Project Flys Down: a live ADS-B and AIS dashboard with geofence projection

A white paper: how the system works, why it is built this way, how it was built, and where every fact in it came from

Jaron M. Wilson[1] and **Claude**[2]

[1] Jaron Dynamics LLC and Liberty University, Lynchburg, Virginia. jaron@jaronwilson.dev

[2] Anthropic. Claude Fable 5.1, working under the direction of the first author; see Author contributions.

Keywords: ADS-B, AIS, geofencing, dead reckoning, closest point of approach, serverless edge computing, live cartography.

Version 1.4, 22 September 2026. Live at `flysdown.jaronwilson.dev`. Source: `Jaron-Wilson/flysdown`.

Contents

Abstract

Project Flys Down plots live aircraft and live ships on one map and runs a detection engine over them that answers one question: if this target holds its current track and speed, does it end up somewhere it should not be, and how long have we got. Aircraft positions come from community ADS-B aggregators, ship positions from a national AIS service, and restricted airspace from the FAA's own published dataset. It runs as static files plus two edge functions on Cloudflare's free tier, plus one background poller on an ordinary connection.

This paper documents the data sources and their quirks, the architecture, the detection mathematics, the airspace pipeline, the visual encoding and the verification. One problem shaped the design more than any other: the community ADS-B services rate-limit by IP address, and serverless edge platforms egress from addresses shared with every other customer, so the obvious architecture does not work. Section 4 measures that and describes the fix.

A second finding is about trusting data rather than fetching it. The origin and destination shown for a flight come from a volunteer database keyed on the callsign, and among aircraft arriving at one busy airport seven routes in ten described a different leg entirely. Section 2.4 measures that and derives four checks, from the aircraft's own position, altitude, vertical rate and track, that reject a route the aircraft is demonstrably not flying: the system reports what it can defend and labels the rest as reported rather than known.

Claims are tagged by provenance. **[measured]** means obtained by instrumenting this system between 16 and 22 September 2026. **[documented]** means from a provider's own documentation, cited in Section 16. **[standard]** means from a published standard or regulation. Section 10 collects the load-bearing claims in one table.

1. Scope

The system shows live aircraft (ADS-B) and ships (AIS) on one dark map, aircraft colored by altitude and ships by whether they are under way, and runs a rules engine on every update: already inside a restricted zone, projected to enter one with a time to the boundary, emergency transponder codes, rapid descent, and orbiting or holding. It carries the FAA's published prohibited areas with their real boundaries and limits, plus the statutory Washington DC Special Flight Rules Area and the two standing Disney restrictions. An operator can draw further watch zones (circle or polygon), set their floor, ceiling and whether they apply to aircraft, ships or both, and export or import them as GeoJSON. By default it tracks only the area on screen, or an operator can pin circles and boxes that keep loading while the map is scrolled anywhere else. Selecting an aircraft draws the track it has been observed flying plus its published origin and destination as great circles. Each feed pauses independently, every contact is timestamped, and the interface reports which path served its data and how old that data is.

It is not a navigation tool and says so on every screen. The projection is straight-line dead reckoning: no turns, no wind, no flight plan, no controller instruction. NOTAM activation is not modeled, so a restricted area that is cold today is still drawn.

The system is 8,033 lines across browser modules, edge functions, the shared fetch layer, the relay, tooling and tests **[measured]**, with no build step and no framework. MapLibre GL JS is vendored as one 954 KB file so the page does not depend on a third-party script host at runtime.

2. The source data

2.1 ADS-B

ADS-B is cooperative surveillance: an aircraft derives its own position, usually from GNSS, and broadcasts it unprompted with identity, altitude, velocity and status. The 1090 MHz Extended Squitter link and its message formats are specified in RTCA DO-260B and ICAO Annex 10 Volume IV **[standard]**. Anyone with an antenna and a software defined radio can decode it, which is why a volunteer receiver network exists and aggregate live data is free.

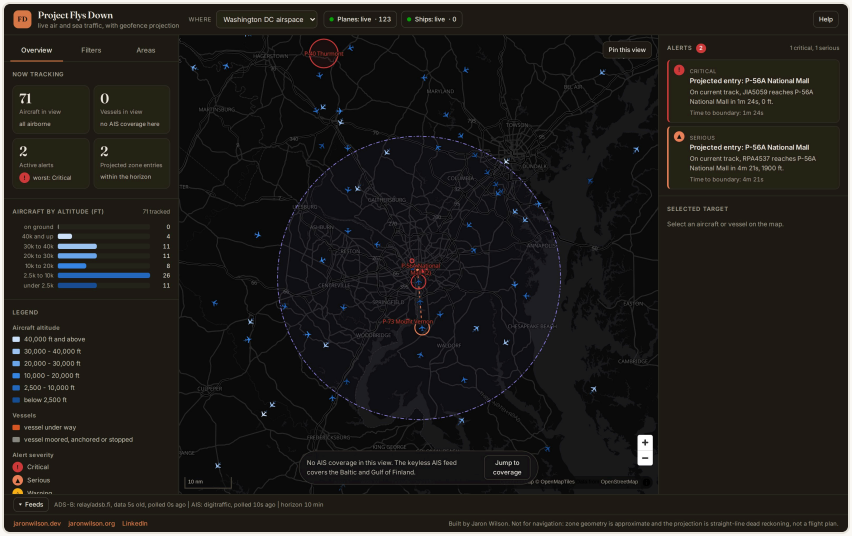

Figure 1. The dashboard over Washington. The left rail is tabbed (Overview, Filters, Areas), the map carries the FAA prohibited areas and the statutory DC Special Flight Rules Area, and the right rail holds alerts and the selected target. 104 aircraft were in the covered area when this was captured.

What arrives here is the JSON that the `readsb` decoder and its `tar1090` interface produce, which aggregators serve directly; field semantics are in the readsb JSON reference **[documented]**. Three quirks are load-bearing:

1. **`alt_baro` is not always a number.** For a target on the ground it is the string `"ground"`. Unchecked arithmetic produces `NaN` altitudes that poison every downstream comparison. The normalizer converts it to an explicit `onGround: true` at altitude zero.
2. **`track` is often absent:** 27 of 115 aircraft over Washington in one sample **[measured]**, typically surface or otherwise limited targets. The normalizer falls back to `true_heading`, then `mag_heading`, but never to `nav_heading`, which is the heading the autopilot is steering towards rather than the one being flown.
3. **`dbFlags` is a bitmask**, not a value to compare: bit 1 military, 2 interesting, 4 privacy ICAO address, 8 limited data display. These become four booleans.

Emergency transponder codes carry a separate channel of meaning in `squawk`: 7500 unlawful interference, 7600 radio failure, 7700 general emergency, per the FAA Aeronautical Information Manual **[standard]**. The engine treats 7500 and 7700 as critical and 7600 as serious, and also reads the ADS-B `emergency` field.

2.2 Aggregators, and their terms

Three keyless aggregators were evaluated, and their terms changed the code, so they are quoted rather than paraphrased.

adsb.lol publishes API data under the Open Data Commons Open Database License 1.0 and states "The API is available to everyone" **[documented]**. ODbL is an attribution license, so the map credits it.

adsb.fi is stricter. It states that "The public endpoints are rate limited to 1 request per second", that invalid requests (400, 401, 403, 404, 429) count towards restrictions and "may trigger temporary IP blocks", that the data "is for personal, non-commercial use only", and that users "must cite adsb.fi and include a link to our home page" **[documented]**. Three consequences: the relay spaces upstream requests 1.1 seconds apart rather than bursting one per region; adsb.fi is never called from the edge, where it returns a 403 bot challenge every time (Section 4) and those 403s would accumulate strikes against a shared address for a call that has never once succeeded; and the map attribution cites adsb.fi with a link to its home page. adsb.fi also documents `v2/lat/lon/dist` as deprecated in favor of `v3`, which returns the same `ac`-keyed shape as its other v2 endpoints **[documented]**; the system now uses v3, with the response shape re-verified before the switch **[measured]**.

OpenSky Network serves state vectors as positional arrays in SI units **[documented]**. It works from an ordinary connection (0.56 s, with `x-rate-limit-remaining: 399` confirming the documented anonymous allowance) **[measured]** but is unreachable from the edge, so it is relay-only.

2.3 AIS

AIS is the maritime analog: transponders broadcast position, course, speed and identity on VHF, with encodings specified in ITU-R Recommendation M.1371 **[standard]**. Three details matter:

- **Sentinel values.** Course over ground 360, heading 511 and speed over ground 102.3 mean "not available", not a measurement. Unhandled, a fleet appears to steam due north at 102 knots. In one 220-vessel sample, 10 had no usable course and 40 no usable heading **[measured]**.

- **Ship type is a coded integer** with structural ranges: 30 fishing, 35 military operations, 36 sailing, 37 pleasure craft, 50 to 59 special craft (pilot, tug, search and rescue), 60 to 69 passenger, 70 to 79 cargo, 80 to 89 tanker **[standard]**.
- **ETA is packed into 20 bits**: month in 4, day in 5, hour in 5, minute in 6 **[standard]**. The decoder unpacks it and rejects impossible combinations. Validated live: raw 625728 decodes to 17 September 17:00 UTC, a plausible arrival the day it was read **[measured]**.

Live AIS is far harder to get without a commercial contract than ADS-B. The one genuinely keyless live source found is Fintraffic's Digitraffic marine service, publishing locations and vessel metadata as separate endpoints under CC BY 4.0 **[documented]**. Coverage is the Baltic and Gulf of Finland: the full feed carried 1,157 vessels spanning latitude 56.9 to 65.8 and longitude 17.0 to 35.3 **[measured]**. That limit is stated in the interface, with a region switcher and an explicit message when the view is outside coverage.

Digitraffic has one requirement that costs an afternoon if unread: compression is mandatory, and "If compression is not allowed in the request, the service returns error code `406`" **[documented]**. This was diagnosed the hard way. The endpoint returned 406 for every `Accept` value tried, including `application/json`, `application/geo+json` and `*/*`, and 200 as soon as `Accept-Encoding: gzip` was present **[measured]**: the header that looked like the problem was not the problem. Digitraffic also asks callers to send a `Digitraffic-User` header, noting that "Using the header increases the amount of requests you can make" **[documented]**, so the proxy sends it.

Position and identity arrive separately (`/locations` keyed by MMSI, `/vessels` for names, call signs, types, dimensions, destinations). The proxy merges them by MMSI and caches metadata for 30 minutes against 12 seconds for positions, because a ship's name changes less often than its position.

2.4 Flight routes

ADS-B broadcasts identity, position, altitude and velocity. It does not broadcast where a flight came from or where it is going, because the aircraft is not telling you its schedule, only its state. Origin and destination therefore come from a fourth source: adsbdb, which resolves an airline callsign to origin and destination airports with coordinates, built on volunteer flight route data and published under the MIT license with its data contributors credited **[documented]**.

Four things about using it. Most general aviation and military callsigns have no scheduled route at all, and the API answers 404 for them, which is a normal result rather than an error: the interface says "no scheduled route for this callsign" instead of looking broken. A callsign's route does not change during a day, so lookups are cached at the edge for six hours, and misses for one hour, which means one upstream request per callsign per six hours no matter how many people click on it. And unlike the ADS-B aggregators in Section 4, adsbdb answers the edge without complaint, verified from the deployed Worker **[measured]**, so this one needs no relay.

The fourth thing is the one that decides whether any of this can be trusted. A callsign is a flight number, not a leg. The same number flies a different pair of airports on a different day, airlines reuse numbers, and volunteer data goes stale, so a lookup keyed on the callsign returns the route that callsign usually flies, which is not necessarily the route in front of you. Observed on 17 September 2026: callsign BCS30A, a Boeing 737-800, resolved to Leipzig/Halle (EDDP) to Cologne Bonn (EDDK) while the aircraft was over France, 489 NM from Leipzig and 304 NM from Cologne on a leg that is 195 NM long **[measured]**. The panel drew that route and quoted a distance remaining and an arrival time from it, all of which were meaningless.

Every reported route is therefore measured against the aircraft's own position and track before any of it is presented, by two checks in `js/route.js`. The first is a detour test: on a real leg, the distance from the origin to the aircraft plus the distance from the aircraft to the destination stays close to the length of the leg, because the aircraft is somewhere on it. Vectoring, holding and weather deviations add tens of miles; flying a different route adds hundreds, and in the case above the two legs summed to four times the length of the leg itself. A route is rejected when that excess passes the larger of 60 NM and 35 percent of the leg. The second is a bearing test: an aircraft more than 25 NM from its destination and tracking more than 75 degrees away from it is not going there. Below 25 NM the heading is about the approach, not the destination, so the check is skipped rather than made to lie.

How often is it wrong? Measured at a busy airport, which is where the answer matters, on 18 September 2026. Of eleven aircraft on the ground or descending within 25 NM of Washington National, ten had route data, and seven of those ten were parked at an airport that was neither end of their reported leg: RPA3466 at Washington reporting Newark to Portland, JIA5344 reporting Portland to Philadelphia, AAL2077 reporting Boston to Dallas, SWA1246 reporting Houston to New Orleans, and three more **[measured]**. This is not corruption in the data. It is what a callsign means: airlines fly a number over several legs in a day, the database holds one of them, and an aircraft that has just arrived is frequently between two legs, so the leg on file is the one it will fly next or flew earlier.

That measurement came from the first author watching about twenty arrivals and checking them against an independent source by hand, and it exposed a gap in the checks above. Six of the seven were already caught by the detour test. One was not: Washington National lies close to the great circle between Boston and Dallas, so an aircraft parked there produces two legs that sum almost exactly to the length of the route, and the geometry looks perfect. Two further checks close that gap, and both are about what an aircraft is doing rather than where it is.

The first: an aircraft on the ground is at one end of its route, or it is not on that route at all. If it is on the ground more than 10 NM from the nearer of the two reported airports, the route is rejected regardless of how well the distances add up. The second: an aircraft below 5,000 feet and descending is landing within a few miles, so if its reported destination is more than 30 NM away it is landing somewhere else. That is the case the first author was watching, a stream of arrivals into one airport whose panels named airports hundreds of miles off.

Re-measured against the same live traffic with both checks in place, eight of the ten routes were flagged and two were accepted, and both of the accepted ones were genuinely right: FFT690, which really had arrived from Denver, and VIR22Q, sitting at Dulles, which is the origin of the Dulles to Heathrow leg it was reported on **[measured]**. No correct route was rejected in that sample.

A second route source was measured rather than assumed to help. hexdb.io answers callsign lookups from the edge and disagreed with adsbdb on every one of the seven sampled callsigns that it knew: ENY4047 as KDFW-KSJT-KDFW against KPHX-KSLC, AAL1314 as KLAX-KMIA against KPHL-KAUS, JZA786 as CYUL-CYSJ against CYYZ-KDCA **[measured]**. The last of those is decisive: JZA786 was checked indepen-

dently and adsbdb had it right, so cross-referencing the two would have flagged a correct route as disputed. Agreement between two databases of the same kind of crowd-sourced data is not evidence, so the second source was not adopted. What is offered instead is a link, per selected flight, to a source that does know the day's leg, which is the check being made by hand anyway.

Neither check can prove a route right, and they are not presented as doing so: a route with both endpoints that contradicts nothing is reported as consistent, a route with one usable endpoint stays unknown. What they do is stop the system asserting a destination the aircraft is demonstrably not flying to. When either fires, the route block is labeled unverified, the caveat names the three distances that disagree, and the arrival estimate is withheld rather than computed from a route that is not being flown. The route is not drawn on the map at all, and the button that frames a whole flight is withheld with it. Drawing it faintly was tried first and abandoned: a line from Houston through an aircraft over Washington and on to New Orleans reads as a path whatever its opacity, and framing it produced a view of half a continent. The claim stays in the panel, where the numbers that disprove it are next to it.

A second instance, the same day, shows why the check earns its place rather than being a tidy-up. Callsign SWA1246 was over Washington, inside the DC Special Flight Rules Area and descending through 5,375 feet, while adsbdb reported it as Houston (KIAH) to New Orleans (KMSY): 1,048 NM from the origin and 842 NM from the destination on a 264 NM leg, a detour of 1,626 NM **[measured]**. The flight that day was Providence to Washington National. The callsign was right and the leg was long out of date, which is exactly the failure mode a callsign-keyed lookup produces and exactly what a position can refute.

3. Architecture

RUNS IN	PIECE	WHAT IT DOES
Browser (`public/`)	`app.js`	Orchestration: polling, selection, per-target URLs, one render per update
	`js/feeds.js`	Polls the edge, holds the target store and its tiered position history
	`js/detect.js`	Rules engine: zones, projection, squawks, descent, orbit, landing, vessel close approach
	`js/route.js`	Route plausibility checks and airport codes
	`js/geo.js`, `js/zones.js`	Geodesy, and the zone store with its FAA geometry
	`js/map.js`, `js/draw.js`, `js/ui.js`	MapLibre layers, drawing zones and areas, the panels
Edge (`functions/api/`)	`/api/aircraft`	Normalizes, quantizes and caches; reads a relay snapshot first, falls back to stale
	`/api/vessels`	Merges Digitraffic positions with vessel metadata
	`/api/route`	Resolves a callsign to a reported route through adsbdb, cached for hours
	`/api/relay`	Tells the relay which areas are being watched, and stores its snapshots in D1
Relay (`tools/relay.mjs`)	an ordinary IP	Fetches adsb.fi and adsb.lol for the watched areas and pushes them to the edge
Upstreams	ADS-B	adsb.lol from the edge; adsb.fi and OpenSky through the relay only
	AIS, routes, airspace	Digitraffic and adsbdb from the edge; FAA Special Use Airspace at build time

The browser is plain ES modules, no framework, no build step. That was decided explicitly: a live map is overwhelmingly client-side rendering work, and the only thing a server must do is proxy two feeds, normalize them and hide rate limits behind a shared cache. A JVM or Python service would need an always-on host for no functional gain. The detection engine and its geodesy are pure functions with no DOM or map dependency, specifically so they can move into a scheduled Worker or another language later without rewriting the rules.

The edge layer is four Cloudflare Pages Functions **[documented]**: one per feed, the route lookup, and the relay endpoint. They exist because the upstreams send no CORS headers so a browser cannot call them; secrets stay server-side; parameters can be quantized so many viewers collapse onto one cache entry; and one cache entry serves every viewer instead of each viewer generating upstream load.

Normalization is a contract. Both endpoints emit one schema whatever answered, so an aircraft is the same object whether it came from adsb.lol's `ac` array, adsb.fi's v3 `ac` array or OpenSky's positional arrays in SI units (conversions in one place: 3.28084 ft/m, 1.943844 kt per m/s, 196.8504 ft/min per m/s). The same code runs at the edge and in the relay, both importing `shared/adsb.js`, so a relayed snapshot cannot drift in shape from a directly fetched one.

Quantization and caching. Latitude and longitude are rounded to 0.1 degrees (about 6 nautical miles) and radius to 25 nautical mile steps, so two people looking at the same city produce the same region key and share one upstream request. Responses are cached in the Cloudflare Cache API **[documented]** for 6 seconds (aircraft) and 12 (vessels) at the edge with `max-age=0` for the browser, so the browser always asks and the edge answers most asks without touching an upstream. A second, longer entry holds the last known good answer for 5 minutes, which is what makes the degradation in 4.4 possible.

4. The problem that shaped everything: egress addresses

4.1 Symptom and measurement

The first deployment worked perfectly locally and returned nothing in production: 179 aircraft over Washington in local development, HTTP 502 with all three upstreams refusing when deployed **[measured]**. Rather than guess, a temporary diagnostic endpoint was deployed that called each upstream from the edge and reported status, timing, headers and a body snippet. From the Cloudflare IAD colocation **[measured]**:

UPSTREAM	FROM THE EDGE	FROM AN ORDINARY IP
adsb.lol	200 in 463 ms when it works, HTTP 429 (nginx) on most attempts	works every time
adsb.fi	HTTP 403 in 4 ms, body is a Cloudflare bot challenge page	works every time, richest fields
OpenSky	HTTP 522 after 19,533 ms, then 4 timeouts in 4 attempts	200 in 0.56 s

Fresh-fetch success rates from the edge: 1 in 8 with no retry logic, 3 in 8 once a jittered retry on 429 was added, 1 in 6 when re-measured an hour later **[measured]**. The variance is itself informative: the limiter is reacting to the aggregate traffic of everything sharing that address, not to this site.

4.2 Diagnosis

The aggregators rate-limit by IP, the only practical identifier for anonymous users. A serverless function has no address of its own; it egresses from addresses shared with every other customer on the platform in that location, so requests compete against the world for a per-address budget. adsb.lol's documentation notes that its limits are dynamic according to load **[documented]**, which fits the observed variance.

Two dead ends were eliminated before redesigning. **Browser-direct fetching:** none of the aggregators return `Access-Control-Allow-Origin`, so a page cannot call them from the user's own address **[measured]**. **A different source:** six further candidates were probed and all failed: `api.adsb.one` (403 even from an ordinary IP), `api.adsb.im` and `api.theairtraffic.com` (unresolvable), `data.adsb.fi` (401), `globe.adsb.fi` (403) and `api.planespotters.net` (404) **[measured]**. The constraint is the address, not the source, so swapping sources cannot fix it, and an OpenSky account would not help either because OpenSky does not answer the edge at all.

4.3 Graceful degradation as the floor

Whenever the relay is not running, the endpoint serves the last known good answer from the 5 minute cache entry, annotated `stale: true` with its age, and the interface says so in three places: the feed chip, the status bar and a map banner. Verified in production by polling one region repeatedly: four refusals, then 323 aircraft fresh, then the same 323 marked stale at 17 and 29 seconds old **[measured]**. The property that matters is honesty: a dashboard silently showing five minute old positions as current is worse than one showing nothing, because the viewer cannot tell.

4.4 The relay

The fix is to move the fetch to an ordinary address. `tools/relay.mjs` runs anywhere with a normal connection, and each cycle it asks the site which regions people are looking at (`GET /api/relay`, bearer token), fetches those from the aggregators preferring adsb.fi for its richer fields while spacing requests 1.1 seconds apart, and pushes each result back (`POST /api/relay`) into Cloudflare D1. `/api/aircraft` reads a fresh snapshot from D1 *before* trying any upstream, so when the relay runs the edge never contacts an aggregator at all: more reliable, and considerably politer than retrying into a rate limiter.

Demand is recorded by the aircraft endpoint itself: a cache miss writes the canonical region key to a `wanted` table. The relay therefore follows the map rather than polling a fixed list of cities, a newly viewed area is covered within one cycle, regions unviewed for ten minutes are swept, and the list is capped at five.

Covering snapshots. The region key includes the radius, which changes with zoom, so panning produced regions the relay had not yet covered and the first request for them returned nothing. Requests are now answered by any snapshot that fully *contains* the requested area: the endpoint picks the tightest containing snapshot and filters its aircraft to the requested radius. Verified in production: a 25 nautical mile request served from the 75 nautical mile snapshot, filtered from 181 aircraft to 71, and an off-center 50 nautical mile request from the same snapshot giving 117 **[measured]**.

Write budget. D1's free allowance is 5 million rows read and 100,000 written per day **[documented]**. One row per region per cycle, five regions on an 8 second interval, is about 54,000 writes per day. Two throttles keep clear of the ceiling: demand recording reads the existing timestamp and only writes if it is over two minutes old (a region stays listed for ten minutes, so refreshing sooner is pure write volume), and housekeeping deletes run on about one poll in twenty. D1 also caps a Worker invocation at 50 queries on the free plan **[documented]**; this endpoint uses at most three.

When the relay stops, nothing breaks: the endpoint falls back to the aggregators, then an older relay snapshot, then the last known good cache entry, then a 502 with a summarized reason. The banner distinguishes "the aggregators are rate-limiting the edge" from "the relay has stopped pushing", because those have different remedies.

5. The detection engine

5.1 Geodesy

Everything geometric happens on a sphere of radius 3,440.065 nautical miles using standard haversine distance and great-circle destination formulas **[documented]**, with distance in nautical miles, speed in knots, altitude in feet, vertical rate in feet per minute and bearings in degrees true. The unit choice is not aesthetic: it makes time, distance and speed relate without conversion factors, removing a category of error from the projection. A spherical model is accurate enough here; the worst-case error against an ellipsoid over 250 nautical miles is a fraction of a percent, far below the uncertainty in extrapolating from one instantaneous velocity sample.

One consequence caught a bug, in the test rather than the code. A test placed an aircraft 60 nautical miles due west of a zone and tracked it 090, expecting a hit; it missed. Great circles converge, so the bearing back along the great circle differs from 090 by roughly the longitude difference times the sine of the latitude, about 0.8 degrees at latitude 38.9, which over 60 nautical miles is 0.85 nautical miles of cross-track error: against a 0.25 nautical mile zone, a clean miss. The fix was to compute the true initial bearing rather than assume the reciprocal.

5.2 Projection

Take the current position, track and ground speed, step forward in time, and extrapolate altitude linearly from the vertical rate. It models nothing else, because the question a geofence warning needs answered is "if nothing changes, what happens", not "what will this aircraft actually do".

Altitude extrapolation is clamped at ground level. A constant 4,000 feet per minute descent extrapolated ten minutes forward is 40,000 feet lower, which for most aircraft is underground, and an unclamped negative altitude silently falls below every zone floor, so a steeply descending aircraft would stop alerting exactly when it became most interesting. Alongside the clamp is an explicit time-to-ground: if a target reaches the surface at its current rate before the horizon, the projection stops there rather than flying a landed aircraft across the map. Both are tested.

5.3 Finding the first intersection efficiently

For each candidate pair the engine finds the first moment the target is inside the zone both horizontally and vertically within the horizon (default 10 minutes, adjustable 2 to 20). Naively that is hundreds of targets times a dozen zones times hundreds of steps, five times a minute. Three things make it cheap.

A reachability prefilter. Maximum travel is speed times horizon; if the distance to the zone center less its radius exceeds that, the zone is skipped with no stepping at all, which eliminates almost every pair.

A step tied to the zone, not the clock. The step is `max(0.05, min(2, zoneRadius / 2, 0.5))` nautical miles, converted to seconds from the target's speed and floored at one second. A fixed 15 second step sounds reasonable and is a real bug: at 500 knots that is 2 nautical miles per step, which steps clean over P-56B (one nautical mile radius) without ever sampling inside it. There is a test using a 0.25 nautical mile zone and a 550 knot target.

Bisection to refine. Stepping establishes a bracket; 12 iterations between the last clear and first inside sample narrow the crossing to roughly one four-thousandth of a step, which is what lets the interface say "reaches P-40 Thurmont in 7m 52s" rather than rounding to the sample interval.

Containment is analytic for circles (distance to center) and ray casting for polygons (count edge crossings; odd is inside), verified against two known truths: the White House is inside P-56A, Dulles is not.

5.4 Rules and severity

RULE	CONDITION	SEVERITY
Inside a zone	Horizontally inside, between floor and ceiling	Prohibited/TFR critical; restricted/custom serious
Projected entry	First intersection within the horizon	Base rank, reduced one step beyond 2 min, two beyond 6
Emergency squawk	7500 or 7700	Critical
Emergency squawk	7600	Serious
Emergency flag	ADS-B `emergency` field set	Critical
Rapid descent	Below -4,000 ft/min, or -3,000 ft/min under 10,000 ft	Warning; serious when low
Orbit or hold	Over 270 degrees cumulative turn inside a 12 NM footprint over 150 s	Notice
Close approach (vessels)	Projected CPA within the limit (default 1 NM) and TCPA inside 30 min	By CPA band, reduced one step beyond 15 min
Landed	On the ground now, or under 250 ft below 60 kt, having been above 1,000 ft within 15 min	Good, an event rather than a fault

The landing rule is the only one that reports something going right, and it exists because watching an aircraft you have been following actually arrive is part of why anyone runs a dashboard like this. ADS-B has no "landed" message, so the rule reads a transition out of the observed history: on the ground now, and demonstrably airborne a few minutes ago. Both halves are needed. An aircraft parked at a gate has been on the ground all along and has not just landed; an aircraft at 400 ft doing 140 kt is on approach and has not landed yet; a taxiing aircraft reporting 200 ft on a barometric setting has. Its severity is `good`, which keeps it in the alert rail and out of the count of things wrong.

Severity is computed, not stored: zone kind gives a base rank and the projected time reduces it, so an aircraft escalates from warning to serious to critical as it closes without any state being kept. Alert identifiers are stable (`targetId:rule:zoneId`) so the interface tracks an alert across updates instead of flashing a new one every five seconds.

Orbit detection needs history, so the rule judges the last ten minutes of each target's retained positions, whatever Section 7.4 says is retained; it sums absolute heading changes and requires the bounding circle of those positions to stay small, separating a hold or survey orbit from an aircraft simply turning en route. Rapid descent uses two thresholds because 3,500 feet per minute at cruise is routine and the same rate at 2,000 feet is not.

5.5 Advisory zones, or not burying the signal

The first version produced 55 simultaneous alerts over Washington, about 50 of them variations of "an airliner is inside the Washington DC Special Flight Rules Area" **[measured]**. True and useless: the SFRA is a 30 nautical mile ring inside which transiting requires a flight plan, a discrete code and two-way radio, so essentially all traffic in it is authorized. Zones therefore carry an `advisory` flag, default true for special-flight-rules zones: such a zone is still drawn, still tested and still reported in a selected target's zone checks, but never raises an alert. The same view then produced 3 alerts, all genuine: a projected entry into P-40 at Camp David and two into P-56A over the National Mall **[measured]**. A detector that fires on authorized behavior trains its operator to ignore it.

5.6 Vessel close approach

Every rule above looks at one target at a time, which cannot see the risk that matters most at sea: two ships converging on each other. That needs a pairwise pass, and the marine standard for it is the Closest Point of Approach.

Reduce the pair to relative motion. Convert both positions into a local tangent plane in nautical miles, take the relative position vector $\mathbf{r}$ and the relative velocity $\mathbf{v}$ (each vessel's speed over ground resolved onto its course over ground), and the time of closest approach is where the range stops shrinking, $TCPA = -(\mathbf{r} \cdot \mathbf{v}) / |\mathbf{v}|^2$, at which moment the two are $CPA = |\mathbf{r} + \mathbf{v}\,TCPA|$ apart.

A negative or zero TCPA means the pair has already passed its closest point or is opening, in which case the closest approach is simply the present range. A relative speed near zero means two ships holding station on each other, so the range is not going to change either. Both are returned as "not closing" rather than as a spurious prediction.

This is the quantity an ARPA radar computes, and the alarm model is the same too: IMO's performance standards for automatic radar plotting aids require alarms on "a preset minimum acceptable passing distance (CPA) and a preset advance warning time (TCPA)" **[standard]**. Both are exposed here, the CPA limit as a slider defaulting to 1 nautical mile and the horizon at 30 minutes, because the right limit depends entirely on the water: a mile is generous in a traffic separation scheme and tight in the open sea.

Two guards keep the output worth reading. Both vessels must be making at least 1 knot and neither may be reporting anchored or moored, because a harbor is full of ships lying within a cable of each other and none of it is a risk, the same lesson as the advisory zones in 5.5. And pairs are screened through a coarse spatial grid sized to the 20 nautical mile screening range, so the pass is linear in practice rather than quadratic in the contact count. Each pair yields exactly one alert, keyed on both identifiers in a stable order, so the same encounter is never reported twice from opposite ends.

Measured against live Baltic traffic, 275 vessels produced 4 alerts at the 1 nautical mile limit and 27 at 5, the tightest being two ships projected to pass 0.32 NM apart in 2 minutes 14 seconds **[measured]**. Each pair is drawn on the map as a line between the two vessels, colored by severity.

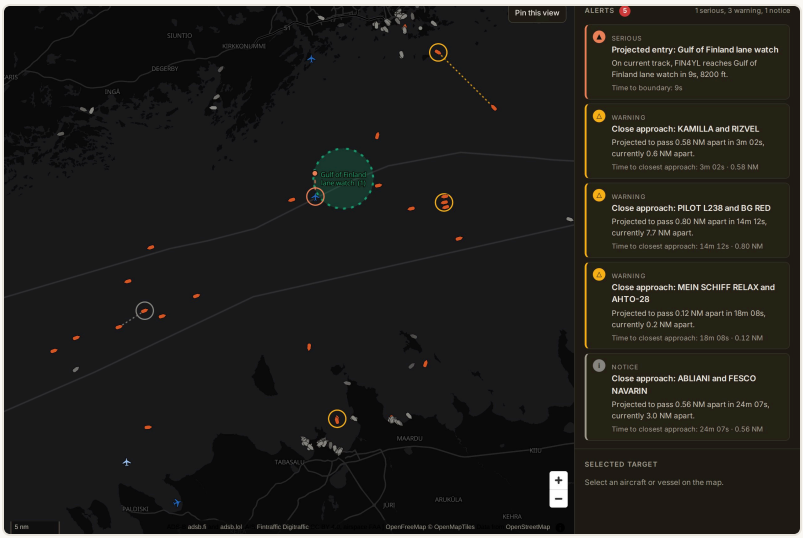

Figure 3. Vessel traffic in the Gulf of Finland with two close-approach warnings in the alert rail. Under way vessels are orange, moored and anchored ones gray; the latter are excluded from the pairwise pass, since a harbor is full of ships lying a cable apart at zero knots. The alert cards carry each pair's projected passing distance and the time to it.

5.7 A worked example

From the live system: `N9287Y`, west of Camp David at 5,000 feet, with track and ground speed from ADS-B. P-40 Thurmont is prohibited, surface to 5,000 feet **[documented]**, geometry from the FAA. The prefilter keeps the pair because the aircraft can reach it inside the horizon; stepping at 0.5 nautical mile granularity finds the first sample inside; bisection refines the crossing to 56 seconds; the projected altitude at that moment is inside the band. Base severity for a prohibited area is critical and 56 seconds is inside the 2 minute imminent band, so no reduction applies. The alert reads "Projected entry: P-40 Thurmont, on current track N9287Y reaches P-40 Thurmont in 56s, 5000 ft", the zone outline thickens, and a dashed projection line is drawn to the predicted entry point **[measured]**.

6. Airspace data

6.1 Using the FAA's own geometry

The first zone file was hand-drawn: P-56A as a rectangle over the National Mall, P-40 as a one mile circle, the DC flight restricted zone as a 15 nautical mile ring. It was labeled approximate and it generated false positives, because a rectangle over the Mall includes the Potomac corridor that Reagan National arrivals actually fly, which the real prohibited area is shaped to exclude.

The FAA publishes Special Use Airspace as a queryable ArcGIS feature service through its Aeronautical Information Services open data portal **[documented]**. Querying for prohibited areas returns 13 features with real boundaries and published limits **[measured]**. The same service holds 555 restricted areas, 718 military operations areas, 212 warning areas, 39 alert areas and 5 of type D **[measured]**; only prohibited areas are ingested, because they are permanent and unconditional, whereas most restricted areas and MOAs are active only by NOTAM or schedule and drawing them as always hot would mislead.

`tools/fetch-zones.mjs` performs the ingest and is committed, so the data is reproducible; the FAA reissues the dataset on the 56 day chart cycle, so the script is re-run rather than its output edited. Three mapping details: `LOWER_VAL` and `UPPER_VAL` are strings with separate code fields where `SFC` means surface, normalized to numeric feet; the country field reads **"UNITED STATES", not "USA"**, and an initial filter comparing against `USA` silently dropped all 13 features, which is why the script now prints every zone it wrote with its vertex count; and P-56 appears twice under one name, disambiguated by geometry, the small circle at 38.9214, -77.0669 being section B over the Naval Observatory and the Mall polygon section A.

6.2 Simplification

The FAA ships circular areas as densely sampled polygons: P-56B, a one nautical mile circle, arrives as 6,285 coordinate pairs, as does P-40 **[measured]**, and 13 zones came to roughly 520 KB of JSON. The ingest applies Douglas-Peucker simplification **[standard]** at a tolerance of 0.0004 degrees (about 44 meters) and rounds to five decimal places. P-56B becomes 17 points, P-40 33, P-56A 12, and the whole file including hand-added zones is 16 zones, 506 polygon vertices and 18.6 KB **[measured]**. The difference is not visible at any zoom the map offers, and the simplified geometry is still tested for containment.

6.3 Zones defined by regulation, and one that was dropped

Two zones are built from their legal definition and are therefore exact rather than approximate. The DC Special Flight Rules Area is defined in 14 CFR part 93 subpart V as the airspace within 30 nautical miles of the Reagan National VOR/DME **[standard]**, so a circle is the true shape. The prohibited areas are established under 14 CFR part 73 **[standard]**, the authority behind the FAA geometry. The two Disney restrictions are permanent TFRs with published centers, 3 nautical mile radius, surface to 3,000 feet AGL.

The DC flight restricted zone was dropped rather than approximated: its real boundary is an irregular coordinate-defined polygon, the dataset does not contain it, and a circular approximation was both wrong and the largest single source of false alerts. Shipping nothing was more honest than shipping a bad shape.

Altitude datum caveat. Floors and ceilings are compared against barometric altitude in feet MSL, but several real restrictions are published in feet AGL (P-40 to 5,000 AGL, Disney to 3,000 AGL). Those zones are flagged `agl: true`, and the comparison error is the terrain elevation, which the system does not model. This is stated in the zone metadata and in the interface.

7. Visual encoding

7.1 Color assigned by role, then validated

Color here encodes data, so it was assigned by the job each color does and then checked with a contrast and color-vision validator against the page's actual dark surface rather than judged by eye.

The chrome is jaronwilson.dev and jaronwilson.org's own palette and type, verbatim, so the three sites read as one person. That moved the data's surface from `#141416` to `#201d16`, so every palette below was re-validated against it. One rule falls out: the brand accent (`#d97a4a`) is seven units of color difference from the vessel orange, inside the range where a viewer cannot tell them apart, so the accent stays in the chrome and never appears on the map, in the legend or in the data panels, and no data color appears in the chrome.

Altitude is a magnitude, so it gets an ordinal ramp on a single hue, monotone in lightness, dark low and light high: `#184f95`, `#256abf`, `#3987e5`, `#6da7ec`, `#9ec5f4`, `#cde2fb` for bands below 2,500 ft, to 10,000, 20,000, 30,000, 40,000 and above, passing all four ordinal checks with the darkest step at 2.08:1 against the surface **[measured]**. A rainbow ramp, which several trackers use, was rejected: hue carries no order, so two colors cannot be ranked without a legend.

Aircraft against vessels is identity, so categorical hues: blue `#3987e5` and orange `#d95926`, a pair that passes all-pairs color-vision separation **[measured]**. Gray `#898781` is reserved for "no data, not moving or stale", never an identity color, and always labeled.

Alerts are status, using a reserved four-step palette (`#d03b3b` critical, `#ec835a` serious, `#fab219` warning, gray notice) never reused for a data series. Red against green is inherently weak under deuteranopia, measured at a difference of 4.1 **[measured]**, which is why every alert pairs its color with a glyph and the severity word: status color never carries meaning alone here.

Zone kind is identity with a safety-critical failure mode, so it is encoded twice. The four hues clear the normal-vision floor comfortably (worst pair 24.6) but sit in the color-vision warning band (7.2) **[measured]**, which is only acceptable with a second channel, so zone kind is also carried by outline dash: prohibited solid, TFR short dash, restricted long dash, special flight rules dash-dot, custom dotted. Because MapLibre's `line-dasharray` cannot be data driven **[documented]**, this is one line layer per kind with a filter: more code, but the redundant channel survives. Every zone is also directly labeled.

Icons are generated at runtime on a canvas, one pre-colored image per altitude band and vessel state, with a dark casing stroke so a light icon stays legible over a light coastline; the alternative, one white icon tinted by data, needs signed distance fields and gives soft edges at small sizes.

7.2 Only what is on screen, and how old it is

The upstreams take a center and radius, so the smallest circle covering a rectangular viewport always includes targets outside it. Those are fetched (they cost nothing) but filtered out of the map, counts, alerts, trails and table, so "105 aircraft in view" is literally true. The coverage ring is drawn only when the 250 nautical mile upstream cap actually cuts into the view. Targets are dropped when they leave the covered area, both on a successful poll and immediately on a view change; without the second rule, switching from Washington to the Gulf of Finland displayed 241 aircraft where the feed reported 1 **[measured]**.

That default can be overridden by pinning up to four **tracking areas**, drawn as circles or boxes. Pinned areas replace the viewport as both the query set and the display filter, so the feeds keep loading them while the map is scrolled somewhere else entirely, and their contacts stay in the counts and the alerts even when off screen. Because the upstreams take a center and a radius, a box is queried by its bounding circle and then filtered back to the rectangle for display. Areas persist in localStorage, and the map says so when none of them is on screen. Each feed also pauses on its own, so the aircraft picture can be frozen for inspection while the ships keep moving.

Every contact carries an age that sums two things: how long ago the receiver network last heard from it (`seen_pos`, or the AIS report timestamp) and how long ago this system fetched that answer. Reporting only the first would let a five minute old relay snapshot claim every contact in it was two seconds old. The age appears in the detail panel, in the tooltip past 20 seconds, as a table column, and on the map as opacity: contacts fade from full to 30 percent between 45 and 240 seconds, so a stale picture looks stale.

7.3 Finding your way around

The first layout put every control on screen at once, in one long scrolling rail. It was complete and it was overwhelming, especially for a visitor arriving from a link with no idea what they were looking at. The current layout groups the controls into three tabs, **Overview** (what is happening), **Filters** (what to show and how far to project) and **Areas** (where to load data and what to alert on), and adds a one-time welcome card with three quick starts that puts a first-time visitor in front of either the Washington airspace or the Baltic shipping before they have to learn anything. On a phone the three panels and the map become four full-screen views behind a bottom tab bar, with an alert count on the Alerts tab, rather than a tall page with the map at the top and everything else stacked underneath.

Every target also has its own address. Selecting one writes a fragment, `#JZA786` for a flight, the ICAO hex for an aircraft with no callsign, the MMSI for a ship, so a particular target can be linked, shared or returned to, the way a flight-tracking site gives each flight a page. It is a fragment rather than a path because this is a single static page with no server-side routing, and it is written with `replaceState` so selecting a dozen aircraft does not leave a dozen entries in the back button. Opening such a link selects that target as soon as it appears in the feed. Because the feed only carries the area being watched, a target named in a link may genuinely not be present, and after twenty seconds of looking the page says so and suggests moving the map or pinning the area, rather than sitting silently on a link that appears broken.

7.4 Flight history and where it is going

Selecting an aircraft answers three questions at once. Its **observed track** is the positions this system has actually seen, seeded from the history already held for that target and then extended for as long as it stays selected, drawn as a solid bright line: it is what we watched, not what we were told. Its **route legs** are the published origin and destination, drawn as great circles from the origin airport to the aircraft's current position and on to the destination, with the airports marked and labeled. The panel adds the distance flown from the origin, the distance remaining and an arrival time at the current ground speed, and a button frames the whole flight. When the route fails the plausibility checks of Section 2.4 none of that is drawn: the legs and the framing button are withheld, and the panel relabels the distances as distances to the two airports rather than progress along a flight.

Airport codes are shown as IATA rather than ICAO, because a flight from Charleston to Washington National is CRW to DCA on every board and boarding pass, not KCRW to KDCA. Where the route data carries no IATA code the code is still usually derivable rather than unknown: an ICAO identifier in the contiguous United States is K followed by the three-letter code, and in Canada C followed by a code beginning Y or Z, both by the structure of the ICAO location indicator system rather than by coincidence **[standard]**. Those two prefixes are dropped; everywhere else, including Alaska and Hawaii where the mapping is not one to one, the ICAO code is shown as it stands.

The origin is marked but never drawn to, which is a correction. A great circle from the departure airport to the aircraft's present position looks like the path flown and is not: real flights follow airways, take vectors, and hold. Jaron put it exactly right, that "it did not come from that spot". What can be drawn of the past is the track this system has watched, and that is the solid line, so the origin keeps its marker and loses its line.

There is no honest way to draw the rest. Historical tracks are not available from any keyless source: the trace endpoints of globe.adsb.fi, globe.airplanes.live and globe.adsbexchange.com all answer 403, and api.adsb.lol has no trace route at all, measured from an ordinary residential address rather than from the edge **[measured]**. The panel therefore says what the line is: positions actually watched, still extending, with nothing before this system first saw the target.

The first attempt to make that line longer widened the observed window for every target to 45 minutes and 400 positions, and it ran a large desktop out of memory, which the first author reported. Measured in a browser over a busy view of about 790 targets, whole-browser memory climbed in a straight line, from 888 MB to 981 MB between the first and eighth minute and still rising, because every target's entire history was also being rebuilt into trail geometry and handed to the map on every five-second tick: 26,335 coordinates per render at eight minutes, growing by about 3,300 a minute toward a plateau more than ten times that **[measured]**. The retained history was not the problem in itself. Using all of it everywhere was.

Retention is now tiered. The selected target keeps 45 minutes and 400 positions at full resolution, because it is the one whose track is drawn. Every other target keeps 15 minutes, which is what the landing rule's window needs, thinned to one position every 15 seconds unless it has turned 8 degrees, changed altitude by 500 feet, touched down or moved 2 NM, so turns and arrivals keep full detail while straight cruise costs a quarter as much. The newest position is always kept as a provisional head so a thinned trail still reaches its icon, and trails draw only the last 20 positions of the targets actually on screen. Re-measured under the same conditions, trail geometry settled at about 11,800 coordinates per render by the sixth minute and stayed there, and browser memory went from 898 MB to 937 MB over the same seven minutes and was flat for the last three **[measured]**.

It also exposed a coupling worth recording. The orbit rule summed turns over the whole retained history, so widening retention had silently changed what counted as an orbit. The rule now judges its own ten-minute window, whatever is retained, and a test pins that down.

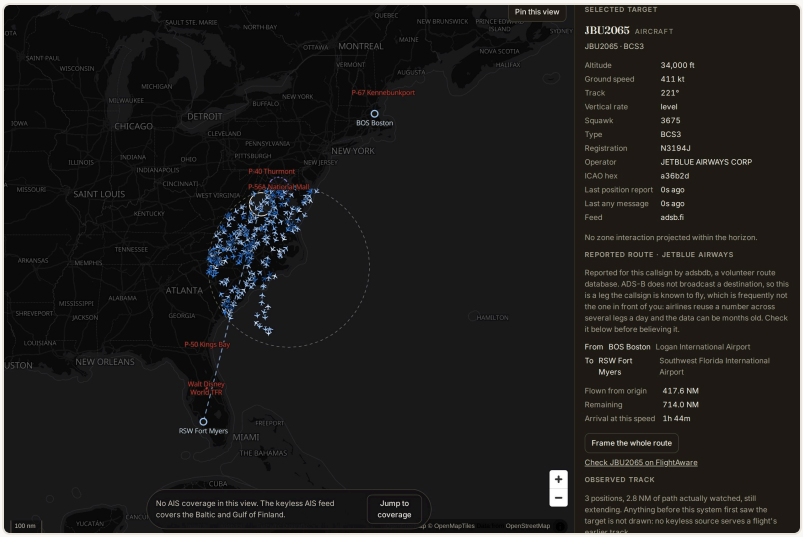

Figure 2. A selected flight. The solid line behind the aircraft is its observed track, the positions this system actually watched; the dashed leg ahead is the reported route on to its destination as a great circle. The departure airport is marked but not drawn to, for the reason given above. The rail lists the distance remaining and an arrival time at the current ground speed.

The legs are interpolated rather than drawn as straight lines, because a straight line between two airports is wrong on a Mercator projection: the shortest path curves. Sixty-five points along the great circle, using the standard intermediate-point formula **[documented]**, make the drawn path the flown path. Longitudes are unwrapped as the path is built, so a transpacific route does not draw itself the long way around the world; there is a test for exactly that, on Tokyo to Los Angeles.

7.5 One layout bug worth recording

The endpoint includes a snippet of the upstream response body in its error detail, which is how the adsb.fi 403 was identified as a bot challenge rather than a plain refusal. That text reached the interface verbatim, and a few hundred characters of nginx error page in a `white-space: nowrap` status chip widened the header enough to push the entire right-hand panel off screen. The fix has two halves: failures are summarized to a few words ("no source available: adsb.lol rate limited, adsb.fi blocked, opensky not responding") with the full text kept for the console, and the header, chips, layout grid and status bar are width-locked so no future content can widen the page whatever its origin. A regression test injects the exact string and asserts the panel is still on screen with zero document overflow. Any external text reaching page chrome is a layout hazard, and summarizing it is not sufficient alone.

The summarizer then failed on a shape it had not been written for. It took everything before the first colon in an upstream detail line as the name of the source, which is right for `adsb.lol: HTTP 429 - ...` and wrong for a proxy that answers with a bare sentence: a Cloudflare 520 page reached the status line in full, as "no source available: The origin web server returned an invalid or incomplete response to Cloudflare. This typically indicates the origin is overloaded or misconfigured." **[measured]**. A source name is one short token and never contains a space, so a pre-colon segment with spaces in it is prose and is now reported as `upstream`, classified by content where possible ("upstream not responding"), and every summary is clamped to 72 characters. The lesson generalizes past this system: parsing structure out of an error string works until something in the path substitutes its own error, and the fallback has to be a shrug rather than a quotation.

The same complaint had a second half: those diagnostics were sharing a line with the site footer, where an upstream's bad day sat next to the brand links. Feed health now has its own line above the footer, clipped to one line with the full text on the element's title, and it folds away to a single button which keeps a severity dot when a feed is down, so hiding the diagnostics cannot hide a dead feed. The footer holds what a footer should: the three site links and the not-for-navigation notice.

8. Verification

Unit tests (57, no network). The geometry and the rules: haversine and destination round-tripping, ray casting against known points, inside and projected alerts, severity escalation with closing time, altitude band exclusion, a descending target entering the band mid-projection, the small-zone step-over case, a target tracking away raising nothing, emergency squawks, orbit detection needing both a full turn and a small footprint, target-kind filtering, worst-first ordering, the ground clamp, the advisory-zone rule, and the real FAA zone file loading with its published limits intact. Eight cover the close-approach math: a head-on pair whose TCPA must equal range over closing speed, a parallel pair that never closes, a crossing pair whose CPA is its offset, a pair already past, one alert per pair regardless of input order, a harbor of 30 moored ships raising nothing, the severity bands, and the pairwise pass appearing in the combined result. Two more check the great-circle path drawn for a route: that its summed length matches the direct distance and that it bows poleward of the chord, and that a path across the antimeridian stays continuous.

Twelve more cover the route and status work of Section 2.4: the BCS30A case as a mismatch with its detour measured, an aircraft on the leg and pointed at the destination as consistent, a 25 NM reroute as consistent rather than wrong, a target on the leg but flying away from it caught by bearing, an aircraft 8 NM out and 90 degrees off left alone because it is turning onto an approach, one-endpoint routes claiming nothing, the Cloudflare 520 page summarized to four words, named upstream failures keeping their names, arbitrary 400-character error bodies clamped, and a paused feed not counting as a fault.

Four failed on first run and all four were genuine: the unclamped altitude extrapolation of 5.2, the great-circle convergence error in the test's own geometry in 5.1, the summarizer still promoting the word "The" to a source name, and a test of mine that asserted a destination-only route can never be contradicted when in fact its bearing check should fire. A suite that passes entirely on first write is usually testing what the code does rather than what it should do.

Browser smoke test (Playwright). Loads the real page in headless Chromium, fails on any console or page error, waits for live targets, asserts nothing rendered lies outside the viewport, selects a vessel and asserts its detail renders visibly, injects the hostile error string and asserts the layout survives, draws a zone and asserts it produces alerts, switches region, and captures desktop and phone screenshots, against local development or production by URL. Two refinements made it trustworthy: MapLibre cancels in-flight tile requests whenever the view moves, which surfaces as dozens of `ERR_ABORTED` failures that are normal behavior, and a 502 from this system's own endpoint is the documented upstream refusal the page is supposed to explain, so the test asserts the explanation appears rather than treating it as fatal.

Production measurement. The Section 4 figures were taken against the deployed system, and so was the end state: 446 to 496 aircraft between 2 and 12 seconds old via the relay, alongside 281 vessels **[measured]**.

9. Operations and limitations

Deployment is one command with no build step, so what is in the repository is what runs, and the zone file is regenerated when the FAA chart cycle turns. The relay runs as a long-lived process wherever there is an ordinary IP, kept alive by `forever`, with a systemd user unit documented for reboot persistence; its shared secret lives in a gitignored file the script reads itself, matching a Pages secret, and the relay endpoints compare the bearer token byte by byte after a length check. For diagnosis the interface comes first: the feed line names the path that served the data, its age and the last poll, and the banner distinguishes a rate-limited edge from a stopped relay.

One deployment property is worth recording because it made a verified fix invisible. Pages served the application's own modules with `cache-control: public, max-age=14400, must-revalidate` **[measured]**, and `must-revalidate` only takes effect once a response is stale, so for four hours after a deploy a browser that already had the page kept running the previous JavaScript without asking. A fix can therefore be deployed, smoke-tested in production and still absent for the person who reported the bug, who is the one most likely to have the page already open. That is not a hypothetical: it happened twice here, with the route check of Section 2.4 reported as missing when it was live and working.

The obvious remedy does not work. A `_headers` file asking for `max-age=0, must-revalidate` on the application's own files was ignored, and `Cache-Control: no-cache` came back from the live site as `max-age=14400` **[measured]**: Pages will not serve its static assets below that floor, though it honors values above it, which is why `vendor/*` successfully keeps a week. With no build step there is no content hashing in the filenames to force the issue either.

What is left is the one response that is always fresh. `index.html` is served with `max-age=0`, so the deployed build stamp can travel in a meta tag, and the same stamp is written into `app.js` in the copy that gets uploaded. A page whose script disagrees with its own HTML is therefore able to detect that it is old code and say so in the banner, above every other message, naming both stamps and how to force the reload. It cannot fix itself, but a stale page that declares itself stale stops costing someone else an afternoon. The general lesson: when a platform caches your code longer than your deploy cycle, verify against a cold cache, and give the page a way to notice.

Known limits, and the first one is the largest: **the route shown for a flight is frequently not the leg being flown.** It comes from a volunteer database keyed on the callsign, and Section 2.4 measures seven of ten wrong among arrivals at a single busy airport. The checks described there catch the ones that contradict the aircraft's own position, altitude and track, and what survives them is labeled as reported rather than as known, with a link to a source that does hold the day's schedule. A route that is wrong in a way the geometry cannot see will still be shown, and it is shown as a claim for that reason. AIS coverage is the Baltic and Gulf of Finland only, because that is what a keyless live feed covers; the projection is a straight line, right for a geofence warning and wrong for predicting behavior; NOTAM activation is not modeled, so a cold restricted area is still drawn and P-40's expansion during a presidential visit is not represented; AGL zones are compared against MSL altitude with terrain as the error; the relay is a dependency, and without it the aircraft feed degrades to intermittent; detection runs in the browser, so nothing is watching when nobody has the page open, which is why the engine is pure functions ready to move to a scheduled Worker; and adsb.fi's data is licensed for personal, non-commercial use **[documented]**, which this project is, while commercial use would need their permission and adsb.lol's ODbL terms would need attention on redistribution.

10. Provenance of the load-bearing claims

CLAIM	HOW IT IS KNOWN
ADS-B message formats and semantics	RTCA DO-260B, ICAO Annex 10 Vol IV **[standard]**; JSON field semantics from the readsb reference **[documented]**
`alt_baro` can be `"ground"`; 27 of 115 lacked `track`	Direct inspection of live responses **[measured]**
AIS encodings, sentinels, ship types, packed ETA	ITU-R M.1371 **[standard]**; ETA decoder validated live **[measured]**
Digitraffic needs gzip, 406 otherwise; `Digitraffic-User` raises limits; CC BY 4.0	Digitraffic instructions **[documented]**; 406 reproduced across four `Accept` values **[measured]**
adsb.lol is ODbL 1.0, open to everyone, limits load-dependent	adsb.lol open data documentation **[documented]**
adsb.fi: 1 req/s, invalid responses may trigger IP blocks, non-commercial, citation required, v2 deprecated for v3	adsb.fi opendata repository **[documented]**; v3 shape verified live **[measured]**
Edge behavior: adsb.lol 429s, adsb.fi 403 challenge, OpenSky 522	Purpose-built diagnostic endpoint in production **[measured]**
Success rates 1 in 8, 3 in 8 with retries, 1 in 6 later	Repeated polling of distinct viewports **[measured]**
No aggregator sends CORS headers; six alternative aggregators unusable	`Origin`-bearing requests inspected, and each candidate probed directly **[measured]**
D1 free tier: 5M read, 100k written per day; 50 queries per invocation	Cloudflare D1 pricing and limits **[documented]**
FAA: 13 prohibited areas, 6,285-point circles, `COUNTRY` is "UNITED STATES"	FAA SUA feature service queried directly **[measured]**
DC SFRA is 30 NM on the DCA VOR; prohibited areas under part 73	14 CFR 93 subpart V; 14 CFR 73 **[standard]**
Simplification 6,285 to 17 points; 506 vertices, 18.6 KB	Output of the committed ingest script **[measured]**
Haversine and destination formulas	Movable Type latitude/longitude reference **[documented]**
Palette contrast and color-vision figures	Validator run against the page's dark surface, re-run after the rebrand **[measured]**
Chrome palette and type are the other two sites' own tokens	Read from the live jaronwilson.dev and jaronwilson.org stylesheets **[measured]**
55 alerts before advisory zones, 3 after	Same view, before and after **[measured]**
241 aircraft shown where the feed reported 1	Region switch with pruning disabled **[measured]**
End state: 446 to 496 aircraft, 2 to 12 s old, 281 vessels	Public endpoints polled directly **[measured]**
CPA and TCPA are the standard measures, with alarms on a preset CPA limit and TCPA warning time	IMO Resolution A.823(19), ARPA performance standards **[standard]**
Origin and destination are not in ADS-B and come from adsbdb; 404 means no scheduled route	adsbdb documentation **[documented]**; live callsigns resolved and 404s observed **[measured]**
adsbdb answers the Cloudflare edge, unlike the ADS-B aggregators	`/api/route` exercised from the deployed Worker **[measured]**
A callsign-keyed route can contradict the aircraft: BCS30A resolved to EDDP-EDDK, 489 NM from one and 304 NM from the other on a 195 NM leg	Observed in the running system, 17 September 2026 **[measured]**
A proxy error with no colon in it was quoted verbatim in the status line	Cloudflare 520 body observed in the interface **[measured]**
Second route case: SWA1246 over Washington reported as KIAH-KMSY, 1,048 and 842 NM from them on a 264 NM leg	Live lookup and position compared, 17 September 2026 **[measured]**
hexdb.io disagrees with adsbdb on every sampled callsign it knows, including one where adsbdb was right	Both sources queried for ten callsigns **[measured]**
Seven of ten routes wrong among aircraft on the ground or descending at Washington National; eight of ten flagged afterwards, with no correct route rejected	Live feed and route lookups compared against position, altitude and vertical rate, 18 September 2026 **[measured]**
No keyless source serves a flight's historical track: every trace endpoint probed answers 403	globe.adsb.fi, globe.airplanes.live, globe.adsbexchange.com and api.adsb.lol probed from a residential address **[measured]**
Keeping 45 minutes of history for every target grew the browser linearly (888 to 981 MB in seven minutes at ~790 targets); tiered retention and 20-point trails flattened it (898 to 937 MB, flat for the last three)	Whole-browser memory and per-render trail coordinates sampled over the same view before and after **[measured]**

CLAIM	HOW IT IS KNOWN
Pages served application modules with `max-age=14400, must-revalidate`, hiding a deployed fix for four hours, and will not honor a shorter value in `_headers`	Response headers read from the live site before and after a `_headers` change **[measured]**
275 vessels gave 4 approach alerts at 1 NM, 27 at 5 NM; tightest 0.32 NM in 2m 14s	Detector run against live Baltic traffic **[measured]**

11. Development method: directing an AI implementer

This system was built by one person directing an AI coding agent, Claude, through Anthropic's Claude Code, over four working days and 27 commits between 16 and 22 September 2026. That arrangement is worth describing as a method in its own right, because it shaped both what went right and what went wrong.

The division of labor was deliberate. The first author owned the goals, the constraints and every decision with more than one reasonable answer: the stack, chosen from a set of offered alternatives; the relay, chosen from four options once the egress problem had been measured; the feature scope; the visual identity, matched to his own sites; and acceptance, meaning whether a change was actually right when used. The AI owned implementation, instrumentation, measurement, tests and first drafts of the documentation, each in response to a specific request.

The working loop was short and always ran through the deployed system rather than a description of it:

1. A request or a defect report, in plain words ("clicking for detail on boats does not work", "it did not come from that spot").
2. A reproduction, preferably a measurement rather than a reading of the code.
3. A root cause, stated before any fix was written.
4. A fix, deployed to production.
5. A guard: a unit test, a browser smoke-test assertion, an overflow check or a pixel check, so that the same defect could not come back silently.
6. Acceptance by the first author, using the live site.

Two properties of an AI implementer made steps 2 and 5 non-negotiable. It is fast, which means a wrong assumption reaches production quickly; and it can be confidently wrong, which means its own report that something works is not evidence that it does. The record in Section 12 shows both: the AI introduced the page-margin workaround that shipped blank pages, the retention change that ran a desktop out of memory, and a text substitution that silently did nothing. The guards caught the first; the first author caught the other two. The provenance tags used throughout this paper are the same discipline applied to prose: a claim is marked as measured, documented or standard so that neither author has to be taken on trust.

12. Defects found in use

The defects below were found by using the deployed system. Most were found by the first author, working from the live site; the rest were caught by the guards added in earlier rounds, which is the point of adding them.

DEFECT	FOUND BY	ROOT CAUSE	FIX AND GUARD
Clicking a ship appeared to do nothing	J.M.W.	Its detail rendered below the fold of a long alert list	Detail moves to the top of the rail on selection; the smoke test clicks a vessel and checks it is on screen
Right-hand panel vanished during an outage	J.M.W.	A few hundred characters of upstream error page widened a no-wrap header	Errors summarized to a few words and the layout width-locked; a test injects the exact string (Section 7.5)
No aircraft in production	J.M.W.	Aggregators rate-limit or block shared cloud egress addresses	The relay of Section 4, his choice of four measured options
A drawn box did nothing	J.M.W.	A code patch that did not check its target had silently matched nothing	Every patch now asserts that it matched before writing
Altitude chart showed stray gray lines	J.M.W.	The bars were inline elements and had never rendered at all	Rendered as blocks
Last slide of the deck clipped	J.M.W.	Copy longer than the fixed square slide	Slides auto-fit, and the build fails on any overflow
White frame around every page of this paper	J.M.W.	Chromium leaves page margins unpainted	CSS page margin boxes, following his `@page` suggestion; every page edge checked by pixel
A blank paper committed	Guard	A header-template workaround covered each page	A dark-pixel count on every page before commit
Status text leaking a proxy error page	J.M.W.	A source name parsed as everything before the first colon	Names must be one token and every summary is clamped; a unit test on the exact text
Routes belonging to a different flight	J.M.W.	A callsign is a flight number, not a leg (Section 2.4)	Four plausibility checks, after measuring seven wrong in ten at DCA
A fixed defect still visible	J.M.W.	The platform caches scripts for four hours and ignores shorter settings	A build stamp that lets a stale page say so (Section 9)
Airport codes shown as ICAO	J.M.W.	The data's identifier, not the one people read	IATA first, with the K and C conventions as fallback; unit tested
A line from the origin that no flight flew	J.M.W.	A great circle to the current position claimed an unobserved path	The origin is marked, never drawn to; the smoke test asserts it
A desktop ran out of memory	J.M.W.	Every target's full history re-rendered as trails on every tick	Tiered retention and 20-point trails, measured flat (Section 7.4)
Vessel detail below the fold again	Guard	Selection layout lived only in the click handler	Layout follows the selection state on every update
A button off the site's theme	J.M.W.	Styled on its own rather than as a standard button	Uses the shared button style; the smoke test compares its computed style

13. Lessons learned

1. **Measure before choosing an architecture.** The obvious design, fetching from the edge, failed for a reason no reading of the documentation would have revealed: the edge shares its egress addresses with everyone else.
2. **Test from outside.** Most of the defects in Section 12 were found by using the deployed site, not by the tests written alongside the code. The tests then kept them from coming back.
3. **Give every fix a guard.** A regression test, an overflow check or a pixel check turns "fixed" from a claim into something the build enforces.
4. **Verify the artifact, not a view of it.** A screenshot of a PDF viewer looked fine while the committed file was blank; counting pixels on the file itself caught it.
5. **Label what you cannot verify, and check it where you can.** Route data was wrong seven times in ten at one airport; the fix was to test it against the aircraft and say plainly what remains a claim.
6. **Two sources of the same kind of data are not corroboration.** A second route database disagreed with the first on every callsign sampled, including one where the first was right.
7. **Know the platform's defaults.** A four-hour script cache made a deployed fix invisible to the person who had reported the bug.
8. **Bound everything per viewer, and let rules own their windows.** Keeping more history for every target cost memory in a straight line, and silently changed what the orbit rule meant.
9. **An AI implementer is a multiplier, not an authority.** It made this system possible in days rather than weeks, and it also introduced several of the defects above. Requirements, judgment and acceptance stayed human, and measurement is what made its work checkable.

14. Author contributions

J.M.W. conceived the project and directed it throughout. He set the requirements and the priorities, chose the platform after being offered the alternatives (a static site with edge functions, over Angular, Java Spring Boot and Python FastAPI), and chose the relay architecture of Section 4 once the egress problem had been measured, which is the decision the whole system rests on. He specified the feature scope: the pinned tracking areas that keep loading while the map is scrolled elsewhere, per-feed pausing, vessel close-approach prediction, flight history and routes, the tabbed navigation, per-target URLs, and landing detection. He runs the relay on his own infrastructure.

He also did the research and the acceptance testing, and they are the reason much of this system is correct rather than merely finished. He verified reported routes flight by flight against an independent schedule source, which is how the route failure of Section 2.4 was found and then measured, and he investigated receiver hardware for a first-party ADS-B feed. Working from the deployed build rather than from a description of it, he found fourteen of the sixteen defects in Section 12, including the one that shaped the design most: the absence of live aircraft in production that led to the relay. Each of them is a defect that testing from the outside catches and testing from the inside does not. He directed the method of Section 11 and reviewed this text.

Claude (Anthropic) implemented most of the software, ran the measurements reported as **[measured]**, wrote the unit and browser tests, produced the figures, and drafted this paper, all under the direction of the first author. The provenance tags in the text and the table in Section 10 exist so that a reader can check any load-bearing claim against its source rather than trust either author.

15. Availability

The system is live at `flysdown.jaronwilson.dev`, served from Cloudflare Pages. The source, including the tests, the tools that regenerate the zone file, the figures and this document, is in the repository `Jaron-Wilson/flysdown` (private at the time of writing; contact the first author). This paper and its slide version are published at `flysdown.jaronwilson.dev/docs/flysdown-paper.pdf` and `flysdown.jaronwilson.dev/docs/flysdown-linkedin.pdf`, linked from the dashboard's footer. Aircraft data is used under adsb.fi's personal, non-commercial terms with the required citation, and under adsb.lol's ODbL 1.0. Vessel data is Fintraffic Digitraffic, CC BY 4.0. Airspace geometry is the FAA's, in the public domain. Route data is from adsbdb (MIT). The basemap is OpenFreeMap, built on OpenStreetMap data, copyright OpenStreetMap contributors. Nothing here is for navigation.

16. References

1. RTCA DO-260B, *MOPS for 1090 MHz Extended Squitter ADS-B*; ICAO Annex 10 Vol IV.
2. readsb JSON output reference.
   https://github.com/wiedehopf/readsb/blob/dev/README-json.md
3. FAA, *Aeronautical Information Manual*, ch. 6 s. 2.
   https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap6_section_2.html
4. ITU-R Recommendation M.1371.
   https://www.itu.int/rec/R-REC-M.1371
5. Fintraffic Digitraffic marine APIs and client instructions, CC BY 4.0.
   https://www.digitraffic.fi/en/marine-traffic/
   https://www.digitraffic.fi/en/support/instructions/
   https://meri.digitraffic.fi/swagger/
   https://creativecommons.org/licenses/by/4.0/
6. ADSB.lol open data API (ODbL 1.0).
   https://www.adsb.lol/docs/open-data/api/
   https://api.adsb.lol/docs
   https://opendatacommons.org/licenses/odbl/1.0/
7. adsb.fi open data API, endpoints, rate limits and usage policy.
   https://github.com/adsbfi/opendata
8. The OpenSky Network REST API.
   https://openskynetwork.github.io/opensky-api/rest.html
9. FAA Aeronautical Information Services open data and the Special Use Airspace feature service.
   https://ais-faa.opendata.arcgis.com/
   https://services6.arcgis.com/ssFJjBXIUyZDrSYZ/ArcGIS/rest/services/Special_Use_Airspace/FeatureServer/0
10. 14 CFR part 73, *Special Use Airspace*.
    https://www.ecfr.gov/current/title-14/chapter-I/subchapter-D/part-73
11. 14 CFR part 93 subpart V, *Washington, DC Metropolitan Area SFRA*.
    https://www.ecfr.gov/current/title-14/chapter-I/subchapter-D/part-93/subpart-V
12. D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", *Cartographica*, 1973.
13. C. Veness, *Calculate distance, bearing and more between latitude and longitude points*.
    https://www.movable-type.co.uk/scripts/latlong.html
14. Cloudflare Pages Functions.
    https://developers.cloudflare.com/pages/functions/

15. Cloudflare Workers Cache API.
    https://developers.cloudflare.com/workers/runtime-apis/cache/
16. Cloudflare D1 pricing and limits.
    https://developers.cloudflare.com/d1/platform/pricing/
    https://developers.cloudflare.com/d1/platform/limits/
17. MapLibre GL JS API and style specification.
    https://maplibre.org/maplibre-gl-js/docs/API/
    https://maplibre.org/maplibre-style-spec/layers/
18. OpenFreeMap keyless OpenStreetMap vector tiles.
    https://openfreemap.org/
19. OpenStreetMap contributors.
    https://www.openstreetmap.org/copyright
20. adsbdb, aircraft and flight route API (MIT license; route data credited to PlaneBase, David Taylor and Jim Mason).
    https://github.com/mrjackwills/adsbdb
    https://api.adsbdb.com
21. IMO Resolution A.823(19), *Performance Standards for Automatic Radar Plotting Aids (ARPAs)*, adopted 23 November 1995.
    https://wwwcdn.imo.org/localresources/en/KnowledgeCentre/IndexofIMOResolutions/AssemblyDocuments/A.823(19).pdf

All URLs were retrieved and confirmed reachable on 17 September 2026. Two sources are cited from their own error responses rather than rendered documentation: adsb.fi's home page returns 403 to automated clients, which is the behavior described in Section 4, and the historical OpenSky REST documentation URL now returns 404 in favor of reference 8.