UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

CONTROL COMPUTACIONAL DE CÁMARAS MULTIESPECTRALES AEROTRANSPORTADAS PARA PERCEPCIÓN REMOTA.

T E S I S
QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
PRESENTA

WILFREDO MARTÍNEZ PAYÁN

Director: Dr. Ricardo Peralta-Fabi

Laboratorio de Ingeniería Aeroespacial
Instituto de Ingeniería
U.N.A.M. México, D.F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

ÍNDICE

RESUMEN

Esta tesis trata sobre un nuevo sistema de control de un grupo de cámaras de video utilizadas en campañas aerotransportadas de percepción remota, que facilita el proceso de adquisición de imágenes multiespectrales. Se describe el diseño, construcción y verificación de la interfaz que vincula las cámaras con una microcomputadora, así como el proceso de adquisición de imágenes multiespectrales.

Además, se describe con detalle tanto el circuito utilizado para marcar cada imagen de video con un número identificador, como el circuito para recuperar dicho número. Se incluye también, un sistema de posicionamiento global (GPS) y sensores de inclinación para obtener el centro de toma y el vector de apuntamiento de las cámaras para cada imagen adquirida durante un vuelo. Finalmente se describe el programa que realiza el control de todo el proceso de adquisición de imágenes, una serie de pruebas, la discusión de los resultados, y las conclusiones y recomendaciones que alcanza esta tesis.

AGRADECIMIENTOS.

Quiero agradecer al Dr. Ricardo Peralta la confianza manifestada a lo largo del desarrollo de esta tesis, así como el apoyo en la redacción del material; a Esaú Vicente y Fernado Segura la constante asesoría técnica durante la realización de este proyecto; a Jorge Prado su colaboración en la revisión del escrito; a Elías Vaquera, Armando Peralta e Iván Portugal todos sus comentarios y sugerencias; finalmente a Ma.Paulina Salas, Rodrigo Montúfar y a todos los integrantes del Grupo de Ingeniería Aeroespacial.

INTRODUCCIÓN

1. 1. Antecedentes

El Laboratorio de Ingeniería Aeroespacial, en donde se elabora este trabajo de tesis, viene trabajando en aspectos de percepción remota o teledetección para realizar estudios, ya sea desde el espacio o desde el aire, del territorio nacional y de sus costas. Los estudios de teledetección se realizan por medio de cámaras multiespectrales, es decir que observan la imagen en ventanas o sectores bien delimitados del espectro, principalmente en las bandas azul, roja y verde; en algunos casos se utiliza también banda del infrarrojo cercana y en otros más del infrarrojo térmico.

La percepción remota es importante porque es un medio con capacidad para analizar grandes extensiones de territorio, y ayuda a aclarar la importancia de los atributos de cada zona estudiada, para permitir la toma de decisiones racionales sobre diferentes aspectos de la organización del territorio, tales como infraestructura, ordenamiento, uso de suelo, tipo y extensión de cultivos, pérdida de suelo por procesos de erosión, explosión demográfica, afloramiento de minerales, contaminación en cuerpos de agua y en suelos, así como muchos otros más.

Dados los antecedentes del Laboratorio en procesamiento digital de imágenes, manejo de equipos ópticos y de sensores electrónicos, fue posible conjuntar suficientes elementos para elaborar una nueva técnica de percepción remota. En un inicio se realizaron experimentos con una sola cámara televisiva, para intentar discriminar objetos basándose en su reflectancia de la iluminación solar. Con la cámara se observa el brillo que presenta cada objeto visto desde diferentes bandas del espectro, de esta manera es posible discriminar objetos con diferentes colores o intensidades de gris. Con los primeros resultados se hizo evidente la importancia de realizar este tipo de observaciones para discriminación de objetos desde el aire y de ahí comenzó una nueva línea de investigación en teledetección para el estudio de diferentes fenómenos de importancia nacional [2].

Una vez obtenidas las imágenes con equipo de video, en donde se observan los objetos a través de diferentes filtros, las imágenes fueron digitadas por medio de circuitos dedicados a la conversión de señales televisivas de análogo a digital, y de esta manera ser manipuladas por medio de microcomputadora. El procesamiento de las imágenes se lleva a cabo por medio de programas que van realizando operaciones, aritméticas y lógicas ya sea punto a punto, o sobre una serie de elementos que forman parte de la imagen. Estos programas nos permiten extraer información cuantitativa del área que cubre el vuelo, tales como, perímetro, de forma y de tamaño y conformar criterios para detección, identificación, discriminación y clasificación de objetos sobre el terreno [4, 5, 6, 7,8].

Generalmente las campañas de teledetección se realizan con varias cámaras, cada una de ellas equipadas con un filtro espectral, de tal manera que normalmente es una serie de cámaras las que funcionan simultáneamente.

Fue necesario automatizar este proceso para asegurar que las cámaras captaran simultáneamente los mismos objetos, al mismo tiempo, para su posterior reconstrucción, y comparación por medio del análisis multiespectral. El sistema controlador de cámaras, debe cumplir con las siguientes funciones: durante la grabación el sistema registra un número único asociado a cada imagen de televisión, utilizando para ello el canal de audio de la cinta que graba las imágenes de video, de modo que posteriormente se pueda recuperar cada imagen tomada simultáneamente con cada una de las cámaras durante el vuelo. Eventualmente el sistema pretende controlar la videograbadora y el digitador para obtener imágenes específicas de cada banda de manera automática. Sin embargo, esto requiere de pasos subsecuentes que implican entender los programas que manejan las tarjetas comerciales de digitación y elaborar un circuito para realizar este control.

El sistema descrito fué desarrollado paso por paso y sometido a una comprobación experimental a bordo de una serie de vuelos de helicóptero, totalizando veintidos horas de vuelo, donde el sistema ejerció el control de hasta cuatro cámaras; cabe aclarar que a este sistema se le añadió un importante elemento con el que no sólo se asocia un identificador a cada toma de video, que ocurren a una tasa de 30 por segundo, sino que se asoció también un archivo en donde se registra la información geográfica, que incluye latitud, longitud, la hora exacta, altitud de vuelo, error

estimado y tiene datos de inclinación del equipo en dos ejes del plano horizontal, para conocer su vector de apuntamiento.

De esta manera se tiene asociado cada número identificador a los cuadros de video de la cinta. Esta información, es posteriormente utilizada para la reconstrucción de un mosaico que cubre la zona sobrevolada.

1.2 Videogrametría.

Los métodos tradicionales de percepción remota implican la utilización de imágenes fotográficas obtenidas a bordo de aeronaves, y fotografías o imágenes electrónicas obtenidas desde satélites en órbita, sin embargo, ambas técnicas tienen aplicaciones limitadas en los paises en desarrollo, más no por carecer de una función clara dentro del proceso de desarrollo, sino por el alto costo asociado a estas tecnologías, por ello en el Laboratorio de Ingeniería Aeroespacial se han dedicado en los últimos cinco años, esfuerzos para tratar de encontrar una solución sin incurrir en los altos costos normalmente asociados a la teledetección.

Los sensores a bordo de satélites se basan en un cristal fotosensible, fabricado por medio de técnicas de microelectrónica, de tal manera que detectan diferentes tipos de radiación. En los últimos 15 años se han desarrollado sensores conocidos como CCD (dispositivos de acoplamiento capacitivo) basados en monocristales de silicio con diferentes capas para formar una matriz de sensores, en donde cada elemento es capaz de captar la luminosidad que le hace llegar el sistema óptico. Esta matriz compuesta de renglones y columnas de pequeños detectores, generalmente del orden de unas 10 a 15 micras cada uno de ellos, es equipada con sistema eléctrico de lectura donde cada sensor de la matriz acumula una cierta carga, que es proporcional a la cantidad de fotones que llegaron a la superficie, y de ahí, al leer la secuencia de sensores localizados en un solo renglón se construyen las líneas de barrido de una televisión. La ventaja sobre las televisiones de tecnología tradicional (de vidicón) que es la que dominó durante los 30 años antes de los CCD, es que presentan una estabilidad geométrica absoluta, porque no hay movimiento de estos sensores sobre la superficie detectora. Por lo tanto, si mantenemos fijo el sistema óptico que enfoca las imágenes sobre el sensor, se puede confiar en que la imagen mantiene un registro geométrico absoluto, introduciendo el equipo televisivo en la tarea de teledetección que no se había podido lograr antes con las cámaras de vidicón [2,3].

Con el advenimiento de las cámaras CCD es posible comenzar a hacer percepción remota por medio de equipo televisivo. El equipo televisivo fué seleccionado siguiendo varios criterios: primero, el que funcionara con alta fiabilidad, para ello no se buscó equipo de investigación, sino por el contrario se buscó equipo de producción masiva, en donde la garantía de calidad resulta de manera natural; asimismo, el equipo presenta la ventaja de haber sido fabricado para usuarios no especialistas, por lo tanto tiene otra serie de ventajas, como sigue: tiene la grabadora integrada, capacidad de cambiar su distancia focal, puede además, aspecto muy importante para tomas en movimiento, ajustar el tiempo de exposición hasta 1/4000 de segundo; puede también ser controlado de manera remota la distancia focal, además de ser muy ligeras, de bajo consumo eléctrico y ser capaces de soportar trato relativamente rudo.

La combinación de esta serie de cámaras con los filtros dió lugar a una nueva tecnología que hemos denominado, en concordancia con la literatura internacional; videogrametría [2, 3, 8].

1.3 Resumen del sistema diseñado.

El sistema diseñado consta de la caja de cámaras, que lleva tres o cuatro cámaras de video con sus respectivos filtros espectrales, una de fotografía en color y una computadora portátil. A esta computadora le llegan las señales provenientes de los satélites del Sistema de Posicionamiento Global (GPS) que se van archivando en una secuencia de tiempo en el disco duro de la PC y como se mencionó, cada coordenada obtenida del GPS va asociada a un número identificador de imagen. La PC, a través de un circuito de control de cámaras que entra a través del control remoto de la cámara, enciende, apaga y graba en cada cámara, por medio de otro circuito independiente se accesa al canal de audio de la cámara por la entrada de micrófono para mandar un código a cada una de las imágenes grabadas durante el vuelo. Asimismo, la PC recibe de la cámara número 1 el pulso de sincronía vertical que juega el papel rector de todas las demás cámaras del sistema. Por otro lado, de la caja de cámaras salen cables que conducen la señal de video de cada una de las cámaras a un multiplexor mecánico, de tal modo que es posible observar el video en tiempo real de cualquiera de las cámaras dentro de la caja de cámaras, para asegurar su funcionamiento correcto.

2. DESCRIPCIÓN DEL EQUIPO UTILIZADO.

2.1 Introducción.

La selección del equipo utilizado para esta investigación siguió criterios centrales: el mantener el costo del sistema lo más bajo posible, la fiabilidad de su funcionamiento lo más alto posible, además de ser componentes fácilmente asequibles en el mercado nacional. Esto principalmente en relación a las cámaras y a los circuitos controladores de las señales de video, de audio y de pulsos de sincronía, así como en el caso de los multiplexores, los monitores y el sistema de computación. Por el lado del Sistema de Posicionamiento Global, también se adquirió el sistema multicanal de menor costo y fácil adquisición. Debido a estos criterios se asegura la repetibilidad de este sistema por cualquier otro grupo interesado, cuyo uso queda garantizado con una pequeña asesoría del Laboratorio de Ingeniería Aeroespacial que ha desarrollado el sistema.

2.2. Características de la microcomputadora y caja de expansión.

La computadora utilizada se basa en el microprocesador Intel 386, es una computadora AT compatible, que incorpora la capacidad de una computadora convencional en un sistema de sólo 3.3 kilogramos de peso. Tiene una velocidad de reloj de 20 megahertz y que tiene la virtud de conmutar a 4 megahertz durante tiempos que no se utiliza, ahorrando energía. Sus principales características son la presencia de una pantalla de cristal líquido con iluminación trasera, que nos permite la clara observación de los parámetros y las gráficas representadas en diferentes condiciones de iluminación, a las que se somete durante los vuelos de prueba. Además presenta la virtud de requerir un muy bajo consumo de potencia, en comparación con cualquier otro tipo de monitor.

El sistema cuenta con dos canales en serie RS232 de tal modo que en uno de ellos está conectado el receptor GPS y en el otro la comunicación con el circuito del

control del canal de audio. La computadora funciona con baterías propias de Niquel-Cadmio, durante períodos que alcanzan hasta dos horas coincidiendo con el tiempo máximo de la mayoría de los vuelos a los que fue sometido este sistema de prueba. La micro está equipada con discos de 60 y 1.44 MB, y cuenta con un compartimiento para añadir memoria de tipo DRAM, y una serie de indicadores que describen el estado de funcionamiento en la pantalla de las baterías, la velocidad del procesador, y lleva el teclado incluido. Por ser una computadora portátil es evidente que su diseño incluye tolerancia a vibraciones, a las cuales normalmente esta sometida durante el vuelo. La computadora está equipada con una caja de expansión que tiene dos ranuras AT y una XT compatible. Ambos equipos se conectan por medio de un conector con 100 terminales cada uno.

También es importante aclarar que debido a la utilización del equipo a bordo de aeronaves la caja de expansión va provista de un conector adicional, especialmente adaptado con el objeto de alimentarlo por medio de un banco de baterías.

2.3 Sistema de posicionamiento global (GPS)

El sistema de posicionamiento global es un sistema de autoevaluación basado en la interpretación de emisiones de radio de una constelación de satélites que orbitan la Tierra constantemente. El sistema proporciona suficiente información para navegación precisa, continua, con cobertura para todo el planeta y funciona en todo tipo de clima: se puede utilizar en tierra, mar y aire. Los satélites en la constelación GPS orbitan la Tierra cada hora y media a una distancia aproximada a 18000 Km de la superficie [21, 22].

A diferencia de los sistemas basados en tierra, el sistema de satélites GPS cubre grandes áreas, debido a su altitud, y debido a que sus señales están libres de interferencias de la geografía local. Están colocados en una constelación de tal modo que siempre, cuando menos cuatro satélites, serán vistos simultáneamente por los receptores GPS en cualquier localidad sobre la Tierra y durante 24 horas al día. El número visible de satélites es importante porque cuatro satélites se necesitan para poder calcular la posición que incluye la longitud, latitud y la altitud, además de la hora precisa.

Utilizando estos mismos receptores de GPS es posible realizar una técnica conocida como GPS diferencial, es decir, que cuando dos receptores se encuentran en una localidad georeferenciada, (punto geodésico donde se conocen con precisión las coordenadas geográficas) es posible nulificar el tipo de errores tan comunes en los sistemas de recepción que tienen que ver principalmente con retrasos de la propagación de las señales de los satélites a través de la ionósfera y la tropósfera; errores relacionados a las efemérides de los satélites, es decir, los datos precisos de su posición orbital, además de errores en los relojes a bordo de los satélites. Esta acumulación de errores impiden que la medición sea absolutamente precisa y es necesario recurrir entonces a la técnica de GPS diferencial, donde los dos receptores pueden restar el error que generan las causas mencionadas [22].

Por medio del GPS es posible conocer la posición de las tomas de imágen cada segundo durante el vuelo del helicóptero o la aeronave, se puede también fijar la trayectoria real seguida por la aeronave y al alimentar estos códigos en el canal de audio de cada cámara, para eventualmente reconstruir la posición geográfica del centro de toma de cada uno de los cuadros de video, en relación al terreno. Ésto nos permite también hacer una relación tridimensional y alimentar esta información al piloto automático de la aeronave. Se adquirieron dos sistemas GPS, dos receptores de cinco canales cada uno, en donde cuatro canales están permanentemente registrando parámetros, mientras se busca la configuración óptima por medio del quinto canal y se sustituye uno de los cuatro canales en utilización en el momento que alguno de los satélites se coloca en una posición más ventajosa para resolver las ecuaciones de autolocalización.

Entre las características operativas del sistema adquirido para el trabajo de esta tesis, se puede incluir: el sistema requiere de 30 segundos para adquisición del almanac, (datos de navegación de la constelación de satélites) el tiempo, la fecha y la posición inicial; en el caso de posicionamiento en tres dimensiones, es necesario duplicar este tiempo hasta 72 segundos. Estando lejos de una estación georeferenciada, la autolocalización puede demorar de 8 a 12 minutos. La máxima velocidad a la que puede viajar este receptor es de 1000 a 1510 kilómetros por hora, por lo cual este sistema cumple con todos los requisitos que pudiéramos tener en el futuro. Su máxima aceleración es de 2 G, dos veces el valor de la gravedad, por lo que tampoco se anticipa problemas con ese parámetro. La precisión de autoposicionamiento es en el plano horizontal de 25 metros RMS y de 30 metros RMS con altitud incluida. La

Utilizando estos mismos receptores de GPS es posible realizar una técnica conocida como GPS diferencial, es decir, que cuando dos receptores se encuentran en una localidad georeferenciada, (punto geodésico donde se conocen con precisión las coordenadas geográficas) es posible nulificar el tipo de errores tan comunes en los sistemas de recepción que tienen que ver principalmente con retrasos de la propagación de las señales de los satélites a través de la ionósfera y la tropósfera; errores relacionados a las efemérides de los satélites, es decir, los datos precisos de su posición orbital, además de errores en los relojes a bordo de los satélites. Esta acumulación de errores impiden que la medición sea absolutamente precisa y es necesario recurrir entonces a la técnica de GPS diferencial, donde los dos receptores pueden restar el error que generan las causas mencionadas [22].

Por medio del GPS es posible conocer la posición de las tomas de imágen cada segundo durante el vuelo del helicóptero o la aeronave, se puede también fijar la trayectoria real seguida por la aeronave y al alimentar estos códigos en el canal de audio de cada cámara, para eventualmente reconstruir la posición geográfica del centro de toma de cada uno de los cuadros de video, en relación al terreno. Ésto nos permite también hacer una relación tridimensional y alimentar esta información al piloto automático de la aeronave. Se adquirieron dos sistemas GPS, dos receptores de cinco canales cada uno, en donde cuatro canales están permanentemente registrando parámetros, mientras se busca la configuración óptima por medio del quinto canal y se sustituye uno de los cuatro canales en utilización en el momento que alguno de los satélites se coloca en una posición más ventajosa para resolver las ecuaciones de autolocalización.

Entre las características operativas del sistema adquirido para el trabajo de esta tesis, se puede incluir: el sistema requiere de 30 segundos para adquisición del almanac, (datos de navegación de la constelación de satélites) el tiempo, la fecha y la posición inicial; en el caso de posicionamiento en tres dimensiones, es necesario duplicar este tiempo hasta 72 segundos. Estando lejos de una estación georeferenciada, la autolocalización puede demorar de 8 a 12 minutos. La máxima velocidad a la que puede viajar este receptor es de 1000 a 1510 kilómetros por hora, por lo cual este sistema cumple con todos los requisitos que pudiéramos tener en el futuro. Su máxima aceleración es de 2 G, dos veces el valor de la gravedad, por lo que tampoco se anticipa problemas con ese parámetro. La precisión de autoposicionamiento es en el plano horizontal de 25 metros RMS y de 30 metros RMS con altitud incluida. La

posición vertical tiene una precisión de 50 metros RMS y la de la velocidad es de .15 nudos RMS [21].

Las características eléctricas requieren de 9 a 16 volts de corriente directa y su consumo es 235 miliamperes con 12 volts. La interfaz que presenta es el estándar industrial de comunicaciones seriales RS-232 y el baudaje es seleccionable entre 1200 y 9600.

La instalación de la antena del GPS se realizó para este experimento junto al rotor de cola de un helicóptero Mi8, de tal modo que las aspas del rotor no interferían con su recepción debido a la frecuencia de paso de las aspas metálicas frente a la antena, y su equivalente en apertura de malla, en ningún momento se notó problemas de recepción durante las 22 horas de vuelo.

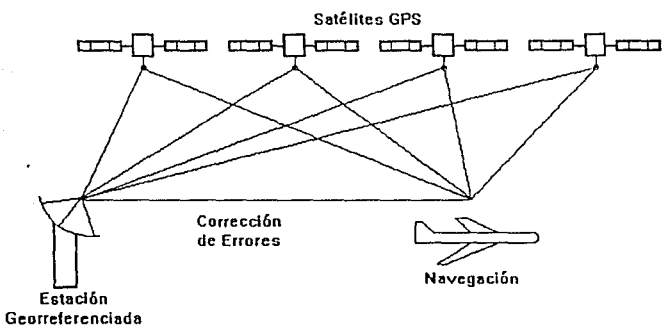


Figura No 1. Ilustración de la técnica GPS diferencial.

2.4 Cámaras multiespectrales.

Las cámaras multiespectrales, como su nombre lo indica, detectan la imagen de un objeto en diferentes sectores del espectro. En este caso se utilizan cámaras que tienen tres filtros, uno de ellos centrado en los 450 nanómetros, que es el filtro azul y

que tiene 80 nanómetros de ancho de banda a 50% de su transmitancia. Asimismo se incluye un filtro cuyo centro está en 550, que es verde y otro en 650 que es el rojo. Se hace uso de estas tres ventanas espectrales por ser las más comunes a bordo de aeronaves de percepción remota y satélites, aún cuando es posible instalar en el mismo sistema otro tipo de filtros que abarquen principalmente el infrarrojo cercano; dadas sus logitudes de onda, la ventana espectral está en 750 centrada y en 850 nanómetros [2,3].

Utilizando filtros de mayor longitud de onda, como el de 950 nm no alcanza la sensibilidad y/o filtros internos de la cámara, para la detección de imágenes, por lo que la mayoría de las imágenes tomadas para esta investigación fueron tomadas en las ventanas azul verde y rojo.

Como se menciona en la sección 1.2 sobre videogrametría, las cámaras basadas en dispositivos de estado sólido son capaces de mantener una alta estabilidad geométrica, por lo que si las cámaras tienen un eje común, es decir que el eje óptico de cada una de las cámaras es paralelo, las cámaras captarán el mismo cuadro, sin embargo, cada una de ellas tendrá una respuesta diferente ante el objeto debido al valor de brillo que cada una de estas ventanas espectrales capta de un mismo objeto.

En la figura 1 del apéndice se muestra la posición de las ventanas espectrales utilizadas en estas investigaciones y la figura 2 muestra un ejemplo donde la misma zona ha sido tomada simultáneamente con las tres cámaras en cada ventana del espectro y muestra también la clasificación automatizada de los objetos contenidos en una de estas imágenes.

Teniendo tres imágenes registradas de la misma zona de estudio, se procede a su procesamiento digital a través de circuitos que recibiendo la señal de video muestrean cada línea de barrido de la cámara. Son 525 líneas para el estándar NTSC (National Television Standard Committe) aunque el período de retroceso vertical (fly back) nos reduce a 483 líneas solamente, por lo que el proceso de digitación resulta en una imagen en donde las 483 líneas se manejan como renglones independientes y en cada una de las líneas de barrido puede haber como máximo 426 valores o tonos de gris formando una matriz de 483 por 426 elementos de imagen o utilizando el nombre más aceptado llega a cerca de 206 mil pixeles por imagen (palabra que proviene de la contración del inglés de picture element) [14]. Para cada punto o cada pixel de una

imagen el valor de brillo va de 0 a 256 valores que en cada banda tendrán un valor numérico diferente; porque para cada ventana del espectro un elemento del objeto tendra una diferente cantidad de luz reflejada en cada ventana particular, y es precisamente con esta diferencia en el valor numérico de los tonos de gris de cada pixel, que se pueden hacer discriminaciones espectrales de diferentes zonas comprendidas dentro de la imagen.

Las imágenes así obtenidas son almacenas en medios magnéticos u ópticos para su posterior procesamiento por medio de programas de procesamiento digital de imágenes y de sistemas de información geográfica. Cabe recordar que el propósito central de esta tesis es asociar a cada toma un identificador que se anota en el canal de audio de cada una de las cámaras durante la grabación de tal modo que sea posible recuperar imágenes en diferentes ventanas espectrales tomadas simultáneamente del mismo objeto. Posteriormente estas imágenes serán marcadas no sólo con el valor numérico guardado en el canal de audio sino en asociación a las coordenadas cartográficas obtendias por medio del sistema de posicionamiento global, del cual se va haciendo un archivo que incluye posiciones y el número de identificación de cada una de las imágenes tomadas, con una tasa normal de 30 imágenes por segundo [8].

3. ELECTRÓNICA PARA CONTROL DE FUNCIONES Y ADQUISICIÓN DE DATOS, Y PROGRAMA DE SUPERVISIÓN Y CONTROL.

3.1 Introducción al control de funciones y adquisición de datos.

Para controlar las funciones de las cámaras, tales como iniciar grabación, detener grabación, reproducir, regresar cintas, etc. se diseñó una interfaz que establece la comunicación entre la microcomputadora, el control remoto de las cámaras y las cámaras de video. La comunicación entre la interfaz diseñada y la microcomputadora se realiza mediante las instrucciones en lenguaje de máquina de entrada/salida, que toda microcomputadora incluye; y es mediante estas instrucciones que se realiza el control de las funciones de las cámaras.

Para asegurar la comunicación entre el control remoto y las cámaras de video, se prescindió del sensor infrarrojo por no ser un medio práctico (para nuestra aplicación) la manera como se utiliza en este tipo de cámaras, por lo que se acondiciona la señal para ser transmitida por cable. Los comandos se envían a las cámaras secuencialmente por medio de un multiplexor, así se acondiciona un solo receptor de control remoto para controlar todas las cámaras.

Durante el proceso de grabación a cada imagen se le asocia un número identificador para su posterior recuperación, la sincronización de este proceso: imagen asociada con un número único, se realiza basándose en el pulso de sincronía vertical de la cámara uno. Para obtener este pulso se alimenta la señal de salida de video de esta cámara a un circuito convencional separador de sincronía (LM1881) y así evitar alterar la compleja integridad de la cámara [24].

La microcomputadora registra el número de imagen y este se transmite al canal de audio de las cámaras, vía puerto serie (formato RS-232). Para ello se diseño un circuito que transforma el formato binario de comunicación del puerto serie a una señal comprendida dentro del ancho de banda del canal de audio (6-8 KHz, modulado en

FSK). Por lo tanto, este circuito se alimenta del puerto serie y su salida se envía a la entrada del micrófono externo de las cámaras, es decir, la grabación del número identificador de imagen es simultánea en todas las cámaras.

La recuperación del número identificador de imagen se realiza también vía

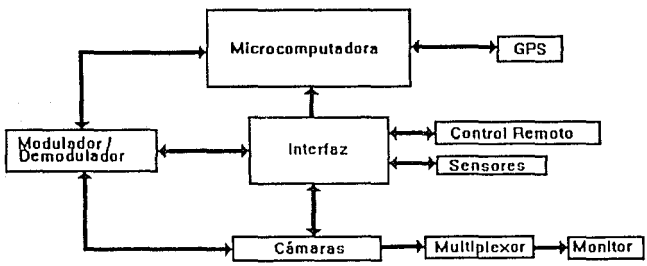


Figura No2. Diagrama general de las conexiones para el control de cámaras.

puerto serie de la microcomputadora. Para ello se utiliza un circuito (que incluye un PLL, XR210) que transforma la señal FSK de salida de audio hacia los audífonos (de las cámaras) al formato RS-232 del puerto serie, y así, accesar el número de cada imagen que aparece en el monitor.

La interfaz diseñada cuenta también con un convertidor analógico-digital de cuatro canales para eventualmente poder adquirir datos de sensores de inclinación en el plano de vuelo (alabeo y cabeceo de la aeronave), y con ello determinar la orientación de apuntamiento de las cámaras en el momento de adquirir las imágenes georreferenciadas con el sistema arriba descrito.

3.2 Interfaz con la microcomputadora para control de funciones.

Una instrucción en lenguaje de máquina de entrada/salida tiene asociada una dirección y un dato (instrucción/dirección/dato), al ejecutarse esta instrucción la dirección y el dato son habilitados en los ductos correspondientes, además de habilitarse la señal de salida IOW (input/output write) o de entrada IOR (input/output read). Como es sabido las señales: direcciones, datos, IOW e IOR son muy importantes para el diseño de interfaces con microcomputadoras [9,11].

Debido a que el acceso a la interfaz diseñada se realiza por instrucciones en lenguaje de máquina de entrada/salida, se requiere de una etapa de decodificación del ducto de direcciones para evitar el conflicto con otras tarjetas instaladas en la microcomputadora, tales como el puerto serie, el puerto paralelo, etc. Esta etapa habilita o deshabilita la interfaz al ejecutarse una instrucción en lenguaje de máquina basándose en el ducto de direcciones y las señales IOW e IOR, y así, responder únicamente a la instrucción de entrada/salida que contenga la misma dirección que la interfaz.

La figura 3 muestra el diagrama de bloques de la interfaz diseñada. Para evitar alterar el funcionamiento de la microcomputadora es necesario mantener la interfaz en circuito abierto (alta impedancia) con el ducto de datos, y sólo mantener continuidad cuando se accesa la interfaz. Para ello se utilizó un circuito de tráfico bidireccional (74LS245) que tiene la característica de presentar una alta impedancia al estar deshabilitado.

La capacidad de almacenamiento la constituyen dos registros de 8 bits. Cada bit del primer registro se utiliza para abrir o cerrar un interruptor analógico conectado al control remoto. El control remoto consta de ocho interruptores cada uno de los cuales se utiliza para transmitir a la cámara una función diferente. Para que las cámaras realicen una de estas ocho funciones (grabar, adelantar, regresar, etc.) basta con cerrar uno de los interruptores durante un tiempo determinado. Como se utiliza un solo control remoto y los relojes de las cámaras no son de precisión ni estan sincronizados, se añadió un multiplexor analógico para conmutar el control remoto con cada una de las cámaras, de esta manera las cámaras no se comunican. Los dos primeros bits del segundo registro controlan este multiplexor.

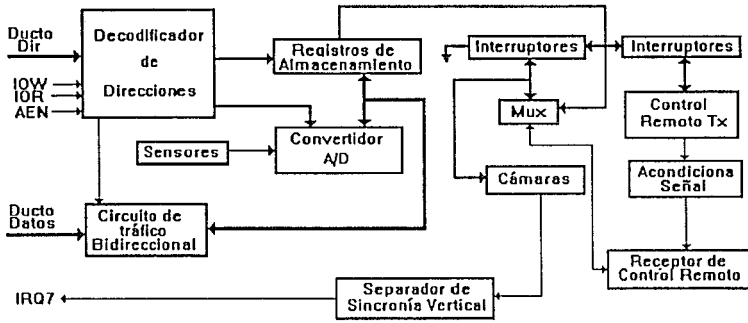


Figura No3. Diagrama de la interfaz para control de funciones y adquisición de datos.

Una característica del tipo de cámaras utilizadas, es que después de cinco minutos de no ejecutar una función se apagan automáticamente, con la finalidad de ahorrar baterias, quedando la línea de comunicación con el control remoto en 5 Volts. Una forma de volverlas a encender es mandando dicha línea a tierra (cero volts) un instante, ello se realiza mediante cuatro interruptores analógicos que unen la línea de comunicación de cada cámara con tierra; estos interruptores se abren o cierran mediante los últimos cuatro bits del segundo registro de almacenamiento.

Finalmente los dos bits restantes se utilizan para controlar el multiplexor analógico que conmuta el circuito demodulador (PLL) con los cuatro canales de audio de las cámaras, lo cual se explicará más adelante.

Para almacenar una palabra de 8 bits en uno de los registros de almacenamiento de la intefaz diseñada, basta enviar el dato por medio de la instrucción en lenguaje de máquina de entrada/salida con la dirección correspondiente. A cada registro corresponde una dirección diferente. La señal que emite el control remoto es una señal cuadrada con cierta distorsión, esta pasa a través de un circuito (dos transistores que

operan en corte/saturación) que elimina la distorsión original y la transmite por cable al receptor del control remoto para establecer la comunicación con las cámaras.

La interfaz también cuenta con un convertidor análogo-digital (A/D) de cuatro canales, ocho bits, 35 microsegundos de tiempo de conversión, reloj interno, salida de tres estados, y compatible con microprocesadores para adquirir datos de sensores de inclinación en el plano de vuelo (alabeo y cabeceo de la aeronave): para poder determinar el vector de apuntamiento de las cámaras.

Para realizar una lectura del convertidor A/D primero se envía una instruccción de salida, y como dato, el número del canal que se utilizará en la conversión (out/dir/canal), 35 microsegundos después se puede leer el valor por el ducto de datos mediante una instrucción de entrada (in/dir/) [9].

3.3 Grabación del número identificador de cada imagen.

Se usa, según se ha descrito, el conector de entrada de micrófono de las cámaras para grabar el identificador de la imagen en la cinta (en la pista de audio). Por esta razón la señal debe acondicionarse dentro del intervalo de amplitud y frecuencia para señales normales para un micrófono. La máxima amplitud de la señal es de 50mV (voltaje de saturación), ya que para mayores amplitudes la señal se distorsiona; la respuesta en frecuencia se muestra en la siguiente gráfica, donde puede apreciarse que el canal de audio de la cámara y la cinta en su conjunto, actúan como un filtro pasa bajas con una frecuencia de corte alrededor de los 10Khz.

Para conocer la respuesta en amplitud y frecuencia del canal de audio de la cámara y cinta, se llevó a cabo una evaluación de señales por medio de un generador de funciones, una cámara y un osciloscopio de dos canales. Primero, se utilizó una señal de 1Khz variando su amplitud hasta alcanzar el nivel de saturación. Posteriormente, se alimentó una señal de amplitud fija y menor que el voltaje de saturación, variando su frecuencia de 100Hz hasta 20Khz. La Figura No.5 muestra el diagrama correspondiente.

El número identificador de cada imagen es transmitido por el puerto serie de la microcomputadora, esta señal es una secuencia de unos y ceros en formato RS-232, un nivel de voltaje para el uno lógico (-12V) y otro nivel de voltaje para el cero lógico

(12V), quedando así una señal tipo cuadrada, es decir, con componentes de alta frecuencia que están fuera del ancho de banda del canal de audio. Por esta razón se requiere de un circuito que convierta esta señal a otra que reuna las características eléctricas, pero sin perder la información (secuencia de unos y ceros).

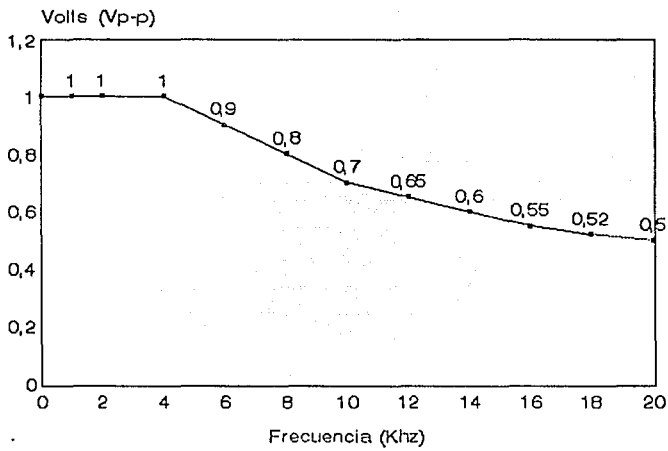


Figura No4. Gráfica de respuesta en frecuencia del canal de audio de la cámara y cinta en conjunto.

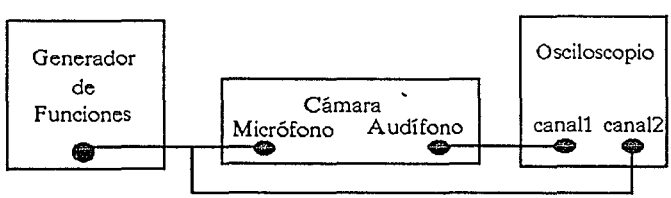


Figura No5. Diagrama del experimento realizado para determinar la respuesta en frecuencia del canal de audio.

Por medio de la técnica de manipulación por corrimiento de frecuencia FSK (Frecuency Shift Keying), se transforma la señal del puerto serie a una senoide de amplitud fija, pero que varía entre dos frecuencias, ésto es, la señal es de una frecuencia para un uno lógico y de otra frecuencia para un cero lógico [13]. La señal transmitida a las cámaras es de 45mV de amplitud, con frecuencias de 6KHz y 7.2Khz para el uno y cero lógico respectivamente, cumpliendo así con las requerimientos necesarios. Una de las propiedades principales de las señales en FSK es que presentan alta inmunidad al ruido, razon por la cual se utilizó, ya que el sistema opera en un ambiente de alta interferencia electromagnética.

La velocidad de transmisión seleccionada para garantizar la recuperación de la información es de 2400 bits por segundo, posibilitando grabar 6.6 palabras en cada imagen. El protocolo de comunicación se estableció como sigue: una palabra de control al comienzo de la imagen, seguida por una secuencia de tres palabras de ocho bits que especifican el número identificador de la imagen, durante las 2.6 palabras restantes no se transmite nada, tiempo muy importante para sincronización en el proceso de recuperar la información, imagen-identificador.

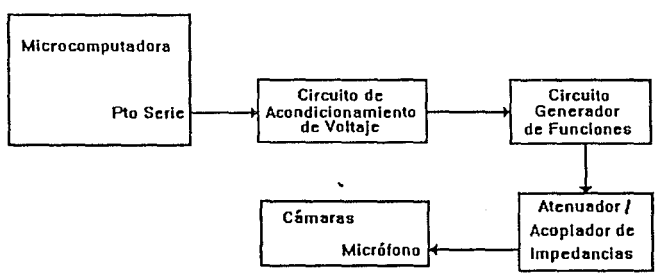


Figura No6. Diagrama del modulador en FSK.

Para obtener la señal en FSK se utiliza un circuito generador de señales (XR8038), donde la frecuencia de la senoide generada depende del nivel de voltaje en una de sus terminales; la señal del puerto serie se acondiciona a través de un circuito (un transistor que opera en corte/saturación y dos potenciómetros) que genera los niveles de voltaje necesarios para obtener las frecuencias de 6KHz y 7.2 KHz. La señal del generador se atenúa en amplitud hasta 45mV, y se acoplan impedancias entre el circuito y las cuatro cámaras, esto se debe a que las entradas de micrófono mantienen niveles de DC que no tiene el circuito, cuyo acoplamiento se logra con capacitores. La Figura No.6 muestra el diagrama del modulador en FSK.

3.4 Lectura del número identificador de cada imagen.

Manteniendo el criterio de no alterar la integridad de las cámaras, se utiliza el conector de salida de audio para audífonos, que todas las cámaras incluyen, para recuperar la información que fué grabada en la pista de audio.

La información en las cintas modulada en FSK, debe ser transformada al formato RS-232 para ser leida por el puerto serie de la microcomputadora. La Figura No.9 muestra el diagrama de bloques del circuito utilizado para dicha transformación.

Se usa un multiplexor analógico (CD4051), controlado por la interfaz descrita en la sección 3.2, para conmutar los canales de audio de las cuatro cámaras con el circuito demodulador de FSK, recuperando así la información de cada cámara en forma secuencial. Debido a que las señales que entregan las cámaras por el conector de audífonos es de 1Volt de amplitud pico a pico, no requieren ser amplificadas para ser alimentadas a este multiplexor.

Para eliminar ruido en la señal FSK, ésta se alimenta a un filtro paso banda tipo Butterworth [12]. La banda de paso del filtro se determinó por la regla de Carson para señales moduladas en FSK, la cual ayuda a especificar el ancho de banda requerido por una señal de este tipo (dicha regla es muy práctica) [10]. El funcionamiento del filtro con la señal modulada en FSK se evaluó experimentalmente. La Figura 7 muestra el diagrama de bloques del experimento.

La salida del filtro se alimenta a un circuito de amarre de fase PLL (Phase Locked Loop) el cual transforma la señal de variaciones en frecuencia (FSK) a una

señal con dos niveles de voltaje, que corresponden a las dos frecuencias de la señal en FSK. De esta manera se pasa de una señal senoidal con variaciones en frecuencia a una señal cuadrada con dos niveles de voltaje. La señal cuadrada corresponde a la secuencia de unos y ceros (información) que especifican el número identiticador de la imagen. Sólo resta acondicionar los niveles de voltaje de la señal cuadrada para que sea aceptada por el puerto serie.

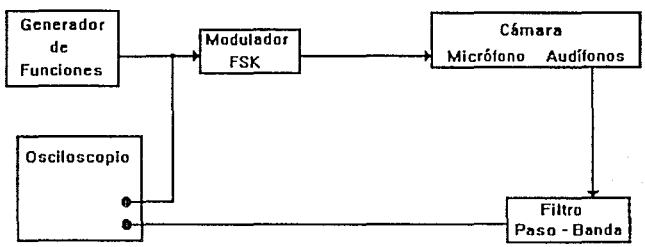


Figura No7. Comprobación del funcionamiento del filtro paso-banda con señales moduladas en FSK.

El PLL consiste principalmente de un circuito detector de fase, un filtro de lazo, un oscilador controlado por voltaje (VCO), y en muchas aplicaciones se incluye un amplificador [10,15]. La Figura No.8 muestra el diagrama de bloques de un PLL.

Se dice que el PLL esta "amarrado en fase" cuando la frecuencia del VCO (f_o) y la frecuencia de la señal de entrada al PLL (f_i) son idénticas ($f_o = f_i$), y sólo una diferencia de fase existe entre ellas. A la frecuencia de la señal del VCO, cuando no existe una señal de entrada al PLL, se le llama frecuencia de libre oscilación. Cuando se alimenta al PLL con una señal, éste entra en un periodo de adquisición de amarre de fase, esto es, la frecuencia de la señal del VCO perseguirá (aumentando o disminuyendo su frecuencia según el caso) a la frecuencia de la señal de entrada hasta

alcanzar el estado de amarre en fase. Al intervalo de frecuencias para las cuales el VCO puede amarrarse en fase con la señal de entrada se le llama "intervalo de captura". Una vez que se alimentó al PLL con una señal y este se amarró en fase con ésta, cualquier cambio en la frecuencia de la señal de entrada producirá un cambio en la señal del VCO para mantener el estado de amarre de fase, al intervalo de frecuencias en las que puede variar la señal de entrada y el PLL mantener la condición de amarre de fase se le llama "intervalo de mantenimiento". Para poder ser demodulada la señal de FSK por un PLL, las frecuencias de la señal FSK deben encontrarse dentro de los dos intervalos antes mencionados.

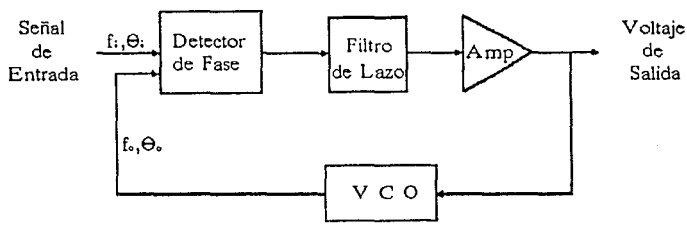


Figura No8. Diagrama de bloques de un PLL.

La función del detector de fase es multiplicar la señal de entrada con la señal del VCO:

$$A\text{sen}(\beta_o t+\mu_o)*2\cos(\beta_i t+\mu_i) = A\text{sen}[(\beta_o t+\mu_o)-(\beta_i t+\mu_i)] + A\text{sen}[(\beta_o t+\mu_o)+(\beta_i t+\mu_i)]$$

La salida del multiplicador se alimenta al filtro pasa-bajo de lazo. Cuando el PLL está amarrado en fase, las frecuencias de las señales son las mismas ($\beta_o=\beta_i$), por lo tanto el segundo término de la ecuación $A\text{sen}[2\beta_o t+\mu_o+\mu_i)]$ es atenuado por el filtro (si está bien calculado) debido a que corresponde a una señal del doble de frecuencia, quedando sólo el primer término $A\text{sen}[\mu_o-\mu_i)]$ el cual, como puede apreciarse, corresponde a un nivel de DC. Si la diferencia de fase es pequeña esta función es lineal, esto nos conduce a que el nivel de voltaje es proporcional a la diferencia de fase de las dos señales, el cual es amplificado (si es necesario) y alimentado al VCO controlando así, por el defasamiento de las dos señales, la frecuencia del VCO y el

amarre en fase del PLL [10]. En nuestra aplicación se utiliza un circuito para comparar el nivel de DC entregado por el filtro de lazo con un voltaje de referencia y así generar los dos niveles de voltaje de salida del PLL correspondientes a las dos frecuencias de la señal en FSK.

Se puede obtener un mejor entendimiento de la operación del lazo si consideramos que inicialmente el lazo no está amarrado en fase, pero que la frecuencia de la señal de entrada y la frecuencia del VCO son bastantes cercanas. Bajo estas condiciones la salida del filtro es una señal de baja frecuencia: $A\text{sen}[\beta_0 t - \beta_1 t + \mu_0 - \mu_1)]$ (debido a que β_0 y β_1 son casi iguales) y como esta señal es alimentada al VCO la frecuencia instantánea del VCO esta cambiando, si en algún instante de tiempo la frecuencia del VCO iguala a la frecuencia de la señal de entrada, resulta el estado de amarre en fase. En este instante, la salida del filtro adquiere el nivel suficiente para mantener la frecuencia del VCO en amarre con la frecuencia de la señal de entrada, este nivel resulta de la diferencia de fase de las dos señales. Si ocurre un cambio en la frecuencia de la señal de entrada, instantáneamente provocaría un cambio de fase de las señales y por lo tanto un cambio en el nivel de DC de salida del filtro, este corrimiento de nivel provocaría un corrimiento en la frecuencia de la señal del VCO manteniendo así el amarre en fase [16].

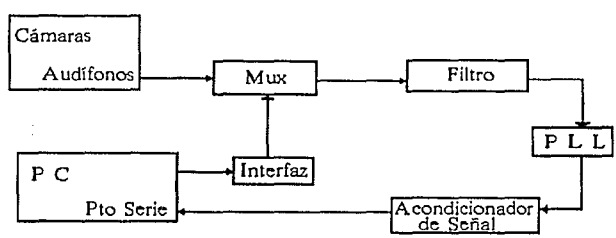


Figura No9. Diagrama de bloques del circuito demodulador de señales en FSK.

Finalmente la salida del PLL es acondicionada para generar los niveles de voltaje especificados en el formato RS-232 y así poder ser transmitida al puerto serie de la microcomputadora. Ésto se realiza mediante un circuito comparador de voltajes.

Se utilizó el componente XR210, el cual consta de un circuito PLL integrado en una sola pastilla y solo requiere de unos cuantos componentes externos para realizar la operación de demodulación de señales en FSK. Este componente reune las características de ser accesible en el mercado nacional, su costo es bajo, además de una gran eficiencia en la demodulación de señales en FSK.

3.5 Introducción al programa de supervisión y control.

Durante un vuelo de videogrametría, periódicamente se adquieren imágenes, simultáneamente el programa de supervisión y control debe cubrir una serie de tareas muy importantes, por sus repercuciones económicas, dentro de éstas cabe mencionar la siguientes:

Informar e interaccionar con el usuario (operador del equipo) en forma clara y oportuna, es decir, que el operador a través de comandos simples pueda controlar el funcionamiento de todo el sistema, asegurando de esta manera una labor correcta y permitiéndole el realizar otro tipo de actividades que se pudieran presentar durante el vuelo.

Segundo, la transmisión de comandos para la ejecución de funciones a las cámaras debe realizarse en forma supervisada, esto con el fin de evitar que el operador introduzca errores, tales como el enviar un comando en un contexto equivocado, así como también asignar el tiempo necesario para que las cámaras respondan a una instrucción.

Durante el proceso de grabación de las imágenes, la comunicación con el GPS no debe ser interrumpida, ya que esta información es de suma importancia en el momento de interpretar dichas imágenes. Si esta comunicación fuera suspendida se perdería la información de la ubicación de la aeronave al tomar las imágenes, el llamado "centro de toma", dificultándose el trabajo de interpretación.

Cada segundo el GPS transmite un bloque de información conteniendo los parámetros que previamente fueron especificados [21]. Una vez recibido el bloque, se realiza la lectura de los sensores de inclinación en el plano de vuelo, los datos de inclinación son anexados al bloque de información. La inclinación del equipo de cámaras, en dos ejes, sirve para calcular las coordenadas geográficas de cada imagen, ya que conociendo el centro de toma y la dirección de apuntamiento se puede asociar una imagen con información cartográfica. Finalmente se incluye en este bloque el número identificador de la imagen instantáneamente, conformando así un bloque de información completo.

El bloque completo debe aparecer en la pantalla de la microcomputadora de una manera clara y sencilla, con la finalidad de que el operador pueda confirmar el adecuado funcionamiento del sistema. También esta información debe ser almacenada en un archivo, el cual será utilizado en tierra durante la reproducción de las imágenes y la reconstrucción del vuelo.

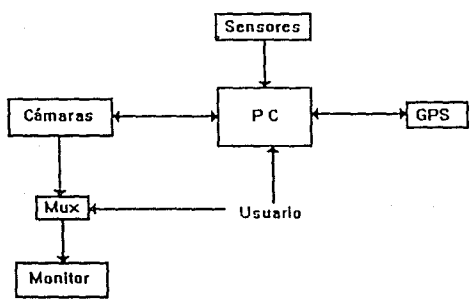


Figura No10. Diagrama del sistema de grabación.

Otro aspecto muy importante durante la grabación de las imágenes, es el no suspender la transmisión del número identificador de cada imagen al canal de audio de

las cámaras, de lo contrario, se perdería la referencia absoluta que existe entre la información en cinta y el archivo en disco duro de la microcomputadora.

En tierra, durante la reproducción de las imágenes y el trabajo de interpretación de las mismas, el programa debe permitir el control del equipo de una manera directa, y continuar con la transmisión de los comandos de ejecución de funciones a las cámaras de una forma supervisada.

Asimismo, recibir el número identificador de la imagen de las cuatro cámaras que estan siendo mostradas en los monitores, para buscar en el archivo el bloque de información correspondiente a la ubicación de la aeronave así como su orientación en el plano de vuelo en el momento de adquirir dichas imágenes. Este bloque de información aparece en la pantalla de la microcomputadora, mostrando así información que ayudará a la utilización de las imágenes.

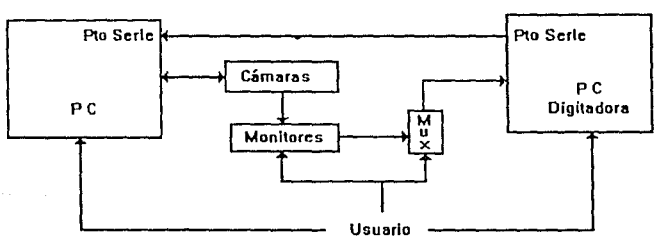


Figura No11. Diagrama del sistema de reproducción de las imágenes.

El programa debe también permitir la sincronización de las cámaras durante la reproducción de las imágenes, de esta manera en los monitores aparecerían las

imágenes tomadas simultáneamente en las diferentes ventanas del espectro, posibilitando el proceso comparativo del personal encargado en la interpretación de las mismas.

Finalmente, se digitan las imágenes a través de una microcomputadora con interfaz para digitación de imagenes de video. El programa debe mostrar el bloque de información correspondiente a cada imagen digitada. Esta información es importante ya que repercute en el tipo de procesamiento que posteriormente se realizará a cada imagen digitada.

El programa fué desarrollado en forma modular para facilitar el mantenimiento e integración con otros sistemas, y siguiendo la línea de conducta en cuanto a programación del Laboratorio. Los módulos principales del programa son los siguientes:

1. Inicialización.
2. Atención al usuario.
3. Transmisión de comandos a las cámaras.
4. Comunicación con el canal de audio.
5. Comunicación con el GPS.
6. Adquisición de datos de sensores.
7. Presentación de información en pantalla.
8. Manejo de Archivos.
9. Sincronización de cámaras.
10. Posicionamiento en un número de imagen de las cámaras.
11. Presentación de información de la imagen digitada.
12. Restauración del ambiente de la microcomputadora.

Como el Laboratorio de Ingeniería Aeroespacial desde hace años viene desarrollando toda su programación interactiva y de control en lenguaje C, para no perder compatibilidad, el programa de supervisión y control se desarrolló en este lenguaje por ser parte de todo un sistema del Laboratorio, y por supuesto porque reune las características necesarias para la funcionalidad del mismo.

3.6 Módulos del programa de supervisión y control.

Módulo de Inicialización. En este módulo se definen las variables globales más importantes, porque en estas se encuentra la información de todo el sistema. También se almacena el estado de la microcomputadora (vector de interrupciones) que pudiera ser modificado durante la ejecución del programa, con la finalidad de restablecerlo al finalizar el programa [17,18].

Módulo de atención al usuario. La atención al usuario se realiza mediante menús de ventanas. Cualquier operación sobre el sistema está contemplada en una opción de algún menú de ventana, de esta manera se facilita su labor, ya que el operador no necesita memorizar comandos, solo requiere el seleccionar la opción. El usuario procede por medio de opciones de los menús, utilizando las teclas con flechas y la tecla de entrada, o como en otros sistemas, presionando la primer letra de la opción o menú correspondiente. La tecla ESC se utiliza para cancelar algún proceso o para cerrar el menú correspondiente.

Módulo de envío de comandos a las cámaras. El envío de comandos a las cámaras se realiza de manera supervisada para evitar el propiciar errores. No se transmiten a las cámaras comandos que no sean los adecuados para la operación de éstas. Asimismo, al enviarse un comando se considera el tiempo necesario para que las cámaras respondan a esa función antes de transmitirles otro comando, debido a que las cámaras tardan un tiempo determinado para pasar de un estado a otro (tiempo que tardan en embobinar la cinta, etc.), esto es, durante los instantes de transición no se permite la transmisión de comandos a las cámaras.

Módulo de comunicación con el canal de audio. La comunicación con el canal de audio se realiza a través del puerto serie número uno de la microcomputadora. A cada imagen se le asocia un bloque de cuatro bytes: el primero es un caracter de control que especifica el encabezado del bloque, los tres restantes bytes especifican el número identificador de la imagen. El caracter de control se utiliza para sincronizar la recepción del número identificador durante la reproducción de las cintas. Tanto la recepción del número identificador de la imagen (en tierra), como la transmisión de éste (en vuelo), se realiza a través de interrupciones, evitando así el tener que supervisar constantemente el estado del puerto serie para verificar si ha recibido un dato o si ya se transmitió el especificado (según el caso). Y más importante es, no

suspender la recepción o transmisión de este parámetro. El puerto serie está programado para operar con las siguientes caracteristicas: la velocidad de operación es de 2400 bits/seg, se utiliza paridad par, dos bits de arresto y se habilita la interrupción de recepción en tierra y de transmisión durante el vuelo. Cabe mencionar que la transmisión del número identificador durante la grabación de las imágenes se realiza cada dos pulsos de sincronía vertical de la cámara uno, debido que el primer pulso sincroniza el barrido de las líneas nones de la imagen y el segundo pulso sincroniza el barrido de las líneas pares de dicha imagen.

Módulo de comunicación con el GPS. La comunicación con el GPS se realiza a través del puerto serie número dos. Tanto la transmisión como la recepción de mensajes se realiza por interrupciones para evitar que esta tarea sea suspendida, así como el tener que supervisar el estado del puerto serie. Los mensajes son transmitidos en formato ASCII con el siguiente protocolo: un encabezado que especifica el tipo de mensaje, a continuación, el bloque de parámetros o datos que corresponden al tipo de mensajes, y finalmente, los caracteres de verificación de error y los de fin de mensaje. No todos los mensajes del GPS se decodifican, solo los referentes a la posición geográfica (tres dimensiones), hora, error estimado, y el de calidad de las señales recibidas de los satélites. Cualquier otro mensaje recibido no es decodificado y es mostrado en la pantalla dentro de la ventana de mensajes generales. El GPS proporciona este bloque de mensajes cada segundo [17,21].

Módulo de adquisición de datos de sensores. La adquisición de datos de sensores se realiza a través de la interfaz (especificada en la sección 3.2). Debido a que la lectura de sensores se realiza de manera muy rápida a través del convertidor A/D (con 35 microsegundos de tiempo de conversión), solo se leen estos parámetros para completar el bloque de datos, es decir, una vez que ya se hayan obtenido todos los datos del GPS.

Módulo de presentación de información en pantalla. La pantalla de la microcomputadora se dividió en tres zonas: zona de datos (GPS, sensores, número identificador), zona de menús, y zona de mensajes. La primera es actualizada cada vez que se tiene un bloque de información completo. La segunda es actualizada, si es necesario, por el módulo de atención al usuario, cada que el operador presiona una tecla. En la tercera zona se presentan mensajes, por ejemplo: el estado del sistema, los errores, los mensajes de GPS que no son decodificados, etc.

Módulo de manejo de archivos. Los archivos son manipulados (lectura y escritura) en forma binaria, debido a que ocupan menos memoria y su acceso es considerablemente más rápido, en consecuencia se pierde menos tiempo en esta tarea. Sin embargo se tiene la opción de cambiar el formato binario al formato ASCII, y así ser leido para otros fines, tales como la reconstrucción de la trayectoria del vuelo sobre el plano terrestre [17,19,20].

Módulo de sincronización de cámaras. Este módulo consta de un algoritmo para sincronizar las cámaras basándose en la cámara uno, es decir, que todas las cámaras muestren en sus monitores correspondientes la misma imagen (dentro de un margen de error aceptable). Ésto facilita el trabajo de interpretación de las imágenes, ya que se estará observando en los monitores la imagen del mismo objeto, pero en las diferentes ventanas del espectro.

Módulo de posicionamiento en un número de imagen. Para realizar esta tarea, primero se fija la cámara uno en la imagen especificada por el operador (dentro del intervalo de error), y después se llama al módulo de sincronización de cámaras. Ésto es con la finalidad de ahorrarle tiempo al operador en la localización de las imágenes que consideró más importantes durante las evaluaciones del contenido de las cintas.

Módulo de presentación de información de la imagen digitada. Dentro de este módulo la microcomputadora de control de cámaras presenta en pantalla, en forma continua, la información correspondiente al número identificador que recibe por el puerto serie del canal de audio de las cámaras. También, se tiene otra PC con una tarjeta digitadora, que transforma la señal analógica de video de las cámaras a un formato digital; para ésto se diseñó un programa que queda residente en la memoria de esta PC, que transmite por el puerto serie un caracter cuando se digita una imagen. Al recibir este caracter por el segundo puerto serie, la microcomputadora encargada del control de las cámaras, suspende la información en pantalla. La imagen seleccionada se muestra en el monitor del equipo digitador y la información de posición de la aeronave e inclinación en el plano de vuelo correspondiente a esta imagen se presenta en la pantalla de la microcomputadora de control de cámaras: mostrando así la información de la imagen que ha sido digitada.

Módulo de restauración del ambiente de la microcomputadora. Este módulo restaura el vector de interrupciones con las direcciones originales, que fueron modificadas por el programa de supervisión y control, restableciéndose así el estado en el que opera normalmente la microcomputadora [18].

4. PRUEBAS EXPERIMENTALES Y DISCUSIÓN DE RESULTADOS.

4.1 Pruebas experimentales.

Se realizaron una serie de pruebas en el laboratorio para evaluar el funcionamiento, el alcance y los posibles defectos de cada módulo del control de cámaras multiespectrales. En las siguientes secciones se describe el proceso con cierto detalle.

4.1.1 Pruebas sobre el control de funciones de las cámaras.

Es posible controlar las funciones de las cámaras con una microcomputadora (por medio de una interfaz) para poder automatizar el proceso de adquisición y posteriormente la muestra de imágenes multiespectrales seleccionadas, facilitando la localización de las mismas. El objetivo de la prueba es evaluar la interfaz diseñada (referida en la sección 3.2) en el control de funciones de dichas cámaras así como el desempeño de los módulos del programa de supervisión y control involucrados en esta función (referido en el capítulo 3).

Pruebas de laboratorio:

a) Se instaló la tarjeta de control de funciones en una de las ranuras de expansión de la microcomputadora, para verificar que esta no interfería con su funcionamiento. Para ello se ejecutaron diferentes programas y paquetes que realizan operaciones tales como: comunicación con otra PC (entrada/salida puerto serie), operaciones de entradas por teclado, lectura y escritura a disco y multiples instrucciones en lenguaje de máquina. El resultado observado fué que no se alteró el funcionamiento normal de la PC.

b) Para comprobar la parte de habilitación de la tarjeta se utilizó un programa que ejecuta una instrucción de salida cada que se presiona una tecla. Esta instrucción

de salida contiene la dirección que corresponde a la tarjeta, el dato enviado a la tarjeta también es mostrado en la pantalla, con la finalidad de que se pueda verificar con un vóltmetro los niveles de voltaje en los registros y comprobar que cada que se transmite un dato a la tarjeta éste es almacenado en los registros; cabe aclarar que el programa también incrementa el dato al presionarse una tecla. Los resultados obtenidos demostraron que siempre se almacenó correctamente en los registros el dato correspondiente. De las dos primeras pruebas podemos concluir que la etapa de decodificación de la tarjeta diseñada opera correctamente.

c) Se utilizó un osciloscopio y un amperímetro para caracterizar el tipo de señal que generan los botones del control remoto al ser presionados, con la finalidad de seleccionar los interruptores analógicos para transmitir estas señales. Asimismo, se caracterizó la señal que transmite el control remoto a las cámaras y viceversa para todas las funciones del control remoto, y así instalar el multiplexor analógico apropiado. Se probaron los componentes utilizando una tableta de pruebas, una cámara de video, un control remoto y el osciloscopio. Se observaron con el osciloscopio las diferentes señales a través de todo el arreglo al cerrarse algún interruptor, y verificando que la cámara realizara la función correspondiente al interruptor accionado. Los componentes analógicos responden perfectamente al tipo de señal a las que fueron sometidos, cubriéndose esta función eficazmente. Estos componentes analógicos son los utilizados en la interfaz diseñada (referirse a la sección 3.2).

d) Posteriormente se comprobó el funcionamiento del circuito acondicionador de la señal de control remoto para ser transmitida por cable. Para ello se utilizó una cámara, un control remoto, el circuito de acondicionamiento y un osciloscopio. Para esta prueba se aisló el emisor infrarrojo. Con el osciloscopio se comparó la señal de salida del control remoto (al ser presionado un botón), y la señal de salida del circuito acondicionador, verificando su frecuencia (ancho de los pulsos) y amplitud. Posteriormente se trasmitieron las diferentes funciones del control remoto a las cámaras, a través del circuito diseñado, para verificar que éstas realizaran la operación correcta. No presentandose ningún problema.

e) Para la prueba final se utilizó una PC, la interfaz diseñada (con todos sus componentes), dos cámaras de video y el control remoto adaptado. La microcomputadora opera bajo el programa de supervisión y control. Se procedió a la transmisión de comandos de ejecución de funciones a las cámaras verificándose su

buen funcionamiento; se transmitieron todos los comandos del control remoto por un largo período realizándose siempre la función deseada. Las rutinas de supervisión del programa se pusieron a prueba seleccionado del menú comandos que no se encontraban en el estado adecuado (mientras las cámaras ejecutaban otra función) y transmitiendo comandos uno tras otro para verificar que el programa estableciera los tiempos necesarios que las cámaras requieren para responder a una de las funciones antes de transmitirles el siguiente comando, obteniéndose resultados consistentes.

4.1.2 Pruebas sobre la escritura y lectura del número identificador de imagen.

Es posible marcar cada imagen de video en la cinta con un número identificador único usando la pista de audio simultáneamente a la adquisición de las imagénes multiespectrales, para posteriormente sincronizar el proceso de muestreo de dichas imágenes, y además, para relocalizar cada imagen junto con los datos de georreferencia que se encuentran en un archivo. El objetivo de estas pruebas es evaluar el circuito diseñado para la escritura y lectura del número identificador de la cinta así como los módulos del programa de supervisión y control involucrados en esta tarea.

Pruebas experimentales:

a) Esta primer prueba se realizó para verificar la escritura y lectura por el canal de audio de las cámaras. Para esta prueba se utilizó un generador de funciones, una cámara, un osciloscopio, el circuito modulador de FSK y el circuito demodulador de FSK (referidos en el capítulo 3). El generador de funciones se utilizó para alimentar al circuito modulador de FSK con una señal cuadrada de 1200Hz y con niveles de voltaje similares al formato RS-232, esta señal corresponde a la mayor frecuencia de la señal que pudiera transmitir el puerto serie operando a 2400 bits/seg; la salida del modulador se alimenta al conector del micrófono remoto de la cámara. Durante todo el período de grabación se observaron tanto la señal del generador de funciones como la señal modulada en FSK verificando sus frecuencias y amplitudes. Posteriormente se recupera la información de la cinta utilizando el conector de salida para audífonos y el circuito demodulador, observando en el osciloscopio la señal de salida del conector de audífonos y la señal de salida del circuito demodulador, nuevamente se verificaron sus frecuencias y amplitudes. El resultado de esta prueba fué que se recupera integramente la señal original de 1200Hz.

b) Esta prueba se realizó para comprobar el proceso de grabación y lectura del número identificador de cada imagen. Para esta prueba se utilizó una PC, el programa de supervisión y control, la interfaz diseñada, el circuito modulador de FSK, el circuito demodulador de FSK, el control remoto adaptado y un osciloscopio. La línea de transmisión del puerto serie de la PC está conectada al circuito modulador, y éste al conector de entrada para micrófono de las cámaras, la línea de recepción del puerto serie está conectada al circuito demodulador, y éste al conector de salida para audífonos de las cámaras. La interfaz diseñada se instaló en una de las ranuras de expansión de la PC, se conectó el control remoto a la interfaz y ésta a los conectores de entrada para control remoto de las cámaras; la PC opera bajo el programa de supervisión y control. Se seleccionó la opción de grabación del menú correspondiente, para observar durante todo el proceso de grabación, en el osciloscopio, las siguientes señales: la que transmite el puerto serie, la de salida del circuito modulador, y el pulso de sincronía vertical de la cámara uno (referirse sección 3.2, interfaz diseñada). Comprobándose que se transmiten las cuatro palabras por el puerto serie especificadas en el protocolo y que esta transmisión se realiza cada dos pulsos de sincronía vertical de la cámara uno. Después de un tiempo considerable se apagaron las cámaras, se regresaron las cintas, y finalmente se seleccionó del menú la opción de reproducir las cintas, se observaron durante todo el proceso de reproducción las siguientes señales: la que recibía el puerto serie, las señales de entrada al circuito demodulador (las 4 de audio) y el pulso de sincronía vertical de la cámara uno. Comprobándose que se reciben las cuatro palabras por el puerto serie especificadas en el protocolo y que esta recepción se realiza cada dos pulsos de sincronía vertical de la cámara uno. Cabe recordar que los números identificadores de cada cámara son mostrados en la pantalla de la PC, esto también permitió verificar el funcionamiento del sistema.

4.1.3 Pruebas sobre la adquisición de datos de sensores.

El objetivo de esta prueba es evaluar el funcionamiento del convertidor A/D en la lectura de datos de sensores así como los módulos del programa de supervisión y control involucrados en esta tarea. Debido a que la etapa de acondicionamiento de sensores se encuentra en proceso, se utilizaron potenciómetros y una fuente para alimentar al A/D con voltajes de referencia conocidos, se verificó que el módulo de lectura de sensores del programa realizara esta función adecuadamente. Cada vez que se realiza una lectura al A/D se muestra en pantalla el dato correspondiente

comprobando así su funcionamiento. El voltaje de referencia se varia de 0-5 Volts contínuamente, siendo este intervalo el máximo permitido para este convertidor. Cabe aclarar que la etapa de acondicionamiento de los sensores entrega un voltaje en el intervalo de 0-5 Volts el cual es directamente proporcional al ángulo de inclinación; la variación de grados para la cual es calibrada esta etapa va desde -10° a 10°.

4.1.4 Pruebas con el GPS.

El objetivo de esta prueba es comprobar el funcionamiento de los módulos de comunicación con el GPS del programa de supervisión y control. El GPS y la PC se comunican a través del puerto serie. Para esta prueba el receptor de GPS se encontraba en un punto georreferenciado, es decir, donde se conocen con toda precisión las coordenadas geométricas. Se transmitieron los mensajes correspondientes para inicializar al GPS, después de unos minutos se recibió cada segundo el bloque de información del GPS, se verificó que estos se decodificarán correctamente; esta información decodificada es mostrada en pantalla también cada segundo. Cabe mencionar que durante toda la prueba el error estimado promedio en la posición calculada por el GPS fué de 50m.

4.1.5 Pruebas con el equipo de digitación.

Para esta prueba se utilizó una PC con el programa de supervisión y control, la tarjeta de control de funciones, la tarjeta modulador/demodulador, otra PC con una tarjeta de digitación de imágenes y el programa de manejo de la tarjeta, un reproductor de cintas, cintas previamente marcadas (en la pista de audio) y el programa que queda residente en la memoria de la PC usada para digitar imágenes (referirse a la sección 3.6). El objetivo de la prueba fué verificar que el programa residente en memoria no alteraba el funcionamiento de la PC dedicada a la digitación y también verificar que cada que se seleccionaba una imagen para ser digitada (presionando una tecla específica), la información correspondiente a dicha imagen aparecía en la pantalla de la otra PC utilizada. Las conexiones fueron las siguientes: La PC con la tarjeta de digitación se conectó vía puerto serie con la PC de control de funciones; la salida de video del reproductor de cintas se alimentó a la tarjeta de digitación y la salida de audio a la tarjeta modulador/demodulador, esta tarjeta está conectada a la PC de control de funciones por el otro puerto serie. La PC de control de funciones opera bajo el programa de supervisión y control, la otra PC opera bajo el programa de manejo de

la tarjeta de digitación, sin embargo previamente se dejó residente en memoria el programa especificado con anterioridad. Se procedió a seleccionar diferentes imágenes para ser digitadas verificándose que cada que aparecía una imagen digitada en la pantalla de la PC en el otro monitor aparecía la información correspondiente, comprobándose su funcionamiento.

4.2 Pruebas de campo.

Las pruebas de campo se realizaron para evaluar tanto el funcionamiento, como el alcance y fuentes de errores del método propuesto para el control de cámaras multiespectrales. Las pruebas se realizaron en varias regiones de la isla de Cuba, a bordo de un helicóptero pesado Mi-8 totalizando 22 horas de vuelo, y se procesaron los datos en un laboratorio con especialistas en la materia estudiando las imágenes duarante dos semanas.

Se probó todo el sistema a su máxima capacidad, es decir, el control de funciones de las cámaras, la escritura y lectura del número identificador, la comunicación con el GPS, el manejo del archivo de información, el proceso de digitación, así como todo el programa de supervisión y control. Durante todas estas pruebas el sistema operó adecuadamente, aún cuando estuvo sometido por vez primera a vibración e interferencia electromagnética durante los vuelos (para atenuar estas perturbaciones se utilizaron cámaras de aire y cables blindados). Cabe aclarar que desafortunadamente no se utilizaron los sensores de inclinación en el plano de vuelo durante las pruebas por encontrarse todavía en proceso de acondicionamiento y calibración en el Laboratorio de Ingeniería Aeroespacial.

Algunos de los problemas que se presentaron fueron los siguientes: por ser un prototipo la tarjeta de modulación/demodulación esta contenía algunos cables soldados, dos de ellos se desoldaron por causas de vibración o por la constante manipulación, causando que las cámaras 3 y 4 no recibieran el número identificador durante algunos periodos. Otro problema fué que la cámara 3 no siempre respondía al comando de suspender el proceso de grabación, por razones todavía no aclaradas. Finalmente, por desconocer el funcionamiento detallado de la tarjeta digitadora y su programa de manipulación, la información que se asocia a cada imagen digitada contenía un error aleatorio, debido al desconocimiento del número de imágenes que avanza después de seleccionar una imágen por medio del teclado.

4.3 Discusión de resultados.

A continuación se presenta una discusión de los resultados obtenidos por el método propuesto para el control de cámaras multiespectrales.

a) Se marcó cada imagen de video con un número identificador único usando un circuito externo a las cámaras. El identificador permite la sincronización de las imágenes con la localización de la posición donde fué tomada cada imagen, y la inclinación de la cámara, ésto facilita el trabajo de interpretación de las mismas. Debido a que los relojes de las cámaras no estan sincronizados ni son de precisión el marcado de las imágenes se realiza basándose en la cámara uno, esto puede ocasionar un error máximo de una imagen en las tres cámaras restantes; volando a 140 Km/hr la aeronave recorre 1.2 metros cada imagen, siendo éstos despreciables. Por otro lado tratar de sincronizar los relojes implicaría entender el complejo funcionamiento de las cámaras para poder estimar una posible solución.

b) Se controló el funcionamiento de cada cámara a través de una PC y con circuitos externos a las cámaras, ésto permitió realizar la transmisión de comandos de una manera supervisada para evitar cometer errores y facilitar el trabajo del operador del equipo. Para solo adaptar un control remoto para todas las cámaras, la transmisión de funciones se realiza secuencialmente, la transmisión simultáneamente sólo se justificaría si los relojes de las cámaras estuvieran sincronizados.

c) Se reciben cada segundo los datos del centro de toma de las imágenes por medio del GPS, esto ocasiona un error, ya que 30 imágenes tendrán un mismo centro de toma. Sin embargo este error puede ser disminuido al trazar la ruta de vuelo seguida por la aeronave en tierra y dividir las distancias entre 30 posiciones, que es el número de imágenes obtenidas cada segundo. Cabe mencionar también que si se utiliza la técnica conocida como GPS diferencial el error del centro de toma es disminuido considerablemente, hasta ± 3 metros.

d) El dato de inclinación de la aeronave en el plano de vuelo se puede realizar para cada imagen y no cada segundo, debido a la alta velocidad de conversión del A/D (35 microsegundos) si esta información es almacenada en un archivo se puede

determinar el vector de apuntamiento de la cámara en el momento de adquirir cada imagen, disminuyendo el error de estimación del centro de toma.

e) El proceso de sincronización de las imágenes durante la reproducción de las cintas es aceptable, considerando que los relojes de las cámaras no estan sincronizados y que éstas no son equipo de presición. El máximo defasamiento presentado durante las pruebas fué de ocho imágenes con respecto a la cámara uno, ésto quiere decir que se pueden observar los mismos objetos en los diferentes monitores.

d) Al conocer a detalle el funcionamiento de la tarjeta de digitación se puede desarrollar un programa que interactúe con el programa de supervisión y control para automatizar el proceso de digitación de imágenes, además de reducir el error considerablemente ya que la imagen a digitar no la seleccionaría el operador sino la PC dedicada al control de las cámaras multiespectrales, y la función del operador sería la de supervisar el proceso.

5. CONCLUSIONES Y RECOMENDACIONES.

Dado que el principal objetivo de esta tesis, es presentar un sistema de control de cámaras de video para la producción secuencial de imágenes multiespectrales y georreferenciadas, con base en los resultados se pueden enumerar las siguientes conclusiones:

1. Para alcanzar los resultados aquí presentados, se ha desarrollado un dispositivo de control que garantiza la inclusión de un número identificador durante la grabación de video multiespectral en varias cámaras.

2. La descripción completa de este sistema, se presenta en los capítulos II y III, y ha sido descrito por primera vez en este trabajo de tesis.

3. La integración automática de la georreferencia o localización geográfica (X, Y, Z) de la aeronave, ha sido utilizada también por primera vez, para conocer el centro de toma de cada imagen de la secuencia de exposiciones videogramétricas.

4. Algunos de los resultados presentados en esta tesis, muestran la alta precisión del procedimiento de registro de datos en tiempo real, mejorando considerablemente los sistemas tradicionales en percepción remota.

5. La aplicación de este método de control proporciona automáticamente las bases para traslapar cartográficamente imágenes de video obtenidas durante una serie de pruebas a bordo de un helicóptero-laboratorio.

6. Se muestra que la interacción entre la información experimental en tiempo real y el personal coordinador de vuelo, a través del monitor a bordo, facilita el desarrollo de los vuelos ahorrando pérdidas innecesarias de imágenes y combustible, cuando algún otro parámetro del vuelo falla.

7. El filtrado por circuitería y por programación constituye un método eficiente y práctico para dar estabilidad a las señales analógicas adquiridas, que de otra forma requerirían de una tecnología costosa de acondicionamiento.

8. Desde el punto de vista de la videogrametría, al delegar el control de sus variables más importantes a un sistema computarizado, se mejora el desarrollo y aplicabilidad de esta novedosa técnica de teledetección.

La culminación de un trabajo de este tipo, necesariamente genera una serie de ideas que resultan del ejercicio del trabajo mismo. Estas ideas o prolongación del trabajo de tesis, se describen a continuación con posible beneficio de quien prosiga el desarrollo del tema aquí expuesto.

El autor de esta tesis considera que dado el grado de desarrollo obtenido en el experimento, sería conveniente, por un lado terminar su completa automatización por medio de un actuador electromecánico que efectue la corrección del apuntamiento de las cámaras videogramétricas, realizado hasta ahora manualmente. Por otro lado, la graficación de variables experimentales, durante y después del experimento, de tal modo que se muestre la historia completa de las variables con objeto de comparar los resultados del experimento con la información esperada del vuelo, promete ser el camino más fructífero.

Considerando que el objetivo del experimento es la adquisición y registro de imágenes de video y el desarrollo de técnicas de posicionamiento y georreferenciación de las tomas en color, se puede concebir el combinar la georrefencia de imágenes con mapas geográficos, utilizando el contorno de razgos destacados de la toma, para reconstruir automáticamente un mosaico de imágenes. Éste podría presentarse en perspectiva, con una dramatización del cambio de geometría segun los parámetros de vuelo (adecuadamente amplificados) que se lograrían con un procesamiento de la información almacenada. Esta información sería muy útil al piloto y navegante de cada misión.

APÉNDICE

```
/*
Tesis: Control Computacional de Cámaras Multiespectrales Aerotransportadas para Percepción Remota
Autor: Wilfredo Martínez Payán
Laboratorio de Ingeniería Aeroespacial
Instituto de Ingeniería, U.N.A.M
Noviembre de 1992
Programa de supervisión y Control.
*/


/* Librerías Utilizadas */
# include <dos.h>
# include <conio.h>
# include <stdlib.h>
# include <bios.h>
# include <stdio.h>
# include <math.h>
# include <string.h>
# include <io.h>


/* Constantes Generales */
# define DirMemVid (char far *)  0xB8000000;     /* Dir Memoria de Video */
# define VidNormal               0x07            /* Colores de caracteres en pantalla */
# define VidInverso              0x70
# define ColorMensaje            0x70
# define DirLatch1               0x302           /* Dir de Puertos usados */
# define DirLatch2               0x303
# define DirA_D                  0x304
# define Max_Frame               5               /* Máximo nivel de submenús */
# define ESC                     27              /* Tecla de Escape */
# define Modo_Off                0               /* Modo de operación de las Cámaras */
# define Modo_Vtr                1
# define Modo_Cam                2
# define NumeroCamaras           4               /* Número de cámaras utilizadas */
# define ImgSegTR6               0.6             /* Imágenes por segundo en rew/ff */
# define Cero                    0x00            /* Funciones de las cámaras */
# define Rec                     0x01            /* Rec y Stby el mismo boton */
# define Stby                    0x11
# define Zoom_menos              0x02
# define Zoom_mas                0x04
# define Fast                    0x08
# define Stop                    0x10
# define Rew                     0x20
# define Play                    0x40
# define Pausa                   0x80
# define GPGGA                   0x01            /* Mensajes del GPS */
# define PMGLB                   0x02
# define GPZDA                   0x03
# define PMGLG                   0x04
# define PMGLF                   0x05
```

```
/* Funciones de cada Módulo del Programa de supervisión y Control */

/* Módulos de Inicialización y Restauración del ambiente de la microcomputadora*/
void ambiente,( ) restaura( );

/* Módulo de Atención al usuario */
void activamenu( ), dibujaborde( ), despliegamenu( ), submenu( ), cierramenu( );
void respuesta( ), pulldown( ), salvavideo( ), restauravideo( ), ejecuta( ), vtr( ), cam( ), salir( );
int make_menu( ), esta_en( ), validacion( );

/* Módulo de Transmisión de comandos a las cámaras */
void enviafuncion( ), a_todas( ), regresacintas( ), adelantacintas( ), pausacintas( ), videocintas( );
void detenercintas( ), grabacintas( ), listascintas( ), enciendecamaras( ), acerca_camara( );
void aleja_camara( ), grabaenuna( );

/* Módulo de Comunicación con el canal de audio */
void com1img( ), com1reset( ), inicializacom1tx( )

/* Módulo comunicación con el GPS y Adquisición de datos de sensores*/
void buscadatogps( ), inicializacom2( ), com2reset( ),
void revisamensajegps( ), gpgga( ), pmglb( ), gpzda( ), pmglg( ), pmglf( );
int checa_gps( ), tipomensajegps( );
void generachecksum( ), tx_gps( ), al_gps( ), leecomandogps( ), inicializagps( );
void pidedatosgps( ), borragps( ), generachecksumtx( ), leeinclinometros( );

/* Módulo de Presentación de información en pantalla */
void pantalla( ),write_string( ),write_char( ),cls( ),borracursor( );
void clsmensajes( ), bordemensajes( ), datosapantalla( ), mensajes( ),pantallafuncion( );

/* Módulo de Manejo de archivos */
void obtienedatos( ), buscadatos( ), almacena( ), abrir( ), leedato(),cerrar( ), convierte( );
void posicionapuntero( );

/* Módulo de Sincronización de cámaras y posicionamiento en un número de imagen */
unsigned retardo(), NoImagen(), sinc_cam1_con();
void sincroniza(), al_numero_imagen();

/* Presentación de información de la imagen digitada */
void digitaliza(), tiempodato(), com2dig();

/* Variables Generales */
int num_menu, opcion, nueva_funcion, mensajesx, mensajesy;        /* De los Menus */
unsigned funcion[NumeroCamaras];
struct registro {                                                 /* Formato de Datos */
        float latitud, longitud;
        long altitud, tiempo, noimagen;
        int error, alabeo, cabeceo;
        } datos;
int modo_operacion;
char archivo[12];                                                 /* Nombre del archivo */
FILE *pfichero;                                                   /* Puntero al archivo */
size_t bytesdatos = sizeof(datos);                                /* Tamaño del formato de datos */
unsigned latch1,latch2;                                           /* Latch's de la interfaz */
```

```
long nocuadro;                                  /* # de imagen leido por com1 */
long camara1, camara2, camara3, camara4;        /* No de imagen de cada cámara */
unsigned conmuta;                               /* Conmutar canal de audio */
unsigned lee_dato;                              /* # de bytes recibidos de com1 */
unsigned inicio, byte1, byte2, byte3;           /* Comunicación por com1 */
unsigned txbander, tx_cual;                     /* Transmisión de # de imagen */
unsigned cual_gps, dato_gps, datos_gps;         /* Comunicación con el gps */
char cad_gps1[100], cad_gps2[100], mensaje_gps[100];
char caracter_gps[1];
int posicioncaracter = 0;
unsigned digita;                                /* Bandera de Digitación */
struct menu_frame {                             /* Marco de los menús */
        int startx, endx, starty, endy;
        unsigned char *p;
        char **menu;
        char *llaves;
        int borde, contador;
        int activo;
        } frame[Max_Frame], i;
char *Principal[] = {                           /* Menú principal */
                "Files > ", "Vtr  > ", "Cam  > " };
char *Files[] = {                               /* Menú Files */
                "Abrir ", "Cerrar", "Texto ","Salir "};
char *Vtr[] = {                                 /* Menú Vtr */
                "Regresar  ", "Adelantar ", "Pausa     ", "Video    ",
                "Detener   ", "Sincronizar", "En No. Img ", "Digitaliza ",
                "Tiempo/dato", "Fin      "};
char *Cam[] = {                                 /* Menú Cam */
                "Gps   > ", "Acercar ", "Alejar  ", "Listo  ", "Grabar ",
                "Encender", "Enc-Rec1", "Fin    " };
char *Gps[] = {                                 /* Menú Gps */
                "Inicializa", "Mensajes  ", "Comenzar  ", "Datos    ",
                "Reiniciar " };



/* Programa principal */
main( )
{
ambiente();                                     /* Establece las condiciones iniciales */
 for(;;) {                                      /* Ciclo de espera para salida */
   while(!kbhit());                             /* Espera a que presione una tecla */
   activamenu();                                /* Activa el menú  */
   switch(modo_operacion) {
        case Modo_Off : if(nueva_funcion) ejecuta();    /* Ejecuta función */
                        break;
        case Modo_Vtr : vtr();                  /*Modo reproducción de Imágenes */
                        break;
        case Modo_Cam : cam();                  /* Modo grabación de Imágenes */
                        break;
        default : ; }
        }
}                                               /* Fin de main( ) */
```

```
/* Establece las condiciones iniciales */
void ambiente( )
{
 outportb(0x0021,0xB8);                    /* Inicializa controlador de interrupciones */
 outportb(DirLatch1,0x00);                 /* Inicializa latch's de la interfaz */
 outportb(DirLatch2,0x00);
 borracursor();                            /* Borra el cursor de la pantalla */
 pantalla();                               /* Despliega la pantalla inicial */
 clsmensajes();                            /* Borra la ventana de mensajes */
 bordemensajes();                          /* Dibuja el borde de la ventana de mensajes */
 make_menu(0, Principal, "fvc", 3, 0, 68, 1);   /* Crea los marcos de menú */
 make_menu(1, Files, "acts", 4, 2, 68, 1);
 make_menu(2, Vtr, "rapvdseitf", 10, 3, 63, 1);
 make_menu(3, Cam, "gajlrenf", 8, 4, 66, 1);
 make_menu(4, Gps, "imcdr", 5, 6, 63, 1);

                                           /* Inicializa variables globales */
 modo_operacion = Modo_Off;                /* Modo de operación de la cámara */
 funcion[0] = 0;                           /* Función que realiza cámara1 */
 funcion[1] = 0;                           /* Función que realiza cámara2 */
 funcion[2] = 0;                           /* Función que realiza cámara3 */
 funcion[3] = 0;                           /* Función que realiza cámara4 */
 nueva_funcion = 0;                        /* Si elige una nueva función */
 num_menu = 0;                             /* No de menú desplegado */
 opcion = 0;                               /* Opción del menú desplegado */
 mensajesx = 19;                           /* Posición x,y de la ventana de mensajes */
 mensajesy = 1;
 archivo[0] = 0x00;                        /* Nombre del archivo utilizado */
 datos.latitud = 0x00;                     /* Valores iniciales principales */
 datos.longitud = 0x00;
 datos.altitud = 0x00;
 datos.tiempo = 0x00;
 datos.error = 0x00;
 datos.alabeo = 0x00;
 datos.cabeceo = 0x00;
 datos.noimagen = 0x00;
 nocuadro = 0x00;                          /* No imagen leida por Com1 */
 latch1 = 0x00;                            /* Valores de los latch's de la interfaz */
 latch2 = 0x00;
 conmuta = 0x00;                           /* Utilizada para conmutar canal de audio */
 lee_dato = 0x00;                          /* No bytes leidos de cada imagen por Com1 */
 cual_gps = 0x01;                          /* Cual cadena se esta utilizando Rx de GPS */
 dato_gps = 0x00;                          /* Si existe un nuevo dato de gps */
 datos_gps = 0x00;                         /* Si se recibió el bloque completo de datos */
 posicioncaracter = 0;                     /* Apunta cadena que utiliza para rx GPS */
 cad_gps1[0] = 0x00;                       /* Mensajes recibidos del GPS */
 cad_gps2[0] = 0x00;
 mensaje_gps[0] = 0x00;
 caracter_gps[1] = 0x00;

                                           /* Despliega el menú principal */
 salvavideo(num_menu);                     /* Salva la porción de pantalla a utilizar */
 frame[num_menu].activo = 1;               /* Activa menú uno */
 dibujaborde(num_menu);
 despliegamenu(num_menu);
 return ; }                                /* Fin de ambiente( ) */
```

```
/* Restaura el estado que existf antes del programa */
void restaura( )
{
 outp(0x0021,0xB8);                          /* Restaura controlador de interrupciones */
 outp(DirLatch1,0x00);                       /* Inicializa lastch's en cero */
 outportb(DirLatch2,0x00);
 return ;
}                                            /* Fin de restaura( ) */


/* Despliega pantalla principal */
void pantalla( )
{
 char *pantalla[]= {                         /* Mensajes a Pantalla Principal */
        " Tiempo: "," Latitud:","Longitud:",
        " Altitud:","  Error:"," Alabeo:","Cabeceo:",
        "Camara1:","Camara2:","Camara3:",
        "Camara4:","Satelite" };

 cls();                                      /* Borra la pantalla */
 write_string(2,1,pantalla[0],VidNormal);    /* Mensajes a pantalla con (ADMV)*/
 write_string(3,1,pantalla[1],VidNormal);
 write_string(4,1,pantalla[2],VidNormal);
 write_string(5,1,pantalla[3],VidNormal);
 write_string(6,1,pantalla[4],VidNormal);
 write_string(8,1,pantalla[5],VidNormal);
 write_string(9,1,pantalla[6],VidNormal);
 write_string(11,1,pantalla[7],VidNormal);
 write_string(12,1,pantalla[8],VidNormal);
 write_string(13,1,pantalla[9],VidNormal);
 write_string(14,1,pantalla[10],VidNormal);
 write_string(1,40,pantalla[11],VidNormal);
 return ;
}                                            /* Fin de pantalla( ) */


/* Despliega una cadena con Acceso Directo a Memoria de Video */
void write_string(ren, col, p, atributo)
int ren, col;                                /* Renglón y columna de inicio */
char *p;                                     /* Apunta al inicio de la cadena */
int atributo;                                /* Color del caracter a desplegar */
{
 register int i;
 char far *v;

 v = DirMemVid;                              /* Dir memoria de video */
 v += (ren*160) + col*2;                     /* Calcula la dirección */
 for (i=col; *p; i++) {
        *v++ = *p++;                         /* Escribe el caracter */
        *v++ = atributo;                     /* Especifica atributo */
        }
 return ;
}                                            /* Fin write_string( ) */
```

```
/* Despliega un caracter con Acceso Directo a Memoria de Video */
void write_char(x, y, ch, atributo)
int x,y;                                    /* Posición en pantalla */
char ch;                                    /* Caracter a mostrar en pantalla */
int atributo;                               /* Color del caracter */
{
 register int i;
 char far *v;
 v = DirMemVid;                             /* Dir memoria de video */
 v += (x*160) + y*2;                        /* Calcula dirección de memoria*/
 *v++ = ch;                                 /* Escribe el caracter */
 *v = atributo;                             /* Especifica atributo */
 return ;
}                                           /* Fin write_char( ) */

/* Borra la pantalla */
void cls( )
{
 union REGS r;
 r.h.ah=6;
 r.h.al=0;
 r.h.ch=0;
 r.h.cl=0;
 r.h.dh=24;
 r.h.dl=79;
 r.h.bh=7;
 int86(0x10, &r, &r);
 return ;
}                                           /* Fin cls( ) */

/* Borra la ventana de Mensajes */
void clsmensajes( )
{
 char far *v;
 int i,j;
 v = DirMemVid;                             /* Dir de memoria de video */
 v += 19*160;                               /* Calcula desplazamiento */
 for(i=19; i<=24; i++)
        for(j=0; j<=79; j++) {
                *v++ = ' ';                 /* Borra caracter */
                 *v++ = ColorMensaje;       /* Especifica atributo */
                }
 return ;
}                                           /* Fin clsmensajes( ) */

/* Borra el cursor */
void borracursor( )
{
   union REGS r;
   r.x.cx=0x2000;
   r.h.ah=0x01;
   int86(0x10, &r, &r);
   return ;
}                                           /* Fin de borracursor( ) */
```

```
/* Crea los menú y asigna memoria para soportarlos */
make_menu(num,menu,llaves,contador,x,y,borde)
int num;
char *menu[], *llaves;
int contador, x, y, borde;
{
 register int i, len;
 int endx, endy, eleccion;
 unsigned char *p;
                                                /* Evita errores */
 if (num>Max_Frame) {                           /* Demasiados menús */
        printf("Demasiados Menues \n");
        exit(0);                                /* Suspende la ejecución del programa */
        }
 if ((x>24) || (x<0) || (y>79) || (y<0)) {      /* Error en las coordenadas */
        printf("Error, coordenadas del menu\n");
        exit(0);                                /* Suspende la ejecución del programa */
        }
 len=0;                                         /* Error en el tamaño del menú*/
 for (i=0; i<contador; i++)
        if (strlen(menu[i])>len) len = strlen(menu[i]);
 endy=len+1+y;
 endx=contador+1+x;
 if ((endx+1>24) || (endy+1>79)) {
        printf("El menu no cabe \n");
        exit(0);                                /* Suspende la ejecución del programa */
        }
                                                /* Asigna memoria suficiente para soportarlo */
 p = (unsigned char *) malloc(2*(endx-x+1)*(endy-y+1)+10);
 if (!p) exit(1);                               /* Aborta en caso de error */
 frame[num].startx=x; frame[num].endx=endx;     /* Construye el marco del menú */
 frame[num].starty=y; frame[num].endy=endy;
 frame[num].p=p;
 frame[num].menu=(char **) menu;
 frame[num].borde=borde;
 frame[num].llaves=llaves;
 frame[num].contador=contador;
 frame[num].activo=0;
 return(1);
}                                               /* Fin make_menu( ) */

/* Activa menú o reponde a tecla presionada */
void activamenu()
{
 switch(frame[0].activo) {
        case 1: respuesta( );                   /* Reponde a tecla presionada */
                break;
        case 0: num_menu = 0;                   /* Activa el menú */
                opcion = 0;
                pulldown(0);
                bioskey(0);                     /* Elimina tecla presionada */
        }
 return ;
}                                               /* Fin activamenu() */
```

```
/* Control de Menús, respuesta a la tecla presionada */
void respuesta()
{
  union llave {                                          /* Tecla presionada */
   char ch[2];
   int i;
  } c;
  int llave_elegida, temporal;                           /* Si es una letra especial */

  c.i = bioskey(0);                                      /* Lee la tecla */
  write_string(frame[num_menu].startx + 1 + opcion,      /* Video Normal la opción del menú */
  frame[num_menu].starty + 1, frame[num_menu].menu[opcion], VidInverso);
  if(c.ch[0]) {                                          /* Si tecla normal */

        llave_elegida = esta_en(frame[num_menu].llaves,
          tolower(c.ch[0]));                             /* Checa si es letra especial */
          temporal = opcion;
          opcion = llave_elegida - 1;
          if (llave_elegida && validacion()) {           /* Si es válida la nueva opción */
                write_string(frame[num_menu].startx + 1 + opcion,
                frame[num_menu].starty + 1, frame[num_menu].menu[opcion],
                 VidNormal);                             /* Video inverso la nueva opción */
                submenu();}                              /* Checa si existe submenú */
        else opcion = temporal;                          /* Restaura opción anterior */

        switch(c.ch[0]) {                                /* Checa si es Enter o Barra espaciadora */




            submenu();                                   /* Checa si existe submenú */
            break;
          case ' ' :                                     /* Checa validación de la opción */
            do ++opcion; while(!validacion()) ;
            break;
          case ESC :                                     /* Cierra el menú abierto */
            cierramenu( );
             if (!frame[num_menu].activo) return;
             break ; }
  }
  else {                                                 /* Checa si se presionó una flecha */
        switch(c.ch[1]) {
          case 72:
            do --opcion; while(!validacion());           /* Validación de opción */
            break;
          case 80:
            do ++opcion; while(!validacion());           /* Validación de opción */
            break; }
  }
  write_string(frame[num_menu].startx + 1 + opcion, frame[num_menu].starty + 1,
  frame[num_menu].menu[opcion], VidNormal);              /* Video inverso la nueva opción */
  return ;
}                                                        /* Fin respuesta( ) */
```

```
/*  Activa el menú correspondiente */
void pulldown(num)
int num;                                        /* Número de menú a activar */
{
 if(!frame[num].activo) {                       /* Activa el menú */
        salvavideo(num);                        /* Salva la parte del video */
        frame[num].activo = 1;
        dibujaborde(num);                       /* Dibuja el borde del menú */
        despliegamenu(num);                     /* Despliega el menú */
        }
 return ;
}                                               /* Fin pulldown( ) */


/* Dibuja el borde del menú */
void dibujaborde(num)
int num;                                        /* Número de menú */
{
 register int i;
 char far *v, far *t;

 v = DirMemVid;                                 /* Dir memoria de video */
 t = v;
 for(i=frame[num].startx+1; i<frame[num].endx; i++) {
        v += (i*160) + frame[num].starty*2;     /* Calcula dirección */
        *v++ = 179;                             /* Escribe caracter */
        *v = VidInverso;                        /* Especifica atributo */
         v = t;
         v += (i*160) + frame[num].endy*2;      /* Calcula dirección */
        *v++ = 179;                             /* Escribe caracter */
        *v = VidInverso;                        /* Especifica atributo */
         v = t;
        }

 for(i=frame[num].starty+1; i<frame[num].endy; i++) {
        v += (frame[num].startx*160) + i*2;     /* Dirección */
        *v++ = 196;                             /* Caracter */
        *v = VidInverso;                        /* Atributo */
        v = t;
        v += (frame[num].endx*160) + i*2;       /* Dirección */
        *v++ = 196;                             /* Caracter */
        *v = VidInverso;                        /* Atributo */
        v = t;
        }                                       /* Caracteres 191,192,217,218 */

 write_char(frame[num].startx, frame[num].starty,218, VidInverso);
 write_char(frame[num].startx, frame[num].endy,191, VidInverso);
 write_char(frame[num].endx, frame[num].starty,192, VidInverso);
 write_char(frame[num].endx, frame[num].endy,217, VidInverso);
 return ;
}                                               /* Fin dibujaborde( ) */
```

```
/* Dibuja el borde de la ventana de mensajes */
void bordemensajes( )
{
 register int i;
 char far *v, far *t;
 v = DirMemVid;                                  /* Dir memoria de video */
 t = v;
 for(i=18; i<=24; i++) {
        v += (i*160);                            /* Calcula dirección */
        *v++ = 179;                              /* Escribe caracter */
        *v = ColorMensaje;                       /* Especifica atributo */
        v = t;
        v += (i*160) + 158;                      /* Calcula dirección */
        *v++ = 179;                              /* Escribe caracter */
        *v = ColorMensaje;                       /* Especifica atributo */
        v = t;
        }

 for(i=0; i<=79; i++) {
        v += (2880) + i*2;                       /* Dirección */
        *v++ = 196;                              /* Caracter */
        *v = ColorMensaje;                       /* Atributo */
        v = t;
        v += (3840) + i*2;                       /* Dirección */
        *v++ = 196;                              /* Caracter */
        *v = ColorMensaje;                       /* Atributo */
        v = t;
        }                                        /* Caracteres 191,192,217,218 */

 write_char(18, 0, 218, ColorMensaje);
 write_char(18, 79, 191, ColorMensaje);
 write_char(24, 0, 192, ColorMensaje);
 write_char(24, 79, 217, ColorMensaje);
 write_string(18,35, " Mensajes ", ColorMensaje);
 return ;
}                                                /* Fin bordemensajes( ) */


/* Despliega el menú */
void despliegamenu(num)
int num;                                         /*Número de menú */
{
 register int i, x;
 char **m;

 x = frame[num].startx + 1;                      /* Renglón en pantalla */
 m = frame[num].menu;                            /* Opciones del menú */
 for (i=0; i<frame[num].contador; i++, x++)      /* Video Normal las opciones */
        write_string(x, frame[num].starty+1, m[i], VidInverso);

 write_string(frame[num].startx + 1 + opcion, frame[num].starty+1,
 m[opcion], VidNormal);                          /* Video inverso la opción */
 return ;
}                                                /* Fin despliegamenu( ) */
```

```
/* Checa si es una letra especial del menú */
int esta_en(s,c)
char *s, c;                                    /* s->letras especiales, c->letra presionada */
{
 register int i;
 for(i=0; *s; i++) if(*s++==c) return (i+1);
 return(0);
}                                              /* Fin esta_en( ) */


/* Salva la porción de video utilizado por el menú */
void salvavideo(num)
int num;                                       /* Número de menú */
{
 register int i,j;
 char *buf_ptr;
 char far *v, far*t;

 buf_ptr = frame[num].p;                       /* Dir donde se almacenará la porción de video */
 v = DirMemVid;                                /* Dir de memoria de video */
 for(i=frame[num].starty; i<frame[num].endy+1; i++)
        for(j=frame[num].startx; j<frame[num].endx+1; j++) {
                t = (v + (j*160) + i*2);       /* Dirección de la porción de video */
                *buf_ptr++ = *t++;             /* Salva el caracter de la porción de video */
                *buf_ptr++ = *t;               /* Salva su atributo */
                *(t-1) = ' ';                  /* Limpia la porción de video */
                *t = VidNormal;
                }
 return ;
}                                              /* Fin salvavideo( ) */


/* Restaura la porción de video utilizado por el menú */
void restauravideo(num)
int num;                                       /* Número de menú */
{
 register int i,j;
 char *buf_ptr;
 char far *v, far*t;

 buf_ptr = frame[num].p;                       /* Dir de respaldo de la porción de video */
 v = DirMemVid;                                /* Dir memoria de video */
 t = v;
 for(i=frame[num].starty; i<frame[num].endy+1; i++)
        for(j=frame[num].startx; j<frame[num].endx+1; j++) {
                v = t;
                v += (j*160) + i*2;            /* Calcula la dirección del renglón y columna */
                *v++ = *buf_ptr++;             /* Restaura caracter */
                *v = *buf_ptr++;               /* Restaura atributo */
                }
 frame[num].activo=0;                          /* Desactiva menú */
 return ;
}                                              /* Fin restauravideo( ) */
```

```
/* Cierra el menú correspondiente */
void cierramenu( )
{
 if(frame[num_menu].activo) {
        restauravideo(num_menu);                    /* Restaura la porción de video utilizada */
        switch(num_menu) {
                case 0 : break;
                case 1 : num_menu = 0;              /* Menú activo principal */
                         opcion = 0;                /* Opción Files del menú principal*/
                         break;
                case 2 : num_menu = 0;              /* Menú activo principal */
                         opcion = 1;                /* Opción Vtr */
                         break;
                case 3 : num_menu = 0;              /* Menú activo principal */
                         opcion = 2;                /* Opción Cam */
                         break;
                case 4 : num_menu = 3;              /* Menú activo Cam */
                         opcion = 0;                /* Opción Gps */
                         break; }
        }
 return ;
}                                                   /* Fin cierramenu( ) */

/* Manda llamar el submenú */
void submenu( )
{
  switch(num_menu) {
    case 0 : switch(opcion) {
                case 0 : num_menu = opcion + 1;
                         if(modo_operacion != Modo_Off) opcion = 0;
                                                else opcion = 3;
                         pulldown(num_menu);            /* Activa submenú Files */
                         break;
                case 1 : modo_operacion = Modo_Vtr;
                         num_menu = opcion + 1;
                         opcion = 0;
                         pulldown(num_menu);            /* Activa submenú Vtr */
                         break;
                case 2 : modo_operacion = Modo_Cam;
                         num_menu = opcion + 1;
                         opcion = 0;
                         pulldown(num_menu);            /* Activa submenú Cam */
                         break; }
             break;
    case 3 : switch(opcion) {
                case 0 : num_menu = 4;
                         opcion = 0;
                         pulldown(num_menu);            /* Activa submenú Gps */
                         break;



  return ;
 }                                                  /* Fin submenu( ) */
```

```
/* Validación de la nueva opción del menú activo */
int validacion()
{
 if(opcion = = frame[num_menu].contador) opcion = 0;          /* Filtro de tope de opción */
 if(opcion < 0) opcion = frame[num_menu].contador - 1;

 switch(num_menu) {
  case 0 : switch(modo_operacion) {
                case Modo_Off : break;                        /* Todas válidas */
                case Modo_Vtr : if(opcion = = 2) return(0);   /* Opción no válida */
                                break;
                case Modo_Cam : if(opcion = = 1) return(0);   /*Opción no válida */
                                break;
           }
           break;
  case 1 : switch(modo_operacion) {
                case Modo_Off : if(opcion < 3) return(0);     /*Opciones no válidas */
                                break;
                case Modo_Cam : if(opcion = = 2) return(0);   /* Opción no válida */
                                break;
           default : ; }
  default: return(1);                                         /* Opción válida */
 }
 return(1);                                                   /* Opción válida */
}                                                             /* Fin validacion( ) */

/* Ejecuta la función elegida */
void ejecuta( )
{
 switch(num_menu) {
  case 0: break;                                              /* No realiza tareas */
  case 1: switch(opcion) {
                case 0 : abrir( );                            /* Abre archivos */
                         break;
                case 1 : cerrar( );                           /* Cierra archivos */
                         break;
                case 2 : convierte( );                        /* Convierte el archivo a ASCII */
                         break;
                case 3 : salir( );                            /* Salir del programa */
                         break;
                default: ; }
          break;
  case 2: switch(opcion) {                                    /* Modo reproducir cintas */
                case 0 : regresacintas( );
                         break;
                case 1 : adelantacintas( );
                         break;
                case 2 : pausacintas( );
                         break;
                case 3 : videocintas( );
                         break;
                case 4 : detenercintas( );
                         break;
                case 5 : sincroniza( );
```

```
                    break;
                case 6 : al_numero_imagen( );
                    break;
                case 7 : digitaliza( );
                    break;
                case 8 : tiempodato( );
                    break;
                case 9 : modo_operacion = Modo_Off;
                    cierramenu( 2 );
                    break;
              default : ; }
        break;
  case 3: switch(opcion) {                          /* Modo grabar cintas*/
                case 1 : acerca_camara( );
                    break;
                case 2 : aleja_camara( );
                    break;
                case 3 : listascintas( );
                    break;
                case 4 : grabacintas( );
                    break;
                case 5 : enciendecamaras( );
                    break;
                case 6 : grabaenuna( );
                    break;
                case 7 : modo_operacion = Modo_Off;
                    cierramenu(3);
                    break;
              default : ; }
        break;
  case 4: switch(opcion) {                          /* Comunicación con el GPS */
                case 0 : inicializagps( );
                    break;
                case 1 : al_gps( );                 /* Tx mensajes al GPS */
                    break;
                case 2 : pidedatosgps( );           /* Datos utilizados */
                    break;
                case 4 : borragps( );               /* Borra la memoria del GPS */
                    break;
              default: break; }
              break;
   default : ; }
 nueva_funcion = 0;                                 /* Restaura nueva_funcion */
 return ;
}                                                   /* Fin ejecuta( ) */

/* Salida del programa */
void salir( )
{
 cerrar( );                                         /* Cierra el archivo */
 restaura( );                                       /* Restaura estado de PC */
 cls( );                                            /* Borra la pantalla */
 exit(0);                                           /* Aborta el programa */
}                                                   /* Fin salir( ) */
```

```
/* Opción de abrir archivos */
void abrir( )
{
 char *nombre;                                                    /* Nombre del archivo */
 union tecla {                                                    /* Tecla presionada */
        char ch[2];
        int i;
 } c;

 nombre = malloc(12);
 if(archivo[0] != 0x00) {                                         /* Filtros */
   mensajes("Ya existe un archivo abierto. ", ColorMensaje);
   return ;}
 mensajes("Nombre del archivo: ", ColorMensaje);
 nombre[0] = 0x00;
 leedato(nombre);                                                 /* Nombre del archivo */
 if(*nombre == 0x00) { mensajes("Ningun archivo abierto.", ColorMensaje);
                       return ; }

 switch(modo_operacion) {
  case Modo_Vtr : if(access(nombre, 0) == 0) {                    /* Checa existencia */
                        strcpy(archivo, nombre);                  /* archivo = nombre */
                        pfichero = fopen(archivo, "rb");          /* Abre archivo lectura */
                        mensajes("Archivo abierto...",ColorMensaje); }
                  else  mensajes("Error no existe...",ColorMensaje);
                  break;
  case Modo_Cam : if(access(nombre, 0) == 0) {                    /* Si existe el archivo */
                        mensajes("Sobrescribir, Escapar? ",ColorMensaje);
                        c.i = 0;                                  /* Inicializa tecla */
                        while(c.ch[0]! = 's' && c.ch[0]! = 'S' && c.ch[0]! = 'e' &&
                        c.ch[0]!='E' && c.ch[0]!=ESC) c.i = bioskey(0); /* Filtro para teclas */
                        if(c.ch[0]=='s' || c.ch[0]=='S') {
                                strcpy(archivo, nombre);          /* archivo = nombre */




                        write_char(mensajesx-2, 24, c.ch[0], ColorMensaje); }
                  else  {                                         /* No existe el archivo */
                        mensajes("Nuevo, Escapar? ",ColorMensaje);
                        c.i = 0;
                        while(c.ch[0]! = 'n' && c.ch[0]! = 'N' && c.ch[0]! = 'e' &&
                        c.ch[0]!='E' && c.ch[0]!=ESC) c.i = bioskey(0);
                        if(c.ch[0] == 'n' || c.ch[0] == 'N') {
                                strcpy(archivo, nombre);          /* archivo = nombre */
                                pfichero = fopen(archivo, "wb");  /*Abre archivo escritura*/
                                mensajes("Archivo creado.", ColorMensaje); }
                        else {  mensajes("Ningun archivo abierto.",ColorMensaje);
                                c.ch[0] = 'E'; }
                        write_char(mensajesx-2, 17, c.ch[0], ColorMensaje); }
                  break;
  default:; }
 free(nombre);                                                    /* Libera memoria */
 return ; }                                                       /* Fin de abrir( ) */
```

```
/* Lee el nombre del archivo del teclado */
void leedato(nombre)
char *nombre;                                          /* Nombre del archivo */
{union tecla {
        char ch[2];
        int i; } c;                                    /* Tecla presionada */
 int j;
 gotoxy(22,mensajesx-1);                               /* x,y en ventana de mensajes */
 c.ch[0] = 1;
 for(j=0; c.ch[0] != '\r'; j++) {                      /* Sale si presiona Return */
        c.i = bioskey(0);                              /* Lee tecla */
        switch(c.ch[0]) {                              /* Tipo de tecla */
                case '\r' : nombre[j] = 0x00;          /* Si enter fin de cadena */
                        break;
                case '\b' : if(j--){                   /* Si tecla <- */
                          write_char(mensajesx-1, 22+j, 0x20, ColorMensaje);
                          j--; }                       /* Actualiza puntero */
                        break;
                case ESC : c.ch[0] = '\r';             /* Return para salir */
                          nombre[0] = 0x00;            /* Cadena vacía */
                          break;
                default                                /* Cualquier tecla */
                        if(c.ch[0] >= '!' && c.ch[0] <= 'z' && j<12) {
                                nombre[j] = c.ch[0];   /* Almacena caracter */
                                nombre[j+1] = 0x00;    /* Fin de cadena */
                                write_char(mensajesx-1, 22+j, c.ch[0], ColorMensaje); }
                        else    j--};                  /*Tecla no válida */
 }
 if(j>12) nombre[12] = 0x00;                           /* Filtro para nombre */
 return ; }                                            /* Fin leedato( ) */

/* Convierte el archivo binario a ASCII */
void convierte( )
{ char *nombre, *ptr, punto = '.';                     /* Nombre del archivo */
  FILE *pascii;
 nombre = malloc(12);                                  /* Asigna memoria */
 if(archivo[0] == 0x00) {                              /* Filtro */
   mensajes("No existe un archivo abierto. ", ColorMensaje);
   return ; }
 strcpy(nombre, archivo);                              /* nombre = archivo */
 ptr = strchr(nombre, punto);                          /* Busca el punto */
 if(ptr) strcpy(ptr, ".TXT");                          /* Cambia extensión */
  else  strcpy(nombre+(strlen(nombre)), ".TXT");
 pascii = fopen(nombre, "wt");                         /* Abre archivo */
 fseek(pfichero, 0,SEEK_SET);                          /* Puntero al inicio del archivo binario*/
 fprintf(pascii," Latitud  Longitud Altitud Tiempo  Imagen Error Alabeo Cabeceo \n");
 while(!feof(pfichero))                                /* Lee del binario y escribe en el ASCII */
        { fread(&datos, bytesdatos, 1, pfichero);
        fprintf(pascii, "%8.2f %9.2f  %6ld %6ld %7ld   %3d    %3d     %3d \n",
        datos.latitud,datos.longitud,datos.altitud, datos.tiempo, datos.noimagen,
        datos.error, datos.alabeo, datos.cabeceo); }
 fclose(pascii);                                       /* Cierra archivo ASCII */
 return ; }                                            /* Fin de convierte( ) */
```

```
/* Posiciona el puntero al final del archivo */
void posicionapuntero()
{
 long apunta;
 if(filelength(fileno(pfichero)) != 0) {                    /* Longitud del archivo */
        apunta = (filelength(fileno(pfichero))/bytesdatos)-1; /* Desplazamiento del archivo */
        fseek(pfichero, apunta, SEEK_SET);                  /* Apunta al último del archivo */
        fread(&datos, bytesdatos, 1, pfichero);             /* Apunta al final del archivo */
        nocuadro = datos.noimagen;                          /* Actualiza número de imagen */
        mensajes("Agregara datos al archivo.", ColorMensaje); }
else    mensajes("Archivo vacio.", ColorMensaje);
 return ;
}                                                           /* Fin posicionapuntero( ) */


/* Maneja la ventana de mensajes */
void mensajes(mensaje, atributo)
char *mensaje;                                              /* Mensaje a la ventana */
int atributo;                                               /* Color */
{
 char far *v, far *t, far *z;
 int i,j;

 if(mensajesx == 24) {                                      /* Recorre los renglones de la ventana */
        z = DirMemVid;                                      /* Dir memoria de video */
        for(i=20; i<=23; i++) {
                t = z + ((i-1)*160) + 2;                    /* Dir renglón anterior */
                v = z + i*160 + 2;                          /* Dir renglón */
                for(j=1; j<=78; j++) {
                        *t++ = *v;                          /* Renglón anterior = renglón */
                        *t++ = *(v+1);                      /* Atributo */
                        *v++ = ' ';                         /* Borra renglón */
                        *v++ = ColorMensaje;                /* Atributo */
                }
        }
        mensajesx=23;                                       /* Último renglón */
}
 write_string(mensajesx++, mensajesy, mensaje, atributo);   /* Mensaje */
 return ;
}                                                           /* Fin mensajes( ) */


/* Opción de Cerrar archivos */
void cerrar( )
{
 if(archivo[0] != 0x00) {                                   /* Si existe archivo abierto */
        fclose(pfichero);                                   /* Cierra el archivo */
        archivo[0] = 0x00;                                  /* Actualiza variable */
        mensajes("Archivo Cerrado. ", ColorMensaje); }
 else   mensajes("Error no existe archivo abierto. ", ColorMensaje);
 return ;
}                                                           /* Fin cerrar( ) */
```

```
/* Modo Reproducción de cintas, condiciones iniciales y tarea general */
void vtr()
{
 void interrupt leecom1( );                    /* Para leer número de imagen */
 void interrupt pulsovertical();               /* Para sincronizar */
 void interrupt (*oldint_0C)();                /* Apuntador a rutina anterior*/
 void interrupt (*oldint_0F)();
 long cuadro;                                  /* # de Imagen leido por Com1*/

 com1img( );                                   /* Inicializa Com1 para Rx No. Imagen */
 com2dig();                                    /* Com2 para Rx del digitador */



 cuadro = 0x00;                                /* Inicializa variables */
 lee_dato = 0x00;
 latch1 = 0x00;
 latch2 = 0x00;
 conmuta = 0x00;
 outportb(DirLatch1, latch1);                  /* Inicializa latch's de la interfaz */
 outportb(DirLatch2, latch2);
 outportb(0x21, 0x28);                         /* Habilita interrupciones IRQ4 e IRQ7 */
 mensajes("Camara en modo Vtr.", ColorMensaje);
 detenercintas( );                             /* Cámaras a stop */

 while(modo_operacion == Modo_Vtr) {           /* Ciclo de espera para salida */
        if(camara1 != cuadro) {                /* Si nueva imagen */
                cuadro = camara1;
                obtienedatos( ); }             /* Obtiene datos de disco y los despliega */
        if(kbhit( )) {                         /* Atiende al usuario */
                activamenu( );                 /* Activa el menú */
                if(nueva_funcion) ejecuta( ); } /* Ejecuta la función */
 }

 latch1 = 0x00;                                /* Limpia latch's de la interfaz */
 outp(DirLatch1, latch1);
 latch2 = 0x00;
 outp(DirLatch2, latch2);
 delay(1);
 outportb(0x21, 0xB8);                         /* Restaura controlador 8259 */
 setvect(0x0C, oldint_0C);                     /* Restaura vector de interrupciones */
 setvect(0x0F, oldint_0F);
 cerrar( );                                    /* Cierra archivo */
 com1reset( );                                 /* Restaura puertos serie */
 com2reset( );
 mensajes("Camara en modo Off. ", ColorMensaje);
 return ;
}                                              /* Fin vrt( ) */
```

```
/* Inicializa Com1 para Rx de No. Imagen */
void com1img( )
{delay(1);
 outportb(0x3fb, 0x80);                          /* Para accesar registro de baud rate */
 delay(1);
 outportb(0x3f8, 0x30);
 delay(1);
 outportb(0x3f9, 0x00);                          /* 2400 bauds */
 delay(1);
 outportb(0x3fb, 0x1f);                          /* Paridad par, 2 stop bit, 8 data bit */
 delay(1);
 outportb(0x3fc, 0x08);                          /* No modem, si interrupciones */
 delay(1);
 outportb(0x3f9, 0x01);                          /* Interrupción por Rx habilitada */
 delay(1);
 mensajes("Com1:2400 bauds,paridad par,2 stop bit,8 data bit,interr por Rx.", ColorMensaje);
 return ; }                                      /* Fin com1img( ) */

/* Inicializa Com1 para Tx de No. Imagen */
void inicializacom1tx()
{delay(1);
 outportb(0x3fb, 0x80);                          /* Para accesar registro de baud rate */
 delay(1);
 outportb(0x3f8, 0x30);
 delay(1);
 outportb(0x3f9, 0x00);                          /* 2400 bauds */
 delay(1);
 outportb(0x3fb, 0x1f);                          /* Paridad par, 2 stop bit, 8 data bit */
 delay(1);
 outportb(0x3fc, 0x08);                          /* No modem, si interrupciones */
 delay(1);
 outportb(0x3f9, 0x02);                          /* Interrupción por Tx habilitada */
 delay(1);
 mensajes("Com1:2400 bauds,paridad par,2 stop bit,8 data bit,interr por Tx.", ColorMensaje);
 return ; }                                      /* Fin inicializacom1tx( ) */

/* Inicializa Com2 para comunicación con el Gps */
void inicializacom2()
{delay(1);
 outportb(0x2fb, 0x80);                          /* Para accesar registro de baud rate */
 delay(1);
 outportb(0x2f8, 0x0C);
 delay(1);
 outportb(0x2f9, 0x00);                          /* 9600 bauds */
 delay(1);
 outportb(0x2fb, 0x03);                          /* No paridad, 1 stop bit, 8 data bit */
 delay(1);
 outportb(0x2fc, 0x08);                          /* No modem , si interrupciones */
 delay(1);
 outportb(0x2f9, 0x01);                          /* Interrupción por Rx habilitada */
 delay(1);
 mensajes("Com2:9600 bauds,no paridad,1 stop bit,8 data bit,interr por Rx.", ColorMensaje);
 return ; }                                      /* Fin inicializacom2( ) */
```

```
/* Inicializa Com2 para Rx comando de digitación */
void com2dig( )
{delay(1);
 outportb(0x2fb, 0x80);                          /* Para accesar registro de baud rate */
 delay(1);
 outportb(0x2f8, 0x0C);
 delay(1);
 outportb(0x2f9, 0x00);                          /* 9600 bauds */
 delay(1);
 outportb(0x2fb, 0x03);                          /* No paridad, 1 stop bit, 8 data bit */
 delay(1);
 outportb(0x2fc, 0x00);                          /* No modem, si interrupciones */
delay(1);
 outportb(0x2f9, 0x00);                          /* Interrupción por Rx habilitada */
 delay(1);
 mensajes("Com2:9600 bauds,no paridad,1 stop bit,8 data bit.", ColorMensaje);
 return ;}                                       /* Fin inicializacom2( ) */

/* Reestablece Com1 */
void com1reset( )
{delay(1);
 outportb(0x3fb, 0x80);                          /* Para accesar registro de baud rate */
 delay(1);
 outportb(0x3f8, 0x0C);
 delay(1);
 outportb(0x3f9, 0x00);                          /* 9600 bauds */
 delay(1);
 outportb(0x3fb, 0x03);                          /* No paridad, 1 stop bit, 8 data bit */
 delay(1);
 outportb(0x3fc, 0x00);                          /* No modem, no interrupciones */
 delay(1);
 outportb(0x3f9, 0x00);                          /* No interrupción */
 delay(1);
 mensajes("Com1:9600 bauds,no paridad,1 stop bit,8 data bit.", ColorMensaje);
 return ; }                                      /* Fin com1reset( ) */

/* Reestablece Com2 */
void com2reset( )
{delay(1);
 outportb(0x2fb, 0x80);                          /* Para accesar registro de baud rate */
 delay(1);
 outportb(0x2f8, 0x0C);
 delay(1);
 outportb(0x2f9, 0x00);                          /* 9600 bauds */
 delay(1);
 outportb(0x2fb, 0x03);                          /* No paridad, 1 stop bit, 8 data bit */
 delay(1);
 outportb(0x2fc, 0x00);                          /* No modem, no interrupciones */
 delay(1);
 outportb(0x2f9, 0x00);                          /* No interrupción */
 delay(1);
 mensajes("Com2:9600 bauds,no paridad,1 stop bit,8 data bit.", ColorMensaje);
 return ; }                                      /* Fin com2reset() */
```

```
/* Obtiene datos de disco y despliega en pantalla */
void obtienedatos( )
{
 if((archivo[0] != 0x00) && (filelength(fileno(pfichero)) != 0)) {    /* Filtro */
        buscadatos(camara1);                                          /* Acceso aleatorio */
        fread(&datos, bytesdatos, 1, pfichero); }                     /* Lee datos de archivo */
        datosapantalla( );                                            /* Despliega datos */
 return ;
}                                                                     /* Fin obtienedatos( )*/

/* Posiciona el puntero del archivo en el número de cámara correspondiente */
void buscadatos(busco)
long busco;
{
 long min=0, max=0;
 ldiv_t m;

 min = 0;
 max = (filelength(fileno(pfichero))/bytesdatos) -1;                  /*# de bloques de información */
 while(min < max) {
        m = ldiv((min+max), 2);
        fseek(pfichero, (m.quot-1)*bytesdatos,SEEK_SET);              /* Mueve el puntero del archivo */
        fread(&datos, bytesdatos, 1, pfichero);                       /* Lee el dato */
        if(datos.noimagen < busco) min = m.quot+1;                    /* Compara con el buscado */
                                else max = m.quot;
        }
 m = ldiv((min+max), 2);
 fseek(pfichero, (m.quot-1)*bytesdatos,SEEK_SET);
 if(m.quot*bytesdatos < = (filelength(fileno(pfichero))/bytesdatos)) {
        fread(&datos, bytesdatos, 1, pfichero);
        if(datos.noimagen > = busco) fseek(pfichero, (m.quot-1)*bytesdatos,SEEK_SET);
 }
 return ;
}                                                                     /* Fin buscadatos( ) */

/* Posiciona el puntero del archivo en la hora correspondiente */
void buscadatogps(busco)
long busco;
{long min=0, max=0;
 ldiv_t m;
 min = 0;
 max = (filelength(fileno(pfichero))/bytesdatos) -1;                  /* Longitud del archivo */
 while(min < max) {
        m = ldiv((min+max), 2);
        fseek(pfichero, (m.quot-1)*bytesdatos,SEEK_SET);              /* Posiciona el puntero */
        fread(&datos, bytesdatos, 1, pfichero);                       /* Lee el dato */
        if(datos.tiempo < busco) min = m.quot+1;                      /* Compara con el buscado */
                        else max = m.quot; }
 m = ldiv((min+max), 2);
 fseek(pfichero, (m.quot-1)*bytesdatos,SEEK_SET);
 if(m.quot*bytesdatos < = (filelength(fileno(pfichero))/bytesdatos)) {
        fread(&datos, bytesdatos, 1, pfichero);
        if(datos.tiempo > = busco) fseek(pfichero, (m.quot-1)*bytesdatos,SEEK_SET);}
return ; }                                                            /* Fin buscadatogps( ) */
```

```
/* Despliega a pantalla los datos */
void datosapantalla( )
{int i;
 gotoxy(11,3);                                          /* Datos a pantalla */
 printf("%6ld",datos.tiempo);
 gotoxy(12,4);
 printf("%8.2f",datos.latitud);
 gotoxy(11,5);
 printf("%9.2f",datos.longitud);
 gotoxy(11,6);
 printf("%6ld",datos.altitud);
 gotoxy(14,7);
 printf("%3d",datos.error);
 gotoxy(14,9);
 printf("%6.2f",((0.46875*datos.alabeo)-60));
 gotoxy(14,10);
 printf("%6.2f",((0.46875*datos.cabeceo)-60));
 gotoxy(11,12);
 printf("%7ld",camara1);
 gotoxy(11,13);
 printf("%7ld",camara2);
 gotoxy(11,14);
 printf("%7ld",camara3);
 gotoxy(11,15);
 printf("%7ld",camara4);
 gotoxy(19,13);                                         /* Diferencia con respecto a cámara1 */
 printf("%8ld",(camara2-camara1));
 gotoxy(19,14);
 printf("%8ld",(camara3-camara1));
 gotoxy(19,15);
 printf("%8ld",(camara4-camara1));
 return ; }                                             /* Fin datosapantalla( ) */
/* Manda la funcion de cada cámara a pantalla */
void pantallafuncion( )
{int i;
  for(i=0; i<NumeroCamaras; i++)
        switch(funcion[i]) {case Cero: write_string(11+i, 27, "     ", VidNormal);
                                break;
                           case Rec: write_string(11+i, 27, "REC  ", VidNormal);
                                break;
                           case Stby: write_string(11+i, 27, "STBY ", VidNormal);
                                break;
                           case Fast: write_string(11+i, 27, "FF   ", VidNormal);
                           break;
                           case Stop: write_string(11+i, 27, "STOP ", VidNormal);
                                break;
                           case Rew: write_string(11+i, 27, "REW  ", VidNormal);
                                break;
                           case Play: write_string(11+i, 27, "PLAY ", VidNormal);
                                break;
                           case Pausa: write_string(11+i, 27, "PAUSA", VidNormal);
                                 break;
                           default: ; }
 return ;}                                                      /* Fin pantallafuncion( ) */
```

```
/* Servicio de interrupción para recepción del No. imagen por Com1 */
void interrupt leecom1()
{
disable( );                                                    /* Deshabilita interrupciones */
inicio = inportb(0x3fd);
if((inicio & 0x1E) != 0x00) {                                  /*Checa error de Rx*/
        inicio = inp(0x3f8);                                   /* Elimina dato erroneo */
        lee_dato = 0x00; }                                     /* Actualiza bandera */
else {
        switch(lee_dato) {
                case 0:  inicio = inportb(0x03f8);             /* Lee palabra de arranque */
                         if(inicio == 0x80) lee_dato = 0x01;
                         break;
                case 1:  byte1 = inportb(0x03f8);              /* Lee byte 1 del No. img */
                         if(byte1 < 0x80)  lee_dato = 0x02;
                                 else      lee_dato = 0x00;    /* En caso de error */
                         break;
                case 2:  byte2 = inportb(0x03f8);              /* Lee byte 2 del No img */
                         if(byte2 < 0x80)  lee_dato = 0x03;
                                 else      lee_dato = 0x00;    /* En caso de error */
                         break;
                case 3:  byte3 = inportb(0x03f8);              /* Lee byte 3 del No img */
                         if(byte3 < 0x80){
                                         nocuadro = byte3;     /* Actualiza No de imagen */
                                         nocuadro = nocuadro*0x4000;
                                         nocuadro = nocuadro+byte1+(byte2*0x80);
                                         switch(latch2 & 0x03) {
                                                                 case 0:  camara1 = nocuadro;
                                                                          break;
                                                                 case 1:  camara2 = nocuadro;
                                                                          break;
                                                                 case 2:  camara3 = nocuadro;
                                                                          break;
                                                                 case 3: camara4 = nocuadro;
                                                                          break;
                                                                 default:; }
                                         if(conmuta != 0x00) {          /* Reset conmuta */




                                                         conmuta = 0x00; }
                                         }
                         lee_dato = 0x00;                      /* Reset lee_dato */
                         break;
                default:lee_dato = 0x00; }
        }
outportb(0x20, 0x64);                                          /* Borra bit 4 , IRQ4 del 8259 */
enable( );                                                     /* Habilita interrupciones */
}                                                              /* Fin leecom1( ) */
```

```
/* Servicio de interrupción para sincronía vertical, conmuta canal de audio en caso de no recibir
   No. imagen después de 5 pulsos de sincronía vertical de la cámara uno */
void interrupt pulsovertical( )
{
 disable( );                                                    /* No interrupciones hardware */
 outportb(0x20, 0x67);                                          /* Borra IRQ7 del 8259 */
 if(++conmuta>=5) {                                             /* 5 cuadros, 2½ imagenes */
        if((latch2 & 0x03) >= (NumeroCamaras-1))
                latch2 = latch2 & 0xFC;                         /* Canal cero, cámara1 */
        else    latch2++;                                       /* Siguiente canal */
        outportb(DirLatch2, latch2);                            /* Cambio de canal */
        conmuta = 0x00;                                         /* Reset conmuta */
        lee_dato = 0x00;                                        /* Reset No de bytes leidos */
        inicio = 0x00;
 }
 enable( );                                                     /* Habilita interrupciones */
}                                                               /* Fin pulsovertical( ) */


/* Envía función a la cámara */
void enviafuncion(funcion_, canal_, mseg)
unsigned char funcion_, canal_;                                 /* Función y canal del control remoto */
unsigned int mseg;                                              /* Duración del interruptor cerrado */
{
 outp(DirLatch2, canal_);                                       /* Canal del control remoto */
 delay(250);                                                    /* Tiempo para sincronización */
 outp(DirLatch1, funcion_);                                     /* Envía función */
 delay(mseg);                                                   /* Tiempo de espera para respuesta */
 outp(DirLatch1, Cero);                                         /* Borra función */
 delay(50);
 canal_ = ((canal_ /0x04) & 0x03);                              /* Número de canal */
 if(funcion_ == Pausa)                                          /* Filtros */
        if(funcion[canal_] == Pausa) funcion_ = Play;
 if(funcion_ == Rec)
        if(funcion[canal_] == Rec) funcion_ = Stby;
 funcion[canal_] = funcion_;                                    /* Actualiza función */
 pantallafuncion( );                                            /* Actualiza la función en pantalla */
 return ;
}                                                               /* Fin enviafuncion( ) */


/* Envía función a todas las cámaras */
void a_todas(funcio_n)
unsigned char funcio_n;
{
 int i;
 latch2 = latch2 & 0x03;                                        /* Control remoto = canal cero */
 for(i=1; i<=NumeroCamaras; i++) {
        enviafuncion(funcio_n, latch2, 400);                    /* Envía la función */
        latch2 = latch2 + 0x04; }                               /* Siguiente canal */
 latch2 = latch2 & 0x03;                                        /* Control remoto = canal cero */
 return ;
}                                                               /* Fin a_todas( ) */
```

```
/* Función regresar cintas, modo Vtr */
void regresacintas( )
{
 unsigned char sistop;                                  /* Bandera para stop */
 int i;

 sistop = 0x00;
 latch2 = latch2 & 0x03;                                /* Canal cero del control remoto */
 for(i=0; i<NumeroCamaras; i++) {                       /* Detiene las cintas */
        if((funcion[i]!=Stop) && (funcion[i]!=Rew)) {
                enviafuncion(Stop, latch2, 400);
                sistop = 0x01; }
        latch2 = latch2 + 0x04; }                       /* Siguiente canal del control remoto */
 if(sistop) {    mensajes("Deteniendo las cintas...", ColorMensaje);
                sleep(4); }                             /* Tiempo de respuesta */

 latch2 = latch2 & 0x03;                                /* Canal cero del remoto */
 for(i=0; i<NumeroCamaras; i++) {                       /* Envía función */
        if(funcion[i] != Rew) enviafuncion(Rew, latch2, 400);
        latch2 = latch2 + 0x04;
}
 mensajes("Cintas regresandose...", ColorMensaje);
 latch2 = latch2 & 0x03;
 return ;
}                                                       /* Fin regresacintas( ) */


/* Función adelantar cintas, modo Vtr */
void adelantacintas( )
{
 unsigned char sistop;                                  /* Bandera para stop */
 int i;

 sistop = 0x00;
 latch2 = latch2 & 0x03;
 for(i=0; i<NumeroCamaras; i++) {                       /* Detiene las cintas */
        if((funcion[i]!=Stop) && (funcion[i]!=Fast)) {
                enviafuncion(Stop, latch2, 400);
                sistop = 0x01; }
        latch2 = latch2 + 0x04;}                        /* Siguiente canal del control remoto */
 if(sistop) {    mensajes("Deteniendo las cintas...", ColorMensaje);
                sleep(4); }                             /* Tiempo de respuesta */

 latch2 = latch2 & 0x03;                                /* Canal cero */
 for(i=0; i<NumeroCamaras; i++) {                       /* Envía comando */
        if(funcion[i] != Fast) enviafuncion(Fast, latch2, 400);
        latch2 = latch2 + 0x04;
}
 mensajes("Cintas adelantandose...", ColorMensaje);
 latch2 = latch2 & 0x03;
 return ;
}                                                       /* Fin adelantacintas( ) */
```

```
/* Función pausa, modo Vtr */
void pausacintas( )
{
 int i;
 latch2 = latch2 & 0x03;                                   /* Canal cero del remoto */
 for(i=0; i<NumeroCamaras; i++) {                          /* Envía comando */
       if((funcion[i] = = Pausa) || (funcion[i] = = Play)) {      /*Solo en caso de estas funciones*/
             if(funcion[i] = = Pausa) mensajes("Camara en avance.", ColorMensaje);
                                 else     mensajes("Camara en pausa.", ColorMensaje);
             enviafuncion(Pausa, latch2, 400); }
       else      mensajes("Error: Funcion inadecuada. ", ColorMensaje);
       latch2 = latch2 + 0x04; }
 latch2 = latch2 & 0x03;
 sleep(1);                                                 /* Tiempo de respuesta */
 return ; }                                                /* Fin pausacintas( )*/


/* Función reproducir cintas, modo Vtr */
void videocintas( )
{
 unsigned char sistop;                                     /* Bandera para stop */
 int i;
 sistop = 0x00;
 latch2 = latch2 & 0x03;
 for(i=0; i<NumeroCamaras; i++) {                          /* Detiene las cintas */
       if((funcion[i]!=Stop) && (funcion[i]!=Play)) {      enviafuncion(Stop, latch2, 400);
                                                           sistop = 0x01; }
       latch2 = latch2 + 0x04; }                           /* Siguiente canal del control remoto */
 if(sistop) {      mensajes("Deteniendo las cintas...", ColorMensaje);
             sleep(4); }                                   /* Tiempo de respuesta */
 latch2 = latch2 & 0x03;                                   /* Canal cero del remoto */
 for(i=0; i<NumeroCamaras; i++) {    if(funcion[i] != Play) enviafuncion(Play, latch2, 400);
                                     latch2 = latch2 + 0x04; }
 mensajes("Reproduciendo cintas...", ColorMensaje);
 latch2 = latch2 & 0x03;
 return ; }                                                /* Fin videocintas( )*/


/* Función detener cintas, modo Vtr */
void detenercintas()
{unsigned char sistop = 0x00;
 int i;
 latch2 = latch2 & 0x03;                                   /* Canal cero */
 for(i=0; i<NumeroCamaras; i++) {                          /* Detiene las cintas */
       if(funcion[i] != Stop) {  enviafuncion(Stop, latch2, 400);
                                 sistop = 0x01; }
       latch2 = latch2 + 0x04; }                           /* Siguiente canal del control remoto */
 if(sistop){       mensajes("Deteniendo las cintas, espera un momento...", ColorMensaje);
             sleep(4);                                     /* Tiempo de respuesta */
             mensajes("Camaras paradas, puedes continuar.", ColorMensaje); }
 else        mensajes("Error: Las cintas ya estan paradas. ", ColorMensaje);
 latch2 = latch2 & 0x03;
 return ; }                                                /* Fin detenercintas( ) */
```

```
/* Calcula el tiempo que tarda en REW/FF para acercarse a la imagen deseada */
unsigned retardo(miles)
long miles;                                          /* Retardo en milisegundos */
{ldiv_t i;
 long j;
 union llave {    char ch[2];
                  int i; } c;
 if(miles>=0)  enviafuncion(Rew, latch2, 400);       /* Manda función Rew o Fast */
       else    enviafuncion(Fast, latch2, 400);
 delay(1100);                                        /* Tiempo para pasar de stop a rew/fast */
 miles = fabs(miles);                                /* Valor absoluto de miles */
 if(miles>2000){          i = ldiv(miles,2000);      /* Duración de fast o Rew */
                  for(j=1; j<=i.quot; j++) {
                          delay(2000);
                          if(kbhit( )) {     c.i = bioskey(0);/* checa si ESC */
                                             if(c.ch[0] = = ESC) {   /* Aborta procedimiento */
                                                    mensajes("Acercamiento a imagen abortado ",
                                                    ColorMensaje);
                                                     return(0); } } }
                  miles = miles-(i.quot*2000); }
delay(miles);
return(1);
}                                                    /* Fin retardo( ) */

/* Aproxima a la imagen deseada */
unsigned NoImagen(imagen1, imagen2, img_seg)
long imagen1, imagen2;                               /* imagen1 = aproximar, imagen2 = deseada */
float img_seg;                                       /* Imágenes por segundo con Rew o Fast */
{
long cuantum;                                        /* Tiempo de fast o rew para aproximar imágenes */
union llave {
        char ch[2];
        int i; } c;

if(labs(imagen1-imagen2)>600) {
        cuantum = ceil((imagen1-imagen2)/img_seg);   /* Aproxima las imágenes */
        if(!retardo(cuantum)) {   mensajes("Acercamiento a imagen abortado ", ColorMensaje);
                                  return(0); }       /* Aborta procedimiento */
        enviafuncion(Play, latch2, 400);
        while(imagen1 = = nocuadro) {                /* Espera No imagen */
                datosapantalla();
                if(kbhit( )) {    c.i = bioskey(0);  /* Checa si ESC */
                                  if(c.ch[0] = = ESC) {   /* Aborta procedimiento */
                                        mensajes("Acercamiento a imagen abortado",ColorMensaje);
                                        return(0); } } }
        enviafuncion(Stop, latch2, 400);
        datosapantalla();
        sleep(4);                                    /* Tiempo mínimo después de un stop */
        NoImagen(nocuadro, imagen2, img_seg); }      /*Recursividad para mayor aproximación */
else {   if((imagen2-nocuadro) < 240) {              /*Menos de 240 Img, regresa 600 Img */
                cuantum = 1200;                      /* Tiempo mínimo para rew o fast */
                if(!retardo(cuantum)) {   mensajes("Acercamiento a imagen abortado",ColorMensaje);
                                          return(0); } } }   /* Aborta procedimiento */
 return(1); }                                        /* Fin NoImagen( )*/
```

```
/* Sincroniza todas las cámaras respecto a la cámara 1 */
void sincroniza( )
{union llave {    char ch[2];
                  int i; } c;
long temporal;
mensajes("Comienza Sincronizacion", ColorMensaje);
videocintas( );                                          /* Obtiene # imagen de cada cámara */
delay(500);                                              /*Tiempo suficiente para leer de la 4 cam*/
pausacintas( );                                          /* Cintas en pausa */
outportb(0x21, 0xA8);                                    /* No IRQ7, para no conmutar canal */
switch(1){case 1:                                        /* Manda a sincronizar una por una */
                if(NumeroCamaras>=2){    nocuadro = camara2; /* Sincroniza cámara 2 */
                                         if(!sinc_cam1_con(0x05, camara1)) break ; }
                if(NumeroCamaras>=3){    nocuadro = camara3; /* Sincroniza cámara 3 */
                                         if(!sinc_cam1_con(0x0A, camara1)) break ; }
                if(NumeroCamaras>=4) {   nocuadro = camara4; /* Sincroniza cámara 4 */
                                         if(!sinc_cam1_con(0x0F, camara1)) break ; }
          break;
          default:; }
latch1 = 0x00;                                           /* Reestablece condiciones */
outportb(DirLatch1, latch1);
latch2 = 0x00;
outportb(DirLatch2, latch2);
outportb(0x21, 0x28);
mensajes("Sincronizacion finalizada. ", ColorMensaje);
return ; }                                               /* Fin sincroniza( ) */


/* Sincroniza las cámaras respecto a la cámara 1 */
unsigned sinc_cam1_con(cual, numeroimagen)
unsigned cual;                                           /* Canal a utilizar */
long numeroimagen;                                       /* # al que tiene que llegar */
{union llave {    char ch[2];
                  int i; } c;
 long temporal;
 lee_dato = 0x00;                                        /* Cero datos leidos por com1 */
 latch2 = cual;                                          /* Canal de la cámara a sincronizar */
 conmuta = 0x00;                                         /* No habilita conmutación de canal */
 outportb(DirLatch2, latch2);
 delay(100);                                             /* Para sincronización canal 0 y cámara1 */
 enviafuncion(Stop, latch2,400);                         /* Detiene cámaras */
 sleep(4);                                               /* Tiempo mínimo de respuesta */
 if(!NoImagen(nocuadro, numeroimagen, ImgSegTR6)){       /* Aproxima la cámara 1 al # imagen */
       mensajes("Sincronizacion abortada ", ColorMensaje);
        return(0); }                                     /* Aborta procedimiento por ESC */
 enviafuncion(Play, latch2, 400);                        /* Avanza cámara */
 temporal = nocuadro;
 while(temporal = = nocuadro){ datosapantalla( );        /* Lee un número de imagen*/
                               if(kbhit()) {             /* Aborta si ESC */
                                     c.i = bioskey(0);
                                     if(c.ch[0] = = ESC) {
                                           mensajes("Sincronizacion abortada",ColorMensaje);
                                           return(0); } } }
```

```
while(nocuadro < = numeroimagen) {                    /* Avanza hasta alcanzar el No. Imagen */
        datosapantalla( );
        if(kbhit()) {       c.i = bioskey(0);
                            if(c.ch[0] = = ESC) {     mensajes("Sincronizacion abortada",ColorMensaje);
                                                      return(0); } } }
enviafuncion(Pausa, latch2, 400);                     /* Cámara en pausa */
datosapantalla( );
return(1);
}                                                     /* Fin sinc_cam1_con( ) */

/* Se posicionan todas las cámaras en un número de imagen especificado */
void al_numero_imagen()
{long numero;                                         /* Número de imagen */
 char *nombre = 0x00;
 nombre = malloc(8);                                  /* Obtiene el número de imagen */
 *nombre = 0x00;
 mensajes("Numero de Imagen ? : ", ColorMensaje);
 leedato(nombre);
 if(*nombre = = 0x00) { mensajes("Cancelada la operacion.", ColorMensaje);
                        return; }                     /* Aborta si vacío */
 numero = atol(nombre);                               /* Convierte cadena a numérico */
 *nombre = 0x00;
 if(!numero) {   mensajes("Error: Dato no num rico, operacion cancelada.", ColorMensaje);
                 return; }                            /* Aborta si dato no numerico */

 mensajes("Comienza Operacion de posicionamiento.", ColorMensaje);




 switch(1) {     case 1:  if(NumeroCamaras > = 1) {    /* Manda cámara 1 al número deseado */
                              nocuadro = camara1;
                              if(!sinc_cam1_con(0x00, numero)) break ; }
                          if(NumeroCamaras > = 2) {    /* Sincroniza cámara 2 */
                              nocuadro = camara2;
                              if(!sinc_cam1_con(0x05, camara1)) break ; }
                          if(NumeroCamaras > = 3) {    /* Sincroniza cámara 3 */
                              nocuadro = camara3;
                              if(!sinc_cam1_con(0x0A, camara1)) break ; }
                          if(NumeroCamaras > = 4) {    /* Sincroniza cámara 4 */
                              nocuadro = camara4;
                              if(!sinc_cam1_con(0x0F, camara1)) break ; }
                          break;
                 default:; }

 latch1 = 0x00;                                       /* Reestablece condiciones */
 outportb(DirLatch1, latch1);
 latch2 = 0x00;
 outportb(DirLatch2, latch2);
 outportb(0x21, 0x28);
 mensajes("Sincronizacion finalizada. ", ColorMensaje);
 free(nombre);
 return; }                                            /* Fin al_numero_imagen( ) */
```

```
/* Sincroniza digitación */
void digitaliza( )
{long cuadro=0x00;
 union llave {    char ch[2];
                  int i; } c;                          /* Tecla presionada */
 void interrupt congela( );                            /* Para suspender información a pantalla */
 void interrupt (*oldcom2)( );                         /* Rutina anterior de serv de int */
 digita = 0x00;
 inicializacom2( );                                    /* Inicializa Com2 para Rx del digitador */
 oldcom2 = getvect(0x0b);                              /* Rutina anterior de serv de int */
 setvect(0x0b, congela);                               /* Nueva rutina de serv de int */
 outportb(0x21, 0x20);                                 /* Habilita int IRQ3, Com2 */
 c.i = 0x00;                                           /* Inicializa tecla */
 mensajes("Proceso  de digitalizacion...", ColorMensaje);
 mensajes("ESC para cancelar.", ColorMensaje);

 while(c.ch[0] != ESC) {                               /* Ciclo de espera para salida */
        if(camara1 != cuadro) {  cuadro = camara1;     /* Ssi nueva imagen */
                                 obtienedatos( ); }    /*Oobtiene datos de disco y los despliega*/
        if(digita) {       mensajes("Imagen congelada.", ColorMensaje);
                           while(digita) if(kbhit( ) digita = 0x00;
                           mensajes("..... ", ColorMensaje); }
        if(kbhit( ) c.i = bioskey(0); }                /* Lee tecla presionada */

 outportb(0x21, 0x28);                                 /*Restaura controlador de interrupciones*/
setvect(0x0b, oldcom2);                                /* Restaura vector de int */
 com2reset( );                                         /* Restaura com 2 */
 mensajes("Digitalizacion terminada.", ColorMensaje);
 reset ; }                                             /* Fin digitaliza( ) */

/* Rutina de servicio para interrupción de digitación */
void interrupt congela( )
{disable( );                                           /* Deshabilita interrupciones */
 if(digita = = 0) {      digita = inportb(0x2f8);      /* Lee caracter de com2 */
                         digita=0x01; }                /* Bandera de digitación True */
        else {           digita = inportb(0x2f8);      /* Lee caracter de com2 */
                            digita = 0x00; }           /* Bandera de digitación false */
 outportb(0x20,0x63);                                  /* Borra bit IRQ3 del controlador */
 enable( );                                            /* Habilita int */
}                                                      /* Fin congela( )*/

/* Despliega datos del gps basados en la hhmmss */
void tiempodato( )
{long hhmmss;                                          /* hhmmss de la imagen */
 char *nombre = 0x00;                                  /* Lee hhmmss del teclado */
 nombre = malloc(8);
 *nombre = 0x00;
 mensajes("Tiempo (hhmmss) ? : ", ColorMensaje);
 leedato(nombre);
 if(*nombre = = 0x00){ mensajes("Cancelada la operacion.", ColorMensaje);
                       return; }                       /* Aborta procedimiento */
 hhmmss = atol(nombre);                                /* Convierte de cadena a numérico */
 if(!hhmmss) { mensajes("Error: Dato no num rico, operacion cancelada.", ColorMensaje);
                       return; }                       /* Aborta procedimiento */
```

```
if((archivo[0] != 0x00) && (filelength(fileno(pfichero)) != 0)){     /* Si no archivo o vacío */
        buscadatogps(hhmmss);                                       /* Acceso aleatorio */
        fread(&datos, bytesdatos, 1, pfichero);                     /* Lee datos de archivo */
        camara1 = datos.noimagen;
        datosapantalla( );                                          /* Despliega datos */
         mensajes("Operacion concluida.", ColorMensaje); }
else    mensajes("Ningun archivo abierto.", ColorMensaje);
return ;
}                                                                   /* Fin tiempodato( ) */

/* Modo Cam grabación de imágenes, condiciones iniciales y tarea general */
void cam( )
{
 int i;
 void interrupt txcom1( );                                          /* Servicio de int a Tx del No de Img*/
 void interrupt vertsync( );                                        /* Servicio de int a sync vertical */
 void interrupt rx_gps( );                                          /* Servicio de int a Rx del GPS */
 void interrupt (*oldcom1)( );                                      /* Apuntadores a rutinas anteriores */
 void interrupt (*oldlpt)( );
 void interrupt (*oldcom2)( );

 inicializacom1tx( );                                               /* Inicializa Com1 para Tx No. Imagen */
 inicializacom2( );                                                 /* Inicializa Com2 comunicación Gps */
 oldcom1 = getvect(0x0a);                                           /* Dir de servicio de int anterior */

 setvect(0x0f, vertsync);
 latch1 = 0x00;                                                     /* Inicializa latch's de la intefaz */
 outportb(DirLatch1, latch1);
 latch2 = 0x00;
 outportb(DirLatch2, latch2);
 byte1 = 0x00;                                                      /* Inicializa variables */
 byte2 = 0x00;
 byte3 = 0x00;
 nocuadro = 0x00;
 txbander = 0x01;
 tx_cual = 0x00;
 datos_gps = 0x00;
 dato_gps = 0x00;
 cual_gps = 0x01;
 *cad_gps1 = 0x00;
 *cad_gps2 = 0x00;
 posicioncaracter = 0;
 for(i=0; i<NumeroCamaras; i++) funcion[i] = Stby;                  /* Arrancan en stby las cámaras*/
 pantallafuncion( );
 outportb(0x21, 0xB0);                                              /* Habilita IRQ3 = com2 = gps */
 mensajes("Camara en modo Cam.", ColorMensaje);

 while(modo_operacion == Modo_Cam) {                                /* Ciclo de espera para salida */
        almacena( );        /* datos a disco */
        if(kbhit()) {       activamenu( );                          /* Atiende al usuario */
                            if(nueva_funcion) ejecuta( ); } }
```

```
latch1 = 0x00;                                  /* Restaura latch's de la interfaz */
outportb(DirLatch1, latch1);
latch2 = 0x00;
outportb(DirLatch2, latch2);
delay(1);
outportb(0x21, 0xB8);                           /* Restaura controlador de int */
setvect(0x0c, oldcom1);                         /* Restaura vector de int */
setvect(0x0b, oldcom2);
setvect(0x0f, oldlpt);
cerrar( );                                      /* Cierra archivo */
com1reset( );                                   /* Restaura puertos com1 y com2*/
com2reset();
mensajes("Camara en modo Off.", ColorMensaje);
return;
}                                               /* Fin cam( ) */

/* Servicio de interrupción para sincronizar tx del # imagen */
void interrupt vertsync( )
{
disable( );                                     /* No interrupciones hardware */
outportb(0x20, 0x67);                           /* Borra IRQ7 de controlador 8259 */
if(txbander) {    if((++byte1) == 128){ byte1 = 0x00;   /* Incrementa número de imagen*/
                                        if((++byte2) == 128){ byte2 = 0x00;
                                                              byte3++; } }
                  nocuadro = byte3;             /* Actualiza nocuadro */
                  nocuadro = nocuadro*0x4000;
                  nocuadro += byte1+(byte2*0x80);
                  txbander = 0x00;              /*Actualiza banderas de Tx */
                  tx_cual = 0x00;
                  outportb(0x21, 0x20); }       /* Habilita IRQ4 */
else              txbander = 0x01;              /* 2 pulsos de sync vert por imagen */
enable( );                                      /* Si interrupciones hardware */
}                                               /* Fin vertsync( ) */

/* Servicio de intrrupción para tx del numero de imagen */
void interrupt txcom1( )
{disable( );                                    /* No interrupciones hardware */
 outportb(0x20, 0x64);                          /* Borra IRQ4 del 8259 */
 switch(tx_cual){ case 0:  outportb(0x3f8, 0x80);   /* Tx cabecera de cada imagen */
                           tx_cual++;
                           break;
                  case 1:  outportb(0x3f8, byte1);  /* Tx byte 1 del # de imagen */
                           tx_cual++;
                           break;
                  case 2:  outportb(0x3f8, byte2);  /* Tx byte 2 del # de imagen */
                           tx_cual++;
                           break;
                  case 3:  outportb(0x3f8, byte3);  /* Tx byte 3 del # de imagen */
                           tx_cual = 0x00;          /* Reset tx_cual */
                           outportb(0x21, 0x30);    /* No IRQ4, si IRQ7 */
                           break;
                  default : ;}
 enable( );                                     /* Si interrupciones hardware */
}                                               /* Fin txcom1( ) */
```

```
/* Servicio de interrupción para Rx de datos del Gps */
void interrupt rx_gps()
{
disable( );                                          /* No interrupciones hardware */
outportb(0x20, 0x63);                                /* Borra IRQ3 del 8259 */
if(cual_gps) {
        cad_gps1[posicioncaracter] = inportb(0x2f8);  /* Lee caracter enviado por gps */
        cad_gps1[posicioncaracter+1] = 0x00;          /* Caracter de fin de cadena */
        if(cad_gps1[posicioncaracter] = = 0x0A){      /* Si caracter de fin de mensaje */
                posicioncaracter = 0;                 /* Actualiza apuntador */
                cual_gps = 0x00;                      /* Conmuta cadena */
                dato_gps = 0x01; }                    /* Bandera de mensaje */
        else    posicioncaracter++; }                 /* Si no incrementa apuntador */
else {  cad_gps2[posicioncaracter] = inportb(0x2f8);
        cad_gps2[posicioncaracter+1] = 0x00;
        if(cad_gps2[posicioncaracter] = = 0x0A){
                posicioncaracter = 0;
                cual_gps = 0x01;
                dato_gps = 0x01; }
        else    posicioncaracter++; }
 enable( );                                          /* Si interrupciones hardware */
}                                                    /* Fin rx_gps( ) */

/* Almacena en disco la información de gps, sensores, #imagen */
void almacena( )
{
datos.noimagen = nocuadro;                           /* Actualiza No de Img*/
leeinclinometros( );                                 /* Adquiere información de inclinación */
if(dato_gps){                                        /* Si mensaje del GPS */
        if(cual_gps){   strcpy(mensaje_gps, cad_gps2); /* Actualiza mensaje del GPS */
                        revisamensajegps( );         /* Revisa el mensaje */
                        cad_gps2[0] = 0x00; }        /* Restaura cadena para otro mensaje */
        else {          strcpy(mensaje_gps, cad_gps1); /* Lo mismo para la otra cadena */
                        revisamensajegps();
                        cad_gps1[0] = 0x00; }
        dato_gps = 0x00; }                           /* Actualiza bandera de mensaje GPS */
if(datos_gps){                                       /*Nuevo bloque de información del GPS*/
        if((funcion[0] = = Rec) && (archivo[0] != 0x00))
                fwrite(&datos, bytesdatos, 1, pfichero);  /* Almacena en disco */
        datos_gps = 0x00; }                          /* Actualiza bandera de bloque de info */
camara1=(camara2=(camara3=(camara4=nocuadro)));      /* Actualiza No de Img */
datosapantalla( );                                   /* Actualiza pantalla */
return }                                             /* Fin almacena( ) */

/* Función de grabar cintas */
void grabacintas( )
{
long checa;
int i;

latch2 = latch2 & 0x03;                              /* Canal cero del remoto */
for(i=0; i<NumeroCamaras; i++) {                     /* Envía comando */
        if(funcion[i] != Rec) enviafuncion(Rec, latch2, 600);
        latch2 = latch2 + 0x04; }                    /* Siguiente canal */
```

```
outportb(0x21, 0x30);                                   /* Habilita IRQ7 */
checa = nocuadro;                                       /* Revisa que recibe sincronía vertical */
delay(100);                                             /* Espera 100 mseg */
if(checa == nocuadro){  enciendecamaras( );             /* Si no recibe sync vert */
                        latch2 = latch2 & 0x03;         /* Envía comando */
                        for(i=0; i<NumeroCamaras; i++) {    uncion[i] = Stby;
                                                            enviafuncion(Rec, latch2, 600);
                                                            latch2 = latch2 + 0x04; } }

mensajes("Grabando cintas...", ColorMensaje);
for(i=1; i<1000; i++){ delay(1);                        /* Tiempo de respuesta de la cam */
                       i++ }
latch2 = latch2 & 0x03;
return ; }                                              /* Fin grabacintas ( ) */

/* Despierta una cámara y manda rec a una sola */
void grabaenuna( )
{
long numero = 0;
char *nombre;
nombre = malloc(8);                                     /* Obtiene el número de cámara */
*nombre = 0x00;
mensajes("Numero de Camara? ", ColorMensaje);
leedato(nombre);                                        /* Lee el número de cámara */
if(*nombre == 0x00){    mensajes("Cancelada la operacion.", ColorMensaje);
                        return ; }                      /* Si vacío */
numero = atol(nombre);                                  /* De caracter a numérico */
if(!numero) {     mensajes("Cancelada la operacion.", ColorMensaje);
                  return ; }                            /* Si no numérico */
enciendecamaras( );                                     /* Enciende las cámaras */
numero - -;                                             /* Canal cero = cámara 1 */
if(funcion[numero] == Rec)      funcion[numero] = Stby; /* Restaura la instrucción anterior */
                        else    funcion[numero] = Rec;
latch2 = 0x04*numero;                                   /* Calcula el canal */
enviafuncion(Rec, latch2, 400);                         /* Envía función */
mensajes("Comando enviado.", ColorMensaje);
latch2 = 0x00;                                          /* Restaura latch de la interfaz */
outportb(DirLatch2, latch2);
return ;
}                                                       /* Fin grabaenuna( ) */

/* Función de cintas listas */
void listascintas( )
{
int i;
latch2 = latch2 & 0x03;                                 /* Canal cero del remoto */
for(i=0; i<NumeroCamaras; i++) {                        /* Envía comando */
        if(funcion[i] != Stby) enviafuncion(Rec, latch2, 700);
        latch2 = latch2 + 0x04; }
outportb(0x21, 0xB0);                                   /* No habilita IRQ7 */
mensajes("Cintas listas para grabar...", ColorMensaje);
for(i=1; i<1000; i++){ delay(1);                        /* Tiempo de respuesta */
                       i++}
latch2 = latch2 & 0x03;
return ; }                                              /* Fin listascintas( ) */
```

```
/* Despierta las cámaras en modo grabación */
void enciendecamaras( )
{
int i;
outportb(DirLatch2, 0xF0);                          /* Despierta las 4 cámaras */
delay(1);
outportb(DirLatch2, 0x00);                          /* Restaura latch de la intefaz */
mensajes("Encendiendo camaras..... ", ColorMensaje);
for(i=1; i<2000; i++){ delay(1);                    /* Tiempo de respuesta */
                       i++ ; }
mensajes("Camaras encendidas, listas para grabar. ", ColorMensaje);
return ; }                                          /* Fin enciendecamaras( ) */

/* Acercamiento, solo sobre la última cámara */
void acerca_camara( )
{
long cuadro;
mensajes("Operacion de acercamiento de la ultima camara. ", ColorMensaje);
latch2 = (NumeroCamaras-1)*0x05;                    /* Canal de última cámara */
outportb(DirLatch2, latch2);
delay(1);
outportb(DirLatch1, Zoom_mas);                      /* Envía comando */
mensajes("Acercando, presiona cualquier tecla para cancelar....", ColorMensaje);
cuadro = nocuadro;
while(!kbhit( ))                                    /* Espera a que cancele operación */
        if(nocuadro != cuadro) almacena();          /* No suspende el almacenamiento */
outportb(DirLatch1, Cero);                          /* Cancela operación */
mensajes("Acercamiento cancelado.",ColorMensaje);
delay(1);
latch2 = 0x00;                                      /* Canal cero */
outportb(DirLatch2, latch2);
return ;
}                                                   /* Fin acerca_camara( ) */

/* Alejamiento, solo sobre la última cámara */
void aleja_camara( )
{
long cuadro;
mensajes("Operacion de alejamiento de la ultima camara. ", ColorMensaje);
latch2 = (NumeroCamaras-1)*0x05;                    /* Canal de la última cámara */
outportb(DirLatch2, latch2);
delay(1);
outportb(DirLatch1, Zoom_menos);                    /* Envía comando */
mensajes("Alejando, presiona cualquier tecla para cancelar....", ColorMensaje);
cuadro = nocuadro;
while(!kbhit( ))                                    /* Espera a que cancele operación */
        if(nocuadro != cuadro) almacena( );         /* No suspende el almacenamiento */
outportb(DirLatch1, Cero);                          /* Restaura el latch de la interfaz */
mensajes("Alejamiento cancelado.",ColorMensaje);
delay(1);
latch2 = 0x00;                                      /* Canal cero */
outportb(DirLatch2, latch2);
return ;
}                                                   /* Fin aleja_camara( ) */
```

```
/* Lee sensores de inclinación */
void leeinclinometros( )
{outportb(DirA_D, 0x04);                          /* Selecciona canal 1 del A/D */
 delay(1);                                        /* Espera tiempo de conversión */
 datos.alabeo = inportb(DirA_D);                  /* Lee el dato del canal uno */
 delay(1);
 outportb(DirA_D, 0x05);                          /* Selecciona canal 2 del A/D */
 delay(1);                                        /* Tiempo de conversión */
 datos.cabeceo = inportb(DirA_D);                 /* Lee el dato del canal dos */
 return ;}                                        /* Fin leeinclinometros( ) */



                case GPGGA:   gpgga(mensaje_gps);
                              break;
                case PMGLB:   pmglb(mensaje_gps);
                              break;
                case GPZDA:   gpzda(mensaje_gps);
                              break;
                case PMGLG:   pmglg(mensaje_gps);
                              break;
                case PMGLF:   pmglf(mensaje_gps);
                              break;
                default:      mensajes(mensaje_gps, ColorMensaje);
                              break; } }
  else     mensajes("Error en lectura de com2.", ColorMensaje);
 return ;}                                        /* Fin revisamensaje( ) */
/* Revisa si contiene error el mensaje del gps */
int checa_gps( )
{char sumcheck1[2], *apunta, asterisco = '*';
  int x = 1;                                      /* Si error a menos de otra cosa */
  generachecksum(sumcheck1);                      /* Genera el caracter de error para compararlo */
  sumcheck1[2] = 0x00;                            /* Fin de cadena */
  apunta = strchr(mensaje_gps, asterisco);        /* Copia el caracter de comprobación error */
  if(apunta)                                      /* Si existe y son iguales, no error*/
        if((sumcheck1[0] = = *(apunta + 1)) && (sumcheck1[1] = = *(apunta + 2))) x = 0;
  return(x); }                                    /* Fin checa_gps( ) */
/* Genera caracteres de check sum, protocolo de comunicación con gps */
void generachecksum(sumcheck2)
char *sumcheck2;
{unsigned chk_gps;
  int temp = 2;



  *sumcheck2 = (chk_gps > > 4);
  if(*sumcheck2 < 0x0A) *sumcheck2 + = 0x30;
                  else      *sumcheck2 + = 0x37;
  if(*(sumcheck2 + 1) < 0x0A)     *(sumcheck2 + 1) + = 0x30;
                        else      *(sumcheck2 + 1) + = 0x37;
*(sumcheck2 + 2) = 0x00;                          /* Fin de cadena */
return }                                          /* Fin generachecksum( ) */
```

```
/* Revisa el tipo de mensaje del gps */
int tipomensajegps( )
{                                                    /* Compara cadenas para determinar el tipo */
if(strstr(mensaje_gps, "$GPGGA,") == mensaje_gps) return(GPGGA);
if(strstr(mensaje_gps, "$PMGLB,") == mensaje_gps) return(PMGLB);
if(strstr(mensaje_gps, "$GPZDA,") == mensaje_gps) return(GPZDA);
if(strstr(mensaje_gps, "$PMGLG,") == mensaje_gps) return(PMGLG);
if(strstr(mensaje_gps, "$PMGLF,") == mensaje_gps) return(PMGLF);
return(0); }                                         /* Fin tipomensajegps( ) */

/* Obtiene parámetros del mensaje $GPGGA del gps */
void gpgga( )
{char *latitud; *longitud; char *signo;
 latitud = malloc(7);
 longitud = malloc(8);
 signo = malloc(1);
 strncpy(latitud, (mensaje_gps+14), 7);              /* Obtiene latitud */
 *(latitud+7) = 0x00;                                /* Fin de cadena */
 strncpy(signo, (mensaje_gps+22),1);                 /* Obtiene signo */
 *(signo+1) = 0x00;                                  /* Fin de cadena */
 datos.latitud = atof(latitud);                      /* Convierte a numérico */
 if(*signo == 'S') datos.latitud = datos.latitud*(-1);

 strncpy(longitud, (mensaje_gps+24), 8);             /* Obtiene longitud */
 *(longitud+8) = 0x00;
 strncpy(signo, (mensaje_gps+33),1);
 *(signo+1) = 0x00;
 datos.longitud = atof(longitud);                    /* A valor numérico */
 if(*signo == 'W') datos.longitud = datos.longitud*(-1);

 free(latitud);                                      /* Libera memoria */
 free(longitud);
 free(signo);
 return ; }                                          /* Fin gpgga( ) */

/* Obtiene parámetros del mensaje $PMGLB del gps */
void pmglb( )
{char *temp;
 temp = malloc(6);                                   /* Obtiene altitud */
 strncpy(temp, (mensaje_gps+10), 6);
 *(temp+6) = 0x00;
 datos.altitud = atol(temp);
 free(temp);                                         /* Libera memoria */
 return ; }                                          /* Fin pmglb( ) */

/* Obtiene parámetros del mensaje $GPZDA del gps */
void gpzda( )
{char *temp;
 temp = malloc(6);                                   /* Obtiene la hora */
 strncpy(temp, (mensaje_gps+7), 6);
 *(temp+6) = 0x00;
 datos.tiempo = atol(temp);
 free(temp);                                         /* Libera memoria */
 return , }                                          /* Fin gpzda( ) */
```

```
/* Obtiene parámetros del mensaje $PMGLG del gps */
void pmglg( )
{char *temp;
 temp = malloc(3);                                   /* Obtiene el error estimado */
 strncpy(temp, (mensaje_gps+24), 3);
 *(temp+3) = 0x00;
 datos.error = atoi(temp);
 free(temp);                                         /* Libera memoria */
 datos_gps = 0x01;                                   /* Último dato del bloque */
 return ;
}                                                    /* Fin pmglg( ) */

/* Obtiene parámetros del mensaje $PMGLF del gps */
void pmglf( )
{int i_gps=1, esta_gps, y=2;
 char *temp;
 temp  = malloc(8);                                  /* Obtiene satélites */
 while(i_gps !=  46){    i_gps = i_gps+9;
                         strncpy(temp, (mensaje_gps+i_gps), 8);
                         *(temp+8) = 0x00;
                         write_string(y++, 40, temp, VidNormal); }
 strncpy(temp, (mensaje_gps+55), 1);                 /* Obtiene estado del GPS */
 *(temp+1) = 0x00;
 esta_gps = atoi(temp);                              /* A numérico */
 switch(esta_gps){       case 1:  write_string(y, 40, "Ocioso  " , VidNormal);
                                  break;
                         case 2:  write_string(y, 40, "Busqueda" , VidNormal);
                                  break;
                         case 3:  write_string(y, 40, "Almanac" , VidNormal);
                                  break;
                         case 4:  write_string(y, 40, "Ephefemeris" , VidNormal);
                                  break;
                         case 5:  write_string(y, 40, "Adquisicion" , VidNormal);
                                  break;
                         case 6:  write_string(y, 40, "Posicion" , VidNormal);
                                  break;
                         case 7:  write_string(y, 40, "Navegacion" , VidNormal);
                                  break;
                         default: write_string(y, 40, "Navegacion" , VidNormal);
                                  break; }
 free(temp);                                         /* Libera memoria */
 return ; }                                          /* Fin pmglf( ) */

/* Transmite comandos al gps */
void tx_gps(gps_mensaje)
char *gps_mensaje;                                   /* Mensaje al GPS */
{
 unsigned temp;
 while(*gps_mensaje != 0x00){
        temp = inportb(0x2fd);                       /* Lee estado de Com2 */
        if(temp & 0x40) outportb(0x2f8, *gps_mensaje++) /* Transmite */
        ;}
return ;
}                                                    /* Fin tx_gps( ) */
```

```
/* Lee mensaje para tx al gps */
void al_gps( )
{
char *gps_mensaje, *sumcheck, *temp;
gps_mensaje = malloc(100);                          /* Asigna memoria */
sumcheck = malloc(3);                               /* Caracter de verificación de error */
*gps_mensaje = 0x00;                                /* Fin de cadena */
*sumcheck = 0x00;
mensajes("Comando:", ColorMensaje);
leecomandogps(gps_mensaje);                         /* Obtiene mensaje */
if(*gps_mensaje == 0x00){       mensajes("Cancelado.", ColorMensaje);
                                return ;}           /* Ningún mensaje al gps */

generachecksumtx(gps_mensaje, sumcheck);            /* Genera el caracter check sum */
temp = gps_mensaje;
while(*++temp != 0x00);                             /* Puntero al final del mensaje */
*temp++ = *sumcheck++;                              /* Agrega los de verificación de error */
*temp++ = *sumcheck++;
*temp++ = 0x0D;                                     /* Agrega los de fin de mensaje */
*temp++ = 0x0A;
*temp = 0x00;                                       /* Fin de cadena */
tx_gps(gps_mensaje);                                /* Transmite mensaje al gps*/
mensajes("Transmitido el comando.",ColorMensaje);
free(gps_mensaje);                                  /* Libera memoria */
free(sumcheck);
return ; }                                          /* Fin al_gps( ) */

/* Lee comando para tx al gps */
void leecomandogps(nombre)
char *nombre;                                       /* Mensaje al GPS */
{
union tecla {     char ch[2];
                  int i;} c;                        /* Tecla presionada */
int j;
c.ch[0] = 1;
for(j=0; c.ch[0] != '\r'; j++) {                    /* Sale si return */
        c.i = bioskey(0);                           /* Lee tecla */
        switch(c.ch[0]) {                           /* Tipo de tecla */
                case'\r': nombre[j] = 0x00;         /* Si enter fin de cadena */
                        break;
                case'\b':if(j--){                   /* Si tecla ← */
                                write_char(mensajesx-1, 10+j, 0x20, ColorMensaje);
                                j--; }              /* Borra anterior y actualiza puntero */
                        break;
                caseESC:c.ch[0] = '\r';             /* Enter para salir */
                        nombre[0] = 0x00;           /* Cadena vacía */
                        break;
                default: nombre[j] = c.ch[0];       /* Almacena caracter */
                        nombre[j+1] = 0x00;         /* Fin de cadena */
                        write_char(mensajesx-1, 10+j, c.ch[0], ColorMensaje);
                        break; } }
if(j>100) nombre[100] = 0x00;                       /* Filtro longitud del comando */
return ; }                                          /* Fin leecomandogps( ) */
```

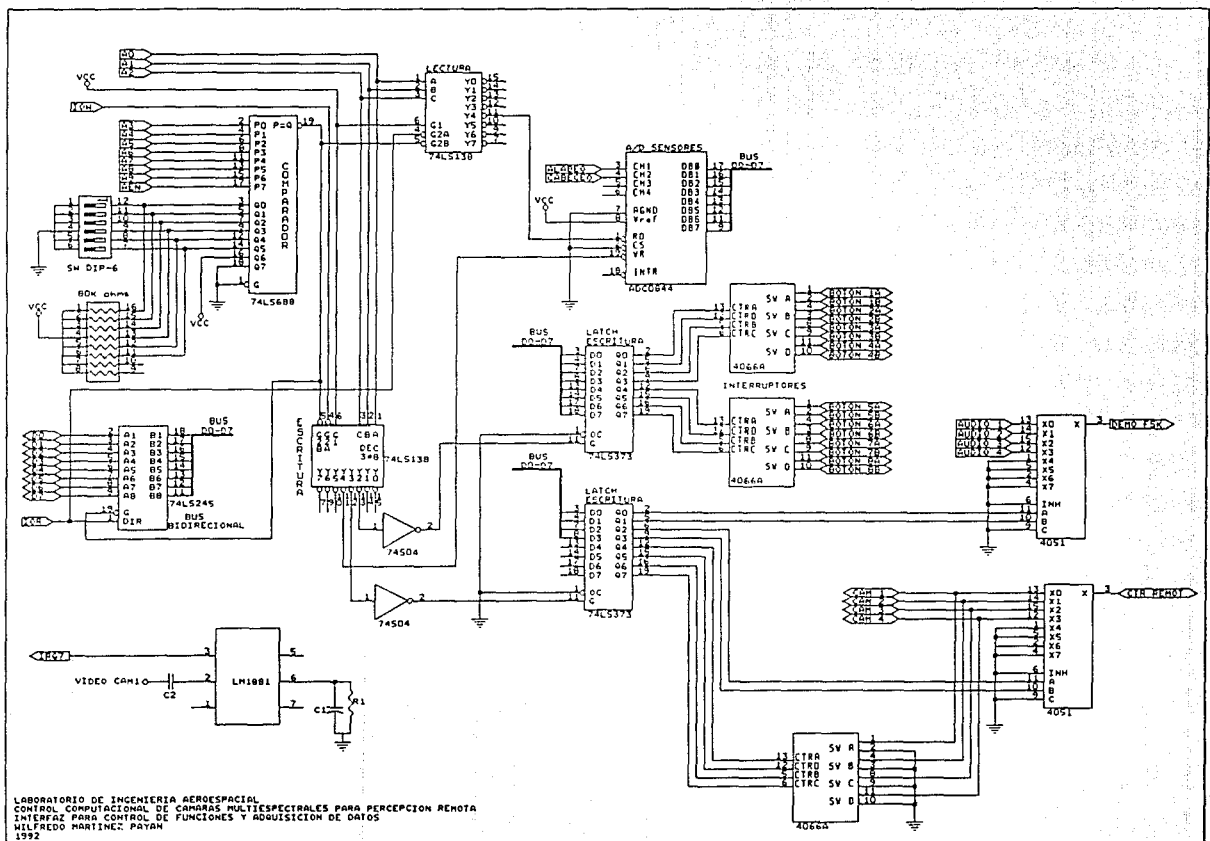
```
/* Inicializa el gps */
void inicializagps( )
{
char *gps_mensaje, *sumcheck, *temp;
mensajes("Inicializando Gps...", ColorMensaje);
gps_mensaje = malloc(70);                    /* Aigna memoria */
sumcheck = malloc(3);
temp = malloc(3);
*gps_mensaje = 0x00;                         /* Mensaje GPS */
*sumcheck = 0x00;                            /* Caracteres de verificación de error */
*temp = 0x0D;                                /* Caracteres de fin de mensaje */
*(temp+1) = 0x0A;
*(temp+2) = 0x00;

strcpy(gps_mensaje, "$PMGLN,00,1*");         /* Mensaje Master Reset */
generachecksumtx(gps_mensaje, sumcheck);     /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);               /* Concatena check sum*/
strcat(gps_mensaje, temp);                   /* Concatena fin de mensaje */
tx_gps(gps_mensaje);                         /* Transmite mensaje al gps*/
delay(100);                                  /* Necesario para tx después de un reset */

strcpy(gps_mensaje, "$PMGLI,00,F02,2,A, *");  /* Pide información de satélites */
generachecksumtx(gps_mensaje, sumcheck);      /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);
strcat(gps_mensaje, temp);
tx_gps(gps_mensaje);                          /* Transmite mensaje al gps*/
delay(1);

free(gps_mensaje);                            /* Libera memoria */
free(sumcheck);
free(temp);
mensajes("Gps inicializado.", ColorMensaje);
return ;
}                                             /* Fin inicializagps( ) */

/* Pide los datos principales al gps */
void pidedatosgps( )
{
char *gps_mensaje, *sumcheck, *temp;
mensajes("Pidiendo datos...", ColorMensaje);
gps_mensaje = malloc(70);                     /* Asigna memoria y obtiene el mensaje*/
sumcheck = malloc(3);
temp = malloc(3);
*gps_mensaje = 0x00;                          /* Mensaje a ser tx al GPS */
*sumcheck = 0x00;                             /* Verificación de error */
*temp = 0x0D;                                 /* Caracteres de fin de mensaje */
*(temp+1) = 0x0A;
*(temp+2) = 0x00;
strcpy(gps_mensaje, "$PMGLD,00,3*");          /* Modo 3D/Auto */
generachecksumtx(gps_mensaje, sumcheck);      /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);
strcat(gps_mensaje, temp);
tx_gps(gps_mensaje);                          /* Transmite mensaje al gps*/
delay(1);
```

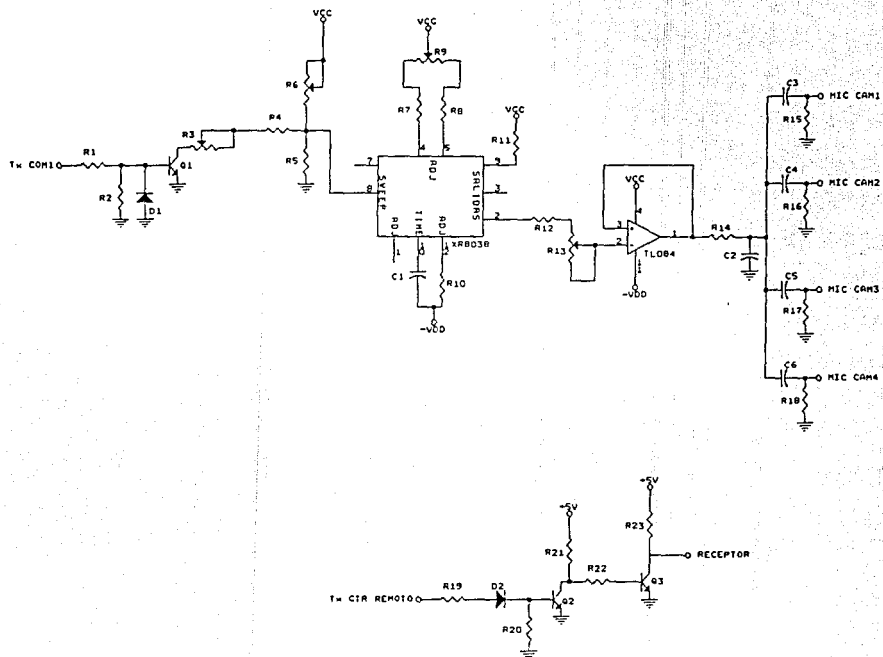
```
tx_gps(gps_mensaje);                                    /* Transmite mensaje al gps*/
delay(1);
strcpy(gps_mensaje, "$PMGLI,00,B02,2,A, *");            /* Altitud */
generachecksumtx(gps_mensaje, sumcheck);                /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);
strcat(gps_mensaje, temp);
tx_gps(gps_mensaje);                                    /* Transmite mensaje al gps*/
delay(1);
strcpy(gps_mensaje, "$PMGLI,00,A00,2,A, *");            /* Tiempo */
generachecksumtx(gps_mensaje, sumcheck);                /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);
strcat(gps_mensaje, temp);
tx_gps(gps_mensaje);                                    /* Transmite mensaje al gps*/
delay(1);
strcpy(gps_mensaje, "$PMGLI,00,G00,2,A, *");            /* Error */
generachecksumtx(gps_mensaje, sumcheck);                /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);
strcat(gps_mensaje, temp);
tx_gps(gps_mensaje);                                    /* Transmite mensaje al gps*/
delay(1);
free(gps_mensaje);                                      /* Libera memoria */
free(sumcheck);
free(temp);
mensajes("Enviados los comandos.", ColorMensaje);
return ; }                                              /* Fin pidedatosgps( ) */

/* Borra la memoria del gps */
void borragps( )
{char *gps_mensaje,*sumcheck, *temp;
mensajes("Borrando memoria del Gps...", ColorMensaje);
gps_mensaje = malloc(70);                               /* Asigna memoria */
sumcheck = malloc(3);
temp = malloc(3);
*gps_mensaje = 0x00;                                    /* Tx al GPS */
*sumcheck = 0x00;                                       /* Verificación de error */
*temp = 0x0D;                                           /* Fin de mensaje */
*(temp + 1) = 0x0A;
*(temp + 2) = 0x00;
strcpy(gps_mensaje, "$PMGLN,00,1*");                    /* Master Reset */
generachecksumtx(gps_mensaje, sumcheck);                /* Genera el caracter check sum */
strcat(gps_mensaje, sumcheck);
strcat(gps_mensaje, temp);
tx_gps(gps_mensaje);                                    /* Transmite mensaje al gps*/
delay(100);                                             /*Necesario para tx despues de un reset */
free(gps_mensaje);
free(sumcheck);
free(temp);
mensajes("Memoria borrada del Gps.", ColorMensaje);
return ;}                                               /* Fin borragps( ) */
```

```
/* Genera caracteres de check sum, protocolo de comunicación con gps tx */
void generachecksumtx(txpalabra,sumcheck2)
char *txpalabra;                                    /* Mensaje tipo gps */
char *sumcheck2;                                    /* Caracter de verificación de error */
{
unsigned chk_gps;
int temp=2;

chk_gps = *(txpalabra+1);                           /* A partir del primer caracter */
for(temp=2; *(txpalabra+temp) != '*'; temp++)       /* Calcula sumcheck */
        if(*(txpalabra+temp) != 0x00) chk_gps = (chk_gps ^ *(txpalabra+temp));
*(sumcheck2+1) = (chk_gps & 0x0F);                  /* Convierte a ASCII */
*sumcheck2 = (chk_gps >> 4);
if(*sumcheck2 < 0x0A)  *sumcheck2 += 0x30;
                else   *sumcheck2 += 0x37;
if(*(sumcheck2+1) < 0x0A)      *(sumcheck2+1) += 0x30;
                else           *(sumcheck2+1) += 0x37;
*(sumcheck2+2) = 0x00;                              /* Fin de cadena */
return ;
}                                                   /* Fin generachecsumtx( ) */
```

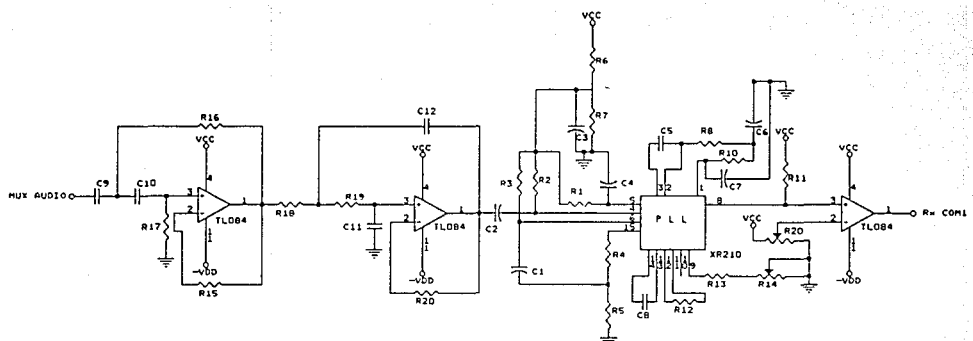
```
/*
Tesis: Control Computacional de Cámaras Multiespectrales Aerotransportadas para Percepción Remota
Autor: Wilfredo Martínez Payán
Laboratorio de Ingeniería Aeroespacial
Instituto de Ingeniería , U.N.A.M.
Noviembre de 1992
Programa residente en la memoria de la microcomputadora digitadora */


#include <dos.h>                        /* Librerías utilizadas */
#include <bios.h>
#include <stdio.h>
#include <conio.h>

#define INTR 0x09                       /* Interrupción de teclado */

extern unsigned _heaplen = 1024;        /* Reduce el espacio asignado en memoria para el programa */
extern unsigned _stklen  = 1512;
void inicializacom1(void);              /* Inicializa el protocolo de comunicación de com1 */
void interrupt  atencion_a_int_teclado(); /* Rutina de atención a la int generada por  teclado */
void interrupt ( *servicio_anterior)();   /* Dirección de rutina de servicio a int por teclado */


int main(void)
{
   inicializacom1();                    /* Inicializa a com1 con el protocolo especificado */
   servicio_anterior = getvect(INTR);   /* Dir de la rutina de atención a la interrupción por teclado */
   setvect(INTR, atención_a_int_teclado); /* Modifica vector de int con la nueva rutina de servicio*/
   keep(0, (_SS + (_SP + 200 /16) - _psp)); /* Deja residente en memoria el programa */
   return 0;
}


void inicializacom1(void)               /* Inicializa el protocolo de comunicación de Com1 */
{
 outportb( 0x3fb, 0x80);                /* Accesa al registro de velocidad de transmisión */
 delay(1);
 outportb( 0x3f8, 0x0c);
 delay(1);
 outportb( 0x3f9, 0x00);                /* Velocidad de 9600 bits/segundo */
 delay(1);
 outportb( 0x3fb, 0x03);                /* No paridad, un bit de paro, datos de ocho bits */
 delay(1);
 outportb( 0x3fc, 0x00);                /* No modem, no interrupciones */
 delay(1);
 outportb( 0x3f9, 0x00);                /* No interrupciones */
}
```

```
void interrupt atencion_a_int_teclado( )        /* Rutina de atención a la int generada por  teclado */
{
    servicio_anterior();                        /* Llama a la rutina anterior de servicio de interrupción del teclado */
    asm push ax                                 /* Guarda en la pila los registros */
    asm push dx
    asm push cx
    asm push bx
    asm mov ah,1
    asm int 16h                                 /* Interrupción al BIOS, teclado */
    asm jz  fin                                 /* Verifica si existe caracter  */
    asm mov ah,0
    asm int 16h                                 /* Interrupción al BIOS, teclado */
    asm cmp al,'c'                              /* Compara el caracter presionado */
    asm jne g2
    asm jmp g1
g2: asm cmp al,'C'
    asm jne fin
g1: asm mov dx,03f8h
    asm mov al,0ah
    asm out dx,al                               /* Si es el caracter utilizado transmite un caracter por puerto serie */
fin: asm pop bx
     asm pop cx
     asm pop dx
     asm pop ax                                 /* Restaura los registros utilizados */
}
```


LECTURA
74LS138
A/D SENSORES
ADC0844
BUS D0-D7
VCC
74LS688
SW DIP-6
80K ohms
74LS245
BUS BIDIRECCIONAL
ESCRITURA
74LS138
LATCH ESCRITURA
74LS373
INTERRUPTORES
4066A
74S04
4051
LM1881
VIDEO CAM1
VIDEO CAM2
C2
C1
R1
LABORATORIO DE INGENIERIA AEROESPACIAL
CONTROL COMPUTACIONAL DE CAMARAS MULTIESPECTRALES PARA PERCEPCION REMOTA
INTERFAZ PARA CONTROL DE FUNCIONES Y ADQUISICION DE DATOS
WILFREDO MARTINEZ PAYAN
1992



LABORATORIO DE INGENIERIA AEROESPACIAL
CONTROL COMPUTACIONAL DE CAMARAS MULTIESPECTRALES PARA PERCEPCION REMOTA
MODULADOR EN FSK Y ACONDICIONADOR DE MENSAJE DE CIA REMOTO
WILFREDO MARTINEZ PAYAN
1992



LABORATORIO DE INGENIERIA AEROESPACIAL
CONTROL COMPUTACIONAL DE CAMARAS MULTIESPECTRALES PARA PERCEPCION REMOTA
DEMODULADOR DE FSK
WILFREDO MARTINEZ PAYAN
1992

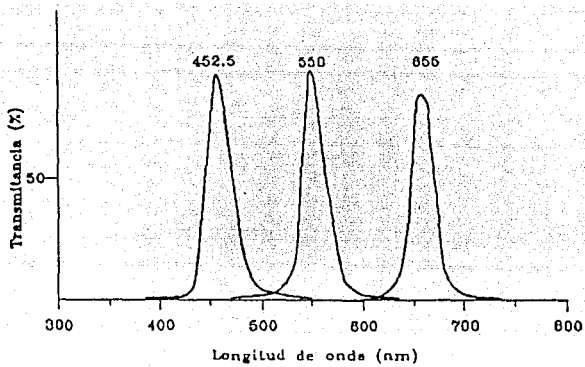


Figura No. 1 Ventanas espectrales utilizadas en las investigaciones.

Figura No.2 Imágenes tomadas en las bandas roja, verde y azul respectivamente, también se muestra la clasificación automatizada.

REFERENCIAS

[1] J. P. Antún, R. Peralta, E. Vicente, D. Hiernaux: "Aplicabilidad de Técnicas de Precepción Remota en Urbanismo y Estructura de Transporte", Informe Técnico, Proyecto 9590. II-UNAM, Jun 1989.

[2] R. Peralta, E. Vicente, J. Prado, A. Peralta: "Sistema CCD de Imágenes Multiespectrales para Discriminación de Objetos a Distancia". XXXII Congreso Nacional de Física, Soc. Mexicana de Física, León Gto., México, Oct 1989.

[3] R. Peralta, A. Peralta, E. Vicente, J. Prado, M. Navarrete: "CCD Image acquisition for multispectral teledetection". Memorias SPIE/IS&T Simposium on Electronic Imaging Sciences & Technology. San Jose, California, USA, Feb 1992.

[4] R. Peralta, A. Peralta, J. Prado, E. Vicente: "Cuantificación de áreas verdes en zonas urbanas". II Reunión SELPER-México, Aguascalientes, Ags, Sep 1991.

[5] J. Prado, R. Peralta, A. Peralta, C. Díaz: "Métodos para cuantificación de pérdidas de suelo por erosión". II Reunión SELPER-México. Aguascalientes, Ags. Sep 1991.

[6]. Peralta, R. Peralta, J. Prado, C. Díaz: "Estudios agrícolas y forestales con imágenes multiespectrales de alta resolución". II Reunión SELPER-México. Aguascalientes, Ags. Sep 1991.

[7] A. Peralta, J. Prado, M. Navarrete, E. Vicente, R. Peralta: "Sistema para obtención de imágenes aéreas multiespectrales de bajo costo", VI Congreso Nacional de Instrumentación, Guanajuato, Gto., Sep 1990.

[8] R. Peralta, A. Peralta, J. Prado, M. Navarrete, C. Díaz, W. Martínez: "Dinámica Campo-Ciudad: Análisis por Videogrametría". Congreso Internacional de Antropología e Historia. Simposio "Cuenca del Golfo y Megalópolis". CECODES. Veracruz, México, Sep 1992.

[9] Lewis C. Eggebrecht: "Interfacing to the IBM Personal Computer", Howard W. Sams & Co. A Division of Macmillam, Inc., 1987.

[10] Paul H. Young: "Electronic Communication Techniques ", Second Edition, Merrill Publishing Company, 1990.

[11] Christopher L. Morgan & Mitchell Waite: "Introducción al Microprocesador 8086/8088 (16 bit)", McGraw-Hill/Interamericana de México, S.A. de C.V., 1988.

[12] Robert F. Coughlin & Frederick F. Driscoll: "Circuitos Integrados Lineales y Amplificadores Operacionales", Segunda edición, Prentice-Hall Hispanoamericana, 1987.

[13] Mischa Schwartz: "Transmisión de Información, Modulación y Ruido", Libros McGraw-Hill de México, S.A. de C.V., 1983.

[14] Bernard Grob: "Basic Television and Video Systems", Fifth edition, McGraw-Hill, Inc, 1984.

[15] Boylestad Nashelsky: "Electrónica Teoría de Circuitos", Prentice-Hall, 1983.

[16] Handbook: "Linear Applications", National Semiconductor, 1978.

[17] Naba Barkakati: "The Waite Group's Microsoft C Bible", Second Edition, SAMS, 1990.

[18] Herbert Schildt: "C: Power User's Guide", McGraw-Hill, Inc., 1988.

[19] Brian W Kernighan & Dennis M. Ritchie: "El lenguaje de programación C", Segunda edición, Prentice Hall Hispanoamericana, S.A., 1991.

[20] Ceballos F. J.: "Curso de programación con C, Microsoft C", Macrobit & RA-MA, 1990.

[21] User Guide: "Magellan OEM GPS Module", Magellan Systems Corporation, 1992.

[22] Brian C. Fenton: "The GPS Navigation System", Popular Electronics, June 1992.

[23] User's Guide: "Mx 4200 PC Controller", Magnavox Electronic Systems Company, 1991.

[24] Service Manual CCD TR7 Video Camera Recorde, SONY