

ADS-B Integrity Monitor: End to End

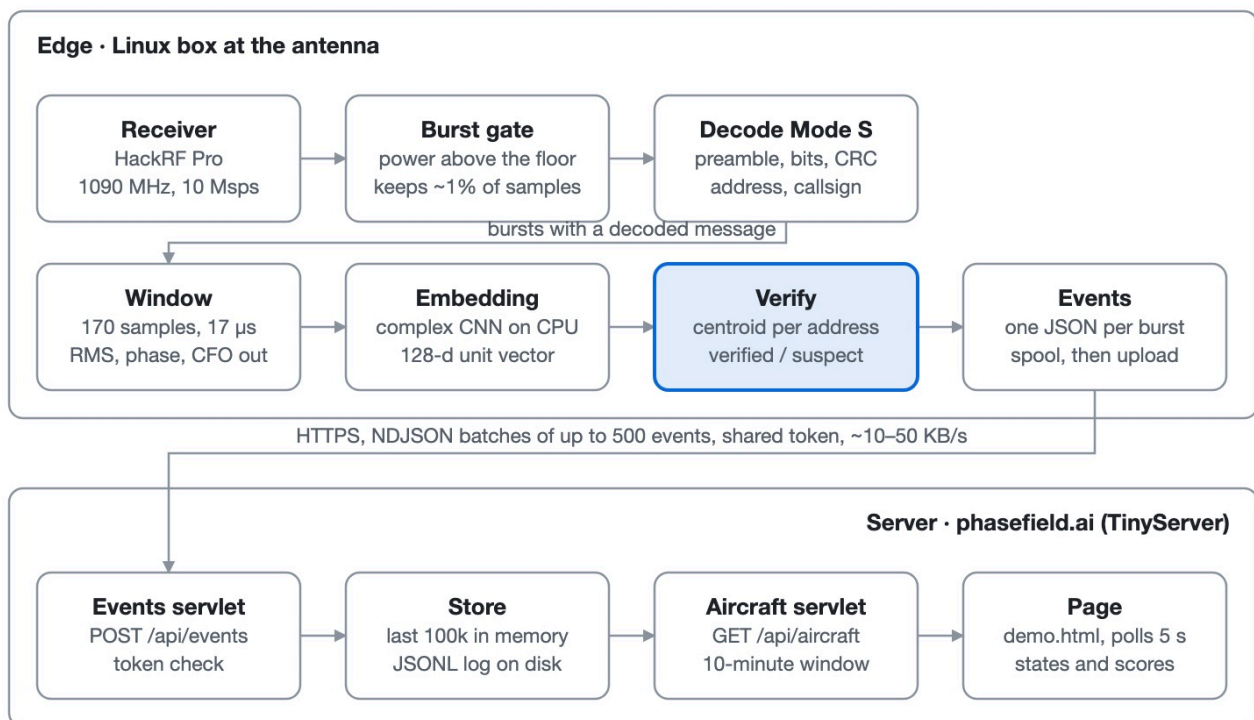
27 October 2026 · Marcel Crasmaru, Phasefield · live page: <https://phasefield.ai/demo.html>

Overview

Every Mode S burst that claims an aircraft address is checked, within seconds of arriving at the antenna, against that address's own transmitter signature, and the verdict is shown on a web page. The signature is enrolled from the aircraft's first bursts in the pass; nothing is trained at run time, so aircraft never seen before are handled. A burst that claims an enrolled address but does not match it is flagged as suspect.

The pipeline has two halves. The edge (a Linux box at the antenna, today linux1 with a HackRF Pro) turns radio samples into a stream of small JSON events; the server (phasefield.ai, a TinyServer Java process) stores the events and serves the live table at phasefield.ai/demo.html. Only events cross the network, about 50 per second at busy hours and under 1 KB each; the radio samples (1 GB per hour gated) stay on the edge.

Only events cross the network; the radio samples stay at the antenna



The top row turns samples into decoded bursts, the middle row turns each decoded burst into a verdict and an event, and the server only ever sees events. The highlighted step, Verify, is where the claim is checked.

Stage 1 · Radio and gating

The receiver listens at 1090 MHz, where aircraft transponders answer radar and broadcast ADS-B, and keeps only the bursts. A HackRF Pro (8-bit samples) runs at 10 Msps; an RSPduo (14-bit) is

supported through SoapySDR at the same rate. The sample stream is 20 MB/s, far too much to store or send, and almost all of it is noise between bursts.

`gated_record.py` detects bursts on 10 μ s windows of power against a running noise floor (threshold 10 dB above it, measured on 5-sample averages at 10 Msps), merges nearby windows, pads 1,000 samples on each side and writes each burst with its start sample to an hourly file. It keeps about 1–2% of the samples: at tonight's rate on linux1, 48 bursts per second, ~1 GB per hour. A

`floor_*.csv` beside the bursts records the noise floor and burst count every second, and gives the absolute time of sample zero.

Two traps found on the way: the HackRF Pro can come up tuned exactly 2 MHz off at 10 Msps (`probe_check.py` runs before every long recording and catches it), and the RSPduo drops off USB on macOS 26 every few minutes while it runs for hours on Windows and Linux.

Stage 2 · Decoding Mode S

Each burst is decoded at 2 Msps (five 10 Msps samples averaged, on each of the five decimation phases so no preamble is missed). A Mode S message starts with a fixed preamble: four 0.5 μ s pulses at 0, 1, 3.5 and 4.5 μ s, then 56 or 112 bits at 1 μ s each, pulse-position coded.

`modes_decode.py` finds the preamble pattern in the magnitude, reads the bits, and checks the 24-bit CRC (polynomial 0xFFFF409).

What comes out per message: the downlink format (DF), the 24-bit ICAO address, and for ADS-B (DF17/18) the callsign and altitude when that message type carries them. The address is in clear in DF11/17/18; in the other formats it is folded into the CRC, so it is recovered only for aircraft already heard in clear ("address/parity" messages). A burst with a valid CRC but an unknown address is kept as address-less; roughly 14% of bursts decode as Mode S, the rest are Mode A/C replies, collisions and noise.

Tonight's live rate on linux1: about 48 bursts and 15–20 decoded messages per second, 10–20 aircraft in range at a time.

Stage 3 · The fingerprint window

The signature lives in the first 17 μ s of a message, before any identifying bits: 8 samples (at 2 Msps) before the preamble, the 16-sample preamble and the first 5 bits, taken at the full 10 Msps rate as 170 complex samples. The preamble start is refined to the sample by matching the ideal pulse pattern within $\pm$ one 2 Msps sample.

Three normalisations make windows comparable (`fpwin.Windower`, the same code as the training set builder):

1. Unit RMS, so transmit power and distance drop out.
2. The first preamble pulse rotated to phase zero, so the window carries relative phase, not the random carrier phase at arrival.
3. The carrier offset — the phase slope across the four preamble pulses, in Hz — is measured and the window is de-rotated by it before the network.

Step 3 matters. The offset (transmitter frequency error, typically ± 200 kHz) is a strong but fragile cue: it is stable within a pass, and airliners that switch to their second transponder jump by 100–200 kHz.

Training on windows that still carried the offset let the network lean on it; removing it costs 4 points within a pass and doubles the cross-day true-accept rate. The offset is still reported in every event (`cfo_khz`) as a separate feature.

Stage 4 • The embedding

The window goes through a small complex-valued convolutional network and comes out as a 128-dimensional unit vector; two bursts from the same transmitter land close (cosine near 1), two from different transmitters far apart. The network: three complex convolutions (45, 45, 90 channels; kernels 5, 5, 3) with cardioid activations, which keep phase, then a complex linear layer to 64 complex numbers read out as 128 reals and normalised. About 2 million parameters, 1 ms per burst on one CPU core, 8 MB of weights.

It was trained with an ArcFace loss (angular margin 0.1, scale 16) to tell apart the aircraft of two recordings (167 training aircraft), on linux1 in 40 minutes. The model is receiver-specific: one for the HackRF at 10 Msps, one for the RSPduo (`models/fp_embed/`). Nothing about the aircraft on the live page is in the weights; the network learned what distinguishes transmitters in general.

Measured, within a pass, enrolling on the first 80% of each aircraft's bursts and testing on the last 20%:

set	aircraft	single burst TAR@1%FAR	20-burst TAR@1%FAR	EER (20 bursts)	never-seen aircraft accepted under a wrong address
RSPduo 14-bit, 27 Oct	175	47%	72%	3.8%	0.8%
HackRF 8-bit, 27 Oct morning	116	18%	41%	4.8%	1.1%
HackRF, two days combined	167	36%	69%	4.7%	1.1%

TAR@1%FAR is the share of genuine bursts accepted when the threshold is set so that 1% of impostor bursts pass; the raw window without the network manages 10–29%. The signature does not drift inside a pass: on 40 never-trained aircraft the cosine to their own enrolled centroid stays at 0.76–0.84 from the first minute to 30 minutes.

Stage 5 • Enrol and verify

The edge keeps one centroid per address heard in the last 10 minutes: the mean of the embeddings of that address's accepted bursts. A burst that carries an address is scored by the cosine between its embedding and that address's centroid (`score`), and by the cosine of the mean of its last 10 bursts (`score10`); the strongest similarity to any *other* active centroid is recorded too (`other`), for calibration and for the operator.

status	when	effect
enrolling	the address has fewer than 10 accepted bursts	burst added to the centroid
verified	score $\geq$ threshold (0.55 HackRF, 0.45 RSPduo) and no other active centroid beats it by 0.05	burst added to the centroid

status	when	effect
suspect	score below threshold, or another centroid closer by 0.05; address already enrolled	not enrolled; the nearest other aircraft is named
attributed	no address, but the nearest active centroid is $\geq$ threshold	the burst is assigned to that aircraft
unattributed	no address, nothing close	kept for the counts only

The thresholds come from a replay of a day-3 hour: genuine bursts have a per-burst cosine median of 0.83 and a 5th percentile of 0.60, so 0.55 flags only bursts that are clearly off; the 10-burst mean is above 0.92 for 88% of genuine bursts while the strongest other centroid's 99th percentile is 0.92. A single stray burst therefore rarely changes an aircraft's state; a run of them does. Not enrolling suspect bursts keeps a spoofer from pulling the victim's centroid toward itself.

Stage 6 · Events and upload

Every scored burst becomes one flat JSON event of a few hundred bytes, for example:

```
{ "t": 1793129342.876, "site": "linux1-hackrf", "icao": "4080E1", "callsign":
"BAW679", "alt_ft": 4550, "df": 17, "status": "verified", "score": 0.919, "score10":
0.981, "nearest": "407534", "other": 0.41, "peak": 15.3, "cfo_khz": -3.4, "mem_mb":
255.2 }
```

`t` is the burst's absolute time from the recorder's clock, `peak` the burst amplitude in sample units, `mem_mb` the edge process's resident memory (read from `/proc/self/statm` once a second), which the page shows as its rightmost tile. Events are appended to a local spool file first (`events.jsonl`), then an uploader thread posts them in batches of up to 500 as newline-delimited JSON to `https://phasefield.ai/api/events` with two headers, `X-Token` (a shared secret, kept in `demo_token.txt` on both machines and never in the repository) and `X-Site`. A failed upload keeps the batch and retries after five seconds, holding up to 20,000 events, so a server outage loses nothing. Events are coalesced into one POST a second; at busy hours this is 10–50 KB/s outbound; nothing else leaves the edge.

The reply to every POST carries the site's target configuration, `{"ok":true,"n":...,"config":{"verify_threshold":...,"margin":...,"simulate_spoof":...,"lna_gain":...,"vga_gain":...}}`, so the same channel that carries events also carries control: a threshold or margin takes effect at the edge within about a second, `simulate_spoof` (one-shot) arms a spoof episode, and requested receiver gains are written to a small file the recorder reads at its next restart. When no aircraft is heard the edge posts a heartbeat line every minute (bursts, messages, current thresholds, memory; not stored) so the channel stays open and the page can show the site as alive. The edge needs no inbound port and polls nothing. Operators change a site's configuration with `POST /api/control` and the token; `GET /api/control` shows every site's target.

Stage 7 · Server and page

The server side is three TinyServer servlets in the `phasefield.ai` site, mapped in its `config.txt`:

endpoint	servlet	what it does
POST /api/events	DemoEventsServlet	checks X-Token , parses each NDJSON line (FlatJson), stores it, answers with the site's target config
GET /api/aircraft?window=600	DemoAircraftServlet	aircraft heard in the window, per-address counts and scores, latest suspects, per-site memory; cached 2 s
POST /api/control	DemoControlServlet	{"cmd": "spoof"} from the page (no token, one a minute); with the token, a site's thresholds, margin and gains
GET /demo.html	static file	the page, polling /api/aircraft every 5 s

DemoStore keeps the last 100,000 events in memory (the raw line is dropped once written: 152,000 events with their lines had filled the 128 MB heap and stalled uploads for an hour on 27 October; the heap is now 256 MB) and appends every event to `logs/demo_events.jsonl`, which it replays at start-up (21,567 events came back after tonight's restart). There is no database yet; the Python reference server in `rf_stream/demo/server.py` uses SQLite and the same API, and moving the Java store to SQLite is a contained change when history queries are wanted.

The page shows, per aircraft in the last 10 minutes: address, callsign, altitude, the state, counts of verified / suspect / enrolling bursts, the last score and the mean 10-burst score, and last seen; above it the totals and the edge's memory; below it the latest suspect bursts with the nearest other aircraft. The verdict comes from the recent window and is latched: **alert** (red, held two minutes) when, on three consecutive scored bursts, the mean of the last ten bursts matches another aircraft's signature better than the claimed one by 0.15 with at least three of the ten flagged (a signature that merely collapses, as on a weak signal, makes the aircraft degraded, not alerted; and the ten bursts' median peak must clear 6, so an aircraft fading out among same-type neighbours, which drifts toward their signatures without any second transmitter, also stays amber: replaying the production log, this removed all nine false alerts and kept the simulated spoofs above that strength); **degraded** (amber, one minute) when four of the last ten are flagged or the 10-burst score is below 0.75; **verified** otherwise; **enrolling** before ten scored bursts. On genuine bursts the aggregate margin exceeds 0.15 in 0.1% of events and never on three in a row in the replays; on a spoof run it reaches 0.9.

The page's "Simulate a spoofed address" button posts a command that reaches the edge in its next events reply; the edge then relabels the next ten real bursts of the aircraft heard most recently with the address of another aircraft in range (drawn at random, near-identical signatures excluded) before verification, marks the events as simulated, never enrolls them, and the page shows the detector's verdict: in the replay tests every one of the ten was flagged, the 10-burst score fell from 0.95 to 0.03, and the alert rose on the third burst; against a near-identical pair (signature cosine 0.94) only two of ten were flagged, which the banner states.

What it supports, and what it does not

The demo claims one thing: within a pass, a burst that claims an address can be checked against that address's own signature, on aircraft never seen before, with about 1% of impostor bursts accepted and most genuine ones passing once ten bursts are in. That is what the measurements support (72% of single 14-bit bursts at a 1% false-accept rate, 89% with 20 bursts on the earlier 10 Msps set).

It does not claim that a signature persists across days. On nine airframes recorded 36 hours apart, verification against the earlier day's centroid fails in aggregate (true accepts 7–24% depending on the setting); four of the nine do persist, two apparently changed transponder, and the rest are in between. So the page re-enrols every pass and never carries identities over.

Other limits worth knowing before showing it:

- Labels come from CRC-valid messages, which favour strong signals; a weak aircraft enrolls slowly.
- An attacker who records a transmitter and replays its waveform would pass; the check is against the signature, not against liveness or position. A direction-of-arrival check with the RSPduo's two tuners is the planned second factor.
- One receiver, one place, two weeks of data. The thresholds are calibrated on one hour of one day.
- Suspect bursts do occur on genuine aircraft (a few per cent of bursts); the aircraft-level rule absorbs them, and a short run on the page is not yet evidence of anything.

Running it

Everything lives in the `ml-go` repository: edge code in `rf_stream/demo/` (`edge.py`, `fpwin.py`, `README.md`), the recorder and decoder in `rf_stream/`, models in `rf_stream/models/fp_embed/`, the servlets in `PhaseFieldAi/src/ai/phasefield/web/Demo*.java` and the page in `PhaseFieldAi/html/demo.html`.

Edge, live (linux1 today; any Linux box with Python 3, numpy, torch on CPU, and the `hackrf` package):

```
cd ~/rf/demo
python3 ../probe_check.py --rate 10e6          # must print OK (tuning
and decode check)
./run_live.sh                                # recorder (10 Msps,
hourly files) + edge --follow
./restart_edge.sh                             # restart the edge only,
e.g. after a code change
tail -f ~/rf_data/live/record.log ~/rf/demo/edge_live.log
```

A site whose radio already feeds another decoder (a Flightradar24 or ADS-B Exchange host, an airport's own system) does not hand over the USB device. The recorder then takes the site's own I/Q stream, teed over localhost or the LAN, and opens no device: `--source tcp://:7355` (listen) or `tcp://host:port` (connect), `udp://:port`, `zmq://host:port` (ZeroMQ PUB), `rtl_tcp://host:port` (the `rtl_tcp` protocol, as served by `rtl_tcp` and `SDRplay's rsp_tcp`) or `file://path`; `--net_format` names the sample format (`cs8`, `cu8`, `cs16`, `cf32`). The smallest tee is `hackrf_transfer -r - ... | tee >(python3 iq_send.py - tcp://edge:7355 --rate 0) | legacy_decoder`. The stream must be at the model's rate and centred on 1090 MHz; everything after the recorder is unchanged. On a 60 s raw capture every network path and format keeps exactly the file source's 6,109 bursts; 16-bit input lets the decoder accept slightly weaker preambles (813 messages against 656), a property of the decoder's 8-bit noise margin rather than of the stream.

Replay of a recording as if live, for a demo without a radio:

```
python3 edge.py BURSTS.bin --model ../models/fp_embed/hackrf_10M_derot_s1.pt --rate
10e6 \
    --realtime --now --server https://phasefield.ai/api/events --token "$(cat
demo_token.txt)" --site replay
```

Server: built and deployed with the site (`PhaseFieldAi/deploy/create_deploy.sh` , `deploy_prod.sh`); the production `config.txt` on the EC2 box carries the two `map` lines and `demo_token.txt` the shared secret; the event log is `~/phasefield/logs/demo_events.jsonl` . The edge's token copy is `~/rf/demo/demo_token.txt` on linux1.

Glossary

term	meaning
address (ICAO 24-bit)	permanent code of the airframe, 6 hex digits (<code>4080E1</code>); first digits give the country (40xxxx UK, A0–AF US). The identity the demo verifies.
callsign	the flight's name typed by the crew (<code>BAW679</code>), sent only in ADS-B identification messages, changes every leg. Context only.
DF	downlink format, the Mode S message type: DF11 all-call reply, DF17/18 ADS-B (address in clear, position, callsign), DF4/5/20/21 surveillance replies with the address folded into the CRC.
burst	a stretch of samples the gate kept: a transmission, a collision, or occasionally noise.
window	the 170 samples (17 μ s at 10 Msps) around a message's preamble that the fingerprint is taken from.
carrier offset (CFO)	the transmitter's frequency error in kHz, measured from the preamble's phase slope; removed before the network, reported in events.
embedding	the 128-number unit vector the network produces for a window; cosine similarity between two embeddings is the score.
centroid	the mean embedding of an address's accepted bursts in the current pass; what a new burst is compared to.
TAR @ 1% FAR	true-accept rate at the threshold where 1% of impostor bursts are accepted.
EER	equal-error rate: the operating point where false accepts equal false rejects.
k-burst score (<code>score10</code>)	cosine of the mean of the last k embeddings to the centroid; steadier than a single burst.