conserve la bande latérale qu'elle a choisie elle-même — LSB sur 80 et 40 m par convention, où imposer l'USB inverserait sa correspondance de tons — et les modes FSK conservent la fenêtre que leur propre panneau déduit du shift ou de la largeur de bande de la variante, plus étroite que n'importe quelle valeur fixe. Appuyer sur **Use** pour le décodeur déjà en marche l'arrête toujours, et le récepteur reste alors exactement tel qu'il était.

**Les modes lents demandent plus de temps.** WSPR suit une grille de deux minutes et les images SSTV durent une à deux minutes, donc ces deux-là sont nommés plus tard que les autres — laissez-leur trois ou quatre minutes sur la fréquence. HF FAX et SSTV ne sont par ailleurs reconnus que s'ils sont accordés de la manière habituelle — la bande image entre environ 1500 et 2300 Hz en audio, là où leurs décodeurs l'attendent également. Changer de fréquence relance la mesure à zéro, car tout ce qui a été recueilli décrit la fréquence que vous venez de quitter.

**Ce qu'il reconnaît** : FT8, FT4, JS8, WSPR, CW, NAVTEX/SITOR-B, RTTY à 45,45 Bd, Weather RTTY (DWD), PSK31, Olivia, HF FAX, SSTV et DSC. Cette version ne comporte pas de décodeur DSC, donc DSC est nommé et marqué *no decoder*.

**Recentre & retry.** La position du signal dans la bande passante audio n'a aucune importance entre environ 400 Hz et 2700 Hz. En dehors de cette plage la mesure se dégrade, et un bouton jaune **Recentre & retry** apparaît. Une pression déplace l'accord pour que le signal se retrouve au milieu de la bande passante et lance une nouvelle mesure. C'est un unique déplacement calculé, pas une recherche.

### Trois limites assumées :

- **FT8, JS8 et FT2 sont le même signal.** Même modulation 8-FSK, même écart entre tons, et JS8 en vitesse Normal utilise la cadence de 15 secondes de FT8. Rien dans le signal lui-même ne les sépare. La fréquence, si : sur une fréquence d'appel FT8 il annonce FT8, sur une fréquence JS8 il annonce JS8, et partout ailleurs il annonce honnêtement la famille — *FT8 / JS8 / FT2*. Démarrez l'un ou l'autre décodeur et voyez lequel produit du texte.
- **Il se tait plutôt que de deviner.** En dessous d'environ 10 dB de rapport signal/bruit, il indique *Nothing above the noise* ou *Not recognised* au lieu de nommer un mode qu'il ne voit pas réellement. C'est délibéré : une réponse fausse et assurée est pire que pas de réponse du tout.
- **Un signal soumis au fading se lit autrement.** WSPR émet 110 secondes sans interruption, soit plus longtemps que ne dure un évanouissement QSB ordinaire : un fading profond découpe donc l'émission en morceaux et son rythme de deux minutes n'y est plus lisible. Il est alors retrouvé par une autre voie — tout l'historique d'écoute est comparé à la grille de deux minutes plutôt qu'aux émissions prises une à une — et la ligne de mesures affiche *120 s grid (through fading)*. C'est une preuve plus faible qu'une émission entendue en entier, aussi la confiance affichée est-elle volontairement plus basse. Un fading assez profond pour noyer le signal dans le bruit ne se rattrape pas du tout. Olivia est plus exigeante que les autres : elle n'est nommée qu'une fois sa rapidité de modulation mesurée, et se tait donc plus tôt.

## 2. FT8

**De quoi s'agit-il :** FT8 est un mode numérique à signaux faibles très répandu chez les radioamateurs du monde entier. Les transmissions durent exactement 15 secondes et le protocole peut copier des signaux jusqu'à 20-25 dB sous le plancher de bruit. C'est le mode le plus utilisé pour les contacts à longue distance (DX).

### Fréquences recommandées (USB)

Bande	Fréquence
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

### Configuration

1. Accordez-vous sur une fréquence FT8 listée ci-dessus et réglez le mode sur **USB**.
2. Activez le décodeur et sélectionnez **FT8** dans le menu déroulant. Ou appuyez simplement sur le bouton **FT8**.
3. Le panneau **FT8 Messages** apparaît automatiquement en dessous.

## Lecture des résultats

La liste des messages affiche les transmissions décodées à mesure qu'elles arrivent. Chaque cycle de 15 secondes produit un nouveau lot de messages. Le champ **Farthest** (en haut à droite du panneau) indique la plus grande distance décodée durant la session en cours, en kilomètres.

Format de message typique : CQ DX AA1BB FN31 — un appel CQ de l'indicatif AA1BB situé dans le carré locator FN31.

Le **SNR** affiché avec chaque décodage est un vrai rapport signal/bruit, mesuré sur le signal décodé par rapport au bruit qui l'entoure et ramené à une bande de 2500 Hz, comme le fait WSJT-X. Les stations faibles affichent des valeurs négatives, jusqu'à environ -20 dB, et les fortes des valeurs positives ; sur les mêmes enregistrements les deux programmes concordent en moyenne, en général à environ 3 dB près pour un décodage isolé. FT4 et FT2 utilisent la même méthode.

**Après une mise à jour :** le décodeur FT8/FT4/FT2 est un fichier à part que les navigateurs gardent longtemps en cache. Si tous les SNR restent entre environ 0 et +5 dB, rechargez la page à fond (**Ctrl+Maj+R**, ou **Cmd+Maj+R** sur Mac).

**Astuce :** FT8 est étroitement synchronisé dans le temps. Votre navigateur utilise l'horloge de votre ordinateur ; si celle-ci dérive de plus de quelques secondes, le décodage échouera. Gardez l'heure système synchronisée sur NTP.

## 3. FT4-FT2

**De quoi s'agit-il :** FT4 est une variante plus rapide de FT8, conçue pour l'exploitation en concours. Chaque cycle de transmission dure 7,5 secondes (la moitié de FT8), ce qui le rend deux fois plus rapide mais exige un signal légèrement plus fort. FT2 est une variante encore plus rapide de FT8. C'est un mode ultrarapide à 77 bits avec des périodes T/R de 3,75 secondes (= la moitié de FT4) — encore expérimental.

### Fréquences recommandées (USB)

Bande	Fréquence 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

Bande	Fréquence FT2
160 m	1.843 à 1.846
80 m	3.578 à 3.581
60 m	5.360 (vérifiez la réglementation régionale locale)
40 m	7.052 à 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

## Configuration

1. Accordez-vous sur une fréquence FT4 et réglez le mode sur **USB**.
2. Activez le décodeur et sélectionnez **FT4** ou **FT2** dans le menu déroulant. Ou appuyez simplement sur le bouton **FT4 / FT2**.
3. Le panneau **FT4 Messages** ou **FT2 Messages** apparaît en dessous, avec la même disposition que le panneau FT8.

**Remarque :** FT4 et FT8 utilisent des formats spectraux différents et ne sont pas interchangeables. Assurez-vous d'être sur une fréquence FT4 lorsque vous utilisez ce décodeur.

## 4. JS8

**De quoi il s'agit :** JS8 (le mode utilisé par JS8Call) reprend le moteur signal faible de FT8 et en fait un mode de conversation au clavier. Là où FT8 envoie des échanges figés de 13 caractères, JS8 transmet du texte libre par petits morceaux et recolle les émissions successives en phrases complètes — tout ce qui dépasse une douzaine de caractères arrive donc sur plusieurs cycles de 15 secondes. Sa sensibilité est proche de celle de FT8, ce qui le rend utilisable là où la phonie et la CW ne passent plus.

## Fréquences recommandées (USB)

Bande	Fréquence
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

## Vitesses

JS8 propose cinq vitesses. Toutes les stations d'une conversation doivent utiliser la même. **Normal** est la vitesse d'appel et concentre la quasi-totalité du trafic — commencez par elle.

Vitesse	Cycle	Largeur	Usage
Slow	30 s	25 Hz	Trajets très faibles ; la plus sensible
Normal	15 s	50 Hz	La vitesse d'appel habituelle
Fast	10 s	80 Hz	Échanges plus rapides, demande plus de signal
Turbo	6 s	160 Hz	Signaux locaux forts
Ultra	4 s	250 Hz	Expérimentale, rarement rencontrée

## Mise en route

1. Accordez-vous sur une fréquence JS8 et passez en **USB**.
2. Activez le décodeur et choisissez **JS8** dans la liste, ou appuyez simplement sur le bouton **JS8**.
3. Le panneau **JS8 Decoder** apparaît en dessous. Laissez **Speed** sur *Normal*, sauf si vous savez que la station visée en utilise une autre.

## Lire le résultat

Le panneau comporte deux listes.

**Les messages en cours d'arrivée** s'affichent en haut, en vert, avec un curseur clignotant. Un message JS8 peut demander quatre cycles — une minute entière en Normal — c'est donc là que l'on voit la phrase se construire. Rien d'anormal s'il y reste un moment.

**Les messages terminés** remplissent la liste principale, une ligne chacun :

Colonne	Signification
Mode	Toujours JS8
Hz	Fréquence audio du signal dans la bande passante
dB	Rapport signal/bruit dans une largeur de référence de 2500 Hz
Message	Le texte décodé ; <b>les indicatifs sont affichés en vert</b>

Messages typiques :

- SV1BTL KM17: HB — un heartbeat : la station signale qu'elle est à l'écoute, avec son locator.
- KN4CRD: K0OG SNR -05 — un message dirigé : KN4CRD indique à K0OG qu'il le reçoit à -5 dB.
- MP 100W 8M/BALUN JN58KH AUGSBURG, MARTIN — texte libre, ici une description de station arrivée en quatre cycles.

Une ligne **grisée et en italique** a expiré avant l'arrivée de sa dernière trame. Le texte est authentique, mais peut être tronqué.

## Sync offset

Le curseur **Sync offset** règle le délai après la limite du cycle UTC avant le début de la capture, ce qui compense le retard de la chaîne audio. Laissez **Auto** coché : le décodeur mesure la synchronisation des signaux reçus et s'ajuste tout seul.

## Repères de spectre

La petite bande de spectre du panneau trace un trait vertical jaune à la fréquence de chaque signal décodé au cycle précédent, ce qui permet de lire la colonne Hz directement sur l'affichage.

## Voir ses propres spots

Si le récepteur envoie des spots JS8 à PSK Reporter (voir l'onglet Spot Reporting du panneau d'administration), le bouton **JS8 map** du cadre **Note:** ouvre la carte de PSK Reporter filtrée sur l'indicatif de ce récepteur et le mode JS8 — exactement ce que font **FT8 map** et **FT4 map** pour ces modes.

**Astuce :** JS8 est bien plus calme que FT8. Un après-midi, une émission toutes les quelques minutes est normale, tout comme de longues périodes sans rien. Laissez-lui cinq à dix minutes avant de conclure à un problème. 20m (14.078) et 40m (7.078) sont les endroits habituels.

**Astuce :** Comme FT8, JS8 exige une synchronisation stricte et utilise l'horloge de votre ordinateur. Si l'heure système dérive de plus d'une ou deux secondes, rien ne sera décodé. Gardez-la synchronisée par NTP.

## 5. CW — code Morse

**De quoi s'agit-il :** le décodeur CW écoute les signaux en code Morse (onde entretenue) et les convertit en texte en temps réel. Il suit automatiquement la fréquence du signal et s'adapte à la vitesse de manipulation de l'opérateur.

### Fréquences recommandées

La CW est active sur toutes les bandes amateurs, généralement dans la partie basse de chaque bande. Endroits courants :

Bande	Segment
40 m	7.000-7.040 MHz
20 m	14.000-14.070 MHz
15 m	21.000-21.080 MHz

## Configuration

1. Accordez-vous sur un signal CW en utilisant le mode de démodulation **CW** ou **CW-L** selon le cas.

2. Activez le décodeur et sélectionnez **CW** dans le menu déroulant. Ou appuyez simplement sur le bouton **CW**.
3. Le panneau **CW Decoder** apparaît en dessous.

## Lecture des résultats

- L'en-tête du panneau indique la fréquence détectée du signal en Hz (p. ex.  $\approx 700$  Hz) et la vitesse de manipulation estimée en mots par minute (p. ex.  $\cdot 22$  WPM).
- Si aucun signal n'est détecté, l'en-tête affiche **scanning...**
- Sur un canal vide rien ne s'affiche, et ni la vitesse ni la tonalité ne sont indiquées. Le décodeur ne commence à écrire que lorsqu'il existe une nette différence entre manipulateur appuyé et relâché, ce que le bruit seul ne produit pas, alors qu'un signal encore lisible vers  $-10$  dB en a largement.
- Le texte décodé défile en caractères ambrés à chasse fixe. Le curseur clignotant (█) marque l'endroit où le texte s'écrit actuellement.
- Cliquez sur **Clear** pour effacer la mémoire tampon de sortie.

**Astuces :** - Centrez votre bande passante sur la tonalité CW. Le décodeur fonctionne au mieux lorsque le signal CW se situe entre environ 400 et 900 Hz dans le spectre audio. - Les vitesses d'environ 8 à 40 mots/min sont lues, y compris la manipulation Farnsworth (caractères envoyés vite avec des pauses plus longues entre eux) : le verrouillage de vitesse tient et les lettres ne sont pas coupées en mots séparés. Un Morse très manuel (irrégulier) peut encore réduire la précision. - Le décodeur donne ses meilleurs résultats sur un signal unique et propre. Un QRM fort provenant de signaux voisins sur la même bande peut le perturber.

## 6. QRSS Grabber

**De quoi il s'agit :** le QRSS est de la CW émise si lentement qu'un point dure des secondes au lieu de millisecondes. À cette vitesse le signal n'occupe qu'une fraction de hertz, si bien qu'une analyse suffisamment étroite fait ressortir la trace depuis 20 à 30 dB sous le plancher de bruit. Il n'y a rien à lire à l'oreille : le QRSS **se regarde**, il ne s'écoute pas. Le grabber calcule sa propre FFT très longue sur l'audio du récepteur et trace l'affichage classique des grabbers : fréquence sur l'axe vertical, temps défilant de gauche à droite, la colonne la plus récente toujours au bord droit.

L'essentiel de ce que vous verrez provient de balises MEPT (Manned Experimental Propagation Transmitter) — des émetteurs de très faible puissance, souvent quelques centaines de milliwatts dans un fil, qui s'identifient en permanence en morse très lent ou par motifs FSK.

### Où chercher

Les balises QRSS occupent des fenêtres étroites de 100 à 200 Hz près du bas de chaque bande. Inutile de les retenir : le grabber propose une liste **Band** des fenêtres ci-dessous, et en choisir une accorde le récepteur dessus et le prépare pour ce mode (voir *Ouvrir le grabber*).

Bande	Centre de la fenêtre	Remarque
30m	10 140,00 kHz	La fenêtre QRSS principale — de loin la plus fréquentée, de jour comme de nuit
40m	7 039,90 kHz	
40m	7 000,85 kHz	Fenêtre Knights, 7 000,8–7 000,9 kHz
20m	14 096,90 kHz	
80m	3 569,90 kHz	
80m	3 568,60 kHz	Ancienne fenêtre
80m	3 500,85 kHz	Fenêtre Knights, 3 500,8–3 500,9 kHz
160m	1 837,90 kHz	
160m	1 843,30 kHz	Ancienne fenêtre
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	Ancienne fenêtre
10m	28 322,00 kHz	Variante
6m	50 294,30 kHz	

Plusieurs bandes figurent deux fois parce que deux conventions coexistent réellement. Les fenêtres modernes se placent 200 Hz sous la fréquence WSPR de la bande ; les anciennes fenêtres *Knights* sont ailleurs, et bien plus bas sur 40 et 80 m. Si une bande est calme sur l'une, essayez l'autre.

Ce sont des conventions, non des règlements, et certaines sont régionales — les valeurs du 10 m varient particulièrement. Prenez la liste comme point de départ et suivez l'usage local.

## Ouvrir le grabber

Le grabber QRSS ne figure **pas** dans le menu des décodeurs. Il dispose de sa propre section **QRSS**, à côté des commandes du spectrogramme et des décodeurs.

1. Cliquez sur **Show**. Le canevas du grabber apparaît et ses commandes se déploient à côté du bouton.
2. Choisissez une fenêtre dans la liste **Band** et appuyez sur **Tune**. Cela fait d'un coup tout ce que le mode réclame : accord sur la fenêtre, passage du récepteur en **CW** et bande passante juste assez large pour la tranche observée. L'affichage indique alors la véritable fréquence QRSS, et les traces tombent sur la ligne centrale de l'écran.
3. Réglez **Speed** sur la durée du point de la balise.
4. **Centre** et **Span** agissent comme avant ; modifier l'un ou l'autre remodèle la bande passante, si bien que l'affichage et le récepteur restent accordés entre eux.
5. Cliquez sur **Hide** pour arrêter. Le mode normal de la bande revient.

Vous pouvez toujours procéder à la main : accordez en **USB** environ 1 kHz sous la fenêtre pour que les traces tombent vers 800 Hz d'audio. La liste **Band** ne fait que vous épargner le calcul.

Tant qu'une fenêtre de la liste est accordée, le grabber tient le récepteur comme le fait un décodeur : la CW et sa bande passante survivent aux changements de fréquence, et la fenêtre suit l'accord, de sorte que la trace reste sur la ligne centrale pendant que vous parcourez la bande. Il rend le récepteur lorsque vous masquez le grabber, cliquez sur un bouton de mode ou démarrez un décodeur.

## Vitesses

Réglage	Durée du point	Résolution	Fenêtre d'analyse	Nouvelle colonne toutes les
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 réglage plus rapide que la balise gaspille de la sensibilité : la trace ressort fine et bruitée. Un réglage plus lent que la balise fond points et traits successifs en une seule barre. Le panneau démarre sur **QRSS 6**, un point de départ raisonnable devant un signal inconnu : plus sensible que QRSS 3, tout en se rafraîchissant plus vite et en tolérant mieux la dérive que QRSS 10. Dès que la forme de la manipulation devient lisible, passez à la vitesse réellement émise par la balise — chaque cran vers un réglage plus rapide coûte 3 dB de sensibilité.

## Commandes

Commande	Effet
<b>Band</b>	Les fenêtres QRSS du tableau ci-dessus. <b>Tune</b> accorde sur celle choisie, en CW avec une bande passante adaptée.
<b>Speed</b>	Fixe la longueur de la transformée, c'est-à-dire le compromis entre résolution en fréquence et en temps (tableau ci-dessus).
<b>Centre</b>	Fréquence audio au milieu de l'affichage, 100–3000 Hz.
<b>Span</b>	Hauteur de la tranche affichée : 20, 50, 100 ou 200 Hz, 100 Hz au démarrage — la largeur d'une sous-bande QRSS. Une tranche plus étroite étale chaque trace sur davantage de pixels. Les tranches larges font entrer dans le panneau plus de bins qu'il n'a de lignes de pixels ; chaque ligne affiche alors le bin le plus fort qu'elle couvre, si bien que rien ne peut se cacher entre les lignes.
<b>Gain</b>	De –10 à +40 dB. Décale la palette par rapport au plancher de bruit mesuré ; augmentez-le pour éclaircir les traces faibles.
<b>Color</b>	Rainbow, Green ou Grayscale.
<b>Clear</b>	Efface le canevas et réinitialise la référence de bruit.

## Lire l'affichage

Les graduations de fréquence courent le long du bord gauche, la fréquence la plus haute en haut. Sous le canevas, une ligne d'état indique la vitesse choisie, la résolution réellement utilisée (par ex. 0.183 Hz/bin · 5.5 s window) et la cadence de colonnes (par ex. 2.7 s/column).

La palette est référencée au bruit **courant**, mesuré en continu sur les parties vides de chaque colonne, et non à un niveau absolu. Changer de volume, d'AGC, d'antenne ou de bande n'oblige donc pas à retoucher le Gain : l'affichage conserve un contraste constant par rapport au bruit de bande du moment. Comme pour les décodeurs, l'audio est prélevé avant l'AGC, la réduction de bruit et le mute ; vous pouvez donc couper le son du récepteur et continuer à regarder.

Ce que signifient les formes :

- Une **porteuse stable** trace une ligne horizontale droite.
- Le **morse lent** trace cette ligne comme une suite de segments courts et longs — points et traits qui se lisent de gauche à droite.
- Les **traces courbes ou dérivantes** viennent d'un oscillateur de balise non stabilisé qui chauffe ou refroidit. C'est normal, et c'est souvent cette signature de dérive qui permet à un observateur régulier de reconnaître une station.
- Les **stries verticales** traversant toute la tranche sont des parasites atmosphériques ou du bruit local, pas du signal.

**Conseils :** - Soyez patient. En QRSS 30 une seule colonne prend 5,5 secondes : un indicatif complet peut demander dix minutes ou plus pour traverser l'écran. Laissez tourner. - Accorder depuis la liste **Band** resserre déjà la bande passante autour de la fenêtre. Si vous avez accordé à la main, faites-le vous-même : cela ne change rien à ce que la transformée résout, mais empêche des voisins puissants d'agir sur l'AGC du récepteur. - Le grabber est indépendant des décodeurs : vous pouvez le laisser tourner pendant qu'un décodeur travaille sur autre chose. - Un span large à vitesse lente est la combinaison la plus lourde ; toute la transformée s'exécute dans le navigateur, alors préférez un span de 50 Hz sur une machine modeste.

## 7. WSPR

**De quoi s'agit-il :** WSPR (Weak Signal Propagation Reporter, prononcé « whisper ») est un mode balise à signaux ultra-faibles qui cartographie les chemins de propagation HF dans le monde entier. Chaque transmission dure environ 110 secondes et tient dans un créneau de 200 Hz de large. Le décodeur attend un créneau complet de 2 minutes aligné sur l'UTC avant de décoder.

### Fréquences recommandées (USB, affichée)

Bande	Fréquence affichée
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

### Configuration

1. Accordez-vous sur une fréquence affichée WSPR ci-dessus et réglez le mode sur **USB**.
2. Le signal WSPR occupe la plage audio 1400-1600 Hz. Sélectionner **WSPR** fait le reste pour vous : le récepteur bascule en USB et la bande passante se resserre sur 1350-1650 Hz, ce qui couvre toute la plage explorée par le décodeur. Aucun autre ajustement n'est nécessaire.
3. Activez le décodeur et sélectionnez **WSPR** dans le menu déroulant. Ou appuyez simplement sur le bouton **WSPR**.
4. Le panneau **WSPR-2 Decoder** apparaît en dessous.

## Lecture des résultats

Le panneau affiche une barre de progression pour le créneau de 2 minutes en cours :

- **Barre cyan qui se remplit** — collecte des données du signal (de 0 à 116 s dans le créneau).
- **Barre ambrée qui pulse** — décodage en cours (les ~4 dernières secondes du créneau).
- **Barre vide** — attente de la prochaine minute UTC paire.

Chaque spot décodé avec succès est affiché dans un tableau avec les colonnes suivantes :

Colonne	Signification
UTC	Heure du spot (minute paire)
Callsign	La station qui a émis
Grid	Locator Maidenhead de l'émetteur
Power	Puissance émise en dBm
Freq	Fréquence audio exacte (Hz) dans la bande passante WSPR
SNR	Rapport signal/bruit en dB

Cliquez sur **Clear** pour effacer la liste des spots.

**Astuce :** le décodage WSPR exige une heure système très précise (à  $\pm 1$  seconde de l'UTC). Le premier créneau après l'activation du décodeur commencera à la prochaine minute UTC paire — une courte attente est normale.

## 8. FAX HF / WEFAX

**De quoi s'agit-il :** le radiofax HF (également appelé WEFAX) est utilisé par les garde-côtes et les services météorologiques du monde entier pour diffuser des cartes météo, des cartes d'état de la mer et des analyses de surface en ondes courtes. Le décodeur reconstitue l'image ligne par ligne au fur et à mesure de sa réception.

### Configuration

1. Activez le décodeur et sélectionnez **HF FAX / WEFAX** dans le menu déroulant. Ou appuyez simplement sur le bouton **FAX**.
2. Le panneau **HF FAX / WEFAX Receiver** apparaît en dessous.
3. **Sélectionnez une station** dans la liste déroulante Station. Plus de 20 stations sont disponibles, couvrant l'Europe, l'Asie, l'Océanie et les Amériques (p. ex. DDH3/DDK3 Allemagne, SVJ4/GR Grèce, JMH Japon, NMG États-Unis La Nouvelle-Orléans). CBM Punta Arenas (Chili) en fait aussi partie. Les horaires suivent le programme mondial de radiofax de la NOAA (édition du 7 mars 2025) ; Charleville (VMC, Australie) inclut ses émissions supplémentaires sur 11030 kHz uniquement.
4. Si la station émet sur plusieurs fréquences, sélectionnez la fréquence souhaitée dans le sous-menu **Frequency**.
5. Cliquez sur ► **Tune** pour accorder automatiquement la cascade sur cette station.
6. Le mode est automatiquement forcé sur **USB**.

## Horaires de diffusion

Lorsqu'une station est sélectionnée, un tableau de compte à rebours **Next Transmissions** apparaît, indiquant les 4 prochaines diffusions programmées en UTC, avec un décompte en direct :

- Normal (gris/vert) — la transmission est à venir.
- **Ambre** — la transmission commence dans moins de 3 minutes ; armez le décodeur maintenant.
- **Rouge** ● — la transmission commence dans moins de 30 secondes ; la réception est imminente.

## Paramètres

La plupart des stations utilisent les valeurs par défaut standard (marquées ★). Ne les modifiez que si vous savez que la station emploie des réglages non standard.

Paramètre	Défaut	Description
LPM	120 ★	Lignes par minute — la vitesse de rotation du tambour
IOC	576 ★	Indice de coopération — détermine le nombre de pixels par ligne
Shift	800 Hz ★	Écart de fréquence entre les tonalités noir et blanc

## Commandes

- ⇄ **Auto-align** — activé par défaut. Se synchronise automatiquement sur le signal de phasage au début de chaque image. Ne le désactivez que si vous rencontrez des problèmes d'alignement sur un signal connu comme bon.
- ⬆ **Invert** — inverse le noir et le blanc. À utiliser si l'image apparaît en négatif (zones blanches là où le noir devrait être).
- ↺ **Refresh** — efface le canevas et réinitialise le décodeur. À utiliser entre deux transmissions ou si l'image se déchire ou dérive.
- ⬇ **Save PNG** — enregistre le canevas actuel sous forme de fichier PNG sur votre ordinateur.

## Indicateurs d'état

En bas de l'image, deux indicateurs de tonalité montrent :

- **300 Hz phasing** — s'allume en cyan lorsque la tonalité de phasage de début d'image est détectée.
- **450 Hz stop** — s'allume en rouge lorsque la tonalité d'arrêt de fin d'image est détectée.

Les deux signaux sont reconnus à leur rythme sur toute la largeur d'une ligne, que les vrais signaux de début et de fin remplissent toujours et que le contenu d'une carte ne remplit pratiquement jamais, si bien que les zones hachurées d'une carte ne sont pas prises pour un signal de début. Ils sont reconnus jusqu'à environ 0 dB de SNR, et sur des signaux très faibles l'image est calée sur la moyenne de plusieurs lignes de phasage, de sorte que la carte commence quand même au bord gauche.

**Remarque :** l'image défile vers le haut — la ligne reçue la plus récente apparaît toujours en bas du canevas. Si vous voyez **[PHASING]** dans l'en-tête, le décodeur s'est verrouillé sur un nouveau début d'image.

## 9. NAVTEX

**De quoi s'agit-il :** NAVTEX est le système international de diffusion maritime des informations de sécurité côtière — avertissements de navigation, prévisions météorologiques et avis de recherche et de

sauvetage. Il utilise le FSK à 100 bauds (SITOR-B avec FEC) et est reçu sur des canaux dédiés dans le monde entier.

## Canaux disponibles

Canal	Fréquence	Usage
International	518 kHz	En anglais, international
National	490 kHz	Diffusions en langue nationale
HF (×5)	4209.5 / 6314 / 8416.5 / 12579 / 16806.5 kHz	NAVTEX HF longue portée

## Configuration

1. Activez le décodeur et sélectionnez **NAVTEX** dans le menu déroulant. Ou appuyez simplement sur le bouton **NAVTEX**. Le récepteur passe automatiquement en **USB**.
2. Le panneau **NAVTEX Receiver** apparaît.
3. Sélectionnez le canal souhaité dans la liste **Station** (p. ex. `International – 518 kHz`).
4. Cliquez sur ⇒ **Tune & Set IF** pour accorder automatiquement la cascade et resserrer la bande passante sur la bonne fenêtre audio. Le mode est réglé automatiquement sur **USB**. La fréquence affichée est placée 500 Hz sous le centre du canal, de sorte que le signal NAVTEX apparaît à 500 Hz en audio.

## Horaires de diffusion

Le tableau des horaires liste toutes les stations connues sur le canal sélectionné avec :

- leur lettre d'identification UIT et le drapeau du pays
- l'heure de la prochaine diffusion en UTC
- un décompte en direct jusqu'à la prochaine transmission
- **Ambre** à moins de 2 minutes / **Rouge** ● à moins de 30 secondes — armez le décodeur maintenant

## Lecture des résultats

Le texte décodé apparaît en caractères sarrcelle à chasse fixe. Les limites des messages sont clairement marquées :

```
— ZCZC MA12 —
... message content ...
— NNNN —
```

ZCZC marque le début d'un message. Les quatre caractères qui suivent identifient la station (M), le sujet (A = avertissements de navigation) et le numéro d'ordre (12). La bannière attend l'en-tête complet, elle affiche donc ZCZC MA12 et non ZCZC ????. NNNN marque la fin.

Si vous vous calez après le début d'un message, ou si son en-tête se perd dans le bruit, le texte s'affiche quand même, sous une bannière — **(header missed)** —. Elle ne s'ouvre que lorsque plusieurs caractères consécutifs arrivent avec accord des deux émissions NAVTEX, ce que le bruit ne produit pratiquement jamais. Elle se ferme à NNNN comme d'habitude, avec — **(signal lost)** — si le signal disparaît avant, ou quand un nouveau message commence.

Cliquez sur **Clear** pour effacer la mémoire tampon des messages.

**Remarque :** NAVTEX fonctionne sur des fréquences MF et LF (518/490 kHz). La portée de réception est généralement de 200 à 400 milles marins depuis l'émetteur. Les canaux HF (4-17 MHz) offrent une portée bien supérieure.

## 10. FSK / RTTY — y compris PSK31 et Olivia

**De quoi s'agit-il :** un décodeur polyvalent pour les modes texte à bande étroite, avec cinq variantes de fonctionnement accessibles depuis un seul menu. Trois d'entre elles sont du véritable FSK (Frequency-Shift Keying) : FSK maritime (SITOR), RTTY météo et RTTY amateur. Les deux autres ne sont pas du FSK du tout mais partagent la même fenêtre : **PSK31**, qui est une modulation par déplacement de phase, et **Olivia**, qui est un FSK multitonale avec correction d'erreurs. Chaque variante est fournie avec un préréglage adapté à ses paramètres standard.

### Variantes et préréglages

Variante	Centre	Shift	Bauds	Trame	Codage
FSK maritime / SITOR	500 Hz	170 Hz	100	7N1	CCIR-476
RTTY météo	1000 Hz	450 Hz	50	5N1.5	ITA2
RTTY amateur	1000 Hz	170 Hz	45.45	5N1.5	ITA2
PSK31 (BPSK)	1000 Hz	—	31.25	—	Varicode
Olivia (MFSK)	1000 Hz	—	voir Mode	—	7 bits + FEC

Le panneau s'adapte à la variante choisie. **Shift**, **Baud**, **Framing**, **Encoding**, **Invert mark / space** et **Auto shift detect** sont masqués pour PSK31 et Olivia, car aucun de ces deux modes ne possède de paire de tonalités mark/space ni de trame de type UART. À leur place, **Center audio** devient un champ numérique libre (la porteuse peut se trouver n'importe où dans la bande passante), et Olivia ajoute un sélecteur **Mode** et un curseur **Squelch**.

### Réglage

1. Activez le décodeur et sélectionnez **FSK / RTTY** dans le menu. Ou appuyez simplement sur le bouton **RTTY**.
2. Le panneau du décodeur apparaît. Son titre suit la variante — *FSK / RTTY Decoder*, *PSK31 Decoder* ou *Olivia Decoder*.
3. Sélectionnez la **Variante**. Les paramètres se mettent à jour automatiquement.
4. Pour Olivia, réglez **Mode** (tonalités / largeur de bande) pour correspondre à l'émission — voir les notes sur Olivia ci-dessous.
5. Utilisez le menu **Known frequency** pour choisir une fréquence courante de la variante sélectionnée, puis cliquez sur **Tune** pour vous y rendre.
6. Affinez l'accord jusqu'à ce que le texte décodé devienne stable et lisible.

### Fréquences connues par variante

#### FSK maritime / SITOR

- 518,0 kHz — NAVTEX international
- 490,0 kHz — NAVTEX national
- 4209,5 / 6314,0 / 8416,5 / 12579,0 / 16806,5 / 22376,0 kHz — SITOR HF

**RTTY météo**

- 4583,0 / 7646,0 / 10100,8 / 11039,0 / 14467,3 kHz — DWD (service météorologique allemand)

**RTTY amateur**

- 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)

**Paramètres**

Paramètre	S'applique à	Description
Center audio (Hz)	toutes	La fréquence audio du milieu entre mark et space ; pour PSK31 la porteuse, pour Olivia le centre du bloc de tonalités. Un menu déroulant pour les variantes FSK, un champ libre pour PSK31 et Olivia
Shift (Hz)	FSK seulement	Écart de fréquence entre les tonalités mark et space
Baud	FSK seulement	Débit de symboles
Framing	FSK seulement	Bits de données, parité, bits d'arrêt (par ex. 7N1 = 7 données, sans parité, 1 arrêt)
Encoding	FSK seulement	Jeu de caractères (CCIR-476, ITA2/Baudot ou ASCII)
Invert mark / space	FSK seulement	Permute les tonalités mark et space
Auto shift detect	FSK seulement	Tente de mesurer automatiquement le shift à partir du signal reçu
Mode (tonalités / Hz)	Olivia seulement	Nombre de tonalités et largeur de bande — doit correspondre exactement à l'émission
Squelch (FEC S/N)	Olivia seulement	Force que doit atteindre la correction d'erreurs avant que du texte soit affiché
Squelch (SNR)	RTTY amateur et météo	Empêche le bruit de s'afficher en lettres aléatoires. Démarre à -5 dB pour le RTTY amateur et à -8 dB pour le RTTY météo ; tout à gauche ( <b>off</b> ) il est désactivé. Le chiffre de SNR passe au gris tant que le squelch est fermé

**Mesures du signal**

La barre d'état affiche des mesures en direct, et les champs changent selon la variante :

Variante	Champs affichés
Variante FSK	<b>Mark / Space</b> (fréquences de tonalité mesurées), <b>SNR</b> , <b>Lock</b> , <b>Timing</b>
PSK31	<b>Carrier</b> (Hz, après correction automatique de fréquence), <b>IMD</b> (dB), <b>SNR</b> , <b>Lock</b> , <b>Timing</b>
Olivia	<b>Centre</b> (Hz), <b>Mode</b> , <b>S/N</b> , <b>FEC</b> (%), <b>Sync</b>

Timing/Sync affiche LOCKED/SYNCED dès que le décodeur suit le signal, et SEARCH tant qu'il le cherche encore.

Le **SNR** des variantes FSK et de PSK31 est mesuré sur le signal lui-même et donné dans une bande de 3 kHz, si bien que les deux se lisent sur la même échelle : à environ 1 dB près de +20 dB à -5 dB. Une fréquence vide affiche environ -12 à -15 dB, et en PSK31 0 dB est déjà un signal fort. Le **S/N** d'Olivia est autre chose : l'indice de qualité de la correction d'erreurs.

Le RTTY amateur (45,45 bauds, 170 Hz) est lu sans erreur jusqu'à environ 0 dB, avec environ 1,5 % de caractères faux à -3 dB et 20 % à -6 dB.

## Commandes supplémentaires

- **⇒ Set IF Band-Pass** — resserre la bande passante du récepteur autour du signal. La largeur suit la variante : 300 Hz pour le RTTY amateur (assez étroit pour éliminer une station plus forte à 250 Hz tout en lisant un signal décalé de 40 Hz), 594 Hz pour le RTTY météo, mark/space plus une marge pour le FSK maritime, environ  $\pm 100$  Hz pour PSK31 et toute la largeur du bloc de tonalités plus une marge pour Olivia. La bande passante suit le décodeur quand **Auto-tune** ou **Auto shift detect** déplace les tonalités, et quand vous changez le réglage **Baud**.
- **🔊 Auto-tune Center** — recherche automatique du signal. Pour les variantes FSK elle cherche une paire équilibrée de tonalités, pour PSK31 elle trouve la porteuse, et pour Olivia le bloc le plus fort de la largeur de bande sélectionnée. **Auto shift detect** ne change le shift que lorsqu'un vrai signal à deux tonalités est présent, il ne saute donc plus sur le bruit.

## Notes sur PSK31

PSK31 est le mode clavier à clavier le plus répandu en HF. Il ne fait que 62 Hz de large, si bien que plusieurs QSO cohabitent côte à côte dans les quelques centaines de hertz autour de la fréquence de rassemblement ; l'accord consiste à choisir une trace dans le groupe visible sur la cascade.

- Le décodeur corrige lui-même son erreur d'accord sur environ  **$\pm 25$  Hz**, il suffit donc de s'approcher. **Carrier**, dans la ligne de mesures, indique où il s'est réellement stabilisé.
- **IMD** mesure la qualité de l'émetteur, pas la réception : un signal propre affiche environ -20 dB ou mieux. Une mauvaise valeur signifie que la station d'en face surmodule, non que vous êtes mal accordé.
- Le texte apparaît caractère par caractère sans correction d'erreurs ; un signal faible se dégrade donc en lettres fausses occasionnelles plutôt que de s'interrompre.

## Notes sur Olivia

Olivia échange la vitesse contre la robustesse. Il est bien plus lent que PSK31 mais décode des signaux inaudibles à l'oreille, ce qui le rend populaire pour les signaux faibles et les liaisons à grande distance.

- **Le réglage Mode doit correspondre exactement à l'émission.** Une combinaison tonalités/largeur de bande erronée ne décode absolument rien — pas du texte brouillé, mais le silence. Le panneau s'ouvre sur **8 / 250**, la configuration étroite laissée en veille sur les fréquences d'appel ; **16 / 500** et **32 / 1000** sont les deux autres d'usage courant, et **16 / 1000** est également proposé.
- Olivia n'envoie pas de préambule, le décodeur doit donc chercher la synchronisation. **Comptez quelques secondes** après l'accord avant que du texte apparaisse. Le champ **Sync** affiche SEARCH jusqu'au verrouillage.
- La correction d'erreurs travaille par blocs : le texte arrive **par rafales plutôt qu'en flux continu**, avec un retard de plusieurs blocs entre l'émission et l'affichage.
- **Squelch (FEC S/N)** définit le degré de confiance que doit atteindre la correction d'erreurs avant d'afficher. La valeur par défaut de 4,0 tient le bruit à l'écart ; 3,0 est le plancher, en dessous duquel

le bruit aléatoire commence à imprimer des caractères isolés. Un bon signal affiche 8-9 sur l'indicateur **FEC**, il reste donc une large marge pour relever le squelch sur une bande encombrée.

**Note sur le mode :** le décodeur prend le contrôle du mode de démodulation et de la bande passante FI tant qu'il est actif. Les deux sont rétablis automatiquement à sa désactivation. Les cinq variantes utilisent l'**USB**.

**Note sur la polarité (variantes FSK uniquement) :** pour le RTTY météo, il faut généralement cocher **Invert mark / space**. Pour le FSK maritime (type SITOR/NAVTEX) et le RTTY amateur, laissez la case décochée — le RTTY amateur émet le mark sur la radiofréquence la plus haute, et l'USB le conserve comme la tonalité audio la plus haute, ce qui correspond précisément au cas décoché. Si le texte décodé est brouillé, la première chose à essayer est de basculer cette case. Elle est masquée pour PSK31 et Olivia, qui n'ont pas de paire mark/space.

**Note sur lettres/chiffres (variantes FSK uniquement) :** le code Baudot gère les lettres et les chiffres dans deux états distincts, et le bruit peut faire basculer le décodeur dans le mauvais — ce qui brouille tous les caractères suivants, pas seulement celui qui a été altéré. Le décodeur revient donc aux lettres à chaque espace ; c'est la pratique courante et cela répare un basculement corrompu en un ou deux mots au lieu d'une ligne entière. En contrepartie, les groupes de chiffres séparés par des espaces exigent que l'émetteur répète le passage aux chiffres après chaque espace, ce que font normalement les émetteurs.

## 11. SSTV

**De quoi s'agit-il :** la télévision à balayage lent transmet des images fixes sur un canal phonie SSB ordinaire, une ligne à la fois, sous forme d'une tonalité modulée en fréquence entre 1500 Hz (noir) et 2300 Hz (blanc). Une image complète prend entre 36 secondes et 4 minutes 30 selon le mode.

### Fréquences recommandées (USB)

Bande	Fréquence	Mode	Remarques
20 m	14.230 MHz	USB	La principale fréquence d'appel SSTV internationale — de loin la plus active
20 m	14.233 MHz	USB	Secondaire, utilisée quand 14.230 est occupée
15 m	21.340 MHz	USB	
10 m	28.680 MHz	USB	Active pendant les ouvertures de bande
40 m	7.171 MHz	LSB	
80 m	3.845 MHz	LSB	Régionale, en soirée

**La bande latérale est choisie pour vous.** Le démarrage du décodeur sélectionne la **LSB en dessous de 10 MHz** et l'**USB au-dessus**, selon l'usage amateur habituel. Sur la mauvaise bande latérale, la correspondance des tonalités est inversée et l'image ne sera pas décodée. Si vous rencontrez une station qui ignore la convention, changez simplement la bande latérale à la main — le décodeur reste actif, efface l'image et repart à zéro avec le nouveau réglage.

### Configuration

1. Activez le décodeur et sélectionnez **SSTV** dans le menu déroulant. Ou appuyez simplement sur le bouton **SSTV**. Le récepteur change automatiquement de bande latérale — USB au-dessus de 10 MHz, LSB en dessous.

2. Accordez-vous de sorte que les tonalités de l'image tombent au milieu de la bande passante. Un signal correctement accordé a ses impulsions de synchronisation à 1200 Hz et le contenu de l'image entre 1500 et 2300 Hz.
3. Laissez **Mode** sur **Auto**, sauf si vous savez déjà ce qui est transmis. L'image se construit ligne par ligne au fur et à mesure de la réception.

## Modes

Réglage	Image	Durée
<b>Auto</b>	Détecté automatiquement	—
Martin M1 / M2	320×256 couleur	114 s / 58 s
Scottie S1 / S2	320×256 couleur	110 s / 71 s
Scottie DX	320×256 couleur	269 s
Robot 36 / 72	320×240 couleur	36 s / 72 s

**Auto** fonctionne de deux façons : il lit l'**en-tête VIS** — le code numérique du mode envoyé dans les 300 premières millisecondes d'une transmission — et, si l'en-tête a été manqué (vous vous êtes accordé trop tard, ou il a été perdu dans le QSB), il identifie le mode d'après la synchronisation des impulsions. Sélectionner un mode précis force ce mode, mais le décodeur vérifie tout de même la synchronisation avant de dessiner quoi que ce soit : un mauvais choix ne produit donc aucune image plutôt que du bruit.

## Lecture des résultats

La ligne d'état sous les commandes indique ce que fait le décodeur :

État	Signification
Waiting for VIS / AUTO lock	À l'écoute ; rien d'identifié pour l'instant
Martin M1? verifying sync...	Un candidat a été trouvé et est confirmé sur les lignes suivantes
Martin M1 lock (VIS)	Verrouillé d'après l'en-tête VIS
Martin M1 lock (AUTO)	Verrouillé d'après la synchronisation
Martin M1 lock (MANUAL)	Démarré par le bouton ● <b>Force</b>
Candidate rejected (no sync)	Le candidat n'a pas été confirmé — normal sur du bruit
— sync lost, resetting	Le signal a disparu en cours d'image
Frame complete	L'image complète a été reçue

Le mode détecté apparaît aussi en vert à côté du titre **SSTV**, et le compteur de lignes indique la progression.

## Commandes

Bouton	Action
► <b>Start</b>	Arme le décodeur. Il ne dessine pas d'image de lui-même — un verrouillage doit encore provenir de l'en-tête VIS ou de la détection de synchronisation.
■ <b>Stop</b>	Arrête le décodage.
↺ <b>Reset</b>	Efface le canevas et réarme sur place. À utiliser entre deux images ou après un réaccord.
● <b>Force</b>	Commence à dessiner <b>immédiatement</b> dans le mode sélectionné dans la liste, en sautant entièrement le VIS et la détection de synchronisation. Disponible uniquement quand Mode n'est pas sur <b>Auto</b> .
<b>Save</b>	Enregistre l'image actuelle au format PNG.

**Quand utiliser Force.** Si vous voyez une image dans la cascade mais que le décodeur ne parvient pas à s'y verrouiller — mode inhabituel, signal trop faible ou trop déformé pour les détecteurs, ou en-tête manqué — sélectionnez le mode à la main et appuyez sur ● **Force**. L'image est calée au moment du clic, et le badge de mode affiche **MANUAL** pour que vous puissiez la distinguer d'un verrouillage VIS ou **AUTO**. Le décodeur se recalcule tout de même sur une véritable impulsion de synchronisation s'il en trouve une : un clic légèrement en avance ou en retard est donc corrigé.

Comme Force contourne toutes les vérifications de sécurité, il peindra volontiers du bruit si vous l'actionnez sur un canal vide, et il ne s'arrêtera pas tout seul — appuyez sur ■ **Stop** ou ↺ **Reset**.

## Remarques

- **Le bruit ne déclenche pas le décodeur.** Les parasites atmosphériques, les étincelles et le bruit du secteur suffisaient autrefois à déclencher un décodage. Chaque étape de détection vérifie désormais que la tonalité est bien une tonalité, et confirme le candidat sur les lignes suivantes avant de dessiner le moindre pixel. Attendez-vous à ce que le décodeur reste silencieux sur une bande vide.
- **L'image dérive en diagonale si vous êtes décalé en fréquence.** La SSTV ne pardonne pas les erreurs d'accord. Si les lignes penchent, ajustez la fréquence par petits pas et laissez l'image suivante démarrer.
- L'équilibre des couleurs et la netteté sont fixes ; il n'y a aucun réglage à faire.

## 12. Conseils généraux

**Decoder: ON doit être activé en premier.** Le menu déroulant est désactivé (grisé) tant que vous n'avez pas cliqué sur le bouton Decoder pour l'activer.

**Un décodeur à la fois.** Sélectionner un nouveau décodeur dans le menu arrête automatiquement celui qui était actif et annule les changements de mode ou de bande passante qu'il avait effectués.

**Le mode est géré pour vous.** Le FAX HF et NAVTEX basculent le récepteur en USB dès que vous les sélectionnez, la SSTV choisit la bande latérale d'après la bande (LSB sous 10 MHz, USB au-dessus), et FAX, NAVTEX et FSK ajustent aussi la bande passante lorsque vous cliquez sur leur bouton Tune. FT8, FT4, FT2, JS8 et WSPR font de même dès que vous les sélectionnez — depuis la rangée de boutons ou la liste déroulante : le récepteur bascule en USB et prend leur propre bande passante — toute la sous-bande de 3 kHz pour la famille FT8, 1350-1650 Hz pour WSPR. Lorsque vous arrêtez le décodeur, le mode par défaut de la bande est restauré.

**Activer un décodeur n'interrompt plus l'audio.** Les décodeurs s'exécutent sur leurs propres threads : il n'y a donc ni blanc, ni clic, ni coupure lorsque l'un démarre, s'arrête ou est remplacé par un autre.

**La précision de l'horloge système compte.** FT8, FT4 et WSPR sont critiques en temps. Ils décodent dans des fenêtres fixes alignées sur l'UTC. Si l'horloge de votre ordinateur est décalée de plus de 1 à 2 secondes, les taux de décodage chuteront nettement. Utilisez un client NTP pour garder votre horloge précise.

**Les filtres de bruit n'atteignent pas les décodeurs.** NR, NB, NS et AN sont des aides à l'écoute, pour vos oreilles seulement. Tous les décodeurs — FT8, FT4/FT2, CW, WSPR, FAX, NAVTEX, FSK/RTTY/PSK31/Olivia, SSTV et le grabber QRSS — prennent l'audio *avant* ces filtres : réglez-les donc comme cela sonne le mieux, sans vous soucier de la qualité de décodage. Pour la même raison, les décodeurs continuent de tourner pendant que le récepteur est coupé ou silencé : vous pouvez éteindre le haut-parleur et laisser un décodeur, ou une capture QRSS nocturne, poursuivre. La seule chose qui suive réellement ce que vous entendez est le spectrogramme audio, qui est fait pour montrer l'audio filtré.

**La qualité du signal prime sur sa force.** La plupart de ces décodeurs sont conçus pour les signaux faibles. Une bande plus calme avec moins de bruit est souvent plus productive qu'un signal fort noyé dans les interférences. Utilisez la cascade et les commandes de bande passante pour repérer et éviter le QRM avant d'activer un décodeur.

**Utilisez les boutons Refresh ou Clear sans hésiter.** Les images FAX dérivent si la fréquence affichée est légèrement décalée, et les décodeurs de texte accumulent des caractères parasites. Un nouveau départ après des ajustements d'accord produit souvent un résultat beaucoup plus propre.

## Report automatique des spots et graphiques du serveur

Les décodages FT8, FT4 et WSPR peuvent aussi être téléversés par le serveur lui-même — FT8/FT4 vers PSK Reporter, WSPR vers WSPRnet — par un démon autorun que le sysop démarre depuis le panneau d'administration. Il est indépendant des décodeurs de votre navigateur : il continue de tourner qu'il y ait des auditeurs ou non, et rien de ce que vous décidez dans le navigateur n'est rapporté.

Le sysop le suit à l'aide de deux compteurs, faciles à confondre : les tuiles par décodeur comptent les spots téléversés depuis le dernier démarrage du démon, tandis que le nombre à côté de chaque case bande/mode est le total depuis toujours de cet emplacement et survit aux redémarrages. Le même panneau propose une page **Graphiques** traçant la fréquence du processeur, la charge, la température et les utilisateurs connectés sur les 15 dernières minutes à 24 heures. Les deux sont décrits dans le [guide du panneau d'administration](#).

## CHAPITRE 9

# RADE README

RADE\_README.md

## Voix numérique RADE v1 pour PhantomSDR-Plus

**RADE** (Radio AutoencoDEr) est le mode phare de voix numérique HF de FreeDV. Il utilise un hybride apprentissage automatique / DSP (le vocodeur neuronal FARGAN) pour offrir une parole de haute qualité en HF avec des SNR descendant jusqu'à  $-2$  dB, dans seulement 1500 Hz de largeur de bande RF — plus étroit qu'un signal BLU.

Ce document couvre l'intégration complète de la réception RADE v1 dans PhantomSDR-Plus, mise en œuvre sous la forme d'un sidecar Python (`rade_helper.py`) qui relie le navigateur à la chaîne de décodage `radae_rxe.py` + `lpcnet_demo`.

### L'installation — la voie courte

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

C'est toute l'installation. Le script installe les paquets système avec le gestionnaire de paquets présent sur la machine (`apt`, `pacman`, `dnf` ou `zypper`), se rabat sur `pip` pour ce qu'une distribution ne fournit pas, met à jour `websockets` si la version de la distribution est antérieure à la 11.0 exigée par le sidecar, clone et compile `radae`, vérifie les poids du modèle, reconstruit le frontend et démarre le sidecar. Sous Ubuntu 22.04, il passe la main à `install_rade_ubuntu22.sh`. `./install.sh` propose de l'exécuter dans le cadre de l'installation normale : sur une machine neuve, RADE est donc déjà là.

**Ce document explique comment RADE fonctionne et comment le faire tourner — ce n'est pas la procédure d'installation.** Si vous devez installer à la main, sur un système que le script ne couvre pas ou pour réparer une étape, suivez le guide lié ci-dessous : les étapes y sont écrites, et nulle part ailleurs. Tout ce qui suit ici — l'architecture, les fichiers corrigés, le port, l'usage de RADE dans le navigateur, les variables d'environnement, le dépannage et la mesure de la simultanéité — vaut pour toute installation, quelle qu'en soit la manière.

- **Installation manuelle sous Linux** - la voie manuelle, étape par étape, pour Ubuntu, Debian, Fedora, Arch et Raspberry Pi OS

### Table des matières

1. L'installation — la voie courte
2. Fonctionnement
3. Vue d'ensemble de l'architecture
4. RADEL contre RADEU
5. Prérequis
6. Installation à la main
7. Déployer les fichiers corrigés

8. Contrôle du sidecar
9. Le port 8074
10. Utiliser RADE dans le navigateur
11. Vérification et débogage
12. Variables d'environnement
13. Fichiers modifiés
14. Résumé du flux du signal
15. Dépannage
16. Mesurer la simultanéité sur votre propre matériel
17. Mettre RADE v1 à jour

## Fonctionnement

RADE v1 ne peut pas s'exécuter dans le navigateur — il exige PyTorch et le vocodeur neuronal FARGAN, trop volumineux pour WASM. La solution est un processus sidecar Python (`rade_helper.py`) qui s'exécute sur le même serveur que PhantomSDR-Plus.

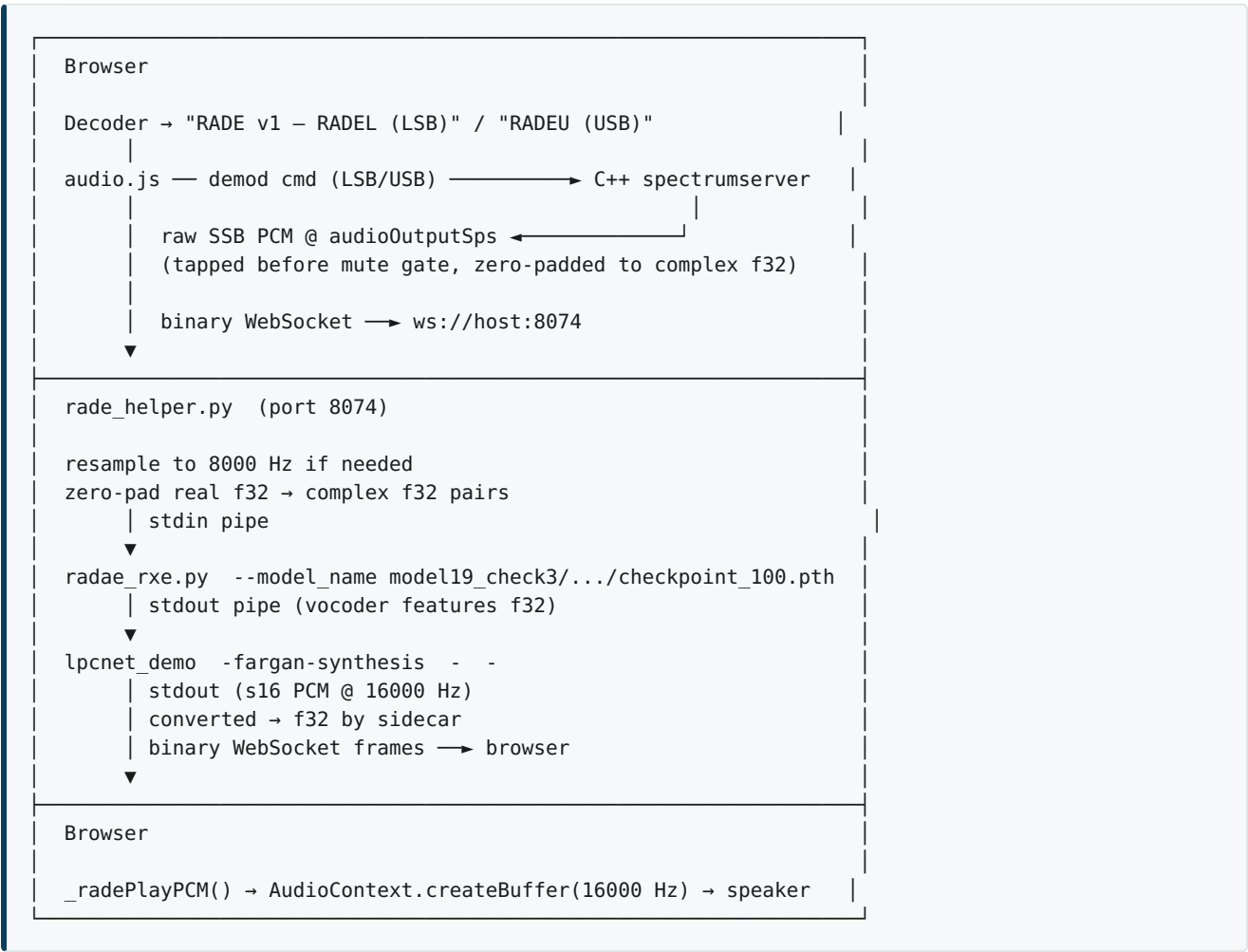
**Important :** `freedv_rx` du dépôt `codec2` ne prend **pas** en charge RADEV1. La chaîne de décodage RADE réside entièrement dans le dépôt distinct `radae` de David Rowe (VK5DGR). N'essayez pas d'utiliser `freedv_rx` de `codec2` pour RADE.

Lorsque vous sélectionnez RADE dans le navigateur :

1. Le frontend règle la démodulation sous-jacente sur USB ou LSB — le serveur C++ démodule le signal BLU normalement.
2. Le PCM démodulé brut est prélevé dans `audio.js` **avant** toute coupure ou porte de squelch — `radae_rxe.py` a besoin d'une entrée continue pour maintenir la synchronisation de trame.
3. Chaque bloc PCM est complété par des zéros pour passer de f32 réel à f32 complexe (réel + 0.0 imaginaire) — c'est ce que `radae_rxe.py` attend sur son entrée standard.
4. Le sidecar transmet ces échantillons à `radae_rxe.py`, qui produit des paramètres de vocodeur.
5. `lpcnet_demo -fargan-synthesis` convertit ces paramètres en parole s16 à 16000 Hz.
6. Le sidecar convertit s16 → f32 et renvoie le résultat au navigateur sous forme de trames WebSocket binaires.
7. Le navigateur restitue la parole décodée via l'API Web Audio à 16000 Hz.

Le serveur C++ (`spectrumserver.cpp`) n'est absolument pas modifié.

## Vue d'ensemble de l'architecture



## RADEL contre RADEU

RADE v1 est toujours transmis en BLU. Par convention :

Mode	Bande latérale	À utiliser sur les bandes
RADEL	LSB	160 m, 80 m, 40 m ( $\leq 10$ MHz)
RADEU	USB	20 m, 17 m, 15 m, 12 m, 10 m ( $> 10$ MHz)

## Prérequis

Élément requis	Version	Remarques
PhantomSDR-Plus	quelconque	avec le frontend Vite/Svelte
Linux	Ubuntu 24.04+ / Debian Bookworm+	testé
Python	3.8+	pour <code>rade_helper.py</code> et <code>radae_rxe.py</code>
dépôt radae	dernier master	fournit <code>radae_rxe.py</code> et <code>lpcnet_demo</code>
cmake	3.10+	pour compiler <code>lpcnet_demo</code> depuis <code>radae</code>
PyTorch	2.0+	requis par <code>radae_rxe.py</code>
Node.js	16+	pour <code>npm run build</code>
websockets (Python)	10-16+	<code>pip3 install websockets</code> — toutes versions prises en charge
matplotlib	quelconque	requis par <code>radae_rxe.py</code> à l'importation
numpy	1.23+	requis ; permet aussi le rééchantillonnage
scipy	quelconque	facultatif, permet un rééchantillonnage précis

## Installation à la main

`./install_rade.sh` réalise toute l'installation. Si vous voulez la faire vous-même — sur un système que le script ne couvre pas, ou pour réparer une étape — la procédure se trouve dans le [Guide d'installation manuelle de RADE v1](#), et là uniquement. Il couvre Ubuntu, Debian, Fedora, Arch et Raspberry Pi OS en une seule passe :

Étape	Description
Étape 1	Paquets système, et la vérification de Node.js 22+
Étape 2	Paquets Python, avec l'option PEP 668 et la roue torch CPU seule
Étape 3	Cloner et compiler le dépôt radae, vérifier <code>lpcnet_demo</code> et les poids du modèle
Étape 4	Vérifier la chaîne de décodage hors ligne, avant de la relier au navigateur
Étape 5	Compiler le frontend de PhantomSDR-Plus
Étape 6	Contrôle du sidecar
Étape 7	Ouvrir le port 8074
Étapes 8 à 10	Démarrer le serveur, vérifier RADE, l'utiliser dans le navigateur

La suite de ce document suppose cela fait.

## Déployer les fichiers corrigés

Copiez les fichiers suivants du jeu de correctifs dans votre dépôt PhantomSDR-Plus.

## Nouveau fichier — à placer à la racine du dépôt, à côté du script de démarrage que vous utilisez :

```
rade_helper.py
```

## Fichiers frontend corrigés — à placer dans `frontend/src/` :

```
audio.js
App.svelte
```

## Ce que font les correctifs

**audio.js** — 5 modifications :

- Constructeur : 5 nouveaux champs d'état RADE (`decodeRADE`, `_radeSideband`, `_radeSocket`, `_radeCallback`, `_radeReady`, `_radeNextTime`)
- Sauvegarde du PCM avant amplification : `pcmArrayPreBoost` enregistré avant le gain FLAC de 300x, afin que RADE reçoive l'amplitude d'origine (l'amplification saturerait `radac_rxe.py`)
- `playAudio()` : prélèvement du PCM pour RADE en utilisant l'audio avant amplification, en amont de la porte de coupure/squelch
- `playAudio()` : garde `if (this.decodeRADE) return` — supprime la lecture BLU brute pendant que la parole décodée par RADE est jouée
- Nouvelles méthodes : `setRADEDecoding()`, `setRADECallback()`, `_radePlayPCM()` avec lecture planifiée sans blanc grâce à l'horloge `_radeNextTime` (lecture à **16000 Hz** — le débit de sortie de `lpcnet_demo`)

**Chaque variante Svelte** — 6 modifications :

- `demodulationDefaults` : `RADEL {type:'LSB', offsets:[2200,-700]}`, `RADEU {type:'USB', offsets:[-700,2200]}` — la bande passante commence à 700 Hz de la porteuse, sur 1500 Hz de large
- Variables d'état : `radeEnabled`, `radeConnected`, `radeSynced`, `radeSnr`, `_radeDeactivate()`
- `_radeDeactivate()` : arrête le décodeur et **restaure le mode par défaut de la bande** d'après `bands-config.js` — comportement identique à la désactivation de FAX, NAVTEX, FSK
- `_deactivateAll()` : ligne de nettoyage RADE
- `activateSelectedDecoder()` : branches `radel` et `radeu`
- Menu déroulant des décodeurs : deux nouvelles entrées `<option>`
- Panneau d'état : point de connexion, état de synchronisation/SNR, bannière d'erreur

## Scripts de démarrage

Les lanceurs du récepteur (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) démarrent eux-mêmes le sidecar dès que le serveur tourne, et `stop-websdr.sh` l'arrête avec le serveur. `rade.sh` reste destiné à faire tourner le sidecar de façon autonome — voir [Contrôle du sidecar](#).

## Contrôle du sidecar

**Vous n'avez normalement pas besoin d'un script de contrôle distinct pour RADE.** Les lanceurs du récepteur (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) démarrent eux-mêmes le sidecar dès que le serveur est confirmé en marche, le relançant s'il s'arrête et signalent son état dans leur résumé de fin — ✓ `RADE sidecar running`, ou une note `•` expliquant pourquoi il n'a pas été activé (non installé, `RADE_ENABLED=0`, ou pas de

python3). `./stop-websdr.sh` l'arrête en même temps que le serveur. Pour faire tourner le serveur sans RADE, démarrez-le avec `RADE_ENABLED=0`.

`rade.sh` reste fourni pour un usage autonome — faire tourner le sidecar sans le lanceur PhantomSDR-Plus, ou le tester seul :

```
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
```

L'activité du sidecar est journalisée dans `~/PhantomSDR-Plus/rade.log` :

```
tail -f ~/PhantomSDR-Plus/rade.log
```

**Ne mélangez pas les deux pendant que le serveur tourne.** Si le chien de garde d'un lanceur est actif, il ramène le sidecar en cinq secondes environ, si bien que `./rade.sh stop` semble avoir échoué. Pour arrêter RADE alors que le serveur tourne, démarrez le serveur avec `RADE_ENABLED=0`, ou arrêtez tout avec `./stop-websdr.sh`.

Si vous lancez `rade.sh` de façon autonome, **utilisez toujours `./rade.sh stop`** — un simple `kill -f rade_helper.py` ne bat pas son propre chien de garde, qui relance le processus en 3 secondes.

## Le port 8074

Le sidecar écoute sur le port TCP **8074** et le navigateur s'y connecte directement : le port doit donc être joignable de l'extérieur. Son ouverture — `ufw`, `firewalld`, `iptables`, la règle NAT du routeur, et le proxy Nginx quand un FAI bloque purement et simplement le port — constitue l'[étape 7 du guide manuel](#).

Deux choses méritent d'être répétées ici, car ce sont elles qui coûtent le plus de temps :

```
Is the sidecar listening?
ss -tlnp | grep 8074
```

**Attention au NAT hairpin** : un test avec `curl` depuis le serveur vers son propre nom public renvoie souvent `Connection refused` alors même que le port est ouvert — beaucoup de routeurs ne renvoient pas le trafic. Testez toujours depuis une machine extérieure, ou utilisez <https://portchecker.co>.

## Utiliser RADE dans le navigateur

1. Ouvrez votre interface web PhantomSDR-Plus
2. Trouvez les stations actives sur [qso.freedv.org](https://qso.freedv.org)
3. Accordez-vous sur la fréquence affichée de la station
4. Dans **Decoder Options**, sélectionnez :
5. **RADE v1 — RADEL (LSB)** pour 40 m / 80 m / 160 m
6. **RADE v1 — RADEU (USB)** pour 20 m / 17 m / 15 m / 12 m / 10 m
7. Cliquez sur **Decoder: ON**

## Les boutons RADEL / RADEU (une pression)

Les étapes 4 et 5 peuvent être remplacées par une seule pression. Une paire de boutons **RADEL / RADEU** est disponible à trois endroits :

- à côté du titre **Modes selector** du panneau principal ;
- dans la fenêtre **Modes** ;
- dans la fenêtre **Bands**.

Une pression sélectionne le décodeur, met le décodeur sur ON, ferme la fenêtre depuis laquelle vous avez appuyé et amène le panneau RADE à l'écran. Le bouton devient bleu tant que RADE tourne.

**Appuyez de nouveau sur le même bouton pour arrêter RADE** — le panneau se ferme, et le mode ainsi que la bande passante reviennent à leur valeur par défaut comme décrit plus bas. Les boutons, le menu déroulant **Decoder Options** et le bouton ON/OFF partagent le même état.

## États de l'indicateur du panneau

Indicateur	Signification
Rouge — « Connecting to sidecar... »	Port 8074 injoignable ou sidecar arrêté
Jaune — « Searching for signal... »	Sidecar connecté, aucune trame RADE encore détectée
Vert — « Synced · SNR x.x dB »	Décodage en cours — la parole est audible

## Quand vous désactivez le décodeur

Le mode et la bande passante reviennent automatiquement à la valeur par défaut correcte pour la fréquence courante, telle que définie dans `bands-config.js` — comme pour FAX, NAVTEX, FSK.

## Vérification et débogage

### Le sidecar tourne-t-il ?

```
ps aux | grep rade_helper | grep -v grep
ss -tlnp | grep 8074
```

### Suivre l'activité en direct

```
tail -f ~/PhantomSDR-Plus/rade.log
```

Lorsqu'un navigateur se connecte :

```
[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)
```

## Test rapide du 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
```

## Console du navigateur (F12)

```
[RADE] ► ENABLED LSB @ 12000 Hz → helper ws://localhost:8074
```

Bon signe — le WebSocket est ouvert. La valeur en Hz correspond à l'`audioOutputSps` de votre serveur (typiquement 8000–12000 Hz).

```
[RADE] sidecar socket error
```

Le port 8074 est injoignable — vérifiez le pare-feu et la règle NAT du routeur.

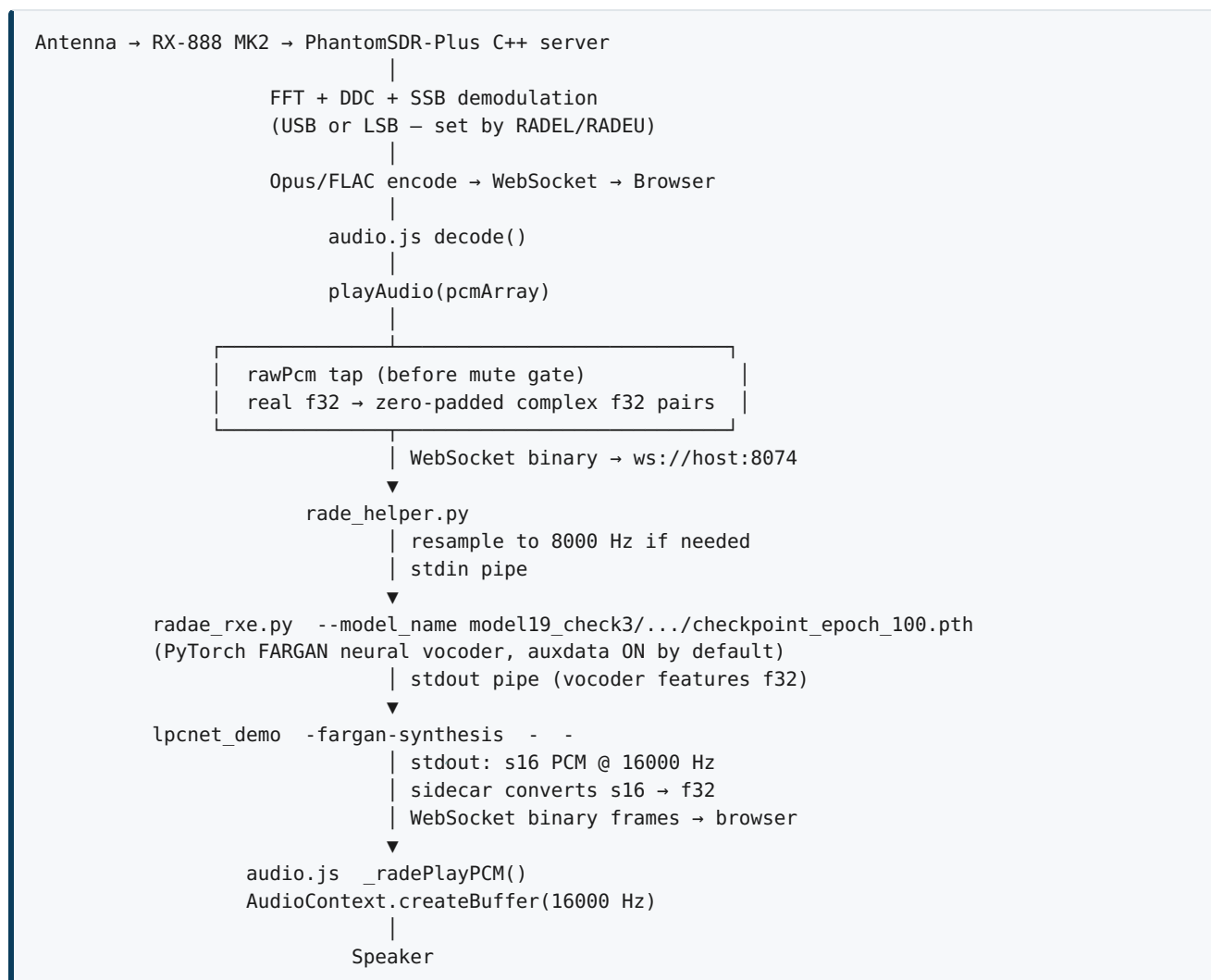
## Variables d'environnement

Variable	Défaut	Description
<code>RADE_HELPER_PORT</code>	8074	Port TCP sur lequel le sidecar écoute
<code>RADE_HELPER_HOST</code>	0.0.0.0	Adresse de liaison (127.0.0.1 derrière un proxy)
<code>RADAE_DIR</code>	<code>~/radae</code>	Racine du dépôt radae
<code>RADE_MODEL</code>	<code>RADAE_DIR/model19_check3/checkpoints/checkpoint_epoch_100.pth</code>	Poids du modèle
<code>LPCNET_DEMO</code>	<code>RADAE_DIR/build/src/lpcnet_demo</code>	Binaire lpcnet_demo
<code>RADE_AUXDATA</code>	1	Mettre à 0 pour passer <code>--noauxdata</code> à <code>radae_rxe.py</code>
<code>RADE_TORCH_THREADS</code>	1	Threads PyTorch/OpenBLAS par instance de <code>radae_rxe.py</code> — limite l'usage CPU
<code>RADE_PIN_CORES</code>	1	Épingle chaque chaîne de décodage sur son propre jeu de cœurs ; 0 désactive l'épingleage
<code>RADE_CORES_PER_CLIENT</code>	2	Cœurs alloués par client lorsque l'épingleage est activé

## Fichiers modifiés

Fichier	Type	Remarques
rade_helper.py	Nouveau	Sidecar Python : serveur WebSocket + chaîne de décodage à deux processus
frontend/src/audio.js	Modifié	4 correctifs
frontend/src/App.svelte	Modifié	6 correctifs
rade.sh	Nouveau	Script de contrôle du sidecar RADE pour un usage autonome (start/stop/restart/status)
start-rx888mk2.sh	Modifié	démarre, surveille et signale le sidecar ; idem pour start-airspyhf.sh, start-rtl.sh, start-rspla.sh, start-fobos-hf.sh, start-fobos.sh, start-hackrf.sh
stop-websdr.sh	Modifié	arrête le sidecar en même temps que le serveur
spectrumserver.cpp	Inchangé	Aucune modification C++ nécessaire

## Résumé du flux du signal



## Dépannage

### Bannière rouge — « 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")
```

### Port 8074 fermé sur portchecker.co malgré la règle iptables

Le FAI filtre peut-être le port, ou la règle NAT du routeur est absente. Options :

1. Essayer un autre port : `RADE_HELPER_PORT=8080 python3 rade_helper.py`
2. Passer par votre port public existant via un proxy Nginx (voir l'[étape 7 du guide manuel](#))

#### `radax_rxe.py: error: unrecognized arguments: model_path`

Le chemin du modèle doit être passé avec `--model_name`, pas en argument positionnel :

```
WRONG – positional argument
python3 radax_rxe.py model19_check3/checkpoints/checkpoint_epoch_100.pth

CORRECT – named argument
python3 radax_rxe.py --model_name model19_check3/checkpoints/checkpoint_epoch_100.pth
```

#### `ModuleNotFoundError: No module named 'matplotlib'`

```
pip3 install matplotlib
```

`radax_rxe.py` importe `matplotlib` sans condition en tête de fichier.

#### `ModuleNotFoundError: No module named 'torch'`

```
pip3 install torch
```

#### `lpcnet_demo: No such file or directory`

La compilation de `radax` n'a pas abouti. Recompilez :

```
cd ~/radax/build
cmake .. && make -j$(nproc)
ls src/lpcnet_demo # should exist now
```

### Erreur `size mismatch` dans `inference.sh`

`model19_check3` exige `--auxdata` lors de l'encodage avec `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
```

À noter : lors du **décodage** avec `radae_rxe.py`, `--auxdata` est la valeur par défaut — ne le passez pas.

## Indicateur jaune — « Searching for signal » — ne se synchronise jamais

- Confirmez la bonne bande latérale : RADEL pour  $\leq 10$  MHz, RADEU pour  $> 10$  MHz
- Consultez [qso.freedv.org](https://qso.freedv.org) pour vérifier qu'une station émet actuellement
- RADE v1 utilise 30 porteuses dans 1500 Hz de bande — il apparaît comme un groupe compact dans la cascade
- Comptez jusqu'à 1,5 seconde pour l'acquisition

## underrun!!! d'aplay pendant le test de la chaîne (étape 3)

Attendu uniquement lors des tests hors ligne sur fichier. `radae_rxe.py` traite plus lentement que les entrées/sorties du fichier, ce qui affame le tampon audio. Cela ne se produit pas en réception en direct, car le navigateur fournit l'audio au rythme temps réel.

## ConnectionClosedError: received 1011 (internal error)

Le sidecar a accepté la connexion WebSocket mais a planté en interne avant de répondre. La cause est une version incompatible de `rade_helper.py` — une version plus ancienne tentait de passer un `StreamReader` `asyncio` comme entrée standard d'un sous-processus, ce qui échoue silencieusement et ferme avec l'erreur 1011.

**Correctif :** remplacez `rade_helper.py` par la version actuelle du jeu de correctifs. La version actuelle utilise `os.pipe()` pour le tube interprocessus et est compatible avec websockets 10.x à 16.x+.

## Plusieurs utilisateurs simultanés

Chaque connexion de navigateur lance sa propre paire indépendante `radae_rxe.py` + `lpcnet_demo`, de sorte que chaque utilisateur peut s'accorder librement sur une fréquence différente.

Par défaut, `radae_rxe.py` utilise **tous les cœurs CPU disponibles** pour les opérations matricielles PyTorch, ce qui provoque une charge d'environ 900 % par instance. `rade_helper.py` limite cela avec deux optimisations :

1. `OMP_NUM_THREADS=1` (et les équivalents MKL/OpenBLAS) — limite PyTorch à 1 thread
2. Fonctions de conversion audio vectorisées avec numpy — éliminent le coût des boucles Python

Résultat : chaque instance consomme environ **8 à 10 %** d'un cœur, suffisant pour un décodage RADE en temps réel. Si vous entendez des coupures audio, passez à 2 threads :

```
RADE_TORCH_THREADS=2 ./start-rx888mk2.sh # or your receiver's launcher
standalone: RADE_TORCH_THREADS=2 ./rade.sh restart
```

Utilisateurs RADE simultanés	CPU approximatif
1	~9 %
5	~45 %
10	~90 %
20	~180 %

Sur un i5-12450H (12 threads) avec spectrumserver à ~42 % et rx888\_stream à ~11 %, vous disposez d'environ **1000 % de marge** — assez pour **20 utilisateurs RADE simultanés ou plus** avant que le CPU ne devienne un souci.

**Le tableau ci-dessus suppose que le coût CPU croît linéairement avec le nombre d'utilisateurs. La mesure dit le contraire.** Une exécution de `rade_loadtest.py` sur ce même i5-12450H a montré un coût CPU par auditeur bien inférieur à ce tableau jusqu'à ~24 auditeurs, puis une hausse marquée au-delà, la machine étant arrêtée par la **température du package à 32 auditeurs** — ni par le CPU, ni par la bande passante. Considérez ces chiffres comme un repère grossier pour de petits nombres seulement et mesurez votre propre machine : voir [Mesurer la simultanéité sur votre propre matériel](#). Notez que le test de charge alimente la chaîne avec du bruit plutôt qu'un vrai signal RADE : la charge CPU absolue avec du trafic réel peut donc différer ; c'est la *forme* de la courbe qui est fiable.

## Mesurer la simultanéité sur votre propre matériel

Les chiffres ci-dessus proviennent d'une machine précise. `rade_loadtest.py` (à la racine du dépôt) mesure la même chose sur **la vôtre** — il monte en charge des auditeurs RADE synthétiques face au sidecar par paliers et indique le dernier nombre d'auditeurs que votre machine a soutenu avant que quelque chose ne cède.

Après chaque palier, il laisse la machine se stabiliser, puis échantillonne la température du package, le CPU total, la RAM disponible, le swap et la RSS combinée de tous les processus `radax_rxe.py` et `lpcnet_demo`. Il s'arrête au premier **coude** — quelle que soit la limite franchie en premier — et affiche le dernier nombre stable.

### Prérequis

```
sudo apt install python3-websockets python3-psutil
```

Exécutez-le **sur l'hôte SDR** (il lit les capteurs locaux et la mémoire des processus), avec spectrumserver et le sidecar RADE déjà en marche :

```
cd ~/PhantomSDR-Plus
python3 rade_loadtest.py
```

Ctrl-C ferme proprement toutes les connexions à tout moment.

### Avant la première exécution

Ouvrez le fichier et vérifiez les deux blocs en tête :

- **CONFIG** — `WS_URL` vaut par défaut `ws://127.0.0.1:8074` ; changez-le si vous avez déplacé le sidecar de son port par défaut. `RADE_SPS` (12000) et `RADE_SIDEHAND` (USB) doivent correspondre à ce que rapporte votre frontend.

- `handshake_messages()` — l'unique trame `init` envoyée doit correspondre à l'init RADE de `frontend/src/audio.js`. Si cette poignée de main a dérivé, chaque auditeur échoue à se connecter et le test se termine immédiatement par des échecs de connexion.

## Seuils du coude

Ajustez-les dans `CONFIG` à votre convenance — ils sont volontairement prudents :

Seuil	Défaut	Arrête lorsque
<code>TEMP_KNEE_C</code>	85.0	La température du package atteint ce plafond
<code>MIN_AVAIL_MB</code>	800	La RAM disponible tombe en dessous
<code>SWAP_GROWTH_MB</code>	50	Le swap dépasse la référence prise au démarrage
<code>FAIL_LIMIT</code>	3	Autant d'auditeurs échouent à se connecter en un palier
<code>MAX_LISTENERS</code>	60	Arrêt ferme même si aucun coude n'est atteint

La forme de la montée est contrôlée par `STEP` (auditeurs ajoutés par palier, 2 par défaut), `SETTLE_S` (25 s de stabilisation avant échantillonnage — c'est ce qui laisse la chaleur s'accumuler) et `SAMPLE_S` (fenêtre de moyennage CPU de 5 s). Une exécution complète jusqu'au plafond par défaut de 60 auditeurs prend donc environ 15 minutes.

## Lire les résultats

Chaque palier affiche une ligne et ajoute une ligne à `rade_loadtest.csv` :

Colonne	Signification
<code>listeners</code>	Auditeurs synthétiques connectés à ce palier
<code>connect_fails</code>	Auditeurs n'ayant pas réussi à établir ou maintenir une connexion
<code>pkg_c</code>	Température du package, médiane de 5 échantillons (résistante aux pics)
<code>cpu_pct</code>	Pourcentage CPU <b>de tout le système</b> , pas par auditeur
<code>avail_mb</code>	RAM disponible
<code>swap_mb</code>	Swap utilisé
<code>dec_rss_mb</code>	RSS combinée de tous les processus de décodage
<code>dec_procs</code>	Nombre de processus de décodage — attendez-vous à <b>2 par auditeur</b> ( <code>radac_rxe.py</code> + <code>lpcnet_demo</code> )

## Une exécution mesurée

Sur l'i5-12450H mentionné plus haut, une exécution complète s'est terminée ainsi :

```
KNEE at n=32: temp 91.0°C ≥ 85.0
Last stable concurrency: 30 RADE listeners
```

**La chaleur était la contrainte déterminante**, et de loin. Au coude, il restait encore 7004 Mo de RAM disponible — 6,2 Go au-dessus du seuil `MIN_AVAIL_MB` — et zéro échec de connexion à chaque palier.

Deux colonnes méritent une lecture attentive :

**`dec_rss_mb` surestime le coût mémoire réel.** Elle croît de façon très régulière de 295 Mo par auditeur (292, 294, 294 ... 295 sur les 16 paliers), mais la *RAM disponible* ne baisse que d'environ **202 Mo par auditeur**. Les deux processus derrière chaque auditeur partagent des pages de bibliothèques, et la RSS

compte ces pages une fois par processus. En extrapolant la pente de `avail_mb`, le coude mémoire arriverait vers **63 auditeurs** — au-delà de l'arrêt ferme à 60 auditeurs, il ne peut donc jamais se déclencher sur cette machine. Dimensionnez la mémoire d'après `avail_mb`, pas d'après `dec_rss_mb`.

**cpu\_pct est plat, puis ne l'est plus.** Il reste à ~5,3 % de la machine entière jusqu'à 24 auditeurs, puis devient superlinéaire :

Auditeurs	24	26	28	30	32
cpu_pct	6,7 %	18,4 %	30,1 %	49,6 %	65,2 %

Soit une hausse de 10× du CPU alors que le nombre d'auditeurs augmente d'un tiers. L'inflexion se reproduit au même endroit d'une exécution à l'autre : traitez-la comme une propriété de la machine et non comme du bruit — le coût CPU par auditeur avoisine 3,4 % d'un cœur sous l'inflexion et environ 20 % au-dessus. N'extrapolez pas un chiffre de CPU par utilisateur relevé à faible nombre d'auditeurs.

**Le test n'a pas de coude CPU.** Les quatre limites sont la température, la RAM disponible, la croissance du swap et les échecs de connexion — le CPU est enregistré mais ne met jamais fin à l'exécution. Dans l'exécution ci-dessus, le CPU a atteint 65 % et aurait continué de grimper si la température n'avait pas cédé la première. Si un plafond CPU vous importe, surveillez la colonne vous-même ou ajoutez un seuil.

**Ce que cela prouve et ne prouve pas.** Chaque auditeur synthétique diffuse du bruit aléatoire de faible amplitude, et non un vrai signal RADE. C'est suffisant pour faire fonctionner toute la chaîne de décodage et consommer des ressources : les chiffres de ressources sont donc significatifs — mais le test ne dit rien de la *qualité* du décodage ni de la continuité audio en charge. Confirmez cela avec de vrais auditeurs sur un vrai signal.

## Si les cœursaturent avant le coude

RADE épingle fortement un thread par instance, si bien que des cœurs individuels peuvent atteindre ~90 °C alors que le CPU total semble au repos. `rade_helper.py` répartit les chaînes sur les cœurs à tour de rôle pour y remédier. Si une exécution montre des coudes thermiques précoces, élargissez l'allocation de cœurs par client :

```
RADE_CORES_PER_CLIENT=3 ./rade.sh restart
```

Aucune modification de fichier n'est nécessaire — il s'agit d'une variable d'environnement (voir [Variables d'environnement](#)) ; `RADE_PIN_CORES=0` désactive entièrement l'épinglage.

## Mettre RADE v1 à jour

L'intégration à PhantomSDR-Plus (`rade_helper.py`, correctifs du frontend) n'est qu'un pont — toute la logique de décodage RADE réside dans le dépôt `radæ`. Les mises à jour se résument donc presque toujours à un simple `git pull` + recompilation, sans aucune modification de PhantomSDR-Plus lui-même.

## Mise à jour standard (nouveau code, même modèle)

```
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
```

C'est tout. Pas de recompilation du frontend, pas de redémarrage du serveur, aucune modification de fichier.

## Nouveaux poids de modèle uniquement

Si un nouveau point de contrôle est publié (p. ex. `model20`) sans changement de code, pointez le sidecar vers les nouveaux poids via la variable d'environnement — aucune modification de fichier n'est nécessaire :

```
One-off: start with new model
RADE_MODEL=~/.radae/model20/checkpoints/checkpoint_epoch_100.pth ./rade.sh start

Or permanently – add to your shell profile (~/.bashrc):
export RADE_MODEL=~/.radae/model20/checkpoints/checkpoint_epoch_100.pth
```

## Ce que chaque type de changement exige

Ce qui a changé dans radae	Action requise
Nouveaux poids de modèle (nouveau fichier <code>.pth</code> )	Définir la variable <code>RADE_MODEL</code> , <code>./rade.sh restart</code>
Changement de code dans <code>radae_rxe.py</code>	<code>git pull</code> , <code>./rade.sh restart</code>
Code C de <code>lpcnet_demo</code> modifié	<code>git pull</code> , recompiler, <code>./rade.sh restart</code>
<code>radae_rxe.py</code> renommé ou déplacé	Mettre à jour le chemin <code>RADAE_RX</code> dans <code>rade_helper.py</code> (une ligne)
Argument <code>--model_name</code> renommé	Mettre à jour <code>radae_cmd</code> dans <code>rade_helper.py</code> (une ligne)
Nouveau taux d'échantillonnage de sortie (≠ 16000 Hz)	Mettre à jour <code>SPS_OUT</code> dans <code>rade_helper.py</code> + <code>createBuffer()</code> dans <code>audio.js</code>
RADE v2 utilise un binaire différent	Mettre à jour <code>RADAE_RX</code> dans <code>rade_helper.py</code> (une ligne)

## Vérifier que la mise à jour a fonctionné

Après le redémarrage, confirmez que le nouveau code s'exécute :

```
Check sidecar picked up new radae_rxe.py
./rade.sh status

Tail log to see startup lines
tail -20 ~/PhantomSDR-Plus/rade.log
```

Le journal doit afficher le chemin de modèle attendu :

```
[RADE] model : /home/sv1btl/radae/model19_check3/checkpoints/checkpoint_epoch_100.pth
```

## Rester à jour

Abonnez-vous à la page des versions du dépôt radae pour être averti des nouveaux modèles et mises à jour de code :

```
https://github.com/drowe67/radae/releases
```

Consultez le blog FreeDV pour les annonces concernant les nouvelles formes d'onde et modèles RADE :

```
https://freedv.org/blog/
```

*Développé et testé sur PhantomSDR-Plus (fork sv1btl/PhantomSDR-Plus) avec un RX-888 MK2 sous Ubuntu 24.04. Le serveur C++ n'est pas modifié.*

*RADE est développé par David Rowe VK5DGR et l'équipe FreeDV. Voir [freedv.org/radio-autoencoder](https://freedv.org/radio-autoencoder).*

## Désinstallation complète et réinstallation propre

Voici le démontage complet avant de relancer install\_rade.sh :

1. Arrêter le sidecar `cd ~/PhantomSDR-Plus && ./rade.sh stop`
2. Supprimer le dépôt radae et sa compilation `rm -rf ~/radae`
3. Supprimer torch (installé dans le répertoire local de l'utilisateur) `pip3 uninstall -y torch` `rm -rf ~/.local/lib/python3.11/site-packages/torch*`
4. Supprimer les paquets Python d'apt (facultatif — à ignorer s'ils servent à autre chose) `sudo apt-get remove -y python3-numpy python3-scipy python3-matplotlib python3-websockets` `sudo apt-get autoremove -y`
5. Vérifier que tout a disparu `python3 -c "import torch" 2>&1 # should say ModuleNotFoundError` `ls ~/radae 2>&1 # should say No such file or directory`
6. Installation propre `chmod +x ~/PhantomSDR-Plus/install_rade.sh` `~/PhantomSDR-Plus/install_rade.sh`

## CHAPITRE 10

# RADE installation générale (Linux)

RADE\_General\_INSTALL\_MANUAL\_LINUX.md

## RADE v1 — Guide d'installation manuelle

### Tout Linux — Ubuntu / Debian / Fedora / Arch / Raspberry Pi OS · PhantomSDR-Plus

**Prérequis :** les fichiers corrigés (`rade_helper.py`, `rade.sh`, `audio.js`, `App.svelte`) sont déjà en place dans l'arborescence de PhantomSDR-Plus. Ce guide construit tout le reste autour d'eux.

#### IMPORTANT

**Vous n'avez pas besoin de ce guide pour une installation normale.** `./install_rade.sh`, dans le dossier PhantomSDR-Plus, effectue pour vous chacune des étapes ci-dessous — paquets système, modules Python, compilation de `radac`, vérification des poids du modèle et démarrage du sidecar — et fonctionne aussi bien sur les systèmes `apt`, `pacman`, `dnf` que `zypper`. `./install.sh` et les quatre installateurs par distribution proposent de le lancer pendant l'installation normale. Ne suivez ce guide que pour une installation manuelle, pour un système que le script ne couvre pas, ou pour réparer une étape à la main.

**Utilisateurs de Raspberry Pi :** ce guide est aussi le vôtre. Raspberry Pi OS est un Debian : suivez donc partout les blocs Debian et lisez les notes **Raspberry Pi / Bookworm** là où elles apparaissent — elles couvrent les deux points qui diffèrent sur un Pi : la PEP 668 et la roue `torch` CPU seule.

## Étape 1 — Paquets système

Choisissez le bloc correspondant à votre distribution.

### 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
```

## Vérification de la version de Node.js (toutes distributions)

La compilation du frontend de RADE nécessite Node.js 22 ou plus récent :

```
node --version
```

Si la version est inférieure à 22.x, installez-le avec `nvm` (toutes distributions, sans 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
```

**Pas via NodeSource.** `deb.nodesource.com` et `rpm.nodesource.com` renvoient désormais HTTP 403 sur tous les chemins du dépôt : l'ancienne ligne `curl -fsSL https://deb.nodesource.com/setup_NN.x | sudo -E bash -` ne fonctionne donc plus, et sous Debian/Ubuntu elle laisse une source apt qui casse tous les apt update suivants. C'est pour cette raison que les scripts d'installation utilisent `nvm`.

## Étape 2 — Paquets Python

### Ubuntu 24.04 et antérieurs / Fedora / Arch

Un `pip3 install` simple fonctionne sur ces systèmes :

```
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 votre serveur **dispose** bien d'un GPU NVIDIA avec CUDA installé, vous pouvez omettre `--index-url` pour obtenir la version CUDA complète. Cela n'apporte aucun gain de performance pour RADE — `raade_rxe.py` n'utilise PyTorch que pour des opérations matricielles sur CPU.

### Ubuntu 23.04+ / Debian Bookworm+ / Raspberry Pi OS (systèmes PEP 668)

Ces distributions bloquent un `pip3 install` nu. Ajoutez l'option :

```
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 — deux règles qui mordent :**

1. Chaque `pip3 install` exige `--break-system-packages` (application de la PEP 668). Sans cela, l'installation est totalement bloquée.
2. Un simple `pip3 install torch` télécharge la version CUDA (~3 Go). Sur un Pi il n'y a pas de CUDA — utilisez l'index CPU seul pour obtenir la version ARM64 allégée (~150 Mo).

Vous pouvez également utiliser un environnement virtuel pour éviter complètement cette option :

```
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 vous utilisez un venv, faites précéder tous les appels `python3` de ce guide de `source ~/rade-venv/bin/activate`, ou utilisez le chemin complet `~/rade-venv/bin/python3`.

**Vérification**

```
python3 -c "import websockets, matplotlib, torch, numpy, scipy; print('All OK')"
```

Sortie attendue :

```
All OK
```

**Étape 3 — Cloner et compiler le dépôt radae**

Le décodeur RADE se trouve dans un dépôt distinct de codec2. `freedv_rx` de codec2 ne prend **pas** en charge RADE v1.

```
Remove any previous incomplete clone
rm -rf ~/radeae

Clone
git clone https://github.com/drowe67/radae.git ~/radeae
cd ~/radeae

Build
mkdir build && cd build
cmake ..
make -j$(nproc)
```

**Vérifiez que lpcnet\_demo a été compilé**

```
ls -la ~/radeae/build/src/lpcnet_demo
```

Attendu : le binaire est présent et exécutable.

**Vérifiez la présence des poids du modèle**

```
ls ~/radeae/model19_check3/checkpoints/checkpoint_epoch_100.pth
```

Attendu : le fichier `.pth` est listé. Les poids sont fournis dans le dépôt — aucun téléchargement séparé n'est nécessaire.

## Étape 4 — Vérifier la chaîne de décodage

Cette étape confirme que la chaîne hors ligne complète fonctionne avant de la relier au navigateur. À exécuter depuis `~/rade` :

```
cd ~/rade
```

### 4a — Générer un signal de test encodé 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
```

Attendez la fin du traitement. Les dernières lignes affichées doivent être :

```
loss: 0.741 Auxdata BER: 0.012
```

### 4b — Décoder et écouter

```
cat rx.f32 \
| python3 rade_rxe.py \
--model_name model19_check3/checkpoints/checkpoint_epoch_100.pth \
| ./build/src/lpcnet_demo -fargan-synthesis - - \
| aplay -f S16_LE -r 16000
```

Vous devriez entendre une voix. La sortie montre l'acquisition de la synchronisation :

```
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
```

**Les messages `underrun!!!` pendant ce test sont normaux et sans conséquence.** Ils surviennent parce que `radae_rxe.py` traite plus lentement que les entrées/sorties du fichier. Ils n'apparaissent pas en réception en direct — le navigateur fournit l'audio au rythme temps réel.

### 4c — Vérifier que le sidecar démarre correctement

```
python3 ~/PhantomSDR-Plus/rade_helper.py
```

Sortie attendue (aucune ligne WARNING) :

```
[RADE] helper starting on ws://0.0.0.0:8074
[RADE] rade_rx.py : /home/<user>/rade/rade_rxe.py
[RADE] model : /home/<user>/rade/model19_check3/checkpoints/checkpoint_epoch_100.pth
[RADE] lpcnet_demo : /home/<user>/rade/build/src/lpcnet_demo
[RADE] auxdata : ON (default)
[RADE] torch threads : 1 per instance (RADE_TORCH_THREADS to override)
[RADE] architecture : per-connection (each user tunes independently)
[RADE] listening : waiting for PhantomSDR-Plus clients
```

Appuyez sur **Ctrl+C** pour arrêter.

`<user>` dans les chemins ci-dessus désigne le compte avec lequel vous êtes connecté — sur une image standard de Raspberry Pi OS, ils deviennent `/home/pi/radae/...`.

## Étape 5 — Compiler le frontend de PhantomSDR-Plus

```
cd ~/PhantomSDR-Plus/frontend

Only if node_modules is missing:
npm install

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

Surveillez les erreurs de l'analyseur Vite/acorn. Les fichiers corrigés évitent délibérément `?.`, `??` et les `catch {}` nus, afin de respecter la contrainte d'acorn.

## Étape 6 — Contrôle du sidecar

**Vous n'avez normalement pas besoin d'un script de contrôle distinct pour RADE.** Les scripts de démarrage (`start-rx888mk2.sh`, `start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos-hf.sh`, `start-fobos.sh`, `start-hackrf.sh`) démarrent eux-mêmes le sidecar dès que le serveur tourne, le relancent s'il s'arrête et signalent son état ; `stop-websdr.sh` l'arrête en même temps que le serveur. Passez directement à l'étape 7.

**`rade.sh` reste fourni pour un usage autonome — faire tourner le sidecar sans le lanceur PhantomSDR-Plus, ou le tester seul. Ses commandes, son journal et les pièges du chien de garde qui vont avec figurent dans Contrôle du sidecar.**

## Étape 7 — Ouvrir le port 8074

Le sidecar écoute sur le port TCP **8074**. Le navigateur se connecte directement à ce port. Vous devez l'ouvrir manuellement.

### 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 (toutes distributions, permanent)

```
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
```

## Routeur

Ajoutez une règle NAT / redirection de port : **TCP 8074 → IP LAN du serveur : 8074**

## Test depuis l'extérieur

Utilisez **<https://portchecker.co>** et vérifiez le port 8074 sur votre nom d'hôte public. Ne testez **pas** avec `curl` depuis le serveur lui-même — le NAT hairpin donne de faux résultats « Connection refused » même lorsque le port est ouvert.

## Alternative — proxy Nginx (si le port 8074 est bloqué par le FAI)

Ajoutez à l'intérieur de votre bloc `server {}` existant :

```
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
```

Modifiez ensuite `audio.js` dans `setRADEDecoding()` :

```
// Change:
var helperUri = uri || ('ws://' + window.location.hostname + ':8074');
// To:
var helperUri = uri || ('ws://' + window.location.host + '/rade');
```

Recompilez le frontend après cette modification (`./recompile.sh`).

## Étape 8 — Démarrer le serveur

Démarrez PhantomSDR-Plus comme d'habitude. RADE n'est **pas** lancé automatiquement :

```
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
```

Utilisez le lanceur correspondant à votre récepteur. Chacun comporte son propre chien de garde et son journal ; `./stop-websdr.sh` arrête celui qui est en cours.

## Étape 9 — Démarrer RADE

```
cd ~/PhantomSDR-Plus
./rade.sh start
./rade.sh status
tail -f rade.log
```

Journal attendu :

```
[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
```

## Étape 10 — Utiliser RADE dans le navigateur

1. Ouvrez votre interface web PhantomSDR-Plus
2. Trouvez les stations actives sur <https://qso.freedv.org>
3. Accordez-vous sur la fréquence affichée de la station
4. Dans **Decoder Options**, sélectionnez :
5. **RADE v1 — RADEL (LSB)** pour 40 m / 80 m / 160 m ( $\leq 10$  MHz)
6. **RADE v1 — RADEU (USB)** pour 20 m / 17 m / 15 m / 12 m / 10 m ( $> 10$  MHz)
7. Cliquez sur **Decoder: ON**

Les étapes 4 et 5 peuvent être remplacées par une seule pression : les boutons **RADEL / RADEU** se trouvent à côté du titre **Modes selector** ainsi que dans les fenêtres **Modes** et **Bands**. Une pression sélectionne le décodeur, le met sur ON et amène le panneau RADE à l'écran ; appuyez de nouveau pour arrêter RADE. Voir le [manuel RADE](#).

Indicateur	Signification
Rouge — « Connecting to sidecar... »	Port 8074 injoignable ou sidecar arrêté
Jaune — « Searching for signal... »	Sidecar connecté, aucune trame RADE encore détectée (comptez ~1,5 s)
Vert — « Synced · SNR x.x dB »	Décodage en cours — la parole est audible

## Remarque sur le CPU

Chaque utilisateur RADE consomme environ 8 à 10 % d'un cœur (le nombre de threads PyTorch est limité à 1 par le sidecar). Si vous entendez des coupures audio, passez à 2 threads :

```
RADE_TORCH_THREADS=2 ./rade.sh restart
```

Utilisateurs simultanés	CPU approximatif
1	~9 %
5	~45 %
10	~90 %
20	~180 %

Ces chiffres ont été mesurés sur x86\_64. Un Raspberry Pi décode RADE sans problème, mais le coût par utilisateur y est plus élevé et le tableau ci-dessus ne se transpose pas — mesurez votre propre carte avec `top` pendant qu'un utilisateur est synchronisé avant d'annoncer une limite d'utilisateurs.

## Mettre RADE à jour par la suite

```
cd ~/radae
git pull
cmake -S . -B build && make -C build -j$(nproc)

cd ~/PhantomSDR-Plus
./rade.sh restart
```

Aucune recompilation du frontend ni redémarrage du serveur n'est nécessaire, sauf si `rade_helper.py` lui-même a changé.

Testé sur Ubuntu 24.04 (x86\_64) et Raspberry Pi 4 / Raspberry Pi OS Bookworm (ARM64, Python 3.11).  
Fork PhantomSDR-Plus : [sv1bt/PhantomSDR-Plus](https://github.com/sv1bt/PhantomSDR-Plus). RADE développé par David Rowe VK5DGR et l'équipe FreeDV — <https://freedv.org>

## CHAPITRE 11

# Édition des variantes

EDITING\_VARIANTS.md

## PhantomSDR-Plus — Comment modifier les variantes du frontend

(après la fusion des quatre fichiers `App__*_smeter_.svelte` en un seul `App.svelte`)

### 1. Ce que sont les variantes

Deux choix indépendants, donnant quatre combinaisons :

<code>smeter = "analog"</code>	l'instrument à aiguille mobile
<code>    = "digital"</code>	l'instrument à barre segmentée
 <code>layout = "v1"</code>	 sélecteur de modes au-dessus, sélecteur de bande en dessous
<code>    = "v2"</code>	sélecteur de bande au-dessus, sélecteur de modes en dessous (déplace aussi la rangée de boutons d'accord fin sur mobile)

Les quatre tiennent dans **une seule** construction, servie à `/`. Le sélecteur ☞ en haut à droite bascule de l'une à l'autre dans la page en cours d'exécution : il affecte les props `smeter/layout` d'`App.svelte`, donc rien n'est rechargé et l'audio, la cascade et les décodeurs continuent de tourner. Le choix est conservé par navigateur dans `localStorage` sous `phantom.variant`, et il l'emporte sur la valeur par défaut de la construction à la visite suivante.

La variante choisie dans `recompile.sh` est donc la variante de **départ** : ce que voit un visiteur qui arrive pour la première fois, avant de toucher au menu.

URL	ce que c'est
<code>/</code>	la page bureau ; les quatre variantes, commutables à l'exécution
<code>/mobile/</code>	page distincte, construite par <code>build-mobile.sh</code> — ce n'est pas une variante

Jusqu'au 10/08/2026, chaque variante était sa propre construction sous `/analog/`, `/digital/`, `/v2-analog/` et `/v2-digital/`, et le sélecteur naviguait entre elles — d'où le rechargement de la page et la coupure du son à chaque changement. Ces constructions n'existent plus ; chacun des quatre chemins ne contient plus qu'un petit `index.html` qui redirige vers `/`, écrit par `frontend/make-redirect-stubs.sh` à chaque construction pour que les anciens signets continuent de fonctionner.

La variante n'est écrite dans **aucun** fichier source. `src/main.js` est un point d'entrée fixe qui transmet au composant deux valeurs définies à la construction :

```
const app = new App({
 target: document.getElementById('app'),
 props: { smeter: __PHANTOM_SMETER__, layout: __PHANTOM_LAYOUT__ }
})
```

`vite.config.js` les remplit au moment de la construction, en prenant la première de ces sources qui est définie :

1. **PHANTOM\_SMETER / PHANTOM\_LAYOUT** — variables d'environnement. Plus rien ne les définit depuis la suppression des constructions par variante ; elles restent utilisables pour une construction ponctuelle avec une autre variante de départ.
2. **frontend/variant.json** — la valeur par défaut du site, écrite par `recompile.sh`. La construction de la racine (/) ne transmet aucune variante et atterrit donc ici.
3. **analog / v1** — la valeur de repli.

Ces deux valeurs ne sont que la graine. `App.svelte` les écrase à l'initialisation depuis `localStorage` si le visiteur a déjà choisi une variante, et le sélecteur 🌀 les réaffecte à chaque changement.

Dans `App.svelte`, les props sont converties en deux drapeaux aux lignes 20-21 :

```
$: isAnalog = smeter !== "digital";
$: isV2 = layout === "v2";
```

Utilisez ces deux drapeaux — jamais `smeter/layout` directement — pour que tout le fichier se lise de la même façon.

## 2. Le rôle de chaque fichier

### `frontend/src/main.js` — 12 lignes

Le point d'entrée. Rien d'autre que l'instanciation de `App` et les deux valeurs `__PHANTOM_*__` qu'elle transmet comme props. Il est **constant** — ni `recompile.sh` ni les scripts de construction ne le réécrivent désormais, et c'est cela qui permet de construire les variantes en même temps.

**Ne placez pas de code applicatif ici.** C'est le fichier d'entrée, pas un endroit pour de la logique.

### `frontend/src/App.svelte` — ~17130 lignes

Toute l'application : cascade, accord, décodeurs, favoris, panneaux, disposition bureau ET la disposition adaptative pour téléphone. Presque toutes les modifications que vous ferez se font ici.

### `frontend/src/lib/SMeterAnalog.svelte` — 824 lignes

L'instrument à aiguille mobile, complet : tracé du cadran, aiguille, la bascule cadran clair/sombre et sa clé `localStorage`, et le lissage de l'aiguille.

- Prop d'entrée : `dbm` — la puissance CALIBRÉE en dBm.
- Accepte aussi : `mobile` — attributs de canvas réduits pour la disposition téléphone.
- Réglage : `SMOOTH_TIME_MS` à la ligne 55. Plus bas = aiguille plus rapide, plus haut = plus lisse. Actuellement 16 (une image à 60 fps, en pratique le plancher).

### Les trois cadrans — et `smeter_theme.sh`

L'instrument analogique possède trois fonds, tous dessinés par le même code :

```
dark le métal brossé sombre d'origine
amber un cadran clair, gris pâle
vintage un cadran chaud, ambre vieilli
```

**À ne pas confondre avec la variante `smeter = analog / digital` ci-dessus.** La variante choisit *quel* composant d'instrument est affiché ; le cadran choisit *comment* l'analogique est peint. Ils sont stockés sous des clés `localStorage` différentes ( `phantom.variant` et `smeterTheme` ) et se règlent à des endroits différents.

**Côté visiteur.** Le canvas de l'instrument est un bouton ( `role="button"` , également Entrée/Espace au clavier, avec l'indication « click to switch style » en dessous). Chaque clic fait défiler `dark` → `amber` → `vintage` et enregistre le choix dans `localStorage.smeterTheme`. Il est relu à chaque chargement de page : le choix du visiteur survit donc indéfiniment aux rafraîchissements et aux redémarrages du navigateur. Ce n'est délibérément **pas** un cookie : un cookie serait envoyé au serveur à chaque requête de cascade et d'audio, pour une valeur que le serveur ne lit jamais.

**Côté sysop.** Deux éléments de `SMeterAnalog.svelte` déterminent ce que reçoivent tous les autres :

Ligne	Rôle
<code>let smeterTheme = 'dark';</code>	Le cadran que reçoit un navigateur <b>sans choix enregistré</b> .
<code>const SMETER_PREF_VERSION = 2;</code>	Un compteur. Lorsque la version stockée d'un navigateur est en retard, celui-ci oublie <b>une seule fois</b> son cadran enregistré et adopte la valeur par défaut ci-dessus. Incrémentez-le pour déplacer les visiteurs ayant déjà choisi.

Modifier la seule valeur par défaut ne touche donc que les visiteurs entièrement nouveaux. Les deux modifications ensemble déplacent tout le monde — c'est à cela que sert `smeter_theme.sh` à la racine du dépôt :

```
./smeter_theme.sh # afficher la valeur actuelle, puis choisir dans un menu
./smeter_theme.sh vintage # la définir aussitôt, puis proposer la reconstruction
./smeter_theme.sh dark --build # lancer frontend/build-all.sh sans demander
./smeter_theme.sh amber --no-reset # nouveaux visiteurs seulement – ne pas toucher aux autres
```

Il affiche la valeur par défaut actuelle, définit la nouvelle, incrémente le compteur de réinitialisation (toujours vers le haut — **ne le baissez jamais**, un nombre plus petit empêche simplement la réinitialisation de se déclencher), puis propose le même choix de construction du frontend que `recompile.sh`. Il ne touche **pas** au backend et ne réécrit **pas** `variant.json`.

Un changement n'atteint un auditeur connecté qu'au rechargement de sa page : la valeur par défaut est compilée dans le bundle JS et la réinitialisation s'exécute au chargement. Il n'y a pas de diffusion en direct. Un rechargement ordinaire suffit, puisque vite hache les noms de fichiers du bundle.

L'effacement unique du compteur s'impose à tout le monde, y compris aux visiteurs ayant délibérément cliqué pour choisir leur cadran. C'est voulu : c'est le seul moyen de ramener tout le site à une apparence unique. Utilisez `--no-reset` si vous préférez les laisser tranquilles.

## frontend/src/lib/SMeterDigital.svelte — 184 lignes

L'instrument à barre segmentée, complet.

- Props d'entrée : `rawDb` — la valeur BRUTE de `audio.getPowerDb()`
- `smeterOffset` — `audio.smeter_offset`
- `mobile` — attributs de canvas réduits

**REMARQUE :** cet instrument est délibérément alimenté par la puissance BRUTE plus l'offset, PAS par la valeur dBm calibrée qu'utilise le cadran analogique. Il a toujours eu sa propre correspondance. Lui fournir la valeur calibrée décalerait toutes les lectures sur chaque station numérique. Ne cherchez pas à « unifier » ces deux entrées.

#### `frontend/src/lib/BandSelector.svelte` — 69 lignes

La grille des boutons de bande. Rendue deux fois dans `App.svelte` (une fois par disposition).

#### `frontend/src/lib/ModesSelector.svelte` — 75 lignes

La grille des boutons de mode. Rendue deux fois dans `App.svelte`.

#### `frontend/src/lib/StatusIndicators.svelte` — 122 lignes

Les voyants d'état (mute, squelch, NR, NB, NS, AN, CTCSS). Prend une prop `wide` : la disposition numérique utilise des voyants plus larges que l'analogique. Les deux classes Tailwind (`w-8` et `w-10`) sont écrites intégralement dans le composant — le scanner de Tailwind ne voit que des noms de classe littéraux, ne les construisez donc jamais par concaténation de chaînes.

Tous les autres fichiers de `frontend/src/lib/` (`Spectrogram`, `PassbandTuner`, `FrequencyInput`, `VersionSelector`, ...) sont antérieurs à ce travail et sont inchangés.

## 3. Faire une modification qui ne concerne que certaines variantes

Encadrez-la par le drapeau. Dans le balisage :

```
{#if isAnalog}
 ...seules les stations analogiques voient ceci...
{:else}
 ...seules les stations numériques voient ceci...
{/if}

{#if isV2} ... {/if} {#if !isV2} ... {/if}
```

Dans un attribut `class` :

```
class="p-4 {isAnalog ? 'text-xs' : 'text-sm'} rounded-md"
```

Écrivez les deux alternatives sous forme de littéraux complets, comme ci-dessus. Tailwind parcourt les sources à la recherche de noms de classe complets ; `text-{size}` ou un nom concaténé est purgé du CSS et le style disparaît silencieusement.

Dans le bloc `<script>` : essayez de NE PAS brancher du tout. Le script calcule délibérément les entrées des DEUX instruments à chaque cycle —

```
smeterDbm = powerDbm; // cadran analogique
smeterRawDb = audio.getPowerDb(); // cadran numérique
smeterOffset = audio.smeter_offset; // cadran numérique
```

— si bien que seul le balisage doit choisir. C'est pourquoi il n'y a que 15 aiguillages de variante dans un fichier de 17000 lignes. Gardez-le ainsi ; c'est le branchement au niveau du script qui a fait diverger les quatre anciens fichiers.

## 4. Où se trouvent les aiguillages de variante

Quinze endroits dans `App.svelte`. Les numéros de ligne bougent quand vous modifiez — la façon fiable de tous les retrouver est :

```
cd frontend/src && grep -n 'isAnalog\|isV2' App.svelte
```

Au moment de la rédaction :

Ligne	Drapeau	Ce qui est commuté
20		<code>isAnalog</code> déclaré
21		<code>isV2</code> déclaré
8062	<code>isV2</code>	bureau : sélecteur de bande en premier (ordre v2)
8090	<code>isV2</code>	bureau : taille du titre du panneau
8686	<code>isV2</code>	bureau : sélecteur de modes en premier (ordre v1)
8717	<code>isAnalog</code>	largeur minimale du panneau de l'instrument (l'aiguille est plus étroite)
8730	<code>isAnalog</code>	analogique uniquement : ligne date/heure au-dessus de l'instrument
8819	<code>isAnalog</code>	analogique uniquement : voyants d'état + séparateur
8834	<code>isAnalog</code>	numérique uniquement : ligne d'heure, voyants larges, et le choix <code>&lt;SMeterDigital&gt;</code> / <code>&lt;SMeterAnalog&gt;</code> lui-même
8861	<code>isAnalog</code>	espacement vertical du bloc de fréquence
8864	<code>isAnalog</code>	marge supérieure du bloc d'accord fin
12288	<code>isV2</code>	téléphone : rangée d'accord fin au-dessus (ordre v2)
12347	<code>isAnalog</code>	téléphone : quel instrument afficher
12433	<code>isV2</code>	téléphone : rangée d'accord fin en dessous (ordre v1)
13204	<code>isAnalog</code>	taille du texte du bouton noise gate

Tout le reste du fichier est commun aux quatre variantes.

## 5. Autres réglages à connaître

Emplacement	Constante	Rôle
App.svelte ligne 5034	<code>const visualGain = 1.1;</code>	Étalonnage de l'instrument analogique. Chaque tranche de 0,10 vaut environ 5 dBm sur la lecture.
SMeterAnalog.svelte ligne 55	<code>const SMOOTH_TIME_MS = 16;</code>	Constante de temps du lissage de l'aiguille, en millisecondes.
SMeterAnalog.svelte ligne 64	<code>let smeterTheme = 'dark';</code>	Cadran par défaut de l'instrument analogique pour un navigateur sans choix enregistré. À définir avec <code>./smeter_theme.sh</code> , pas à la main.
SMeterAnalog.svelte ligne 69	<code>const SMETER_PREF_VERSION = 2;</code>	Compteur de réinitialisation unique du cadran enregistré. À incrémenter pour amener les visiteurs existants sur la valeur par défaut ; il ne fait que monter.
SMeterDigital.svelte ligne 149	<code>const DIGITAL_BAR_TRIM = 0;</code>	Décale la barre par segments entiers.
SMeterDigital.svelte ligne 31	<code>const numberOfDots = 35;</code>	Nombre de segments de la barre.

## 6. Après modification — reconstruire

Le plus simple, depuis la racine du dépôt :

```
./recompile.sh
```

L'entrée de menu **[2]** reconstruit le frontend et demande quelle variante doit voir un visiteur qui arrive pour la première fois. Ce choix est consigné dans `frontend/variant.json` — aucun fichier source n'est copié ni réécrit. (Elle régénérerait aussi `VersionSelector.svelte` à partir d'un heredoc ; cela a été supprimé lorsque le sélecteur a cessé de naviguer, car cela aurait annulé la commutation à l'exécution à chaque passage.) L'entrée **[1]** ne concerne que le backend, la **[3]** les deux.

Ou directement, depuis `frontend/` :

```
./build-all.sh construit la page bureau ET /mobile/ <-- à utiliser normalement
./build-default.sh seulement la page bureau
./build-mobile.sh seulement /mobile/
```

`build-default.sh` laisse vite vider `dist/`, ce qui supprime `dist/mobile/` jusqu'au prochain `build-all.sh` ou `build-mobile.sh`. C'est pourquoi `build-all.sh` est le choix normal.

`build-all.sh` possède sa propre copie de la logique de construction (`build_version` `SMETER` `LAYOUT` `NAME` `OUTDIR` `BASE` `LOG`) plutôt que d'appeler `build-default.sh`. Si vous changez la façon dont la page est construite, changez-la aux DEUX endroits, sinon les deux chemins divergeront.

Quelques points à connaître :

- Il pose `PHANTOM_KEEP_OUTDIR=1` pour que Vite ne vide pas le répertoire de sortie, et nettoie `dist/` lui-même : il en supprime le contenu mais conserve `users.json`, que le `spectrumserver` en fonctionnement réécrit à chaque connexion et déconnexion d'auditeur.
- Il construit `/mobile` en dernier, une fois la sortie bureau terminée.

- Sa mécanique de construction parallèle (`PHANTOM_BUILD_JOBS`, 3 par défaut) est un vestige de l'époque où il y avait cinq constructions bureau. Avec une seule, rien ne fait la queue.

**Bits d'exécution :** `recompile.sh` exécute `chmod +x` sur un script avant de le lancer, et `build-all.sh` invoque délibérément `bash build-mobile.sh` plutôt que `./build-mobile.sh` : un bit manquant ne casse donc plus la construction. Téléverser un script via l'interface web de GitHub le remet toutefois à 644, donc si vous en lancez un directement :

```
chmod +x frontend/build-*.sh
```

## 7. Comment vérifier votre modification avant de lui faire confiance

Un vite build au vert **NE SUFFIT PAS**. La construction ne peut pas voir un identifiant disparu lors du déplacement de code entre fichiers — c'est une `ReferenceError` à l'exécution, et la page ne se charge tout simplement pas. C'est arrivé deux fois.

### 1) Contrôle des identifiants non définis (attrape exactement cette panne)

```
cd frontend
npx eslint --no-eslintrc \
 --parser svelte-eslint-parser \
 --rule '{"no-undef":"error"}' \
 --env browser,es2022 \
 src/App.svelte src/lib/SMeter*.svelte
```

Un signalement connu et bénin est attendu :

```
'dBmCalOffset' is not defined
```

`typeof x === 'number'` sur un nom non déclaré ne lève jamais d'erreur : cette expression vaut déjà 0. Tout le reste est un vrai problème.

### 2) Construisez, puis CHARGEZ LA PAGE ET PARCOUREZ LES QUATRE VARIANTES

Pas seulement celle que vous avez modifiée — une modification sous `isAnalog/isV2` peut casser les trois autres alors que la vôtre fonctionne. Ouvrez `/`, parcourez les quatre entrées du menu ⚙, puis ouvrez `/mobile/`.

Ouvrez la console du navigateur sur chacune. Une page blanche ou à moitié dessinée avec une `ReferenceError` dans la console est la signature du bogue ci-dessus.

### 3) Ne laissez jamais `frontend/dist/` dans un état partiel

Un simple `npx vite build` vide le répertoire de sortie et emporte avec lui `dist/mobile/` ainsi que le `dist/users.json` vivant du serveur — la page mobile renvoie 404 jusqu'à la prochaine construction réussie. Construisez toujours via les scripts : `build-all.sh` pose `PHANTOM_KEEP_OUTDIR=1` et fait lui-même le nettoyage, en préservant `dist/users.json`. Dans le doute, relancez simplement `./build-all.sh`.

## CHAPITRE 12

# Guide de l'utilisateur

---

USER\_GUIDE.md

## Guide de l'utilisateur PhantomSDR-Plus

Bienvenue dans PhantomSDR-Plus ! Ce guide vous aidera à tirer le meilleur parti de votre écoute WebSDR.

---

### Table des matières

---

1. Introduction
  2. Premiers pas
  3. Vue d'ensemble de l'interface
  4. Opérations de base
  5. Fonctions avancées
  6. Modes de démodulation
  7. Décodeurs numériques
  8. Raccourcis clavier
  9. Favoris
  10. Utilisation sur mobile
  11. Conseils et bonnes pratiques
  12. Dépannage
  13. Foire aux questions
- 

### Introduction

---

#### Qu'est-ce que PhantomSDR-Plus ?

PhantomSDR-Plus est une radio logicielle (SDR) accessible par le web qui vous permet d'écouter des signaux radio via Internet. Aucun logiciel ni matériel particulier n'est requis de votre côté — il suffit d'un navigateur web moderne !

#### Que pouvez-vous écouter ?

Selon la configuration du WebSDR, vous pouvez vous accorder sur :

- **Radioamateurs** : opérateurs radioamateurs du monde entier
- **Stations de radiodiffusion** : radio AM/FM, radiodiffusion en ondes courtes
- **Aviation** : contrôle aérien, communications aéronautiques
- **Maritime** : liaisons navire-côte, météo maritime
- **Satellites météo** : NOAA, METEOR-M
- **Modes numériques** : FT8, RTTY, PSK31 et davantage
- **Stations utilitaires** : signaux horaires, militaires, gouvernementaux

## Configuration requise

- **Navigateur** : Chrome/Edge (recommandé), Firefox, Safari
  - **Connexion** : Internet haut débit (1 Mbit/s ou plus recommandé)
  - **Audio** : haut-parleurs ou écouteurs fonctionnels
  - **Facultatif** : souris à molette pour un accord plus facile
- 

## Premiers pas

---

### 1. Accéder au WebSDR

Ouvrez votre navigateur et rendez-vous à l'adresse du WebSDR fournie par l'opérateur.

Exemple : `http://websdr.example.com:9002`

### 2. Chargement initial de la page

Au chargement de la page, vous verrez :

- un affichage en cascade coloré montrant l'activité radio
- un panneau de commande avec l'affichage de fréquence et des boutons
- un S-mètre indiquant la force du signal
- un indicateur du nombre d'utilisateurs

### 3. Commencer à écouter

1. **Cliquez sur un signal** dans l'affichage en cascade
  2. **L'audio démarre automatiquement**
  3. **Régalez le volume** avec la commande de volume de votre navigateur ou le curseur à l'écran
  4. **Affinez la fréquence** en cliquant précisément sur le signal
-

## Vue d'ensemble de l'interface

## Composants principaux

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 ]

Les trois panneaux sous la cascade défilent comme une seule page ; dans une fenêtre étroite ils s'empilent au lieu de se placer côte à côte. **RADEL** et **RADEU** se trouvent sur la ligne de titre *Modes selector* elle-même, à sa droite.

## 1. Affichage en cascade

La cascade est une représentation visuelle des signaux radio :

- **Axe horizontal** : la fréquence
- **Axe vertical** : le temps (défile vers le bas)
- **Couleurs** : la force du signal
- **Bleu foncé/noir** : aucun signal (plancher de bruit)
- **Vert/jaune** : signaux faibles à moyens
- **Orange/rouge** : signaux forts
- **Blanc** : signaux très forts

## 2. Affichage de la fréquence

Indique la fréquence actuellement accordée dans divers formats :

- **MHz** : 7.100.000 MHz (bandes HF)
- **kHz** : 14200 kHz
- **Hz** : 145500000 Hz (VHF/UHF)

### 3. S-mètre (analogique ou numérique)

Affiche la force du signal :

- **S1-S9** : échelle standard de force de signal
- **+dB** : signaux au-dessus de S9 (p. ex. S9+20dB)
- **Analogique ou numérique** : selon la configuration

## 4. Boutons de mode

Sélectionnez le mode de démodulation :

- **AM** : modulation d'amplitude
- **FM** : modulation de fréquence
- **USB** : bande latérale supérieure
- **LSB** : bande latérale inférieure
- **CW** : code Morse (onde entretenue)
- **WBFM** : FM à large bande (radiodiffusion)
- **QUAM** : AM stéréo C-QUAM — le libellé passe au vert dès qu'un pilote stéréo est détecté

À côté du titre **Modes selector** se trouvent les boutons **RADEL** et **RADEU**, qui lancent la voix numérique RADE v1 en une seule pression — voir [Décodeurs numériques](#).

## 5. Panneau de commande

Commandes supplémentaires :

- **AGC** : contrôle automatique de gain
- **NR** : réduction de bruit (spectrale)
- **NB** : silencieux d'impulsions
- **NS** : suppression du bruit de fond
- **AN** : notch automatique
- **CTCSS** : squelch à sous-tonalité (FM). Un appui ouvre à côté du bouton une petite fenêtre avec *Any tone* et les 49 sous-tonalités normalisées (67,0–254,1 Hz) ; une fois la tonalité choisie, le bouton l'affiche (par exemple 88.5, ou ANY) et l'audio ne passe que tant que cette tonalité est reçue — l'ouverture prend environ 0,7 s. Un nouvel appui désactive le CTCSS ; Échap ou un clic à l'extérieur ferme la fenêtre sans rien changer. Les signets conservent la tonalité choisie.
- **SQL** : squelch
- **AI** : réduction de bruit par IA (modes phonie)
- **Zoom** : niveau de zoom de la cascade
- **Wheel Tuning Steps** : pas d'accord de la molette de la souris
- **Decoders** : boutons qui démarrent et arrêtent chaque décodeur en une pression

## 6. Superposition du plan de bandes

Barres colorées indiquant les attributions de fréquences :

- des couleurs différentes pour différents services
- aide à identifier ce qui est autorisé sur chaque fréquence

## Opérations de base

### S'accorder sur une fréquence

#### Méthode 1 : cliquer sur la cascade

1. Cliquez directement sur un signal dans la cascade
2. Le récepteur s'accorde sur cette fréquence
3. L'audio démarre

## Méthode 2 : saisir la fréquence

1. Cliquez sur l'affichage de la fréquence
2. Tapez la fréquence souhaitée
3. Appuyez sur Entrée

Exemples :

- 7100 → 7.100 MHz
- 14200.0 → 14.200 MHz
- 145.500 → 145.500 MHz

## Méthode 3 : utiliser la molette de la souris

1. Placez le pointeur sur l'affichage de la fréquence
2. Molette vers le haut pour augmenter la fréquence
3. Molette vers le bas pour la diminuer

## Méthode 4 : boutons de pas

1. Utilisez les boutons ▲ et ▼ à côté de la fréquence
2. Le pas varie selon le mode :
3. **AM/FM** : pas de 1 kHz
4. **BLU** : pas de 100 Hz
5. **CW** : pas de 10 Hz

## Scanner

Le scanner promène le récepteur sur une plage de canaux et s'arrête sur le premier qui porte un signal. Il se trouve sur la ligne **Fine Tuning (kHz)**, tout à droite :



Commande	Rôle
◀ ▶	Balayer vers le bas / le haut. À l'arrêt sur un signal, une flèche relance le balayage.
■	Arrêter.
○	Exclure ce canal du balayage — pour une porteuse permanente ou un birdie.
30 dB · 4 ▼	Le seuil, et à côté le niveau mesuré à l'instant. Ouvre les réglages.

Le texte à gauche indique ce que fait le scanner : **Scanner** au repos, la fréquence atteinte pendant le balayage, ● 3s pendant le décompte avant reprise, ● 30·22s tant que le canal est encore occupé, ● hold quand il reste en place.

## Réglages

- **Range** — *Scan Band* balaie la bande où commence le balayage ; *Scan Visible* balaie exactement ce que montre la cascade et la suit si vous zoomez ou la déplacez.
- **Scan** — *Every channel* accorde et écoute chaque canal à son tour ; *Skip empty* lit le spectre et saute directement aux signaux.
- **Stop at** — de combien un canal doit dépasser le bruit de fond de la bande pour arrêter le balayage, en dB. Le bruit est suivi en continu : le même réglage convient de jour comme de nuit. Un canal vide n'affiche pas 0 dB — regardez un instant la valeur sur le bouton et placez le seuil au-dessus.

- **Resume after** — combien de temps un canal doit rester calme avant que le balayage reparte. Les pauses de la parole ne le relancent pas. *Hold* reste en place jusqu'à ce que vous appuyiez sur un bouton.
- **Max stay** — repart au bout de ce temps même si le signal est toujours là, pour qu'une porteuse permanente ne bloque pas le balayage indéfiniment.

Le pas suit le mode — 1 kHz en BLU, 0,1 kHz en CW, 5 kHz en AM, 9 ou 10 kHz en ondes moyennes et 9 kHz en grandes ondes — et les arrêts tombent sur la grille des canaux. Le balayage reste dans sa plage et dans ce que le récepteur peut accorder ; à une extrémité il reprend à l'autre.

Les réglages et les canaux exclus sont mémorisés dans votre navigateur.

## Choisir le mode de démodulation

Choisissez le mode adapté au signal :

### Pour les communications vocales :

- **AM** : aviation, radiodiffusion AM, certains trafics amateurs
- **FM** : relais VHF/UHF, radiodiffusion FM
- **USB** : radioamateur HF (20 m, 17 m, 15 m, 12 m, 10 m)
- **LSB** : radioamateur HF (160 m, 80 m, 40 m, 30 m)

### Pour les données / le numérique :

- **USB** : la plupart des modes numériques (FT8, PSK31, RTTY)
- **LSB** : certains modes numériques sur les bandes HF basses

### Pour le code Morse :

- **CW** : signaux télégraphiques / Morse

## Régler le volume

- **Curseur à l'écran** : faites glisser le curseur de volume
- **Volume du navigateur** : utilisez les commandes multimédias du navigateur
- **Volume système** : ajustez le volume de votre ordinateur
- **Clavier** : utilisez les touches + et - (si prises en charge)

## Utiliser le S-mètre

Le S-mètre indique la force du signal :

- **S0-S3** : signal très faible, difficile à copier
- **S4-S6** : signal faible à moyen
- **S7-S9** : signal bon à fort
- **S9+** : signal extrêmement fort

**Astuce** : pour un meilleur son, accordez-vous sur des signaux affichant S7 ou plus.

**Changer le cadran de l'instrument** : sur le S-mètre analogique (à aiguille), cliquez sur l'instrument lui-même — ou sélectionnez-le et appuyez sur Entrée ou Espace — pour faire défiler trois fonds : métal brossé sombre, un cadran clair gris pâle et un cadran ambre chaud de style vintage. Votre navigateur retient le choix : il est toujours là après un rafraîchissement ou un redémarrage. Il vaut par navigateur et par adresse : ouvrir le récepteur par nom d'hôte et par IP donne deux réglages distincts, et une fenêtre privée repart toujours du réglage par défaut du site.

**Au-dessus de 30 MHz**, le S-mètre suit la norme VHF de l'IARU Région 1 : S9 y vaut  $-93$  dBm au lieu de  $-73$  dBm en ondes courtes, toujours avec six dB par point S. L'aiguille et la barre basculent d'elles-mêmes selon la fréquence accordée ; les valeurs en dBm et dB $\mu$ V indiquent toujours le niveau mesuré. Certaines stations règlent en outre leurs récepteurs VHF/UHF pour que les instruments restent à **0 sur un canal vide**, comme sur un transceiver VHF/UHF : au-dessus de 60 MHz, l'aiguille et la barre ne montent alors que lorsqu'un signal sort du bruit. Les valeurs dBm, dB $\mu$ V, SNR et NF indiquent toujours les valeurs réelles.

## Choisir un récepteur

Certaines stations ont plus d'un récepteur — par exemple un pour les ondes courtes et un pour la bande des 2 m. Elles affichent dans l'en-tête de la page une ligne *Receivers* : avec un bouton par récepteur ; celui sur lequel vous êtes apparaît en jaune. Appuyer sur un autre bouton ouvre la page de ce récepteur, avec sa propre cascade, ses bandes, son chat et sa liste d'auditeurs. Sur la page /mobile, les mêmes boutons se trouvent dans la barre supérieure. Une station à un seul récepteur n'affiche pas cette ligne.

## Fonctions avancées

### Contrôle automatique de gain (AGC)

L'AGC ajuste automatiquement les niveaux audio :

- **Off** : aucun réglage automatique du gain
- **Slow** : variations de niveau progressives (idéal pour la BLU)
- **Medium** : réponse équilibrée
- **Fast** : ajustement rapide (idéal pour l'AM)

**Recommandation** : commencez par « Fast » pour l'AM, « Slow » pour la BLU.

### Les quatre commandes de bruit

NR, NB, NS et AN sont des boutons marche/arrêt distincts, chacun s'attaquant à un type de bruit différent. Ils sont indépendants — en activer un n'en active aucun autre — et se combinent librement.

Aucun n'atteint les décodeurs : FT8, CW, WSPR, SSTV, FAX, NAVTEX, RTTY/PSK31/Olivia et le grabber QRSS lisent l'audio *avant* ces filtres. Vous pouvez donc les régler purement à l'oreille sans influencer sur ce qui est décodé. Voir le [manuel des décodeurs](#).

### Réduction de bruit (NR)

Réduction de bruit spectrale. Elle estime le niveau de bruit dans chaque partie du spectre audio et baisse ces parties, en laissant tranquille ce qui dépasse du bruit.

**À utiliser quand** : vous entendez un souffle régulier ou un bruit blanc derrière le signal.

Les tonalités continues — une note CW, une porteuse — sont reconnues comme du signal et protégées : NR ne mange donc pas un signal CW comme le ferait un filtre naïf. Effet typique : 10 dB de bruit en moins entre les mots, pour quelques dixièmes de dB perdus sur la parole elle-même.

### Silencieux d'impulsions (NB)

Supprime le bruit impulsif : clics, claquements, parasites atmosphériques, allumage et bruit de ligne électrique. Il surveille l'enveloppe audio et ne silence que les échantillons qui la dépassent largement — environ une milliseconde par parasite, avec des rampes d'entrée et de sortie pour que le blanking ne claque pas lui-même.

**À utiliser quand** : vous entendez des clics ou des claquements dus aux lignes électriques, aux moteurs, aux orages ou à l'allumage des véhicules.

Les notchs de ronflement secteur 50 Hz et 60 Hz suivent ce bouton.

## Suppression du bruit de fond (NS)

Mesure sur plusieurs secondes le plancher de bruit propre à la bande et applique une atténuation fixe à ce qui s'y trouve, plus profonde dans l'aigu que dans le grave. Là où NR réagit d'instant en instant, NS est la main lente et sûre : elle abaisse le souffle de la bande sans changer la sonorité du signal.

**À utiliser quand** : la bande est calme mais soufflante et vous voulez faire descendre le bruit sans la sonorité « sous-marine » d'une NR agressive.

Active seulement en USB, LSB et AM — CW, FM et les modes numériques restent intacts. Elle remesure le plancher à chaque changement de fréquence ou de mode et se stabilise en quelques secondes.

## Notch automatique (AN)

Trouve et supprime automatiquement les tonalités parasites continues — hétérodynes, porteuses, sifflements — sans que vous ayez à placer un notch à la main. Il s'adapte en continu : plusieurs tonalités peuvent être supprimées à la fois, et une tonalité qui dérive reste notchée.

**À utiliser quand** : vous entendez un sifflement ou une tonalité par-dessus le signal voulu.

Contourné en CW, où le signal utile est une tonalité continue.

**Note sur le retard** : NR et NS ajoutent chacun environ 40 ms de retard audio lorsqu'ils sont activés (les deux ensemble, environ 80 ms). NB et AN n'en ajoutent aucun. Cela ne concerne que l'écoute, jamais les décodeurs.

## Squelch automatique (SQL)

Coupe l'audio en l'absence de signal :

- **Off** : audio permanent (souffle audible)
- **Auto** : seuil défini automatiquement
- **Manual** : réglage manuel du seuil

**À utiliser quand** : vous surveillez une fréquence en attendant du trafic.

## Réduction de bruit par IA (AI)

Le bouton **AI**, avec un curseur d'intensité à côté, se trouve dans le panneau Audio & Buffer, juste sous SQ. Sur la page /mobile, il est dans l'onglet Audio, sous Squelch. Il retire le bruit de bande de la parole grâce à RNNoise, un petit réseau de neurones entraîné sur la voix.

Tout se passe dans votre propre navigateur : rien n'est envoyé à un serveur extérieur, aucun compte n'est nécessaire, et le récepteur n'a aucun travail supplémentaire.

- **Marche/arrêt** : cliquez sur **AI** ; le bouton devient bleu. Le premier clic télécharge le module (environ 1,3 Mo), et le bouton pulse pendant le chargement.
- **Intensité** : règle l'intensité de son action pendant que quelqu'un parle — des valeurs plus basses gardent davantage du son d'origine. Le haut de l'échelle est volontairement plus doux qu'une coupure totale, pour que la parole ne soit pas abîmée même à 100 %, qui nettoie pourtant davantage que 50 %. Entre les mots, elle agit plus fort d'elle-même, si bien que le souffle y baisse davantage. La valeur par défaut est 50 %. Vous pouvez la régler même quand l'IA est coupée.
- **Voyant d'état** : le voyant AI sous l'affichage de fréquence s'allume en cyan quand AI travaille, et apparaît à demi atténué quand AI est actif mais que le mode courant n'est pas un mode phonie.

**À utiliser quand** : vous écoutez de la phonie SSB ou AM sur une bande bruyante. Le souffle entre les mots baisse en général d'environ 10 dB avec les 50 % par défaut, et davantage à 100 %, jusqu'à environ 20 dB, tandis que la parole garde son niveau.

Modes phonie seulement — USB, LSB, AM et SAM. En CW, FM, modes numériques et C-QUAM, le son passe sans modification, car le réseau traite une tonalité CW ou de la musique comme du bruit. Comme les quatre commandes de bruit, elle n'atteint jamais les décodeurs. Sur les stations faibles (en dessous d'environ +6 dB de SNR), elle mesure elle-même le rapport signal/bruit et lisse davantage ses décisions, pour que la parole ne prenne pas le son gargouillant, « aqueux », que peut donner ce type de filtre ; les stations fortes sont traitées exactement comme avant. Si un signal très faible sonne encore traité, baissez le curseur. Elle ajoute environ 50 ms de retard au son.

## Tampon audio et retard

Le son arrive au navigateur par petits paquets, de 35 ms chacun sur un récepteur typique, et le navigateur en garde un peu en réserve pour qu'un paquet en retard ne provoque pas de trou. Cette réserve est le **tampon**, et elle représente l'essentiel du retard entre le récepteur et votre haut-parleur.

- **Ordinateur** : dans le panneau Audio & Buffer, le bouton à l'icône réseau monte le tampon d'un cran par clic (il passe au vert au-dessus du cran le plus bas), et le curseur à côté choisit directement un cran de 1 à 6. Le cran le plus bas convient à une connexion filaire ; les téléphones commencent au cran 2.
- **/mobile** : le bouton **BUF** et le curseur **Buffer** de l'onglet Audio font la même chose.

Le tampon se règle tout seul. La page mesure l'irrégularité avec laquelle votre connexion livre le son et ne garde en réserve que ce que la dernière minute a montré nécessaire, plus la marge de sécurité du cran choisi : 20 ms au cran 1, 50 ms au cran 2, davantage aux crans supérieurs. Après un moment de difficulté réseau, le retard supplémentaire se résorbe en une ou deux minutes, et une petite différence entre l'horloge de votre carte son et celle du récepteur ne provoque plus de coupure toutes les quelques minutes. Les ajustements sont de quelques millisecondes à la fois, faits là où le son se répète, et sont inaudibles. Les décodeurs et vos enregistrements ne sont jamais touchés : ils reçoivent chaque échantillon.

Si le son se coupe, montez le tampon d'un cran. Sur un téléphone en Wi-Fi domestique, le retard du récepteur au haut-parleur est en général d'environ 100-150 ms ; le réseau entre vous et le récepteur s'y ajoute.

## Chat

Quand l'opérateur a activé le chat, il a son propre panneau sur la page de bureau et un onglet **Chat** sur / mobile. Saisissez votre nom ou indicatif, tapez votre message et envoyez-le.

Pour répondre à un message précis, cliquez sur ➔ **Reply** à côté de son heure. Une ligne au-dessus de la zone de saisie indique à qui et à quoi vous répondez ; tapez et envoyez comme d'habitude, ou appuyez sur × ou Échap pour annuler. La réponse apparaît en retrait sous le message auquel elle répond, avec une courte ligne de citation. Une réponse à une réponse rejoint le même fil. Le serveur garde les 20 derniers messages ; quand l'original a déjà quitté cet historique, la réponse n'affiche que le nom et l'heure auxquels elle répond.

## Fonction de zoom

Agrandit l'affichage en cascade :

- **1x** : vue normale (large couverture)
- **2x** : grossissement ×2
- **4x** : grossissement ×4
- **8x** : grossissement ×8

**À utiliser quand** : vous devez mieux voir les signaux ou vous accorder finement.

## Pilotage du transceiver (CAT)

Votre propre transceiver et la page du récepteur peuvent rester sur la même fréquence : tournez le bouton d'accord du poste et la cascade suit, ou cliquez sur un signal dans la cascade et le poste s'y accorde. Cela ne concerne que *votre* session d'écoute ; personne d'autre sur le récepteur n'est affecté.

**Avec Desktop PhantomSDR+ (4.0 ou ultérieur).** L'application de bureau possède un menu **Rig**. *Rig* → *Rig control...* ouvre une fenêtre où vous choisissez votre poste et la façon de le joindre, et le menu lui-même active ou coupe la synchronisation. Elle synchronise la fréquence, le mode et la largeur de filtre, dans un sens ou dans les deux, et peut couper le son du récepteur pendant que vous émettez. Outre PhantomSDR-Plus, elle pilote de la même façon les récepteurs **KiwiSDR**, **PA3FWM WebSDR** et **UberSDR**. Le poste est joint de l'une de ces quatre façons :

- **Intégré** — sans autre logiciel : Icom (CI-V), Yaesu (nouveau CAT et FT-817/857/897), Kenwood, Elecraft, FlexRadio SmartSDR CAT, QRP Labs et autres postes compatibles Kenwood.
- **Hamlib** — tous les postes que connaît Hamlib, plus de 300, choisis dans une liste avec recherche. Les installateurs Windows incluent Hamlib ; sous Linux, installez `libhamlib-utils`.
- **rigctld sur le réseau** — un `rigctld` déjà lancé.
- **flrig** — pour un poste que flrig partage déjà avec fldigi, WSJT-X ou un logiciel de carnet de trafic.

L'application, ses installateurs et son manuel complet se trouvent sur [Desktop PhantomSDR+ \(Dropbox\)](#).

**Avec un navigateur web.** Le [CATsync Tool for WebSDRs](#) (Windows) couple un poste à la page du récepteur ouverte dans votre navigateur. Il synchronise la fréquence et le mode.

**La largeur de filtre et la coupure du son en émission** fonctionnent sur les récepteurs KiwiSDR, WebSDR et UberSDR, et sur un récepteur PhantomSDR-Plus en 4.1.0 ou ultérieure. Sur un PhantomSDR-Plus plus ancien, la fréquence et le mode se synchronisent toujours ; le filtre non.

Le manuel complet — chaque réglage, les postes pris en charge, comment on empêche les deux côtés de se battre, et le dépannage — est [Pilotage du transceiver](#).

**Pour les développeurs.** Chaque page de récepteur propose ces fonctions sur `window` ; ce sont elles qu'utilisent les deux outils :

Fonction	Rôle
<code>catsync_getFrequency()</code> / <code>catsync_setFrequency(hz)</code>	Fréquence d'accord, 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>	Largeur totale de la bande passante, en Hz. À régler après le mode : changer de mode réinitialise la bande passante
<code>catsync_getMute()</code> / <code>catsync_setMute(on)</code>	Coupure du son, via le bouton muet de la page
<code>catsync_ready</code>	<code>true</code> dès que les fonctions sont installées

Les trois dernières lignes sont nouvelles avec la 4.1.0 ; vérifiez qu'une fonction existe avant de l'appeler.

## Modes de démodulation

---

### AM (modulation d'amplitude)

#### Utilisé pour :

- les communications aéronautiques
- la radiodiffusion AM
- certains trafics radioamateurs
- les communications maritimes

#### Caractéristiques :

- large bande passante (typiquement 10 kHz)
- sensible au bruit
- facile à accorder (il suffit de cliquer sur le signal)

#### Bonnes pratiques :

- utilisez l'AGC rapide (Fast)
- activez le silencieux d'impulsions si vous entendez des clics
- accordez-vous sur le maximum du signal dans la cascade

### QUAM (AM stéréo C-QUAM)

#### Utilisé pour :

- Les émetteurs en ondes moyennes diffusant en AM stéréo

#### Caractéristiques :

- 10 kHz de large, décodé en paire stéréo plutôt qu'en AM mono
- Le libellé du bouton passe au **vert** de lui-même dès que le pilote stéréo de 25 Hz est présent : on voit ainsi quelles stations émettent réellement en stéréo avant de sélectionner le mode
- Depuis l'AM, un nouvel appui bascule en détection synchrone sur la porteuse ; le libellé passe au **jaune** et affiche SAM. Un troisième appui revient à l'AM simple
- L'audio C-QUAM est transporté en Opus ; tous les autres modes utilisent FLAC

#### Bonnes pratiques :

- Cherchez le libellé vert sur les signaux puissants en ondes moyennes la nuit
- Si la stéréo est instable, SAM est plus stable sur une porteuse faible

### FM (modulation de fréquence)

#### Utilisé pour :

- les relais radioamateurs VHF/UHF
- les services publics (police, pompiers, secours)
- la radio professionnelle bidirectionnelle
- certaines communications par satellite

#### Caractéristiques :

- bande passante étroite (typiquement 12,5 ou 25 kHz)
- excellente immunité au bruit
- « effet de capture » (le signal le plus fort l'emporte)

**Bonnes pratiques :**

- accordez-vous précisément au centre du signal
- utilisez le squelch pour couper l'audio au repos
- désactivez la réduction de bruit (inutile)

**USB (bande latérale supérieure)****Utilisé pour :**

- le radioamateurisme HF (au-dessus de 10 MHz)
- la plupart des modes numériques HF
- les communications maritimes (au-dessus de 8 MHz)

**Caractéristiques :**

- bande passante étroite (typiquement 2.4 kHz)
- utilisation efficace du spectre
- exige un accord précis

**Bonnes pratiques :**

- utilisez l'AGC lente (Slow)
- accordez-vous sur le bord inférieur du signal dans la cascade
- activez la réduction de bruit si nécessaire

**LSB (bande latérale inférieure)****Utilisé pour :**

- le radioamateurisme HF (en dessous de 10 MHz)
- certains modes numériques HF
- les communications maritimes (en dessous de 8 MHz)

**Caractéristiques :**

- identique à l'USB mais en image miroir
- convention : LSB sur les bandes HF basses

**Bonnes pratiques :**

- utilisez l'AGC lente (Slow)
- accordez-vous sur le bord supérieur du signal dans la cascade
- activez la réduction de bruit si nécessaire

**CW (onde entretenue / code Morse)****Utilisé pour :**

- la télégraphie radioamateur
- les balises de radionavigation
- les stations de signaux horaires

**Caractéristiques :**

- bande passante très étroite (100-500 Hz)
- grande efficacité
- nécessite d'apprendre le code Morse pour être compris

**Bonnes pratiques :**

- utilisez un filtre étroit (400-500 Hz)
- accordez-vous précisément sur le centre de la tonalité
- activez le filtre audio pour une meilleure tonalité

**CW-L (CW, bande latérale inférieure)**

Le même filtre morse de  $\pm 250$  Hz que **CW**, mais la tonalité est prise sous la porteuse au lieu d'au-dessus. Utile lorsqu'un signal se copie mieux du côté bas, ou lorsqu'une porteuse brouilleuse se trouve juste au-dessus de celle visée.

**Il n'y a pas de bouton CW-L sur la page de bureau.** Sa rangée de modes est USB · LSB · CW · AM · QUAM · FM. CW-L est proposé sur la page simplifiée /mobile ; sur le bureau il reste accessible par un lien ou un signet qui le nomme.

**WBFM (FM à large bande)****Utilisé pour :**

- la radiodiffusion FM (88-108 MHz)
- certaines liaisons descendantes satellite

**Caractéristiques :**

- bande passante très large (200 kHz)
- excellente qualité audio
- haute fidélité

**Bonnes pratiques :**

- accordez-vous exactement sur la fréquence centrale
- aucun squelch nécessaire pour la radiodiffusion
- profitez d'un son de haute qualité !

**Décodeurs numériques**

PhantomSDR-Plus intègre des décodeurs pour les modes numériques. Pour un guide complet, voir [Décodeurs](#).

Le plus rapide est la rangée de boutons **Decoders** du panneau principal, juste sous **Wheel Tuning Steps** : dix boutons — **FT8, FT4, FT2, JS8, CW, WSPR, FAX, SSTV, NAVTX, RTTY** — qui, en une seule pression, démarrent leur décodeur, basculent l'interrupteur principal Decoder sur ON et amènent la fenêtre du décodeur à l'écran. Le bouton reste bleu tant que le décodeur tourne ; appuyez de nouveau pour l'arrêter et fermer sa fenêtre. Le menu déroulant **Decoder Options** fonctionne exactement comme avant et reste synchronisé avec les boutons.

**RADEL** et **RADEU** (voix numérique RADE v1) disposent de leurs propres boutons à côté du titre **Modes selector** ainsi que dans les fenêtres **Modes** et **Bands**, et fonctionnent de la même manière : une pression pour démarrer, une seconde pour arrêter.

Les décodeurs SSTV, FAX HF, NAVTEX, FSK/RTTY et CW s'exécutent chacun sur leur propre thread d'arrière-plan : activer ou désactiver un décodeur n'interrompt jamais l'audio, et la cascade reste fluide pendant le décodage. Ils reçoivent l'audio prélevé avant l'AGC, la réduction de bruit et la coupure du son — vous pouvez couper le son du récepteur, le décodage se poursuit sans être affecté.

**Tant qu'un décodeur tourne, il conserve le mode et la bande passante.** Normalement le mode suit le plan de bande : accordez dans un segment marqué LSB ou AM et le récepteur bascule. Un décodeur en marche l'emporte sur ce comportement et garde le mode et la bande passante étroite qu'il lui faut, même si vous partez sur une autre bande. Le mode propre à la bande revient dès que vous arrêtez le décodeur, et les boutons de mode restent utilisables à tout moment si vous voulez reprendre la main. Le décodeur CW fait exception : il décode dans le mode où vous écoutez.

## Décodeur FT8, FT4

### Qu'est-ce que FT8, FT4 ?

- mode numérique radioamateur très répandu
- communication à signaux faibles
- transmissions de 15 secondes pour FT8, de 7,5 secondes pour FT4

### Comment l'utiliser :

1. Accordez-vous sur les fréquences FT8 ou FT4
2. Sélectionnez le mode USB
3. Activez le décodeur FT8 — appuyez sur le bouton **FT8** ou choisissez-le depuis le menu
4. Observez l'apparition des messages décodés

### Fréquences FT8 courantes (USB) :

- 160 mètres : 1.840 MHz
- 80 mètres : 3.573 MHz
- 60 mètres : 5.357 MHz
- 40 mètres : 7.074 MHz
- 30 mètres : 10.136 MHz
- 20 mètres : 14.074 MHz
- 17 mètres : 18.100 MHz
- 15 mètres : 21.074 MHz
- 12 mètres : 24.915 MHz
- 10 mètres : 28.074 MHz
- 6 mètres : 50.313 MHz (50.323 MHz pour le DX)

### Fréquences FT4 courantes (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

## Décodeur JS8

### Qu'est-ce que JS8 ?

- Le moteur signal faible de FT8, utilisé pour converser au clavier
- Du texte libre au lieu d'échanges figés : les messages longs arrivent sur plusieurs cycles
- Cinq vitesses ; **Normal** (15 s) est la vitesse d'appel et porte presque tout le trafic

### Utilisation :

1. Accordez-vous sur une fréquence JS8
2. Choisissez le mode **USB**
3. Activez le décodeur JS8 — bouton **JS8** ou depuis le menu
4. Laissez **Speed** sur *Normal* et le sync offset sur **Auto**

### Lire le panneau :

- Les messages en cours d'arrivée s'affichent en haut en vert avec un curseur clignotant — en Normal, un message peut demander une minute entière
- Les messages terminés se listent en dessous sous la forme Mode | Hz | dB | Message, **indicatifs en vert**
- Une ligne grisée et en italique a expiré avant sa dernière trame : le texte est authentique mais peut être tronqué

### Fréquences JS8 courantes (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 est bien plus calme que FT8 : une émission toutes les quelques minutes est normale, et les silences ne sont pas une panne. Guide complet : [Décodeurs](#).

## Décodeur CW

Appuyez simplement sur le bouton CW et le texte décodé apparaîtra. Appuyez de nouveau sur le bouton CW pour effacer la fenêtre et redémarrer le décodeur.

## WSPR

WSPR (Weak Signal Propagation Reporter, prononcé « whisper ») est un mode balise à signaux ultra-faibles qui cartographie les chemins de propagation HF dans le monde entier. Chaque transmission dure environ 110 secondes et tient dans un créneau de 200 Hz de large. Le décodeur attend un créneau complet de 2 minutes aligné sur l'UTC avant de décoder. Activez le décodeur et sélectionnez **WSPR** dans la liste déroulante. Ou appuyez simplement sur le bouton **WSPR**.

## FAX HF / WEFAX

Le radiofax HF (également appelé WEFAX) est utilisé par les garde-côtes et les services météorologiques du monde entier pour diffuser des cartes météo, des cartes d'état de la mer et des analyses de surface en ondes courtes. Le décodeur reconstitue l'image ligne par ligne au fur et à mesure de sa réception.

## NAVTEX

NAVTEX est le système international de diffusion maritime des informations de sécurité côtière — avertissements de navigation, prévisions météorologiques et avis de recherche et de sauvetage.

## FSK / RTTY, PSK31 et Olivia

Un décodeur polyvalent pour les modes texte à bande étroite, avec cinq variantes dans une seule fenêtre : FSK maritime (SITOR), RTTY météo, RTTY amateur, **PSK31** (modulation par déplacement de phase, 31,25 bauds) et **Olivia** (FSK multitonale avec correction d'erreurs). Le panneau s'adapte à la variante — les commandes de shift, de bauds et de trame disparaissent pour PSK31 et Olivia, et Olivia ajoute un sélecteur Mode et un curseur de squelch.

Le RTTY amateur et le RTTY météo ont un curseur **Squelch (SNR)** qui empêche le bruit de s'afficher en lettres aléatoires, et toutes les variantes sauf Olivia affichent un vrai SNR sur la même échelle de 3 kHz. Deux points à retenir : PSK31 corrige lui-même son erreur d'accord sur environ  $\pm 25$  Hz, il suffit donc de s'approcher ; Olivia exige que **Mode** (tonalités / largeur de bande) corresponde exactement à l'émission, n'envoie pas de préambule et met donc quelques secondes à se synchroniser avant que du texte apparaisse. Le panneau s'ouvre sur Olivia **8 / 250**.

## Grabber QRSS

La QRSS est de la CW envoyée si lentement qu'un seul point dure plusieurs secondes ; elle se regarde plutôt qu'elle ne s'écoute — le grabber trace le signal sur son propre affichage. Il ne figure pas dans le menu des décodeurs : il a sa propre section **QRSS** et peut tourner en même temps qu'un décodeur.

Cliquez sur **Show**, choisissez ensuite une fenêtre dans la liste **Band** et appuyez sur **Tune** : le récepteur se place sur cette fréquence en **CW**, avec une bande passante taillée pour le mode, et la trace tombe sur la ligne centrale de l'affichage. La fenêtre la plus fréquentée est celle du 30 m (10 140,00 kHz). Réglez **Speed** sur la balise — **QRSS 10** si vous l'ignorez — et soyez patient : un indicatif peut mettre dix minutes à traverser l'écran. Tous les détails dans [Décodeurs](#).

## SSTV

La télévision à balayage lent transmet des images fixes sur un canal BLU ordinaire, une ligne à la fois. Activez le décodeur, sélectionnez **SSTV** et accordez-vous sur une fréquence SSTV — 14.230 MHz est la principale fréquence d'appel internationale. Laissez **Mode** sur **Auto** : le décodeur lit l'en-tête VIS de la transmission et, si vous vous êtes accordé après l'en-tête, identifie le mode d'après la synchronisation. Les modes Martin, Scottie et Robot sont pris en charge, et l'image se construit ligne par ligne à mesure qu'elle arrive. Ou appuyez simplement sur le bouton **SSTV**.

## Report automatique des spots et graphiques du serveur

Les décodages FT8, FT4 et WSPR peuvent aussi être téléversés par le serveur lui-même — FT8/FT4 vers PSK Reporter, WSPR vers WSPRnet — par un démon autorun que le sysop démarre depuis le panneau d'administration. Il est indépendant des décodeurs de votre navigateur : il continue de tourner qu'il y ait des auditeurs ou non, et rien de ce que vous décidez dans le navigateur n'est rapporté.

Le sysop le suit à l'aide de deux compteurs, faciles à confondre : les tuiles par décodeur comptent les spots téléversés depuis le dernier démarrage du démon, tandis que le nombre à côté de chaque case bande/mode est le total depuis toujours de cet emplacement et survit aux redémarrages. Le même panneau propose une page **Graphiques** traçant la fréquence du processeur, la charge, la température et les utilisateurs connectés sur les 15 dernières minutes à 24 heures. Les deux sont décrits dans le [guide du panneau d'administration](#).

## Raccourcis clavier

Raccourcis clavier pour une utilisation plus rapide.

### Commande de fréquence

L'affichage de fréquence est une rangée de chiffres que l'on pilote directement. **Cliquez d'abord sur un chiffre** : cela le sélectionne, et tout ce qui suit agit sur la sélection.

- **Flèche gauche / droite** : déplace la sélection au chiffre voisin (huit chiffres, de 100 MHz à 10 Hz)
- **Flèche haut / bas** : incrémente ou décrément le chiffre *sélectionné* de sa valeur de position — 1 MHz sur le chiffre des MHz, 10 Hz sur le dernier
- **0 - 9** : saisit ce chiffre directement à la position sélectionnée

**Molette au-dessus de l'affichage de fréquence** : avance du pas d'accord de la bande (1 kHz sauf indication du plan de bandes). **Shift** pour 1 kHz, **Alt** pour 10 kHz.

**Molette au-dessus de la cascade** : zoome. Maintenez **Ctrl** (ou **Cmd**) ou **Shift** pour accorder au lieu de zoomer, et **Shift + Ctrl** ensemble pour caler sur le kHz entier le plus proche.

Il n'y a pas de raccourcis Page précédente / Page suivante.

## Favoris

Enregistrez vos fréquences préférées pour y accéder rapidement. Vous pouvez aussi exporter la liste des favoris et la sauvegarder localement, puis l'importer dans n'importe quel autre PhantomSDR.

Lorsque favoris et marqueurs se chevauchent :

Les favoris bleus apparaissent au-dessus

Les marqueurs jaunes apparaissent en dessous

Les clics sur les favoris sont prioritaires

### Ajouter un favori

1. Accordez-vous sur la fréquence souhaitée
2. Cliquez sur le bouton « Bookmarks »
3. Cliquez sur « Add Bookmark »
4. Saisissez une description
5. Cliquez sur « Save »

			Bookmark name   Label (optional)   [Add]
--	--	--	------------------------------------------

## Comment l'utiliser :

### Ajouter un favori :

- Nom : « Local News Station »
- Étiquette : « NEWS »
- Cliquez sur Add

### Affichage dans la cascade :

- zoomez jusqu'à ce que les marqueurs apparaissent
- vous verrez un cadre bleu marine avec « NEWS » en jaune gras

### Cliquer sur un favori :

- s'accorde sur la fréquence
- règle le mode de démodulation
- fonctionne exactement comme un clic sur un marqueur

## Gérer les favoris

- **Modifier** : cliquez sur l'icône crayon à côté du favori
- **Supprimer** : cliquez sur l'icône corbeille à côté du favori
- **Exporter** : téléchargez les favoris sous forme de fichier JSON
- **Importer** : envoyez des favoris depuis un fichier JSON

## Partager des favoris

1. Cliquez sur « Export Bookmarks »
2. Partagez le fichier JSON avec d'autres personnes
3. Les destinataires cliquent sur « Import Bookmarks »
4. Ils sélectionnent votre fichier

## Utilisation sur mobile

PhantomSDR-Plus fonctionne très bien sur les appareils mobiles !

### Deux vues mobiles

Il y a deux façons d'utiliser le récepteur sur un téléphone :

- **`http://votre_serveur:PORT/mobile`** — la page simplifiée. Pas de cascade, donc environ moitié moins de données. Saisie de fréquence, pas d'accord, modes, S-mètre, bandes, favoris, utilisateurs et chat.
- **La vue étendue** — la disposition téléphone de l'interface principale, avec la cascade et l'ensemble des commandes.

On passe de l'une à l'autre avec les boutons en bas de `/mobile` (**Mobile extended view, Full desktop view**) et avec **Simplified mobile** dans la vue étendue.

**Votre fréquence vous suit.** Changer de vue vous laisse sur le même signal — la fréquence et le mode voyagent dans le lien, vous n'atterrissez donc plus sur la fréquence par défaut du récepteur. La barre d'adresse suit également votre accord : recharger la page, la mettre en favori ou envoyer le lien à quelqu'un ramènent exactement à cette fréquence.

**Le mode suit le plan de bande sur les deux pages.** Accordez-vous sur un segment déclaré AM, LSB, USB ou CW — en saisissant une fréquence, en avançant par pas ou en appuyant sur un bouton de bande

— et le récepteur bascule sur ce mode, sur la page simplifiée comme dans l'interface complète. Un mode choisi à la main est conservé tant que vous vous déplacez à l'intérieur du même segment, et hors des bandes définies votre mode n'est pas touché. Tant que **RADE** (RADEL/RADEU) tourne, il garde le récepteur : s'accorder ne l'interrompt pas.

Lors d'un changement de vue, votre mode actuel vous accompagne ; si la vue d'arrivée n'a pas d'équivalent, le plan de bande décide pour cette fréquence — une fréquence de radiodiffusion arrive en AM, le 40 m en LSB, un segment CW en CW. **SAM** sur la page simplifiée devient AM avec le détecteur synchrone dans la vue étendue, et inversement. RADEL et RADEU n'existent que sur la page simplifiée : en la quittant, la fréquence est conservée et le plan de bande choisit le mode.

Une fréquence hors de la couverture du récepteur est ramenée au bord le plus proche : un ancien lien ne peut donc jamais vous laisser hors bande.

## Fonctions propres au mobile

- **Commandes tactiles** : grands boutons et curseurs
- **Balayage pour s'accorder** : balayez vers la gauche/droite sur la cascade
- **Pincement pour zoomer** : pincez la cascade pour zoomer/dézoomer
- **Mode paysage** : tournez l'appareil pour une meilleure vue

## Conseils pour mobile

1. **Utilisez le Wi-Fi** : la diffusion audio consomme des données
2. **Orientation paysage** : meilleure vue de la cascade
3. **Écouteurs** : meilleure qualité audio
4. **Enregistrez vos favoris** : plus facile de revenir sur des stations
5. **Fermez les autres applications** : garantit des performances fluides

## Navigateurs mobiles recommandés

- **Android** : Chrome ou Samsung Internet
- **iOS** : Mozilla
- **Les deux** : veillez à ce que le navigateur soit à jour

## Conseils et bonnes pratiques

### Pour une réception optimale

1. **Choisissez des signaux forts** : cherchez l'orange/rouge dans la cascade
2. **Accordez-vous précisément** : cliquez au centre du signal
3. **Sélectionnez le bon mode** : adaptez-le au type de signal
4. **Réglez l'AGC** : rapide pour l'AM, lente pour la BLU
5. **Utilisez NR/NB** : utile en conditions bruyantes

### Trouver de l'activité

1. **Observez la cascade** : les couleurs indiquent la force des signaux
2. **Écoutez sur les fréquences populaires** :
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. **Consultez la superposition du plan de bandes** : elle montre les attributions de fréquences
7. **Utilisez les favoris** : accès rapide aux fréquences actives

## Comprendre les conditions de propagation

### HF de jour (hautes fréquences) :

- les bandes hautes fonctionnent mieux (20 m, 15 m, 10 m)
- communications à longue distance (DX) possibles
- stations de radiodiffusion audibles

### HF de nuit :

- les bandes basses fonctionnent mieux (80 m, 40 m)
- schémas de propagation différents
- stations différentes audibles

### VHF/UHF :

- principalement à vue directe
- communications locales
- conditions plus stables

## Savoir-vivre

1. **Ne monopolisez pas le récepteur** : d'autres veulent aussi écouter
2. **Utilisez le chat avec respect** : soyez courtois envers les autres utilisateurs
3. **Signalez les problèmes** : aidez l'opérateur à entretenir la station
4. **Ne demandez pas d'assistance technique** : c'est une plateforme d'écoute. Envoyez un message au SysOp pour obtenir de l'aide.

## Dépannage

### Aucun son

#### Causes possibles :

1. Navigateur en sourdine → vérifiez les commandes de volume du navigateur
2. Système en sourdine → vérifiez le volume de l'ordinateur
3. Signal faible → accordez-vous sur un signal plus fort (S7+)
4. Mauvais mode → essayez d'autres modes de démodulation

#### Solutions :

1. Cliquez sur un signal fort (orange/rouge dans la cascade)
2. Vérifiez que le navigateur n'est pas en sourdine (icône de coupure dans l'onglet)
3. Essayez une autre fréquence
4. Rechargez la page (F5)

## Son déformé

### Causes possibles :

1. Signal saturé → signal trop fort
2. Mauvais mode → signal AM écouté en BLU, etc.
3. Interférences → signaux adjacents qui débordent

### Solutions :

1. Baissez le volume
2. Essayez un autre mode de démodulation
3. Utilisez un filtre plus étroit
4. Éloignez-vous des signaux perturbateurs

## Le son se coupe ou traîne

### Causes possibles :

1. Connexion irrégulière → les paquets arrivent en retard et le tampon se vide
2. Téléphone ou ordinateur surchargé → le navigateur ne suit pas

### Solutions :

1. Montez le tampon d'un cran (voir *Tampon audio et retard*)
2. Préférez le Wi-Fi ou le câble aux données mobiles
3. Fermez les autres onglets et applications

Pour y regarder de plus près, ouvrez la page en ajoutant `?diag=1` à son adresse (par exemple `http://your_server:PORT/mobile?diag=1`). La page /mobile affiche alors sous ses commandes le retard, l'irrégularité de la connexion et l'état du tampon ; la page de bureau écrit la même ligne dans la console du navigateur (F12) toutes les dix secondes. Utile pour signaler un problème à l'opérateur de la station.

## La cascade ne se met pas à jour

### Causes possibles :

1. Problème réseau → connexion lente ou interrompue
2. Performances du navigateur → trop d'onglets ouverts
3. Serveur surchargé → trop d'utilisateurs

### Solutions :

1. Vérifiez votre connexion Internet
2. Fermez les onglets inutiles
3. Rechargez la page (F5)
4. Réessayez plus tard, quand il y a moins d'utilisateurs

## Impossible de s'accorder sur une fréquence

### Causes possibles :

1. Fréquence hors plage → le SDR ne couvre pas cette fréquence
2. Format de saisie incorrect → utilisez le bon format (p. ex. « 14200 » et non « 14.200.000 »)

### Solutions :

1. Vérifiez la couverture en fréquence du SDR (indiquée sur la page)
2. Utilisez les exemples de format de fréquence fournis

3. Cliquez plutôt dans la cascade

## Son haché ou saccadé

### Causes possibles :

1. connexion Internet lente
2. charge élevée du serveur
3. problèmes de performances du navigateur

### Solutions :

1. Fermez les autres applications qui consomment de la bande passante
2. Réessayez en dehors des heures de pointe
3. Fermez les onglets inutiles
4. Utilisez une connexion filaire plutôt que le Wi-Fi

## Foire aux questions

### Questions générales

**Q : Ai-je besoin d'un équipement spécial pour utiliser un WebSDR ?** R : Non ! Juste un ordinateur ou un appareil mobile connecté à Internet.

**Q : L'utilisation du WebSDR est-elle gratuite ?** R : Oui, la plupart des WebSDR sont gratuits. Ils sont exploités par des bénévoles.

**Q : Puis-je émettre avec un WebSDR ?** R : Non, le WebSDR est uniquement en réception. Vous ne pouvez pas émettre.

**Q : Quelles fréquences puis-je écouter ?** R : Cela dépend de la configuration du WebSDR. Consultez les informations de la station.

**Q : Puis-je enregistrer l'audio ?** R : Certains navigateurs permettent l'enregistrement. Vérifiez les fonctions de votre navigateur.

### Questions techniques

**Q : Quel taux d'échantillonnage le SDR utilise-t-il ?** R : Cela varie selon la station. Consultez la page d'informations de la station.

**Q : Quelle est la latence ?** R : En général quelques dixièmes de seconde entre le signal radio et vos haut-parleurs : le traitement dans le récepteur, le réseau et le tampon de lecture du navigateur, qui s'adapte à votre connexion (environ 100–150 ms sur un téléphone en Wi-Fi domestique). Voir *Tampon audio et retard*.

**Q : Puis-je ouvrir plusieurs instances ?** R : En général oui, mais cela peut solliciter le serveur. Soyez prévenant.

**Q : Cela fonctionne-t-il hors ligne ?** R : Non, le WebSDR nécessite une connexion Internet.

**Q : Quels navigateurs sont pris en charge ?** R : Chrome, Firefox, Edge, Safari (toutes versions récentes)

### Questions d'utilisation

**Q : Combien de personnes peuvent écouter en même temps ?** R : Cela dépend de la capacité du serveur. Souvent 50 à 200 utilisateurs ou plus.

**Q : Puis-je voir ce que les autres écoutent ?** R : Si la fonction est activée, oui. Cherchez les indicateurs « autres utilisateurs ».

**Q : Puis-je discuter avec les autres auditeurs ?** R : Si l'opérateur l'a activé. Cherchez la zone de chat ; ➔ **Reply** répond à un message précis (voir *Chat* plus haut).

**Q : Pourquoi certaines fréquences n'affichent-elles rien ?** R : Aucun signal sur cette fréquence pour le moment. Essayez-en d'autres !

**Q : Que sont les bandes colorées sur la cascade ?** R : La superposition du plan de bandes, qui montre les attributions de fréquences.

---

## Ressources

---

### Pour en savoir plus sur la radio

- **Plans de bandes** : recherchez « amateur radio band plan » + votre région
- **Propagation** : renseignez-vous sur la propagation des ondes HF
- **Modes numériques** : documentez-vous sur FT8, PSK31, RTTY
- **Radioamateurisme** : pensez à passer une licence radioamateur !

### Trouver d'autres WebSDR

- **Annuaire WebSDR** : <http://sdr-list.xyz>
- **WebSDR.org** : <http://websdr.org>
- **KiwiSDR** : <http://kiwisdr.com/public/>

### Obtenir de l'aide

1. **Opérateur de la station** : coordonnées indiquées sur la page
  2. **Chat des utilisateurs** : demandez aux autres auditeurs (si disponible)
  3. **Forums en ligne** : cherchez des communautés WebSDR
  4. **Documentation** : reportez-vous à ce guide !
-

## Annexe : fréquences courantes

### Bandes radioamateurs HF

Bande	Plage de fréquences	Mode	Activité
160 m	1.800-2.000 MHz	LSB	Nuit/local
80 m	3.500-4.000 MHz	LSB	Nuit/régional
40 m	7.000-7.300 MHz	LSB	Jour/nuit/DX
30 m	10.100-10.150 MHz	USB	Données/CW uniquement
20 m	14.000-14.350 MHz	USB	Jour/DX
17 m	18.068-18.168 MHz	USB	Jour/DX
15 m	21.000-21.450 MHz	USB	Jour/DX
12 m	24.890-24.990 MHz	USB	Jour/DX
10 m	28.000-29.700 MHz	USB	Sporadique/DX

### Bandes radioamateurs VHF/UHF

Bande	Plage de fréquences	Mode	Activité
6 m	50.000-54.000 MHz	USB/FM	Sporadique
2 m	144.000-148.000 MHz	FM	Très active
70 cm	420.000-450.000 MHz	FM	Active

### Bandes de radiodiffusion

Service	Plage de fréquences	Mode
Radio AM	530-1710 kHz	AM
Ondes courtes	2.3-26.1 MHz	AM
Radio FM	88-108 MHz	WBFM

### Aviation

Service	Plage de fréquences	Mode
Contrôle du trafic aérien	118-137 MHz	AM
ACARS (données)	130-136 MHz	Données

### Maritime

Service	Plage de fréquences	Mode
VHF marine	156-162 MHz	FM
HF marine	2-22 MHz	USB

## Glossaire

---

- AGC** : contrôle automatique de gain — ajuste automatiquement les niveaux audio
- AM** : modulation d'amplitude — mode phonie utilisé en aviation et en radiodiffusion
- Bande passante** : la plage de fréquences occupée par un signal
- CW** : onde entretenue — signaux en code Morse
- DX** : communication à longue distance
- FFT** : transformée de Fourier rapide — convertit le domaine temporel en domaine fréquentiel
- FM** : modulation de fréquence — mode phonie pour la VHF/UHF
- HF** : hautes fréquences (3-30 MHz) — bandes à longue portée
- kHz** : kilohertz (1 000 Hz)
- LSB** : bande latérale inférieure — mode phonie pour les bandes HF basses
- MHz** : mégahertz (1 000 000 Hz)
- NB** : silencieux d'impulsions — supprime le bruit impulsionnel
- NR** : réduction de bruit — atténue le bruit de fond
- PSK** : modulation par déplacement de phase — mode numérique
- RTTY** : télétype radio — mode texte numérique
- S-mètre** : indicateur de force du signal
- SDR** : radio logicielle
- SQL** : squelch — coupe l'audio en l'absence de signal
- BLU (SSB)** : bande latérale unique (USB ou LSB)
- USB** : bande latérale supérieure — mode phonie pour les bandes HF hautes
- VHF** : très hautes fréquences (30-300 MHz) — propagation à vue directe
- UHF** : ultra hautes fréquences (300-3000 MHz) — propagation à vue directe
- Cascade** : affichage visuel du spectre radio au fil du temps
- 

### Bonne exploration du spectre radio avec PhantomSDR-Plus !

#### 73 (meilleures salutations) de SV1BTL & SV2AMK

Pour les instructions d'installation, voir [INSTALLATION.md](#). Pour les détails techniques, voir [PROJECT\\_STRUCTURE.md](#).

## CHAPITRE 13

# Émulation de client KiwiSDR

Aether\_config.md

## Émulation des clients KiwiSDR

### Connecter AetherSDR, kiwiclient et les autres logiciels KiwiSDR à PhantomSDR-Plus

Depuis la v4.1.0, PhantomSDR-Plus sait aussi répondre au **protocole KiwiSDR**, si bien qu'un logiciel écrit pour un KiwiSDR — **AetherSDR**, **kiwiclient** et les autres — se connecte directement à votre récepteur. C'est une passerelle interne au même processus serveur, sur le même hôte et le même port que vous publiez déjà : pas de second service, pas de second port, pas de mandataire.

**Elle est désactivée tant que vous ne l'activez pas.** Rien de la passerelle ne tourne — ni socket, ni vérification — tant que `[kiwi_emulation]` est absent ou à `false`, ce qui est l'état d'une arborescence fraîchement installée.

## Table des matières

1. Ce que reçoit un client Kiwi
2. Installer la passerelle
3. L'activer
4. Référence de configuration
5. Connecter un client
6. Niveau audio
7. Le S-mètre
8. Cascade et spectre
9. Si quelque chose ne va pas
10. Jusqu'où cela a été vérifié

## 1. Ce que reçoit un client Kiwi

<b>Audio</b>	<code>ws://&lt;hôte&gt;:&lt;port&gt;/kiwi/&lt;id&gt;/SND</code> — audio démodulé dans le format de trame SND propre au Kiwi
<b>Cascade</b>	<code>ws://&lt;hôte&gt;:&lt;port&gt;/kiwi/&lt;id&gt;/W/F</code> — le spectre, en trames W/F du Kiwi
<b>Accord</b>	réel, pas cosmétique : <code>SET mod=...</code> venant du client change la fréquence, la bande latérale et la bande passante côté PhantomSDR, exactement comme le ferait un auditeur au navigateur
<b>S-mètre</b>	alimenté par la puissance par bloc du démodulateur lui-même, sur la même échelle que le S-mètre web

L'accord est pris en charge pour une entrée **réelle** (`signal = "real"` dans votre `.toml`). Pour une entrée IQ, la relation entre case et fréquence diffère et n'est pas implémentée : un récepteur IQ fournira donc l'audio et la cascade mais ne suivra pas l'accord d'un client Kiwi.

## 2. Installer la passerelle

La passerelle est un ensemble de correctifs aux sources du backend, plus un nouvel en-tête, `src/kiwi_bridge.h`. Il y a deux façons de les mettre en place.

**Avec l'installateur.** `install.sh` (ainsi que `install_arch.sh`, `install_fedora.sh`, `install_opensuse.sh`) ont une étape dédiée — l'étape 17, *Kiwi client emulation*. Elle est proposée, non imposée : répondez `n` et rien n'est modifié. Pour une exécution sans surveillance, `PHANTOM_KIWI=y|n` en décide, et la valeur par défaut est oui.

**Sur une arborescence déjà installée**, lancez le script livré à la racine du dépôt :

```
cd ~/PhantomSDR-Plus
./kiwi_install.sh
./recompile.sh # choisir [1] Backend seul
```

`kiwi_install.sh` sauvegarde d'abord chaque fichier qu'il touche dans `backup_kiwi_bridge_<horodatage>/` et applique chaque correctif par correspondance exacte de texte — si une ancre n'est pas là où il l'attend (arborescence modifiée localement, autre version), il s'arrête aussitôt, nomme le fichier et le laisse intact. Il est idempotent : sur une arborescence qui possède déjà la passerelle, il signale chaque correctif comme déjà appliqué et ne change rien.

**Si vous venez d'appliquer une archive de mise à jour qui contient déjà la passerelle, vous n'avez pas besoin de le lancer.** Les sources corrigées sont dans le paquet ; `kiwi_install.sh` vous dirait seulement que chaque correctif est déjà en place.

Les fichiers qu'il corrige sont `src/client.h`, `src/signal.cpp`, `src/waterfall.cpp`, `src/spectrumserver.h`, `src/spectrumserver.cpp`, `src/websocket.cpp` et `src/http.cpp`, et il copie `kiwi_bridge.h` dans `src/`.

**Il installe la passerelle, il ne la met pas à jour.** Si un correctif *échoue* un jour sur une arborescence où la passerelle fonctionne déjà, c'est que le code a pris de l'avance sur le script — et c'est le script qu'il faut mettre à jour, pas votre arborescence.

## 3. L'activer

Ajoutez ceci au `.toml` avec lequel votre récepteur tourne réellement — le fichier que vous passez à `spectrumserver`, celui que nomme votre `start-<radio>.sh`. Pas `config.example.*` :

```
[kiwi_emulation]
enabled = true
```

Puis redémarrez le récepteur :

```
./stop-websdr.sh
./start-<votre-radio>.sh
```

`kiwi_install.sh` ajoute le bloc pour vous, avec `enabled = true`, à ceux qui existent parmi `config.toml`, `config-rx888mk2.toml`, `config-airspyhf.toml`, `config-rtl.toml`, `config-rspla.toml`, `config-fobos.toml`, `config-fobos-hf.toml` et `config-hackrf.toml` à la racine du dépôt. Il ne peut rien savoir d'une configuration que vous gardez ailleurs ou sous un autre nom, et **les archives de mise à jour ne livrent jamais de `.toml`** : après une mise à jour, le bloc est à vous d'ajouter à la main. Les `config.example.*.toml` livrés le portent, documenté et à `false`.

## 4. Référence de configuration

Tout ce que lit la passerelle tient dans une seule section `[kiwi_emulation]`. Seul `enabled` est obligatoire ; les quatre réglages fins ont tous pour défaut la valeur étalonnée, si bien qu'un bloc ne contenant que `enabled = true` est déjà correct.

Clé	Défaut	Rôle
<code>enabled</code>	<code>false</code>	Répondre au protocole KiwiSDR. Désactivé = totalement inerte.
<code>audio_gain</code>	<code>0</code>	dB. Gain de sortie, pour les clients Kiwi seulement. Voir <a href="#">Niveau audio</a> .
<code>smeter_offset</code>	<code>input.analog_smeter_offset</code>	dB. Remplace le décalage du S-mètre, pour les clients Kiwi seulement. Voir <a href="#">Le S-mètre</a> .
<code>wf_cal</code>	<code>0</code>	dB. Réglage d'affichage de la cascade et du spectre. Voir <a href="#">Cascade et spectre</a> .
<code>wf_fps_max</code>	<code>23</code>	Images de cascade par seconde. Voir <a href="#">Cascade et spectre</a> .

Un bloc complet, avec les valeurs qu'utilise le récepteur de ce projet :

```
[kiwi_emulation]
enabled = true
audio_gain = 60.0 # dB, clients Kiwi seulement. 0 = intact.
wf_cal = -10.0 # dB, affaire de goût. 0 = concorde avec le S-mètre.
wf_fps_max = 28 # images/s. 23 est le maximum du protocole.
smeter_offset = 5.0 # dB. Par défaut, input.analog_smeter_offset.
```

**Toutes ces clés sont lues au démarrage.** En changer une est un **redémarrage**, pas une recompilation — `./stop-websdr.sh` puis votre script de démarrage. Vous n'avez jamais besoin de recompiler pour les régler.

Les formes entière et décimale sont acceptées indifféremment : 28 et 28.0 se lisent de la même façon. Seule une clé réellement absente retombe sur son défaut.

## 5. Connecter un client

Pointez le client sur la même adresse et le même port que vos auditeurs — pas de chemin, pas de préfixe. AetherSDR demande un hôte et un port ; `kiwiclient` prend `-s` et `-p` :

```
kiwiclient, enregistrement de 30 secondes sur 7100 kHz LSB
python3 kiwirecorder.py -s votre.recepteur.example -p 8073 -f 7100 -m lsb --tlimit=30
```

Le client s'accorde, la cascade se remplit, le S-mètre indique. Si rien ne se passe du tout, la passerelle est presque à coup sûr encore désactivée — voir la [section 3](#).

## 6. Niveau audio

La première chose que l'on remarque, c'est qu'un client Kiwi sonne plus maigre que la page web de PhantomSDR. Ce n'est pas que la passerelle perde quoi que ce soit : un même tampon alimente tous les encodeurs, et la même fréquence avec la même bande passante, mesurée par le chemin audio du

navigateur et par `/SND`, donne des échantillons identiques. La page web est forte parce qu'`audio.js` reconstruit le son dans le navigateur — accentuation des graves, passe-bande, relèvement de présence, un compresseur avec gain de compensation et le curseur de volume. Un client Kiwi n'a rien de tout cela et ne peut pas l'obtenir.

`audio_gain` comble l'écart, pour les clients Kiwi seulement :

```
[kiwi_emulation]
enabled = true
audio_gain = 55.0 # dB, clients Kiwi seulement. 0 = intact.
```

**Commencez autour de 55-60 dB.** 55 dB n'est pas un calcul : c'est là qu'un client Kiwi et la page web de PhantomSDR ont été mesurés au même niveau, à l'écoute d'une forte station locale sur 729 kHz. En pratique on préfère souvent un peu plus — ce récepteur s'est arrêté à 60 — choisissez donc à l'oreille.

Vous ne pouvez pas abîmer l'audio avec cela. Un **limiteur de crête à anticipation** se place entre le gain et l'écrêtage 16 bits : chaque échantillon est retardé de 4 ms pendant que le gain est calculé sur de l'audio qui n'a pas encore été envoyé, si bien qu'un passage fort arrive sur un gain déjà redescendu pour lui, et les crêtes se replient au lieu d'être aplaties. Poussé à 60 et même 70 dB — bien au-delà de la marge dont dispose le gain seul — pas un échantillon sur un quart de million n'a été écrêté. Sous le seuil du limiteur, le gain vaut exactement 1,0 : un récepteur qui ne pousse pas l'audio reçoit donc les échantillons intacts.

Une valeur trop élevée coûte du volume, pas de la qualité. Au-delà d'environ 60 dB, le gain supplémentaire est simplement limité : mesuré ici, le niveau a monté d'environ 1 dB de 55 à 60 et n'a presque pas bougé de 60 à 70. Passé ce point, vous achetez de la compression, pas du volume.

Rien de tout cela ne touche à la page web, et rien ne déplace le S-mètre — celui-ci vient de la puissance du démodulateur, pas des échantillons audio : monter le volume ne peut donc pas faire mentir l'instrument.

## 7. Le S-mètre

Le S-mètre Kiwi n'est pas étalonné à part : il reproduit ce qu'affiche votre page web, étage par étage. La puissance par bloc du démodulateur est le point de départ des deux, puis la page applique `input.analog_smeter_offset`, dilate le résultat autour de  $-130$  dBm et ajoute son propre décalage d'affichage. La passerelle fait de même, si bien qu'un client Kiwi et la page montrent le même signal à la même force — vérifié sur une forte station locale, où les deux indiquaient  $-37$  dBm.

Si, et seulement si, vous voulez que les clients Kiwi affichent autre chose que votre page, donnez à la passerelle son propre décalage :

```
[kiwi_emulation]
enabled = true
smeter_offset = 5.0 # remplace input.analog_smeter_offset pour les clients Kiwi seulement
```

**Comparez les deux instruments avec le même filtre, ou pas du tout.** La lecture vient de la puissance dans la bande passante : un filtre plus large ramasse plus de bruit et indique plus haut — sur une bande calme, la même fréquence a mesuré 5,6 dB de plus à 9 kHz qu'à 2,4 kHz. Un client Kiwi avec un filtre de 6 kHz face à une page web à 2,4 kHz divergera de plusieurs dB quel que soit l'étalonnage. Accordez d'abord le mode et la largeur de bande.

**C'est l'échelle de la page, pas une échelle physique.** L'instrument web dilate sa plage pour la lisibilité : 10 dB de variation réelle s'affichent comme environ 11. La reproduire signifie que les clients Kiwi s'accordent avec votre récepteur et s'écartent d'un vrai KiwiSDR d'une quantité qui croît avec la force du signal. C'est le bon compromis quand votre propre page est la référence à laquelle tout le monde se compare ; si vous préférez que la passerelle reste physiquement honnête, c'est cette section qu'il faut changer.

## 8. Cascade et spectre

### L'échelle en dB — `wf_cal`

Chaque case de la cascade est convertie en dBm réels avant d'être envoyée, sur l'échelle même à laquelle le S-mètre est étalonné. Une porteuse indique donc **le même niveau à tous les zooms**, ce que donne un vrai KiwiSDR, et le spectre concorde avec le S-mètre de la page web sur la même bande passante.

```
[kiwi_emulation]
enabled = true
wf_cal = 0.0 # dB, clients Kiwi seulement
```

**`wf_cal` n'est pas un étalonnage.** 0 est la valeur pour laquelle le spectre concorde avec votre S-mètre, et c'est là qu'il faut le laisser si vous voulez que les chiffres veuillent dire quelque chose. Il existe parce que la *force apparente* d'un spectre est affaire de goût et de ce que votre client fait de la plage — le récepteur de ce projet tourne à -10 simplement parce que c'est ce qui rend bien dans AethersDR. Une seule valeur déplace ensemble le spectre et la cascade : ce sont les mêmes octets sur le fil, et le protocole Kiwi ne peut pas les mettre à l'échelle séparément. Si vous voulez qu'ils diffèrent, cela doit venir des réglages d'affichage du client lui-même.

Pour vérifier l'échelle, comparez la cascade sommée sur la bande passante au S-mètre sur une fréquence **calme**. Sur une porteuse, une bande passante SSB exclut la porteuse elle-même et les deux ne sont pas comparables.

### La cadence d'images — `wf_fps_max`

Un client Kiwi demande la cascade la plus rapide qu'il puisse avoir (SET `wf_speed=4`) et cale son défilement — ainsi que le moyennage de son spectre — sur la cadence que le serveur annonce. Si les deux divergent, l'affichage paraît poussif alors que rien n'est réellement en retard.

Le récepteur produit  $2 \times \text{sps} / \text{fft\_size}$  spectres par seconde. Les clients Kiwi sont servis à partir de chacun d'eux et éclaircis à la cadence qu'ils ont demandée ; la page web garde son propre rythme, plus lent, et n'est pas affectée.

```
[kiwi_emulation]
enabled = true
wf_fps_max = 23 # images par seconde
```

23 est le maximum que déclare le protocole KiwiSDR lui-même, et le défaut sûr. **C'est toujours la plus petite de `wf_fps_max` et de la cadence propre du récepteur qui l'emporte** : l'augmenter ne fait donc quelque chose que si votre FFT est assez rapide pour avoir des images en réserve. Exemple, pour un récepteur à 60 Méch/s avec une FFT de 4194304 points —  $2 \times 60000000 / 4194304 = 28,6$  images par seconde :

<code>wf_fps_max</code>	livré
absent (défaut 23)	23,0 i/s
28	28,0 i/s
40	28,6 i/s — le plafond propre du récepteur l'emporte

SET `wf_speed` venant du client est également honoré : 0 arrêt, 1 = 1 i/s, 2 le quart du maximum, 3 la moitié, 4 le maximum. Un client sur une liaison étroite peut ainsi en demander moins.

**Que l'augmenter aide relève du client, pas du serveur.** Un client qui dessine chaque image à son arrivée vous donne un défilement plus fluide ; un client qui se cale sur les 23 qu'il attend se contentera de mettre les images en trop en file d'attente, et la latence grandira. Essayez, et si la cascade paraît plus poussive plutôt que plus fluide, remettez 23.

## Ce qui fixe le reste du retard

Si vous traquez la latence, l'essentiel n'est pas dans la passerelle. La fenêtre d'analyse fait `fft_size / sps` de large — 70 ms pour une FFT de 4194304 points à 60 Méch/s — et un échantillon attend en plus jusqu'à la moitié de cela que le saut se remplit. Diviser `fft_size` par deux divise les deux par deux et double la cadence d'images, mais divise aussi par deux votre résolution en fréquence, dont vivent les décodeurs à bande étroite (WSPR, FT8, CW). Sur la plupart des récepteurs c'est un mauvais marché pour quelques dizaines de millisecondes. C'est une décision dans `[input]`, et non quelque chose que la passerelle puisse changer.

## 9. Si quelque chose ne va pas

Chaque commande envoyée par un client Kiwi est écrite dans `/tmp/kiwi_retune.log` — les demandes d'accord, les changements de mode, l'authentification, et tout ce que la passerelle n'a pas reconnu. Cela ne coûte rien de mesurable et est fait pour rester en place. Quand un client se comporte bizarrement, ce fichier montre ce qu'il a réellement demandé, ce qui est en général toute la réponse.

Symptôme	Regardez
Le client se connecte et se déconnecte aussitôt	<code>[kiwi_emulation] enabled = true</code> absent de la configuration avec laquelle le serveur a été <b>lancé</b>
Pas d'audio, pas de cascade, pas de lignes de journal	le backend n'a pas été recompilé après <code>kiwi_install.sh</code> — lancez <code>./recompile.sh</code> , option <code>[1]</code>
<code>kiwi_install.sh</code> s'arrête sur un correctif	le message nomme le fichier et l'ancre ; l'arborescence a divergé de ce qu'attend le correctif, et ce fichier est resté intact
Audio présent mais maigre et faible	c'est attendu — réglez <code>audio_gain</code> , voir la <a href="#">section 6</a>
La cascade défile poussivement	<code>wf_fps_max</code> , voir la <a href="#">section 8</a>
L'accord dans le client ne fait rien	une entrée IQ ( <code>signal = "iq"</code> ) ; l'accord n'est implémenté que pour les entrées réelles
Un réglage modifié n'a rien changé	ces clés sont lues au démarrage — redémarrez le récepteur

**L'AGC est du côté client.** `SET agc=` venant d'un client Kiwi est accepté et ignoré — l'audio arrive avec l'AGC propre de PhantomSDR appliqué, et les commandes d'AGC du client agissent sur ce qu'il reçoit.

## 10. Jusqu'où cela a été vérifié

Le format sur le fil a été contrôlé face à un récepteur en fonctionnement et relu ligne à ligne face au code des deux clients qu'il vise — le `KiwiSdrProtocol.cpp` d'AetherSDR et le `kiwi/client.py` de `kiwiclient`. Disposition et taille des trames, numéros de séquence, ordre des octets, échelle du S-mètre, accord et cascade concordent tous. La passerelle a en outre été **confirmée fonctionnelle avec l'application de bureau AetherSDR** elle-même, par son mainteneur, face à ce récepteur.

Mesuré sur ce récepteur, avec un client de test sur les sockets en service : l'accord aboutit en 35–50 ms, l'audio tourne en temps réel sans dérive, et la cascade délivre la cadence configurée sans blocage ni image perdue.

Si malgré tout quelque chose paraît faux sur votre propre installation, `/tmp/kiwi_retune.log` est le premier endroit où regarder, et un retour est bienvenu dans les deux cas.

**Voir aussi :** [Guide d'installation](#) · [Structure du projet](#) · [Guide de l'utilisateur](#)

## CHAPITRE 14

# Diversité de réception

RECEIVE\_DIVERSITY.md

## PhantomSDR-Plus — Diversité de réception

Associez votre récepteur à un **second récepteur situé ailleurs** et écoutez celui des deux qui a, à cet instant, le meilleur signal. Quand l'un des sites tombe dans un évanouissement, l'autre en général non, et la copie continue.

Le second récepteur peut être un autre PhantomSDR-Plus, un KiwiSDR, un UberSDR ou un WebSDR. Pour les trois premiers, tout s'exécute dans le navigateur : aucune modification du serveur, aucun fichier de configuration, rien à installer et aucun accès particulier à l'autre récepteur. Le WebSDR fait exception et demande un petit programme sur votre propre serveur — voir [Utiliser un WebSDR](#) plus bas.

### Ce que cela fait, et ne fait pas

Ici, diversité signifie **sélection** : à chaque instant vous écoutez l'un ou l'autre site, avec un bref fondu au changement. Ce n'est pas un réseau d'antennes en phase et les deux signaux ne sont pas additionnés.

C'est un choix délibéré, et la raison relève de la physique, pas du logiciel. Deux récepteurs distants de centaines de kilomètres entendent la même émission par des trajets ionosphériques différents. Les deux formes d'onde arrivent avec des phases sans rapport et un Doppler légèrement différent ; les additionner donne un son creux, filtré en peigne — l'effet « phasey » bien connu. Une vraie combinaison cohérente exige deux récepteurs sur une **horloge commune**, alignés à l'échantillon ; deux récepteurs indépendants reliés par internet ne pourront jamais l'offrir.

Ce que l'on gagne est donc de la **continuité, pas de la puissance de signal** :

- Sur un trajet stable où un récepteur est simplement meilleur, vous entendrez celui-là et ne gagnerez rien. C'est normal.
- Sur un trajet à QSB profond, les évanouissements de deux sites éloignés sont largement décorrélés. Le site A disparaît trois secondes, le site B non, et l'audio continue. Attendez-vous à « *j'ai cessé de perdre des mots* », pas à « *on est passé de S3 à S7* ».
- Le second bénéficie tient souvent au **bruit local et au QRM**. Deux sites ont des voisins différents, des réseaux électriques différents et des éclaboussures différentes. Sur une bande bruyante, cela peut compter davantage que l'évanouissement.

Il y a un prix : vous écoutez **avec une à trois secondes de retard**, car le flux en avance doit être retardé pour coïncider avec l'autre. Sans importance pour l'écoute et le décodage, gênant si vous cherchez à trafiquer.

#### REMARQUE

Les décodeurs numériques (FT8, JS8, WSPR, RADE et les autres) continuent délibérément d'utiliser votre récepteur **local**, pas l'audio combiné. Ces modes intègrent de façon cohérente sur tout un intervalle, et changer de site au milieu d'un intervalle constitue une discontinuité de phase qui peut vous coûter précisément le décodage que la diversité devait sauver. La diversité sert le haut-parleur ; les décodeurs conservent le flux local continu.

## Prise en main rapide

1. Ouvrez le panneau **Receive Diversity** — il se trouve juste en dessous des fenêtres de décodeurs, replié, affichant `Receive Diversity – off`. Cliquez pour le déplier.
2. Choisissez ce qu'est le second récepteur : **PhantomSDR+**, **KiwiSDR**, **UberSDR** ou **WebSDR**.
3. Saisissez son adresse — ou choisissez-en une déjà enregistrée — et appuyez sur **Start**.
4. Accordez votre propre récepteur normalement. Le second **suit automatiquement** — fréquence, mode et bande passante — à chaque changement d'accord.

C'est toute la configuration. Rien à régler sur l'autre récepteur, et pas de commande d'accord séparée : il vous suit toujours.

## Adresses

Second récepteur	Ce qu'il faut saisir	Remarques
PhantomSDR+	<code>host:8900</code>	Le port normal du récepteur
KiwiSDR	<code>host:8073</code>	Le port propre au KiwiSDR
UberSDR	<code>host</code>	Son <b>port web normal</b> , pas le 8073
WebSDR	<code>host:8901</code>	Le port propre du site. Nécessite le relais — voir plus bas

Un simple nom d'hôte, une adresse `http://` ou une adresse `ws://` complète sont tous acceptés. Si l'autre récepteur est servi en HTTPS, saisissez le nom d'hôte et la connexion sera sécurisée.

## Récepteurs enregistrés

Chaque adresse utilisée est mémorisée, dans une liste propre à chaque type de récepteur — une adresse KiwiSDR n'est jamais une suggestion utile quand le sélecteur affiche WebSDR. Les listes restent dans votre propre navigateur : elles ne sont envoyées nulle part, et chaque auditeur a la sienne.

Le bouton  à côté du champ d'adresse ouvre la liste.

- **Les noms.** Toute entrée peut recevoir un nom — `Twente` se lit mieux que `websdr.ewi.utwente.nl:8901`. Les entrées nommées apparaissent en vert au-dessus de leur adresse, et le nom s'affiche à côté de l'adresse dans la liste déroulante pendant la saisie. Une entrée est identifiée par son adresse : la nommer ne fait donc jamais figurer la même station deux fois. Tant que la diversité fonctionne, le récepteur auquel elle est reliée affiche son nom **en gras et clignotant lentement**, si bien que la liste indique aussi lequel vous entendez réellement.
- **add** saisit un récepteur à la main, sans s'y connecter d'abord.  corrige le nom et l'adresse sur place,  retire une entrée, et **clear all** vide la liste du type affiché. Entrée valide, Échap annule.
- **\*\*\*\*** ouvre la page web propre à ce récepteur dans un nouvel onglet. Ce qui est enregistré, c'est l'adresse que le logiciel appelle : elle est donc d'abord retransformée en adresse consultable — `ws://` devient `http://`, et le chemin `/audio`, `/ws` ou, pour un KiwiSDR, `/kiwi/.../SND` dont chaque type de récepteur a besoin est retiré, puisque la page du site se trouve à la racine. C'est un vrai lien : clic du milieu et appui long se comportent comme d'habitude.
- **L'ordre est le vôtre.** Déplacez une ligne avec  , ou faites-la glisser à la place qui lui revient. C'est dans cet ordre que la liste déroulante propose les adresses.
- **export et import.** Les listes sont conservées dans le stockage du navigateur, effacé en même temps que les données du site — c'est précisément ce contre quoi ceci protège. Export écrit les quatre listes, noms compris, dans un fichier JSON que vous pouvez garder ou emporter vers un autre navigateur ou une autre machine. Import le relit et demande s'il faut le **fusionner** (merge) avec l'enregistré ou tout **remplacer** (replace).

-  **QR**. Dessine toutes les listes en code QR à l'écran, pour les emporter sur un téléphone — voir « Sur un téléphone » plus bas. Une liste trop grande pour être scannée se rabat sur le texte à côté, qui se copie et se colle n'importe où.

Le nombre de récepteurs conservés n'est pas limité.

## Sur un téléphone

La page `/mobile` offre la même fonction dans un onglet **Div**, à côté d'Audio, Bands, Marks, Users et Chat : type de récepteur, adresse, Start et Stop, les récepteurs enregistrés en liste à toucher, les valeurs en direct et la correction de SNR. Chaque ligne porte aussi le lien `****` vers la page propre du récepteur. Le second récepteur suit l'accord du téléphone tout seul, exactement comme sur la page de bureau. Ce qui manque, c'est l'édition — renommer, réordonner, supprimer — qui reste sur la page de bureau, où la place ne manque pas.

La liste enregistrée appartient à un navigateur : un téléphone commence donc avec une liste vide, quel que soit le nombre de récepteurs enregistrés sur l'ordinateur. C'est à cela que sert le code QR : sur l'ordinateur appuyez sur  puis  **QR**, scannez le code avec l'appareil photo du téléphone et collez le texte dans **Import** de l'onglet Div. Il demande de fusionner ou de remplacer, et Export sur le téléphone envoie une liste dans l'autre sens.

## Utiliser un WebSDR

Un WebSDR — le logiciel écrit par Pieter-Tjerk de Boer, PA3FWM, qui fonctionne à Twente et sur plusieurs centaines d'autres sites — peut servir de second récepteur, mais pas directement depuis le navigateur.

La raison est un contrôle délibéré de leur côté. Un WebSDR refuse la connexion audio si l'en-tête `Origin` ne nomme pas son propre site, et `Origin` est un *nom d'en-tête interdit* : le navigateur le fixe d'après la page où vous êtes et aucun script n'a le droit de le modifier. Il n'existe aucun contournement côté client, et il ne devrait pas en exister.

La connexion est donc établie par un petit programme sur votre propre serveur, `websdr_relay.py`. Votre navigateur parle au relais, et le relais parle au WebSDR.

## Installer le relais

Les quatre installateurs de distribution le proposent en étape facultative. Pour une installation déjà en place :

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

Il demande un port, reprend votre indicatif et l'adresse de votre station depuis `frontend/site_information.json`, et propose d'installer un service systemd pour que le relais démarre au boot. Les réglages sont dans `websdr_relay.json` et se modifient à tout moment.

**Une chose que l'installateur ne peut pas faire à votre place : ouvrir le port du relais sur votre routeur.** Les navigateurs des auditeurs se connectent directement au relais — il ne passe pas par le récepteur — sans cette redirection, la diversité avec un WebSDR ne fonctionne donc que depuis votre propre réseau. Les trois autres types de récepteur ne sont pas concernés.

Si le panneau d'administration est installé, son tableau de bord affiche une carte **WebSDR Diversity Relay** avec l'état, le port, le nombre de sessions en cours et vers quels sites.

## Être un invité correct

Le relais se connecte aux récepteurs des autres depuis votre serveur et non depuis l'adresse de chaque auditeur ; il est donc conçu pour bien se tenir :

- **Une limite de dix sessions simultanées vers un même WebSDR**, et soixante au total. C'est ce qui évite qu'une journée chargée chez vous ressemble à une attaque chez quelqu'un d'autre.
- **Un User-Agent nommant votre station et son opérateur**, pour qu'un opérateur qui préfère ne pas être utilisé ainsi sache exactement à qui écrire. Remplissez-le honnêtement.
- **Seules les commandes d'accord sont transmises**. La connexion ne peut servir à envoyer autre chose.
- **Les adresses privées, de loopback et de NAT opérateur sont refusées**, afin qu'aucun visiteur ne puisse pointer le relais vers quelque chose dans votre machine ou votre réseau local.

### IMPORTANT

Le contrôle d'Origin existe parce que les auteurs de WebSDR ne voulaient pas que leurs récepteurs soient pilotés depuis les pages d'autrui. Utiliser le relais est une décision réfléchie, pas un oubli de leur part. Laissez la limite où elle est, gardez le User-Agent honnête, et arrêtez si un opérateur vous le demande.

## Ce qui change avec un WebSDR

- **Beaucoup de sites ne couvrent que quelques tranches étroites du spectre**, et non une plage continue — par exemple une fenêtre de 256 kHz sur 40 m. La diversité ne s'engage qu'à l'intérieur, et le panneau lit la liste des bandes du site pour savoir où elles se trouvent.
- **La fréquence d'échantillonnage audio varie** avec la largeur du filtre et s'écarte davantage du nominal que chez les autres types : l'alignement peut donc être un peu plus long.
- **La CW n'est pas adaptée** à la convention de WebSDR, qui place la bande passante entièrement sous la porteuse. La BLU et l'AM sont correctes.

## Lire le panneau

En fonctionnement, le panneau affiche un petit tableau :

Ligne	Signification
<b>link</b>	La connexion au second récepteur. <code>ready</code> est l'objectif. En cas d'échec, le texte d'erreur du serveur distant et le code de fermeture WebSocket s'affichent
<b>aligned</b>	Le décalage mesuré entre les deux flux une fois verrouillés, sinon <code>searching</code>
<b>corr</b>	La force de corrélation des deux flux. Significatif seulement après l'alignement
<b>remote audio</b>	Audio décodé en provenance du second récepteur. <code>none</code> en ambre signifie que rien n'est décodé
<b>SNR local / remote</b>	Les deux estimations de rapport signal/bruit comparées
<b>switches</b>	Le nombre de changements de site

À côté du titre, un mot vous dit où vous en êtes :

off (gris)	à l'arrêt
<b>Please wait...</b> (ambre)	en cours de connexion, ou connecté et encore en alignement
<b>Ready</b> (vert)	relié, verrouillé et suivant le meilleur site
l'erreur du récepteur distant (rouge)	échec, et cela ne se rétablira pas tout seul

La distinction entre ambre et rouge est celle qui fait gagner du temps : ambre veut dire continuez d'attendre, rouge veut dire cessez d'attendre et lisez le message.

Le point coloré ajoute quel site vous entendez réellement — vert votre propre récepteur, cyan le distant.

## L'alignement demande environ 15 secondes

C'est normal. Les deux flux sont alignés en corrélant leurs **enveloppes audio**, ce qui exige une fenêtre de son. La recherche se fait en deux temps : une recherche étroite de  $\pm 1,5$  seconde peut démarrer dès qu'environ 9 secondes de son ont été réunies, et une seconde mesure indépendante, cinq secondes plus tard, doit **concorder** avec la première avant que l'alignement soit accepté. Lorsque les deux stations sont plus éloignées que cela dans le temps — une ou deux secondes de tampon supplémentaire quelque part sur le trajet —, la recherche complète de  $\pm 4$  secondes prend le relais vers 14 secondes et la confirmation suit à 19.

Cette étape de confirmation compte. Lorsque deux sites s'évanouissent en opposition — sans jamais porter le signal en même temps —, une corrélation unique se verrouille volontiers sur un décalage parfaitement faux mais assuré, qui s'entend comme un écho. Exiger deux mesures indépendantes concordantes l'écarte.

Une seconde chose doit être juste avant qu'un verrouillage tienne : les cadences d'échantillonnage des deux récepteurs. Deux récepteurs, ce sont deux horloges et deux chaînes de décimation : leurs flux audio n'arrivent **pas** tout à fait à la même cadence, même lorsque tous deux annoncent 12 kHz — 2% d'écart est normal, soit 240 échantillons de dérive par seconde, bien au-delà de la tolérance d'alignement. Le flux distant est donc rééchantillonné en continu à la cadence de votre propre récepteur. Ce rapport est obtenu en **comptant les échantillons**, non par corrélation : il n'a donc pas besoin de son propre verrouillage et il est établi dès les premières secondes — avant même la première recherche d'alignement. Un signal faible ou intermittent peut demander deux ou trois tentatives ; 30 secondes ne sont donc pas inquiétantes, une minute sans rien l'est.

Pendant `searching`, les deux valeurs de SNR affichent `0.0`. C'est voulu : les deux estimations sont mesurées sur des échantillons **alignés en contenu**, donc rien n'est mesuré avant qu'il y ait alignement. `0.0 / 0.0` veut dire « pas encore verrouillé », pas « pas de signal ».

## Remote SNR trim

Ce curseur pondère le choix entre les deux sites. Il est ajouté au SNR mesuré du récepteur distant avant la comparaison — positif favorise le distant, négatif le vôtre. Il ne change **rien d'autre** : ni le niveau audio, ni la combinaison, seulement le site retenu.

Il existe parce que les deux valeurs de SNR ne sont pas toujours comparables. Toutes deux sont mesurées de la même façon — comme un écart entre centiles de l'audio — mais un récepteur dont le codec relève son propre plancher de bruit affiche moins qu'il ne le mérite. Un KiwiSDR applique un fort gain et un limiteur, ce qui comprime l'écart ; un codec avec pertes fixe un plancher de bruit sous lequel le signal ne descend jamais.

**Comment le régler :**

1. Accordez les deux récepteurs sur un signal présent de façon fiable et attendez `aligned`.
2. Observez les deux valeurs de SNR pendant une demi-minute. Notez l'écart habituel.
3. Utilisez les extrémités comme test d'écoute : **+15** impose le site distant, **-15** le vôtre. Écoutez chacun quelques secondes. C'est le seul moyen d'entendre un site isolément.
4. Si le distant affiche par exemple 5 dB de moins mais sonne aussi bien, réglez **+5**. Vous faites concorder les deux lectures quand les deux sites sonnent pareil.
5. Surveillez le compteur **switches** sur dix minutes d'écoute. Des bascules incessantes signifient que le réglage est trop proche de l'égalité ; ne jamais basculer alors que votre récepteur s'évanouit audiblement signifie qu'il est trop décalé dans l'autre sens.

Points de départ : **0** pour un autre PhantomSDR-Plus, qui emploie une chaîne audio identique et est comparable par construction ; une petite valeur **positive** pour un KiwiSDR ou un UberSDR.

**ASTUCE**

S'il vous faut plus de  $\pm 8$  dB environ, arrêtez de corriger. À ce stade, l'explication honnête est en général qu'un récepteur est réellement moins bon pour ce trajet, et pondérer au-delà d'une différence réelle revient simplement à écouter le site le plus faible.

Le réglage s'applique immédiatement et est mémorisé dans votre navigateur.

## Choisir un second récepteur

**La distance compte.** Les évanouissements se décorrèlent avec la distance — quelques centaines de kilomètres en HF constituent un bon objectif. Deux récepteurs dans la même ville s'évanouissent ensemble et n'apportent rien. Trop éloignés, et le second récepteur risque de ne pas entendre votre signal du tout.

**Les deux doivent réellement entendre le signal.** Aucune logiciel ne contourne cette exigence. Si le second récepteur n'entend pas ce que vous écoutez, la corrélation reste faible et il ne se verrouillera pas — volontairement, car un alignement erroné sonne plus mal que pas de diversité du tout.

**Couverture.** Un second PhantomSDR-Plus ou un KiwiSDR annonce la plage de fréquences qu'il couvre, et le panneau vous avertit si vous vous accordez en dehors. Un WebSDR annonce ses bandes, souvent des tranches étroites plutôt qu'une plage continue. UberSDR n'annonce pas sa couverture : là, le refus de se verrouiller est votre seul indice.

**IMPORTANT**

Les récepteurs publics sont tenus par des bénévoles et disposent d'un nombre limité de places d'écoute. Une session de diversité en occupe une tant qu'elle dure, exactement comme un auditeur humain. Soyez prévenant avant d'en laisser une connectée indéfiniment, et demandez à l'opérateur si vous comptez utiliser son récepteur intensivement.

## Dépannage

**Link ne quitte jamais connecting, ou enchaîne connexion/fermeture** L'adresse ou le port sont probablement erronés. Voyez le tableau ci-dessus — en particulier, UberSDR utilise son port web normal, pas le 8073. Le panneau affiche le code de fermeture WebSocket, et la console du navigateur (F12) l'erreur sous-jacente.

**Link affiche une erreur suivie d'un texte** Ce texte vient de l'autre récepteur et il est généralement précis : session refusée, serveur plein, ou récepteur qui n'accepte pas les clients extérieurs.

**Le KiwiSDR refuse la connexion** Certains KiwiSDR exigent un mot de passe, et sur un UberSDR le point d'accès compatible KiwiSDR est désactivé par défaut. Pour un UberSDR, utilisez le type **UberSDR** — cette voie est toujours disponible.

**WebSDR : the WebSDR relay is not reachable** Le relais ne tourne pas, ou votre navigateur n'atteint pas son port. Il doit être joignable sur le **même nom d'hôte que celui qui sert la page du récepteur**, car le navigateur s'y connecte directement et non à travers le récepteur. Ouvrez `http://<ce nom d'hôte>:<port du relais>/status` dans le même navigateur pour savoir lequel des deux. S'il répond sur votre réseau local mais pas depuis l'extérieur, le port n'est pas redirigé sur le routeur.

**WebSDR : une erreur mentionnant une connexion refusée** Ce site est verrouillé plus strictement que le contrôle d'`Origin` habituel, ou son opérateur a bloqué cette station.

**Link est ready mais remote audio indique none** Aucun audio n'est décodé. Rechargez la page ; si cela persiste, la console du navigateur indiquera quel décodeur a échoué.

**L'audio arrive mais ne s'aligne jamais** Les deux récepteurs n'entendent pas la même chose. Essayez une station de radiodiffusion puissante, ou une bande où les deux sites bénéficient d'une bonne propagation. Une bande calme avec seulement du bruit des deux côtés ne corrélera jamais — et c'est très bien ainsi.

**Cela bascule sans arrêt** Les deux sites sont trop proches en SNR. Décalez légèrement le trim pour qu'un site soit préféré de façon stable.

**Le second récepteur n'est jamais utilisé** Vérifiez que `remote audio` augmente et que les valeurs de SNR paraissent cohérentes. Si le distant affiche bien moins qu'il ne sonne, c'est précisément à cela que sert le trim.

## Limites

- **Ce n'est pas cohérent.** Pas de phasing, pas de nulling, pas de radiogoniométrie. Annuler une source de bruit exige deux antennes sur une même horloge et un même site ; c'est une autre technique pour un autre problème.
- **Une à trois secondes de retard**, inévitablement.
- **Deux récepteurs seulement.**
- **Mono.** Si votre récepteur est dans un mode stéréo comme le C-QUAM, l'audio passe sans modification et la diversité ne s'engage pas.
- **L'alignement exige du signal.** En dessous d'environ 10 dB de SNR sur l'un des deux sites, attendez-vous à rester sur `searching`.
- **Un WebSDR exige le relais** sur votre propre serveur, et son port redirigé. Les trois autres types de récepteur n'ont besoin ni de l'un ni de l'autre.

## CHAPITRE 15

# Pilotage du transceiver

RIG\_CONTROL.md

## PhantomSDR-Plus — Pilotage du transceiver (CAT)

Gardez **votre propre transceiver** et un **récepteur PhantomSDR-Plus** sur la même fréquence, le même mode et le même filtre. Tournez le bouton d'accord du poste et la cascade suit ; cliquez sur un signal dans la cascade et le poste s'y accorde. Passez en émission, et le récepteur peut se taire pour ne pas vous renvoyer votre propre signal.

Cela fonctionne avec les récepteurs **PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR et UberSDR**, le vôtre ou celui d'un autre, et ne déplace jamais que *votre* session d'écoute — personne d'autre sur le récepteur n'entend ni ne voit rien. L'opérateur du récepteur n'a rien à installer ni à configurer.

### Ce qu'il vous faut

Deux façons de raccorder un poste, et une condition côté récepteur :

Élément	Ce que c'est	Synchronise
<b>Desktop PhantomSDR+ 4.0 ou ultérieur</b>	L'application de bureau, avec un menu <b>Rig</b> . Linux (PC et Raspberry Pi) et Windows.	Fréquence, mode, largeur de filtre, coupure du son en émission — dans un sens ou dans les deux
<b>CATsync Tool for WebSDRs</b>	Un programme Windows séparé qui couple un poste à la page du récepteur dans votre navigateur.	Fréquence et mode
<b>TCI-CAT sur la page du récepteur</b>	Un bouton sur la page PhantomSDR-Plus elle-même, dans n'importe quel navigateur. Parle TCI avec ExpertSDR, AetherSDR ou Thetis, ou avec tout poste pris en charge par Hamlib grâce à un petit pont. Uniquement sur les récepteurs PhantomSDR-Plus en version 4.1.0 ou ultérieure.	Fréquence, mode et largeur de filtre dans les deux sens ; coupure du son en émission
<b>Le récepteur</b>	PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR ou UberSDR. Un récepteur PhantomSDR-Plus demande la <b>4.1.0</b> ou ultérieure pour la largeur de filtre et la coupure du son.	Un PhantomSDR-Plus plus ancien synchronise toujours fréquence et mode

L'essentiel de ce manuel décrit Desktop PhantomSDR+. Le bouton TCI-CAT a sa propre section, [TCI-CAT sur la page du récepteur](#). Le CATsync Tool a sa propre documentation sur son site.

### Récepteurs pris en charge

L'application reconnaît le type de récepteur d'une fenêtre de station et le pilote par les commandes propres à cette page. La fenêtre Rig control indique le type trouvé à côté du nom de la station.

Récepteur	Fréquence et mode	Largeur de filtre	Coupure du son en émission
PhantomSDR-Plus	Oui	Avec la 4.1.0 ou ultérieure	Avec la 4.1.0 ou ultérieure
KiwiSDR (y compris Web-888)	Oui	Oui	Oui
PA3FWM WebSDR	Oui, avec changement de bande sur un site multibande	Oui	Oui
UberSDR	Oui	Oui	Oui

Un récepteur web a moins de modes que la plupart des postes, si bien que certains modes du poste partagent un mode du récepteur : KiwiSDR et WebSDR n'ont qu'un seul CW pour CW et CW-R. Les modes de récepteur cités dans ce manuel sont ceux de PhantomSDR-Plus ; les autres récepteurs utilisent l'équivalent le plus proche. Un WebSDR garde un filtre CW sous 1 kHz et les autres filtres à 1 kHz ou plus, parce que c'est ainsi que la page elle-même reconnaît le CW, et une fréquence hors de toutes les bandes d'un site WebSDR est laissée de côté. UberSDR règle son filtre par pas de curseur, de sorte qu'une largeur peut tomber jusqu'à environ 50 Hz de celle du poste. Tout autre type de page — OpenWebRX, par exemple — affiche *not a receiver this app can drive*, et rien n'est synchronisé.

## Ce qu'il fait, et ce qu'il ne fait pas

- Il synchronise **un poste avec une fenêtre de récepteur** à la fois.
- Il lit les deux côtés plusieurs fois par seconde et, s'ils diffèrent, règle l'un sur l'autre. Il **n'émet pas**, ne commande pas le PTT et n'envoie de son nulle part.
- Déplacer le récepteur ne change que votre propre session. Les autres auditeurs du même récepteur ne sont pas affectés, et l'opérateur n'a pas à l'autoriser.
- Un port série ne peut être ouvert que par **un programme à la fois**. Si WSJT-X, un carnet de trafic ou un utilitaire du constructeur tient déjà le port, utilisez le choix **flrig** ou **rigctl on network** pour partager le poste au lieu de vous disputer le port.

## Démarrage rapide

1. Ouvrez une station dans Desktop PhantomSDR+ comme d'habitude.
2. **Rig → Rig control...**
3. Sous **Connection**, choisissez **Built-in** si votre poste est dans la liste, sinon **Hamlib (all rigs)**.
4. Choisissez votre poste, le port série et la vitesse réglée dans le menu CAT ou CI-V du poste.
5. Sous **Sync**, laissez **Both directions** sélectionné.
6. Appuyez sur **Connect**. Les deux affichages du haut — transceiver et récepteur — doivent montrer la même fréquence en moins d'une seconde.

Chaque réglage est enregistré dès que vous le modifiez. La fois suivante, **Rig → Connect** suffit, ou cochez **Connect when the app starts**.

## Choisir comment joindre le poste

Choix	À utiliser quand	Il faut
<b>Built-in</b>	Votre poste figure dans la liste ci-dessous.	Rien d'autre
<b>Hamlib (all rigs)</b>	Votre poste est autre chose — Hamlib en connaît plus de 300. L'application lance pour vous le <code>rigctld</code> de Hamlib, sur un port local privé, et l'arrête à la déconnexion.	Windows : rien, Hamlib est inclus. Linux : <code>sudo apt install libhamlib-utils</code>
<b>rigctld on network</b>	Un <code>rigctld</code> tourne déjà, sur cet ordinateur ou un autre de votre réseau.	L'hôte et le port (4532 par défaut)
<b>flrig</b>	flrig pilote déjà le poste pour fldigi, WSJT-X ou un carnet de trafic.	flrig lancé, avec son port XML-RPC (12345 par défaut)

## Postes avec pilote intégré

La vitesse et l'adresse CI-V indiquées sont les valeurs d'usine que l'application préremplit. **Ce n'est qu'un point de départ — réglez-les selon le menu de votre poste.**

Famille	Postes	Vitesse par défaut	Remarques
<b>Icom CI-V</b>	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 et tout autre poste CI-V	19200	Adresse CI-V préremplie par modèle (IC-7300 94, IC-705 A4, IC-9700 A2, IC-7610 98 ...)
	IC-746PRO, IC-718, IC-R75	9600	
<b>Xiegu (CI-V)</b>	G90, X6100	19200	Adresse 70 ; vérifiez le menu
<b>Yaesu nouveau CAT</b>	FTDX101D/MP, FTDX10, FT-710, FT-991/A, FT-891, FTDX5000, FTDX3000, FTDX1200, FT-950, FT-2000, FT-450/450D	38400	
<b>Yaesu CAT classique</b>	FT-817/818, FT-857/857D, FT-897/897D	38400	2 bits de stop ; s'accorde par pas de 10 Hz
<b>Kenwood</b>	TS-990S, TS-890S, TS-590S/SG	115200	
	TS-480, TS-2000, TS-870S	57600	
<b>Elecraft</b>	K4, K3/K3S, KX3, KX2	38400	La largeur de filtre est synchronisée
<b>Compatibles Kenwood</b>	FlexRadio SmartSDR CAT (port virtuel), QRP Labs QMX/QMX+/QDX, (tr)uSDX, Lab599 Discovery TX-500, autres postes compatibles Kenwood	9600–38400	

Un poste censé être compatible mais qui ne dialogue pas avec un pilote intégré fonctionne généralement avec **Hamlib**, qui s'accommode de bien plus de variantes.

## Réglages du port série

Réglage	Quoi y mettre
<b>Serial port</b>	Le port du poste. Les adaptateurs USB et les postes à port USB sont listés en premier. <b>Other / network address...</b> accepte un port absent de la liste — COM7, /dev/ttyUSB1 — ou tcp://hôte:port pour un port série servi sur le réseau par ser2net ou équivalent.
<b>Speed (baud)</b>	Exactement ce qu'indique le menu de débit CAT / CI-V du poste. Un débit erroné ressemble à un poste qui ne répond jamais.
<b>Stop bits</b>	1 pour presque tout ; 2 pour la famille FT-817/857/897.
<b>CI-V address</b>	Icom uniquement, en hexadécimal (94, pas 148). Doit correspondre au menu d'adresse CI-V.
<b>DTR / RTS</b>	Laissez-les <b>désactivés</b> sauf si votre interface en a besoin. Beaucoup de câbles CAT déclenchent l'émetteur, ou réinitialisent le poste, sur l'une de ces lignes.
<b>Hardware flow control</b>	Désactivé, sauf si le manuel demande RTS/CTS.

Avec **Hamlib**, les mêmes réglages sont transmis à `rigctld`. Les bits de stop y ont en plus un choix *Rig default*, et **Extra rigctld options** accepte toute autre option de `rigctld`, par exemple `--set-conf=post_write_delay=10`. **rigctld program** permet de désigner un `rigctld` précis si plusieurs sont installés.

## Synchronisation

### Sens

Choix	Ce qui se passe
<b>Rig → receiver</b>	La fenêtre du récepteur suit le poste. Une modification dans la cascade est ramenée à la fréquence du poste.
<b>Receiver → rig</b>	Le poste suit la fenêtre du récepteur. Tourner le bouton du poste est annulé.
<b>Both directions</b>	Le côté touché <b>en dernier</b> l'emporte. Au moment de la connexion, avant qu'aucun ne soit touché, c'est le poste.

Le sens se change aussi depuis le menu **Rig** en cours de connexion.

### Comment on empêche les deux côtés de se battre

Chaque valeur écrite par l'application réapparaît un instant plus tard comme un changement de l'autre côté. Pris au pied de la lettre, le poste et le récepteur se courraient après sans fin. L'application l'évite de trois façons :

- La page du récepteur applique un changement immédiatement ; elle est donc relue juste après l'écriture, et cette lecture devient le nouveau point de départ.
- Un poste applique un changement un peu plus tard ; chaque valeur envoyée est donc mémorisée. Quand le poste renvoie cette valeur, elle est reconnue comme une écriture de l'application et non comme une main sur le bouton.
- Une valeur que le poste refuse — la FM large sur un poste HF, par exemple — est envoyée **deux fois** puis laissée tranquille jusqu'à ce que le côté source change, au lieu d'être répétée plusieurs fois par seconde.

Réaccorder la page du récepteur peut lui faire choisir le mode par défaut de la bande (LSB sous 10 MHz, par exemple). Quand c'est le poste qui mène, l'application remet aussitôt le mode du poste, si bien qu'un poste en USB sur 40 m garde le récepteur en USB.

## Quelle fenêtre de récepteur

**Receiver window** choisit quelle station suit le poste :

- **The station window last in front** (par défaut) — avec deux stations ouvertes, cliquez dans l'une et le poste la suit.
- **Une station précise** — le poste y reste attaché, qu'elle soit au premier plan ou non. Si cette station n'est pas ouverte, rien n'est synchronisé jusqu'à son ouverture.

## Cadence de mise à jour

**Update every** fixe la fréquence de lecture des deux côtés : 150 ms, 300 ms (par défaut), 500 ms ou 1 s. Plus rapide paraît plus immédiat au bouton ; plus lent ménage un vieux poste à 4800 ou 9600 bauds, où chaque lecture prend un vrai temps sur la ligne.

## Modes

Mode du poste	Le récepteur écoute en
USB, LSB	USB, LSB
CW	CW
CW-R (inversé)	CW-L
AM, AM synchrone, DSB	AM
FM, FM étroite	FM
FM large	WBFM
RTTY / FSK	LSB
RTTY-R / FSK-R	USB
Modes données (USB-D, DATA-U, PKTUSB, DIG)	USB
Données LSB, données FM	LSB, FM

Mode du récepteur	Le poste est mis en
USB, LSB	USB, LSB
CW	CW
CW-L	CW-R
AM, QUAM	AM
FM	FM
WBFM	WFM — la plupart des postes HF le refusent et sont laissés tranquilles après deux essais
RADE (supérieur / inférieur)	USB / LSB

Un poste en **mode données** y reste : l'USB du récepteur est considéré comme d'accord avec l'USB-D du poste, si bien que le récepteur ne fait jamais sortir le poste du mode données.

## Largeur de filtre

Cochez **Sync filter width** pour aligner les bandes passantes. Des écarts de moins de 60 Hz sont tenus pour égaux, puisque deux filtres n'ont jamais les mêmes pas.

Liaison avec le poste	Largeur de filtre
Hamlib	Oui, là où Hamlib la prend en charge pour ce poste
flrig	Oui
Icom CI-V intégré	Oui — pas de 50 Hz jusqu'à 500 Hz, puis de 100 Hz jusqu'à 3,6 kHz ; AM par pas de 200 Hz jusqu'à 10 kHz ; pas en FM
Elecraft intégré	Oui, par pas de 10 Hz
Kenwood, Yaesu, famille FT-817 intégrés	Non — ces postes choisissent leurs filtres dans des tables propres à chaque modèle. Utilisez Hamlib si vous avez besoin du filtre

Les récepteurs KiwiSDR, WebSDR et UberSDR ont toujours la commande. Un récepteur PhantomSDR-Plus doit avoir la **4.1.0** ou ultérieure ; sur un plus ancien, la fréquence et le mode se synchronisent toujours, et la fenêtre Rig control explique pourquoi le filtre ne suit pas.

## Coupure du son en émission

Cochez **Mute receiver while transmitting**. Tant que le poste est en émission, la fenêtre du récepteur est muette, et le son revient au retour en réception. Le bouton muet du récepteur l'indique, et vous pouvez toujours rétablir le son à la main.

Si vous aviez déjà coupé le son vous-même, il reste coupé ensuite.

Il faut une liaison qui signale l'état d'émission — tous les pilotes intégrés, flrig, et Hamlib pour la plupart des postes — et, sur un récepteur PhantomSDR-Plus, la 4.1.0 ou ultérieure.

## Décalage de fréquence

**Frequency offset** est ajouté à la fréquence du poste pour obtenir celle du récepteur :

fréquence du récepteur = fréquence du poste + décalage

Installation	Décalage
Transverter 2 m sur un poste 10 m (144,100 MHz s'affiche 28,100 MHz)	116000000
Transverter 70 cm sur un poste 2 m (432 → 144)	288000000
Pas de transverter	0

## Le menu Rig

Élément	Rôle
<b>Rig control...</b>	Ouvre la fenêtre Rig control
<b>Connect / Disconnect</b> <i>nom du poste</i>	Démarre ou arrête la synchronisation ; avec Hamlib, démarre ou arrête aussi <code>rigctld</code>
<b>Rig to receiver / Receiver to rig / Both directions</b>	Sens de synchronisation
<b>Sync filter width</b>	Oui / non
<b>Mute receiver while transmitting</b>	Oui / non
Ligne d'état	<i>Not connected, Connecting..., Connected: nom</i> , ou la dernière erreur

L'affichage de fréquence en direct se trouve dans la fenêtre Rig control et non dans le menu, qui sinon se refermerait à chaque changement.

## Linux

**Droits sur le port série.** Les ports série appartiennent au groupe `dialout`. Un utilisateur qui n'en fait pas partie obtient *Could not open ttyUSB0*. Ajoutez-vous une fois, puis déconnectez-vous et reconnectez-vous :

```
sudo usermod -aG dialout $USER
```

**Hamlib.** Installez-le depuis votre distribution :

```
sudo apt install libhamlib-utils
```

Le paquet `.deb` de Desktop PhantomSDR+ le recommande, donc `sudo apt install ./phantomsdr-plus-desktop_4.0.0_amd64.deb` l'installe avec ; `dpkg -i` n'installe pas les paquets recommandés. Les pilotes intégrés et flrig n'ont pas besoin de Hamlib.

## Windows

Le `rigctld.exe` de Hamlib lui-même est inclus dans l'installateur 64 bits comme dans le 32 bits. Les ports COM apparaissent dans la liste sous leur nom (`COM3`). Si le poste demande un pilote USB, installez d'abord celui du constructeur — le port n'existe pas avant.

## TCI-CAT sur la page du récepteur

Une page de récepteur PhantomSDR-Plus peut aussi suivre un transceiver **toute seule**, sans l'application de bureau : le navigateur dialogue directement avec un serveur **TCI** sur votre propre ordinateur. TCI est le protocole de pilotage par WebSocket d'ExpertSDR2/ExpertSDR3 (SunSDR), d'AetherSDR (FlexRadio) et de Thetis (Apache Labs ANAN, Hermes). Pour un poste sans TCI, un petit pont fourni avec PhantomSDR-Plus fait passer n'importe quel poste Hamlib pour un serveur TCI.

Il faut un récepteur PhantomSDR-Plus en version **4.1.0 ou ultérieure**. Les pages KiwiSDR, WebSDR et UberSDR ne l'ont pas.

## Le bouton TCI-CAT

Le bouton se trouve dans la rangée **VFO, Modes, Bands** et **IF Filters** — **TCI-CAT** dans la mise en page large, **CAT** dans la compacte. Son point est **vert** tant qu'un serveur TCI est connecté, **gris** sinon. Il ouvre une fenêtre avec : Jusqu'à la 4.2.0, le bouton s'appelait **QRG Sync**.

Commande	Rôle
État	<i>Active</i> — TCI (port 50001) une fois connecté ; <i>Inactive</i> avec <i>Wait</i> pendant la recherche
<b>CAT Sync</b>	Active ou désactive la synchronisation de la fréquence, du mode et du filtre. La coupure du son en émission fonctionne dans les deux cas
<b>Host</b>	L'ordinateur qui fait tourner le serveur TCI : localhost pour cet ordinateur, sinon son adresse sur le réseau local, par ex. 192.168.1.42. Appuyez sur Entrée ou cliquez ailleurs pour reconnecter

Rien d'autre à choisir : les ports **50001** (ExpertSDR3, AetherSDR) et **40001** (ExpertSDR2, Thetis) sont essayés en même temps et réessayés toutes les quelques secondes, si bien que la page se connecte dès que le logiciel démarre. **CAT Sync** et **Host** sont mémorisés par le navigateur.

Les navigateurs qui demandent l'autorisation avant qu'une page web n'accède au réseau local — les versions récentes de Chrome et d'Edge le font — posent la question une fois, lors de la première connexion. Acceptez, sinon le point reste gris.

## Ce qui est synchronisé

	Comportement
<b>Poste → récepteur : fréquence</b>	Seul le cadran bouge. Zoom, luminosité et contraste restent comme vous les avez laissés ; la cascade ne défile que si la fréquence sort de l'écran, et le plan de bandes ne change pas le mode
<b>Poste → récepteur : mode</b>	Suit quand le mode du poste change — USB, LSB, CW, AM, FM ; les modes numériques en USB ou LSB. Un mode que le récepteur n'a pas le laisse inchangé. Un décodeur en cours garde son propre mode
<b>Poste → récepteur : largeur de filtre</b>	La bande passante du récepteur prend la largeur de filtre du poste : changer le filtre du poste, ou appuyer sur sa touche <b>FIL</b> , change la bande passante. Un décodeur en cours garde sa propre bande passante
<b>Récepteur → poste : fréquence</b>	Taper une fréquence, une étiquette ou un signet, un clic dans la cascade et le glissement de la bande passante déplacent le VFO du poste. Un glissement n'envoie que les fréquences qui changent
<b>Récepteur → poste : mode</b>	Chaque changement de mode sur la page — une touche de mode, le plan de bandes, un décodeur, un signet — règle le mode du poste (USB, LSB, CW, AM, FM ; RADE en USB ou LSB). Un poste en mode numérique (USB-D) y reste
<b>Récepteur → poste : largeur de filtre</b>	Les boutons <b>IF Filters</b> , le curseur IF, le glissement de la bande passante et un changement de mode règlent le filtre du poste ; un glissement n'envoie que la largeur finale. Sur les postes Icom, le pont sélectionne à la place FIL1, FIL2 ou FIL3 — voir <a href="#">Filtres Icom</a>
<b>Pas d'écho</b>	Un changement venu du poste ne lui est jamais renvoyé, et un rapport du mode ou du filtre précédent du poste qui arrive juste après un changement sur la page est ignoré
<b>Coupure du son en émission</b>	Toujours active, même avec CAT Sync désactivé : la page se tait pendant que le poste émet. Déplacer le curseur de volume pendant l'émission la laisse muette ; en réception, le son revient au niveau actuel du curseur
<b>Non géré</b>	Split, VFO B, RIT/XIT et décalages de transverter. Seul le VFO A du premier récepteur est suivi

Le mode et la largeur de filtre passent par les commandes propres au TCI (`modulation`, `rx_filter_band`), donc ExpertSDR, AetherSDR et Thetis devraient les suivre aussi ; seul le pont Hamlib a été essayé. Pour

un décalage de transverter, utilisez plutôt Desktop PhantomSDR+. N'utilisez pas les deux en même temps sur le même poste — deux contrôleurs se disputeraient le cadran.

## Avec quels transceivers cela fonctionne

Transceiver	Fonctionne	Comment
SunSDR (ExpertSDR2/3), FlexRadio (AetherSDR), ANAN/Hermes (Thetis)	Oui	Directement — activez le serveur TCI dans le logiciel. Pas encore essayé avec ces logiciels ; testé avec un serveur TCI simulé
Un poste doté d'un port CAT pris en charge par Hamlib — la plupart des Icom, Yaesu, Kenwood, Elecraft, Xiegu, QRP Labs	Oui	Via le pont Hamlib ci-dessous. Essayé avec un <b>Icom IC-7300</b> sous Linux et Windows ; d'autres pilotes peuvent différer sur les noms de modes ou le signalement du PTT
Un poste sans port CAT, ou non pris en charge par Hamlib	Non	Il n'y a rien où lire la fréquence

## Le pont Hamlib (IC-7300 et autres postes sans TCI)

`tci-bridge/tci-rigctld.mjs`, dans l'arborescence PhantomSDR-Plus, fait de tout poste que Hamlib sait piloter un serveur TCI pour la page. Il tourne sur **votre** ordinateur — celui qui est relié au poste, à côté du navigateur — pas sur le récepteur. Quatre fois par seconde il interroge Hamlib sur la fréquence, le mode, la largeur de filtre et l'état d'émission, n'envoie à la page que ce qui a changé et transmet au poste les changements de fréquence, de mode et de filtre de la page.

Il atteint Hamlib par l'une de deux voies :

Voie	Chaîne	Quand
<code>--rigctl</code> (recommandée)	<code>transceiver → rigctl → tci-rigctld.mjs → page</code>	Normalement. Le pont lance lui-même le <code>rigctl</code> de Hamlib : une fenêtre, aucun port réseau. <b>À utiliser sous Windows</b> , où <code>rigctld.exe</code> est souvent refusé avec <i>Access is denied</i>
<code>rigctld</code>	<code>transceiver → rigctld → tci-rigctld.mjs → page</code>	Un autre programme, comme WSJT-X, doit utiliser le poste en même temps — voir <a href="#">Partager le poste</a>

## Ce qu'il lui faut

	Linux	Windows
Hamlib	<code>sudo apt install libhamlib-utils</code> , ou le paquet hamlib de votre distribution	<code>hamlib-w64-....zip</code> depuis <a href="https://github.com/Hamlib/Hamlib/releases">github.com/Hamlib/Hamlib/releases</a> , extrait dans <code>C:\hamlib</code> — <code>rigctl.exe</code> se trouve dans <code>C:\hamlib\bin</code>
Node.js	18 ou plus récent	Le <i>Windows Installer (.msi)</i> LTS de <a href="https://nodejs.org">nodejs.org</a> , avec les options par défaut
Le paquet ws	Trouvé automatiquement dans l'arborescence PhantomSDR-Plus	Copiez <code>tci-rigctld.mjs</code> dans un dossier comme <code>C:\tci-bridge</code> et lancez-y une fois <code>npm install ws</code>
Le port du poste	<code>ls /dev/serial/by-id/</code> . Rejoignez une fois le groupe <code>dialout</code> : <code>sudo usermod -aG dialout \$USER</code> , puis déconnexion et reconnexion	Installez le pilote USB du fabricant ; le numéro COM figure dans <b>Gestionnaire de périphériques → Ports (COM et LPT)</b>

Un seul programme peut tenir le port CAT du poste. Fermez WSJT-X, flrig, JS8Call, RS-BA1 ou la connexion au poste de Desktop PhantomSDR+ avant de lancer le pont.

## Exemple : Icom IC-7300

Sur le poste : **MENU → SET → Connectors → CI-V — CI-V USB Baud Rate 115200, CI-V Transceiver ON**. Le numéro de modèle Hamlib de l'IC-7300 est 3073.

**Linux.** Le poste est la ligne contenant IC-7300 dans `ls /dev/serial/by-id/`, en général aussi `/dev/ttyUSB0`.

1. Vérifiez que Hamlib atteint le poste — il doit afficher la fréquence, par ex. 14280000 : `bash rigctl -m 3073 -r /dev/ttyUSB0 -s 115200 f`
2. Lancez le pont et laissez-le tourner : `bash cd ~/PhantomSDR-Plus/tci-bridge node tci-rigctld.mjs --rigctl rigctl -m 3073 -r /dev/ttyUSB0 -s 115200`

**Windows.** Le poste apparaît dans le Gestionnaire de périphériques comme *Silicon Labs CP210x USB to UART Bridge (COM4)* — utilisez votre propre numéro COM. Dans une invite de commandes :

1. Vérifiez que Hamlib atteint le poste — il doit afficher la fréquence : `bat C:\hamlib\bin\rigctl.exe -m 3073 -r COM4 -s 115200 f`
2. Lancez le pont et laissez la fenêtre ouverte :  
`bat cd C:\tci-bridge node tci-rigctld.mjs --rigctl C:\hamlib\bin\rigctl.exe -m 3073 -r COM4 -s 115200` La ligne est longue : vérifiez qu'elle se termine bien par `-s 115200`, sinon `rigctl` s'arrête avec *Type: rigctl --help*.

Sur les deux systèmes le pont affiche, en une seconde :

```
rig answered through rigctl
rig -> page 14280000 Hz
rig -> page mode USB
rig -> page RX
```

Ouvrez ensuite la page du récepteur dans un navigateur sur le même ordinateur, ouvrez **TCI-CAT** et activez **CAT Sync**. Le point passe au vert et le pont affiche `page connected`. Tournez le VFO et le récepteur suit ; cliquez dans la cascade et le poste suit (`page -> rig ... Hz`) ; passez en émission et la page se tait. **Ctrl+C** arrête le pont, et `rigctl` avec lui.

Tout ce qui suit `--rigctl` est le programme `rigctl` et ses propres options — exactement celles qui ont fonctionné lors de la vérification. C'est la règle pour tout poste : **quand rigctl ... f affiche la fréquence, le pont fonctionne avec les mêmes options.**

## Filtres Icom (FIL1, FIL2, FIL3)

Les postes Icom comme l'IC-7300 n'acceptent pas n'importe quelle largeur de filtre : ils ont trois filtres, **FIL1**, **FIL2** et **FIL3**, chacun avec une largeur réglée dans le menu du poste. Avec une largeur, Hamlib sélectionnerait l'un d'eux et écraserait aussi sa largeur — et sur l'IC-7300 cette largeur peut atterrir sur le filtre sélectionné auparavant, ce qui mélange les réglages. C'est pourquoi, sur les postes Icom, le pont n'envoie jamais de largeur : il prend le filtre dont la largeur de référence est la plus proche de la bande passante de la page et se contente de le **sélectionner**, avec la commande CI-V du poste lui-même. Les largeurs réglées sur le poste ne changent jamais.

Mode	FIL1	FIL2	FIL3
USB, LSB (et leurs modes numériques)	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

Sur l'IC-7300 ( -m 3073 ) c'est automatique. Réglez les filtres SSB du poste en conséquence — FIL1 2,7 kHz, FIL2 2,4 kHz, FIL3 1,8 kHz (maintenez **FIL** enfoncé sur le poste) — et choisir 2,7 / 2,4 / 1,8 kHz dans **IF Filters** sélectionne FIL1 / FIL2 / FIL3, tandis qu'appuyer sur **FIL** sur le poste met la bande passante de la page à la largeur de ce filtre. Le pont affiche par exemple `page -> rig filter 2398 Hz LSB -> FIL2 sent`.

Option	Usage
<code>--filters 3000,2400,1800</code>	D'autres largeurs de référence SSB, dans l'ordre FIL1,FIL2,FIL3 ; active aussi la sélection de filtre pour un autre poste Icom
<code>--civ A4</code>	L'adresse CI-V du poste en hexadécimal, quand ce n'est pas 94 (IC-705 A4, IC-9700 A2, IC-7610 98)
<code>--filters off</code>	Envoyer les largeurs par Hamlib, comme pour les autres marques

Les deux options se placent avant `--rigctl`, par ex. `node tci-rigctld.mjs --filters 3000,2400,1800 --civ A4 --rigctl rigctl -m 3085 -r /dev/ttyACM0 -s 19200`. Essayé uniquement avec un IC-7300. Les autres marques — Yaesu, Kenwood, Elecraft et les autres — reçoivent la largeur de la page par Hamlib, qui la règle d'après près que le poste le permet.

## Autres postes

Les mêmes étapes valent pour tout poste pris en charge par Hamlib ; seules les options changent.

1. **Préparez le poste.** Dans son menu, notez la vitesse CAT (ou CI-V) et activez un réglage comme *CAT par USB* ou *CI-V transceive* s'il en a un. Sous Windows, installez le pilote USB du fabricant.
2. **Trouvez le numéro de modèle Hamlib** dans la liste affichée par `rigctl -l` :
3. Linux : `rigctl -l | grep -i 991`
4. Windows : `C:\hamlib\bin\rigctl.exe -l | findstr /i 991`
5. **Trouvez le port.** Linux : `ls /dev/serial/by-id/`. Windows : Gestionnaire de périphériques. Certains postes créent **deux** ports — les FT-991A, FTDX10 et FT-710 de Yaesu les nomment *Enhanced* et *Standard* ; le CAT est sur le port **Enhanced**.
6. **Vérifiez, puis lancez le pont** avec `-m <modèle> -r <port> -s <vitesse>`, comme dans l'exemple de l'IC-7300 : d'abord `rigctl -m ... -r ... -s ... f`, puis les mêmes options après `--rigctl`.

Quelques numéros de modèle, d'après Hamlib 4.5 — confirmez-les avec `rigctl -l`, car une autre version de Hamlib peut numéroter un poste différemment :

Poste	-m	Poste	-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

Options supplémentaires, seulement si nécessaire :

- **Un poste Icom dont l'adresse CI-V a été changée** : ajoutez `-c` et l'adresse en *décimal* — 94h donne `-c 148`.
- **Un réglage que Hamlib propose pour ce poste** : `rigctl -m <modèle> -L` les liste ; fixez-en un avec `-C nom=valeur`, par ex. `-C post_write_delay=10` pour une interface lente.
- **Un poste qui passe en émission ou redémarre à l'ouverture du port** : certains câbles CAT utilisent DTR ou RTS pour le PTT ; ajoutez `-C dtr_state=0FF -C rts_state=0FF`.

### Exemple : Yaesu FT-991A

Non essayé avec un vrai FT-991A — il suit les réglages de Hamlib pour ce poste ; la vérification `rigctl ... f` indique aussitôt si cela fonctionne.

Sur le poste, réglez **CAT RATE** (menu 031) sur 38400. Le numéro de modèle Hamlib des FT-991 et FT-991A est 1035.

Le câble USB du FT-991A crée **deux** ports série. Le CAT est sur le port **Enhanced** :

- **Linux** : `ls /dev/serial/by-id/` affiche deux lignes pour le poste ; celle qui se termine par `-if00-port0` est l'Enhanced, en général `/dev/ttyUSB0`.
- **Windows** : le Gestionnaire de périphériques affiche *Silicon Labs Dual CP2105 USB to UART Bridge: Enhanced COM Port (COM5)* et un *Standard COM Port* ; utilisez le numéro de l'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 vérification n'affiche rien d'utile, la cause habituelle est le port Standard au lieu de l'Enhanced, ou un CAT RATE différent de `-s`.

### Exemple : Kenwood TS-590SG

Non essayé avec un vrai TS-590SG — il suit les réglages de Hamlib pour ce poste ; la vérification `rigctl ... f` indique aussitôt si cela fonctionne.

Dans le menu du poste, réglez la vitesse du port **USB** sur 115200 (le numéro de menu figure dans le manuel du TS-590SG). Le numéro de modèle Hamlib est 2037 ; l'ancien TS-590S est 2031. Sous Windows, installez d'abord le pilote de port COM virtuel de Kenwood pour le port USB.

Le câble USB crée **un** port série :

- **Linux** : la ligne du poste dans `ls /dev/serial/by-id/`, en général `/dev/ttyUSB0`.
- **Windows** : Gestionnaire de périphériques → Ports (COM et LPT), par exemple `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 vérification échoue, la cause habituelle est une vitesse USB différente de `-s`, ou un câble branché sur la prise RS-232 (COM) du poste alors que `-r` désigne le port USB.

### Autres exemples

Aucun n'a été essayé sur du matériel réel. Chaque ligne donne les options à placer après `rigctl` pour la vérification et après `--rigctl rigctl` pour le pont ; réglez la même vitesse dans le menu du poste. Le port Linux est le port habituel — confirmez-le avec `ls /dev/serial/by-id/`. Sous Windows, remplacez le port par le numéro COM du Gestionnaire de périphériques et `rigctl` par `C:\hamlib\bin\rigctl.exe`.

Poste	Menu du poste	Options (Linux)	Remarques
Icom IC-705 (USB)	CI-V USB Baud Rate 19200	-m 3085 -r /dev/ ttyACM0 -s 19200	Deux ports apparaissent ; le CI-V est le premier (ttyACM0)
Icom IC-7100 (USB)	CI-V USB Baud Rate 19200	-m 3070 -r /dev/ ttyUSB0 -s 19200	La vitesse maximale de Hamlib pour ce poste est 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 pour le FTDX101MP. Deux ports ; utilisez l'Enhanced
Yaesu FTDX10	CAT RATE 38400	-m 1042 -r /dev/ ttyUSB0 -s 38400	Deux ports ; utilisez l'Enhanced, comme sur le FT-991A
Yaesu FT-710	CAT RATE 38400	-m 1049 -r /dev/ ttyUSB0 -s 38400	Deux ports ; utilisez l'Enhanced, comme sur le FT-991A
Yaesu FT-891	CAT RATE 38400	-m 1036 -r /dev/ ttyUSB0 -s 38400	Deux ports ; utilisez l'Enhanced, comme sur le FT-991A
Yaesu FT-450D	CAT RATE 38400	-m 1046 -r /dev/ ttyUSB0 -s 38400	Prise RS-232 : utilisez un adaptateur USB-série
Yaesu FT-817 / FT-818	CAT RATE 38400	-m 1020 -r /dev/ ttyUSB0 -s 38400	-m 1041 pour le FT-818. Demande un câble CAT sur la prise ACC
Yaesu FT-857 / FT-897	CAT RATE 38400	-m 1022 -r /dev/ ttyUSB0 -s 38400	-m 1023 pour le FT-897. Demande un câble CAT
Kenwood TS-890S (USB)	Vitesse USB 115200	-m 2041 -r /dev/ ttyUSB0 -s 115200	
Kenwood TS-990S (USB)	Vitesse USB 115200	-m 2039 -r /dev/ ttyUSB0 -s 115200	
Kenwood TS-590SG / TS-590S (USB)	Vitesse USB 115200	-m 2037 -r /dev/ ttyUSB0 -s 115200	Exemple détaillé ci-dessus
Kenwood TS-480	Vitesse du port COM 57600	-m 2028 -r /dev/ ttyUSB0 -s 57600	Prise RS-232 : utilisez un adaptateur USB-série
Kenwood TS-2000	Vitesse du port COM 57600	-m 2014 -r /dev/ ttyUSB0 -s 57600	Prise RS-232 ; la vitesse maximale de Hamlib pour ce poste est 57600
Elecraft K4 (USB)	Vitesse RS232 115200	-m 2047 -r /dev/ ttyUSB0 -s 115200	
Elecraft K3 / K3S	Vitesse RS232 38400	-m 2029 -r /dev/ ttyUSB0 -s 38400	K3S : USB ; K3 : port série ou câble KUSB
Elecraft KX3	Vitesse RS232 38400	-m 2045 -r /dev/ ttyUSB0 -s 38400	Câble KXUSB
Elecraft KX2	Vitesse RS232 38400	-m 2044 -r /dev/ ttyUSB0 -s 38400	Câble KXUSB
Xiegu G90	Vitesse CI-V 19200	-m 3088 -r /dev/ ttyUSB0 -s 19200	Hamlib utilise l'adresse CI-V par défaut du G90

Poste	Menu du poste	Options (Linux)	Remarques
Xiegu X6100	Vitesse CI-V 19200	-m 3087 -r /dev/ ttyUSB0 -s 19200	La vitesse maximale de Hamlib pour ce poste est 19200
Lab599 TX-500	—	-m 2050 -r /dev/ ttyUSB0 -s 9600	Hamlib utilise uniquement 9600
ELAD FDM-DUO	—	-m 33001 -r /dev/ ttyUSB0 -s 115200	
QRP Labs QDX	—	-m 2052 -r /dev/ ttyACM0 -s 9600	Un port série USB ; la vitesse ne compte pas mais doit être indiquée

Commande Linux complète pour l'IC-705, pour montrer comment lire une ligne :

```
rigctl -m 3085 -r /dev/ttyACM0 -s 19200 f
node tci-rigctld.mjs --rigctl rigctl -m 3085 -r /dev/ttyACM0 -s 19200
```

### Postes pilotés par un autre logiciel

Certains postes sont déjà pilotés par un logiciel capable de faire le travail — parfois sans aucun pont. Aucune de ces voies n'a été essayée.

Poste et logiciel	Que faire
<b>SunSDR</b> avec ExpertSDR2/3, <b>FlexRadio</b> avec AetherSDR, <b>Apache Labs ANAN</b> / <b>Hermes</b> avec Thetis	Pas de pont. Activez le <b>serveur TCI</b> du logiciel et utilisez <b>TCI-CAT</b> directement — les ports 50001 et 40001 sont trouvés automatiquement
<b>FlexRadio</b> avec SmartSDR (Windows)	Ajoutez un port dans <b>SmartSDR CAT</b> . Il parle le CAT Kenwood, donc un port série y fonctionne comme un TS-2000 : -m 2014 -r COM8 -s 57600. Un port TCP fonctionne avec le modèle FlexRadio de Hamlib : -m 2036 -r 127.0.0.1:<port>
<b>Tout poste piloté par flrig</b>	Laissez flrig tourner et utilisez le modèle flrig de Hamlib ; le poste reste partagé avec fldigi, WSJT-X et les logiciels de trafic : -m 4 -r 127.0.0.1:12345
<b>Un rigctld en cours d'exécution</b> , ou un logiciel offrant un serveur <i>Hamlib NET rigctl</i>	Lancez le pont sans --rigctl, avec cette adresse : node tci-rigctld.mjs 50001 127.0.0.1:4532

Comme partout, les options se placent après `rigctl` pour la vérification et après `--rigctl rigctl` pour le pont. Une adresse réseau comme `127.0.0.1:12345` n'a pas besoin de `-s`.

### Partager le poste : la voie rigctld

Quand WSJT-X ou un logiciel de trafic doit utiliser le poste pendant que le pont tourne, lancez le serveur de Hamlib `rigctld` avec les mêmes options, laissez-le tourner et reliez les deux programmes à lui :

```
rigctld -m 3073 -r /dev/ttyUSB0 -s 115200
node tci-rigctld.mjs
```

Sans `--rigctl`, le pont cherche `rigctld` sur `127.0.0.1:4532` et affiche `rigctld connected` ; `node tci-rigctld.mjs 50001 192.168.1.50:4532` utilise un `rigctld` sur un autre ordinateur. Dans WSJT-X, choisissez le poste *Hamlib NET rigctl* à la même adresse. Sous Windows le serveur est `C:\hamlib\bin\rigctld.exe` avec les mêmes options ; si Windows répond *Access is denied*, utilisez `--rigctl` et fermez l'autre programme pendant que le pont tourne.

**Ports.** Le pont sert le TCI sur le port 50001 ; un autre port se place en premier sur la ligne de commande, par ex. `node tci-rigctld.mjs 40001 --rigctl ...`. Si le pont et le navigateur sont sur des ordinateurs différents, indiquez dans **Host** l'adresse de l'ordinateur du pont.

## Dépannage de TCI-CAT

Symptôme	Cause probable	Que faire
Le point reste gris	Serveur TCI ou pont arrêté ; <b>Host</b> erroné ; l'autorisation réseau local du navigateur a été refusée	Activez TCI dans ExpertSDR/Thetis/AetherSDR ou lancez le pont ; vérifiez <b>Host</b> ; autorisez l'accès au réseau local dans les paramètres du site
<code>rigctl ... f</code> renvoie une erreur ou reste en attente	Mauvais port ou mauvaise vitesse ; pas de droit <code>diagnostics</code> (Linux) ou pas de pilote USB (Windows) ; un autre programme tient le port	Alignez <code>-s</code> sur le menu du poste ; vérifiez le port ( <code>ls /dev/serial/by-id/</code> , Gestionnaire de périphériques) ; fermez les autres programmes de pilotage
Le pont affiche <i>rigctl stopped ... Type: rigctl --help</i>	La commande est incomplète — en général la vitesse après <code>-s</code> manque	Retapez la ligne entière
Le pont affiche <i>The value after -s is missing</i>	La ligne de commande a été coupée au collage	Retapez la fin de la ligne, par ex. <code>-s 115200</code>
Le filtre ne suit pas dans un sens ou dans l'autre	<b>CAT Sync</b> est désactivé, un décodeur tourne sur la page, ou (IC-7300) les largeurs FIL du poste différent de 2700 / 2400 / 1800 Hz	Activez CAT Sync ; arrêtez le décodeur ; réglez les largeurs FIL sur le poste, ou indiquez <code>--filters</code> avec les largeurs du poste
Le pont affiche <i>rigctl stopped ... rig_open: error</i>	<code>rigctl</code> ne peut pas ouvrir le port	Comme pour <code>rigctl ... f</code> ci-dessus
Le pont affiche <i>the rig is not answering</i>	Le port s'ouvre mais le poste ne répond pas : mauvaise vitesse ou mauvais modèle, CAT désactivé dans le menu, ou adresse CI-V Icom changée	Refaites la vérification <code>rigctl ... f</code> ; ajoutez <code>-c</code> si l'adresse CI-V a été changée
Le pont affiche <i>rigctld not reachable</i>	Lancé sans <code>--rigctl</code> , et aucun <code>rigctld</code> ne tourne	Ajoutez <code>--rigctl ...</code> , ou lancez d'abord <code>rigctld</code>
<code>rigctld.exe</code> répond <i>Access is denied</i> (Windows)	Windows refuse de l'exécuter	Utilisez <code>--rigctl</code> — il ne demande que <code>rigctl.exe</code>
Connecté, mais le récepteur ne bouge pas	<b>CAT Sync</b> est désactivé	Activez-le — la coupure en émission fonctionne sans lui
Le mode du poste ne suit pas	Mode sans équivalent sur le récepteur, ou pilote Hamlib qui renvoie un nom inhabituel	La fréquence reste synchronisée ; réglez le mode sur la page
Pas de coupure en émission	Le poste ou son pilote Hamlib ne signale pas l'état d'émission	Rien à régler sur la page
Le cadran fait des allers-retours	Desktop PhantomSDR+ ou un autre programme synchronise aussi le poste	Un seul contrôleur à la fois

## Pour les opérateurs de récepteurs

Rien à configurer. Le pilotage du transceiver passe par une petite interface JavaScript que chaque page PhantomSDR-Plus possède déjà ; il ne demande ni réglage serveur, ni port ouvert, ni droit d'administration. Les fonctions de filtre et de coupure du son sont arrivées avec la version 4.1.0 — après l'avoir appliquée, recompilez le frontend ( `./recompile.sh`, option 2 ) ; inutile d'arrêter le récepteur. Les récepteurs KiwiSDR, WebSDR et UberSDR n'ont besoin de rien non plus : l'application utilise les commandes que leurs pages possèdent déjà.

Le bouton **TCI-CAT** est arrivé lui aussi avec la 4.1.0, là encore une simple recompilation du frontend. Il n'ouvre aucun port sur le récepteur : la connexion va du navigateur de chaque auditeur vers son propre ordinateur. Le pont Hamlib, `tci-bridge/tci-rigctld.mjs`, est destiné aux auditeurs, chez eux ; le récepteur ne l'utilise pas.

## Pour les développeurs : l'interface de la page

Desktop PhantomSDR+ comme le CATsync Tool utilisent ces fonctions, que chaque page PhantomSDR-Plus place sur `window` une fois chargée (les pages KiwiSDR, WebSDR et UberSDR sont pilotées par leurs propres commandes, différentes) :

Fonction	Renvoi / fait
<code>catsync_ready</code>	<code>true</code> dès que les fonctions ci-dessous sont installées
<code>catsync_getFrequency()</code>	Fréquence d'accord, Hz
<code>catsync_setFrequency(hz)</code>	S'accorde sur <code>hz</code>
<code>catsync_getMode()</code>	USB, LSB, CW, CW-L, AM, QUAM, FM, WBFM, RADEU, RADEL
<code>catsync_setMode(mode)</code>	Règle le mode ; ramène la bande passante à la valeur par défaut du mode
<code>catsync_getBandwidth()</code>	Largeur totale de la bande passante, Hz
<code>catsync_setBandwidth(hz)</code>	Règle la largeur — s'élargit vers le haut en USB, vers le bas en LSB, également des deux côtés sinon. À appeler <b>après</b> <code>catsync_setMode</code>
<code>catsync_getMute()</code>	<code>true</code> si le son est coupé
<code>catsync_setMute(on)</code>	Coupe ou rétablit le son, via le bouton muet de la page

Les quatre dernières sont arrivées avec la 4.1.0 ; testez donc avant de les appeler :

```
if (window.catsync_ready) {
 window.catsync_setFrequency(7074000)
 window.catsync_setMode('USB')
 if (typeof window.catsync_setBandwidth === 'function') window.catsync_setBandwidth(2400)
}
```

Régler la fréquence réaccorde l'audio : n'appellez un setter que lorsque la valeur a réellement changé — un setter appelé en boucle avec la même valeur s'entend. Les anciens points d'entrée de style KiwiSDR/ WebSDR (`setfreq`, `set_mode`, `freqset_complete`) restent disponibles pour les outils qui les attendent.

## Dépannage

Symptôme	Cause probable	Que faire
<i>Could not open ttyUSB0</i> (Linux)	Pas dans le groupe <code>dialout</code> , ou un autre programme tient le port	<code>sudo usermod -aG dialout \$USER</code> , se déconnecter et se reconnecter ; fermer WSJT-X, carnets de trafic, utilitaires du poste
L'affichage du récepteur indique <i>not a receiver this app can drive</i>	Un autre type de récepteur web, ou la page est encore en chargement	Sont pris en charge PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR et UberSDR ; laisser quelques secondes à une page lente
<i>The rig did not answer</i>	Mauvaise vitesse, mauvais type de poste, mauvaise adresse CI-V, poste éteint	Aligner la vitesse sur le menu du poste ; vérifier l'adresse CI-V ; essayer Hamlib
<i>Hamlib is not installed</i>	Aucun <code>rigctld</code> trouvé	Linux : <code>sudo apt install libhamlib-utils</code> . Ou indiquer son chemin dans <b>rigctld program</b>
<i>rigctld stopped: ...</i>	Hamlib n'a pas pu ouvrir le poste — son propre message suit	Généralement le port ou la vitesse ; le texte après les deux-points est la raison donnée par Hamlib
<i>flrig is not running at ...</i>	flrig fermé, ou son port XML-RPC diffère	Lancer flrig ; vérifier le port dans sa configuration
Connecté, mais le récepteur ne bouge pas	Aucune fenêtre de station ouverte, ou <b>Receiver window</b> attaché à une station fermée	Ouvrir la station, ou choisir <i>The station window last in front</i>
Le poste passe en émission à la connexion	DTR ou RTS déclenche le poste via votre interface	Décocher <b>DTR on</b> et <b>RTS on</b>
Le filtre ne suit pas	Récepteur antérieur à la 4.1.0, ou pilote intégré Kenwood/Yaesu	Fréquence et mode se synchronisent toujours ; utiliser Hamlib pour le filtre sur Kenwood/Yaesu
La coupure en émission ne fait rien	Récepteur sans la mise à jour, ou le poste ne signale pas l'état d'émission	Comme ci-dessus
<i>Lost the rig ... reconnecting</i>	Câble débranché, poste éteint ou <code>rigctld</code> arrêté	Rien — nouvel essai toutes les 3 secondes, puis reprise dès le retour du poste
Les deux côtés sautent sans arrêt	Deux programmes pilotent le poste en même temps	Ne laisser qu'un programme régler le poste, ou le partager via flrig

## Limites connues

- Un poste, une fenêtre de récepteur à la fois.
- Les récepteurs autres que PhantomSDR-Plus, KiwiSDR, PA3FWM WebSDR et UberSDR — OpenWebRX, par exemple — ne sont pas pris en charge.
- Les pilotes intégrés suivent les protocoles publiés par les constructeurs et ont été testés contre des postes simulés et un vrai Hamlib ; pour un poste qui se comporte autrement, Hamlib est la solution de repli.
- Le split, le VFO B, le RIT/XIT et les canaux mémoire ne sont pas synchronisés — seulement la fréquence du VFO actif.
- TCI-CAT sur la page du récepteur ne synchronise ni le split, ni le VFO B, ni le RIT/XIT, ni les décalages de transverter, et n'a été essayé qu'avec un IC-7300 via le pont Hamlib.

- Sous Linux, les paquets utilisent le Hamlib de la distribution ; aucun n'est embarqué.

## CHAPITRE 16

# Structure du projet

---

PROJECT\_STRUCTURE.md

## Structure du projet PhantomSDR-Plus

Ce document offre une vue d'ensemble complète de la structure de répertoires de PhantomSDR-Plus, de l'organisation des fichiers et des relations entre composants.

---

### Table des matières

---

1. Arborescence des répertoires
  2. Répertoire racine
  3. Code source ( `src/` )
  4. Frontend ( `frontend/` )
  5. Listes de fréquences ( `frequencylist/` )
  6. Fichiers de configuration
  7. Système de compilation
-

## Arborescence des répertoires

---

```

PhantomSDR-Plus
├─ add-receiver.sh # ajoute un récepteur de plus : pilote, instances/<nom>/,
receivers.toml – voir 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 : garde I du CF32 de rx_sdr, mis à l'échelle en s16
(compilé par setup-fobos.sh)
├─ chat_history.txt
├─ check-go.sh # ancien : chien de garde de la chaîne 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 # absent du dépôt : extrait de a26allx2.zip par update-markers.sh
│ ├── a26allx2.zip
│ ├── admin.txt
│ ├── antenna.txt
│ ├── broadcas.txt
│ ├── curl-output.txt
│ ├── currentUpdateFile.txt
│ ├── fmorg.txt
│ ├── generate-current-shortwave.py
│ ├── language.txt
│ ├── mymarkers.json
│ ├── README.md
│ ├── shortwavestations.json
│ ├── site.txt
│ └── update-markers.sh
├── frontend
│ ├── build-all.sh
│ ├── build-default.sh
│ ├── build-mobile.sh
│ ├── debug-title.sh
│ ├── favicon.ico
│ ├── fix-title-python.py
│ ├── index.html
│ └── jsconfig.json

```

```

├─ LICENSE
├─ make-redirect-stubs.sh
├─ mobile
│ └─ index.html
├─ package.json
├─ package-lock.json
├─ pnpm-lock.yaml
├─ postcss.config.cjs
├─ .prettierrc.json # règles de formatage Prettier pour les sources du frontend
├─ public
│ ├── analyze_users.py
│ ├── decoders
│ │ ├── ft8_lib.wasm
│ │ ├── js8_dict.bin # dictionnaire de mots JS8, chargé seulement au besoin
│ │ └─ js8.wasm # décodeur JS8, compilé depuis 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 # décodeur FLAC qui joue chaque paquet dès son arrivée
│ ├── fsk.js
│ ├── fsk.worker.js
│ ├── fskWorkerProxy.js
│ ├── kiwiSource.js
│ ├── lib
│ │ ├── backend.js
│ │ ├── BandSelector.svelte
│ │ ├── catsync.js
│ │ ├── chatReply.js # marqueur et fils des réponses du 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 # données du sélecteur de récepteur (/receivers.json)
├── rnnnoise.js # charge RNNNoise pour la réduction de bruit par IA
├── rx.js # à quel récepteur appartient une page (?rx=)
├── SMeterAnalog.svelte
├── SMeterDigital.svelte
├── Spectrogram.svelte
├── StatusIndicators.svelte
├── storage.js
├── sUnits.js # échelle des points S selon la fréquence (S9 = -93 dBm au-
dessus 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 # mesure la connexion, garde petit le tampon de lecture
├── psk31.js
├── refused.js
├── remoteSource.js
├── scanner.js
├── siteInfo.js # site_information.json, recouvert par la copie propre au

```

```

récepteur
├── 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 # ancienne chaîne de lancement, remplacée par 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 # récepteurs supplémentaires, un dossier chacun (hors git) – voir
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 # décodeur JS8 en C, compilé en WebAssembly, avec ses tests et son
banc de référence – voir son 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 # ancien : tue les processus du serveur, appelé par go.sh
├── kiwi_install.sh # installe la passerelle KiwiSDR dans une arborescence existante
├── LICENSE
├── logproxy # copies tournantes des journaux panneau/mandataire/autorun
├── logrotate
├── └── phantomsdr # configuration logrotate pour proxy.log + admin.log (adapter les

```

```

chemins avant installation)
├─ manage_admin.sh
├─ markers.json
├─ meson.build
├─ meson_options.txt
├─ phantom_fftw_wisdom
├─ phantomsdr-admin.service # unité systemd d'exemple pour le panneau (démarrage au boot,
redémarrage après plantage)
├─ phantomsdr-proxy.service # unité systemd d'exemple pour le proxy, à installer avec celle du
panneau
├─ phantomsdr-websdr-relay.service # unité systemd d'exemple pour le relais de diversité WebSDR
├─ proxy.py
├─ rade_helper.py
├─ rade_loadtest.csv # sortie de rade_loadtest.py (une ligne par palier de charge)
├─ rade_loadtest.py # test de charge RADE : combien de décodeurs simultanés la machine
supporte – voir docs/RADE_README.md
├─ rade.sh
├─ README.md
├─ recompile.sh
├─ receivers.toml.example # liste des récepteurs pour proxy.py – à copier en receivers.toml
├─ _relaunch.sh # ancien : relance différée de la chaîne go.sh
├─ request.hpp
├─ setup_admin.sh
├─ setup-airspyhf.sh # chaîne de pilote Airspy HF+ : libairspyhf + SoapyAirspyHF + rx_sdr
+ udev
├─ setup-cpufreq-perms.sh # donne au groupe l'écriture sur scaling_max_freq pour que la
protection puisse brider sans root
├─ setup-firewall.sh # garde nftables optionnelle – voir docs/CONNECTION_LIMITS.md
├─ setup-fobos.sh # chaîne de pilote Fobos : libfobos + SoapyFobosSDR + rx_sdr +
cf32_to_real + udev
├─ setup-hackrf.sh # HackRF : paquet hackrf de la distribution + udev (rien de compilé)
├─ setup-rspla.sh # chaîne de pilote RSP1A : libmirisdr-5 + SoapyMiri + rx_sdr + liste
noire msi2500 + udev
├─ setup-rtlsdr.sh # pilote RTL-SDR : paquet de la distribution, ou pilote Blog V4
compilé + blocage DVB + udev
├─ setup-rx888-udev.sh
├─ setup-sdr-common.sh # fonctions communes des trois scripts setup ci-dessus
├─ setup_websdr_relay.sh # installe le relais de diversité WebSDR (port, identité, 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 # passerelle du protocole KiwiSDR – voir 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 # démarre chaque récepteur de receivers.toml
├── start-fobos-hf.sh
├── start-fobos.sh
├── start-hackrf.sh
├── start-rspla.sh
├── start-rtl.sh
├── start-rx888mk2.sh
├── stop-websdr.sh
├── subprojects
│ ├── fftw3.wrap
│ ├── flac.wrap
│ ├── glaze.wrap
│ ├── libcds.wrap
│ ├── libflac.wrap
│ ├── libvolk.wrap
│ ├── ogg.wrap
│ ├── opus.wrap
│ ├── tomlplusplus-3.4.0
│ │ ├── CHANGELOG.md
│ │ ├── cmake
│ │ │ ├── install-rules.cmake
│ │ │ ├── project-is-top-level.cmake
│ │ │ ├── tomlplusplusConfig.cmake
│ │ │ ├── tomlplusplusConfig.cmake.meson.in
│ │ │ ├── tomlplusplusConfigVersion.cmake.meson.in
│ │ │ └── variables.cmake
│ │ ├── CMakeLists.txt
│ │ ├── CODE_OF_CONDUCT.md
│ │ ├── CONTRIBUTING.md
│ │ ├── cpp.hint
│ │ └── docs
│ │ ├── images
│ │ │ ├── badge-awesome.svg
│ │ │ ├── badge-C++17.svg
│ │ │ ├── badge-gitter.svg
│ │ │ ├── badge-license-MIT.svg
│ │ │ ├── badge-TOML.svg
│ │ │ ├── banner.ai
│ │ │ ├── banner.png
│ │ │ ├── banner.svg
│ │ │ ├── favicon.ico
│ │ │ ├── logo.ai
│ │ │ └── logo.svg
│ │ └── pages
│ │ └── main_page.md
│ └── poxy.toml
├── examples
├── benchmark_data.toml

```

```

├─ CMakeLists.txt
├─ error_printer.cpp
├─ error_printer.vcxproj
├─ examples.hpp
├─ example.toml
├─ merge_base.toml
├─ merge_overrides.toml
├─ meson.build
├─ parse_benchmark.cpp
├─ parse_benchmark.vcxproj
├─ simple_parser.cpp
├─ simple_parser.vcxproj
├─ toml_generator.cpp
├─ toml_generator.vcxproj
├─ toml_merger.cpp
├─ toml_merger.vcxproj
├─ toml_to_json_transcoder.cpp
├─ toml_to_json_transcoder.vcxproj
├─ include
├─ meson.build
├─ toml++
├─ impl
├─ array.hpp
├─ array.inl
├─ at_path.hpp
├─ at_path.inl
├─ date_time.hpp
├─ formatter.hpp
├─ formatter.inl
├─ forward_declarations.hpp
├─ header_end.hpp
├─ header_start.hpp
├─ json_formatter.hpp
├─ json_formatter.inl
├─ key.hpp
├─ make_node.hpp
├─ node.hpp
├─ node.inl
├─ node_view.hpp
├─ parse_error.hpp
├─ parse_result.hpp
├─ parser.hpp
├─ parser.inl
├─ path.hpp
├─ path.inl
├─ preprocessor.hpp
├─ print_to_stream.hpp
├─ print_to_stream.inl
├─ simd.hpp
├─ source_region.hpp
├─ std_except.hpp
├─ std_initializer_list.hpp
├─ std_map.hpp
├─ std_new.hpp
├─ std_optional.hpp
├─ std_string.hpp
├─ std_string.inl
├─ std_utility.hpp
├─ std_variant.hpp
├─ std_vector.hpp
├─ table.hpp
├─ table.inl
├─ toml_formatter.hpp
├─ toml_formatter.inl
├─ unicode_autogenerated.hpp

```

```

├── unicode.hpp
├── unicode.inl
├── value.hpp
├── version.hpp
├── yaml_formatter.hpp
├── yaml_formatter.inl
├── toml.h
├── toml.hpp
├── LICENSE
├── meson.build
├── meson_options.txt
├── README.md
├── src
│ ├── meson.build
│ └── toml.cpp
├── tests
│ ├── at_path.cpp
│ ├── conformance_burntsushi_invalid.cpp
│ ├── conformance_burntsushi_valid.cpp
│ ├── conformance_iarna_invalid.cpp
│ ├── conformance_iarna_valid.cpp
│ ├── cpp.hint
│ ├── for_each.cpp
│ ├── formatters.cpp
│ ├── impl_toml.cpp
│ ├── leakproof.hpp
│ ├── lib_catch2.hpp
│ ├── main.cpp
│ ├── manipulating_arrays.cpp
│ ├── manipulating_parse_result.cpp
│ ├── manipulating_tables.cpp
│ ├── manipulating_values.cpp
│ ├── meson.build
│ ├── odr_test_1.cpp
│ ├── odr_test_2.cpp
│ ├── parsing_arrays.cpp
│ ├── parsing_booleans.cpp
│ ├── parsing_comments.cpp
│ ├── parsing_dates_and_times.cpp
│ ├── parsing_floats.cpp
│ ├── parsing_integers.cpp
│ ├── parsing_key_value_pairs.cpp
│ ├── parsing_spec_example.cpp
│ ├── parsing_strings.cpp
│ ├── parsing_tables.cpp
│ ├── path.cpp
│ ├── settings.hpp
│ ├── tests.cpp
│ ├── tests.hpp
│ ├── user_feedback.cpp
│ ├── using_iterators.cpp
│ ├── visit.cpp
│ └── vs
│ ├── odr_test.vcxproj
│ ├── test_debug_x64_cpplatest_noexcept_unrel.vcxproj
│ ├── test_debug_x64_cpplatest_noexcept.vcxproj
│ ├── test_debug_x64_cpplatest_unrel.vcxproj
│ ├── test_debug_x64_cpplatest.vcxproj
│ ├── test_debug_x64_noexcept_unrel.vcxproj
│ ├── test_debug_x64_noexcept.vcxproj
│ ├── test_debug_x64_unrel.vcxproj
│ ├── test_debug_x64.vcxproj
│ ├── test_debug_x86_cpplatest_noexcept_unrel.vcxproj
│ ├── test_debug_x86_cpplatest_noexcept.vcxproj
│ └── test_debug_x86_cpplatest_unrel.vcxproj

```

```

├── test_debug_x86_cpplatest.vcxproj
├── test_debug_x86_noexcept_unrel.vcxproj
├── test_debug_x86_noexcept.vcxproj
├── test_debug_x86_unrel.vcxproj
├── test_debug_x86.vcxproj
├── test_release_x64_cpplatest_noexcept_unrel.vcxproj
├── test_release_x64_cpplatest_noexcept.vcxproj
├── test_release_x64_cpplatest_unrel.vcxproj
├── test_release_x64_cpplatest.vcxproj
├── test_release_x64_noexcept_unrel.vcxproj
├── test_release_x64_noexcept.vcxproj
├── test_release_x64_unrel.vcxproj
├── test_release_x64.vcxproj
├── test_release_x86_cpplatest_noexcept_unrel.vcxproj
├── test_release_x86_cpplatest_noexcept.vcxproj
├── test_release_x86_cpplatest_unrel.vcxproj
├── test_release_x86_cpplatest.vcxproj
├── test_release_x86_noexcept_unrel.vcxproj
├── test_release_x86_noexcept.vcxproj
├── test_release_x86_unrel.vcxproj
├── test_release_x86.vcxproj
├── windows_compat.cpp
├── toml++.code-workspace
├── toml.hpp
├── toml++.natvis
├── toml++.props
├── toml++.sln
├── toml-test
│ ├── meson.build
│ ├── README.md
│ ├── tt_decoder.cpp
│ ├── tt_decoder.vcxproj
│ ├── tt_encoder.cpp
│ ├── tt_encoder.vcxproj
│ └── tt.hpp
├── toml++.vcxproj
├── toml++.vcxproj.filters
├── tools
│ ├── ci_single_header_check.py
│ ├── clang_format.bat
│ ├── generate_conformance_tests.py
│ ├── generate_single_header.bat
│ ├── generate_single_header.py
│ ├── generate_windows_test_targets.py
│ ├── requirements.txt
│ ├── utils.py
│ └── version.py
├── vendor
│ ├── catch.hpp
│ ├── json.hpp
│ └── README.md
├── tomlplusplus.wrap
├── websocketpp.wrap
├── zlib.wrap
├── zstd.wrap
├── tci-bridge
│ └── tci-rigctld.mjs # serveur TCI pour postes Hamlib (IC-7300...), tourne sur le PC de
 # l'auditeur – voir docs/RIG_CONTROL.md
├── thermal_guard.py # Protection contre la surchauffe du processeur pour le panneau
 # (fonctionne aussi seule)
├── thermal-guard.service # unité systemd d'exemple pour le garde, sans panneau
 # d'administration
├── tmpfiles
│ └── phantomsdr-logs.conf # garde admin.log + proxy.log à l'utilisateur du panneau (adapter
 # les chemins avant installation)

```

```

├─ update.sh
├─ waterfall.sh
├─ websdr_relay.json.example # modèle de configuration (port, limites, identité de la station)
├─ websdr_relay.py # le relais lui-même – voir docs/RECEIVE_DIVERSITY.md
├─ websocketpp_asio_connection.hpp # adaptateur Boost >= 1.87 : ws_post au lieu de
io_service::post
├─ websocketpp_asio_endpoint.hpp # adaptateur Boost >= 1.87 : ws_work / ws_restart,
max_listen_connections
├─ websocketpp_asio.hpp # adaptateur Boost >= 1.87 pour websocketpp (io_context,
executor_work_guard)
├─ xgo.sh # ancien : démarre spectrumserver, appelé par check-go.sh

```

## Répertoire racine

### Fichiers de configuration

Fichier	Rôle	Quand le modifier
config.toml	Configuration par défaut	Installation initiale, tests
config-rtl.toml	Configuration propre au RTL-SDR	Utilisation d'un RTL-SDR
config-rspla.toml	Configuration SDRplay RSP1A	Utilisation d'un RSP1A
config-airspyhf.toml	Configuration Airspy HF+	Utilisation d'un Airspy
config-fobos-hf.toml	RigExpert Fobos SDR, HF1/HF2 en échantillonnage direct (0-25 MHz)	Utilisation d'un Fobos en HF
config-fobos.toml	RigExpert Fobos SDR, voie RF (25-6000 MHz)	Utilisation d'un Fobos au-dessus de 25 MHz
config-hackrf.toml	Configuration HackRF One	Utilisation d'un HackRF
config-rx888mk2.toml	Configuration RX888 MK2	Utilisation d'un RX888
config.example.hackrf.toml	Exemple HackRF One	Utilisation d'un HackRF

### Scripts de démarrage/arrêt et de maintenance

Chaque `start-*.sh` ci-dessous est un **lanceur autonome + chien de garde + journaliseur** : il arrête toute instance en cours, lance le récepteur + `spectrumserver`, se détache en arrière-plan, relance automatiquement la chaîne si elle s'arrête, et journalise dans `logwebsdr.txt`. Ils partagent un même script d'arrêt et un unique verrou `lock` (un seul récepteur à la fois). Ne modifiez que le bloc **RECEIVER CONFIGURATION** en tête de chacun (arguments du récepteur / configuration / nom du processus). Les six scripts de lancement issus du même modèle (`start-airspyhf.sh`, `start-rtl.sh`, `start-rspla.sh`, `start-fobos.sh`, `start-fobos-hf.sh`, `start-hackrf.sh`) sont identiques sous ce bloc ; `start-rx888mk2.sh` est un script à part. Depuis la v5.0.0, plusieurs récepteurs peuvent tourner côte à côte : un script de démarrage lancé avec `INSTANCE=<nom>` donne à ce récepteur son propre dossier `instances/<nom>/`, son propre journal, son verrou et son FIFO — voir [Plusieurs récepteurs](#).

Script	Rôle
<code>install.sh</code>	Installation et compilation automatisées
<code>add-receiver.sh</code>	Ajoute un récepteur de plus à cet ordinateur : demande quel récepteur et ce qu'il couvre, installe son pilote, crée instances/<nom>/ et l'entrée de <code>receivers.toml</code> , et propose de le démarrer et de redémarrer le proxy. Les installeurs le proposent à la fin — voir <a href="#">Plusieurs récepteurs</a>
<code>start-rtl.sh</code>	Lancement + chien de garde du serveur avec RTL-SDR ( <code>rtl_sdr</code> )
<code>start-rspla.sh</code>	Lancement + chien de garde du serveur avec SDRplay RSP1A ( <code>rx_sdr</code> ) ; utilise <code>libmirisdr-5</code> , ou l'API SDRplay quand elle est installée ( <code>RX_DRIVER</code> en impose un)
<code>start-airspyhf.sh</code>	Lancement + chien de garde du serveur avec Airspy HF+ ( <code>rx_sdr</code> )
<code>start-fobos-hf.sh</code>	Lancement + chien de garde du serveur avec RigExpert Fobos SDR, HF1/HF2 en échantillonnage direct ( <code>rx_sdr \ cf32_to_real</code> )
<code>start-fobos.sh</code>	Lancement + chien de garde du serveur avec RigExpert Fobos SDR, voie RF ( <code>rx_sdr</code> )
<code>start-hackrf.sh</code>	Lancement + chien de garde du serveur avec HackRF One ( <code>hackrf_transfer</code> )
<code>start-rx888mk2.sh</code>	Lancement + chien de garde du serveur avec RX888 MK2 ( <code>rx888_stream</code> )
<code>start-all.sh</code>	Démontre chaque récepteur de <code>receivers.toml</code> par son propre script — à choisir comme script de démarrage du panneau d'administration sur une station à plusieurs récepteurs ; voir <a href="#">Plusieurs récepteurs</a>
<code>stop-websdr.sh</code>	Arrêt du serveur et de son chien de garde — partagé par tous les récepteurs
<code>stop-websdr.sh</code> <code>&lt;instance&gt;</code>	Arrête un seul récepteur ( <code>main</code> pour le principal) ; sans argument, tous s'arrêtent, comme avant
<code>setup-rx888-udev.sh</code>	Installe les règles udev pour que <code>rx888_stream</code> du RX-888 tourne sans sudo
<code>setup-rspla.sh</code> / <code>setup-fobos.sh</code> / <code>setup-airspyhf.sh</code>	Installent la chaîne de pilote SoapySDR du RSP1A, du Fobos ou de l'Airspy HF+ — SoapySDR depuis la distribution, pilote et <code>rx_tools</code> compilés dans <code>sdr_drivers/</code> , une règle udev — sur l'une des quatre distributions prises en charge. Les options 3, 5 et 6 de l'installateur les lancent ; ils fonctionnent aussi seuls. Code commun dans <code>setup-sdr-common.sh</code> — voir le <a href="#">Guide d'installation</a>
<code>setup-hackrf.sh</code>	Installe le paquet <code>hackrf</code> de la distribution et une règle udev pour le HackRF One, sur l'une des quatre distributions prises en charge ; l'option 7 de l'installateur le lance
<code>setup-rtlsdr.sh</code>	Installe le pilote RTL-SDR sur l'une des quatre distributions prises en charge — le paquet <code>rtl-sdr</code> de la distribution ou, avec <code>RTL_V4=y</code> , le pilote du RTL-SDR Blog V4 compilé depuis les sources —, bloque le pilote DVB-T et ajoute une règle udev ; <code>add-receiver.sh</code> le lance
<code>setup-firewall.sh</code>	Garde anti-afflux optionnelle au niveau du noyau : charge une table nftables avec un plafond de connexions simultanées et un débit par adresse source sur les ports du récepteur, un frein contre la force brute SSH et le partage de fichiers Windows fermé hors des plages privées. Nécessite root, ne peut pas vous enfermer dehors ( <code>policy accept</code> , les connexions établies acceptées en premier) et <code>--apply</code> se retire de lui-même s'il n'est pas confirmé en 60 s — voir <a href="#">Limites de connexion</a>
<code>setup-cpufreq-perms.sh</code>	Donne à un groupe <code>cpufreq</code> le droit d'écriture sur la limite de fréquence du processeur, afin que l'étage throttle du garde fonctionne sans root. Installe une règle <code>tmpfiles.d</code> pour survivre à un redémarrage ; <code>--revoke</code> annule tout
<code>update.sh</code>	Mettre à jour l'installation depuis l'arborescence publiée sans toucher à votre configuration, vos marqueurs, la liste des fréquences ni vos modifications — voir le <a href="#">Guide d'installation</a>

Script	Rôle
<code>recompile.sh</code>	Reconstruire le backend et/ou le frontend et choisir la variante servie à /
<code>smeter_theme.sh</code>	Définir le cadran par défaut du S-mètre analogique (dark / amber / vintage) pour tous les utilisateurs, et proposer la reconstruction du frontend — voir <a href="#">Modifier les variantes</a>
<code>waterfall.sh</code>	Modifie le niveau minimal de cascade par défaut (dB) dans <code>waterfall.js</code> + <code>App.svelte</code> — voir <a href="#">README</a>
<code>kiwi_install.sh</code>	Installe l'émulation des clients KiwiSDR sur une arborescence qui ne l'a pas : corrige les sources du backend, copie <code>src/kiwi_bridge.h</code> et ajoute un bloc <code>[kiwi_emulation]</code> documenté aux fichiers de configuration de la racine. Idempotent, et sauvegarde chaque fichier qu'il touche — voir <a href="#">Émulation des clients KiwiSDR</a>
<code>tci-bridge/tci-rigctld.mjs</code>	Non utilisé par le récepteur. Un petit programme Node.js que l'auditeur lance à côté de son propre transceiver : il lit la fréquence, le mode et l'état d'émission dans <code>rigctld</code> de Hamlib et les sert en TCI sur le port 50001, pour que le bouton <b>TCI-CAT</b> de la page pilote un poste sans TCI, comme l'IC-7300 — voir <a href="#">Pilotage du transceiver</a>

**Ancienne chaîne de lancement.** `go.sh`, `xgo.sh`, `check-go.sh`, `kill.sh` et `_relaunch.sh` sont la génération précédente de scripts de lancement, de surveillance et d'arrêt. Tout ce qu'ils faisaient se trouve désormais dans chaque `start-<radio>.sh`, qui est ce qu'il faut utiliser. Ils restent sur le disque parce que des installations existantes les référencent, et ne sont plus maintenus.

## Fichiers de données

Fichier	Rôle	Format
<code>markers.json</code>	Favoris et marqueurs de fréquence	JSON
<code>chat_history.txt</code>	Messages de chat des utilisateurs	Texte brut
<code>favicon.ico</code>	Icône du site	Image ICO
<code>fftw_wisdom</code>	Données d'optimisation FFT	Binaire FFTW
<code>phantom_fftw_wisdom</code>	Optimisation FFT supplémentaire	Binaire FFTW

## Panneau d'administration et report de spots

Fichier	Rôle
<code>admin_server.py</code>	Le panneau d'administration lui-même. Outre les pages de gestion, il fait tourner l'échantillonneur de la page <b>Graphiques</b> : un thread d'arrière-plan relève toutes les 2 secondes la fréquence du processeur, la charge, la température et les utilisateurs connectés, et les conserve uniquement en mémoire — 1 heure en pleine résolution plus 24 heures de moyennes sur 30 secondes. Rien n'est écrit sur le disque : l'historique est perdu au redémarrage.
<code>admin_config.json</code>	Réglages du panneau d'administration (empreinte du mot de passe, ports, options, seuils de la protection thermique)
<code>thermal_guard.py</code>	Protection contre la surchauffe du processeur. Arrête le serveur lorsque le processeur devient trop chaud et le relance une fois refroidi, en déduisant ses seuils du seuil critique que publie votre propre processeur plutôt que d'un nombre fixe. Elle ne cherche jamais à identifier ce qui supervise le serveur : tant qu'il fait trop chaud, elle réémet l'arrêt toutes les 2 s, si bien qu'un watchdog, une unité systemd ou une tâche cron qui le relance est défait jusqu'au refroidissement de la machine. Bibliothèque standard uniquement ; importée par <code>admin_server.py</code> (cadencée par l'échantillonneur des Graphiques) et exécutable seule pour les installations sans panneau. Par défaut en journalisation seule, elle n'agit donc sur rien tant qu'elle n'est pas activée — voir [Panneau d'administration]voir le <a href="#">manuel du Thermal Guard</a>
<code>autorun/</code>	Le démon de report de spots — voir <a href="#">Rapporteur de spots autorun</a>
<code>autorun.json</code>	Bandes et modes à décoder, identité et destinations
<code>autorun-status.json</code>	État courant du démon : alimente les compteurs <b>de l'exécution en cours</b> (les tuiles par décodeur), remis à zéro par Stop/Start
<code>autorun-totals.json</code>	Spots téléversés <b>depuis toujours</b> par bande et mode : le nombre à côté de chaque case ; écrit par le démon afin de survivre aux redémarrages

## Fichiers du système de compilation

Fichier	Rôle
<code>meson.build</code>	Configuration principale de compilation
<code>meson_options.txt</code>	Options de compilation configurables
<code>.gitattributes</code>	Attributs du dépôt Git

## Code source (`src/`)

Le répertoire `src/` contient l'implémentation du backend en C++.

### Composants clés

#### 1. Application principale (`main.cpp`)

- Analyse les arguments de la ligne de commande
- Charge le fichier de configuration
- Initialise les composants du serveur
- Démarre la boucle d'événements

## 2. Serveur de spectre ( `spectrumserver.cpp` )

- Coordonne tous les composants
- Gère les connexions des utilisateurs
- Distribue les données de spectre
- Traite les requêtes des utilisateurs

## 3. Pilotes SDR ( `drivers/` )

- Interface abstraite pour le matériel SDR
- Lecture et mise en forme des données d'échantillons
- Gestion des fonctions propres à chaque appareil

## 4. Moteur DSP ( `dsp/` )

- Calcul de FFT (accéléré par CPU/GPU)
- Démodulation (AM, FM, BLU, CW, etc.)
- Filtrage et rééchantillonnage audio
- AGC et réduction de bruit

## 5. Serveur web ( `server/` )

- Communication WebSocket
- Service de fichiers statiques HTTP
- Gestion des sessions utilisateur
- Diffusion de données en temps réel

## 6. Encodage audio ( `audio/` )

- Compression FLAC
- Compression Opus
- Optimisation de la diffusion

# Frontend ( `frontend/` )

L'interface utilisateur web construite avec Svelte et Vite.

## 1. Application principale ( `App.svelte` )

- Composant de plus haut niveau
- Structure de la mise en page
- Orchestration des composants
- **Rangée de boutons de décodeur** — un bouton par décodeur sur le panneau principal, juste sous le sélecteur de modes ; une pression démarre le décodeur et ouvre sa fenêtre, une seconde l'arrête. Elle a remplacé l'ancienne rangée de largeur de bande. RADEL/RADEU se trouvent dans `lib/ModesSelector.svelte`, à côté du sélecteur de modes et dans les fenêtres **Modes** et **Bands**.

## 2. Affichage en cascade ( `waterfall.js` + `lib/` )

- Rendu du spectre et de la cascade sur canevas, palettes de couleurs et ajustement automatique adaptatif, le tout dans `waterfall.js` (JS pur, pas un composant)
- Accord interactif via `lib/PassbandTuner.svelte`
- Superpositions du plan de bandes et des marqueurs via `lib/FrequencyMarkers.svelte`
- Le spectrogramme audio est un composant distinct : `lib/Spectrogram.svelte`

### 3. Commandes (App.svelte + lib/)

- Saisie et affichage de la fréquence — `lib/FrequencyInput.svelte`
- Choix du mode (AM/FM/BLU/CW) — `lib/ModesSelector.svelte`, changement de bande — `lib/BandSelector.svelte`
- L'AGC, le NR, le NB et les autres commandes se trouvent dans `App.svelte` lui-même ; il n'existe pas de `Controls.svelte` distinct

#### 3a. Scanner (`scanner.js`)

- Balayage de canaux : promène le récepteur sur une plage et s'arrête au premier canal portant un signal. Du JS pur, pas un composant — `App.svelte` fournit le VFO, le mode, le plan de bandes et l'appel d'accord, et reçoit en retour l'état de l'interface par un seul rappel
- Le seuil est exprimé en dB au-dessus du bruit de fond de la bande plutôt qu'en niveau absolu, en réutilisant le bruit que `waterfall.js` suit déjà (`snrNoiseDb`) : aucun étalonnage propre n'est nécessaire
- Deux façons de parcourir la bande : accorder et écouter chaque canal, ou examiner d'abord le spectre et n'accorder que ce qu'il ne peut pas écarter. Le spectre peut sauter un canal mais jamais s'arrêter sur un — tout arrêt provient d'une écoute réelle
- La plage est soit la bande du plan de bandes, soit exactement ce que montre la cascade ; la reprise automatique, la durée de séjour maximale et la liste d'exclusion sont ici également, conservées dans `localStorage`

### 4. Système audio (`audio.js`)

- Flux audio WebSocket
- Décodage FLAC/Opus
- Contrôle de la lecture audio
- Distribue le PCM brut (prélevé avant l'AGC, la réduction de bruit et la coupure du son) aux décodeurs de modes
- Réduction de bruit par IA : `lib/rnnoise.js` charge le module WebAssembly RNNoise (`@jitsi/rnnoise-wasm`) à la première utilisation ; `audio.js` ne l'applique qu'aux modes phonie, après la prise des décodeurs, qui ne l'entendent donc jamais
- Le FLAC passe par `flacLowLatency.js`, qui décode chaque paquet dès qu'il arrive et ramène la fréquence réelle du flux (par exemple 12016,3 Hz) à 12000 Hz avec un rééchantillonneur à 64 coefficients (2,7 ms de retard). Le décodeur WASM de `modules/` livrait le son par blocs fixes de 1024 échantillons et retenait environ 160 ms ; il sert encore pour Opus, pour les sources de diversité et pour le FLAC quand la page est ouverte avec `?flac=wasm`
- Contrôle du tampon de lecture (`playoutControl.js`) : `ArrivalMeter` mesure l'irrégularité d'arrivée des paquets par rapport à la chronologie propre du flux, ainsi que la dérive entre l'horloge d'échantillonnage du récepteur et la carte son de l'auditeur ; `CushionController` maintient le plus petit tampon de la dernière minute près de la marge du cran de tampon choisi, en coupant ou en répétant une période accordée à la hauteur du son (`spliceOut` / `spliceIn`) dans l'audio envoyé au haut-parleur. Les décodeurs et les enregistrements reçoivent le flux intact. S'applique au chemin à tampons planifiés qu'utilise tout auditeur d'une station en http simple ; `?drain=0` le désactive, `?diag=1` affiche ses valeurs

#### 4a. Décodeurs de modes et leurs workers

Chacun des décodeurs de modes lourds s'exécute dans son propre Web Worker, de sorte que le décodage ne bloque jamais la lecture audio ni la cascade. Ils suivent un schéma commun — trois fichiers par décodeur :

Décodeur	Moteur	Worker	Proxy sur le thread principal
SSTV	sstv.js	sstv.worker.js	sstvWorkerProxy.js
FAX 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

- Le **moteur** est du code DSP pur qui ignore tout des workers : il peut donc aussi être exécuté directement (tests unitaires, ou repli dans le thread principal).
- Le **worker** détient une instance du moteur et retransmet ses événements tels quels.
- Le **proxy** reproduit la surface de méthodes du moteur, de sorte qu'`audio.js` l'appelle exactement comme il appellerait le décodeur. Il crée le worker paresseusement à la première activation et bascule vers une exécution dans le thread principal si les workers ne sont pas disponibles.

Deux détails sont porteurs : le PCM est **copié** dans un nouveau tampon avant d'être transféré au worker (transférer une vue sur l'accumulateur audio le détacherait et tuerait la lecture), et le message `init` du worker réapplique la configuration à un moteur déjà en cours plutôt que d'en supposer un neuf.

`fsk.js` sert à la fois NAVTEX et FSK/RTTY à partir d'un seul moteur, sélectionné par instance via un champ `role` ; chaque instance possède son propre état, de sorte que les deux peuvent tourner indépendamment.

Le rôle `fsk` héberge en outre deux décodeurs qui ne sont pas du FSK du tout. Lorsque la variante `psk31` ou `olivia` est sélectionnée, `fsk.js` transmet l'audio à `psk31.js` ou `olivia.js` au lieu de sa propre chaîne à discriminateur, tout en empruntant sa configuration, son worker et sa gestion d'événements — ainsi `fsk.worker.js`, `fskWorkerProxy.js` et `audio.js` n'ont besoin de rien savoir de ces deux modes, et l'interface consomme partout les mêmes événements `char/status/metrics`.

- `psk31.js` — BPSK31 : bande de base complexe, filtre adapté, détection différentielle et varicode, avec une acquisition spectrale grossière puis un AFC fin couvrant environ  $\pm 25$  Hz.
- `olivia.js` — Olivia MFSK : un portage du récepteur MFSK de Pawel Jalocho issu de `fldigi` (`pj_mfsk.h`, GPL-3, comme ce projet), y compris la correction d'erreurs de Walsh/Hadamard et la recherche aveugle de synchronisation sur la phase de bloc et le décalage de fréquence.
- `broadcastSchedules.js` — les horaires UTC que les décodeurs FAX, NAVTEX et RTTY proposent en présélections, issus des programmes de fac-similé maritime de la NOAA/NWS et des listes publiées de stations NAVTEX

#### 4b. Diversité de réception (`diversity.js`)

- Associe le récepteur local à un second situé ailleurs et suit le site qui a, à cet instant, le meilleur signal. Du JS simple, pas un composant — il s'insère à un seul endroit de `audio.js`, qui lui passe le PCM local et restitue ce qu'il renvoie
- **Sélection, pas addition.** Deux sites entendent la même émission par des trajets ionosphériques différents, donc leurs formes d'onde ont des phases sans rapport ; les additionner donne un filtrage en peigne. Une combinaison cohérente exigerait une horloge commune, que deux récepteurs reliés par internet ne partagent pas. Le mélange n'a lieu que pendant un fondu de 30 ms
- L'alignement corrèle les deux **enveloppes audio** (puissance logarithmique à 100 Hz), jamais les formes d'onde — l'enveloppe survit au trajet comme à n'importe quel codec. Un verrouillage n'est retenu que si une seconde recherche indépendante concorde, ce qui écarte le décalage assuré mais faux que produiraient deux sites s'évanouissant en opposition

- Le flux distant est d'abord **verrouillé en cadence** sur le local. Deux récepteurs, ce sont deux horloges et deux chaînes de décimation : leur audio arrive jusqu'à 2% décalé même quand tous deux annoncent 12 kHz — 240 échantillons par seconde de dérive, qu'aucune corrélation ne retient. Le rapport est mesuré d'après le nombre d'échantillons réellement livré de chaque côté et appliqué par un rééchantillonneur qui conserve sa phase fractionnaire d'un bloc à l'autre, si bien qu'un rapport quelconque tient indéfiniment
- Le choix du site s'appuie sur un SNR par centiles mesuré sur des échantillons **alignés en contenu**, avec hystérésis, temporisation de maintien et une sortie rapide si le site actif s'effondre. Les niveaux sont appariés bruit à bruit, pour qu'un changement ne modifie pas le souffle de fond
- Les décodeurs conservent le flux **local** : FT8, JS8, WSPR et RADE intègrent de façon cohérente sur un intervalle, et un changement en cours d'intervalle est une discontinuité de phase qui peut coûter le décodage

#### 4c. Sources de diversité ( `remoteSource.js`, `kiwiSource.js`, `uberSource.js`, `webSdrSource.js` )

- Un seul contrat — `onPcm` / `onState` / `tune` / `canReceive` — si bien que `diversity.js` ignore toujours ce qui se trouve en face. Ajouter un type de récepteur revient à ajouter un fichier
- `remoteSource.js` — un autre PhantomSDR-Plus via `/audio` (cbor + FLAC), réutilisant `createDecoder()` de `lib/wrappers.js`
- `kiwiSource.js` — un KiwiSDR : trames SND, PCM gros-boutiste, avec l'horodatage GPS de 10 octets qu'un paquet stéréo insère avant l'audio
- `uberSource.js` — un UberSDR via son `/ws` natif : Opus derrière un en-tête de 21 octets, réaccord sur la connexion ouverte. L'identifiant de session doit d'abord être déclaré par `POST /connection` et doit être un UUID
- `webSdrSource.js` — un WebSDR, via `websdr_relay.py` sur ce serveur : le navigateur ne peut pas se connecter directement, car WebSDR vérifie l'en-tête `Origin` et aucun script n'a le droit de le modifier. L'accord passe en trame texte sur la même connexion ; la couverture des bandes vient du relais
- `webSdrCodec.js` — le format audio de WebSDR : un flux étiqueté octet par octet dont les blocs compressés alimentent un prédicteur leaky-LMS à 20 prises. Porté depuis le client WebSDR et vérifié échantillon par échantillon
- `diversityList.js` — la liste des récepteurs enregistrés et les règles d'adresse des quatre types, partagées par le panneau de bureau et la page mobile pour qu'un seul format serve aux deux. Ainsi que le transfert — un blob compact, son code QR et un analyseur qui accepte toutes les formes que les deux pages ont jamais écrites — et `browseUrl()`, qui retransforme une adresse appelée en une adresse qu'un navigateur peut ouvrir
- `lib/DiversityPanel.svelte` — l'interface : adresse, type de source, correction de SNR et état en direct, ainsi que les récepteurs enregistrés — nommés, réordonnables, exportés et importés sous forme de fichier JSON, par type de source dans `localStorage`. Tout s'édite à l'intérieur du panneau : `prompt()` et `confirm()` bloquent le thread principal, celui-là même par lequel l'audio est poussé vers le worklet de lecture. Un bouton  **QR** dessine la liste en code scannable, car `localStorage` appartient à un navigateur et un téléphone démarre vide. `mobile/Mobile.svelte` reprend la même fonction dans un onglet **Div**, dans le CSS simple propre à cette page. Voir [Diversité de réception](#)

#### 4d. Identificateur de mode ( `modeId.js`, `modePriors.js` )

- Répond à « qu'est-ce que j'écoute ? ». Il lit la même prise PCM brute que les décodeurs et classe les modes probables, pour que l'opérateur choisisse le bon décodeur au lieu de les essayer tous les dix. Il ne décode jamais : il mesure des propriétés physiques du signal et les note face à une table de modes connus
- La largeur occupée est la largeur contiguë à -15 dB autour du pic, délibérément pas une valeur à 99 % de puissance : les clics de manipulation laissent de longues traînes sur l'intégrale de puissance et faisaient paraître tout mode étroit plusieurs fois trop large. La rapidité de modulation vient de la

**fréquence instantanée** et non des énergies des tonalités, car aucune durée d'intégration ne résout à la fois un décalage de 170 Hz et un symbole de 100 Bd. La cadence de manipulation vient de l'enveloppe on/off et sert aussi d'indicateur de vitesse en CW

- `modePriors.js` apporte le seul indice que l'audio ne peut pas porter : l'endroit où vous êtes accordé. Un signal 100 Bd / 170 Hz sur 518 kHz est du NAVTEX ; le même signal sur 14,070 MHz ne l'est pas. Il ne fait que répondre ce que le signal soutenait déjà, et n'invente jamais un candidat
- FT8, JS8 et FT2 sont annoncés comme un seul groupe à dessein : ils ne se séparent pas sur la seule largeur et le seul espacement des tonalités, et prétendre le contraire serait une réponse fausse énoncée avec assurance
- En dessous d'environ 10 dB de SNR, il se tait plutôt que de deviner
- Il tourne dans son propre Web Worker (`modeId.worker.js` + `modeIdWorkerProxy.js`), selon le même schéma moteur/worker/proxy que les décodeurs ci-dessus ; le résultat est la pastille de `lib/ModeIdChip.svelte`

#### 4e. Connexions refusées (`refused.js`, `clientVersion.js`)

- `refused.js` est le vocabulaire commun d'une connexion que le serveur éconduit : code de fermeture **4003** (au-delà d'une limite par adresse, ou page plus ancienne que `[server] min_client_version`) et **4001** (une exclusion par le sysop). Les deux sont définitifs. `audio.js`, `waterfall.js` et `events.js` importent tous `isRefusal()` de là, car tous trois ouvrent une connexion qui peut être refusée et tous trois doivent alors régler leur promesse d'initialisation — sinon la page attend indéfiniment une connexion qui ne viendra jamais
- Rien ne réessaie. Une connexion `/audio` tombée met fin à la session, délibérément : `/waterfall` et `/events` ne revenaient jamais avec une reconnexion, une session réessayée était donc de l'audio vivant sur une cascade figée, et face à une limite de débit chaque tentative prolongerait justement le refus qu'elle cherchait à contourner. La page de bureau explique un refus survenu au chargement et s'arrête simplement quand il survient en cours de session ; `/mobile` affiche une ligne invitant l'auditeur à recharger
- `clientVersion.js` contient un seul entier, `CLIENT_VERSION`, que la page ajoute à sa connexion audio sous la forme `/audio?v=N`. Le serveur éconduit tout ce qui est inférieur à `[server] min_client_version` : c'est ainsi qu'une station force les onglets exécutant encore une version antérieure à recharger — le seul levier existant, puisque le serveur n'atteint pas le JavaScript déjà en cours dans un navigateur. Incrémentez-le quand une modification du frontend ne doit plus être ignorée par les onglets ouverts
- Référence complète : [Limites de connexion](#)

#### 4f. Réponses dans le chat (`lib/chatReply.js`)

- Une réponse est un message de chat ordinaire qui commence par un marqueur nommant le message auquel il répond, `[RE:<horodatage>|<nom>]` ; le serveur le stocke et le relaie comme toute autre ligne et n'a besoin d'aucune modification
- `chatReply.js` crée et lit ce marqueur et regroupe les réponses sous leur original, sur un niveau. `App.svelte` et `mobile/Mobile.svelte` l'utilisent tous deux, si bien que les deux pages organisent le chat de la même façon
- Quand l'original a déjà quitté l'historique de 20 messages du serveur, la réponse reste là où elle est arrivée et n'affiche que le nom et l'heure auxquels elle répond

### 5. Gestion d'état (`stores/`)

- Magasins de données réactifs
- État applicatif partagé
- Gestion des événements

## Fonctions clés

- Traitement audio en temps réel
- Décodage de modes numériques (FT8, RTTY, etc.)
- Filtrage audio
- Analyse spectrale

## Listes de fréquences ( `frequencylist/` )

Les repères de fréquence sur la cascade. `mymarkers.json` est la liste que le récepteur affiche réellement ; le reste est la matière première que `update-markers.sh` transforme en elle, actualisée depuis les grilles en ligne. Le `README.md` de ce dossier explique la mise à jour dans les sept langues.

```
frequencylist/
├─ mymarkers.json # les repères affichés par le récepteur – édités à la main, survivent
à une mise à jour
├─ shortwavestations.json # stations en ondes courtes, généré par generate-current-shortwave.py
├─ 0.TXT # la grille HF mondiale A26, décompressée et analysée par update-
markers.sh
├─ a26allx2.zip # archive source téléchargée par update-markers.sh
├─ admin.txt · antenna.txt · broadcas.txt · fmorg.txt · language.txt · site.txt
| # listes annexes de la même source (sites, langues, administrateurs,
antennes)
├─ generate-current-shortwave.py
├─ update-markers.sh # actualise tout ce qui précède depuis les grilles en ligne
└─ README.md
```

## Format ( `mymarkers.json` , `shortwavestations.json` )

```
[
 { "frequency": 77500, "name": "DCF77", "mode": "CW" },
 { "frequency": 2485000, "name": "Vanuatu Broadcasting", "mode": "AM" }
]
```

## Fichiers de configuration

### Configuration du serveur (fichiers `.toml` )

Structure des fichiers de configuration :

```

[server]
Web server settings
port = 9002
html_root = "frontend/dist/"
threads = 2
otherusers = 1

[limits]
Limites par adresse – toutes désactivées ou à une valeur raisonnable par défaut,
une configuration sans elles se comporte comme avant. Voir CONNECTION_LIMITS.md.
per_ip = 3 # auditeurs simultanés depuis une adresse
per_ip_rate = 40 # nouvelles connexions par minute depuis une adresse

[websdr]
Online registration
register_online = true
name = "WebSDR Name"
antenna = "Antenna Type"
grid_locator = "AB12cd"
hostname = "domain.com"

[input]
SDR input settings
sps = 2048000 # Sample rate
fft_size = 131072 # FFT size
frequency = 145000000 # Base frequency
signal = "iq" # Signal type: "iq" or "real"
audio_sps = 12000 # Audio sample rate
audio_compression = "opus" # "flac" or "opus"
accelerator = "opencl" # "none", "cuda", "opencl"

[input.driver]
Driver settings
name = "stdin"
format = "u8" # Sample format

[input.defaults]
User interface defaults
frequency = 145000000
modulation = "FM"

```

## Informations du site (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
}

```

# Système de compilation

## Configuration de compilation Meson

### meson.build (racine)

Définit :

- Les métadonnées du projet
- Les dépendances
- Les options du compilateur
- Les listes de fichiers sources
- Les cibles de compilation

### meson\_options.txt

Options 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')
```

## Dépendances entre fichiers

### Dépendances de compilation du 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
```

### Dépendances de compilation du frontend

```
frontend/dist/ depends on:
├─ Source files (frontend/src/**/*.*)
├─ npm packages (node_modules/):
│ ├── Svelte
│ ├── Vite
│ ├── @wasm-audio-decoders/opus-ml
│ └─ ...
└─ Static assets (frontend/public/)
```

## Flux de données

### Flux de fonctionnement du serveur

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

### Flux d'interaction utilisateur

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

## Guide de modification des fichiers

### Lorsque vous modifiez le code du backend ( `src/**` ) :

```
cd PhantomSDR-Plus
meson compile -C build
Server restart required
```

### Lorsque vous modifiez le code du frontend ( `frontend/src/**` ) :

```
cd PhantomSDR-Plus/frontend
npm run build
cd ..
Server restart required (for static files)
```

### Lorsque vous modifiez la configuration ( `.toml`, `.json` ) :

```
Restart server
./stop-websdr.sh
./start-rtl.sh # (or appropriate start script)
```

## Lorsque vous modifiez les marqueurs (markers.json) :

```
Reload page in browser
No server restart needed
```

## Chemins importants

### Chemins d'exécution

- **Configuration** : `./config-*.toml`
- **Racine HTML** : `./frontend/dist/`
- **Marqueurs** : `./markers.json`
- **Historique du chat** : `./chat_history.txt`
- **FFTW wisdom** : `./fftw_wisdom`, `./phantom_fftw_wisdom`

### Chemins de compilation

- **Binaire produit** : `./build/spectrumserver`
- **Sortie du frontend** : `./frontend/dist/`
- **Modules Node** : `./frontend/node_modules/`

### Chemins des sources

- **Sources du backend** : `./src/`
- **Sources du frontend** : `./frontend/src/`
- **Bibliothèques DSP** : `./jsdsp/`

## Opérations courantes sur les fichiers

### Ajouter une nouvelle configuration SDR

1. Copiez une configuration existante : `cp config-rtl.toml config-mydevice.toml`
2. Modifiez les paramètres : `nano config-mydevice.toml`
3. Créez un script de démarrage : `cp start-rtl.sh start-mydevice.sh`
4. Modifiez le script de démarrage : `nano start-mydevice.sh` — ne changez que le bloc **RECEIVER CONFIGURATION** en tête (`RX_LABEL`, `RX_COMM` = le nom du processus du récepteur, `RX_ARGS`, `CONFIG`, `FIFO`, ainsi que le hook `prestart` si l'appareil en a besoin). La logique de lancement/chien de garde/journalisation en dessous est générique et ne nécessite aucune modification.
5. Rendez-le exécutable : `chmod +x start-mydevice.sh`

`stop-websdr.sh` arrête déjà n'importe quel `start-*.sh --watchdog` ; si votre récepteur utilise un nom de processus autre que `rx888_stream/rx_sdr/rtl_sdr/hackrf_transfer/cf32_to_real`, ajoutez-y aussi une ligne `killall -9 <nom>`.

### Personnaliser le frontend

1. Modifiez la source : `nano frontend/src/App.svelte`
2. Recompilez : `cd frontend && npm run build && cd ..`
3. Redémarrez le serveur : `./stop-websdr.sh && ./start-rtl.sh`

## Ajouter des marqueurs personnalisés

1. Modifiez le fichier de marqueurs : `nano markers.json`
  2. Format : `json { "markers": [ { "frequency": 145500000, "label": "2m Calling", "mode": "FM" } ] }`
  3. Rechargez le navigateur (aucun redémarrage du serveur nécessaire)
- 

## Gestion de versions

---

### Fichiers à suivre dans Git

- Code source (`src/`, `frontend/src/`, `jsdsp/`)
- Exemples de configuration (`config.example*.toml`)
- Système de compilation (`meson.build`, `meson_options.txt`)
- Documentation (`*.md`, `docs/`)
- Scripts (`*.sh`)

### Fichiers à ignorer (`.gitignore`)

- Résultats de compilation (`build/`, `frontend/dist/`)
  - Dépendances (`frontend/node_modules/`)
  - Données utilisateur (`chat_history.txt`)
  - Configurations personnelles (`config-rtl.toml` s'il est personnalisé)
  - Données binaires (`*.o`, `*.so`)
- 

**Cette documentation de la structure devrait vous aider à naviguer dans le code de PhantomSDR-Plus et à le comprendre.**

Pour les instructions d'installation, voir [INSTALLATION.md](#). Pour les informations d'utilisation, voir [USER\\_GUIDE.md](#).

### 73 de SV1BTL & SV2AMK