Intelligent low-altitude air traffic management system

DESIGN DOCUMENT

Group 30
Prof. Peng Wei
Humaid Alkaabi
Jun An Tan
Saad Alsudayri
Suhail Aldhaheri

Revised: 30 Nov 2017-Version 2

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

Our group is working with professor Peng Wei to create a system that manages air-traffic in low altitudes. The system will be a simulation interface, a software that simulates the delivery process using drones in 2D where you have an x number of drones fulfilling an ever growing demand over time. Similar to a flight tracker for commercial planes, our software will display and update each specific drone movement at any point of time. One of the main key features is to make sure that drones will not collide into each other at any point of time. Our project, being software based requires little to no hardware utilities. Therefore we will not be provided with any monetary aid for the creation of our system. In our second semester, our group will continue to modify and improve the quality of our simulation system to cater for more realistic conditions that could potentially be faced by drones.

1.2 Problem and Project Statement

In today's world of technology, wouldn't it be great if we have our package arrive hours after placing an order? The idea of expanding all possibilities and putting customers first is every delivery company's dream. As such, using drones has become a possible, plausible alternative to help speed up the delivery process. After all, the company's profits work hand in hand with the number of "happy customers" they have. But this seemingly lucrative solution has common problem i.e: Regulations from the FAA. Hence if we were to consider all of the rules listed by the FAA on drones for the purpose of delivery or not, this project will not be able to fully provide solutions to all of the rules. But, for a start this project will be a starting point for future attempts on providing full solutions for abiding all of the list rules regarding drone activities.

Our project aims to mainly solve the issue of air to air collision between autonomous drones with preset flight paths. In our system there will be two main parameters: warehouses and demands. Each warehouses will be allocated a unique drone where they will be responsible for the completion of demands appearing randomly on the map as time goes. The pathway of drone always follows this rule: they will fly from their warehouse to the place of interest and eventually get back to their warehouse.

Hence our system will eventually be a simulation software that when we run it, checks for the possibility of intersecting flight paths, and delays the delivery till its preset paths are clear. A huge assumption on our part is that the drone will not fail in any cases except for colliding with other drones during simulation. It will also have unlimited battery life throughout the simulating process. The fanciness of our simulation software to make it as realistic as possible would only be done after solving the main issue of air to air collision and the display of updated position on the GUI.

1.3 OPERATIONAL ENVIRONMENT

The final product, will be a simulation software created using java eclipse. This simply means that there won't be any potential life threatening hazards to any of the users or hazards that will in any form damage/destroy our product. To prevent our code from being accidentally deleted, we will circulate among members of the team through the use of email on the latest copy of the code. Using the fact that we have a small team, we decided to not complicate our progress by using gitlab. The software that we will be using is called Eclipse, it's free and members of the team will each be downloading it to work on the code for our project.

1.4 INTENDED USERS AND USES

There will be two groups of users that we are targeting.

1) Delivery companies such as amazon air prime and Google are prying into the prospects of using drones from delivery.
2) Transport companies such as uber elevate that plan to develop autonomous flying cars.

In both cases, our system would be able to simulate air traffic conditions during a time when the airspace is getting populated with many flying objects. The concept of delivering packages from point A to B is the same as delivering humans from point A to B. Flying objects must be able to reach their destination safely especially if they were flying autonomously.

1.5 Assumptions and Limitations

The assumptions, we made, are:

1) Each warehouse has an unlimited number of goods for the purpose or delivery.
2) Each warehouse will be assigned a drone for the purpose of fulfilling customer demands (Number of drones might be subjected to change depending on cilent's interest).
3) Drones will always depart from their warehouse to a demand location and back to the warehouse to "refill" and wait for the next order in line.
4) Drones will all be flying at the same altitude at the same speed given by the specifications of the Dji 3 phantom aircraft.
5) Other possible obstructions such as tall trees, building, power-plants… etc will not be present.
6) Preset trajectory paths would always be assumed as a "straight line" in between two points.
7) Population distribution in Ames will be assumed "uniformly distributed".
8) No actual drones will be incorporated/use as part of the simulation/testing process
9) Windy,snowy, rainy conditions will not considered.
10) Drones will not fail under any circumstances except for an air to air collision with other drones.

Limitations:

1) Map size will be limited to the city of Ames (think of this as an imaginary square surround ames)
2) FAA drone regulations to be followed.

1.6 Expected End Product and Deliverables

By the end of this semester, we will have a prototype software with basic functionality that will primarily solve our listed problems for this semester.

The problems we are expected to solve are:

1- Ensuring no collisions between drones

2- Updating drone's latest position on the map by displaying it either at the front end display or some temporary output

3- Erasing completed demands off the map

4- Generation of demands on map

5- Developing an algorithm that will help to satisfy demands in an efficient manner

6-Displaying simulated results on some form of User Interface (front end display) /Visual output

Hence, the above six points will be serving as our checklist and deliverables in monitoring our progress for this semester. These points would be subjected to change accordingly to the intentions of the client. The six deliverables will either be shown individually as a single output on a single screen or as a combination of conditions in a single output. The most ideal scenario would be on a working GUI that has all of the five conditions working in the background: displaying the “correct” output on the GUI.

As for our tasks for the second semester, we would foresee that considerations of new problems to be added to the above list to allow our simulation system to simulate conditions that are more realistic and suitable for the purpose of low altitude flight analysis. Due to the fact that we were only briefed slightly on the client’s “final product” , we will not be discussing anything that isn’t confirmed in our “to achieve” list for the next semester. All tests/prototype demo will be done using the software that we had chosen: eclipse, and in the event where we fail to achieve the display of certain portions in the given software, we will be displaying the results in a temporary “stand in” software: Matlab.

2. Specifications and Analysis

In order to solve the given problems, we came up with an idea that basically uses four different array list and many other functions that work hand in hand to eventually simulate the movements of the drones fulfilling demands of customers. Each functions has specific tasks, in which will be discussed shortly. The lists are customer demand list, request flight list,warehouse location list and ongoing flight list. These four list served as our main databases where we would constantly retrieve or store information in them when the simulation is ongoing to ensure the conditions are met.

The first function will handle information from the user (customer/demands); the function will read or receive data from them which will allow them to manipulate the necessary information for the computations of the remaining backend codes..

The second function will handle the calculation in finding the exact distance between two points. This function will be used frequently in conjunction to other parts of the system code.

The third function will be constantly checking for collision possibilities between drones. According to the algorithm that we had created, this function will utilize the "stored databases"- flight request list and ongoing flight list, by comparing their virtually plot flight paths with one another. Between the flight paths taken from the databases, this function will tell us whether or not a collision will occur. In the event if no collision occurs, permission to fly will be granted to both flight paths. But in the event where the systems detects potential collision, this function would then prevent the drone that has yet to take off from taking off. It will constantly be checking for the completion of the ongoing drone. Here "completion" means that the drone has to return back to its original departed location.

The fourth function is to calculate the new location of ongoing flight with respect to the time or distance traveled. This being a complicated problem, leaves us with only a way to implement this function. We will be making use of the "typical four quadrants" methods used in polar calculations for the calculation of this "new location" with respect to time.

Finally, the simulation function can be done in many different ways: the best of them is that the function could use an external website or program such as google maps to show how flights are moving on a 2D paths. If not, we could create a map that incorporates the distinct places of our bounded Ames, IA and show how drones work to fulfil the rising demands of customers over time.

2.1 Proposed Design

The design our team proposes consists of many functions each with specific tasks. We will be creating four arraylist to act as our database which keeps track of customer demands, initiating drone clearance between the pathway from warehouse to a demand, storing of warehouse information and information for ongoing flight. We will have another function specifically the trajectory function, which will constantly be updating and displaying the current location of the drone on our map using the data from the ongoing flight arraylist. Tasks of other functions that will be responsible in achieving our deliverables are shown below.

Front-end: We will be creating a java based GUI

Back-end:
We had decided to create multiple simple functions that will be assigned in doing a specific task. Following which we will be making use of these functions for the creation of more complex functions.

The functions are:

1) Distance function:

Calculates distance between two points of the map using latitude and longitude parameters. These information can be obtained from google maps.

2) Calculating current drone position function:

Calculate current drone position using basic math techniques of sine,cosine, tangent properties, projection of axis. For e.g, in a x-y plot, when x and y are both positive, angle is positive.Likewise, when x-y is negative, angle is negative. So by incorporating all four possibilities of " different angles" we can get our current drone position.

3) Random demand generator:

Using a random algorithm to generate different locations for the representation of "random customer demands".

4) Creating arbitrary map function:

This function is needed till we can find out a way to import all data from google map for the purpose of getting latitude and longitude information of demands. This serves as an arbitrary map to get the latitude and longitude information by creating a boundary with 10000 points vs 10000 points.

5) Collision checking function:

This will be one of the more complex function that uses previous functions to ensure that whenever there is an intersecting path/paths, permission will not be granted for both flight paths. In the event where one path is granted permission(ongoing), the other intersecting path/paths will not take off, it will be assigned to fulfil other non-intersecting paths demands.

6) Few arraylist for storing different types of data (acts as a database of the system)

There will be four arraylist used as a database of our system. In future should there require a modification of our current system, arraylist can easily be added for that purpose. The four list includes: warehouse list,flight-request list, demands- list and ongoing flight list. The primary data found in these lists would include the latitude, longitude information and possibly time information at which the demand has been made.

The amount of data parameters will be constantly evaluated during the process of developing the code.

7)Transferring demands data to flight request list:

This function will literally do the task as indicated by its name.

8)Selecting the nearest warehouse function:

This function will allocate demands to the nearest warehouse associated to it.Although in the limitations section, it is indicated that the only way for our drone to fail is through an air to air collision, "battery" life would not be posing as a problem in our simulation. But to simulate the realisticness of drones having a short flight time, the best way to show this feature is to follow this algorithm.

9) Plotting of current drone position function through the use of stored database:

This function would be the main function used to output our code to a front end GUI.

10) time taken function:

This function acts as an upgrade for the realisticness of our product. For now, this is secondary and something "bonus" but will eventually be completed in the next semester if not completed already. This function will calculate the time taken from A to B and work out the arrival and departure times for drones.

2.2 Design Analysis

The software is made up of multiple functions including the arraylist discussed earlier. Functions discussed earlier involved in doing of specific tasks. Such as checking for the possibility of collision by "comparing" flight paths. In the event when flight paths are crossed, either one of them will be allowed for flight or both will not be allowed for flight. The software will then the assign the drone to a "next in flight" path for it to do its delivery. All in all, the basis of our design works by first getting the locations of demands,then transfer them into a database(arraylist in our case) , then assign them to their closest warehouse, following which checks for potential collision by looking at their pathways. Finally, when all is done internally, the last function will be the real time plotting which showcases the movement of the drones on the map. This in fact is yet to be completed. We are currently having some difficulty in doing this function, but we will continue to improve and debug them.

The strengths of our proposed design includes:

1) 100% collision free algorithm between drones
2) Algorithm that fulfills demands the most efficient way
3) Allowing users to see a possible "what if" scenario when drones are to be allowed for the purpose of delivery
4) Providing users with the basic idea of "what will happen if..."
5) Allowing users to brainstorm for more possible scenarios
6) Allowing users to save money on real time drone testing scenarios for data interpolation
7) User friendly usage of software

The weakness of our proposed design includes:

1) limitations of the map size
2) Not much options as to what the users can do when running the simulation (minimal user to software interaction)
3) Massive amount of assumptions
4) Drone's flight paths are only in a "straight line"
5) Depicting only a scenario

Observations of the proposed solution:

1) Depicting only a scenario
2) Massive amount of assumptions to get things started and keeping things simple
3) 1 drone per warehouse which is not that realistic

3 Testing and Implementation

The Team have established a "standard testing procedure" for all kind of code related work. They are:

1) Testing the written code individually as a function by itself

2) Testing the same code again when the function is added to the main portion of the code

The Team’s method of attempting to try and solve a potential problem would comprise of :

- Personal attempt in solving a given task

- Researching on relevant methods that might help in the attempt to solve a given task

- Establishing a group discussion on tasks that cannot be solved alone

Hence, using the above as a guide, we'll be using this approach to go about new and current problems in working towards our desired outcome.In cases where the completion of a function could not be achieved by a particular member, the team will then be working together to help resolve the issue.

Functional testing:

Amongst the indicated deliverables discussed earlier, the team has came into an agreement that our basic prototype will be made up of all previously indicated deliverables with the option of temporarily replacing the display of results on a java GUI with any visual form of representation in the event we fail to achieve the pairing of back-end and front-end systems. The basic prototype should be able to convey the intentions of the client with any visual form of output.As such we will ensure that all back-ends conditions are fully met and in the event where we fail to create a working GUI in time, we will be definitely be showing our intended outcomes in some visual representations with no interactive capabilities.

The basic visual representation could be one of the following:

1) A figure based depiction of the many “what ifs scenario” with the conditions bases on the deliverables
2) A matlab based plot function that represents the key idea of the entire project.Parts of our results of the deliverables will be presented through matlab.
3) A working java GUI as indicated in our deliverables (ideal case)

Some of the back-end validation test that had been successful:

1) Distance function validation:

$$\operatorname{haversin}\left(\frac{d}{R}\right) = \operatorname{haversin}(\Delta\varphi) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{haversin}(\Delta\lambda).$$

Fig.1. Distance formula

After the completion of this function with the help of the given formula, we tested it alongside with the actual coordinates found in google maps. First on our arbitrary point based map, we will select the same 2 points found in google maps and pass it through our written function which eventually returns us a result(distance). This calculated distance is then compared to the distance provided in google maps. Usually for the validation process, we will choose two points that will provides us with a “linear line”. Since our assumption of drone pathways are a straight line, choosing a “linear path” fits our criteria.

2) Current position function that will be used for the plotting of drone location:

Using google maps, we tested it by selecting random points from it in the form of (x1,y1) and (x2 , y2) and observed its result. The function that we have should return a new (x,y) location that will be on the “linear path” between the 2 selected points. If the returned results are not on/close to this “linear path”, it simply means that this function is not working as intended.

The above discussed validation test, involves the process of meeting our deliverables. For a detailed list of validation test of every function that we have, please refer to our project plan section 2.10.

Non functional testing:

Although it is not explicitly stated by our client, and that our project is basically made of blocks of codes, we will ensure that our code is always kept at minimum to allow the system to process at a “fast enough” speed for a smooth simulation.We will be setting the “baseline” of our definition of lag to under 2s. We will be following the standard typical way of organizing ,commenting our code as and when during the process of manipulating our code.

Technical Approach:

In order to approach this project and be able to deliver our 1st prototype on time, the team has decided to divide the project into two parts:

In the first part, the team will work mainly on all of the back end codes which simply fulfills the calculation, algorithm of most parts of our deliverables. Due to the fact that there will be no visual interactive output, the team will constantly be checking for accuracy of the written codes by validating results through a simple “printf” equivalent function in java. The team will also be making use “other” preloaded programs such as excel and matlab for a temporary visual assurance of the printed results. This temporary visual reassurance acts like a redundancy check.

In the second part, after the completion of the backend coding, the team will then move onto the creation of the front end display (GUI). This front end would then be responsible for the display of our intended results to users. Although this part involves only a “one step process” in processing the backend and transferring it over to the front end, we would foresee that there would be various obstacles associated to it. This would mainly be attributed to the team composition and unfamiliarity with the process of creating a GUI. Ultimately, we would try to complete the first phase as fast as possible so we could have more time to work on part 2 of our approach.

3.1 Interface specifications

The client has no preference on the types of programming languages for the development of the project. Since the fact that this project is software based, and that it requires us to create the software from start, we were given the freedom of choice in the selection of programming language. Therefore, the team has agreed and decided that ultimately this project will be delivered through the use of java eclipse.

There will only be one interfacing specificification: to make sure that the output GUI is able to interface with the back-end code.

3.2 Hardware and software

We will be using eclipse, a java based software to write our simulation project. In some cases, for the sake of testing and the validation of our work, we will be using excel,matlab and google maps as a temporary form of checking our work without the need for our GUI to be up and running. Due to the fact that it’s a simulation based project, there will not be any hardware interfacing process or any hardware assembly process. We will be referencing the model of our drones to be that of DJI Phantom 3 with the assumption that it will have unlimited flight time. The GUI will eventually be a GUI created from eclipse that will showcase the entire simulation process.

3.3 Present project version vs future project version

Our design details will not have much changes pertaining to the overall “plan” associated with it. The main difference is that in our present plan, because that we are still facing problems in the creating a plot that works in a java GUI, we will be presenting a visual plot of what is expected eventually as a matlab based plot function. This visual output will be able to convey what we are trying to achieve eventually but there will not be any interaction between users since the output is not a GUI. Also we will definitely be adding in more features to make our simulation as realistic as possible.

One of the features that will probably be added would be: increasing the number of drones in each warehouse. We will continue to brainstorm and research on plausible scenarios that might be posing as a problem for autonomous object,drones and FAA rules. Eventually, the later parts of our project simply involves the process of “making our basic prototype” fancier and more realistic.

3.4 Process details

Currently we have completed all of our back-end sections of the code. These back-end codes consists of functions that had been introduced at the start of the document (2.1). To reiterate, proposed functions involves the completion of a particular task. With multiple functions done and checked for integrity issues,we are currently working on the process of integrating all of our back-end codes to a java based GUI.

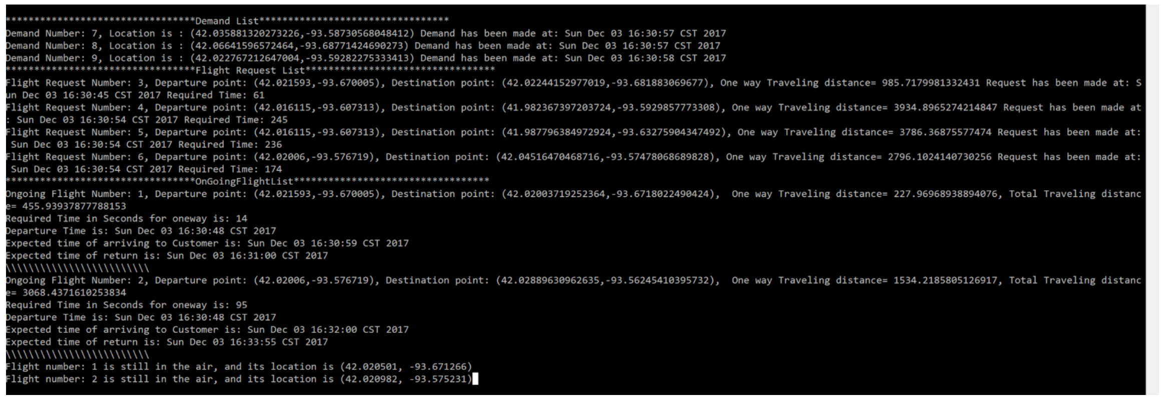
Until now there had been some struggle with regards to the creation of a java based GUI. Since time is running out in this semester, we decided to at least be able to convey our intended intent of the project through the creation of a temporary “matlab compatible” code by displaying the output from our java code in that form. Then in the later stages, mostly in the next semester we will continue to troubleshoot the issues that we were facing in the creation of the java based GUI.

3.5 Results

There had been some minor failures encountered during our coding process but fortunately they were solved within a couple of days. These minor failures that cause things to work a little weird was due to the lack of understanding and the lack of experience in java. Such setbacks were expected mainly due to the composition of our team and that half of us had to “learn on the spot” a new programming language. Judging from how we have our plans laid out, our progress has been great when attempting to solve the “back-end” problems, we managed to achieve the minor milestones by completing functions that are the solution to our designated deliverables.

Things has been relatively smooth for us but we have encountered a situation where we currently is struggling to solve. We had been trying to figure out a way to plot and refresh the drone’s position and a way to create a java GUI that allows the pairing of our backend codes. We had the issue of not being able to properly troubleshoot the pairing of our backend codes to the GUI and we are facing issues on updating the current location of the drone by plotting it. In the future, more functions will be added to cater to our cilent’s request in turning our software into a more realistic simulation program.

Some screenshots of parts of the code that works and are in progress:



```
********************************Demand List********************************
Demand Number: 7, Location is : (42.035881320273226,-93.58730568048412) Demand has been made at: Sun Dec 03 16:30:57 CST 2017
Demand Number: 8, Location is : (42.06641596572464,-93.68771424690273) Demand has been made at: Sun Dec 03 16:30:57 CST 2017
Demand Number: 9, Location is : (42.022767212647004,-93.59282275333413) Demand has been made at: Sun Dec 03 16:30:58 CST 2017
********************************Flight Request List********************************
Flight Request Number: 3, Departure point: (42.021593,-93.670005), Destination point: (42.02244152977019,-93.681883069677), One way Traveling distance= 985.7179981332431 Request has been made at: S
un Dec 03 16:30:45 CST 2017 Required Time: 61
Flight Request Number: 4, Departure point: (42.016115,-93.607313), Destination point: (41.982367397203724,-93.5929857773308), One way Traveling distance= 3934.8965274214847 Request has been made at
: Sun Dec 03 16:30:54 CST 2017 Required Time: 245
Flight Request Number: 5, Departure point: (42.016115,-93.607313), Destination point: (41.987796384972924,-93.63275904347492), One way Traveling distance= 3786.36875577474 Request has been made at:
 Sun Dec 03 16:30:54 CST 2017 Required Time: 236
Flight Request Number: 6, Departure point: (42.02006,-93.576719), Destination point: (42.04516470468716,-93.57478068689828), One way Traveling distance= 2796.1024140730256 Request has been made at:
 Sun Dec 03 16:30:54 CST 2017 Required Time: 174
********************************OnGoingFlightList********************************
Ongoing Flight Number: 1, Departure point: (42.021593,-93.670005), Destination point: (42.02003719252364,-93.6718022490424),  One way Traveling distance= 227.96968938894076, Total Traveling distanc
e= 455.93937877788153
Required Time in Seconds for oneway is: 14
Departure Time is: Sun Dec 03 16:30:48 CST 2017
Expected time of arriving to Customer is: Sun Dec 03 16:30:59 CST 2017
Expected time of return is: Sun Dec 03 16:31:00 CST 2017
\\\\\\\\\\\\\\\\\\\\\\\\\
Ongoing Flight Number: 2, Departure point: (42.02006,-93.576719), Destination point: (42.02889630962635,-93.56245410395732),  One way Traveling distance= 1534.2185805126917, Total Traveling distanc
e= 3068.4371610253834
Required Time in Seconds for oneway is: 95
Departure Time is: Sun Dec 03 16:30:48 CST 2017
Expected time of arriving to Customer is: Sun Dec 03 16:32:00 CST 2017
Expected time of return is: Sun Dec 03 16:33:55 CST 2017
\\\\\\\\\\\\\\\\\\\\\\\\\
Flight number: 1 is still in the air, and its location is (42.020501, -93.671266)
Flight number: 2 is still in the air, and its location is (42.020982, -93.575231)
```

Fig.2. Some Results

4 Closing Material

4.1 Conclusion

Up to this time, we have tested all of our back-end functions that does the following: calculates distance between two points, getting the gps location of the drone, plotting of map with limits in numbers, changing google map’s degree coordinate into numbers, generating random demands numbers, arraylists for database storages and flight initiation, developed an algorithm that will satisfy the fulfilment of demands in an efficient manner (in progress),a function that prevent collisions of drone with intersecting flight paths,the trajectory function that will update drone’s gps location along with time which is in its testing phase. These functions have already been both tested individually and validated. However, the results that we are seeing are basically just “numbers”.We plan to stick to our goals by first completing the codes that will solve our designated problems before working on the front end of our system(output screen). In terms of work progress wise, we would say that we are on our way in successfully making a “basic enough” prototype that will be upgraded in the next semester.

4.2 References

[1]"Stack Overflow - Where Developers Learn, Share, & Build Careers", *Stackoverflow.com*, 2017. [Online]. Available: https://stackoverflow.com/. [Accessed: 30- Oct- 2017].

[2]w. Chris Veness, "Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript", *Movable-type.co.uk*, 2017. [Online]. Available:
http://www.movable-type.co.uk/scripts/latlong.html. [Accessed: 30- Oct- 2017].

[3]"Build software better, together", *GitHub*, 2017. [Online]. Available: http://github.com. [Accessed: 30- Oct- 2017].

[4]P. geotools, "Plot the longitude and latitudes on map using geotools", *Gis.stackexchange.com*, 2017. [Online]. Available:
https://gis.stackexchange.com/questions/178890/plot-the-longitude-and-latitudes-on-map -using-geotools. [Accessed: 30- Oct- 2017].

4.3 Appendices

We hoped to achieve the typical flight tracking output by working towards figure 3.

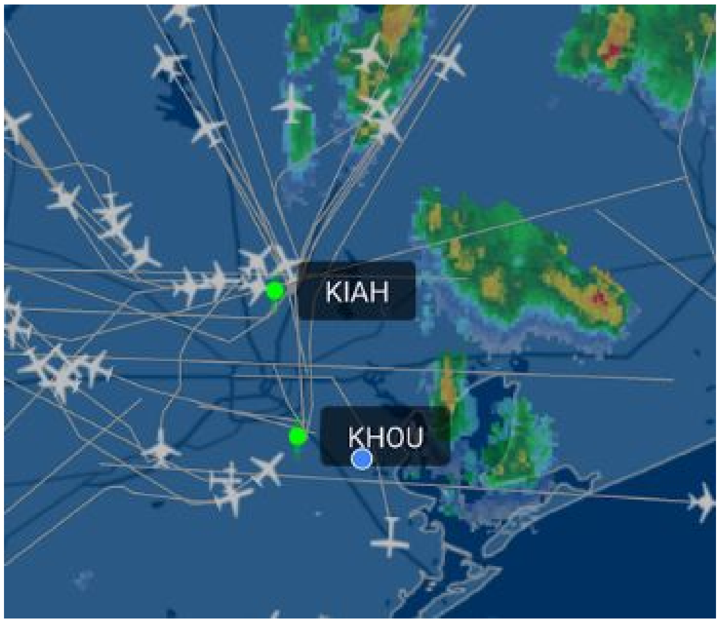


Fig.3. Example Output