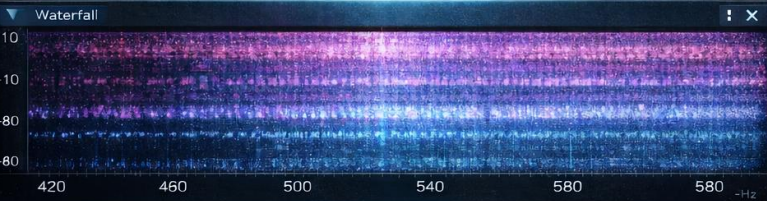


# DRONE-SCAN

RF DETECTION PROGRAM



Cursor: Bin=476 F=455.200 MHz P=-31.3 dB



RF SCAN



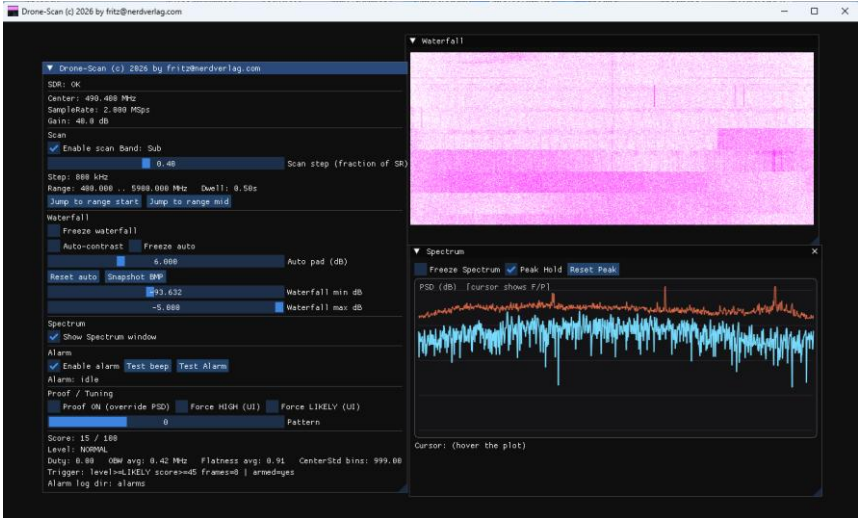
SPECTRUM & WATERFALL



ALARM DETECTION

© 2026 by fritz@nerdverlag.com

# Inhaltsverzeichnis



.....	8
Drone-Scan – Installation Guide .....	8
<b>Windows &amp; Linux</b> .....	8
1. Windows Installation (Windows 10 / 11, x64) .....	8
1.1 Überblick .....	8
1.2 Voraussetzungen .....	9
1.3 ZIP-Paket entpacken .....	9
1.4 WICHTIG: Programmstart über <code>run.bat</code> .....	10
1.5 Windows-Firewall-Meldung (normal!) .....	10
1.6 Treiberinstallation (Windows) .....	11
ADALM-PlutoSDR .....	11
RTL-SDR .....	11
HackRF One .....	12
1.7 Konfiguration Hardware (config.ini) .....	12

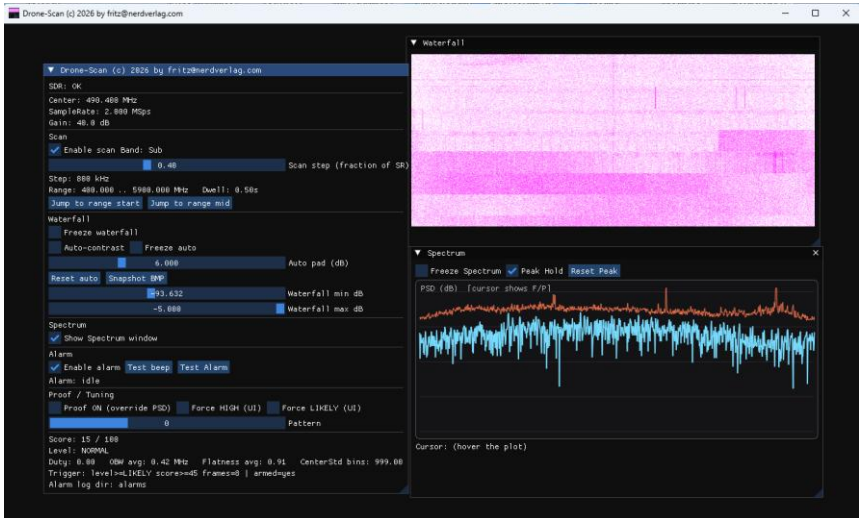
PlutoSDR.....	12
RTL-SDR.....	12
HackRF .....	12
2. Linux Installation (Ubuntu / Debian / Arch).....	13
2.1 Überblick.....	13
2.2 Voraussetzungen.....	13
2.3 Installation mit <code>install_environment.sh</code> (empfohlen) ..	14
Schritt 1 – Skript ausführbar machen .....	14
Schritt 2 – Installation starten.....	14
Was macht das Skript? .....	14
2.4 Manuelle Installation (optional).....	15
Ubuntu / Debian .....	15
Arch Linux.....	15
2.5 Build & Start .....	16
2.6 USB-Zugriff & Rechte .....	16
2.7 SDR-Erkennung testen .....	16
2.8 Konfiguration (Linux identisch zu Windows) .....	17
PlutoSDR.....	17
RTL-SDR.....	17
HackRF .....	17
3. Wichtiger Hinweis (Linux) .....	17
4. Typische Fehler & Lösungen (Kurz) .....	18
Drone-Scan – Benutzerhandbuch .....	19
5. Was ist Drone-Scan?.....	19
6. Grundprinzip (wichtig zum Verständnis) .....	19
7. Unterstützte Hardware.....	20

7.1 ADALM-PlutoSDR (empfohlen).....	20
7.2 HackRF One.....	21
7.3 RTL-SDR (eingeschränkt!) .....	21
8. Start & Installation (Windows) .....	21
8.1 Wichtig: Start immer über <code>run.bat</code> .....	21
8.2 Windows-Firewall Meldung (kein Grund zur Sorge) .....	22
9. Kapitel: Konfiguration ( <code>config.ini</code> ).....	23
9.1. Abschnitt [ <code>sdr</code> ] – SDR-Hardware & Basisparameter.....	23
<b>Parameterbeschreibung</b> .....	23
<b>Hinweise</b> .....	24
9.2. Abschnitt [ <code>scan</code> ] – Frequenzscan .....	24
<b>Grundprinzip</b> .....	24
<b>Parameterbeschreibung</b> .....	24
<b>Beispiel Schrittweite</b> .....	25
9.3. Abschnitt [ <code>ui</code> ] – Darstellung .....	25
9.4. Abschnitt [ <code>alarm</code> ] – Akustischer Alarm .....	25
<b>Verhalten</b> .....	26
9.5. Abschnitt [ <code>detection</code> ] – Detektor-Trigger (kritisch!) .....	26
<b>Bedeutung</b> .....	26
<b>Ziel</b> .....	27
9.6. Abschnitt [ <code>proof</code> ] – Proof / Nachweis-Modus .....	27
<b>Zweck</b> .....	27
<b>Pattern</b> .....	27
9.7. Wichtige Hinweise .....	28
10. Empfohlene Default-Konfiguration (praxisnah).....	28

Was passiert bei falschen Einstellungen? .....	29
11. Scan-Logik (sichtbar & bewusst langsam) .....	29
12. Waterfall-Anzeige .....	29
Manuelle Einstellung:.....	29
Auto-Contrast: .....	29
Freeze Waterfall:.....	30
Snapshot: .....	30
13. Detektion & Alarm .....	30
<b>Wichtige Erkenntnis:</b> .....	30
<b>Alarmbedingungen:</b> .....	30
<b>Deshalb gilt:</b> .....	31
14. Alarm-Steuerung .....	31
15. Scannen außerhalb ISM-Bänder.....	31
<b>Technisch:</b> .....	31
<b>Rechtlich:</b> .....	32
16. Typische Fehler & Lösungen.....	32
„Kein Alarm obwohl Signal sichtbar“ .....	32
„Waterfall leer“ .....	32
„RTL-SDR stürzt bei 2.4 GHz ab“ .....	32
17. Linux-Version .....	33
18. Wichtiger Hinweis .....	33
19. Hauptfenster „Drone-Scan“ .....	34
19.1 SDR-Status .....	34
20. Scan-Steuerung .....	35
<b>Bereich: Scan</b> .....	35
21. Waterfall-Anzeige .....	36

<b>Bereich: Waterfall</b> .....	36
22. Spectrum-Fenster.....	37
<b>Fenster: Spectrum</b> .....	37
23. Alarm-Steuerung.....	38
<b>Bereich: Alarm</b> .....	38
<b>Alarmverhalten (wichtig!)</b> .....	38
24. Proof / Tuning.....	39
<b>Bereich: Proof / Tuning</b> .....	39
25. Detektionsstatus.....	40
<b>Bereich: Detection output</b> .....	40
26. Fenster-Layout .....	40
27. Empfohlener Bedienablauf.....	40
28. Alarm Logging & Ereignisdokumentation .....	42
28.1 Wann wird geloggt? .....	42
28.2 Was wird gespeichert? .....	43
1) Spektrum-Snapshot (PNG) .....	43
2) Textdatei (TXT) .....	43
28.3 Speicherort .....	44
28.4 Dateinamen und Struktur .....	45
28.5 Unterschied: Alarm (latched) vs. Logging (einmalig) .....	45
29. Anhang .....	47
<b>1. Was du <i>jetzt schon</i> damit machen kannst (ohne Code-Änderung)</b> .....	47
<b>1.1 RF-Lagebild erstellen</b> .....	47
<b>1.2 Störquellen erkennen &amp; klassifizieren (indirekt)</b> .....	47
<b>1.3 Schulung &amp; Training</b> .....	48

<b>2. Dinge, die fast schon „auf der Hand liegen“ .....</b>	<b>48</b>
<b>2.1 Silent Perimeter Monitor .....</b>	<b>48</b>
<b>2.2 RF-Recorder (Post-Event-Analyse).....</b>	<b>49</b>
<b>2.3 Vergleich von Hardware / Antennen.....</b>	<b>49</b>
<b>3. Dinge, die man <i>gezielt</i> daraus machen kann .....</b>	<b>50</b>
<b>3.1 „Drohnenwahrscheinlichkeit“, nicht Alarm.....</b>	<b>50</b>
<b>3.2 Mobile Einsatzlösung.....</b>	<b>51</b>
<b>3.3 Ergänzung zu anderen Systemen .....</b>	<b>51</b>



## Drone-Scan – Installation Guide

# Windows & Linux

---

## 1. Windows Installation (Windows 10 / 11, x64)

### 1.1 Überblick

Drone-Scan für Windows wird als fertiges ZIP-Paket ausgeliefert.

- ✓ Kein Installer
- ✓ Keine Registry-Einträge

- ✓ Keine Systemänderungen
  - ✓ Alles läuft **Portable aus einem Ordner**
- 

## 1.2 Voraussetzungen

- Windows 10 oder Windows 11 (64-bit)
  - Administratorrechte **nur** für Treiberinstallation
  - Unterstützte SDR-Hardware:
    - ADALM-PlutoSDR
    - HackRF One
    - RTL-SDR (E4000 / R820T2, eingeschränkt)
- 

## 1.3 ZIP-Paket entpacken

1. ZIP-Datei herunterladen
2. Das Drone-Scan-Windows-x64.exe ausführen.
3. Das zip entpackt in z.B.

C:\Drone-Scan\

Der Ordner enthält u. a.:

- drone\_scan.exe
- run.bat
- config.ini
- benötigte .dll Dateien
- SoapySDR\modules0.8\ (SDR-Treiber)

**⚠ Dateien nicht einzeln verschieben** – alles muss im selben Verzeichnis bleiben.

---

## 1.4 WICHTIG: Programmstart über `run.bat`

👉 **Drone-Scan darf unter Windows immer nur über `run.bat` gestartet werden.**

### Warum?

Windows sucht DLLs zuerst im Systempfad.

`run.bat` stellt sicher, dass **die mitgelieferten, getesteten DLLs** verwendet werden.

### Richtig:

Doppelklick auf `run.bat`

### Falsch:

`drone_scan.exe` direkt starten

---

## 1.5 Windows-Firewall-Meldung (normal!)

Beim **ersten Start** erscheint möglicherweise:

*„Möchten Sie den Zugriff dieser App auf private Netzwerke zulassen?“*

### Grund:

- SoapySDR sucht beim Start nach SDR-Geräten
- ADALM-PlutoSDR kann über USB **oder Netzwerk** angesprochen werden

### Wichtig:

- Drone-Scan öffnet **keinen Server**
- Es werden **keine Daten gesendet**
- Kein Internetzugang nötig

### Empfohlene Auswahl:

- ✓ Privates Netzwerk → **Zulassen**
- ✗ Öffentliches Netzwerk → optional

Diese Meldung erscheint **nur einmal**.

---

## 1.6 Treiberinstallation (Windows)

### ADALM-PlutoSDR

1. Treiber herunterladen:
2. <https://github.com/analogdevicesinc/plutosdr-m2k-drivers-win/releases>
3. Installer ausführen
4. Pluto anschließen
5. Im Geräte-Manager prüfen (kein Warnsymbol)

---

### RTL-SDR

- Treiber wird **mitgeliefert**
  - Falls nötig: Zadig (WinUSB) verwenden  
(*nur bei Problemen*)
-

## HackRF One

1. HackRF-Treiber installieren:
    - o libusb + HackRF Driver (z. B. über Zadig oder Paket)
  2. libhackrf.dll muss im Drone-Scan-Ordner liegen
- 

## 1.7 Konfiguration Hardware (config.ini)

Die Hardware wird über `config.ini` ausgewählt:

Es liegt für jede mögliche Hardware eine sample `config.ini` vor.

### PlutoSDR

```
sdr.uri = ip:pluto.local  
driver=pluto
```

oder

```
sdr.uri = usb:*  
driver=pluto
```

### RTL-SDR

```
sdr.uri = driver=rtlsdr
```

### HackRF

```
sdr.uri = driver=hackrf
```

Tragen Sie nur `driver= ....` ein!

Nach Änderungen:

→ Programm neu starten

---

## 2. Linux Installation (Ubuntu / Debian / Arch)

### 2.1 Überblick

Unter Linux wird Drone-Scan **nativ ausgeführt**.

Zur Vereinfachung der Installation liegt ein **Setup-Skript** bei:

```
install_deps.sh
```

Dieses Skript installiert **alle benötigten Abhängigkeiten automatisch**

(Compiler, SDL2, SoapySDR, passende SDR-Module).

- ✓ Empfohlen für alle Nutzer
  - ✓ Spart Zeit
  - ✓ Vermeidet Versionsfehler
- 

### 2.2 Voraussetzungen

- Linux x86\_64
  - Internetzugang für Paketinstallation
  - sudo-Rechte
  - Unterstützte Distributionen:
    - Ubuntu / Debian
    - Arch Linux
    - kompatible Derivate
-

## 2.3 Installation mit `install_environment.sh` (empfohlen)

### Schritt 1 – Skript ausführbar machen

Das **Drone-Scan-linux-multi.tar.gz** entpacken

```
tar -xzf Drone-Scan-linux-multi.tar.gz
```

```
Falls erforderlich chmod +x install_deps.sh
```

---

### Schritt 2 – Installation starten

```
./install_deps.sh
```

oder (falls erforderlich):

```
sudo ./install_deps.sh
```

---

## Was macht das Skript?

`install_deps.sh` erledigt automatisch:

- Installation von:
  - GCC / Clang
  - CMake
  - SDL2
  - SoapySDR
  - Erkennt automatisch welches OS und installiert das erforderliche binary `drone_scan`.
- Installation der passenden SDR-Module:
  - RTL-SDR
  - HackRF
  - LimeSDR
  - PlutoSDR
- Setzt notwendige udev-Regeln (USB-Zugriff)

- Prüft SoapySDR-Installation mit:
- `SoapySDRUtil --find`

**⚠ Das Skript installiert keine proprietären Treiber**  
(z. B. Pluto Firmware bleibt unberührt).

---

## 2.4 Manuelle Installation (optional)

**⚠ Nur für fortgeschrittene Nutzer**

**⚠ Geht nur wenn Sie im Besitz der Sourcen sind**

Wenn das Skript nicht verwendet werden soll:

### Ubuntu / Debian

```
sudo apt update
sudo apt install -y \
  build-essential cmake git \
  libsdl2-dev \
  soapy-sdr libsoapysdr-dev \
  soapysdr-tools \
  soapysdr-module-rtlsdr \
  soapysdr-module-hackrf \
  soapysdr-module-lms7 \
  soapysdr-module-plutosdr
```

---

### Arch Linux

```
sudo pacman -S \
  base-devel cmake git \
  sdl2 \
  soapysdr \
  soapysdr-rtlsdr \
  soapysdr-hackrf \
  soapysdr-lms7 \
  soapysdr-plutosdr
```

---

## 2.5 Build & Start

```
mkdir build
cd build
cmake ..
make -j
```

Start:

```
./drone_scan
```

---

## 2.6 USB-Zugriff & Rechte

Empfohlen:

```
sudo usermod -a -G plugdev $USER
```

Danach:

```
logout / login
```

Alternativ (nicht empfohlen):

```
sudo ./drone_scan
```

---

## 2.7 SDR-Erkennung testen

```
SoapySDRUtil --find
```

Beispiel:

```
Found device 0
  driver = hackrf
```

Wenn hier kein Gerät erscheint → Treiberproblem, **kein Drone-Scan-Problem.**

---

## 2.8 Konfiguration (Linux identisch zu Windows)

Die `config.ini` ist **plattformunabhängig**.

Es liegt für jede mögliche Hardware eine `sample config.ini` vor.

Beispiele:

### PlutoSDR

```
sdr.uri = ip:pluto.local  
driver=pluto
```

### RTL-SDR

```
sdr.uri = driver=rtlsdr  
driver=rtlsdr
```

### HackRF

```
sdr.uri = driver=hackrf  
driver = hackrf
```

Tragen Sie nur `driver = ...` ein !

---

## 3. Wichtiger Hinweis (Linux)

Drone-Scan nutzt **keine speziellen Kernelmodule** und **verändert nichts am System**, außer:

- Installation von Paketen
- optionale `udev`-Regeln

Alle Änderungen sind **transparent und rückgängig machbar**.

---

## 4. Typische Fehler & Lösungen (Kurz)

### Programm startet nicht (Windows):

- `run.bat` verwenden
- alle DLLs im Ordner?

### Kein SDR gefunden:

- Treiber installiert?
- `SoapySDRUtil --find` prüfen
- `config.ini` korrekt?

### Kein Alarm:

- Signal  $\neq$  Drohne
- Konfidenz zu niedrig
- Detektionsparameter prüfen

# Drone-Scan – Benutzerhandbuch

## RF-basierte Drohnensignalerkennung mit SoapySDR

Version: 0.1.x

Plattformen: **Windows (x64), Linux (x86\_64)**

Unterstützte SDRs: **ADALM-Pluto, HackRF, RTL-SDR (eingeschränkt)**

---

## 5. Was ist Drone-Scan?

Drone-Scan ist ein **passives RF-Analysewerkzeug**, das Funkaktivität in typischen Drohnen-Frequenzbereichen analysiert und **drohnentypische Signalmuster** erkennt.

Das Programm:

- empfängt **keine Telemetrie**
- dekodiert **keine Remote-ID**
- steuert **keine Geräte**
- sendet **keine Daten ins Internet**

Drone-Scan erkennt **Signalverhalten**, nicht Drohnenmodelle.

---

## 6. Grundprinzip (wichtig zum Verständnis)

Drone-Scan arbeitet in vier Schritten:

1. **SDR empfängt IQ-Daten**
2. **FFT / PSD Analyse**
3. **Merkmal-Extraktion**  
(Bandbreite, Zeitverhalten, Spektralform)
4. **Bewertung (Score 0–100)**

Ein Alarm entsteht **nicht** durch „ein starkes Signal“, sondern durch:

- ausreichend **breite Signale**
- **zeitlich stabiles Verhalten**
- typische **Dynamik** (Duty-Cycle, Drift, Flachheit)

👉 Deshalb kann ein starkes Signal **ohne Alarm** auftreten  
👉 und ein schwächeres, aber typisches Signal **mit Alarm**

---

## 7. Unterstützte Hardware

### 7.1 ADALM-PlutoSDR (empfohlen)

- Frequenzbereich: **70 MHz – 6 GHz**
- Volle Unterstützung aller Scan-Bänder
- Sehr stabil
- Beste Wahl für 2.4 GHz / 5.8 GHz

#### Empfohlene Nutzung:

- USB oder Netzwerk (pluto.local)
  - Fester `sdr.uri` empfohlen
-

## 7.2 HackRF One

- Frequenzbereich: **1 MHz – 6 GHz**
  - Unterstützt **2.4 GHz & 5.8 GHz**
  - Geringere Dynamik als Pluto
  - Sehr flexibel
- 

## 7.3 RTL-SDR (eingeschränkt!)

- Frequenzbereich: ca. **24 MHz – 1.7 GHz**
- **KEIN 2.4 GHz**
- **KEIN 5.8 GHz**
- Nur **Sub-GHz** sinnvoll

👉 Drone-Scan **blockiert automatisch** unzulässige Scan-Bänder bei RTL-SDR.

---

## 8. Start & Installation (Windows)

### 8.1 Wichtig: Start immer über `run.bat`

**Nicht direkt `drone_scan.exe` starten!**

`run.bat` sorgt für:

- korrekte DLL-Pfadsetzung
- korrektes Laden der SoapySDR-Module
- reproduzierbares Verhalten

- **✗ absolut keine Änderung am Windows System!**
- 

## 8.2 Windows-Firewall Meldung (kein Grund zur Sorge)

Beim **ersten Start** erscheint:

„Zugriff auf öffentliche/private Netzwerke erlauben?“

**Warum?**

- SoapySDR sucht nach Geräten
- ADALM-Pluto kann per **Netzwerk (pluto.local)** angesprochen werden

**Wichtig:**

- **✗ Drone-Scan öffnet keinen Server**
- **✗ keine Telemetrie**
- **✗ kein Internet-Traffic**

**Empfehlung:**

- **✓ Privates Netzwerk: Zulassen**
  - **✗ Öffentlich: optional**
-

## 9. Kapitel: Konfiguration (config.ini)

Die Datei `config.ini` steuert **Hardware, Scanverhalten, Detektion, Alarmierung und Darstellung** von *Drone-Scan*. Alle Einstellungen werden **beim Programmstart** geladen.

⚠ Änderungen an der `config.ini` erfordern einen **Neustart** der Anwendung.

---

### 9.1. Abschnitt `[sdr]` – SDR-Hardware & Basisparameter

```
[sdr]
uri =
driver =
sample_rate = 2000000
bandwidth = 2000000
fft_size = 1024
gain = 20
center_freq = 433000000
```

### Parameterbeschreibung

Parameter	Bedeutung
<code>uri</code>	Optionale Geräteadresse (z. B. <code>ip:192.168.2.1</code> für PlutoSDR)
<code>driver</code>	Optionaler Treiber-Hinweis ( <code>plutosdr</code> , <code>hackrf</code> , <code>rtl_sdr</code> , leer = automatisch)
<code>sample_rate</code>	Abtastrate in Hz
<code>bandwidth</code>	RF-Bandbreite in Hz (meist identisch zur Sample-Rate)
<code>fft_size</code>	FFT-Größe für Analyse und Darstellung
<code>gain</code>	Empfangsverstärkung in dB

Parameter	Bedeutung
<code>center_freq</code>	Start-Mittenfrequenz (wird beim Scan automatisch überschrieben)

## Hinweise

- Unterstützt **PlutoSDR, HackRF, RTL-SDR**
- `center_freq` dient nur als **Initialwert**, nicht als Scan-Grenze

## 9.2. Abschnitt `[scan]` – Frequenzscan

```
[scan]
enabled = true

start_freq_sub = 433000000
end_freq_sub   = 434800000

dwell_time = 0.35
step_frac  = 0.40
```

## Grundprinzip

➡ **Es darf immer nur EIN Frequenzband aktiv sein.**  
 Ein Band ist aktiv, wenn **Start- und Endfrequenz > 0** sind.

### Priorität der Bänder:

1. Sub-GHz
2. 2.4 GHz
3. 5.8 GHz

## Parameterbeschreibung

<b>Parameter</b>	<b>Bedeutung</b>
enabled	Scan aktiv / inaktiv
start_freq_sub / end_freq_sub	Sub-GHz Scanbereich 2.4 GHz Scanbereich 5.8 GHz Scanbereich
dwell_time	Verweildauer pro Frequenzsprung (Sekunden)
step_frac	Schrittweite relativ zur Sample-Rate

## Beispiel Schrittweite


```
sample_rate = 2.000.000 Hz
step_frac   = 0.40
→ Schrittweite = 800 kHz
```

---

### 9.3. Abschnitt [ui] – Darstellung

```
[ui]
waterfall_min_db = -95
waterfall_max_db = -25
```

<b>Parameter</b>	<b>Bedeutung</b>
waterfall_min_db	Untere dB-Grenze der Wasserfall-Darstellung
waterfall_max_db	Obere dB-Grenze

 Diese Werte dienen als **Startwerte**.  
Auto-Contrast kann sie während des Betriebs dynamisch anpassen.

---

### 9.4. Abschnitt [alarm] – Akustischer Alarm

```
[alarm]
```

```
enabled = true
repeat_interval = 2.0
sound_frequency = 1000
sound_duration = 0.4
```

## Verhalten

- Alarm ist **latching** (bleibt aktiv)
- Alarm endet **nur nach manueller Quittierung**
- Tonwiedergabe über **SDL Audio** (Windows & Linux)

Parameter	Bedeutung
enabled	Alarmfunktion aktiv
repeat_interval	Zeit zwischen Beep-Wiederholungen
sound_frequency	Tonfrequenz in Hz
sound_duration	Dauer eines Beeps in Sekunden

---

## 9.5. Abschnitt [detection] – Detektor-Trigger (kritisch!)

```
[detection]
trigger_level = LIKELY
trigger_score = 45
trigger_frames = 8
rearm_release_frames = 20
```

## Bedeutung

Parameter	Erklärung
trigger_level	Mindest-Detektionslevel (LIKELY oder HIGH)
trigger_score	Mindest-Score (0–100)
trigger_frames	Anzahl aufeinanderfolgender Treffer
rearm_release_frames	Frames ohne Treffer bis Wieder-

## Parameter

## Erklärung

Scharfschaltung

### Ziel

- **keine Zufallsalarme**
- **keine Flattereffekte**
- stabile, reproduzierbare Auslösung

---

### 9.6. Abschnitt [proof] – Proof / Nachweis-Modus

```
[proof]  
enabled = false  
pattern = 0
```

### Zweck

Der Proof-Mode erzeugt **synthetische, drohnenähnliche Spektren** zur:

- Funktionsprüfung
- Demonstration
- Dokumentation
- Schulung
- Kundennachweis

**⚠️ Kein HF-Senden**, reine Software-Simulation.

### Pattern

#### Wert

#### Charakteristik

0 FHSS + breite Belegung (DJI / ELRS ähnlich)

Wert	Charakteristik
1	Mehrträger / OFDM-ähnlich
2	Sehr breites, rauschähnliches Band

---

## 9.7. Wichtige Hinweise

- Änderungen an `config.ini` wirken **erst nach Neustart**
  - Scan und Proof-Mode schließen sich **nicht aus**
  - UI-„Force HIGH / LIKELY“ beeinflusst **nur die Anzeige**, nicht den Alarm
  - Alarm bleibt aktiv, **bis quittiert**
- 

## 10. Empfohlene Default-Konfiguration (praxisnah)

```
[detection]
trigger_level = LIKELY
trigger_score = 45
trigger_frames = 8
rearm_release_frames = 20
```

➔ Stabil, realistisch, demo-tauglich, ohne Zwang zu „Force HIGH“.

## Was passiert bei falschen Einstellungen?

- RTL-SDR + 2.4 GHz → **Scan wird automatisch deaktiviert**
  - Frequenzen außerhalb Hardware-Bereich → **kein Crash**
  - Scan deaktiviert → **feste Center-Frequenz**
- 

## 11. Scan-Logik (sichtbar & bewusst langsam)

- Scan springt **schrittweise**
- Schrittweite  $\approx \text{sample\_rate} \times \text{Faktor}$
- Absichtlich **nicht zu schnell**, damit:
  - Wasserfall visuell nachvollziehbar bleibt
  - Detektion stabiler ist

## 12. Waterfall-Anzeige

### Manuelle Einstellung:

- `waterfall_min_db`
- `waterfall_max_db`

### Auto-Contrast:

- passt Kontrast automatisch an
- basiert auf aktuellem PSD-Minimum/Maximum

- **Freeze Auto** hält aktuellen Kontrast fest

### Freeze Waterfall:

- stoppt visuelle Aktualisierung
- SDR & Detektion laufen weiter

### Snapshot:

- speichert aktuellen Waterfall als BMP
  - hilfreich für Dokumentation / Analyse
- 

## 13. Detektion & Alarm

### Wichtige Erkenntnis:

**Nicht jedes „starke Signal“ ist eine Drohne**

Detektor bewertet u.a.:

- belegte Bandbreite
- zeitliche Stabilität
- spektrale Flachheit
- Drift des Zentrums

### Alarmbedingungen:

- Score > Schwellwert
- über mehrere Frames stabil
- typisch drohnenähnlich

## Deshalb gilt:

- WLAN → meist **kein Alarm**
  - schmalbandige Träger → **kein Alarm**
  - modulierte, stabile Breitbandsignale → **Alarm**
- 

## 14. Alarm-Steuerung

```
[alarm]
enabled = true
repeat_interval = 2.0
sound_frequency = 1000
sound_duration = 0.4
```

### UI-Funktionen:

- Alarm an/aus
- Test-Beep
- Alarm quittieren

Alarmton läuft **asynchron**, blockiert die UI nicht.

---

## 15. Scannen außerhalb ISM-Bänder

### Technisch:

- ✓ möglich
- ✓ kein Problem für SDR
- ✓ Detektor funktioniert

## Rechtlich:

- ⚠ abhängig vom Land
- ⚠ Empfang meist erlaubt
- ⚠ **kein Senden!**

Drone-Scan **sendet niemals**, bleibt passiv.

---

## 16. Typische Fehler & Lösungen

### „Kein Alarm obwohl Signal sichtbar“

- Bandbreite zu klein
- Signal zu instabil
- Gain falsch eingestellt
- Detektionsparameter zu streng

### „Waterfall leer“

- Gain zu niedrig
- falscher Frequenzbereich
- Scan deaktiviert

### „RTL-SDR stürzt bei 2.4 GHz ab“

- Hardware-Limit → korrekt
  - Lösung: Sub-GHz verwenden
-

## 17. Linux-Version

- gleiche `config.ini`
- gleiche Scan-Logik
- gleiche UI
- gleiche Module

Unterschiede:

- kein Windows-Beep → Terminal-Beep
  - udev-Rechte nötig für SDR
- 

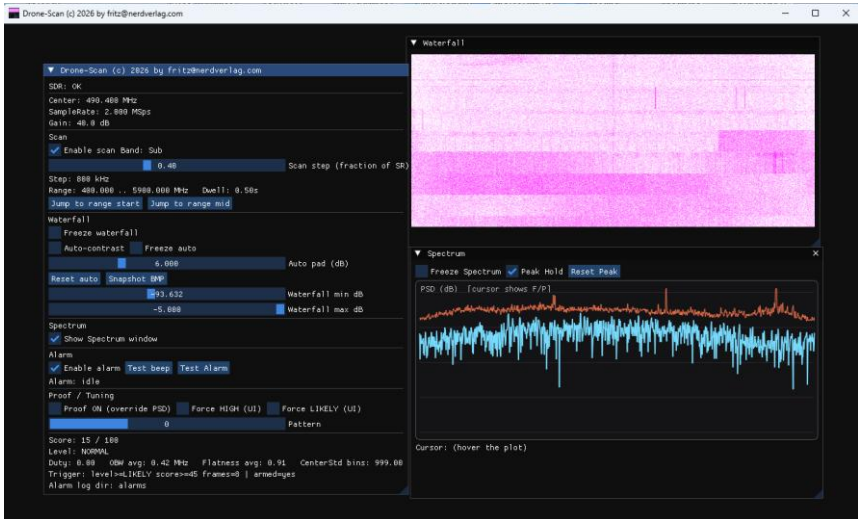
## 18. Wichtiger Hinweis

Drone-Scan ist:

- **Analyse-Tool**
- **kein Überwachungssystem**
- **keine Identifikation**
- **keine rechtliche Bewertung**

Ergebnisse dienen der **Situationsbewertung**, nicht als Beweis.

# 19. Hauptfenster „Drone-Scan“



Das Hauptfenster ist die **zentrale Steuer- und Statusanzeige**.

## 19.1 SDR-Status

Oben im Fenster wird der Zustand des SDR-Geräts angezeigt:

- **SDR: OK**  
→ Gerät korrekt erkannt und in Betrieb
- **SDR: NOT OK**  
→ Gerät nicht verfügbar oder Initialisierung fehlgeschlagen  
→ Ursache wird darunter angezeigt

Zusätzlich sichtbar:

- Aktuelle **Mittenfrequenz**
  - **Sample-Rate**
  - **Gain**
- 

## 20. Scan-Steuerung

### Bereich: Scan

Hier wird der Frequenzscan kontrolliert.

#### Elemente:

- **Enable scan**  
Aktiviert oder deaktiviert den automatischen Frequenzscan
- **Band:**  
Zeigt das aktuell aktive Scanband  
(Sub, 2.4G, 5.8G – gemäß `config.ini`)
- **Scan step (fraction of SR)**  
Regelt die Schrittweite pro Hop  
(relativ zur Sample-Rate)
- **Step:**  
Anzeige der effektiven Schrittweite in kHz
- **Range:**  
Aktueller Scanbereich in MHz
  - Verweildauer pro Schritt (Dwell Time)

#### Tasten:

- **Jump to range start**  
Springt sofort an den unteren Rand des Scanbereichs

- **Jump to range mid**  
Springt in die Mitte des Scanbereichs
- 

## 21. Waterfall-Anzeige

### **Bereich: Waterfall**

Der Waterfall zeigt die **zeitliche Entwicklung des Spektrums**.

### **Bedienelemente:**

- **Freeze waterfall**  
Friert die Anzeige ein (keine neuen Zeilen)
- **Auto-contrast**  
Passt dB-Skalierung automatisch an das Signal an
- **Freeze auto**  
Hält die aktuell ermittelte Auto-Skalierung fest
- **Auto pad (dB)**  
Abstand zwischen Signalspitzen und Farbsättigung

### **Tasten:**

- **Reset auto**  
Setzt Auto-Kontrast neu zurück
- **Snapshot BMP**  
Speichert den aktuellen Waterfall als BMP-Datei

### **Manuelle Skalierung:**

- **Waterfall min dB**
- **Waterfall max dB**

---

## 22. Spectrum-Fenster

### Fenster: Spectrum

Das Spectrum zeigt das **momentane Leistungsspektrum**.

#### Funktionen:

- **Live-Kurve** (aktuelles PSD)
- **Peak Hold** (Maximalwert je Bin)
- **Cursor / Marker** bei Mausposition

#### Anzeige bei Mauszeiger:

- FFT-Bin
- Absolute Frequenz (MHz)
- Leistung (dB)

#### Bedienelemente:

- **Freeze Spectrum**  
Hält das aktuelle Spektrum an
- **Peak Hold**  
Aktiviert Maximalwert-Speicherung
- **Reset Peak**  
Löscht gespeicherte Peak-Werte

## 23. Alarm-Steuerung

### Bereich: Alarm

Der Alarm meldet erkannte Drohnensignale **akustisch und visuell**.

#### Elemente:

- **Enable alarm**  
Aktiviert die Alarmfunktion
  - **Test beep**  
Spielt einen kurzen Testton
  - **Test Alarm**  
Simuliert einen Alarm (inkl. Ton)
- 

#### Alarmverhalten (wichtig!)

- Alarm ist **latching**
- Alarm bleibt **aktiv**, bis er quittiert wird
- Kein automatisches Abschalten

#### Anzeige:

-  **ALARM ACTIVE (latched)**

#### Taste:

- **Acknowledge alarm**  
Quittiert den Alarm und schaltet ihn zurück

Nach der Quittierung ist eine **kurze Re-Arm-Phase** aktiv, um sofortige Wiederalarme zu verhindern.

---

## 24. Proof / Tuning

### **Bereich: Proof / Tuning**

Dieser Bereich dient **Test, Demonstration und Schulung**.

#### **Elemente:**

- **Proof ON (override PSD)**  
Aktiviert synthetische, drohnenähnliche Signale
- **Force HIGH (UI)**  
Erzwingt Anzeige „HIGH“ (nur visuell)
- **Force LIKELY (UI)**  
Erzwingt Anzeige „LIKELY“ (nur visuell)

#### **⚠ Wichtig:**

Diese Optionen beeinflussen **nicht den Alarm** – sie sind rein visuell.

- **Pattern (0–2)**  
Auswahl des synthetischen Signaltyps
-

## 25. Detektionsstatus

### Bereich: Detection output

Anzeige der aktuellen Auswertung:

- **Score** (0–100)
- **Level** (NORMAL, LIKELY, HIGH)
- **Duty**
- **Occupied Bandwidth (OBW)**
- **Spectral Flatness**
- **Center Stability**

Zusätzlich:

- Aktive **Trigger-Parameter**
  - Status **armed / not armed**
- 

## 26. Fenster-Layout

- Alle Fenster sind **frei verschiebbar**
  - Fenster können **geschlossen und wieder geöffnet** werden
  - Größen sind **dynamisch anpassbar**
- 

## 27. Empfohlener Bedienablauf

1. SDR anschließen

2. Scan aktivieren
3. Waterfall & Spectrum beobachten
4. Alarm aktivieren
5. Bei Alarm:
  - visuelle Kontrolle
  - Alarm quittieren
6. Optional:
  - Snapshot speichern
  - Proof-Mode für Demonstration nutzen

## 28. Alarm Logging & Ereignisdokumentation

Drone-Scan kann bei einem **echten Alarmereignis** automatisch eine **Ereignisdokumentation** erstellen. Diese besteht aus einem **Spektrum-Snapshot (PNG)** und einer **Textdatei (TXT)** mit allen relevanten Parametern. Ziel ist, dass ein Alarm später **nachvollziehbar** bleibt – inklusive *Zeitpunkt, Frequenz* und *Detektionsbewertung*.

---

### 28.1 Wann wird geloggt?

Ein Log-Eintrag wird **nur dann** erzeugt, wenn die **Detektionslogik** einen Alarm **tatsächlich auslöst**.

Das bedeutet:

✓ Logging passiert bei:

- einem Alarm durch die Detektor-Triggerbedingungen (Trigger-Level + Trigger-Score + Trigger-Frames)

✗ Logging passiert **nicht** bei:

- **Test beep**
- **Test Alarm**
- **Proof Mode**
- **Force HIGH / Force LIKELY** (UI-Debug / Visualisierung)
- reinem „Signal sichtbar“ ohne erfüllte Triggerbedingungen

Wichtig: Ein sichtbares Signal im Spektrum oder Waterfall führt **nicht automatisch** zu einem Log. Erst wenn die Detektionslogik die eingestellten Kriterien erfüllt, wird geloggt.

---

## 28.2 Was wird gespeichert?

Pro Alarmereignis werden zwei Dateien erzeugt:

### 1) Spektrum-Snapshot (PNG)

- Ein PNG-Bild des Spektrums zum Alarmzeitpunkt
- Zweck:
  - Sichtbarer Nachweis „wie das Spektrum aussah“
  - Dokumentation der belegten Bandbreite / Peaks / Struktur
  - Vergleichbarkeit mit späteren Ereignissen

### 2) Textdatei (TXT)

Die Textdatei enthält die wichtigsten Metadaten, z.B.:

- Datum & Uhrzeit (Zeitstempel)
- aktuelle Mittenfrequenz (Center Frequency) zum Alarmzeitpunkt
- aktives Scanband (Sub / 2.4G / 5.8G)
- Detektionsstatus:
  - Level (NORMAL / LIKELY / HIGH)
  - Score (0..100)
- Trigger-Konfiguration:
  - trigger\_level
  - trigger\_score
  - trigger\_frames

- SDR/Signalparameter:
  - Sample Rate
  - FFT Größe
  - Gain
  - ggf. Scan-Einstellungen (Dwell, Step)

So kann man später exakt nachvollziehen:

- „Wann war es?“
- „Wo im Spektrum?“
- „Warum wurde Alarm ausgelöst?“
- „Welche Einstellungen waren aktiv?“

---

## 28.3 Speicherort

Standardmäßig wird im Programmverzeichnis (oder unterhalb davon) ein Ordner für Logs verwendet, typischerweise z.B.:

- `./alarms/`
- oder im Arbeitsverzeichnis, aus dem Drone-Scan gestartet wurde

Empfehlung: Den Ordner **nicht** manuell löschen, wenn Logs zur Nachvollziehbarkeit gebraucht werden.

Wenn du möchtest, kann man später zusätzlich eine Option in `config.ini` ergänzen, um den Log-Ordner frei zu wählen (z.B. `alarm.log_dir = alarms`).

## 28.4 Dateinamen und Struktur

Die Dateien werden so benannt, dass sie:

- eindeutig sind (kein Überschreiben)
- chronologisch sortierbar sind
- Frequenzbezug enthalten

Beispiel:

- `alarm_2025-12-29_08-41-12_433.920MHz.txt`
- `alarm_2025-12-29_08-41-12_433.920MHz_spectrum.png`

Bedeutung:

- Datum/Zeit = exakter Alarmzeitpunkt
  - MHz-Angabe = Mittenfrequenz zum Alarmmoment
  - `_spectrum.png` = Spektrum-Bild
- 

## 28.5 Unterschied: Alarm (latched) vs. Logging (einmalig)

Drone-Scan nutzt einen **latching Alarm**:

- Sobald Alarm aktiv ist, bleibt er aktiv, bis er quittiert wird.
- Das Alarm-Logging ist davon getrennt:
  - In der Regel wird **ein** Log-Eintrag beim Auslösen erzeugt
  - nicht permanent alle Frames während der Alarm anliegt

Das verhindert „Log-Spam“ (tausende Dateien pro Minute), wenn der Alarm länger ansteht.

## 29. Anhang

# 1. Was du *jetzt schon* damit machen kannst (ohne Code-Änderung)

## 1.1 RF-Lagebild erstellen

Drone-Scan ist im Kern ein **RF-Situational-Awareness-Tool**.

Du kannst damit:

- sehen **welche Bänder „leben“**
- erkennen **wann Aktivität beginnt / endet**
- vergleichen **Tageszeiten, Orte, Events**

👉 Ideal für:

- Geländeerkundung
- Vorab-Checks (z. B. Veranstaltung)
- Langzeitbeobachtung

---

## 1.2 Störquellen erkennen & klassifizieren (indirekt)

Auch wenn es „Drone-Scan“ heißt:

- WLAN
- Video-Links
- FHSS-Systeme
- breitbandige Störer
- defekte Geräte

...verhalten sich **anders** als Drohnen.

👉 Du lernst:

- was *kein* Drohnensignal ist
- was „typisch technisch“ vs. „typisch UAV“ aussieht

Das ist **praktisch Funk-Fingerprinting** (ohne ML).

---

## 1.3 Schulung & Training

Extrem unterschätzt, aber sehr wertvoll:

- Ausbildung von Personal
- Demonstration von RF-Grundlagen
- „So sieht 2.4 GHz wirklich aus“
- Vergleich verschiedener SDRs

Drone-Scan ist visuell, stabil und erklärbar → **perfekt für Training**

---

## 2. Dinge, die fast schon „auf der Hand liegen“

### 2.1 Silent Perimeter Monitor

- feste Frequenz
- Alarm bei Änderung
- kein Scan

## → RF-Wachposten

Einsatz:

- Dach
  - Perimeter
  - temporäre Installationen
  - kritische Infrastruktur (passiv!)
- 

## 2.2 RF-Recorder (Post-Event-Analyse)

Du hast schon:

- PSD
- Zeitbezug
- Detektionsmetriken
- Events loggen
- Offline analysieren

👉 „Was war da um 21:37?“

---

## 2.3 Vergleich von Hardware / Antennen

Drone-Scan ist ideal zum:

- Antennen vergleichen
- Positionen vergleichen
- Gain-Settings vergleichen
- Pluto vs HackRF vs RTL-SDR vergleichen

Du hast:

- gleiche Software
- gleiche Darstellung
- gleiche Logik

→ objektiver Vergleich

---

## 3. Dinge, die man *gezielt* daraus machen kann

### 3.1 „Drohnenwahrscheinlichkeit“, nicht Alarm

Du hast bereits:

- Score
- Level (NORMAL / LIKELY / HIGH)

Das schreit nach:

- Ampel-Logik
- Heatmap
- Zeitdiagramm

👉 Nicht „Alarm!“, sondern:

„RF-Aktivität mit hoher UAV-Ähnlichkeit“

Das ist **operativ viel besser**.

---

## 3.2 Mobile Einsatzlösung

Mit:

- Laptop
- Pluto oder HackRF
- Richtantenne

kannst du:

- Bereiche abgehen
- Hotspots finden
- Richtungsänderungen beobachten (manuell)

Kein Peilen, aber **sehr gutes Gefühl für Richtung & Nähe.**

---

## 3.3 Ergänzung zu anderen Systemen

Drone-Scan ist perfekt als:

- **passiver Sensor**
- **Vorwarnsystem**
- **Bestätigungssensor**

Kombinierbar mit:

- optischen Systemen
  - Radar
  - ADS-B
  - akustischen Sensoren
-