RDF-J/ECM-J SYSTEM
Radio Direction Finding — JANBLE

Overview · How It Works · Server · Pi Node · Security · ECM-J · API · Hardware · ← Dashboard

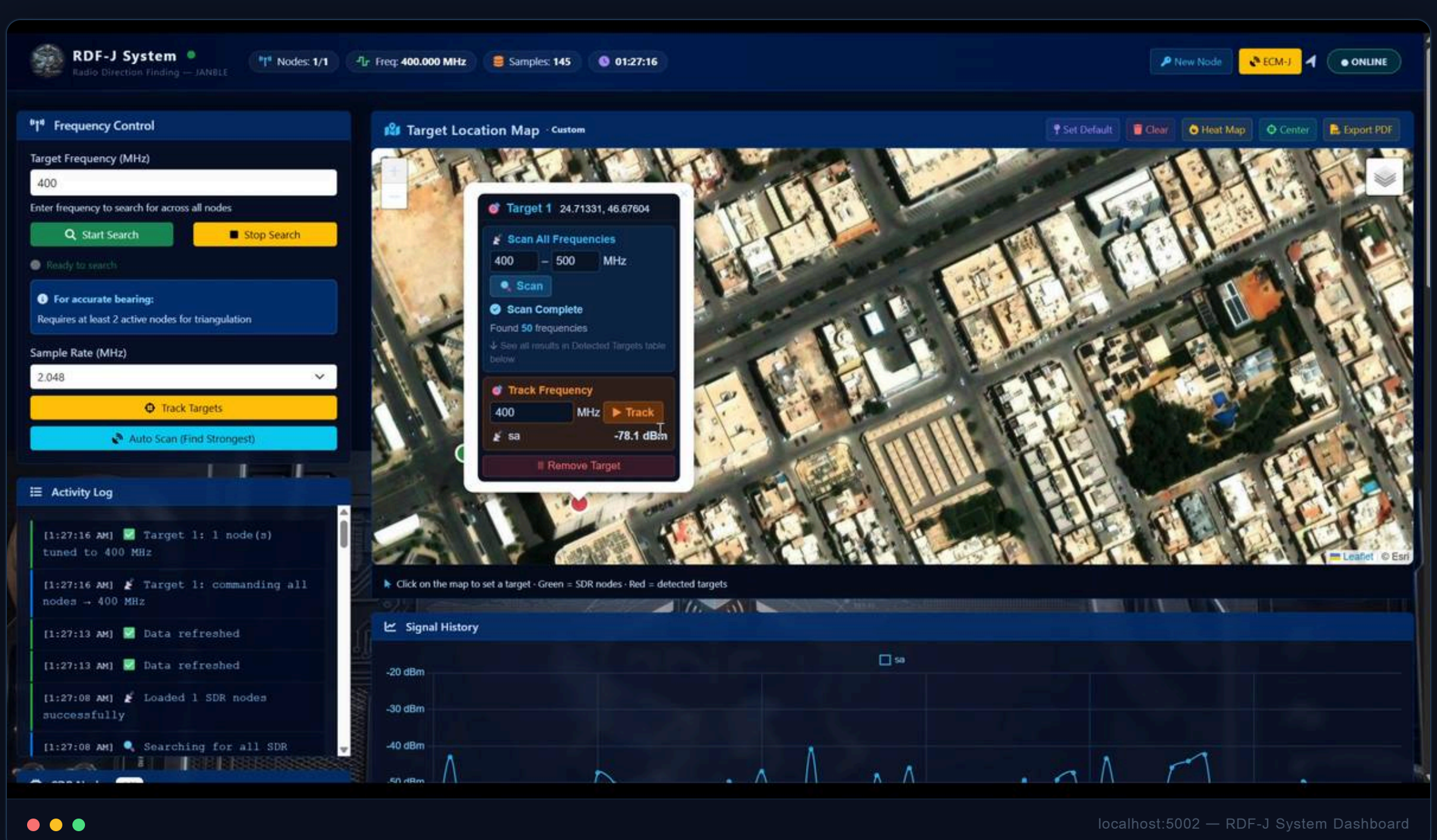
RDF-J/ECM-J SYSTEM

RDF-J Radio Direction Finding · ECM-J Electronic Counter Measures — JANBLE

A professional multi-node platform for radio signal direction finding using low-cost RTL-SDR receivers distributed on Raspberry Pi devices, with real-time triangulation and a live interactive dashboard.

Multi-Node SDR · Live Triangulation · WebSocket Real-Time · Secure API · SQLite Persistent · Waterfall Spectrum · Heat Map · Advanced Filters · ECM-J RF Jamming · HackRF TX Support · HMAC-SHA256

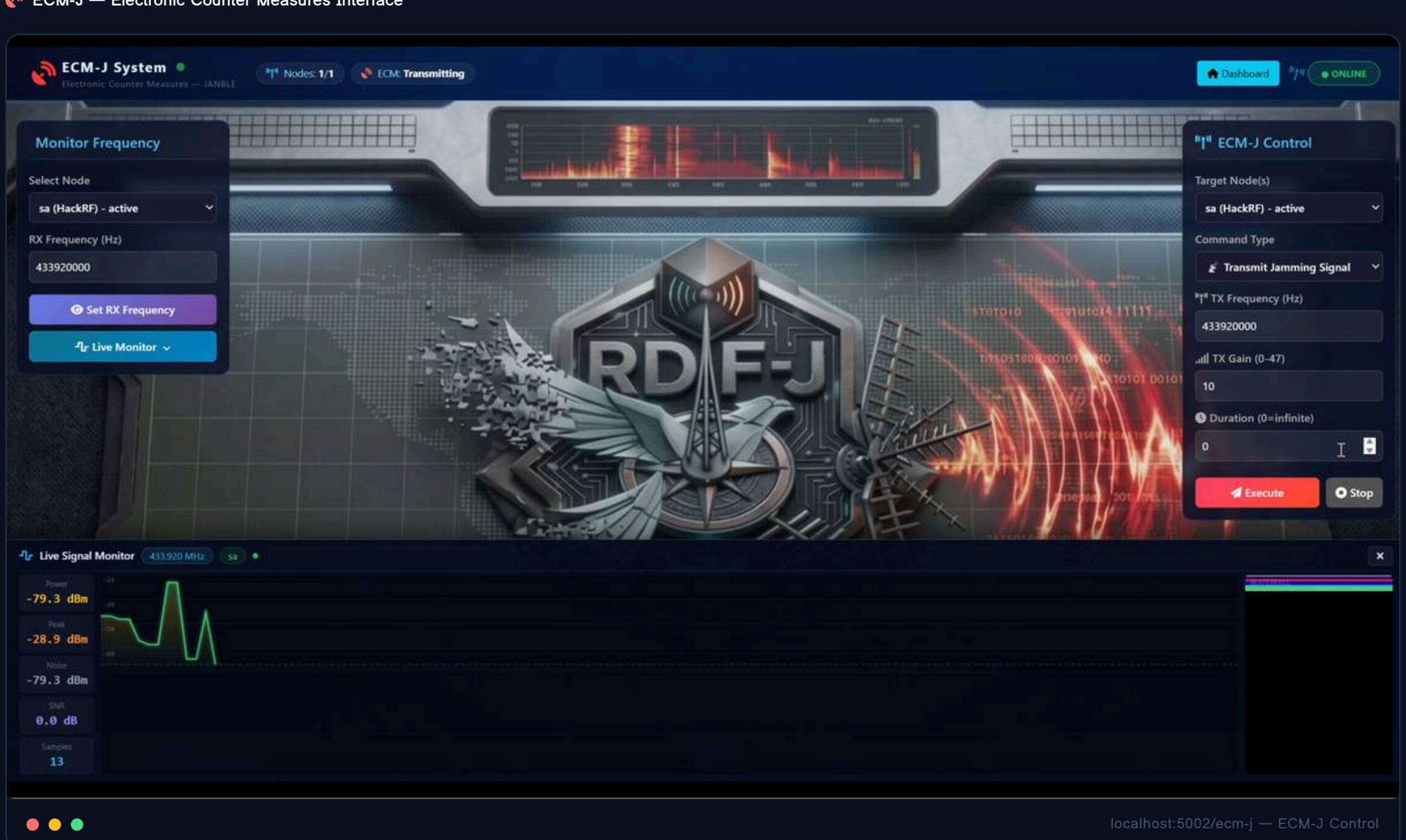
Main Dashboard — Server Web Interface



localhost:5002 — RDF-J System Dashboard

Interactive Map · Frequency Control · Signal History · Activity Log · Waterfall Display · Heat Map · Advanced Filters · Historical Data

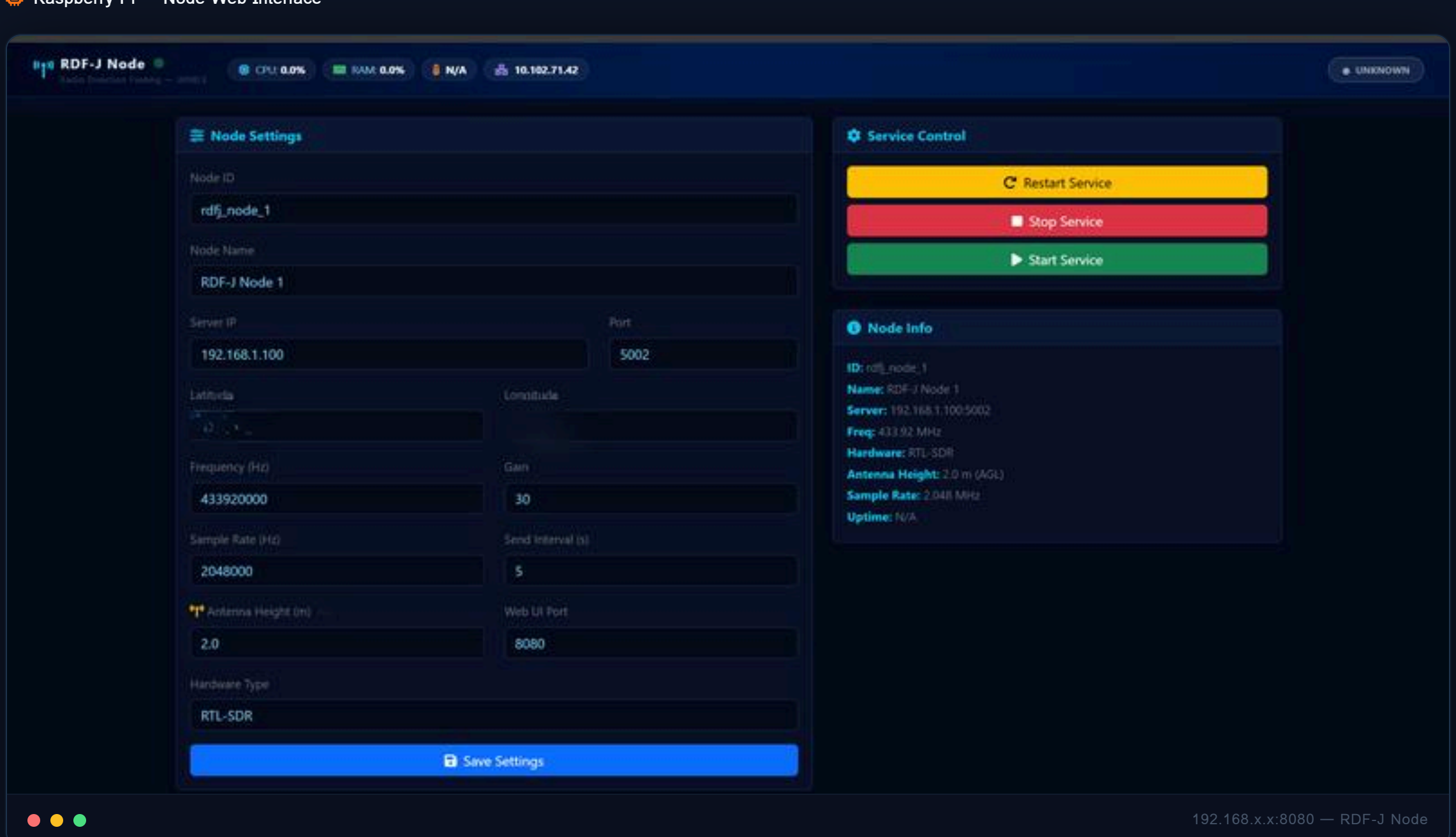
ECM-J — Electronic Counter Measures Interface



localhost:5002/ecm-j — ECM-J Control

RF Jamming TX · HackRF One · Monitor Frequency · ALL NODES · HMAC-SHA256

Raspberry Pi — Node Web Interface



192.168.x.x:8080 — RDF-J Node

Node Settings · Service Control · Node Info

How It Works

End-to-end flow from node registration to target location

01 Node Registration
Each Raspberry Pi registers with the server using a Registration Secret and receives a unique API Key stored locally.

02 SDR Scanning
A dedicated scan thread continuously measures signal strength (RSSI) at the configured frequency and sends readings to the server.

03 Heartbeat
Each node sends periodic heartbeats carrying CPU%, RAM%, temperature, and uptime. Missing heartbeats trigger an auto-offline flag.

04 Triangulation
The server intersects bearing lines from 2+ nodes using least-squares geometry to estimate the signal source location with a confidence score.

05 Real-Time Visualization
Waterfall spectrum shows frequency activity over time, heat maps display signal coverage intensity, and advanced filters let you search and sort detected targets instantly.

06 Live Dashboard
Results are pushed to the browser over WebSocket — bearing lines, radar pulse, confidence bar, and target pin all update in real-time with historical data tracking.

Server Features

Central Flask server — port 5002 — Windows & Linux compatible

WebSocket Broadcasting
Flask-Sock pushes node status, readings, and triangulation results to all connected dashboards instantly — no polling needed.

API Key Authentication
Every node owns a unique API Key. The server validates each request and rejects unauthorized access with HTTP 401.

Rate Limiting
Maximum 60 requests per minute per IP address. Excess requests receive HTTP 429, protecting server stability.

Auto Purge
A background thread checks nodes every 60 seconds and marks any node that hasn't sent a heartbeat in 5 minutes as Offline.

SQLite Persistence
All nodes, readings, and events are stored permanently. Full CSV export available directly from the dashboard.

Health Endpoint `/api/health`
Returns server uptime, active node count, total nodes, and database file path for monitoring integrations.

Triangulation Engine
Computes the intersection of bearing vectors from all active nodes using least-squares, providing a confidence percentage.

Event Logging
Every register, disconnect, purge, or error event is recorded with timestamp and source IP for full auditability.

Interactive Map
Leaflet.js map with real-time bearing lines, radar pulse animation, confidence bar, and draggable target pins.

Provision Tokens
Admin-only one-time provisioning tokens allow controlled node registration without exposing the master secret.

Signal History Chart
Live scrolling chart of signal strength in dBm per node, helping operators observe signal trends over time.

CSV Export
Export all SDR readings as a CSV file in one click from the dashboard control panel for offline analysis.

Waterfall Spectrum Display
Real-time spectrum waterfall visualization showing frequency activity over time with color-coded signal strength from weak (blue) to strong (red).

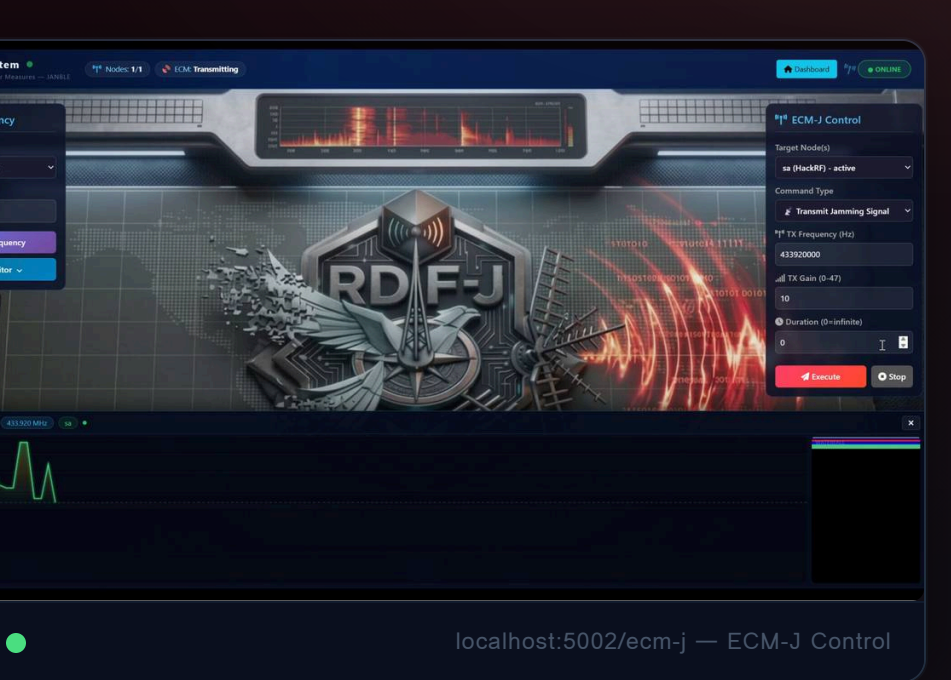
Signal Heat Map
Interactive heat map overlay using Leaflet.heat displaying signal coverage intensity across geographic locations with real-time updates.

Historical Database
Comprehensive historical data viewer with time-range filtering, Chart.js visualization, sortable table, and dedicated CSV export functionality.

Advanced Filters & Search
Powerful filtering system for detected targets: live search by frequency/type, signal strength filter, frequency type filter, and multi-sort options.

ECM-J — Electronic Counter Measures

RF jamming via HackRF One — WebSocket-instant commands — HMAC-SHA256 secured



localhost:5002/ecm-j — ECM-J Control

ECM-J MODULE

Tactical RF Jamming for RDF-J Nodes

The ECM-J module extends every registered RDF-J Raspberry Pi node with RF jamming capability via HackRF One. Commands are delivered instantly over WebSocket with SQLite queue fallback — secured end-to-end with HMAC-SHA256 signatures and anti-replay nonce protection.

WebSocket Instant · HackRF TX · HMAC-SHA256 · ALL NODES

WebSocket Instant Commands
Commands are pushed over a persistent WebSocket connection for near-zero delivery latency. SQLite queue provides reliable fallback when a node is momentarily offline.

HackRF One TX Support
Auto-detected via USB VID:PID 1d50:6089. SoapySDR Python bindings drive the TX engine. Configurable frequency (1 MHz–6 GHz), gain (0–47 dB), and duration (0 = infinite).

Monitor Frequency Control
Set the SDR receive frequency on any single node independently from jamming operations. Targets one node at a time via its own dropdown selector in the ECM-J interface.

Dual Hardware Detection
Nodes auto-detect both RTL-SDR (0bda:2838) for receive and HackRF One (1d50:6089) for transmit by scanning USB VID:PID on startup — no manual configuration needed.

Infinite TX Mode
Setting duration to 0 activates continuous jamming mode. The transmission runs on a daemon thread until a stop_tx command is received, allowing indefinite RF output.

ALL NODES Targeting
The "🎯 ALL NODES" option broadcasts the jamming command simultaneously to every registered online node using parallel WebSocket delivery for instant coordinated operation.

COMMAND TYPES

`set_frequency`
Set the RX monitoring frequency on a single node. Sent from the Monitor Frequency panel. The node's RTL-SDR tunes to the new frequency immediately.

`tx_tone`
Transmit a continuous wave jamming signal via HackRF One. Parameters: frequency (Hz), gain (0–47 dB), duration (seconds — 0 = infinite mode).

`stop_tx`
Stop active RF transmission on target node(s). Sets the _tx_active flag to False — the HackRF TX daemon thread terminates gracefully and the hardware goes idle.

COMMAND DELIVERY FLOW

ECM-J UI → POST /api/nodes/ecm-j → WebSocket Push → SQLite Queue (fallback) → Node Executes

Raspberry Pi Node

RTL-SDR client running as a Systemd service

Auto Re-Registration
If the server rejects the API Key (HTTP 403), the node automatically triggers a new registration flow without operator intervention.

Hardware Monitoring
Each heartbeat carries CPU%, RAM%, and CPU temperature — all visible as live pills in the node's web interface navbar.

Log Rotation
Rotating log files capped at 5 MB each, keeping 3 backups — preventing storage exhaustion on low-capacity SD cards.

Connection Recovery
Detects server disconnection and retries with exponential back-off, keeping the scan thread alive during network outages.

Flexible Configuration
`config.json` controls frequency, gain, antenna height, server address, sample rate, and send interval — editable via the web UI.

Local Web Interface
Built-in Flask web UI on port 8080: view node info, change settings, and start/stop/restart the SDR service remotely.

Antenna Height
Configurable antenna height (AGL) in meters — included in registrations and displayed per-node on the server dashboard.

Systemd Service
Shipped with a `.service` unit file — starts on boot, restarts on failure, fully managed by Systemd.

Security Architecture

Multi-layer protection from registration to data transmission

Registration Secret
A server-side secret (env var or auto-generated file) is required for any node to register. Nodes without it receive HTTP 401 and are blocked.

Per-Node API Keys
After registration each node receives a unique cryptographic API Key. All subsequent requests require this key in the Authorization header.

Rate Limiting
Sliding window rate limiter: maximum 60 requests/minute/IP. Prevents DDoS, brute-force, and data flooding attacks.

One-Time Tokens
Admin-issued provisioning tokens are single-use and time-scoped, preventing re-use or token harvesting attacks.

Full Audit Log
Every registration, disconnection, auto-purge, and error is logged with a Unix timestamp and source IP address in SQLite.

Key Revocation
Admin API endpoints allow instant revocation of any node's API Key, removing its access without restarting the server.

HMAC-SHA256 Signing
All ECM-J commands are signed with HMAC-SHA256 using a shared secret. The server verifies the `X-RDFJ-Sig` header before executing any RF operation.

Replay Protection
Each request carries a unique nonce and Unix timestamp. The server rejects any request outside a 5-minute window or with a previously seen nonce, blocking replay attacks.

IP Auto-Blacklist
After 5 consecutive authentication failures from the same IP, that address is automatically blocked for 10 minutes. All ban events are recorded in the audit log.

System Architecture

Two-tier distributed architecture

Central Server
- Python 3 + Flask + Flask-Sock
- SQLite — permanent storage
- Default port: **5002**
- WebSocket: live broadcast
- Rate limiting + API keys
- Auto purge thread: every 60s
- Windows & Linux compatible
- Leaflet.js interactive map

↔ HTTP / WS · API Key Auth · 1–N nodes

Raspberry Pi Node
- RTL-SDR hardware receiver
- Python 3 + Requests
- Web UI on port **8080**
- Log rotation: 5 MB × 3
- config.json — fully editable
- Systemd service — auto start
- CPU / RAM / Temp monitoring
- Auto reconnect on failure
- HackRF One — TX + RX (1 MHz–6 GHz)
- WebSocket client — instant commands
- RF Jamming — SoapySDR TX engine

API Reference

All endpoints are served on the central server at port 5002

Endpoint	Method	Description / الوصف	Auth
`/api/nodes/register`	POST	Register a new node with Registration Secret	Secret
`/api/sdr/data`	POST	Submit an SDR reading (RSSI, bearing...)	API Key
`/api/nodes/heartbeat`	POST	Send periodic heartbeat with hardware stats	API Key
`/api/status`	GET	Full system status — all nodes & stats	Public
`/api/nodes`	GET	List all registered nodes with latest data	Public
`/api/health`	GET	Server health — uptime, active nodes, DB	Public
`/api/events`	GET	Event audit log — register, disconnect, purge	Public
`/api/sdr/history`	GET	Historical SDR readings with time-range filtering	Public
`/api/export/csv`	GET	Download all SDR readings as CSV	Public
`/api/admin/nodes/provision`	POST	Issue a one-time provisioning token (admin)	Secret

Supported Hardware

All hardware devices supported by the RDF-J/ECM-J system — tested and auto-detected

RTL-SDR — RX ONLY
Low-cost software-defined radio receiver used as the primary signal scanning device on each Raspberry Pi node.
- USB VID:PID `0bda:2838`
- Frequency range: **500 kHz – 1.75 GHz**
- Function: **Signal RX — RSSI + Bearing**
- Interface: USB 2.0
- Auto-detected on node startup
- Works with rtl_power / pyrtlsdr
- Used by: **All RDF-J nodes**

Receive · Triangulation

HackRF One — TX + RX
Full-duplex capable SDR used for RF jamming (TX) in ECM-J mode. Driven via SoapySDR Python bindings on each node.
- USB VID:PID `1d50:6089`
- Frequency range: **1 MHz – 6 GHz**
- Function: **RF Jamming TX — CW Tone**
- TX Gain: **0 – 47 dB**
- Duration: **Timed or Infinite (0)**
- Interface: USB 2.0 — max 20 Msps
- Auto-detected on node startup
- Driver: **SoapySDR + SoapyHackRF**
- Used by: **ECM-J module**

Transmit · ECM-J Jamming · Receive

Raspberry Pi — NODE
The field node computer. Runs the RDF-J node agent as a Systemd service, manages attached SDR hardware, and communicates with the central server.
- Raspberry Pi 3 / 4 / 5
- OS: Raspberry Pi OS (Debian)
- Python 3 + Flask + SoapySDR
- USB: RTL-SDR + HackRF connected
- Network: WiFi / Ethernet to server
- Service: auto-start on boot (systemd)
- CPU / RAM / Temp monitoring
- Local Web UI on port 8080

Field Node · Auto-Start · Networked

HARDWARE COMPATIBILITY MATRIX

Device	Role	Frequency Range	Module	Auto-Detect
RTL-SDR	RX Only	500 kHz – 1.75 GHz	RDF-J	0bda:2838
HackRF One	TX & RX	1 MHz – 6 GHz	ECM-J	1d50:6089
Raspberry Pi 3/4/5	Node Host	—	RDF-J + ECM-J	Manual install