Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Utilització de Google Earth per al seguiment remot d'un UAV

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Marçal de las Heras Mandome

DIRECTOR: Eduard Santamaria Barnadàs

DATA: 22 d'Octubre de 2008

Títol: Utilització de Google Earth per al seguiment remot d'un UAV

Autor: Marçal de las Heras Mandome

Director: Eduard Santamaria Barnadàs

Data: 22 d'Octubre de 2008

Resum

L'objectiu d'aquest TFC és la consecució d'una plataforma que sigui capaç de monitoritzar el vol d'un UAV *(Unmanned Aerial Vehicle*) utilitzant l'eina Google Earth com a marc on representar les dades de vol de l'aparell. Aquesta monitorització ha de tenir un compromís amb la representació en temps real de les dades. Entre d'altres variables de pilotatge aeri, aquesta plataforma representa la posició de l'aparell (longitud, latitud) al mapa , la direcció i la ruta seguida fins al moment per l'UAV. Aquesta plataforma també és capaç de representar gràfiques d'altitud i velocitat dintre del mateix marc, i també en temps real.

L'eleció de Google Earth com a contenidor de les dades que representarem, està justificada per ser una eina d'informació geogràfica molt potent degut a la qualitat dels seus mapes i a les possibilitats que ofereix per dibuixar marques de posició i altres elements definits per l'usuari. Aquestes característiques el fan especialment apte per la realització del projecte. El tipus d'arxius que Google Earth gestiona s'anomenen KML *(Keyhole Markup Language)* i serveixen per representar dades geogràfiques en tres dimensions. En aquest projecte s'explota la capacitat del Google Earth per obtenir aquesta informació a través de la xarxa mitjançant el protocol http. Les aplicacions d'aquesta plataforma han d'estar instal·lades en una màquina servidor, i la monitorització s'ha de portar a terme amb màquines client connectades al servidor en xarxa local o a través d'Internet, totes les màquines client han de disposar de Google Earth i configurar-hi enllaços de xarxa per poder realitzar les peticions al servidor. Aquesta plataforma diposa també d'un directori i una base de dades que emmagatzema les missions de vol realitzades fins al moment. Les missions són guardades cadascuna en un arxiu KML diferent, on es representa tota la ruta seguida per l'avió i l'altitud en cada punt de la ruta. L' interfície d'accés a aquests arxius és de tipus web amb un entorn molt intuïtiu. D'aquesta manera podem descarregar-nos les missions anteriors per mitjà d' Internet o xarxa local.

Title: Using Google Earth for remote monitoring of UAVs

Author: Marçal de las Heras Mandome

Director: Eduard Santamaria Barnadàs

Date: October, 22nd 2008

Overview

The aim of this Project is the implementation of a platform capable of monitoring a UAV flight (*Unmanned Aerial Vehicle*). This platform uses Google Earth as the environment for representing the flight data from the UAV in real time. We represent the next flight variables: Longitude and Latitude expressing the current plane position in the Earth map, the direction of the plane and the route followed up to the moment. In addition, this platform is capable to represent Altitude and Airspeed graphs on Google Earth interface, this graphs are refreshed periodically.

Google Earth is the most powerful cartographic tool (in civil use) due to the quality of it maps and its capabilities it offers to draw placemarks and other user-defined items. These features make Google Earth specially suited to implement the project. Google Earth manages the KML (Keyhole Markup Language) file type. We can use these files to represent geographic data in three dimensions on the Earth's maps. In this project we take advantage of the ability of Google Earth for getting this information from the network using http. The programs of this platform must be installed in a Server device. The monitoring should be made from Client devices (connected to the server through a local or remote network) with Google Earth installed in each one. In addition, the correct configuration of the Networklinks is necessary in order for the Google Earth client to perfom the HTTP requests. The project also provides a database that stores past UAV missions. Each mission is saved in a different KML file. In this file the airplane route throughout the mission and the altitude at each point are represented. The access interface to these files is a web site that allows the user to download KML files from the server to any host connected to internet or to the server's local network.

ÍNDEX

1 Introducció

En aquest Projecte es desenvolupa una plataforma a través de la qual un o més usuaris , mitjançant Internet o un entorn local, puguin seguir en temps real la trajectòria i posició d'una nau aèria no tripulada mentre aquesta es troba en vol realitzant una missió. Tot aquest mostreig de dades es donarà en temps real de manera gràfica dintre d'un marc, és a dir, dins d'una aplicació de mapes reals de la Terra. Aquesta aplicació és la coneguda Google Earth de la companyia Google. En aquest capítol veurem en detall què és una nau aèria no tripulada (UAV) i perquè s'utilitzen actualment, veurem com utilitzarem l'eina Google Earth , així com les necessitats que satisfà la nostra aplicació, el seu ús i les seves funcionalitats.

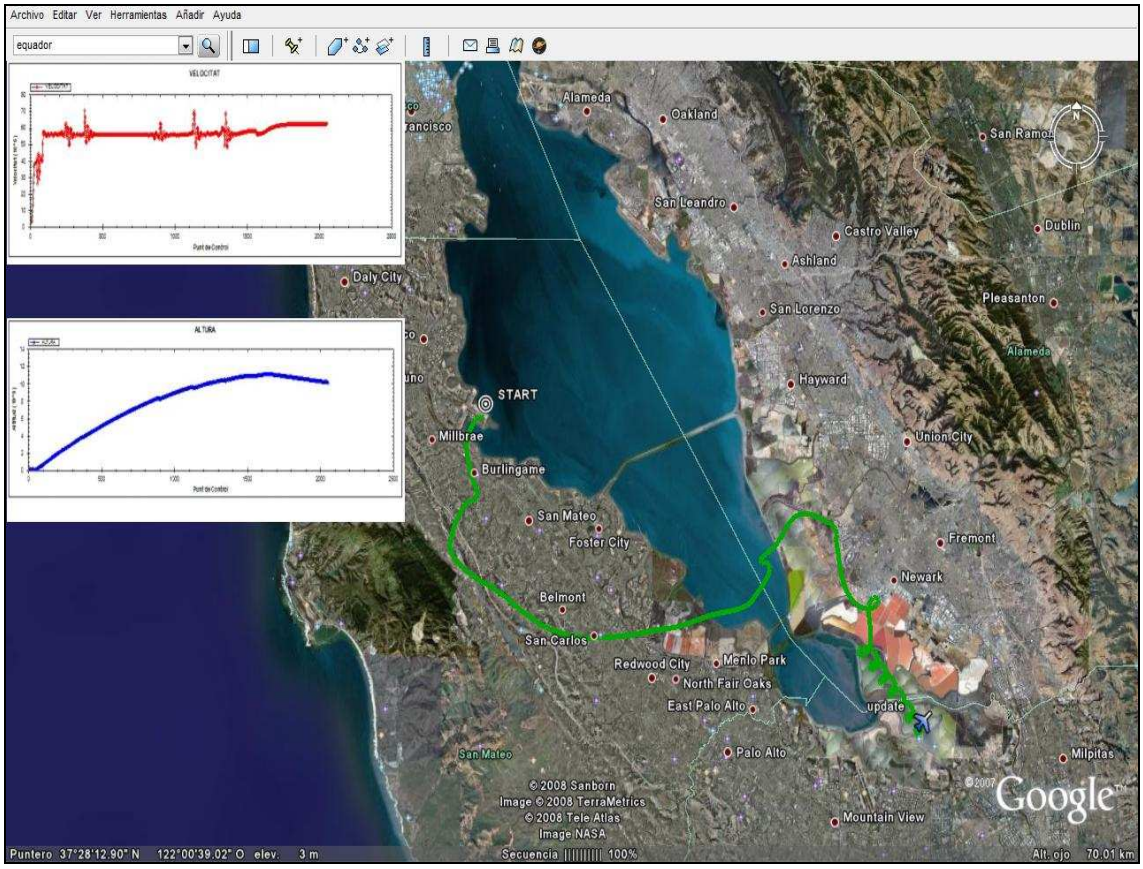


Figura 1. *Exemple de mostreig d'una missió d'un UAV concret. A la part esquerre tenim les gràfiques sobreposades a la pantalla i en verd tenim la trajectòria seguida per la nau. Finalment la icona blava del avió ens mostra la posició actual del aparell indicant la seva azimut o orientació respecte del nord.*

1.1 Les naus aèries no tripulades

1.1.1 Característiques

Un vehicle aeri no tripulat (*les seves sigles en anglès són **UAV** Unmanned Aerial Vehicle)* és un vehicle aeri capaç de volar sense la necessitat de pilot humà, gràcies a un sistema de pilotatge autònom. Els UAV posseeixen mecanismes que combinen informació procedent de sistemes de posicionament global GPS, sistemes SIG (Sistema d'Informació Geogràfica) i servomecanismes que accionen els diferents sistemes de navegació aèria que té. La unitat central de control que porta a bord s'encarrega de pilotar l'aparell combinant aquests sistemes sense la necessitat de que un humà s'encarregui presencialment.

Un Sistema d'Informació Geogràfica (SIG) [10] És un sistema que modela la realitat terrestre, empra un sistema de coordenades i s'utilitza per satisfer unes necessitats concretes d'informació referides a la Terra. Google Earth és un sistema d'informació geogràfica.

Un sistema SIG pot gestionar, entre d'altres, els següents aspectes:

1. Localització actual
2. Càlcul de rutes òptimes entre dos o més punts
3. Pautes: Detecció de pautes espacials
4. Models: Generació de models a partir de fenòmens o actuacions simulades.

Els EEUU encapçalen la llista de països en el desenvolupament d'aplicacions UAV [2] [11]. També és la nació que més dispositius en disposa [11]. Avui dia, l'ús de UAV es centra en missions de reconeixement i vigilància. Els EEUU exploten aquest recurs principalment en la vessant militar des de la Guerra del Golf al 1992, els coneguts com "avions espies". Cal aclarir que aquest projecte es centra en una vessant totalment diferent del àmbit militar i no contempla cap ús d'aquest tipus. Com en molts altres avenços en el camp de la ciència, els conflictes armats han jugat un paper protagonista en el desenvolupament i implementació de projectes que haguessin trigat molts més anys en ser traduïts a mecanismes palpables. Les primeres potències mundials van apostar en els inicis del desenvolupament de les UAV per obtenir resultats en el camp militar, en aquest cas són els EEUU els primers que van poder treure'n partit. Cal recordar però, que la transcendència de les UAV en la població civil pot representar molt elevada en aspectes més positius i de més importància, ja que gràcies a aquestes naus es podrien combatre més eficientment catàstrofes naturals com incendis i inundacions. És en aquest punt on aquest projecte intenta ser útil.

1.1.2 Classificació dels UAV

Podríem fer una classificació de les naus aèries no tripulades segons l'aplicació que tindran un cop implementats:

- Reconeixement – transmet informació per a ús militar
- Investigació i desenvolupament – Proves de mecanismes en procés d'implementació
- UAV comercials i civils – Dissenyats per a propòsits civils i mediambientals, el nostre projecte es centra en aquesta utilitat.

1.1.3 Aplicacions

Tal i com hem vist en apartats anteriors, l'ús principal d'aquestes naus és l'àmbit militar. L'aparició d'aquests mecanismes pertany a l'història molt recent i és un camp que encara està en estudi i desenvolupament. Cal destacar que aquest projecte pertany a una part què ,en principi, és transparent a l'ús del aparell, ja que es centra en la part de monitorització de les missions; però si més no la perspectiva de l'implementació i la memòria estan orientades als usos civils que poden realitzar els UAV. Aquesta ha estat la filosofia en el desenvolupament.

L'equipament necessari d'una nau ve definit segons el propòsit per al qual es projecten. Poden ser equipats amb càmeres de vídeo i càmeres d'infrarojos, també amb sensors per a la detecció de molts actius. En el nostre àmbit es podrien instal·lar càmeres amb capacitat de percepció de calor per a la detecció de "punts calents" en incendis (es podria detectar en quin punt un incendi pot revifar), també sensors climatològics com per exemple sensors temperatura, pressió i sobre factors contaminants (com CO_2). Seria molt interessant rebre aquestes variables i poder dibuixar una indicació al Google Earth (en forma d'icona en la coordenada exacta de detecció) si l'avió detectés algun valor anormal dintre d'uns llindars. També seria molt interessant anar dibuixant gràfiques (també a Google Earth) amb els valors rebuts; aquesta funcionalitat està implementada en aquest projecte per a les variables d'altitud i velocitat del aparell, però es podrien afegir més mòduls gràfics per a altres variables. El desenvolupament dels UAV està revolucionant el món de la aeronàutica, l'objectiu sempre ha estat poder disposar d'aeronaus que compleixin un llarg abast i la seva pèrdua no impliqui un cost humà. Altres avantatges d'aquests aparells són la possibilitat de operació amb control remot, la possibilitat d'operació amb programació de ruta, el muntatge ràpid pre-enlairament, la transmissió en temps real d'informació i el seu baix cost de construcció, amb comparació amb altres aeronaus. Es poden aplicar en ambients d'alta toxicitat química i radiològica per als humans, on és necessària una intervenció per extraure mostres o per fer un estudi de danys i actuacions de salvament en cas de catàstrofe. Altres aplicacions mediambientals dels UAVs podrien ser el control de la pesca i la navegació marítima. Altres aplicacions civils podrien ser la vigilància de fronteres i la elaboració de mapes topogràfics , amb una logística molt més senzilla i menys costosa que l' utilitzada fins ara amb avions tripulats.

1.2 Simulació de vols de UAVs per a la programació de les aplicacions del Projecte

Alhora de desenvolupar les aplicacions del projecte ens cal un software que ens generi un flux d'informació sobre posicionament geogràfic (coordenades) a la Terra. Per la simulació del vol d'una nau aèria no tripulada hem fet servir una aplicació gratuïta de simulació de vol. Més concretament el *FlightGear Flight Simulator V1.0.0.*

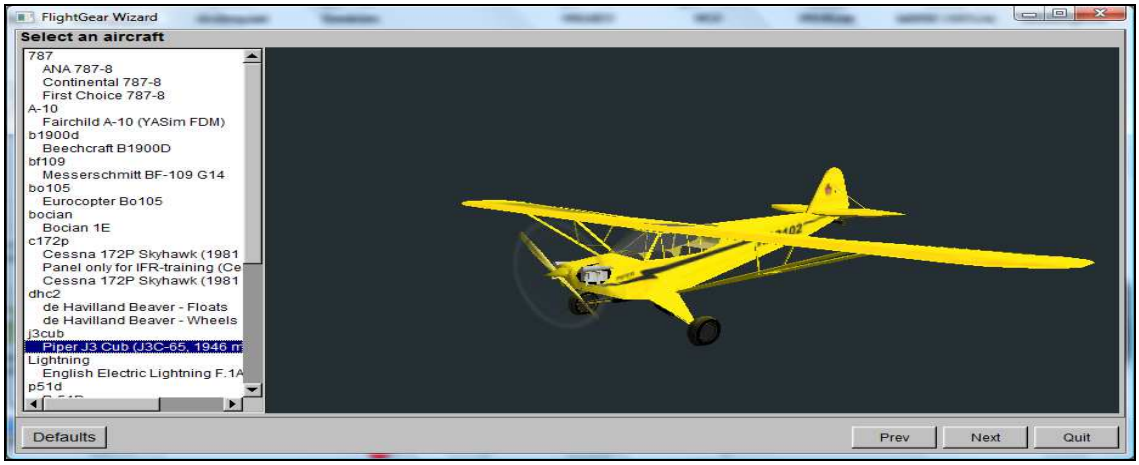


Figura 2. *Pantalla de presentació del FlightGear, a l'esquerra tenim un menú en el qual sel·leccionarem la nau més idònia per la nostra missió.*

Aquesta eina ens ha permès obtenir dades que podrien ser perfectament reals de missions de vol d'aeronaus. La comunicació entre el simulador del vol i l'aplicació principal dissenyada pel projecte, es fa mitjançant una comunicació en xarxa entre programes (*Socket*) en un port lògic determinat i modificable.

Figures 3 i 4. *Havilland Beaver realitzant un aterratge (esquerre). Concorde estàtic a la pista d'aterratge (dreta)*

1.3 L' ús de Google Earth

Al llarg dels últims anys s'han creat diverses aplicacions de mapes reals de la Terra. Normalment són programes amb poca definició i detall, amb aquesta sentència em refereixo a programes a l'abast d'usuaris civils. En entorns militars i aviació comercial s'han desenvolupat eines molt potents. La meva opinió és que l'aparició de Google Earth va ser un punt d'inflexió en aquest tipus d'aplicacions. Aquesta eina permet tenir una visió amb imatges per satèl·lit de la Terra, és a dir, amb una qualitat fotogràfica elevada i un detall molt bó en la gran part dels indrets. A part de la qualitat del seus mapes , la navegació entre aquests es fa el més senzilla possible. Depenent de la distància a la que volem veure la Terra (més lluny o més a prop) les imatges (mapes) van canviant, d'aquesta manera aconseguim un efecte de navegació real del globus terraqui.

Apart de totes les avantatges en quant a eina cartogràfica que presenta Google Earth, aquest projecte es desenvolupa amb l'ajuda d'aquest degut a que ens permet interactuar-ne mitjançant els seus arxius propis, els arxius KML. Aquests arxius són una variant de l'estàndard XML. Altres funcionalitats que presenta i que són imprescindibles per a la realització d'aquest projecte són els enllaços de xarxa per a els arxius KML (*NetworkLinks*) i enllaços per a imatges (*ScreenOverlays*). A continuació es detallaran a grans trets aquets aspectes i es farà un aprofundiment més exhaustiu en el cos d'aquesta memòria.

1.3.1 Arxius KML

És una variant de l'estàndard XML. En concret són arxius XML amb una sèrie d'etiquetes concretes que Google Earth entén. Google Earth utilitza aquestes etiquetes per dibuixar marques de posició a sobre dels mapes, així com rutes amb molts punts marcats, polígons geomètrics, entre d'altres. L'estàndard KML aglutina una jerarquia d'etiquetes que fan referència a aspectes del món de la cartografia i de la navegació terrestre; també aspectes que no tenen res a veure però que es fan servir per crear funcionalitats que si que fan referència als mapes de la Terra.

1.3.2 NetworkLinks

Tal i com el seu nom indica és un enllaç de xarxa. Fa una connexió lògica per obtenir un arxiu KML (dintre de la mateixa màquina, en xarxa o a través d'Internet). S'actualitzen un cop cada cert temps configurable.

1.3.3 ScreenOverlays

Té un funcionament molt similar al *NetworkLink*, però en comptes de demanar arxius KML treballa amb imatges. Un cop cada període de temps es descarrega una imatge des d'una adreça (en aquest cas de xarxa) determinada.

1.4 Introducció a la plataforma creada

Les funcionalitats que proporciona la plataforma amb l'ajuda de Google Earth les podríem agrupar en tres blocs.

En primer lloc tenim el bloc que representen les funcionalitats principals d'aquest projecte:

- Mostreig en temps real de la posició actual de l'aparell.
- Mostreig de la ruta que ha seguit la l'avió fins al moment.
- Visualització de la direcció que té l'avió en cada moment (*Heading*).
- Gràfica de l'Altura de la nau durant tota la ruta.
- Gràfica de la Velocitat de la nau durant tota la ruta.

En segon lloc tenim les funcionalitats referents a la visualització del contingut anterior dintre de Google Earth:

- Vista en la perspectiva del pilot de l'avió en temps real (càmera en primera persona).
- Vista del seguiment de la ruta de l'avió en temps real (càmera seguiment).

En tercer i últim lloc, tenim les funcionalitats que complementen la plataforma però que no intervenen en la monitorització de les missions:

- Generació a posteriori d'un arxiu KML amb tota la ruta seguida per la nau indicant l'altura en cada moment.
- *Site* web amb l'històric de totes les missions on es poden descarregar els arxius KML del punt anterior.

Per aconseguir les funcionalitats descrites he emprat diferents mòduls o aplicacions. Podríem diferenciar-les en 5 blocs. En aquest apartat d'introducció i contextualització veurem els diferents blocs però entrarem en el detall del seu funcionament més endavant.

1.4.1 Aplicació 1: Programa principal

Aquest programa s'encarrega de comunicar-se amb el simulador de vol i obtenir les dades que aquest ens proporciona per després construir les gràfiques d'altura i velocitat de la nau i anar omplint un arxiu de text amb les coordenades del avió. Aquest arxiu de text serà imprescindible pels aplicatius següents.

1.4.2 Aplicació 2: Mostreig de la posició en temps real i ruta seguida fins al moment

Aplicació web que ens permet obtenir la ruta que ha seguit l'avió fins al moment i la posició actual que presenta l'avió en el moment de la consulta. Aquest aplicatiu va dibuixant la ruta (per mitjà d'arxius KML) que segueix l'avió al Google Earth llegint les dades proporcionades per l'arxiu de text que es va actualitzant gràcies al programa principal.

1.4.3 Aplicació 3: Càmera 1 (Seguiment de la trajectòria)

Aplicació web que cerca , en tot moment, la situació de l'avió en pantalla (del Google Earth) i ens reprodueix una vista de la trajectòria de l'aparell basada en l'altitud real que presenta en aquell moment. També té en compte l'inclinació de la cabina alhora de mostrar-nos els mapes de la Terra.

1.4.4 Aplicació 4: Càmera 2 (Primera persona)

Aplicació web que ens permet obtenir la vista d'una possible càmera interior. En comptes de generar una ruta visual, actualitza la vista en primera persona amb les coordenades que presenta en aquell instant.

1.4.5 Aplicació 5: Web de missions

Un cop s'acaba la monitorització d'una missió, es genera un arxiu KML on s'indica la ruta seguida per l'aparell i l'altura d'aquest en tot moment. Aquesta aplicació web ens permet la descàrrega d'aquests arxius KML via Internet o xarxa local.

1.5 Entorn de treball

Aquesta plataforma de monitorització està dissenyada per que sigui accessible per sistemes informàtics des de qualsevol ubicació de xarxa possible. Aquest projecte pretén ser transparent en aquest sentit i oferir les seves funcionalitats independentment de la tecnologia IP que s'utilitzi per arribar al servidor. L'utilització d'HTTP com a protocol a nivell d'aplicació és un benefici ja que el seu ús és molt popular i totes les distribucions de qualsevol fabricant de sistemes operatius el tenen implementat en les seves màquines i dispositius.

Podríem distingir dos clars entorns de treball per la nostra aplicació. En primer lloc, aquesta plataforma es podria utilitzar en un entorn local, és a dir, dins d'una xarxa local, *Ethernet* per exemple.

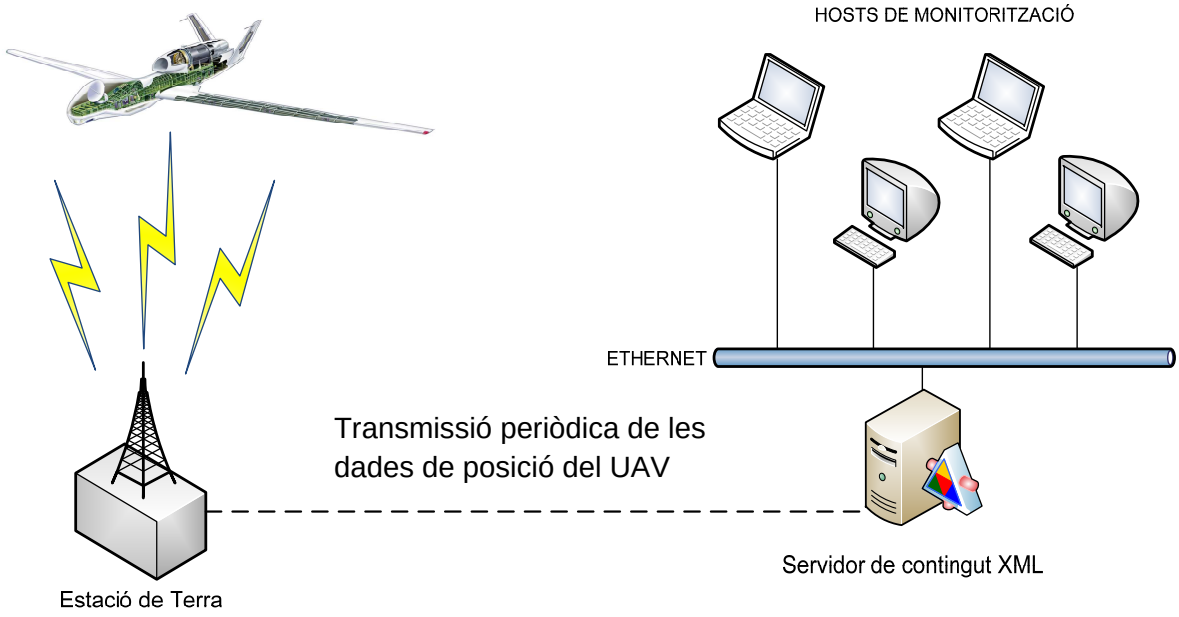


Figura 5. *Implementació en Xarxa Local*

En segon lloc, aquesta plataforma també es podria utilitzar en un entorn remot, a través d'Internet.

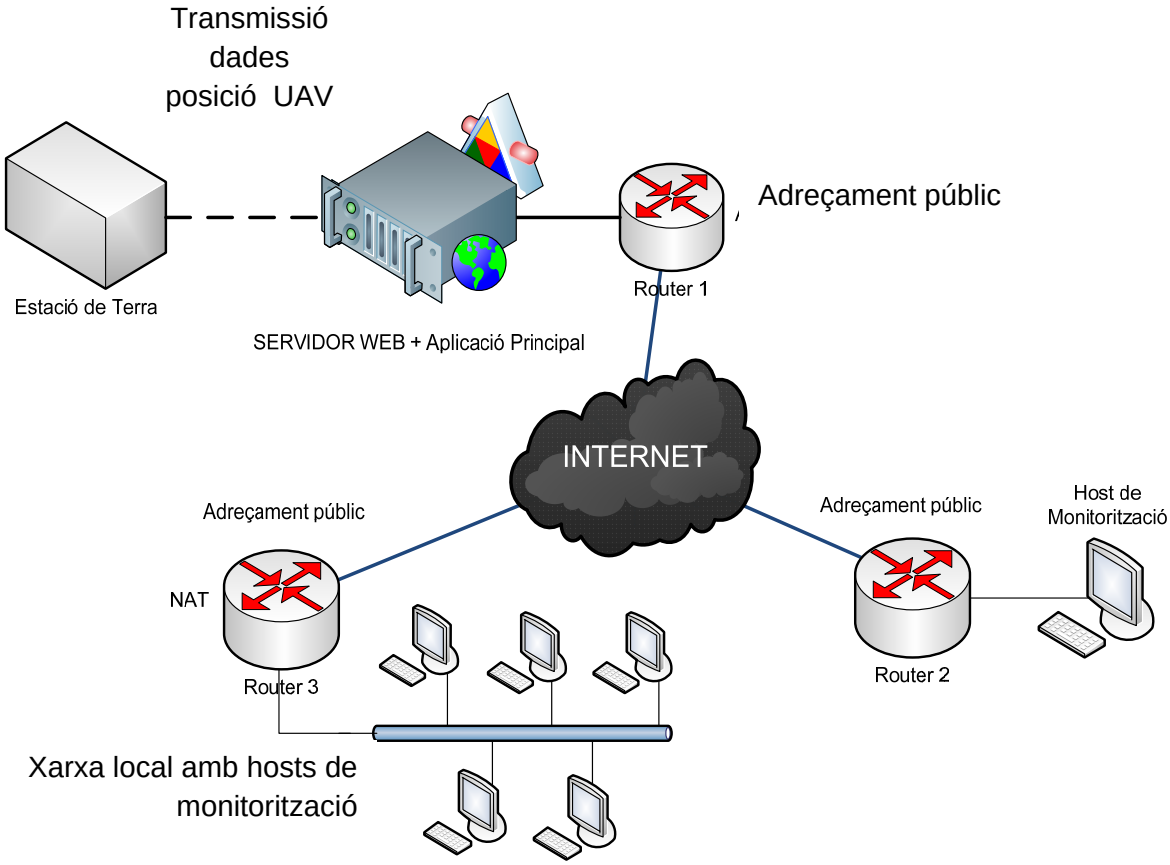


Figura 6. *Implementació a través d'Internet*

2 Google Earth i els arxius KML

Tal i com s'ha explicat en la part d'introducció de la memòria, aquest projecte implementa una monitorització en temps real del vol d'una nau aèria mitjançant Google Earth. La raó clau en l'elecció d'aquesta eina és molt senzilla: és una aplicació que ens permet treballar a sobre dels seus mapes, és a dir, no només es un software cartogràfic , si no que a part d'això podem dibuixar rutes i polígons a sobre dels mapes i interactuar amb l'aplicació per aconseguir que aquestes rutes, punts , imatges i polígons (entre d'altres) puguin mostrar-se en el moment que nosaltres decidim. Google Earth es caracteritza per ser un Sistema informàtic d'informació geogràfica (SIG) que ens permet visualitzar imatges en tres dimensions del planeta. Combina imatges per satèl·lit, mapes cartogràfics i el motor de cerca propi de Google. Alhora de fer una navegació virtual de la Terra, el nostre ordinador es comunica constantment amb una potent base de dades a Internet i es descarrega (per mitjà de *streaming*) el contingut necessari per cobrir la zona en la qual ens situem de la Terra. Google Earth ens ofereix característiques en tres dimensions com per exemple la visualització de massissos i valls. Treballa amb un tipus d'arxius basats en XML que ens permeten representar dades geogràfiques en 3D. La qualitat dels seus mapes i imatges són excel·lents i gràcies a això podem aconseguir un detall molt gran, amb distàncies d'apropament molt petites.
Els arxius KML (*Keyhole Markup Language*) ens proporcionen la capacitat de dibuixar o afegir dades geogràfiques dins del relleu de la Terra, els exemples més clars són polígons en 3D , rutes en 2D i 3D, sobre-posicionament d'imatges i sobretot marques de posició. Un arxiu KML és un arxiu que segueix l'estàndard XML però amb una jerarquia d'etiquetes establerta que Google Earth és capaç d'interpretar. Un exemple molt senzill d'arxiu KML seria el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
<name> Marca de posició </name>
<Placemark>
<name> Punt "A" </name>
<coordinates> -95.44543716852063,40.42556202233022 </coordinates>
</Placemark>
</Document>
</kml>
```

Codi 1. *Exemple de codi d'un arxiu KML senzill que representa una marca de posició en el mapa. Aquest arxiu dibuixa una marca de posició amb una icona per defecte en les coordenades indicades.*

L'objectiu principal d'aquest projecte és aconseguir veure la trajectòria en temps real d'un avió a sobre dels mapes de Google Earth. Podríem dir que gran part d'aquest objectiu és viable gràcies a la capacitat que té Google Earth de poder dibuixar un punt en l'espai i després poder actualitzar-lo o modificar-lo. Per poder realitzar aquest efecte de trajectòria visual ens cal un flux automàtic de punts (coordenades) que van modificant i actualitzant un punt inicial, que podria ser la pista d'aterratge de l'aeroport on l'avió emprèn el vol.

El flux de punts continu l'aconseguim amb el simulador *FlightGear* , el qual ens entrega un cop per segon (configurable) la posició del avió . Situant aquest projecte fóra d'un entorn de desenvolupament o laboratori i traslladant-lo a un entorn útil posterior, aconseguiríem aquest flux mitjançant un receptor situat en una estació de Terra que seria l'encarregat de rebre els valors de posició de la nau mitjançant un enllaç ràdio, per exemple (recordem la Figura 6 d'aquesta memòria). Aquest projecte s'encarregaria de visualitzar les dades rebudes en temps real i fer un històric amb les missions realitzades.

2.1 Utilitats de Google Earth emprades

2.1.1 NetworkLinks

Els enllaços de xarxa són una eina de Google Earth que ens permeten fer una referència a arxius KML situats en algun node d'una xarxa local o remota. Ens permeten carregar aquest arxius remotament. Per definir-los utilitzem les següents propietats:

<Link> : mitjançant aquesta etiqueta enllacem l'arxiu remot

<refreshVisibility> : variable booleana que defineix si l'element és actualitzable o només es carrega un cop.

<refreshmode> : per definir el mode d'actualització, en aquest cas serà periòdicament.

<flyToView> : variable booleana que indica si la vista de Google Earth s'ha de desplaçar cap a la coordenada de l'arxiu, un cop carregat.

2.1.2 Actualitzacions o Updates

Per poder actualitzar les dades carregades sobre un enllaç de xarxa, utilitzem l'element *Update*, que és un descendent de l'element *NetworkLinkControl* (element que permet gestionar les dades d'un enllaç de xarxa). L'actualització pot ser incremental, decremental o amb modificació [8]. Per poder establir una sessió d'esdeveniments on es carreguen dades geogràfiques que seran actualitzades periòdicament, s'ha de seguir el següent procediment:

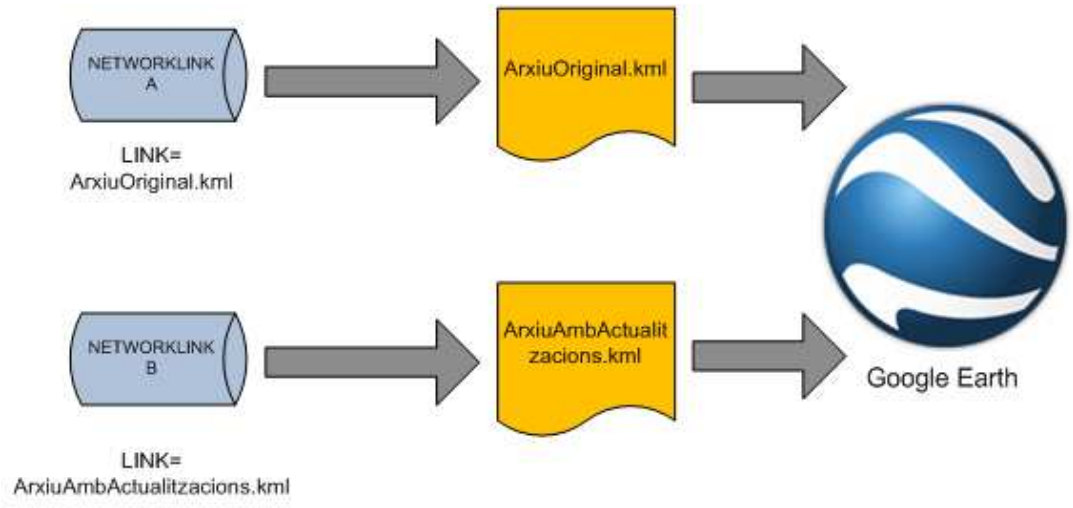


Figura 7. *Procés d'implementació d'actualitzacions de Google Earth.*

1. Un *NetworkLink* carrega l'arxiu KML original a Google Earth. L'element que dintre d'aquest arxiu serà modificat posteriorment, necessita un identificador únic dintre del arxiu quan és definit.
2. Un altre *NetworkLink* carrega un segon arxiu KML que conté les actualitzacions (qualsevol combinació de **<Change>** per modificació de dades, **<Create>** per afegir dades o punts geogràfics i **<Delete>** per esborrar informació) a l'objecte a modificar. Aquest segon arxiu KML conté dos referències per identificar les dades originals:
3. Per localitzar les dades dins Google Earth, l'element **<Update>** utilitza *targetHref* per identificar l'arxiu original. Per identificar l'objecte situat dintre del arxiu original que volem actualitzar, els elements **<Change>**, **<Create>** o **<Delete>** contenen l'atribut *TargetId*.

Exemple d'arxiu original:

```
<kml xmlns="http://earth.google.com/kml/2.2">
    <Document id="1">
    <name>PrimerPunt.kml</name>
    <Placemark id="primerpunt">
                        <Point>
                        <coordinates>-122.357246,37.613544,0</coordinates>
                        </Point>
   </Placemark>
   </Document>
</kml>
```

Codi 2. *Exemple de codi d'un arxiu KML on es representen les dades originals d'una marca de posició que serà modificat.*

Exemple d'arxiu d'actualització:

```
<kml xmlns="http://earth.google.com/kml/2.2">
<NetworkLinkControl>
<Update>
        <targetHref>http://localhost/Primerpunt.kml</targetHref>
        <Create>
                <Document targetId="1">
                <Placemark Id="segonpunt">
                <LineString>
                <coordinates>-122.372,37.567,0 -122.34789,37.566,0</coordinates>
                </LineString>
                </Placemark>
                </Document>
        </Create>

        <Change>
                <Placemark targetId="primerpunt">
                <Point>
                <coordinates>-122.45344,37.78876,0</coordinates>
                </Point>
                </Placemark>
        </Change>
</Update>
</NetworkLinkControl>
</kml>
```

Codi 3. *Exemple de codi d'un arxiu KML on es representa la modificació de les dades pròpies del exemple Codi 2.*

2.1.3 Sobreexposició d'imatges auto-actualitzables

Per complementar l'objectiu anterior volíem disposar de gràfiques de velocitat i altitud de l'avió alhora que nosaltres estem veient la seva posició desplaçant-se en ple vol.
Google Earth accepta , gràcies al seu disseny i la seva vocació, sobreexposició d'imatges (de diferents formats) sobre dels mapes reals de la Terra, també accepta sobreexposició de models d'edificis o monuments (per completar el ventall de opcions que podem trobar navegant pels mapes). Nosaltres, per contra, no volem situar una imatge en qualsevol coordenada terrestre, el què volem exactament és poder situar una imatge en una posició fixa de pantalla i que es pugui actualitzar. Al manual de consulta per a desenvolupadors de KML vàrem descobrir la propietat *ScreenOverlay*. Gràcies a aquesta propietat de la jerarquia KML podem fer una connexió contra un arxiu de mapa de bits , amb formats *jpg* o *gif* entre d'altres. Aquests arxius romandran fixes a pantalla i els-hi podrem variar l'opacitat per aconseguir un efecte de semi-transparència, alhora que s'actualitzen periòdicament. Nosaltres disposem d'una imatge que representa la gràfica de l'altura i una altra de velocitat en una carpeta a la qual accedim a través del nostre servidor.

Google Earth per mitjà de *ScreenOverlay* es descarrega periòdicament (cada 3 segons) l'imatge servida pel nostre servidor web; és a dir , Google Earth no s'adona si l'imatge es la mateixa o ha canviat, simplement la descarrega cada cert temps i mostra el contingut. Cal dir però, que l'actualització de les imatges es fa transparentment a aquest, ja que les imatges canvien contínuament gràcies a que el programa principal les sobreescriu amb noves dades afegides a l'eix de coordenades. La part d'obtenció de les gràfiques serà explicada en el bloc de funcionament modular, en el seu corresponent apartat. A continuació trobarem un breu resum del codi KML emprat :

```
<ScreenOverlay>
        <name>Gràfic ALTURA</name>
        <Icon>
        <href>http://localhost/GraficaAltura.jpg</href>
        <refreshMode>onInterval</refreshMode>
        <refreshInterval>3</refreshInterval>
        </Icon>
        <overlayXY x="0" y="1" xunits="insetpixels" yunits="insetpixels"/>
        <screenXY x="0" y="318" xunits="insetpixels" yunits="insetpixels"/>
        <size x="450" y="190" xunits="pixels" yunits="pixels"/>
</ScreenOverlay>
```

Codi 4. *Codi emprat per l' implementació de la sobreexposició d'imatges en un client Google Earth.*

Del codi KML anterior, destacarem:

Href: Enllaç a l'imatge que ens descarreguem periòdicament i que serà mostrada
RefreshMode: Representa el mode d'actualització, en aquest cas periòdicament
RefreshInterval: Representa la periodicitat, és un complement de la propietat anterior

Podem veure que per a l'enllaç utilitzem una adreça web, la qual ens serveix una imatge que està allotjada en el servidor. Per poder carregar correctament les gràfiques alhora de monitoritzar una missió d'un UAV, caldria executar un arxiu KML amb el Google Earth amb un codi molt similar al **Codi 4**. Només hauríem de tenir en compte on està l'imatge que volem enllaçar, o quina adreça ens la pot proporcionar. En les últimes línies de **Codi 4** veiem les etiquetes *overlayXY* i *screenXY*, que ens són útils per definir la posició de les imatges en la pantalla (amb píxels o posicions absolutes) i també la grandària de la imatge respectivament. Podem tenir una imatge més o menys gran i reduir-la o ampliar-la tenint en compte la pèrdua de qualitat que això comporta.

2.1.4 Actualització de vista de la càmera

Defineix la càmera virtual que veu l'escena en aquell moment. Aquest element defineix la posició de la càmera en relació a la superfície de la Terra. Definim aquesta càmera amb dos conceptes, el de posició de càmera i el de direcció de vista. La posició la definim amb:

Longitud i **latitud** que formarien la coordenada on ens hem de situar i l'altitud d'elevació des d'aquest punt

La direcció de la vista la definim amb:

Heading, que serà la direcció Azimut (direcció respecte del Nord) a la qual ens situarem des del punt anterior. El **tilt** o inclinació abatible , i finalment **roll** o inclinació lateral.

Igual que les actualitzacions (*Updates*) els objectes de càmera poden ser descendents d'un element *NetworkLinkControl*. Per tant podem anar proporcionant al Google Earth diferents vistes de càmera en cada interval de temps configurable en el *NetworkLink*.
A diferència dels *Updates*, els arxius de càmera no cal que tinguin un enllaç apuntant a un arxiu original i un segon apuntant a arxius d'actualització. Si no què amb un sol enllaç de xarxa actualitzable, Google Earth és capaç de moure la càmera de posició i direcció si les coordenades del nou punt així ho requereixen.

```
<kml>
	<NetworkLinkControl>
		<Camera>
			<longitude>-122.320898</longitude>
			<latitude>37.636674</latitude>
			<altitude>3895.275288</altitude>
			<heading>318.229660</heading>
			<tilt>88,66334</tilt>
			<roll>51.446306</roll>
			<altitudeMode>clampToGround</altitudeMode>
		</Camera>
	</NetworkLinkControl>
</kml>
```

***Codi 5** : Exemple de la construcció d'un arxiu de Càmera KML*

3 Comparativa de servidors web i tecnologies a utilitzar

3.1 Comparativa de servidors

Per tal d'alimentar el Google Earth amb arxius KML, necessitem un software servidor web que ens permeti enviar aquests arxius tant en un entorn de xarxa local com en un entorn remot a través d'Internet. Distingim dos necessitats diferenciades: una per a ús en laboratori i una altra per a ús en explotació real servint dades via Internet a usuaris ubicats a qualsevol lloc del Món.
Aquesta plataforma no respon a les peticions del Google Earth amb una pàgina web HTML amb imatges i etiquetes pròpies d'aquest llenguatge (aquest seria l'intercanvi més comú), aquesta plataforma executa un codi darrera de cada petició per donar una resposta actualitzada de la posició de l'aparell en un moment determinat.
El nostre servidor web es mantindrà a l'espera de peticions HTTP per part del client, que serà l'aplicació Google Earth de l'usuari o usuaris que vulguin fer la monitorització de la missió UAV. El nostre servidor web respondrà amb una pàgina web basada en XML que Google Earth interpretarà. El funcionament és molt semblant a la descàrrega d'una pàgina web comú: El client , un cop rebut el codi mitjançant una petició, és l'encarregat d'interpretar el codi HTML que rep, és a dir, mostrar les fonts, la disposició del text i els objectes de la pantalla. A priori ,el servidor es limita a transferir el codi de la pàgina, un cop elaborat, sense interpretar-la. Sobre un servei web clàssic, també podem disposar d'aplicacions web. Aquestes són porcions de codi que s'executen quan es realitzen certes peticions HTTP. Podem trobar dos classes:

- Aplicacions en el costat del client: El client web és l'encarregat d'executar el codi en qüestió a la màquina del usuari. Són aplicacions del tipus Java o Javascript: el servidor proporciona el codi de les aplicacions al client i aquest, mitjançant el navegador, les executa. En aquests casos és necessari, per tant, que el client disposi d'un navegador amb capacitat d'executar aquestes aplicacions o *scripts*.

- Aplicacions del costat del servidor: El servidor web executa l'aplicació, aquesta, un cop executada, genera un cert codi HTML (o XML); el servidor pren aquest codi i l'envia al navegador client per mitjà del protocol HTTP

Les aplicacions al costat del servidor moltes vegades resulten la millor opció per a realitzar aplicacions web. Un avantatge significatiu és que al executar-se el codi en la màquina servidor i no en la màquina del client, aquest no necessita tenir una capacitat extra , a diferència de l'aplicació de Java o Javascript. D'aquesta manera qualsevol navegador web bàsic pot ser usuari d'aquest tipus d'aplicacions. D'altra banda també presenten inconvenients, en el nostre cas però són l'única opció ja que Google Earth no té la capacitat d'execució de codi.

En referència a l'elecció del servidor web , a part de contemplar els servidors web més comuns, ens vam plantejar l'idea d'utilitzar els anomenats servidors web lleugers.

Aquest tipus de servidors es caracteritzen per ser servidors menys potents i flexibles que els servidors web més comuns com Apache o IIS, si més no, són servidors molt més ràpids en quant a velocitat de procés i proveïment de pàgines web. La seva principal utilitat, encara que no l'única, es basa en servir webs estàtics o continguts estàtics com imatges per a les que no cal tota la flexibilitat que ens poden aportar servidors web més grans. Un exemple molt clar són *Sites* amb multitud de visites en els quals és molt important aprofitar al màxim els recursos del sistema.
Aquests servidors han estat dissenyats per córrer a sobre de màquines amb pocs recursos degut al hardware disponible o a l'entorn en què es troben. En els últims anys ha hagut una proliferació d'interessants postes en marxa de servidors web com per exemple *Lighttpd* o *Litespeed*, entre d'altres. Aquests servidors gaudeixen de combinacions diferents de funcionament, velocitat, flexibilitat, facilitat d'administració, transportabilitat i seguretat entre d'altres valors afegits.
Apache i IIS no poden optimitzar tants factors simultàniament, els servidors lleugers poden sobrepassar en prestacions als anteriors en més d'una característica. La conclusió que es pot extraure de tot aquesta informació és que podíem trobar un servidor alternatiu que fóra adequat a les necessitats d'aquest projecte. Vam fer un estudi amb tot un conjunt de servidors a tenir en compte gràcies a la seva referència. Dintre de les característiques imprescindibles que volíem trobar en el candidats, vam considerar com vinculants la flexibilitat en quant a l'execució de pàgines dinàmiques i la capacitat de gestió intuïtiva.

A continuació podrem veure la taula comparativa realitzada per a l'elecció del servidor web. Els valors que hem tingut en compte son els següents:

- Open Source: servidor gratuït o no.
- Suport del protocols segur d'HTTP (HTTPS).
- Suport de virtual hosting, es refereix a si està suportat el fet de controlar més d'un domini en un sol servidor.
- Suport de CGI i FastCGI, tipus d'aplicacions que s'executen al costat del servidor. El servidor executa un programa extern en cada petició.
- Suport de Servlet. Tipus d'aplicacions Java que s'executen al costat del servidor.
- Suport de SSI, és un llenguatge per a petites aplicacions (scripts) que s'executen al costat del servidor.
- Suport d'aplicacions web ASP.NET
- Si tenen o no Interfície gràfica

Servidor (SW)	OpenSource	HTTPS	Virtual Hosting	CGI	FastCGI	Servlet	SSI	ASP.NET	Interf. Gràfica	SO *
Abyss Web Server	No	Si	Si	Si	Si	No	Si	Si (W)	Si	Si
Apache HTTP Server	Si	Si	Si	Si	Si	No	Si	No	Si	Si
Apache Tomcat	Si	Si	SI	Si	No	Si	Si	No	Si	Si
Cherokee HTTP Server	Si	Si	Si	Si	Si	No	Si	?	Si	Si
IIS	No	Si	Si	Si	Si	No	Si	Si	Si	No
Lighttpd	Si	Si	Si	Si	Si	No	Si	?	No	Si
Oracle HTTP Server	No	Si	Si	Si	Si	No	Si	No	Si	No
Caudium	Si	Si	Si	Si	Si	Si	Si	?	?	No
bozohttpd	Si	Si	Si	Si	No	No	No	No	No	No
LiteSpeed Web Server	No	Si	Si	Si	Si	No	No	?	Si	No
Sun Java System Web Server	No	Si	Si	Si	Si	Si	Si	No	Si	Si
MyServer	Si	Si	Si	Si	Si	?	?	?	Si	Si
Fnord	si	?	Si	Si	No	No	No	?	?	No
THttpd	Si	Si	Si	Si	No	No	No	No	No	Si
Null Httpd	Si	Si	Si	Si	No	No	No	No	No	Si
WxWebServer	Si	Si	Si	Si	?	No	?	No	Si	Si
HFS	Si	Si	Si	Si	No	No	No	?	?	No

Figura 8. En la taula comparativa disposem d'una sel·lecció dels servidors web més utilitzats a Internet, tant lleugers com no-lleugers. D'entre tots, destacarem IIS, Apache i Abyss en el segment estàndard; com a lleugers, destacarem el Lighttpd i LiteSpeed. Del primer bloc cal destacar com a punt a favor per l'Apache que és gratuït. En el segment dels servidors lleugers podem veure que Lighttpd és més flexible i és gratuït.

3.2 Tecnologies utilitzades

A continuació recordem les principals funcionalitats del projecte:

1) Mostreig de la posició i ruta seguida per l'avió en temps real
2) Càmeres
3) Històric

Tots els punts anteriors representen aplicacions que s'executen a petició del Google Earth quan hi ha una actualització en els seus enllaços de xarxa. D'altra banda serà necessari que aquestes aplicacions consultin arxius de text d'entrada/sortida. Això podria concloure en el següent:

1) Cada petició representa una execució. Amb l' obertura, cerca i tancament d'arxius d'entrada/sortida que això representa.
2) Cada petició representa una execució, per tant un nou procés diferent al processador cada vegada

Per pal·liar aquests punts buscàvem una solució en la qual cada petició no representés un procés diferent cada vegada i que els arxius d'entrada i sortida que utilitzen els serveis web estiguessin carregats sempre en memòria, en comptes d'obrir-los i tancar-los cada vegada.

La solució va ser ASP.NET, treballant amb el *Framework* Visual Studio 2005 de Microsoft. Aquesta tecnologia ens permet resoldre els punts abans comentats. Tenim una pàgina web que retorna un codi KML(XML) en cada petició, on cadascuna de les peticions representen un Thread (o fil d'execució) d'un mateix procés principal. Els arxius romanen carregats a memòria sempre que el procés estigui en execució, la qual cosa estalvia recursos en quant a temps i càrrega del processador . Una pàgina web en ASP.NET pot estar implementada en diferents llenguatges de programació, l'elecció escollida va ser C#, igual que el programa central d'aquest projecte.

Després del anàlisi de les necessitats del projecte i amb les idees presentades anteriorment, vàrem fer l'elecció del servidor web. En primer lloc vàrem prescindir de servidors web lleugers ja que principalment treballen servint contingut estàtic [5]; i en segon lloc necessitàvem un servidor que pogués treballar amb pàgines ASP.NET. L'escolli't, per tant, va ser el servidor Internet Information Services (IIS versió 7) de Microsoft. Molt recentment (després d'implementar la plataforma) vam assabentar-nos que Apache també és capaç de treballar amb ASP.NET [14]. En el moment de l'elecció no teníem encara la opció d'Apache per triar, en qualsevol cas, això és un benefici ja que augmenta la compatibilitat de la nostra plataforma.

3.3 ASP.NET

Les aplicacions web basades en Active Server Pages (ASP) pertanyen a una tecnologia de Microsoft. S'executen al costat del servidor i serveixen principalment per generar pàgines web amb contingut dinàmic. Aquesta tecnologia ha estat comercialitzada com a un complement del servidor web Internet Information Service (IIS). L'evolució d'aquesta tecnologia fou ASP.NET. Desde 2002, l'ASP *clàssic* s'està substituint per ASP.NET, que entre d'altres avantatges, substitueix els llenguatges *interpretats* (com *VBScript o JScript*) per llenguatges *compilats* com Visual Basic, C#, o qualsevol altre llenguatge que estigui suportat per la plataforma. ASP.NET és un marc de desenvolupament per aplicacions web, s'utilitza per a la construcció de llocs web dinàmics, aplicacions web i serveis web XML. Aquesta tecnologia sorgeix com a resposta a la necessitat de crear una eina potent alhora de desenvolupar aplicacions web complexes en les quals es barregen etiquetes de varis llenguatges (HTML, XML...) amb *scripts* de codi i plataformes de servidor.

El model Code-behind

Aquest model de programació diferencia molt ASP d'ASP.NET. Es recomana que per a realitzar una programació dinàmica d'aplicacions web (amb ASP.NET) s'utilitzi el model *code-behind*. Aquesta modalitat de programació es caracteritza per col·locar el codi de l'aplicació en un fitxer separat o en una etiqueta *script* especialment dissenyada. Les extensions dels arxius *code-behind* estàn basats en el nom d'arxiu ASPX, aquesta pràctica es realitza per defecte en Microsoft Visual Studio. Alhora d'utilitzar aquest estil de programació el desenvolupador escriu el codi corresponent als esdeveniments (Càrrega de la pàgina, *Click* a un botó etc) en funcions o mètodes, en comptes d'un recorregut lineal de l'arxiu de codi. El concepte, per tant, és la construcció d'aplicacions diferenciant entre presentació i contingut. És el mètode que s'utilitza en les aplicacions web que formen aquest projecte.

4 Implementació

En aquest apartat del projecte explicarem de manera exhaustiva cadascuna de les parts del projecte i el seu funcionament. Hi haurà explicacions del funcionament global de tota la plataforma i podrem veure al detall el recorregut del flux d'informació des de que surt del simulador fins que veiem al Google Earth el resultat.

4.1 Explicació per mòduls

4.1.1 Simulador Flight Gear

És la primera part del projecte; l'objectiu principal d'aquest mòdul és obtenir les dades que ens interessen de la posició de l'aparell en un instant de temps determinat. Primer hem de decidir quines de les dades que pot presentar un avió necessitem, les dades sel·leccionades són les següents:

- **Longitud:** Expressa la distància angular entre el *meridià de Greenwich* (meridià 0º) i el meridià d'un punt donat, els meridians terrestres són cercles complets que van de pol Sud a pol Nord. Pot ser Longitud Est o Oest, segons estigui posicionat cap a orient o ponent del meridià 0º. Els valors s'expressen en graus entre 0º a +180º i 0º a –180º, amb positivitat cap a l'Est i negativitat cap a l'Oest.
- **Latitud:** Expressa la distància angular, mesurada sobre un meridià, entre la línea de l'*Ecuador* i el paral·lel d'una localització terrestre concreta. Es mesura en graus. A l'hemisferi nord és positiva i negativa l'hemisferi sud. Varia entre 0º y 90º (nord) y entre 0º i - 90º (sud). També es pot descriure com la distància angular entre qualsevol punt de la Terra i el Paral·lel 0.
- **Altura** de l'avió en el moment de la consulta.
- **Velocitat** de l'avió en el moment de la consulta.
- **Heading o Yaw:** És el moviment del avió que oscil·la entre els eixos X i Y de la figura següent, considerarem aquest valor com la direcció del avió en el moment en que es mostra.

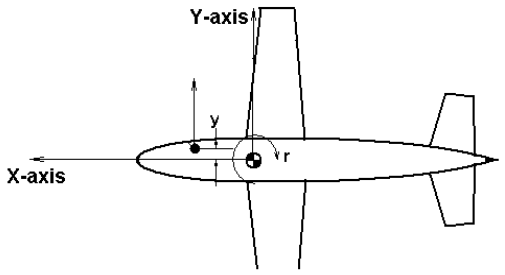


Figura 9. *Representació de la Variable Heading o Yaw que presenta un avió enlairat.*

El valor de *Yaw* ens és molt útil. L'icona de l'avió utilitzada al Google Earth pren el valor del *Yaw* en cada refresc de la ruta. Ens indica en tot moment cap a on es dirigeix l'avió.

- **Roll:** És el moviment de les ales de l'avió que oscil·la entre els eixos Y i Z de la figura següent.

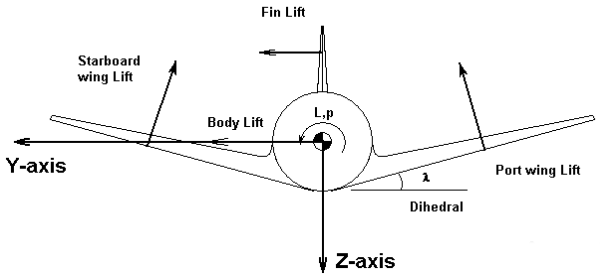


Figura 10. *Representació de la variable Roll que presenta un avió enlairat.*

- **Pitch:** És el moviment de l'avió que oscil·la entre els eixos X i Z de la figura següent. És l'inclinació cap amunt (foto) o cap avall de la cabina. Tant el valor Pich com el Roll ens són útils alhora del mostreig de l'avió en l'apartat de les vistes i càmeres.

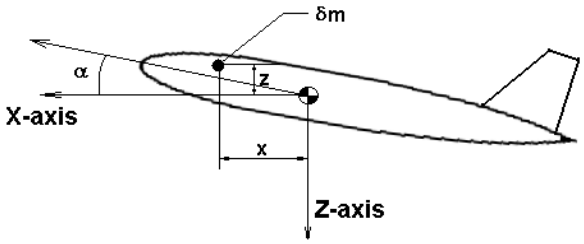


Figura 11. *Representació de la variable Pitch o Tilt que presenta un avió enlairat.*

El simulador de vol utilitzat ens permet indicar quines són les dades que volem extraure abans de començar el vol, així com la manera de representar-les posteriorment.

Les instruccions de les dades que volem rebre les fem mitjançant un arxiu XML. En aquest arxiu especifiquem els elements que ens interessen i les unitats de mesura que s'utilitzaran. Les unitats de mesura venen predefinides en uns fulls de configuració pròpies de l'aplicació, que estan donades per defecte. Per exemple, l'alçada del avió s'expressa en peus. Cal que modifiquem aquest arxiu (seguint la jerarquia d'etiquetes que utilitza *Flight Gear*) per que l'aplicació sàpiga quines dades ha de mesurar durant el vol, i posteriorment, enviar-les de la manera que nosaltres ho configurem.

```
<?xml version="1.0" ?>
<PropertyList>
  <generic>
    <output>
        <line_separator>newline</line_separator>
        <var_separator>tab</var_separator>
      <chunk>
          <name>latitude-deg</name>
          <type>float</type>
          <format>%f</format>
          <node>/position/latitude-deg</node>
      </chunk>
      <chunk>
          <name>longitude-deg</name>
          <type>float</type>
          <format>%f</format>
          <node>/position/longitude-deg</node>
        </chunk>
        <chunk>
          <name>altitude-ft</name>
          <type>float</type>
          <format>%f</format>
          <node>/position/altitude-ft</node>
        </chunk>
            .............
            .............
```

***Codi 6**. Exemple de configuració de variables de sortida del Simulador, porció del arxiu Telemetry.XML.*

Gràcies a l'arxiu propi de configuració del Flight Gear, *Telemetry.XML* especifiquem les dades de posició de l'avió , que necessitem per duur a terme el nostre projecte. En el codi anterior,que és una part del codi que utilitzem, cada etiqueta <chunk>...</chunk> representa cada un dels valors per obtenir. Dintre de cada etiqueta és obligatori especificar el nom de la variable, el tipus de variable amb la qual es representarà , el format i el node on hi hà definits els paràmetres per al càlcul d'aquesta variable.
La configuració del sistema de comunicació (per part del simulador) de sortida de dades que utilitzarem es fa a través d'un menú molt intuïtiu. El mode de sortida que hem utilitzat en aquest projecte és mitjançant un *socket* UDP, concretament en el port 7789, amb una freqüència d'enviament de paquets d'1 Hz (1paq/seg). Un *Socket* designa un concepte abstracte mitjançant el qual dos aplicacions (en la mateixa màquina o en xarxa) poden intercanviar-se un flux de dades. Un *socket* queda definit per una adreça IP, un protocol i un número de port. Per que dos programes puguin comunicar-se entre si és necessari que un programa sigui capaç de trobar a l'altre, i que tots dos siguin capaços de intercanviar-se seqüències de bytes intel·ligibles. Per tot això és necessari els tres recursos abans comentats. És necessari un protocol de comunicacions a nivell de xarxa (si les aplicacions estan a la mateixa màquina utilitzaran la

adreça *loopback* de la tarja) que permeti l' intercanvi de bytes, així com una adreça de protocol de xarxa que identificarà a la màquina o màquines. També és necessari un número de port lògic que serà l'identificador dels programes dintre de l'ordinador. Els *sockets* permeten implementar un arquitectura *Client-Servidor*.
La comunicació ha se ser iniciada per un dels dos programes (aquest farà de Client), en el nostre cas serà l'aplicació principal del projecte. El Servidor serà el simulador de vol que esperarà a que el primer inicií la comunicació. Un *socket* és com un "arxiu" existent a les dues màquines [7] (o lògicament en una si treballem en local) que serveix en última instància per que el programa servidor i el client llegeixin i escriguin informació a l'interior. Aquesta informació serà transmesa per les diferents capes de xarxa del model OSI. Les propietats d'un *socket* depenen de les característiques del protocol de transport utilitzat.

Nosaltres utilitzem UDP per la seva senzillesa de funcionament i per no ser un protocol orientat a connexió. Alhora de treballar les dues aplicacions en una mateixa màquina, format el "servidor d'informació" del projecte, no són necessaris el control de flux ni el control congestió propis de TCP. Tampoc l'ordenació de paquets i la garantia d'arribada de tots els bytes. El protocol UDP ens resulta ideal al ser un protocol sense mecanismes de garantia de servei i per tenir una capçalera de datagrama molt més reduïda que TCP. A la nostra plataforma li convé un protocol senzill. En quant a la càrrega, el fet de que el flux d'enviament de paquets no sigui molt alt però el processament posterior (provinent dels aplicatius d'aquest projecte) sigui elevat, la senzillesa d'UDP és un benefici.
Quan nosaltres executem l'aplicació principal (després de configurar correctament el simulador), se'ns activa el botó "Connexió FG" . Quan *Clickem* aquest botó es posa en marxa el Thread que s'encarrega d'escoltar el port UDP 7789 i és quan comença el procés de rebuda d'informació.
En el **Codi 7** següent veurem el nucli de la funció del programa principal on es reben les dades: bàsicament definim un punt IP que descriurà el node de la comunicació remota, en aquest cas és *Localhost*. Posteriorment, instanciem una cadena de bytes on posar el resultat de l'escolta al port i el codifiquem a codi ASCII per transformar els bytes en caràcters (números). Aquests caràcters seran utilitzats en altres funcions. Cal puntualitzar que s'utilitza *Localhost* perquè la comunicació entre aquests dos programes es farà sempre en local.

```
IPHostEntry ipHostInfo = Dns.Resolve("localhost");

IPAddress ipAddress = ipHostInfo.AddressList[0];

IPEndPoint RemoteIpEndPoint2 = new IPEndPoint(ipAddress, 7889);
Byte[] receiveBytes = udpClient.Receive(ref RemoteIpEndPoint2);

string returnData = Encoding.ASCII.GetString(receiveBytes);
```

***Codi 7**. Nucli de la funció encarregada de rebre els datagrames UDP.*

4.1.2 Aplicació principal

Abans de fer l'explicació detallada del funcionament de l'aplicació principal, veurem a continuació un gràfic on es representa l'ubicació dels diferents elements de la plataforma. Podem veure com l'aplicació principal rep l'informació del simulador, aquesta informació és tractada per l'aplicació principal i es dipositada en arxius de sortida (arxius de text i gràfiques). Les aplicacions web consultaran aquests arxius per obtenir l'informació de posicionament de l'avió, d'aquesta manera podran generar informació útil pel Google Earth.

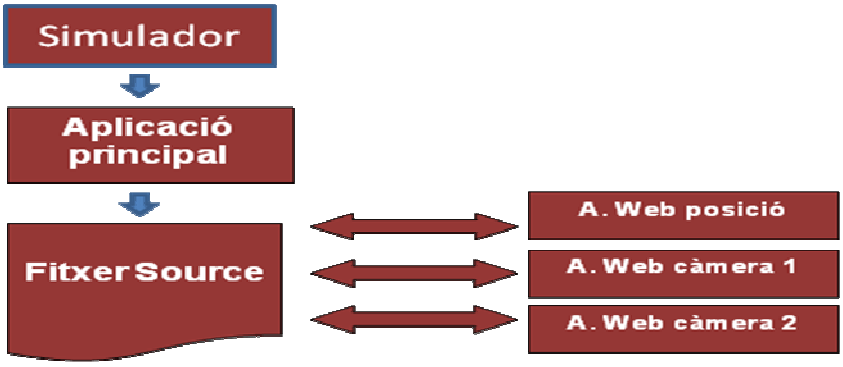


Figura 12. *Diferents elements que formen la Plataforma*

4.1.2.1 Input o valors d'entrada a l'Aplicació principal

Els inputs de la nostra aplicació els podríem definir en dos blocs. En un primer bloc definiríem els esdeveniments que rep el programa abans de posar-se en marxa (*Clickar* el botó Connexió FG, seria l'*start*), el segon bloc serien els valors de les variables de posició que ens envia el simulador. L' esdeveniment principal que rep el programa abans de posar-se en marxa es l'elecció de la IP del servidor. La raó per la qual necessitem saber l'adreça IP amb la que treballa el servidor és la següent:
Recordant la lògica utilitzada per la jerarquia KML i Google Earth alhora de fer les actualitzacions *(Updates),* veiem que tenim un arxiu original i un altre amb les modificacions pertinents. Doncs bé, en cadascuna de les actualitzacions necessitem especificar quin és l'arxiu que estem modificant; per donar-li aquesta informació al Google Earth, ens cal saber l'adreça IP de la màquina en la que estem treballant. S'utilitza el següent codi en les aplicacions web:
Programació:

```
w.WriteStartElement("targetHref");
w.WriteCData("http://"+IPserv+"/Primerpunt.kml");
w.WriteEndElement();
```

Resultat: <targetHref>http://192.168.1.4/Primerpunt.kml</targetHref>

Codi 8*. Implementació del path correcte i necessari pels Updates KML.*

On “**Ipserv**” és una variable que ens especifica la IP del servidor en aquell moment i **“w”** és un context de escriptura de sortida XML per l’aplicació web. Obliguem al usuari a escollir l’adreça en el programa principal abans de començar la missió (no l’obtenim en l’aplicació web). Es detecten totes les adreces IP de la màquina amb una funció pròpia i l’usuari fa l’elecció. Aquesta IP és guardada en un fitxer de text per què sigui consultada posteriorment pels *Sites* Web.
Com a especificació del segon tipus d’inputs que rep l’aplicació principal, les dades de posició de l’avió són rebudes un cop per segon (provinents del simulador via *Socket*) i són tractades per ser intel·ligibles als *Sites* web i per permetre’ns dibuixar les gràfiques d’Altura i Velocitat. La periodicitat d’enviament es configurable.

4.1.2.2 Tractament de les dades en l’aplicació principal

Principalment en aquest apartat, tractarem dos conceptes bàsics pel funcionament de la plataforma de monitorització. En primer lloc, veurem com ubiquem les dades de posició rebudes i en segon lloc veurem com construïm les gràfiques que després seran enviades als hosts que ho demanin.
Un cop rebem un datagrama UDP, la funció explicada anteriorment (**Codi 7**) codifica a ASCII els bytes que rep. Posteriorment aquesta funció envia una cadena de caràcters a la funció encarregada de dividir les variables adequadament. Aquesta cadena de caràcters inclou tots els valors que demanàvem al Simulador de vol (mitjançant l’arxiu Telemetry.XML). Aquestes dades estan dividides mitjançant un caràcter auxiliar com per exemple una tabulació (“\t”). Per cada cicle, és a dir, per cada datagrama rebut (en endavant anomenarem cicle a aquest procés), separem les variables i les assignem a una estructura pròpia de dades, que en cada cicle pren valors diferents. En principi això podria semblar poc eficient, però no és així, ja que en cada cicle es processa l’informació i es deixa intel·ligible en fitxers de sortida de diferents tipus, l’aplicació mai es queda aquesta informació com a històric. També disposem d’una estructura de dades auxiliar idèntica a la primera que servirà per guardar la posició del datagrama actual per al pròxim cicle. Durant el següent cicle es convertirà en les dades de la posició anterior. De la mateixa manera que la estructura principal, aquesta estructura de dades es sobreescriu.

Un cop per cada datagrama rebut, afegim en un fitxer de text de sortida *(Source*) els valors de les variables que obtenim; cada datagrama representa una línia d’aquest arxiu. Abans d’entrar a la funció de tractament de dades, a cada datagrama lògic se li assigna un identificador únic i incremental per portar un control de tots els que entren en el sistema. Cal remarcar que a l’entrada d’aquesta funció, primer es comparen les dades de coordenades amb les de la posició anterior. Els datagrames amb coordenades iguals a l’anterior es descarten. La raó és molt senzilla, si alhora de començar un vol l’avió roman a la pista d’aterratge molta estona, no té sentit inundar l’arxiu de text de sortida amb moltes línies iguals entre elles. Sempre s’intenta aconseguir un fitxer de sortida amb les dades més concises possibles perque el mostreig resulti eficient.

Buscant un recurs adequat per a l'elaboració de les gràfiques vam trobar unes llibreries encarregades exclusivament per aquest afer; les llibreries emprades per elaborar les gràfiques són les anomenades *ZedGraph* [13]. Ens permeten obtenir uns resultats de molt bona qualitat, molt intuïtives de programar i eren exactament el que estàvem buscant. A grans trets ,la programació de les gràfiques funciona de la següent manera:

En primer lloc, definim les corbes de la funció per a cada gràfica:

```
this.zedGraphControl1.GraphPane.AddCurve("VELOCITAT",this.LlistaVeloci
tat2, Color.Red, SymbolType.Diamond);
this.zedGraphControl2.GraphPane.AddCurve("ALTURA",  this.LlistaAltura2,
Color.Blue, SymbolType.Diamond);
```

***Codi 9**. Definició de les corbes per a cada gràfica*

On "**LlistaVelocitat2**" i "**LlistaAltura2**" són dos estructures de dades on afegim les dades (tant en l'eix X com l'Y) en cada cicle de l'aplicació de forma incremental, és a dir, són vector de punts de les gràfiques que es conserven durant tota la missió, i que van creixent de forma lineal.

```
LlistaVelocitat2.Add(PuntEixX, valorVelocitat);
      this.zedGraphControl1.AxisChange();
      this.zedGraphControl1.Refresh();
LlistaAltura2.Add(PuntEixX, valorAltura);
      this.zedGraphControl2.AxisChange();
      this.zedGraphControl2.Refresh();
```

***Codi 10**. Procés que es duu a terme per cada datagrama en quant a les gràfiques.*

On "**PuntEixX**" i "**ValorVelocitat**" són l'identificador de datagrama i el valor de la velocitat del aparell en aquell cicle respectivament, i on "**PuntEixX**" i "**ValorAltura**" són l'identificador de datagrama i el valor de l'Altura del aparell en aquell cicle respectivament. La comanda "**Add**" afegeix cada dupla de dades en una posició nova del vector. Finalment, Actualitzem els arxius jpg a la carpeta del servidor:

```
this.zedGraphControl1.DrawToBitmap(this.graficaV,this.zedGraphControl1
.Bounds);
this.graficaV.Save("C:\\inetpub\\wwwroot\\GraficaVelocitat.jpg");

this.zedGraphControl2.DrawToBitmap(this.graficaA,this.zedGraphControl1
.Bounds);
this.graficaA.Save("C:\\inetpub\\wwwroot\\GraficaAltura.jpg");
```

***Codi 11**. Actualització del contingut de les gràfiques en cada cicle.*

4.1.2.3 Output o valors de sortida del programa principal

En primer lloc, l'adreça IP del servidor és escollida dintre de les adreces IP disponibles de la màquina per l'administrador i el valor d'aquesta és guardat en un fitxer a la carpeta del servidor IIS7. l'anomenem "IPServ.txt".

En segon lloc tenim el fitxer de text on guardem les dades de posició "Source.txt", com es comentava abans cada línea representa un datagrama rebut per l'aplicació, el fitxer presenta el següent aspecte:

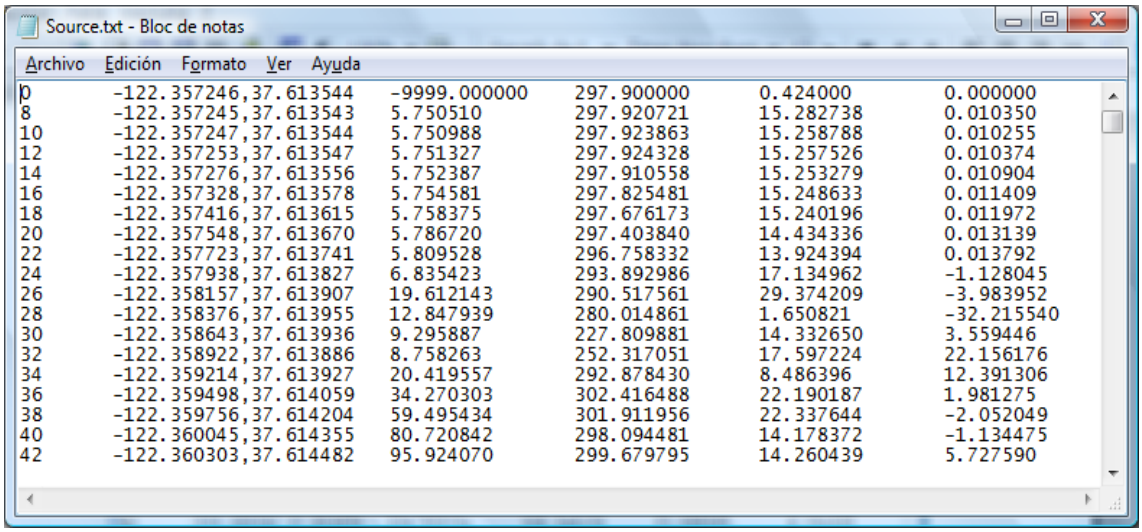


Figura 13. *Arxiu Source.txt, nucli de la comunicació entre els aplicatius d'aquest projecte. Proporciona les dades necessàries pels llocs webs i és actualitzat en cada cicle per l'aplicació principal.*

Cada variable està separada per una tabulació. En primer lloc trobem l'identificador de datagrama, en segon lloc trobem les coordenades terrestres juntes, tant longitud com latitud. Posteriorment trobem l'altura del avió; les últimes tres columnes són *Heading*, *Pitch* i *Roll* respectivament.

En l'apartat de tractament de l'informació s'ha presentat la manera en què es generen les gràfiques. A continuació explicarem el seu resultat i com apareixen finalment en el Google Earth. Per cada datagrama rebut en l'aplicació principal, es sobreescriu l'arxiu d'imatge codificat en *jpg* amb l'inclusió de les noves dades tant a la gràfica d'altura com en la de velocitat. Aquest arxiu està situat a la carpeta del servidor per què es serveixi amb més facilitat (exemple "http://192.168.1.4/GraficaAltura.jpg"). Aquestes imatges són servides directament al Google Earth amb la propietat descrita en l'apartat 2.1.3 anomenada *ScreenOverlays*. Mitjançant aquesta propietat ordenada amb l'execució d'un arxiu KML , Google Earth es descarrega automàticament aquesta imatge amb una periodicitat configurable, en aquest cas és de tres segons. El resultat és satisfactori, el presentem a continuació:

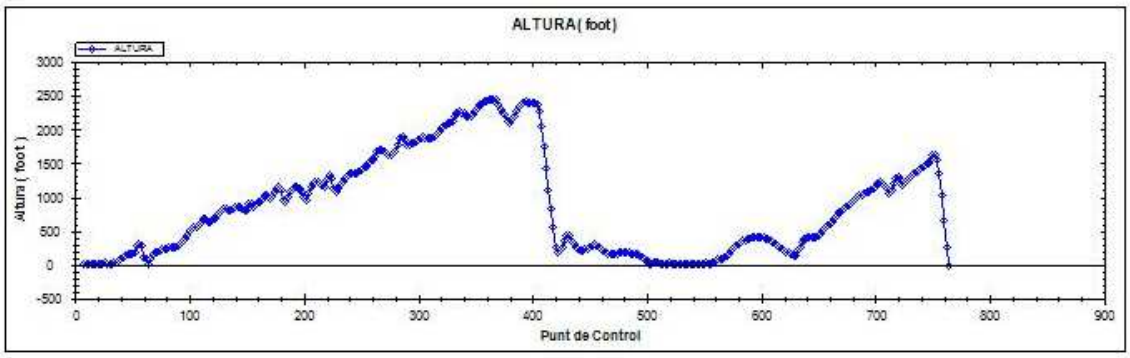


Figura 14 *. Gràfica Altura: Expressada en Peus, l'eix d'absices presenta el punt de control en que s'ha pres la mesura, coincideix amb l'identificador de datagrama i representa el temps.*

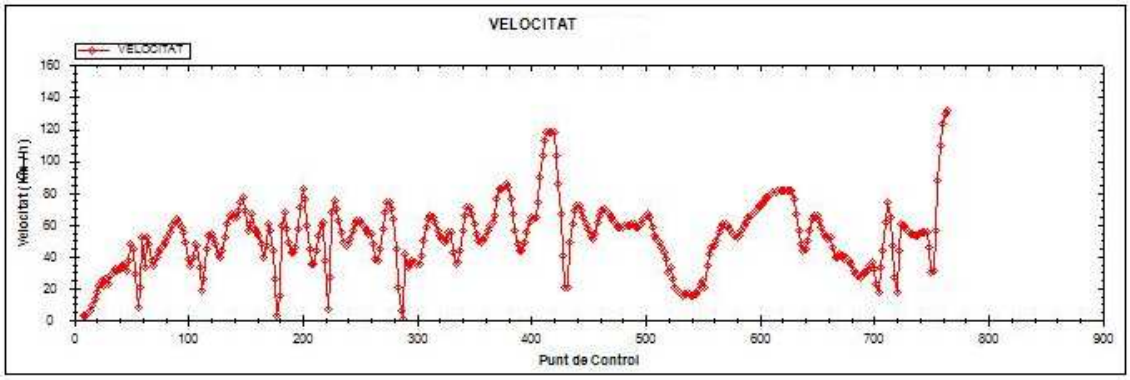


Figura 15*. Gràfica Velocitat: Expressada en Milles per hora (mph), l'eix d'absisses presenta el punt de control en que s'ha pres la mesura, representa Temps i conincideix amb ID datagrama.*

Un aspecte molt important d'aquest apartat d'especificació d'arxius de sortida, és l'aparició d'un altre tipus de fitxers de sortida diferents als arxius de text i els *bitmaps* (gràfiques). També la nostra aplicació genera arxius KML. Aquests arxius KML es generen en dos moments molt determinats de la missió que estem realitzant. El primer arxiu es genera quan es processa el primer paquet de dades rebut pel simulador. Aquest fet es dóna en referència a la manera d'implementar les actualitzacions de Google Earth i els arxius KML (veure secció 2.1.2) , recordem que és necessari un arxiu original i les actualitzacions periòdiques. Doncs l'arxiu original és generat amb l'aplicació principal i conté l'informació del primer datagrama UDP que envia el simulador; és el punt de partida i les coordenades pertanyen al a la pista d'aterratge del aeroport de sortida. Aquest arxiu és guardat a la carpeta arrel del servidor IIS7 i serà utilitzat pel Google Earth. La resta de datagrames que van arribant a l'aplicació es van tractant normalment tal i com s'ha explicat fins ara, s'utilitzen per actualitzar l'arxiu *Source* i construir les gràfiques.

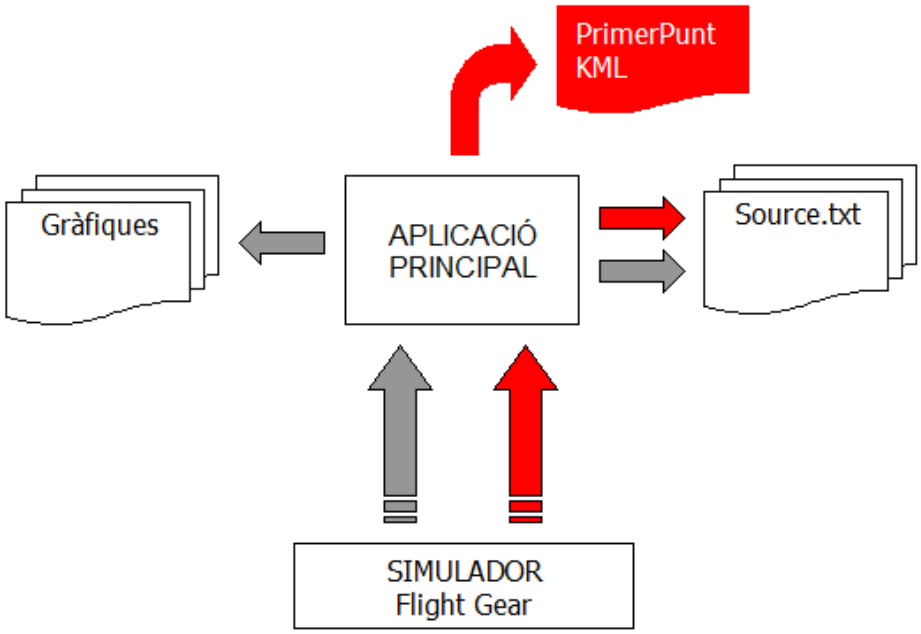


Figura 16 : *En vermell s'indica el primer cicle de l'aplicació i en gris tots els cicles següents, s'inicien a partir que clickem el botó d'inici de l'aplicació.*

El segon arxiu KML de la nostra aplicació ,es genera quan sortim de la mateixa, en el event *FormClosing* de l'aplicació principal. Aquest aglutina tot el recorregut seguit per l'avió mostrant l'altura en cada moment. Aquest arxiu es genera en motiu d'històric de missions.

Alhora de dissenyar aquesta plataforma es va creure convenient dotar-la de un sistema que emmagatzemés tota l'informació relativa a les missions un cop aquestes estiguin finalitzades. Aquest fet resulta molt útil ja que ens permetrà poder accedir a les missions anteriors remotament i descarregar-nos el seu contingut en un arxiu KML executable en el Google Earth.

Un cop la nostra aplicació rep l'event *FormClosing,* automàticament obre el fitxer "Source.txt" , el llegeix i emmagatzema totes les dades de posició en una cadena de text. Posteriorment, insereix en un arxiu KML (utilitzant les etiquetes pròpies per definir rutes de molts punts en l'espai terrestre) aquesta cadena de text. Aquest arxiu es guarda en una carpeta ubicada dins del sistema arrel del servidor web IIS7. Per realitzar aquest "servei" és imprescindible una Base de Dades on poder guardar informació relativa a l'ubicació dels arxius generats automàticament. La nostra Base de dades és molt senzilla, només té una taula amb tres camps. Degut a la seva senzillesa es va implementar en *Microsoft Access*, que ens proporcionava sobradament les nostres pretensions. Apart, al ser de la mateixa companyia que el marc de treball alhora de desenvolupar la programació , les consultes SQL van resultar molt intuïtives de programar. La Base de dades consta del camp Identificador de Missió, nom de la missió i *path* o ruta d'accés a la mateixa. L'identificador és un camp auto-numèric que fixa el propi *Microsoft Acces* quan introduïm un nou registre. El nom de la missió està format per el dia i l'hora de la seva finalització amb el format següent: DDMMAAHHMMSS (dia, mes, any, hora, minut, segon) amb la referència

horària UTC. El camp més important però és el camp de ruta d'accés. Guardem en el següent format la adreça web per després poder-nos descarregar el fitxer KML:

"http://IpServidor/Missions/NomArxiu.kml"

Un exemple del dia 15/09/2008 a les vuit i quinze minuts de la tarda seria el següent: "http://192.168.1.4/Missions/15092008201508.kml"

Cal recordar que la IP del servidor s'assigna automàticament a la IP escollida en l'aplicació justament abans de començar la missió. D'aquesta manera l'aplicació pot ser utilitzada en qualsevol màquina.

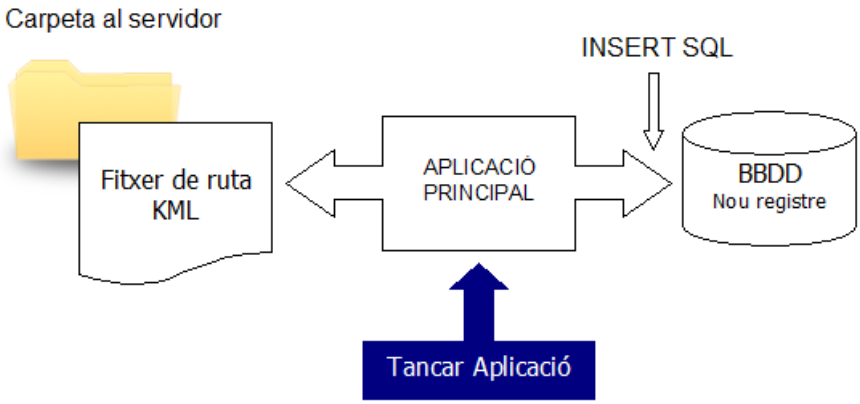


***Figura 17** : Procés que es duu a terme després de tancar la aplicació, per a la gestió d'un històric de missions.*

Al finalitzar la missió es genera una consulta INSERT SQL a la base de dades que genera un nou registre per poder accedir posteriorment. A la figura següent podem veure un diagrama del funcionament global de l'aplicació. Hi ha diferenciades en àrees els esdeveniments principals de l'aplicació. En primer lloc (Pas num. 1) es tria l'adreça IP del servidor, és imprescindible per poder passar al Pas num. 2, que es tracta de la monitorització de la missió en si. L'últim pas representa la finalització de la missió, on es creen els arxius d'històric.

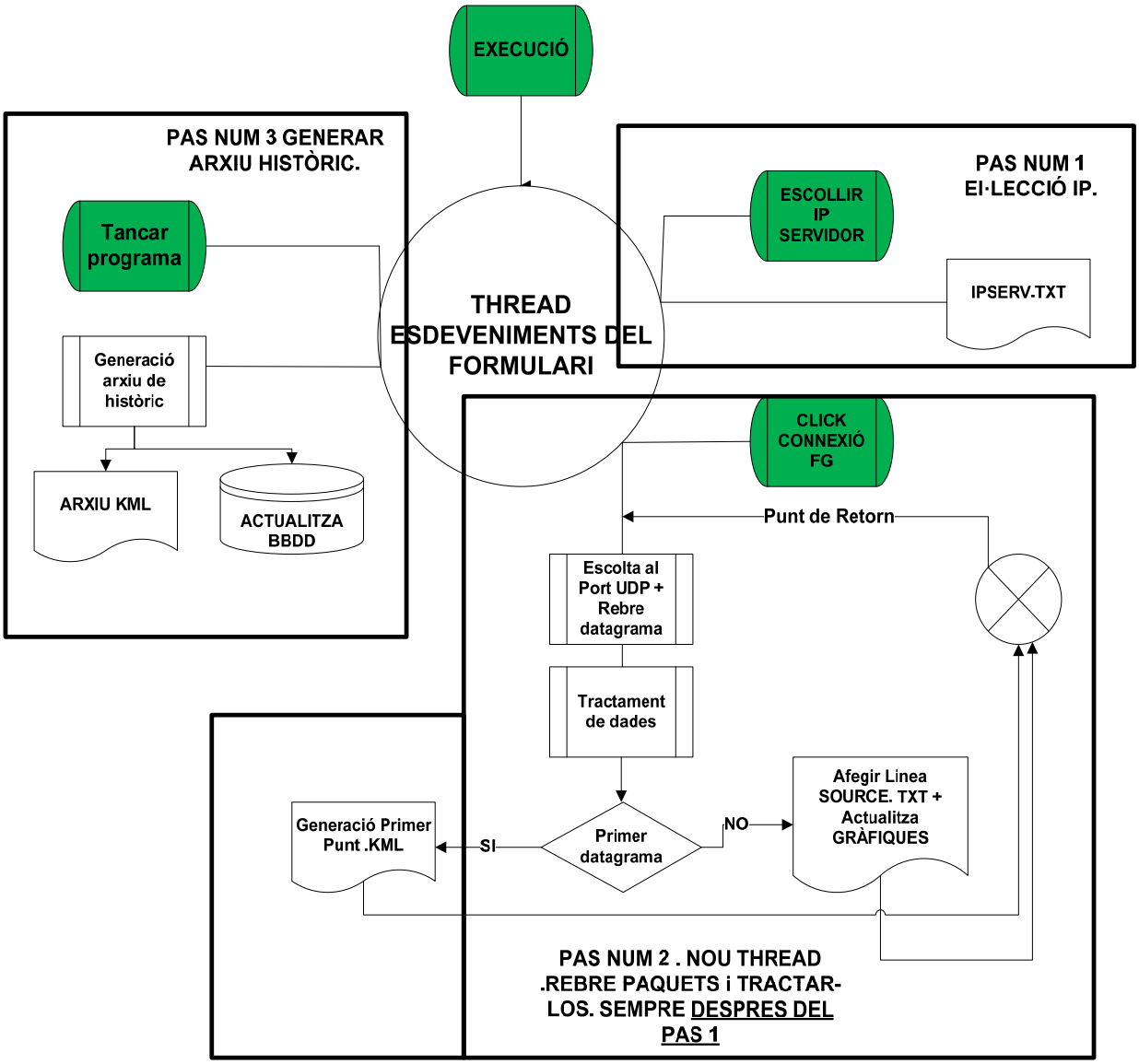


Figura 18 : *Diagrama del funcionament de l'aplicació Principal.*

4.1.3 Posició i ruta

Quan el Google Earth fa una petició HTTP (HTTP Request) per demanar una actualització de la ruta de l'avió, aquesta aplicació és la que respon. La manera de demanar aquesta petició es fa via Enllaç de Xarxa . Aquest aplicatiu segueix la filosofia de la programació de pàgines web (*Sitio web ASP.NET*) . El llenguatge propi és C#. La característica més important d'aquesta pàgina web dista molt del concepte més general que es té avui en dia dels *Sites* web de consulta més generals. Parlem de què en aquesta aplicació el cos propi de codi HTML està buit, només mantenim la capçalera de la plana i només s'especifica que el codi que en treurà posteriorment serà XML (KML). Els llocs web

ASP.NET es divideixen en dos grans blocs, el primer bloc consta de la aparièncía de la pàgina, com per exemple amb les fotos, links, DataGrids , títols, menús etc. El segon bloc consta de la programació, ja sigui en C# o Visual Basic que es durà a terme depenent dels esdeveniments que nosaltres produïm alhora de la consulta de la web. Per tant es segueix el model *Code-Behind* (veure apartat 3.3.1)

Clarament, a nosaltres el primer bloc de construcció de llocs web ASP.Net no ens interessava, ja que el nostre objectiu no era l'obtenció d'una plana web, sinó que el que volíem era retornar un codi XML que Google Earth sigui capaç d'entendre. Més concretament, volem que Google Earth dibuixi un tros de ruta entre dos punts donats i dibuixi la nova posició de l'aparell. En cap moment es mostra cap plana web, si no que és una transmissió d'informació transparent a la vista de l'usuari. Com comentava abans, l'única part de la pàgina web d'aparièncía pròpiament dita que es conserva és la capçalera on s'especifica el tipus de dades que contindrà la web i la pàgina que conté el codi. La capçalera és la següent:

```
<%@ Page Language="C#" AutoEventWireup="true"  ContentType="text/xml"
Codefile="Default.aspx.cs" Inherits="_Default" %>
```

Codi 12 : *Capçalera de la pàgina principal del servei.*

Com podem veure, s'especifica com a fitxer de codi "Default.aspx.cs". Aquesta pàgina mostra el contingut que necessita el Google Earth per actualitzar l'informació. Consta de dos mòduls de programació molt diferenciats, tots dos imprescindibles i dependents entre ells, són els següents:

- Bloc 1. Funcions generadores de codi XML:

Aquesta part representa el codi que genera la sortida de dades XML. Aquest codi es generarà automàticament a partir d'unes dades, és a dir, no és codi XML escrit a mà per nosaltres amb variables anidades. És una construcció programada que rep l'indicació de la jerarquia que volem presentar amb les variables del programa que ha de presentar segons el cas. Aquesta part del codi només pot estar situada en una part molt concreta del *Site* web, en el event *Page_Load*. Quan es carrega la pàgina s'executa aquest codi. Un tret molt característic és que utilitzem la classe *XmlTextWriter* per treure aquest codi com al cos de la pàgina web, l'indicació que ha de rebre aquesta classe per treure com a codi web la seva jerarquia construïda i no deixar-la a cap fitxer de text estàtic és la següent:

```
XmlTextWriter wazz = new XmlTextWriter(Response.Output);
wazz.WriteStartDocument();
```

Codi 13 *: Instància de la sortida de Codi XML que donarà forma a l'arxiu que representarà Google Earth un cop per cicle o petició.*

El mètode *Response.Output* ens permet que aquest sigui el cos de la nostra pàgina web (per això deixàvem buit el codi HTML). La raó per la qual aquest codi només pot estar en cap altre funció que no sigui *PageLoad* (sempre s'executa) resideix aquí. A continuació veurem un exemple de creació de codi XML de sortida:

```
XmlTextWriter wazz = new XmlTextWriter(Response.Output);
wazz.WriteStartDocument();
wazz.WriteStartElement("kml");
wazz.WriteAttributeString("xmlns", "http://earth.google.com/kml/2.2");
wazz.WriteStartElement("Placemark");
wazz.WriteAttributeString("Id", "segonpunt");
wazz.WriteStartElement("name");
wazz.WriteCData("actual");
wazz.WriteEndElement();
wazz.WriteStartElement("coordinates");
wazz.WriteCData("" + Connexions[i].PosicioAnterior + ",0 " +
Connexions[i].PosicioActual + ",0");
wazz.WriteEndElement();
wazz.WriteEndElement();
wazz.WriteEndDocument();
```

***Codi 14** : Exemple de la construcció d'un arxiu KML senzill des de una funció de codi*

- Bloc 2: Funcions encarregades de la gestió de les dades.

Aquesta part del codi del *Site* web aglutina 4 funcions per complir generalment dos empreses, la primera és aconseguir les últimes dades de posicionament disponibles en el fitxer "Source.txt", la segona s'encarrega d'aconseguir la gestió més acurada possible de les sessions per als diferents usuaris d'aquesta aplicació.

Un cop s'executa aquesta aplicació, el procés relacionat es carrega en memòria i es manté aquí. Per cada consulta es crea un nou fil d'execució d'aquest procés (d'un mateix usuari o d'un altre diferent). Mai es tanca el procés i es torna a executar en cada petició, de no ser així el rendiment es veuria afectat molt negativament si el projecte s'utilitzés a gran escala. La gestió de les sessions actives es fa mitjançant vectors d'estructures de dades, on cada sessió representa una posició del vector. En cada posició del vector s'emmagatzema una estructura i aquesta conté informació sobre les variables que aniran a la part de codi del *Bloc 1*. Les més destacades són les coordenades actuals i anteriors (per dibuixar un petit tros de ruta necessitem el punt actual i l'últim punt dibuixat) i el *Heading* (que ens mostrarà la direcció de l'aparell en aquell moment de temps). Quan entra una nova petició, es comprova si hi ha una sessió activa per a aquella IP remota , si hi ha es dibuixa una nova porció de ruta amb les coordenades actuals de l'avió. Si no, es realitza un pas previ, que consisteix en afegir una nova posició al vector de sessions. La porció de ruta que es dibuixa consisteix en un únic punt, ja que no disposem de l'ultima coordenada dibuixada. Com comentàvem abans el vector de sessions es manté en memòria i podem mantindre una sessió d'usuaris durant tota la missió. A continuació veurem de forma gràfica el procés:

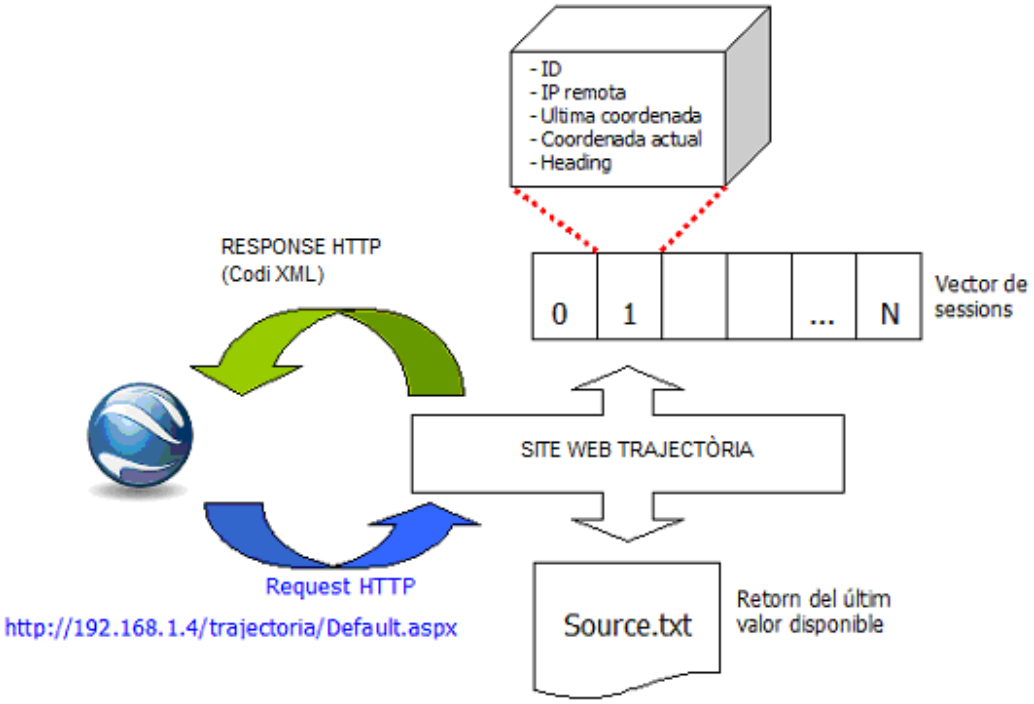


Figura 19 *: Diagrama del funcionament de l'aplicatiu de Vista trajectòria.*

Per aconseguir les dades més recents que provenen del simulador, aquesta aplicació consulta l'última entrada en el fitxer "Source.txt" i assigna les dades de posició a la sessió que es tracta en aquell moment. Per últim, aquest web recull l'adreça IP del servidor en aquell moment, aquesta està disponible en un fitxer de text gestionat per l'aplicació principal. Com es comentava anteriorment, ens cal aquesta adreça per indicar en tot moment quin és el arxiu original que estem modificant. Aquest lloc web ens proporciona la següent informació:

4.1.3.1 Vista posició en temps real

Ens proporciona la posició de l'avió en el moment de la consulta mitjançant una marca de posició (Placemark) situada al mapa del Google Earth, s'actualitza en cada petició. És a dir, es substitueixen els punts. Sempre es retorna les dades de l'última entrada del fitxer *Source* per tenir un compromís amb el temps real el més estret possible. Hem de tenir en compte que cada segon s'afegeix una nova entrada en aquest fitxer i el període entre consultes del enllaç de xarxa (*NetworkLink*) del Google Earth és configurable. El que no preteníem era que cada usuari recorregués tots els punts que hi ha al fitxer seqüencialment , ja que amb un refresc superior al refresc fix entre Simulador i programa principal (1 seg) , perdríem absolutament el concepte de temps real i es podrien presentar retards de temps molt elevats.

Caldria afegir que modificant el valor entre peticions no perdre'm mai el compromís amb el temps real (sempre hi ha una variació per processament i també un retard de xarxa, encara que és mínim degut a que en un segon dóna temps a tot el procés). Modificant el període entre peticions per un valor alt o baix sempre obtindrem la coordenada de la nau més actual. La variació que ens trobarem és la definició en la ruta. Amb un període entre peticions alt la ruta tindrà una definició baixa mentre que amb un valor petit de la mateixa, la ruta quedarà definida per més punts. Això farà que tingui un detall molt més il·lustratiu. La figura següent n'és un exemple.

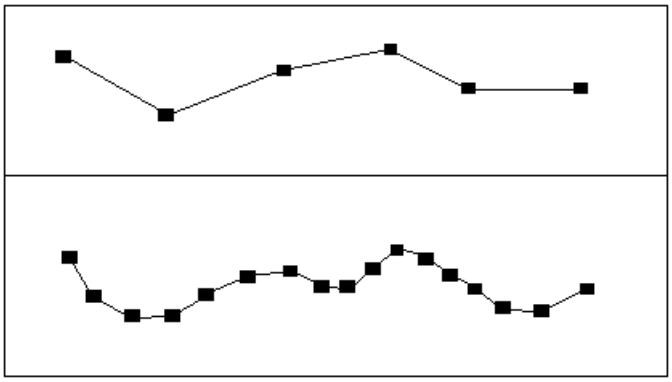


Figura 20. *El primer esquema mostra un període entre peticions alt i el segon un període més baix.*

4.1.3.2 Vista de la direcció

Idènticament al mostreig de la posició de l'avió, gràcies a la icona utilitzada (en forma d'avió) mostrem el seu *Yaw* o *Heading*. S'actualitza amb la posició i és substitueix , si és el cas, en cada petició.

4.1.3.3 Vista ruta recorreguda

A diferència de les dos característiques anteriors aquesta funciona de diferent manera. En cada petició es dibuixa una porció de ruta amb dos punts, les coordenades actuals de l'aparell, i les coordenades dibuixades en l'ultima consulta (per no tenir cap "forat" sense dibuix en la ruta). No tindria sentit que s'actualitzés aquesta informació , sinó que s'afegeix a l'estructura aconseguida fins al moment. En cas de que utilitzéssim aquest *Site* web sense cap missió en procés, retornaríem un codi KML amb les etiquetes que especifiquen les coordenades (entre d'altres) sense valors, El Google Earth no presentaria cap error però no és veuria cap actualització al mapa.

4.1.4 Càmera seguiment

Aquesta funcionalitat s'encarrega de fer un seguiment de la posició de l'aparell amb una vista en perspectiva. Mentre l'avió està fent la missió, aquest es va movent, fent el seu recorregut a sobre del mapes de Google Earth. Els mapes però, romanen estàtics i quan l'avió ha sobrepassat els límits del mapa que hi ha en aquell moment el deixem de veure, ja que aquest segueix avançant. Amb aquesta nova funcionalitat volem que els mapes del Google Earth, amb un efecte de càmera, es vagin movent al ritme del avió i que sempre tinguem l'aparell visible. De tal manera , nosaltres sense moure ni redimensionar els mapes del Google Earth, estem sempre visualitzant la posició de l'aparell (vista amb més detall -més a prop- o menys detall –més lluny-). En quant al detall dels mapes, dependrà de l'altura que tingui l'avió en aquell moment. En resum, l'efecte és com si tinguéssim instal·lada una càmera de vídeo a la part inferior de l'avió, sota el pilot per exemple i enfocant cap a Terra.

S'ha utilitzat el component o objecte "Camera" de la jerarquia KML, ja que és l'element adient per aconseguir el proposat anteriorment. Per a la seva construcció s'han utilitzat les següents dades:

```
<longitude>-122.320898</longitude>
<latitude>37.636674</latitude>
<altitude>3895.275288</altitude>
<heading>318.229660</heading>
<tilt>88,66334</tilt>
<roll>51.446306</roll>
```

***Codi 15** : Exemple de dades utilitzades en la construcció d'un arxiu de Càmera KML*

El codi emprat en l'aplicació web per a la realització del arxiu KML que s'envia al Google Earth és el següent:

```
XmlTextWriter wax = new XmlTextWriter(Response.Output);
wax.WriteStartDocument();
wax.WriteStartElement("kml");
wax.WriteAttributeString("xmlns",
"http://earth.google.com/kml/2.2");
wax.WriteStartElement("NetworkLinkControl");
wax.WriteStartElement("Camera");
wax.WriteStartElement("longitude");
wax.WriteCData("" + Connexions[i].Longitud + "");
wax.WriteEndElement();
wax.WriteStartElement("latitude");
wax.WriteCData("" + Connexions[i].Latitud + "");
wax.WriteEndElement();
wax.WriteStartElement("altitude");
```

```
wax.WriteCData("" + Connexions[i].Altitud+ "");
wax.WriteEndElement();
wax.WriteStartElement("heading");
wax.WriteCData("" + Connexions[i].heading+ "");
wax.WriteEndElement();
wax.WriteStartElement("tilt");
wax.WriteCData("" + Connexions[i].pitch+ "");
wax.WriteEndElement();
wax.WriteStartElement("roll");
wax.WriteCData("" + Connexions[i].roll + "");
wax.WriteEndElement();
wax.WriteStartElement("altitudeMode");
wax.WriteCData("clampToGround");
wax.WriteEndElement();
wax.WriteEndElement();
wax.WriteEndElement();
wax.WriteEndElement();
```

***Codi 16**: Dades utilitzades en la construcció d'un arxiu de Càmera KML*

Totes les dades necessàries són les dades reals que obtenim del Simulador i estan presents en l'arxiu "Source.txt". Aquesta capacitat del nostre projecte necessita anar-se actualitzant cada interval de temps per tenir la sensació de moviment dels mapes (el moviment del avió en la vista Posició no te res a veure en aquest sentit, aquest s'aconsegueix a part i en paral·lel). A diferència de la funcionalitat "Posició i ruta" aquest apartat no utilitza la tècnica dels *Updates KML.* Aquí no és necessari tenir un arxiu KML especificant la càmera inicial i periòdicament la creació d'arxius KML que contenen un component càmera (per substituir o afegir vistes cada fracció de temps). Només ens cal la segona proposta, és a dir, Google Earth es descarrega del nostre servidor, un arxiu KML diferent cada vegada. El contingut del qual sempre serà un element *Camera* però amb els valors diferents en cada petició.

Aquest servei disposa dels mateixos blocs de programació que el Site Web 1 Vista Posició (Punt 4.1.3), amb un bloc de generació d'un arxiu XML de sortida que serà diferent, ja que no implementa *Updates* i l'element central és *Camera*, i amb un bloc d'obtenció de dades molt semblant.

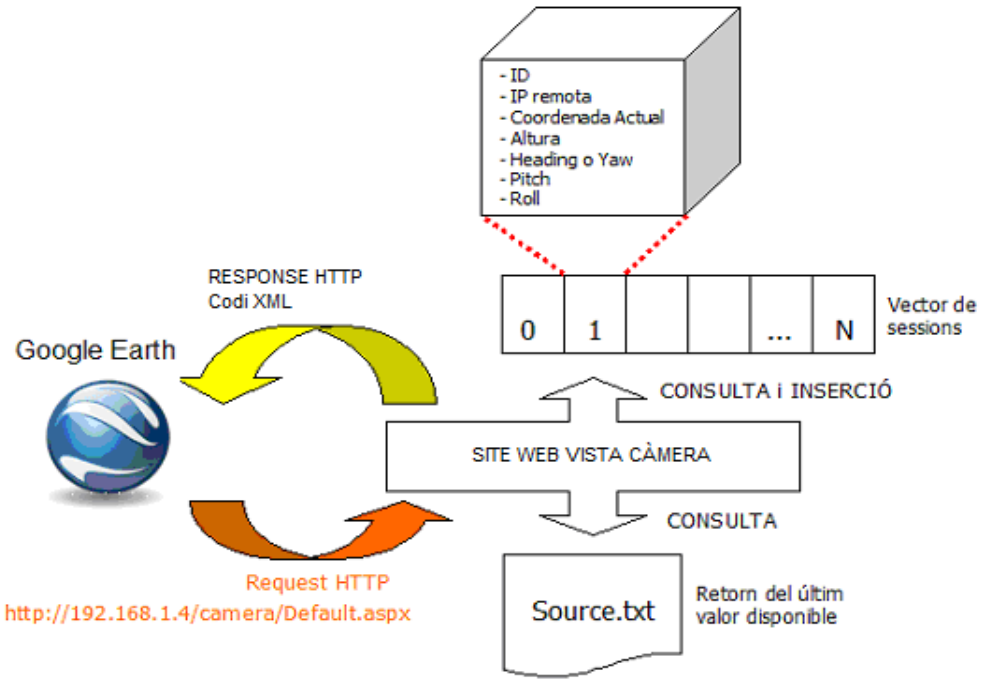


Figura 21 : *Diagrama del funcionament dels aplicatius de Càmera.*

4.1.5 Càmera primera persona

Aquesta nova funcionalitat complementa la vista anterior. Ambdues són aplicacions molt semblants en quant a funcionament, la diferència entre les dues radica en la perspectiva. L'anterior vista ens dóna una perspectiva a vista d'ocell de la trajectòria de l'aparell, tot fent un seguiment. En canvi aquesta ens dóna la perspectiva del que podríem veure si estiguéssim a la cabina del pilot. Per això l'anomenem "vista en primera persona". La panoràmica que aconseguim resulta molt vistosa. Per aconseguir la perspectiva que volem tant en la Càmera seguiment com en la de primera persona hem de configurar correctament el punt de visió a la coordenada en què es troba l'avió en aquell moment. La diferència entre les dues càmeres està en el *tilt* o inclinació. Per la càmera seguiment, l'inclinació de la càmera seria vertical enfocant cap a la superfície. Veuríem el punt a vista d'ocell i els valors variarien depenent del *tilt* de l'avió en aquell moment (Proporcionats pel simulador). Per aconseguir la Càmera en primera persona sumem 90 graus al *tilt* de l'avió per aconseguir l'efecte de visió cap a l'horitzó (vista perpendicular a la càmera seguiment). Aquesta càmera en primera persona ens és útil ja que el PFC està enfocat a naus no tripulades, d'aquesta manera podem tenir una visió similar a la que tindria un pilot embarcat a l'aeronau.

D'altra banda les variables *Heading* i *Roll* també són presents en la configuració de la vista Càmera (tant la primera com la segona). En quant al *Heading*, ens indicarà la direcció des d'on es mirarà el punt (orientació Nord, Sud, Est , Oest i totes les seves variants expressades en graus). Gràcies al Roll podrem veure el resultat de la vista quan l'inclinació de les ales varia d'un estat estable, especialment quan es produeixen girs en la trajectòria.

4.1.6 Diagrama

A continuació trobem una figura que representa el funcionament global de les tres aplicacions web explicades fins ara (punts **4.1.3**, **4.1.4** i **4.1.5**). El què varia es cada aplicació és el codi KML generat finalment, ja que cada aplicació dóna a Google Earth instruccions diferents (caixa de color groc). Però el procés que es duu a terme abans de generar el codi KML és molt semblant.

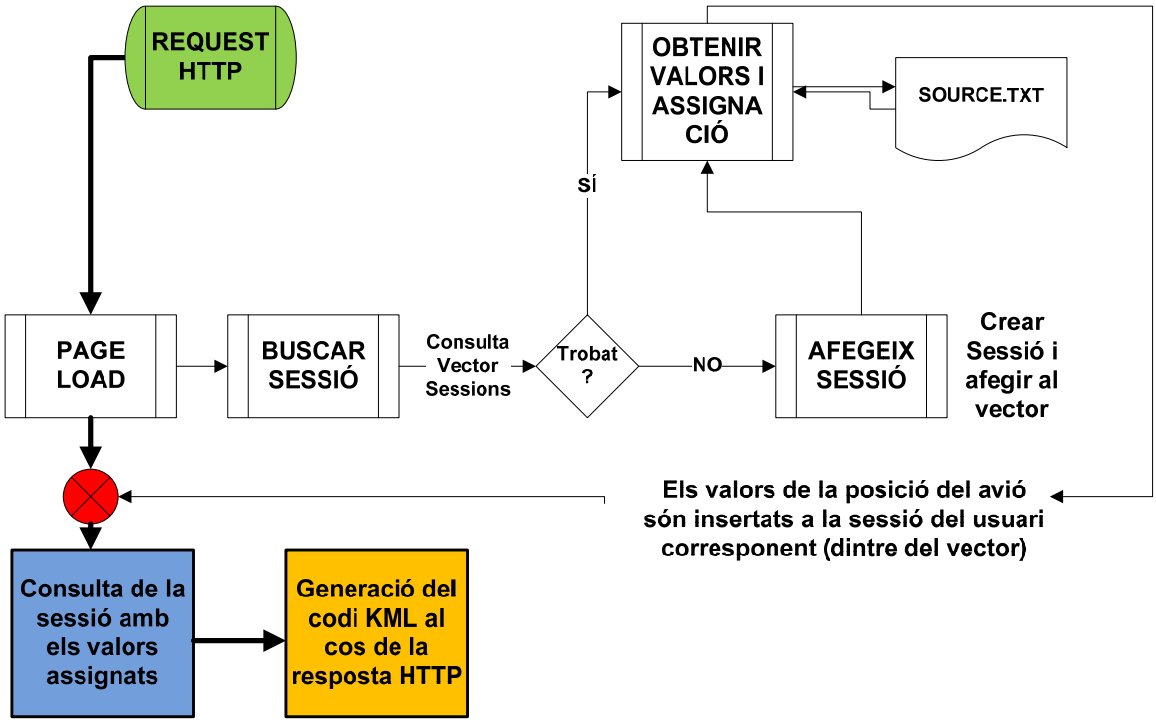


Figura 22 : *Diagrama del funcionament de les aplicacions web*

4.1.7 Pàgina web de Missions

És una aplicació senzilla però crec que complementa molt aquest projecte. És un mòdul totalment diferent a la resta , que no entra en joc alhora de fer la monitorització de la missió. És molt útil ja que ens permet descarregar-nos via web totes les missions realitzades fins al moment , disponibles a la Base de dades. L'element principal d'aquesta pàgina web és un contenidor d'informació (*Datagrid*) connectat a la base de dades i on estan representats tots els *items* que conté aquesta. Ens permet cercar i escollir quina és la missió que volem descarregar-nos. El seu funcionament és molt senzill, Mitjançant el *Datagrid,* se'ns mostren totes les missions disponibles. Sel·leccionant la fila del DataGrid que conté la missió que necessitem, i *Clickant* en un *link* de la part inferior, ens podem descarregar l'arxiu KML que conté la ruta que va seguir l'avió durant la missió.

5 Resultats

En aquest apartat de la memòria representarem de forma gràfica els resultats d'aquest projecte per cada mòdul del mateix.

5.1 Programa Principal

Aquest és l'aspecte que presenta l'aplicació principal. Podem veure el progrés de les gràfiques de velocitat (vermell) i altura (blau). A la dreta podem veure els botons de començament de missió (verd) i el botó per finalitzar missió (vermell). Sota d'aquests botons podem veure el menú encarregat de la sel·lecció de l'adreça IP de la màquina.

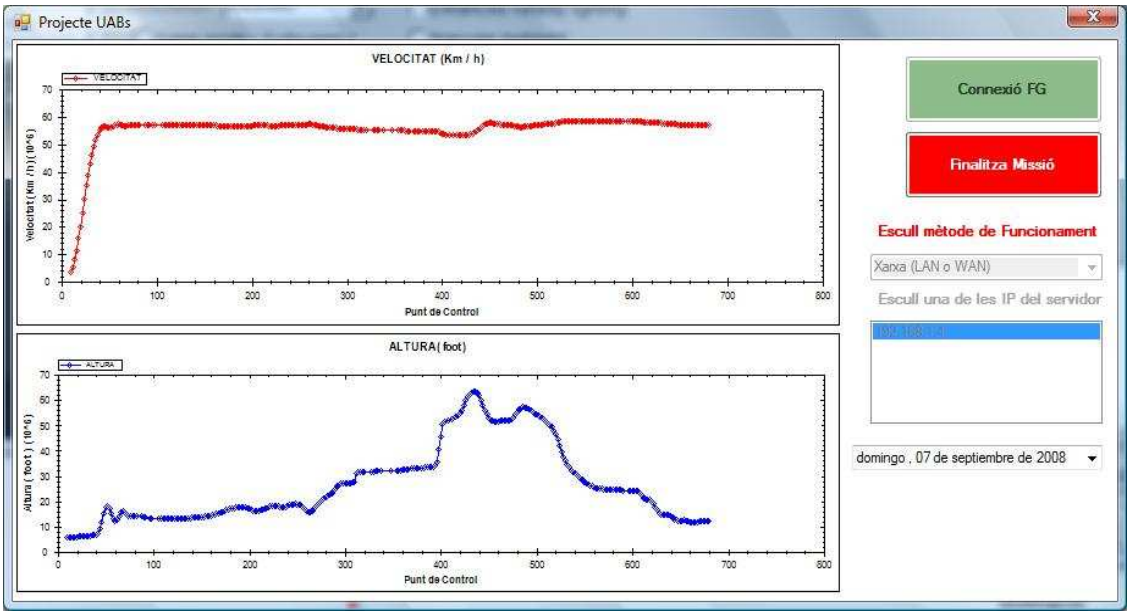


Figura 23. *Vista del programa principal realitzant una missió*

5.2 Posició i Trajectòria del aparell

En les següents imatges podrem veure el resultat d'una monitorització en procés. A la figura 24 , el fons correspon a l'aplicació Google Earth, on veiem la ruta i posició de l'avió, la finestra de la part dreta de la figura correspon al programa Flight Gear. A la figura 25 podem observar com també hi ha presents les gràfiques situades a la pantalla del Google Earth.

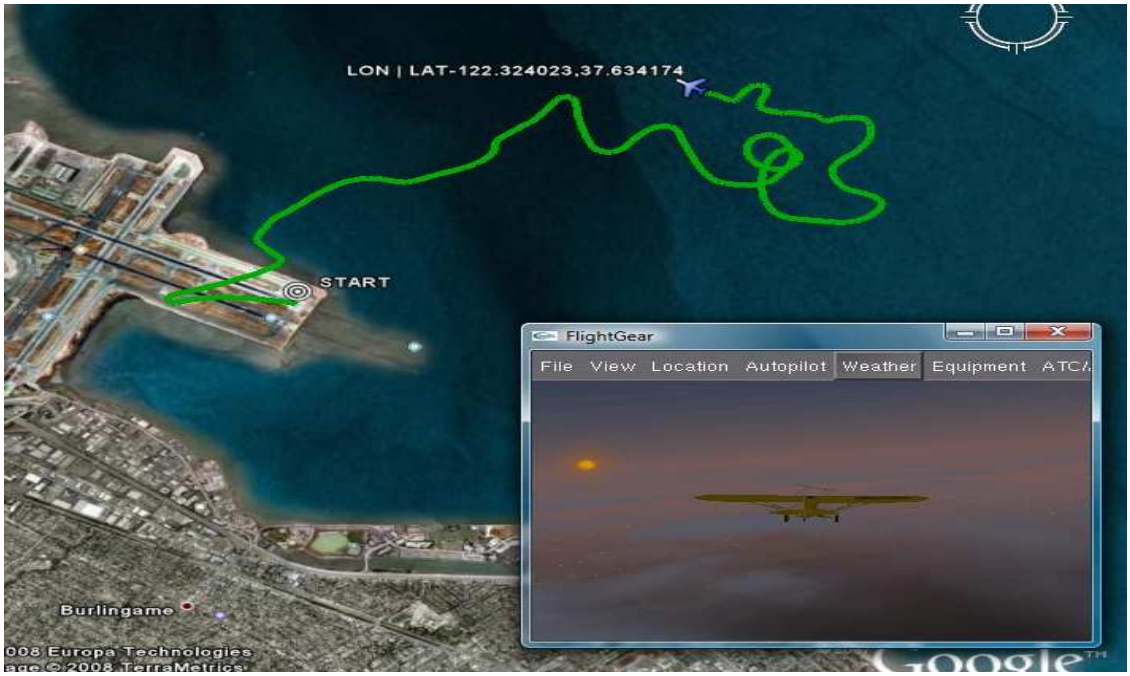


Figura 24. *Monitorització de la posició actual del avió més la ruta seguida fins al moment. Podem observar que el punt de partida també està indicat, aquest pertany a una de les pistes de l'aeroport internacional de San Francisco, als EEUU.*

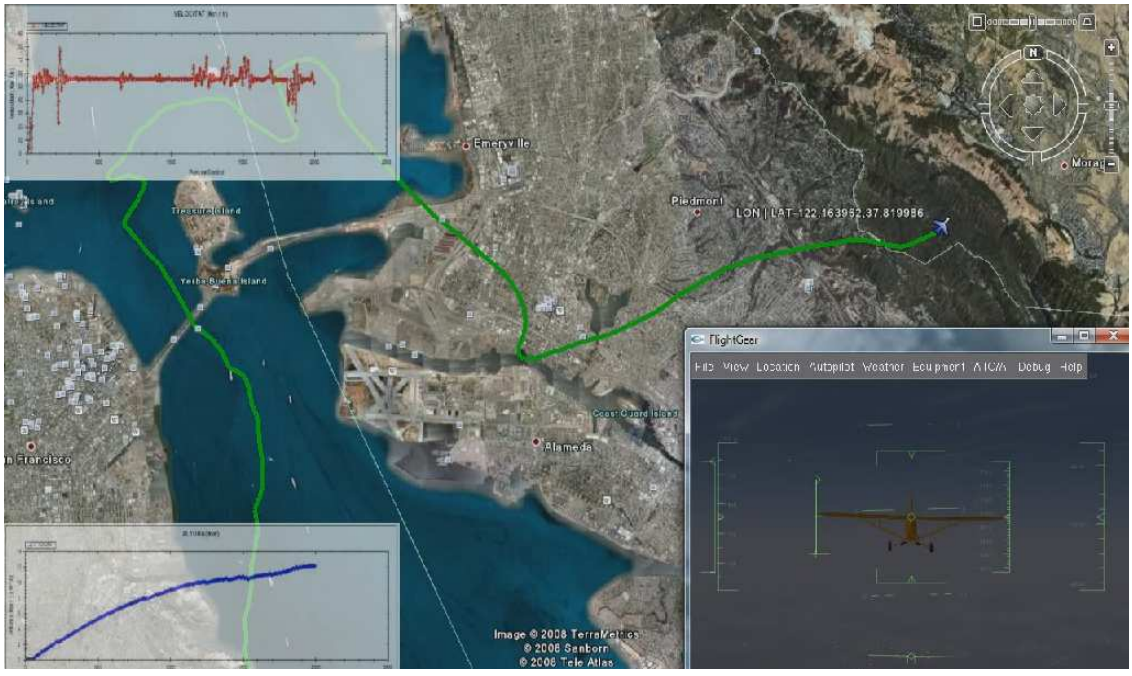


Figura 25. *Monitorització de la posició actual del avió més la ruta seguida fins al moment. En aquesta figura apareixen també les gràfiques. En aquesta missió s'està sobrevolant l'espai aeri de San Francisco i San Lorenzo, als EEUU.*

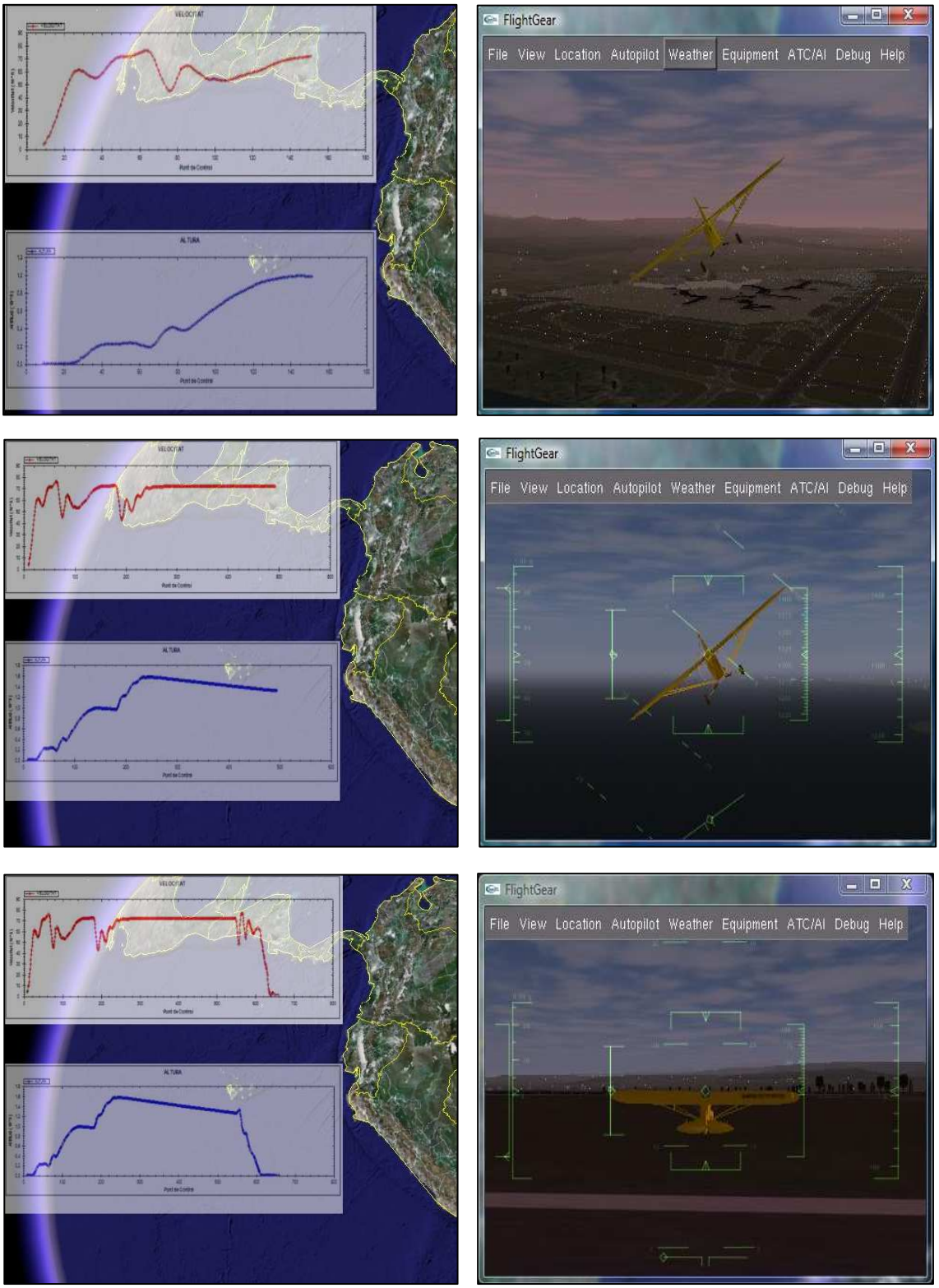
5.3 Direcció

En cada actualització que fem de la posició i trajectòria de l'aparell en vol, també s'indica la direcció *Heading* del avió en aquell moment. Ho indiquem mitjançant la rotació de la icona en forma d'avió utilitzada. D'altra banda, també es mostren els valors de les coordenades (Longitud i Latitud).

Figures 26, 27 i 28. *Direcció del Avió*

5.4 Gràfiques de Velocitat i d'Altitud

En les següents imatges veurem una seqüència d'una missió realitzada, des de l'enlairament fins l'aterratge. Aquesta seqüència es centra en les gràfiques. Podem veure com van canviant les gràfiques a mesura que la missió avança. En l'eix X de totes dues gràfiques s'especifica el punt de control (representa el temps), a mesura que aquest avança podem veure com es reconfigura l'escala en aquest eix. Passa al mateix en l'eix Y, encara que aquest es reconfigura depenent dels valor d'altura i velocitat depenent la gràfica.



Figures 29, 30, 31, 32, 33 i 34. *Detall de les gràfiques de Velocitat i Altitud. Podem observar l'evolució que presenten les gràfiques allarg de la missió, des de el seu enlairament fins l'Aterratge.*

5.5 Càmera Seguiment

La Càmera de seguiment cerca , en tot moment, la situació de l'avió en pantalla i ens reprodueix una vista basada en l'altitud de l'avió. També té en compte l'inclinació de l'aparell.

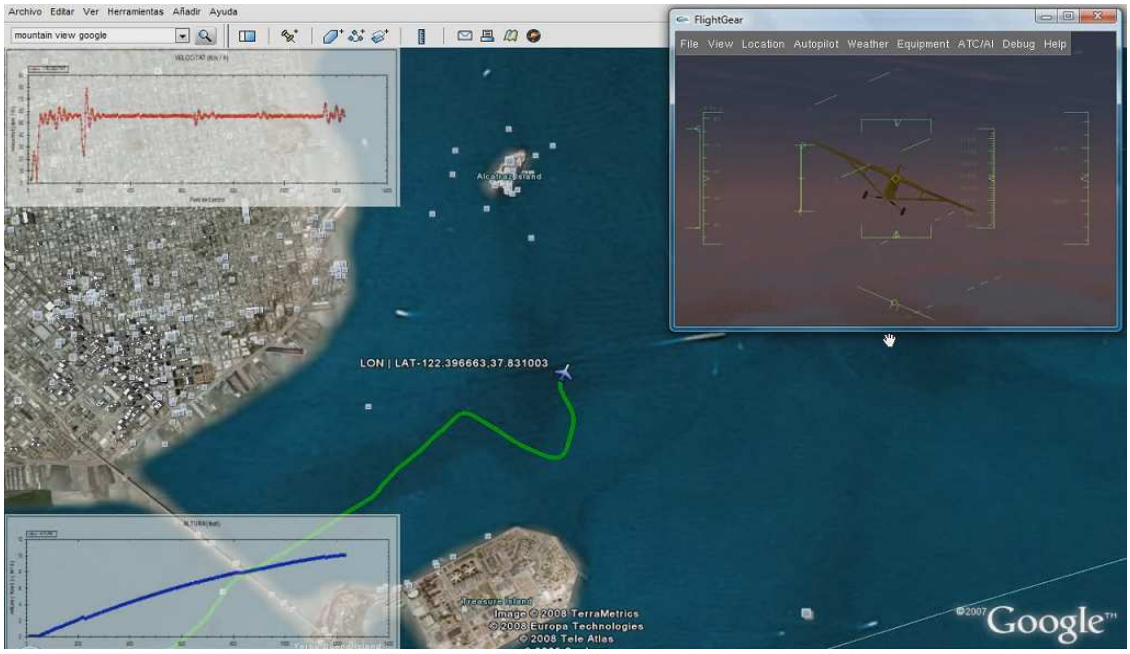


Figura 35. *Càmera de seguiment*

5.6 Càmera en Primera persona

Un possible *handicap* d'aquesta funcionalitat del projecte es presenta alhora de configurar el refresc en el qual Google Earth demana una actualització de càmera. En cada petició, la visió dels mapes de Google Earth "vola" fins a la coordenada indicada en l'arxiu kml. Si es produeix un refresc mentre la visió del programa encara s'està traslladant al punt anterior es crea un efecte de talls, contràriament a un efecte de visió continua i fluida presentant-nos un efecte de velocitat. És molt difícil pal·liar completament aquest efecte però configurant correctament el refresc de pantalla de l'enllaç de xarxa podem conservar una vista fluïda. S'ha planificat per defecte un refresc de 3 segons per donar-li temps a l'aplicació de produir cada efecte de translació i no interrompre'l en el seu afer. Cal dir però, que l'efecte de velocitat que s'aconsegueix no és gaire alt però és molt similar al que es produeix en el simulador, per tant això ens indica la concordança entre l'abast de coordenades que es produeix entre la font d'informació i la monitorització d'aquesta.

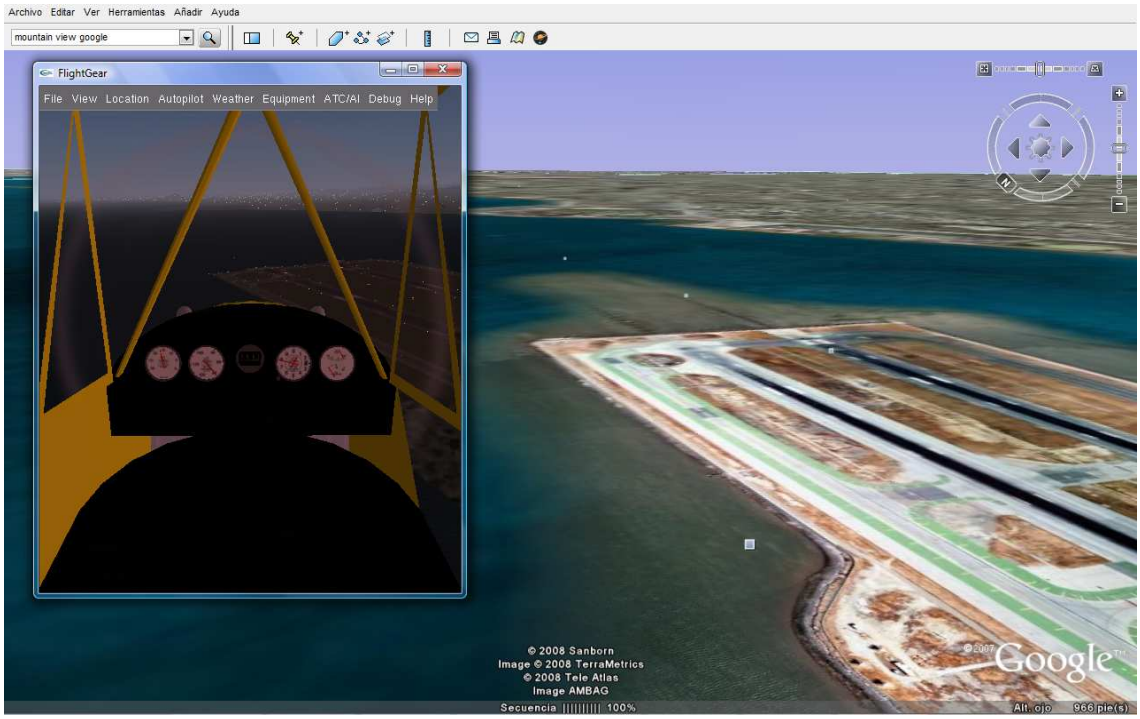


Figura 36 . *La Càmera en primera persona ens mostra els mapes de Google Earth de la mateixa manera que els veuríem si fóssim el pilot de l'avió. La configuració de la velocitat del refresc de les vistes (en el menú de configuració de Google Earth) són critiques per obtenir bons resultats.*

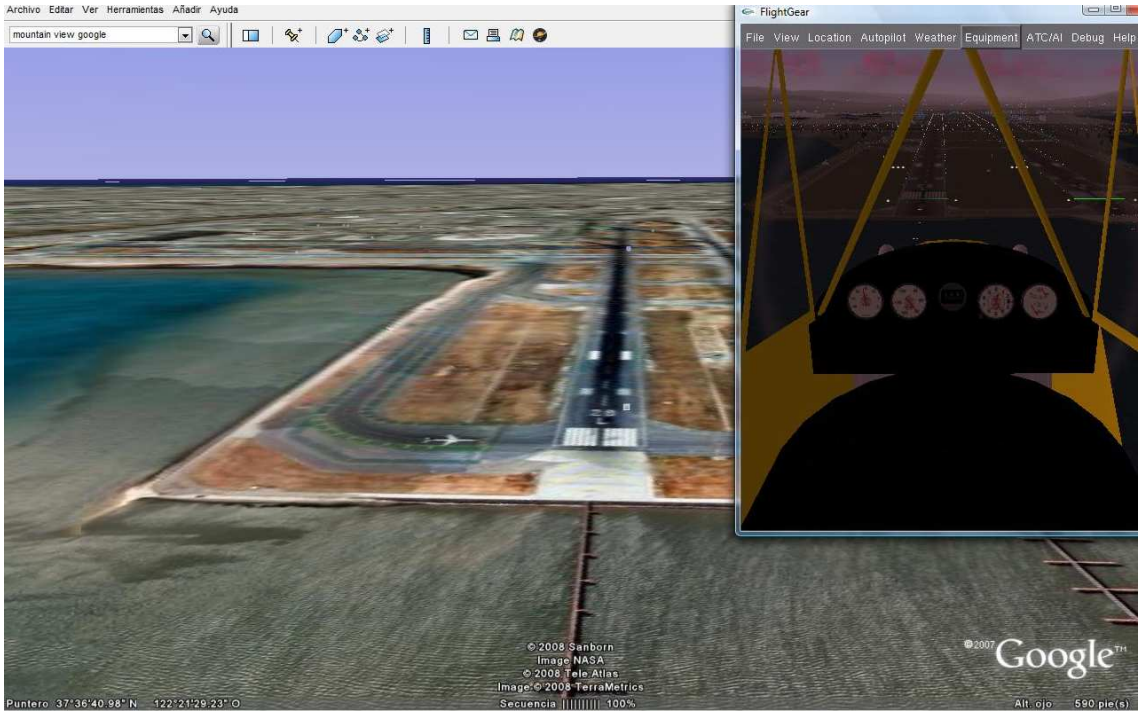


Figura 37 . *Podem veure com la vista en primera persona del simulador es correspon amb la visió que obtenim al Google Earth.*

5.7 Arxius d'històric de les missions

Aquests arxius es generen un cop acabada la missió i el seu objectiu és deixar constància del recorregut i l'alçada que va seguir l'aparell en tot moment. Ens serà útil per poder analitzar les missions (en tres dimensions) a posteriori.

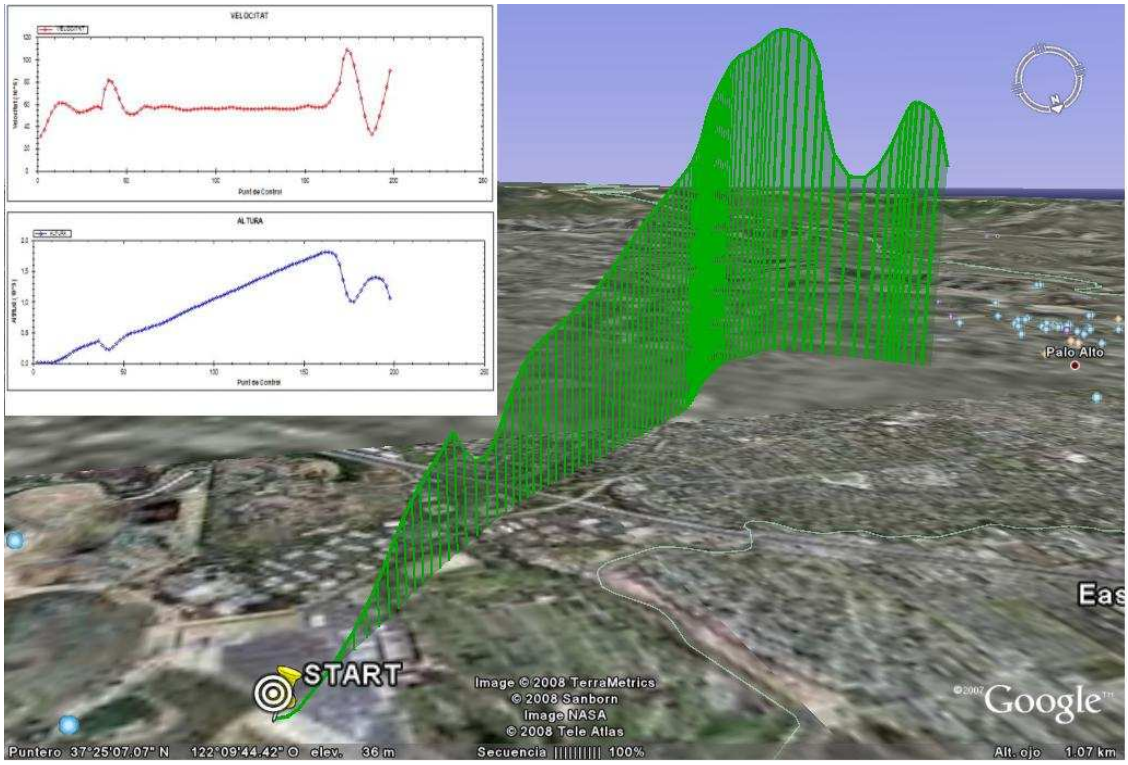


Figura 38. Arxius d'històric de missions

5.8 Pàgina web de Missions

Mitjançant aquesta interfície web podrem descarregar-nos via Internet o xarxa local els arxius d'històric de totes les missions realitzades fins aquell moment. Alhora que es generen els arxius d'històric, també s'afegeix a la Base de Dades una entrada que li fa referència per poder-lo recuperar després. Aquest Site Web connecta amb la base de dades per mitjà d'un contenidor de dades *DataGrid*. Mitjançant el *link* de la part inferior i prèvia sel·lecció de l'arxiu es produeix la descàrrega.

Figura 39. Pàgina web de missions

6 Ambientalització

Acotant com a plataforma de monitorització, l'impacte en el medi d'aquest projecte no seria molt representatiu. Crec que en aquest apartat hauríem de tractar no només el fet de la monitorització, si no les missions civils UAV en conjunt.

En el punt **1.1.3** d'aquesta memòria, es deixa palesa de la vessant en la qual aquest projecte pretén ser útil, les aplicacions UAV d'ús civil. Crec que en aquest punt l'impacte mediambiental és positiu. L' utilització d'aquests aparells en l'ajuda d'extinció d'incendis és una característica favorable per a la conservació dels espais naturals. Tanmateix, els usos com a eina de mesura de paràmetres contaminants ajudarien a combatre la contaminació. També ens permetrien conèixer més a fons les característiques del nostre entorn, entre d'altres usos mediambientals que es podrien aplicar. Crec que l'utilització d'aquestes naus amb aquests fins són un benefici per a la comunitat i l'entorn.

En el món de l'aeronàutica actual (incloent el desenvolupament d'aplicacions UAV), s'està prioritzant el disseny de naus de baix consum i baixes emissions contaminants [11]. S'ha de tenir en compte que l'aviació en general contribueix a la contaminació per emissions de CO_2 en un 2% del total [11]. Per tant, tenint en compte l'ús molt menor d'UAV enfront de l'aviació comercial i militar, i també tenint en compte la diferència elevada de prestacions entre uns i altres, podríem concloure amb un impacte contaminant en el medi menor.

7 Conclusions

Com a conclusions, caldria indicar que s'han assolit les funcionalitats descrites en el punt **1.4** d'aquesta memòria. En l'apartat de resultats s'indica detalladament cada part per separat. El comportament de la plataforma en conjunt ha sigut correcte en la fase de proves que s'ha realitzat. Aquestes proves s'han realitzat en un entorn local amb més d'un usuari simultàniament.

Els resultats obtinguts han complert les expectatives de desenvolupament d'aplicacions amb la utilització de Google Earth. És a dir, s'ha aconseguit utilitzar un sistema d'informació geogràfica ja desenvolupat, com a marc on representar unes dades útils per a la monitorització de missions UAV. Caldria indicar que hem trobat altres projectes que utilitzen Google Earth de manera similar, un exemple seria el **Real Time Mission Monitor** [12].

En quant a tecnologies utilitzades, tant el comportament del servidor IIS7 com de la plataforma ASP.NET ha sigut molt correcte. La plataforma per a desenvolupar ASP.NET ens ha proporcionat el marc que necessitàvem així com les eines necessàries per l' implementació d'aquest projecte.

Gràcies a l'apartat de *Posta en marxa de la plataforma,* als *Annexos,* es podria posar en funcionament aquesta plataforma per transportar-la a un escenari real de monitorització. Caldria primer una aplicació UAV real i que les dades de posicionament d'aquest fossin enviades periòdicament a la nostra plataforma, d'una manera semblant a com les envia el simulador utilitzat.

En quant a possibles ampliacions d'aquest projecte (podrien ser molt variades) jo destacaria una funcionalitat descrita a l'apartat **1.1.3** d'aquesta memòria. Seria molt interessant poder utilitzar Google Earth per a monitoritzar dades recollides per sensors ubicats a l'UAV. D'aquesta manera, a més de saber la posició de l'avió en tot moment, podríem saber com s'està desenvolupant la missió. Aquesta informació es podria disseminar a través d'Internet per exemple, i fins i tot podria ser útil en la presa de decisions.

8 Bibliografia

[1] Site oficial de la Universidad Nacional Mayor de San Marcos. Informe sobre vehicles aèris no tripulats
http://www.unmsm.edu.pe/Destacados/contenido.php?mver=19

[2] Wikipedia. Plana d'Usos i Característiques UAV
http://es.wikipedia.org/wiki/UAV

[3] Wikipedia. Plana d'Aerodinàmica en el món dels avions
http://en.wikipedia.org/wiki/Flight_dynamics

[4] Site About.com : Inventors. Característiques l'aerodinàmica dels avions i del pilotatge aeri
http://inventors.about.com/library/inventors/blairplanedynamics.htm

[5] Cameron Laird, "Lightweight Web Servers" .IBM web site. 10 Juliol 2007
http://www.ibm.com/developerworks/web/library/wa-ltwebserv/

[6] Wikipedia. Plana Comparativa de servidors web
http://en.wikipedia.org/wiki/Tiny_web_servers

[7] Site Tldp.org Tutorial sobre programació de Sockets
http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html

[8] KML Documentation
http://code.google.com/apis/kml/documentation/index.html

[9] Site web oficial d'ASP
http://www.asp.net/

[10] Wikipedia Plana d'especificació dels sistemes SIG
http://es.wikipedia.org/wiki/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica

[11] Edició digital d'Aeronautica Andaluza. *Número 5 Octubre-Desembre 07*
http://www.aeropolis.es/opencms/export/sites/aeropolis/es/modules/revistas/adjuntos/AERONAUTICA-05.pdf

[12] Site oficial del projecte Real Time Mission Monitor de la NASA
http://rtmm.nsstc.nasa.gov/

[13] Site oficial de ZedGraph
http://zedgraph.org/wiki/index.php?title=Main_Page

[14] Site Oficial de Mono
http://www.mono-project.com/ASP.NET

Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEXOS

TÍTOL DEL TFC/PFC: Utilització de Google Earth per al seguiment remot d'un UAV

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Marçal de las Heras Mandome

DIRECTOR: Eduard Santamaria Barnadàs

DATA: 22 d'Octubre de 2008

Posta en marxa de la plataforma

Aquesta part del projecte s'encarregarà de donar l'informació necessària per a poder utilitzar aquesta plataforma de manera adequada, disposarem aquesta informació diferenciant pels mòduls més importants. El projecte ha estat desenvolupat i provat sobre el sistema Operatiu Windows Vista de Microsoft.

Instal·lació de IIS 7

IIS 7 és un mòdul de Windows, per tant ha de ser instal·lat com a una característica d'aquest. El contingut de IIS7 està al disc d'instal·lació de Windows, si ja tenim el Sistema Operatiu disponible a la màquina, per instal·lar IIS haurem d'accedir a:

Panel de Control > Programas sel·leccionem Activar o desactivar las características de Windows.

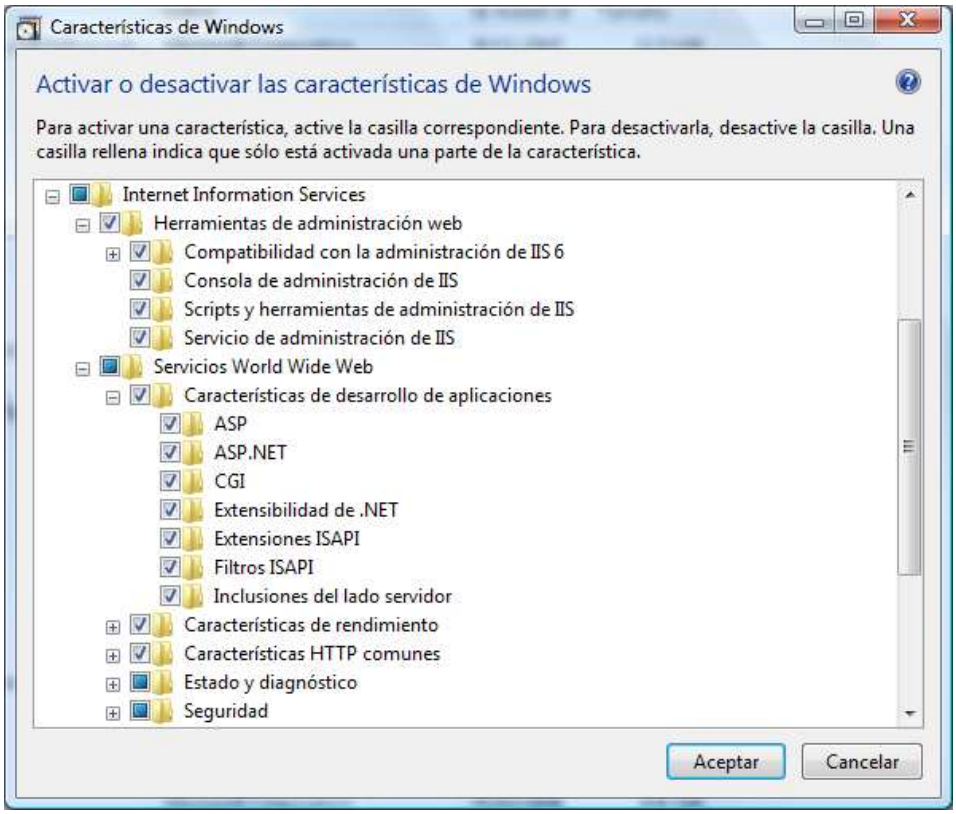


***Figura 42**. Menú d'instal·lació d'IIS7*

Ens trobarem el menú anterior, cal sel·leccionar les caselles que ens interessen, a la figura trobem un exemple amb la configuració utilitzada per al desenvolupament del projecte.

Elements necessaris a la carpeta *Wwwroot*

Un cop instal·lat IIS 7 se'ns crearà automàticament un directori que serà imprescindible per al desenvolupament del projecte. El directori és "C:\inetpub\wwwroot" i serà la carpeta arrel del nostre nou servidor web. Es proveirà al usuari de certs fitxers i arxius necessaris per que la plataforma es pugui posar en funcionament. Tots els arxius auxiliars que utilitza aquesta plataforma estan recollits en aquest directori per poder tenir la informació centralitzada en una mateixa ubicació. Per evitar problemes de seguretat i intrusió IIS 7 té filtrats automàtics per a extensions de base de dades (*mdb* per exemple) , també podem configurar paràmetres de seguretat pels altres arxius.

Arxius necessaris haurem de moure a aquest directori un cop creat

Els arxius "Grafiques.jpg" i "Grafiques2.jpg" seran proporcionats a l'usuari i hauran de ser copiats a la carpeta arrel del Servidor web. La seva funció és la de proporcionar al programa principal una tela (mapa de bits en blanc) per tenir una referència alhora de dibuixar les gràfiques. Els arxius d'imatge amb les gràfiques dibuixades seran guardats en uns altres arxius diferents, també en aquesta carpeta; s'actualitzaran periòdicament en cada "cicle" del programa principal.
Cada aplicació web consta d'uns arxius executables que estan recollits en un fitxer. A l'usuari se li proporcionaran quatre fitxers (un per aplicació) i hauran de ser copiats a la carpeta arrel. Els fitxers són: WebSite16 (mostreig ruta), WebSite18(Càmera seguiment), WebSite19 (Càmera en primera persona) i Webmissions. D'altra banda la base de dades "Missions.mdb" s'haurà de copiar en la següent ubicació:
C:\inetpub\wwwroot\Missions
Haurem de crear, per tant, el fitxer "Missions" dintre de la carpeta arrel. Tots els arxius d'històric de missions s'aniran guardant aquí.

Arxius que es crearan automàticament

"GraficaAltura.jpg" i "GraficaVelocitat.jpg" seran generats automàticament tal i com s'indica en l'apartat anterior. "Primerpunt.kml" serà l'arxiu KML que representarà el punt de partida del avió en la missió. Serà l'arxiu de referència per a les actualitzacions que es duran a terme en el procés de mostreig de la ruta del avió. "Source.txt" serà l'arxiu d' on treuran l' informació les aplicacions web, es crearà automàticament i serà actualitzat periòdicament. "Ipserv.txt" és l'arxiu on quedarà constància de l'adreça IP del servidor que sel·lecciona l'usuari en l'aplicació principal.

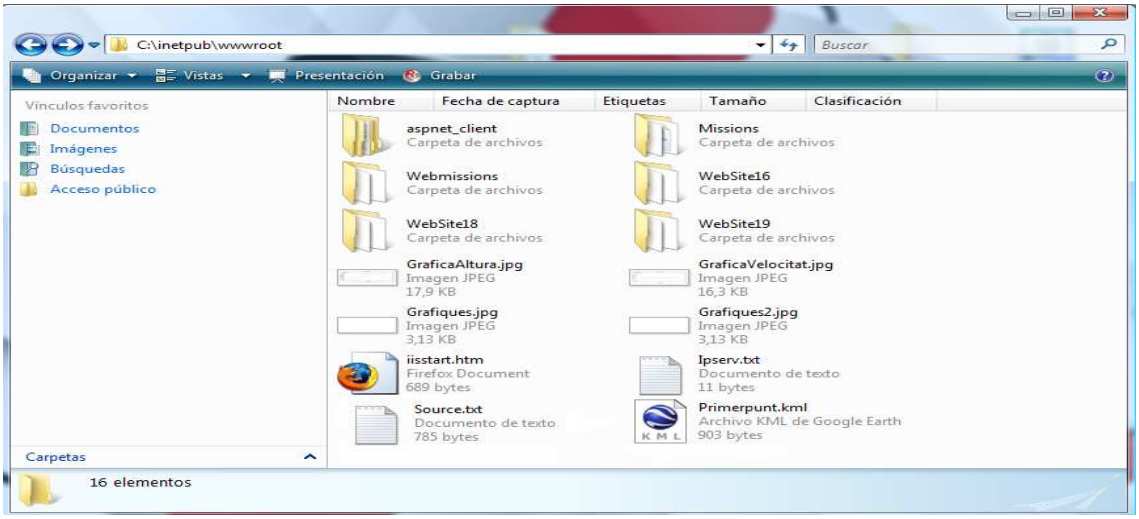


Figura 43. *Contingut de la carpeta arrel del servidor web*

Aplicació principal

L'aplicació principal es proporcionarà a l'usuari i només caldrà que l'executi quan vulgui començar a monitoritzar una missió. L'única condició especial és dóna a alhora de treballar amb la base de dades de les missions. Si ubiquem la base de dades (que també serà proporcionada) tal i com s'indica en l'apartat anterior no hi haurà problemes. Si volem escollir una altra ubicació (per motius de seguretat no cal) haurem de modificar els components d'origen de dades del programa principal. Haurem d'obrir l'arxiu projecte de la aplicació amb el *Visual Studio C#* i haurem de fer alguns retocs. Haurem d'accedir al menú *"Datos-Agregar nuevo origen de datos"* i seguir el *wizard* per ubicar en un altre camí la nostra base de dades. Els components *DataSet* i *Table Adapter* se'ns generaran automàticament i haurem de crear manualment la consulta SQL INSERT que es produeix quan tanquem l'aplicació. Tot això es realitzarà amb un entorn de finestres molt intuïtiu.

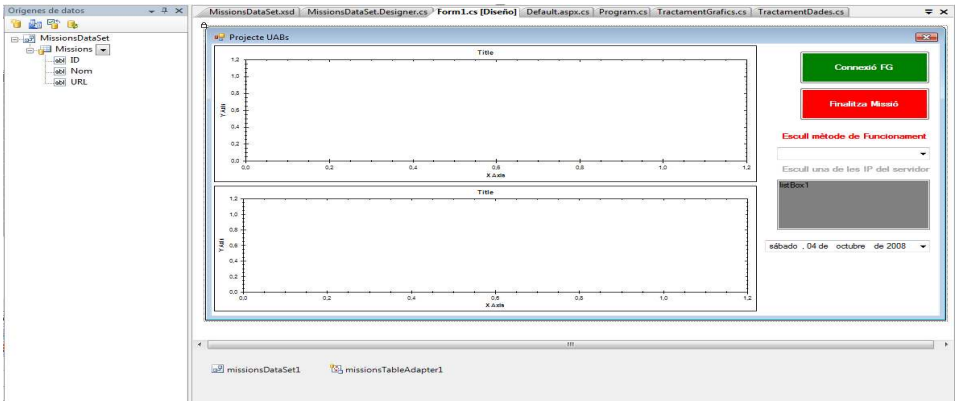


Figura 44*. Vista disseny de la programació de l'aplicació principal. A la part inferior trobem els components d'enllaç amb la BBDD, a l'esquerra trobem la pestanya d'origen de dades*

Webmissions

La seva connexió amb la base de dades és diferent a la de l'aplicació principal. En aquest cas s'utilitza un component de connexió *SqlDataSource*. Per crear aquesta connexió amb la base de dades haurem de crear un origen de dades ODBC (*Panel de control\Herramientas administrativas\Orígenes de datos ODBC*) tal i com es mostra en la figura:

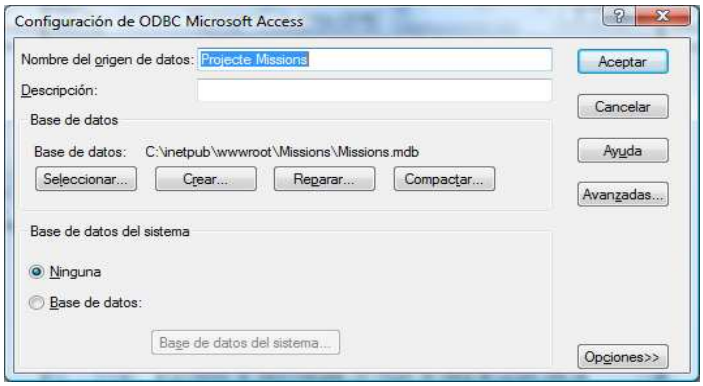


***Figura 45**. Finestra de configuració de l'origen de les dades ODBC*

També haurem de configurar la connexió de l'aplicació web amb la base de dades en la consola d'administració d'IIS7 (*Panel de control\Herramientas administrativas\Administrador de Internet Information Services*), s'haurà de sel·leccionar l'aplicació que volem (a l'esquerra de la figura següent) i relacionar-la amb l'origen ODBC creat anteriorment .

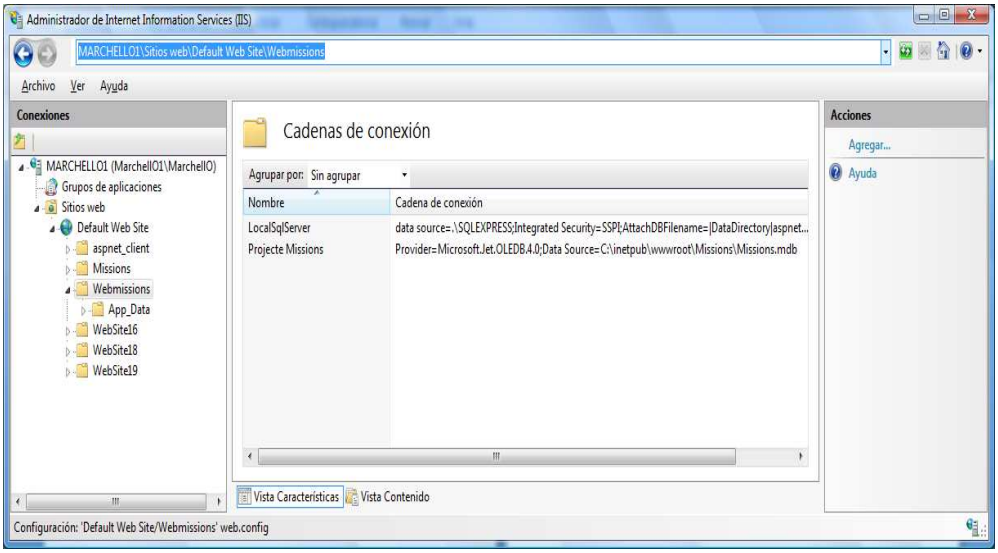


***Figura 46.** Finestra de configuració d'IIS7*

Podem veure en la figura anterior que se'ns genera la cadena de connexió següent:

Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\inetpub\wwwroot\Missions\Missions.mdb

Aquesta cadena de connexió s'haurà d'introduir en el camp adequat del component *SqlDataSource* de la aplicació web:

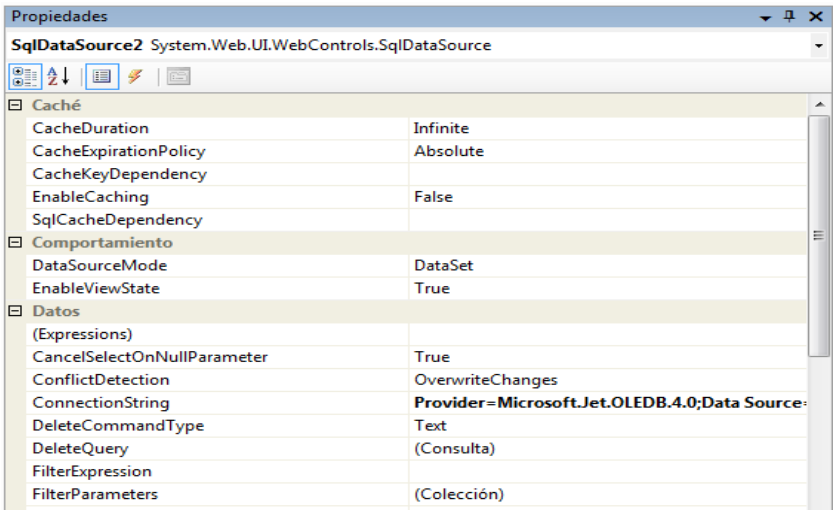


***Figura 47**. Configuració de l'Objecte SqlDataSource*

Configuració de Google Earth

Els components que haurem de crear amb el Google Earth són quatre enllaços de xarxa (NetworkLinks) i dos sobrexposicions d'imatge (ScreenOverlay), aquests seran proporcionats al usuari en uns fitxers KML, que només s'hauran d'executar i moure'ls al directori *Mis Lugares* si no volem haver d'executar-los en cada missió ja que per defecte es situaran a la carpeta *Lugares Temporales*. La figura següent n'és un exemple.

***Figura 48.** Carpeta de contingut actiu de Google Earth A la imatge dreta trobem el contingut que genera l'aplicació web alhora de dibuixar la ruta i la posició del avió*

A continuació veurem la configuració dels dos *NetworkLinks* per al seguiment de ruta, *Heading* i posició actual del avió:

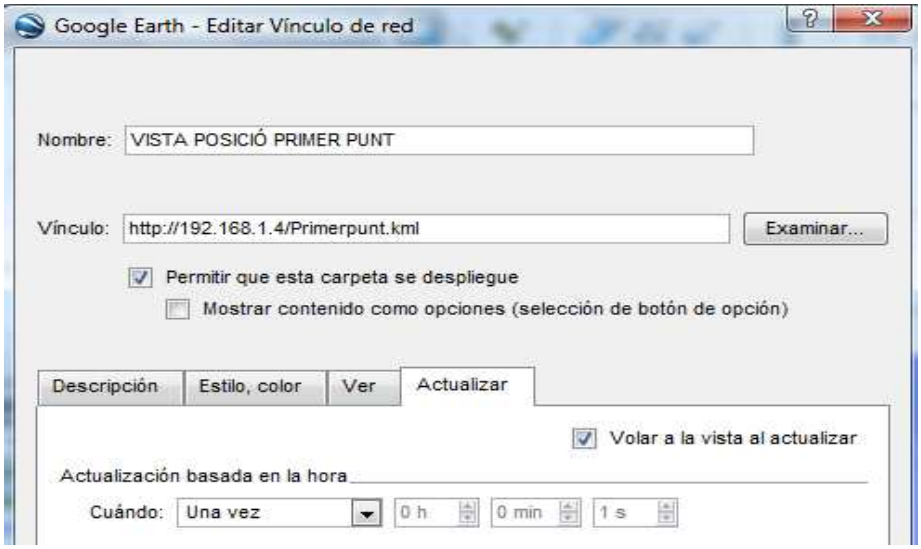


Figura 49. *Primer NetworkLink, connecta amb el punt de partida de la missió*

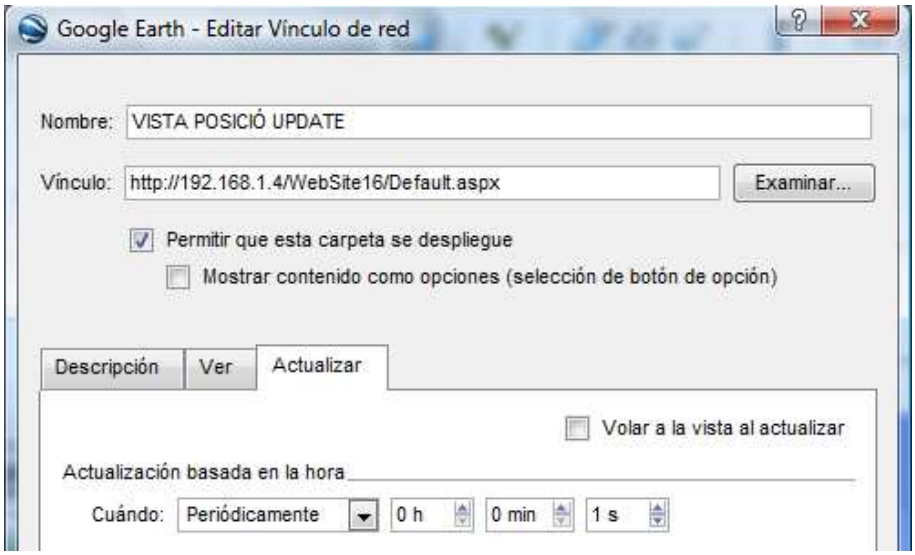


Figura 50*. Segon NetworkLink, actualitza periòdicament la posició del avió i va dibuixant la trajectòria seguida. Aquesta aplicació web està ubicada en un fitxer anomenat WebSite16.*

Configuració del NetworkLink per a la Càmera Seguiment:

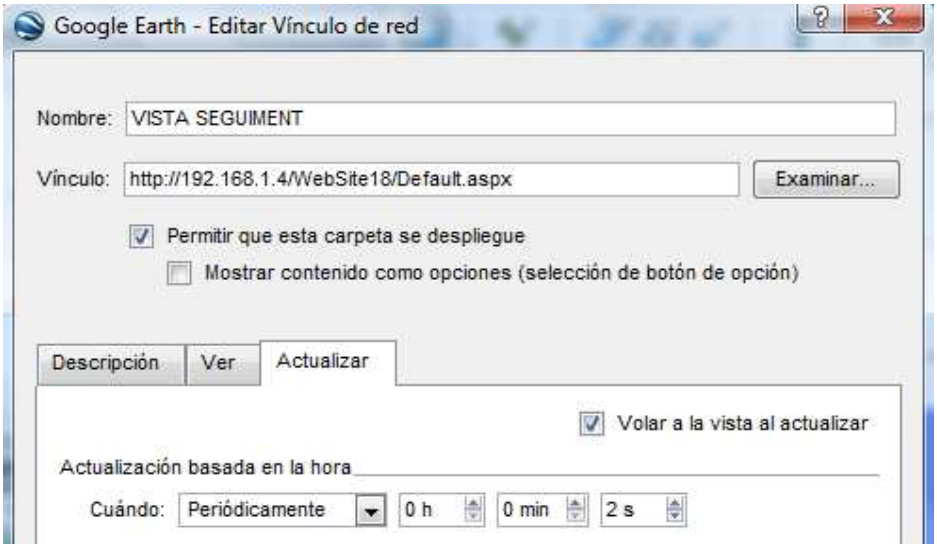


***Figura 51**. NetworkLink, actualitza periòdicament la vista de seguiment que se li fa al aparell. Aquesta aplicació web està ubicada en un fitxer anomenat WebSite18.*

Configuració del NetworkLink Càmera en primera persona:

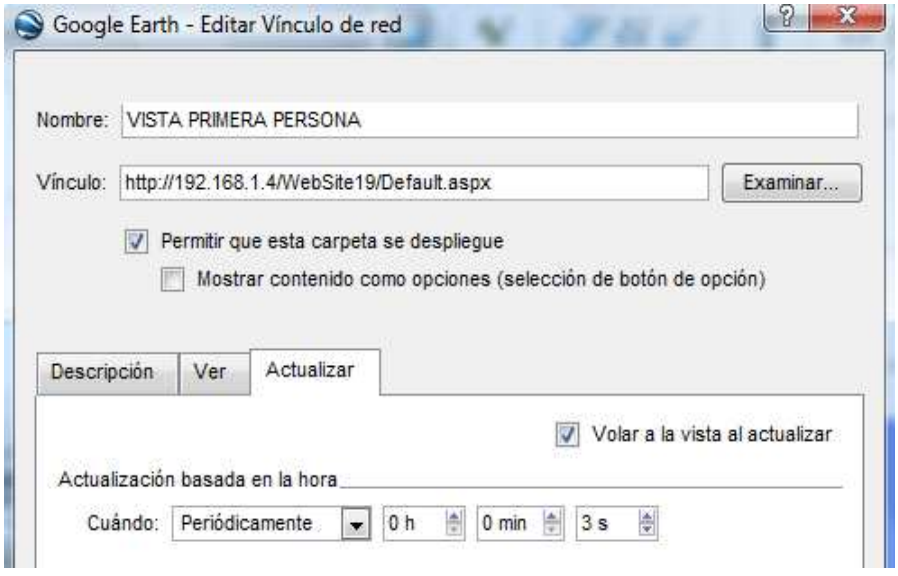


***Figura 52**. NetworkLink, actualitza periòdicament la vista en primera persona del aparell Aquesta aplicació web està ubicada en un fitxer anomenat WebSite19.*