TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: On-Board Autonomous Conflict Detection for UAS Platforms.

TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació

AUTOR: Óscar Vázquez Navarro

DIRECTOR: Enric Pastor Llorens

DATA: 23 de març de 2011

Títul: On-Board Autonomous Conflict Detection for UAS Platforms.

Autor: Óscar Vázquez Navarro

Director: Enric Pastor Llorens

Data: 23 de març de 2011

Resum

En aquest treball volem crear un algorisme de detecció de conflictes entre aeronaus, mes especificament per un avió no tripulat (UAV). Per això abans de tot necessitem saber i coneixer tot referent als missatges ADS-B, els missatges pels quals es comuniquen les dades del avió al centre de control aeri.

En segon lloc fem un breu estudi sobre els algorismes ja existents tan per detecció de conflictes com per a resolució de conflictes.

Un cop acabada la investigació previa, harem de investigar una mica sobre els software que farem servir com es el cas del programa eDEP de simulació de trafic aeri i el programa C# per la implementació del nostre algorisme.

A continuació ja tenim dades suficients com per poder crear el nostre algorisme de detecció que un cop acabat podrem evaluar els resultats amb satisfacció.

Title: On-Board Autonomous Conflict Detection for UAS Platforms.

Author: Óscar Vázquez Navarro

Director: Enric Pastor Llorens

Date: March 23rd, 2011

Overview

In this Project we want to create an algorithm for conflict detection between aircrafts, more specifically for UAV.

The first thing we need to understand is how the ADS-B messages work and are sent to the ATC. After this study we also need to know which conflict detection and resolution algorithms exists and how are implemented.

Once the previous documentation is done we need to know and understand the software environment we are going to work on. Specially the eDEP simulation platform and also the C# platform in which we are going to implement our detection algorithm.

After having done the previous work either in theoretical information or in software practice we are going to start with the implementation of our detection algorithm.

Index of Contents

Index of Figures

Index of Tables

Glossary

ADS-B	Automatic Dependent Surveillance - Broadcast
ATC	Air Traffic Control
CAD	Closest Approach Distance
CDR	Conflict and Detection Resolution
CDU	Control Display Unit
EDEP	Early Demonstration & Evaluation Platform
FIS-B	Flight Information Services - Broadcast
FMS	Flight Management System
IFR	Instrumental Flight Rules
RSSP	Radar Systems Specialists Panel
SASS-C	Surveillance Analysis Support System for ATC-Centre
STFRDE	Surveillance Task Force for Radar Data Exchange
TCAP	Time of Closest Approach Point.
TCAS	Traffic Collision Avoidance System.
TIS-B	Traffic Information Service - Broadcast
UAC	User Account Control
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
VFR	Visual Flight Rules

1. Introduction

Nowadays, passenger's Air Traffic Transport is considered the safest mean of transport. According to IATA, in 2010 the air accident rate was 0.61, which is equivalent to 1 accident for each 1.6 million of flights, the lowest rate of history followed by 2006 that was 0.65.

To keep the low rates, aircrafts safety either in ground or in air is a very important factor. For that reason, we need to know their exact position in every moment and keep them well separated and guarantee a fluid air traffic flow.

For that reason, to keep aircrafts well separated and well communicated we need the roll of ATC. Air Traffic Control is a service provided by controllers with the main purpose to separate aircrafts to avoid collisions by providing information and any support pilots request when possible. Controllers also organise the flow of traffic which is important because in a "controlled airspace" safety is ensured and less aircrafts are involved in possible conflicts.

As a backup for controllers the aircrafts have their own technology for conflict detection, it is called TCAS. Safety studies of TCAS have estimated that this system improves safety in airspace by a factor between 3 and 5. The display as shown in Figure 1 below, is located in the cockpit and warns the pilot of the presence of other aircrafts around. This system works only if the transponder of the rest of the aircrafts around are active, if not the aircrafts will not be shown in the screen.

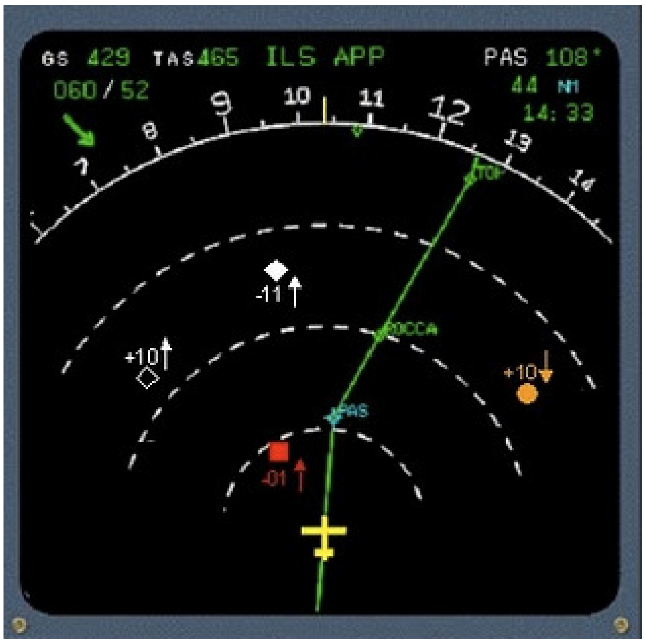


Figure 1: Display of TCAS system situated in the cockpit

The future of this technology will be based on ADS-B messages and data links between aircrafts themselves or between aircrafts and ground stations. The existing TCAS are able to process the ADS-B messages. Once the ADS-B transponders will be compulsory for every aircraft, TCAS performance will be enhanced using techniques known as "hybrid surveillance". Making TCAS using the ADS-B messages will reduce the rate in which the TCAS equipment interrogates the nearby aircrafts.

In this Project we will focused on conflict detection between aircrafts, but we are going to be more focused on UAVs. Conflict resolution is as important as conflict detection, for that reason we are going to make an overview in chapter 2.

Regarding our implementation in C# code we are going to create a program that will be able to know the exact position of all the aircrafts of a simulation with the data received by eDEP Platform. Knowing the position of the aircrafts is basic because having that information we are going to be able to calculate the possible conflict point between them, the conflict time until they reach the collision point and the distance to their actual position to the conflict point. Some other calculations we are going to implement is if the calculated conflict point is in a look ahead time or just a convergence of the aircrafts 'headings in the past.

1.1 Objectives of the Project

- Explain what is a conflict between aircrafts.
- Explain some methods of conflict resolution.
- Localise conflicts between aircrafts.
- What is an UAV and what are they used for.
- To understand the ADS-B messages send by an aircraft.
- Get to know the eDEP environment with Java Eclipse.
- Creation of a C# software for conflict detection through data received from eDEP Program.

The main objective of this project regarding all the objectives mentioned above is to achieve that an UAV detects a conflict with another aircraft.

An UAV is an aerial vehicle, powered that doesn't carry any human crew and uses aerodynamic forces to fly and can be remotely controlled or can fly autonomously based on pre-programmed flight plans. The UAV history started with some drones remotely controlled but now they are very well improved and they can fly autonomously. Although the main use for UAVs is in a military concept there are a huge variety of uses for UAVs.

- In the military applications, UAV are used for surveillance and target designation, weapon delivery or electronic countermeasures among others.

- In the security applications, UAV are used for border surveillance, maritime surveillance, anti-terrorism or sensitive sites surveillance.

- In the civil applications, UAV are used for forest fire detection, pollution detection, agriculture and fishing among others.

There is a challenge for UAV that is to design a sense and avoid system that would sense the presence of other aircrafts nearby and would take some steps to diverts the UAV from the other aircrafts flight plan. This challenge is developed by ASTM and the standard is called ASTM F 2411.The main objective is to create a mechanical system designed to take the place of a pilot maintaining the level of safety. This sense and avoid system has not yet been certificated.

The UAV is just the most known part of a bigger system called Unmanned Aircraft System. To achieve the goals and make a UAV fly, there are a lot of systems working behind.

A typical UAS consists of the following parts:
Unmanned aircraft
Control system, such as ground control station.
- Control link.
- Other related support equipment.

In figure 2 and 3 we can see two different UAV with different shapes, sizes and configurations. In figure 2 the UAV is used for forest fire detection and beneath the left wing carries NASA's Autonomous Modular Scanner, a self-contained IR-thermal imager. In figure 3 we can see a weaponized UAV, for military applications.

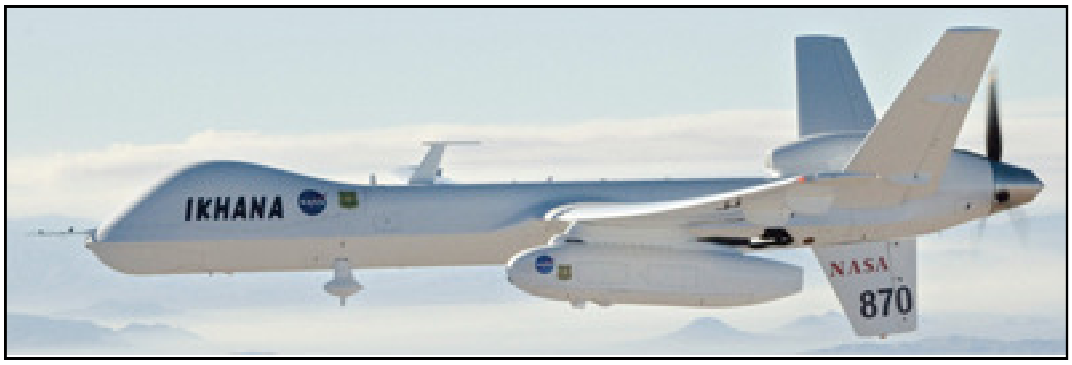


Figure 2: Ikhana, an UAV modified predator

Figure 3: Weaponized UAV

1.2 Structure of the Document

In chapter 2, we give some information on previous work done on conflict detection between aircrafts and also a brief information on conflict resolution.

In chapter 3, eDEP, ADS-B messages and Asterix take place. We will explain what is eDEP and its software. Regarding ADS-B messages we will introduce the ADS program and which information will be useful in ADS-B messages for our project and concerning Asterix we will explain what it is and what it is used for.

In chapter 4, we are just focused in eDEP software and what we can get from it, how it is displayed and which screens are we going to use. Afterwards with this information we are going to program a conflict detection algorithm in C#.

Chapter 5 is fully dedicated to our implementation work. We are going to describe the calculations we need to implement a C sharp program to receive data from eDEP software that will detect the conflicts between aircrafts and they will be shown in a console screen.

In chapter 6, we will analyse our results from C sharp program crossing references from our C# code and what it is displayed in eDEP consoles.

Chapter 7 brings the Projects conclusions and our final observations.

2. Previous work on conflict detection

2.1 Conflict Detection

One of the most important design parameters for UAV system is the collision avoidance system. Its design needs to be reliable and intelligent so it can be integrated in civilian airspace. These algorithms have to assure conflict free trajectory planning but also provide basic autonomous navigation capabilities.
Conflict detection is useful in a short lookahead time, usually 5 minutes. If this lookahead time is longer we won't even look for possible conflicts in real life.
It also has been very studied and there are several different algorithms for conflict detection. To describe the aircraft trajectory, the simplest algorithms just need a position, velocity vector and a time interval data.

According to Hyo-Sang Shin [9], to design the CD&R algorithm we first need to introduce the definitions of conflict detection and conflict resolution as it is done in TCAS system. These definitions are: Closest Approach Distance (CAD) and Time to Closest Approach Point (TCAP). His study is based on geometric conflict detection and resolution system. In the following image we can see the CD&R geometry.

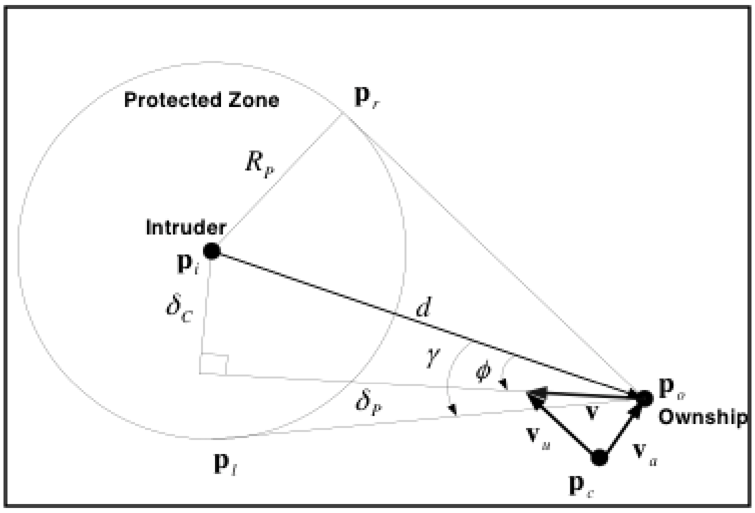


Figure 4: CD&R geometry

As conflict detection and resolution can be as reliable as the ability of the model to predict the future, James K. Kuchar [4] has made a study in which the most concrete difference between modeling approaches involves the method by which the current states are projected into the future.

Three different extrapolation methods have been identified, termed nominal, worst case, and probabilistic. The tree methods are shown in Fig 2.1.2, the nominal projection (a), provides the best estimate of where the aircrafts will be.

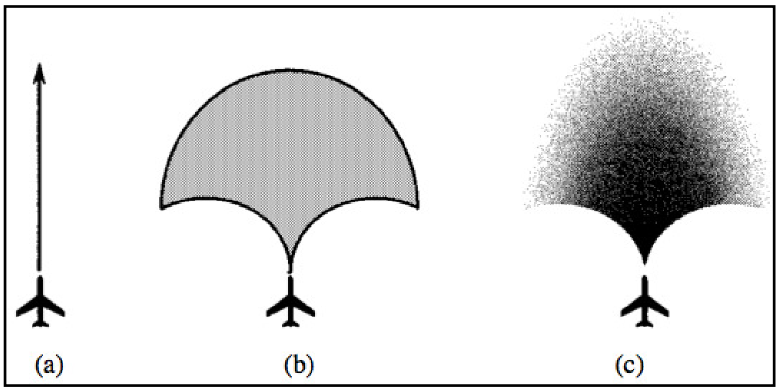


Figure 5: State propagation methods

The probabilistic method (c) is the most general: the nominal and worst-case are subsets of probabilistic trajectories. The worst-case model is one in which the aircraft will follow any trajectory with equal likelihood.

2.2 Conflict Resolution

The main point in this Project is conflict detection as we said previously but we also need to make a little overview in Conflict Resolution.

The aircrafts before taking off have their own trajectory planning algorithm that follows the path from the source airport to the destination airport following a sequence of waypoints. This path is calculated to be optimal and conflict-free and then consulted with Air Traffic Control to approve it. However, this path can be modified due to bad weather, high winds or schedule delays. This deviations from the original path are calculated by the central ATC and each aircraft before modifying its course has to have the clearance from ATC. The average of conflicts occurring between two aircrafts is 90% but conflicts involving three or four aircrafts may occur.

There are many algorithms to resolve a conflict, the first one is human factor, in which the air traffic controller receives the state information from the aircrafts and then he is the one who decides which manoeuvres need to be done to solve the existing conflict and then he communicates the pilot what to do.

The next resolution method is Altitude step & TCAS manoeuvres, the altitude step just calculates the altitude needed to reach before the conflicts takes place. The advantage of TCAS's manoeuvres is that they are very effective thanks to the shape of the protected zone, anyway, it avoids a huge bearing deviation. The disadvantage is that a lot of communication is needed for the manoeuvres and extra hardware need to be implemented or just increasing the bandwidth of TCAS system.

Another method exists, cross product of speed vectors. This method uses the cross product of speed vectors from the aircrafts in conflict. To solve the conflict we use a non-commutative method to obtain the result. Using this method we can establish the changing directions of bearing in the speed vector of the aircraft.

The last method is extended VFR rules. This method is very simple because it uses the methods from visual flights. EUROCONTROL has modified some visual rules so we can determine who has the priority in the airspace. For example, to determine the priority one of the rules is the state of flight (Ascending, descending, cruise, etc.).

The following two figures show two different conflict resolution methods, in figure 6 we can see how both aircrafts are going to turn left then continue straight and finally they will turn right to get to their initial heading and continue with their flight plan. in figure 7 the three aircrafts that could be in conflict the three of them turn right and they will start flying in circle like if they were in a roundabout and then once they reach their opposite site of the roundabout they will change heading to get the their initial one and continue with their flight plan..

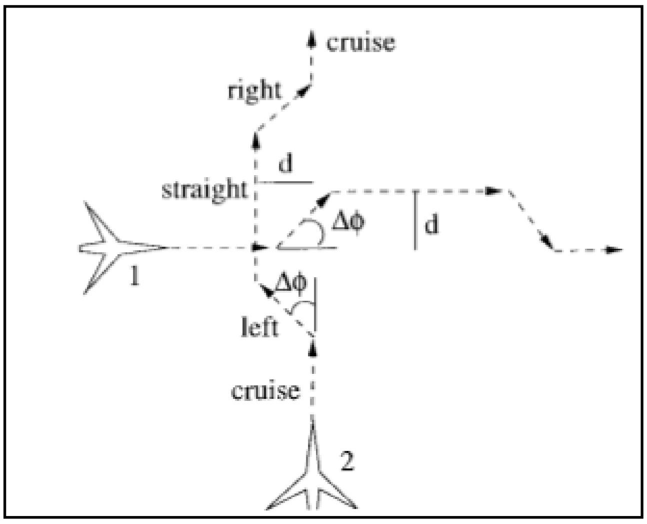


Figure 6: Heading changes maneuvers

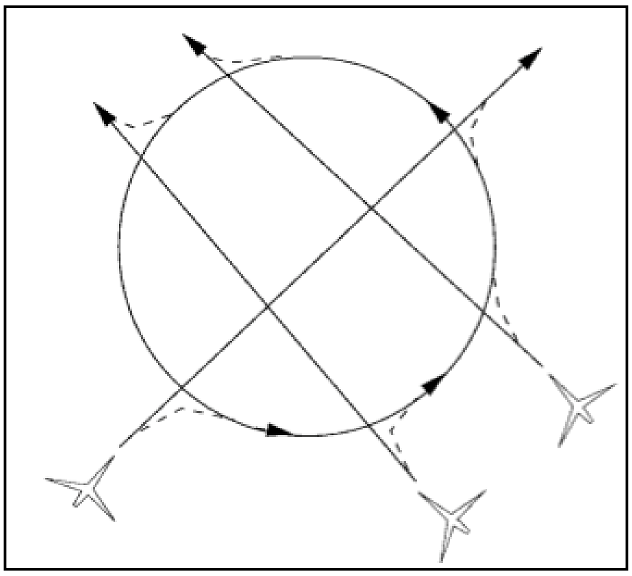


Figure 7: The roundabout or circle maneuver

In addition, aircrafts have their own automatic conflict solver that pilots would use depending on workload in the flight deck.
If pilots use this automatic conflict solver, there are three resolution modes: automatic, semiautomatic and manual.

- The automatic mode is when the pilot accepts the solution suggested by the automatic conflict solver.

- The semiautomatic mode is when the pilots accepts the suggestion from the conflict solver but he introduces some changes such as type of manoeuvre.

- The manual mode is when the pilot defines himself the conflict and introduces the changes to be done in the flight plan.

All this changes can be done via Flight Management Display (FMS) with this Control Display Unit (CDU) or directly by manipulating the aircraft Navigation Display.

3. EDEP overview and ADS-B messages

3.1 Early demonstration & Evaluation Platform (eDEP)

The eDEP software was created by Graffica enterprise for EUROCONTROL. The aim was to create a low cost, lightweight and web-enabled ATM simulation platform. The eDEP code has over 400k lines while the other simulation platforms have between 700,000 to 1.5 million lines.

The eDEP platform has been designed to be compact and to ensure a good code-reuse, for that reason all subsystems use the same code base. The optimum code length is the one that a group of 4 people can handle and maintain according to Graphica philosophy.
Since 2002 the code has increased from 90,269 lines to 399,230 lines in May 2007.

The platform is written in Java language to be executed in different operating systems; anyway it is a high quality platform since it is well documented about its progression. The platform offers their users a modern technology as well as development toolkit. The platform is focused for different purposes: investigation, advanced projects, human factors studies for ATC as fatigue or work load, simulations, experiments (HF Lab) and training.

The eDEP architecture is built in layers that encourage the software reuse. In the next figure the eDEP architecture is shown and the layers are well defined.

The Graffica System Development Kit (GSDK) offers the following functionality[1]:

- (terrain) map management
- Entity (model) management
- Pluggable entity parser (read-in, write-out) framework
- Scenario & simulation time management
- Efficient graphics engine
- Pre-built widget set
- Application management
- Geometry and projection functionality
- Rich event management
- Middleware functionality (discovery)
- State machine management
- Configuration resource management
- Various conflict (MTCD & HIPS) algorithms
- Simple Trajectory Prediction algorithms

[1] Reference: http://www.eurocontrol.fr/projects/edep/

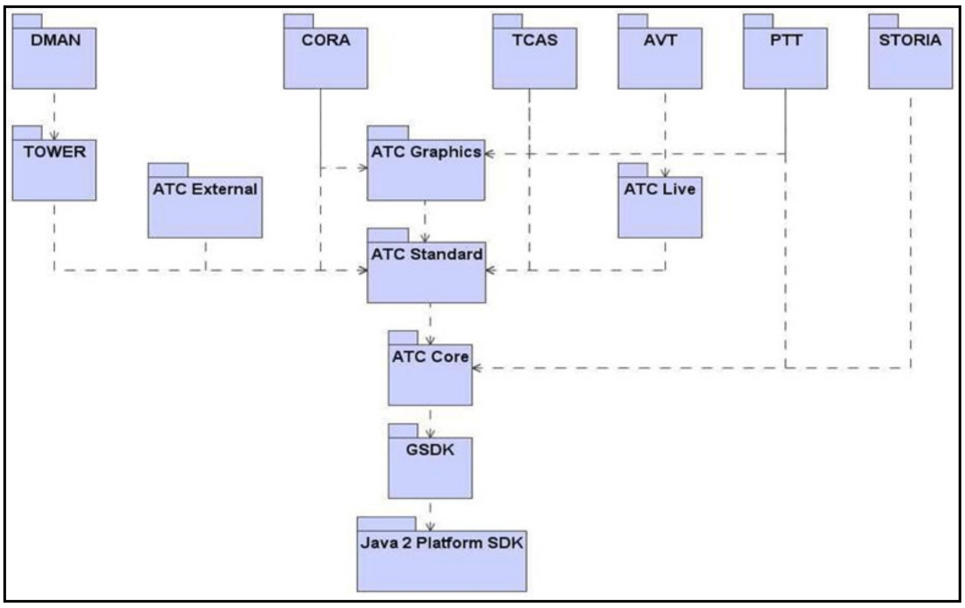


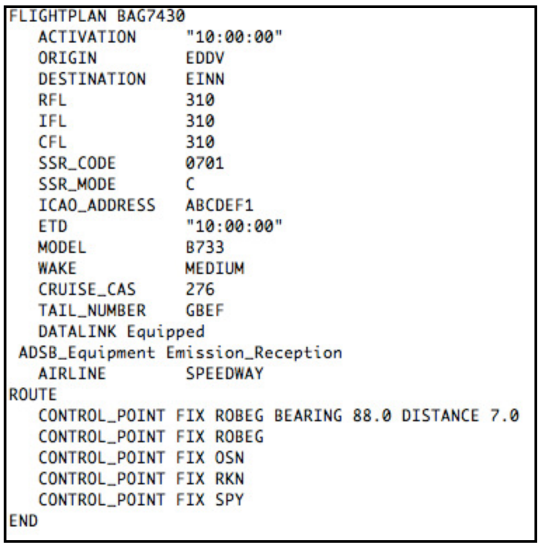
Figure 8: The eDEP Architecture

As eDEP is a program that simulates Air Traffic Control. For our project we just need any simulation with aircrafts flying.

In our case we are going to use two different files to make our demonstration and calculations:
eDEP/ATC-2009_mellejmi/ATC/src/atcapp/resources/demos/ians/luxdema_ADSB.gsdk
eDEP/ATC-2009_mellejmi/ATC/src/atcapp/resources/demos/ians/luxtraffic.dat

In the first file is where we put our IP address for the UDP connection, we put the scale values of the screens for the simulation, we write the files we are going to need to execute the file and we determine the type of message we want to sent through the UDP port, in our case messages category 021.

In the second file is where all the traffic data is kept in case we want to add another plane or we want to change their initial values as flight level or change their flight plan. In figure 9 we can see the information of one plane among other that will be shown in the simulation.
The target identification of this aircraft is BAG7430, we also can see the departure and destination point, the flight level during the simulation, the time in which the aircraft will start in the simulation and among other information the last information is its flight plan.



```
FLIGHTPLAN BAG7430
   ACTIVATION     "10:00:00"
   ORIGIN         EDDV
   DESTINATION    EINN
   RFL            310
   IFL            310
   CFL            310
   SSR_CODE       0701
   SSR_MODE       C
   ICAO_ADDRESS   ABCDEF1
   ETD            "10:00:00"
   MODEL          B733
   WAKE           MEDIUM
   CRUISE_CAS     276
   TAIL_NUMBER    GBEF
   DATALINK Equipped
 ADSB_Equipment Emission_Reception
   AIRLINE        SPEEDWAY
ROUTE
   CONTROL_POINT FIX ROBEG BEARING 88.0 DISTANCE 7.0
   CONTROL_POINT FIX ROBEG
   CONTROL_POINT FIX OSN
   CONTROL_POINT FIX RKN
   CONTROL_POINT FIX SPY
END
```

Figure 9: Initial information of aircraft BAG7430

Figure 10 shows just the eDEP console, the main purpose of this screen is to start the simulation, pause it, stop it and the last option is fast forward if we want to accelerate the time. In figure 11 we can see the main screen as if we were a traffic controller. In this screen we can select an aircraft and change its altitude, flight plan, heading, speed, change his frequency to another airspace. In figure 12 we see the traffic flow and the label from each aircrafts showing us their identification number, their flight level, their next sector.

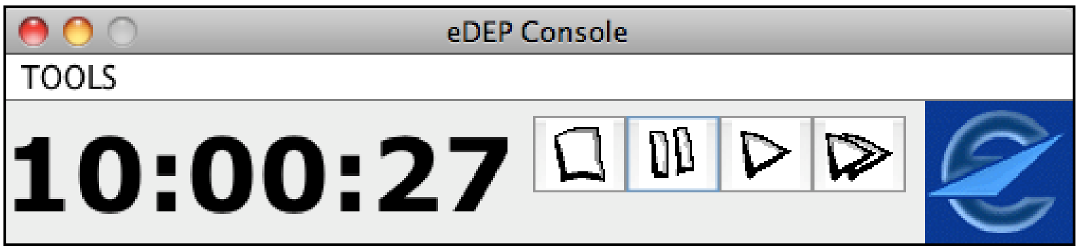
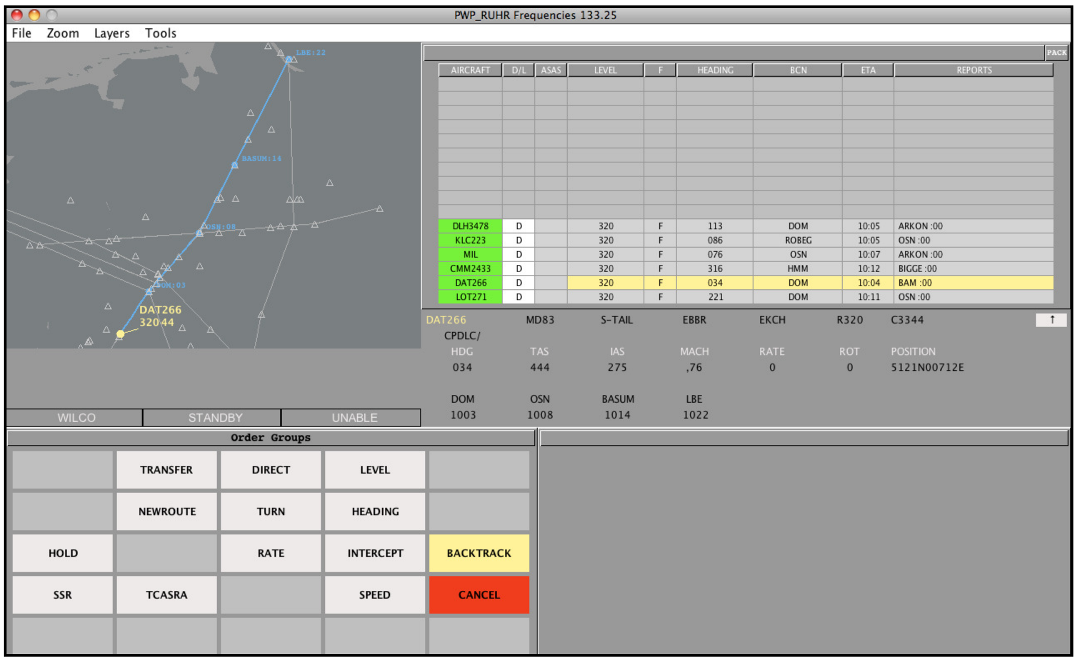


Figure 10: eDEP console for time control



AIRCRAFT	D/L	ASAS	LEVEL	F	HEADING	BCN	ETA	REPORTS
DLH3478	D		320	F	113	DOM	10:05	ARKON :00
KLC223	D		320	F	086	ROBEG	10:05	OSN :00
MIL	D		320	F	076	OSN	10:07	ARKON :00
CMM2433	D		320	F	316	HMM	10:12	BIGGE :00
DAT266	D		320	F	034	DOM	10:04	BAM :00
LOT271	D		320	F	221	DOM	10:11	OSN :00

DAT266 MD83 S-TAIL EBBR EKCH R320 C3344
CPDLC/

HDG	TAS	IAS	MACH	RATE	ROT	POSITION
034	444	275	,76	0	0	5121N00712E

DOM	OSN	BASUM	LBE
1003	1008	1014	1022

Figure 11: Screenshot for the main screen used as an air traffic controller where we can see all the aircrafts controlled by an ATC, from this console we can change any parameter of any aircraft using the ATC frequency

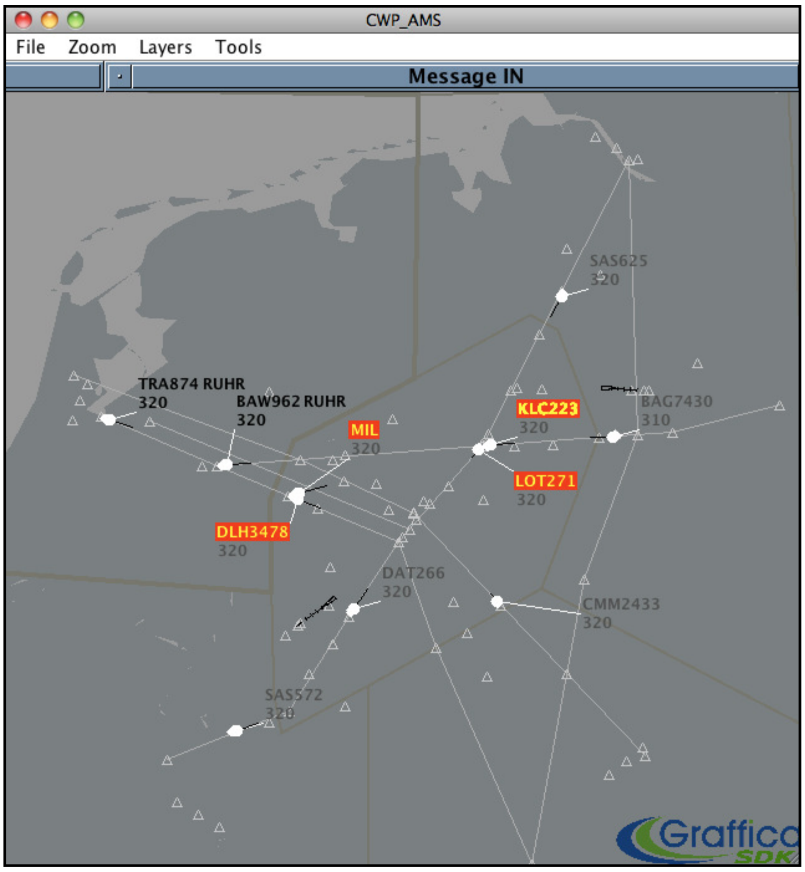


Figure 12: Screenshot where we can see all the traffic flow from our simulation

3.2 The Asterix Messages

The acronym of Asterix stands for: "All Purpose STructured Eurocontrol SuRveillance Information EXchange."

Historically every National Administration had its own format for delivering radar data to ATC Centres. This caused complications to exchange data from radars from different countries. In 1984, Maastricht UAC presented to RSSP what a common European data format could look.Two years after, in 1986 ASTERIX was officially approved and its manual was presented describing the initial structure and describing the data items that monoradar and weather data would cover in 1988. In 1994, the STFRDE was created to continue the work on ASTERIX Users Group and the responsibilities were passed from RSSP to Surveillance Team.

Nowadays the main users of ASTERIX are the Air Traffic Control (ATC) Centres and almost all the ECAC members. An Asterix message is a structured encoded binary data from ATM surveillance allowing a well synchronised transmission of information from any surveillance and automation system.The structure data that has to be exchanged is taken from the first encoded bit of information up to all the entire block of data without any lose of information during the whole process.

The transmission of any Asterix message can be done by any of the following communication medium: Local Area Network (LAN), Wide Area Network (WAN), Internet Protocols (IP) or any others belonging to lower layers shown in the next figure.

In the next figure we can see the Open Systems Interconnection (OSI) layers.
Asterix refers just to Presentation and Application layers which are layers six and seven.

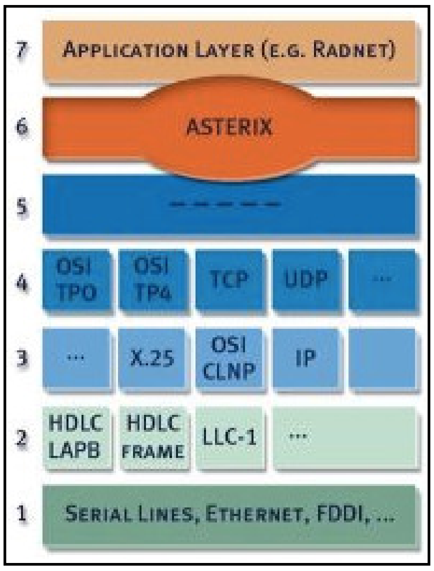


Figure 13: OSI layers

For the transmission of this information there is up to 256 categories of Asterix Messages, one category for each specific application using this encoding format.

The use and definition of this categories are as followed:
Data Categories 000 to 127 for standard civil and military applications.
Data Categories 128 to 240 reserved for special civil and military applications.
Data Categories 241 to 255 used for both civil and military non-standard applications.

For our project the information we are going to need is included in category 021. This category describes the structure of the message from a CNS/ATM ground station, also covers a multitude of ground stations with different types of services. Those services are: ADS-B, TIS-B, FIS_B, GRAS and MLT.
So we can say that ADS-B messages are part of Asterix data categories.

The eDEP program is able to send messages with category 021 and category 244, this last category is reserved for the exchange of information for the SASS-C system. The SASS-C is a software toolbox developed by Eurocontrol to provide standarised methods and tools for assessing the performance of Surveillance Infrastructures.

3.3 The ADS-B messages

The Automatic Dependent Surveillance (ADS) is a surveillance technology for tracking aircrafts as part of the next Generation Air Transportation System (NextGen). The aim of NextGen is to guide and track air traffic more precisely and efficiently to save fuel, reduce noise and pollution through a continuous extended of improvements and upgrades.

Through the data from the positioning satellite system (GNSS), the aircraft knows its exact position and then broadcasts periodically its position and some other important informations with ADS-B messages to the ground station and to the other aircrafts equipped with ADS-B technology. The aircrafts using this technology have a little screen inside the cockpit to see the information received.

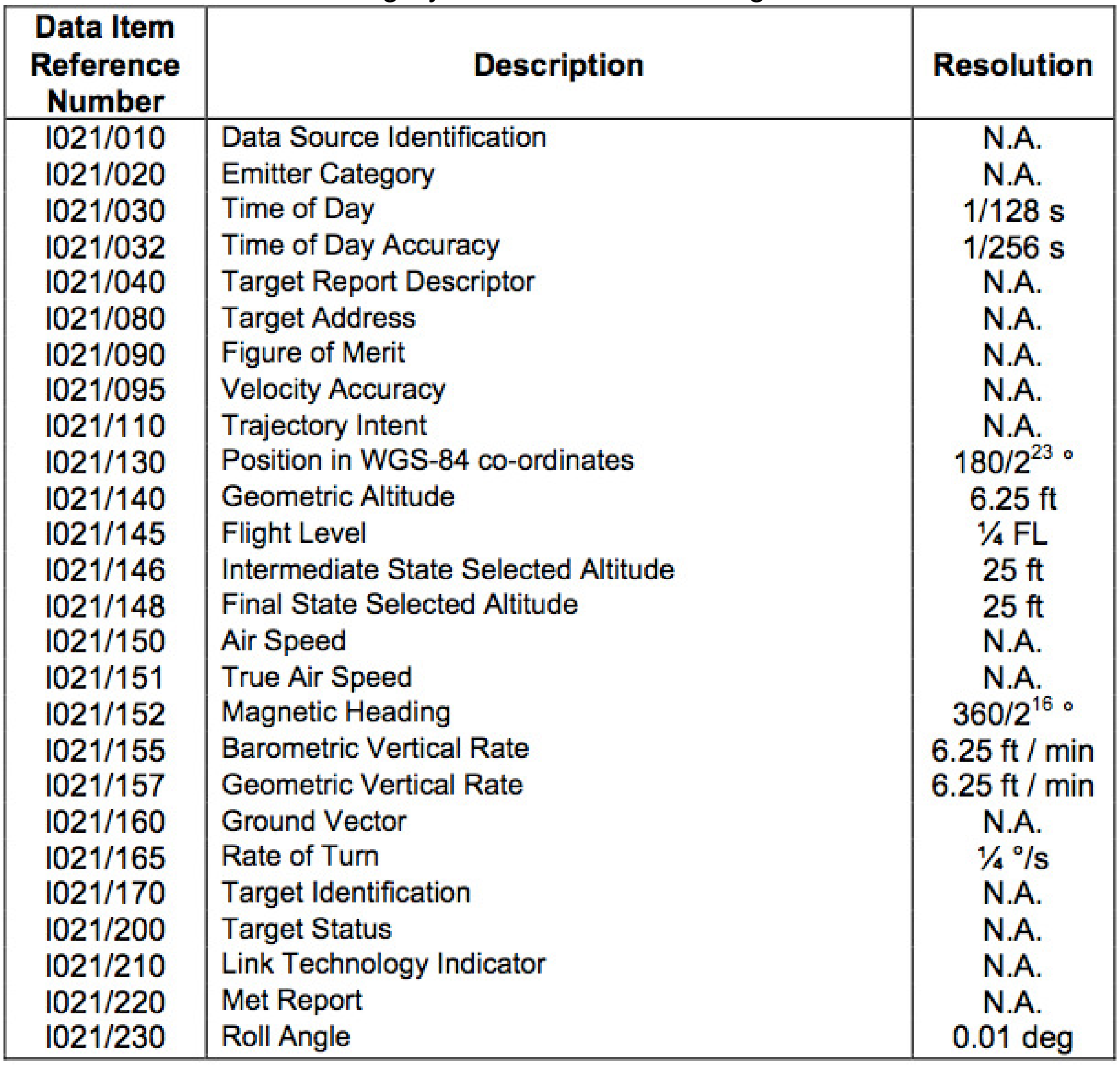
The ADS-B system works completely different to other systems, every second the system broadcast its position and other important information calculated on board to the controllers and airspace users without any action of the pilot or controller. The ADS-B messages are broadcast in the band of 1090 MHz radio frequency but they are also carried on a Universal Access Transceiver (UAT) in the 978 MHz band. The broadcast of the information don't need an acknowledgment or reply from the users receiving the information, so the ADS-B system never knows who receives the information.

ADS-B can be used over several different data link technologies:

Mode-S Extended Squitter operating at 1090 MHz
Universal Access Transceiver (978 MHz UAT)
VHF Data Link (VDL Mode 4)

All data sent through an ADS-B message are the following ones shown in the table.

Table 1: Data Items of Category 021 in ADS-B messages.

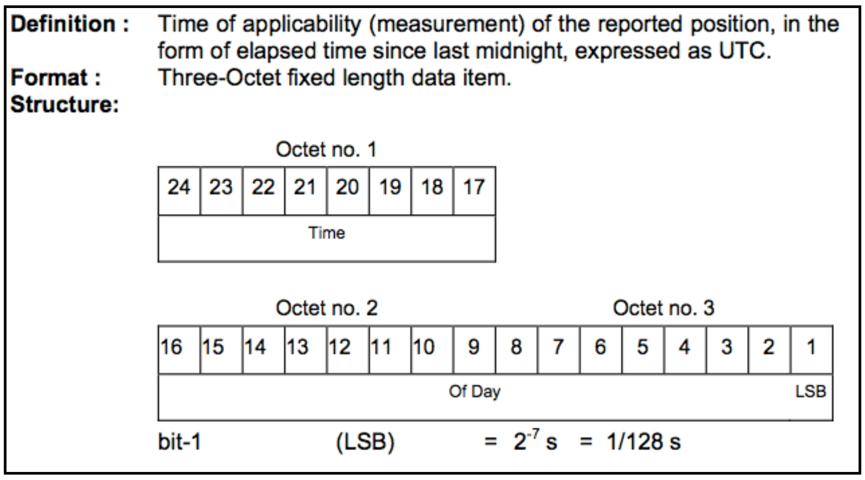


Data Item Reference Number	Description	Resolution
I021/010	Data Source Identification	N.A.
I021/020	Emitter Category	N.A.
I021/030	Time of Day	1/128 s
I021/032	Time of Day Accuracy	1/256 s
I021/040	Target Report Descriptor	N.A.
I021/080	Target Address	N.A.
I021/090	Figure of Merit	N.A.
I021/095	Velocity Accuracy	N.A.
I021/110	Trajectory Intent	N.A.
I021/130	Position in WGS-84 co-ordinates	$180/2^{23}$ °
I021/140	Geometric Altitude	6.25 ft
I021/145	Flight Level	¼ FL
I021/146	Intermediate State Selected Altitude	25 ft
I021/148	Final State Selected Altitude	25 ft
I021/150	Air Speed	N.A.
I021/151	True Air Speed	N.A.
I021/152	Magnetic Heading	$360/2^{16}$ °
I021/155	Barometric Vertical Rate	6.25 ft / min
I021/157	Geometric Vertical Rate	6.25 ft / min
I021/160	Ground Vector	N.A.
I021/165	Rate of Turn	¼ °/s
I021/170	Target Identification	N.A.
I021/200	Target Status	N.A.
I021/210	Link Technology Indicator	N.A.
I021/220	Met Report	N.A.
I021/230	Roll Angle	0.01 deg

We are going to look to the structure of the data we are going to use in this project.

3.3.1 Time of day

In the figure below we can see the structure of the time in the ADS-B message. The structure is composed by three octets and the accuracy of the data is 2^{-7} seconds.



Definition : Time of applicability (measurement) of the reported position, in the form of elapsed time since last midnight, expressed as UTC.
Format : Three-Octet fixed length data item.
Structure:

Octet no. 1

24	23	22	21	20	19	18	17
Time							

Octet no. 2 / Octet no. 3

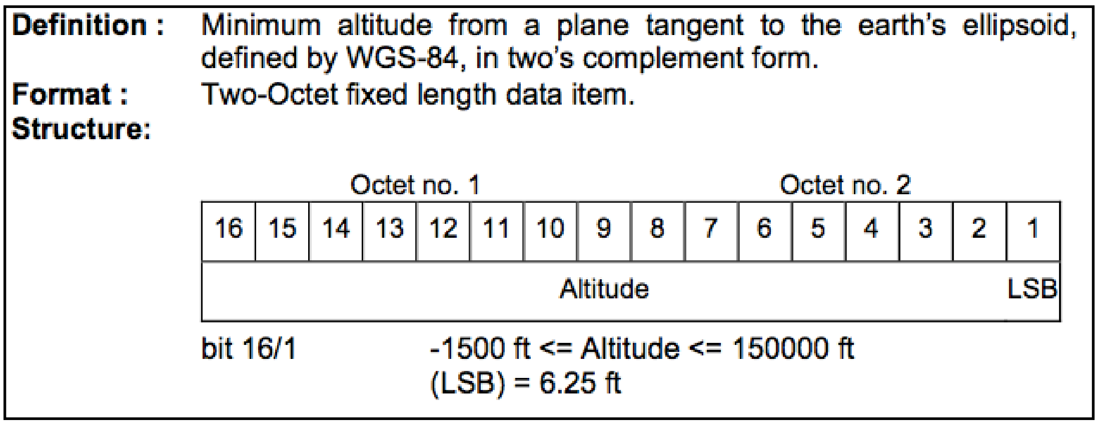
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Of Day															LSB

bit-1 (LSB) = 2^{-7} s = 1/128 s

Figure 14: Structure of Time of day data

3.3.2 Geometric Altitude

The geometric altitude data is received in two octets. The altitude is measured from -1500 feet up to 150000 feet with an accuracy of 6.25 feet.



Definition : Minimum altitude from a plane tangent to the earth's ellipsoid, defined by WGS-84, in two's complement form.
Format : Two-Octet fixed length data item.
Structure:

Octet no. 1 / Octet no. 2

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Altitude															LSB

bit 16/1 -1500 ft <= Altitude <= 150000 ft
(LSB) = 6.25 ft

Figure 15: Structure of Geometric Altitude data

3.3.3 Position in WGS-84 Co-ordinates

The position coordinates data is received in 6 octets. The three firsts octets include the longitude data and the other three include the latitude data.

Definition : Position in WGS-84 Co-ordinates.
Format : Six-octet fixed length Data Item.
Structure:

Octet no. 1								Octet no. 2							
48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
Latitude in WGS - 84															

Octet no. 3								Octet no. 4							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
							LSB								

Octet no. 5								Octet no. 6							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Longitude in WGS - 84															LSB

bits-48/25	(Latitude)	In WGS.84 in two's complement. Range -90 <= latitude <= 90 deg. LSB = $180/2^{23}$ degrees = $2.145767 * 10^{-05}$ degrees. This corresponds to a resolution of at least 2.4 meters
bits-24/1	(Longitude)	In WGS.84 in two's complement. Range -180 <= longitude < 180 deg. LSB = $180/2^{23}$ degrees = $2.145767 * 10^{-05}$ degrees. This corresponds to a resolution of at least 2.4 meters.

Figure 16: Structure of WGS 84 Co-ordinates data

3.3.4 Flight Level

The Flight Level data is received in two octets with a range from -15 FL to 1500 FL. The accuracy of this data is a quarter of FL.

Definition : Flight Level from barometric measurements, not QNH corrected, in two's complement form.
Format : Two-Octet fixed length data item.
Structure:

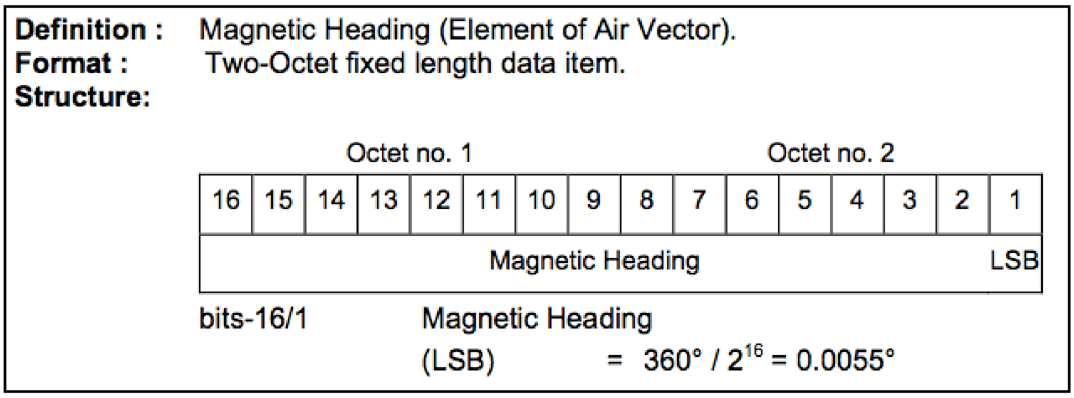
Octet no. 1								Octet no. 2							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Flight Level															LSB

bit 16/1	-15 FL <= Flight Level <= 1500 FL (LSB) = 1/4 FL

Figure 17: Structure of Flight Level data

3.3.5 Magnetic heading

The magnetic heading is received in two octets with an accuracy of 0.0055 degrees.



Definition : Magnetic Heading (Element of Air Vector).
Format : Two-Octet fixed length data item.
Structure:

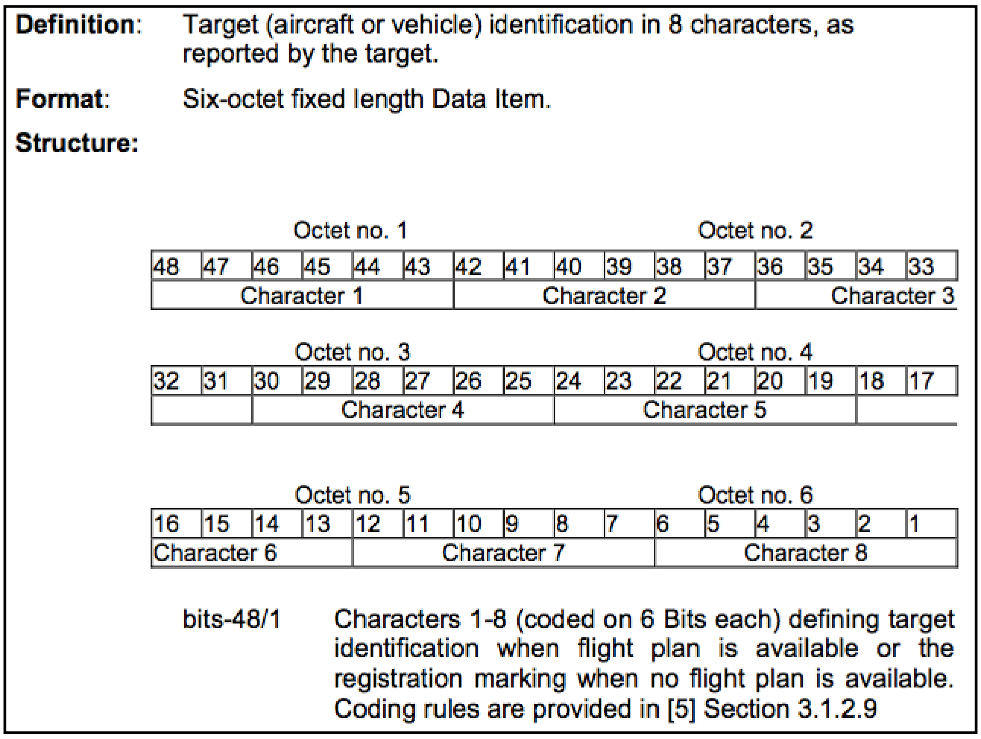
Octet no. 1								Octet no. 2							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Magnetic Heading															LSB

bits-16/1 Magnetic Heading
(LSB) = $360° / 2^{16} = 0.0055°$

Figure 18: Structure of magnetic heading data

3.3.6 Target Identification

The data of the identification of the aircrafts is received in 6 octets. Each octets is coded in 6 bits. Octet 8 defines the first character of the identification, octet 7 defines the second character and so on until octet 1 that defines the last character of the target identification.



Definition: Target (aircraft or vehicle) identification in 8 characters, as reported by the target.

Format: Six-octet fixed length Data Item.

Structure:

Octet no. 1								Octet no. 2							
48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
Character 1						Character 2						Character 3			

Octet no. 3								Octet no. 4							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
		Character 4						Character 5							

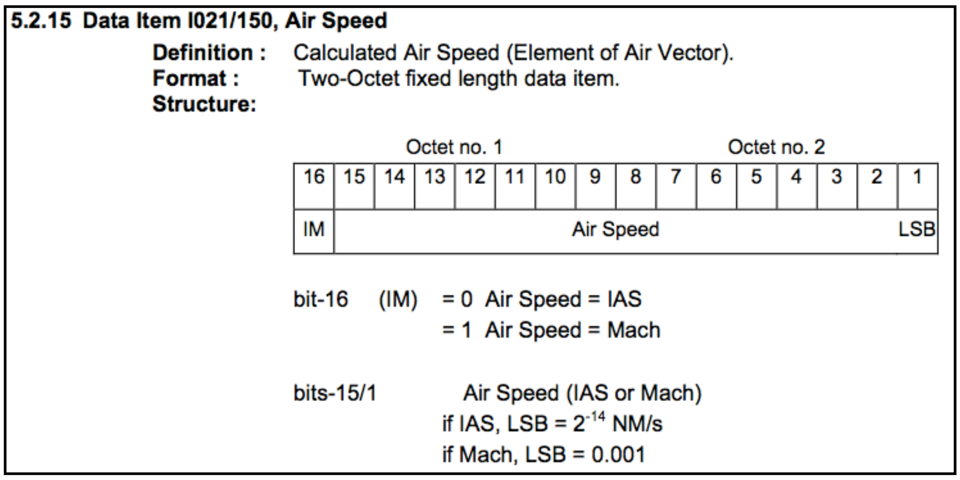
Octet no. 5								Octet no. 6							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Character 6				Character 7						Character 8					

bits-48/1 Characters 1-8 (coded on 6 Bits each) defining target identification when flight plan is available or the registration marking when no flight plan is available. Coding rules are provided in [5] Section 3.1.2.9

Figure 19: Structure of Target Identification data

3.3.7 Air Speed

Data from Air Speed is the one we are going to use in our project because all the other data related to speed as True Air Speed or Velocity Accuracy have no data in the message.
Air Speed data is received is two octets with an accuracy of 0,001.



5.2.15 Data Item I021/150, Air Speed

Definition : Calculated Air Speed (Element of Air Vector).
Format : Two-Octet fixed length data item.
Structure:

Octet no. 1								Octet no. 2							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
IM	Air Speed														LSB

bit-16 (IM) = 0 Air Speed = IAS
= 1 Air Speed = Mach

bits-15/1 Air Speed (IAS or Mach)
if IAS, LSB = 2^{-14} NM/s
if Mach, LSB = 0.001

Figure 20: Structure of True Air Speed data

The ADS-B technology is not yet compulsory for the aircrafts. Every airspace has its own programs to implement this technology[2].

- In United States, they are implementing the ADS-B technology with UAT technology for regional level and Extended Squitter for global level. In 2020 the United States will require to almost all the aircrafts operating within its airspace to be equipped with this technology.

- The South Pacific airspace is covered with ADS-C thanks to the use of satellite communication and standard FANS-1/A.

- In the Asian airspace, Japan has introduced ADS-C in its Tokyo's oceanic airspace and they are evaluating the ADS-B.

- In the Oceanic airspace, Australia has began its operational evaluation project of ADS-B based on Extended Squitter. They are planning to install 40 ADS-B ground stations and the will equipped several aircrafts too with ADS-B.

- Referring to Europe, there are several programs trying to find out and testing if the ADS technology is worthwhile for the nearest future. AENA is ahead of some programs as SACCAN or SEAP. The main idea is to prove that ADS is a technology that will make a safer airspace. This technology is not yet validated.

[2] Reference: http://www.aena.es/csee/Flash/html/adsImplantacion.jsp

4. Architecture of the simulation environment

In this project we are going to use the latest Java Eclipse Platform which is: Java Eclipse Helios. With this platform we are going to be able to run the eDEP program. Once eDEP program is running, we are going to send all the aircraft data through an UDP port from eDEP simulation to our C# platform.

Our C# platform is MonoDevelop in which, once receiving the data from Eclipse, we are going to decode that information and then we are going to make all the calculations for our conflict detection algorithm. This algorithm will be described in the next point but now I would like to specify the main structure of the code to understand better the next point.

When do we know that the aircrafts are in possible conflict?

Knowing all the data from each aircraft received through the UDP port and taking into assumption that our UAV is controlled by us all time. The first thing to know is the distance between our UAV and the rest of the traffic planes. Once we know that distance we need to fix a imaginary circle around our UAV as a protection zone. The zone will have a radius of 10 kilometres.

When we have an aircraft crossing this 10 kilometres boundary we need to know if they are flying with a vertical secure separation. If this vertical separation is 1000 feet there will be no conflict. When the vertical separation is less than 1000 feet, the next thing we need to know is the convergence of their headings. The different convergence of headings is explained in the following point.

In case their headings converge in a look ahead point, is now when we calculate the coordinates of this conflict point and then the distance from each aircraft to the collision point. We also calculate the time that will last each aircraft to arrive to that specific point. If the difference between each aircrafts' time is more than 2 minutes then we are not going to do nothing because there is no conflict situation.

Once the time difference is less than 2 minutes we need to change heading from one of the aircrafts to avoid the collision. In our case, we just want to detect the possible collision. Conflict avoidance is explained a little bit in our previous work and there are two different ways explained to solve the conflict among a huge possibilities of conflict resolution ways because every situation is different and there will be one resolution for each situation that exists.

In the following figure we can see a diagram that shows how the C# code files are related between them Each name is related to a different file of our Csharp program. The "CAT21 Message", "Asterix Message" and "Asterix Factory" are the files that decode the information received by eDEP program and then the data is sent directly to "Myplane" and "Traffic" files. All the code the is related as

the arrows show. Each files is directly related with one or more other files, that way it is easy to get a simple code easy to understand.
The files “Point” and “WgsPoint” are just used to convert the position coordinates received in degrees by the eDEP program to UTM positions.

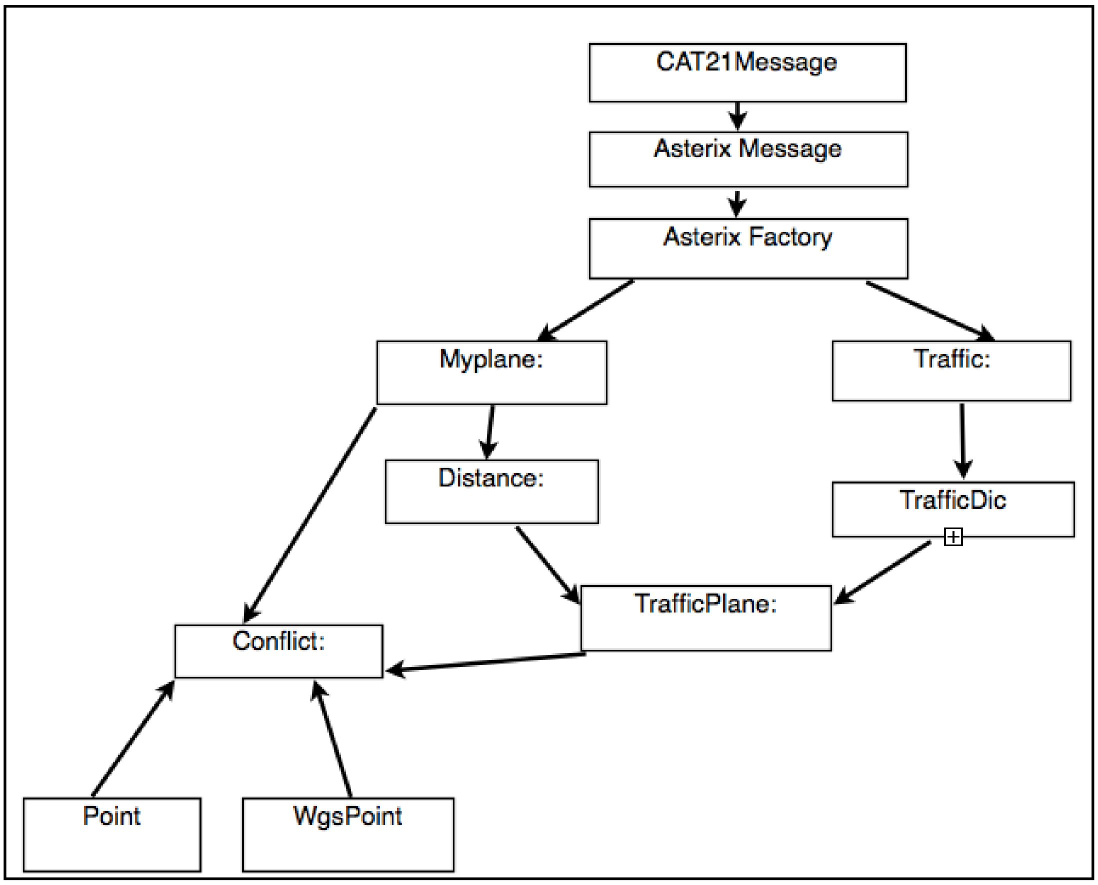


Figure 21: Diagram of the C# code files

Referring to the main aspect of how our C# code works, we first have to say that our code has been implemented with threading. That means that while we are receiving data from the eDEP simulation our program will not make and calculation and once we have received the package of data our code will start automatically to run the algorithm in order to calculate all we have implemented. We are going to receive data packages each 6 seconds from the simulation. In this case we can separate our code files in two different blocks. The first block that will be called the writer is the one in charge of decoding the information received and the one that stores the information in a local dictionary in order to keep the information. In this first block the name files among others are: CAT21 Message, Asterix Message, Asterix Factory, Traffic, *Myplane* The second block is called the consumer that will use all the information stored in the local dictionary to make all the calculations implemented. This last block is composed by the rest of the files that are not included in the writer block.

5. Design and implementation of the collision detection algorithm

The idea to implement the conflict detection algorithm is very simple. Through data received by ADS-B messages seen in point 3.2 we know everything we need to make the calculations.

In the next figure we can see a schematic draw that represents two aircrafts(1 and 2) in conflict and the estimated point of conflict P.

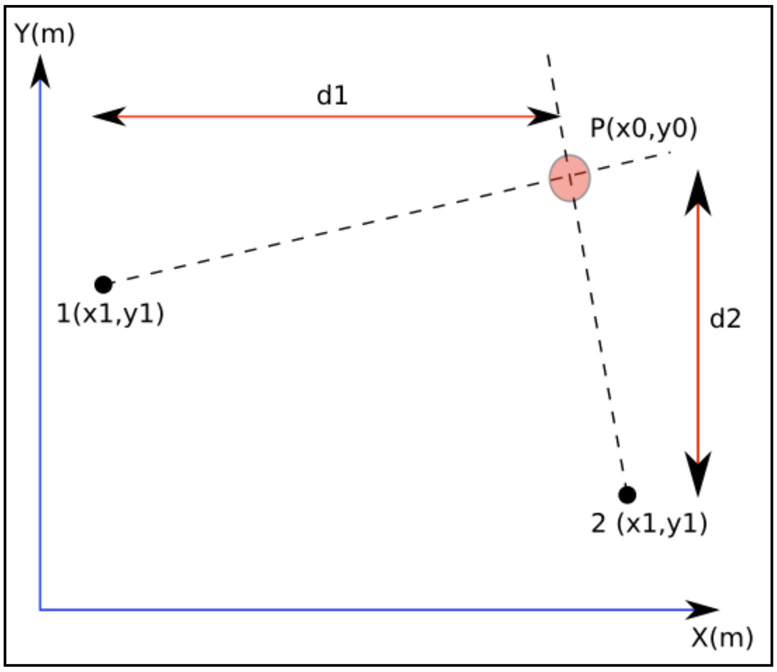


Figure 22: Schematic draw of a conflict

The ADS-B message give us the exact position of the aircrafts in degrees. Through a specific function explained in the previous point we get the position in metres that will be much more easier to make the calculations.

The points 1 and 2 (representing the aircrafts) can be described by the next equation:

$$y_1 = m_1 * x_1 + b_1$$
$$y_2 = m_2 * x_2 + b_2$$

As we know the exact position of the aircrafts in this equations we know the value of the variable “y” and “x” thanks to a code that converts from position in degrees to position in UTM coordinates (meters).

Then we have two equations with two unknown variables “m” and “b”. The “m” variable is the slope of the equation so knowing the magnetic heading from the ADS-B message we are going to be able to find “m”.

The problem to determine the “m” variable is that in theoretical estimations or calculations the magnetic heading is 0 degrees at the North and then 90 degrees at the East, but in real life the East heading equals to 0 degrees and the north heading equals 90 degrees. Taking this two assumptions, we need to determine a equation that help us transforming from one heading to the other. The equation will have this form: $\mu = ß * x + \partial$.

The final equation will be: $\mu = (-1) * x + 90$

$\mu = 90 - ß$. (ß is equal to our magnetic heading)

$m = tg(\mu)$ (rad)

At this point we have the slope of our aircraft position equation in radians but we need it in degrees so we transform it: $m = tg(\mu) * (180 / \pi)$.

Now we just need to find the value of the b variable for both equations, one for each aircraft in conflict.

Lets work out the value of b.

$y_1 = m_1 * x_1 + b_1$
$b_1 = y_1 - m_1 * x_1$

$y_2 = m_2 * x_2 + b_2$
$b_2 = y_2 - m_2 * x_2$

Now that we have work out the values of the slope m and the independent variable b, we are going the find the conflict point. For that reason, in both equation of both aircraft un conflict the variables X and Y will have the same value. When both equation will have the same value it will mean that the two aircraft will be in the same place.

So our next step is to equal the two equations:

$m_1 * x_1 + b_1 = m_2 * x_2 + b_2$
$x_1 = x_2$
$x = (b_1 - b_2) / (m_2 - m_1)$

Once we have the value of the variable x we just need to put that value in one of the equations either aircraft 1 or aircraft 2 and we will find the value of variable y.

At this point we have the equation of the aircrafts in conflict and the exact point of conflict.

Once we have all of this, the next step will be to determine if the point found as the conflict point is a real conflict point for our case or just a point in the past where the two headings crossed each other in a different time interval.

In the next figure we can see this two different points just explained.

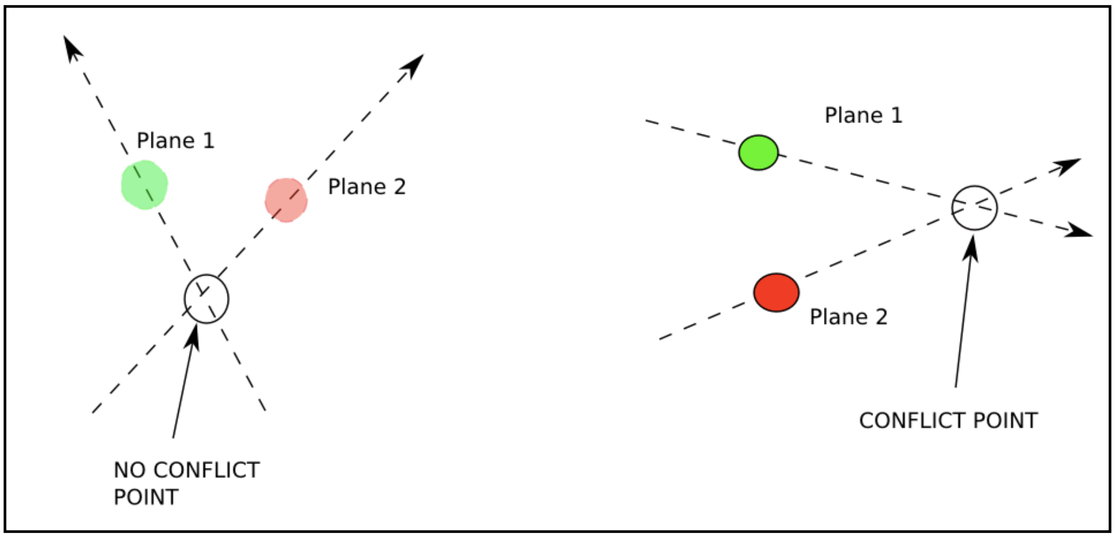


Figure 23: Two different cases of convergence points

This heading interception is called heading convergence. To calculate this we are going to calculate the equation of a plane knowing three points that belong to it.
The general equation of this plane is: π: Ax + By+ Cz + D = 0.

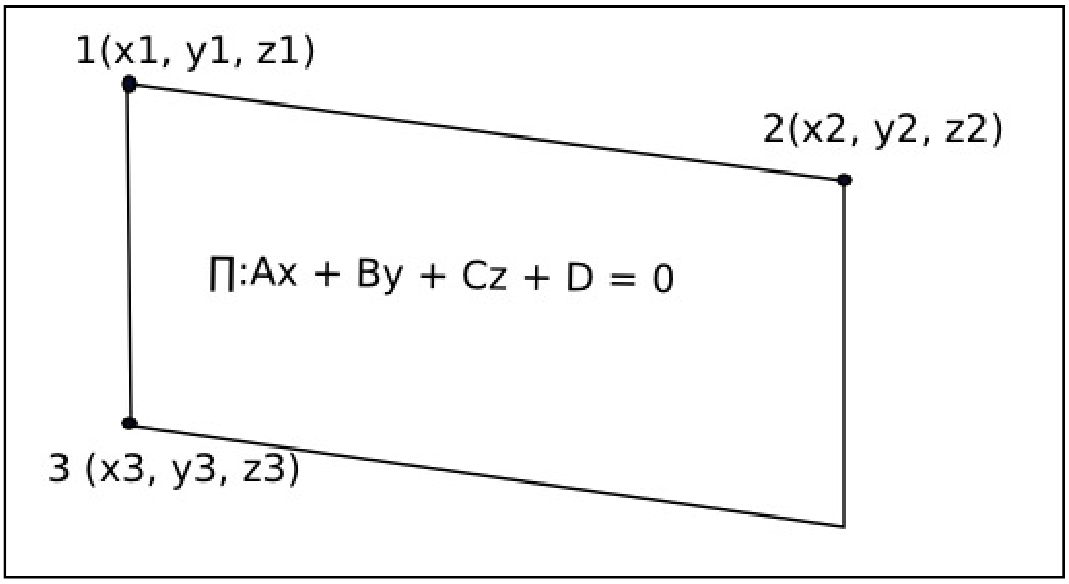


Figure 24: Schematic draw of a plane with three points

As it is shown in the figure above, point 1 and 2 are the aircrafts in conflict and point 3 has the same coordinates than point 1 but the only difference with point

1 is that the value of the variable z for point 3 is 0.Point 3 is chosen with this specific coordinates because it will help to make easy the following calculations. With this assumption we will always be able to create this plane and able then to calculate if there is convergence.

Calculations to create the plane and resolve convergence.

$$12 = (x_2 - x_1; y_2 - y_1; z_2 - z_1) = (A; B;\Gamma)$$

$$13 = (x_3 - x_1; y_3 - y_1; z_3 - z_1) = (\Delta; E;K)$$

$$x = x_1 + A\lambda + \Delta\mu$$

The result of this two first equations are the subtraction o each component of

$$y = y_1 + B\lambda + E\mu$$

$$z = z_1 + \Gamma\lambda + K\mu$$

Now we are going to calculate de coefficients from the general plane equation. This coefficients are calculated as we calculate the determinant of 2x2 matrix.

$$A = \begin{bmatrix} B & E \\ \Gamma & K \end{bmatrix} = (B\times K - E\times\Gamma)$$

$$B = \begin{bmatrix} \Lambda & \Delta \\ \Gamma & K \end{bmatrix} = (\Lambda\times K - \Delta\times\Gamma)$$

$$C = \begin{bmatrix} A & \Delta \\ B & E \end{bmatrix} = (A \times E - \Delta \times B)$$

Once we have the three first coefficients for the plane equation we just need to replace it on the equation and with the UTM coordinates from the conflict point (x, y, z) we will be able to determine the last coefficient D.
This coefficient is going to be very useful because for the same conflict point between two aircrafts while time increases the coefficient D must decrease in order to get a conflict point in a look ahead time if not there will be just

$$D = -(A \times x + B \times y + C \times z)$$

convergence between headings in the past but not in the future.

Once we have all this information, we need to calculate the distance from each aircraft to the conflict point and the time they are going to take to it. The variable who has the subindex equal to 0 are the UTM coordinates from the conflict point.

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

This last equation is going to tell us the time that takes to each aircraft to get to the conflict point knowing their speed and distance to the conflict point

$$t_i = \frac{d_i}{v_i}$$

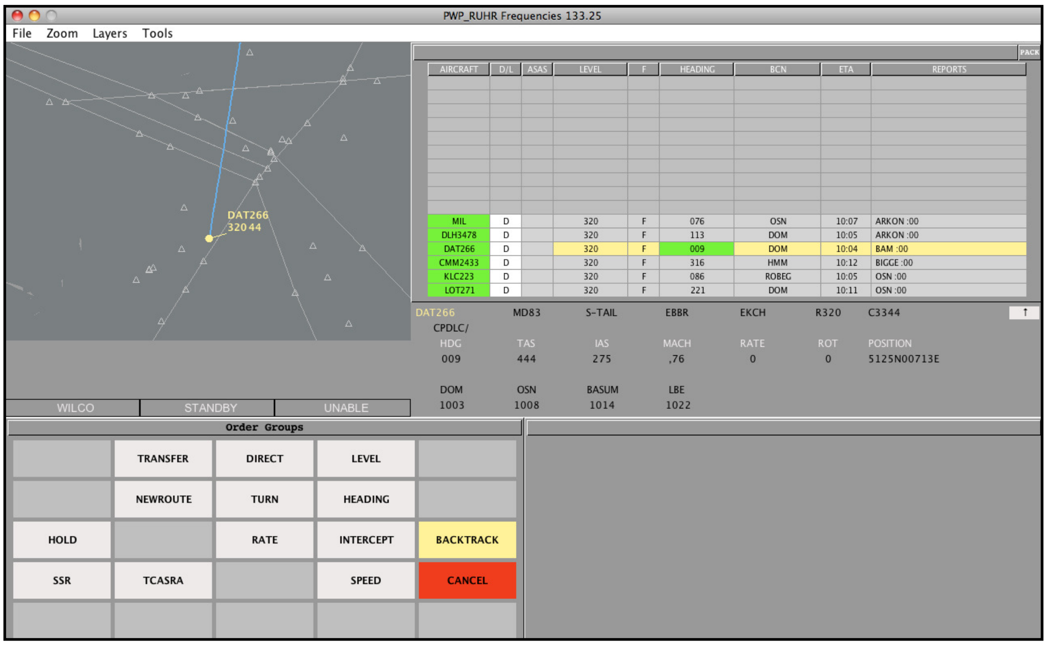
previously calculated with the above equation.

6. Evaluation

The evaluation of our results is going to be separated in two different scenarios:
Conflict point is behind the aircrafts and in the opposite direction of their current headings.
Conflict point is in a look ahead time in front of the aircrafts and collision will take place if we don't change any parameter of the aircrafts.

6.1 Conflict point in a look ahead time

In this scenario we are going to prove that our Csharp program detects the possible collision between two aircrafts. However, the same program detects conflicts between more than 2 aircrafts but it will be easy to show how it works with two aircrafts in conflict detection.
For this demonstration our UAV's target identification will be: DAT266. In the figures 25 and 26 we can see all the information regarding our UAV and the other aircraft involved in the collision, respectively. We can also see its routes plotted at the top of the left side.

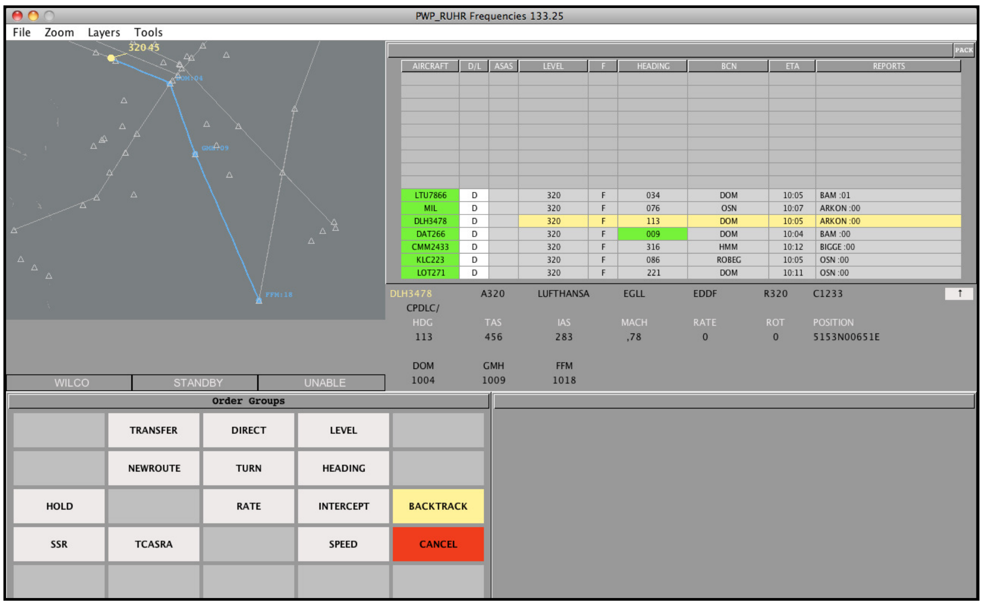


AIRCRAFT	D/L	ASAS	LEVEL	F	HEADING	BCN	ETA	REPORTS
MIL	D		320	F	076	OSN	10:07	ARKON :00
DLH3478	D		320	F	113	DOM	10:05	ARKON :00
DAT266	D		320	F	009	DOM	10:04	BAM :00
CMM2433	D		320	F	316	HMM	10:12	BIGGE :00
KLC223	D		320	F	086	ROBEG	10:05	OSN :00
LOT271	D		320	F	221	DOM	10:11	OSN :00

```
DAT266    MD83    S-TAIL   EBBR    EKCH    R320    C3344
  CPDLC/
  HDG      TAS     IAS      MACH    RATE    ROT     POSITION
  009      444     275      ,76     0       0       5125N00713E

  DOM      OSN     BASUM    LBE
  1003     1008    1014     1022
```

Figure 25: eDEP' screenshot of our UAV route information, showing all the information that can be changed of an aircraft in the simulation and also the flight plan.



AIRCRAFT	D/L	ASAS	LEVEL	F	HEADING	BCN	ETA	REPORTS
LTU7866	D		320	F	034	DOM	10:05	BAM :01
MIL	D		320	F	076	OSN	10:07	ARKON :00
DLH3478	D		320	F	113	DOM	10:05	ARKON :00
DAT266	D		320	F	009	DOM	10:04	BAM :00
CMM2433	D		320	F	316	HMM	10:12	BIGGE :00
KLC223	D		320	F	086	ROBEG	10:05	OSN :00
LOT271	D		320	F	221	DOM	10:11	OSN :00

DLH3478 A320 LUFTHANSA EGLL EDDF R320 C1233
CPDLC/

HDG	TAS	IAS	MACH	RATE	ROT	POSITION
113	456	283	,78	0	0	5153N00651E

DOM	GMH	FFM
1004	1009	1018

Figure 26: eDEP' screenshot for the aircraft in possible collision with our UAV showing the same information than the previous figure.

After a few minutes of simulation the conflict detection take place as we can see in figure 27 where we have marked a black circle. The aircrafts involved in the collision are DLH3478 and DAT266. The black circle identifies the collision point and we know that this point will be inside the circle. At this point we are able to change headings and avoid the collision.

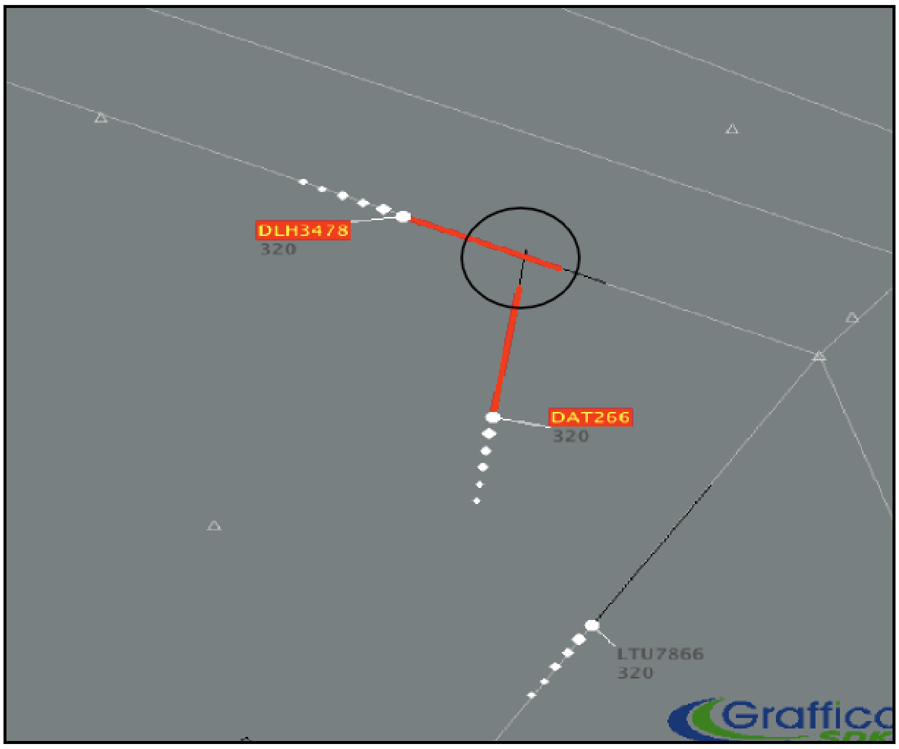
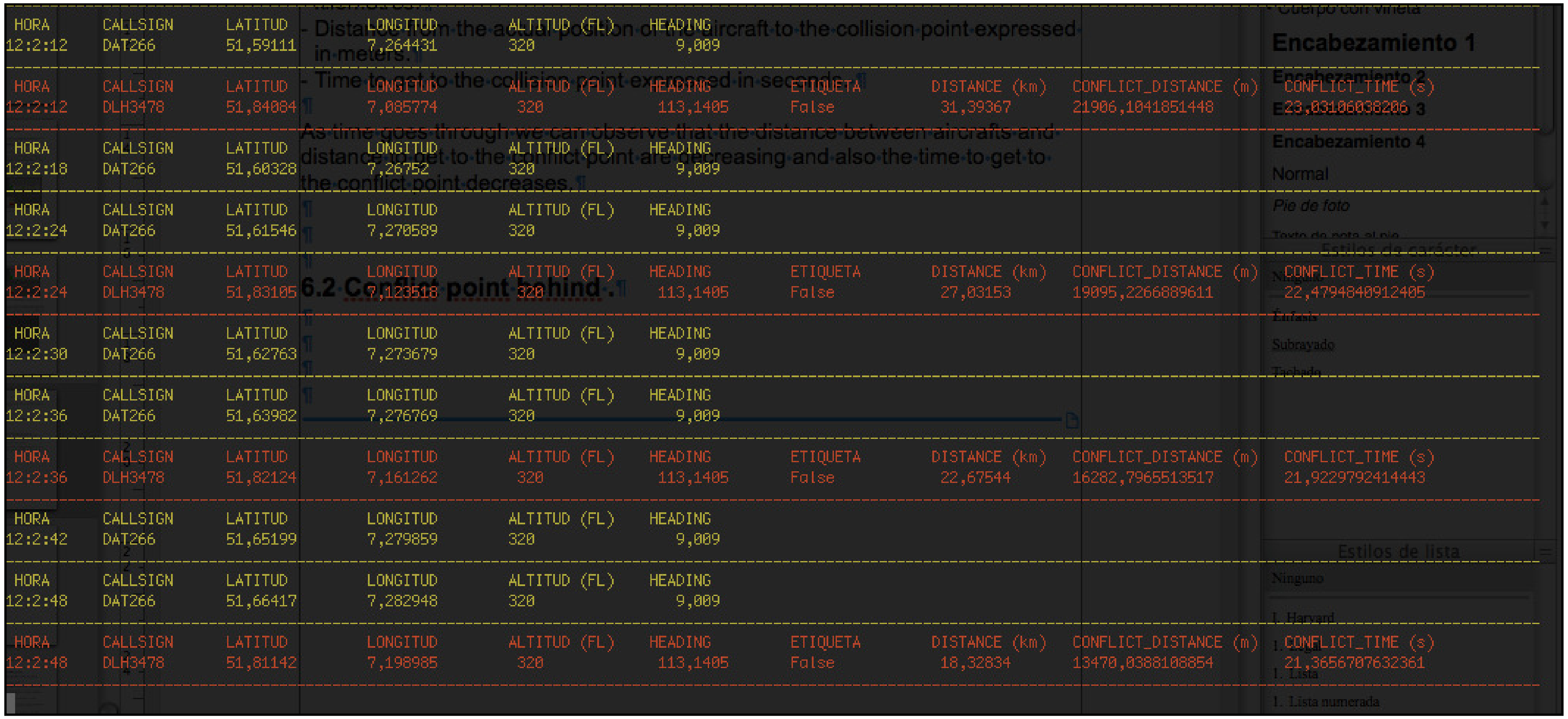


Figure 27: eDEP 'screenshot for conflict detection involving two aircrafts.

The next figure, figure 28, shows the parameters of the aircrafts of the eDEP simulation printed in the Csharp console that shows the evolution of the conflict.
In yellow we can see the information of our UAV:
Callsign.
Latitude position.
Longitude position.
Flight level.
Heading.

The parameters coloured in red show the parameters of the other aircraft involved in the conflict:
Callsign
Latitude position
Longitude position
Flight level
Heading
Information if the aircraft is inside the security area of the UAV.
Distance between the two aircrafts involved in the conflict expressed in kilometres.
Distance from the actual position of the aircraft to the collision point expressed in meters.
Time to get to the collision point expressed in seconds.

As time goes through we can observe that the distance between aircrafts and distance to get to the conflict point are decreasing and also the time to get to the conflict point decreases.



```
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:12  DAT266     51,59111  7,264431    320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING    ETIQUETA    DISTANCE (km)  CONFLICT_DISTANCE (m)  CONFLICT_TIME (s)
12:2:12  DLH3478    51,84084  7,085774     320          113,1405   False        31,39367      21906,1041851448       23,03106038206
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:18  DAT266     51,60328  7,26752     320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:24  DAT266     51,61546  7,270589    320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING    ETIQUETA    DISTANCE (km)  CONFLICT_DISTANCE (m)  CONFLICT_TIME (s)
12:2:24  DLH3478    51,83105  7,123518     320          113,1405   False        27,03153      19095,2266889611       22,4794840912405
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:30  DAT266     51,62763  7,273679    320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:36  DAT266     51,63982  7,276769    320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING    ETIQUETA    DISTANCE (km)  CONFLICT_DISTANCE (m)  CONFLICT_TIME (s)
12:2:36  DLH3478    51,82124  7,161262     320          113,1405   False        22,67544      16282,7965513517       21,9229792414443
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:42  DAT266     51,65199  7,279859    320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING
12:2:48  DAT266     51,66417  7,282948    320             9,009
HORA     CALLSIGN   LATITUD   LONGITUD    ALTITUD (FL)  HEADING    ETIQUETA    DISTANCE (km)  CONFLICT_DISTANCE (m)  CONFLICT_TIME (s)
12:2:48  DLH3478    51,81142  7,198985     320          113,1405   False        18,32834      13470,0388108854       21,3656707632361
```

Figure 28: Csharp console showing the parameters received from eDEP simulation.

6.2 Conflict point behind

The next scenario is the one involving two aircrafts. There will be conflict for short time but then the conflict will disappear because one of the aircrafts reaches the conflict point before the other with a lap of time of two minutes before the other aircraft reaches the estimated conflict point.

In figure 29, we can see the routes of each aircraft, in red we have DAT266 ‘s route and in green DLH3478’s route, and the estimated point of conflict that would be inside the blue circle. This screenshot shows that the conflict point is behind both aircrafts so once the last aircraft has passed the estimated point, the conflict disappears.

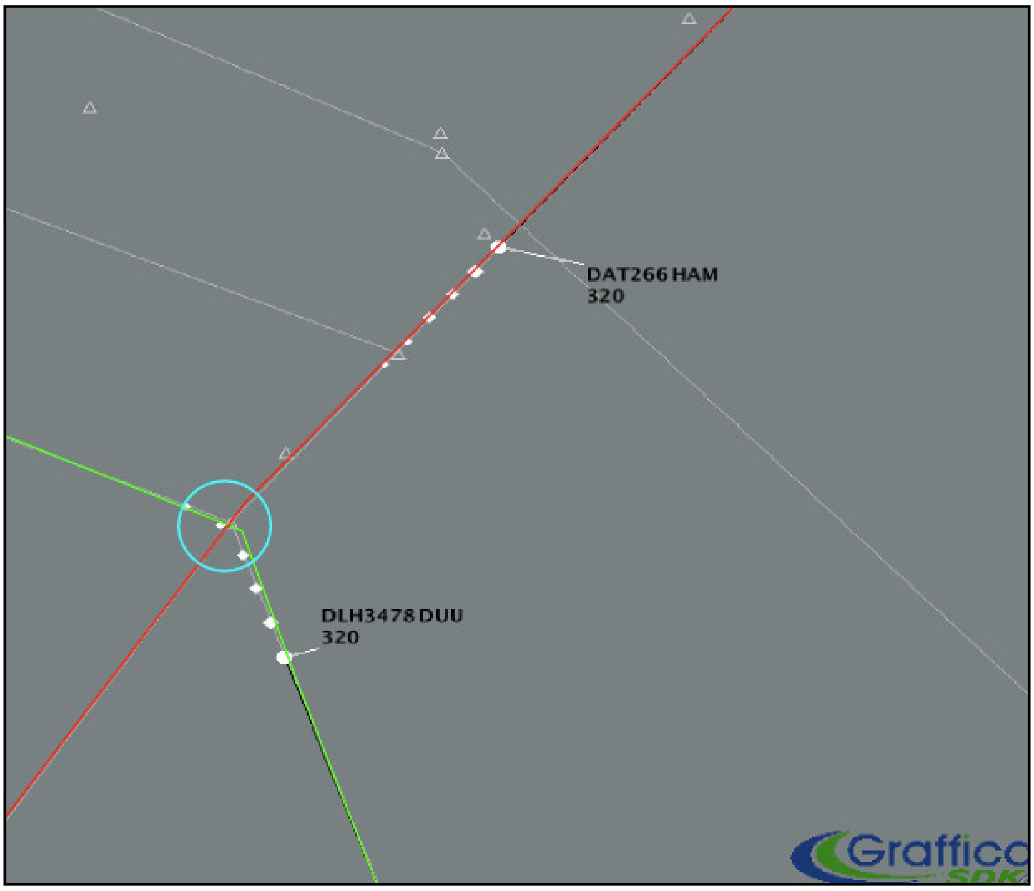
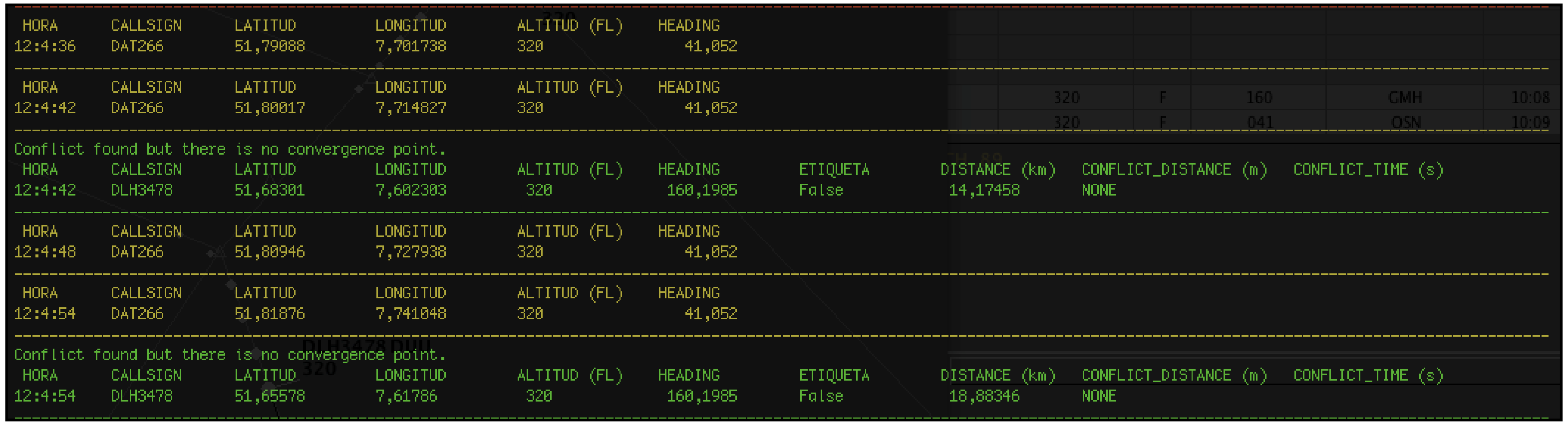


Figure 29: eDEP ’screenshot showing in red our UAV’s flight plan and in green DLH3478’s flight plan. Also the estimated point of conflict is situated inside the blue circle.

Regarding the Csharp code, once the conflict disappears our program will not calculate any more the distance conflict and conflict time because its very unlikely that our two aircrafts will have an intersection in their flight plans. As it is shown in figure 30, once the last plane has passed through the estimated point of conflict, the distance between the two aircrafts will start to increase.



```
 HORA     CALLSIGN    LATITUD     LONGITUD    ALTITUD (FL)   HEADING
12:4:36   DAT266      51,79088    7,701738    320              41,052
---------------------------------------------------------------------------------------------------
 HORA     CALLSIGN    LATITUD     LONGITUD    ALTITUD (FL)   HEADING
12:4:42   DAT266      51,80017    7,714827    320              41,052
---------------------------------------------------------------------------------------------------
Conflict found but there is no convergence point.
 HORA     CALLSIGN    LATITUD     LONGITUD    ALTITUD (FL)   HEADING      ETIQUETA     DISTANCE (km)   CONFLICT_DISTANCE (m)   CONFLICT_TIME (s)
12:4:42   DLH3478     51,68301    7,602303     320             160,1985   False         14,17458       NONE
---------------------------------------------------------------------------------------------------
 HORA     CALLSIGN    LATITUD     LONGITUD    ALTITUD (FL)   HEADING
12:4:48   DAT266      51,80946    7,727938    320              41,052
---------------------------------------------------------------------------------------------------
 HORA     CALLSIGN    LATITUD     LONGITUD    ALTITUD (FL)   HEADING
12:4:54   DAT266      51,81876    7,741048    320              41,052
---------------------------------------------------------------------------------------------------
Conflict found but there is no convergence point.
 HORA     CALLSIGN    LATITUD     LONGITUD    ALTITUD (FL)   HEADING      ETIQUETA     DISTANCE (km)   CONFLICT_DISTANCE (m)   CONFLICT_TIME (s)
12:4:54   DLH3478     51,65578    7,61786      320             160,1985   False         18,88346       NONE
---------------------------------------------------------------------------------------------------
```

Figure 30: Csharp console showing the parameters received from eDEP simulation

7. Conclusions

To sum up, the objectives of this project at the beginning were to explain what a conflict is and how can be detected and solved. We also needed to understand how aircrafts get communicated with the ATC referring to ADS-B messages and our last point was all the informatics related with the eDEP simulator and C# platform.

At this point, we can say that the objectives of this Project have been achieved as we have seen along all the memory. We have seen what is a conflict and some methods for conflict detection. We also have seen that an UAV is just a part of an UAS Platform, the different uses and the most important point is that we have achieved to create a conflict detection algorithm with C# platform using different mathematic methods. We also have spoken about how we receive the data from the aircrafts, thanks to the future of data link technology, ADS-B messages, specially messages category 021, which gives us all the information needed about an aircraft.

This project can be complemented in the future with a conflict avoidance algorithm so we could create an autonomous algorithm for an UAV. The main idea would be to give a flight plan with specific waypoints and in case our UAV interferes in any flight plan from any aircraft flying around it, we should be able to change automatically our UAV's flight plan. This would have a positive impact in air traffic flow because we would make an algorithm that would disturb the other aircrafts as less as possible.

8. Bibliography

[1] http://www.eluniversal.com/2011/02/23/bajan-indices-de-accidentes-aereos.shtml [March, 2011]

[2] http://www.eurocontrol.fr/projects/edep/ [March, 2011]

[3] http://upcommons.upc.edu/pfc/bitstream/2099.1/9660/1/memoria.pdf [March, 2011]

[4] James K. Kuchar and Lee C. Yang, A review of conflict Detection and Resolution Modeling Methods, IEEE transactions on intelligent transportation systems, VOL 1, NO 2, DECEMBER 2000.

[5] RTCA Task force 3, Final report of RTCA Task force 3 Free Flight Implementation, October 26, 1995.

[6] H.A.P Blom, G.J. Bakker, P.J.G. Blanker, J. Daams, M.H.C. Everdij and M.B. Klompstra. Accident risk assessment for advanced ATM. National Lucht-en Rulmtevaartlaboratorium 1999.

[7] Nicolas Durand. Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion detrafic aérien. 5 octobre 2004.

[8] D.J. Brudnicki, K.S. Lindsay and A.L. McFarland. Assessment of field trials, algorithmic performance, and benefits of the User Request Evaluation Tool (URET) conflict probe. The MIRET corporation 1997.

[9] Hyo-Sang Shin, A. Tsourdos and B.A. White. UAV Conflict Detection and Resolution Using Differential Geometry. 2011

[10] J.M. Hoekstra, R.N.H.W van Gent and R.C.J. Ruigrok. Conceptual design of Free Flight with airborne separation assurance. National Lucht-en Rulmtevaartlaboratorium 1998

[11] Vu N. Duong and Karim Zeghal. Conflict Resolution Advisory for Autonomous Airborne Separation un Low-Density Airspace. 1997

[12] Claire Tomlin, George J. Pappas and Shankar Sastry. Conflict Resolution for Air Traffic Management: A study in Multiagent Hybrid Systems. IEEE transactions on automatic control, vol 43, no 4, april 1998.

[13] Douglas R. Isaacson and Heinz Erzberger. Desing od a conflict detection algorithm for the CENTER/TRACON automation system.

[14] Martin S. Eby and Wallace e. Kelly III. Free flight separation assurance using distributed algorithms.

[15] Nicolas Durant and Jean-Marc Alliot. Optimal Resolution of En Route Conflicts. Laboratoire d'optimisation globale.

[16] P. K. Menon, G.D. Sweriduk and B. Sridhar. Optimal Strategies for Free Flight Air Traffic Conflict Resolution.

[17] Lee C. Yang and James K. Kuchar. Performance Metric Alerting: A New Design Approach for Complex Alerting Problems. 2002

[18] Lee C.Yang and James K. Kuchar. Survey of conflict detection and resolution modeling methods. 1997

[19] Lee C. Yang and James K. Kuchar. Using intent information in probabilistic conflict. 1998

[20] http://en.wikipedia.org/wiki/Automatic_dependent_surveillance-broadcast#cite_note-upgarde-0 [March, 2011]

[21] http://www.obsa.org/PaginasOBSA/Navegacion/CiasAereas_AirEuropa_2.aspx [April, 2011]

[22] http://www.aena.es/csee/Flash/html/adsImplantacion.jsp [April, 2011]

[23] http://www.eurocontrol.int/asterix/public/subsite_homepage/homepage.html [June, 2011]

[24] http://www.eurocontrol.int/sass/public/standard_page/Overview.html [June, 2011]

[25] http://www.faa.gov/nextgen [July, 2011]