Cloud-based Air Traffic Control

Introduction

With the advent of drones and the forecast of robust growth in the aviation industry, the skies are going to overly crowded in the future. As new regulations for safety are being mandated to address the growing needs, it is time to redesign archaic Air Traffic Control systems and bring them into the 21st century.

This document illustrates a working prototype of a modern Air Traffic Control that will scale to the needs of the burgeoning aviation industry using latest paradigms and technologies such as Internet of Things (IoT), Software Defined Radio (SDR), Augmented Reality (AR), etc. without significant capital expenditure.

Overview

The current Air Traffic Control systems rely on Ground Stations with outdated technologies such as Primary Surveillance Radar (PSR). PSR involves a large dish that spins around and sends high-powered pulses and waits for reflected responses from the aircraft. These responses are then relayed to the Air Traffic Control to calculate the aircraft's position using radar scopes. However, PSR cannot uniquely identify an aircraft and it's not as effective if the range increases or the weather deteriorates. Furthermore, the current day Air Traffic Control requires a human to inform the pilot about the air traffic, the weather conditions, and the terrain.

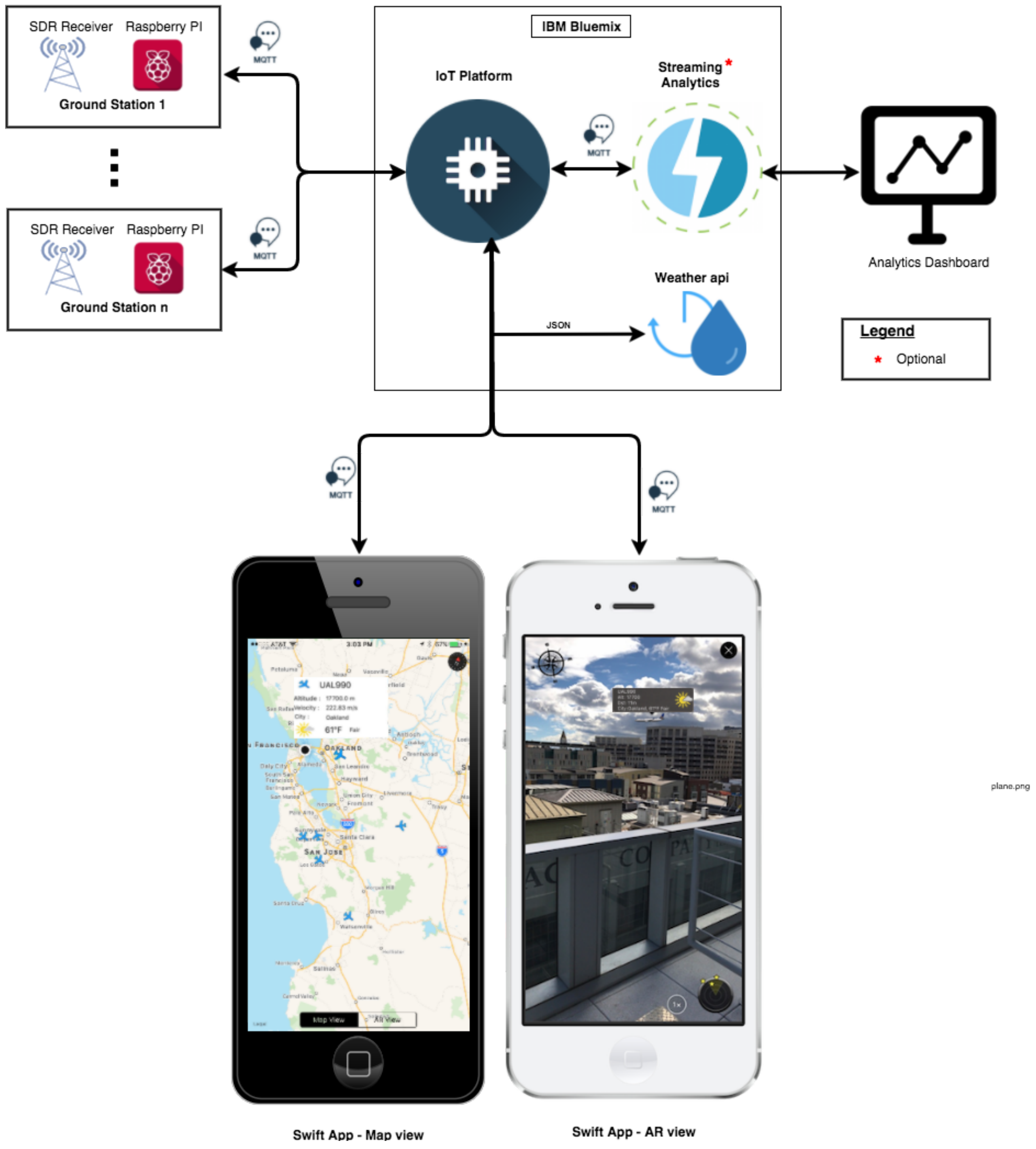
To address these issues, a new surveillance technology called Automatic Dependent Surveillance - Broadcast (ADS-B) has been developed which enables an aircraft to determine its own position via satellite navigation and periodically broadcast it on 1090 MHz radio frequency, so that it can be tracked. The Ground Stations can receive ADS-B messages on the specified frequency, instead of reflected responses and radar scopes, to determine not only the aircraft's position but also other useful information such as call-sign, velocity, altitude, heading, etc. Furthermore, aircrafts can receive ADS-B messages from other aircrafts in the vicinity to automatically determine air traffic. ADS-B also provides support for aircrafts/pilots to figure out weather and terrain conditions without needing any human intervention.

Under FAA regulation 14 CFR 91.227, all commercial flights must be equipped with ADS-B avionics equipment by January 2020 to be able to fly in the majority of the US airspace. This presents a great opportunity to lay the groundwork for building Air Traffic Control system for the 21st century. The proposed Air Traffic Control system would rely on network-connected ADS-B compliant Ground Stations and can scale to handle the projected exponential increase in aircraft traffic volumes which would make flying not only safer but also significantly cheaper.

Architecture

Figure 1 shows the high-level architecture of a Cloud-based Air Traffic Control that relies on cheap Ground Stations to track flights:

Figure 1: Air Traffic Controller



With the advances in the field of avionics and the availability of cheap computing resources such as Raspberry Pi (RPi), one can very easily build a state-of-the-art Ground Station. These Ground Stations can be replicated trivially using virtualization technologies such as Docker to be able to cover large swathes of areas. The RPi-powered Ground Stations, scattered all over the world, will do the following:

- Use a SDR receiver with an antenna to receive information about flights that are in approximately 100-150 miles radius depending on the altitude and the line-of-sight.
- Act as network-connected IoT devices to publish the flight information as Message Queuing Telemetry Transport (MQTT) messages to a Cloud-based Air Traffic Control running in scalable, secure, and, reliable, and open cloud infrastructure.

The Cloud-based Air Traffic Control can be implemented using IBM’s Bluemix Cloud Platform-As-A-Service (PaaS) which is an implementation of IBM’s Open Cloud Architecture based on CloudFoundry open technology and based on SoftLayer infrastructure. Since the Ground Stations are modeled as IoT devices that are network-connected and send flight information as MQTT messages, it makes sense to use the *Internet of Things(IoT) Platform* service within IBM Bluemix as it can not only scale elastically with the number of Ground Stations but also serve as funneling point to receive all the events so that one can compose analytics applications, visualization dashboards, etc. using the flight data.

IoT Platform service will also be able to serve the flight information to all the iOS devices that are connected to it. A Swift-based mobile app running on an iOS device can use Augmented Reality to render flights that are headed in that direction on the screen -- before they show up outside one’s window!

The following sections try to zoom into the three main aspects of tracking flights in real-time -- RPi powered ADS-B Ground Station, Air Traffic Control in IBM Bluemix, and Swift-based iOS app to render flights using AR.

IoT Devices - Raspberry Pi Powered ADS-B Ground Stations

SDR devices such as NooElec's RTL-SDR receiver set with antenna, which support USB 2.0 interface, can be used to receive ADS-B messages that are being broadcasted in clear text by commercial flights, within a decent radius, on practically any device with a USB port.

The hardware configuration of an ADS-B compliant Ground Station built using a Raspberry Pi 3 and a NooElec's RTL-SDR receiver set with antenna is shown in Figure 2:

Figure 2: Hardware Configuration of ADS-B compliant Ground Station

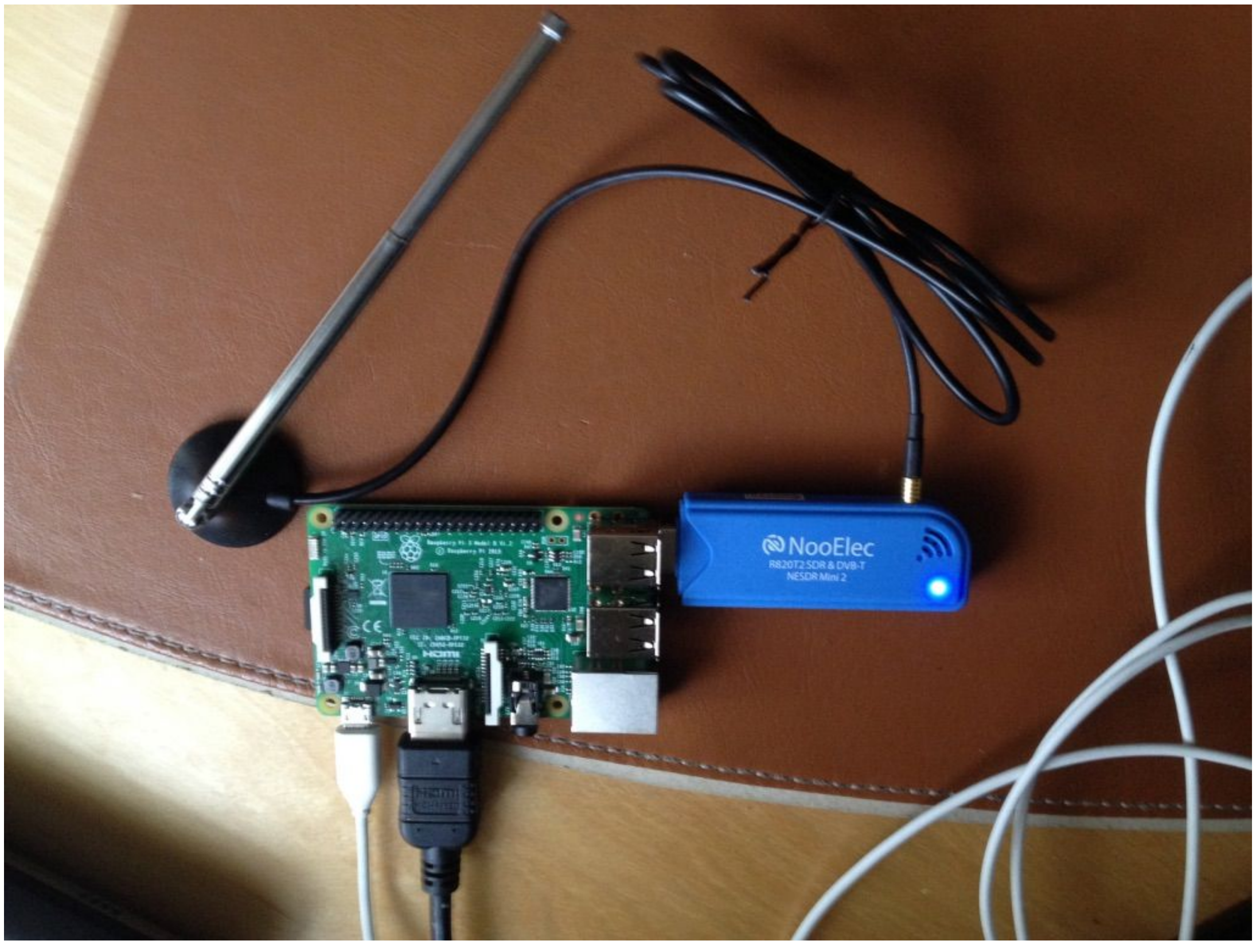
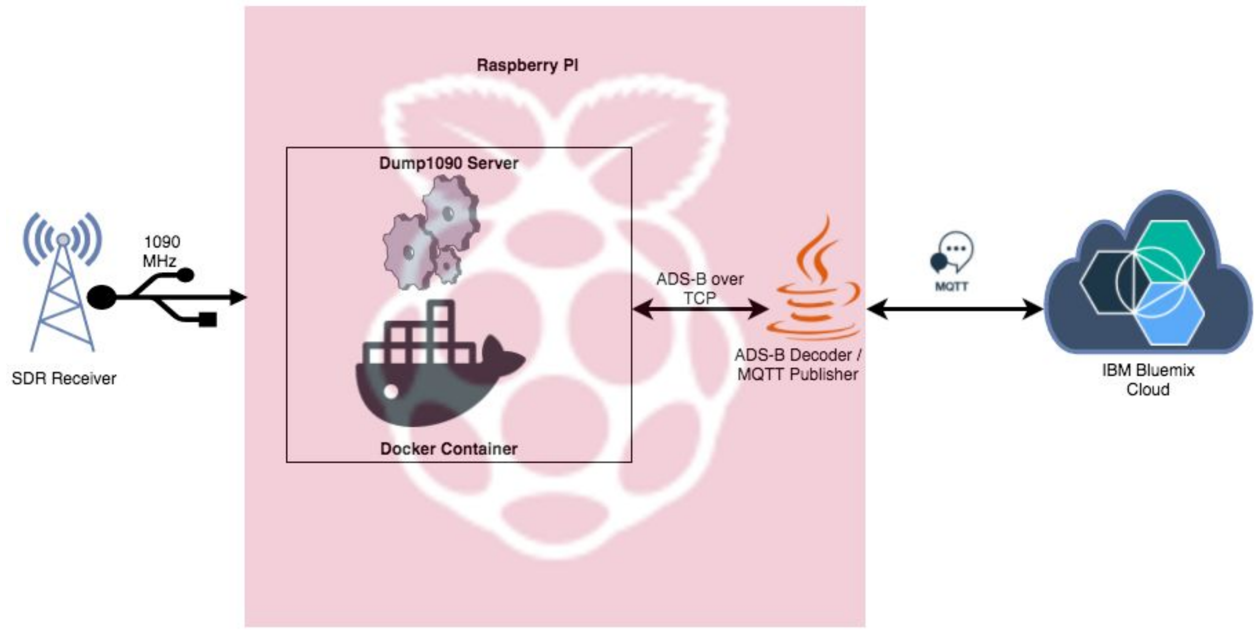


Figure 3 shows the anatomy of an ADS-B compliant Ground Station that serves as an IoT device sending MQTT messages by decoding the ADS-B messages received from the SDR:

Figure 3: ADS-B Ground Station as an IoT Device



ADS-B compliant Ground Station will do the following:

- *receive* ADS-B messages broadcasted by commercial flights by tuning the SDR receiver to 1090 MHz frequency
- *decode* ADS-B messages from commercial flights
- *publish* corresponding MQTT messages to IoT Platform service running in IBM Bluemix.

On a Raspberry Pi, a Docker container running *Dump1090 Server* is used to *receive* ADS-B messages from the SDR connected to a USB port and tuned to 1090 MHz frequency. *Dump1090 Server* acts as a conduit for ADS-B messages and makes them available to any TCP client connected to port 30002.

On the same Raspberry Pi, a Java application -- acts as a TCP client one one side and connects to the local port 30002 to *decode* the incoming ADS-B messages containing flight information such as latitude, longitude, call-sign, altitude, heading, velocity, etc.; and acts as a MQTT client on the other side as it connects to IoT Platform service running in the IBM Bluemix to *publish* MQTT messages with JSON payload containing the aforementioned decoded attributes.

Detailed instructions on how to successfully build a Raspberry Pi powered ADS-B Ground Station are available in the github repository.

Air Traffic Control in IBM Bluemix

The modern-day Air Traffic Control can be built using secure, scalable, reliable, and open cloud platform such as IBM Bluemix. IBM Bluemix provides rich set of services that can be used to create interesting applications using the flight information being published from various ADS-B Ground Stations.

Even though IBM Bluemix provides a rich set of services, the only service that is needed for the Air Traffic Control to work is the IoT Platform service. The system can be augmented with other services based on the needs.

IoT Platform Service

IoT Platform service can be created in IBM Bluemix as described here. This service is required for the proposed Air Traffic Control to function. Using the IoT Platform dashboard, an IoT device-type and IoT device can be provisioned. Also, the developer can generate the *API-Key* and the *Authentication Token* for the service. This information is used by the ADS-B Ground Stations to connect to the IoT Platform service running in IBM Bluemix.

IoT Platform service aggregates the flight information that is being sent by all the ADS-B Ground Station. It can scale easily as more and more ADS-B Ground Stations get activated. Since various services within IBM Bluemix are tightly integrated, the flight information can be persisted to the IBM Cloudant database running within IBM Bluemix. Or, the flight information can be made available to a Streaming Analytics service, once again running within IBM Bluemix, to analyze and present information in interesting ways.

Streaming Analytics Service

Optionally, developers can add Streaming Analytics service to the mix. Streaming Analytics service can be setup to receive all the IoT device events from the IoT Platform service. This way, the payload of the MQTT messages that are being sent by the ADS-B Ground Stations will be available in the Streaming Analytics service. As an example, this data can be used to visualize flights leaving one ADS-B Ground Station and being picked up by another ADS-B Ground Station.

Weather API Service

Developers can also add Weather API services to the mix. Weather api can be setup from bluemix console and using the credentials provided, applications can use api to access weather based on locations using gps coordinates (longitude and latitude) or cities.This will give user the ability to figure out the weather based on flight location and also the weather on arriving and destination airport of the flight.

Swift based iOS app

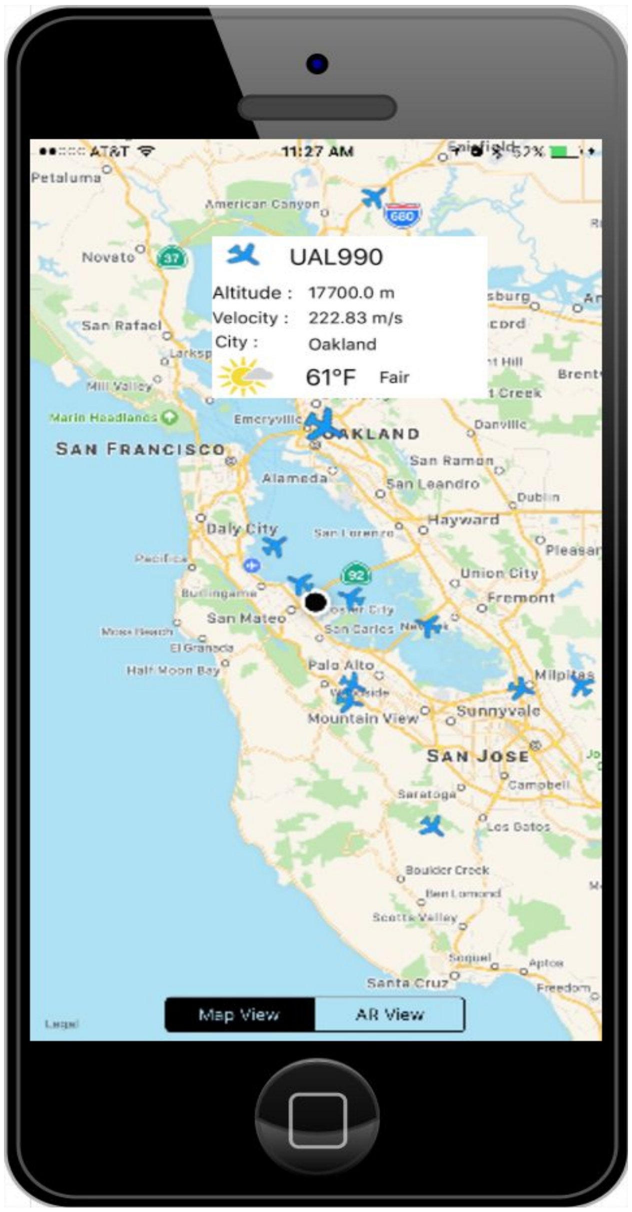
Swift-based iOS app connects to the IoT Platform and receives MQTT messages whose JSON payload includes the coordinates (longitude and latitude) and other useful information that can be used to render the flights on either a simple map or use AR to correlate the data with the flight that is being seen on the screen in the camera view.

The source code and the instructions to run the app are available in the github repository.

Map View

Map View displays all the flights on a default map provided in the iOS device. The flight orientation is adjusted based on the current heading information in the payload. As the app receives MQTT messages, the flight will be seen moving in the direction towards its destination. A flight can be tapped to see more details such as such as flight number, altitude, distance etc. Figure 4 below shows the rendering of the Map View with flights in the Swift-based app on an iOS device:

Figure 4: Map View



UAL990

Altitude : 17700.0 m

Velocity : 222.83 m/s

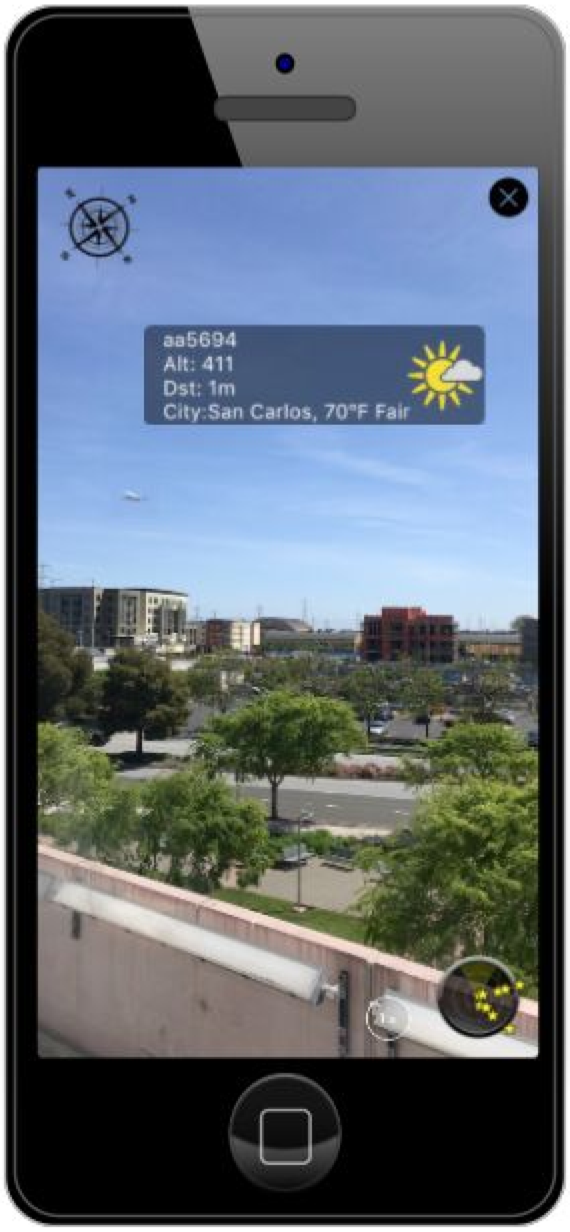
City : Oakland

61°F Fair

Augmented Reality View

The user can tap the *AR View* tab in the app to switch to the AR-based View. In this mode, the app opens up a camera view where the user can point to a flight to be able to see the flight data on a callout that overlays on top of the flight in the real world. As the flight is moving in the real world, the callout with the information moves along with the flight in the camera view. The AR view also displays a compass based on the device heading and a radar view displaying all the flights within the viewing angle. Figure 5 below shows the rendering of the Augmented Reality View with flights in the Swift-based app on an iOS device.

Figure 5: Augmented Reality View



Reference Links

- [Rendering 3D scene in 2D canvas]
- [Sample augmented reality swift based app]
- [AR engine for iOS device]
- [AR toolkit]