A Comparative Analysis of Aireon's Space-based ADS-B Technology and Other Alternatives for Air Traffic Management

Philip Beswick @00662943

The University of Salford

Aircraft Engineering with Pilot Studies

Miss Frances Wolff, Dr Viktoriia Myroniuk

25/04/25

Author Note

Firstly I'd like to thank Terminal2Solutions for their support throughout the course of their project. They have kindly allowed me to use their experimental software free of charge in order to complete my simulations, while providing many, many hotfixes at all times of the day and night to make sure I can complete this research. I would also like to thank my parents, friends and family for constantly being supportive throughout my academic journey. Thank you.

Abstract

Space-Based ADS-B is a new method of receiving and transmitting ADS-B communication signals from aircraft to air traffic surveillance and tracking stations. It is an advantageous method of surveillance and tracking since it requires no ground stations. Instead, coverage is provided by a constellation of satellites, which achieves global reach. This project aims to explore the implications of this when used for air traffic management, in particular over the North Atlantic Ocean. Ultimately, simulations conducted show that the use of Space-Based ADS-B methods of air traffic management reduces flight times at the expense of the agencies needing to manage the airspace; they have to deal with more conflicts than if tracks were used.

Keywords: ADS-B, RADAR, Aircraft Tracking, Aircraft Surveillance, Nav Canada, NATS, The Organised Track System, GMT, IFF, Open-Source, Sector File, AIRAC Cycle, Rhumb Line, Great Circle,

A Comparative Analysis of Aireon's Space-based ADS-B Technology and Other Alternatives for Air Traffic Management

Contents

List of Figures

List of Tables

1 Introduction

Space-based ADS-B has the potential to be the next generation in aircraft surveillance and aircraft tracking due to its global coverage, which is especially important for remote areas, such as the North Atlantic Ocean, where it was not previously possible to track aircraft on traditional RADAR. From 2017 to 2023, SpaceX launched 66 satellites, which housed the Space-based ADS-B technology developed by Aireon, and since the technology has been trialled by NATS and Nav Canada over the North Atlantic Ocean.

Traditionally, aircraft are surveilled in all phases of flight, from before they power up their engines on the apron to when they power down at their destination, aircraft are constantly monitored by and communicate with Air Traffic Controllers. Aircraft are monitored by Primary and Secondary RADAR as well as ADS-B systems. Usually, a combination of these different approaches is used to get the most accurate information about the state of an aircraft. ADS-B has, however, become increasingly popular and more widely used in contrast with primary and secondary radar for aircraft surveillance. This is because ADS-B technology is comparatively much cheaper than RADARs while also providing additional capabilities, such as Air-to-Air surveillance, that RADAR does not (Avionix Technologies, n.d.). Yet, both ADS-B and RADAR have the same limitation - they require a line-of-sight between aircraft and ground stations to function.

In some parts of the world, the installation of a ground station is simply not possible. This could be due to an extreme financial cost that would be incurred in very remote areas, or due to physical impossibilities, such as over Oceans. Therefore, over large portions of the planet, it is not possible to provide aircraft tracking or surveillance services. This is especially true over the North Atlantic Ocean.

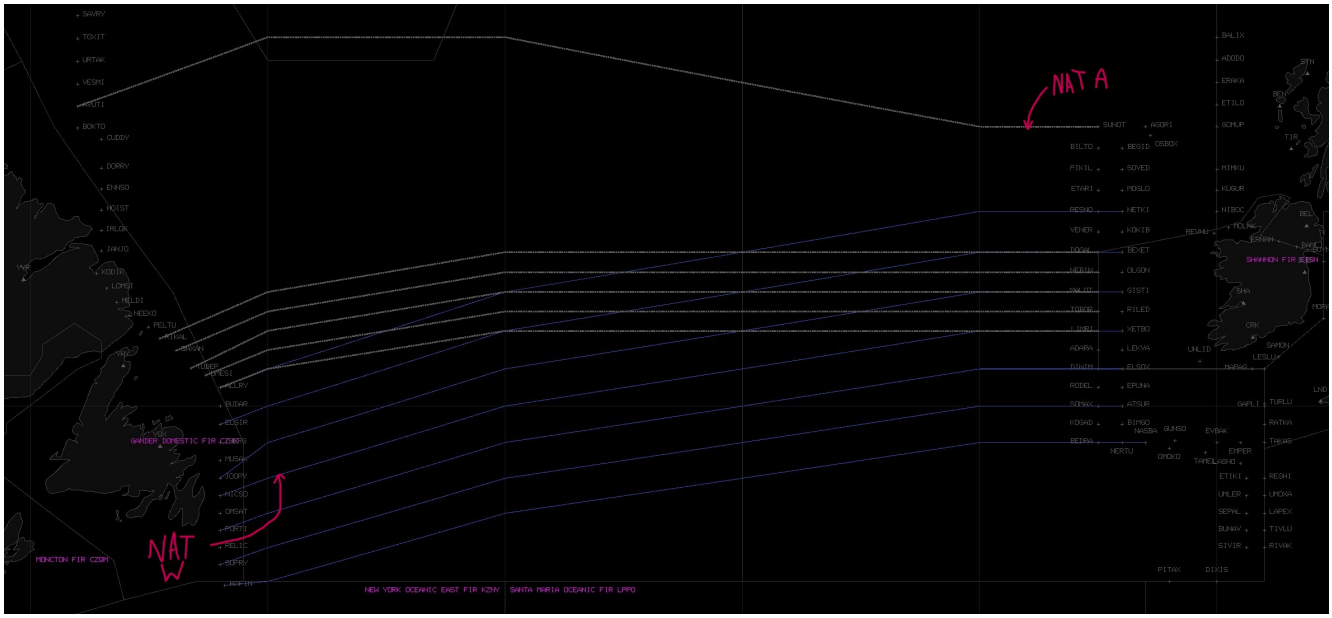
In 2012, approximately 460000 flights crossed the North Atlantic Ocean (SKYbrary, n.d.), with this number rising to over 580000 in 2023 (ForeFlight, n.d.) making the airspace some of the busiest in the world. This means that around 1500 aircraft make the trans-Atlantic crossing every single day. Keeping aircraft safe while making this crossing is made difficult by the lack of tracking or surveillance services that were previously mentioned. So, the Organised Track System was developed.

1.1 The Organised Track System

The Organised Track System (OTS) was designed to ensure aircraft remain safely separated when flying across the North Atlantic Ocean. Each day, Nav Canada and NATS jointly publish a list of directional "tracks" across the North Atlantic airspace, with each track following a great circle (Rodionova et al., 2014). Figure 1.1 below, shows the Eastbound and Westbound tracks that were in use on April 7th 2025.

Figure 1.1

An image taken from the RADAR software showing the North Atlantic tracks that were used on April 7th 2025. NAT A and NAT W are labelled.



As alluded to previously, these tracks are directional, this means that each track has traffic operating in only one direction. For example, in Figure 1.1 above, North Atlantic Track (NAT) A carries traffic moving westbound, whereas NAT W carries traffic moving eastbound. In order for an aircraft to make the crossing over the Atlantic, they must fly on one of the designated tracks, starting and ending at that track's respective entry and exit points. To further aid in the smooth running, at any given time, tracks are only ever in use in one direction. For 12 hours each day only the eastbound tracks are in use and for the other 12 hours westbound tracks are in use. For anyone who has ever made a trans-atlantic flight, this is why flights from Europe to the United States of America tend to leave Europe in the early

morning, but when making the return journey, they leave the United States in the late evening.

So, the existence of tracks means at any given point in time, Air Traffic Controllers are aware of which track an aircraft is on and by extension, which way they are going. However, there is still no way of monitoring their progress once they enter the track. Therefore, strict separation requirements are enforced. These are (SKYbrary, n.d.):

1. One Degree of latitude, approximately 60 NM, for aircraft on the same track,
2. 1000ft of vertical separation.

This is monitored by voice communications over HF Radio. Every 15 minutes, aircraft provide a position report, stating their current coordinates in latitude and longitude, speed, heading and altitude, following where they expect to be in 15 minutes' time. Using this, Air Traffic Controllers are able to build up a picture of all the aircraft along a track, and can ensure there is enough vertical and horizontal separation between all aircraft at any given point in time.

Additionally, Air Traffic Controllers are able to plan for aircraft operating at different speeds, so faster moving aircraft won't be held up by other aircraft flying much slower.

The main limitation of the OTS is the tracks themselves. On any given day, only a small number of routes are available across the Atlantic. The routes are designed to be the most optimal for all traffic, generally, when accounting for the wind direction and speed. Unfortunately, these tracks can never be perfect for all operators flying across the Atlantic each day. Often, aircraft have to fly out of their way to reach a NAT entry point, or are taken out of their way to a NAT exit point, when there was a more direct great circle route that could have been taken. Indirect routes ultimately mean that aircraft burn more fuel, which increases the flight's impact on the climate while also making the cost of the ticket more expensive for the consumer.

Furthermore, the strict separation requirements (due to a lack of tracking or surveillance) mean that the airspace quickly reaches maximum capacity, which means that all Atlantic crossings have to be carefully planned, ensuring that there will be space for the aircraft to make their journey.

This is why Aireon developed the Space-Based ADS-B technology as previously mentioned.

1.2 The Alternative: Space-Based ADS-B

Space-Based ADS-B essentially replaces ground station receivers with satellites, which orbit the Earth. Therefore, traditional ADS-B coverage can be provided anywhere on the planet, including remote areas that were previously inaccessible. This means that NATS have been able to operate several zero track days, essentially meaning that there were no westbound NAT tracks over the North Atlantic Ocean; aircraft were free to plot their own, random route, under surveillance from Air Traffic Controllers. The first of these days occurred on the 9th of March 2021 (Young, 2021). The Aireon Space-Based ADS-B system is not expected to replace the OTS system for several years, as changing how thousands of aircraft use a bit of airspace every day takes significant planning and resources.

Additionally, there have been no concrete publications comparing how airspace is used on zero track days vs the normal OTS. This, therefore, is the aim of this research: to simulate air traffic over the North Atlantic, using both OTS procedures and zero track procedures, in order to determine key differences in both the performance of the airspace and of different types of traffic, operating typical routes across the North Atlantic Ocean.

1.3 Project Plan

The plan for this project was completed by the author of this paper and agreed upon by the project supervisor. The simulation engine that was developed as part of this project was provided by an organisation, Terminal2Solutions (Terminal2Solutions, 2024), free of charge, with extra development work being completed by the author to make the engine usable for this project. This means that multiple delays were experienced throughout the Project's execution due to some software not being ready. However, due to contingency planning, which will be explored in Chapter 1.2.3, the project remained on time and was completed by the required deadline.

1.3.1 Aims and Objectives

Ultimately, this project aims to understand how Space-based ADS-B air traffic management methods (zero track) compare to traditional air traffic management methods

(OTS) over the North Atlantic Ocean. In doing so, a full and complete understanding of the importance of Space-Based ADS-B will be developed. Additionally, this research will also fill a gap in the current published literature, being the first to directly compare OTS and Zero Track systems of air traffic management. In order to accomplish this, three key objectives have been defined:

1. To identify other viable alternatives to Space-based ADS-B and identify how they function,
2. To simulate Traffic Flow and Traffic Efficiency with the different systems in place,
3. To present findings and make suggestions for where Space-based ADS-B could be implemented in the future.

By keeping these key objectives in mind, the project will remain focused and on time, resulting in a well-rounded final piece of work that will meet the aim successfully.

1.3.2 Areas of Project Integration

This project has many moving parts, most of them revolving around the software development for the simulation engine that had to be created. Generally speaking, though, this project has four key areas of integration.

First of all, Theory was studied to understand the current state of Space-Based ADS-B on the market, and how it is used. This aspect of the project was studied and completed in the Literature Review, which is summarised in Chapter 1.2.

Next comes Software Development and Simulation. This aspect of the project has taken approximately 4 months to complete. The key details of the software development and simulations will be explored in Chapter 2. It is also important to emphasise that several pieces of software were developed for this project; these range from simple scripts to aid in data creation and collection (using Python) all the way to the development of the complex simulation engine created in Java. To make the simulations realistic, real-world air traffic data was collected and sampled, so that the simulation engine could be effectively tuned.

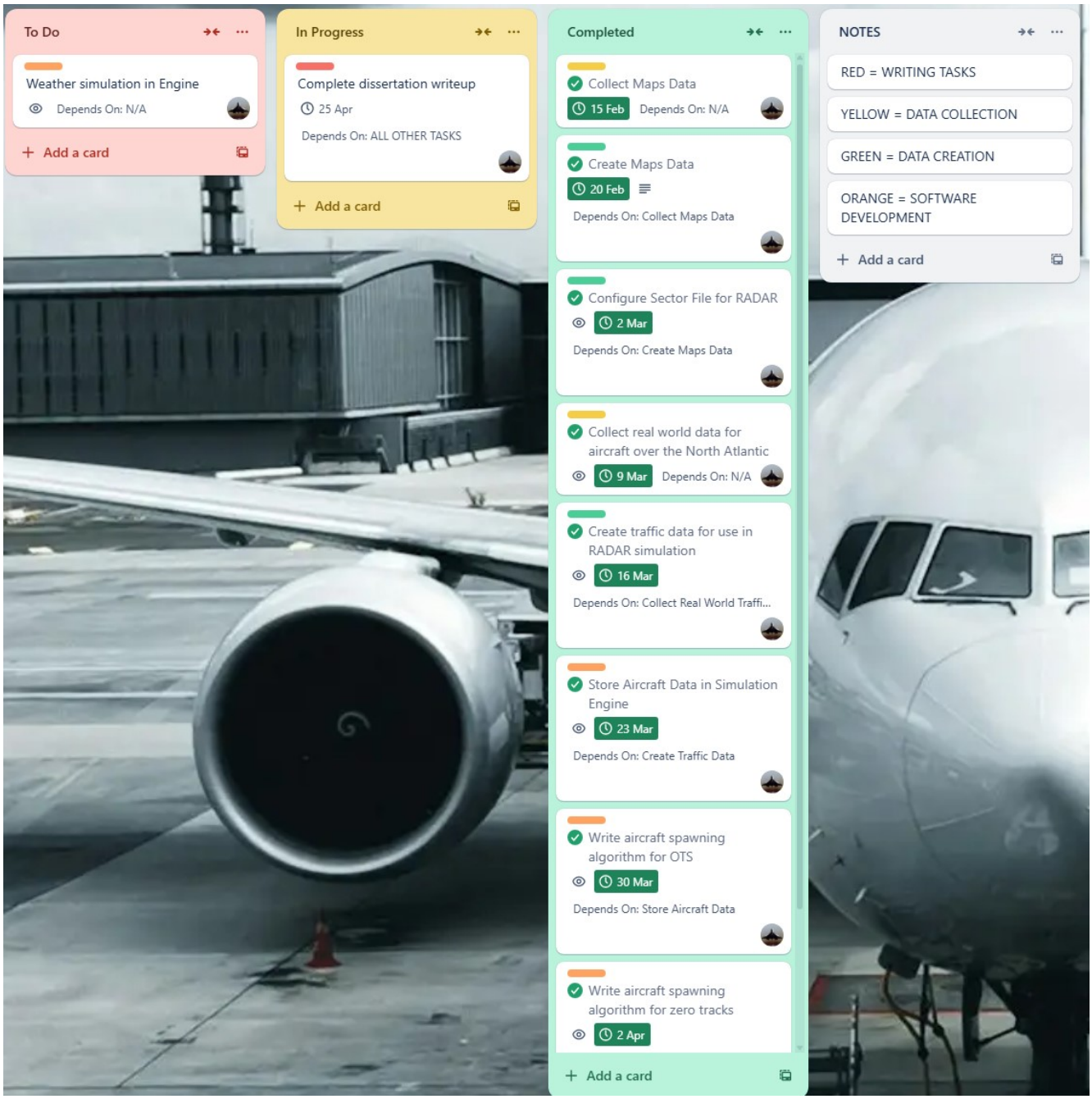
Finally, with the results from the simulations, recommendations and optimisations can be suggested, which will be explored in Chapter 6.

1.3.3 Project Management

Attempting to simulate air traffic is an incredibly complex and intricate process, with a significant amount of software development required. It is, therefore, incredibly important to ensure the project remains on time and on task. Instead of using a Gantt chart for this phase of the project, the online tool Trello was used. Trello has all the advantages of a Gantt chart, while also being very visual and interactive. The project's "Trello Board" can be seen in Figure 1.2.

Figure 1.2

An image taken from the project's TRELLO Board, showcasing a selection of the project's activities while demonstraiting how the Project Management application has been used.



As seen in the above figure, each of the main phases of the project has a card, which can be passed between three different phases: To Do, In Progress and Completed. As well as this, each card displays the task's due date and all other tasks that depend upon it. This tool was incredibly useful in conducting the project and ultimately ensured that it was completed

on time.

When setting task due dates, the intention was to provide more time than was reasonably needed for each task. Primarily, this was done to combat delays pre-emptively while creating a stress-free environment to work in. Ultimately, this approach was very successful and kept the project running on time throughout the process.

1.4 Literature Review Summary

As Space-based ADS-B is very modern technology, very little literature exists that is relevant to this piece of work. Similarly, since the technology is currently undergoing testing, it has not yet been certified for commercial use, and therefore, technical information about Aireon's satellites is limited. Therefore, the literature reviewed for this project focused on the theoretical foundations of Space-Based ADS-B, how ADS-B works and methods of air traffic management over the North Atlantic Ocean. A thematic literature review was conducted, with the goal of spotting key trends and potential gaps in the current literature.

1.4.1 North Atlantic Tracks and OTS System

In *North Atlantic Aircraft Trajectory Optimisation*, Rodionova (and others) discuss how the OTS manages trans-atlantic flights by assigning structured routes that are east-bound in the evening (GMT) and west-bound in the morning (GMT) (Rodionova et al., 2014). This leads to suboptimal routing for many aircraft and therefore increased flight times and higher fuel burn, which ultimately leads to higher ticket prices and greater environmental impact. This system is used because there is a lack of surveillance equipment over the ocean and so aircraft have large spacing requirements (60 NM horizontally and 1000ft vertically), while also making a position report every 15 minutes (Rodionova et al., 2014).

1.4.2 Technological Foundations of Space-Based ADS-B

In *Aireon Space Based ADS-B Performance Model* Garcia (and others) discuss how ADS-B evolved from World War 2 Identification Friend-Or-Foe systems (IFF) (Garcia et al., 2015). Initial systems like Mode S used interrogation and reply mechanics, which were then upgraded in the 1990s to 1090ES ADS-B. This new standard of ADS-B technology allowed aircraft to automatically broadcast their GPS position, without being first interrogated; as they would have been if Mode S was being used.

However, conventional ADS-B has two major limitations:

1. Line of sight is required,
2. Signal range is limited to about 450 km(Flightradar24, 2024).

In *Space-Based ADS-B: Performance, Architecture and Market* Baker suggests that Space-Based ADS-B is the solution to these limitations as with a sufficient satellite constellation (in this case Aireon has 66 satellites), global coverage can be achieved, including oceanic and remote regions Baker, 2019. In *Aircraft Surveillance from Space: The Future of Air Traffic Control* Budroweit (and others) make a similar observation Budroweit et al., 2024.

Furthermore, Baker distinguishes between ADS-B and ADS-C, the later being a satellite communication based tracking tool, used for non-surveillance purposes. This is because ADS-C is limited by update intervals, making it unsuitable for real-time ATC surveillance (Baker, 2019).

1.4.3 System Performance and Metrics

In their respective publications, Garcia and Baker discuss that to be ICAO compliant, any ATC surveillance system must meet the following criteria (International Civil Aviation Organization, 2007):

- An up-time of 99.9% or greater per year,
- Less than 2 seconds of latency,
- Update intervals at intervals of no more than 8 seconds, at a 95-96% confidence level.

Aireon's system is currently the only Space-Based ADS-B provider that meets these ICAO standards; other providers currently on the market, such as Globalstar and Spire, can only meet tracking standards, and some do not offer global coverage. Aireon is able to meet these strict requirements due to its use of the Iridium-NEXT satellites, which are Low Earth Orbit (LEO) systems; their proximity to Earth reduces signal propagation times.

1.4.4 Implementation Challenges and Solutions

Garcia suggests that there are two major challenges when implementing Space-Based ADS-B systems.

First of all, signal interference from older systems, like Mode S and IFF, which cause False Replies Uncorrelated In Time, or FRUIT, errors (Garcia et al., 2015). These errors mean that the newer system, in this case Aireon's Space-Based ADS-B system, spends time processing old, useless data, and therefore ignores real signals that are important.

Secondly, overlapping satellite coverage can lead to signal garbling in areas of high traffic density (Garcia et al., 2015). This means that signals may never reach a satellite, or possibly arrive in an unreadable state.

Baker says that Aireon uses probabilistic modelling to combat both of these problems. This essentially involves mathematically predicting when a message is expected to be received by a satellite and what it is expected to say (Baker, 2019). This method can be used to reconstruct an unreadable signal or replace a signal that was never received.

1.4.5 Applications and Market Implications

As suggested by Garcia and Baker, the primary application of Space-Based ADS-B is in air traffic surveillance, providing real-time position data for collision prevention and traffic management (Baker, 2019 and Garcia et al., 2015). This differs from solutions currently available on the market as Space-Based ADS-B can monitor aircraft in areas of the planet not currently accessible to Air Traffic Controllers, such as Oceanic regions and remote areas. An article published by Pyramid Media Group Inc. reinforces this idea and that the implementation of Space-Based ADS-B will eventually lead to improved efficiency and capacity over the North Atlantic Ocean (once it replaces the OTS) Business, 2022.

This, however, is not the only use-case for Space-Based ADS-B. The technology can also be used to aid in aircraft incident investigations, especially those that occur in remote areas. Additionally, airlines can use the technology to track their fleet of aircraft for operational planning and information. Both of these use cases do not require the level of fidelity that aircraft surveillance systems must meet. Instead, they can comply with Aircraft Tracking solutions, such as those provided by Globalstar and Spire.

1.4.6 Gaps in the Literature

While conducting the literature review, it became clear that no publication attempted to discuss Space-Based ADS-B's impact on Air Traffic Management over the North Atlantic

ocean using numerical and factual data. The article published by Pyramid Media Group Inc. quoted a NATS air traffic controller as saying the testing of Aireon's system was "transformational" and resulted in aircraft saving time, money, fuel and therefore fewer emissions (Business, 2022). Still, there was no attempt to quantify this claim, however logical it may seem. This gap will attempt to be filled by the research and simulations conducted in this paper.

1.4.7 Final Summary

The literature review conducted finds that Space-Based ADS-B is a significant technological evolution with transformative potential, especially over remote areas. Currently, Aireon is the only provider that meets ICAO standards, and therefore, the testing of this technology is limited to just the North Atlantic region under the supervision of NATS and Nav Canada.

The paper written by Garica (and others), *Aireon Space-Based ADS-B Performance Model* was produced by a team from Aireon and therefore it is important to discuss if any bias is present. While there is a reasonable expectation for the paper to have some bias, overall it makes claims very similar to the other pieces of work that have previously been discussed and therefore it is a useful body of work to consider for this project.

Based on this literature review, the research outcome for this project was decided to investigate, by quantifying, the actual usefulness of Aireon's Space-Based ADS-B system, in terms of creating more efficient air traffic and airspace management over the North Atlantic Ocean.

2 Methodology

The goal of this research project is to compare two different methods of air traffic management, in particular over the North Atlantic Ocean. Since NATS and Nav Canada do not publish data on when they operate “zero tracks”, real-world data cannot be directly used to make this comparison. Therefore, it was decided that the best approach would be to create a simulation engine, which could be used to make a comparison. For the simulation engine to be a useful method of comparison, it is important to build it using real-world data and statistics.

With this in mind, much of the methodology will explore how the simulation engine was built and then how it was calibrated to match how aircraft behave over the North Atlantic Ocean in reality. The engine was not created entirely from scratch for the project; rather, an existing, in-development RADAR application created by Terminal2Solutions (Terminal2Solutions, 2024) was sourced. The product produced by Terminal2Solutions provided the basic framework, which was then customised and built upon to the required specifications for this project. This application was provided voluntarily and at no expense to the researcher. The following subchapters will explore how this application was modified and expanded for the purposes of this project.

2.1 Simulation Engine Data Creation and Collection

In order for the application display to be useful and readable, especially to a person unfamiliar with the geography of navigation aids on either side of the Atlantic ocean, it was decided to import geographical map data into the software. The following subchapter will explore how that was accomplished.

2.1.1 Collection and Formatting Maps Data

A charitable organisation called Natural Earth (Patterson and Kelso, 2009) provides open-source geographical data for the entire world, which includes everything from maps of roads and cities, to basic coastline data. For the purposes of this project, a map only needs to display the coastline of counties, specifically,

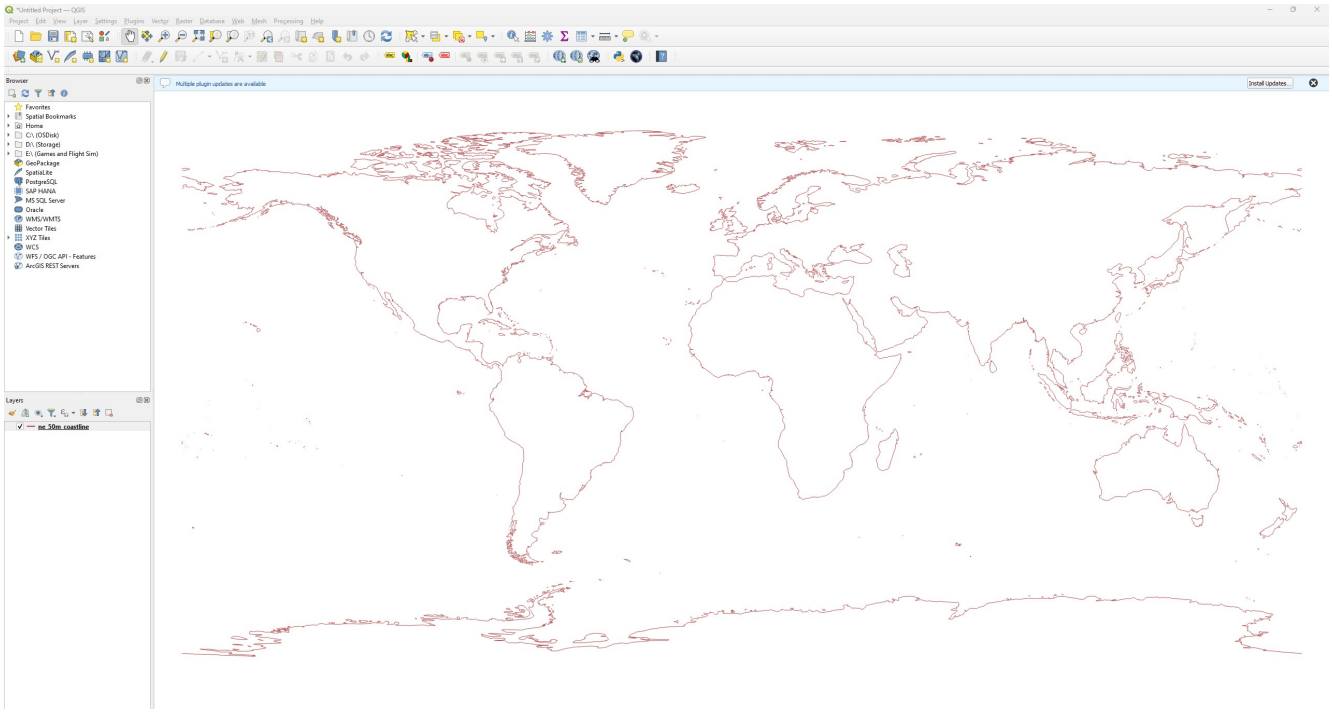
1. The United Kingdom,
2. The Republic of Ireland,

3. Iceland,
4. The entirety of western Europe, including Scandinavia and Greenland,
5. The eastern seaboard of the United States of America and Canada.

However, Natural Earth provides this data for the whole globe, not specific countries. Therefore, a resolution of 1:50m was selected and then information was extracted from the map of the world using a piece of open-source software called QGis (QGIS Development Team, 2002). Figure 2.1 shows the coastline data for the entire world, from Natural Earth, imported inside of QGis.

Figure 2.1

Coastline data for the entire Earth, imported inside of QGis.

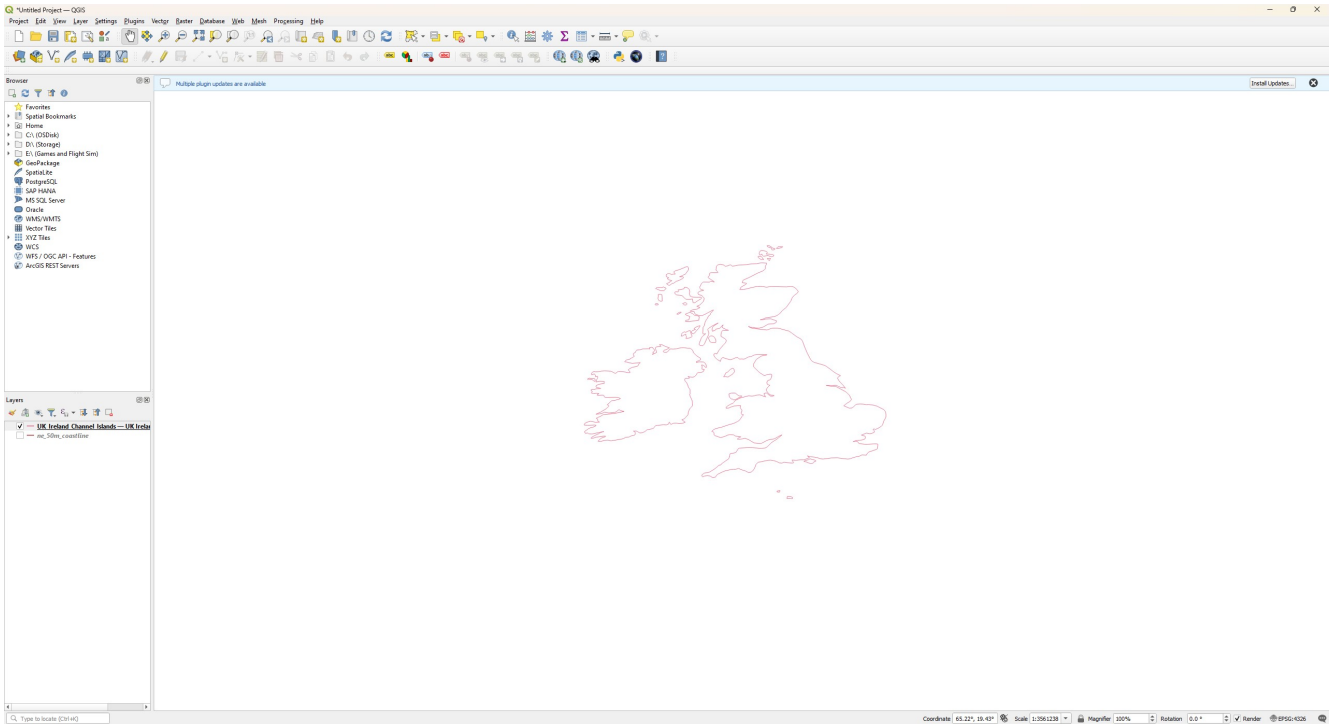


QGis is a useful tool as it allows specific regions, and the data belonging to that region to be extracted. At this point, it is important to understand the details behind the coastline data. Simply put, the image seen in Figure 2.1 is created by drawing a straight line between two coordinates in latitude and longitude. Each individual point is spaced 50m away from the others, hence the ratio mentioned earlier. The RADAR software that is used for this project stores map data in a very similar way, so the goal of this process is to extract a list of

coordinates which the software can then draw lines between to create a complex shape, in this case, the outline of a land mass. The map shown in Figure 2.1 contains hundreds of thousands of these coordinate pairs, which is why QGis is used to isolate just the regions of interest - it is simply not practical to complete this step by hand. For example, Figure 2.2 shows the isolated map data for the United Kingdom, the Republic of Ireland and the smaller Islands that comprise the British Isles.

Figure 2.2

Coastline data for the British Isles, imported inside of QGis.



For the purposes of this subchapter, the entire process will be shown to import the British Isles into the RADAR software, all other landmasses were produced in an identical manner.

Once the data, as seen in Figure 2.2, has been isolated, it needs to be imported into the RADAR software. Unfortunately, this is not a straightforward process due to the different ways data is stored between QGis and the RADAR. QGis outputs the data in a .geojson file type, a subsection of this data can be seen in Figure 2.3.

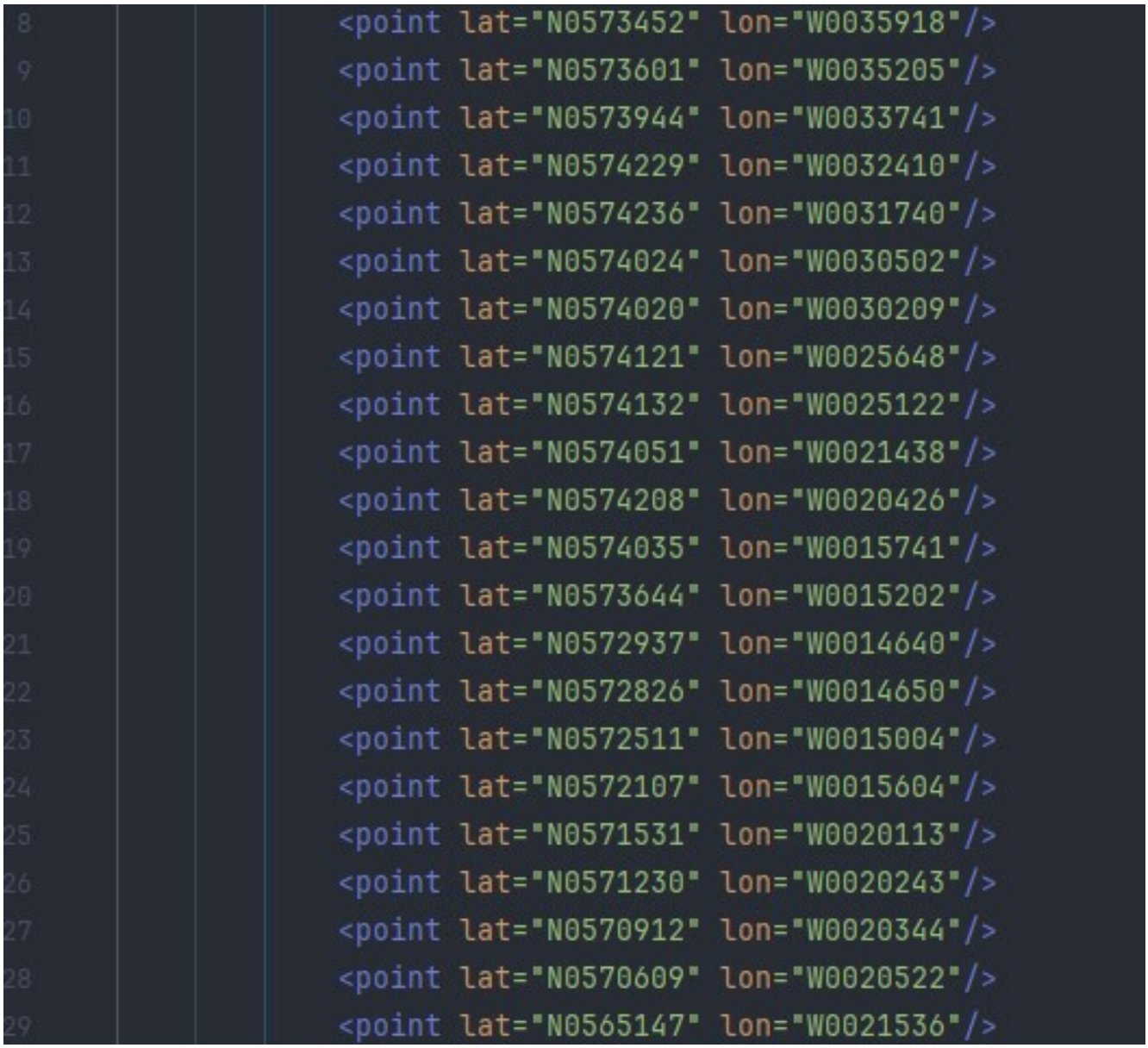
Figure 2.3

A subsection of GEOJSON data for the British Isles. The contents of this data is not important, the aim is to show the vast nature of the data.

The first problem is that RADAR expects to receive coordinates of latitude and longitude in the format DDDMMSS, not in decimal format as shown here. Additionally, the format of the data needs to change from a .geojson file to an .xml file. The researcher did not make these decisions; it is simply a requirement of the RADAR software, which must be conformed to. In particular, the RADAR software expects to receive map data in the format shown in Figure 2.4.

Figure 2.4

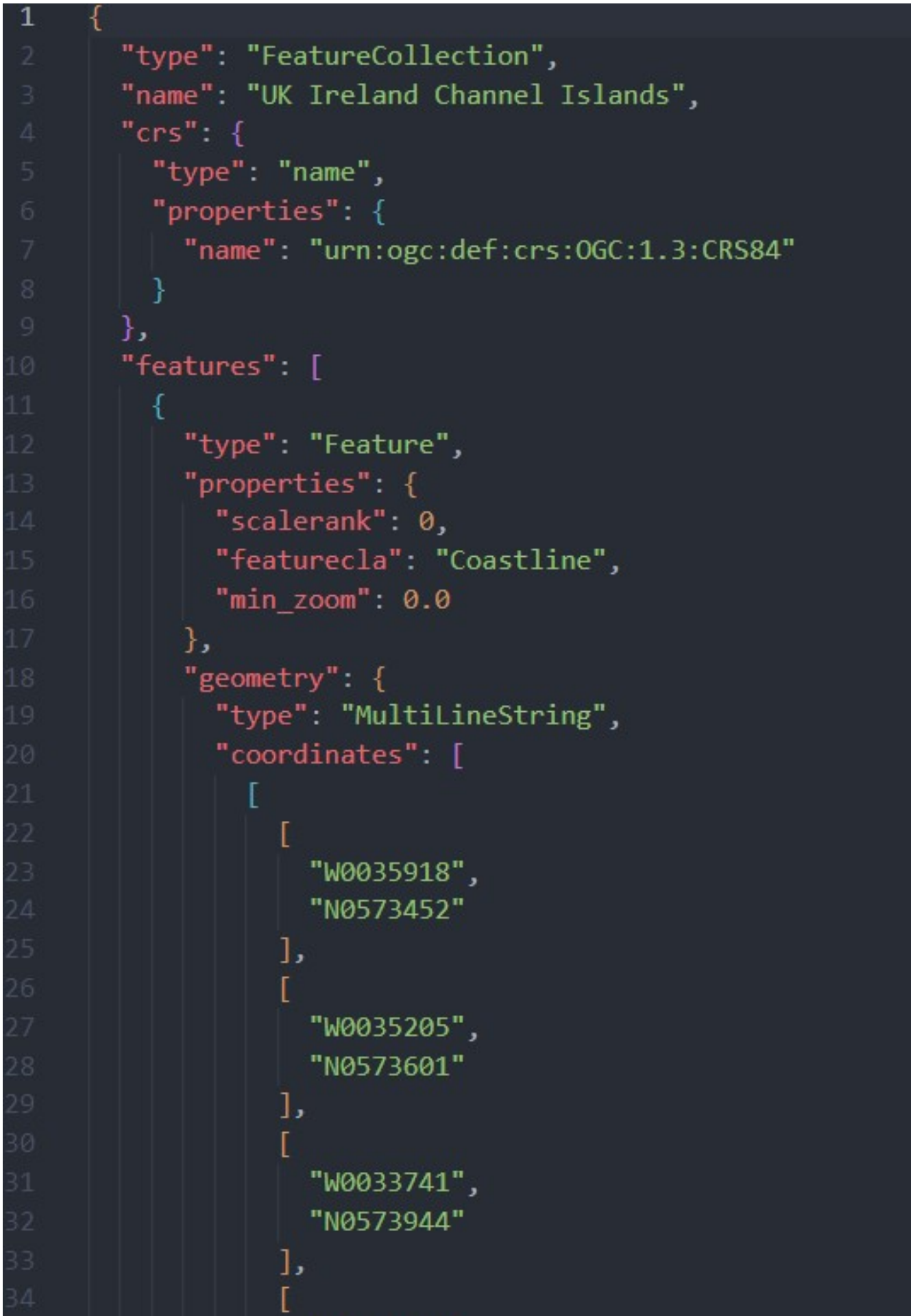
Expected format of Coastline data for the RADAR software; its quite different from the format provided in the GEOJSON file.



Due to the vast amount of data that had to be handled for this phase of the project, it was decided to write two simple scripts in Python3. The first script took the output from QGis shown in Figure 2.3 and converted all the coordinates from radians to DDDMMSS as well as eliminating the unnecessary data from the original .geojson file. This produces an output, a subsection of which can be seen in Figure 2.5.

Figure 2.5

A subsection of the GEOJSON data trimmed down by the first Python3 script.

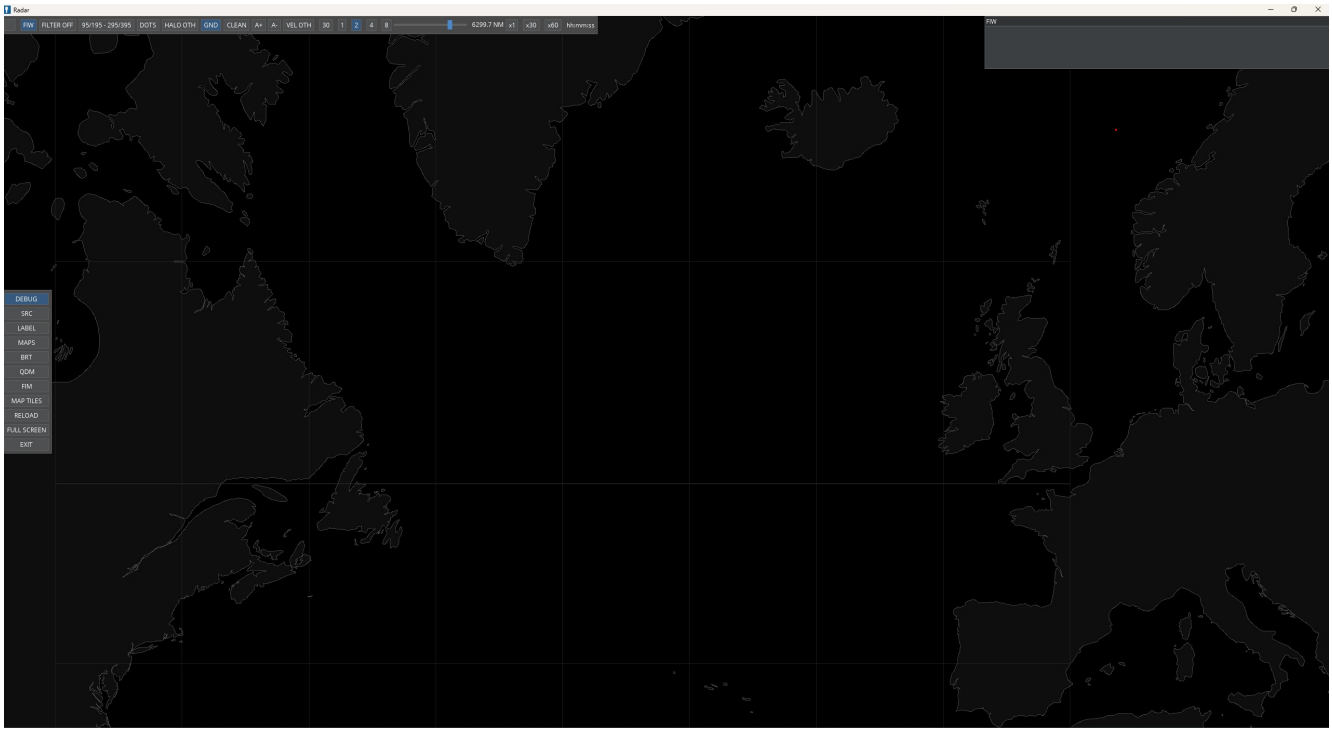


```
{
  "type": "FeatureCollection",
  "name": "UK Ireland Channel Islands",
  "crs": {
    "type": "name",
    "properties": {
      "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
    }
  },
  "features": [
    {
      "type": "Feature",
      "properties": {
        "scalerank": 0,
        "featurecla": "Coastline",
        "min_zoom": 0.0
      },
      "geometry": {
        "type": "MultiLineString",
        "coordinates": [
          [
            [
              "W0035918",
              "N0573452"
            ],
            [
              "W0035205",
              "N0573601"
            ],
            [
              "W0033741",
              "N0573944"
            ],
            [
```

At this stage, very obvious coordinates start to appear in the file; these now need to be manipulated to be put in the format shown in Figure 2.4. This was the purpose of the second Python3 script. This puts the data into the correct format, which can then be injected into the RADAR software, giving an outcome as seen in Figure 2.6.

Figure 2.6

Finalised coastline data inside of the RADAR software.



To create this image, 16912 lines of .xml map data were created, using the two Python3 scripts previously mentioned. Both of these scripts, in their entirety, can be found in Appendix A. While this data is not strictly necessary for the project to be completed successfully (aircraft can be simulated on a blank RADAR), it adds much-needed context for people who are not as aware of the geography of the region, or the position of Nav-adis either side of the Atlantic, making this body of work easier to understand.

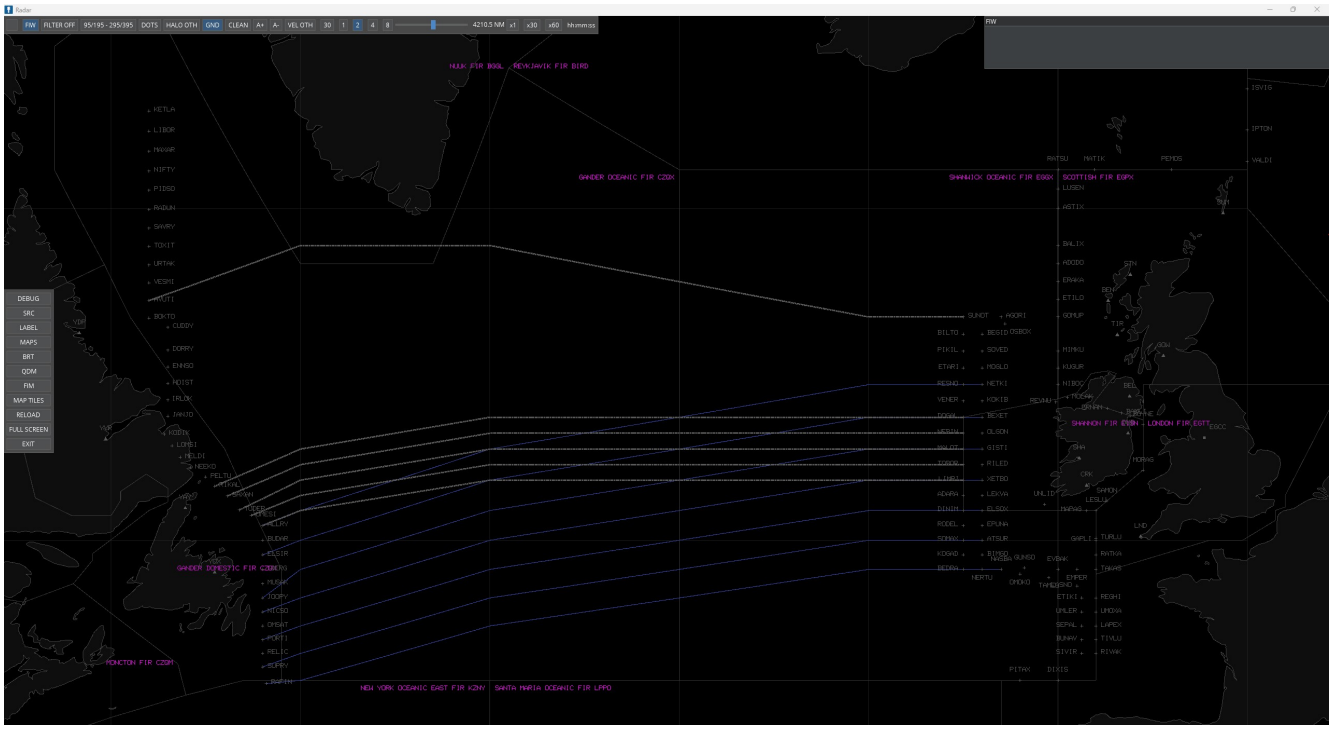
2.1.2 Sector File Creation

In the context of the RADAR software, a Sector file displays all the aviation information and data on top of geographical maps. This data was created by hand, from sources published by various international and national organisations responsible for maintaining air traffic services. This data is vitally important as the “AI” aircraft that will

eventually be simulated uses this data to navigate across the airspace. Figure 2.7 shows the RADAR screen depicted in Figure 2.6, but with the sector file information also displayed.

Figure 2.7

Finalised sector file displayed in the RADAR software. This image shows all relevant Nav-aids and oceanic tracks overlayed on the coastline data.



Information shown in this image includes:

- Flight Information Region (FIR) boundaries,
- Oceanic Entry/Exit Points (displayed as a cross with their respective names),
- A small selection of VORs (displayed as triangles with their 3 character ICAO definitions),
- The North Atlantic Oceanic tracks in use on April 7th 2025:
  - Grey tracks operate from Europe to the United States,
  - Blue tracks operate from the United States to Europe.

All of the FIR region boundaries were sourced from ICAO and processed identically to the geographical coastline data as discussed in the previous subchapter (International Civil

Aviation Organization, n.d.). Whereas, all the navigation aids (entry/exit points and VORs) were obtained from the FAA's chart of the North Atlantic Ocean from the most recent AIRAC Cycle (Federal Aviation Administration, n.d.-a) and the North Atlantic Tracks were obtained from the FAA's NAT track publisher(Federal Aviation Administration, n.d.-b).

As this data volume is much smaller than the geographical data, it was hand-formatted in .xml files in a format required by the RADAR software.

2.1.3 Air Traffic Data

Now that the RADAR software has been configured, aircraft data needs to be created. So that the traffic can be made as realistic as possible, real-world air traffic over the North Atlantic Ocean was monitored over a 24-hour period. This subsection will explore how the air traffic was monitored and how it became useful for the overall simulation that was conducted.

The Flight Aware AeroAPI (FlightAware, n.d.) was used to automatically monitor live air traffic over the North Atlantic Ocean. To achieve this, a third Python script was developed (you can find this in Appendix B). This script had a very basic function; every hour, it queried the AeroAPI to get a list of all aircraft over the North Atlantic Ocean at that given time. Specific data regarding each aircraft was also obtained, such as the aircraft type and the destination and arrival airports. With this information in hand, the simulation engine can be tuned to ensure that it is representative of an average day's traffic over the North Atlantic.

It is important to specify that since the AeroAPI depends on ADS-B signals, it is not always 100% effective. In fact, sometimes data cannot be communicated, which may result in an unknown parameter for some tracked aircraft.

However, in total, 1476 aircraft were captured throughout the 24-hour period. Table 2.1 shows the top 19 aircraft in this period (in terms of frequency) and their respective probability of performing a flight. These numbers are corrected for aircraft that returned an "unknown" value as described above.

Table 2.1

This table shows each of the different aircraft that are simulated, as well as their weighted probability of making a flight across the Atlantic.

Aircraft Type	Probability
B789	0.125313
B77W	0.112782
B788	0.097744
B772	0.090226
A359	0.087719
A333	0.085213
A332	0.058897
A339	0.057644
A35K	0.052632
B77L	0.045113
B748	0.027569
B78X	0.025063
B763	0.036341
B774	0.02381
B764	0.02005
A388	0.02005
A343	0.012531
B752	0.011278
A346	0.010025

With this data, the RADAR software can ensure that the most common aircraft to make a trans-Atlantic crossing in reality are also the most common in the simulation. This is important as all aircraft have unique performance parameters, which will impact not only how they interact with one another in the air, but also how much fuel they burn when flying in different conditions.

Inside the RADAR software, each of the 19 aircraft types was defined along with their respective performance characteristics, in particular maximum clean airspeed, minimum clean airspeed and maximum service ceiling height. These values were retrieved from information published by Eurocontrol (EUROCONTROL, n.d.) and are values of their True Airspeed (TAS) in knots. Figure 2.8 shows this data structure being initialised inside the RADAR software.

Figure 2.8

This shows the aircraft data as raw code inside of the RADAR simulation engine. This code was wrote inside of JAVA.

```
static private HashMap<String, AircraftType> initialiseTypeData() {  1 usage

    HashMap<String, AircraftType> data = new HashMap<>();

    data.put("L4J", new AircraftType( type: "L4J",  manufacturer: "Aerospatiale",  family: "CONC",  minCleanAirspeed: 250,  maxCleanAirspeed: 1178,  maxCeiling: 60000));

    data.put("A332", new AircraftType( type: "A332",  manufacturer: "Airbus",  family: "A330",  minCleanAirspeed: 250,  maxCleanAirspeed: 470,  maxCeiling: 41000));
    data.put("A333", new AircraftType( type: "A333",  manufacturer: "Airbus",  family: "A330",  minCleanAirspeed: 250,  maxCleanAirspeed: 475,  maxCeiling: 41000));
    data.put("A339", new AircraftType( type: "A339",  manufacturer: "Airbus",  family: "A330",  minCleanAirspeed: 250,  maxCleanAirspeed: 475,  maxCeiling: 41000));
    data.put("A343", new AircraftType( type: "A343",  manufacturer: "Airbus",  family: "A340",  minCleanAirspeed: 250,  maxCleanAirspeed: 490,  maxCeiling: 41000));
    data.put("A346", new AircraftType( type: "A346",  manufacturer: "Airbus",  family: "A340",  minCleanAirspeed: 250,  maxCleanAirspeed: 480,  maxCeiling: 41000));
    data.put("A359", new AircraftType( type: "A359",  manufacturer: "Airbus",  family: "A350",  minCleanAirspeed: 250,  maxCleanAirspeed: 490,  maxCeiling: 43000));
    data.put("A35K", new AircraftType( type: "A35K",  manufacturer: "Airbus",  family: "A350",  minCleanAirspeed: 250,  maxCleanAirspeed: 490,  maxCeiling: 41000));
    data.put("A388", new AircraftType( type: "A388",  manufacturer: "Airbus",  family: "A380",  minCleanAirspeed: 250,  maxCleanAirspeed: 520,  maxCeiling: 43000));

    data.put("B744", new AircraftType( type: "B744",  manufacturer: "Boeing",  family: "B747",  minCleanAirspeed: 250,  maxCleanAirspeed: 510,  maxCeiling: 45000));
    data.put("B748", new AircraftType( type: "B748",  manufacturer: "Boeing",  family: "B747",  minCleanAirspeed: 250,  maxCleanAirspeed: 510,  maxCeiling: 43000));
    data.put("B752", new AircraftType( type: "B752",  manufacturer: "Boeing",  family: "B757",  minCleanAirspeed: 250,  maxCleanAirspeed: 470,  maxCeiling: 42000));
    data.put("B763", new AircraftType( type: "B763",  manufacturer: "Boeing",  family: "B767",  minCleanAirspeed: 250,  maxCleanAirspeed: 460,  maxCeiling: 45000));
    data.put("B764", new AircraftType( type: "B764",  manufacturer: "Boeing",  family: "B767",  minCleanAirspeed: 250,  maxCleanAirspeed: 460,  maxCeiling: 45000));
    data.put("B772", new AircraftType( type: "B772",  manufacturer: "Boeing",  family: "B777",  minCleanAirspeed: 250,  maxCleanAirspeed: 480,  maxCeiling: 43000));
    data.put("B77L", new AircraftType( type: "B77L",  manufacturer: "Boeing",  family: "B777",  minCleanAirspeed: 250,  maxCleanAirspeed: 490,  maxCeiling: 43000));
    data.put("B77W", new AircraftType( type: "B77W",  manufacturer: "Boeing",  family: "B777",  minCleanAirspeed: 250,  maxCleanAirspeed: 490,  maxCeiling: 43000));
    data.put("B788", new AircraftType( type: "B788",  manufacturer: "Boeing",  family: "B787",  minCleanAirspeed: 250,  maxCleanAirspeed: 470,  maxCeiling: 43000));
    data.put("B789", new AircraftType( type: "B789",  manufacturer: "Boeing",  family: "B787",  minCleanAirspeed: 250,  maxCleanAirspeed: 490,  maxCeiling: 43000));
    data.put("B78X", new AircraftType( type: "B78X",  manufacturer: "Boeing",  family: "B787",  minCleanAirspeed: 250,  maxCleanAirspeed: 488,  maxCeiling: 41000));

    return data;
}
```

It is important to note that although the aircraft type “Concorde” is shown here, it was never used in the final simulation. Instead, it was a fallback aircraft, in the event that the software could not assign a valid aircraft type to a flight across the Atlantic, Concorde was assigned because it would give an obvious, detectable error, which is helpful, instead of the software crashing. This is very beneficial, as software crashes can be challenging and time-consuming to analyse.

2.2 Simulation Engine Development

With all the data collated and created, the additional software development process can take place. Since the RADAR software was largely pre-existing, only a small amount of

additional code had to be written, most of which was to do with creating air traffic in a specific way and then managing that traffic once it had been created.

The simulation engine needs to handle aircraft in two distinct scenarios: the Organised Track System (OTS) and Zero Tracks. Additionally, the software needs to be easily configurable, with a simple setting to operate with, or without the organised tracks.

When aircraft cross the Atlantic without the OTS, they will fly directly between the Oceanic airspace's entry/exit points. Due to a limitation in the RADAR software, they can only fly along a rhumb line, not great circles, so there is a slight inefficiency in the route they will plot.

2.2.1 Aircraft Creation Algorithm

Since the simulations conducted only monitor aircraft over the North Atlantic Ocean, aircraft need to be created (spawned) at the start of the oceanic track and then despawned at the end. In terms of this simulation, the route/flight plan before and after these points is completely irrelevant. However, the aircraft's departure and arrival aerodromes are used to decide which North Atlantic Track they use to cross the Atlantic when simulating with the Organised Track system.

Due to the complexity of a potential algorithm to match the latitude and longitude coordinates of a possible departure airport to the closest oceanic airspace entry point (same goes for destination and exit point) when using the zero track system, the arrival and destination airports are not computed inside the RADAR software. Instead, the zero track routes will be compared to the routes used during the use of the OTS. An approximation will be asserted, stating that a potential Atlantic crossing would be suitable for an aircraft operating between two airports.

In terms of actual traffic simulation, there are a few necessary features to ensure the final data is as realistic as possible:

1. Traffic should not enter the oceanic airspace from the North American and European sides at the same time. Instead, traffic should mimic real-world operations by flying from North America to Europe between the hours of 1800 GMT and 0600 GMT, and fly from Europe to North America for the other 12 hours of the day.

2. Traffic should be intelligently created at the entry points of oceanic airspace, ensuring that all traffic spacing rules are enforced. This means that,
   (a) For OTS operations, traffic should be spaced by 60 nm of lateral separation,
   (b) Traffic should be spaced by 20nm of lateral separation for zero track operations.
3. There should not be a constant flow of traffic over the oceanic airspace, there should be some variation with the time of day.

With each of the three requirements listed above, the aircraft creation algorithm was constructed.

The first requirement was met using a reasonably straightforward method. The RADAR engine's internal clock system, called SkyTime was used to define several key sections of the simulation. For simplicity, it was decided that SkyTime 0 should represent midnight GMT and that the simulation would run for a total of 48 hours. Therefore, boundaries were set inside of the RADAR software's SkyTime so that:

- For the first 6 hours, the aircraft would fly from North America to Europe (00:00 - 06:00),
- For the next 12 hours, the aircraft would fly from Europe to North America (06:00 - 18:00),
- For the next 12 hours, the aircraft would fly from North America to Europe (18:00 - 30:00),
- For the next 12 hours, the aircraft would fly from Europe ot North America (30:00 - 42:00),
- For the final 6 hours, the aircraft would fly from North America to Europe (42:00 - 48:00).

This represents the whole 48-hour simulation time. With this logic implemented, it is ensured that aircraft fly the correct routes at the correct time of day.

It is important to note at this point that the internal SkyTime parameter is scalable. This means that at 60 times normal speed, a full 48 hours can be simulated in 48 minutes. At 600 times normal speed, 48 hours could be simulated within 4.8 minutes. Due to the software still being early in development (and therefore can be prone to small discrepancies), it was found that at anything beyond 100 times normal speed, the aircraft no longer behave properly. Therefore, all simulations were run at 60 times normal speed.

The second requirement is a piece of code that acts as an air traffic controller for the unsimulated airspace. The algorithm is very simple. When attempting to create an aircraft at an oceanic entry point, it first checks to see if another aircraft is already in range of the point it wishes to enter at; so in the case of OTS operations, less than 60 nm and in the case of zero track operations, 20 nm.

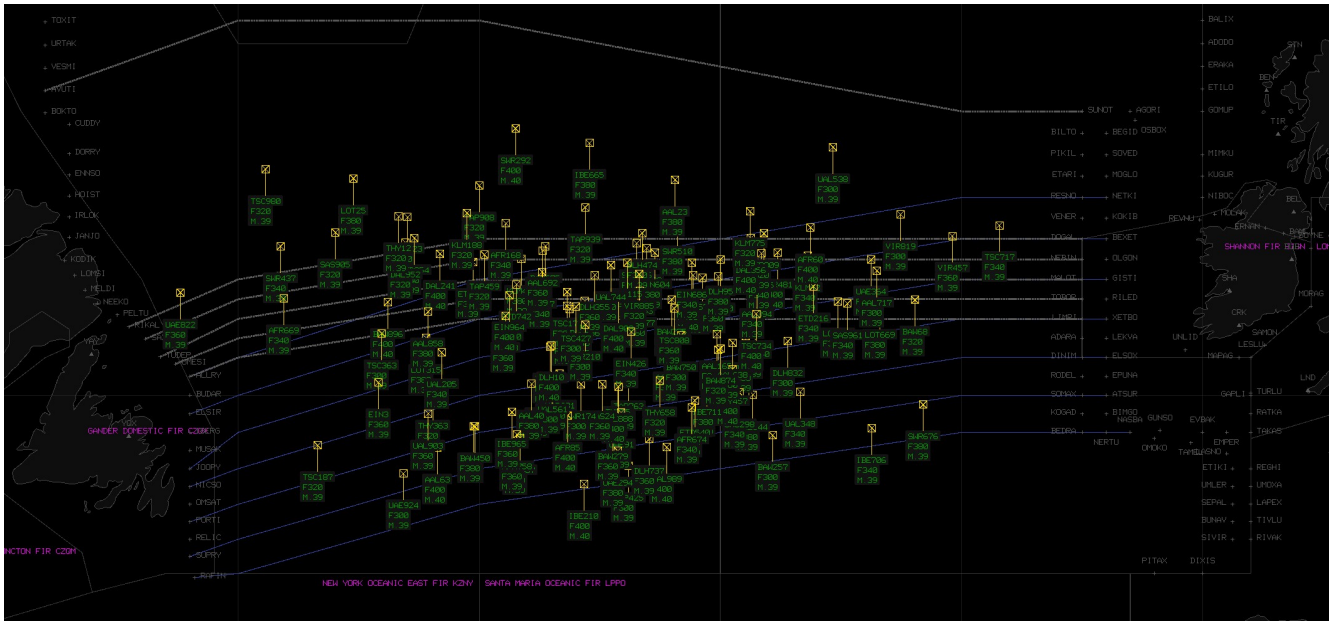
Unfortunately, the code implementation of that was not as simple as it sounds, keeping track of potentially dozens of aircraft and the points they were just created at can be very resource-intensive on a system if not done properly. Therefore, great care had to be taken to make sure that aircraft were being stored in a sensible data structure, and then removed from it when they were further than the minimum separation distance from another aircraft behind them. This was achieved by tracking not in terms of aircraft, but in terms of the static entry points. Whenever an aircraft is created at an entry point, it checks to see if another aircraft is already in range. If not, the aircraft is successfully created. In the event that another aircraft is in range, the code computes the distance between the aircraft that was in range of the point, to see if it still is. If the aircraft is no longer in range of the point, it marks the point as clear next time an aircraft spawn is attempted. This method ensures that the resource-intensive distance check calculation is only computed when the simulation requires it.

Finally, to have traffic operate in a realistic manner (not a constant flow of traffic at all times of day), a normal distribution was used to spawn aircraft in the different phases, while also keeping in mind a total maximum number of aircraft that can operate over the 48-hour period. This maximum value will be tweaked and changed at a later date to see how both systems of air traffic management cope under different workloads. Each of the 5 phases in the bullet points above can be seen in Figure 2.9, each showing a clear bell-curve (normal

distribution) for aircraft being created in the different phases of the simulation.

Figure 2.9

This figure shows air traffic on the RADAR software's display. A clear normal distribution can be seen in the traffic density, with infrequent traffic at the start followed by a high density region of traffic in the middle and then a low density period at the end.



2.2.2 Traffic Separation Algorithm

As previously stated, in OTS operations and zero track operations, there are various minimum separation requirements which must be adhered to at all times. While the "air traffic controller" will ensure that no aircraft can be within the minimum separation distance when entering oceanic airspace, naturally, faster aircraft will catch up with slower flying ones. This poses a problem - do we make all aircraft slow down, or provide a capacity for aircraft to overtake one another?

The latter option is most favourable for almost all circumstances, simply because different aircraft operate optimally at different speeds, therefore burning less fuel on their trans-Atlantic crossing. Therefore, when entering oceanic airspace, a separate process will take place, sorting aircraft's altitude; slower aircraft fly lower, and therefore faster aircraft fly higher. For both OTS operations and zero track operations, minimum vertical separation is 1000ft, so increments of that value will be used to store aircraft. Furthermore, the

semi-circular rule can be employed, so aircraft flying east will do so at an odd altitude, and aircraft flying west will do so at an even altitude. Figure 2.10 illustrates the semi-circular rule.

Implementing this altitude-based separation system will solve most conflicts before aircraft get too close, especially for OTS operations. However, there also must be a system to monitor potential conflicts and take corrective action if needed; to do this, a primitive version of the Traffic Avoidance Collision System, or TCAS, was implemented into the RADAR software. In reality, TCAS is an emergency system that will take corrective action to avoid a mid-air collision between aircraft. It was chosen to mimic this functionality as ultimately any air traffic controller would have to take a similar course of action to prevent conflicts, whether using the OTS or zero tracks, the key difference is that TCAS activation in reality is a last-resort emergency action, when an air traffic controller should make such a corrective action in plenty of time.

In terms of the RADAR software, each aircraft essentially operates with a bubble around it; in the event that different aircraft's bubbles begin to intersect, they will take corrective action to avoid a collision. Importantly, the threshold for taking corrective action is highly configurable, so it can be modified to mimic keeping OTS separation requirements and zero track separation requirements. In addition, the RADAR software is capable of tracking the number of collisions per aircraft. Therefore, it can be studied which method of air traffic management requires more interventions for an air traffic controller. This will be one of the key parameters that are studied for the two methods of air traffic management.

2.3 Weather Simulation

Weather remains an unpredictable element in aviation, even today, micro-bursts cause a significant danger to aircraft on final approaches. While weather is not as dangerous at thirty thousand feet and above (typical minimum altitude for trans-atlantic crossings) it can still have a major impact on aircraft efficiency. Headwinds can cause significant delays, while tailwinds can carry aircraft across the ocean much faster. In fact, each day the North Atlantic tracks are selected in part due to the weather prediction for that day, the goal is to provide aircraft with the best possible path over the ocean. However, there is still a significant degree of randomness when it comes to weather, take micro-bursts as a prime example. Simulating this

randomness is non-trivial when it comes to computer software.

When attempting to solve this problem, there were many different outcomes considered:

1. A constant wind direction and speed throughout the whole simulation. This was ultimately not implemented due to the unrealistic nature of the weather being simulated.

2. Implement weather simulation based upon real world weather data published by the Met Office. This approach was considered and ultimately attempted, however it became apparent that it was simply too challenging and time consuming to implement into this project.

3. Make no attempt to simulate the weather at all. Ultimately, this was the option selected. Simply but, this means the final simulation will not be flawless in terms of accuracy. However, it was decided that subjecting aircraft to all the same conditions (no weather) would provide a fairer result than subjecting aircraft in only one part of the sky to a weather condition at a constant rate, as described in the first bullet point.

The lack of weather simulation will be considered and discussed in relation to the final results, which will be discussed in Chapter 4.

2.4 Calibration

The RADAR software is an experimental piece of equipment. While the level of confidence is high, there have been small bugs that have needed to be patched throughout the course of this project. Therefore, data produced by the software must be verified by real-world sources. This is another reason why real-world traffic data was analysed for this project. This means that each simulated aircraft can be compared with a real aircraft to make sure that the flight times are roughly similar, for the same flight types along the same route. Once this had been confirmed, the complete experimental testing process could begin.

2.5 Experimental Procedure

Once the RADAR software was finalised and ready for testing, the experimental procedure was conducted. Several different traffic simulations were completed for both

systems of air traffic management. The significant difference between the different simulations (other than method of air traffic management) was the number of aircraft simulated. In 2023, 580000 (ForeFlight, n.d.) aircraft made the crossing, so this means that approximately 1589 aircraft operated per day.

Therefore, the simulation was conducted for 1589 aircraft each day, for both OTS and zero tracks and then 10% more and less than this value for both systems of air traffic management. From these simulations, the data relating to each aircraft will be collected, most importantly the simulated time for each aircraft as well as the number of correcting actions the aircraft experienced to maintain minimum separation. The results from these experiments will be found in the following chapter.

Each of these simulations will display information regarding how aircraft and the airspace itself performed in each scenario. The main metric that will be investigated for aircraft is their flight time to cross the Atlantic, whereas the main metric measured for the airspace is the total number of conflicts between aircraft; a conflict would require manual intervention by an air traffic controller. Remember, the simulation defines a conflict by an aircraft's TCAS being activated and then avoiding action taking place to resolve the incident. In reality, TCAS is a system used in emergency situations only, not for traffic separation; therefore, the TCAS sensitivity was turned up, so it essentially functions as an air traffic controller, separating aircraft to ensure they comply with all regulations.

Based on flight times for the aircraft flying between similar or the same destinations, a comment on fuel efficiency can be made (a shorter flight time means less fuel).

3 Results

The results will be divided into six separate scenarios, each operating with different simulation parameters. The simulations conducted will show traffic for both the Organised Track System, and the zero track system of air traffic management at three levels of traffic density each: 100% density, 1589 aircraft per day, 110% density, 1748 aircraft per day and 90% density, 1430 aircraft per day. Remember, each simulation lasts for 48 hours and therefore these numbers will be doubled.

The actual number of aircraft simulated in each scenario may not be exactly equal to the number indicated; this is due to the normal distribution spawning pattern of aircraft having some variation. However, the number of aircraft spawned is no more than 10 different for each scenario. Each scenario was run at 60 times normal speed, so it took approximately 48 minutes to complete.

Due to the fact that thousands of aircraft were simulated in each scenario, the full results will not be presented here, only a subsection. The full tables will be available in Appendix C.

3.1 Scenario 1: OTS System, 100% Traffic Density

The distribution of aircraft entering oceanic airspace for Scenario One can be seen below in Figure 3.1.

Figure 3.1

This figure shows the normal distribution of aircraft entering oceanic airspace in Scenario 1 over a 48 hour period.

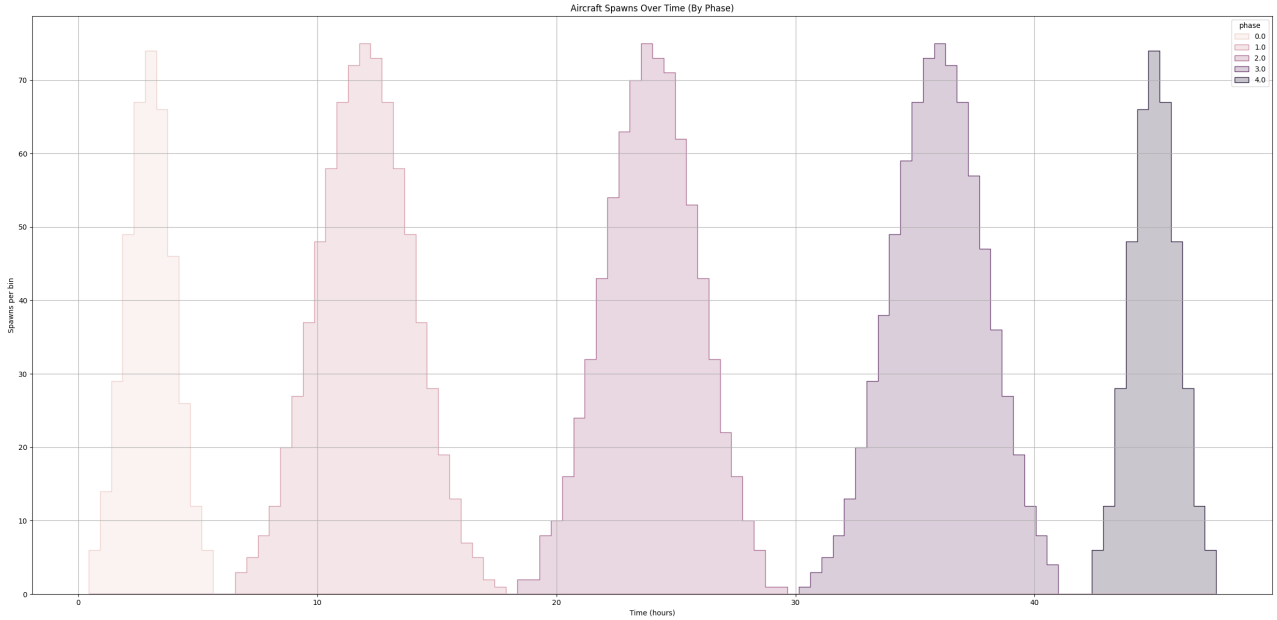


Figure 3.1 is useful as it provides clarification that the aircraft was created in a realistic and expected way by the software. It can easily be observed that the busiest time periods are in the middle of each oceanic crossing period. Furthermore, the first and final phases of the scenario have peaks that are much narrower; this is because their respective phases are cut short by the 48-hour simulation; they only last for 6 hours, whereas the others last for 48. Therefore it can be expected to see a similar occurrence in the distribution of aircraft entering oceanic airspace in the other scenarios. Table 3.1 shows the average information for the different types of air traffic that made the Atlantic crossing in this simulation. Throughout this 48-period, on 129 occasions, a traffic intervention occurred - preventing two aircraft from colliding.

Table 3.1

This table shows the simulation data for scenario one, for each of the different aircraft types that were included. The crossing time is given in minutes.

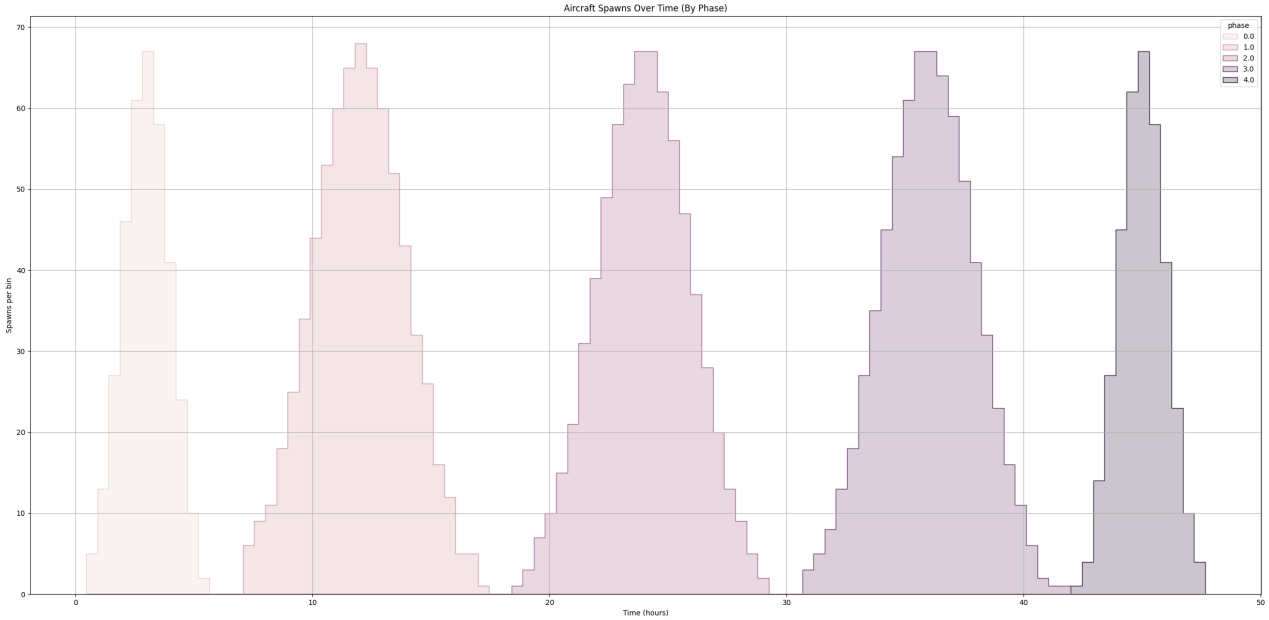
Aircraft Type	Number of Aircraft	Average Crossing Time
B789	277	177.3
B77W	259	176.3
B788	272	185.2
B772	382	181
A359	229	177.4
A333	235	183.4
A332	174	183.9
A339	166	182.3
A35K	132	177.5
B77L	94	175.9
B748	154	169.7
B78X	111	178.3
B763	212	188.3
B744	139	169.6
B764	102	187.6
A388	105	166.5
A343	25	176.5
B752	38	182.4
A346	53	179.2

3.2 Scenario 2: OTS System, 90% Traffic Density

The distribution of aircraft entering oceanic airspace for Scenario Two can be seen below in Figure 3.2.

Figure 3.2

This figure shows the normal distribution of aircraft entering oceanic airspace in Scenario 2 over a 48 hour period.



As expected, for Scenario Two, the peaks are lower than the distribution in Figure 3.1. This is another indication that the simulation was running as expected.

Table 3.2 shows the average information for the different types of air traffic that made the Atlantic crossing in this simulation. Throughout this 48-period, on 124 occasions, a traffic intervention occurred - preventing two aircraft from colliding. The smaller number of conflicts is expected, as the amount of traffic operating has reduced by 10%. However, the small difference observed between Scenarios One and Two suggests that there is not a proportional relationship between conflicts and the amount of air traffic.

Table 3.2

This table shows the simulation data for scenario two, for each of the different aircraft types that were included. The crossing time is given in minutes.

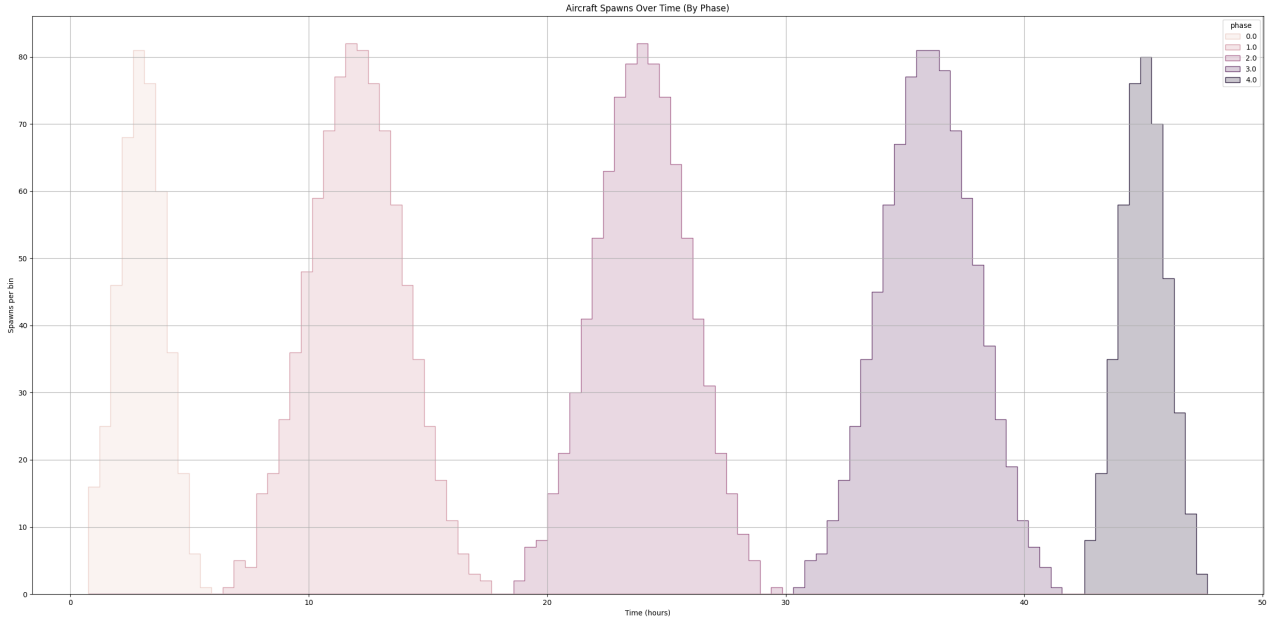
Aircraft Type	Number of Aircraft	Average Crossing Time
B789	259	177.3
B77W	240	176.2
B788	245	185.8
B772	340	180.4
A359	194	176.7
A333	217	182
A332	166	185.3
A339	135	181.6
A35K	106	177.3
B77L	96	177.3
B748	130	170.3
B78X	117	176.7
B763	185	188.7
B744	106	169.6
B764	98	189.3
A388	99	166.9
A343	39	175.7
B752	37	183.4
A346	31	179.4

3.3 Scenario 3: OTS System, 110% Traffic Density

The distribution of aircraft entering oceanic airspace for Scenario Three can be seen below in Figure 3.3.

Figure 3.3

This figure shows the normal distribution of aircraft entering oceanic airspace in Scenario 3 over a 48 hour period.



As expected, in Scenario Three, the peaks are much higher than those observed in Figures 3.1 and 3.2 as there is 10% more traffic than normal operating on the Oceanic Tracks.

Table 3.3 shows the average information for the different types of air traffic that made the Atlantic crossing in this simulation. Throughout this 48 hour period, on 164 occasions, a traffic intervention occured - preventing two aircraft from colliding. This is a significantly higher number of interventions than those reported in Scenario 1, approximately 27% more, again this suggests that there is a complex relationship between traffic density and the number of traffic interventions.

Table 3.3

This table shows the simulation data for scenario three, for each of the different aircraft types that were included. The crossing time is given in minutes.

Aircraft Type	Number of Aircraft	Average Crossing Time
B789	329	177
B77W	299	177.2
B788	285	184.3
B772	443	180.4
A359	200	176.9
A333	267	182.4
A332	168	184.6
A339	176	182.8
A35K	128	176.6
B77L	98	178.3
B748	156	169.8
B78X	125	177.9
B763	245	188.5
B744	158	169.5
B764	149	188.1
A388	152	166.1
A343	23	177.8
B752	26	182.5
A346	46	180.3

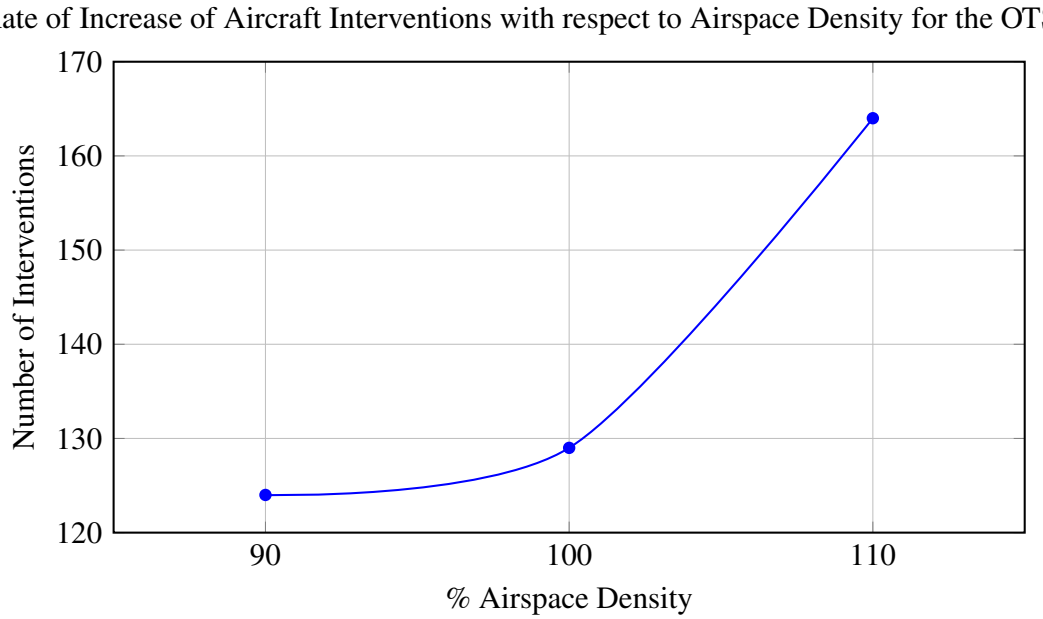
3.4 Summary of Scenarios 1, 2 and 3

Figure 3.4 shows the relationship between airspace density and number of aircraft interventions for the Organised Track System. This relationship appears to be exponential in nature, which ultimately makes sense considering the context, the more aircraft in a given piece of airspace, the higher the higher the likelihood is that it will conflict with something

else.

Figure 3.4

This graph shows the relationship between airspace density and number of aircraft interventions.



Additionally, Table 3.4 shows the percentage difference in the time it takes each aircraft type to fly over the Atlantic ocean, given the different airspace densities.

Table 3.4

This table shows the percentage difference in Atlantic crossing times for the different aircraft types, when operating with the OTS at different airspace densities.

Aircraft Type	Absolute Difference
B789	0.17%
B77W	0.57%
B788	0.81%
B772	0.33%
A359	0.40%
A333	0.77%
A332	0.76%
A339	0.66%
A35K	0.51%
B77L	1.35%
B748	0.35%
B78X	0.90%
B763	0.21%
B744	0.06%
B764	0.90%
A388	0.48%
A343	1.19%
B752	0.55%
A346	0.61%

Ultimately, Table 3.4 shows that the airspace density has a very small to negligible impact on aircraft's flight times across the Atlantic Ocean.

3.5 Scenario 4: Zero Tracks System, 100% Traffic Density

The distribution of aircraft entering oceanic airspace for Scenario Four can be seen below in Figure 3.5.

Figure 3.5

This figure shows the normal distribution of aircraft entering oceanic airspace in Scenario 4 over a 48 hour period.

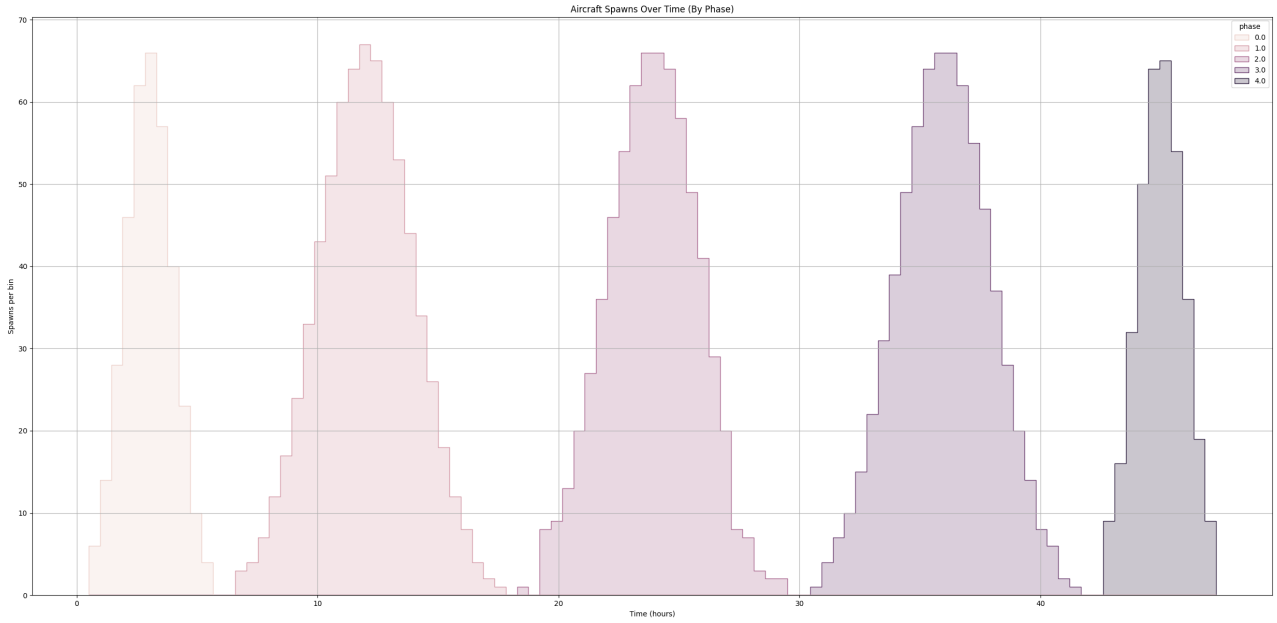


Table 3.5 shows the average information for the different types of air traffic that made the Atlantic crossing in this simulation. According to the software, on 118358 occasions throughout the 48-hour period, a traffic intervention occured - preventing two aircraft from colliding. This number is incredibly high and is likely incorrect due to several factors: When visually inspecting the simulation (while in progress), there are not this many conflicts. If true, there would be a conflict roughly every two seconds (assuming a constant stream of traffic, which is not the case). Unfortunately, very late into the project, a “bug” inside of the RADAR software was encountered, which triggered the logging of a conflict numerous times per conflict. The effect of this bug got exponentially larger as the number of conflicts increased.

Table 3.5

This table shows the simulation data for scenario four, for each of the different aircraft types that were included. The crossing time is given in minutes.

Aircraft Type	Number of Aircraft	Average Crossing Time
B789	372	141.6
B77W	319	147.6
B788	320	150.6
B772	316	158.2
A359	273	141.7
A333	242	144.9
A332	206	149.7
A339	178	156.1
A35K	141	132.6
B77L	127	143
B748	98	158.5
B78X	92	159.8
B763	131	177.7
B744	96	161.5
B764	68	178.6
A388	81	148.3
A343	27	142.9
B752	29	171.2
A346	41	137.8

3.6 Scenario 5: Zero Tracks System, 90% Traffic Density

The distribution of aircraft entering oceanic airspace for Scenario Five can be seen below in Figure 3.6.

Figure 3.6

This figure shows the normal distribution of aircraft entering oceanic airspace in Scenario 5 over a 48 hour period.

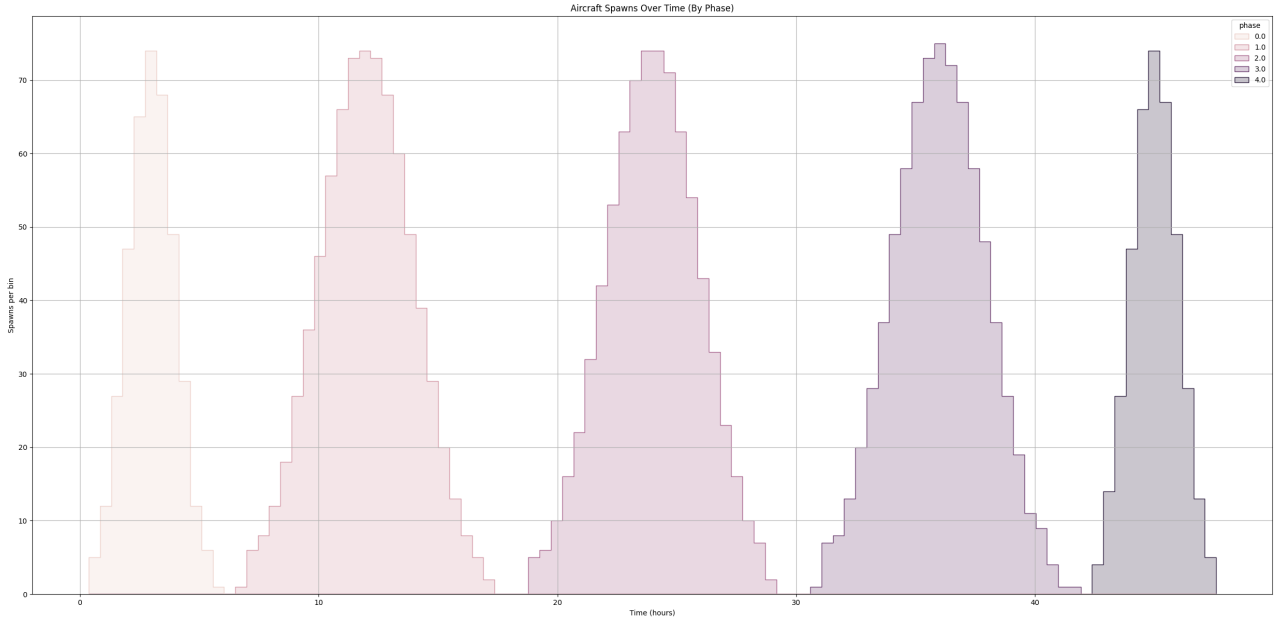


Table 3.6 shows the average information for the different types of air traffic that made the Atlantic crossing in this simulation. According to the software, on 60719 occasions throughout the 48-hour period, a traffic intervention occured - preventing two aircraft from colliding. Clearly, the “bug” discussed in Scenario 4 is still present here. However, the difference can be observed to be significantly smaller. From this it can be inferred that for the zero track system, traffic density has a much larger impact on conflicts compared to the OTS.

Table 3.6

This table shows the simulation data for scenario five, for each of the different aircraft types that were included. The crossing time is given in minutes.

Aircraft Type	Number of Aircraft	Average Crossing Time
B789	267	135.5
B77W	247	128.3
B788	231	133.6
B772	229	164.4
A359	174	129.7
A333	185	139.5
A332	137	142
A339	132	142.3
A35K	106	144.5
B77L	78	134.5
B748	72	160.1
B78X	66	154.1
B763	108	182.1
B744	79	156.2
B764	53	180.9
A388	45	159.4
A343	26	141.6
B752	19	152
A346	23	165.6

3.7 Scenario 6: Zero Tracks System, 110% Traffic Density

The distribution of aircraft entering oceanic airspace for Scenario Five can be seen below in Figure 3.7.

Figure 3.7

This figure shows the normal distribution of aircraft entering oceanic airspace in Scenario 6 over a 48 hour period.

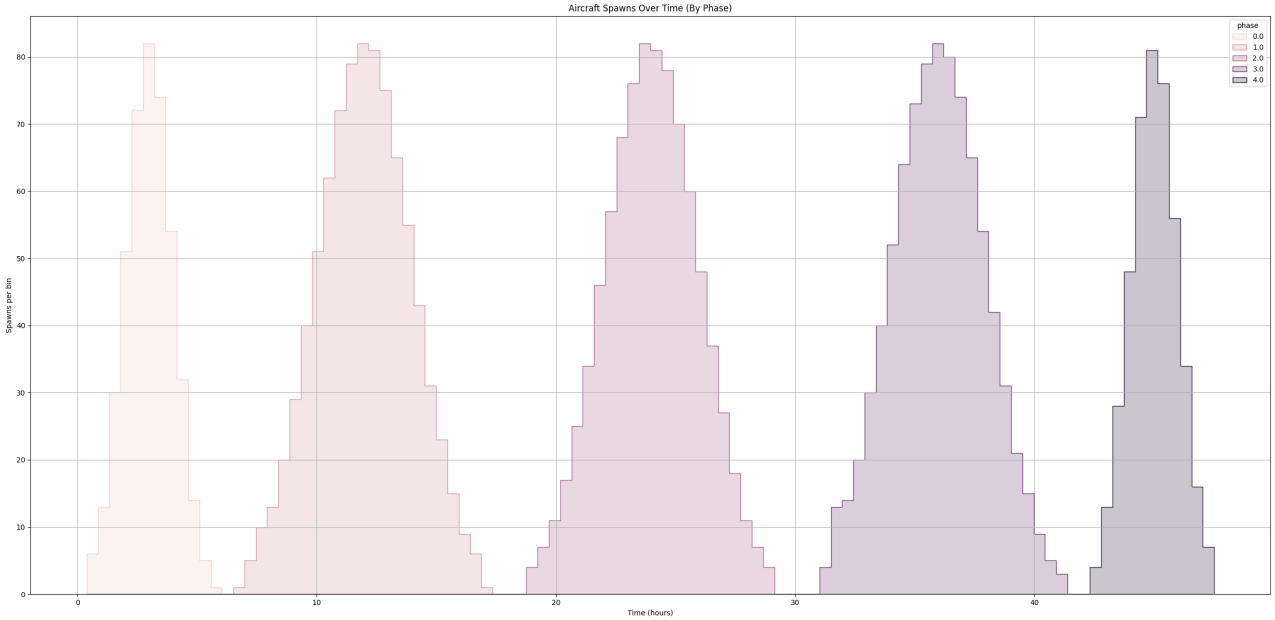


Table 3.7 shows the average information for the different types of air traffic that made the Atlantic crossing in this simulation. According to the software, on 177004 occasions throughout the 48-hour period, a traffic intervention occured - preventing two aircraft from colliding. The same issue is clearly present again. However, it should be noted that the difference between the value observed in Scenario 4 could still be useful.

Table 3.7

This table shows the simulation data for scenario six, for each of the different aircraft types that were included. The crossing time is given in minutes.

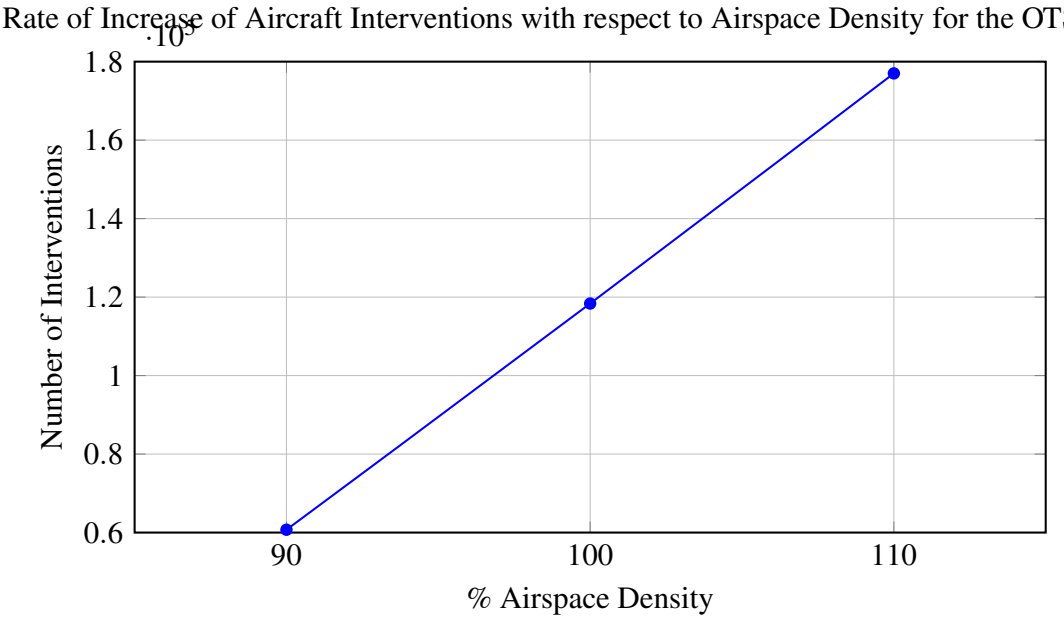
Aircraft Type	Number of Aircraft	Average Crossing Time
B789	426	140.1
B77W	359	140.8
B788	311	154.6
B772	360	156.9
A359	274	136.9
A333	307	150
A332	168	148.3
A339	199	137.7
A35K	169	143.4
B77L	141	142.6
B748	107	158
B78X	113	155.5
B763	154	174.8
B744	102	152.3
B764	82	170.5
A388	85	151.6
A343	42	127.7
B752	33	149.5
A346	42	156.8

3.8 Summary of Scenarios 4, 5 and 6

Figure 3.8 shows the relationship between airspace density and number of aircraft interventions for the Zero Track System. This relationship appears to be linear in nature. However, it must be remembered that the values obtained are clearly incorrect due to the bug in the RADAR software. The linear relationship, however, is useful to be aware of.

Figure 3.8

This graph shows the relationship between airspace density and number of aircraft interventions.



Additionally, Table 3.8 shows the percentage difference in the time it takes each aircraft type to fly over the Atlantic Ocean, given the different airspace densities.

Table 3.8

This table shows the percentage difference in Atlantic crossing times for the different aircraft types, when operating with the Zero Track System at different airspace densities.

Aircraft Type	Absolute Difference
B789	4.39%
B77W	13.89%
B788	14.36%
B772	4.69%
A359	8.82%
A333	7.25%
A332	5.25%
A339	12.66%
A35K	8.49%
B77L	6.07%
B748	1.32%
B78X	3.64%
B763	4.10%
B744	5.87%
B764	5.89%
A388	7.25%
A343	11.06%
B752	13.77%
A346	18.12%

Ultimately, Table 3.8 shows that the airspace density has a somewhat significant impact on the time it takes to cross the Atlantic, with the highest difference being 18.12%.

3.9 Overall Summary of Results

Ultimately, there are several key observations that can be taken away from the simulations:

1. The time taken for aircraft to cross the North Atlantic Ocean is much smaller using the Zero Track system compared to the OTS.

2. Traffic Density has a much bigger impact on flight times for the Zero Track system compared to the OTS.

3. The relationship of interventions to traffic density is exponential for the OTS, but linear for the Zero Track System (although there is a flaw with the software that could have manipulated this relationship).

4 Discussion

Assessing the two different methods of air traffic management is not an easy task, both systems clearly have positives and negatives.

First of all, the OTS is limited by its own design; there can only be so many aircraft on each oceanic track at any given moment in time due to the strict separation requirements. Additionally, the OTS does not tend to follow the most direct path across the Atlantic, in Tables 3.1 and 3.4 (100% traffic density for OTS and zero tracks, respectively), time differences between 20 and 30 minutes for most aircraft can be observed. This is not an insignificant amount of time, both in terms of the environmental impact of burning aviation fuel, but also in terms of extra time that is freed up in the aircraft's schedule. The longer a flight takes, the more fuel it requires. Not only does this extra fuel come at a cost to the environment, but it also impacts those who wish to travel by aircraft; more fuel means an increased ticket price. This is why low-cost airlines charge so much for extra baggage and are so strict on the maximum suitcase dimensions and weight. Making travel by aviation cheaper, and therefore more accessible, will only result in positives for the industry; more customers mean more streams of revenue. For the airlines, shorter flights mean that there is the potential for longer turnaround times at major airports (assuming the airport has capacity for such a thing). More importantly, it adds extra slack in a very tight schedule. If a long turnaround is planned, it means that there is a bigger opportunity to make up time in the event of delays.

However, the Zero track system is also limited as it requires a more "hands on approach" by air traffic controllers. Even though the raw data provided by the simulation is clearly wrong, it is clear that there is a higher number of traffic conflicts when using the zero track system. This assertion can be made because of the nature of the bug, which was previously explained. The more conflicts that are detected, the more out of sync the RADAR software gets, logging the same conflicts an increasing number of times each. However, this is hardly surprising. Aircraft operating between any entry and exit point in oceanic airspace is bound to result in aircraft flying over eachother's paths, resulting in conflicts.

In addition, the simulation produced in the RADAR software is purely reactionary in nature in its approach to dealing with conflicts in the sense that if a problem comes up, it will

deal with it then. A better approach would be to plan aircraft in a sequence that minimises potential route crossings. This simply could not be simulated for this project due to the intense processing power required for such a calculation; it was simply not possible inside the software used. However, it would be very achievable for two large governmental organisations such as NAV Canada and NATS.

The lack of weather simulation cannot be ignored. In reality, the Oceanic Tracks are picked partially to make use of the weather, giving aircraft the most optimum route, usually involving a large tailwind component. That, of course, has not been considered in this simulation, nor has the effect of any potential headwind. This is not an insignificant factor, as some zero track paths may end up taking longer to complete than the North Atlantic Tracks because of the headwind component. Additionally, the aircraft in the RADAR simulation fly along rhumb lines when not following a NAT track; in reality, they would be more likely to follow a great circle, as they are always the shortest possible distance. These two factors combined may result in the Oceanic crossing time for the zero track system to increase, but it is not possible to reasonably quantify how big this impact is.

Outside of the Aircraft Surveillance scope, the introduction of Space-Based ADS-B technology has resulted in the ability to track aircraft anywhere on the globe. This has major implications, mostly in the event of catastrophic failure. Take Malaysian Airlines Flight 370, for example, an aircraft that vanished while operating in a “RADAR dead” part of south-east Asia. With the introduction of Aircraft Tracking, thanks to Space-Based ADS-B technology provided by Aireon, aircraft can be monitored anywhere on the globe. While this will not prevent accidents, like the kind that happened to MH 370, they will aid in the subsequent investigation and therefore provide information that can be learned from going forward.

5 Conclusions

To conclude, there is no doubt that Space-Based ADS-B will revolutionise how air traffic is tracked and surveilled in the years to come. The global coverage achieved by Aireron, thanks to the Iridium Constellation of Satellites, cannot be beaten by any conventional primary or secondary RADAR sources, as aircraft are always in range.

This is useful for airlines and air crash investigators in piecing together traffic movements through aircraft tracking solutions; this will ultimately help airlines to plan more efficiently and give additional insight to investigators in the event of a disaster. Furthermore, as proven by the simulation trials, the zero track system (which can be implemented thanks to Space-Based ADS-B) allows flight times to be reduced by up to 30 minutes (without taking the weather into account). This will save airlines time and consumers considerable amounts of money, whilst also making the aviation industry more environmentally sustainable.

The introduction of zero tracks, however, cannot be rushed, as proven by the simulation, operating a zero track system over the Atlantic will be very resource-intensive for air traffic controllers compared to the Organised Track system. Conventional air traffic services will have to be instituted over the Atlantic to prevent hundreds of potential conflicts each and every day. However, in the context of massive financial and environmental savings for airlines, this seems like the best approach.

5.1 Recommendations

Therefore, based on these conclusions, several recommendations should be made:

1. Zero Track Operations should continue to be trialled by Nav Canada and NATS over the North Atlantic Ocean, and they should receive continuous support from their respective governments, in terms of resources, to assist them in delivering this service.

2. Other Governments should consider implementing this system too. In particular, the FAA should look to implement air traffic management by means of Space-Based ADS-B over the Pacific Ocean, with the aid of Airservices Australia, Airways (New Zealand's air traffic service provider), The Japan Civil Aviation Bureau and others.

3. Airlines should continue to cooperate with one another so that efficient routes can be

planned across airspace that operates without designated tracks, assisting in the work provided by the air traffic surveillance agencies.

4. Prepare for the future of Aviation. For example, Boom (the in-development Supersonic Airliner). When it was in operation, Concorde had its own tracks across the Atlantic Ocean, because of its high cruising speed and altitude. Air traffic management agencies must stay agile and able to implement new strategies for the needs of different types of aircraft.

By implementing these four recommendations, Space-Based ADS-B can become the future of Air traffic management, not just over the North Atlantic Ocean, but across the whole world.

References

Avionix Technologies. (n.d.). The difference between radar and ADS-B. https://www.avionix-tech.com/blog/The-difference-between-Radar-and-ADS-B/

Baker, K. Space-based ads-b: Performance, architecture and market. In: Summit Space Corporation. 5075 Highbourne Lane, Centreville, VA 20120, 2019, April, pg.4G1-1 - pg.4G1–10.

Budroweit, J., Eichstaedt, F., & Delovski, T. (2024). Aircraft surveillance from space: The future of air traffic control? *IEEE Microwave Magazine*, pg.68–pg.76.

Business, A. (2022). How space-based ads-b plane-tracking tech is changing flying across the north atlantic. *Pyramid Media Group, Inc*.

EUROCONTROL. (n.d.). Aircraft performance database [Accessed: March 2025].

Federal Aviation Administration. (n.d.-a). Digital enroute charts (dec) [Accessed: March 2025].

Federal Aviation Administration. (n.d.-b). North atlantic tracks (nat) notams [Accessed: March 2025].

FlightAware. (n.d.). Aeroapi: Flight tracking and flight status api [Accessed: March 2025].

Flightradar24. (2024, April). *How does ads-b work?* Flightradar24. https://www.youtube.com/watch?v=7K1xFb1REHU

ForeFlight. (n.d.). North atlantic tracks. https://ba.foreflight.com/blog/north-atlantic-tracks

Garcia, D. M. A., Stafford, J., Minnix, D., & Dolan, J. Aireon space based ads-b performance model. In: Integrated Communications Navigation and Surveillance (ICNS) Conference. 2015, April, pg.C2-1 - pg.C2–10.

International Civil Aviation Organization. (2007, September). Guidance material on comparison of surveillance technologies (gmst).

International Civil Aviation Organization. (n.d.). Icao flight information regions (fir) gis database [Accessed: March 2025].

Patterson, T., & Kelso, N. V. (2009). Natural earth: Free vector and raster map data at 1:10m, 1:50m, and 1:110m scales [Accessed: March 2025].

QGIS Development Team. (2002). Qgis geographic information system [Accessed: March 2025].

Rodionova, O., Sbihi, M., Delahaye, D., & Mongeau, M. (2014). North atlantic aircraft trajectory optimization. *Intelligent Transportation Systems, IEEE Transactions on*, *15*, 2202–2212. https://doi.org/10.1109/TITS.2014.2312315

SKYbrary. (n.d.). North atlantic operations - airspace. https://skybrary.aero/articles/north-atlantic-operations-airspace

Terminal2Solutions. (2024). T2:controller. https://t2s.aero/products/controller

Young, J. (2021, March). Nats records first day with zero westbound north atlantic tracks.

6 Appendix A

```
import json

def degrees_to_dms_with_direction(deg, is_latitude):
    """
    Converts degrees to DDDMMSS format with directional prefix (N/S/E/W).
    :param deg: Degree value (float or int)
    :param is_latitude: Boolean indicating if the value is latitude
    :return: Formatted string in "DDDMMSS" with directional prefix
    """
    direction = ''
    if is_latitude:
        direction = 'N' if deg >= 0 else 'S'
    else:
        direction = 'E' if deg >= 0 else 'W'

    deg = abs(deg)
    d = int(deg)
    md = (deg - d) * 60
    m = int(md)
    sd = int((md - m) * 60)

    return f"{direction}{d:03d}{m:02d}{sd:02d}"

def convert_coordinates(coords, is_latitude=False):
    """
    Recursively converts coordinates to "DDDMMSS" format with directional prefix.
```

```
27          :param coords: List of coordinates or a single
                coordinate pair
28          :param is_latitude: Boolean indicating if the
                coordinate is latitude
29          :return: Converted coordinates
30          """
31          if isinstance(coords[0], (float, int)):
32              lon = degrees_to_dms_with_direction(coords[0],
                    is_latitude=False)
33              lat = degrees_to_dms_with_direction(coords[1],
                    is_latitude=True)
34              return [lon, lat]
35          elif isinstance(coords[0], list):
36              return [convert_coordinates(coord) for coord in
                    coords]
37          else:
38              raise ValueError(f"Unexpected coordinate format: {
                    coords}")
39
40      # Read GEOJSON file
41      with open('Airspace_Boundary.geojson', 'r') as f:
42          geojson_data = json.load(f)
43
44      # Iterate through features and process coordinates
45      for feature in geojson_data['features']:
46          geometry = feature['geometry']
47          geometry['coordinates'] = convert_coordinates(geometry
                ['coordinates'])
48
49      # Write back to GEOJSON file
50      with open('firBoundaryConverted.geojson', 'w') as f:
```

```
        json.dump(geojson_data, f, indent=2)

    print("Conversion complete. Output written to output file.")
    input("Press Enter to close...")
```

Listing 1: Convert GeoJSON Coordinates (in radians) to DDDMMSS Format with Direction. This script was used as described in Chapter 2.

```
import json

# Function to convert coordinate string to lat/lon
# N/S for latitude, E/W for longitude
def parse_coordinate(coord):
    return coord

def convert_geojson_to_xml(input_file, output_file):
    # Load JSON data
    with open(input_file, 'r') as infile:
        data = json.load(infile)

    # Initialize the output XML content
    output_lines = ["<?xml version=\"1.0\" encoding=\"UTF-8\"?>"]

    for feature in data.get("features", []):
        output_lines.append("<path>")

        # Extract coordinates
        coordinates = feature.get("geometry", {}).get("coordinates", [])
        for line in coordinates:
```

```
                for coord_pair in line:
                    lat = parse_coordinate(coord_pair[1])
                    lon = parse_coordinate(coord_pair[0])

                    if lat is not None and lon is not None:
                        output_lines.append(f"    <point lat
                            =\"{lat}\" lon=\"{lon}\"/>")

            output_lines.append("</path>")

        # Write the output to file
        with open(output_file, 'w') as outfile:
            outfile.write("\n".join(output_lines))

    # Example usage
    convert_geojson_to_xml('firBoundaryConverted.geojson', '
        firBoundaryMap.xml')

    print("Conversion complete. Output written to output file.
        ")
    input("Press Enter to close...")
```

Listing 2: Convert GeoJSON to Simple XML Format

7 Appendix B

```
import requests
import csv
import time
from datetime import datetime, timedelta

# FlightAware API credentials
API_KEY = "API KEY REMOVED FOR PRIVACY"
BASE_URL = "https://aeroapi.flightaware.com/aeroapi"

# Define the bounding box for the region (North Atlantic example)
NORTH_LAT = 65.0   # Northernmost latitude
SOUTH_LAT = 30.0   # Southernmost latitude
WEST_LON = -70.0   # Westernmost longitude
EAST_LON = -10.0   # Easternmost longitude

# Run the script for 24 hours (once per hour)
TOTAL_RUNS = 24
INTERVAL = 3600  # 1 hour in seconds

def get_flights():
    """Fetch flights within a specific latitude/longitude bounding box."""
    flights_data = []
    headers = {"x-apikey": API_KEY}

    url = f"{BASE_URL}/flights/search"

    # Correctly formatted lat/long query
```

```
28          query = f'-latlong "{NORTH_LAT} {WEST_LON} {SOUTH_LAT}
                {EAST_LON}"'
29
30          params = {"query": query, "max_pages": 10}
31
32          try:
33              response = requests.get(url, headers=headers,
                    params=params)
34              response.raise_for_status()
35              flights = response.json().get("flights", [])
36
37              for flight in flights:
38                  callsign = flight.get("ident", "N/A")
39                  aircraft_type = flight.get("aircraft_type", "N
                        /A")
40
41                  # Safe extraction of departure airport ICAO
                        code
42                  departure_airport = flight.get("origin", {}).
                        get("code_icao", "Unknown")
43
44                  # Safe extraction of destination airport ICAO
                        code
45                  if flight.get("destination"):
46                      destination_airport = flight["destination"
                            ].get("code_icao", "Unknown")
47                  else:
48                      destination_airport = "Unknown"
49
50                  # Extract and format departure time (if
                        available)
```

```
51                departure_time = flight.get("
                      actual_departure_time", "N/A")
52                if departure_time != "N/A":
53                    departure_time = datetime.utcfromtimestamp
                         (departure_time).strftime("%Y-%m-%d %H
                         :%M:%S")
54
55                # Append cleaned data
56                flights_data.append([callsign, aircraft_type,
                      departure_airport, destination_airport,
                      departure_time])
57
58            except requests.exceptions.RequestException as e:
59                print(f"Error fetching flight data: {e}")
60
61            return flights_data
62
63
64        def save_to_csv(flights, timestamp):
65            """Saves flight data to a uniquely named CSV file."""
66            filename = f"flights_log_{timestamp}.csv"
67
68            with open(filename, mode="w", newline="", encoding="
                  utf-8") as file:
69                writer = csv.writer(file)
70                writer.writerow(["Callsign", "Aircraft Type", "
                      Departure", "Destination", "Departure Time (UTC
                      )"])
71                writer.writerows(flights)
72
73            print(f"Flight data saved to {filename}")
```

```
74
75
76      def main():
77          """Runs the flight data collection every hour for 24
                hours."""
78          for i in range(TOTAL_RUNS):
79              print(f"Fetching flight data... Run {i+1} of {
                    TOTAL_RUNS}")
80
81              flights = get_flights()
82
83              if not flights:
84                  print("No flights found in the specified
                        region.")
85              else:
86                  # Generate timestamped filename
87                  timestamp = datetime.utcnow().strftime("%Y%m%
                        d_%H%M%S")
88                  save_to_csv(flights, timestamp)
89
90              if i < TOTAL_RUNS - 1:  # Don't sleep after the
                    last run
91                  print("Waiting for 1 hour before next run...\n
                        ")
92                  time.sleep(INTERVAL)  # Wait for 1 hour
93
94          print("Completed 24-hour flight recording.")
95
96
97      if __name__ == "__main__":
98          main()
```

Listing 3: 24-Hour Flight Data Collection using FlightAware API