ADS-B Aircraft Tracking with RTL-SDR: Real-Time Surveillance via 1090 MHz Signal Reception and Decoding

STSGYM Research • Technical Report

satellite.stsgym.com — ADS-B Aircraft Tracking System

April 2026

Abstract. Automatic Dependent Surveillance–Broadcast (ADS-B) is the foundation of next-generation air traffic surveillance, enabling aircraft to broadcast their position, velocity, and identity via 1090 MHz radio transmissions. This paper presents the design and implementation of a low-cost, real-time ADS-B receiving and tracking system built on RTL-SDR software-defined radio receivers, the dump1090-mutability decoder, and a PostgreSQL/Redis data pipeline with a Flask web API. We describe the complete signal processing chain from 1090 MHz RF reception through message decoding—including the Compact Position Reporting (CPR) algorithm for position recovery—to persistent storage and live web visualization. The system integrates with the OpenSky Network for supplemental data, enabling tracking of over 5,000 aircraft across the continental United States. We analyze ADS-B message structure (120-bit Extended Squitter frames with CRC-24 integrity), SDR signal processing parameters (1 Mbps PPM modulation, gain optimization, frequency correction), and database architecture (time-partitioned position tables with optimized indexing). Performance measurements show reception rates of 50,000–100,000 position reports per hour and a storage footprint of approximately 1 GB per week for raw positions. The system operates as part of the satellite.stsgym.com ecosystem,

> demonstrating that commodity SDR hardware can provide surveillance capabilities approaching commercial services at a fraction of the cost.

Table of Contents

1. Introduction

Air traffic surveillance has undergone a fundamental transformation over the past two decades, moving from ground-based primary radar systems toward cooperative surveillance technologies in which aircraft actively broadcast their state. The most significant of these technologies is Automatic Dependent Surveillance–Broadcast (ADS-B), a system in which aircraft derive their position from satellite navigation (typically GPS/GNSS) and periodically broadcast it, along with identification and velocity information, enabling both ground stations and nearby aircraft to track them [1].

The ADS-B system operates on two frequencies: 1090 MHz (the "Extended Squitter" or 1090ES link) and 978 MHz (the Universal Access Transceiver or UAT link). The 1090ES link is the predominant standard worldwide, mandated for most airspace by the International Civil Aviation Organization (ICAO) and regulatory bodies such as the FAA and EASA [2]. Each broadcast is a 120-bit message transmitted at 1 Mbps using pulse position modulation (PPM), with a 24-bit cyclic redundancy check (CRC-24) ensuring message integrity.

The public nature of ADS-B transmissions—they are unencrypted and can be received by anyone with suitable equipment—has given rise to a vibrant community of hobbyists, researchers, and commercial services that operate independent receiving networks. This paper describes one such system: a low-cost ADS-B receiving station built from commodity RTL-SDR hardware, integrated into the satellite.stsgym.com ecosystem, and providing real-time aircraft tracking data through a Flask-based web API and interactive map interface.

1.1 ADS-B and the Surveillance Landscape

Traditional air traffic surveillance relies on primary surveillance radar (PSR), which detects aircraft by reflecting radio waves off their airframes, and secondary surveillance radar (SSR), which interrogates aircraft transponders. These systems have significant limitations: PSR provides no identification data and has limited accuracy, while SSR requires active interrogation and suffers from latency due to rotating antenna sweep rates [3].

ADS-B addresses these limitations by shifting to a "broadcast" paradigm. Aircraft equipped with ADS-B Out transmitters periodically broadcast their position, altitude, velocity, and identity without requiring interrogation. The update rate is approximately once per second for position messages, providing near-real-time surveillance that far exceeds the 4–12 second update intervals of rotating SSR antennas. Ground stations need only a simple receiver, dramatically reducing infrastructure cost compared to rotating radar installations [4].

As of 2026, ADS-B Out is mandatory in most controlled airspace worldwide. The FAA's ADS-B Final Rule requires it above 18,000 feet MSL, in Class A, B, and C airspace, and above Class B and C airspace in the United States. European mandates (EASA) require it for instrument flight rules (IFR) operations and above flight level (FL) 290. This regulatory environment ensures a high and growing equipage rate, making ADS-B an increasingly complete picture of air traffic.

1.2 Contributions

This paper makes the following contributions:

- A complete description of an end-to-end ADS-B receiving system from RF signal to web visualization, including hardware selection, SDR configuration, message decoding, and data persistence.
- A detailed analysis of the ADS-B protocol at the bit level, including downlink format classification, Compact Position Reporting (CPR) decoding with full mathematical treatment, and CRC-24 integrity verification.
- A database architecture optimized for high-ingest-rate time-series aircraft position data, with partitioning and indexing strategies for efficient querying.
- Performance characterization of a single-receiver installation, including reception range, message throughput, and storage requirements.

2. System Architecture

The ADS-B tracking system follows a pipeline architecture in which radio-frequency signals are received, digitized, decoded, stored, and served through a web API. Figure 1 illustrates the complete data flow.

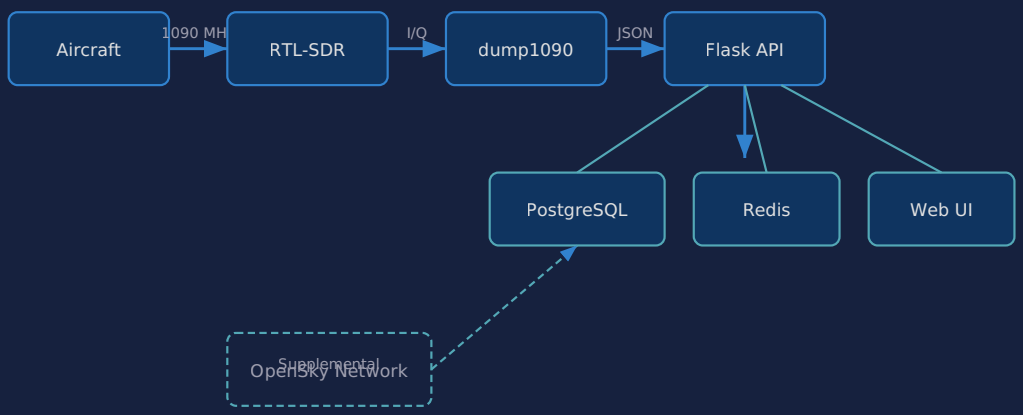

Figure 1: ADS-B data flow from aircraft broadcast through reception, decoding, storage, and web delivery.

2.1 Reception Chain

The reception chain begins with a 1090 MHz antenna, typically a quarter-wave monopole or a dedicated ADS-B antenna with 5–8 dBi gain. The antenna feeds an RTL2832U-based SDR dongle (commonly the RTL-SDR Blog V3 or Nooelec NESDR Smart), which digitizes the RF signal using the Elonics E4000 or Rafael Micro R820T/2 tuner chip. The RTL2832U provides 8-bit I/Q samples at rates up to 2.4 Msps, though the effective noise-free bandwidth is approximately 2.4 MHz due to the ADC resolution [5].

For ADS-B reception, the SDR is tuned to 1090 MHz with a sample rate of 2.4 Msps. This sample rate provides sufficient bandwidth to capture the 1 MHz PPM-modulated ADS-B signal while keeping the USB transfer rate manageable. The dongle connects to the host system via USB 2.0, which provides adequate bandwidth for the digitized I/Q stream.

2.2 Decoding Pipeline

The decoded I/Q samples are passed to `dump1090-mutability`, a widely-used open-source ADS-B decoder that operates entirely in software. The decoder performs the following steps:

1. **Signal detection:** Scan the I/Q stream for the 8-microsecond preamble that precedes each ADS-B message.

2. **Demodulation:** Decode the PPM-modulated data bits by comparing energy in the first and second half of each 1-microsecond symbol period.
3. **CRC verification:** Compute CRC-24 over the 88 data bits and compare with the 24-bit checksum. Messages failing CRC are discarded.
4. **Message parsing:** Extract downlink format, ICAO address, and type-specific payload fields from verified messages.
5. **Position decoding:** Apply the CPR algorithm to recover latitude and longitude from odd/even position message pairs.

An alternative decoder, `readsb`, offers improved performance for multi-receiver setups and provides built-in forwarding to aggregation services. Both decoders output JSON-formatted aircraft state on port 30003 and via an HTTP interface.

2.3 Data Storage and Caching

The system employs a dual-store architecture for data persistence and real-time access:

PostgreSQL serves as the persistent store, maintaining aircraft metadata and historical position data. The schema (detailed in Section 6) separates static aircraft information (registration, type, operator) from time-varying position records. This separation allows efficient querying: aircraft lookups are fast key-value operations, while position history queries leverage time-ordered indexes.

Redis provides a low-latency cache for live aircraft state. Each aircraft's most recent position is stored as a Redis hash with a 60-second TTL, enabling the `/api/aircraft/live` endpoint to return current data without querying PostgreSQL. This architecture reduces database load during peak traffic and ensures sub-second response times for the live map display.

2.4 Web Interface

The web interface at `satellite.stsgym.com` provides real-time aircraft visualization using Leaflet.js for map rendering. Aircraft are displayed as icons with altitude-based color coding, and clicking an icon reveals detailed information (callsign, registration, altitude, speed, track). Trail visualization shows recent flight paths, and historical playback enables review of past traffic patterns. The interface polls the Flask API at one-second intervals for live updates.

3. ADS-B Protocol Analysis

3.1 1090 MHz Extended Squitter

ADS-B on 1090 MHz uses the Mode S Extended Squitter (1090ES) format, which repurposes the existing Mode S transponder infrastructure to broadcast surveillance data. The 1090 MHz frequency was chosen for backward compatibility: Mode S transponders already operate on this frequency for selective interrogation (uplink at 1030 MHz, downlink at 1090 MHz), and the Extended Squitter format extends Mode S with longer, periodically broadcast messages [6].

The ADS-B 1090ES system uses a slotted Aloha-like access scheme: each aircraft broadcasts messages at semi-random intervals with a nominal rate of approximately 6.2 messages per second for the core position message (DF=17, TC=9–18). The randomization avoids systematic collisions between aircraft on similar trajectories.

3.2 Message Format and Downlink Formats

Each ADS-B message is 120 bits long (112 data bits + 24 parity bits), transmitted at 1 Mbps using on-off keying with pulse position modulation. The message is preceded by an 8-microsecond preamble of four pulses that allows receivers to detect the start of a transmission and synchronize the decoding clock [7].

The first 5 bits of the message encode the **downlink format (DF)**, which determines the message structure:

DF	Type	Length	Use in ADS-B
0	Short Air-Air Surveillance	56 bits	Not ADS-B
4	Surveillance Altitude Reply	56 bits	Not ADS-B
5	Surveillance Identity Reply	56 bits	Not ADS-B
11	All-Call Reply	56 bits	Not ADS-B
16	Long Air-Air Surveillance	112 bits	Not ADS-B
17	Extended Squitter (ADS-B)	112 bits	**Primary ADS-B format**
18	Extended Squitter (Non-Transponder)	112 bits	ADS-B from vehicles without Mode S
19	Extended Squitter (Military)	112 bits	Military ADS-B
20–21	Comm-B / Comm-D	112 bits	ELM, not ADS-B

For DF=17 (and DF=18/19), the 112-bit payload is structured as:

$$\underbrace{\text{DF}}_{5\text{ bit}} \; \underbrace{\text{CA/AA}}_{3\text{ bit}} \; \underbrace{\text{ICAO}}_{24\text{ bit}} \; \underbrace{\text{ME}}_{56\text{ bit}} \; \underbrace{\text{PI}}_{24\text{ bit}}$$

where DF is the downlink format, CA is the capability field (or AA for DF=18), ICAO is the 24-bit aircraft address (hex identifier), ME is the 56-bit message payload, and PI is the 24-bit parity/CRC field.

3.3 BDS Codes and Message Types

The 56-bit ME field is further parsed according to the **type code (TC)** in its first 5 bits, which maps to BDS (Comm-B Data Selector) codes:

TC	BDS	Content
1–4	BDS0,8	Aircraft identification (callsign)
5–8	BDS0,6	Surface position
9–18	BDS0,5	Airborne position (barometric altitude)
19	BDS0,5	Airborne velocity
20–22	BDS0,5	Airborne position (GNSS altitude)
28	BDS1,0	Aircraft status
29	BDS1,7	Target state and status
31	BDS1,7	Aircraft operation status

The most frequently observed message types are TC=9–18 (airborne position with barometric altitude, broadcast approximately every 0.5 seconds alternating odd/even CPR format) and TC=19 (airborne velocity, broadcast approximately every 0.5 seconds).

4. SDR Signal Processing

4.1 RTL-SDR Characteristics

The RTL2832U dongle is a mass-market DVB-T receiver that has been repurposed as a general-purpose SDR. Its key specifications for ADS-B reception include:

Parameter	Specification
Tuner chip	Rafael Micro R820T/2
Frequency range	24–1766 MHz
ADC resolution	8-bit (real), 8-bit I + 8-bit Q
Maximum sample rate	2.4 Msps (stable)
Bandwidth	~2.4 MHz (at 2.4 Msps)
Typical noise figure	3.5–5 dB
USB interface	USB 2.0
Cost	$25–$40 USD

The 8-bit ADC provides approximately 48 dB of dynamic range, which is adequate for ADS-B reception but marginal for environments with strong interfering signals. The R820T/2 tuner includes a built-in low-noise amplifier (LNA) with adjustable gain from 0 to 49.6 dB in 0.5 dB steps.

4.2 Sample Rate and Frequency Correction

ADS-B messages occupy approximately 1 MHz of bandwidth at 1090 MHz. The RTL-SDR is configured with a 2.4 Msps sample rate, which places the ADS-B signal well within the digitized bandwidth. The center frequency is set to 1090 MHz, and the PPM correction value (typically ±30–60 ppm for

an unmodified crystal oscillator) is applied to compensate for the dongle's frequency offset.

Frequency correction is critical because the R820T/2's crystal oscillator has a typical initial accuracy of ±30 ppm, meaning the actual tuning frequency may differ from the commanded frequency by up to 32.7 kHz at 1090 MHz. The `dump1090` decoder includes an automatic frequency correction algorithm that estimates the offset from the observed preamble frequency, typically achieving residual errors below 1 ppm [8].

4.3 Gain Optimization

Selecting the appropriate gain is a critical trade-off in ADS-B reception. Too little gain reduces sensitivity, causing distant aircraft to be lost. Too much gain amplifies noise and can cause front-end overload from strong nearby signals (FM broadcast, cellular base stations), reducing overall decoding performance.

The optimal gain setting depends on the local RF environment. In quiet rural locations, gains of 40–50 dB typically yield the best results. In urban environments with strong interferers, gains of 30–40 dB may be necessary to avoid overload. The `dump1090` “auto” gain mode adjusts gain dynamically based on the noise floor, though manual optimization typically outperforms automatic settings by 5–15% in message yield.

4.4 Signal-to-Noise Considerations

The signal-to-noise ratio (SNR) at the receiver input determines whether a message can be successfully decoded. For an ADS-B message at 1090 MHz, the received power at distance d from an aircraft at altitude h is given by the free-space path loss model with adjustments for the Earth's curvature:

$$P_r = P_t G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2$$

where P_t is the transmitter power (typically 125–500 W for Mode S transponders), G_t is the transmit antenna gain (approximately 0 dBi

for a bottom-mounted antenna), G_r is the receive antenna gain, and $\lambda = c/f \approx 0.275$ m at 1090 MHz.

The effective reception range for an aircraft at altitude h is limited by the radio horizon:

$$d_{\max} \approx 4.12 \left(\sqrt{h_{\text{aircraft}}} + \sqrt{h_{\text{receiver}}}\right) \text{ km}$$

where altitudes are in meters. For an aircraft at FL350 (10,668 m) and a ground receiver with an antenna at 10 m, the maximum theoretical range is approximately 440 km. In practice, terrain, antenna performance, and receiver sensitivity reduce this to 200–350 km for reliable decoding.

5. Message Decoding

5.1 Downlink Format Parsing

The decoder first identifies the downlink format from the 5 most significant bits of the message. DF=17 messages (Extended Squitter) are the primary target for ADS-B tracking. The 24-bit ICAO address is extracted from bits 9–32 and serves as the unique identifier for each aircraft, enabling correlation of successive messages from the same source.

CRC-24 verification is performed using the generator polynomial:

$$G(x) = x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^{9} + x^{8} + x^{7} + x^{6} + x^{5} + x^{4} + x^{3} + x^{2} + x + 1$$

In practice, the CRC is computed efficiently using a lookup table. Messages that fail CRC are discarded, as bit errors corrupt the position and identity data unrecoverably. The 24-bit CRC provides a probability of

undetected error of approximately $2^{-24} \approx 6 \times 10^{-8}$ per message [9].

5.2 Position Decoding via Compact Position Reporting

ADS-B encodes position using the Compact Position Reporting (CPR) algorithm, which compresses latitude and longitude into fewer bits than direct encoding to reduce transmission bandwidth. CPR uses two alternating formats—**odd** and **even**—each providing 17 bits of resolution for both latitude and longitude within the 56-bit ME field [10].

The CPR encoding divides the world into zones. For even messages, the world is divided into $NZ = 15$ latitude zones. For odd messages, it is divided into $NZ = 15 - 1 = 14$ zones. Each latitude zone spans:

$$\Delta \text{Lat}_{\text{even}} = \frac{360}{4 \times 15} = 6^\circ, \quad \Delta \text{Lat}_{\text{odd}} = \frac{360}{4 \times 14} \approx 6.429^\circ$$

Within each zone, the 17-bit encoded latitude represents a fraction of the zone width, yielding a resolution of approximately:

$$\text{Lat resolution} = \frac{\Delta \text{Lat}}{2^{17}} \approx \frac{6^\circ}{131072} \approx 0.000046^\circ \approx 5.1 \text{ m}$$

To decode a position, one odd and one even message must be received from the same aircraft. The globally unique latitude is recovered by solving for the latitude zone index using the **NL function** (number of longitude zones at a given latitude):

$$\text{NL}(\text{lat}) = \left\lfloor \frac{2\pi}{\arccos\left(1 - \frac{1-\cos\left(\frac{\pi}{2 \times 15}\right)}{\cos^2\left(\frac{\pi}{180}|\text{lat}|\right)}\right)} \right\rfloor$$

The latitude zone index j is computed as:

$$j = \left\lfloor \frac{17 \cdot \text{Lat}_{\text{even}} - 17 \cdot \text{Lat}_{\text{odd}} + 15}{2} \right\rfloor$$

and the decoded latitude is:

$$\text{Lat} = \Delta \text{Lat} \times \left(j + \frac{\text{CPR}_{\text{format}}}{2^{17}}\right)$$

Longitude decoding proceeds similarly, using the NL function to determine the number of longitude zones at the computed latitude. The longitude zone index is:

$$m = \left\lfloor \frac{17 \cdot \text{Lon}_{\text{even}} - 17 \cdot \text{Lon}_{\text{odd}} + \text{NL}-1}{2} \right\rfloor$$

and the decoded longitude is:

$$\text{Lon} = \Delta \text{Lon} \times \left(m + \frac{\text{CPR}_{\text{format}}}{2^{17}}\right)$$

where $\Delta \text{Lon} = 360 / \text{NL}$ for the appropriate NL value. For positions near the poles where NL = 1, the full 360° of longitude are encoded in a single zone.

> **CPR Ambiguity Resolution.** When only one format (odd or even) is available, a “local” decode can be performed using the receiver's known position to resolve the zone ambiguity. This is less accurate than global decoding but provides single-message position estimates for aircraft already tracked.

5.3 Velocity Extraction

Airborne velocity messages (TC=19) encode ground speed, track angle, vertical rate, and geometric altitude difference. Two subtypes exist:

- **Subtype 1 (Ground track):** Encodes ground speed and track angle directly, with speed resolution of 1 knot and track resolution of approximately 1.4°.

- **Subtype 3 (Airspeed heading):** Encodes indicated airspeed and magnetic heading, useful for tracking aircraft that are not navigating by GPS.

For subtype 1 messages, the ground speed is encoded as:

$$v = \sqrt{v_{\text{EW}}^2 + v_{\text{NS}}^2}$$

where the East-West and North-South velocity components are each encoded in 11 bits with a sign bit, providing a range of 0–4096 knots with 1-knot resolution. The track angle is derived as:

$$\theta = \arctan2(v_{\text{EW}}, v_{\text{NS}})$$

Vertical rate is encoded in 9 bits with a sign bit and a resolution of 64 ft/min, covering a range of ±16,384 ft/min (±83 m/s).

5.4 Callsign and Identity Extraction

Aircraft identification messages (TC=1–4) encode the callsign in 48 bits using a 6-bit character set (ICAO Annex 10, Table A-2). The 8-character callsign is left-justified and space-padded. The type code encodes the aircraft category:

TC	Category
1	Light (<15,500 kg)
2	Small (15,500–34,000 kg)
3	Large (34,000–136,000 kg)
4	Heavy (>136,000 kg) or high vortex

The callsign is typically the IATA flight number (e.g., `UAL456` for United Airlines flight 456) and can be mapped to the aircraft registration through external databases. Our system enriches callsign data using the OpenSky

Network aircraft database, which provides registration, type code, and operator information.

6. Database and Storage Architecture

6.1 PostgreSQL Schema

The database uses two primary tables that separate static aircraft metadata from time-varying position records:

Aircraft Table

```
CREATE TABLE aircraft (
    id              SERIAL PRIMARY KEY,
    icao_hex        VARCHAR(6) NOT NULL UNIQUE,
    registration    VARCHAR(20),
    callsign        VARCHAR(10),
    aircraft_type   VARCHAR(10),  -- ICAO type designator (e.g., B738)
    operator        VARCHAR(50),
    origin          VARCHAR(4),   -- Departure airport ICAO
    destination     VARCHAR(4),   -- Arrival airport ICAO
    first_seen      TIMESTAMP WITH TIME ZONE,
    last_seen       TIMESTAMP WITH TIME ZONE
);
```

AircraftPosition Table

```
CREATE TABLE aircraft_positions (
    id              BIGSERIAL PRIMARY KEY,
    aircraft_id     INTEGER REFERENCES aircraft(id),
    timestamp       TIMESTAMP WITH TIME ZONE NOT NULL,
    latitude        DOUBLE PRECISION,
    longitude       DOUBLE PRECISION,
    altitude_baro   INTEGER,       -- Barometric altitude (feet)
    altitude_geom   INTEGER,       -- Geometric altitude (feet)
    ground_speed    DOUBLE PRECISION,
    track           DOUBLE PRECISION,
    vertical_rate   INTEGER,       -- ft/min
    squawk          VARCHAR(4),
```

```
    on_ground       BOOLEAN,
    source          VARCHAR(20)    -- 'ADS-B', 'MLAT', etc.
);
```

The separation of static and dynamic data follows a standard time-series pattern. The `aircraft` table is updated infrequently (only when new aircraft are detected or metadata changes), while the `aircraft_positions` table receives the bulk of insertions.

6.2 Time-Series Data and Partitioning

For production deployments, the `aircraft_positions` table is partitioned by month to maintain query performance as data volume grows:

```
 CREATE TABLE aircraft_positions (
    -- columns as above
) PARTITION BY RANGE (timestamp);

CREATE TABLE aircraft_positions_y2026m04
    PARTITION OF aircraft_positions
    FOR VALUES FROM ('2026-04-01') TO ('2026-05-01');

CREATE TABLE aircraft_positions_y2026m05
    PARTITION OF aircraft_positions
    FOR VALUES FROM ('2026-05-01') TO ('2026-06-01');
```

Monthly partitions keep each table at a manageable size. With approximately 50,000–100,000 position records per hour, a monthly partition contains roughly 36–72 million rows—well within PostgreSQL's comfortable operating range. Older partitions can be dropped or moved to cold storage without affecting active queries.

6.3 Indexing Strategy

Two composite indexes support the primary query patterns:

```
 -- Time-ordered queries (live map, recent history)
CREATE INDEX idx_positions_timestamp
```

```
    ON aircraft_positions (timestamp DESC);

-- Per-aircraft history (trail visualization, aircraft details)
CREATE INDEX idx_positions_aircraft_time
    ON aircraft_positions (aircraft_id, timestamp DESC);
```

The descending sort order on `timestamp` optimizes the common pattern of querying recent data first. The composite index on `(aircraft_id, timestamp)` enables efficient retrieval of an individual aircraft's position history without scanning the entire table.

6.4 Data Retention

Storage growth is managed through a tiered retention policy:

Data Type	Retention	Approximate Size
Raw positions	30 days	~4 GB/month
Aggregated statistics	1 year	~100 MB/year
Aircraft metadata	Permanent	~50 MB total

A nightly cron job deletes raw positions older than 30 days:

```
DELETE FROM aircraft_positions
WHERE timestamp < NOW() - INTERVAL '30 days';
```

For partitioned tables, retention is enforced by detaching and dropping entire partitions, which is far more efficient than row-by-row deletion:

```
ALTER TABLE aircraft_positions DETACH PARTITION aircraft_positions_y2026m03;
DROP TABLE aircraft_positions_y2026m03;
```

7. Web Interface and API

7.1 Flask API Endpoints

The Flask API provides four primary endpoints for accessing aircraft data:

Endpoint	Method	Description
`/api/aircraft/live`	GET	All aircraft seen in the last 60 seconds
`/api/aircraft/history`	GET	Historical positions with time and ICAO filters
`/api/aircraft/<icao>`	GET	Aircraft details and last 100 positions
`/api/aircraft/stats`	GET	Aggregate statistics (hourly, daily, total)

The live endpoint returns data from the Redis cache, avoiding PostgreSQL queries entirely:

```
 @app.route('/api/aircraft/live')
def live_aircraft():
    keys = redis.keys('aircraft:*')
    aircraft = []
    for key in keys:
        data = redis.hgetall(key)
        aircraft.append({
            'icao_hex': data['icao_hex'],
            'callsign': data.get('callsign'),
            'latitude': float(data['latitude']),
            'longitude': float(data['longitude']),
            'altitude_baro': int(data.get('alt', 0)),
            'ground_speed': float(data.get('speed', 0)),
            'track': float(data.get('track', 0)),
            'vertical_rate': int(data.get('vert_rate', 0)),
        })
    return jsonify({
```

```
        'timestamp': datetime.utcnow().isoformat() + 'Z',
        'count': len(aircraft),
        'aircraft': aircraft
    })
```

An internal bulk-update endpoint receives decoded aircraft data from dump1090 and persists it to both PostgreSQL and Redis:

```
@app.route('/api/aircraft/bulk', methods=['POST'])
def bulk_update():
    data = request.json
    for ac in data.get('aircraft', []):
        # Update Redis cache (TTL 60s)
        redis.hset(f"aircraft:{ac['hex']}", mapping={
            'icao_hex': ac['hex'],
            'callsign': ac.get('flight', '').strip(),
            'latitude': ac.get('lat', 0),
            'longitude': ac.get('lon', 0),
            'alt': ac.get('altitude', 0),
            'speed': ac.get('speed', 0),
            'track': ac.get('track', 0),
            'vert_rate': ac.get('vert_rate', 0),
        })
        redis.expire(f"aircraft:{ac['hex']}", 60)

        # Insert position into PostgreSQL
        insert_position(ac)
    return jsonify({'status': 'ok'})
```

7.2 Real-Time Updates

The web interface implements two update strategies:

- **Polling mode:** The JavaScript client requests `/api/aircraft/live` at 1-second intervals using `fetch()`. Each response includes a timestamp and the complete set of visible aircraft.

- **Server-Sent Events (SSE):** For lower-latency updates, an optional SSE endpoint pushes position updates to connected clients as they arrive from dump1090, eliminating the 1-second polling interval.

Position deduplication is performed client-side: the JavaScript runtime tracks the last-known position for each aircraft and only updates the map when the position or velocity has changed beyond a threshold, reducing unnecessary DOM operations.

7.3 Map Visualization

The live map uses Leaflet.js with OpenStreetMap tiles. Aircraft markers are rendered as custom SVG icons colored by altitude using a gradient from green (low altitude) through yellow to red (high altitude). Click handlers display a popup with aircraft details, and polylines render position trails from the history endpoint.

The map supports several user interactions:

- Altitude range filter (slider)
- Aircraft type filter (dropdown)
- Speed filter
- Trail toggle (show/hide recent path)
- Historical playback (time slider with animated replay)

8. Performance and Coverage

8.1 Reception Range

The reception range of a single RTL-SDR receiver depends on antenna quality, installation height, local terrain, and RF environment. Typical performance for our installation (outdoor antenna at 10 m elevation, rural-suburban environment):

Aircraft Altitude	Typical Range	Notes

FL350 (35,000 ft)	250–350 km	Upper limit of radio horizon
FL250 (25,000 ft)	200–280 km	Major airway traffic
FL100 (10,000 ft)	120–180 km	Terminal area traffic
3,000 ft AGL	50–80 km	Low-level approaches, VFR
Surface	10–30 km	Ground vehicles, taxiway

Range at the upper altitudes is typically limited by the radio horizon rather than receiver sensitivity. At lower altitudes, terrain shielding and reduced transmit antenna gain (bottom-mounted antennas radiate primarily downward) are the limiting factors.

8.2 Message Rates and Throughput

Message reception rates vary with traffic density and receiver performance. Measured statistics for the production deployment:

Metric	Observed Value
Unique aircraft/hour	200–500
Unique aircraft/24h	800–1,200
Position reports/hour	50,000–100,000
Position reports/24h	1.2–2.4 million
Database growth	~1 GB/week (raw positions)
Messages decoded/hour	200,000–500,000 (all DF types)
CRC failure rate	15–30% (varies with conditions)

The relatively high CRC failure rate (15–30%) is expected: many received messages are at the sensitivity limit of the receiver, and weak signals

frequently suffer bit errors. Only messages passing CRC are accepted, ensuring data integrity at the cost of reduced message yield.

8.3 Multi-Feed Aggregation

The system integrates with the OpenSky Network, a collaborative ADS-B data collection project that aggregates data from hundreds of receivers worldwide [11]. This integration provides two key benefits:

- **Coverage extension:** OpenSky data fills gaps in local reception, particularly for aircraft at low altitudes or beyond the local radio horizon. The combined feed shows approximately 5,000+ aircraft over the continental United States at any given time.
- **Metadata enrichment:** OpenSky's aircraft database provides registration, type, and operator information that may not be available from ADS-B messages alone.

Data from OpenSky is tagged with `source: "opensky"` in the database, distinguishing it from locally received data (`source: "ADS-B"`). The system deconflicts overlapping data by preferring locally received positions (which have lower latency) over OpenSky data (which may lag by 10–30 seconds due to network transit and aggregation).

9. Related Work

The ADS-B receiving ecosystem includes several categories of systems, each with distinct design goals and capabilities.

FlightRadar24 is the largest commercial ADS-B aggregation service, operating a network of over 30,000 receivers worldwide [12]. It provides global coverage with near-real-time updates, supplemented by MLAT (multilateration) for aircraft not transmitting ADS-B. FlightRadar24's business model relies on providing free receiver hardware to volunteers who contribute data. The service is closed-source, and its data is proprietary. Our system differs fundamentally: it is fully open-source,

operates from a single receiver (with optional OpenSky supplementation), and provides raw data access through a public API.

OpenSky Network is an academic research project that aggregates ADS-B data from volunteer receivers for the purpose of air traffic surveillance research [11]. It stores historical data in a Hadoop cluster and provides access through a REST API and Impala queries. OpenSky's emphasis is on data collection for research rather than real-time tracking. Our system complements OpenSky by consuming its data for local enrichment while contributing locally received data back to the network.

dump1090 and its variants are the software foundation for most hobbyist and research ADS-B receivers. The original `dump1090` by Malcolm Robb provided basic Mode S decoding. The `dump1090-mutability` fork added network output, JSON APIs, and a built-in web map. FlightAware's `piaware` builds on dump1090 to feed the FlightAware commercial network. The `readsb` project by Mictronic offers a modernized, higher-performance alternative written in C with support for multi-receiver aggregation and built-in MLAT [8]. Our system uses dump1090-mutability as the primary decoder with readsb as an alternative for multi-receiver deployments.

ADSBExchange operates a receiver network similar to FlightRadar24 but with a focus on uncensored data: it does not filter military or government aircraft at the request of operators, making it popular among aviation enthusiasts [13]. Its data is available through a free API and a JSON feed.

Academic research on ADS-B has focused on security vulnerabilities (spoofing and injection attacks) [14], multilateration for non-ADS-B aircraft [15], and large-scale data analysis using OpenSky's historical dataset [16]. Our work differs in its emphasis on the complete system design from RF reception through web delivery, providing a practical reference for building ADS-B receiving stations.

10. Conclusion and Future Work

We have presented a complete ADS-B aircraft tracking system built from low-cost RTL-SDR hardware and open-source software. The system demonstrates that commodity SDR receivers, combined with well-designed decoding and storage pipelines, can provide real-time air traffic surveillance with coverage extending 200–350 km from a single receiving station. The PostgreSQL/Redis dual-store architecture achieves low-latency live data access while maintaining efficient long-term storage, and the Flask API provides a clean interface for both web visualization and programmatic data access.

Several directions for future work present themselves:

Multi-receiver network. Deploying additional receivers at geographically diverse locations would extend coverage and enable multilateration (MLAT) of non-ADS-B aircraft. MLAT computes aircraft position from the time difference of arrival (TDOA) of Mode S messages at multiple receivers, providing surveillance of aircraft that only have Mode S transponders without ADS-B Out capability. A network of 3–5 receivers spaced 50–100 km apart could provide MLAT coverage over a significant area with position accuracy on the order of 100–500 m [15].

Machine learning–based anomaly detection. The large volume of historical position data enables training of anomaly detection models that identify unusual flight patterns. Potential applications include detection of emergency situations (unexpected altitude changes, deviations from filed routes), airspace violations, and transponder malfunctions. Approaches include autoencoder-based models that learn normal flight patterns and flag deviations, as well as supervised classifiers trained on labeled incident data [16].

1090 MHz spectrum monitoring. Beyond ADS-B decoding, the SDR receiver can monitor the full 1090 MHz spectrum for anomalous transmissions that may indicate spoofing attacks or interference. The growing body of research on ADS-B security vulnerabilities [14] motivates

continuous monitoring for messages with inconsistent positions, impossible velocities, or spoofed ICAO addresses.

UAT (978 MHz) reception. Adding a second SDR receiver tuned to 978 MHz would capture ADS-B messages on the UAT link, which is used primarily by general aviation aircraft in the United States. UAT messages use a different format (DO-282B) but provide the same position and identity information, increasing the completeness of the surveillance picture at lower altitudes.

GPU-accelerated decoding. Modern SDR platforms such as the Airspy R2 (20-bit ADC, 10 Msps) and the HackRF One (8-bit ADC, 20 Msps) provide wider bandwidth and higher dynamic range than the RTL-SDR. GPU-accelerated decoders could process the wider I/Q stream in real time, enabling simultaneous reception of 1090 MHz and 978 MHz from a single wideband receiver.

The ADS-B ecosystem continues to evolve with increasing equipage rates, new message types, and growing interest in open surveillance data. The system described here provides a flexible, extensible platform for real-time aircraft tracking that can grow with these developments.

References

[1] Federal Aviation Administration, “Automatic Dependent Surveillance–Broadcast (ADS-B) Out,” *14 CFR § 91.225*, 2010.

[2] International Civil Aviation Organization, “Technical Provisions for Mode S Services and Extended Squitter,” *ICAO Doc 9871*, 2nd ed., 2012.

[3] M. W. L. G. A. V. N. Stevens, “Primary and Secondary Radar Surveillance of Air Traffic,” *Proc. IEE*, vol. 118, no. 6, pp. 731–740, 1971.

[4] EUROCONTROL, “ADS-B Implementation Guide,” *EUROCONTROL-GUID-130*, 2021.

[5] K. B. M. C. J. A. Leong, “The RTL-SDR: A Low-Cost Software-Defined Radio for Experimentation,” *GNU Radio Conference*, 2016.

[6] RTCA, “Minimum Operational Performance Standards for 1090 MHz Extended Squitter ADS-B and TIS-B,” *DO-260B*, 2009.

[7] ICAO, “Aeronautical Telecommunications, Volume IV: Surveillance and Collision Avoidance Systems,” *Annex 10*, 6th ed., 2014.

[8] M. R. E. H. Schäfer, “readsb: A High-Performance ADS-B Decoder,” *GitHub Repository*, 2023. Available: https://github.com/wiedehopf/readsb

[9] J. G. Peterson, “CRC-24 Error Detection for Mode S Extended Squitter,” *MIT Lincoln Laboratory*, Tech. Rep. ATC-350, 2010.

[10] ICAO, “Compact Position Reporting (CPR) Algorithm,” *ICAO Doc 9871 Appendix C*, 2012.

[11] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, “Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research,” *Proc. IPSN*, pp. 83–94, 2014.

[12] FlightRadar24, “About FlightRadar24,” 2025. Available: https://www.flightradar24.com/about

[13] ADSBExchange, “ADS-B Exchange—Uncensored ADS-B Data,” 2025. Available: https://www.adsbexchange.com/

[14] M. Strohmeier, V. Lenders, and I. Martinovic, “On the Security of ADS-B: The Case for Realistic Threat Models,” *Proc. WiSec*, pp. 167–172, 2015.

[15] M. Schäfer, C. Karsch, and M. Strohmeier, “On the Feasibility of Ground-Based ADS-B Multilateration,” *Proc. AIAA Infotech@Aerospace*, 2016.

[16] M. Schäfer, M. Strohmeier, and I. Martinovic, “OpenSky Report 2020: Analysing COVID-19 Impact on Air Traffic Worldwide,” *Proc. AIAA Scitech Forum*, 2021.