
# TREBALL DE FI DE CARRERA

**TÍTOL DEL TFC: Disseny d'un simulador d'aviònica per a una cabina de DC9/A320 basat en Flight Simulator**

**TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació**

**AUTORS: Bernat Font Garcia**

**DIRECTORS: Oscar Casas Piedrafita, Joshua Tristancho Martínez**

**DATA: 25 de Juliol del 2012**

**Títol:** Disseny d'un simulador d'aviònica per a una cabina de DC9/A320 basat en Flight Simulator
**Autors:** Bernat Font Garcia
**Director:** Oscar Casas Piedrafita, Joshua Tristancho Martínez
**Data:** 25 de Juliol del 2012

**Resum**

Aquest projecte tracta sobre el desenvolupament d'un software que permeti enllaçar un simulador de vol i una cabina real de DC9, situada al Institut de la Illa dels Banyols al Prat de Llobregat - Barcelona.

Mitjançant una llibreria externa al simulador, s'estableix una comunicació entre aquest i el programa en qüestió. Per una banda s'aconsegueix extreure les dades que es necessiten mostrar en la cabina del DC9 i per l'altra es poden generar certes fallades en el simulador a mode de pràctica per als estudiants que utilitzin el software.

El programa ha estat desenvolupat mitjançant el llenguatge C# a través de la plataforma Microsoft Visual Studio 2010, versió 10.0.30319.1 RTMRel de ©2010 Microsoft Corporation.

**Paraules clau:** DC9, cabina virtual, simulador de vol, FSUIPC

**Title:** Disseny d'un simulador d'aviònica per a una cabina de DC9/A320 basat en Flight Simulator
**Authors:** Bernat Font Garcia
**Director:** Oscar Casas Piedrafita, Joshua Tristancho Martínez
**Date:** July, 25th 2012

**Overview**

This project is developing software capable of binding a flight simulator and a real DC9 cabin, located on the Island Institute of Banyols in Prat de Llobregat - Barcelona.

Using an external library, it establishes a communication between the simulator and the program. On one hand, it extracts the data needed to show in the cabin of the DC9. On the other hand, certain faults can be generated in the simulator as a practice for students using the software.

The program has been developed with C# language, through Microsoft Visual Studio 2010 platform, version 10.0.30319.1 RTMRel of ©2010 Microsoft Corporation.

**Keywords:** DC9, virtual cockpit, flight Simulator, FSUIPC

*M’agradaria donar les gràcies als meus dos directors del projecte: l’Oscar Casas i en Joshua Tristancho. En tot moment han tingut una actitud immillorable i s’han mostrat predisposats a ajudar-me a desenvolupar un bon projecte.*

*D’altra banda també m’agradaria agrair la tasca que ha fet en Sergio Mainar, el client del programa. Hem dedicat moltes hores a enfocar correctament el programa i refinar les seves funcionalitats per a obtenir el producte adequat.*

# ÍNDEX

## LIST OF FIGURES

# INTRODUCCIÓ

Els simuladors de vol són sistemes que intenten replicar el vol d'una aeronau de la manera més acurada possible. Actualment, hi ha diversos softwares que ofereixen aquesta possibilitat: Microsoft © Flight Simulator[1], el © XPlane[2] o el © Flight Gear[3], entre d'altres. Tots aquests programes ofereixen unes característiques semblants, però el que realment pot donar un notable potencial en aquest camp és el rendiment que en pot treure l'usuari.

Tradicionalment, els usuaris utilitzen aquests programes des del seu ordinador personal asseguts en una cadira de despatx, controlant la aeronau mitjançant el teclat i un joystick. Però, com ja s'ha implementat des de fa anys, aquests simuladors poden adaptar-se a entorns molt més adient com serien les cabines reals de l'avió que s'està simulant. I aquest és precisament l'objectiu del projecte.

Programant un mòdul extern al simulador, s'adaptaran les dades que aquest pot oferir a l'usuari per a poder transmetre-les a la aviònica d'una cabina de DC-9 i a l'inrevés. Així doncs, un persona externa (en aquest cas l'instructor) podrà modificar dades del simulador per a introduir fallades en els sistemes i que un alumne (el tècnic) provi de resoldre-les.

Amb tot això, en aquest projecte es desenvoluparà una aplicació que permeti a l'usuari visualitzar les dades de cabina del simulador facilitant d'aquesta manera la extracció de dades cap a la cabina real. D'altra banda l'aplicació també gestionarà aquestes dades de manera senzilla i eficaç.
Finalment, inclourà un mòdul de generació de dades per complir amb els requisits que s'han exposat.

A tall d'explicació, la memòria es dividirà en quatre apartats; el primer detalla els diferents panells i indicadors de la cabina del McDonnell Douglas DC-9, que es té en l'institut d'Illa dels Banyols, i en fa una comparació amb les cabines actuals, en aquest cas la del Airbus A320.
El segon capítol explica quin és el disseny de la arquitectura del sistema global que es vol desenvolupar i on hi té cabuda aquest projecte. El tercer capítol mostra el disseny dels mòduls que conté el programa i n'explica les funcionalitats. Per acabar, hi ha un quart capítol exposant les conclusions finals del projecte.

[1] www.microsoft.com/games/flightsimulatorx/

[2] www.x-plane.com/

[3] http://www.flightgear.org/

# ACRÒNIMS, ABREVIACIONS I DEFINICIONS

| | |
|---|---|
| A320 | Airbus A320 |
| ADI | Attitude Director Indicator |
| AP | Auto Pilot |
| DC9 | McDonnell Douglas DC-9 |
| ECAM | Electronic Centralized Aircraft Monitoring |
| EDP | Engine Driven Pump |
| EFIS | Electronic Flight Information System |
| EGT | Exhaust Gas Temperature |
| EPR | Engine Pressure Ratio |
| FCU | Flight Control Unit |
| FF | Fuel Flow |
| FMA | Flight Mode Annunciator |
| FS | Flight Simulator |
| FSUIPC | Llibreria utilitzada per comunicar el FS amb el programa |
| FSX | Flight Simulator X |
| GW | Ground Weight |
| HSI | Horizontal Situation Indicator |
| IAS | Indicated Air Speed |
| LDG | Landing Gear |
| MCDU | Multipurpose Control Display Unit |
| N1 | Etapa de baixa velocitat del motor |
| N2 | Etapa d'alta velocitat del motor |
| ND | Navigation Display |
| Offset | Variable que comunica el FS amb el programa |
| OMI | Marker beacon lights (outer, middle, inner) |
| PFD | Primary Flight Display |
| PTU | Power Transfer Unit |
| QRH | Quick Reference Handbook |
| RAT | RAM Air Temperature |
| RDI | Radio Direction Indicator |
| RMI | Radio Magnetic Indicator |
| SAT | Static Air Temperature |
| SDK | Software Development Kit |
| STD | Standard |
| TAT | Total Air Temperature |
| VOR | VHF Omnidirectional Radio Range |
| VSI | Vertical Speed Indicator |

# CAPÍTOL 1. CABINA

En aquest capítol es veurà en detall la cabina del DC9 que es té en l'institut Illa dels Banyols, revisant les configuracions dels indicadors i on s'ubiquen. Posteriorment es farà una comparativa amb la cabina d'un A320 i finalment es veuran algunes de les fallades més típiques per al DC9.

## 1.1 Cabina del DC9

La cabina del DC9 segueix les directrius estàndards per a la ubicació dels instruments. Aquests estàndards relacionen la configuració dels instruments amb a capacitat cognitiva del pilot, de manera que la primera informació que rep el pilot al mirar el panell sigui la més important. En aquest cas l'horitzó artificial, que mostra l'actitud de l'avió. Aquesta distribució es pot apreciar en la següent figura.

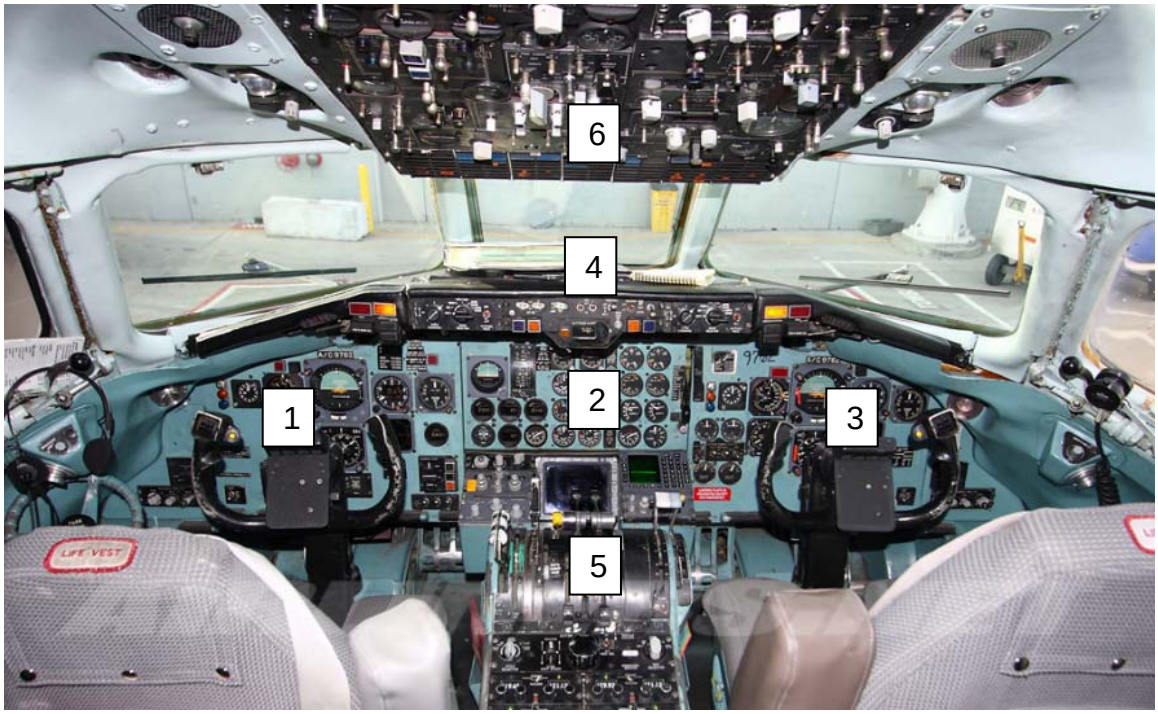



**Figura 1.1 Cabina DC9**

Com a qualsevol cabina actual, tenim sis panells clarament diferenciats:

1. Panell esquerra (pilot)
2. Panell central
3. Panell dret (copilot)
4. Visera
5. Pedestal
6. Panell superior

Pel que fa a indicadors, cada panell mostra les següents informacions:

### 1.1.1 Panell esquerra (pilot)

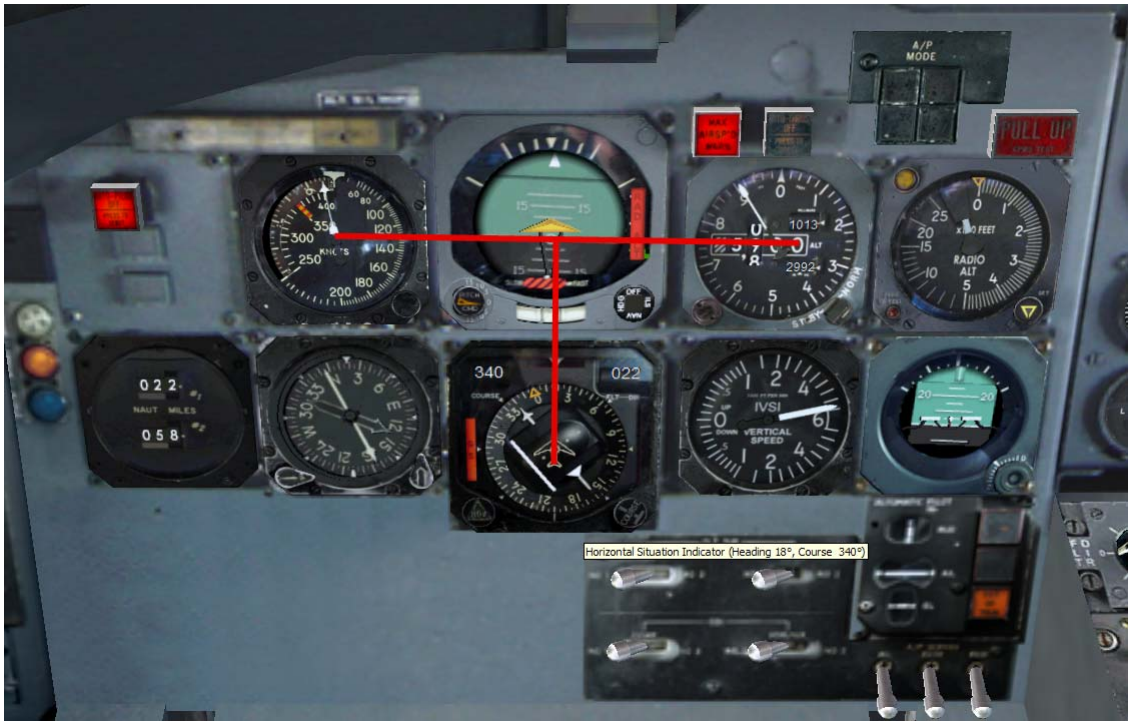



**Figura 1.2 Panell esquerra DC9**

En la figura 1.2 es pot comprovar la forma de mostrar la informació bàsica al pilot (línies vermelles de la figura 1.2). A aquesta distribució se li anomena forma de T: A la part superior dreta el pilot disposa del anemòmetre. En la part superior central de la T hi té l'horitzó artificial. En la part superior esquerra l'altímetre. I, al peu de la T, el rumb.

D'aquesta manera, amb un escanejat ràpid de la instrumentació sempre passarà pel centre de la T on podrà comprovar l'actitud actual de l'avió.

### 1.1.2 Panell central


**Figura 1.3 Panell central DC9**

Aquest és el panell central del DC9. Aquí si pot trobar la informació referent al combustible (un indicador per a cada tanc i un per al fuel total) i als motors.

Els indicadors dels motors controlen diversos paràmetres: EPR, temperatura, quantitat i pressió d'oli, temperatura dels gasos expulsats, temperatura del fuel, revolucions en % de les etapes del motor (N1 i N2) i FF (*Fuel Flow*). També disposa d'un indicador que mostra la posició física dels flaps en graus.

### 1.1.3 Panell dret (copilot)


**Figura 1.4 Panell dret DC9**

Aquest panell és el que utilitza el copilot per informar-se de l'estat del vol. Té els mateixos indicadors que el del pilot, excepte que conté també els indicadors dels circuits hidràulics de l'avió.

### 1.1.4 Visera


**Figura 1.5 Visera DC9**

Els indicadors importants que ens mostra la visera són el MASTER CAUTION i el MASTER WARNING. Aquests indicadors ens avisen d'una fallada de l'avió i segons la importància s'encén el MASTER CAUTION (fallada poc important) o el MASTER WARNING (fallada important).

S'encenen en funció del panell anunciador. Si en aquest panell s'hi encén una llum taronja (CAUTION) o vermella (WARNING) aquests dos indicadors també s'encendran respectivament.

### 1.1.5 Pedestal

**Figura 1.6 Pedestal DC9**

El pedestal de la cabina des de on es controla la potència dels motors i les comunicacions. També els flaps i els frens aerodinàmics. Per al control dels elements mecànics s'utilitzen palanques, politges i cables que actuen directament sobre les superfícies del vol o el motor. Això fa que tot plegat tingui un pes força elevat i sigui car de mantenir.

Com a indicador principal, hi ha el radar meteorològic.

### 1.1.6 Panell superior

**Figura 1.7 Panell superior DC9**

El panell superior tenim com a indicador principal el panell anunciador. Aquest panell ens mostra totes les possibles fallades del avió i és el que fa encendre els avisos de MASTER CAUTION i MASTER WARNING.
També conté el control i la indicació respectiva del sistema elèctric, el sistema antigel, la pressurització de cabina i el sistema d'aire condicionat, entre d'altres.

### 1.1.7 Indicadors implementats

Finalment, cal dir que els panells que s'han desenvolupat en el programa han estat **els laterals i el central**, ja que són els que contenen la informació més rellevant del vol i de l'avió.

A continuació es detalla una llista dels instruments que s'han pogut tractar i els que no:

| **Panell Esquerra** | |
|---|---|
| Rellotge | ✓ |
| Desviació de seguiment | x |
| Mach | ✓ |
| IAS | ✓ |
| VOR/RMI | x |
| Pitch | ✓ |

| | |
|---|---|
| Bank | ✓ |
| Heading | ✓ |
| Course | x |
| RDI | x |
| OMI | x |
| Altímetre baromètric | ✓ |
| Velocitat vertical | ✓ |
| AP Trim | x |
| Fuel total | ✓ |
| Radio altímetre | ✓ |
| **Panell Central** | |
| Nivell del tanc de fuel esquerra | ✓ |
| Nivell del tanc de fuel central | ✓ |
| Nivell del tanc de fuel dret | ✓ |
| Fuel restant | ✓ |
| EPR | ≈ |
| RAT | x |
| Temperatura del fuel | x |
| Flux de fuel | ✓ |
| Fuel usat | x |
| EGT | ≈ |
| N1 rpm | ✓ |
| N2 rpm | ✓ |
| Pressió d'oli | ✓ |
| Quantitat de d'oli | ✓ |
| Posició de flaps | ✓ |
| Tren d'aterratge | ✓ |
| TAT | ✓ |
| **Panell Esquerra** | |
| Pressió hidràulica | ✓ |
| Quantitat de líquid en el sistema hidràulic | ✓ |
| Temperatura d'oli | ✓ |
| Altímetre mètric | x |
| Temperatura de frenada | x |

**Figura 1.8 Instrumentació implementada**

Llegenda:

- ✓ Indicador tractat correctament
- x Indicador no implementat o de funcionament incorrecte
- ≈ Indicador que es pot comportar diferent del simulat segons la cabina seleccionada o la situació actual del vol.

El indicadors que no s'han tractat han estat principalment perquè la llibreria utilitzada, FSUIPC, no conté cap offset vinculat a aquests. O bé, perquè el que conté té un funcionament molt erroni respecte el que es pot visualitzar des de la cabina del FS, com és el cas del fuel usat.

## 1.2 Cabina del A320

La cabina del A320 es pot considerar la mare de les cabines actuals d'aviació comercial. Tant per la distribució instrumental, com per la senzillesa i la ergonomia que ofereix.

**Figura 1.9 Cabina A320**

La principal distinció amb la cabina d'un DC9 és que aquesta és analògica mentre que la del DC9 és digital en quasi tota la seva implementació (a excepció dels instruments de *standby* analògics i els mecanismes de control de l'avió).

### 1.2.1 Panell esquerra (pilot)


**Figura 1.10 Panell esquerra A320**

Com es pot observar, el panell esquerra, el del pilot, té tots els instruments necessaris integrats en dos displays digitals. Un és el PFD (*Primary Flight Display*) que conté la informació relativa al vol (velocitat, altitud, actitud...) i l'altra és el ND (*Navigation Display*), que mostra la informació relacionada amb la navegació.

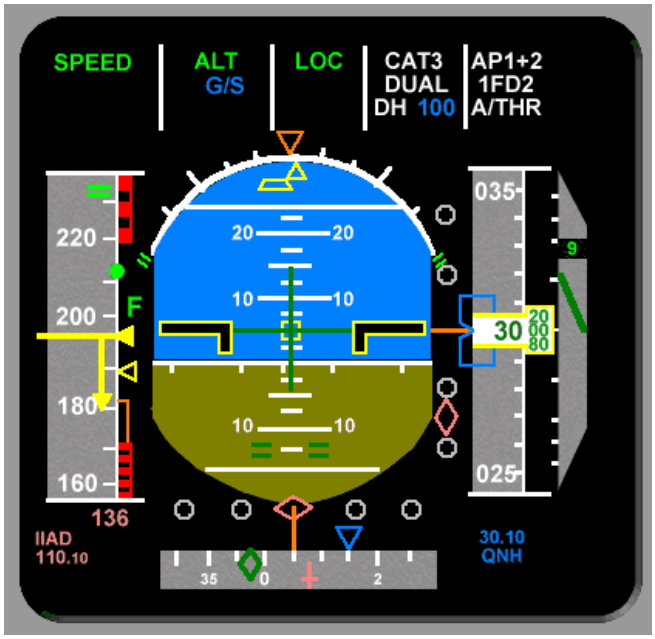



**Figura 1.11 PFD**

En la figura 1.11 es pot comprovar que la informació està distribuïda, com en la cabina del DC9, en forma de T.

El ND i el PFD formen part del que s'anomena EFIS (*Electronic Flight Information System*).

Cal indicar que el pilot ja no vola l'avió amb la palanca/volant com en el cas del DC9, sinó que ho fa mitjançant el joystick, anomenat *sidestick*, que té a mà esquerra i que es mostra en la figura 1.12. Aquesta és un pas important en les cabines d'aviació comercial ja que actualment quasi totes disposen d'aquests mecanisme per controlar l'avió. Va ser una millor implementada per la casa Airbus.

**Figura 1.12 Sidestick**

### 1.2.2 Panell central

**Figura 1.13 Panell central A320**

Igual que en el DC9, aquí es troba la informació relativa als motors i al fuel. La diferència és, com en el panell del pilot, que aquesta informació es visualitza en format digital mitjançant displays.

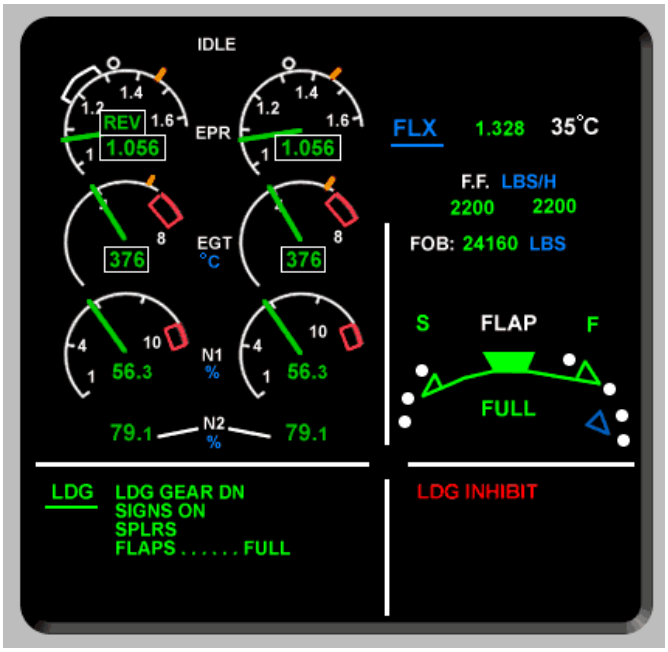



**Figura 1.14 Primary Engine Display**

La figura 1.14 mostra el display superior que s'anomena *Primary Engine Display* i informa dels paràmetres N1, N2, EGT i FF. També la posició dels flaps i del tren d'aterratge.


**Figura 1.15 Secondary Engine Display**

El display de la figura 1.15 és el display inferior del panell central i mostra informació relativa al sistema d'oli dels motors i al fuel utilitzat. També indica temperatures com la TAT i la SAT i el GW (*Ground Weight*) de l'avió.

Aquests dos display formen el ECAM (*Electronic Centralized Aircraft Monitoring*).

El panell central també conté els instruments de *standby* que s'utilitzen en cas de que fallis els sistemes digitals de l'avió. Són instruments analògics com els de la cabina del DC9. En aquests instruments s'inclou el rellotge de cabina.

Finalment es pot controlar el tren d'aterratge, com en el DC9, mitjançant una palanca.

### 1.2.3 Panell esquerra (copilot)

**Figura 1.16 Panell esquerra A320**

Com es pot veure en la figura 1.16, el panell del copilot és molt semblant al del pilot i conté els mateixos displays.

### 1.2.4 Visera

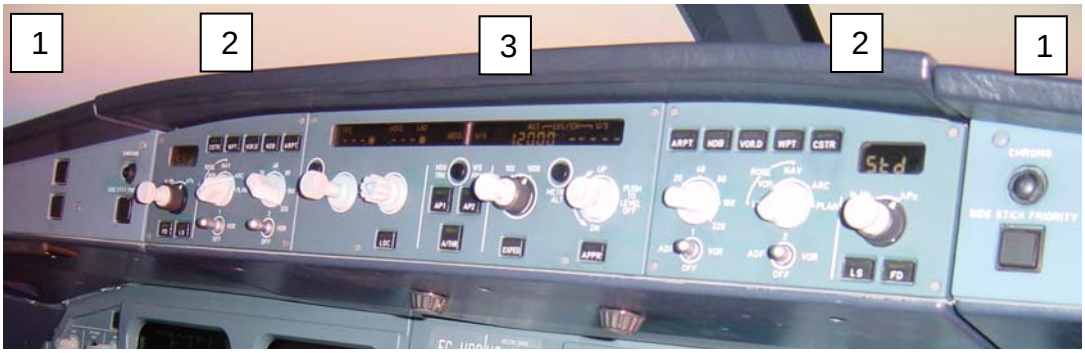



**Figura 1.17 Visera A320**

La figura 1.16 mostra la visera de l'A320. Aquesta conté el control del AutoLanding (1), els indicadors de MASTER CAUTION i MASTER WARNING, el control del EFIS per a pilot i copilot (2) i el FCU (*Flight Control Unit*) (3).

El FCU gestiona com es vola en un moment determinat (automàtic, *selected* o manual) i el *AutoThrust*, que és el control automàtic de la potència del motor.

Com es veu, aquesta visera és més complexa que la del DC9 ja que ha de poder controlar els displays digitals dels pilots.

### 1.2.5 Pedestal


**Figura 1.18 Pedestal A320**

El pedestal també varia molt respecte el del DC9. Aquest incorpora un sistema vital per a l'avió que el DC9 no té, el MCDU (*Multifunction Control Display Unit*).


**Figura 1.19 MDCU A320**

Tot el que l'avió pot fer, es pot controlar des del MDCU. Sense la configuració prèvia al vol que s'ha introduir mitjançant aquests sistema (centratge, dades de navegació, temps, etc...) l'avió no podrà volar.

D'altra banda, en el pedestal, també hi ha la palanca de gasos i dels flaps i els frens aerodinàmics, igual que en el DC9. També incorpora el control de les comunicacions de l'avió i el radar meteorològic.

### 1.2.6 Panell Superior

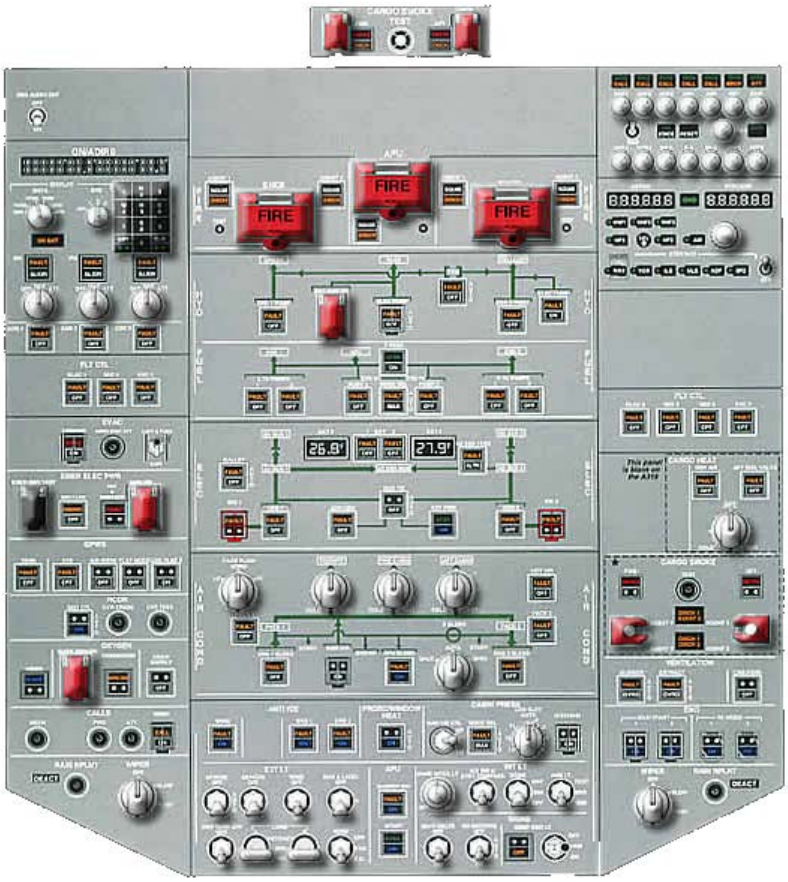



**Figura 1.20 Panell superior A320**

El panell superior del A320 conté el control del sistema de fuel, el sistema hidràulic i l'elèctric de l'avió. També el sistema d'aire acondicionat, el sistema pneumàtic i el d'antigel, entre molt altres.

Així doncs, es tracte del centre principal de control de l'avió. En aquest cas, no es diferencia excessivament del DC9 ja que igual que en aquest, s'utilitzen botons de *switch* o de *pull* per enviar les ordres de control.

## 1.3 Fallades

Els pilots dels avions de tecnologia analògica havien de reparar les fallades seguint el procediment habitual que se'ls hi havia ensenyat i ajudant-se del QRH (*Quick Reference Handbook*) que els feia recórrer un diagrama de possibles causes de la fallada i així s'aconseguia identificar el component que causava la fallada. Actualment, les cabines disposen d'una CPU que s'encarrega de detectar-les i informar al pilot dels sistemes afectats.

En els DC9, les fallades més comunes segons personal[4] que ha treballat directament amb cabina eren les següents:

1. Desviació Heading # 1 respecte del Heading # 2 superior a 6º.
   > "NO AUTOLAND" Light ON als dos FMAs
   > AP Desconnectat.

2. Indicador de velocitat vertical (VSI) # 1 indica 250 ft/min de descens VSI # 2, durant creuer amb vol estabilitzat.
   > Es recupera si es commuta amb el Air Data Computer # 2.

3. Fallada llaç de detecció de foc "A" ò "B", del Motor # 1 ò Motor # 2.
   > Es necessita commutar el *switch* de llaços, de manera que passi de la posició BOTH a la posició del llaç operatiu.

4. Foc Motor # 1 ó Motor # 2.

5. Fallada Generador Elèctric Motor # 1
   > Es produeix transferència automàtica des del Generador # 2, es commuta de manera automàtica.

6. Fallada Generador Elèctric Motor # 1 + "Transfer Lock-Out" light ON, no es tanca el relé per alimentar el sistema esquerra des del generador dret.

7. Alta temperatura en cabina de passatgers, sistema de control de temperatura automàtic PACK # 2 fallada, s'ha d'operar en mode manual.

8. Fallada bomba Hidràulica Dreta EDP#2 (Bomba de motor).
   > Entra a funcionar la PTU i es recupera la pressió.

9. Pèrdua del líquid hidràulic sistema esquerra.
   > Desconnectar PTU (a OFF).
   > Pèrdua total del sistema hidràulic esquerra.

10. Fallada d'una fase del sistema elèctric Motor # 1, per exemple, el relé del Generador # 1 té els contactes d'una fase cremats.
    > Tots els *breakers* de les dos fases bones dels motors trifàsics alimentats per aquest sistema salten.

[4] Alfredo Castro i Fernando Pardo, professors de la Illa dels Banyols.

11. Auto-Throttle OFF.

12. EPR Motor 1 fluctua durant creuer (aigua en les línees de sensors).

# CAPÍTOL 2. DISSENY DE L'ARQUITECTURA

En aquest capítol es tractarà el disseny que segueix el sistema i quines relacions hi ha entre els diferents mòdul d'aquest. Es veurà detalladament els fluxos d'entrada i sortida de cada mòdul per a comprendre la naturalesa del sistema.

## 2.1 Diagrama d'ús

A continuació s'exposa el diagrama que segueix el sistema.

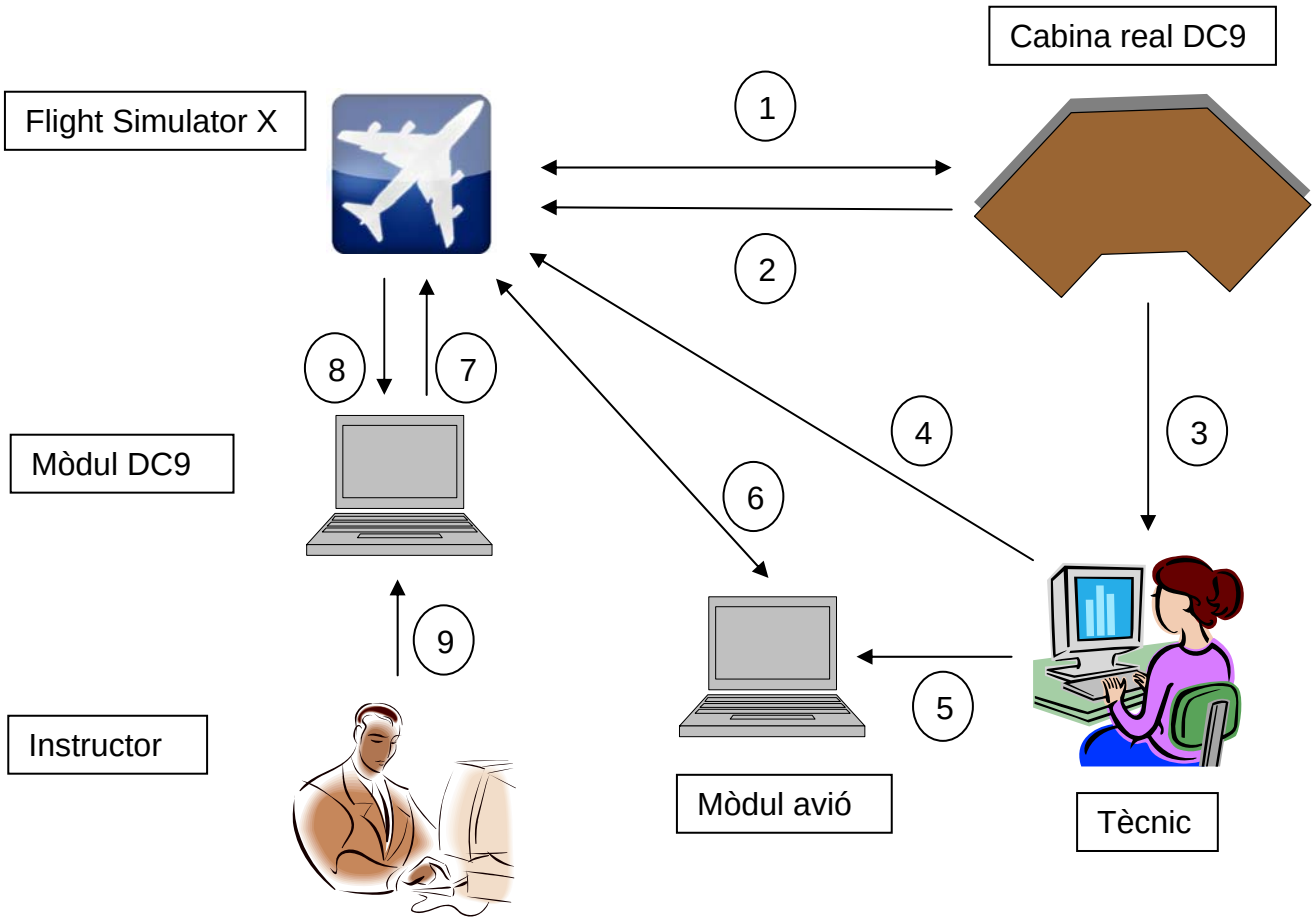



**Figura 2.1 Disseny de l'arquitectura**

### 2.1.1 Escena

L'escena que el programari desenvolupat vol resoldre és la següent: Un instructor genera unes fallades en el simulador de vol i un tècnic ha de resoldre-les, ja sigui apagant el sistema, activant un sistema redundant o realitzant qualsevol acció que millori la seguretat del vol.

### 2.1.2 Actors

- **Instructor:** Encarregat d'introduir les fallades en el simulador per a que el tècnic les resolgui.
- **Tècnic:** Encarregat de resoldre les fallades introduïdes per l'instructor.
- **Simulador:** Programa que replica un vol real i del qual n'obtindrem tots els paràmetres necessaris per a interactuar amb els altres actors.
- **Cabina real DC9:** Part de hardware que relacionarà el simulador amb el tècnic mostrant l'estat dels sistemes o les alarmes possibles del vol simulat.
- **Mòdul DC9:** Programa que conté el **mòdul de cabina DC9** i el **mòdul de fallades**. El primer obtindrà les dades del vol simulat en el FS i les mostrarà a l'instructor. També ha de permetre gestionar aquestes dades. El segon ha de permetre programar una o vàries fallades i que aquestes es reprodueixin en el simulador.
- **Mòdul avió:** Programa que permet realitzar canvis en els sistemes de l'avió per poder reparar una fallada.

## 2.2 Diagrama de flux de dades

1. **Dades de vol:** Totes les dades que la cabina ha de mostrar al tècnic (tant dades del vol com dels sistemes). Per exemple, els indicadors o les alarmes.
2. **Servos:** Encarregats de transmetre les ordres del simulador als actudors de cabina.
3. **Informació:** Indicadors i alarmes que mostren l'estat dels sistemes simulats.
4. **Comandaments:** Ordres de control per arreglar les fallades dels sistemes.
5. **Reparacions:** Accions del tècnic que reparen les fallades mitjançant el mòdul avió.
6. **Flags:** Es transmeten els estats dels sistemes de l'avió entre el mòdul avió i el simulador.

7. **Modificacions:** Fallades que introdueix el mòdul de fallades al simulador ja sigui discretes (només un canvi) o depenent del temps (fluctuants).

8. **Variables:** Paràmetres, tant de vol com dels sistemes, que el simulador envia al mòdul de fallades.

9. **Fallades:** Alteracions que l'instructor introdueix en el mòdul DC9, que conté el mòdul de fallades, perquè es produeixi una errada els sistemes del vol simulat.

# CAPÍTOL 3. DISSENY DELS MÒDULS

Vist l'esquema que haurà de seguir el sistema, tenim tres mòduls a desenvolupar:

1. Mòdul cabina DC9
2. Mòdul de fallades
3. Mòdul avió

El primer i el segon mòdul estan compresos dins el **mòdul DC9** de la figura 2.1. A continuació s'introduirà detalladament cada un i les funcions que incorporen.

## 3.1 Software utilitzat

Per a programar els mòduls s'ha utilitzat l'entorn Microsoft Visual Studio 2010, versió 10.0.30319.1 RTMRel, de ©2010 Microsoft Corporation.

El llenguatge utilitzat ha estat el C# ja que permet combinar el llenguatge C/C++ amb un entorn visual que permet crear una interfície ergonòmica i còmode per a l'usuari.

El desenvolupament dels mòduls s'ha basat en la llibreria FSUIPC, que és la que estableix la comunicació entre el FS i el mòdul. Les seves funcions i forma amb la que s'hi treballa es pot consultar en l'annex B.

## 3.2 Mòdul DC9

Seguidament s'explicarà quins són els objectius d'aquest mòdul i com s'han dut a terme.

### 3.2.1 Mòdul cabina DC9. Visualització de les dades

El primet objectiu del mòdul és visualitzar les dades de l'avió a temps real per a poder tenir un control dels sistemes i de l'estat actual del vol. Així doncs, s'hauran de reproduir els indicadors que proporcionin aquesta informació.

La majoria d'aquests indicadors els troben, com s'ha vist anteriorment, en els panells laterals i central de la cabina. De manera que seran aquests panells els que es volen replicar.

També, però, s'ha de tenir en compte que no tots els indicadors que necessitem els trobarem en els panells citats anteriorment. Per això que cal oferir un eina per a poder mostrar qualsevol altre informació que l'usuari requereixi.

Es proposa la següent interfície per a complir amb aquesta demanda:

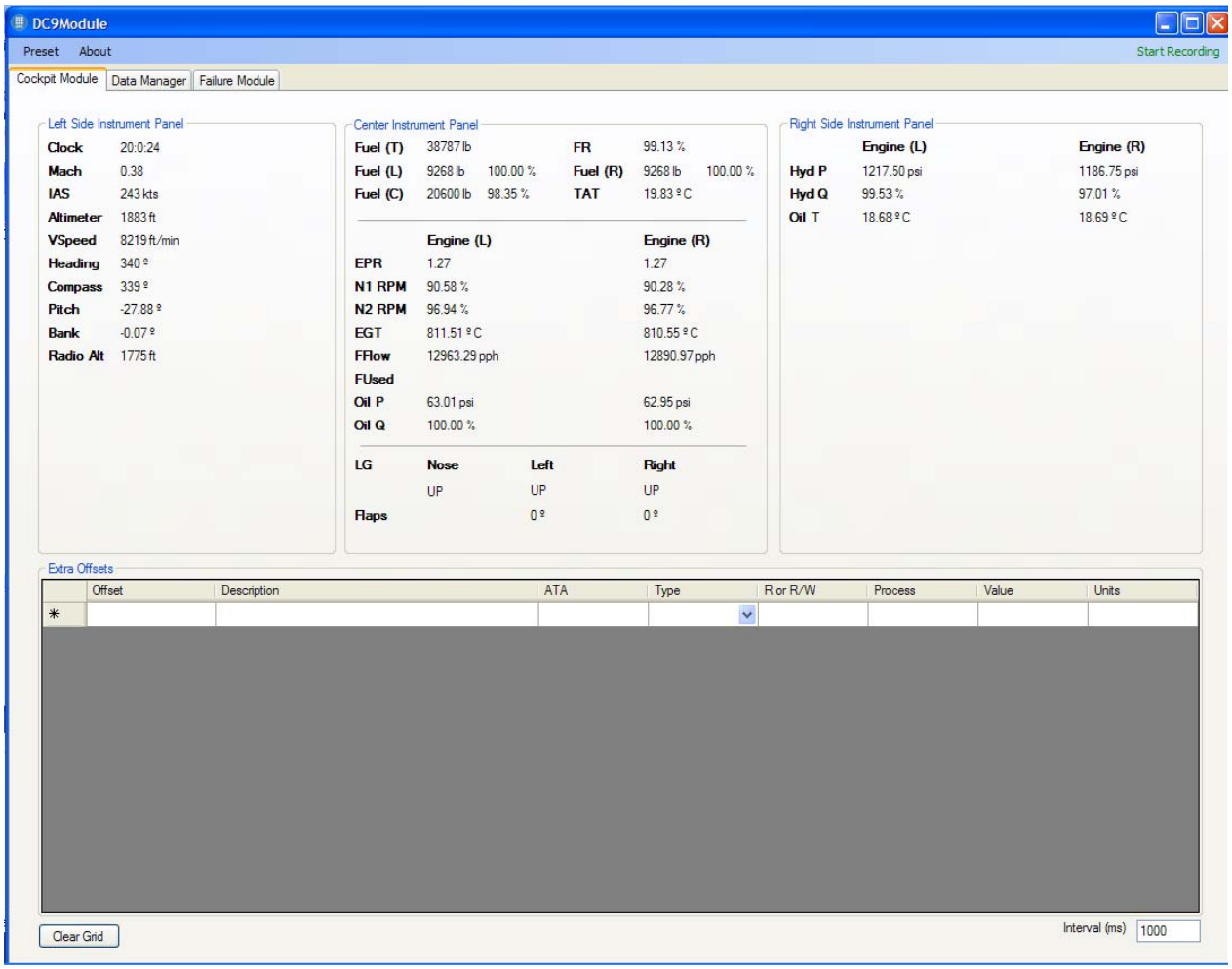



**Figura 3.1 Mòdul cabina DC9**

A través de la interfície de la figura 3.1, l'usuari rep la informació dels panells laterals i central (la informació redundant del panell dret s'ha obviat).

També pot obtenir informació addicional que no es mostra en els panells mitjançant la taula de la part inferior (figura 3.2). Aquesta taula és la que permet a l'usuari obtenir altres dades que no estan reflectides ens els indicadors de la part superior i que desitja tenir-hi accés. Així pot obtenir gran part de la informació del vol.

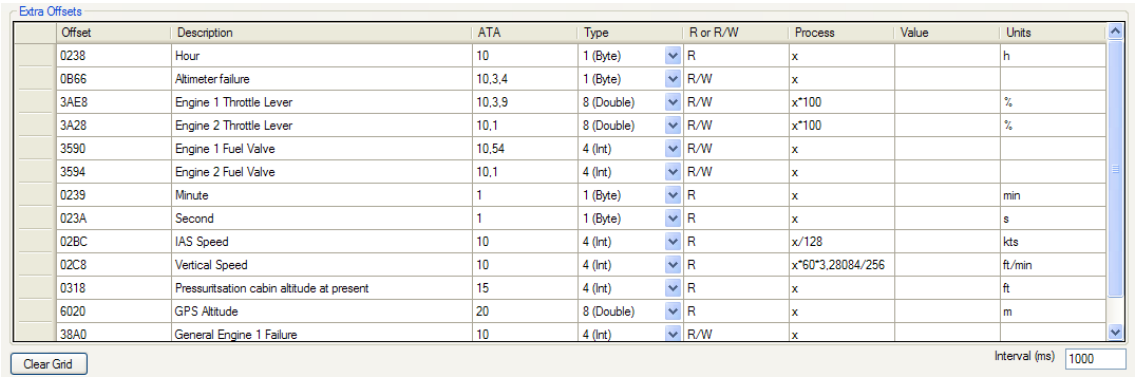



**Figura 3.2 Taula offsets extres**

Com a funcionalitat addicional, s'ha implementat una opció de permet guardar a l'usuari la taula en qüestió per a poder-la tornar a carregar en futures noves sessions i/o també carregar-la en altres aplicacions del programa (s'explica a continuació).

Per altra banda, es poden guardar totes les dades del vol. Tant les dels panells laterals i central com les dades extres en un fitxer de format *.csv.

## 3.2.2 Programació mòdul cabina DC9. Visualització de dades

Tot seguit s'explica quina metodologia s'ha utilitzat per a programar el mòdul.

### *3.2.2.1 Panells d'indicadors*

La llibreria FSUIPC disposa d'unes variables, anomenades offsets, que són les que és relacionen amb el FS. A cada indicador se li assigna un offset. El valor d'aquest offset el dóna el FS, tot i que també se li pot assignar manualment (procés que s'utilitza per a generar fallades). Aquest valor té un dels següents formats digitals: *byte, short, int, double o string* segons els bits que necessiti per ser emmagatzemat. És per això que per mostrar el valor lògic de l'offset cal una conversió.

Tota aquesta informació es troba en un document que s'adjunta amb el SDK de la llibreria. Aquesta taula indica quin offset és (clau alfanumèrica), de quin format es tracta, a quina dada fa referència (velocitat, rpm N1, EGT, etc...) i quina conversió s'ha d'aplicar per obtenir el valor en certes unitats.

Vist això, només cal trobar quina offset correspon a quin indicador i anar refrescant el valor constantment mitjançant un temporitzador perquè l'usuari pugui visualitzar el valor del indicador correctament.

### *3.2.2.2 Taula offsets extres*

Com s'ha explicat anteriorment, aquesta taula permet a l'usuari obtenir el valor de l'offset que vulgui. Aquest ha d'estar disponible en el document d'offset que s'ha citat anteriorment.
A banda de la clau alfanumèrica de l'offset, l'usuari també a d'introduir el format, la descripció de la dada a la que fa referència, el capítol ATA al que pertany (serveix per a funcionalitats que s'explicaran més endavant), si és de lectura o lectura i escriptura, la conversió que se li aplica i les unitats que té després de fer la conversió.

Amb tot això, es pot crear una *arraylist* [5] dels offsets i processar-los igual que si haguessin estat programat per defecte com els indicadors. Finalment amb un temporitzador es van actualitzant el valor dels offsets que es pot veure, feta ja la conversió, en la setena columna de la taula.

Aquesta taula es pot guardar en un fitxer *.cfg per a ser carregada quan ho desitgi l'usuari. Serveix tant per tornar a carregar aquesta taula com per a altres funcionalitats que s'expliquen en el següent apartat.

### 3.2.3 Mòdul cabina DC9. Gestió de dades

El segon objectiu del mòdul cabina DC9 és poder gestionar les dades que anteriorment s'han explicat. Per a poder-ho fer és necessari un guardat de les dades i una aplicació que permeti manipular-les al gust del client.

També s'ha d'implementar un filtre que permeti distingir les diferents dades segons un criteri universal, com són els capítols ATA.

[5] Estructura que emmagatzema les dades en un *array* (o vector), en forma d'objecte, i que pot se'n pot modificar la mida dinàmicament. Un *array* és una zona d'emmagatzematge continu que conté una sèrie d'element dels mateix tipus. Des del punt de vista lògic, aquests elements estan disposats en una matriu unidimensional.

Una possible resposta a aquesta demanda és mitjançant la següent aplicació del programa:

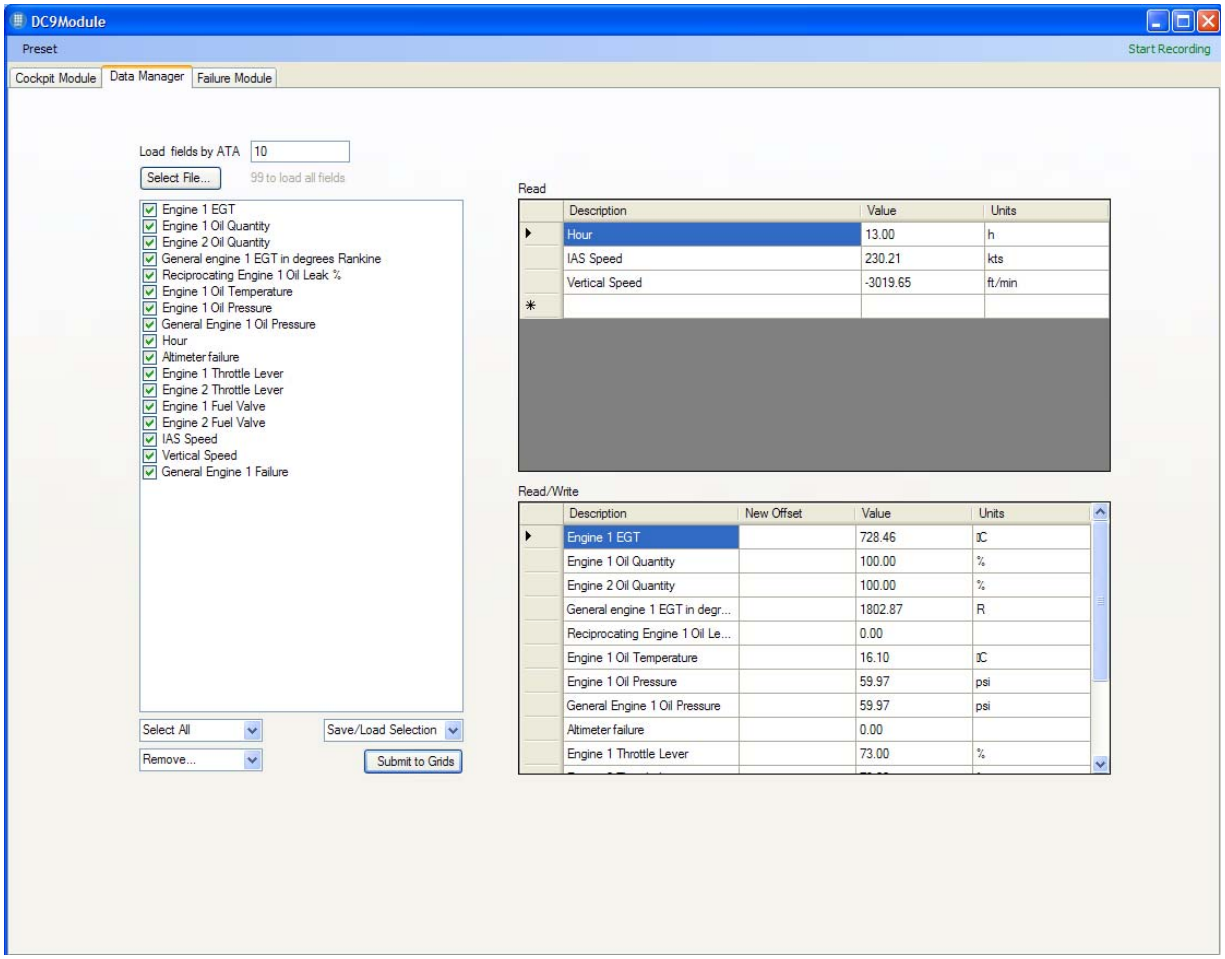



**Figura 3.3 Gestió de dades**

El quadre esquerra permet carregar a l'usuari les diferents variables que hagi guardat prèviament mitjançant la taula d'offsets extres. Les variables a carregar del documents es poden filtrar per els capítols ATA que corresponguin i d'aquesta manera acotar els resultats. Finalment es seleccionen les que es vulgui gestionar i es trameten a les dues taules de la dreta segons siguin variables de lectura o de lectura/escriptura (l'usuari podrà tractar manualment aquestes últimes). Hi ha una opció per guardar aquesta selecció i carregar-la més endavant.

### 3.2.4 Programació mòdul cabina DC9. Gestió de dades

Aquesta aplicació del mòdul cabina DC9 treballa directament dels fitxes que l'usuari hagi pogut crear en la taula d'offsets extres.
Primer de tot, quan l'usuari selecciona el fitxer a carregar, a cada offset se li aplica el filtre del capítol ATA. Si l'offset en qüestió està especificat en tots els

capítols del filtre podrà ser carregar en la llista de la selecció. Si en conté menys dels que es demana no podrà carregar-se. Per exemple, un offset que l'usuari ha especificat en la taula d'offset extra que està contingut en els capítols 3, 5 i 24 podrà ser carregat en cas que el filtre sigui 3, sigui 5, sigui 3 i 24 o qualsevol combinació d'aquests tres números no essent tots necessaris en aquesta. Però en canvi si el filtre és 3, 8 i 24 en cap cas es podrà carregar ja que el capítol 8 no està especificat per aquest offset.
Per no aplicar aquest criteri s'ha pactat amb el client introduir com a filtre el capítol 99. En aquest cas es carregarien tots els offsets del fitxer.

Una vegada es tenen els offset filtrats per capítols ATA en la llista es pot procedir a seleccionar els que es volen gestionar. Aquesta selecció es pot guardar en un fitxer *.cfg per a carregar-se posteriorment si es desitja (normalment s'utilitza per a llistes molt llargues).
Un cop ja feta la selecció es trameten els offset en les taules del costat de la llista segons siguin de lectura (taula superior) o de lectura i escriptura (taula inferior). El que permet fer aquesta distinció és el camp de lectura o lectura i escriptura que l'usuari ha indicat en la columna 5 de la taula d'offsets extres (figura 3.2) i que s'ha guardat posteriorment en el fitxer.
Així doncs, els offset de lectura aniran a la taula superior on l'usuari podrà veure el seu valor actualitzat per un temporitzador i ja aplicada la conversió que s'ha guardat en el fitxer.
Pel que fa als offset de lectura i escriptura, l'usuari també podrà veure el seu valor actualitzat en tot moment com s'ha indicat anteriorment. La diferència però, és que ara hi ha una camp de la taula on ell mateix podrà introduir el valor, en format digital (no lògic), de l'offset en qüestió. Una altra vegada, mitjançant un temporitzador, aquest nou valor es tramet per a cada *tick* al simulador de manera que queda fixat fins que no s'esborri el valor de la cel·la.

## 3.3 Mòdul de fallades

El mòdul de fallades que es requereix ha de poder ser capaç de generar una fallada programada en l'avió. Aquesta fallada s'ha de poder controlar segons criteris temporals i de magnitud.

D'altra banda s'ha de poder generar fallades paral·lelament o que siguin conseqüència una de l'altre.

Per a resoldre aquesta demanda, es proposa la següent interfície:

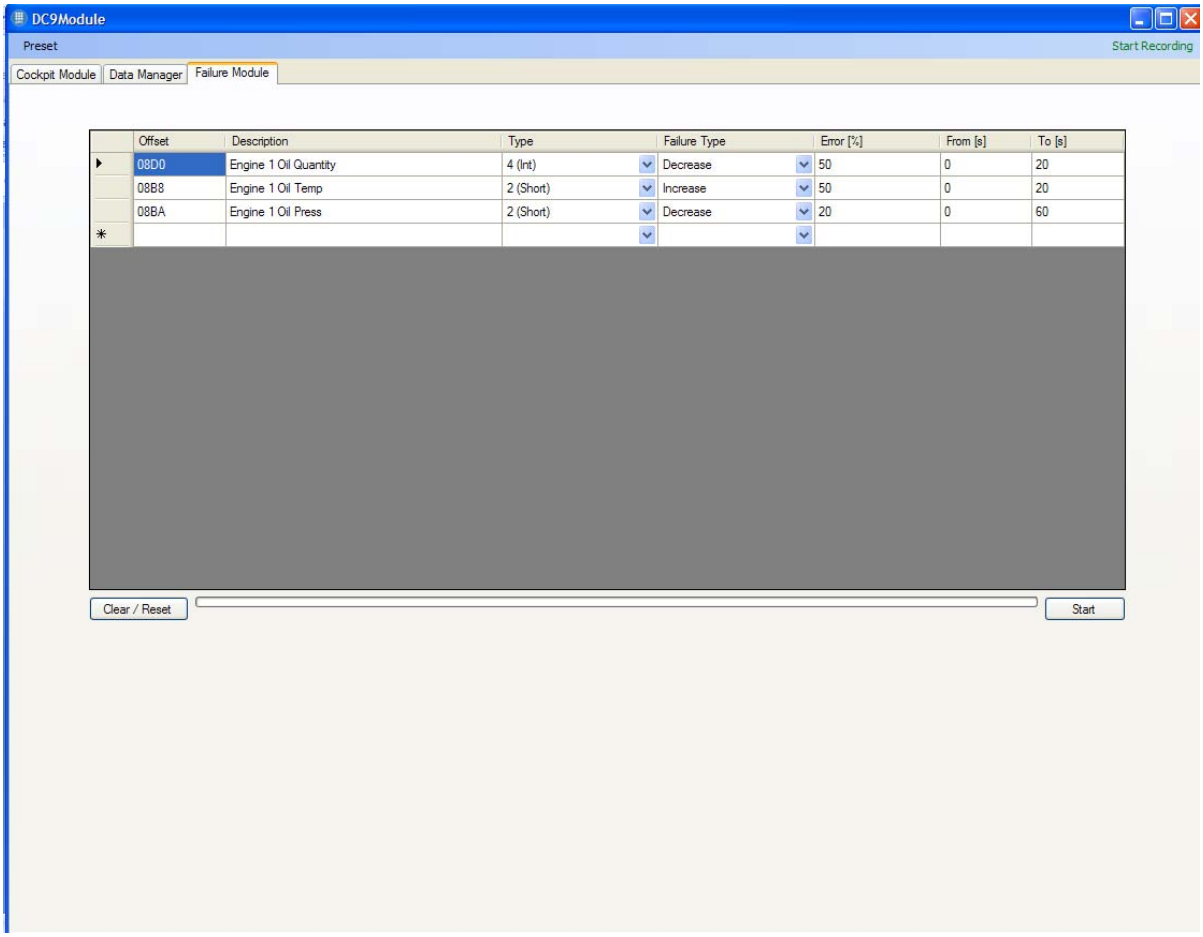



**Figura 3.4 Mòdul de fallades**

Mitjançant la taula que s'observa en la figura 3.3, l'usuari pot programar una fallada que estigui composta per la fallada de més d'un component/sistema. Es podem programar fallades que es reprodueixen paral·lela o successivament
L'usuari també selecciona quin tipus de fallada desitja:

1. **Increment**: El valor de la variable en qüestió augmenta linealment el % indicat en la cinquena columna.

2. **Decrement**: El valor de la variable en qüestió disminueix linealment el % indicat en la cinquena columna.

3. **Fluctuació**: El valor de la variable en qüestió fluctua entre el marge de ± % indicat en la cinquena columna.

Les fallades programades es poden guardar per a poder-les carregar quan ho desitgi l'usuari.

També hi ha la opció d'aturar el procés de la i reprendre'l més endavant.

### 3.3.1 Programació mòdul de fallades

Per a generar una fallada cal, primer de tot, que els offsets que l'usuari indiqui en la taula de fallades siguin de lectura i escriptura. Sinó no se li podrà assignar un nou valor a l'offset des del programa.

Amb això, l'usuari ha d'introduir el codi alfanumèric de l'offset que es tractarà, el tipus d'offset que és, el tipus de fallada i quina magnitud de fallada desitja. Per acabar només cal introduir en quin espai temporal vol reproduir la fallada.

#### *3.3.1.1 Fallades d'increment i decrement*

Una vegada es sap l'offset que es vol modificar primer de tot se'n guarda el seu valor en una *hashtable*[6], anomenada *original_value*, abans d'alterar-lo per a poder aplicar el següent algorisme:

```
percent = (int)original_value[i] * step / 100;
result = Convert.ToDouble(((Offset<int>)failure[i]).Value) ± percent;
((Offset<int>)failure[i]).Value = Convert.ToInt32(result);
```

(1)

La primera instrucció determina quant s'ha d'incrementar o disminuir el valor de l'offset (en format digital) per a cada pas del temporitzador durant l'espai temporal en el que es produeix la fallada. Això s'aconsegueix multiplicant el valor original de l'offset per la variable *step*. Aquesta es pot calcular segons la següent fórmula:

$$step = (Error\ [\%] / t\_max - t\_min)$$

(2)

Així doncs, per a cada pas de temporitzador el valor de l'offset augmentarà o disminuirà segons es sumi o resti la quantitat calculada, anomenada *percent*, al

[6] Estructura de dades que associa claus amb valors.

valor actual de l'offset. Això només es produeix quan la fallada es troba dins el seu espai temporal.

#### *3.3.1.2 Fallades de fluctuació*

Per a produir una fluctuació continguda dins el marge que indiqui l'usuari s'utilitza el següent algorisme:

```
Random rand = new Random();
double random = (rand.NextDouble() * 2 - 1);
percent = (int)original_value[i] * random *
          Convert.ToDouble(Row.Cells[4].Value) / 100;
result = Convert.ToDouble((int)original_value[i]) + percent;
((Offset<int>)failure[i]).Value = Convert.ToInt32(result);
```

(3)

Aquest algorisme genera una número aleatori comprés entre el -1 i l'1 que es multiplica per al valor original de l'offset i per al error que l'usuari hagi indicat en la taula del mòdul de fallades (finalment es divideix entre 100 per a corregir l'error, que és en %). D'aquesta manera s'obté un valor, positiu o negatiu, i dins del marge indicat per l'usuari com a error, que s'emmagatzema en la variable *percent*.
En acabat, aquest valor se li suma a l'original de l'offset i es transmet el valor al FS.

## 3.4 Mòdul avió

El mòdul avió és el que permetrà al tècnic de manteniment reparar la fallada que ha generat l'instructor. Així doncs, mitjançant una interfície que conté diferents opcions per diagnosticar i reparar la fallada, el tècnic haurà d'anar deduint quin és la causa de la fallada i finalment procedir a la seva reparació o capar el sistema afectat.

Aquest mòdul no ha estat desenvolupat en aquest projecte. Es preveu que es programi en futurs projectes relacionats amb aquest.

Actualment hi ha diverses maneres d'elaborar aquest mòdul. Una d'elles és fer passejar virtualment al mecànic per l'avió perquè pugui analitzar les parts que poden estar afectant a la fallada i així aconseguir identificar la font de l'error.
Una altra manera de representar aquest mòdul és mitjançant un test dinàmic, és a dir, les preguntes que es formulen varien en funció de la resposta anterior. Així es pot crear un diagrama de fallades en el que el tècnic mecànic va recorrent per veure finalment l'origen de l'error.

# CAPÍTOL 4. CONCLUSIONS

## 4.1 Conclusions Generals

Tenint el comptes els objectius inicials es pot concloure que: el programa desenvolupat ha estat capaç d'obtenir les dades dels indicadors principals de cabina i mostrar-les a través d'una interfície senzilla i intuïtiva. També s'ha implementat una funcionalitat addicional que permet obtenir les dades que necessiti l'usuari (en funció de la llibreria FSUIPC) i que poden no estar mostrades per defecte. A més a més, les dades obtingudes es poden gestionar i modificar a través de l'aplicació de gestió de dades.
D'altra banda, s'ha desenvolupat i implementat el gestor de fallades que es demanava. Aquest està dotat de 3 tipus de fallades i permet sincronitzar fallades entre sí seqüencial i simultàniament. Tot i això, es contempla que pugui haver futures millores per a aquest mòdul com, per exemple, la implementació de nous tipus de fallades (la variació de l'offset al llarg del temps no sigui lineal) i també una automatització de la sincronització de fallades.

Ara per ara, l'únic mòdul de software que falta per desenvolupar és el mòdul avió, que és el que permet al tècnic reparar la fallada. Una vegada implementat aquest, ja es podrà utilitzar com a suport a la docència de la Illa dels Banyols.

## 4.2 Impacte Ambiental

El desenvolupament d'aquest projecte basat no causat cap impacte directe al medi ambient. Únicament s'ha utilitzar electricitat per alimentar l'ordinador portàtil des del que s'ha treballat.

# CAPÍTOL 5. BIBLIOGRAFIA

- **Llibres**

[1] Mark Michaelis, *Essential C# 4.0*, Addison-Wesley Professional, 2010

- **Internet**

[2] http://www.schiratti.com/dowson.html

[3] http://msdn.microsoft.com/en-us/library/ms123401.aspx

[4] http://www.a320homecockpit.de/downloads/a320flightdeckdocumentation26202.pdf

[5] http://www.s-techent.com/ATA100.htm

[6] http://stackoverflow.com/

[7] http://www.c-sharpcorner.com/

[8] http://www.lundin.info/mathparser.aspx

[9] http://www.hilmerby.com/dc9/cockpit.html

[10] http://es.wikipedia.org/wiki/McDonnell_Douglas_DC-9

[11] http://en.wikipedia.org/wiki/Airbus_A320_family

[12] http://s80.flight1.net/

[13] http://www.codeproject.com/

[14] http://www.seguridadaerea.es/NR/rdonlyres/4B0EB17F-FC7E-4E82-933A-4A3189BE978B/39826/Modulo05_cap01_REV1.pdf

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEXOS

**TÍTOL DEL TFC: Disseny d'un simulador d'aviònica per a una cabina de DC9/A320 basat en Flight Simulator**

**TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació**

**AUTOR: Bernat Font Garcia**

**DIRECTOR: Oscar Casas Piedrafita, Joshua Tristancho Martínez**

**DATA: 25 de juliol de 2012**

# ANNEX A
# MANUAL D'USUARI

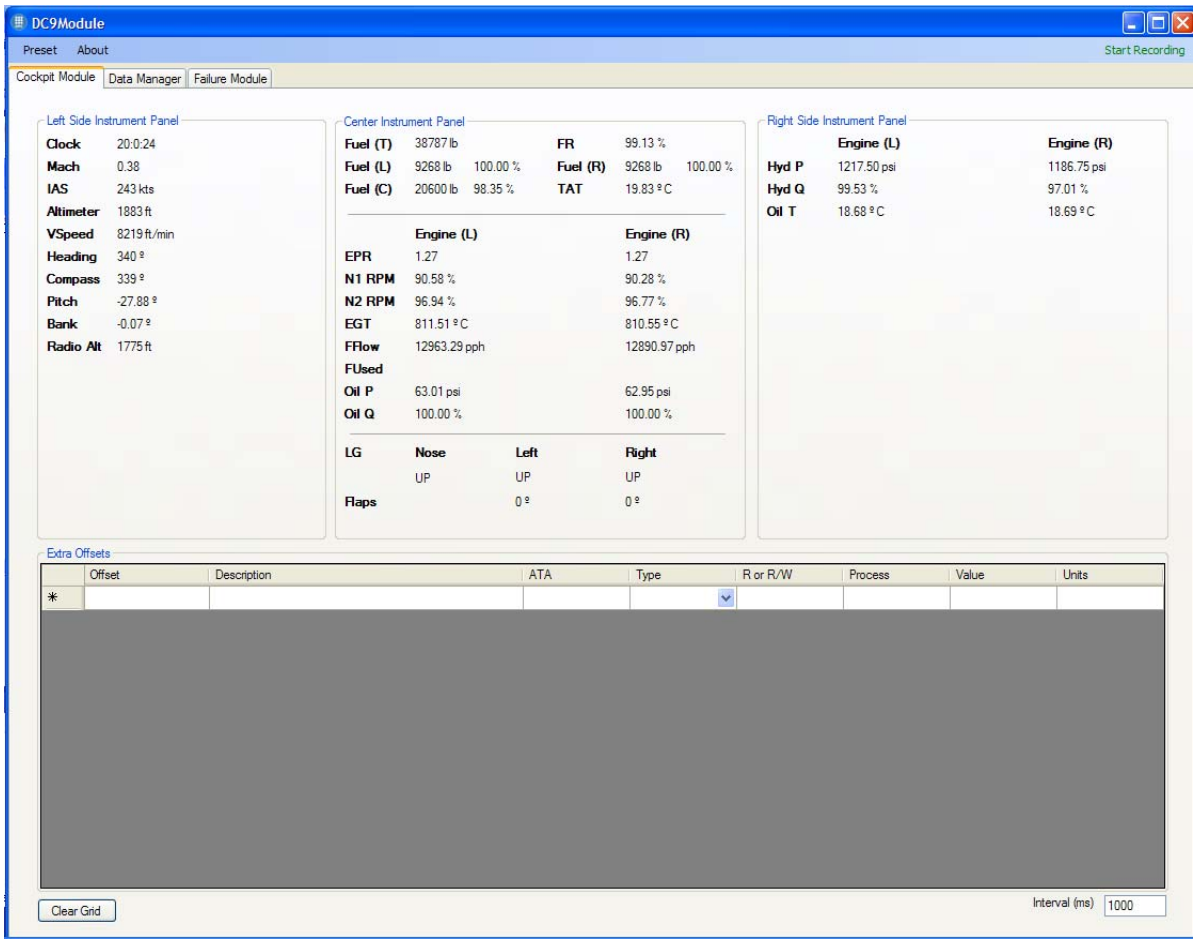
# INTRODUCCIÓ

La finalitat d'aquest annex és ajudar a l'usuari a entendre el programa i a treballar amb ell, donant a conèixer totes les seves funcions i utilitats perquè pugui treure'n el màxim rendiment.

S'explicaran cada una de els parts del programa detallada i acuradament.

En cas de que tot i això l'usuari tingui algun dubte de com s'utilitza qualsevol funció o hagi detectat un error, em poso a disposició per a qualsevol aclariment o reparació que es necessiti del programa: **bernatfontgarcia@gmail.com**

# A.1 Mòdul cabina DC9




**1. Mòdul cabina DC9**

La figura 0.1 mostra la pestanya (la primera) on es troba el mòdul de cabina del programa.

La part superior mostra a l'usuari la majoria dels indicadors que es troben en els panells laterals i central de la cabina. Està descrit com *Left Side Instrument Panel, Center Instrument Panel i Right Side Instrument Panel*.

D'altra banda tenim la taula d'offsets extres, que són aquelles dades que l'usuari vol extraure del FS i no estan mostrades en els panells descrits anteriorment.

La taula conté vuit columnes que fan referència als següents paràmetres:

Extra Offsets

| | 1 Offset | 2 Description | 3 ATA | 4 Type | 5 R or R/W | 6 Process | 7 Value | 8 Units |
|---|---|---|---|---|---|---|---|---|
| | 3AE8 | Engine 1 Throttle Lever | 10,3,9 | 8 (Double) | R/W | x*100 | | % |
| | 3A28 | Engine 2 Throttle Lever | 10,1 | 8 (Double) | R/W | x*100 | | % |
| | 3590 | Engine 1 Fuel Valve | 10,54 | 4 (Int) | R/W | x | | |
| | 3594 | Engine 2 Fuel Valve | 10,1 | 4 (Int) | R/W | x | | |
| ▶ | 0318 | Pressuritsation cabin altitude at present | 15 | 4 (Int) | R | x | | ft |
| | 6020 | GPS Altitude | 20 | 8 (Double) | R | x | | m |
| | 38A0 | General Engine 1 Failure | 10 | 4 (Int) | R/W | x | | |
| * | | | | | | | | |

**2. Taula offsets extres**

1. **Offset**: Número de referència de la variable per a que el programa pugui fer la petició al FS. Cada variable del FS té un offset diferent. Són valors alfanumèrics. Es poden trobar en el document que s'adjunta en l'annex C. A partir d'ara es farà referència a aquest document com a: document d'offsets.

2. **Description:** Descripció del offset perquè es vegi a què està fent referència.

3. **ATA:** Capítols ATA als que pot pertànyer l'offset. Serveix per a que l'usuari pugui filtrar els offsets en utilitats que s'explicaran més endavant. Es pot escriure més d'un capítol per offset però han d'anar separats per comes i sense espai. Per exemple: '9,27,4'.

4. **Type:** Tipus d'offset. Al document d'offsets on es troba el codi de l'offset també s'informa de quin tipus d'offset és. Poden ser: Byte (1 sol byte), Short (2 bytes), Integer (4 bytes), Double (8 bytes) i String (més e 8 bytes).

5. **R or R/W:** Els offsets poden ser de lectura o de lectura i escriptura. S'ha d'indicar com a 'R' quan són de lectura i 'R/W' quan són de lectura i escriptura. Si no es segueix aquest metodologia el programa no reconeixerà l'offset en utilitats que s'expliquen més endavant. Això també es troba en el document d'offsets. Cal veure, però, que a vegades poden ser offsets que no tenen un bon funcionament si així ho indica el document: **?-SimC**. O directament no es poden llegir o llegir i escriure: **No**.

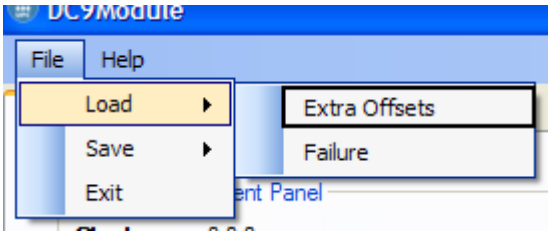
6. **Process:** És el processat que l'offset necessita per mostrar-se de manera lògica i acord amb les seves unitat. Com s'ha vist en el punt 4, obtenim valors digitals i cal passar-los a valors reals. Així doncs, s'indica quina és la operació matemàtica que cal aplicar per obtenir el valor correcte (moltes vegades ve indicat en la pròpia taula d'offsets).
   La variable 'x', escrita en minúscula, simbolitza l'offset que s'obté directament del FS (valor digital) i s'utilitza en la equació per a obtenir el resultat lògic. Per exemple, el document d'offsets indica que per a l'offset 0988 (Motor 2 Temperatura de turbina) el valor que obtenim és: °C * 16384. Llavors el que s'ha d'introduir en aquest camp de la taula és: 'x/16384'. Així s'aplicarà la conversió i s'obtindrà el resultat en °C.

   Nota: Els números decimals es separen amb ',' (coma).

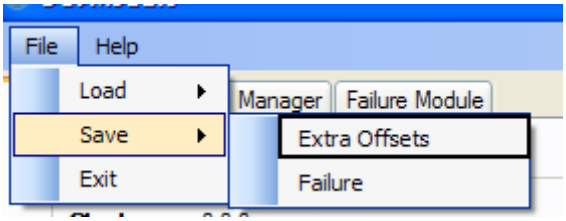
7. **Value:** És el resultat ja tractat amb el processat que s'ha indicat que s'obté una vegada s'indica el fitxer on es guardaran els resultats (s'explica més endavant). És la única columna que l'usuari no pot modificar.

8. **Units:** Unitats del offset ja tractat en qüestió. Si no en té es deixa en blanc.

Per a guardar la taula que s'ha creat i poder-la tornar carregar quan desitgi l'usuari s'han d'utilitzar els següents controls:



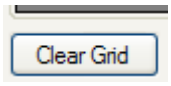

**3. Carregar taula offsets extres**




**4. Guardar taula offsets extres**

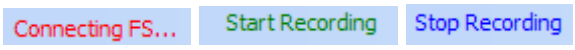
1. **Load Extra Offsets:** Carregar la taula d'offsets extres guardada anteriorment per l'usuari en un fitxer *.cfg.
2. **Save Extra Offsets:** Guarda la taula generada per l'usuari en un fitxer *.cfg.

Es pot esborrar la taula d'offsets extres amb el següent botó:




**5. Esborrar taula**

Per guardar les dades del vol que es van generant (tant les dels panells d'indicadors com les de la taula d'offsets extra s'utilitza el següent control:



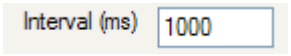

**6. Connexió amb FS i guardar dades**

Quant el control està en vermell indica el programa no està connectat al FS, pel que no es poden obtenir dades ni utilitzar cap funcionalitat del programa.

Una vegada està en verd vol dir que la connexió amb el FS ha estat establerta i podem començar a guardar les dades generades pels indicadors dels panells laterals i centrals i la taula d'offsets extres. Per a fer això, demanarà generar un arxiu amb format *.csv.

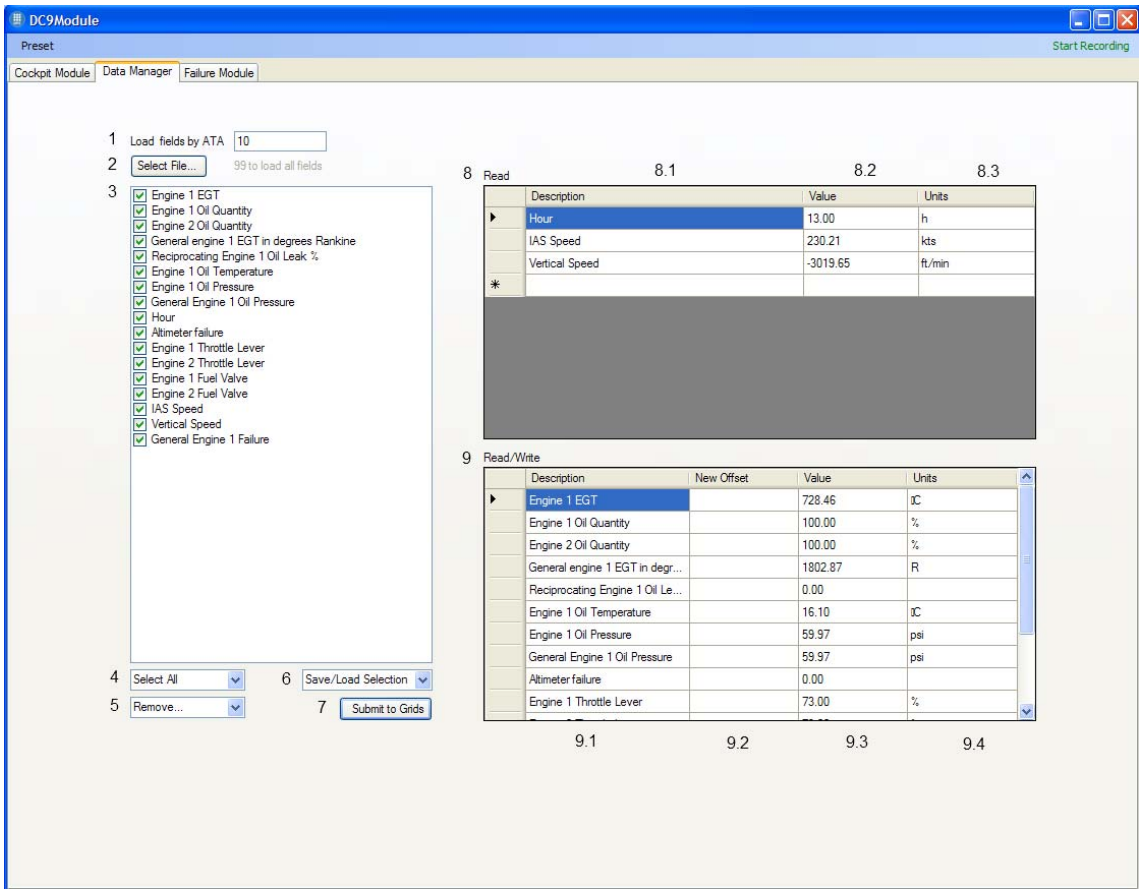
Finalment quan el control està en blau indica que es pot aturar el guardat de dades si es prem.

- L'usuari pot seleccionar quin és el temps d'interval per al guardat entre dada i dada mitjançant el següent control:




**7. Interval**

## A.2 Gestió de dades




**8. Gestió de dades**

La figura 0.2 mostra la pestanya (la segona) on es troba la gestió de dades que obtenim del FS.

1. **Load fields by ATA:** L'usuari indica el filtre dels offsets que vol carregar d'un fitxer que es selecciona en el control 2. Si ho deixa en blanc no es carregarà cap offset. Si s'introdueix el número 99 es carregaran tots. Es pot posar més d'un capítol ATA utilitzant el mateix format que en la columna 3 de la taula d'offsets extres.

2. **Select file...:** Permet a l'usuari seleccionar el fitxer des del qual s'obtindran els offsets. Aquest fitxer ha d'haver estat generat mitjançant la taula d'extra offsets del mòdul de cabina (primera pestanya).

3. **Offsets Checklist:** Apareixen els offsets segons els capítols ATA indicats en el control 1 i del fitxer seleccionat en el control 2.

4. **Select ComboBox:** Control múltiple que permet seleccionar o desseleccionar tots els offsets de la llista.

5. **Remove ComboBox:** Control múltiple que permet eliminar l'offset seleccionat o tots els offsets de a llista

6. **Save/Load Selection ComboBox:** Control múltiple que permet guardar la selecció actual de la llista o carregar una selecció ja guardada. Està pensat sobretot per a seleccions molt extenses, de manera que l'usuari no haurà de tornar a fer la selecció en una nova sessió del programa.

7. **Submit to grids:** Botó que permet trametre els offsets seleccionats a les graelles indicades com a 8 i 9 en la figura 8.

8. **Read Grid:** Taula d'offsets de lectura segons s'hagin indicat en la taula d'offsets extres de la primera pestanya.

   8.1. **Description:** Descripció del offset perquè es vegi a què està fent referència.

   8.2. **Value:** Valor obtingut del FS i ja processats segons hagi indicat l'usuari.

   8.3. **Units:** Unitats de l'offset en qüestió.

9. **Read/Write Grid:** Taula d'offset de lectura i escriptura segons s'hagin indicat en la taula d'offsets extres de la primera pestanya.

   9.1. **Description:** Descripció del offset perquè es vegi a què està fent referència.

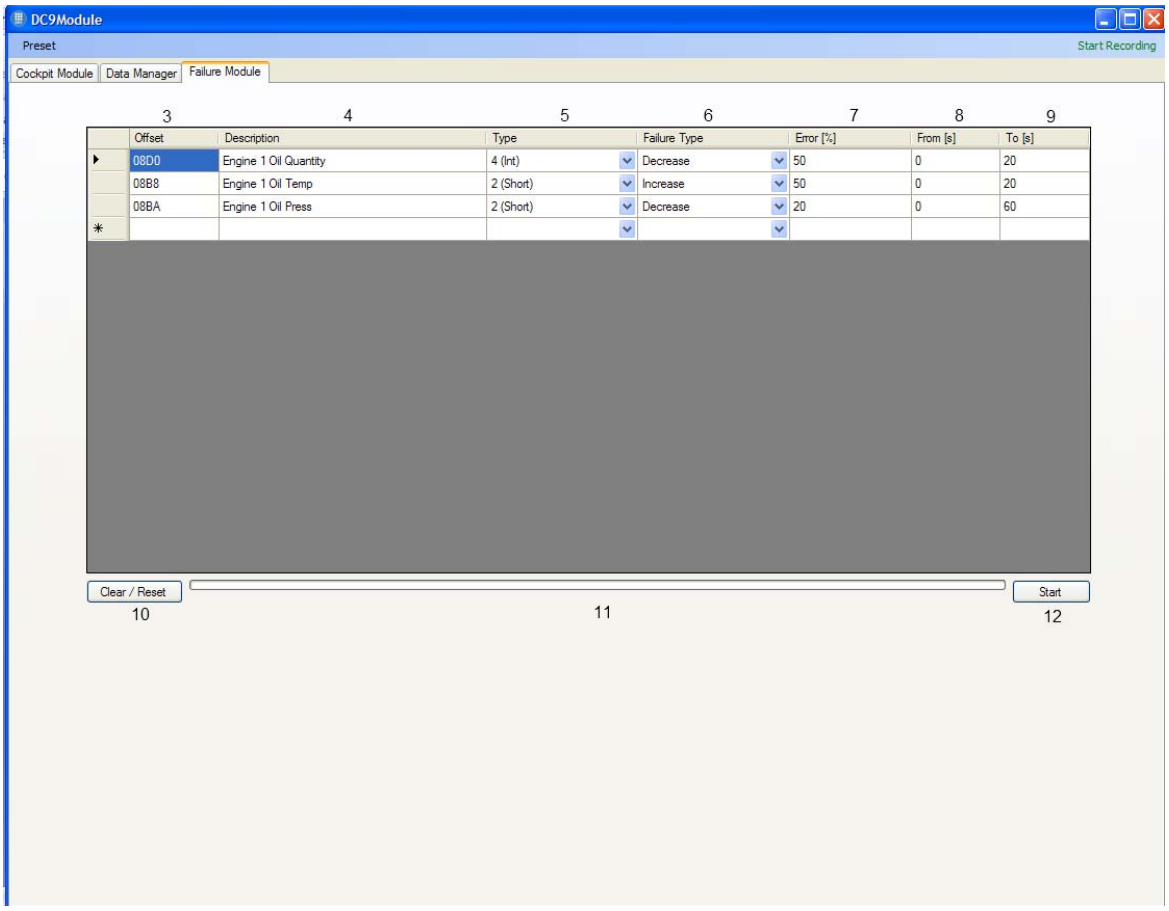
   9.2. **New Offset:** Nou valor en format digital (sense processar,) que se li vol donar a l'offset. Si està en blanc no s'aplicarà cap canvi en l'offset i no es veurà cap canvi des del FS. Si s'indica un valor, el FS llegirà aquest valor constantment i el processarà. Serveix per introduir nous valors a offsets i veure el comportament de l'avió a tals canvis.

   Nota: Si mentre en la pestanya 3 (Mòdul de fallades) s'està executant una fallada que afecta a un dels offset que hi ha en aquesta taula i aquest camp conté un valor, la fallada no es produirà ja que el FS llegirà sempre el valor d¡aquesta taula. Per tant, s'ha de vigilar en no està modificant cap offset des d'aquesta taula si es vol executar una fallada.

   9.3. **Value:** Valor obtingut del FS i ja processats segons hagi indicat l'usuari.

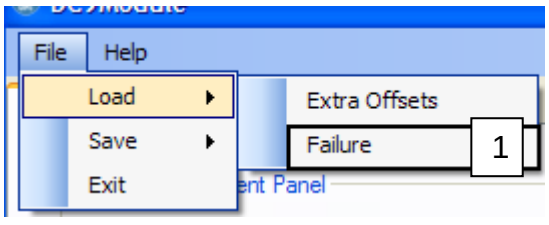
   9.4. **Units:** Unitats de l'offset en qüestió.

## A.3 Mòdul de fallades

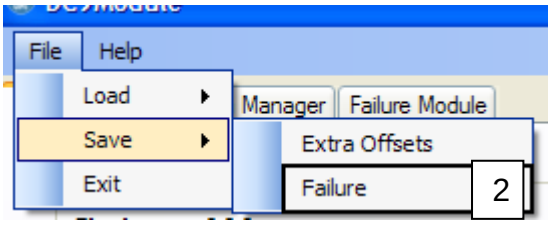



**9. Mòdul de fallades**

La figura 9 mostra la pestanya (la tercera) on es troba el mòdul de fallades. Bàsicament és una eina per a que l'usuari pugi generar una fallada controlada des el programa.




**10. Carregar fallada**




**11. Guardar fallada**

1. **Load failure:** Permet carregar a l'usuari una fallada que s'hagi guardat anteriorment i hagi estat generada mitjançant el mòdul de fallades. El fallada només es carregarà correctament si l'arxiu ha estat generat pel mòdul de fallades.

2. **Save failure:** Permet guardar una fallada que s'hagi programat a la taula de generació de fallades. L'usuari ha d'indicar el directori on vol que es generi el fotxer per a guardar la fallada.

3. **Offset:** Número de referència de la variable per a que el programa pugui fer la petició al FS. Cada variable del FS té un offset diferent. Són valors alfanumèrics.

4. **Description:** Descripció del offset perquè es vegi a què està fent referència.

5. **Type:** Tipus d'offset. Al document d'offsets on es troba el codi de l'offset també s'informa de quin tipus d'offset és. Poden ser: Byte (1 sol byte), Short (2 bytes), Integer (4 bytes), Double (8 bytes) i String (més e 8 bytes).

6. **Failure type:** Tipus de fallada que es vol generar. N'hi ha 3 tipus:

   6.1. **Increase:** El valor de l'offset en qüestió s'incrementa fins on desitgi l'usuari.

   6.2. **Decrease:** El valor de l'offset en qüestió disminueix fins on desitgi l'usuari.

   6.3. **Fluctuant:** El valor de l'offset en qüestió fluctua entre els límits que vulgui l'usuari. Ho fa de manera aleatòria.

7. **Error [%]:** L'usuari indica el % que vol que l'offset incrementi o disminueixi. Pel cas de fallada fluctuant, el valor de l'offset fluctuarà en el marge del ± %.

8. **From [s]:** Moment en el que l'usuari vol que s'iniciï la fallada (en segons).

9. **To [s]:** Moment en el que l'usuari vol que s'aturi la fallada (en segons).

   Nota: Si el interval entre el inici i l'aturada de la fallada és molt gran pot ser que el FS recuperi el valor correcte de l'offset cada vegada que se li intenta

assignar un nou valor referent a la fallada i aquesta no es pugui simular correctament. Això és així ja que, al ser un interval gran, el valor de la fallada no varia gaire vers el valor de l'offset en condicions normals i això fa que el FS pugui recuperar el valor original ràpidament.
També cal tenir en compte que aquest fenomen es produeix per a alguns offsets i no per tots.

10. **Clear/Reset Button:** Esborra tots els camps de la taula i atura la fallada.

11. **Progress Bar:** Indica el percentatge de progrés temporal de la fallada generada.

12. **Start Button:** Una vegada programada la fallada en la taula s'inicia prement aquest botó. Mentre es produeix la fallada es pot prémer per aturar-la i tornar-lo a prémer per reprendre la fallada de nou.

Nota: No es poden superposar temporalment varis errors per a un mateix offset. S'han de programar de manera seqüencial.

# ANNEX B
# MANUAL DEL PROGRAMADOR

# INTRODUCCIÓ

La finalitat d'aquest annex és explicar al programador que disposi del codi i de la autorització corresponent la estructura del codi i les llibreries externes utilitzades. D'aquesta manera espero poder ajudar a desenvolupar un futur millor programa.

En cas de que tot i això el programador tingui algun dubte de com s'ha implementat alguna funció o hagi detectat un error, em poso a disposició per a qualsevol aclariment o reparació que es necessiti del programa: **bernatfontgarcia@gmail.com**

## B.1 Llenguatge i plataforma de programació

El programa ha estat desenvolupat mitjançant el llenguatge C#, que permet combinar el llenguatge C/C++ amb una interfície visual agradable i funcional.

La plataforma que s'ha utilitzat és el Microsoft Visual Studio 2010. Versió 10.0.30319.1 RTMRel de ©2010 Microsoft Corporation.

## B.2 Llibreries externes

Les llibreries són paquets de funcions ja programades i preparades per a utilitzar segons la necessitat del programador que les estigui explotant. Les que s'han utilitzat per a desenvolupar el programa són:

1. **FSUIPC**
2. **MathParser**

A continuació s'explicaran detalladament cada una d'elles.

### 1. FSUIPC

La FSUIPC és una llibreria desenvolupada pel senyor Peter Dowson i que s'acobla al FS com a mòdul extern. Com a trets generals, permet un control personalitzat de l'aeronau introduint canvis en el simulador al gust de l'usuari.

D'altra banda, també interactua amb el FS amb el que s'anomenen **offsets** que no és res més que una variable del FS que s'ha pogut accedir mitjançant algorismes de la pròpia llibreria.

Per instal·lar aquesta llibreria cal descarregar-la de la pàgina:
http://www.schiratti.com/dowson.html

El següent pas és ja la instal·lació en si. És senzilla i guiada. Simplement cal tenir en compte que abans d'instal·lar aquesta llibreria s'ha d'haver executat ni que sigui un vol en el FS per a que es puguin crear els fitxers que requereix la instal·lació.

De cara al programador, cal copiar la .dll que s'inclou en el paquet a la carpeta del. Finalment s'ha de declarar de la següent forma:

```
using FSUIPC;
```

(1)

Nota: Aquests procediments ja s'han executat en el programa que s'adjunta.

Així doncs, el procediment per a crear un nou indicador d'un offset disponible seria el següent:

1.1. **Declarar l'offset**: Per a declarar un offset és necessari serguir el següent format:

```
private Offset<int> altimeter = new Offset<int>("altimeter", 0x3324);
```

(2)

S'ha de tenir en compte quin tipus d'offset és (ho indica en el document d'offsets) per a la seva declaració.
El primer argument ("altimeter") fa referència a la etiqueta que se li posa a l'offset. D'aquesta manera es poden crear llistes de strings i després processar els offsets d'una manera ordenada (s'explica més endavant).
També, però, es podria prescindir d'aquest argument.
El segon argument fa referència a la clau de l'offset que s'utilitza per comunicar-se amb el FS. Cada offset té un número de referència diferent.

1.2. **Afegir a una llista de strings**: Per a ordenar els offsets he creat llistes de strings que contenen les paraules que s'identifiquen amb cada offset. D'aquesta manera, si es vol, es pot processar només aquells offsets que s'identifiquin amb les paraules que conté la llista. En aquest cas afegiríem la paraula "altimeter" en la llista dels offsets del panell esquerra (primer cal declarar aquesta llista).

```
private List<string> left_panel = new List<string>();
.
.
.
left_panel.Add("altimeter");
```

(3)

1.3. **Processar l'offset**: Una vegada ja tenim declarat l'offset només cal processar-lo i obtenir-ne el valor.

```
FSUIPCConnection.Process(left_panel);
int altimeter1 = (int)altimeter.Value;
```

(4)

Com es pot veure, en aquest cas demanem a la llibreria que processi els offest continguts en la llista de string anomenada 'left_panel'. D'aquesta manera sabem quan s'actualitza el valor dels instruments d'aquest panell.

També es podria utilitzar la expressió:

```
FSUIPCConnection.Process();
int altimeter1 = (int)altimeter.Value;
```

(5)

Així la llibreria processaria tots els offsets que s'hagin declarat al llarg del programa.

Si es consulta el document d'offsets es veurà que en aquest cas, no cal aplicar-li cap processat al valor que ens retorna l'offset ja que aquest és directament l'alçada d l'avió en peus.

## 2. MathParser

MathParser és una llibreria destinada a resoldre operacions matemàtiques que han emmagatzemades en una variable string (cadena de caràcters).

La seva funció és analitzar caràcter a caràcter el string i extreure'n la operació corresponent per a poder resoldre-la.

És de codi lliure i es pot descarregar de la següent adreça:
http://www.lundin.info/mathparser.aspx

La seva funció en el programa és resoldre el processat dels offsets que l'usuari ha indicat en la taula d'offsets extres.

Funciona de la següent manera.

### 2.1. HashTable

La llibreria treballa des de una hashtable. Una hashtable és una estructura de dades que associa claus amb valors. D'aquesta manera es pot emmagatzemar els valors de les variables de la equació.

Primer es declara la hashtable:

```
Hashtable h = new Hashtable();
```

(6)

I després se li assigna una clau amb un valor:

```
h.Add("x", ((Offset<byte>)Offsets[i]).Value.ToString());
```

(7)

Ara el valor de l'offset es guarda amb la clau 'x' en la taula hash.

### 2.2. Resoldre la equació

Aquest és el punt en que s'utilitza el parser de la llibreria.
Se li indica quina és la cadena de caràcters que conté la equació matemàtica i a continuació quina és la tala hash amb la que ha de treballar.
El resultat és en format *double*.

```
result = parser.Parse(Row.Cells[5].Value.ToString(), h);
```

(8)

Nota: La cel·la 5 indicada com a 'Row.Cells[5]' és la que conté la cadena de caràcters.

## B.3 Algorismes de fallades

Tal i com s'ha vist en l'apartat 3.2 del cos principal de la memòria, el programa està dotat de tres tipus de fallades:

1. Increment
2. Disminució
3. Fluctuació.

Tant el d'increment com el de disminució funcionen de la mateixa manera. Al iniciar la fallada es guarda el valor inicial de l'offset en qüestió en una taula hash que s'utilitza pe calcula el canvi que s'aplicarà a l'offset per a cada tick del temporitzador del mòdul de fallades. Per exemple:

```
original_value.Add(i, Convert.ToInt32(((Offset<int>)failure[i]).Value.ToString()));
```

(9)

La clau 'i' la proporciona l'índex de la fila de la taula del mòdul de fallades.

Per a generar el % d'error necessari per a cada tick del temporitzador s'utilitza el següent algorisme:

```
percent = (int)original_value[i] * step / 100;
result = Convert.ToDouble(((Offset<int>)failure[i]).Value) + percent;
((Offset<int>)failure[i]).Value = Convert.ToInt32(result);
```

(10)

El ‘step’ és el pas necessari per a cada tick del temporitzador i es calcula de la següent manera:

```
double step = Convert.ToDouble(Row.Cells[4].Value) /
(Convert.ToDouble(Row.Cells[6].Value) - Convert.ToDouble(Row.Cells[5].Value));
```

(11)

Això ve a ser la següent fórmula:

$$step = (Error\ [\%] / t.MAX - t.MIN)$$

Al multiplicar aquesta variable per al valor original (i dividint per 100, per corregir el % de error) obtenim el valor (en el format de l’offset que sigui) que ha d’incrementar o disminuir, segons el tipus de fallada, i es guarda en la variable ‘percent’.

Finalment aquesta variable es suma o resta al valor actual del offset en qüestió i en acaba s’actualitza en seu valor mitjançant:

```
FSUIPCConnection.Process();
```

(12)

Pel que fa a la fallada fluctuant, s’utilitza el següent algorisme:

```
Random rand = new Random();
double random = (rand.NextDouble() * 2 - 1);
percent = (int)original_value[i] * random *
          Convert.ToDouble(Row.Cells[4].Value) / 100;
result = Convert.ToDouble((int)original_value[i]) + percent;
((Offset<int>)failure[i]).Value = Convert.ToInt32(result);
```

(13)

Com es pot observar, primer es declara un nou número aleatori que posteriorment genera un valor entre -1 i 1. Aquest número aleatori es multiplica per al valor original de l’offset i per al error que l’usuari hagi indicat en la taula del mòdul de fallades (finalment es divideix entre 100 per a corregir l’error, que és en %). D’aquesta manera s’obté un valor, positiu o negatiu, i dins del marge indicat per l’usuari com a error, que s’emmagatzema en la variable ‘percent’.

En acabat, aquest valor se li suma al original de l’offset i s’actualitza com s’ha vist anteriorment.