ADSB FLIGHT TRACKING SYSTEM

This system provides two components for processing real-time flight data from an installed ADSB tracker (such as one with a USB dongle and ADSB antenna, which often run on a Raspberry PI).

The **“status”** component is an updated webpage that runs on the tracker to provide a cute radar-like display of the tracker’s real-time flight data with a menu of links to the ADSB tracker’s local and online services (supporting FlightAware, FlightRadar24, ADSB-Exchange, AirNav.radar and OpenSky). The radar refreshes on the client side and illustrates local airports (and their ATZ) and the location of aircraft and their recent trailing trajectory.

The **“monitor”** component is a standalone JavaScript (Node.js) based backend system that processes the tracker’s real-time flight data to generate analysis, insights and alerts which can be delivered to console as text or as structured messages to an MQTT broker. Alerts can trigger for flights of special types (e.g. military), specific names (e.g. royal or VIP flights), proximate (within a specified radius and height), landing or lifting (at an identified airports, or not), projected to pass overhead, have weather or operating anomalies (such as unusual speeds for the type of aircraft), or match other attributes (squawk codes, flight numbers, etc). The criteria and thresholds triggering these alerts are reasonably configurable.

The top level directory of the repository also contains an **“airports” directory**, with Node.js scripts to download and generate airports data from the online data at “https://github.com/davidmegginson/ourairports-data”, and a **“content” directory**, with data files including GB and SE squawk codes and the AIP entries (for GB) that they were extracted from. The airport data is used by both components, and the squawks are used by the “monitor” component.

The system is intended to provide an ADSB tracking installation with a capability to investigate and understand aircraft operating in the local vicinity. The author developed the system primarily for use at his property in remote Sweden, where it is installed and subject to light traffic: the alerts provide awareness about aircraft approaching overhead and operating locally. The author also lives in Central London where it is installed and subject to a significant volume and diversity of traffic. Both systems are rich environments for the system’s ongoing development.

Please note the LICENSE: “Attribution-NonCommercial-ShareAlike 4.0”.

COMMON

INSTALLATION

Both components should be retrieved by cloning the **“tracking-adsb” repository** into the **“/opt/tracking-adsb” directory**.

AIRPORTS DIRECTORY - AIRPORTS DATABASE

Both components require a narrow selection of the airports from a larger database. This can be accomplished with the tools provided in the **“airports” directory**. The **“airports-data-converter.js” script** is used to download, join and transform the larger database’s CSV files into a single “airports-data.json” file, after which

the **“airports-data-generator.js”** script is used to filter out a subset of airports within a specified radius of the station’s co-ordinates to generate a “airports-data.{hostname}.json” file which is used by both the “status” and the “monitor” components.

```
root@adsb:/opt/tracking-adsb/airports# ls -l
total 12
-rwxr-xr-x 1 root root 8165 Jun  8 13:04 airports-data-converter.js
-rwxr-xr-x 1 root root 4040 Jun  8 13:05 airports-data-generator.js

root@adsb:/opt/tracking-adsb/airports# npm install csv-parse
...
Run `npm audit` for details.

root@adsb:/opt/tracking-adsb/airports# ./airports-data-converter.js
Local file not found: airports.csv
Fetching from https://davidmegginson.github.io/ourairports-data/airports.csv...
Successfully fetched airports.csv from remote URL
Parsed airports.csv: 83121 records
Local file not found: runways.csv
Fetching from https://davidmegginson.github.io/ourairports-data/runways.csv...
Successfully fetched runways.csv from remote URL
Parsed runways.csv: 46863 records
Local file not found: airport-frequencies.csv
Fetching from https://davidmegginson.github.io/ourairports-data/airport-frequencies.csv...
Successfully fetched airport-frequencies.csv from remote URL
Parsed airport-frequencies.csv: 30129 records

Processing complete:
- Airports: 83121
- Runways: 46863
- Frequencies: 30129
- Airports with runways: 39935
- Airports with frequencies: 11063

Output written to airports-data.json

root@adsb:/opt/tracking-adsb/airports# ls -l
total 81672
-rwxr-xr-x 1 root root     8165 Jun  8 13:04 airports-data-converter.js
-rwxr-xr-x 1 root root     4040 Jun  8 13:05 airports-data-generator.js
-rw-r--r-- 1 root root 83618152 Jun  8 20:43 airports-data.json
```

```
root@adsb:/opt/tracking-adsb/airports# ./airports-data-generator.js 51.50092998192453 -0.20671121337722095 450nm
Searching for airports within 450nm (833.40km) of 51.50092998192453, -0.20671121337722095
Found 4596 airports out of 83121 total airports
Output written to airports-data.adsb.js

root@adsb:/opt/tracking-adsb/airports# ls -l
total 86936
-rwxr-xr-x 1 root root     8165 Jun  8 13:04 airports-data-converter.js
-rwxr-xr-x 1 root root     4040 Jun  8 13:05 airports-data-generator.js
-rw-r--r-- 1 root root  5388532 Jun  8 20:43 airports-data.adsb.js
-rw-r--r-- 1 root root 83618152 Jun  8 20:43 airports-data.json

root@adsb:/opt/tracking-adsb/airports# jq '.EGLW' airports-data.json
{
  "id": 43211,
  "ident": "EGLW",
  "type": "heliport",
  "name": "London Heliport",
  "latitude_deg": 51.46972274779999,
  "longitude_deg": -0.179444000125,
  "elevation_ft": 18,
  "continent": "EU",
  "iso_country": "GB",
  "iso_region": "GB-ENG",
  "municipality": "London",
  "scheduled_service": "no",
  "icao_code": "",
  "iata_code": "",
  "gps_code": "EGLW",
  "local_code": "",
  "home_link": "http://www.londonheliport.co.uk/",
  "wikipedia_link": "https://en.wikipedia.org/wiki/London_Heliport",
  "keywords": "",
  "runways": [
    {
      "id": 307325,
      "airport_ref": 43211,
      "airport_ident": "EGLW",
      "length_ft": 114,
      "width_ft": 52,
      "surface": "Concrete",
      "lighted": true,
```

```
      "closed": false,
      "le_ident": "03",
      "le_latitude_deg": null,
      "le_longitude_deg": null,
      "le_elevation_ft": 18,
      "le_heading_degT": 24,
      "le_displaced_threshold_ft": null,
      "he_ident": "21",
      "he_latitude_deg": null,
      "he_longitude_deg": null,
      "he_elevation_ft": 18,
      "he_heading_degT": 204,
      "he_displaced_threshold_ft": null
    }
  ],
  "frequencies": [
    {
      "id": 71950,
      "airport_ref": 43211,
      "airport_ident": "EGLW",
      "type": "RAD/APP",
      "description": "London Heliport RAD/APP",
      "frequency_mhz": 122.9
    },
    {
      "id": 71949,
      "airport_ref": 43211,
      "airport_ident": "EGLW",
      "type": "TWR",
      "description": "London Heliport TWR",
      "frequency_mhz": 122.9
    }
  ]
}
```


CONTENT DIRECTORY - SQUAWKS DATABASE

The “monitor” component uses a squawks database for certain analysis purposes, which is stored in the **“content” directory**.

The squawk database for the United Kingdom (GB) was extracted from a recent (May 2025) release of the AIP, the source of which (in HTML format) is stored in the directory. The squawks database is stored in the **“squawk-codes-gb.js” file**. The codes and the descriptions and details are directory obtained from the AIP, whereas the “type” designation is determined by AI.

There is a squawk database for Sweden (SE), but it is rather lightweight due to the Swedish AIP having a very bare AIP.

Where an installation tracks aircraft over multiple flight regions, the region specific squawks would need to be combined. The system does not yet have the ability to select a geographically relevant squawk based upon the aircraft’s coordinates: this would not be difficult (though not trivial to add).

```
root@adsb:/opt/tracking-adsb/content# ls -l
total 352
-rw-r--r-- 1 root root 298685 May 29 14:20 EG-ENR-1.6.2-en-GB.html
-rw-r--r-- 1 root root  57273 May 29 23:11 squawk-codes-gb.js
-rw-r--r-- 1 root root   1008 May 29 23:11 squawk-codes-se.js

root@adsb:/opt/tracking-adsb/content# head -20 squawk-codes-gb.js

// UK AIP ENR 1.6
// https://nats-uk.ead-it.com/cms-nats/opencms/en/Publications/AIP/

module.exports = {
  codes: [
    // 0xxx Series
    { begin: '0000', description: ['SSR data unreliable'], type: 'special' },
    { begin: '0001', description: ['Height Monitoring Unit'], type: 'ground', details: ['Ground based transponder
equipment'] },
    { begin: '0002', description: ['Ground Transponder Testing'], type: 'ground', details: ['Refer to ENR 1.6,
paragraph 2.2.3'] },
    { begin: '0003', description: ['Surrey/Sussex HEMS (HLE60)'], type: 'hems' },
    { begin: '0004', end: '0005', description: ['Scottish Non-standard Flights'], type: 'assigned' },
    { begin: '0006', description: ['British Transport Police ASU'], type: 'police' },
    { begin: '0007', description: ['Off-shore Safety Area (OSA) Conspicuity'], type: 'conspicuity', details: ['See
note 9'] },
    { begin: '0010', description: ['Birmingham Airport Frequency Monitoring'], type: 'monitoring', details: ['This
code may be used when flying in the vicinity of Birmingham within the area defined in EGBB AD 2.22 FLIGHT
PROCEDURES'] },
    { begin: '0011', description: ['Bournemouth Control Zone Frequency Monitoring'], type: 'monitoring', details:
['Aircraft operating in the vicinity of Bournemouth Control Zone and monitoring Bournemouth Radar Frequency', 'The
delineation with Solent is west of a line between Stoney Cross VRP and Hurst Castle VRP'] },
    { begin: '0012', description: ['London City/London Heathrow Frequency Monitoring'], type: 'monitoring',
```

```
details: ['This code may be used when flying in the vicinity of London City and London Heathrow and operating
outside of the London CTR/London City CTR/London CTA but within the area defined in EGLC AD 2.22 FLIGHT PROCEDURES
and EGLL AD 2.22 FLIGHT PROCEDURES'] },
    { begin: '0013', description: ['London Luton Frequency Monitoring'], type: 'monitoring', details: ['This code
may be used when flying in the vicinity of London Luton and operating outside of the London Luton CTR/ London
Luton CTA but within the area defined in EGGW AD 2.22 FLIGHT PROCEDURES'] },
    { begin: '0014', description: ['Kent Air Ambulance (HLE21)'], type: 'hems' },
    { begin: '0015', description: ['Essex Air Ambulance (HLE07)'], type: 'hems' },
```

STATUS COMPONENT

For clarity, this is not intended to be any sort of replacement for the existing detailed tracking pages provided by the ADSB feeding services either on the tracker device itself, or online. This “status” page is only designed to be some cute eye-candy with links to services: a glorified bookmark list, if you must.

The **‘lighttpd’ webserver** needs to be installed with the **“php-cgi” module** (needed for the “radar-data.php” script to act as a server side proxy to fetch data from the running FlightRadar24 feeder). It is only designed to operate on the ADSB tracking device itself. Node.js command line is optional and only needed to run the airports scripts as a one time installation activity: the scripts can be run on another machine if desired.

The “/var/www/html” directory (as used by lighttpd) must be symlinked to “/opt/tracking-adsb/status” (the “status” directory).

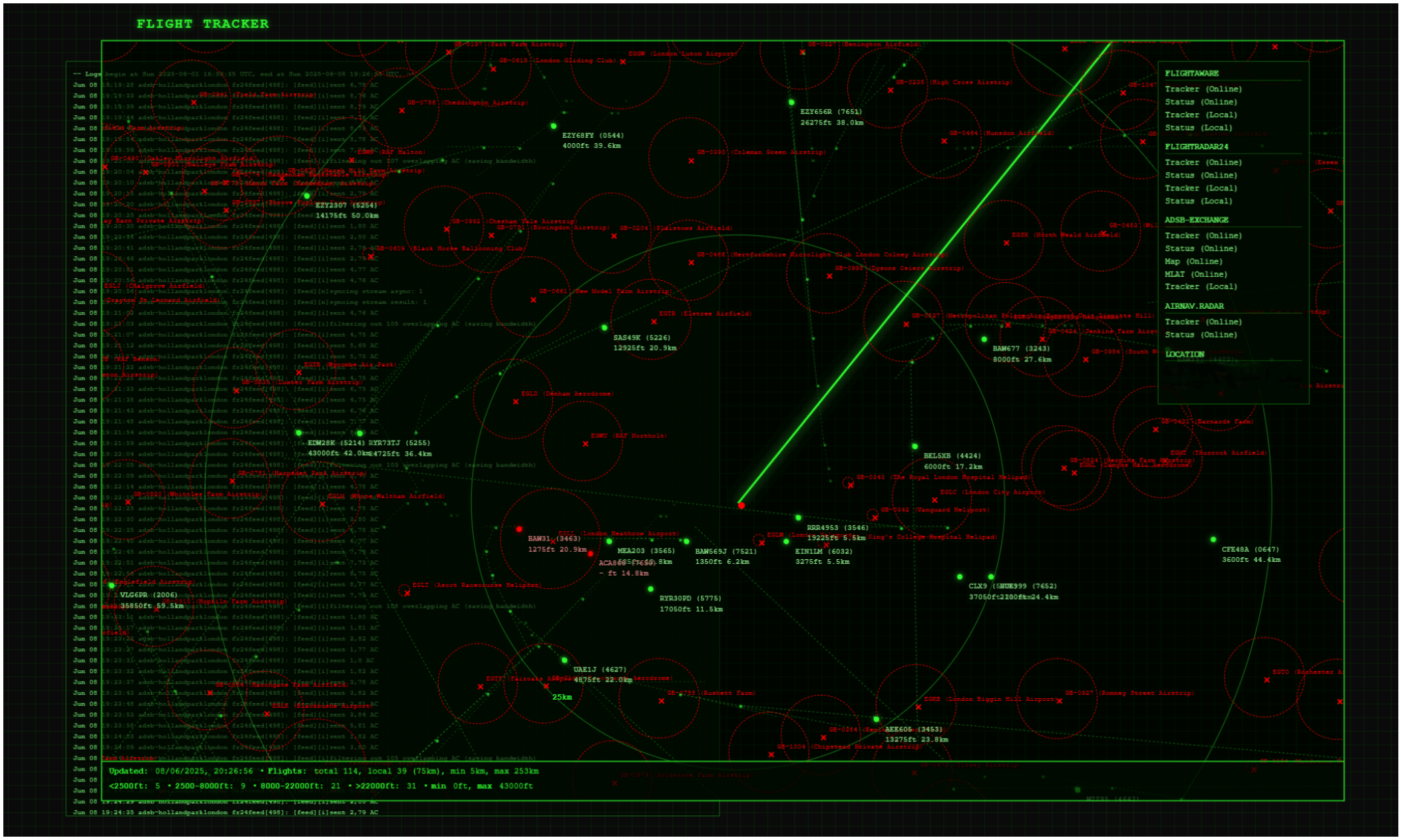
In the “status” directory, the **“config.js” file** must be copied to **“config.adsb.js”** and edited. The station’s location data (address, latitude, longitude and altitude) must be specified in the location section. The relative position of the radar on the screen can be configured in the radar section, if desired. The flight feeder service identifiers (e.g. station identifiers) for use in the menu of links, should be specified in the “services” section: services can be commented out or removed and they will not show in the menu on the radar display. The airports database (as generated in the “airports” directory) should be stored into the directory as **“airports-data.adsb.js”**. Both of the “config.adsb.js” and “airports-data.adsb.js” files are loaded from the server into the browser client side.

```
root@adsb:/opt/tracking-adsb/status# ls -l
total 6188
-rw-r--r-- 1 root root 6223788 Jun  8 19:23 airports-data.adsb.js
-rw-r--r-- 1 root root     653 Jun  8 19:23 config.adsb.js
-rw-r--r-- 1 root root   15406 May 24 07:19 favicon.ico
-rw-r--r-- 1 root root   15406 May 24 07:19 favicon.png
-rw-r--r-- 1 root root    1076 May 24 07:19 favicon.svg
-rw-r--r-- 1 root root    2539 May 24 07:19 flightaware.gif
-rw-r--r-- 1 root root    3242 May 24 07:19 flightaware.html
-rw-r--r-- 1 root root   16355 May 24 07:19 flightaware.js
drwxr-xr-x 2 root root    4096 May 24 07:19 flightaware.lib
-rw-r--r-- 1 root root    3112 May 24 07:21 index.html
```

```
-rw-r--r-- 1 root root     843 May 24 07:19 radar-data.php
-rw-r--r-- 1 root root   20727 Jun  8 19:22 radar-script.js
-rw-r--r-- 1 root root    6825 May 24 07:19 radar-styles.css

root@adsb:/opt/tracking-adsb/status# cat config.adsb.js
const config = {
    location :{
        address: 'Holland Park, London',
        lat: 51.50092998192453,
        lon: -0.20671121337722095,
        alt: 36,
    },
    radar: {
        bottom: -70,
        right: -20
    },
    services: {
        flightaware: {
            site: 160771,
        },
        flightradar24: {
            id: 33663,
        },
        adsbexchange: {
            uid: '4UYG4bVwptz0',
            name: 'hollandparklondon',
            region: 20,
        },
        airnavradar: {
            station: 'EXTRPI014714',
        },
        //opensky: {
            //id: 85111,
        //},
    },
};
```

The source code, also delivered to the client side, is contained in the **“radar-script.js” file**. The page’s style information is contained in the **“radar-styles.css” file** and the shell of the web page (as its document is otherwise filled out by the radar script) is contained in **the “index.html” file**. There are also favicons. Note that the directory also contains the flightware default webpage and assets, which have all been renamed to begin with “flightaware”.


FLIGHT TRACKER
FLIGHTAWARE
Tracker (Online)
Status (Online)
Tracker (Local)
Status (Local)
FLIGHTRADAR24
Tracker (Online)
Status (Online)
Tracker (Local)
Status (Local)
ADSB-EXCHANGE
Tracker (Online)
Status (Online)
Map (Online)
MLAT (Online)
Tracker (Local)
AIRNAV.RADAR
Tracker (Online)
Status (Online)
LOCATION
EZY68FY (0544)
4000ft 39.6km
EZY656R (7651)
26275ft 38.0km
EZY2307 (5254)
14175ft 50.0km
SAS49K (5226)
12925ft 20.9km
BAW677 (3243)
8000ft 27.6km
EDW28K (5214) RYR73TJ (5255)
43000ft 42.0km 24725ft 36.4km
BEL5XB (4424)
6000ft 17.2km
BRR4953 (3546)
19225ft 5.5km
BAW31 (3463)
1275ft 20.9km
MEA203 (3565)
BAW569J (7521)
1350ft 6.2km
EIN1LM (6032)
3275ft 5.5km
CFE48A (0647)
3600ft 44.4km
CLX9 (SHUE999 (7652)
37050ft2100ft∞24.4km
RYR30PD (5775)
17050ft 11.5km
VLG6PR (2006)
35850ft 59.5km
UAE1J (4627)
AEE605 (3453)
13275ft 23.8km
25km
Updated: 08/06/2025, 20:26:56 • Flights: total 114, local 39 (75km), min 5km, max 253km
<2500ft: 5 • 2500-8000ft: 9 • 8000-22000ft: 21 • >22000ft: 31 • min 0ft, max 43000ft

MONITOR COMPONENT

INSTALLATION

The “monitor” component does not need to be installed on the ADSB tracking device, but for colocation, it’s probably worth doing so if the device has sufficient processing capability (which you can determine from long-running load average information). The component requires access to the ADSB tracker to read flight data every 30 seconds (and it uses the ADSB-exchange feeder for this, which must be installed and active on the ADSB tracking device), and (if desired) access to an MQTT broker to publish alerts: thus far, only the ‘mosquitto’ MQTT broker has been used.

The component requires Node.js (tested with v18.20.8 at minimum) and NPM packages to be installed in its operating directory (the **“monitor” directory**): a **“package.json” file** is provided for this purpose (so you can just run “npm install”). The component should be configured and operated manually before it is run automated: this will help understand its functioning and to tune the thresholds and criteria (according to interest and local circumstances).

The component is manually operated by executing the **“monitor” script** with, if desired, the configuration filename as an argument, otherwise the default filename is used. Considerable output is provided to both stderr (such as operating processes and execution) and stdout (operating results: flight alerts and status information) on each 30 seconds cycle.

Automatic operation can be achieved using the systemd service file (the **“monitor.service” file**, which can be installed using “make install” on the provided Makefile). It assumes that the system is installed in “/opt/tracking-adsb”. The operating output can then be viewed through the system journal.

CONFIGURATION

The default configuration (in the “monitor” directory), is in the **“config.js” file**, which should be copied to **“config.{hostname}.js”** and edited. It is necessary to specify the station's location and elevation, the flight data url (if not running on the same machine), the specific airport and squawk code databases (as appropriate to the geographic location of the station), and the MQTT server (if it is not running on the localhost). Further configuration exists for virtually all of the modules and functions described herein.

OPERATION

At startup, the various modules are initialized. Airport data is read and pre-processed into a spatial grid for fast lookup. This data contains airport names, ICAO/IATA codes, and positioning (latitude, longitude, elevation) information, plus runways and frequencies. Further data, including squawk codes, is loaded and indexed for optimal lookup.

The system operates on a 30-second cycle to fetch flight data from the ADSB-Exchange feeder. This data is then processed, filtered, and used to generate structured alerts and statistics that are delivered to both an MQTT broker and the console.

```
root@adsb:/opt/tracking-adsb/monitor# ./monitor
***
```

```
config: ./config.workshop.js
config: {
  location: {
    address: 'Holland Park, London',
    lat: 51.50092998192453,
    lon: -0.20671121337722095,
    alt: 36
  },
  range_max: 300,
  flights: {
    link: 'http://adsb.local/adsbx/data/aircraft.json',
    exclude: [ 'TEST1234' ],
    debug: true
  },
  airports: {
    spatial_indexing: true,
    source: '../airports/airports-data.workshop.js'
  },
  further: { squawks: { file: 'squawk-codes-gb.js' } },
  filters: {
    emergency: {},
    military: {},
    airport: { priorities: [Array] },
    squawks: {},
    anomaly: {},
    weather: { temperatureInversion: [Object] },
    vicinity: { distance: 10, altitude: 10000 },
    overhead: { radius: 1, time: 1800, distance: 20, altitude: 20000 },
    landing: { radius: 10, distance: 100, altitude: 2500 },
    lifting: { altitude: 2500, radius: 9.26, minClimbRate: 300 },
    loitering: {},
    airprox: {
      horizontalThreshold: 1,
      verticalThreshold: 1000,
      airportExclusionRadius: 5
    },
    specific: { flights: [Array] }
  },
  alerts: { warn_suppress: { weather: true } },
  deliver: {
    mqtt: {
      enabled: true,
```

```
      server: 'mqtt://localhost:1883',
      clientId: 'adsb-monitor',
      publishTopics: [Object],
      debug: false
    }
  },
  debug: { sorting: false }
}
flights: mappings[hex/flight]: loaded from .monitor-hextoflight.cache (2960 entries)
airports: spatial index built with 686 grid cells [tot=3965, cnt=3965, avg=5.8, max=29] (cache: limit=1000,
trim=100)
airports: 3965 loaded; 2342 in range (300nm/556km); 2 in vicinity (10km): EGLW [London Heliport], GB-1092 [King's
College Hospital Helipad]
data [squawks]: codes: possible=4096, unique=4870, actual=6600, types: count=28
filters: 13 available, 13 enabled: emergency:1, airprox:1, landing:2, lifting:2, specific:3, squawks:3,
overhead:3, anomaly:4, loitering:4, vicinity:4, weather:5, airport:5, military:6
mqtt: connecting to 'mqtt://localhost:1883'
mqtt: loaded using 'server=mqtt://localhost:1883, client=adsb-monitor'
mqtt: connected
flights: mappings[hex/flight]: replace=12, updated=10, cleaned=0 (2962 entries)
2025-06-10T08:17:28.464Z ALERT/insert [squawks] CASTLE21 squawk 7031 anomaly: Military transponder code on
non-military callsign <A7: Rotorcraft> (14.9km NNW 1,250 ft)
2025-06-10T08:17:28.464Z ALERT/insert [squawks] TARTN29 squawk 6401 anomaly: Military transponder code on
non-military callsign <A5: Heavy (>300k lbs)> (185.2km NW FL140)
2025-06-10T08:17:28.464Z ALERT/insert [squawks] GECRM squawk 7004: Aerobatics and Display <A1: Light (<15.5k lbs)>
(67.8km WNW 3,800 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] BAW628 near EGLL [London Heathrow Airport] <A3: Large (75-300k
lbs)> (19.2km W 1,125 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] CASTLE21 near EGTR [Elstree Airfield] <A7: Rotorcraft> (14.9km NNW
1,250 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] EJU72BE near GB-0782 [East Haxted Farm Airfield] <A3: Large
(75-300k lbs)>
2025-06-10T08:17:28.464Z ALERT/insert [airport] IBE07SF near EGLL [London Heathrow Airport] <A3: Large (75-300k
lbs)> (11.6km W 875 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] BAW16HV near EGLL [London Heathrow Airport] <A3: Large (75-300k
lbs)> (14.9km W 300 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] KLM25Y near EGLC [London City Airport] <A3: Large (75-300k lbs)>
(23.2km E 1,400 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] [4079d9] near GB-0927 [Romney Street Airstrip] <A7: Rotorcraft>
(31.1km SE 1,500 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] GCEEU near GB-0781 [Harpsden Park Airstrip] (49.7km W 1,400 ft)
2025-06-10T08:17:28.464Z ALERT/insert [airport] EJU36ZE near GB-0282 [Newnham Grounds Airstrip] <A3: Large
```

```
(75-300k lbs)>
2025-06-10T08:17:28.464Z ALERT/insert [lifting] BAW628 climbing from EGLL [London Heathrow Airport] at 2944 ft/min
<A3: Large (75-300k lbs)> (19.2km W 1,125 ft)
2025-06-10T08:17:28.464Z ALERT/insert [vicinity] BAW5KY nearby, look SSE direction tracking WNW <A3: Large
(75-300k lbs)> (5.3km SSE 2,850 ft) [NOTICE]
2025-06-10T08:17:28.464Z ALERT/insert [vicinity] LOT1VZ nearby, look WSW direction tracking WNW <A3: Large
(75-300k lbs)> (6.4km WSW 1,900 ft) [NOTICE]
2025-06-10T08:17:28.464Z ALERT/insert [vicinity] BAW771C nearby, look SE direction tracking W <A3: Large (75-300k
lbs)> (7.8km SE 4,000 ft) [NOTICE]
2025-06-10T08:17:28.464Z ALERT/insert [overhead] KMM100A overhead descending at 576 ft/min, in about 3 minutes at
4,850 ft, look NE nearly overhead <A3: Large (75-300k lbs)> (14.8km SE 4,850 ft) [NOTICE]
2025-06-10T08:17:28.515Z STATUS/flights received=154, filtered=15, cached=154, stored=(tracked=154, entries=154,
avg=1, age=0s/0s), mappings=2962/1d16h22m59s
2025-06-10T08:17:28.515Z STATUS/alerts new=17, all=17: lifting=1, squawks=3, overhead=1, vicinity=3, airport=9
2025-06-10T08:17:28.515Z STATUS/system okay=1, error=0, empty=0, started=2025-06-10T08:17:28.211Z

...
```

… Flights

Flights are fetched by HTTP from a running instance of an ADSB-Exchange feeder. They are received in JSON format. Flights can be suppressed after being fetched (configurable) and the default configuration suppresses flights named "TEST1234".

The flight data is preprocessed to replace non-existent flight names with previously cached mappings from hex code to flight names, or if none such exists, then the textual representation of the hex code. These mappings are retained for up to 7 days (configurable) and cached on the filesystem to survive across restarts.

The flight data is cached to retain a history for use by the filtering modules, such as those that use trajectory data. This history is retained for up to 30 minutes (configurable) after the flight is no longer visible. This is not cached on the filesystem.

… Filters

Flights are first preprocessed to calculate basic metrics (such as distance from the station) and fill in missing data (such as altitude). This establishes a baseline dataset for filter analysis.

Each flight then passes through the filters described below. Filters can access: (1) current flight data and historical trajectory; (2) external databases (airports, squawk codes); (3) other flights in the airspace (for proximity detection).

Filters range from simple checks (emergency squawk codes) to complex trajectory analysis (holding pattern detection). When a filter matches, it returns: (1) an evaluation result (true/false); (2) associated data (e.g., for airprox: the other aircraft, separation distances, risk category); (3) a severity level (low/medium/high),

Matched flights are prioritized using a two-level sorting system: (1) filter priority: emergency has the highest priority; landing and lifting share equal priority; and (2) within-filter priority: criteria specific to that filter (e.g., airprox category A ranks higher than D).

Filters maintain state data across processing cycles to track patterns over time. Each filter can be: (1) disabled globally, (2) disabled for specific flights (by callsign or hex code), (3) configured with custom thresholds and parameters.

emergency	The emergency filter detects aircraft in emergency situations by checking for emergency status flags or emergency squawk codes (7500 for hijacking, 7600 for radio failure, 7700 for general emergency). When an aircraft broadcasts an emergency squawk or has its emergency flag set to anything other than 'none', the filter triggers an alert with high priority. This provides immediate notification of aircraft declaring emergencies in the monitored airspace.
military	The military filter identifies military aircraft by examining flight callsigns for known military prefixes (such as RCH, PLF, RRR, ASY, etc.) or patterns that match military flight naming conventions (four letters followed by two digits). The filter maintains a configurable list of military prefixes from various nations' air forces. When detected, these flights are flagged for special attention as they may involve military operations, training exercises, or transport missions.
specific	The specific filter tracks aircraft of special interest based on configurable pattern matching rules applied to flight numbers or aircraft categories. It can identify various types including government flights, VIP/royal flights, emergency services (air ambulances, police), test flights, aerial surveys, military transport, and special operations. Each pattern match includes a category and description, allowing customized monitoring of particular aircraft types based on operational requirements or local interests.
squawks	The squawks filter analyzes transponder codes against the database of assigned squawk codes and their meanings. Beyond simple code matching, it performs anomaly detection to identify misuse or unusual squawk usage, such as military codes on civilian aircraft, VFR codes at IFR altitudes, ground testing codes on airborne aircraft, or emergency codes without corresponding emergency flags. The filter assigns severity levels to anomalies and provides detailed descriptions of both the squawk meaning and any detected inconsistencies.
anomaly	The anomaly filter identifies unusual aircraft operational behaviors through pattern analysis of flight parameters. It detects multiple anomaly types including: high-speed at low altitude (e.g., >350 knots below 10,000ft), low-speed at high altitude, temperature anomalies compared to expected values, altitude oscillations (repeated climbs/descents), deviations from assigned altitudes, extreme vertical rates (>6000 ft/min), rapid vertical rate changes, and rapid speed changes. Each anomaly is assigned a severity level (low/medium/high) based on configurable thresholds. The filter uses trajectory history data to analyze patterns over time to identify erratic flight behaviors, potential equipment issues, or unusual maneuvering.
weather	The weather filter detects aircraft encountering or operating in challenging weather conditions by analyzing flight parameters for weather-related indicators. It identifies potential icing conditions based on temperature and altitude combinations, severe icing risks in supercooled large droplet conditions, turbulence through vertical rate variations, strong winds by comparing ground speed to true airspeed differences, and temperature inversions. Each condition is evaluated against configurable thresholds and assigned severity levels to help

	identify aircraft that may be experiencing weather-related operational challenges
airprox	The airprox filter detects potential aircraft proximity conflicts by analyzing the relative positions and trajectories of aircraft pairs. It calculates horizontal separation and vertical separation between aircraft, excluding those near airports. The filter computes closure rates and time to closest approach, then assigns risk categories A through D based on separation thresholds - with category A representing serious collision risk (under 0.25nm/500ft) and D representing undetermined risk. High closure rates (>400 knots) increase the severity assessment.
airport	The airport filter identifies aircraft operating near airports by checking their position against the database of airports. It determines when aircraft are within an airport's ATZ based on distance and altitude thresholds. The filter considers different ATZ dimensions for different airport types (heliports have smaller zones than major airports) and provides information about which specific airports the aircraft is near, enabling monitoring of airport approach and departure activity.
landing	The landing filter detects aircraft on approach to landing by analyzing descent rate, altitude, ground speed, and projected touchdown location. It calculates the estimated landing position based on current trajectory and determines if this intersects with a known airport. The filter distinguishes between approaches to recognized airports and potential off-airport landings, providing estimated time to touchdown and identifying the destination airport when possible.
lifting	The lifting filter identifies aircraft that have recently taken off by detecting sustained climb rates at low altitudes with appropriate ground speeds. It calculates a "lifting score" based on altitude, climb rate, and speed factors, then searches for nearby airports to identify the departure point. The filter estimates departure time and provides information about climb performance, helping track aircraft in their initial climb phase after takeoff.
loitering	The loitering filter detects aircraft engaged in holding patterns or persistent area operations by analyzing flight trajectory over time. It examines the bounding box of recent positions, looking for confined movement patterns including circling (based on heading changes), figure-8 or racetrack patterns (track reversals), and hovering (for rotorcraft). The filter calculates pattern confidence scores based on multiple factors and requires sustained patterns over several minutes to avoid false positives from normal maneuvering.
vicinity	The vicinity filter provides simple proximity alerts for aircraft within configured distance and altitude thresholds from the monitoring station. It triggers when aircraft are close enough to be of immediate local interest, providing basic information about the aircraft's position and direction relative to the station. This filter serves as a general awareness tool for nearby aircraft activity.
overhead	The overhead filter calculates when aircraft will pass directly overhead or near-overhead of the monitoring station. It computes the cross-track distance from the aircraft's current trajectory to determine the closest point of approach, estimates time until overhead passage, and calculates the altitude at that point. The filter provides observation guidance including approach direction and vertical angle, helping predict visible aircraft passages for spotting or monitoring purposes.

… Alerts

The results of the filters, run in each 30 second cycle, are a list of flights with positive evaluated outcomes and are used to generate alerts for onward delivery.

Alerts are “inserted” when a flight is positively evaluated for a filter, such as the emergency filter detecting the presence of an emergency squawk code. Flights may have multiple alerts, up to one alert per filter, but no flight can have more than one alert for the same filter.

An alert remains active (“inserted”) while continuing to have a positive evaluation generated by the filter, on each 30 second cycle.

An alert is then “removed” after 5 minutes of that positive evaluation no longer being true. This ensures that transient positive evaluations exist for a minimum period of time, and that alerts do not “flip flop” in and out of evaluation (this is effectively a hysteresis mechanism).

An alert consists of structured data and a human readable textual description and each alert is assigned a unique identifier. An alert is inserted with its full set of data and text, and then removed only with its identifier.

… Stats

A set of statistics and status are produced in each 30 second cycle. These “stats” consist of details about (a) the flights retrieved, processed, and stored, including any associated data (such as the hex code to flight mappings), (b) the filter evaluation results as alerts, including the number inserted, removed and active, (c) the system's operational state, including processing times and errors.

… Delivery

At the end of every 30 second cycle, alerts and stats are delivered to both MQTT and the console

The default MQTT topics are **“adsb/alert/insert”**, **“adsb/alert/remove”** and **“adsb/stats/loop”** with messages encoded in JSON. The alert insert and stats messages contain structured data, while the remove contains only the alert identifier (to match the original insert).

The console displays alerts and stats each on a separate line with timestamp data in a sequential format. Only alert inserts (and its data) are shown on the console. Alerts and stats are sent to stdout, while stderr will convey ongoing operational and executional output from the system, which may be extensive if debug options are enabled.

```
root@adsb:/opt/tracking-adsb/monitor# stdbuf -i0 -o0 -e0  mosquitto_sub -t "adsb/#" -v -h localhost
...
adsb/alert/insert
{"id":"squawks-4079e6","time":1749543448464,"timeFormatted":"2025-06-10T08:17:28.464Z","type":"squawks","text":"squaw
k 7031 anomaly: Military transponder code on non-military callsign <A7: Rotorcraft> (14.9km NNW 1,250
ft)","warn":false,"position":{"distance":14.869497409800681,"bearing":322.1266617372818,"relativeTrack":-196.63666173
728177,"cardinalBearing":"NNW","approachingStation":false},"altitude":1250,"speed":"133
kts","aircraftType":"A7","squawkInfo":{"anomalies":[{"type":"military-squawk-civilian","severity":"medium","details":
"Military squawk 7031 on apparent civilian flight","description":"Military transponder code on non-military
```

```
callsign"}],"matches":[{"begin":"7030","end":"7044","description":["Aldergrove
Approach"],"type":"approach","beginNum":7030,"endNum":7044},{"begin":"7030","end":"7046","description":["TC Thames/TC
Heathrow"],"type":"tc","beginNum":7030,"endNum":7046},{"begin":"7030","end":"7066","description":["Teesside
International
Airport"],"type":"approach","beginNum":7030,"endNum":7066},{"begin":"7030","end":"7077","description":["Aberdeen
(Northern North Sea Off-shore)"],"type":"offshore","details":["See note
3"],"beginNum":7030,"endNum":7077},{"begin":"7031","end":"7077","description":["RNAS
Culdrose"],"type":"military","beginNum":7031,"endNum":7077}]},"flight":"CASTLE21","timeLast":1749543448464}
adsb/alert/insert
{"id":"squawks-43c6f9","time":1749543448464,"timeFormatted":"2025-06-10T08:17:28.464Z","type":"squawks","text":"squaw
k 6401 anomaly: Military transponder code on non-military callsign <A5: Heavy (>300k lbs)> (185.2km NW
FL140)","warn":false,"position":{"distance":185.23844214447107,"bearing":305.6099594433146,"relativeTrack":29.7800405
5668538,"cardinalBearing":"NW","approachingStation":true},"altitude":14000,"speed":"315
kts","aircraftType":"A5","squawkInfo":{"anomalies":[{"type":"military-squawk-civilian","severity":"medium","details":
"Military squawk 6401 on apparent civilian flight","description":"Military transponder code on non-military
callsign"}],"matches":[{"begin":"6401","end":"6457","description":["Swanwick (Military)
Radar"],"type":"military","beginNum":6401,"endNum":6457}]},"flight":"TARTN29","timeLast":1749543448464}
adsb/alert/insert
{"id":"squawks-402988","time":1749543448464,"timeFormatted":"2025-06-10T08:17:28.464Z","type":"squawks","text":"squaw
k 7004: Aerobatics and Display <A1: Light (<15.5k lbs)> (67.8km WNW 3,800
ft)","warn":false,"position":{"distance":67.78593474359839,"bearing":270.2035027484809,"relativeTrack":-14.5535027484
80923,"cardinalBearing":"WNW","approachingStation":true},"altitude":3800,"speed":"44
kts","aircraftType":"A1","squawkInfo":{"type":"display","description":"Aerobatics and
Display","matches":[{"begin":"7004","description":["Aerobatics and Display"],"type":"display","details":["For use by
civil or military aircraft conducting solo or formation aerobatic manoeuvres, whilst displaying, practising or
training for a display or for aerobatics training or general aerobatic practice","Unless a discrete Mode A code has
already been assigned, pilots should select 7004 five minutes before commencement of their aerobatic manoeuvres until
they cease and resume normal operations","May be selected at pilot's
discretion"],"beginNum":7004,"endNum":7004}]},"flight":"GECRM","timeLast":1749543448464}
```


Implementation and testing

The system operating directory (“monitor”) contains the entirety of the source code (with, as standard, third party modules residing in the “node_modules” subdirectory). The source code is prettied and linted during development. The code is highly modular and extensible, particularly for extending and adding filters.

The system is live in two places. Firstly, within the London CTR, with a considerable amount of activity, including false positives. Secondly, remotely in Sweden, with considerably light activity. Both installations run on a Raspberry PI with branded (FlightAware and AirNav.Radar) sticks and software feeders for FlightAware, FlightRadar24, AirNav.Radar and ADSB-Exchange. In both cases, the system and the MQTT broker (mosquitto) run on the ADSB tracker hardware. In Sweden,

the alerts are fed into a local HTTP/HTML web server which renders a weather, camera and display. Certain alerts are then pushed with web-push to clients of that web server.

Tools and utilities

A "watcher" shell script exists in the source code directory. This processes the MQTT messages, as piped into it from the command line mosquitto client, to generate a formatted display.

```
root@adsb:/opt/tracking-adsb/monitor# stdbuf -i0 -o0 -e0  mosquitto_sub -t "adsb/#" -v -h localhost | ./watcher
...
```

```
08:25:49  | vicinity  | BAW9EW  | 9km WSW       | 1400 ft  | APPR (27)  | 141 kts  | A5  | nearby, look WSW direction tracking WNW
08:27:24  | squawks   | EXM44   | 65km NNW      | 3000 ft  | -          | 158 kts  | A1  | squawk 7417 anomaly: Military transpond
4.7km NNW 3,000 ft)
08:27:24  | squawks   | TARTN29 | 168km N       | 14000 ft | -          | 377 kts  | A5  | squawk 6060 anomaly: Military transpond
140)
08:27:24  | squawks   | GNBOX   | -             | 1100 ft  | -          | -        | B0  | squawk 0430 anomaly: Military transpond
08:27:24  | squawks   | GECRM   | 78km WNW      | 4400 ft  | APPR (50)  | 70 kts   | A1  | squawk 7004: Aerobatics and Display <A1
08:27:24  | anomaly   | DLH2YA  | 24km S        | 13675 ft | APPR (-74) | 405 kts  | A3  | anomalies: [1] high-speed-low-altitude
> (24.2km S FL137)
08:27:24  | anomaly   | OOOYS   | 191km ESE     | 32950 ft | -          | 232 kts  | A1  | anomalies: [1] low-speed-high-altitude
> (191.1km ESE FL330)
08:27:24  | anomaly   | BAW628  | 76km ESE      | 20000 ft | APPR (-10) | 457 kts  | A3  | anomalies: [1] high-speed-low-altitude
4km ESE FL200)
08:27:24  | airport   | AUA45XD | 19km W        | 775 ft   | APPR (9)   | 139 kts  | A3  | near EGLL [London Heathrow Airport] <A3
08:27:24  | airport   | GCEEU   | 38km W        | 800 ft   | -          | 123 kts  |     | near EGLM [White Waltham Airfield] (37.
08:27:24  | airport   | [4079d9] | 28km SSE     | 1175 ft  | -          | 46 kts   | A7  | near GB-0472 [Titsey Airstrip] <A7: Rot
08:27:24  | airport   | BAW9EW  | 15km W        | 275 ft   | APPR (15)  | 132 kts  | A5  | near EGLL [London Heathrow Airport] <A5
08:27:24  | airport   | GENTW   | 28km NNW      | 2100 ft  | APPR (-59) | 62 kts   | A1  | near GB-0204 [Plaistows Airfield] <A1:
08:27:24  | airport   | CFE35N  | -             | -        | -          | -        | A3  | near EBFN [Koksijde Air Base] <A3: Larg
08:27:24  | lifting   | AUA45XD | 19km W        | 775 ft   | APPR (9)   | 139 kts  | A3  | From: London Heathrow Airport, Climb: 2
08:27:24  | lifting   | GCEEU   | 38km W        | 800 ft   | -          | 123 kts  |     | From: White Waltham Airfield, Climb: 64
08:27:24  | vicinity  | BAW19W  | 4km S         | 3000 ft  | -          | 168 kts  | A5  | nearby, look S direction tracking WNW <
```