Project 1

ASTERIX DECODER

Projectes en Gestió del Trànsit Aeri

Grup 2:

Jaime Morales

Mauro Sánchez

Juan Pablo Pineda

Albert Marrugat

Quim Lucchini

Natalia Castellano

Contents

Introduction 2
Software Development 2
Planning 2
Classes 2
Fichero 2
CAT 10 3
CAT 21 3
Marker 3
Windows Forms 3
Load File 4
Data View 5
Simulation 6
Help 8
Extras 9
Computation of the probability of detection 9
Computation of the probability of identification 9
Future improvements 9

Introduction

The aim of this project is the thorough comprehension of the decryption of the ASTERIX Eurocontrol standard categories 10 and 21, as well as the development of a software to do so.

This document contains the detailed description of the decoding of the files as well as the development of the software. The programme must manage to achieve the correct decryption of the ASTERIX codes from Eurocontrol as well as to showcase the information correctly. In addition, this software shall also contain the tools to correctly display a simulation with the data extracted from the files. Furthermore, some functions the programme shall have, include the conversion of the data from the messages to CSV files and the conversion of aircraft trajectories into KML files.

Software Development

Planning

The development of the software was done in different phases and with previous planning to successfully achieve the objectives of the project. These steps were taken for the correct development of the programme within the given weeks:

1. Thorough study of the ASTERIX files given using the Asterix Inspector.
2. Creation of the class CAT10 and creation of the codes for the decryption of ASTERIX codes for category 10 messages.
3. Successful decryption of the CAT10 file, but still only shown in console.
4. Creation of the Windows Forms Project and first approach in the design of the software.
5. Creation of the class CAT21 and beginning of the programming to achieve the decoding of the CAT21 messages.
6. Development of the forms to load the files and for data viewing.
7. Successful connection between the forms.
8. Development of the code to decrypt CAT21 messages completed.
9. Beginning of the development for the simulation.
10. At the same time, studying the possibility to implement filters in the data view form to find only the desired information.
11. Creation of the class Marker for its implementation in the simulation code.
12. Development of the Simulation form to implement the simulation code, using the GMap.NET library to represent the required points in a Google Maps map.
13. Creation of the necessary code to achieve exportation of the data in CSV files.
14. Creation of the Help form to contain the instructions of use of the software.
15. Development of the code to export the simulation data into a KML file.
16. Implementation of the last visual characteristics and checking for last minute improvements of the code and the interface.

Classes

Fichero

This class is responsible for the reading of the file and classifying the messages between categories 10 and 21. The first thing we shall find in the definition of this class is the initialization of the categories 10 and 21 lists to be filled later. The class Fichero is initialized using a file's path, which is read using the method readFile().

To filter the messages, the method classifyMessage() is used, which consists of reading the category of each message and adding the message in the list that corresponds to it using a switch.

Once all the messages are read, both lists are returned publicly so they can be accessed to be displayed later.

CAT 10

As the own name of the class suggests, the CAT 10 class is the responsible of decoding the messages of category 10 of the ASTERIX Standard. The first thing to be found in the code, is the initialization of the different attributes and lists. Depending on the contents of the data item, public string, public double, or public integer type attributes are used to define the sets and gets.

The initial defined lists, consist of a type string list named “itemList” to store only the data items that are present in the message; the second one is a byte type list containing the different data fields.

The method readFSPEC() identifies the FSPEC to know which data items are present in the message. This method is then used in the method createDataItem(), which, as an input, has the FSPEC list returned earlier. The createDataItem() method uses the FSPEC list to initialize the data items that are present in the message, this way, we are able to know how many data items form each message and decode only the ones which are necessary.

The methods to decode each data item are extracted from the documents provided in Atenea.

CAT 21

The class for decoding messages from category 21 uses the same base methods as the class for decoding category 10 messages, the only difference between these two codes are the data items contained in the messages, which are different.

Marker

For the correct development of this class, the usage of the GMarkerGoogle inheritance was necessary. This class is based on a List of points (coordinates) that will be represented in a map. Together with the coordinates, the list includes a series of important information about the flight that will be later displayed when clicking the marker in the software.

To achieve that, public constructor type marker class is designed to filter between SMR, MLAT, and ADS-B type messages to be located with different colours in the map. In this class as well as in the simulation form, the GMap.NET library is used. In addition, also in this class, the change of coordinates takes place from cartesian, to latitude and longitude. These are obtained by the vehicle’s data in MLAT and SMR and go through this conversion to be represented correctly in the map. For that, the classes GeoUtils and GeneralMatrix are used.

Windows Forms

The software’s initial appearance is based in a fixed left side menu and a right-side empty panel. When clicking the different tabs in the menu, the different forms appear in the right-side panel.

The side menu is formed by the following tabs:

- Load file
- Data View
- Simulation

- Help

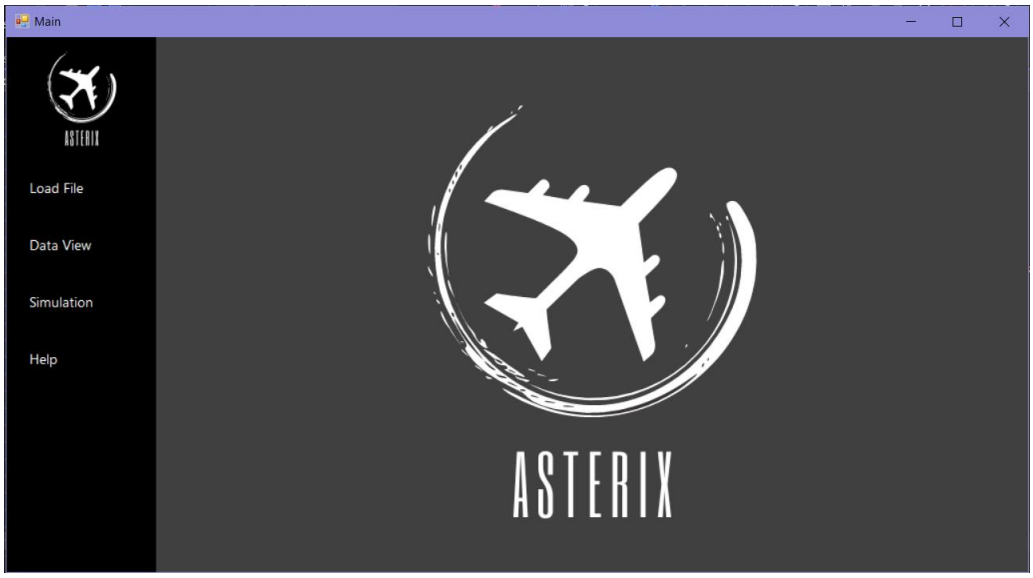

Figure #. Initial presentation of the software

Load File

As the name of the tab describes, this tab is used to load a file to the software to be decrypted. Its initial appearance shows only one button, which, when clicked, opens the File Explorer to choose a file saved in your computer. Once you choose the desired file, a new button appears which is used to start decoding the file. Once the software is done decrypting the file, the message above changes to indicate that you can go ahead and click the other tabs with the decoded information.

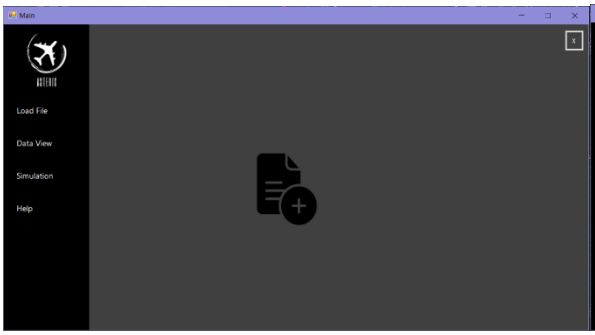

Figure #. Initial presentation of the Load File tab.

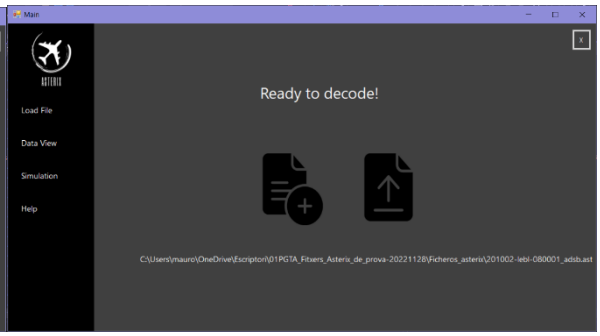

Figure #. Load File tab after selecting the desired file.

Regarding the coding to operate this specific form is based in two buttons. The first button opens the File Dialog in order to choose a file from your computer, when the button “OK” is clicked, the software takes the path of the file to be used later.

The load button appears once the desired file to be decoded is selected. This button initializes the class Fichero (using the file path provided by the File Dialog) and computes the decoding of the CAT10 and CAT21 of that specific file. Once both categories are decrypted, the data is passed to the Main form using get methods so the information does not get lost when closing the current form.

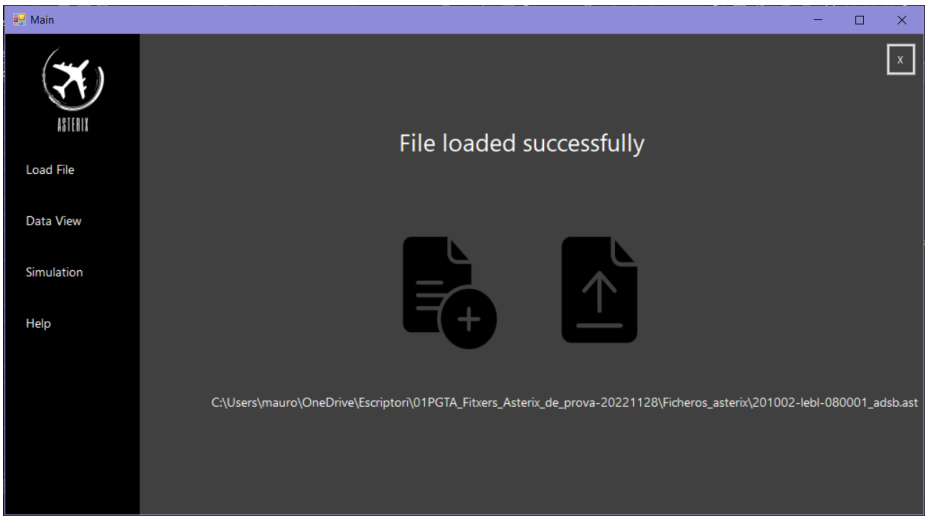

Figure #. Change of message on the top when the software has finished the decoding process.

Data View

The Data View form is used to display the data decoded from the files. To access it, you click in the Data View button located on the side menu and you choose the category you are interested in viewing, either CAT10 or CAT21, in the submenu. Once the form opens, two data grid views appear on the screen, the one in the left contains the decoded messages with the ID, the category, the time of the day, the length of the message and the number of data items it contains. To display more information about each message, the only thing needed is a simple click on the desired message and the data items which the message contains will appear in the right-side grid view. Again, to see the contents of the attributes of each data item, a simple click is enough for the software to display them in the textbox located below.

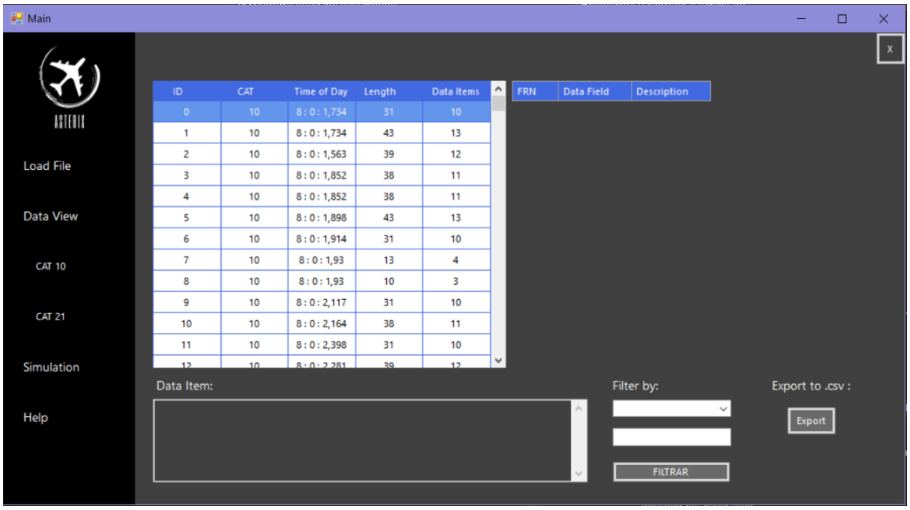

Figure #. Initial appearance of the Data View tab, in this case in the CAT10.

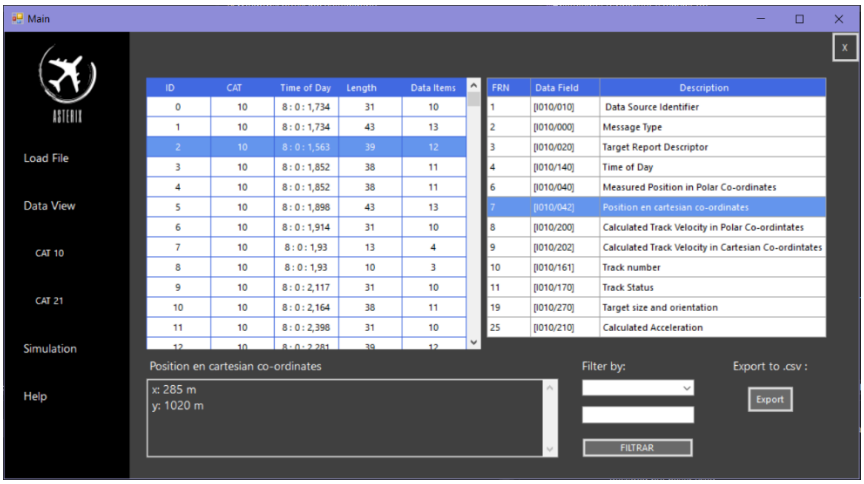

Figure #. Data View Form with the full extent of information contained in a message.

In addition, the Data View form is able to filter the information by message type, target ID and address, and track number. By clicking the button located on the right side below the data grid view, it is also possible to extract the information in the messages and turn it into a CSV file.

To develop the coding for the Data View form, a method named createDataGridView() was created to display the information in the Grid View depending on the category (achieved by using a switch), and fill it in using loops.

To show the data items after clicking the message, we used a simple code inserted in the event CellClick() which takes the information of the message and the data items which are contained in it and just show the corresponding data in the grid. The same process is done with the second data grid view to show the attributes in the text box below.

To create the filter, a switch was again used to separate between categories and another one to search data depending on the selected information you wish to be displayed. Once it is known which filter is selected, a loop is the responsible for filling the data view again with the new information.

Finally, the exportation to CSV is done by creating a list and adding all the messages and their information and then writing them in a CSV type file. When clicking the button, a Save File Dialog appears in the screen for you to write the name of the fine and select the location in which you want to save it.

Simulation

The simulation form is displayed in the screen once you click the simulation button in the side menu. Once clicked, the display shows a map that occupies more or less two thirds of the form and then some controls of the simulation below the map. In the left side of the form below the map, we have all controls related with the time control of the simulation. Above, you can find tools to set the time you want the simulation to start; and below, you can find the controls of the simulation like the speed, and the restart, pause, and skip buttons.

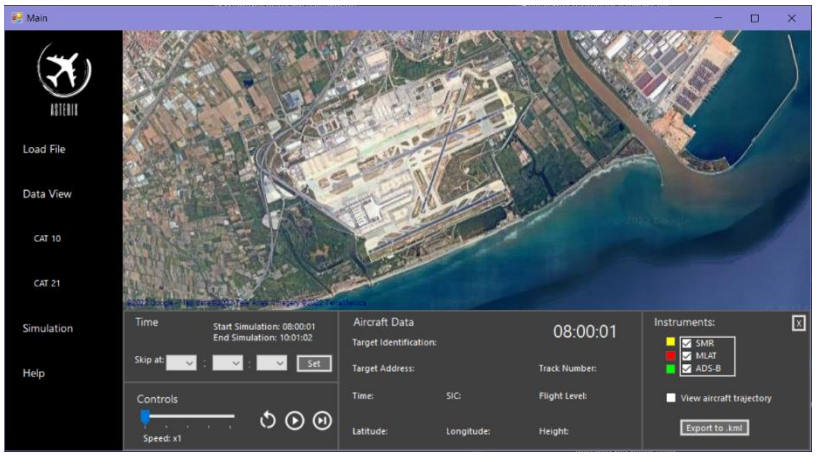

Figure #. Initial presentation of the Simulation Form.

Right in the centre of the below panel, we can find the area for Aircraft Data. At first, it displays no information. To activate it, a marker must be clicked, and then, the aircraft information will display in the corresponding attribute.

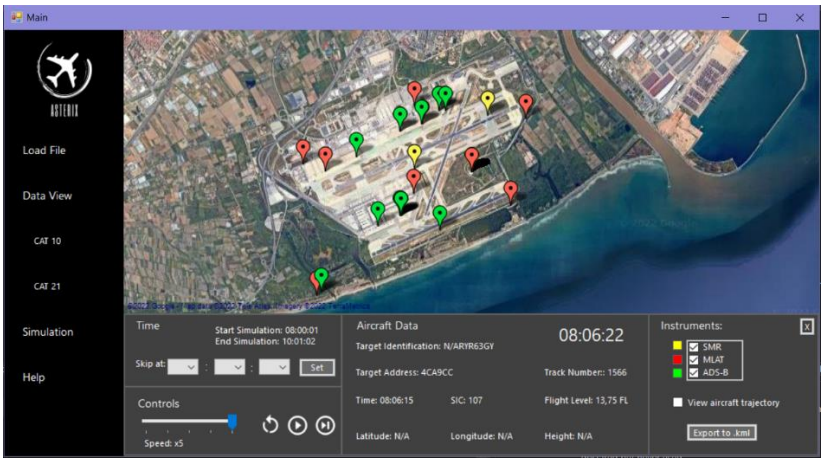

Figure #. Simulation Form with the automated simulation activated and with aircraft data from a marker.

Finally, on the left side of the below panel, we can find a map filter to choose whether to show all the signals markers or just the ones selected. Below that, we can choose to show the trajectory followed by the different vehicles showed in the map. The button below that exports the data into a KML file to be saved in the computer.

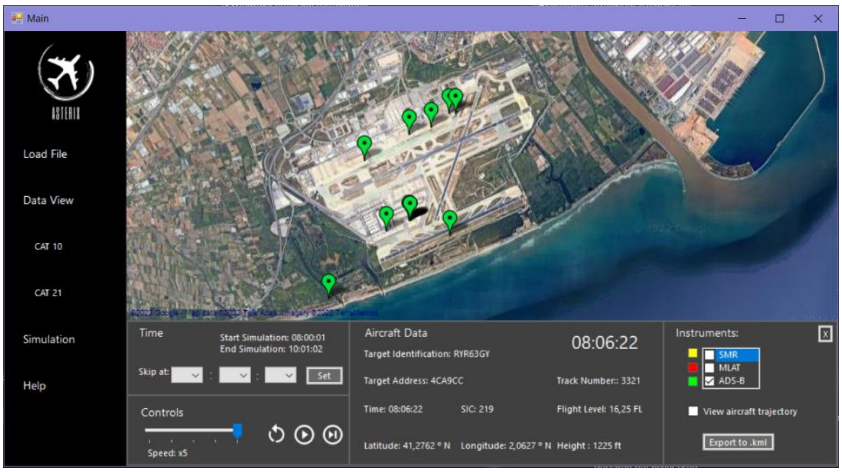

Figure #. Simulation form using the map filter only showing ADS-B markers.

To code this specific form, the library GMap.NET was used. To display the markers in the map, the list containing all the decoded messages is arranged in a chronological order to be displayed correctly. To control the simulation, you have two options: the timer, which controls the simulation automatically; or the move function, to control the slots of time you want to be displayed in the map. To achieve the move function, a slot of time is selected, and it only shows the vehicles in that instant of time and deletes the others.

The trajectories are done by a function named Create Route, which finds all the points where the aircraft has passed through. For the MLAT and ADS-B the Target Address and the SMR is done with the track number.

Finally, to export to KML, a StringBuilder was used to create the KML type file.

Help

The Help Form is a simple additional form that contains the instructions of use in case assistance is needed when using the software. It is based on a picture box and 5 buttons located below it which indicate the steps to take in order to correctly decode a file. It has a really simple code which consists in changing the picture shown depending on the clicked button.

Figure #. Help form showing the Step 1 picture.

Extras

Computation of the probability of detection

To compute the probability of MLAT detection, it has to be considered that a target has to be detected in a period of time between the first and last detection in a limited area. The probability of MLAT detection is then, the actually detected divided by the total expected detections.

Computation of the probability of identification

To compute the probability of identification, we need to know which messages have an incorrect ICAO Address. Each target gets a Track Number which is unvaried during the time the target is inside the antenna's range. It is assigned based in its relative position instead of basing it on the vehicle's identification; this allows the comparison between ICAO Addresses and track numbers to see which ones are incorrect.

Future improvements

The making of this project has helped us to correctly decode ASTERIX codes as well as improving our programming skills. However, we think that some improvements could be made to our software:

The first thing that we consider could be an improvement is the development of a Data Base which contains the data of the files (once read and decoded) and, that allows the information to flow quicker between forms.

The second thing we think we could improve is the calculation of the probabilities since the result that we have is not up to standards.

Lastly, adding more categories could mean the software is much more versatile and practical.