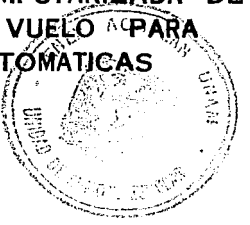
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"A C A T L A N"

REPRESENTACION COMPUTARIZADA DE PARAMETROS DE VUELO PARA AERONAVES AUTOMATICAS



T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

LUIS ALFONSO LEON GARCIA

Acatlán, Edo. de México 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

APENDICES.

BIBLIOGRAFIA.

INTRODUCCION.

La UNAM, con la colaboración del IPN, desarrolla actualmente el proyecto denominado Aeronave de control remoto multimisional para aplicaciones de percepción remota, cuyo propósito principal es la adquisición de imágenes multiespectrales, tanto de video como fotográficas. Este tipo de tecnología permite, con medianos recursos, establecer una plataforma para obtener fotografías con características comparables a las obtenidas con satélites comerciales actuales como el SPOT (Francés), el Landsat (EUA) y el Soyuz (Soviético). La percepción remota constituye la herramienta más eficiente para obtener información para localizar, explotar y cuantificar nuestros recursos, y problemas como erosión, desertificación, etc. Algunas de las aplicaciones de este tipo de aeronaves, son: estudios de crecimiento urbano, clasificación de la producción agrícola, estudio de estructuras geológicas, determinación de índices de contaminación en mantos acuíferos, prospección minera, vigilancia costera, combate y cuantificación de daños por incendios, inundaciones, sismos, etc. El sistema aeronave de control remoto está compuesto por una estación terrena móvil, para controlar y guiar el avión, la cual es una aeronave semi-automatizada, y su carga útil, intercambiable según el tipo de misión.

La estación terrena movil contiene equipo de comunicaciones, cómputo, tablero con bastones de mando y monitores para el despliegue de telemetría, video y alarmas, este equipo es comandado por una persona experimentada que le llamaremos "piloto". El avión cuenta con equipo de comunicaciones, computadora de vuelo, sensores, servos y actuadores. A través del enlace de comunicaciones la estación terrena movil transmite al avión los comandos generados por el piloto. En el avión, la computadora de vuelo recibe los comandos por medio de equipo receptor y dosifica señales electricas a los servomecanismos apropiados. Posteriormente la computadora de vuelo transmite a la estación terrena movil las variables principales de la aeronave, en forma de paquete telemétrico, cerrando así el lazo de control.

Para efectuar el control del avión es importante mostrarle al piloto información clara y actualizada de las variables primordiales de vuelo. En este caso, los datos de la aeronave se presentan por medio de una pantalla de computadora, en forma similar al empleado en los más modernos equipos de aviación y simuladores de vuelo.

Ante estas necesidades, los objetivos de esta tesis se dirigen hacia la realización de un paquete de programas para graficar los instrumentos de vuelo de la aeronave de control remoto en una pantalla de computadora, estableciendo así la interfaz hombre-máquina del sistema. La programación en modulos, utilizando la técnica denominada programación orientada por objetos, en donde, cada módulo programado realiza una función en especifico, desde la más sencilla, como la graficación de puntos, rectas y curvas, hasta las más complejas, como el diseño y animación de instrumentos. Pruebas de lectura de un archivo de disco que contenía parámetros de un vuelo simulado. Con esto se probó la velocidad de lectura de los programas y la velocidad de animación. Tales datos simulan un despegue y una trayectoria de forma elíptica que culmina con el aterrizaje. Pruebas entre dos computadoras, una para simular el comportamiento de la aeronave y la otra para realizar el despliegue de instrumentos; esta última, además, superviza el tablero, captura y procesa el bloque telemétrico y, para finalmente, generar un archivo de respaldo.

CAPITULO I. Representación de instrumentos en el proyecto Aeronave de Control Remoto.

1.1 Introducción.

En este capítulo se hace una breve descripción del proyecto aeronave de control remoto, sus alcances y objetivos, posteriormente se describe el tablero de instrumentos de la estación terrena, frente de este tablero se coloca el piloto. Finalmente, se plantea el problema de la representación de parámetros de vuelo para informar el estado de vuelo.

La intención de utilizar este tipo de aeronaves es que son de bajo costo en: su construcción, su mantenimiento y su operación. Otra de las razones es que no son tripuladas, esto es, el piloto y el copiloto controlan la aeronave desde tierra. El equipo fotográfico que se puede instalar en estas aeronaves se puede obtener en cualquier centro comercial, a diferencia de los utilizados por el INEGI o la Procuraduría General de la República, que tienen que importarlo y gastar en el extranjero para su mantenimiento, dado que sus dispositivos, tales como las lentes, son muy sensibles y frágiles.

1.2 Objetivos del proyecto aeronave de control remoto.

La aeronave de control remoto constituye una alternativa tecnológica para realizar teledetección de bajo costo con resultados similares a los obtenidos a través de satélites comerciales. No obstante que la mayor parte de las experiencias en este tipo de aeronaves se han concentrado en costosas aplicaciones militares, los avances de dominio público recientes en cuanto a: capacidad de cómputo, técnicas de control digital, telecomunicaciones de alta fiabilidad, automatización y materiales aeroespaciales, hacen posible el diseño y construcción de aeronaves como estas, con recursos financieros de alcance a las investigaciones civiles.

Para realizar la integración de la aeronave de control remoto se utilizan aquellos componentes y subsistemas comerciales que destacan por sus altos niveles de control de calidad y que al ser producidos masivamente tienen precios moderados, la aplicación de este concepto en todo el proyecto, es una de las causas principales de ahorro.

El alto valor agregado de la tecnología aeroespacial se debe en parte al ingenio con que los grupos de ingeniería desarrollan e integran equipo para solucionar problemas que hasta entonces habían sido nulas o escasamente abordados. En el proyecto aeronave de control remoto se persigue la integración de equipo conocido y probado, de costo reducido, para producir instrumentación y automatización de alto valor agregado. Son por tanto objetivos generales del proyecto aeronave de control remoto:

1) diseñar, construir y probar una plataforma de adquisición de imágenes para hacer investigación en percepción remota;

2) desarrollar y probar diversas cargas útiles para evaluar alcances del equipo completo y demostrar la facilidad de las aplicaciones del sistema;

3) desarrollar e integrar equipo de alta calidad, utilizando para ello los productos más adecuados del mercado para asegurar el buen rendimiento de sus sistemas;

4) hacer investigación y desarrollo tecnológico de alto nivel para alcanzar altos índices de robustez y confiabilidad en los subsistemas electrónico, computacional, de comunicaciones y control; y

5) dar lugar a proyectos de colaboración con usuarios internos y externos a la UNAM.

1.3 Interfaz Piloto-Aeronave.

Según se mencionó, el proyecto aeronave de control remoto está compuesto por una aeronave semi-automatizada y por la estación terrena movil. En la figura 1, al final del trabajo, se muestra la unidad móvil usada para trasladar el equipo al lugar donde se efectuará la misión. El equipo instalado en la estación terrena movil es el siguiente:

1) Antenas y equipo de comunicaciones para enlazarse con el avión.

2) Tablero de instrumentos compuesto por las siguientes partes:

a) Instrumentos (bastones de mando) para mover las superficies de control (aletas, alerones, timones y canard) y parámetros de operación de la aeronave (acelerador, dirección y freno del tren durante despegues y aterrizajes).

b) Monitores de video transmitido en tiempo real por la aeronave, mapa digital con la ruta a seguir durante la misión, mapa cartográfico de la zona de vuelo, áreas restringidas de vuelo, (líneas de alta tensión, etc.), diagramas de mantenimiento preventivo y correctivo para casos de emergencia, datos climatológicos, alarmas e instrumentos de la aeronave de control remoto actualizados contínuamente.

c) Computadoras para la captura y graficación de datos de telemetría y lectura de instrumentos de control y guía ubicados en el tablero.

3) Equipo adicional para control de temperatura dentro de la estación terrena movil, mantenimiento y comodidad.

Durante una secuencia típica de trabajo, el piloto, sentado frente al tablero de instrumentos, ver figura 2, manipula los bastones de control. Una de las computadoras de tierra adquiere las señales anteriores, a través de circuitos de conversión análogo-digital, y las transmite al avión por medio del equipo de comunicaciones. La aeronave de control remoto se encuentra enlazado a tierra con un equipo de comunicaciones similar al empleado en la estación terrena movil, este recibe los comandos.

En seguida procesa los datos y una vez reconocidos los retrasmite a tierra para validar el comando, evitando así la ejecución de ordenes erróneas por problemas de transmisión. En tierra se recibe el comando, se compara para detectar errores, en caso de falla se envía una serie de avisos de error al avión para anular la operación. De no haber error, se retransmite la orden a la aeronave de control remoto y este la ejecutará, dosificando señales eléctricas a los servomecanismos respectivos. Una vez ejecutada la operación, la computadora de vuelo procede a enviar a tierra un bloque más de telemetría en donde se incluyen el estado de las variables de operación de la aeronave. En tierra la computadora respectiva recibe y procesa la información para actualizar los instrumentos en alguna de las pantallas del tablero. Cuando el piloto observa los cambios en los instrumentos, por reflejo desplaza los bastones de mando, cerrando así el lazo de control del sistema. Ver apéndice A, el cual describe el flujo de

acciones a ejecutar.

El equipo de comunicaciones envía video y telemetría. Los datos se transmiten, del avión a la estación terrena movil y viceversa, de manera asíncrona debido a las características de los puertos serie de las computadoras, sin embargo, el video se transmite analógicamente y de forma síncrona por lo que la estación terrena movil recibe ininterrumpidamente el video capturado. Para reducir el ancho de banda del enlace de comunicaciones se utiliza la banda de audio de la señal de video para transmitir la telemetría del avión, de tal forma que en la estación terrena movil es posible desplegar video contínuo para observación del piloto y llevar directamente el video junto con la telemetría a una cinta magnética para alamacenar permanentemente los datos de la misión. De esta forma la estación terrena movil cuenta con una "caja negra", en donde se guardan por tiempo indefinido los datos de operación de la aeronave en todo momento.

Las imágenes son captadas por dos cámaras de video, colocadas en la parte frontal del avión, con las características de acción de barrido, del horizonte hacia abajo, hasta 95^0, y de izquierda hacia la derecha haciendo un arco de más de 180^0. Ver figura 3.

1.4 Pantalla de instrumentos.

Para proporcionar la información idonea al piloto acerca de las variables de la aeronave se grafican en la pantalla de la computadora los diferentes instrumentos de la aeronave. Este monitor se encuentra colocado en el tablero de instrumentos, frente al piloto.

Para desarrollar computacionalmente este tablero se tomaron en cuenta las formas y modos de operación de los instrumentos aeronáuticos digitales de reciente uso para la aviación comercial y militar. La presentación y funcionalidad de los instrumentos elaborados en este trabajo es similar a la encontrada en productos comerciales, con la ventaja de que al ser desarrollada localmente se ajustó a las necesidades particulares. Considerando sobre todo el presupuesto con el que se cuenta, el cual, no obstante que constituye un costo de desarrollo de equipo, debe ser superado

en una forma notoria a los precios involucrados por este tipo de tecnología.

Los instrumentos aeronáuticos se dividen en cinco grupos[23]: navegación, orientación, control del vuelo, estado del motor y los auxiliares. Dentro de los instrumentos de navegación integrados a la aeronave, tenemos el sistema de posicionamiento global (GPS) y la brújula; como instrumentos de orientación espacial se incluyen el horizonte artificial, el indicador de viraje y derrape, y el de posición de las aletas; como instrumentos de control de vuelo, tenemos, altímetro, velocímetro, indicador de velocidad vertical; en cuanto a los indicadores del motor se tiene el indicador de nivel de combustible, una señal de encendido, el tacómetro, el de porcentaje de potencia generada, e indicadores de temperatura en la cabeza del cilindro, temperatura del aire a la entrada al carburador y de los gases de escape; y, por último, los instrumentos auxiliares como el indicador de temperatura del medio ambiente y el indicador de presión barométrica. A continuación se presenta una breve descripción de la función de cada instrumento:

Instrumentos de navegación.

GPS. El sistema de posicionamiento global es un receptor de radio comercial que efectúa operaciones de triangulación con base en señales de satélites útiles para calcular la posición latitud (x), longitud (y) y altitud (z) de la aeronave; como la ubicación de la estación terrena movil es conocida, con éstos datos se calcula la distancia y dirección a la que se encuentra el avión.

Brújula. Es el indicador del rumbo de la aeronave. El instrumento lleva en el centro una figura descriptiva de referencia relativa a una aeronave similar a la real. Los cambios de rumbo se visualizan al rotar la líneas expresadas en grados que estan marcadas en la base guia que envuelve a la aeronave, la cual no cambia de posición. La parte superior de la figura (nariz de la nave) apunta a la dirección de vuelo.

Instrumentos de orientación espacial.

Horizonte artificial. Presenta la orientación espacial que mantiene el avión en el aire en un momento dado. Este instrumento indica los cambios de inclinación de la aeronave (alabeo y cabeceo), ver figura 4.

Indicador de viraje y derrape. Su función es similar a la acción que efectúa un péndulo. Por ejemplo, si la aeronave se desplaza hacia la derecha y hacia arriba, el indicador en pantalla se moverá hacia la derecha con una inclinación.

Alerones. Indica la posición angular de los alerones con respecto a su plano de operación normal.

Instrumentos de control de vuelo.

Altímetro. Indica la altitud real de la aeronave. Como norma aeronáutica se toma como relación el nivel del mar o la altitud de la pista de operaciones o estación de referencia.

Velocímetro. Conocido también como "indicador de velocidad aérea". Este instrumento nos informa la velocidad relativa de la aeronave respecto al viento.

Velocidad vertical. Muestra la velocidad de ascenso o descenso. Este indicador registrará un cero durante el vuelo nivelado.

Instrumentos de motor.

Nivel de combustible. Indica el nivel de combustible en el tanque a bordo.

Motor encendido. Es un indicador de operación del motor. Indica si el motor está encendido, apagado o si tiene fallas.

Tacómetro. Proporciona las revoluciones por minuto del motor.

Temperaturas. Se muestran tres temperaturas: la del aire que entra al carburador, la de la cabeza del cilindro más caliente y la de los gases de escape.

Potencia del motor. Informa el porciento de potencia total generada por el motor.

Instrumentos auxiliares.

TMA.(Temperatura del medio ambiente), Es útil para anticipar condiciones de formación de hielo en las superficies del avión .

Presión barométrica. Sus lecturas están relacionadas con los cambios de las corrientes de viento y presencia de humedad.

CAPITULO II. Equipo de desarrollo.

2.1 Introducción.

En esta sección se describen las características del equipo de cómputo empleado en el desarrollo de la presente tesis, también se describe el lenguaje de programación adoptado, especificando sus ventajas y comparándolo con otros lenguajes existentes. Se dedica especial atención a las técnicas para programar la tarjeta de video VGA y se comenta sobre los programas elaborados para el despliegue de imágenes. Se presentan los beneficios obtenidos al integrar un paquete de graficación y, finalmente, se describen las características de modularidad de la programación diseñada.

2.2 Computadora, periféricos y lenguaje de programación.

La computadora que se emplea es una PC-386, con un monitor de despliegue en color RGB de 14", un disco duro de 80 Mbytes, 1 Mbyte de memoria RAM, un puerto serie y un puerto paralelo y una unidad de disco flexible interna de 5¼", un sistema operativo MS-DOS versión 3.3.

Para desarrollar la programación del proyecto se eligió el lenguaje C. Este lenguaje fué desarrollado en la década de los 70s por un equipo de programadores de los laboratorios BELL, en New Jersey. En los últimos años ha tomado gran auge entre la comunidad que diseña programas que interaccionan intensivamente con periféricos. Dentro de las ventajas que se obtienen al trabajar con el lenguaje C tenemos:

1) El acceso a la arquitectura del procesador, que comúnmente se efectúa con lenguaje Ensamblador.

2) Se trata de un lenguaje estructurado que permite crear programas ordenados en módulos los que se pueden accesar rápidamente para su mantenimiento.

3) Es un lenguaje que al generar código eficiente permite reducir la longitud de los programas, aumentando también la rapidez de ejecución.

4) Al tener versiones estándar, que se ejecutan en todo tipo de máquinas, se asegura el manejo de programas fuentes en diversas computadoras, aún entre aquellas que manejan sistemas operativos distintos, como el caso de UNIX.

5) Existen suficientes proveedores de paquetería especial que se integra a C para incrementar así su potencial de trabajo y reducir el tiempo para desarrollo de programas.

2.3 Programación de la tarjeta VGA.

En una computadora se pueden hacer accesos a sus periféricos mediante llamadas a interrupciones. En el apéndice B, Tabla 1, se muestran los diferentes tipos de resolución que adopta la tarjeta, así como los datos de control e interrupciones necesarias para invocarlos.

El adaptador de despliegue VGA utilizado, permite manejar diferentes tipos de resolución, y según la resolución empleada se cuenta con un número de colores en la pantalla, para lograrlo, se asignan valores específicos en registros del procesador. También hay acceso a unidades de disco, el manejo de teclado, uso de puertos, etc. En el mismo apéndice B, en la Tabla 2, se muestra un listado que incluye las diferentes interrupciones del programa ROM BIOS.

Cuando en el programa se incluye una llamada de interrupción en particular se hace un acceso al microprocesador, en este caso al 80386, el cual determina la tarea a ejecutar. Este microprecesador es de 32 bits diseñado para soportar sistemas operativos optimizados para multitareas. Es capaz de direccionar hasta 4 Gigabytes de memoria física y 64 tetrabytes de memoria virtual[24].

Los accesos empleados por los programas son a la arquitectura base en especial a los registros de propósito general. Estos registros soportan operandos de datos de 1, 8, 16 y 32 bits, y campos de 1 a 32 bits . Los ocho registros son el EAX (Acumulador), EBX (Base), ECX (Contador), EDX (Datos), ESP (Puntero de pila), EBP (Puntero base), ESI (Indice fuente) y EDI (Indice de destino).

En el caso de una interrupcion en particular, se tiene que definir el registro AH, el cual determina la instrucción a ejecutar. Los otros registros de la unidad de procesamiento central sirven para almacenar información adicional de la función BIOS por ejecutar. En el apéndice B, tabla 3, se presenta la composición de los registros de propósito general para el microprocesador mencionado anteriormente.

Para iniciar la programación con la tarjeta de video se debe de definir el tipo de resolución que se desea. Cuando se trabaja en graficación y se habla de resolución nos estamos refiriendo a la cantidad de puntos desplegados en pantalla (en sentido horizontal y vertical). La resolución empleada para la graficación del tablero es de 640 columnas por 480 renglones y 16 colores. Para lograr que la tarjeta de gráficas genere dicha resolución se debe proporcionar la información en los registros correspondientes de la forma siguiente:

```
ax = 0x12;   /* 0x indica que es un valor hexadecimal */
int86 (MODOGRAF, &regs, &regs);
```

La primera parte asigna al registro AX el modo de graficación, la segunda invoca a la interrupción int86(). La parte int significa interrupción y el 86 se refiere a la familia de microprocesadores 8086. Los argumentos incluídos definen el número de interrupción empleada y variables tipo union. El primer **&regs** representa a los registros enviados a la rutina ROM BIOS y el segundo **&regs** representa los registros devueltos desde la rutina ROM BIOS.

Una vez definido el modo de graficación se pueden realizar programas para posicionar puntos en pantalla con una gama de 16 colores diferentes. En seguida se muestra un procedimiento que asigna las coordenadas renglón, columna y color a los registros correspondientes, para graficar un punto en la pantalla:

```
ax = color;
dx = renglón;
cx = columna;
int86 (MODOGRAF, &regs, &regs);
```

AX, DX y CX son los registros de control enviados a la ROM BIOS para el empleo de la tarjeta de video. MODOGRAF define la interrupción de la pantalla de gráficos. En el apéndice C se explica la programación de las uniones y las correspondientes estructuras en el lenguaje C, muestra la construcción de las estructuras y cómo son empleadas en la macro dos.h, en la que nos apoyamos para el despliegue de gráficas.

Una vez colocado un punto en pantalla, se puede hacer acceso a él repetidamente cambiando sus coordenadas y así lograr el trazo de una línea. Las líneas más sencillas de graficar son la vertical y la horizontal. Las más complicadas resultan ser las trazas de arcos.

Para desarrollar los programas que grafican los instrumentos aeronáuticos, se recurre a algoritmos ya existentes, los de Bresenham, que generan rectas y círculos. Con la ayuda de éstos, se realizaron cuerpos geométricos con fondo de color utilizado como relleno.

Otra aplicación de la tarjeta VGA consiste en el despliegue de imágenes almacenadas en archivos, que se obtienen con un barredor digital, con un digitador de imágenes, etc. Para el despliegue de ellas en la pantalla se emplea el modo de graficación de 320 columnas por 200 renglones y 256 colores, por ejemplo, con lo cual es posible mostrar formas con distorsión geométrica reducida. Se programó un algoritmo para realizar lecturas en el archivo que contiene la imagen, la información capturada se almacena temporalmente y al término de cada acceso a disco se despliega en pantalla el renglón respectivo. Esto se ejecuta repetidamente hasta llenar la pantalla con la imágen. A continuación se muestra, como ejemplo, el listado del programa referido.

```
/* ponima.c
   el programa lee un archivo y despliega una imagen en pantalla,
   programo: Alfonso Leon,    Abril / 1990  */

#define REN 200
#define COL 320
#define TAMANO 0x10
int x, y;
main ()
  {
    struct
      {
        int color;
      } tacr;
    regs.x.ax = 0x13;
    int86 (TAMANO, &regs, &regs);
    for (x=0; x<COL; x++)
      for (y=0; y<REN; y++)
        {
          fread (&tacr, sizeof(tacr), 1, tablero);
          regs.x.dx = x;
          regs.x.cx = y;
          regs.x.ax = tacr.color;
          int86 (TAMANO, &regs, &regs);
        }
  }
```

2.4 Paquete de graficación propio versus comercial.

Las rutinas programadas para trazar puntos, rectas y círculos, se integraron en un archivo relocalizable, el cual se liga a los programas de aplicación durante el proceso de compilación. Para graficar cada instrumento se desarrollaron programas específicos que utilizan el archivo anterior a manera de paquete de graficación. Las rutinas básicas generadas se pueden utilizar, además, para otro tipo de aplicaciones en las que se deseen realizar gráficas de tamaños y colores deseados.

El programa que contiene las funciones básicas de graficación permite:

1) elegir el tamaño de los instrumentos,
2) colorear los marcos que lo componen,
3) definir avisos de alarmas,
4) cambiar el color del fondo de la pantalla,
5) desplegar letras, números y caracteres especiales diferentes a los proporcionados por el modo de graficación,
6) generar la animación requerida en tiempo real para cada instrumento.

En el caso de optar por un programa comercial para graficación, se enfrentaría al problema de no existir comercialmente programas exclusivos para graficación de instrumentos aeronáuticos. Es posible contratar su desarrollo, pero deben solicitarse como un producto especial con el consiguiente costo elevado. Por lo tanto uno de los objetivos del presente trabajo es economizar y optimizar éstos recursos; al aportar un programa de graficación básica, que está constituído por el diseño de los instrumentos y de su animación asociada, no se resolvería en forma sencilla. Desde los anteriores puntos de vista el uso de programas comerciales comparados con el desarrollo de programación especifica no ofrece ventajas claras, y si serias limitaciones para continuar el desarrollo del mismo.

2.5 Modularidad para uso del programa de graficación.

El lenguaje C permite crear módulos o funciones particulares, que al ser empleadas por un programa o módulo principal facilitan el desarrollo de subprogramas sencillos, que en base a cada nivel se va incrementando su complejidad. Con características ventajosas en cuanto a estructura y facilidad de mantenimiento. La presente tesis utilizó esta técnica durante la integración de la interfaz hombre-máquina para la aeronave de control remoto.

En el apéndice D se observa la colección completa de los programas realizados para este trabajo. A continuación se describe, a manera de ejemplo, con separaciones por niveles de la estructura de acceso a la ejecución de los programas, una de las funciones desarrolladas:

1er. Nivel	2o. nivel	3er. nivel	4o. nivel
mha	hz	arc	pto
		cua	pto
		lre	pto
		rll	pto
		lim	pto
		pos	

El ejemplo define los pasos empleados para graficar y animar el instrumento denominado horizonte artificial. El primer nivel indica el nombre de la función, en este caso, el movimiento del horizonte artificial (MHA); en el segundo nivel define el nombre del instrumento por trazar, en

este caso, horizonte artificial (HZ); el tercer nivel indica las rutinas básicas de graficación empleadas para el trazo del instrumento, en el ejemplo el HZ utiliza, arcos (ARC), cuadros (CUA), líneas rectas (LRE), relleno de cielo y tierra (RLL), líneas de giro (LIN) y nombre del instrumento (POS); por último, el cuarto nivel define los primitivos de graficación, en este caso, la función punto (PTO), encargada de enviar puntos de colores específicos en las coordenadas requeridas. De forma similar se despliegan en pantalla el resto de los instrumentos, con las variantes necesarias de color, forma y localización dentro del monitor.

El programa principal de despliegue del instrumental de la aeronave está dividido en las subfunciones: HZ, CU, AA, RA, TE y ACR3. Las primeras cinco grafican los instrumentos, el nombre asociado de cada uno y la lectura inicial de datos. La subfunción ACR3 toma los bloques de telemetría, actualiza los datos en la pantalla, ejecuta la animación requerida y, en su caso, envía alarmas para señalar anomalías.

Para ampliar la forma adoptada para hacer gráficas en una pantalla, se describen en el siguiente capítulo los algoritmos principales para el trazo de figuras geométricas. Cómo funcionan los algoritmos de Bresenham y basándonos en ellos se desarrolló en forma particular uno para la brújula.

Cap. III. Desarrollo computarizado de instrumentos aeronáuticos.

3.1 Introducción.

En este capítulo se describe el trabajo de programación realizado. Se explican las diferentes técnicas de graficación empleadas para desarrollar la interfaz hombre-máquina para la aeronave y el cómo cumplen con la funcionalidad adecuada para el piloto. Dentro de los algoritmos descritos en esta sección, se encuentran aquellos empleados para desplegar líneas, círculos y la brújula.

Se describe el proceso de diseño de cada uno de los instrumentos de la aeronave y se comentan aspectos sobre la ubicación que ocupa el instrumental en la pantalla para comodidad y rápida localización por el piloto. Finalmente, se explica el diseño, uso de íconos, caracteres especiales, así como la generación de avisos de alerta para orientar al piloto en situaciones de emergencia.

3.2 Técnicas de graficación.

Las técnicas de graficación mencionadas en el capítulo anterior fueron desarrolladas en la década de los 60s[9], en tanto que el empleo y generación de la imágen en pantalla es una aplicación y mejora de la idea inicial a través de aportar animación y funcionalidad vanguardista. A continuación se explican los algoritmos desarrollados sin enfocarnos en este capítulo a la descripción de los mismos.

Los dispositivos digitales trazan líneas graficando puntos entre los extremos de una recta. Como los puntos en la pantalla son elementos discretos de forma cuadrada y tamaño constante, cuando se traza una línea, esta sólo aproxima líneas ideales entre los puntos extremos especificados.

La calidad de la línea trazada puede mejorar ya sea mediante el uso de sistemas de alta resolución, o bien, aplicando técnicas específicas para reducir la distorsión entre puntos.

El algoritmo de Bresenham inicia en uno de los extremos de la línea y se mueve horizontalmente hasta alcanzar el otro extremo. Durante la graficación, la línea permanece en el mismo renglón hasta que la inclinación por representar implique un cambio de dirección, lo cual se logra mediante un cambio de renglón, hacia arriba o abajo, según el caso. El problema básico consiste en determinar el momento en que se debe efectuar el cambio de renglón, ver figura 5. El programa para trazo de líneas puede verse en el apéndice E.

A continuación se presentan los pasos del algoritmo de Bresenham para el trazo de líneas, suponiendo que se conocen los puntos extremos $P_1(x_1,y_1)$ y $P_2(x_2,y_2)$.

1) Se almacenan en memoria las coordenadas de los puntos extremos P_1 y P_2.

2) El primer punto que se seleccionará para desplegarse es el punto extremo izquierdo $P_1(x_1,y_1)$.

3) Se calcula $Dx = x_2-x_1$, $Dy = y_2-y_1$ y $p_1 = 2Dy - Dx$. Si $p_1 = 0$, el siguiente punto por desplegar es (x_1+1, y_1). En caso contrario, el siguiente punto es (x_1+1, y_1+1).

4) Se continúa incrementando la coordenada x en unidades. En la posición x_i+1 la coordenada y_{i+1} que se seleccionará será y_i, o bien y_{i+1}, dependiendo de que $p_i<0$ o $p_i>=0$. Los cálculos de cada parámetro P dependen del cálculo anterior. Si $p_i<0$, la forma del siguente parámetro será $p_{i+1} = p_i + 2Dy$, pero si $p_i>=0$, el siguiente parámetro es $p_{i+1} = p_i + 2(Dy - Dx)$. Por lo tanto, si $p_{i+1}<0$, la siguiente coordenada y que se seleccionará es y_{i+1}. En caso contrario se selecciona $y_{i+1}+1$. (La coordenada y_{i+1} se determinó como y_i, o bien y_{i+1}, por medio del parámetro del paso 3).

5) Se repite la secuencia descrita en el paso 4 hasta que la coordenada x_i sea igual a la coordenada x_2.

Esta técnica es aparentemente difícil si se le da un seguimiento con papel y lápiz. La lógica empleada conlleva a trazar rectas en cualquier sentido y dirección, actualmente se usan algoritmos más sofisticados para hacer el trazo de líneas rectas la cual hace un despliegue de puntos de colores en diferente frecuencia de tono para darle a la vista el despliegue de una línea sin escalonamiento, este algoritmo no se incluyó ya que sólo para el modo de graficación seleccionado se pueden desplegar 16 colores y su frecuencia al ser alterada cambia muy drásticamente el color en la pantalla: El algoritmo sí funciona cuando se trabaja con el modo gráfico de 200 x 320 cuadros y 256 colores.

Bresenham también derivó un camino eficiente para graficar círculos, una vez más la clave consiste en generar desplazamientos crecientes y selectivos, uno a la vez, tomando el centro del círculo como origen y graficando inicialmente en la coordenada (0,r). Ver figura 6.

Los puntos por graficar, a lo largo de la trayectoria circular, se obtienen determinando el punto más próximo al centro de la circunferencia y que se encuentra a una distancia r del origen. El algoritmo considera una circunferencia con centro en el origen, luego se calculan los puntos de un arco de 45^0. Los cálculos anteriores se continúan hasta completar un arco de 180^0 y los puntos restantes se obtienen por simetría; se le conoce comunmente como hacerle un espejo al punto ya desplegado. En seguida se describen los pasos del algoritmo que genera círculos.

1) Se selecciona como primer punto de despliegue el que se encuentra en las coordenadas $(x_1, y_1) = (0, r)$.

2) Para obtener el segundo punto, se calcula $p_1 = 3 - 2r$ y si $p < 0$, el punto estará ubicado en la posición (x_1+1, y_1). De lo contrario estará ubicado en (x_1+1, y_1-1).

3) Se incrementa unitariamente la coordenada x y se calcula cada parámetro sucesivo p a partir del anterior. Si para el punto anterior se encontró que $p_1 < 0$, entonces

$$p_{i+1} = p_i + 4\,x_i + 6$$

de lo contrario para p >= 0,

$$p_{i+1} = p_i + 4\,(x_i - y_i) + 10$$

si $p_{i+1} < 0$, el siguiente punto seleccionado es $(x_i+2,\ y_{i+1})$, de lo contrario, el siguiente punto es $(x_i+2,\ y_{i+1} - 1)$. La coordenada y es $y_{i+1} = y_i$ si $p_i < 0$ o bien $y_{i+1} = y_i - 1$, si p >= 0.

4) Se repite el paso 3 hasta llegar a igualar la coordenada x con y.

Con este par de algoritmos descritos se generarán la mayoría de los instrumentos graficados en la pantalla incluyendo a los íconos y caracteres de información general. El instrumento al que se le hizo una programación extra es el indicador de rumbo en vuelo, brújula. Este instrumento consiste de un par de círculos entre los cuales se trazan líneas con una separación de 10 grados una de otra; para lograr ésto se emplean las ecuaciones de coordenadas polares que se describen a continuación. Un punto p se puede localizar por medio de coordenadas rectangulares (x, y), por coordenadas polares (r, h), donde h es el ángulo, las ecuaciones de transformación son:

$$x = r \cos h$$

$$y = r \operatorname{sen} h$$

para rectangulares

$$r = (x^2+y^2)^{1/2}$$

$$h = \tan^{-1}(y/x)$$

para polares

Al inicio del trazo de este instrumento se da un par de puntos extremos para graficar una línea, al termino de esta se calcula el cambio de 10^0 por medio de las coordenadas polares descritas. Se traza otra línea con los cambios calculados; para el siguiente cambio, el cálculo se

realizará considerando una inclinación de 10^0 mayor que el anteror para cada punto extremo y se trazará una nueva línea. Esto se realizará en forma continua hasta completar los 360^0 de la circunferencia. Los cálculos pueden ser obtenidos al ser ejecutado el algoritmo siguiente:

1) Se dan las coordenadas de los puntos extremos de la línea inicial, ver figura 7.
2) Se accesa a la subfunción de líneas para hacer el trazo.
3) Se ejecuta la iteración mientras i esté contenida en el intervalo $0 < i < 36$.

   3.1) Se calcula $j = i*p/18$ para obtener el ángulo de la próxima línea a trazar.

   3.2) Se calcula el seno y coseno de la variable j.

   3.3) Se obtienen los valores x_1, x_2, y_1 y y_2, con las formulas para cálculo de coordenadas polares siguientes:

   $$x_{i+1} = x_0+(x_i-x_0) * \cos j - (y_i-y_0) * \text{sen } j,$$

   $$y_{i+1} = y_0+(y_i-y_0) * \cos j + (x_i-x_0) * \text{sen } j.$$

   obteniéndose así el siguiente par de puntos que representan la próxima línea a graficar.

   3.4) Se grafica la nueva línea con las coordenadas obtenidas y se incrementa el valor de la variable i.

4) Se ejecuta el paso 3 hasta que el valor de i sea mayor al límite, en cuyo caso el algoritmo termina su ejecución.

Con la programación de este último algoritmo se tiene concluido el despliegue de los instrumentos de la aeronave en la pantalla. Así podremos describir todos los instrumentos a detalle y la coloración aplicada para la mejor localización dentro del tablero.

3.3 Instrumentos creados.

En el capítulo 1, sección 1.4, se describieron someramente los distintos instrumentos que se utilizan para supervisar el estado de vuelo de una aeronave. A continuación se describen los diseños de cada uno de los instrumentos, así como los detalles de programación elaborados para el proyecto de la aeronave de control remoto. Para mayor claridad de estos se puede ver la figura 8, la cual contiene todo el instrumental.

Horizonte artificial. Este instrumento se divide en dos partes, la mitad superior representa el cielo y la inferior la tierra, la primer parte aparece en azul en tanto que la segunda utiliza fondo naranja dando una impresión natural. Al centro del instrumento se muestran líneas horizontales que representan ángulos de inclinación, ascendente o descendente, del avión, los cuales varían 30^0 hacia arriba y 30^0 hacia abajo. En el semicírculo superior del instrumento se tienen los indicadores de derrape y viraje. De esta forma, el horizonte artificial informa al piloto sobre la intensidad de cambio en alabeo, cabeceo, viraje y derrape experimentados por el avión.

Con el objeto de facilitar la comprensión de las variables anteriores, dentro del instrumento se grafica continuamente la traza de un avión, cuya posición especifica el nivel de cabeceo, así como la magnitud del alabeo. La graficación del avión se realiza en tiempo real de acuerdo a los datos recibidos en los bloques de telemetría, los cuales contienen la información de cada uno de los sensores colocados dentro del avión. Debido a que todavía no se encuentra terminada la instrumentación física, la prueba de funcionamiento, captura y despliegue de datos de telemetría se simuló con la ayuda de otra computadora, como se explicará con mayor detalle en el capítulo 4.

Brújula. Para este instrumento se traza un ícono similar a la aeronave. Para indicar la dirección de avance, una parte del algoritmo calcula la rotación de los ejes cardinales de acuerdo a los datos de telemetría. La silueta del avión permanece siempre estática y sólo los cursores, ubicados en el contorno del ícono, son móviles.

Además de la gráfica anterior, en el semicírculo superior se presenta el dato de la dirección de vuelo en forma digital, representado con tres números. En investigación sobre integración hombre-máquina, se ha encontrado que la presentación híbrida digital-analógica es ventajosa para los operadores, porque la información de un modelo complementa la del otro.

Altímetro e indicadores de velocidad. Los instrumentos de altitud, velocidad aérea y velocidad vertical se grafican en pantalla con formato similar. Estos instrumentos se componen de tres partes: la primera, ubicada en la parte central y de forma rectangular, dentro de la cual se despliegan las lecturas en forma digital; la segunda, compuesta por líneas dinámicas que simulan el movimiento de una cinta métrica, y en cuya parte central se localiza la lectura instantánea ofrecida por el instrumento; la tercera, ubicada en la parte inferior, de forma rectangular, en donde se muestran alarmas visuales con efectos de color y sonido cuando las lecturas sobrepasan umbrales de tolerancia preestablecidos.

Nivel de combustible. Este instrumento consiste en una barra de color verde que se reduce de acuerdo al gasto de combustible. La parte inferior tiene un par de franjas rojas que indican la porción de reserva de combustible, en el caso de la aeronave de control remoto es de 3 litros. En la parte superior de la gráfica se muestra el nombre del instrumento.

Encendido del motor. Informa al piloto el estado de trabajo de motor. A través del despliegue de colores se muestran estados de funcionamiento normal (color verde), motor apagado (rojo) con alarama audible si ocurre en vuelo y amarillo si existen problemas de operación debidos a las alteraciones de la mezcla aire-combustible.

Los demás indicadores se representan por medio de íconos, letras y números que muestran las lecturas de los sensores en tiempo real. Entre éstos tenemos: tacómetro (rpm), temperatura de cabeza de cilindro, temperatura de gases de escape, temperatura del aire en la entrada del carburador, voltaje de batería, porcentaje de potencia del motor, temperaturas de diversos puntos internos del avión y del medio ambiente,

presión barométrica, ángulo de trabajo de las superficies de control, y finalmente las coordenadas del avión provistas por el sistema de posicionamiento global (GPS).

3.4 Distribución de instrumentos en pantalla.

En cuanto a la distribución de los instrumentos en tableros aeronáuticos, se ha normalizado un grupo central llamado "T básica", en la que aparece el indicador de horizonte artificial como instrumento principal en el centro de la "T", rodeado de los instrumentos de apoyo para el control de la aeronave, de la siguente forma: en orden del centro a la izquirda, el altímetro y el velocímetro; del centro a la derecha el combustible y los intrumentos de motor; en la parte baja, la brújula, a la izquierda de ella, el indicador de velocidad vertical y temperatura del medio ambiente; a la derecha de la brujula se encuentra el sistema de posicionamiento global.

El diseño del tablero de instrumentos para la aeronave de control remoto se basa en el principio anterior, aunque con ligeros cambios, basados éstos en recomendaciones hechas por ingenieros aeronáuticos.

En el proyecto aeronave de control remoto se utiliza, como ya se mencionó en el capítulo 1, una pantalla de computadora para representar los diversos instrumentos aeronáuticos de la nave. La distribución de los distintos indicadores en el monitor de la PC es como sigue: al centro, en la parte superior, se despliega el horizonte artificial; en la parte inferior de este, se tiene la brújula; a la izquierda de ellos, en el orden de derecha a izquierda, el altímetro; bajo este los metros sobre el nivel del terreno y la presión barométrica; velocidad aérea junto con la posición actual de las aletas; debajo de todos ellos se muestra la velocidad vertical y bajo este la temperatura del medio ambiente. En la parte derecha del horizonte artificial se despliega, de izquierda a derecha, el indicador de combustible; el foco de motor encendido; y hacia abajo, las revoluciones por minuto, temperatura de las cabezas de los cilindros, la temperatura del gas de escape, temperatura del combustible en el carburador, el voltímetro y el porcentaje de la potencia en el motor; debajo del indicador de

combustible se indican las coordenadas GPS del avión respecto a la estación terrena. Para un mejor entendimiento de este orde vease la figura 8, la cual muestra el nombre de cada instrumento.

3.5 Elaboración de iconos y caracteres especiales.

El desarrollo de la interfaz hombre-máquina para la aeronave contiene detalles de graficación que permiten de forma amigable y sencilla mostrar al piloto el estado de las diferentes variables de la aeronave. En especial, en el caso de indicadores auxiliares del aeroplano, es más simple captar la idea del instrumento en sí a través de un dibujo que con un texto que lo describa. En seguida se detallan los diferentes gráficos o iconos diseñados para complementar la representación de variables aeronáuticas.

Uno de ellos es el de la aeronave que aparece en el centro de la brújula que, como ya se dijo, sirve como referencia para indicar la dirección de vuelo de la aeronave. Esta figura se traza con líneas y un par de arcos, uno en cada extremo del dibujo.

La posición actual de los alerones, formados por un cuadro que simula el cuerpo del ala y una línea que representa el alerón. El ícono experimenta cambios en tiempo real de acuerdo al estado de vuelo del avión.

Los indicadores de: temperatura de las cabezas de los cilindros, representado con un cuadro que simula un pistón con líneas rectas que hacen las veces de los anillos y en la parte superior la bujía; temperatura del gas de escape, representado por un cuadro en forma de mofle, a cuyos lados se tienen cuadros pequeños que simulan la entrada y salida de los gases; temperatura del combustible en el carburador, representada por un corte longitudinal de un carburador y en medio de este, una línea con un círculo para simular la mariposa que controla la entrada de aire; al voltímetro se le representa con una batería, en cuyos polos se muestran los signos + y -.

3.6 Programación de alarmas.

En la sección 3.4 se describieron las características, el tipo y colocación del instrumental desarrollado para la aeronave. También se indicó que algunos de ellos muestran variables que, cuando rebasan límites conocidos, producen alarmas, las cuales deben indicarse en forma redundante al piloto. Los programas elaborados envían señales de alarma al tablero en forma de sonido y de color.

Los instrumentos que tienen alarmas programadas son: el altímetro, velocidad aérea, velocidad vertical, foco de encendido y nivel de combustible. Las alarmas utilizan indicadores en color verde para denotar operación dentro de valores límites tolerados; el color ambar se despliega cuando se opera en niveles de operación crítico; y por último, el rojo indica que se ha rebasado un valor límite establecido.

Los valores límites para instrumentos con alarmas programadas son:

Altímetro (mts sobre el nivel del terreno):

de 301 a adelante.	verde.
de 101 a 300 mts.	ambar y sonido.
de 0 a 100 mts.	rojo y sonido.

Velocidad aérea (Km / hora):

de 105 a 225 km/hr	verde
de 95 a 105 km/hr	ambar
o de 210 a 225 km/hr	
menor a 95 km/hr	rojo y sonido
o mayor a 225 km/hr	

Velocidad vertical (m / seg):

de 0 a 500 m/seg.	verde.
de 501 a 1000 m/seg.	ambar y sonido.
de 1001 m/seg en adelante	rojo y sonido.

Foco que indica el estado del motor:

funcionamiento normal.	verde.
funcionamiento anormal.	ambar y sonido.
apagado.	rojo y sonido.

Nivel de combustible:

de 20 a 6 lts.	verde.
de 5 a 3 lts.	ambar y sonido.
de 3 a 0 lts.	rojo y sonido.

CAPITULO IV. Simulación de la adquisición de datos para instrumentos.

4.1 Introducción.

Como capítulo final, describiremos la metodología empleada para alimentar información simulada a los programas elaborados. La prueba anterior se utilizó para comprobar funcionamiento y velocidad de respuesta durante el despliege de datos.

En vista de que la instrumentación física se encuentra en desarrollo, la captación de bloques de telemetría no se pudo realizar con el modelo real, sin embargo, el mismo tipo de pruebas se puede efectuar en forma eficiente y rápida, simulando la llegada de información telemétrica. Describiremos el formato de los bloques de telemetría enviados por el avión y la forma en que los programas elaborados accesan dicha información, para posteriormente efectuar la actualización de lecturas y su correspondiente animación en cada uno de los instrumentos mostrados al piloto.

También se menciona el procedimiento utilizado para generar los valores telemétricos simulados, el almacenamiento de datos en memoria y el proceso de recuperación. En cuanto a la representación de los datos capturados, esta se puede efectuar ya sea actualizando las lecturas de los instrumentos, o bien, mediante una tabulación.

Finalmente se describe un caso de simulación, el cual se utilizó para integrar un programa de demostración. En la siguiente y última sección se ofrece una serie de conclusiones acerca del trabajo elaborado y se hacen recomendaciones sobre posibles actualizaciones e ideas para ampliar el campo de aplicación del trabajo aquí presentado.

4.2 Creación de bases de datos con parámetros de vuelo.

Se desarrolló un programa de lectura de datos desde el teclado de la computadora, con el propósito de crear una base de datos que integrara las diferentes variables de la aeronave y que representaran, a manera de simulación, los bloques telemétricos. Al enviar los datos de los bloques

hacia los instrumentos, como se explicó en el capítulo 3, se produce la animación en el tablero.

El programa de lectura de datos es interactivo con el usuario y solicita los datos a través de preguntas. La lectura se realiza en el orden siguiente: altitud; velocidad aérea; velocidad vertical; ángulo de cabeceo; ángulo de alabeo; orientación de la nariz; posición de las aletas; nivel de combustible; revoluciones por minuto; temperaturas de cabeza de cilindro, gases de escape; motor encendido; voltímetro; potencia del motor; temperatura del medio ambiente; presión barométrica y finalmente, las coordenadas x, y, z, de posicionamiento global.

Los datos capturados se almacenan con un formato que hemos denominado bloque de telemetría (BT). Para agilizar el proceso de lectura iterada de datos, la información capturada desde el teclado se almacena temporalmente en memoria RAM, y al final de la sesión se efectúa el respaldo global. En seguida se muestra el formato BT de lectura de datos y en la figura 9 se puede ver el formato que aparece en pantalla.

```
struct
  {
    int   alt,vel,ad,acab,aala,ona,ale,nc,rpm,tac,tcc,tge,
          vol,pot,tma;
    float pb;
    char  m;
    int x,y,z;
  } bt;
```

Al finalizar el almacenamiento de datos el programa puede listar la información recibida, la cual se despliega desde el registro uno hasta el veinte y si se desea visualizar más datos sólo se presiona la tecla s y se listarán los siguientes veinte datos, y así sucesivamente hasta ver toda la información dentro del archivo. Si al finalizar un despliegue el usuario desea abandonar la sesión, basta con presionar la tecla n, con lo cual termina la ejecución del programa.

4.3 Proceso de recuperación de parámetros y opciones de representación de la primer técnica de simulación.

Como se indicó en la sección anterior los bloques de telemetría simulados tienen el formato BT, y con ellos se integra la información necesaria para simular la animación de variables aeronáuticas. La base de datos se accesa continuamente para actualizar la información de la pantalla de instrumentos; con ello, verificamos el funcionamiento del paquete de programas elaborado.

De la misma forma en que se leen bloques y se envían éstos hacia los instrumentos, el programa ofrece la opción de desplegar la información leída en forma tabular.

4.4 Descripción de la segunda técnica de simulación.

Otra técnica utilizada para adquirir bloques de telemetría simulados se basa en el trabajo sincronizado de dos computadoras, que se interconectaron a través de puertos tipo serie. El procedimiento se detalla a continuación: la computadora que despliega instrumentos en tiempo real se comporta como una unidad receptora de datos, en esta, la información leída se procesa y almacena permanentemente en disco para su posterior análisis; la computadora utilizada para transferir información telemétrica se comporta como una unidad transmisora de datos. Los datos transmitidos los toma de una base de datos previamente generada.

La técnica de simulación anterior constituye un tipo de prueba muy apegada al funcionamiento real mediante equipo de comunicaciones por microondas. Las pruebas elaboradas hasta el momento han sido totalmente exitosas, por lo que el presente trabajo, además de resolver el problema de representación de parámetros, también contiene la primera versión, básica, de protocolo de comunicaciones.

4.5 Creación de un programa de demostración.

La segunda técnica de simulación, permitió evaluar la funcionalidad de los programas. Para propósitos de prueba se consideró como problema secundario el que los datos telemétricos tuviesen o no relación con estados reales de vuelo de la aeronave. Sin embargo, para desarrollar un programa de demostración, fue necesario generar una base de datos con información apegada al funcionamiento real de la aeronave.

Los datos generados simulan que la aeronave efectúa el siguiente recorrido: La aeronave parte de un punto p de la pista, enseguida gana velocidad y despega, recorre cierta distancia e inicia un giro a la izquierda, el cual se continúa hasta describir una semi elipse, posteriormente se inicia el aterrizaje disminuyendo velocidad y alineándose con la pista, descendiendo toca el suelo y frena hasta llegar a un punto q, terminando así la demostración.

En el apéndice E se incluyen los programas desarrollados, ACR3, así como las rutinas que efectúan el trabajo de simulación mencionado anteriormente.

Finalmente, en la siguiente sección se discuten los resultados obtenidos y se dan recomendaciones para futuras aplicaciones.

CONCLUSIONES Y RECOMENDACIONES.

I. Conclusiones.

Resumiendo el trabajo desarrollado para representar instrumentos aeronáuticos, se pueden señalar lo siguiente:

1) La programación por objetivos es eficiente para estructurar y facilitar la actualización de programas. Esta técnica permitió dividir el trabajo total en grupos de tareas particulares, traduciéndose ésto en la reducción de los tiempos asignados para el desarrollo de la programación.

2) El desarrollo de la paquetería con ayuda del lenguaje 'C' permitió generar programas ejecutables que contienen código eficiente, traduciéndose ésto en mejorías notables en cuanto a tiempos de ejecución, los cuales son particularmente importantes para aplicaciones de graficación y de control en tiempo real, como es el caso en instrumentación aeronáutica.

3) Se aplicaron principios matemáticos por medio de algoritmos para el despliegue de gráficos.

4) Para efectuar la programación de la graficación y animación en el monitor se emplean las llamadas de interrupción al BIOS (Basic Input Output System), para hacer mas eficiente la labor de cada programa.

5) La paquetería desarrollada constituye una herramienta de gran utilidad y de alto valor agregado para aplicaciones de supervisión y guía de aeronaves, sean estas tripuladas o controladas remotamente.

II. Recomendaciones.

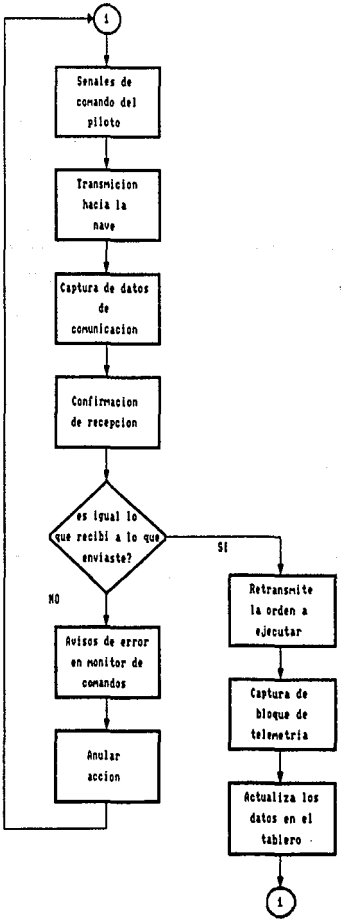
1) Los programas deben ejecutarse en equipo de cómputo que cuente con un monitor a color para percibir mejor los detalles de los instrumentos y para reconocer las alarmas visuales que emplean diferente color para indicar los distintos estados de trabajo.

2) La memoria RAM de la máquina no debe de ser inferior a los 640 Kbytes. Si esta es menor existe la posibilidad de conflicto en los programas.

3) Sería conveniente desarrollar un equipo electrónico autónomo que contuviera los programas desarrollados y controladores electrónicos dedicados, de tal forma que el producto generado pueda ser aplicado en instrumentación aeronáutica en general.

4) Los accesos de lectura de datos se obtienen de mejor calidad accesando al puerto serie con una interrupción al CPU sin contar con la ayuda del ROM-BIOS, ya que este último tiene deficiencias al accesar dicho puerto.

Apendice A. Diagrama de flujo de ejecucion de comandos enviados por la estacion terrena hacia la aeronave



APENDICE B. TABLAS DE INFORMACION COMPLEMENTARIA.

Tabla 1) Tipos de resolución de la tarjeta VGA.

Modo texto o gráfico:	Efecto en la tarjeta.
0 y 1	40 columnas alfanuméricas (tarjeta CGA y compatibles).
2 y 3	80 columnas alfanuméricos (tarjeta CGA y compatibles).
4 y 5	320 col. x 200 ren y 4 colores para gráficos limitados a 2 paletas (tarjeta CGA y compatibles).
6	640 col. x 200 ren. y 2 colores para gráficos, uno puede ser el negro (tarjeta CGA y compatibles).
7	alfanumérico monocromático (adaptador monocromático compatible, texto normal).
8 y Och	reservado para el uso de la tarjeta.
0dh	320 col. x 200 ren. y 16 colores.
0eh	640 col. x 200 ren. y 16 colores.
0fh	640 col. x 350 ren. gráficos moncromáticos.
10h	640 col. x 350 ren. y 4 colores y, para las tarjetas EGA, si tienen 64K de memoria se despliegan hasta 16 colores.
11h	640 col. x 480 ren. monocromático (sólo para la tarjeta VGA).
12h	640 col. x 480 col. y 16 colores (sólo para la tarjeta VGA).
13h	320 col. x 200 ren. y 256 colores (sólo para la tarjeta VGA).

TABLA 2) Tipos de interrupción de la ROM BIOS. Sintaxis INT 10H (cuando los siguientes parámetros se inicilizan a los valores requeridos).

Valor AH	Función	Entrada	Salida
		Control de interfaz del CRT	
AH = 0	Inicializa el modo de visualizar	AL = 0	40 × 25 B/W
		AL = 1	40 × 25 color
		AL = 2	80 × 25 B/W
		AL = 3	80 × 25 color
		AL = 4	320 × 200 color gráficas
		AL = 5	320 × 200 B/W gráficas
		AL = 6	640 × 200 B/W gráficas
		AL = 10	640 × 350 E.G.A. Gráficas
AH = 1	Inicializa tipo de cursor	CH =	Bits 4-0 principio de línea para cursor
		CL =	Bits 4-0 fin de línea para cursor
AH = 2	Inicializa posición de cursor	DH =	Fila
		DL =	Columna
		BH =	Número página de pantalla
AH = 3	Lee posición de cursor (valores hasta ejecución)	DH =	Fila
		DL =	Columna
		CH =	Modo cursor
		CL =	Modo cursor
		BH =	Número página de pantalla
AH = 4	Obtiene posición lapiz luminoso (valores hasta ejecución)	AH = 0	Conmutación no abajo/disparo
		AH = 1	Respuestas válidas, sigue:
		DH =	Fila
		DL =	Columna
		CH =	Grafo líneas (0-199)
		BX =	Columna gráfica (0-319,639)
AH = 5	Inicializa página pantalla activa	AL =	Valor nueva página Modos 0 y 1 (0-7) Modos 2 y 3 (0-3)
AH = 6	Enrolla página activa hacia arriba	AL =	Número de líneas; 0 para pantalla entera
		CH =	Fila, esquina superior izquierda
		CL =	Columna, esquina superior izquierda
		DH =	Fila, esquina inferior derecha
		DL =	Columna, esquina inferior derecha
		BH =	Atributo a ser usado
AH = 7	Enrolla página activa hacia abajo	AL =	Número de líneas; 0 para pantalla entera
		CH =	Fila, esquina superior izquierda
		CL =	Columna, esquina superior izquierda
		DH =	Fila, esquina inferior derecha
		DL =	Columna, esquina inferior derecha
		BH =	Atributo a usar
		Manipulación caracteres	
AH = 8	Lee atributo/carácter en posición cursor	BH =	Página pantalla
		AL =	Carácter leído
		AH =	Atributo de carácter
AH = 9	Escribe atributo/ carácter en posición cursor	BH =	Página pantalla
		CX =	Cuenta de caracteres a escribir
		AL =	Carácter a escribir
		BL =	Atributo de carácter
AH = 10	Escribe carácter en posición cursor	BH =	Visualiza página
		CX =	Cuenta de caracteres a escribir
		AL =	Carácter a escribir
		Interfaz de gráficos	
AH = 11	Selecciona paleta colores	BH =	Paleta ID (0-127)
		BL =	Color para paleta ID 0—Fondo (0-15) 1—Paleta 0—Verde (1), rojo (2), amarillo (3) 1—Cyan (1), magenta (2), blanco
AH = 12	Dibuja punto en pantalla	DX =	Fila (0-199)
		CX =	Columna (0-319 639)
		AL =	Color de punto
AH = 13	Lee información punto	DX =	Fila (0-199)
		CX =	Columna (0-319,639)
		AL =	Valor de punto
		Salida teletipo ASCII	
AH = 14	Escribe a página activa	AL =	Carácter a escribir
		BL =	Color primer plano
AH = 15	Devuelve estado vídeo	AL =	Modo actual
		AH =	Número de columnas de pantalla
		BH =	Página pantalla actual
AH = 16	Reservado		
AH = 17	Reservado		
AH = 18	Reservado		
AH = 19	Escribe cadena	ES:BP =	Señala a cadena
		CX =	Longitud de cadena
		DX =	Posición de cursor para comienzo
		BH =	Número de página
		AL = 0	BL = atributo Cursor no transferido
		AL = 1	BL = atributo Cursor es transferido
		AL = 2	Cursor no transferido
		AL = 3	Cursor es transferido

TABLA 2.1) Sintaxis INT 1AH.

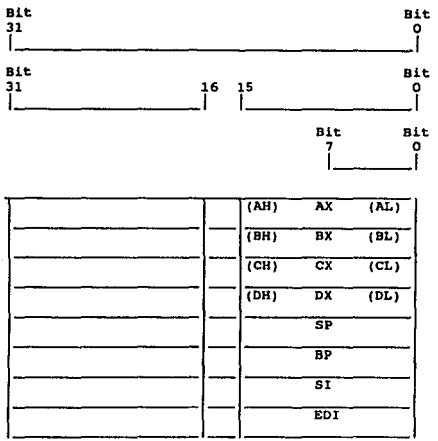
Valor AH	Función	Entrada	Salida
AH = 1	Espera y visualiza carácter de teclado con comprobación CTRL-BREAK	=	AL—Carácter introducido
AH = 2	Visualiza carácter con comprobación CTRL-BREAK	DL =	Carácter a visualizar
AH = 3	Entrada asincrona de carácter		AL—Carácter introducido
AH = 4	Salida asincrona de carácter	DL =	Carácter a enviar
AH = 5	Carácter a escribir	DL =	Carácter a escribir
AH = 6	Entrada carácter de teclado	DL = 0FFH	Carácter introducido, 0 si no
AH = 7	Espera para carácter de teclado (no visualiza)		AL—Carácter introducido
AH = 8	Espera para carácter de teclado (no visualiza—comprueba CTRL BREAK)		AL—Carácter introducido
AH = 9	Visualiza cadena	DS:DX =	Dirección de cadena. Debe finalizar con centinela $
AH = A	Cadena teclado a buffer	DS:DX =	Dirección de buffer. Primer byte = tamaño, segundo byte = número de caracteres leidos
AH = B	Status entrada de teclado		AL—No carácter = 0FFH Carácter = 0
AH = C	Borra buffer teclado y llama función	AL =	1,6,7,8,0A—Número función
AH = D	Unidad implicita disco (reset)	Ninguna	Ninguna
AH = E	Unidad implicita disco (selecciona)	DL =	AL—Número de unidades 0—Unidad A 1—Unidad B, etc.
AH = 19	Codigo unidad (implicita)		AL—0—Unidad A 1—Unidad B, etc.
AH = 25	Interrupción (inicializa)	DS:DX =	Dirección de vector de interrupción
AH = 2A	Fecha (lee)	AL =	Número de interrupción CX—Año (80 a 99) DH—Mes (1 a 12) DL—Día (1 a 31)
AH = 2B	Fecha (inicializa)	CX:DX	Igual que antes AL—0 si válido OFF si no válido
AH = 2C	Hora (lee)		CH—Horas (0-23) CL—Minutos (0-59)
AH = 2D	Hora (inicializa)	CX:DX	Igual que antes AL—0 si válido OFF si no válido
AH = 2E	Verifica estado (inicializa)	DL = AL =	0 0, verifica off 1, verifica on
AH = 35	Dirección interrupción (lee)	AL =	Número de interrupción ES:BX—apunta a dirección vector
AH = 36	Disco espacio disponible	DL =	Unidad (0—implicita, 1—A, 2—B, etc.) AX—Sectores/apilamiento (FFFF si unidad-mal) BX—número apilamientos libres CX—Bytes por sector DX—Número total de apilamientos
AH = 39	Hace directorio	DS:DX =	Dirección de cadena para directorio
AH = 3A	Elimina directorio	DS:DX =	Dirección de cadena para directorio
AH = 3B	Cambia directorio	DS:DX =	Dirección de cadena para nuevo directorio
AH = 3C	Archivo (crea)	DS:DX =	Dirección de cadena para archivo AX—Retorno manipula archivo
		CX =	Atributo archivo
AH = 3D	Archivo (abre)	DS:DX =	Dirección de cadena para archivo
		AL =	0—Abre para leer 1—Abre para escribir 2—Abre para ambos AX—Retorno manipula archivo
AH = 3E	Manipula archivo (cierra)	BX =	Manipula archivo
AH = 3F	Archivo o dispositivo (lee)	BX = CX = DS:DX =	Manipula archivo Número de bytes a leer Dirección de buffer AX—Número de bytes a leer
AH = 40	Archivo o dispositivo (escribe)	BX = CX = DS:DX =	Manipula archivo Numero de bytes a escribir Dirección de datos a escribir AX—Numero de bytes escritos
AH = 41	Archivo (suprime)	DS:DX =	Direccion de archivo cadena
AH = 42	Archivo (pone atributo)	CX:DX AL = 0	Desplazamiento en bytes Puntero desplazado (CX:DX) bytes desde el comienzo del archivo
		AL = 1	Puntero a posición actual más desplazamiento
		AL = 2	Puntero a EOF más desplazamiento
AH = 43	Archivo (inicializa atributo)	DS:DX =	Direccion de archivo cadena SI AL = 0, atributo devuelto a CX SI AL = 1, inicializa archivo con atributo de CX
AH = 47	Directorio actual	DL =	Numero unidad (0—implicita, 1—unidad A, 2—unidad B)
		DS:SI =	Direccion buffer DS:SI—Devuelve dirección de cadena
AH = 54	Verifica estado	Ninguna	AL—0 si verifica off 1 si verifica on
AH = 56	Archivo (renombra)	DS:DX =	Dirección de cadena para información antigua
		ES:DI =	Direccion de cadena para información nueva

TABLA 2.2) Sintaxis INT 21H.

Función: Permite que la hora del sistema sea leída o inicializada.

Valor AH	Función	Entrada	Salida
AH = 0	Lee inicialización reloj actual		CX—Bytes superiores del reloj DX—Bytes inferiores del reloj AL—0, si temporizador no ha pasado 24 horas
AH = 1	Inicializa reloj actual	CX = DX =	Bytes superiores de reloj Bytes inferiores del reloj
AH = 2	Lee hora—reloj de tiempo real		CH—BCD horas CL—BCD minutos DH—BCD segundos
AH = 3	Inicializa hora—reloj de tiempo real	CH = CL = DH = DL =	BCD horas BCD minutos BCD segundos 1—Hora del día 0—Hora estándar
AH = 4	Lee fecha—reloj de tiempo real		CH—BCD siglo CL—BCD año CL—BCD año DH—BCD mes DL—BCD día
AH = 5	Inicializa fecha—reloj de tiempo real	CH = CL = DH = DL =	BCD siglo BCD año BCD mes BCD día
AH = 6	Inicializa alarma (hasta 23:59:59)	CH = CL = DH =	BCD horas BCD minutos BCD segundos
AH = 7	Reinicializa alarma		

Tabla 3) Registros de propósito general en el microprocesador 80386.



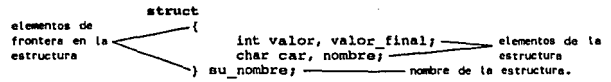
Los registros de propósito general soportan operandos de datos de 1, 8, 16 y 32 bits y campos de 1 a 32 bits. Para acceder a los 32 bits de un registro, todas las referencias deben comenzar con la letra 'E'[24].

APENDICE C. PROGRAMACION DE LAS ESTRUCTURAS Y UNIONES.

Las variables simples pueden alamcenar una pieza de información a la vez y los arreglos pueden alamcenar varias piezas de información del mismo tipo de datos. Estas dos formas de almacenamiento de datos pueden manejar una gran variedad de situaciones, pero frecuentemente necesitamos operar con diferentes tipos de datos como si fueran una unidad.

C provee un tipo especial de datos: la **estructura**. Una estructura se constituye por un grupo de datos, los cuales no necesitan ser del mismo tipo.

Con las estructuras se puede formar la base de estructuras complejas. El formato de declaración de una estructura se presenta a continuación:



Las uniones tienen la misma relación que la estructuras. Ambas se usan para agrupar variables diferentes. Las uniones permiten acceder datos de diferente tipo mediante el nombre de una sola variable. En seguida se muestran algunos ejemplos de estructuras agrupadas en una unión:

```
/* dos.h
   Define estructuras, uniones, macros, y funciones para interactuar
 con MS-DOS y la familia de microprocesadores Intel iAPX86.
 Copyright (c) Borland International 1987,1988. All Rights Reserved. */
...
struct WORDREGS {
unsigned int ax, bx, cx, dx, si, di, cflag, flags;
                };
struct BYTEREGS {
unsigned char al, ah, bl, bh, cl, ch, dl, dh;
                };
union REGS {
structWORDREGS x;
structBYTEREGS h;
        };
...
int _Cdecl int86(int intno, union REGS *inregs, union REGS *outregs);
int _Cdecl int86x(int intno, union REGS *inregs, union REGS *outregs,
 struct SREGS *segregs);
int _Cdecl intdos(union REGS *inregs, union REGS *outregs);
int _Cdecl intdosx(union REGS *inregs, union REGS *outregs, struct SREGS
*segregs);
...
```

La utilería dos.h nos es útil para hacer los accesos correspondientes hacia los microprocesadores, específicamente a la tarjeta de video VGA.

APENDICE D. NIVELES DE LLAMADAS A EJECUCUION DE LOS MODULOS DE GRAFICOS PARA EL DESPLIEGUE DEL TABLERO DE INSTRUMENTOS DE LA ACR.

Programa Principal ACR.C

1er. Nivel	2o. nivel	3er. nivel	4o. nivel	5o. nivel
acr	hz	arc	pto	
		cua	pto	
		lre	pto	
		rll	pto	
		lin	pto	
		pos		
		ax8	lre	pto
			pto	
			lin	pto
			cir	pto
	cu	mar	lre	pto
		cua	pto	
		lre	pto	
		pos		
		ax8	lre	pto
			lin	pto
			cir	pto
	aa	mar	lre	pto
		cua	pto	
		lre	pto	
		pos		
		ax8	lre	pto
			lin	pto
			cir	pto
		lin	pto	
	ra	bru	lin	pto
		mar	lre	pto
		lre	pto	
		lin	pto	
		arc	pto	
		pos		
	te	cua	pto	
		pos		
		lre	pto	
		pto		
		lin	pto	
		cir	pto	
	acr3	pos		
		rac	cua	pto
			son	
		alh	lre	pto
		lin	pto	
		dlc	pto	
		dcc	pto	
		gcc	bca	
			cir	pto
		gce	bca	

1er. Nivel	2o. nivel	3er. nivel	4o. nivel	5o. nivel
		lin	pto	
		bru	lin	pto
		gya	lre	pto
			cua	pto
		cir	pto	
		pna	cua	pto

Programa que pone en modo texto la pantalla de la computadora.

1er. Nivel	2o. nivel	3er. nivel	4o. nivel	5o. nivel
normal				

Programa que guarda datos de simulación de una misión.

1er. Nivel	2o. nivel	3er. nivel	4o. nivel	5o. nivel
acr4	pos			
	mar	pto		

Programa que lista datos de simulación de una misión.

1er. Nivel	2o. nivel	3er. nivel	4o. nivel	5o. nivel
acr5	pos			

Descripción de las actividades de cada programa.

acr	Programa principal de control de la ACR.
hz	Indicador del horizonte artificial.
arc	Dibuja un arco en la pantalla.
pto	Dibuja los puntos de la figura.
cua	Dibuja un cuadro de color en pantalla.
lre	Dibuja líneas rectas, verticales u horizontales.
rll	Rellena un figura.
lin	Dibuja una línea en cualquier dirección.
pos	Coloca al cursor en la posición solicictada.
ax8	Dibuja alfanuméricos de 8x8 puntos.
cir	Dibuja círculos en la pantalla.
ku	Indicador del combustible.
mar	Dibuja marcos en la pantalla.
aa	Indicadores de altitud, velocidad aérea y velocidad vertical.
ra	Indicador de la brújula.
bru	Dibuja las líneas de la brújula.
te	Indicadores de RPM's y temperaturas.
acr3	Animación del tablero.
rac	Alarmas para los instrumentos de altitud, velocidad aérea, y velocidad verical,
son	Hace sonar dos veces un pitido de la computadora.
alh	Animación de las líneas horizontales para la altitud, velocidad aérea y la velocidad vertical.
dlc	Dibuja una línea con los colores encontrados anteriormente,
dcc	Dibuja un círculo con los colores encontrados.
gcc	Guarda los colores del círculo previo para dibujar uno nuevo.
bca	Busca un color en la coordenada solicitada.
gce	Guarda los colores encontrados en las coordenada solicitada.
gya	Actualización del nivel de combustible.
pna	Potencia en el motor.

APENDICE E. LISTADO DE LOS PROGRAMAS DEL PROYECTO AERONAVE DE CONTROL REMOTO.

```
/* acr.c
   Aeronave de Control Remoto para Teledetección Multimisional (ACR),
   programó: Alfonso León,          Mayo / 1990
   revisión: Alfonso León,       Febrero / 1991                */
main ()
    hz ();      /* horizonte artificial */
    ku ();      /* nivel del combustible */
    aa ();      /* altitud y las velocidades */
    ra ();      /* RPM's y temperaturas */
    te ();      /* brújula */
                /* Animación del instrumental */
    acr3 ();
  } /* Fin del programa acr.c */
```

```
/* pto.c
   pone puntos en pantalla con una resolucion
   de 640col por 480ren y 16 colores,
   programó: Alfonso León,      Febrero / 1990 */
pto (xren, ycol, color)
int xren, ycol, color;
  { regs.x.ax = 0x0c + color;
    regs.x.dx = xren;
    regs.x.cx = ycol;
    int86 (MODOGRAF, &regs, &regs);
  } /* fin de la función pto.c */
```

```
/* cua.c
   Dibuja un cuadro con relleno en pantalla,
   programó: Alfonso León,     Abril / 1990 */
cua (xi, yi, xf, yf, color)
int xren, ycol, color;
int xi, yi, xf, yf;
  { for(xren=xi; xren<=xf; xren++)
      for(ycol=yi; ycol<=yf; ycol++)
        pto(xren, ycol, color);
  } /* Fin de la función cua.c */
```

```
/* hz.c
   Graficación del horizonte artificial, parte tierra
   y parte cielo, instrumento de navegación,
   programó: Alfonso León,                Marzo / 1990    */
hz ()
  { int i;
    for (i=108; i<=111; i++)
```

```
    { arc (130,330,i,1,7);   arc (130,330,i,2,7);   }
  cua ( 56,219,204,250,0);   cua ( 56,410,204,441,0);
  lre (130,249,411, 7,'h');
  cua ( 56,249,204,252,7);   cua ( 56,408,204,411,7);
  rll (100,330, 7, 0, 3);   rll (140,330, 7, 0, 6);
  lre (  8,320,340, 7,'h');
  lin ( 8,320, 18,330,7);   lin ( 8,340, 18,330,7);
  for (i=40; i<=220; i+=20) lre ( i,320,340, 0,'h');
  for (i=50; i<=210; i+=20) lre ( i,310,350, 0,'h');
  lre (130,252,408, 0,'h');
  lin ( 19,310,  9,308,10); lin ( 24,291, 15,288,10);
  lin ( 33,274, 24,269,10); lin ( 82,246, 77,237,10);
  lin ( 81,414, 76,423,10); lin ( 33,386, 24,391,10);
  lin ( 24,368, 15,371,10); lin ( 19,349,  9,351,10);
  pos (17,38); printf ("Horizonte");
  for (i=66; i<=186; i+=20)
    { ax8 ('0', i,301, 0); ax8 ('0', i,360, 0);
      if (i == 106) i += 20;   }
  ax8 ('3', 66,293, 0); ax8 ('6', 73,219, 7); ax8 ('3', 66,352, 0);
  ax8 ('0', 73,228, 7); ax8 ('2', 86,293, 0); ax8 ('3', 16,258, 7);
  ax8 ('2', 86,352, 0); ax8 ('0', 16,268, 7); ax8 ('1',106,293, 0);
  ax8 ('2',  6,271, 7); ax8 ('1',106,352, 0); ax8 ('0',  6,280, 7);
  ax8 ('1',146,293, 0); ax8 ('1',  0,293, 7); ax8 ('1',146,352, 0);
  ax8 ('0',  0,301, 7); ax8 ('2',166,293, 0); ax8 ('1',  0,350, 7);
  ax8 ('2',166,352, 0); ax8 ('0',  0,357, 7); ax8 ('3',186,293, 0);
  ax8 ('2',  6,367, 7); ax8 ('3',186,352, 0); ax8 ('0',  6,376, 7);
  ax8 ('3', 16,387, 7); ax8 ('0', 16,396, 7); ax8 ('6', 73,424, 7);
  ax8 ('0', 73,433, 7);
 } /* Fin de la función hz.c */
```

```
/* arc.c
   Dibuja un arco en la pantalla,
   programó: Alfonso León,    Septiembre / 1990 */
arc (xc, yc, radio,lado, color)
int xc, yc, color, radio, lado;
  { int p, x, y;
    x = 0; y = radio; p = 3 - 2 * radio;
    do { ponp ( x, y, xc, yc, lado, color);
         if (p < 0)   p += 4 * x + 16;
         else {   p += 4 * (x - y) + 10;   y--;   }
         x++;
         if (x == y)   ponp ( x, y, xc, yc, lado, color);
       } while (x < y);
  } /* Fin de la función arc.c */
ponp (x, y, xc, yc, lado, color)
int x, y;
  { switch (lado){
      case 1: pto(xc-x, yc+y, color); pto(xc-x, yc-y, color);
              pto(xc-y, yc+x, color); pto(xc-y, yc-x, color);
              break;
      case 2: pto(xc+x, yc+y, color); pto(xc+x, yc-y, color);
              pto(xc+y, yc+x, color); pto(xc+y, yc-x, color);
              break;
```

```
      case 3: pto(xc-y, yc+x, color); pto(xc+x, yc+y, color);
              pto(xc+y, yc+x, color); pto(xc-x, yc+y, color);
              break;
      case 4: pto(xc+y, yc-x, color); pto(xc-x, yc-y, color);
              pto(xc-y, yc-x, color); pto(xc+x, yc-y, color);
              break;   }
  } /* Fin de la sub-función ponp () */
```

```
/* pos.c
   coloca en coordenadas solicitadas un caracter invocado,
   programó: Alfonso León, Agosto / 1990 */
pos (rglón, clmna)
int  rglón, clmna;
  { regs.x.dx = (rglón - 1) * 256 + (clmna - 1);
    regs.x.ax = 2 * 256;
    regs.x.bx = 0x00;
    int86(MODOGRAF,&regs, &regs);
  } /* fin de la función pos.c */
```

```
/* lre.c
   traza líneas horizontales o líneas verticales,
   programó: Alfonso León, Abril    / 1990    */
lre (x, vi, vf, color, id)
int x, vi, vf, color;
char id;
  { int i, hi, hf, y;
    if (id == 'h')
      { if(vi > vf)
        { i  = vi; vi = vf; vf = i;   }
        for (i=vi; i<=vf; i++)   pto (x, i, color);   }
    else if (id == 'v')
      { hi = x; hf = vi; y  = vf;
        if(hi > hf)
          { i  = hi; hi = hf; hf = i;   }
        for (i=hi; i<=hf; i++)   pto (i, y, color);   }
  } /* fin de la función lre.c */
```

```
/* mar.c
   dibuja una caja rectangular,
   programó: Alfonso León, Agosto / 1990 */
mar (xi, yi, xf, yf, nl, color)
int xi, yi, xf, yf, nl, color;
  { int i;
    for (i=0; i<nl; i++)
      { lre (xi+i, yi, yf, color,'h');
        lre (xf-i, yi, yf, color,'h');
        lre (xi, xf, yi+i, color,'v');
        lre (xi, xf, yf-i, color,'v'); }  } /* fin de la función mar.c */
```

```
/* son.c
   hace sonar un # de veces a una chicharra,
   programó: Alfonso León, Febrero / 1991 */
son (num)
int num;
  { int veces;
    for (veces = 0; veces <= num; veces++)   printf ("7");
  } /* fin de la función son.c */
```

```
/* rll.c
   Esta función rellena una figura,
   programó: Alfonso León,  Febrero / 1991  */
rll (cx, cy, cormar, corant, cornvo)
int cx, cy, cormar, corant, cornvo;
  { int k, ccx, ccy, i;
    char cve = 'b';
    ccx = cx; ccy = cy;
    if ((k = busca (ccx, ccy, cormar)) == corant)
     pto (ccx, ccy, cornvo);
    do { if ((k = busca (ccx, ccy-1, cormar)) == corant)
         for (i=0;  k != cormar; i++)
           { pto (ccx, ccy--, cornvo);
             k = busca (ccx, ccy-1, cormar);   }
      ccy = cy;
      if ((k = busca (ccx, ccy+1, cormar)) == corant)
        for (i=0; k != cormar; i++)
        { pto (ccx, ccy++, cornvo);
          k = busca (ccx, ccy+1, cormar);   }
      ccy = cy;
      if ((k = busca (ccx-1, ccy, cormar)) == cormar)   cve = 'a';
      else if (cve == 'b')   ccx--;
      if (cve == 'a')
        { ccx = cx; cve = 'c';   }
      if ((k = busca (ccx+1, ccy, cormar)) == cormar)   cve = 'q';
      else if (cve == 'c')   ccx++;
    } while (cve != 'q');
  } /* fin de la función rll.c */
busca (x, y, cormar)
int x, y, cormar;
  { int color;
    regs.x.dx = x;
    regs.x.cx = y;
    regs.h.ah = 0x0d;
    int86 (MODOGRAF, &regs, &regs);
    color = regs.h.al;
    if (color != cormar)   return (color);
    else   return (cormar);
  } /* fin de la sub-función busca() */
```

```
/* lin.c
   traza líneas en pantalla,
   programó: Alfonso León,  Mayo / 1990 */
lin (ix, iy, x, y, color)
int ix, iy, x, y;
  { int sx, sy, dx, dy, qq, er;
    dx = x-ix; dy = y-iy;
    sx = 1; sy = 1;
    if (dx < 0) { sx = -1; dx = -dx;}
    if (dy < 0) { sy = -1; dy = -dy;}
    if (dy >= dx)
      { er = 2 * dx - dy;
        for (qq=0; qq<=dy; qq++)
         { pto (ix, iy, color);
           if (er >= 0)
             { er += 2 * dx - 2 * dy; iy += sy; ix += sx;    }
           else
             { er += 2*dx; iy += sy;    }   }    }
    else
      { er = 2 * dy - dx;
        for (qq=0; qq<=dx; qq++)
          { pto (ix, iy, color);
            if (er >= 0)
              { iy += sy; ix += sx; er += 2 * dy - 2 * dx;    }
            else
              { er += 2 * dy; ix += sx;    }   }    }
    if ((ix > x) | (iy > y))
      pto (ix, iy, color);
  } /* fin de la función lin.c */
```

```
/* pna.c
   Potencia en el motor,
   programó: Alfonso León,  Mayo / 1990 */
pna (act, ant)
int act, ant;
  { cua (261,628-ant,266,631-ant,0);
    cua (261,601,266,620,14);
    cua (261,621,266,631,4);
    cua (261,628-act,266,631-act,3);
    return(act);
  } /* fin de la función pna.c */
```

```
/* ax8.c
   Alfa-numéricos de 8x8 pixeles
   programó: Alfonso León,   Octubre / 1990 */
ax8 (car, x, y, color)
int x, y, color;
char car;
  { switch (car) {
      case 'a': lre ( x+3, y+2, y+4, color,'h');
                lre ( x+4, x+6, y+1, color,'v');
```

```
            lre ( x+4, x+6, y+5, color,'v');
            lre ( x+7, y+2, y+4, color,'h');
            lre ( x+3, x+7, y+6, color,'v'); break;
case 'b':   lre ( x  , x+7, y+1, color,'v');
            lre ( x+3, y+3, y+5, color,'h');
            lre ( x+4, x+6, y+2, color,'v');
            lre ( x+4, x+6, y+6, color,'v');
            lre ( x+7, y+3, y+5, color,'h'); break;
case 'c':   lre ( x+3, y+2, y+4, color,'h');
            lre ( x+4, x+6, y+1, color,'v');
            lre ( x+7, y+2, y+4, color,'h');
            pto ( x+4, y+5, color);                 break;
case 'd':   lre ( x  , x+7, y+6, color,'v');
            lre ( x+3, y+2, y+4, color,'h');
            lre ( x+4, x+6, y+1, color,'v');
            lre ( x+4, x+6, y+5, color,'v');
            lre ( x+7, y+2, y+4, color,'h'); break;
case 'e':   lre ( x+3, y+2, y+4, color,'h');
            lre ( x+5, y+1, y+5, color,'h');
            lre ( x+7, y+2, y+4, color,'h');
            pto ( x+4, y+1, color); pto ( x+4, y+5, color);
            pto ( x+6, y+1, color);          break;
case 'f':   lin ( x, y+3, x+1, y+2, color);
            lin ( x, y+4, x+1, y+5, color);
            lre ( x+2, x+7, y+2, color,'v');
            lre ( x+4, y+1, y+4, color,'h'); break;
case 'g':   lin ( x+1, y+6, x+2, y+5, color);
            lin ( x+2, y+3, x+3, y+2, color);
            lin ( x+4, y+6, x+5, y+5, color);
            lin ( x+6, y+6, x+7, y+5, color);
            pto ( x+2, y+4, color); pto ( x+3, y+3, color);
            pto ( x+4, y+2, color);
            lre ( x+5, y+3, y+4, color,'h');
            lre ( x+7, y+3, y+4, color,'h'); break;
case 'h':   lre ( x  , x+7, y+1, color,'v');
            lre ( x+4, x+7, y+5, color,'v');
            lre ( x+3, y+3, y+4, color,'h');
            pto ( x+4, y+2, color);                 break;
case 'i':   lre ( x+4, x+6, y+3, color,'v');
            lre ( x+7, y+2, y+4, color,'h');
            pto ( x+4, y+2, color);
            pto ( x+2, y+3, color); break;
case 'j':   lre ( x+4, x+5, y+5, color,'v');
            lin ( x+6, y+5, x+7, y+4, color);
            lin ( x+6, y+2, x+7, y+3, color);
            pto ( x+2, y+5, color);                 break;
case 'k':   lre (   x, x+7, y+2, color,'v');
            lin ( x+1, y+6, x+3, y+4, color);
            lin ( x+4, y+3, x+6, y+6, color); break;
case 'l':   lre ( x  , x+6, y+4, color,'v');
            lre ( x+7, y+3, y+5, color,'h'); break;
case 'm':   lre ( x+3, y+1, y+3, color,'h');
            lre ( x+4, x+7, y+2, color,'v');
            lre ( x+4, x+7, y+4, color,'v');
            lre ( x+4, x+7, y+6, color,'v');
```

```
            pto ( x+3, y+5, color);                 break;
case 'n':   lre ( x+3, y+1, y+4, color,'h');
            lre ( x+4, x+7, y+2, color,'v');
            lre ( x+4, x+7, y+5, color,'v'); break;
case 'ñ':   lre ( x+1, y+2, y+5, color,'h');
            lre ( x+3, y+3, y+4, color,'h');
            lre ( x+4, x+7, y+2, color,'v');
            lre ( x+4, x+7, y+5, color,'v'); break;
case 'o':   cir (x+5, y+5, 2, color);           break;
case 'p':   lre ( x+2, x+7, y+1, color,'v');
            lre ( x+2, y+3, y+5, color,'h');
            lre ( x+5, y+3, y+5, color,'h');
            lre ( x+3, x+4, y+2, color,'v');
            lre ( x+3, x+4, y+6, color,'v'); break;
case 'q':   lre ( x+3, x+4, y+1, color,'v');
            lre ( x+2, y+2, y+4, color,'h');
            lre ( x+4, y+2, y+4, color,'h');
            lre ( x+3, x+4, y+5, color,'v');
            lre ( x+2, x+7, y+6, color,'v'); break;
case 'r':   lre ( x+3, x+7, y+3, color,'v');
            lin ( x+3, y+5, x+4, y+3, color); break;
case 's':   lre ( x+3, y+3, y+5, color,'h');
            lre ( x+5, y+3, y+4, color,'h');
            pto (x+4,y+2,color); pto (x+6,y+5,color);
            lre ( x+7, y+2, y+4, color,'h'); break;
case 't':   lre ( x+1, x+6, y+3, color,'v');
            lre ( x+3, y+2, y+4, color,'h');
            pto ( x+7, y+4, color);             break;
case 'u':   lre ( x+3, x+6, y+2, color,'v');
            lre ( x+3, x+6, y+5, color,'v');
            lre ( x+7, y+3, y+4, color,'h');
            pto (x+7, y+6, color);              break;
case 'v':   lre ( x+3, x+6, y+2, color,'v');
            lre ( x+3, x+6, y+5, color,'v');
            lre ( x+7, y+3, y+4, color,'h'); break;
case 'w':   lin ( x+3, y+1, x+7, y+2, color);
            lin ( x+3, y+3, x+7, y+2, color);
            lin ( x+3, y+3, x+7, y+4, color);
            lin ( x+3, y+5, x+7, y+4, color); break;
case 'x':   lin ( x+3, y+2, x+6, y+6, color);
            lin ( x+3, y+6, x+6, y+2, color); break;
case 'y':   lre ( x+3, x+4, y+2, color,'v');
            lre ( x+3, x+4, y+5, color,'v');
            lre ( x+5, y+3, y+4, color,'h');
            lre ( x+7, y+2, y+4, color,'h');
            pto ( x+6, y+5, color);             break;
case 'z':   lre ( x+3, y+1, y+4, color,'h');
            lre ( x+7, y+2, y+5, color,'h');
            lin ( x+3, y+5, x+6, y+1, color); break;
case '0':   lre ( x  , y+3, y+4, color,'h');
            lre ( x+2, x+5, y+1, color,'v');
            lre ( x+2, x+5, y+6, color,'v');
            lre ( x+7, y+3, y+4, color,'h');
            pto ( x+1, y+2, color); pto ( x+1, y+5, color);
            pto ( x+6, y+2, color); pto ( x+6, y+5, color);
```

```
                lin ( x+3, y+4, x+4, y+3, color); break;
        case '1': lre ( x  , x+6, y+3, color,'v');
                pto ( x+2, y+2, color);
                lre ( x+7, y+2, y+4, color,'h'); break;
        case '2': lre ( x  , y+3, y+4, color,'h');
                lre ( x+1, x+2, y+2, color,'v');
                lre ( x+1, x+3, y+5, color,'v');
                lin ( x+4, y+4, x+6, y+2, color);
                lre ( x+7, y+2, y+5, color,'h'); break;
        case '3': lre ( x  , y+2, y+5, color,'h');
                lin ( x+1, y+5, x+3, y+3, color);
                pto ( x+4, y+4, color);
                lre ( x+5, x+6, y+5, color,'v');
                lre ( x+7, y+2, y+4, color,'h'); break;
        case '4': lre ( x  , x+7, y+5, color,'v');
                lin ( x+1, y+4, x+3, y+2, color);
                lre ( x+4, y+2, y+4, color,'h'); break;
        case '5': lre ( x  , y+2, y+5, color,'h');
                lre ( x+1, x+2, y+2, color,'v');
                lre ( x+3, y+3, y+4, color,'h');
                lre ( x+4, x+6, y+5, color,'v');
                lre ( x+7, y+2, y+4, color,'h'); break;
        case '6': lre ( x  , y+3, y+4, color,'h');
                pto ( x+1, y+5, color);
                lre ( x+1, x+6, y+2, color,'v');
                lre ( x+4, y+3, y+4, color,'h');
                lre ( x+5, x+6, y+5, color,'v');
                lre ( x+7, y+3, y+4, color,'h'); break;
        case '7': lre ( x  , y+2, y+6, color,'h');
                lin ( x+1, y+6, x+5, y+2, color);
                lre ( x+6, x+7, y+2, color,'v'); break;
        case '8': cir (x+2, y+4, 2, color); cir (x+5, y+4, 2, color);
                break;
        case '9': cir (x+2, y+4, 2, color); cir (x+5, y+4, 2, color);
                lre ( x+4, x+5, y+2, 0,'v'); break;
        case '/': lin ( x, y+6, x+6, y, color); break;  }
  } /* fin de la función ax8.c */
```

```
/* cir.c
   Dibuja un círculo en la pantalla
   programó: Alfonso León, Abril / 1990 */
cir (xc, yc, radio, color)
int xc, yc, color, radio;
  { int p, x, y;
    x = 0; y = radio; p = 3 - 2 * radio;
    do  { ponpunto ( x, y, color, xc, yc);
        if (p < 0) p += 4 * x + 16;
        else  { p += 4 * (x - y) + 10; y--;   }
        x++;
        if (x == y)   ponpunto ( x, y, color, xc, yc);
      } while (x < y);
  } /* Fin de la función cir.c */
ponpunto (x, y, color, xc, yc)
```

```
int x, y;
  {
    pto (xc+x, yc+y, color); pto (xc-x, yc+y, color);
    pto (xc+x, yc-y, color); pto (xc-x, yc-y, color);
    pto (xc+y, yc+x, color); pto (xc-y, yc+x, color);
    pto (xc+y, yc-x, color); pto (xc-y, yc-x, color);
  } /* Fin de la sub-funcion ponpunto () */
```

```
/* ku.c
   Graficación del indicador del combustible,
   instrumento de motor, una mezcla de aceite y gasolina,
   programó: Alfonso León,                Marzo / 1990
   revisión: Alfonso León,               Febrero / 1991      */
ku ()
  { int i;
   mar ( 32,431, 45,481,2,7); cua ( 34,433, 43,479,1);
   mar (242,431,255,481,2,7); cua (244,433,253,479,1);
   cua (213,440,241,470,4);   cua ( 46,450,240,460,10);
   for (i=48; i<=236; i+=9)   lre ( 1,452,458, 0,'h');
   for (i=58; i<=196; i+=45)  lre ( 1,452,458, 0,'h');
   pos ( 2,55); printf ("Combus");
   ax8 ('1', 35,440,14); ax8 ('t', 35,448,14); ax8 ('s', 35,456,14);
   ax8 ('2', 54,463, 7); ax8 ('0', 54,471, 7); ax8 ('1',100,463, 7);
   ax8 ('5',100,471, 7); ax8 ('1',145,463, 7); ax8 ('0',145,471, 7);
   ax8 ('5',190,463,7);
  } /* Fin de la función ku.c */
```

```
/* aa.c
   Graficación de los instrumentos de altitud, velocidad aérea y
   velocidad Vertical, instrumentos de navegación,
   programó: Alfonso León,                Marzo / 1990
   revisión: Alfonso León,               Febrero / 1991          */
aa ()
  { int i, j, k, l;
    /* - - - - - altitud - - - - - - */
    mar ( 32,164, 45,221,2,7); mar (147,164,160,221,2,7);
    mar (174,174,209,217,2,7); mar ( 78,142, 97,193,2,5);
    mar (222,182,238,217,2,7); mar (238,166,257,217,2,7);
    cua ( 34,166, 43,219,1); cua (149,166,158,219,10);
    lre ( 46,146,212, 7,'v'); lre ( 96,168,197,10,'h');
    for (i=50; i<=140; i+=10) lre ( i,202,211, 7,'h'); k=93; l=99;
    for (j=194; j<197; j++)
      { lre ( k, l, j,10,'v'); k++; l--;   }
    pos ( 2,21);   printf ("Altitud");
    ax8 ('m', 34,170,14); ax8 ('s', 34,192,14); ax8 ('n', 34,200,14);
    ax8 ('m', 34,208,14); pos ( 6,19); printf ("%6d",0);
    pos (12,23); printf ("%5d",0); pos (13,23); printf ("m snt");
    pos (15,24); printf ("%2.1f",0.0); pos (16,22); printf ("P.Baro");
    /* - - - - - velocidad aérea - - - - - - */
    mar ( 32, 48, 45,123,2,7); mar (147, 48,170,123,2,7);
    mar ( 78, 62, 97, 97,2,5); cua (174, 38,193,121, 7);
```

```
  cua ( 34, 50, 43,121,1); cua (149, 50,168,121,10);
  lre ( 46,146,114, 7,'v'); lre ( 96, 80,102,10,'h');
  for (i=50; i<=140; i+=10) lre ( 1,104,113, 7,'h'); k = 93; l = 99;
  for (j=98; j<102; j++)
    { lre ( k, l, j,10,'v'); k++; l--;   }
  mar (195, 70,225,100,2,7); cua (207, 73,215, 78,7);
  cua (207, 79,209, 81,7); lre (210, 79, 92, 7,'h');
  lin (210,96,210,98,10); lin (214,96,215,98,10);
  lin (217,94,218,96,10); lin (220,92,222,94,10);
  pos ( 2, 7); printf ("Velocidad");
  ax8 ('k', 35, 66,14); ax8 ('m', 35, 76,14); ax8 ('/', 35, 86,14);
  ax8 ('h', 35, 96,14); ax8 ('r', 35,106,14); ax8 ('v',155, 57,14);
  ax8 ('n',155, 67,14); ax8 ('e',155, 77,14); ax8 ('2',155, 90,14);
  ax8 ('7',155,100,14); ax8 ('8',155,110,14);
  pos ( 6, 9); printf ("%4d",0); pos (12, 6); printf ("%2d",0);
  pos (12, 9); printf ("xAletas");
  /* - - - - - - velocidad vertical - - - - - - */
  mar (256, 70,270,153, 2, 7); mar (371, 70,385,153, 2, 7);
  cua (258, 72,268,151,1); cua (373, 72,383,151,10);
  mar (302, 70,321,121, 2, 5); lin (271,140,370,140, 7);
  lin (320, 99,320,129,10); j = 323; k = 126;
  for (i=317; i<=319; i++)
    { lin ( i, k, j, k,10); j--; k++;   }
  for (i=275; i<=365; i+=10) lre ( i,130,139, 7,'h');
  pos (16,10); printf ("Asc./Desc.");
  ax8 ('m',260,102,14); ax8 ('/',260,112,14); ax8 ('s',260,120,14);
  pos (20,10); printf ("%6d",0);
 } /* Fin de la función aa.c */
```

```
/* ra.c
   Graficación del instrumento de dirección de la nave,
   brújula, instrumento de navegación,
   programó: Alfonso León,                Marzo / 1990
   revisión: Alfonso León,               Febrero / 1991     */
ra ()
 { bru (370,331,300,331,290,331,7,10); mar (302,318,321,345,2,7);
   lre (335,326,334,4,'h'); lre (345,315,325,4,'h');
   lre (345,335,344,4,'h'); lre (399,295,325,4,'h');
   lre (399,335,365,4,'h'); lre (410,320,340,4,'h');
   lre (336,338,325,4,'v'); lre (336,338,335,4,'v');
   lre (345,385,325,4,'v'); lin (338,325,342,315,4);
   lin (338,335,342,344,4); lin (385,325,393,295,4);
   lin (385,335,393,365,4); lre (345,385,335,4,'v');
   lre (393,399,295,4,'v'); lre (393,399,365,4,'v');
   lre (399,405,325,4,'v'); lre (399,405,335,4,'v');
   lre (342,345,315,4,'v'); lre (342,345,344,4,'v');
   arc (405,330,5,2,4); pos (18,39); printf ("Brujula");
   pos (20,41); printf ("%3d",0);
 } /* fin de la función ra.c */
```

```
/* bru.c
   Brújula del tablero en la pantalla,
   programó: Alfonso León, Septiembre / 1990 */
bru (xc, yc, xi, yi, xe, ye, clr1, clr2)
float s[35], c[35];
int xc, yc, xi, yi, xe, ye, clr1, clr2;
  { double j;
    int xin, yin, xex, yex, i, k=0; xin = xi; yin = yi; xex = xe; yex = ye;
    for (i=0; i<=35; ++i)
      { j = i*pi/18; s[i] = sin ((double) j); c[i] = cos ((double) j);
        xin = xc + (xi-xc) * c[i] - (yi-yc) * s[i];
        yin = yc + (yi-yc) * c[i] + (xi-xc) * s[i];
        xex = xc + (xe-xc) * c[i] - (ye-yc) * s[i];
        yex = yc + (ye-yc) * c[i] + (xe-xc) * s[i];
        if (i == k)
          { if (i == 0)
              { lin (xin, yin, xex, yex, clr2);
                cir ((xin+xex)/2,(yin+yex)/2,3,clr2);    }
            else { lin (xin, yin, xex, yex, clr2);
                   xin++; yin++; xex++; yex++;
                   lin (xin, yin, xex, yex, clr2);
                   xin--; yin--; xex--; yex--;
                   lin (xin, yin, xex, yex, clr2);    }
            k += 9; if (k > 27) k = 0;    }
        else lin (xin, yin, xex, yex, clr1);    }
  } /* Fin de la función bru.c */
```

```
/* te.c
   Graficación de los instrumentos de: RPM, temperaturas
   cabeza de motor, gases de escape, gases en el carburador,
   medio ambiente, foco del motor encendido, potencia,
   voltímetro, instrumentos de motor,
   programó: Alfonso León,                Marzo / 1990
   revisión; Alfonso León,               Febrero / 1991          */
te ()
  { int i, k, l;
    /* - - - - - - Rev. Por Min. - - - - - - */
    cua ( 78,550, 97,601,7); cua ( 78,606, 97,633,7);
    pos ( 6,77); printf ("RPM"); pos ( 6,70); printf ("%6d",0);
    /* - Temperaturas Cabeza del Cilindro - */
    cua (110,550,129,633,7); cua (113,587,114,595,0);
    cua (115,590,125,591,0); cua (117,600,127,622,0);
    lre (115,608,614, 0,'h'); lre (112,613,615, 0,'h');
    lre (119,604,619,14,'h'); lre (121,605,618,14,'h');
    lre (123,606,617,14,'h'); lre (125,607,616,14,'h');
    pto (113,612,0); pto (114,611,0); pto (116,608,0);
    pto (116,614,0); pto (126,600,7); pto (127,600,7);
    pto (127,601,7); pto (126,622,7); pto (127,621,7);
    pto (127,622,7); pos ( 8,70); printf ("%4d",0);
    /* - - - Gas de Escape - - - */
    cua (142,550,161,633,7); cua (145,587,146,595,0);
    cua (147,590,159,591,0); cua (147,600,156,622,0);
    lre (147,148,601, 7,'v'); lre (147,150,600, 7,'v');
```

```
  lre (155,156,600, 7,'v'); cua (151,598,154,599,0);
  cua (151,623,153,626,0); lre (152,154,627, 0,'v');
  lin (153,628,155,628,0); lre (154,156,629, 0,'v');
  k = 575; l = 575;
  for (i=210; i<=205; i++)
    { lin ( i, k, i, l,10); k--; l++;   }
  pos (10,70); printf ("%4d",0);
  /* - - - Gases en el carburador  - - - */
  cua (174,550,193,633,7); cua (176,612,178,614,0);
  cua (177,586,178,595,0); cua (179,590,191,591,0);
  cua (176,603,191,604,0); cua (176,621,191,622,0);
  lre (178,189,605, 0,'v'); lre (178,189,620, 0,'v');
  lin (180,617,186,608,0); cir (183,612,3,0);
  pos (12,70); printf ("%4d",0);
  /* - - - Fuente de Potencia  - - - */
  cua (206,550,225,633,7); cua (211,612,212,614,0);
  cua (208,586,209,595,0); cua (210,586,223,587,0);
  cua (215,587,216,591,0); cua (208,597,223,598,0);
  cua (208,599,209,605,0); cua (209,605,215,606,0);
  cua (215,599,216,605,0); cua (211,625,212,627,0);
  cua (213,610,221,629,0); lre (222,612,627, 0,'h');
  lre (215,611,615, 7,'h'); lre (215,624,628, 7,'h');
  pto (214,613,7); pto (216,613,7); pos (14,70);
  printf ("%4d",0);
  /* - - - % de Potencia - - - */
  cua (238,550,269,633, 7); cua (255,525,269,549, 7);
  cua (255,552,269,549, 0); cua (257,527,267,600, 0);
  cua (257,601,267,620,14); cua (257,621,267,631, 4);
  cua (261,627,266,631, 3);
  pos (16,75); printf ("% Pot"); pos (16,70); printf ("%4d",0);
  pos (18,66); printf ("100      30  R");
  /* - - - Tem Med Amb - - - */
  cua (396, 76,419,147,7); cua (398, 78,417,105, 1);
  cua (398,110,417,145, 1);
  pos (26,11); printf ("TMA"); pos (26,15); printf ("%4d",0);
  /* - - - - - - Foco de Encendido - - - - - */
  for (i=1; i<=10; i++) cir (40,590, i,10);
  pos ( 1,73); printf ("FME");
 } /* fin de la función te.c */
```

```
/* acr3.c
   Actualización del instrumental en la pantalla
   programó: Alfonso León,          Mayo / 1990      */
#define PI 3.1415926536
#define DCS 200
#define C   50
#define MC -50
#define DAT 20
#define NUEVE  9
#define VENVE 29
int cuenta, color, acab, i, j, k, nivel, mant, nave[DCS];
int ulec[NUEVE] = { 4,4,0,4,0,0,0,20,0};
int vaux[VENVE] = { 210, 93, 22,330, 27,325, 32,320, 27,335,
```

```
                    32,340,130,280,130,335,130,335,130,380,
                   109,330,125,330,130,300,331,290,331};
double jj;  float  s, c;  char   um = 'n';
acr3 ()
  { struct
      {
        int   alt,vel,ad,acab,aala,ona,ale,nc,rpm,tac,tcc,tge,
              vol,pot,tma;
        float pb;
        char  m;
      } at;
    /* - - - -  M S N M - - - - */
    pos (29,5); printf("La altura sobre el nivel del mar es de... ");
    scanf ("%d",&nivel);
    while (fread (&at,sizeof(at),1,dete) == 1)
      { /* - - - - Altitud - - - - */
        pos (6,19); printf("%6d",at.alt);
        mant = at.alt - nivel;
        pos (12,23); printf ("%5d",mant);
        rac ('a', at.alt);
        ulec[0] = alh (at.alt,ulec[0],46,146,202,211,7,'a');
        pos (15,24); printf("%.1f",at.pb);
        /* - - - - Velocidad - - - - */
        pos ( 6, 9); printf("%4d",at.vel); rac ('v', at.vel);
        ulec[1] = alh (at.vel,ulec[1],46,146,104,113,7,'v');
        pos (12, 6); printf("%2d",at.ale);
        if (at.ale != ulec[2])
          { lin (210,79,vaux[0],vaux[1],0);
            switch (at.ale) {
              case  0: lin (210,79,(vaux[0]-210),(vaux[1]-93),7); break;
              case 15: lin (210,79,(vaux[0]-214),(vaux[1]-93),7); break;
              case 30: lin (210,79,(vaux[0]-216),(vaux[1]-92),7); break;
              case 45: lin (210,79,(vaux[0]-219),(vaux[1]-90),7); break;
            } ulec[2] = at.ale;   }
        /* - - - - Vel. Vertical - - - - */
        pos (20,10); printf("%6d",at.ad); rac ('d', at.ad);
        if (at.ad < 0)
          at.ad *= (-1);
        ulec[3] = alh (at.ad,ulec[3],271,370,130,139,7,'d');
        /* - - - - Horizonte Artificial - - - - */
        acab = at.acab * 2;
        if (acab != ulec[4] | at.aala != ulec[5])
          { ulec[5] = at.aala; cuenta = 0;
            cuenta =dlc(vaux[ 2],vaux[ 3],vaux[ 6],vaux[ 7], nave, cuenta);
            cuenta =dlc(vaux[ 2],vaux[ 3],vaux[10],vaux[11], nave, cuenta);
            cuenta =dlc(vaux[ 4],vaux[ 5],vaux[ 8],vaux[ 9], nave, cuenta);
            cuenta =dlc(vaux[ 6],vaux[ 7],vaux[10],vaux[11], nave, cuenta);
            cuenta =dlc(vaux[12],vaux[13],vaux[14],vaux[15], nave, cuenta);
            cuenta =dlc(vaux[16],vaux[17],vaux[18],vaux[19], nave, cuenta);
            cuenta =dlc(vaux[20],vaux[21],vaux[22],vaux[23], nave, cuenta);
            if (acab != ulec[4])
              { ulec[4] = acab;
                cuenta = dcc (vaux[24],330, 5, nave, cuenta);
                vaux[24] = 130 - acab;
                cuenta = gcc (vaux[24],330, 5, nave, cuenta, 14);   }
```

```
        jj = at.aala * PI / 180;
        s = sin ((double) jj); c = cos ((double) jj);
        vaux[ 2] = 130 + ( 22 - 130) * c;
        vaux[ 3] = 330 + ( 22 - 130) * s;
        vaux[ 4] = 130 + ( 27 - 130) * c - (325 - 330) * s;
        vaux[ 5] = 330 + (325 - 330) * c + ( 27 - 130) * s;
        vaux[ 6] = 130 + ( 32 - 130) * c - (320 - 330) * s;
        vaux[ 7] = 330 + (320 - 330) * c + ( 32 - 130) * s;
        vaux[ 8] = 130 + ( 27 - 130) * c - (335 - 330) * s;
        vaux[ 9] = 330 + (335 - 330) * c + ( 27 - 130) * s;
        vaux[10] = 130 + ( 32 - 130) * c - (340 - 330) * s;
        vaux[11] = 330 + (340 - 330) * c + ( 32 - 130) * s;
        vaux[12] = vaux[24] - (MC) * s;
        vaux[13] = 330 + (MC) * c;
        vaux[14] = vaux[24] - (-5) * s;
        vaux[15] = 330 + (-5) * c;
        vaux[16] = vaux[24] - (5) * s;
        vaux[17] = 330 + (5) * c;
        vaux[18] = vaux[24] - (C) * s;
        vaux[19] = 330 + (C) * c;
        vaux[20] = vaux[24] + (109 - acab - vaux[24] ) * c;
        vaux[21] = 330 + (109 - acab - vaux[24]) * s;
        vaux[22] = vaux[24] + (125 - acab - vaux[24]) * c;
        vaux[23] = 330 + (125 - acab - vaux[24]) * s;
        cuenta = 0;
  cuenta = gce (vaux[ 2],vaux[ 3],vaux[ 6],vaux[ 7], nave, cuenta, 14);
  cuenta = gce (vaux[ 2],vaux[ 3],vaux[10],vaux[11], nave, cuenta, 14);
  cuenta = gce (vaux[ 4],vaux[ 5],vaux[ 8],vaux[ 9], nave, cuenta, 14);
  cuenta = gce (vaux[ 6],vaux[ 7],vaux[10],vaux[11], nave, cuenta, 14);
  cuenta = gce (vaux[12],vaux[13],vaux[14],vaux[15], nave, cuenta, 14);
  cuenta = gce (vaux[16],vaux[17],vaux[18],vaux[19], nave, cuenta, 14);
  cuenta = gce (vaux[20],vaux[21],vaux[22],vaux[23], nave, cuenta, 14);
} /* - - - - Brújula - - - - */
    if (at.ona != ulec[6])
      { bru (370,331,vaux[25],vaux[26],vaux[27],vaux[28],0,0);
        jj = at.ona * PI/180;
        s = sin ((double)  jj); c = cos ((double)  jj);
        vaux[25] = 370 + (-70) * c; vaux[26] = 331 + (-70) * s;
        vaux[27] = 370 + (-80) * c; vaux[28] = 331 + (-80) * s;
        bru (370,331,vaux[25],vaux[26],vaux[27],vaux[28],7,10);
        ulec[6] =  at.ona;    }
    pos (20,41); printf ("%3d",at.ona);
    /* - - - - Combustible  - - - - */
    if (at.nc != ulec[7]) ulec[7] = gya (at.nc, ulec[7]);
    /* - - - RPM - - - */
    pos ( 6,70); printf ("%6d",at.rpm);
    /* - - - Aire carburador - - - */
    pos (12,70); printf ("%4d",at.tac);
    /* - - - Temp. en la Cabeza del Cilindros - - - */
    pos (8,70); printf ("%4d",at.tcc);
    /* - - - Temp. en la salida de los gases de escape - - - */
    pos (10,70); printf ("%4d",at.tge);
    /* - - - - Foco de encendido - - - - */
    if (at.m != um)
      { switch (at.m)
```

```
            { case 's': color = 10; break;
              case 'p': color = 14; break;
              case 'n': color = 4; son ( 5); break;   }
          for (j=1; j<=10; j++)
            cir ( 40,590, j, color);
          um = at.m;   }
        /* - - - - Voltimetro - - - - */
        pos (14,70); printf ("%4d",at.vol);
        /* - - - - % de Potencia - - - - */
        pos (16,70); printf ("%4d",at.pot);
        if (at.pot != ulec[8])
          ulec[8] = pna (at.pot,ulec[8]);
        /* - - - - TMA - - - - */
        pos (26,15); printf("%4d", at.tma);    }
  } /* Fin de la función acr3.c */
```

```
/* rac.c
   Alarmas para los instrumentos de aceleracion, velocidad,
   y velocidad verical,
   programó: Alfonso León,   Febrero / 1991       */
rac (ins, lec)
char ins;
int lec;
  { int ca, cv, cd;
    ca = cv = cd = 10;
    if (ins == 'a')
      { if (lec <= 100) ca = 4;
        if (lec > 100 & lec <= 300) ca = 14;
        if (lec > 300) ca = 10;
        switch (ca)
          { case ( 4):
              cua (153,166,157,209,14); son (3);
              cua (149,166,158,219,ca); break;
            case (14):
              cua (153,166,157,209,10); son (1);
              cua (149,166,158,219,ca); break;
            case (10):
              cua (153,166,157,219,ca); break;   }   }
    else if (ins == 'v')
      { if  (lec >= 250 | lec <= -110) cv = 4;
        else if ((lec < 250 & lec >= 170) | (lec < -95 & lec > -110))
          cv = 14;
        else if  (lec < 170 | lec >  -95) cv = 10;
        switch (cv)
          { case ( 4):
              cua (153, 50,157,121,14); son (3);
              cua (149, 50,168,121,cv); break;
            case (14):
              cua (153, 50,157,121,10); son ( 1);
              cua (149, 50,168,121,cv); break;
            case (10):
              cua (149, 50,168,121,cv); break;   }   }
    else if (ins == 'd')
```

```
      { if  (lec > 999 | lec < -999) cd = 4;
        else if ((lec < 999 & lec >= 500) | (lec < -500 & lec > -999))
          cd = 14;
        else if  (lec < 500 | lec >  -500) cd = 10;
        switch (cd)
          { case ( 4):
              cua (377, 72,381,151,14); son (3);
              cua (373, 72,383,151,cd); break;
            case (14):
              cua (377, 72,381,151,10); son ( 1);
              cua (373, 72,383,151,cd); break;
            case (10):
              cua (373, 72,383,151,cd); break;   }   }
  } /* fin de la función rac.c */
/* alh.c
   Animación de las líneas horizontales para la altura,
   velocidad aérea y la velocidad vertical,
   programó: Alfonso León,          Noviembre / 1990      */
alh (lec,ulec,xi,xf,yi,yf,color,alar)
int lec,ulec,xi,xf,yi,yf,color;
char alar;
  { int i, j = lec+50;
    lec = red (lec);
    if (lec != ulec)
      for (i=xi; i<xf; i+=10)
        { lre (i+ulec, yi, yf, 0,'h');
          switch (alar)
            { case 'a': if (j < 0) color = 0;
                        if (j >= 0 & j <= 100) color = 4;
                        if (j >= 100 & j <= 300) color = 14;
                        if (j > 300) color = 7;   break;
              case 'v': if (j > 250 | j < -110)
                          { color = 4; break;   }
                        if ((j <= 250 & j > 170) | (j < -95 & j >= -110))
                          { color = 14; break;   }
                        if (j <= 170 | j >= -95)
                          { color = 7; break;   }
              case 'd': if (j > 999 | j < -999)
                          { color = 4; } break;   }
                        if ((j > 500 & j <= 999) | ( j >= -999 & j < -500))
                          { color = 14; break;   }
                        if (j <= 500 | j >= -500)
                          { color = 7; break;   } break;   }
          lre (i+lec ,yi, yf, color,'h'); j -= 10;   }
    return (lec);
  } /* Fin de la función alh.c */
red (xxx)
int xxx;
  { while (xxx >= 1000) xxx -= 1000;
    while (xxx >= 100) xxx -= 100;
    while (xxx >= 10) xxx  -= 10;
    return (xxx);
  } /* Fin de la sub-función red () */
```

```
/* dlc.c
   Dibuja una línea con los colores encontrados anteriormente,
   programó: Alfonso León,      Febrero / 1991              */
dlc (ix, iy, x, y, avion, cont)
int ix, iy, x, y, avion[], cont;
  { int sx, sy, dx, dy, qq, er;
    dx = x-ix; dy = y-iy; sx = sy = 1;
    if (dx < 0)
      { sx = -1; dx = -dx;   }
    if (dy < 0)
      { sy = -1; dy = -dy;   }
    if (dy >= dx)
      { er = 2 * dx - dy;
        for (qq=0; qq<=dy; qq++)
         { pto (ix, iy, avion[cont]); cont++;
           if (er >= 0)
             { er += 2 * dx - 2 * dy; iy += sy; ix += sx;   }
           else
             { er += 2 * dx; iy += sy;   }   }   }
    else
      { er = 2 * dy - dx;
        for (qq=0; qq<=dx; qq++)
          { pto (ix, iy, avion[cont]); cont++;
            if (er >= 0)
              { iy += sy; ix += sx; er += 2 * dy - 2 * dx;   }
            else
              { er += 2 * dy; ix += sx;   }   }   }
    if ((ix > x) | (iy > y))
      pto (ix, iy, avion[cont]);
    return (cont);
  } /* Fin de la función dlc.c */
```

```
/* bca.c
   Busca, un Color en la coordenada actual (x, y),
   programó: Alfonso León,          Febrero / 1991 */
bca (xx, yy)
int xx, yy;
  { rega.x.dx = xx;
    rega.x.cx = yy;
    rega.h.ah = 0x0d;
    int86 (MODOGRAF, &rega, &rega);
    return (rega.h.al);
  }  /* Fin de la función bca.c */
```

```
/* dcc.c
   Dibuja un círculo con los colores encontrados;
   programó: Alfonso León,          Febrero / 1991 */
dcc (xc, yc, radio, nave, cu)
int xc, yc, radio, nave[], cu;
  { int p, x, y;
    x = 0; y = radio; p = 3 - 2 * radio;
```

```
    do { pop (x, y, nave, xc, yc);
         if (p < 0) p += 4 * x + 16;
         else { p += 4 * (x - y) + 10; y--;   } x++;
         if (x == y) pop (x, y, nave, xc, yc);
        } while (x < y);
     return (cu);
  } /* Fin de la función dcc.c */
pop (xa, ya, nave, xb, yb)
int xa, ya, nave[], xb, yb;
  {pto (xb+xa, yb+ya, nave[cu]); cu++; pto (xb-xa, yb+ya, nave[cu]); cu++;
   pto (xb+xa, yb-ya, nave[cu]); cu++; pto (xb-xa, yb-ya, nave[cu]); cu++;
   pto (xb+ya, yb+xa, nave[cu]); cu++; pto (xb-ya, yb+xa, nave[cu]); cu++;
   pto (xb+ya, yb-xa, nave[cu]); cu++; pto (xb-ya, yb-xa, nave[cu]); cu++;
  } /* Fin de la sub-función pop () */
```

```
/* gcc.c
   Guarda los colores del círculo previo a dibujar uno nuevo;
   programó: Alfonso León,                  Febrero / 1991      */
gcc (xc, yc, radio, nave, cc, color)
int xc, yc, radio, nave[], cc, color;
  { int p, x, y;
    x = 0; y = radio; p = 3 - 2 * radio;
    do { pp ( x, y, nave, xc, yc, color);
         if (p < 0) p += 4 * x + 16;
         else { p += 4 * (x - y) + 10; y--;   } x++;
         if (x == y) pp ( x, y, nave, xc, yc, color);
        } while (x < y);
     return (cc);
  } /* Fin de la función dcc.c */
pp (xa, ya, nave, xb, yb, color)
int xa, ya, nave[], xb, yb, color;
  { nave [cc++] = bca (xb+xa, yb+ya); pto (xb+xa, yb+ya, color);
    nave [cc++] = bca (xb-xa, yb+ya); pto (xb-xa, yb+ya, color);
    nave [cc++] = bca (xb+xa, yb-ya); pto (xb+xa, yb-ya, color);
    nave [cc++] = bca (xb-xa, yb-ya); pto (xb-xa, yb-ya, color);
    nave [cc++] = bca (xb+ya, yb+xa); pto (xb+ya, yb+xa, color);
    nave [cc++] = bca (xb-ya, yb+xa); pto (xb-ya, yb+xa, color);
    nave [cc++] = bca (xb+ya, yb-xa); pto (xb+ya, yb-xa, color);
    nave [cc++] = bca (xb-ya, yb-xa); pto (xb-ya, yb-xa, color);
  } /* Fin de la sub-función pp () */
```

```
/* gce.c
   Guarda los colores encontrados en las coordenadas (X,Y),
   programó: Alfonso León,                  Febrero / 1991      */
gce (xi, yi, xf, yf, nav, cta, color)
int xi, yi, xf, yf, nav[], cta, color;
  { int negx, negy, disx, disy, aj, i;
    disx = xf - xi; disy = yf - yi; negx = negy = 1;
    if (disx < 0)
      { negx = -1; disx = -disx;   }
    if (disy < 0)
```

```
    { negy = -1; disy = -disy;   }
  if (disy >= disx)
    { aj = 2 * disx - disy;
      for (i=0; i<=disy; i++)
       { nav[cta] = bca (xi, yi); cta++; pto (xi, yi, color);
         if (aj >= 0)
           { aj += 2 * disx - 2 * disy; yi += negy; xi += negx;   }
         else
           { aj += 2 * disx; yi += negy;   }   }   }
  else  { aj = 2 * disy - disx;
      for (i=0; i<=disx; i++)
        { nav[cta] = bca (xi, yi); cta++; pto (xi, yi, color);
          if (aj >= 0)
            { yi += negy; xi += negx; aj += 2 * disy - 2 * disx;   }
          else { aj += 2 * disy; xi += negx;   }   }   }
  if ((xi > xf) | (yi > yf))
    { nav[cta] = bca (xi, yi); pto (xi, yi, color);   }
  return (cta);
 } /* Fin de la función gce.c */
```

```
/* gya.c
   Actualización del nivel de combustible,
   programó: Alfonso León.    Julio / 1990 */
#define LIMGYA 4
#define LIMGYAR 2
gya (nc, unc)
int nc, unc;
  { ilim = 46 + 9 * (20 - nc) + 12;
    for (j=46; j<=ilim; j++)  lre ( j,450,459, 0,'h');
    for (j=58; j<=ilim; j+=9)  lre ( j,452,458,14,'h');
    for (j=59; j<=ilim; j+=45)  lre ( j,452,458,14,'h');
    if (nc <= LIMGYA)  cua (244,433,253,479,14);
    if (nc <= LIMGYAR)
      { for (j=0; j != 10; j++)
          { printf ("\7"); cua (244,433,253,479,14);
            for (k=1; k<500; k++)  ; cua (244,433,253,479,4);   }
      }
    return (nc);
  } /* Fin de la función gya.c */
```

```
/* normal.c
   devuelve el modo texto a la pantalla,
   programó: Alfonso León, Febrero / 1990 */
main ()
  { char ch;
    regs.x.ax = 0x03;
    int86(0x10, &regs, &regs);
  } /* fin de la función normal.c */
```

```
/* acr4.c
   Lectura y despliege de datos para la simulación de una misión,
   creación del archivo el cual contiene información para los instrumentos,
   programó: Alfonso León,        Noviembre / 1990     */
char re;
int  mant, i, nivel;
main (argc,argv)
int argc;
char *argv[];
  { struct
      { int   alt,vel,ad,acab,aala,ona,pa,nc,rpm,tac,tcc,tge,vol,pot,tma;
        float pb; char  m;
      } at;
    FILE *deta;     FILE *dete;
    if ((deta = fopen (argv[1],"ab")) == NULL)
      { pos (25,9);
        printf("No se pudo abrir el archivo %s", argv[1]);
        exit (1);    }
    pos ( 2, 7);
    printf("LECTURA DE DATOS PARA LA ACR Aeronave de Control Remoto");
    pos ( 3, 8); printf("-Altitud que lleva... ");
    pos ( 4, 8); printf("-Velocidad alcanzada... ");
    pos ( 5, 8); printf("-Velocidad de Ascenso / Descenso... ");
    pos ( 6, 8); printf("-Angulo de cabeceo... ");
    pos ( 7, 8); printf("-Angulo de alabeo... ");
    pos ( 8, 8); printf("-Orientacion de la nariz... ");
    pos ( 9, 8); printf("-Posicion de las aletas (0 a 45 gds.)... ");
    pos (10, 8); printf("-Nivel del combustible... ");
    pos (11, 8); printf("-Cuantas RPM lleva el motor... ");
    pos (12, 8); printf("-Temperatura del aire al carburador... ");
    pos (13, 8); printf("-Temperatura de la cabeza de los cilindros... ");
    pos (14, 8); printf("-Temperatura de los Gases de escape... ");
    pos (15, 8); printf("-El motor sigue encendido (s/n)... ");
    pos (16, 8); printf("-Voltimetro... ");
    pos (17, 8); printf("-Potencia en el motor... ");
    pos (18, 8); printf("-Temperatura medio ambiente... ");
    pos (19, 8); printf("-Presion Barometrica... ");
    pos (26, 9); printf("RTRN Continua");
    pos (26,33); printf("ESC Salida");
    do  { borra (); pos ( 3,60); printf(" "); scanf("%d",&at.alt);
          pos ( 4,60); printf(" "); scanf("%d",&at.vel);
          pos ( 5,60); printf(" "); scanf("%d",&at.ad);
          pos ( 6,60); printf(" "); scanf("%d",&at.acab);
          pos ( 7,60); printf(" "); scanf("%d",&at.aala);
          pos ( 8,60); printf(" "); scanf("%d",&at.ona);
          pos ( 9,60); printf(" "); scanf("%d",&at.pa);
          pos (10,60); printf(" "); scanf("%d",&at.nc);
          pos (11,60); printf(" "); scanf("%d",&at.rpm);
          pos (12,60); printf(" "); scanf("%d",&at.tac);
          pos (13,60); printf(" "); scanf("%d",&at.tcc);
          pos (14,60); printf(" "); scanf("%d",&at.tge);
          pos (15,60); printf(" "); scanf("%s",&at.m);
          pos (16,60); printf(" "); scanf("%d",&at.vol);
          pos (17,60); printf(" "); scanf("%d",&at.pot);
          pos (18,60); printf(" "); scanf("%d",&at.tma);
```

```
        pos (19,60); printf(" "); scanf("%f",&at.pb);
        fflush (stdin); fwrite (&at, sizeof(at), 1, deta);
        pos (26, 9); printf("RTRN Continua"); pos (26,33);
        printf("ESC Salida");
    } while ((re = getch()) == 13);
  fclose (deta); limpia (0); pos (29,10);
  printf ("QUIERES EL DESPLIEGE DE LOS DATOS ESCRITOS [s/n]");
  scanf ("%c",&re);
  if (re == 's')
    { pos (29,10);
      printf ("La altura sobre el nivel del mar es... ");
      scanf ("%d",&nivel);
        if ((dete = fopen(argv[1],"rb")) == NULL)
        { pos (30,10);
          printf    ("Error no se pudo abrir el archivo %s", argv[1]);
          exit      (2);    }
      pos ( 1,29); printf ("LOS DATOS LEIDOS HASTA AHORA SON...");
      pos ( 3, 1);
      printf ("Altitud Veloc Asc/Desc Bru m.snt Ale Comb");
      pos ( 3,43);
      printf ("RPM TAC TCC TGE VOL POT TMA PBARO MOT");
      i = 5;
      while (fread(&at, sizeof(at),1,dete) == 1)  {
          pos ( i, 1); printf("%6d",at.alt);
          pos ( i, 9); printf("%5d",at.vel);
          pos ( i,16); printf("%6d",at.ad);
          pos ( i,24); printf("%3d",at.ona);
          msnt = at.alt - nivel;
          if (msnt < 0) msnt *= (-1);
          pos ( i,28); printf("%5d",msnt);
          pos ( i,34); printf("%2d",at.pa);
          pos ( i,39); printf("%2d",at.nc);
          pos ( i,43); printf("%3d",at.rpm);
          pos ( i,47); printf("%3d",at.tac);
          pos ( i,51); printf("%3d",at.tcc);
          pos ( i,55); printf("%3d",at.tge);
          pos ( i,59); printf("%3d",at.vol);
          pos ( i,64); printf("%3d",at.pot);
          pos ( i,67); printf("%3d",at.tma);
          pos ( i,71); printf("%.1f",at.pb);
          pos ( i,78); printf("%c",at.m);
          if (i == 25)
            { pos (29,15);
              printf ("RTRN para continuar");   }
          else i++;   }
      pos (29,60); printf ("ESC para salir");
    } fclose (dete);
} /* Fin del programa acr4.c  */
```

```
borra ()
  { int i;
    for (i=3; i<=19; i++)
      { pos (i,61);
        printf("          ");   }
    pos (26, 9); printf("              ");
    pos (26,33); printf("          ");
  }     /* fin de la subfunción borra */
```

```
/* acr5.c
   Despliege de datos para la simulación de una misión,
   programó: Alfonso León,          Noviembre / 1990      */
main ()
  { struct  {
      int   alt,vel,ad,acab,aala,ona,pa,nc,rpm,tac,tcc,tge,vol,pot,tma;
      float pb; char  m;   } at;
    FILE *dete;
    pos (29,10);
    printf ("QUIERES EL DESPLIEGE DE LOS DATOS ESCRITOS [s/n]");
    scanf ("%c",&re);
    if (re == 's')
      { pos (29,10);
        printf ("La altura sobre el nivel del mar es... ");
        scanf ("%d",&nivel);
          if ((dete = fopen(argv[1],"rb")) == NULL)
           { pos (30,10);
            printf   ("Error no se pudo abrir el archivo %s", argv[1]);
            exit     (0);   }
        pos ( 1,29); printf ("LOS DATOS LEIDOS HASTA AHORA SON...");
        pos ( 3, 1); printf ("Altitud Veloc Asc/Desc Bru m.snt Ale Comb");
        pos ( 3,43); printf ("RPM TAC TCC TGE VOL POT TMA PBARO MOT");
        i = 5;
        while (fread(&at, sizeof(at),1,dete) == 1)
          { pos ( i, 1); printf("%6d",at.alt);
            pos ( i, 9); printf("%5d",at.vel);
            pos ( i,16); printf("%6d",at.ad);
            pos ( i,24); printf("%3d",at.ona);
            msnt = at.alt - nivel;
            if (msnt < 0) msnt *= (-1);
            pos ( i,28); printf("%5d",msnt);
            pos ( i,34); printf("%2d",at.pa);
            pos ( i,39); printf("%2d",at.nc);
            pos ( i,43); printf("%3d",at.rpm);
            pos ( i,47); printf("%3d",at.tac);
            pos ( i,51); printf("%3d",at.tcc);
            pos ( i,55); printf("%3d",at.tge);
            pos ( i,59); printf("%3d",at.vol);
            pos ( i,64); printf("%3d",at.pot);
            pos ( i,67); printf("%3d",at.tma);
            pos ( i,71); printf("%.1f",at.pb);
            pos ( i,78); printf("%c",at.m);
            if (i == 25)
              { pos (29,15); printf ("RTRN para continuar"); i = 5;
```

```
            limpia (5);   }
          else i++;
        }
      pos (29,60); printf ("ESC para salir");   }  fclose (dato);
} /* fin del programa acr5.c  */
```

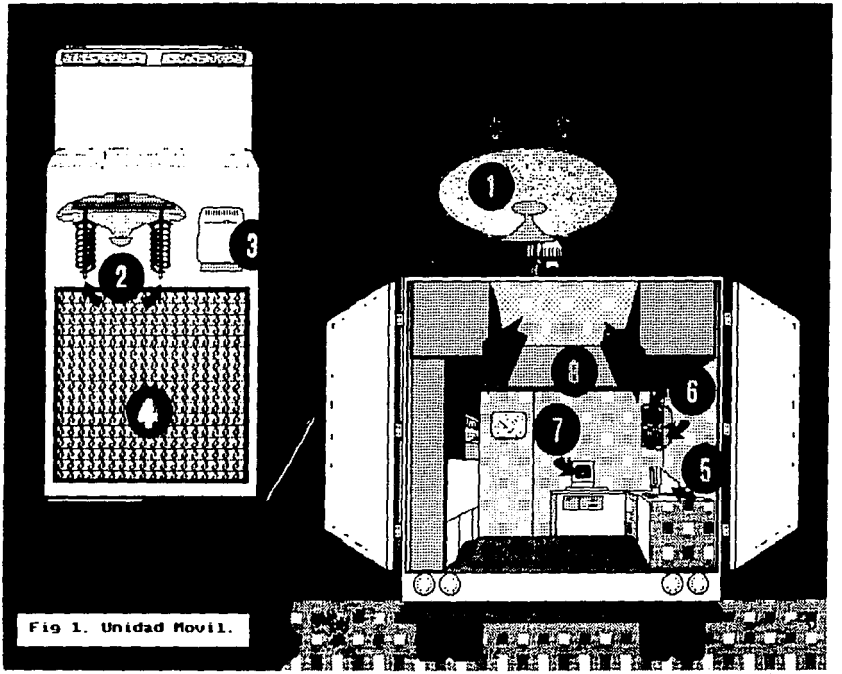


Fig 1. Unidad Movil.

1. Parabólica de dirección movil.
2. Antenas helicoidales, un par, para recepción de datos.
3. Clima artificial para refrescamiento de la estación terrena.
4. Canastilla para transporte de arreos de instrumentos de la aeronave y la estación terrena.
5. Pantalla de despliegue de instrumentos de la aeronave.
6. Monitores de ruta de vuelo y mapa de la zona.
7. Pantalla auxiliar de instrumentos de la aeronave.
8. Gabinetes para almacenaje de cintas de video, cables e implementos extras para diferentes situaciones.

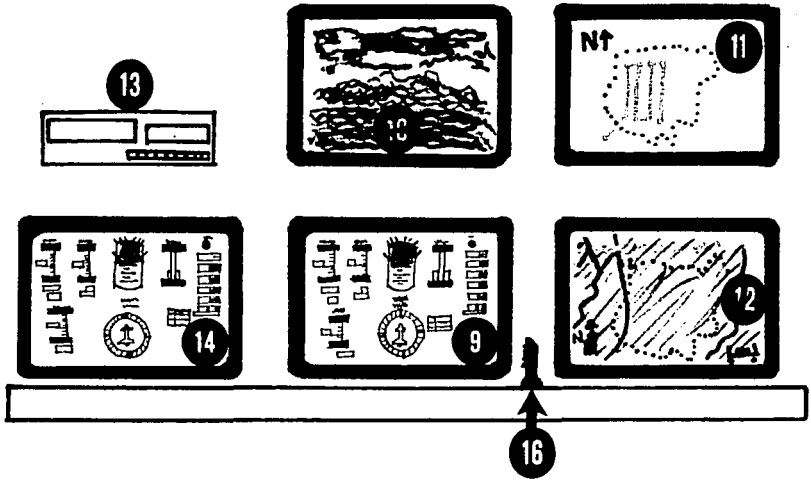


Fig. 2 Tablero de Instrumentos.

9. Monitor principal de los instrumentos de comando de la aeronave.
10. Vista frontal, monitor con la imágen emitida por las cámaras delanteras de la aeronave.
11. Ruta de vuelo que lleva actualmente la aeronave.
12. Mapa del area de trabajo.
13. Video grabadora, hace las veces de caja negra.
14. Monitor auxiliar de instrumentos de comando.
15. Pedales para mover el timon de profundidad.
16. Palanca multiusos para alabeo, cabeceo y derrape.

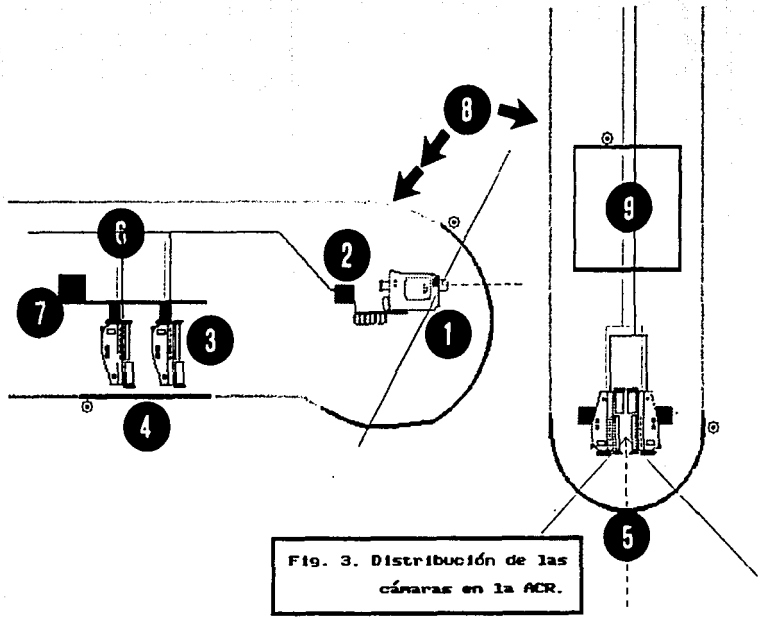


Fig. 3. Distribución de las cámaras en la ACR.

1. Par de cámaras para toma frontal del terreno o zona de vuelo.
2. Caja de electrónica para el movimiento de las cámaras.
3. Sección de cámaras, 3 para video y 1 fotográfica, para muestras del terreno.
4. Ventana para protección de cámaras.
5. Semiesfera para cortar el aire y protección de cámaras.
6. Cables de corriente y comunicación con la electrónica.
7. Caja de electrónica para iniciar tarea con las cámaras.
8. Cuerpo de la aeronave, parte frontal.
9. Caja que guarda el equipo fotográfico, intercambiable.

⊙ Materiales compuestos, carbon epoxi.

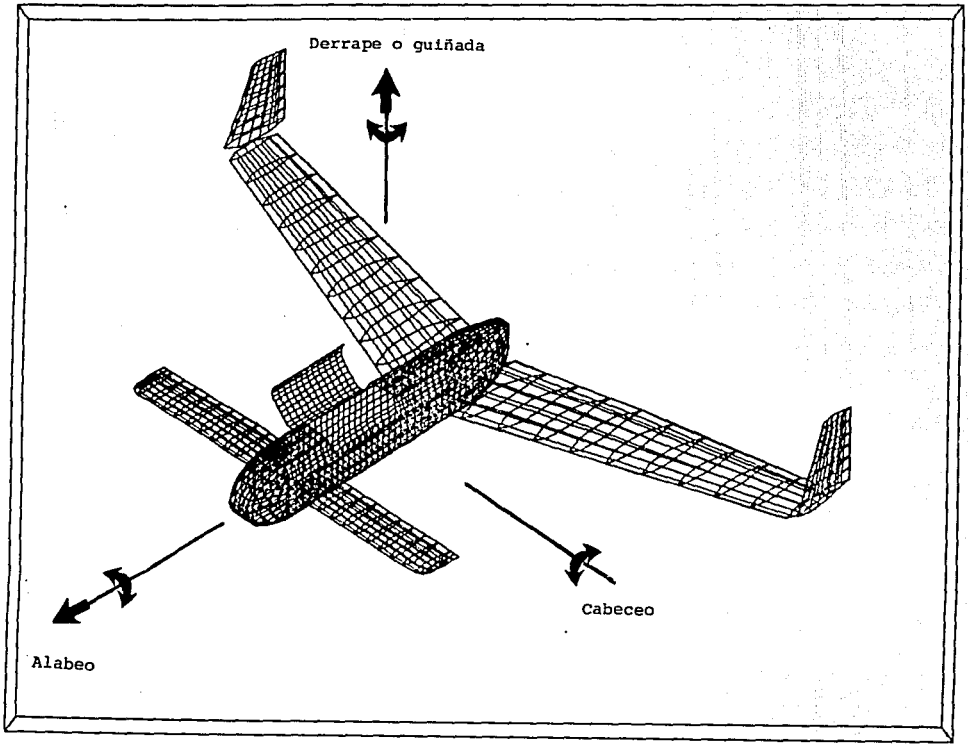


Fig. 4. Dibujo estructural de la aeronave de control remoto, diseñado por el IPN en la ESIME Ticoman.

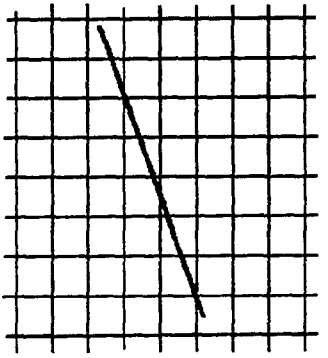


Fig. 5 Trazo de una línea recta.
Al seguir el trazo de una línea recta se efectuan movimientos hazta alcanzar el otro extremo.

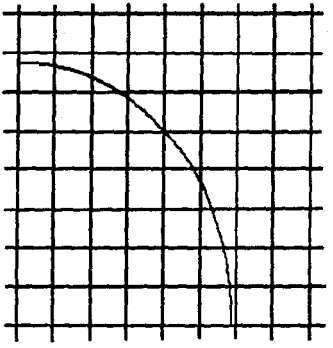


Fig 6. Trazo de una circunferencia.
Al seguir el trazo del circulo se reqiere de movimientos verticales, diagonales y horizontales.

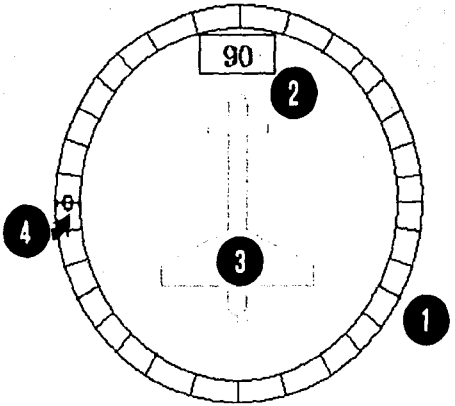


Fig. 7. Brújula de dirección de la ACR.

1. División en grados de la brújula, lectura analógica.
2. Lectura digital de la dirección.
3. Icono principal, figura del modelo real de la aeronave.
4. El símbolo 0 indica el Norte.

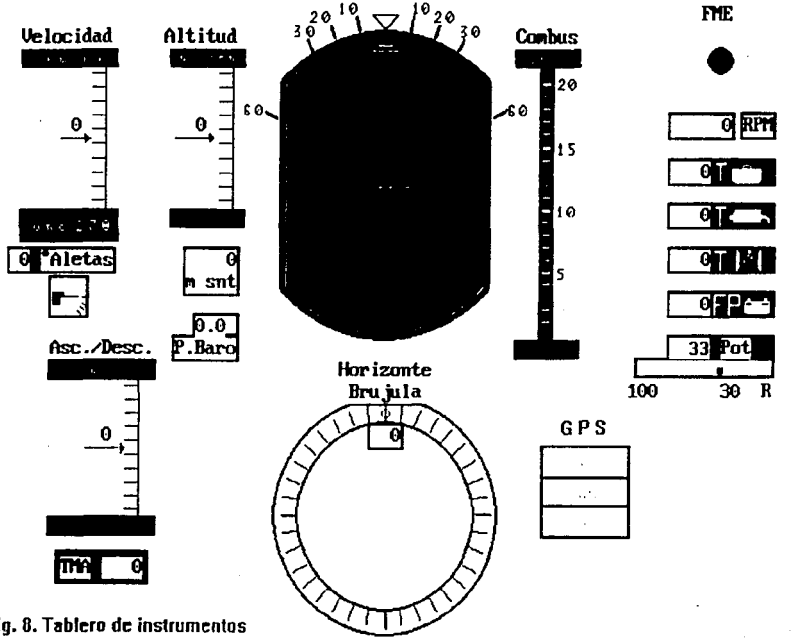


Fig. 8. Tablero de instrumentos de la ACR.

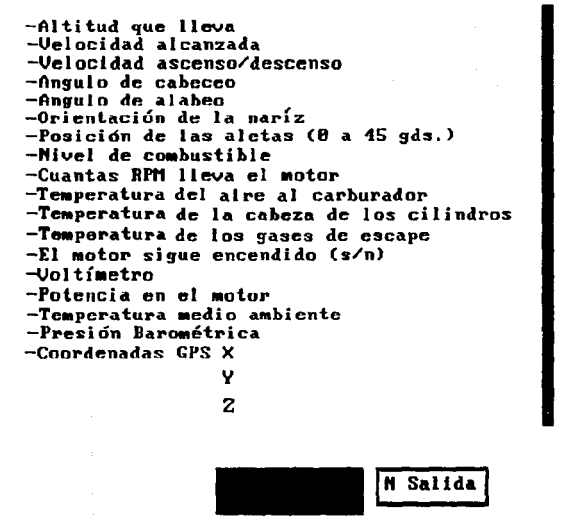


Fig. 9. Petición de datos de telemetría.

BIBLIOGRAFIA.

1. McCORMICK, BARNES W.
Aerodynamics, Aeronautics and Flyght Mechanics.
Wiley & Sons, 1979, p.p. 652.

2. PERALTA RICARDO & VICENTE ESAU.
Procesamiento Digital de Imágenes. Aplicación y Teoría.
Ins. Igeniería, 1986,p.p. 100.

3. LAFORE, ROBERT.
Turbo C. Programming the PC.
SAMS, 1985, p.p. 681.

4. JOHNSON, NELSON.
Advanced Graphics in C. Programming and Techniques.
Osborne McGraw-Hill.
Bekerley, Califoria. 1987, p.p. 670.

5. GOODMAN, S.E. & HEDETNIEMI, S.T.
Introduction to the design and analysis of algorithms.
McGraw-Hill Book Company. 1977, p.p. 369.

6. DARNELL, PETER R. & MARGOLIS, PHILLIP E.
Software Engineering in C.
Spriner-Velag, New York, 1988, p.p. 612.

7. KELLEY, AL & POHL, IRA.
A book on C. An introduction to programming in C.
The Benjamin/Cummings Publishing Company, Inc.
Menlo Park, California. 1984, p.p. 355.

8. KLIEWER, DYCK BRADLEY.
EGA/VGA A programmer's Reference Guide.
McGraw-Hill Book Co.
New York, N.Y. 1988, p.p.269.

9. HEARN, D. & BAKER, M. PAULINE.
Gráficas por Computadora.
Prentice-Hall Hispanoamérica, S. A.,
México, D. F., 1989, p.p. 380.
(Trad. Juan Carlos Vega Fagoaga).

10. SEDGEWICK, ROBERT.
Algorithms.
Addison-Wesley Publishing Co.,
U. S. A. 1983, p.p. 551.

11. JANE'S POCKET BOOK OF REMOTELY PIOTED VEHICLES.
Edit. Jonh W.R. Tayor.
New York, N.Y., 1977, p.p. 235.

12. NIEDERHEITMANN, ALFREDO.
Aviación Apicada con vuelo por instrumentos.
Guatemala, C.A., 1978, 3a. Ed., p.p. 500.

13. TOKHEIM, ROGER L.
Pricipios Digitales.

Series Schaum-Mackgrow-Hill.
1980, México, D. F., 1989. (trad. Raúl Varela G.)

14. SKLAR, BERNARD.
Digital Comunications, Fundamenals y Aplications.
Prentice-Hall, Engewood Cliffs, New Jersey,
1988, p.p. 775.

15. ADAMS, LEE.
High-Performance Graphics in C:
Animation and Simulation.
Winderest Books.
1988, USA. p.p. 518.

16. GRAEF, GERALD L.
Graphics Formats.
Byte Magazine,
September, 1989.

17. REHAN, RICK.
Stroke-Character Graphics.
Byte Magazine,
January, 1990.

18. KLASS, PHILIP J.
Pentagon Urged to Confirm Policy
Allow Civilian Use of GPS Navigation.
Aviation Week & Space Technoogy.
January 8, 1990.

19. RANSEN, OWEN F.
The Art of Ray Tracing.
Byte Magazine.
February, 1990.

20. VOGELGESANG, PETER.
Drowing in Data.
Byte Magazine.
February, 1990.

21. GLASS, BRETT.
The IBM PC BIOS.
Byte Magazine.
April, 1990.

22. KLIEWER, BRADLEY DYCK.
Debunking 16-bit VGA.
Byte Maazine.
June, 1988.

23. AGENCIA FED. DE AVIACION (F.A.A.).
Manual de Aeronautica para el pilotoparticular.
Ed. Diana, México, 1974.

24. PAPPAS & MURRAY.
80386/80286 Programación en Lenguaje Ensamblador.
McGraw-Hill Ed.
1989, México,