Build a Pocket P25 Scanner with a Raspberry Pi, RTL-SDR and OP25

A plug-and-play, battery-powered scanner that follows a P25 Phase 1/Phase 2 trunked radio system, plays clear (unencrypted) calls through earbuds or a speaker, and starts on its own when you power it on.

Written from a real build that monitors the Regional Wireless Cooperative (RWC) in Phoenix, AZ. Replace the example frequencies and NAC with your own system's.

1. Parts list

Part	Notes
Raspberry Pi 3 or newer	Pi 3B+ / 4 work well. Pi OS with desktop (needed for the autostart method below).
microSD card, 16 GB+	Raspberry Pi OS (Desktop).
RTL-SDR dongle	A TCXO model (RTL-SDR Blog V3/V4, Nooelec NESDR SMArt) holds simulcast signals far better than a no-name dongle.
Antenna	Must cover your system's band (700/800 MHz for most). A VHF/UHF whip can work if the signal is strong, but an antenna cut for 700/800 MHz is better.
Power	USB battery bank rated 5V / 2.5A or better, plus a short, thick cable.
Audio out	Bluetooth earbuds/speaker (free), a USB sound dongle (about $5), or the 3.5 mm jack (noisier).

Optional: short USB extension cable, small heatsink for the dongle, cable ties.

2. Install the software

Install Raspberry Pi OS with desktop, connect to Wi-Fi, then open a terminal:

```
sudo apt update && sudo apt full-upgrade -y
sudo apt install -y git rtl-sdr
git clone https://github.com/boatbod/op25
cd op25
./install.sh
```

The OP25 installer takes a while. Check the repository's README if the steps have changed.

Programs live in:

```
~/op25/op25/gr-op25_repeater/apps
```

Plug in the dongle and check it works:

```
rtl_test -t
```

(Press Ctrl+C to stop it. Only one program can use the dongle at a time, so always close `rtl_test` before launching OP25.)

3. Gather your system's info from RadioReference

Go to radioreference.com and find your trunked system and the site that covers your area. For each site you need:

- **Control channel frequencies** (the red "CTRL" ones)
- **NAC**
- Whether the site is **simulcast** (modulation `cqpsk`)

IMPORTANT: NAC is hex

RadioReference lists NACs in **hexadecimal**. If it shows `531`, that is `0x531`, **not** decimal 531. Using the wrong NAC is the most common reason OP25 never decodes anything. (The OP25 status line shows the NAC it decodes, so you can compare.)

Example (RWC Simulcast A, Phoenix PD and City Services):

- NAC: `0x531`
- Control channels: 771.68125, 771.79375, 772.04375, 772.18125 MHz

Choose the site closest to you. Distant sites (for example Simulcast D, far north of central Phoenix) may be too weak to decode with a small antenna.

---

4. Check you can actually hear the control channel

Before blaming OP25, confirm there is a signal:

```
rtl_power -f 771.5M:772.5M:5k -i 5 -e 20s test.csv
```

(Swap in your own frequencies.) You want a clear narrow peak at least 10 dB above the average noise at a control channel frequency. If the sweep is flat, the problem is the antenna or the site is out of range.

---

5. First test by hand

Close anything else using the dongle, then run (from the `apps` folder):

```
./rx.py --args 'rtl=0' -N 'LNA:40' -f 771.79375e6 -t -q 0 -S 960000 -D cqpsk -2 -U -v 5
```

What each part means:

Option	Meaning
`--args 'rtl=0'`	Use the first RTL-SDR dongle
`-N 'LNA:40'`	Tuner gain (try 30 to 49)
`-f 771.79375e6`	Control channel frequency (replace with yours)
`-t`	Start following trunk traffic
`-S 960000`	Sample rate (see section 9)
`-D cqpsk`	Simulcast demodulator
`-2`	Enable Phase 2 (TDMA) voice
`-U`	Play audio through the local sound device
`-v 5`	Verbose logging

Success looks like: the top line fills in WACN and SYSID, `tsbks` counts up, and active talkgroups and voice frequencies appear. If WACN and SYSID stay at `0x-1` and `tsbks` stays at 0, it is not decoding. See troubleshooting.

Keys in the OP25 terminal: `f` enter frequency, `h` hold talkgroup, `s` skip, `l` lock out, `q` quit.

Optional: add `--nocrypt` to skip encrypted calls instead of playing noise.

---

6. Audio

- **Bluetooth earbuds/speaker (easy and clean):** click the Bluetooth icon in the taskbar, put the earbuds in pairing mode, pair them, and select them as the output. In `bluetoothctl`, run `trust <MAC>` so they reconnect automatically. Turn the earbuds on before or just after powering the Pi.
- **USB sound dongle:** plug in, find the card with `aplay -l`, and add `-O plughw:CARD,0` to the `rx.py` command.
- **3.5 mm jack:** works, but is noisy and distorts if the volume is too high. Set the Pi volume low and use the speaker's own volume:

```
amixer sset 'Headphones' 40%
```

Test the output first:

```
speaker-test -t wav -c 2
```

Remember: you will only hear sound during a call. Many police talkgroups are encrypted and will be silent (or garbled if you don't use `--nocrypt`). Fire, EMS and city services are usually clear.

---

7. Make it start automatically on power-up

This is the part that causes the most grief. Use the **desktop autostart** method. It runs after login, when USB and audio are already up. (Cron `@reboot` and systemd services run too early or without audio access and are much harder to get right.)

Step 1: create the launcher script

```
nano ~/op25/op25/gr-op25_repeater/apps/scanner.sh
```

Contents (replace the frequency with yours):

```
#!/bin/bash
sleep 10
cd /home/YOURUSER/op25/op25/gr-op25_repeater/apps
./rx.py --args 'rtl=0' -N 'LNA:40' -f 771.79375e6 -t -q 0 -S 960000 -D cqpsk -2 -U -v 5
```

Replace `YOURUSER` with your username (case matters, `personal` is not `Personal`). Save with Ctrl+O, Enter, Ctrl+X.

Make it executable:

```
chmod +x ~/op25/op25/gr-op25_repeater/apps/scanner.sh
```

Step 2: test it by hand

```
~/op25/op25/gr-op25_repeater/apps/scanner.sh
```

It must work when run manually before you automate it.

Step 3: create the autostart entry (paste the whole block; do not use `printf` with `\n`, which can break when copied from a phone)

```
mkdir -p ~/.config/autostart
cat > ~/.config/autostart/scanner.desktop << 'EOF'
[Desktop Entry]
Type=Application
Name=Scanner
Exec=lxterminal -e /home/YOURUSER/op25/op25/gr-op25_repeater/apps/scanner.sh
EOF
```

Check it printed as four separate lines:

```
cat ~/.config/autostart/scanner.desktop
```

Step 4: make the Pi log in by itself

```
sudo raspi-config
```

System Options, Boot / Auto Login, Desktop Autologin.

Step 5: reboot and wait about 40 seconds. A terminal window with OP25 should open and start decoding on its own.

If you previously added an `@reboot` line to cron, remove it (`crontab -e`) so two copies don't fight over the dongle.

---

8. Running on a battery bank

- Use a bank and cable rated for at least 5V / 2.5A. Weak power causes dropouts and the dongle failing to start.
- While running, check for undervoltage:

```
vcgencmd get_throttled
```

`0x0` is good. Anything else means power problems.

- Headless (no screen) still works as long as desktop autologin is on. Test once with the monitor unplugged.
- Keep the dongle away from the bank and the Pi's hot spots. A short USB extension cable lets you mount it with air around it.
- Support the dongle so it is not levering on the USB port.

---

9. Sample rate: what it does

The sample rate is the width of spectrum the dongle sees at once. OP25 can only follow voice channels inside that window, centered on the control channel.

Sample rate	Approx. window	Notes
960000	about +/-480 kHz	Lightest load and heat, but misses voice channels farther from the control channel
1800000 to 2000000	about +/-0.9 to 1 MHz	Good compromise
2400000	about +/-1.2 MHz	Best coverage, most heat and CPU

RTL-SDR dongles top out around 2.56 MHz, so values like 20,000,000 will fail. Count your zeros.

To change it in your script:

```
sed -i 's/-S 960000/-S 2000000/' ~/op25/op25/gr-op25_repeater/apps/scanner.sh
```

Restart OP25 afterwards. A change does not affect an already-running copy.

---

10. Troubleshooting

Symptom	Likely cause and fix
`[R82XX] PLL not locked!` at startup	Usually harmless on R820T dongles. Ignore it unless it repeats constantly.
WACN/SYSID stay `0x-1`, `tsbks 0`	Not decoding. Check the NAC is hex, the frequency is a real control channel for the site you can hear, and try gain 30, 40, 49.
Locks on a far site and never decodes	That site is out of range. Use the site nearest you.
`osmosdr source_c creation failure`	Dongle is busy or not ready. Run `pkill -f rx.py; pkill -f rtl_`, wait, replug, and relaunch. Close `rtl_test` and `rtl_power` first. Add `sleep 10` at the top of the script for boot.
Nothing at boot	Check the `.desktop` file is four separate lines, the script is executable, desktop autologin is on, and no old cron line exists.
Audio garbled or distorted	Lower the Pi volume (analog jack clips), try Bluetooth or a USB audio dongle, check `vcgencmd get_throttled`, lower the sample rate if the CPU is maxed.
Silence during activity	Encrypted talkgroup, or wrong audio output selected.
Drops out after 20 to 30 minutes	Heat or power. Add a heatsink, give the dongle air, use a better power supply.
Frequency error keeps drifting	Non-TCXO dongle warming up. A TCXO dongle fixes this.

When in doubt, run the script by hand and read the first error that appears. That error is almost always the real cause.

11. Optional: dedicated site file (`-T`)

Instead of `-f` and `-t`, you can load a sites file so OP25 knows the NAC and a list of control channels. Fields must be separated by real **tab** characters:

```
Sysname Control Channel List    Offset  NAC Modulation  TGID Tags File  Whitelist   Blacklist   Center Frequency
RWC_SimA    771.79375,771.68125,772.04375,772.18125 0   0x531   cqpsk   tags.tsv
```

Launch with `-T yourfile.tsv` instead of `-f ... -t`. Use one file per dongle and per site, because a single instance with several sites hops between them whenever one is quiet.

Full structure of the sites file

The file is tab-separated, and the extension is `.tsv`. The first row is the header, and every field after it is separated by a single tab character. A file with two sites looks like this:

```
Sysname Control Channel List    Offset  NAC Modulation  TGID Tags File  Whitelist   Blacklist   Center Frequency
RWC_SimA    771.79375,771.68125,772.04375,772.18125 0   0x531   cqpsk   tags.tsv
RWC_SimD    772.08125,772.33125,772.58125,772.83125 0   0x531   cqpsk   tags.tsv
```

Column	What goes in it
Sysname	Any label you want, with no spaces
Control Channel List	Control channel frequencies in MHz, separated by commas, with no spaces
Offset	Dongle frequency correction in Hz. Start at 0, then trim using the frequency error on the status line.
NAC	In hex, such as `0x531`
Modulation	`cqpsk` for simulcast sites
TGID Tags File	Optional file that names talkgroups
Whitelist	Optional file listing the only talkgroups to follow
Blacklist	Optional file listing talkgroups to skip
Center Frequency	Optional, so leave it blank

Notes:

- **Check the tabs:** copy and paste from a phone or chat app often turns tabs into spaces, which breaks the file. Run `cat -A yourfile.tsv`. Each field separator should show as `^I`, and each line should end in `$`. The most reliable way to create a file is `printf` with `\t` for tabs and `\n` for line breaks, typed in the terminal.
- **One site per dongle:** with both rows in one file, a single instance hops between the sites whenever one goes quiet. That hurts when one site is out of range, so use one file per dongle and site.
- **Tags file:** a minimal `tags.tsv` has one line, such as `2991<TAB>TG 2991`, with a real tab between the ID and the name.
- **Order matters:** list the strongest control channel first, since OP25 tries them in order until one locks.

---

12. Ideas for next steps

- A talkgroup whitelist so you only hear the groups you care about (fire dispatch, for example).
- A small I2C LCD showing the frequency and whether the call is clear or encrypted.
- Buttons for hold / skip / lock.
- A second dongle to monitor another site.

---

13. Legal note

Listening to unencrypted public safety radio is legal in most places, but laws differ, especially about using scanners in vehicles and about acting on what you hear. Check your local rules. This guide does not cover decrypting encrypted traffic, which is not possible without authorization and is not something to attempt.

Credit: built on OP25 (boatbod fork) and RadioReference system data. Always check RadioReference for current control channels, since systems get reconfigured.