Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Diseño del software de alto nivel para una aplicación multisensorial

Autor: Andy Martín Ramos

Tutor: MSc, Alain Sebastián Martínez Laguardia

Santa Clara

2008

"Año 50 de la Revolución”

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Diseño del software de alto nivel para una aplicación multisensorial

Autor: Andy Martín Ramos

martin@uclv.edu.cu

Tutor: MSc. Alain Sebastián Martínez Laguardia

Profesor Auxiliar

Departamento de Automática y Sistemas Computacionales

Facultad de Ingeniería Eléctrica

E-mail: amguardia@uclv.edu.cu

Consultante: Dr. Luís Hernández Santana.

Santa Clara

2008

"Año 50 de la Revolución”

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Autor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Intente lo último, y nunca acepte la duda.

Nada es tan duro, y la búsqueda lo encontrará.

Robert Herrick

DEDICATORIA

A mi madre y demás familiares por la educación que me han dado, por las fuerzas que me han brindado para llegar hasta aquí y para seguir superándome tanto espiritual como profesionalmente. A mis amistades por apoyarme en todos los momentos difíciles que e tenido a lo largo de todos estos años.

AGRADECIMIENTOS

Quiero agradecer a mi tutor Alain Martínez y al Dr. Luís Hernández por la ayuda que me han brindado en la realización de este trabajo, también agradecer a todos los profesores de la carrera. Agradecer a todos mis familiares en especial a mi madre y a mi tío que ha sido un ejemplo a seguir a lo largo de mi vida. No pueden faltar mis amistades por el apoyo que me han brindado.

TAREA TÉCNICA

-Plantear la arquitectura de software de alto nivel para un vehículo autónomo.

-Programar los módulos del supervisor que interactuaran con el UAV para mostrar los parámetros fundamentales de operación en tiempo real.

Firma del Autor

Firma del Tutor

RESUMEN

Gracias a los grandes avances en la aeronáutica y en las telecomunicaciones, en nuestros días se puede, por raro que parezca, que unos aviones sin tripulación alguna y sin ser teledirigidos vuelen a través de la atmósfera de la tierra; es decir, por sí mismos. A este tipo de vehículo se los conoce como UAV, acrónimo proveniente del inglés que significa vehículos aéreos no tripulados. Esta es una tecnología que ya está desarrollada y se viene implementando hace ya algunos años con fines militares, y más recientemente en aplicaciones civiles.

Es de gran interés aprovechar esta tecnología y desarrollarla aun más para conseguir dar a estos vehículos nuevas aplicaciones que puedan ser aprovechados por nuestra sociedad y contribuyan al ahorro de recursos. Dentro de sus usos más frecuentes se encuentran la exploración y la supervisión.(Rosas 2007)

Por tanto el objetivo fundamental de este proyecto es contribuir con el desarrollo de un UAV nacional para la toma de imágenes que con un procesamiento posibilitaran la toma de decisiones. En concreto diseñar e implementar la aplicación de alto nivel para un sistema multisensorial con caso de aplicación vehículo autónomo, con núcleo en la visualización y creación de un entorno de los datos brindados por los distintos sensores que componen el sistema.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Actualmente, la tecnología UAV (Unmanned Aerial Vehicle) un tipo de robot de servicio, está siendo desarrollada por un gran número de empresas a nivel mundial y pretende ser un punto de ruptura en la evolución de la aviación militar y civil. Su principal característica es la de realizar vuelos sin tripulación a bordo, permite abarcar un amplio abanico de aplicaciones sin poner en peligro la vida de seres humanos.(Valdivia 2006)

En el campo de la robótica se han desarrollado vehículos terrestres autónomos, robots submarinos sin tripulación, y en estos momentos han tomado un mayor auge los vehículos de vuelo autónomo. Muchas instituciones investigativas, así como importantes universidades del mundo, se han dedicado al desarrollo de estos vehículos por su importancia en la realización de tareas que pueden ser extremadamente difíciles o peligrosas para vehículos tripulados. (Vélez 2000)

Entre los vehículos de vuelo autónomo una de las plataformas más utilizada en la actualidad es el avión. Esto se debe a sus cuantiosas capacidades, y sus habilidades para poder moverse en ambientes de difícil acceso donde se requiere de una buena maniobrabilidad.

En el sector militar, los aviones de bajo coste sin tripulación se están preparando y desarrollando para misiones de reconocimiento, de vigilancia urbana, la búsqueda y rescate, entre otras aplicaciones.(Barrientos 2007)

En nuestro proyecto general la tarea consiste en la toma de fotos aéreas de determinadas zonas geográficas y para llevarla a cabo se opto por un UAV.

Debido a la tarea fundamental del proyecto general no esta de más pensar en el por que se pensó en un avión y no en un satélite; y entre una de dichas razones se encuentra el gasto

económico que traería consigo la implementación de un sistema con un satélite, las imágenes satelitales son costosas, no están siempre disponibles en el momento que son necesarias y por lo general abarcan un área mucho mayor a la de interés, sobre todo en entornos como el cubano donde nuestra geografía es alargada y estrecha .

Otra opción seria la de emplear un vehículo aéreo tripulado, opción actualmente empleada con un gasto considerable de medios, tripulación y combustible superior a varios miles de dólares por vuelo.

Por tanto con el desarrollo de este trabajo se ampliaran las capacidades de operación y las posibilidades de servicio de varias empresas nacionales.

Los UAV requieren de un sistema supervisor que recibe la información y los datos de interés, así mismo le pasa la planificación de tareas al vehículo. Esa interface de usuario será el objeto de trabajo de nuestro proyecto.

Para el desarrollo de este trabajo se han necesitado conocimientos de asignaturas cursadas durante la carrera como lo es la programación en C/C++. También se han necesitado algunos conocimientos de programación grafica en OpenGl.

Con el propósito de dar cumplimiento a los objetivos propuestos se plantean las siguientes actividades:

- Revisión bibliográfica.
- Análisis de requerimientos.
- Desarrollo de la arquitectura del software para el sistema.
- Implementación de la aplicación (versión 1.0).
- Análisis de los resultados y pruebas de la aplicación.
- Mejoras sugeridas para la próxima versión.
- Confección del informe final.

La planificación del proyecto se ha realizado para seis meses como la generalidad de los trabajos de fin de carrera. El trabajo de diploma en su conjunto está dividido en tres

capítulos, antecedidos por la introducción del tema, llegándose al final de cada capítulo a determinadas conclusiones y al final del trabajo se sugieren recomendaciones que resultan de consulta obligada para posteriores desarrollos.

En el primer capítulo se comienza con una revisión bibliográfica y análisis del problema; se mencionan algunas de las aplicaciones de los UAV, sus ventajas, se sitúa el problema haciendo un análisis de diferentes trabajos realizados y se propone una solución.

En el segundo capítulo se realiza un análisis del diseño de la aplicación, como debe quedar visualmente la interfaz de usuario, análisis de los casos de uso y se representa el diagrama de bloques de la aplicación.

Por ultimo, en el tercero se plantea el resultado de nuestro trabajo, comentamos acerca de las especificaciones que se debe cumplir a la hora ejecutar la aplicación y un análisis económico de la misma.

El informe final servirá de manual de usuario y servicio de la aplicación.

CAPÍTULO 1. OBJETIVOS Y ESTADO DEL ARTE

En los últimos años, el ser humano ha contado con las herramientas necesarias para sustituirle de aquellas tareas que resultan repetitivas y peligrosas por sistemas automatizados que permiten alcanzar un mayor rendimiento, aumentan la productividad y ofrecen mayor seguridad. Unos de esos ejemplos son los vehículos aéreos autónomos.

Un vehículo aéreo autónomo se define como un vehículo capaz de desplazarse a través de la atmósfera por sí mismo, desempeñando misiones previamente programadas. Se pueden diferenciar dos grupos de vehículos aéreos según su grado de autonomía; los completamente autónomos, que son vehículos capaces de realizar tareas en ambientes no estructurados sin la ayuda de un operador humano y los sistemas híbridos, en los que se utiliza la ayuda del operador humano para solventar ciertas situaciones a través de la tele operación. (Hernández 2006)

En este capítulo se pondrán de manifiesto algunas de las aplicaciones de los vehículo aéreos autónomos y se realizará un estudio sobre la tarea que se nos asigno en el proyecto que está relacionada con la elaboración de una interfaz de supervisión, la cual ha sido desarrollada por diferentes centros de educación superior; viendo las particularidades que distinguen a cada una de ellas y que lo convierten en una herramienta de considerable importancia en tareas en las cuales se requieren elevados gastos de recursos y que pueden ser realizadas por estos vehículos disminuyendo los costos concernientes, tanto económicos como humanos. Ofreceremos además una panorámica del proyecto en general

1.1 Motivación: antecedentes, y motivación personal

En la actualidad el Grupo de Automatización, Robótica y Percepción de la UCLV se encuentra enfrascado en una serie de proyectos relacionados con el desarrollo de varios

prototipos de UAV empleando aviones y helicóptero como plataformas. Estos proyectos son realizados en la UCLV, en colaboración con las empresas CEDAI y GEOCUBA con el fin de diseñar una nueva herramienta de exploración basada en aviones UAV. La idea es volar un vehículo aéreo no tripulado que desempeñe una misión sobre el terreno tomando imágenes y transmitiéndolas a tierra para su procesamiento

Con el empleo de un UAV, conseguiremos obtener información útil de un objetivo de forma precisa, rápida y sin peligro para las vidas humanas. Es un proyecto de aproximadamente 1 año de duración en el que se tocan diversos temas relacionados con la ingeniería en automática: desde la identificación matemática hasta la implementación del sistema empotrado abordo. Una de estas tareas es la creación de un diseño de la estación en tierra.

Para ello el proyecto se divide en distintos grupos de estudiantes que realizan diferentes tareas.

Teniendo en cuenta la envergadura del proyecto, y coincidiendo con el final de mi carrera, me ha parecido interesante unirme a este grupo de investigación con el fin de realizar mi trabajo de fin de carrera.

De esta manera comienza una etapa de análisis de propuestas, especificación y diseño de la aplicación de la estación de tierra.

1.2 Aplicaciones de los UAV

Los vehículos aéreos autónomos pueden realizar gran variedad de tareas dentro de las que se encuentran:

1.2.1 Supervisión de límites fronterizos o áreas de seguridad

Haciendo uso de vehículos aéreos se realizan tareas para vigilar las zonas de paso entre países con fronteras. En la actualidad estas tareas de supervisión son realizadas por los cuerpos militares del estado, requiriendo gran número de personal y recursos. Los helicópteros y aviones tripulados empleados en estas tareas tienen un alto costo de explotación y mantenimiento. La utilización de vehículos aéreos no tripulados de pequeño

tamaño con sensores y cámaras incorporados a la aeronave que permitan capturar información, permitiría realizar la inspección desde una estación segura ubicada en tierra.

1.2.2 Reconocimiento y evaluación de daños en zonas de desastres y contaminación.

Debido a la gran maniobrabilidad y poco tamaño de este tipo de aeronaves se puede llegar a sitios de difícil acceso y evaluar zonas de desastres, donde sería mortal para el ser humano realizar cualquier tipo de actividad, como es el caso de zonas contaminadas por radiación o agentes químicos.

Este tipo de aeronaves no tripuladas podrían utilizarse en zonas donde se requiere de una evaluación rápida y sin riesgos de pérdidas humanas, como es el caso tras el paso de huracanes, crecidas de ríos, deslizamientos de tierra y terremotos.

1.2.3 Inspección de tráfico terrestre en autovías.

A través de la inspección en autovías se puede conocer con exactitud las zonas donde existe una mayor densidad de vehículos, detectar las causas de los atascos y en caso de haber accidentes, precisar el lugar donde ocurren y la magnitud de los mismos, enviando la información a la estación ubicada en tierra.

1.2.4 Agricultura de precisión.

La utilización de pequeños vehículos aéreos para realizar fotografías de áreas de cultivo con el fin de evaluar crecimiento, infestación por malas hierbas y maduración, o para realizar estimados de cosecha, son viables con este tipo de equipamiento a un costo razonable permitiendo obtener información de alto valor en el momento deseado de acuerdo a la logística de la cosecha. De igual forma, el tener el control de acciones como la fumigación controlada de pequeñas áreas es viable; trabajo realizado en la actualidad a mano o con avionetas que esparcen por igual los pesticidas en toda el área de cultivo con el consiguiente sobre gasto y contaminación del medio ambiente.(Barrientos 2007)

1.2.5 Levantamientos topográficos y cartografía.

El actual desarrollo de los sistemas GPS que muestran precisiones extremadamente altas, así como el avance de la fotografía y el video digital, permiten avanzar en un nuevo tipo de cartografía digital con procesamiento 3D de áreas de difícil acceso o áreas de las que se

desea una cartografía de muy alta resolución. Esto se logra con el vuelo estático a baja altura de las zonas en cuestión y fotografías en varias gamas del espectro.(Barrientos 2007)

1.2.6 Inspección y detección de fallos en carreteras, puentes y represas.

Debido a las grandes prestaciones de los vehículos aéreos autónomos, como lo son el acceder a zonas de difícil acceso en un lapso de tiempo breve, las obras de mantenimiento de grandes obras civiles como represas, puentes y carreteras, pueden ser llevadas a cabo de manera segura y efectiva con este tipo de aeronaves.

1.2.7 Lucha contra incendios forestales.

Los UAVs pueden ser aplicados en un gran número de actividades relacionadas con la lucha contra incendios, como ha sido resaltado por numerosos autores. Son útiles para la obtención de mapas de riesgo, o de vegetación, empleando para ello distintos sensores. Los UAVs también pueden ser utilizados para actividades post-incendio tales como supervisión de cenizas activas, estimación de daños y de áreas quemadas mediante mapas de alta resolución.(Ollero 2004)

1.3 Instrumentos situados en un vehículo aéreo autónomo.

Los vehículos aéreos autónomos tienen que ser equipados por un conjunto de instrumentos que les permitan interactuar con el entorno, lograr su posicionamiento referenciado, todos estos instrumentos estarán interconectados a un medio de cómputo capaz de desarrollar el sistema de control. En resumen, como elementos a bordo por lo general deben poseer: sistema de posicionamiento, unidad de cómputo, módulos de comunicación, altímetros, radares y medidores de condiciones atmosféricas y de operación. (Hernández 2006).

Una arquitectura general podría ser la mostrada en la figura 1.1.

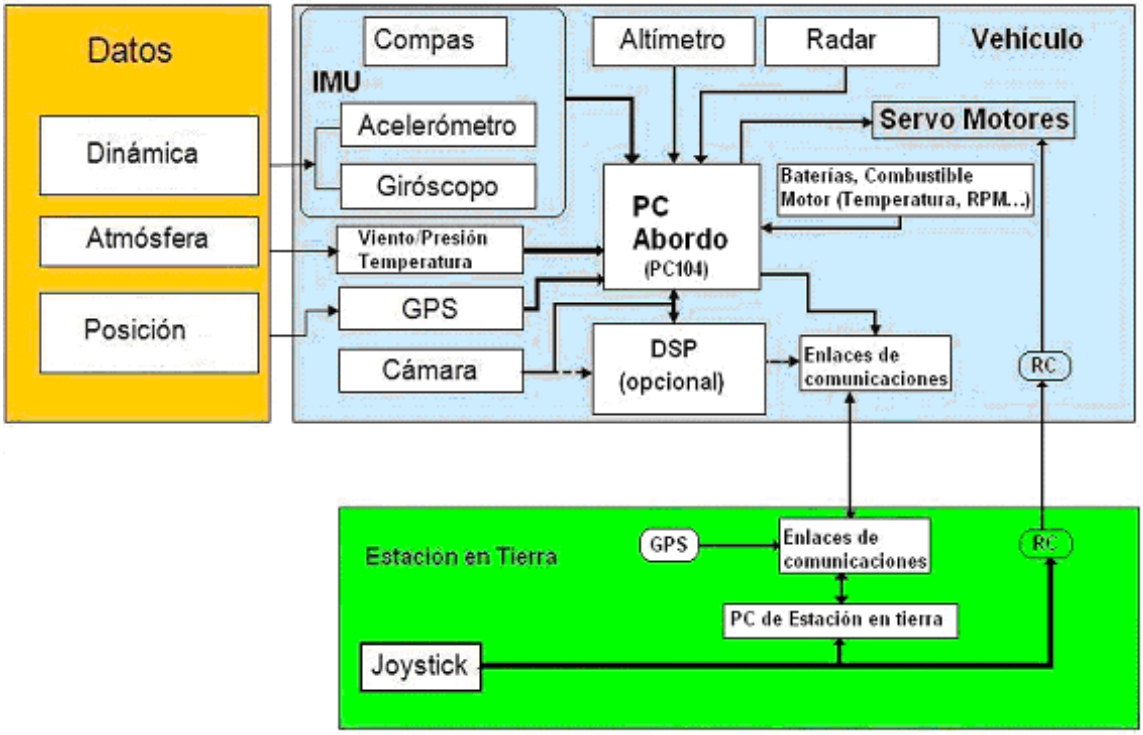


Figura 1.1 Esquema de una Arquitectura General

Como se puede apreciar el sistema está dividido en dos grandes segmentos el aéreo y el terrestre. El primer segmento esta constituido por el vehículo y toda la electrónica a bordo y el segundo constituido por la estación base en tierra. Ambos segmentos tienen tareas muy claras en el marco de la aplicación. La estación terrestre constituye el eslabón más alto en el marco de la planificación de tareas, la supervisión y el control de alarmas.(Laguardia 2008)

1.3.1 IMU.

Dentro de los sensores a bordo podemos destacar la IMU (Unidad de Medición Inercial), este es el sensor principal del sistema de control. Está compuesto por sensores de actitud basados en giroscopios que puede ser mecánicos, piezo eléctricos, u ópticos. Contiene tres plataformas con un giroscopio que mide las proporciones angulares a lo largo de todos los ejes del vehículo, también proporciona acelerómetros que pueden usarse en la estimación de la posición. La actitud y posición pueden estar integradas en la IMU. Otra buena manera

de medir su comportamiento es usando magnetómetros para determinar la actitud de un vehículo midiendo el campo magnético de la tierra. (Fernández 2005)

1.3.2 GPS.

El GPS (Global Position System). Este sistema de navegación tiene la habilidad de determinar la posición actual de un vehículo dada en longitud, latitud y altura. La alta exactitud, simplicidad, y disponibilidad del sistema GPS permite que sea usado en la aviación. La mayoría del los UAVs utilizan GPS como su sensor de navegación primario. En algunos puntos su funcionamiento y precisión pueden estar sujetos a las condiciones atmosféricas. Por lo general son sistemas de referencia aceptables para muchas aplicaciones logrando precisiones métricas y submétricas con sistemas diferenciales.(Fernández 2005)

El sistema GPS permite que la localización de cada usuario la conozca solamente el usuario debido a que no se emite ningún tipo de señal, por lo que la privacidad del servicio se garantiza. (Laguardia 2008)

1.3.3 Sistema de mando manual

El sistema de mando manual esta compuesto por un emisor que como su nombre lo indica, es la unidad que mediante ondas de radio moduladas transmite al receptor situado en el avión las órdenes que se le envían con los movimientos de los controles de la emisora.

El receptor es el elemento que situado en el interior del avión, recibe las señales codificadas del transmisor; este las descodifica y envía los mandos al servo adecuado para que actúe.

Los servos son unos dispositivos que convierten las señales transmitidas por el emisor en movimiento.

Por su parte las baterías son los elementos que proporcionan la energía necesaria para hacer funcionar todo el conjunto del equipo de radio.

1.3.4 Enlaces de datos

Para enviar órdenes y recibir información del UAV se usan enlaces de datos inalámbricos, estos enlaces pueden ser divididos en digitales y analógicos. Un ejemplo de transmisión analógica es el transmisor de UHF que transmite el video en tiempo real. Los enlaces

digitales proporcionan una manera de comunicación entre tierra y la computadora colocada en el vehículo. El ancho de banda del MODEM y la frecuencia a la que opera estarán dados por el volumen de datos a transmitir y la distancia a la que se quiera operar. Típicamente en la banda de 900 MHz se alcanzan las mayores propagaciones en espacio libre y por eso a sido seleccionada esta frecuencia, con una velocidad de transmisión de 19200bps.

1.4 Ventajas y desventajas de los (UAV)

Como principal ventaja de estos vehículos debemos resaltar la posibilidad de recoger información en ambientes peligrosos sin el riesgo de perder o herir las tripulaciones de vuelo. En estas tareas y en otras han demostrado ser más eficiente, ya que utilizan tecnologías de punta, diseñadas específicamente para cada aplicación.(Fernández 2005)

Los Vehículos Aéreos sin tripulación han potenciado el desarrollo aplicaciones civiles tales como la gestión de situaciones de alarma, la inspección en sitios no accesibles, vigilancia, y búsqueda y rescate. La maniobrabilidad de los UAV les permite obtener mejores vistas que los medios aéreos tradicionales para determinadas circunstancias.

1.4.1 Ventajas Medio Ambientales

Estos vehículos son utilizables en zonas de valor ecológico ya que evitan discurrir por dichas zonas con vehículos terrestres. Menor emisión de ruido, menor impacto durante la inspección. Al consumir poco combustible reducen la emisión de gases que producen el efecto invernadero. Al tener menor tamaño reducen el impacto avifauna y disminuyen las consecuencias en caso de accidente. (Mata 2006)

1.4.2 Desventajas

La desventaja principal que puede tener un UAV es que alguna falla entre las estaciones, tanto en la de tierra como en la de aire o cualquier fallo no apreciable en alguno de los sensores o parte del sistema puede provocar la caída del vehículo lo que implica el uso de rigurosas técnicas de tolerancia a fallos.

1.5 Escenario

Nuestro UAV cuenta con un sistema empotrado a bordo para garantizar el control y seguimiento de trayectorias, y una estación terrestre con una aplicación de supervisión.

El esquema que mostraremos a continuación comprende la concepción de nuestro sistema

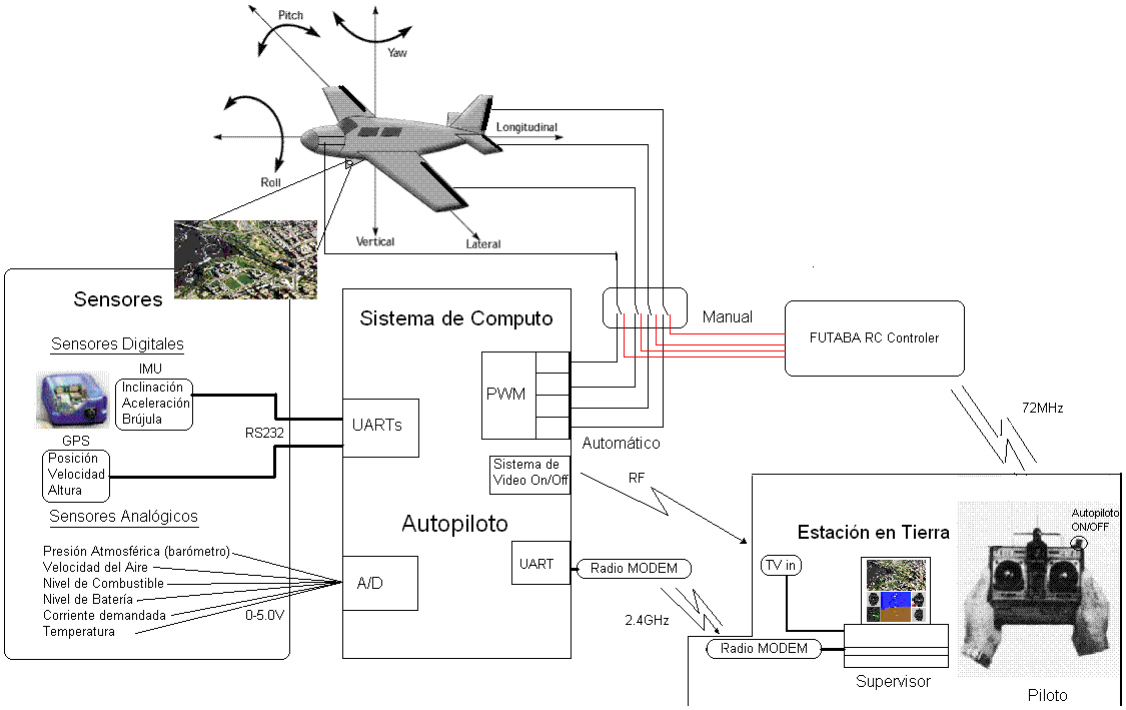


Figura 1.2 Concepción General del Sistema

La plataforma embarcada en el UAV constará de un sistema de control de vuelo similar al mostrado anteriormente en la figura 1.2 para la parte de hardware y con un software de control. Este sistema se dedicará a lograr la misión con todas las acciones sobre el payload, ya sean cámaras de fotografías, de video u otros instrumentos.

1.6 Objetivos

1.6.1 Objetivo General

El objetivo inmediato para el proyecto es la creación de un sistema empotrado que permita implementar un UAV con su estación supervisión y monitoreo. Este prototipo se presentará en la Habana en Julio de este año.

1.6.2 Objetivos específicos

- Análisis de requerimientos de la aplicación.
- Arquitectura de Software necesaria.
- Diseño y construcción del hardware del sistema empotrado que irá a bordo de la aeronave.
- Diseño del controlador.
- Implementación de la base de datos que agrupara las variables de interés tanto a bordo como en la estación en tierra para el uso por el sistema de control y de supervisión.
- Desarrollo del sistema empotrado.
- Diseño e implementación de una interfaz grafica de supervisión que irá en tierra para mostrar los datos adquiridos por los diferentes sensores que intervienen en el proceso.

1.6.3 Objetivo especifico de este trabajo dentro del sistema

Diseñar un sistema de supervisión para un UAV.

- ¿De que se ocupa la supervisión? Es la actividad de apoyar y vigilar la coordinación de actividades de tal manera que se realicen en forma satisfactoria.

Por lo tanto en nuestra aplicación, el supervisor es el encargado de mostrar datos como son la posición del avión, su altura, velocidad, nivel de baterías, nivel de inclinación con respecto a tierra, etc. Debe contar con un planificador de misión; es decir se le debe señalar las trayectorias a seguir; además debe tener alguna forma de avisar cuando exista un fallo o desperfecto en el vuelo.(Giner 2007)

Como principal problema a resolver tenemos el desarrollo de un software de supervisión que cumpla con las características anteriormente señaladas, además debe ser un software en el cual las herramientas que se usen para su desarrollo sean de fácil adquisición; debe tener la flexibilidad de que al emigrar de plataforma para realizar algún cambio no se tenga que convertir el código que es un problema que sucede con frecuencia en ciertas aplicaciones que las hace en muchas ocasiones dependientes de un sistema operativo único.

1.6.4 Objetivos individuales

- Referenciar cartográficamente la posición y trayectoria del UAV.

Visualizar la situación del vehículo en un mapa dadas diferentes coordenadas de latitud longitud.

- Referencia inercial del UAV.

Visualizar en un horizonte artificial los parámetros correspondientes al estado del vehículo como lo son el alabeo, el cabeceo y la guiñada.

- Referencia Magnética del UAV.

Representar la orientación del vehículo con respecto al norte magnético.

- Referencia de las variables atmosférica que inciden en el UAV.

Representar las diferentes variables atmosféricas como lo es la velocidad del viento, presión atmosférica, etc.

- Referencia de los parámetros tecnológicos del UAV.

Representación de los parámetros tecnológicos como es el nivel de combustible, temperatura, revoluciones por minuto del motor, nivel de baterías.

Existen otro objetivo no menos importantes como lo es crear un trabajo de fin de carrera de interés, con buenos resultados, que sea el principio de una serie de investigaciones que se sigan desarrollando sobre el tema. Al margen del trabajo final de carrera, están los intereses propios de la persona. Los míos son conocer más acerca del mundo profesional de un ingeniero y el trabajo en equipo donde hay que cumplir con ciertas y determinadas normas.

1.7 Trabajos Relacionados

A continuación mostraremos algunos proyectos realizados por diferentes centros de educación superior donde se implementan algunos sistemas de supervisión con características similares apoyados en diversas herramientas de diseño.

1.7.1 Proyecto SKY-EYE

Este proyecto fue llevado a cabo por la universidad politécnica de Cataluña en el año 2007. El trabajo realizado estudia y desarrolla un sistema modular embarcado en un avión no tripulado (UAV). El sistema embarcado esta formado por módulos intercambiables de control de vuelo, de misión y captura de información. Los módulos de captura de información podrían ser cámaras fotográficas de espectro visible, cámaras de vídeo, cámaras térmicas por infrarrojo, etc. En tierra hay dos tipos de estaciones, una estación de control para cada zona en la que actúe una flota de UAVs y una estación central de seguimiento conectada a cada una de las estaciones de control. En el proyecto las aplicaciones consideradas son el control de incendios y la vigilancia de grandes infraestructuras.(Rosas 2007)

-Estación de control

Estación de control encargada de controlar la flota y establecer las comunicaciones con cada uno de los UAV figura 1.3.

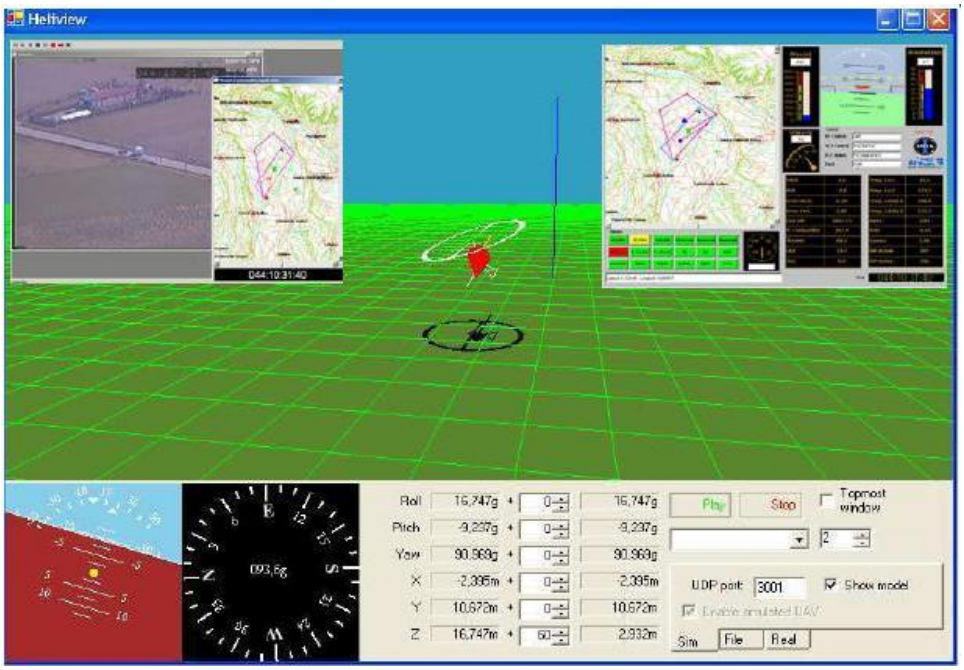


Figura 1.3 Ejemplo de una Estación de Control

-Estación de seguimiento

Esta es la estación de seguimiento que se podrá situar en cualquier punto del territorio, es única y todas las estaciones de control deberán conectarse a ella. Para la conexión hacia la estación de seguimiento se utilizarán conexiones de red de comunicación convencionales. En esta estación se dan los planes de seguimiento juntamente con los planes de vuelo, misión y recepción de información. Para esta estación se diseña una aplicación de software con diferentes funciones, ha de ser capaz de tratar toda la información recibida, poderla visualizar en múltiples pantallas simultáneamente. En esta Interfaz de usuario se crea una capa de presentación, con acceso a cada gestión de la aplicación. Este diseño tiene en cuenta el aspecto visual de los sistemas de información geográfica y de parte de un control de vuelo Figura. 1.4.

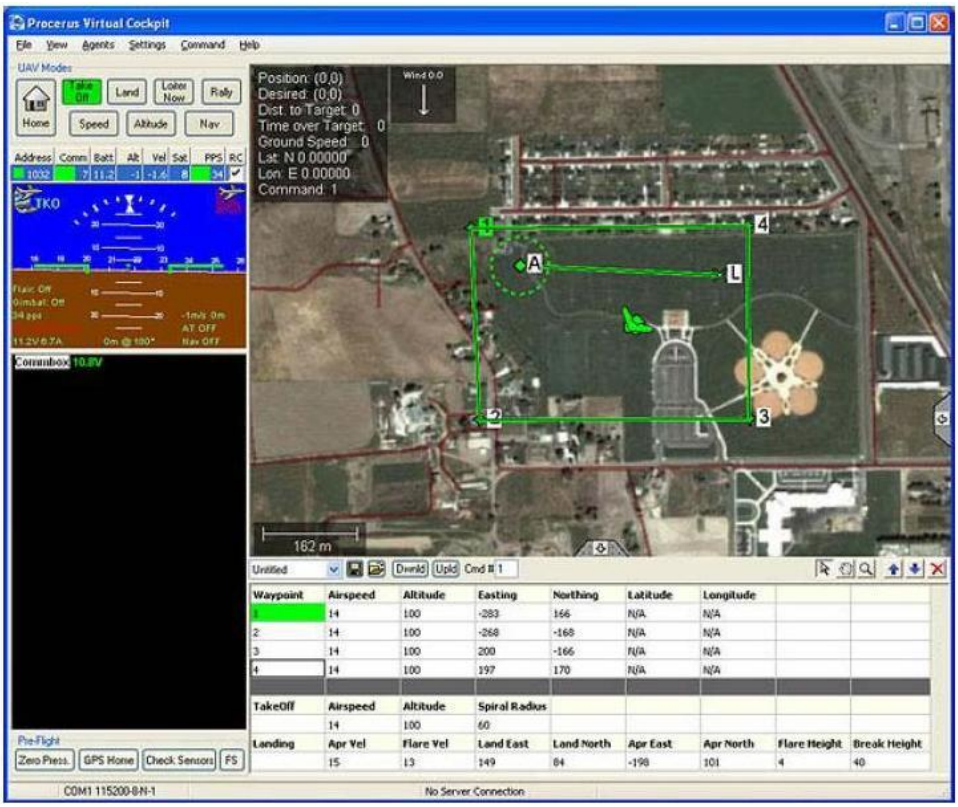


Figura 1.4 Ejemplo de una Estación de Seguimiento

1.7.2 Trabajo de fin de carrera de la Universidad de Monash

Trabajo de fin de carrera de la Universidad de Monash (Australia) en el año 2006, donde se muestra un sistema de supervisión basado en una estación terrestre de control, fue realizado usando Glade. La estación recibe mensajes acerca de la posición del vehículo y la altitud. El software es capaz de describir la posición del avión cuando un usuario señala el mapa Glade-2 es un paquete de software de código abierto usado para crear interfaces gráficas de usuario. También proveen la funcionalidad para la manipulación de imágenes y otras herramientas necesarias para el software, soporta código en C++ .Presenta funciones llamadas callback que se encargan de controlar acontecimientos, por ejemplo cuando se pulsa un botón o se da un clic, los callback son capaces de gestionar el redimensionamiento de una o varias ventanas. (Fernando 2006)

En la ventana el mapa y la altitud se ven claramente mostrados. Estos dos componentes muestran la parte de mayor peso en el software debido a que a medida que el vehículo se va

desplazando se tendrá que dibujar constantemente los puntos recorridos o waypoint como se muestra en la figura1.5.

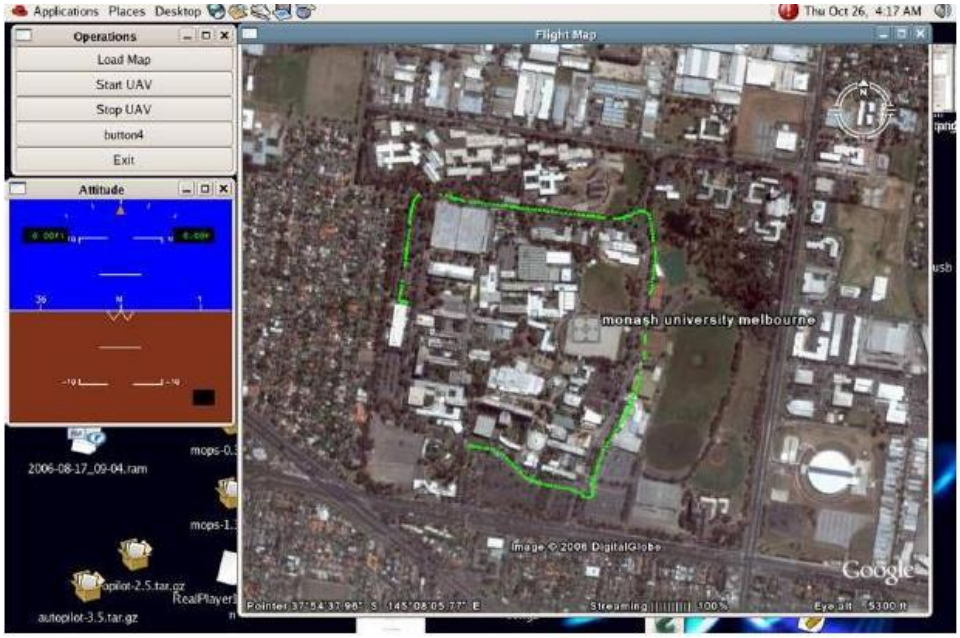


Figura 1.5 Interfaz realizada con Glade-2

1.7.3 Trabajo desarrollado por el grupo QSS Inc.

En este trabajo se propone un supervisor que contiene un mapa, visualización 3D. Los comandos se ejecutan desde una ventana cmd.

Los waypoints pueden ser manipulados en una representación 2D del área. Los waypoints son cubiertos con una capa en imágenes localizadas en GPS 2D bitmap, Una vez que una misión es programada el usuario puede enviar las órdenes para la aeronave. (Ippolito 2005)

La figura muestra una estación terrestre con visualizador 3D.

Figura 1.6 Estación terrestre con visualizador 3D

1.7.4 Trabajo desarrollado con Matlab/Simulink

A continuación mostraremos una interfaz de usuario donde se muestra una plataforma apropiada para la simulación y el monitoreo de las señales provenientes de los sensores. La plataforma es fácilmente configurable para diversas pruebas de hardware y de software. Se seleccionó Matlab/Simulink pues permite integrar fácilmente el control con la lectura de sensores y los actuadores. Simulink agiliza el desarrollo de prototipos y su simulación en tiempo real es una de las características principales. La integración de la plataforma con el target (equipo de cómputo que ejecutará la lectura de sensores, control y actuación) Para la comunicación con el target se usaron tecnologías estándares como Ethernet, y los protocolos UDP/IP y TCP/IP para transferencia de código y monitoreo. Una WLAN ad-hoc se forma con la computadora de vuelo del helicóptero y uno o más computadores portátiles que desempeñan el papel de estación de tierra o estaciones de supervisión. La telemetría durante el vuelo es de gran importancia debido a que las pruebas de vuelo requieren una vista actualizada del estado del vehículo. (Agudelo. 2007)

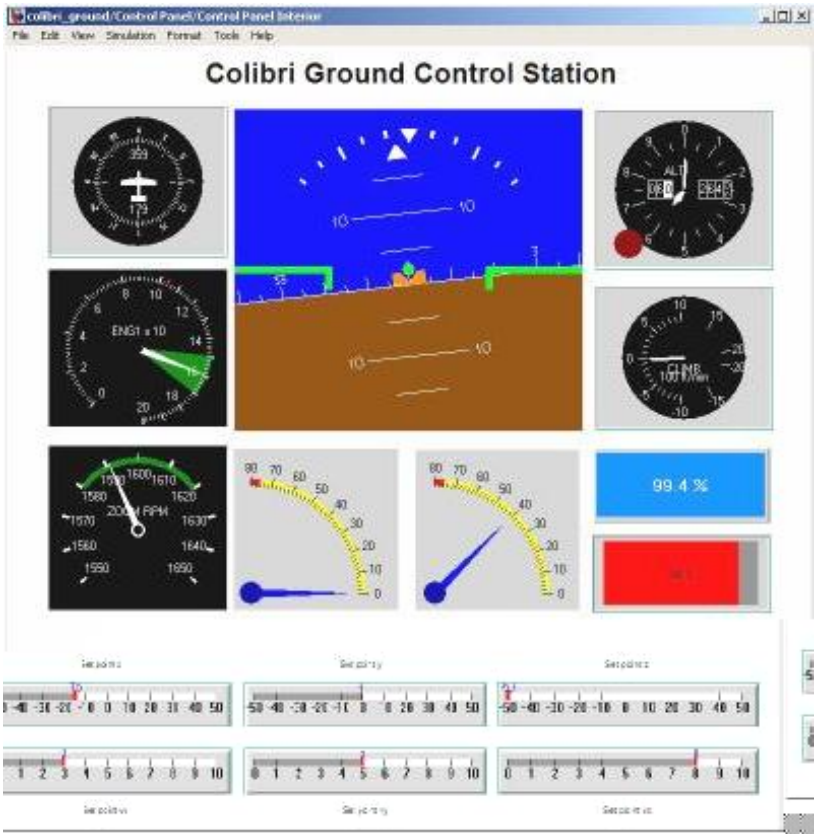


Figura 1.7 Interfaz desarrollada con Matlab/Simulink

1.7.5 Proyecto de la Universidad Politécnica de Madrid

Este visualizador que mostraremos a continuación ha sido creado basado en las librerías SMR (Simulación Modular de Robots). (Giner 2007)

La aplicación se ha desarrollado para ejecutarse sobre una plataforma Win32, utilizando un API de Windows para programar el sistema de ventanas de la interfaz.

Toda la parte de simulación gráfica ha sido programada con ayuda de las librerías "OpenGl" y la librería auxiliar estándar "Glux". Se incluyen por tanto aspectos que dotan de gran realismo a las escenas simuladas como definición de los materiales, efectos ambientales como niebla, cámaras y luces completamente configurables.

En el caso del simulador que se presenta, se ha utilizado controles ActiveX, que posibilita la visualización de los entornos simulados en cualquier plataforma Win32. Igualmente se ha diseñado una serie de entornos para simulación de diferentes misiones.

La aplicación que se ha desarrollado para ejecutarse sobre una plataforma Win32, utilizando la API de Windows para programar el sistema de ventanas de la interfaz que se muestra en la figura.

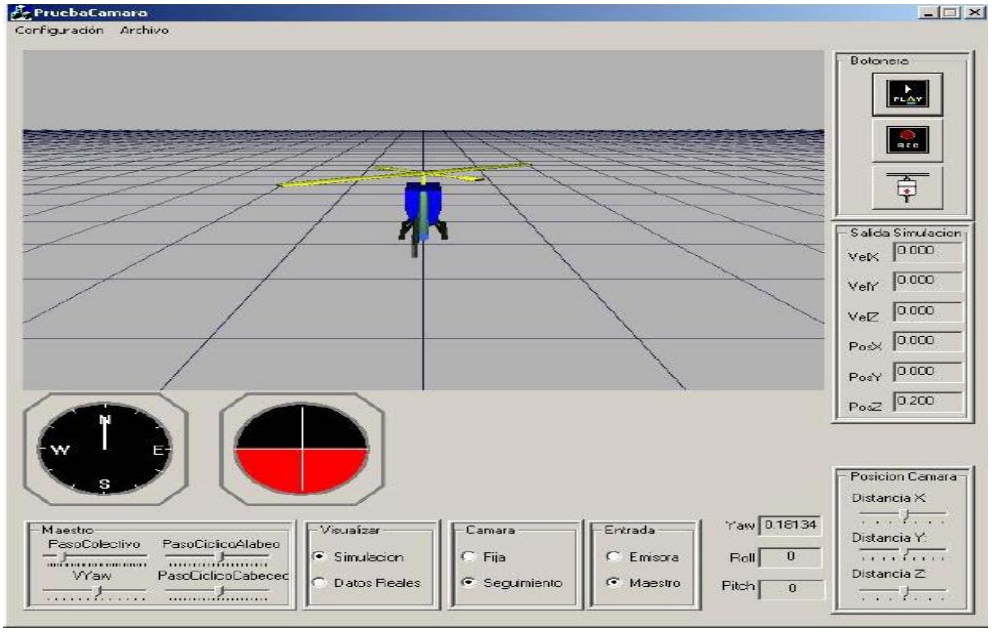


Figura 1.8 Interfaz desarrollada sobre Win32 y OpenGl

1.8 Propuesta de diseño

Para resolver esta tarea es necesario implementar un software de alto nivel que cumpla con los requisitos mostrados con anterioridad en el comienzo de este Capitulo y que además sea de código abierto, y que su desarrollo no dependa de herramientas propietarias.

Es por ello que se pensó en realizar el código en C/C++ para lo que respecta el trabajo con objetos debido a que este lenguaje brinda una gran potencialidad a la hora de la encapsulación de los datos que hace que el desarrollo de los programas sean mas entendibles y eficientes. Otra de las razones por las cuales se escogió el lenguaje C fue para crear un diseño de software homogéneo en el sentido de su fácil integración a otros sistemas ya desarrollados en C, como es la aplicación que se encarga de leer los datos de los sensores y almacenarlos en una base de datos. También se hará uso de una API grafica llamada OpenGl (Open Grafic Libreries) también codificada en C encargada de dibujar

todo el escenario de supervisión y una pequeña biblioteca que también puede utilizarse llamada MUI (Micro Users Interfaces), para lo que respecta el uso de botones y barras de desplazamientos al estilo que todos o la gran mayoría de los usuarios estamos acostumbrados a usar; aclarar que no se debe utilizar controles ActiveX, ni herramientas que dependan totalmente de Windows.

1.9 Consideraciones finales del capítulo

Se ha realizado una explicación de las principales aplicaciones de los vehículos de vuelo autónomo, sus potencialidades y posibilidades de desarrollo en nuestro país. Además, se han referenciado los instrumentos necesarios para el funcionamiento de este proyecto. Con posterioridad se detallaron los objetivos y algunas de la interfaces de supervisión que se emplean en estos casos y planteamos nuestros paradigmas de diseño del software.

En el próximo capítulo se realizará una descripción relacionada con el diseño de la aplicación, con el objetivo de analizar su funcionalidad y casos de uso, aspecto visual de la aplicación, diagrama de flujo de datos, etc.

CAPÍTULO 2. DISEÑO DEL SOFTWARE

En este capitulo hablaremos acerca de las plataformas de desarrollo utilizadas en el proyecto, los indicadores que serán representados en las ventanas de visualización de la aplicación, el porque son ellos y no otros los que se toman en cuenta a la hora de una representación grafica. Mostraremos la funcionalidad y casos de uso de la aplicación de supervisión, así como un diagrama de flujo general desde que los datos llegan a la computadora hasta que se muestran por pantalla.

2.1 Plataformas de desarrollo

En este epígrafe trataremos sobre la plataforma de desarrollo del trabajo, el Visual Estudio 6.0 de conjunto con la librería grafica OpenGl. Esta plataforma de desarrollo de Microsoft brinda grandes posibilidades a la hora de construir aplicaciones, además de ser de fácil utilización para el usuario.

2.1.1 Microsoft Visual Studio 6.0

Visual Studio es un completo entorno integrado de desarrollo para la construcción de aplicaciones de escritorio. Soporta lenguajes de programación C/C++, Visual Basic, entre otros; permite el desarrollo de proyectos con mucha facilidad que simplifican considerablemente el desarrollo de aplicaciones. Aunque el visual estudio 6.0 es una herramienta de Microsoft sólo debemos cambiar de compilador y añadirles las bibliotecas gráficas que deseemos utilizar y conseguiremos el mismo resultado.

Nos apoyamos en el lenguaje de programación C++ para explotar al máximo las capacidades que nos brinda la programación orientada a objetos, además es imprescindible

para lograr una alta compatibilidad con el resto de las aplicaciones desarrolladas, así mismo manteniendo altos niveles de portabilidad con un diseño modular.

2.1.2 Librería OpenGL (API)

Es un estándar creado por Silicon Graphics en el año 1992 para el diseño de una biblioteca 2D/3D. Actualmente es una de las tecnologías más empleadas en el diseño de aplicaciones 3D, y la última versión es la 3.7 que salió en el (2008). Esta diseñada para que pueda ser codificada en el lenguaje de programación C.

Se divide en tres partes funcionales:

- La biblioteca OpenGl, que proporciona todo lo necesario para acceder a las funciones de dibujado.
- La librería GLU (OpenGl Utility Library), es una biblioteca de utilidades que proporciona accesos rápidos a algunas de las funciones más comunes de OpenGl, a través de la ejecución de comandos de más bajo nivel, pertenecientes a la biblioteca OpenGl propiamente dicha.
- GLX (OpenGl Extension to the X Window System) proporciona un acceso a OpenGl para poder interactuar con un sistema de ventanas X Window, y está incluido en la propia implementación de OpenGl (su equivalente en Windows es la librería WGL, externa a la implementación de OpenGl).

Además de estas tres bibliotecas, la biblioteca GLUT (OpenGl Utility Toolkit) proporciona una interfaz independientemente de plataformas para crear aplicaciones de ventanas totalmente independientes.

- Aparté de las bibliotecas antes mencionadas también se consideró el uso de la biblioteca MUI (Micro Users Interfaces). Esta es una biblioteca codificada por Tom Davis, un trabajador de Silicón Grafic. Esta biblioteca de aplicaciones gráficas ha quedado hoy en día como un estándar dentro de OpenGl. Gracias a esta librería podemos generar barras de desplazamientos y botones en múltiples ventanas MUI.

2.2 Enlace Software Hardware del Sistema

A continuación se describe la arquitectura del software elaborada y su enlace con el hardware. Incluye los drivers diseñados para interactuar con los sensores IMU y GPS, se detalla el programa de adquisición y almacenamiento de datos.

El propósito del programa de adquisición es obtener datos de los sensores y almacenarlos en una estructura, que servirá de fuente para el sistema de control a bordo del vehículo. Así mismo transmitirá esa información a la estación de tierra donde un software similar tomará la información colocándola en una estructura similar para que sea leída por el programa de visualización en la estación de tierra, estos datos procesados pueden ser usados también para la verificación y optimización del modelo matemático del avión.

Se presentará además, una descripción general de las funciones y flujos de datos del sistema para comunicar a el visualizador con la base de datos en tierra, y la forma de actualizar esta con la información proveniente del vehículo.

Los sensores abordo estarán conectados a través de varias interfaces entre las que se incluyen protocolos como RS-232. La estructura general del sistema es la que se muestra en la figura 2.1.

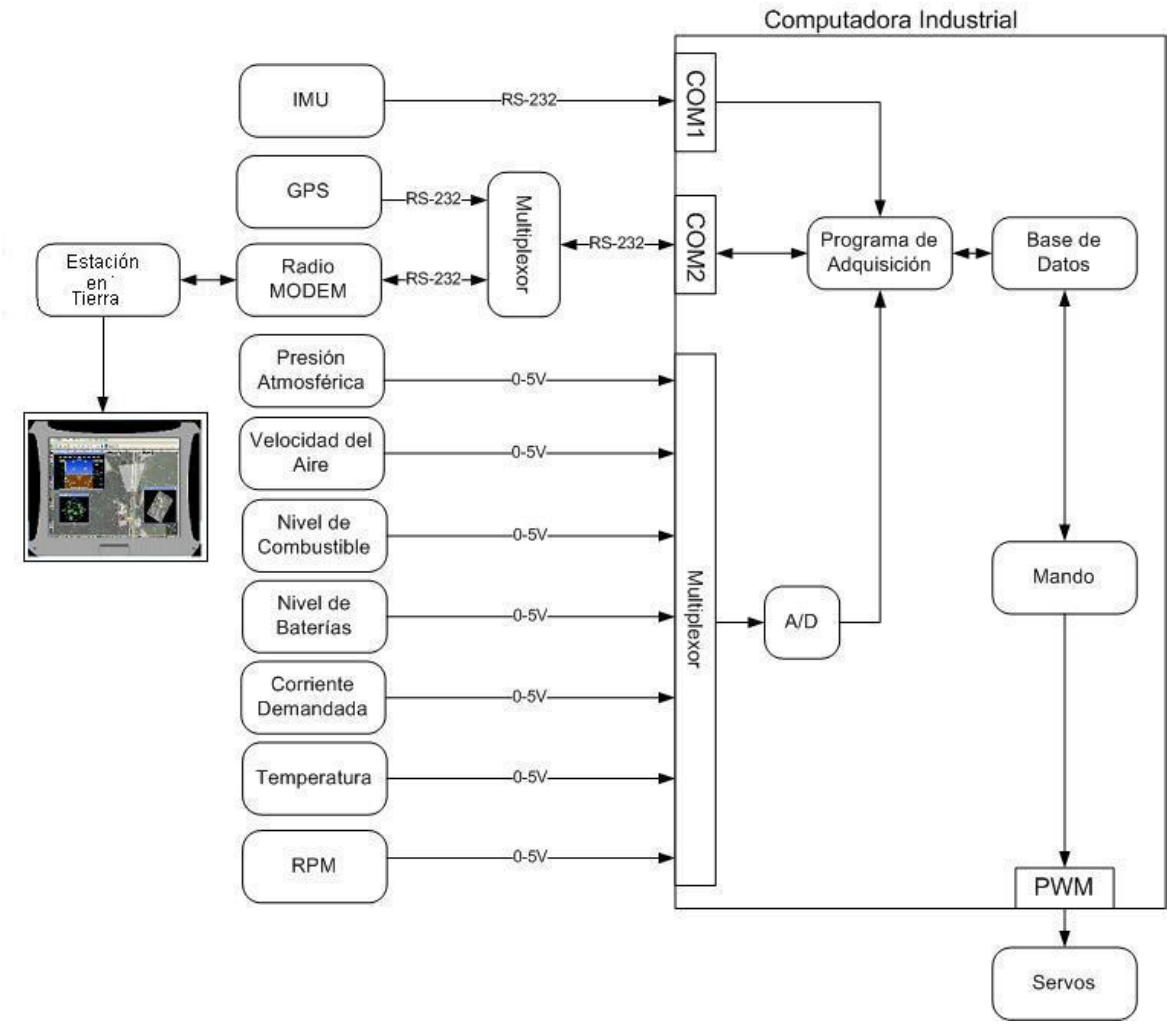


Figura 2.1 Enlace Software Hardware

A continuación explicaremos como fluyen los datos desde los sensores hasta la estación en tierra.

Los datos obtenidos por la (IMU) están conformados por un preámbulo que constituye el inicio de cadena, un contenido del mensaje, y un fin de la misma; estos valores están representados en notación hexadecimal. Los datos de la (IMU) son enviados a un sistema empotrado mediante un canal RS232. En el sistema empotrado existirá un programa de adquisición para trabajar con los datos muestreados y enviara los mismos a una base de datos que trabajara con el control y será retransmitida a la estación en tierra; el programa interactuará de forma bidireccional con la base de datos. Los datos obtenidos por el GPS se enviarán de forma similar a la base de datos lo que en este caso están multiplexados con el

radio MODEM que envía y recibe los datos que hacen interactuar directamente el sistema empotrado con la estación en tierra.

Los valores analógicos son multiplexados y enviados a la unidad de cómputo a través de un canal A/D. El mando actúa sobre los servos y toma los valores de la base de datos localizada en el sistema de cómputo abordo.

2.3 Diagrama en bloque de las subrutinas para el uso del autopiloto.

En este diagrama se representan una serie de pasos que describen el proceso de puesta a punto del autopiloto para una misión en tiempo real sobre una trayectoria predefinida. La participación humana se vuelve indispensable en los primeros pasos hasta quedar en una función básica de supervisión asistida por la computadora.

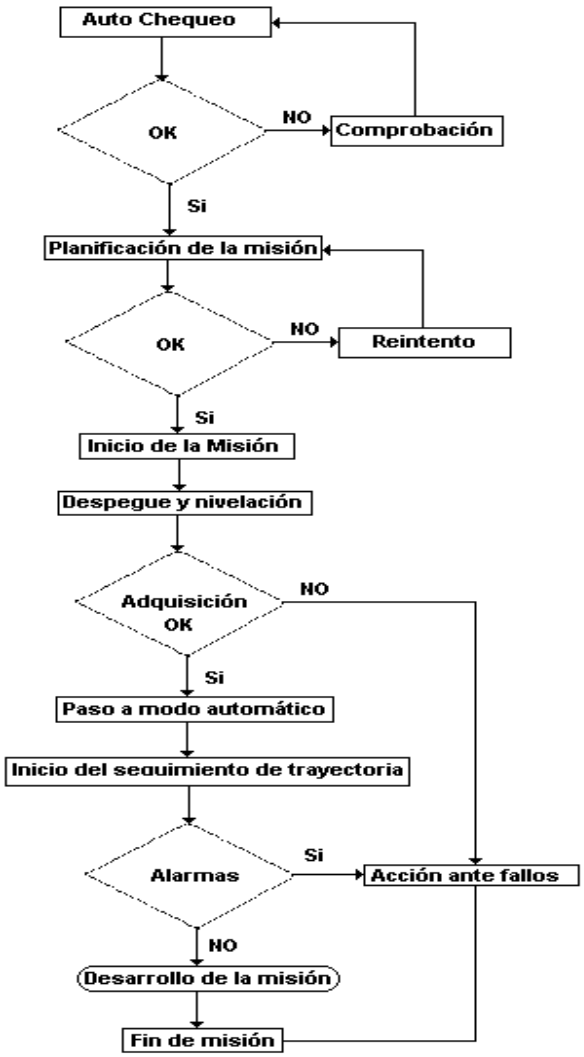


Figura 2.2 Diagrama de Flujo de Datos

En las subrutinas que a continuación describiremos interviene el supervisor.

2.3.1 Planificación de la misión

Este estado es desencadenado por el operador al mover un interruptor en el supervisor que pone al autopiloto en modo de configuración listo para recibir (vía MODEM) la trayectoria que deberá seguir el mismo durante la misión. Dicha trayectoria está constituida por un grupo de puntos en coordenadas GPS que serán almacenados a bordo del autopiloto, una vez terminado el almacenado el autopiloto devolverá una señal de listo al supervisor.

2.3.2 Adquisición OK

Si la adquisición se produjo correctamente la aplicación ejecutará los siguientes pasos mostrados en este diagrama en bloques de la figura.

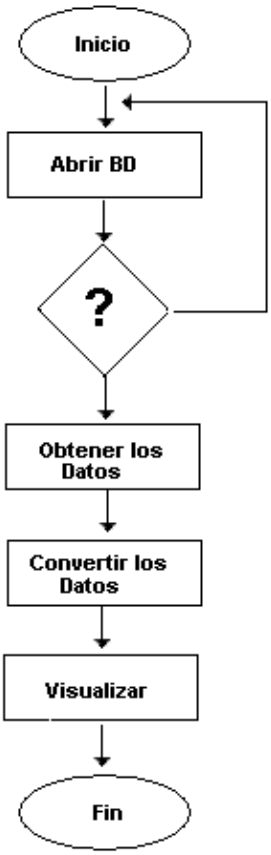


Figura 2.3 Diagrama de Adquisición

La tarea de adquisición se verificará ejecutando constantemente una aplicación que estará preguntando a la base de datos si se puede leer de ella los valores que estarán dispuestos en forma de arreglos numéricos donde en la primera fila se encontraran los datos actuales, en la segunda y tercera fila los datos posteriores; de este modo siempre se encontrará leyendo de la primera fila y sólo en caso de errores, es decir que exista un cambio considerable en los datos obtenidos se leerán las filas posteriores. Consideraremos como un error una variación del diez porciento en variables como el combustible y nivel de baterías, debido a la lenta dinámica de las mismas. Si la base de datos se abre correctamente se pasa a la siguiente sentencia en caso contrario el sistema se mantiene esperando por un tiempo determinado posterior al cual dará un error de Time Out.

La siguiente sentencia ejecuta una serie de subrutinas encargadas de obtener los datos de un arreglo y los irá almacenando según la variable a la que corresponda en forma de cadena de caracteres y se ejecutara la próxima sentencia (convertir datos), que estará relacionada como bien habíamos dicho con la conversión de los datos que estaban anteriormente en forma de cadenas a números de doble precisión para que puedan ser entendidos por el operario que esta observando el proceso.

Ya convertidos los datos solo nos quedara asignar cada valor de la variable a su objeto correspondiente en el programa y posteriormente visualizarlos hasta que se desee salir del mismo.

2.3.3 Paso a modo automático

Tras garantizar el vuelo estable del avión se comprueba en el supervisor la correcta a operación del proceso de adquisición de información por parte del autopiloto, y si todo esta correcto se le transfiere el mando al autopiloto con la con una señal.

2.3.4 Inicio del seguimiento de trayectoria

El autopiloto en esta fase comienza a seguir la trayectoria prefijada enviando a tierra todos los datos necesarios para mostrar su ubicación y estado. En esta fase se debe producir el encendido del sistema de transmisión de video a través de señal de control en el supervisor u monitor de televisión, en dependencia del caso.

2.3.5 Acción ante fallos

Si el sistema detecta algún error abordo será mostrado en el supervisor esperándose una acción de control por parte del operador (típicamente se regresara a la base o se continuara la misión) si dicha acción no es atendida en el plazo de 3 minutos el avión iniciara una secuencia de emergencia tratando de regresar al punto de partida. Esta acción puede estar representada por la aparición de una alarma en la pantalla, mensaje de error etc.

2.4 Funcionalidad y Casos de uso

2.4.1 Funcionalidad

El objetivo de la aplicación es que un usuario pueda visualizar toda la información relacionada con el vuelo del UAV.

Para ello la primera función a la que se recurre es a la de cargar en la ventana de supervisión los indicadores que se corresponden con los valores medidos por los sensores a bordo del vehículo y que son transmitidos vía inalámbrica a tierra.

Se deberá poder seleccionar lo que se desea visualizar, dar la posibilidad de maximizar o minimizar los sensores para personalizar el mímico, la configuración tiene que ser salvable ya sea el mímico cargado en la ventana o el mapa donde se encuentra el avión en el momento que se desee dadas ciertas coordenadas de latitud - longitud.

2.4.2 Casos de uso

El diagrama de casos de uso representa la forma en cómo un Usuario (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en cómo los elementos interactúan (operaciones o casos de uso).

Para empezar se debe identificar los actores que interactúan con el sistema. Al ser un visor de información, por ahora no habrá más que un operario que busque el fin anteriormente mencionado.

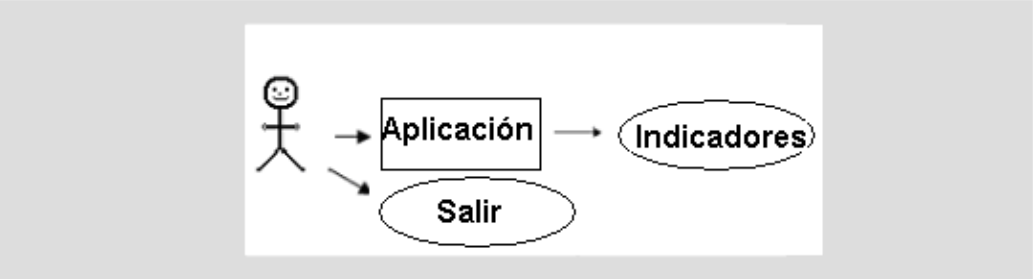


Figura 2.4 Caso de Uso 1

Ahora se mostrara lo qué puede hacer este actor con la aplicación.

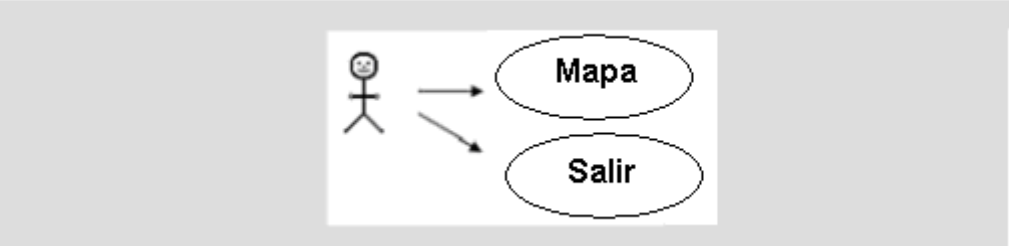


Figura 2.5 Caso de Uso 2

Una vez cargado el mapa aparece un nuevo caso de uso.

Caso de uso 1(Aplicación)

Actores: Gestor (iniciador)

Propósito (visualizar la información de los indicadores)

Resumen: En este caso el principal objetivo es visualizar la información ofrecida por los diferentes sensores cuyos valores se encuentran en una base de datos en diferentes posiciones de un arreglo numérico.

Tipo: Primario Esencial

Curso Típico de Acciones

Tabla 2.1 Caso de Uso 1

Acciones del Actor	Respuesta del sistema
1. Un usuario visualiza la ventana principal.	2. El sistema debe ir a la base de datos y buscar la información referente a cada uno de los indicadores mostrados en pantalla.
3 El usuario da un clic derecho en cualquier región dentro de la ventana principal.	4. El sistema debe mostrar un menú con las opciones de ver el mapa o salir de la aplicación.

Actores: Gestor (iniciador)

Propósito (visualizar la información de la posición del vehículo en un mapa)

Resumen: En este caso el objetivo es visualizar la información ofrecida por el mapa referente a la posición del avión dadas las coordenadas del GPS en latitud longitud también colocadas en posiciones del arreglo numérico.

Tipo: Primario Esencial

Curso Típico de Acciones

Caso de uso 2 y3 (Mapa y Seguimiento)

Tabla 2.2 Caso de Uso 2 y 3

Acciones del Actor	Respuesta del sistema
1. El usuario desea visualizar la información que brinda la situación del vehículo en coordenadas de latitud longitud.	2. El sistema debe mostrar la información refrescando constantemente la situación del vehículo en el mapa.
3.El usuario desea volver a la ventana principal para ver el comportamiento de los indicadores de vuelo	4.El sistema debe tener la capacidad de cargar nuevamente la ventana principal para mostrar los datos de indicadores

2.5 Aspecto Visual de la aplicación

Es importante pensar en cómo se distribuirán las herramientas de navegación y visualización sobre la pantalla. En este epígrafe describiremos como quedará planteado el aspecto visual en ventana principal.

Como toda aplicación de visualización se deberá disponer de un amplio panel en el que se represente el mapa que estará situado en una ventana secundaria.

En el centro de la ventana principal se colocará la imagen del horizonte que representa la inclinación con respecto a tierra del avión.

Debajo y a ambos lados del horizonte se mostraran los diferentes indicadores que representan los valores relacionados con la altura, nivel de combustible, nivel de baterías, revoluciones por minuto del motor, velocidad con respecto a tierra y orientación con respecto al norte magnético.

Como una opción el aspecto visual de la aplicación pudiera quedar distribuido de la siguiente manera:

En la esquina superior izquierda deberá aparecer el título de la ventana que se este mostrando; en este caso el título de la ventana es:

Estación Terrestre

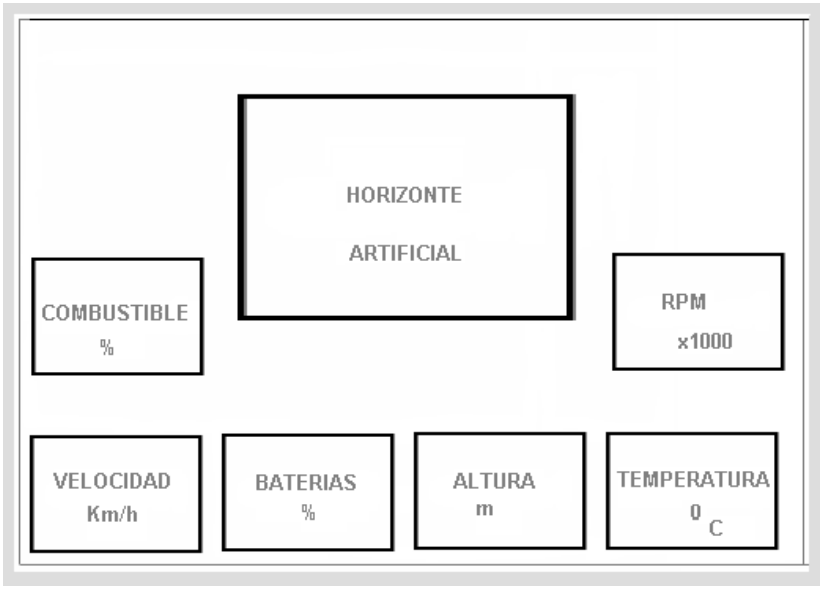


Figura 2.7 Estación Terrestre

Este es el aspecto visual que debe tener el mapa de nuestra aplicación.

El título que tendrá la ventana será Mapa.

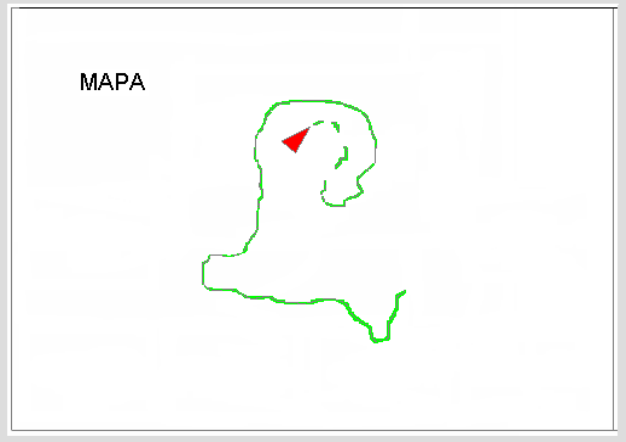


Figura 2.8 Mapa

2.6 Relación entre instrumentos e indicadores de visualización

2.6.1 Instrumentos de medición

Para obtener una representación correcta de la medición de los parámetros de vuelo del avión es necesario representar los distintos indicadores en nuestra ventana. En este apartado se describen cada uno de los valores que deben tomar los indicadores y el porque de su representación gráfica, además de su relación con los instrumentos de medición. Se debe tratar de que los indicadores se asemejen a los utilizados en simuladores.

La figura que mostraremos a continuación representa la relación entre los sensores digitales del sistema, la instrumentación auxiliar y la estación en tierra donde se encuentran los indicadores correspondientes a los sensores.

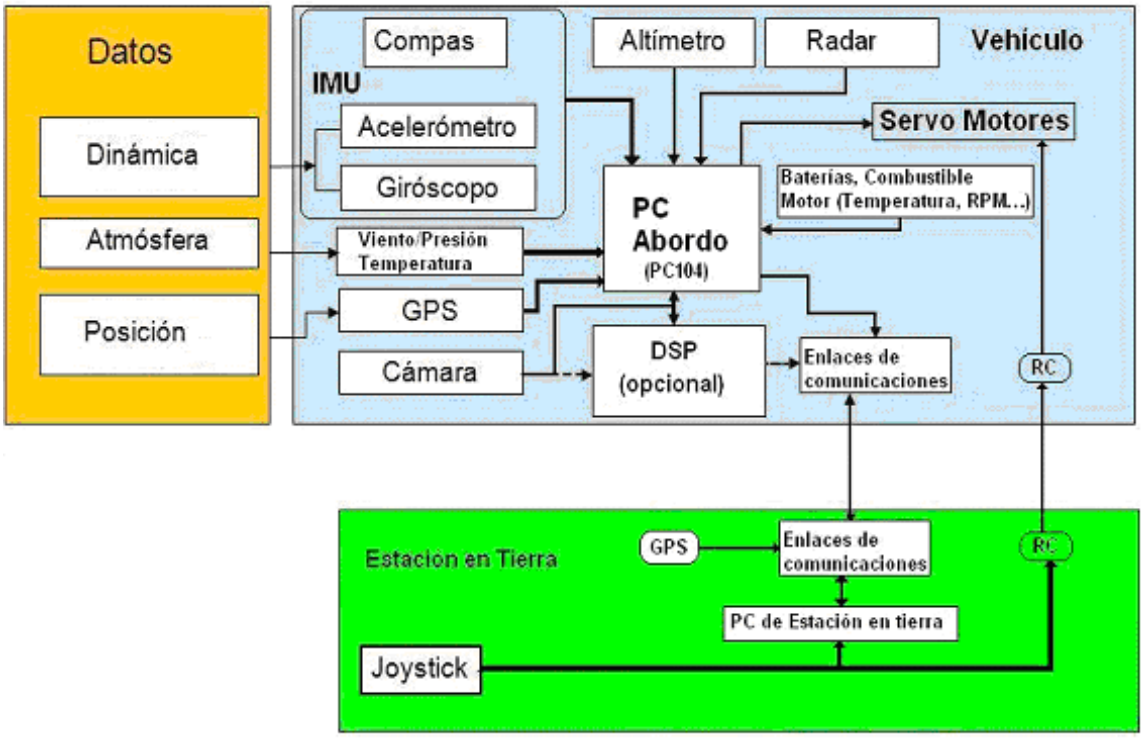


Figura 2.9 Esquema de la Arquitectura General.

- **MODEM digital**

Es el encargado de enviar los datos a tierra para que sean almacenados en una base de datos, opera en 900 MHz con una tasa de transferencia de 19200 bps empleando técnicas de espectro extendido.

- **Joystick**

Habitualmente el control manual sigue estando auxiliado por el sistema de control, de modo que se facilita el pilotaje del avión mediante un mando de tipo joystick.

- **Instrumentación Auxiliar (Variables atmosféricas, temperatura, velocidad, altura, rpm del motor, nivel batería, combustible)**

Estos sensores son los encargados de realizar determinadas funciones críticas para garantizar el vuelo del UAV. Entre éstas, la disponibilidad de la alimentación eléctrica y del combustible, por lo que se suele incluir elementos que proporcionen alarmas cuando se alcancen valores por debajo de umbrales predefinidos. Estos avisos pueden ser comunicados a la estación de tierra, desde donde se deben tomar las acciones oportunas, o

incluso desencadenar procedimientos de seguridad en el propio sistema embarcado, como el aterrizaje. La temperatura en el interior del alojamiento del sistema de control puede estar sometida a elevadas temperaturas motivadas por el propio funcionamiento de sus sistemas electrónicos y especialmente por el flujo de calor procedente del motor. (Barrientos 2007)

2.6.2 Indicadores de visualización

El refrescamiento del valor de los indicadores se realizará una vez por segundo

- Combustible. Este es un indicador que representa los niveles de combustible del avión y es indispensable su uso, los valores de combustible se encuentran en % y van desde un rango de 100% a 0 en orden descendente. Optaremos además por representar los indicadores por colores; para este caso el color verde toma valores entre (100% y 80%) representa máximo nivel de combustible; el valor entre (80 y 40%) representado por el color amarillo indicara una zona de trabajo operable y por último el color rojo (40 y 0%) poco combustible.

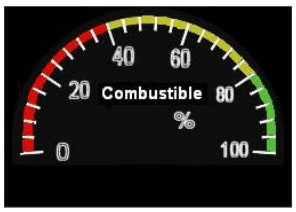


Figura 2.10 Indicador de Combustible

- Velocidad. Indicador que representa la velocidad contra el viento del vehículo, tomando en cuenta de que el avión tiene una velocidad crucero de aproximadamente 120 km/h y máxima de 170 km/h se diseñara este indicador con un rango de (0 a 180 km/h). Para este caso el color verde representa un rango de operación entre (45 y 120 km/h) que es la velocidad promedio a la que se mantendrá el vehículo; el color amarillo ya representa una zona de mayor peligro en la escala que va desde (120 a 160 km/h) y se considerara como un valor máximo en la escala 180 km/h.

Figura 2.11 Indicador de Velocidad

- Baterías. Este indicador representara el nivel de baterías del vehículo puesto que existen diferentes juegos de estas, uno representa la aviónica y el otro es el del sistema de mando manual y los actuadores. También los niveles de baterías se representarán en el rango de (100 a 0) % en orden descendente tomando el color verde como máximo nivel de baterías que estará entre (100 y 80) %, el valor entre (80 y 40) % representado por el color amarillo indicara una zona de trabajo operable y por último el color rojo (40 y 0) % baja carga eléctrica. El operador debe seleccionar los valores de entrada para su representación en %.

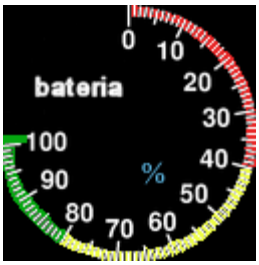


Figura 2.12 Indicador de Nivel de Baterías

- Altura. Indicador que representará el altímetro del avión que poseerá una altura de 400 m como valor limite, es por ello que para diseñar el indicador se le asignará una altura máxima de 480 m. El rango de valores que se mostraran en el indicador de será de (0 a 480) m y en orden ascendente y el color carmelita representará los valores máximos entre (440 y 480) m.

Figura 2.13 Indicador de Altura

- Brújula Este indicador representara la posición del avión con respecto al norte magnético. La brújula como posee características diferentes debe tener números y letras que representen el norte el sur u otro punto cardinal. Es de vital importancia la representación grafica de este indicador para obtener la orientación de hacia donde nos dirigimos El color de la aguja deben ser diferente para mejor visualización para el operario, la aguja tendrá ángulos de giros de (0 a 360 grados).

Figura 2.14 Indicador de Referencia Magnética

- Temperatura .Es el encargado de mostrar el valor de la temperatura del motor que tendrá como valor máximo en la escala 100 ^{o}C con rango de 0 a 100 ^{o}C; aunque la temperatura generalmente no debe sobrepasar los 70 ^{o}C; y como máximo dentro del indicador el valor critico de temperatura serán los 90 ^{o}C que se representará con color rojo.

Figura 2.15 Indicador de Temperatura

- RPM. Este indicador es el encargado de medir las revoluciones por minuto del motor, para una mejor representación en la escala tendrá que tener algún tipo de multiplicador, los multiplicadores serán mayormente múltiplos de 10 para que el

usuario no tenga que estar perdiendo tiempo en cuentas cuando la aguja marque un valor; tendrá una escala de (0 a 10) y para su representación necesitamos un multiplicador de 1000. El color entre 0 y 8000 será el verde y como color critico el rojo que indicara la zona de 10000 rpm, el color blanco representara el máximo número de revoluciones por minuto del motor que es de 9000.

Figura 2.16 Indicador de rpm del Motor

- Horizonte Artificial. Es junto a la brújula uno de los indicadores mas importante dentro de la ventana de supervisión debido a que nos brinda la información del nivel de inclinación del avión con respecto a tierra, este indicador debe tener la capacidad de estar rotando y desplazándose constantemente en dependencia de los grados del movimiento de alabeo que no es mas que (el ángulo Φ creado por el movimiento del avión cuando éste gira en torno al eje horizontal (eje X)) y Cabeceo **(**es el ángulo creado por el movimiento del avión cuando éste gira en torno al eje latitudinal (eje Y)).

Debe estar dividido a la mitad por dos colores, el color carmelita que representa la tierra y el azul que representa el cielo. Otra característica no menos importante que deberá presentar el horizonte artificial es su graduación; es decir tiene que representarse cuantos grados de inclinación se esta con respecto a cielo o a tierra respectivamente.

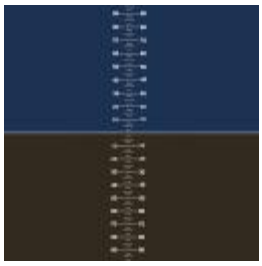


Figura 2.17 Indicador Horizonte Artificial

El sensor que nos da la información para generar el horizonte virtual y actualizar la brújula es uno solo la IMU, este sensor nos brinda posición angular, aceleración, razón de giro, orientación magnética, temperatura como habíamos señalado en epígrafes anteriores, variables que son enviadas a la unidad de cómputo mediante protocolo RS232 como ya se menciono antes.

Los sensores como el GPS y la IMU tienen capacidad de procesamiento interno que les permite procesar la información que adquieren y exportarla en diferentes formatos de acuerdo a las necesidades del control o supervisión.

CAPÍTULO 3. IMPLEMENTACIÓN Y RESULTADOS

La idea general de este capitulo es crear una guía de usuario que servirá para la continuación de este proyecto. Se representará la manera en que se organizó y se estructuró el trabajo con las clases y diferentes objetos en el programa. Y mostraremos posteriormente el resultado de la aplicación

3.1 Implementación

Para la realización de la aplicación fue necesario implementar algunas clases que dibujan y rigen el comportamiento de los diferentes objetos que se mostraran en la ventana de visualización, para un mejor entendimiento se mostrara un diagrama de clases con cada uno de los métodos que se implementan en ellas y sus atributos pertenecientes.

3.1.1 Clases CTexture e Image

Estas clases presentan una relación de asociación como se mostrará posteriormente en la figura 3.1 por el símbolo de asociación que lo describe, debido a que un objeto CTexture es creado a partir de un objeto imagen. El método de la clase CTexture getTextureHandle no es más que el encargado de obtener el identificador de textura que hace falta para la asignación de una imagen cualquiera a un polígono que creemos con OpenGl, por su parte el método ApplyTexture es el encargado de habilitar el trabajo con texturas. El método image es el encargado de obtener la imagen de un fichero en formato mapa de bit. El método invertImage invierte los colores de las imágenes; addAlphaChannel agrega los canales de la imagen cargada pertenecientes a colores de rojo, verde y azul; en resumen esta clase Image esta diseñada para el trabajo con imágenes

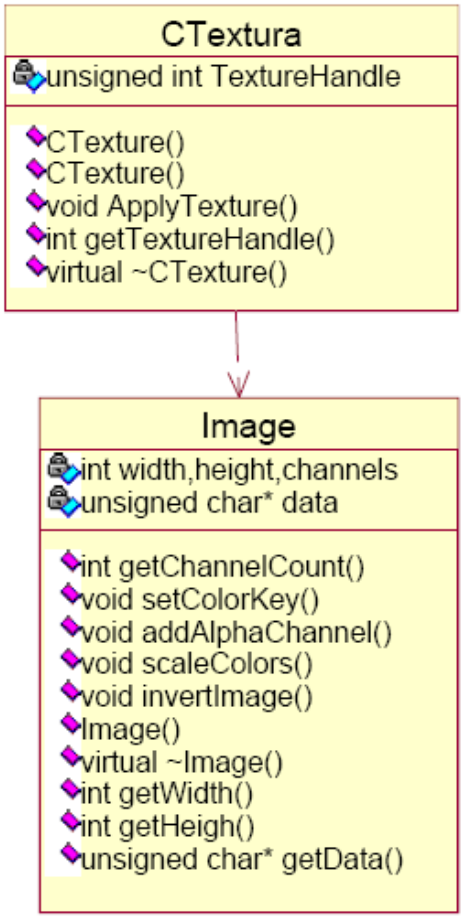


Figura 3.1 Diagrama de las clases CTexture e Image

3.1.2 Las clases Reloj

Estas clases presentan una relación de herencia, tanto las clases Reloj Aguja como Reloj Horizonte heredan de la clase Reloj como se podrá observar en la figura 3.2 por el símbolo de asociación que lo describe. La clase reloj tiene cuatro métodos y es utilizada para crear el objeto mapa. El método PositionR es el encargado de asignar las posiciones que tendrá el polígono que representara al objeto en la ventana de visualización, el método TranslateR es el que traslada el polígono al centro de la ventana donde se comenzará a dibujar, RotationR se encargará de la rotación del polígono que describe al mapa con respecto a los tres ejes coordenados (x,y,z); initDrawR método que dibujara la imagen que representa al indicador en la ventana de visualización después de haber asignado una textura a un polígono determinado. En la siguiente figura se muestran las relaciones entre las clases Reloj, Reloj Aguja y Reloj Horizonte.

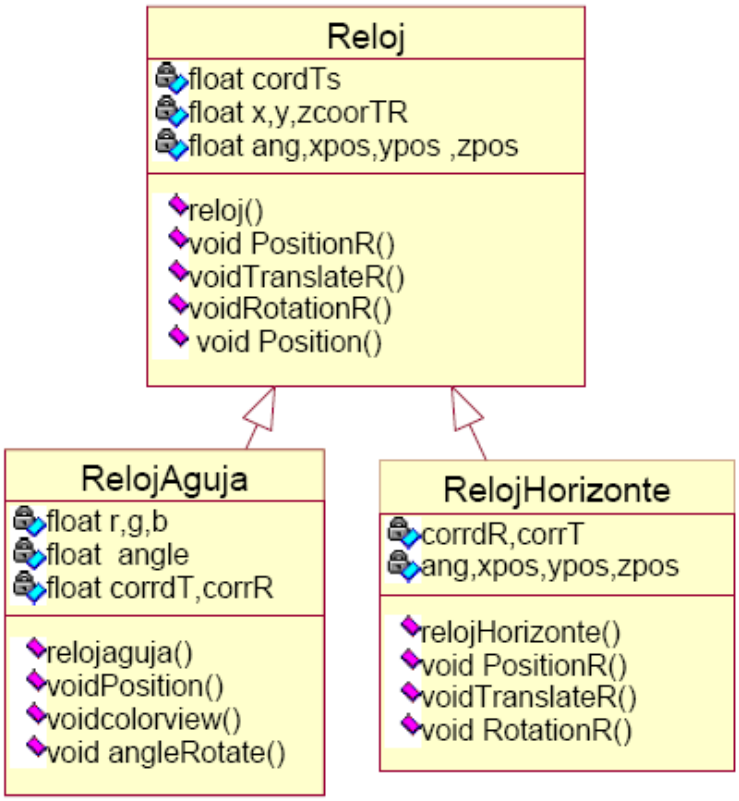


Figura 3.2 Diagrama de las clases Reloj, Reloj Aguja y Reloj Horizonte

La clase Reloj Aguja tiene como principal particularidad que al crearse una instancia perteneciente a ella dibujara un polígono en forma de aguja que será asignado a una imagen de un indicador. No será creado ningún objeto de esa clase aguja hasta que no se le pase como parámetro al constructor de esta coordenadas en el espacio (x,y,z) que representaran a un polígono en forma de aguja, cumpliéndose con esto un concepto fundamental de la programación orientada a objeto como lo es la encapsulación de los datos, el método Position traslada el polígono diferentes coordenadas en pantalla. Esta clase además de heredar de la clase Reloj posee sus propios métodos que aunque no se encuentran en el diagrama también forman parte de el, initDrawNeedle nos asigna al polígono la imagen del indicador ejemplo (indicador de altura), colorview indica el color que tomara la aguja cuando dibujemos el polígono, angleRotate método que describe el ángulo que rotara el polígono que describe la aguja del indicador. La clase Reloj Aguja posee atributos que son comunes para todos los objetos creados de este tipo como son el ángulo de giro, las coordenadas de dibujo, color de la aguja, coordenadas de traslación en pantalla.

La clase Reloj Horizonte es la encargada de crear como lo describe prácticamente su nombre el Horizonte Artificial y posee como datos miembros los métodos PositionR que es el encargado de crear el polígono que representa al horizonte, RotationR método el cual se encarga de la rotación del polígono que contiene a la imagen de textura en cualquiera de los tres ejes tres ejes coordenados, TranslateR se encarga de trasladar el polígono a las diferentes zonas de la pantalla, initDrawR nos asigna al polígono la imagen del indicador y para este caso es la imagen del horizonte artificial.

3.1.3 Clase Mask

Esta es la clase correspondiente a la mascara que hace que el Horizonte se mueva por detrás del un rectángulo centrado en la ventana de visualización, esta clase no tiene ningún tipo de relación con ninguna de las anteriormente mencionadas. Los métodos para esta clase son initDrawMask() que se encargara de dibujar el fondo de la ventana de visualización y initDrawLine() que dibujará las líneas divisorias entre cielo y tierra en el Horizonte artificial.

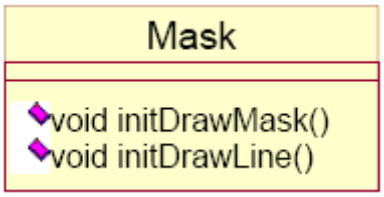


Figura 3.3 perteneciente a la clase Mask

3.2 Resultado del trabajo

Como resultado del trabajo mostraremos a continuación la interfaz de usuario que se obtuvo para la interacción con los sensores.

En la figura se puede observar la situación en coordenadas de pantalla de cada uno de los indicadores de nuestro sistema de supervisión.

Figura 3.4 Ventana Principal de la Aplicación

Esta segunda figura representa un punto rojo que se corresponderá con el avión en pleno vuelo, este punto debe estarse actualizando constantemente a medida que cambie su posición en coordenadas de latitud longitud, en el código del programa dichas coordenadas deben ser convertidas a coordenadas de pantalla de largo y ancho. La imagen fue tomada de Google Heart; cuando se desee hacer un vuelo real se deberá tomar una foto de la región y cargarla en el proyecto.

Figura 3.5 Seguimiento de la Trayectoria

3.3 Requerimientos técnicos para el uso de la aplicación

Proponemos según las pruebas realizadas con esta aplicación que el ordenador donde se ejecute deba cumplir con los siguientes requerimientos mínimos:

- Procesador Pentium III o equivalente de otro fabricante.
- 256 MB de memoria RAM.
- 512 MB de tarjeta de video.

Este último factor de trabajo es muy importante debido a que la aplicación esta realizada con la ayuda de OpenGl que es limitada por el hardware gráfico, es decir mientras mejor sea nuestra tarjeta aceleradora obtendremos mejores resultados y mayor calidad de visualización.

3.4 Análisis económico

Por lo general las interfaces de supervisión para vehículos aéreos autónomos son aplicaciones patentizadas; es decir para obtenerlas hay que pagar por ellas y generalmente el precio de estas aplicaciones suelen ser elevados; lo que implicaría un gasto económico apreciable para el proyecto general. En la actualidad no es muy usual que encontremos en el mercado software gratis de este tipo.

Por lo tanto es necesario lograr un software que cumpla con las características de diseño que se propusieron en este trabajo y que a continuación señalamos.

3.4.1 Open source

Open source (Código Abierto). Al ser un software de código abierto nos permite ciertas libertades que mostraremos a continuación.

- Se podrá estudiar como funciona el programa, y adaptarlo a sus necesidades. Significa que podemos estudiar su funcionamiento al tener acceso al código fuente, lo que nos va a permitir, entre otras cosas; averiguar como realiza determinada tarea, descubrir que otras posibilidades tiene, que es lo que le falta para hacer algo, etc. El personalizar el programa a determinadas necesidades implica que se le puede suprimir partes que no son de interés, agregarle partes que se consideran de mayor importancia, añadirle código creado por otro desarrollador , etc.
- Distribuir copias. Quiere decir que soy libre de distribuir el programa, de forma gratuita o con algún costo, ya sea por email, FTP o en CD, el receptor puede ser una persona o a varias, dentro del país o fuera de el.
- Mejorar el programa, y liberar las mejoras al público. Tengo la libertad de hacer mejor el programa, o sea que puedo hacer menores los requerimientos de hardware para funcionar, que tenga mayores prestaciones, que ocupe menos espacio, o que se corrijan errores conocidos o detectados por los usuarios y reportados.

3.4.2 Uso de bibliotecas libres

El uso de bibliotecas libres nos da la facilidad de que al cambiar el compilador o plataforma no cambiaremos el código sino que solamente es necesario encontrar la forma en que estas bibliotecas son añadidas a la plataforma de desarrollo que estemos utilizando.

Cumpliendo con estas características anteriormente señaladas el único gasto que tendríamos en el proyecto por la parte de supervisión sería el uso de una computadora para el desarrollo, la cual tendría un precio aproximado de 1000 CUC.

3.5 Otras plataformas para trabajos futuros

Al ser este un proyecto que se le dará continuidad en los próximos meses sugerimos otras plataformas que podrán ser útiles a la continuidad de este trabajo.

3.5.1 Qt Designer

QT Designer es una herramienta para el desarrollo de formularios y presentaciones gráficas para las aplicaciones. Permite acelerar el desarrollo de interfaces de alto rendimiento, a la vez que proporciona una forma fácil de diseñar interfaces gráficas de usuario avanzadas generando el código fuente para las mismas, lo que permite al desarrollador ajustarlo a sus necesidades.

Este generador de interfaces fue creado inicialmente por la empresa TROLLTECH para trabajar en varias distribuciones Linux. No obstante, actualmente puede instalarse en otras plataformas como Windows y Mac OsX.

La librería QT es una librería implementada en C++ y orientada a objetos. Actualmente hay enlaces para otros lenguajes como Python o Perl. En el caso de java, se ha desarrollado un API (Interfaz para Programación de Aplicaciones) para poder utilizar esta librería C++ en aplicaciones de este lenguaje.

3.5.2 NetBeans

El constructor de interfaces de usuario del IDE NetBeans conocido como el “Proyecto Matisse” es un módulo del Entorno de Desarrollo Integrado NetBeans.

Dicho sistema de software es un proyecto de código abierto que, antes de la versión 5.5 se distribuía bajo la Licencia Pública de Sun. Este proyecto está patrocinado por Sun Microsystems aunque se inició originalmente como un proyecto estudiantil. Tiene dos modos de vista: diseño y código fuente. En el modo de vista de diseño se pueden arrastrar y soltar los diversos componentes, mientras que en el modo de vista de código fuente se puede ver y editar el mismo.

3.6 Conclusiones del capitulo

El capítulo fue dedicado a la implementación del software de supervisión mediante la programación estructurada, de esta forma tenemos la posibilidad de lograr una mayor eficiencia cuando se ejecute el programa.

Como resultado obtuvimos una aplicación que se ejecutara desde la estación de tierra capaz de representar gráficamente los datos obtenidos por los diferentes sensores que componen el sistema.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1 Conclusiones Generales

Con el desarrollo de este trabajo se han podido llevar a la práctica conocimientos relacionados con la programación gráfica usando C++ y OpenGl que son de gran utilidad en nuestros días. Además, se ha tenido una valiosa experiencia en el entendimiento de la programación orientada a objetos, lo que repercutió en una correcta organización a la hora de realizar el código del programa.

Este informe de investigación constituye una fuente bibliográfica para próximas indagaciones sobre el tema de desarrollo de sistemas de supervisión para vehículos aéreos autónomos (UAV).

El principal logro de la presente Tesis, es su resultado global; acerca de la concepción general del sistema, y la creación de un ambiente de usuario apropiado para la monitorización de parámetros de un vehículo aéreo autónomo totalmente independiente a herramientas propietarias.

No obstante el trabajo no se ha limitado a una implementación y fusión de técnicas, sino que en algunos de estos casos se han realizado aportes originales como es el uso de clases con OpenGl lo que permite una mejora en la encapsulación de datos ya que OpenGL trabaja de forma procedural.

2 Revisión de objetivos

“Un objetivo es el conjunto de resultados cualitativos que se propone alcanzar a través de determinadas acciones”. Con esta definición, que se puede encontrar en algunos

diccionarios, entendemos que un objetivo significa marcarse una meta e intentar conseguirla. Puede que no la consigamos. Podemos aprovechar esta situación y analizar porqué no se han conseguido.

Los objetivos propuestos se han cumplido, pues se han logrado solucionar problemas que aparecían al comienzo de este trabajo, por lo que arribamos a las siguientes conclusiones:

- Se planteó una arquitectura de software de alto nivel que fuera de código abierto, diseñada en C/C++ y sin ninguna dependencia de controles ActiveX.
- Se logró diseñar una interfaz de usuario capaz de interactuar con los módulos correspondientes a los principales sensores (IMU y GPS).

3 Conclusiones personales

Además de haber hecho un trabajo final de carrera, he aprovechado el tiempo empleado para colaborar en un proyecto de cierta envergadura, formarme como ingeniero, y aprender de lecciones que no se dan en las clases. El trabajo de fin de carrera es la última tarea académica que desarrolla el ingeniero. Es la conexión entre la carrera y el mundo laboral, y también se puede aprender cosas con ello; como por ejemplo estructurar y planificar un proyecto de mediana duración, plantearse objetivos y buscar la manera de cumplirlos.

Recomendaciones

El trabajo ha cumplido con los objetivos iníciales, pero ello no imposibilita que se realicen las siguientes recomendaciones.

1 Para lograr un mejor desempeño cuando se ejecute el programa se deberá independizar el sistema de ventanas de OpenGL brindado por la biblioteca glut.h. Es decir se debe recurrir solo a OpenGl cuando se vallan a cargar las imágenes que necesitemos para la animación y a cuando se usen primitivas. Esto reduce considerablemente el gasto computacional debido a que la ventana de la aplicación no se dibujará constantemente.

2 Como principal recomendación para futuros trabajos relacionados con el tema se encuentra la asignación dinámica de imágenes para lograr una mayor flexibilidad cuando se desee representar el punto que mostrara al avión en nuestra pantalla y se

describa la trayectoria, deberá existir una subrutina que sea capas de conectar la aplicación con algún sistema que permita la actualización cada cierto tiempo de la posición del vehículo en el mapa debido a que en nuestra aplicación se realiza de forma un poco rígida y esto no es deseable.

REFERENCIAS BIBLIOGRÁFICAS

Agudelo, A. (2007). "Diseño e Implementación del Sistema de Comunicaciones basado en can para la aviónica en un vehículo aéreo

Autónomo no tripulado".

Barrientos, A. (2007). "Vehículos aéreos no tripulados para uso civil. Tecnología y aplicaciones."

Fernández, R. (2005). "Aplicación de los sistemas de control en la industria aeronáutica: Vehículos aéreos no tripulados."

Fernando, B. (2006). Autonomous Fixed Wing Unmanned Aerial Vehicle. Monash, Universidad de Monash.

Giner, J. D. C. (2007). "Arquitectura Abierta para el Control Autónomo y Tele operado de un Mini-Helicóptero."

Hernández, L. (2006). "Desarrollo de mini helicóptero autónomo de bajo costo para la toma de imágenes con aplicaciones en la agricultura."

Ippolito, C. (2005). "An Autonomous Autopilot Control System Design for Small-Scale UAVs."

Laguardia, A. S. M. (2008). "Arquitectura de hardware para un vehículo autónomo aéreo."

Mata, J. L. (2006). "UAV: Una alternativa para la inspección medioambiental y de infraestructuras."

Ollero, A. (2004). "Sistema Basado en el Empleo Vehículos Aéreos no Tripulados para la Lucha Contra Incendios Forestales."

Rosas, J. M. L. (2007). Estación de seguimiento SKY-EYE para UAVs:
Integración visual de componentes de seguimiento y georeferenciación
sobre GIS, Universidad Politécnica de Cataluña.

Valdivia, B. G. (2006). "Estudio de viabilidad de aplicaciones de observación aérea con UAVs y
elaboración de un plan de empresa."

Vélez, C. M. (2000). "A Rapid Prototyping Methodology for the Control of Unmanned Aerial Vehicles."

ANEXOS

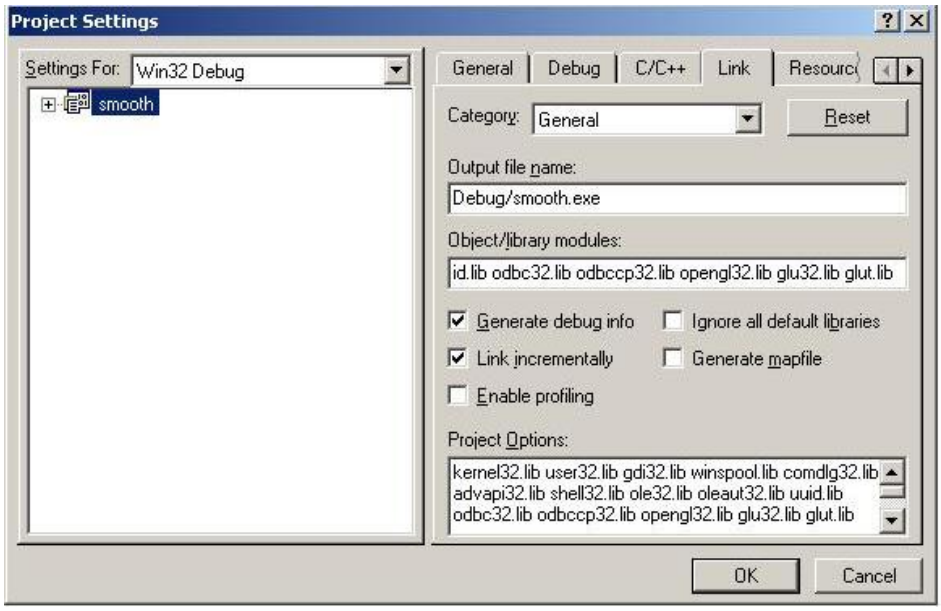
Anexo I Como evitar errores en linking

Cuando se instalen las bibliotecas de OpenGl con visual C++ se deben desarrollar los pasos siguientes que mostraremos a continuación:

Archivos	Directorios
*.H	C:\Archivos de programa\Microsoft VisualStudio\VC98 \Include\GL
*.LIB	C:\Archivos de programa\Microsoft VisualStudio\VC98\Lib
*.DLL	C:\Windows\SYSTEM o en SYSTEM32

A continuación mostramos detalles para impedir esos molestos errores en construcción.

En Project > Setting > Link > Object/Library modules (ver figura) incorpore al final las siguientes bibliotecas: opengl32.Lib. glu32.Lib. glut.Lib



Anexo II Código comentado y funciones importantes

Este anexo tiene como objetivo comentar algunas funciones y bibliotecas usadas en el código del programa principal para un mejor entendimiento del mismo.

```
#include <GL/gl.h>
#include <GL/glu.h>
#include<Math.h>
#include <math.h>
#include <stdlib.h>
#include <iostream.h>
#include <stdio.h>
#include "Texture.h"
#include "Image.h"
#include "Mask.h"
#include<Gl/glut.h>
#include "Reloj.h"

#include"RelojAguja.h"

#include"RelojComplejo.h"

#include <mui/mui.h>
```

Constantes

```
#define ANCHO  800
#define ALTO   500
#define ORIGENX 100
#define ORIGENY 100
#define pi      3.141592654
#define FILA  50
```

Creación de Objetos

Estos punteros que apuntan a la clase CTextura deben tener el mismo nombre del fichero BMP creado en el proyecto.

```
 CTexture * horizonte;
CTexture * velocidad
```

Ejemplo de cómo se crea y se inicializa un objeto; todos son creados de igual forma por eso con un ejemplo vasta para obtener la idea esencial.

relojaguja velocidadr(coordenadas del dibujo de la aguja);
velocidadr.PositionR(tamaño del objeto);
velocidadr.TranslateR(posición en la ventana);
velocidadr.RotationR(ángulo de rotación);
velocidadr.initDrawR(objeto CTextura);

Se crea la mascara del horizonte

mask horizont;
horizont.initDrawMask();
Se crea la linea divisoria entre cielo y tierra del horizonte artificial

mask line;
line.initDrawLine();

Funciones

void inicio (). Esta función se encarga de inicializar el escenario Opengl y dentro del cuerpo de esta se encuentra el apuntador img.; se inicializa además la matriz de proyección a través de una matriz identidad, se carga una proyección ortogonal. Se carga también la matriz de modelado del sistema a través de la matriz identidad que nos traslada hacia el centro de la ventana y con el color del buffer gris representado por 0.5.

Image* img. Apuntador a la clase image que obtiene la información referente al bmp que se le pasa al constructor de la clase Image y luego es liberado su espacio en memoria con un delete

void nuevaventana(). Función esta que se encarga de crear una nueva ventana cuando se pulse con el clic derecho en el interior de una ventana, en su cuerpo se un apuntador image y se inicializan las matrices de modelado y proyección respectivamente mencionadas con anterioridad.

void dibujar2(). Esta función dibuja el mapa, y el punto que se moverá en el interior será el referente al vehículo

void idle2() En su interior posee otra función **glutPostRedisplay()** que le dice a OpenGl que se mantenga dibujando.

void dibujar() En el cuerpo de esta función se encuentra la función **leerDatos()** que actualiza los datos obtenidos del arreglo numérico, dibujar luego es llamada desde **main()**; además se crean todos los objetos dentro de esta función.

glutSwapBuffers(). Sin esta función no podemos ver los dibujos.

void menú() Se crean el menú de nivel uno y nivel dos referente a salida del programa y mapa.

void Mapa(). función que define como debe estar creado el mapa y dentro de ella se encuentran las funciones de inicialización del modelo de display **glutInitDisplayMode()**, posición de la ventana en la pantalla glutInitWindowPosition(), tamaño del a ventana

glutInitWindowSize();y nombre de la ventana **glutCreateWindow()**; como funciones principales.

void Botones(). Se crean los botones a partir de la biblioteca mui

int leerDatos(). A esta función se le pasa como parámetro el tamaño del número y la cantidad de números que se desea leer. En el cuerpo de esta función se reserva espacio dinámicamente para la variables out, num y data; dentro de esta función definimos una variable tipo filename que se le asignara el contenido del fichero del cual se esta leyendo los datos a través de la función fopen (nombre del fichero, acción que se ejecuta).Si el archivo es diferente del nulo se ejecutan una serie de bucles for y obtenemos su contenido de lo contrario el fichero esta vacío y de todas formas mostramos su contenido.

int main() Función principal del programa que recoge los parámetros de la línea de comandos y los cuenta. Dentro de ella se ejecuta el programa principal. Todas las funciones anteriormente comentadas son globales. En e cuerpo de la función main se ejecutan todas las llamadas a la funciones que tienen que ver con la creación de la ventana principal y el menú, si es de interés crear botones sus funciones tendrán que estar dentro del cuerpo de esta función principal. Al final la función main devuelve un cero al sistema operativo.

glutMainLoop(). Función que le dice a Opengl que vuelva a dibujar la escena mientras que nada ocurra, ya sea un redimensionamiento de ventana o algún evento de teclado y en caso de que algo de esto ocurra se volverá a dibujar todo el escenario.