AMATEUR RADIO · NOVEMBER 20, 2026

How uSDR's Automatic Voice-Scan Loop Does Wideband Decoding: Carrier Search, Parallel Test Decodes and Window Stepping

uSDR's SCAN loop finds carriers in one wideband FFT, re-mixes up to four channels inside the span without retuning, test-decodes each for about 1.2 s, and retunes the radio only when a window is exhausted. Based on the project README as of 20 Nov 2026.

13 min read · 10 sources · Updated Nov 20, 2026

P25 SDR configuration open source trunking

CONTENTS

Where this lives, and what "the loop" is 2
The loop at a glance 2
Stage 1: one wideband capture, cleaned and searched 3
- Capture and front-end correction 3
- The display FFT doubles as the scanner's sensor 3
- Peak detection with spur masking 3

Stage 2: narrowband chains inside the span 4
- Re-mixing without retuning 4
- What runs on a candidate 4
- Parallel testing 5

Stage 3: what counts as a lock 5
- Analog cross-check 5
- Skips and strikes 6
- Call endings and records 6
- Labels are not proof 6

Stage 4: moving the window and living with the hardware 6
- Retries and stream health 6
- SDRplay differences 7

What it costs: AUTO and classifier load 7
Configuring and controlling it 8
- Not in this first version 8
- Related tools 9

Bottom line 9
Sources 9

uSDR's automatic loop does not decode a wide span in one pass. It uses one wideband FFT to find where the carriers are, then re-mixes a few narrow channels out of the same sample stream and test-

decodes each one. The hardware is retuned only when a whole window has been worked through. This note describes that design as the project README documents it[1].

Where this lives, and what "the loop" is

The repository is at `github.com/innerpulsenet/uSDR`. The path in the request, `github.com/innerpulse.net/uSDR`, returns 404, because the GitHub account is `innerpulsenet`[3]. The README describes uSDR as "a single-radio spectrum inspector for RTL-SDR and SDRplay hardware". It is written in Rust and served to a web page and a native desktop console[1][2]. The studio's portfolio page calls it open source and in daily use as a lab receiver[2]. At the time of reading the repository had 269 commits and no stars[1]. The workspace has six crates: `desktop`, `dsp`, `engine`, `proto`, `radio` and `server`[9]. The studio lists SDR receivers and decoders among its services[10].

The README does not use the phrase "monitoring loop". Two features fit it:

- `SCAN` is the unattended loop. It sweeps a configured range, holds whatever it can decode, plays it when `MONITOR` is on, and files each call into Last Heard[1].
- `AUTO` is the decode arbiter the loop reuses. It runs the classifier, both digital voice receivers and every packet and paging decoder on the same channel at once, so a signal is identified without being told what it is[1].

The code in `crates/dsp`, `crates/engine` and `crates/radio` started as copies from the author's `scannerd` project (commit `5dda1c4`). The README says changes made in uSDR have not been sent back upstream[1].

> **CAUTION**
>
> This report rests on the README and portfolio text. I did not read the Rust source, so function names, thread layout and exact thresholds beyond those the README states are unverified. The README is also the project's own account, with measurements taken on the author's hardware.

The loop at a glance

3.2 MHz	10 MHz	~1.2 s	4	30 min
Max span, RTL-SDR	Max span, SDRplay	Test decode per candidate	Parallel channels (1–8)	Park time, control channels

Step 1 — **Capture the span**
Front end removes DC, LO leakage and image; Welch-averaged FFT every frame

Step 2 — **Find candidates**
Peak search inside the configured range, tuner spurs masked

Step 3 **Centre and test**
Inspect chains re-mix each candidate; centroid centring; ~1.2 s test decode

Step 4 **Lock or reject**
Evidence-based lock; failures earn strikes and skips

Step 5 **Hold and record**
Each held call records itself while the pool keeps sweeping

Step 6 **Move the window**
Only when candidates are exhausted, retune to the next overlapping window; wrap at the top

The steps above are stage order, not calendar dates. The central design choice is stated directly in the README: the scanner "spends its slow hardware retunes on span moves instead of channel steps, and finds carriers the cheap way"[1].

Stage 1: one wideband capture, cleaned and searched

Capture and front-end correction

The radio layer fans I/Q out to the DSP and engine crates through a bounded queue[1]. Every block first passes through `dsp::FrontEnd`, which removes DC offset and LO leakage and corrects the I/Q image before the spectrum measures it or any channel is mixed out[1]. Measured on a quiet band at 2.048 MS/s, this flattens the centre of the span to **0.04–1.7 dB** above the noise floor[1].

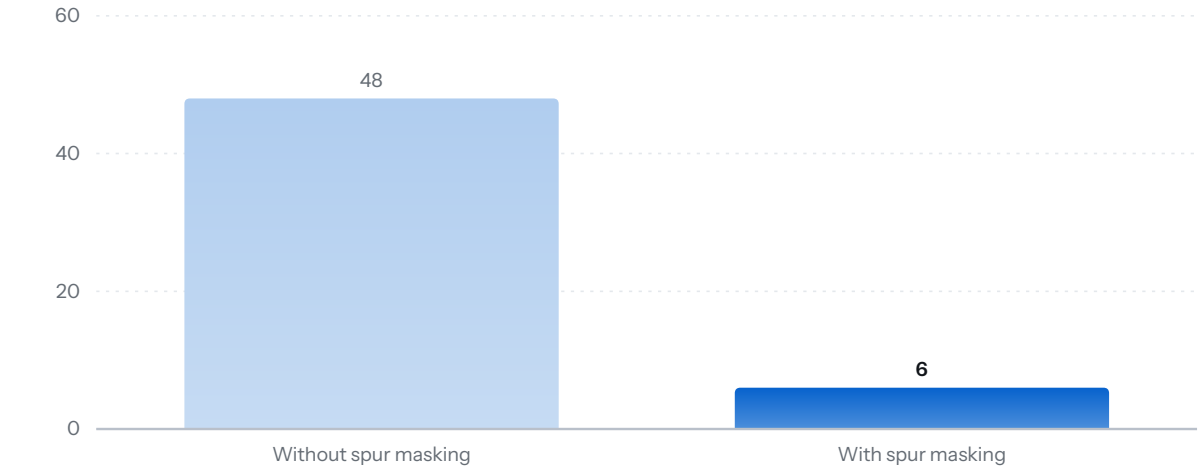
The display FFT doubles as the scanner's sensor

The FFT computed for the display every frame covers the whole span. Each frame is a Welch average of every half-overlapped segment since the last one, so a 2 ms burst between frames lands in the next frame instead of being dropped. Full duty at 10 MS/s costs under 5% of a core[1]. The scanner reuses this spectrum rather than running a separate sweep[1].

Peak detection with spur masking

Peak detection is the same local-maximum search the display uses. It runs against its own noise floor and produces the candidate carriers inside the configured range[1]. The peak list ignores the tuner's own artifacts: DC and LO leakage, plus the RTL2832's ±fs/4 images, within half a channel either side[1].

This masking matters in practice. A VHF/UHF/700/800 MHz survey without it returned **48** strong "signals" that all sat at the same baseband offset and moved with the tuner. With masking the same survey returned **6** candidates[1]. A manual `SPUR CHK` also exists. It nudges the LO by an eighth of a span and keeps every bin that did not move. Its threshold is 4 dB[1].



Source: uSDR README, Front end section; author's own hardware.

Gain affects this stage. The README says a wide-open front end on an RTL dongle makes its own images from nearby transmitters, and its advice is to turn gain down first when the screen fills with stationary "signals"[1]. Automatic gain is not the tuner's own AGC but a clip-guarded loop. It starts at 40% of range, steps down 2 dB within two seconds of ADC clipping, and creeps up only while the peak sample stays under a quarter of full scale[1].

Stage 2: narrowband chains inside the span

Re-mixing without retuning

This is the step that makes the decoding "wideband". The narrowband inspect chain "can re-mix onto any channel inside the current span without touching the tuner"[1]. Every candidate from Stage 1 is therefore reachable at no hardware cost. The README also says P25, DMR and NXDN each "extract and equalize their own channel out of the span and decode it to audio in real time"[1].

For each candidate the loop:

1. Centres it by spectral centroid.
2. Test-decodes it for about **1.2 s** with the receivers for the selected modes, all at once, using the same arbitration as `AUTO`[1].

What runs on a candidate

`AUTO` runs the classifier, both digital voice receivers and every packet and paging decoder on the same channel. A receiver that has locked outranks the classifier's heuristic for the label[1]. The engine recognises the following[1]:

- P25 Phase 1 and 2
- DMR Tier II/III, including Connect Plus, Capacity Plus and Hytera XPT control telemetry
- NXDN
- Motorola SmartNet/SmartZone, LTR and Passport
- POCSAG, FLEX, APRS/AX.25, NOAA SAME and MDC-1200
- CTCSS/DCS
- Frame-sync-qualified identification for ProVoice, EDACS, X2-TDMA, dPMR, M17, YSF and D-STAR

The scan itself only takes **AM, NFM, P25 and DMR** as modes. NXDN is not yet a scan mode[1].

Parallel testing

Candidates are not tested one at a time. Up to **four** parallel channels (configurable, 1–8) each carry their own receivers. On every sweep frame the idle ones take the strongest carriers in the window[1]. The stated reason is that a birdie's test dwell no longer holds up a short call two channels away. That matters on airband, where a transmission can be over in a few seconds[1].

When channels prove calls, each holds its own. Every held call records its own audio while the pool keeps sweeping, the newest call takes the speaker, and the window only moves once the last held call has dropped[1]. On screen, dashed amber brackets mark candidates under test, solid green brackets mark held calls recording in the background, and the filled passband is the call on the speaker[1].

Stage 3: what counts as a lock

A candidate "earns a lock on evidence, not on power alone"[1]. The criteria differ by mode:

MODE	LOCK EVIDENCE	IF IT FAILS OR IS THE WRONG KIND
P25 / DMR	Channel receiver found frame sync **and** a call started	Locks but only grants/control traffic: treated as a trunked control channel and parked for **30 minutes**, with the skip explained in the decode log. Locks but never talks: retested after **a minute**
NFM / AM	Hysteresis squelch on channel SNR (default **8 dB**, configurable) says a carrier is present; energy through the gated voice audio says someone is talking	Squelch blip with no voice behind it is filed nowhere

Analog cross-check

The two analog paths check each other. FM has a near-constant envelope, so the AM detector hears almost nothing from it. The FM discriminator hears almost nothing from AM. "Whichever path carries voice is what the channel was"[1].

Skips and strikes

Carriers that keep failing the test are treated as birdies or persistent hash. After a few strikes (configurable) they park for a long stretch and stop costing the sweep dwell time. They stay in the skip list, labelled with frequency, reason and time left, until re-tested. The panel has a per-entry re-test button, because "a memory that cannot be inspected cannot be trusted"[1]. Learned skips survive reconfiguration and the sweep wrapping around. Only a server restart forgets them[1].

Call endings and records

A call drops a couple of seconds after it goes quiet (configurable), which releases its slot back to the sweep. Analog calls are bracketed by the slot's squelch with the hang time trimmed off the tail. Each finished call lands in Last Heard tagged `P25`, `DMR`, `NFM` or `AM` with its frequency[1]. Last Heard keeps the most recent sixteen calls with their audio, and retains each call's decoded WAV and channelized raw CS16LE I/Q[1].

A decoded call is not always an audible one. Encrypted voice is vocoded to silence, so the list says `encrypted · ALG 0x84 — muted` or `no audio decoded` rather than leaving silence looking like a fault[1].

Labels are not proof

The README warns that "probable" labels are modulation and clock candidates, not validated frames. With no antenna the classifier fitted `M17 · 4-FSK @ 4800 Bd` to noise at 0.015 dB SNR[1]. The Protocol Decode/FEC panel shows rejected frame attempts alongside accepted ones for this reason. The legacy-family sweep (three baud rates across eight phases of a 0.65 s history) now runs only when something is above the noise floor. The README calls that a correctness fix as well as a saving, because sweeps over noise produced confident labels on empty channels[1].

Stage 4: moving the window and living with the hardware

The reason for this design is the tuner. An RTL-SDR's tuner faults above about one control write per ~250 ms. A 40 MHz range at 6.25 kHz steps would take "half an hour per sweep, nearly all of it spent listening to empty channels"[1]. That is roughly 6,400 steps at 0.25 s each, about 27 minutes (my arithmetic, consistent with the README's figure). With a span of up to 3.2 MHz, the same range needs a few window moves instead.

Only when the current window's candidates are exhausted does the tuner move, to the next overlapping window up the range. At the top it wraps[1].

Retries and stream health

The radio layer is built to ride out tuner faults[1]:

- Every tune is retried up to three times a few milliseconds apart.

- A failed tune is not recorded as done.
- A failed tune never tears the stream down.
- Reopening the device is a last resort. It needs six failures across thirty seconds with no successful tune, then backs off from 2 s to 30 s.
- The display never claims a frequency the tuner did not confirm.

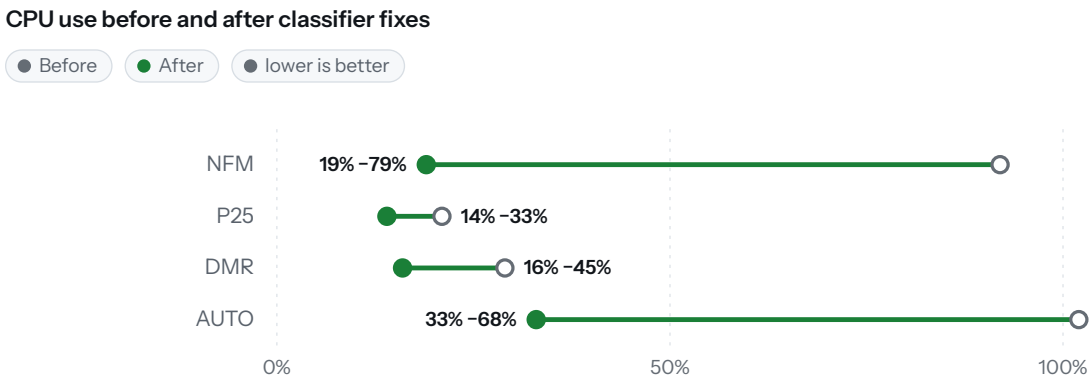
The README says plainly that a persistently failing dongle is a physical problem. Its advice is to unplug it and let it cool[1].

SDRplay differences

On an SDRplay RSP1A the span reaches 10 MHz. The tool runs LO offset by default, parking the LO just past the channel so the DC hump is not under the tuned signal[1]. Before a tune the LNA state is lowered to the deepest the new band has, because the API refuses a tune into a band that lacks the current LNA state[1].

What it costs: AUTO and classifier load

`AUTO` is the most expensive thing the loop leans on. The README reports that profiling found three scans re-examining work already done, and fixing them cut every mode by two to five times[1].



Source: uSDR README, Classifier cost; percent of one core, 2.4 MS/s per the AUTO text.

The three fixes were:

- **P25 frame detection** was 45% of all CPU on its own. It re-correlated about 7,400 positions per push against a sync it had already rejected, and now resumes where the previous scan stopped.
- **DMR acquisition** swept the whole retained buffer while unlocked, classifying each position twice (normal and inverted). It now keeps a marker that resets when the fit changes.
- **Legacy family detection** now runs only when something is above the noise floor.

Before the work, `AUTO` ran at 102% and dropped blocks at 2.4 MS/s. After it, 33% with none dropped[1]. Back-pressure is handled by design: DSP runs on a standard thread, and the I/Q

subscriber has a bounded queue. A slow classifier drops its own blocks instead of stalling the radio thread, and `dropped_blocks` and `lagged_blocks` report it[1]. If `AUTO` falls behind it says so instead of decoding patchily: "AUTO cannot keep up at 2.400 MS/s — dropping blocks; try a narrower span"[1].

Configuring and controlling it

Scan settings go through the HTTP API[1]:

```JSON
POST /api/sdr/scan/config
{"start_hz": 118000000, "stop_hz": 137000000,
 "modes": ["am","nfm","p25","dmr"],
 "threshold_db": 8, "resume_delay_s": 2}

POST /api/sdr/scan/control
{"action": "pause" | "resume" | "skip" | "forget"}
{"action": "unskip", "freq_hz": 123450000}
```

The field names come from the README. The values in this example are illustrative, not defaults, apart from the 8 dB threshold.

Behaviour to know:

- **Persistence.** Range, modes, threshold and resume delay persist in `usdr.toml`. A restart with `scan` as the saved mode comes straight back up scanning[1].
- **Operator priority.** Tuning by hand pauses the sweep, and the panel says so[1].
- **Voice keys.** P25 keys can be supplied with `--voice-keys`, but the README says a configured key cannot be authenticated. The `decrypted` flag means a key was applied, not that it was correct[1].
- **Security.** The server has no authentication. It binds to localhost by default and should not be exposed to a network you do not control[1].

Not in this first version

The README lists three gaps[1]:

1. Following a trunked control channel's voice grants. The engine already surfaces grants, and this is "the natural next step".
2. NXDN as a scan mode.
3. A manual per-channel lockout list.

The README says explicitly that explicitly configured P25 Phase 2 reception is not "automatic Phase 2 trunk-following parity" with OP25[1].

Related tools

Other open-source projects take a similar approach to wideband channelising. WaveShark describes itself as channelizing "a wide span once" and decoding every signal in it at once [6]. SDRTrunk is the established Java application for decoding, monitoring and recording trunked protocols [5]. uSDR's scan is narrower than either, being single-radio with no call database and no trunking follow [1].

Bottom line

uSDR's automatic loop is a search-then-verify design. One wideband FFT per frame finds candidates, and the inspect chains re-mix up to four of them out of the same samples at no tuner cost. Each is test-decoded for about 1.2 s against evidence-based lock rules. The slow, fault-prone hardware retune happens only per window. Held calls record independently, and control channels and birdies are parked by timer.

For practical use: keep the gain low enough that spur masking has little to hide, check SNR and the FEC panel before trusting a label, and do not expect it to follow trunked voice grants yet. If you need to confirm thresholds or the exact scheduling, read the Rust source, which this report did not.

Terms used in this document (4)

- **Birdie** — A steady spurious carrier, generated by the receiver or its surroundings, that looks like a signal.
- **Control channel** — In a trunked system, the channel carrying grants and signalling rather than voice.
- **Inspect chain** — uSDR's narrowband path that mixes one channel out of the already-captured span and decodes it, with no hardware retune.
- **Welch average** — A spectrum estimate made by averaging FFTs of overlapping sample segments, which smooths the noise floor.

SOURCES

1 innerpulsenet/uSDR README (GitHub)
https://github.com/innerpulsenet/uSDR

2 uSDR: Rust software-defined radio console (innerpulse portfolio)
https://www.innerpulse.net/portfolio/usdr

3 innerpulsenet GitHub profile
https://github.com/innerpulsenet

4 wavelet-lab/usdr-lib (unrelated uSDR hardware library)
https://github.com/wavelet-lab/usdr-lib

5 DSheirer/sdrtrunk
https://github.com/DSheirer/sdrtrunk

6 v0l/waveshark (super-radio)
https://github.com/v0l/super-radio

7 The Hidden Life of Public Safety Communications Signals (arXiv 2608.05247)
https://arxiv.org/pdf/2608.05247

8 innerpulse: custom software development and design
https://www.innerpulse.net/

9 uSDR crates directory (GitHub)
https://github.com/innerpulsenet/uSDR/tree/main/crates

10 innerpulse Firmware & radio services
https://www.innerpulse.net/services/firmware-radio

Researched and written with claude-sonnet-5-5, from the sources listed above. Figures are as reported by those sources. Exported November 29, 2026 from https://intel.innerpulse.net/doc/how-usdr-s-automatic-voice-scan-loop-does-wideband-decoding-carrier-search, where the current version lives.